Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

Кафедра радіоелектронних
і комп’ютерних систем

КУРСОВА РОБОТА

«Аналіз даних сенсорних систем засобами машиного навчання»

Виконав: ст.гр. ФеІм-14
_____ Осадчук О.Я.
(дата)
Науковий керівник:
_____ проф. Оленич І. Б.
(дата)

Львів – 2023

АНОТАЦІЯ

У курсовій роботі порівнюються і аналізуються різні методи машинного навчання у задачі розпізнавання типу газів. Аналіз проводиться на основі даних з сенсорних елементів, зібраних на кафедрі. У практичній частині порівнюються алгоритми класифікації SVM, Gradient Boosting, K-Neighbors та Decision Tree. Роботу алгоритмів досліджено у середовищі Jupyter на основі мови Python.

ANNOTATION

In the course work compares and analyzes various methods of machine learning in the task of recognizing the type of gases. The analysis is carried out on the basis of data from sensory elements collected at the department. In the practical part, classification algorithms SVM, Gradient Boosting, K-Neighbors and Decision Tree are compared. The work of the algorithms was studied in the Jupyter environment based on the Python language.

ЗМІСТ

ВСТУП5

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ ПРО СЕНСОРИ5

1.1 Загальні положення 6

1.2 Базова архітектура 8

1.3 Обробка сенсорних даних 10

1.4 Об’єданання даних 12

1.5 Аналіз даних 14

РОЗДІЛ 2. ВИКОРИСТАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ. РОЗРОБКА МОДЕЛЕЙ МАШИНОГО НАВЧАННЯ МОВОЮ PYTHON5

2.1 Переваги використання Python 16

2.2 Jupyter Notebook як середовище розробки мовою Python 18

2.3 Фреймворки Python та бібліотеки використані у роботі 20

2.3.1 NumPy 20

2.3.2 Pandas 22

2.3.3 MatPlotLib 23

2.3.4 Sclearn 25

РОЗДІЛ 3. ДЕМОНСТРАЦІЯ РЕЗУЛЬТАТІВ НАВЧАННЯ КЛАСИФІКАТОРА ГАЗІВ18

3.1 Використані алгоритми машиного навчання 26

3.1.1 Опис алгоритму класифікації SVM 26

3.1.2 Опис алгоритму класифікації Gradient Boosting 28

3.1.3 Опис алгоритму класифікації K-Neighbors 30

3.1.4 Опис алгоритму класифікації Decision Tree 32

3.2 Результати навчання та роботи класифікаторів 34
3.2.1 Опис сенсора на якому збирались дані 34
3.2.2 Процес навчання та огляд результатів роботи класифікатора 36
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 49

ВСТУП

Протягом останніх 50 років вчені досліджували можливості людського мозку та намагалися відтворити різні функції, які він виконує. У той же час, розвиток комп'ютерів пройшов по зовсім іншому шляху. Сучасні комп'ютерні архітектури, операційні системи та програмування мають мало спільного з обробкою інформації, яку здійснює мозок. Внаслідок цього, ми переоцінюємо здібності мозку, але моделі обробки інформації, які базуються на алгоритмах, стали широко поширеними.

Основним будівельним блоком цих моделей мозку, відомих як нейронні мережі, є блок обробки інформації, що моделює роботу нейрона. Штучний нейрон виконує прості математичні операції, а його ефективність залежить від того, як велика кількість нейронів з'єднана між собою, утворюючи мережу. Кожна нейронна модель відтворює різні здібності мозку і може бути використана для вирішення різних типів задач, таких як класифікація об'єктів, моделювання функціональних зв'язків, зберігання та пошук інформації, а також представлення великих обсягів даних. Хімічний аналіз, прогнозування реакцій та управління хімічними електростатичними потенціалами вже стали популярними областями застосування нейронних мереж.

У цій роботі описано роботу декількох алгоритмів класифікації, наведені їх переваги та недоліки, а також проведено порівняння цих алгоритмів машинного навчання. Для порівняння класифікаторів використовувались сенсорні дані, отримані в рамках науково-дослідної роботи на кафедрі радіоелектронних і комп'ютерних систем. Це дозволяє оцінити ефективність та придатність різних алгоритмів для обробки інформації в хімічних даних та різних задачах хімічного аналізу.

Розділ 1. Загальні положення про сенсори

1.1 Загальні положення

Сенсори є невід'ємною частиною нашого сучасного технологічного світу. Вони використовуються в різних галузях, починаючи від електроніки та промисловості до медицини та науки. Сенсори сприймають фізичні, хімічні або біологічні величини та перетворюють їх на вимірювання або сигнали, які можуть бути інтерпретовані.

Однією з ключових характеристик сенсорів є їх функція сприймання. Різні сенсори спеціалізуються на сприйманні різних типів величин, таких як температура, тиск, світло, звук, відстань, рух, вологість, хімічні речовини та багато інших. Кожен тип сенсора має властивості, що дозволяють йому сприймати конкретну величину та виконувати відповідні вимірювання.

Точність є ще однією важливою характеристикою сенсорів. Вона визначає, наскільки точно сенсор може вимірювати величину і вимірюється у відсотках або у величинах вимірювання. Чим вища точність сенсора, тим більш достовірні вимірювання він здійснює.

Діапазон вимірювання є іншою важливою характеристикою сенсорів. Він вказує на максимальне і мінімальне значення величини, яку сенсор може вимірювати. Наприклад, температурний сенсор може мати діапазон від -50°C до +150°C. Залежно від потреби, можуть бути доступні сенсори з різними діапазонами вимірювання.

Чутливість сенсора вказує на його спроможність реагувати на навколишні зміни величини, яку він сприймає. Висока чутливість означає, що сенсор реагує

на навколишні зміни, навіть незначної величини. Це особливо важливо в деяких застосуваннях, де потрібна висока реакція на зміну величини.

Багато сенсорів мають інтерфейс для передачі отриманих даних до системи або пристрою, який обробляє ці дані. Це може бути аналоговий або цифровий сигнал, який передається через провідники або безпроводові технології. Інтерфейс сенсора може варіюватися залежно від його призначення та вимог до передачі даних.

Надійність сенсорів є важливою характеристикою, оскільки вони повинні бути стійкими до зовнішніх впливів. Сенсори можуть бути піддані впливу вологості, пилу, вібрації, температурних змін та інших факторів. Висока надійність сенсора залежить від якості його компонентів та виготовлення.

Сенсори можуть мати різні розміри і форми, від мініатюрних до великих пристроїв, в залежності від їх застосування. Розмір і форма сенсора можуть впливати на його встановлення та використання в конкретному середовищі.

Взагалі, сенсори є ключовими компонентами технологічного прогресу і використовуються в різних сферах людської діяльності. Вони дозволяють нам отримувати важливі дані та інформацію про навколишній світ. Розуміння загальних положень про сенсори допомагає нам краще усвідомити їх значення та потенціал.

1.2 Базова архітектура

Базова архітектура сенсорів включає декілька ключових компонентів, які допомагають забезпечити їх правильну роботу. Ось основні компоненти базової архітектури сенсорів:

Сенсорний елемент: Це основна частина сенсора, яка взаємодіє з фізичною величиною, яку потрібно виміряти. Сенсорний елемент може бути виконаний у вигляді датчика, трансдюсера або спеціальної структури, здатної перетворювати фізичні зміни в електричні сигнали. Наприклад, температурний сенсор може мати термістор або термопару як свій сенсорний елемент.

Перетворювач сигналу: Цей компонент забезпечує перетворення сигналу, отриманого від сенсорного елемента, у форму, яку можна обробляти і аналізувати. Він може бути аналоговим або цифровим, залежно від типу сенсора та вимог до вихідного сигналу. Наприклад, аналоговий сенсор може використовувати опір або напругу як свій вихідний сигнал, тоді як цифровий сенсор може використовувати протокол передачі даних, такий як I2C або SPI.

Інтерфейс: Цей компонент відповідає за комунікацію між сенсором і зовнішніми пристроями або системою, яка обробляє дані. Інтерфейс може бути провідним або безпровідним. Провідний інтерфейс може використовувати різні комунікаційні протоколи, такі як UART або USB, для передачі даних. Безпровідний інтерфейс може використовувати радіочастотну технологію, таку як Bluetooth або Wi-Fi.

Живлення: Багато сенсорів потребують джерела живлення для своєї роботи. Це може бути батарея, джерело постійного струму або будь-яке інше джерело живлення, в залежності від вимог сенсора. Деякі сенсори можуть бути споживачами низької потужності, що дозволяє їх використання в мобільних пристроях або системах, де енергія є обмеженою.

Обробка даних: Цей компонент відповідає за обробку та аналіз отриманих даних з сенсора. Він може бути вбудованим в сам сенсор або відокремленим пристроєм, який отримує дані з сенсора через інтерфейс. Обробка даних може включати фільтрацію, калібрування, розрахунки або будь-яку іншу обробку, необхідну для отримання корисної інформації з отриманих вимірювань.

Це лише загальні компоненти, які можуть бути присутні в базовій архітектурі сенсорів. В реальних сенсорах можуть бути додаткові компоненти або функції, залежно від їх конкретних застосувань та вимог.

