МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность АС-66

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

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

Брест 2025

Цель работы: Получить практические навыки работы с данными с использованием библиотек **Pandas** для манипуляции и **Matplotlib** для визуализации. Научиться выполнять основные шаги предварительной обработки данных, такие как очистка, нормализация и работа с различными типами признаков.

Вариант 10.

Выборка German Credit Data. Содержит информацию о заемщиках, включая их кредитную историю, цель кредита, возраст, и оценку кредитоспособности (хороший/плохой).

Задачи:

1. Загрузите данные и выведите информацию о них.
2. Проанализируйте распределение цели кредита (Purpose).
Визуализируйте 5 самых популярных целей.
3. Преобразуйте категориальные признаки Sex и Housing в числовой формат.
4. Постройте "ящик с усами" для Credit amount, чтобы сравнить суммы кредитов у "хороших" и "плохих" заемщиков.
5. Создайте сводную таблицу, показывающую средний возраст (Age) и среднюю длительность кредита (Duration) для каждой категории кредитной истории (Credit history).
6. Нормализуйте числовые столбцы Age, Credit amount, Duration.Код программы:

```
# analyze_german_credit.py
import os
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler

sns.set(style="whitegrid", font_scale=1.05)

# --- Настройки ---
DATA_PATH = "german_credit.csv"  # <- поменяй путь/имя файла при необходимости
OUT_DIR = "german_credit_output"
os.makedirs(OUT_DIR, exist_ok=True)


def load_data(path):
    if not os.path.exists(path):
        raise FileNotFoundError(f"Файл {path} не найден. Помести CSV/Excel файл и обнови DATA_PATH.")

    ext = os.path.splitext(path)[1].lower()

    if ext in [".xls", ".xlsx"]:
        df = pd.read_excel(path)
        # если получилась одна колонка с запятыми — пробуем перечитать как CSV
        if df.shape[1] == 1:
            first_col = df.columns[0]
            if isinstance(first_col, str) and "," in first_col:
                print("Похоже, это CSV внутри Excel — пробуем читать как CSV")
                try:
```

