|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

**ОТЧЁТ ПО**

**Лабораторной работе №2**

Выполнил: Проверил:

студент группы ИУ5-65Б преподаватель каф. ИУ5

Кулешова И. А. Гапанюк Ю. Е.

Подпись и дата: Подпись и дата:

Москва

2024

1. **Описание задания:**
2. 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.)
3. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   1. обработку пропусков в данных;
   2. кодирование категориальных признаков;
   3. масштабирование данных.
4. **Текст программы:**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

sns.set(style="ticks")

**Загрузка и первичный анализ данных**

data **=** pd**.**read\_csv('data/car\_prices.csv', sep**=**",")

data**.**shape

data**.**dtypes

*# проверим есть ли пропущенные значения*

data**.**isnull()**.**sum()

*# Первые 5 строк датасета*

data**.**head()

total\_count **=** data**.**shape[0]

print('Всего строк: {}'**.**format(total\_count))

**Обработка пропусков в данных**

*# Удаление колонок, содержащих пустые значения*

data\_new\_1 **=** data**.**dropna(axis**=**1, how**=**'any')

(data**.**shape, data\_new\_1**.**shape)

*# Удаление строк, содержащих пустые значения*

data\_new\_2 **=** data**.**dropna(axis**=**0, how**=**'any')

(data**.**shape, data\_new\_2**.**shape)

data**.**head()

**Обработка пропусков в числовых данных**

*# Выберем числовые колонки с пропущенными значениями*

*# Цикл по колонкам датасета*

num\_cols **=** []

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

strategies=['mean', 'median', 'most\_frequent']

*# Более сложная функция, которая позволяет задавать колонку и вид импьютации*

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]

test\_num\_impute\_col(data, 'condition', strategies[0])

test\_num\_impute\_col(data, 'condition', strategies[1])

test\_num\_impute\_col(data, 'condition', strategies[2])

data.fillna({'condition': 19}, inplace=True)

test\_num\_impute\_col(data, 'odometer', strategies[0])

test\_num\_impute\_col(data, 'odometer', strategies[1])

test\_num\_impute\_col(data, 'odometer', strategies[2])

test\_num\_impute\_col(data, 'mmr', strategies[0])

test\_num\_impute\_col(data, 'mmr', strategies[1])

test\_num\_impute\_col(data, 'mmr', strategies[2])

test\_num\_impute\_col(data, 'sellingprice', strategies[0])

test\_num\_impute\_col(data, 'sellingprice', strategies[1])

test\_num\_impute\_col(data, 'sellingprice', strategies[2])

**Обработка пропусков в категориальных данных**

*# Выберем категориальные колонки с пропущенными значениями*

*# Цикл по колонкам датасета*

cat\_cols = []

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=='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))

cat\_temp\_data **=** data[['make']]

cat\_temp\_data**.**head()

cat\_temp\_data['make']**.**unique()

cat\_temp\_data[cat\_temp\_data['make']**.**isnull()]**.**shape

*# Импьютация наиболее частыми значениями*

imp2 **=** SimpleImputer(missing\_values**=**np**.**nan, strategy**=**'most\_frequent')

data\_imp2 **=** imp2**.**fit\_transform(cat\_temp\_data)

data[['make']] **=** data\_imp2

data\_imp2

np**.**unique(data\_imp2)

cat\_temp\_data **=** data[['model']]

cat\_temp\_data**.**head()

cat\_temp\_data['model']**.**unique()

cat\_temp\_data[cat\_temp\_data['model']**.**isnull()]**.**shape

*# Импьютация наиболее частыми значениями*

imp2 **=** SimpleImputer(missing\_values**=**np**.**nan, strategy**=**'most\_frequent')

data\_imp2 **=** imp2**.**fit\_transform(cat\_temp\_data)

data[['model']] **=** data\_imp2

data\_imp2

np**.**unique(data\_imp2)

cat\_temp\_data **=** data[['trim']]

cat\_temp\_data**.**head()

cat\_temp\_data['trim']**.**unique()

cat\_temp\_data[cat\_temp\_data['trim']**.**isnull()]**.**shape

*# Импьютация наиболее частыми значениями*

imp2 **=** SimpleImputer(missing\_values**=**np**.**nan, strategy**=**'most\_frequent')

data\_imp2 **=** imp2**.**fit\_transform(cat\_temp\_data)

data[['trim']] **=** data\_imp2

data\_imp2

np**.**unique(data\_imp2)

cat\_temp\_data **=** data[['body']]

cat\_temp\_data**.**head()

cat\_temp\_data[cat\_temp\_data['body']**.**isnull()]**.**shape

*# Импьютация наиболее частыми значениями*

imp2 **=** SimpleImputer(missing\_values**=**np**.**nan, strategy**=**'most\_frequent')

data\_imp2 **=** imp2**.**fit\_transform(cat\_temp\_data)

data[['body']] **=** data\_imp2

data\_imp2

np**.**unique(data\_imp2)

cat\_temp\_data **=** data[['transmission']]

cat\_temp\_data**.**head()

cat\_temp\_data['transmission']**.**unique()

cat\_temp\_data[cat\_temp\_data['transmission']**.**isnull()]**.**shape

*# Импьютация наиболее частыми значениями*

imp2 **=** SimpleImputer(missing\_values**=**np**.**nan, strategy**=**'most\_frequent')

data\_imp2 **=** imp2**.**fit\_transform(cat\_temp\_data)

data[['ransmission']] **=** data\_imp2

data\_imp2

np**.**unique(data\_imp2)

*# Удаление строк, содержащих пустые значения*

data\_new\_2 **=** data**.**dropna(axis**=**0, how**=**'any')

(data**.**shape, data\_new\_2**.**shape)

**Преобразование категориальных признаков в числовые**

**Кодирование категорий целочисленными значениями**

from sklearn.preprocessing import OrdinalEncoder

data\_oe = data[['make', 'color']]

data\_oe.head()

imp4 = SimpleImputer(missing\_values=np.nan, strategy='constant', fill\_value='NA')

data\_oe\_filled = imp4.fit\_transform(data\_oe)

data\_oe\_filled

oe = OrdinalEncoder()

cat\_enc\_oe = oe.fit\_transform(data\_oe\_filled)

cat\_enc\_oe

*# Уникальные значения 1 признака*

np**.**unique(cat\_enc\_oe[:, 0])

