Міністерство освіти і науки України

Київський національний університет імені Тараса Шевченка

ННІ «Інститут геології»

Кафедра геоінформатики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Спеціальність 103 – Науки про Землю

освітня програма «Геологія»

блок «Геоінформатика»

ТЕМА: «Використання нейронних мереж для літологічного розчленування свердловин»

Виконав студент 4-го курсу

Кафедри Геоінформатика

Білик Ігор Олегович

Науковий керівник Доцент

Кандидат фізико-математичних наук

Демидов Всеволод Кирилович

Робота рекомендується до захисту (протокол № 14 засідання кафедри геоінформатики від 13.06.23р.)

Завідувач кафедри Професор

Доктор технічних наук

Зацерковний Віталій Іванович

Київ-2023

РЕФЕРАТ

Актуальність теми

Літологічне розчленування свердловині є критичним фактором в нафтовій інженерії та впливає на усі етапи видобутку нафти, включаючи характеристику пласта, оцінку запасів, геологічні дослідження, моделювання колекторів, покращені процеси видобутку та планування свердловин, включаючи буріння та управління завершенням. Точне визначення літології на визначеній глибині, зокрема в гетерогенних карбонатних колекторах, є необхідним для ефективного прийняття рішень в нафтотехніці. Використання машинного навчання в прогнозуванні літології та геологічних дослідженнях допоможе ефективніше вирішувати ці питання.

Мета роботи

У роботі автор має за мету створити модель класифікатора, яка може використовувати дані методу ГДС для аналізу геологічного розчленування свердловини та визначення літологічного розрізу. Для цього буде використано штучну нейронну мережу, із врахуванням специфіки досліджуваних даних. Для підтвердження результатів буде проведено експериментальне навчання класифікатора з детальним налаштуванням параметрів.

Розв’язувані в роботі задачі

- Створення штучної нейронної мережі, підбір гіперпараметрів
- Створення моделі класифікатора
- Побудова літологічної колонки на основі моделі класифікатора

Основна ідея

Ідея полягає в обробці геофізичної інформації та виявленні патернів, які можуть бути використані для класифікації з використанням методів машинного навчання для автоматизації процесу.

Основні результати

Результатом роботи є модель класифікатора який здатний проводити класифікацію геофізичних даних, виділяти літологію на основі тренувального набору даних.

Оригінальність роботи

Новизна роботи полягає у створенні моделі, на основі поєднання алгоритмів штучних нейронних мереж, для геологічного розчленування свердловин, які були пробурені у Північному морі.

Практичне значення роботи

Практичне значення полягає у створенні моделі нейромережі для автоматизації виділення літологічних шарів гірських порід, які розмежовує свердловина.

Анотація

У дипломній роботі представлено дослідження моделі, створеної з використанням штучних нейронних мереж для геологічної класифікації. Робота ведеться за допомогою емпірично-експериментального підходу, типового для машинного навчання. Дослідження проведено на даних геофізичного дослідження свердловин із комплексом методів, що був здійснений в більш ніж двадцяти свердловинах поблизу узбережжя Північного моря. Розглянутий регіон у роботі характеризується покладами вуглеводневих, зокрема нафти. Для порівняння також проведено класифікацію за геофізичними даними що отримані на узбережжі Чорного моря.

Методи дослідження з яких складається комплекс описані у роботі для кращого порозуміння геофізичних властивостей залягаючих порід, які притаманні досліджуваному регіону.

Висновок представляє результати створення мультиперцептронної нейронної мережі для класифікації. Мережа має три приховані шари, в яких кожен містить 200 нейронів. Використовується функція активації ReLU. Оптимізація навчання здійснюється за допомогою методу "adam", з параметром "альфа" встановленим на 0,004, що вказує на швидкість навчання. Розмір пакету для навчання встановлено на 235 прикладів. Максимальна кількість ітерацій навчання обмежена 500 епохами. Дані перемішуються випадковим чином, і мережа не використовує ваги з попереднього навчання. Максимальна точність класифікації, отримана з цією архітектурою та параметрами, складає 86%, але вона може змінюватися в залежності від використовуваних наборів даних і їх особливостей.

Ключові слова

Штучна нейронна мережа, літологічне розчленування, машинне навчання, класифікація, каротаж, шутчний інтелект.

Abstract

The diploma thesis presents a study of a model created using artificial neural networks for geological classification. The work is carried out using an empirical-experimental approach, typical for machine learning. The research was conducted on data from geophysical well surveys with a set of methods carried out in more than twenty wells near the North Sea coast. The region under consideration in the work is characterized by hydrocarbon deposits, particularly oil. For comparison, classification was also carried out based on geophysical data obtained on the Black Sea coast. The research methods that make up the complex are described in the work for a better understanding of the geophysical properties of the underlying rocks, which are inherent in the region under study.

The conclusion presents the results of creating a multilayer perceptron neural network for classification. The network has three hidden layers, each containing 200 neurons. The activation function ReLU is used. Training optimization is performed using the "adam" method, with the "alpha" parameter set to 0.004, indicating the learning rate. The batch size for training is set to 235 examples. The maximum number of training iterations is limited to 500 epochs. The data are shuffled randomly, and the network does not use weights from previous learning. The maximum classification accuracy obtained with this architecture and parameters is 86%, but it can vary depending on the data sets used and their characteristics.

Key words

Neural network, lithology, machine learning, classification, logging.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГДС	Геофізичні дослідження свердловин
Ом*м	Ом на метр
ПВР	Прострілювально-вибухові роботи
CNN	Convolutional Neural Network
GR	Gamma Ray
LAS	Log ASCII Standard
ML	Machine Learning
MLP	Multi Perceptron
R	Resistivity log
DeltaPHI	Neutron-Density Porosity Difference
PE	Photoelectric Effect
PHIND	Average Neutron-Density Porosity

ВСТУП

За декілька десятків років, машинне навчання стало важливим і ефективним інструментом у вирішенні задач у багатьох сферах людського життя. У промисловості, інструменти машинного навчання почали активно використовуватися разом із наростанням обчислювальної потужності комп'ютерів, адже це стало високоефективною і невід'ємною складовою.

Алгоритми машинного навчання, вже більше двох десятиліть активно використовуються в геології, особливо у вирішенні геофізичних задач. Однією із таких задач є розчленування літологічного розрізу свердловини. Визначення літологічного типу певної породи, що базується на її петрофізичних властивостях, характеристика яких отримується за допомогою геофізичних досліджень. Нерідко висновки про визначення певної літофації робить людина, на основі емпіричних знань, або на основі зразків керну, які відбираються у процесі буріння. Такий підхід може забирати багато часу і нейронні мережі покликані прискорити, та допомогти у вирішення цієї задачі. *(J. Wang, 2017)*

Головною задачею дослідження , є літологічне розчленування свердловин, за даними методами ГДС, із використанням засобів машинного навчання, а саме алгоритму багатошарового перцептрону.

Вирішення поставленої задачі виконується за допомогою мови програмування Python, іх використанням таких пакетів та бібліотек.

Pandas: Це потужна бібліотека в Python для аналізу даних, часових рядів та статистики, яка використовує ефективні структури даних.

Matplotlib: Це бібліотека для створення двовимірних графіків в Python, яка дозволяє створювати якісні ілюстрації в різних форматах та інтерактивних середовищах на різних платформах.

Scipy: SciPy - це екосистема відкритого програмного забезпечення на основі Python для математики, науки та інженерії.

Numpy: Це основний пакет для наукових обчислень в Python. Він містить, серед іншого: потужний об'єкт масиву N-вимірного, складні (broadcasting) функції, інструменти для інтеграції коду C/C++ та Fortran, корисні засоби лінійної алгебри, перетворення Фур'є, та здатності генерації випадкових чисел.

Методами вирішення задачі є побудова моделі на основі алгоритму багатошарового перцептрону із розпізнаванням певних закономірностей на основі каротажних даних, для подальшої класифікації гірських порід.

Досліджувана територія, звідки були взяті дані для подальшого аналізу а саме геологічна будова західного узбережжя Норвегії та Північного моря, може бути описана як складне поєднання осадових і вулканічних порід, що утворилися протягом тривалого періоду часу. Згідно з даними Норвезького Нафтового Директорату (NPD), берегові райони західної Норвегії складаються в основному з докембрійських і палеозойських метаморфічних і магматичних порід, в той час як на шельфі переважають мезозойські і кайнозойські осадові породи. NPD також стверджує, що регіон Північного моря є областю «інтенсивної нафтової діяльності» і що «основними резервуарними породами є юрські і крейдяні пісковики». "Ділянки західного узбережжя Норвегії складні, характеризуються глибокими долинами, гребнями, і великими ділянками мілководдя." (NPD 2019)

У Північному морі підповерхневу геологію можна розділити на два основні регіони: Центральний Грабен і Грабен вікінгів. Область Центрального Грабена характеризується «товстими послідовностями пізнього юрського періоду до раннього крейдяного пісковика і мудастонів», в той час як регіон Вікінга складається в основному з «середнього юрського пісковика і вапняка». (NPD) Ця інформація надходить з Норвезького Нафтового Директорату (NPD), який є урядовим органом, відповідальним за управління нафтовими ресурсами Норвегії.

1. **МАШИННЕ НАВЧАННЯ В ГЕОФІЗИЦІ**

За останні роки, машинне навчання стало все популярнішою інструментальною технологією у геофізиці, яка дозволяє аналізувати великі та складні набори даних. Цей розділ має на меті дослідити поточний стан досягнень машинного навчання у геофізиці та їхні можливі застосування у цій галузі. Машинне навчання є галуззю штучного інтелекту, яка включає у себе розробку алгоритмів, які можуть навчатися на даних і робити прогнози або приймати рішення без явного програмування. Вона використовується у різних галузях, таких як розпізнавання зображень, обробка природної мови, тощо. У геофізиці, машинне навчання використовується для розв'язання завдань таких як інтерпретація сейсмічних даних, дослідження мінералів, картографування. Існують різні методи машинного навчання, які застосовуються у геофізиці, включаючи супервізоване навчання, несупервізоване навчання та глибоке навчання. Супервізоване навчання включає навчання моделі з використанням міток на даних, де мета полягає в прогнозуванні цільової змінної на основі набору вхідних ознак. Несупервізоване навчання ж займається пошуком закономірностей та структур у ненавчальних даних. Глибоке навчання є підмножиною машинного навчання, яке базується на нейронних мережах з кількома шарами, це особливо корисно в обробці зображень та сигналів. *(McCormack, K. D., Zaiane, O. R., & Goebel, R. 2012)*

Машинне навчання підходить для вирішення проблем там, де недостатньо теоретичних знань, але є значна кількість спостережень, і інших даних. Типовими прикладами ML є алгоритми штучних нейронних мереж, машини опорних векторів самоорганізаційна карта, дерева рішень, та багато інших.

Методи основані на машинному навчанні, уже близько 30 років широко використовуються для вирішення інженерних та наукових проблем, в той час як в геології ці методи порівняно нові. Не зважаючи на їх новизну, вони за короткий час стали необхідним інструментом в арсеналі будь-якого інженера-геофізика.

Прикладами використання цих методів, є аналіз сейсмічних даних за допомогою алгоритмів кластеризації, в науковій роботі *(Z. Fana , X. Xua, 2019)* пояснюється доцільність і практичність застосування алгоритмів кластерного аналізу.

Методи засновані на штучних нейронних мережах, застосовуються у таких задачах геофізики.

Наприклад:

- Класифікації сейсмічних подій
- Передбачення землетрусів
- Оцінці геофізичних параметрів
- Літологічному розчленуванні
- Моделюванні родовищ

Та у вирішенні багатьох інших задач галузей геофізики.

Стосовно проблематики машинного навчання, то виділяють декілька основних проблем з якими можна зіштовхнутися під час використання методів машинного навчання, це *(Sandham, W.,Leggett. M., 2003), (Aminzadeh, F., & Chatterjee, S., 2000)*:

- Недостатня кількість даних: Машинне навчання зазвичай потребує великої кількості даних для навчання моделі. Якщо недостатньо даних, модель може бути неправильно навчена і не мати достатньої точності для вирішення задачі.
- Неправильне вибір моделі: Існує велика кількість різних моделей машинного навчання, і вибір неправильної моделі може призвести до низької точності класифікації або регресії.
- Недостатня увага до якості даних: Іноді данні можуть бути недостатньо чистими або містити помилки, що можуть призвести до неправильного навчання моделі.
- Проблема перенавчання: Якщо модель занадто добре навчена на навчальних даних, вона може не правильно класифікувати нові дані, що називається

перенавчанням. Ця проблема може виникнути, якщо модель занадто складна або якщо навчальні дані не репрезентують всі можливі варіації вхідних даних.

- Проблема побудови зрозумілої моделі: Деякі моделі машинного навчання можуть бути дуже складними та складно інтерпретованими, що ускладнює розуміння того, як вони працюють та які фактори впливають на результат.

Використання нейронних мереж для літологічного розчленування свердловин за методами ГДС, вперше застосували на початку 2000х років *(Sandham, W.,Leggett. M., 2003), (Aminzadeh, F., & Chatterjee, S., 2000)* До того часу, літофації отримували за даними опису керну та каротажними даними, які інтерпретувала людина. Але алгоритми машинного навчання, допомогли автоматизувати процес літологічного розчленування. У 2019 році згідно роботи *(Zhili Wei, Hao Hu, Hua-Wei Zhou, And August Lau., 2019*), вчені із Університету Хьюстона, реалізували літологічне розчленування свердловин, за допомогою конволюційних нейронних мереж, що відносяться до глибокого навчання. Такі нейронні мережі застосовуються переважно для класифікації візуальних даних та природної мови, але знайшли своє використання і у геології.

Але якщо заглянути в історію нафто-газової галузі, і повернутися в XIX початок XX століття, коли люди працювали із даними лише емпіричним шляхом, коли такий підхід був повністю оснований на досвіді інженера, та на аналізі аналогічних випадків, то можна зробити висновок про високі та малоефективні затрати. В результаті такого підходу була занадто густа сітка свердловин розбурювання, що понесло за собою тяжкі наслідки для екології шляхом вирубування лісів, та забрудненням родючих шарів грунту. І як згодом з'ясувалося, з розвитком нафтової інженерії, така система розробки родовищ призводила до того, що видобувалося лише декілька відсотків нафти, від усього запасу родовища. І ділянки густо розбурених родовищ фактично були загубленими *(Z. Fana , X. Xua, 2019).*

З роками все дуже змінилося завдяки появі каротажу, та розробці нових і вдосконалення старих методів ГДС. У нас є емпіричні знання, безліч вимірів і їх

інтерпретація, зроблена людьми. Також у нас є фізичне моделювання, і зараз ми за допомогою нього намагаємося вибрати максимально оптимальний варіант розробки родовищ *(Zhao, T., 2018)*. Цей підхід схожий на науковий, здається дуже обгрунтованим, але це не зовсім так, тому що при побудові геологічних моделей інженери враховують непрямі вимірювання, які не дають прямої інформації про те, скільки нафти знаходиться далеко від свердловини і як легко її можна змістити в сторону свердловин за допомогою різних методів. Зараз з розвитком методів аналізу даних, в тому числі регресійного аналізу даних, є можливість зробити цей підхід суттєво більш науковим. Тому що методи машинного навчання дозволяють враховувати дані, які виходять з великою точністю, і з їх допомогою оптимізувати різні компоненти складних процесів видобутку нафти і газу.

Зокрема, дані, які ілюструють, скільки кожна свердловина віддає нафти, води, газу, можуть бути в оперативному режимі враховані, щоб оновити і уточнити геологічні та гідродинамічні моделі, які описують функціонування всього родовища, всього пласта. І над власне оптимізацією різних компонентів роботи з родовищами, починаючи з самого першого етапу, коли відбувається розвідка, сейсморозвідка, буріння так званих розвідувальних свердловин і їх дослідження, закінчуючи етапом, коли вже пласт повністю введений у видобуток і відбуваються процеси, управління виробництвом. Люди намагаються впроваджувати різні алгоритми, пов'язані із застосуванням методів машинного навчання. Зараз вся індустрія, яка має на увазі застосування методів передбачуваної аналітики для нафтової галузі, має великий потенціал *(Sun, H., Demanet, L., (2018)*.

Машинне навчання допомагає моделювати, за відсутності фізичної моделі. І це вже відбувається на американських сланцевих родовищах, коли за даними про профілі видобутку зі свердловин можна отримати уявлення про те, чого очікувати від нових свердловин і як оптимально ці нові свердловини розташувати (*Atanu Basu., 2014*).

У минулому десятилітті, компанія Schlumberger почала активно використовувати інструменти машинного навчання, для вирішення багатьох задач, від буріння до моделювання. Згодом, нейронні мережі тільки вдосконалювалися, про це свідчать роботи науковців, особливо цікавою та надзвичайно деталізованою роботою, є дослідження вчених із китайського університету нафти.

Згідно дослідження вченими із китайського університету *(Zhenyu Yuan, Yuxin Jiang, Jingjing Li , Handong Huang, 2020)*, вони застосували незвичайну архітектуру нейронної мережі, для вирішення поставленої задачі. Особливість полягає у подачі на вхід нейронної мережі, будь-якого типу даних, яким можуть бути відео, картинка, аудіо та текст, окрім цього варіація даних може бути від 1D до 2D, 3D і набагато більше. Дослідники іменують свою мережу як гібридна глибока нейронна мережа (HDNN Hybrid Deep Neural Network). Гібридна вона через комбінацію двох мереж, мультиперцептрон, і конволюційну мережу, що і дозволяє використовувати таку варіативність типу даних, на вхід мережі.

Взявши змішані дані з різних джерел в якості вхідних даних, вони розробили загальну архітектуру HDNN. Врахування змішаних даних використовує в повній мірі великі дані і підходить для виявлення більш точних взаємозв'язків між вимірюваннями та мітками.

Крім того, запропонована модель HDNN реалізує мітки до кінця навчання, уникаючи виснажливих робіт, таких як вилучення та вибір функцій активації. Загальні HDNN можуть розглядатися як сукупність декількох мережевих модулів. Конфігурація цих модулів залежить від цільового застосування і змінюється залежно від змішаних даних та цільових міток. Ця інноваційна архітектура забезпечує мережу гнучкістю та широким застосуванням. Принцип, що кілька факторів сприяють результату та наявності різноманітним вимірюванням збільшують спектр використання для різноманітних програм глибокого навчання.

Концентруючись на прогнозуванні видобутку вуглеводнів, модель HDNN приймає зображення, каротажні криві. Враховуються інженерні параметри, що

надає більше аспектів функцій, структурні зображення та криві містять більше деталей і підходять для визначення пластів для проведення таких задач як наприклад ПВР. Модель HDNN може застосовуватися для прогнозування виробництва, так як вміщує переваги двох мереж моделі MLP та моделі CNN.

Автори дослідження підкреслили переваги моделі HDNN над моделлю MLP або CNN, з кращою оптимізацією конвергенції та узагальненням виконання. Крім того, продуктивність моделей HDNN та CNN продемонструвала, що структурні журнали особливо підходять для оцінки продуктивності. Подальше застосування у трьох нових свердловинах для прогнозування виробництва на практиці підтвердили доцільність моделі HDNN. Він може бути використаний для прогнозування виробничого потенціалу цілі пластових або цільових свердловин і надалі керувати інженерними процесами при видобутку нафти.

Крім цього, можна згадати таких дослідників, як Андреас Рудер (Andreas Rüdiger) (A. Ruediger, F.Rosei Interfaces: AFM extends its reach Nature Nanotechnology, 5 (2010) 388.), який застосовував нейронні мережі для класифікації геофізичних даних, і Крістофер Ліндгрен (Christopher Liner) , який вивчає застосування машинного навчання для розпізнавання тектонічних структур на основі геофізичних даних (Liner, Christopher L.and McGilvery, Thomas, The Art and Science of Seismic Interpretation, 2017, Springer, c. 140 p (in preparation).

Досягнення цих дослідників полягають в тому, що вони застосовують методи машинного навчання для аналізу складних геофізичних даних, що дозволяє зробити більш точні прогнози та знайти залежності між різними параметрами. Наприклад, Георг Кліффорд застосував машинне навчання для моделювання розвитку нафтового родовища, що дало можливість зменшити час та витрати на розробку нафтових родовищ. Деніел Петро застосовував машинне навчання для прогнозування параметрів нафтогазових родовищ та оцінки їхнього потенціалу. Клаус Мосбауер використовував машинне навчання для розпізнавання аномалій в

гідрогеологічних даних, що дозволило покращити якість прогнозів щодо підземних вод.

Завданням дипломної роботи є дослідження можливостей використання методів машинного навчання для класифікації літологічних розрізів, отриманих з результатів геофізичних досліджень свердловин.

Для вирішення завдання класифікації літологічних розрізів буде використаний метод машинного навчання, такий як штучна нейронна мережа. Для цього будуть використані дані з геофізичних вимірювань, такі як електрорезистивність, гравітаційне поле, магнітне поле, індукція електричного струму та інші.

