

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Аналіз та прогнозування параметрів якості води за допомогою
алгоритмів K-Nearest Neighbors, Random Forest і Decision Tree.»

Студента 2 курсу групи ІП-21

Спеціальності: 121

«Інженерія програмного забезпечення»

Скрипець Ольги Олександрівни

«ПРИЙНЯВ» з оцінкою

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

Підпис

Дата

Київ - 2024 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІІ-21

Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Скрипець Ольги Олександрівни

1.Тема роботи Аналіз та прогнозування параметрів якості води за допомогою алгоритмів K-Nearest Neighbors, Random Forest і Decision Tree

2.Строк здачі студентом закінченої роботи 29.05.2024

3. Вхідні дані до роботи методичні вказівки до курсової роботи, обрані дані з сайту <https://www.kaggle.com/datasets/vanthanadevi08/water-quality-prediction>

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1.Постановка задачі

2.Аналіз предметної області

3.Розробка сховища даних

4.Інтелектуальний аналіз даних

5.Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6.Дата видачі завдання 30.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	30.03.2024	
2.	Ідентифікація зовнішніх джерел даних	15.04.2024	
3.	Пошук та аналіз літератури з обраної тематики	27.04.2024	
4.	Обґрунтування вибору методів аналізу даних	02.05.2024	
4.	Застосування та оцінка ефективності методів аналізу даних	07.05.2024	
5.	Підготовка пояснювальної записки	12.05.2024	
8.	Здача курсової роботи на перевірку	22.05.2024	
9.	Захист курсової роботи	03.06.2024	

Студент

(підпис)

Скрипець О. О.

(прізвище, ім'я, по батькові)

Керівник

(підпис)

доц. Ліхоузова Т.А

(прізвище, ім'я, по батькові)

Керівник

(підпис)

доц. Олійник Ю.О.

(прізвище, ім'я, по батькові)

"29" травня 2024 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 53 сторінок, 33 рисунки, 8 посилань.

Об'єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: дослідження та порівняння алгоритмів K-Nearest Neighbors, Random Forest та Decision Tree для аналізу та прогнозування параметрів якості води, розробка програмного забезпечення для обробки та аналізу даних, а також визначення оптимального алгоритму для прогнозування на основі отриманих результатів.

Дана курсова робота включає в себе: опис створення програмного забезпечення для інтелектуального аналізу даних, їх графічного відображення та прогнозування за допомогою різних моделей.

DATASET, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, DECISION TREE CLASSIFIER, K-NEAREST NEIGHBORS, RANDOM FOREST CLASSIFIER.

ЗМІСТ

ВСТУП.....	5
1.ПОСТАНОВКА ЗАДАЧІ	6
2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	7
3.2 Редагування даних	10
3.3 Первинний аналіз	14
3.3 Розподіл даних.....	23
4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ	25
4.1 Обґрунтування вибору методів інтелектуального аналізу даних	25
4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors.....	26
4.3 Аналіз отриманих результатів для методу Random Forest Classifier	31
4.4 Аналіз отриманих результатів для методу Decision Tree Classifier .	36
ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ	43

ВСТУП

Якість води є критично важливою складовою здоров'я людини та екологічного благополуччя. Забруднення водних ресурсів та зміни у їх складі можуть мати серйозні наслідки для здоров'я населення та стану навколишнього середовища. Сучасні технології дозволяють здійснювати аналіз і моніторинг параметрів якості води з високою точністю, що сприяє своєчасному виявленню проблем і запобіганню негативним наслідкам. Саме тому задачі аналізу та прогнозування параметрів якості води є надзвичайно важливими для забезпечення стабільного та безпечного водопостачання.

У рамках даної курсової роботи проаналізовано дані про якість води з різних джерел, які мають різні характеристики. Використовуючи ці дані, застосовано три методи машинного навчання для прогнозування параметрів якості води: K-Nearest Neighbors, Random Forest та Decision Tree. Метою даного дослідження є порівняння ефективності цих алгоритмів для визначення оптимального методу прогнозування.

У даній курсовій роботі використано такі технології: Python, Pandas, Matplotlib, Sklearn, Seaborn, NumPy.

1. ПОСТАНОВКА ЗАДАЧІ

Виконання курсової роботи передбачає виконання наступних задач: аналіз предметної області, завантаження, опис та обробка даних, первинний аналіз даних; поділ даних для навчання моделі, вибір методів для прогнозування, аналіз та порівняння результатів кожного методу.

Прогнозування виконується за допомогою методів K-Nearest Neighbors, Decision Tree Classifier та Random Forest Classifier – усі ці методи призначені для вирішення задач класифікації. Потрібно проаналізувати та порівняти результати виконання кожного методу, що дозволить обрати найбільш ефективний.

Вхідними даними є рН, залізо, нітрати, хлорид, колір, каламутність, фтор, мідь, запах, сульфат, провідність, хлор, марганець та загальна кількість розчинених твердих речовин.

Додаток можна використовувати для прогнозування якості води залежно від різних показників.

Для реалізації застосунку використовується мова програмування Python. Курсовий проект має бути зданий до дедлайну 29.05.2024 включно та виконаний у єдиному стилі написання коду.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Якість води є однією з найважливіших складових здоров'я людей та екологічного благополуччя. Вода використовується в різних сферах життя: для пиття, приготування їжі, побутових потреб та промислових процесів. Загальноприйнятою нормою води, якої достатньо для підтримки нормального обміну речовин, визнаний 1 л води на 30 кг ваги для кожної людини кожен день. Оскільки вода є основним елементом, необхідним для життя, її якість має безпосередній вплив на здоров'я людей. Забруднена або неякісна вода може спричинити серйозні захворювання і навіть епідемії.

На якість води впливають численні фактори, такі як рН, вміст заліза, нітратів, хлоридів, колір, каламутність, фтор, мідь, запах, сульфати, провідність, хлор, марганець та загальна кількість розчинених твердих речовин. У цьому дослідженні ми будемо аналізувати ці параметри для прогнозування якості води.

Цей аналіз може бути корисним для різних сторін:

1. Для водопостачальних підприємств, адже розуміння основних факторів, що впливають на якість води, дозволить їм поліпшити процеси очищення та забезпечити споживачів якісною водою. Це також допоможе оптимізувати витрати на виробництво та постачання води.

2. Для контролюючих органів та наглядових установ, бо прогнозування якості води може бути важливим для моніторингу та дотримання стандартів якості води. Це сприятиме забезпеченню нормативних вимог і захисту здоров'я населення.

3. Для споживачів інформація про якість води допоможе робити обізнані вибори при використанні водних ресурсів. Знання про якість води забезпечить споживачам впевненість у безпеці та придатності води для пиття і побутових потреб.

4. Для дослідників та екологів аналіз якості води є важливим для вивчення екологічного стану водних ресурсів та розробки стратегій їх збереження та поліпшення.

У програмному забезпеченні буде реалізовано наступну функціональність:

- завантаження вибірки даних;
- обробка та дослідження завантажених даних;
- інтелектуальний аналіз даних;
- використання трьох моделей прогнозування даних;
- прогнозування якості води;
- аналіз факторів впливу на якість води;
- візуалізація отриманих результатів та їх аналіз.

Це дослідження дозволить визначити найбільш ефективний алгоритм для прогнозування якості води, що є важливим для забезпечення її безпеки та придатності для використання.

3. РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Було обрано набір даних під назвою «Water Quality Prediction», що складається з 1.05m проб води. Датасет містить 24 колонки: Index, pH, Iron, Nitrate, Chloride, Lead, Zinc, Color, Turbidity, Fluoride, Copper, Odor, Sulfate, Conductivity, Chlorine, Manganese, Total Dissolved Solids, Source, Water Temperature, Air Temperature, Month, Day, Time of Day, Target

Стовпці несуть дану інформацію:

- Index – індекс проби води.
- pH – показник кислотності води, значення якого коливаються від 2.06 до 12.9.
- Iron – вміст заліза у воді, вимірюється в міліграмах на літр (мг/л) від 0 до 15.7.
- Nitrate – вміст нітратів у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 0.29 до 73.1.
- Chloride – вміст хлоридів у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 29.4 до 1.43к.
- Lead – вміст свинцю у воді, вимірюється в мікрограмах на літр (мкг/л) коливаються від 0 до 3.5.
- Zinc – вміст цинку у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 0 до 28.4.
- Color – колір води, має різні значення, наприклад colorless або near colorless.
- Turbidity – каламутність води, вимірюється в одиницях NTU (Nephelometric Turbidity Units) коливаються від 0 до 19.3.
- Fluoride – вміст фтору у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 0 до 12.9.
- Copper – вміст міді у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 0 до 11.4.

- Odor – запах води, який є булевим параметром (1 – гарний, 0 – поганий) коливаються від 0.01 до 41.4.
- Sulfate – вміст сульфатів у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 11.9 до 1.39k.
- Conductivity – провідність води, вимірюється в мікросіменсах на сантиметр ($\mu\text{S}/\text{cm}$) коливаються від 13.1 до 1.89k.
- Chlorine – вміст хлору у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 1 до 10.7.
- Manganese – вміст марганцю у воді, вимірюється в міліграмах на літр (мг/л) коливаються від 0 до 23.7.
- Total Dissolved Solids (TDS) – загальна кількість розчинених твердих речовин, вимірюється в міліграмах на літр (мг/л) коливаються від 0.01 до 580.
- Source – джерело води (наприклад, Well, Aquifer, Stream, Ground, River, Spring).
- Water Temperature – температура води, вимірюється в градусах Цельсія ($^{\circ}\text{C}$) коливаються від 0.67 до 243.
- Air Temperature – температура повітря, вимірюється в градусах Цельсія ($^{\circ}\text{C}$) коливаються від -33.9 до 144.
- Month – місяць збору проби.
- Day – день збору проби (від 1 до 31).
- Time of Day – час доби, коли була взята проба (від 0 до 23).
- Target – цільовий параметр, який визначає якість води (1 – якісна, 0 – неякісна).

3.2 Редагування даних

Зчитаємо дані з файлу в датафрейм, виведемо перші 5 рядків датафрейму, щоб перевірити коректність імпорту даних (рис. 3.1) про нього, таку можливість нам надає Python бібліотека pandas[1].

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/My Drive/data/water_quality_prediction.csv"
df = pd.read_csv(path)
df.head()
```

/content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

Index	Nitrate	Chloride	Lead	Zinc	Color	Turbidity	Fluoride	...	Chlorine	Manganese	Dissolved Solids
83	8.605777	122.799772	3.710000e-52	3.434827	Colorless	0.022683	0.607283	...	3.708178	2.270000e-15	33
81	3.734167	227.029851	7.850000e-94	1.245317	Faint Yellow	0.019007	0.622874	...	3.292038	8.020000e-07	28
06	3.816994	230.995630	5.290000e-76	0.528280	Light Yellow	0.319956	0.423423	...	3.560224	7.007989e-02	57
88	8.224944	178.129940	4.000000e-176	4.027879	Near Colorless	0.166319	0.208454	...	3.516907	2.468295e-02	10
67	9.925788	186.540872	4.170000e-132	3.807511	Light Yellow	0.004867	0.222912	...	3.177849	3.296139e-03	16

Рисунок 3.1 – Імпорт даних.

Для аналізу ми не будемо використовувати всі стовпці, тому приберемо 'Index', 'Color', 'Source','Month','Day','Time of Day','Conductivity','Water Temperature','Air Temperature' та виведемо загальну інформацію щоб оцінити коректність даних (рис. 3.2).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 15 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   pH                                    1028344 non-null float64
1   Iron                                 1041584 non-null float64
2   Nitrate                             1029880 non-null float64
3   Chloride                             1017741 non-null float64
4   Lead                                 1043891 non-null float64
5   Zinc                                 1020900 non-null float64
6   Turbidity                            1039881 non-null float64
7   Fluoride                             1015357 non-null float64
8   Copper                               1013693 non-null float64
9   Odor                                 1017243 non-null float64
10  Sulfate                              1014050 non-null float64
11  Chlorine                             1038413 non-null float64
12  Manganese                            1029236 non-null float64
13  Total Dissolved Solids                1048277 non-null float64
14  Target                               1048575 non-null int64
dtypes: float64(14), int64(1)
memory usage: 120.0 MB
```

Рисунок 3.2 – Загальна інформація.

На даному етапі можна помітити, що дані було зчитано коректно, але є пропуски в деяких стовпцях. З інформації зрозуміло, що набір даних містить 14 колонок типу `float64` та одну колонку типу `int64`, яка виступає таргетом. Отже, всі предиктори мають числові типи даних, тому нам не потрібно кодувати дані.

Також можна побачити, що дані мають 1,048,575 записів, але не всі колонки мають однакову кількість non-null значень, що свідчить про наявність пропусків.

Для обробки пропусків у нашому датасеті можна застосувати два підходи: заповнення пропусків медіанними значеннями або видалення стовпців з великою кількістю пропусків. Заповнення пропусків медіанними значеннями є доречним, коли кількість пропусків невелика відносно загальної кількості даних, а видалення стовпців – коли кількість пропусків значна.

Почнемо з аналізу кількості пропусків у кожному стовпці, а потім застосуємо відповідні методи обробки (рис. 3.3).

```
# Виведення кількості пропусків у кожному стовпці
missing_values = df.isnull().sum()
print(missing_values)

# Відсоток пропусків у кожному стовпці
missing_percentage = (missing_values / len(df)) * 100
print(missing_percentage)
```

pH	20231
Iron	6991
Nitrate	18695
Chloride	30834
Lead	4684
Zinc	27675
Turbidity	8694
Fluoride	33218
Copper	34882
Odor	31332
Sulfate	34525
Chlorine	10162
Manganese	19339
Total Dissolved Solids	298
Target	0
dtype: int64	
pH	1.929380
Iron	0.666714
Nitrate	1.782896
Chloride	2.940562
Lead	0.446701
Zinc	2.639296
Turbidity	0.829125
Fluoride	3.167918
Copper	3.326610
Odor	2.988055
Sulfate	3.292564
Chlorine	0.969125
Manganese	1.844313
Total Dissolved Solids	0.028420
Target	0.000000
dtype: float64	

Рисунок 3.3 – Відсоток пропусків.

