SELINUS UNIVERSITY
OF SCIENCES AND LITERATURE

ARTIFICIAL INTELLIGENCE WITH CHARGE TRANSFER IN DEEP BRAIN STIMULATION: NEW APPROACHES BASED ON AI/ML, LLM, GENERATIVE AI, AND CYBERSECURITY IN THE DIAGNOSIS AND TREATMENT OF BRAIN DISEASE

By Agnieszka Maria Mietz-Blijleven

A DISSERTATION

Presented to the Department of
Artificial Intelligence
program at Selinus University

Faculty of Computer Science
in fulfillment of the requirements
for the degree of Doctor of Philosophy
in Artificial Intelligence

2025

Declaration

I hereby confirm that I am the sole author of this dissertation, and its content is solely the result of my readings, study and research.

I declare that this dissertation entitled **“ARTIFICIAL INTELLIGENCE WITH CHARGE TRANSFER IN DEEP BRAIN STIMULATION: NEW APPROACHES BASED ON AI/ML, LLM, GENERATIVE AI, AND CYBERSECURITY IN THE DIAGNOSIS AND TREATMENT OF BRAIN DISEASE”**, is entirely my own work and has not been submitted for any other degree or qualification at this or any other institution.

All references and sources of information in this study used have been appropriately acknowledged.

Agnieszka Maria Mietz-Blijleven
FINAL THESIS - UNISE0873EG

Table of Contents

Abstract ... 8

Introduction ... 12

Chapter 1: Introduction, including the background, problem statement, objectives, research questions, significance, and scope... 15

Background of the Study ... 15

Solution and Architecture Overview ... 16

Problem statement ... 17

Graphene Diodes Embedded in the Brain Cortex for Monitoring Cerebral Blood Flow and Neurological Stimulation ... 20

Data Measurement and Analysis Using AI, LLM, and ML ... 20

Objectives of PhD Thesis ... 23

Research questions ... 24

Brain stroke and migraine with aura - common points and root causes of occurrence... 24

How do brain strokes and migraines with aura occur in infants and seniors, and under what circumstances? ... 25

How can we optimize the treatment of brain strokes in seniors by comparing the pathophysiology of strokes in infants and migraines with aura, utilizing advanced techniques such as charge transfer, deep brain stimulation (DBS), artificial intelligence (AI), machine learning (ML), large language models (LLM), and the Vojta method? ... 27

The significance of PhD thesis ... 29

Scope of PhD Thesis ... 31

1. Integration of Advanced Technologies 31

2. Innovative Use of Graphene Diodes 31

3. Focus on Charge Transfer Transitions 31

4. Personal conclusions on statement of a problem and Observational Insights...... 32

5.AI/ML/LLM mechanism of DBS with graphene diodes with charge transfer 32

6. Development of a Centralized Healthcare System... 33

7. Potential for Broad Applications... 33

Chapter 2: Literature Review, covering existing research on AI, ML, DBS, the Vojta method, charge transfer transitions, graphene applications, and migraine treatment. 34

Chapter 3: Methodology, detailing the research design, data collection, and analysis methods. 39

Statement of the Problem 39

Deep Brain Stimulation with AI/genAI/ML/LLM/Charge transfer/Vojta Method. Graphene diode – nanotechnology - Approach, outlining the framework for research design, techniques for gathering data, and strategies for analysis. 41

NEURAL MASS MODEL with R.. 47

Integration of AI, ML and Advanced Algorithms... 48

Vojta Method Integration 49

Charge Transfer in Deep Brain Stimulation Science and Nanotechnology Diodes. 49

Graphene and Graphene Diodes in Enhancing Deep Brain Stimulation and Neuromodulation Techniques 50

The Mechanism of Charge Transfer in Physics, Including Conduction and Induction, can be related to the Occurrence of Strokes and Migraines with Aura Through Several Key Aspects 60

AI/ML Integration in Electrode Design 61

Neurological Rehabilitation 63

Migraine with aura treatment 63

Recovery of Preterm Infants and Seniors 64

Graphene diodes for DBS 65

The Vojta Method in Nervous System Rehabilitation: The Vojta method is a neurological rehabilitation technique that involves stimulating specific points on the patient's body to evoke proper movement patterns. It is particularly effective in infants and children with central nervous system damage. 66

How Retrieval-Augmented Generation (RAG) Enables LLMs to Utilize The Data Sources Without Training? 77

RAG – data retrieval with DB and cybersecurity 80

Nanotechnology for Non-Invasive Monitoring 88

Chapter 4: Results, presenting the findings of the study. 90

Artificial Intelligence with platform of services – Azure AI, Azure Open AI, Azure Databricks, Azure AI Foundry, Azure Fabric, Machine Learning, Deep Learning of DBS.... 90

Regulatory & Compliance, responsible AI 94

Cybersecurity in Deep Brain Stimulation 96

Application with neural network integrated with Graphene diodes. 98

Optimizing Deep Learning Models Using Azure Databricks 117

Introduction 117

Experimentation and Results 118

Patient Features 118

Deep Learning Model Training 118

Horovod for Distributed Training 118

Conclusion of researches 118

Chapter 5: Discussion, interpreting the results in the context of the research questions and objectives.. 143

Results explanation with Azure AI Foundry 143

Graphene Diodes and Charge Transfer Mechanisms 144

Mechanism with LLM, Generative AI, AI/ML, and MASS Model with R and Julia for Machine Learning 145

Specific Examples of Generative AI Applications in Azure AI Foundry for Deep Brain Stimulation 145

Example 1: Predictive Modelling with Charge Transfer 145

Example 2: Real-time Monitoring and Adjustment 146

Example 3: Personalized Treatment Plans 147

Azure Open AI Services 147

Example 4: Fine-tuning Language Models for Medical Applications 151

Example 5: Responsible Implementation and Evaluation 155

Initiating Charge Transfer through Graphene Diodes and Its Effects on Muscle Tension Relief and Brain Stimulation.......166Chapter 6: Conclusion and Recommendations, summarizing the study and suggesting future research directions. 171

Potential for Advanced Applications 171

Data Collection from Graphene Diodes 183

Neural Networks for Data Processing 183

Integration with AI/ML and Cybersecurity 185

Cybersecurity, observability and monitoring solution for central patient monitoring health application and nanotechnology diodes and future solutions of equipment on multi-platform OS 185

Nexthink Capabilities 185

Integration with Healthcare Applications 185

AI and Charge Transfer in DBS 186

Future Solutions and Multi-Platform Integration 186

How to enrol solutions on multi-platforms and integration with other apps? 197

Regulatory and Ethical Considerations 200

Clinical Validation 201

Recommendations from PhD Thesis: Artificial Intelligence with Charge Transfer in Deep Brain Stimulation 203

Graphene-Based Nasal Electrodes for Neural Stimulation and Diagnostic Monitoring Enhanced by AI and Nanotechnology 205

References 208

List of tables and figures 212

Abstract

I do hereby attest that I am the sole author of this project/thesis, the sole idea maker and that its contents are only the result of the readings and research I have done.

Artificial Intelligence in Healthcare: Deep Brain Stimulation and Data Collection with Graphene Diodes

Artificial Intelligence (AI) is revolutionizing healthcare in numerous ways, enhancing both patient care and the efficiency of medical processes. AI algorithms are being used to analyze medical images such as X-rays, CT scans, and MRIs, detecting abnormalities and diagnosing conditions with high accuracy, sometimes even surpassing human radiologists. This helps in the early detection and treatment of diseases like cancer or brain disorders. AI can analyze vast amounts of patient data to predict disease outbreaks, patient admissions, and even individual health risks, allowing for proactive healthcare measures, improving patient outcomes, and reducing healthcare costs. By analyzing genetic information and other personal health data, AI can help create personalized treatment plans tailored to individual patients, increasing the effectiveness of treatments and minimizing side effects. AI accelerates the drug discovery process by predicting how different compounds will interact with targets in the body, significantly reducing the time and cost involved in bringing new drugs to market. AI-powered robotic systems assist surgeons in performing complex procedures with greater precision and control, reducing the risk of complications and improving recovery times for patients. AI-driven chatbots and virtual assistants provide patients with medical information, reminders for medication, and even mental health support, enhancing patient engagement and ensuring continuous care outside of traditional clinical settings. AI helps streamline administrative tasks such as scheduling, billing, and managing patient records, reducing the burden on healthcare staff and allowing them to focus more on patient care. AI is also used to analyze large datasets from clinical trials and medical research, identifying patterns and insights that can lead to new medical discoveries and innovations. The integration of AI in healthcare is still evolving, but

its potential to transform the industry is immense. As AI technologies continue to advance, we can expect even more innovative solutions that will further improve patient care and healthcare delivery.

Deep Brain Stimulation and Data Collection with Graphene Diodes

Deep Brain Stimulation (DBS) is a neurosurgical procedure that involves the implantation of electrodes within certain areas of the brain. These electrodes produce electrical impulses that regulate abnormal impulses or affect certain cells and chemicals within the brain. The integration of graphene diodes in DBS is particularly promising due to their exceptional electrical conductivity and biocompatibility, which can potentially improve the efficacy of DBS treatments.

Graphene diodes provide real-time measurements of neural activity due to their exceptional electrical conductivity and biocompatibility. These measurements can be used to create comprehensive datasets that capture the dynamic changes in brain activity and symptoms. By incorporating this data into the training process, AI models can learn to predict and respond to changes in neural activity, enhancing the precision and efficacy of DBS treatments.

AI and Machine Learning (ML) algorithms can continuously monitor patient responses to treatment, adjusting the stimulation parameters in real-time to maximize efficacy. Large Language Models (LLMs) can be used to analyze patient data and generate personalized treatment plans. Generative AI can simulate various treatment scenarios and predict outcomes based on different stimulation patterns.

The utilization of graphene diodes in this context is particularly promising due to their exceptional electrical conductivity and biocompatibility, which can potentially improve the efficacy of DBS treatments. Furthermore, neural mass models serve as effective surrogates for electrophysiological signals in brain simulations, providing a robust framework for understanding and predicting neural dynamics. The incorporation of bifurcation analysis is crucial in this regard, as it facilitates the exploration of non-linear systems and aids in the design and optimization of brain stimulation regimes.

The integration of advanced technologies such as artificial intelligence (AI), machine learning (ML), large language models (LLM), and generative AI in healthcare applications presents a transformative approach to cybersecurity, observability, and monitoring in deep brain stimulation and recovery from brain diseases. This thesis explores the development and commercial application of selected AI/ML models and software development in central patient monitoring health applications and nanotechnology graphene diodes, with a focus on AI-driven charge transfer mechanisms in deep brain stimulation (DBS) for the diagnosis and treatment of brain diseases.

Rehabilitation and Vojta Methods

The rehabilitation of premature infants and the application of Vojta methods enabled the full recovery of my twin children from early-born brain strokes and intracranial hemorrhages. The trauma experienced by parents of premature infants, who initially hear about diseases typically associated with older individuals, is profound. The diagnoses of my children, tragic in the initial information, motivated us as parents to seek holistic methods of treating brain strokes and hemorrhages, culminating in strokes, based on observations of treatment methods, years of rehabilitation including the Vojta method, and years of consistent analysis of advanced algorithms combined with AI, LLM, and safe methods. This consistent search for answers led me to question how the treatment and recovery process can be scientifically predicted and based on facts. How can the stimulation of the nervous system, such as the Vojta method, influence brain recovery and consequently other organs and senses connected to the central nervous system? How do premature infants experience so-called "elderly diseases," which theoretically only occur exclusively in seniors? How can the metabolic processes, including the transformation of matter and energy in young individuals, be mirrored in the treatment processes of older individuals struggling with neurological diseases such as brain strokes or migraines with aura? How can seniors undergo faster and more effective recovery if we examine the dynamic healing and regeneration of organisms under the influence of pharmacology and numerous rehabilitations of premature infants? What are the common points? How can artificial intelligence with machine learning and LLMs be applied for the benefit of others struggling with strokes and migraines? Does not each of us go through personal tragedies to use this life school for scientific or purely human purposes, supporting others and giving them hope for a better tomorrow? These questions are answered by the author, Agnieszka Maria Mietz-Blijleven, based on her very personal, difficult experiences from the perspective of a mother, scientist, IT specialist, entrepreneur, and person who has been dealing with medical sciences, artificial intelligence, programming, and cybersecurity in IT for years.

Furtherly, the aims of the PhD Thesis open new approaches on treatment, recovery of patients from brain diseases, because Artificial Intelligence (AI) is revolutionizing healthcare in numerous ways, enhancing both patient care and the efficiency of medical processes. The aims of this PhD thesis are:

- Enhancing Diagnostic Accuracy: AI algorithms are being used to analyze medical images such as X-rays, CT scans, and MRIs, detecting abnormalities and diagnosing conditions with high accuracy, sometimes even surpassing human radiologists. This helps in the early detection and treatment of diseases like cancer or brain disorders.
- Predictive Healthcare: AI can analyze vast amounts of patient data to predict disease outbreaks, patient admissions, and even individual health risks, allowing for proactive healthcare measures, improving patient outcomes, and reducing healthcare costs.
- Personalized Treatment Plans: By analyzing genetic information and other personal health data, AI can help create personalized treatment plans tailored to individual patients, increasing the effectiveness of treatments and minimizing side effects.
- Accelerating Drug Discovery: AI accelerates the drug discovery process by predicting how different compounds will interact with targets in the body, significantly reducing the time and cost involved in bringing new drugs to market.
- AI-Powered Surgical Assistance: AI-powered robotic systems assist surgeons in performing complex procedures with greater precision and control, reducing the risk of complications and improving recovery times for patients.
- Continuous Patient Engagement: AI-driven endpoints (mobiles, computers, diodes), but also chatbots and virtual assistants provide patients with medical information, reminders for medication, and even mental health support, enhancing patient engagement and ensuring continuous care outside of traditional clinical settings.
- Streamlining Administrative Tasks: AI helps streamline administrative tasks such as scheduling, billing, and managing patient records, reducing the burden on healthcare staff and allowing them to focus more on patient care.

- Medical Research and Innovation: AI is also used to analyze large datas.ets from clinical trials and medical research, identifying patterns and insights that can lead to new medical discoveries and innovations

Introduction

In the course of my life, I have had the unique opportunity to integrate years of holistic, interdisciplinary education and experience across various domains of computer science, AI_genAI/ML/LLM, cybersecurity, charge transfer of advanced physics, medical engineering, and genetic engineering, combined with Vojta methods of rehabilitation. This journey also included the extensive process of treating my own premature children, born at the seventh month of pregnancy, classified as disabled with full recovery later on upon intensive therapy requiring hard work from physicians, doctors, rehabilitators, and myself. During their hospitalization in incubators, my twins were diagnosed with several brain and intracranial hemorrhages caused by prematurity. Over time, while being with my children at various specialized hospitals and medical centers for long months and later years of various interdisciplinary medical treatments on the path to recovery, I have noted the recovery process of my twin-children and other early-born children between the 22nd and 37th week of pregnancy—highlighting that this is classified as a normal pregnancy and under direct observation at the hospital with intensive therapy. Additionally, I have meticulously observed my own organism concerning migraines with aura throughout my development, alongside a genetic examination of migraines that have afflicted my family for generations. This has allowed me to identify and scrutinize common, previously unaddressed points, with the aim of early intervention, potentially preventing migraines with aura and, consequently, strokes.

Furthermore, I have delved deeply into the ramifications of brain hemorrhages, strokes, and the prevention of exacerbating migraine pain, which can paralyze or significantly impair the quality of both professional and personal life. With over 23 years of experience working in hospitals, medical laboratories, later at many R&D based corporates duties and as an IT architect specializing in AI_genAI/ML/Cybersecurity, I have combined this knowledge with biochemistry, analyses, and uncertainty of analytical research methods applied with medical equipment estimation for all types of analysis. This has led me to develop advanced algorithms, prompting me to attempt the creation of AI_genAI/ML/LLM models for early response and brain stimulation using charge transfer, graphene, neural networks, AI_genAI/ML/MML, cybersecurity, and the Vojta method in three specific, yet widely - holistically investigated scenarios:

- Comparing and loading data models in machine learning for the regeneration of premature infants after secondary brain hemorrhages, micro strokes, and strokes, to complementarily diagnose and treat strokes and hemorrhages in seniors using charge transfer and graphene with AI/ML.
- Stimulating the brain to prevent the development of migraines with aura by incorporating the Vojta method into the AI/ML model and observing the response pathways of neural connections in the brain.
- In cases of migraines with aura due to cervical lordosis and muscle tension, utilizing nanotechnology equipment to alleviate muscle tension and stimulate the nervous system in such a manner that blood flow through deep layers continues with oxygen without interruptions, thereby preventing the constriction of blood vessels in the brain.

My research also resulted in the development of a centralized system for international healthcare. This means that a patient diagnosed in country X and then transported for treatment in country Y does not need to undergo repeated examinations. Doctors have all the results available in one application to make decisions regarding further medical treatment. I added to this solution aspects of cybersecurity to protect data of the patient.

Artificial Intelligence (AI) is revolutionizing healthcare in numerous ways, enhancing both patient care and the efficiency of medical processes. AI algorithms are being used to analyze medical images such as X-rays, CT scans, and MRIs, detecting abnormalities and diagnosing conditions with high accuracy, sometimes even surpassing human radiologists. This helps in early detection and treatment of diseases like cancer or brain abnormalities. AI can analyze vast amounts of patient data to predict disease outbreaks, patient admissions, and even individual health risks, allowing for proactive healthcare measures, improving patient outcomes, and reducing healthcare costs. By analyzing genetic information and other personal health data, AI can help create personalized treatment plans tailored to individual patients, increasing the effectiveness of treatments and minimizing side effects. AI accelerates the drug discovery process by predicting how different compounds will interact with targets in the body, significantly reducing the time and cost involved in bringing new drugs to market. AI-powered robotic systems assist surgeons in performing complex procedures with greater precision and control, reducing the risk of complications and improving recovery times for patients. AI-driven chatbots and virtual assistants provide patients with medical information, reminders for medication, and even mental health support, enhancing patient engagement and ensuring continuous care outside of traditional clinical settings. AI helps streamline administrative tasks such as scheduling, billing, and managing patient records, reducing the burden on healthcare staff and allowing them to focus more on patient care. AI is also used to analyze large datasets from clinical trials and medical research, identifying patterns and insights that can lead to new medical discoveries and innovations. The integration of AI in healthcare is still evolving, but its potential to transform the industry is immense. As AI technologies continue to advance, we can expect even more innovative applications that will further improve patient care and healthcare delivery.

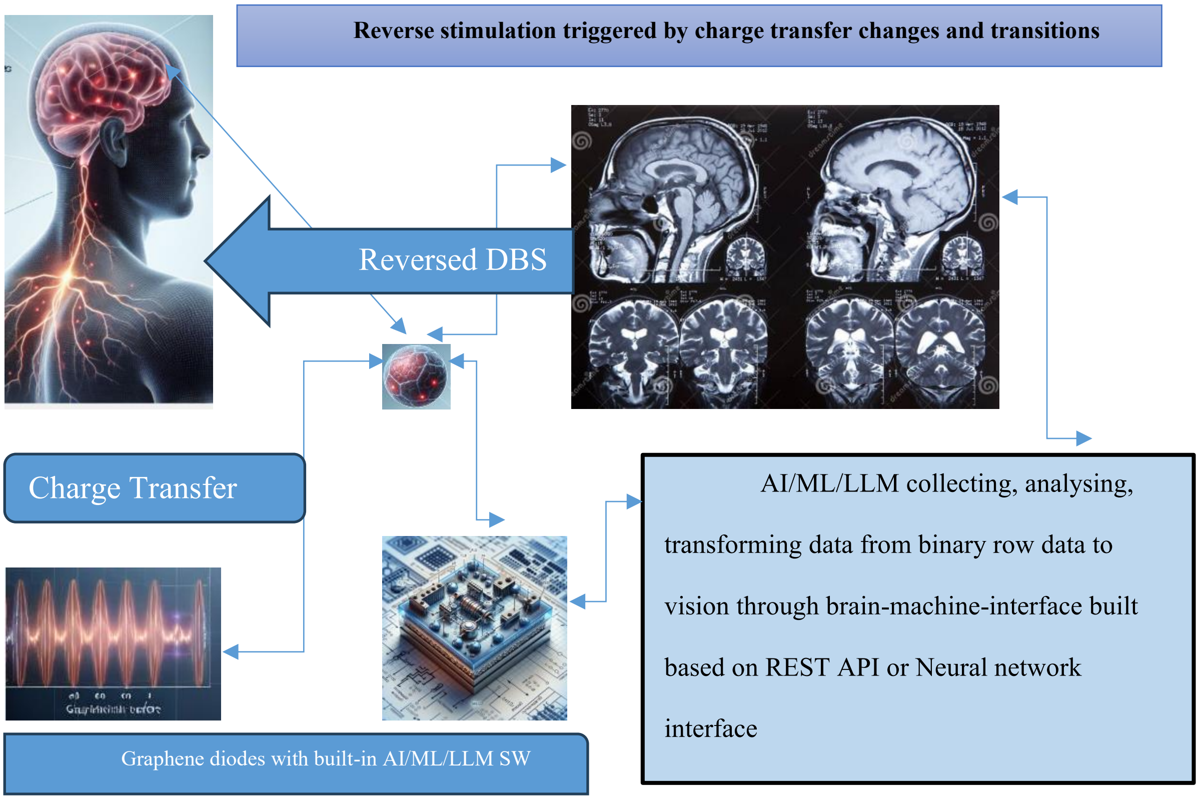
Chapter 1: Introduction, including the background, problem statement, objectives, research questions, significance, and scope.

Background of the Study

The integration of Artificial Intelligence (AI), Machine Learning (ML), and advanced algorithms with medical techniques such as the Vojta method, charge transfer in Deep Brain Stimulation (DBS) has shown promising results in various fields. This study explores the interdisciplinary applications of these technologies in neurological rehabilitation, migraine treatment, and the recovery of preterm infants and seniors. Additionally, the use of graphene in conjunction with AI and DBS is examined for its potential to enhance treatment efficacy. The majority of these insights stem from my personal experiences with my twins and myself, forming the foundation of my research methodology after years of treatment.

Solution and Architecture Overview

Graphene diodes integrated within the brain cortex are designed to monitor cerebral blood flow dynamics. These diodes are equipped with built-in stimulators capable of detecting charge transfer transitions. Upon detection, they provide an electrical stimulation impulse aimed at either dilating blood vessels or preventing their rapid constriction within the brain. This mechanism is intended to enhance oxygenation of the brain tissue and mitigate the risk of stroke or migraine with aura. Additionally, these diodes offer brain stimulation to alleviate focal points of stroke and migraine pain. This dual functionality aims to not only prevent vascular issues but also reduce the neurological pain associated with these conditions.


Reverse stimulation triggered by charge transfer changes and transitions
Reversed DBS
Charge Transfer
Graphene diodes with built-in AI/ML/LLM SW

AI/ML/LLM collecting, analysing, transforming data from binary row data to vision through brain-machine-interface built based on REST API or Neural network interface

Figure 2 Revolutionizing Brain Stimulation with Graphene Diodes, AI, ML, LLM, Vojta method.

Revolutionizing Brain
Stimulation Graphene

Video no. 1 Visual Creator AI Agent – prepared with Copilot for M365 – author: Agnieszka Mietz-Blijleven

Problem statement

The main concern is an adoption of AI, ML, LLM, Generative AI, and Cybersecurity in the diagnosis and treatment of brain diseases. Specifically, it focuses on the application of charge transfer in Deep Brain Stimulation (DBS) to enhance the efficacy of treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors. Despite extensive research, no one has yet attempted to create graphene diodes with AI/ML software that can be placed in the brain through the nostrils in key brain areas. Additionally, no attempts have been made to develop a model for stimulating the central nervous system using charge transfer transitions and stimulating the brain and nerves at anatomical points, based on the Vojta method's pressure points.

I am confident in the success presented in my PhD thesis, yet that will require next steps of development and sponsors in collaboration applying simulated solution proposed on DBS.

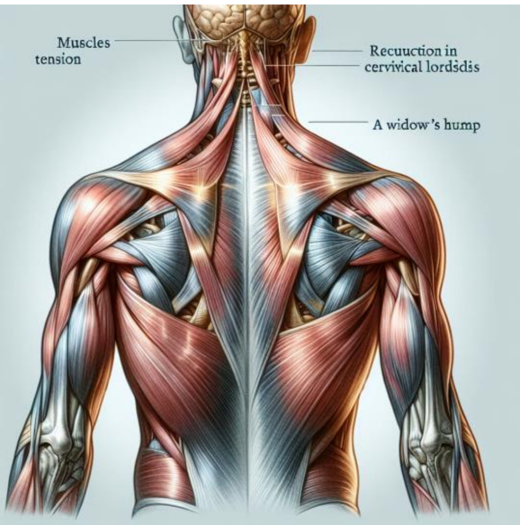


Figure 1. An anatomical image of the torso and shoulder girdle with muscle tensions leading to migraine with aura, including cervical lordosis reduction with a widow's hump.

A graphene diode with AI-Gen AI/ML/LLM multidirectional for deep brain stimulation. The image indicates the muscle tensions causing migraine, changing the electric charge of neurons and the brain. It depicts the charge transfer captured by the graphene diode, which records data with built-in AI/ML software. The diode stimulates the nervous system to neutralize muscle tensions, regulate blood flow, and alleviate migraines and migraines with aura.

In addition, the application of the mathematical MASS model with R in deep brain stimulation (DBS) or use of commercially available solutions like for instance Azure AI Foundry can be significantly enhanced by integrating artificial intelligence (AI) and machine learning (ML) algorithms with medical techniques such as the Vojta method and charge transfer processes. The utilization of graphene diodes in this context is particularly promising due to their exceptional electrical conductivity and biocompatibility, which can potentially improve the efficacy of DBS treatments. Furthermore, neural mass models serve as effective surrogates for electrophysiological signals in brain simulations, providing a robust framework for understanding and predicting neural dynamics. The incorporation of bifurcation analysis is crucial in this regard, as it facilitates the exploration of non-linear systems and aids in the design and optimization of brain stimulation regimes.

Aspect	Brain Stroke	Migraine with Aura
Increased Risk of Brain Diseases or Disorders	**Possible consequence of migraine with aura**	**Migraine, especially with aura, is a risk factor for both ischemic and haemorrhagic stroke.**
Sensory Disturbances	**Not typically associated with sensory disturbances.**	**Characterized by sensory disturbances such as seeing flashes of light, zigzag lines, tingling sensations, and difficulty speaking.**
Enhanced Cerebral Excitability	**Involves enhanced cerebral excitability and increased incidence of microembolic events.**	**Involves enhanced cerebral excitability and increased incidence of microembolic events.**
Spreading Depolarization (SD)	**Not typically associated with SD.**	**This electrophysiologic event underlies migraine aura and is a known headache trigger. Increased SD susceptibility has been demonstrated in migraine animal models.**
Microembolic Events	**Patients are at risk for cardioembolic stroke due to increased incidence of coagulopathy, atrial fibrillation, and patent foramen ovale.**	**Patients are at risk for cardioembolic stroke due to increased incidence of coagulopathy, atrial fibrillation, and patent foramen ovale.**

Aspect	Brain Stroke	Migraine with Aura
Genetic Factors	**Genetic mutations associated with stroke can lead to increased frequency of ischemia-triggered SDs.**	**Genetic mutations associated with migraine can lead to increased frequency of ischemia-triggered SDs upon experimental stroke.**

Table 1. Factors that may cause stroke and migraine with aura

Graphene Diodes Embedded in the Brain Cortex for Monitoring Cerebral Blood Flow and Neurological Stimulation

Graphene diodes integrated within the brain cortex are designed to monitor cerebral blood flow dynamics. These diodes are equipped with built-in stimulators capable of detecting charge transfer transitions. Upon detection, they provide an electrical stimulation impulse aimed at either dilating blood vessels or preventing their rapid constriction within the brain. This mechanism is intended to enhance oxygenation of the brain tissue and mitigate the risk of stroke or migraine with aura.

Additionally, these diodes offer brain stimulation to alleviate focal points of stroke and migraine pain. This dual functionality aims to not only prevent vascular issues but also reduce the neurological pain associated with these conditions.

Data Measurement and Analysis Using AI, LLM, and ML

The data collected by the graphene diodes can be analysed using advanced AI techniques, including large language models (LLMs) and machine learning (ML) algorithms. Here's how it works:

1. **Data Collection**: Graphene diodes continuously monitor and record electrical signals and charge transfer transitions within the brain. They also analyse the root cause of migraine attempts, whether the biochemical change in the organism had triggered the

attempt of migraine with aura or perhaps the muscle tension of the shoulder girdle (muscle tension, widow's hump).

2. **Data Transmission**: The recorded data is transmitted to a central processing unit equipped with AI capabilities.

3. **Data Analysis**:

    - **Machine Learning (ML)**: ML algorithms process the data to identify patterns and anomalies in cerebral blood flow and neural activity. These algorithms can learn from historical data to improve accuracy over time.

    - **Large Language Models (LLM)**: LLMs can interpret complex data sets and provide insights into the brain's physiological and pathological states. They can also generate detailed reports and recommendations for medical professionals.

4. **Real-Time Feedback**: The AI system provides real-time feedback to the graphene diodes, enabling immediate adjustments in electrical stimulation to optimize brain function and prevent adverse events like strokes or migraines.

5. **Predictive Analysis**: By leveraging AI and ML, the system can predict potential issues before they occur, allowing for proactive intervention.

This integration of graphene diodes with AI, LLM, and ML represents a cutting-edge approach to brain health, combining real-time monitoring with advanced data analysis to improve patient outcomes

Innovative technologies such as artificial intelligence (AI), brain-computer interfaces (BCIs), and nanotechnology are accelerating neuroscience research in the quest to improve human health and daily lives. Researchers at the University of California San Diego (UCSD) have developed a novel transparent brain-computer interface capable of providing high-resolution neural recordings from the brain’s surface, utilizing AI, machine learning, and a nanomaterial called graphene.

Graphene diodes integrated within the brain cortex are designed to monitor cerebral blood flow dynamics. These diodes are equipped with built-in stimulators capable of detecting charge transfer transitions. Upon detection, they provide an electrical stimulation impulse aimed at either dilating blood vessels or preventing their rapid constriction within the brain. This mechanism is intended to enhance oxygenation of the brain tissue and mitigate the risk of stroke or migraine with aura. Additionally, these diodes offer brain stimulation to alleviate focal points of stroke and migraine pain.

Every one out of six people, approximately 16% of the global population, experience significant disability according to the World Health Organization (WHO). Brain-computer interfaces, also called brain-machine interfaces (BMIs), are enabling technologies that offer hope to those who have lost the ability to speak or move after brain stroke. With the help of a brain-computer interface, a person can manage and operate external electronic devices with just thoughts to communicate via synthesized speech, move prosthetic limbs, operate a computer, and perform other important functions that improve the quality of life for those with disabilities.

The brain-computer interface market, a USD 2 billion industry in 2023, is expected to reach USD 6.2 billion by 2030 with a compound annual growth rate (CAGR) of 17.5% during 2020-2030 according to the Brain Computer Interface Market Size & Share Report 2030 by Grand View Research. Per the report, North America had the largest revenue share globally at 39.5% in 2022. A growing aging population is expected to contribute to the BCI market growth as the prevalence of Alzheimer's disease, Parkinson's disease, Huntington's disease, and other neurodegenerative disorders increases.

"Recordings of neural activity at depth without implanting invasive neural probes could extend the lifetime of neural implants and improve the longevity of BCI technologies and pave the way for their medical translation," wrote UCSD researchers Duygu Kuzum, Mehrdad Ramezani, Jeong-Hoon Kim, Xin Liu, Chi Ren, Abdullah Alothman, Chawina De-Eknamkul, Madison N. Wilson, Ertugrul Cubukcu, Vikash Gilja, and Takaki Komiyama.

What sets this brain-computer interface apart is the ability to record brain activity via both optical imaging and electrical signals simultaneously. Unlike conventional BCI implants which are opaque, this new BCI is transparent, providing neuroscientists with a window for observation via microscopy. As the transparent graphene electrode array records electrical signals from the neurons located in the brain's outer layers, at the same time, the calcium spikes from neurons up to 250 micrometres deep are imaged using a two-photon microscope shining laser lights through the array. In this manner, the researchers were able to correlate the electrical signals at the brain's outer layers with calcium spike activity in the deeper parts of the brain.

The correlation data was used as training data for an AI artificial neural network. The UCSD researchers created an AI model with a linear hidden layer, a single-layer bidirectional LSTM (Long Short-Term Memory), or BiLSTM, and a linear readout layer. The AI model learned from the correlation data in order to predict the calcium activity in the deeper parts of the brain based on the electrical signals on the outer layer. This enables neuroscientists to observe brain activity for longer periods as the organism is moving around freely versus being locked under a microscope for a short duration.

Objectives of PhD Thesis

The objectives of my PhD are focused on integrating Artificial Intelligence (AI), Machine Learning (ML), Large Language Models (LLM), Generative AI, and Cybersecurity with charge transfer and Vojta method in Deep Brain Stimulation (DBS) to enhance the diagnosis and treatment of brain diseases. Specifically, my research aims to:

1. Regenerate premature infants after secondary brain hemorrhages, micro strokes, and strokes by comparing and loading data models in machine learning. This also includes complementarily diagnosing and treating strokes and hemorrhages in seniors using charge transfer and graphene.
2. Prevent the development of migraines with aura by incorporating the Vojta method into the AI/ML model and observing the response pathways of neural connections in the brain.
3. Alleviate muscle tension and stimulate the nervous system in cases of migraines with aura due to cervical lordosis and muscle tension. This involves using nanotechnology equipment to ensure continuous blood flow through deep layers without interruptions.
4. Develop mechanism based on graphene diodes with AI/ML/LLM/gen AI software with charge transfer focused on stimulation of areas in brain to preven strokes and migraine with aura occurence. Such mechanism should be corresponding to Vojta Method in reverse respond of diodes stimulation.

5. Develop a centralized system for international healthcare to ensure that patients diagnosed in one country and transported to another for treatment do not need repeated examinations. This system includes aspects of cybersecurity to protect patient data

Research questions

Some potential research questions for my PhD thesis that I have risen to myself before approaching researches, observations:

1. How can AI, ML, LLM, and Generative AI be integrated into the diagnosis and treatment of brain diseases to enhance the efficacy of Deep Brain Stimulation (DBS)?
2. What are the potential benefits and challenges of using graphene diodes with AI/ML software for electrical stimulation in key brain areas through the nostrils?
3. How can charge transfer transitions be utilized to stimulate the central nervous system and improve treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors?
4. What is the role of the Vojta method's pressure points in developing an AI/genAI model for stimulating the brain and nerves at anatomical points using charge transfer transitions?
5. How can cybersecurity measures be implemented to ensure the safety and reliability of AI/ML applications in DBS treatments?

These questions, beyond of many others aim to explore the innovative aspects of my research and address the key concerns related to the adoption of advanced technologies in brain disease treatments.

Brain stroke and migraine with aura - common points and root causes of occurrence.

While brain stroke and migraine with aura are closely related, there is no direct evidence to suggest that a brain stroke can lead to migraine with aura. However, there is a known link between the two conditions. People who experience migraine with aura may be at a higher risk of stroke, particularly ischemic stroke, which occurs when a blood clot blocks blood flow to a part of the brain.

Migraine with aura involves visual disturbances and other sensory symptoms that can sometimes resemble stroke symptoms. It's important to understand the differences between the two conditions to ensure proper diagnosis and treatment.

How do brain strokes and migraines with aura occur in infants and seniors, and under what circumstances?

Aspect	**Infants/Early-born**	**Seniors**
Brain Strokes		
Blood	Blood clots more easily due to immature clotting systems, e.g., congenital heart defects, like VDA, AVD.	Blood clots due to atherosclerosis, atrial fibrillation, or other cardiovascular issues, e.g., plaque buildup in arteries.
Pressure	Fragile blood vessels can lead to haemorrhagic strokes, e.g., intracranial haemorrhage.	High blood pressure is a major risk factor for ischemic and haemorrhagic strokes, e.g., hypertension exceeding 140/90 mmHg.
Brain	Brain injury due to lack of oxygen or bleeding, e.g., perinatal stroke	Brain tissue damage due to blocked or burst blood vessels, e.g., ischemic stroke
Heart	Congenital heart defects can increase stroke risk, e.g., patent ductus arteriosus.	Heart disease and atrial fibrillation can lead to embolic strokes, e.g., irregular heartbeats causing clots.
Symptoms	Seizures, extreme sleepiness, weakness on one side, e.g., hemiparesis.	Sudden numbness, confusion, trouble speaking, vision problems, severe headache, e.g., transient ischemic attack.

Aspect	**Infants/Early-born**	**Seniors**
Migraine with Aura		
Blood	Possible genetic factors affecting blood vessels, e.g., familial hemiplegic migraine.	Blood flow changes and inflammation in blood vessels, e.g., vasospasm.
Pressure	Blood pressure changes can trigger migraines, e.g., sudden spikes in blood pressure.	High or low blood pressure can influence migraine severity, e.g., hypertension or hypotension.
Brain	Visual disturbances due to temporary changes in brain activity, e.g., cortical spreading depression	Visual and sensory disturbances due to altered blood flow in the brain, e.g., aura without headache or opposite: vast headache pain
Heart	Rare in infants, but possible genetic predispositions, e.g., mitochondrial disorders.	Increased risk of heart disease and stroke with migraines, e.g., cardiovascular comorbidities.
Symptoms	Visual disturbances, throbbing head pain, nausea, e.g., seeing flashing lights.	Visual aura, sensory changes, speech difficulties, e.g., temporary vision loss.

Table 2 Factors that may cause stroke and migraine with aura in premature infants and seniors

How can we optimize the treatment of brain strokes in seniors by comparing the pathophysiology of strokes in infants and migraines with aura, utilizing advanced techniques such as charge transfer, deep brain stimulation (DBS), artificial intelligence (AI), machine learning (ML), large language models (LLM), and the Vojta method?

Aspect	Infants/Early-born	Seniors
Brain Strokes		
Pathophysiology	**Congenital heart defects, blood clotting disorders, blood vessel malformations.**	**Atherosclerosis, hypertension, atrial fibrillation.**
Symptoms	**Seizures, extreme sleepiness, hemiparesis.**	**Sudden numbness, confusion, severe headache.**
Charge Transfer	**Limited application due to delicate physiology.**	**Improves blood flow, reduces clot formation.**
Deep Brain Stimulation (DBS)	**Rarely used; potential for future applications.**	**Manages chronic pain, improves motor function post-stroke.**
Artificial Intelligence (AI)	**Early diagnosis, personalized treatment plans.**	**Predicts stroke risk, optimizes rehabilitation strategies.**

Aspect	Infants/Early-born	Seniors
Machine Learning (ML)	**Analyzes genetic data to predict stroke risk.**	**Monitors recovery, adjusts treatments based on real-time data.**
Large Language Models (LLM)	**Provides educational resources for caregivers.**	**Offers personalized health advice and support.**
Vojta Method	**Effective for early intervention in neurological issues.**	**Improves motor skills and coordination.**
Migraine with Aura		
Pathophysiology	**Genetic factors, somatic stress in incubator and separation from mother/father, environmental stimuli.**	**Main focus in PhD thesis: Loss of cervical lordosis, muscle tension due to stress, years of working hunched over computers, dowager's hump, leading to numerous spinal diseases and nerve compressions. Hormonal changes, dehydration, underlying health conditions.**
Symptoms	**Visual disturbances, nausea, throbbing head pain.**	**Visual aura, sensory changes, speech difficulties.**

Aspect	Infants/Early-born	Seniors
Charge Transfer	Limited application.	Can influence blood flow changes.
Deep Brain Stimulation (DBS)	Rarely used.	Potential for managing chronic migraine pain.
Artificial Intelligence (AI)	Early diagnosis, personalized treatment plans.	Predicts migraine triggers, optimizes management strategies.
Machine Learning (ML)	Analyzes genetic predispositions.	Monitors symptoms, adjusts treatments.
Large Language Models (LLM)	Provides educational resources.	Offers personalized health advice.
Vojta Method	Effective for early intervention.	Can be adapted for improving coordination.

Table 3 Optimizing Treatment of Brain Strokes in Seniors by Comparing Pathophysiology of Strokes in Infants and Migraines with Aura Using Advanced Techniques.

The significance of PhD thesis

The significance of my PhD thesis, titled "Artificial Intelligence with Charge Transfer in Deep Brain Stimulation: New Approaches Based on AI/ML, LLM, Generative AI, and Cybersecurity in the Diagnosis and Treatment of Brain Disease", lies in its innovative, holistic and interdisciplinary approach to enhancing the diagnosis and treatment of brain diseases. Here are the key points of significance:

1. Integration of Advanced Technologies: My research integrates Artificial Intelligence (AI), Machine Learning (ML), Large Language Models (LLM), Generative AI, and Cybersecurity with medical techniques such as the Vojta method and from advanced, applied Physics – the charge transfer in Deep Brain Stimulation (DBS). This interdisciplinary approach aims to improve the efficacy of treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors.

2. Innovative Use of Graphene Diodes: The thesis explores the potential of using graphene diodes with AI/ML software for electrical stimulation in key brain areas. This novel approach aims to enhance the efficacy of DBS treatments by leveraging the unique properties of graphene, such as its exceptional electrical conductivity and biocompatibility.

3. Focus on Charge Transfer Transitions: My research delves into the application of charge transfer transitions to stimulate the central nervous system. This involves developing a model for stimulating the brain and nerves at anatomical points based on the Vojta method's pressure points. This innovative approach aims to improve treatments for neurological disorders and migraines with aura.

4. Personal and Observational Insights: The thesis is grounded in my personal experiences and observations, particularly in treating my own premature children and managing migraines with aura. This unique perspective adds a valuable dimension to the research, highlighting the practical implications and potential benefits of the proposed solutions.

5. Development of a Centralized Healthcare System: My research also resulted in the development of a centralized system for international healthcare. This system ensures that patients diagnosed in one country and transported to another for treatment do not need repeated examinations. It includes aspects of cybersecurity to protect patient data, enhancing the efficiency and security of medical treatments.

6. Potential for Broad Applications: The findings of my research have the potential to be applied to a wide range of neurological conditions, including brain strokes, migraines with aura, and the recovery of preterm infants and seniors. The interdisciplinary approach and innovative use of advanced technologies could lead to significant improvements in patient outcomes and the overall quality of healthcare.

My PhD thesis represents a significant contribution to the field of medical science by integrating advanced technologies with traditional medical techniques to enhance the diagnosis and treatment of brain diseases. The innovative use of graphene diodes, charge transfer transitions, and AI/ML models holds great promise for improving the efficacy of treatments and the quality of patient care. Such study had not been performed so far in such holistic way based on my current awareness.

Scope of PhD Thesis

The scope of my PhD thesis, titled "Artificial Intelligence with Charge Transfer in Deep Brain Stimulation: New Approaches Based on AI/ML, LLM, Generative AI, and Cybersecurity in the Diagnosis and Treatment of Brain Disease", is comprehensive and interdisciplinary. It encompasses several key areas of research and application:

1. Integration of Advanced Technologies

My research integrates Artificial Intelligence (AI), Generative Artificial Intelligence (genAI), Machine Learning (ML), Large Language Models (LLM), and Cybersecurity with medical techniques such as the Vojta method and charge transfer with graphene diodes in Deep Brain Stimulation (DBS). This interdisciplinary approach aims to improve the efficacy of treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors.

2. Innovative Use of Graphene Diodes

The thesis explores the potential of using graphene diodes with AI/ML software for electrical stimulation in key brain areas. This novel approach aims to enhance the efficacy of DBS treatments by leveraging the unique properties of graphene, such as its exceptional electrical conductivity and biocompatibility.

3. Focus on Charge Transfer Transitions

My research delves into the application of charge transfer transitions to stimulate the central nervous system. This involves developing a model for stimulating the brain and nerves at anatomical points based on the Vojta method's pressure points. This innovative approach aims to improve treatments for neurological disorders and migraines with aura.

4. Personal conclusions on statement of a problem and Observational Insights

The thesis is grounded in my personal experiences and observations, particularly in treating my own premature children and managing migraines with aura. This unique perspective adds a valuable dimension to the research, highlighting the practical implications and potential benefits of the proposed solutions.

5.AI/ML/LLM mechanism of DBS with graphene diodes with charge transfer

Graphene diodes, due to their exceptional electrical conductivity and biocompatibility, can be integrated with charge transfer mechanisms to stimulate neuronal activity. This approach can be particularly beneficial for treating neurological conditions such as migraines with aura and brain strokes. The treatment involves the application of graphene diodes to specific regions of the brain, where they facilitate charge transfer to modulate neuronal activity. AI and ML algorithms can continuously monitor patient responses to treatment, adjusting the stimulation parameters in real-time to maximize efficacy. Large Language Models (LLMs) can be used to analyze patient data and generate personalized treatment plans. Generative AI can simulate various treatment scenarios and predict outcomes based on different stimulation patterns. The MASS package in R provides statistical tools to analyze the effectiveness of the treatment. By applying these tools, researchers can evaluate the impact of graphene diodes and charge transfer mechanisms on recovery processes. The stimulation process involves the application of graphene diodes to the cortex of the brain, where they facilitate with neural networks and charge transfer to modulate neuronal activity. The organism's response to this stimulation can be monitored using AI/ML algorithms, which adjust the stimulation parameters in real-time to ensure optimal results. For patients with musculoskeletal disorders, graphene diodes can be used to target specific muscle groups, reducing tension and improving mobility. Deep brain stimulation (DBS) using graphene diodes can be tailored to individual patients by adjusting the stimulation parameters based on real-time monitoring and AI/ML algorithms. By integrating these advanced techniques and technologies, we can develop more effective treatment strategies for neurological conditions, tailored to the specific needs of different patient groups. This holistic approach can significantly improve recovery outcomes and quality of life for both early-born infants and seniors.

6. Development of a Centralized Healthcare System

My research also resulted in the development of a centralized system for international healthcare. This system ensures that patients diagnosed in one country and transported to another for treatment do not need repeated examinations. It includes aspects of cybersecurity to protect patient data, enhancing the efficiency and security of medical treatments.

7. Potential for Broad Applications

The findings of my research have the potential to be applied to a wide range of neurological conditions, including brain strokes, migraines with aura, and the recovery of preterm infants and seniors. The interdisciplinary approach and innovative use of advanced technologies could lead to significant improvements in patient outcomes and the overall quality of healthcare.

Conclusion

My PhD thesis represents a significant contribution to the field of medical science by integrating advanced technologies with traditional medical techniques to enhance the diagnosis and treatment of brain diseases. The innovative use of graphene diodes, charge transfer transitions, and AI/ML models holds great promise for improving the efficacy of treatments and the quality of patient care.

Chapter 2: Literature Review, covering existing research on AI, ML, DBS, the Vojta method, charge transfer transitions, graphene applications, and migraine treatment.

The integration of advanced technologies such as artificial intelligence (AI), machine learning (ML), large language models (LLM), generative AI, Azure AI Services, Foundry, Open AI, and software development in healthcare applications presents a transformative approach to cybersecurity, observability, and monitoring. This literature review explores the development and commercial application of these technologies in central patient monitoring health applications and nanotechnology diodes, with a focus on AI-driven charge transfer mechanisms in deep brain stimulation (DBS) for the diagnosis and treatment of brain diseases.

AI and ML in Healthcare

AI algorithms are being used to analyse medical images such as X-rays, CT scans, and MRIs, detecting abnormalities and diagnosing conditions with high accuracy. This helps in the early detection and treatment of diseases like cancer or brain disorders. AI can analyse vast amounts of patient data to predict disease outbreaks, patient admissions, and individual health risks, allowing for proactive healthcare measures. Generative AI is being implemented for personalized treatment plans and real-time monitoring and adjustment of DBS. This includes predictive modelling and fine-tuning language models for medical applications.

Deep Brain Stimulation (DBS)

The application of charge transfer in DBS enhances the efficacy of treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors. Graphene diodes are being explored for their potential to improve DBS treatments due to their exceptional electrical conductivity and biocompatibility. Utilizing graphene diodes for monitoring cerebral blood flow dynamics and providing electrical stimulation to enhance brain oxygenation and mitigate stroke risks. AI and ML algorithms can continuously monitor patient responses to treatment, adjusting stimulation parameters in real-time to maximize efficacy. Generative AI can simulate various treatment scenarios and predict outcomes based on different stimulation patterns.

Vojta Method

The Vojta method, pioneered by Dr. Vaclav Vojta, is a transformative approach to neurological rehabilitation that harnesses the power of reflexive locomotion to reignite the brain's innate ability to rewire itself and restore lost motor function. Developed in the 1950s, this groundbreaking technique has revolutionized the field of physiotherapy, offering hope to countless individuals struggling with neurological and musculoskeletal disorders.

At its core, Vojta Therapy aims to activate the central nervous system through carefully applied stimuli. By doing so, it seeks to unlock innate movement patterns that may have been disrupted due to neurological damage or developmental issues. These patterns form the building blocks of all complex movements, from crawling as babies to walking as adults.

Vojta Therapy uses pressure points to trigger innate movement patterns. By applying pressure to specific areas of the body, therapists can activate these patterns, setting off a chain reaction throughout the entire central nervous system. This activation doesn't just affect the muscles being directly stimulated but influences the entire nervous system.

Reflex locomotion is activated from three main positions: prone, supine, and side lying. To stimulate the patterns of movement, there are ten available zones on the body, including the arms and legs . Vojta Therapy can be used in adults with acquired impairments that affect the peripheral and central nervous regulation of movement. It is effective even in the acute phase of illness and subsequent rehabilitation. Reflex creeping is a technique used in adults to stimulate movement patterns.

Charge Transfer Transitions

The mechanism of charge transfer in physics, including conduction and induction, can be related to the occurrence of strokes and migraines with aura through several key aspects.

Graphene Applications

Graphene's exceptional electrical conductivity and biocompatibility make it a promising material for various medical applications, including DBS and monitoring devices. Graphene diodes provide real-time measurements of neural activity due to their exceptional electrical conductivity and biocompatibility. These measurements can be used to create comprehensive datasets that capture dynamic changes in brain activity and symptoms.

Recent studies have highlighted several breakthrough applications of graphene

Industrial Applications: Graphene is enabling a new generation of breakthrough industrial applications. Its unique properties, such as electrical and thermal conductivity, thermal stability, and lubricating characteristics, have significantly impacted industries like electronics, batteries, composites, and refractories .

Market Growth

The global graphene market is expected to grow significantly, with the demand for graphene and graphene-based materials on the rise due to their applications across various industries .

Innovative Applications

A recent study demonstrated the creation of an electric circuit using a simple marker and a laser beam, showing that sustainable materials can generate innovative applications on any surface.

Fire Detection

A study in ACS Applied Materials & Interfaces highlights graphene's potential as an effective material for quicker, more reliable fire detection due to its high electron mobility, superior thermal conductivity, high mechanical properties, and structural stability under high temperatures.

Migraine Treatment

AI and ML are being used to develop predictive models for migraine treatment, enabling personalized treatment plans and real-time monitoring and adjustment of therapies.

Regulatory aspects

Doora Requirements and NIS2: These topics were not specifically covered in the retrieved documents, but they are essential for understanding regulatory and compliance aspects in AI and cybersecurity applications in healthcare.

Advanced Physics of Charge Transfer: This area includes the study of conduction and induction mechanisms, which are crucial for understanding the application of charge transfer in medical treatments.

Personal Insights and Experiences

My personal experiences with your twins and yourself have significantly contributed to your research methodology. My can-do attitude, consequences, and dedication to recovery and rehabilitation have driven your extensive study of medicine and the healing process.

PhD Thesis: Development and Commercial Application of Nexthink for Cybersecurity, Observability, and Monitoring in Central Patient Monitoring Health Applications and **Nanotechnology Diodes**

This thesis explores the integration of advanced technologies such as Nexthink, AI, ML, LLM, and generative AI in healthcare applications, focusing on AI-driven charge transfer mechanisms in DBS for the diagnosis and treatment of brain diseases.

By leveraging AI, ML, generative AI, and charge transfer mechanisms in DBS, we can develop robust cybersecurity, observability, and monitoring solutions for central patient monitoring health applications and nanotechnology diodes. These approaches will enhance patient outcomes and ensure the responsible use of AI in healthcare. This holistic approach, combining medical science, advanced physics, AI, generative AI, LLM, ML, software development, experimentation, and observability, is demanding but essential for advancing healthcare technology.

Chapter 3: Methodology, detailing the research design, data collection, and analysis methods.

Statement of the Problem

Over years of research and observation, and despite extensive searches for study results, no one has yet attempted to create graphene diodes with AI/ML software that can be placed in the brain through the nostrils in key brain areas. Additionally, no attempts have been made to develop a model for stimulating the central nervous system using charge transfer transitions and stimulating the brain and nerves at anatomical points, based on the Vojta method's pressure points.

Let's have a look at particular stages of brain disease and possible stimulation

Brain – migraine with aura:

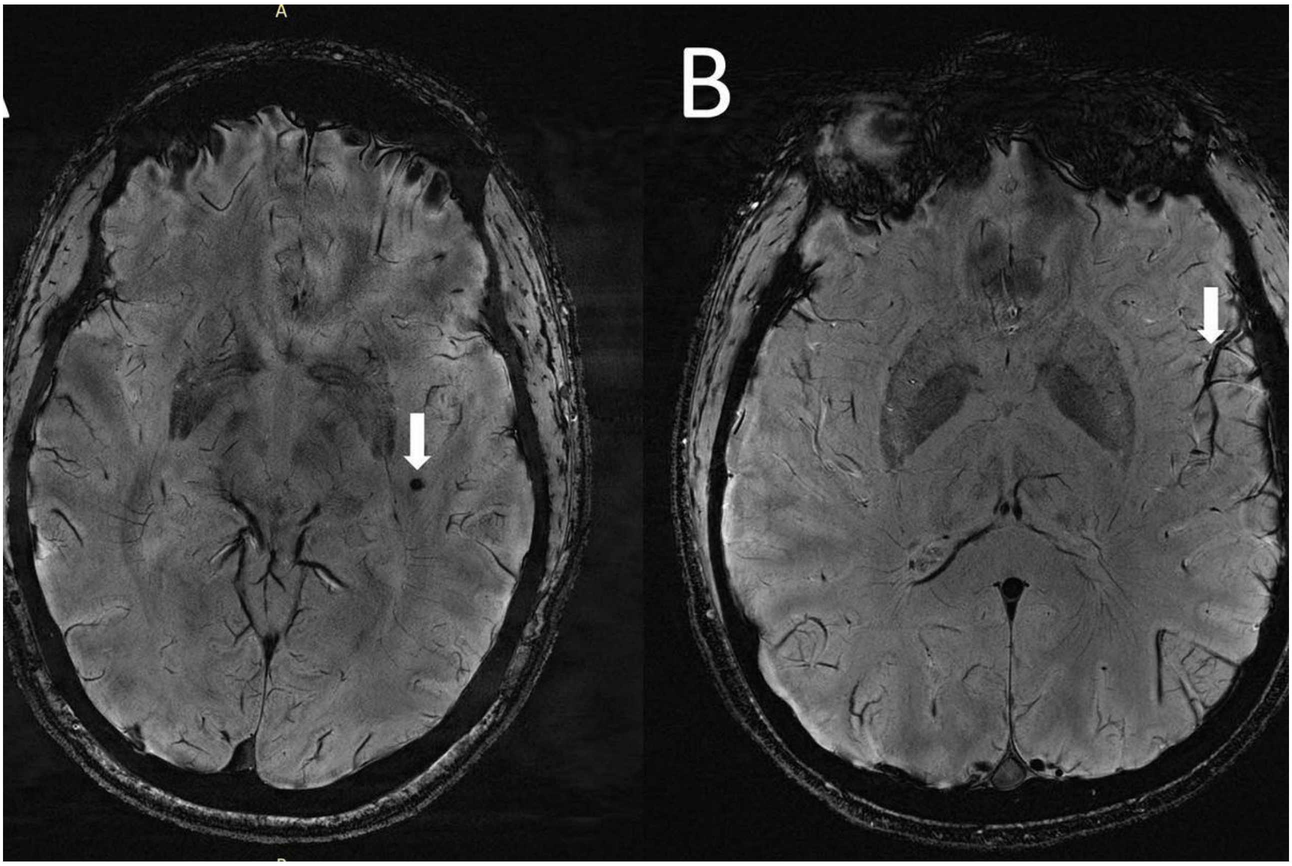


Figure 3 (A) Enlarged brain spaces visible as round, dark changes (arrow) in the left temporal lobe in the case of migraine with aura. (B) Asymmetry in the appearance of cortical vessels. (RSNA & Wilson Xu).

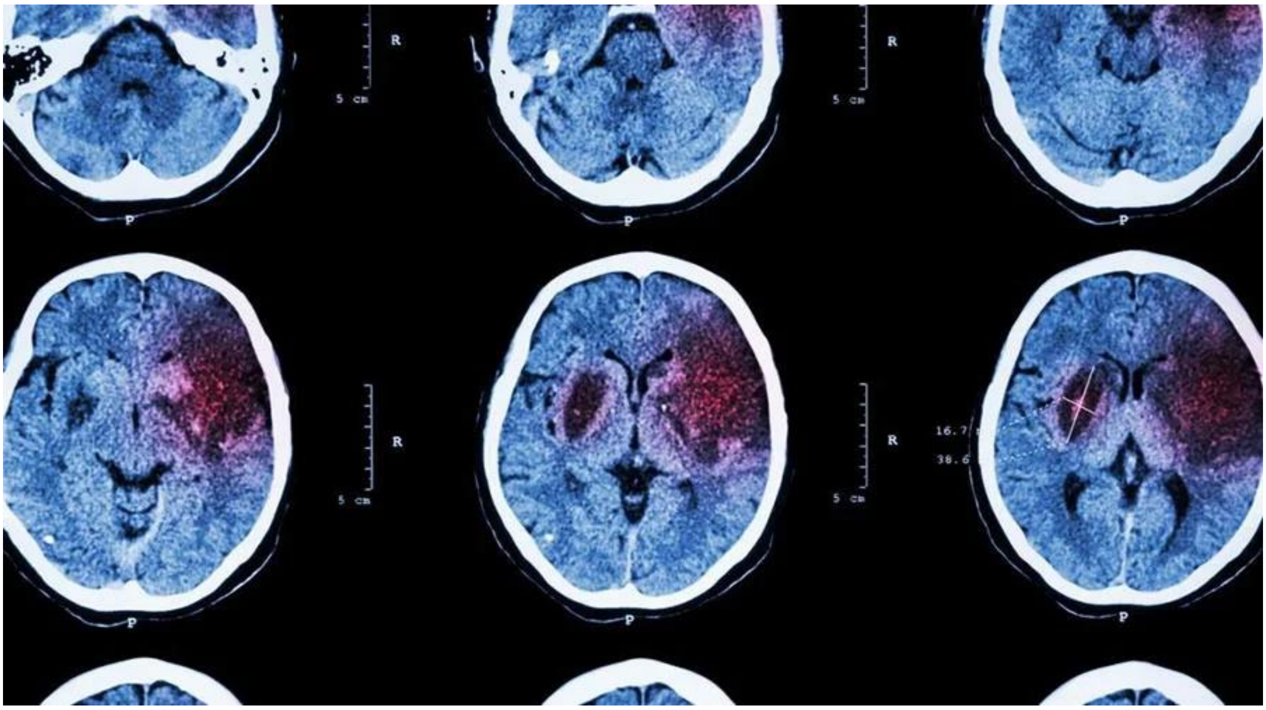


Figure 4 Brain with stroke

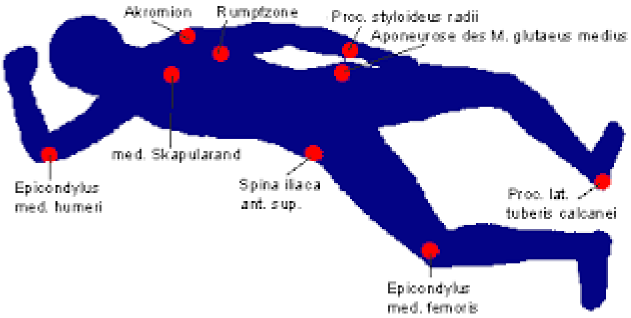


Figure 5 Critical points of anatomical stimulation with use of Vojta method

Despite advancements in medical technology, there remain significant challenges in effectively treating neurological disorders, migraines with aura, and ensuring the optimal development of preterm infants. This study aims to address these challenges by leveraging AI, genAI, ML, nanotechnology, and graphene to enhance existing medical treatments. The research is based on personal observations and experiences, as testing on other individuals was not possible.

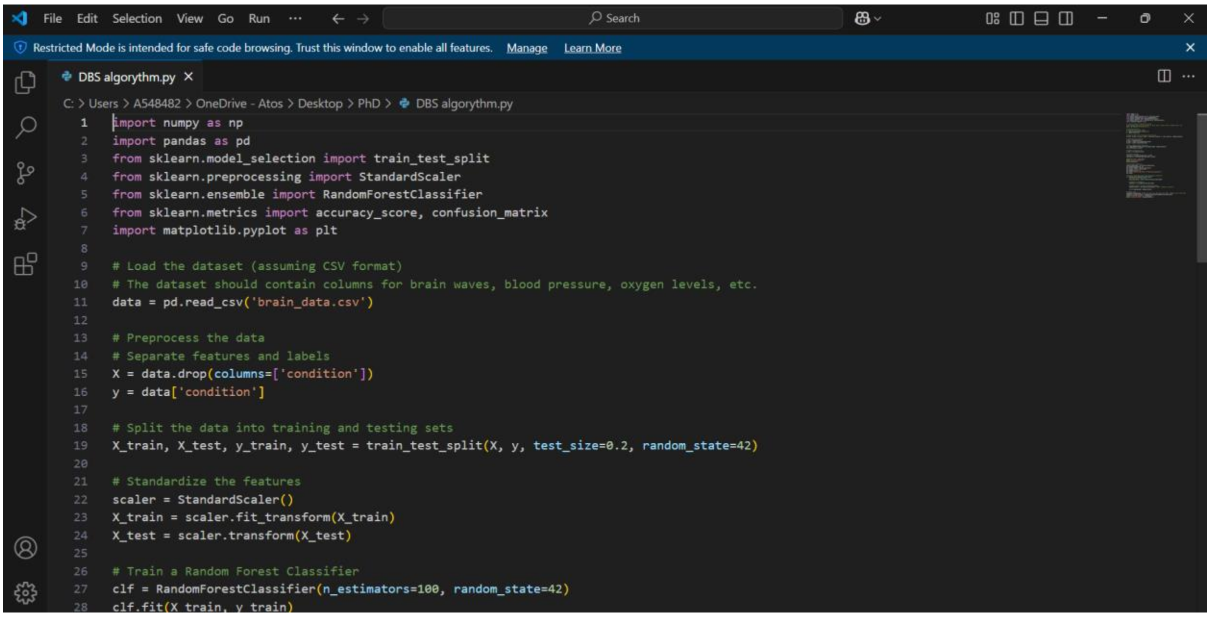
So far Graphene Flagship scientists have developed a sensor based on CVD graphene that detects brain signals in a wide frequency band, from extremely low frequencies to high frequency oscillations. The sensor is biocompatible and could be used to measure and predict brain states.

A newly developed graphene-based implant can record electrical activity in the brain at extremely low frequencies and over large areas, unlocking the wealth of information found below 0.1 Hz.

Another option under consideration are Graphene tattoos are ultra-thin, flexible patches that can monitor brain wave activity continuously with minimal impact on a person's daily life.

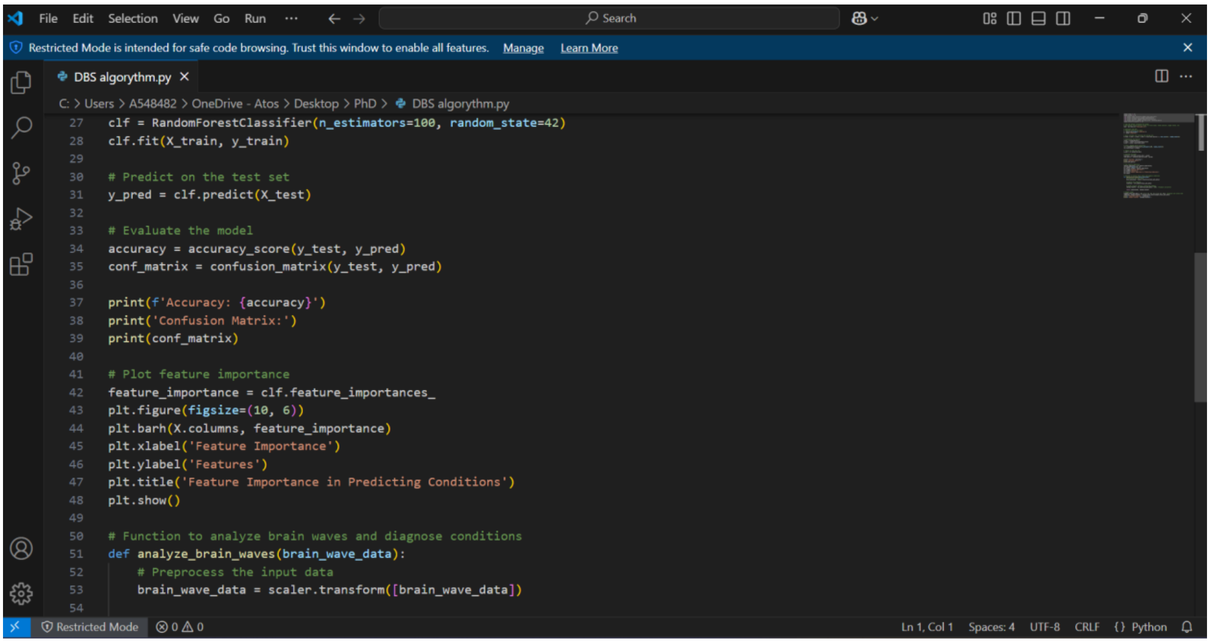
Deep Brain Stimulation with AI/genAI/ML/LLM/Charge transfer/Vojta Method. Graphene diode – nanotechnology - Approach, outlining the framework for research design, techniques for gathering data, and strategies for analysis.

First step is to build algorithm that provides a basic framework for analyzing brain waves and diagnosing conditions using machine learning.


File
Edit
Selection
View
Go
Run
Search
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py

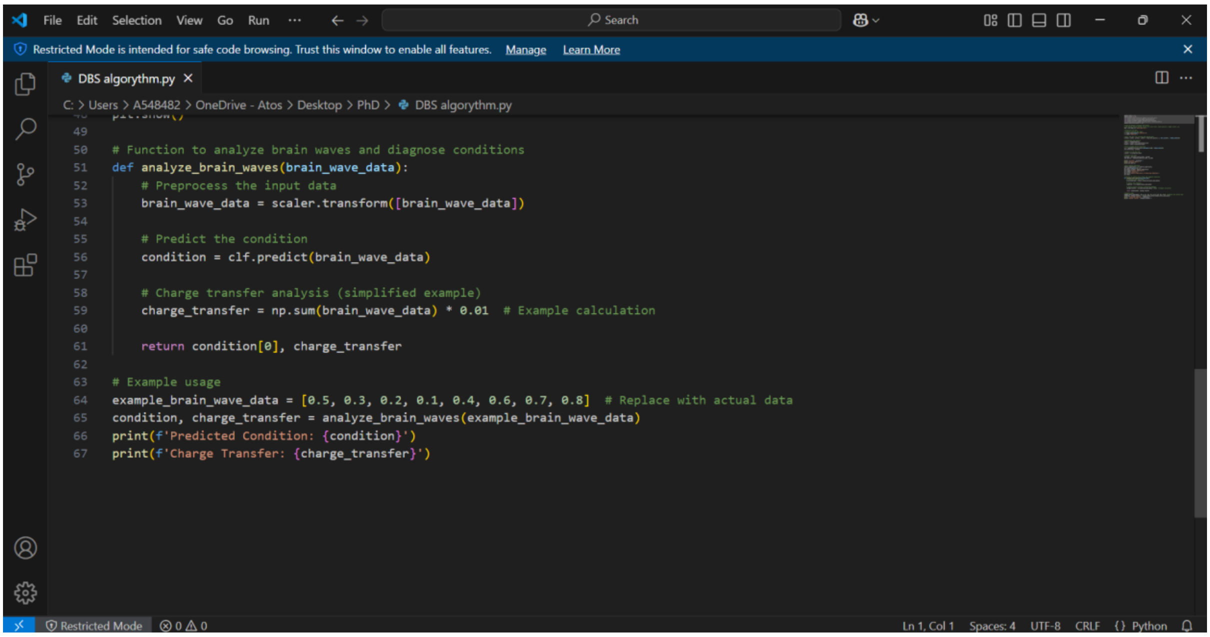
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# Load the dataset (assuming CSV format)
# The dataset should contain columns for brain waves, blood pressure, oxygen levels, etc.
data = pd.read_csv('brain_data.csv')

# Preprocess the data
# Separate features and labels
X = data.drop(columns=['condition'])
y = data['condition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train a Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```python
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)

# Plot feature importance
feature_importance = clf.feature_importances_
plt.figure(figsize=(10, 6))
plt.barh(X.columns, feature_importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance in Predicting Conditions')
plt.show()

# Function to analyze brain waves and diagnose conditions
def analyze_brain_waves(brain_wave_data):
    # Preprocess the input data
    brain_wave_data = scaler.transform([brain_wave_data])

```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```python
plt.show()

# Function to analyze brain waves and diagnose conditions
def analyze_brain_waves(brain_wave_data):
    # Preprocess the input data
    brain_wave_data = scaler.transform([brain_wave_data])

    # Predict the condition
    condition = clf.predict(brain_wave_data)

    # Charge transfer analysis (simplified example)
    charge_transfer = np.sum(brain_wave_data) * 0.01  # Example calculation

    return condition[0], charge_transfer

# Example usage
example_brain_wave_data = [0.5, 0.3, 0.2, 0.1, 0.4, 0.6, 0.7, 0.8]  # Replace with actual data
condition, charge_transfer = analyze_brain_waves(example_brain_wave_data)
print(f'Predicted Condition: {condition}')
print(f'Charge Transfer: {charge_transfer}')
```

Figure 6 Advanced algorithms for DBS

Process steps with algorithms use

1. **Data Loading and Preprocessing**:

    - Load the dataset containing brain wave data, blood pressure, oxygen levels, etc.

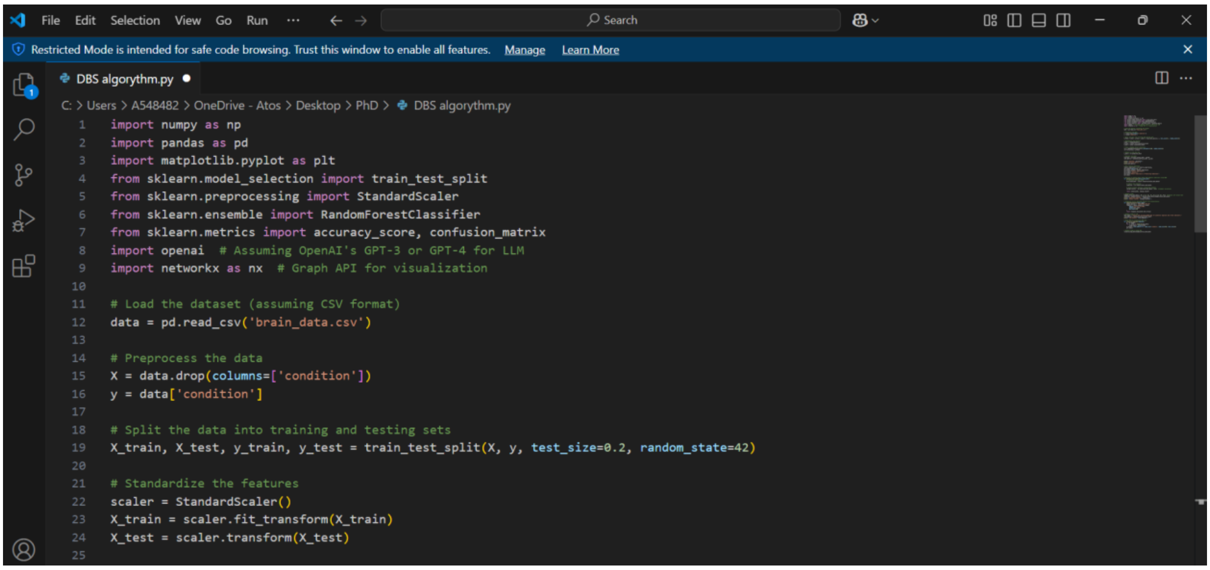
    - Separate features (X) and labels (y).

- Split the data into training and testing sets.
- Standardize the features using StandardScaler.

2. **Model Training**:
    - Train a RandomForestClassifier on the training data.
    - Predict conditions on the test set and evaluate the model using accuracy and confusion matrix.
3. **Feature Importance**:
    - Plot the feature importance to understand which features contribute most to the predictions.
4. **Brain Wave Analysis Function**:
    - Define a function analyze_brain_waves that takes brain wave data as input, preprocesses it, predicts the condition, and performs a simplified charge transfer analysis.

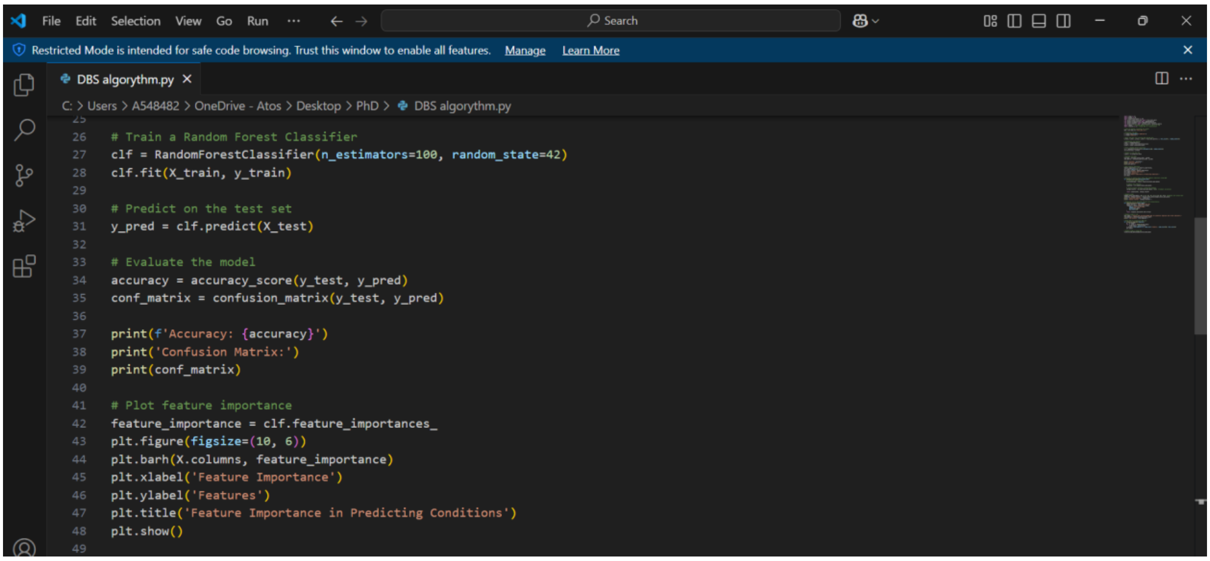
There is an option to include QEEG, also known as quantitative EEG, is often referred to as "brain mapping" because it allows for visual analysis and interpretation of EEG, significantly expanding the understanding of EEG and brain functions.

