

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет
Кафедра інформаційно-вимірювальної техніки**

До захисту допущено:
Завідувач кафедри
_____ Володимир ЄРЕМЕНКО
«___» червня 2023 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Метрологія та вимірювальна
техніка»
спеціальності 152 «Метрологія та інформаційно-вимірювальна
техніка»
на тему: «Програмно-методичний комплекс сегментації
осцилографічних зображень»**

Виконав:
студент IV курсу, групи ПВ – 91
Теличко Ростислав Сергійович

Керівник:
Доцент, к.т.н., доцент
Синиця Валентин Іванович

Консультант з нормоконтролю:
Ст. викладач, к.т.н.
Щербань Анастасія Павлівна

Рецензент:
Професор, д.п.н.
Протасов Анатолій Георгійович

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших
авторів
без відповідних посилань.
Студент _____

КИЇВ – 2023

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Приладобудівний факультет
Інформаційно-вимірювальних технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 152 «Метрологія та інформаційно-вимірювальна техніка»

Освітньо-професійна програма «Метрологія та вимірювальна техніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Володимир ЄРЕМЕНКО

«22» лютого 2023р.

ЗАВДАННЯ

на дипломну роботу студенту

Теличко Ростислав Сергійович

1. Тема роботи «Програмно-методичний комплекс сегментації осцилографічних зображень»

керівник роботи Синиця Валентин Іванович, к.т.н., доцент

затверджені наказом по університету від «30» травня 2023 р. № 2057-с

2. Строк подання студентом роботи 09.06.2023 р.

3. Вихідні дані до роботи

Програмно-методичний комплекс призначений для дослідження стану водного середовища методами машинного навчання і повинен забезпечити: побудову моделі класифікації стану середовища відносно подій, що відбуваються, з попереднім застосуванням метода кластеризації; обробку вихідного датасета розміром 119 чорно-білих зображень фазових портретів; розділення датасета на сегменти (тест, тренінг); оцінку якості класифікації з ймовірністю не менш ніж 0,95;

перевірку адекватності роботи алгоритму кластеризації іншим алгоритмом; візуалізацію окремих етапів виконання алгоритмів і результатів роботи. Середовище проектування - Python v 3.8.

4. Зміст роботи ТЗ. Вступ. Призначення та галузь застосування. Огляд і аналіз існуючих технічних рішень. Розробка структури та програмних компонентів. Обробка результатів експериментальних досліджень.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) Блок-схеми алгоритмів (1л). Результати роботи програми (1л). Експериментальні дослідження (2л).

6. Дата видачі завдання 20.02.2023 р

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1	Розробка та узгодження технічного завдання	22.02.2023 р.	
2	Огляд і аналіз існуючих технічних рішень	05.03.2023 р.	
3	Розробка програмного забезпечення кластеризації	13.03.2023 р.	
4	Розробка програмного забезпечення класифікації	02.04.2023 р.	
5	Оцінювання результатів роботи програмних компонентів	14.04.2023 р.	
7	Оформлення графічних матеріалів	01.06.2023 р.	
8	Оформлення пояснювальної записки	01.06.2023 р.	

9	Попередній захист дипломного проекту	09.06.202, р.	
10	Рецензування дипломного проекту	до 14.06.2023 р.	
11	Захист дипломного проекту	до 20.06.2023 р.	

Студент

(підпис)

Ростислав ТЕЛИЧКО

Керівник роботи

(підпис)

Валентин СИНИЦЯ

АНОТАЦІЯ

Дипломна робота складається зі вступу, чотирьох розділів, висновків. Загальний об'єм пояснювальної записки становить 43 сторінки, 30 рисунків та 2 таблиці.

Об'єкт дослідження – системи дослідження морського середовища.

Мета роботи полягає в розробці програмно-методичного комплексу сегментації осцилографічних зображень, які отриманні в ході дослідження підводного середовища, за допомогою алгоритмів машинного навчання.

Актуальність роботи полягає в необхідності застосуванні сучасних технологій для автоматизації обробки осцилографічних зображень.

В ході виконання дипломного проекту розроблено програмне забезпечення в середовищі Python, яке реалізує алгоритми машинного навчання для обробки осцилограм.

Під час розробки вирішені наступні завдання:

1. Розробка програмного забезпечення в середовищі Python для сегментації осцилографічних зображень
2. Проведення експериментальних досліджень з наявними експериментальними даними у вигляді осцилографічних зображень.
3. Оцінка та аналіз отриманих результатів

ABSTRACT

The thesis consists of an introduction, four chapters, and conclusions. The total volume of the explanatory note is 43 pages, 30 figures and 2 tables.

The object of the study is marine environment research systems.

The purpose of the work is to develop a software-methodological complex for the segmentation of oscillographic images obtained during the study of the underwater environment, using machine learning algorithms.

The urgency of the work lies in the need to apply modern technologies for automating the processing of oscillographic images.

In the course of the diploma project, software was developed in the Python environment, which implements machine learning algorithms for processing oscillograms.

During development, the following tasks were solved:

1. Development of software in the Python environment for the segmentation of oscillographic images
2. Conducting experimental studies with available experimental data in the form of oscillographic images
3. Evaluation and analysis of the obtained results

Пояснювальна записка до дипломної роботи

на тему: «Програмно-методичний комплекс сегментації осцилографічних зображень»

Київ – 2023

ЗМІСТ

Вступ.....	3
1 Призначення та галузь застосування.....	5
2 Огляд і аналіз існуючих технічних рішень.....	6
2.1 Загальні відомості.....	6
2.2 Характеристики програмного забезпечення.....	10
2.3 Алгоритм KMeans.....	11
2.4 Алгоритм ієрархічної кластеризації.....	14
2.5 Алгоритм Спектральної кластеризації.....	19
2.6 Кероване машинне навчання.....	21
2.6.1 SVM алгоритм.....	22
2.6.2 KNN алгоритм.....	24
2.7 Вихідні дані.....	26
3 Розробка програмного компоненту.....	28
3.1 Розробка алгоритму роботи програми.....	28
3.2 Реалізація алгоритму.....	29
4 Експериментальні дослідження.....	32
4.1 Кластеризація.....	32
4.2 Класифікація.....	36
Висновок.....	40
Список використаних джерел.....	41
Додаток А.....	43

ВСТУП

Люди намагались автоматизувати рутинні процеси ще до появи комп'ютерів, що не дивно, бо це істотно скорочує витрату сил та ресурсів на реалізацію проектів незалежно від того виробнича це діяльність чи дослідницька. Приріст продуктивності, зумовлений впровадженням рішень, які замінюють людську працю, або суттєво підвищують її ефективність, призвів до стрімкого розвитку суспільства, поява мануфактур стала передвісником індустріальної революції. З науково-технічним прогресом змінювалась і уява про те, яким має бути майбутнє, все амбітнішою ставала наукова думка. Ще в далекому 1943 році логік Уолтер Піттс та нейробіолог Уоррен Мак Каллох опублікували статтю: «Логічне числення ідей, властивих нервовій діяльності», в якій намагались математично описати процеси мислення та прийняття рішень в людському пізнанні. Потім у 1949 році, Дональд Гебб у своїй книзі «Організація поведінки» описував те, як поведінка пов'язана з нейронними мережами та мозковою діяльністю. Згодом ці труди стали основою машинного навчання. Перша комп'ютерна програма яка навчалась, була розроблена Артуром Самуелем у 1957 році, це була гра в шашки, де комп'ютер фірми IBM удосконалювався, вивчаючи які ходи дозволяють вигравати ігри [1]. За останні десятиліття, завдяки появі достатньо потужних електронно-обчислювальних машин, алгоритми машинного навчання стають все більш інтегрованими в усі процеси людської діяльності. Зараз неймовірною популярністю користуються такі моделі штучного інтелекту як Dalle-2, яка дозволяє перетворювати текст у зображення, ChatGPT, який може налагоджувати комп'ютерні програми, складати музику, писати вірші, бути співрозмовником.

Основним інструментом розробки для машинного навчання є мова програмування Python. Він став основою для багатьох програм обробки даних. Це поєднує потужність мов програмування загального призначення з

простотою використання доменно-спеціальні мови сценаріїв, таких як MATLAB або R.

Python має бібліотеки для завантаження даних, візуалізації, статистики, обробки природної мови, обробки зображень та іншого. Цей величезний інструментарій надає дослідникам великий набір загальних і спеціальних функцій. Однією з головних переваг використання Python є можливість безпосередньої взаємодії з кодом, використовуючи термінал або інші інструменти, такі як Jupyter Notebook. Машинне навчання та аналіз даних є фундаментальними ітераційними процесами, в яких дані керують аналізом. Це необхідно для цих процесів мати інструменти, які забезпечують швидку ітерацію та легку взаємодію. Як мова програмування загального призначення, Python також дозволяє створювати складні графічні інтерфейси користувача (GUI) і веб-сервіси, а також для інтеграції в існуючі системи [2].

1 ПРИЗНАЧЕННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Призначенням цієї роботи є класифікація різних явищ в експериментальних даних та автоматизація процесу обробки осцилографічних зображень, за допомогою розробленого програмного забезпечення, яке реалізує обробку вихідних експериментальних даних алгоритмами машинного навчання, що дозволяє значно знизити час класифікації явищ.

Оскільки осцилографічні зображення є широко використовуваними їх обробка залишається актуальною задачею, можливі галузі застосування розробленого рішення:

1. Спостереження за морським середовищем
2. Медицина

Висновок : розроблене рішення спрощує обробку осцилографічних зображень та може використовуватись в різних галузях, для яких є характерною робота з осцилографічними зображеннями.

2 ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ

2.1 Загальні відомості

Машинне навчання — це підгалузь штучного інтелекту (ШІ), пов'язана з алгоритмами, які дозволяють комп'ютерам навчатися. У більшості випадків це означає, що алгоритм отримує набір даних і виводить інформацію про властивості даних, і ця інформація дозволяє робити прогнози щодо інших даних, які вона містить.

Щоб зрозуміти, як виникають моделі, розглянемо просту задачу фільтрації електронної пошти. Припустімо, ви отримуєте багато спаму, який містить слова «інтернет-аптека». Як людина, ви добре вмієте розпізнавати шаблони й швидко визначаєте, що будь-яке повідомлення зі словами «інтернет-аптека» є спамом і має бути переміщено безпосередньо до кошика. Це узагальнення – ви маєте, по суті, створити уявну модель того, що таке спам. Алгоритм машинного навчання, призначений для фільтрації спаму здатний зробити таке ж узагальнення.

Існує багато різних алгоритмів машинного навчання, кожен з яких має різні переваги і підходить для різних типів проблем. Деякі, такі як дерева рішень, є прозорими, тож спостерігач може повністю зрозуміти процес міркування, здійснюваний машиною. Інші, такі як нейронні мережі, є чорною скринькою, тобто вони дають відповідь, але часто дуже важко відтворити аргументацію, що стоїть за нею.

Багато алгоритмів машинного навчання значною мірою покладаються на математику та статистику, можна навіть сказати, що проста кореляція аналіз і регресія є основними формами машинного навчання [3].

Числові методи отримання класифікацій виникли в основному в природничих науках, таких як біологія та зоологія, намагаючись позбавити систематики її традиційно суб'єктивного характеру. Маючи на меті забезпечити об'єктивність та стабільність класифікації. Об'єктивною вважається така, в якій аналіз однієї й тієї ж сукупності організмів

здійснюваний однією і тією ж послідовністю чисельних методів дає однакову класифікацію; стабільною класифікація вважається, якщо залишається незмінною за різноманітного додавання організмів або нових характеристик, що їх описують.

