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: «Класифікація біопсій молочної залози:
злоякісні чи доброякісні пухлини»

Студента 2-го курсу групи ІП-13
Спеціальності: 121
«Інженерія програмного забезпечення»
Ал Хадама Мурата Резговича

«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

Ал Хадама Мурата Резговича

1.Тема роботи «Класифікація біопсій молочної залози: злоякісні чи доброякісні пухлини»

2.Строк здачі студентом закінченої роботи 09.06.2023

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/utkarshx27/breast-cancer-wisconsin-diagnostic-dataset

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	02.06.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	03.06.2023	
5.	Обґрунтування методів інтелектуального аналізу даних	04.06.2023	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	04.06.2023	
7.	Підготовка пояснювальної записки	05.06.2023	
8.	Здача курсової роботи на перевірку	08.06.2023	
9.	Захист курсової роботи	08.06.2023	

Студент _____ Ал Хадам М.Р.

(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)

"8" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 40 сторінок, 26 рисунків, 8 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних для задачі бінарної класифікації біопсії молочної залози.

Предмет дослідження: моделі для вирішення задач бінарної класифікації, такі як: K-Nearest Neighbors, Logistic Regression та Decision Tree Classifier та методи для аналізу, графічного відображення та прогнозування даних.

Мета роботи: створення програмного забезпечення, яке здатне класифікувати біопсії молочної залози на злоякісні та доброякісні пухлини, використовуючи різні моделі та методи.

Дана курсова робота включає в себе: збір та підготовка даних для аналізу, очищення даних для подальшого використання, вибір та побудова різних моделей для класифікації, оцінка та порівняння результатів цих моделей.

МОДЕЛЬ ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, K-NEAREST NEIGHBORS, LOGISTIC REGRESSION, DECISION TREE CLASSIFIER.

ЗМІСТ

ВСТУП

Діагностика та класифікація пухлин молочної залози є важливою задачею в онкологічній медицині. Від правильного розпізнавання злоякісних та доброякісних пухлин залежить вчасне та ефективне лікування пацієнтів. Застосування комп'ютерних алгоритмів та методів машинного навчання може значно полегшити цей процес та допомогти лікарям у прийнятті рішень.

У рамках даної курсової роботи проаналізовано дані показників біопсій молочних залоз за тридцятьма різними характеристиками. Ознаки були отримані обчислювальним шляхом з цифрових зображень мазків тонкоголкової аспіраційної біопсії. Ознаки відповідають властивостям клітинних ядер, таким як розмір, форма та регулярність.

Для інтелектуального аналізу було використано три методи для прогнозування злоякісності новоутворень.

У даній курсовій роботі було використано дані технології: Python, Pandas[1], Matplotlib[2], Seaborn[3], Sklearn[4], NumPy[5].

1.ПОСТАНОВКА ЗАДАЧІ

Виконання курсової роботи включає такі етапи: проведення аналізу предметної області; завантаження, опис і обробка даних; проведення первинного аналізу даних; розподіл даних для навчання моделі; вибір методів прогнозування; аналіз та порівняння результатів кожного методу.

Для проведення даного дослідження було попередньо поділено отримані дані на тренувальні та тестові набори для моделей.

Прогнозування виконується за допомогою методів K-Nearest Neighbors[6], Logistic Regression[7], Decision Tree Classifier[8] – усе це методи для вирішення задачі класифікації. Потрібно проаналізувати та порівняти результати виконання кожного методу, що дозволить обрати найбільш ефективний.

Ознаки, що використовуються, отримані шляхом обчислення з цифрових зображень екземплярів тонкоголкової аспіраційної біопсії, виступають в якості вхідних даних.

Створенні моделі машинного навчання, які є результатом дослідження можна використовувати для прогнозування злоякісності новоутворень за результатами біопсії молочних залоз.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Молочна залоза - важлива частина жіночої анатомії, яка забезпечує продукцію та виділення молока. Хоча більшість пухлин, що формуються в молочній залозі, є доброякісними, виявлення злоякісних пухлин, таких як рак молочної залози, є серйозною загрозою для жіночого здоров'я. Визначення характеру пухлини є критичним для прийняття вирішальних рішень щодо лікування та подальшого управління хворобою. Сучасні методи діагностики, зокрема біопсія молочної залози, надають можливість отримати тканинний матеріал для детального аналізу та встановлення діагнозу.

Одним із способів виявлення та класифікації пухлин молочної залози є біопсія тонким голками. У процесі біопсії отримуються клітини або тканинні зразки з пухлини, які пізніше досліджуються під мікроскопом.

Задача класифікації біопсій молочної залози полягає в тому, щоб визначити, чи є пухлина злоякісною чи доброякісною на основі аналізу отриманих даних. Це може бути складним завданням, оскільки вимагає врахування різноманітних ознак та характеристик клітин або тканин.

Остаточне визначення характеру пухлини молочної залози є вирішальним для встановлення діагнозу та подальшого лікування пацієнтів. Правильна класифікація пухлини дозволяє призначити оптимальну терапію, забезпечити пацієнтові належне управління та покращити прогнози у випадку злоякісних пухлин. Тому подальші дослідження та розвиток методів класифікації є важливим напрямом у полі онкологічної діагностики.

У рамках розробки програмного забезпечення будуть впроваджені такі функції:

- завантаження вибірки даних для подальшої обробки;
- обробка та первинний аналіз завантажених даних;
- застосування інтелектуального аналізу даних ;
- використання трьох моделей прогнозування даних;
- прогнозування характеру новоутворень;
- візуалізація отриманих результатів з висновками дослідження.

3.РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Було обрано набір даних під назвою «Breast Cancer Wisconsin Diagnostic Dataset», що складається з 569 результатів біопсії молочних залоз. Датасет складається з таблиці, що містить 32 стовпця. Стовпці несуть дану інформацію:

Таблиця 3.1 – Опис стовпців вхідного датасету

Назва	Опис значення
x.radius_mean	Середній радіус клітин пухлини
x.texture_mean	Середня текстура клітин пухлини
x.perimeter_mean	Середній периметр клітин пухлини
x.area_mean	Середня площа клітин пухлини
x.smoothness_mean	Середня гладкість клітин пухлини
x.compactness_mean	Середня компактність клітин пухлини
x.concavity_mean	Середня вогнутість клітин пухлини
x.concave_points_mean	Середня кількість вогнутих частин контуру клітин пухлини
x.symmetry_mean	Середня симетрія клітин пухлини
x.fractal_dimension_mean	Середня "приближення берегової лінії" клітин пухлини

x.radius_se	Стандартна похибка радіуса клітин пухлини
x.texture_se	Стандартна похибка текстури клітин пухлини
x.perimeter_se	Стандартна похибка периметра клітин пухлини
x.area_se	Стандартна похибка площі клітин пухлини
x.smoothness_se	Стандартна похибка гладкості клітин пухлини
x.compactness_se	Стандартна похибка компактності клітин пухлини
x.concavity_se	Стандартна похибка вогнутості клітин пухлини
x.concave_points_se	Стандартна похибка кількості вогнутих частин контуру клітин пухлини
x.symmetry_se	Стандартна похибка симетрії клітин пухлини
x.fractal_dimension_se	Стандартна похибка "приближення берегової лінії" клітин пухлини
x.radius_worst	Найгірший (найбільший) радіус клітин пухлини
x.texture_worst	Найгірша (найважча) текстура клітин пухлини