Python algorithm that integrates a Large Language Model (LLM) and Graph API along with QEEG (quantitative EEG) analysis. This algorithm will use the Graph API to visualize EEG data and leverage LLM for advanced analysis and interpretation I have coded with below steps:


File
Edit
Selection
View
Go
Run
Search
Manage
Learn More
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py

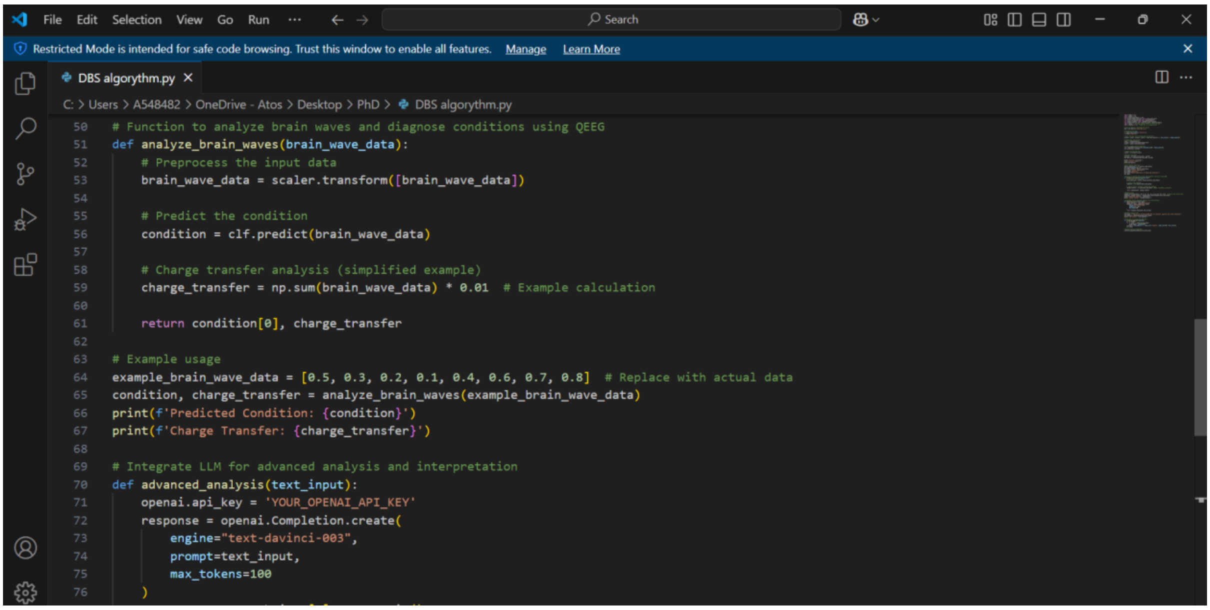
Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import openai  # Assuming OpenAI's GPT-3 or GPT-4 for LLM
import networkx as nx  # Graph API for visualization

# Load the dataset (assuming CSV format)
data = pd.read_csv('brain_data.csv')

# Preprocess the data
X = data.drop(columns=['condition'])
y = data['condition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

```


File
Edit
Selection
View
Go
Run
Search
Manage
Learn More
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py

Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

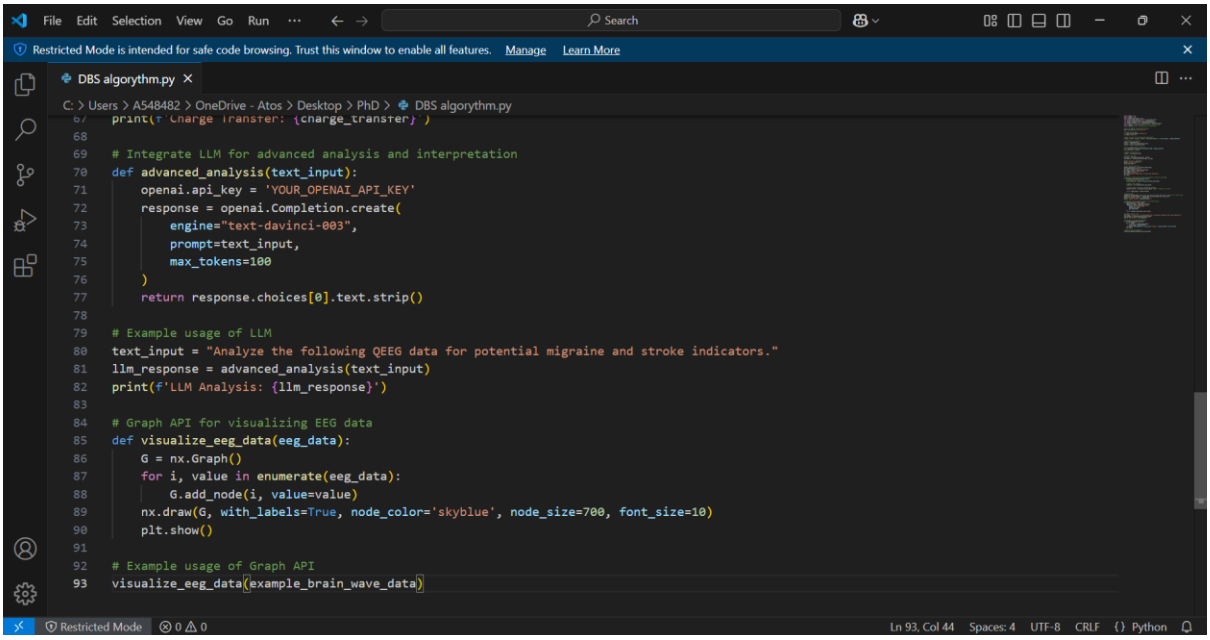
```python
# Train a Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)

# Plot feature importance
feature_importance = clf.feature_importances_
plt.figure(figsize=(10, 6))
plt.barh(X.columns, feature_importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance in Predicting Conditions')
plt.show()

```



File
Edit
Selection
View
Go
Run
Search
Manage
Learn More
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py

Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```python
# Function to analyze brain waves and diagnose conditions using QEEG
def analyze_brain_waves(brain_wave_data):
    # Preprocess the input data
    brain_wave_data = scaler.transform([brain_wave_data])

    # Predict the condition
    condition = clf.predict(brain_wave_data)

    # Charge transfer analysis (simplified example)
    charge_transfer = np.sum(brain_wave_data) * 0.01  # Example calculation

    return condition[0], charge_transfer

# Example usage
example_brain_wave_data = [0.5, 0.3, 0.2, 0.1, 0.4, 0.6, 0.7, 0.8]  # Replace with actual data
condition, charge_transfer = analyze_brain_waves(example_brain_wave_data)
print(f'Predicted Condition: {condition}')
print(f'Charge Transfer: {charge_transfer}')

# Integrate LLM for advanced analysis and interpretation
def advanced_analysis(text_input):
    openai.api_key = 'YOUR_OPENAI_API_KEY'
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=text_input,
        max_tokens=100
    )
```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```python
print(f'Charge Transfer: {charge_transfer}')

# Integrate LLM for advanced analysis and interpretation
def advanced_analysis(text_input):
    openai.api_key = 'YOUR_OPENAI_API_KEY'
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=text_input,
        max_tokens=100
    )
```

```python
    return response.choices[0].text.strip()

# Example usage of LLM
text_input = "Analyze the following QEEG data for potential migraine and stroke indicators."
llm_response = advanced_analysis(text_input)
print(f'LLM Analysis: {llm_response}')

# Graph API for visualizing EEG data
def visualize_eeg_data(eeg_data):
    G = nx.Graph()
    for i, value in enumerate(eeg_data):
        G.add_node(i, value=value)
    nx.draw(G, with_labels=True, node_color='skyblue', node_size=700, font_size=10)
    plt.show()

# Example usage of Graph API
visualize_eeg_data(example_brain_wave_data)
```

Figure 7 Python algorithm that integrates a Large Language Model (LLM) and Graph API along with QEEG (quantitative EEG) analysis

Explanation:

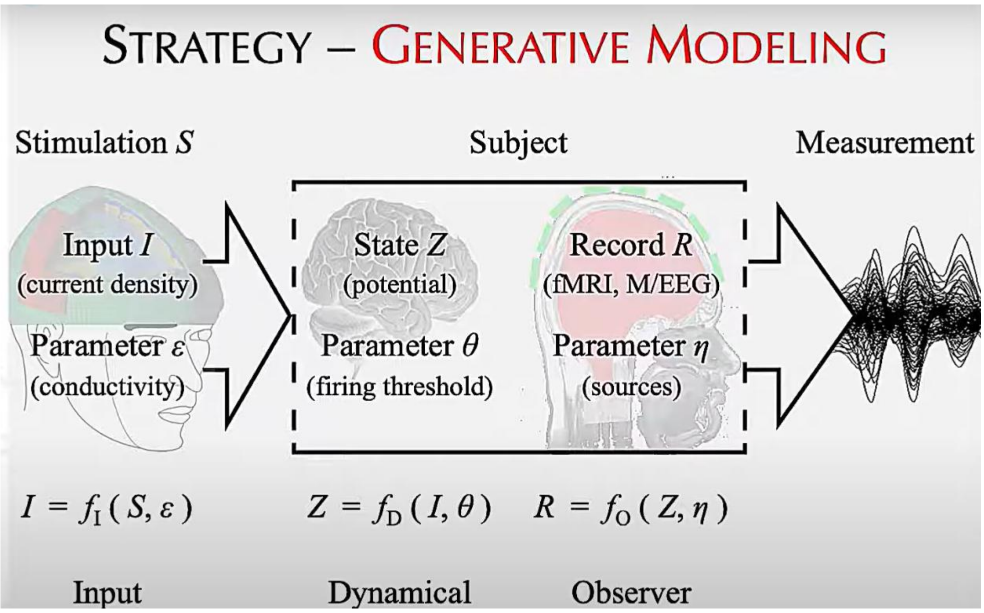
1. **Data Loading and Preprocessing**:

   - Load the dataset containing brain wave data, blood pressure, oxygen levels, etc.
   - Separate features (X) and labels (y).

- Split the data into training and testing sets.
- Standardize the features using StandardScaler.

2. **Model Training**:
    - Train a RandomForestClassifier on the training data.
    - Predict conditions on the test set and evaluate the model using accuracy and confusion matrix.
3. **Feature Importance**:
    - Plot the feature importance to understand which features contribute most to the predictions.
4. **Brain Wave Analysis Function**:
    - Define a function analyze_brain_waves that takes brain wave data as input, preprocesses it, predicts the condition, and performs a simplified charge transfer analysis.
5. **LLM Integration**:
    - Use OpenAI's GPT-3 or GPT-4 for advanced analysis and interpretation of QEEG data.
    - Define a function advanced_analysis to interact with the LLM and get insights.
6. **Graph API (or own- built API) Integration**:
    - Use NetworkX to visualize EEG data as a graph.
    - Define a function visualize_eeg_data to create and display the graph.

This algorithm provides a comprehensive framework for analyzing brain waves, diagnosing conditions, and leveraging AI/ML, LLM, and Graph API for enhanced insights and visualization.



STRATEGY – GENERATIVE MODELING

$I = f_{\mathrm{I}}(S, \varepsilon)$ $Z = f_{\mathrm{D}}(I, \theta)$ $R = f_{\mathrm{O}}(Z, \eta)$

Figure 8 Strategy of Generative Modelling with R

The integration of the mathematical MASS model with R in deep brain stimulation (DBS) encompasses several advanced methodologies, including predictive models and large language models (LLMs). This integration is pivotal for enhancing the precision and efficacy of DBS treatments. Predictive models, developed using R, leverage historical data to forecast future outcomes, thereby enabling the optimization of DBS protocols. The incorporation of LLMs, such as those facilitated by the tidychatmodels package in R, further augments this process by providing sophisticated natural language processing capabilities.

I have used the regression to predict numeric (continuous) y-variables and classification to predict categorical (discrete) y-variables.

Measurements from graphene diodes, neurons, and neural networks are integral to this framework. Graphene diodes, known for their exceptional electrical conductivity and biocompatibility, facilitate accurate recording of neural activity and charge transfer processes. These measurements can be seamlessly integrated into the MASS model, allowing for real-time monitoring and adjustment of DBS parameters. Neural networks, both biological and artificial, play a crucial role in interpreting these measurements and predicting the outcomes of various stimulation regimes.

The Vojta method, a renowned technique in neurological rehabilitation, can be revisited to enhance recovery if DBS proceeds well. This method focuses on stimulating specific reflex points to improve motor function and alleviate symptoms such as migraines with aura. By incorporating the Vojta method into the rehabilitation protocol, it is possible to mitigate the risk of brain strokes and further enhance the therapeutic outcomes of DBS.

Hypothesis: the application of the MASS model with R in DBS involves the integration of predictive models and LLMs, the incorporation of measurements from graphene diodes and neural networks, and the utilization of the Vojta method for rehabilitation. This interdisciplinary approach holds significant promise for advancing the field of neurological treatment and improving patient outcomes.

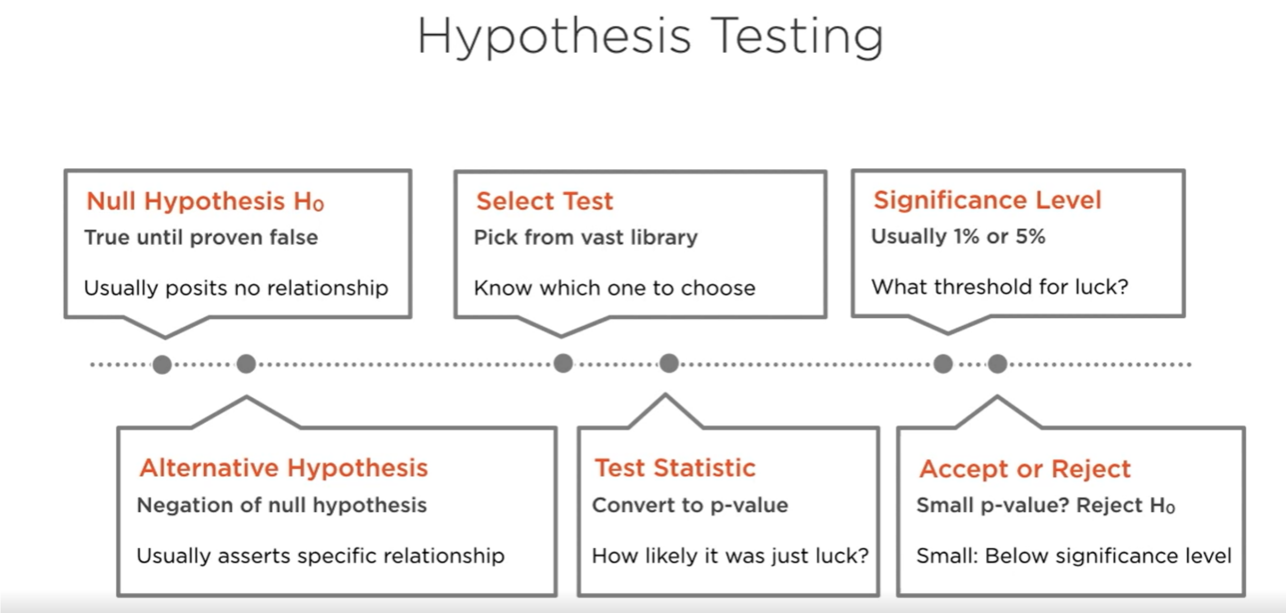


Figure 9 MASS model with R and LLM

Integration of AI, ML and Advanced Algorithms

AI and ML Integration brings together below assumptions:

- **Data Collection and Analysis**: Usage of AI and ML algorithms to collect and analyze patient data, including neurological signals, motor patterns, and treatment outcomes. This data can be used to train models that predict treatment efficacy and personalize therapy.

- **Predictive Modeling**: Development of predictive models to identify optimal stimulation parameters for DBS and to forecast the progression of neurological conditions.
- **Real-time Monitoring**: Implementation of real-time monitoring systems that use AI to adjust stimulation parameters dynamically based on patient responses.

Vojta Method Integration

Vojta Therapy:

- **Reflex Locomotion Activation**: Usage of AI to analyze and optimize the activation of reflex locomotion patterns in patients undergoing Vojta therapy. This can help in fine-tuning the pressure points and stimulation zones for better outcomes
- **Patient Monitoring**: Implementation of sensors and AI algorithms to monitor patient movements and provide feedback to therapists, ensuring precise application of the Vojta method.

Charge Transfer in Deep Brain Stimulation Science and Nanotechnology Diodes

The integration of charge transfer measurements in brain science is crucial for understanding the electrical activity of the brain. Nanotechnology diodes, particularly graphene-based diodes, can facilitate precise charge transfer and enhance the efficiency of DBS. These diodes can collect comprehensive input data from various physiological factors, such as brain activity, neurons, the central nervous system, blood pressure, glucose levels, age, gender, diet, cortisol (stress), muscle tension, spine defects, and other factors. This data can be used to train AI/ML models for early diagnosis and treatment of neurological conditions. Trained models must include all changes happening during day and night observation of patients willing to attend such research and take into account responses given from graphene diodes.

DBS Optimization:

- **Electrode Design**: Utilize advanced algorithms to design electrodes that maximize charge transfer efficiency and minimize tissue damage. This exercise must be treated as continued activity of its calibration with the uncertainty of measurements and observability approach.

Graphene and Graphene Diodes in Enhancing Deep Brain Stimulation and Neuromodulation Techniques

Graphene and graphene diodes possess unique properties that significantly enhance the effectiveness of deep brain stimulation (DBS) and other neuromodulation techniques. Further researches explore the relationship between these materials and the use of upconversion nanoparticles (UCNPs) in near-infrared (NIR) deep brain stimulation. For PhD purposes, I will elaborate on graphene diodes with charge transfer stimulation on the brain.

Electrical Conductivity: Graphene is renowned for its exceptional electrical conductivity, making it an ideal material for fabricating highly efficient electrodes. These electrodes are utilized in DBS to deliver precise electrical stimulation to targeted brain regions.

Flexibility and Biocompatibility: The remarkable thinness and flexibility of graphene allow it to conform to the brain's surface without causing significant damage or discomfort. This property is crucial for developing minimally invasive neural interfaces.

High Charge Injection Capacity: Graphene diodes and electrodes can handle high charge densities, which is essential for effective neural stimulation. This capacity ensures that the electrical signals delivered during DBS are sufficiently strong to modulate neuronal activity without degrading the electrode material.

Optical Properties: Graphene's optical transparency and its ability to interact with light make it compatible with optogenetic techniques. When combined with UCNPs, graphene-based devices can facilitate the conversion of NIR light to visible light, which can then activate light-sensitive ion channels in neurons.

Integration with UCNPs: Graphene can be integrated with UCNPs to create hybrid devices that leverage the strengths of both materials. For instance, graphene's high conductivity can enhance the efficiency of UCNPs in converting NIR light to visible light, thereby improving the overall effectiveness of optogenetic stimulation.

Charge Transfer in Physics: Charge transfer is the process by which electrons move from one atom or molecule to another. This can occur through various mechanisms, including conduction and induction. In conduction, charge transfer happens when a charged particle touches a conductive material, allowing electrons to move between them.

In induction, charge transfer occurs without direct contact, as the electric field of a charged object induces electron movement in another object. These principles are fundamental in understanding how graphene and UCNPs interact at the molecular level to facilitate effective neuromodulation.

Conclusion: Graphene and graphene diodes offer several advantages for neuromodulation, including high electrical conductivity, flexibility, biocompatibility, and compatibility with optical techniques. These properties make them valuable components in advanced DBS systems and other neural stimulation technologies.

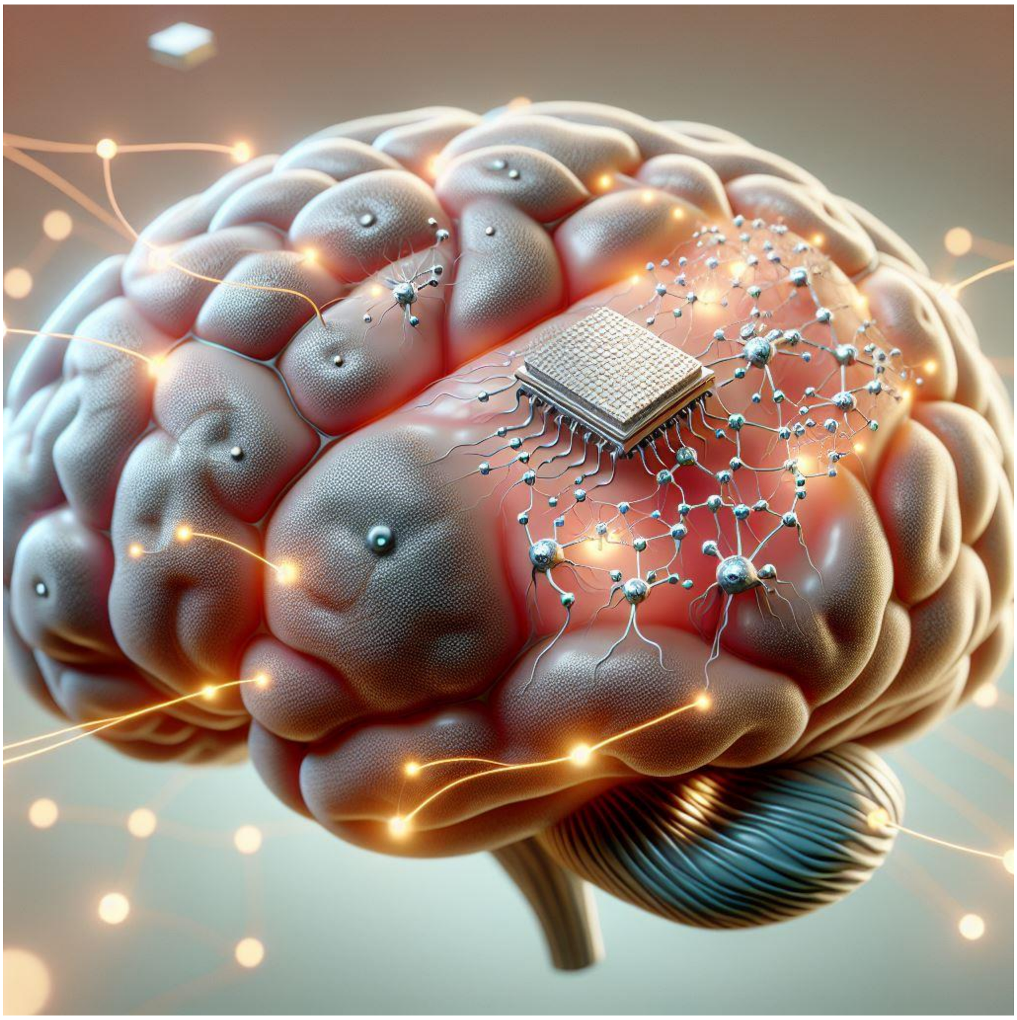


Figure 10 An image illustrating charge transfer and deep brain stimulation with graphene diodes.

The photo shows the interaction between graphene diodes and neural tissue, highlighting the electrical conductivity and biocompatibility of graphene. That includes elements representing the charge transfer process and the effects of deep brain stimulation.

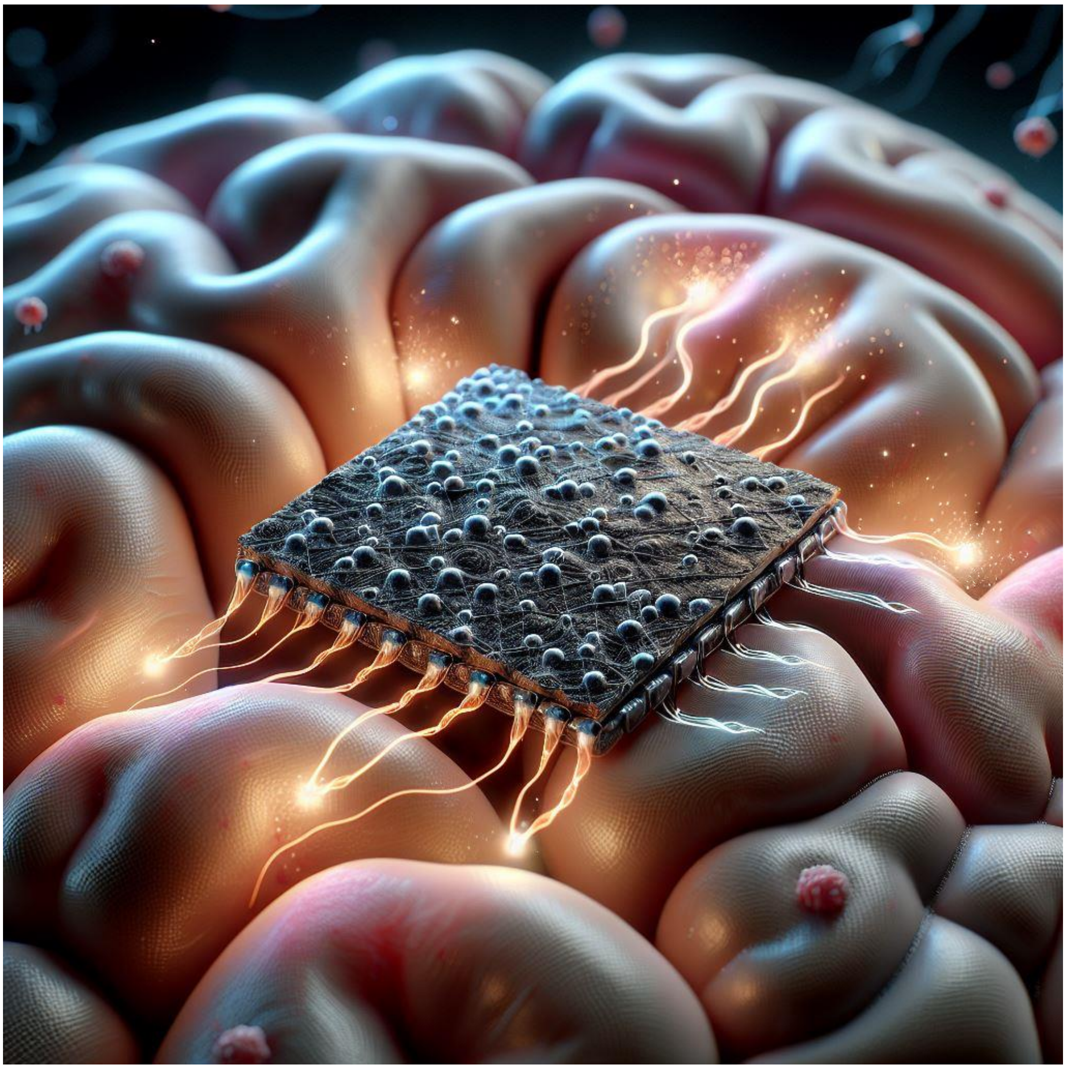


Figure 11 Graphene diodes with charge transfer, AI, genAI, ML, LLM in DBS

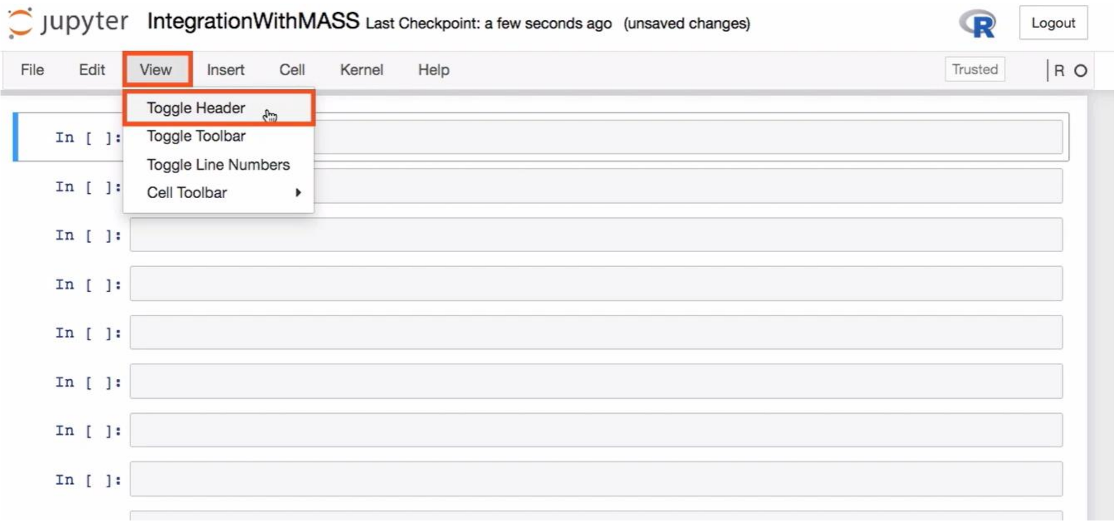
The integration of the mathematical MASS model with R in deep brain stimulation (DBS) encompasses several advanced methodologies, including predictive models and datasets. The MASS package in R, which stands for "Modern Applied Statistics with S," provides functions and datasets to support statistical analysis and modelling.

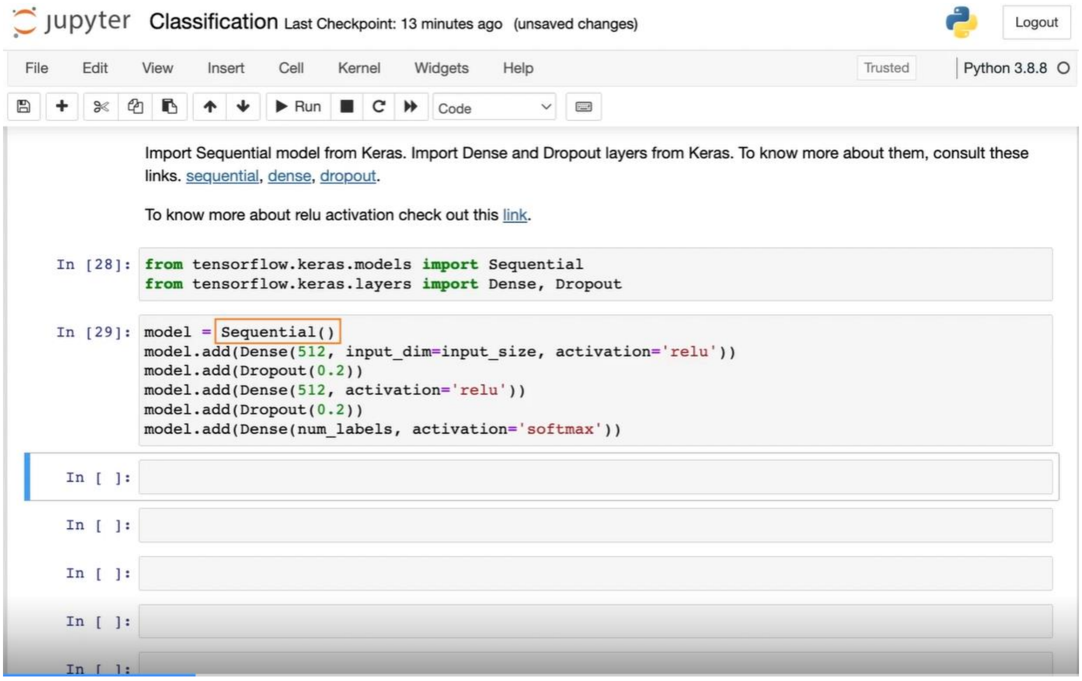
S is a statistical programming language that originated at Bell Labs in 1975-76. It was designed for data analysis and statistical computing. R is the main modern implementation of S and is widely used for statistical analysis and data science. R is free and part of the GNU project, making it accessible to researchers and practitioners worldwide. S-PLUS is a commercial version of S, offering additional features and support.

In the context of DBS, the MASS package in R can be used to develop predictive models that leverage historical data to forecast future outcomes, thereby optimizing DBS protocols. The package includes various functions for statistical analysis, such as linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and robust regression techniques. These methods are essential for interpreting complex datasets and predicting the outcomes of different stimulation regimes.

By integrating the MASS model with R, researchers can enhance the precision and efficacy of DBS treatments, utilizing advanced statistical techniques and comprehensive datasets to inform their decisions and improve patient outcomes. That is just one from many solutions integrated with Neural Networks that could be considered for Deep Brain Stimulation techniques with design introduced by myself.

The Jupiter program, specifically the NeuralCMS model, is a machine-learning model designed to compute the gravitational moments and mass of Jupiter given seven chosen parameters setting its interior model. This model is trained on over a million interior model solutions computed with the accurate but computationally demanding concentric Maclaurin spheroid method (CMS). The NeuralCMS model can be adapted to study neural dynamics and predict outcomes in DBS by leveraging its ability to handle complex computations and large datasets efficiently

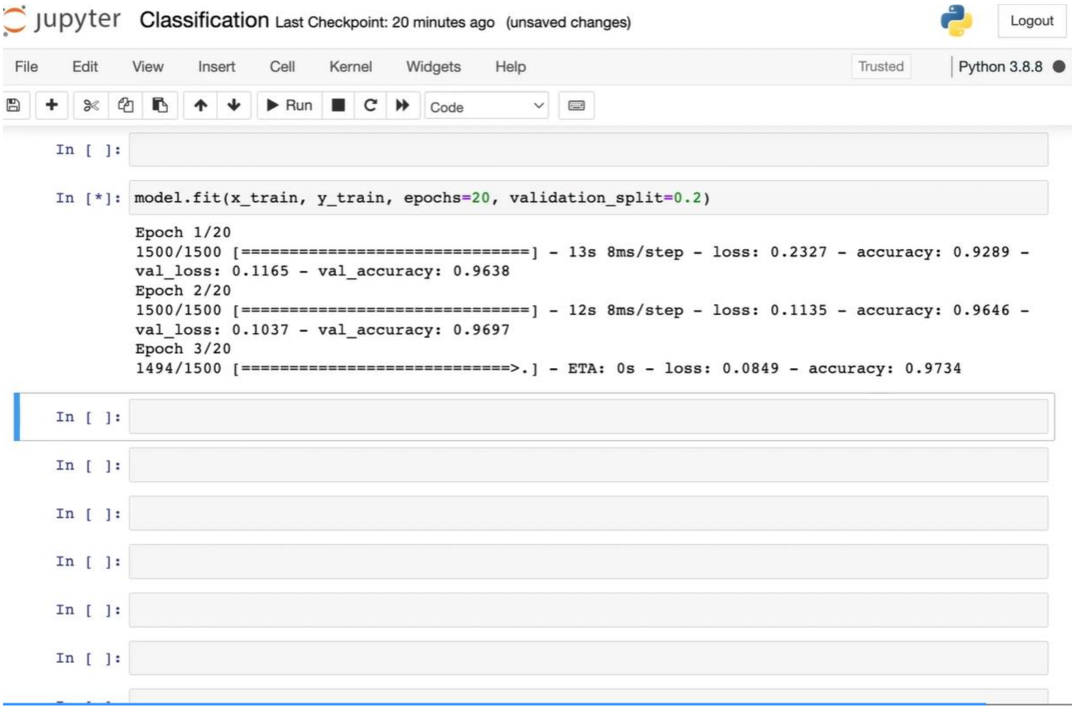



jupyter Classification Last Checkpoint: 13 minutes ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3.8.8
Run
Code

Import Sequential model from Keras. Import Dense and Dropout layers from Keras. To know more about them, consult these links. sequential, dense, dropout.

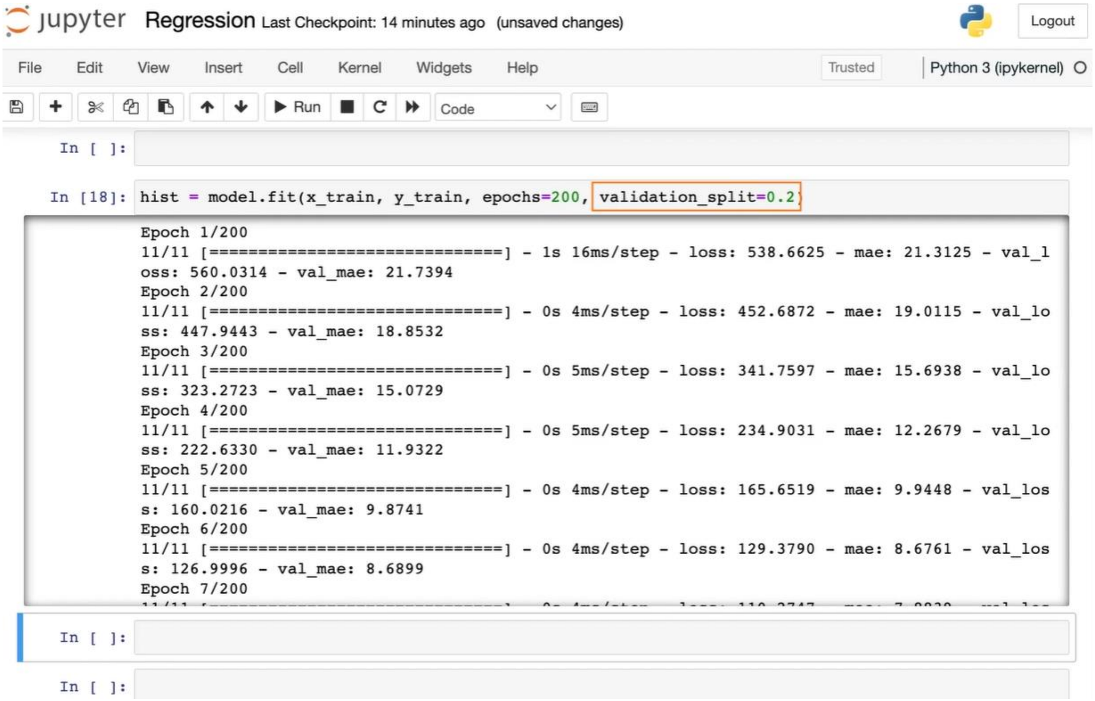
To know more about relu activation check out this link.

```
In [28]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Dropout

In [29]: model = Sequential()
         model.add(Dense(512, input_dim=input_size, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(512, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(num_labels, activation='softmax'))

In [ ]:
```

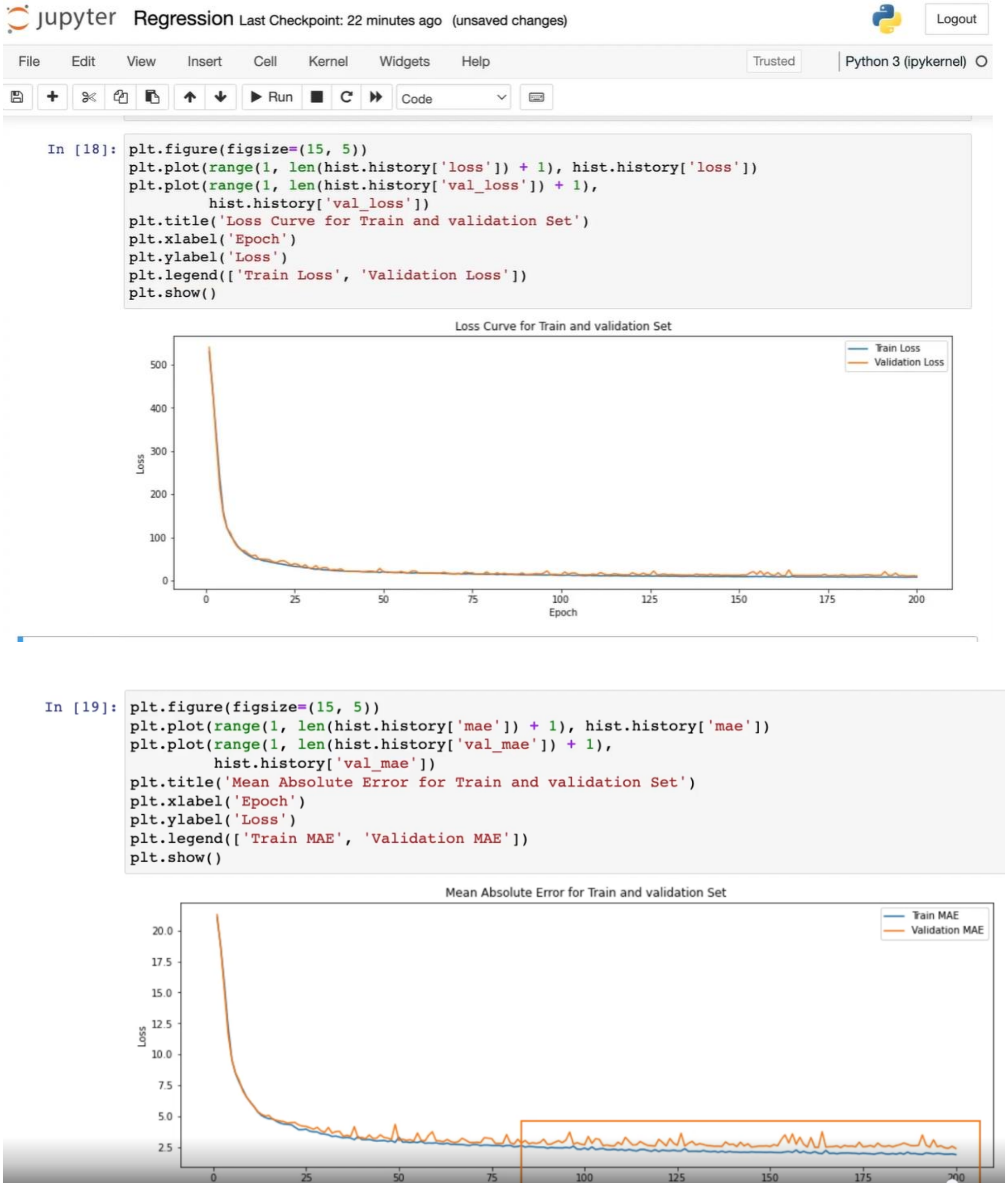


```
In [ ]:

In [*]: model.fit(x_train, y_train, epochs=20, validation_split=0.2)

        Epoch 1/20
        1500/1500 [==============================] - 13s 8ms/step - loss: 0.2327 - accuracy: 0.9289 -
        val_loss: 0.1165 - val_accuracy: 0.9638
        Epoch 2/20
        1500/1500 [==============================] - 12s 8ms/step - loss: 0.1135 - accuracy: 0.9646 -
        val_loss: 0.1037 - val_accuracy: 0.9697
        Epoch 3/20
        1494/1500 [============================>.] - ETA: 0s - loss: 0.0849 - accuracy: 0.9734

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

Upon having the classification completed, time to train the model with use of regression data.



```
In [ ]:

In [18]: hist = model.fit(x_train, y_train, epochs=200, validation_split=0.2)

         Epoch 1/200
         11/11 [==============================] - 1s 16ms/step - loss: 538.6625 - mae: 21.3125 - val_l
         oss: 560.0314 - val_mae: 21.7394
         Epoch 2/200
         11/11 [==============================] - 0s 4ms/step - loss: 452.6872 - mae: 19.0115 - val_lo
         ss: 447.9443 - val_mae: 18.8532
         Epoch 3/200
         11/11 [==============================] - 0s 5ms/step - loss: 341.7597 - mae: 15.6938 - val_lo
         ss: 323.2723 - val_mae: 15.0729
         Epoch 4/200
         11/11 [==============================] - 0s 5ms/step - loss: 234.9031 - mae: 12.2679 - val_lo
         ss: 222.6330 - val_mae: 11.9322
         Epoch 5/200
         11/11 [==============================] - 0s 4ms/step - loss: 165.6519 - mae: 9.9448 - val_los
         s: 160.0216 - val_mae: 9.8741
         Epoch 6/200
         11/11 [==============================] - 0s 4ms/step - loss: 129.3790 - mae: 8.6761 - val_los
         s: 126.9996 - val_mae: 8.6899
         Epoch 7/200

In [ ]:

In [ ]:
```



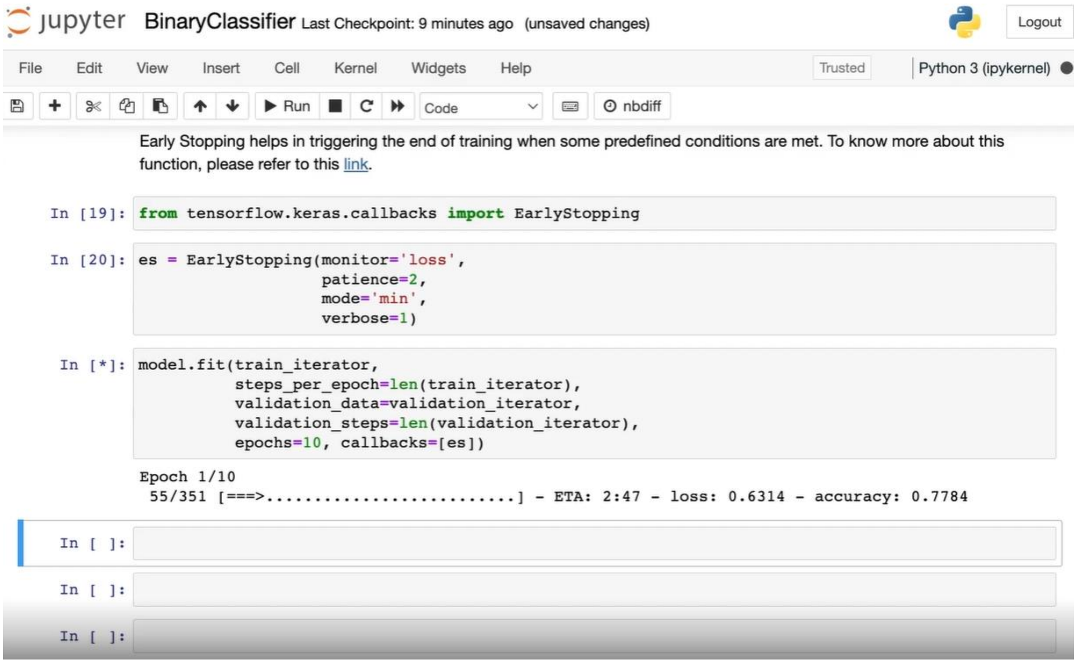
In [18]:

```
plt.figure(figsize=(15, 5))
plt.plot(range(1, len(hist.history['loss']) + 1), hist.history['loss'])
plt.plot(range(1, len(hist.history['val_loss']) + 1),
         hist.history['val_loss'])
plt.title('Loss Curve for Train and validation Set')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train Loss', 'Validation Loss'])
plt.show()
```

In [19]:

```
plt.figure(figsize=(15, 5))
plt.plot(range(1, len(hist.history['mae']) + 1), hist.history['mae'])
plt.plot(range(1, len(hist.history['val_mae']) + 1),
         hist.history['val_mae'])
plt.title('Mean Absolute Error for Train and validation Set')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train MAE', 'Validation MAE'])
plt.show()
```

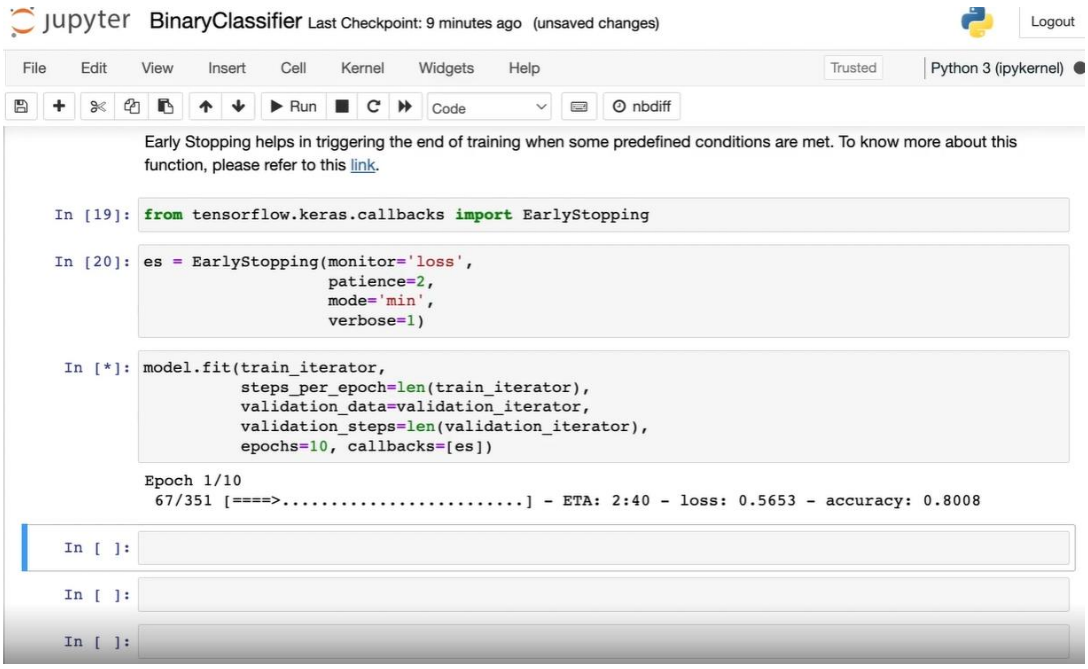
Results of trained model on first patience’s data of brain impulses collected with graphene diodes:


jupyter BinaryClassifier Last Checkpoint: 9 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
nbdiff

Early Stopping helps in triggering the end of training when some predefined conditions are met. To know more about this function, please refer to this link.

```
In [19]: from tensorflow.keras.callbacks import EarlyStopping

In [20]: es = EarlyStopping(monitor='loss',
                            patience=2,
                            mode='min',
                            verbose=1)

In [*]: model.fit(train_iterator,
                  steps_per_epoch=len(train_iterator),
                  validation_data=validation_iterator,
                  validation_steps=len(validation_iterator),
                  epochs=10, callbacks=[es])

         Epoch 1/10
          55/351 [===>..........................] - ETA: 2:47 - loss: 0.6314 - accuracy: 0.7784

In [ ]:

In [ ]:

In [ ]:
```

Monitoring of accuracy, loss, ETA changes over a time:


jupyter BinaryClassifier Last Checkpoint: 9 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
nbdiff

Early Stopping helps in triggering the end of training when some predefined conditions are met. To know more about this function, please refer to this link.

```
In [19]: from tensorflow.keras.callbacks import EarlyStopping

In [20]: es = EarlyStopping(monitor='loss',
                            patience=2,
                            mode='min',
                            verbose=1)

In [*]: model.fit(train_iterator,
                  steps_per_epoch=len(train_iterator),
                  validation_data=validation_iterator,
                  validation_steps=len(validation_iterator),
                  epochs=10, callbacks=[es])

         Epoch 1/10
          67/351 [====>.........................] - ETA: 2:40 - loss: 0.5653 - accuracy: 0.8008

In [ ]:

In [ ]:

In [ ]:
```

In the context of deep learning models training with TensorFlow, patience data plays a crucial role in optimizing the training process. Patience refers to the number of epochs to wait before stopping the training if the model's performance does not improve. This concept is particularly important when using early stopping techniques to prevent overfitting and save computational resources.

ETA (Estimated Time of Arrival): ETA is an important metric that indicates the expected time remaining for the training process to complete. It helps researchers and practitioners manage their time and resources effectively. The ETA can be influenced by various factors, including the complexity of the model, the size of the dataset, and the computational power available.

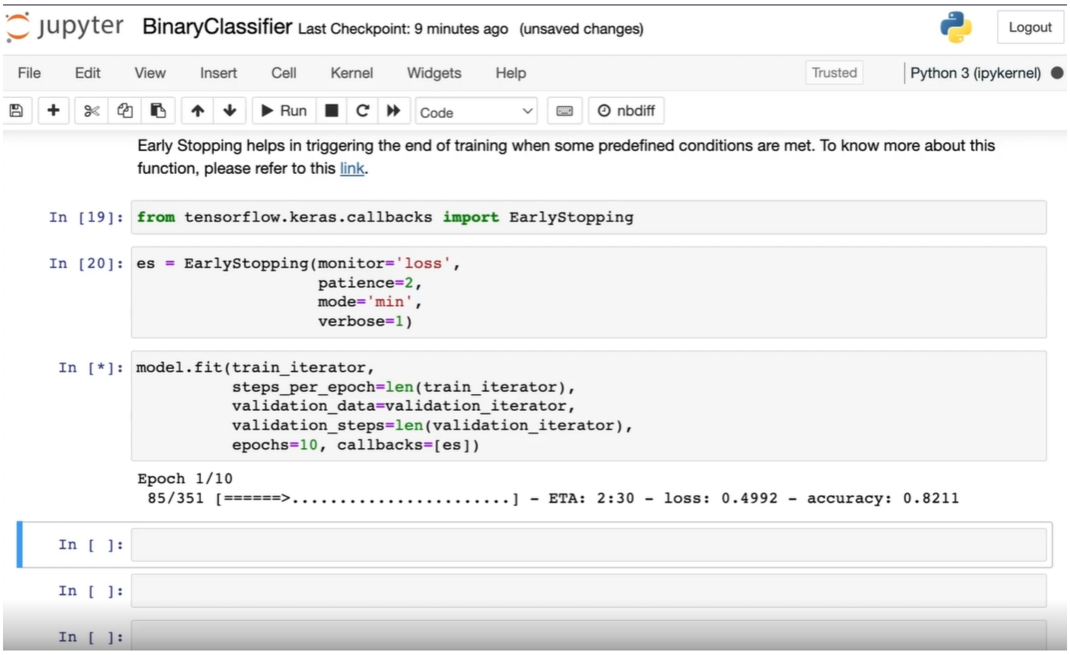
Loss: Loss is a measure of how well the model's predictions match the actual data. It is calculated using a loss function, which quantifies the difference between the predicted values and the true values. During training, the goal is to minimize the loss, indicating that the model is improving its predictions. Changes in loss values can provide insights into the model's learning progress and help identify potential issues such as overfitting or underfitting.

Accuracy: Accuracy is a metric that measures the proportion of correct predictions made by the model. It is often used in classification tasks to evaluate the model's performance. Higher accuracy values indicate better performance. Monitoring accuracy during training helps assess the model's ability to generalize to new data and provides a benchmark for comparing different models.

Variables Influencing Loss and Accuracy: Several factors can influence the loss and accuracy values during training, including:

- **Learning Rate**: The learning rate determines the size of the steps taken during gradient descent. A higher learning rate can speed up training but may lead to instability, while a lower learning rate can result in slower convergence.
- **Batch Size**: The batch size refers to the number of samples processed before updating the model's parameters. Larger batch sizes can lead to more stable training but require more memory, while smaller batch sizes can introduce more noise into the training process.
- **Model Architecture**: The complexity and design of the model, including the number of layers and neurons, can significantly impact the training process and the resulting loss and accuracy values.

- **Regularization Techniques**: Techniques such as dropout, L1/L2 regularization, and early stopping can help prevent overfitting and improve the model's generalization capabilities.
- **Data Quality and Preprocessing**: The quality and preprocessing of the data, including normalization, augmentation, and handling missing values, can affect the model's performance and the stability of the training process.



Early Stopping helps in triggering the end of training when some predefined conditions are met. To know more about this function, please refer to this link.

```
In [19]: from tensorflow.keras.callbacks import EarlyStopping

In [20]: es = EarlyStopping(monitor='loss',
                            patience=2,
                            mode='min',
                            verbose=1)

In [*]: model.fit(train_iterator,
                  steps_per_epoch=len(train_iterator),
                  validation_data=validation_iterator,
                  validation_steps=len(validation_iterator),
                  epochs=10, callbacks=[es])

         Epoch 1/10
          85/351 [======>.......................] - ETA: 2:30 - loss: 0.4992 - accuracy: 0.8211

In [ ]:

In [ ]:

In [ ]:
```

Figure 12 A series of screenshots from the Jupyter environment, showcasing the Python BinaryClassifier with the MASS model, adjusted for the purpose of training learning models using TensorFlow.

Integration of Data from Graphene Diodes: In the context of deep brain stimulation (DBS), data from graphene diodes that monitor brain activity and symptoms of brain stroke and migraine with aura can be integrated into the training process. Graphene diodes provide real-time measurements of neural activity due to their exceptional electrical conductivity and biocompatibility. These measurements can be used to create comprehensive datasets that capture the dynamic changes in brain activity and symptoms. By incorporating this data into the training process, the model can learn to predict and respond to changes in neural activity, enhancing the precision and efficacy of DBS treatments.

Patience data, ETA, loss, and accuracy are critical metrics in the training of deep learning models with TensorFlow. Understanding and monitoring these metrics, along with the variables influencing them, can help optimize the training process and improve the model's performance. The integration of real-time data from graphene diodes further enhances the model's ability to monitor and respond to neural activity, providing valuable insights for DBS treatments.

For this PhD Thesis, I have modified my own models and application of my art that is a subject of patent, thus only on highlights introduced. Started point was to modify and adjust neural model, thus selected this example: https://github.com/zivmaaya/NeuralCMS

The Mechanism of Charge Transfer in Physics, Including Conduction and Induction, can be related to the Occurrence of Strokes and Migraines with Aura Through Several Key Aspects

Brain's stroke

1. **Neurotransmitters and Ion Flow**: In stroke, particularly ischemic stroke, blood flow to the brain is disrupted, leading to a deficiency of oxygen and glucose. This causes disturbances in ion flow, including sodium, potassium, and calcium, which are crucial for neuron function. Charge transfer in neurons is essential for maintaining membrane potential and transmitting nerve signals. Disruptions in this process can lead to cell death and brain damage.
2. **Cerebral Autoregulation**: The mechanisms of cerebral autoregulation, which maintain constant blood flow despite changes in blood pressure, can be disrupted during a stroke. Charge transfer in endothelial cells of blood vessels is key to regulating vascular tone and blood flow.

Migraine with Aura

1. **Cortical Spreading Depression**: Migraine with aura is often associated with a phenomenon called cortical spreading depression, a wave of depolarization of neurons that moves across the cerebral cortex. Charge transfer, particularly ion flow of sodium and calcium, plays a crucial role in this process. Cortical spreading depression can lead

to changes in blood flow and activation of the trigeminovascular system, which is associated with migraine pain.

2. **Visual Cortex Reactivity**: Patients with migraine with aura exhibit increased reactivity of the visual cortex to visual stimuli. Charge transfer in neurons of the visual cortex may be more intense, leading to excessive activation and the occurrence of aura symptoms such as flashes of light and visual disturbances.

Integrating graphene with AI/ML in electrode design for Deep Brain Stimulation (DBS) involves leveraging the unique properties of graphene and the predictive capabilities of AI/ML algorithms to enhance charge transfer efficiency and minimize tissue damage. Here's a detailed approach:

Graphene Properties

Graphene is a single layer of carbon atoms arranged in a hexagonal lattice, known for its exceptional electrical conductivity, mechanical strength, and biocompatibility. These properties make it an ideal material for electrodes (internally situated in the brain's cortex zones or externally sustained on a head) in medical applications.

AI/ML Integration in Electrode Design

Material Optimization

- Defect & uncertaintity of measurements control: AI algorithms can be used to analyze and control the number of defects in graphene sheets and uncertaintity of measurements of brain abnormal functions in the observability model, ensuring optimal electrical conductivity and mechanical stability.
- Composite Formation: AI can help in designing graphene composites with other materials to enhance specific properties like conductivity and biocompatibility.

Electrode Geometry

- Non-stacking and 3D Structures: AI can optimize the design of non-stacking and three-dimensional graphene structures to maximize the surface area and improve charge transfer efficiency.

- High Packing Density: Utilize ML to design electrodes with high packing density, ensuring efficient charge transfer and minimizing tissue damage.

Predictive Modeling

- Simulation and Modeling: Use ML algorithms to simulate the electrode-tissue interface and predict the volume of tissue activated (VTA) during DBS3. This helps in optimizing electrode placement and stimulation parameters.
- Real-time Adjustments: Implement AI-driven systems that can adjust stimulation parameters in real-time based on patient feedback and neurological signals.

Graphene Functionalization

- Heteroatom Doping: AI can assist in functionalizing graphene with heteroatoms like oxygen, nitrogen, boron, or phosphorus to enhance its electrochemical properties.
- Surface Modification: ML can be used to design surface modifications that improve the biocompatibility and reduce the inflammatory response of graphene electrodes.

Practical process steps

- **Data Collection:** Gather extensive data on patient responses to DBS, including neurological signals and treatment outcomes. Create hypothesis and algorythms responsible for uncertainty of measurements and false-positive signals detection.
- **Algorithm Development:** Develop AI/ML algorithms to analyze this data and optimize electrode design and stimulation parameters.
- **Prototype Testing:** Create prototypes of graphene-based electrodes and test them in clinical settings to validate their performance.
- **Continuous Improvement:** Use feedback from clinical trials to continuously improve the AI/ML models and electrode designs.

Benefits

- **Enhanced Efficacy:** Improved charge transfer efficiency leads to more effective DBS treatments.
- **Reduced Tissue Damage:** Optimized electrode designs minimize the risk of tissue damage and inflammation.

- **Personalized Therapy:** AI-driven adjustments ensure that each patient receives personalized treatment based on their specific needs.

Stimulation Parameters: AI can be used to optimize stimulation parameters such as pulse width, frequency, and amplitude for individual patients that influence the overall brain operability.

Graphene Electrodes: Incorporate graphene-based electrodes to enhance electrical conductivity and reduce the size of the implants, improving patient comfort and treatment efficacy.

Conclusion

The mechanisms of charge transfer in physics, including conduction and induction, are fundamental to the functioning of neurons and endothelial cells of blood vessels. Disruptions in these processes can lead to serious consequences such as stroke and migraine with aura. Understanding these mechanisms can help develop more effective methods for treating and preventing these conditions.

Neurological Rehabilitation

Rehabilitation Programs:

- **Personalized Therapy**: Develop personalized rehabilitation programs using AI to tailor exercises and stimulation protocols based on patient progress and specific needs.
- **Feedback Systems**: Implement feedback systems (often referred to feedback loops process) that use sensors and AI to track patient progress and adjust therapy in real-time.

Migraine with aura treatment

Neuromodulation Devices:

- **Device Optimization**: AI can be used to optimize the settings of neuromodulation devices like transcutaneous vagus nerve stimulators and transcranial magnetic stimulation devices for migraine treatment.
- **Patient Data Analysis**: Analyze patient data to identify patterns and triggers for migraines, allowing for more targeted and effective treatments. Take into researches all

factors, like: age, weight, style of working over last 10-20 years, working in so called unhealthy conditions, diseases, hormones researches, brain topography with focus on nerves and veins, blood pressure, stress resistance of patience, smoking, drugs, widow's hump presence, introverted person vs. extraverted, diet, emotions, among many other factors. GenAI model must be feeded with all scenarios that will lead to precision, accuracy, repeatability and reproducibility of results per group classification, following regression, distractors elimination and included artefacts of uncertainty of data process of gathering, analyzing, reproducing and utilizing.

Recovery of Preterm Infants and Seniors

Monitoring and Support:

- **Vital Signs Monitoring**: AI can be used to monitor vital signs and neurological signals in preterm infants and seniors, providing early detection of potential issues.
- **Therapeutic Interventions**: Develop AI-driven therapeutic interventions that can be adjusted based on real-time data to support recovery and development.

Process of implementation steps

1. **Data Infrastructure**: Establish a robust data infrastructure to collect, store, and process patient data securely.
2. **Algorithm Development**: Develop and train AI and ML algorithms using historical and real-time patient data.
3. **Device Integration**: Integrate AI and ML algorithms with medical devices and sensors to enable real-time monitoring and adjustments.
4. **Clinical Trials**: Conduct clinical trials to validate the efficacy and safety of the integrated solutions.
5. **Regulatory Compliance**: Ensure all solutions comply with medical regulations and standards.

Herein proposal for PhD research topics focusing on the comparison of early-born infants and seniors in deep brain stimulation, with a priority on the use of Artificial Intelligence, Large Language Models, Machine Learning, Cybersecurity, graphene, the Vojta method, charge transfer for neurons, and nanotechnology to combat migraines with aura, micro strokes, strokes, and blood flow to the brain with below assumptions:

Graphene diodes for DBS

- Graphene is a promising bionanomaterial due to its superior solubility, conductivity, scalability, and biocompatibility.
- Distinctive characteristics of graphene nanomaterials can be directly applicable to the brain in health and disease.
- Future models with graphene such as injectable, biodegradable, and implantable graphene are suggested for clinical use.

Its recent contribution to neurotechnology is particularly noteworthy because its superior conductivity, chemical resilience, biocompatibility, thermal stability, and scalable nature make it well-suited for measuring brain activity and plasticity in health and disease. I find that graphene-mediated compounds are microfabricated in two central methods: chemical processes with natural graphite and chemical vapor deposition of graphene in a film form. They are widely used as biosensors and bioelectronics for neurodiagnostic and neurotherapeutic purposes in several brain disorders, such as Parkinson's disease, stroke, migraine with aura, glioma, epilepsy, tinnitus, and Alzheimer's disease.

As highlighted before - my researches are focused on brain stroke and migraine with aura.

This review provides an overview of studies that have demonstrated the technical advances of graphene nanomaterials in neuroscientific and clinical applications. I also discuss current limitations and future demands in relation to the clinical application of graphene, highlighting its potential technological and clinical significance for treating brain disorders. My review underscores the potential of graphene nanomaterials as powerful tools for advancing the understanding of the brain and developing new therapeutic strategies.

Observing Properties of Graphene at Home

To observe the properties of graphene at home, I can try the following simple experiments:

Electrical Conductivity Experiment:

Materials Needed: Pencil (graphite), paper, 9-volt battery, LED, insulated wire leads, 330-ohm resistor.

Procedure:

- Draw a thick, dark box on the paper using the pencil. This box will act as a conductor.
- Connect the positive terminal of the battery to one side of the resistor.
- Connect the resistor to one end of the LED.
- Connect the other end of the LED to an insulated lead.
- Attach a lead to the negative terminal of the battery.
- Touch the ends of the leads to the ends of the graphite box and observe if the LED lights up.

Optical Absorbance Test:

Materials Needed: Glass slide, graphene sample.

Procedure:

- Place the graphene sample on the glass slide.
- Observe the sample under a light source to check for the characteristic 2.3% broadband optical absorbance of single-layer graphene

Such simple experiment executed at home conditions, still without advanced labs expertise already provides real example data based on facts of values in electrical and energy consumption of graphene in conduction of electricity and use as organic material.

The Vojta Method in Nervous System Rehabilitation: The Vojta method is a neurological rehabilitation technique that involves stimulating specific points on the patient's body to evoke proper movement patterns. It is particularly effective in infants and children with central nervous system damage.

Rehabilitation using the Vojta Method for infants and Seniors

Czech pediatric neurologist, Professor Vaclav Vojta, spent nearly 20 years of his life developing the concept of neurophysiological rehabilitation for children with central nervous system damage, which is still successfully used in many leading rehabilitation centers worldwide. It is very important to understand the principles and assumptions of rehabilitation using the Vojta method and to start it at the appropriate time in the child's life to achieve the desired effects.

Vojta Method – Who is the rehabilitation using the Vojta method intended for?

As mentioned earlier, motor rehabilitation using the Vojta method is aimed at infants, children and seniors with central nervous system damage. It is particularly indicated in cases of risk of cerebral palsy. It is also used in the treatment of paresis, orthopedic posture defects, spina bifida, scoliosis, hydrocephalus, Down syndrome, or central coordination disorders (CCD).

My twin children experienced four instances of brain haemorrhage and two strokes while they were on intensive therapy in incubators. During this time, I increasingly suffered from migraines with aura due to stress, lack of sleep, and vast concern for the lives of my children, who were at high risk. During the rehabilitation of my children using the Vojta method, I noticed that applying similar techniques from this method reduced my own pain and sometimes, if implemented early, prevented the development of migraines with aura. This was due to the alleviation of cervical lordosis, shoulder girdle tension, significant stress related to the fear for my twins' lives, and chronic long-term sleep deprivation. For my twins, week by week, the parameters of their brain examinations through transcranial ultrasound confirmed the healing process of their brains—almost simultaneously, as is typical for twins. Additionally, we alleviated muscle tension so that, according to their age, they could turn from side to side and eventually walk.

The observations lasted over the time of my children's healing process at hospitals and early rehabilitation. My frequently recurring migraines with aura became a serious bottleneck in everyday care for my twins at the hospital. I lived with my children at the hospital for many months, feeding them with my own milk, which I had to prepare and express from my breasts. Early-born infants must first undergo other types of rehabilitation before they learn to suck milk from their mother's breast. I kangarooed them, stayed with them, and talked to them day and night. I truly recommend parents to bring this invaluable effort for their own children as the process of recovery is vastly sped up. This part of the spiritual experience I will dedicate to next book. In this thesis, I focus solely on scientific aspects, while leaving a direct understanding that the soul, feelings, and emotions of children and parents in the healing process are priceless,

The Vojta method is also addressed to premature infants if they need rehabilitation. The Vojta method can be used even when the child is still in the incubator. The earlier the therapy is started, the faster the effects will be achieved, that I can confirm by results achieved in case of my twins treatment.

Vojta Method – What are the main assumptions of rehabilitation using the Vojta method?

Vojta's main assumption was that every child is born with a programmed development pattern in the brain, according to which their motor development will proceed. Vojta later called this motor ontogenesis. It is somewhat "built-in" in every human being from birth.

However, he noticed that damage to the nervous system results in the mentioned program not being able to be properly implemented. Consequently, the child does not have access to the correct movement pattern and begins to exhibit incorrect patterns. Over time, these patterns become more entrenched and result in motor problems for the child.

Vojta discovered that appropriate stimulation of certain points on the child's body triggers specific, correct motor reactions. The conclusion was that the child's muscles were not damaged, but the center controlling them, i.e., the brain. A child with a damaged nervous system has a blockade that prevents them from developing proper movement patterns, which, according to Vojta's assumptions, are programmed from birth.

The premise of the Vojta method is the existence of a genetically encoded global locomotion pattern. This means that all mechanisms necessary for changing body position, standing up, and moving forward exist from the moment of conception of the human organism and gradually reveal themselves automatically as the central nervous system matures.

Vojta concluded that through appropriate and systematic stimulation, it is possible for the child to encode the correct movement patterns in the brain. These, once encoded, can be used in the future as correct movement patterns.

These mechanisms fall into three groups:

- support and extension mechanisms,
- automatic control of body posture,
- purposeful phased movement.

The diagnostic part consists of:

1. Movement analysis, including the assessment of the child's spontaneous motor skills in the supine and prone positions – quantitative and qualitative assessment.
2. Seven positional reactions of the body in space and their assessment.
3. Assessment of selected primitive reflexes.

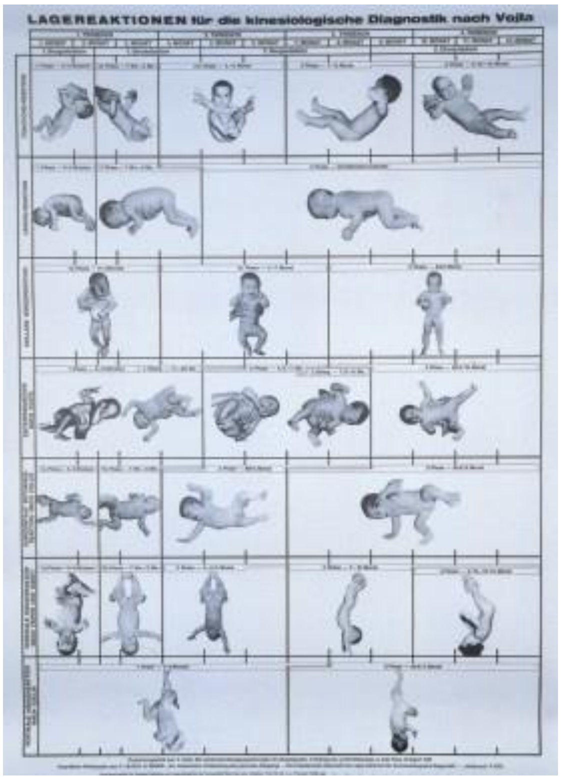

LAGEREAKTIONEN für die kinesiologische Diagnostik nach Vojta

Figure 13 Stimulation points of the body using the Vojta method affecting feedback and response of the central nervous system in the healing process.

Movement analysis of spontaneous motor skills allows us to determine the developmental stage of an infant without knowing their chronological age and, importantly according to Vojta, to assess the quality of the patterns they possess.

The assessment of positional reactions is a very sensitive method for evaluating so-called postural maturity. It allows us to determine the stage and quality of postural control by the central nervous system from the first days of a newborn's life. In the following months, it enables monitoring the direction of development and the potential risk of pathological development (e.g., cerebral palsy).

For early diagnostics, Vojta introduced the concept of central coordination disorder (in Polish literature, central nervous coordination disorder, abbreviated as ZOKN). This is not a disease entity but a temporary term resulting from the neurokinesiological examination of an infant. Depending on the number of abnormal positional reactions found during the examination, we distinguish:

1. Mild ZOKN – 1-3 abnormal positional reactions.
2. Moderate ZOKN – 4-5 abnormal positional reactions.
3. Moderately severe ZOKN – 6-7 abnormal positional reactions.
4. Severe ZOKN – 7 abnormal positional reactions with concurrent severe muscle tone disorder.

This examination is complemented by the assessment of so-called primitive reflexes, which more precisely allows us to determine the likelihood of such a problem occurring.

Vojta's neurokinesiological diagnostics is a quick, sensitive, and cost-effective method that allows for the selection of infants who do not fully achieve the correct locomotion program, enabling the earliest possible start of therapy. If the neurokinesiological examination of the child's spontaneous motor skills, positional reactions, and primitive reflexes shows deviations from normality, the earliest possible application of the reflex locomotion method can prevent pathological development or at least minimize its effects.

The result of such action can be normal development, and if pathological development (such as cerebral palsy) occurs, its extent and effects will be as minimal as possible. The necessity for the earliest possible introduction of therapy is dictated by the greatest brain plasticity in the first months of a child's life. The best period to start is the first three months of life, at the latest by the end of the fifth month. The first three months are crucial because, under normal conditions, physiological spinal extension is formed during this period. However, it is important to remember that Vojta therapy cannot create something new! The possibilities and limits of therapy will depend on the time of the child's life when the central nervous system was damaged, its location, extent, and the possible occurrence of congenital diseases or brain defects. We can only influence the undamaged areas of the brain, mobilizing them to take over the functions of the damaged parts.

Vojta developed a therapy method based on reflex activation and triggering genetically encoded locomotion mechanisms, known as reflex locomotion. It consists of two artificial movement patterns: reflex crawling and reflex turning. Since this therapy uses deep sensation receptors, or proprioceptors, it belongs to the group of neurophysiological methods. Therefore, its application requires adherence to the principle of painlessness. The arrangement of these receptors, resulting from the appropriate starting position and stimulation of the so-called stimulation zones and application of appropriate resistance, allows for reaching and activating the motor development matrix.

The key to successful rehabilitation using the Vojta method is the PARENTS, who are responsible for its systematic implementation. Parents work with their child under the supervision of a physiotherapist. In the initial period of rehabilitation, visits to the physiotherapist's office should be frequent, every 2-3 days, gradually becoming less frequent as parents gain confidence in conducting the therapy, eventually extending to 2, 3, or 4 weeks. Solving technical problems of exercises is the responsibility of the physiotherapist. To achieve the maximum effect, it is essential for parents to be aware of the existing problem in their child's development and convinced of the necessity to start and conduct such therapy. Vojta therapy is not easy. It requires long-term training for the physiotherapist. For the child, it involves undertaking significant physical effort comparable to the load in competitive sports, and for the parents, it requires systematic enforcement despite frequent dissatisfaction from the child.

Vojta Method – How does rehabilitation using the Vojta method proceed?

To achieve the desired effects, rehabilitation using the Vojta method must be started as early as possible. Therefore, it is recommended for premature infants and children in the first trimester of life at risk of, for example, cerebral palsy.

Depending on the condition, an individual stimulation program is arranged by the therapist. One rehabilitation session lasts from a few to several minutes, depending on the child's age and condition. Sessions should be held four times a day.

Rehabilitation using the Vojta method involves pressing specific points on the child's body. It is worth noting that the child must be naked during the sessions so that clothing does not restrict their movements and does not hinder their perception of their own body. Additionally, the lack of clothing allows the therapist to assess how the child's muscles work and how they respond to stimulation. Rehabilitation using the Vojta method is not painful. It should not stimulate the child's pain receptors, as it would then become ineffective.

Vojta Method – What is the role of parents in the rehabilitation process?

Parents play a huge role in the rehabilitation process using the Vojta method. Rehabilitation must be carried out four times a day. This is a task for parents, who are previously properly trained and prepared by the therapist.

Sessions with a specialist usually take place weekly. During these sessions, the therapist checks the child's or senior’s progress and ensures that everything is proceeding correctly. It is also a time for parents' questions and doubts, as they must find themselves in the new role of their child's therapist.

When parents become more confident in conducting rehabilitation, meetings can be held less frequently. Additionally, parents should be aware of how important the atmosphere in which daily exercises are performed is, alongside properly conducted stimulation.

Vojta Method – Is rehabilitation using the Vojta method effective?

The effectiveness of rehabilitation using the Vojta method has been confirmed multiple times in independent studies. The Vojta method, applied early enough, as research shows, can protect a greater number of children from serious complications in the form of cerebral palsy than other methods. In many cases, it is possible to achieve full normalization of the child's development.

How did I integrate the Vojta method with graphene diodes, charge transfer with neural networks, deep brain stimulation, AI/ML/LLM, and cybersecurity?

I am absolutely convinced that stimulation using the Vojta method combined with stimulation using graphene diodes with AI/ML software and charge transfer supports the treatment of brain hemorrhages, strokes in premature infants, as well as seniors with the same conditions. Observations resulting from Deep Brain Stimulation (DBS) with AI/ML using graphene diodes, Vojta method, and charge transfer in premature infants can then be replicated with adjusted parameters in seniors with brain strokes, migraines with aura, and other neurological diseases.

Charge Transfer Transitions: Charge transfer transitions involve the movement of an electron from one molecule (donor) to another (acceptor), which is crucial in many chemical reactions and biological processes. These mechanisms are significant in materials science and nanotechnology research.

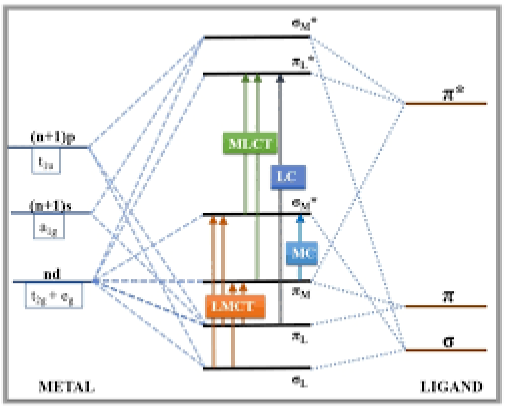


Figure 14 Transition of charge transfer mechanism.

Graph: Possible electronic transitions in a typical octahedral complex. [LMCT- Ligand to metal charge transfer; MLCT- Metal to ligand charge transfer; MC–Metal centered transition; LC–Ligand centered transition].

Electronic transitions in octahedral complexes, such as LMCT (Ligand to Metal Charge Transfer), MLCT (Metal to Ligand Charge Transfer), MC (Metal Centered Transition), and LC (Ligand Centered Transition), can be leveraged in various innovative ways when combined with graphene, AI/ML, LLM, cybersecurity, and graphene diodes for deep brain stimulation (DBS) and analysis. Here's how:

Graphene and Graphene Diodes

Graphene's exceptional electrical conductivity and biocompatibility make it an ideal material for neural interfaces. Graphene diodes can be used to create highly sensitive and precise electrodes for DBS. These electrodes can facilitate efficient charge transfer processes, enhancing the stimulation and recording of neural activity.

AI/ML and LLM

Artificial Intelligence (AI) and Machine Learning (ML) can analyze the vast amount of data generated during DBS to optimize stimulation parameters and predict outcomes. Large Language Models (LLMs) can assist in processing and interpreting complex medical data, improving decision-making in real-time.

For DBS I suggest to follow new solution of LLM, referring to **KBLaM: Knowledge Base augmented Language Model (source code: GitHub - microsoft/KBLaM: Official Implementation of "KBLaM: Knowledge Base augmented Language Model").**

The Knowledge Base-Augmented Language Model (KBLaM) represents a novel approach to augmenting large language models (LLMs) with external knowledge. Unlike Retrieval-Augmented Generation (RAG), KBLaM eliminates the need for external retrieval modules, thereby streamlining the integration process. Additionally, KBLaM addresses the computational inefficiencies associated with in-context learning by ensuring that its computational overhead scales linearly with the size of the knowledge base, rather than quadratically. This innovative design allows KBLaM to efficiently embed structured knowledge within the model's attention layers, facilitating dynamic updates without the need for retraining.

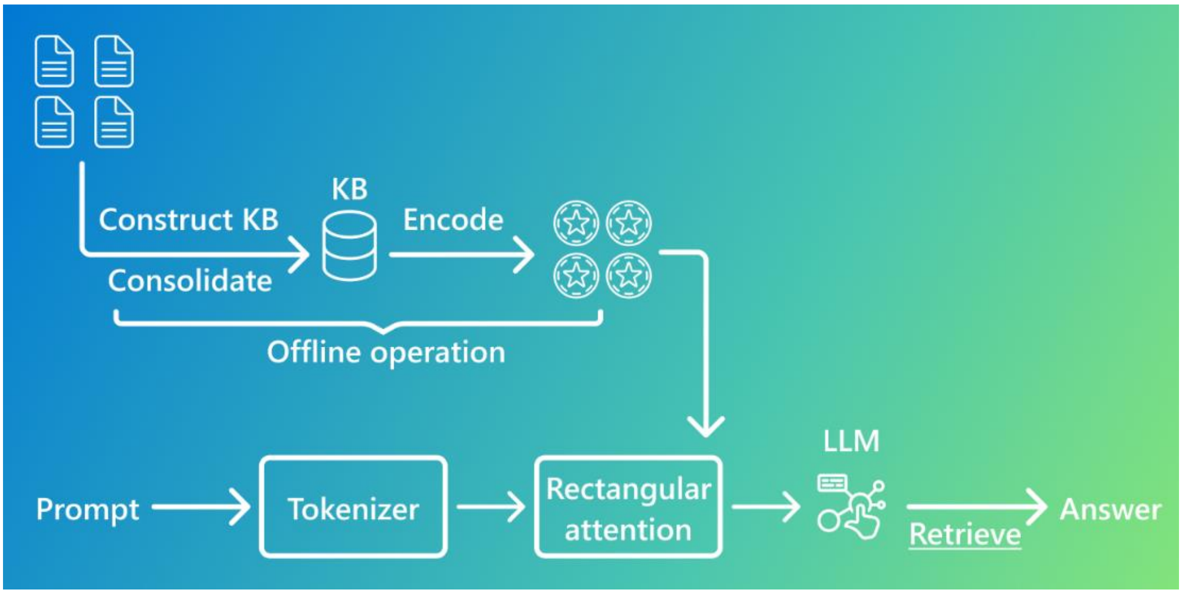


Figure 15 The Knowledge Base-Augmented Language Model (KBLaM) process with augmenting large language models (LLMs) and external knowledge.

Knowledge Encoding: Each knowledge triple is mapped into a key-value vector pair using a pre-trained sentence encoder with lightweight linear adapters. The key vector, derived from the entity name and property, encodes index information, while the value vector captures the corresponding property value, creating continuous, learnable key-value representations.

Integration with LLMs: These key-value pairs, or knowledge tokens, are integrated into the model's attention layers using a specialized rectangular attention structure. Unlike traditional transformer models that process all tokens equally and incur quadratic costs, rectangular attention enables the model to attend to knowledge with linear cost. Language tokens attend to all knowledge tokens, while knowledge tokens do not attend to one another or back to the language tokens. This selective attention pattern significantly reduces computational cost while preserving the model's ability to incorporate external knowledge effectively.

Efficient Knowledge Retrieval: Through rectangular attention, the model dynamically retrieves relevant knowledge tokens during inference, eliminating the need for separate retrieval steps. This linear cost assumption treats each fact independently, ensuring efficient integration of external knowledge.

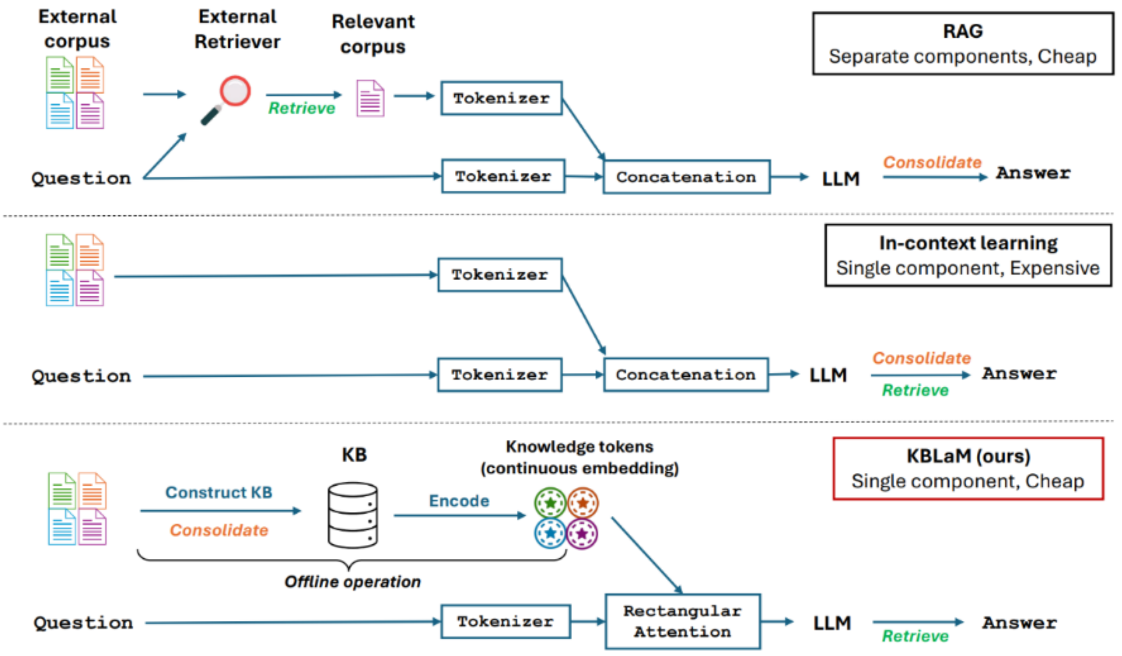


Figure 16: Microsoft KBLaM allows for attention over the entire knowledge base instead of having an external retriever.

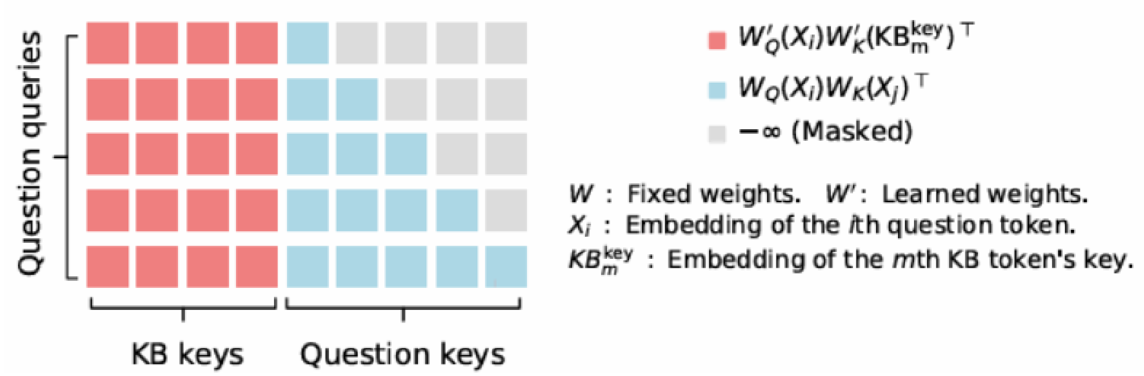

Question queries
KB keys
Question keys
🟥 $W'_Q(X_i)W'_K(KB^{key}_m)^\top$
🟦 $W_Q(X_i)W_K(X_j)^\top$
⬜ $-\infty$ (Masked)
W : Fixed weights. W': Learned weights.
X_i : Embedding of the ith question token.
KB^{key}_m : Embedding of the mth KB token's key.

Figure 17: Source: Microsoft. By having the user's question attend to the knowledge base, while treating facts in the knowledge base independently, KBLaM scales efficiently and linearly with the size of the knowledge base.

Unlike Retrieval-Augmented Generation (RAG), which appends retrieved document chunks to prompts, the Knowledge Base-Augmented Language Model (KBLaM) allows for the direct integration of knowledge into the model. Compared to in-context learning, KBLaM's rectangular attention maintains a linear memory footprint, making it vastly more scalable for large knowledge bases.

The efficiency of KBLaM is a significant advancement. Traditional in-context learning methods struggle with quadratic memory growth due to self-attention overhead. In contrast, KBLaM's linear overhead allows for the storage of a much larger amount of knowledge in the context. Practically, KBLaM can store and process over 10,000 knowledge triples, equivalent to approximately 200,000 text tokens on a single GPU—a feat that would be computationally prohibitive with conventional in-context learning. The results across a wide range of triples, as illustrated in Figure 18, demonstrate this capability. Remarkably, KBLaM achieves this while extending a base model with a context length of only 8K tokens. Additionally, KBLaM enables dynamic updates: modifying a single knowledge triple does not require retraining or re-computation of the entire knowledge base.

Cybersecurity

Ensuring the security of neural data and DBS systems is crucial. Cybersecurity measures can protect sensitive neural data from unauthorized access and manipulation. This includes encryption of data transmission and secure protocols for device communication.

How Retrieval-Augmented Generation (RAG) Enables LLMs to Utilize The Data Sources Without Training?

Large Language Models (LLMs) possess extensive knowledge bases acquired through training. For most scenarios, an LLM can be selected based on specific requirements. However, these models often require additional training to comprehend specific data. Retrieval-Augmented Generation (RAG) allows LLMs to access and utilize my data without the need for further training.

To implement RAG, embeddings are created for my data along with common questions related to it. This can be done dynamically or by storing the embeddings using a vector database solution. When a user poses a question, the LLM uses these embeddings to compare the query to my data and identify the most relevant context. This context, along with the user's question, is then sent to the LLM in a prompt, enabling the model to generate a response based on my data.

To perform RAG, each data source intended for retrieval must be processed. The basic steps are as follows:

- Large datasets are divided into manageable pieces.
- The chunks are converted into a format that can be efficiently searched.
- The converted data is stored in a location that allows efficient access.
- Relevant metadata for citations or references is also stored to ensure accurate responses.
- The converted data is fed to LLMs in prompts, enabling the models to generate responses based on the provided context.

Ensuring cybersecurity in the RAG process involves several key measures: All data, including embeddings and metadata, should be encrypted both at rest and in transit to prevent unauthorized access. The strict access controls should be included to ensure that only authorized personnel can access and modify the data. Next, regular security audits should be conducted to identify and mitigate potential vulnerabilities. Finally, there should be utilized any anomaly detection systems to monitor for unusual access patterns or data usage, which could indicate a security breach.

Vastest challenge: Data: Preparation, App integration, Orchestration & Ingestion, Maintenance.

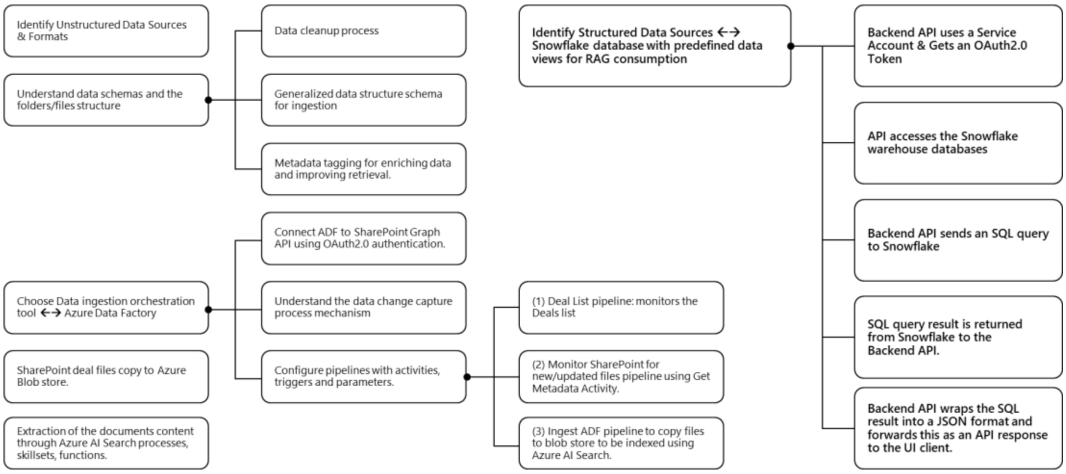

Identify Unstructured Data Sources & Formats
Understand data schemas and the folders/files structure
Data cleanup process
Generalized data structure schema for ingestion
Metadata tagging for enriching data and improving retrieval.
Connect ADF to SharePoint Graph API using OAuth2.0 authentication.
Choose Data ingestion orchestration tool ←→ Azure Data Factory
Understand the data change capture process mechanism
SharePoint deal files copy to Azure Blob store.
Configure pipelines with activities, triggers and parameters.
Extraction of the documents content through Azure AI Search processes, skillsets, functions.
(1) Deal List pipeline: monitors the Deals list
(2) Monitor SharePoint for new/updated files pipeline using Get Metadata Activity.
(3) Ingest ADF pipeline to copy files to blob store to be indexed using Azure AI Search.
Identify Structured Data Sources ←→ Snowflake database with predefined data views for RAG consumption
Backend API uses a Service Account & Gets an OAuth2.0 Token
API accesses the Snowflake warehouse databases
Backend API sends an SQL query to Snowflake
SQL query result is returned from Snowflake to the Backend API.
Backend API wraps the SQL result into a JSON format and forwards this as an API response to the UI client.