*# Уникальные значения 2 признака*

np**.**unique(cat\_enc\_oe[:, 1])

*# Наименования категорий в соответствии с порядковыми номерами*

oe**.**categories\_

**Масштабирование данных**

from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

**MinMax масштабирование**

sc1 **=** MinMaxScaler()

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

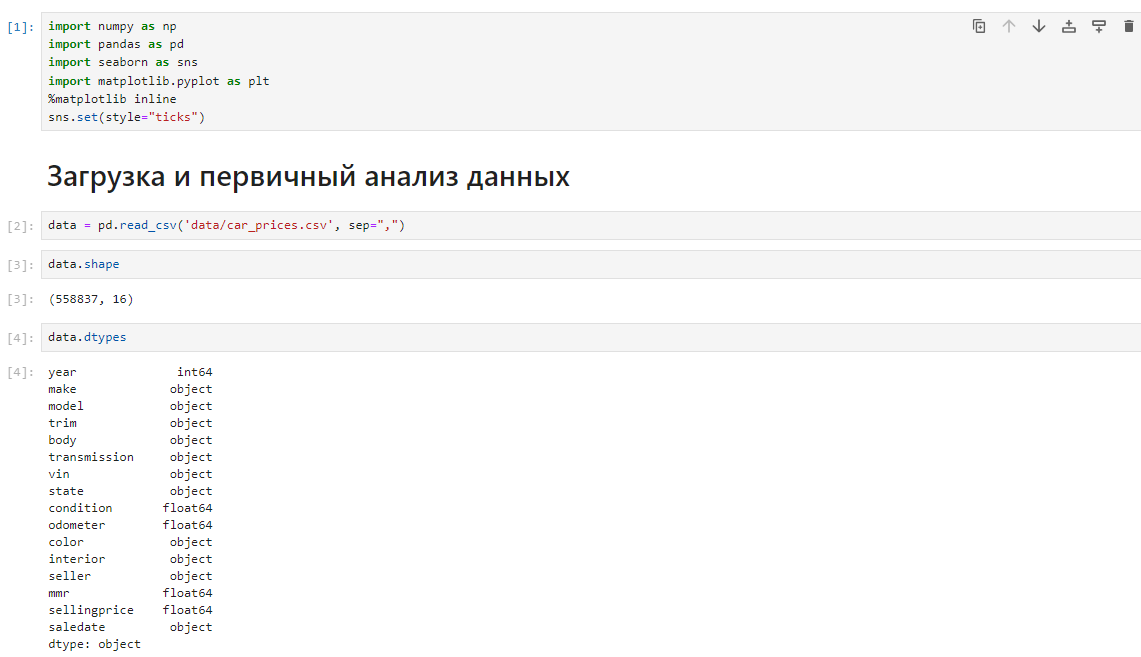
plt**.**hist(data['condition'], 50)

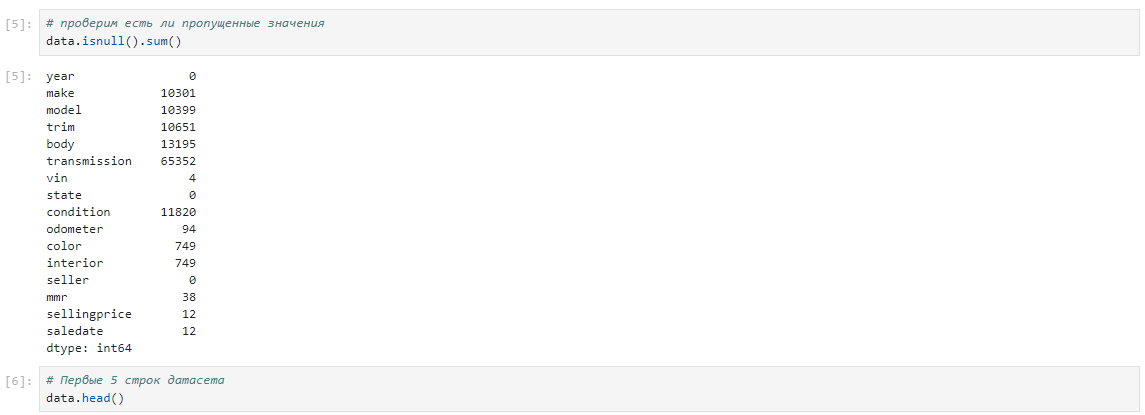
plt**.**show()

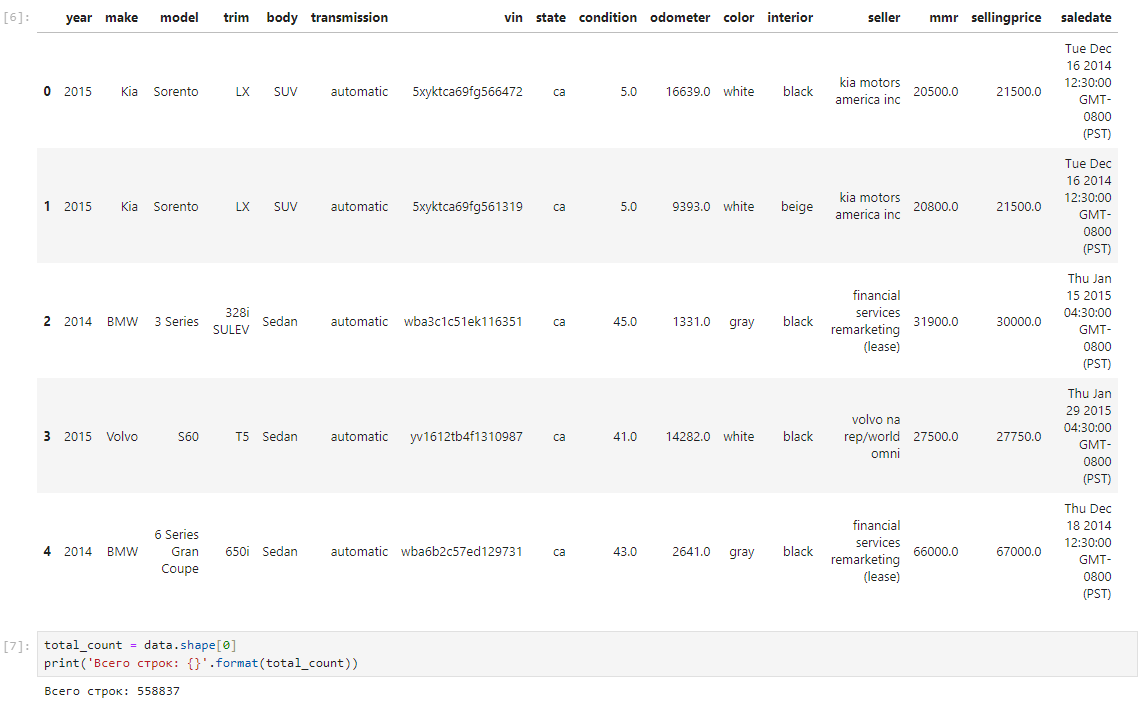
plt**.**hist(sc1\_data, 50)