Ми отримали кількість пропусків у кожному стовпці датасету, а також відсоток пропусків відносно загальної кількості записів. Загалом цей відсоток не переважає 3%, тому замінимо відсутні значення медіаною та виведемо результат (рис. 3.4).

```
[14] # Заповнення пропусків медіанними значеннями в решті стовпців
data = df.fillna(df.median())

# Перевірка результатів
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 15 columns):
#   Column                                Non-Null Count  Dtype  
---  -
0   pH                                     1048575 non-null float64
1   Iron                                  1048575 non-null float64
2   Nitrate                              1048575 non-null float64
3   Chloride                             1048575 non-null float64
4   Lead                                 1048575 non-null float64
5   Zinc                                 1048575 non-null float64
6   Turbidity                            1048575 non-null float64
7   Fluoride                             1048575 non-null float64
8   Copper                               1048575 non-null float64
9   Odor                                 1048575 non-null float64
10  Sulfate                              1048575 non-null float64
11  Chlorine                             1048575 non-null float64
12  Manganese                            1048575 non-null float64
13  Total Dissolved Solids                1048575 non-null float64
14  Target                               1048575 non-null int64  
dtypes: float64(14), int64(1)
memory usage: 120.0 MB
None
```

Рисунок 3.4 – Результат заповнення пропусків.

Після виявлення та обробки пропущених значень у нашому датасеті, ми успішно заповнили відсутні дані медіанними значеннями для відповідних стовпців. Тепер наш датасет готовий для подальшого аналізу та моделювання. Ця процедура дозволить нам отримати більш точні та достовірні результати при вивченні якості води та прогнозуванні її параметрів. Заповнення пропусків медіанними значеннями допомагає зберегти цінність даних та забезпечити нам надійну основу для подальшого аналізу.

3.3 Первинний аналіз

Зараз ми проведемо первинний аналіз даних, щоб отримати уявлення про розподіл кожного з параметрів води. Для цього ми побудуємо графіки, які дозволять нам оцінити форму розподілу кожного параметра, виявити викиди та визначити, чи потрібна додаткова обробка даних перед подальшим аналізом. Цей етап дозволить нам зрозуміти особливості наших даних та підготувати їх для

подальшого моделювання та прогнозування. Код для побудови гістограм розподілу параметрів (рис. 3.5). Для відображення графіків використовуємо бібліотеки `matplotlib[2]` та `seaborn[3]`.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Визначення стовпців для аналізу
columns_to_analyze = list(data.columns)
columns_to_analyze.remove('Target') # Вилучаємо таргет для аналізу

