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
Макарский А. Э.
Проверил:
Крощенко А. А.

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 10

Ход работы:

Задание:

• 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";

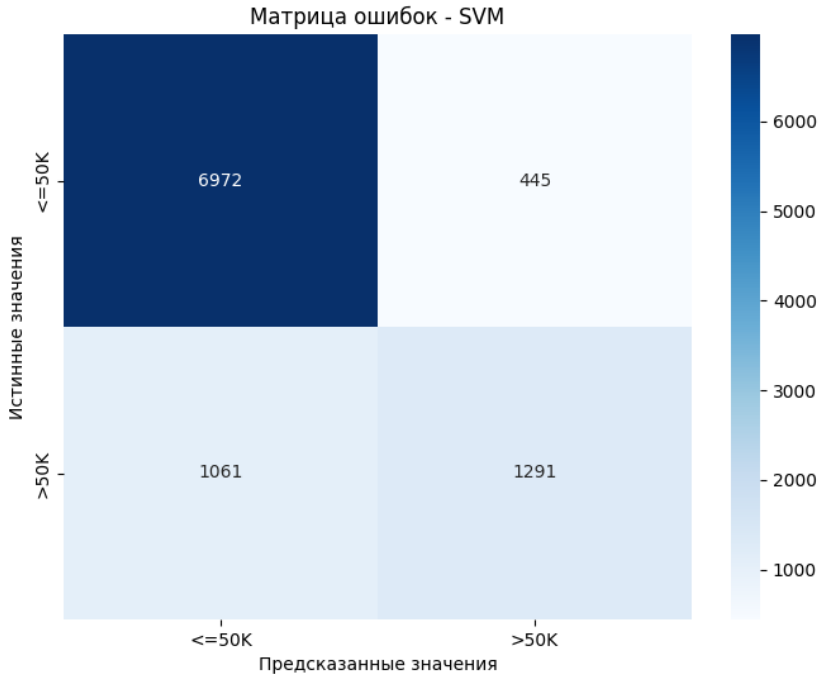
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import precision_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('adult.csv')

print("Информация о датасете:")
print(data.info())
print("\nПервые 5 строк:")
print(data.head())
print("\nПропущенные значения:")
print(data.isnull().sum())

data = data.replace('?', np.nan)

categorical_columns = ['workclass', 'occupation', 'native.country']
for col in categorical_columns:
    mode_value = data[col].mode()[0]
    data[col] = data[col].fillna(mode_value)

print("\nПропуски после обработки:")
print(data.isnull().sum())

label_encoders = {}
categorical_features = ['workclass', 'education', 'marital.status', 'occupation',
                        'relationship', 'race', 'sex', 'native.country']

for feature in categorical_features:
    le = LabelEncoder()
    data[feature] = le.fit_transform(data[feature].astype(str))
    label_encoders[feature] = le

income_encoder = LabelEncoder()
data['income'] = income_encoder.fit_transform(data['income'])

X = data.drop('income', axis=1)
y = data['income']
```

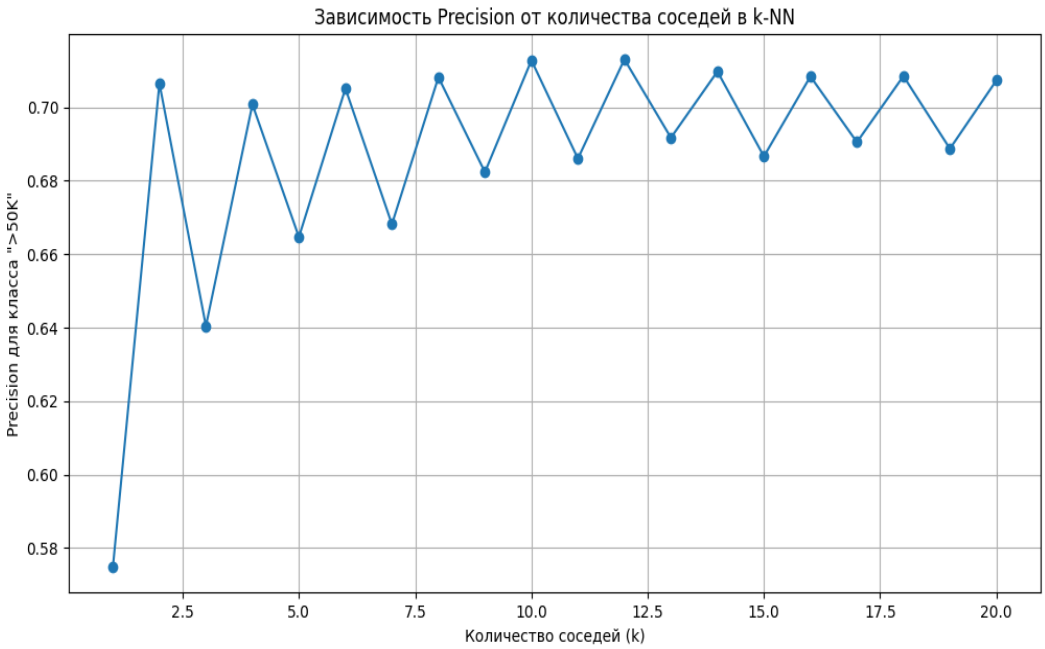
```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    print(f"\nРазмеры выборок:")
    print(f"Обучающая: {X_train.shape}")
    print(f"Тестовая: {X_test.shape}")
    print(f"Распределение классов в обучающей выборке: {np.bincount(y_train)}")
    print(f"Распределение классов в тестовой выборке: {np.bincount(y_test)}")

