ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст. преподаватель |  |  |  | В.В. Боженко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| КЛАССИФИКАЦИЯ |
| по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4216 |  |  |  | С.Д. Комолова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель лабораторной работы:** изучение алгоритмов и методов классификации на практике.

**Ход работы:**

Ссылка на репозиторий github: <https://github.com/sufferix/data-analysis>

**Предобработка данных**

Для выполнения классификации необходимо совершить заново предобработку данных датасета с прошлой лабораторной, Были импортированы необходимые библиотеки

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn import metrics

import matplotlib.pyplot as plt

Был загружен датасет `1heart.csv`, в котором находится информация о пациентах с анализами для определения наличия заболевания сердца.

data = pd.read\_csv('1heart.csv', sep=',')

data.head(5)

data['ExerciseAngina'] = data['ExerciseAngina'].map({'N': 0, 'Y': 1})

data['ExerciseAngina'].value\_counts()

nans = data[data.isnull().any(axis=1)]

data = data.dropna()

invalid\_rows = data[pd.to\_numeric(data['Cholesterol'], errors='coerce').isna()]

invalid\_rows

data['Cholesterol'] = data['Cholesterol'].replace('a241','241')

data['Cholesterol'] = pd.to\_numeric(data['Cholesterol'])

data['RestingBP'] = data['RestingBP'].replace(0, data['RestingBP'].median())

data['Cholesterol'] = data['Cholesterol'].replace(0, data['Cholesterol'].median())

Предобработка завершена, можно анализировать данный датасет.

Была выделена целевая переменная `HeartDisease`, по которой была выведена матрица диаграмм (см. Рисунок 1)

import seaborn as sns

sns.pairplot(data, hue= 'HeartDisease')