# Побудова графіків
for col in columns_to_analyze:
    # Побудова графіків для кожного таргету окремо
    g = sns.FacetGrid(data, col="Target", hue="Target", palette="Set1", height=5)
    g.map(sns.histplot, col, bins=20, kde=True)
    g.set_axis_labels(col, "Frequency")
    g.add_legend()
    plt.show()
```

Рисунок 3.5 – Код для побудови гістограм розподілу параметрів.

Першим параметр є рН середовище (рис. 3.6), з розподілу якого видно, що не якісна вода та якісна коливається в межах від 4 до 10. Візуально також не дуже зрозуміло, чи цей параметр є дуже гарним для класифікації якості саме у нашій вибірці. Ідеальний діапазон рН для муніципальної питної води знаходиться від 6.5 до 8.5. Вода з рН в цьому діапазоні вважається найбільш придатною для споживання. Якщо вода має значення рН поза цим діапазоном, це може вказувати на проблеми з якістю води, такі як кислотність або лужність. Тому важливо враховувати інші параметри разом з рН при класифікації якості води. Тому важливо враховувати інші параметри разом з рН при класифікації якості води.

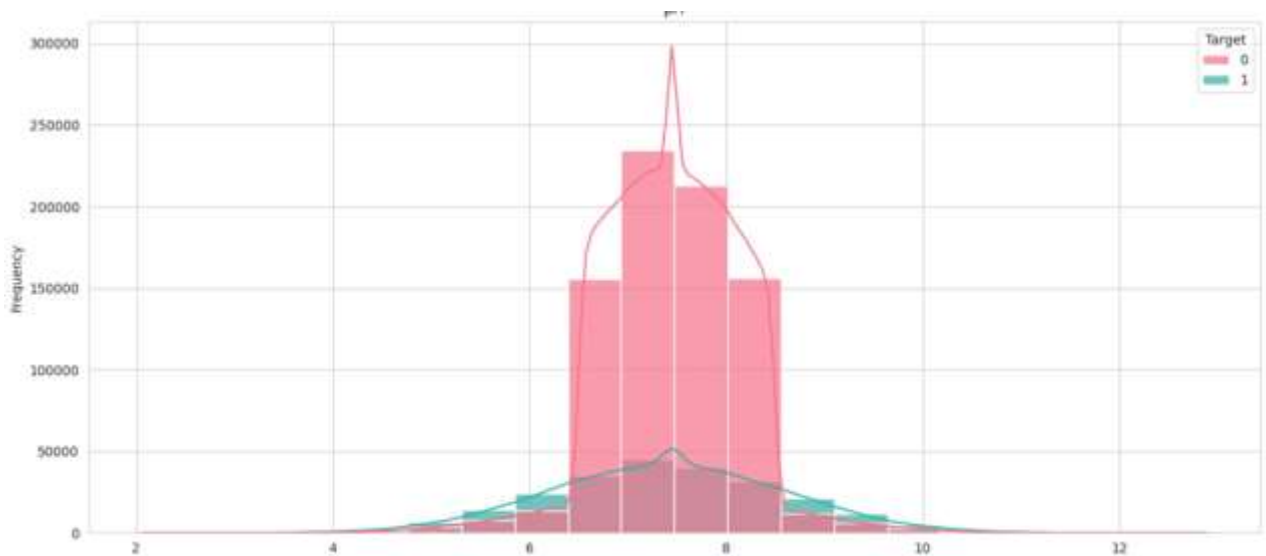


Рисунок 3.6 – Гістограма розподілу pH води.

Другий параметр це залізо, з гістограми видно (рис. 3.7), що вода низької та високої якості майже не відрізняється за вмістом заліза. Проте, важливо враховувати, що загальна концентрація заліза в воді зазвичай не повинна перевищувати 0.3 мг/літр. Це рекомендована межа для забезпечення безпеки споживання води. Якщо вода містить більше заліза, це може вплинути на смак, але небезпеки для здоров'я не виникають. Таким чином, важливо враховувати інші параметри разом з вмістом заліза при класифікації якості води.

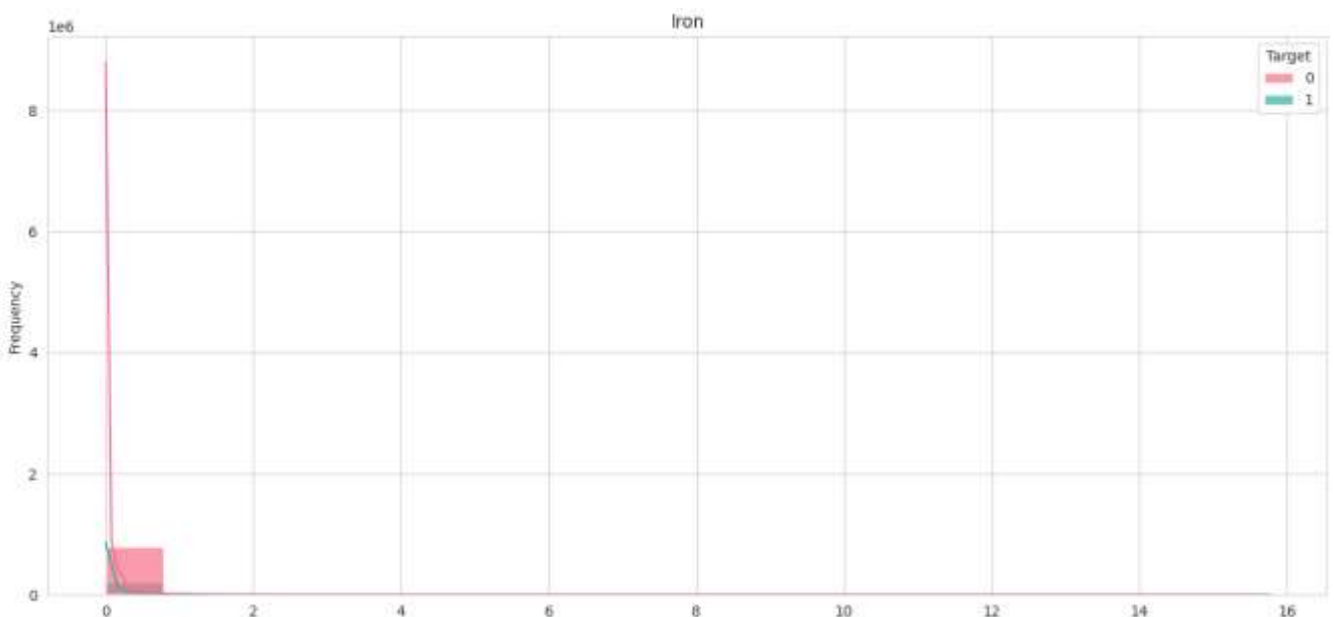


Рисунок 3.7 – Гістограма розподілу заліза у воді.

Третій параметр це нітрати (рис. 3.8) на гістограмах видно, що краща вода має менший вміст нітратів, що може бути зв'язано з меншою забрудненістю, тому це хороший показник для подальшого аналізу.

Згідно з рекомендаціями Всесвітньої організації охорони здоров'я, максимально допустимий рівень нітратів у питній воді становить 50 мг/літр. Це обмеження сприяє забезпеченню безпеки для споживачів.

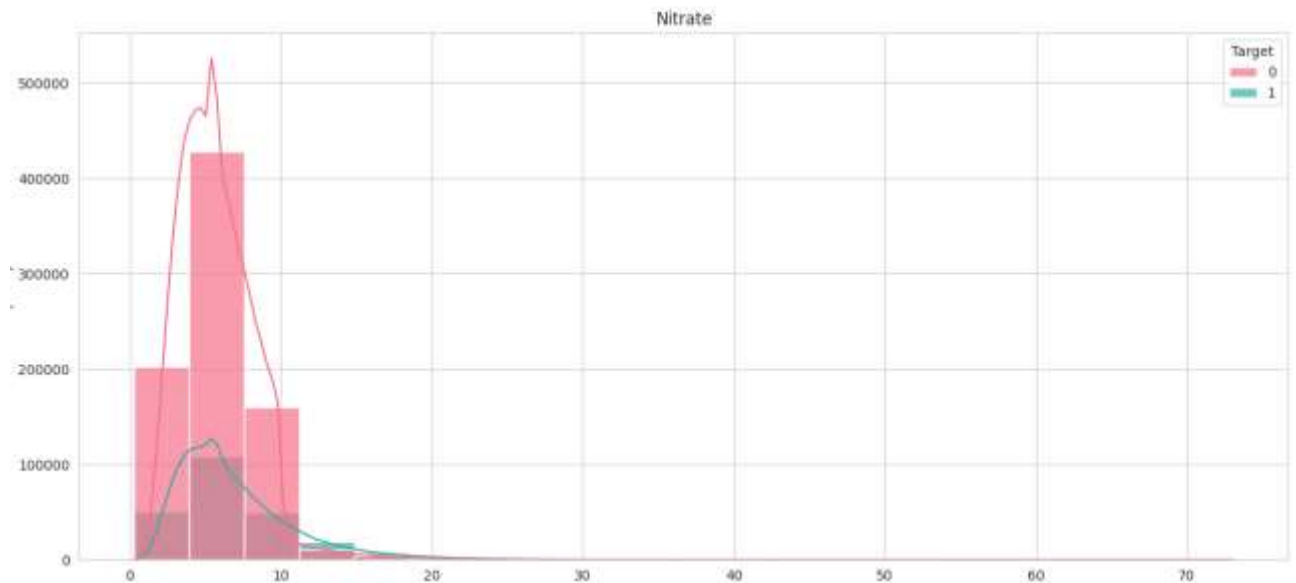


Рисунок 3.8 – Гістограма розподілу нітратів у воді.

Четвертий параметр це свинець, на гістограмі (рис. 3.9) видно, що чим менше свинцю, тим якісніша вода, але цей параметр. Звідси видно, що вміст свинцю в воді є важливим показником. Вода високої якості має менший вміст свинцю, що може бути зв'язано з меншою забрудненістю.

Згідно з рекомендаціями Всесвітньої організації охорони здоров'я, максимально допустимий рівень свинцю у питній воді становить 10 мкг/літр. Це обмеження сприяє забезпеченню безпеки споживання води. Якщо вміст свинцю перевищує цей рівень, це може вплинути на здоров'я, особливо у дітей.

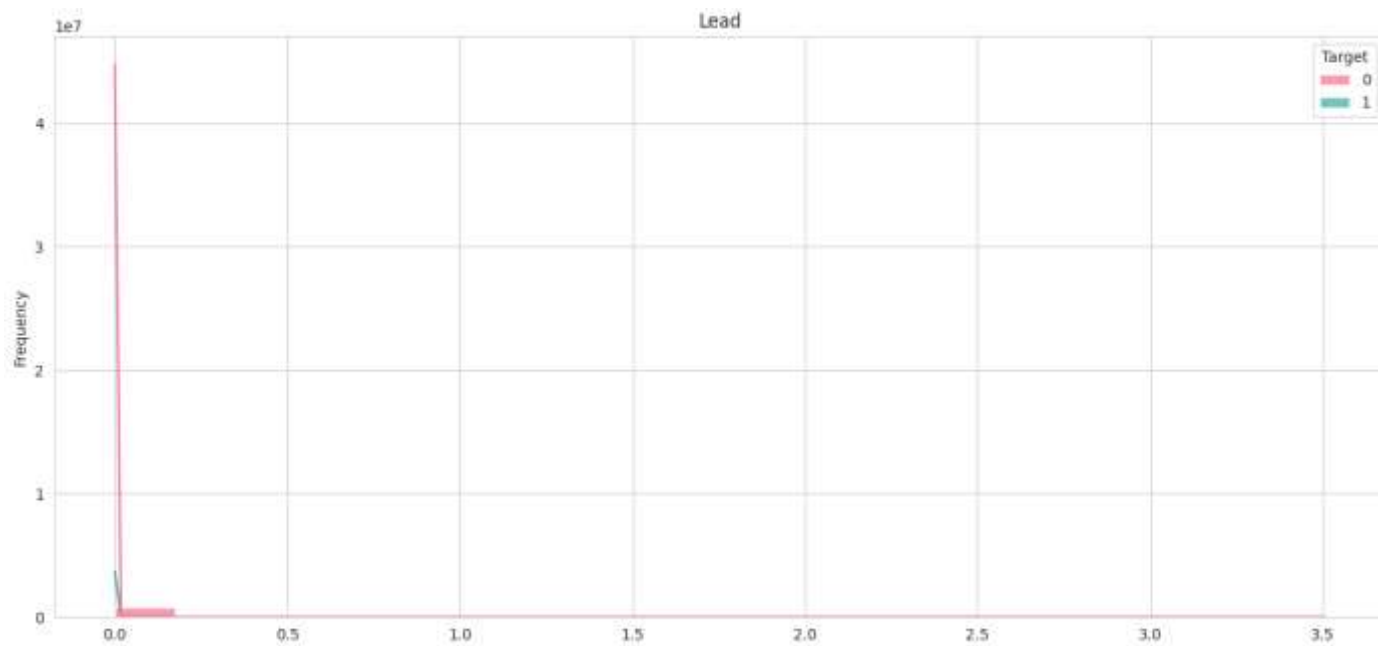
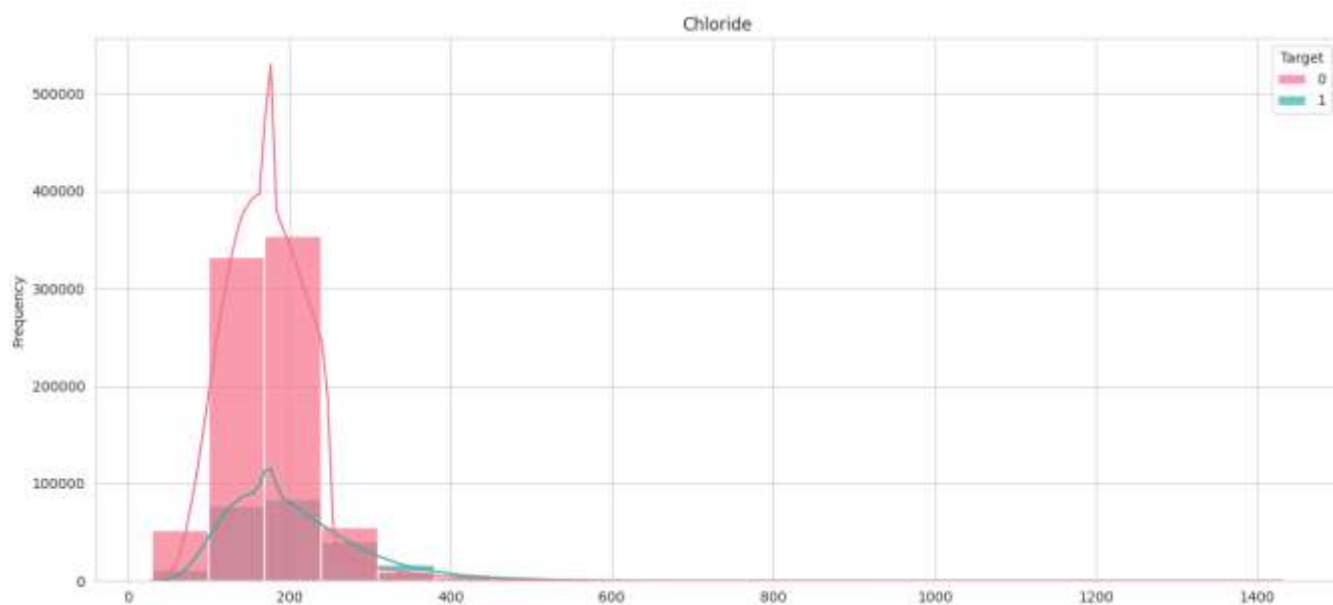


Рисунок 3.9 – Гістограма розподілу свинцю у воді

П'ятий параметр це хлорид, на гістограмі бачимо (рис. 3.10), що не дуже зрозуміло, чи цей параметр є дуже гарним для класифікації якості саме у нашій



