Министерство образования Республики Беларусь  
Учреждение образования

«Брестский государственный технический университет»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ  
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Выполнил

В.Д.Головкина,

студент группы АС66

Проверил

А. А. Крощенко,

доц. кафедры ИИТ,

« \_\_ » \_\_\_\_\_\_\_\_\_\_2025 г.

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

• 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.

import numpy as np  # Импорт библиотеки NumPy для работы с массивами и числовыми операциями

import pandas as pd  # Импорт библиотеки pandas для загрузки и обработки табличных данных

import matplotlib.pyplot as plt  # Импорт библиотеки matplotlib для построения графиков

# Импорт функций из scikit-learn для разделения данных, кросс-валидации и настройки разбиений

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, StratifiedKFold

# Импорт стандартизатора признаков — приводит признаки к одному масштабу

from sklearn.preprocessing import StandardScaler

# Импорт Pipeline — позволяет объединить несколько шагов (например, масштабирование + модель) в одну цепочку

from sklearn.pipeline import Pipeline

# Импорт трёх моделей классификации

from sklearn.neighbors import KNeighborsClassifier  # Метод k-ближайших соседей

from sklearn.tree import DecisionTreeClassifier     # Дерево решений

from sklearn.svm import SVC                         # Метод опорных векторов (Support Vector Machine)

# Импорт метрики accuracy — точность классификации

from sklearn.metrics import accuracy\_score

# 1. Загрузка данных

url = 'https://drive.google.com/uc?id=1WNG53OkZav0xxCI\_D7Q5qNeERxMdRPfM'

df = pd.read\_csv(url)

# Преобразование целевой переменной в числа

df['variety'] = df['variety'].astype('category').cat.codes

# Разделение на признаки и целевую переменную

X = df.drop(columns='variety')

y = df['variety']

# Фиксируем random\_state для воспроизводимости

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42, stratify=y

)

# 2. Подбор оптимального k с помощью кросс-валидации на обучающей выборке

k\_values = range(1, 21)

cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

mean\_scores = []

std\_scores = []

for k in k\_values:

    pipe = Pipeline([

        ('scaler', StandardScaler()),

        ('knn', KNeighborsClassifier(n\_neighbors=k))

    ])

    scores = cross\_val\_score(pipe, X\_train, y\_train, cv=cv, scoring='accuracy')

    mean\_scores.append(scores.mean())

    std\_scores.append(scores.std())

mean\_scores = np.array(mean\_scores)

std\_scores = np.array(std\_scores)

best\_k = k\_values[int(np.argmax(mean\_scores))]

print(f"Лучшее значение k по кросс-валидации (на обучающей выборке): {best\_k}")

# График: средняя точность ± std

plt.figure(figsize=(8, 5))

plt.plot(list(k\_values), mean\_scores, marker='o', label='mean CV accuracy')

plt.fill\_between(list(k\_values), mean\_scores - std\_scores, mean\_scores + std\_scores, alpha=0.2)

plt.scatter([best\_k], [mean\_scores[int(np.argmax(mean\_scores))]], color='red', zorder=5, label=f'best k={best\_k}')

plt.xticks(list(k\_values))

plt.xlabel('k (количество соседей)')

plt.ylabel('CV Accuracy (mean)')

plt.grid(True)

plt.legend()

plt.tight\_layout()

plt.show()

# 3. Обучение финальных моделей и оценка на тесте

models = {

    'k-NN': Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier(n\_neighbors=best\_k))]),

    'Decision Tree': Pipeline([('scaler', StandardScaler()), ('dt', DecisionTreeClassifier(random\_state=42))]),

    'SVM': Pipeline([('scaler', StandardScaler()), ('svm', SVC())])

}

results = {}

for name, model in models.items():

    model.fit(X\_train, y\_train)

    y\_pred = model.predict(X\_test)

    acc = accuracy\_score(y\_test, y\_pred)

    results[name] = acc

    print(f"{name} Accuracy on test set: {acc:.4f}")

# 4. Визуализация сравнения

plt.figure(figsize=(7, 4))

names = list(results.keys())

values = [results[n] for n in names]

bars = plt.bar(names, values, color=['tab:blue', 'tab:green', 'tab:orange'])

plt.ylim(0.0, 1.0)

plt.ylabel('Accuracy (test)')

plt.title('Сравнение точности моделей на тестовой выборке')

# Аннотация точности над столбиками

for bar, val in zip(bars, values):

    plt.text(bar.get\_x() + bar.get\_width() / 2, val + 0.01, f"{val:.3f}", ha='center', va='bottom')