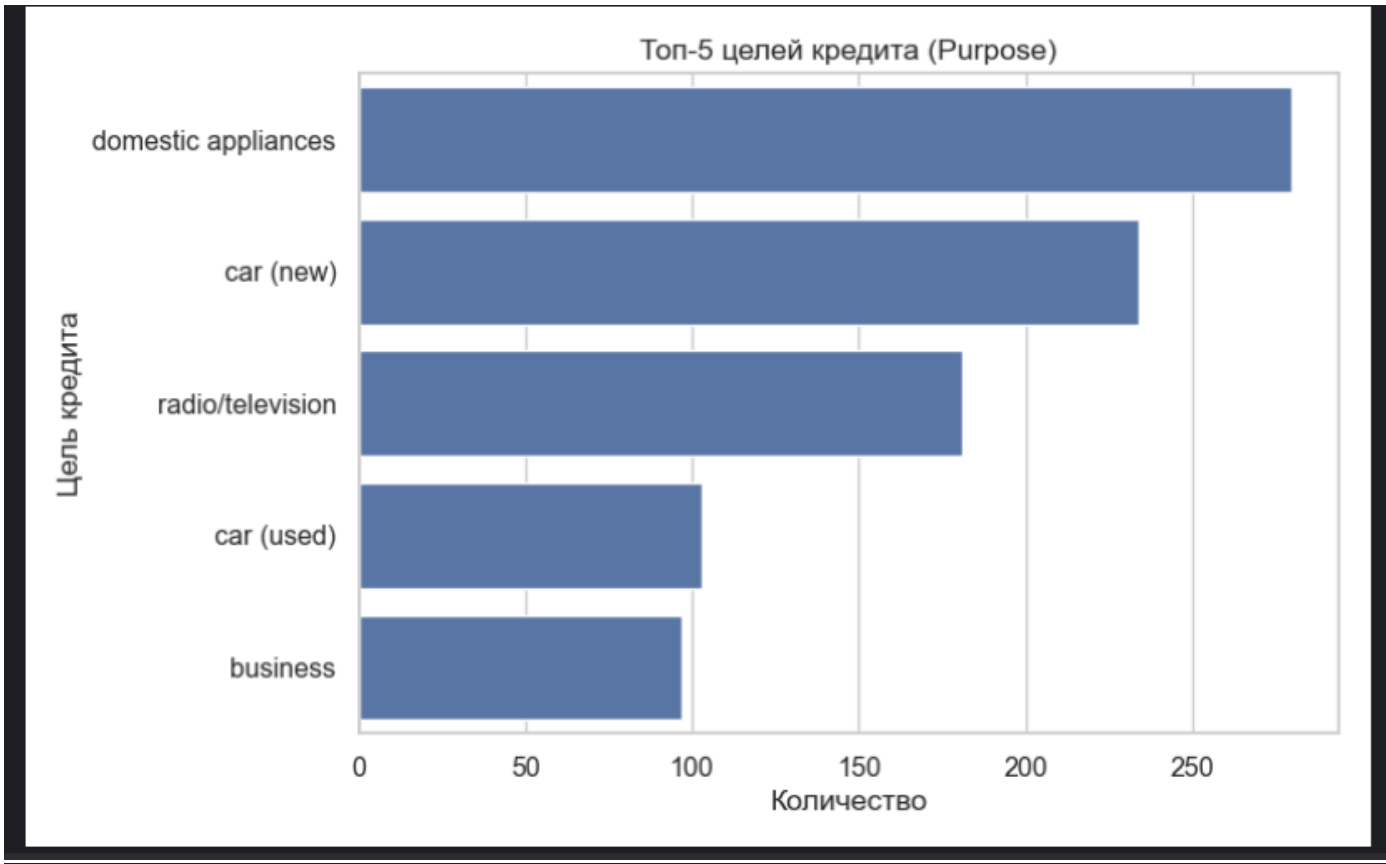
```
                    df = pd.read_csv(path)
                except Exception as e:
                    print("Не удалось прочитать как CSV, оставляем
Excel-версию:", e)
        else:
            df = pd.read_csv(path)

        return df


    def infer_and_prepare_columns(df):
        # Показываем колонки
        print("Колонки в наборе данных:\n", list(df.columns), "\n")

        colmap = {}
        cols = df.columns.str.lower()

        # Sex
        if "sex" in df.columns:
            colmap["sex"] = "Sex"
        else:
            cand = [c for c in df.columns if "personal" in c.lower() and
"sex" in c.lower() or "personal status" in c.lower()]
            if cand:
                colmap["sex"] = cand[0]
            else:
                for c in df.columns:
                    vals = df[c].dropna().astype(str).str.lower()
                    if vals.isin(["male", "female", "m", "f", "man",
"woman"]).any():
                        colmap["sex"] = c
                        break

        # Housing
        for name in ["Housing", "housing", "house"]:
            if name in df.columns:
                colmap["housing"] = name
                break
        if "housing" not in colmap:
            cand = [c for c in df.columns if "housing" in c.lower() or
"home" in c.lower()]
            if cand:
                colmap["housing"] = cand[0]

        # Risk
        for target in ["Risk", "risk", "class", "target", "credit_risk",
"creditability"]:
            if target in df.columns:
                colmap["risk"] = target
                break
        if "risk" not in colmap:
            for c in df.columns:
                vals = df[c].dropna().astype(str).str.lower().unique()
                if set(vals).intersection({"good", "bad", "g", "b"}):
                    colmap["risk"] = c
                    break

        # Purpose
        for name in ["Purpose", "purpose", "purpose of loan"]:
            if name in df.columns:
                colmap["purpose"] = name
                break
        if "purpose" not in colmap:
            cand = [c for c in df.columns if "purpose" in c.lower() or
"use" in c.lower()]
```