1.3 Обробка сенсорних даних

Обробка сенсорних даних є важливим етапом у використанні сенсорів для отримання корисної інформації. Цей процес включає аналіз, фільтрацію, калібрування та інші методи обробки, щоб перетворити сирий сигнал сенсора на корисну інформацію. Ось деякі загальні методи обробки сенсорних даних:

Фільтрація: Фільтрація даних є важливим процесом для видалення шуму, перешкод або артефактів зі сирих сенсорних сигналів. Це може бути досягнуто за допомогою різних фільтрів, таких як низькочастотні фільтри для зменшення шуму, високочастотні фільтри для видалення інтерференції та інші. Фільтрація допомагає отримати чистіші та більш точні дані з сенсорів.

Калібрування: Калібрування сенсорів полягає в налаштуванні сенсора для досягнення точності та надійності вимірювань. Це може включати корекцію відхилень, компенсацію показників температури або позиції, а також збалансування сенсорних параметрів. Калібрування допомагає забезпечити правильні вимірювання та усунути систематичні похибки.

Обробка сигналів: Обробка сигналів включає розрахунки та аналіз даних для отримання корисної інформації. Це може включати визначення статистичних характеристик сигналу, виявлення певних змін або подій, сегментацію сигналу для виділення основних компонентів та інші методи обробки. Обробка сигналів допомагає зрозуміти та інтерпретувати дані, отримані з сенсорів.

Інтерпретація та аналіз: Цей етап включає інтерпретацію оброблених даних та аналіз їх з метою отримання висновків або прийняття рішень. Це може включати порівняння зі стандартами, виявлення аномалій або відстеження змін у часі. Інтерпретація та аналіз сенсорних даних допомагає розуміти ситуацію та приймати обґрунтовані рішення.

Це лише загальні методи обробки сенсорних даних, і конкретні методи будуть залежати від типу сенсора, його застосування та вимог. Процес обробки сигналів може бути складним і вимагати використання спеціалізованих алгоритмів та програмного забезпечення.

1.4 Об’єданання даних

Об'єднання даних, також відоме як синтез даних або ф'южн даних, є процесом комбінування інформації з різних джерел або сенсорів з метою отримання більш повної та точної картини або розуміння досліджуваного явища чи середовища. Об'єднання даних може відбуватися на різних рівнях: рівень даних, рівень функцій та рівень знань. Ось кілька загальних методів об'єднання даних:

Об'єднання на рівні даних: На цьому рівні дані з різних джерел або сенсорів комбінуються для створення єдиного набору даних. Це може включати об'єднання сирих сигналів, вимірювань або векторів, отриманих від окремих джерел, у єдину матрицю або вектор. Методи об'єднання на рівні даних можуть включати прості операції, такі як злиття або конкатенація даних, а також більш складні алгоритми, такі як фільтрація, регресія або класифікація.

Об'єднання на рівні функцій: На цьому рівні інформація з різних джерел або сенсорів комбінується на основі конкретних функцій або моделей. Наприклад, можна використовувати методи калманівського фільтрування або часткової мутації для об'єднання даних з багатьох сенсорів і прогнозування стану системи на основі цих даних. Об'єднання на рівні функцій може використовувати статистичні, ймовірнісні, логічні або інші моделі для комбінування даних та отримання більш точних результатів.

Об'єднання на рівні знань: На цьому рівні об'єднуються високорівневі або контекстуальні знання з різних джерел або сенсорів для отримання глибшого розуміння. Наприклад, можна об'єднати інформацію з різних сенсорів та джерел, щоб розпізнавати складні об'єкти або ситуації. Об'єднання на рівні знань може використовувати експертні системи, штучний інтелект, машинне навчання або інші методи для аналізу та інтерпретації даних з метою винесення висновків або прийняття рішень.

Об'єднання даних відіграє важливу роль у багатьох сферах, включаючи розпізнавання образів, навігацію, моніторинг довкілля, медицину та багато інших. Воно дозволяє отримати більш повну, точну та зрозумілу інформацію з використанням різних джерел та сенсорів.

1.5 Аналіз даних

Аналіз даних - це процес виявлення, інтерпретації та отримання корисних знань зі зібраних даних. Він включає різні методи, техніки та інструменти для оцінки, розуміння та використання даних з метою виявлення тенденцій, залежностей, патернів та інших важливих аспектів.

Основні етапи аналізу даних:

- Зрозуміння задачі аналізу: Перед початком аналізу даних важливо чітко сформулювати мету та задачі, які потрібно вирішити. Це допоможе установити контекст та визначити підходи та методи, які необхідно застосовувати.
- Збір та підготовка даних: У цьому етапі здійснюється збір потрібних даних та їх підготовка для аналізу. Це може включати очищення даних, заповнення пропусків, перетворення форматів, видалення аномалій або інших процедур для підготовки даних до подальшого аналізу.
- Вибір методів аналізу: На цьому етапі визначаються методи, які будуть використовуватися для аналізу даних. Це можуть бути статистичні методи, машинне навчання, класифікація, кластеризація, регресія, візуалізація даних та інші підходи, залежно від задачі та характеристик даних.
- Виконання аналізу та інтерпретація результатів: На цьому етапі застосовуються обрані методи аналізу до даних, і отримані результати інтерпретуються для отримання корисних знань. Це може включати виявлення залежностей, важливих факторів, виявлення випадковості, прогнозування, вирішення проблем або прийняття рішень на основі отриманих висновків.

- Візуалізація та комунікація результатів: Один із важливих аспектів аналізу даних - це представлення результатів в зрозумілій та доступній формі. Це може бути графіки, діаграми, звіти, презентації або інші візуальні засоби, які допомагають комунікувати результати аналізу іншим людям.
- Перевірка та перегляд результатів: Після отримання результатів аналізу важливо перевірити їх достовірність та коректність. Це можна зробити шляхом перевірки чутливості до параметрів, повторного застосування методів аналізу на нових даних або порівняння з відомими результатами.

Важливо зазначити, що аналіз даних є ітеративним процесом, і може вимагати додаткових етапів та коригування підходів для досягнення бажаних результатів.

Розділ 2. Використане програмне забезпечення. Розробка моделей машиного навчання мовою Python

2.1 Переваги використання Python

Використання мови програмування Python має багато переваг, особливо у сфері аналізу даних. Ось кілька головних переваг використання Python:

- Простота вивчення та використання: Python має простий синтаксис, що робить його дуже легким для вивчення, навіть для початківців у програмуванні. Це робить Python популярним вибором для багатьох людей, що входять у сферу аналізу даних.
- Багатий екосистема бібліотек: Python має велику кількість бібліотек, призначених для аналізу даних, машинного навчання, візуалізації та інших завдань. Найвідоміші бібліотеки для аналізу даних включають NumPy, Pandas, Matplotlib, SciPy та Scikit-learn. Ці бібліотеки надають потужні функції та інструменти для роботи з даними та виконання аналізу.
- Широке застосування: Python є універсальною мовою програмування, що використовується не тільки в аналізі даних, але й у веб-розробці, наукових дослідженнях, штучному інтелекті, автоматизації та багатьох інших областях. Це означає, що вміння програмувати на Python може мати широкі можливості кар'єрного росту та застосування в різних сферах.
- Підтримка спільноти та документація: Python має велику та активну спільноту розробників, яка надає підтримку, допомогу та спільну роботу над проектами. Крім того, Python має високоякісну

документацію, яка допомагає розробникам швидко знайти необхідну інформацію та розв'язати проблеми.

- Розширюваність та інтеграція з іншими мовами: Python може бути легко поєднаний з кодом, написаним на інших мовах програмування, таких як C, C++, Java. Це дає можливість використовувати швидкі обчислення або спеціалізовані бібліотеки для певних завдань, які можуть бути більш ефективними.
- Платформонезалежність: Python підтримується на багатьох операційних системах, таких як Windows, macOS, Linux, і може запускатися на різних апаратних платформах. Це робить його варіантом для розробки крос-платформових програм та аналізу даних на різних пристроях.

Python є потужним і гнучким інструментом для аналізу даних, який має широку підтримку, багато бібліотек та простоту використання, що робить його популярним вибором серед аналітиків даних та програмістів.

2.2 Jupyter Notebook як середовище розробки мовою Python

Jupyter Notebook - це інтерактивне середовище розробки, яке дозволяє створювати та виконувати код мовою Python у вигляді окремих комірок, а також комбінувати код, текст та візуалізації в одному документі.

Інтерактивність: Jupyter Notebook надає можливість виконувати код у вигляді окремих комірок, що дозволяє миттєво бачити результати та проводити експерименти. Це зручно під час аналізу даних, візуалізації та тестування коду.

Візуалізація даних: Jupyter Notebook підтримує інтеграцію з багатьма бібліотеками візуалізації, такими як Matplotlib, Seaborn, Plotly, що дозволяє зручно створювати графіки, діаграми та інші візуалізації прямо в ноутбуці.

Комбінування коду та документації: Jupyter Notebook дозволяє вводити текстові комірки за допомогою розмітки Markdown, що дозволяє описувати код, коментарі, пояснення та результати аналізу. Це допомагає створювати зрозумілі та зрізані документи, які можна легко поділитися з іншими.

Реорганізація та повторне використання коду: Jupyter Notebook дозволяє зміщувати, редагувати та повторно використовувати окремі комірки коду в будь-якому порядку. Це дозволяє розробникам легко організовувати та тестувати різні частини коду, а також повторно використовувати певні фрагменти.

Підтримка різних мов програмування: Jupyter Notebook не обмежується лише Python і підтримує багато мов програмування, таких як R, Julia, Scala та інші. Це дає можливість розробляти проекти, використовуючи різні мови програмування в одному документі.