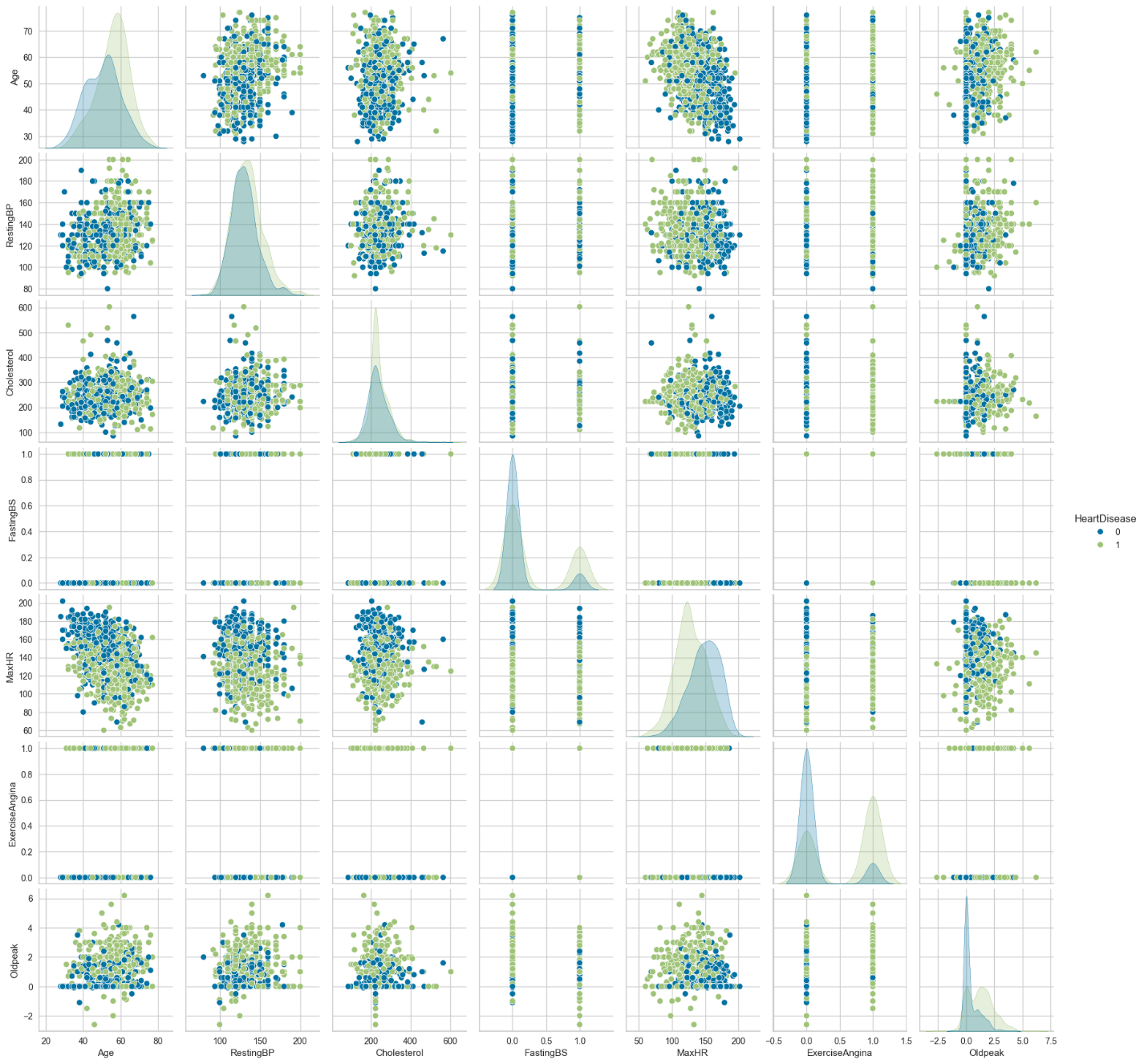


Рисунок 1 – Матрица диаграмм с целевой переменной HeartDisease

По данной матрице диаграмм видно, что данные имеют высокую степень перекрытия между группами, что затрудняет чёткое разделение классов. Наиболее отличающиеся признаки — это `Oldpeak`, `ExerciseAngina`, и `MaxHR`. Для пациентов с больным сердцем значения `Oldpeak` и `ExerciseAngina` смещены вверх, а `MaxHR` наоборот вниз.

**Выполнение классификации**

Был создан датасет только из числовых столбцов и удален столбец, который будет использоваться как целевая переменная. Также все значения были нормализованы

from sklearn.model\_selection import train\_test\_split

num\_data = data.select\_dtypes(exclude=['object'])

indie\_num\_data = num\_data.drop(columns='HeartDisease')

X = indie\_num\_data

y = num\_data['HeartDisease']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Далее были разработаны предсказательные модели `K-Nearest Neighbours`, `Decision Tree`, `Logistic Regression` и `RandomForest` и выведены основные метрики для оценивания их точности и эффективности классификации в данном контексте (см. Рисунок 2)

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, balanced\_accuracy\_score,

models = {

"K-Nearest Neighbors": KNeighborsClassifier(n\_neighbors=5),

"Decision Tree": DecisionTreeClassifier(random\_state=42),

"Logistic Regression": LogisticRegression(random\_state=42, max\_iter=500),

"Random Forest": RandomForestClassifier(random\_state=42, n\_estimators=100)

}

# Train and evaluate models

results = []

for name, model in models.items():

model.fit(X\_train, y\_train) # Train the model

y\_pred = model.predict(X\_test) # Predict on the test set

accuracy = accuracy\_score(y\_test, y\_pred) # Evaluate accuracy

precision = precision\_score(y\_test, y\_pred, average='binary')

recall = recall\_score(y\_test, y\_pred, average='binary')

f1 = f1\_score(y\_test, y\_pred, average='binary')

balanced\_accuracy = balanced\_accuracy\_score(y\_test, y\_pred)

classification\_error = 1 - accuracy\_score(y\_test, y\_pred)

results.append({

"Model": name,

"Accuracy": accuracy,

"Precision": precision,

"Recall": recall,

"F1 Score": f1,

"Balanced Accuracy": balanced\_accuracy,

"Classification Error": classification\_error

})

# Prepare and display the updated results

summary = pd.DataFrame(results)

display(summary)

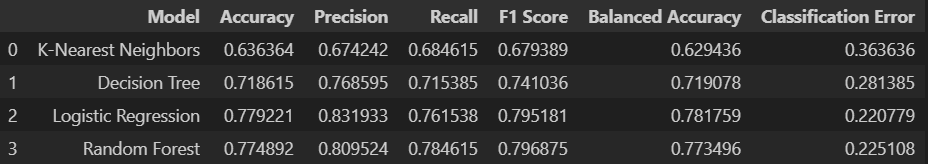


Рисунок 2 – Таблица с метриками по каждой модели

`Logistic Regression` и `Random Forest` имеют наивысшие знаения метрик среди остальных моделей, что указывает на их способность эффективно классифицировать данные. `Decision Tree` демонстрирует средние результаты, а `K-Nearest Neighbors` имеет наименьшие значения всех метрик, что указывает на его слабую производительность.