x.perimeter_worst	Найгірший (найбільший) периметр клітин пухлини
x.area_worst	Найгірша (найбільша) площа клітин пухлини
x.smoothness_worst	Найгірша (найважча) гладкість клітин пухлини
x.compactness_worst	Найгірша (найважча) компактність клітин пухлини
x.concavity_worst	Найгірша (найважча) вогнутість клітин пухлини
x.concave_points_worst	Найгірша (найважча) кількість вогнутих частин контуру клітин пухлини
x.symmetry_worst	Найгірша (найважча) симетрія клітин пухлини
x.fractal_dimension_worst	Найгірша (найважча) "приближення берегової лінії" клітин пухлини
y.target	Цільова змінна

3.2 Перевірка та первинний аналіз даних

Зчитаємо дані з файлу в структуру DataFrame модуля Pandas, виведемо перші 5 рядків датафрейму, щоб перевірити коректність імпорту даних, та основну інформацію (рис. 3.2) про нього. Попередньо імпортуємо всі необхідні пакети для роботи (рис 3.1).

Ввод [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from datetime import datetime

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report

import warnings
from sklearn.exceptions import FitFailedWarning, ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)
warnings.filterwarnings("ignore", category=FitFailedWarning)
```

Рисунок 3.1 – Імпортування необхідних пакетів.

Ввод [2]:
```
brca = pd.read_csv('input/brca.csv')
brca
```

Out[2]:

	Unnamed: 0	x.radius_mean	x.texture_mean	x.perimeter_mean	x.area_mean	x.smoothness_mean	x.compactness_mean	x.concavity_mean	x.concave_pts_me
0	1	13.540	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.047
1	2	13.080	15.71	85.63	520.0	0.10750	0.12700	0.04568	0.031
2	3	9.504	12.44	60.34	273.9	0.10240	0.06492	0.02956	0.020
3	4	13.030	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.029
4	5	8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005
...	...	...	...	...	...	...	...	...	
564	565	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.147
565	566	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.138
566	567	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.097
567	568	16.600	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.053
568	569	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.152

569 rows × 32 columns

Рисунок 3.2 – Загальний вигляд вхідного датасету.

На даному етапі можна помітити, що дані було зчитано коректно, але є проблема зі створенням зайвого стовпця індексів “Unnamed: 0” та з нерепрезентативною назвою стовпців вхідних характеристик з початком вигляду “x.” (рис. 3.3), одразу виправимо це.

Ввод [3]:
```
brca = brca.drop('Unnamed: 0', axis=1)

for col in brca.columns:
    if col.startswith('x.'):
        new_col = col[2:]
        brca = brca.rename(columns={col: new_col})
```

Ввод [4]:
```
brca.columns
```

Out[4]:
```
Index(['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave_pts_mean', 'symmetry_mean', 'fractal_dim_mean', 'radius_se',
       'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave_pts_se', 'symmetry_se',
       'fractal_dim_se', 'radius_worst', 'texture_worst', 'perimeter_worst',
       'area_worst', 'smoothness_worst', 'compactness_worst',
       'concavity_worst', 'concave_pts_worst', 'symmetry_worst',
       'fractal_dim_worst', 'y'],
      dtype='object')
```

Рисунок 3.3 – Видалення стовпця індексів
та перейменування колонок.

Для загального розуміння кількості пропущених значень та інформації про кожну з колонок виведемо інформацію про виправлений датасет (рис. 3.4).

```
Ввод [5]: brca.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   radius_mean        569 non-null    float64
 1   texture_mean       569 non-null    float64
 2   perimeter_mean     569 non-null    float64
 3   area_mean          569 non-null    float64
 4   smoothness_mean    569 non-null    float64
 5   compactness_mean   569 non-null    float64
 6   concavity_mean     569 non-null    float64
 7   concave_pts_mean   569 non-null    float64
 8   symmetry_mean      569 non-null    float64
 9   fractal_dim_mean   569 non-null    float64
 10  radius_se          569 non-null    float64
 11  texture_se         569 non-null    float64
 12  perimeter_se       569 non-null    float64
 13  area_se            569 non-null    float64
 14  smoothness_se      569 non-null    float64
 15  compactness_se     569 non-null    float64
 16  concavity_se       569 non-null    float64
 17  concave_pts_se     569 non-null    float64
 18  symmetry_se        569 non-null    float64
 19  fractal_dim_se     569 non-null    float64
 20  radius_worst       569 non-null    float64
 21  texture_worst      569 non-null    float64
 22  perimeter_worst    569 non-null    float64
 23  area_worst         569 non-null    float64
 24  smoothness_worst   569 non-null    float64
 25  compactness_worst  569 non-null    float64
 26  concavity_worst    569 non-null    float64
 27  concave_pts_worst  569 non-null    float64
 28  symmetry_worst     569 non-null    float64
 29  fractal_dim_worst  569 non-null    float64
 30  y                  569 non-null    object
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

Рисунок 3.4 – Загальна інформація про датафрейм.

З інформації зрозуміло, що набір даних містить 30 колонок типу 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 569 записів, і всі колонки мають по 569 non-null значень, отже наш датасет не має пропусків та не вимагає додаткових перевірок на пусті значення.

Для полегшеного аналізу структури виведемо дані описової статистики датафрейму (рис. 3.5).

```
Ввод [6]: brca.describe()
```

Out[6]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_pts_mean	symmetry_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000

8 rows × 30 columns

Рисунок 3.5 – Виведення описової статистики.

Звідси можна побачити середні значення, стандартні відхилення, мінімальні значення, медіану, значення для кожного з квартилів та 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.

Для візуального розуміння впливу кожної з ознак на кінцевий результат є зміст побудувати гістограми розподілу (рис. 3.6) для кожного з стовпців предикторів в залежності від кінцевої характеристики пухлини.