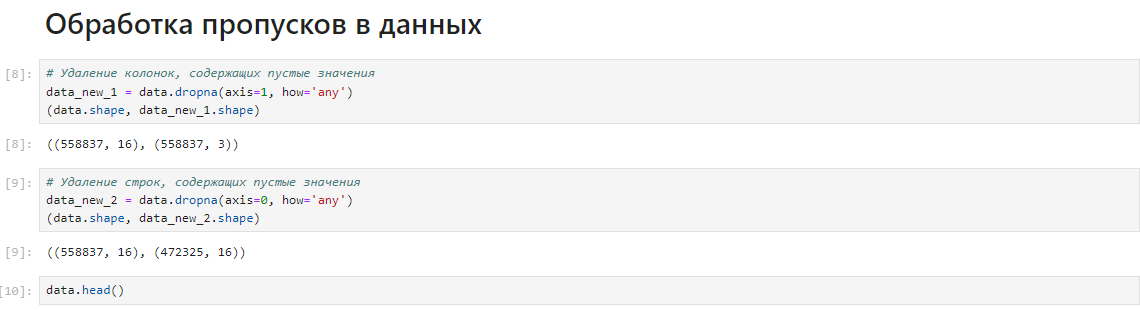
plt**.**show()

1. **Экранные формы с примерами выполнения программы:**

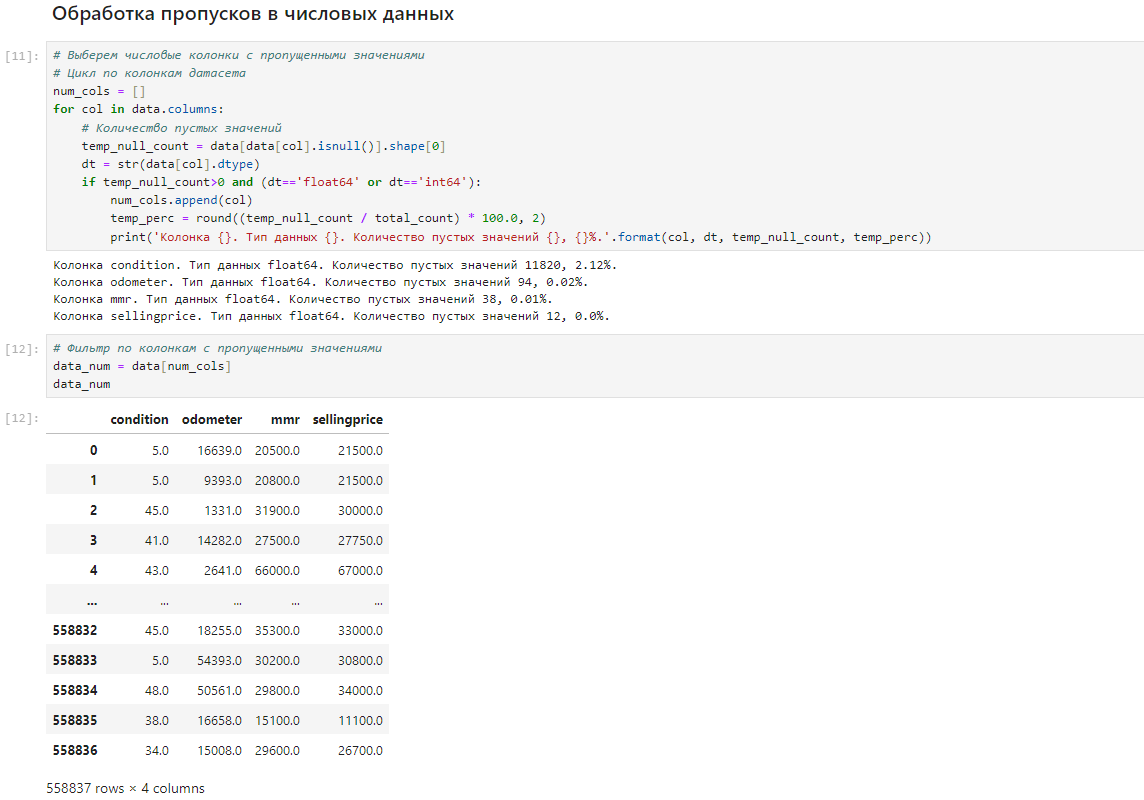