Основні завдання дипломної роботи включають збір та обробку даних з геофізичних вимірювань, розробку та порівняння різних методів машинного навчання для класифікації літологічних розрізів, побудову моделей та їх тестування на реальних даних, а також аналіз результатів та визначення їх точності. Очікується, що результати дослідження допоможуть в поліпшенні розуміння геологічної будови об'єкту та використовуватимуться у практичній геології, зокрема при пошуку корисних копалин.

2. **РОЗЧЛЕНУВАННЯ РОЗРІЗУ ЗА ДОПОМОГОЮ МЕТОДІВ ГДС**

2.1. Інтерпретація кривої ГДС класичним шляхом

Із технологічним розвитком багато чого змінюється і в методиці досліджень та інтерпретації. До прикладу, класична інтерпретація полягає в створенні літологічної колонки по свердловині базуючись на теоретичних знаннях геолога – інтерпретатора. Знання базуються на емпіричних дослідження, як і дослідження геофізичних полів у свердловині, так і відбір керну. Кореляція між керном та показниками методів дослідження лягає в основу створення літологічної колонки.

Гірські породи мають різні петрофізичні характеристики, при дослідженні одного фізичного поля даних для ідентифікації породи недостатньо, тому зазвичай проводять комплекс методів який підбирають в залежності від геологічного нарису та пошукових задач.

2.2. Електричні методи

Метод самочинної поляризації базується на вимірах електричних потенціалів, які виникають в гірських породах при їх розчленуванні свердловиною. Виникнення природного електричного поля в свердловині обумовлюється фізико-хімічними процесами, що відбуваються на границях між гірськими породами та буровим розчином і на границях між пластами.

Важливим чинником є такі фізичні процеси як: дифузія солей із пластових вод в буровий розчин і навпаки, адсорбція(поглинання) іонів розчинених солей породою. Фільтрація вод із бурового розчину в гірські породи і пластових вод – у свердловину. Окисно-відновні реакції, що відбуваються в гірських породах і на контакті порід з буровим розчином і металами.

Здатність гірських порід поляризуватися під впливом зазначених та інших природних фізико-хімічних процесів називається природною електрохімічною активністю (A). В результаті цих процесів виникають дифузійно-адсорбційні ($U_{\text{да}}$),

фільтраційні ($U_{\text{ф}}$) і окисно-відновні ($U_{\text{ов}}$) потенціали, інтенсивність і знак яких залежить головним чином від електрохімічної активності гірських порід (відповідно, від $A_{\text{да}}$, $A_{\text{ф}}$ та $A_{\text{ов}}$), відношення мінералізацій фільтрату бурового розчину ($\text{C}_{\text{ф}}$) та пластової води ($\text{C}_{\text{в}}$), різниці між гідростатичним тиском стовпу бурового розчину та тиском у пласті (гірським тиском) *(В. М. Курганський, І. В. Тішаєв, 2011).*

В основу методу покладене фізичне явище поляризації іонів солей, чому сприяють явища дифузії та адсорбції. Слід зазначити, що в такому методі велику роль буде відігравати промивальна рідина, її мінералізація. Класичним представленням і застосуванням цього методу є виділення границь пластів-колекторів.

Окрім методу поляризації, існує також досить популярний метод дослідження який оснований на вимірюванні питомого опору, рівняння (2.1) гірських порід ρ (Ом/м), тобто здатності гірських порід проводити електричний струм.

$$\rho = R \times \frac{S}{l} \tag{2.1}$$

ρ - це питомий опір матеріалу, вимірюється в омах-метрах (Ом/м).

R - це електричний опір матеріалу, вимірюється в омах (Ом).

S - це площа поперечного перерізу матеріалу, вимірюється в метрах квадратних (м²).

l - це довжина матеріалу, вимірюється в метрах (м).

Таким чином, формула ρ = R × S / l описує залежність між питомим опором матеріалу, електричним опором, площею поперечного перерізу та довжиною цього матеріалу.

І залежить головним чином від оберненої величини, -провідності рівняння (2.2) мінералів δ (См/м), які складають породу.

$$\delta = \frac{1}{\rho} \tag{2.2}$$

Але не тільки цей фактор впливає на провідність порід. Присутність мінералізованої води у порах, значно може зменшити питомий електричний опір. Найменший опір мають породи пелітової фракції із розмірами зерен від 0,005-0,01 мм. Збільшення фракції, також збільшує і електричний опір. Таке явище пояснюється гідролізом глинистих мінералів, які складають пелітову породу (глини). Утворені іони надають пелітовій породі додаткову поверхневу електропровідність, що відчутно проявляється під час насичення породи прісними водами. Зміни показів опору у породах, залежать від типу насичення. Це пов'язано з тим, що електропровідність порід залежить від їх хімічного складу та наявності у них іонів. Якщо пелітова порода містить іони, то ці іони можуть створювати додаткові електропровідні шляхи в породі, що збільшує загальну електропровідність породи. *(В. М. Курганський, І. В. Тішаєв, 2011)*

Крім того, при насиченні породи прісною водою, вода може розчиняти деякі мінерали у породі та виводити іони на поверхню породи. Це також може сприяти збільшенню електропровідності породи.

Зміни показів опору у породах, які залежать від типу насичення, можуть бути пов'язані з різними концентраціями іонів у воді, яка насичує породу, та з різними типами іонів. Наприклад, деякі іони можуть мати більшу здатність до провідності, ніж інші.

Отже, пелітові породи можуть мати додаткову поверхневу електропровідність, яка залежить від наявності іонів, а також від типу насичення породи прісною водою. Зміни показів опору у породах можуть відображати різні рівні концентрації та типи іонів у воді, яка насичує породу. Якщо порода-колектор насичена нафтою, то опір значно зросте.

Крива розділяє породи згідно їх електричних властивостей. Магматичні породи характеризуються високим опором, якщо вони не складені провідними мінералами (борніт, кобальтин, нікелін, пірит та ін.). Серед осадових порід низьким опором характеризуються глинисті породи, а високим опором – ущільнені породи, піски, вапняки, ущільнені пісковики, тобто є залежність і між пористістю породи. Зазвичай метод використовують у парі із методом СП, що називається стандартним каротажем. Результати стандартного каротажу дають загальне уявлення про геологічний розріз свердловини, полегшують кореляцію кривих.

2.3. Гамма-метод

Гамма-метод або метод природної радіоактивності оснований на природній радіоактивності порід. Радіоактивність порід спричинена розпадом радіоактивних елементів (радіонуклідів) уранового (^{238}U, період напіврозпаду $T_{\frac{1}{2}}$= 4,5-10 років, 16 ланок ряду), торієвого (Th, $T_{\frac{1}{2}}$= 13,9-109 років, 13 ланок) і актино- уранового рядів (AcU, $T_{\frac{1}{2}}$= 0,713-109 років, 15 ланок). Це так звані радіоактивні сімейства. Всі вони закінчуються утворенням стабільного хімічного елемента (свинцю). В середині кожного ряду утворюються радіоактивні гази (еманації): радон (Rn, $T_{\frac{1}{2}}$= 3,85 дня), торон (Tn, $T_{\frac{1}{2}}$ = 54 с) і актинон (An, $T_{\frac{1}{2}}$= 3,9 с). Багато досліджень та робіт були проведені в цій галузі, зокрема відомий геофізик Майкл Георгієвич Ляхов.

Природна радіоактивність гірських порід прямо пропорційна за вмістом в них зазначених радіоактивних елементів. Крім цього, встановлено, що осадові породи, утворені в різних умовах осадконакопичення, містять різні концентрації урану, торію і калію. Це служить петрофізичною основою якісного літологічного розчленування розрізу осадових порід за величиною їх природної радіоактивності.

Радіоактивність глинистих порід в порівнянні з іншими породами осадового комплексу пояснюється їх великою питомою поверхнею і здатністю до адсорбції радіоактивних елементів, тривалістю накопичення пелітового матеріалу, що

забезпечує збільшення вмісту ^{238}U, ^{232}Th, ^{40}K. Відома також здатність важких окислених нафт, в тому числі і асфальтоподібних органічних речовин, збагачуватися ураном за рахунок вилучення його з підземних вод. Легкі нафти і вугілля цією характеристикою не володіють. Таким чином, в осадових гірських породах максимальною радіоактивністю володіють глинисті породи, а радіоактивність більшості колекторів, представлених теригенними і карбонатними породами, сильно залежить від глинистості.

Гамма метод застосовують для вирішення наступних задач:

- розчленування геологічного розрізу на окремі пласти та пропластки (метод виділяє пропластки до 30-40см.)
- визначення та уточнення пластів-колекторів.
- визначення колекторських властивостей пластів.
- літологічне розчленування за природною радіоактивністю.
- кореляція пластів та пропластків.
- контроль ПВР.

2.4. Акустичний каротаж

Метод характеризується визначенням інтервального часу проходження повдовжньої хвилі. Інтервальний час проходження сейсмічної хвилі – Δt (2.3), залежить від текстур гірської породи, особливо помітно зменшується із збільшенням ефективної пористості. Швидкість обчислюється шляхом вимірювання часу проходження від передавача до приймача. Для компенсації коливань бурового розчину, є два приймачі, це пояснюється тим, що час подорожі в буровому розчині буде спільним для обох приймачів, тому інтервальний час всередині пласта визначається як різниця між приймачами

$$\Delta t = \Delta t_{far} - \Delta t_{near} \quad (2.3)$$

Де Δt_{far} – інтервальний час проходження сейсмічної хвилі від випромінювача до дальнього приймача,

Δt_{near} – інтервальний час проходження сейсмічної хвилі до ближнього приймача.

Покази методу залежать від пористості гірської породи. Якщо порода є щільною, тобто має низький коефіцієнт пористості, то хвилі будуть розповсюджуватися швидше, і інтервальний час Δt, буде меншим. Наприклад, швидкість поширення пружних хвиль у глині та гіпсі будуть зовсім різними, що спричинено пористістю породи. Так, ущільненим гірським породам таким як ущільнені пісковики, вапняки, кам'яна сіль будуть відповідати низькі покази кривої, а глині, глинистим сланцям і аргілітам – високі, проміжними значеннями характеризуються пісковики та алевроліти підвищеної пористості. *(В. Н. Косков, Б. В. Косков, 2007)*

Метод використовують з метою:

- літологічного розчленування розрізу свердловини
- виділення пластів колекторів
- визначення коефіцієнту пористості гірських порід

2.5. Метод корекції густини

Метод корекції густини, в Україні більше відомий як гамма-гамма каротаж густини, базується на вимірюванні кількості залишкових гамма-променів, які потрапляють на сцинтиляційний детектор. Радіоактивний елемент, який знаходиться у нижній частині свердловинного приладу, випромінює гамма-промені, які поглинаються навколишніми гірськими породами. Але не всі промені можуть бути поглинуті, деякі надходять до сцинтиляційних лічильників, які знаходяться у приладі на 0,4 та 0,6 метра вище за саме джерело випромінювання. Кількість гамма-променів, що надходять до дальнього детектору, обернено пропорційна електронній щільності гірської породи, яка в свою чергу є пропорційна фактичній

щільності породи. Дані із ближнього детектору, використовуються для корекції ефектів свердловини. Таким чином, за результатами методу визначають:

- щільність гірських порід;
- якість цементування обсадних колон;
- визначення інтервалів надходження, і типу флюїду, у експлуатаційних свердловинах *(W. Sandham, M. Leggett 2003).*

2.6. Нейтрон-нейтронний метод

Метод дослідження базується на реєстрації густини потоку теплових нейтронів у процесі опромінення свердловини швидкими нейтронами. Криві ННК розчленовують розріз свердловини за вмістом водню у гірських породах, наприклад найбільш водненасиченими є глинисті породи (глини, аргіліти, сланці та мергелі), тому що мають найбільшу пористість в осадовому розрізі, і відповідно покази будуть характеризуватися низькими значеннями Як правило, для таких порід є характерна кавернозність, що також спричиняє низькі покази кривої. Мінімальними показами характеризується гіпс там кам’яне вугілля, в силу їх високого водненасичення.

Найменшою пористістю характеризують щільні породи, до таких відносяться вапняки, пісковики, доломіти, відповідно до високої щільності, вони мають низьку водненасиченість. У з’вязку з цим, покази методу у них є найбільшими.

Нафтонасичені пласти-колектори належать до порід середньої пористості, на кривій відмічаються значно меншими показами у порівнянні із ущільненими породами. Газонасичені пласти-колектори вважаються середовищем не насиченим воднем, а тому вони відмічаються на кривій як ущільнені породи, і мають максимальні покази. Таке явище пояснюється меншим вмістом водню в газі ніж у нафті. *(Joseph R. Hearst, Philip H. Nelson, Frederick L. Paillet, 2012)*

Метод ННК застосовується у нафтових та газових свердловинах з метою визначення:

- характеру насичення пластів-колекторів;

- коефіцієнтів пористості;
- водонафтових і газорідинних контактів;
- літологічного розчленування свердловини.

2.7. Кавернометрія

Цей метод забезпечує безперервне вимірювання розміру та форми стовбуру свердловини вздовж її глибини. Сутність методу полягає у вимірюванні діаметру свердловини спеціальним пристроєм – каверноміром. Каверномір піднімається із забою стовбуру свердловини, та має дві або більше шарнірних кронштейнів, кожен з яких з'єднаний із потенціометром, який представляє зміну електричного опору із зміною діаметру стовбура свердловини. Таким чином створюється різний електричний сигнал, який представляє зміну форми свердловини. *(Darwin V. Ellis, Julian M. Singer, 2007)*

Отримані дані про діаметр свердловини використовуються для:

- деталізації геологічного розрізу;
- виділення пластів-колекторів;
- визначення об'єму затрубного простору.

3. ТЕОРЕТИЧНІ ОСНОВИ НЕЙРОННОЇ МЕРЕЖІ

3.1 Нейронна мережа як метод машинного навчання

Штучна нейронна мережа є моделлю, яка базується на концепції нейронних зв'язків та їх структурі в головному мозку. Вона складається з нейронів, які взаємодіють один з одним. Кожен нейрон отримує сигнал від попереднього, оцінює його і може реагувати або ні, в залежності від результату порогового значення. Штучні нейронні мережі розвиваються окремо від класичних методів і змінюють уявлення про машинне навчання і розпізнавання об'єктів. З часом, після розвитку базових моделей, напрямок нейромережі почав розвиватися, проявляючись у різних топологіях і методах навчання. *(Raul Rohas, 1996)*

Нейронна мережа складається з нейронів, які виступають як основні обчислювальні вузли, що отримують, обробляють і передають інформацію. Нейрони можуть бути вхідними, прихованими і вихідними, а також можуть мати додаткові функції, такі як нейрон зміщення та контекстний нейрон. Мережа може містити кілька шарів з нейронів (рис.3.1) , з яких кожен шар містить кілька нейронів того самого типу. Наприклад, вхідний шар містить нейрони, які отримують вхідні дані, прихований шар - нейрони, які виконують обчислення над цими даними, і вихідний шар - нейрони, які передають результат обчислення. Нейронна мережа може мати один або декілька прихованих шарів, що залежить від кількості та складності обчислення. Важливо зауважити, що кількість шарів і кількість нейронів в кожному шарі є ключовими факторами, які впливають на точність і ефективність нейронної мережі. Кількість нейронів в кожному шарі може впливати на моделі та навчання і збільшувати чи зменшувати ефективність мережі. *(Christopher Bishop, 2006)*

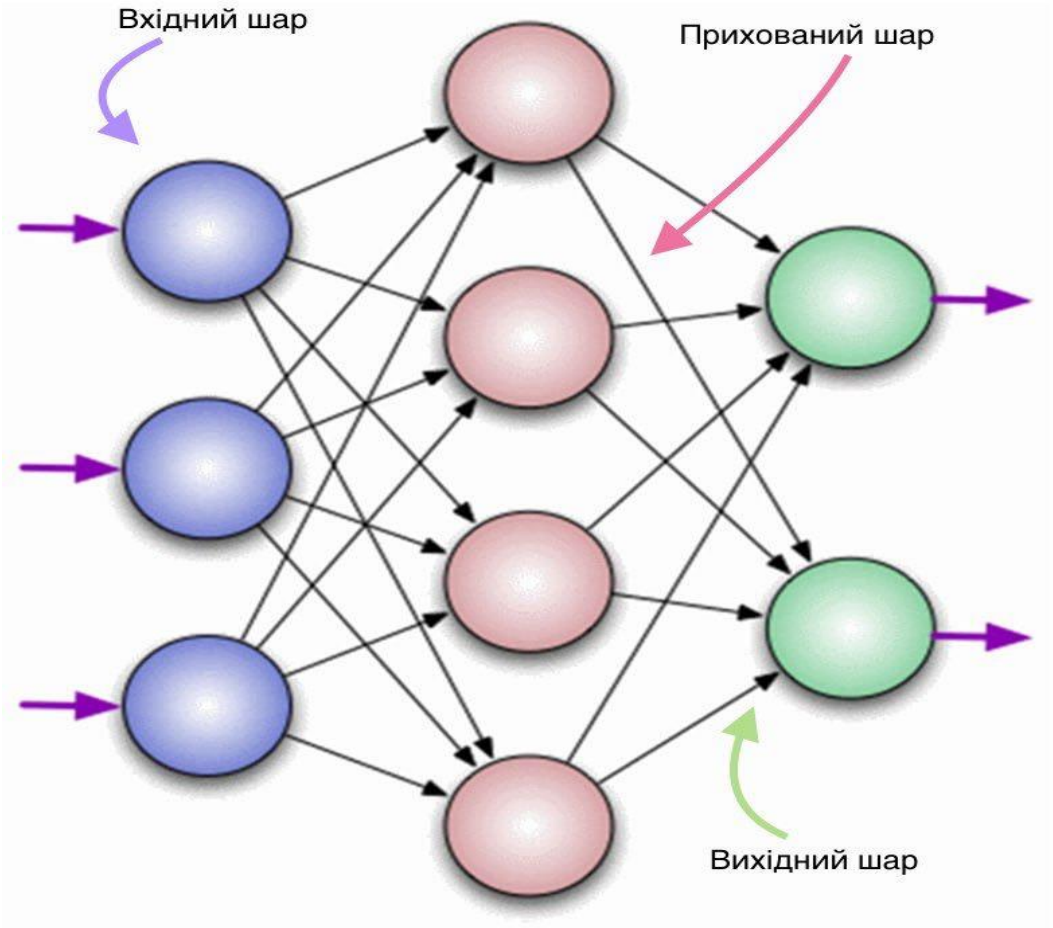

Рис.3.1 Схема штучної нейромережі із одним прихованим шаром *(Joel Grus, 2015)*

Штучна нейромережа з одним прихованим шаром є одним з найпоширеніших видів нейромереж, які використовуються для вирішення задач машинного навчання. Ця нейромережа містить один вхідний шар (на рисунку зображено синім кольором), один прихований шар (на рисунку зображено рожевим кольором) та один вихідний шар (на рисунку що зображено зеленим кольором). Кожен шар складається з нейронів, які сполучені між собою.

Вхідний шар приймає числові дані, та передає їх прихованому шару. Прихований шар обчислює значення на основі вхідних даних та передає їх вихідному шару. Вихідний шар зазвичай містить один або кілька нейронів, кожен з яких відповідає за вихідні значення нейромережі.

Кожен нейрон у штучній нейромережі з одним прихованим шаром має свої вагові коефіцієнти, які використовуються для обчислення вихідних значень нейромережі. Ці вагові коефіцієнти зазвичай визначаються під час навчання нейромережі, коли вона пропускається через набір тренувальних даних і коригує свої ваги залежно від результатів. *(Michael Nielsen, 2015)*

У кожного з нейронів є два головні параметри: вхідні дані (input data) і вихідні дані (output data). У випадку вхідних нейронів, вхідні дані дорівнюють вихідним. На інші ж нейрони, потрапляє сумарна інформація з попереднього шару, як саме інформація буде розподілена – залежить від топології нейронної мережі.

Оскільки штучна нейронна мережа є концепцією біологічної, то між нейронами має бути з'єднання у вигляді синапсу, і воно є. У штучній мережі, синапс має параметри, а саме – вагу. Кожен синапс має ваговий коефіцієнт *w.* Завдяки якому, інформація змінюється при передачі від одного нейрона до іншого. Схему вагових коефіцієнтів зображено на (рис 3.2).

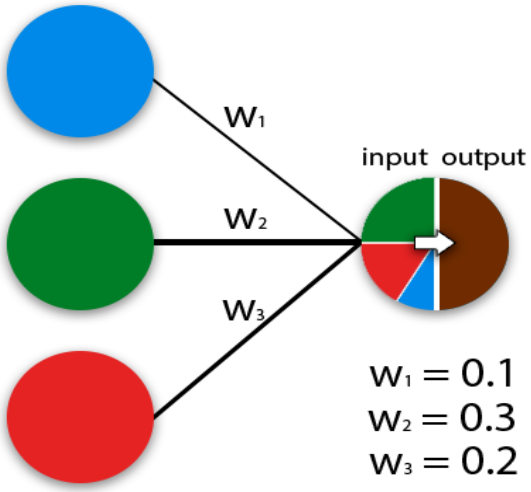

Рис. 3.2 Зображення нейронів із ваговими коефіцієнтами між ними *(Joel Grus, 2015)*

Як видно із схеми на (рис. 3.2) три вагових коефіцієнта w_1 w_2 w_3, кожен ваговий коефіцієнт пов'язаний зі зв'язком між певним вхідним нейроном та певним

нейроном наступного шару, вагові коефіцієнти перемають різну вагу, від величини якої буде залежати домінуюча інформація із попереднього нейрона у наступному. Завдяки саме ваговим коефіцієнтам, які змінюються в процесі тренування, вся інформація обробляється, і перетворюється в результат.