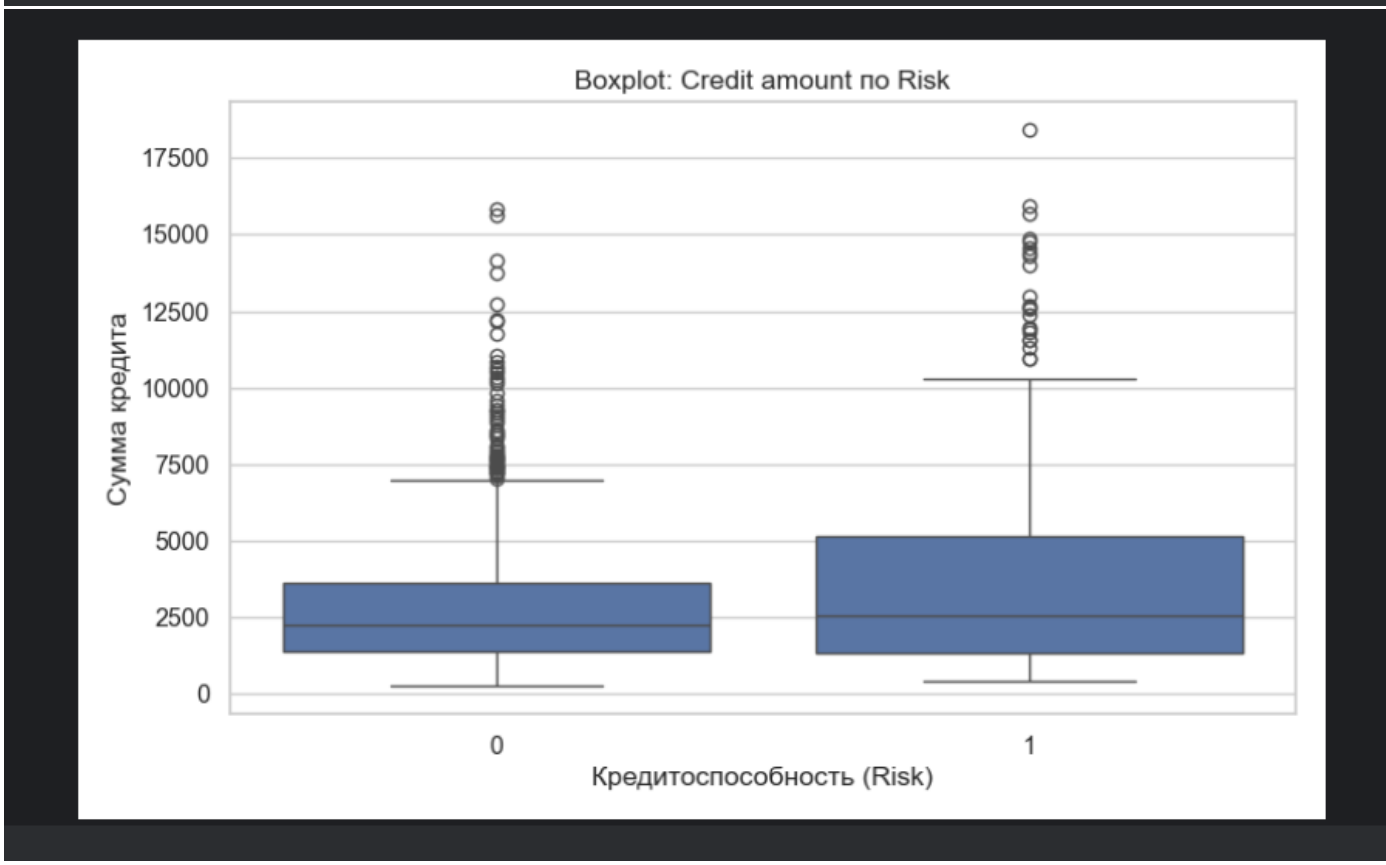
```
            if cand:
                colmap["purpose"] = cand[0]

        # Credit amount
        for name in ["Credit amount", "CreditAmount", "credit amount",
"credit_amount", "amount"]:
            if name in df.columns:
                colmap["credit_amount"] = name
                break
        if "credit_amount" not in colmap:
            cand = [c for c in df.columns if "credit" in c.lower() and
"amount" in c.lower()]
            if cand:
                colmap["credit_amount"] = cand[0]

        # Age
        for name in ["age", "customer_age"]:
            if name in df.columns:
                colmap["age"] = name
                break

        # Duration
        for name in ["Duration", "duration", "duration in month",
"duration_month"]:
            if name in df.columns:
                colmap["duration"] = name
                break
        if "duration" not in colmap:
            cand = [c for c in df.columns if "duration" in c.lower() or
"month" in c.lower() and "duration" in c.lower()]
            if cand:
                colmap["duration"] = cand[0]

        # Credit history
        for name in ["Credit history", "CreditHistory", "credit history",
"credit_history"]:
            if name in df.columns:
                colmap["credit_history"] = name
                break
        if "credit_history" not in colmap:
            cand = [c for c in df.columns if "credit" in c.lower() and
"history" in c.lower()]
            if cand:
                colmap["credit_history"] = cand[0]

        print("Найденные важные колонки (по возможности):", colmap, "\n")
        return colmap


    def exploratory_info(df):
        print("=== Общая информация ===")
        print(df.info())
        print("\n--- Количество пропусков по столбцам ---")
        print(df.isnull().sum())
        print("\n--- Основные статистические показатели для числовых
столбцов ---")
        try:
            print(df.describe().T[['mean', '50%',
'std']].rename(columns={"50%": "median"}))
        except Exception:
            print("Не удалось вывести describe() — возможно, нет числовых
колонок.")
        print("\n")
```

```
   def handle_missing(df):
       num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
       cat_cols = df.select_dtypes(include=["object",
"category"]).columns.tolist()
       for c in num_cols:
           if df[c].isnull().any():
               df[c] = df[c].fillna(df[c].mean())
       for c in cat_cols:
           if df[c].isnull().any():
               df[c] = df[c].fillna(df[c].mode().iloc[0])
       return df


   def extract_sex(df, sex_col_name):
       if sex_col_name not in df.columns:
           return df
       sample = df[sex_col_name].astype(str).str.lower()
       if sample.isin(["male", "female", "m", "f", "man",
"woman"]).any():