Залежно від цих чисельних методів існує кілька назв в основному від області застосування. Числова систематика зазвичай використовується в біології. У психології іноді використовується термін Q-аналіз. У літературі яка стосується штучного інтелекту – неконтрольоване розпізнавання образів. Але в наш час кластеризація це, ймовірно, кращий загальний термін для процедур, спрямованих на виявлення груп в даних. У більшості застосувань кластерного аналізу ведеться пошук розділення даних, у якому кожен індивід або об'єкт належить до одного кластеру та повний набір кластерів містить усі об'єкти.

Методи кластеризації не слід плутати з методами розрізнення та присвоєння (у світі штучного інтелекту використовується термін кероване навчання), де групи відомі апріорі, і метою аналізу є побудова правил для класифікації нових індивідів у ту чи іншу з відомих груп.

Здається що такі терміни як: кластер, група та клас є повністю інтуїтивно зрозумілими. На ділі виходить так, що формальне визначення є не тільки складним, але навіть може бути недоречним. Бонер (1964), наприклад, вважав якщо використання такого терміну як «кластер» дає відповідь для дослідника, це все що потрібно.

Кормак (1971) і Гордон (1999), намагаються визначити просто що таке кластер з точки зору внутрішньої згуртованості – однорідності – і зовнішньої ізоляції – відокремлення. Такі властивості можна проілюструвати, принаймні неофіційно, за допомогою діаграм, як на рисунку 2.1.

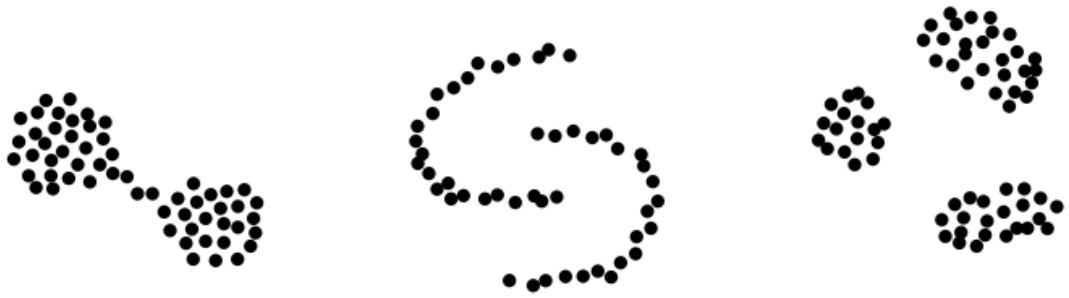


Рисунок 2.1 – Зображення різних кластерів

Кластери», присутні на цьому рисунку, будуть зрозумілі більшості спостерігачів, не намагаючись дати чітке формальне визначення цього терміну.

Дійсно, приклад показує, що жодного окремого визначення, ймовірно, не буде достатньо для всіх ситуації. Це може пояснити, чому спроби зробити концепції однорідності та поділу математично точними в термінах явних чисельних індексів призвели до численних і різноманітних критеріїв. Не зовсім зрозуміло, як розпізнається «кластер», коли він відображається в площині, але однією з особливостей процесу розпізнавання, є оцінка відносних відстаней між точками, які малюють люди-спостерігачі.

Ще один набір двовимірних даних зображено на рисунку 2.2.

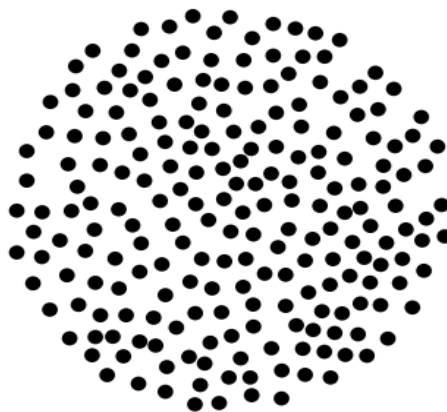


Рисунок 2.2 – Набір двовимірних даних

Тут більшість спостерігачів прийшли б до висновку, що не існує «природної» кластерної структури, просто єдиний однорідний набір даних. Тоді в ідеалі можна очікувати що метод кластерного аналізу, застосований до таких даних, дійде подібного висновку, але це може бути не так, і багато (більшість) методів кластерного аналізу розділить тип даних, показаний на рисунку 2.2, на «групи» (рисунок 2.3).

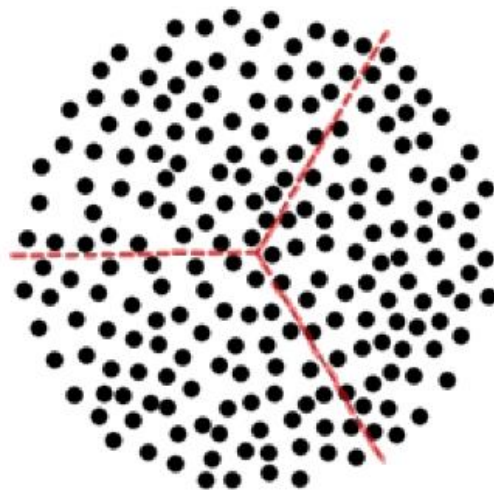


Рисунок 2.3 – Приклад поділу даних на неочевидні кластери

Дослідник апріорі не знає структури даних (кластерний аналіз, зрештою, покликаний допомогти розкрити будь-яку структуру), існує небезпека інтерпретації всіх рішень кластеризації з точки зору існування різних (природних) кластерів. Потім дослідник може «ігнорувати» можливість того, що класифікація, створена за допомогою кластерного аналізу, є артефактом методу, і насправді вона нав'язує структуру своїм даним, а не відкриває щось про реальну структуру, це дуже реальна проблема в застосуванні методів кластеризації [4].

2.2 Характеристики програмного забезпечення

Python — це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом із його інтерпретованим характером роблять його ідеальною мовою для написання сценаріїв і швидкої розробки додатків у багатьох областях на більшості платформ [5].

Візуалізація даних реалізована за допомогою використання бібліотек:

- matplotlib (бібліотека призначена для створення статичних, або анімованих та інтерактивних візуалізацій на Python)
- seaborn (бібліотека для створення статистичної графіки. Він будується на основі matplotlib і тісно інтегрується зі структурами даних pandas).

Розробка алгоритмів

NumPy — це основний пакет для наукових обчислень на Python. Це бібліотека Python, яка надає:

- об'єкт багатовимірного масиву
- різні похідні об'єкти (такі як замасковані масиви та матриці)
- набір процедур для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції формою, сортування, вибір, введення/виведення, дискретне перетворення Фур'є, базова лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого [6].

Pandas – це бібліотека з відкритим вихідним кодом, ліцензована BSD, що надає високопродуктивні, прості у використанні структури даних та інструменти аналізу даних для мови програмування Python [7].

Scikit-learn – бібліотека розроблена для реалізації алгоритмів машинного навчання.

OpenCV – Бібліотека комп'ютерного зору з відкритим кодом, написана

мовою C, яка містить кілька сотень алгоритмів для обробки зображень, широко використовується в таких мовах програмування як Python, C++, Java. Оскільки комп'ютерний зір і машинне навчання часто йдуть рука об руку, OpenCV також містить повну бібліотеку машинного навчання загального призначення (MLL), але вона достатньо загальна, щоб використовувати її для будь-якої проблеми машинного навчання [8].

Jupyter Notebook — це інтерактивна веб-програма з відкритим вихідним кодом, яка дозволяє писати та запускати комп'ютерний код понад 40 мовами програмування, включаючи Python, R, Julia та Scala. Jupyter Notebook, продукт від Project Jupyter, корисний для ітеративного кодування, оскільки дозволяє написати невеликий фрагмент коду, запустити його та повернути результат.

Jupyter Notebook надає можливість створювати документи блокнотів, які називають просто «записними книжками». Блокноти, створені на основі Jupyter Notebook, є дослідницькими документами, які можна відтворювати, ними можна ділитися, вони містять елементи тексту, рівняння, код та їхні результати (рисунки, таблиці, інтерактивні графіки). Блокноти також можна експортувати у файли необробленого коду, документи HTML або PDF або використовувати для створення інтерактивних слайд-шоу чи веб-сторінок [9].

2.3 Алгоритм KMeans

KMeans — це ітеративна процедура, яка розбиває N об'єктів на K непересічних кластерів. KMeans є, мабуть, найбільш широко використовуваним методом кластеризації, і особливо найвідомішим із методів кластеризації на основі поділу, який використовує центроїди (евклідова відстань між точками даних і центром кластера, де центроїд є центром геометричного об'єкта, і він розглядається як узагальнення середнього значення) для представлення кластерів. Якість кластеризації KMeans вимірюється за допомогою критерію квадратичної помилки всередині кластера [10].

Алгоритм KMeans працює наступним чином:

- Вказується кількість кластерів K .
- Обираються центроїди, спочатку перетасувавши набір даних, а потім випадково вибравши K точок даних для центроїдів.
- Обчислюється сума квадратів відстані між точками даних і всіма центроїдами.
- Призначається кожна точка даних найближчому центроїду.
- Обчислюються нові центроїди для кластерів, взявши середнє значення всіх точок даних, які належать кожному кластеру.
- Повторюється до тих пір поки центроїди не перестануть змінюватись.

Підхід KMeans для вирішення проблеми називається очікування-максимізація. Очікування призначає точки даних найближчому кластеру. Максимізація обчислює центроїд кожного кластера. Нижче наведено розбивку того, як можна вирішити це математично.

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \|x^i - \mu_k\|^2 \quad (1)$$

Де w_{ik} дорівнює одиниці для точки даних x^i , якщо вона належить кластеру K ; інакше w_{ik} дорівнює нулю. μ_k є центроїдом кластера K .

Це проблема мінімізації з двох частин. Спочатку ми мінімізуємо J по w_{ik} і відношення μ_k виправлено. Потім ми мінімізуємо J по μ_k і відношення w_{ik} виправлено. Технічно кажучи, ми розрізняємо похідну J по w_{ik} і оновлюємо призначення кластерів (очікування). Тоді ми розрізняємо похідну J по μ_k і повторно обчислюємо центроїди після кластерних призначень з попереднього

кроку (максимізація). Отже, очікування це:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^m \sum_{k=1}^K \|x^i - \mu_k\|^2 \Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_k \|x^i - \mu_k\|^2 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Іншими словами, призначення точки даних x^i найближчому кластеру за сумою квадратів відстані від центроїда кластера.

А максимізація це:

$$\frac{\partial J}{\partial w_{ik}} = 2 \sum_{i=1}^m w_{ik} (x^i - \mu_k) = 0 \Rightarrow \mu_k = \frac{\sum_{i=1}^m w_{ik} x^i}{\sum_{i=1}^m w_{ik}} \quad (3)$$

Тут варто зауважити кілька речей:

- Оскільки алгоритми кластеризації, включаючи KMeans, використовують вимірювання на основі відстані для визначення подібності між точками даних, рекомендується стандартизувати дані, щоб мати середнє значення нуль і стандартне відхилення одиниці, оскільки майже завжди об'єкти в будь-якому наборі даних матимуть різні одиниці вимірювання наприклад вік проти доходу.
- Враховуючи ітераційну природу KMeans і випадкову ініціалізацію центроїдів на початку алгоритму, різні ініціалізації можуть призвести до різних кластерів, оскільки алгоритм KMeans може застрягти в локальному оптимумі та не збігатися з глобальним оптимумом. Тому рекомендується запускати алгоритм, використовуючи різні ініціалізації центроїдів і вибирати результати виконання, які давали б нижчу суму квадратів відстані.

- Призначення прикладів не змінюється – це те саме, що відсутність змін у варіації всередині кластера [11].