Figure 18: Source Microsoft, Data quality process

Data preparation, orchestration, indexing

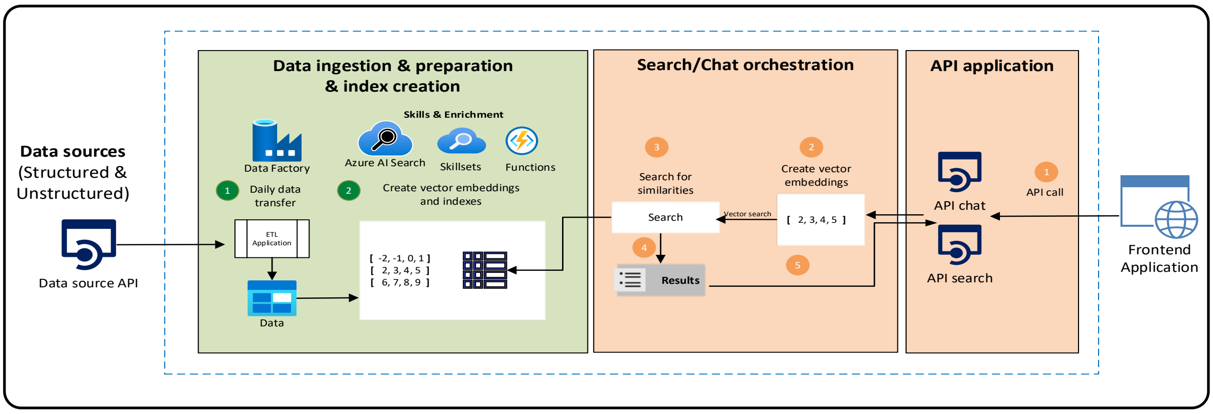
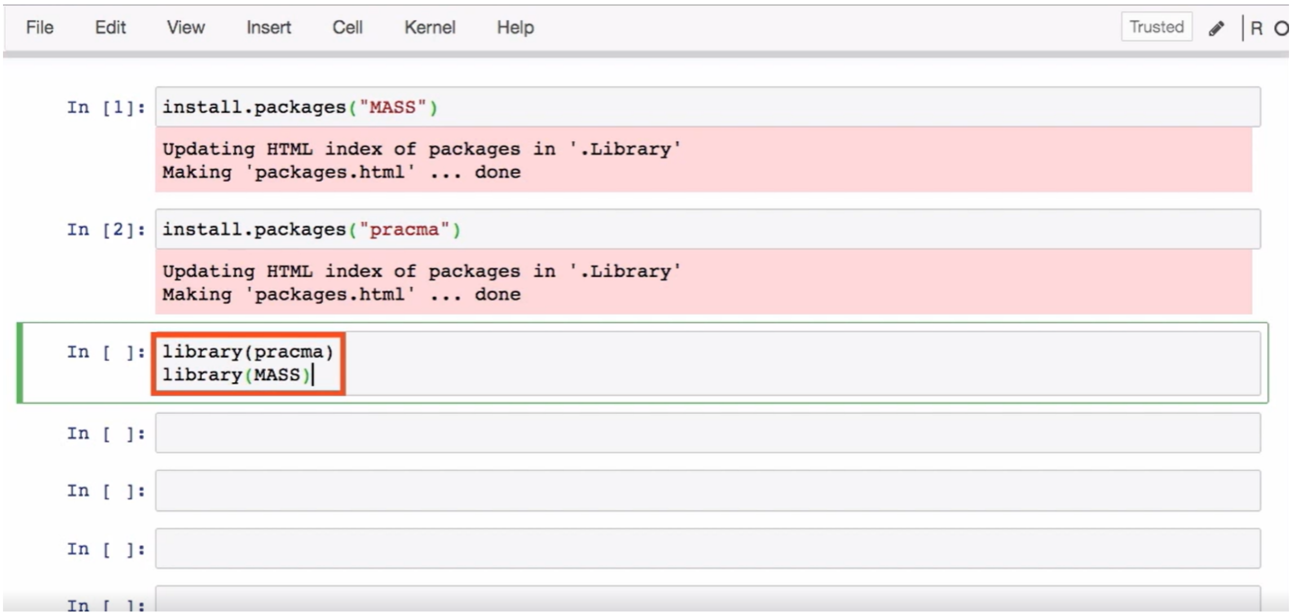


Figure 19: Source: Microsoft, Data governance

Once data pre-processing is ended, we can use cleaned data for further analysis and ML training.



```
In [1]: install.packages("MASS")

        Updating HTML index of packages in '.Library'
        Making 'packages.html' ... done

In [2]: install.packages("pracma")

        Updating HTML index of packages in '.Library'
        Making 'packages.html' ... done

In [ ]: library(pracma)
        library(MASS)

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

Figure 20 Data analysis and incorporated at Jupiter for ML model training

RAG – data retrieval with DB and cybersecurity

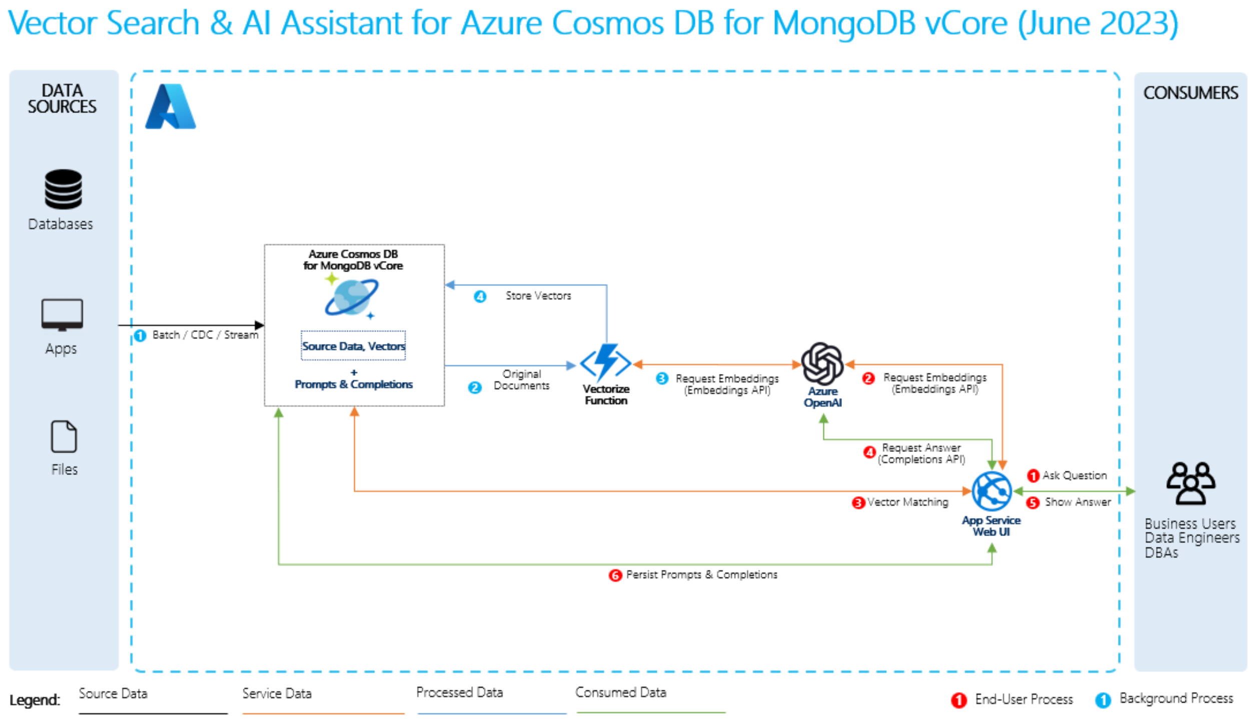


Figure 21 Source: Microsoft: RAG, CosmosDB, Azure OpenAI for MongoDB vCore in ML training process.

Integration of Charge Transfer Mechanisms

The charge transfer mechanisms in octahedral complexes can be applied to enhance the functionality of graphene-based DBS systems. For instance:

- **LMCT and MLCT:** These transitions can be used to design graphene electrodes that efficiently transfer charges between the electrode and neural tissue, improving the effectiveness of stimulation.
- **MC and LC:** These transitions can help in the development of sensors that monitor the neural activity and the effects of DBS, providing real-time feedback and adjustments.

By integrating these technologies, we can create advanced DBS systems that are more effective, secure, and capable of providing personalized treatment based on real-time data analysis.

Artificial Intelligence and Nanotechnology in Deep Brain Stimulation: Artificial Intelligence (AI) and nanotechnology play an increasingly important role in medicine, including deep brain stimulation (DBS). AI can help precisely adjust stimulation parameters, while nanotechnology can enable more effective and less invasive treatment methods.

Stimulation in Combating Migraines with Aura and Strokes: Deep brain stimulation (DBS) and other stimulation techniques are being studied as potential treatments for migraines with aura and stroke prevention. Stimulation can help regulate neuronal activity and reduce migraine symptoms.

Preterm Infants and Seniors: The Vojta method is used for both preterm infants and adult patients, including seniors, to improve movement patterns and neurological functions. Intergenerational integration, such as joint activities for seniors and children, can benefit both groups.

The essence of scientific inquiry can be encapsulated in several fundamental vectors:

- **Observations:** The systematic collection and analysis of empirical data form the cornerstone of scientific research. Observations provide the raw material from which hypotheses are generated and theories are constructed.
- **Repeatability and Reproducibility with Statistical Process Control (SPC):** The ability to replicate results under consistent conditions is crucial for validating scientific findings. Statistical Process Control (SPC) techniques ensure that experiments are conducted with precision and consistency, thereby enhancing the reliability of the results.
- **Data Science:** The application of advanced analytical methods and computational techniques to interpret complex datasets is integral to modern scientific research. Data science enables researchers to uncover patterns, make predictions, and derive insights from vast amounts of data. Julia language should be considered as a very first technology choice or in integration with Python.
- **Intuition:** While empirical evidence and rigorous analysis are paramount, intuition also plays a significant role in scientific discovery. The ability to perceive connections and generate innovative ideas often stems from an intuitive understanding of the subject matter.

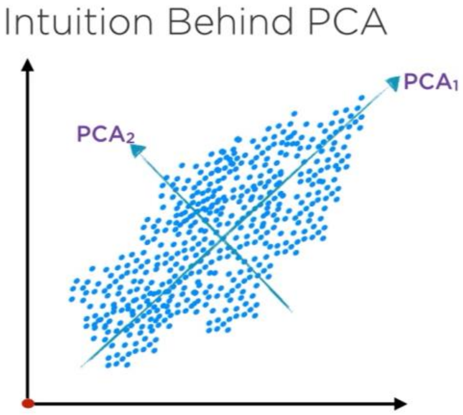
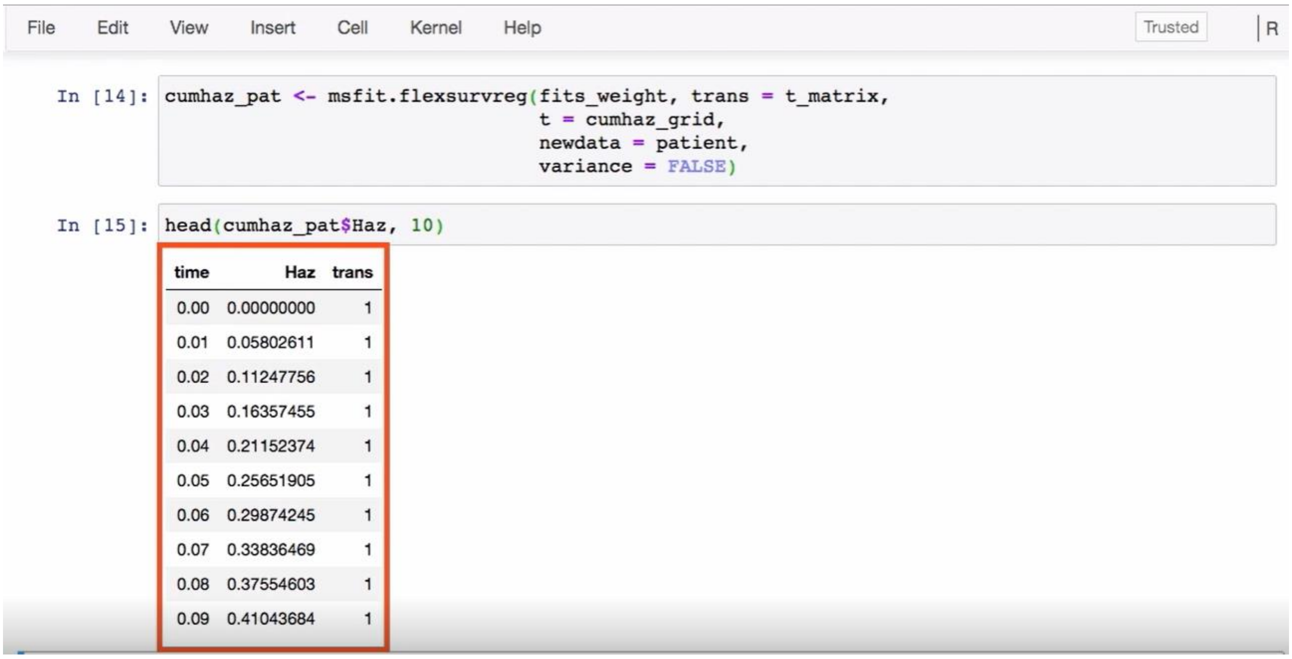


Figure 22 Intuition behind PCA researches

Model MASS with R - An R package containing functions and datasets designed to complement the renowned textbook "Modern Applied Statistics with S" by Venables and Ripley.

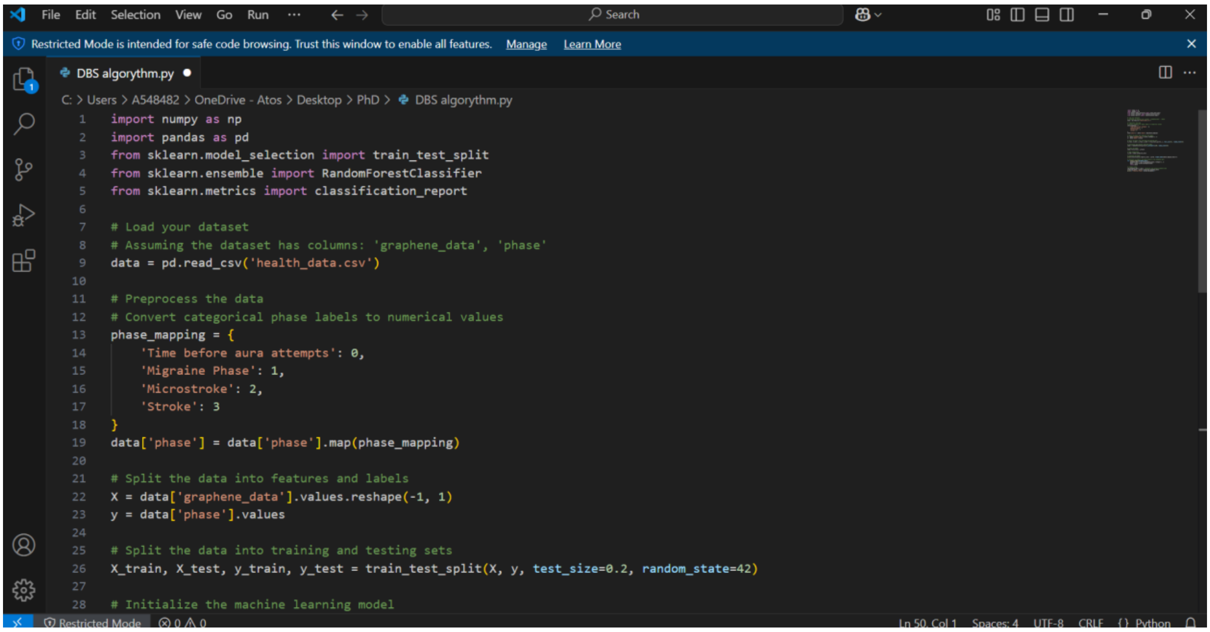


```
In [14]: cumhaz_pat <- msfit.flexsurvreg(fits_weight, trans = t_matrix,
                                          t = cumhaz_grid,
                                          newdata = patient,
                                          variance = FALSE)

In [15]: head(cumhaz_pat$Haz, 10)
```

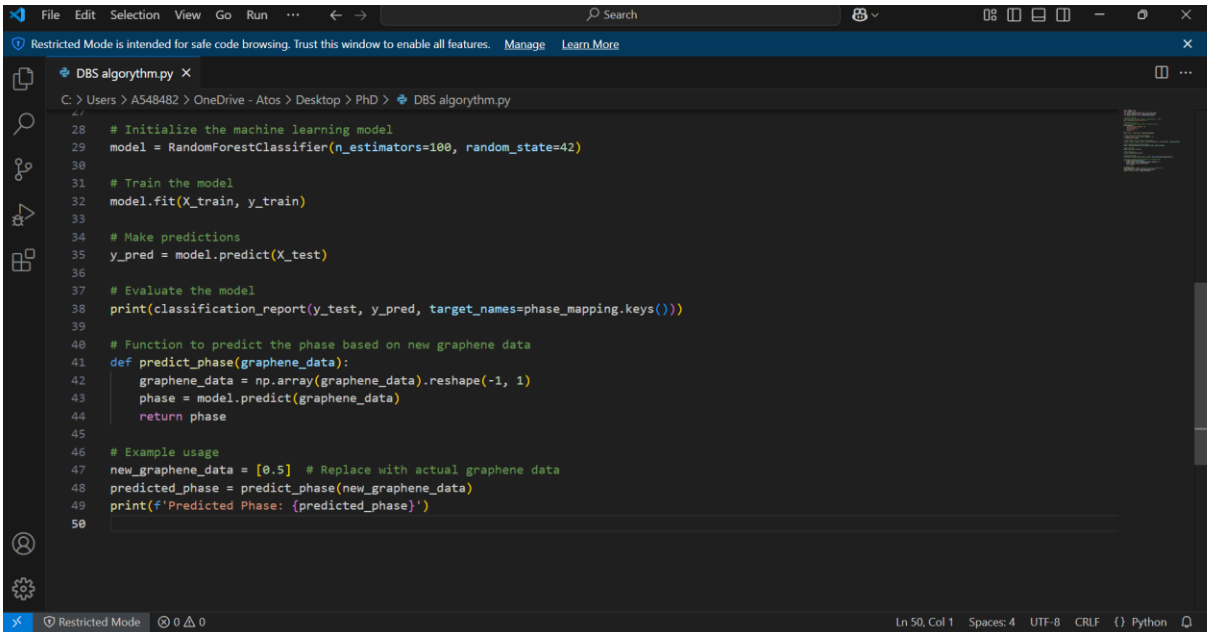
```
time        Haz  trans
0.00  0.00000000     1
0.01  0.05802611     1
0.02  0.11247756     1
0.03  0.16357455     1
0.04  0.21152374     1
0.05  0.25651905     1
0.06  0.29874245     1
0.07  0.33836469     1
0.08  0.37554603     1
0.09  0.41043684     1
```

Figure 23 Modern Applied Statistics with S by Venables and Ripley.

A simplified Python algorithm that outlines the structure for analyzing the four phases: Time before aura attempts, Migraine Phase, Microstroke, and Stroke. This algorithm uses machine learning and incorporates the use of graphene data. Note that this is a basic framework and would need to be expanded with actual data and specific machine learning models.


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py
Restricted Mode ⊗ 0 ⚠ 0
Ln 50, Col 1 Spaces: 4 UTF-8 CRLF {} Python

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Load your dataset
# Assuming the dataset has columns: 'graphene_data', 'phase'
data = pd.read_csv('health_data.csv')

# Preprocess the data
# Convert categorical phase labels to numerical values
phase_mapping = {
    'Time before aura attempts': 0,
    'Migraine Phase': 1,
    'Microstroke': 2,
    'Stroke': 3
}
data['phase'] = data['phase'].map(phase_mapping)

# Split the data into features and labels
X = data['graphene_data'].values.reshape(-1, 1)
y = data['phase'].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the machine learning model
```


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
DBS algorythm.py
C: > Users > A548482 > OneDrive - Atos > Desktop > PhD > DBS algorythm.py
Restricted Mode ⊗ 0 ⚠ 0
Ln 50, Col 1 Spaces: 4 UTF-8 CRLF {} Python

```python
# Initialize the machine learning model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred, target_names=phase_mapping.keys()))

# Function to predict the phase based on new graphene data
def predict_phase(graphene_data):
    graphene_data = np.array(graphene_data).reshape(-1, 1)
    phase = model.predict(graphene_data)
    return phase

# Example usage
new_graphene_data = [0.5]  # Replace with actual graphene data
predicted_phase = predict_phase(new_graphene_data)
print(f'Predicted Phase: {predicted_phase}')

```

Figure 24 A simplified Python algorithm that outlines the structure for analyzing the four phases: Time before aura attempts, Migraine Phase, Microstroke, and Stroke.

Explanation:

1. **Data Loading and Preprocessing**: The dataset is loaded and preprocessed. The phases are mapped to numerical values for machine learning.

2. **Feature and Label Extraction**: The graphene data is extracted as features, and the phase labels are extracted as targets.
3. **Data Splitting**: The data is split into training and testing sets.
4. **Model Initialization and Training**: A RandomForestClassifier is used to train the model on the training data.
5. **Prediction and Evaluation**: The model makes predictions on the test data, and the performance is evaluated using a classification report.
6. **Prediction Function**: A function is provided to predict the phase based on new graphene data.

This is a basic example to get started with data analysis and pre-processing. It can be expanded and refined this algorithm by incorporating more sophisticated preprocessing, feature engineering, and model tuning based on the specific requirements and data.

Breaking down the code step by step:

1. Importing Libraries

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report
```

- **numpy**: Used for numerical operations.
- **pandas**: Used for data manipulation and analysis.
- **sklearn.model_selection.train_test_split**: Used to split the dataset into training and testing sets.
- **sklearn.ensemble.RandomForestClassifier**: A machine learning model used for classification tasks.

- **sklearn.metrics.classification_report**: Used to evaluate the performance of the classification model.

2. Loading the Dataset

```
data = pd.read_csv('health_data.csv')
```

- **pd.read_csv**: Reads the dataset from a CSV file named 'health_data.csv'.

3. Preprocessing the Data

```
phase_mapping = {
    'Time before aura attempts': 0,
    'Migraine Phase': 1,
    'Microstroke': 2,
    'Stroke': 3
}
data['phase'] = data['phase'].map(phase_mapping)
```

- **phase_mapping**: A dictionary that maps phase names to numerical values.
- **data['phase'].map(phase_mapping)**: Converts the phase labels in the dataset to numerical values using the mapping dictionary.

4. Feature and Label Extraction

```
X = data['graphene_data'].values.reshape(-1, 1)
y = data['phase'].values
```

- **X**: Extracts the 'graphene_data' column as features and reshapes it to a 2D array.
- **y**: Extracts the 'phase' column as labels.

5. Splitting the Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **train_test_split**: Splits the data into training and testing sets. 80% of the data is used for training, and 20% is used for testing. The **random_state** parameter ensures reproducibility.

6. Initializing the Model

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

- **RandomForestClassifier**: Initializes a Random Forest classifier with 100 trees. The **random_state** parameter ensures reproducibility.

7. Training the Model

```
model.fit(X_train, y_train)
```

- **model.fit**: Trains the model using the training data.

8. Making Predictions

```
y_pred = model.predict(X_test)
```

- **model.predict**: Makes predictions on the test data.

9. Evaluating the Model

```
print(classification_report(y_test, y_pred, target_names=phase_mapping.keys()))
```

- **classification_report**: Prints a report showing the precision, recall, and F1-score for each class.

10. Prediction Function

```
def predict_phase(graphene_data):
    graphene_data = np.array(graphene_data).reshape(-1, 1)
    phase = model.predict(graphene_data)
    return phase
```

- **predict_phase**: A function that takes new graphene data as input, reshapes it, and uses the trained model to predict the phase.

11. Example Usage

```
new_graphene_data = [0.5]  # Replace with actual graphene data

predicted_phase = predict_phase(new_graphene_data)

print(f'Predicted Phase: {predicted_phase}')
```

- **new_graphene_data**: An example of new graphene data.
- **predict_phase(new_graphene_data)**: Uses the prediction function to predict the phase for the new data.
- **print**: Prints the predicted phase.

This code provides a basic framework for analyzing the four phases using machine learning with graphene data. It can be expanded and refined based on the specific requirements and data.

Here are a few sources where the healthcare datasets can be found that might be useful for the machine learning projects:

1. **Kaggle**: Kaggle is a popular platform for data science competitions and hosts a variety of datasets. It can be found with healthcare-related datasets by searching for terms like "healthcare," "medical," "brain," "stroke," etc. Kaggle Datasets
2. **UCI Machine Learning Repository**: The UCI repository is a well-known source for machine learning datasets. They have several healthcare-related datasets that can be explored furtherly.
3. **PhysioNet**: PhysioNet offers a large collection of physiological and clinical data, including data related to brain activity, strokes, and other medical conditions. PhysioNet
4. **MIMIC-III**: The MIMIC-III (Medical Information Mart for Intensive Care) database contains de-identified health data associated with over 40,000 critical care patients. It includes information on demographics, vital signs, laboratory tests, medications, and more. MIMIC-III
5. **Stanford Health Care AI**: Stanford provides access to longitudinal electronic health record (EHR) datasets that reflect the diversity and complexity of real-world healthcare. These datasets are useful for developing scalable AI systems. Stanford Health Care AI

Graphene is an exciting material in the field of brain monitoring due to its unique properties, such as high conductivity, flexibility, and biocompatibility. Below I explain how graphene can be used in brain monitoring, along with relevant nanotechnologies to ensure non-invasive and safe patient monitoring:

Graphene-Based Brain Implants

Graphene-based implants can be used to monitor brain activity with high precision. These implants are designed to be biocompatible, meaning they can be safely integrated into the brain without causing harm. The implants can detect electrical activity across a wide range of frequencies, including very low frequencies that are often missed by traditional electrodes

Graphene Tattoos

A novel approach involves the use of graphene tattoos, which are ultra-thin, flexible patches that can be applied to the skin. These tattoos can monitor brain wave activity continuously with minimal impact on a person's daily life. They work by detecting electrical signals from the brain and transmitting them wirelessly to a receiver

Nanotechnology for Non-Invasive Monitoring

Nanotechnology plays a crucial role in enhancing the capabilities of graphene-based devices. Here are some key technologies:

1. **Wireless Transmission**: Graphene sensors can be paired with wireless transmission systems to send data to external devices without the need for wires or invasive procedures. This allows for continuous monitoring of brain activity in real-time
2. **Miniaturized Electronics**: Advances in miniaturized electronics enable the development of small, lightweight devices that can be comfortably worn by patients. These devices can process and transmit data efficiently, reducing the need for bulky equipment
3. **Biocompatible Materials**: The use of biocompatible materials ensures that the graphene implants or tattoos do not cause adverse reactions in the body. This is essential for long-term monitoring and patient safety.

Applications in Healthcare

Graphene-based monitoring systems can be used to track various brain conditions, including:

- **Migraines with Aura**: By monitoring brain activity, these systems can help identify patterns that precede migraines, allowing for early intervention.
- **Microstrokes and Strokes**: Continuous monitoring can detect early signs of microstrokes and strokes, enabling timely medical response.
- **Epilepsy**: Graphene sensors can detect the onset of epileptic seizures, providing critical information for managing the condition

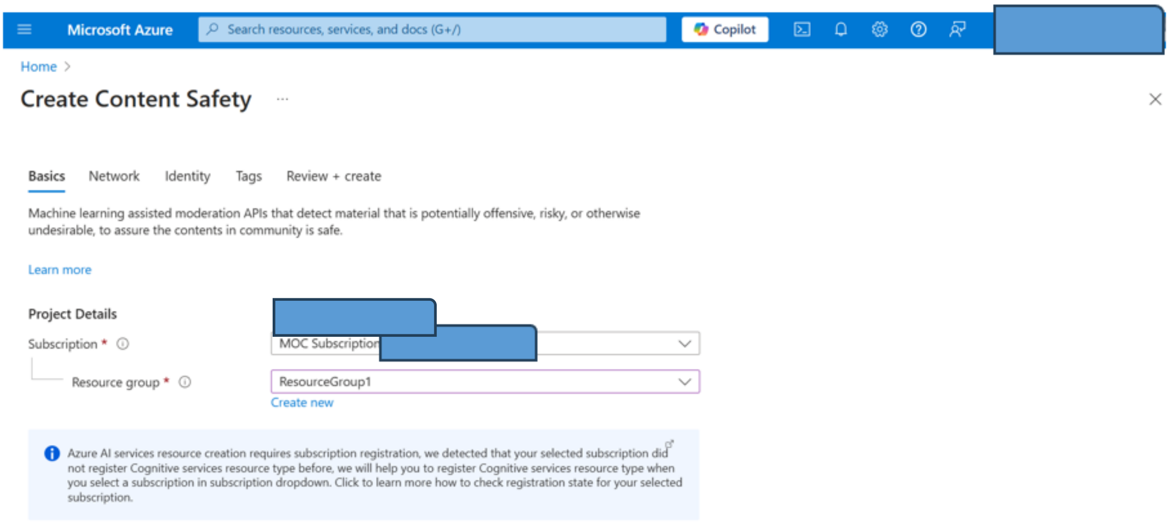
Conclusion

Graphene, combined with advanced nanotechnologies, offers a promising solution for non-invasive brain monitoring. These technologies provide accurate, real-time data without the need for needles or other invasive equipment, ensuring patient safety and comfort.

Chapter 4: Results, presenting the findings of the study.

Artificial Intelligence with platform of services – Azure AI, Azure Open AI, Azure Databricks, Azure AI Foundry, Azure Fabric, Machine Learning, Deep Learning of DBS

I will selectively present results utilizing Information Technology (IT) tools and methodologies, while also incorporating non-data-driven insights such as intuition and observability. While observability is a subject of DevOps and can already be incorporated into large language models (LLMs), machine learning (ML), and neural networks, intuition - although confirmed by scientists as a factor inherent to every individual - remains challenging to express through data.



Create Content Safety

Machine learning assisted moderation APIs that detect material that is potentially offensive, risky, or otherwise undesirable, to assure the contents in community is safe.

Learn more

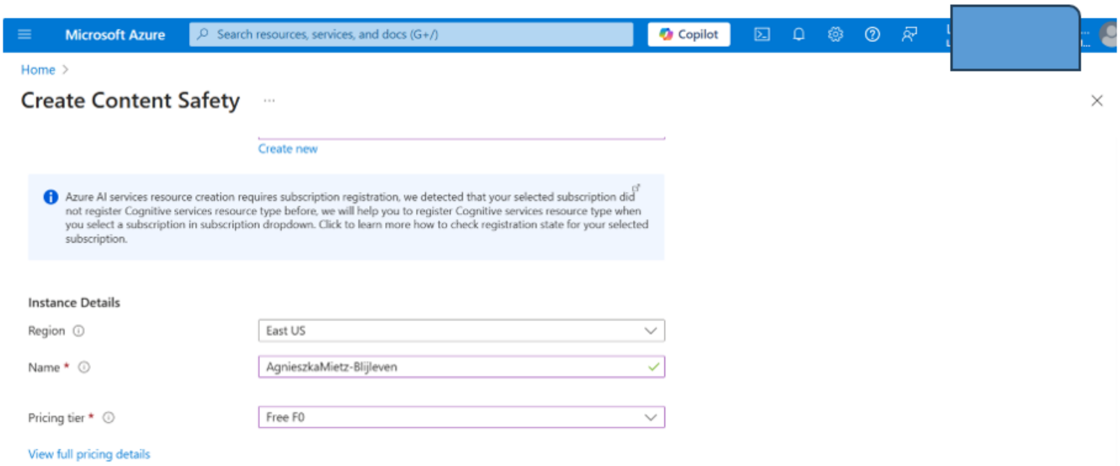
Project Details

Subscription * : MOC Subscriptio [redacted]

Resource group * : ResourceGroup1

Create new

Azure AI services resource creation requires subscription registration, we detected that your selected subscription did not register Cognitive services resource type before, we will help you to register Cognitive services resource type when you select a subscription in subscription dropdown. Click to learn more how to check registration state for your selected subscription.



Create Content Safety

Create new

Azure AI services resource creation requires subscription registration, we detected that your selected subscription did not register Cognitive services resource type before, we will help you to register Cognitive services resource type when you select a subscription in subscription dropdown. Click to learn more how to check registration state for your selected subscription.

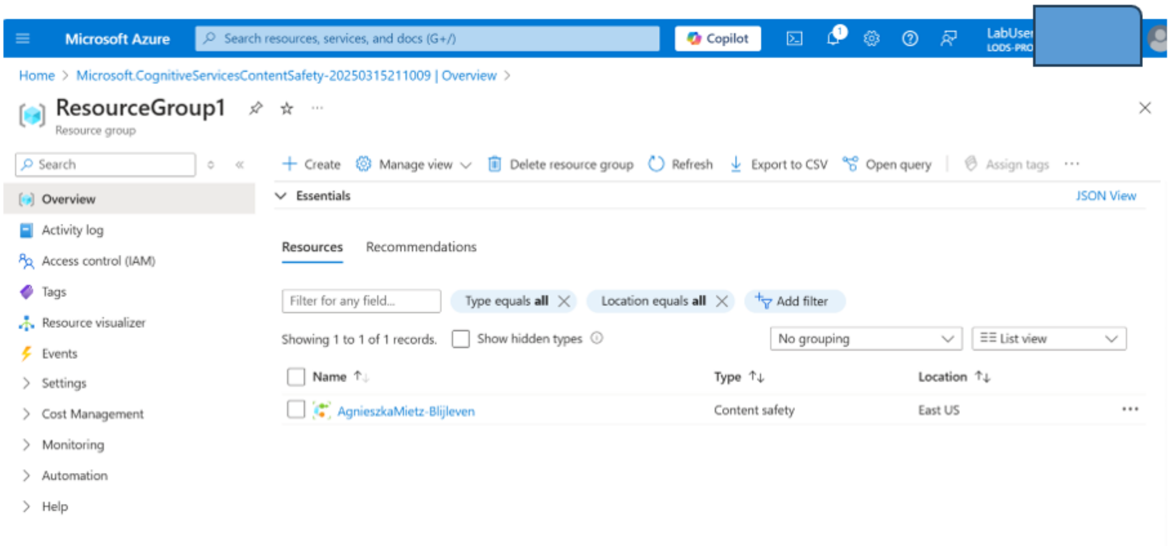
Instance Details

Region : East US

Name * : AgnieszkaMietz-Blijleven

Pricing tier * : Free F0

View full pricing details



[redacted]

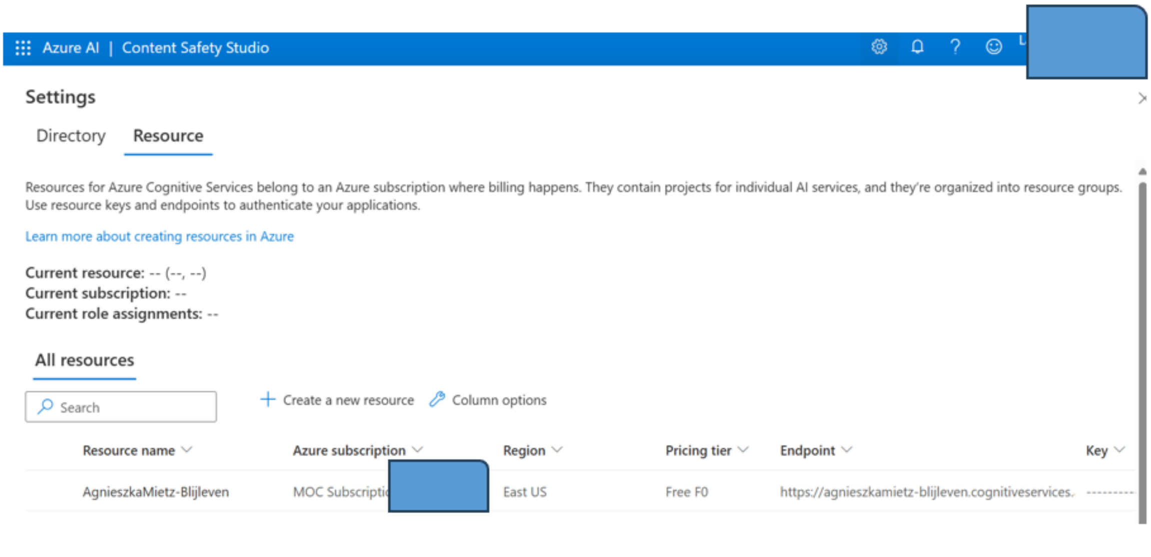
Home > Microsoft.CognitiveServicesContentSafety-20250315211009 | Overview >

ResourceGroup1

Resource group

Showing 1 to 1 of 1 records. □ Show hidden types

□ Name	Type	Location
□ AgnieszkaMietz-Blijleven	Content safety	East US



[redacted]

Settings

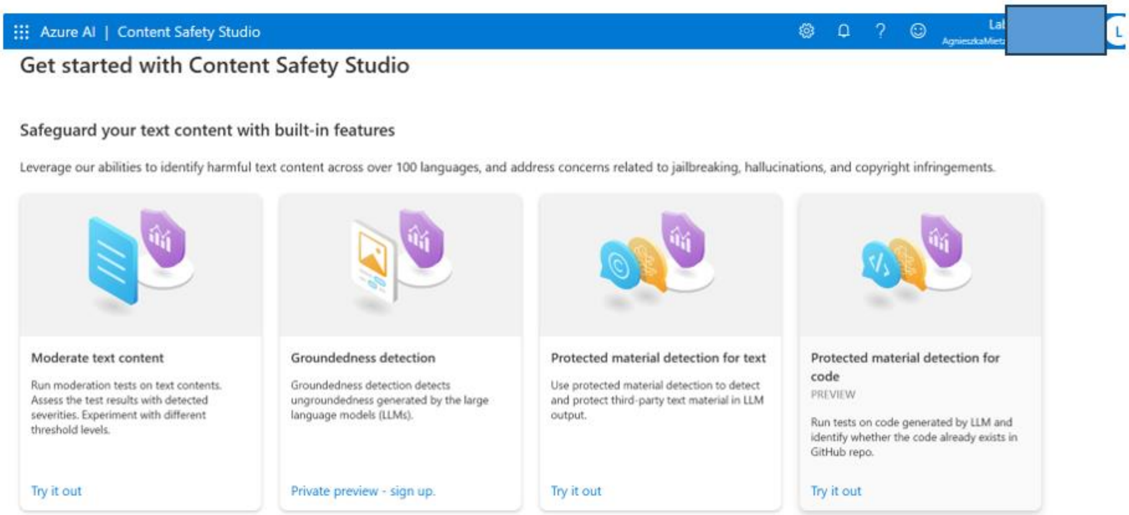
Resources for Azure Cognitive Services belong to an Azure subscription where billing happens. They contain projects for individual AI services, and they're organized into resource groups. Use resource keys and endpoints to authenticate your applications.

Learn more about creating resources in Azure

Current resource: -- (--, --)
Current subscription: --
Current role assignments: --

All resources

Resource name	Azure subscription	Region	Pricing tier	Endpoint	Key
AgnieszkaMietz-Blijleven	MOC Subscripti [redacted]	East US	Free F0	https://agnieszkamietz-blijleven.cognitiveservices.	---------



Azure AI | Content Safety Studio [redacted]

Get started with Content Safety Studio

Safeguard your text content with built-in features

Leverage our abilities to identify harmful text content across over 100 languages, and address concerns related to jailbreaking, hallucinations, and copyright infringements.

Moderate text content

Run moderation tests on text contents. Assess the test results with detected severities. Experiment with different threshold levels.

Try it out

Groundedness detection

Groundedness detection detects ungroundedness generated by the large language models (LLMs).

Private preview - sign up.

Protected material detection for text

Use protected material detection to detect and protect third-party text material in LLM output.

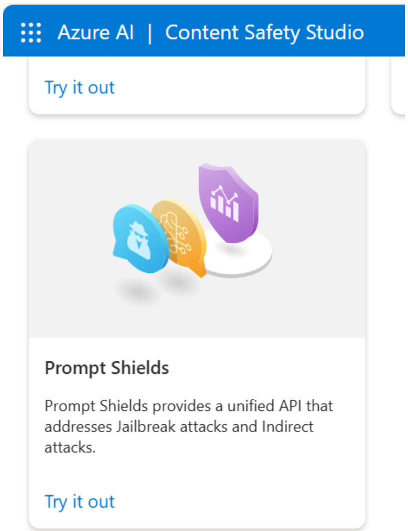
Try it out

Protected material detection for code

PREVIEW

Run tests on code generated by LLM and identify whether the code already exists in GitHub repo.

Try it out



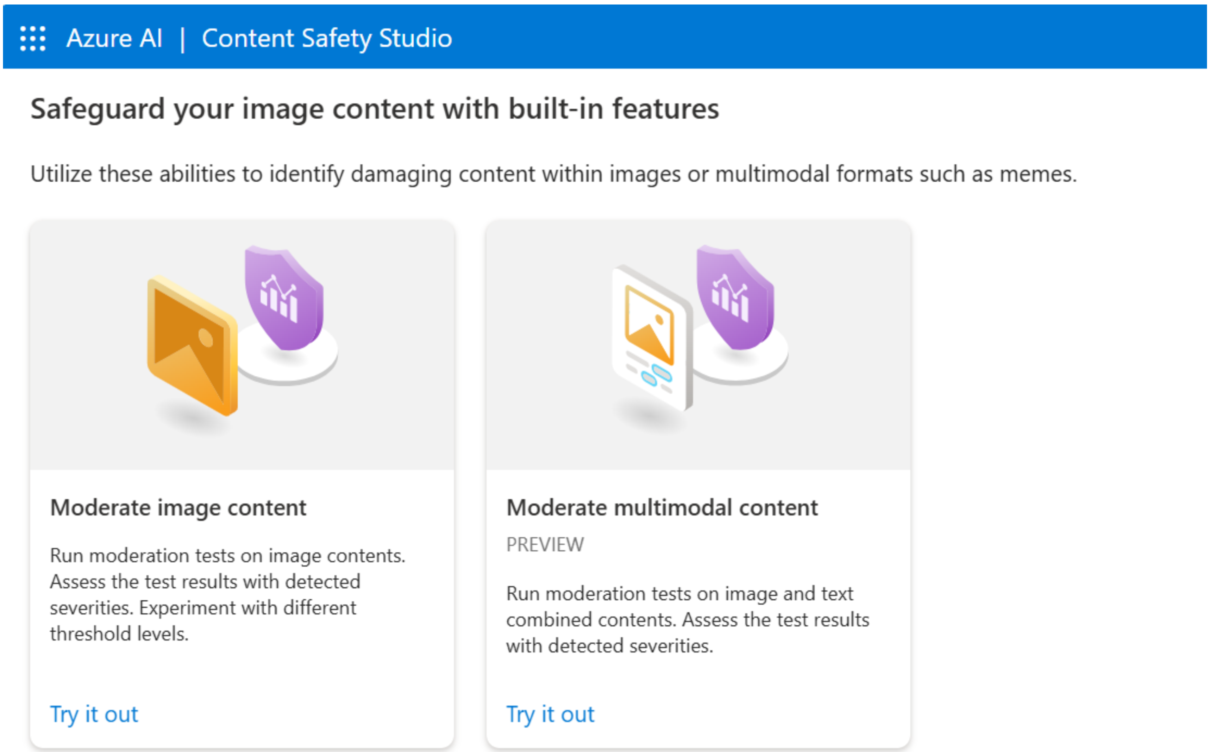
Azure AI | Content Safety Studio

Try it out

Prompt Shields

Prompt Shields provides a unified API that addresses Jailbreak attacks and Indirect attacks.

Try it out



Azure AI | Content Safety Studio

Safeguard your image content with built-in features

Utilize these abilities to identify damaging content within images or multimodal formats such as memes.

Moderate image content

Run moderation tests on image contents. Assess the test results with detected severities. Experiment with different threshold levels.

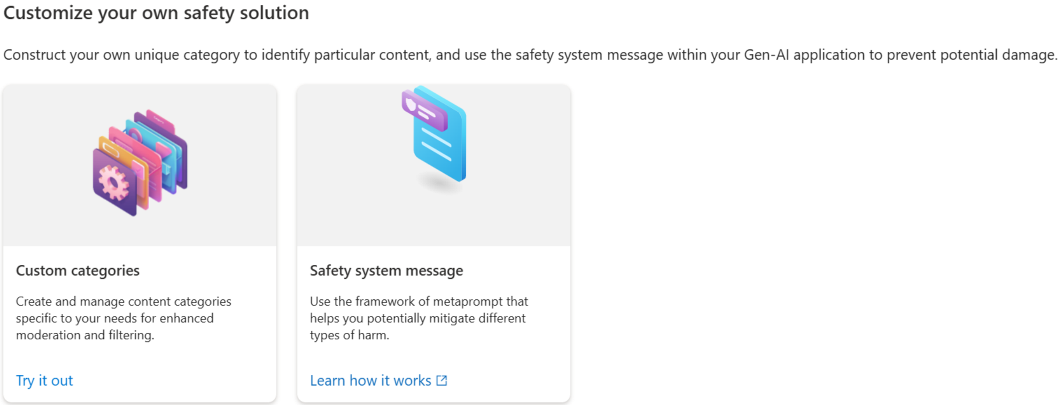
Try it out

Moderate multimodal content

PREVIEW

Run moderation tests on image and text combined contents. Assess the test results with detected severities.

Try it out



Customize your own safety solution

Construct your own unique category to identify particular content, and use the safety system message within your Gen-AI application to prevent potential damage.

Custom categories

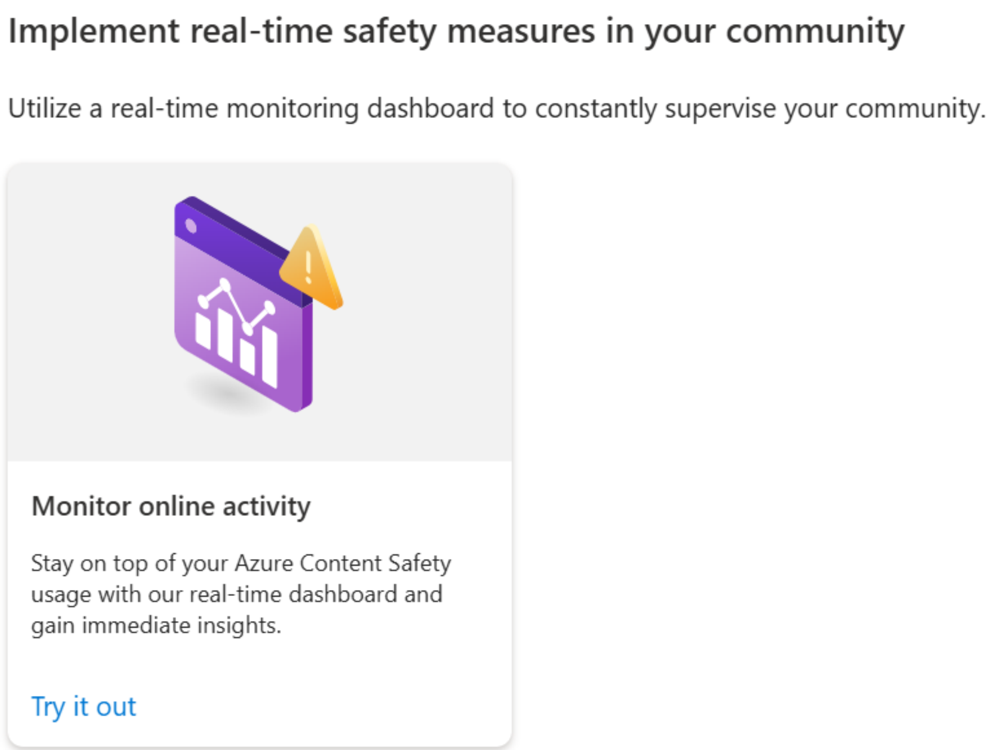
Create and manage content categories specific to your needs for enhanced moderation and filtering.

Try it out

Safety system message

Use the framework of metaprompt that helps you potentially mitigate different types of harm.

Learn how it works



Implement real-time safety measures in your community

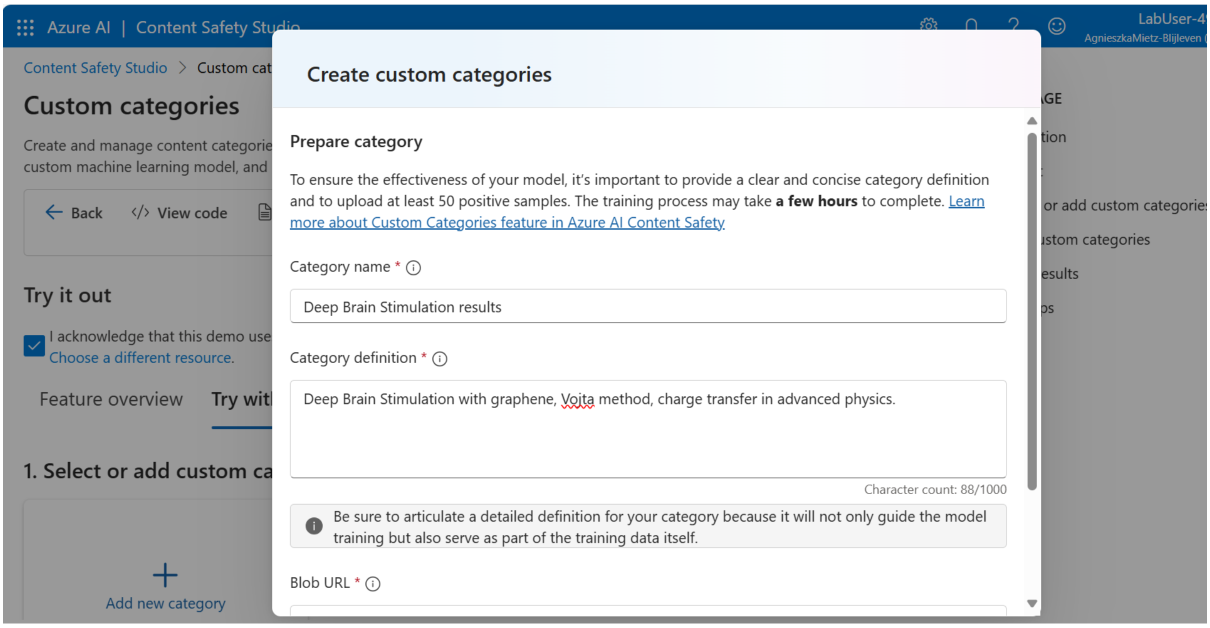
Utilize a real-time monitoring dashboard to constantly supervise your community.

Monitor online activity

Stay on top of your Azure Content Safety usage with our real-time dashboard and gain immediate insights.

Figure 25 Microsoft Azure AI, OpenAI services platform

Regulatory & Compliance, responsible AI



Azure AI | Content Safety Studio
LabUser-4
AgnieszkaMietz-Blijleven (
Content Safety Studio > Custom cat
Custom categories
Create and manage content categorie
custom machine learning model, and
Back
View code
Try it out
I acknowledge that this demo use
Choose a different resource.
Feature overview
Try wit
1. Select or add custom ca
Add new category
GE
tion
or add custom categorie
istom categories
esults
ps

Create custom categories

Prepare category

To ensure the effectiveness of your model, it's important to provide a clear and concise category definition and to upload at least 50 positive samples. The training process may take **a few hours** to complete. Learn more about Custom Categories feature in Azure AI Content Safety

Category name *

Deep Brain Stimulation results

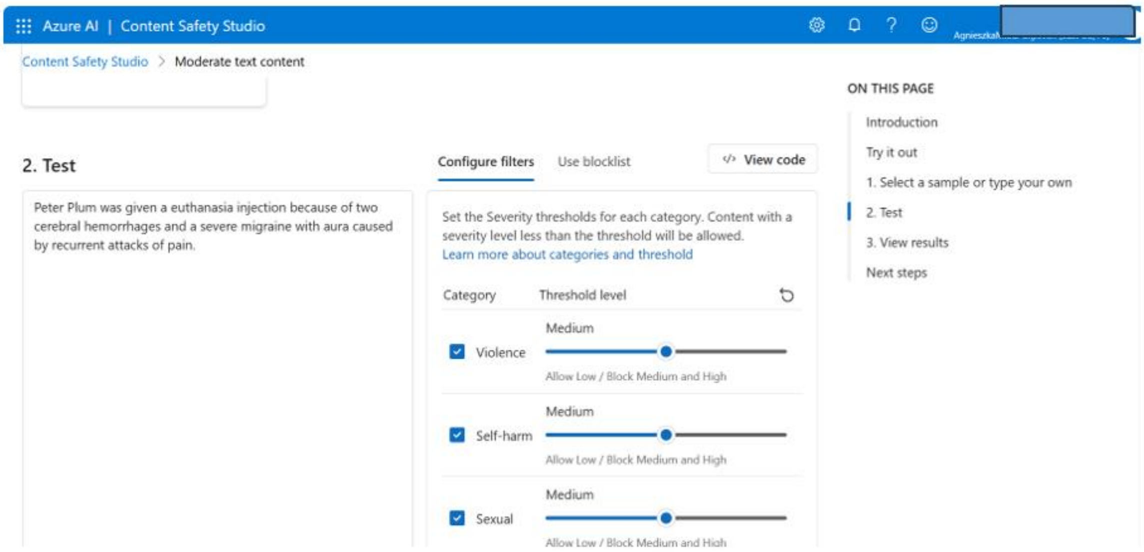
Category definition *

Deep Brain Stimulation with graphene, Vojta method, charge transfer in advanced physics.

Character count: 88/1000

Be sure to articulate a detailed definition for your category because it will not only guide the model training but also serve as part of the training data itself.

Blob URL *

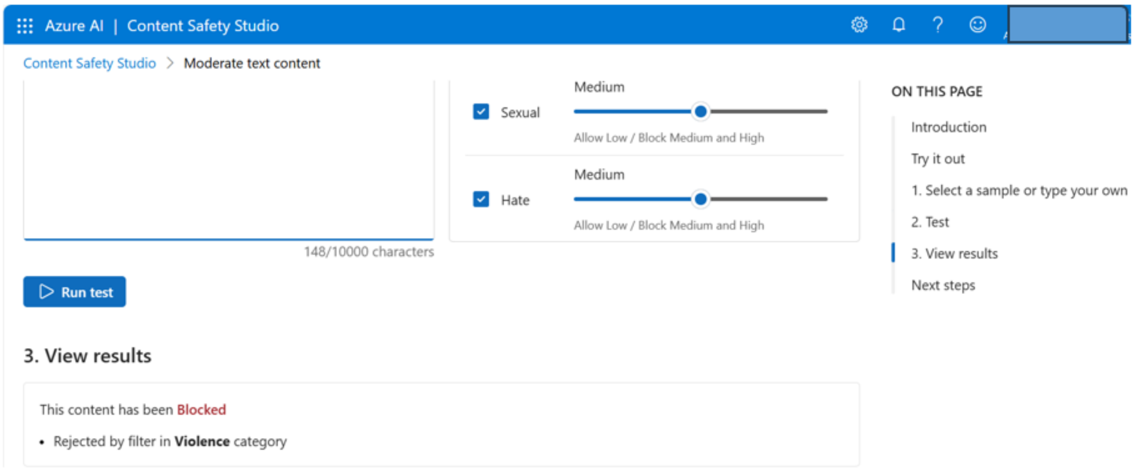


2. Test

Peter Plum was given a euthanasia injection because of two cerebral hemorrhages and a severe migraine with aura caused by recurrent attacks of pain.

Set the Severity thresholds for each category. Content with a severity level less than the threshold will be allowed.
Learn more about categories and threshold

Category	Threshold level
☑ Violence	Medium Allow Low / Block Medium and High
☑ Self-harm	Medium Allow Low / Block Medium and High
☑ Sexual	Medium Allow Low / Block Medium and High

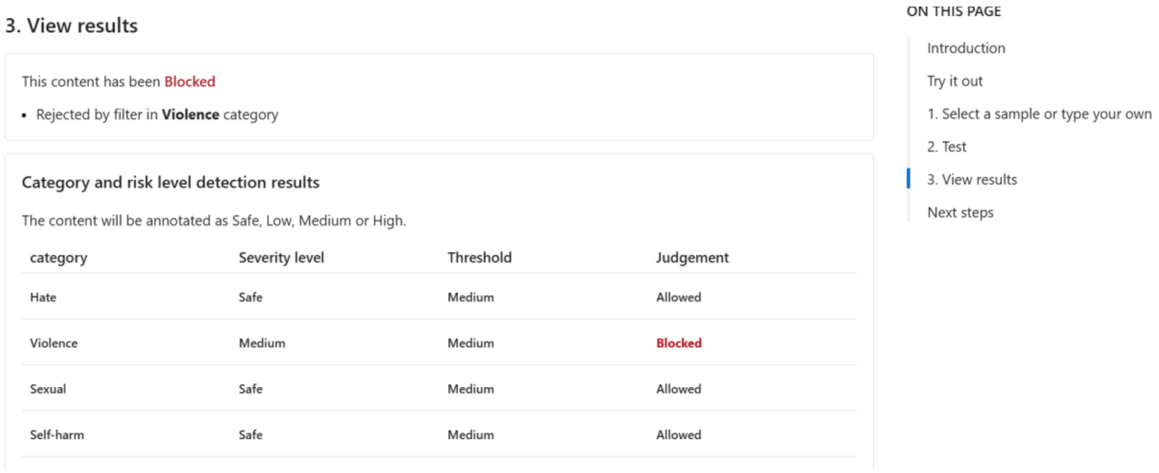

Azure AI | Content Safety Studio
[redacted]
Content Safety Studio > Moderate text content
ON THIS PAGE
Introduction
Try it out
1. Select a sample or type your own
2. Test
3. View results
Next steps
148/10000 characters
Run test

☑ Sexual	Medium Allow Low / Block Medium and High
☑ Hate	Medium Allow Low / Block Medium and High

3. View results

This content has been Blocked

- Rejected by filter in **Violence** category



3. View results

This content has been Blocked

- Rejected by filter in **Violence** category

Category and risk level detection results

The content will be annotated as Safe, Low, Medium or High.

category	Severity level	Threshold	Judgement
Hate	Safe	Medium	Allowed
Violence	Medium	Medium	Blocked
Sexual	Safe	Medium	Allowed
Self-harm	Safe	Medium	Allowed

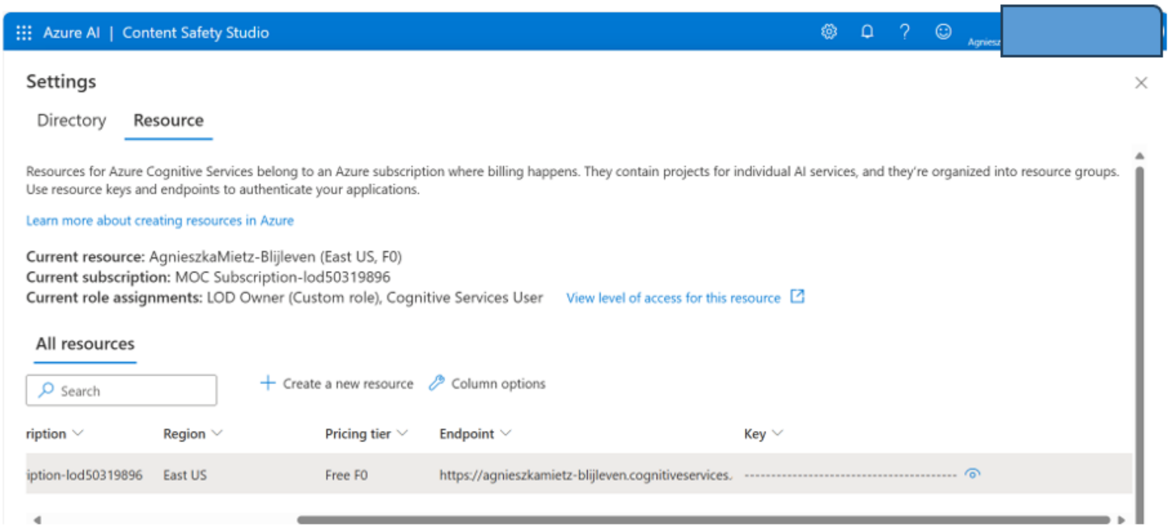
Figure 26 Governance of responsible AI

In the Results panel, inspect the results. There are four severity levels from safe to high, and four types of harmful content. Does the Content Safety AI service consider this sample to be acceptable or not? What's important to note is that the results are within a confidence interval. A well-trained model, like one of Azure AI's out-of-the-box models, can return results that have a high probability of matching what a human would label the result. Each time I run a test, I call the model again.

Cybersecurity in Deep Brain Stimulation

Endpoints and keys

In the Azure Portal, I will see that these are the *same* endpoint and *different* keys for my resource. To check it out, head to the Azure Portal. Search for *Content Safety* on the top search bar. Find the resource and click on it. On the left-hand menu, look under *Resource Management* for *Keys and Endpoints*. Select **Keys and Endpoints** to view the endpoint and keys for the given resource.



[redacted]

Settings

Directory Resource

Resources for Azure Cognitive Services belong to an Azure subscription where billing happens. They contain projects for individual AI services, and they're organized into resource groups. Use resource keys and endpoints to authenticate your applications.

Learn more about creating resources in Azure

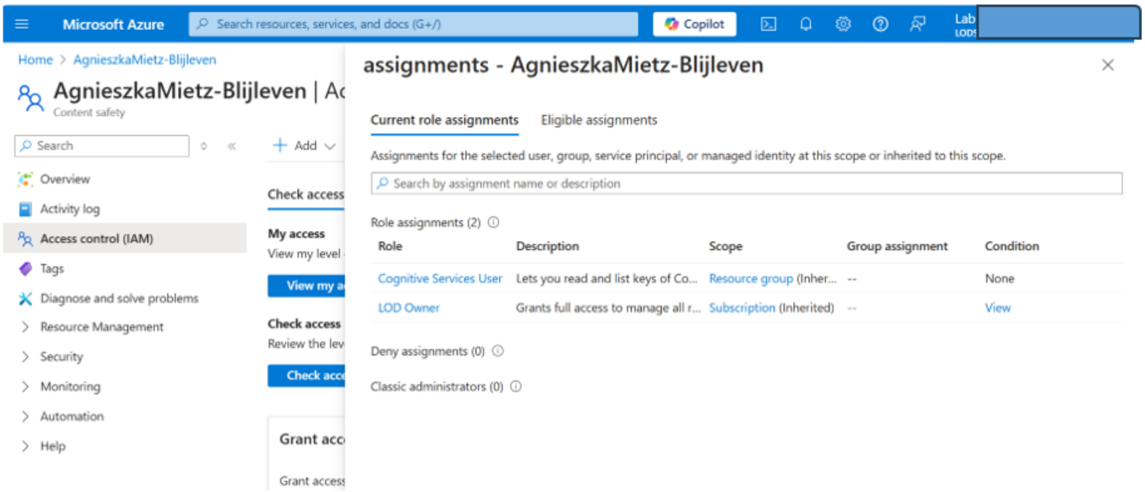
Current resource: AgnieszkaMietz-Blijleven (East US, F0)
Current subscription: MOC Subscription-lod50319896
Current role assignments: LOD Owner (Custom role), Cognitive Services User View level of access for this resource

All resources

ription	Region	Pricing tier	Endpoint	Key
iption-lod50319896	East US	Free F0	https://agnieszkamietz-blijleven.cognitiveservices.	-------------------------------------

Figure 27 Microsoft Azure Cognitive Services

Here I decide if only me or others implement the sensitive data:



[redacted]

assignments - AgnieszkaMietz-Blijleven

Current role assignments Eligible assignments

Assignments for the selected user, group, service principal, or managed identity at this scope or inherited to this scope.

Role assignments (2)

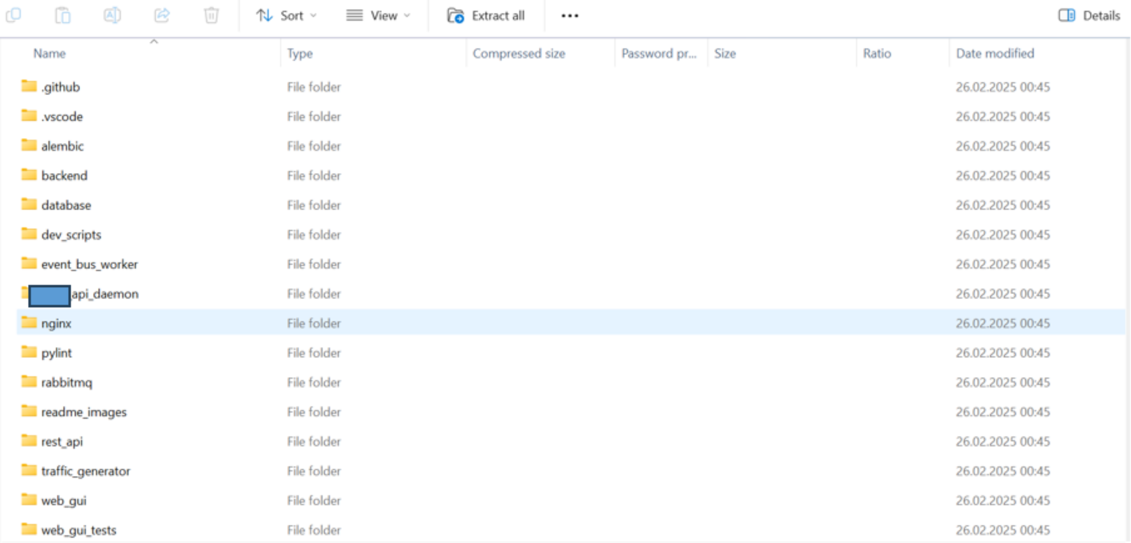
Role	Description	Scope	Group assignment	Condition
Cognitive Services User	Lets you read and list keys of Co...	Resource group (Inher...	--	None
LOD Owner	Grants full access to manage all r...	Subscription (Inherited)	--	View

Deny assignments (0)

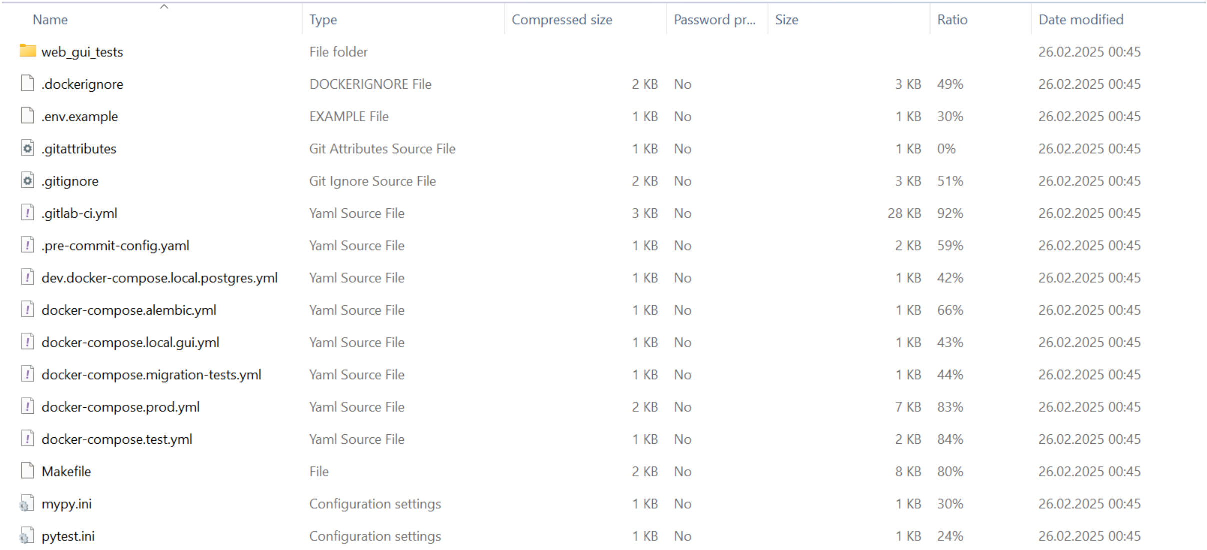
Classic administrators (0)

Solution:

Note: Due to patent proceedings I shall not show end-to-end solution – just main context.



Name	Type	Compressed size	Password pr...	Size	Ratio	Date modified
.github	File folder					26.02.2025 00:45
.vscode	File folder					26.02.2025 00:45
alembic	File folder					26.02.2025 00:45
backend	File folder					26.02.2025 00:45
database	File folder					26.02.2025 00:45
dev_scripts	File folder					26.02.2025 00:45
event_bus_worker	File folder					26.02.2025 00:45
[redacted]api_daemon	File folder					26.02.2025 00:45
nginx	File folder					26.02.2025 00:45
pylint	File folder					26.02.2025 00:45
rabbitmq	File folder					26.02.2025 00:45
readme_images	File folder					26.02.2025 00:45
rest_api	File folder					26.02.2025 00:45
traffic_generator	File folder					26.02.2025 00:45
web_gui	File folder					26.02.2025 00:45
web_gui_tests	File folder					26.02.2025 00:45



Name	Type	Compressed size	Password pr...	Size	Ratio	Date modified
web_gui_tests	File folder					26.02.2025 00:45
.dockerignore	DOCKERIGNORE File	2 KB	No	3 KB	49%	26.02.2025 00:45
.env.example	EXAMPLE File	1 KB	No	1 KB	30%	26.02.2025 00:45
.gitattributes	Git Attributes Source File	1 KB	No	1 KB	0%	26.02.2025 00:45
.gitignore	Git Ignore Source File	2 KB	No	3 KB	51%	26.02.2025 00:45
.gitlab-ci.yml	Yaml Source File	3 KB	No	28 KB	92%	26.02.2025 00:45
.pre-commit-config.yaml	Yaml Source File	1 KB	No	2 KB	59%	26.02.2025 00:45
dev.docker-compose.local.postgres.yml	Yaml Source File	1 KB	No	1 KB	42%	26.02.2025 00:45
docker-compose.alembic.yml	Yaml Source File	1 KB	No	1 KB	66%	26.02.2025 00:45
docker-compose.local.gui.yml	Yaml Source File	1 KB	No	1 KB	43%	26.02.2025 00:45
docker-compose.migration-tests.yml	Yaml Source File	1 KB	No	1 KB	44%	26.02.2025 00:45
docker-compose.prod.yml	Yaml Source File	2 KB	No	7 KB	83%	26.02.2025 00:45
docker-compose.test.yml	Yaml Source File	1 KB	No	2 KB	84%	26.02.2025 00:45
Makefile	File	2 KB	No	8 KB	80%	26.02.2025 00:45
mypy.ini	Configuration settings	1 KB	No	1 KB	30%	26.02.2025 00:45
pytest.ini	Configuration settings	1 KB	No	1 KB	24%	26.02.2025 00:45

Figure 27 Reference to source code of self-developed application with neural network integrated with Graphene diodes

Application with neural network integrated with Graphene diodes.

Using graphene in Deep Brain Stimulation (DBS) offers several significant benefits:

Enhanced Electrical Conductivity - Graphene's exceptional electrical conductivity allows for more efficient and precise stimulation of neural tissues. This can improve the effectiveness of DBS treatments, leading to better outcomes for patients.

High Charge Injection Capacity

Graphene electrodes have a high charge injection capacity, which means they can deliver electrical signals more effectively without causing damage to the surrounding tissue. This is crucial for maintaining the safety and efficacy of DBS.

Reduced Impedance

Graphene's low impedance ensures that electrical signals are transmitted with minimal resistance, enhancing the quality of neural stimulation and recording.

MRI Compatibility

Graphene fiber electrodes exhibit little-to-no artifacts in MRI scans, allowing for full activation pattern mapping of the brain during DBS. This compatibility is essential for understanding the neuromodulator effects of DBS and improving treatment strategies.

Flexibility and Biocompatibility

Graphene is highly flexible and biocompatible, making it suitable for long-term implantation in the brain. Its flexibility allows for better integration with neural tissues, reducing the risk of inflammation and rejection.

Precision and Stability

Graphene-based neurotechnology provides high precision in sensing and stimulating neural activity. This stability is critical for conducting high-precision resections while preserving the patient's functional capacities, such as movement, language, or cognition.

I have used MLflow to train and serve machine learning models in Azure Databricks.

Azure Databricks can be a powerful platform for integrating AI/ML applications with advanced technologies like Deep Brain Stimulation (DBS), graphene diodes, charge transfer, and the Vojta method. Here's how these elements can be combined:

AI/ML on Azure Databricks

Azure Databricks provides a unified environment for developing and deploying AI/ML models. It supports the entire machine learning lifecycle, including data collection, preparation, model development, and deployment1. Key features include:

- Unity Catalog for governance, discovery, versioning, and access control.
- MLflow for tracking model development.
- Mosaic AI Gateway for governing and monitoring access to generative AI models.
- Vector Search for storing and retrieving embeddings, crucial for applications like recommendation systems and image recognition.

Deep Brain Stimulation (DBS)

DBS involves implanting electrodes in specific brain areas to treat neurological conditions. AI/ML models can be used to analyze patient data, optimize stimulation parameters, and predict treatment outcomes. Azure Databricks can facilitate this by:

- Data Analysis: Using Databricks to analyze large datasets from DBS treatments to identify patterns and optimize therapy.
- Model Development: Developing predictive models to personalize DBS settings for individual patients.

Graphene Diodes and Charge Transfer

Graphene diodes, particularly Metal-Insulator-Graphene (MIG) diodes, are used in RF applications due to their high on-current density and work function tunability. Efficient charge transfer doping of graphene can be achieved using materials like cesium carbonate, which significantly reduces sheet resistance. Azure Databricks can facilitate the analysis and optimization of these processes by:

Data Analysis: Using Databricks to analyze experimental data and optimize doping processes.

Simulation: Running simulations to model the behavior of graphene diodes under different conditions.

Vojta Method

The Vojta method is a neurological rehabilitation technique that uses reflexive locomotion to restore motor function. Azure Databricks can support this method by:

- **Data Collection and Analysis:** Collecting and analyzing patient data to track progress and optimize therapy.
- **Machine Learning Models:** Developing models to predict therapy outcomes and personalize treatment plans.

By leveraging Azure Databricks, there is a possibility to integrate AI/ML with advanced technologies like DBS, graphene diodes, charge transfer, and the Vojta method, enhancing the efficiency and effectiveness of the research and clinical applications.

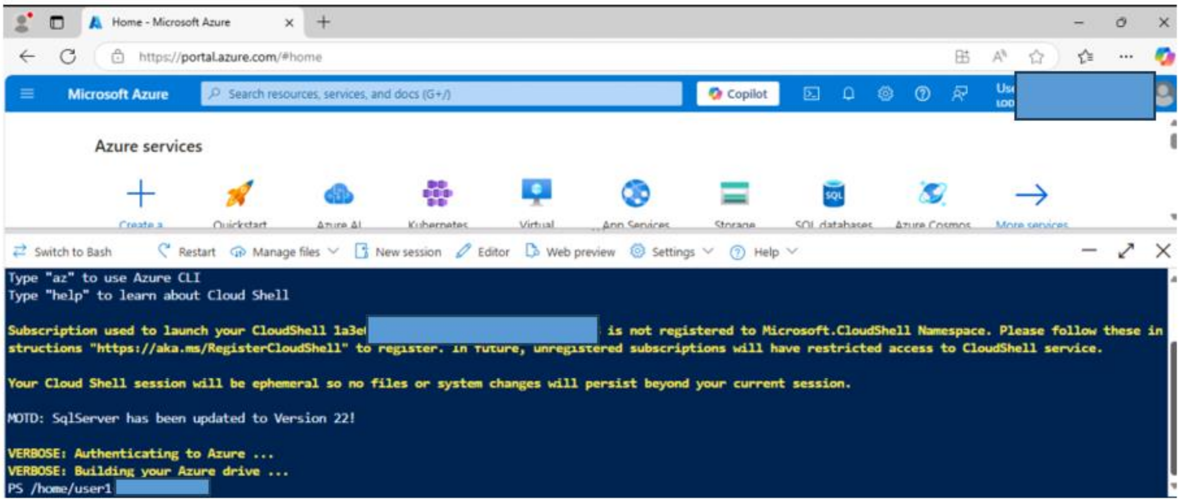
Following steps to have the AI/ML settled down:

- I need an Azure subscription with administrative-level access.
- Provision an Azure Databricks workspace

This experiment includes a script to provision a new Azure Databricks workspace. The script creates a Premium tier Azure Databricks workspace resource in a region where my Azure subscription has sufficient quota for the compute cores required in this experiment; it assumes the user account has sufficient permissions in the subscription to create an Azure Databricks workspace resource. If the script fails due to insufficient quota or permissions, there is an option to create an Azure Databricks workspace interactively in the Azure portal.

Below instructions should be followed:

Use the **[>]** *button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a* **PowerShell**_ environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal, as shown here:



```
Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

Subscription used to launch your CloudShell 1a3e[redacted] is not registered to Microsoft.CloudShell Namespace. Please follow these in
structions "https://aka.ms/RegisterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service.

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.

MOTD: SqlServer has been updated to Version 22!

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/user1 [redacted]
```

Figure 29 Results driven test environment setup

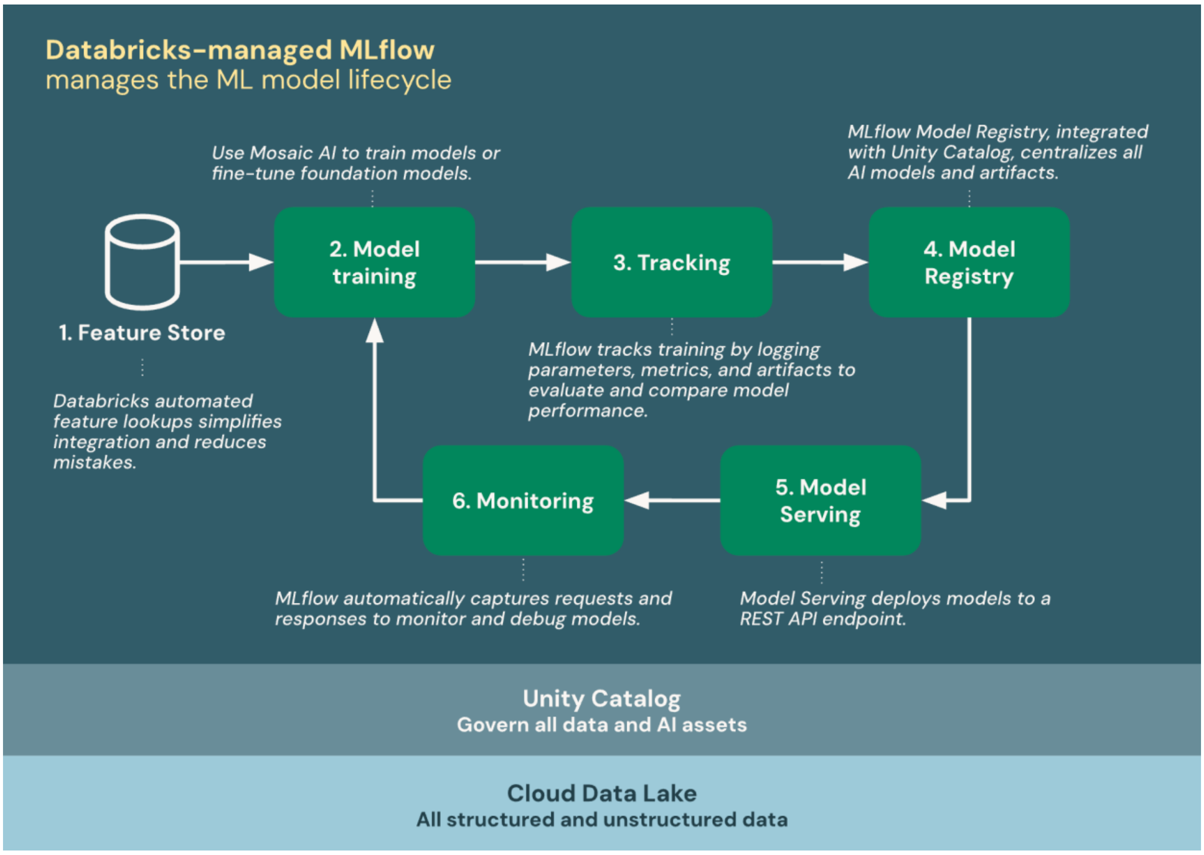
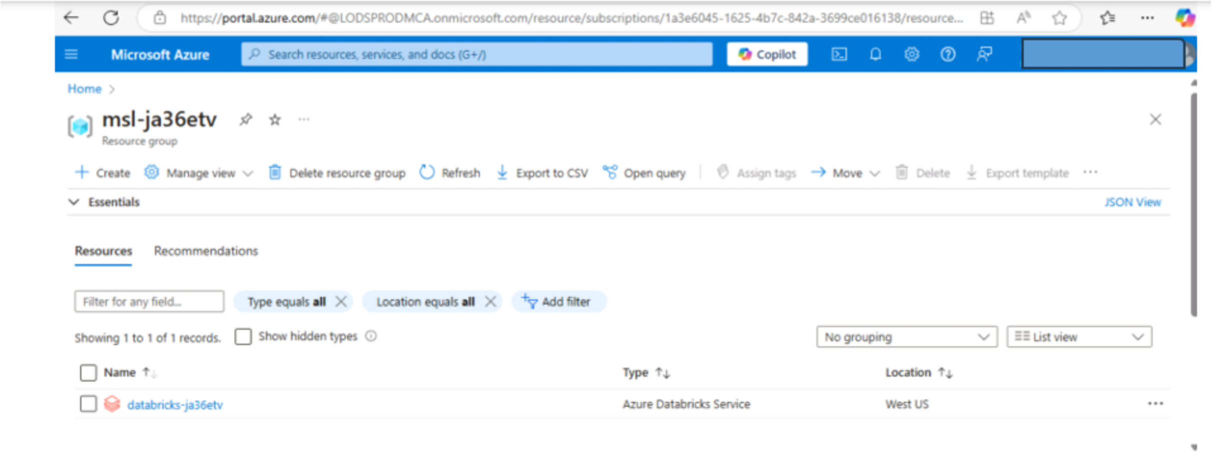

Databricks-managed MLflow
manages the ML model lifecycle
Use Mosaic AI to train models or fine-tune foundation models.
MLflow Model Registry, integrated with Unity Catalog, centralizes all AI models and artifacts.
1. Feature Store
2. Model training
3. Tracking
4. Model Registry
Databricks automated feature lookups simplifies integration and reduces mistakes.
MLflow tracks training by logging parameters, metrics, and artifacts to evaluate and compare model performance.
6. Monitoring
5. Model Serving
MLflow automatically captures requests and responses to monitor and debug models.
Model Serving deploys models to a REST API endpoint.
Unity Catalog
Govern all data and AI assets
Cloud Data Lake
All structured and unstructured data

Figure 30 Source: MLflow for gen AI agent and ML model lifecycle - Azure Databricks | Microsoft Learn

Azure Databricks serves as a distributed processing platform, leveraging the capabilities of Apache Spark clusters to execute parallel data processing across multiple nodes. Each cluster comprises a driver node, responsible for coordinating the work, and several worker nodes tasked with performing processing operations. For the purposes of this experiment, a single-node cluster configuration will be utilized to minimize the consumption of compute resources within a constrained lab environment. It is worth noting that in a production setting, it is standard practice to deploy clusters with multiple worker nodes to enhance processing efficiency and throughput.



https://portal.azure.com/#@LODSPRODMCA.onmicrosoft.com/resource/subscriptions/1a3e6045-1625-4b7c-842a-3699ce016138/resource...

Microsoft Azure | Search resources, services, and docs (G+/) | Copilot | [redacted]

Home >

msl-ja36etv
Resource group

+ Create | Manage view | Delete resource group | Refresh | Export to CSV | Open query | Assign tags | → Move | Delete | Export template | ···

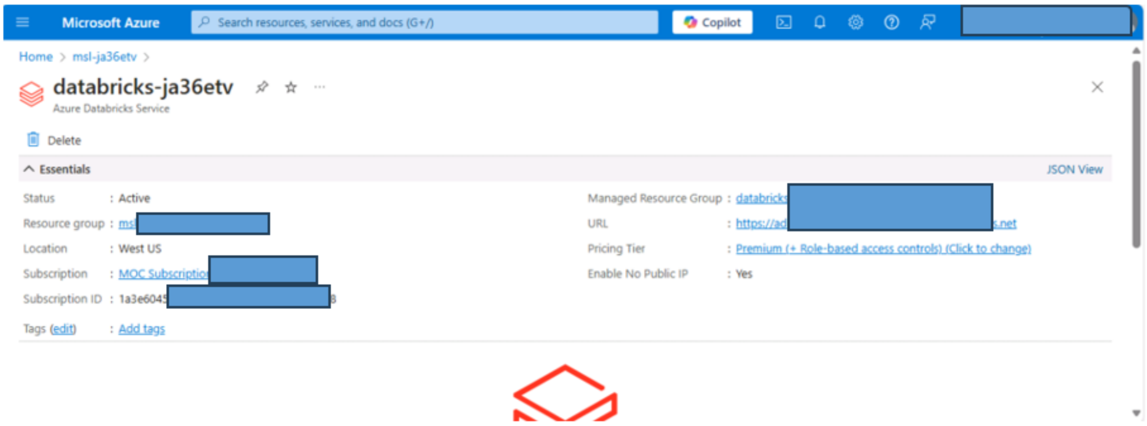
Essentials | JSON View

Resources | Recommendations

Filter for any field... | Type equals **all** ✕ | Location equals **all** ✕ | Add filter

Showing 1 to 1 of 1 records. □ Show hidden types | No grouping | List view

□ Name ↑↓	Type ↑↓	Location ↑↓
□ databricks-ja36etv	Azure Databricks Service	West US



Microsoft Azure | Search resources, services, and docs (G+/) | Copilot | [redacted]

Home > msl-ja36etv >

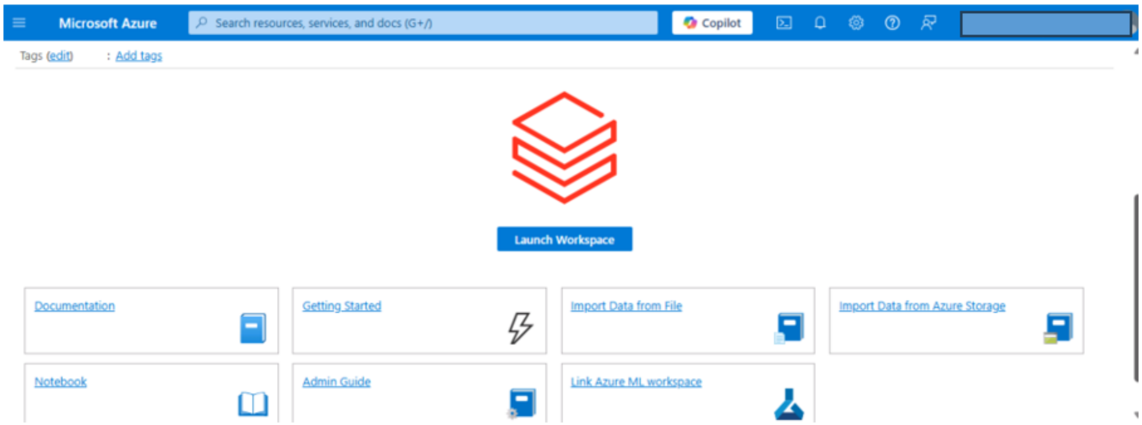
databricks-ja36etv
Azure Databricks Service

Delete

Essentials | JSON View

Status : Active
Resource group : msl[redacted]
Location : West US
Subscription : MOC Subscriptio[redacted]
Subscription ID : 1a3e6045[redacted]8
Tags (edit) : Add tags

Managed Resource Group : databricks[redacted].net
URL : https://ad
Pricing Tier : Premium (+ Role-based access controls) (Click to change)
Enable No Public IP : Yes



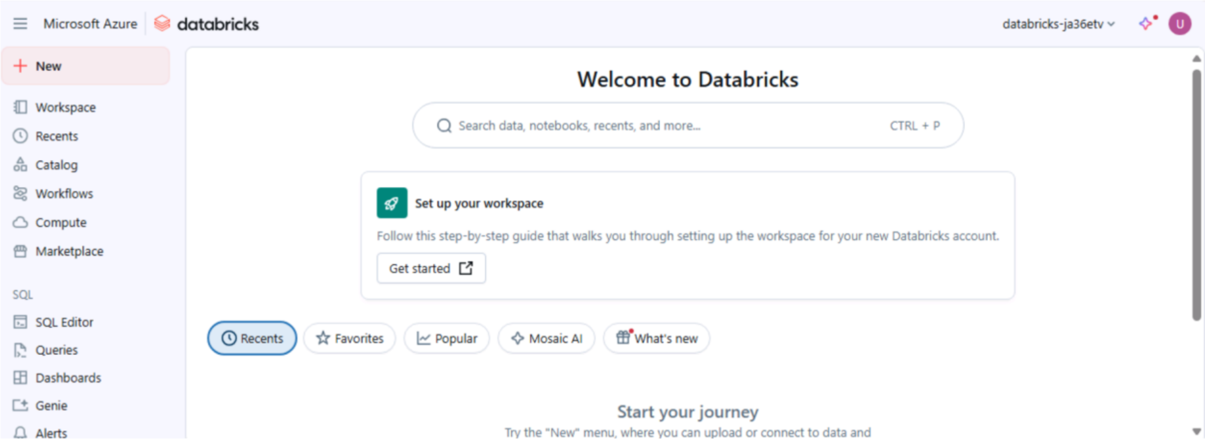
Microsoft Azure | Search resources, services, and docs (G+/) | Copilot | [redacted]

Tags (edit) : Add tags

Launch Workspace

Documentation | Getting Started | Import Data from File | Import Data from Azure Storage

Notebook | Admin Guide | Link Azure ML workspace



Welcome to Databricks

Set up your workspace

Follow this step-by-step guide that walks you through setting up the workspace for your new Databricks account.

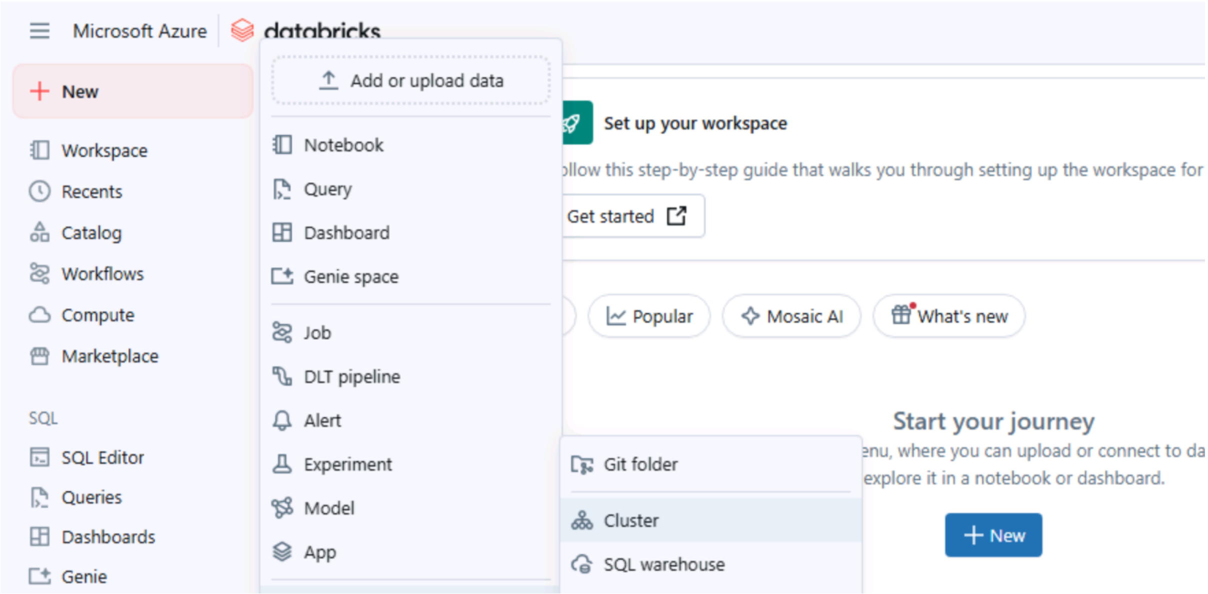
Start your journey

Try the "New" menu, where you can upload or connect to data and

Figure 31 Azure Databrick with ML study of results verification and validation

Explore to use MLflow to train and serve machine learning models in Azure Databricks.

I select Cluster:



Set up your workspace

ollow this step-by-step guide that walks you through setting up the workspace for

Start your journey

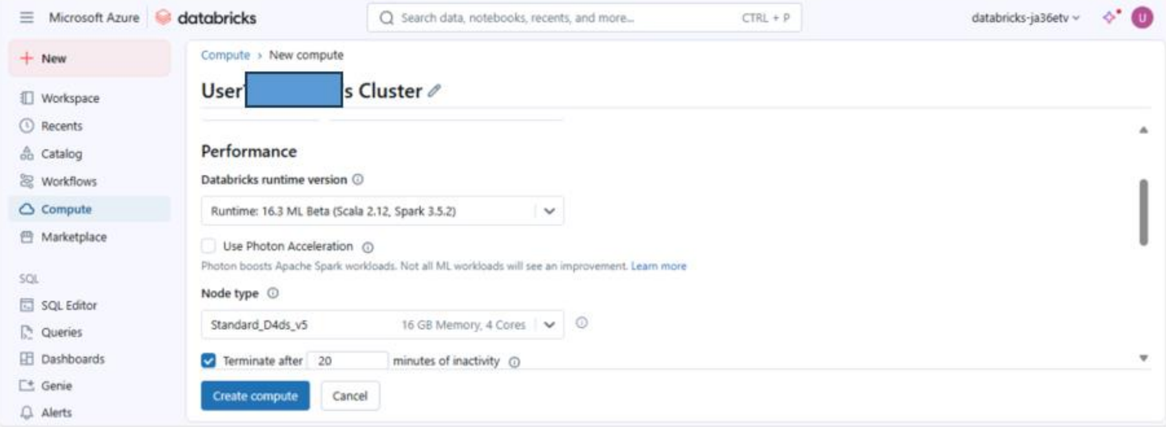
enu, where you can upload or connect to da

explore it in a notebook or dashboard.

In the **New Cluster** page, create a new cluster with the following settings:

- **Cluster name**: *User Name's* cluster (the default cluster name)
- **Policy**: Unrestricted
- **Cluster mode**: Single Node
- **Access mode**: Single user (*with the user account selected*)

- **Databricks runtime version**: *Select the **ML** edition of the latest non-beta version of the runtime (**Not** a Standard runtime version) that:*
  - *Does **not** use a GPU*
  - *Includes Scala > **2.11***
  - *Includes Spark > **3.4***
- **Use Photon Acceleration**: Unselected
- **Node type**: Standard_D4ds_v5
- **Terminate after** *20* **minutes of inactivity**



Compute › New compute

User[redacted]s Cluster

Performance

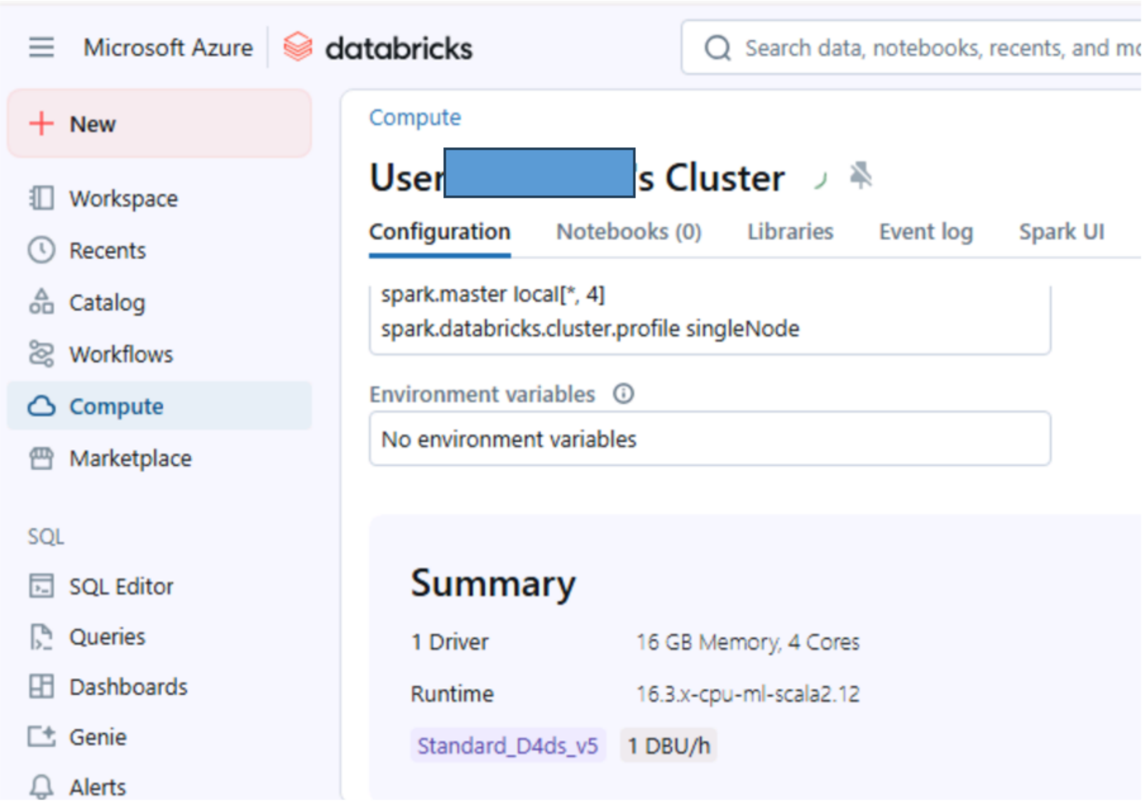
Databricks runtime version: Runtime: 16.3 ML Beta (Scala 2.12, Spark 3.5.2)

□ Use Photon Acceleration

Photon boosts Apache Spark workloads. Not all ML workloads will see an improvement. Learn more

Node type: Standard_D4ds_v5 16 GB Memory, 4 Cores

☑ Terminate after 20 minutes of inactivity



User[redacted]s Cluster

