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

Выполнил
А. С. Рогожин,
студент группы АС-66

Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«___» _____2025 г.

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.

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:

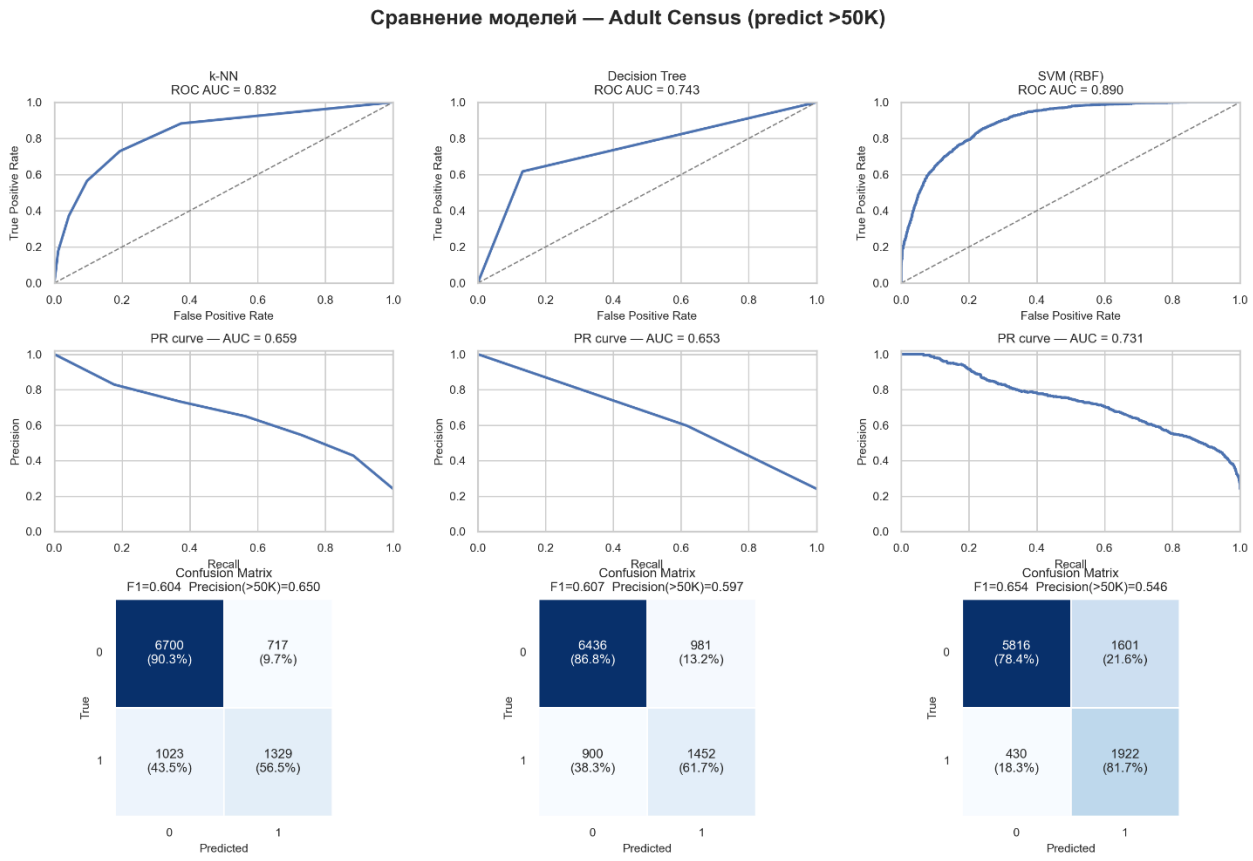
1. Загрузите данные, обработайте пропуски и категориальные признаки;
2. Разделите данные на обучающую и тестовую выборки;
3. Обучите k-NN, Decision Tree и SVM;
4. Сравните модели по метрике precision для класса ">50K";
5. Определите, какой алгоритм лучше всего идентифицирует людей с высоким доходом.

```
import matplotlib
matplotlib.use("Agg")

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_curve,
    roc_auc_score,
    precision_recall_curve,
    auc,
    f1_score,
    precision_score
)

# Параметр: показывать окно (True) или только сохранять (False)
SHOW_PLOT = False
OUTFILE = "comparison_adult_clean.png"

# --- Загрузка данных (как у тебя) ---
```

```
script_dir = os.path.dirname(os.path.abspath(__file__))
csv_path = os.path.join(script_dir, "adult.csv")
df = pd.read_csv(csv_path, sep=None, engine="python")  # автоопределение sep
df['income'] = df['income'].astype(str).str.strip()
df['income'] = df['income'].replace({'>50K.': '>50K', '<=50K.': '<=50K'})
df['target'] = (df['income'] == '>50K').astype(int)
df = df.drop(columns=['income'])

X = df.drop(columns=['target'])
y = df['target']
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
X_encoded = pd.get_dummies(X, columns=cat_cols, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, 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)

models = {
    "k-NN": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM (RBF)": SVC(kernel='rbf', class_weight='balanced', probability=True,
     random_state=42)
}

results = {}
for name, model in models.items():
    if name == "Decision Tree":
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_proba = model.predict_proba(X_test)[:, 1]
    else:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        y_proba = model.predict_proba(X_test_scaled)[:, 1]

    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_proba)
    prec, rec, _ = precision_recall_curve(y_test, y_proba)
    pr_auc = auc(rec, prec)
    cm = confusion_matrix(y_test, y_pred)
    precision_pos = precision_score(y_test, y_pred, pos_label=1)

    fpr, tpr, _ = roc_curve(y_test, y_proba)

    results[name] = {
        "f1": f1,
        "roc_auc": roc_auc,
        "pr_auc": pr_auc,
        "fpr": fpr,
        "tpr": tpr,
        "precision_curve": prec,
