Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

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

Выполнил:
Студент 3-го курса
Группы АС-65
Осовец М. М.
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.

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;

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='whitegrid')

# Загрузка данных
df = pd.read_csv('winequality-white.csv', sep=';')
print("Первые 5 строк данных:")
display(df.head())

print("\nИнформация о данных:")
print(df.info())

# Создание бинарной целевой переменной: good (1) — если качество >= 7, иначе
0
df['good'] = (df['quality'] >= 7).astype(int)

print("\nРаспределение классов (0 — обычное, 1 — хорошее):")
print(df['good'].value_counts())

sns.countplot(x='good', data=df)
plt.title('Распределение классов (0 — обычное, 1 — хорошее)')
plt.show()
```

```
Первые 5 строк данных:
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
None

Распределение классов (0 — обычное, 1 — хорошее):
good
0    3838
1    1060
Name: count, dtype: int64
```

2. Стандартизируйте признаки и разделите выборку;

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Разделение признаков и целевой переменной
X = df.drop(columns=['quality', 'good'])
y = df['good']

# Стандартизация признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y
)

print(f"Размер обучающей выборки: {X_train.shape[0]}")
print(f"Размер тестовой выборки: {X_test.shape[0]}")
```

```
Размер обучающей выборки: 3428
Размер тестовой выборки: 1470
```

3. Обучите модели k-NN, Decision Tree и SVM;

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

from sklearn.metrics import f1_score

# --- k-NN ---
f1_scores_knn = []
k_values = range(1, 21)

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred_knn = knn.predict(X_test)
    f1_scores_knn.append(f1_score(y_test, y_pred_knn))
```

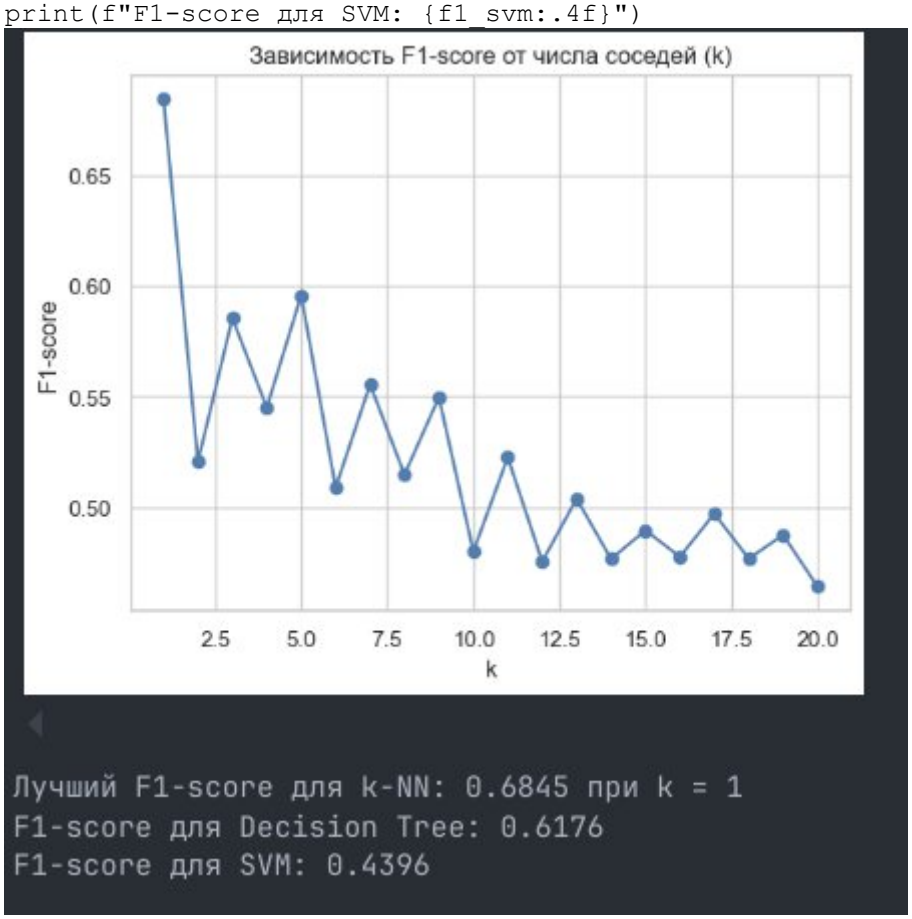
```
best_k = k_values[np.argmax(f1_scores_knn)]
best_f1_knn = max(f1_scores_knn)

plt.plot(k_values, f1_scores_knn, marker='o')
plt.title('Зависимость F1-score от числа соседей (k)')
plt.xlabel('k')
plt.ylabel('F1-score')
plt.show()

print(f"Лучший F1-score для k-NN: {best_f1_knn:.4f} при k = {best_k}")

# --- Decision Tree ---
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)
f1_tree = f1_score(y_test, y_pred_tree)
print(f"F1-score для Decision Tree: {f1_tree:.4f}")

# --- SVM ---
svm = SVC(kernel='rbf', random_state=42)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
f1_svm = f1_score(y_test, y_pred_svm)
print(f"F1-score для SVM: {f1_svm:.4f}")
```



4. Сравните F1-score для каждой модели, так как классы могут быть несбалансированы;

```
import pandas as pd

results = pd.DataFrame({
    'Модель': ['k-NN', 'Decision Tree', 'SVM'],
    'F1-score': [best_f1_knn, f1_tree, f1_svm]
})
```

```
display(results.sort_values(by='F1-score', ascending=False))

best_model = results.loc[results['F1-score'].idxmax(), 'Модель']
print(f"Лучшая модель по F1-score: {best_model}")
```

	Модель	F1-score
0	k-NN	0.684543
1	Decision Tree	0.617555
2	SVM	0.439560

Лучшая модель по F1-score: k-NN

5. Определите, какой алгоритм показал наилучший баланс между точностью и полнотой

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Разделение признаков и целевой переменной
X = df.drop(columns=['quality', 'good'])
y = df['good']

# Стандартизация признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y
)

print(f"Размер обучающей выборки: {X_train.shape[0]}")
print(f"Размер тестовой выборки: {X_test.shape[0]}")
```

```
Размер обучающей выборки: 3428
Размер тестовой выборки: 1470
```

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