Велика спільнота та розширення: Jupyter Notebook має широку спільноту користувачів та розробників, яка надає багато розширень та додаткових інструментів. Наприклад, JupyterLab - це розширена версія Jupyter Notebook з багатьма додатковими можливостями, яка надає більш гнучке та розширене середовище розробки.

Jupyter Notebook є популярним інструментом серед аналітиків даних та програмістів, оскільки він дозволяє поєднувати код, документацію та візуалізації в одному місці, спрощує експерименти та спільну роботу, а також надає зручне середовище для аналізу та розробки мовою Python.

2.3 Фреймворки Python та бібліотеки використані у роботі

2.3.1 NumPy

Бібліотека NumPy (Numerical Python) є однією з найпопулярніших та потужних бібліотек для числових обчислень у мові програмування Python. Вона була створена з метою полегшити та оптимізувати роботу з числовими даними, зокрема з масивами чисел.

Одним з головних принципів NumPy є робота з масивами, що дозволяє ефективно зберігати та опрацьовувати числові дані. Масиви NumPy є багатовимірними структурами даних, які можуть мати від 1 до N вимірів. Це дозволяє зручно та ефективно працювати зі структурованими даними, такими як вектори, матриці та тензори.

Однією з найбільших переваг NumPy є його швидкість та продуктивність. Внутрішньо бібліотека використовує оптимізовані алгоритми та операції над масивами, що дозволяє прискорити обчислення. Це особливо важливо при роботі з великими обсягами даних, де швидкість обробки має велике значення.

NumPy надає велику кількість вбудованих функцій та операцій, які дозволяють виконувати різні математичні, логічні та статистичні обчислення. Це включає в себе операції знаходження максимального та мінімального значення, обчислення середнього значення, сортування, фільтрацію та багато інших.

Бібліотека NumPy також відкриває доступ до розширеного функціоналу для індексації та зрізів масивів. Це дозволяє легко виконувати операції над конкретними елементами масиву, отримувати підмасиви за певними умовами та виконувати інші операції, що спрощують маніпулювання даними.

NumPy має широке застосування у багатьох галузях, включаючи наукові дослідження, обробку та аналіз даних, статистику, машинне навчання, комп'ютерне зору, обробку сигналів та інші. Вона є основою для інших популярних бібліотек та фреймворків, таких як Pandas, SciPy, scikit-learn, TensorFlow та інші, що розширюють її функціональність та дозволяють розв'язувати складні завдання у наукових та інженерних областях.

NumPy є потужним інструментом для числових обчислень у мові програмування Python. Вона надає засоби для ефективної роботи з масивами чисел, оптимізовані алгоритми та широкий функціонал для обробки та аналізу даних. Її використання дозволяє забезпечити швидкість, продуктивність та зручність в роботі з числовими даними.

2.3.2 Pandas

Pandas є однією з найпопулярніших бібліотек для обробки та аналізу даних у мові програмування Python. Вона надає потужні засоби для роботи з табличними даними, такими як даних у форматі CSV, Excel, SQL і багатьох інших.

Однією з головних переваг Pandas є його структура даних, відома як DataFrame. DataFrame є двовимірною таблицею даних з рядками та стовпцями, подібно до таблиць у базах даних або електронних таблицях. Він дозволяє зручно маніпулювати та аналізувати дані, здійснювати фільтрацію, сортування, групування, об'єднання та багато інших операцій.

Pandas надає велику кількість функцій та методів для роботи з даними. Це включає в себе зчитування та запис даних у різних форматах, індексування та вибірка даних, обчислення статистичних показників, заповнення пропущених значень, виконання операцій над стовпцями та рядками, візуалізацію даних і багато іншого.

Однією з сильних сторін Pandas є його інтеграція з іншими бібліотеками для наукових обчислень та візуалізації, такими як NumPy, Matplotlib і Seaborn. Це дозволяє зручно поєднувати функціональність цих бібліотек для виконання складних аналітичних завдань.

Pandas знайшов широке застосування у багатьох галузях, включаючи наукові дослідження, фінанси, економіку, соціальні науки, аналітику даних, машинне навчання та багато інших. Він є незамінним інструментом для роботи з великими обсягами даних та виконання складних аналітичних завдань.

Pandas є потужним інструментом для обробки та аналізу даних у мові програмування Python. Він надає зручні засоби для роботи з табличними даними, має багато функціональності для маніпулювання та аналізу даних.

2.3.3 MatPlotLib

Matplotlib є однією з найбільш використовуваних бібліотек для візуалізації даних у мові програмування Python. Вона була створена у 2003 році Джоном Гантем на базі бібліотеки для графічного відображення MATLAB. Протягом років Matplotlib набула великої популярності і стала стандартом у галузі візуалізації даних у Python.

Головною метою Matplotlib є забезпечення потужних та гнучких засобів для створення високоякісних графіків та візуалізацій. Вона пропонує широкий спектр можливостей, що дозволяють відтворювати графіки з високою точністю, незалежно від їх складності. Вона підтримує різні типи графіків, включаючи лінійні графіки, стовпчикові діаграми, кругові діаграми, графіки розсіювання, контурні графіки, теплові карти та багато інших.

Matplotlib надає розширені можливості налаштування візуалізацій, дозволяючи контролювати кожну деталь графічного представлення. За допомогою багатьох функцій та методів можна змінювати колір, стиль та товщину ліній, встановлювати підписи осей, додавати легенду, налаштовувати шрифти та багато іншого. Всі ці можливості дозволяють створювати графіки, які відповідають конкретним потребам та стилістиці візуалізації даних.

Matplotlib також підтримує роботу з багатьма форматами виведення, такими як PNG, JPEG, PDF, SVG, що дозволяє зберігати графіки в потрібному форматі для подальшого використання. Крім того, бібліотек
а має інтерактивний режим, що дозволяє маніпулювати графіками під час виконання програми, збільшувати, перетворювати та зберігати їх в реальному часі.

Matplotlib знайшла широке застосування у багатьох галузях, включаючи наукові дослідження, аналіз даних, фінансову аналітику, машинне навчання, візуалізацію веб-даних, графічний дизайн та багато інших. Вона є важливим

інструментом для відображення та сприйняття даних, дозволяючи створювати інформативні та привабливі візуалізації, які допомагають зрозуміти та знайти залежності, виявити тенденції та розкрити складність даних.

2.3.4 Sclearn

Scikit-learn, також відомий як sklearn, є однією з найпопулярніших бібліотек машинного навчання у мові програмування Python. Вона надає широкий спектр алгоритмів та інструментів для класифікації, регресії, кластеризації, зменшення розмірності, виявлення аномалій та багатьох інших завдань машинного навчання.

Однією з головних переваг Scikit-learn є його простота використання та уніфікований інтерфейс, що дозволяє легко виконувати стандартні процеси машинного навчання, такі як навчання моделей, оцінювання та прогнозування. Вона також надає зручний інструментарій для підготовки та очищення даних, а також для оцінки та вибору моделей з використанням метрик якості.

Scikit-learn має велику колекцію реалізованих алгоритмів машинного навчання, включаючи класифікатори, регресори, кластеризатори, детектори аномалій, зменшувачі розмірності та інші. Вона також надає інструменти для роботи з текстовими даними, зображеннями та іншими типами даних.

Scikit-learn базується на інших популярних бібліотеках Python, таких як NumPy, SciPy та Matplotlib, що робить його інтеграцію з цими інструментами простою та зручною. Вона також підтримує розширення функціональності за допомогою різноманітних плагінів та розширень.

Scikit-learn знаходить широке застосування в різних галузях, включаючи науку про дані, фінанси, медицину, соціальні науки, маркетинг та багато інших. Вона є важливим інструментом для вирішення задач машинного навчання та аналізу даних, дозволяючи легко експериментувати з різними моделями, налаштовувати їх та робити прогнози на основі навчених моделей.

Розділ 3. Демонстрація результатів навчання класифікатора газів

3.1 Використані алгоритми машиного навчання

3.1.1 Опис алгоритму класифікації SVM

Алгоритм опорних векторів (SVM) є потужним методом класифікації, який широко використовується в машинному навчанні. Він базується на ідеї знайти гіперплощину, яка максимально відділяє два або більше класи даних. Цей алгоритм знайшов широке застосування в різних галузях, таких як обробка зображень, біоінформатика, фінанси, медицина та багато інших.

Основні кроки алгоритму SVM включають наступне:

1. Підготовка даних: Починаємо з підготовки наших даних, включаючи вибір навчального набору даних та підготовку ознак для подальшого використання. Це може включати масштабування ознак, видалення аномальних значень або нормалізацію даних.
2. Вибір ядра: SVM може працювати з лінійно роздільними даними, але він також може використовувати ядерні функції для розв'язання нелінійних задач класифікації. Ядро визначає перетворення простору ознак, що дозволяє побудувати нелінійну границю рішень. Найпоширеніші ядра, які використовуються в SVM, це лінійне, поліноміальне, радіальне та сигмоїдне.
3. Побудова границі рішень: Наступним кроком є побудова гіперплощини, яка максимально відділяє класи даних. Гіперплощина визначається таким чином, щоб максимізувати відстань між нею та найближчими точками кожного класу, які називаються опорними векторами. Цей процес зводиться до розв'язання оптимізаційної задачі,

яка забезпечує максимальну віддаль від опорних векторів до гіперплощини.

