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.

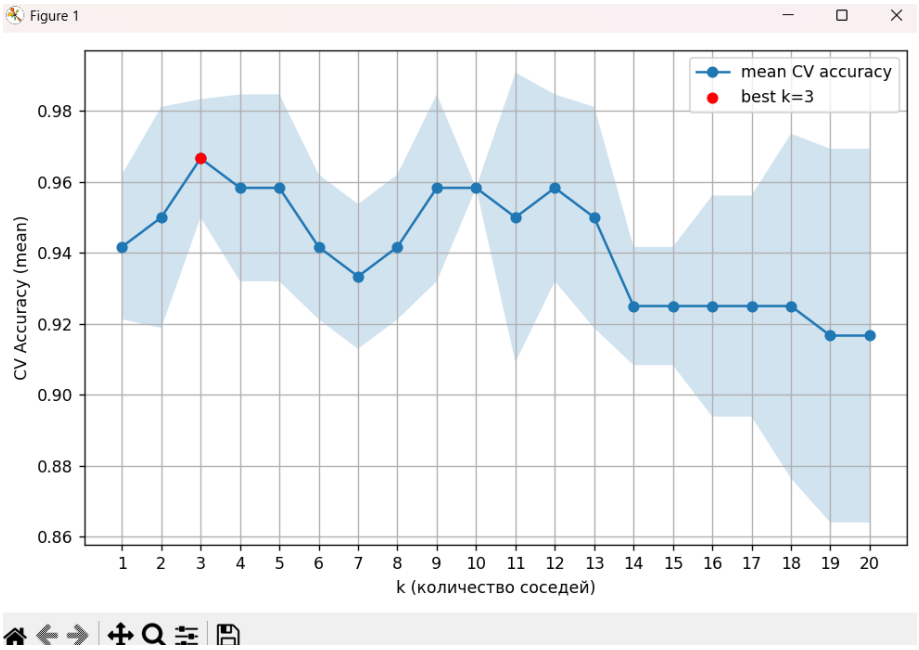
Брест 2025

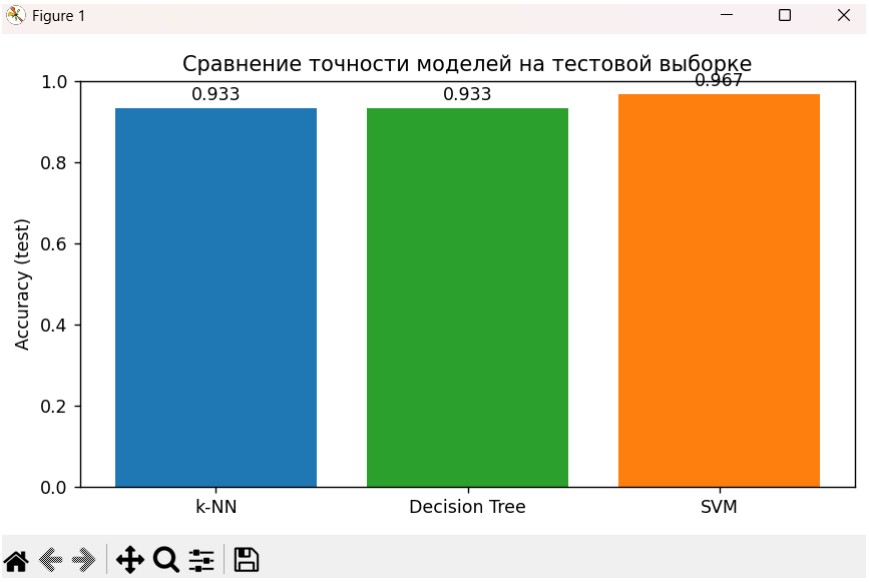
Цель работы: На практике сравнить работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов
(SVM). Научиться подбирать гиперпараметры моделей и оценивать их влияние на результат.
Вариант 1
• Iris
• Определить вид ириса (setosa, versicolor, 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
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from pathlib import Path

try:
    df = pd.read_csv('iris.csv')
    print(" Данные успешно загружены из iris.csv")
except FileNotFoundError:
    #try:
        # Альтернативный путь
        #file_path = Path(r"D:\y--6a\3 kurs\omo\3\iris.csv")
        #df = pd.read_csv(file_path)
        #print(f"Данные успешно загружены из: {file_path}")
    #except FileNotFoundError:
        print(" Файл не найден. Проверьте путь к файлу.")
        exit()

#df = pd.read_csv(file_path)
#print(f" Данные успешно загружены из: {file_path}")

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

X = df.drop(columns='variety')
y = df['variety']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```
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"\n Лучшее значение k по кросс-валидации: {best_k}")

# Гр: ср точность ± std
plt.figure(figsize=(8, 5))
plt.plot(list(k_values), mean_scores, marker='o', label='Средняя точность')
plt.fill_between(list(k_values), mean_scores - std_scores, mean_scores + std_scores, al-
pha=0.2)
plt.scatter([best_k], [mean_scores[int(np.argmax(mean_scores))]], color='red', zorder=5,
label=f'Лучшее k={best_k}')
plt.xticks(list(k_values))
plt.xlabel('k (количество соседей)')
plt.ylabel('Точность (CV)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

# 3. Обуч и оц на тесте
models = {
    'k-NN': Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier(n_neigh-
bors=best_k))]),
    'Decision Tree': Pipeline([('scaler', StandardScaler()), ('dt', DecisionTreeClassi-
fier(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}")
```

```
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')

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()

best_model_name = max(results, key=results.get)
print(f"\n Лучшая модель по точности: {best_model_name} (accuracy = {re-
sults[best_model_name]:.4f})")
```





```
PS D:\y--6a\3 kurs\omo\3> & C:/Users/Пипка/AppData/Local/Programs/Pyth
Лучшее значение k по кросс-валидации (на обучающей выборке): 3
k-NN Accuracy on test set: 0.9333
Decision Tree Accuracy on test set: 0.9333
SVM Accuracy on test set: 0.9667
Лучшая модель по точности на тесте: SVM (accuracy = 0.9667)
PS D:\y--6a\3 kurs\omo\3>
```

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.