2.4 Алгоритм ієрархічної кластеризації

Ієрархічна кластеризація є однією з популярних і простих для розуміння методів кластеризації. Ця техніка кластеризації поділяється на два види:

1. Агломераційний
2. Розділовий

Техніка агломераційної ієрархічної кластеризації: у цій техніці спочатку кожна точка даних розглядається як окремий кластер. На кожній ітерації подібні кластери зливаються з іншими кластерами, поки не буде сформований один або K кластерів.

Базовий агломераційний алгоритм є простим:

- Обчислюється матриця близькості. Нехай кожна точка даних буде кластером
- Об'єднуються два найближчі кластери та оновлюється матриця близькості
- Повторюється поки не залишиться єдиний кластер

Щоб краще зрозуміти, давайте подивимося графічне представлення техніки агломераційної ієрархічної кластеризації. Припустимо, у нас є шість точок даних $\{A, B, C, D, E, F\}$.

- Крок 1: на початковому кроці ми обчислюємо близькість окремих точок і розглядаємо всі шість точок даних як окремі кластери, як показано на рисунку 2.4.

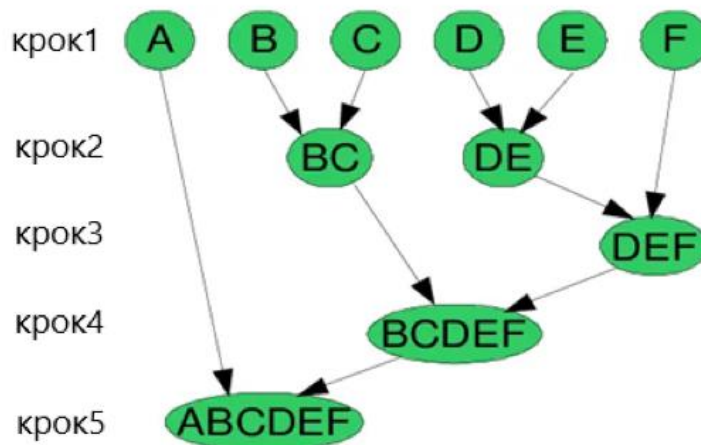


Рисунок 2.4 – Принцип роботи агломераційного алгоритму кластеризації

- Крок 2: На другому кроці подібні кластери об'єднуються разом і утворюються як єдиний кластер. Давайте розглянемо B, C і D, E — подібні кластери, які об'єднано на другому кроці. Тепер у нас залишилося чотири кластери: A, BC, DE, F.
- Крок 3: ми знову обчислюємо близькість нових кластерів і об'єднуємо подібні кластери, щоб утворити нові кластери A, BC, DEF.
- Крок 4: Обчислюється близькість нових кластерів. Кластери DEF і BC подібні та об'єднані разом, щоб утворити новий кластер. Тепер залишилося два кластери A, BCDEF.
- Крок 5: Нарешті, усі кластери об'єднуються разом і утворюють єдиний кластер [12].

Для візуалізації результатів агломераційної кластеризації прийнято використовувати дендрограму.

Дендрограма ілюструє, як складається кожен кластер, малюючи U-подібний зв'язок між неодиначним кластером і його дочірніми елементами. Верхня частина U-ланки вказує на злиття кластерів. Дві частини U-ланки вказують, які кластери були об'єднані. Довжина двох кінців U-ланки представляє відстань між дочірніми кластерами. [13].

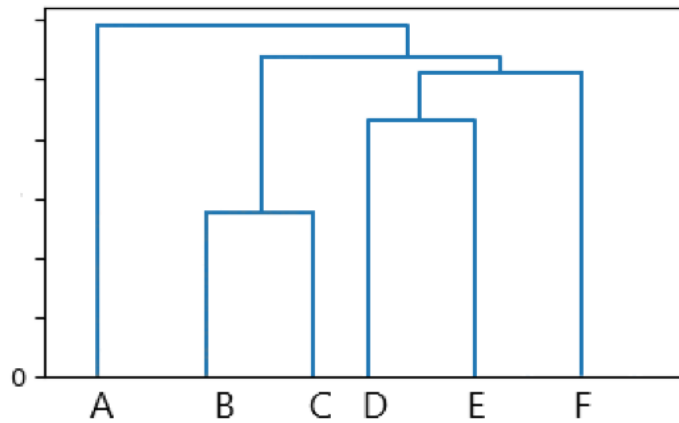


Рисунок 2.5 – Дендрограма яка відповідає рисунку 2.4

Техніка роздільної ієрархічної кластеризації: Оскільки техніка роздільної ієрархічної кластеризації мало використовується в реальному світі, я коротко розповім про техніку роздільної ієрархічної кластеризації.

Простими словами, ми можемо сказати, що розділова ієрархічна кластеризація є прямою протилежністю агломераційній ієрархічній кластеризації. У роздільній ієрархічній кластеризації ми розглядаємо всі точки даних як один кластер і на кожній ітерації ми відокремлюємо точки даних від кластера, які не схожі. Кожна відокремлена точка даних розглядається як окремий кластер. Зрештою, у нас залишиться n кластерів.

Оскільки ми ділимо окремі кластери на n кластерів, це називається роздільною ієрархічною кластеризацією.

Розрахунок подібності між двома кластерами важливий для об'єднання або розділення кластерів. Існують певні підходи, які використовуються для обчислення подібності між двома кластерами:

- MIN – простими словами, вибирається дві найближчі точки так, що одна точка лежить у кластері один, а інша точка лежить у кластері два, і береться відстань між ними та оголошується це як подібність між двома кластерами, рисунок 1.6.

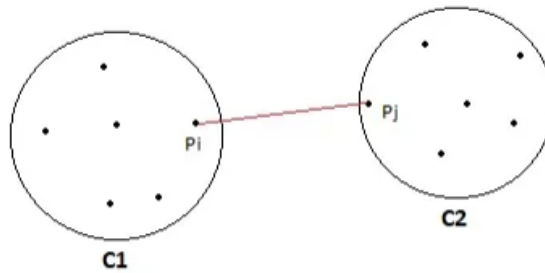


Рисунок 2.6 – Метод MIN

Переваги MIN:

Цей підхід може розділяти нееліптичні форми, якщо проміжок між двома кластерами не малий.

Недоліки MIN:

Підхід MIN не може правильно розділити кластери, якщо між кластерами є шум.

- **MAX** Простими словами, вибираються дві найдальші точки так, щоб одна точка лежала в кластері один, а інша точка лежала в кластері два, і береться відстань між ними і оголошується це як подібність між двома кластерами

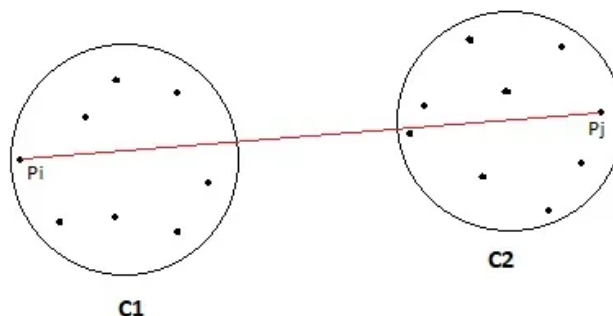


Рисунок 2.7 – Метод MAX

Переваги MAX:

Підхід MAX добре справляється з розділенням кластерів, якщо між

кластерами є шум.

Недоліки MAX :

Максимальний підхід спрямований на кульові скупчення.

Максимальний підхід має тенденцію розбивати великі кластери.

- Середня група – середня відстань між усіма парами точок.

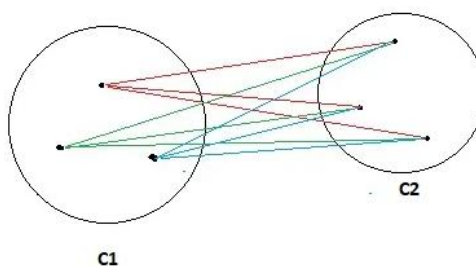


Рисунок 2.8 – Метод середня група

Переваги середньої групи:

Підхід групового усереднення добре справляється з розділенням кластерів, якщо між кластерами є шум.

Недоліки середньої групи:

Середньо груповий підхід спрямований на кульові скупчення.

- Відстань між центроїдами Обчислюються центроїди двох кластерів C1 і C2 і приймається відстань між центроїдами як подібність між двома кластерами [12].

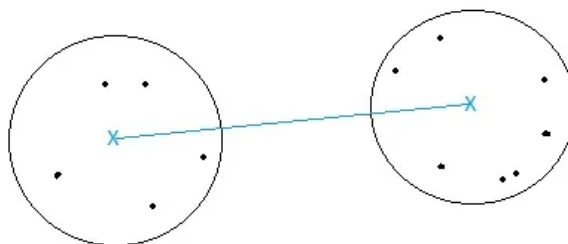


Рисунок 2.9 – Метод відстань між центроїдами

2.5 Алгоритм Спектральної кластеризації

У спектральній кластеризації точки даних розглядаються як вузли графа. Таким чином, спектральна кластеризація є проблемою розбиття графа. Потім вузли відображаються в низько-вимірному просторі, який можна легко відокремити, щоб сформувати кластери. Ніяких припущень щодо форми кластерів не робиться. Метою спектральної кластеризації є кластеризація даних, які пов'язані (точки, які з'єднані або розташовані безпосередньо одна від одної, поміщаються в один кластер. Навіть якщо відстань між 2 точками менша, якщо вони не з'єднані, вони не групуються разом. Спектральна кластеризація — це техніка, яка дотримується цього підходу.), але не обов'язково компактні або згруповані в опуклих межах.

Компактність — точки, які розташовані близько одна до одної, потрапляють в один кластер і є компактними навколо центру кластера. Близькість можна виміряти відстанню між спостереженнями.

На рисунку 2.10 зображено приклад розташування даних

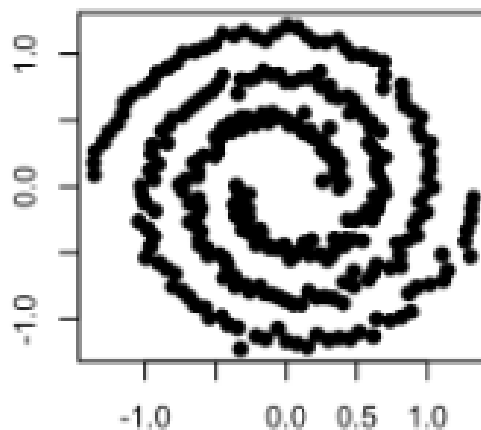


Рисунок 2.10 – Приклад розташування точок

Тут ми маємо точки даних, щільно представлені у формі спіралі, порівняємо кластеризацію алгоритмом KMeans, та спектральним алгоритмом, рисунок 2.11.

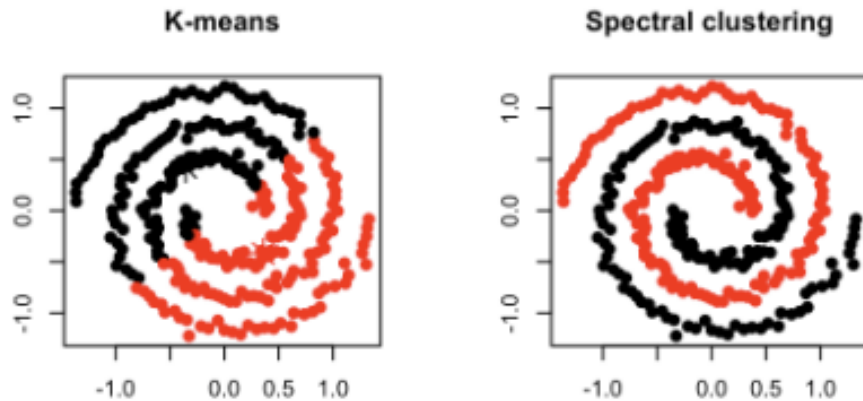


Рисунок 2.11 – Порівняння кластеризації алгоритмами KMeans та спектральним

KMeans не зміг ефективно здійснити кластеризацію даних, навіть якщо справжня кількість кластерів K відома алгоритму. KMeans, як алгоритм кластеризації даних, ідеальний для виявлення кульових кластерів, де всі члени кожного кластера знаходяться в безпосередній близькості один від одного (в евклідовому розумінні).