4. Оптимізація гіперплощини: Для побудови оптимальної гіперплощини використовується метод опорних векторів. Цей метод включає мінімізацію функціоналу втрат та налагання обмежень на параметри моделі. Оптимізація може бути проведена за допомогою різних числових методів, таких як метод підйому градієнта або метод внутрішньої точки.
5. Класифікація нових зразків: Після побудови моделі SVM, можемо використовувати її для класифікації нових зразків. Застосовуємо функцію прийняття рішень до нових даних, яка визначає, до якого класу належить зразок, залежно від його розташування відносно гіперплощини.

SVM має кілька важливих властивостей:

- Висока ефективність: SVM використовує ефективні алгоритми для розв'язання оптимізаційної задачі, що дозволяє працювати з великими наборами даних.
- Робастність до шуму: SVM має вбудовану регуляризацію, що робить його робастним до шуму та аномалій в даних.
- Гнучкість: Застосування різних ядерних функцій дозволяє SVM моделювати нелінійні залежності між ознаками.
- Інтерпретованість: SVM надає інтерпретованість, оскільки опорні вектори можуть служити для розуміння впливу кожної ознаки на розділення класів.

Застосування SVM включає такі області, як класифікація тексту, розпізнавання образів, біоінформатика, фінанси, медицина та інші. Його можна

використовувати для багатьох видів задач класифікації, де важлива точність та здатність роботи з різноманітними типами даних.

3.1.2 Опис алгоритму класифікації Gradient Boosting

Алгоритм градієнтного бустінгу є потужним методом машинного навчання, який використовується для задач класифікації та регресії. Він поєднує в собі декілька слабких моделей (зазвичай дерев рішень) та навчає їх послідовно, додавши нову модель, яка коригує помилки попередніх моделей. Цей процес триває досягнення оптимальної точності або досягнення заданого критерію зупинки.

Основні кроки алгоритму градієнтного бустінгу включають наступне:

1. Підготовка даних: Починаємо з підготовки навчального набору даних, включаючи вибір та підготовку ознак для подальшого використання. Це може включати масштабування ознак, видалення аномальних значень або обробку пропущених даних.
2. Базова модель: Встановлюємо початкову базову модель, зазвичай це дерево рішень. Це буде першою моделлю, яка буде навчатись на нашому навчальному наборі даних.
3. Резидуальна помилка: Обчислюємо резидуальну помилку між прогнозами базової моделі та реальними значеннями. Резидуальна помилка вказує, де базова модель погано передбачає дані.
4. Градієнтний спуск: Наступним кроком є обчислення градієнта функції втрат на основі резидуальної помилки. Градієнт представляє собою напрямок та величину, в якому повинні змінити параметри моделі, щоб зменшити помилку.
5. Побудова нової моделі: Будуємо нову модель, яка навчається передбачати резидуальну помилку, використовуючи градієнтний спуск. Ця нова модель стає наступним "корекційним" кроком, який додається до попередньої базової моделі.

6. Ансамблювання моделей: Додаємо нову модель до ансамблю, об'єднуючи прогнози всіх попередніх моделей разом з прогнозом нової моделі. Це дозволяє покращити прогнози і зменшити загальну помилку.
7. Повторення кроків: Повторюємо кроки 3-6 декілька разів, додаючи нові моделі та коригуючи помилки попередніх моделей, поки не досягнемо критерію зупинки (наприклад, заданої точності моделі) або не досягнено максимальної кількості ітерацій.
8. Прогнозування: Після завершення процесу навчання можна використовувати отриману модель для прогнозування класів або значень на нових невідомих даних.

Градієнтний бустінг є потужним методом, оскільки він може адаптуватися до складних залежностей в даних та робити точні прогнози. Він широко використовується в багатьох областях, включаючи обробку природних мов, рекомендаційні системи, обробку зображень та багато інших. Його гнучкість та ефективність роблять його одним з найпопулярніших алгоритмів машинного навчання.

3.1.3 Опис алгоритму класифікації K-Neighbors

Алгоритм K-Neighbors (K-найближчих сусідів) є одним з простих, але ефективних алгоритмів класифікації. Він використовуєся для вирішення задач класифікації, де об'єкти класифікуються на основі їх схожості з навчальним набором даних. Алгоритм K-Neighbors використовуєся для призначення класу новому об'єкту на основі його K найближчих сусідів.

Основні кроки алгоритму K-Neighbors включають наступне:

1. Підготовка даних: Починаємо з підготовки навчального набору даних, включаючи вибір та підготовку ознак. Також розділяємо навчальний набір даних на навчальний та валідаційний набори.
2. Визначення K: Визначаємо значення K, яке вказує на кількість найближчих сусідів, які будуть використовуватись для класифікації нового об'єкта. Зазвичай значення K вибирається експериментально або за допомогою методу перехресної перевірки.
3. Вимірювання відстані: Обчислюємо відстань між новим об'єктом та всіма об'єктами у навчальному наборі даних. Часто використовується Евклідова відстань, але можуть використовуватися й інші метрики відстані.
4. Вибір K найближчих сусідів: Обираємо K найближчих сусідів нового об'єкта на основі їх відстані. Це може бути зроблено шляхом сортування об'єктів за відстанню та вибору перших K об'єктів.
5. Визначення класу: Визначаємо клас нового об'єкта на основі класів K найближчих сусідів. Це може бути зроблено шляхом використання голосування більшості (наприклад, бінарне голосування у випадку двох класів) або використання вагових коефіцієнтів, що залежать від відстані до кожного сусіда.

Алгоритм K-Neighbors використовується в багатьох областях, таких як розпізнавання образів, обробка природних мов, рекомендаційні системи та біоінформатика. Він простий у реалізації і має високу гнучкість, але вимагає належної обробки та підготовки даних. Крім того, вибір оптимального значення K та відповідної метрики відстані є важливими факторами успіху алгоритму.

3.1.4 Опис алгоритму класифікації Decision Tree

Алгоритм класифікації Decision Tree (дерево рішень) є одним з найпопулярніших і зрозумілих алгоритмів машинного навчання для вирішення задач класифікації. Він моделює прийняття рішень на основі деревоподібної структури, в якій кожен вузол представляє признак, а коже- навчальному наборі даних, він розподіляє їх на підмножини на основі значень признаків. Алгоритм дерева рішень може бути використаний для класифікації нових невідомих об'єктів шляхом пройдення по розгалуженнях дерева на основі значень їх признаків.

Основні кроки алгоритму Decision Tree включають наступне:

1. Побудова дерева: Починаємо з побудови порожнього дерева. На кожному кроці алгоритм вибирає найкращий признак, який розділяє навчальний набір даних на дві підмножини. Вибір найкращого признака зазвичай здійснюється на основі певної метрики, такої як ентропія або покращення Gini.
2. Рекурсивна побудова: Після вибору признака, алгоритм рекурсивно повторює крок побудови дерева для кожної підмножини даних. Процес продовжується до досягнення певної умови зупинки, наприклад, коли всі об'єкти в підмножині належать до одного класу або досягнуто максимальної глибини дерева.
3. Класифікація: Після побудови дерева можна використовувати його для класифікації нових об'єктів. Новий об'єкт проходить через розгалуження дерева, починаючи з кореня, і йому призначується клас, який є більшістю об'єктів відповідного листка.

Алгоритм Decision Tree має кілька переваг, таких як простота інтерпретації, здатність моделювати нелінійні залежності та враховувати важливість признаків. Він також може використовуватись для вирішення задач

класифікації та регресії. Однак, важливо враховувати, що дерева рішень можуть схильні до перенавчання, особливо при великій глибині дерева або великій кількості признаків. Це може бути вирішено за допомогою технік обрізки дерева або використання ансамблевих методів, таких як random forest.

3.2 Результати навчання та роботи класифікаторів

3.2.1 Опис сенсора на якому збирались дані

Для виконання науково-дослідної роботи "Багатофункціональні нанокомпозити на основі кремнію та карбону для інтелектуальних систем екологічного та радіаційного моніторингу" на кафедрі радіоелектронних і комп'ютерних систем використовувалися сенсорні дані. Чутливим елементом сенсорної системи був польовий транзистор з гібридною структурою, який складався з поруватого кремнію (por-Si) та плівки відновленого оксиду графену (rGO).

У даній системі, як провідний канал польового транзистора, використовувалася плівка rGO, що була нанесена на поверхню por-Si. Графен, завдяки своїй безщілинній конусоподібній енергетичній структурі, має високу мобільність носіїв заряду, високу електропровідність і можливість легко впроваджувати носії заряду (позитивного або негативного) за допомогою зовнішнього електричного поля. Тому, провідність графенової плівки можна контролювати за допомогою напруги затвору (UGs). Крім того, локальне електричне поле адсорбованих молекул також впливає на електропровідність графену. Це дозволяє використовувати гібридну структуру por-Si-rGO в режимі польового транзистора як газочутливий сенсорний елемент.

У польовому транзисторі з графеновим каналом спостерігається мінімум провідності (максимум опору) в точці Дірака, яка відповідає нейтральності заряду. Локальне електричне поле, створене адсорбованими газами, змінює електронейтральність графену, що призводить до зсуву положення точки Дірака (максимуму електричного опору). Цей зсув залежить від концентрації та природи адсорбованих газів. Використовуючи цю особливість, сенсорна

система досліджувала залежності опору чутливого елемента від концентрації молекул аміаку, ацетону, етанолу та водяної пари біля точки Дірака.

