МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

Специальность АС-66

Выполнила
Е. С. Неруш,
студент группы АС-66

Проверил
А. А. Крощенко,
ст. преп. 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.

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 10

• Adult Census Income

• Предсказать, превышает ли доход человека $50 тыс. в год

• Задания:

1. Загрузите данные, обработайте пропуски и категориальные признаки;
2. Разделите данные на обучающую и тестовую выборки;
3. Обучите k-NN, Decision Tree и SVM;
4. Сравните модели по метрике precision для класса ">50K";
5. Определите, какой алгоритм лучше всего идентифицирует людей с высоким доходом.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC

# 1. Загрузка и предобработка данных
df = pd.read_csv("E:/Projects/ml_as66/reports/Nerush/lab3/src/adult.csv",
na_values='?')
print(f"Загружено строк: {len(df)}")

# Удаление строк с пропущенными значениями
df = df.dropna()
print(f"После удаления пропусков осталось строк: {len(df)}")
```

```
# Преобразование целевой переменной
df['income'] = df['income'].map({'<=50K': 0, '>50K': 1})
print("Целевая переменная преобразована в бинарный формат.")

# Кодирование категориальных признаков
categorical_cols = df.select_dtypes(include='object').columns
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
print(f"После кодирования категориальных признаков: {df.shape[1]} признаков.")

# 2. Разделение на обучающую и тестовую выборки
X = df.drop('income', axis=1)
y = df['income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
print("Данные разделены на обучающую и тестовую выборки.")
```

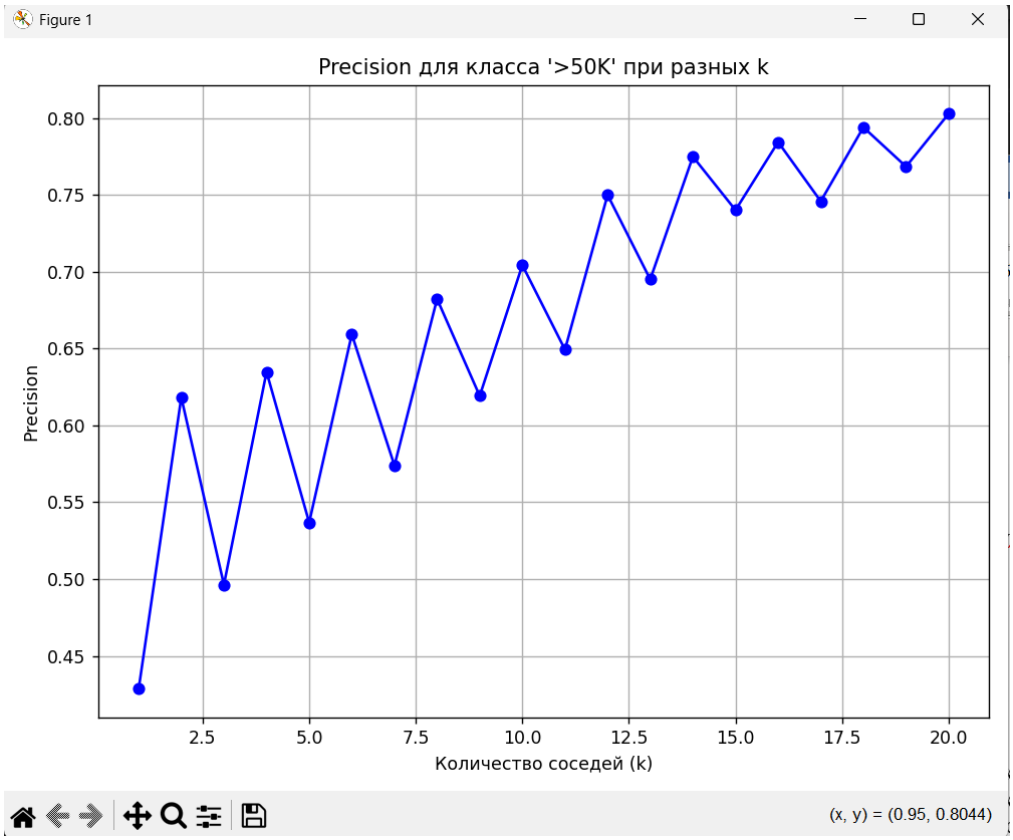
```
Загружено строк: 32561
После удаления пропусков осталось строк: 30162
Целевая переменная преобразована в бинарный формат.
После кодирования категориальных признаков: 97 признаков.
Данные разделены на обучающую и тестовую выборки.
```

```
# 3. Обучение моделей
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_knn = knn.predict(X_test)
precision_knn = precision_score(y_test, y_knn)

tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
y_tree = tree.predict(X_test)
precision_tree = precision_score(y_test, y_tree)

svm = LinearSVC(max_iter=10000)
svm.fit(X_train, y_train)
y_svm = svm.predict(X_test)
precision_svm = precision_score(y_test, y_svm)

# 4. Сравнение моделей
print("\nPrecision для класса '>50K'")
print(f"k-NN (k=5): {precision_knn:.3f}")
print(f"Decision Tree: {precision_tree:.3f}")
print(f"LinearSVC: {precision_svm:.3f}")
```

```
Precision для класса '>50K'
k-NN (k=5): 0.537
Decision Tree: 0.598
LinearSVC: 0.706
```

```
# 5. Исследование влияния параметра k
k_values = range(1, 21)
precisions = []
```

```
for k in k_values:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    p = precision_score(y_test, y_pred)
    precisions.append(p)

plt.figure(figsize=(8, 6))
plt.plot(k_values, precisions, marker='o', linestyle='-', color='blue')
plt.title("Precision для класса '>50K' при разных k")
plt.xlabel("Количество соседей (k)")
plt.ylabel("Precision")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Вывод: Научился применять классические алгоритмы классификации — метод k-ближайших соседей, дерево решений и метод опорных векторов — для решения задачи бинарной классификации. Получил практический опыт загрузки и предобработки данных, кодирования категориальных признаков, построения моделей с использованием библиотек Pandas, Scikit-learn и визуализации результатов с помощью Matplotlib и Seaborn. Освоил подбор гиперпараметров, расчёт метрик качества моделей и интерпретацию precision для оценки способности алгоритмов выявлять целевой класс.