Московский государственный технический

университет им. Н.Э. Баумана

Факультет Радиотехнический

Кафедра РТ5

Курс «Сети и телекоммуникации»

Отчет по лабораторной работе №2

«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.»

Выполнил:

студент группы РТ5-61Б:

Бабасанова Н. С.

Руководитель:

преподаватель каф. ИУ5

Нардид А. Н.

Москва, 2025г.

Задание:

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.)

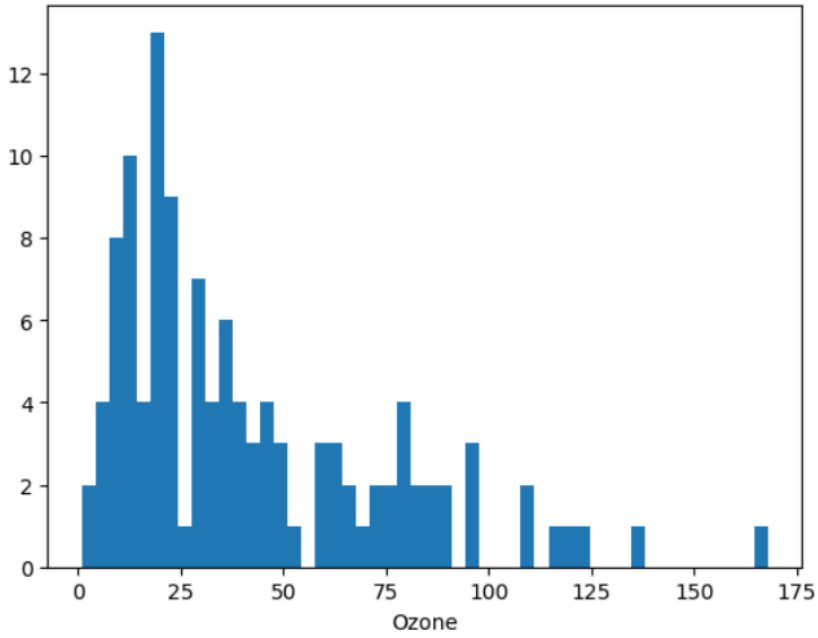
Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:

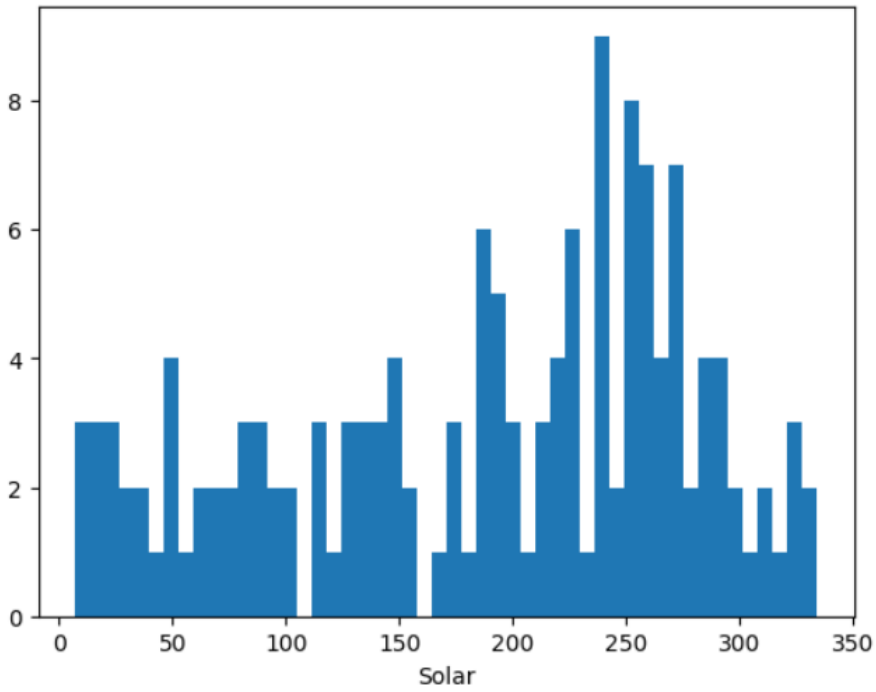
- обработку пропусков в данных;
- кодирование категориальных признаков;
- масштабирование данных.

