Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»
Отчет по рубежному контролю №2
«Методы построения моделей машинного обучения»
Вариант №2

Выполнил:
студент группы ИУ5-62Б
Балабанов
Алексей
Олегович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

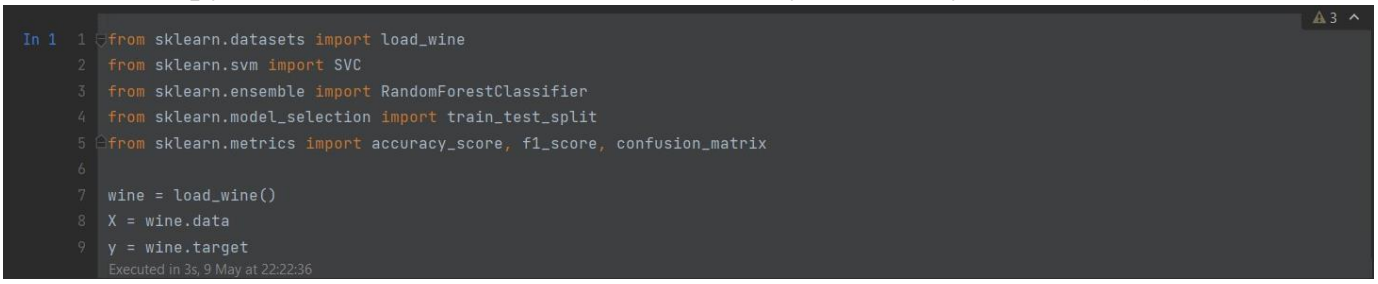
Дата:_____

Москва, 2023 г.

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

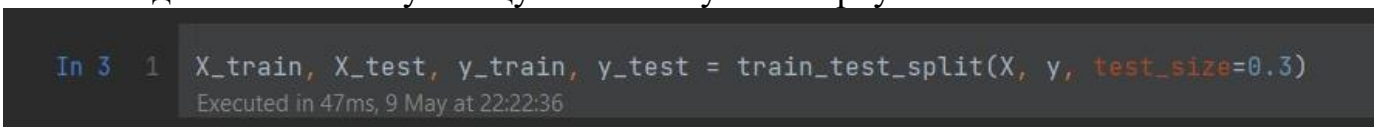
Выполнение работы

Загрузим выданный датасет вин используя команду load_wine().

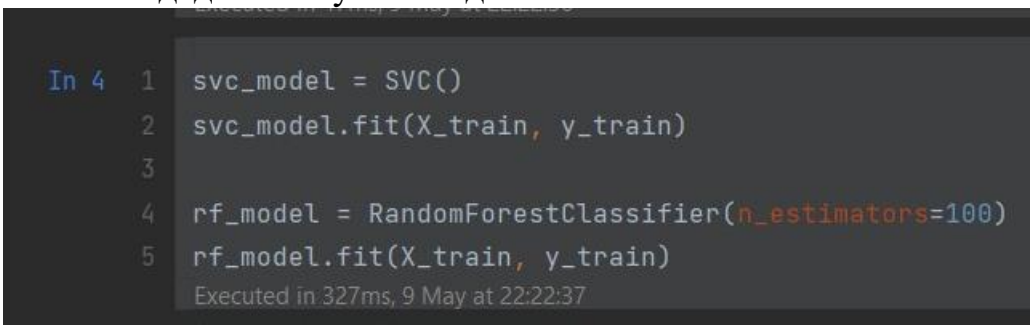


Масштабируем его с помощью StandardScaler.

Разделим его на обучающую и тестовую выборку.



Создадим и обучим модели SVM и Random Forest.



Оценим производительность разными методами. В данном случае использовались следующие метрики: Accuracy_score - показывает, какая доля из всех предсказаний была правильной. Эта метрика хорошо подходит для сбалансированных классов, но может давать неверные результаты, если классы не сбалансированы. 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. 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.

```python
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    matrix = confusion_matrix(y_test, y_pred)
    return accuracy, f1, matrix

svc_accuracy, svc_f1, svc_matrix = evaluate_model(svc_model, X_test, y_test)
rf_accuracy, rf_f1, rf_matrix = evaluate_model(rf_model, X_test, y_test)

print("SVC Model Results:")
print("Accuracy: ", svc_accuracy)
print("F1 Score: ", svc_f1)
print("Matrix: ", svc_matrix)

print("Random Forest Model Results:")
print("Accuracy: ", rf_accuracy)
print("F1 Score: ", rf_f1)
print("Matrix: ", rf_matrix)
```

Получившийся результат

```
SVC Model Results:
Accuracy:  0.9444444444444444
F1 Score:  0.9451735389235388
Matrix:  [[18  1  0]
 [ 0 18  0]
 [ 0  2 15]]
Random Forest Model Results:
Accuracy:  0.9814814814814815
F1 Score:  0.9814543481210146
Matrix:  [[19  0  0]
 [ 1 17  0]
```

Обе модели показали высокие результаты, но модель случайного леса показала более высокие значения точности (0.98) и F1-меры (0.98). Кроме того, матрица ошибок также показывает, что модель случайного леса имеет меньше ложноотрицательных и ложноположительных результатов, что свидетельствует о ее лучшей производительности в сравнении с моделью S