Після того, як нейрон отримав інформацію із попереднього шару, вона нормалізується за допомогою функції активації, і потрапляє на наступний нейрон.

Розрізняють навчання нейронної мережі із учителем, та без нього. Для задачі класифікації використовують навчання із учителем. У такому методі, кожна пара буде являти собою пару із вектору ознак, та вектору міток, тобто класу якому відповідають ознаки. Процес навчання є невід'ємним етапом створення нейронної мережі, на цьому етапі виконуються алгоритми для досягнення високої точності. *(Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016)*

3.2. Функція активації

Функція активації в штучних нейронних мережах використовується для визначення того, як нейрон відповідає на отриману інформацію. Вона виконує функцію перетворення вхідної інформації в вихідну. Функція активації може бути лінійною або нелінійною. Приклади нелінійних функцій активації: сигмоїда, ReLU, tanh. *(Chris Albon, 2018)*

Функції активації дозволяють мережі здійснювати складні обчислення і класифікувати дані з більшою точністю. Функції активації можуть мати різну форму, що дозволяє мережі адаптуватися до різних задач і приймати різні рішення. Це дозволяє мережі бути більш ефективною і достовірнішою у прогнозуванні і класифікації даних. Така функції дозволяє мережі "вибирати" ті вхідні дані, які важливі для розпізнавання, та відкидати непотрібні. Вибір функції активації може впливати на якість моделі і швидкість навчання, тому є важливим вибирати підходящу функцію активації в залежності від задачі, яку потрібно вирішити.

Формула для розрахунку вхідної активації нейрона штучної нейронної мережі. Вона показує, як вхідні дані x_n і ваги w_n перемножуються разом і додаються для отримання вхідної активації нейрона. Це значення подається на вхід функції активації і використовується для визначення вихідного значення нейрона.

$$\sum_{i=1}^{n} \mathrm{x} = (x_1 \times w_1) + (x_2 \times w_2) + (x_3 \times w_3) + \cdots + (x_n \times w_n) \quad (3.1)$$

Це формула (3.1) для обчислення зваженої суми n значень x, де кожне значення множиться на свій ваговий коефіцієнт w та додається до загальної суми.

- n - кількість значень, які додаються до суми
- x_1, x_2, w_3, ..., x_n - значення, що додаються до суми
- w_1, w_2, w_3, ..., w_n- вагові коефіцієнти, що відповідають кожному значенню
- i - індекс, що змінюється від 1 до n, використовується для звернення до кожного окремого значення та його відповідного вагового коефіцієнту.

Пропускаючи суму добутків, через функцію активації, ми отримуємо значення в певному діапазоні, який є визначеним згідно типу функції. Є багато функцій активації, і кожна з них використовується згідно вирішуваної задачі, але найчастіше використовується лише декілька з них.

1. Linear function – лінійна функція активації (рис. 3.3), діапазон вихідних значень не обмежений.

$$f(x) = x \quad (3.2)$$

Ця функція (3.2) називається ідентичною функцією. Вона повертає значення, що їй подається як аргумент. Іншими словами, для будь-якого вхідного значення x, значення функції буде рівним самому x.

До цієї функції неможливо використати зворотне розповсюдження, що використовується для навчання нейронної мережі, це тому що похідна функція є постійною і не має відношення до вводу, x. Тому повернутися назад і зрозуміти, які ваги вхідних нейронів можуть дати кращий прогноз, не можливо *(Sun, J., Slang, S., Elboth, T., Larsen Greiner, T., McDonald, S., Sebastian Rashka, 2015)*

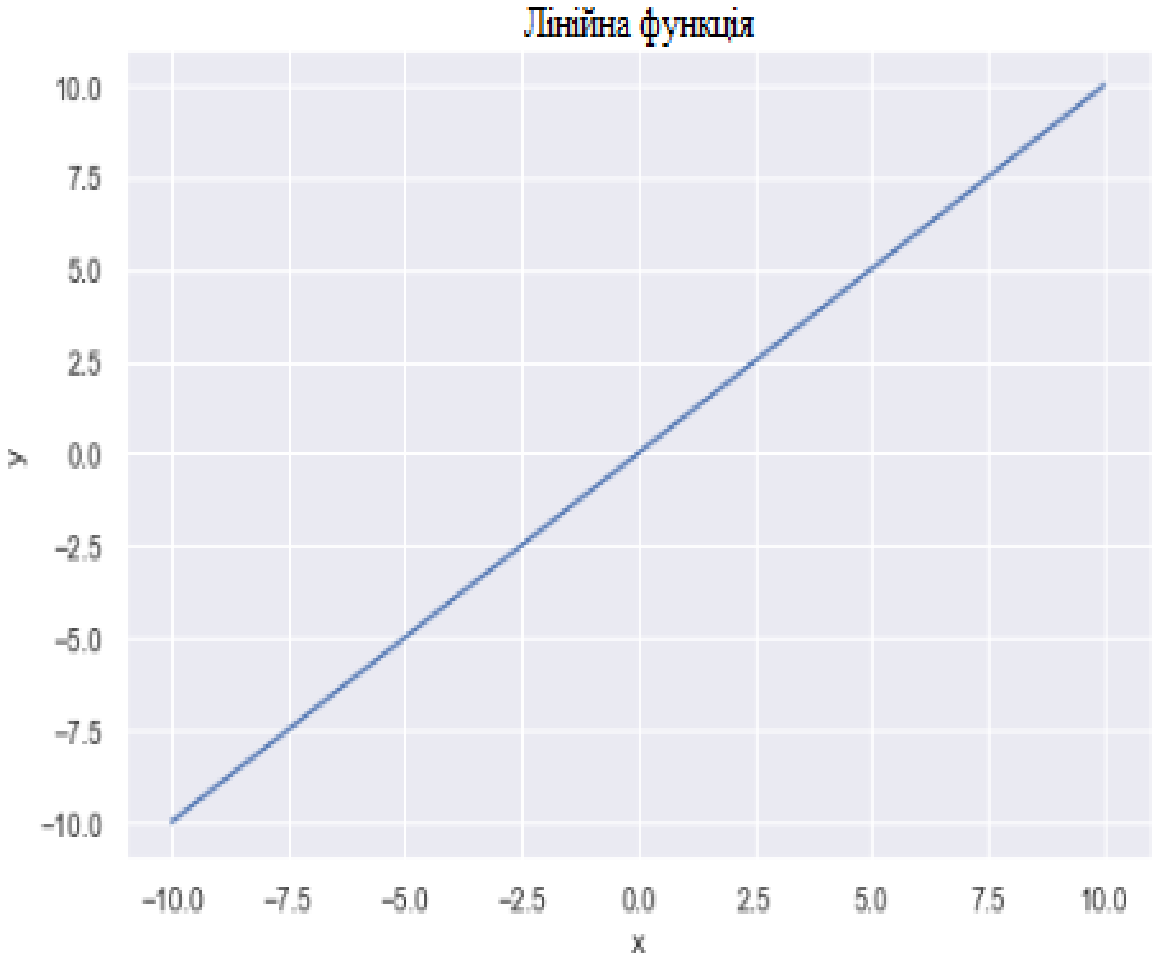

Рис. 3.3 Графік лінійної функції

2. Sigmoid – сигмоїд або логічна функція активації, графік зображений на (рис. 3.4).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

Сигмоподібна функція часто використовується, головна причина полягає у фіксованому діапазоні, в якому існує функція від 0 до 1, тобто значення нормалізуються. Тому вона знайшла своє застосування у моделях де передбачується ймовірність як вихідне значення. Існує проблема при навчанні із використанням такої функції, зміна градієнту при зміні вагових коефіцієнтів, для дуже високих або низьких значень x, прогнозування майже не змінюється, що призводить до проблеми зникаючого градієнта. Це може призвести до того, що мережа може відмовитися від подальшого навчання, або навчатися занадто повільно. *(Ioffe, S., Szegedy, C., 2015).*

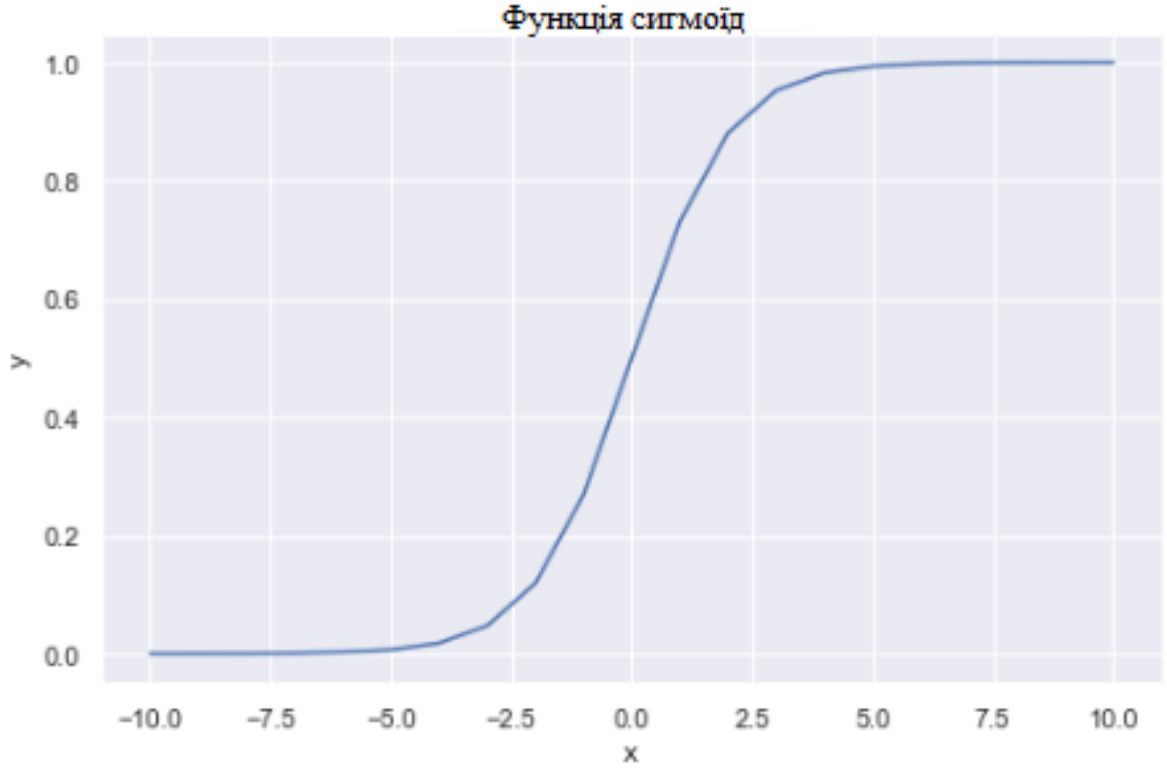

Рис.3.4 Графік функції сигмоїд

3. Hyperbolic Tangent – гіперболічний тангенс, схожий із сигмоїдом, але має деякі відмінності, графік функції (рис. 3.5).

$$f(x) = \frac{2}{1 + e^{-x}} - 1 \tag{3.4}$$

де *e* - це число Ейлера, а x - це вхід до функції.

Ця функція приймає значення між -1 і 1. Вона зазвичай використовується для того, щоб перетворити вагові суми вхідних сигналів у діапазон значень, що можуть бути передані наступному шару нейромережі.

Це зручно, оскільки значення вагових сум можуть бути дуже великими або дуже малими, і це може призвести до переповнення або недооцінки після застосування функції активації. Функція активації допомагає зробити значення більш стабільними та діапазон значень більш обмеженим *(Yoshua Bengio, Aaron Courville, 2016).*

Ця функція активації також має симетричний вигляд, тобто $f(-x) = -f(x)$, що дозволяє нейромережі розпізнавати як позитивні, так і негативні входи.

Функція також нормалізує вхідні дані, але діапазон нормалізації більший, від -1 до 1, що дає змогу центрувати значення по 0, та за рахунок більшого розподілу, показує кращі результати у роботі із великими вибірками. Має схожі проблеми із сигмоподібною функцією.

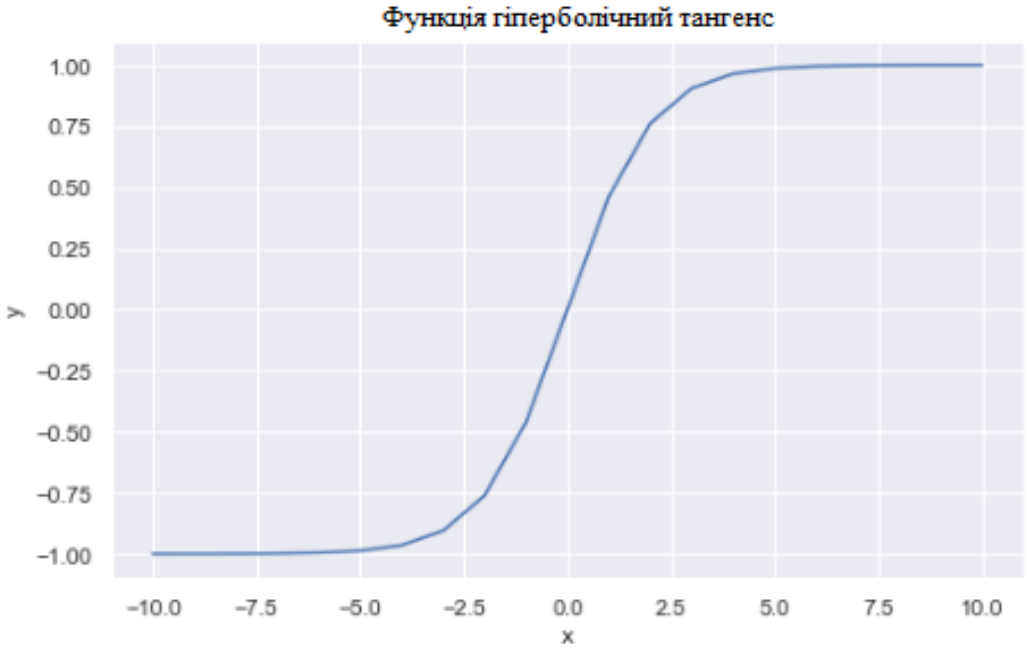

Рис. 3.5 Графік функції активації гіперболічний тангенс

4. Випрямлена функція ReLU – функція активації, що має діапазон значень від 0 до безкінечності, графік функції (рис. 3.6). Є досить популярною у конволюційних нейронних мережах, із-за специфіки даних вводу, таких мереж.

$$f(x) = max(0, x) \tag{3.5}$$

Функція дозволяє використовувати зворотнє розповсюдження для навчанян, але якщо значення x від'ємне, то воно автоматично стане рівним 0, це знижує здатність моделі правильно тренуватися.

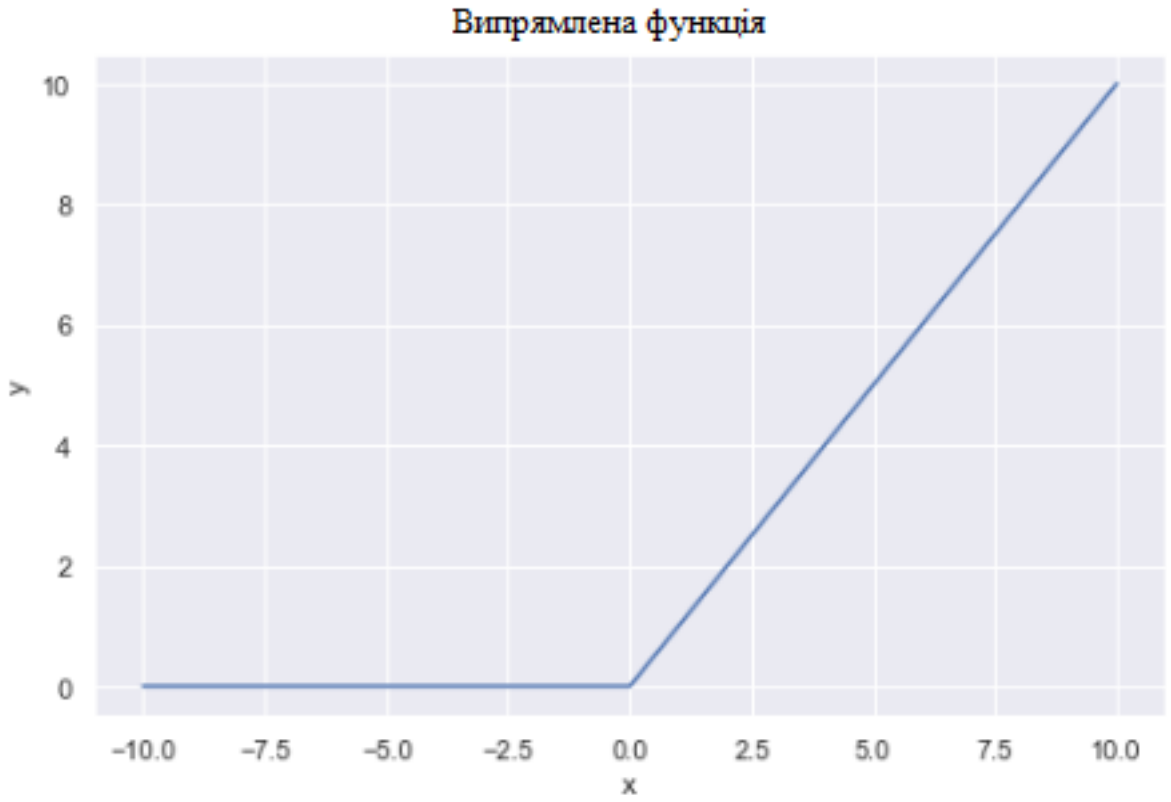

Рис. 3.6 Графік випрямленої функції активації

5. Порогова функція активації, або функція Гевісайда – це функція активації що має обробляє вхідні значення в 0 або 1, не маючи діапазону, графік зображений на (рис. 3.7).

$$f(x) = \{0\ if\ x < 0\ 1\ if\ x \geq 0\} \quad (3.6)$$

Зважаючи на те, що функція не є диференційованою, вона не може використовуватися в мережах, що навчаються за алгоритмом зворотнього поширення помилки, або іншим алгоритмом, що потребують диференційованої передавальної функції. Використовується для бінарної класифікації. (*Charu C. Aggarwal, 2018)*

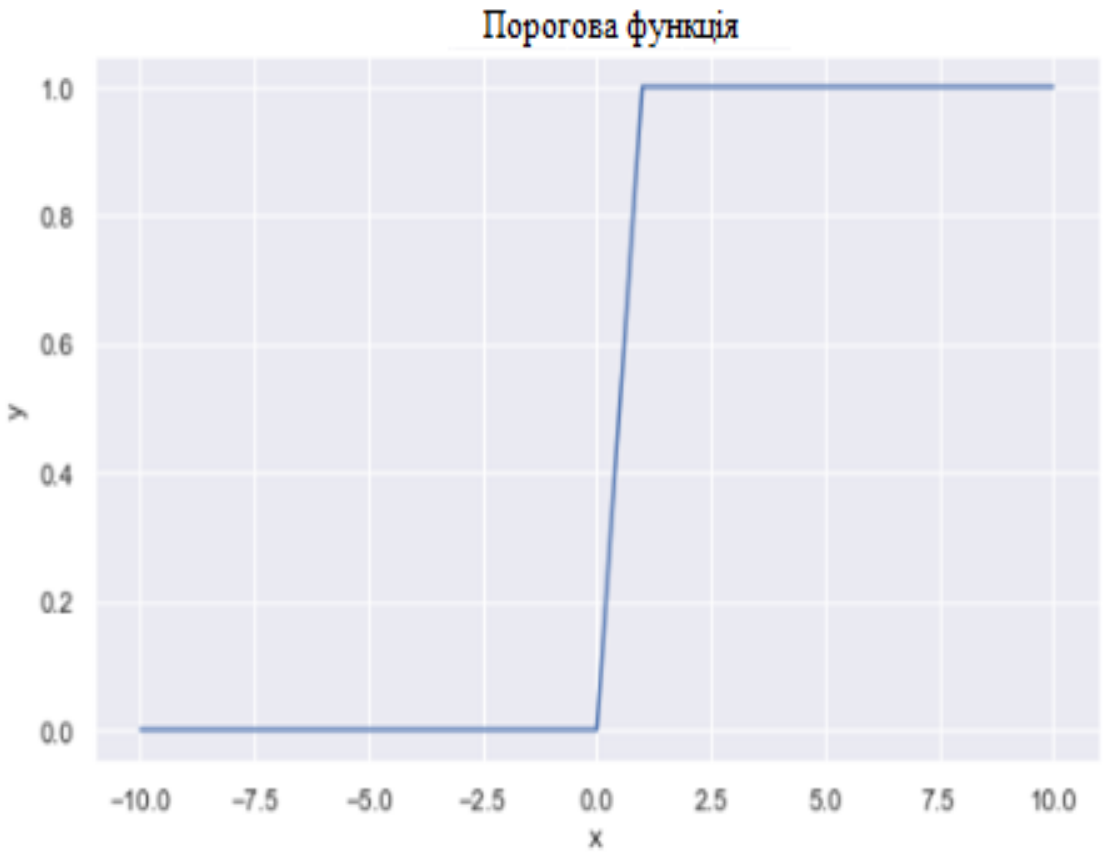

Рис. 3.7 Графік порогової активаційної функції

6. Нормалізована експоненціальна функція, або SoftMax – активаційна функція, повертає розподіл вірогідності за цільовими класами в мультикласовій класифікації, графік функції (рис 3.8)

$$f(x) = \frac{exp(x_i)}{\Sigma expexp\,(x_j)}, \tag{3.7}$$

Згідно рівняння (3.7), для кожного вхідного вектора, значення функції це показник індивідуального входу поділений на суму показників усіх входів. Діапазон значень функції від 0 до 1, сума всіх значень по класам, буде дорівнювати 1. Функція використовується для вирішення задач класифікації, із функцією втрат в ролі перехресної ентропії. *(Simon Haykin, 2008)*

Рис. 3.8 Графік нормалізованої експоненціальної функції активації

3.3 Мінімізація похибки

Середня квадратична похибка, також відома як L2 вимірюється як середня квадратична різниця між прогнозованими та фактичними мітками які отримані у результаті розрахунків нейронної мережі

$$RMSE = \frac{\sum_{i=1}^{n} \cdot (y_i - y_i')^2}{n} \quad (3.8)$$

Де y_i , y_i', параметр який є фактичним значенням для навчання, та передбачена мітка відповідно. *(Eugene Charniak, 2018)*

Із використанням такої функції, завдяки квадрату прогнози які далекі від фактичних значень будуть сильно штрафуватися оптимізатором порівняно із тими, де відхилення менше.