Спектральна кластеризація є більш загальною (і потужною), оскільки якщо ми просто використовуємо Евклідову відстань у її матриці подібності, вона поводитиметься як KMeans. Однак зворотне не вірно [14].

Алгоритм спектральної кластеризації працює наступним чином:

Будується графік подібності даних: на цьому етапі будується графік подібності у формі матриці суміжності. Матрицю суміжності можна побудувати таким чином:

- Графік епсилон-околиці: параметр епсилон фіксується заздалегідь. Потім кожна точка з'єднується з усіма точками, які лежать в її епсилон-радіусі. Якщо всі відстані між будь-якими двома точками однакові за масштабом, зазвичай ваги ребр, тобто відстань між двома точками, не зберігаються, оскільки вони не надають жодної додаткової інформації.

- K-найближчі сусіди: параметр k фіксується заздалегідь. Тоді для

двох вершин u і v ребро спрямоване від u до v , лише якщо v є серед k -найближчих сусідів u . Не завжди для кожного u , який має v як одного із k -найближчих сусідів, це буде той самий випадок для v , що має u серед своїх k -найближчих сусідів.

- Повністю зв'язаний граф: щоб побудувати цей граф, кожна точка з'єднується неорієнтованим ребром, зваженим за відстанню між двома точками та кожною іншою точкою. Оскільки цей підхід використовується для моделювання зв'язків місцевого сусідства, для обчислення відстані зазвичай використовується метрика подібності Гауса.

Проектування даних на нижчий вимірний простір: цей крок робиться для врахування можливості того, що члени одного кластера можуть бути далеко в даному вимірному просторі. Таким чином розмірний простір зменшується так, що ці точки знаходяться ближче в зменшеному розмірному просторі і, таким чином, можуть бути згруповані разом за допомогою традиційного алгоритму кластеризації [15].

2.6 Кероване машинне навчання

Кероване машинне навчання, є підкатегорією машинного навчання та штучного інтелекту. Він визначається використанням позначених наборів даних для навчання алгоритмів, які класифікують дані або точно прогнозують результати. Коли вхідні дані надходять у модель, вона коригує свої ваги, доки модель не буде підігнано належним чином, що відбувається як частина процесу перехресної перевірки. Навчання під наглядом допомагає організаціям вирішувати різноманітні реальні проблеми в масштабі, наприклад класифікувати спам в окремій папці з папки «Вхідні».

Кероване навчання використовує навчальний набір, щоб навчити моделі отримувати бажаний результат. Цей навчальний набір даних включає вхідні та правильні вихідні дані, які дозволяють моделі навчатися з часом.

Алгоритм вимірює свою точність за допомогою функції втрат,

коригуючи, доки помилка не буде достатньо мінімізована.

Кероване навчання можна розділити на два типи проблем під час інтелектуального аналізу даних — класифікація та регресія:

Класифікація використовує алгоритм для точного віднесення тестових даних до певних категорій. Він розпізнає конкретні сутності в наборі даних і намагається зробити деякі висновки про те, як ці сутності слід позначати або визначати.

Поширеними алгоритмами класифікації є лінійні класифікатори, опорні векторні машини (SVM), дерева рішень, KNN і випадковий ліс.

Регресія використовується для розуміння зв'язку між залежними та незалежними змінними. Він зазвичай використовується для прогнозування, наприклад, доходів від продажів для даного бізнесу. Лінійна регресія, логістична регресія та поліноміальна регресія є популярними алгоритмами регресії [16].

2.6.1 SVM алгоритм

Алгоритм SVM використовувався в багатьох задачах розпізнавання образів і регресійної оцінки, а також був застосований до проблем оцінки залежностей, прогнозування та побудови інтелектуальних машин. SVM — це максимальна периферійна гіперплощина, яка лежить у деякому просторі та класифікує дані, розділені нелінійними межами, які можна побудувати шляхом визначення місцезнаходження набору гіперплощин, які розділяють два або більше класів точок даних. Після побудови гіперплощин SVM виявляє межі між вхідними класами та вхідними елементами, що визначають межі (опорні вектори) [17].

Давайте розглянемо дві незалежні змінні x_1 , x_2 і одну залежну змінну, яка є синім або червоним колом, рисунок 2.13.

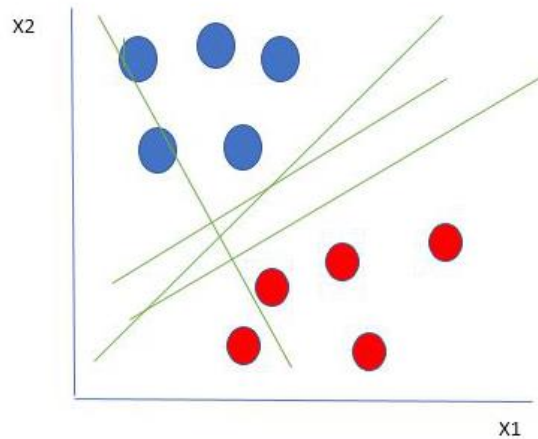


Рисунок 2.13 – Приклад побудови гіперплощини

З рисунка вище дуже чітко видно, що існує кілька ліній (наша гіперплощина тут є лінією, оскільки ми розглядаємо лише дві вхідні характеристики x_1 , x_2), які розділяють наші точки даних або роблять класифікацію між червоними та синіми колами. Отже, як вибрати найкращу пряму або загалом найкращу гіперплощину, яка розділяє наші точки даних?

Найкраща гіперплощина – це та, яка представляє найбільше розділення або маржу між двома класами, рисунок 2.14.

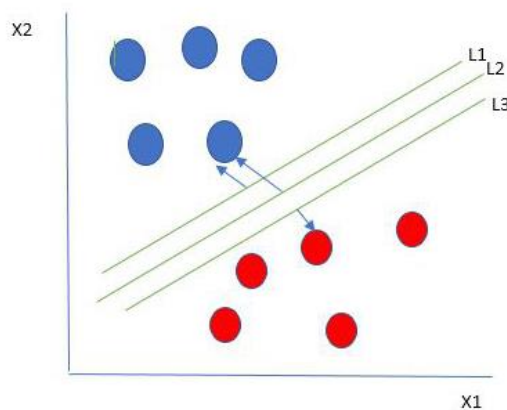


Рисунок 2.14 – Пошук найкращої гіперплощини

Ми вибираємо гіперплощину, відстань від якої до найближчої точки даних з кожного боку є максимальною. Отже, з наведеного вище рисунка ми вибираємо L_2 [18].

2.6.2 KNN алгоритм

Алгоритм KNN або k-NN, — це класифікатор із керованим навчанням, який використовує близькість для класифікації або прогнозування щодо групування окремої точки даних. Хоча його можна використовувати як для регресії, так і для задач класифікації, зазвичай він використовується як алгоритм класифікації, виходячи з припущення, що подібні точки можна знайти поруч.

Для проблем класифікації мітка класу призначається на основі більшості голосів, тобто використовується мітка, яка найчастіше представлена навколо певної точки даних. Хоча технічно це вважається «plurality voting», у літературі частіше використовується термін «majority vote». Різниця між цими термінами полягає в тому, що «plurality voting» технічно вимагає більшості понад 50%, що в основному працює, коли є лише дві категорії. Якщо є кілька класів, наприклад чотири, не обов'язково потрібно 50% голосів, щоб зробити висновок про клас; можна призначити мітку класу з голосуванням більше ніж 25%.

Щоб визначити, які точки даних є найближчими до заданої точки запиту, необхідно обчислити відстань між точкою запиту та іншими точками даних. Ці показники відстані допомагають сформулювати межі рішень, які розділяють запиту на різні регіони.

Популярні способи обчислення відстані:

- Евклідова відстань: це найбільш часто використовувана міра відстані, яка обмежена дійсними векторами. Використовуючи наведену нижче формулу, він вимірює пряму лінію між точкою запиту та іншою точкою, що вимірюється.
- Манхеттенська відстань: це ще один популярний показник відстані, який вимірює абсолютне значення між двома точками. Її також

називають відстанню таксі або відстанню міського кварталу, оскільки її зазвичай візуалізують за допомогою сітки, яка ілюструє, як можна переміщатися з однієї адреси на іншу вулицями міста.

Значення K в алгоритмі KNN визначає, скільки сусідів буде перевірено для визначення класифікації конкретної точки запиту. Наприклад, якщо $K=1$, екземпляр буде віднесено до того ж класу, що й його єдиний найближчий сусід. Визначення k може бути врівноваженим актом, оскільки різні значення можуть призвести до надмірного або недостатнього встановлення. Нижчі значення K можуть мати високу дисперсію, але низьке зміщення, а більші значення K можуть призвести до високого зміщення та меншої дисперсії. Вибір K значною мірою буде залежати від вхідних даних, оскільки дані з більшою кількістю викидів або шуму, швидше за все, працюватимуть краще з вищими значеннями K . Загалом, рекомендується мати непарне число для K , щоб уникнути зв'язків у класифікації.

Переваги:

- Легко реалізувати: враховуючи простоту й точність алгоритму, це один із перших класифікаторів, який вивчає новачок у дослідженні даних.
- Легко адаптується: коли додаються нові навчальні зразки, алгоритм налаштовується з урахуванням будь-яких нових даних, оскільки всі навчальні дані зберігаються в пам'яті.
- Мало гіперпараметрів: KNN вимагає лише значення K і метрики відстані, що небагато порівняно з іншими алгоритмами машинного навчання.

Недоліки:

- Погано масштабується: оскільки KNN є ледачим алгоритмом, він займає більше пам'яті та зберігання даних порівняно з іншими класифікаторами. Це може бути дорогим як з точки зору часу, так і з точки зору грошей. Більше пам'яті та сховища призведе до збільшення витрат бізнесу, а обчислення більшої кількості даних може зайняти більше часу.

- Прокляття розмірності: алгоритм KNN, як правило, стає жертвою прокляття розмірності, що означає, що він погано працює з вхідними даними великої розмірності. Це іноді також називають феноменом піку, коли після того, як алгоритм досягає оптимальної кількості функцій, додаткові функції збільшують кількість помилок класифікації, особливо коли розмір вибірки менший.
- Схильність до перенавчання: через «прокляття розмірності» KNN також більш схильний до перенавчання. Хоча методи вибору функцій і зменшення розмірності використовуються для запобігання цьому, значення K також може впливати на поведінку моделі. Нижчі значення K можуть бути причиною перенавчання, тоді як вищі значення k мають тенденцію до «згладжування» прогнозованих значень, оскільки це усереднення значень на більшій території або околиці. Однак, якщо значення K занадто велике, воно може не відповідати даним [19].

2.7 Вихідні дані

Вхідними даними є осцилограми отримані з приладів в ході дослідження морського середовища, збережені як зображення в форматі PNG. Кількість зображень :119. Приклад вхідних даних наведено на рисунку 2.15.