           def norm(x):
               x = str(x).lower()
               if "male" in x or x in ("m", "man"):
                   return "male"
               if "female" in x or x in ("f", "woman"):
                   return "female"
               return x
           df["Sex_extracted"] = df[sex_col_name].apply(norm)
       else:
           def find_mf(x):
               s = str(x).lower()
               if "male" in s or "m" in s.split():
                   return "male"
               if "female" in s or "f" in s.split():
                   return "female"
               return s
           df["Sex_extracted"] = df[sex_col_name].apply(find_mf)
       return df


   def encode_sex_housing(df, colmap):
       if "sex" in colmap:
           df = extract_sex(df, colmap["sex"])
           sex_col = "Sex_extracted" if "Sex_extracted" in df.columns
else colmap["sex"]
           df["Sex_male"] =
df[sex_col].astype(str).str.lower().map(lambda x: 1 if "male" in x else
0)
       else:
           print("Колонка пола не найдена — пропускаем кодирование
Sex.")

       if "housing" in colmap:
           housing_col = colmap["housing"]
           df[housing_col] = df[housing_col].astype(str)
           d = pd.get_dummies(df[housing_col], prefix="Housing",
drop_first=True)
           df = pd.concat([df, d], axis=1)
       else:
           print("Колонка Housing не найдена — пропускаем кодирование
Housing.")

       return df


   def plot_top5_purpose(df, purpose_col):
```

```
        if purpose_col not in df.columns:
            print("Не найдена колонка Purpose — пропускаем анализ
распределения целей.")
            return
        counts = df[purpose_col].astype(str).value_counts().head(5)
        plt.figure(figsize=(8, 5))
        ax = sns.barplot(x=counts.values, y=counts.index)
        ax.set_xlabel("Количество")
        ax.set_ylabel("Цель кредита")
        ax.set_title("Топ-5 целей кредита (Purpose)")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "top5_purpose.png"))
        plt.close()
        print("График top5_purpose.png сохранён в", OUT_DIR)


    def boxplot_credit_by_risk(df, credit_col, risk_col):
        if credit_col not in df.columns or risk_col not in df.columns:
            print("Отсутствует колонка для boxplot (Credit amount или
Risk). Пропускаем.")
            return
        plt.figure(figsize=(8, 5))
        ax = sns.boxplot(x=df[risk_col].astype(str), y=df[credit_col])
        ax.set_xlabel("Кредитоспособность (Risk)")
        ax.set_ylabel("Сумма кредита")
        ax.set_title("Boxplot: Credit amount по Risk")
        plt.tight_layout()
        plt.savefig(os.path.join(OUT_DIR, "boxplot_credit_by_risk.png"))
        plt.close()
        print("График boxplot_credit_by_risk.png сохранён в", OUT_DIR)


    def pivot_age_duration_by_credit_history(df, credit_history_col):