Середня абсолютна похибка відома як L1, вимірюється як середня сума абсолютних різниць між прогнозованими значеннями та фактичними

$$MAE = \frac{\sum_{i=1}^{n} \cdot |y_i - y_i'|}{n} \quad (3.9)$$

Перехресна ентропія перехресна ентропія вимірює розбіжність між двома розподілами ймовірностей, якщо логічна ентропія велика, це означає, що різниця між двома розподілами теж є великою, а у випадку коли перехресна ентропія мала, то розподіли схожі між собою

$$H(p, q) = E_p[-logq] \quad (3.10)$$

$H(p, q)$ - це крос-ентропія між розподілами p та q.

$E_p[-logq]$ - це очікуване значення функції -log q, коли випадкова величина розподіляється згідно з розподілом p. У випадку із бінарною класифікацією кожну передбачувану ймовірність, порівнюють із фактичним значенням виходу класу (0 або 1) і підраховують бал, який враховує ймовірність на основі відстані від очікуваного значення.

$$CE = -(y_i \, log \, log \, (y_i') + (1 - y_i) log \, (1 - y_i')) \quad (3.11)$$

Основною метою бінарної класифікації є передбачення класу об'єкта (наприклад, чи зображений на зображенні кіт). При цьому можна зустріти два види

помилок: об'єкт, який насправді належить до класу 0, був визначений як 1 (false positive) або об'єкт, який насправді належить до класу 1, був визначений як 0 (false negative).

У функції хиби для кожного прикладу існує відповідне значення y (0 або 1) - правильна відповідь, та ймовірність y' (число від 0 до 1), що передбачає модель нейронної мережі. Функція хиби CE (cross-entropy) порівнює відповіді моделі з правильними відповідями та обчислює величину помилки.

У формулі, що наведена в питанні, ∑ означає сумування від *i*=1 до n (кількість прикладів), y_i - правильна відповідь, $y_i^{`}$ - відповідь моделі, $log$ - логарифм за основою 2, (1-$y_i$) та (1-$y_i^{`}$) - значення, що дорівнюють 1, або 0, якщо відповідно y_i та $y_i^{`}$ дорівнюють 0 або 1. Функція хиби CE буде мінімальною, якщо модель нейронної мережі добре передбачає класи об'єктів.

Точність похибки є важливою складовою у навчанні нейромережі, адже саме на неї опирається оптимізатор перед зміною вагових коефіцієнтів, від яких у свою чергу і залежить точність класифікації.

3.4. Оптимізатори вагових коефіцієнтів

Як було сказано вище, нейронна мережа навчається шляхом оновлення вагових коефіцієнтів, які відіграють ключову роль у результаті який виведе мережа. На початку ініціації, значення вагових коефіцієнтів задається випадково, а точніше псевдо випадково, адже задаються певні параметри розподілу у генераторі випадкових чисел. Але якщо їх оновлювати випадково і при навчанні, то це не буде ефективним підходом, адже у такому разі значення похибки втрат не враховується. Для того щоб навчання було якомога ефективнішим, використовують оптимізатори – алгоритми для оновлення параметрів моделі.

Градієнтний спуск – це алгоритм оптимізації, який використовується для знаходження мінімального значення функції втрат, шляхом ітеративного

переміщення в напрямку найкрутішого спуску. Використовується для оновлення параметрів лінійної регресії, та вагових коефіцієнтів у нейронних мережах. *(Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2009)*

За допомогою градієнтного спуску, ми знаходимо значення вагового коефіцієнта у мінімальній точці глобальної функції втрат. На (рис. 3.9), зображено умовну форму графіку функції втрат.

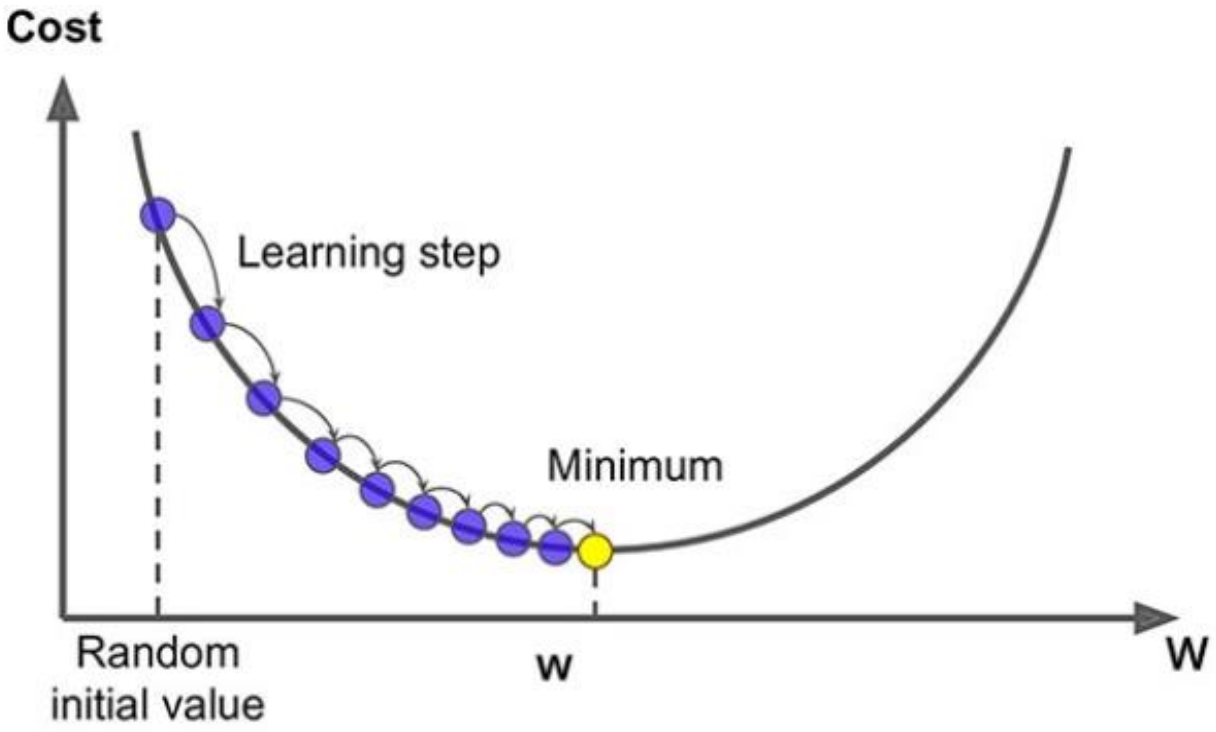

Рис 3.9 Візуальне представлення роботи градієнтного спуску, із умовним графіком функції втрат *(Joel Grus, 2015)*

Окрім функції втрат, алгоритм градієнтного спуску потребує сам власне градієнт функції, який є похідною функції втрат ∂J відносного одного вагового коефіцієнта, виконаного для всіх вагових коефіцієнтів *(Grzegorz Nowakowski,Yaroslaw Dorogyy, Olena Doroga-Ivaniuk, 2017)*

$$grad = \frac{\partial J}{\partial w} \tag{3.12}$$

Формула знаходження градієнта функції, буде різною в залежності від використаної функції втрат. Для середньоквадратичної функції втрат формула знаходження градієнта, виглядає так:

$$\frac{\partial}{\partial W_j} MSE(\theta) = \frac{2}{m} \Sigma (W^T \times x^{(i)} - y^{(i)}) x_j^{(i)} \quad (3.13)$$

Де θ - це вектор параметрів моделі, що включає в себе ваги W.

W^T - транспонований вектор ваг моделі.

$x^{(i)}$ - це і-й приклад з навчального набору даних.

$y^{(i)}$ - це істинна мітка для і-го прикладу з навчального набору.

$x_j^{(i)}$ - це j-й елемент вектора ознак і-го прикладу

$\Sigma (W^T \times x^{(i)} - y^{(i)}) x_j^{(i)}$ - це сума по всіх прикладах навчального набору від різниці між передбаченням моделі і істинною міткою, помножену на j-й елемент вектора ознак.

Ця формула представляє собою вираз для обчислення градієнта для середньоквадратичної помилки (Mean Squared Error, MSE) за вагами W. Тут градієнт використовується для обчислення часткової похідної від середньоквадратичної помилки по вагам моделі. Множник 2/m на початку - це нормалізація, де m - кількість прикладів в навчальному наборі.

В алгоритмі спуску градієнта, ініціалізація починається з параметрів випадкової моделі, обчислюючи помилку для кожної ітерації навчання, продовжується оновлення параметри моделі, щоб наблизитись до значень, що призводить до мінімальних втрат. Формула (3.14) представляє алгоритм градієнтного спуску, який є основним алгоритмом оптимізації у машинному навчанні.

$$W_j = W_j - \alpha \frac{\partial}{\partial w_j} J(W) \tag{3.14}$$

$W_j -$ це j-й вага в наборі параметрів моделі.

α це швидкість навчання, яка визначає, наскільки далеко крок робиться на кожному етапі оптимізації.

$\frac{\partial}{\partial w_j} -$ це часткова похідна функції втрат $J(W)$ по j-му вагу. Вона вказує напрямок найбільшого зростання функції втрат.

Якщо параметр α буде занадто малим, є ризик потрапити у локальний мінімум, а якщо занадто великим, то є ризик взагалі не знайти мінімум, в обох випадках модель не зможе навчитися із максимальною ефективністю, тому необхідно визначати оптимальне значення.

Існує декілька типів градієнтного спуску, які використовуються у навчанні нейромереж.

Пакетний спуск градієнта: використовує всі навчальні екземпляри, для оновлення параметрів моделі в кожній ітерації.

Міні-пакетний градієнт спуск: Замість використання всіх прикладів, міні-пакетний градієнтний спуск розділяє навчальний набір на менший розмір, що називається партією. Таким чином, міні-пакет використовується для оновлення параметрів моделі в кожній ітерації.

Стохастичний градієнтний спуск (SGD): оновлює параметри, використовуючи лише один навчальний екземпляр у кожній ітерації. Навчальний екземпляр зазвичай вибирається випадковим чином. Стохастичний градієнтний спуск часто вважають доцільним для оптимізації функції витрат, коли є сотні тисяч навчальних екземплярів або більше, оскільки він буде сходитися швидше, ніж пакетний

градієнтний спуск *(Grzegorz Nowakowski,Yaroslaw Dorogyy, Olena Doroga-Ivaniuk, 2017).*

Окрім градієнтного спуску, використовується наступні оптимізатори:

Adagrad (адаптивний градієнт) – індивідуально встановлює параметри оновлення, для кожного вагового коефіцієнта. Параметри, які мають більш високі градієнти або часті оновлення, повинні мати повільнішу швидкість навчання, щоб ми не перевищували мінімальне значення. Параметри, які мають низькі градієнти або нечасті оновлення, повинні швидше навчатись, щоб для оптимального результату тренування.

Оптимізатор ділить ступінь навчання на суму квадратів усіх попередніх градієнтів параметра. Коли сума квадратних минулих градієнтів має високе значення, вона, в основному, ділить рівень навчання на високе значення, тому швидкість навчання стане меншою. Аналогічно, якщо сума квадратних минулих градієнтів має низьке значення, вона ділить ступінь навчання на нижчу величину, тому значення коефіцієнта навчання стане високим. Це означає, що швидкість навчання обернено пропорційна сумі квадратів усіх попередніх градієнтів параметра, записується формулою *(Grzegorz Nowakowski,Yaroslaw Dorogyy, Olena Doroga-Ivaniuk, 2017):*

$$W = W - \alpha \frac{\partial J(w_t^i)}{\sqrt{\sum_{r\cdot}^{t} (g_t^i)^2}} \tag{3.15}$$

Де g_t^i – градієнт функції втрат, в певній ітерації,

α – значення ступеня навчання,

$\partial J(w_t^i)$ – похідна функції втрат у ваговому коефіцієнті, при ітерації *i*

Momentum - використовуючи стохастичний градієнт спуск (sgd) або міні-пакетний градієнт спуск, враховує минулі градієнти, щоб згладити оновлення. Це видно в змінній v, яка є середньозваженою в експоненціальному відношенні градієнтом на попередніх кроках (3.16). Це призводить до мінімізації коливань і більш швидкого сходження, формула (3.17).

$$v_{dW} = \beta_{vdW} + (1+\beta)\frac{\partial J}{\partial W} \tag{3.16}$$

$$W = W - \alpha v_{dW} \tag{3.17}$$

Де v – експоненціальне зважене середнє значення минулих градієнтів**,**

$\frac{\partial J}{\partial W}$ – градієнт функції втрат,

W – матриця вагових коефіцієнтів,

β – гіперпараметр який потрібно змінити,

RMSprop - алгоритм адаптивної оптимізації швидкості навчання, Root Mean Square Prop (RMSProp) працює, зберігаючи експоненціально зважене середнє значення квадратів минулих градієнтів (3.18). Потім RMSProp градієнт функції, на середнє для прискорення сходження значень *(Ioffe, S., Szegedy, C., 2015)*:

$$S_{dW} = \beta s_{dW} + (1-\beta)(\frac{\partial J}{\partial W})^2 \tag{3.18}$$

$$W = W - \alpha \frac{\frac{\partial J}{\partial W}}{\sqrt{S_{dW}^{corrected}}} \tag{3.19}$$

Де s - експоненціальне зважене середнє значення минулих квадратів градієнтів,

$\frac{\partial J}{\partial W}$ – градієнт функції втрат,

W – матриця вагових коефіцієнтів, які мають оновитися,

β – гіперпараметр який потрібно змінити,

α – ступінь навчання.

Adam - Адаптивна оцінка моменту, оптимізатор обчислює адаптивну швидкість навчання для кожного параметра.

Спершу, він обчислює експоненціально зважене середнє значення минулих градієнтів функції втрат відносно до поточного шару. Потім він обчислює експоненціально зважене середнє значення квадратів минулих градієнтів: *(Tarun Kumar Gupta, Khalid Raza, 2018)*

$$S_{dW} = \beta_2 s_{dW} + (1 - \beta_2)(\frac{\partial J}{\partial W})^2 \tag{3.20}$$

Середні значення мають зміщення до нуля, і для протидії цьому застосовується корекція зміщення (3.21) та (3.22), *(L., Zheng, X., Duan, Y., Yan, X., Hu, Y., Zhang, X., 2019)*

$$v_{dW}^{corrected} = \frac{v_{dW}}{1 - (\beta_1)^t} \tag{3.21}$$

формула описує оновлення параметру ваг (v_{dW}) в градієнтному спуску з моментом (momentum), який є популярним алгоритмом оптимізації нейронних мереж. У формулі $v_{dW}^{corrected}$- це скоректоване оновлення параметру ваг, v_{dW} - це оновлення параметру ваг, β_1- це експоненціальний коефіцієнт затухання, який

зазвичай дорівнює 0,9, а t - це номер ітерації. У знаменнику формули $1-(\beta_1)^t$- це затухаючий коефіцієнт, який зменшує вплив попередніх значень градієнтів на оновлення параметру ваг, тобто зменшує швидкість затухання градієнту під час навчання.

Отже, використовуючи цю формулу, можна оновлювати параметри ваг у градієнтному спуску з моментом, що дозволяє ефективно навчати нейронні мережі і забезпечує швидкий збіг до оптимальних значень параметрів ваг.

$$s_{dW}^{corrected} = \frac{S_{dW}}{1-(\beta_1)^t} \tag{3.22}$$

Дана формула використовується для корекції ваги при підрахунку експоненційно зваженого ковзного середнього (EWMA) для даних з часовою залежністю.Корекція полягає у врахуванні впливу всіх попередніх значень на поточне значення ваги. Вага кожного з попередніх значень зменшується з часом в залежності від параметра згасання β. У формулі, s_{dW} - це вага попереднього значення, $s_{dW}^{corrected}$ - це вага попереднього значення, скорегована з урахуванням попередніх значень та параметра згасання β, t - це кількість кроків в EWMA.

Таким чином, формула описує, як коригувати вагу попереднього значення в EWMA з урахуванням попередніх значень та параметра згасання β.

Параметри оновлюються за допомогою інформації з розрахованих середніх значень.

$$W = W - \alpha \frac{v_{dW}^{corrected}}{\sqrt{s_{dW}^{corrected}}} \tag{3.23}$$

Де W - це вагові коефіцієнти моделі, які оновлюються;

alpha - коефіцієнт швидкості навчання, що контролює величину кроку, на який ваги оновлюються під час кожного ітераційного кроку;

$v_{dW}^{corrected}$ - це експоненціально згладжена оцінка градієнту вагів моделі;

$s_{dW}^{corrected}$- це експоненціально згладжена оцінка квадрату градієнту вагів моделі.

4. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА ТА МЕТОДИКА ДОСЛІДЖЕНЬ

4.1 Підготовка даних

Досліджувана територія, звідки були взяті дані для подальшого аналізу а саме геологічна будова західного узбережжя Норвегії та Північного моря, взяті з Норвезького Нафтового Директорату (NPD).

Дані були прийняті в стандартному текстовому форматі LAS, який використовується для запису каротажних кривих. Спочатку було вибрано ті інтервали, в яких присутня літологія, основана на описі керну.

	CALI	DEPT	DRHO	DTC	DTE	GR	NPHI	RDEP	RMED	RSHA	SP	well	LITHOLOGY	FaciesLal
0	13.227000	2190.033203	0.015000	117.766327	131.976196	57.478718	0.461820	0.731000	0.768000	0.851011	42.607983	15_9-17	5	
1	13.352000	2190.185791	0.011000	113.726906	127.294159	49.500000	0.434127	0.751000	0.753000	0.857888	42.341866	15_9-17	5	
2	13.241000	2190.338135	0.013000	109.179535	122.598312	46.629601	0.444732	0.754000	0.787000	0.823474	41.586853	15_9-17	5	
3	13.025000	2190.490478	0.014000	107.199783	120.198967	48.996193	0.427210	0.754000	0.866000	0.971589	42.180199	15_9-17	5	
4	13.282999	2190.642822	0.018000	106.086380	119.060043	48.001602	0.420740	0.757000	0.922000	1.154846	42.575893	15_9-17	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
243926	18.073999	2276.009033	0.047958	101.582390	100.142174	48.278000	-0.055140	1.683030	0.903010	0.903010	71.094002	7_3-1	5	
243927	18.000999	2276.161377	0.045010	103.808281	102.133095	50.699120	-0.056777	1.692984	0.909014	0.909014	72.750999	7_3-1	5	
243928	18.020000	2276.313721	0.025096	103.750862	102.244469	51.971962	-0.058328	1.701986	0.918000	0.918000	74.087997	7_3-1	5	
243929	18.066999	2276.466064	0.023010	103.986992	102.597061	53.304863	-0.059498	1.710971	0.929000	0.929000	72.960999	7_3-1	5	
243930	18.090000	2276.618652	0.020010	104.392097	102.765923	53.308998	-0.060680	1.719986	0.936002	0.936002	73.473999	7_3-1	5	

Рис. 4.1 Візуальне представлення набору даних

Для подальшої обробки даних, необхідно очистити набір від пустих або некоректних значень, запис із некоректними значеннями не використовуються для тренувальної вибірки, тому що це може спотворити результат навчання нейронної мережі. Для відображення літології гірських порід було використано маркування, яке представлено на рисунку 4.2. Це позначення які в подальшому будуть використовуватися для відображення гірських порід які вміщуються в літологічній колонці.