Далее была построена матрица неточностей, которая состоит из 4 элементов:

* `True Positive` (TP) - значения, которые были правильно идентифицированы как положительные
* `False Positive` (FP) - значения, которые были неправильно инентифицированы как положительные
* `True Negative` (TN) - значения, которые были правильно определены как отрицательные
* `False Negative` (FN) -значения, которые были неправильно определены как отрицательные

Построить данную матрицу можно с помощью метрики `confusion\_matrix` и библиотеки `seaborn` (см. Рисунки 3-6)

from sklearn.metrics import confusion\_matrix

import seaborn as sns

confusion\_matrices = {}

for name, model in models.items():

y\_pred = model.predict(X\_test)

# Store confusion matrix

confusion\_matrices[name] = confusion\_matrix(y\_test, y\_pred)

# Plot confusion matrices

for name, cm in confusion\_matrices.items():

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["No Disease", "Disease"], yticklabels=["No Disease", "Disease"])

plt.title(f"Confusion Matrix for {name}")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

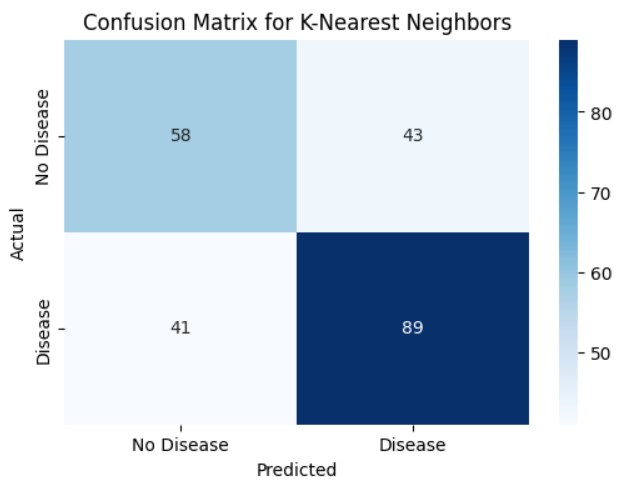


Рисунок 3 – Матрица неточностей для модели KNN

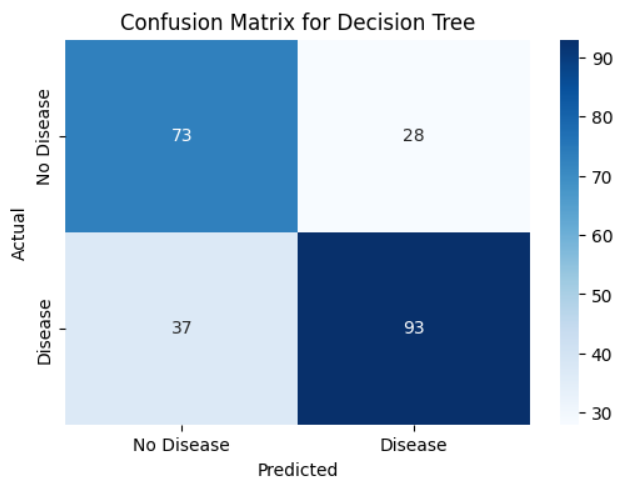


Рисунок 4 – Матрица неточностей для модели Дерево решений

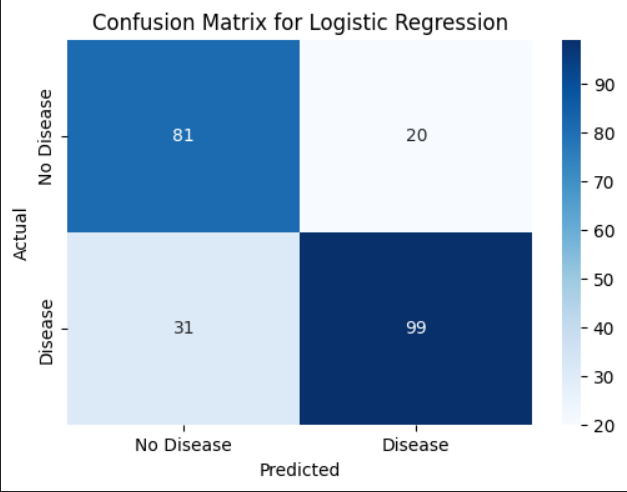


Рисунок 5 – Матрица неточностей для Логической регрессии

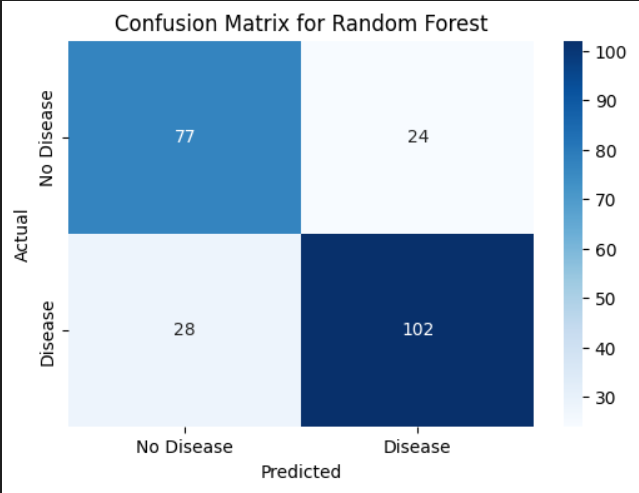


Рисунок 6 – Матрица неточностей для Случайного леса

Матрицы подтверждают, что Logistic Regression и Random Forest совершают меньше ошибок классификации, особенно ложных положительных и отрицательных. Decision Tree допустим, но менее точен, а K-Nearest Neighbors показывает высокий уровень ошибок, что подтверждает его низкую производительность.