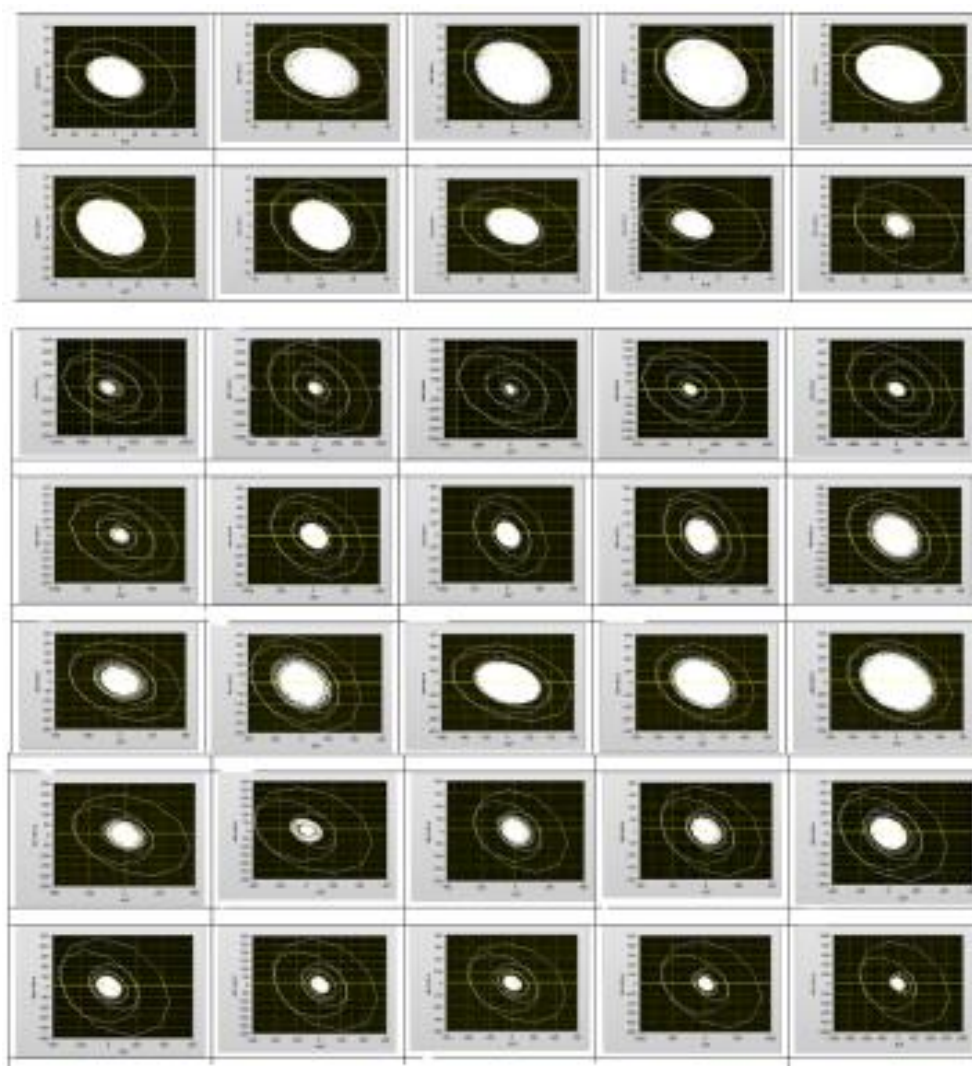


Рисунок 2.15 – Приклад вихідних даних

Висновок: розглянуто теоретичні відомості, та обрано спосіб реалізації завдання.

3. РОЗРОБКА ПРОГРАМНОГО КОМПОНЕНТУ

3.1 Розробка алгоритму роботи програми

Алгоритм роботи зображено на рисунку 3.1.

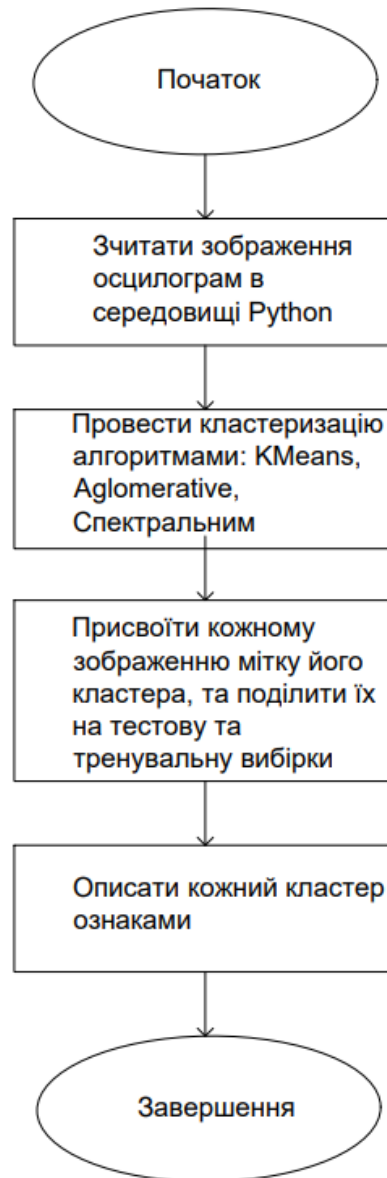


Рисунок 3.1 – Алгоритм роботи програми

3.2 Реалізація алгоритму

Відкриємо зображення, за допомогою функції бібліотеки CV2 : `cv2.imread(path)`, та конвертуємо його в чорно-біле за допомогою функції `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)` результат зображено на рисунку 3.2.

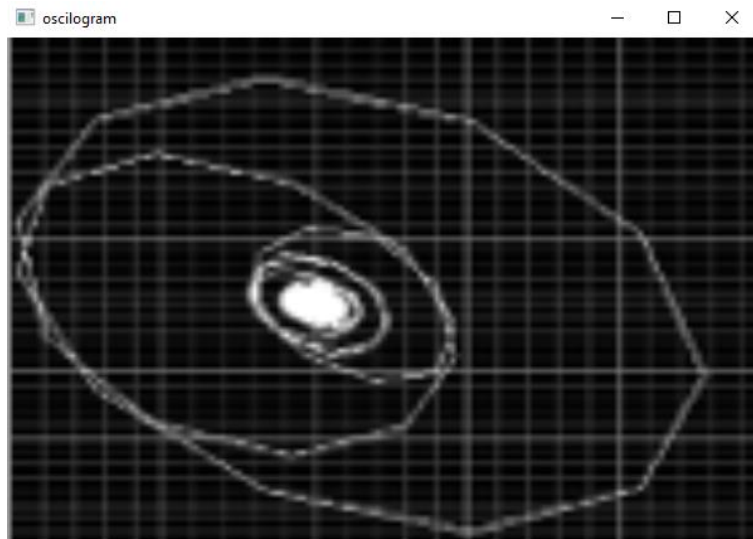


Рисунок 3.2 – Відкрите зображення осцилограми в середовищі Python

Конвертація в чорно-біле проведена з метою спрощення подальшої обробки, оскільки зображення можна представити як масив масивів які відповідають за колір пікселів рисунок 3.3

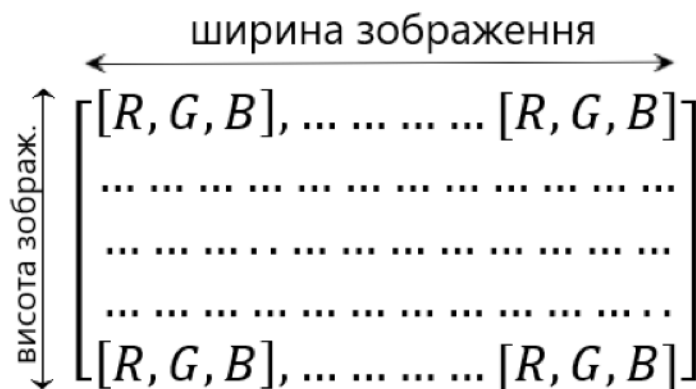


Рисунок 3.3 – Представлення зображення у вигляді масиву

Для чорно-білого зображення колір буде представлений, замість масиву R, G, B (red, green, blue) лише градацією чорного.

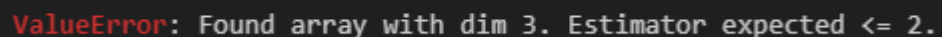
Оскільки ми будемо обробляти усі осцилограми разом, потрібно їх об'єднати в один масив, для чого створимо змінну типу лист яка буде в собі містити n (кількість зображень) листів: `y = [[] for i in range(n)]`. Впишемо всі зображення в змінну `y`.

Для чого використаємо цикл `for`, в якому значення `i` буде змінюватись від 0 до n, та послідовно впишемо в кожний лист масив (зображення):

```
for i in range(n):  
    y[i] = imgi
```

Для подальшої роботи переведемо змінну `y` в тип `np.ndarray`: `np.asarray(y)`, отже ми отримали тримірний масив розміром (n, висота зображення, ширина зображення). Змінимо порядок зображень в масиві за допомогою функції `np.shuffle()`. Перетасування даних має на меті зменшити дисперсію та переконатися, що моделі залишаються загальними.

Оскільки функції, які реалізують алгоритми кластеризації, очікують на вході отримати двовимірний масив, на рисунку 3.4 зображено помилку яка отримана при спробі подати на вхід наш масив зображень.



```
ValueError: Found array with dim 3. Estimator expected <= 2.
```

Рисунок 3.4 – Похибка при спробі подати на вхід функції тривимірний масив

Змінимо форму масиву з зображеннями за допомогою функції `np.reshape(arg,(n,-1))`, що замість представлення зображення матрицею висота x ширина, запише всі пікселі в одну строку, результат – рисунок 3.5.

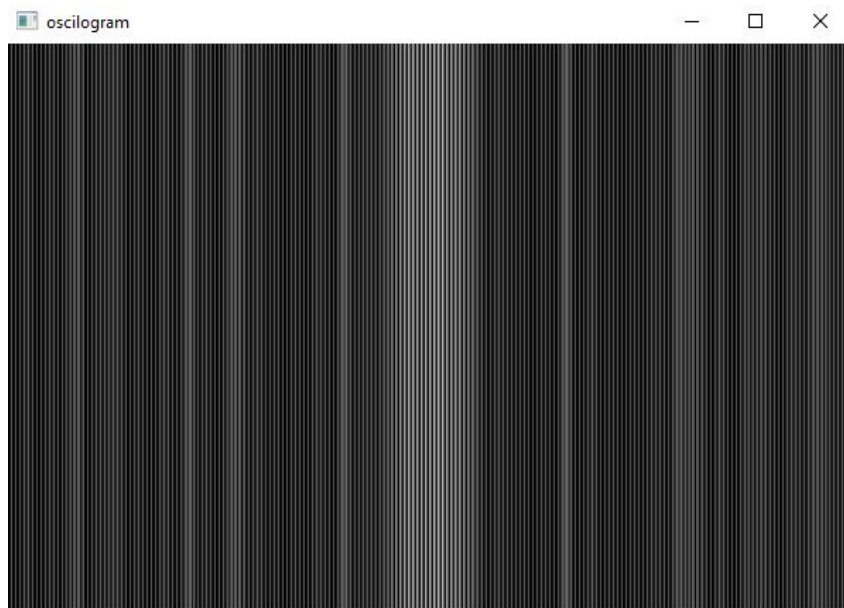


Рисунок 3.5 – Зображення після зміни форми

Підготовка даних до обробки алгоритмами машинного навчання завершена, тому перейдемо до самого експерименту. Код програми наведено в додатку А.

