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: «Прогнозування якості молока на основі різних показників методами Decision Tree, Random Forest та K-Nearest Neighbors»

Студента 2 курсу групи ІП-13
Спеціальності: 121
«Інженерія програмного забезпечення»
Бондаренка Максима Вікторовича

«ПРИЙНЯВ» з оцінкою

_____

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

_____
Підпис Дата

Київ - 2023 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 ІП-13 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Бондаренка Максима Вікторовича

1.Тема роботи Прогнозування якості молока на основі різних показників методами Decision Tree, Random Forest та K-Nearest Neighbors.

2.Строк здачі студентом закінченої роботи 07.06.2023

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/cpluzshrijayan/milkquality

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

1.Постановка задачі

2.Аналіз предметної області

3.Робота з даними

4.Інтелектуальний аналіз даних

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

6.Дата видачі завдання 09.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	09.02.2023	
2.	Визначення зовнішніх джерел даних	01.06.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	02.06.2023	
5.	Обґрунтування методів інтелектуального аналізу даних	05.06.2023	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	06.06.2023	
7.	Підготовка пояснювальної записки	07.06.2023	
8.	Здача курсової роботи на перевірку	07.06.2023	
9.	Захист курсової роботи	07.06.2023	

Студент _____ Бондаренко М.В.

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Ліхоузова Т.А

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім'я, по батькові)

"7" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 36 сторінок, 24 рисунків, 7 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: пошук, обробка та аналіз даних, реалізація програмного забезпечення для роботи з даними, їх подальшого аналізу та прогнозування.

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

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

ЗМІСТ

ВСТУП

Якість харчування завжди було важливою частиною здоров’я кожної людини, але сьогодні всі харчові компанії намагаються зменшити вартість виробництва за рахунок зменшення якості продуктів. Так, наприклад, молоко є одним з базових продуктів харчування більшості людей. Оскільки це є одним з базових продуктів харчування, з якого виробляється багато інших: кефір, йогурт, сир тощо — важливо вживати молоко гарної якості. Саме тому задача прогнозування якості молока допоможе фермерам робити молоко вищої якості за меншу вартість виробництва, а виробництвам молочних продуктів підбирати якісне та з виправданою ціною молоко для власних продуктів.

У рамках даної курсової роботи проаналізовано дані проб різного молока, що має різні характеристики. На основі отриманих даних використано три методи для прогнозування якості молока.

У даній курсовій роботі було використано дані технології: Python, Pandas[1], Matplotlib[2], Sklearn[4], Seaborn[3], 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[5], Decision Tree Classifier[7], Random Forest Classifier[6] – усе це методи для виконання задачі класифікації. Потрібно проаналізувати та порівняти результати виконання кожного методу, що дозволить обрати найбільш ефективний.

Вхідними даними є pH середовище, температура, смак, запах, жирність, каламутність, колір та оцінка якості.

Додаток можна використовувати для прогнозування якості молока в залежності від різних показників.

Використати мову програмування Python для реалізації застосунку.

Курсовий проект здати до дедлайну 08.06.2023 включно та виконати у єдиному стилі написання коду.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Молоко та молочні продукти – одні з найпопулярніших харчових продуктів не тільки в Україні, а і у світі. У 2021 році один українець у домогосподарствах у середньому споживав 18,8 кг молока й молочних продуктів на місяць у перерахунку на первинний продукт. Якість молока безпосередньо впливає на здоров'я людини, оскільки це біологічно складний продукт, що може завдати шкоди організму, якщо має погану якість. Важливим фактором є те, що молоко є основним продуктом харчування майже всіх дітей, що прямо впливає на здоров'я нації, тому задача прогнозування даного продукту є без перебільшення дуже важливою. На саму якість впливають багато факторів: pH середовище, температура, смак, запах, жирність, колір та каламутність. Їх ми і будемо досліджувати для прогнозування якості молока.

Даний аналіз може бути корисним для різних зацікавлених сторін:

1. Для фермерських господарств, щоб розуміти найбільші фактори впливу на якість та сприяти їх поліпшенню. Можливо певним критерієм можна знехтувати, але придати більше уваги для поліпшення іншого, це дозволить підвищити якість молока в цілому, але залишити вартість виробництва на попередньому рівні.
2. Прогнозування може бути важливим для контролюючих органів та наглядових установ, що встановлюють стандарти якості молочних продуктів. Вони можуть використовувати такі прогнози для моніторингу та забезпечення виконання нормативних вимог.
3. Може бути корисним для споживачів, оскільки допомагає зробити обізнані вибори при покупці молочних продуктів. Інформація про якість молока може дати споживачам впевненість у безпеці та якості продукту.
4. Виробники молочних продуктів можуть застосовувати даний аналіз для вибору найбільш оптимальних рецептур молока, щоб досягати високої якості продуктів за найменших витрат.

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

– завантаження вибірки даних;

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

3.РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Було обрано набір даних під назвою «Milk Quality Prediction», що складається з 1059 проб різного молока. Датасет складається з таблиці, що містить 8 стовпців: pH, Temperature, Taste, Odor, Fat, Turbidity, Colour, Grade. Стовпці несуть дану інформацію:

- pH – pH середовище молока, яке коливається від 3 до 9,5.
- Temperature – температура свіжого молока, що коливається від 34'C до 90'C.
- Taste – смак молока, який є булевим параметром. (1 – гарний, 0 – поганий.)
- Odor – запах молока, який є булевим параметром. (1 – гарний, 0 – поганий.)
- Fat – жирність молока, яка є булевим параметром. (1 – висока, 0 – низька.)
- Turbidity – каламутність молока, яка є булевим параметром. (1 – висока, 0 – низька.)
- Colour – колір молока, який визначається його RGB кодом від 240 до 255.
- Grade – оцінка якості молока, яка ділиться на три класи (висока, середня, низька).

3.2 Перевірка та первинний аналіз даних

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

```
In [1]: import pandas as pd

        df = pd.read_csv('dataset/milknew.csv')

        print(df.head())
```

```
   pH  Temprature  Taste  Odor  Fat  Turbidity  Colour   Grade
0 6.6          35      1     0    1          0     254    high
1 6.6          36      0     1    0          1     253    high
2 8.5          70      1     1    1          1     246     low
3 9.5          34      1     1    0          1     255     low
4 6.6          37      0     0    0          0     255  medium
```

```
In [2]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   pH          1059 non-null   float64
 1   Temprature  1059 non-null   int64
 2   Taste       1059 non-null   int64
 3   Odor        1059 non-null   int64
 4   Fat         1059 non-null   int64
 5   Turbidity   1059 non-null   int64
 6   Colour      1059 non-null   int64
 7   Grade       1059 non-null   object
dtypes: float64(1), int64(6), object(1)
memory usage: 66.3+ KB
None
```

Рисунок 3.1 – Загальна інформація про датафрейм.

На даному етапі можна помітити, що дані було зчитано коректно, але є проблема з назвами двох стовпців у самому датасеті: “Temprature” та “Fat ” мають некоректні назви, тому одразу ж виправимо це (рис. 3.2). З інформації зрозуміло, що набір даних містить 6 колонок типу int64, одну float64, а колонка, що виступає таргетом, має тип даних object, отже всі предиктори мають числові типи даних, тому нам не потрібно кодувати дані. Також можна побачити, що дані мають 1059 записів, і всі колонки мають по 1059 non-null значень, отже наш датасет не має пропусків та не вимагає додаткових перевірок на пусті значення.

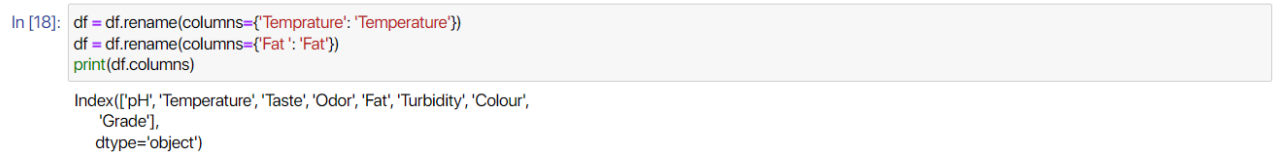


Рисунок 3.2 – Виправлення назв колонок.

Також для аналізу структури датасету згенеруємо описову статистику (рис 3.3).