вибірці.

Рисунок 3.10 – Гістограма розподілу свинцю у воді.

Шостий параметр це цинк, на гістограмі бачимо (рис. 3.10), що чим меншим є цей параметр, тим якіснішою є вода, ця різниця маленька, але вона є.

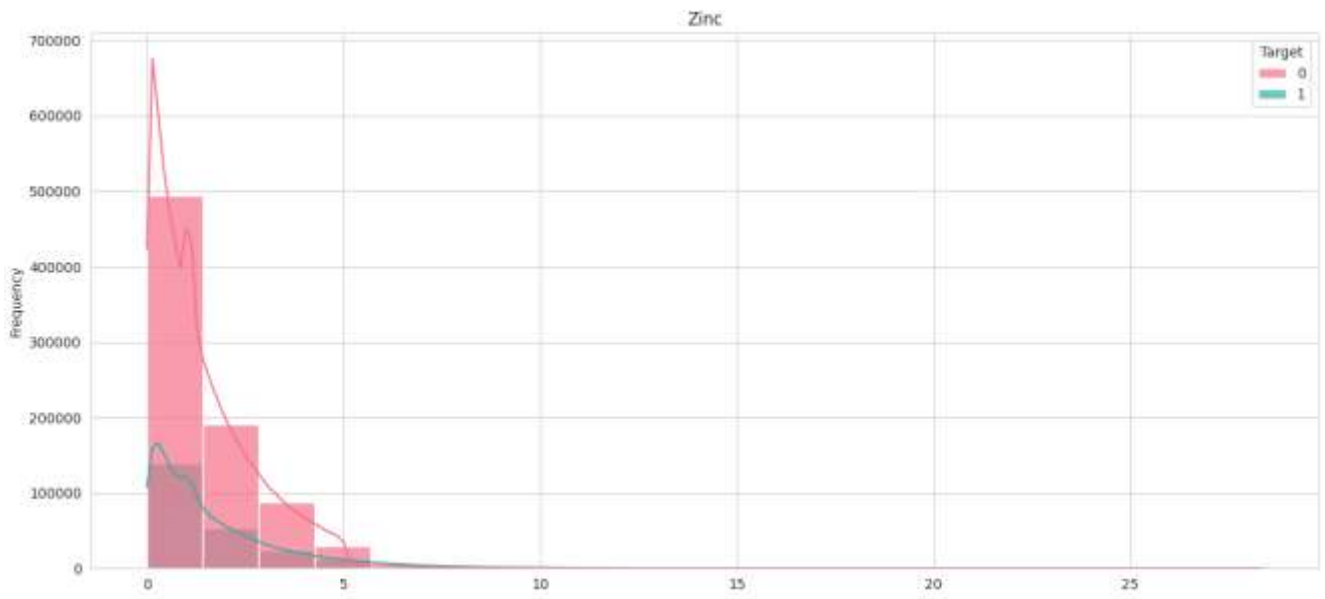


Рисунок 3.11 – Гістограма розподілу цинку у воді.

Сьомий параметр це каламутність води, на гістограмі (рис. 3.12) ми бачимо кращу залежність і будемо використовувати цей показник у подальшому аналізі.

Чим менша каламутність води, тим вона якісніша.

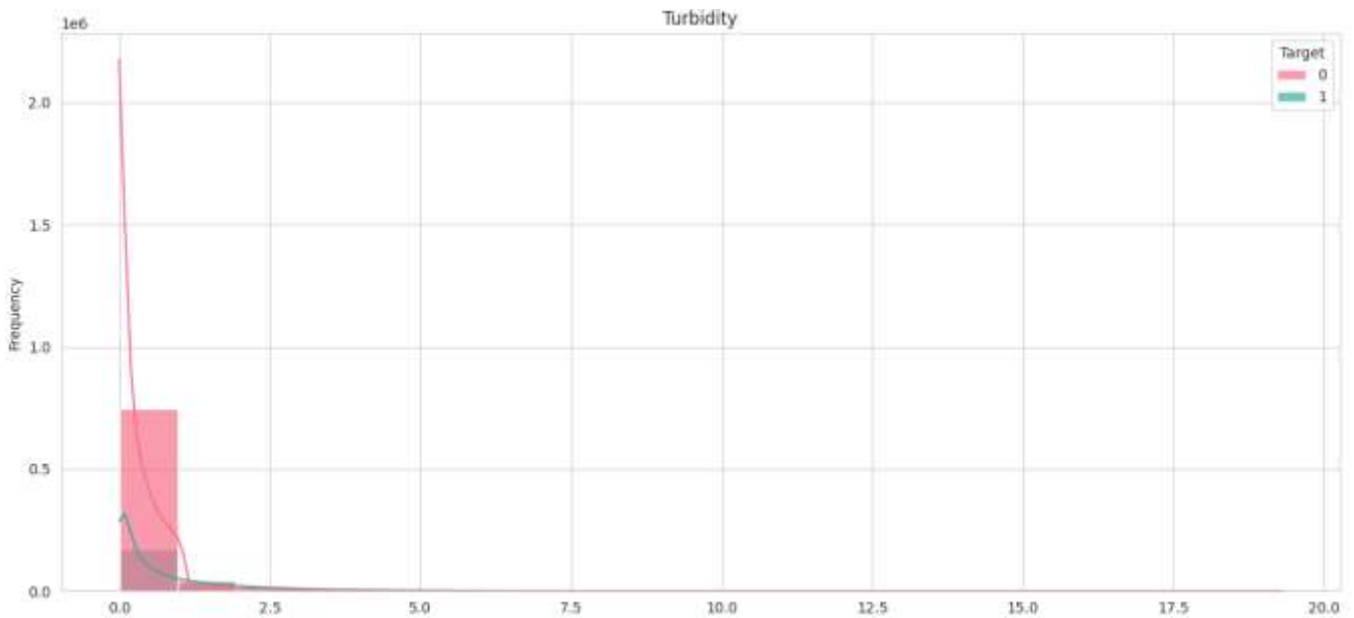


Рисунок 3.12 – Гістограма розподілу каламутності води.

Восьмий параметр це фтор, на гістограмі (рис. 3.13) ми бачимо кращу залежність і будемо використовувати цей показник у подальшому аналізі. Чим менша кількість фтору у воді води, тим вона якісніша.

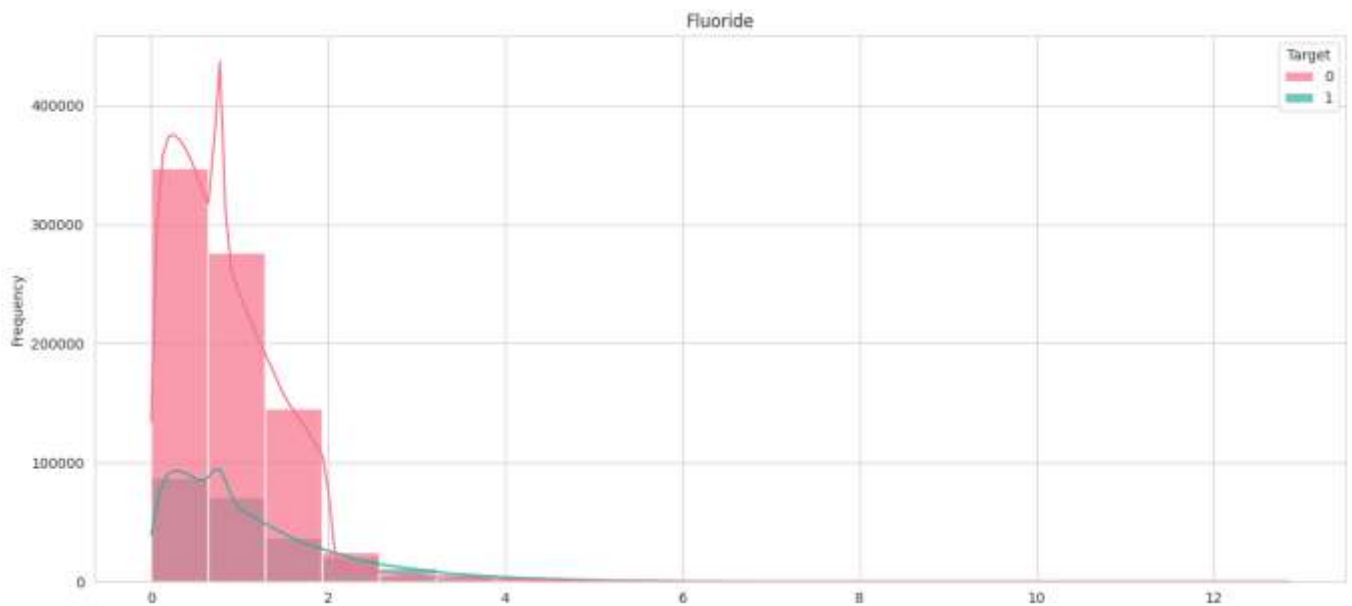


Рисунок 3.13 – Гістограма розподілу фтору у воді.

Дев'ятий параметр це мідь, на гістограмі (рис. 3.14) ми бачимо залежність і будемо використовувати цей показник у подальшому аналізі. Чим менша кількість фтору у воді води, тим вона якісніша.

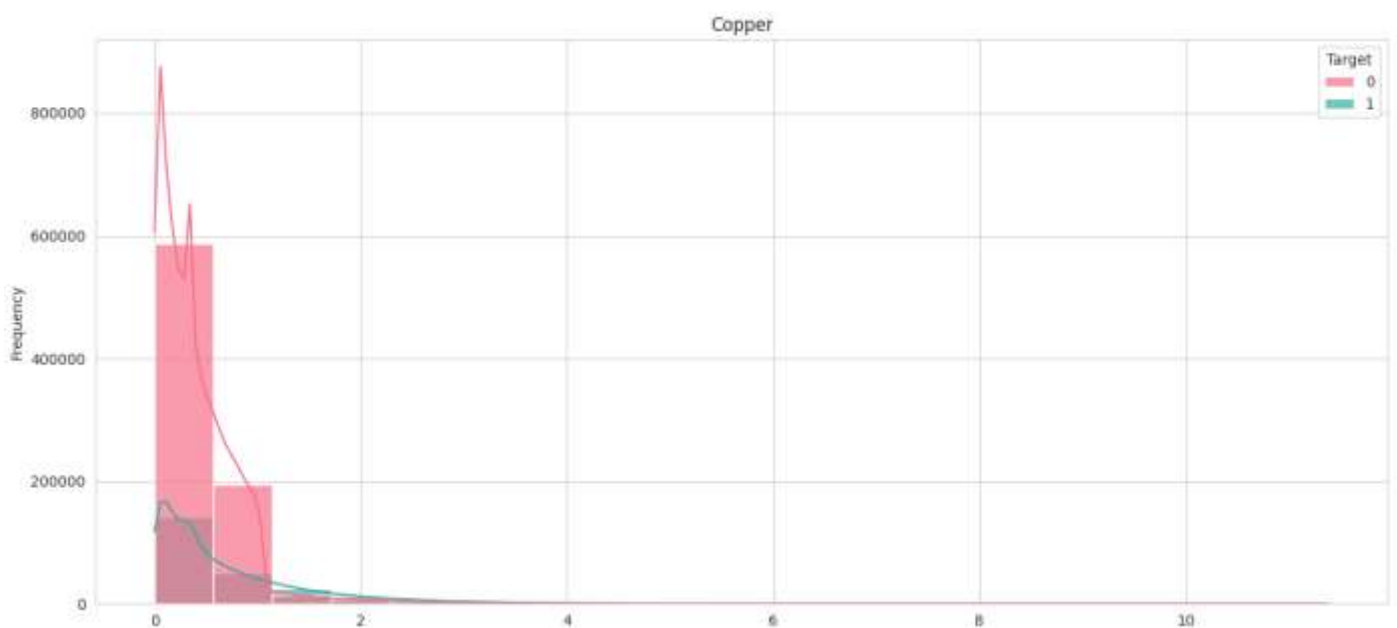


Рисунок 3.14 – Гістограма розподілу міді у воді.

Десятий параметр це запах, на гістограмі (рис. 3.15) ми бачимо залежність і будемо використовувати цей показник у подальшому аналізі. Тут 4 – це гарний запах, 0 - поганий. Тут видно, що найбільше якісної води на позначці 4.

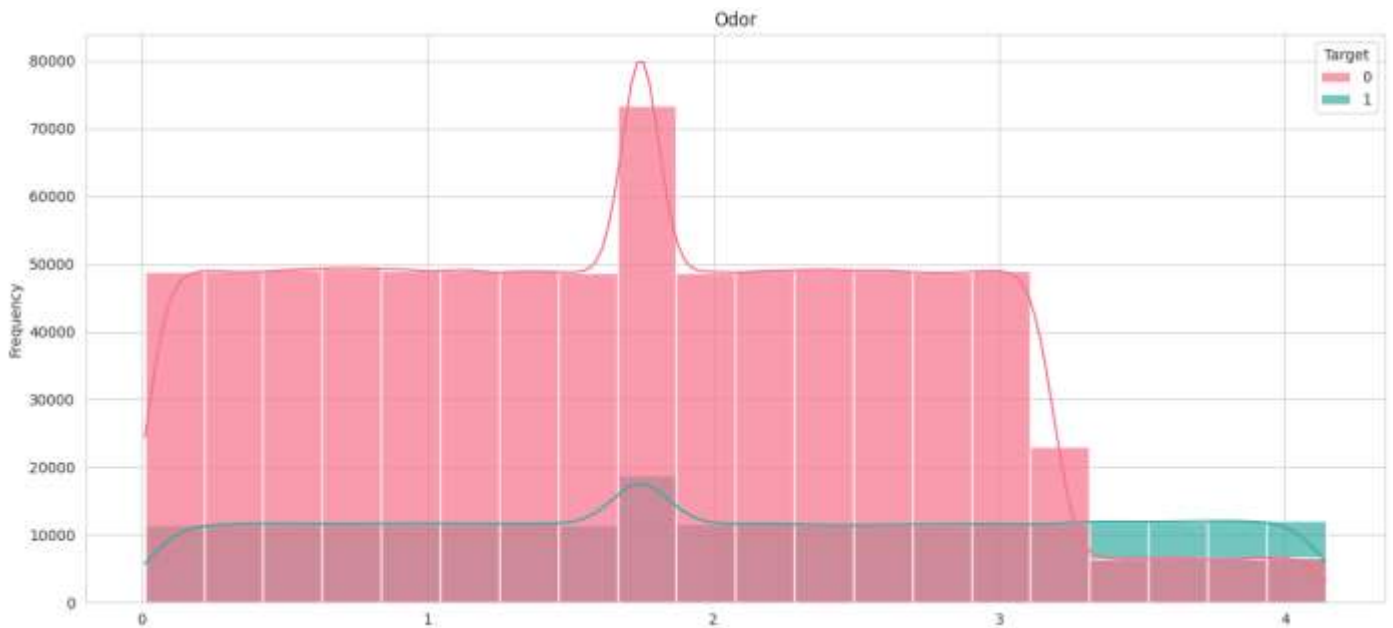


Рисунок 3.15 – Гістограма розподілу запаху води.

Одинадцятий параметр це сульфати, на гістограмі (рис. 3.16) ми бачимо незначну залежність. Чим менша кількість сульфатів у воді, тим вона якісніша.

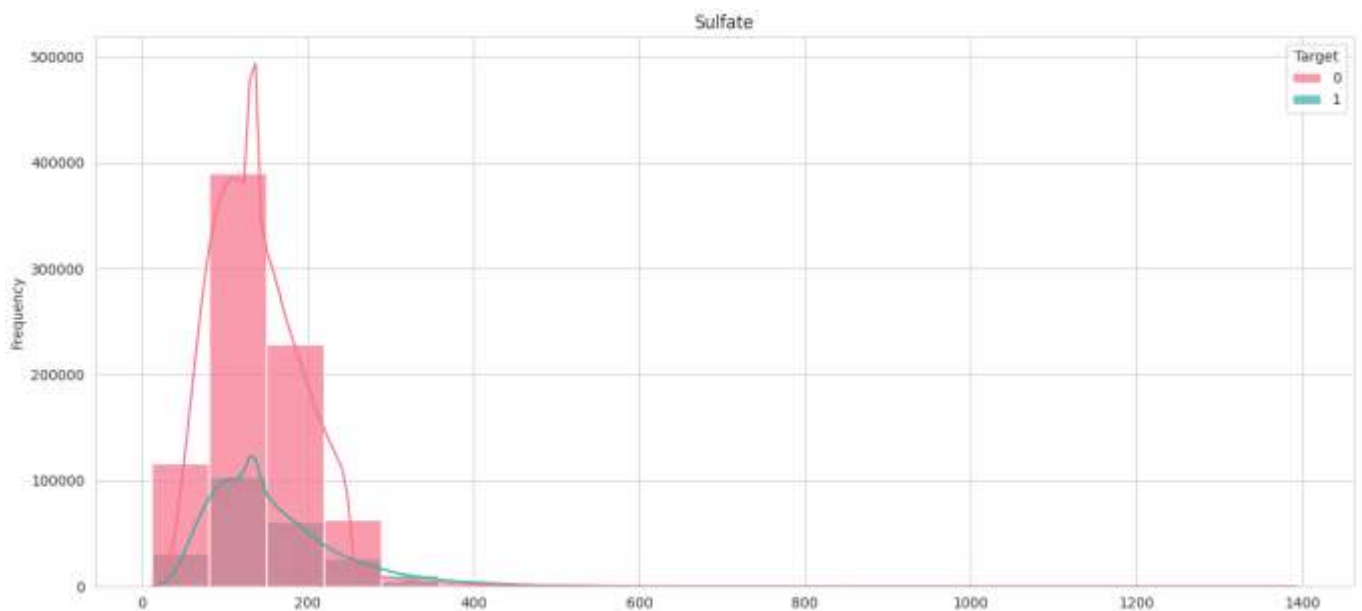


Рисунок 3.16 – Гістограма розподілу сульфатів у воді.

Дванадцятий параметр це хлор, на гістограмі (рис. 3.17) ми бачимо незначну залежність, хлор просто має знаходитись у межах норми, якщо вода якісна.

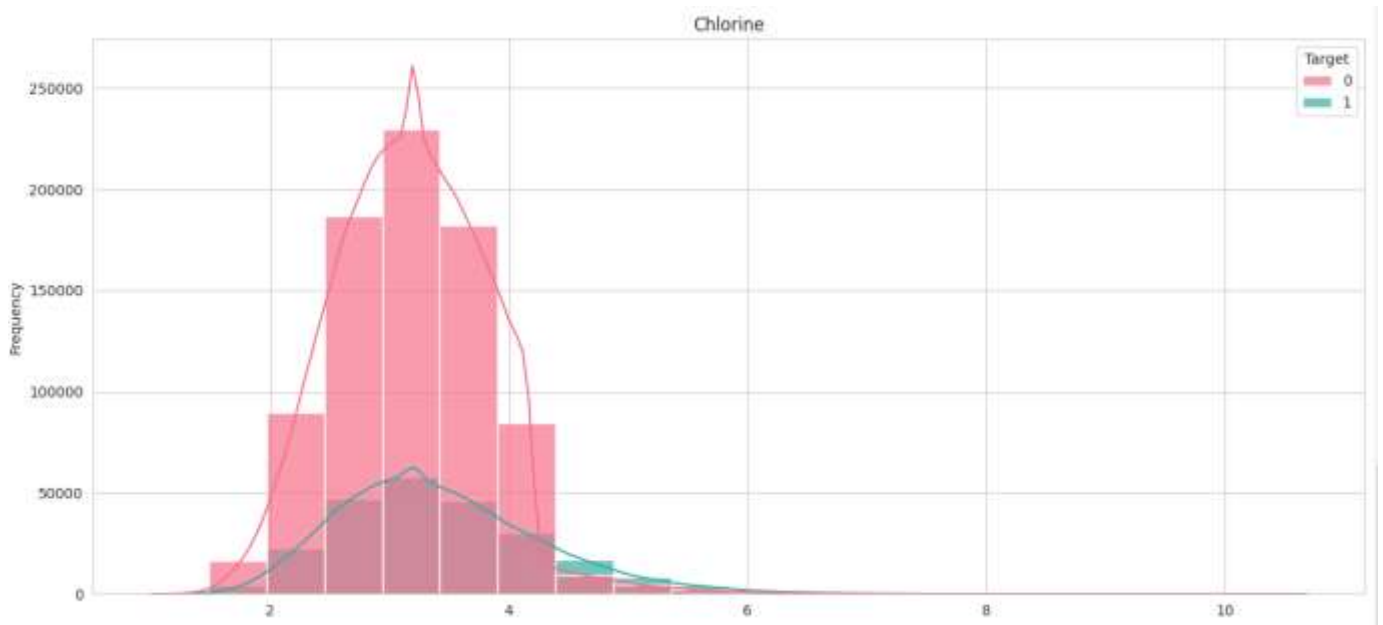


Рисунок 3.17 – Гістограма розподілу хлору у воді.

Тринадцятий параметр це вміст марганцю у воді, на гістограмі (рис. 3.18) ми бачимо незначну залежність. Чим менша кількість марганцю у воді, тим вона якісніша.

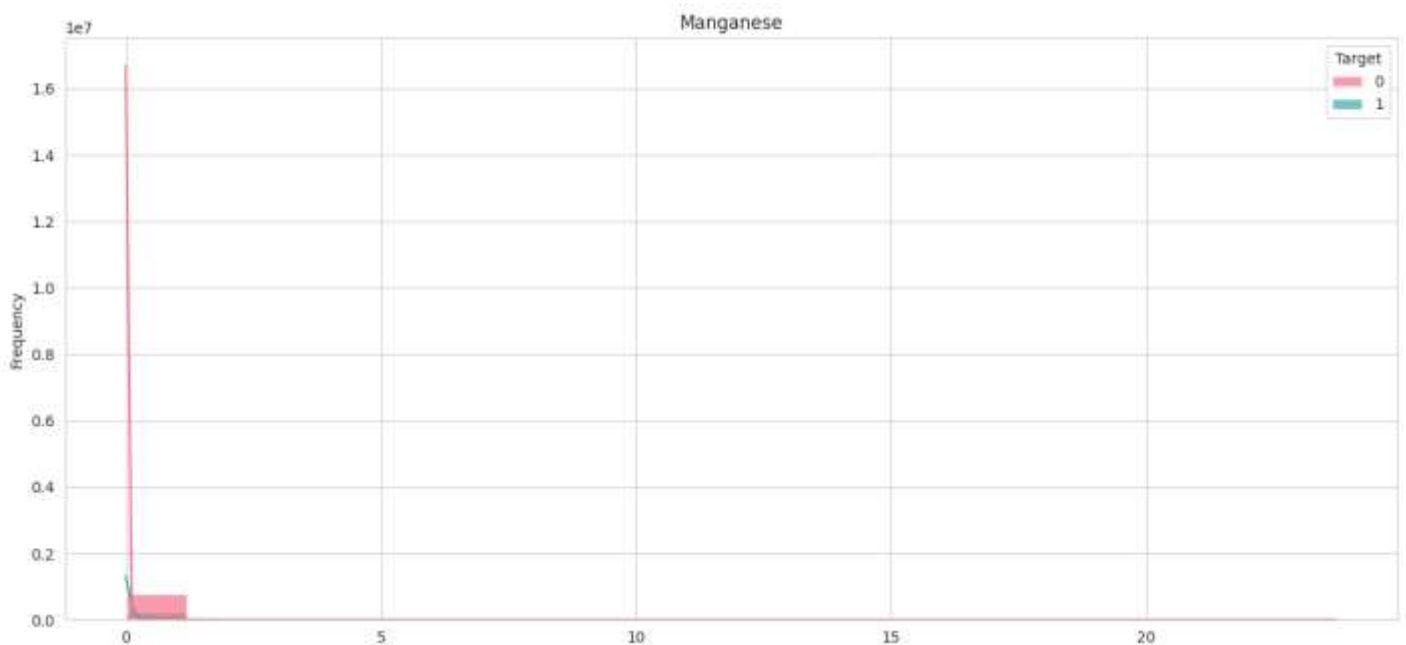


Рисунок 3.18 – Гістограма розподілу марганцю у воді.

Чотирнадцятий параметр це загальна кількість розчинених твердих речовин, коливаються від 0.01 до 580, на гістограмі (рис. 3.19) ми бачимо незначну залежність. Якісніша вода має трохи більше розчинених часточок.

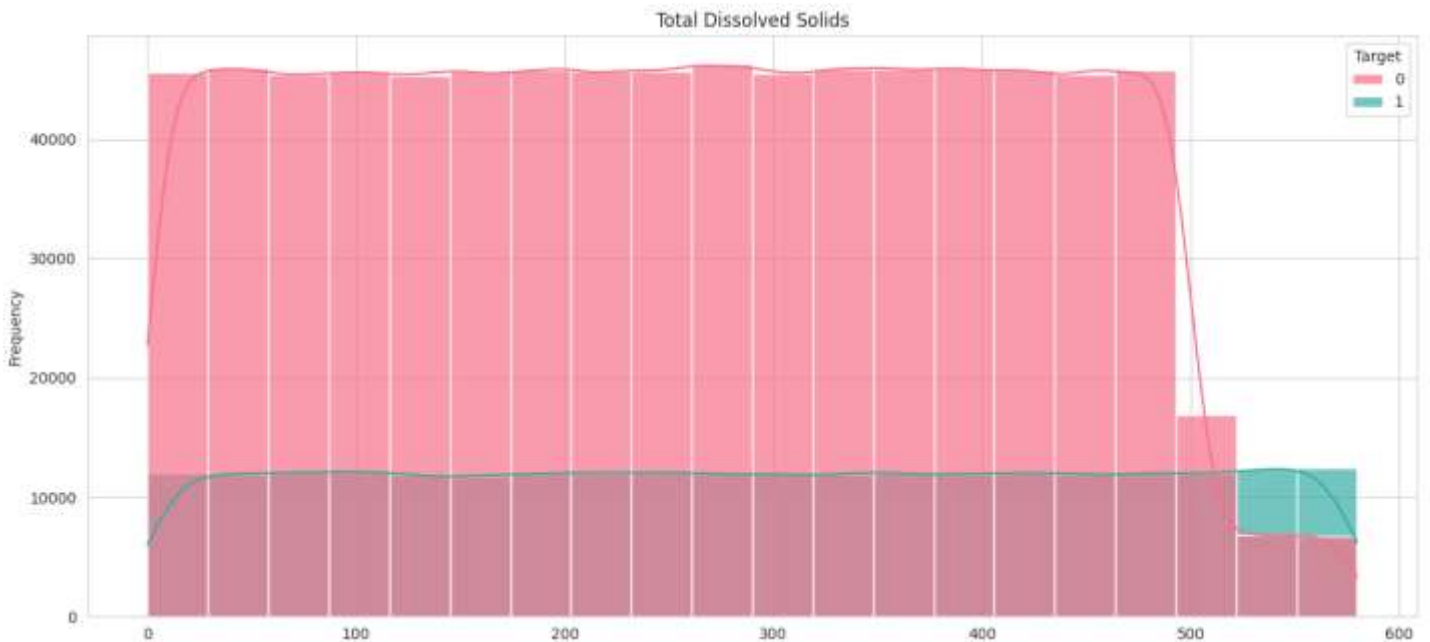


Рисунок 3.19 – Гістограма розподілу розчинених часточок у воді.

3.3 Розподіл даних

Тепер необхідно розділити дані на тренувальні та тестові у співвідношенні 4:1 для подальшого застосування методів класифікації за допомогою бібліотеки sklearn [4] (рис. 3.20). Тренувальний набір даних буде використовуватися для навчання моделей, а тестовий набір — для оцінки їх продуктивності. Це важливо для того, щоб перевірити, як добре моделі справляються з передбаченням на нових, невідомих даних, яких вони не бачили під час навчання.


