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 курса
Группы АС-66
Лысюк Р. А.
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

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.

Вариант 4

- `Digits`
- Распознать, какая цифра (от 0 до 9) изображена на картинке 8x8 пикселей
- **Задания:**
  1. Загрузите встроенный в `scikit-learn` набор данных `digits`;
  2. Разделите данные на обучающую и тестовую выборки;
  3. Обучите три модели (k-NN, Decision Tree, SVM) для многоклассовой классификации;
  4. Для каждой модели выведите `classification_report` (sklearn.metrics), содержащий основные метрики для каждого класса;
  5. Сравните общую точность моделей и определите, какая из них лучше всего подходит для этой задачи.

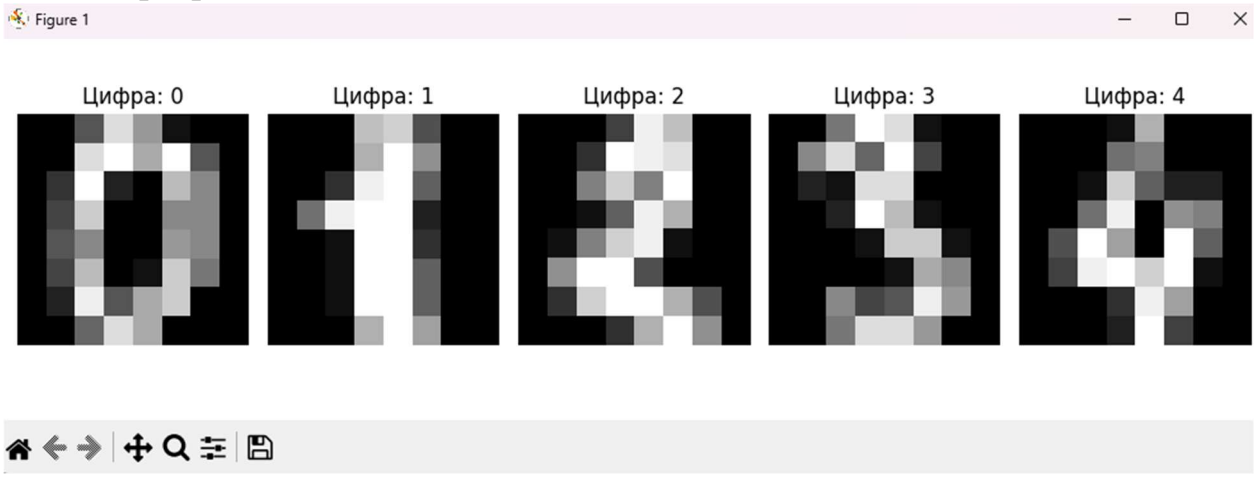
Код:

```
import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_digits

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

```
digits = load_digits()
X, y = digits.data, digits.target

fig, axes = plt.subplots(1, 5, figsize=(10, 3))
for i, ax in enumerate(axes):
    ax.imshow(digits.images[i], cmap='gray')
    ax.set_title(f'Цифра: {digits.target[i]}')
    ax.axis('off')
plt.tight_layout()
plt.show()

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

models = {
    "k-NN": KNeighborsClassifier(n_neighbors=3),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(kernel='linear', random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n=== {name} ===")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Матрица ошибок: {name}")
    plt.xlabel("Предсказано")
    plt.ylabel("Истинное")
    plt.show()

k_values = range(1, 11)
accuracies = []
```

```
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)