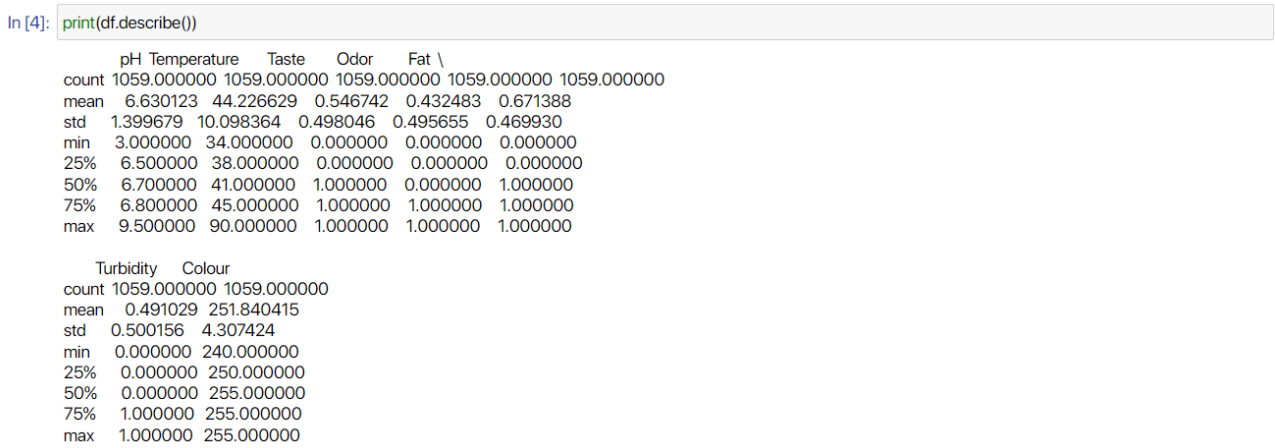


Рисунок 3.3 – Виведемо описову статистику.

Звідси можна побачити співвідношення смаку, запаху, жирності та каламутності, оскільки це булеві значення, то їх середнє гарно показує їх розподіл. Отже, частка смачного молока становить 54.6%, а ось молока з гарним запахом уже менше (43.2%), жирного молока значно більше (67.1%), а каламутність розподілена майже порівну (49% каламутного). Також можна побачити середні значення, стандартні відхилення, мінімальні значення, медіани, максимальні значення для інших параметрів. Мінімальні значення

свідчать про те, що набір даних не має від'ємних значень, тому в цій перевірці також немає потреби.

Виходячи з висновків вище можна стверджувати, що дані чисті та готові до подальшого аналізу.

Тепер можемо подивитись розподіл кількості молока для кожної оцінки якості (рис. 3.4). Для відображення графіків використовуємо бібліотеки matplotlib[2] та seaborn[3].

```
In [22]: import matplotlib.pyplot as plt
         import seaborn as sns

         sns.countplot(x="Grade", data=df, palette="Blues")
         plt.show()
```

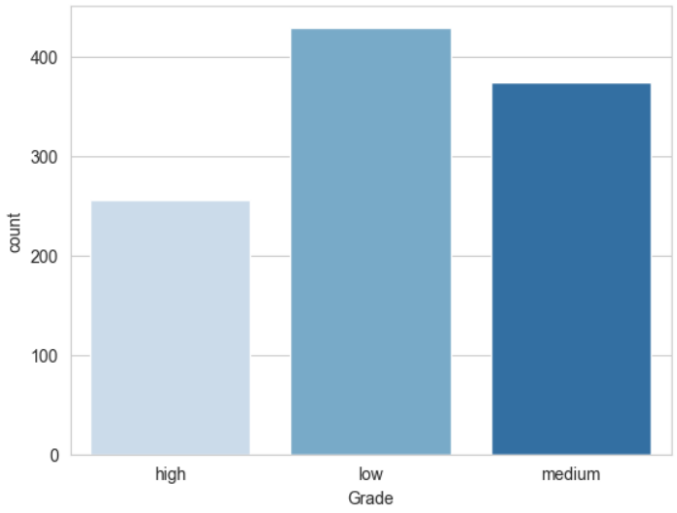


Рисунок 3.4 – Кількість молока для кожної оцінки якості.

З даного графіку видно, що найбільша кількість молока низької якості, а найменша – високої якості. Звідси зрозуміло, що на низькоякісне молоко найбільший попит через його відповідно низьку ціну.

Далі подивимось розподіл усіх параметрів для кожної оцінки якості окремо (рис. 3.5).

```
In [8]: sns.set_style("whitegrid")

        for parameter in df.columns[:-1]:
            fig, axs = plt.subplots(figsize=(13, 7), nrows=1, ncols=3, sharey=True)
            fig.suptitle('Distribution of {} by Milk Grade'.format(parameter))

            for i, grade in enumerate(['low', 'medium', 'high']):
                data_grade = df[df['Grade'] == grade]
                sns.histplot(data=data_grade[parameter], kde=True, ax=axs[i])
                axs[i].set_xlabel('Value')
                axs[i].set_ylabel('Frequency')
                axs[i].set_title(grade.capitalize())

            plt.tight_layout()
            plt.show()
```

Рисунок 3.5 – Код для побудови гістограм розподілу параметрів.

Першим параметр є pH середовище (рис. 3.6), з розподілу якого видно, що молоко середньої та високої якості коливається в межах від 6.4 до 6.8, усе правильно, тому що за дослідженнями pH свіжого молока становить 6,3-6,7. В цей же час pH молока поганої якості досить рівномірно коливається в межах від 3 до 9. Отже, візуально зрозуміло, що цей параметр є дуже гарним для класифікації якості, а метод Decision Tree може використовувати його на найвищому рівні, бо якщо pH не входить в досить вузьку межу 6.4 – 6.8, то молоко ніяк не може бути середньої чи високої якості.

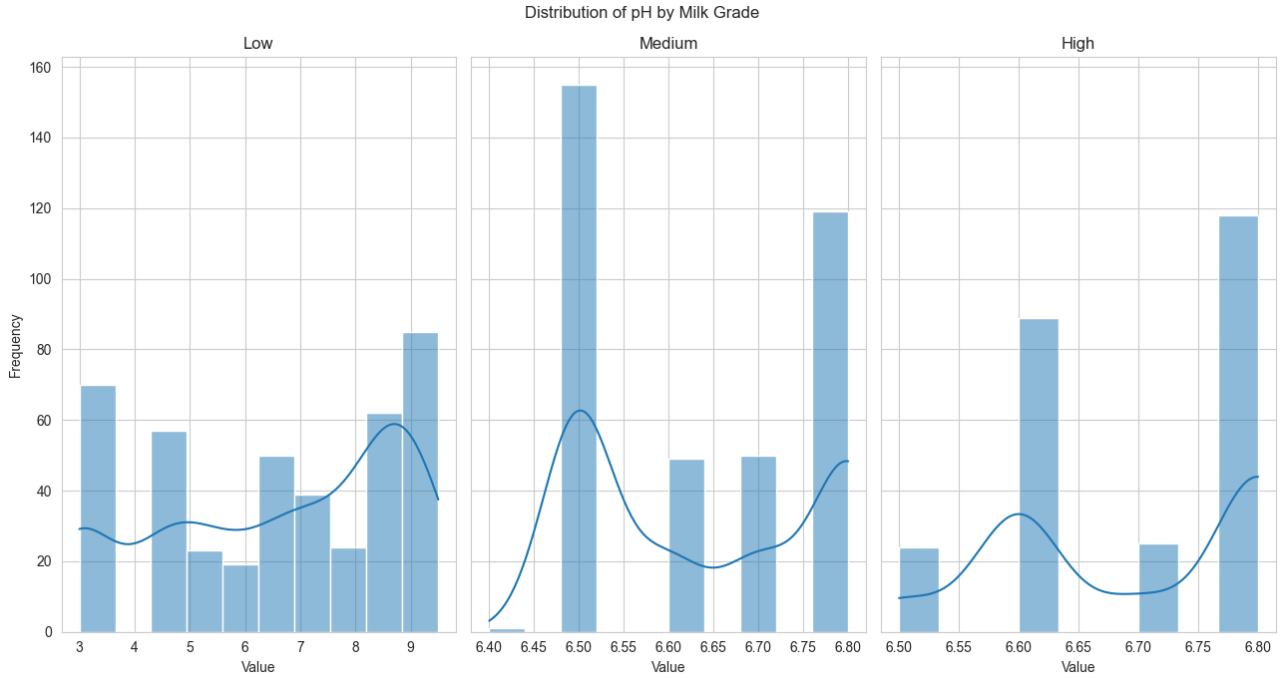


Рисунок 3.6 – Гістограма розподілу pH середовища.

Другим параметром є температура (рис. 3.7), з розподілу якої видно, що свіже молоко середньої та високої якості входить в межі від 34 до 45 градусів, тобто це температура близька до температури тіла корови, коли як досить велика кількість молока низької якості перевищує ці межі, отже ми знову маємо досить вузькі межі для класифікації молока середньої та високої якості, тому цей параметр також дозволяє гарно класифікувати молоко по якості.