№	Скорочення	Повна назва
1	Ss	Пісковик
2	SIS	Пісковик мулистий
3	CBSS	Шаруватий пісковик
4	SnS	Пісчаний мул
5	SI	Мул
6	ShSI	Вапняковий мул
7	SISh	Мулистий вапняк
8	Cn	Карналіт
9	Ch	Крейда
10	PLs	Пористий вапняк
11	Ls	Вапняк
12	Ag	Аргіліт
13	Ms	Мергель
14	Gp	Гіпс
15	DI	Доломіт
16	CIC	Вапняковий цемент
17	Cng	Конгломерат
18	C	Вугілля
19	Cr	Вулканічний туф
20	Ah	Ангідрит
21	H	Галіт

Рис. 4.2 Кількість гірських порід у вибірці

Досліджувані дані були отримані за методами ГДС перелік яких наведено на рисунку 4.3. Скорочення використовуються в подальшому для відображення каротажних кривих.

Номер	Скорочення	Метод	Одиниці вимірювання
1	CALI	Кавернометрія	in
2	DTC	Акустичний каротаж	us/ft
3	NPHI	Нейтрон-нейтронний каротаж	m3/m3
4	DRHO	Метод корекції густини	g/cc
5	GR	Гамма каротаж	gAPI
6	SP	Метод самочинної поляризації	mV
7	R	Метод уявного опору	OHM.m

Рис. 4.3 Данні ГДС які були використані

Переглянемо розподіл класів у наборі даних на рисунку 4.4. Кольори та підписи індексовані, тому будуть використовуватися в таких же парах і в подальшій візуалізації.

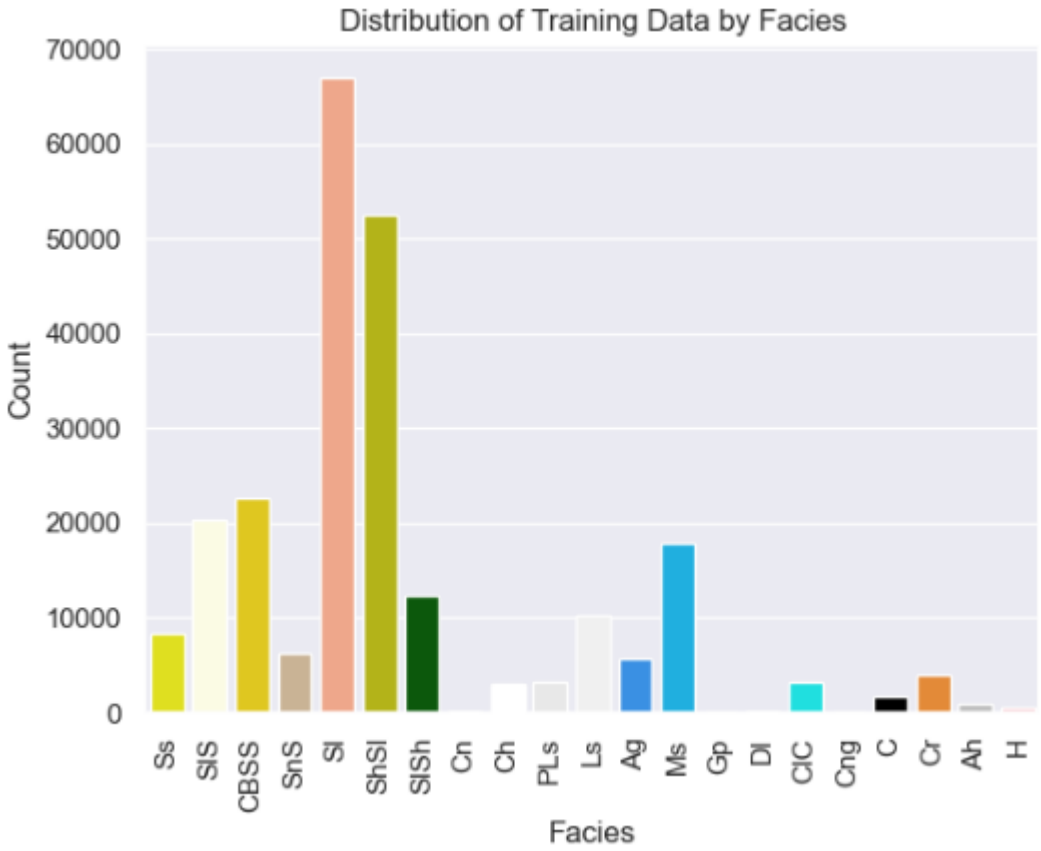

Рис. 4.4 Гістограма розподілу класів у наборі даних

На етапі підготовки даних для класифікатора були виконані кілька кроків. Спочатку, для перевірки точності класифікатора у подальшому, з вибірки було вирізано одну свердловину, яка не входить до тренувального набору даних. Далі, з вектору ознак були видалені зайві параметри, які не мають відношення до розподілення класів, наприклад, глибина, індекс та назва свердловини.

Крім того, категоріальна ознака літології була перетворена на вектор міток. Після цього, дані були розподілені між тренувальною та валідаційною вибірками, 75% даних відноситься до тренувальної вибірки та 25% до валідаційної вибірки. Останнім кроком є нормалізація векторів ознак. Цей процес приводить значення ознак до відомого діапазону. Це дозволяє уникнути недооцінення або переоцінення значимості деяких ознак при виконанні класифікації. Після нормалізації даних, вони передаються в класифікатор, який виконує свою роботу за допомогою алгоритму машинного навчання.

У результаті класифікатор визначає клас для кожного елемента вибірки. Щоб перевірити точність класифікатора, буде використано валідаційну вибірку, відомість про яку класифікатор не має. Після отримання результатів, вони можуть візуалізуватися для кращого порозуміння результатів класифікації. Це також допоможе виявити можливості та забезпечити якісне виконання проекту. Процес контролю та оцінки дозволяє виявляти ризики та вплив на результат, визначати необхідні коригувальні дії та забезпечувати ефективне управління процесом навчання та згодом класифікації.

4.2 Створення нейронної мережі

Для побудови моделі яка буде найбільш ефективнішою і точною, в роботі було використано підбір гіперпараметрів. У нейронних мережах до гіперпараметрів належать кількість шарів, кількість нейронів у кожному шарі, тип функції активації, швидкість навчання, розмір партії та кількість епох. Загалом гіперпараметри для конкретного набору даних, були підібрані за допомогою випадкового пошуку. Випадковий пошук є популярним методом для гіперпараметричного налаштування в нейронних мережах. У випадковому пошуку, гіперпараметри вибірки з заздалегідь визначеного розподілу і продуктивність моделі оцінюється для кожного набору гіперпараметрів. Потім вибирається найкращий набір гіперпараметрів на основі результатів оцінки. Випадковий пошук був показаний для того, щоб добре виконати в порівнянні з ґратковим пошуком, який оцінює всі можливі комбінації гіперпараметрів. Такий тип підбору параметрів є більш ефективним і має вищі шанси знайти хороший набір гіперпараметрів в меншій кількості випробувань.

Розглянемо перший випадок із оптимізатором стохастичний градієнтний спуск, та функціями активації гіперболічний тангенс, випрямленою функцією relu, та логічною функцією активації (рис 4.4).

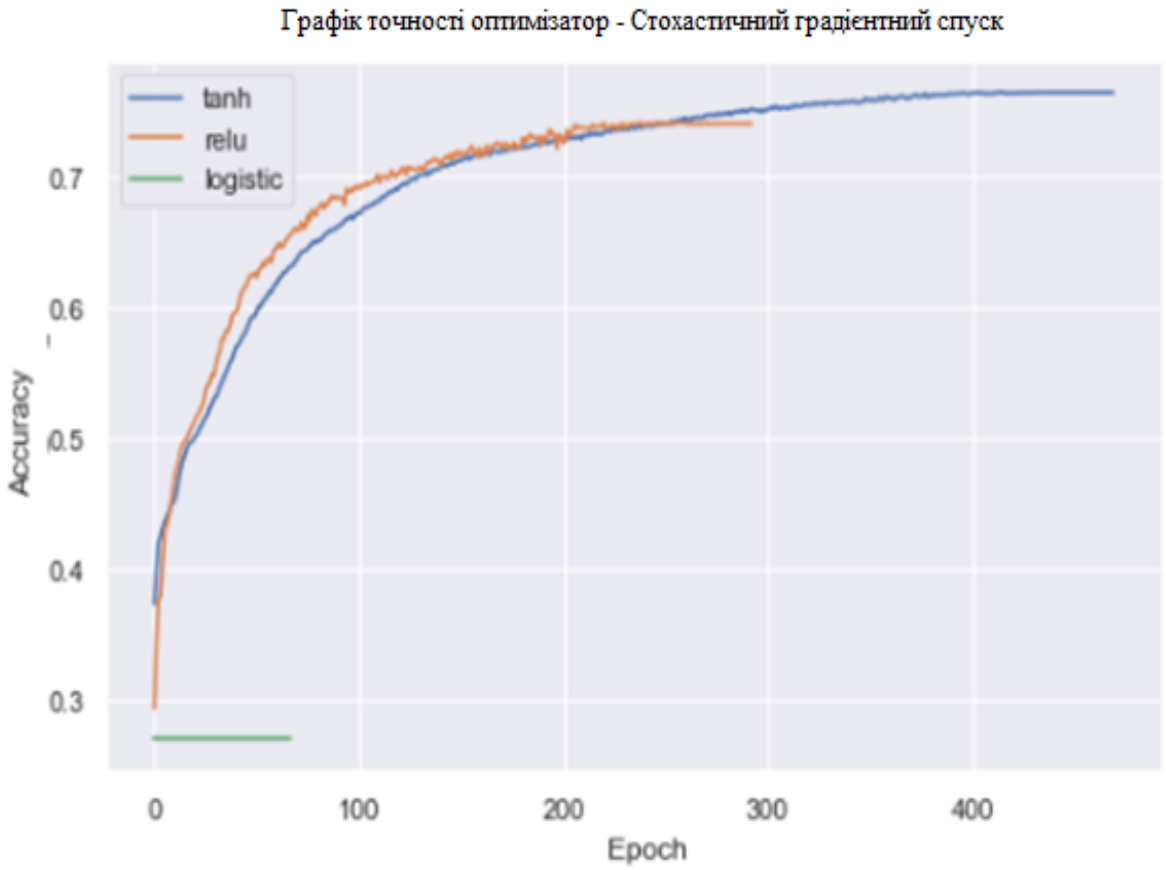

Рис. 4.5 Точність мережі, з оптимізатором стохастичного градієнтного спуску при використанні різних функцій активації

```
# Розбиття на тренувальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Створення моделі
model = MLPClassifier()

# Задання гіперпараметрів та їх значень для пошуку
parameters = {
    'hidden_layer_sizes': [(50,8), (100,4), (200,2)],
    'activation': ['tanh', 'relu', 'logistic'],
    'solver': ['sgd', 'adam', 'lbfgs']
}

# Пошук гіперпараметрів за допомогою GridSearchCV
grid_search = GridSearchCV(model, parameters, n_jobs=-1, cv=5)
grid_search.fit(X_train, y_train)
```

Рис 4.6 Представлення лістингу коду для підбору гіперпараметрів

Використовуючи такий оптимізатор, функції активації поводять себе по-різному. Перш за все, ми бачимо сигмоїдальну функцію, яка не працює із заданими параметрами. Інші дві показують результати, але вони відрізняються між собою. Відмінність полягає у ініціалізації нейронної мережі, а саме різних значеннях точності на старті, при таких стартових показниках із функцією relu, мережі знадобилося менше часу на навчання, близько 300 епох. Чого не можна сказати про гіперболічний тангенс, цій функції знадобилося близько 480 епох при навчанні мережі. Найвищий показник точності при тренуванні, є із використанням гіперболічного тангенсу, але так мережа навчалася найдовше.

Іншим використаним оптимізатором, є оптимізатор адаптивної оцінки моменту – Adam. Цей оптимізатор використано із тими параметрами, що і попередній результат, зображено на (рис. 4.5).

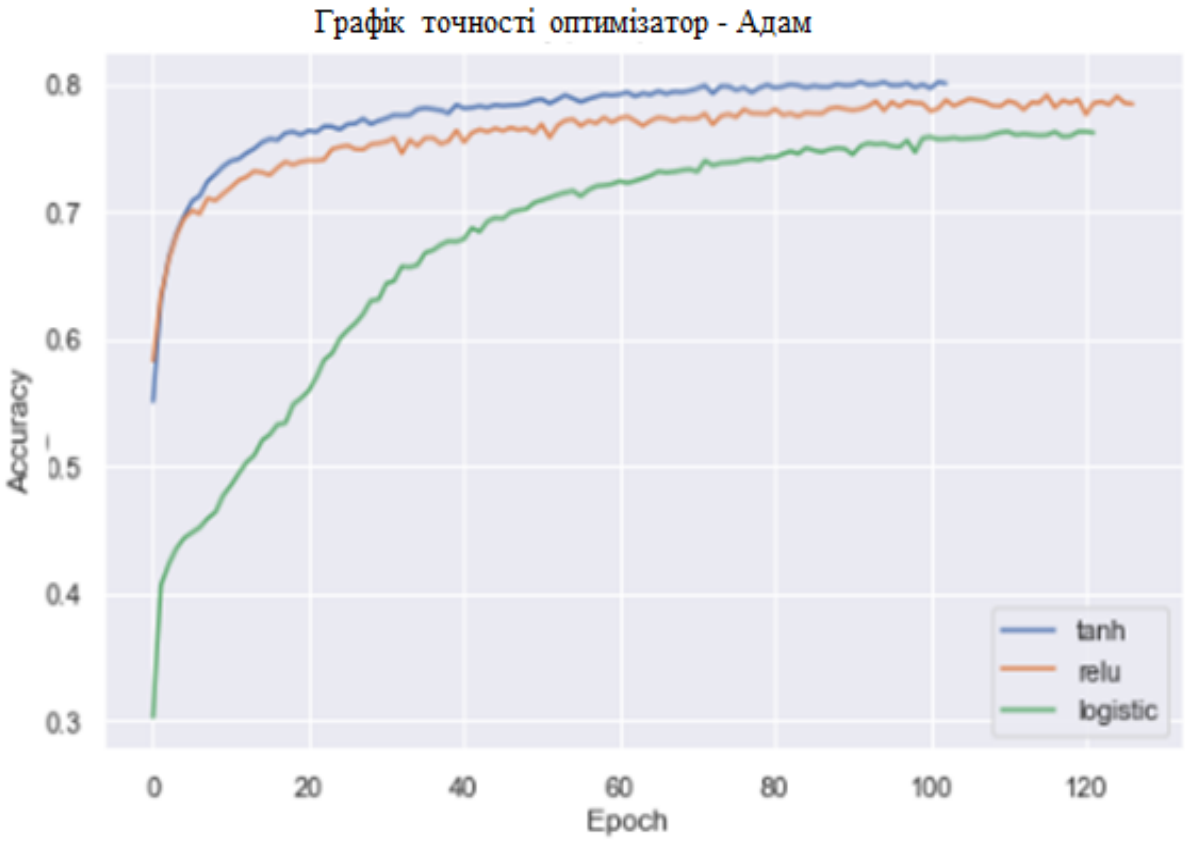

Рис. 4.7 Точність мережі, з оптимізатором Adam при використанні різних функцій активації

Порівнюючи результати із попереднім оптимізатором, можна зробити висновок, що Adam показує більш точні результати при навчанні нейронної мережі. Але це не єдина перевага, слід звернути увагу на швидкість навчання, із використанням різних функцій активації. При використанні функції гіперболічний тангенс, отримано максимальну точність за два випробування, а також найбільшу швидкість навчання мережі, що свідчить про ефективність обраних параметрів, та доцільність їх використання у подальшому.

Велику роль у працездатності нейронної мережі, відіграє кількість прихованих шарів, та кількість нейронів, що в них знаходяться. Експериментально проведено декілька розрахунків точності із різною кількістю прихованих шарів, та нейронів. При дослідженні із використанням різних оптимізаторів, нейронна мережа вміщувала 4 прихованих шари по 100 нейронів у кожному. Загалом у мережі 400 вузлів, у яких проводяться обчислення. На графіку (рис. 4.7) зображено криві, які відповідають одному числу нейронів, але різній кількості прихованих шарів. Використаний оптимізатор – Adam, активаційна функція – гіперболічний тангенс, інші параметри залишаються незмінними із попередніх досліджень.

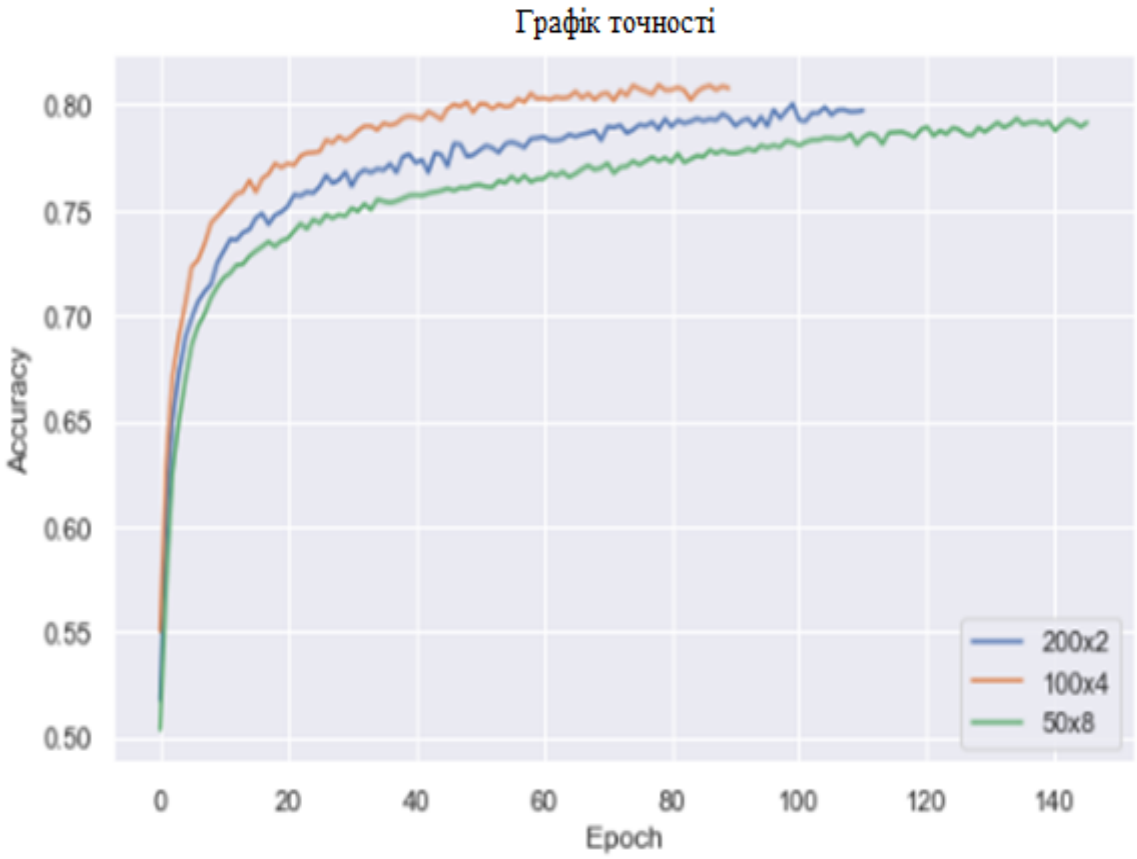

Рис. 4.8 Графік зміни точності, та швидкості навчання, залежно від кількості прихованих шарів

Кількість прихованих шарів із однаковим сумарним числом нейронів на мережу, значно впливає на показники точності, і швидкості навчання. На графіку показано чітку різницю кривими, мережа із двома шарами, дає усереднені значення відносно двох інших, мережа яка має 4 приховані шари, показує найкращі результати із досліджуваних. Чого не можна сказати про мережу яка має 8 шарів, та найгірший результат у порівнянні із іншими досліджуваними структурами мереж.

Із результатів проведеного дослідження, висновком є використання 100 нейронів у одному прихованому шарі. Тому що зменшення вузлів призводить до втрати точності та повільного навчання, навіть за умови більшого числа прихованих шарів.

Важливо підібрати оптимальну кількість, для точної класифікації. На (рис. 4.8) зображено графік із різною кількістю прихованих шарів, і найбільшу точність має нейронна мережа із семи шарів. Мережа яка має 9 прихованих шарів, не дає покращення результату, а навіть навпаки, є деякі втрати у точності.