        if credit_history_col not in df.columns:
            print("Колонка Credit history не найдена — пропускаем сводную
таблицу.")
            return None
        pivot = df.pivot_table(values=["age", "duration_in_month"],
index=credit_history_col, aggfunc="mean")
        print("\nСводная таблица: средний Age и Duration по Credit
history:\n")
        print(pivot)
        pivot.to_csv(os.path.join(OUT_DIR,
"pivot_age_duration_by_credit_history.csv"))
        print("\nPivot сохранён в", os.path.join(OUT_DIR,
"pivot_age_duration_by_credit_history.csv"))
        return pivot


    def normalize_numeric_columns(df, cols):
        missing = [c for c in cols if c not in df.columns]
        if missing:
            print("Невозможно нормализовать, т.к. не найдены столбцы:",
missing)
        present = [c for c in cols if c in df.columns]
        scaler = MinMaxScaler()
        df_norm = df.copy()
        df_norm[present] = scaler.fit_transform(df_norm[present])
        df_norm[present].to_csv(os.path.join(OUT_DIR,
"normalized_numeric_columns.csv"), index=False)
        print("\nНормализованные колонки сохранены в",
os.path.join(OUT_DIR, "normalized_numeric_columns.csv"))
        return df_norm
```

```
    def main():
        df = load_data(DATA_PATH)
        colmap = infer_and_prepare_columns(df)
        exploratory_info(df)

        df = handle_missing(df)
        print("Пропуски обработаны (числовые -> среднее, категориальные -
> мода).\n")

        plot_top5_purpose(df, colmap.get("purpose"))
        df = encode_sex_housing(df, colmap)
        boxplot_credit_by_risk(df, "credit_amount", "default")
        pivot_age_duration_by_credit_history(df,
colmap.get("credit_history"))

        numeric_to_norm = []
        for name in ["age", "credit_amount", "duration"]:
            if name in colmap:
                numeric_to_norm.append(colmap[name])
        for cand in ["age", "Credit amount", "CreditAmount", "credit
amount", "Duration", "duration"]:
            if cand in df.columns and cand not in numeric_to_norm:
                low = cand.lower()
                if "age" in low or ("credit" in low and "amount" in low)
or "duration" in low:
                    numeric_to_norm.append(cand)
        numeric_to_norm = list(dict.fromkeys(numeric_to_norm))
        print("Колонки, которые будут нормализованы:", numeric_to_norm)
        df_norm = normalize_numeric_columns(df, numeric_to_norm)

        out_csv = os.path.join(OUT_DIR, "german_credit_processed.csv")
        df_norm.to_csv(out_csv, index=False)
        print("\nПолный обработанный датасет сохранён:", out_csv)

        print("\nГотово. Проверь файлы в папке", OUT_DIR)


    if __name__ == "__main__":
        try:
            main()
        except Exception as e:
            print("Ошибка при выполнении:", e)
            sys.exit(1)
```