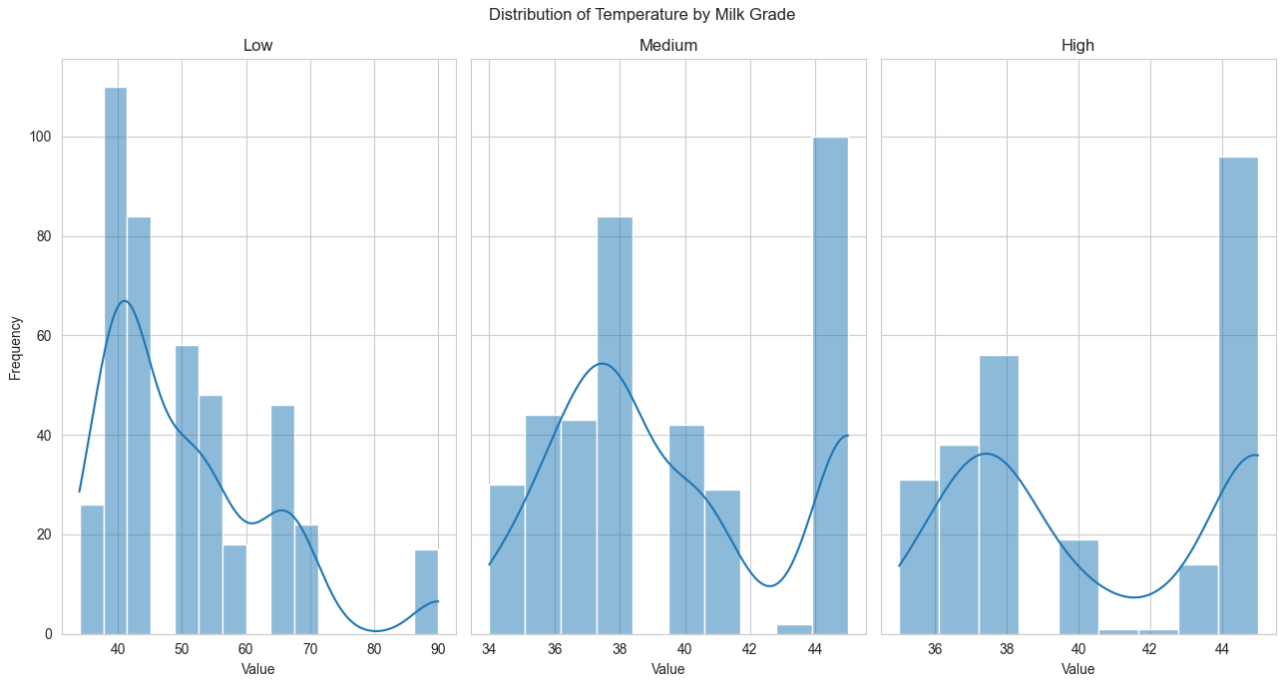


Рисунок 3.7 – Гістограма розподілу температури молока.

Третім параметром є смак (рис. 3.8), з розподілу якого видно, що молоко високої та низької якості досить схожі по співвідношенню гарного смаку до поганого, це можна пояснити різними добавками в поганому молоці, яке підсилює смак, а молоко високої якості смачне від природи. В цей час молоко середньої якості знаходиться на проміжному рівні, де природній смак недостатньо гарний, але також відмовляються від добавок, бо смак є досить нормальним, але зображений поганим, бо це недолік датасету і маленької варіації смаку (або гарний, або поганий), через що неможливо показати середній смак.

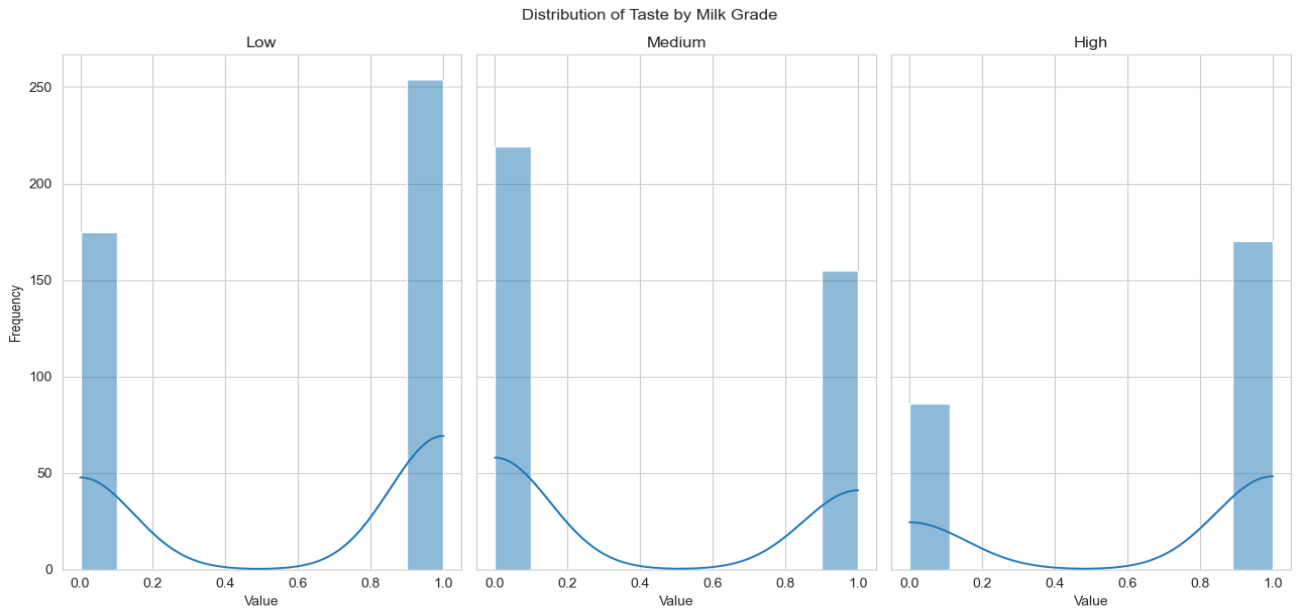


Рисунок 3.8 – Гістограма розподілу смаку молока.

Четвертим параметром є запах молока (рис. 3.9), з розподілу якого видно, що молоко високої якості майже завжди має гарний запах, молоко низької якості приблизно порівну має поганий та гарний запах, але знову ж гарний запах у цьому випадку отримують шляхом добавок, а ось з молоком середньої якості ситуація дуже погана, але це пояснюється так само, як і зі смаком, бо воно має недостатньо гарний запах від природи, але і не потребує добавок, бо той запах, який воно має, повністю відповідає його якості. Знову ж недолік датасету в маленькій варіації запаху, він або гарний, або поганий, нема усереднених значень. Цей параметр гарно використовувати для прогнозування, оскільки наявна велика різниця результатів. Так, якщо молоко має поганий запах, то маленька ймовірність того, що воно буде високої якості, але досить велика ймовірність того, що воно буде середньої якості.

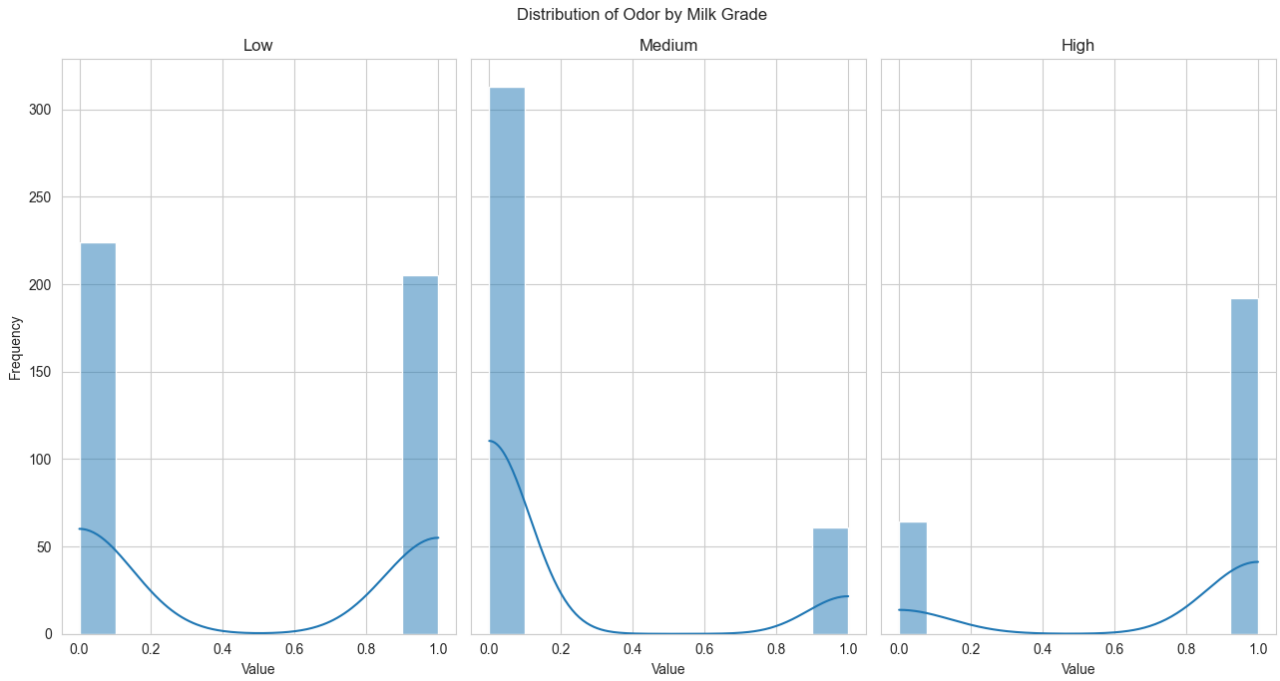


Рисунок 3.9 – Гістограма розподілу запаху молока.