```
spark.master local[*, 4]
spark.databricks.cluster.profile singleNode
```

Environment variables

No environment variables

Summary

1 Driver 16 GB Memory, 4 Cores

Runtime 16.3.x-cpu-ml-scala2.12

Standard_D4ds_v5 1 DBU/h

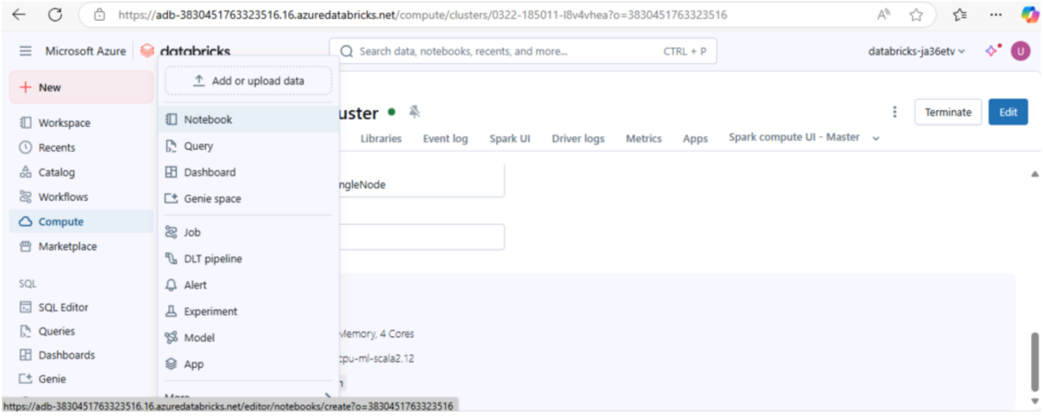
Create a notebook

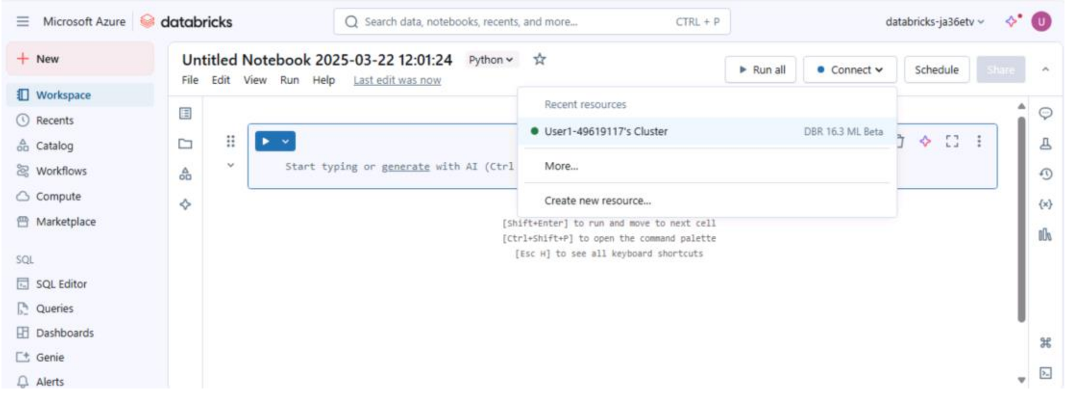
I am going to run code that uses the Spark MLLib library to train a machine learning model, so the first step is to create a new notebook in the workspace.

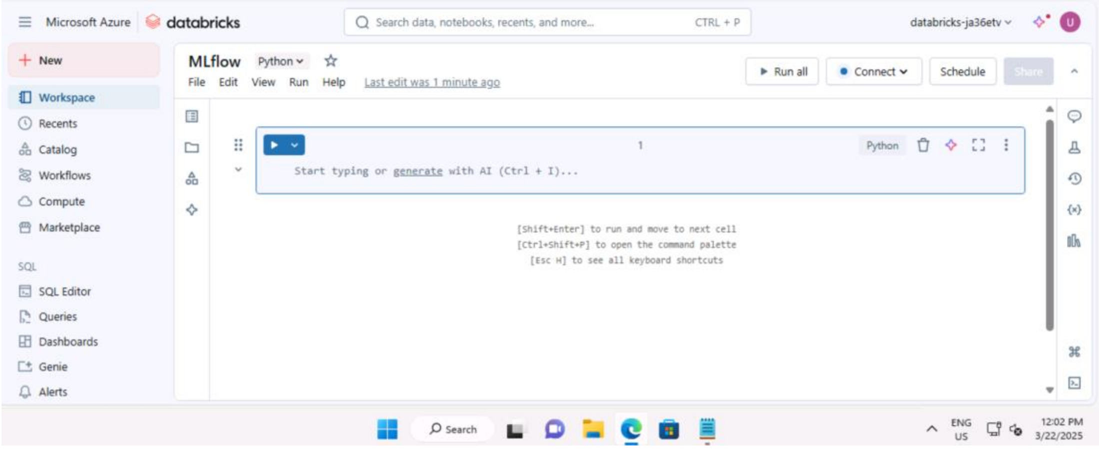
1. In the sidebar, use the **(+) New** link to create a **Notebook**.
2. Change the default notebook name (**Untitled Notebook *[date]***) to **MLflow** and in the **Connect** drop-down list, select the cluster if it is not already selected. If the cluster is not running, it may take a minute or so to start.

   Ingest and prepare data

The scenario for this exercise is based on observations of brain stroke and migraine with aura Infants and Seniors analytical measurements taken from my family members.







In the first cell of the notebook, enter the following code, which uses shell commands to download the patients data from GitHub into the file system used by the cluster.

bash

```
%sh
```

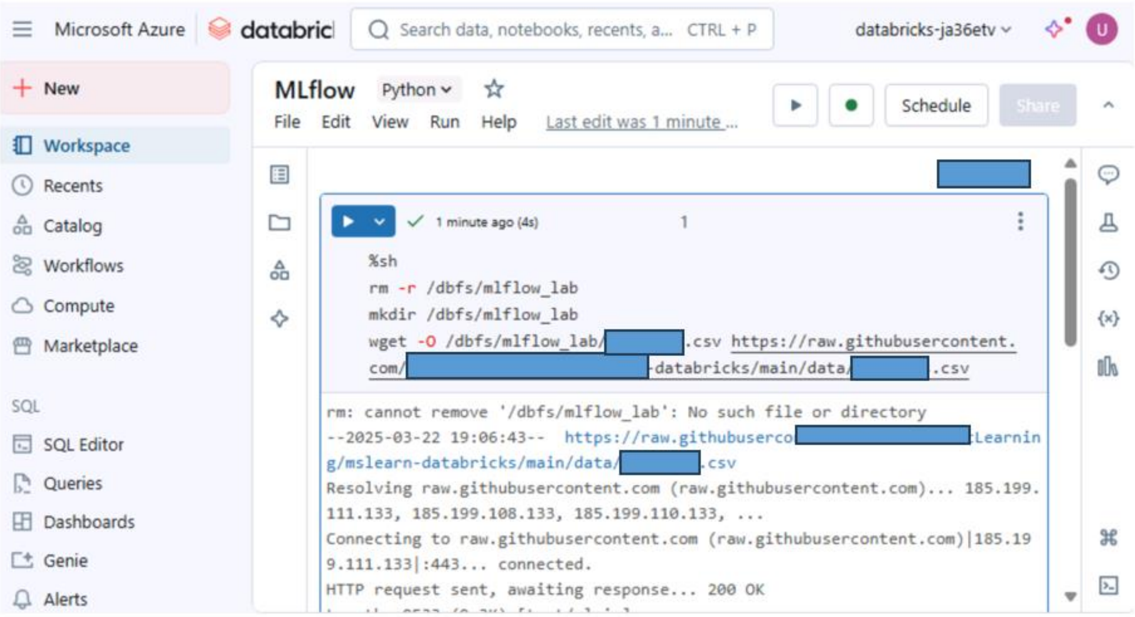
rm -r /dbfs/mlflow_lab

mkdir /dbfs/mlflow_lab

wget -O /dbfs/mlflow_lab/patients.csv

https://raw.githubusercontent.com/[redacted]

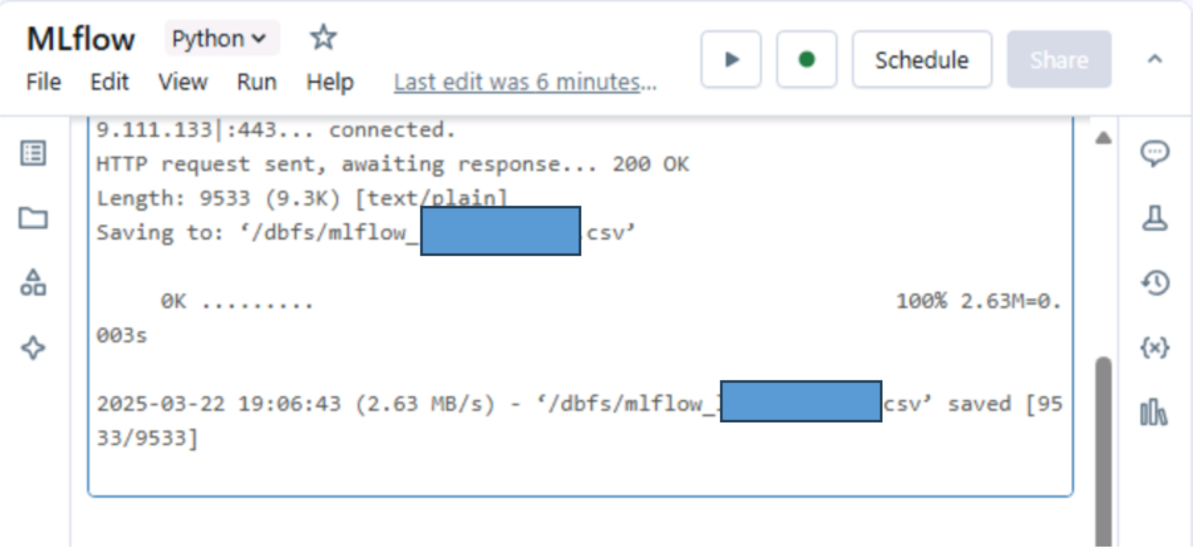
databricks/main/data/patients.csv



[redacted]

```
%sh
rm -r /dbfs/mlflow_lab
mkdir /dbfs/mlflow_lab
wget -O /dbfs/mlflow_lab/[redacted].csv https://raw.githubusercontent.
com/[redacted]databricks/main/data/[redacted].csv
```

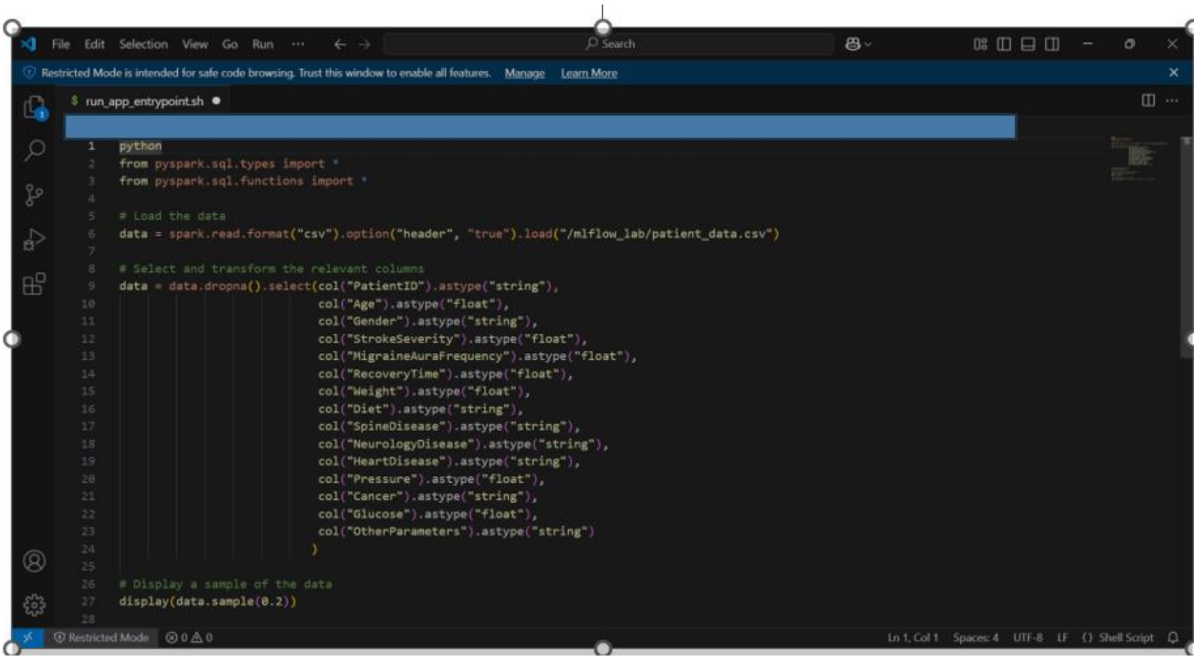
```
rm: cannot remove '/dbfs/mlflow_lab': No such file or directory
--2025-03-22 19:06:43--  https://raw.githubuserco[redacted]Learnin
g/mslearn-databricks/main/data/[redacted].csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.
111.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
```



```
9.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9533 (9.3K) [text/plain]
Saving to: '/dbfs/mlflow_[redacted]csv'

     0K .........                                      100% 2.63M=0.
003s

2025-03-22 19:06:43 (2.63 MB/s) - '/dbfs/mlflow_[redacted]csv' saved [95
33/9533]
```

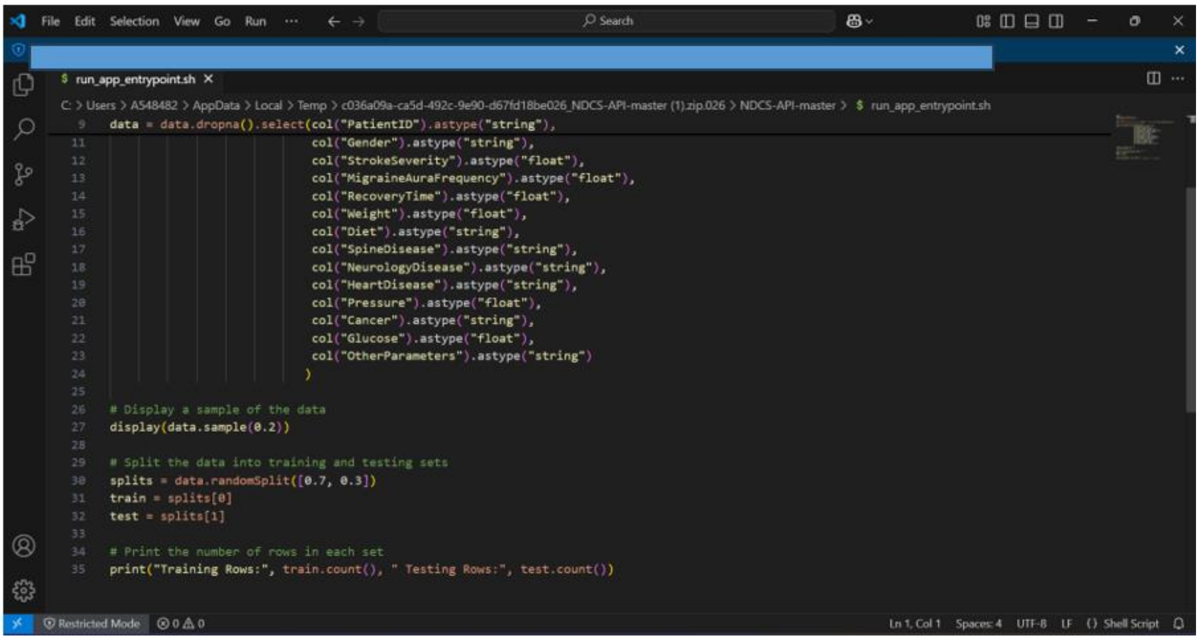
Now I prepare the data for machine learning. Under the existing code cell, use the + icon to add a new code cell. Then in the new cell, enter and run the following code to:

- Remove any incomplete rows
- Apply appropriate data types
- View a random sample of the data
- Split the data into two datasets: one for training, and another for testing.



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```
python
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Load the data
data = spark.read.format("csv").option("header", "true").load("/mlflow_lab/patient_data.csv")

# Select and transform the relevant columns
data = data.dropna().select(col("PatientID").astype("string"),
                            col("Age").astype("float"),
                            col("Gender").astype("string"),
                            col("StrokeSeverity").astype("float"),
                            col("MigraineAuraFrequency").astype("float"),
                            col("RecoveryTime").astype("float"),
                            col("Weight").astype("float"),
                            col("Diet").astype("string"),
                            col("SpineDisease").astype("string"),
                            col("NeurologyDisease").astype("string"),
                            col("HeartDisease").astype("string"),
                            col("Pressure").astype("float"),
                            col("Cancer").astype("string"),
                            col("Glucose").astype("float"),
                            col("OtherParameters").astype("string")
                           )

# Display a sample of the data
display(data.sample(0.2))
```

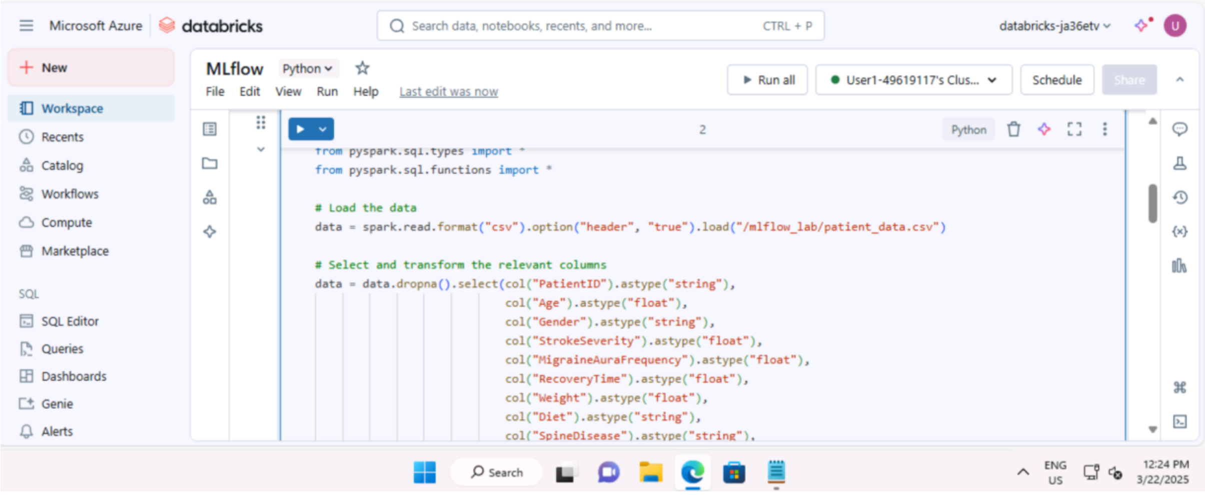

File Edit Selection View Go Run ···
Search
run_app_entrypoint.sh
C: > Users > A548482 > AppData > Local > Temp > c036a09a-ca5d-492c-9e90-d67fd18be026_NDCS-API-master (1).zip.026 > NDCS-API-master > $ run_app_entrypoint.sh
Restricted Mode
⊗ 0 ⚠ 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Shell Script

```
data = data.dropna().select(col("PatientID").astype("string"),
                            col("Gender").astype("string"),
                            col("StrokeSeverity").astype("float"),
                            col("MigraineAuraFrequency").astype("float"),
                            col("RecoveryTime").astype("float"),
                            col("Weight").astype("float"),
                            col("Diet").astype("string"),
                            col("SpineDisease").astype("string"),
                            col("NeurologyDisease").astype("string"),
                            col("HeartDisease").astype("string"),
                            col("Pressure").astype("float"),
                            col("Cancer").astype("string"),
                            col("Glucose").astype("float"),
                            col("OtherParameters").astype("string")
                           )

# Display a sample of the data
display(data.sample(0.2))

# Split the data into training and testing sets
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]

# Print the number of rows in each set
print("Training Rows:", train.count(), " Testing Rows:", test.count())
```

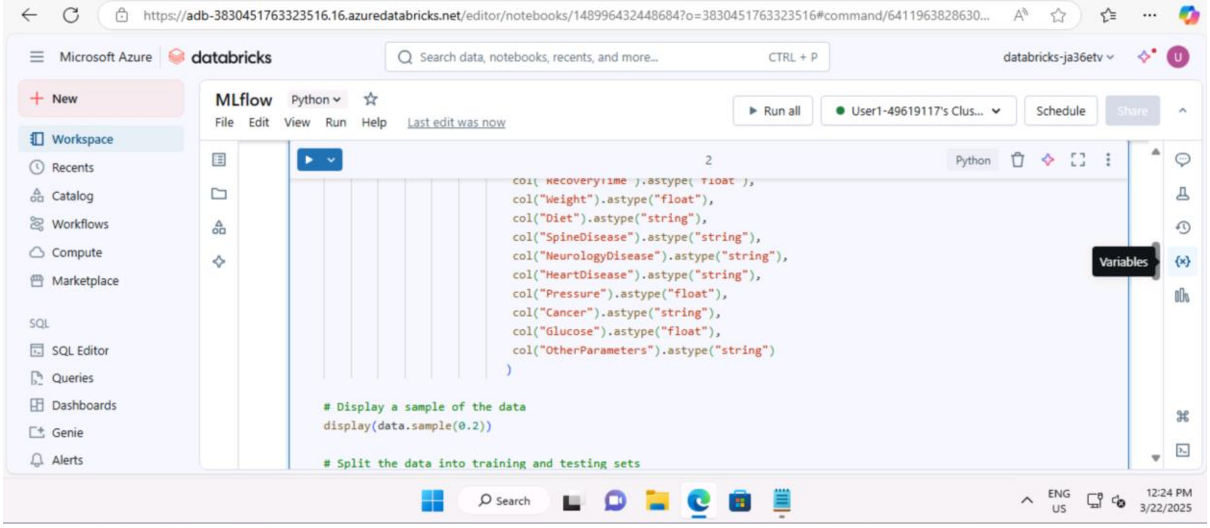
Run an MLflow experiment

MLflow allows for the execution of experiments that monitor the model training process and log evaluation metrics. The capability to record specific details of model training runs is highly beneficial in the iterative development of effective machine learning models.

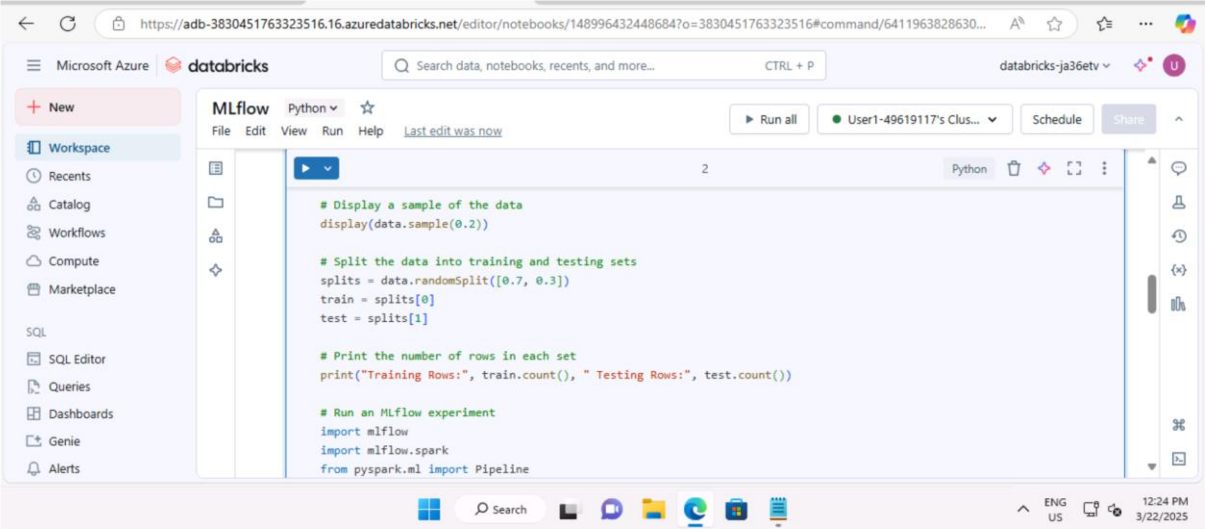
fStandard libraries and techniques can be employed to train and evaluate a model, such as Spark MLLib, within the framework of an MLflow experiment. This approach incorporates additional commands to log crucial metrics and information throughout the process.



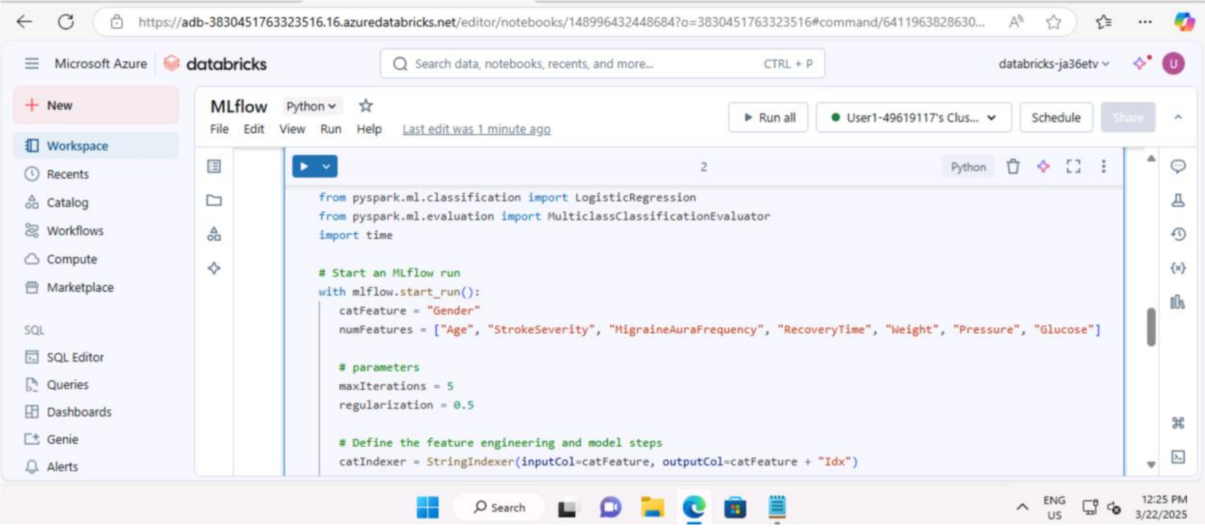
```
from pyspark.sql.types import *
from pyspark.sql.functions import *

# Load the data
data = spark.read.format("csv").option("header", "true").load("/mlflow_lab/patient_data.csv")

# Select and transform the relevant columns
data = data.dropna().select(col("PatientID").astype("string"),
                            col("Age").astype("float"),
                            col("Gender").astype("string"),
                            col("StrokeSeverity").astype("float"),
                            col("MigraineAuraFrequency").astype("float"),
                            col("RecoveryTime").astype("float"),
                            col("Weight").astype("float"),
                            col("Diet").astype("string"),
                            col("SpineDisease").astype("string"),
```



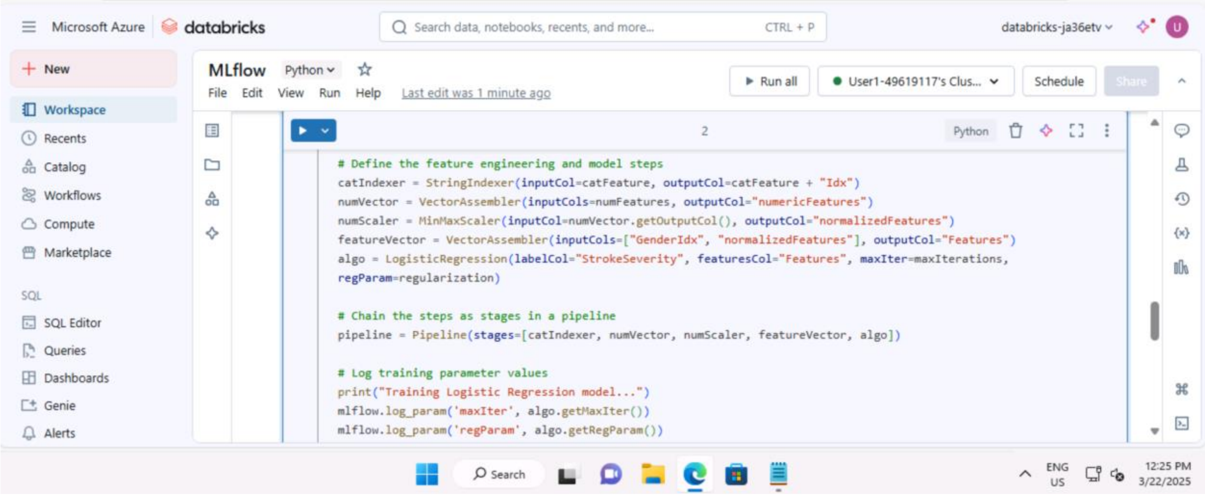
```
                            col("RecoveryTime").astype("float"),
                            col("Weight").astype("float"),
                            col("Diet").astype("string"),
                            col("SpineDisease").astype("string"),
                            col("NeurologyDisease").astype("string"),
                            col("HeartDisease").astype("string"),
                            col("Pressure").astype("float"),
                            col("Cancer").astype("string"),
                            col("Glucose").astype("float"),
                            col("OtherParameters").astype("string")
                           )

# Display a sample of the data
display(data.sample(0.2))

# Split the data into training and testing sets
```



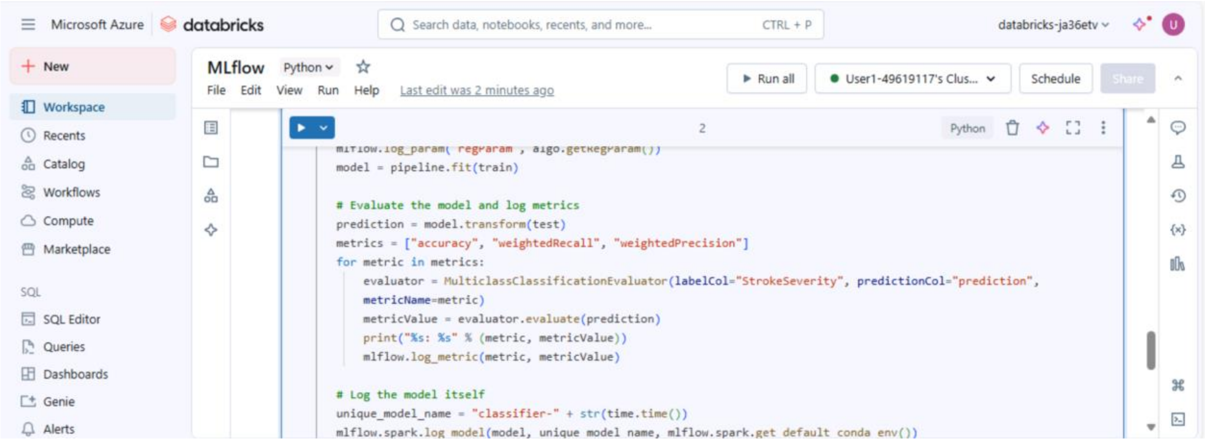
```
# Display a sample of the data
display(data.sample(0.2))

# Split the data into training and testing sets
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]

# Print the number of rows in each set
print("Training Rows:", train.count(), " Testing Rows:", test.count())

# Run an MLflow experiment
import mlflow
import mlflow.spark
from pyspark.ml import Pipeline
```



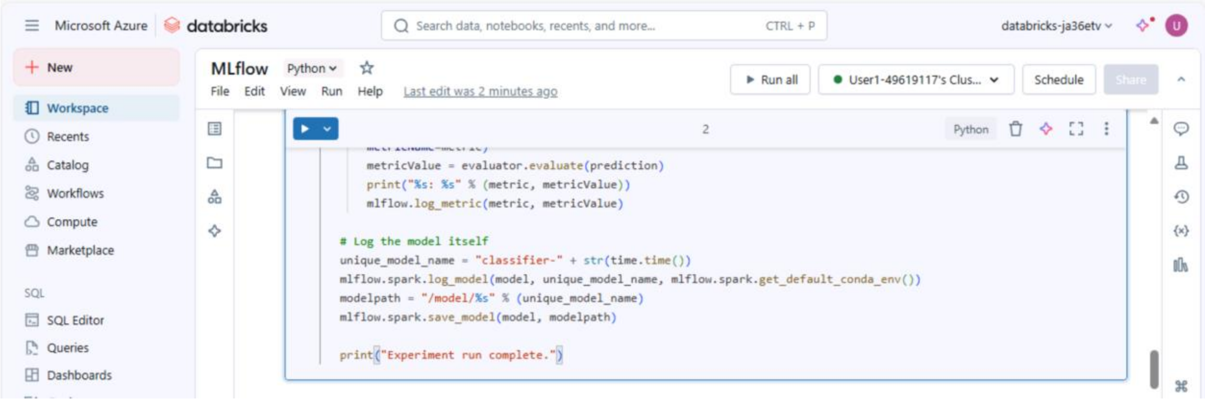
```python
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import time

# Start an MLflow run
with mlflow.start_run():
    catFeature = "Gender"
    numFeatures = ["Age", "StrokeSeverity", "MigraineAuraFrequency", "RecoveryTime", "Weight", "Pressure", "Glucose"]

    # parameters
    maxIterations = 5
    regularization = 0.5

    # Define the feature engineering and model steps
    catIndexer = StringIndexer(inputCol=catFeature, outputCol=catFeature + "Idx")
```



```python
    # Define the feature engineering and model steps
    catIndexer = StringIndexer(inputCol=catFeature, outputCol=catFeature + "Idx")
    numVector = VectorAssembler(inputCols=numFeatures, outputCol="numericFeatures")
    numScaler = MinMaxScaler(inputCol=numVector.getOutputCol(), outputCol="normalizedFeatures")
    featureVector = VectorAssembler(inputCols=["GenderIdx", "normalizedFeatures"], outputCol="Features")
    algo = LogisticRegression(labelCol="StrokeSeverity", featuresCol="Features", maxIter=maxIterations,
    regParam=regularization)

    # Chain the steps as stages in a pipeline
    pipeline = Pipeline(stages=[catIndexer, numVector, numScaler, featureVector, algo])

    # Log training parameter values
    print("Training Logistic Regression model...")
    mlflow.log_param('maxIter', algo.getMaxIter())
    mlflow.log_param('regParam', algo.getRegParam())
```



```python
    mlflow.log_param('regParam', algo.getRegParam())
    model = pipeline.fit(train)

    # Evaluate the model and log metrics
    prediction = model.transform(test)
    metrics = ["accuracy", "weightedRecall", "weightedPrecision"]
    for metric in metrics:
        evaluator = MulticlassClassificationEvaluator(labelCol="StrokeSeverity", predictionCol="prediction",
        metricName=metric)
        metricValue = evaluator.evaluate(prediction)
        print("%s: %s" % (metric, metricValue))
        mlflow.log_metric(metric, metricValue)

    # Log the model itself
    unique_model_name = "classifier-" + str(time.time())
    mlflow.spark.log_model(model, unique_model_name, mlflow.spark.get_default_conda_env())
```

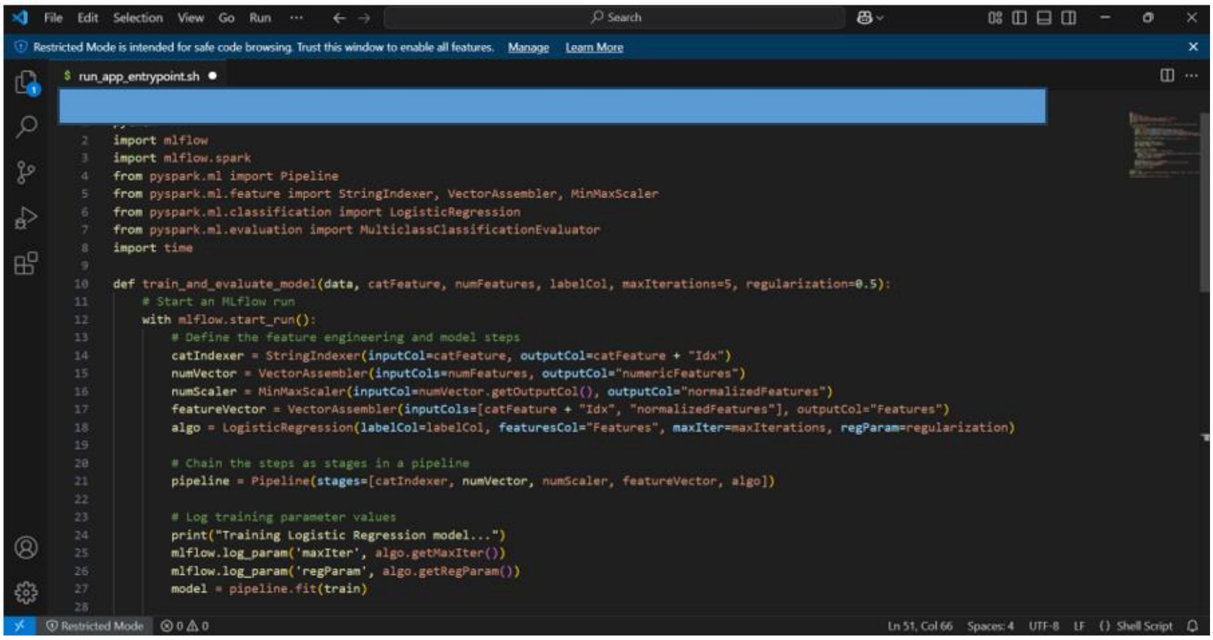


```
        metricValue = evaluator.evaluate(prediction)
        print("%s: %s" % (metric, metricValue))
        mlflow.log_metric(metric, metricValue)

    # Log the model itself
    unique_model_name = "classifier-" + str(time.time())
    mlflow.spark.log_model(model, unique_model_name, mlflow.spark.get_default_conda_env())
    modelpath = "/model/%s" % (unique_model_name)
    mlflow.spark.save_model(model, modelpath)

    print("Experiment run complete.")
```

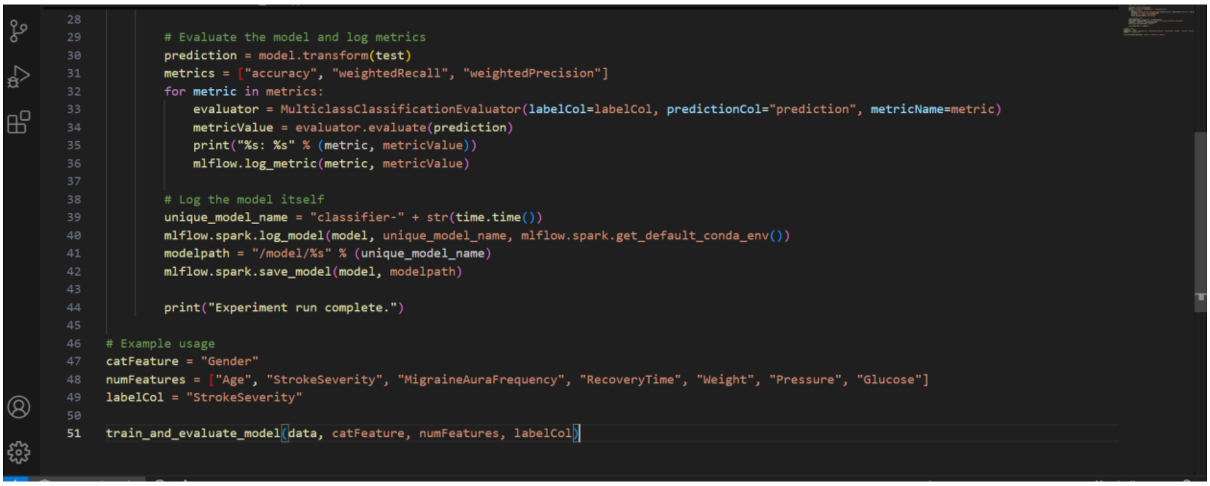
Create a function

In machine learning projects, data scientists often try training models with different parameters, logging the results each time. To accomplish that, it's common to create a function that encapsulates the training process and call it with the parameters I want to try.

In a new cell, run the following code to create a function based on the training code I have used previously



```
import mlflow
import mlflow.spark
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, VectorAssembler, MinMaxScaler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import time

def train_and_evaluate_model(data, catFeature, numFeatures, labelCol, maxIterations=5, regularization=0.5):
    # Start an MLflow run
    with mlflow.start_run():
        # Define the feature engineering and model steps
        catIndexer = StringIndexer(inputCol=catFeature, outputCol=catFeature + "Idx")
        numVector = VectorAssembler(inputCols=numFeatures, outputCol="numericFeatures")
        numScaler = MinMaxScaler(inputCol=numVector.getOutputCol(), outputCol="normalizedFeatures")
        featureVector = VectorAssembler(inputCols=[catFeature + "Idx", "normalizedFeatures"], outputCol="Features")
        algo = LogisticRegression(labelCol=labelCol, featuresCol="Features", maxIter=maxIterations, regParam=regularization)

        # Chain the steps as stages in a pipeline
        pipeline = Pipeline(stages=[catIndexer, numVector, numScaler, featureVector, algo])

        # Log training parameter values
        print("Training Logistic Regression model...")
        mlflow.log_param('maxIter', algo.getMaxIter())
        mlflow.log_param('regParam', algo.getRegParam())
        model = pipeline.fit(train)

```



```python

        # Evaluate the model and log metrics
        prediction = model.transform(test)
        metrics = ["accuracy", "weightedRecall", "weightedPrecision"]
        for metric in metrics:
            evaluator = MulticlassClassificationEvaluator(labelCol=labelCol, predictionCol="prediction", metricName=metric)
            metricValue = evaluator.evaluate(prediction)
            print("%s: %s" % (metric, metricValue))
            mlflow.log_metric(metric, metricValue)

        # Log the model itself
        unique_model_name = "classifier-" + str(time.time())
        mlflow.spark.log_model(model, unique_model_name, mlflow.spark.get_default_conda_env())
        modelpath = "/model/%s" % (unique_model_name)
        mlflow.spark.save_model(model, modelpath)

        print("Experiment run complete.")

# Example usage
catFeature = "Gender"
numFeatures = ["Age", "StrokeSeverity", "MigraineAuraFrequency", "RecoveryTime", "Weight", "Pressure", "Glucose"]
labelCol = "StrokeSeverity"

train_and_evaluate_model(data, catFeature, numFeatures, labelCol)
```

I have used the train_and_evaluate_model to train and evaluate the dataset with MLflow. Adjust parameters as needed.

And test the solution

python

train_patients_model(train, test, 10, 0.2)

Register and Deploy a Model with MLflow

In addition to tracking details of training experiment runs, I can use MLflow to manage the machine learning models I've trained. I've already logged the model trained by each experiment run. I can also register models and deploy them so they can be served to client applications.

Note: Model serving is only supported in Azure Databricks Premium workspaces and is restricted to certain regions.

- ✓ View the details page for the most recent experiment run.
- ✓ Use the **Register Model** button to register the model that was logged in that experiment and when prompted, create a new model named **Patient DBS Predictor**.

- ✓ When the model has been registered, view the **Models** page (in the navigation bar on the left) and select the **Patient DBS Predictor** model.
- ✓ In the page for the **Patient DBS Predictor** model, use the **Use model for inference** button to create a new real-time endpoint with the following settings:

- **Model:** Patient DBS Predictor
- **Model version:** 1
- **Endpoint:** predict-patient
- **Compute size:** Small

The serving endpoint is hosted in a new cluster, which may take several minutes to create. When the endpoint has been created, use the **Query endpoint** button at the top right to open an interface from which I can test the endpoint. Then in the test interface, on the **Browser** tab, enter the following JSON request and use the **Send Request** button to call the endpoint and generate a prediction.

```
{
  "dataframe_records": [
    {
      "Gender": "Female",
      "Age": 65,
      "StrokeSeverity": 3.5,
      "MigraineAuraFrequency": 2,
      "RecoveryTime": 180,
      "Weight": 70,
      "Pressure": 120,

      "Glucose": 5.6

    }

  ]

}
```

Experiment with a few different values for the patient features and observe the results that are returned. Then, close the test interface.

Machine Learning with Azure Databricks

Train a deep learning model

In this exercise, I'll use the **PyTorch** library to train a deep learning model in Azure Databricks. Then I'll use the **Horovod** library to distribute deep learning training across multiple worker nodes in a cluster.

Optimizing Deep Learning Models Using Azure Databricks

Introduction

This study investigates the use of Azure Databricks for optimizing deep learning model training. By leveraging the combination of PyTorch and Horovod libraries, this research aims to provide a comprehensive guide on how to effectively distribute deep learning training across multiple worker nodes, ensuring efficiency and scalability.

Experimentation and Results

I began by experimenting with various patient features to observe the impact of different values on model performance. The key metrics utilized in these experiments included "Weight," "Pressure," and "Glucose."

Patient Features

- Weight: 70
- Pressure: 120
- Glucose: 5.6

Deep Learning Model Training

Building on the initial experiments, I used the PyTorch library within Azure Databricks to train a deep learning model. PyTorch's flexibility and ease of use enabled efficient model construction and training.

Horovod for Distributed Training

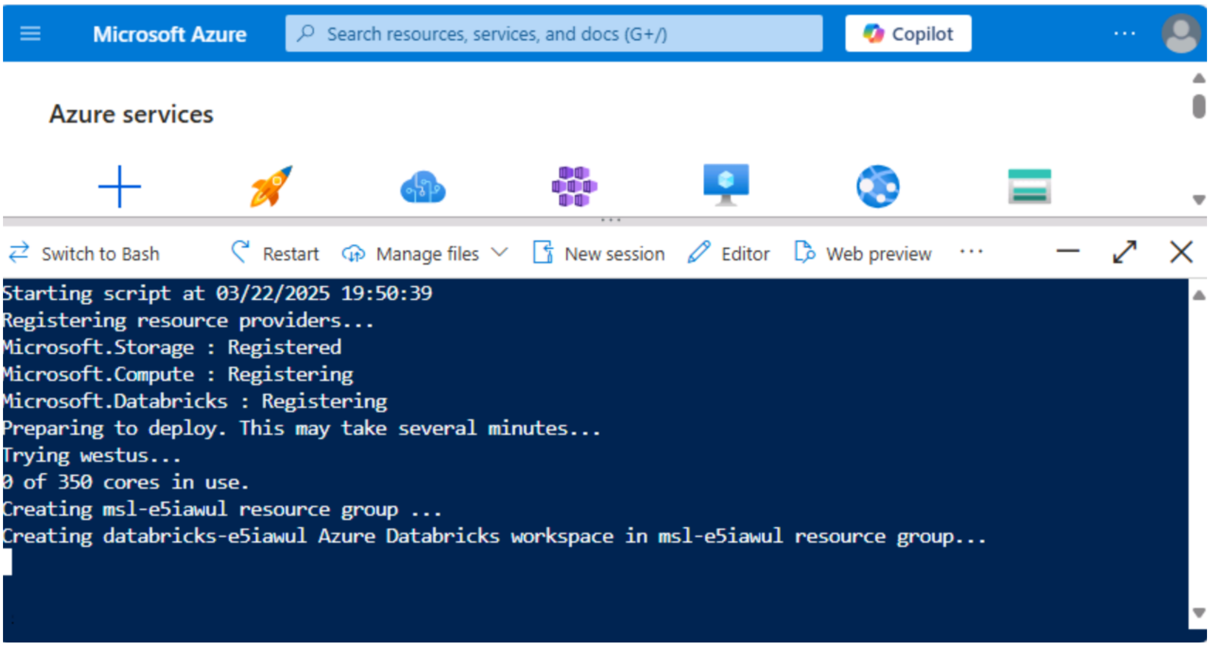
After establishing the model architecture, I utilized the Horovod library to distribute the training process across multiple worker nodes in the cluster. Horovod's seamless integration with Databricks allowed for parallel computation, significantly reducing training time and enhancing model accuracy.

Conclusion of researches

My research highlights the potential of Azure Databricks in optimizing deep learning model training through its robust infrastructure and integrated libraries. The combination of PyTorch and Horovod offers a powerful solution for scaling deep learning tasks, providing valuable insights for future applications in machine learning and artificial intelligence in DBS - Deep Brain Stimulation.

It is recommended to sustain own code in highly secured environment – like for example Git or GitHub, any other change control and version control code repository.

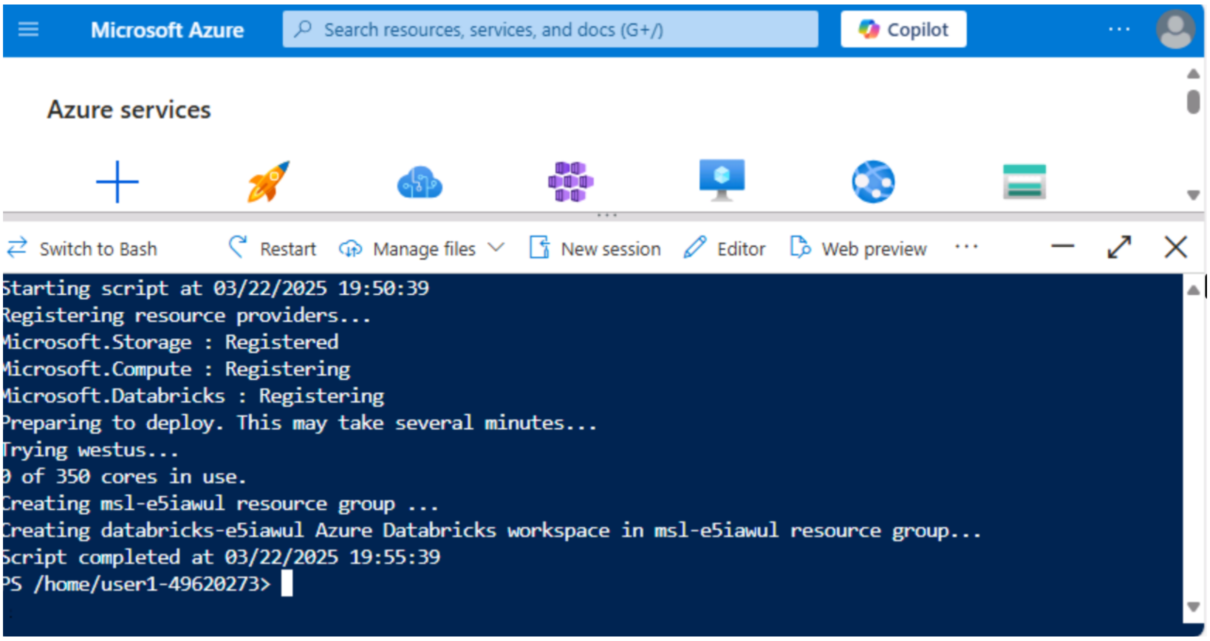
After the repo has been cloned, enter the following command to run the **setup.ps1** script, which provisions an Azure Databricks workspace in an available region.



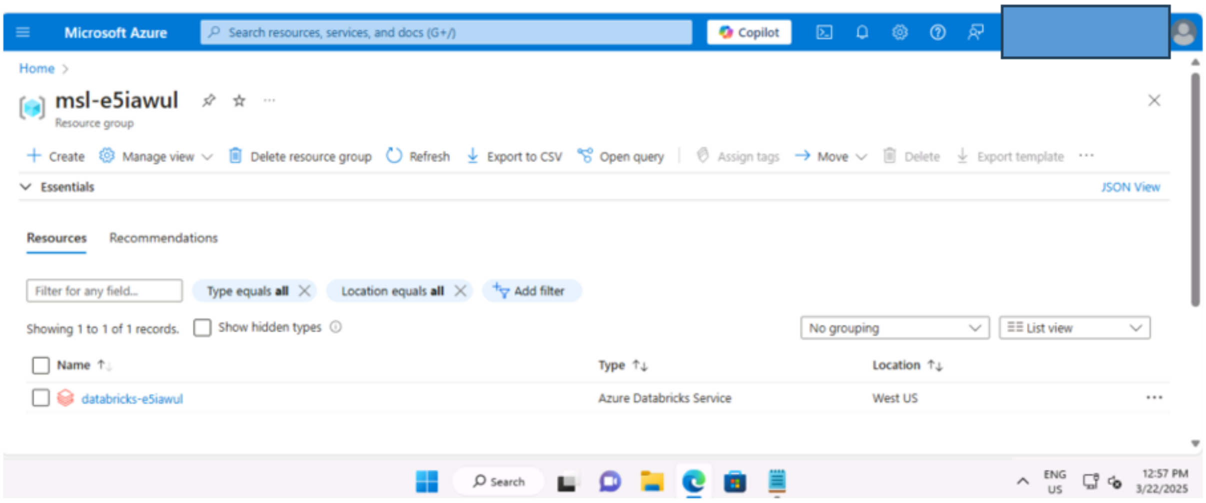
```
Starting script at 03/22/2025 19:50:39
Registering resource providers...
Microsoft.Storage : Registered
Microsoft.Compute : Registering
Microsoft.Databricks : Registering
Preparing to deploy. This may take several minutes...
Trying westus...
0 of 350 cores in use.
Creating msl-e5iawul resource group ...
Creating databricks-e5iawul Azure Databricks workspace in msl-e5iawul resource group...
```

Create a cluster

Azure Databricks uses Apache Spark clusters to process data in parallel across multiple nodes. Each cluster has a driver node for coordination and worker nodes for processing. For this exercise, I'll create a single-node cluster to use fewer resources. In production, I'd usually have multiple worker nodes.



```
Starting script at 03/22/2025 19:50:39
Registering resource providers...
Microsoft.Storage : Registered
Microsoft.Compute : Registering
Microsoft.Databricks : Registering
Preparing to deploy. This may take several minutes...
Trying westus...
0 of 350 cores in use.
Creating msl-e5iawul resource group ...
Creating databricks-e5iawul Azure Databricks workspace in msl-e5iawul resource group...
Script completed at 03/22/2025 19:55:39
PS /home/user1-49620273>
```

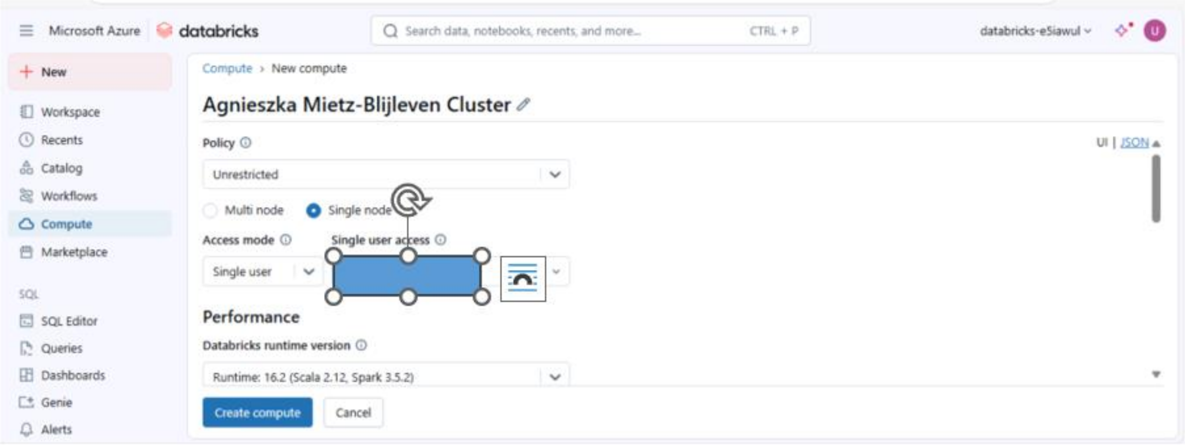


[redacted]

Showing 1 to 1 of 1 records. □ Show hidden types

□ Name	Type	Location
□ databricks-e5iawul	Azure Databricks Service	West US

Creating the cluster:



Compute › New compute

Agnieszka Mietz-Blijleven Cluster

Policy: Unrestricted

○ Multi node ● Single node

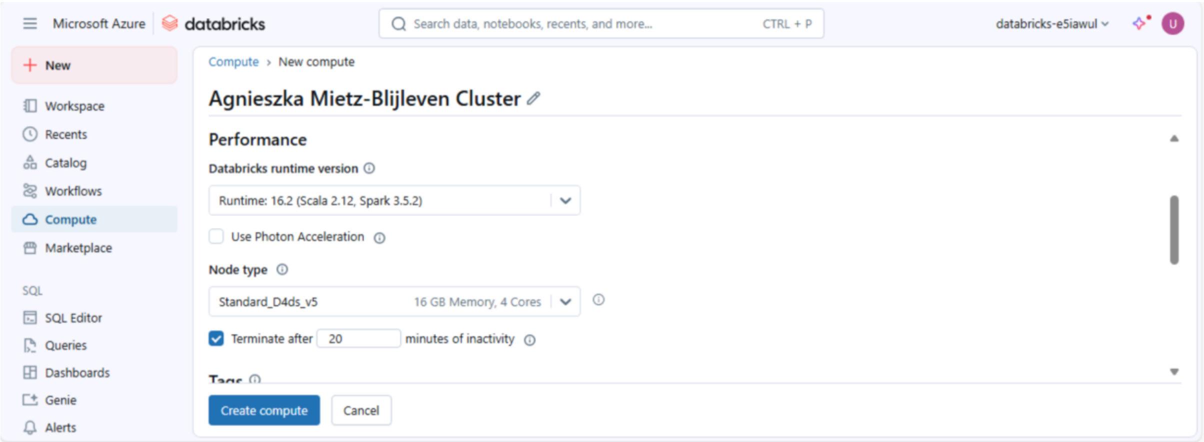
Access mode: Single user

Single user access

Performance

Databricks runtime version: Runtime: 16.2 (Scala 2.12, Spark 3.5.2)

Adjusting for deep training model parameters



Compute › New compute

Agnieszka Mietz-Blijleven Cluster

Performance

Databricks runtime version: Runtime: 16.2 (Scala 2.12, Spark 3.5.2)

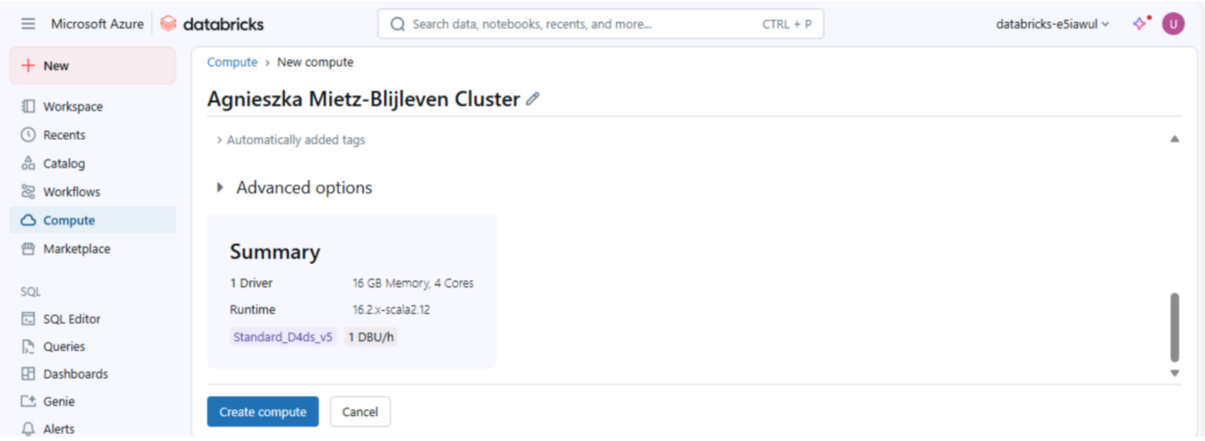
□ Use Photon Acceleration

Node type: Standard_D4ds_v5 16 GB Memory, 4 Cores

☑ Terminate after 20 minutes of inactivity

Tags

Off course for more advanced ML models we have to have much more storage capacity than taken for sampled data:

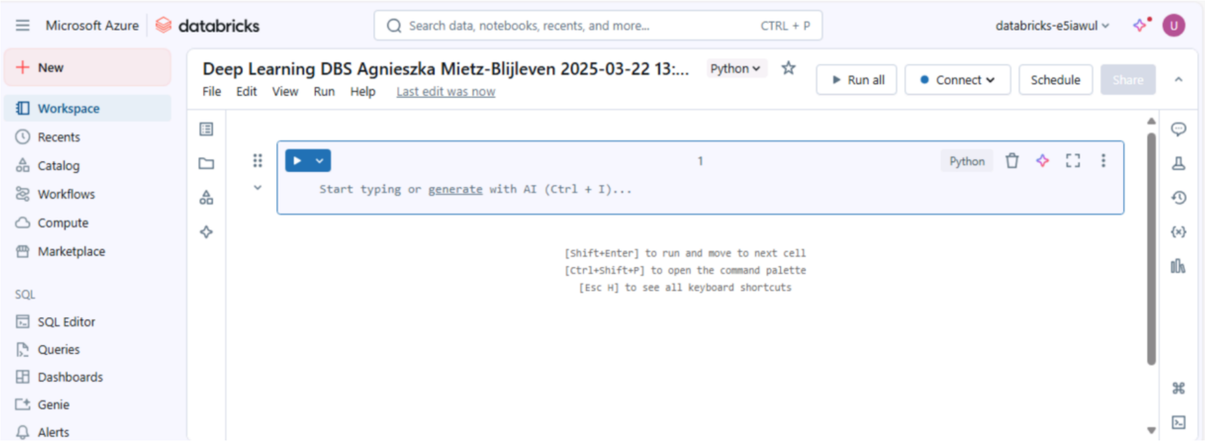


Agnieszka Mietz-Blijleven Cluster

Summary

1 Driver	16 GB Memory, 4 Cores
Runtime	16.2.x-scala2.12
Standard_D4ds_v5	1 DBU/h

I am creating the Notebook



Deep Learning DBS Agnieszka Mietz-Blijleven 2025-03-22 13:...

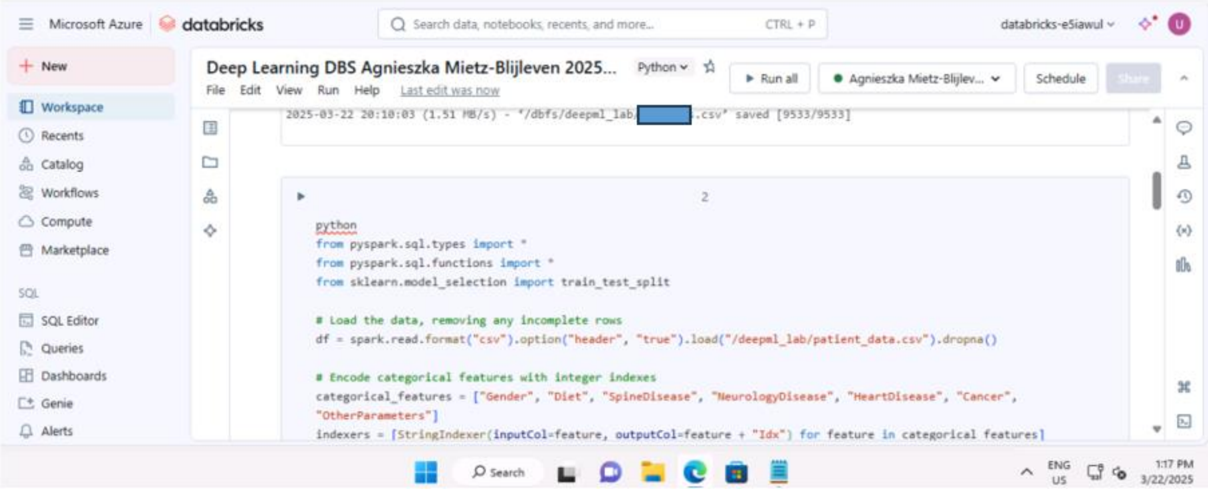
```
Start typing or generate with AI (Ctrl + I)...
```

```
[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts
```

To repeat an experiment on deep learning level I had to prepare the data for machine learning. Under the existing code cell, use the + icon to add a new code cell. Then in the new cell, enter and run the following code to:

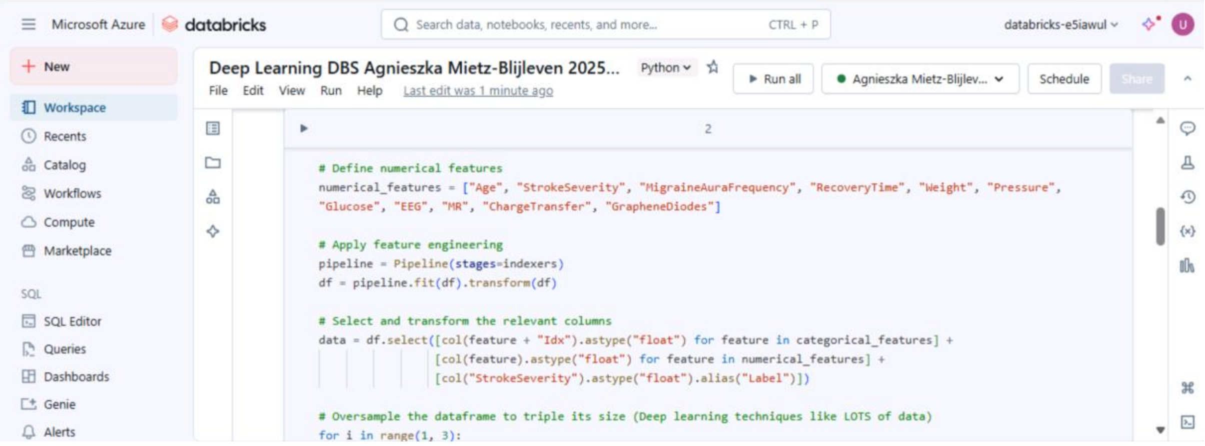
- Remove any incomplete rows
- Encode the (string) island name as an integer
- Apply appropriate data types
- Normalize the numeric data to a similar scale
- Split the data into two datasets: one for training, and another for testing.

To handle advanced parameters related to NLP, neurons, brain strokes, brain migraine with aura for infants and seniors, organism parameters (blood, pressure, EEG, MR), brain disease symptoms, charge transfer, Vojta method, and graphene diodes for deep brain stimulation:

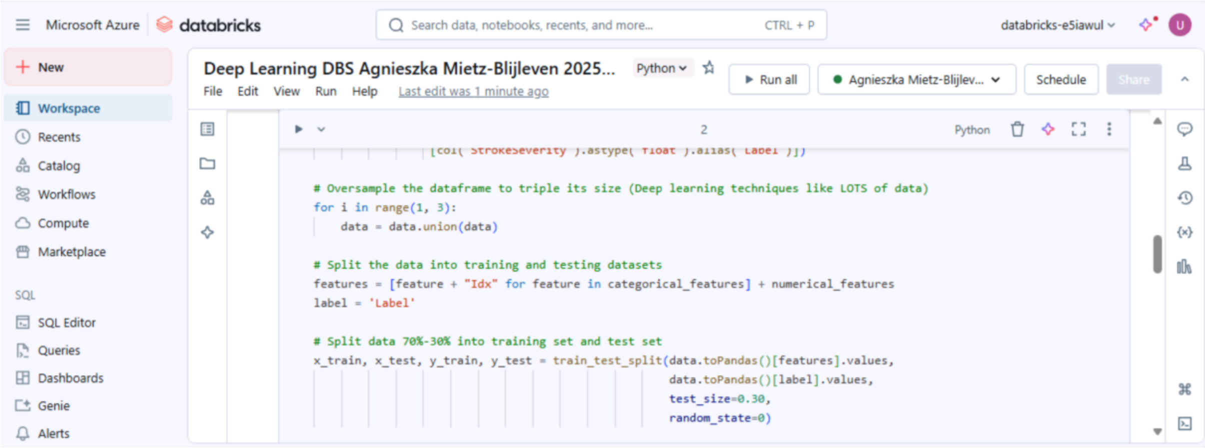


```
2025-03-22 20:10:03 (1.51 MB/s) - '/dbfs/deepml_lab/[redacted].csv' saved [9533/9533]
```

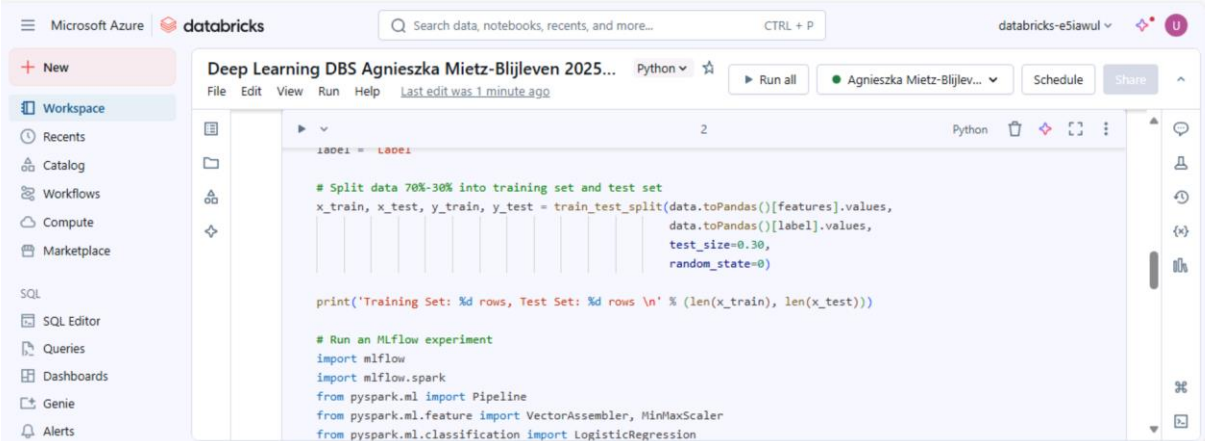
```
python
from pyspark.sql.types import *
from pyspark.sql.functions import *
from sklearn.model_selection import train_test_split

# Load the data, removing any incomplete rows
df = spark.read.format("csv").option("header", "true").load("/deepml_lab/patient_data.csv").dropna()

# Encode categorical features with integer indexes
categorical_features = ["Gender", "Diet", "SpineDisease", "NeurologyDisease", "HeartDisease", "Cancer",
"OtherParameters"]
indexers = [StringIndexer(inputCol=feature, outputCol=feature + "Idx") for feature in categorical features]
```



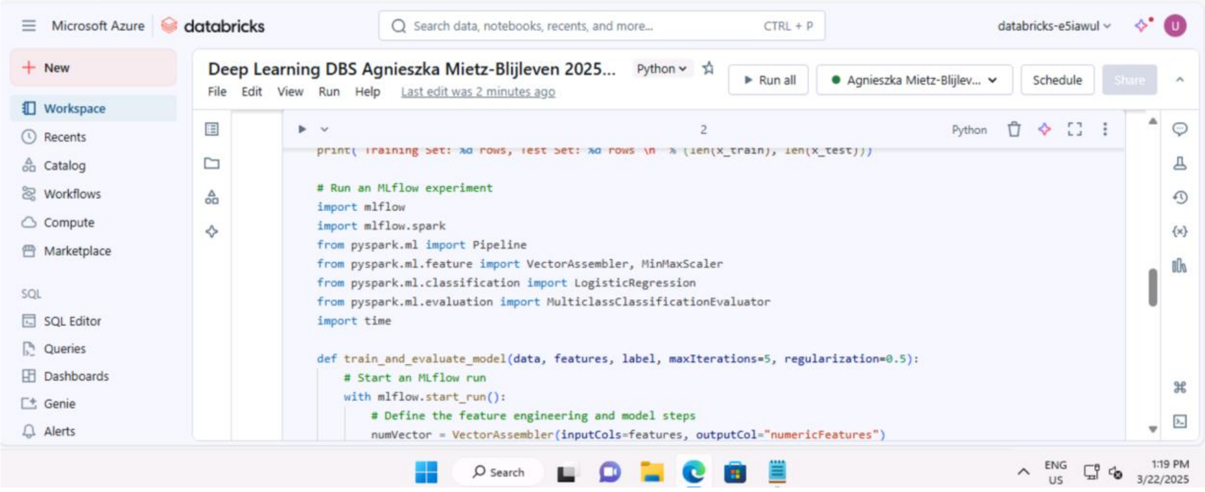
```
# Define numerical features
numerical_features = ["Age", "StrokeSeverity", "MigraineAuraFrequency", "RecoveryTime", "Weight", "Pressure",
"Glucose", "EEG", "MR", "ChargeTransfer", "GrapheneDiodes"]

# Apply feature engineering
pipeline = Pipeline(stages=indexers)
df = pipeline.fit(df).transform(df)

# Select and transform the relevant columns
data = df.select([col(feature + "Idx").astype("float") for feature in categorical_features] +
                 [col(feature).astype("float") for feature in numerical_features] +
                 [col("StrokeSeverity").astype("float").alias("Label")])

# Oversample the dataframe to triple its size (Deep learning techniques like LOTS of data)
for i in range(1, 3):
```



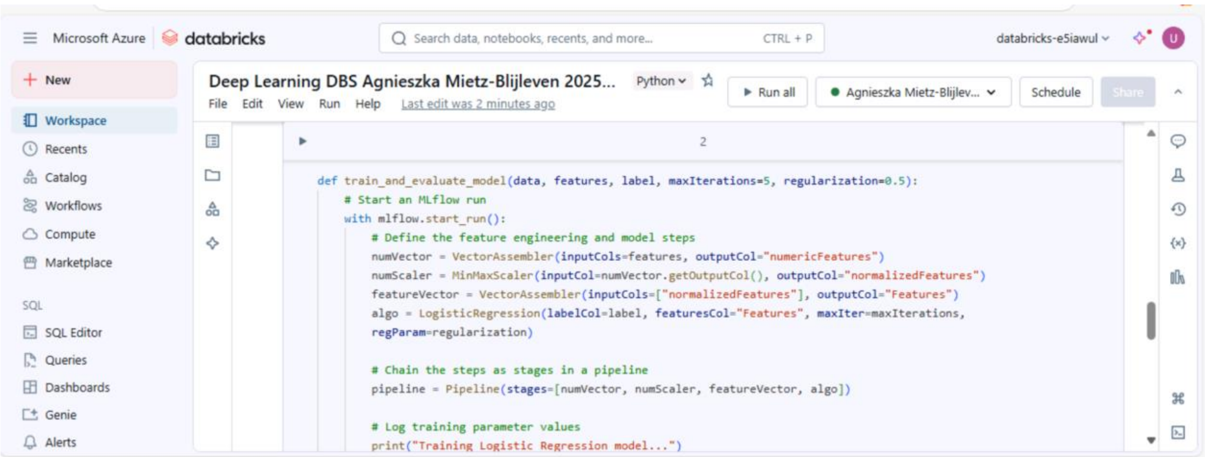
```
                    [col('StrokeSeverity').astype('float').alias('Label')])

# Oversample the dataframe to triple its size (Deep learning techniques like LOTS of data)
for i in range(1, 3):
    data = data.union(data)

# Split the data into training and testing datasets
features = [feature + "Idx" for feature in categorical_features] + numerical_features
label = 'Label'

# Split data 70%-30% into training set and test set
x_train, x_test, y_train, y_test = train_test_split(data.toPandas()[features].values,
                                                    data.toPandas()[label].values,
                                                    test_size=0.30,
                                                    random_state=0)
```



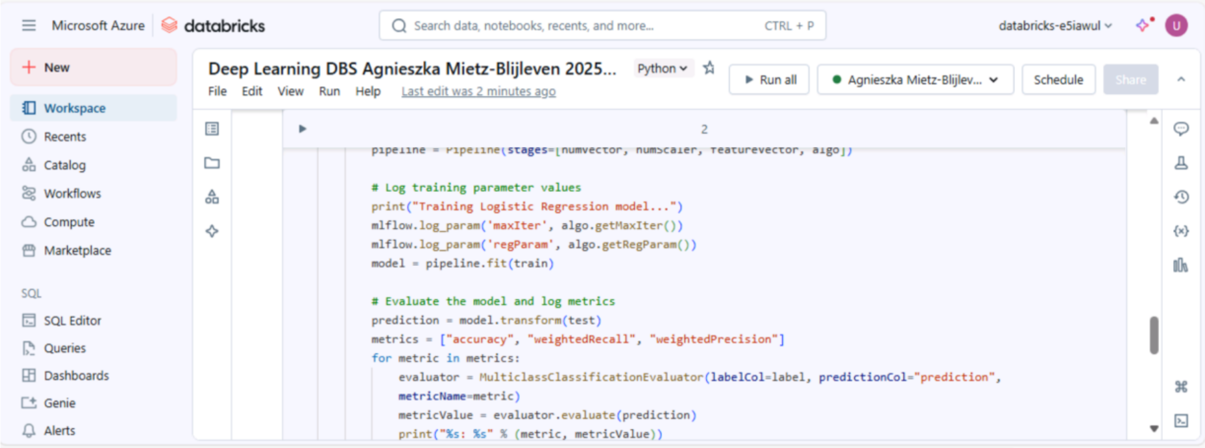
```
label = 'Label'

# Split data 70%-30% into training set and test set
x_train, x_test, y_train, y_test = train_test_split(data.toPandas()[features].values,
                                                    data.toPandas()[label].values,
                                                    test_size=0.30,
                                                    random_state=0)

print('Training Set: %d rows, Test Set: %d rows \n' % (len(x_train), len(x_test)))

# Run an MLflow experiment
import mlflow
import mlflow.spark
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler, MinMaxScaler
from pyspark.ml.classification import LogisticRegression
```



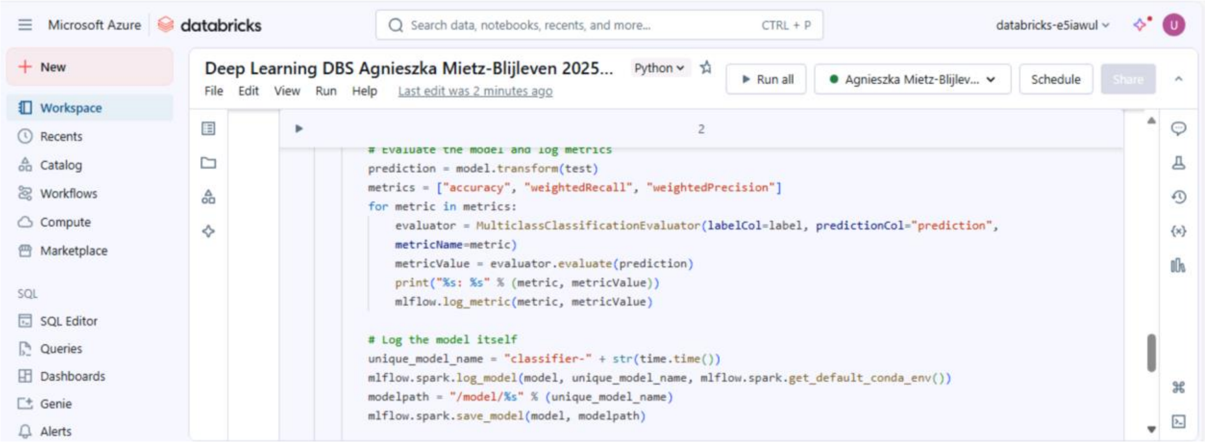
```
print('Training Set: %d rows, Test Set: %d rows \n' % (len(x_train), len(x_test)))

# Run an MLflow experiment
import mlflow
import mlflow.spark
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler, MinMaxScaler
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import time

def train_and_evaluate_model(data, features, label, maxIterations=5, regularization=0.5):
    # Start an MLflow run
    with mlflow.start_run():
        # Define the feature engineering and model steps
        numVector = VectorAssembler(inputCols=features, outputCol="numericFeatures")
```



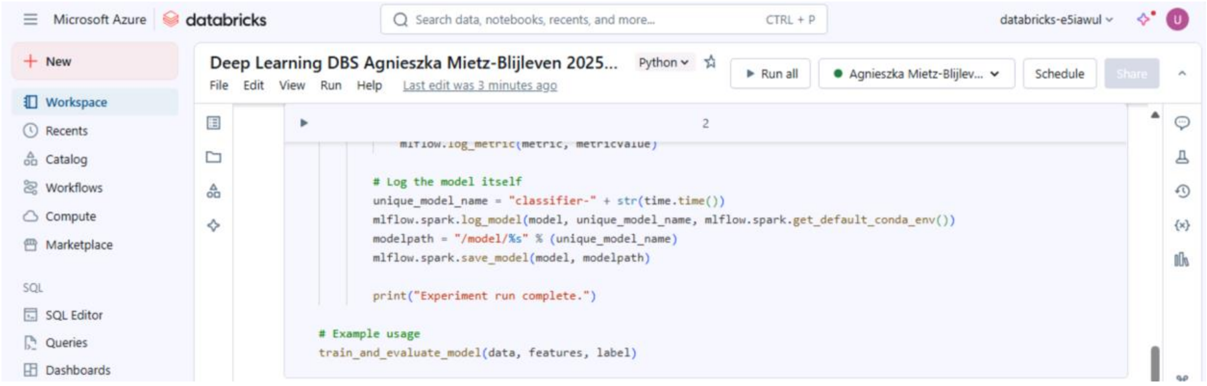
```python
def train_and_evaluate_model(data, features, label, maxIterations=5, regularization=0.5):
    # Start an MLflow run
    with mlflow.start_run():
        # Define the feature engineering and model steps
        numVector = VectorAssembler(inputCols=features, outputCol="numericFeatures")
        numScaler = MinMaxScaler(inputCol=numVector.getOutputCol(), outputCol="normalizedFeatures")
        featureVector = VectorAssembler(inputCols=["normalizedFeatures"], outputCol="Features")
        algo = LogisticRegression(labelCol=label, featuresCol="Features", maxIter=maxIterations,
        regParam=regularization)

        # Chain the steps as stages in a pipeline
        pipeline = Pipeline(stages=[numVector, numScaler, featureVector, algo])

        # Log training parameter values
        print("Training Logistic Regression model...")
```



```python
        pipeline = Pipeline(stages=[numVector, numScaler, featureVector, algo])

        # Log training parameter values
        print("Training Logistic Regression model...")
        mlflow.log_param('maxIter', algo.getMaxIter())
        mlflow.log_param('regParam', algo.getRegParam())
        model = pipeline.fit(train)

        # Evaluate the model and log metrics
        prediction = model.transform(test)
        metrics = ["accuracy", "weightedRecall", "weightedPrecision"]
        for metric in metrics:
            evaluator = MulticlassClassificationEvaluator(labelCol=label, predictionCol="prediction",
            metricName=metric)
            metricValue = evaluator.evaluate(prediction)
            print("%s: %s" % (metric, metricValue))
```

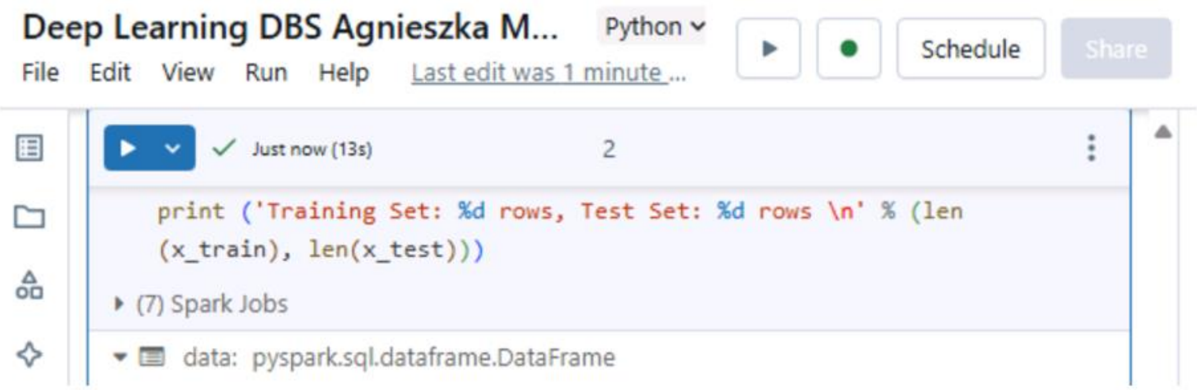


```python
        # Evaluate the model and log metrics
        prediction = model.transform(test)
        metrics = ["accuracy", "weightedRecall", "weightedPrecision"]
        for metric in metrics:
            evaluator = MulticlassClassificationEvaluator(labelCol=label, predictionCol="prediction",
            metricName=metric)
            metricValue = evaluator.evaluate(prediction)
            print("%s: %s" % (metric, metricValue))
            mlflow.log_metric(metric, metricValue)

        # Log the model itself
        unique_model_name = "classifier-" + str(time.time())
        mlflow.spark.log_model(model, unique_model_name, mlflow.spark.get_default_conda_env())
        modelpath = "/model/%s" % (unique_model_name)
        mlflow.spark.save_model(model, modelpath)
```



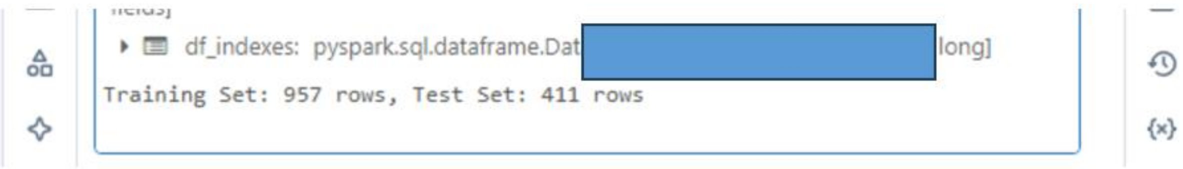
```
            mlflow.log_metric(metric, metricvalue)

        # Log the model itself
        unique_model_name = "classifier-" + str(time.time())
        mlflow.spark.log_model(model, unique_model_name, mlflow.spark.get_default_conda_env())
        modelpath = "/model/%s" % (unique_model_name)
        mlflow.spark.save_model(model, modelpath)

        print("Experiment run complete.")

# Example usage
train_and_evaluate_model(data, features, label)
```

And execute:


Deep Learning DBS Agnieszka M...
Python
File
Edit
View
Run
Help
Last edit was 1 minute ...
Schedule
Share
Just now (13s)
2

```
print ('Training Set: %d rows, Test Set: %d rows \n' % (len
(x_train), len(x_test)))
```

(7) Spark Jobs
data: pyspark.sql.dataframe.DataFrame

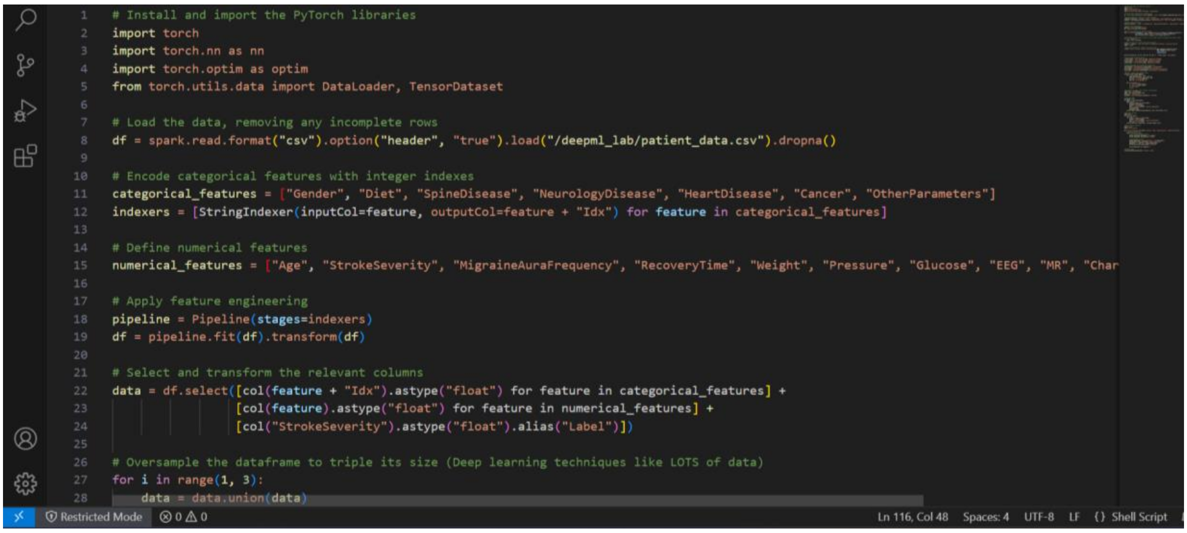


```
Training Set: 957 rows, Test Set: 411 rows
```

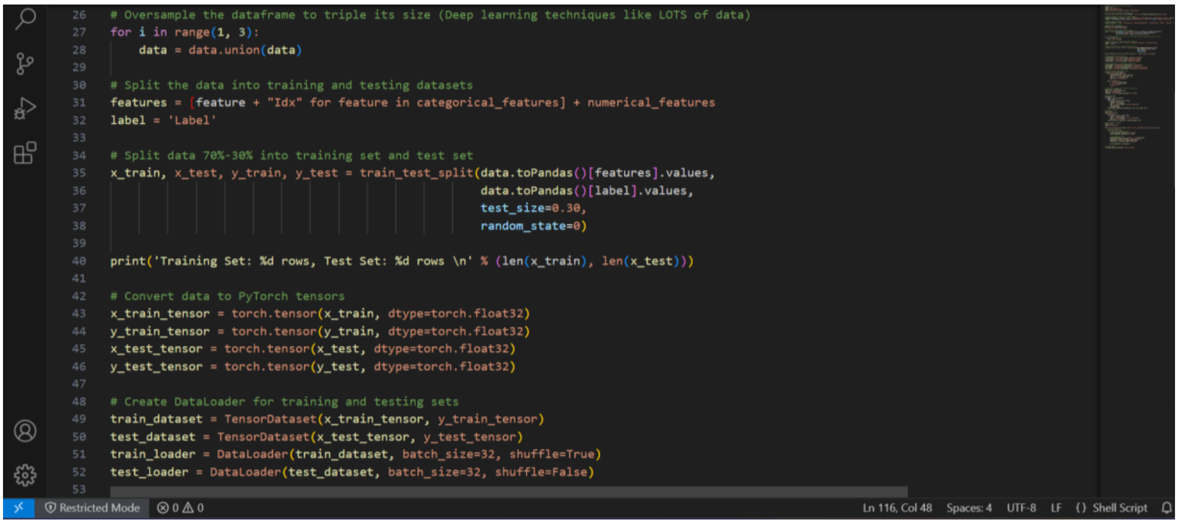
Install and import the PyTorch libraries

PyTorch is a framework for creating machine learning models, including deep neural networks (DNNs). Since I plan to use PyTorch to create my penguin classifier, I'll need to import the PyTorch libraries I intend to use. PyTorch is already installed on Azure databricks clusters with an ML Databricks runtime (the specific installation of PyTorch depends on whether the cluster has graphics processing units (GPUs) that can be used for high-performance processing via cuda).

