Министерство образования Республики Беларусь
Учреждение образования

«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
По дисциплине: «ОМО»
Тема: «Сравнение классических методов классификации»

Выполнил:
Студент 3-го курса
Группы АС-65
Гуща И.В.
Вариант 3
Проверил:
Крощенко А.А.

Брест 2025

Цель работы: На практике сравнить работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научиться подбирать гиперпараметры моделей и оценивать их влияние на результат.

Ход работы

Вариант 3

Задачи:

1. Загрузить датасет по варианту;
2. Разделить данные на обучающую и тестовую выборки;
3. Обучить на обучающей выборке три модели: k-NN, Decision Tree и SVM;
4. Для модели k-NN исследовать, как меняется качество при разном количестве соседей (k);
5. Оценить точность каждой модели на тестовой выборке;
6. Сравнить результаты, сделать выводы о применимости каждого метода для данного набора данных.

Wine Quality

• Классифицировать вино на "хорошее" (оценка >= 7) и "обычное" (оценка < 7)

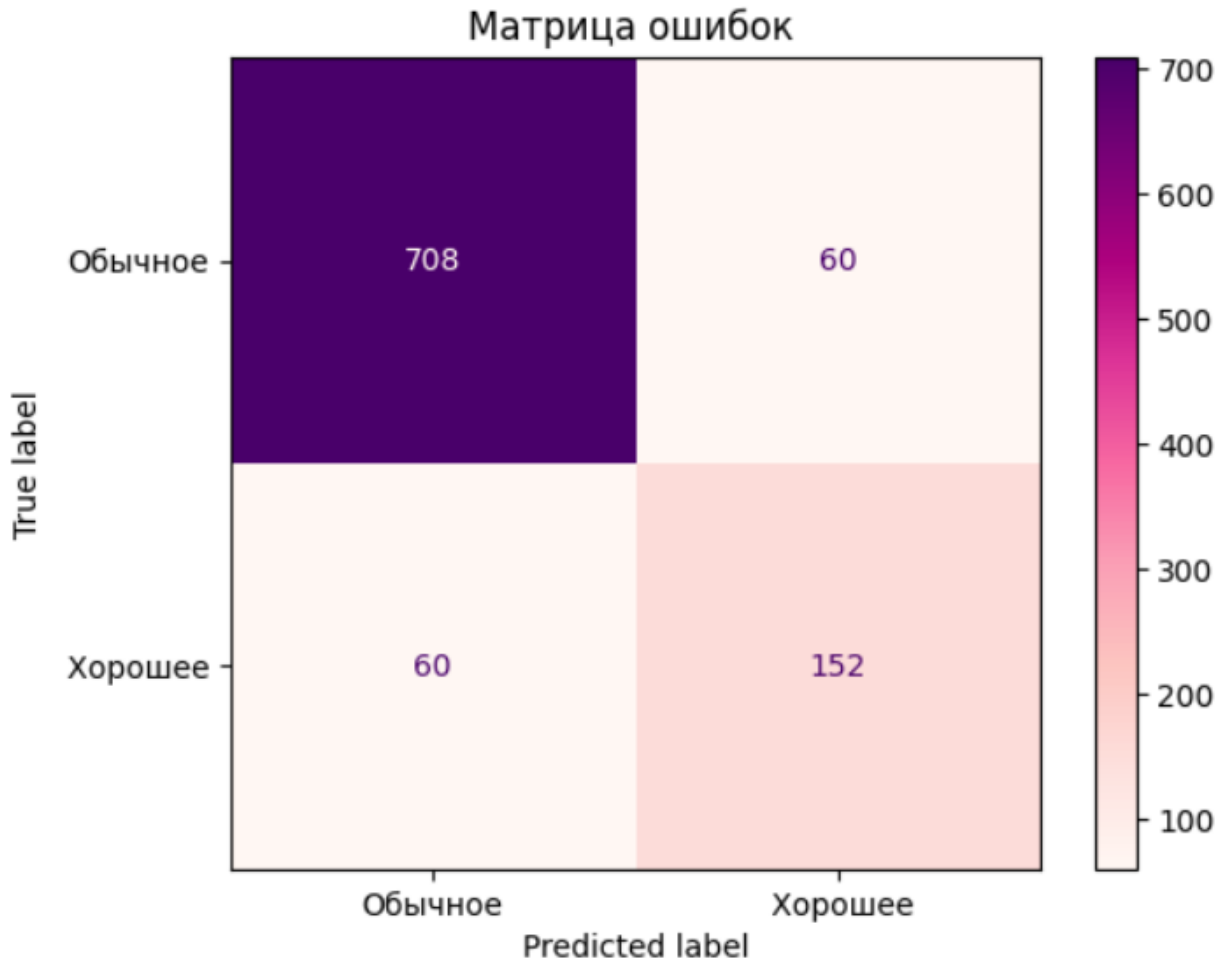
• Задания:

1. Загрузите данные и создайте бинарную целевую переменную;
2. Стандартизируйте признаки и разделите выборку;
3. Обучите модели k-NN, Decision Tree и SVM;
4. Сравните F1-score для каждой модели, так как классы могут быть несбалансированы;
5. Определите, какой алгоритм показал наилучший баланс между точностью и полнотой.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, accuracy_score, precision_score, re-
call_score, confusion_matrix, ConfusionMatrixDisplay
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from sklearn import tree as tree1

# 1. Загрузка данных и создание бинарной
data = pd.read_csv("winequality-white.csv", sep=";")
data['good'] = (data['quality'] >= 7).astype(int)
print(data)
x = data.drop(columns=['quality', 'good'])
y = data['good']

# 2. Стандартизация и разделение
scaler = StandardScaler()
x_scaled=scaler.fit_transform(x)
x_scaled = pd.DataFrame(x_scaled, columns=x.columns)
X_train, X_test, y_train, y_test = train_test_split(
    x_scaled, y, test_size=0.2, random_state=42, stratify=y
)

# 3. Обучение и оценка k-NN
k=1
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
```

```
y_knn = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_knn)
precision = precision_score(y_test, y_knn)
recall = recall_score(y_test, y_knn)
print(f"Accuracy: { accuracy: .4f} ")
print(f"Precision: { precision: .4f} ")
print(f"Recall: { recall: .4f} ")
matrix = confusion_matrix(y_test, y_knn)
print(matrix)
disp = ConfusionMatrixDisplay(confusion_matrix=matrix, display_labels=[
'Обычное' , ' Хорошее' ])
disp.plot(cmap='RdPu' )
plt.title("Матрица ошибок ")
plt.show()
print(f"Ложноположительные: {matrix[0][ 1]} ")
print(f"Ложноотрицательные: {matrix[1][ 0]} ")
print( f1_score(y_test, y_knn))
```