Висновок: проведено підготовку вихідних даних до обробки алгоритмами машинного навчання, та розроблено програмне забезпечення результати

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Кластеризація

Спочатку знайдемо оптимальну кількість кластерів для чого скористаємось методом ліктя [20], це один із найпоширеніших способів вибору значення кількості кластерів, який передбачає створення графіка з кількістю кластерів на осі абсцис і загальною сумою квадратів помилок моделі на осі у, а потім визначення, де з'являється «лікоть» або вигин.

Точка на осі абсцис, де виникає «лікоть», говорить нам про оптимальну кількість кластерів для використання в алгоритмі кластеризації KMeans [21] (рисунок 4.1).

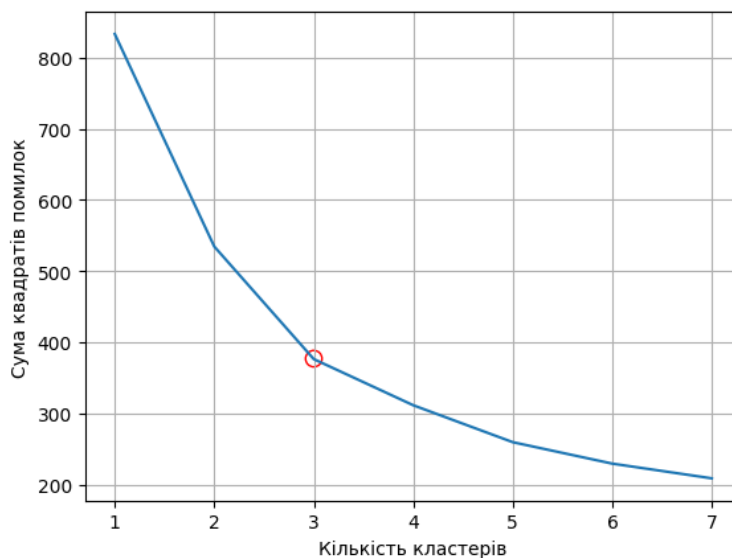


Рисунок 4.1 – Пошук оптимальної кількості кластерів методом ліктя

На рисунку 4.1 червоним колом виділено шукану точку отже, оптимальна кількість кластерів дорівнює трьом.

Перевіримо це побудувавши дендрограму агломераційної кластеризації рисунок 4.2.

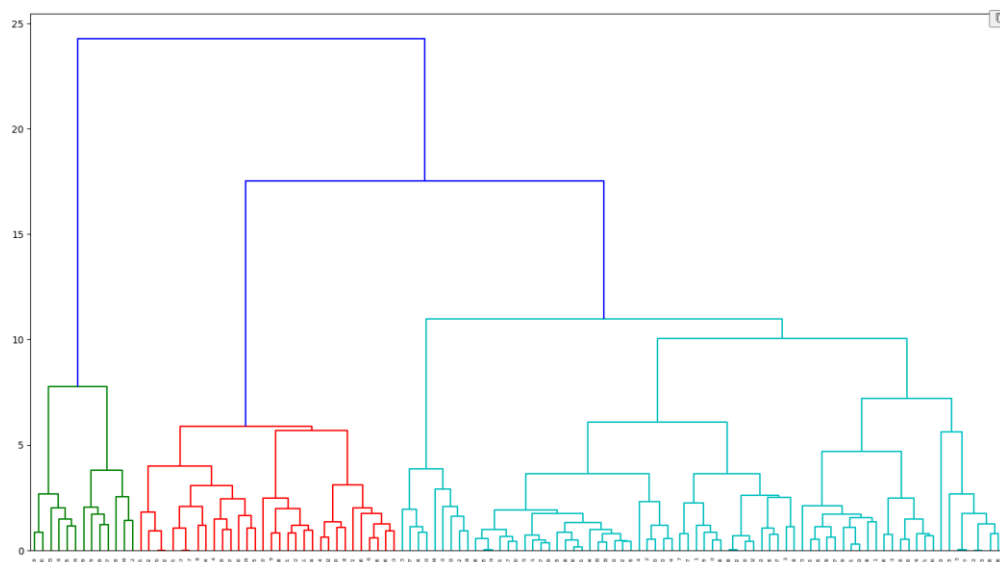


Рисунок 4.2 – Дендрограма на якій зображено результат кластеризації

Отже з рисунку 4.2 оптимальна кількість кластерів також три (вони виділені кольорами зелений, червоний, світло-голубий). Отже оптимальна кількість кластерів знайдена, тому подивимось на результати кластеризації.

Оскільки кількість параметрів для побудови точки даних дорівнює кількості пікселів, щоб зобразити це на діаграмі необхідно зменшити розмірність даних, для чого використано функцію бібліотеки `scikit-learn PCA`.

`PCA` обчислює вектори, на які можна проектувати свої дані, щоб зменшити їх розмірність. Кожен рядок `PCA.components_` — це єдиний вектор, на який проектуються дані, і він матиме той самий розмір, що й кількість стовпців у даних. Перший із цих векторів максимізує дисперсію прогнозованих даних. 2-й максимізує дисперсію того, що залишилося після першої проекції [22].

Отже ми отримаємо матрицю розмірністю 119x2. Побудуємо діаграму розподілу точок на кластери для алгоритму `KMeans`, рисунок 4.3.

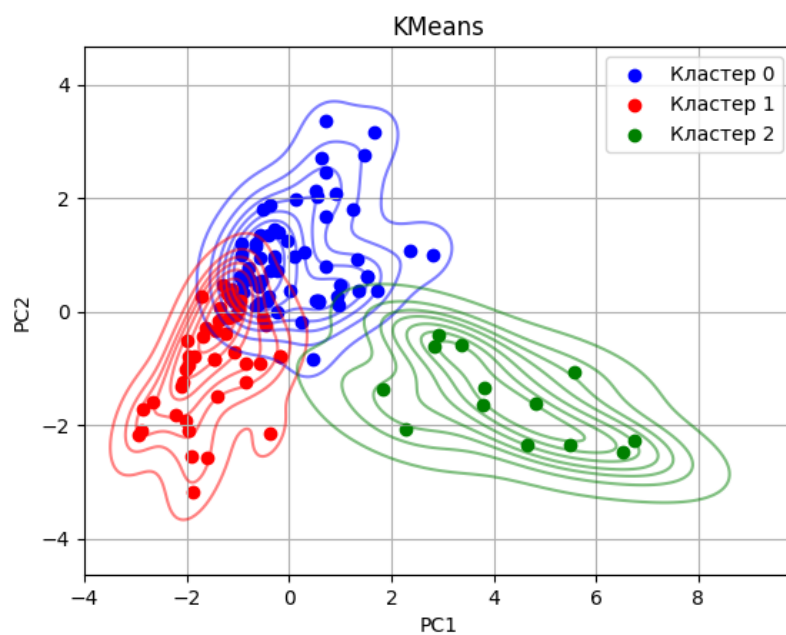


Рисунок 4.3 – Результат кластеризації алгоритмом KMeans

Проведемо кластеризацію агломераційним алгоритмом рисунок 4.4

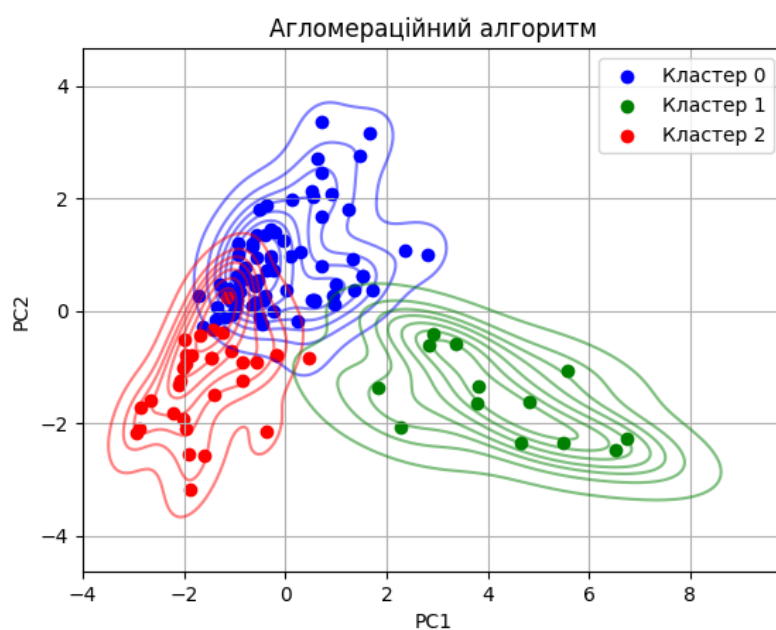


Рисунок 4.4 – Результат кластеризації агломераційним алгоритмом

Проведемо кластеризацію спектральним алгоритмом рисунок 4.5

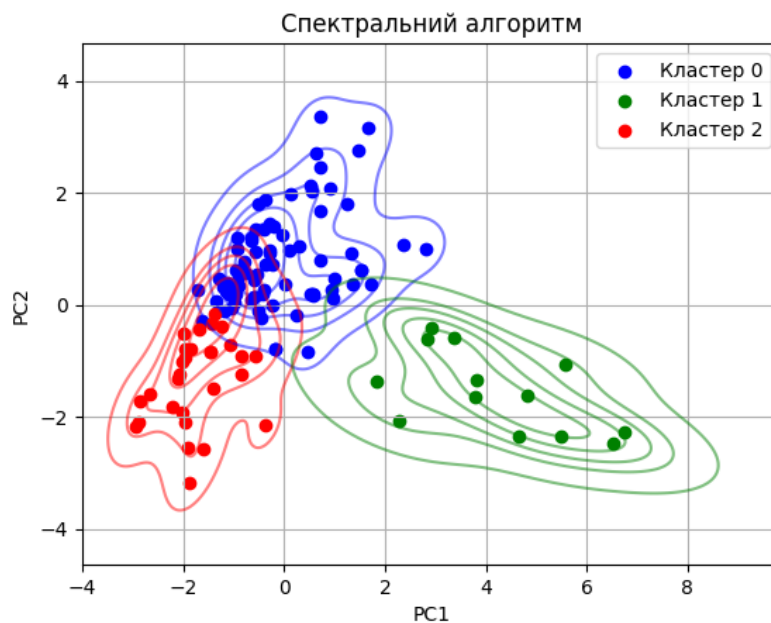


Рисунок 4.5 – Результат кластеризації спектральним алгоритмом

Результати кластеризації різними алгоритмами схожі, подивимось на скільки рисунок 4.6.