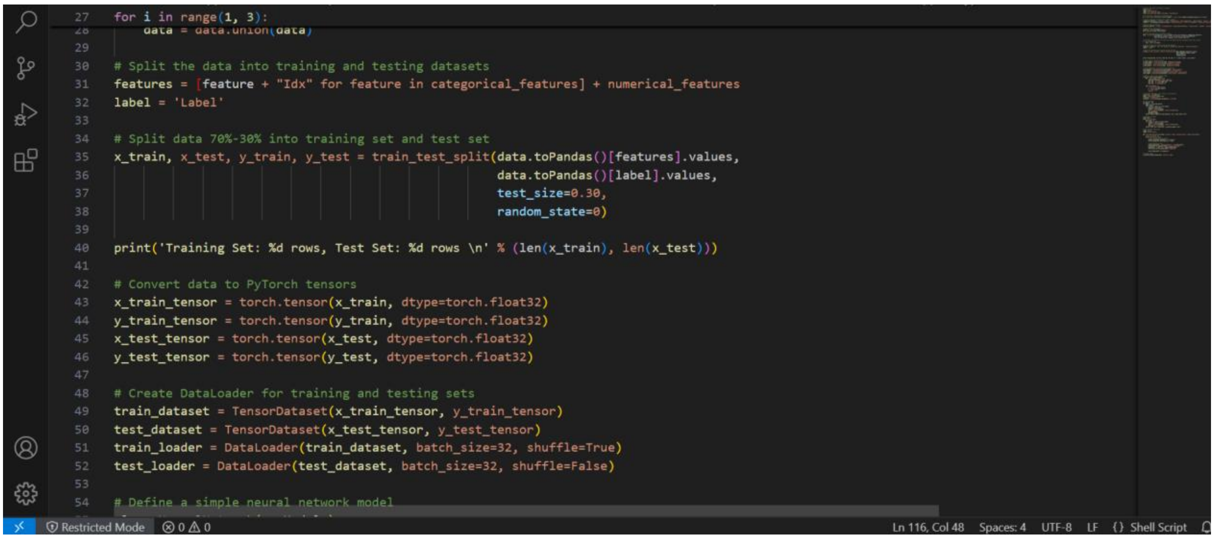
The code to include PyTorch for creating machine learning models, including deep neural networks (DNNs):



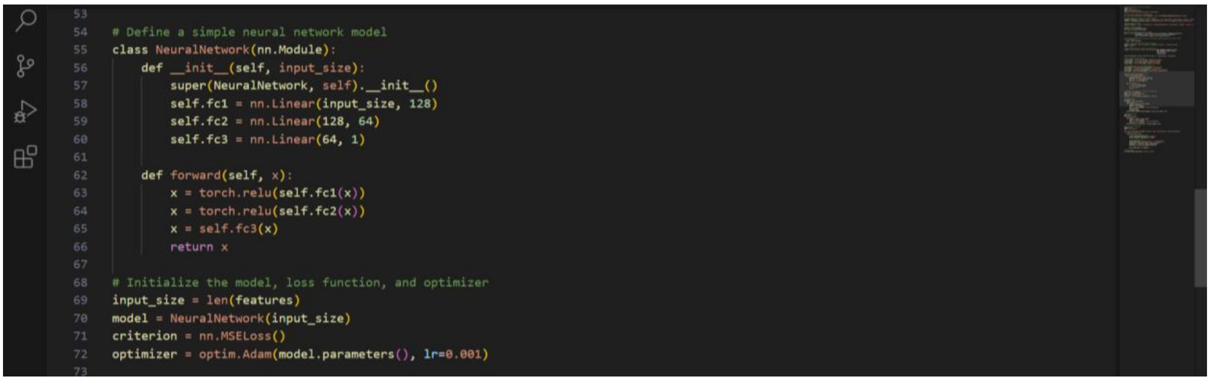
```python
# Install and import the PyTorch libraries
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

# Load the data, removing any incomplete rows
df = spark.read.format("csv").option("header", "true").load("/deepml_lab/patient_data.csv").dropna()

# Encode categorical features with integer indexes
categorical_features = ["Gender", "Diet", "SpineDisease", "NeurologyDisease", "HeartDisease", "Cancer", "OtherParameters"]
indexers = [StringIndexer(inputCol=feature, outputCol=feature + "Idx") for feature in categorical_features]

# Define numerical features
numerical_features = ["Age", "StrokeSeverity", "MigraineAuraFrequency", "RecoveryTime", "Weight", "Pressure", "Glucose", "EEG", "MR", "Char

# Apply feature engineering
pipeline = Pipeline(stages=indexers)
df = pipeline.fit(df).transform(df)

# Select and transform the relevant columns
data = df.select([col(feature + "Idx").astype("float") for feature in categorical_features] +
                 [col(feature).astype("float") for feature in numerical_features] +
                 [col("StrokeSeverity").astype("float").alias("Label")])

# Oversample the dataframe to triple its size (Deep learning techniques like LOTS of data)
for i in range(1, 3):
    data = data.union(data)
```



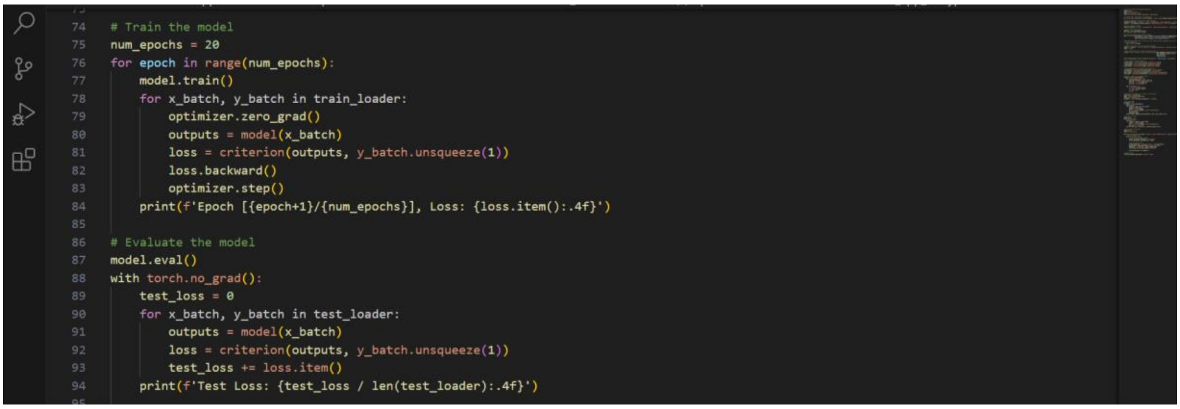
```python
# Oversample the dataframe to triple its size (Deep learning techniques like LOTS of data)
for i in range(1, 3):
    data = data.union(data)

# Split the data into training and testing datasets
features = [feature + "Idx" for feature in categorical_features] + numerical_features
label = 'Label'

# Split data 70%-30% into training set and test set
x_train, x_test, y_train, y_test = train_test_split(data.toPandas()[features].values,
                                                    data.toPandas()[label].values,
                                                    test_size=0.30,
                                                    random_state=0)

print('Training Set: %d rows, Test Set: %d rows \n' % (len(x_train), len(x_test)))

# Convert data to PyTorch tensors
x_train_tensor = torch.tensor(x_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
x_test_tensor = torch.tensor(x_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)

# Create DataLoader for training and testing sets
train_dataset = TensorDataset(x_train_tensor, y_train_tensor)
test_dataset = TensorDataset(x_test_tensor, y_test_tensor)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```



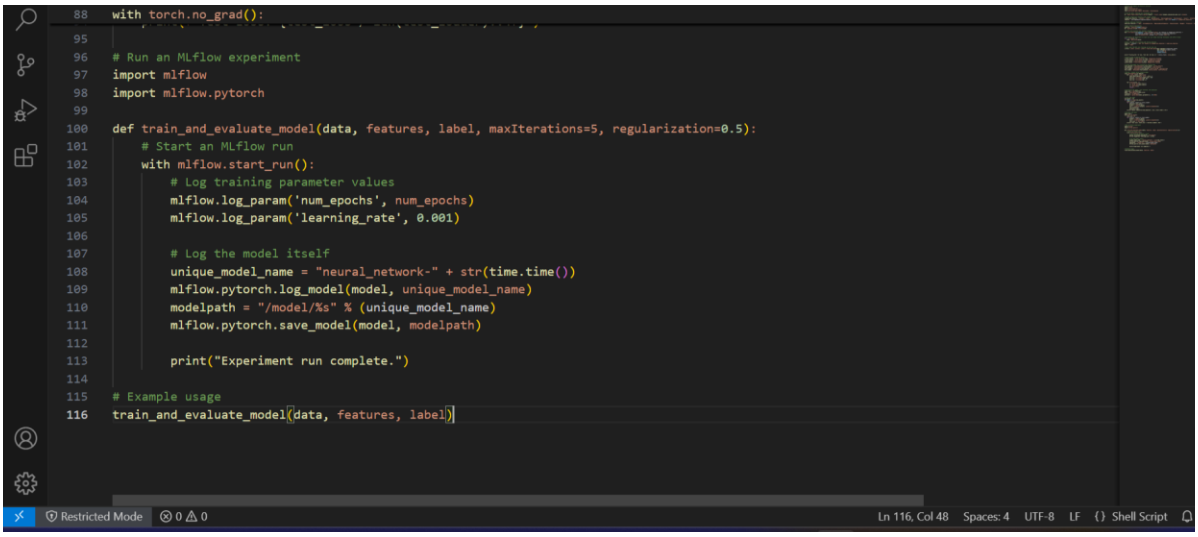
```python
for i in range(1, 3):
    data = data.union(data)

# Split the data into training and testing datasets
features = [feature + "Idx" for feature in categorical_features] + numerical_features
label = 'Label'

# Split data 70%-30% into training set and test set
x_train, x_test, y_train, y_test = train_test_split(data.toPandas()[features].values,
                                                    data.toPandas()[label].values,
                                                    test_size=0.30,
                                                    random_state=0)

print('Training Set: %d rows, Test Set: %d rows \n' % (len(x_train), len(x_test)))

# Convert data to PyTorch tensors
x_train_tensor = torch.tensor(x_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32)
x_test_tensor = torch.tensor(x_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32)

# Create DataLoader for training and testing sets
train_dataset = TensorDataset(x_train_tensor, y_train_tensor)
test_dataset = TensorDataset(x_test_tensor, y_test_tensor)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

# Define a simple neural network model
```



```python
# Define a simple neural network model
class NeuralNetwork(nn.Module):
    def __init__(self, input_size):
        super(NeuralNetwork, self).__init__()
        self.fc1 = nn.Linear(input_size, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# Initialize the model, loss function, and optimizer
input_size = len(features)
model = NeuralNetwork(input_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```



```python
# Train the model
num_epochs = 20
for epoch in range(num_epochs):
    model.train()
    for x_batch, y_batch in train_loader:
        optimizer.zero_grad()
        outputs = model(x_batch)
        loss = criterion(outputs, y_batch.unsqueeze(1))
        loss.backward()
        optimizer.step()
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')

# Evaluate the model
model.eval()
with torch.no_grad():
    test_loss = 0
    for x_batch, y_batch in test_loader:
        outputs = model(x_batch)
        loss = criterion(outputs, y_batch.unsqueeze(1))
        test_loss += loss.item()
    print(f'Test Loss: {test_loss / len(test_loader):.4f}')
```



```
with torch.no_grad():

# Run an MLflow experiment
import mlflow
import mlflow.pytorch

def train_and_evaluate_model(data, features, label, maxIterations=5, regularization=0.5):
    # Start an MLflow run
    with mlflow.start_run():
        # Log training parameter values
        mlflow.log_param('num_epochs', num_epochs)
        mlflow.log_param('learning_rate', 0.001)

        # Log the model itself
        unique_model_name = "neural_network-" + str(time.time())
        mlflow.pytorch.log_model(model, unique_model_name)
        modelpath = "/model/%s" % (unique_model_name)
        mlflow.pytorch.save_model(model, modelpath)

        print("Experiment run complete.")

# Example usage
train_and_evaluate_model(data, features, label)
```

In this code:

- PyTorch: was used to create and train a neural network model.
- Categorical features: Gender, Diet, SpineDisease, NeurologyDisease, HeartDisease, Cancer, OtherParameters.
- Numerical features: Age, StrokeSeverity, MigraineAuraFrequency, RecoveryTime, Weight, Pressure, Glucose, EEG, MR, ChargeTransfer, GrapheneDiodes.
- Label: StrokeSeverity.

What is needed - to adjust the column names and data types based on the specific dataset and requirements.

Create data loaders

PyTorch makes use of *data loaders* to load training and validation data in batches. We've already loaded the data into numpy arrays, but we need to wrap those in PyTorch datasets (in which the data is converted to PyTorch *tensor* objects) and create loaders to read batches from those datasets.

```
    # Install and import the PyTorch libraries
    import torch
    import torch.nn as nn
    import torch.optim as optim
    from torch.utils.data import DataLoader, TensorDataset
    import mlflow
    import mlflow.pytorch

    # Load the data, removing any incomplete rows
    df                    =                    spark.read.format("csv").option("header",
"true").load("/deepml_lab/patient_data.csv").dropna()

    # Encode categorical features with integer indexes
    categorical_features = ["Gender", "Diet", "SpineDisease", "NeurologyDisease",
"HeartDisease", "Cancer", "OtherParameters"]
    indexers = [StringIndexer(inputCol=feature, outputCol=feature + "Idx") for feature in
categorical_features]

    # Define numerical features
    numerical_features   =   ["Age",   "StrokeSeverity",   "MigraineAuraFrequency",
"RecoveryTime", "Weight", "Pressure", "Glucose", "EEG", "MR", "ChargeTransfer",
"GrapheneDiodesMeasurement","…",  ]

    # Apply feature engineering
    pipeline = Pipeline(stages=indexers)
    df = pipeline.fit(df).transform(df)

    # Select and transform the relevant columns
    data = df.select([col(feature + "Idx").astype("float") for feature in categorical_features]
+
            [col(feature).astype("float") for feature in numerical_features] +
            [col("StrokeSeverity").astype("float").alias("Label")])

    # Oversample the dataframe to triple its size (Deep learning techniques like LOTS of
data)
    for i in range(1, 3):
      data = data.union(data)

    # Split the data into training and testing datasets
    features = [feature + "Idx" for feature in categorical_features] + numerical_features
    label = 'Label'

    # Split data 70%-30% into training set and test set
    x_train, x_test, y_train, y_test = train_test_split(data.toPandas()[features].values,
                                                        data.toPandas()[label].values,
                                                        test_size=0.30,
                                                        random_state=0)
```

```
print('Training Set: %d rows, Test Set: %d rows \n' % (len(x_train), len(x_test)))

# Create a dataset and loader for the training data and labels
train_x = torch.Tensor(x_train).float()
train_y = torch.Tensor(y_train).long()
train_ds = TensorDataset(train_x, train_y)
train_loader = DataLoader(train_ds, batch_size=20, shuffle=False, num_workers=1)

# Create a dataset and loader for the test data and labels
test_x = torch.Tensor(x_test).float()
test_y = torch.Tensor(y_test).long()
test_ds = TensorDataset(test_x, test_y)
test_loader = DataLoader(test_ds, batch_size=20, shuffle=False, num_workers=1)

print('Ready to load data')

# Define a simple neural network model
class NeuralNetwork(nn.Module):
    def __init__(self, input_size):
        super(NeuralNetwork, self).__init__()
        self.fc1 = nn.Linear(input_size, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# Initialize the model, loss function, and optimizer
input_size = len(features)
model = NeuralNetwork(input_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Train the model
num_epochs = 20
for epoch in range(num_epochs):
    model.train()
    for x_batch, y_batch in train_loader:
        optimizer.zero_grad()
        outputs = model(x_batch)
        loss = criterion(outputs, y_batch.unsqueeze(1))
        loss.backward()
        optimizer.step()
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
```

```
# Evaluate the model
model.eval()
with torch.no_grad():
    test_loss = 0
    for x_batch, y_batch in test_loader:
        outputs = model(x_batch)
        loss = criterion(outputs, y_batch.unsqueeze(1))
        test_loss += loss.item()
    print(f'Test Loss: {test_loss / len(test_loader):.4f}')

# Run an MLflow experiment
def train_and_evaluate_model(data, features, label, maxIterations=5, regularization=0.5):
    # Start an MLflow run
    with mlflow.start_run():
        # Log training parameter values
        mlflow.log_param('num_epochs', num_epochs)
        mlflow.log_param('learning_rate', 0.001)

        # Log the model itself
        unique_model_name = "neural_network-" + str(time.time())
        mlflow.pytorch.log_model(model, unique_model_name)
        modelpath = "/model/%s" % (unique_model_name)
        mlflow.pytorch.save_model(model, modelpath)

        print("Experiment run complete.")

# Example usage
train_and_evaluate_model(data, features, label)
```

Design of a code for training model -Notepad - Agnieszka Mietz-Blijleven source training code.

Model can also be trained based on scans of brain achieved from tomography or magnetic resonance. During training, an algorithm is presented with chest X-ray images labeled to indicate the presence or absence of a mass. The algorithm learns from these images and labels. Eventually, the algorithm learns to analyze a chest X-ray and determine whether it contains a mass. This algorithm can be referred to by various names, such as deep learning algorithm, model, neural network, or convolutional neural network.

The algorithm produces output in the form of scores, which represent probabilities that the image contains a mass. For example, the probability that one image contains a mass might be 0.48, while another might be 0.51. Initially, these probability scores do not match the desired labels.

Let's say the desired label for a mass is 1, and for normal is 0. You can see that 0.48 is far from 1, and 0.51 is far from 0. We measure this error by computing a loss function, which quantifies the difference between the output probability and the desired label. There is a need to explore how this loss is computed[1].

As the algorithm continues to learn, it is presented with new sets of images and desired labels, gradually producing scores that are closer to the desired labels. Notice how the output probability is getting closer to 1 for one image and closer to 0 for another, with the help of Azure Databricks for migraine with aura and stroke.

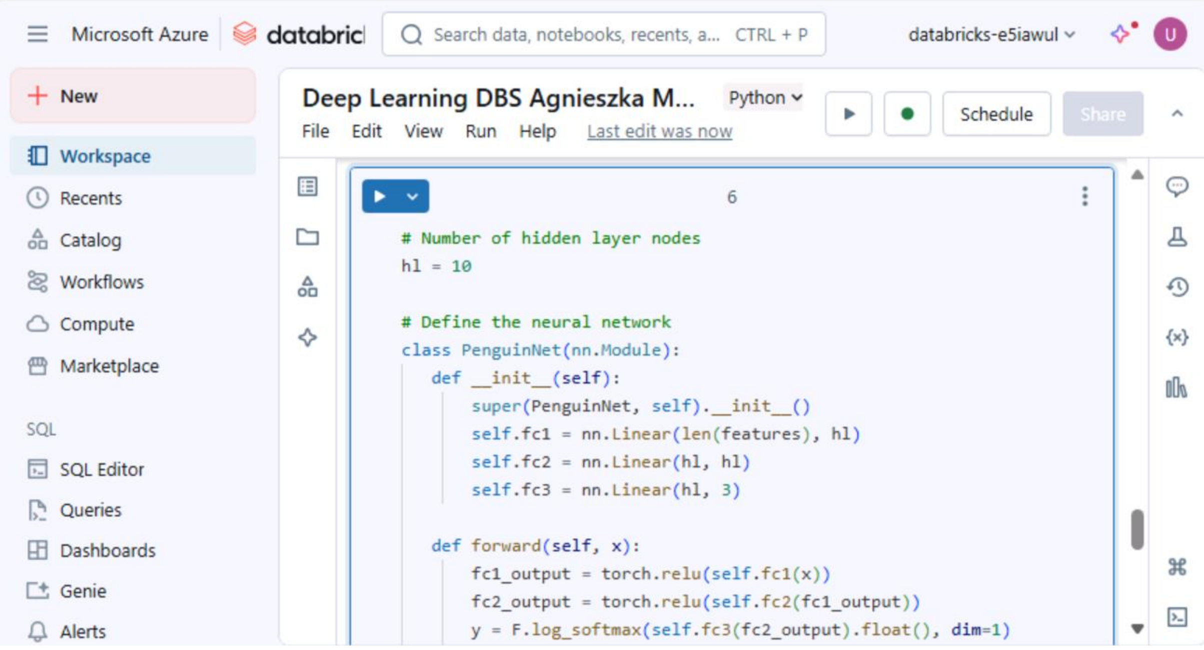
Define a neural network

Now I am ready to define our neural network. In this case, we'll create a network that consists of 3 fully-connected layers:

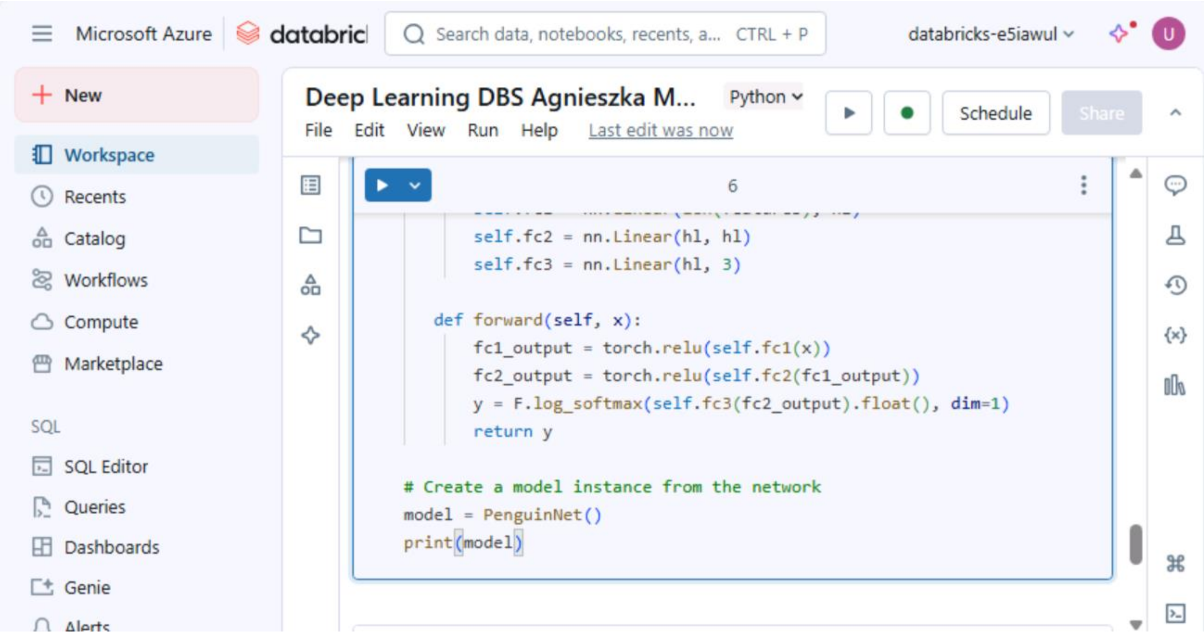
- An input layer that receives an input value for each feature (in this case, the island index and four penguin measurements) and generated 10 outputs.
- A hidden layer that receives ten inputs from the input layer and sends ten outputs to the next layer.
- An output layer that generates a vector of probabilities for each of the three possible penguin species.

[1] Error and uncertainty are explained in the secondary book that I authored.

As I train the network by passing data through it, the **forward** function will apply *RELU* activation functions to the first two layers (to constrain the results to positive numbers) and return a final output layer that uses a *log_softmax* function to return a value that represents a probability score for each of the three possible classes.



```
# Number of hidden layer nodes
hl = 10

# Define the neural network
class PenguinNet(nn.Module):
    def __init__(self):
        super(PenguinNet, self).__init__()
        self.fc1 = nn.Linear(len(features), hl)
        self.fc2 = nn.Linear(hl, hl)
        self.fc3 = nn.Linear(hl, 3)

    def forward(self, x):
        fc1_output = torch.relu(self.fc1(x))
        fc2_output = torch.relu(self.fc2(fc1_output))
        y = F.log_softmax(self.fc3(fc2_output).float(), dim=1)
```

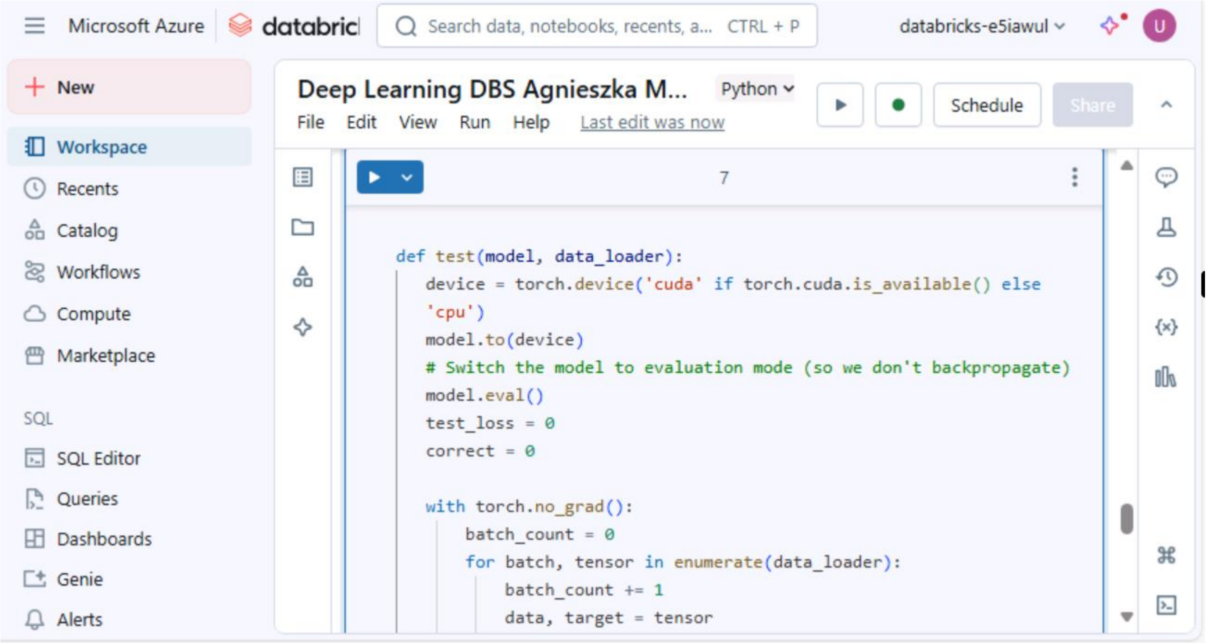


```
        self.fc2 = nn.Linear(hl, hl)
        self.fc3 = nn.Linear(hl, 3)

    def forward(self, x):
        fc1_output = torch.relu(self.fc1(x))
        fc2_output = torch.relu(self.fc2(fc1_output))
        y = F.log_softmax(self.fc3(fc2_output).float(), dim=1)
        return y

# Create a model instance from the network
model = PenguinNet()
print(model)
```

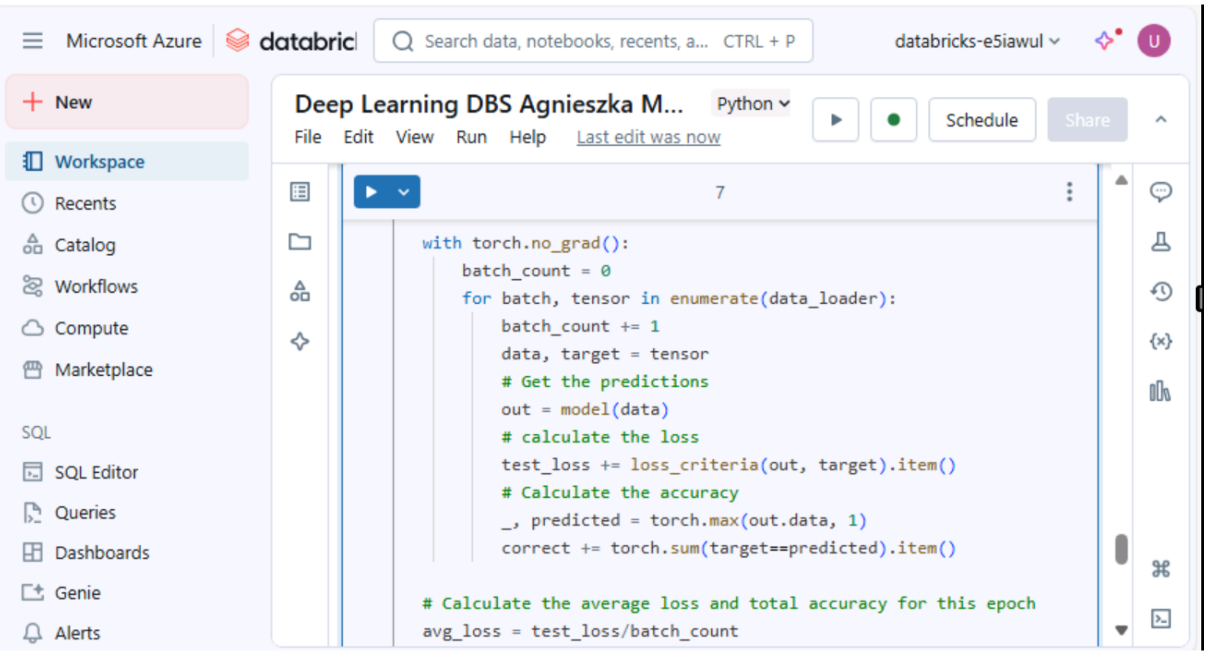
Create functions to train and test a neural network model

Training the model involves forwarding training values through the network, calculating loss with a loss function, backpropagating adjustments using an optimizer, and validating with test data.

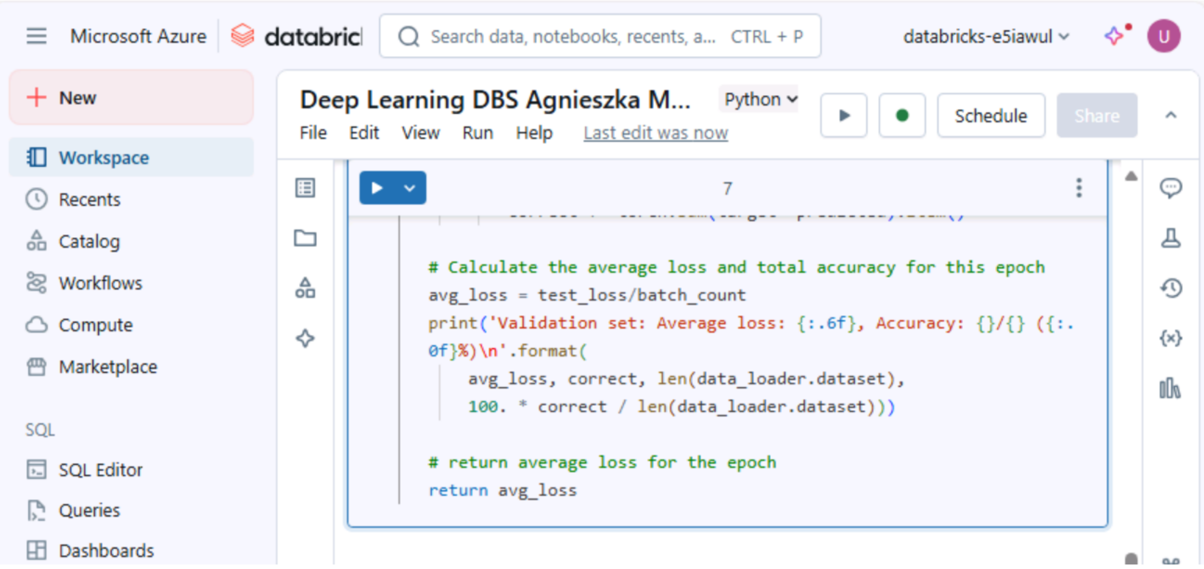
To do this, I used the following code to create a function to train and optimize the model, and function to test the model.



```
def test(model, data_loader):
    device = torch.device('cuda' if torch.cuda.is_available() else
    'cpu')
    model.to(device)
    # Switch the model to evaluation mode (so we don't backpropagate)
    model.eval()
    test_loss = 0
    correct = 0

    with torch.no_grad():
        batch_count = 0
        for batch, tensor in enumerate(data_loader):
            batch_count += 1
            data, target = tensor
```



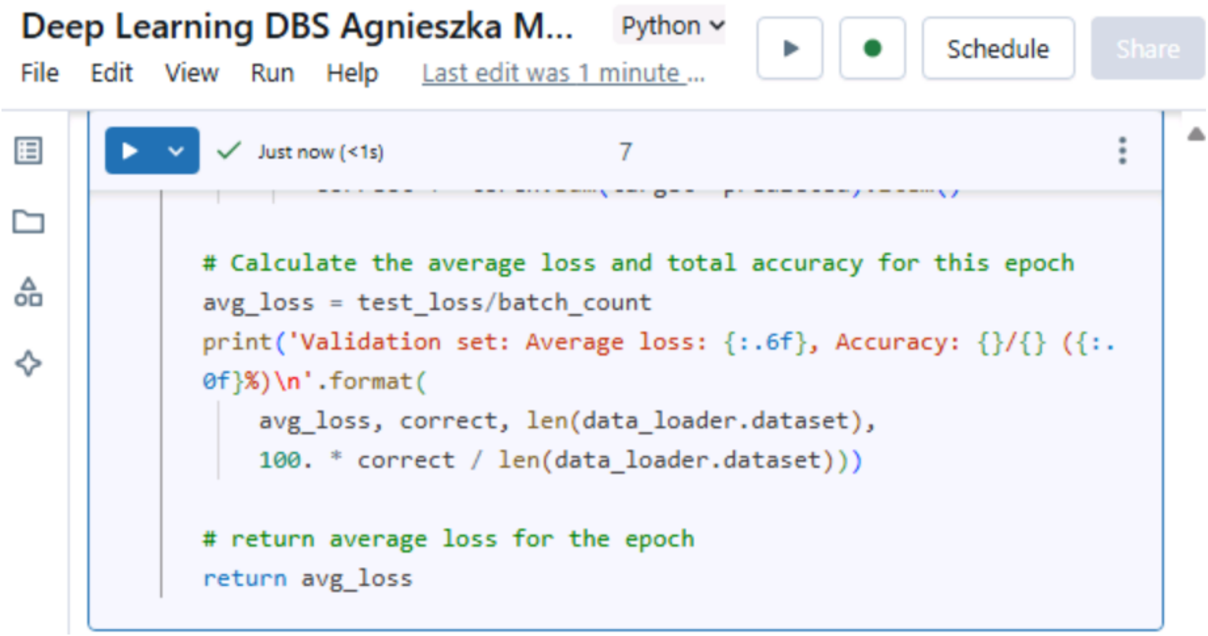
```
    with torch.no_grad():
        batch_count = 0
        for batch, tensor in enumerate(data_loader):
            batch_count += 1
            data, target = tensor
            # Get the predictions
            out = model(data)
            # calculate the loss
            test_loss += loss_criteria(out, target).item()
            # Calculate the accuracy
            _, predicted = torch.max(out.data, 1)
            correct += torch.sum(target==predicted).item()

    # Calculate the average loss and total accuracy for this epoch
    avg_loss = test_loss/batch_count
```



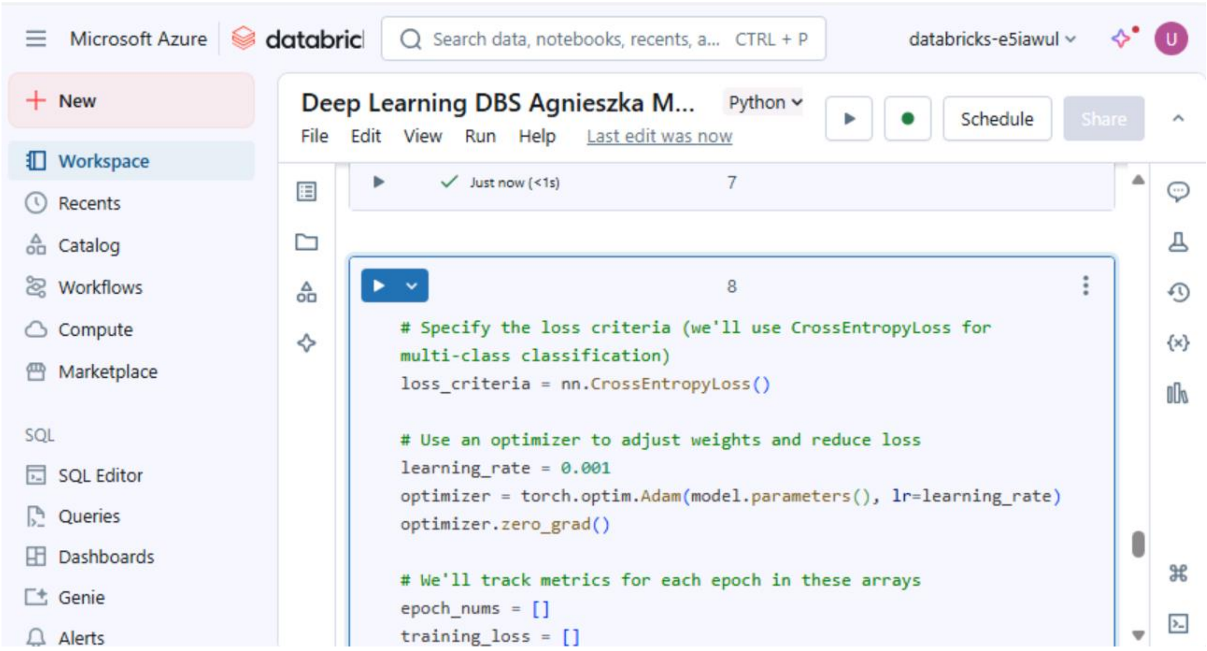
```
    # Calculate the average loss and total accuracy for this epoch
    avg_loss = test_loss/batch_count
    print('Validation set: Average loss: {:.6f}, Accuracy: {}/{} ({:.
0f}%)\n'.format(
        avg_loss, correct, len(data_loader.dataset),
        100. * correct / len(data_loader.dataset)))

    # return average loss for the epoch
    return avg_loss
```

Train a Deep Learning model for DBS

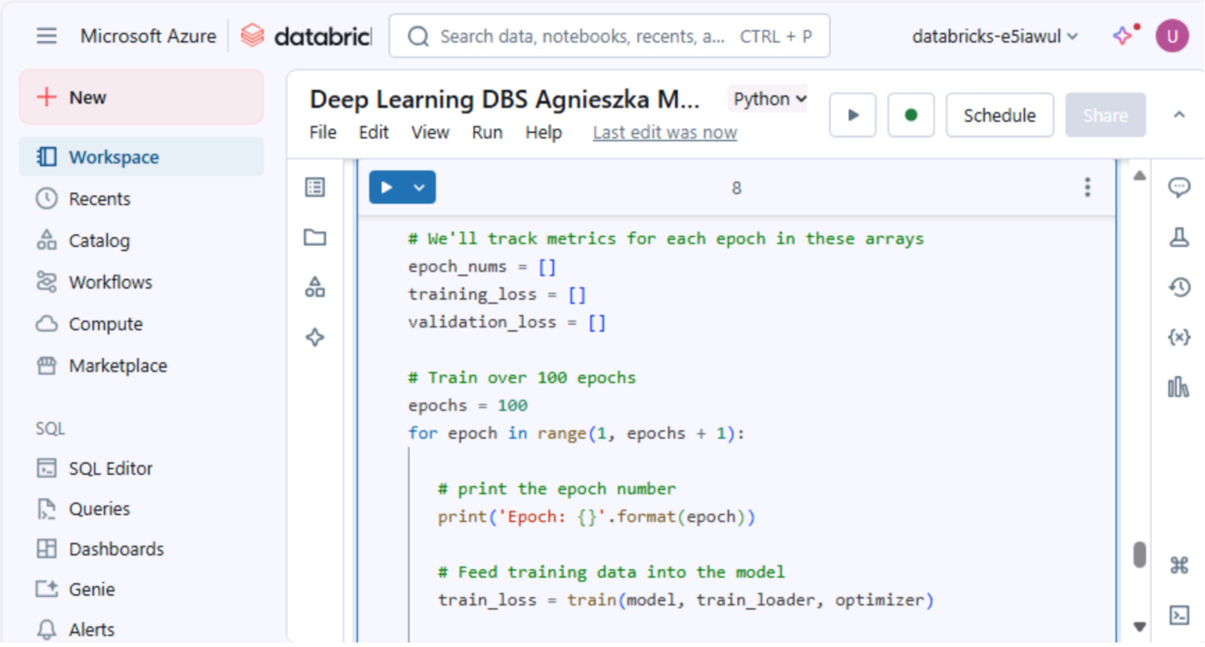
Now I can use the **train** and **test** functions to train a neural network model. I train neural networks iteratively over multiple *epochs*, logging the loss and accuracy statistics for each epoch.



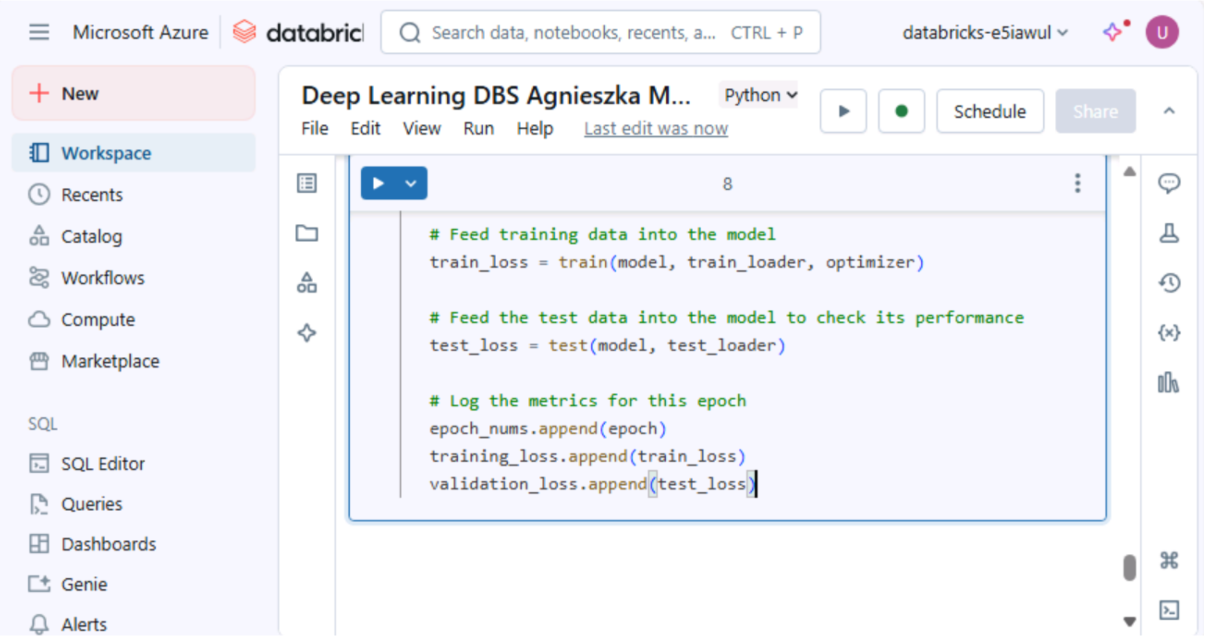
```
# Calculate the average loss and total accuracy for this epoch
avg_loss = test_loss/batch_count
print('Validation set: Average loss: {:.6f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
    avg_loss, correct, len(data_loader.dataset),
    100. * correct / len(data_loader.dataset)))

# return average loss for the epoch
return avg_loss
```



```
# Specify the loss criteria (we'll use CrossEntropyLoss for multi-class classification)
loss_criteria = nn.CrossEntropyLoss()

# Use an optimizer to adjust weights and reduce loss
learning_rate = 0.001
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
optimizer.zero_grad()

# We'll track metrics for each epoch in these arrays
epoch_nums = []
training_loss = []
```



```
# We'll track metrics for each epoch in these arrays
epoch_nums = []
training_loss = []
validation_loss = []

# Train over 100 epochs
epochs = 100
for epoch in range(1, epochs + 1):

    # print the epoch number
    print('Epoch: {}'.format(epoch))

    # Feed training data into the model
    train_loss = train(model, train_loader, optimizer)
```



```
    # Feed training data into the model
    train_loss = train(model, train_loader, optimizer)

    # Feed the test data into the model to check its performance
    test_loss = test(model, test_loader)

    # Log the metrics for this epoch
    epoch_nums.append(epoch)
    training_loss.append(train_loss)
    validation_loss.append(test_loss)
```

Figure 32 A set of screenshots illustrating the use of Deep Learning in the diagnosis of brain disease.

During the training process, several steps occur:

In each epoch, the complete set of training data is sent through the network. Each observation has five features, corresponding to five nodes in the input layer, thus forming a vector of five values for each observation. For efficiency, these feature vectors are grouped into batches, so a matrix containing multiple feature vectors is processed at once.

This matrix undergoes a function that performs a weighted sum using initialized weights and bias values. The result is then processed by the activation function for the input layer, constraining the values passed to the nodes in the subsequent layer.

The weighted sum and activation functions are repeated in each layer, operating on vectors and matrices rather than individual scalar values. Essentially, the forward pass involves a series of linear algebra functions. This is why GPUs are preferred, as they are optimized for such calculations.

In the network's final layer, output vectors represent calculated values for each class (e.g., classes 0, 1, and 2). A loss function evaluates these values against expected ones based on actual classes. For instance, if an observation for a Gentoo penguin (class 1) results in [0.3, 0.4, 0.3], it should be [0.0, 1.0, 0.0]. The variance between predicted and actual values ([0.3, 0.6, 0.3]) aggregates over each batch to calculate overall error (loss) for the epoch.

At each epoch's end, validation data is processed through the network, calculating its loss and accuracy (correct predictions based on the highest probability value in the output vector). Comparing the model's performance on validation data helps assess generalization capability and potential overfitting.

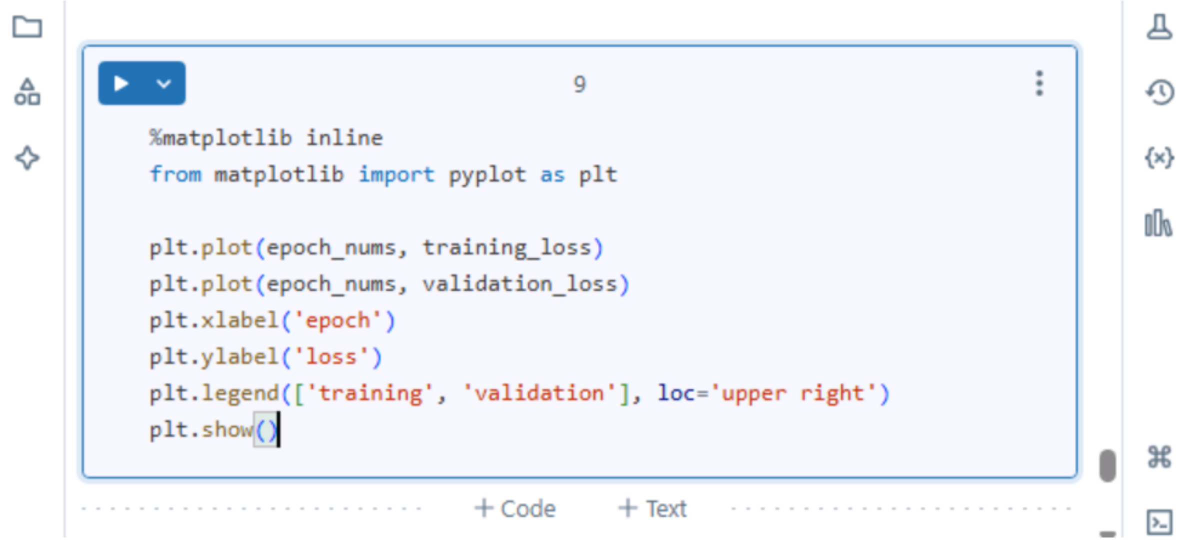
After processing all data, the optimizer receives the training data's loss function output but not the validation data's loss. Depending on the optimization algorithm, differential calculus calculates partial derivatives concerning each weight and bias value used in the network via the chain rule, determining whether to adjust these values to minimize loss.

The optimizer uses the learning rate to decide adjustment magnitude and applies backpropagation to update weights and biases in each layer. Subsequent epochs repeat the training, validation, and backpropagation process with revised weights and biases, aiming to lower loss levels.

This process continues for 100 epochs. Review training and validation loss

After training, examine the recorded loss metrics. Look for two things:

- The loss should decrease with each epoch, indicating the model is learning correctly.
- Training and validation loss should show similar trends, indicating the model is not overfitting.



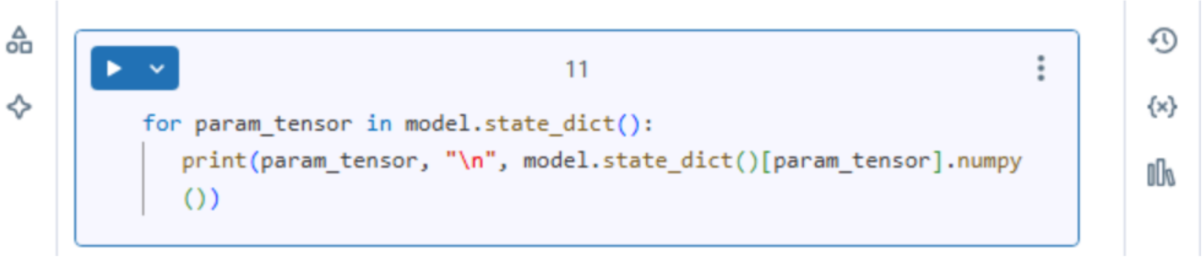
```
%matplotlib inline
from matplotlib import pyplot as plt

plt.plot(epoch_nums, training_loss)
plt.plot(epoch_nums, validation_loss)
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['training', 'validation'], loc='upper right')
plt.show()
```

Figure 33 Training and validation loss – DBS model

View the learned weights and biases

The trained model includes final weights and biases determined by the optimizer. Expected values for each layer are:

- Layer 1 (fc1): 10 x 5 weights, 10 bias values.
- Layer 2 (fc2): 10 x 10 weights, 10 bias values.
- Layer 3 (fc3): 3 x 10 weights, 3 bias values.

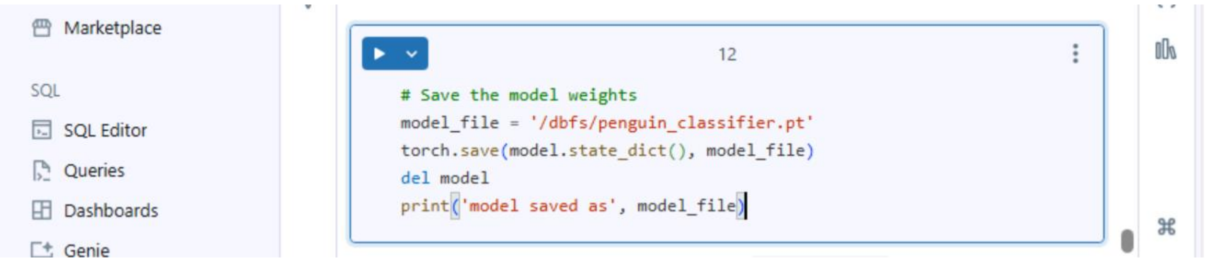


```
for param_tensor in model.state_dict():
    print(param_tensor, "\n", model.state_dict()[param_tensor].numpy
    ())
```

Figure 34 Labeling of data with bias and weights

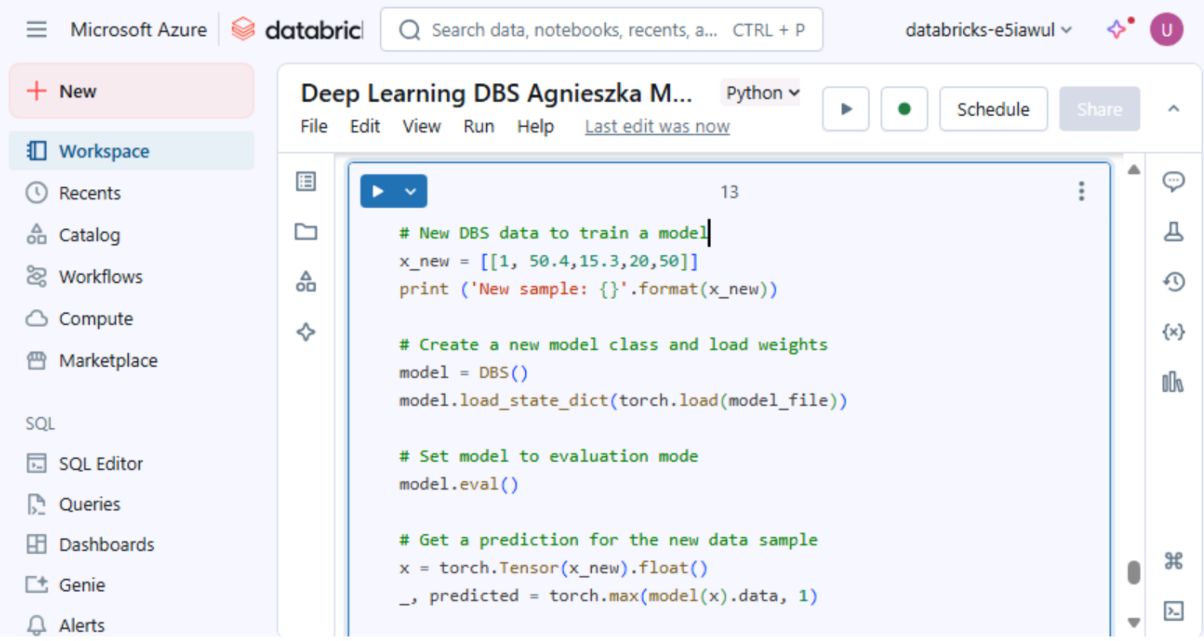
Saving and Utilizing the Trained Model

Upon training completion, the model's weights can be preserved for subsequent use.



```
# Save the model weights
model_file = '/dbfs/penguin_classifier.pt'
torch.save(model.state_dict(), model_file)
del model
print('model saved as', model_file)
```

I have used the code to load model weights and predict the probability of brain stroke or migraine with aura. Then, adjust to another DBS model for recovery of infants and seniors after brain stroke and migraine, based on observation, doctor examination, and treatment.



```
# New DBS data to train a model
x_new = [[1, 50.4,15.3,20,50]]
print ('New sample: {}'.format(x_new))

# Create a new model class and load weights
model = DBS()
model.load_state_dict(torch.load(model_file))

# Set model to evaluation mode
model.eval()

# Get a prediction for the new data sample
x = torch.Tensor(x_new).float()
_, predicted = torch.max(model(x).data, 1)
```

Chapter 5: Discussion, interpreting the results in the context of the research questions and objectives.

Results explanation with Azure AI Foundry

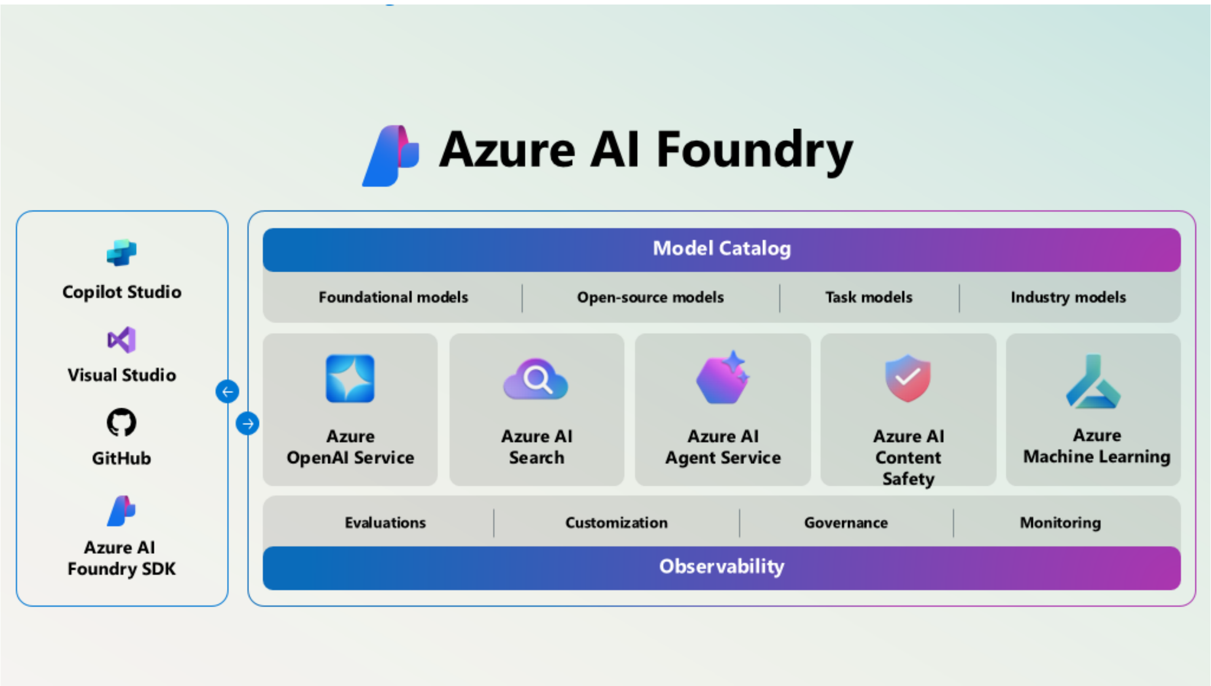


Figure 35 Microsoft Azure AI Foundry Platform

In the realm of artificial intelligence (AI) and machine learning (ML), the application of advanced technologies to medical research and treatment has shown promising results. This thesis explores the utilization of many self-deployed but also commercially available solutions, like Azure AI Foundry in the context of deep brain stimulation (DBS) for diagnosing and treating brain diseases. The integration of AI/ML, large language models (LLM), generative AI, and cybersecurity within Azure AI Foundry provides a robust framework for enhancing the accuracy and efficiency of medical interventions.[2]

[2] In day-to-day research, the author of this PhD thesis utilizes not only commercially available large language models (LLM), artificial intelligence (AI), generative AI, and machine learning (ML) but also self-developed solutions. While this thesis focuses on the scientific aspects, the author does not omit non-scientific justifications witnessed in the healing of her own children (early-born twins) throughout years of their rehabilitation and recovery process. Observations of other early-born infants born between the 22nd and 37th week of pregnancy, along with extensive notes undertaken in mapping to seniors with a focus on recovery from brain strokes and migraines with aura (which can lead to brain strokes), are included. The hope is that computer

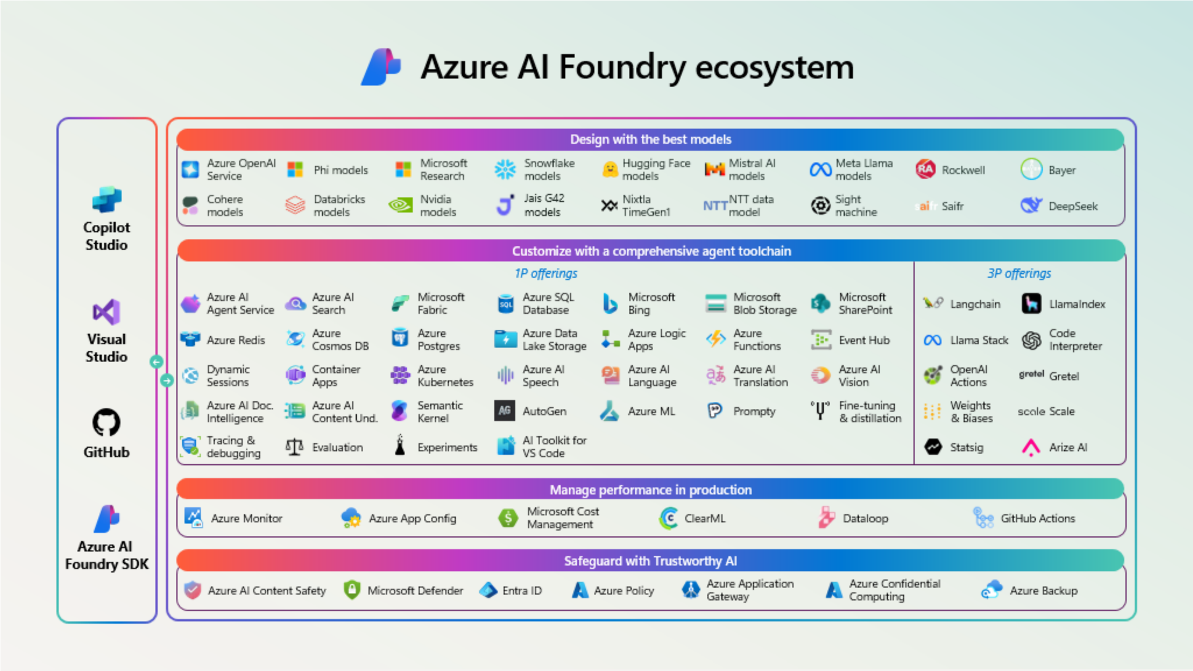


Figure 36 Microsoft Azure AI Foundry ecosystem

Graphene Diodes and Charge Transfer Mechanisms

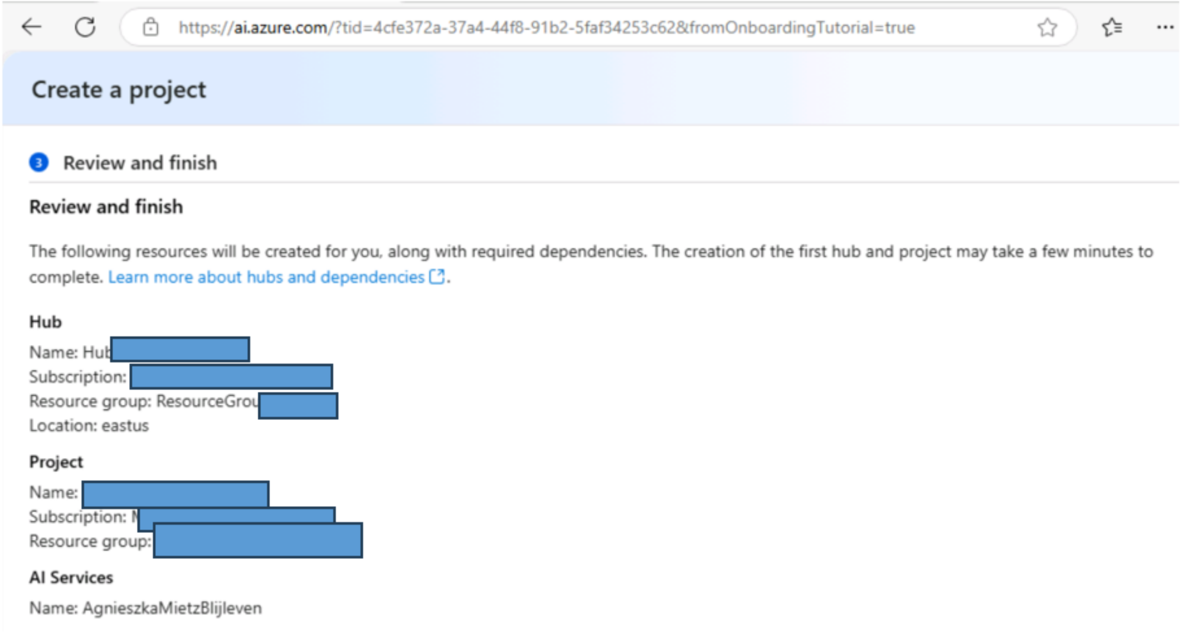
Graphene diodes, due to their exceptional electrical conductivity and biocompatibility, can be integrated with charge transfer mechanisms to stimulate neuronal activity. This approach can be particularly beneficial for treating neurological conditions such as migraines with aura and brain strokes. The treatment involves the application of graphene diodes to specific regions of the brain, where they facilitate charge transfer to modulate neuronal activity. Graphene diodes can be provided to the brain as stimulators and analysers through the nose without the need for opening the skull. In more serious diseases, surgeons shall decide on ways of implementing graphene diodes in layers of the brain in collaboration with neurologists.[3]

science, particularly AI, will ultimately work in concert with medicine towards the common goal of aiding individuals with neurological disorders and facilitating quicker recovery.

[3] In herein PhD Thesis an Author describes just selected options from many that were used in researches and are a subject of patent proceedings, thus introduced in limited scope.

Mechanism with LLM, Generative AI, AI/ML, and MASS Model with R and Julia for Machine Learning

The integration of generative AI applications developed in Azure AI Foundry can significantly enhance the efficacy of DBS treatments. This section provides specific examples of how these applications can be integrated into my PhD thesis results.



https://ai.azure.com/?tid=4cfe372a-37a4-44f8-91b2-5faf34253c62&fromOnboardingTutorial=true

Create a project

3 Review and finish

Review and finish

The following resources will be created for you, along with required dependencies. The creation of the first hub and project may take a few minutes to complete. Learn more about hubs and dependencies.

Hub
Name: Hu[redacted]
Subscription: [redacted]
Resource group: ResourceGro[redacted]
Location: eastus

Project
Name: [redacted]
Subscription: [redacted]
Resource group: [redacted]

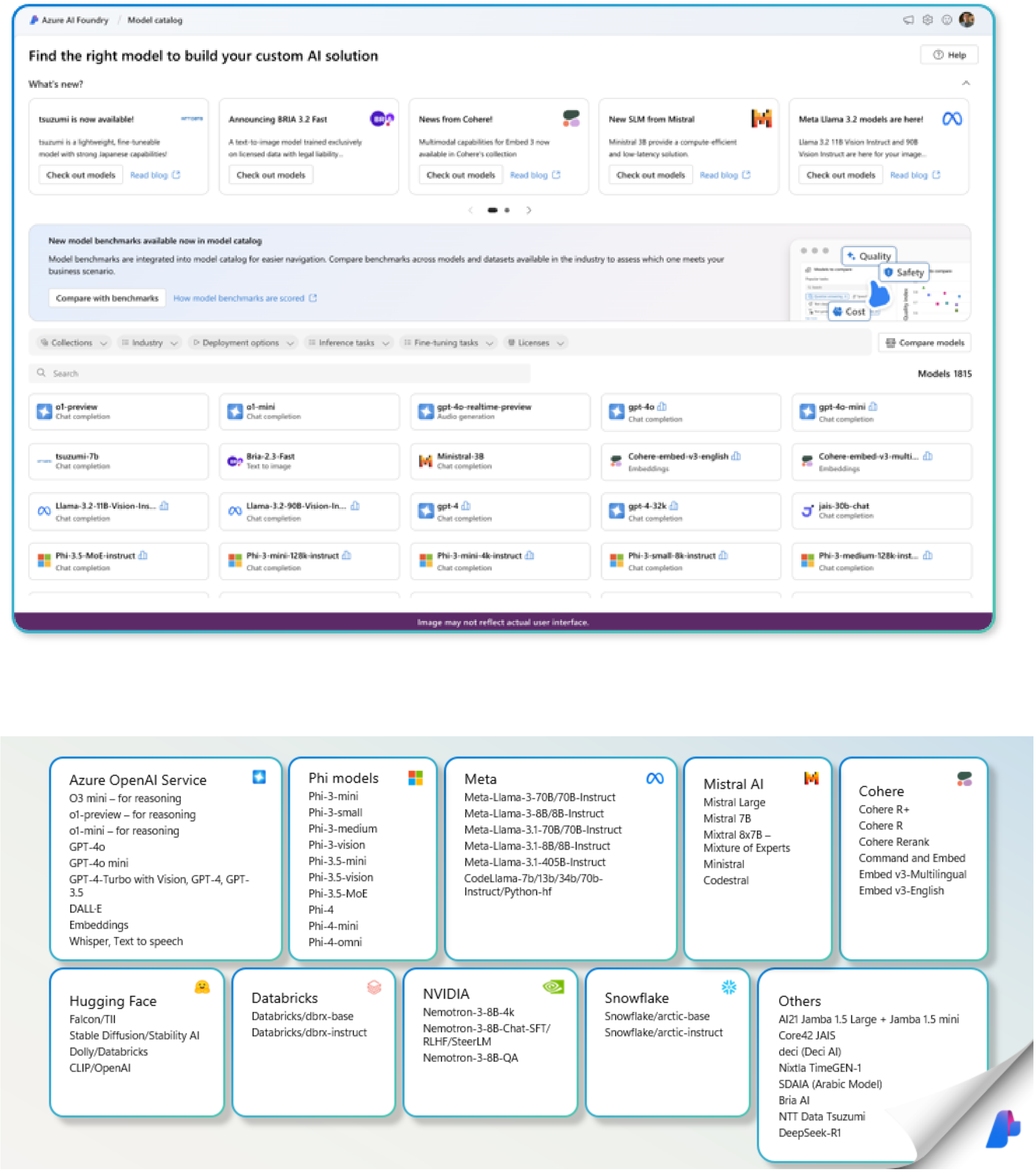
AI Services
Name: AgnieszkaMietzBlijleven

Figure 37 Generative AI application development in Azure AI Foundry – environment set up for results validation and interpretation

Specific Examples of Generative AI Applications in Azure AI Foundry for Deep Brain Stimulation

Example 1: Predictive Modelling with Charge Transfer

Azure AI Foundry SDK enables the development of AI models that can predict optimal charge transfer patterns required to stimulate specific neuronal pathways. By leveraging the SDK, researchers can create models with graphene diodes that analyze patient-specific data and predict the most effective charge transfer mechanisms for DBS treatments.



Azure AI Foundry / Model catalog

Find the right model to build your custom AI solution

Help

What's new?

tsuzumi is now available!
tsuzumi is a lightweight, fine-tuneable model with strong Japanese capabilities!
Check out models | Read blog

Announcing BRIA 3.2 Fast
A text-to-image model trained exclusively on licensed data with legal liability...
Check out models

News from Cohere!
Multimodal capabilities for Embed 3 now available in Cohere's collection
Check out models | Read blog

New SLM from Mistral
Ministral 3B provide a compute-efficient and low-latency solution.
Check out models | Read blog

Meta Llama 3.2 models are here!
Llama 3.2 11B Vision Instruct and 90B Vision Instruct are here for your image...
Check out models | Read blog

New model benchmarks available now in model catalog

Model benchmarks are integrated into model catalog for easier navigation. Compare benchmarks across models and datasets available in the industry to assess which one meets your business scenario.

Compare with benchmarks | How model benchmarks are scored

Collections | Industry | Deployment options | Inference tasks | Fine-tuning tasks | Licenses | Compare models

Search

Models 1815

o1-preview Chat completion	o1-mini Chat completion	gpt-4o-realtime-preview Audio generation	gpt-4o Chat completion	gpt-4o-mini Chat completion
tsuzumi-7b Chat completion	Bria-2.3-Fast Text to image	Ministral-3B Chat completion	Cohere-embed-v3-english Embeddings	Cohere-embed-v3-multi... Embeddings
Llama-3.2-11B-Vision-Ins... Chat completion	Llama-3.2-90B-Vision-In... Chat completion	gpt-4 Chat completion	gpt-4-32k Chat completion	jais-30b-chat Chat completion
Phi-3.5-MoE-instruct Chat completion	Phi-3-mini-128k-instruct Chat completion	Phi-3-mini-4k-instruct Chat completion	Phi-3-small-8k-instruct Chat completion	Phi-3-medium-128k-inst... Chat completion

Image may not reflect actual user interface.

Azure OpenAI Service
O3 mini – for reasoning
o1-preview – for reasoning
o1-mini – for reasoning
GPT-4o
GPT-4o mini
GPT-4-Turbo with Vision, GPT-4, GPT-3.5
DALL-E
Embeddings
Whisper, Text to speech

Phi models
Phi-3-mini
Phi-3-small
Phi-3-medium
Phi-3-vision
Phi-3.5-mini
Phi-3.5-vision
Phi-3.5-MoE
Phi-4
Phi-4-mini
Phi-4-omni

Meta
Meta-Llama-3-70B/70B-Instruct
Meta-Llama-3-8B/8B-Instruct
Meta-Llama-3.1-70B/70B-Instruct
Meta-Llama-3.1-8B/8B-Instruct
Meta-Llama-3.1-405B-Instruct
CodeLlama-7b/13b/34b/70b-Instruct/Python-hf

Mistral AI
Mistral Large
Mistral 7B
Mixtral 8x7B – Mixture of Experts
Ministral
Codestral

Cohere
Cohere R+
Cohere R
Cohere Rerank
Command and Embed
Embed v3-Multilingual
Embed v3-English

Hugging Face
Falcon/TII
Stable Diffusion/Stability AI
Dolly/Databricks
CLIP/OpenAI

Databricks
Databricks/dbrx-base
Databricks/dbrx-instruct

NVIDIA
Nemotron-3-8B-4k
Nemotron-3-8B-Chat-SFT/RLHF/SteerLM
Nemotron-3-8B-QA

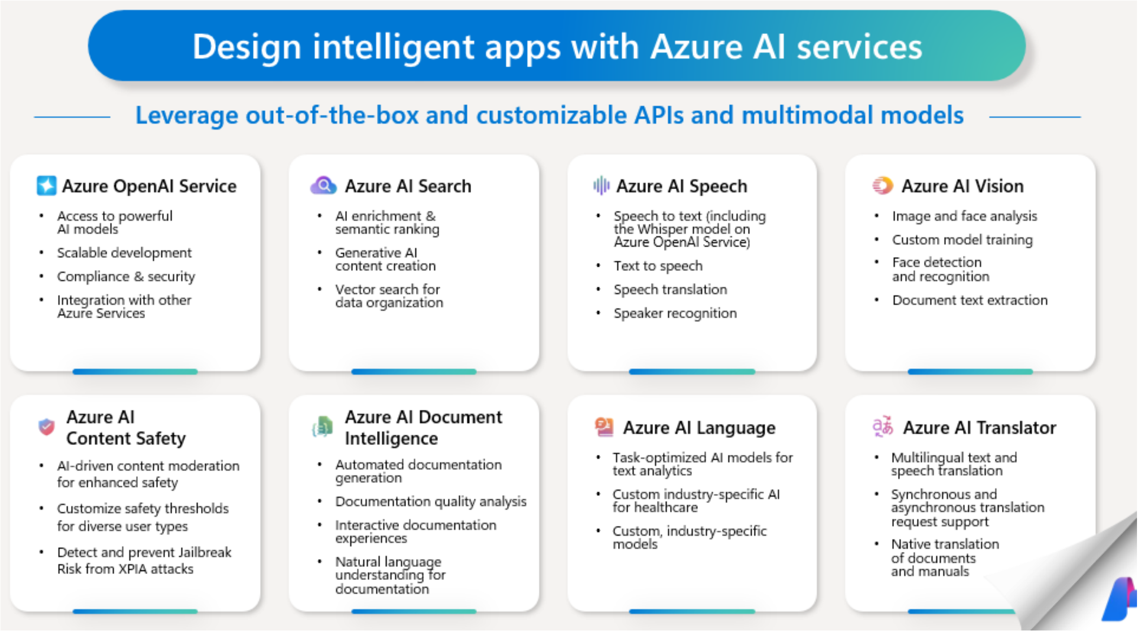
Snowflake
Snowflake/arctic-base
Snowflake/arctic-instruct

Others
AI21 Jamba 1.5 Large + Jamba 1.5 mini
Core42 JAIS
deci (Deci AI)
Nixtla TimeGEN-1
SDAIA (Arabic Model)
Bria AI
NTT Data Tsuzumi
DeepSeek-R1

Figure 38 AI Azure Foundry SDK

Example 2: Real-time Monitoring and Adjustment

Azure AI Foundry Portal offers tools for real-time monitoring of DBS effects. AI applications can dynamically adjust treatment parameters based on real-time feedback, ensuring that neuronal activity is modulated effectively. This approach can be particularly beneficial for managing conditions like migraines with aura and brain strokes.



