

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

Лабораторная работа №3
По дисциплине «Основы машинного обучения»
Тема: «Сравнение классических методов классификации»

Выполнил:
Студент 3 курса
Группы АС-65
Дацевич М.Г.
Проверил:
Крощенко А.А.

Брест 2025

Цель: 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 4

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.

```
# 1. Загружаем датасет digits 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 import matplotlib.pyplot as plt import numpy as np
print("=== ЗАГРУЗКА ДАТАСЕТА DIGITS
===")
digits = load_digits()
X = digits.data # данные (1797 изображений, каждое 8x8 = 64 пикселя) y
= digits.target # метки (цифры от 0 до 9)
print(f"Размер данных:
{X.shape}") print(f"Размер меток:
{y.shape}") print(f"Классы:
{np.unique(y)}")

# Покажем пример цифры
plt.figure(figsize=(8, 4)) for
i in range(10):
    plt.subplot(2, 5, i + 1)
plt.imshow(digits.images[i], cmap='binary')
plt.title(f"Цифра: {digits.target[i]}")
plt.axis('off') plt.tight_layout() plt.show()

# 2. Разделяем на обучающую и тестовую выборки print("\n===
РАЗДЕЛЕНИЕ ДАННЫХ ===")
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
) print(f"Обучающая выборка: {X_train.shape[0]}
примеров") print(f"Тестовая выборка: {X_test.shape[0]}
примеров")

# 3. Обучаем три модели
print("\n=== ОБУЧЕНИЕ МОДЕЛЕЙ ===")

models = {
```

```

    'k-NN': KNeighborsClassifier(n_neighbors=3),
    'Decision Tree': DecisionTreeClassifier(random_state=42, max_depth=10),
    'SVM': SVC(random_state=42, kernel='rbf')
}

# Словарь для хранения результатов
results = {}

# 4. Обучение и оценка для каждой модели
for name, model in models.items():
    print(f"\n{'='*60}")
    print(f"МОДЕЛЬ: {name}")
    print(f"{'='*60}")

    # Обучение модели
    model.fit(X_train, y_train)

    # Предсказания на тестовой выборке
    y_pred = model.predict(X_test)

    # Сохраняем результаты
    results[name] = {
        'model': model,
        'predictions': y_pred,
        'accuracy': accuracy_score(y_test, y_pred)
    }

    # Classification report для каждой модели
    print(f"ОТЧЕТ ПО КЛАССИФИКАЦИИ ДЛЯ {name}:")
    print(classification_report(y_test, y_pred, digits=4))
    print(f"ОБЩАЯ ТОЧНОСТЬ: {results[name]['accuracy']:.4f}")

# 5. Сравнение моделей
print(f"\n{'='*70}") print("СРАВНЕНИЕ
МОДЕЛЕЙ") print(f"{'='*70}")

# Сортируем модели по точности sorted_results = sorted(results.items(), key=lambda x:
x[1]['accuracy'], reverse=True)
print("РЕЙТИНГ МОДЕЛЕЙ ПО ТОЧНОСТИ:") for i, (name,
result) in enumerate(sorted_results, 1):
print(f"{i}. {name}: {result['accuracy']:.4f}")

# Определяем лучшую модель best_model_name, best_result =
sorted_results[0] print(f"\n ЛУЧШАЯ МОДЕЛЬ:
{best_model_name} с точностью
{best_result['accuracy']:.4f}")

# Демонстрация работы лучшей модели на нескольких примерах print(f"\n===
ДЕМОНСТРАЦИЯ РАБОТЫ ЛУЧШЕЙ МОДЕЛИ ({best_model_name}) ===")
best_model =
best_result['model']

# Берем несколько случайных примеров из тестовой выборки
np.random.seed(42) sample_indices =
np.random.choice(len(X_test), 5, replace=False)
plt.figure(figsize=(12, 3)) for i, idx
in enumerate(sample_indices): #
Берем пример из тестовой выборки
test_image = X_test[idx].reshape(8, 8)
true_label = y_test[idx]

    # Предсказываем цифру predicted_label =
best_model.predict([X_test[idx]])[0]