Зазначені адсорбо-електричні ефекти є основою для використання цієї сенсорної системи в розпізнаванні газів та визначенні їх концентрації. При зміні концентрації адсорбованих газів, відбуваються відповідні зміни в опорі чутливого елемента. Це дозволяє виявляти та кількісно аналізувати присутність різних газів у досліджуваному середовищі.

Отримані залежності опору від концентрації аміаку, ацетону, етанолу, толуолу та водяної пари дозволяють використовувати цю сенсорну систему для ідентифікації цих газів та визначення їх концентрації в зразках. Цей підхід заснований на взаємодії газів з поверхнею графену, що змінює електричні характеристики сенсорного елемента.

3.2.2 Процес навчання та огляд результатів роботи класифікатора

Для дослідження було використано вищевказані алгоритми класифікації SVM, GradientBoosting, K-Neighbors та Decision Tree.

На сенсор попадали випари етанолу, ацетону, аміаку та толуолу у близько 30 концентраціях. Для кожної концентрації отримували дані опору активного шару сенсора при напругах від 0 до 1.5 Вольт з інтервалом в 0.1 Вольт.

Дані представлено у вигляді 4 сторінок таблиць у Excel.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
19	4.5	0.62696	0.68327	0.75381	0.8492	0.91833	0.91035	0.82982	0.74756	0.69105	0.6421	0.60711	0.58089	0.54942	0.53227	0.51536	0.5021
20	4.75	0.62289	0.67907	0.74934	0.84454	0.91435	0.9057	0.82603	0.74378	0.68697	0.63766	0.60244	0.57621	0.54627	0.52921	0.51233	0.49924
21	5	0.61903	0.6751	0.74511	0.84	0.91047	0.90129	0.82244	0.74018	0.68375	0.63402	0.59856	0.57228	0.54333	0.52636	0.50951	0.49657
22	5.25	0.61478	0.6707	0.74041	0.83519	0.90633	0.89654	0.81855	0.73632	0.67975	0.62965	0.59397	0.56766	0.53981	0.52294	0.50613	0.49338
23	5.5	0.61071	0.6665	0.73593	0.82976	0.90152	0.89338	0.81609	0.73375	0.67661	0.6261	0.59017	0.56381	0.53625	0.51948	0.50272	0.49014
24	5.75	0.60645	0.6621	0.73123	0.82387	0.89622	0.88898	0.81251	0.73017	0.67225	0.62139	0.58525	0.55888	0.53198	0.51533	0.49864	0.48626
25	6	0.60226	0.65777	0.72661	0.81887	0.89186	0.88421	0.8086	0.72629	0.66854	0.61729	0.58093	0.55452	0.52833	0.51179	0.49515	0.48295
26	6.25	0.59783	0.65318	0.7217	0.81358	0.88721	0.87977	0.80499	0.72267	0.66503	0.61339	0.57679	0.55034	0.52507	0.50863	0.49202	0.48
27	6.5	0.59343	0.64863	0.71684	0.80864	0.88292	0.8752	0.80125	0.71895	0.6617	0.60966	0.57281	0.54632	0.52115	0.50481	0.48827	0.47643
28	6.75	0.5893	0.64436	0.71228	0.80332	0.87823	0.8697	0.79668	0.71447	0.65809	0.60565	0.56857	0.54204	0.51701	0.5008	0.48432	0.47268
29	7	0.58459	0.63947	0.70705	0.79812	0.87366	0.86542	0.79321	0.71098	0.65466	0.60182	0.5645	0.53793	0.51384	0.49772	0.48127	0.4698
30	7.25	0.57945	0.63412	0.7013	0.79275	0.86891	0.86183	0.79037	0.70806	0.65113	0.5979	0.56034	0.53373	0.50982	0.49382	0.47743	0.46615
31	7.5	0.57525	0.62978	0.69667	0.78791	0.86473	0.85812	0.78742	0.70505	0.64756	0.59394	0.55615	0.5295	0.50591	0.49002	0.47369	0.46259
32	7.75	0.57117	0.62557	0.69218	0.78263	0.86009	0.85463	0.78466	0.70222	0.64409	0.59007	0.55204	0.52535	0.50241	0.48663	0.47034	0.45942
33	8	0.56726	0.62155	0.68789	0.77724	0.85532	0.85063	0.78144	0.69896	0.64031	0.58591	0.54765	0.52093	0.49895	0.48326	0.46702	0.45628
34	8.25	0.56334	0.61751	0.68359	0.77181	0.85052	0.84559	0.77729	0.69486	0.63665	0.58186	0.54337	0.51661	0.49538	0.4798	0.4636	0.45304
35	8.5	0.55941	0.61347	0.67929	0.76578	0.84507	0.84157	0.77406	0.69159	0.63298	0.57781	0.53909	0.51229	0.49179	0.47631	0.46016	0.44978
36	8.75	0.5554	0.60934	0.67488	0.76035	0.84026	0.83765	0.77092	0.68841	0.62963	0.57405	0.53509	0.50825	0.48823	0.47286	0.45675	0.44655
37	9	0.55131	0.60511	0.67038	0.75505	0.83558	0.83394	0.76796	0.68539	0.62582	0.56986	0.53067	0.5038	0.48467	0.4694	0.45334	0.44332
38	C, %	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1.1	1.2	1.3	1.4	1.5
39																	
40																	

ethanol | ammonia | acetone | **toluol**

Рисунок 3.1 Формат вхідних даних

Дані для різних речовин (етанол, аміак, ацетон, толуол) завантажуються з файлу Excel і зберігаються у відповідних об'єктах DataFrame. Перейменовуються стовпці на основі останнього рядка, встановлюється індекс та видаляється останній рядок.

toluol

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0.70446	0.76343	0.83945	0.94233	1.00000	0.98624	0.89122	0.80931	0.75623	0.71429	0.68349	0.65793	0.63434	0.61468	0.59633	0.57929
1	0.25	0.69985	0.75865	0.83433	0.93732	0.99563	0.98160	0.88743	0.80553	0.75230	0.70999	0.67897	0.65339	0.62862	0.60913	0.59089	0.57409
2	0.5	0.69483	0.75341	0.82872	0.93229	0.99125	0.97731	0.88395	0.80204	0.74845	0.70576	0.67451	0.64890	0.62304	0.60372	0.58558	0.56901
3	0.75	0.69018	0.74859	0.82356	0.92764	0.98727	0.97262	0.88011	0.79822	0.74459	0.70152	0.67005	0.64441	0.61765	0.59849	0.58045	0.56411
4	1	0.68599	0.74427	0.81894	0.92222	0.98247	0.96995	0.87810	0.79606	0.74083	0.69738	0.66568	0.64001	0.61232	0.59331	0.57537	0.55926
5	1.25	0.68156	0.73967	0.81402	0.91697	0.97786	0.96518	0.87419	0.79218	0.73711	0.69328	0.66136	0.63565	0.60675	0.58791	0.57007	0.55419
6	1.5	0.67751	0.73550	0.80957	0.91208	0.97362	0.96095	0.87076	0.78873	0.73343	0.68922	0.65706	0.63132	0.60125	0.58257	0.56484	0.54919
7	1.75	0.67322	0.73106	0.80484	0.90753	0.96974	0.95671	0.86733	0.78528	0.72974	0.68513	0.65275	0.62697	0.59592	0.57740	0.55977	0.54434
8	2	0.66889	0.72658	0.80005	0.90269	0.96556	0.95226	0.86371	0.78166	0.72621	0.68122	0.64859	0.62278	0.59060	0.57224	0.55471	0.53951
9	2.25	0.66488	0.72245	0.79564	0.89778	0.96130	0.94793	0.86019	0.77813	0.72278	0.67738	0.64452	0.61866	0.58537	0.56716	0.54972	0.53475
10	2.5	0.66072	0.71815	0.79105	0.89252	0.95668	0.94417	0.85720	0.77508	0.71918	0.67339	0.64029	0.61440	0.58096	0.56288	0.54551	0.53074
11	2.75	0.65638	0.71366	0.78626	0.88730	0.95209	0.93946	0.85333	0.77124	0.71563	0.66944	0.63611	0.61018	0.57677	0.55882	0.54151	0.52694
12	3	0.65252	0.70969	0.78203	0.88187	0.94728	0.93550	0.85016	0.76802	0.71200	0.66543	0.63187	0.60590	0.57310	0.55525	0.53799	0.52360
13	3.25	0.64828	0.70531	0.77735	0.87656	0.94260	0.93180	0.84722	0.76502	0.70858	0.66161	0.62780	0.60178	0.56876	0.55104	0.53385	0.51965
14	3.5	0.64374	0.70060	0.77232	0.87049	0.93711	0.92773	0.84394	0.76171	0.70505	0.65769	0.62364	0.59759	0.56446	0.54687	0.52975	0.51575
15	3.75	0.63923	0.69593	0.76731	0.86541	0.93267	0.92312	0.84017	0.75795	0.70151	0.65375	0.61947	0.59338	0.56002	0.54256	0.52551	0.51171
16	4	0.63514	0.69171	0.76281	0.86006	0.92795	0.91902	0.83686	0.75462	0.69795	0.64979	0.61528	0.58915	0.55610	0.53875	0.52176	0.50815
17	4.25	0.63128	0.68774	0.75859	0.85499	0.92352	0.91524	0.83385	0.75155	0.69452	0.64597	0.61121	0.58503	0.55286	0.53561	0.51866	0.50522
18	4.5	0.62696	0.68327	0.75381	0.84920	0.91833	0.91035	0.82982	0.74756	0.69105	0.64210	0.60711	0.58089	0.54942	0.53227	0.51536	0.50210

