Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №2

Выполнила:
Беспалова В. А.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата:

Подпись:

Дата:

Подпись:

Москва, 2025 г.

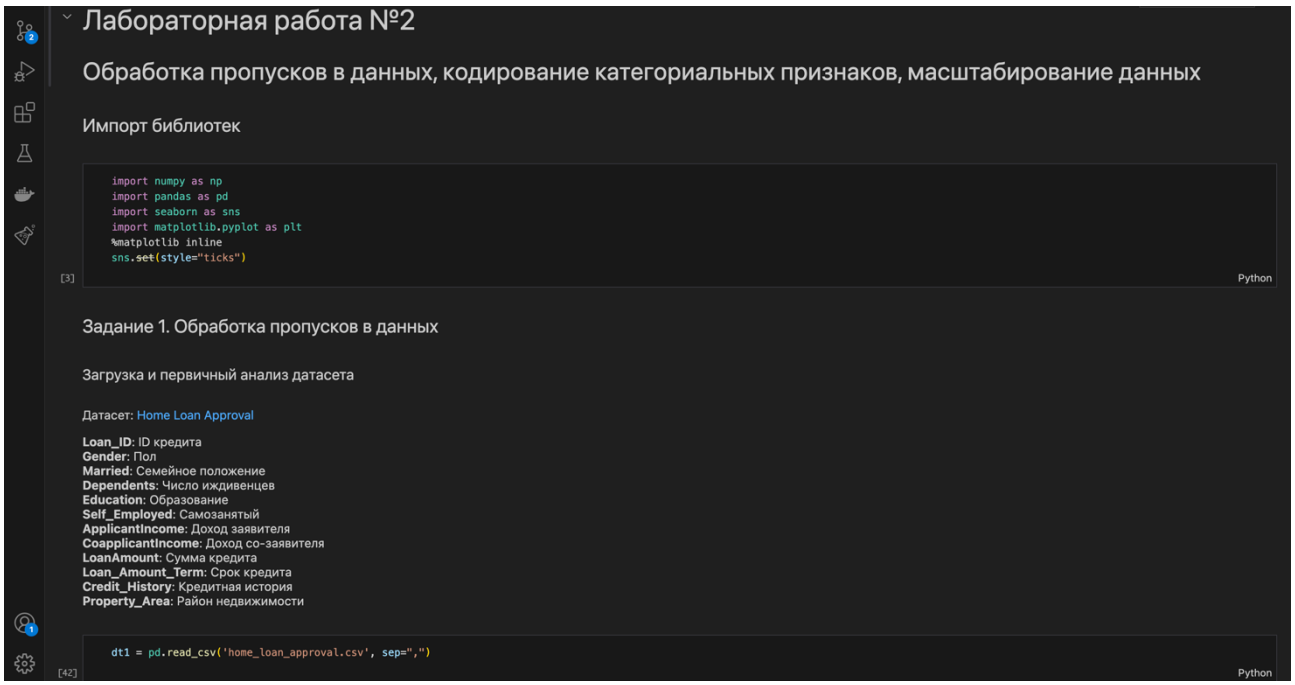
Цель лабораторной работы

Цель: изучение способов предварительной обработки данных для дальнейшего формирования моделей.

Задание

1. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   - обработку пропусков в данных;
   - кодирование категориальных признаков;
   - масштабирование данных.

Код программы и экранные формы



```python
# Размер набора данных
print('Всего строк: {}'.format(dt1.shape[0]))
print('Всего столбцов: {}'.format(dt1.shape[1]))
```

[4] Python

```
Всего строк: 367
Всего столбцов: 12
```

```python
# Типы колонок
dt1.dtypes
```

[23] Python

```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome      int64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
dtype: object
```

```python
# Проверим есть ли пропущенные значения
dt1.isnull().sum()
```

[24] Python

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

```python
# Первые 5 строк датасета
dt1.head()
```

[31] Python

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

Обработка пропусков

```python
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
total_count = dt1.shape[0]
num_cols = []
for col in dt1.columns:
    # Количество пустых значений
    temp_null_count = dt1[dt1[col].isnull()].shape[0]
    dt = str(dt1[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

[8] Python

```
Колонка LoanAmount. Тип данных float64. Количество пустых значений 5, 1.36%.
Колонка Loan_Amount_Term. Тип данных float64. Количество пустых значений 6, 1.63%.
Колонка Credit_History. Тип данных float64. Количество пустых значений 29, 7.9%.
```

```python
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
for col in dt1.columns:
    # Количество пустых значений
    temp_null_count = dt1[dt1[col].isnull()].shape[0]
    dt = str(dt1[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

[25] Python

```
Колонка Gender. Тип данных object. Количество пустых значений 11, 3.0%.
Колонка Dependents. Тип данных object. Количество пустых значений 10, 2.72%.
Колонка Self_Employed. Тип данных object. Количество пустых значений 23, 6.27%.
```

Удаление или заполнение нулями

```python
# Удаляем строки с пропусками в поле Credit_History
dt1_1 = dt1.dropna(subset=['Credit_History'])
```