        "recall_curve": rec,
        "cm": cm,
        "report": classification_report(y_test, y_pred, digits=4),
        "precision_for_>50K": precision_pos
    }

sns.set(style="whitegrid")
n_models = len(models)
col_width = 5.5
fig_height = 10
fig = plt.figure(figsize=(col_width * n_models, fig_height))
```

```
gs = gridspec.GridSpec(nrows=3, ncols=n_models, height_ratios=[1, 1, 1.2],
     hspace=0.35, wspace=0.25)

title_font = {"fontsize": 12, "fontweight": "bold"}
subtitle_font = {"fontsize": 10}
label_font = {"fontsize": 10}
tick_fontsize = 9

model_names = list(results.keys())
for idx, name in enumerate(model_names):
    res = results[name]

    # ROC
    ax_roc = fig.add_subplot(gs[0, idx])
    ax_roc.plot(res["fpr"], res["tpr"], lw=2)
    ax_roc.plot([0, 1], [0, 1], linestyle="--", lw=1, color="gray")
    ax_roc.set_xlim(0, 1)
    ax_roc.set_ylim(0, 1)
    ax_roc.set_xlabel("False Positive Rate", fontsize=9)
    ax_roc.set_ylabel("True Positive Rate", fontsize=9)
    ax_roc.set_title(f"{name}\nROC AUC = {res['roc_auc']:.3f}", **subtitle_font)
    ax_roc.tick_params(labelsize=tick_fontsize)

    # PR
    ax_pr = fig.add_subplot(gs[1, idx])
    ax_pr.plot(res["recall_curve"], res["precision_curve"], lw=2)
    ax_pr.set_xlim(0, 1)
    ax_pr.set_ylim(0, 1.02)
    ax_pr.set_xlabel("Recall", fontsize=9)
    ax_pr.set_ylabel("Precision", fontsize=9)
    ax_pr.set_title(f"PR curve — AUC = {res['pr_auc']:.3f}", fontsize=10)
    ax_pr.tick_params(labelsize=tick_fontsize)

    # Confusion matrix (counts + проценты)
    ax_cm = fig.add_subplot(gs[2, idx])
    cm = res["cm"]
    cm_sum = cm.sum()
    # percent matrix relative to true labels (rows)
    row_sums = cm.sum(axis=1, keepdims=True)
    with np.errstate(divide='ignore', invalid='ignore'):
        cm_pct = (cm / row_sums) * 100
        cm_pct = np.nan_to_num(cm_pct)

    annot = np.empty_like(cm).astype(object)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            annot[i, j] = f"{cm[i, j]:d}\n({cm_pct[i, j]:.1f}%)"

    sns.heatmap(cm, annot=annot, fmt="", cmap="Blues", cbar=False, ax=ax_cm,
                annot_kws={"size": 10}, linewidths=0.5, linecolor="white",
     square=True)
    ax_cm.set_xlabel("Predicted", fontsize=9)
    ax_cm.set_ylabel("True", fontsize=9)
    ax_cm.set_title(f"Confusion
     Matrix\nF1={res['f1']:.3f}  Precision(>50K)={res['precision_for_>50K']:.3f}",
     fontsize=10)
    ax_cm.tick_params(labelsize=tick_fontsize)
    ax_cm.set_xticklabels(["0", "1"])
    ax_cm.set_yticklabels(["0", "1"], rotation=0)