Ввод [7]:
```
def histplots(df):
    n = 5
    m = 6
    col_num = 0

    fig, axes = plt.subplots(nrows=n, ncols=m, figsize=(15, 10))
    for i in range(n):
        for j in range(m):
                sns.histplot(df, x=df[df.columns[col_num]], ax=axes[i][j], kde=True, hue=df.columns[-1])
                col_num += 1

    plt.tight_layout()
    plt.show()


histplots(brca)
```

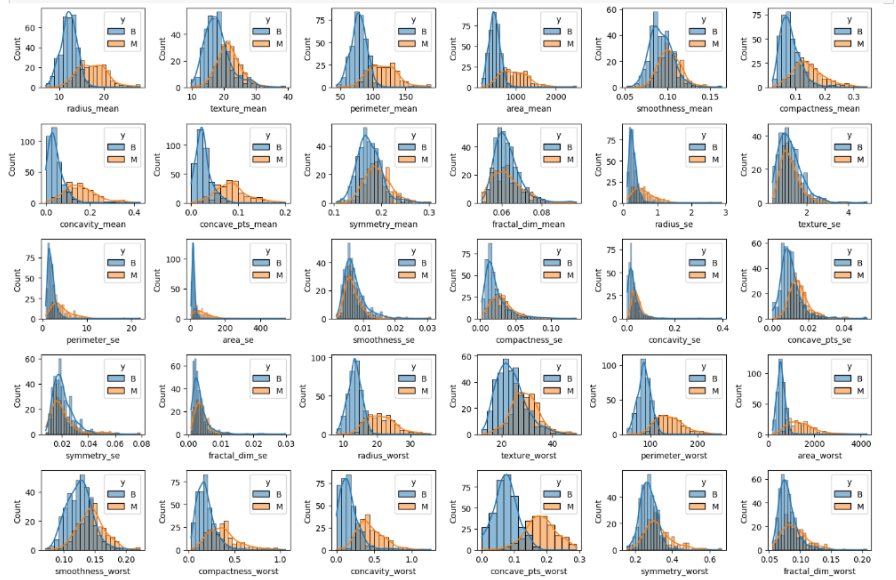

Рисунок 3.6 - Виведення гістограм розподілу для всіх стовпців.

На цьому етапі дослідження можемо звернути увагу на певні стовпці значення яких для доброякісних та злоякісних пухлин принципово відрізняється. Тобто найімовірніше вплив саме цих значень на кінцевий результат прогнозування є найбільш високим, потенційні колонки з найбільшим впливом: radius_mean, perimeter_mean, concave_pts_mean, radius_worst, perimeter_worst, concave_pts_worst.

Подивитись розподіл за кількістю злоякісних та доброякісних новоутворень у датасеті та візуалізуємо результати (рис. 3.7).

```
Ввод [8]: target = brca["y"].value_counts()
          target

Out[8]: y
        B    357
        M    212
        Name: count, dtype: int64

Ввод [9]: plt.pie(target, labels=target.index, autopct='%1.1f%%')
          plt.title('Mass is malignant("M") or benign("B")')
          plt.show()
```

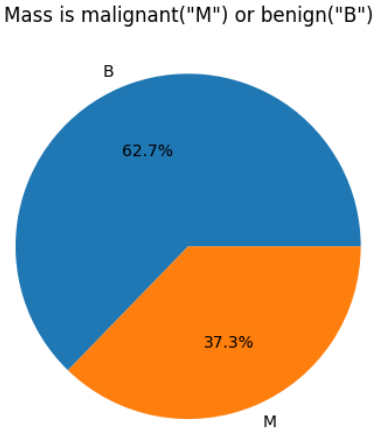

Рисунок 3.7 – Кількісний та відносний розподіл типів новоутворень.

Для більш точного ствердження з приводу факторів, які впливають найбільше на вирішення задачі класифікації є зміст вивести предиктори, значення кореляції з цільовою змінної є високим/дуже високим ($r_{xy} > 0.75$) (рис. 3.8, рис. 3.9).

```
Ввод [10]: brca_corr = brca.copy()
           brca_corr['y'] = brca_corr['y'].replace({'B': 0, 'M': 1})

           corr_coef = 0.75

           correlation_matrix = brca_corr.corr()
           high_correlation_cols = correlation_matrix[correlation_matrix['y'] > corr_coef].index

           plt.figure(figsize=(10, 8))
           sns.heatmap(correlation_matrix.loc[high_correlation_cols, high_correlation_cols], annot=True, cmap='coolwarm')

           plt.title(f'High Correlation Features (Coefficient > {corr_coef})')
           plt.show()

           print(f"Features with correlation coefficient > {corr_coef}:")
           for col in high_correlation_cols:
               if col != 'y':
                   print(col)
```

Рисунок 3.8 – Код для побудови матриці кореляційної залежності

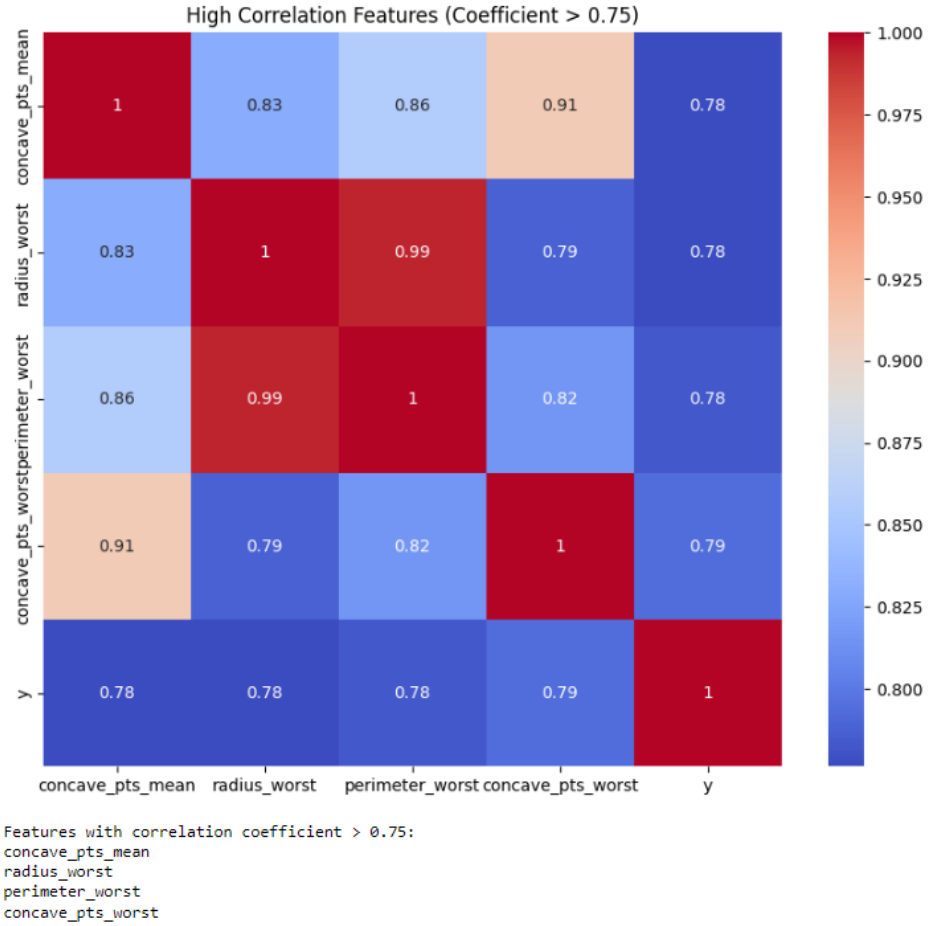

Рисунок 3.9 – Матриця кореляції

Припущення, раніше висунуте щодо прогнозування предикторів з найбільшим впливом на цільовий результат за виглядом розподілу в залежності від значення цільового стовпця, виявилось коректне, так як кожен з них має коефіцієнт кореляції з цільовою змінною більше 0.75.