plt.figure(figsize=(8, 4))
sns.lineplot(x=list(k_values), y=accuracies, marker='o')
plt.title("Зависимость точности от числа соседей в k-NN")
plt.xlabel("Число соседей (k)")
plt.ylabel("Точность")
plt.grid(True)
plt.show()
```

Вывод программы:



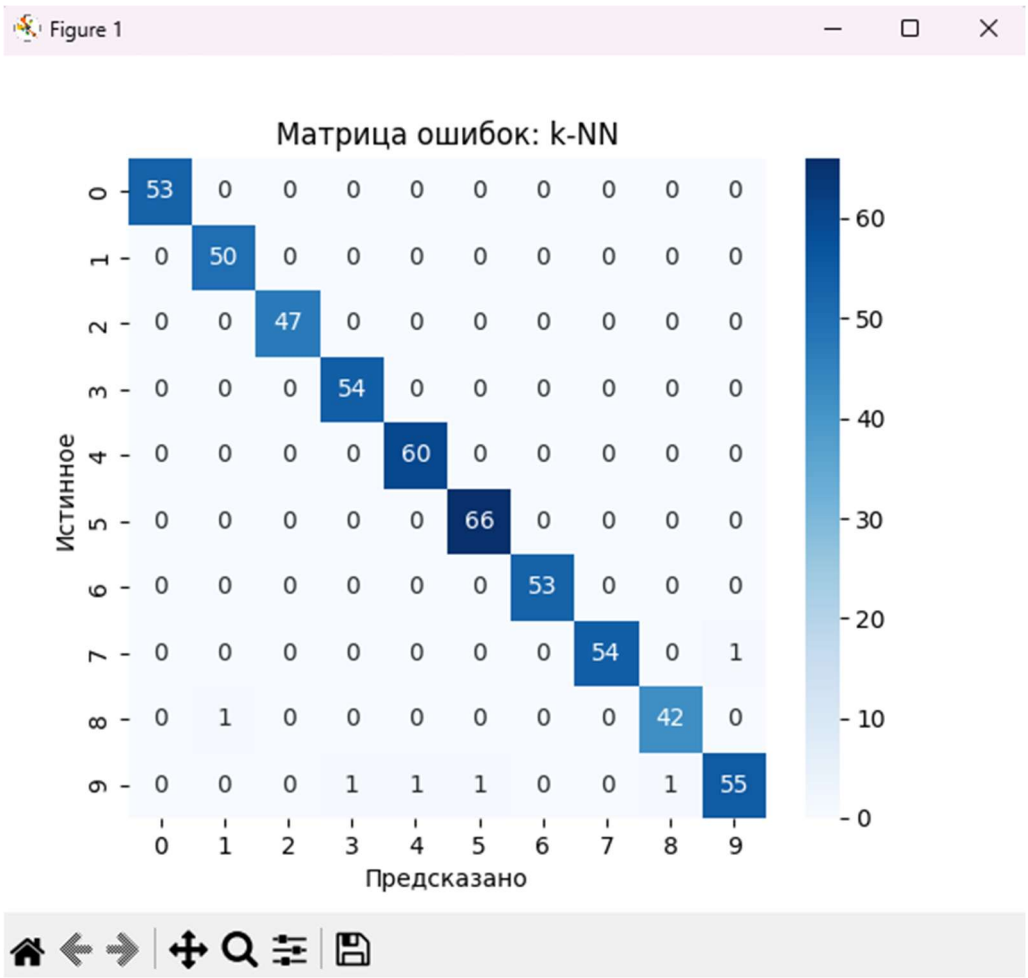


Figure 1
Матрица ошибок: k-NN
Истинное
Предсказано
0 1 2 3 4 5 6 7 8 9
53 0 0 0 0 0 0 0 0 0
0 50 0 0 0 0 0 0 0 0
0 0 47 0 0 0 0 0 0 0
0 0 0 54 0 0 0 0 0 0
0 0 0 0 60 0 0 0 0 0
0 0 0 0 0 66 0 0 0 0
0 0 0 0 0 0 53 0 0 0
0 0 0 0 0 0 0 54 0 1
0 1 0 0 0 0 0 0 42 0
0 0 0 1 1 1 0 0 1 55
60 50 40 30 20 10 0

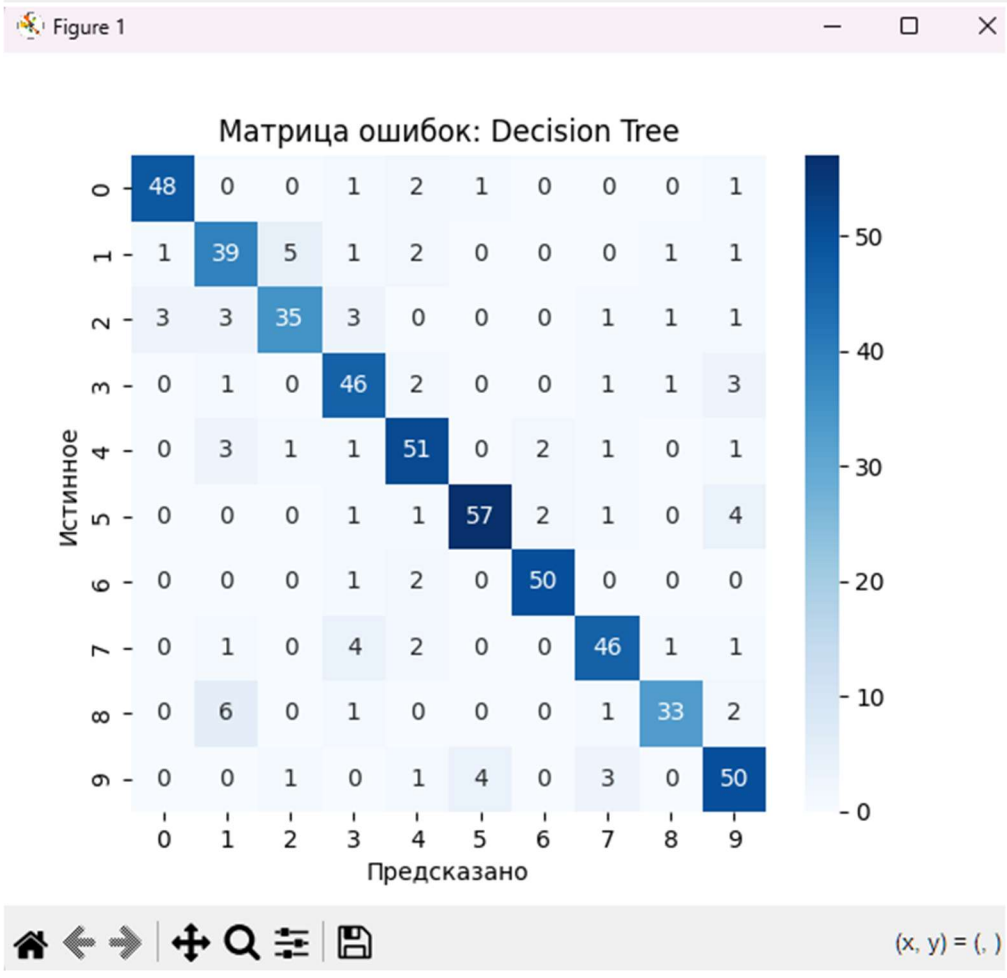


Figure 1
Матрица ошибок: Decision Tree
Истинное
Предсказано
0 1 2 3 4 5 6 7 8 9
48 0 0 1 2 1 0 0 0 1
1 39 5 1 2 0 0 0 1 1
3 3 35 3 0 0 0 1 1 1
0 1 0 46 2 0 0 1 1 3
0 3 1 1 51 0 2 1 0 1
0 0 0 1 1 57 2 1 0 4
0 0 0 1 2 0 50 0 0 0
0 1 0 4 2 0 0 46 1 1
0 6 0 1 0 0 0 1 33 2
0 0 1 0 1 4 0 3 0 50
50 40 30 20 10 0
(x, y) = (,)

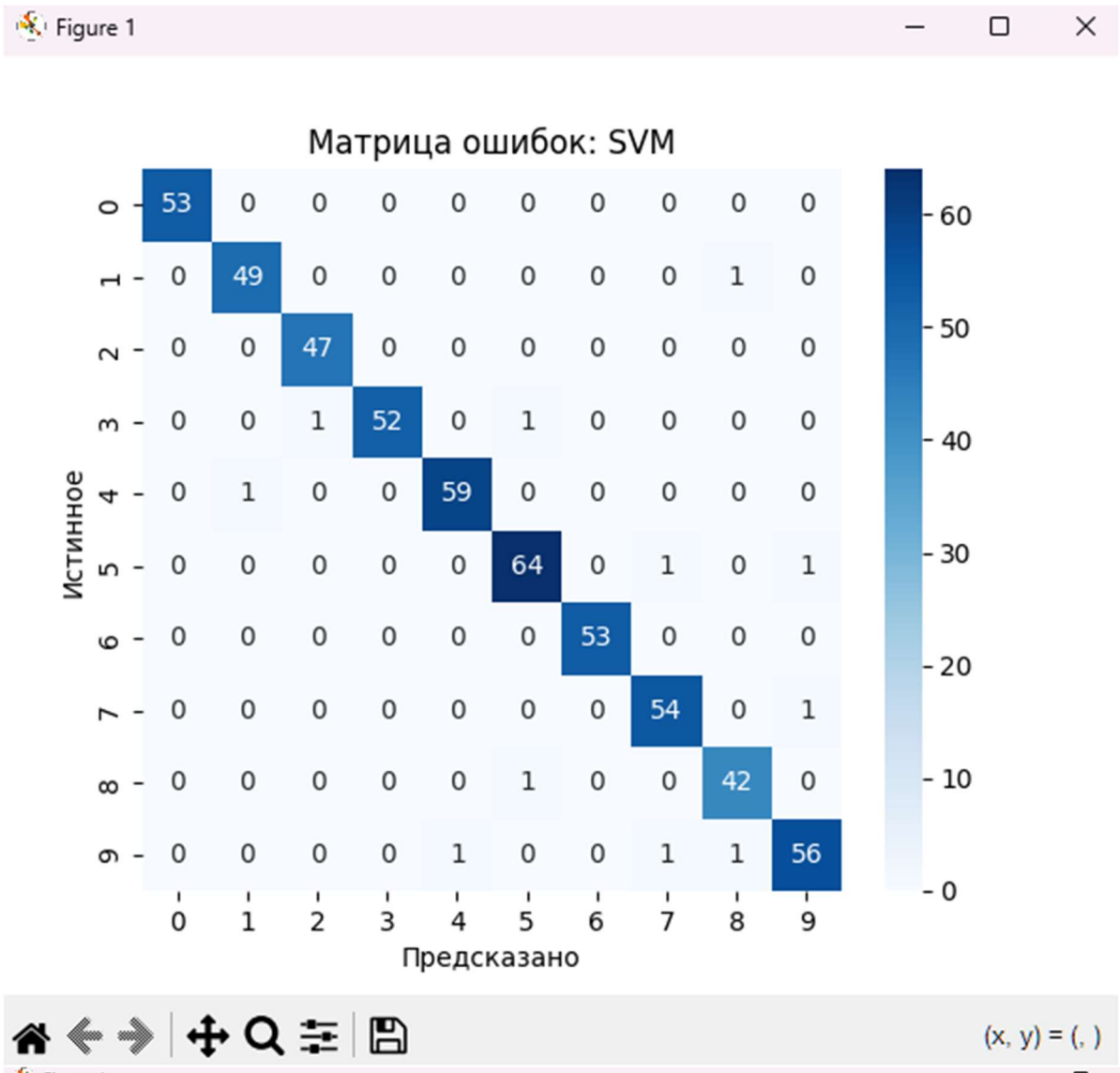