П’ятим параметром є жирність молока (рис. 3.10), з розподілу якого видно, що молоко високої якості не може мати маленьку жирність, це цілком справедливо, бо від природи молоко має середню жирність 3.8%, а для зменшення жирності його просто розбавляють водою, звісно розбавлене молоко не може бути високої якості. Розбавляти молоко може бути притаманним явищем для молока низької та середньої якості, а робити вони це можуть з різних причин, але в будь-якому випадку це приведе до зменшення вартості

виробництва. Цей параметр досить гарний для прогнозування, оскільки маючи факт того, що молоко має маленьку жирність, воно не може бути високої якості, а це відсіює один з класів.

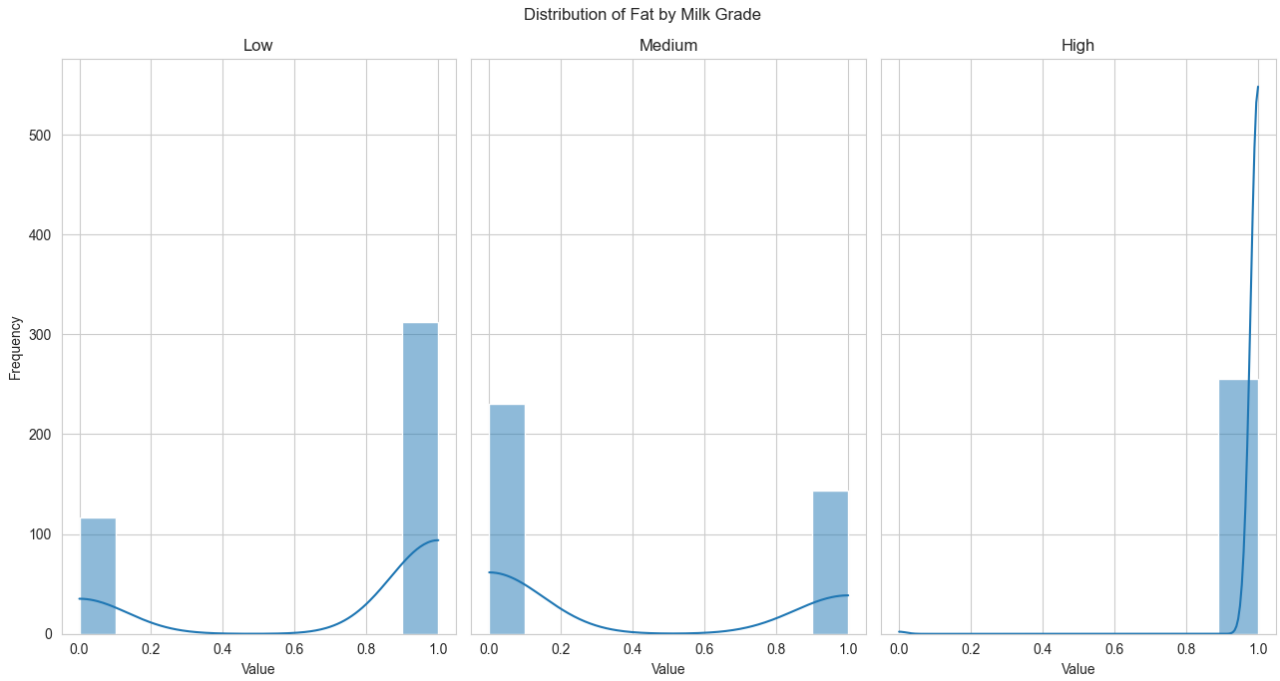


Рисунок 3.10 – Гістограма розподілу жирності молока.

Шостим параметром є каламутність молока (рис. 3.11), з розподілу якого видно, що у молока високої та низької якості переважає каламутне молоко, а от в молоці середньої якості навпаки. За візуальною оцінкою не видно певної закономірності між каламутністю та якістю.

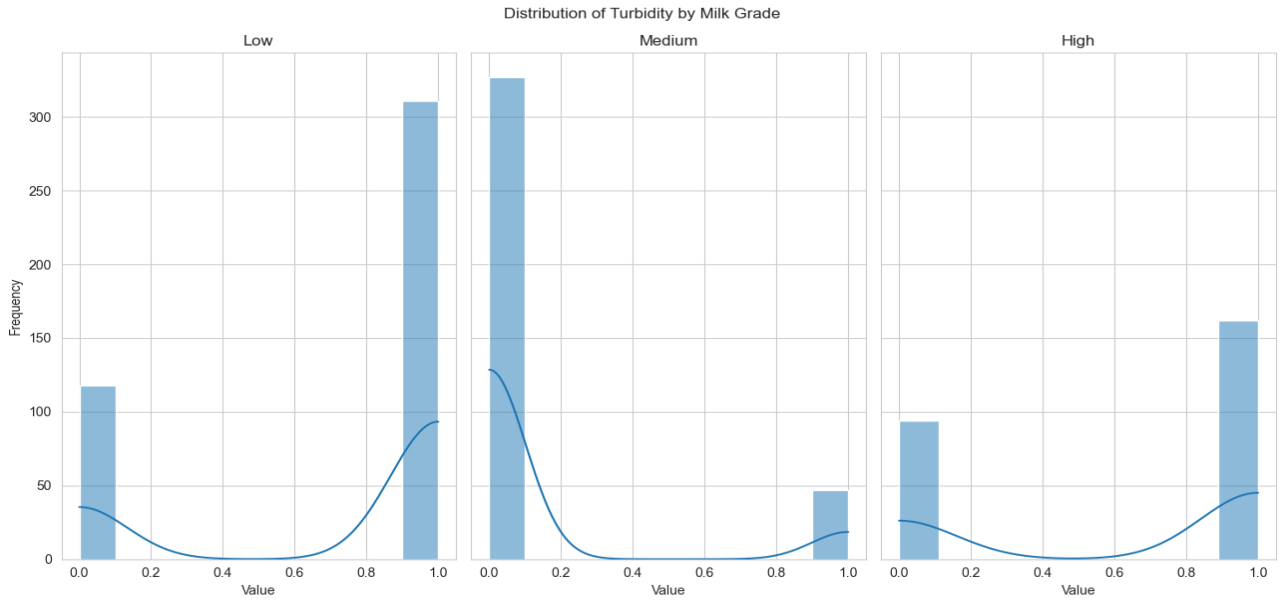


Рисунок 3.11 – Гістограма розподілу каламутності молока.

Останнім параметром є колір молока (рис. 3.12), розподіл якого схожий для всіх класів, тому це досить поганий параметр для визначення якості молока.