Текст программы и формы с примерами выполнения программы:

```
import numpy as np
import pandas as pd

data = pd.read_csv('/usr/data/Air Quality Missing Data.csv', sep=',')

data.shape

data.head()
data.isnull().sum()
data_new = data.fillna(0)
data_new.head()

num_cols = []
total_count = data.shape[0]
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[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))
```

```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num

# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

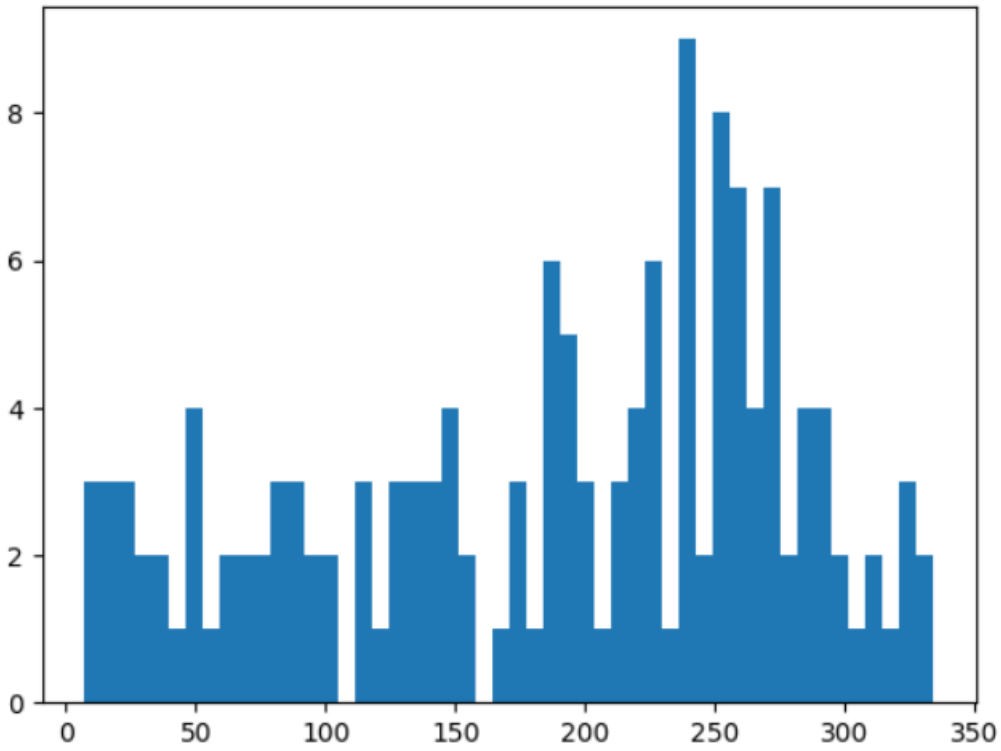




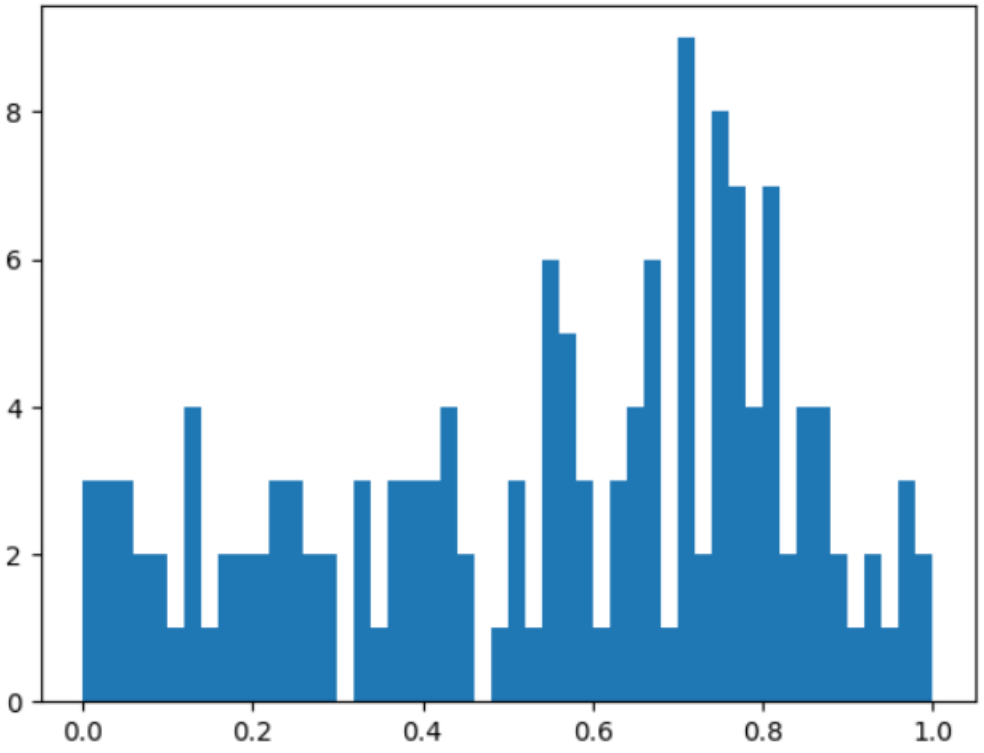