Рисунок 3.2 Датафрейм з вхідними даними

Розширюємо датафрейми за допомогою заповнення пропущених значень шляхом обчислення середнього значення між двома сусідніми стовпцями. Ці кроки застосовуються до кожної речовини. Також виконуються обчислення середнього значення між двома сусідніми рядками для кожної речовини. Отримані значення додаються до відповідних DataFrame.

ethanol

	0.0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	...	1.1	1.15	1.2	1.25	1.3	1.35
0.0	0.704460	0.733945	0.763430	0.801440	0.839450	0.890890	0.942330	0.971165	1.00000	0.993120	...	0.657930	0.646135	0.634340	0.624510	0.614680	0.605505
0.1	0.702775	0.731895	0.761015	0.798959	0.836902	0.888084	0.939265	0.968098	0.99693	0.989464	...	0.655102	0.643299	0.631495	0.621841	0.612187	0.603014
0.2	0.701090	0.729845	0.758600	0.796477	0.834355	0.885277	0.936200	0.965030	0.99386	0.985807	...	0.652275	0.640462	0.628650	0.619172	0.609695	0.600522
0.3	0.699405	0.727795	0.756185	0.793996	0.831808	0.882471	0.933135	0.961963	0.99079	0.982151	...	0.649447	0.637626	0.625805	0.616504	0.607202	0.598031
0.4	0.697720	0.725745	0.753770	0.791515	0.829260	0.879665	0.930070	0.958895	0.98772	0.978495	...	0.646620	0.634790	0.622960	0.613835	0.604710	0.595540
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13.6	0.275710	0.293585	0.311460	0.331055	0.350650	0.377400	0.404150	0.429105	0.45406	0.463110	...	0.305220	0.291560	0.277900	0.268390	0.258880	0.250010
13.7	0.273250	0.290865	0.308480	0.328085	0.347690	0.373646	0.399603	0.424511	0.44942	0.459305	...	0.302630	0.288956	0.275282	0.265719	0.256155	0.247290
13.8	0.270790	0.288145	0.305500	0.325115	0.344730	0.369893	0.395055	0.419918	0.44478	0.455500	...	0.300040	0.286353	0.272665	0.263047	0.253430	0.244570
13.9	0.268330	0.285425	0.302520	0.322145	0.341770	0.366139	0.390508	0.415324	0.44014	0.451695	...	0.297450	0.283749	0.270047	0.260376	0.250705	0.241850
14.0	0.265870	0.282705	0.299540	0.319175	0.338810	0.362385	0.385960	0.410730	0.43550	0.447890	...	0.294860	0.281145	0.267430	0.257705	0.247980	0.239130

Рисунок 3.3 Розширений датафрейм з всіма вимірами та назвою газу

Після обробки даних побудовуються графіки, які демонструють залежність опору від напруги та концентрації речовини.

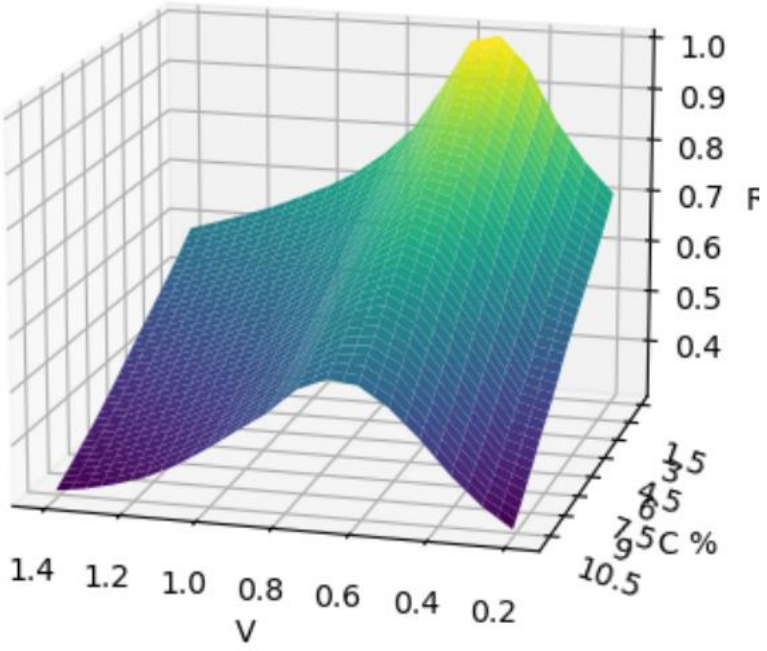

Рисунок 3.4 3D графік для аміаку

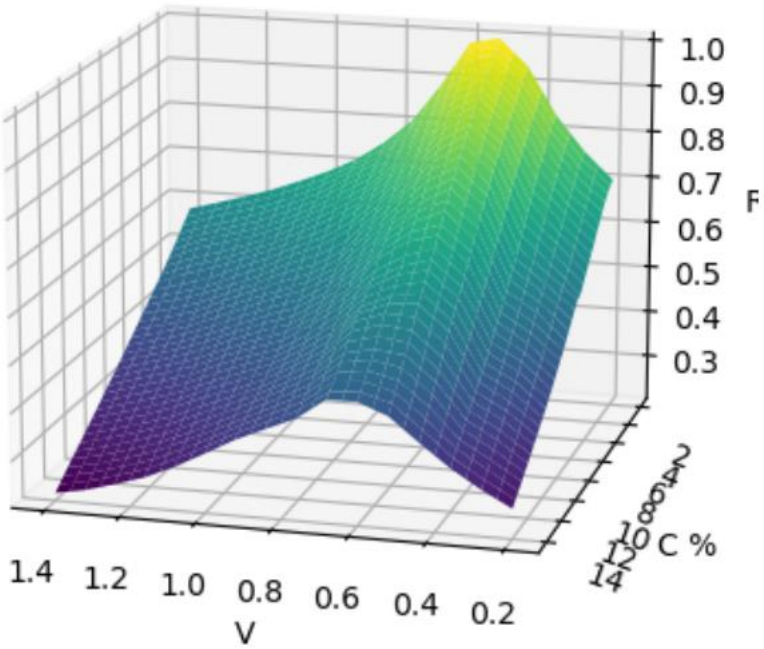

Рисунок 3.5 3D графік для етанолу

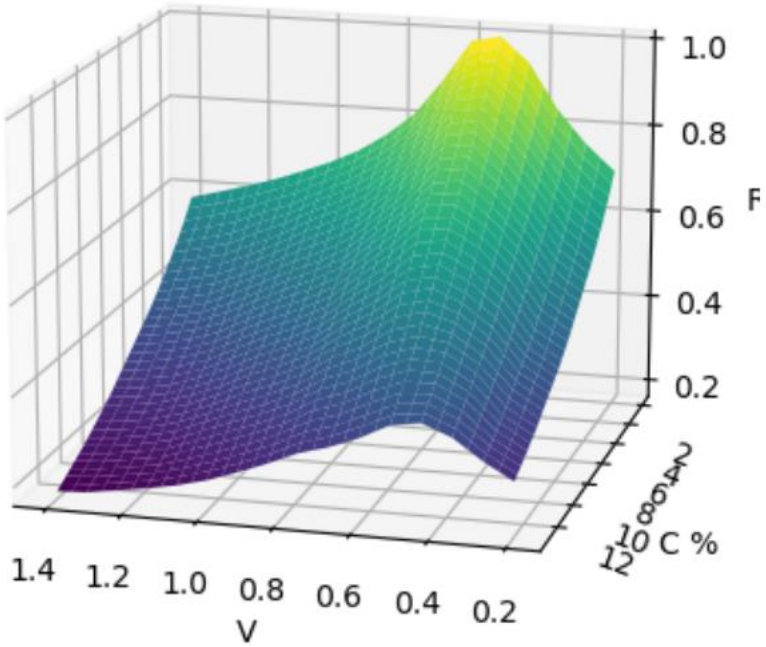

Рисунок 3.6 3D графік для ацетону

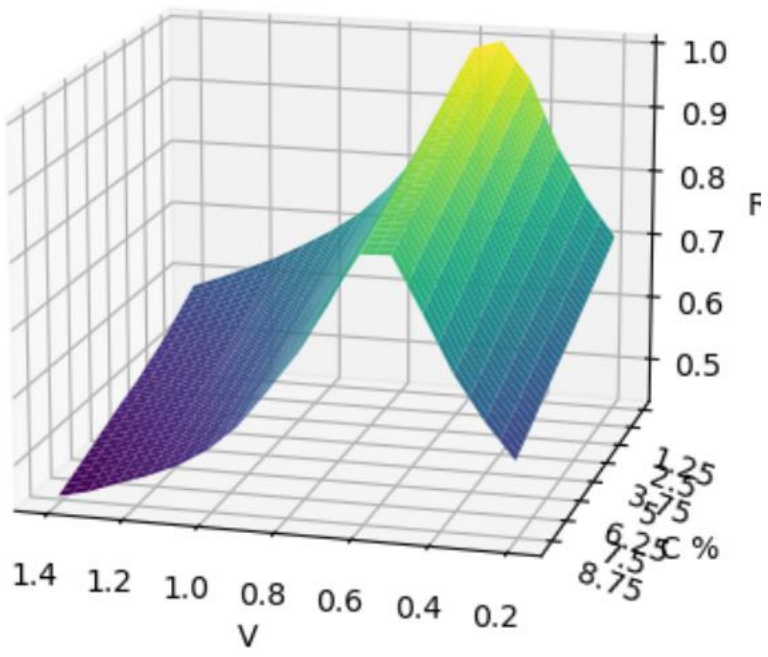

Рисунок 3.7 3D графік для толуолу

Далі дані готуються для моделей машинного навчання. DataFrame для кожної речовини об'єднується в один загальний DataFrame. Категоріальні значення типу речовини перетворюються за допомогою LabelEncoder, щоб моделі могли працювати з ними.