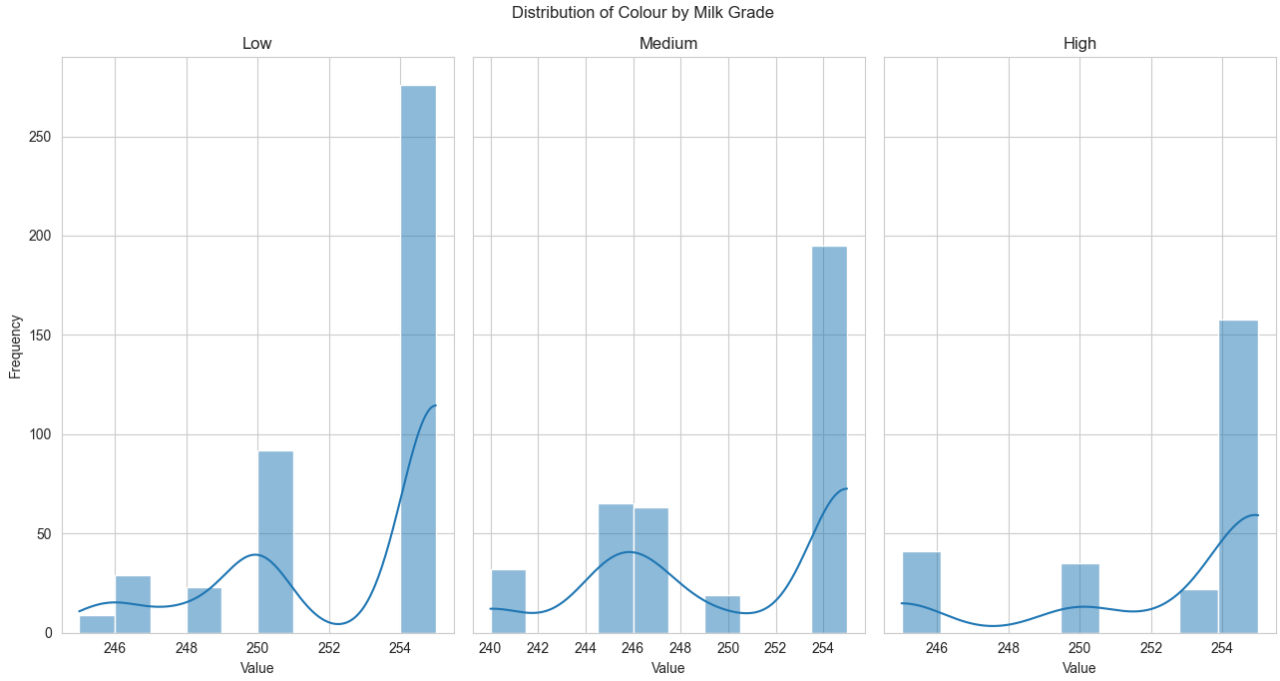


Рисунок 3.12 – Гістограма розподілу кольору молока.

3.3 Поділ даних

На даному етапі потрібно розділити дані на тестові та тренувальні у відношенні 1 до 4 відповідно для подальшого застосування методів класифікації за допомогою бібліотеки sklearn[4] (рис. 3.13). Тренувальні дані використовуються для навчання самої моделі, а на тестувальних будемо перевіряти саму модель, оскільки це дані, які модель не бачила при навчанні.

```
In [9]: from sklearn.model_selection import train_test_split

X = df.drop('Grade', axis=1)
y = df['Grade']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 3.13 – Поділ даних на тестові та тренувальні.

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Я обрав три методи для виконання поставленої задачі: K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC) та Random Forest Classifier (RFC) – усе це методи класифікації, оскільки наша задача полягає в оцінці якості молока, яка ділиться на три класи, тобто потрібно віднести до одного з трьох класів.

K-Nearest Neighbors є простим в реалізації методом, який ефективний для невеликих наборів даних з чітко виділеними класами, тому ідеально підходить для нашого датасету з 1059 записами і трьома класами “high”, “medium”, “low”. Що важливо, алгоритм достатньо швидкий, бо не потребує тренування моделі, а виконує прогнозування в реальному часі. Також алгоритм має певні недоліки, наприклад, він вимагає вибору оптимального значення параметру K, який визначає кількість найближчих сусідів для класифікації. Вибір даного параметру може вплинути на точність моделі, тому це дійсно важливо. Даний метод погано працює з даними великої розмірності, оскільки алгоритму стає складно обчислити відстань у кожному вимірі, але це не стосується нашого набору даних. KNN метод дуже чутливий до шумів та викидів, оскільки базується на близькості до сусідів, тому це може вплинути на точність класифікації.

Decision Tree Classifier є деревом, яке дозволяє розбивати дані на основі умов та ділити на відповідні класи, це в свою чергу допомагає прогнозувати якість молока на основі різних ознак, які є чітко визначеними в нашому наборі даних. Однією з переваг є легкість розуміння логіки моделі, оскільки вона базується на відповідності даних певним чітко визначеним умовам. Також DTC може працювати не лише з числовими даними, а і з категоріальними, що прибирає вимогу в кодуванні категоріальних даних. Побудова рішення та прогнозування відбувається достатньо швидко через легкий вигляд самої моделі, яка складається лише з одного дерева порівняно з RFC, але іноді DTC може створювати достатньо складні дерева, які погано застосовуються для нових даних, що призводить до потреби перенавчання моделі, а незначні зміни в даних можуть значно змінити структуру дерева. У випадку великої кількості

характеристик дерево може стати складним для розуміння та візуалізації, оскільки матиме багато рівнів та розділів.

Random Forest Classifier використаємо для порівняння з іншої деревовидною моделлю DTC, але вони багато чим різняться. Цей метод поєднує переваги дерева рішень з елементом випадковості, це дозволяє покращити точність та зменшити перенавчання моделі. Цей метод має високу точність порівняно з методами описаними вище, бо складається з кількох дерев, і результат визначається кількома деревами, що сильно впливає на точність моделі. RFC так само може працювати з різними типами ознак, як і DTC. Також цей метод здатен ефективно оброблювати великі набори даних на відміну від KNN. Цей метод також має певні мінуси: інтерпретація результатів значно складніша, оскільки використовується ансамбль дерев, а не одне дерево рішень, також порівняно з DTC даний модель вимагає більшої обчислювальної потужності та часу навчання.

Основними відмінностями між методами є різні підходи до класифікації. Так, наприклад, KNN використовує відстань до найближчих сусідів для класифікації, DTC використовує дерево рішень, яке розділяє дані на основі умов, RFC схожий на DTC, але використовує ансамбль дерев з елементом випадковості. Для наших даних малої розмірності гарно підійдуть KNN та DTC методи, оскільки вони гарно та швидко працюють з такою кількістю даних, також ми маємо 4 булевих параметри, що гарно підійдуть для побудови дерева рішень. Для забезпечення більш високої точності можна використати RFC, але це задіє значно більше ресурсів, які можуть не виправдатись.

4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors

Перед навчанням моделі треба попередньо обрати найкращу модель за найкращим параметром K (рис. 4.1), в нашому випадку найкращим параметром K виявилось 1, тож використаємо цю модель.

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
param_grid = {'n_neighbors': range(1, 10)}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_k = grid_search.best_params_['n_neighbors']

print("The best k value:", best_k)
```

```
The best k value: 1
```

Рисунок 4.1 – Підбір найкращої моделі.

Для початку спробуємо дану модель на тренувальних даних, потім перевіримо точність прогнозування на тестових (рис. 4.2).

```
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)
```

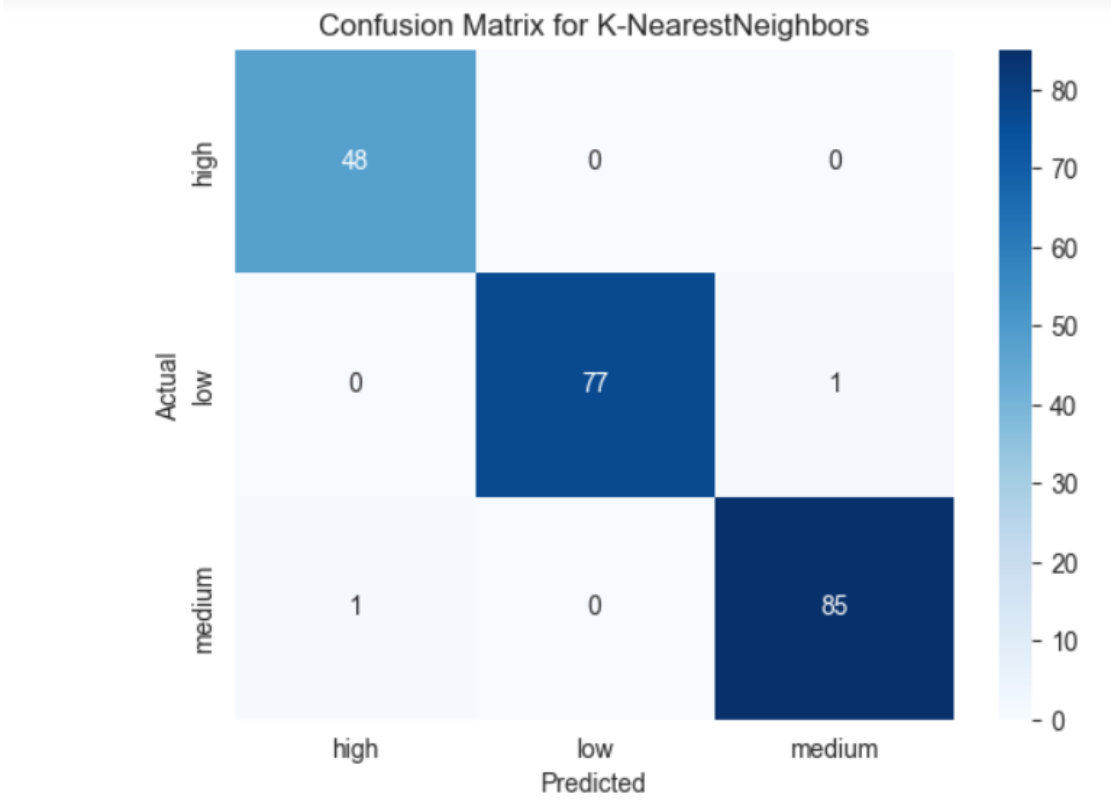
```
[KNN] Training Accuracy: 1.0
[KNN] Test Accuracy: 0.9905660377358491
```

Рисунок 4.2 – Перевірка точності моделі.

