

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  
Вавдийчик Н.Д.  
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.

## 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 1

- Iris
- Определить вид ириса (setosa, versicolor, virginica) по измерениям его цветка
- Задания:
  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.

### Код программы (1):

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# 1. Загрузка датасета
df = pd.read_csv('iris.csv')
print("Датасет загружен. Размер:", df.shape)

# 2. Разделение на обучающую и тестовую выборки
X = df.drop('variety', axis=1)
y = df['variety']

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

# Масштабирование для k-NN и SVM
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```

# 3. Обучение моделей и исследование k-NN
print("\nИсследование k для k-NN:")
k_values = range(1, 16)
knn_accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)
    accuracy = accuracy_score(y_test, y_pred)
    knn_accuracies.append(accuracy)
    print(f"k={k}: Точность = {accuracy:.4f}")

# Находим оптимальное k
optimal_k = k_values[np.argmax(knn_accuracies)]
print(f"\nОптимальное k: {optimal_k}")

# 4. Обучение всех моделей с оптимальными параметрами
models = {
    f'k-NN (k={optimal_k})': KNeighborsClassifier(n_neighbors=optimal_k),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'SVM': SVC(random_state=42)
}

results = {}

for name, model in models.items():
    if 'k-NN' in name or 'SVM' in name:
        # Для k-NN и SVM используем масштабированные данные
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
    else:
        # Для Decision Tree используем исходные данные
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy
    print(f"{name}: Точность = {accuracy:.4f}")

# 5. Сравнение результатов
print("\n" + "=" * 40)
print("СРАВНЕНИЕ РЕЗУЛЬТАТОВ")
print("=" * 40)

best_model = max(results, key=results.get)
print(f"Лучшая модель: {best_model} с точностью {results[best_model]:.4f}")

print("\nВыводы:")
print("1. Все модели показывают высокую точность на датасете Iris")
print("2. Набор данных хорошо разделим и подходит для всех трех методов")
print("3. Для практического применения рекомендуется: ", best_model)
print("\n" + "="*50)
print("ДЕТАЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ")
print("="*50)

print(f"\nТочности моделей:")
for model_name, accuracy in results.items():
    print(f" {model_name}: {accuracy:.4f} ({accuracy*100:.1f}%)")

```

```
print(f"\nРазница между лучшей и худшой моделью: {max(results.values()) - min(results.values()):.4f}")

# Анализ k-NN
print(f"\nАнализ k-NN:")
print(f"- Лучшее k: {optimal_k}")
print(f"- Диапазон точности при разных k: {min(knn_accuracies):.4f} - {max(knn_accuracies):.4f}")
print(f"- k-NN показал стабильно высокую точность при k > 3")

# Сравнительный анализ
print(f"\nСравнительный анализ моделей:")
print(f"1. k-NN (k=9): Наивысшая точность - хорошо подходит для этого набора данных")
print(f"2. Decision Tree: Немного ниже точность, но хорошая интерпретируемость")
print(f"3. SVM: Такая же точность как у Decision Tree, но требует масштабирования")

print(f"\nВЫВОДЫ:")
print(f"Все модели показали точность выше 93%")
print(f"Разница между моделями незначительна (максимум 2.2%)")
print(f"Для практического применения рекомендуется k-NN с k=9")
print(f"Decision Tree может быть предпочтительнее если важна интерпретируемость")
```

```
датасет загружен. Размер: (150, 5)
```

```
Исследование k для k-NN:
```

```
k=1: Точность = 0.9333  
k=2: Точность = 0.8889  
k=3: Точность = 0.9111  
k=4: Точность = 0.9111  
k=5: Точность = 0.9111  
k=6: Точность = 0.9111  
k=7: Точность = 0.9333  
k=8: Точность = 0.9111  
k=9: Точность = 0.9556  
k=10: Точность = 0.9333  
k=11: Точность = 0.9556  
k=12: Точность = 0.9556  
k=13: Точность = 0.9333  
k=14: Точность = 0.9556  
k=15: Точность = 0.9333
```

```
Оптимальное k: 9
```

```
k-NN (k=9): Точность = 0.9556  
Decision Tree: Точность = 0.9333  
SVM: Точность = 0.9333
```

```
=====
```

```
СРАВНЕНИЕ РЕЗУЛЬТАТОВ
```

```
=====
```

```
Лучшая модель: k-NN (k=9) с точностью 0.9556
```

```
Выводы:
```

1. Все модели показывают высокую точность на датасете Iris
2. Набор данных хорошо разделим и подходит для всех трех методов
3. Для практического применения рекомендуется: k-NN (k=9)

```
=====
```

```
СРАВНЕНИЕ РЕЗУЛЬТАТОВ
```

```
=====
```

```
Лучшая модель: k-NN (k=9) с точностью 0.9556
```

```
Выводы:
```

1. Все модели показывают высокую точность на датасете Iris
2. Набор данных хорошо разделим и подходит для всех трех методов
3. Для практического применения рекомендуется: k-NN (k=9)

```
=====
```

```
ДЕТАЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ
```

```
=====
```

```
Точности моделей:
```

```
    k-NN (k=9): 0.9556 (95.6%)  
    Decision Tree: 0.9333 (93.3%)  
    SVM: 0.9333 (93.3%)
```

```
Разница между лучшей и худшней моделью: 0.0222
```

```
Анализ k-NN:
```

- Лучшее k: 9
- Диапазон точности при разных k: 0.8889 - 0.9556
- k-NN показал стабильно высокую точность при k > 3

```
Сравнительный анализ моделей:
```

1. k-NN (k=9): Наивысшая точность - хорошо подходит для этого набора данных
2. Decision Tree: Немного ниже точность, но хорошая интерпретируемость
3. SVM: Такая же точность как у Decision Tree, но требует масштабирования

```
ВЫВОДЫ:
```

```
Все модели показали точность выше 93%
```

```
Разница между моделями незначительна (максимум 2.2%)
```

```
Для практического применения рекомендуется k-NN с k=9
```

```
Decision Tree может быть предпочтительнее если важна интерпретируемость
```

Вывод: На практике сравнили работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов(SVM).