Топ-5 целей кредита (Purpose)
domestic appliances
car (new)
radio/television
car (used)
business
Цель кредита
0
50
100
150
200
250
Количество


Boxplot: Credit amount по Risk
17500
15000
12500
10000
7500
5000
2500
0
Сумма кредита
0
1
Кредитоспособность (Risk)

	age	credit_amount	duration_in_month
1	0.8571428571428572	0.0505667436997909	0.029411764705882346
2	0.0535714285714286	0.3136898866512601	0.6470588235294117
3	0.5357142857142858	0.10157367668097282	0.1176470588235294
4	0.46428571428571425	0.4199405744470122	0.5588235294117647
5	0.6071428571428572	0.25420931000330144	0.2941176470588235
6	0.28571428571428575	0.4844833278309673	0.47058823529411764
7	0.6071428571428572	0.14223616154946628	0.2941176470588235
8	0.28571428571428575	0.3685484758446132	0.47058823529411764
9	0.75	0.15456146142841423	0.1176470588235294
10	0.16071428571428575	0.27423792230659183	0.38235294117647056
11	0.10714285714285715	0.05749972488169913	0.1176470588235294
12	0.0892857142857143	0.22328601298558381	0.6470588235294117
13	0.0535714285714286	0.07246616044899307	0.1176470588235294
14	0.7321428571428572	0.052217453505007144	0.2941176470588235
15	0.16071428571428575	0.06344228018047761	0.16176470588235292
16	0.23214285714285715	0.056784417299438755	0.2941176470588235
17	0.6071428571428572	0.11962143721800374	0.2941176470588235
18	0.10714285714285715	0.43039506988004844	0.38235294117647056
19	0.44642857142857145	0.6783867062837019	0.2941176470588235

	credit_history	age	duration_in_month
1	all credits at this bank paid back duly	36.265306122448976	22.693877551020407
2	critical account/ other credits existing (not at this bank)	38.436860068259385	19.488054607508534
3	delay in paying off in the past	36.13636363636363	26.21590909090909
4	existing credits paid back duly till now	33.87735849056604	20.11132075471698
5	no credits taken/ all credits paid back duly	34.3	27.875

	default	account_check_status	duration_in_month	credit_his
1	0,< 0 DM,0.029411764705882346,critical account/ othe…	<unset>	<unset>	<unset>
2	1	0 <= ... < 200 DM	0.6470588235294117	existing cr
3	0	no checking account	0.1176470588235294	critical ac
4	0	< 0 DM	0.5588235294117647	existing cr
5	1	< 0 DM	0.2941176470588235	delay in pa
6	0,no checking account,0.47058823529411764,existing c…	<unset>	<unset>	<unset>
7	0	no checking account	0.2941176470588235	existing cr
8	0,0 <= ... < 200 DM,0.47058823529411764,existing cre…	<unset>	<unset>	<unset>
9	0	no checking account	0.1176470588235294	existing cr
10	1,0 <= ... < 200 DM,0.38235294117647056,critical acc…	<unset>	<unset>	<unset>
11	1,0 <= ... < 200 DM,0.1176470588235294,existing cred…	<unset>	<unset>	<unset>
12	1	< 0 DM	0.6470588235294117	existing cr
13	0,0 <= ... < 200 DM,0.1176470588235294,existing cred…	<unset>	<unset>	<unset>
14	1,< 0 DM,0.2941176470588235,critical account/ other …	<unset>	<unset>	<unset>
15	0,< 0 DM,0.16176470588235292,existing credits paid b…	<unset>	<unset>	<unset>
16	1,< 0 DM,0.2941176470588235,existing credits paid ba…	<unset>	<unset>	<unset>
17	0	no checking account	0.2941176470588235	critical ac
18	0,< 0 DM,0.38235294117647056,no credits taken/ all c…	<unset>	<unset>	<unset>
19	1,0 <= ... < 200 DM,0.2941176470588235,existing cred…	<unset>	<unset>	<unset>

Вывод: научился разрабатывать простые программы на Python с использованием библиотек **Pandas** для манипуляции и **Matplotlib** для визуализации, получил практический опыт работы с данными.