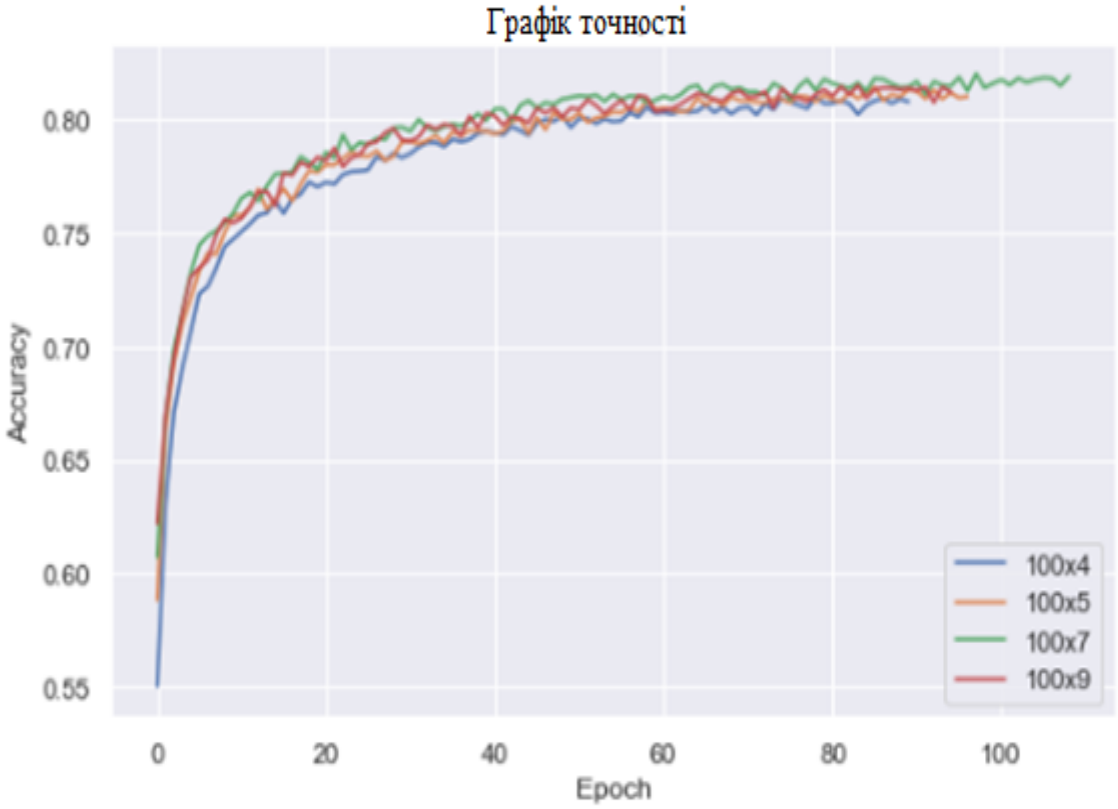

Рис. 4.9 Графік зміни точності, та швидкості навчання, залежно від кількості прихованих шарів

Підібравши необхідні гіперпараметри з якими класифікатор показує найкращий за точністю результат, спробуємо класифікувати гірські породи які в сліпій свердловині, що була вирізана із загального тренувального набору даних.

4.3 Застосування нейронної мережі

Дані свердловин для навчання та перевірки, були отримані в результаті комплексу геофізичних методів. Які проведені на території біля Норвежського узбережжя Північного моря, зображено на рисунку 4.10. Свердловини які були використані у роботі, здебільшого розташовані у квадраті 25. Нафтова та газова промисловість на шельфі Північного моря біля Норвегії є однією з найбільш

розвинених та значущих галузей господарства цього регіону. Цей район славиться своїми великими нафтовими та газовими резервами, які вимагають високого рівня технологій, досліджень та видобутку для ефективного використання цих природних ресурсів. Геологія цього регіону включає ряд особливостей, які мають важливе значення для нафтової та газової промисловості.

Структура шельфу: Шельф Північного моря біля Норвегії має складну геологічну структуру. Він включає гірські формації, вулканічні області, розломи та плити. Ці структурні особливості утворюють потенційні резервуари для нафти та газу. Після закінчення льодовикових періодів, регіон шельфу Північного моря був покритий післяльодовиковими осадами. Ці осади включають піски, гравій та глину, які створюють потенційні резервуари для нафтових і газових відкладень.

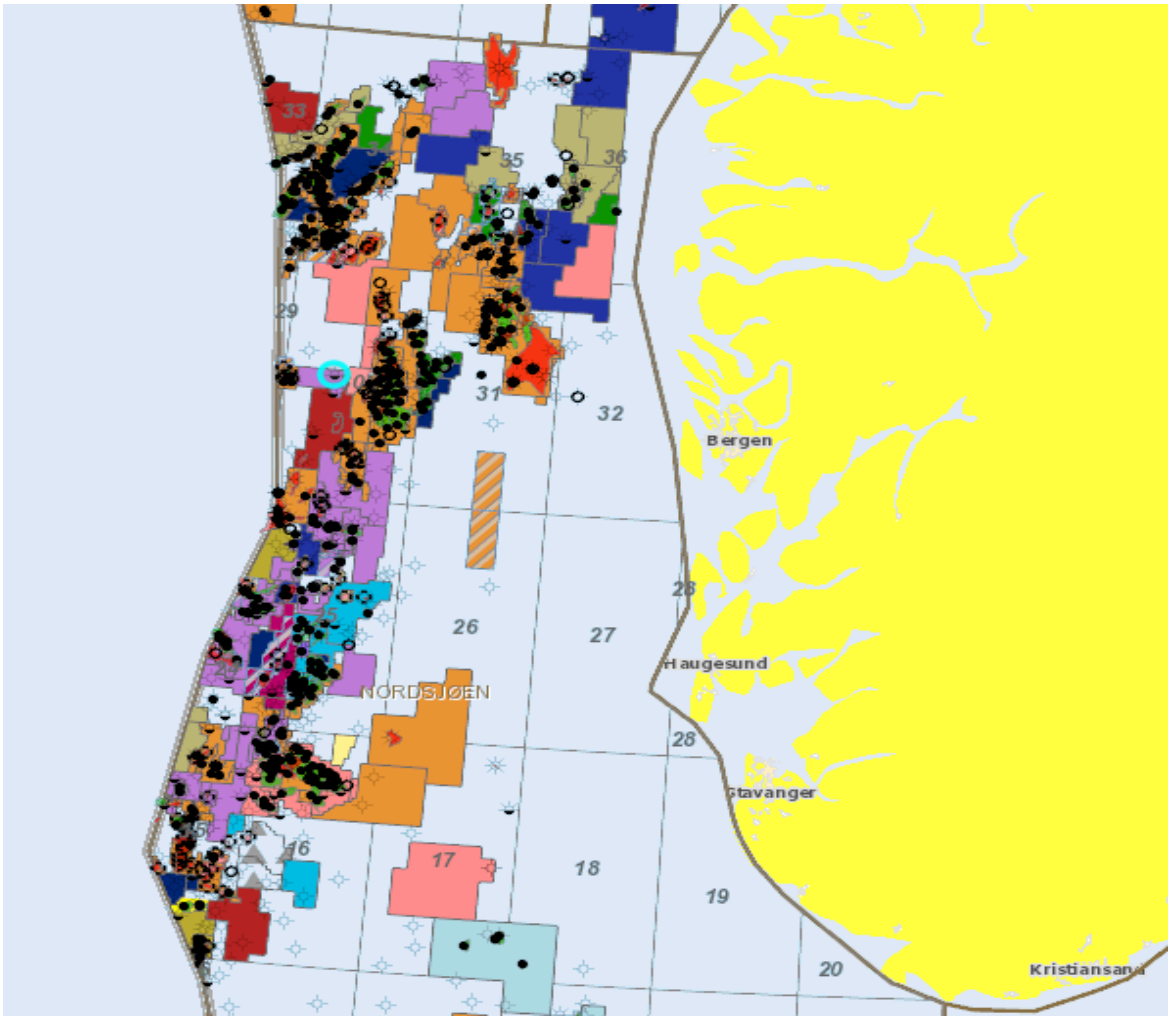

Рис 4.10 Місця свердловин які були використані для дослідження

У досліджуваних свердловинах використовувався комплекс методів, який дає повне уявлення про фізичні властивості залягаючих порід, що також дає змогу визначити фації.

Добуток швидкості компресійних хвиль (DTC): DTC (Density-Compensated Time) є одним з основних геофізичних параметрів, який вимірюється при дослідженні свердловини. Він представляє собою швидкість поширення компресійних хвиль у гірських породах. Значення DTC використовується для визначення інших геофізичних властивостей порід, таких як їх густини та проникності.

Нейтронний фізичний показник (NPHI): NPHI (Neutron Porosity Index) є мірою пористості гірських порід. Цей метод вимірює кількість води, яка може міститися в породах. NPHI використовується для визначення пористості та проникності порід, що є важливими параметрами для оцінки резервуарів нафти і газу. Диференційна щільність (DRHO): DRHO (Density Difference) є вимірюванням різниці щільності між гірськими породами та рідинами, що їх заповнюють. Цей параметр використовується для визначення границь між різними типами порід та для ідентифікації наявності нафти або газу.

Густина порід (G/CC): Густина порід є мірою маси одиниці об'єму гірських порід. Цей параметр вимірюється в г/см³ і використовується для визначення типу порід (наприклад, пісковик, вапняк, сланець) і їх складу.

Гамма-радіаційний індекс (GR gAPI): GR (Gamma Ray) gAPI вимірює рівень гамма-випромінювання, що випромінюють гірські породи.

Зворотний потенціал (SP): SP (Spontaneous Potential) вимірює електричний потенціал, що виникає між зонами з різною проникністю порід. Цей потенціал виникає через рух заряджених частинок в породах, викликаний різницею в проникності. Вимірювання SP допомагає ідентифікувати границі різних порідних зон, з'ясувати наявність розривів та розрізнити проникні породи від ізоляційних.

Опір порід (R OHM): R OHM (Resistivity) є мірою опору, який породи протиставляють потоку електричного струму. Вимірювання опору дозволяє оцінити проникність порід та їх геологічні властивості, включаючи наявність рідини (наприклад, води, нафти або газу) у породах. Зміна опору допомагає ідентифікувати пористість, породу та потенційні пласти-колектори.

Після підбору оптимальних параметрів, мережу використали для класифікації літології в свердловині, що була використана для перевірки її ефективності. Розподіл класів у такій свердловині представлений на рисунку 4.11.

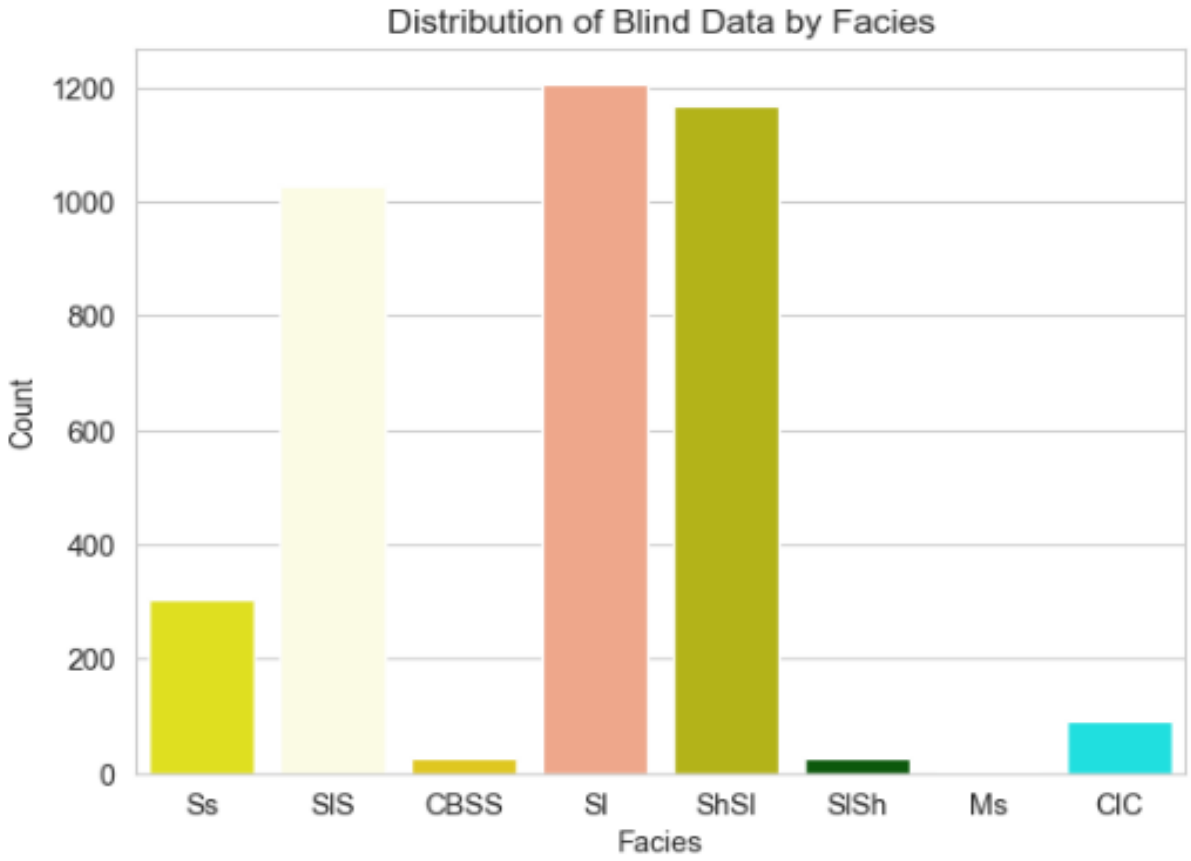

Рис 4.11 Розподіл класів у контрольній свердловині

В цій свердловині ми можемо врахувати не тільки числові показники точності, але й візуальну складову, що дозволяє оцінити точність найменших деталей розчленування. Для відображення точності класифікації на контретних класах,

створено матрицю незгідностей, яка відображає відношення прогнозованого класу до реального, зображено на рисунку 4.12.

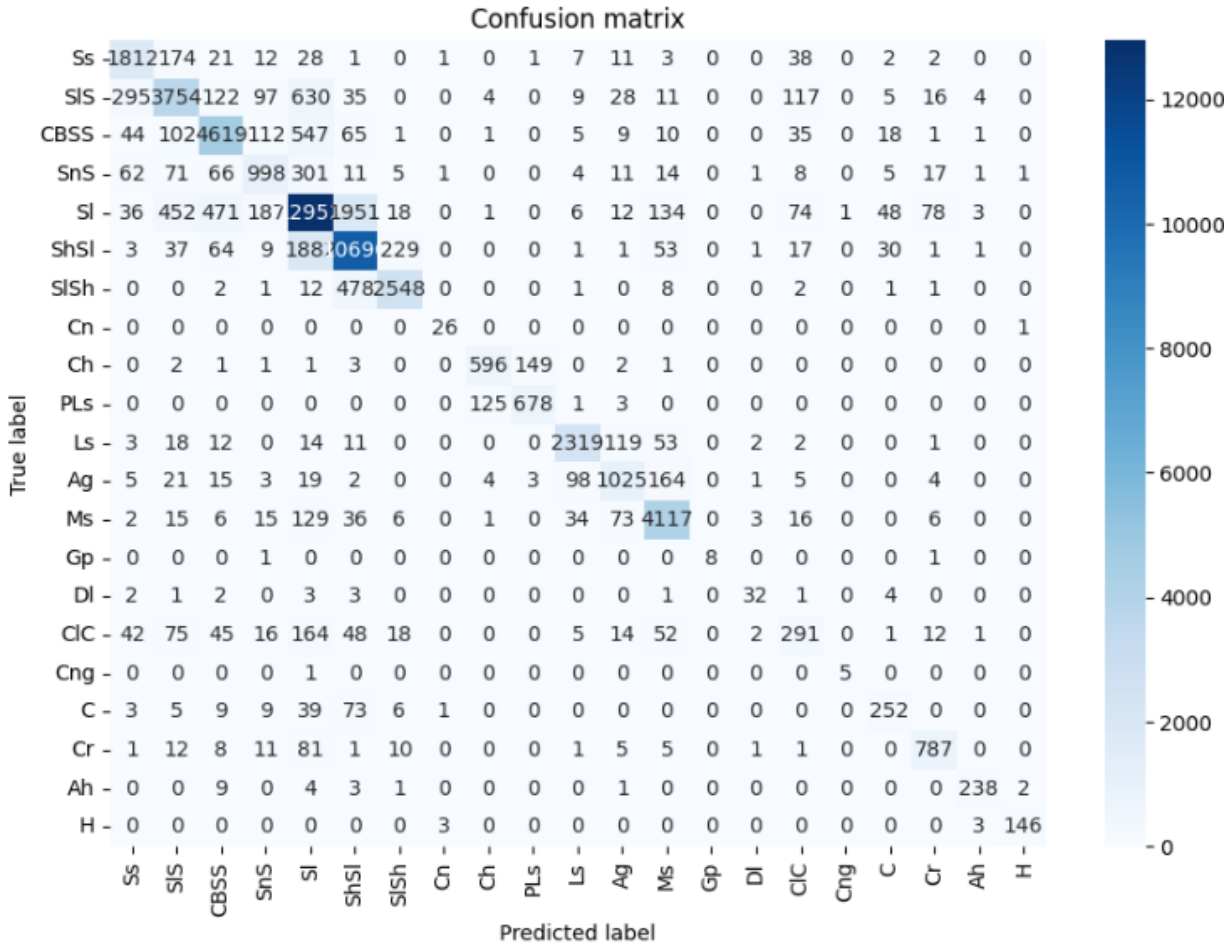

Рис 4.12 Матриця незгідностей після навчання мережі

Ця матриця незгідностей відображає результати класифікації геологічних формацій за допомогою певної системи фацій. У першому рядку і стовпці вказані передбачувані (Pred) та правильні (True) фації. Решта комірок містить кількість випадків, коли певна формація була помилково класифікована як інша фація.

Наприклад, у першому рядку, модель передбачила, що 1939 зразків належать до фації "Пісковик" (Ss), і для 165 зразків вона помилково визначила іншу фацію. У цьому ж рядку в комірці CBSS знаходиться число 21, що означає, що модель помилково класифікувала 21 зразків як "Шаруватий пісковик" (CBSS), якщо насправді вони належать до іншої фації.

У кінці кожного рядка та стовпця наведена сума помилкових класифікацій для кожної фації, а в останній комірці знаходиться загальна кількість помилкових класифікацій. У цьому випадку, згідно з матрицею, точність класифікації фацій становить 81.57%, а точність класифікації суміжних фацій (Adjacent facies classification accuracy) - 88.10%.

Матриця незгідностей на рисунку 4.12 (англ. confusion matrix) є інструментом для оцінки ефективності алгоритмів машинного навчання в задачах класифікації. Вона представляє собою таблицю, в якій рядки відповідають істинним міткам класів, а стовпці - передбаченим міткам класів. Кожна клітина матриці показує кількість екземплярів, для яких істинний клас дорівнює рядку, а передбачений - стовпцю.

Як видно на матриці, навіть класи з малою кількістю екземплярів мають досить точне передбачення, на прикладі галіту (H). В порівнянні з кількістю інших класів, галіт має дуже малу вибірку, але навіть такої за обсягом вибірки вистачило для навчання із високою точністю, по галіту відхилення близько 2%. Інші класи, які складають велику частину всієї вибірки, в основному мають високу точність визначення, але деякі екземпляри помилково визначаються до суміжних класів. Інші класи також відображуються із високим показником точності, що в загальному цілком відображає отримані результати.

Результат класифікації відображений на планшеті з кривими (рис. 4.13), де у стовбці Facies зображено фактичну літологію свердловини, а у стовбці Prediction - літологічне розчленування, зроблене нейронною мережею. Для оцінки ефективності класифікатора було зроблено дві гістограми з кількістю екземплярів класів: одна зображує розподіл класів за результатами відбору керну, а інша - за результатами класифікації.