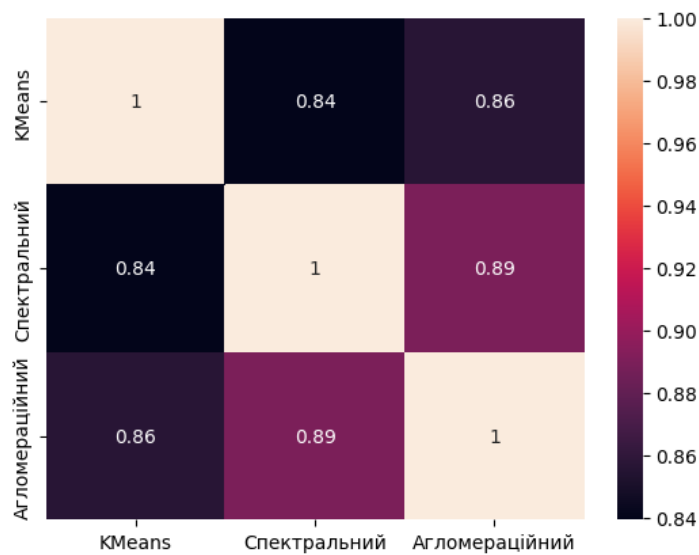


Рисунок 4.6 – Матриця кореляції результатів кластеризації різними алгоритмами

Типові зображення для кожного кластера приведені на рисунку 4.7

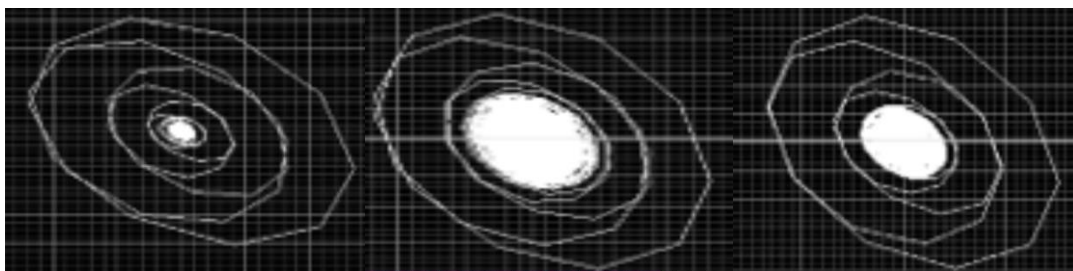


Рисунок 4.7 – Типові зображення для кожного кластеру

З рисунку 4.7: кластер 1 – мінімальна активність в досліджуваному середовищі, кластер 2 – надзвичайна активність, кластер 3 – помірна активність.

4.2 Класифікація

Поділимо зображення з мітками кластерів на тренувальну та тестові вибірки у відношенні 75% - тестова, 25% - тренувальна. Проведемо класифікацію алгоритмом KNN, результат передбачень для тренувального сету рисунок 4.8, де по діагоналі зазначено кількість вірно класифікованих зображень.

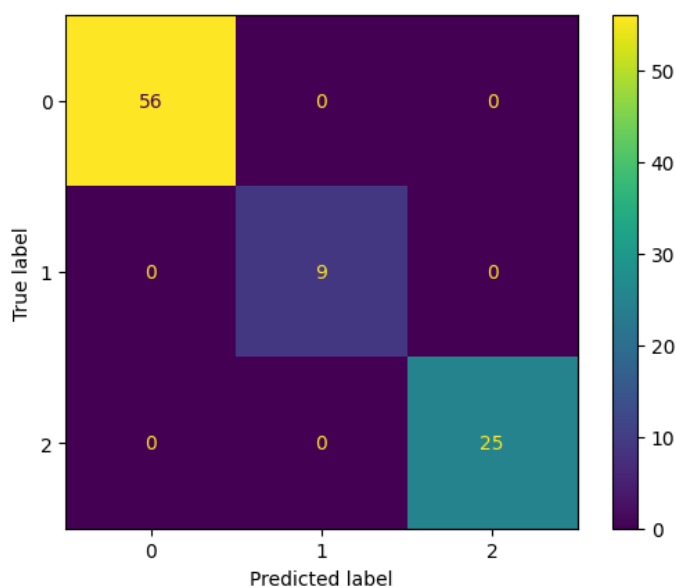


Рисунок 4.8 – Матриця похибок для тренувального сету

Результат роботи на тестовому сеті приведено на рисунку 4.9

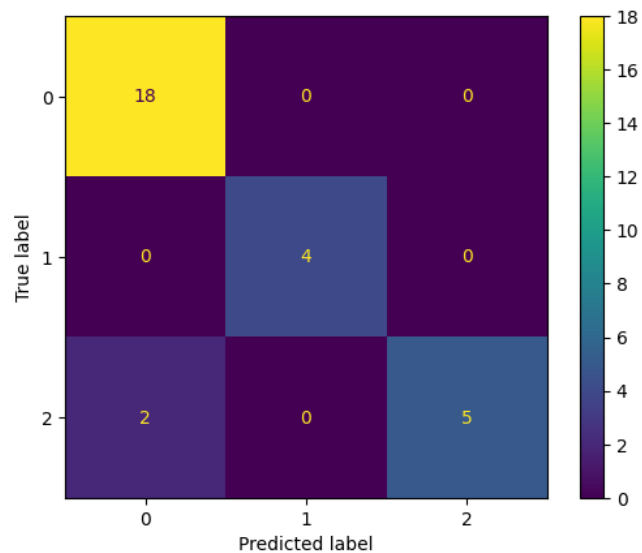


Рисунок 4.9 – Матриця похибок для тестового сету

Загальний звіт про роботу алгоритму представлено в таблиці 4.1

Таблиця 4.1 – Звіт про роботу алгоритму KNN

	влучність	відгук	F1- оцінка	кількість
0	0.97	1.00	0.99	74
1	1.0	1.00	1.00	13
2	1.0	0.94	0.97	32
Точність			0.98	119

Де влучність – здатність класифікатора не позначати негативний зразок як позитивний, розраховується як відношення правильно розпізнаних об’єктів одного класу до суми правильно розпізнаних об’єктів цього ж класу та об’єктів які помилково позначені як об’єкти цього класу.

Відгук – здатність класифікатора знаходити всі позитивні проби, розраховується як відношення правильно розпізнаних об’єктів класу до суми позитивно розпізнаних об’єктів та об’єктів які помилково позначені як інший клас.

F1 оцінка – зважене гармонічне середнє значення влучності та відгуку.
Точність – відношення правильно розпізнаних об’єктів до їх кількості.
Порівняємо з результатом класифікації алгоритмом SVM.
Результат роботи алгоритму на тренувальному сеті рисунок 4.10

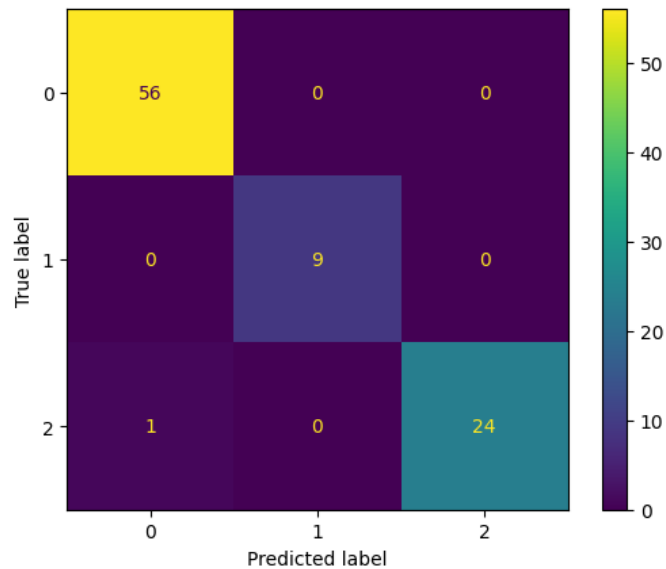


Рисунок 4.10 – Матриця похибок для тренувального сету

Результат класифікації для тестової вибірки, рисунок 4.11

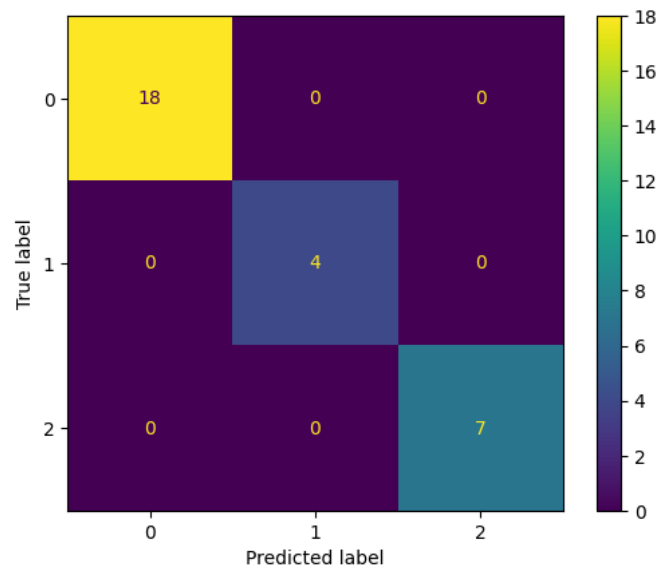


Рисунок 4.11 – Матриця похибок для тестового сету

Загальний звіт роботи алгоритму SVM представлено в таблиці 4.2

Таблиця 4.2 – Звіт про роботу алгоритму SVM

	влучність	відгук	F1- оцінка	кількість
0	0.99	1.00	0.99	74
1	1.0	0.92	1.00	13
2	1.0	0.97	0.98	32
Точність			0.99	119

Висновок: експериментальні дослідження дали позитивні результати та підтвердили працездатність розробленого програмного забезпечення.

ВИСНОВОК

В дипломній роботі було розроблено програмно-методичний комплекс сегментації осцилографічних зображень. Розроблене рішення використовує алгоритми машинного навчання для обробки експериментальних даних.

Кластеризація зображень дозволила поділити наявні дані на три кластери: мінімальна активність (таких зображень найбільше), помірна активність, надзвичайна активність (найменша кількість зображень).

Побудовано модель яка може класифікувати зображення з гарною точністю, та може бути використана для подальших досліджень тієї ж серії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LightsOnData [Електронний ресурс]: Режим доступу: <https://www.lightsondata.com/the-history-of-machine-learning/>
2. A. C. Muller, S. Guido Introduction to Machine Learning with Python – O`REILLY, 2016. – 376с.
3. T. Sagarán Programming Collective Intelligence – O`REILLY, 2007. – 323с.
4. Cluster Analysis B.S. Everitt, S. Landau, M. Leese, D. Stahl – WILEY, 2011 – 321с.
5. Документація Python [Електронний ресурс]: Режим доступу: <https://docs.python.org/3/tutorial/index.html>
6. Документація Numpy [Електронний ресурс]: Режим доступу: <https://numpy.org/doc/stable/>
7. Документація Pandas [Електронний ресурс]: Режим доступу: <https://pandas.pydata.org/docs/>
8. G. Bradski, A. Kaehler Learning OpenCV – O`REILLY, 2008. – 543с.
9. Digital Ocean [Електронний ресурс]: Режим доступу: <https://www.digitalocean.com/community/tutorials/how-to-install-run-connect-to-jupyter-notebook-on-remote-server>
10. E.U. Oti, M.O. Olusola, F.C. Eze, S.U. Enogwe «Comprehensive Review of K-Means Clustering Algorithms», International Journal of Advances in Scientific Research and Engineering, №7(8), с. 64-68, 2021
11. Towards datascience [Електронний ресурс]: Режим доступу: <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>
12. Towards datascience [Електронний ресурс]: Режим доступу: <https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec>

13. Документація Scipy [Електронний ресурс]: Режим доступу:
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html>
14. Kaggle [Електронний ресурс]: Режим доступу:
<https://www.kaggle.com/code/vipulgandhi/spectral-clustering-detailed-explanation>
15. Geeks for Geeks [Електронний ресурс]: Режим доступу:
<https://www.geeksforgeeks.org/ml-spectral-clustering/>
16. IBM [Електронний ресурс]: Режим доступу:
<https://www.ibm.com/topics/supervised-learning>
17. J. Nayak, B. Naik, H. Behera « A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges», International Journal of Database Theory and Application, № 8(1), с. 169-186, 2015
18. Geeks for Geeks [Електронний ресурс]: Режим доступу:
<https://www.geeksforgeeks.org/support-vector-machine-algorithm/>
19. IBM [Електронний ресурс]: Режим доступу:
<https://www.ibm.com/topics/knn>
20. Теличко Р.С., Синиця В.І. «Кластеризація подій в осцилограмах», Збірник праць XVI науково-практичної конференції студентів, аспірантів та молодих вчених, с. 261–264, 2023
21. STATOLOGY [Електронний ресурс]: Режим доступу:
<https://www.statology.org/elbow-method-in-python/>
22. Stack overflow [Електронний ресурс]: Режим доступу:
<https://stackoverflow.com/questions/42167907/understanding-scikitlearn-pca-transform-function-in-python>