Потім визначаються чотири моделі машинного навчання: SVM, GradientBoostingClassifier, KNeighborsClassifier та DecisionTreeClassifier. Кожна модель навчається на тренувальних, що складають 20% від всього розміру вибірки.

В кінці, застосовуються навчені моделі до тестових. Результати класифікації оцінюються за допомогою classification_report, який надає звіт з метриками класифікації, такими як точність, відновлення та F1-показник.

У звіті про оцінку класифікатора, результати представлені за допомогою метрик precision, recall, f1-score і support. Розшифрування їх значення:

F1-показник є мірою узгодженості між точністю (precision) і відновленням (recall) в задачі класифікації. Він об'єднує ці дві метрики узгодженість, що дозволяє отримати узагальнену оцінку ефективності моделі.

F1-показник розраховується за формулою:

F1 = 2 * (precision * recall) / (precision + recall)

де precision = TP / (TP + FP) і recall = TP / (TP + FN), де

TP - кількість правильно класифікованих екземплярів позитивного класу

FP - кількість неправильно класифікованих екземплярів позитивного класу

FN - кількість неправильно класифікованих екземплярів негативного класу

F1-показник приймає значення від 0 до 1, де 1 означає ідеальну узгодженість між точністю і відновленням, а 0 означає повну відсутність узгодженості.

У випадку класифікації речовин F1-показник допомагає оцінити ефективність моделей в розпізнаванні та класифікації речовин на основі спектральних даних.

Precision: Це метрика, яка визначає точність моделі у визначенні позитивних прикладів. Для кожного класу вираховується співвідношення правильно класифікованих позитивних екземплярів до загальної кількості екземплярів, які були класифіковані як позитивні.

Recall: Ця метрика визначає, наскільки ефективно модель знаходить всі позитивні приклади. Вона представляє співвідношення правильно класифікованих позитивних екземплярів до загальної кількості позитивних екземплярів у наборі даних.

Support: Це кількість прикладів у кожному класі. Він показує, скільки екземплярів з'являються для кожного класу в наборі даних.

Поміж цих метрик, у відомостях також присутні дві інші статистики:

1. Macro avg: Це середнє значення метрик (precision, recall, f1-score) для кожного класу окремо без урахування розмірів класів. Це означає, що кожен клас має однакову вагу у розрахунках.

2. Weighted avg: Це середнє значення метрик, де вага кожного класу залежить від його розміру. Більш великі класи матимуть більший вплив на обчислення, ніж менші кл

Враховуючи, що при концентрації меншій 5% графіки речовин проявляють схожість у даних. Зроблено додаткові датасети для концентрації до 5% і для концентрації більше 5%.

```
Model SVM
              precision    recall  f1-score   support

           0       0.89      0.76      0.82        33
           1       1.00      0.72      0.84        32
           2       1.00      0.68      0.81        25
           3       0.51      1.00      0.68        23

    accuracy                           0.78       113
   macro avg       0.85      0.79      0.79       113
weighted avg       0.87      0.78      0.79       113

Actual:    [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 0, 1, 1, 1, 2, 1, 1, 3, 3, 0, 1, 1, 1, 1, 3]
Predicted: [3, 3, 1, 3, 3, 3, 1, 3, 2, 1, 0, 3, 1, 1, 2, 3, 1, 3, 3, 0, 3, 3, 1, 1, 3]
```

Рисунок 3.8 Результати для всього набору даних з використанням алгоритму SVM

```
Model GradientBoostingClassifier
              precision    recall  f1-score   support

           0       0.96      0.76      0.85        33
           1       1.00      0.94      0.97        32
           2       0.79      0.88      0.83        25
           3       0.79      1.00      0.88        23

    accuracy                           0.88       113
   macro avg       0.89      0.89      0.88       113
weighted avg       0.90      0.88      0.89       113

Actual:    [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 0, 1, 1, 1, 2, 1, 1, 3, 3, 0, 1, 1, 1, 1, 3]
Predicted: [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 0, 1, 1, 1, 2, 1, 1, 3, 3, 0, 3, 1, 1, 1, 3]
```

Рисунок 3.9 Результати для всього набору даних з використанням алгоритму Gradient Boosting

```
Model KNeighborsClassifier
              precision    recall  f1-score   support

           0       0.83      0.61      0.70        33
           1       1.00      0.91      0.95        32
           2       0.67      0.72      0.69        25
           3       0.70      1.00      0.82        23

    accuracy                           0.80       113
   macro avg       0.80      0.81      0.79       113
weighted avg       0.82      0.80      0.79       113

Actual:    [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 0, 1, 1, 1, 2, 1, 1, 3, 3, 0, 1, 1, 1, 1, 3]
Predicted: [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 2, 3, 1, 1, 2, 1, 1, 3, 3, 0, 3, 1, 1, 1, 3]
```

Рисунок 3.10 Результати для всього набору даних з використанням алгоритму K-Neighbors

```
Model DecisionTreeClassifier
              precision    recall  f1-score   support

           0       0.96      0.73      0.83        33
           1       0.82      0.84      0.83        32
           2       0.74      0.92      0.82        25
           3       0.79      0.83      0.81        23

    accuracy                           0.82       113
   macro avg       0.83      0.83      0.82       113
weighted avg       0.84      0.82      0.82       113

Actual:    [3, 3, 1, 3, 3, 2, 1, 3, 2, 1, 0, 1, 1, 1, 2, 1, 1, 3, 3, 0, 1, 1, 1, 1, 3]
Predicted: [3, 1, 1, 1, 3, 2, 1, 3, 2, 1, 0, 2, 1, 1, 2, 1, 1, 3, 3, 0, 3, 1, 1, 1, 3]
```

Рисунок 3.11 Результати для всього набору даних з використанням алгоритму Decision Tree

```
Model SVM
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        13
           1       0.56      0.50      0.53        10
           2       1.00      0.33      0.50         6
           3       0.54      1.00      0.70        21

    accuracy                           0.56        50
   macro avg       0.52      0.46      0.43        50
weighted avg       0.46      0.56      0.46        50

Actual:    [0, 0, 3, 1, 2, 3, 3, 1, 0, 3, 3, 3, 3, 3, 1, 3, 3, 0, 3, 0, 2, 0, 1, 1, 3]
Predicted: [1, 3, 3, 1, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3]
```

Рисунок 3.12 Результати для набору даних при концентрації меншій 5% з використанням алгоритму SVM

```
Model GradientBoostingClassifier
              precision    recall  f1-score   support

           0       0.90      0.69      0.78        13
           1       0.70      0.70      0.70        10
           2       0.33      0.33      0.33         6
           3       0.79      0.90      0.84        21

    accuracy                           0.74        50
   macro avg       0.68      0.66      0.67        50
weighted avg       0.75      0.74      0.74        50

Actual:    [0, 0, 3, 1, 2, 3, 3, 1, 0, 3, 3, 3, 3, 3, 1, 3, 3, 0, 3, 0, 2, 0, 1, 1, 3]
Predicted: [0, 0, 3, 1, 2, 3, 3, 1, 2, 3, 3, 3, 3, 1, 3, 3, 3, 2, 3, 0, 1, 0, 1, 3, 3]
```

Рисунок 3.13 Результати для набору даних при концентрації меншій 5% з використанням алгоритму Gradient Boosting

```
Model KNeighborsClassifier
              precision    recall  f1-score   support

           0       1.00      0.62      0.76        13
           1       1.00      1.00      1.00        10
           2       1.00      0.33      0.50         6
           3       0.70      1.00      0.82        21

    accuracy                           0.82        50
   macro avg       0.93      0.74      0.77        50
weighted avg       0.87      0.82      0.80        50

Actual:    [0, 0, 3, 1, 2, 3, 3, 1, 0, 3, 3, 3, 3, 3, 1, 3, 3, 0, 3, 0, 2, 0, 1, 1, 3]
Predicted: [0, 0, 3, 1, 2, 3, 3, 1, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 0, 3, 0, 1, 1, 3]
```

Рисунок 3.14 Результати для набору даних при концентрації меншій 5% з використанням алгоритму K-Neighbors

```
Model DecisionTreeClassifier
              precision    recall  f1-score   support

           0       0.88      0.54      0.67        13
           1       0.75      0.60      0.67        10
           2       0.33      0.33      0.33         6
           3       0.71      0.95      0.82        21

    accuracy                           0.70        50
   macro avg       0.67      0.61      0.62        50
weighted avg       0.72      0.70      0.69        50

Actual:    [0, 0, 3, 1, 2, 3, 3, 1, 0, 3, 3, 3, 3, 3, 1, 3, 3, 0, 3, 0, 2, 0, 1, 1, 3]
Predicted: [0, 0, 3, 1, 2, 3, 2, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 2, 1, 3, 3]
```