Ми маємо ідеальну точність для тренувальних даних, що свідчить про те, що модель ідеально обраховує все, але нас більше цікавлять тестові дані, які також мають дуже гарний результат. Така висока точність обумовлена простотою даних, оскільки 4 з 7 параметрів є булевими, а pH середовище та температура мають чітко окреслені межі для кожного з класів, усе це дає змогу чітко класифікувати дані.

Для перевірки роботи моделі використаємо матрицю невідповідностей та звіт про класифікацію (рис. 4.3). З матриці видно, що модель майже ідеально визначила якість молока (значення головної діагоналі – правильно прогнозовані), але було всього дві похибки: один раз модель прогнозувала середнє значення, яке насправді було низьким, а ще один раз прогнозувала високе значення, коли насправді воно середнє. Можна помітити, що обидва рази модель трохи завищила якість молока, але не було грубих помилок, коли низькоякісне молоко було б назване високоякісним чи навпаки, що досить добре. Також модель безпомилково ідентифікувала все високоякісне молоко, отже воно має найбільш виразні параметри, хоча усі класи мають рівний

параметр f1-score, який є поєднанням двох інших параметрів та показує баланс між ними, що свідчить про те, що модель однаково гарно ідентифікує всі класи.



Classification report for K-NearestNeighbors:

	precision	recall	f1-score	support
high	0.98	1.00	0.99	48
low	1.00	0.99	0.99	78
medium	0.99	0.99	0.99	86
accuracy			0.99	212
macro avg	0.99	0.99	0.99	212
weighted avg	0.99	0.99	0.99	212

Рисунок 4.3 – матриця невідповідностей та звіт про класифікацію.

Також було написано алгоритм (рис. 4.4) для побудови стовпчикової діаграми (рис. 4.5) з важливістю параметрів для методу KNN, оскільки він не має вбудованої функції для розрахунку.

```
feature_importance = {}
for feature in X_train.columns:
    X_train_subset = X_train.drop(feature, axis=1)
    X_val_subset = X_test.drop(feature, axis=1)

    knn_subset = KNeighborsClassifier()
    knn_subset.fit(X_train_subset, y_train)
    y_pred_subset = knn_subset.predict(X_val_subset)
    accuracy_subset = accuracy_score(y_test, y_pred_subset)

    feature_importance[feature] = accuracy_score(test_prediction_knn, y_test) - accuracy_subset

sorted_importance = sorted(feature_importance.items(), key=lambda x: x[1], reverse=True)

features = [item[0] for item in sorted_importance]
importance_values = [item[1] for item in sorted_importance]

total_importance = sum(importance_values)

normalized_importance = [importance / total_importance for importance in importance_values]

plt.figure(figsize=(10, 6))
plt.bar(features, normalized_importance)
plt.xlabel('Features')
plt.ylabel('Normalized Importance')
plt.title('Normalized Feature Importance in KNN Model')
plt.xticks(rotation=90)
plt.show()
```

Рисунок 4.4 – алгоритм розрахунку важливості параметрів.

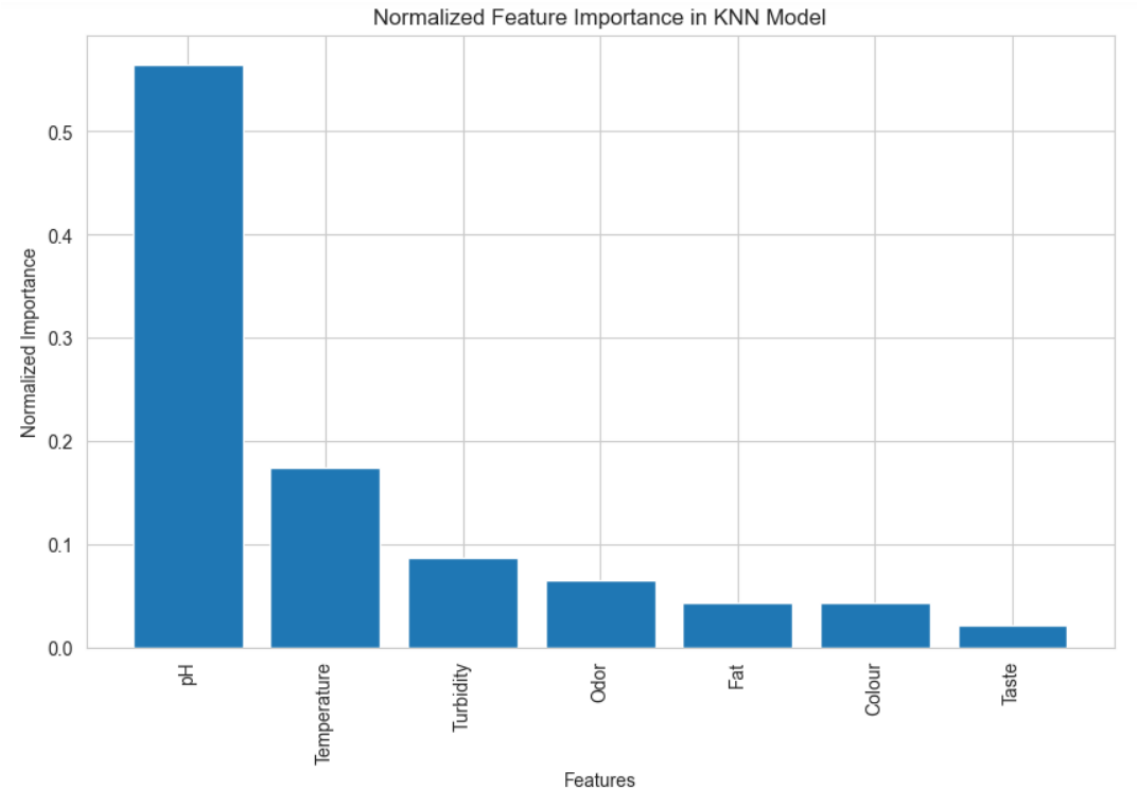


Рисунок 4.5 – діаграма важливості параметрів.

За даною діаграмою можна побачити, що з відривом найважливішим параметром є pH середовище, як і було помітно при візуальному аналізі розподілу параметрів. Вплив pH середовища на оцінку якості приблизно 0.55, наступним по важливості параметром є температура, яка також була помічена важливою при візуальному аналізі. Усі інші параметри дуже слабко впливають на результат, що дуже цікаво, бо здавалося б, що запах та смак мають бути найважливішими, а насправді це не так.

