**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: |  | Руководитель: |
| студент группы РТ5-61Б: |  | преподаватель каф. ИУ5 |
| Бабасанова Н. С. |  | 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

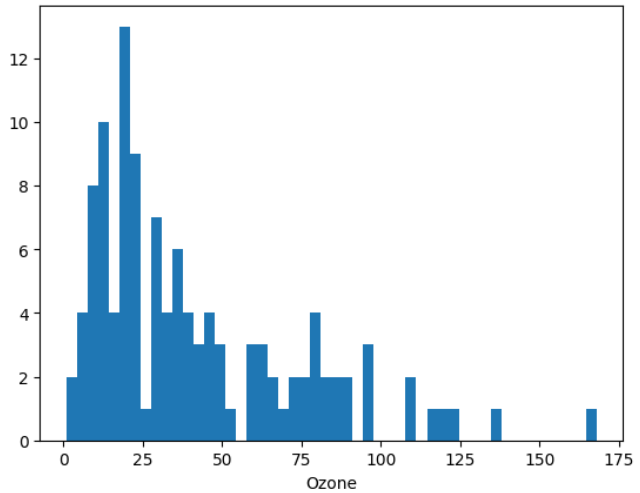
# Гистограмма по признакам

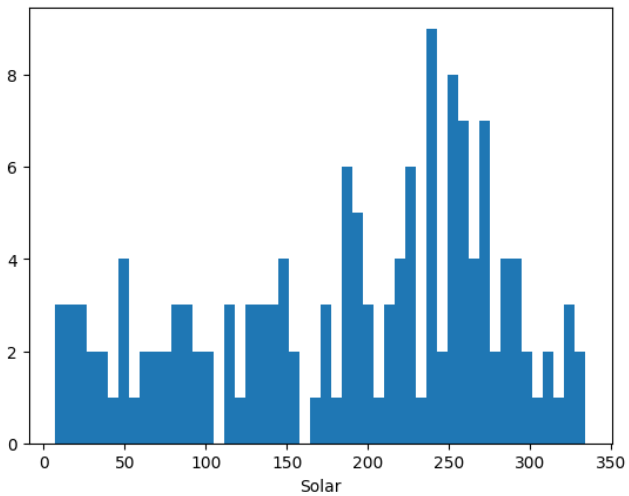
for col in data\_num:

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!

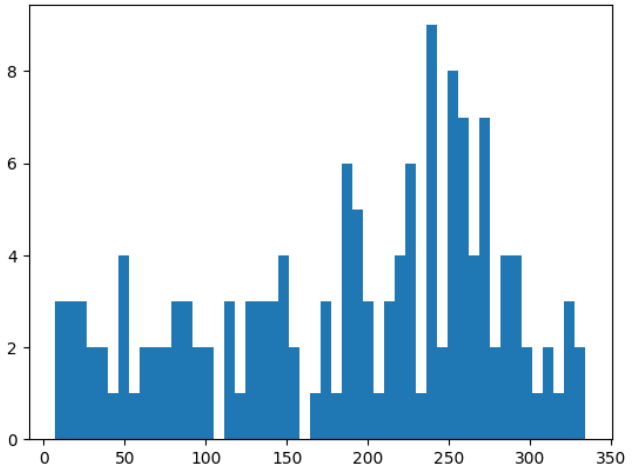
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

sc1 = MinMaxScaler()

sc1\_data = sc1.fit\_transform(data[['Solar']])

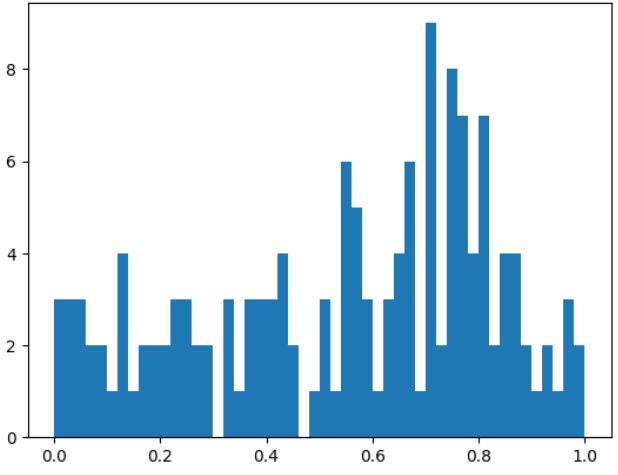
plt.hist(data['Solar'], 50)

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()