****

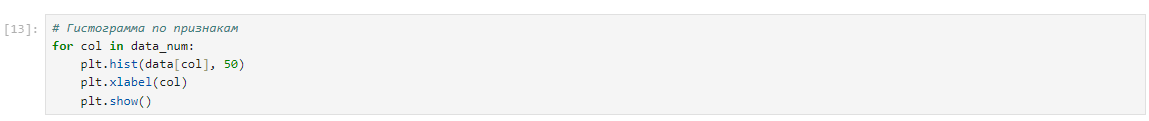
****

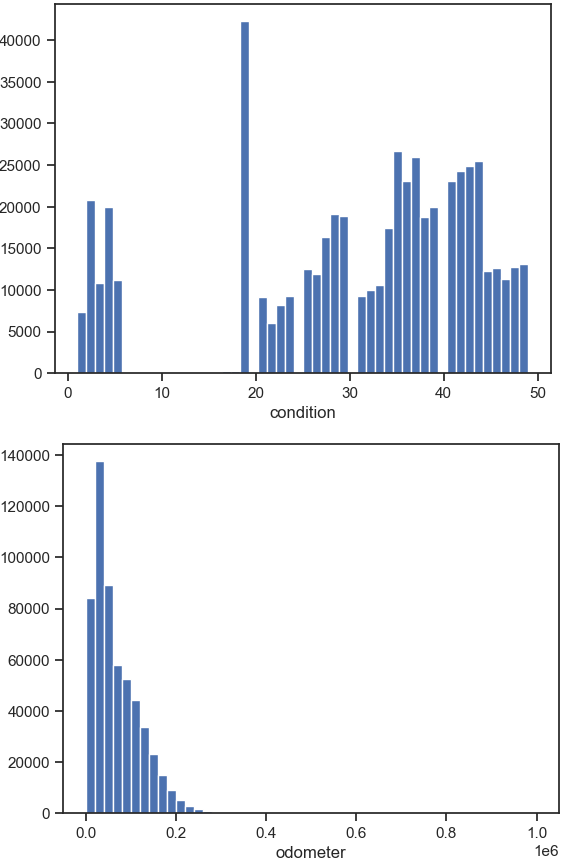
****

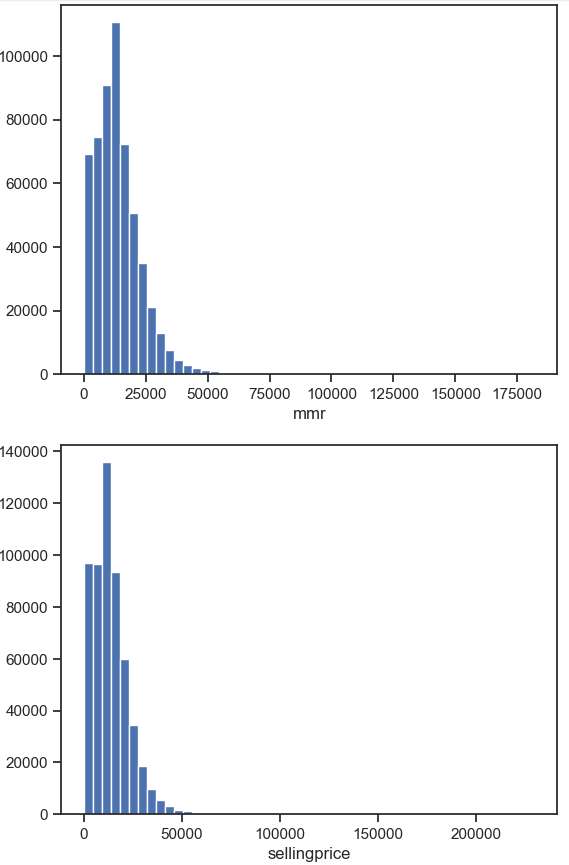