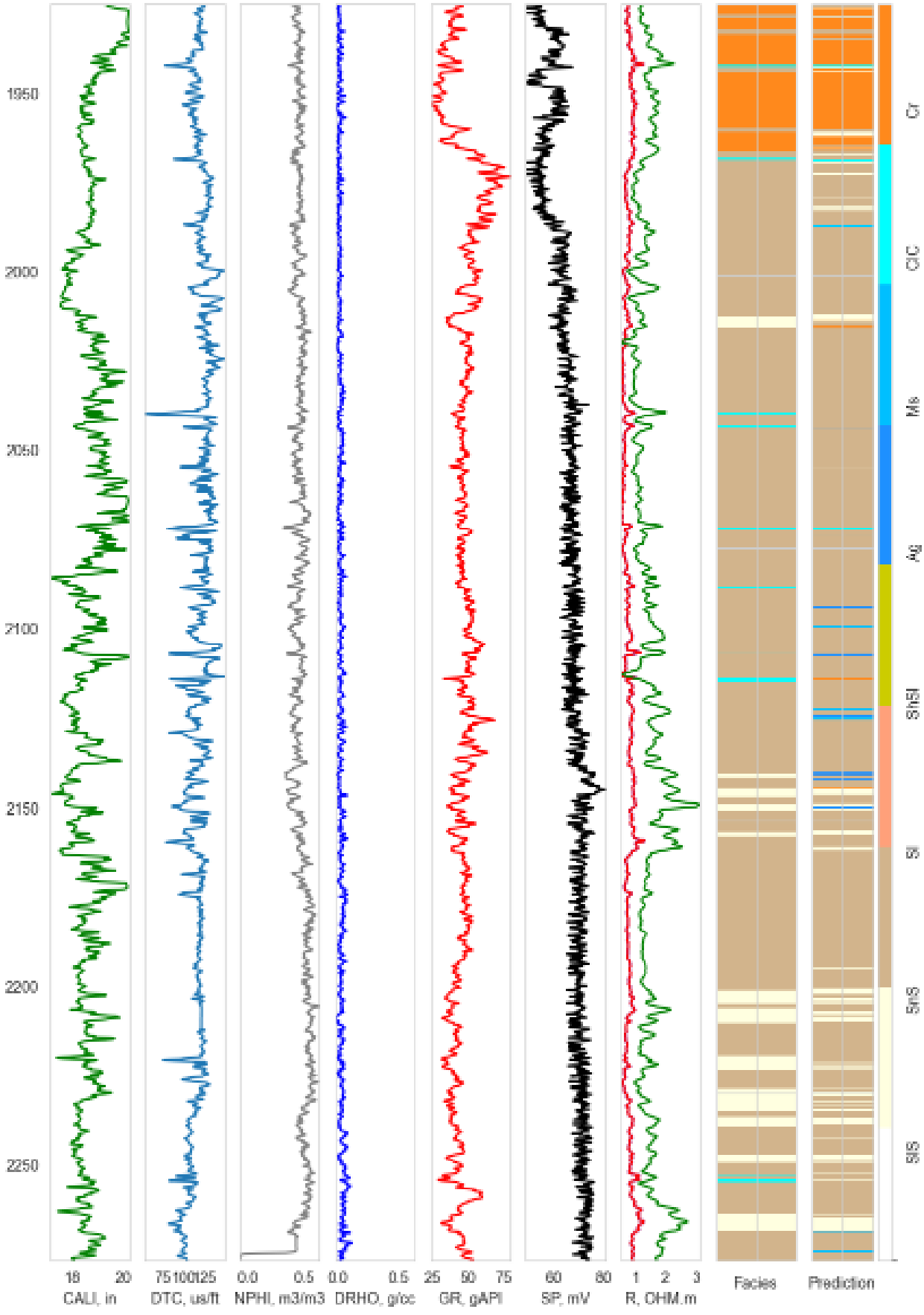

Рис 4.13 Планшет кривих, шельф Північного моря

Літологія по досліджуваній свердловині в інтервалі глибин 1900 - 2275м, представлена здебільшого осадовими породами, переважно мулом (Sl). В інтервалі 1900 - 1950м виділяється вулканічний туф (Cr) із прошарками вапнякового цементу (ClC). Класифікатор точно визначив літологію цього інтервалу, враховуючи прошарки із незначною потужністю. Стосовно решти досліджуваного інтервалу, то тут здебільшого мул із прошарками пісчаного мулу (SnS), вапнякового цементу (ClC). Прошарки із схожими літологічними характеристиками складніше класифікувати, що видно із літологічної колонки класифікатора. В інтервалах де зустрічаються прошарки пісчаного мулу 2150 - 2160м та 2200 - 2275м класифікатор показує певні результати і визначає пропластки, але точність тут не максимальна. Він помилково може визначати глибину залягання таких пропластків, але класифікує точно породу.

Чого не можна сказати про класифікацію у інтервалі 2100 - 2150м, де класифікатор помилково визначає прошарки Аргіліту, якого взагалі немає у досліджуваному інтервалі. Отже, можна зробити висновок, що класифікатор має певні незначні складнощі у точності визначення залягання пропластків, за умови, що в певному інтервалі є декілька пропластків різної потужності.

Для перевірки роботи класифікатора було також використано дані із Причорноморського регіону (рис. 4.14), де було проведено комплекс методі для геофізичного дослідження свердловини. Геологія цього регіону складна та має складний геологічний профіль. Він складається з різноманітних гірських порід, що утворюються від еоцену до кінця кайнозою. Загальна структура регіону складається з кристалічної основи та покривів відкладеннями.

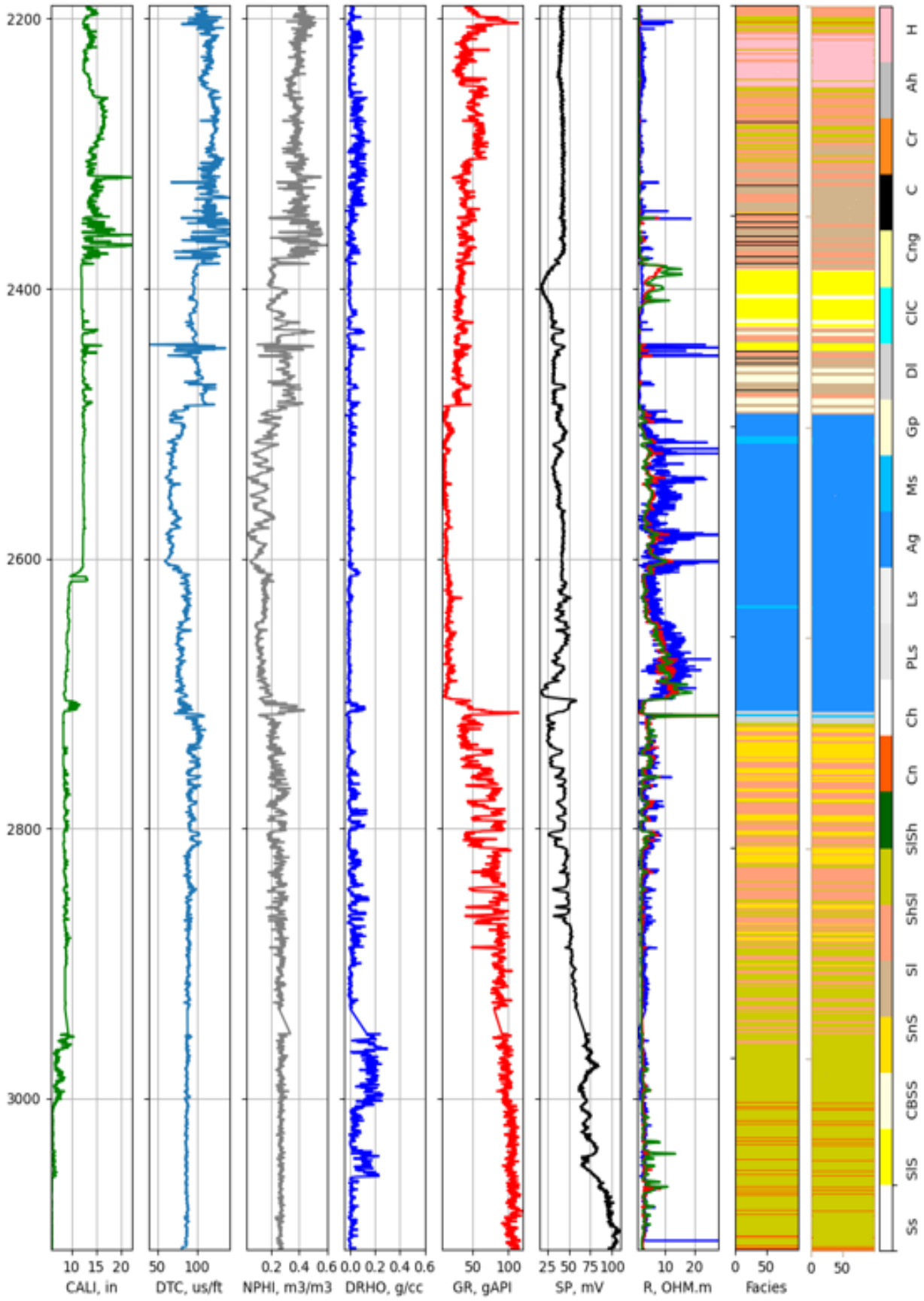

Рис 4.14 Планшет кривих Причорноморський регіон

На північному заході прибережна рівнина складається з відкладень річкових та морських відкладень голоцену. Далі на північний схід відбувається підвищення дна моря, тому донний рельєф представлений найбільшим узбережжям моря з відкладень голоцену і пліоцену. На схилі підводного возвища знаходяться відклади міоцену та пліоцену.

На південному заході регіон представлений чергуванням південноукраїнського кристалічного щита з басейном Причорноморської западини. У басейні Причорноморської западини відкладення товще, як правило, з глин, а в більш глибоких зонах - зваженість переважно складається з пісків та глин мелівської та палеоценової формацій.

Також у регіоні присутні солончакові відкладення в западинному басейні, а також глибинні ділянки з карбонатними породами та великими ділянками розривів в північній та південній частині регіону.

Як видно на планшеті кривих, класифікатор допустився помилки в пропластках 2200 – 2250 м, 3300 – 3400 м та в інтервалі 2500 -2700 м. Тобто, модель робить деякі помилки на пропластках малої потужності.

ВИСНОВКИ

Під час підготовки вибірки на самому початку роботи, було виявлено, що багато свердловин не відповідають вимогам, які необхідні для максимально точної класифікації. Ці вимоги включають криві комплексу методів, який описаний у другому розділі, їх цілісність та наявність літології. Деяка частина даних, отриманих з джерела, не була включена до вибірки.

Результати експериментів, де були змінені параметри, показали, що початкове значення має неоднозначність залежно від обсягу вибірки. Тому оптимальна кількість шарів та нейронів в них буде залежати від кількості векторів на вході. Це проблема, яка вже потребує розв'язання в галузі глибокого навчання.

Результати були значно вплинуті ще одним важливим фактором - значною різницею в розподілі класів. У деяких класах було дуже мало екземплярів, що призвело до зниження точності класифікатора. Такі свердловини будуть мати меншу точність порівняно з тими, в яких кількість екземплярів класів велика. Це пов'язано з низькою кількістю міток класів, які були використані при навчанні, тому що вибірка була розділена, і, фактично, інформації було ще менше, ніж на гістограмі розподілу кількості класів.

Варто розуміти, що такий гострий перехід від однієї фації до іншої, не завжди наявний у реальному середовищі. Тому слід зробити поправку на суміжні фації, що можуть залягати поруч із породою та мають схожі петрофізичні властивості, і тим самим врахувати деякі незгідності між класами, як правильно зроблену класифікацію. Це слід враховувати на етапі визначення точності класифікатора.

Загалом, кількість класів є досить значною, що дозволяє детально розчленовувати свердловину. Але за такої кількості класів, необхідний великий об’єм вибірки, для повноцінного навчання. Слід зазначити, що досліджувані свердловини, і взагалі досліджувана територія, є шельфом моря, тобто готову

мережу не слід використовувати до каротажних даних які отримані наприклад в межах суходолу. Це обумовлено типом та осадко накопиченням гірських порід, петрофізичні властивості яких будуть серйозно відрізнятися від порід на суші.

При класифікації свердловини із літологією, отримано значення 86% точності, без урахування суміжних фацій. Розроблену модель слід використовувати для літологічного розчленування, але з деякими поправками на об’єм вибірки, для якої потрібно більше вузлів, і відповідно більше обчислювальної потужності, та особливості досліджуваних даних.

Для вирішення задачі класифікації була створена нейронна мережа мультиперцептрон, що містить три прихованих шари, кожен з яких містить 200 нейронів. Ця архітектура була вибрана на основі принципу глибокого навчання, який максимізує можливість виявлення складних залежностей в даних. Функція активації, що використовується в цих шарах, - це ReLU (Rectified Linear Unit), яка добре працює в багатошарових мережах і допомагає зберегти градієнт великим при навчанні.

Спосіб оптимізації, що використовується для навчання мережі, - це "adam", який є покращеним методом стохастичного градієнтного спуску. Параметр "альфа" було встановлено на 0.004, що вказує на швидкість навчання мережі. Розмір пакету для навчання встановлено на 235, що означає, що під час кожного кроку оновлення ваг використовується 235 прикладів з навчального набору даних.

Швидкість навчання визначена як стала, з початковим значенням 0.001, що вказує на початковий крок для оновлення ваг мережі. Максимальна кількість ітерацій навчання встановлена на 500, що означає, що мережа навчатиметься не більше 500 епох.

Дані для навчання були впорядковані випадковим чином ("shuffle" = True), що забезпечує більш ефективне навчання, запобігаючи певному порядку, який може

виникнути в даних. Застосування техніки "warm_start" = False означає, що мережа не використовує ваги з попереднього навчання при повторному використанні.

У загальному, ця архітектура та параметри навчання були обрані з метою найбільш ефективного використання навчальних даних, з максимальною точністю класифікації свердловин з літологією в 86%, без урахування суміжних фацій. Однак слід зауважити, що результати можуть змінюватися в залежності від конкретного набору даних, що використовуються для навчання мережі, а також від специфічних особливостей досліджуваних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Ю.М. Заворотько, (2010), Фізичні основи геофізичних методів дослідження свердловини .

В. М. Курганський, І. В. Тішаєв, (2011), Електричні та Електромагнітні методи дослідження свердловин, Київ, Київський університет.

Charu C. Aggarwal, (2018), Neural Networks and Deep Learning: A Textbook

Chris Albon, (2018), Machine Learning with Python Cookbook, Sebastopol, O'Reilly Media, Inc.

Christopher Bishop, (2006), Pattern Recognition and Machine Learning

Darwin V. Ellis, Julian M. Singer, (2007), Well Logging for Earth Scientists

Eugene Charniak, (2018), Introduction to Deep Learning, Massachusetts The MIT Press

Giuseppe Ciaburro, Prateek Joshi, (2019), Python Machine Learning Cookbook Second Edition, Birmingham, Packt Publishing Ltd.

Hu, L., Zheng, X., Duan, Y., Yan, X., Hu, Y., Zhang, X., (2019). First arrival picking with a U-net convolutional network. Pekin, Geophysics.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, (2016), Deep Learning

J. Wang, (2017), Lithofacies Classification using Machine Learning

Joel Grus, (2015), Data Science from Scratch First Principles with Python 1st Edition, Sebastopol, O`Reilly Media Inc.

Joseph R. Hearst, Philip H. Nelson, Frederick L. Paillet, (2000), Well Logging for Physical Properties

Kyran Dale, (2016), Data Visualization with Python and JavaScript, Sebastopol, O'Reilly Media, Inc.

Michael Nielsen, (2015), Neural Networks and Deep Learning

Raul Rojas, (1996), Neural Networks: A Systematic Introduction

Simon Haykin, (2008), Neural Networks and Learning Machines

Trevor Hastie, Robert Tibshirani, Jerome Friedman, (2009), The Elements of Statistical Learning: Data Mining, Inference, and Prediction

W. Sandham, M. Leggett (2003) Geophysical Applications of Artificial Neural Networks and Fuzzy Logic

William Sandham, Miles Leggett ,(2003), Geophysical Applications of Artificial Neural Networks and fuzzy logic, Glasgow, Springer science business media, b. V.

Grzegorz Nowakowski,Yaroslaw Dorogyy, Olena Doroga-Ivaniuk, (2017), Neural Network Structure Optimization Alghorithm, Journal of Automation, Mobile Robotics & Intelligent Systems.

Huang, L., Dong, X., Clee, T.E., (2017). A scalable deep learning platform for identifying geologic features from seismic attributes. The Leading Edge 36, 249-256.

Huang, S.J., Zhang, J., Cheng, L.S., (2015). A new formula for calculating the productivity of fracturing directional wells in low permeability reservoirs. Journal of Xian Shiyou University.

Ioffe, S., Szegedy, C., (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, International conference on machine learning, pp. 448-456.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15, 1929-1958.

Sun, H., Demanet, L., (2018). Low frequency extrapolation with deep learning, SEG Technical Program Expanded Abstracts 2018. Society of Exploration Geophysicists, pp. 2011-2015.

Sun, J., Slang, S., Elboth, T., Larsen Greiner, T., McDonald, S., Sebastian Rashka, (2015), Python machine learning, Birmingham: Packt Publishing Ltd.

Tarun Kumar Gupta, Khalid Raza, (2018), Optimizing Deep Neural Network Architecture, Department of Computer Science, Jamia Millia Islamia

Zed A., (2013), Shaw Learn Python the hard way: a very simple introduction to

the terrifying beautiful world of computers and code – Third edition, Crawfordsville: RR Donnelley.

Z. Fana , X. Xua, (2019), Application and visualization of typical clustering algorithms in seismic data analysis

Zhang, G., Wang, Z., Chen, Y., (2018). Deep learning for seismic lithology prediction. Geophysical Journal International 215, 1368-1387.

Zhang, M., Liu, Y., Bai, M., Chen, Y., (2019). Seismic Noise Attenuation Using Unsupervised Sparse Feature Learning. IEEE Transactions on Geoscience and Remote Sensing, 1-15.

Zhang, Y., Lu, P., Yu, H., Morris, S., (2019). Enhancement of seismic imaging: An innovative deep learning approach. arXiv preprint arXiv:1909.06016.

Zhao, T., 2018. Seismic facies classification using ANN, arXiv preprint arXiv:1894.00012

Benjamin Bryan Bougher, (2016), Machine learning applications to geophysical data analysis, B.Sc. in Physics, Dalhousie University

A. Ruediger, F.Rosei Interfaces: AFM extends its reach Nature Nanotechnology, 5 (2010) 388.

C. Nauenheim, A. Ruediger, C. Kuegeler, R. Waser Investigation of the electroforming process in resistively switching TiO2 nanocrosspoint junctions Applied Physics Letters, 96 (2010) 122902

Liner, Christopher L.and McGilvery, Thomas, The Art and Science of Seismic Interpretation, 2017, Springer, c. 140 p (in preparation)

Liner, Christopher L., Elements of Seismic Dispersion: A somewhat practical guide to frequency-dependent phenomena, 2012, Society of Exploration Geophysicists, 179 p