    k_values = range(1, 21)
    knn_precisions = []

    print("\nИсследование k-NN с разным количеством соседей:")
    for k in k_values:
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train_scaled, y_train)
        y_pred_knn = knn.predict(X_test_scaled)
        precision = precision_score(y_test, y_pred_knn, pos_label=1)  # precision для
класса ">50K"
        knn_precisions.append(precision)
        print(f"k={k}: Precision = {precision:.4f}")

    plt.figure(figsize=(12, 6))
    plt.plot(k_values, knn_precisions, marker='o')
    plt.title('Зависимость Precision от количества соседей в k-NN')
    plt.xlabel('Количество соседей (k)')
    plt.ylabel('Precision для класса ">50K"')
    plt.grid(True)
    plt.savefig('knn_precision_analysis.png')  # Сохраняем график
    plt.show()

    best_k = k_values[np.argmax(knn_precisions)]
    best_knn_precision = max(knn_precisions)
    print(f"\nЛучший результат k-NN: k={best_k}, Precision = {best_knn_precision:.4f}")

    dt = DecisionTreeClassifier(random_state=42)
    dt.fit(X_train, y_train)
    y_pred_dt = dt.predict(X_test)
    dt_precision = precision_score(y_test, y_pred_dt, pos_label=1)

    print(f"\nDecision Tree Precision: {dt_precision:.4f}")

    svm = SVC(random_state=42, probability=True)
    svm.fit(X_train_scaled, y_train)
    y_pred_svm = svm.predict(X_test_scaled)
    svm_precision = precision_score(y_test, y_pred_svm, pos_label=1)

    print(f"SVM Precision: {svm_precision:.4f}")

    models_comparison = pd.DataFrame({
        'Модель': ['k-NN', 'Decision Tree', 'SVM'],
        'Precision для ">50K"': [best_knn_precision, dt_precision, svm_precision],
        'Лучшие параметры': [f'k={best_k}', 'default', 'default']
    })

    print("\nСравнение моделей:")
    print(models_comparison)

    plt.figure(figsize=(10, 6))
    models = ['k-NN', 'Decision Tree', 'SVM']
    precisions = [best_knn_precision, dt_precision, svm_precision]
```

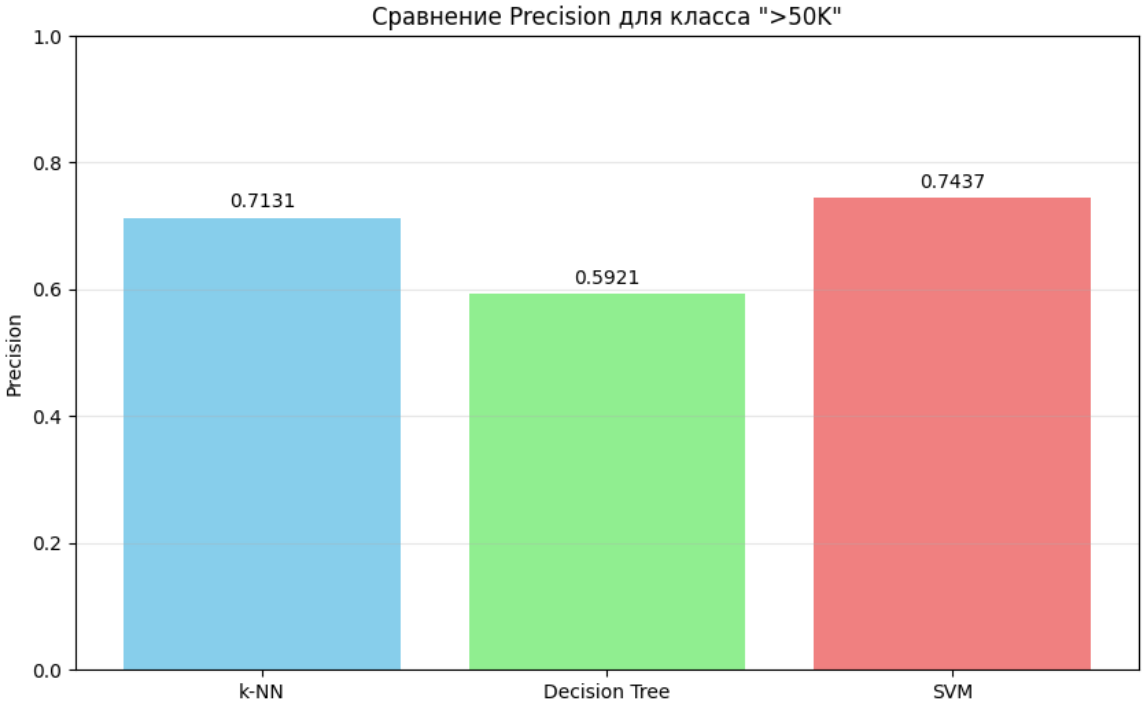
```
    bars = plt.bar(models, precisions, color=['skyblue', 'lightgreen', 'lightcoral'])
    plt.title('Сравнение Precision для класса ">50K"')
    plt.ylabel('Precision')
    plt.ylim(0, 1)