Design intelligent apps with Azure AI services

Leverage out-of-the-box and customizable APIs and multimodal models

Azure OpenAI Service

- Access to powerful AI models
- Scalable development
- Compliance & security
- Integration with other Azure Services

Azure AI Search

- AI enrichment & semantic ranking
- Generative AI content creation
- Vector search for data organization

Azure AI Speech

- Speech to text (including the Whisper model on Azure OpenAI Service)
- Text to speech
- Speech translation
- Speaker recognition

Azure AI Vision

- Image and face analysis
- Custom model training
- Face detection and recognition
- Document text extraction

Azure AI Content Safety

- AI-driven content moderation for enhanced safety
- Customize safety thresholds for diverse user types
- Detect and prevent Jailbreak Risk from XPIA attacks

Azure AI Document Intelligence

- Automated documentation generation
- Documentation quality analysis
- Interactive documentation experiences
- Natural language understanding for documentation

Azure AI Language

- Task-optimized AI models for text analytics
- Custom industry-specific AI for healthcare
- Custom, industry-specific models

Azure AI Translator

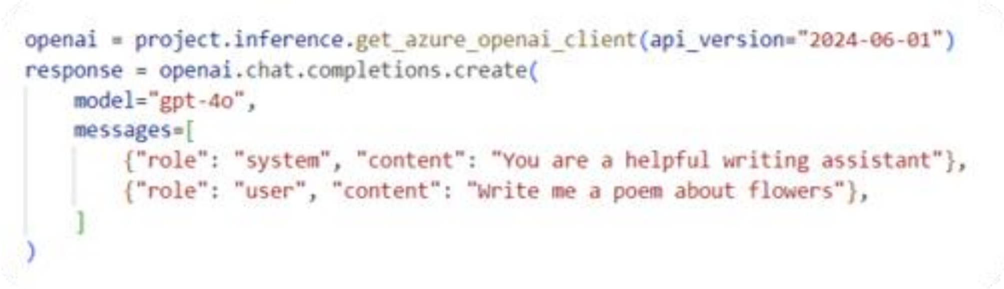
- Multilingual text and speech translation
- Synchronous and asynchronous translation request support
- Native translation of documents and manuals

Figure 39 Intelligent AI apps creation with AI Azure Foundry

Example 3: Personalized Treatment Plans

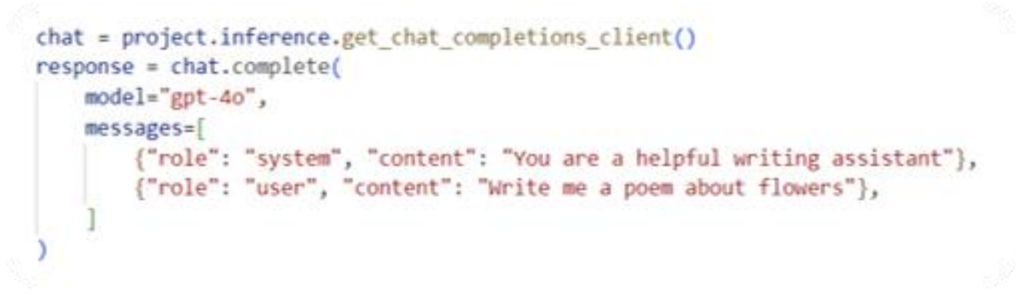
Retrieval-Augmented Generation (RAG) solutions in Azure AI Foundry allow AI models to access and utilize vast amounts of medical data. By integrating RAG solutions, researchers can develop personalized treatment plans tailored to individual patient needs, improving outcomes and reducing side effects.

Azure Open AI Services



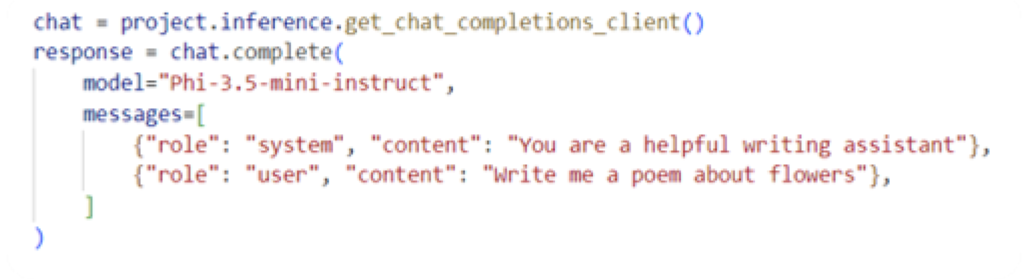
```
openai = project.inference.get_azure_openai_client(api_version="2024-06-01")
response = openai.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a helpful writing assistant"},
        {"role": "user", "content": "Write me a poem about flowers"},
    ]
)
```

Azure AI Services based on model of GPT-40



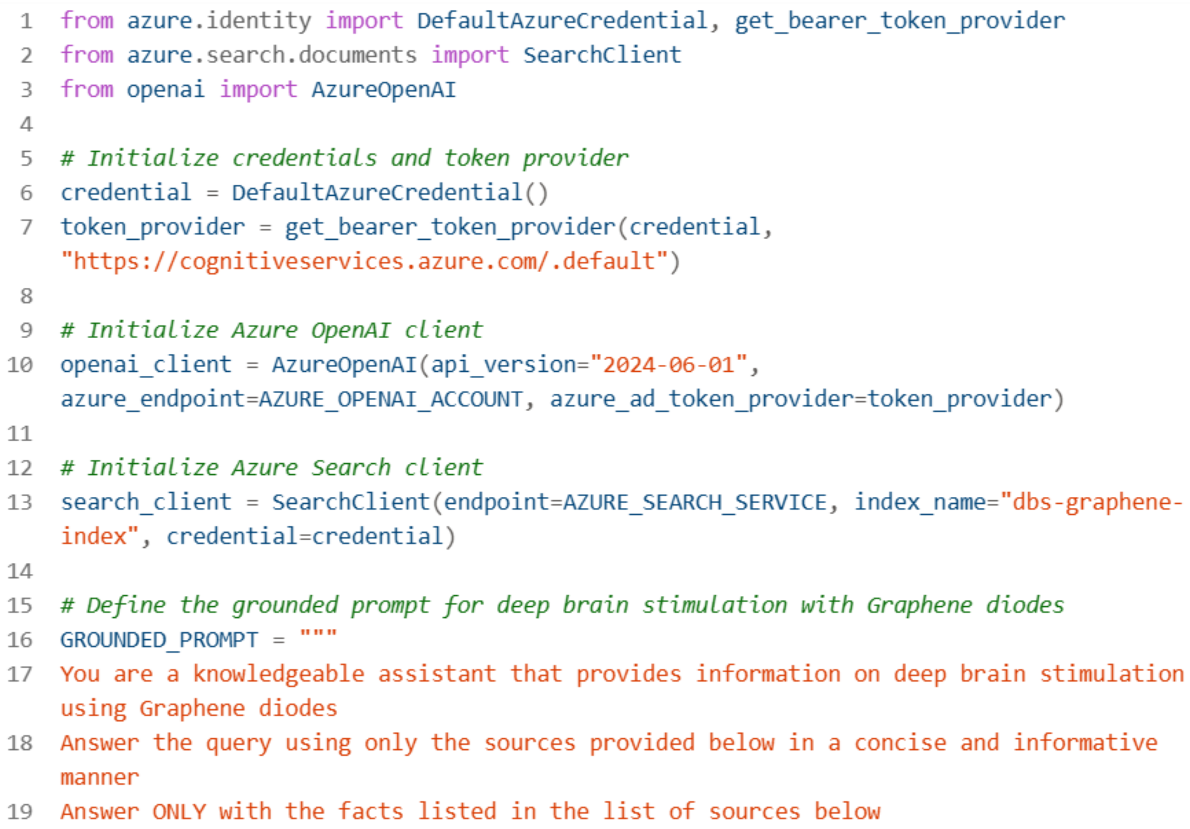
```
chat = project.inference.get_chat_completions_client()
response = chat.complete(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a helpful writing assistant"},
        {"role": "user", "content": "Write me a poem about flowers"},
    ]
)
```

AI model based on Phi-3.5 -mini-instruct

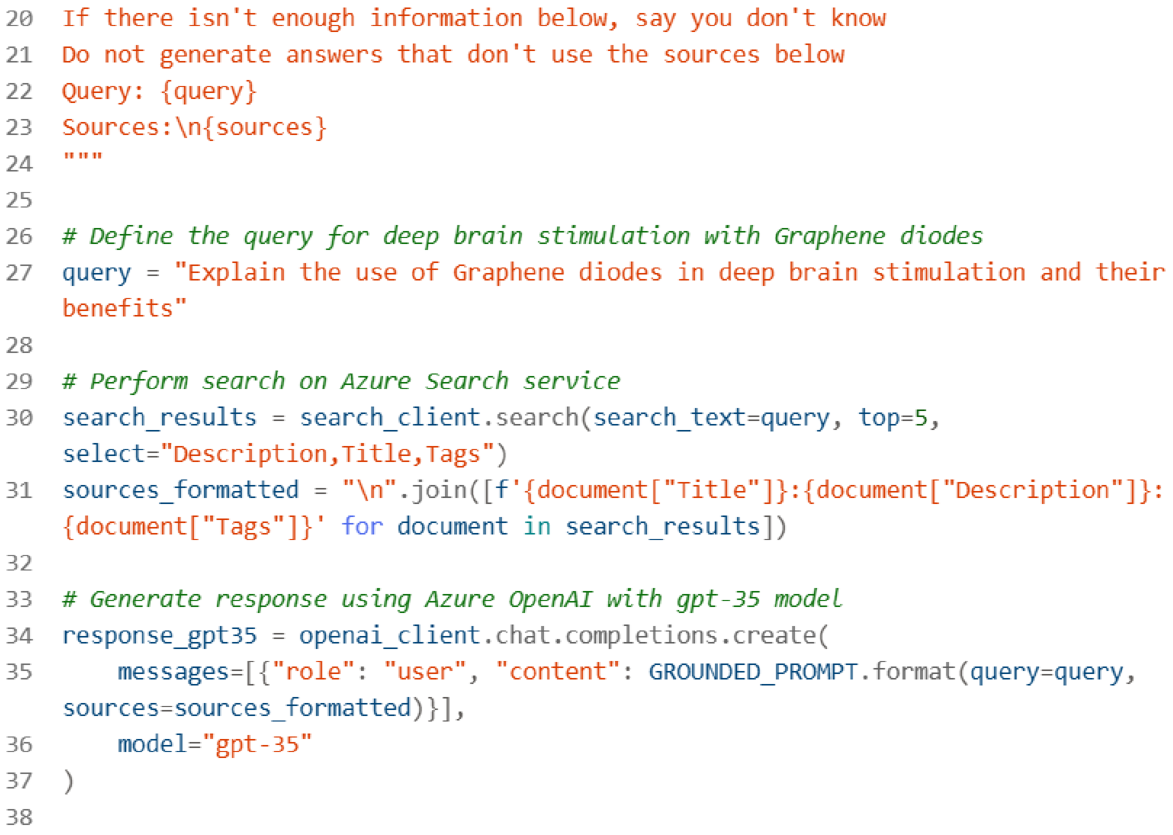


```
chat = project.inference.get_chat_completions_client()
response = chat.complete(
    model="Phi-3.5-mini-instruct",
    messages=[
        {"role": "system", "content": "You are a helpful writing assistant"},
        {"role": "user", "content": "Write me a poem about flowers"},
    ]
)
```

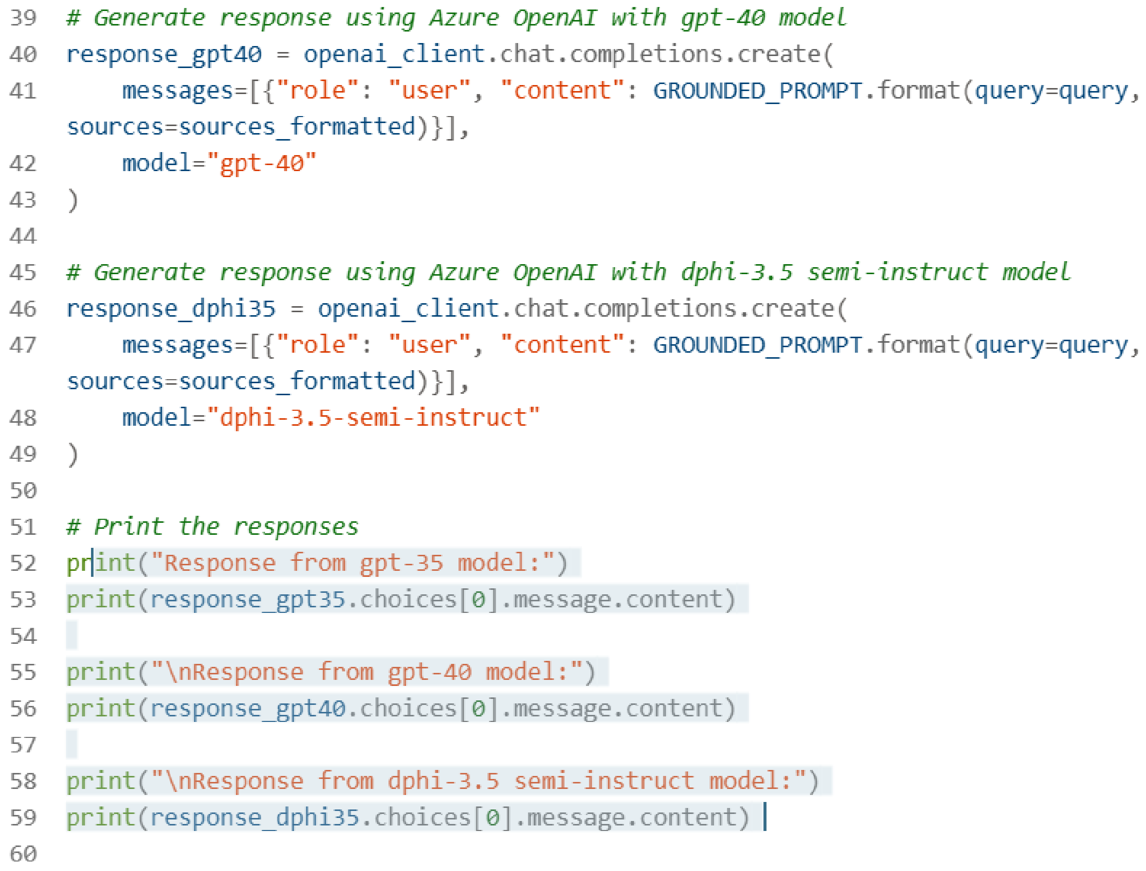
Python code for deep brain stimulation using Graphene diodes for RAG (Retrieval-Augmented Generation) based on the example of a RAG workflow in Python using Azure AI Search. Additionally, it includes the use of Azure OpenAI models gpt-40 and dphi-3.5 semi-instruct:



```python
from azure.identity import DefaultAzureCredential, get_bearer_token_provider
from azure.search.documents import SearchClient
from openai import AzureOpenAI

# Initialize credentials and token provider
credential = DefaultAzureCredential()
token_provider = get_bearer_token_provider(credential, "https://cognitiveservices.azure.com/.default")

# Initialize Azure OpenAI client
openai_client = AzureOpenAI(api_version="2024-06-01", azure_endpoint=AZURE_OPENAI_ACCOUNT, azure_ad_token_provider=token_provider)

# Initialize Azure Search client
search_client = SearchClient(endpoint=AZURE_SEARCH_SERVICE, index_name="dbs-graphene-index", credential=credential)

# Define the grounded prompt for deep brain stimulation with Graphene diodes
GROUNDED_PROMPT = """
You are a knowledgeable assistant that provides information on deep brain stimulation using Graphene diodes
Answer the query using only the sources provided below in a concise and informative manner
Answer ONLY with the facts listed in the list of sources below
```



```python
If there isn't enough information below, say you don't know
Do not generate answers that don't use the sources below
Query: {query}
Sources:\n{sources}
"""

# Define the query for deep brain stimulation with Graphene diodes
query = "Explain the use of Graphene diodes in deep brain stimulation and their benefits"

# Perform search on Azure Search service
search_results = search_client.search(search_text=query, top=5, select="Description,Title,Tags")
sources_formatted = "\n".join([f'{document["Title"]}:{document["Description"]}:{document["Tags"]}' for document in search_results])

# Generate response using Azure OpenAI with gpt-35 model
response_gpt35 = openai_client.chat.completions.create(
    messages=[{"role": "user", "content": GROUNDED_PROMPT.format(query=query, sources=sources_formatted)}],
    model="gpt-35"
)

```



```
# Generate response using Azure OpenAI with gpt-40 model
response_gpt40 = openai_client.chat.completions.create(
    messages=[{"role": "user", "content": GROUNDED_PROMPT.format(query=query,
sources=sources_formatted)}],
    model="gpt-40"
)

# Generate response using Azure OpenAI with dphi-3.5 semi-instruct model
response_dphi35 = openai_client.chat.completions.create(
    messages=[{"role": "user", "content": GROUNDED_PROMPT.format(query=query,
sources=sources_formatted)}],
    model="dphi-3.5-semi-instruct"
)

# Print the responses
print("Response from gpt-35 model:")
print(response_gpt35.choices[0].message.content)

print("\nResponse from gpt-40 model:")
print(response_gpt40.choices[0].message.content)

print("\nResponse from dphi-3.5 semi-instruct model:")
print(response_dphi35.choices[0].message.content)

```

This code sets up a workflow for deep brain stimulation using Graphene diodes with Retrieval-Augmented Generation (RAG) based on Azure AI Search. It initializes the necessary clients, defines a grounded prompt, performs a search, and generates responses using Azure OpenAI models gpt-35, gpt-40, and dphi-3.5 semi-instruct.

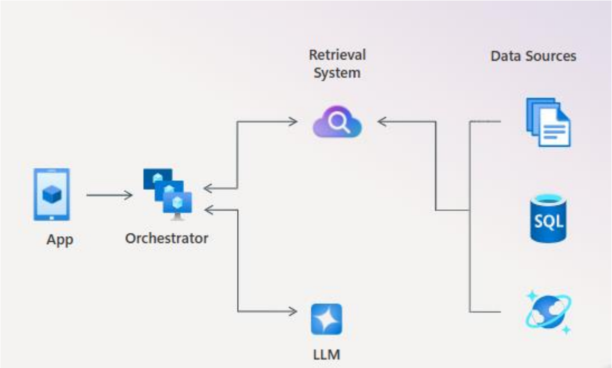
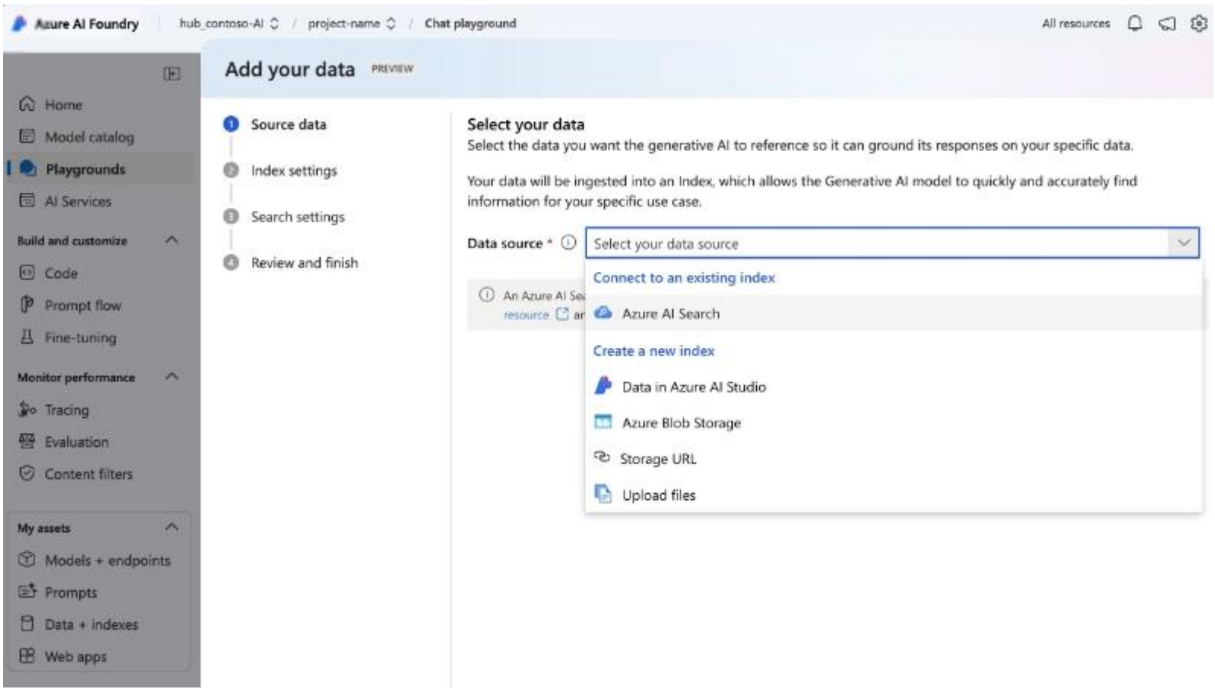


Figure 40 Microsoft Azure OpenAI models for DBS

Adding own data:



Select your data

Select the data you want the generative AI to reference so it can ground its responses on your specific data.

Your data will be ingested into an Index, which allows the Generative AI model to quickly and accurately find information for your specific use case.

Data source * Select your data source

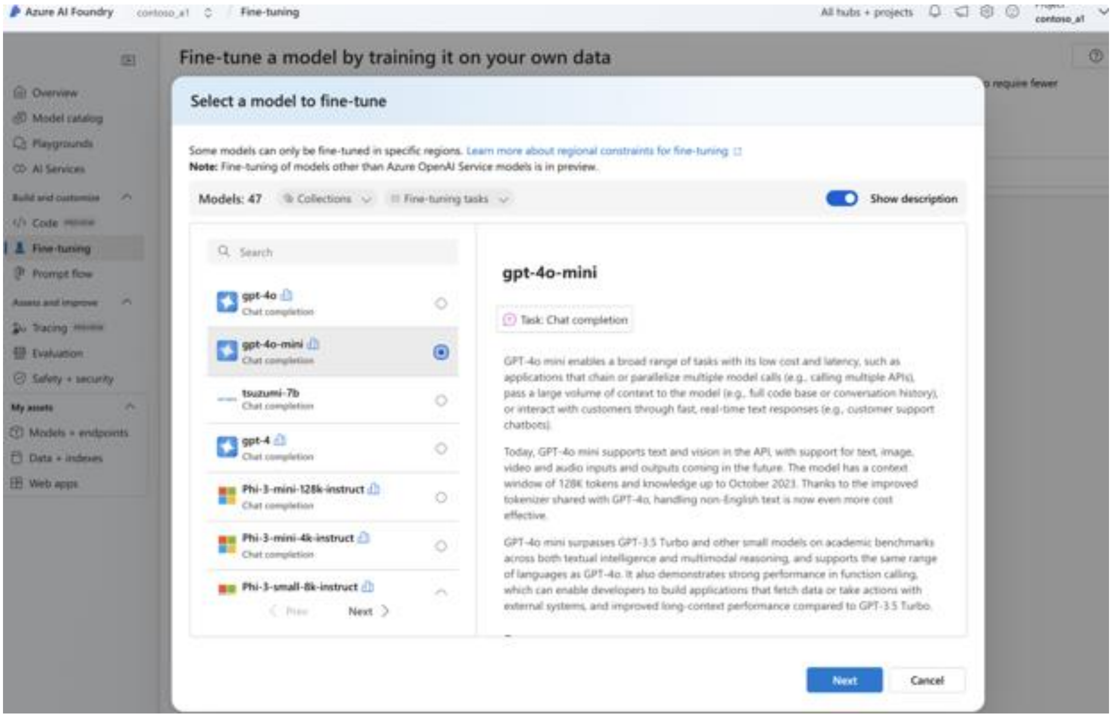
Figure 41 Data feeding mechanism with AI Foundry platform

There exists the possibility of integration multi-platforms and multi-vendor solutions, as below reference:

- **Azure AI Search**
- **Blob storage**
- **Local files/folders**
- **Storage URLs including OneLake in Microsoft Fabric**
  - Azure Databricks
  - S3 buckets via Amazon S3 Shortcuts

Example 4: Fine-tuning Language Models for Medical Applications

Fine-tuning a language model with Azure AI Foundry can enhance the precision of AI applications in DBS. Researchers can train base language models on specific medical tasks, such as chat-completion, to improve their performance in predicting and optimizing charge transfer mechanisms.



Fine-tune a model by training it on your own data

Select a model to fine-tune

Some models can only be fine-tuned in specific regions. Learn more about regional constraints for fine-tuning
Note: Fine-tuning of models other than Azure OpenAI Service models is in preview.

gpt-4o-mini

Task: Chat completion

GPT-4o mini enables a broad range of tasks with its low cost and latency, such as applications that chain or parallelize multiple model calls (e.g., calling multiple APIs), pass a large volume of context to the model (e.g., full code base or conversation history), or interact with customers through fast, real-time text responses (e.g., customer support chatbots).

Today, GPT-4o mini supports text and vision in the API, with support for text, image, video and audio inputs and outputs coming in the future. The model has a context window of 128K tokens and knowledge up to October 2023. Thanks to the improved tokenizer shared with GPT-4o, handling non-English text is now even more cost effective.

GPT-4o mini surpasses GPT-3.5 Turbo and other small models on academic benchmarks across both textual intelligence and multimodal reasoning, and supports the same range of languages as GPT-4o. It also demonstrates strong performance in function calling, which can enable developers to build applications that fetch data or take actions with external systems, and improved long-context performance compared to GPT-3.5 Turbo.

Figure 42 Fine-tune a model by training – data analysis from graphene diodes and measurements of charge transfer

There are three amin types of fine-tunning of apps performance:

- Supervised fine-tuning
- Direct preference optimization
- Reinforcement learning

Fine-Tuning of Performance of Apps in Azure AI Foundry for Deep Brain Stimulation

Azure AI Foundry offers various methods to fine-tune the performance of applications, ensuring they meet specific enterprise needs and deliver optimal results. Here are the three main types of fine-tuning available, specifically applied to deep brain stimulation (DBS) using Graphene diodes:

1. **Supervised Fine-Tuning**

   - Supervised fine-tuning involves training a pre-trained model on a labeled dataset specific to the task at hand. For DBS applications, this could mean using patient-specific data to train the model to predict optimal stimulation parameters. This method enhances the model's performance by providing it with examples of the desired output for given inputs, significantly improving the accuracy and relevance of the stimulation settings.

2. **Direct Preference Optimization**

    - Direct preference optimization focuses on aligning the model's behavior with human preferences. In the context of DBS, this could involve collecting feedback from patients and clinicians on the effectiveness of the stimulation settings and using this feedback to adjust the model's parameters. This ensures that the model's outputs are more aligned with patient-specific therapeutic goals, enhancing treatment outcomes.

3. **Reinforcement Learning**

    - Reinforcement learning is a method where the model learns to make decisions by receiving rewards or penalties based on its actions. For DBS, this could involve the model learning to optimize stimulation parameters based on real-time feedback from the patient's neurological responses. This approach helps the model to continuously improve its performance by learning from its interactions with the patient's brain.

Azure AI Foundry Capabilities

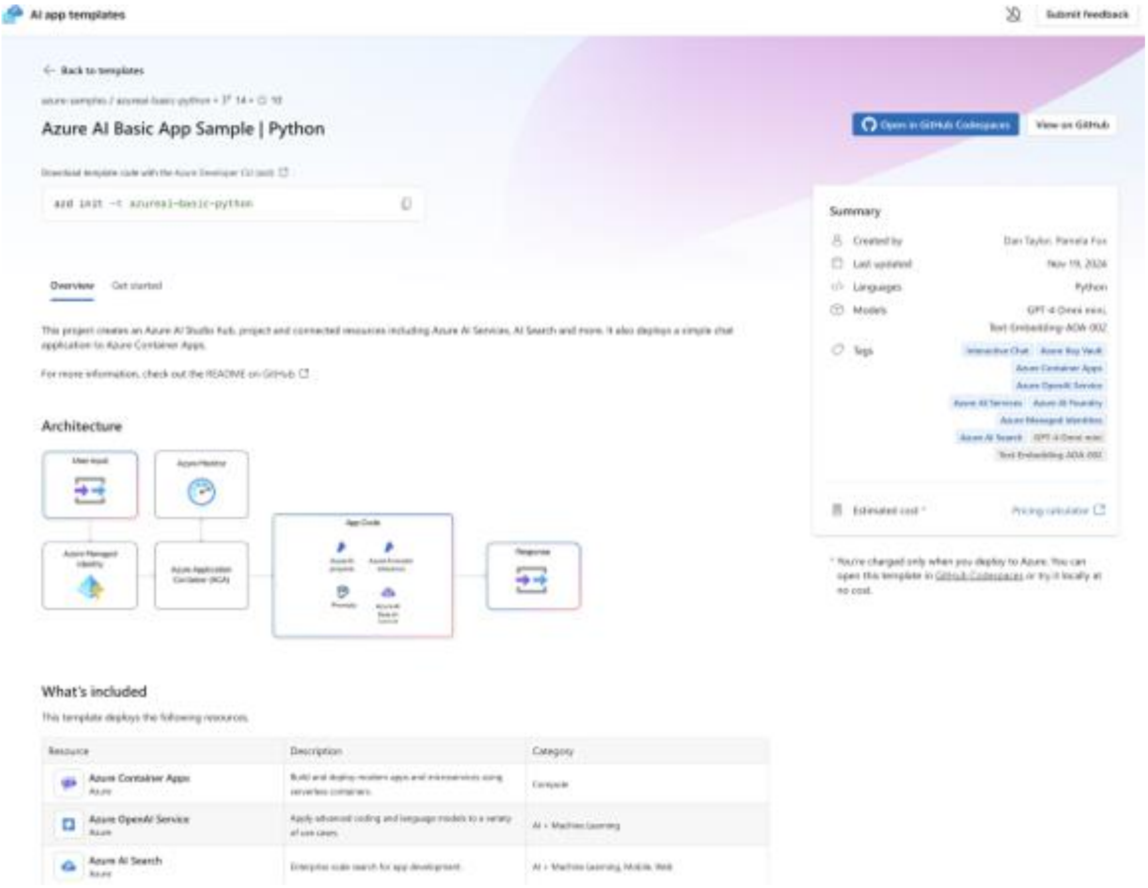
Azure AI Foundry provides a comprehensive platform for fine-tuning models, offering both managed and unmanaged options. Managed options include serverless fine-tuning and built-in ML OPS pipelines, which simplify the process and provide robust infrastructure support. Unmanaged options offer more control over the infrastructure, allowing for customized fine-tuning setups.

Practical Applications and Benefits

Fine-tuning in Azure AI Foundry can be applied to various domains, including medical applications like DBS. It allows for domain-specific performance improvements, making the models more effective in specialized tasks. Additionally, fine-tuning supports low-resource languages, enabling the development of applications that cater to diverse linguistic needs

Example of Fine-Tuning

An example of fine-tuning using Azure AI Foundry involves the use of Low-Rank Adaptation (LoRA) and Group Relative Policy Optimization (GRPO). LoRA is a parameter-efficient fine-tuning method that is computationally cheap, fast, and modular. GRPO simplifies the reinforcement learning process by removing the critic function and using a group baseline for optimization. This combination has been shown to improve the model's performance in reasoning and providing correct answers compared to the base model. Azure AI Foundry's fine-tuning capabilities provide a powerful toolset for optimizing the performance of AI applications, including those used in deep brain stimulation. By leveraging supervised fine-tuning, direct preference optimization, and reinforcement learning, organizations can enhance their models to meet specific needs and deliver superior results.



azure-samples / azureai-basic-python • 14 • 10

Azure AI Basic App Sample | Python

Download template code with the Azure Developer CLI (azd)

```
azd init -t azureai-basic-python
```

This project creates an Azure AI Studio hub, project and connected resources including Azure AI Services, AI Search and more. It also deploys a simple chat application to Azure Container Apps.

For more information, check out the README on GitHub

Summary

Created by	Dan Taylor, Pamela Fox
Last updated	Nov 19, 2024
Languages	Python
Models	GPT-4 Omni mini, Text Embedding ADA 002
Tags	Interactive Chat Azure Key Vault Azure Container Apps Azure OpenAI Service Azure AI Services Azure AI Foundry Azure Managed Identities Azure AI Search GPT-4 Omni mini Text Embedding ADA 002
Estimated cost *	Pricing calculator

* You're charged only when you deploy to Azure. You can open this template in GitHub Codespaces or try it locally at no cost.

Architecture

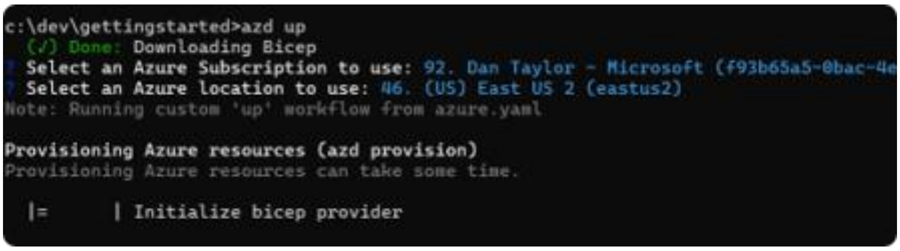
What's included

This template deploys the following resources.

Resource	Description	Category
Azure Container Apps Azure	Build and deploy modern apps and microservices using serverless containers.	Compute
Azure OpenAI Service Azure	Apply advanced coding and language models to a variety of use cases.	AI + Machine Learning
Azure AI Search Azure	Enterprise scale search for app development.	AI + Machine Learning, Mobile, Web

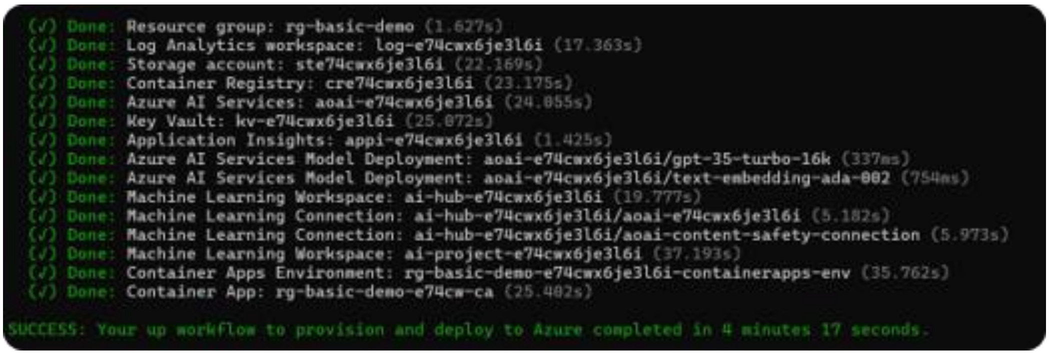
Figure 43 Deployment of a model and application with command line interface.

Run azd app



```
c:\dev\gettingstarted>azd up
  (✓) Done: Downloading Bicep
? Select an Azure Subscription to use: 92. Dan Taylor - Microsoft (f93b65a5-0bac-4e
? Select an Azure location to use: 46. (US) East US 2 (eastus2)
Note: Running custom 'up' workflow from azure.yaml

Provisioning Azure resources (azd provision)
Provisioning Azure resources can take some time.

  |=      | Initialize bicep provider
```

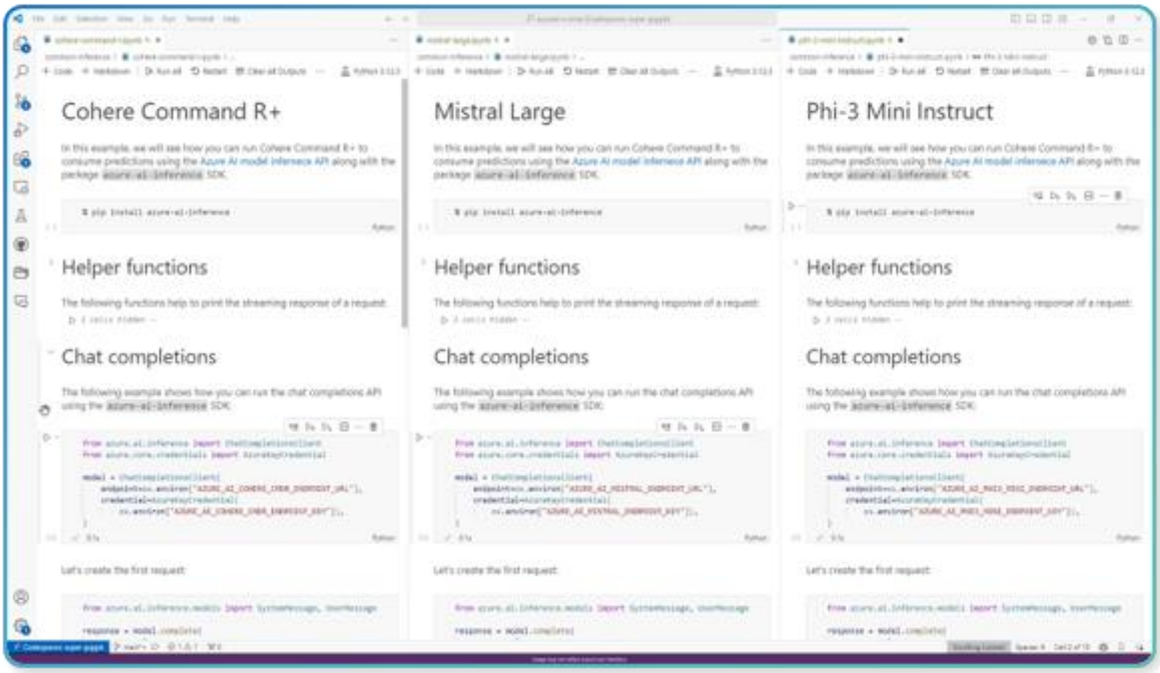
Resources creation and app deployment



```
  (✓) Done: Resource group: rg-basic-demo (1.627s)
  (✓) Done: Log Analytics workspace: log-e74cwx6je3l6i (17.363s)
  (✓) Done: Storage account: ste74cwx6je3l6i (22.169s)
  (✓) Done: Container Registry: cre74cwx6je3l6i (23.175s)
  (✓) Done: Azure AI Services: aoai-e74cwx6je3l6i (24.855s)
  (✓) Done: Key Vault: kv-e74cwx6je3l6i (25.072s)
  (✓) Done: Application Insights: appi-e74cwx6je3l6i (1.425s)
  (✓) Done: Azure AI Services Model Deployment: aoai-e74cwx6je3l6i/gpt-35-turbo-16k (337ms)
  (✓) Done: Azure AI Services Model Deployment: aoai-e74cwx6je3l6i/text-embedding-ada-002 (754ms)
  (✓) Done: Machine Learning Workspace: ai-hub-e74cwx6je3l6i (19.777s)
  (✓) Done: Machine Learning Connection: ai-hub-e74cwx6je3l6i/aoai-e74cwx6je3l6i (5.182s)
  (✓) Done: Machine Learning Connection: ai-hub-e74cwx6je3l6i/aoai-content-safety-connection (5.973s)
  (✓) Done: Machine Learning Workspace: ai-project-e74cwx6je3l6i (37.193s)
  (✓) Done: Container Apps Environment: rg-basic-demo-e74cwx6je3l6i-containerapps-env (35.762s)
  (✓) Done: Container App: rg-basic-demo-e74cw-ca (25.402s)

SUCCESS: Your up workflow to provision and deploy to Azure completed in 4 minutes 17 seconds.
```

Example 5: Responsible Implementation and Evaluation

Implementing responsible generative AI solutions in Azure AI Foundry ensures that AI applications are safe and trustworthy. The platform provides tools for evaluating the performance of generative AI applications, helping researchers refine their models and minimize the risk of harmful content generation.



Cohere Command R+

In this example, we will see how you can run Cohere Command R+ to consume predictions using the Azure AI model inference API along with the package azure-ai-inference SDK.

```
# pip install azure-ai-inference
```

Helper functions

The following functions help to print the streaming response of a request:

Chat completions

The following example shows how you can run the chat completions API using the azure-ai-inference SDK.

```
from azure.ai.inference import ChatCompletionsClient
from azure.core.credentials import AzureKeyCredential

model = ChatCompletionsClient(
    endpoint=os.environ["AZURE_AI_COHERE_CMDR_ENDPOINT_URL"],
    credential=AzureKeyCredential(
        os.environ["AZURE_AI_COHERE_CMDR_ENDPOINT_KEY"]),
)
```

Let's create the first request:

```
from azure.ai.inference.models import SystemMessage, UserMessage

response = model.complete(
```

Mistral Large

In this example, we will see how you can run Cohere Command R+ to consume predictions using the Azure AI model inference API along with the package azure-ai-inference SDK.

```
# pip install azure-ai-inference
```

Helper functions

The following functions help to print the streaming response of a request:

Chat completions

The following example shows how you can run the chat completions API using the azure-ai-inference SDK.

```
from azure.ai.inference import ChatCompletionsClient
from azure.core.credentials import AzureKeyCredential

model = ChatCompletionsClient(
    endpoint=os.environ["AZURE_AI_MISTRAL_ENDPOINT_URL"],
    credential=AzureKeyCredential(
        os.environ["AZURE_AI_MISTRAL_ENDPOINT_KEY"]),
)
```

Let's create the first request:

```
from azure.ai.inference.models import SystemMessage, UserMessage

response = model.complete(
```

Phi-3 Mini Instruct

In this example, we will see how you can run Cohere Command R+ to consume predictions using the Azure AI model inference API along with the package azure-ai-inference SDK.

```
# pip install azure-ai-inference
```

Helper functions

The following functions help to print the streaming response of a request:

Chat completions

The following example shows how you can run the chat completions API using the azure-ai-inference SDK.

```
from azure.ai.inference import ChatCompletionsClient
from azure.core.credentials import AzureKeyCredential

model = ChatCompletionsClient(
    endpoint=os.environ["AZURE_AI_PHI3_MINI_ENDPOINT_URL"],
    credential=AzureKeyCredential(
        os.environ["AZURE_AI_PHI3_MINI_ENDPOINT_KEY"]),
)
```

Let's create the first request:

```
from azure.ai.inference.models import SystemMessage, UserMessage

response = model.complete(
```

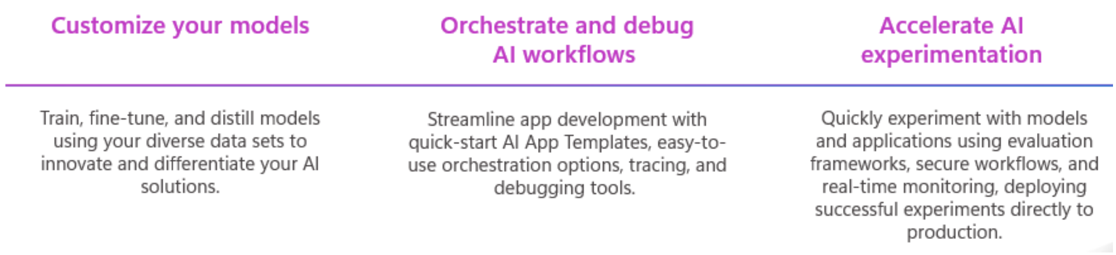
Figure 44 MASS with R and Azure AI Foundry results analysis.

Mechanism with LLM, Generative AI, AI/ML, and MASS Model with R:

- Large Language Models (LLM): LLMs can be used to analyze patient data and generate personalized treatment plans. They can interpret complex medical data and provide insights into the most effective stimulation parameters for graphene diodes.
- Generative AI: Generative AI can simulate various treatment scenarios and predict outcomes based on different stimulation patterns. This helps in optimizing the treatment process and ensuring the best possible results for patients.
- AI/ML: AI and ML algorithms can continuously monitor patient responses to treatment, adjusting the stimulation parameters in real-time to maximize efficacy. These technologies can also identify patterns in patient data that may indicate the need for changes in the treatment approach.
- MASS Model with R: The MASS package in R provides statistical tools to analyze the effectiveness of the treatment. By applying these tools, researchers can evaluate the impact of graphene diodes and charge transfer mechanisms on recovery processes, comparing results between early-born infants and seniors.
- Stimulation and Organism Response: The stimulation process involves the application of graphene diodes to the cortex of the brain, where they facilitate charge transfer to

modulate neuronal activity. The organism's response to this stimulation can be monitored using AI/ML algorithms, which adjust the stimulation parameters in real-time to ensure optimal results.

Generative AI applications for DBS (Database Systems) with Model as a Service of Azure AI Foundry offer a range of capabilities that can significantly enhance the efficiency and effectiveness of database management and operations. Here are some key insights and applications:



Customize your models

Train, fine-tune, and distill models using your diverse data sets to innovate and differentiate your AI solutions.

Orchestrate and debug AI workflows

Streamline app development with quick-start AI App Templates, easy-to-use orchestration options, tracing, and debugging tools.

Accelerate AI experimentation

Quickly experiment with models and applications using evaluation frameworks, secure workflows, and real-time monitoring, deploying successful experiments directly to production.

Evaluation and Performance Assessment

Azure AI Foundry provides tools to thoroughly assess the performance of generative AI models and applications when applied to substantial datasets. This evaluation process involves testing the model or application with the given dataset and measuring its performance using both mathematical and AI-assisted metrics. The Azure AI Foundry portal offers comprehensive insights into the application's capabilities and limitations, allowing users to log, view, and analyze detailed evaluation metrics.

Operational Excellence

Azure AI Foundry offers practical, personalized guidance for testing generative AI operations knowledge, practices, and overall experience level. This platform helps organizations to quickly assess their generative AI operations and provides step-by-step guidance through the AI app development process, from prototyping to optimizing and operationalizing.

Customization and Specialized Solutions

Azure AI Foundry's AI model catalogue includes over 1,800 options, offering tailored and specialized task and industry-specific models. This flexibility ensures that organizations can explore AI models that advance their business priorities. The platform also supports fine-tuning capabilities, such as vision fine-tuning and distillation workflows, which allow smaller models to replicate the behaviour of larger models.

Integration and Automation

Azure AI Foundry enables the integration of external data sources and streamlines the process of building and consuming APIs. This includes plugins for Azure Cognitive Search, Azure SQL, Azure Cosmos DB, Microsoft Translator, and Bing Search. The platform also offers a Provisioned Throughput Model for dedicated capacity, enhancing developer productivity and simplifying the integration process.

Security and Trustworthiness

Securing generative AI models on Azure AI Foundry is crucial for maintaining robust security while leveraging new advancements. Microsoft focuses on making its AI development platform a secure and trustworthy place for exploration and innovation. This involves thoughtful risk assessment to balance leveraging new advancements with maintaining robust security.

Research and Development

The Audio and Acoustics Research Group at Microsoft Research (MSR) highlights the role of audio in generative AI. Audio-language models can generate captions, labels, or free-form text from audio signals, facilitating applications such as question-answering. The generation of audio from prior audio inputs or other modalities leads to fascinating multi-modal models and applications.

Learning and Documentation

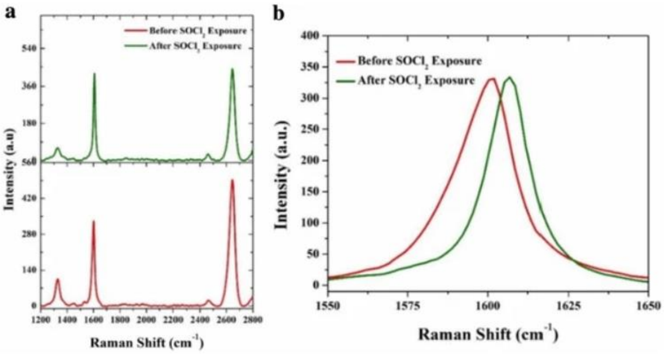
Azure AI Foundry provides extensive learning resources, including training plans, documentation, and e-books. These resources guide users through the AI app development process and help them scale their projects. The platform also offers insights into AI adoption through Azure AI Foundry's Model as a Service, which enables organizations to enhance their database systems with advanced generative AI capabilities, ensuring operational excellence, customization, integration, security, and continuous learning and development.

Examples:

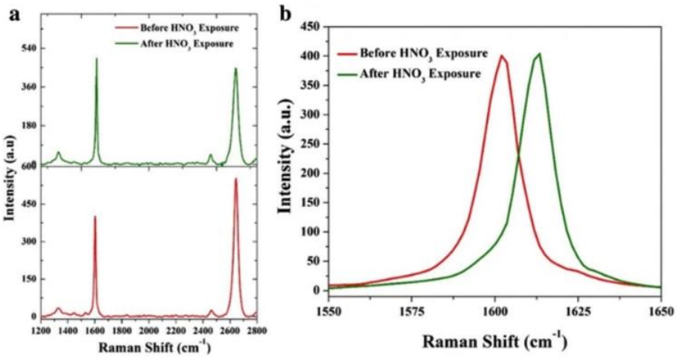
Condition	**Stimulation Method**	**Response Method**	**Application**	**Expected Outcome**
Migraine with Aura	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Modulating brain activity	Reduced migraine symptoms
Brain Strokes	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Enhancing motor recovery	Improved mobility and coordination
Early-born Infants	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Developing motor skills with quick brain recovery	Enhanced motor development and process of healing
Seniors	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Regaining motor functions	Better mobility and reduced disability

Table 4 Methods of brain stimulation summary

By integrating these advanced techniques and technologies, we can develop more effective treatment strategies for neurological conditions, tailored to the specific needs of different patient groups. This holistic approach can significantly improve recovery outcomes and quality of life for both early-born infants and seniors.

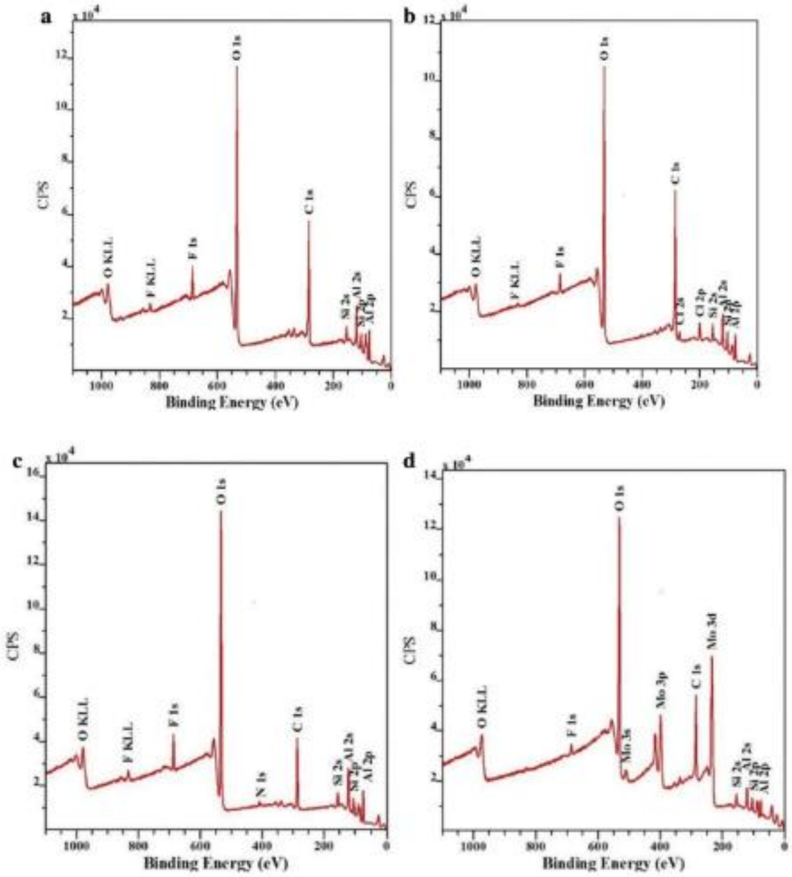


Raman spectra of graphene sample before and after $SOCl_2$ treatment: **a** full spectra, **b** G band region. (Color figure online)



Raman spectra of graphene sample before and after HNO_3 treatment: **a** full spectra, **b** G band region. (Color figure online)

Figure 45 Raman spectra of graphene sample before and after HNO3 treatment in DBS



The XPS survey spectra of pristine graphene and after doping. **a** Pristine, **b** $SOCl_2$, **c** HNO_3 and **d** MoO_3

Figure 46 Source: ***Understanding noninvasive charge transfer doping of graphene: a comparative study | Journal of Materials Science: Materials in Electronics***

Change transfer and graphene diodes can potentially stimulate muscle tension relief and brain activity through mechanisms involving electrical stimulation. Here's how they might work:

1. **Electrical Stimulation**: Graphene diodes can be used to deliver precise electrical impulses to muscles and nerves. This technique, known as neuromuscular electrical stimulation (NMES), can activate muscles artificially, causing them to contract and relax. This can help relieve muscle tension and reduce spasticity.

2. **Neuroplasticity**: Electrical stimulation can also promote neuroplasticity, which is the brain's ability to reorganize itself by forming new neural connections.

3. By stimulating specific areas of the brain and muscles, it may be possible to enhance the brain's ability to manage pain and prevent migraines with aura or even reduce the risk of stroke.

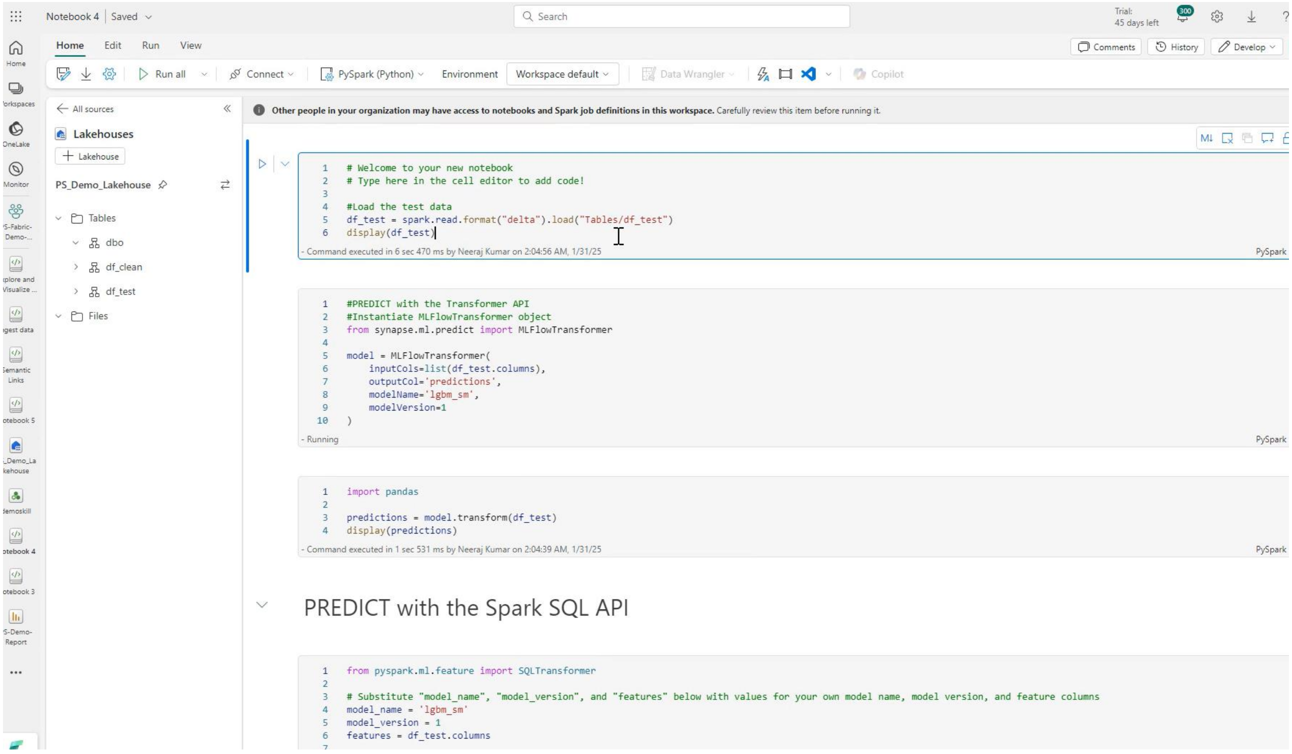
4. **Functional Electrical Stimulation (FES)**: This technique involves using electrical impulses to generate functional movements, such as grasping or walking. FES can induce short- and long-term neurophysiological changes in the central nervous system, potentially improving motor function and reducing the frequency of migraines.

5. **Preventive Strategies**: Lifestyle changes, such as maintaining a healthy weight, avoiding smoking, and controlling hypertension, can also help reduce the risk of migraines with aura and stroke.

Combining these strategies with electrical stimulation techniques may offer a comprehensive approach to managing these conditions.

Microsoft Fabric

Microsoft Fabric enhances machine learning (ML) through several key features and capabilities:



Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

```
# Welcome to your new notebook
# Type here in the cell editor to add code!

#Load the test data
df_test = spark.read.format("delta").load("Tables/df_test")
display(df_test)
```

- Command executed in 6 sec 470 ms by Neeraj Kumar on 2:04:56 AM, 1/31/25

```
#PREDICT with the Transformer API
#Instantiate MLFlowTransformer object
from synapse.ml.predict import MLFlowTransformer

model = MLFlowTransformer(
    inputCols=list(df_test.columns),
    outputCol='predictions',
    modelName='lgbm_sm',
    modelVersion=1
)
```

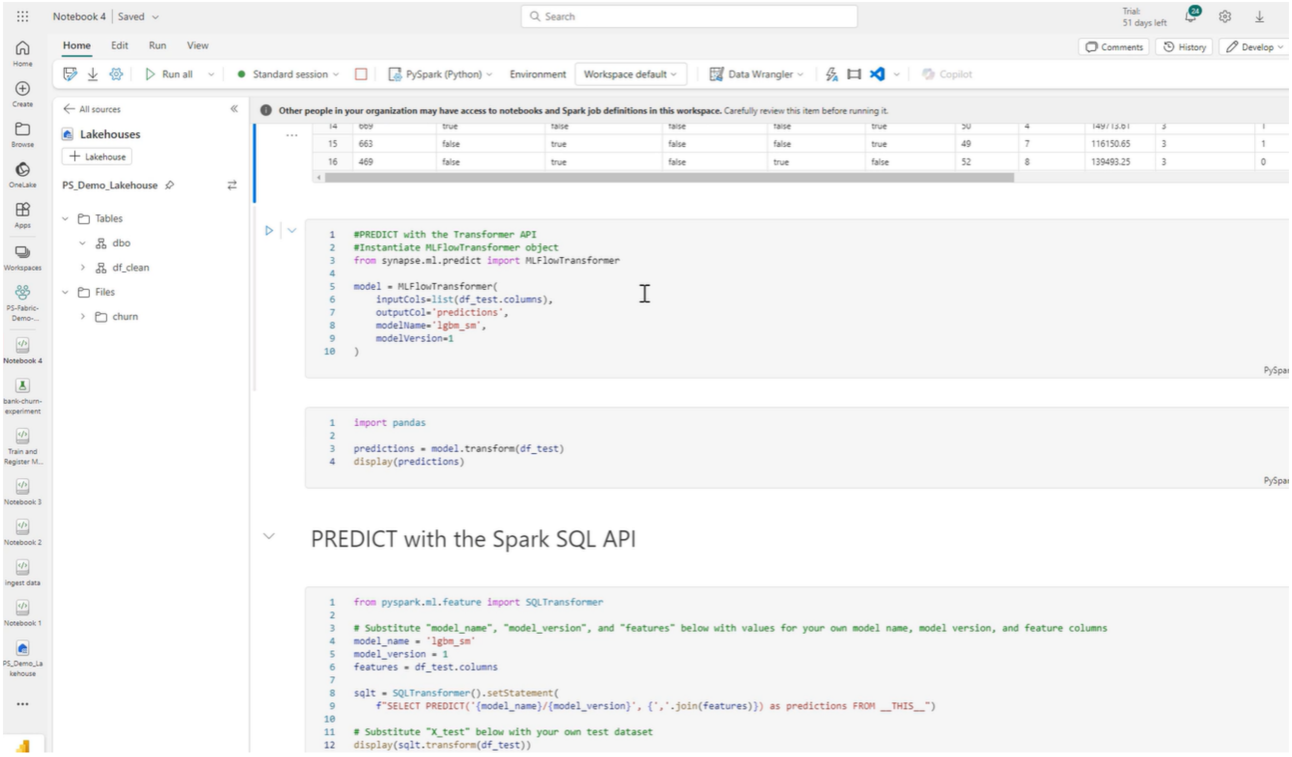
- Running

```
import pandas

predictions = model.transform(df_test)
display(predictions)
```

- Command executed in 1 sec 531 ms by Neeraj Kumar on 2:04:39 AM, 1/31/25

PREDICT with the Spark SQL API

```
from pyspark.ml.feature import SQLTransformer

# Substitute "model_name", "model_version", and "features" below with values for your own model name, model version, and feature columns
model_name = 'lgbm_sm'
model_version = 1
features = df_test.columns
```



Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

14	669	true	false	false	false	true	50	4	149713.61	3	1
15	663	false	true	false	false	true	49	7	116150.65	3	1
16	469	false	true	false	true	false	52	8	139493.25	3	0

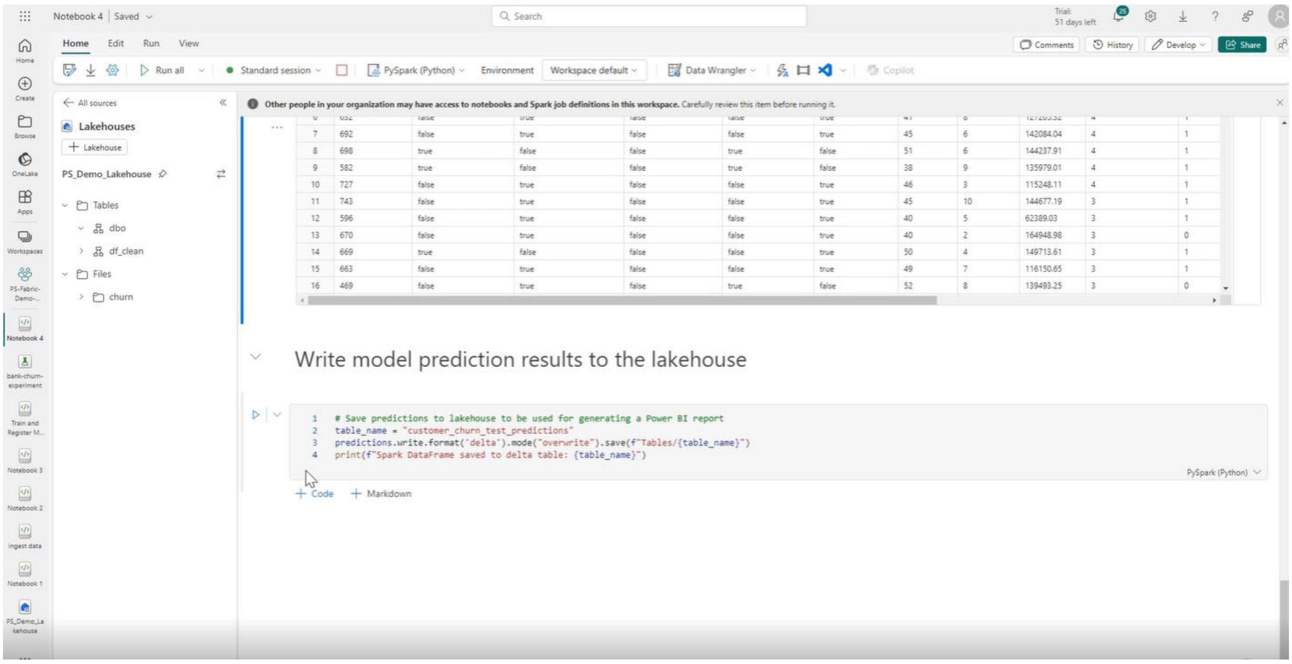
```
#PREDICT with the Transformer API
#Instantiate MLFlowTransformer object
from synapse.ml.predict import MLFlowTransformer

model = MLFlowTransformer(
    inputCols=list(df_test.columns),
    outputCol='predictions',
    modelName='lgbm_sm',
    modelVersion=1
)
```

```
import pandas

predictions = model.transform(df_test)
display(predictions)
```

PREDICT with the Spark SQL API

```
from pyspark.ml.feature import SQLTransformer

# Substitute "model_name", "model_version", and "features" below with values for your own model name, model version, and feature columns
model_name = 'lgbm_sm'
model_version = 1
features = df_test.columns

sqlt = SQLTransformer().setStatement(
    f"SELECT PREDICT('{model_name}/{model_version}', {','.join(features)}) as predictions FROM __THIS__")

# Substitute "X_test" below with your own test dataset
display(sqlt.transform(df_test))
```

Predictions are transferred to Data Lakehouse



Other people in your organization may have access to notebooks and Spark job definitions in this workspace. Carefully review this item before running it.

7	692	false	true	false	false	true	45	6	142084.04	4	1
8	698	true	false	false	true	false	51	6	144237.91	4	1
9	582	true	false	false	true	false	38	9	135979.01	4	1
10	727	false	true	false	false	true	46	3	115248.11	4	1
11	743	false	true	false	false	true	45	10	144677.19	3	1
12	596	false	true	false	false	true	40	5	62389.03	3	1
13	670	false	true	false	false	true	40	2	164948.98	3	0
14	669	true	false	false	false	true	50	4	149713.61	3	1
15	663	false	true	false	false	true	49	7	116150.65	3	1
16	469	false	true	false	true	false	52	8	139493.25	3	0

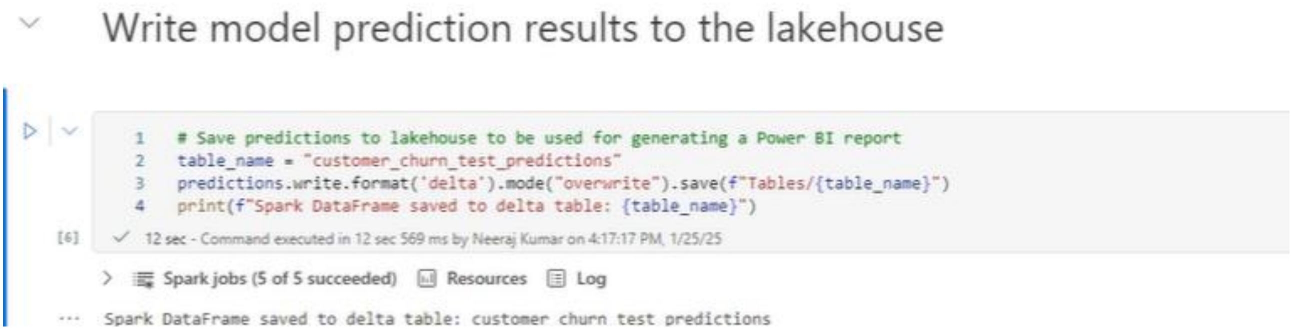
Write model prediction results to the lakehouse

```
# Save predictions to lakehouse to be used for generating a Power BI report
table_name = "customer_churn_test_predictions"
predictions.write.format('delta').mode("overwrite").save(f"Tables/{table_name}")
print(f"Spark DataFrame saved to delta table: {table_name}")
```

PySpark (Python)

Figure 48 MS Fabric with Data Lakehouse for ML and results validation

And next written to PowerBI:



Write model prediction results to the lakehouse

```
# Save predictions to lakehouse to be used for generating a Power BI report
table_name = "customer_churn_test_predictions"
predictions.write.format('delta').mode("overwrite").save(f"Tables/{table_name}")
print(f"Spark DataFrame saved to delta table: {table_name}")
```

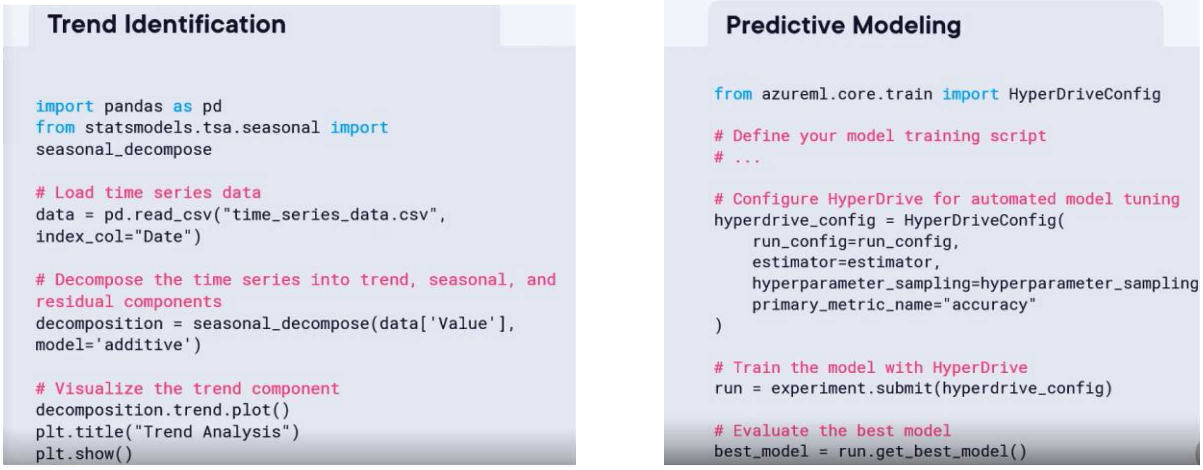
[6] ✓ 12 sec - Command executed in 12 sec 569 ms by Neeraj Kumar on 4:17:17 PM, 1/25/25

```
Spark DataFrame saved to delta table: customer churn test predictions
```

AutoML (Automated Machine Learning):

Simplification and Speed: AutoML in Microsoft Fabric automates the process of training and optimizing ML models, reducing the need for extensive human involvement. This speeds up the model development process and simplifies the selection of the best model and hyperparameters for a given dataset.

MS Fabric transforms data with Azure ML via Trend Identification and Prediction models with Python, as reference or Julia language:



Trend Identification

```
import pandas as pd
from statsmodels.tsa.seasonal import
seasonal_decompose

# Load time series data
data = pd.read_csv("time_series_data.csv",
index_col="Date")

# Decompose the time series into trend, seasonal, and
residual components
decomposition = seasonal_decompose(data['Value'],
model='additive')

# Visualize the trend component
decomposition.trend.plot()
plt.title("Trend Analysis")
plt.show()
```

Predictive Modeling

```
from azureml.core.train import HyperDriveConfig

# Define your model training script
# ...

# Configure HyperDrive for automated model tuning
hyperdrive_config = HyperDriveConfig(
    run_config=run_config,
    estimator=estimator,
    hyperparameter_sampling=hyperparameter_sampling
    primary_metric_name="accuracy"
)

# Train the model with HyperDrive
run = experiment.submit(hyperdrive_config)

# Evaluate the best model
best_model = run.get_best_model()
```

Figure 49 Model prediction conforming results with MLFlow in Python or Julia SW language.

Integration with MLflow: Fabric integrates with MLflow, allowing users to track and compare different model versions and experiment runs. This helps in managing the lifecycle of ML models efficiently.

FLAML (Fast and Lightweight AutoML)

Efficiency: FLAML powers the AutoML capabilities in Fabric, enabling users to build, optimize, and deploy ML models seamlessly within the platform's data ecosystem. This ensures that ML solutions are developed quickly and efficiently.

Data Management and Analysis

Comprehensive Tools: Microsoft Fabric provides a suite of tools for data management and analysis, supporting the entire ML workflow from data preparation to model deployment. This includes handling various data formats and integrating with other data science tools.

User-Friendly Interface

Ease of Use: Fabric offers a user-friendly interface for creating and managing ML models. Users can create new models directly from the Fabric UI or use APIs for more advanced operations. This makes it accessible to both novice and experienced data scientists.

Collaboration and Integration

Collaborative Environment: Fabric supports collaboration among data scientists and developers, facilitating the sharing of models and insights. This enhances the overall productivity and innovation within ML projects.

Additionally, in the process of developing my own ML solutions, there is already a possibility to leverage existing market solutions like Microsoft Fabric. This can significantly reduce development time and costs, allowing me to focus on customizing and optimizing my specific applications.

By leveraging these features, Microsoft Fabric significantly enhances the efficiency, accuracy, and scalability of machine learning projects, making it a powerful tool for data scientists and ML solutions whether throughout REST API connectivity or direct linking with Dataverse enabling access to ETL data ecosystem of own hosted solutions for Deep Brain Stimulation.

Initiating Charge Transfer through Graphene Diodes and Its Effects on Muscle Tension Relief and Brain Stimulation

Theoretical Model for PhD Thesis:

Introduction: Graphene diodes, due to their exceptional electrical conductivity and biocompatibility, can be integrated with charge transfer mechanisms to stimulate neuronal activity. This approach can be particularly beneficial for treating neurological conditions such as migraines with aura and brain strokes. The treatment involves the application of graphene diodes to specific regions of the brain, where they facilitate charge transfer to modulate neuronal activity.

Technical Description:

1. **Graphene Diodes and Charge Transfer Mechanism**:
    - **Graphene Diodes**: Graphene diodes are semiconductor devices that utilize graphene's unique electronic properties to facilitate charge transfer. The high conductivity and flexibility of graphene make it an ideal material for interfacing with biological tissues.
    - **Charge Transfer**: Charge transfer in graphene diodes occurs through the movement of electrons across the graphene-semiconductor interface. This process is influenced by the Fermi level shift of graphene, which can be modulated by external electrical stimulations.
2. **Application to Neuronal Stimulation**:
    - **Electrode Placement**: Graphene diodes are placed on the cortex of the brain, targeting specific regions responsible for motor and sensory functions.
    - **Electrical Stimulation**: Electrical impulses are applied to the graphene diodes, causing a redistribution of charge within the neurons. This can result in either hyperpolarization or depolarization, depending on the stimulation parameters.

Medical Description:

1. **Muscle Tension Relief**:
   - **Mechanism**: The application of graphene diodes and charge transfer mechanisms can enhance blood circulation and tissue temperature, leading to muscle relaxation and pain relief. This is achieved through capacitive and resistive electronic transfer, which provides endogenous thermal treatment to the affected muscles.
   - **Clinical Application**: For patients with musculoskeletal disorders, graphene diodes can be used to target specific muscle groups, reducing tension and improving mobility.
2. **Brain Stimulation**:
   - **Mechanism**: Electrical stimulation through graphene diodes can modulate neuronal activity by altering the charge distribution within the neurons. This can enhance motor recovery and sensory functions, particularly in patients with neurological conditions such as migraines with aura and brain strokes.
   - **Clinical Application**: Deep brain stimulation (DBS) using graphene diodes can be tailored to individual patients by adjusting the stimulation parameters based on real-time monitoring and AI/ML algorithms.

Examples:

Condition	**Stimulation Method**	**Response Method**	**Application**	**Expected Outcome**
Migraine with Aura	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Modulating brain activity	Reduced migraine symptoms. Alleviation of muscle tension
Brain Strokes	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Enhancing motor recovery	Improved mobility and coordination

Condition	**Stimulation Method**	**Response Method**	**Application**	**Expected Outcome**
Early-born Infants	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Developing motor skills	Enhanced motor development
Seniors	Graphene Diodes + Charge Transfer	AI/ML Monitoring	Regaining motor functions	Better mobility and reduced disability

Table 5 DBS methods

Conclusion

The development of generative AI applications in Azure AI Foundry represents a significant advancement in the field of deep brain stimulation. By leveraging AI/ML, LLM, and cybersecurity, researchers can enhance the diagnosis and treatment of brain diseases, offering new hope for patients suffering from neurological conditions.

Integration of these advanced techniques and technologies, we can develop more effective treatment strategies for neurological conditions, tailored to the specific needs of different patient groups. This holistic approach can significantly improve recovery outcomes and quality of life for both early-born infants and seniors.

Personal Observations

After losing excess weight, I noticed a reduction in the fatty tissue around the widow's hump, which developed over more than 20 years of working in a bent-over position with instrumentation, like for example PCS. While weight loss is generally healthy, in my case, I had to simultaneously stimulate the development of neck and shoulder girdle muscles to ensure that any spinal defects and reduced cervical lordosis were supported by muscles rather than fat. This helped prevent many pains and migraines with aura, which became increasingly bothersome during weight loss, as my spine was losing the “support” of fatty tissue.

Whenever I sensed an impending migraine attack, based on years of experience with migraines with aura, I would lie down in a dark, quiet place to avoid stimuli. I used special rehabilitation rollers to apply pressure on anatomical points connecting the main neural pathways around the shoulder girdle. Each time, the pressure would relax the shoulder girdle muscles, allowing me to either halt the progression of the migraine and prevent the development of the aura or minimize its occurrence.

The overall brain condition of my twins returned to normal operability after a series of Vojta Method treatments, combined with pharmacological and healing procedure adjustments, formally adopted at hospitals for years in recovery.

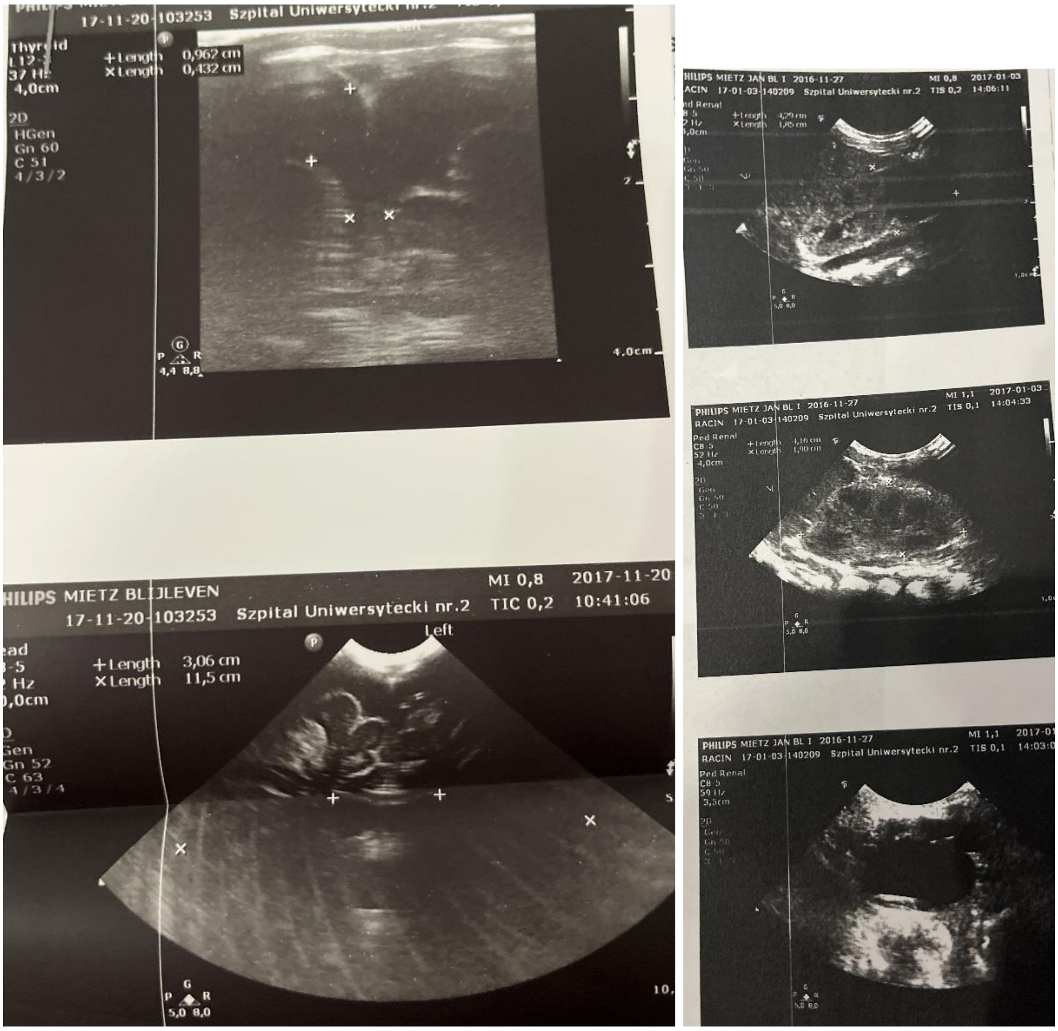


Figure 50 Healing process of brains of my twins – state in past during Vojta rehabilitation and pharmacological treatment and brain conductivity with charge transfer method.

I do not encourage neglecting pharmacology and medical treatment – their guidance should be considered a priority. My observations are intended to encourage collaboration on a professional and substantive level between the medical and IT worlds.

Recognizing Warning Signs

Individuals experiencing aura can reduce their risk of migraine-related stroke by being aware of stroke warning signs. According to Tietjen, a transient ischemic attack (TIA) can act as a "warning." Symptoms of TIAs may include vision loss in one eye, slurred speech, or weakness/numbness on one side of the body. These attacks typically last up to one hour.

If individuals notice changes in their aura—such as increased frequency, longer duration (lasting over an hour), or symptoms like weakness on one side—it may indicate a more serious issue, and they should seek medical attention. Tietjen emphasizes that people with migraine with aura need to recognize stroke warning signs and manage other stroke risk factors. Lifestyle changes and medications can help reduce the frequency of aura.

A migraine specialist can assist in identifying and treating stroke risk factors. Connect with a headache specialist in the area to learn more about managing the symptoms and understanding the condition.

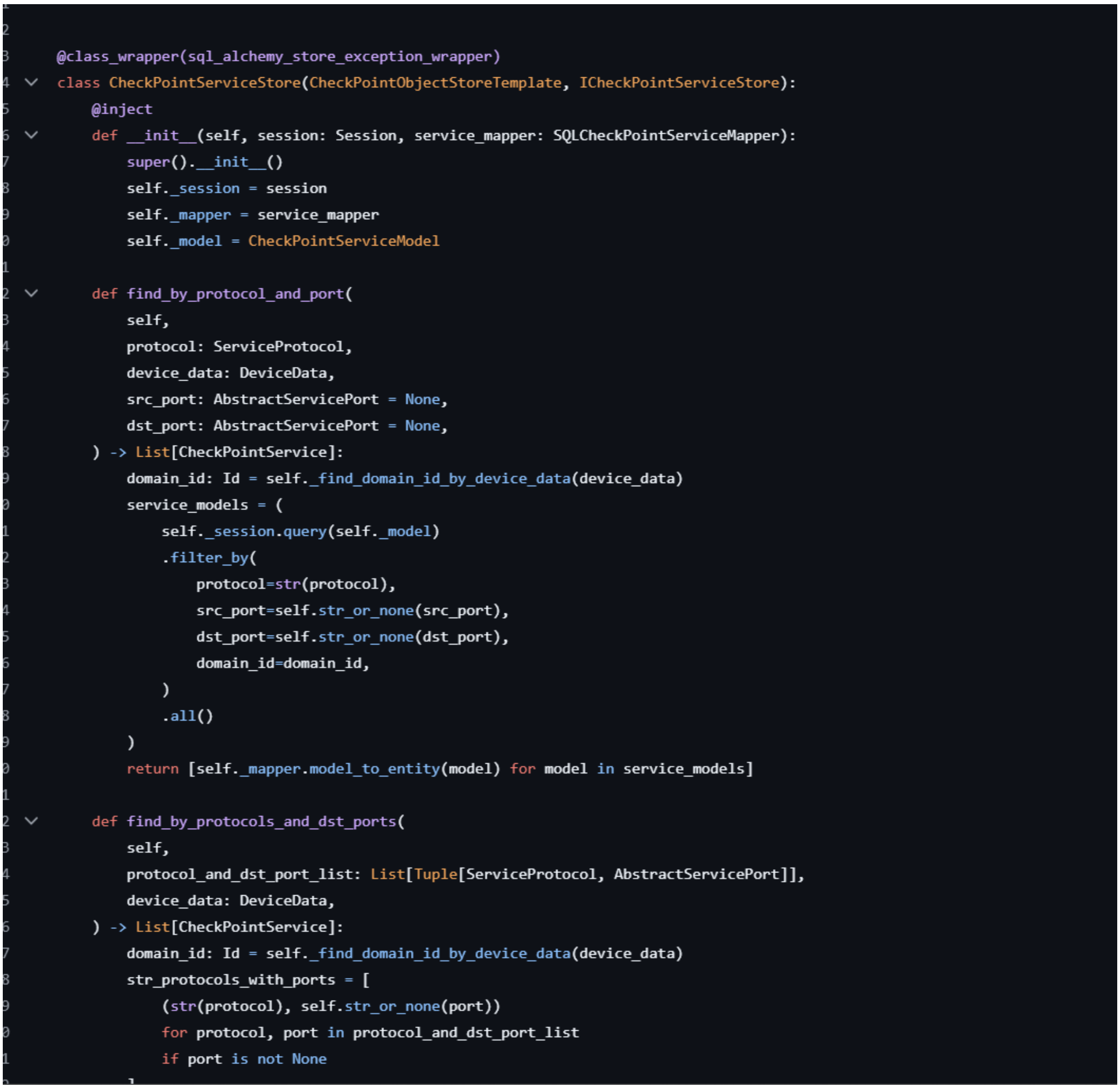
Chapter 6: Conclusion and Recommendations, summarizing the study and suggesting future research directions.

Potential for Advanced Applications

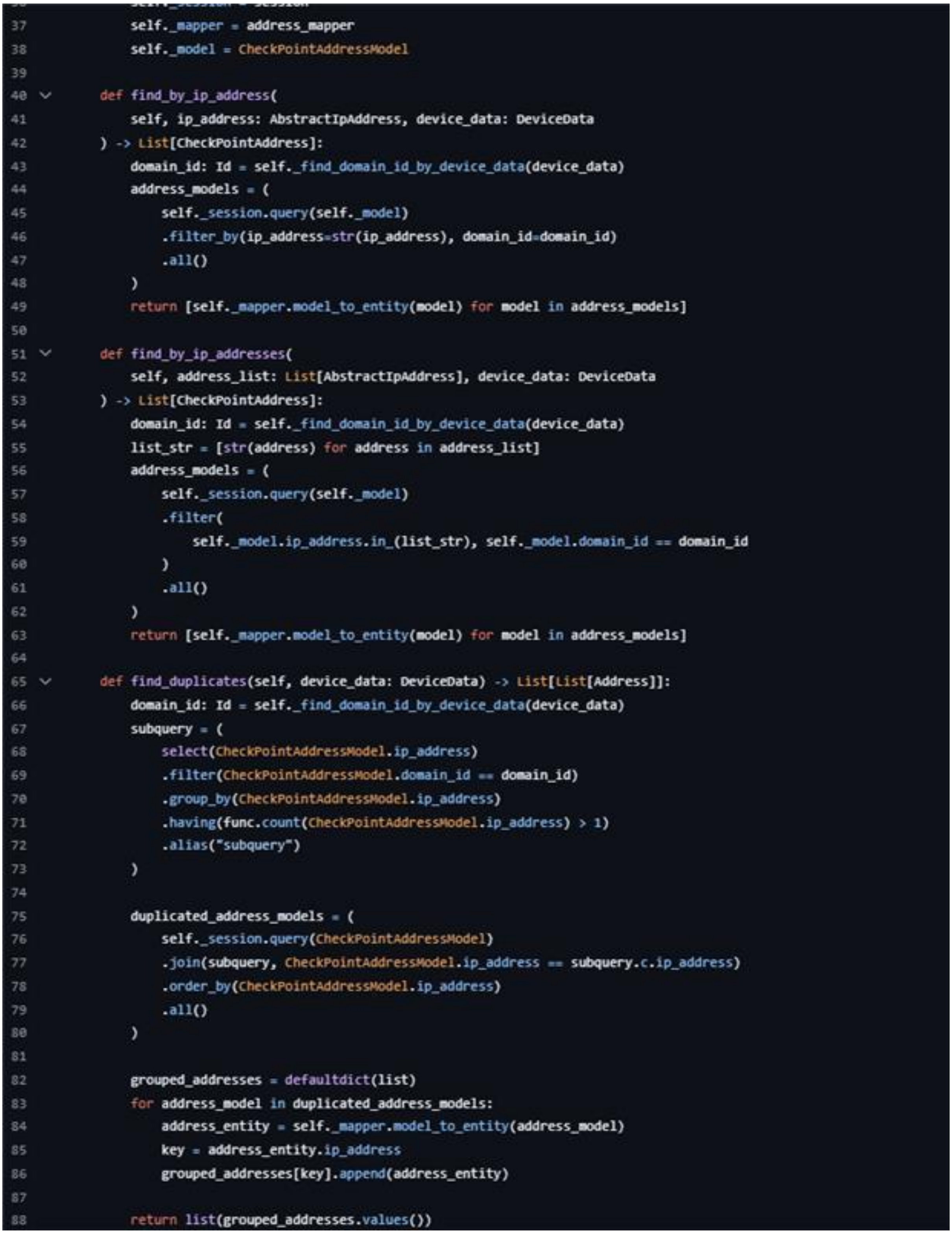
Graphene's unique properties open up possibilities for integrating advanced technologies like AI/ML and cybersecurity into DBS systems. This can lead to more personalized and secure treatment options.

