Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения»

Вариант №5

Выполнила:

студент группы ИУ5-62Б

Долинский А.А.

Преподаватель:

Гапанюк Ю. Е.

2023 г.

Задание. Для заданного набора данных – Heart Disease Dataset постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте: Метод опорных векторов и случайный лес. Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик).

Выполнение работы

Импортируем нужные библиотеки и загружаем датасет.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precision_score
%matplotlib inline
```

```
df = pd.read_csv('heart_1.csv')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Определяем целевой признак, делим датасет на обучающую и тестовую выборки и масштабируем его с помощью StandardScaler.

```
y = df['target']
X = df.drop('target', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
scaler = StandardScaler()
X_train_st = scaler.fit_transform(X_train)
X_test_st = scaler.transform(X_test)
```

Создадим и обучим модель SVM.

```
svc_model = SVC()
svc_model.fit(X_train_st, y_train)
svc_predictions = svc_model.predict(X_test_st)
print('accuracy:{}\nf1_score:{}\nprecision_score:{}\nconfusion_matrix:{}'.format(
    accuracy_score(y_test, svc_predictions),
    f1_score(y_test, svc_predictions),
    precision_score(y_test, svc_predictions),
    confusion_matrix(y_test, svc_predictions)
))
```

Создадим и обучим модель Random Forest.

```
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train_st, y_train)
rf_predictions = rf_model.predict(X_test_st)
print('accuracy:{}\nf1_score:{}\nprecision_score:{}\nconfusion_matrix:{}'.format(
    accuracy_score(y_test, rf_predictions),
    f1_score(y_test, rf_predictions),
    precision_score(y_test, rf_predictions),
    confusion_matrix(y_test, rf_predictions)
))
```

Была произведена оценка производительности с помощью методов accuracy_score, f1_score, precision_score и confusion_matrix.

Confusion_matrix - это таблица, которая показывает, насколько часто классификатор ошибается. Выводится матрица размером n x n, где n - количество классов. В каждой ячейке (i, j) матрицы указывается количество

примеров класса i, которые были помечены как класс j. Эта метрика позволяет проанализировать, какие типы ошибок допускает модель

F1_score - это гармоническое среднее между точностью и полнотой. Она используется для оценки результатов бинарной классификации, а также в многоклассовой классификации, когда интересует среднее значение показателя F1.

Precision_score – это доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Accuracy_score показывает, какая доля из всех предсказаний была правильной.

Результаты:

```
svc_model = SVC()
svc_model.fit(X_train_st, y_train)
svc_predictions = svc_model.predict(X_test_st)
print('accuracy:{}\nf1_score:{}\nprecision_score:{}\nconfusion_matrix:{}'.format(
    accuracy_score(y_test, svc_predictions),
    f1_score(y_test, svc_predictions),
    precision_score(y_test, svc_predictions),
    confusion_matrix(y_test, svc_predictions)
))
```

```
accuracy:0.9658536585365853
f1_score:0.9680365296803651
precision_score:0.9464285714285714
confusion_matrix:[[ 92   6]
 [  1 106]]
```

```
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train_st, y_train)
rf_predictions = rf_model.predict(X_test_st)
print('accuracy:{}\nf1_score:{}\nprecision_score:{}\nconfusion_matrix:{}'.format(
    accuracy_score(y_test, rf_predictions),
    f1_score(y_test, rf_predictions),
    precision_score(y_test, rf_predictions),
    confusion_matrix(y_test, rf_predictions)
))
```

```
accuracy:1.0
f1_score:1.0
precision_score:1.0
confusion_matrix:[[ 98   0]
 [  0 107]]
```

Вывод:

Обе модели показали высокие результаты, но модель случайного леса показала более высокие значения accuracy – 1, precision_score – 1 и f1_score – 1. Матрица ошибок также показала, что модель Random Forest не имеет ложноотрицательных или ложноположительных результатов, что свидетельствует о ее лучшей производительности по сравнению с моделью svm.