****

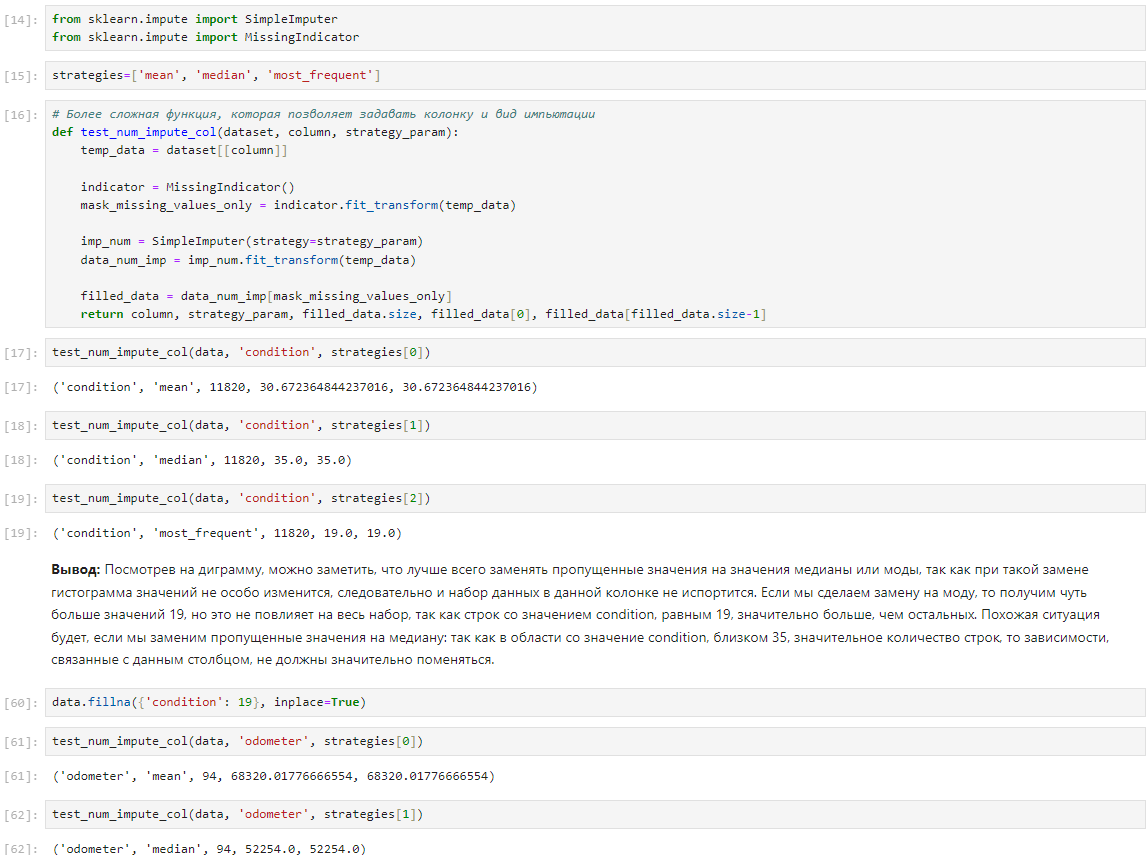
****

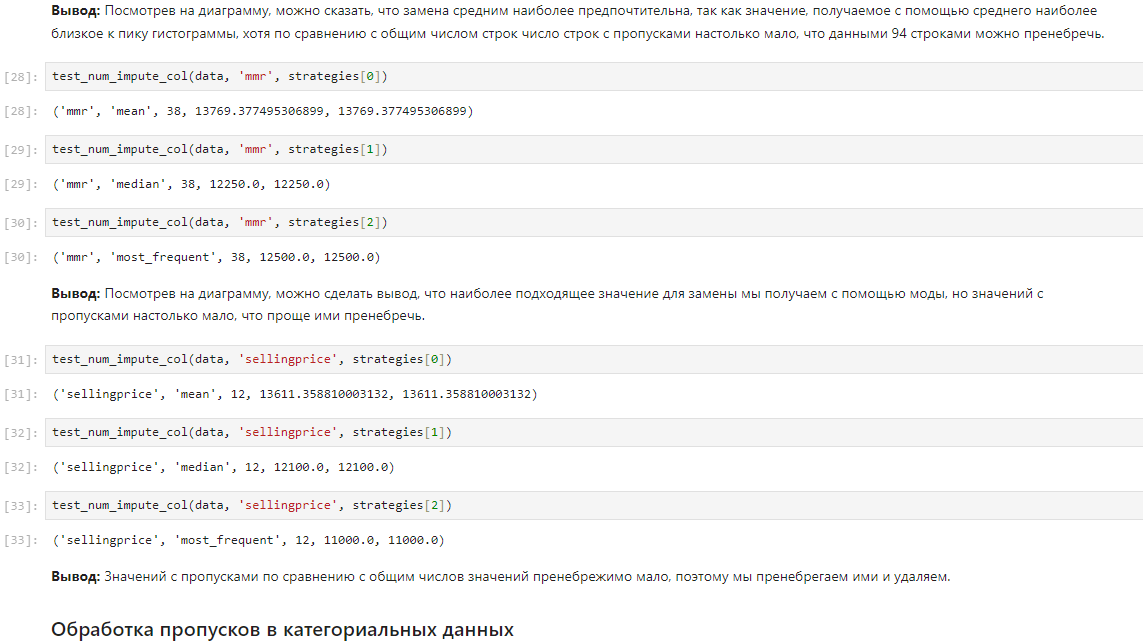
****

****

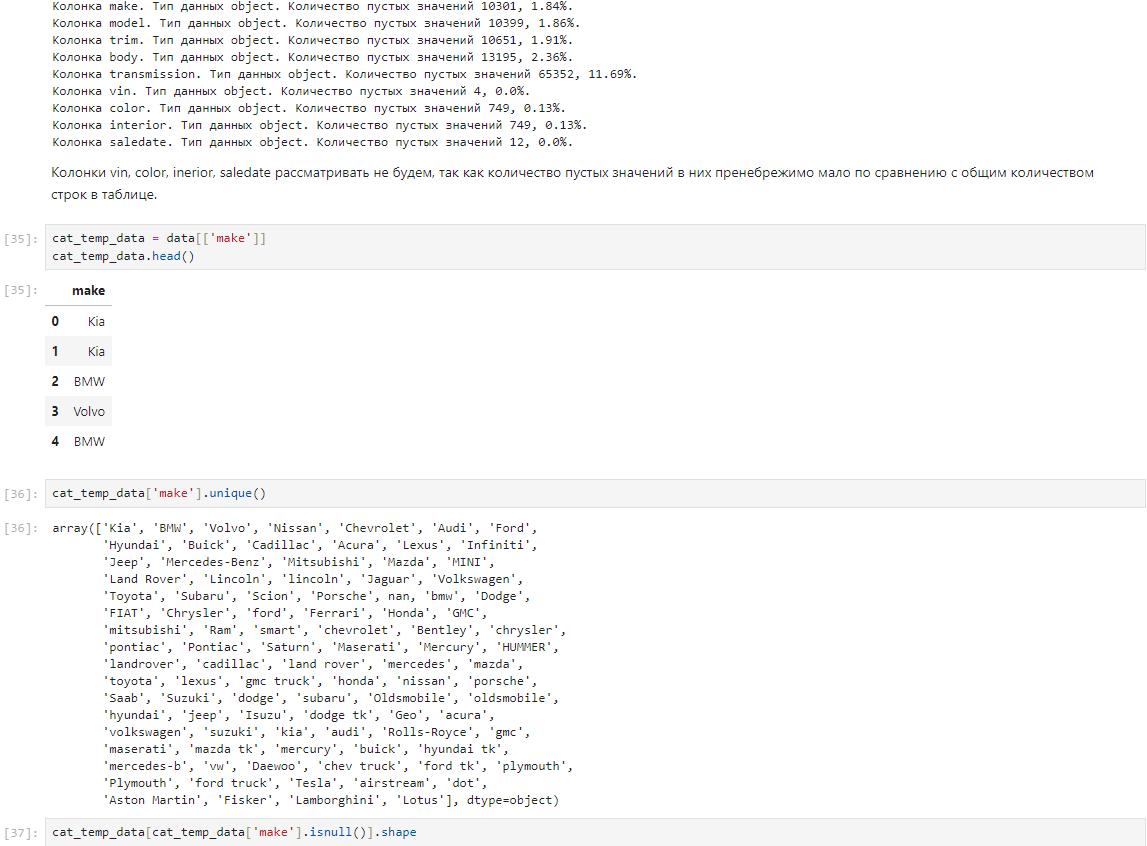