```
Accuracy:  0.8776
Precision:  0.7170
Recall:  0.7170
[[708  60]
 [ 60 152]]
```

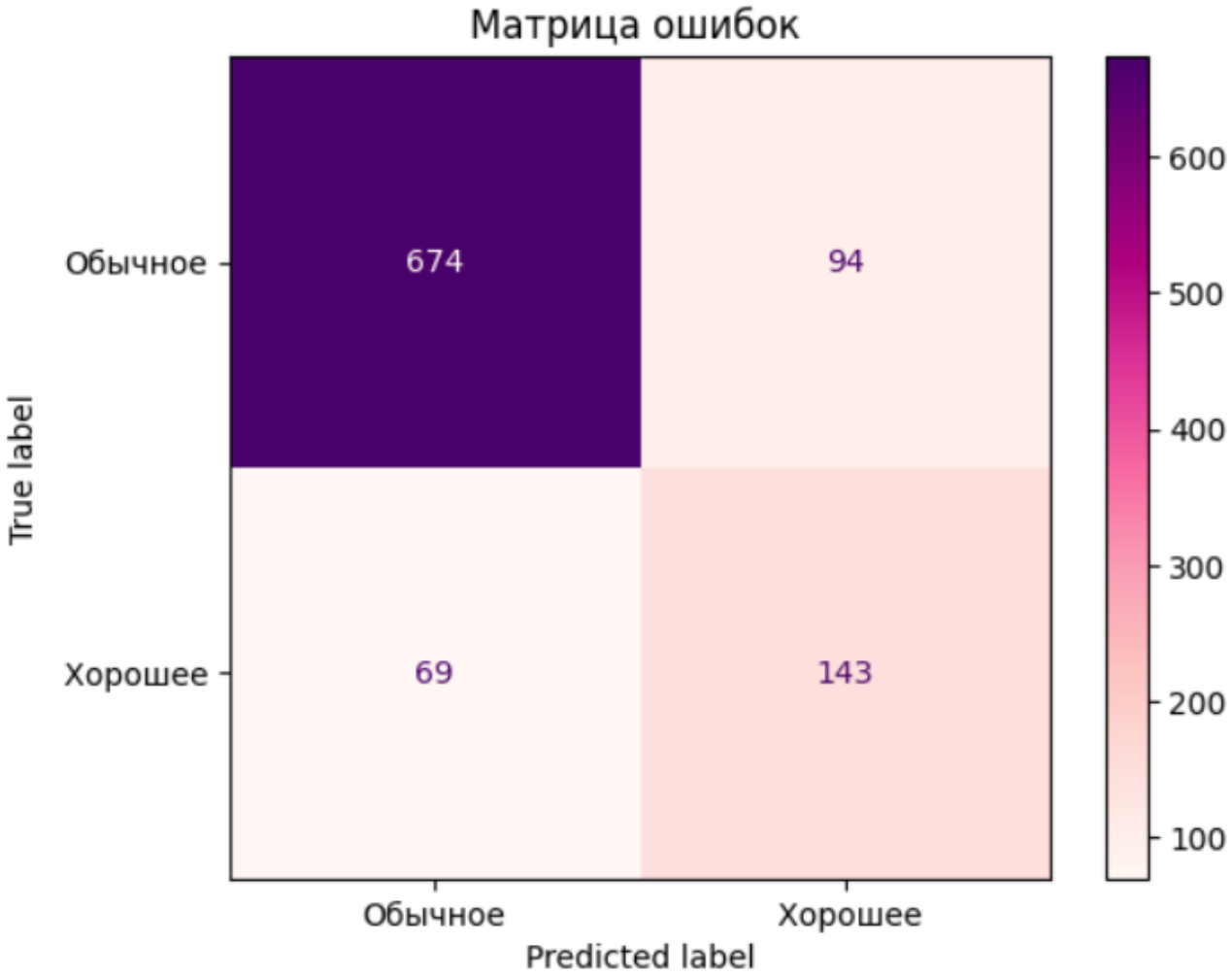



```
Ложноположительные: 60
Ложноотрицательные: 60
0.7169811320754716
```

```
# 4. Обучение и оценка Decision Tree
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
y_tree = tree.predict(X_test)
accuracy = accuracy_score(y_test, y_tree)
precision = precision_score(y_test, y_tree)
```

```
recall = recall_score(y_test, y_tree)
print(f"Accuracy: { accuracy: .4f} ")
print(f"Precision: { precision: .4f} ")
print(f"Recall: { recall: .4f} ")
matrix = confusion_matrix(y_test, y_tree)
print(matrix)
disp = ConfusionMatrixDisplay(confusion_matrix=matrix, display_labels=[
'Обычное' , ' Хорошее' ])
disp.plot(cmap='RdPu' )
plt.title("Матрица ошибок ")
plt.show()
print(f"Ложноположительные: {matrix[0][ 1]} ")
print(f"Ложноотрицательные: {matrix[1][ 0]} ")
print( f1_score(y_test, y_tree))
```