3.3 Поділ даних

На даному етапі потрібно розділити дані на тестові та тренувальні за нестандартним співвідношенням, так як кількість даних є відносно невеликою, тому 80% даних виділимо на тренувальний набір (рис. 3.10).

```
Ввод [11]: X = brca.drop('y', axis=1)
           y = brca['y']

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

Рисунок 3.10 – Поділ даних на тестові та тренувальні.

Тренувальні дані використовуються для навчання самих моделей, а на тестовому наборі будемо перевіряти точність самої моделі, оскільки це дані, які модель не бачила при навчанні.

Абсолютна кількість тренувальних та тестових записів за класами цільового стовпця зображено на рисунку 3.11.

```
Ввод [12]: y_train.value_counts()

Out[12]: y
         B    286
         M    169
         Name: count, dtype: int64

Ввод [13]: y_test.value_counts()

Out[13]: y
         B    71
         M    43
         Name: count, dtype: int64
```

Рисунок 3.11 – Кількість записів для тренувального та тестового наборів

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Було обрано три методи для виконання поставленої задачі: K-Nearest Neighbors (KNN), Logistic Regression (LG) та Decision Tree Classifier (DTC) – усе це методи класифікації, оскільки наша задача полягає в оцінці характеристики новоутворень, які ділиться на два класи: злоякісні та доброякісні.

K-Nearest Neighbors (KNN) є методом класифікації, який визначає клас нового прикладу, порівнюючи його зі значенням гіперпараметру K-найближчими сусідами у навчальному наборі даних. Основна перевага KNN полягає в його простоті реалізації та можливості працювати з будь-яким типом ознак. Він може показати добрі результати, коли дані мають чітко виділені класи, у нашому випадку бінарно обрані, та коли кількість ознак невелика. Однак, KNN має деякі недоліки, зокрема вимагає збереження всього навчального набору даних у пам'яті та обчислювальну складність для пошуку найближчих сусідів у великому наборі даних. Більш того, під час підбору найоптимальнішого значення гіперпараметру треба врахувати багато моментів: не обрати мале значення сусідів, так як викиди в даних зможуть суттєво впливати на результат дослідження або упередити себе від обрання занадто велике значення гіперпараметру, в такому випадку значення точності моделі не буде змінюватися, а обчислювальна складність буде навпаки рости. Варто зауважити, що при виборі великого значення сусідів для KNN вплив менш численних класів буде прагнути до 0.

Logistic Regression (LR) є методом класифікації, який використовує логістичну функцію для визначення ймовірності належності прикладу до певного класу. Його перевагою є легкість інтерпретації та здатність робити ймовірнісні прогнози. LR показує хорошу ефективність, коли дані мають лінійну роздільну здатність та коли немає великої кількості шуму в даних, для нашого випадку ідеального підходить. Однак, цей метод може бути обмежений в складних нелінійних взаємозв'язках та вимагає попередньої обробки даних для роботи з категоріальними ознаками.

Decision Tree Classifier (DTC) є потужним методом класифікації, який може бути застосований і до задачі класифікації біопсії молочних залоз. Завдяки легкої реалізації: дерево рішень забезпечує легку інтерпретацію, оскільки його структура подібна до логічних правил. Можна легко зрозуміти, які ознаки відіграють важливу роль у прийнятті рішень і які шляхи ведуть до класифікації. В межах цієї роботи є доцільно використовувати алгоритм з репрезентацією дерева рішень, так як вони можуть працювати з різними типами ознак, включаючи категоріальні та числові. Це робить їх універсальними для вирішення задач, де дані можуть мати різноманітну природу. З урахуванням переваг та недоліків DTC для задачі класифікації біопсії молочних залоз, можна сказати, що він є потенційно ефективним методом, проте, слід бути обережним з перенавчанням та враховувати можливу чутливість моделі до незначних змін у даних.

Основними відмінностями між методами є різні підходи до класифікації. Так, наприклад, KNN використовує відстань до найближчих сусідів для класифікації, LR використовує логістичну функцію, а DTC використовує дерево рішень, яке розділяє дані на основі умов. Для наших даних малої розмірності гарно підійдуть LR та DTC моделі, оскільки вони гарно та швидко працюють з такою кількістю даних з явно задними числовими параметрами.

4.2 Аналіз отриманих результатів для методу K-Nearest Neighbors

Перед навчанням моделі треба попередньо обрати модель за найкращим значенням гіперпараметру K-сусідів за метрикою точності, в нашому випадку найкращим параметром K виявилось 8 (рис. 4.1), тож використаємо цю модель.

KNeighborsClassifier

Ввод [14]:
```
knn_classificator = KNeighborsClassifier()
knn_parameters = {
    'n_neighbors': range(1, 15)
}

knn_grid_search = GridSearchCV(knn_classificator, knn_parameters, cv=5, scoring='accuracy')
knn_grid_search.fit(X_train, y_train)

print("Best value for 'n_neighbors' hyperparameter is", knn_grid_search.best_params_['n_neighbors'])
```

```
Best value for 'n_neighbors' hyperparameter is 8
```

Рисунок 4.1 – Підбір найкращого гіперпараметру K.

Виведемо результати даної моделі на тестовому та тренувальному наборі на рисунку 4.2.

```
Ввод [15]: knn = knn_grid_search.best_estimator_

knn_test_acc = knn.score(X_test, y_test)
print("Test accuracy: ", knn_test_acc)

knn_train_acc = knn.score(X_train, y_train)
print("Train accuracy: ", knn_train_acc)