```

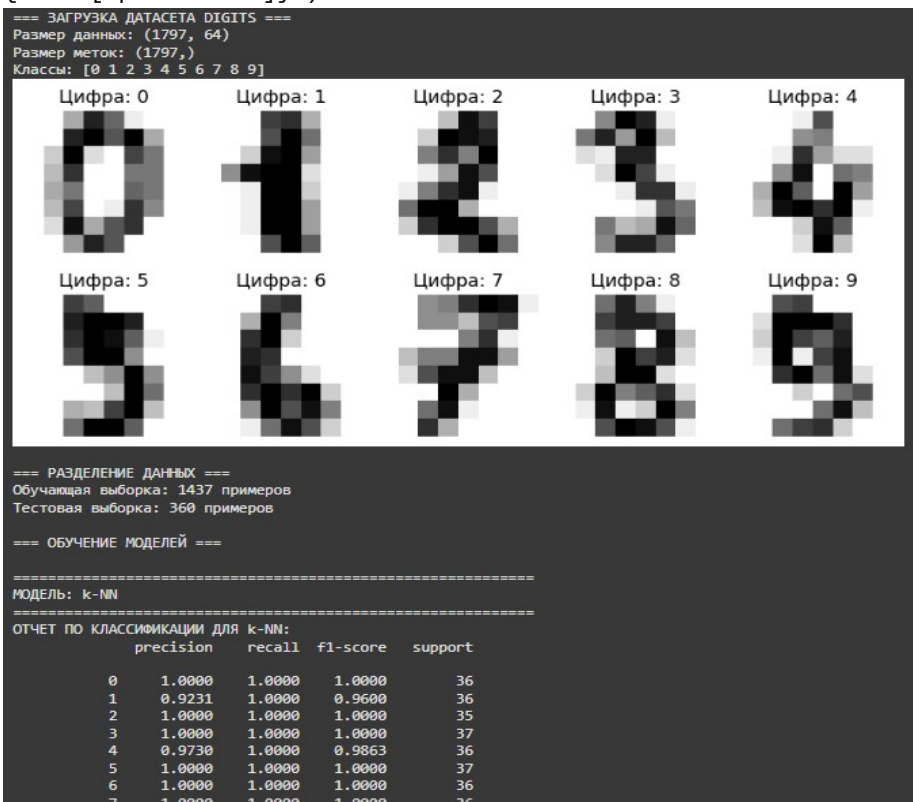
```

# Визуализируем plt.subplot(1, 5, i + 1)
plt.imshow(test_image, cmap='binary') plt.title(f"Истина:
{true_label}\nПредсказано: {predicted_label}",
color='green' if true_label == predicted_label else 'red')
plt.axis('off')
plt.tight_layout()
plt.show()

# Анализ ошибок print("\n=== АНАЛИЗ ОШИБОК ===")
wrong_predictions = [] for i in range(len(X_test)): if
y_test[i] != results[best_model_name]['predictions'][i]:

wrong_predictions.append({ 'ind
ex': i,
'true': y_test[i],
'predicted': results[best_model_name]['predictions'][i]
})
print(f"Количество ошибок у лучшей модели:
{len(wrong_predictions)}") print(f"Точность: {(1 -
len(wrong_predictions)/len(X_test)):.4f}")
if
wrong_predictions:
print("\nПримеры ошибок:") for i, error in
enumerate(wrong_predictions[:3]):
print(f" Пример {error['index']}: истинная цифра {error['true']}, предсказана
{error['predicted']}")

```



```

      8      0.9697      0.9143      0.9412      35
      9      1.0000      0.9444      0.9714      36

accuracy      0.9861
macro avg      0.9866      0.9859      0.9859      360
weighted avg    0.9867      0.9861      0.9861      360

ОБЩАЯ ТОЧНОСТЬ: 0.9861

=====
МОДЕЛЬ: Decision Tree
=====
ОТЧЕТ ПО КЛАССИФИКАЦИИ ДЛЯ Decision Tree:
precision      recall      f1-score      support

      0      0.9444      0.9444      0.9444      36
      1      0.6579      0.6944      0.6757      36
      2      0.9333      0.8000      0.8615      35
      3      0.7895      0.8108      0.8000      37
      4      0.8571      0.8333      0.8451      36
      5      0.8750      0.9459      0.9091      37
      6      0.8571      0.8333      0.8451      36
      7      0.8108      0.8333      0.8219      36
      8      0.6316      0.6857      0.6575      35
      9      0.7879      0.7222      0.7536      36

accuracy      0.8111
macro avg      0.8145      0.8104      0.8114      360
weighted avg    0.8147      0.8111      0.8119      360

ОБЩАЯ ТОЧНОСТЬ: 0.8111

=====
МОДЕЛЬ: SVM
=====
ОТЧЕТ ПО КЛАССИФИКАЦИИ ДЛЯ SVM:
precision      recall      f1-score      support

      0      1.0000      1.0000      1.0000      36
      1      0.9474      1.0000      0.9730      36
      2      1.0000      1.0000      1.0000      35
      3      1.0000      1.0000      1.0000      37
      4      1.0000      1.0000      1.0000      36
      5      1.0000      1.0000      1.0000      37
      6      1.0000      1.0000      1.0000      36
      7      0.9730      1.0000      0.9863      36
      8      1.0000      0.9429      0.9706      35
      9      1.0000      0.9722      0.9859      36

accuracy      0.9917
macro avg      0.9920      0.9915      0.9916      360
weighted avg    0.9920      0.9917      0.9917      360

ОБЩАЯ ТОЧНОСТЬ: 0.9917

```

```

=====
СРАВНЕНИЕ МОДЕЛЕЙ
=====
РЕЙТИНГ МОДЕЛЕЙ ПО ТОЧНОСТИ:
1. SVM: 0.9917
2. k-NN: 0.9861
3. Decision Tree: 0.8111

🏆 ЛУЧШАЯ МОДЕЛЬ: SVM с точностью 0.9917

=== ДЕМОНСТРАЦИЯ РАБОТЫ ЛУЧШЕЙ МОДЕЛИ (SVM) ===

Истина: 4      Истина: 9      Истина: 9      Истина: 1      Истина: 4
Предсказано: 4      Предсказано: 9      Предсказано: 9      Предсказано: 1      Предсказано: 4

[Изображения: 5 примеров цифр 4, 9, 1, 4, 4 с их истинными и предсказанными значениями]

=== АНАЛИЗ ОШИБОК ===
Количество ошибок у лучшей модели: 3
Точность: 0.9917

Примеры ошибок:
Пример 51: истинная цифра 8, предсказана 1
Пример 93: истинная цифра 9, предсказана 7
Пример 127: истинная цифра 8, предсказана 1

```

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