****

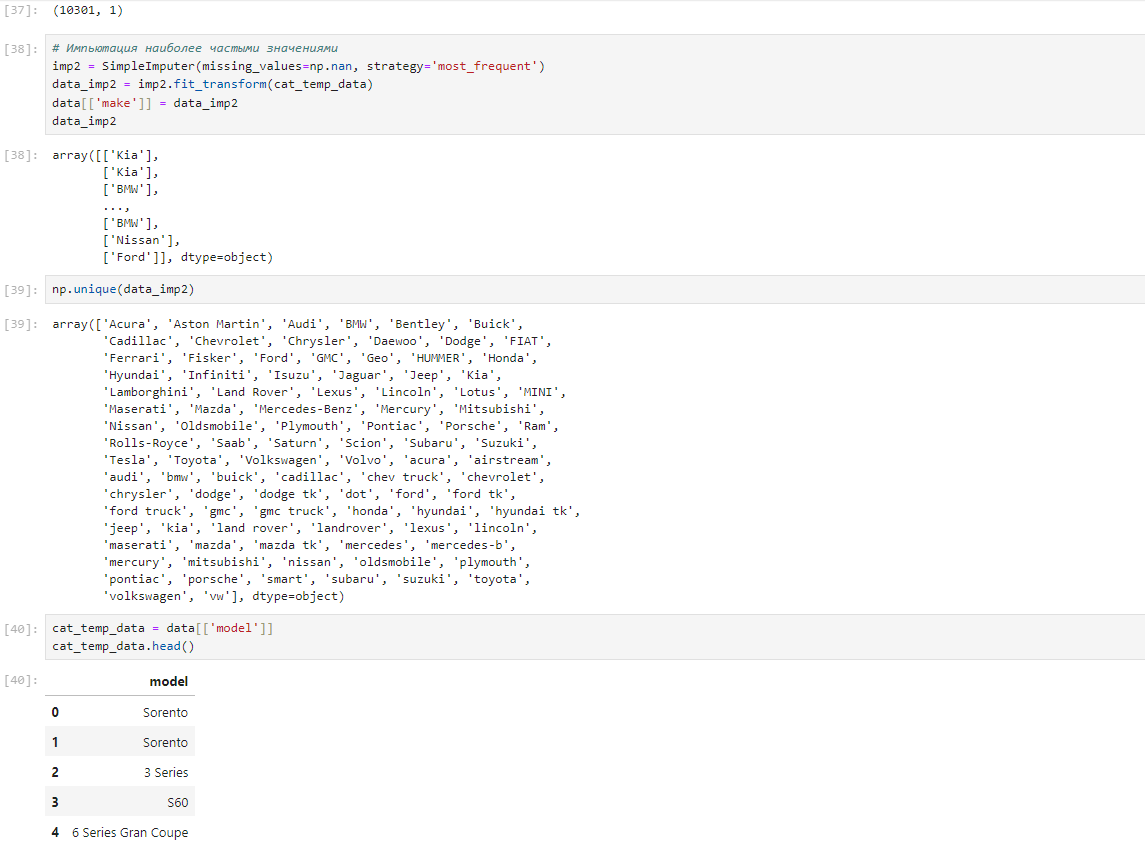
****

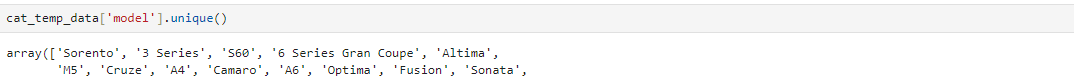
****

****

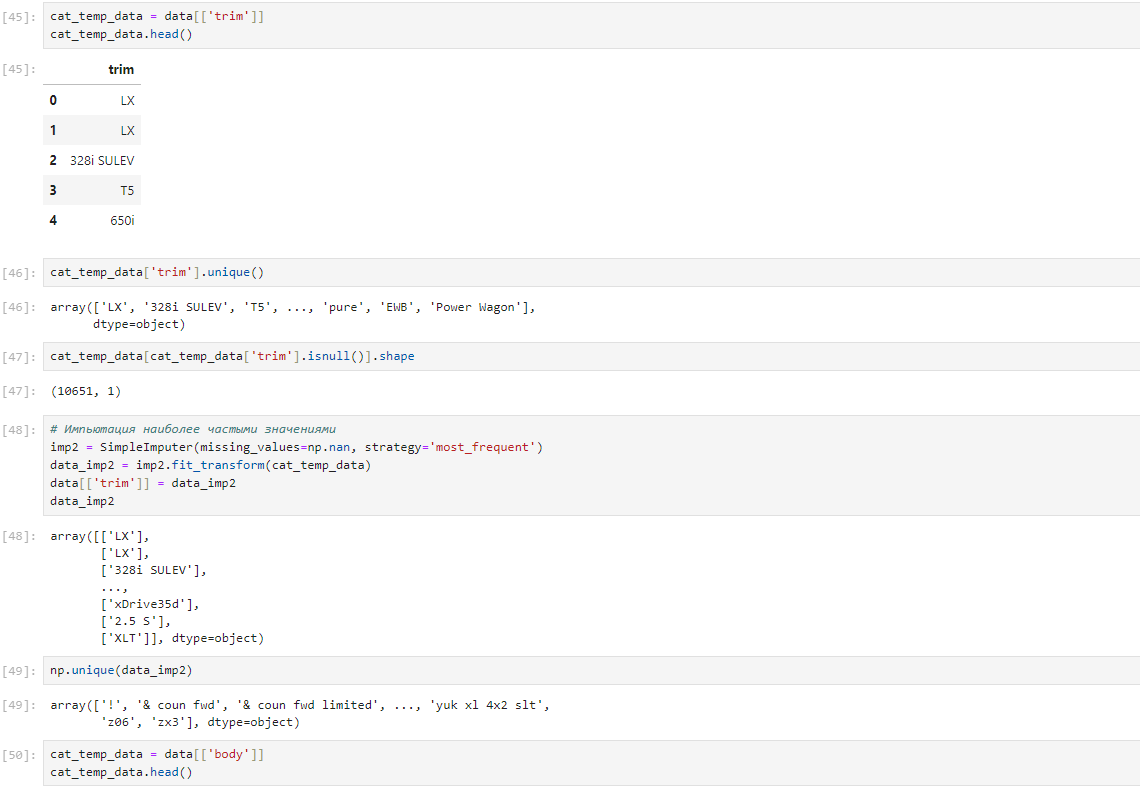