Test accuracy:  0.9385964912280702
Train accuracy:  0.9340659340659341
```

Рисунок 4.2 – Перевірка точності моделі KNN.

Ми маємо майже ідеальну точність для тестових даних, що свідчить про те, що модель відносно ідеально обраховує все. Така висока точність обумовлена оптимальним значенням сусідів 8, що відхиляється від значення за замовчуванням (5) на 3 одиниці.

Один з методів перевірки продуктивності роботи моделі є матриця невідповідності. Побудуємо приклад такої матриці для результатів класифікації методом KNN (рис. 4.3).

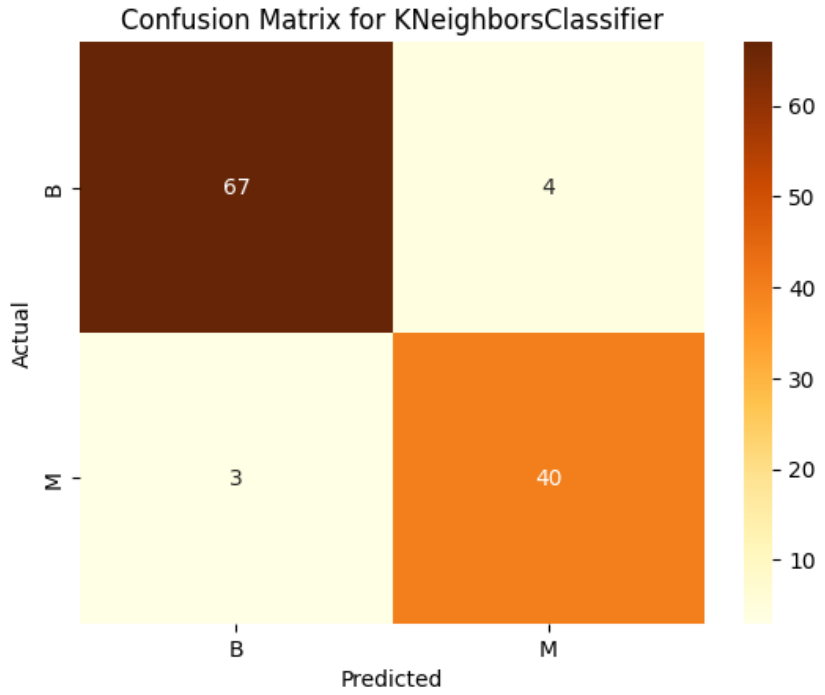

Рисунок 4.3 – Матриця невідповідності для KNN.

Таке зображення репрезентує фактичні значення вибірки у вигляді двовимірної таблиці, де по осі OY зображені фактичні значення класів, а по по OX – передбачені цільові значення обраним класифікатором.

В даному випадку можно побачити, що $67/71$ доброякісних пухлин були передбачені коректно, $4/67$ – помилково було класифіковані до злоякісних нашою моделю. З приводу іншого класу пухлин: $40/43$ всіх злоякісні пухлин

були класифіковані правильно, інші 3 помилково були віднесені до доброякісних.

Для повного розуміння результатів виведемо звіт для класифікації цим методом, завдяки якому дізнаємося значення метрик precision, recall, f1-score для двох класів (рис 4.4).

Classification report for KNeighborsClassifier:

	precision	recall	f1-score	support
B	0.96	0.94	0.95	71
M	0.91	0.93	0.92	43
accuracy			0.94	114
macro avg	0.93	0.94	0.93	114
weighted avg	0.94	0.94	0.94	114

Рисунок 4.4 – звіт класифікації для KNN моделі.

Для класу "B" (доброякісні) precision (точність) складає 0.96, що означає, що 96% прикладів, які модель віднесла до класу "B", були коректно класифіковані. Для класу "M" (злоякісні) точність складає 0.91, що означає, що 91% прикладів, які модель віднесла до класу "M", були коректно класифіковані.

Для класу "B" recall (повнота) складає 0.94, що означає, що модель правильно виявила 94% всіх доброякісних прикладів. Для класу злоякісних пухлин повнота складає 0.93, що означає, що модель правильно виявила 93% всіх злоякісних екземплярів.

Для класу доброякісних пухлин F1-оцінка становить 0.95, для злоякісних - 0.92. F1-оцінка є гармонічним середнім між точністю і повнотою і враховує обидві ці метрики.

4.3 Аналіз отриманих результатів для методу Logistic Regression

Перед навчанням другої моделі треба було також попередньо обрати модель за найоптимальнішим значенням максимальної кількості ітерацій за метрикою точності, алгоритмом жадібного пошуку було виявлено, що оптимальним значенням буде 10^4 ітерацій (рис. 4.5).

LogisticRegression

Ввод [16]:
```
logr_classificator = LogisticRegression()
logr_parameters = {
    'max_iter': [10**i for i in range(1, 5)]
}

logr_grid_search = GridSearchCV(logr_classificator, logr_parameters, cv=5, scoring='accuracy')
logr_grid_search.fit(X_train, y_train)

print("Best value for 'max_iter' hyperparameter is", logr_grid_search.best_params_['max_iter'])
```

```
Best value for 'max_iter' hyperparameter is 10000
```

Рисунок 4.5 – Підбір найкращого значення max_iter.

Така висока максимальна кількість ітерацій може призвести до більшого часу виконання, у даному відносно малому наборі майже не впливає на швидкодію, але забезпечує достатньо часу для збіжності та отримання оптимальних параметрів.

Виведемо результати даної моделі на тестовому та тренувальному наборі на рисунку 4.6.

Ввод [17]:
```
logr = logr_grid_search.best_estimator_

logr_test_acc = logr.score(X_test, y_test)
print("Test accuracy: ", logr_test_acc)

logr_train_acc = logr.score(X_train, y_train)
print("Train accuracy: ", logr_train_acc)
```

```
Test accuracy:  0.9473684210526315
Train accuracy:  0.9648351648351648
```

Рисунок 4.6 – Перевірка точності моделі LR.

Знову таки маємо відносно ідеальну точність для тестових даних, що свідчить про те, що модель добре вирішує задачу.

Для оцінки продуктивності роботи моделі також побудуємо матрицю невідповідності (рис. 4.7).

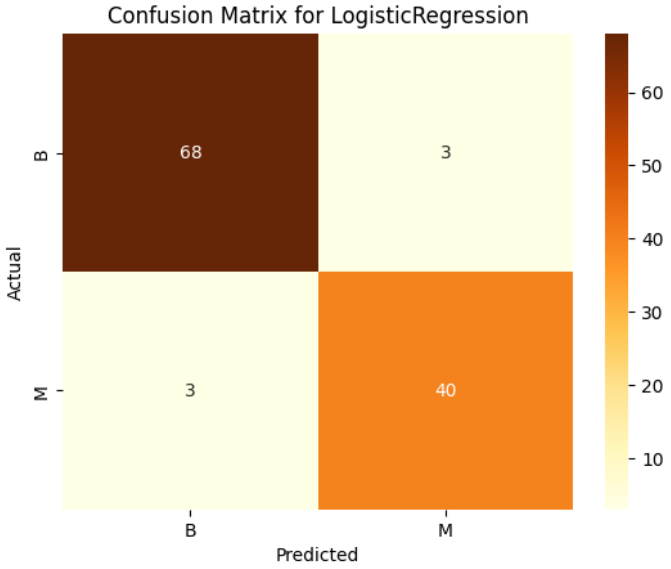

Рисунок 4.7 – Матриця невідповідності для LR

Для комплексного розуміння значень даної матриці виведемо звіт класифікації для даної моделі на рисунку 4.8.

Classification report for LogisticRegression:

	precision	recall	f1-score	support
B	0.96	0.96	0.96	71
M	0.93	0.93	0.93	43
accuracy			0.95	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.95	0.95	0.95	114

Рисунок 4.8 – звіт класифікації для моделі LR.

Для класу "B" (доброякісні) точність складає 0.96, що означає, що 96% прикладів, які модель віднесла до класу "B", були коректно класифіковані. Для класу "M" (злоякісні) точність складає 0.93, що означає, що 93% прикладів, які модель віднесла до класу "M", були коректно класифіковані.

Для класу "B" recall (повнота) складає 0.96, що означає, що модель правильно виявила 96% всіх доброякісних прикладів. Для класу злоякісних пухлин повнота складає 0.93, що означає, що модель правильно виявила 93% всіх злоякісних екземплярів.

Для класу доброякісних пухлин F1-оцінка становить 0.96, для злоякісних - 0.93. F1-оцінка є гармонічним середнім між точністю і повнотою і враховує обидві ці метрики.

Отримані значення показують, що модель LogisticRegression досягає високої точності, яка складає 0.95. Вона добре працює як для класу "B", так і для класу "M", з високими значеннями точності, повноти та F1-оцінки. Це означає, що модель здатна ефективно класифікувати доброякісні та злоякісні пухлини молочних залоз.

4.4 Аналіз отриманих результатів для методу Decision Tree Classifier

Перед навчанням третьої моделі завдяки класу з модуля Sklearn[4] знайдемо значення глибини дерева для якого значення точності досягає максимуму (рис. 4.9).