Overall, graphene's properties make it a promising material for enhancing the effectiveness, safety, and precision of DBS therapies.

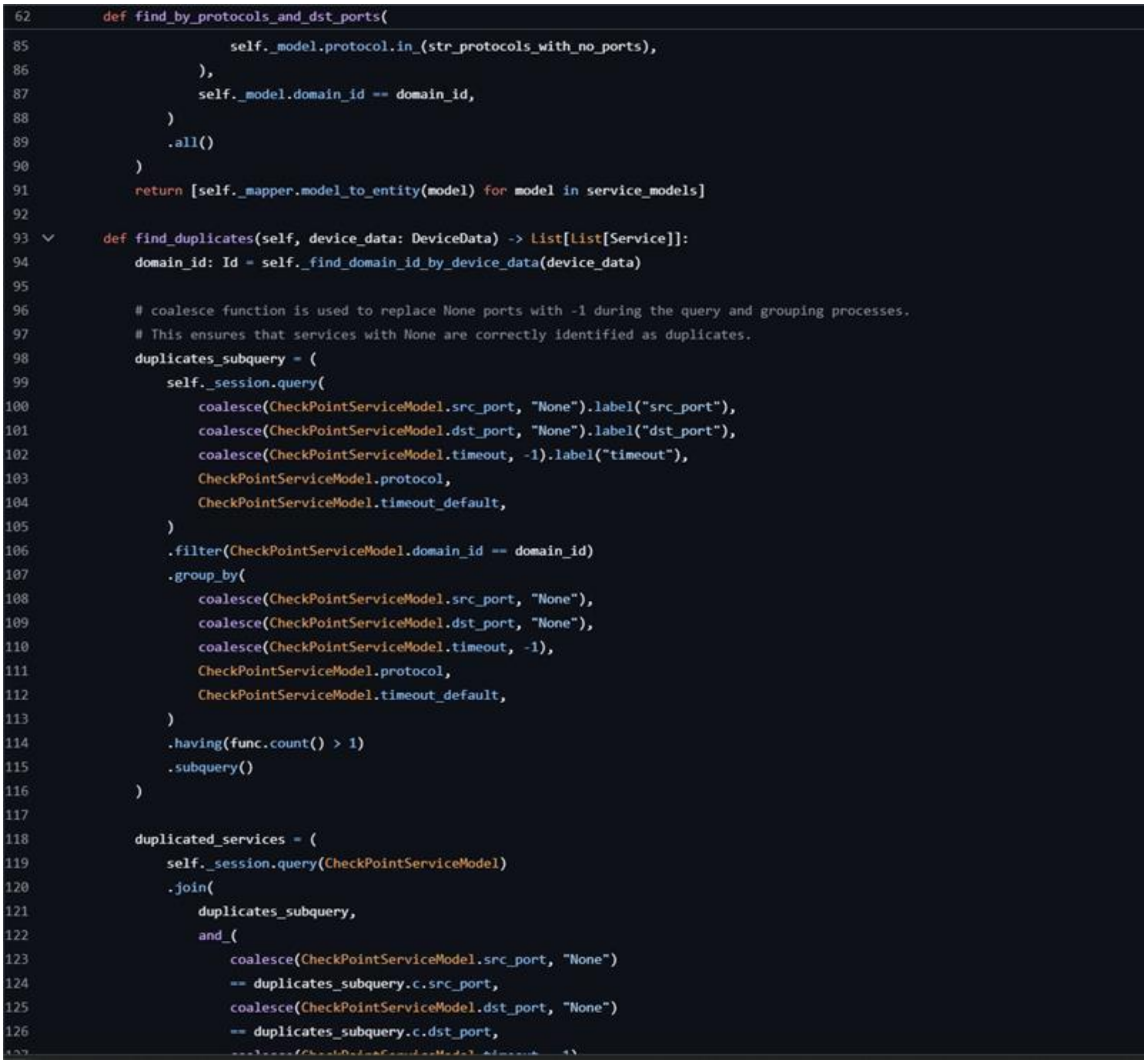
I have created the prototype of application that can transfer data received from the graphene diodes with analysis of brain parameters, blood pressure with use of integration with neural networks on external systems or devices to track those data for further machine learning and data science analysis in integration with other devices available for patients, doctors, any end users.



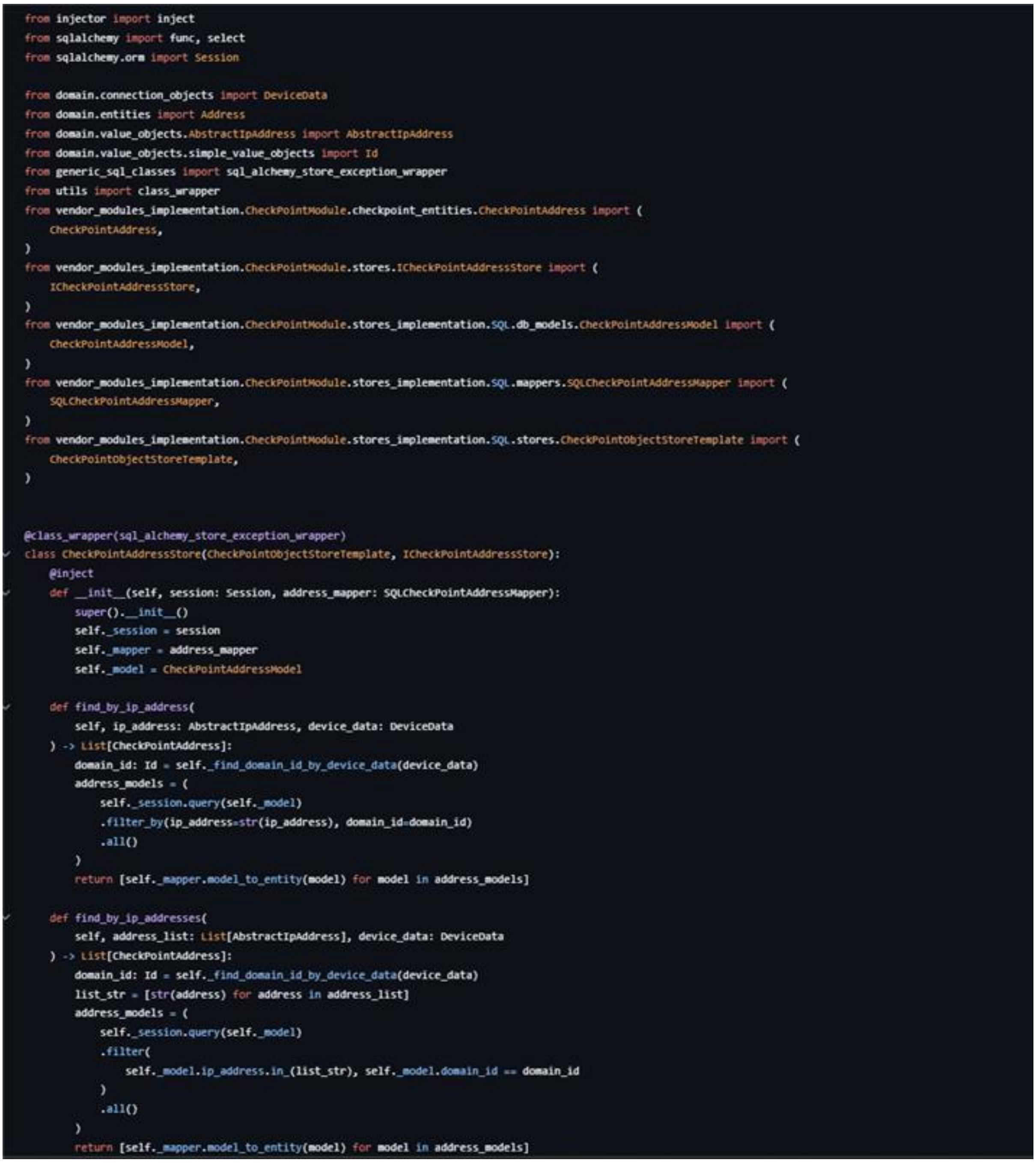
```python

@class_wrapper(sql_alchemy_store_exception_wrapper)
class CheckPointServiceStore(CheckPointObjectStoreTemplate, ICheckPointServiceStore):
    @inject
    def __init__(self, session: Session, service_mapper: SQLCheckPointServiceMapper):
        super().__init__()
        self._session = session
        self._mapper = service_mapper
        self._model = CheckPointServiceModel

    def find_by_protocol_and_port(
        self,
        protocol: ServiceProtocol,
        device_data: DeviceData,
        src_port: AbstractServicePort = None,
        dst_port: AbstractServicePort = None,
    ) -> List[CheckPointService]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        service_models = (
            self._session.query(self._model)
            .filter_by(
                protocol=str(protocol),
                src_port=self.str_or_none(src_port),
                dst_port=self.str_or_none(dst_port),
                domain_id=domain_id,
            )
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in service_models]

    def find_by_protocols_and_dst_ports(
        self,
        protocol_and_dst_port_list: List[Tuple[ServiceProtocol, AbstractServicePort]],
        device_data: DeviceData,
    ) -> List[CheckPointService]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        str_protocols_with_ports = [
            (str(protocol), self.str_or_none(port))
            for protocol, port in protocol_and_dst_port_list
            if port is not None
        ]
```



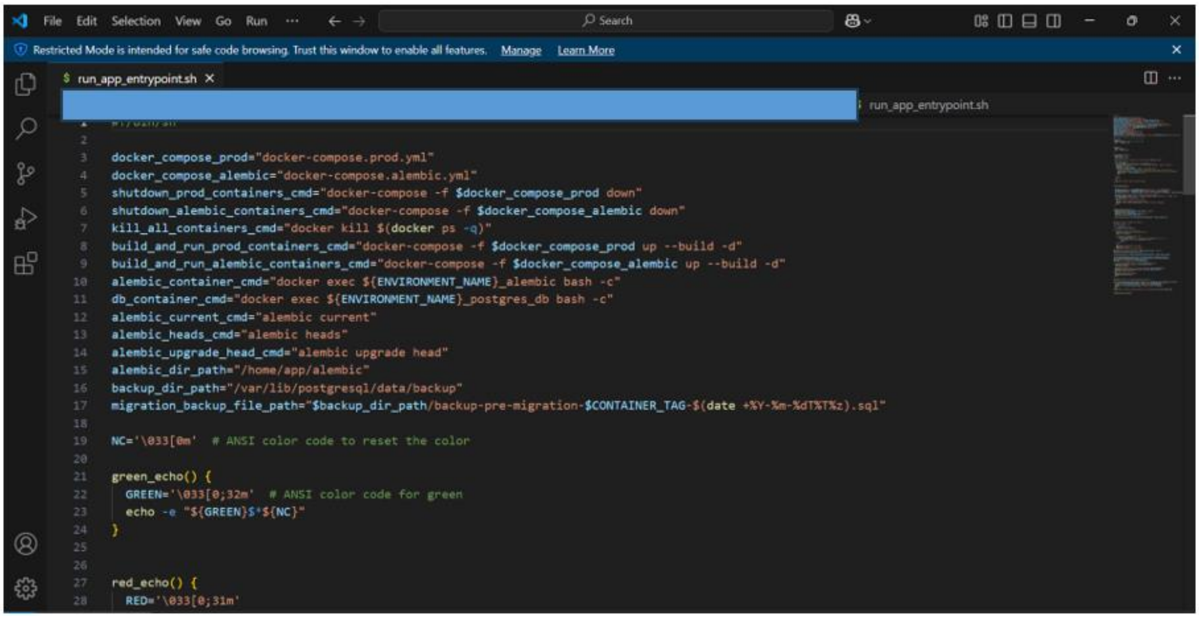
```
        self._session = session
        self._mapper = address_mapper
        self._model = CheckPointAddressModel

    def find_by_ip_address(
        self, ip_address: AbstractIpAddress, device_data: DeviceData
    ) -> List[CheckPointAddress]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        address_models = (
            self._session.query(self._model)
            .filter_by(ip_address=str(ip_address), domain_id=domain_id)
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in address_models]

    def find_by_ip_addresses(
        self, address_list: List[AbstractIpAddress], device_data: DeviceData
    ) -> List[CheckPointAddress]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        list_str = [str(address) for address in address_list]
        address_models = (
            self._session.query(self._model)
            .filter(
                self._model.ip_address.in_(list_str), self._model.domain_id == domain_id
            )
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in address_models]

    def find_duplicates(self, device_data: DeviceData) -> List[List[Address]]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        subquery = (
            select(CheckPointAddressModel.ip_address)
            .filter(CheckPointAddressModel.domain_id == domain_id)
            .group_by(CheckPointAddressModel.ip_address)
            .having(func.count(CheckPointAddressModel.ip_address) > 1)
            .alias("subquery")
        )

        duplicated_address_models = (
            self._session.query(CheckPointAddressModel)
            .join(subquery, CheckPointAddressModel.ip_address == subquery.c.ip_address)
            .order_by(CheckPointAddressModel.ip_address)
            .all()
        )

        grouped_addresses = defaultdict(list)
        for address_model in duplicated_address_models:
            address_entity = self._mapper.model_to_entity(address_model)
            key = address_entity.ip_address
            grouped_addresses[key].append(address_entity)

        return list(grouped_addresses.values())
```



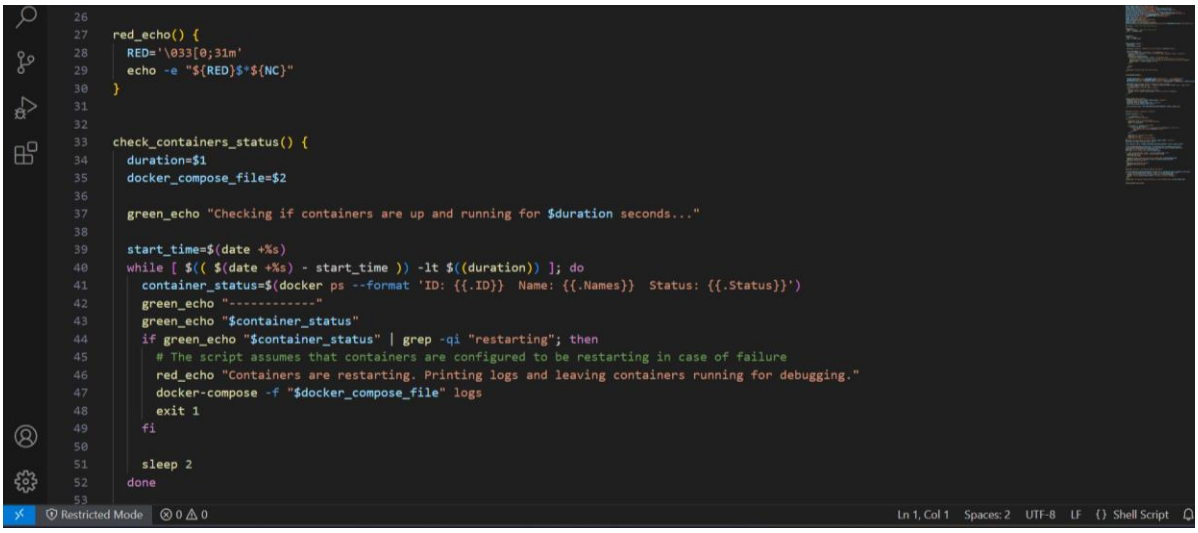
```python
    def find_by_protocols_and_dst_ports(
                    self._model.protocol.in_(str_protocols_with_no_ports),
                ),
                self._model.domain_id == domain_id,
            )
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in service_models]

    def find_duplicates(self, device_data: DeviceData) -> List[List[Service]]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)

        # coalesce function is used to replace None ports with -1 during the query and grouping processes.
        # This ensures that services with None are correctly identified as duplicates.
        duplicates_subquery = (
            self._session.query(
                coalesce(CheckPointServiceModel.src_port, "None").label("src_port"),
                coalesce(CheckPointServiceModel.dst_port, "None").label("dst_port"),
                coalesce(CheckPointServiceModel.timeout, -1).label("timeout"),
                CheckPointServiceModel.protocol,
                CheckPointServiceModel.timeout_default,
            )
            .filter(CheckPointServiceModel.domain_id == domain_id)
            .group_by(
                coalesce(CheckPointServiceModel.src_port, "None"),
                coalesce(CheckPointServiceModel.dst_port, "None"),
                coalesce(CheckPointServiceModel.timeout, -1),
                CheckPointServiceModel.protocol,
                CheckPointServiceModel.timeout_default,
            )
            .having(func.count() > 1)
            .subquery()
        )

        duplicated_services = (
            self._session.query(CheckPointServiceModel)
            .join(
                duplicates_subquery,
                and_(
                    coalesce(CheckPointServiceModel.src_port, "None")
                    == duplicates_subquery.c.src_port,
                    coalesce(CheckPointServiceModel.dst_port, "None")
                    == duplicates_subquery.c.dst_port,
                    coalesce(CheckPointServiceModel.timeout, -1)
```



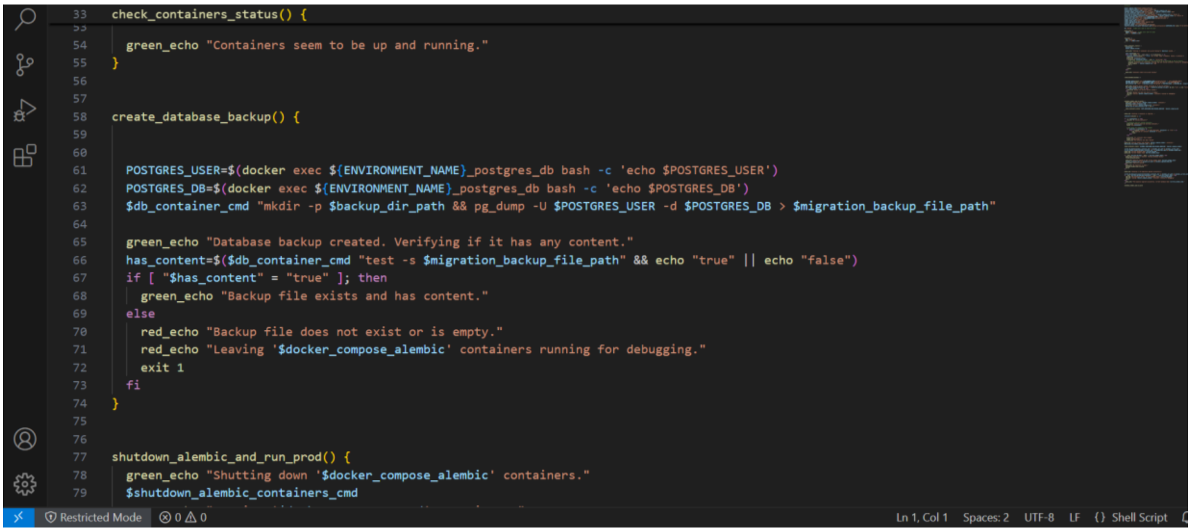
```python
from injector import inject
from sqlalchemy import func, select
from sqlalchemy.orm import Session

from domain.connection_objects import DeviceData
from domain.entities import Address
from domain.value_objects.AbstractIpAddress import AbstractIpAddress
from domain.value_objects.simple_value_objects import Id
from generic_sql_classes import sql_alchemy_store_exception_wrapper
from utils import class_wrapper
from vendor_modules_implementation.CheckPointModule.checkpoint_entities.CheckPointAddress import (
    CheckPointAddress,
)
from vendor_modules_implementation.CheckPointModule.stores.ICheckPointAddressStore import (
    ICheckPointAddressStore,
)
from vendor_modules_implementation.CheckPointModule.stores_implementation.SQL.db_models.CheckPointAddressModel import (
    CheckPointAddressModel,
)
from vendor_modules_implementation.CheckPointModule.stores_implementation.SQL.mappers.SQLCheckPointAddressMapper import (
    SQLCheckPointAddressMapper,
)
from vendor_modules_implementation.CheckPointModule.stores_implementation.SQL.stores.CheckPointObjectStoreTemplate import (
    CheckPointObjectStoreTemplate,
)


@class_wrapper(sql_alchemy_store_exception_wrapper)
class CheckPointAddressStore(CheckPointObjectStoreTemplate, ICheckPointAddressStore):
    @inject
    def __init__(self, session: Session, address_mapper: SQLCheckPointAddressMapper):
        super().__init__()
        self._session = session
        self._mapper = address_mapper
        self._model = CheckPointAddressModel

    def find_by_ip_address(
        self, ip_address: AbstractIpAddress, device_data: DeviceData
    ) -> List[CheckPointAddress]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        address_models = (
            self._session.query(self._model)
            .filter_by(ip_address=str(ip_address), domain_id=domain_id)
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in address_models]

    def find_by_ip_addresses(
        self, address_list: List[AbstractIpAddress], device_data: DeviceData
    ) -> List[CheckPointAddress]:
        domain_id: Id = self._find_domain_id_by_device_data(device_data)
        list_str = [str(address) for address in address_list]
        address_models = (
            self._session.query(self._model)
            .filter(
                self._model.ip_address.in_(list_str), self._model.domain_id == domain_id
            )
            .all()
        )
        return [self._mapper.model_to_entity(model) for model in address_models]
```


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
run_app_entrypoint.sh
run_app_entrypoint.sh

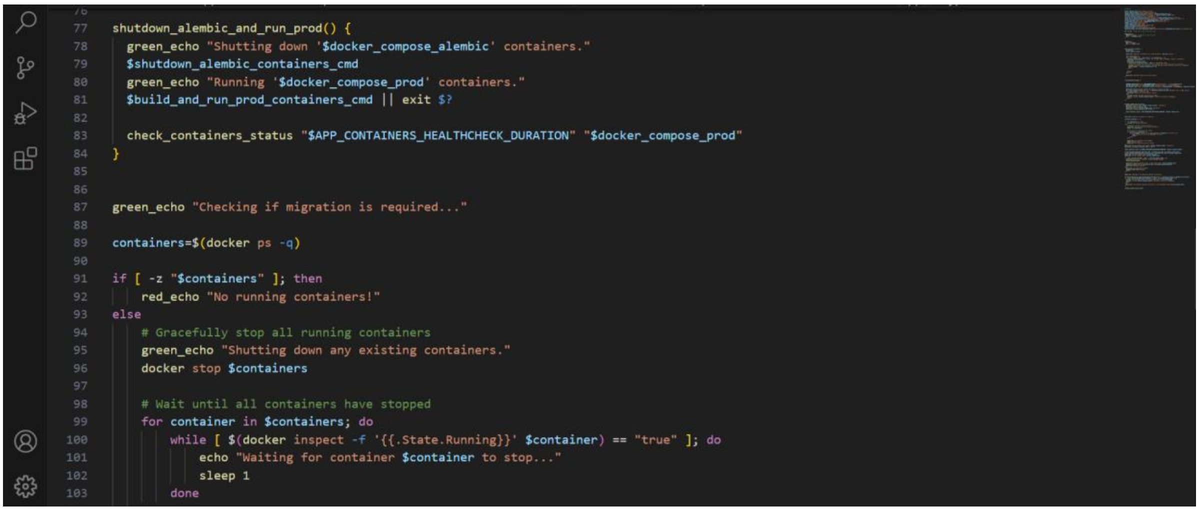
```
#!/bin/sh

docker_compose_prod="docker-compose.prod.yml"
docker_compose_alembic="docker-compose.alembic.yml"
shutdown_prod_containers_cmd="docker-compose -f $docker_compose_prod down"
shutdown_alembic_containers_cmd="docker-compose -f $docker_compose_alembic down"
kill_all_containers_cmd="docker kill $(docker ps -q)"
build_and_run_prod_containers_cmd="docker-compose -f $docker_compose_prod up --build -d"
build_and_run_alembic_containers_cmd="docker-compose -f $docker_compose_alembic up --build -d"
alembic_container_cmd="docker exec ${ENVIRONMENT_NAME}_alembic bash -c"
db_container_cmd="docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c"
alembic_current_cmd="alembic current"
alembic_heads_cmd="alembic heads"
alembic_upgrade_head_cmd="alembic upgrade head"
alembic_dir_path="/home/app/alembic"
backup_dir_path="/var/lib/postgresql/data/backup"
migration_backup_file_path="$backup_dir_path/backup-pre-migration-$CONTAINER_TAG-$(date +%Y-%m-%dT%T%z).sql"

NC='\033[0m'  # ANSI color code to reset the color

green_echo() {
  GREEN='\033[0;32m'  # ANSI color code for green
  echo -e "${GREEN}$*${NC}"
}


red_echo() {
  RED='\033[0;31m'
```



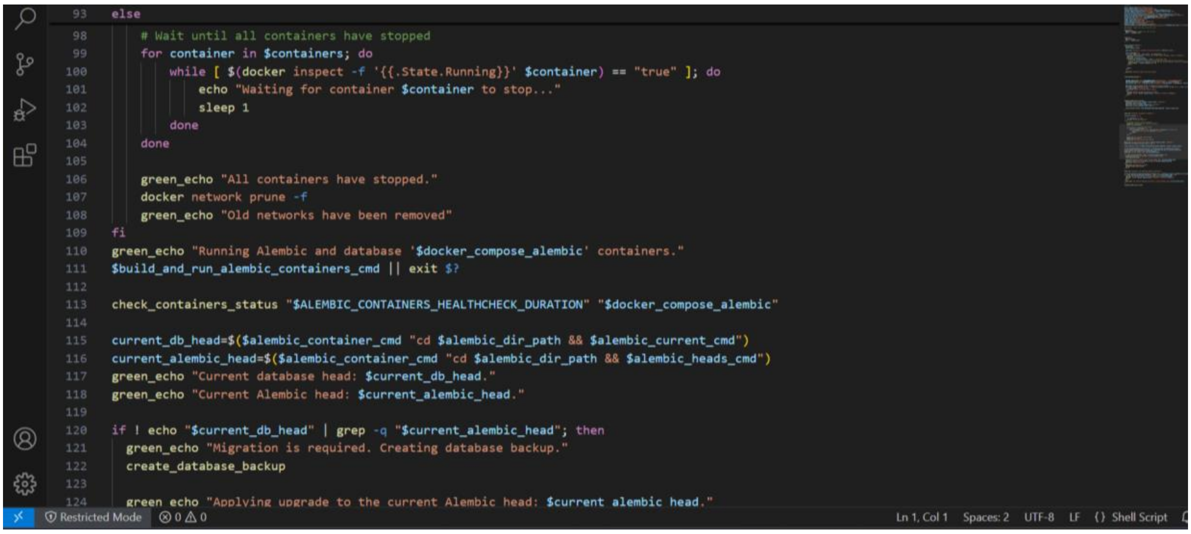
```

red_echo() {
  RED='\033[0;31m'
  echo -e "${RED}$*${NC}"
}


check_containers_status() {
  duration=$1
  docker_compose_file=$2

  green_echo "Checking if containers are up and running for $duration seconds..."

  start_time=$(date +%s)
  while [ $(( $(date +%s) - start_time )) -lt $((duration)) ]; do
    container_status=$(docker ps --format 'ID: {{.ID}}  Name: {{.Names}}  Status: {{.Status}}')
    green_echo "------------"
    green_echo "$container_status"
    if green_echo "$container_status" | grep -qi "restarting"; then
      # The script assumes that containers are configured to be restarting in case of failure
      red_echo "Containers are restarting. Printing logs and leaving containers running for debugging."
      docker-compose -f "$docker_compose_file" logs
      exit 1
    fi

    sleep 2
  done

```



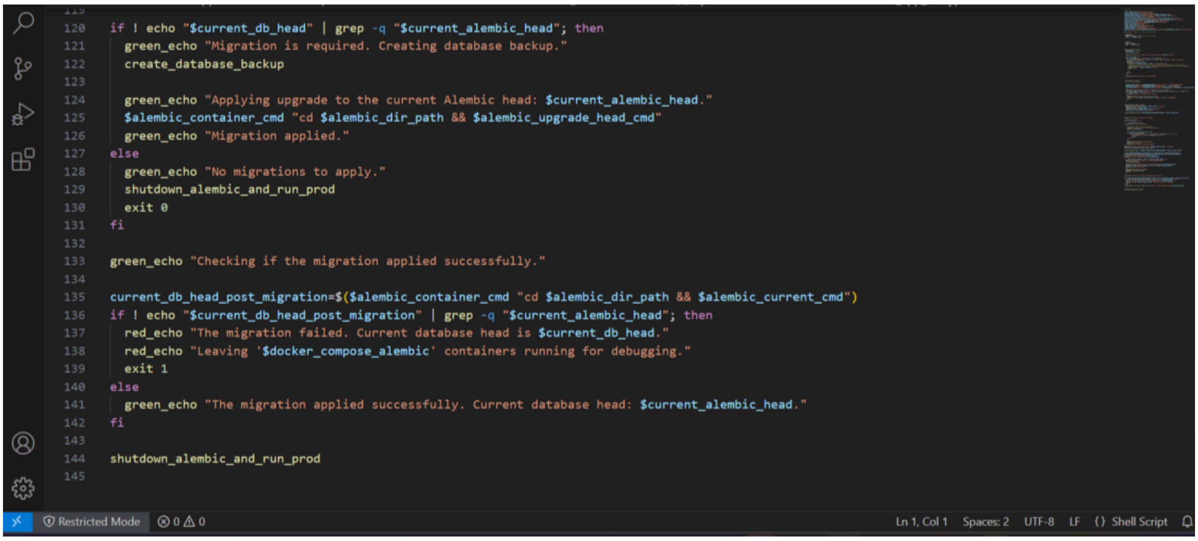
```
check_containers_status() {

  green_echo "Containers seem to be up and running."
}


create_database_backup() {


  POSTGRES_USER=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_USER')
  POSTGRES_DB=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_DB')
  $db_container_cmd "mkdir -p $backup_dir_path && pg_dump -U $POSTGRES_USER -d $POSTGRES_DB > $migration_backup_file_path"

  green_echo "Database backup created. Verifying if it has any content."
  has_content=$($db_container_cmd "test -s $migration_backup_file_path" && echo "true" || echo "false")
  if [ "$has_content" = "true" ]; then
    green_echo "Backup file exists and has content."
  else
    red_echo "Backup file does not exist or is empty."
    red_echo "Leaving '$docker_compose_alembic' containers running for debugging."
    exit 1
  fi
}


shutdown_alembic_and_run_prod() {
  green_echo "Shutting down '$docker_compose_alembic' containers."
  $shutdown_alembic_containers_cmd
```



```
shutdown_alembic_and_run_prod() {
  green_echo "Shutting down '$docker_compose_alembic' containers."
  $shutdown_alembic_containers_cmd
  green_echo "Running '$docker_compose_prod' containers."
  $build_and_run_prod_containers_cmd || exit $?

  check_containers_status "$APP_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_prod"
}


green_echo "Checking if migration is required..."

containers=$(docker ps -q)

if [ -z "$containers" ]; then
    red_echo "No running containers!"
else
    # Gracefully stop all running containers
    green_echo "Shutting down any existing containers."
    docker stop $containers

    # Wait until all containers have stopped
    for container in $containers; do
        while [ $(docker inspect -f '{{.State.Running}}' $container) == "true" ]; do
            echo "Waiting for container $container to stop..."
            sleep 1
        done
```



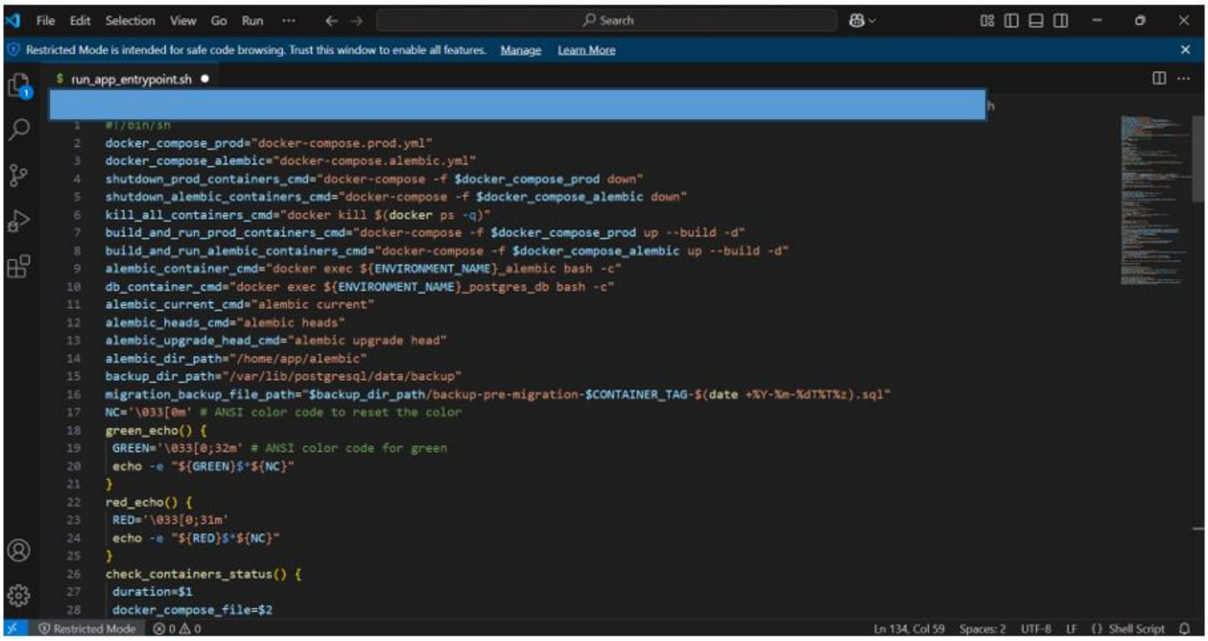
```
else
    # Wait until all containers have stopped
    for container in $containers; do
      while [ $(docker inspect -f '{{.State.Running}}' $container) == "true" ]; do
        echo "Waiting for container $container to stop..."
        sleep 1
      done
    done

    green_echo "All containers have stopped."
    docker network prune -f
    green_echo "Old networks have been removed"
fi
green_echo "Running Alembic and database '$docker_compose_alembic' containers."
$build_and_run_alembic_containers_cmd || exit $?

check_containers_status "$ALEMBIC_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_alembic"

current_db_head=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_current_cmd")
current_alembic_head=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_heads_cmd")
green_echo "Current database head: $current_db_head."
green_echo "Current Alembic head: $current_alembic_head."

if ! echo "$current_db_head" | grep -q "$current_alembic_head"; then
  green_echo "Migration is required. Creating database backup."
  create_database_backup

  green_echo "Applying upgrade to the current Alembic head: $current_alembic_head."
```



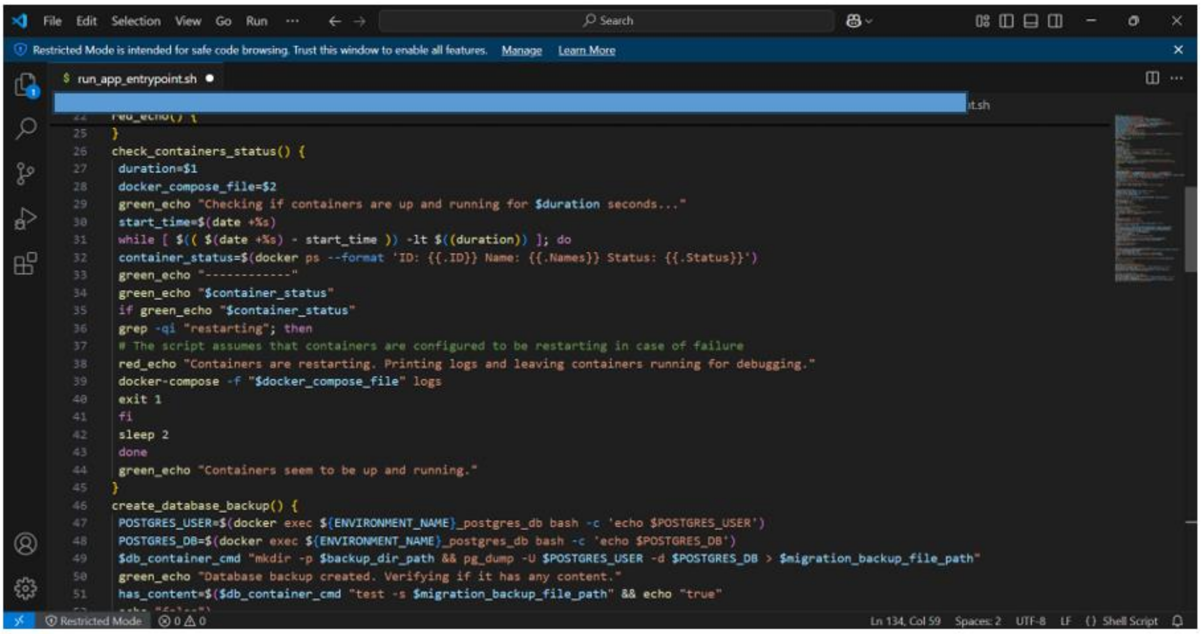
```
if ! echo "$current_db_head" | grep -q "$current_alembic_head"; then
  green_echo "Migration is required. Creating database backup."
  create_database_backup

  green_echo "Applying upgrade to the current Alembic head: $current_alembic_head."
  $alembic_container_cmd "cd $alembic_dir_path && $alembic_upgrade_head_cmd"
  green_echo "Migration applied."
else
  green_echo "No migrations to apply."
  shutdown_alembic_and_run_prod
  exit 0
fi

green_echo "Checking if the migration applied successfully."

current_db_head_post_migration=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_current_cmd")
if ! echo "$current_db_head_post_migration" | grep -q "$current_alembic_head"; then
  red_echo "The migration failed. Current database head is $current_db_head."
  red_echo "Leaving '$docker_compose_alembic' containers running for debugging."
  exit 1
else
  green_echo "The migration applied successfully. Current database head: $current_alembic_head."
fi

shutdown_alembic_and_run_prod
```

The run_app_entrypoint.sh script is designed to manage Docker containers, including starting, stopping, and monitoring production containers and Alembic containers used for database migrations. The script also includes functions for creating database backups and checking the status of containers.

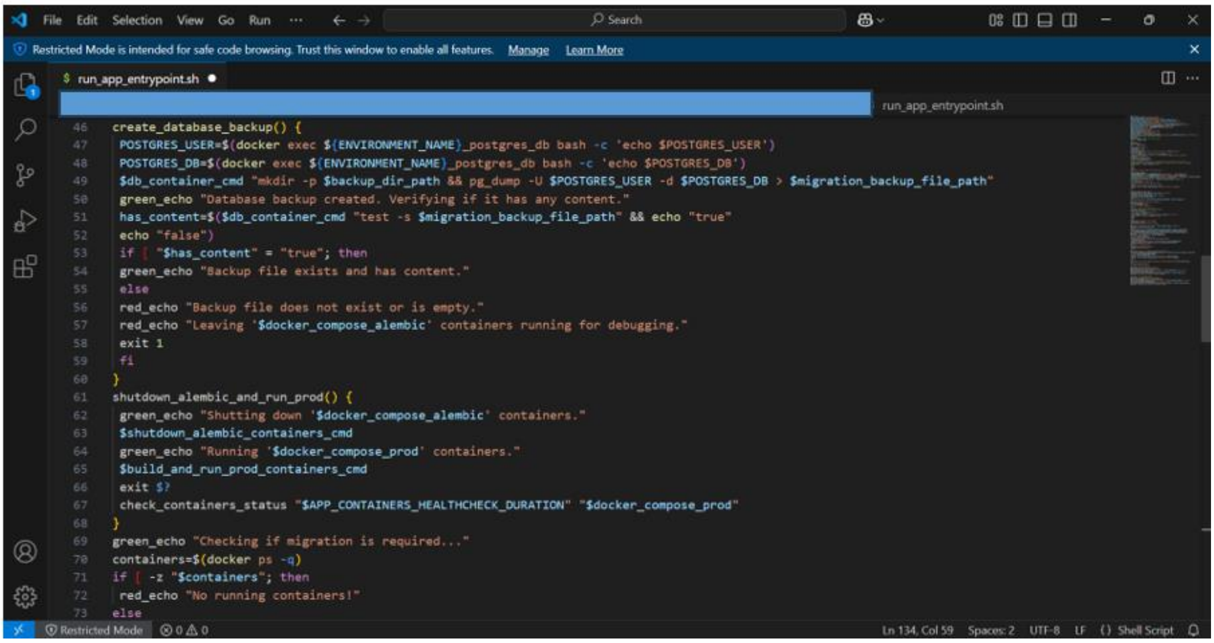
I have modified above script, including the integration of AI/ML models and graphene diodes:


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
$ run_app_entrypoint.sh ●
[redacted] h
Restricted Mode ⊗ 0 ⚠ 0
Ln 134, Col 59 Spaces: 2 UTF-8 LF {} Shell Script

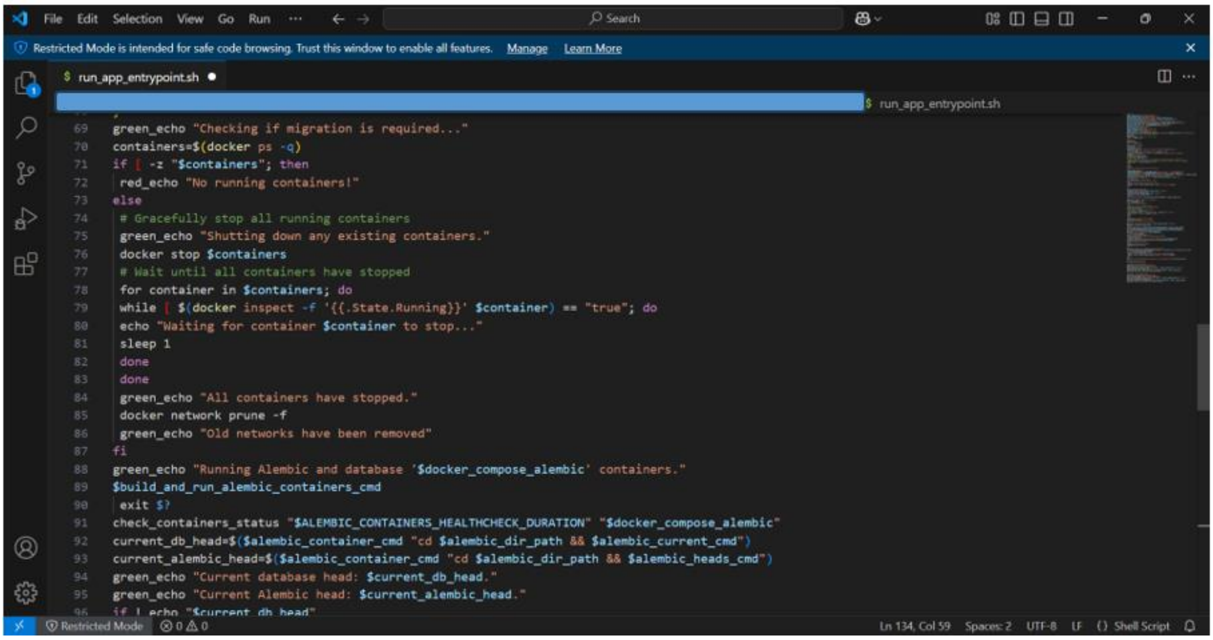
```sh
#!/bin/sh
docker_compose_prod="docker-compose.prod.yml"
docker_compose_alembic="docker-compose.alembic.yml"
shutdown_prod_containers_cmd="docker-compose -f $docker_compose_prod down"
shutdown_alembic_containers_cmd="docker-compose -f $docker_compose_alembic down"
kill_all_containers_cmd="docker kill $(docker ps -q)"
build_and_run_prod_containers_cmd="docker-compose -f $docker_compose_prod up --build -d"
build_and_run_alembic_containers_cmd="docker-compose -f $docker_compose_alembic up --build -d"
alembic_container_cmd="docker exec ${ENVIRONMENT_NAME}_alembic bash -c"
db_container_cmd="docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c"
alembic_current_cmd="alembic current"
alembic_heads_cmd="alembic heads"
alembic_upgrade_head_cmd="alembic upgrade head"
alembic_dir_path="/home/app/alembic"
backup_dir_path="/var/lib/postgresql/data/backup"
migration_backup_file_path="$backup_dir_path/backup-pre-migration-$CONTAINER_TAG-$(date +%Y-%m-%dT%T%z).sql"
NC='\033[0m' # ANSI color code to reset the color
green_echo() {
  GREEN='\033[0;32m' # ANSI color code for green
  echo -e "${GREEN}$*${NC}"
}
red_echo() {
  RED='\033[0;31m'
  echo -e "${RED}$*${NC}"
}
check_containers_status() {
  duration=$1
  docker_compose_file=$2
```


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
$ run_app_entrypoint.sh ●
[redacted] t.sh
Restricted Mode ⊗ 0 ⚠ 0
Ln 134, Col 59 Spaces: 2 UTF-8 LF {} Shell Script

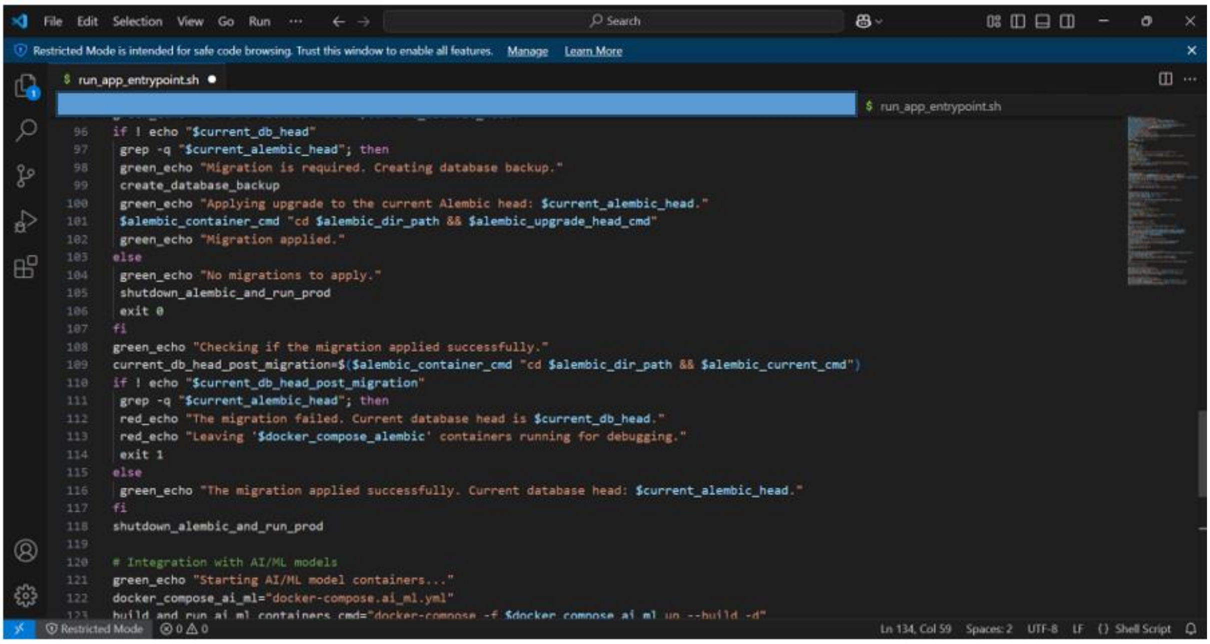
```sh
red_echo() {
}
check_containers_status() {
  duration=$1
  docker_compose_file=$2
  green_echo "Checking if containers are up and running for $duration seconds..."
  start_time=$(date +%s)
  while [ $(( $(date +%s) - start_time )) -lt $((duration)) ]; do
  container_status=$(docker ps --format 'ID: {{.ID}} Name: {{.Names}} Status: {{.Status}}')
  green_echo "-----------"
  green_echo "$container_status"
  if green_echo "$container_status"
  grep -qi "restarting"; then
  # The script assumes that containers are configured to be restarting in case of failure
  red_echo "Containers are restarting. Printing logs and leaving containers running for debugging."
  docker-compose -f "$docker_compose_file" logs
  exit 1
  fi
  sleep 2
  done
  green_echo "Containers seem to be up and running."
}
create_database_backup() {
  POSTGRES_USER=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_USER')
  POSTGRES_DB=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_DB')
  $db_container_cmd "mkdir -p $backup_dir_path && pg_dump -U $POSTGRES_USER -d $POSTGRES_DB > $migration_backup_file_path"
  green_echo "Database backup created. Verifying if it has any content."
  has_content=$($db_container_cmd "test -s $migration_backup_file_path" && echo "true"
  echo "false")
```


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
run_app_entrypoint.sh
Restricted Mode ⊗ 0 ⚠ 0
Ln 134, Col 59 Spaces: 2 UTF-8 LF {} Shell Script

```
create_database_backup() {
 POSTGRES_USER=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_USER')
 POSTGRES_DB=$(docker exec ${ENVIRONMENT_NAME}_postgres_db bash -c 'echo $POSTGRES_DB')
 $db_container_cmd "mkdir -p $backup_dir_path && pg_dump -U $POSTGRES_USER -d $POSTGRES_DB > $migration_backup_file_path"
 green_echo "Database backup created. Verifying if it has any content."
 has_content=$($db_container_cmd "test -s $migration_backup_file_path" && echo "true"
 echo "false")
 if [ "$has_content" = "true"; then
 green_echo "Backup file exists and has content."
 else
 red_echo "Backup file does not exist or is empty."
 red_echo "Leaving '$docker_compose_alembic' containers running for debugging."
 exit 1
 fi
}
shutdown_alembic_and_run_prod() {
 green_echo "Shutting down '$docker_compose_alembic' containers."
 $shutdown_alembic_containers_cmd
 green_echo "Running '$docker_compose_prod' containers."
 $build_and_run_prod_containers_cmd
 exit $?
 check_containers_status "$APP_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_prod"
}
green_echo "Checking if migration is required..."
containers=$(docker ps -q)
if [ -z "$containers"; then
 red_echo "No running containers!"
else
```

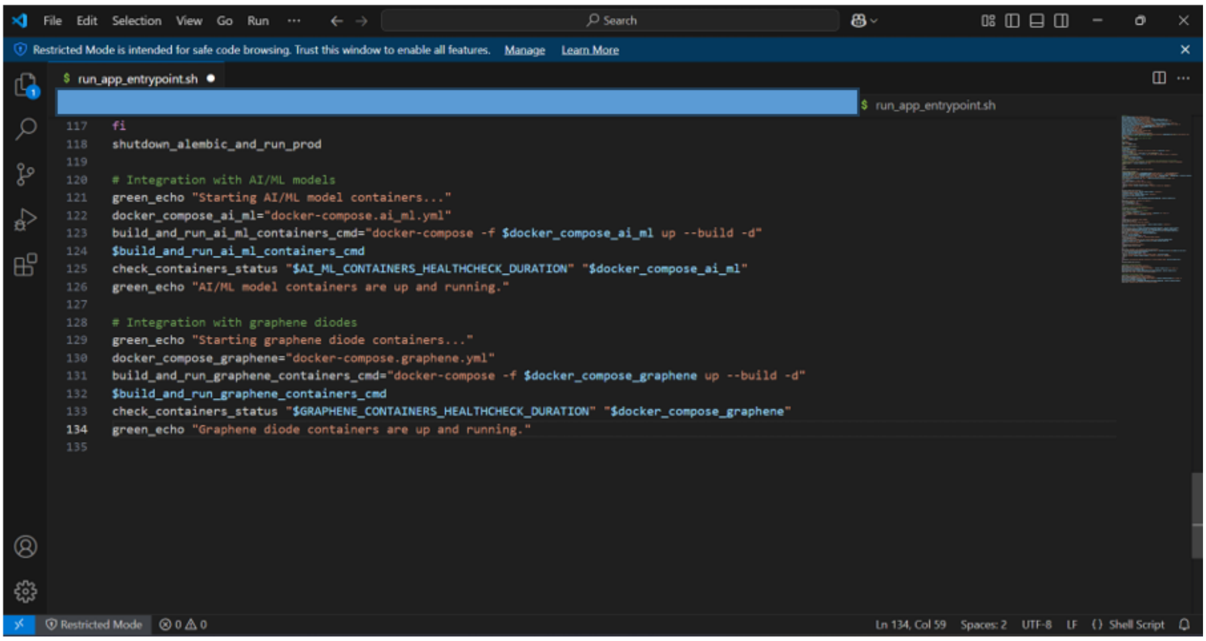

File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
run_app_entrypoint.sh
run_app_entrypoint.sh
Restricted Mode ⊗ 0 ⚠ 0
Ln 134, Col 59 Spaces: 2 UTF-8 LF {} Shell Script

```
green_echo "Checking if migration is required..."
containers=$(docker ps -q)
if [ -z "$containers"; then
 red_echo "No running containers!"
else
 # Gracefully stop all running containers
 green_echo "Shutting down any existing containers."
 docker stop $containers
 # Wait until all containers have stopped
 for container in $containers; do
 while [ $(docker inspect -f '{{.State.Running}}' $container) == "true"; do
 echo "Waiting for container $container to stop..."
 sleep 1
 done
 done
 green_echo "All containers have stopped."
 docker network prune -f
 green_echo "Old networks have been removed"
fi
green_echo "Running Alembic and database '$docker_compose_alembic' containers."
$build_and_run_alembic_containers_cmd
 exit $?
check_containers_status "$ALEMBIC_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_alembic"
current_db_head=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_current_cmd")
current_alembic_head=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_heads_cmd")
green_echo "Current database head: $current_db_head."
green_echo "Current Alembic head: $current_alembic_head."
if ! echo "$current_db_head"
```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```
if ! echo "$current_db_head"
  grep -q "$current_alembic_head"; then
  green_echo "Migration is required. Creating database backup."
  create_database_backup
  green_echo "Applying upgrade to the current Alembic head: $current_alembic_head."
  $alembic_container_cmd "cd $alembic_dir_path && $alembic_upgrade_head_cmd"
  green_echo "Migration applied."
else
  green_echo "No migrations to apply."
  shutdown_alembic_and_run_prod
  exit 0
fi
green_echo "Checking if the migration applied successfully."
current_db_head_post_migration=$($alembic_container_cmd "cd $alembic_dir_path && $alembic_current_cmd")
if ! echo "$current_db_head_post_migration"
  grep -q "$current_alembic_head"; then
  red_echo "The migration failed. Current database head is $current_db_head."
  red_echo "Leaving '$docker_compose_alembic' containers running for debugging."
  exit 1
else
  green_echo "The migration applied successfully. Current database head: $current_alembic_head."
fi
shutdown_alembic_and_run_prod

# Integration with AI/ML models
green_echo "Starting AI/ML model containers..."
docker_compose_ai_ml="docker-compose.ai_ml.yml"
build_and_run_ai_ml_containers_cmd="docker-compose -f $docker_compose_ai_ml up --build -d"
```



Restricted Mode is intended for safe code browsing. Trust this window to enable all features.

```
fi
shutdown_alembic_and_run_prod

# Integration with AI/ML models
green_echo "Starting AI/ML model containers..."
docker_compose_ai_ml="docker-compose.ai_ml.yml"
build_and_run_ai_ml_containers_cmd="docker-compose -f $docker_compose_ai_ml up --build -d"
$build_and_run_ai_ml_containers_cmd
check_containers_status "$AI_ML_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_ai_ml"
green_echo "AI/ML model containers are up and running."