```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator

data_num_solar = data_num[['Solar']]
```

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_solar)
mask_missing_values_only



def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0],
filled_data[filled_data.size-1]

strategies=['mean', 'median', 'most_frequent']

test_num_impute_col(data, 'Solar', strategies[0])



data_cat = pd.read_csv('/usr/data/Penguins.csv', sep=',')

Целевая переменная - пол.

data_cat.head()

data_cat.isnull().sum()

# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета

total_counts = data.shape[0]
cat_cols = []
for col in data_cat.columns:
    # Количество пустых значений
    temp_null_count = data_cat[data_cat[col].isnull()].shape[0]
```

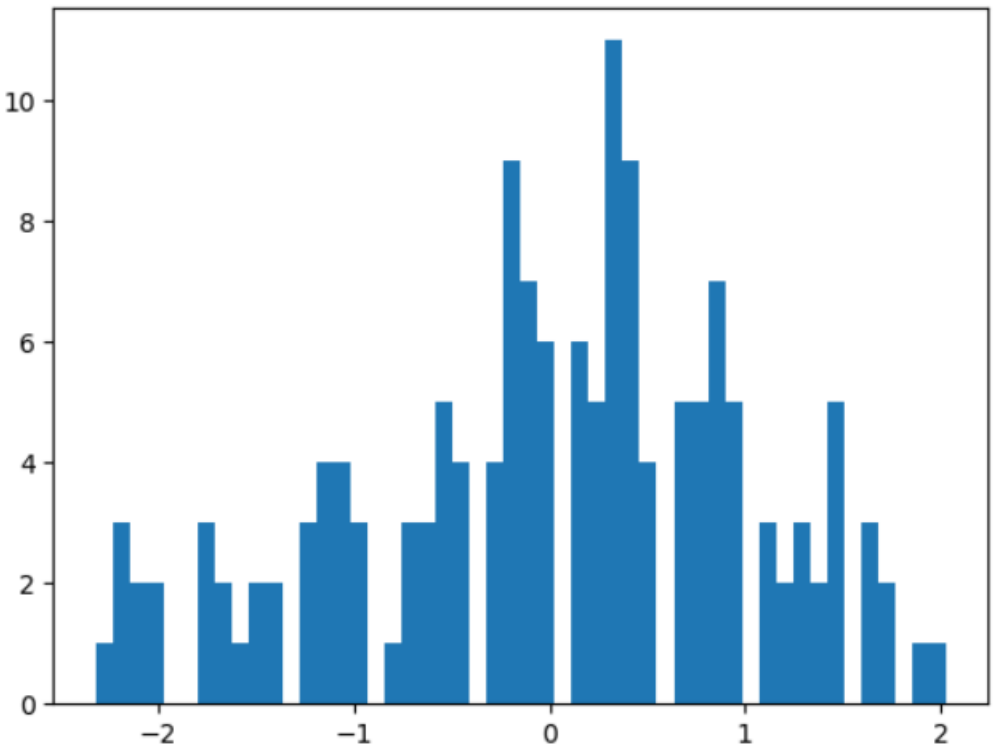
```
            dt = str(data_cat[col].dtype)
            if temp_null_count>0 and (dt=='object'):
                cat_cols.append(col)
                temp_perc = round((temp_null_count / total_counts) * 100.0, 2)
                print('Колонка {}. Тип данных {}. Количество пустых значений {},
{}%.'.format(col, dt, temp_null_count, temp_perc))

        cat_temp_data = data_cat[['sex']]
        cat_temp_data.head()

        cat_temp_data['sex'].unique()

        cat_temp_data[cat_temp_data['sex'].isnull()].shape

        # Импьютация наиболее частыми значениями
        imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
        data_imp2 = imp2.fit_transform(cat_temp_data)
        data_imp2

        # Пустые значения отсутствуют?
        np.unique(data_imp2)
        # Да!


        from   sklearn.preprocessing   import   MinMaxScaler,   StandardScaler,
Normalizer

        sc1 = MinMaxScaler()
        sc1_data = sc1.fit_transform(data[['Solar']])

        plt.hist(data['Solar'], 50)
        plt.show()
```




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




```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Temp']])

plt.hist(sc2_data, 50)
plt.show()
```


10
8
6
4
2
0
−2
−1
0
1
2