****

****

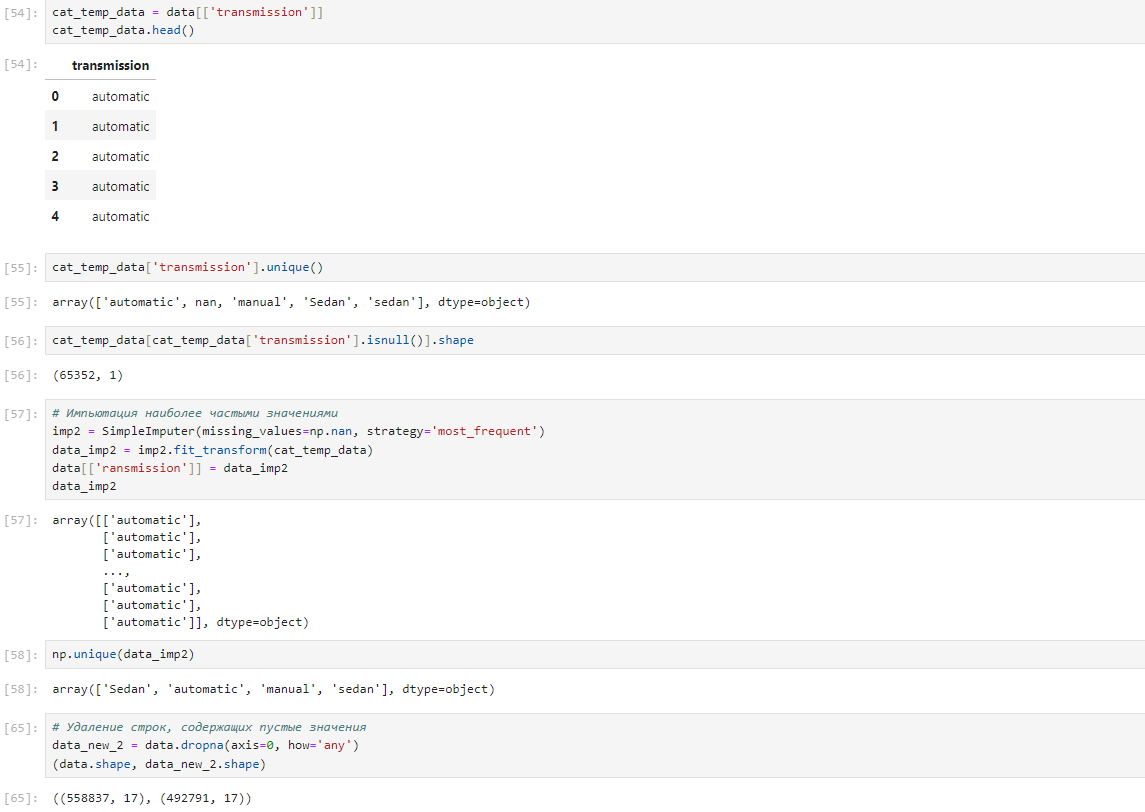
****

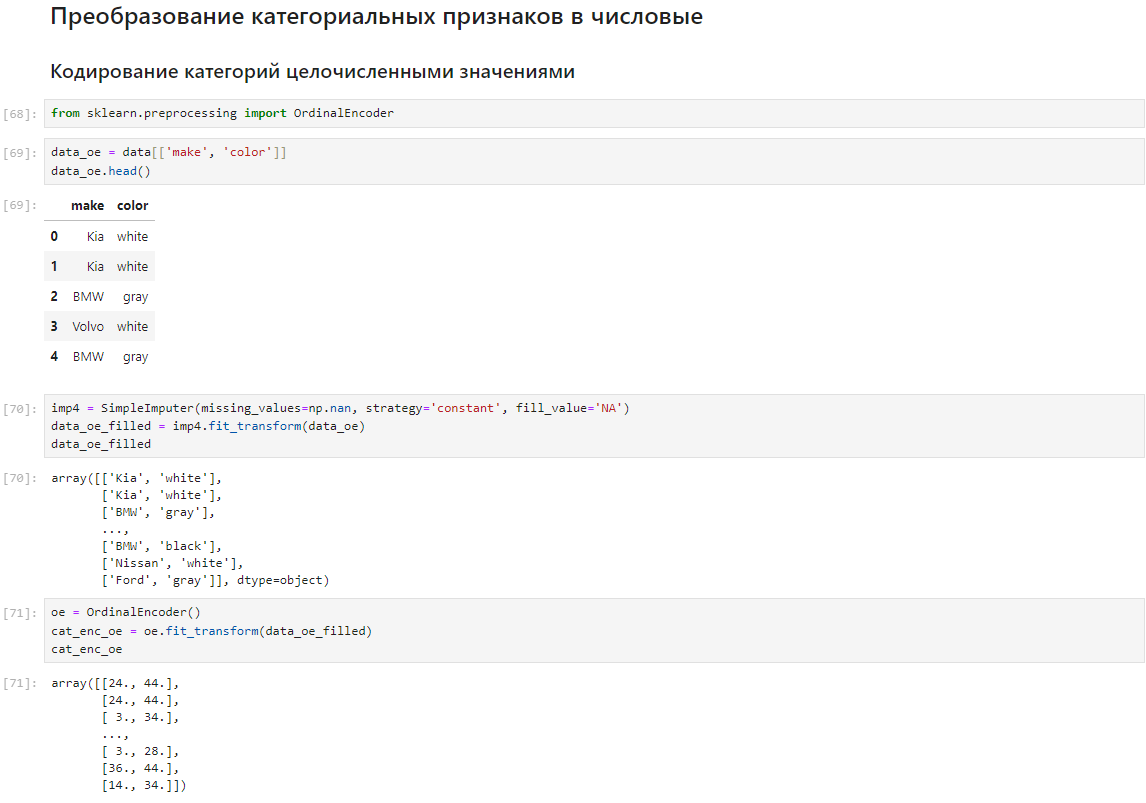
****

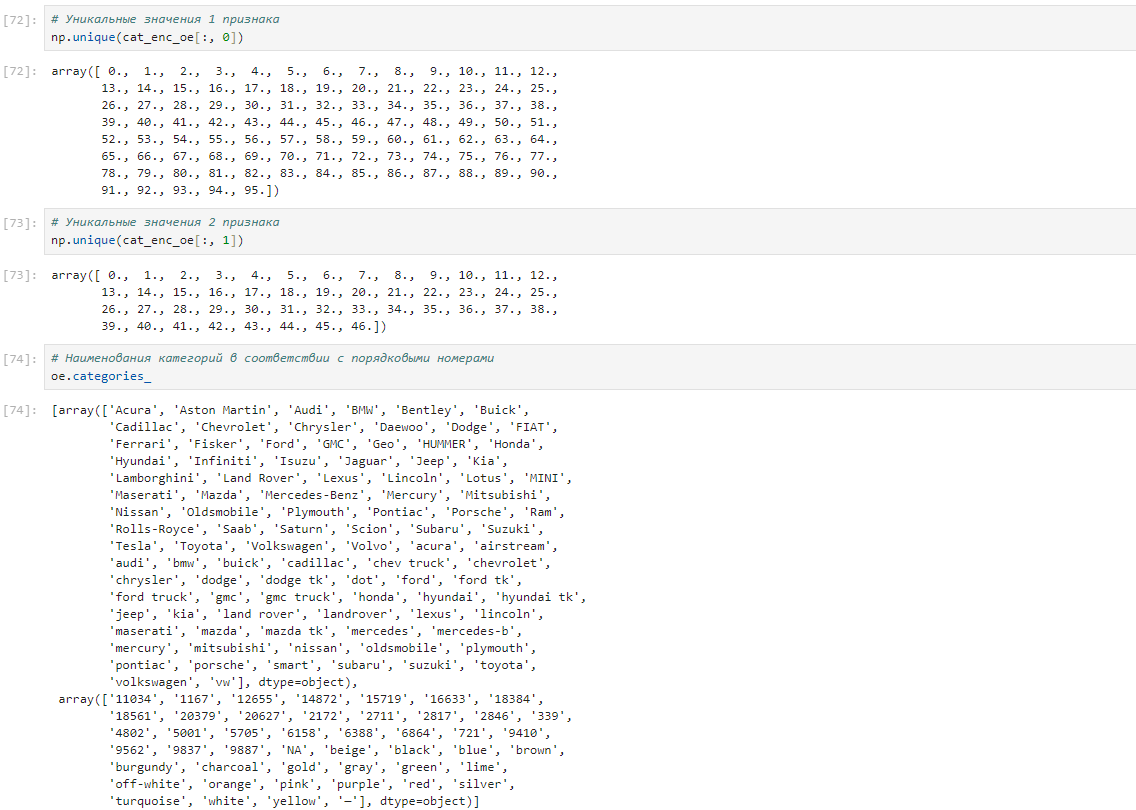