ДОДАТОК А

Код програми

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import cv2

img1 = cv2.imread('photos/1.PNG')
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.imread('photos/2.PNG')
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
img3 = cv2.imread('photos/3.PNG')
img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)
img4 = cv2.imread('photos/4.PNG')
img4 = cv2.cvtColor(img4, cv2.COLOR_BGR2GRAY)
img5 = cv2.imread('photos/5.PNG')
img5 = cv2.cvtColor(img5, cv2.COLOR_BGR2GRAY)
img6 = cv2.imread('photos/6.PNG')
img6 = cv2.cvtColor(img6, cv2.COLOR_BGR2GRAY)
img7 = cv2.imread('photos/7.PNG')
img7 = cv2.cvtColor(img7, cv2.COLOR_BGR2GRAY)
img8 = cv2.imread('photos/8.PNG')
img8 = cv2.cvtColor(img8, cv2.COLOR_BGR2GRAY)
img9 = cv2.imread('photos/9.PNG')
img9 = cv2.cvtColor(img9, cv2.COLOR_BGR2GRAY)
img10 = cv2.imread('photos/10.PNG')
```

```

img10 = cv2.cvtColor(img10, cv2.COLOR_BGR2GRAY)
img11 = cv2.imread('photos/11.PNG')
img11 = cv2.cvtColor(img11, cv2.COLOR_BGR2GRAY)
img12 = cv2.imread('photos/12.PNG')
img12 = cv2.cvtColor(img12, cv2.COLOR_BGR2GRAY)
x1 = 49
images1 = [[] for i in range(10)]
for i in range(5):
    images1[i].append(img1[61:160,x1:(x1+141)].tolist())
    images1[9-i].append(img1[230:329,x1:(x1+141)].tolist())
    x1+= 212
x2 = 50
images2 = [[] for i in range(10)]
for i in range(5):
    images2[i].append(img2[45:144,x2:(x2+141)].tolist())
    images2[9-i].append(img2[211:310,x2:(x2+141)].tolist())
    x2+= 212
x3 = 65
y3 = 63
images3 = [[] for i in range(10)]
for i in range(5):
    images3[i].append(img3[35:134,x3:(x3+141)].tolist())
    images3[9-i].append(img3[198:297,y3:(y3+141)].tolist())
    x3+= 211
    y3+= 211
x4 = 50
images4 = [[] for i in range(5)]
for i in range(5):
    images4[i].append(img4[45:144,x4:(x4+141)].tolist())
    x4+= 212

```

```

x5 = 50
images5 = [[] for i in range(15)]
for i in range(5):
    images5[i].append(img5[52:151,x5:(x5+141)].tolist())
    images5[9-i].append(img5[215:314,x5:(x5+141)].tolist())
    images5[14-i].append(img5[377:476,x5:(x5+141)].tolist())
    x5+= 212
x6 = 60
images6 = [[] for i in range(10)]
for i in range(5):
    images6[i].append(img6[40:139,x6:(x6+141)].tolist())
    images6[9-i].append(img6[200:299,x6:(x6+141)].tolist())
    x6+= 212
x7 = 55
images7 = [[] for i in range(10)]
for i in range(5):
    images7[i].append(img7[55:154,x7:(x7+141)].tolist())
    images7[9-i].append(img7[220:319,x7:(x7+141)].tolist())
    x7+= 212
x8 = 50
images8 = [[] for i in range(15)]
for i in range(5):
    images8[i].append(img8[55:154,x8:(x8+141)].tolist())
    images8[9-i].append(img8[215:314,x8:(x8+141)].tolist())
    images8[14-i].append(img8[377:476,x8:(x8+141)].tolist())
    x8+= 212
x9 = 50
images9 = [[] for i in range(5)]
for i in range(5):
    images9[i].append(img9[55:154,x9:(x9+141)].tolist())

```

```

x9+= 212
x10 = 54
images10 = [[] for i in range(15)]
for i in range(5):
    images10[i].append(img10[55:154,x10:(x10+141)].tolist())
    images10[9-i].append(img10[216:315,x10:(x10+141)].tolist())
    images10[14-i].append(img10[380:479,x10:(x10+141)].tolist())
    x10+= 212
x11 = 60
images11 = [[] for i in range(10)]
for i in range(5):
    images11[i].append(img11[55:154,x11:(x11+141)].tolist())
    images11[9-i].append(img11[222:321,x11:(x11+141)].tolist())
    x11+= 212
x12 = 55
images12 = [[] for i in range(5)]
for i in range(5):
    images12[i].append(img12[57:156,x12:(x12+141)].tolist())
    x12+= 212
y = [[] for i in range(120)]
for i in range(10):
    if i <5:
        y[34-i] = images4[i]
        y[39-i] = images5[i]
        y[74-i] = images8[i]
        y[89-i] = images9[i]
        y[94-i] = images10[i]
        y[119-i] = images12[i]
    y[i] = images1[i]
    y[19-i] = images2[i]

```

```

y[29-i] = images3[i]
y[49-i] = images5[i+5]
y[59-i] = images6[i]
y[69-i] = images7[i]
y[84-i] = images8[i+5]
y[104-i] = images10[i+5]
y[114-i] = images11[i]
y.pop(38)
np.random.seed(100)
X = np.asarray(y)
X[:, :, 138:] = X[:, :, 138:] * 0
X[:, :, :3] = X[:, :, :3] * 0
np.random.shuffle(X)
X_scaled = X / 255
X_train = np.reshape(X_scaled.copy(), (119, -1))

from sklearn.cluster import KMeans
wcc = []
for i in range(1, 8):
    k_means = KMeans(n_clusters = i)
    k_means.fit(X_train)
    wcc.append(k_means.inertia_)
import seaborn as sns
k = np.linspace(1, 7, 7)
fig, ax = plt.subplots()
plt.scatter(3, 377, s = 80, facecolors = 'none', edgecolors='r')
sns.lineplot(x = k, y = wcc)
ax.set_xticks(k)
plt.xlabel('Кількість кластерів')
plt.ylabel('Сума квадратів помилок')

```

```
ax.grid()
```

```
k_means = KMeans(n_clusters = 3)
```

```
k_means.fit(X_train)
```

```
labels = k_means.labels_
```

```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=3)
```

```
pca_transformed = pd.DataFrame(pca.fit_transform(X_train))
```

```
from scipy.spatial import ConvexHull, convex_hull_plot_2d
```

```
fig, ax = plt.subplots()
```

```
plt.scatter(pca_transformed[labels==0][0], pca_transformed[labels==0][1],
```

```
label='Кластер 0', c='blue')
```

```
plt.scatter(pca_transformed[labels==1][0], pca_transformed[labels==1][1],
```

```
label='Кластер 1', c='red')
```

```
plt.scatter(pca_transformed[labels==2][0], pca_transformed[labels==2][1],
```

```
label='Кластер 2', c='green')
```

```
colors = ['blue','red','green']
```

```
for i in range(3):
```

```
    points =
```

```
[pca_transformed[labels==i][0].to_list(),pca_transformed[labels==i][1].to_list()]
```

```
    sns.kdeplot(points[0],points[1],color=colors[i], levels= 8, alpha = 0.5)
```

```
plt.title('KMeans')
```

```
plt.legend(loc=1)
```

```
plt.xlabel('PC1')
```

```
plt.ylabel('PC2')
```

```
plt.grid()
```

```
plt.show()
```

```

from scipy.cluster.hierarchy import dendrogram, linkage, leaves_list, cut_tree
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
linkage_data = linkage(X_train, method='ward', metric='euclidean')
fig, ax = plt.subplots()
dendrogram(linkage_data)
tick_names = leaves_list(linkage_data)
fig.set_figheight(10)
fig.set_figwidth(18)
plt.show()
from sklearn.cluster import AgglomerativeClustering
agc = AgglomerativeClustering(n_clusters=3).fit(X_train)
agc_res = agc.labels_
from scipy.spatial import ConvexHull, convex_hull_plot_2d
fig, ax = plt.subplots()
plt.scatter(pca_transformed[agc_res==0][0], pca_transformed[agc_res==0][1],
label='Кластер 0', c='blue')
plt.scatter(pca_transformed[agc_res==1][0], pca_transformed[agc_res==1][1],
label='Кластер 1', c='green')
plt.scatter(pca_transformed[agc_res==2][0], pca_transformed[agc_res==2][1],
label='Кластер 2', c='red')

colors = ['blue','red','green']
for i in range(3):
    points =
[pca_transformed[labels==i][0].to_list(),pca_transformed[labels==i][1].to_list()]
    sns.kdeplot(points[0],points[1],color=colors[i], levels= 8, alpha = 0.5)

plt.title('Агломераційний алгоритм')
plt.legend(loc=1)

```

```
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.grid()
plt.show()
```

```
from sklearn.cluster import SpectralClustering
spc = SpectralClustering(n_clusters=3,affinity='nearest_neighbors')
spc_lab = spc.fit_predict(pca_transformed)
```

```
fig, ax = plt.subplots()
plt.scatter(pca_transformed[spc_lab==0][0], pca_transformed[spc_lab==0][1],
label='Кластер 0', c='blue')
plt.scatter(pca_transformed[spc_lab==1][0], pca_transformed[spc_lab==1][1],
label='Кластер 1', c='green')
plt.scatter(pca_transformed[spc_lab==2][0], pca_transformed[spc_lab==2][1],
label='Кластер 2', c='red')
colors = ['blue','red','green']
for i in range(3):
    points =
[pca_transformed[labels==i][0].to_list(),pca_transformed[labels==i][1].to_list()]
    sns.kdeplot(points[0],points[1],color=colors[i], levels= 6, alpha = 0.5)
plt.title('Спектральный алгоритм')
plt.legend(loc=1)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.grid()
plt.show()

cor_df = pd.DataFrame ({
    'KMeans':labels,
    'Спектральный':spc_lab,
```

```

'Агломераційний':agc_res

})
matrix = cor_df.corr()
print(matrix)
import seaborn as sns
ax = sns.heatmap(matrix, annot=True)

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
y_pred = neigh.predict(X_train[90:])

cm = confusion_matrix(agc_res[90:],y_pred)

cf = ConfusionMatrixDisplay(confusion_matrix= cm)

cf.plot()

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
y_pred = neigh.predict(X_train[:90])

cm = confusion_matrix(agc_res[:90],y_pred)

cf = ConfusionMatrixDisplay(confusion_matrix= cm)

cf.plot()

from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score,
classification_report
y_pred = neigh.predict(X_train)
print(classification_report(agc_res, y_pred))

```

```
from sklearn.svm import SVC
clf = SVC(gamma='auto')
clf.fit(X_train[:90], agc_res[:90])
clf.score(X_train[90:], agc_res[90:])
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
y_pred1 = clf.predict(X_train[:90])
```

```
cm = confusion_matrix(agc_res[:90], y_pred1)
```

```
cf = ConfusionMatrixDisplay(confusion_matrix= cm)
```

```
cf.plot()
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
y_pred1 = clf.predict(X_train[90:])
```

```
cm = confusion_matrix(agc_res[90:], y_pred1)
```

```
cf = ConfusionMatrixDisplay(confusion_matrix= cm)
```

```
cf.plot()
```

```
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score,
classification_report
```

```
y_pred = clf.predict(X_train)
```

```
print(classification_report(agc_res, y_pred))
```