fig.suptitle("Сравнение моделей — Adult Census (predict >50K)", fontsize=16,
     fontweight="bold", y=0.98)
fig.tight_layout(rect=[0, 0, 1, 0.95])
fig.savefig(OUTFILE, dpi=200)
plt.close(fig)
```

```
print("График сохранён в", os.path.abspath(OUTFILE))
print("\nМетрики по моделям (на тестовой выборке):\n")
for name, res in results.items():
    print(f"=== {name} ===")
    print(res["report"])
    print(f"Precision (класс '>50K'): {res['precision_for_>50K']:.4f}")
    print(f"ROC AUC: {res['roc_auc']:.4f}, PR AUC: {res['pr_auc']:.4f}, F1:
     {res['f1']:.4f}")
    print("-" * 60)

best_by_precision = max(results.items(), key=lambda kv: kv[1]['precision_for_>50K'])
print(f"\nЛучший алгоритм по precision (класс '>50K'): {best_by_precision[0]}
     (precision = {best_by_precision[1]['precision_for_>50K']:.4f})")
if SHOW_PLOT:
    import matplotlib.pyplot as plt
    img = plt.imread(OUTFILE)
    plt.figure(figsize=(12, 6))
    plt.imshow(img)
    plt.axis("off")
    plt.show()
```



```
=== k-NN ===
              precision    recall  f1-score   support

           0     0.8675    0.9033    0.8851      7417
           1     0.6496    0.5651    0.6044      2352

    accuracy                         0.8219      9769
   macro avg     0.7585    0.7342    0.7447      9769
weighted avg     0.8151    0.8219    0.8175      9769

Precision (класс '>50K'): 0.6496
ROC AUC: 0.8322, PR AUC: 0.6593, F1: 0.6044
------------------------------------------------------------
=== Decision Tree ===
              precision    recall  f1-score   support

           0     0.8773    0.8677    0.8725      7417
           1     0.5968    0.6173    0.6069      2352

    accuracy                         0.8075      9769
   macro avg     0.7371    0.7425    0.7397      9769
weighted avg     0.8098    0.8075    0.8086      9769

Precision (класс '>50K'): 0.5968
ROC AUC: 0.7425, PR AUC: 0.6531, F1: 0.6069
------------------------------------------------------------
=== SVM (RBF) ===
              precision    recall  f1-score   support

           0     0.9312    0.7841    0.8514      7417
           1     0.5456    0.8172    0.6543      2352

    accuracy                         0.7921      9769
   macro avg     0.7384    0.8007    0.7528      9769
weighted avg     0.8383    0.7921    0.8039      9769

Precision (класс '>50K'): 0.5456
ROC AUC: 0.8901, PR AUC: 0.7309, F1: 0.6543
------------------------------------------------------------

Лучший алгоритм по precision (класс '>50K'): k-NN (precision = 0.6496)
```

Вывод: я изучил и на практике сравнил 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.