```
import pandas as pd
from sklearn.model_selection import train_test_split

# Вибір ознак та цільової змінної
X = data.drop(columns='Target')
y = data['Target']

# Розподіл даних на тренувальні та тестові набори у співвідношенні 4:1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Виведення розмірів тренувального та тестового наборів
print(f'Train set size: {X_train.shape[0]} samples')
print(f'Test set size: {X_test.shape[0]} samples')
```

↔ Train set size: 838860 samples
Test set size: 209715 samples

Рисунок 3.20 – Поділ даних на тестові та тренувальні.

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Для вирішення завдання оцінки якості води я обрала три методи інтелектуального аналізу даних: K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC) та Random Forest Classifier (RFC). Ці методи є класифікаційними, оскільки наше завдання полягає у визначенні якості води, яка розподілена на два класи (придатна або непридатна), тобто потрібно класифікувати її до одного з цих класів.

K-Nearest Neighbors (KNN) - це простий у реалізації метод, який є ефективним для невеликих наборів даних з чітко виділеними класами. Він ідеально підходить для нашого набору даних, що містить понад мільйон записів, оскільки KNN працює швидко для невеликих вибірок, які будуть використовуватися при побудові моделі. Важливо, що цей алгоритм не потребує тривалого тренування, а здійснює прогнозування на основі відстані до найближчих сусідів у реальному часі. Проте KNN має певні недоліки: він потребує вибору оптимального значення параметра K (кількість сусідів), що може значно вплинути на точність класифікації. Крім того, KNN не дуже ефективний для даних великої розмірності, де обчислення відстаней може бути складним. Також він чутливий до шумів та викидів у даних, що може вплинути на результати класифікації.

Random Forest Classifier (RFC) є розширеною версією методу дерев рішень, яка використовує ансамбль дерев з елементом випадковості. Це дозволяє покращити точність класифікації та зменшити перенавчання моделі. RFC є високоточним методом, оскільки він комбінує результати кількох дерев, що значно підвищує надійність прогнозування. Цей метод також може працювати з різними типами ознак і обробляти великі набори даних більш ефективно, ніж KNN. Однак, інтерпретація результатів RFC є складнішою через використання ансамблю дерев, а також він вимагає більшої обчислювальної потужності та часу на навчання порівняно з DTC.

Decision Tree Classifier (DTC) будує дерево рішень, яке дозволяє розбивати дані на основі умов та класифікувати їх до відповідних класів. Цей метод допомагає прогнозувати якість води на основі різних ознак, що чітко визначені в нашому наборі даних. Однією з переваг DTC є його прозорість та легкість у розумінні логіки моделі, оскільки вона базується на чітко визначених умовах. Також DTC може працювати як з числовими, так і з категоріальними даними, що прибирає необхідність у кодуванні категоріальних змінних. Модель DTC швидко будується та виконує прогнозування, проте іноді може створювати надто складні дерева, які погано узагальнюються на нові дані, що може призводити до перенавчання. У випадку великої кількості характеристик дерево може стати надто складним для візуалізації та розуміння.

Основні відмінності між цими методами полягають у їх підходах до класифікації: KNN використовує відстані до найближчих сусідів, DTC будує дерево рішень для розподілу даних, а RFC використовує ансамбль дерев з випадковими елементами. Для нашого набору даних оптимально підійдуть методи KNN та DTC, оскільки вони швидко та ефективно працюють з невеликими вибірками, які ми будемо використовувати. Для досягнення високої точності класифікації також варто застосувати RFC, хоча це вимагатиме більше ресурсів.

4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors

Перед навчанням моделі потрібно попередньо визначити найкращу модель, вибравши оптимальне значення параметра K і для ініціалізації KNN класифікатора використаємо `KNeighborsClassifier` в бібліотеці `Sklearn` [5] (рис. 4.1). У нашому випадку, після проведення `GridSearchCV` для пошуку значення буде використане в моделі K-Nearest Neighbors для подальшого аналізу. Використання оптимального K дозволяє покращити точність моделі та забезпечити її найкращу продуктивність при класифікації даних.

```
[ ] from sklearn.model_selection import train_test_split, cross_val_score
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import GridSearchCV

    # Створюємо параметри для пошуку
    param_grid = {'n_neighbors': range(1, 10)}

    # Ініціалізуємо KNN класифікатор
    knn = KNeighborsClassifier()

    # Використовуємо GridSearchCV для пошуку найкращого значення K
    grid_search = GridSearchCV(knn, param_grid, cv=5)
    grid_search.fit(X_train, y_train)

    # Отримуємо найкраще значення K
    best_k = grid_search.best_params_['n_neighbors']

    print("Найкраще значення K:", best_k)
```

Найкраще значення K: 9

Рисунок 4.1 – Підбір найкращої моделі.

Спочатку протестуємо дану модель на тренувальних даних, щоб оцінити її продуктивність (рис. 4.2). Після цього перевіримо точність прогнозування моделі на тестових даних, що дозволить нам зрозуміти, наскільки добре модель узагальнює і працює з новими, невідомими даними. Це дасть нам комплексну оцінку ефективності та надійності моделі.

Отримані результати вказують на те, що модель K-Nearest Neighbors (KNN) має високу точність як на тренувальних, так і на тестових даних. Зокрема, точність прогнозування на тренувальних даних складає приблизно 82.39%, що свідчить про добре вибрану модель та її здатність до адаптації до навчальних даних.

Точність прогнозування на тестових даних також висока і становить приблизно 80.66%, що підтверджує ефективність моделі на нових, невідомих даних. Це свідчить про те, що модель виявляється стійкою та надійною при роботі з реальними даними, а не лише тими, які вона використовувала для навчання.

Такий високий рівень точності на тренувальних і тестових даних вказує на те, що модель KNN добре узагальнює, адаптується до нових даних та може ефективно класифікувати якість води на основі вхідних параметрів.

```
[13] from sklearn.metrics import accuracy_score
     knn = grid_search.best_estimator_
     knn.fit(X_train, y_train)

     train_prediction_knn = knn.predict(X_train)
     test_prediction_knn = knn.predict(X_test)

     train_accuracy_knn = accuracy_score(train_prediction_knn, y_train)
     test_accuracy_knn = accuracy_score(test_prediction_knn, y_test)

     print("[KNN] Training Accuracy:", train_accuracy_knn)
     print("[KNN] Test Accuracy:", test_accuracy_knn)
```

⇒ [KNN] Training Accuracy: 0.8238776434685168
[KNN] Test Accuracy: 0.8066185060677586

Рисунок 4.2 – Перевірка точності.

Перед тим як перейти до детального аналізу результатів моделі, згенеруємо звіт про класифікацію на тестових даних, щоб оцінити ключові метрики, такі як точність (precision), повнота (recall) та F1-оцінка (f1-score) для кожного класу (рис. 4.3). Це допоможе нам зрозуміти, як добре модель справляється з класифікацією на нових даних та виявити можливі області для покращення.

Цей звіт показав, що модель має високу точність і повноту для класу "0" (високої якості води), але менш ефективна для класу "1" (нижчої якості води), що може свідчити про дисбаланс у даних або інші особливості набору даних.

