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

Отчёт по лабораторной работе №3

Выполнил:
Студент 3 курса
Группы АС-65
Романюк Д. А.
Проверил:
Крощенко А. А.

Брест 2025

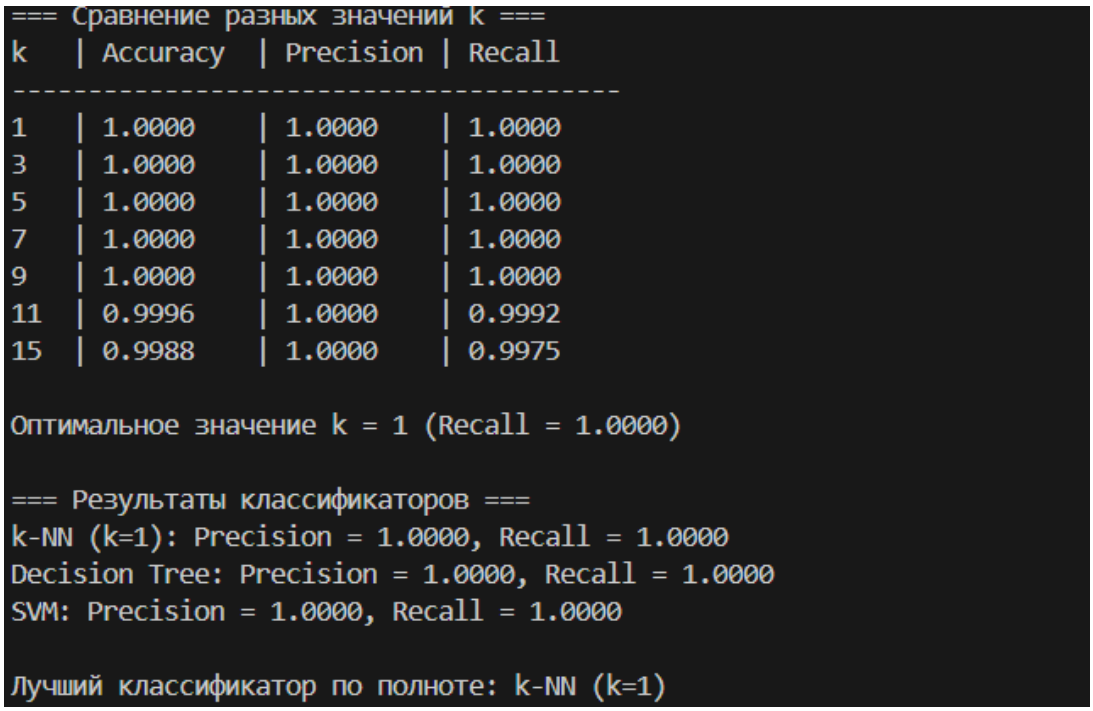
Цель: На практике сравнить работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научиться подбирать гиперпараметры моделей и оценивать их влияние на результат.

Вариант 5

• Mushroom Classification
• Определить, является ли гриб ядовитым или съедобным
• Задания:
1. Загрузите данные и преобразуйте все категориальные признаки в числовые (например, с помощью One-Hot Encoding);
2. Разделите данные на обучающую и тестовую части;
3. Обучите классификаторы k-NN, Decision Tree и SVM;
4. Рассчитайте точность и полноту (precision и recall) для класса "ядовитый";
5. Сделайте вывод о том, какой классификатор лучше всего справляется с этой задачей, где цена ошибки очень высока.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, accuracy_score

df = pd.read_csv("mushrooms.csv")

X = df.drop('class', axis=1)
y = df['class']

encoder = OneHotEncoder(sparse_output=False)
X_encoded = encoder.fit_transform(X)

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

k_values = [1, 3, 5, 7, 9, 11, 15]
k_results = []

for k in k_values:
    model_k = KNeighborsClassifier(n_neighbors=k)
    model_k.fit(X_train, y_train)
    y_pred_k = model_k.predict(X_test)

    acc = accuracy_score(y_test, y_pred_k)
    prec = precision_score(y_test, y_pred_k, pos_label='p')
    rec = recall_score(y_test, y_pred_k, pos_label='p')
```

```
        k_results.append((k, acc, prec, rec))

    print("=== Сравнение разных значений k ===")
    print(f"{'k':<3} | {'Accuracy':<9} | {'Precision':<9} | {'Recall':<9}")
    print("-" * 40)
    for k, acc, prec, rec in k_results:
        print(f"{k:<3} | {acc:<9.4f} | {prec:<9.4f} | {rec:<9.4f}")

    best_k = max(k_results, key=lambda x: x[3])
    print(f"\nОптимальное значение k = {best_k[0]} (Recall = {best_k[3]:.4f})")

    models = {
                                        f"k-NN (k={best_k[0]})": KNeighborsClassifier(n_neighbors=best_k[0]),
        "Decision Tree": DecisionTreeClassifier(random_state=42),
        "SVM": SVC(kernel='rbf', random_state=42)
    }

    results = {}

    for name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        precision = precision_score(y_test, y_pred, pos_label='p')
        recall = recall_score(y_test, y_pred, pos_label='p')

        results[name] = {'Precision': precision, 'Recall': recall}

    print("\n=== Результаты классификаторов ===")
    for model_name, metrics in results.items():
        print(f"{model_name}: Precision = {metrics['Precision']:.4f}, Recall = {metrics['Recall']:.4f}")

    best_model = max(results.items(), key=lambda x: x[1]['Recall'])
    print(f"\nЛучший классификатор по полноте: {best_model[0]}")
```



Вывод: Все модели показывают почти идеальные результаты, но если выбирать с точки зрения *безопасности* (максимизация Recall для класса “ядовитый”) — лучше взять **k-NN**.