4.3 Аналіз отриманих результатів для методу Decision Tree Classifier

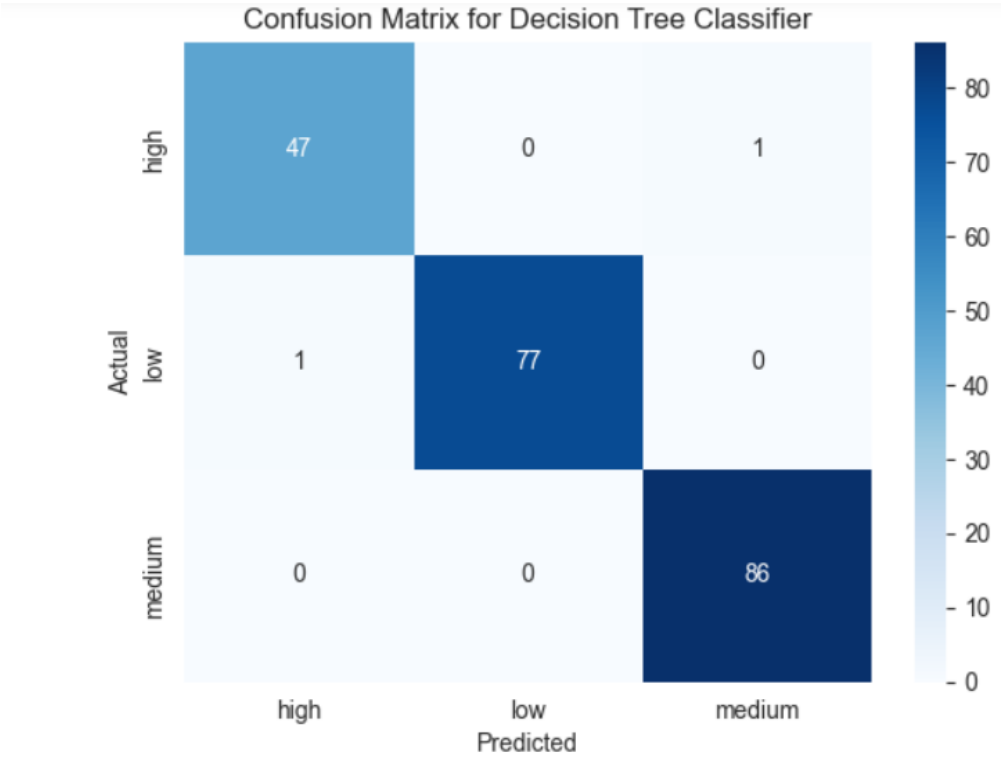
DTC метод не вимагає попереднього визначення додаткових параметрів, отже можна одразу приступати до навчання моделі та перевірки її точності на тренувальних та тестових даних (рис. 4.6). Можна помітити, що точність на тренувальних даних так само ідеальна, що свідчить про те, що модель ідеально натренована, на тестових даних точність співпадає з KNN методом, отже ми так само маємо дві помилки.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
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: 0.9905660377358491
```

Рисунок 4.6 – Тренування та перевірка точності моделі.

Для перевірки роботи моделі так само використаємо матрицю невідповідностей та звіт про класифікацію (рис. 4.7). З матриці так само видно, що модель майже ідеально визначила якість молока (значення головної діагоналі – правильно прогнозовані), даний метод так само має дві помилки, але вже інші: у першій молоко високої якості було названо середнім, у другій молоко низької якості було названо високоякісним. У цьому випадку модель допустила грубу помилку, назвавши низькоякісне молоко високоякісним, тому можна було б зробити висновок, що ця модель відпрацювала гірше, але ця помилка на рівні похибки, тому вони відпрацювали однаково добре. За f1-score модель найгірше відпрацювала з високоякісним молоком. В цілому цей метод також дуже гарно впорався з задачею.



Classification report for Decision Tree Classifier:

	precision	recall	f1-score	support
high	0.98	0.98	0.98	48
low	1.00	0.99	0.99	78
medium	0.99	1.00	0.99	86
accuracy			0.99	212
macro avg	0.99	0.99	0.99	212
weighted avg	0.99	0.99	0.99	212

Рисунок 4.7 – Матриця невідповідностей та звіт про класифікацію.

Побудуємо стовпчикову діаграму з важливістю параметрів (рис. 4.8) для прогнозування методом DTC.

```
dtree_importance = dtc.feature_importances_
sorted_indices = np.argsort(dtree_importance)[::-1]
sorted_importance = dtree_importance[sorted_indices]

feature_labels = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour']

plt.figure(figsize=(10, 6))
plt.bar(range(len(dtree_importance)), sorted_importance, align='center')
plt.xticks(range(len(dtree_importance)), [feature_labels[i] for i in sorted_indices], rotation=45)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Decision Tree Classifier')
plt.tight_layout()
plt.show()
```

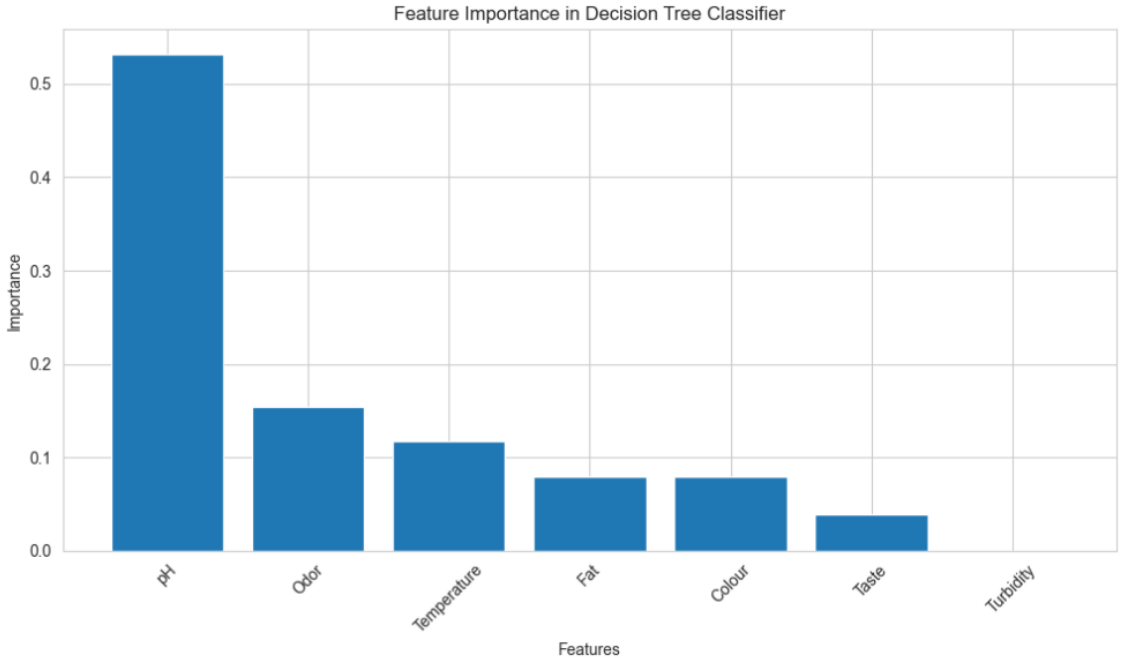


Рисунок 4.8 – Стовпчикова діаграма з важливістю параметрів.

З даної діаграми видно, що даний метод трошки інакше розставив важливість параметрів, але основний параметр не змінився, pH середовище так само впливає на результат більше, ніж на половину, а ось температура поступилась запаху, хоча в KNN запах був значно менш важливим параметром. Каламутність же взагалі не впливає на фінальний результат. Загальна тенденція сильно не змінилась.

4.4 Аналіз отриманих результатів для методу Random Forest Classifier

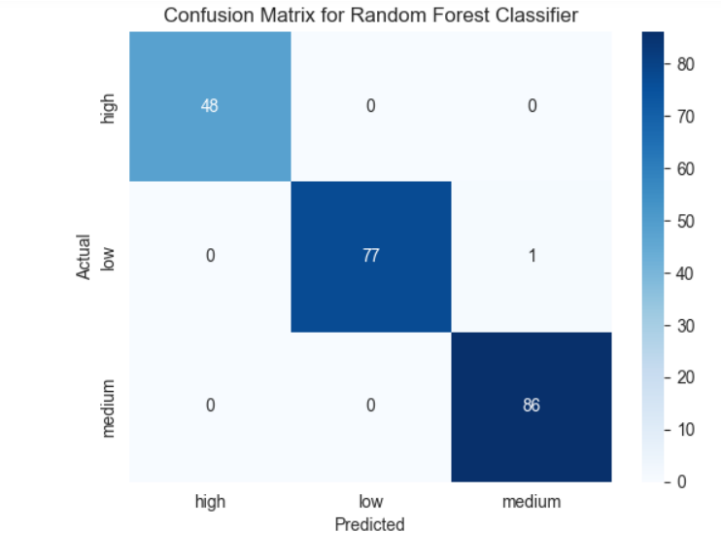
RFC так само, як і DTC не потребує визначення додаткових параметрів перед навчанням моделі, тому одразу перейдемо до навчання моделі та перевірки її точності на тренувальних та тестових даних (рис. 4.9). Одразу ж можна помітити, що на тестових даних цей метод спрацював краще за попередні, і це цілком очікувано. Дана модель не змогла правильно спрогнозувати лиш одне значення.