[43] Python

"Внедрение значений" - импьютация

```python
from sklearn.impute import SimpleImputer

# Инициализация Imputer для числовых данных
mean_imputer = SimpleImputer(strategy='mean')
median_imputer = SimpleImputer(strategy='median')

# Заполнение пропусков для числовых признаков
dt1_1['LoanAmount'] = mean_imputer.fit_transform(dt1_1[['LoanAmount']])
dt1_1['Loan_Amount_Term'] = median_imputer.fit_transform(dt1_1[['Loan_Amount_Term']])
```

[38] Python

```python
cat_temp_data = dt1[['Gender', 'Dependents', 'Self_Employed']]
cat_temp_data.head()

# Получаем уникальные значения для каждого столбца
unique_values = {
    'Gender': cat_temp_data['Gender'].unique(),
    'Dependents': cat_temp_data['Dependents'].unique(),
    'Self_Employed': cat_temp_data['Self_Employed'].unique()
}

# Создаем DataFrame из уникальных значений
unique_values_df = pd.DataFrame(dict([(k, pd.Series(v)) for k, v in unique_values.items()]))

# Выводим таблицу с уникальными значениями
print("Таблица уникальных значений категориальных признаков:")
print(unique_values_df)
```

[51] Python

```
Таблица уникальных значений категориальных признаков:
   Gender Dependents Self_Employed
0    Male          0            No
1  Female          1           Yes
2     NaN          2           NaN
3     NaN         3+           NaN
4     NaN        NaN           NaN
```

```python
# Инициализация Imputer для категориальных данных
imputer_mf = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imputer_const = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value="NO_INFO")

# # Заполнение пропусков для категориальных признаков
dt1_1[['Gender']] = imputer_mf.fit_transform(dt1_1[['Gender']])
dt1_1[['Dependents']] = imputer_const.fit_transform(dt1_1[['Dependents']])
dt1_1[['Self_Employed']] = imputer_const.fit_transform(dt1_1[['Self_Employed']])
```

```python
dt1_1.head()
```

[46] Python

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	360.0	1.0	Urban

```python
# Проверим есть ли пропущенные значения
dt1_1.isnull().sum()
```

[52] Python

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           5
Loan_Amount_Term     5
Credit_History       0
Property_Area        0
dtype: int64
```

Задание 2. Кодирование категориальных признаков

Описание датасета

Датасет: Laptop Price

Company: Производитель
Product: Продукт
TypeName: Категория
Inches: Дюймы
ScreenResolution: Разрешение экрана
CPU_Company: Производитель процессора
CPU_Type: Тип процессора
CPU_Frequency (GHz): Частота процессора (ГГц)
RAM (GB): Оперативная память (ГБ)
Memory: Память
GPU_Company: Компания видеокарты
GPU_Type: Тип видеокарты
OpSys: Операционная система
Weight (kg): Вес (кг)
Price (Euro): Цена (евро)

This dataset contains a variety of laptop specifications and the price of each device in Euros.

```python
dt2 = pd.read_csv('laptop_price.csv', sep=",")
```

[16] Python

```python
# размер набора данных
dt2.shape
```

[36] Python

```
(1275, 15)
```

```python
# типы колонок
dt2.dtypes
```

[5] Python

```
Company                object
Product                object
TypeName               object
Inches                float64
ScreenResolution       object
CPU_Company            object
CPU_Type               object
CPU_Frequency (GHz)    float64
RAM (GB)                 int64
Memory                 object
GPU_Company            object
GPU_Type               object
OpSys                  object
Weight (kg)           float64
Price (Euro)          float64
dtype: object
```

```python
# проверим есть ли пропущенные значения
dt2.isnull().sum()
```

[6] Python

```
Company               0
Product               0
TypeName              0
Inches                0
ScreenResolution      0
CPU_Company           0
CPU_Type              0
CPU_Frequency (GHz)   0
RAM (GB)              0
Memory                0
GPU_Company           0
GPU_Type              0
OpSys                 0
Weight (kg)           0
Price (Euro)          0
dtype: int64
```

```python
from sklearn.preprocessing import LabelEncoder

label_encoders = {}
encoder_tables = {}

dt2_2 = dt2

for col in categorical_columns:
    le = LabelEncoder()
    dt2_2[col] = le.fit_transform(dt2_2[col].astype(str))  # Преобразуем в строку для обработки
    label_encoders[col] = le  # Сохраняем энкодер, если понадобится для обратного преобразования
    # Создаем таблицу с оригинальными и закодированными значениями
    encoder_tables[col] = pd.DataFrame({
        'Original': le.classes_,
        'Encoded': le.transform(le.classes_)
    })

# Проверка изменений
dt2_2.head()
```

[17] Python

	Company	Product	TypeName	Inches	ScreenResolution	CPU_Company	CPU_Type	CPU_Frequency (GHz)	RAM (GB)	Memory	GPU_Company	GPU_Type	OpSys	Weight (kg)	Price (Euro)
0	1	300	4	13.3	23	1	40	2.3	8	4	2	56	8	1.37	1339.69
1	1	301	4	13.3	1	1	40	1.8	8	2	2	50	8	1.34	898.94
2	7	50	3	15.6	8	1	46	2.5	8	16	2	52	4	1.86	575.00
3	1	300	4	15.4	25	1	54	2.7	16	29	0	76	8	1.83	2537.45
4	1	300	4	13.3	23	1	40	3.1	8	16	2	57	8	1.37	1803.60