Рисунок 3.15 Результати для набору даних при концентрації меншій 5% з використанням алгоритму Decision Tree

```
Model SVM
              precision    recall  f1-score   support

           0       0.95      1.00      0.98        20
           1       1.00      1.00      1.00        16
           2       1.00      0.94      0.97        18
           3       1.00      1.00      1.00         9

    accuracy                           0.98        63
   macro avg       0.99      0.99      0.99        63
weighted avg       0.98      0.98      0.98        63

Actual:    [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
Predicted: [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
```

Рисунок 3.16 Результати для набору даних при концентрації більшій 5% з використанням алгоритму SVM

```
Model GradientBoostingClassifier
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        20
           1       0.89      1.00      0.94        16
           2       1.00      1.00      1.00        18
           3       1.00      1.00      1.00         9

    accuracy                           0.97        63
   macro avg       0.97      0.97      0.97        63
weighted avg       0.97      0.97      0.97        63

Actual:    [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
Predicted: [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 1, 0, 2, 0, 2, 2, 1, 0, 1, 1, 1, 3]
```

Рисунок 3.17 Результати для набору даних при концентрації більшій 5% з використанням алгоритму Gradient Boosting

```
Model KNeighborsClassifier
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00        16
           2       1.00      1.00      1.00        18
           3       1.00      1.00      1.00         9

    accuracy                           1.00        63
   macro avg       1.00      1.00      1.00        63
weighted avg       1.00      1.00      1.00        63

Actual:    [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
Predicted: [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
```

Рисунок 3.18 Результати для набору даних при концентрації більшій 5% з використанням алгоритму K-Neighbors

```
Model DecisionTreeClassifier
              precision    recall  f1-score   support

           0       0.94      0.85      0.89        20
           1       0.83      0.94      0.88        16
           2       1.00      1.00      1.00        18
           3       1.00      1.00      1.00         9

    accuracy                           0.94        63
   macro avg       0.94      0.95      0.94        63
weighted avg       0.94      0.94      0.94        63

Actual:    [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 0, 0, 2, 0, 2, 2, 1, 0, 1, 0, 1, 3]
Predicted: [1, 0, 2, 0, 3, 2, 0, 0, 2, 2, 3, 2, 2, 1, 0, 2, 0, 2, 2, 1, 0, 1, 1, 1, 3]
```

Рисунок 3.19 Результати для набору даних при концентрації більшій 5% з використанням алгоритму Decision Tree

ВИСНОВКИ

В роботі було досліджено принципи роботи сенсорів даних та проведено їх фільтрування, обробку та аналіз. Для аналізу використовувалися методи машинного навчання, нейронних мереж і інших алгоритмів. Окремо розглянуто основи мови Python, середовища Jupyter та використані бібліотеки. У практичній частині роботи були описані моделі машинного навчання і продемонстровано їх роботу в конкретній задачі розпізнавання газів.

Аналізуючи роботу алгоритмів, можна зробити висновок, що обидва підходи ефективно справляються зі ставленою задачею. Проте, слід зауважити, що результати можуть змінюватися при роботі з великими датасетами. Зауважимо, що навчальний та загальний датасет були відносно невеликими, тому результати на більших обсягах даних можуть бути іншими. Однак, при подальшому навчанні моделей на більшому обсязі даних можна очікувати поліпшення результатів розпізнавання.

Серед недоліків існуючого додатку можна відзначити його нездатність враховувати відносну вологість. Проте, цей аспект планується допрацювати, щоб забезпечити більш точне розпізнавання в реальних умовах зі змінною вологістю.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
2. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
3. "Neural Networks and Deep Learning" by Michael Nielsen
4. Офіційна документація Python [Електронний ресурс]: https://docs.python.org/
5. Офіційна документація Jupyter [Електронний ресурс] : https://jupyter.org/documentation
6. "Machine Learning in Chemistry" by Jan H. Jensen
7. "Chemoinformatics: Concepts, Methods, and Tools for Drug Discovery" by Edgar Jacoby
8. "DeepChem: A Comprehensive Chemical Informatics Library" by Bharath Ramsundar et al.
9. NumPy [Електронний ресурс] : https://numpy.org/
10. Pandas [Електронний ресурс] : https://pandas.pydata.org/
11. 3. Scikit-learn [Електронний ресурс] : https://scikit-learn.org/

ДОДАТКИ

Додаток 1. Код програми

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

import warnings
warnings.filterwarnings('ignore')

ethanol = pd.read_excel('data.xls', sheet_name="ethanol", header=None)
ammonia = pd.read_excel('data.xls', sheet_name="ammonia", header=None)
acetone = pd.read_excel('data.xls', sheet_name="acetone", header=None)
toluol = pd.read_excel('data.xls', sheet_name="toluol", header=None)

def transform_dataframe(dataframe):
    new_columns = dataframe.iloc[-1]
    dataframe = dataframe.rename(columns=new_columns)
    dataframe = dataframe.set_index('C, %')
    dataframe = dataframe[:-1]
    return dataframe

ethanol = transform_dataframe(ethanol)
ammonia = transform_dataframe(ammonia)
acetone = transform_dataframe(acetone)
```

```
toluol = transform_dataframe(toluol)

def imputations(dataframe):
    df_columns = list(dataframe.columns)
    for i in range(len(df_columns)):
        try:
            dataframe[round((float(df_columns[i]) + float(df_columns[i+1]))/2, 2)] = (dataframe[df_columns[i]] + dataframe[df_columns[i+1]]) / 2
        except Exception:
            pass
    sorted_columns  = sorted(dataframe.columns, key=float)
    sorted_dataframe = dataframe[sorted_columns]
    return sorted_dataframe

ethanol = imputations(ethanol)
ammonia = imputations(ammonia)
acetone = imputations(acetone)
toluol = imputations(toluol)

def build_plot(df):
    x_size = df.shape[1]
    y_size = df.shape[0]

    x = np.arange(x_size)
    y = np.arange(y_size)
    X, Y = np.meshgrid(x, y)
    Z = df.values

    x_values = df.columns
    x_labels = x_values[::4]
    y_values = df.index
    y_labels = y_values[::5]
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='viridis', edgecolor='none')
    ax.set_xticklabels(x_labels)
    ax.set_yticklabels(y_labels, rotation=-20)
    ax.set_xlabel('V')
    ax.set_ylabel('C %')
    ax.set_zlabel('R')
    ax.view_init(15, 105)
    plt.show()

build_plot(ethanol)
build_plot(ammonia)
build_plot(acetone)
build_plot(toluol)

def calculate_average_sum(df):
    result = pd.DataFrame(columns=df.columns)
    for i in range(len(df) - 1):
        row_sum = (df.iloc[i] + df.iloc[i + 1]) / 2
        row_index = round((df.index[i] + df.index[i+1]) /2, 2)
        result.loc[row_index] = row_sum
    return result

ethanol_df = calculate_average_sum(ethanol)
ethanol = pd.concat([ethanol, ethanol_df]).sort_index()
ammonia_df = calculate_average_sum(ammonia)
ammonia = pd.concat([ammonia, ammonia_df]).sort_index()
acetone_df = calculate_average_sum(acetone)
acetone = pd.concat([acetone, acetone_df]).sort_index()
toluol_df = calculate_average_sum(toluol)
```

```
toluol = pd.concat([toluol, toluol_df]).sort_index()

ethanol['Type'] = '0'
ammonia['Type'] = '1'
acetone['Type'] = '2'
toluol['Type'] = '3'

df_all = pd.concat([ethanol, ammonia, acetone, toluol])
df_split_1 = df_all.groupby(df_all.index < 5).get_group(True).reset_index(drop=True)
df_split_2 = df_all.groupby(df_all.index > 5).get_group(True).reset_index(drop=True)
df_all = df_all.reset_index(drop=True)

df = df_all.copy()
X = df.iloc[:,:-1]
y = df['Type']

le = LabelEncoder()
y = le.fit_transform(y)

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

model_list = list()
model_name = ['Model SVM','Model GradientBoostingClassifier','Model KNeighborsClassifier','Model DecisionTreeClassifier']
# Створення моделі SVM
model_svc = SVC().fit(X_train, y_train)
model_list.append(model_svc)

# Створення моделі GradientBoostingClassifier
model_grad = GradientBoostingClassifier().fit(X_train, y_train)
model_list.append(model_grad)
```

```
# Створення моделі KNeighborsClassifier
model_knc = KNeighborsClassifier(n_neighbors = 10).fit(X_train,y_train)
model_list.append(model_knc)

# Створення моделі DecisionTreeClassifier
model_dtc = DecisionTreeClassifier().fit(X_train,y_train)
model_list.append(model_dtc)

def prediction_score(model, name):
    y_pred = model.predict(X_test)

    predictions = model.predict(X_test)
    predicted_materials = le.inverse_transform(predictions)

    test = list(map(int, y_test))
    predicted_materials = list(map(int, predicted_materials))
    actual, predicted = test[:25], predicted_materials[:25]
    print(name)
    print(classification_report(y_test, y_pred))
    print(f'Actual:    {actual}', f'Predicted: {predicted}', "\n", sep="\n")

for i in range(len(model_list)):
     prediction_score(model_list[i], model_name[i])
```