```
Ввод [18]: dtc_classificator = DecisionTreeClassifier()
           dtc_parameters = {
               'max_depth': [i for i in range(1, 10)]
           }

           dtc_grid_search = GridSearchCV(dtc_classificator, dtc_parameters, cv=5, scoring='accuracy')
           dtc_grid_search.fit(X_train, y_train)

           print("Best value for 'max_depth' hyperparameter is", dtc_grid_search.best_params_['max_depth'])

           Best value for 'max_depth' hyperparameter is 2
```

Рисунок 4.9 – Підбір найкращого значення max_depth.

Найкраща максимальна глубина – 2. Саме таке значення глибини є оптимальним, так зі збільшенням параметру точність не збільшується, а швидкодія погіршується. Також така маленька глибина пов’язана з тим, що маємо попередньо визначені предикатори з сильним рівнем зв’язку з цільовим значенням, адже саме вони будуть найбільше впливати на логіку класифікація певного екземляру і нашій моделі навіть значень лише цих параметрів буде достатньо для відносно успішній класифікації.

Виведемо результати даної моделі на тестовому та тренувальному наборі на рисунку 4.10.

```
Ввод [19]: dtc = dtc_grid_search.best_estimator_

           dtc_test_acc = dtc.score(X_test, y_test)
           print("Test accuracy: ", dtc_test_acc)

           dtc_train_acc = dtc.score(X_train, y_train)
           print("Train accuracy: ", dtc_train_acc)

           Test accuracy:  0.9210526315789473
           Train accuracy:  0.9516483516483516
```

Рисунок 4.10 – Перевірка точності моделі DTC.

Отримали 92% точності класифікації цим методом на тестових даних. Зумовлено це тим, що характеристики з малим рівнем кореляції з бажаним результатом на результат DTC не сильно впливають. Значення 95% для тестових також є нормальним, так як ця модель здатна до перенавчання і запам'ятовування конкретних екземплярів задачі.

Додатково візуалізуємо дерево рішень цього класифікатора нижче (рис 4.11). Всі виконання наступного коду будуть зберігатися в локальній директорії у вигляді зображень.

```
Ввод [20]: current_datetime = datetime.now()
timestamp_str = current_datetime.strftime("%Y-%m-%d_%H-%M-%S")

dpi = 750

plt.figure(figsize=(10, 8), dpi=dpi)
plot_tree(dtc, feature_names=brca.columns[:-1], class_names=target.index, filled=True)
filename = f'output/decision-tree/decisoin_tree_with_{dtc.max_depth}_max_depths_{timestamp_str}'
plt.savefig(filename)
plt.show()
```

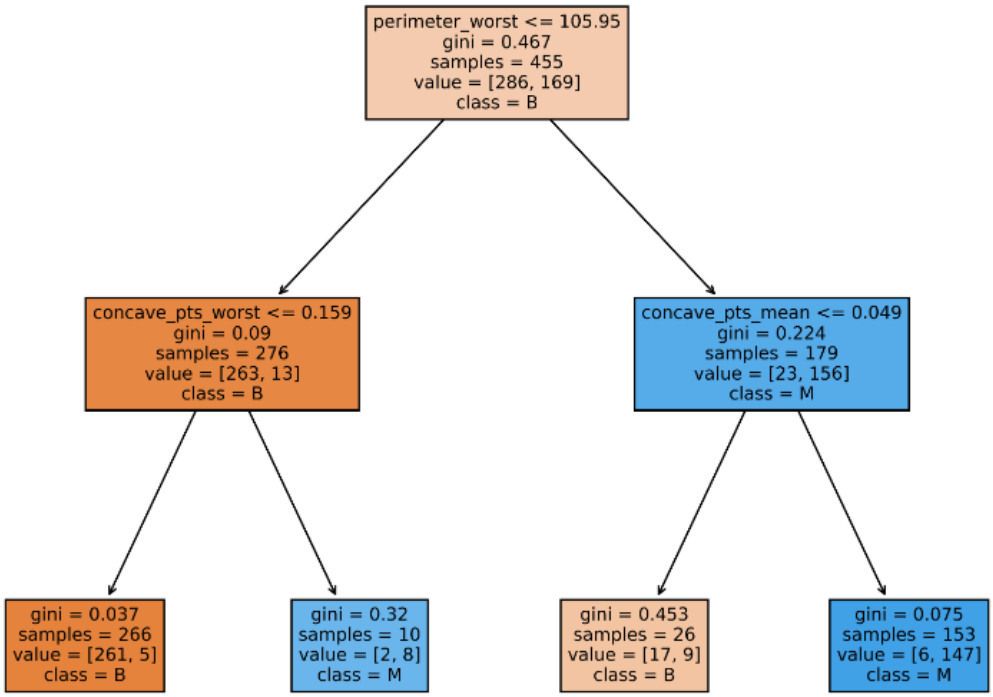

Рисунок 4.11 – Візуалізація DTC.

Щоб зрозуміти, які стовпці даних впливають найбільше на прийняття рішення моделлю побудуємо розподіл значущих предикторів на рисунку 4.12 та 4.13.

```
Ввод [21]: importances = dtc.feature_importances_

sorted_indices = np.argsort(importances)[::-1] #desc
sorted_importances = importances[sorted_indices]

nonzero_indices = np.nonzero(sorted_importances)
sorted_importances = sorted_importances[nonzero_indices]

sorted_features = brca.columns[:-1][sorted_indices][nonzero_indices]
```

Рисунок 4.12 – Фрагмент коду для побудови діаграми важливості предикторів моделі DTC.