```
Accuracy:  0.8337
Precision:  0.6034
Recall:  0.6745
[[674  94]
 [ 69 143]]
```

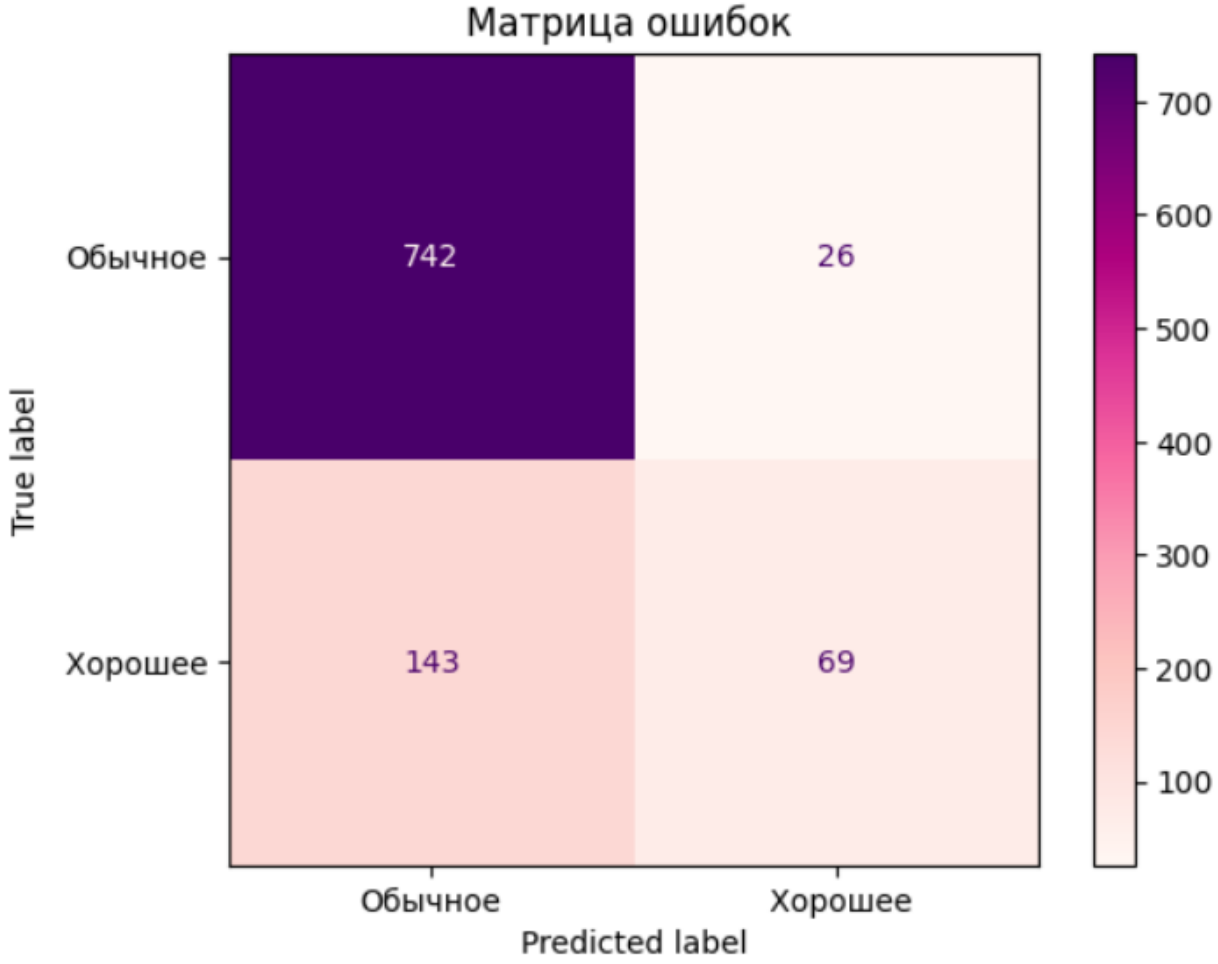



```
Ложноположительные: 94
Ложноотрицательные: 69
0.6369710467706013
```

```
# 5. Обучение и оценка SVM
svm = SVC()
svm.fit(X_train, y_train)
y_svm = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_svm)
precision = precision_score(y_test, y_svm)
recall = recall_score(y_test, y_svm)
```

```
print(f"Accuracy: { accuracy: .4f} ")
print(f"Precision: { precision: .4f} ")
print(f"Recall: { recall: .4f} ")
matrix = confusion_matrix(y_test, y_svm)
print(matrix)
disp = ConfusionMatrixDisplay(confusion_matrix=matrix, display_labels=[
'Обычное' , ' Хорошее' ])
disp.plot(cmap='RdPu' )
plt.title("Матрица ошибок ")
plt.show()
print(f"Ложноположительные: {matrix[0][ 1]} ")
print(f"Ложноотрицательные: {matrix[1][ 0]} ")
print( f1 score(y test, y svm))
```

```
Accuracy:  0.8276
Precision:  0.7263
Recall:  0.3255
[[742  26]
 [143  69]]
```




```
Ложноположительные: 26
Ложноотрицательные: 143
0.4495114006514658
```

Вывод: Модель k-NN наиболее эффективно классифицирует вина по качеству на данном наборе данных, метрика F1-score у данной модели наилучшее при k=1. Вероятнее всего это связано с тем, что хорошее вино при соблюдении параметров, но если взять среднее между ними, то это не будет показателем качества. Следующая по метрике модель Decision Tree. Минимальный показатель у SVM.