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
Лопато А. В.
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 1

Ход работы:

Задание:

• Iris

• Определить вид ириса (setosa, versicolor, virginica) по измерениям его цветка

1. Загрузите данные и ознакомьтесь с ними;

2. Разделите выборку на обучающую и тестовую;

3. Обучите модели k-NN, Decision Tree и SVM. Для k-NN попробуйте найти оптимальное значение k;

4. Оцените точность (accuracy) каждой модели на тестовой выборке;

5. Сравните результаты и сделайте вывод, какая модель лучше всего справилась с этой задачей.

```
# Импорт библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
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 accuracy_score, classification_report

# 1. Загрузка данных и ознакомление с ними
print("=== 1. ЗАГРУЗКА ДАННЫХ ===")
iris = load_iris()
X = iris.data
y = iris.target

# Создаем DataFrame для анализа
iris_df = pd.DataFrame(X, columns=iris.feature_names)
iris_df['species'] = [iris.target_names[i] for i in y]

print("Первые 5 строк данных:")
print(iris_df.head())
```

```
    print("\nСтатистика данных:")
    print(iris_df.describe())

    print("\nРаспределение по классам:")
    print(iris_df['species'].value_counts())

    # Простая визуализация
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=iris_df, x='petal length (cm)', y='petal width (cm)',
hue='species')
    plt.title('Визуализация данных ирисов')
    plt.show()

    # 2. Разделение выборки
    print("\n=== 2. РАЗДЕЛЕНИЕ ДАННЫХ ===")
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    print(f"Обучающая выборка: {X_train.shape[0]} примеров")
    print(f"Тестовая выборка: {X_test.shape[0]} примеров")

    # 3. Обучение моделей
    print("\n=== 3. ОБУЧЕНИЕ МОДЕЛЕЙ ===")

    # Поиск оптимального k для k-NN
    print("Поиск оптимального k для k-NN...")
    best_k = 1
    best_score = 0

    for k in range(1, 16):
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        score = knn.score(X_test, y_test)
        if score > best_score:
            best_score = score
            best_k = k

    print(f"Оптимальное k: {best_k} (точность: {best_score:.4f})")
```

```
# Создаем и обучаем модели
models = {
    f'k-NN (k={best_k})': KNeighborsClassifier(n_neighbors=best_k),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'SVM': SVC(random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    print(f"Обучена модель: {name}")

# 4. Оценка точности
print("\n=== 4. ОЦЕНКА ТОЧНОСТИ ===")
results = {}

for name, model in models.items():
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

    print(f"\n{name}:")
    print(f"Точность: {accuracy:.4f}")
    print("Отчет классификации:")
    print(classification_report(y_test, y_pred, target_names=iris.target_names))

# 5. Сравнение результатов
print("\n=== 5. СРАВНЕНИЕ РЕЗУЛЬТАТОВ ===")

# Визуализация сравнения моделей
plt.figure(figsize=(8, 5))
models_names = list(results.keys())
accuracies = list(results.values())

bars = plt.bar(models_names, accuracies, color=['lightblue', 'lightgreen', 'salmon'])
plt.ylabel('Точность (accuracy)')
plt.title('Сравнение точности моделей')
plt.ylim(0.8, 1.0)
```

```
# Добавляем значения на столбцы
for bar, accuracy in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.005,
             f'{accuracy:.4f}', ha='center', va='bottom')

plt.xticks(rotation=15)
plt.tight_layout()
plt.show()

# Вывод лучшей модели
best_model = max(results, key=results.get)
print(f"\nЛучшая модель: {best_model}")
print(f"Точность: {results[best_model]:.4f}")

# Дополнительно: важность признаков для Decision Tree
print("\nВажность признаков (Decision Tree):")
dt_model = models['Decision Tree']
feature_importance = pd.DataFrame({
    'Признак': iris.feature_names,
    'Важность': dt_model.feature_importances_
}).sort_values('Важность', ascending=False)

print(feature_importance)
```

```
=== 1. ЗАГРУЗКА ДАННЫХ ===
Первые 5 строк данных:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm) species
0                5.1               3.5                1.4               0.2  setosa
1                4.9               3.0                1.4               0.2  setosa
2                4.7               3.2                1.3               0.2  setosa
3                4.6               3.1                1.5               0.2  setosa
4                5.0               3.6                1.4               0.2  setosa

Статистика данных:
       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
count         150.000000        150.000000         150.000000        150.000000
mean            5.843333          3.057333           3.758000          1.199333
std             0.828066          0.435866           1.765298          0.762238
min             4.300000          2.000000           1.000000          0.100000
25%             5.100000          2.800000           1.600000          0.300000
50%             5.800000          3.000000           4.350000          1.300000
75%             6.400000          3.300000           5.100000          1.800000
max             7.900000          4.400000           6.900000          2.500000

Распределение по классам:
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64

=== 2. РАЗДЕЛЕНИЕ ДАННЫХ ===
Обучающая выборка: 105 примеров
Тестовая выборка: 45 примеров

=== 3. ОБУЧЕНИЕ МОДЕЛЕЙ ===
Поиск оптимального k для k-NN...
Оптимальное k: 1 (точность: 1.0000)
Обучена модель: k-NN (k=1)
Обучена модель: Decision Tree
Обучена модель: SVM

=== 4. ОЦЕНКА ТОЧНОСТИ ===

k-NN (k=1):
Точность: 1.0000
Отчет классификации:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45

Decision Tree:
Точность: 1.0000
Отчет классификации:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


SVM:
Точность: 1.0000
Отчет классификации:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


=== 5. СРАВНЕНИЕ РЕЗУЛЬТАТОВ ===

Лучшая модель: k-NN (k=1)
Точность: 1.0000

Важность признаков (Decision Tree):
             Признак  Важность
2  petal length (cm)  0.893264
3   petal width (cm)  0.087626
1   sepal width (cm)  0.019110
0  sepal length (cm)  0.000000
```

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