Далее был построен график ROC-кривой, который показывает, как меняется качество бинарного классификатора при разных значениях порога вероятности. В данном графике чем выше находится линия кривой от диагонали случайного классификатора (который показывает, что алгоритм классифицировал значения случайным образом), тем больше у него AUC (площадь под кривой), тем соответственно модель лучше классифицирует данные.

Данный график был построен с помощью метода `roc\_curve`, а значения AUC были посчитаны с помощью метода `auc`

from sklearn.metrics import roc\_curve, auc

# Plot ROC curves for each model

plt.figure(figsize=(10, 8))

for name, model in models.items():

y\_proba = model.predict\_proba(X\_test)[:, 1] # Get probability scores for the positive class

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f"{name} (AUC = {roc\_auc:.2f})")

# Plot formatting

plt.plot([0, 1], [0, 1], 'k--', lw=2) # Diagonal line

plt.title("ROC Curves for All Models")

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.legend(loc="lower right")

plt.grid(alpha=0.3)

plt.show()

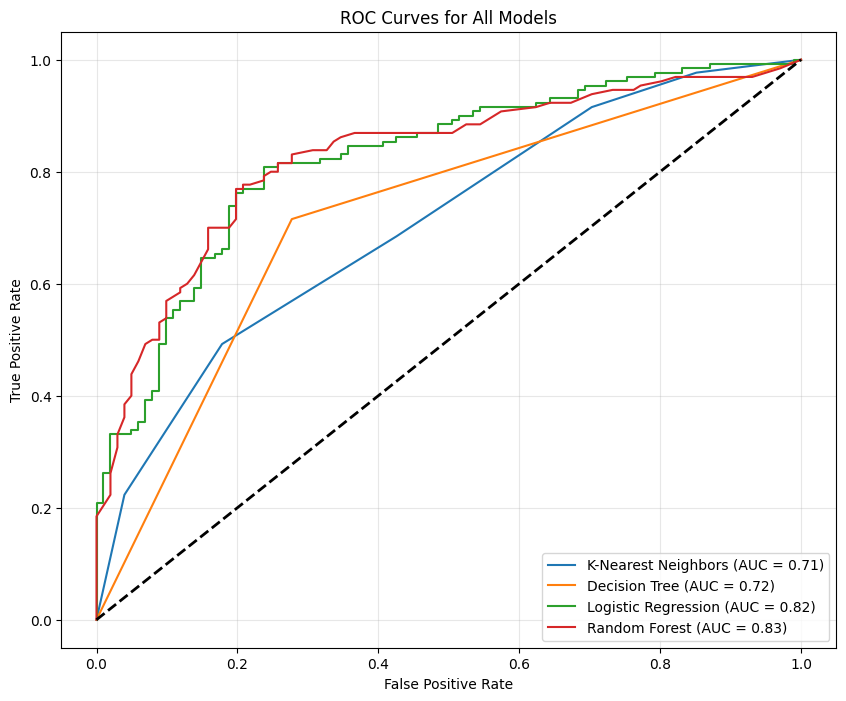


Рисунок 7 – ROC-кривые для всех моделей

На основе построенного графика ROC-кривых видно, что модели Logistic Regression и Random Forest демонстрируют наиболее высокие показатели качества классификации, так как их кривые находятся ближе к верхнему левому углу. Значения AUC для этих моделей выше по сравнению с остальными, что указывает на их способность более точно различать положительный и отрицательный классы.

Метод K-Nearest Neighbours уступает остальным моделям, так как её кривая находится ближе к диагональной линии, что говорит о меньшей эффективности классификации.

**Вывод**

В данной лабораторной работе была проведена бинарная классификация с помощью моделей K-Nearest Neighbours, Decision Tree, Logistic Regression и Random Forest для предсказания постановления диагноза заболевания сердца пациентам. Были обучены 4 модели для решения этого задания, а также просчитаны метрики для каждой модели, построены матрицы неточностей и график ROC-кривых.

Результаты проведенной работы показывают, что по всем метрикам лучше всего решают данную задачу модели Logistic Regression (accuracy 0.779, f1 score 0.795, AUC 0.83) и Random Forest (accuracy 0.775, f1 score 0.797 AUC 0.72), которые показывают довольно высокие значения, близкие друг к другу. Средние результаты по всем метрикам показала модель Decision Tree (accuracy 0.72, AUC 0.83), а хуже всех справилась модель KNN, показавшая наименьшие значения метрик - accuracy 0.64, f1 score 0.68, значение AUC 0.71