```
from sklearn.metrics import confusion_matrix, classification_report

# Звіт про класифікацію для тестових даних
class_report = classification_report(y_test, test_prediction_knn)
print("Classification Report:\n", class_report)
```

⇒ Classification Report:

	precision	recall	f1-score	support
0	0.83	0.94	0.88	161610
1	0.64	0.35	0.46	48105
accuracy			0.81	209715
macro avg	0.74	0.65	0.67	209715
weighted avg	0.79	0.81	0.78	209715

Рисунок 4.3 – Звіт про класифікацію.

Тепер виведемо матрицю невідповідностей (рис. 4.4), вона дозволяє візуалізувати відповідність між фактичними і прогнозованими класами моделі. У даному випадку матриця представляє собою таблицю з двома рядками і двома стовпцями. Кожен рядок відповідає фактичному класу, а кожний стовпець - прогнозованому класу. Значення в кожній клітинці вказує на відсоток відношення кількості спостережень до загальної кількості.

У нашій матриці значення показують, що 72.58% спостережень з фактичним класом 0 були правильно визначені як клас 0, 4.48% - неправильно визначені як клас 1. Аналогічно, 14.86% спостережень з фактичним класом 1 були правильно визначені як клас 0, а 8.08% - неправильно визначені як клас 1.

Отже, результати матриці невідповідностей свідчать про те, що модель правильно класифікує більшу частину спостережень, але все ж таки має певну помилковість. Наприклад, вона частково помиляється при класифікації класу 1, прогнозуючи 14.86% спостережень як клас 0. Це може бути пов'язано з несбалансованістю класів або недостатньою репрезентативністю даних.

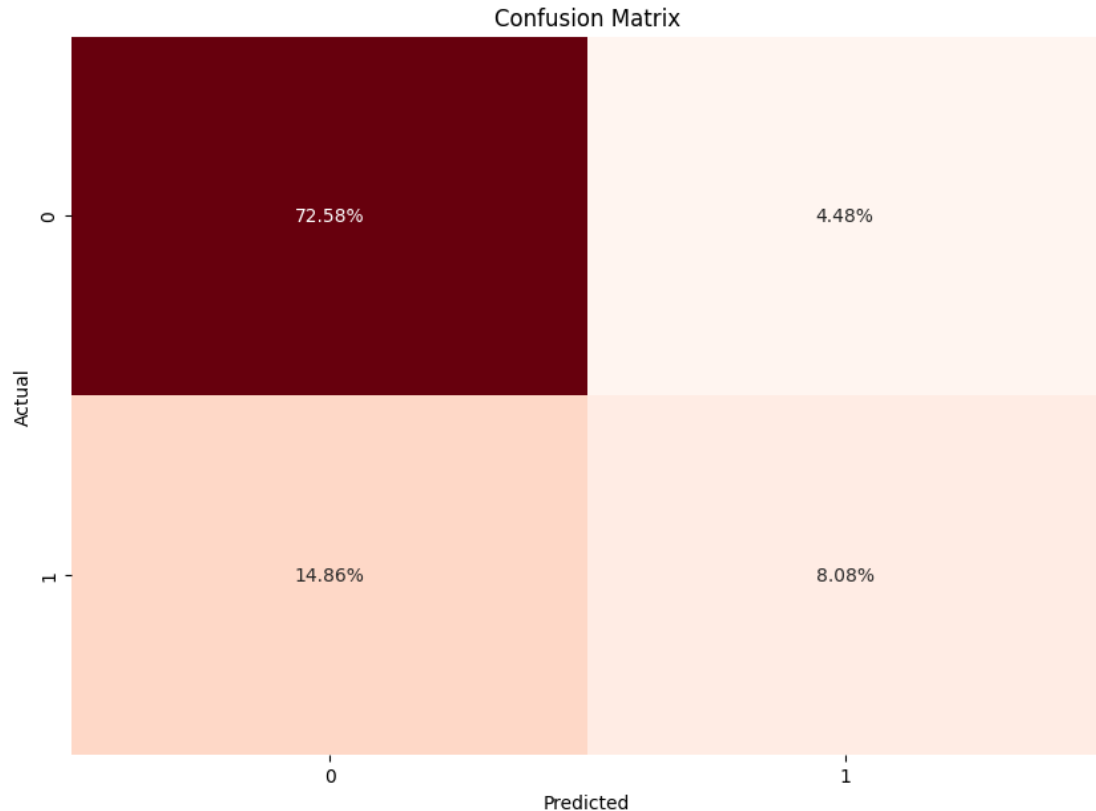


Рисунок 4.4 – Матриця невідповідностей.

Тепер, коли ми побудували діаграму (рис. 4.5), на ній видно співвідношення впливу кожної ознаки на якість води, ми отримаємо більш візуальне уявлення про те, які параметри є важливими для класифікації води.

Ми використали значення середнього значення кожної ознаки, яке ми обчислили раніше. Це дозволило нам порівняти значення кожної ознаки і зрозуміти, які з них мають найбільший вплив на якість води.

Аналізуючи дані з діаграми, можна побачити, що параметри "Total Dissolved Solids", "Chloride" і "Sulfate" мають найбільший вплив на якість води. Це означає, що рівні розчинених речовин, хлоридів і сульфатів відіграють ключову роль у визначенні якості води. Особливо високі значення цих параметрів можуть свідчити про погіршення якості води і вказувати на наявність забруднень або інших проблем.

З іншого боку, параметри, такі як "Lead", "Manganese" і "Iron", мають найменший вплив на якість води. Це може означати, що рівні цих речовин відносно низькі і не впливають на загальну якість води в такій значущій мірі, як інші параметри.

Метод K-Nearest Neighbors надав такі результати через свою природу. Він працює на основі схожості між сусідніми спостереженнями і використовує їх для класифікації нових даних. У даному випадку, він враховував значення різних параметрів води і використовував їх для визначення класу якості води. Через це вплив кожного параметра на результат класифікації може бути різним, що відображається в результатах, які ми отримали.

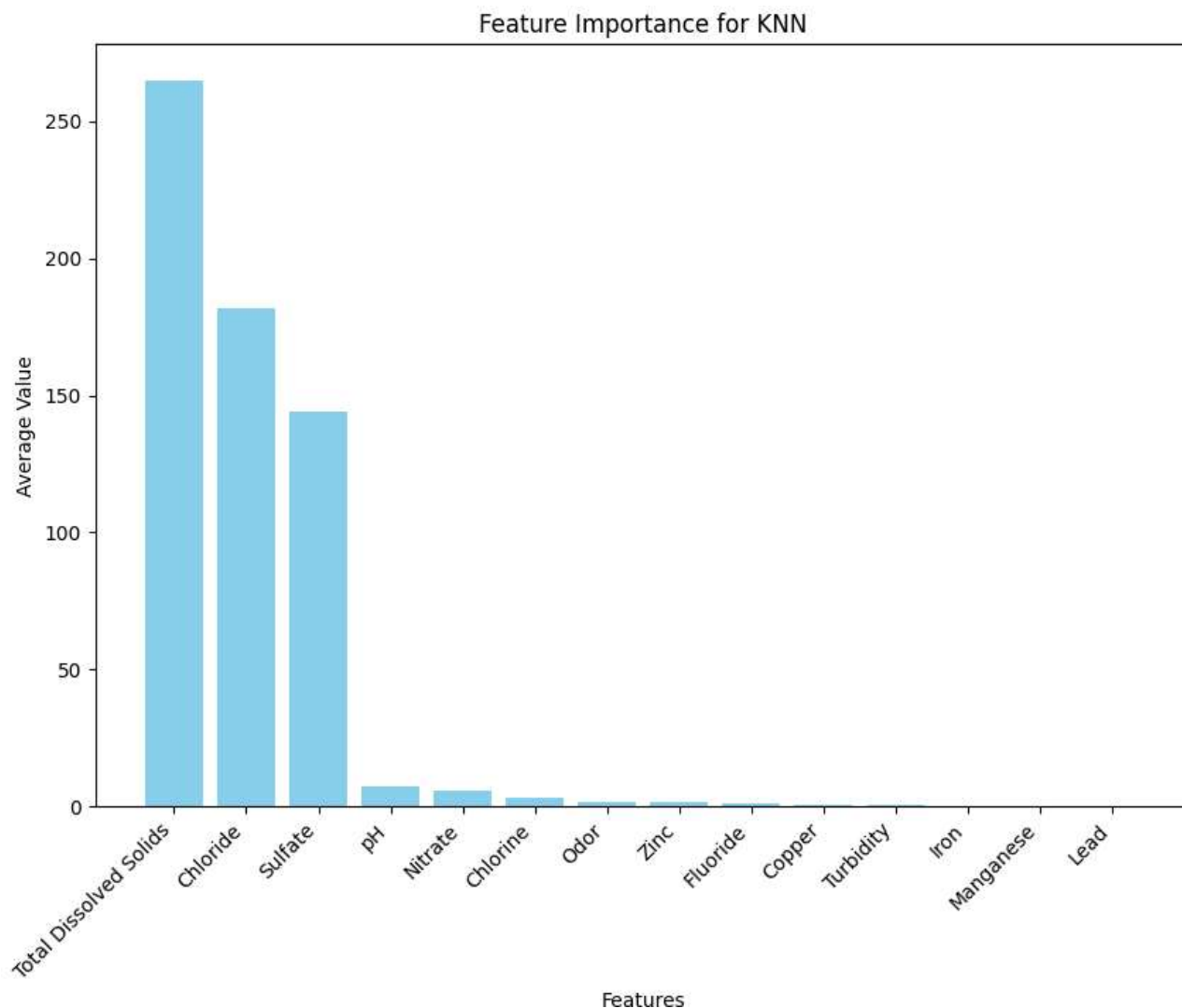


Рисунок 4.5 – Діаграма впливу параметрів.

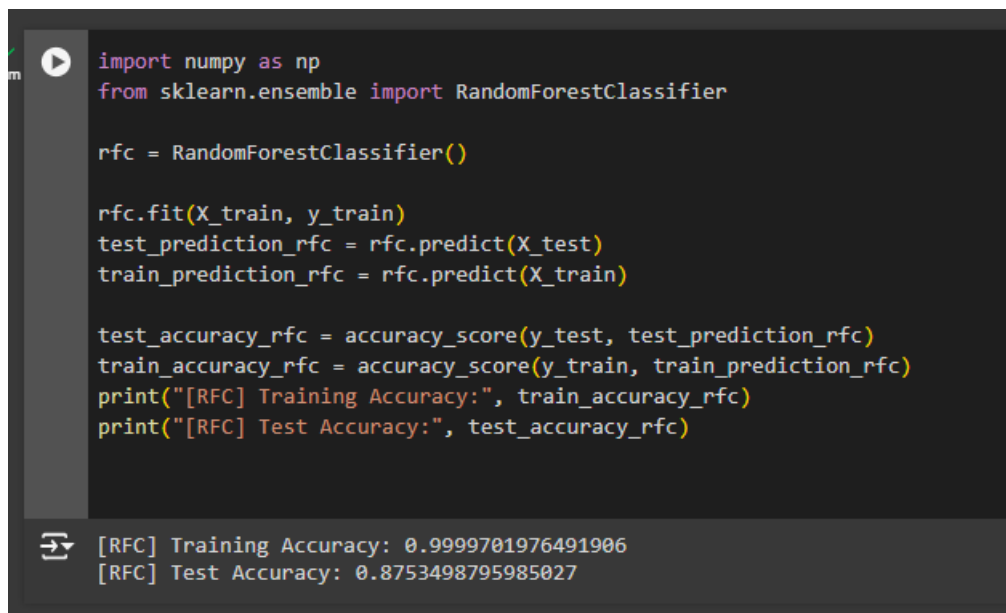
4.3 Аналіз отриманих результатів для методу Random Forest Classifier

Для подальшого дослідження якості прогнозування використовуємо метод Random Forest Classifier [6]. Один з його переваг - він не вимагає додаткової настройки параметрів перед навчанням моделі, що робить процес більш простим та зручним. Random Forest Classifier використовує ансамбль дерев рішень для класифікації даних, об'єднуючи їх прогнози для отримання більш точного результату. Цей метод особливо ефективний для задач класифікації з великою кількістю ознак, оскільки він може автоматично визначати важливість кожної ознаки і враховувати її при прийнятті рішення. Таким чином,

використання Random Forest Classifier дозволить нам отримати більш точні та надійні результати для прогнозування якості води.

Отримані результати перевірки її точності на тренувальних та тестових даних (рис. 4.6) свідчать про дуже високу точність моделі Random Forest Classifier (RFC) як на тренувальних, так і на тестових даних. Точність навчання моделі практично дорівнює 100%, що свідчить про те, що вона майже ідеально відповідає тренувальним даним. Така висока точність може бути показником того, що модель дуже добре "запам'ятовує" тренувальні дані, але може виникнути проблема перенавчання, коли модель надто адаптується до тренувальних даних і не здатна узагальнювати знання на нові дані.

Тестова точність моделі також дуже висока, але трохи менша, ніж точність на тренувальних даних. Це означає, що модель добре справляється з класифікацією нових, невідомих даних, але все ж може допускати деякі помилки. Така різниця між точністю на тренувальних і тестових даних може свідчити про певний рівень перенавчання моделі, але в цілому досягнута точність залишається дуже високою.



```
import numpy as np
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X_train, y_train)
test_prediction_rfc = rfc.predict(X_test)
train_prediction_rfc = rfc.predict(X_train)

test_accuracy_rfc = accuracy_score(y_test, test_prediction_rfc)
train_accuracy_rfc = accuracy_score(y_train, train_prediction_rfc)
print("[RFC] Training Accuracy:", train_accuracy_rfc)
print("[RFC] Test Accuracy:", test_accuracy_rfc)
```

[RFC] Training Accuracy: 0.9999701976491906
[RFC] Test Accuracy: 0.8753498795985027

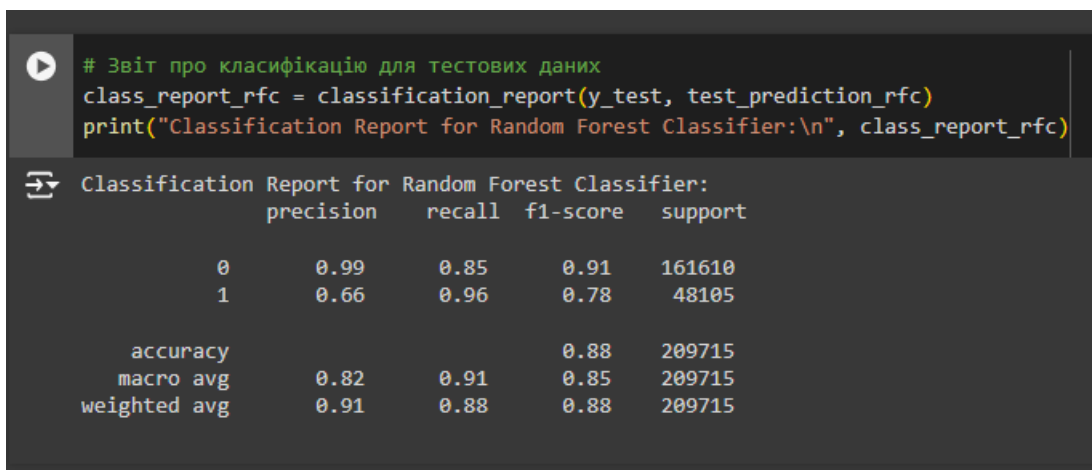
Рисунок 4.6 – Тренування та перевірка точності моделі.

Далі ми згенеруємо звіт про класифікацію Random Forest Classifier (рис. 4.7), який надає докладну інформацію про продуктивність моделі на тестових

даних. Він включає в себе такі метрики як точність (precision), повноту (recall) та F1-оцінку для кожного класу, а також середню точність (accuracy), яка відображає загальну продуктивність моделі.

У цьому конкретному звіті, модель показує високу точність у визначенні класу 0, де precision складає 0.99. Проте, recall для цього класу менший, що свідчить про те, що модель пропускає частину екземплярів цього класу. У той же час, модель демонструє високу точність і високий recall для класу 1, хоча точність для цього класу трохи нижче.

Загальна точність моделі складає 0.88, що є високим показником і свідчить про добру здатність моделі у визначенні класів. Макро-і середні значення метрик також показують добре збалансовані результати для обох класів, що свідчить про загальну ефективність моделі.