****

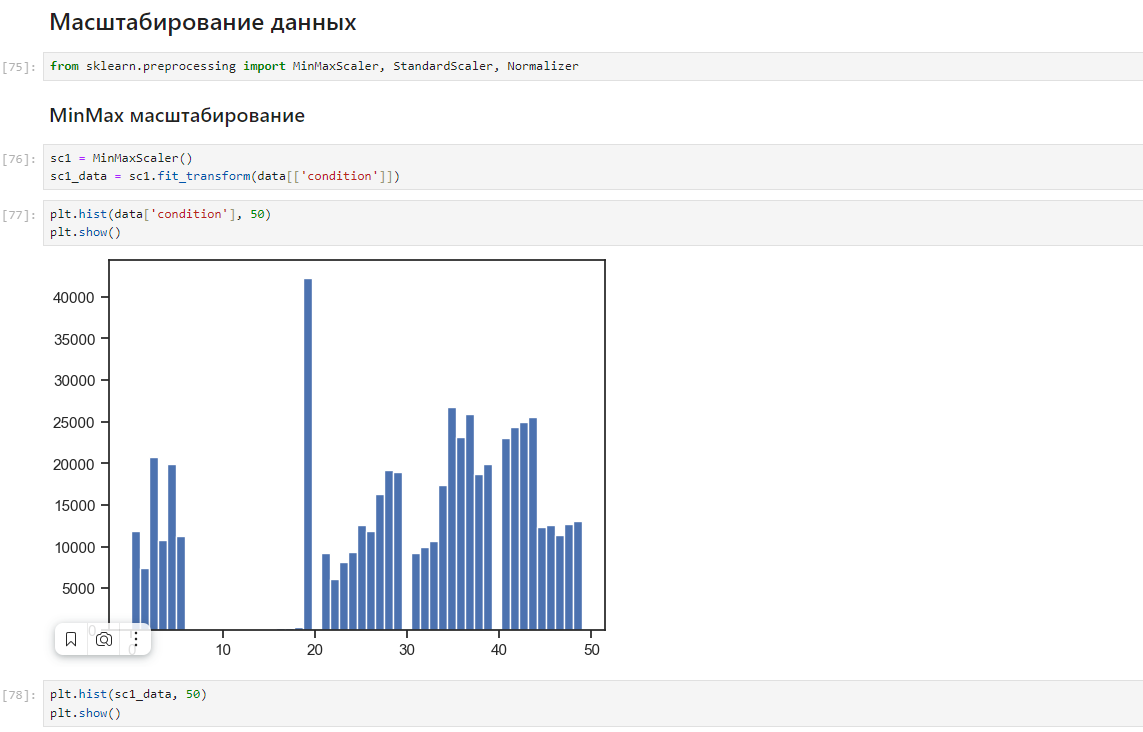
****

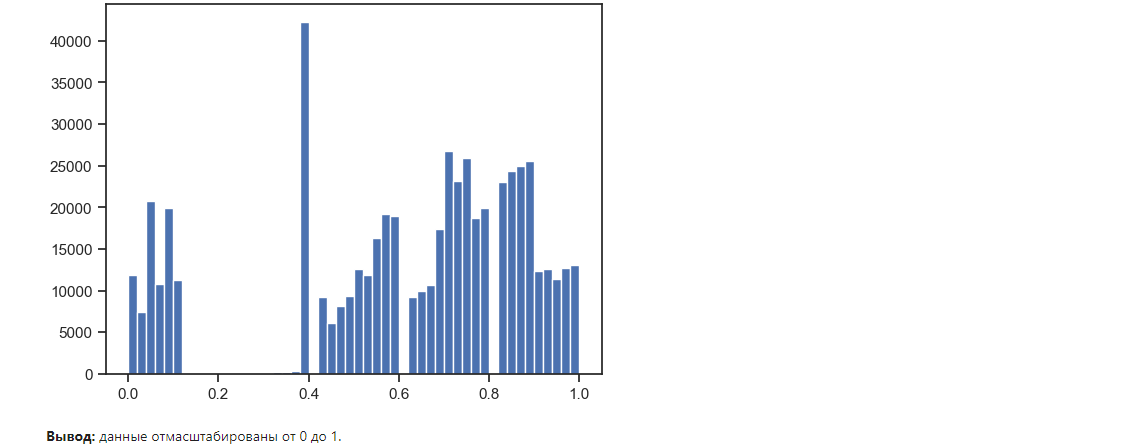
****

****

****

****

****

****