Додаток А

```
# import modules

"""""""""""
Цей код імпортує необхідні модулі для проекту, такі як Pandas,
numpy, і matplotlib.
Він також встановлює деякі параметри для панди, такі як кількість
рядків для показу,
і встановлює деякі параметри для numpy, щоб ігнорувати певні
помилки.

Потім він завантажує тренувальний набір даних з CSV-файлу за
допомогою функції pd.read_csv()
і присвоює стовпцю  категорійну змінну за допомогою функції
astype().
Функція descripted () використовується для відображення основної
статистики набору даних.
Комірка коду відображає назви стовпчиків набору даних за допомогою
атрибута стовпчиків.
"""""""""""

# import modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

# Load training dataset
filename = 'training_data.csv'
training_data = pd.read_csv(filename, index_col=0)
training_data['well'] = training_data['well'].astype('category')

# Display dataset information
print(training_data.info())

# Display the shape of the dataset
print(training_data.shape)
```

```
"""""""""""
Цей код визначає список лейблів (підписів) і відповідний список
кольорів для
кожного підпису. Після цього він створює словник,
який відображає кожну мітку фейс до відповідного її кольору.
Потім він визначає функцію `label_facies()`,
яка приймає в ряд тренувальні дані і список міток,
і повертає мітку, відповідну значенню в стовпці 'LITHOLOGY'
рядка. Потім він застосовує функцію `label_facies()`
до кожного рядка тренувальних даних, використовуючи функцію
`Apply()`,
для створення нового стовпчика 'FaciesLabels' в тренувальних даних
DataFrame,
```

```
з відповідними фазами для кожного рядка.
"""""""""

#create map with labels and colors
facies_labels = ['Ss', 'SlS', 'CBSS', 'SnS', 'Sl',
                 'ShSl', 'SlSh', 'Cn', 'Ch', 'PLs',
                 'Ls', 'Ag', 'Ms',  'Gp', 'Dl',
                 'ClC', 'Cng', 'C', 'Cr', 'Ah',
                 'H']
facies_colors = ['#FFFF00', '#FFFFE0', '#FFDF00', '#D2B48C',
'#ffa07a',
                 '#CCCC00', '#006400', '#FF5B02', '#FFFFFF',
'#E8E8E8',
                 '#F0F0F0', '#1e90ff', '#00bfff', '#FFFDD2',
'#D3D3D3',
                 '#00FFFF', '#FFFF99', '#000000', '#FF891C',
'#BEBEBE',
                 '#ffc0cb']

facies_color_map = {}
for ind, label in enumerate(facies_labels):
    facies_color_map[label] = facies_colors[ind]


def label_facies(row, labels):
    return labels[row['LITHOLOGY'] - 1]


training_data.loc[:, 'FaciesLabels'] = training_data.apply(lambda
row: label_facies(row, facies_labels), axis=1)
"""""""""

#create map with labels and colors
facies_labels = ['Ss', 'SlS', 'CBSS', 'SnS', 'Sl',
                 'ShSl', 'SlSh', 'Cn', 'Ch', 'PLs',
                 'Ls', 'Ag', 'Ms',  'Gp', 'Dl',
                 'ClC', 'Cng', 'C', 'Cr', 'Ah',
                 'H']
facies_colors = ['#FFFF00', '#FFFFE0', '#FFDF00', '#D2B48C',
'#ffa07a',
                 '#CCCC00', '#006400', '#FF5B02', '#FFFFFF',
'#E8E8E8',
                 '#F0F0F0', '#1e90ff', '#00bfff', '#FFFDD2',
'#D3D3D3',
                 '#00FFFF', '#FFFF99', '#000000', '#FF891C',
'#BEBEBE',
                 '#ffc0cb']

facies_color_map = {}
for ind, label in enumerate(facies_labels):
    facies_color_map[label] = facies_colors[ind]
```

```
def label_facies(row, labels):
    return labels[row['LITHOLOGY'] - 1]


training_data.loc[:, 'FaciesLabels'] = training_data.apply(lambda
row: label_facies(row, facies_labels), axis=1)
```

```
"""""""""""
Цей код створює нову DataFrame під назвою `blind`,
яка містить тільки рядки з тренувальних даних DataFrame де стовпчик
"Well"
вміщує назву свердловини
Потім він відфільтровує ті рядки з оригінальних тренувальних даних
DataFrame, щоб вони не були включені в подальший аналіз.
Далі відраховує кількість унікальних записів для кожної фації
в оновлених тренувальних даних DataFrame, сортує їх за
номером (замість кількості записів), і присвоює фаціям підписи як
індекс facies_Counts DataFrame. Потім він друкує «зразки графа» і
розмальовує штрих-діаграму факсів, використовуючи facies_colors list
як кольори поділок, і встановлює назву сюжету на «розподіл
тренувальних
даних за фасисами» (Distribution of Training Data by Facies).
"""""""""""
print(training_data['well'].unique)
blind = training_data[training_data['well'] == '7_3-1']
print(blind['LITHOLOGY'])
training_data = training_data[training_data['well'] != '7_3-
1']#delete blind data  from training set
#count the number of unique entries for each facies, sort them
# facies number (instead of by number of entries)
facies_counts =
training_data['LITHOLOGY'].value_counts().sort_index()

# use facies labels to index each count
facies_counts.index = facies_labels
print("Count facies samples")
facies_counts.plot(kind='bar', color=facies_colors,
                   title='Distribution of Training Data by Facies')
plt.show() # facies counts
blind['LITHOLOGY']
```

```
#facies distribution for blind data
"""""""""""
Цей код створить гістограму розподілу класів (або фацій) в наборі
даних сліпої свердловии,
при цьому вісь x показує різні фактичні мітки і вісь y, що показує
кількість кожної грані.
Він спочатку створює нову DataFrame під назвою `blind_counts`, яка
містить кількість унікальних
```

```
записів для кожної фації в сліпому колоді DataFrame, відсортованому.
Далі визначається функція `facies_LB_col(сліпі, мітки, кольори)`,
яка приймає в `сліпі` дані,
список `labels`, і список `кольорів`. Функція сортує унікальні
значення фацій в незрячих даних,
потім створює два порожні списки `facies_lb` і `facies_col` і ітерує
над відсортованими унікальними значеннями.
Після цього він додає відповідну мітку і колір граней до двох
списків.
"""""""""

import seaborn as sns

# Count values and sort by index
blind_counts = blind['LITHOLOGY'].value_counts().sort_index()

# Get facies labels and colors
facies_lb, facies_col = facies_lb_col(blind['LITHOLOGY'],
facies_labels, facies_colors)

# Set plot style
sns.set_style('whitegrid')

# Create bar plot
sns.barplot(x=facies_lb, y=blind_counts.values, palette=facies_col)

# Set plot title and axis labels
plt.title('Distribution of Blind Data by Facies')
plt.xlabel('Facies')
plt.ylabel('Count')

# Show the plot
plt.show()
```

```
## Підготовка до класифікації
На цьому етапі відбувається видалення даних, що не приймають участь
в класифікації, нормалізація, та розподіл тренувальної вибірки на
тестову, яка складає 25% від тренувальої вибірки.
Створення масиву для додавання суміжних фацій, для матриці
незгідностей.
#%%
# prepare to classification
correct_facies_labels = training_data['LITHOLOGY'].values

feature_vectors = training_data.drop(
    ['LITHOLOGY', 'well', 'DEPT', 'FaciesLabels'], axis=1)
feature_vectors.describe()

#PREPROCESSING, NORMALIZATION
from sklearn import preprocessing
```

```
scaler = preprocessing.StandardScaler().fit(feature_vectors)
scaled_features = scaler.transform(feature_vectors)

from sklearn.model_selection import train_test_split

#SPLIT DATASET
X_train, X_test, y_train, y_test = train_test_split(
    scaled_features, correct_facies_labels, test_size=0.25)


adjacent_facies = [[1], [0], [0,1], [4], [3], [4,3], [5], [17],
[9,10], [8],
                                [8,9], [9], [10], [10], [10,19], [13],
[0], [5,6], [16], [17,10], [10]]
print(adjacent_facies)
```

```
"""
код використовується для підбору оптимальних значень гіперпараметрів
для нейронної мережі,
що використовується для класифікації даних. Для цього
використовується RandomizedSearchCV з
бібліотеки scikit-learn, який випадковим чином вибирає комбінації
гіперпараметрів з деякого діапазону
і оцінює результат класифікації за допомогою крос-валідації.
Параметри для пошуку передаються у вигляді словника
param_dist, який містить діапазони і/або допустимі значення для
кожного гіперпараметра.

Нейронна мережа для класифікації задається за допомогою
MLPClassifier з бібліотеки scikit-learn.
Перед початком підбору гіперпараметрів, дані X_train та X_test
нормалізуються за допомогою preprocessing.normalize().

Після завершення пошуку, найкращі параметри зберігаються у
random_search.best_params_,
а найкращий результат класифікації - у random_search.best_score_. Ці
результати можуть
бути використані для підвищення точності класифікації даних за
допомогою підбору оптимальних
гіперпараметрів для нейронної мережі. Логування (logging)
використовується для моніторингу процесу підбору гіперпараметрів.
"""

from sklearn.model_selection import RandomizedSearchCV
import logging

import scipy.stats as stats
from sklearn.neural_network import MLPClassifier

param_dist = {'hidden_layer_sizes': stats.randint(50, 200),
              'activation': ['tanh', 'relu'],
              'solver': ['adam', 'sgd'],
```

```
              'alpha': stats.uniform(0.001, 0.01),
              'batch_size': stats.randint(200, 800),
              'learning_rate': ['constant', 'invscaling',
'adaptive'],
             'max_iter': [100]}

clf = MLPClassifier(verbose=True)

X_train = preprocessing.normalize(X_train)
X_test = preprocessing.normalize(X_test)

random_search = RandomizedSearchCV(clf,
param_distributions=param_dist, n_iter=100, cv=3, verbose=True)
logging.info("Starting randomized search")

random_search.fit(X_train, y_train)

logging.info("Randomized search completed")
logging.info("Best parameters found: %s",
random_search.best_params_)
logging.info("Best score found: %s", random_search.best_score_)

#збереження результатів підбору гіперпараметрів в csv файл

import pandas as pd
results = pd.DataFrame(random_search.cv_results_)
results.to_csv('random_search_results.csv', index=False)
print(random_search.best_params_)

#%%
"""
Цей код використовує бібліотеку Scikit-learn для побудови нейронної
мережі
класифікатора з вказаними параметрами і для передбачення класів для
тестових даних.

Спочатку, визначається об'єкт MLPClassifier, з параметрами, такими
як розмір прихованих шарів,
функція активації, метод оптимізації, коефіцієнт регуляризації та
інші.

Далі, тренувальні і тестові дані нормалізуються за допомогою
normalize() з бібліотеки Scikit-learn.

Потім, класифікатор навчається з використанням навчальних даних за
допомогою fit().
Після цього передбачення класів виконується для тестових даних з
використанням predict().

Нарешті, виводяться параметри, отримані за допомогою get_params(),
звіт про середній бал
точності навчальної вибірки за допомогою score() та логарифмовані
```

```
ймовірності передбачених
класів навчальної вибірки з використанням predict_log_proba(). Також
передбачені класи для
тестових даних виводяться за допомогою predict().
"""


import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(hidden_layer_sizes = (200, 200, 200),
                    activation = "relu", solver = "adam",
                    alpha = 0.004, batch_size = 235,
                    learning_rate = 'constant', learning_rate_init =
0.001,
                    max_iter = 500, shuffle = True,
                    verbose = True, early_stopping = True,
warm_start = False )

X_train = preprocessing.normalize(X_train)
X_test = preprocessing.normalize(X_test)

clf.fit(X_train, y_train)

params = clf.get_params()
score = clf.score(X_train, y_train)
pred_prob = clf.predict_log_proba(X_train)
predicted_labels = clf.predict(X_test)

print("Parameters for estimator: \n", params)
print("Score mean accuracy on the given test data and labels: \n",
score)
print("Log of probability estimates: \n", pred_prob)

fig, ax = plt.subplots()
ax.plot(predicted_labels, label='Predicted labels')
ax.legend()
ax.set_title("Comparison of predicted labels")
plt.show()

#%%
import seaborn as sns
import matplotlib.pyplot as plt

facies_counts =
training_data['LITHOLOGY'].value_counts().sort_index()

##use facies labels to index each count
facies_counts.index = facies_labels

sns.barplot(x=facies_counts.index, y=facies_counts.values,
palette=facies_colors)
```

```
plt.title('Distribution of Training Data by Facies')
plt.xlabel('Facies')
plt.ylabel('Count')

plt.xticks(rotation=90)

plt.show() # facies counts
## Імпортування матриці незгідностей, та функцій для вимірювання
точності класифікатора
Матриця незгідностей дозволяє визначити точність класифікатора,
шляхом відношення кількості правильних передбачень до загальної
кількості передбачень. Точність визначається сумою всіх правильних
передбачень до суми всіх передбачень.

`accuracy_adjacent` - функція, яка зараховує передбачення, що
розподілені до суміжних фацій - як правильне передбачення.
#%%
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix

conf = confusion_matrix(y_test, predicted_labels)

# Plotting the confusion matrix using seaborn
plt.figure(figsize=(10,7))
sns.heatmap(conf, annot=True, cmap='Blues', fmt='d',
xticklabels=facies_labels, yticklabels=facies_labels)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion matrix')
plt.show()

def accuracy(conf):
    total_correct = np.trace(conf)
    acc = total_correct / np.sum(conf)
    return acc

def accuracy_adjacent(conf, adjacent_facies):
    nb_classes = conf.shape[0]
    total_correct = 0.
    for i in range(nb_classes):
        total_correct += conf[i][i]
        for j in adjacent_facies[i]:
            total_correct += conf[i][j]
    return total_correct / np.sum(conf)

print('Facies classification accuracy =
{:.3f}'.format(accuracy(conf)))
print('Adjacent facies classification accuracy =
```

```
{:.3f}'.format(accuracy_adjacent(conf, adjacent_facies)))

#%%
def blind_well_facies(y_blind, cv_conf, facies_labels):
    x = y_blind
    x = pd.Series(x)
    x = x.sort_values()
    x = x.unique()
    cv_conf.shape[0]
    print(x, y_blind)
    facies_well_lb = []
    for i in x:
        i = i - 1
        item = facies_labels[i]
        facies_well_lb.append(item)
    return facies_well_lb


y_blind = blind['LITHOLOGY'].values

#%%
well_features['Prediction']
#X_blind = scaler.transform(well_features)

#%% md
## Перевірка результату на сліпій свердловині.
#%%
"""
y_blind = blind['LITHOLOGY'].values витягує з DataFrame blind
колонку із справжніми мітками фацій
та зберігає її у змінній y_blind як масив numpy.

well_features = blind.drop(['LITHOLOGY', 'well', 'DEPT',
'FaciesLabels', 'Prediction'], axis=1)
видаляє з DataFrame blind непотрібні для передбачення колонки і
зберігає решту функцій (дані з кривих свердловин)
у змінній well_features.

print(well_features.keys()) друкує назви залишених колонок (дані з
кривих свердловин) у консоль.

X_blind = scaler.transform(well_features) масштабує дані з кривих
свердловин
у well_features з використанням раніше навченого об'єкту scaler та
зберігає масштабовані дані у змінній X_blind.

X_blind = preprocessing.normalize(X_blind) нормалізує масштабовані
дані
з кривих свердловин з використанням L2-нормалізації, що масштабує
кожен рядок даних таким чином, щоб його Євклідова норма (L2-норма)
була рівна 1.
```

```
y_pred = clf.predict(X_blind) використовує раніше навчений об'єкт
класифікатора
clf для передбачення міток фацій для нормалізованих даних з кривих
свердловин у X_blind, і зберігає передбачені мітки у змінній y_pred.

blind['Prediction'] = y_pred створює нову колонку з назвою
"Prediction"
у DataFrame blind та зберігає у ній передбачені мітки фацій.

facies_well_lb = blind_well_facies(y_blind, conf, facies_labels)
обчислює матрицю помилок (conf)
та відповідні мітки фацій (facies_labels) для справжніх міток фацій
(y_blind) у DataFrame blind
та зберігає їх у змінній facies_well_lb.
"""



y_blind = blind['LITHOLOGY'].values
well_features = blind.drop(['LITHOLOGY', 'well', 'DEPT',
'FaciesLabels', 'Prediction'], axis=1)
print(well_features.keys())
X_blind = scaler.transform(well_features)
#X_blind  preprocessingnormalize -1,1
X_blind = preprocessing.normalize(X_blind)
y_pred = clf.predict(X_blind)#predict on blind data
blind['Prediction'] = y_pred#create new column, for predicted data
facies_well_lb = blind_well_facies(y_blind, conf, facies_labels);
#%% md
## Графік кривих, із порівнянням  літології blind свердловини, і
передбаченої літології, за даними blind
#%% md
## Матриця незгідностей для blind свердловини, міра точності,
повноти, та Ф1
На завершальному етапі перевіряється точність класифікатора, та
порівнюються літологічні колонки.
#%%
blind_counts = blind['Prediction'].value_counts().sort_index()

def facies_lb_col(blind, labels, colors):
    x = pd.Series(blind)
    x = x.sort_values()
    x = x.unique()
    facies_lb = []
    facies_col = []
    for i in x:
        i = i - 1
        item = facies_labels[i]
        item2 = facies_colors[i]
        facies_lb.append(item)
        facies_col.append(item2)
    return facies_lb, facies_col
```

```
facies_lb, facies_col = facies_lb_col(blind['Prediction'],
facies_labels, facies_colors)
blind_counts.index = facies_lb
blind_counts.plot(kind='bar', color=facies_col,
                  title='Distribution of Predicted Data by Facies')
plt.show()
```

```
"""
compare_facies_plot(logs, compare, facies_col, facies_lb):
Ця функція приймає pandas dataframe (logs), що містить дані зі
свердловин, рядок,
який вказує назву стовпця з передбаченими фациями (compare), список
кольорів (facies_col),
що відповідають міткам фацій, і список міток фацій (facies_lb). Вона
створює графік фацій,
який є візуалізацією даних зі свердловин та передбачених фацій,
розміщених поруч. Функція спочатку
сортує дані зі свердловин за глибиною, створює кольорову мапу зі
списку кольорів, який було надано,
і встановлює діапазон глибин графіка. Потім вона створює масиви, щоб
зберігати дані передбачених та фактичних
фацій, і використовує їх, щоб створити підграфіки для кожної кривої
свердловини та двох графіків фацій. Нарешті,
функція встановлює різні параметри для графіка, включаючи мітки
осей, мітки шкали та заголовок, і повертає графік.

blind_counts = blind['Prediction'].value_counts().sort_index(): Цей
рядок
коду розраховує кількість входжень кожної передбаченої мітки фації в
стовпці Prediction dataframe
blind і зберігає результат у новому dataframe під назвою
blind_counts. Метод value_counts() підраховує кількість
входжень кожного унікального значення в вказаному стовпці, а метод
sort_index() сортує отриманий dataframe за індексом.

"""


def compare_facies_plot(logs, compare, facies_col, facies_lb):
    # make sure logs are sorted by depth
    logs = logs.sort_values(by='DEPT')
    cmap_facies = colors.ListedColormap(
        facies_col)




    ztop = logs.DEPT.min();
    zbot = logs.DEPT.max()




    cluster = np.repeat(np.expand_dims(logs['LITHOLOGY'].values, 1),
100, 1)
    cluster2 = np.repeat(np.expand_dims(logs[compare].values, 1),
100, 1)

    f, ax = plt.subplots(nrows=1, ncols=9, figsize=(10, 15))
```

```
    ax[0].plot(logs.CALI, logs.DEPT, '-g')
    ax[1].plot(logs.DTC, logs.DEPT, '-')
    ax[2].plot(logs.NPHI, logs.DEPT, '-', color='0.5')
    ax[3].plot(logs.DRHO, logs.DEPT, '-', color = 'b', label =
'DRHO')
    ax[4].plot(logs.GR, logs.DEPT, '-', color='r')
    ax[5].plot(logs.SP, logs.DEPT, '-', color='black')
    ax[6].plot(logs.RSHA, logs.DEPT, '-', color = 'b', label =
'RSHA')
    ax[6].plot(logs.RMED, logs.DEPT, '-', color = 'r', label =
'RMED')
    ax[6].plot(logs.RDEP, logs.DEPT, '-', color = 'g', label =
'RDEP')
    im = ax[7].imshow(cluster, interpolation='none', aspect='auto',
                      cmap=cmap_facies)
    im2 = ax[8].imshow(cluster2, interpolation='none',
aspect='auto',
                       cmap=cmap_facies)
    divider = make_axes_locatable(ax[8])

    cax = divider.append_axes("right", size="20%", pad=0.05)
    cbar = plt.colorbar(im2, cax=cax)
    cbar.set_label((30* ' ').join(facies_lb))
    #cbar.set_label((7* ' ').join( [' Ss ', ' SlS', 'CBSS', 'SnS',
'Sl',
               # 'ShSl', 'SlSh', ' Cn ', ' Ch ', 'PLs ',
                #'Ls ', 'Ag', 'Ms',  'Gp', 'Dl',
                # 'ClC ', ' Cng', ' C  ', ' Cr ', ' Ah ',
                # ' H  ']))
    cbar.set_ticks(range(0, 1));
    cbar.set_ticklabels('')


    for i in range(len(ax) - 2):
        ax[i].set_ylim(ztop, zbot)
        ax[i].invert_yaxis()
        ax[i].grid()
        ax[i].locator_params(axis='x', nbins=2)


    ax[0].set_xlabel("CALI, in")
    ax[0].set_xlim(logs.CALI.min(), logs.CALI.max())
    ax[1].set_xlabel("DTC, us/ft")
    ax[1].set_xlim(logs.DTC.min(), logs.DTC.max())
    ax[2].set_xlabel("NPHI, m3/m3")
    ax[2].set_xlim(logs.NPHI.min(), logs.NPHI.max())
    ax[3].set_xlabel('DRHO, g/cc')
    ax[3].set_xlim(logs.DRHO.min(), logs.NPHI.max())
    ax[4].set_xlabel("GR, gAPI")
    ax[4].set_xlim(logs.GR.min(), logs.GR.max())
    ax[5].set_xlabel("SP, mV")
    ax[5].set_xlim(logs.SP.min(), logs.SP.max())
```

```
    ax[6].set_xlabel('R, OHM.m')
    ax[6].set_xlim(logs.RDEP.min(), logs.RDEP.max())
    ax[7].set_xlabel('Facies')
    ax[8].set_xlabel(compare)


    ax[1].set_yticklabels([]);
    ax[2].set_yticklabels([]);
    ax[3].set_yticklabels([])
    ax[4].set_yticklabels([]);
    ax[5].set_yticklabels([])
    ax[5].set_yticklabels([])
    ax[6].set_yticklabels([])
    ax[7].set_yticklabels([])
    ax[7].set_xticklabels([])
    ax[8].set_yticklabels([])
    ax[8].set_xticklabels([])

    f.suptitle('Well: %s' % logs.iloc[0]['well'], fontsize=14,
y=0.99)
    return plt.show()
compare_facies_plot(blind, 'Prediction', facies_col, facies_lb)
plt.savefig('well_compare_res.png')
blind_counts = blind['Prediction'].value_counts().sort_index()


def facies_lb_col(blind, labels, colors):
    x = pd.Series(blind)
    x = x.sort_values()
    x = x.unique()
    facies_lb = []
    facies_col = []
    for i in x:
        i = i - 1
        item = facies_labels[i]
        item2 = facies_colors[i]
        facies_lb.append(item)
        facies_col.append(item2)
    return facies_lb, facies_col


plt.show()
```