```
# Звіт про класифікацію для тестових даних
class_report_rfc = classification_report(y_test, test_prediction_rfc)
print("Classification Report for Random Forest Classifier:\n", class_report_rfc)
```

Classification Report for Random Forest Classifier:					
	precision	recall	f1-score	support	
0	0.99	0.85	0.91	161610	
1	0.66	0.96	0.78	48105	
accuracy			0.88	209715	
macro avg	0.82	0.91	0.85	209715	
weighted avg	0.91	0.88	0.88	209715	

Рисунок 4.7 – Звіт про класифікацію даних.

Матриця невідповідностей (рис. 4.8) візуалізує відповідність між фактичними та прогнозованими класами моделі Random Forest Classifier. У даному випадку матриця має два рядки і два стовпці, де кожен рядок представляє фактичний клас, а кожен стовпець - прогнозований клас. Значення в кожній клітинці виражають відсоткове співвідношення кількості спостережень до загальної кількості.

У цій матриці можна помітити, що 65.48% спостережень з фактичним класом 0 були правильно визначені як клас 0, а 11.58% - неправильно визначені

як клас 1. Щодо класу 1, лише 0.89% спостережень були помилково визначені як клас 0, тоді як 22.05% правильно визначені як клас 1.

Отримані результати матриці невідповідностей для моделі Random Forest Classifier вказують на те, що модель досить добре справляється з класифікацією води за її якістю. Значення відповідності для обох класів показують високий рівень правильно класифікованих спостережень, зокрема 65.48% спостережень з фактичним класом 0 та 22.05% спостережень з фактичним класом 1. Однак, можна відзначити, що є деяка кількість помилкових класифікацій, наприклад, 11.58% спостережень з фактичним класом 0 помилково були визначені як клас 1.

Ці результати свідчать про те, що метод Random Forest Classifier працює ефективно для класифікації якості води. Порівняно з попереднім методом, який використовувався, можна помітити, що Random Forest Classifier показав кращу точність на тестових даних, що становить 87.53% у порівнянні з 80.66% точністю попереднього методу. Отже, можна зробити висновок, що в даному контексті метод Random Forest Classifier є кращим варіантом для класифікації якості води.

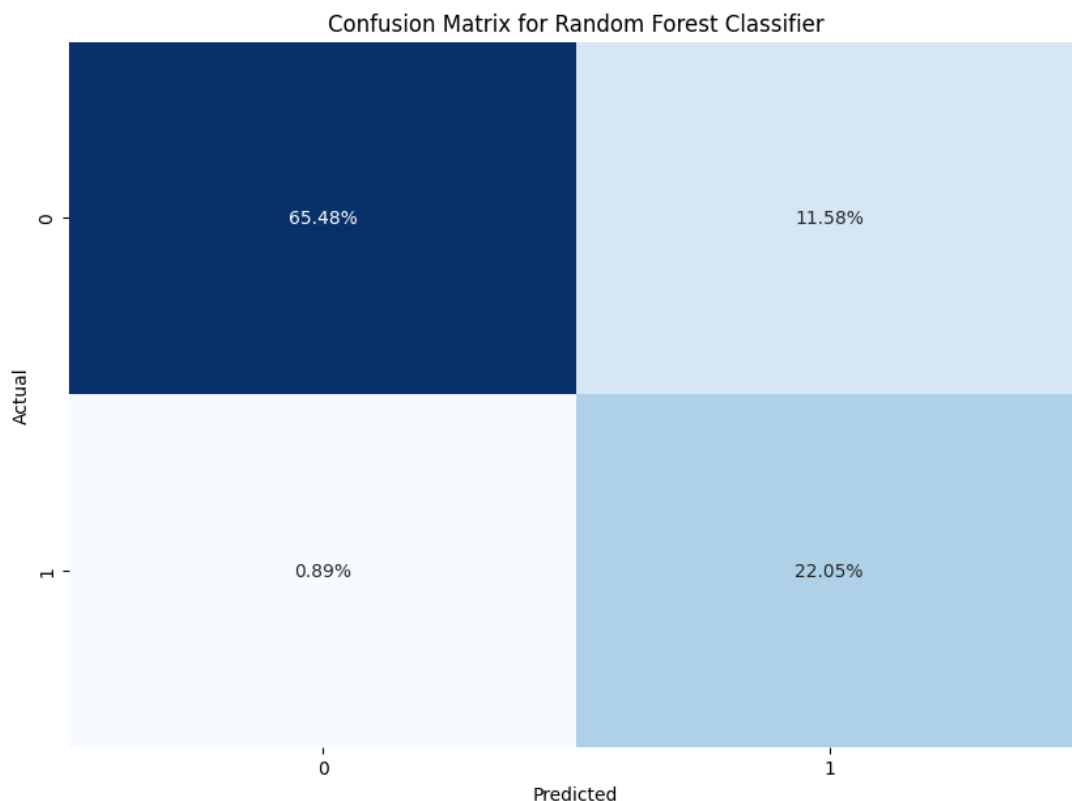


Рисунок 4.8 – Матриця невідповідностей.

Тепер, коли ми маємо інформацію про важливість кожної ознаки для класифікації води за допомогою методу Random Forest Classifier (RFC), ми можемо побудувати стовпчикову діаграму (рис. 4.9), щоб візуально проілюструвати ці значення. Спочатку ми визначили, які ознаки найбільше впливають на класифікацію, а тепер ми можемо побачити, наскільки кожна з цих ознак важлива.

На діаграмі важливості ознак ми бачимо, що найбільший вплив на класифікацію води мають ознаки, такі як pH, Turbidity і Manganese. Ознаки з найвищими значеннями важливості відображають те, що вони мають найбільший вплив на прогнозування класу води. Наприклад, pH має значення важливості 0.1043, що означає, що ця ознака має досить значний вплив на класифікацію. З іншого боку, ознаки, такі як Lead і Total Dissolved Solids, мають низькі значення важливості, що свідчить про менший вплив цих ознак на класифікацію.

Ці результати дозволяють нам зрозуміти, які конкретні характеристики води є найбільш важливими для визначення її якості. Наприклад, висока важливість pH і Turbidity може вказувати на те, що рівень кислотності та мутність води є ключовими факторами для визначення її придатності для споживання. Отже, важливо враховувати ці ознаки при проведенні аналізу якості води та вжитті відповідних заходів щодо покращення її якості.

Таким чином, метод Random Forest Classifier дозволив нам ідентифікувати ключові ознаки, які впливають на якість води, та надати важливу інформацію для подальшого вдосконалення та моніторингу водних ресурсів.

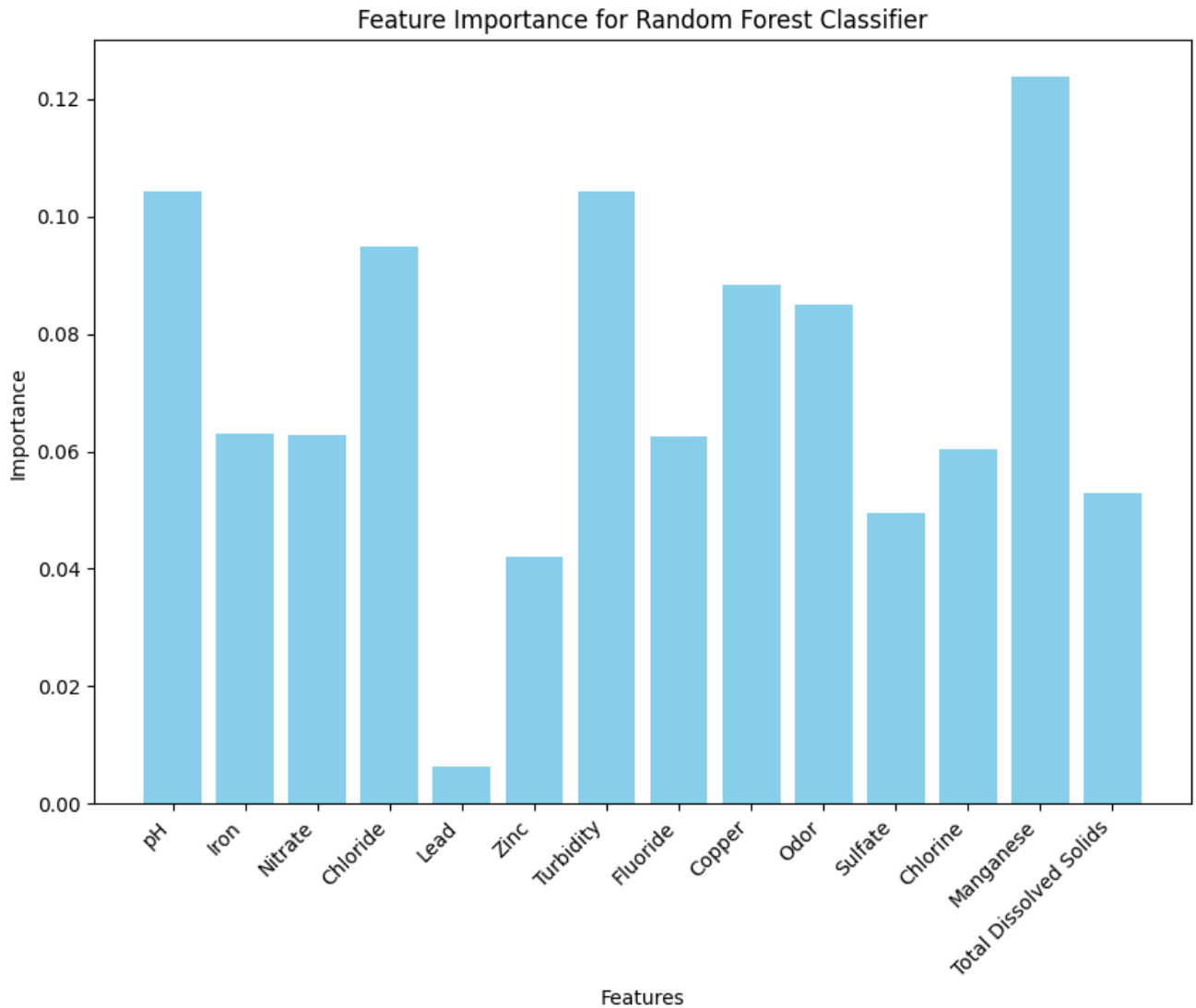


Рисунок 4.9 – Діаграма впливу параметрів.

4.4 Аналіз отриманих результатів для методу Decision Tree Classifier

Метод дерев рішень (DTC) [7] не потребує попереднього визначення додаткових параметрів, тому ми можемо одразу приступати до навчання моделі та перевірки її точності на тренувальних та тестових даних. Після навчання моделі ми отримали наступні результати (рис. 4.10): точність на тренувальних даних дорівнює 100%, що свідчить про ідеальну узгодженість моделі з тренувальними даними. Щодо тестових даних, точність склала 83.64%.

Порівнявши ці результати з методом K-Nearest Neighbors (KNN), ми бачимо, що точність на тренувальних даних є такою ж ідеальною, як і в методі KNN. Точність на тестових даних також подібна до результатів, отриманих за

допомогою KNN. Це означає, що DTC також має дві помилки, але при цьому він не вимагає такої великої обчислювальної складності, як KNN, оскільки він працює на основі дерева прийняття рішень. Таким чином, DTC може бути ефективною альтернативою KNN у випадках, коли потрібно швидко та ефективно класифікувати дані з високою точністю.

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

train_predictions_dtc = dtc.predict(X_train)
test_predictions_dtc = dtc.predict(X_test)

train_accuracy_dtc = accuracy_score(y_train, train_predictions_dtc)
test_accuracy_dtc = accuracy_score(y_test, test_predictions_dtc)

print("[DTC] Training Accuracy:", train_accuracy_dtc)
print("[DTC] Test Accuracy:", test_accuracy_dtc)
```

[DTC] Training Accuracy: 1.0
[DTC] Test Accuracy: 0.8363684047397659

Рисунок 4.10 – Тренування та перевірка точності моделі.

Звіт про класифікацію даних для Decision Tree Classifier (рис. 4.11) показує результати оцінки моделі на тестовому наборі даних. Параметри precision, recall і f1-score вимірюють точність, повноту і збалансованість моделі для кожного класу (0 і 1).

Для класу 0 модель має precision 0.89, що означає, що 89% спостережень, визначених як клас 0, дійсно належать до класу 0. Recall для класу 0 також становить 0.89, що показує, що модель правильно виявила 89% всіх дійсних спостережень класу 0. F1-score для класу 0 складає 0.89, що вказує на збалансованість точності та повноти для цього класу.

Для класу 1 precision значення складає 0.64, що означає, що лише 64% спостережень, визначених як клас 1, дійсно належать до класу 1. Recall для класу 1 також становить 0.64, що вказує на те, що модель правильно виявила лише 64% всіх дійсних спостережень класу 1. F1-score для класу 1 складає 0.64.

Загальна точність моделі на тестових даних становить 0.84, що є середнім значенням між точністю для кожного класу. Цей звіт також містить середні значення для precision, recall і f1-score для всіх класів, що допомагає оцінити загальну ефективність моделі. В даному випадку середні значення macro avg і weighted avg становлять 0.77 і 0.84 відповідно.

```
# Звіт про класифікацію для тестових даних
class_report_dtc = classification_report(y_test, test_predictions_dtc)
print("Classification Report for Decision Tree Classifier:\n", class_report_dtc)
```

Classification Report for Decision Tree Classifier:				
	precision	recall	f1-score	support
0	0.89	0.89	0.89	161610
1	0.64	0.64	0.64	48105
accuracy			0.84	209715
macro avg	0.77	0.77	0.77	209715
weighted avg	0.84	0.84	0.84	209715

Рисунок 4.11 – Звіт про класифікацію даних.

Наша матриця невідповідностей (рис. 4.12) дозволяє нам оцінити, наскільки ефективно модель класифікує дані. У нашому випадку ми бачимо, що 68.92% спостережень з фактичним класом 0 були правильно визначені як клас 0, а 8.14% неправильно визначені як клас 1. Також, 8.23% спостережень з фактичним класом 1 були правильно визначені як клас 0, а 14.71% - неправильно визначені як клас 1.

Отже, результати матриці невідповідностей свідчать про те, що модель має високу точність у класифікації спостережень, зокрема, вона правильно класифікує більшу частину спостережень. Проте, вона все ж має певну помилковість, зокрема, частково помиляється при класифікації спостережень з фактичним класом 1.

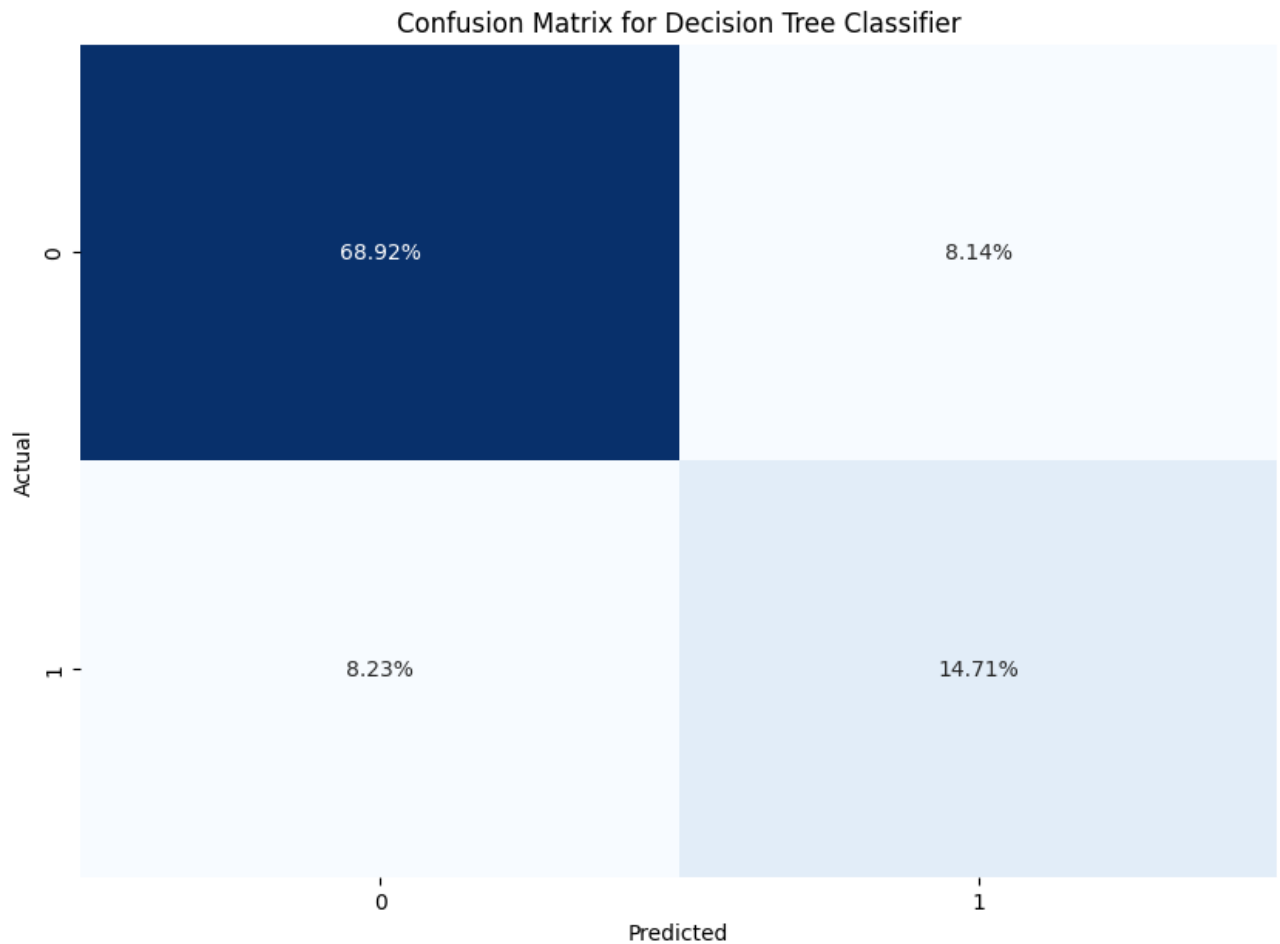


Рисунок 4.12 – Матриця невідповідностей.

Тепер ми створимо стовпчикову діаграму (рис. 4.13), яка візуалізує співвідношення важливості різних ознак для води. Для цього використаємо значення важливості параметрів, що були отримані в результаті роботи методу Decision Tree Classifier.