    for bar, precision in zip(bars, precisions):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
                 f'{precision:.4f}', ha='center', va='bottom')

    plt.grid(axis='y', alpha=0.3)
    plt.savefig('models_comparison.png')
    plt.show()

    best_model_idx = np.argmax(precisions)
    best_model_name = models[best_model_idx]
    best_model_precision = precisions[best_model_idx]

    print(f"\nЛучшая модель: {best_model_name} с Precision = {best_model_precision:.4f}")

    if best_model_name == 'k-NN':
        best_model = KNeighborsClassifier(n_neighbors=best_k)
        best_model.fit(X_train_scaled, y_train)
        y_pred_best = best_model.predict(X_test_scaled)
    elif best_model_name == 'Decision Tree':
        best_model = dt
        y_pred_best = y_pred_dt
    else:
        best_model = svm
        y_pred_best = y_pred_svm

    print(f"\nДетальный отчет для {best_model_name}:")
    print(classification_report(y_test, y_pred_best,
                                target_names=['<=50K', '>50K']))

    plt.figure(figsize=(8, 6))
    cm = confusion_matrix(y_test, y_pred_best)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['<=50K', '>50K'],
                yticklabels=['<=50K', '>50K'])
    plt.title(f'Матрица ошибок - {best_model_name}')
    plt.ylabel('Истинные значения')
    plt.xlabel('Предсказанные значения')
    plt.savefig('confusion_matrix.png')
    plt.show()

    if best_model_name == 'Decision Tree':
        feature_importance = pd.DataFrame({
            'feature': X.columns,
            'importance': dt.feature_importances_
        }).sort_values('importance', ascending=False)

        plt.figure(figsize=(10, 6))
        plt.barh(feature_importance['feature'][:10],
feature_importance['importance'][:10])
        plt.title('Топ-10 важнейших признаков (Decision Tree)')
        plt.xlabel('Важность')
        plt.tight_layout()
        plt.savefig('feature_importance.png')
        plt.show()

    print("\n" + "=" * 50)
    print("ВЫВОДЫ:")
    print("=" * 50)
    print(f"1. Лучшая модель для идентификации людей с высоким доходом (>50K):
{best_model_name}")
```

```
    print(f"2. Precision лучшей модели: {best_model_precision:.4f}")
    print(f"3. k-NN показал лучшие результаты при k={best_k}")
    print(f"4. Все модели демонстрируют разную эффективность в идентификации
высокодоходной группы")

    results_summary = f"""
    РЕЗУЛЬТАТЫ АНАЛИЗА:
    ==================
    Лучшая модель: {best_model_name}
    Precision для класса ">50K": {best_model_precision:.4f}
    Оптимальное k для k-NN: {best_k}

    Сравнение моделей:
    - k-NN (k={best_k}): {best_knn_precision:.4f}
    - Decision Tree: {dt_precision:.4f}
    - SVM: {svm_precision:.4f}
    """

    with open('results_summary.txt', 'w', encoding='utf-8') as f:
        f.write(results_summary)

    print("\nРезультаты сохранены в файлы:")
    print("- knn_precision_analysis.png")
    print("- models_comparison.png")
    print("- confusion_matrix.png")
    print("- results_summary.txt")
```

```
РЕЗУЛЬТАТЫ АНАЛИЗА:
==================
Лучшая модель: SVM
Precision для класса ">50K": 0.7437
Оптимальное k для k-NN: 12

Сравнение моделей:
- k-NN (k=12): 0.7131
- Decision Tree: 0.5921
- SVM: 0.7437
```







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