```
Ввод [22]: plt.figure(figsize=(10, 8))
plt.bar(range(len(sorted_features)), sorted_importances, tick_label=sorted_features)

plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.xticks(rotation=70)

filename = f'output/decision-tree/feature_importance_{timestamp_str}'
plt.savefig(filename)

plt.show()
```

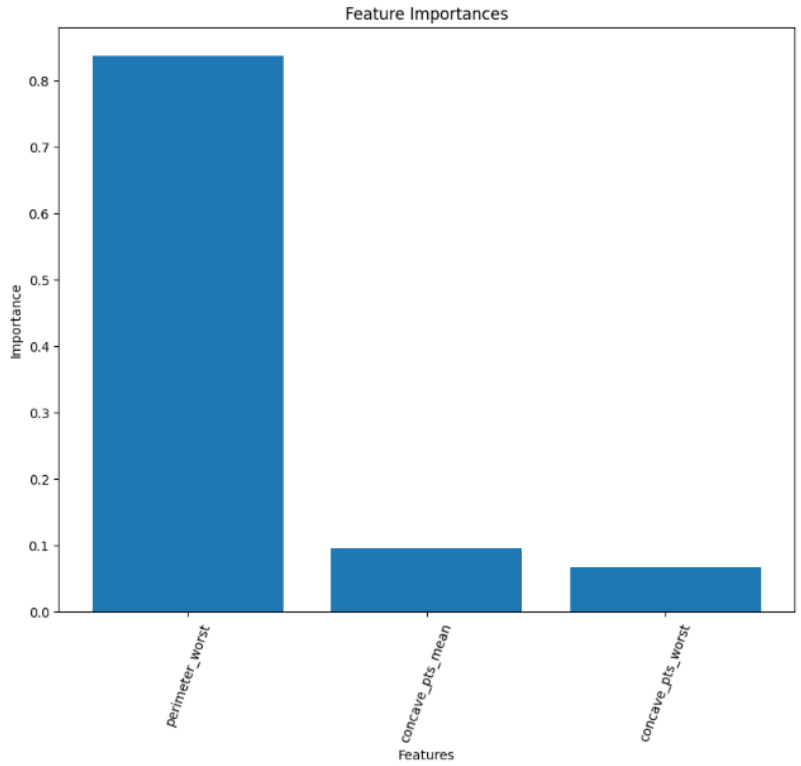

Рисунок 4.13 – діаграма важливості предикторів моделі DTC

Для оцінки продуктивності роботи моделі також побудуємо матрицю невідповідності (рис. 4.14).

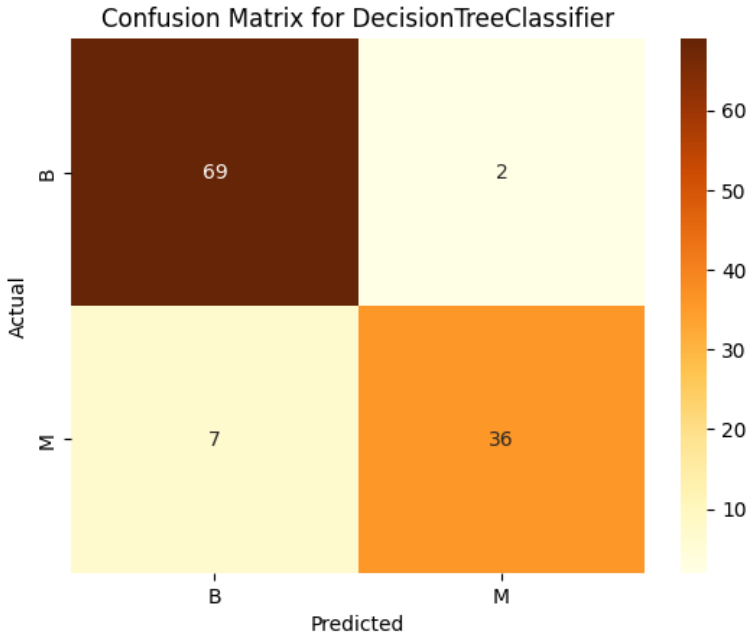

Рисунок 4.14 – Матриця невідповідності для DTC.

Для розуміння значень даної матриці виведемо звіт класифікації для даної моделі на рисунку 4.15.

Classification report for DecisionTreeClassifier:

	precision	recall	f1-score	support
B	0.91	0.97	0.94	71
M	0.95	0.84	0.89	43
accuracy			0.92	114
macro avg	0.93	0.90	0.91	114
weighted avg	0.92	0.92	0.92	114

Рисунок 4.15 – звіт класифікації для моделі DTC.

Для звіту класифікації моделі DTC опишимо значення далі.

Для класу "B" (доброякісні) точність складає 0.91, що означає, що 91% прикладів, які модель віднесла до класу "B", були коректно класифіковані. Для класу "M" (злоякісні) точність складає 0.95, що означає, що 95% прикладів, які модель віднесла до класу "M", були коректно класифіковані.

Для класу "B" recall (повнота) складає 0.97, що означає, що модель правильно виявила 97% всіх доброякісних прикладів. Для класу злоякісних пухлин повнота складає 0.84, що означає, що модель правильно виявила 84% всіх злоякісних екземплярів.

Для класу доброякісних пухлин F1-оцінка становить 0.94, для злоякісних - 0.89. F1-оцінка є гармонічним середнім між точністю і повнотою і враховує обидві ці метрики.

Отримані значення показують, що модель DecisionTreeClassifier досягає високої точності, яка складає 0.92. Вона добре працює для класу "B" з високими значеннями точності, повноти та F1-оцінки. Однак, для класу "M" її продуктивність не така висока, з показниками точності і повноти 0.95 і 0.84 відповідно. Це може вказувати на те, що модель має певні труднощі в класифікації злоякісних пухлин.

4.5 Порівняння отриманих результатів

За результатами вище можна порівняти між собою дані методи. Найкращу точність показала модель Logistic Regression, але її точність виявилась лиш на

1% та 3% краще за конкурентні методи K-Nearest Neighbors та Decision Tree Classifier відповідно, в цілому всі методи показали дуже гарний результат.

За результатами всіх тестувань KNN модель має схожі значення метрик оцінювання класифікації з LR, але все ж таки має на 1 % меншу точність .

Decision Tree Classifier має проблеми з коректною класифікацією злоякісних пухлин, що може фатально вплинути на подальший стан здоров’я потенційного пацієнта цієї моделі.

Загалом, з урахуванням всіх метрик, модель Logistic Regression виявляється найбільш оптимальною, оскільки вона досягає високих значень точності, повноти та F1-оцінки для обох класів (доброякісні та злоякісні пухлини). Вона є найбільш збалансованою моделлю з урахуванням передбачень для обох типів пухлин.

ВИСНОВКИ

У рамках даної курсової роботи було успішно виконано всі поставлені завдання та досягнуто мету дослідження. Основною метою було створення програмного забезпечення, яке здатне класифікувати біопсії молочної залози на злоякісні та доброякісні пухлини, використовуючи різні моделі та методи інтелектуального аналізу даних.

У процесі виконання курсової роботи були зібрані та підготовлені дані для аналізу, проведено очищення даних для подальшого використання. Для розв'язання задачі бінарної класифікації були вибрані та побудовані три моделі: K-Nearest Neighbors, Logistic Regression та Decision Tree Classifier.

Після побудови моделей було проведено оцінку та порівняння їх результатів. Аналізуючи звіти класифікації, можна зробити висновок, що всі три моделі показали хорошу точність і повноту для класу доброякісних пухлин. Однак, модель Logistic Regression виявилась найбільш оптимальною, оскільки вона досягла високих значень точності, повноти та F1-оцінки для обох класів (доброякісні та злоякісні пухлини). Вона є найбільш збалансованою моделлю з урахуванням передбачень для обох типів пухлин.

Таким чином, мета роботи була досягнута, а всі завдання, включаючи збір та підготовку даних, побудову моделей для класифікації, оцінку та порівняння результатів, були успішно виконані. Програмне забезпечення, розроблене в рамках цієї роботи, може бути використане для прогнозування класу біопсій молочної залози з високою точністю та надійністю, що важливо для підтримки медичних діагностичних процедур.

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. Документація бібліотеки Numpy. [Електронний ресурс] – URL: https://numpy.org/doc/
6. KNeighborsClassifier in Sklearn. [Електронний ресурс] – URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
7. LogisticRegression in Sklearn. [Електронний ресурс] – URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
8. 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

Класифікація біопсій молочної залози:

злоякісні чи доброякісні пухлини

(Найменування програми (документа))

SSD

(Вид носія даних)

40 арк, 1.57 Мб

(Обсяг програми (документа), арк.,

Кб)