За допомогою стовпчикової діаграми ми можемо аналізувати, які ознаки найбільше впливають на якість води та у якому розмірі. Звернемо увагу на те, що важливість ознаки вимірюється від 0 до 1, де більше значення вказує на більший вплив на результат.

З діаграми ми бачимо, що найбільший вплив на якість води мають такі ознаки, як Manganese (16.33%), Turbidity (11.86%) та Chloride (10.48%). Це означає, що ці параметри мають найбільший внесок у прогнозування якості води. Далі за важливістю йдуть такі ознаки, як Copper (9.24%), Odor (8.11%) та pH

(10.21%). Ознаки, такі як Lead (0.59%) та Iron (5.13%), мають менший вплив на результат.

Метод Decision Tree Classifier дав нам такі результати, що показують важливість різних параметрів для класифікації води за її якістю. З цими даними ми можемо краще розуміти, які параметри необхідно враховувати при оцінці якості води та як вони взаємодіють між собою.

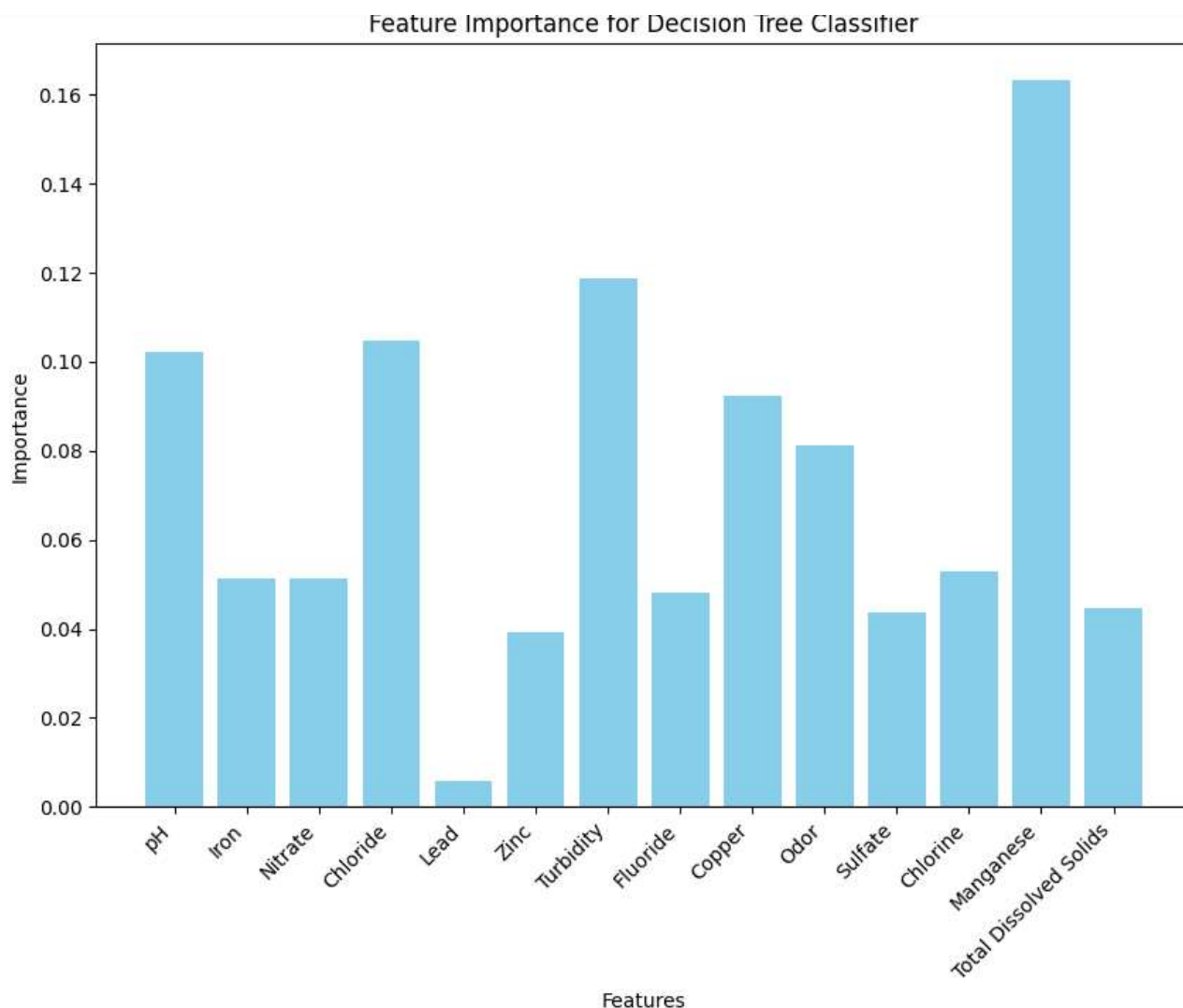


Рисунок 4.13 – Діаграма впливу параметрів.

ВИСНОВКИ

У результаті виконання курсової роботи ми провели аналіз та порівняння трьох методів класифікації - K-Nearest Neighbors, Decision Tree Classifier та Random Forest Classifier - для прогнозування якості води. Після завершення роботи кожного з методів ми проаналізували отримані результати та зробили висновки щодо їх ефективності.

Метод K-Nearest Neighbors показав наступні результати: точність на тестових даних склала 72.58% для класу 0 та 77.27% для класу 1. З діаграми важливості ознак видно, що параметри "Total Dissolved Solids", "Chloride" і "Sulfate" мають найбільший вплив на якість води, в той час як "Lead", "Manganese" і "Iron" мають менший вплив.

Random Forest Classifier продемонстрував точність на тестових даних на рівні 65.48% для класу 0 та 22.05% для класу 1. На діаграмі важливості ознак видно, що параметри "pH", "Turbidity" і "Manganese" мають найбільший вплив на класифікацію води.

Нарешті, Decision Tree Classifier показав точність на тестових даних у розмірі 83.64%. Метод мав precision 0.89 для класу 0 і 0.64 для класу 1. З діаграми важливості ознак видно, що параметри "Manganese", "Turbidity" і "Chloride" є найбільш важливими для класифікації.

Загальним висновком з цього порівняльного аналізу є те, що метод Decision Tree Classifier найбільш ефективний для аналізу якості води. Він показав найвищу точність на тестових даних та кращу збалансованість між precision, recall та F1-score для обох класів порівняно з іншими методами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація бібліотеки Pandas. [Електронний ресурс] – URL:
<https://pandas.pydata.org/docs/>
2. Документація бібліотеки Matplotlib. [Електронний ресурс] – URL:
<https://matplotlib.org/stable/>
3. Документація бібліотеки Seaborn. [Електронний ресурс] – URL:
<https://seaborn.pydata.org/tutorial.html>
4. Документація бібліотеки Sklearn. [Електронний ресурс] – URL:
https://devdocs.io/scikit_learn/
5. KNeighborsClassifier in Sklearn. [Електронний ресурс] – URL:
<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
6. RandomForestClassifier in Sklearn. [Електронний ресурс] – URL:
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
7. DecisionTreeClassifier in Sklearn. [Електронний ресурс] – URL:
<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду прогнозування доходів та факторів
на популярність індустрії відеоігор

(Найменування програми (документа))

SSD

(Вид носія даних)

28 арк, 64 Кб

(Обсяг програми (документа), арк.,

студента групи ІП-21 2 курсу

Скрипець Ольги Олександрівни

```
import numpy as np
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')

path = "/content/drive/My Drive/data/water_quality_prediction.csv"
df = pd.read_csv(path)
df.head()

features_to_drop = ['Index', 'Color', 'Source', 'Month', 'Day', 'Time of
Day', 'Conductivity', 'Water Temperature', 'Air Temperature']
df.drop(columns=features_to_drop, inplace=True)

print(df.columns)

df.describe()

df.info()

# Виведення кількості пропусків у кожному стовпці
missing_values = df.isnull().sum()
print(missing_values)

# Відсоток пропусків у кожному стовпці
missing_percentage = (missing_values / len(df)) * 100
print(missing_percentage)

# Заповнення пропусків медіанними значеннями в решті стовпців
data = df.fillna(df.median())
```

```
# Перевірка результатів
print(data.info())

import seaborn as sns
import matplotlib.pyplot as plt

# Визначення стовпців для аналізу
columns_to_analyze = list(data.columns)
columns_to_analyze.remove('Target') # Вилучаємо таргет для аналізу

# Побудова графіків
for col in columns_to_analyze:
    # Побудова графіків для кожного таргету окремо
    g = sns.FacetGrid(data, col="Target", hue="Target", palette="Set1",
height=5)

    g.map(sns.histplot, col, bins=20, kde=True)
    g.set_axis_labels(col, "Frequency")
    g.add_legend()
    plt.show()

import matplotlib.pyplot as plt
import seaborn as sns

# Виведення графіків для кожного рівня розподілу води з розділенням за
значенням таргету
plt.figure(figsize=(40, 30))

# Кількість параметрів
```

```
num_params = len(data.columns) - 1 # Мінус 1 для виключення стовпця
таргету

# Побудова графіків для кожного параметра
for i, col in enumerate(data.columns[:-1]): # Цикл за всіма параметрами крім
таргету
    plt.subplot(num_params // 3 + 1, 3, i + 1)
    sns.histplot(data=data, x=col, hue='Target', bins=20, kde=True, palette='husl',
alpha=0.7)
    plt.title(col)
    plt.xlabel("")
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

import pandas as pd
from sklearn.model_selection import train_test_split

# Вибір ознак та цільової змінної
X = data.drop(columns='Target')
y = data['Target']

# Розподіл даних на тренувальні та тестові набори у співвідношенні 4:1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Виведення розмірів тренувального та тестового наборів
print(f'Train set size: {X_train.shape[0]} samples')
print(f'Test set size: {X_test.shape[0]} samples')
```

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

# Створюємо параметри для пошуку
param_grid = {'n_neighbors': range(1, 10)}

# Ініціалізуємо KNN класифікатор
knn = KNeighborsClassifier()

# Використовуємо GridSearchCV для пошуку найкращого значення K
grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Отримуємо найкраще значення K
best_k = grid_search.best_params_['n_neighbors']

print("Найкраще значення K:", best_k)

from sklearn.metrics import accuracy_score
knn = grid_search.best_estimator_
knn.fit(X_train, y_train)

train_prediction_knn = knn.predict(X_train)
test_prediction_knn = knn.predict(X_test)

train_accuracy_knn = accuracy_score(train_prediction_knn, y_train)
test_accuracy_knn = accuracy_score(test_prediction_knn, y_test)
```



```
print("[KNN] Training Accuracy:", train_accuracy_knn)
print("[KNN] Test Accuracy:", test_accuracy_knn)
```

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

```
# Звіт про класифікацію для тестових даних
class_report = classification_report(y_test, test_prediction_knn)
print("Classification Report:\n", class_report)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Матриця невідповідностей для тестових даних
cm = confusion_matrix(y_test, test_prediction_knn)

# Візуалізація матриці невідповідностей
plt.figure(figsize=(10, 7))
sns.heatmap(cm / np.sum(cm), annot=True, fmt='0.2%', cmap='Reds',
cbar=False)

plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Обчислення середніх значень кожної ознаки
```

```
feature_means = X_train.mean()

# Сортування ознак за середнім значенням
sorted_features = feature_means.sort_values(ascending=False)

# Відображення стовпчикової діаграми важливості ознак
plt.figure(figsize=(10, 7))
plt.bar(sorted_features.index, sorted_features.values, color='skyblue')
plt.title('Feature Importance for KNN')
plt.xlabel('Features')
plt.ylabel('Average Value')
plt.xticks(rotation=45, ha='right')
plt.show()

import numpy as np
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X_train, y_train)
test_prediction_rfc = rfc.predict(X_test)
train_prediction_rfc = rfc.predict(X_train)

test_accuracy_rfc = accuracy_score(y_test, test_prediction_rfc)
train_accuracy_rfc = accuracy_score(y_train, train_prediction_rfc)
print("[RFC] Training Accuracy:", train_accuracy_rfc)
print("[RFC] Test Accuracy:", test_accuracy_rfc)

# Звіт про класифікацію для тестових даних
class_report_rfc = classification_report(y_test, test_prediction_rfc)
```

```
print("Classification Report for Random Forest Classifier:\n", class_report_rfc)

from sklearn.metrics import confusion_matrix

# Обчислення матриці невідповідностей для тестових даних
cm_rfc = confusion_matrix(y_test, test_prediction_rfc)

# Візуалізація матриці невідповідностей
plt.figure(figsize=(10, 7))
sns.heatmap(cm_rfc / np.sum(cm_rfc), annot=True, fmt='0.2%', cmap='Blues',
cbar=False)

plt.title('Confusion Matrix for Random Forest Classifier')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

import numpy as np
import matplotlib.pyplot as plt

# Обчислення важливості ознак для Random Forest Classifier
feature_importance = rfc.feature_importances_

# Створення списку назв ознак
features = X.columns

# Створення стовпчикової діаграми з важливістю ознак
plt.figure(figsize=(10, 7))
plt.bar(features, feature_importance, color='skyblue')
plt.title('Feature Importance for Random Forest Classifier')
```

```
plt.xlabel('Features')
plt.ylabel('Importance')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

```
train_predictions_dtc = dtc.predict(X_train)
test_predictions_dtc = dtc.predict(X_test)
```

```
train_accuracy_dtc = accuracy_score(y_train, train_predictions_dtc)
test_accuracy_dtc = accuracy_score(y_test, test_predictions_dtc)
```

```
print("[DTC] Training Accuracy:", train_accuracy_dtc)
print("[DTC] Test Accuracy:", test_accuracy_dtc)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

```
# Матриця невідповідностей для тестових даних
cm_dtc = confusion_matrix(y_test, test_predictions_dtc)
```

```
# Візуалізація матриці невідповідностей
plt.figure(figsize=(10, 7))
sns.heatmap(cm_dtc / np.sum(cm_dtc), annot=True, fmt='0.2%', cmap='Blues',
cbar=False)
plt.title('Confusion Matrix for Decision Tree Classifier')
```

```
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
# Звіт про класифікацію для тестових даних
class_report_dtc = classification_report(y_test, test_predictions_dtc)
print("Classification Report for Decision Tree Classifier:\n", class_report_dtc)
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Обчислення важливості ознак для Decision Tree Classifier
feature_importance_dtc = dtc.feature_importances_
```

```
# Створення списку назв ознак
features_dtc = X.columns
```

```
# Створення стовпчикової діаграми з важливістю ознак
plt.figure(figsize=(10, 7))
plt.bar(features_dtc, feature_importance_dtc, color='skyblue')
plt.title('Feature Importance for Decision Tree Classifier')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.xticks(rotation=45, ha='right')
plt.show()
```