# Integration with graphene diodes
green_echo "Starting graphene diode containers..."
docker_compose_graphene="docker-compose.graphene.yml"
build_and_run_graphene_containers_cmd="docker-compose -f $docker_compose_graphene up --build -d"
$build_and_run_graphene_containers_cmd
check_containers_status "$GRAPHENE_CONTAINERS_HEALTHCHECK_DURATION" "$docker_compose_graphene"
green_echo "Graphene diode containers are up and running."

```

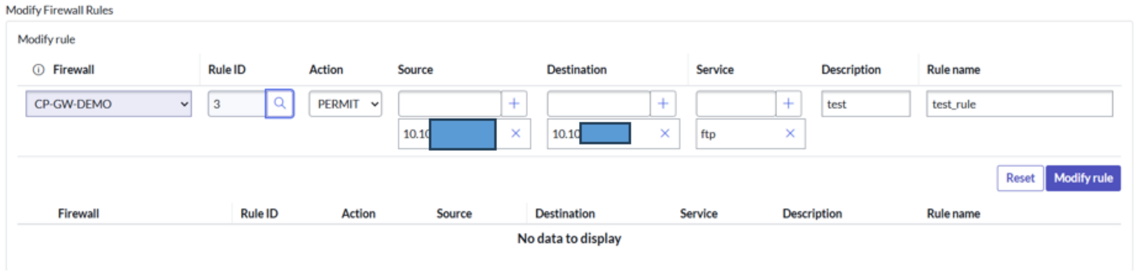
Figure 51 Application developed to integrate data collection from graphene diodes with endpoint management (for example with firewall, MS Endpoint Manager or Configuration Manager of Microsoft, or AirWatch for data protection, routing, collection, storage, transfer, loading mechanism.

In this solution, I have added sections for starting AI/ML model containers and graphene diode containers. The AI/ML model containers are managed using a new Docker Compose file (docker-compose.ai_ml.yml), and the graphene diode containers are managed using another Docker Compose file (docker-compose.graphene.yml). The script builds and runs these containers and checks their status to ensure they are up and running.

I can customize the Docker Compose files (docker-compose.ai_ml.yml and docker-compose.graphene.yml) to include the specific configurations and dependencies needed for my AI/ML models and graphene diodes to adjust for brain strokes and migrene with aura diagnosis.

While not sharing with data of my children and my own, I have introduced the system that can be used by any medical center, doctors for taking decisions on patient treatments.

Implementing firewall rules in neural networks and Deep Brain Stimulation (DBS) systems after collecting data from graphene diodes involves several steps to ensure security and efficient data processing.



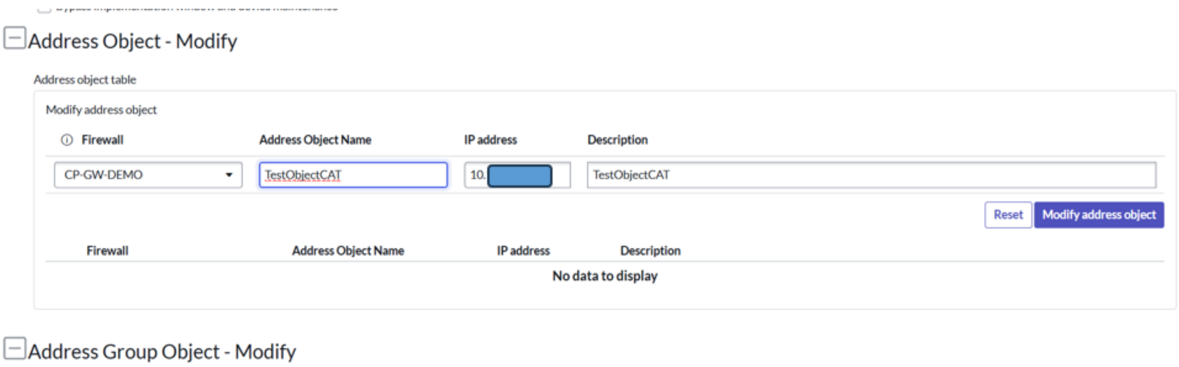
Modify Firewall Rules

Modify rule

Firewall	Rule ID	Action	Source	Destination	Service	Description	Rule name
CP-GW-DEMO	3	PERMIT	10.1[redacted]	10.1[redacted]	ftp	test	test_rule

Firewall	Rule ID	Action	Source	Destination	Service	Description	Rule name

No data to display



Address Object - Modify

Address object table

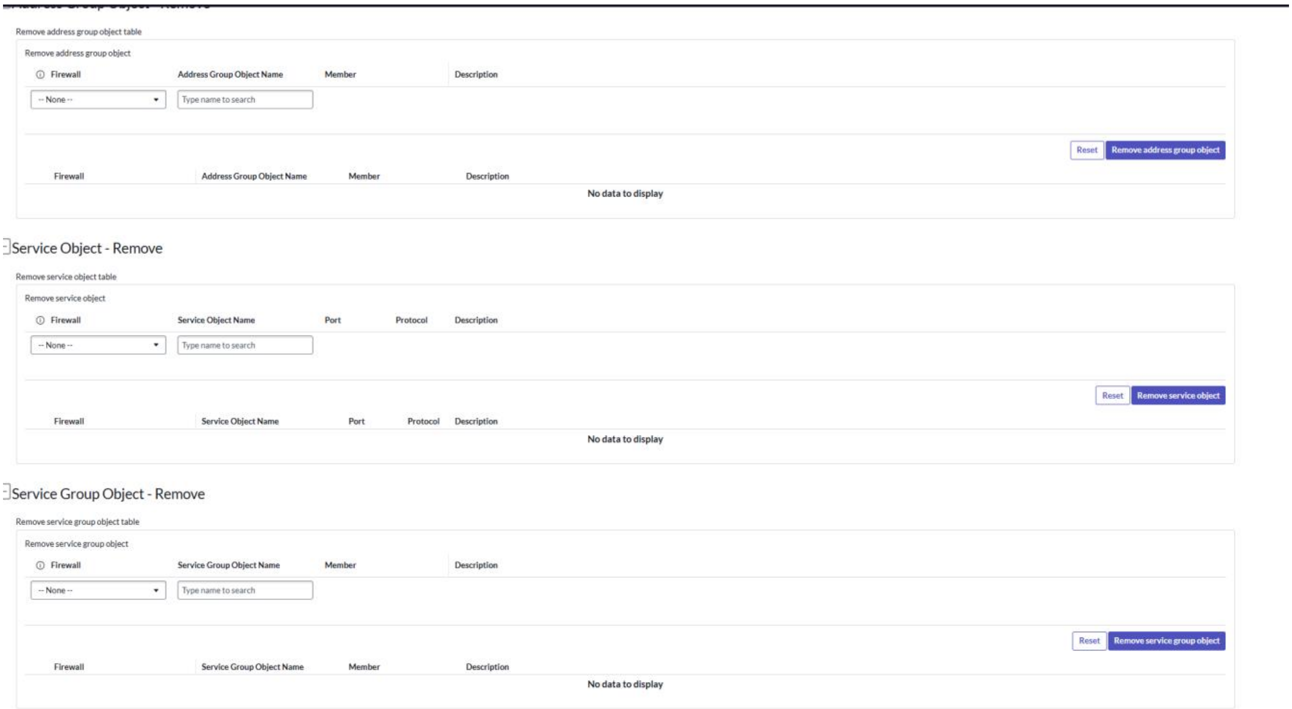
Modify address object

Firewall	Address Object Name	IP address	Description
CP-GW-DEMO	TestObjectCAT	10.[redacted]	TestObjectCAT

Firewall	Address Object Name	IP address	Description

No data to display

Address Group Object - Modify



Remove address group object table

Remove address group object

ⓘ Firewall | Address Group Object Name | Member | Description

-- None -- ▾ | Type name to search

Reset | Remove address group object

Firewall | Address Group Object Name | Member | Description

No data to display

Service Object - Remove

Remove service object table

Remove service object

ⓘ Firewall | Service Object Name | Port | Protocol | Description

-- None -- ▾ | Type name to search

Reset | Remove service object

Firewall | Service Object Name | Port | Protocol | Description

No data to display

Service Group Object - Remove

Remove service group object table

Remove service group object

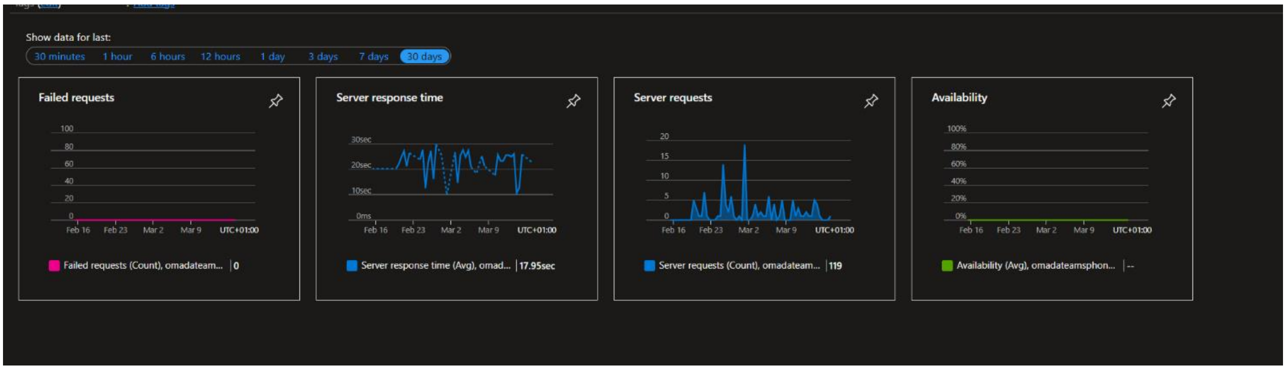
ⓘ Firewall | Service Group Object Name | Member | Description

-- None -- ▾ | Type name to search

Reset | Remove service group object

Firewall | Service Group Object Name | Member | Description

No data to display



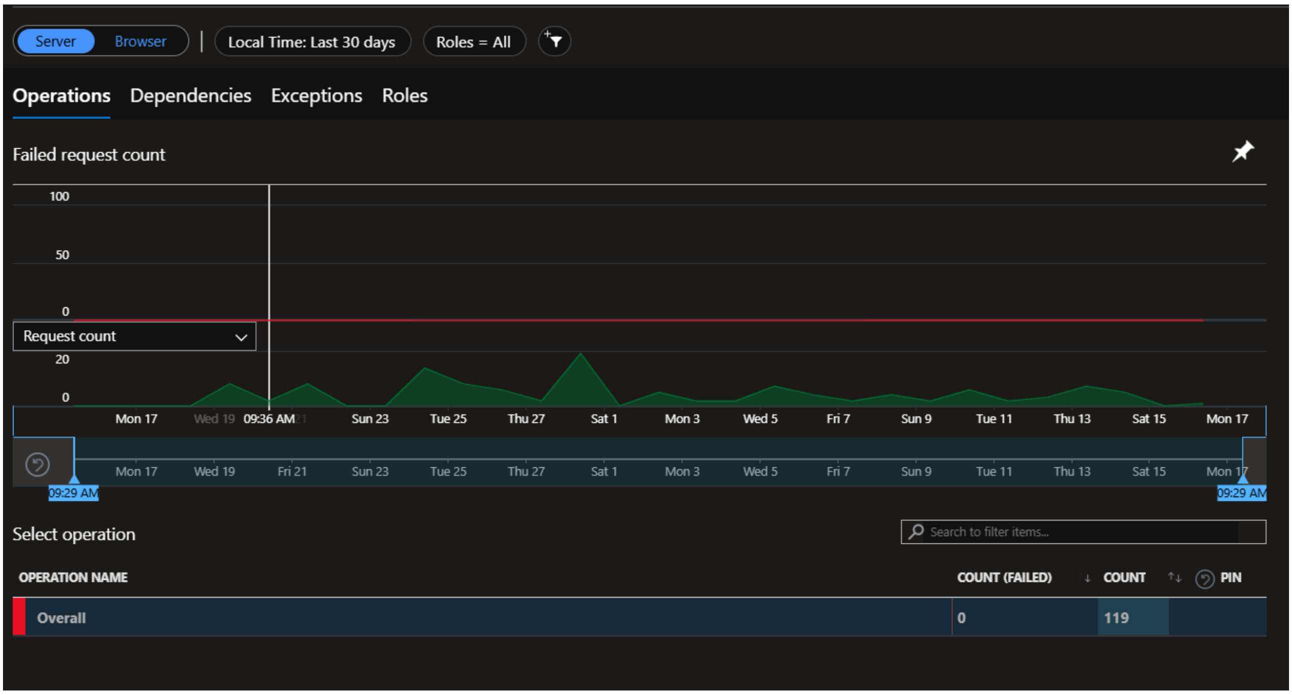
Show data for last:

Data Collection from Graphene Diodes

Graphene diodes are used to collect high-quality neural data due to their excellent electrical conductivity and biocompatibility. This data includes neural signals and other relevant metrics that are crucial for DBS.

Neural Networks for Data Processing

Neural networks, particularly deep learning models, can process and analyze the collected data to optimize DBS parameters. These models can identify patterns and predict outcomes, enhancing the effectiveness of DBS treatments.



Local Time: Last 30 days

Roles = All

Select operation

OPERATION NAME	COUNT (FAILED)	COUNT	PIN
Overall	0	119	

Implementing Firewall Rules

To secure the neural data and DBS systems, firewall rules are implemented as follows:

Data Encryption: Encrypting data collected from graphene diodes ensures that it is protected during transmission and storage. This prevents unauthorized access and tampering.

Access Control: Firewall rules are set to control access to the DBS system. Only authorized personnel and devices can access the neural data and control the DBS system.

Intrusion Detection: Machine learning algorithms can be used to detect anomalies and potential security threats in the data traffic. These algorithms can identify unusual patterns that may indicate a security breach.

Network Segmentation: Segmenting the network ensures that sensitive neural data is isolated from other parts of the network. This minimizes the risk of data leakage and enhances security.

Regular Updates: Firewall rules and security protocols are regularly updated to address new threats and vulnerabilities. This ensures that the DBS system remains secure over time.

Integration with AI/ML and Cybersecurity

AI and ML models can continuously monitor the neural data and firewall performance, making real-time adjustments to enhance security and efficiency. Cybersecurity measures, such as intrusion prevention systems, can be integrated to provide an additional layer of protection.

By combining these technologies, we can create a secure and efficient DBS system that leverages the high-quality data collected from graphene diodes

Cybersecurity, observability and monitoring solution for central patient monitoring health application and nanotechnology diodes and future solutions of equipment on multi-platform OS

The integration of advanced technologies such as Nexthink, artificial intelligence (AI), machine learning (ML), large language models (LLM), and generative AI in healthcare applications presents a transformative approach to cybersecurity, observability, and monitoring. This thesis explores the development and commercial application of Nexthink in central patient monitoring health applications and nanotechnology diodes, with a focus on AI-driven charge transfer mechanisms in deep brain stimulation (DBS) for the diagnosis and treatment of brain diseases.

Nexthink Capabilities

Nexthink's platform offers comprehensive tools for digital employee experience management, which can be adapted for healthcare applications. Key capabilities include:

- AI-Powered Diagnostics: Nexthink uses AI to monitor and compare technology performance across devices, operating systems, applications, and network connectivity.
- Real-Time Alerting and Remediation: The platform provides real-time insights and recommendations for fixing issues, ensuring optimal digital experiences.
- Visibility and Analytics: Nexthink Infinity unlocks visibility across employee devices, applications, operating systems, physical locations, and network connectivity.

Integration with Healthcare Applications

For central patient monitoring health applications, Nexthink's capabilities can be leveraged to ensure robust cybersecurity, observability, and monitoring from different perspective, like:

- Cybersecurity: Implementing Nexthink's AI-powered diagnostics can help identify and mitigate security threats in real-time, ensuring patient data protection.

- Observability: Nexthink's real-time alerting and analytics can provide continuous monitoring of patient health data, enabling proactive interventions.
- Monitoring: The platform's visibility across devices and applications can ensure seamless integration and monitoring of health applications and nanotechnology diodes.

AI and Charge Transfer in DBS

Integrating AI with charge transfer mechanisms in DBS involves several innovative approaches:

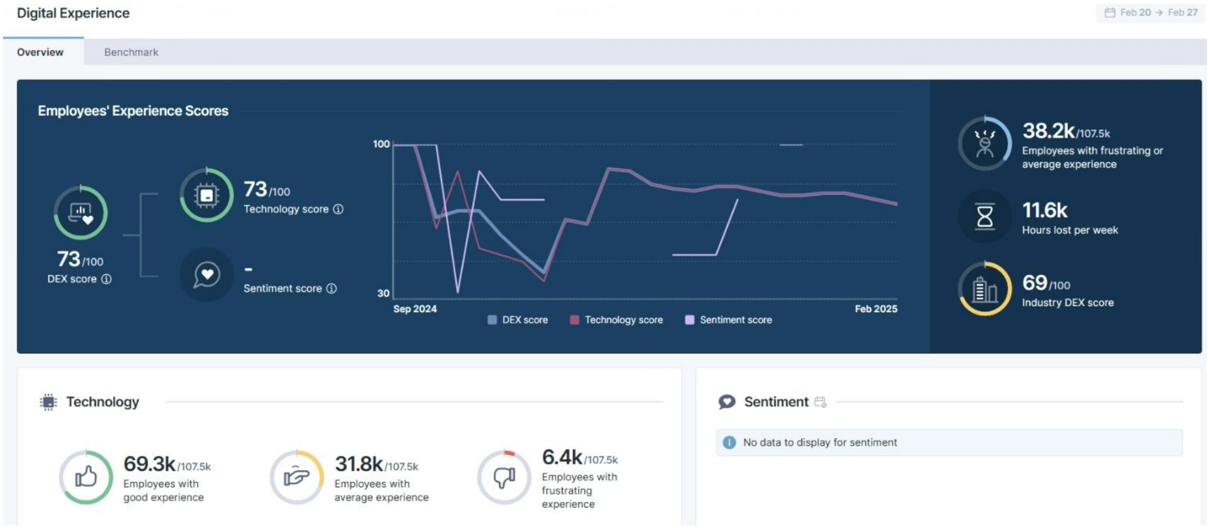
- Graphene Diodes: Utilizing graphene diodes for monitoring cerebral blood flow dynamics and providing electrical stimulation to enhance brain oxygenation and mitigate stroke risks.
- Predictive Modeling: Leveraging AI and ML for predictive analysis to foresee potential health issues and enable proactive interventions.
- Generative AI: Implementing generative AI for personalized treatment plans and real-time monitoring and adjustment of DBS.

Future Solutions and Multi-Platform Integration

To develop future solutions for equipment on multi-platform OS, consider the following steps:

- Multi-Platform Integration: Ensure compatibility and seamless integration of health applications across various operating systems using Nexthink's AI-powered insights
- Regulatory and Ethical Considerations: Address regulatory and ethical considerations for AI and cybersecurity in healthcare applications.
- Advanced Applications: Explore advanced applications such as responsible AI implementation and evaluation, optimizing deep learning models, and integrating neural networks for data processing.

By leveraging Nexthink's capabilities and integrating AI with charge transfer mechanisms in DBS, there is a possibility to develop robust cybersecurity, observability, and monitoring solutions for central patient monitoring health applications and nanotechnology diodes. These approaches will enhance patient outcomes and ensure the responsible use of AI in healthcare.



Digital Experience

Feb 20 → Feb 27

Overview | Benchmark

Employees' Experience Scores

73/100 DEX score ⓘ

73/100 Technology score ⓘ

- Sentiment score ⓘ

38.2k/107.5k Employees with frustrating or average experience

11.6k Hours lost per week

69/100 Industry DEX score

Technology

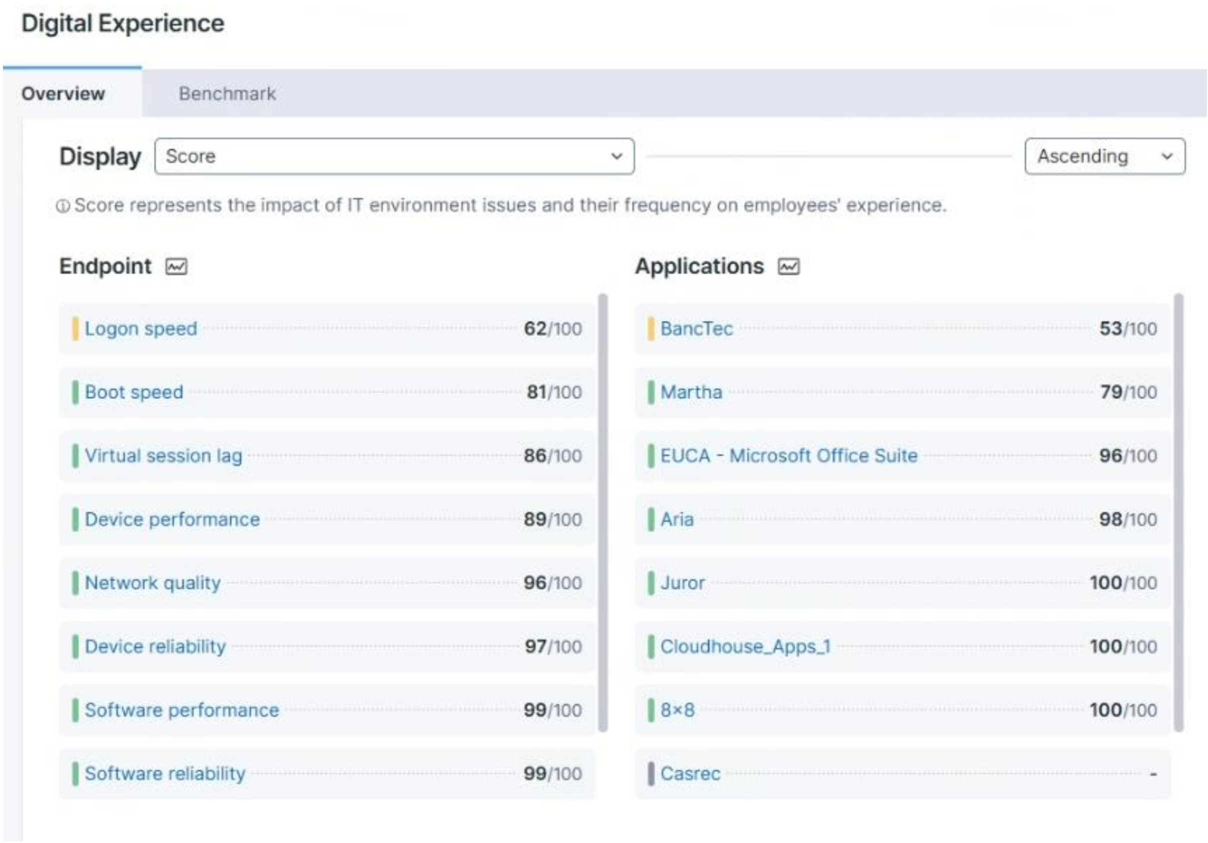
69.3k/107.5k Employees with good experience

31.8k/107.5k Employees with average experience

6.4k/107.5k Employees with frustrating experience

Sentiment

ⓘ No data to display for sentiment



Digital Experience

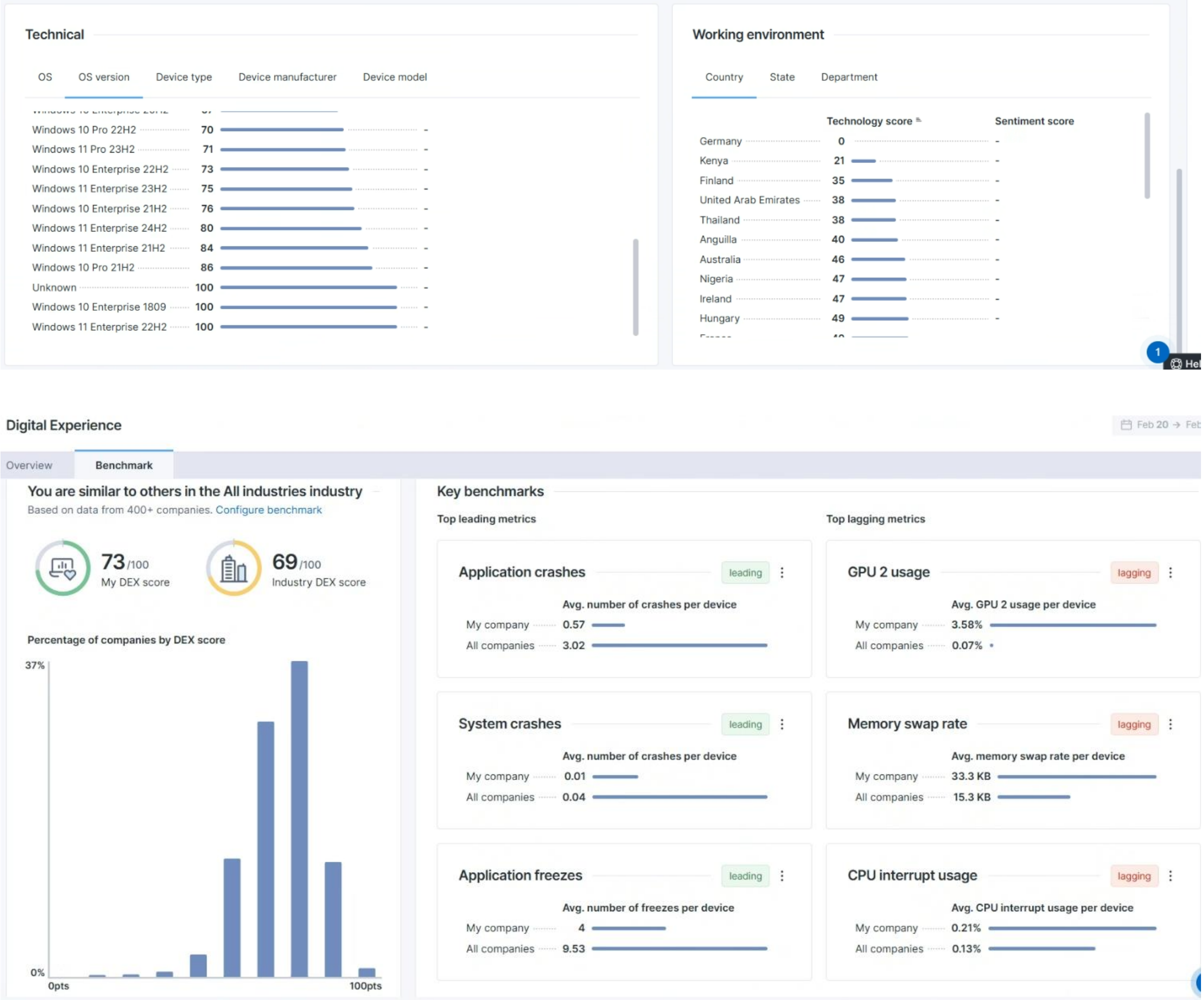
Overview | Benchmark

Display: Score | Ascending

ⓘ Score represents the impact of IT environment issues and their frequency on employees' experience.

Endpoint	
Logon speed	62/100
Boot speed	81/100
Virtual session lag	86/100
Device performance	89/100
Network quality	96/100
Device reliability	97/100
Software performance	99/100
Software reliability	99/100

Applications	
BancTec	53/100
Martha	79/100
EUCA - Microsoft Office Suite	96/100
Aria	98/100
Juror	100/100
Cloudhouse_Apps_1	100/100
8×8	100/100
Casrec	-



Technical

Windows 10 Pro 22H2	70	-
Windows 11 Pro 23H2	71	-
Windows 10 Enterprise 22H2	73	-
Windows 11 Enterprise 23H2	75	-
Windows 10 Enterprise 21H2	76	-
Windows 11 Enterprise 24H2	80	-
Windows 11 Enterprise 21H2	84	-
Windows 10 Pro 21H2	86	-
Unknown	100	-
Windows 10 Enterprise 1809	100	-
Windows 11 Enterprise 22H2	100	-

Working environment

	Technology score	Sentiment score
Germany	0	-
Kenya	21	-
Finland	35	-
United Arab Emirates	38	-
Thailand	38	-
Anguilla	40	-
Australia	46	-
Nigeria	47	-
Ireland	47	-
Hungary	49	-

Digital Experience

You are similar to others in the All industries industry

Based on data from 400+ companies. Configure benchmark

73/100 My DEX score

69/100 Industry DEX score

Percentage of companies by DEX score

Key benchmarks

Top leading metrics

Application crashes — leading

Avg. number of crashes per device

My company	0.57
All companies	3.02

System crashes — leading

Avg. number of crashes per device

My company	0.01
All companies	0.04

Application freezes — leading

Avg. number of freezes per device

My company	4
All companies	9.53

Top lagging metrics

GPU 2 usage — lagging

Avg. GPU 2 usage per device

My company	3.58%
All companies	0.07%

Memory swap rate — lagging

Avg. memory swap rate per device

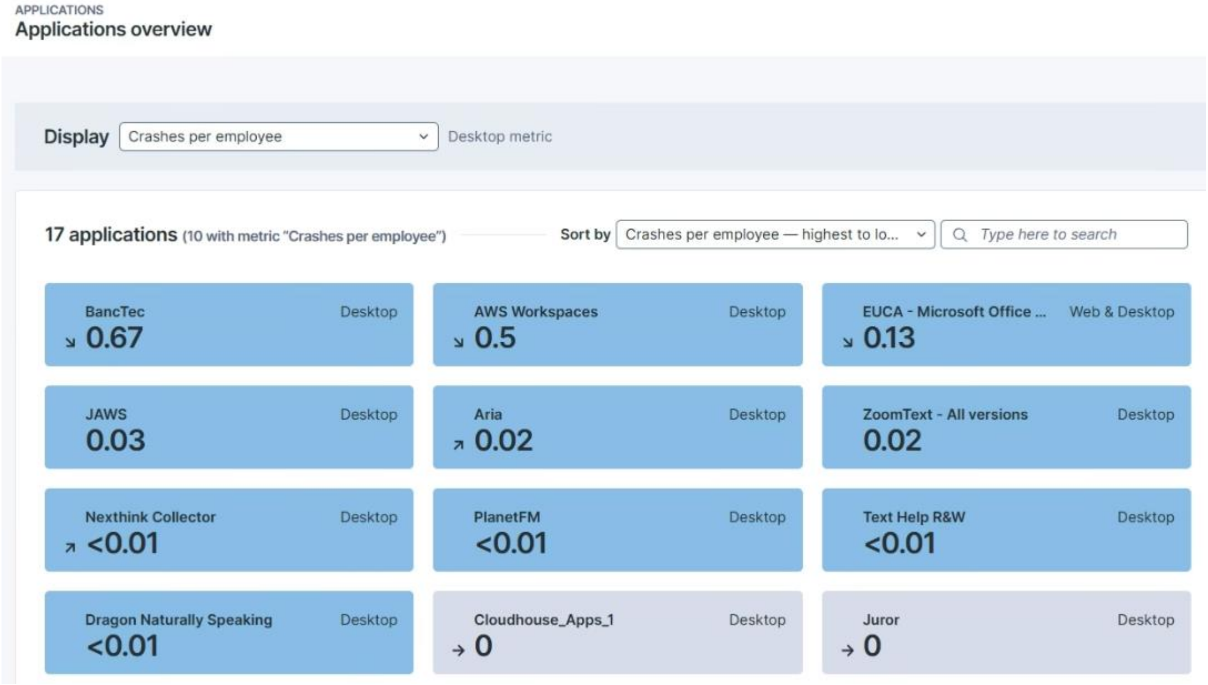
My company	33.3 KB
All companies	15.3 KB

CPU interrupt usage — lagging

Avg. CPU interrupt usage per device

My company	0.21%
All companies	0.13%

The development of a comprehensive monitoring system across multiple platforms is inherently complex and likely warrants a separate PhD thesis. In the interim, for multi-platform adoption, it is essential to meet minimum equipment and software requirements. Ultimately, this endeavor is of paramount importance as it pertains to human life and the integrity of human data.



APPLICATIONS

Applications overview

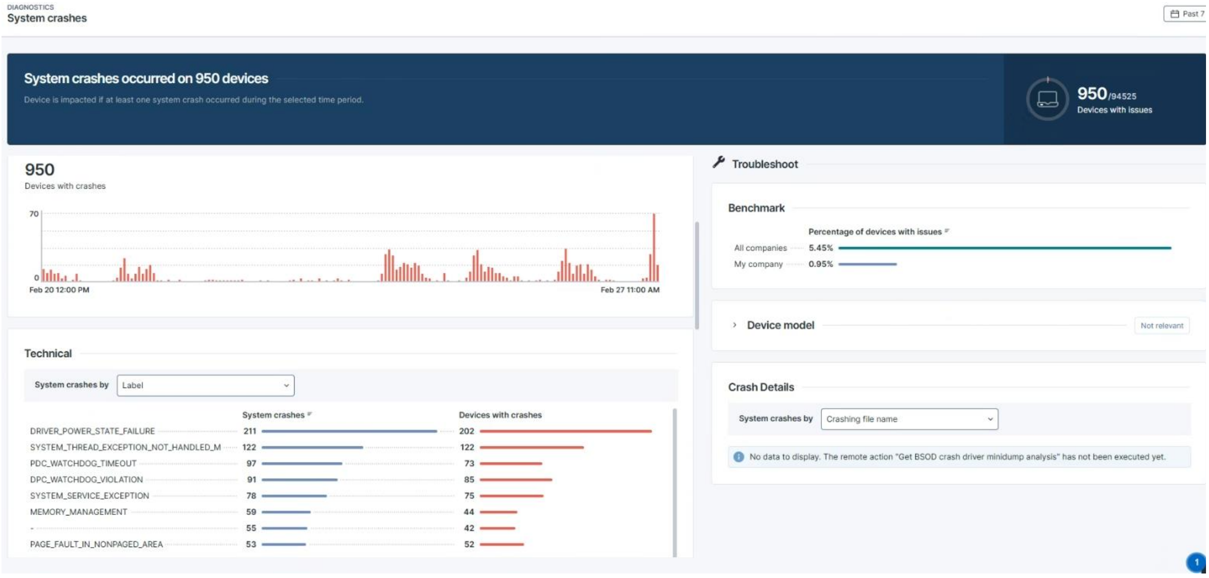
Display: Crashes per employee — Desktop metric

17 applications (10 with metric "Crashes per employee")

Sort by: Crashes per employee — highest to lo...

Type here to search

BancTec	Desktop	↘ 0.67
AWS Workspaces	Desktop	↘ 0.5
EUCA - Microsoft Office ...	Web & Desktop	↘ 0.13
JAWS	Desktop	0.03
Aria	Desktop	↗ 0.02
ZoomText - All versions	Desktop	0.02
Nexthink Collector	Desktop	↗ <0.01
PlanetFM	Desktop	<0.01
Text Help R&W	Desktop	<0.01
Dragon Naturally Speaking	Desktop	<0.01
Cloudhouse_Apps_1	Desktop	→ 0
Juror	Desktop	→ 0



DIAGNOSTICS

System crashes

Past 7

System crashes occurred on 950 devices

Device is impacted if at least one system crash occurred during the selected time period.

950/94525 Devices with issues

950

Devices with crashes

Technical

System crashes by: Label

	System crashes	Devices with crashes
DRIVER_POWER_STATE_FAILURE	211	202
SYSTEM_THREAD_EXCEPTION_NOT_HANDLED_M	122	122
PDC_WATCHDOG_TIMEOUT	97	73
DPC_WATCHDOG_VIOLATION	91	85
SYSTEM_SERVICE_EXCEPTION	78	75
MEMORY_MANAGEMENT	59	44
-	55	42
PAGE_FAULT_IN_NONPAGED_AREA	53	52

Troubleshoot

Benchmark

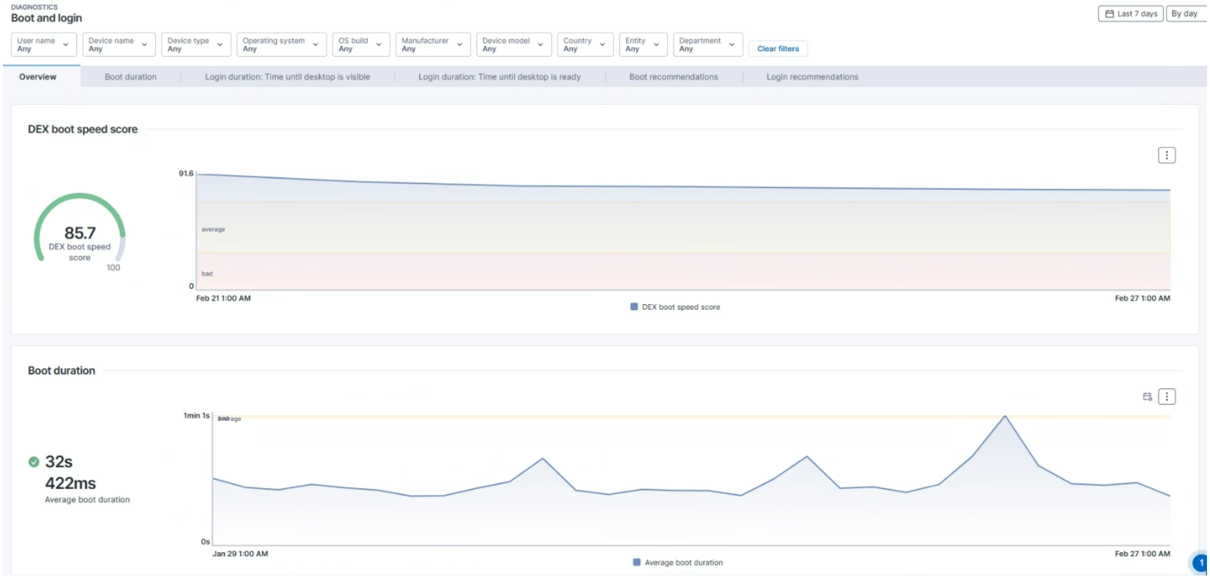
	Percentage of devices with issues
All companies	5.45%
My company	0.95%

Device model — Not relevant

Crash Details

System crashes by: Crashing file name

No data to display. The remote action "Get BSOD crash driver minidump analysis" has not been executed yet.



DIAGNOSTICS

Boot and login

Last 7 days | By day

User name: Any | Device name: Any | Device type: Any | Operating system: Any | OS build: Any | Manufacturer: Any | Device model: Any | Country: Any | Entity: Any | Department: Any | Clear filters

Overview | Boot duration | Login duration: Time until desktop is visible | Login duration: Time until desktop is ready | Boot recommendations | Login recommendations

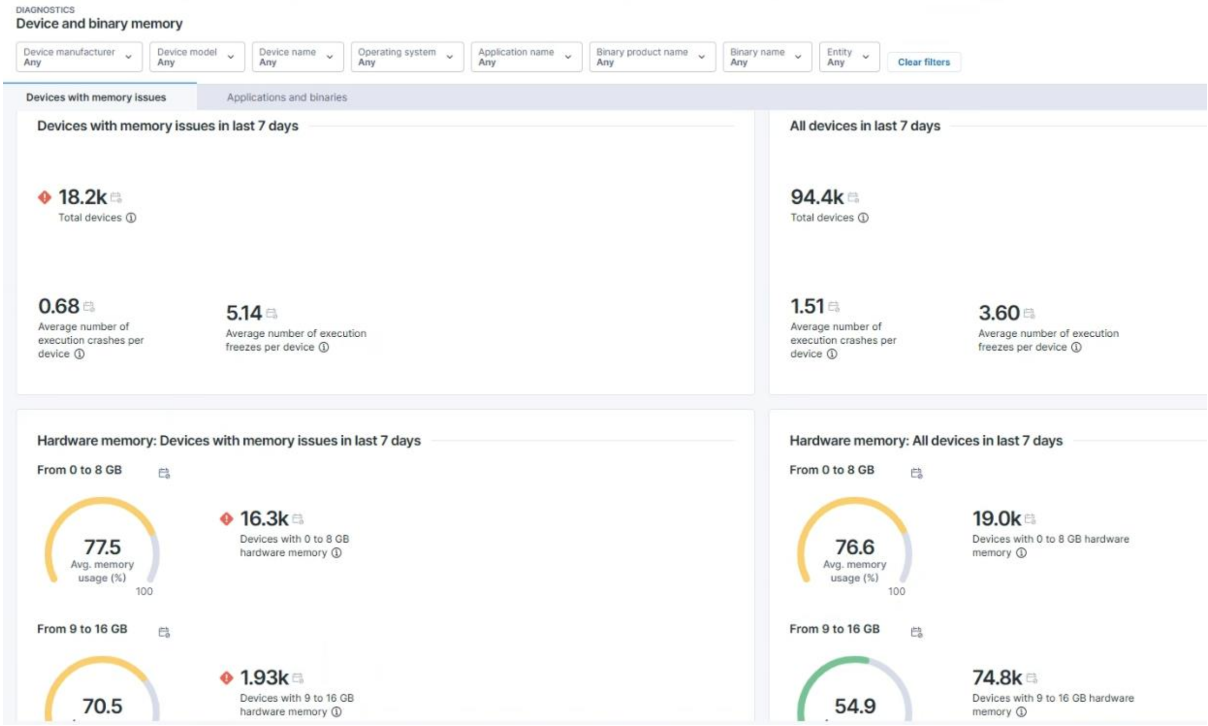
DEX boot speed score

85.7 DEX boot speed score

Boot duration

32s 422ms

Average boot duration



DIAGNOSTICS

Device and binary memory

Device manufacturer: Any | Device model: Any | Device name: Any | Operating system: Any | Application name: Any | Binary product name: Any | Binary name: Any | Entity: Any | Clear filters

Devices with memory issues | Applications and binaries

Devices with memory issues in last 7 days

18.2k Total devices

0.68 Average number of execution crashes per device

5.14 Average number of execution freezes per device

All devices in last 7 days

94.4k Total devices

1.51 Average number of execution crashes per device

3.60 Average number of execution freezes per device

Hardware memory: Devices with memory issues in last 7 days

From 0 to 8 GB

77.5 Avg. memory usage (%)

16.3k Devices with 0 to 8 GB hardware memory

From 9 to 16 GB

70.5

1.93k Devices with 9 to 16 GB hardware memory

Hardware memory: All devices in last 7 days

From 0 to 8 GB

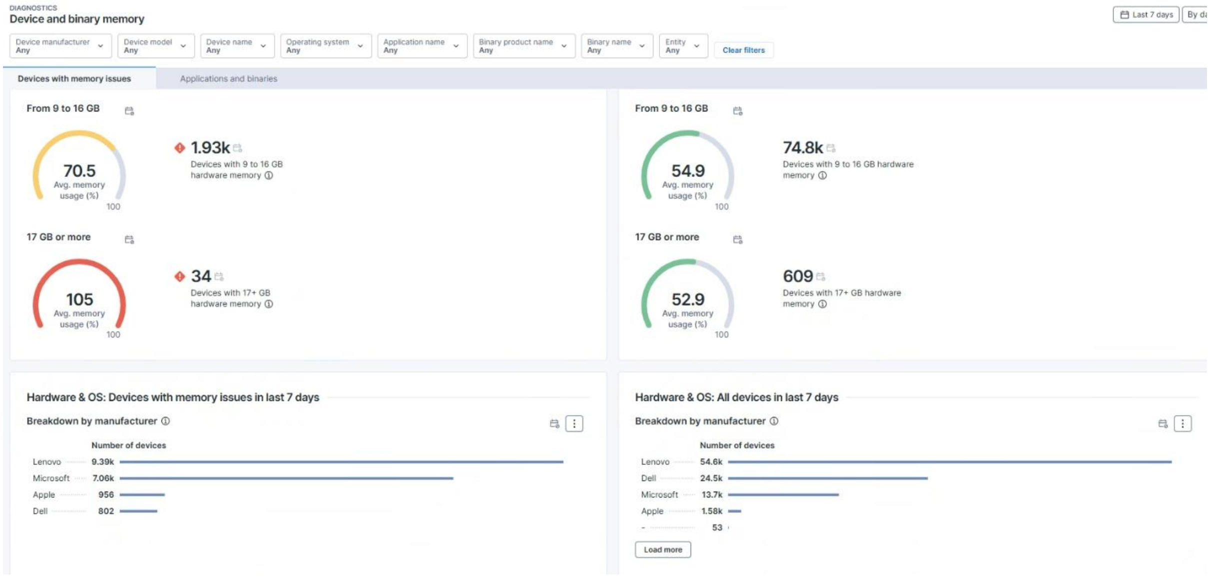
76.6 Avg. memory usage (%)

19.0k Devices with 0 to 8 GB hardware memory

From 9 to 16 GB

54.9

74.8k Devices with 9 to 16 GB hardware memory



Device and binary memory

Device manufacturer Any | Device model Any | Device name Any | Operating system Any | Application name Any | Binary product name Any | Binary name Any | Entity Any | Clear filters

From 9 to 16 GB

70.5 Avg. memory usage (%) 100

1.93k Devices with 9 to 16 GB hardware memory

17 GB or more

105 Avg. memory usage (%) 100

34 Devices with 17+ GB hardware memory

From 9 to 16 GB

54.9 Avg. memory usage (%) 100

74.8k Devices with 9 to 16 GB hardware memory

17 GB or more

52.9 Avg. memory usage (%) 100

609 Devices with 17+ GB hardware memory

Hardware & OS: Devices with memory issues in last 7 days

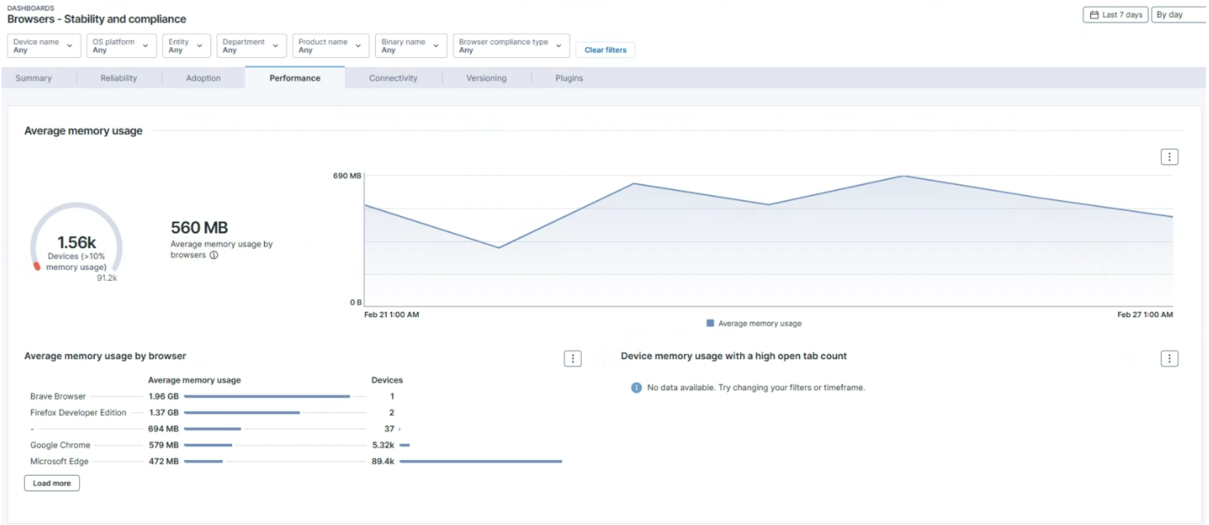
Breakdown by manufacturer

	Number of devices
Lenovo	9.39k
Microsoft	7.06k
Apple	956
Dell	802

Hardware & OS: All devices in last 7 days

Breakdown by manufacturer

	Number of devices
Lenovo	54.6k
Dell	24.5k
Microsoft	13.7k
Apple	1.58k
-	53



Browsers - Stability and compliance

Device name Any | OS platform Any | Entity Any | Department Any | Product name Any | Binary name Any | Browser compliance type Any | Clear filters

Average memory usage

1.56k Devices (>10% memory usage) 91.2k

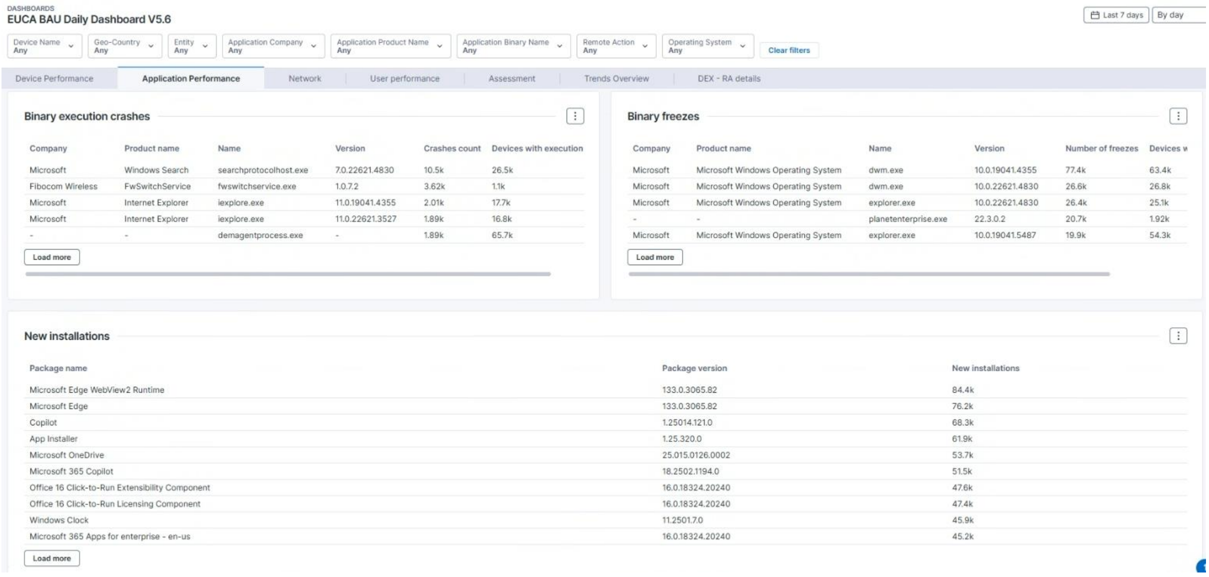
560 MB Average memory usage by browsers

Average memory usage by browser

	Average memory usage	Devices
Brave Browser	1.96 GB	1
Firefox Developer Edition	1.37 GB	2
-	694 MB	37
Google Chrome	579 MB	5.32k
Microsoft Edge	472 MB	89.4k

Device memory usage with a high open tab count

No data available. Try changing your filters or timeframe.



EUCA BAU Daily Dashboard V5.6

Device Name Any | Geo-Country Any | Entity Any | Application Company Any | Application Product Name Any | Application Binary Name Any | Remote Action Any | Operating System Any | Clear filters

Binary execution crashes

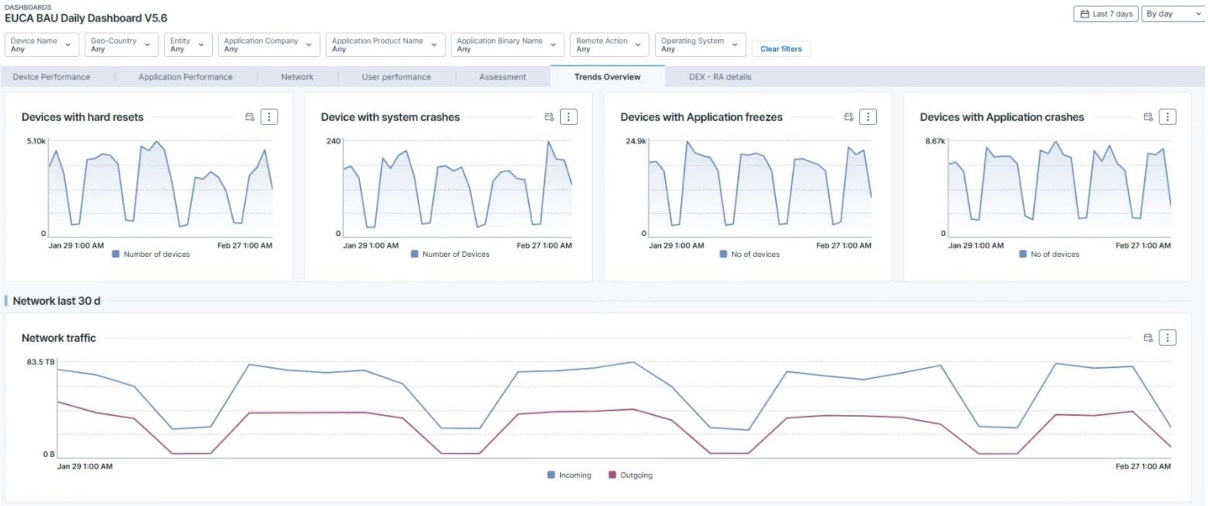
Company	Product name	Name	Version	Crashes count	Devices with execution
Microsoft	Windows Search	searchprotocolhost.exe	7.0.22621.4830	10.5k	26.5k
Fibocom Wireless	FwSwitchService	fwswitchservice.exe	1.0.7.2	3.62k	1.1k
Microsoft	Internet Explorer	iexplore.exe	11.0.19041.4355	2.01k	17.7k
Microsoft	Internet Explorer	iexplore.exe	11.0.22621.3527	1.89k	16.8k
-	-	demagentprocess.exe	-	1.89k	65.7k

Binary freezes

Company	Product name	Name	Version	Number of freezes	Devices w
Microsoft	Microsoft Windows Operating System	dwm.exe	10.0.19041.4355	77.4k	63.4k
Microsoft	Microsoft Windows Operating System	dwm.exe	10.0.22621.4830	26.6k	26.8k
Microsoft	Microsoft Windows Operating System	explorer.exe	10.0.22621.4830	26.4k	25.1k
-	-	planetenterprise.exe	22.3.0.2	20.7k	1.92k
Microsoft	Microsoft Windows Operating System	explorer.exe	10.0.19041.5487	19.9k	54.3k

New installations

Package name	Package version	New installations
Microsoft Edge WebView2 Runtime	133.0.3065.82	84.4k
Microsoft Edge	133.0.3065.82	76.2k
Copilot	1.25014.121.0	68.3k
App Installer	1.25.320.0	61.9k
Microsoft OneDrive	25.015.0126.0002	53.7k
Microsoft 365 Copilot	18.2502.1194.0	51.5k
Office 16 Click-to-Run Extensibility Component	16.0.18324.20240	47.6k
Office 16 Click-to-Run Licensing Component	16.0.18324.20240	47.4k
Windows Clock	11.2501.7.0	45.9k
Microsoft 365 Apps for enterprise - en-us	16.0.18324.20240	45.2k



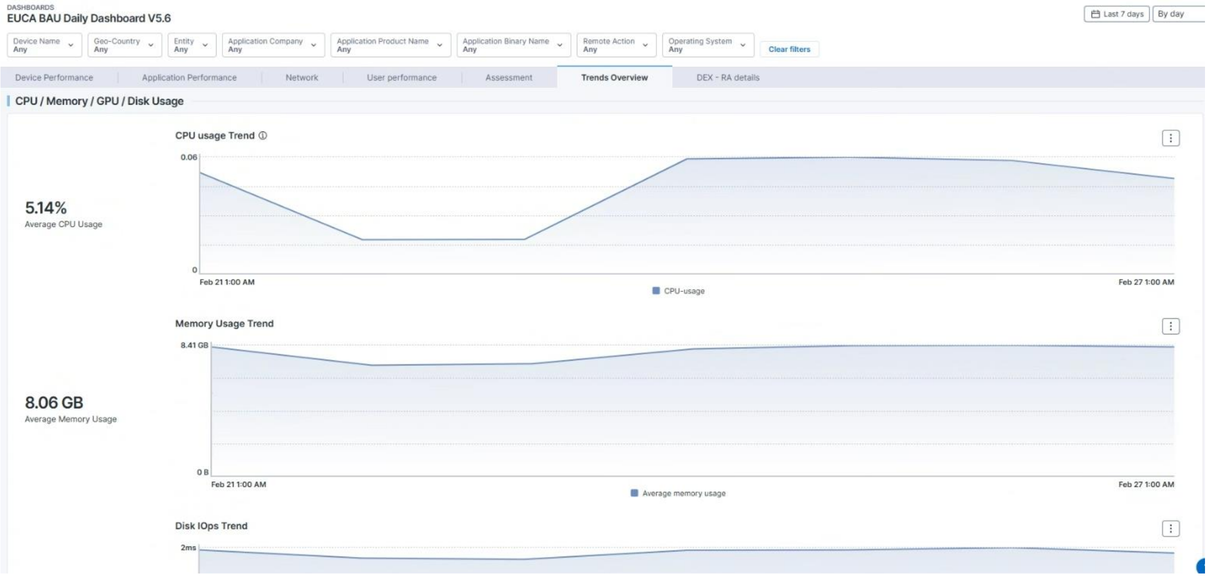
DASHBOARDS
EUCA BAU Daily Dashboard V5.6

Last 7 days | By day

Device Name: Any | Geo-Country: Any | Entity: Any | Application Company: Any | Application Product Name: Any | Application Binary Name: Any | Remote Action: Any | Operating System: Any | Clear filters

Device Performance | Application Performance | Network | User performance | Assessment | Trends Overview | DEX - RA details

Network last 30 d



DASHBOARDS
EUCA BAU Daily Dashboard V5.6

Last 7 days | By day

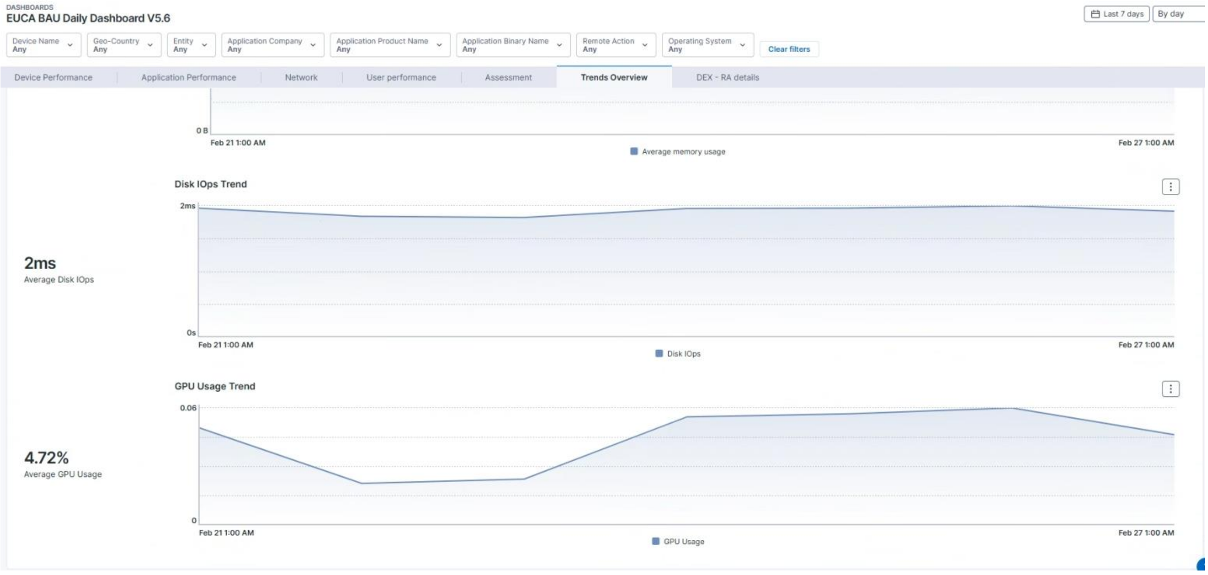
Device Name: Any | Geo-Country: Any | Entity: Any | Application Company: Any | Application Product Name: Any | Application Binary Name: Any | Remote Action: Any | Operating System: Any | Clear filters

Device Performance | Application Performance | Network | User performance | Assessment | Trends Overview | DEX - RA details

CPU / Memory / GPU / Disk Usage

5.14%
Average CPU Usage

8.06 GB
Average Memory Usage



DASHBOARDS
EUCA BAU Daily Dashboard V5.6

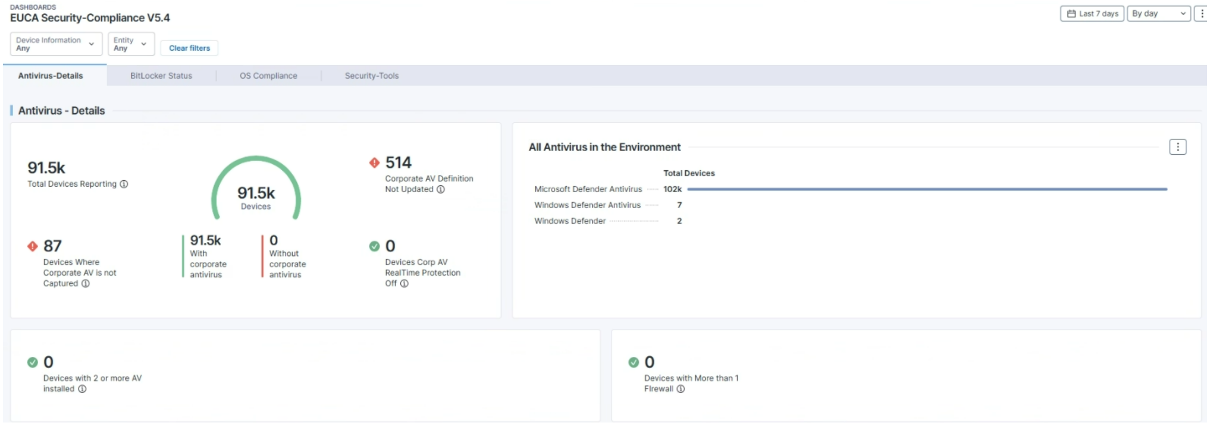
Last 7 days | By day

Device Name: Any | Geo-Country: Any | Entity: Any | Application Company: Any | Application Product Name: Any | Application Binary Name: Any | Remote Action: Any | Operating System: Any | Clear filters

Device Performance | Application Performance | Network | User performance | Assessment | Trends Overview | DEX - RA details

2ms
Average Disk IOps

4.72%
Average GPU Usage



EUCA Security-Compliance V5.4

Antivirus - Details

91.5k
Total Devices Reporting ⓘ

91.5k
Devices

514
Corporate AV Definition Not Updated ⓘ

87
Devices Where Corporate AV is not Captured ⓘ

91.5k
With corporate antivirus

0
Without corporate antivirus

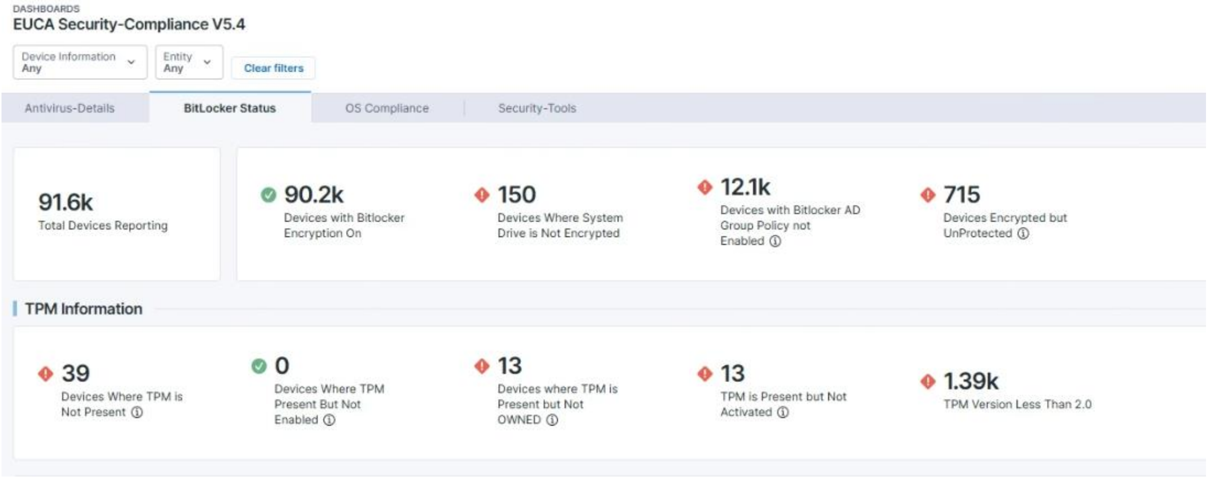
0
Devices Corp AV RealTime Protection Off ⓘ

All Antivirus in the Environment

	Total Devices
Microsoft Defender Antivirus	102k
Windows Defender Antivirus	7
Windows Defender	2

0
Devices with 2 or more AV installed ⓘ

0
Devices with More than 1 Firewall ⓘ



EUCA Security-Compliance V5.4

91.6k
Total Devices Reporting

90.2k
Devices with Bitlocker Encryption On

150
Devices Where System Drive is Not Encrypted

12.1k
Devices with Bitlocker AD Group Policy not Enabled ⓘ

715
Devices Encrypted but UnProtected ⓘ

TPM Information

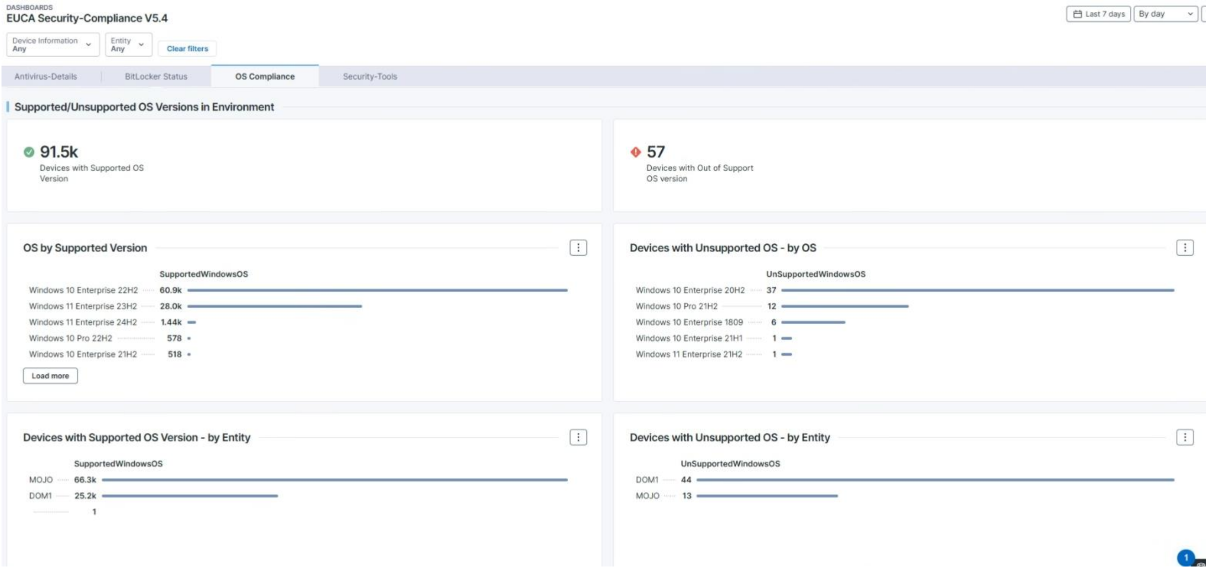
39
Devices Where TPM is Not Present ⓘ

0
Devices Where TPM Present But Not Enabled ⓘ

13
Devices where TPM is Present but Not OWNED ⓘ

13
TPM is Present but Not Activated ⓘ

1.39k
TPM Version Less Than 2.0



EUCA Security-Compliance V5.4

Supported/Unsupported OS Versions in Environment

91.5k
Devices with Supported OS Version

57
Devices with Out of Support OS version

OS by Supported Version

	SupportedWindowsOS
Windows 10 Enterprise 22H2	60.9k
Windows 11 Enterprise 23H2	28.0k
Windows 11 Enterprise 24H2	1.44k
Windows 10 Pro 22H2	578
Windows 10 Enterprise 21H2	518

Devices with Unsupported OS - by OS

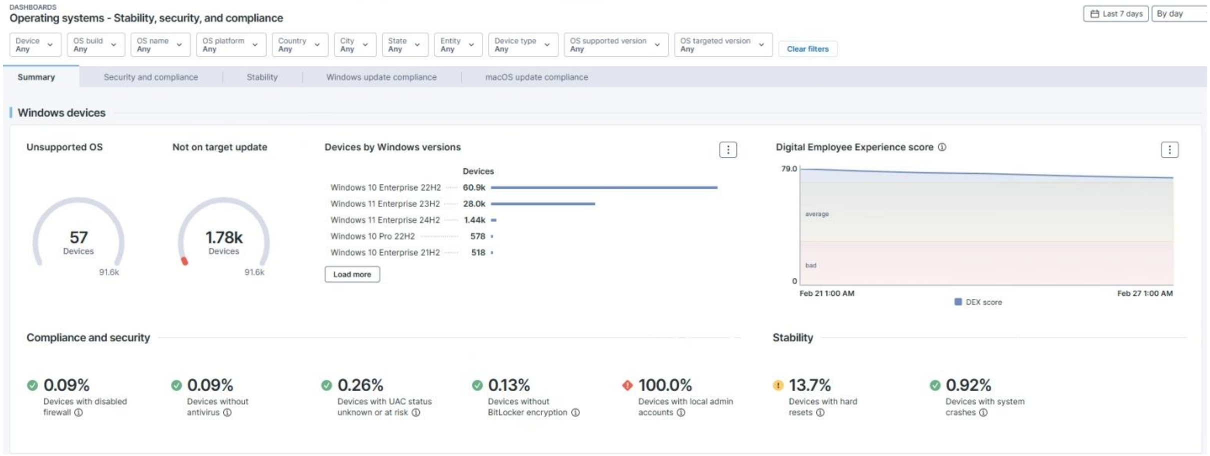
	UnSupportedWindowsOS
Windows 10 Enterprise 20H2	37
Windows 10 Pro 21H2	12
Windows 10 Enterprise 1809	6
Windows 10 Enterprise 21H1	1
Windows 11 Enterprise 21H2	1

Devices with Supported OS Version - by Entity

	SupportedWindowsOS
MOJO	66.3k
DOM1	25.2k
	1

Devices with Unsupported OS - by Entity

	UnSupportedWindowsOS
DOM1	44
MOJO	13



Operating systems - Stability, security, and compliance

Device Any | OS build Any | OS name Any | OS platform Any | Country Any | City Any | State Any | Entity Any | Device type Any | OS supported version Any | OS targeted version Any

Windows devices

Unsupported OS

57 Devices (91.6k)

Not on target update

1.78k Devices (91.6k)

Devices by Windows versions

	Devices
Windows 10 Enterprise 22H2	60.9k
Windows 11 Enterprise 23H2	28.0k
Windows 11 Enterprise 24H2	1.44k
Windows 10 Pro 22H2	578
Windows 10 Enterprise 21H2	518

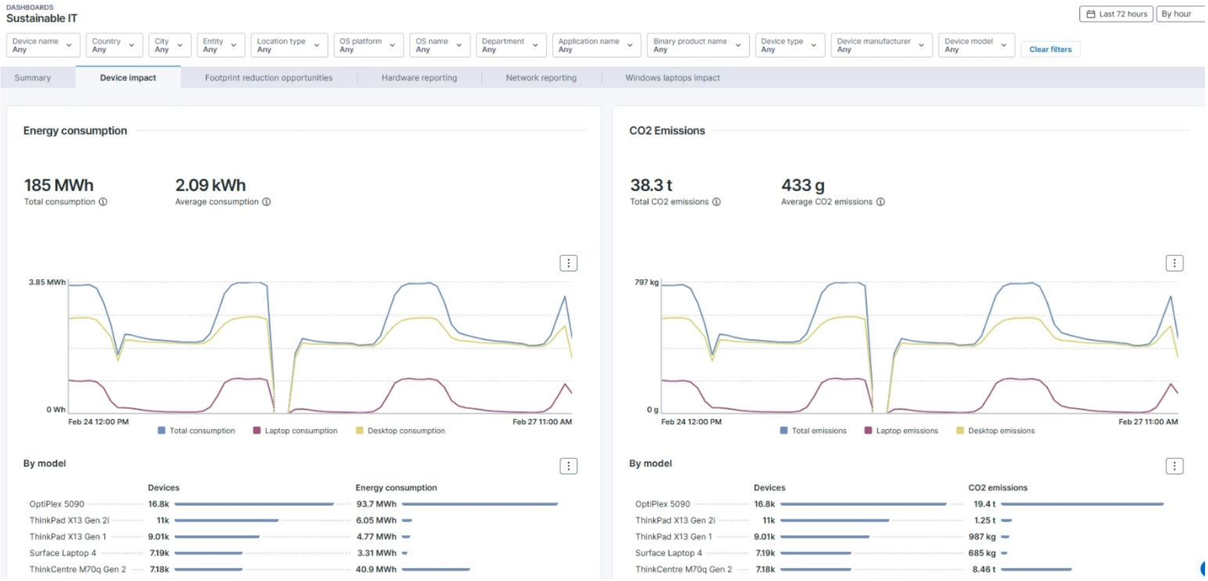
Digital Employee Experience score ⓘ

Compliance and security

- ✓ 0.09% Devices with disabled firewall ⓘ
- ✓ 0.09% Devices without antivirus ⓘ
- ✓ 0.26% Devices with UAC status unknown or at risk ⓘ
- ✓ 0.13% Devices without BitLocker encryption ⓘ
- ! 100.0% Devices with local admin accounts ⓘ

Stability

- ! 13.7% Devices with hard resets ⓘ
- ✓ 0.92% Devices with system crashes ⓘ



Sustainable IT

Device name Any | Country Any | City Any | Entity Any | Location type Any | OS platform Any | OS name Any | Department Any | Application name Any | Binary product name Any | Device type Any | Device manufacturer Any | Device model Any

Energy consumption

185 MWh Total consumption ⓘ

2.09 kWh Average consumption ⓘ

By model

	Devices	Energy consumption
OptiPlex 5090	16.8k	93.7 MWh
ThinkPad X13 Gen 2i	11k	6.05 MWh
ThinkPad X13 Gen 1	9.01k	4.77 MWh
Surface Laptop 4	7.19k	3.31 MWh
ThinkCentre M70q Gen 2	7.18k	40.9 MWh

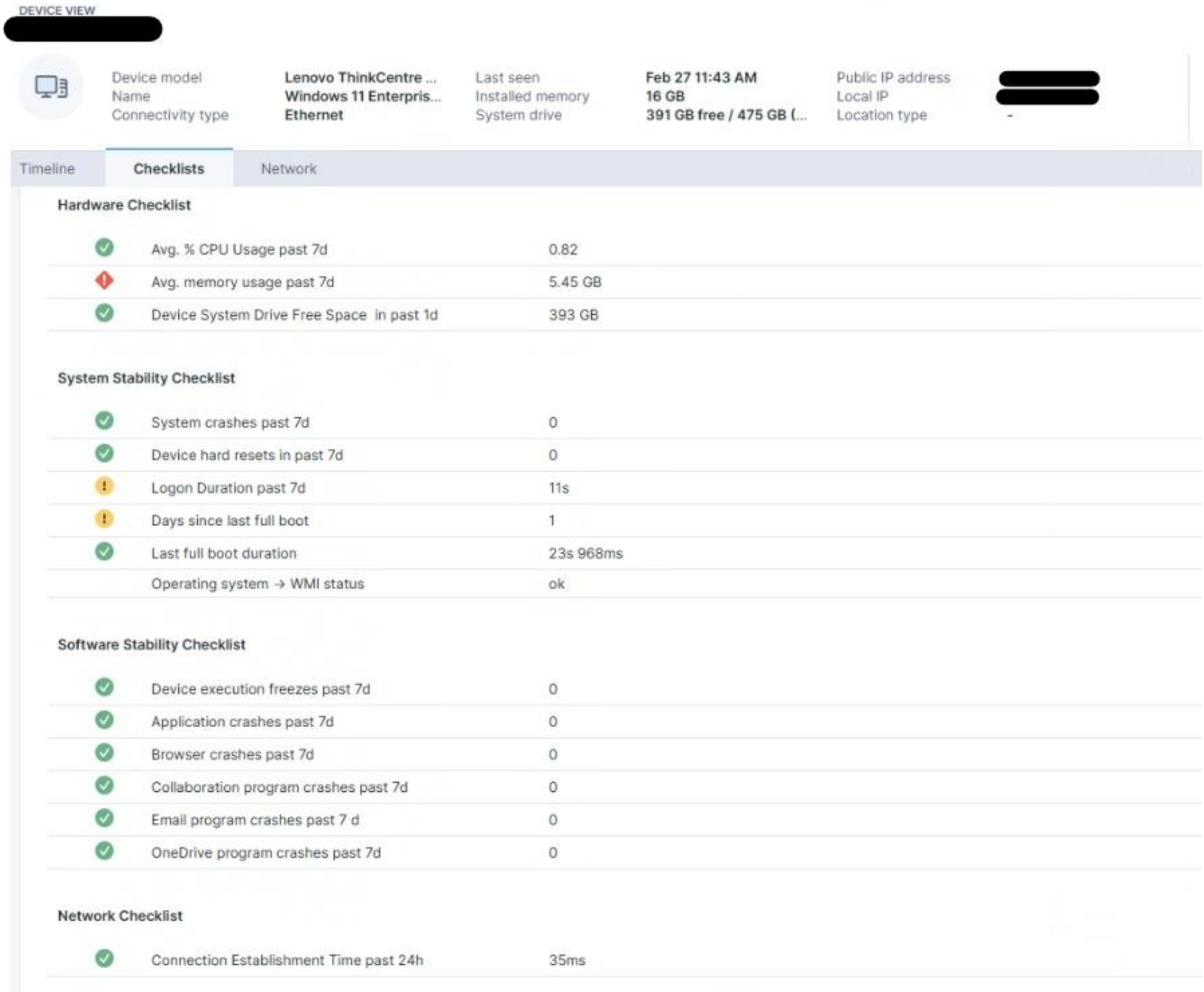
CO2 Emissions

38.3 t Total CO2 emissions ⓘ

433 g Average CO2 emissions ⓘ

By model

	Devices	CO2 emissions
OptiPlex 5090	16.8k	19.4 t
ThinkPad X13 Gen 2i	11k	1.25 t
ThinkPad X13 Gen 1	9.01k	987 kg
Surface Laptop 4	7.19k	685 kg
ThinkCentre M70q Gen 2	7.18k	8.46 t



DEVICE VIEW

[redacted]

Device model Lenovo ThinkCentre ...
Name Windows 11 Enterpris...
Connectivity type Ethernet

Last seen Feb 27 11:43 AM
Installed memory 16 GB
System drive 391 GB free / 475 GB (...

Public IP address [redacted]
Local IP [redacted]
Location type -

Hardware Checklist

✓	Avg. % CPU Usage past 7d	0.82
◆	Avg. memory usage past 7d	5.45 GB
✓	Device System Drive Free Space in past 1d	393 GB

System Stability Checklist

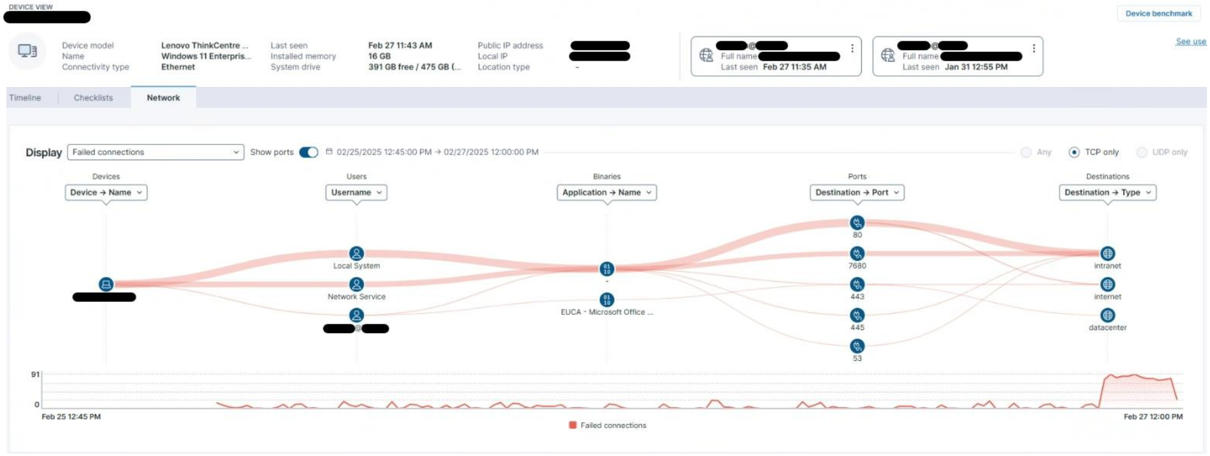
✓	System crashes past 7d	0
✓	Device hard resets in past 7d	0
!	Logon Duration past 7d	11s
!	Days since last full boot	1
✓	Last full boot duration	23s 968ms
	Operating system → WMI status	ok

Software Stability Checklist

✓	Device execution freezes past 7d	0
✓	Application crashes past 7d	0
✓	Browser crashes past 7d	0
✓	Collaboration program crashes past 7d	0
✓	Email program crashes past 7 d	0
✓	OneDrive program crashes past 7d	0

Network Checklist

✓	Connection Establishment Time past 24h	35ms



DEVICE VIEW

[redacted]

Device model Lenovo ThinkCentre ...
Name Windows 11 Enterpris...
Connectivity type Ethernet

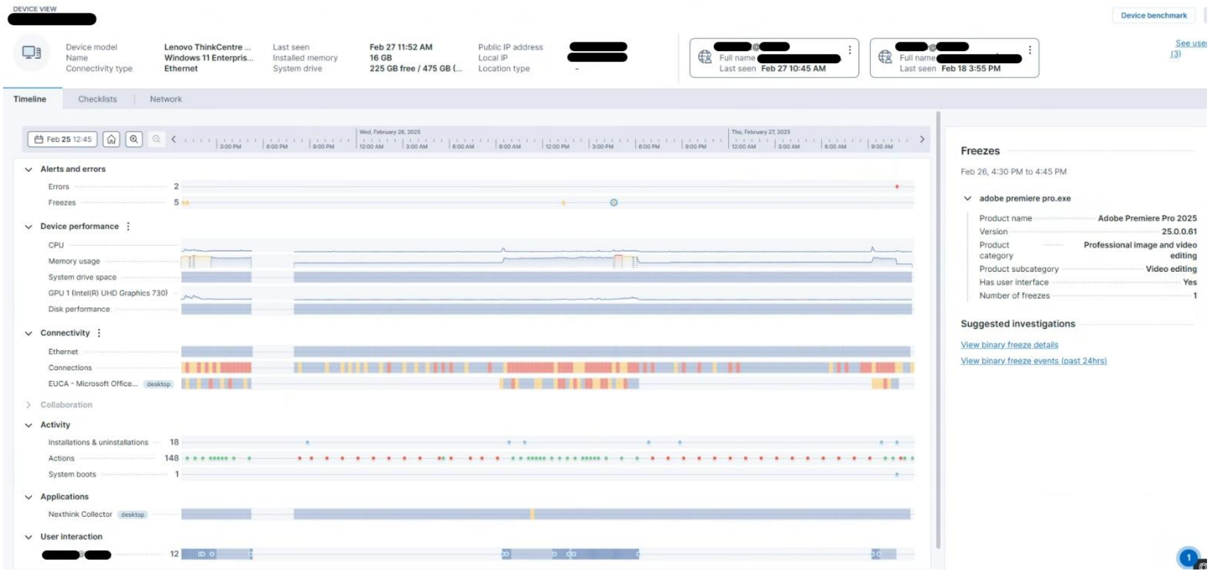
Last seen Feb 27 11:43 AM
Installed memory 16 GB
System drive 391 GB free / 475 GB (...

Public IP address [redacted]
Local IP [redacted]
Location type -

[redacted]@[redacted]
Full name [redacted]
Last seen Feb 27 11:35 AM

[redacted]@[redacted]
Full name [redacted]
Last seen Jan 31 12:55 PM

Display: Failed connections
Show ports ●
02/25/2025 12:45:00 PM → 02/27/2025 12:00:00 PM
○ Any ● TCP only ○ UDP only



DEVICE VIEW

[redacted]

Device model	Lenovo ThinkCentre ...	Last seen	Feb 27 11:52 AM	Public IP address	[redacted]
Name	Windows 11 Enterpris...	Installed memory	16 GB	Local IP	[redacted]
Connectivity type	Ethernet	System drive	225 GB free / 475 GB (...	Location type	-

[redacted]@[redacted]
Full name [redacted]
Last seen Feb 27 10:45 AM

[redacted]@[redacted]
Full name [redacted]
Last seen Feb 18 3:55 PM

See user (3)

Timeline | Checklists | Network

Feb 25 12:45

Alerts and errors

Errors	2
Freezes	5

Device performance

CPU
Memory usage
System drive space
GPU 1 (Intel(R) UHD Graphics 730)
Disk performance

Connectivity

Ethernet
Connections
EUCA - Microsoft Office... desktop

Collaboration

Activity

Installations & uninstallations	18
Actions	148
System boots	1

Applications

Nexthink Collector desktop

User interaction

[redacted]@[redacted]	12

Freezes

Feb 26, 4:30 PM to 4:45 PM

adobe premiere pro.exe

Product name	Adobe Premiere Pro 2025
Version	25.0.0.61
Product category	Professional image and video editing
Product subcategory	Video editing
Has user interface	Yes
Number of freezes	1

Suggested investigations

View binary freeze details

View binary freeze events (past 24hrs)

Figure 52 Nexthink analyzer for underpinned devices with regards to DBS data storage, analyzes, transformation compromised with digitalization and end user experience – Patient data availability on demand with responsible AI of its analysis.

The advent of supercomputers and quantum computers holds significant promise for advancing medical technology, particularly in the realm of deep brain stimulation. The integration of graphene diodes to collect comprehensive input data from various physiological factors—such as brain activity, neurons, the central nervous system, blood pressure, glucose levels, age, gender, diet, cortisol (stress), muscle tension, spine defects, and other factors—could revolutionize the detection and prevention of brain strokes and migraines with aura.

Training models with this extensive data necessitates advanced analysis and equipment, which supercomputers and quantum computers can provide. These powerful machines are capable of handling the complex computations required to process and analyze the data, thereby enabling the development of more accurate and effective models for Vojta rehabilitation and reverse stimulation of the brain in cases of dysfunction.

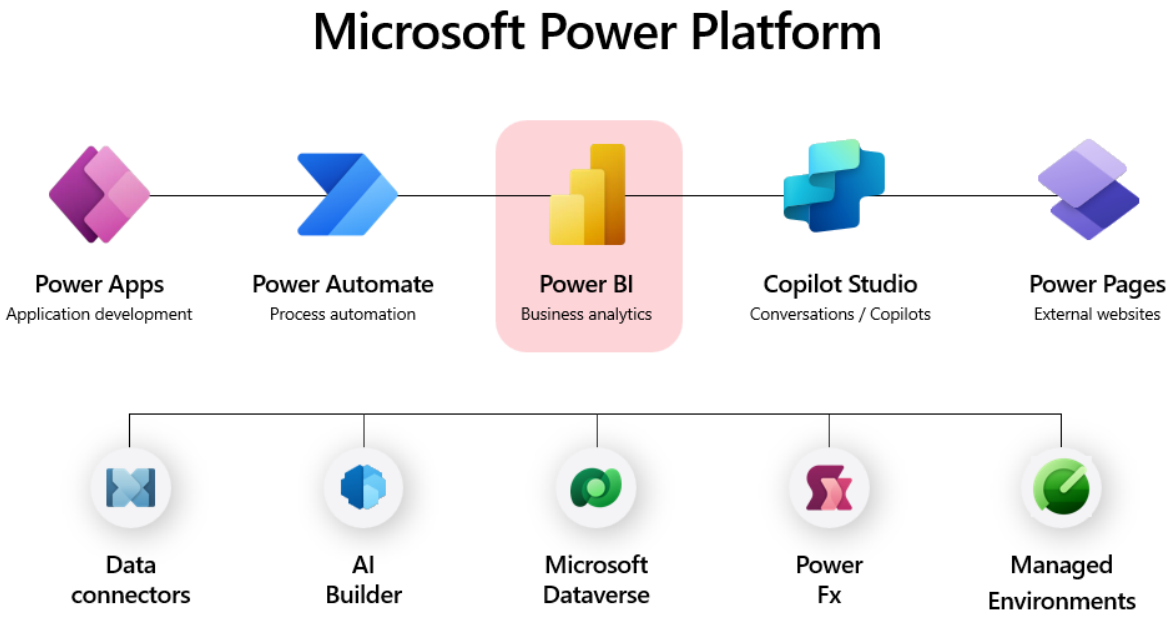
Moreover, integrating charge transfer measurements of the brain with large language models (LLMs) and neural networks could enhance the precision and effectiveness of these models. Ensuring cybersecurity, observability, and monitoring solutions for central patient monitoring health applications, along with the use of nanotechnology diodes and future equipment solutions on multi-platform operating systems, is essential to safeguard the data and ensure reliable performance.

In summary, the release and utilization of supercomputers and quantum computers could significantly advance the field of deep brain stimulation. These technologies provide the necessary computational power to analyze complex data and develop innovative solutions for medical treatments and preventive measures. This approach could lead to more effective and personalized healthcare solutions, ultimately improving patient outcomes and overall health management.

How to enrol solutions on multi-platforms and integration with other apps?

Despite I encourage to create from Python, Julia code level application, there are many other ways to have analysis of brain functioning and underpinned in system of monitoring diodes and devices, like with use both well-known solutions: Configuration Manager (SCCM) at on-prem environment, or Microsoft Endpoint Manager (MS Intune) standalone in cloud or as co-management with SCCM to enroll applications on devices or enroll devices on multi-platforms of Windows, Linux/MacOS, Android, iOS.

There is a possibility to integrate real programming with solutions type: no code/low code – like MS Power Platform with Power apps and Power Automate indicates:



Nowadays there's a powerful synergy between traditional programming (like Python or Julia) and modern enterprise-grade device management and low-code/no-code platforms. There are more and more architectural landscapes that can work together for advanced brain-function monitoring systems or similar applications.

🔧 Enterprise Device Management Integration

Using tools like Microsoft Endpoint Manager (Intune) and System Center Configuration Manager (SCCM) or Airwatch, there are beyond options to:

- Enroll and manage devices across platforms (Windows, macOS, Linux, Android, iOS).
- Deploy and monitor applications that collect or process data from sensors (e.g., EEG, fNIRS, or diode-based systems).
- Ensure compliance and security for sensitive health or neuroscience data.
- Enable co-management to bridge on-prem and cloud environments.

This is especially useful when deploying software that interfaces with brain-monitoring hardware across a distributed network of research or clinical devices.

🧠 Real-Time Brain Function Analysis

Python or Julia can be used for:

- Signal processing (e.g., EEG, EMG, fMRI data).
- Machine learning models for pattern recognition or anomaly detection.
- Real-time data visualization and dashboards.

These can be packaged into apps or services and deployed via Intune/SCCM to target devices.

⚙️ Low-Code/No-Code Integration

Microsoft Power Platform (Power Apps, Power Automate, Power BI) allows to:

- Build dashboards for clinicians or researchers to view brain activity in real time.
- Automate workflows, like triggering alerts when abnormal patterns are detected.

- Integrate with APIs from Python-based services or Azure ML models.
- Create mobile-friendly apps for field use or patient interaction.

For example:

- A Power App could display real-time EEG data streamed from a Python backend.
- Power Automate could trigger Teams alerts or emails when seizure-like patterns are detected.

🔄 Bridging the Worlds

There is a possibility to bridge Python/Julia with Power Platform using:

- Azure Functions or Logic Apps to expose Python models as APIs.
- Dataverse or Azure SQL to store processed data.
- Custom connectors in Power Apps to call my backend services.

Limitations:

Integrating AI, ML, and QEEG with graphene electrodes for medical applications like diagnosing migraines and strokes is promising, but it comes with several limitations:

1. Data Quality and Availability

- **Data Quality**: High-quality, labeled data is essential for training accurate AI/ML models. Inconsistent or noisy data can lead to poor model performance.
- **Data Availability**: Access to comprehensive datasets that include diverse patient populations is often limited, which can affect the generalizability of the models.

2. Model Interpretability

- **Black Box Models**: Many AI/ML models, especially deep learning models, are often considered "black boxes" because their decision-making processes are not easily interpretable. This lack of transparency can be a barrier to clinical adoption.

- **Clinical Trust**: Clinicians need to trust and understand AI recommendations to effectively integrate them into their practice. Lack of interpretability can hinder this trust

3. Bias and Fairness

- **Bias in Data**: AI models can inherit biases present in the training data, leading to unfair or inaccurate predictions for certain groups. This is particularly concerning in healthcare, where biased models can exacerbate health disparities.
- **Fairness**: Ensuring that AI models are fair and do not discriminate against any group is a significant challenge

4. Integration with Existing Systems

- **Interoperability**: Many healthcare systems use legacy electronic health record (EHR) systems that are not designed for AI integration. This can make it difficult to implement AI solutions seamlessly.
- **Data Fragmentation**: Patient data is often siloed across different departments and institutions, complicating the integration of comprehensive datasets needed for accurate AI predictions.

Regulatory and Ethical Considerations

- **Regulation**: The regulatory landscape for AI in healthcare is still evolving. Ensuring compliance with regulations while fostering innovation is a delicate balance.
- **Ethical Concerns**: Issues such as patient consent, data privacy, and the potential misuse of AI-generated insights need to be carefully managed.

6. Technical Challenges

- **Real-time Processing**: Analyzing QEEG data and making real-time adjustments to DBS parameters require significant computational resources and efficient algorithms
- **Graphene Integration**: While graphene has excellent properties, integrating it into medical devices and ensuring its long-term stability and biocompatibility is challenging

Clinical Validation

- **Clinical Trials**: Robust clinical trials are necessary to validate the efficacy and safety of AI-integrated medical devices. These trials can be time-consuming and expensive
- **Evidence Generation**: Generating evidence that demonstrates real clinical applicability and benefits to patient outcomes is crucial for widespread adoption.

Addressing these limitations requires a multidisciplinary approach, involving collaboration between AI researchers, clinicians, regulatory bodies, and ethicists. Continuous improvement and rigorous validation are essential to ensure that these technologies can be safely and effectively integrated into clinical practice.

The integration of AI in healthcare brings significant benefits but also raises several ethical concerns that need careful consideration:

1. Data Privacy and Consent

- **Patient Privacy**: AI systems often require large amounts of personal health data, raising concerns about how this data is stored, accessed, and shared.

  Ensuring patient privacy and confidentiality is paramount.
- **Informed Consent**: Patients must be fully informed about how their data will be used by AI systems, including potential risks and benefits.

  As one of the solution to address data privacy and compliance management can be

  MS Purview with MS Priva, as reference.

2. Bias and Fairness

- **Algorithmic Bias**: AI models can inherit biases present in the training data, leading to unfair or inaccurate predictions for certain groups.

  This can exacerbate existing health disparities.
- **Equitable Access**: Ensuring that AI-driven healthcare solutions are accessible to all populations, including marginalized and underserved communities, is crucial.

3. Transparency and Accountability

- **Black Box Models**: Many AI systems, especially deep learning models, are not easily interpretable, making it difficult to understand how decisions are made.

  . This lack of transparency can hinder trust and accountability.

- **Responsibility**: Determining who is responsible for AI-driven decisions, especially in cases of errors or adverse outcomes, is a complex issue.

4. Patient Safety

- **Clinical Validation**: AI systems must be rigorously tested and validated to ensure they are safe and effective in clinical settings.

  This includes continuous monitoring and updating of algorithms.

- **Risk Management**: Identifying and mitigating risks associated with AI applications in healthcare is essential to protect patient safety.

5. Ethical AI Development

- **Ethical Guidelines**: Developing and adhering to ethical guidelines for AI in healthcare is necessary to ensure responsible use.

  This includes considerations for fairness, transparency, and accountability.

- **Stakeholder Involvement**: Engaging various stakeholders, including patients, healthcare providers, and ethicists, in the development and deployment of AI systems can help address ethical concerns.

6. Impact on Healthcare Professionals

- **Role of Physicians**: AI can change the role of healthcare professionals, potentially reducing their autonomy and altering the patient-provider relationship.

  Ensuring that AI complements rather than replaces human judgment is important.

- **Training and Education**: Healthcare professionals need to be trained to work effectively with AI systems, understanding their capabilities and limitations.

Addressing these ethical concerns requires a multidisciplinary approach, involving collaboration between AI researchers, clinicians, regulatory bodies, and ethicists. Continuous improvement and rigorous validation are essential to ensure that AI technologies can be safely and effectively integrated into clinical practice.

What is next step?

Recommendations from PhD Thesis: Artificial Intelligence with Charge Transfer in Deep Brain Stimulation

Introduction: The integration of Artificial Intelligence (AI), Machine Learning (ML), Large Language Models (LLM), Generative AI, and Cybersecurity with charge transfer and the Vojta method in Deep Brain Stimulation (DBS) presents a promising approach to enhancing the diagnosis and treatment of brain diseases. This thesis explores the application of these technologies to improve treatments for neurological disorders, migraines with aura, and the recovery of preterm infants and seniors.

Recommendations:

1. **Integration of AI, ML, and Advanced Algorithms**:
   - **Data Collection and Analysis**: Utilization of AI and ML algorithms for the collection and analysis of patient data, including neurological signals, motor patterns, and treatment outcomes. This data can be employed to train models that predict treatment efficacy and personalize therapy.
   - **Predictive Modeling**: Develop predictive models to identify optimal stimulation parameters for DBS and forecast the progression of neurological conditions.
   - **Real-time Monitoring**: Implement real-time monitoring systems that employ AI to adjust stimulation parameters dynamically based on patient responses.
2. **Vojta Method Integration**:
   - **Reflex Locomotion Activation**: Utilize AI to analyze and optimize the activation of reflex locomotion patterns in patients undergoing Vojta therapy. This can assist in fine-tuning the pressure points and stimulation zones for better outcomes.

- **Patient Monitoring**: Implement sensors and AI algorithms to monitor patient movements and provide feedback to therapists, ensuring precise application of the Vojta method.

3. **Charge Transfer in Deep Brain Stimulation Science and Nanotechnology Diodes**:
    - **Electrode Design**: Utilize advanced algorithms to design electrodes that maximize charge transfer efficiency and minimize tissue damage.
    - **Graphene and Graphene Diodes**: Explore the usage of graphene and graphene diodes for enhancing deep brain stimulation and neuromodulation techniques.
4. **Neurological Rehabilitation**:
    - **Personalized Therapy**: Develop personalized rehabilitation programs using AI to tailor exercises and stimulation protocols based on patient progress and specific needs.
    - **Feedback Systems**: Implement feedback systems that use sensors and AI to track patient progress and adjust therapy in real-time.
5. **Migraine Treatment**:
    - **Neuromodulation Devices**: Utilize AI to optimize the settings of neuromodulation devices like transcutaneous vagus nerve stimulators and transcranial magnetic stimulation devices for migraine treatment.
    - **Patient Data Analysis**: Analyze patient data to identify patterns and triggers for migraines, allowing for more targeted and effective treatments.
6. **Recovery of Preterm Infants and Seniors**:
    - **Vital Signs Monitoring**: Utilize AI to monitor vital signs and neurological signals in preterm infants and seniors, providing early detection of potential issues.
    - **Therapeutic Interventions**: Develop AI-driven therapeutic interventions that can be adjusted based on real-time data to support recovery and development.

Process of Implementation Steps:

- **Data Infrastructure**: Establish a robust data infrastructure to collect, store, and process patient data securely.
- **Algorithm Development**: Develop and train AI and ML algorithms using historical and real-time patient data.
- **Device Integration**: Integrate AI and ML algorithms with medical devices and sensors to enable real-time monitoring and adjustments.
- **Clinical Trials**: Conduct clinical trials to validate the efficacy and safety of the integrated solutions.
- **Regulatory Compliance**: Ensure all solutions comply with medical regulations and standards.

Graphene-Based Nasal Electrodes for Neural Stimulation and Diagnostic Monitoring Enhanced by AI and Nanotechnology

In recent years, the convergence of nanotechnology, artificial intelligence (AI), and biomedical engineering has enabled the development of highly sophisticated neurotechnological devices. This chapter presents the conceptualization and design of a novel electrode system engineered for insertion via the nasal cavity, with the dual purpose of stimulating cerebral activity and collecting physiological data for diagnostic evaluation. The system is further enhanced by the integration of AI-driven analytics, generative models, and neurorehabilitation principles such as the Vojta method.

The nasal route provides a minimally invasive access point to the brain's frontal regions, including the olfactory bulb and prefrontal cortex. The proposed electrode system utilizes this anatomical advantage to deliver targeted electrical stimulation aimed at modulating neural activity and enhancing cerebral blood flow. Improved perfusion is expected to increase oxygen delivery to critical brain regions, offering therapeutic potential for conditions such as stroke, neurodegenerative diseases, and cognitive decline.

Simultaneously, the electrodes are equipped with biosensors capable of capturing electrophysiological signals, hemodynamic responses, and biochemical markers. These data streams are processed in real time using machine learning (ML) algorithms and large language models (LLMs), which enable adaptive signal interpretation, anomaly detection, and personalized therapeutic feedback. Generative AI models are employed to simulate neural responses and optimize stimulation protocols based on individual patient profiles, thereby enhancing the precision and efficacy of the intervention.

At the core of the device are graphene-based diodes, chosen for their exceptional electrical conductivity, mechanical flexibility, and surface adaptability. These diodes facilitate efficient charge transfer across the graphene-semiconductor interface, a process modulated by shifts in the Fermi level of graphene. External electrical stimuli can dynamically influence this Fermi level, allowing for fine-tuned control of neural stimulation. The use of nanotechnology in the fabrication of these components ensures high-resolution interfacing with biological tissues while maintaining structural integrity and biocompatibility.

The biocompatibility of graphene remains a critical consideration. While graphene exhibits promising properties for biomedical applications, its interaction with biological tissues is influenced by factors such as particle size, surface chemistry, and exposure route. Studies have shown that unmodified graphene may induce oxidative stress and inflammatory responses. However, functionalization techniques—such as polymer coatings and bioactive surface modifications—have been shown to mitigate these effects and enhance compatibility with neural and mucosal tissues.

To further support neurorehabilitation, the system incorporates principles from the Vojta method, a therapeutic approach that activates innate movement patterns through specific pressure points. By integrating stimulation protocols aligned with Vojta zones, the device may facilitate neuroplasticity and motor recovery in patients with neurological impairments. AI algorithms analyze patient responses to stimulation and adjust the protocol dynamically, creating a closed-loop system that mirrors the adaptive nature of traditional Vojta therapy.

In conclusion, the proposed nasal-inserted electrode system represents a multidisciplinary innovation at the intersection of nanotechnology, AI, and neurorehabilitation. The integration of graphene-based electronics, real-time AI analytics, and therapeutic frameworks such as the Vojta method offers a comprehensive platform for non-invasive brain stimulation and diagnostic monitoring. Future research will focus on in vivo validation, long-term safety assessments, and the development of personalized stimulation algorithms to fully realize the clinical potential of this technology.

By following the recommendations of my PhD thesis, there exists an opportunity to develop a comprehensive and effective approach to enhancing the diagnosis and treatment of brain diseases using AI, ML, graphene diodes, and charge transfer mechanisms. This initiative is currently underway, with the hope of early introduction to ethical medical centres, as these are my primary interests in collaborative efforts. I testify my solution to those who protect human lives. This interdisciplinary approach holds significant promise for advancing the field of neurological treatment and improving patient outcomes.

References

- **Agnieszka Mietz, Evolution of uncertainty of measurements for Accreditation Processes in Medical Analytical Research Laboratories, "Young Commodity Scientists", Poznań University of Economics, 2006.**
- **Agnieszka Mietz-Blijleven – Years of Author's Experience and Self-Driven Observations on Brain Recovery and Mitigation of Migraine with Aura**
- **Medical examination introduced in documenatation of Author's twins (above 4000 pages)**
- **Agnieszka Mietz-Blijleven – Radiology of Brain before and after sitimulation for Migrene with Aura**
- Health Equity and Ethical Considerations in Using Artificial Intelligence in Public Health and Medicine
- Ethical Dimensions of Using Artificial Intelligence in Health Care
- The ethical issues of the application of artificial intelligence in healthcare
- The ethical quandaries of AI in healthcare - EthicAI
- **1.F. Bonaccorso, Z. Sun, T. Hasan, and A. C. Ferrari, Nat. Photonics 4, 611 (2010).https://doi.org/10.1038/nphoton.2010.186**
- **Google ScholarCrossref**

  2.C. Chen, M. Aykol, C. Chang, A. F. J. Levi, and S. B. Cronin, Nano Lett. 11, 1863 (2011).https://doi.org/10.1021/nl104364c
- **Google ScholarCrossref PubMed**

  3.S. Tongay, T. Schumann, X. Miao, B. R. Appleton, and A. F. Hebard, Carbon 49, 2033 (2011).https://doi.org/10.1016/j.carbon.2011.01.029
- **Google ScholarCrossref**

  4.K. Chung, C. H. Lee, and G. C. Yi, Science 330, 655 (2010).https://doi.org/10.1126/science.1195403
- **Google ScholarCrossref PubMed**

  5.G. Jo et al., Nanotechnology 21, 175201 (2010).https://doi.org/10.1088/0957-4484/21/17/175201
- **Google ScholarCrossref PubMed**

6.T. Sun, Z. L. Wang, Z. J. Shi, G. Z. Ran, W. J. Xu, Z. Y. Wang, Y. Z. Li, L. Dai, and G. G. Qin, Appl. Phys. Lett. 96, 133301 (2010).https://doi.org/10.1063/1.3373855

- Google ScholarCrossref

7.S. Tongay, T. Schumann, and A. F. Hebard, Appl. Phys. Lett. 95, 222103 (2009).https://doi.org/10.1063/1.3268788

- Google ScholarCrossref

8.X. M. Li et al., Adv. Mater. 22, 2743 (2010).https://doi.org/10.1002/adma.200904383

- Google ScholarCrossref PubMed

9.K. Ihm et al., Appl. Phys. Lett. 97, 32113 (2010).https://doi.org/10.1063/1.3464319

- Google ScholarCrossref

10.Y. Gao, H. Yip, S. K. Hau, K. M. O'Malley, N. C. Cho, H. Chen, and A. K. Y. Jen, Appl. Phys. Lett. 97, 203306 (2010).https://doi.org/10.1063/1.3507388

- Google ScholarCrossref

11.K. T. He, J. C. Koepke, S. Barraza-Lopez, and J. W. Lyding, Nano Lett. 10, 3446 (2010).https://doi.org/10.1021/nl101527e

- Google ScholarCrossref PubMed

12.L. Magaud, F. Hiebel, F. Varchon, P. Mallet, and J. Y. Veuillen, Phys. Rev. B 79, 161405 (2009).https://doi.org/10.1103/PhysRevB.79.161405

- Google ScholarCrossref

13.A. Mattausch and O. Pankratov, Phys. Rev. Lett. 99, 76802 (2007).https://doi.org/10.1103/PhysRevLett.99.076802

- Google ScholarCrossref

14.S. Y. Zhou, G. H. Gweon, A. V. Fedorov, P. N. First, W. A. De Heer, D. H. Lee, F. Guinea, A. Neto, and A. Lanzara, Nature Mater. 6, 770 (2007).https://doi.org/10.1038/nmat2003

- Google ScholarCrossref

15.H. Zhong, Z. Liu, G. Xu, Y. Fan, J. Wang, X. Zhang, L. Liu, K. Xu, and H. Yang, Appl. Phys. Lett. 100, 122108 (2012).https://doi.org/10.1063/1.3696671

- Google ScholarCrossref

16.W. Zhao, P. H. Tan, J. Liu, and A. C. Ferrari, J. Am. Chem. Soc. 133, 5941 (2011).https://doi.org/10.1021/ja110939a

- **Google ScholarCrossref PubMed**

17.H. Hibino, H. Kageshima, M. Kotsugi, F. Maeda, F. Z. Guo, and Y. Watanabe, Phys. Rev. B 79, 125437 (2009).https://doi.org/10.1103/PhysRevB.79.125437

- **Google ScholarCrossref**

18.A. H. Castro Neto, F. Guinea, N. Peres, K. S. Novoselov, and A. K. Geim, Rev. Mod. Phys. 81, 109 (2009).https://doi.org/10.1103/RevModPhys.81.109

- **Google ScholarCrossref**

19.G. Giovannetti, P. A. Khomyakov, G. Brocks, V. M. Karpan, J. van den Brink, and P. J. Kelly, Phys. Rev. Lett. 101, 26803 (2008).https://doi.org/10.1103/PhysRevLett.101.026803

- **Google ScholarCrossref**

20.S. M. Sze and K. K. Ng, Physics of Semiconductor Devices, 3rd ed. (Wiley, Hoboken, 2007).

- **Google Scholar**

21.K. S. Kim et al., Nature 457, 706 (2009).https://doi.org/10.1038/nature07719

- **Google ScholarCrossref PubMed**

22.K. S. Novoselov, D. Jiang, F. Schedin, T. J. Booth, V. V. Khotkevich, S. V. Morozov, and A. K. Geim, Proc. Natl. Acad. Sci. U.S.A. 102, 10451 (2005).https://doi.org/10.1073/pnas.0502848102

- **Google ScholarCrossref PubMed**

23.C. H. Lui, L. Liu, K. F. Mak, G. W. Flynn, and T. F. Heinz, Nature 462, 339 (2009).https://doi.org/10.1038/nature08569

- **Google ScholarCrossref PubMed**

24.A. C. Ferrari et al., Phys. Rev. Lett. 97, 187401 (2006).https://doi.org/10.1103/PhysRevLett.97.187401

- **Google ScholarCrossref PubMed**

25.C. R. Dean et al., Nat. Nanotechnol. 5, 722 (2010).https://doi.org/10.1038/nnano.2010.172

- **Google ScholarCrossref PubMed**

26.E. Lee, K. Balasubramanian, R. T. Weitz, M. Burghard, and K. Kern, Nat. Nanotechnol. 3, 486 (2008).https://doi.org/10.1038/nnano.2008.172

- **Google ScholarCrossref PubMed**

27.K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, Y. Zhang, S. V. Dubonos, I. V. Grigorieva, and A. A. Firsov, Science 306, 666 (2004).https://doi.org/10.1126/science.1102896

- **Google ScholarCrossref PubMed**

28.J. B. Pethica and W. C. Oliver, Phys. Scr. T 19A, 61 (1987).https://doi.org/10.1088/0031-8949/1987/T19A/010

- **Google ScholarCrossref**

29.R. Nowak, M. Pessa, M. Suganuma, M. Leszczynski, I. Grzegory, S. Porowski, and F. Yoshida, Appl. Phys. Lett. 75, 2070 (1999).https://doi.org/10.1063/1.124919 Google ScholarCrossref

- First-principles study of the interaction and charge transfer between ...
- Charge transport mechanisms of graphene/semiconductor Schottky barriers ...
- Direct Electrical Stimulation in Electrocorticographic Brain–Computer ...
- Modeling Deep Brain Stimulation Incorporating Electrode-Tissue ...
- EFFECTS OF CAPACITIVE AND RESISTIVE ELECTRONIC TRANSFER THERAPY IN ...
- 8 Methods Physical Therapists Use to Break Up Muscle Tension
- A charge analysis of non-invasive electrical brain stimulation
- Engineering high charge transfer n-doping of graphene electrodes and ...
- PEMF vs Grounding Therapy: Benefits, Comparison, and Synergistic ...
- Microsoft Learn webpages: Training | Microsoft Learn

List of tables and figures

Figure 1. An anatomical image of the torso and shoulder girdle with muscle tensions leading to migraine with aura, including cervical lordosis reduction with a widow's hump… Page 17

Figure 2 Revolutionizing Brain Stimulation with Graphene Diodes, AI, ML, LLM, Vojta method…………………………………………………………………………Page 16

Figure 3 (A) Enlarged brain spaces visible as round, dark changes (arrow) in the left temporal lobe in the case of migraine with aura. (B) Asymmetry in the appearance of cortical vessels. (RSNA & Wilson Xu)………………………………………………………… Page 39

Figure 4 Brain with stroke……………………………………………………………..Page 40

Figure 5 Critical points of anatomical stimulation with use of Vojta method………Page 40

Figure 6 Advanced algorithms for DBS……………………………………………………Page 42

Figure 7 Python algorithm that integrates a Large Language Model (LLM) and Graph API along with QEEG (quantitative EEG) analysis…………………………………………Page 45

Figure 8 Strategy of Generative Modelling with R……………………………………...Page 47

Figure 9 MASS model with R and LLM……………………………………………...Page 48

Figure 10 An image illustrating charge transfer and deep brain stimulation with graphene diodes……………………………………………………………………………Page 51

Figure 11 Graphene diodes with charge transfer, AI, genAI, ML, LLM in DBS…Page 52

Figure 12 A series of screenshots from the Jupyter environment, showcasing the Python BinaryClassifier with the MASS model, adjusted for the purpose of training learning models using TensorFlow………………………………………………………………Page 59

Figure 13 Stimulation points of the body using the Vojta method affecting feedback and response of the central nervous system in the healing process……………………..Page 69

Figure 14 Transition of charge transfer mechanism………………………………..Page73

Figure 15 The Knowledge Base-Augmented Language Model (KBLaM) process with augmenting large language models (LLMs) and external knowledge………………..Page 75

Figure 16: Microsoft KBLaM allows for attention over the entire knowledge base instead of having an external retriever…………………………………………………………Page 76

Figure 17: Source: Microsoft. By having the user's question attend to the knowledge base, while treating facts in the knowledge base independently, KBLaM scales efficiently and linearly with the size of the knowledge base……………………………………………Page 76

Figure 18: Source Microsoft, Data quality process……………………………………Page 78

Figure 19: Source: Microsoft, Data governance………………………………………Page 79

Figure 20 Data analysis and incorporated at Jupiter for ML model training………..Page 79

Figure 21 Source: Microsoft: RAG, CosmosDB, Azure OpenAI for MongoDB vCore in ML training process……………………………………………………………………...Page 79

Figure 22 Intuition behind PCA researches…………………………………………..Page 81

Figure 23 Modern Applied Statistics with S by Venables and Ripley…………………Page 82

Figure 24 A simplified Python algorithm that outlines the structure for analyzing the four phases: Time before aura attempts, Migraine Phase, Microstroke, and Stroke……..Page 83

Figure 25 Microsoft Azure AI, OpenAI services platform……………………………Page 94

Figure 26 Governance of responsible AI………………………………………………Page 96

Figure 27 Microsoft Azure Cognitive Services………………………………………..Page 97

Figure 27 Reference to source code of self-developed application with neural network integrated with Graphene diodes……………………………………………………Page 98

Figure 28 Results driven test environment setup……………………………………..Page 102

Figure 29 Results driven test environment setup……………………………………..Page 102

Figure 30 Source: MLflow for gen AI agent and ML model lifecycle - Azure Databricks | Microsoft Learn ………………………………………………………………….Page 103

Figure 31 Azure Databrick with ML study of results verification and validation…..Page 105

Figure 32 A set of screenshots illustrating the use of Deep Learning in the diagnosis of brain disease…………………………………………………………………………Page 132

Figure 33 Training and validation loss – DBS model……………………………..Page 141

Figure 34 Labeling of data with bias and weights…………………………………..Page 141

Figure 35 Microsoft Azure AI Foundry Platform…………………………………..Page 143

Figure 36 Microsoft Azure AI Foundry ecosystem………………………………...Page 144

Figure 37 Generative AI application development in Azure AI Foundry – environment set up for results validation and interpretation……………………………………...Page 145

Figure 38 AI Azure Foundry SDK…………………………………………………Page 146

Figure 39 Intelligent AI apps creation with AI Azure Foundry…………………...Page 147

Figure 40 Microsoft Azure OpenAI models for DBS……………………………...Page 150

Figure 41 Data feeding mechanism with AI Foundry platform…………………….…Page 151

Figure 42 Fine-tune a model by training – data analysis from graphene diodes and measurements of charge transfer…………………………………………………….Page 152

Figure 43 Deployment of a model and application with command line interface….Page 154

Figure 44 MASS with R and Azure AI Foundry results analysis…………………….Page 156

Figure 45 Raman spectra of graphene sample before and after HNO3 treatment in DBS…………………………………………………………………………………Page 159

Figure 46 Source: Understanding noninvasive charge transfer doping of graphene: a comparative study | Journal of Materials Science: Materials in Electronics……….Page 161

Figure 48 MS Fabric with Data Lakehouse for ML and results validation…………Page 163

Figure 49 Model prediction conforming results with MLFlow in Python or Julia SW language………………………………………………………………………Page 164

Figure 50 Healing process of brains of my twins – state in past during Vojta rehabilitation and pharmacological treatment and brain conductivity with charge transfer method…………………………………………………………………...Page 169

Figure 51 Application developed to integrate data collection from graphene diodes with endpoint management (for example with firewall, MS Endpoint Manager or Configuration Manager of Microsoft, or AirWatch for data protection, routing, collection, storage, transfer, loading mechanism…………………………………………………………Page 181

Figure 52 Nexthink analyzer for underpinned devices with regards to DBS data storage, analyzes, transformation compromised with digitalization and end user experience – Patient data availability on demand with responsible AI of its analysis…………………….Page 196

Table 1. Factors that may cause stroke and migraine with aura page 20

Table 2 Factors that may cause stroke and migraine with aura in premature infants and seniors page 26

Table 3 Optimizing Treatment of Brain Strokes in Seniors by Comparing Pathophysiology of Strokes in Infants and Migraines with Aura Using Advanced Techniques. Page 29

Table 4 Methods of brain stimulation summary Page 154

Table 5 DBS methods Page 168