студента групи ІП-13 ІІ курсу

Ал Хадама М.Р.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from datetime import datetime

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report

import warnings
from sklearn.exceptions import FitFailedWarning, ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)
warnings.filterwarnings("ignore", category=FitFailedWarning)


# In[2]:

brca = pd.read_csv('input/brca.csv')
brca
```

```
# In[3]:

brca = brca.drop('Unnamed: 0', axis=1)

for col in brca.columns:
    if col.startswith('x.'):
        new_col = col[2:]
        brca = brca.rename(columns={col: new_col})

# In[4]:

brca.columns

# In[5]:

brca.info()

# In[6]:

brca.describe()

# In[7]:

def histplots(df):
    n = 5
    m = 6
    col_num = 0

    fig, axes = plt.subplots(nrows=n, ncols=m, figsize=(15, 10))
    for i in range(n):
```

```
            for j in range(m):
                sns.histplot(df, x=df[df.columns[col_num]], ax=axes[i][j], kde=True,
hue=df.columns[-1])
                col_num += 1

        plt.tight_layout()
        plt.show()

histplots(brca)

# In[8]:

target = brca["y"].value_counts()
target

# In[9]:

plt.pie(target, labels=target.index, autopct='%1.1f%%')
plt.title('Mass is malignant("M") or benign("B")')
plt.show()

# In[10]:

brca_corr = brca.copy()
brca_corr['y'] = brca_corr['y'].replace({'B': 0, 'M': 1})

corr_coef = 0.75

correlation_matrix = brca_corr.corr()
high_correlation_cols = correlation_matrix[correlation_matrix['y'] >
corr_coef].index
```

```
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix.loc[high_correlation_cols,
high_correlation_cols], annot=True, cmap='coolwarm')

    plt.title(f'High Correlation Features (Coefficient > {corr_coef})')
    plt.show()

    print(f"Features with correlation coefficient > {corr_coef}:")
    for col in high_correlation_cols:
        if col != 'y':
            print(col)

    # In[11]:

    X = brca.drop('y', axis=1)
    y = brca['y']

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

    # In[12]:

    y_train.value_counts()

    # In[13]:

    y_test.value_counts()
```

```
# KNeighborsClassifier
# In[14]:

knn_classificator = KNeighborsClassifier()
knn_parameters = {
   'n_neighbors': range(1, 15)
}

knn_grid_search = GridSearchCV(knn_classificator, knn_parameters, cv=5, scoring='accuracy')
knn_grid_search.fit(X_train, y_train)

print("Best value for 'n_neighbors' hyperparameter is", knn_grid_search.best_params_['n_neighbors'])

# In[15]:

knn = knn_grid_search.best_estimator_

knn_test_acc = knn.score(X_test, y_test)
print("Test accuracy: ", knn_test_acc)

knn_train_acc = knn.score(X_train, y_train)
print("Train accuracy: ", knn_train_acc)

# LogisticRegression

# In[16]:

logr_classificator = LogisticRegression()
logr_parameters = {
```

```
    'max_iter': [10**i for i in range(1, 5)]
}

logr_grid_search = GridSearchCV(logr_classificator, logr_parameters, cv=5, scoring='accuracy')
logr_grid_search.fit(X_train, y_train)

print("Best value for 'max_iter' hyperparameter is", logr_grid_search.best_params_['max_iter'])

# In[17]:

logr = logr_grid_search.best_estimator_

logr_test_acc = logr.score(X_test, y_test)
print("Test accuracy: ", logr_test_acc)

logr_train_acc = logr.score(X_train, y_train)
print("Train accuracy: ", logr_train_acc)

# DecisionTreeClassifier

# In[18]:

dtc_classificator = DecisionTreeClassifier()
dtc_parameters = {
    'max_depth': [i for i in range(1, 10)]
}

dtc_grid_search = GridSearchCV(dtc_classificator, dtc_parameters, cv=5, scoring='accuracy')
```

```
dtc_grid_search.fit(X_train, y_train)

print("Best value for 'max_depth' hyperparameter is", dtc_grid_search.best_params_['max_depth'])

# In[19]:

dtc = dtc_grid_search.best_estimator_

dtc_test_acc = dtc.score(X_test, y_test)
print("Test accuracy: ", dtc_test_acc)

dtc_train_acc = dtc.score(X_train, y_train)
print("Train accuracy: ", dtc_train_acc)

# In[20]:

current_datetime = datetime.now()
timestamp_str = current_datetime.strftime("%Y-%m-%d_%H-%M-%S")

dpi = 750

plt.figure(figsize=(10, 8), dpi=dpi)
plot_tree(dtc, feature_names=brca.columns[:-1], class_names=target.index, filled=True)
filename = f'output/decision-tree/decisoin_tree_with_{dtc.max_depth}_max_depths_{timestamp_str}'
plt.savefig(filename)
plt.show()
```

```
# In[21]:

importances = dtc.feature_importances_

sorted_indices = np.argsort(importances)[::-1] #desc
sorted_importances = importances[sorted_indices]

nonzero_indices = np.nonzero(sorted_importances)
sorted_importances = sorted_importances[nonzero_indices]

sorted_features = brca.columns[:-1][sorted_indices][nonzero_indices]

# In[22]:

plt.figure(figsize=(10, 8))
plt.bar(range(len(sorted_features)), sorted_importances, tick_label=sorted_features)

plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.xticks(rotation=70)

filename = f'output/decision-tree/feature_importance_{timestamp_str}'
plt.savefig(filename)

plt.show()
```

```
# Confusion matrix for each model

# In[23]:

models = [knn, logr, dtc]
names = ['KNeighborsClassifier', 'LogisticRegression', 'DecisionTreeClassifier']
class_labels = ['B', 'M']

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

    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')
```