```python
# Вывод таблиц энкодеров
for col, encoder_df in encoder_tables.items():
    print(f"\nТаблица энкодера для '{col}':")
    print(encoder_df)
```

[18] Python

```
Таблица энкодера для 'Company':
     Original  Encoded
0        Acer        0
1       Apple        1
2        Asus        2
3       Chuwi        3
4        Dell        4
5     Fujitsu        5
6      Google        6
7          HP        7
8      Huawei        8
9          LG        9
10     Lenovo       10
11        MSI       11
12   Mediacom       12
13  Microsoft       13
14      Razer       14
15    Samsung       15
16    Toshiba       16
17       Vero       17
18     Xiaomi       18

Таблица энкодера для 'Product':
                          Original  Encoded
...
5    Windows 10        5
6  Windows 10 S        6
7     Windows 7        7
8         macOS        8
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Задание 3. Масштабирование данных

Масштабирование предполагает изменение диапазона измерения величины.

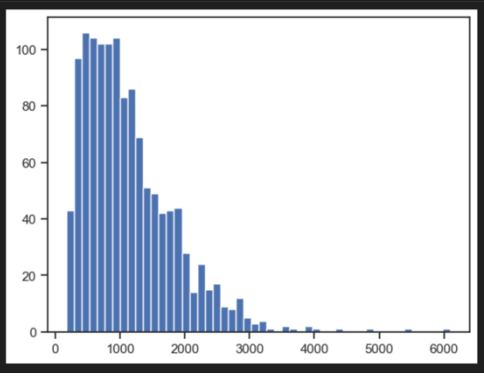
MinMax масштабирование

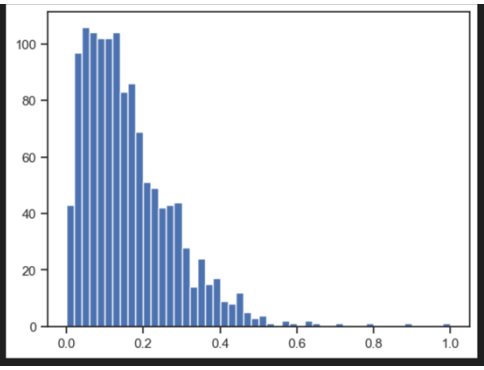
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(dt2[['Price (Euro)']])
plt.hist(dt2['Price (Euro)'], 50)
plt.show()

plt.hist(sc1_data, 50)
plt.show()
```

[50] Python

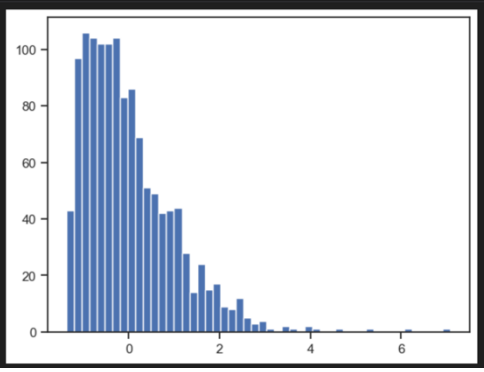




Масштабирование данных на основе Z-оценки - StandardScaler

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(dt2[['Price (Euro)']])
plt.hist(sc2_data, 50)
plt.show()
```

[52] Python



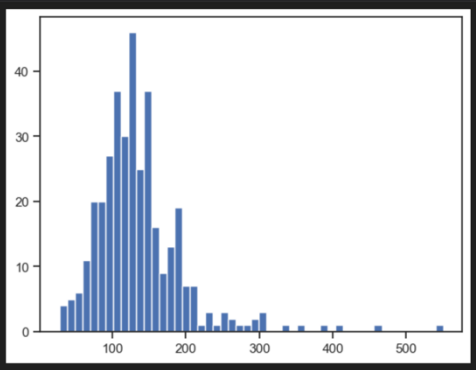
MinMax масштабирование

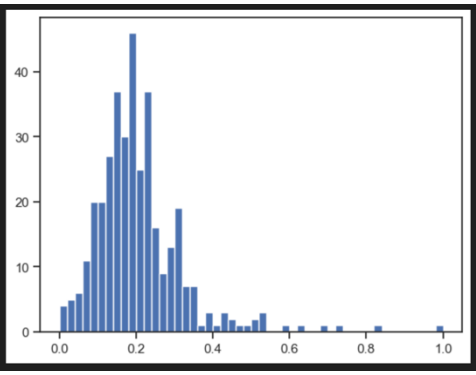
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(dt1[['LoanAmount']])
plt.hist(dt1['LoanAmount'], 50)
plt.show()

plt.hist(sc1_data, 50)
plt.show()
```

[51] Python





Масштабирование данных на основе Z-оценки - StandardScaler

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(dt1[['LoanAmount']])
plt.hist(sc2_data, 50)
plt.show()
```

[53] Python