Figure 1
Матрица ошибок: SVM
Истинное
Предсказано
0 1 2 3 4 5 6 7 8 9
53 0 0 0 0 0 0 0 0 0
0 49 0 0 0 0 0 0 1 0
0 0 47 0 0 0 0 0 0 0
0 0 1 52 0 1 0 0 0 0
0 1 0 0 59 0 0 0 0 0
0 0 0 0 0 64 0 1 0 1
0 0 0 0 0 0 53 0 0 0
0 0 0 0 0 0 0 54 0 1
0 0 0 0 0 1 0 0 42 0
0 0 0 0 1 0 0 1 1 56
60 50 40 30 20 10 0
(x, y) = (,)


Figure 1
Зависимость точности от числа соседей в k-NN
Точность
Число соседей (k)
0.994
0.992
0.990
0.988
0.986
0.984
0.982
2 4 6 8 10

Вывод в консоли:

```
=== k-NN ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       0.98      1.00      0.99        50
           2       1.00      1.00      1.00        47
           3       0.98      1.00      0.99        54
           4       0.98      1.00      0.99        60
           5       0.99      1.00      0.99        66
           6       1.00      1.00      1.00        53
           7       1.00      0.98      0.99        55
           8       0.98      0.98      0.98        43
           9       0.98      0.93      0.96        59

    accuracy                           0.99       540
   macro avg       0.99      0.99      0.99       540
weighted avg       0.99      0.99      0.99       540

Accuracy: 0.9889

=== Decision Tree ===
              precision    recall  f1-score   support

           0       0.92      0.91      0.91        53
           1       0.74      0.78      0.76        50
           2       0.83      0.74      0.79        47
           3       0.78      0.85      0.81        54
           4       0.81      0.85      0.83        60
           5       0.92      0.86      0.89        66
           6       0.93      0.94      0.93        53
           7       0.85      0.84      0.84        55
           8       0.89      0.77      0.82        43
           9       0.78      0.85      0.81        59

    accuracy                           0.84       540
   macro avg       0.85      0.84      0.84       540
weighted avg       0.85      0.84      0.84       540

Accuracy: 0.8426
```