```
import numpy as np
import matplotlib.pyplot as plt
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: 1.0
[RFC] Test Accuracy: 0.9952830188679245
```

Рисунок 4.9 – Тренування та перевірка точності моделі.

Для перевірки роботи моделі так само використаємо матрицю невідповідностей та звіт про класифікацію (рис. 4.10). На головній діагоналі можна побачити правильно визначені екземпляри, та лиш один екземпляр, що має низьку якість, було спрогнозовано як молоко середньої якості. Тобто модель трошки завищила якість цього екземпляру. Отже, метод RFC відпрацював найкраще, цього варто було очікувати, але він так само використовує найбільше ресурсів для прогнозування, тож різниця у 0.5% я би не вважав справді виправданою. За f1-score цей метод ідеально відпрацював з молоком високої якості, а з двома іншими класами трошки гірше, але це все одно результат близький до ідеального.



Classification report for Random Forest Classifier:

	precision	recall	f1-score	support
high	1.00	1.00	1.00	48
low	1.00	0.99	0.99	78
medium	0.99	1.00	0.99	86
accuracy			1.00	212
macro avg	1.00	1.00	1.00	212
weighted avg	1.00	1.00	1.00	212

Рисунок 4.10 – Матриця невідповідностей та звіт про класифікацію.

Побудуємо стовпчикову діаграму з важливістю параметрів (рис. 4.11) для прогнозування методом RFC.

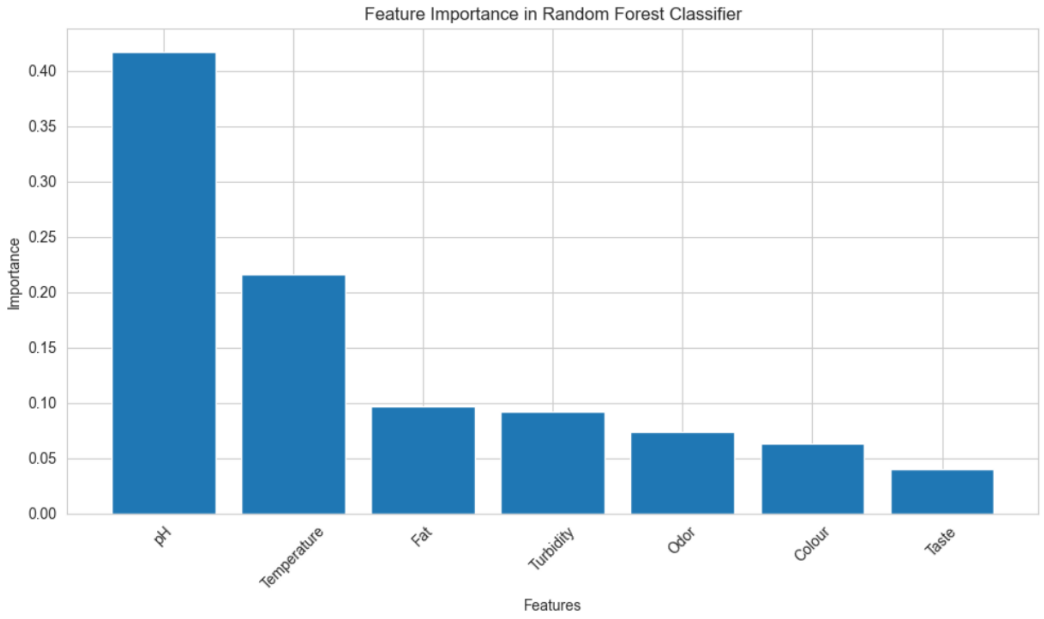


Рисунок 4.11 – Стовпчикова діаграма з важливістю параметрів.

Дана діаграма досить схожа на діаграму метода KNN, тільки pH середовище має менше впливу на результат, а всі інші параметри впливають трошки більше. Тільки от параметри, які на першу думку здаються найбільш важливими (смак та запах), так само мають дуже маленький вплив на якість молока, особливо смак.

4.5 Порівняння отриманих результатів

За результатами вище можна порівняти між собою дані методи. Найкращу точність показав Random Forest Classifier, але його точність виявилась лиш на 0.5% краще за конкурентні методи K-Nearest Neighbors та Decision Tree Classifier, в цілому всі методи показали дуже гарний результат, але RFC використовує значно більше ресурсів для прогнозування, тому я не вважаю різницю в 0.5% виправданою, оскільки два інших методи працюють швидше і використовують менше ресурсів. Усі методи приблизно схожим чином розцінюють важливість параметрів для прогнозування, і без сумнівів pH середовище є найбільш важливим параметром. Я вважаю, що найбільш доцільно використовувати Decision Tree Classifier, оскільки він показав гарні результати точності, споживає не так багато ресурсів та гарно справляється з більш великими вибірками, на відміну від KNN.

ВИСНОВКИ

В результаті виконання курсової роботи було розбірно досить важливу тему, що стосується базового продукту харчування, а отже кожного. Задачею роботи було прогнозування якості молока методами K-Nearest Neighbors, Random Forest Classifier, Decision Tree Classifier. Для реалізації поставленої задачі було використано мову програмування Python та бібліотеки, що належать їй: seaborn[3], matplotlib[2], pandas[1], sklearn[4], numpy.

На основі детального опису, проведеного аналізу предметної області та інтелектуального аналізу даних для прогнозування якості молока було отримано результати, на основі яких було визначено найкращий метод для прогнозування в даному випадку. Результати досліджень показують, що усі методи справились з роботою дуже гарно, найбільш точним опинився Random Forest Classifier, а найбільш ефективним Decision Tree Classifier, оскільки він показав майже такі ж результати, але застосував менше ресурсів, також цей метод буде достатньо ефективним у випадку збільшення обсягу вибірки.

Отже, поставлені задачі були виконані, було зроблено певні висновки та обрано найкращий метод для подальшого прогнозування якості молока – Decision Tree Classifier.

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)

36 арк, 1.37 Мб

(Обсяг програми (документа), арк.,

студента групи ІП-13 ІІ курсу

Бондаренка М.В.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
####
df = pd.read_csv('dataset/milknew.csv')
print(df.head())
####
print(df.info())
####
df = df.rename(columns={'Temprature': 'Temperature'})
df = df.rename(columns={'Fat ': 'Fat'})
print(df.columns)
####
print(df.describe())
####
sns.countplot(x="Grade", data=df, palette="Blues")
plt.show()
####
sns.set_style("whitegrid")

for parameter in df.columns[:-1]:
    fig, axs = plt.subplots(figsize=(13, 7), nrows=1, ncols=3, sharey=True)
    fig.suptitle('Distribution of {} by Milk Grade'.format(parameter))
```

```
    for i, grade in enumerate(['low', 'medium', 'high']):
        data_grade = df[df['Grade'] == grade]
        sns.histplot(data=data_grade[parameter], kde=True, ax=axs[i])
        axs[i].set_xlabel('Value')
        axs[i].set_ylabel('Frequency')
        axs[i].set_title(grade.capitalize())

    plt.tight_layout()
    plt.show()
####
X = df.drop('Grade', axis=1)
y = df['Grade']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
####
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)
####
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)
####
param_grid = {'n_neighbors': range(1, 10)}

knn = KNeighborsClassifier()

grid_search = GridSearchCV(knn, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_k = grid_search.best_params_['n_neighbors']

print("The best k value:", best_k)
####
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)
####
```

```
dtree_importance = dtc.feature_importances_
sorted_indices = np.argsort(dtree_importance)[::-1]
sorted_importance = dtree_importance[sorted_indices]

feature_labels = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour']

plt.figure(figsize=(10, 6))
plt.bar(range(len(dtree_importance)), sorted_importance, align='center')
plt.xticks(range(len(dtree_importance)), [feature_labels[i] for i in sorted_indices],
rotation=45)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Decision Tree Classifier')
plt.tight_layout()
plt.show()
####
rfc_importance = rfc.feature_importances_

sorted_indices = np.argsort(rfc_importance)[::-1]
sorted_importance = rfc_importance[sorted_indices]

plt.figure(figsize=(10, 6))
plt.bar(range(len(rfc_importance)), sorted_importance, align='center')
plt.xticks(range(len(rfc_importance)), [feature_labels[i] for i in sorted_indices],
rotation=45)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Random Forest Classifier')
plt.tight_layout()
plt.show()
####
```

```
feature_importance = {}
for feature in X_train.columns:
    X_train_subset = X_train.drop(feature, axis=1)
    X_val_subset = X_test.drop(feature, axis=1)

    knn_subset = KNeighborsClassifier()
    knn_subset.fit(X_train_subset, y_train)
    y_pred_subset = knn_subset.predict(X_val_subset)
    accuracy_subset = accuracy_score(y_test, y_pred_subset)

    feature_importance[feature] = accuracy_score(test_prediction_knn, y_test) -
accuracy_subset

sorted_importance = sorted(feature_importance.items(), key=lambda x: x[1],
reverse=True)

features = [item[0] for item in sorted_importance]
importance_values = [item[1] for item in sorted_importance]

total_importance = sum(importance_values)

normalized_importance = [importance / total_importance for importance in
importance_values]

plt.figure(figsize=(10, 6))
plt.bar(features, normalized_importance)
plt.xlabel('Features')
plt.ylabel('Normalized Importance')
plt.title('Normalized Feature Importance in KNN Model')
plt.xticks(rotation=90)
plt.show()
```

```
####
models = [dtc, rfc, knn]
names = ['Decision Tree Classifier', 'Random Forest Classifier', 'K-
NearestNeighbors']
class_labels = ['high', 'low', 'medium']

for model, name in zip(models, names):
    cm = confusion_matrix(y_true=y_test, y_pred=model.predict(X_test))
    cm_df = pd.DataFrame(cm, index=class_labels, columns=class_labels)
    sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues')

    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix for {name}')
    plt.show()

    report = classification_report(y_test, model.predict(X_test))
    print(f'Classification report for {name}:', report, sep='\n')
```