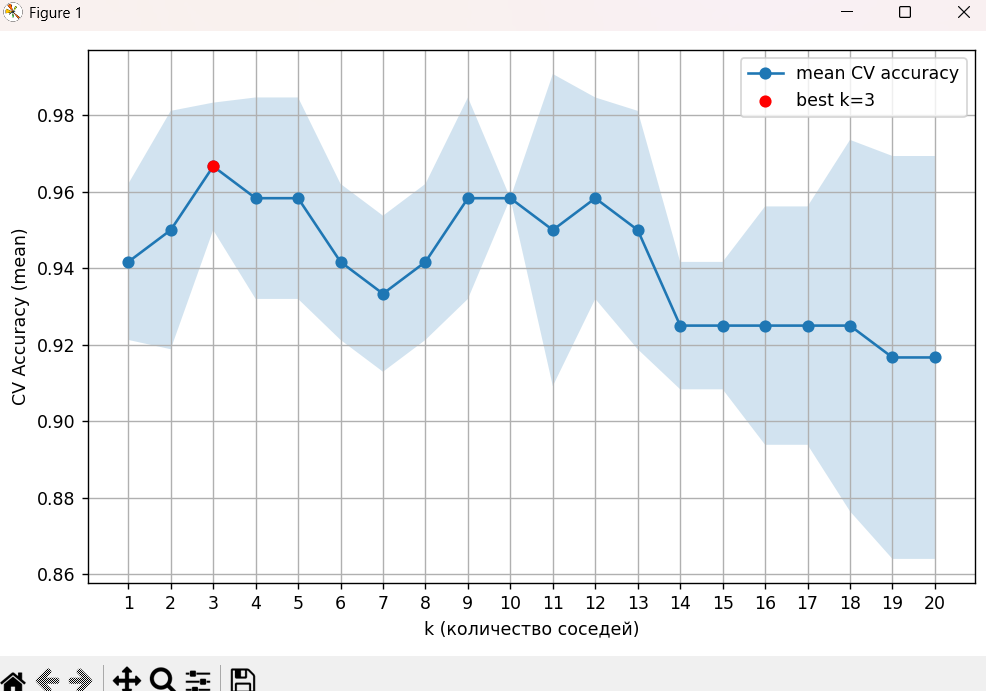
plt.tight\_layout()

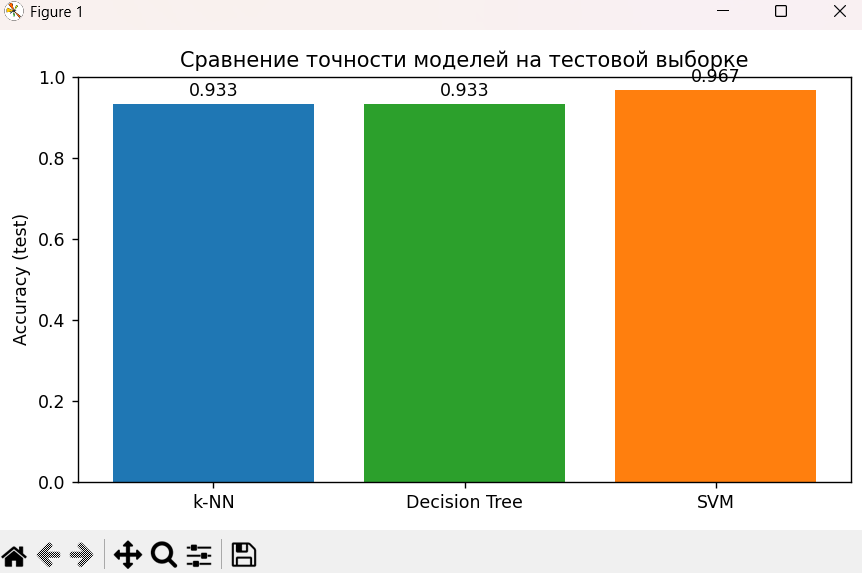
plt.show()

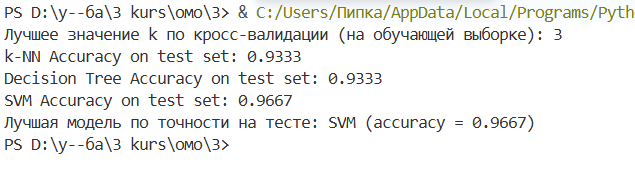
# 5. Вывод лучшей модели по тестовой точности

best\_model\_name = max(results, key=results.get)

print(f"Лучшая модель по точности на тесте: {best\_model\_name} (accuracy = {results[best\_model\_name]:.4f})")







Вывод: я изучила и на практике сравнила работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научилась подбирать гиперпараметры моделей и оценивать их влияние на результат. По accuracy на этом датасете модели дают практически одинаковый высокий результат; если нужен выбор — я бы выбирала Decision Tree для интерпретируемости, а для производительности и контроля — SVM/k-NN с дальнейшей тонкой настройкой гиперпараметров.