|  |  |
| --- | --- |
| Изображение выглядит как эмблема, герб, нашивка, символ  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | ГОЛОВНОЙ УЧЕБНО-ИССЛЕДОВАТЕЛЬСКИЙ И МЕТОДИЧЕСКИЙ ЦЕНТР |
| ПРОФЕССИОНАЛЬНОЙ РЕАБИЛИТАЦИИ ЛИЦ С ОГРАНИЧЕННЫМИ |
| ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ |
| КАФЕДРА | СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ |

Отчёт по лабораторной работе №2 по курсу «Технологии машинного обучения».

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

Выполнил: Проверил:

Новиков С. А. Гапанюк Ю.Е.

студент группы ИУ5-62Б

Подпись и дата: Подпись и дата:

*2025 г.*

1. **Задание лабораторной работы**

* 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; масшта- бирование данных.

1. **Ячейки Jupyter-ноутбука**

# Выбор и загрузка данных

В качестве датасета будем использовать набор данных, содержащий дан- ные по продажам автомобилей в США. Данный набор доступен по адресу: [https://www.kaggle.com/datasets/gagandeep16/car-sales](http://www.kaggle.com/datasets/gagandeep16/car-sales)

Набор данных имеет следующие атрибуты:

* Manufacturer - марка
* Model - модель
* Sales\_in\_thousands - продажи в тысячах
* year\_resale\_value - годовой объем продаж
* Vehicle\_type - тип автомобиля
* Price\_in\_thousands - цена в тысячах
* Engine\_size - объем двигателя
* Horsepower - лошадиные силы
* Wheelbase - колесная база
* Width - ширина
* Length - длина
* Curb\_weight - масса
* Fuel\_capacity - топливный бак
* Fuel\_efficiency - расход топлива
* Latest\_Launch - начало производства модели
* Power\_perf\_factor - мощностной коэффициент

[1]:

## Импорт библиотек

Импортируем библиотеки с помощью команды import:

**import numpy as np import pandas as pd import seaborn as sns**

**import matplotlib.pyplot as plt**

%**matplotlib** inline sns.set(style=DticksD)

[2]:

## Загрузка данных

Загрузим набор данных:

data = pd.read\_csv('Car\_sales.csv')

[3]:

# Первичный анализ данных

Выведем первые 5 строк датасета:

data.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [3]: | Manufacturer | Model | Sales\_in\_thousands | year\_resale\_value | Vehicle\_type | \ |
| 0 | Acura | Integra | 16.919 | 16.360 | Passenger |  |
| 1 | Acura | TL | 39.384 | 19.875 | Passenger |  |
| 2 | Acura | CL | 14.114 | 18.225 | Passenger |  |
| 3 | Acura | RL | 8.588 | 29.725 | Passenger |  |
| 4 | Audi | A4 | 20.397 | 22.255 | Passenger |  |

Price\_in\_thousands Engine\_size Horsepower Wheelbase Width Length \

0 21.50 1.8 140.0 101.2 67.3 172.4

1 28.40 3.2 225.0 108.1 70.3 192.9

2 NaN 3.2 225.0 106.9 70.6 192.0

3 42.00 3.5 210.0 114.6 71.4 196.6

4 23.99 1.8 150.0 102.6 68.2 178.0

Curb\_weight Fuel\_capacity Fuel\_efficiency Latest\_Launch \

0 2.639 13.2 28.0 2/2/2012

1 3.517 17.2 25.0 6/3/2011

2 3.470 17.2 26.0 1/4/2012

3 3.850 18.0 22.0 3/10/2011

4 2.998 16.4 27.0 10/8/2011

Power\_perf\_factor

0 58.280150

1 91.370778

2 NaN

3 91.389779

4 62.777639

Определим размер датасета:

[4]:

data.shape

[4]: (157, 16)

В датасете 157 строк и 16 столбцов. Определим тип столбцов:

[5]:

data.dtypes

1. : Manufacturer object Model object

Sales\_in\_thousands float64

year\_resale\_value

float64

Vehicle\_type object Price\_in\_thousands float64 Engine\_size float64

Horsepower float64

Wheelbase float64

Width float64

Length float64

Curb\_weight float64

Fuel\_capacity float64

1. :

Fuel\_efficiency float64

Latest\_Launch object Power\_perf\_factor float64 dtype: object

Проверим наличие пропусков:

data.isnull().sum()

1. : Manufacturer 0

Model 0

Sales\_in\_thousands 0

year\_resale\_value 36

Vehicle\_type 0

Price\_in\_thousands 2

Engine\_size 1

Horsepower 1

Wheelbase 1

Width 1

Length 1

Curb\_weight 2

Fuel\_capacity 1

Fuel\_efficiency 3

Latest\_Launch 0

Power\_perf\_factor 2

dtype: int64

Видим, что пропуски наблюдаются в множестве столбцов.

1. :

# Обработка пропусков данных

Удалим колонки, содержащие пустые значения:

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

1. : ((157, 16), (157, 5))

Выведем первые строки датасета на экран:

1. :

data\_new\_1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [8]: | Manufacturer | Model | Sales\_in\_thousands | Vehicle\_type | Latest\_Launch |
| 0 | Acura | Integra | 16.919 | Passenger | 2/2/2012 |
| 1 | Acura | TL | 39.384 | Passenger | 6/3/2011 |
| 2 | Acura | CL | 14.114 | Passenger | 1/4/2012 |
| 3 | Acura | RL | 8.588 | Passenger | 3/10/2011 |
| 4 | Audi | A4 | 20.397 | Passenger | 10/8/2011 |
| .. | … | … | … | … | … |
| 152 | Volvo | V40 | 3.545 | Passenger | 9/21/2011 |
| 153 | Volvo | S70 | 15.245 | Passenger | 11/24/2012 |
| 154 | Volvo | V70 | 17.531 | Passenger | 6/25/2011 |
| 155 | Volvo | C70 | 3.493 | Passenger | 4/26/2011 |
| 156 | Volvo | S80 | 18.969 | Passenger | 11/14/2011 |

[157 rows x 5 columns]

Удалим строки, содержащие пустые значения:

1. :

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

1. : ((157, 16), (117, 16))
2. :

data\_new\_2.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [10]: | Manufacturer | Model | Sales\_in\_thousands | year\_resale\_value | Vehicle\_type | \ |
| 0 | Acura | Integra | 16.919 | 16.360 | Passenger |  |
| 1 | Acura | TL | 39.384 | 19.875 | Passenger |  |
| 3 | Acura | RL | 8.588 | 29.725 | Passenger |  |
| 4 | Audi | A4 | 20.397 | 22.255 | Passenger |  |
| 5 | Audi | A6 | 18.780 | 23.555 | Passenger |  |

Price\_in\_thousands Engine\_size Horsepower Wheelbase Width Length \

0 21.50 1.8 140.0 101.2 67.3 172.4

1 28.40 3.2 225.0 108.1 70.3 192.9

3 42.00 3.5 210.0 114.6 71.4 196.6

4 23.99 1.8 150.0 102.6 68.2 178.0

5 33.95 2.8 200.0 108.7 76.1 192.0

Curb\_weight Fuel\_capacity Fuel\_efficiency Latest\_Launch \

0 2.639 13.2 28.0 2/2/2012

1 3.517 17.2 25.0 6/3/2011

3 3.850 18.0 22.0 3/10/2011

4 2.998 16.4 27.0 10/8/2011

5 3.561 18.5 22.0 8/9/2011

1. :
2. :

Power\_perf\_factor

0 58.280150

1 91.370778

3 91.389779

4 62.777639

5 84.565105

Заполним все пропущенные значения нулями:

data\_new\_3 = data.fillna(0)

Выведем на экран:

data\_new\_3.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [12]: | Manufacturer | Model | Sales\_in\_thousands | year\_resale\_value | Vehicle\_type | \ |
| 0 | Acura | Integra | 16.919 | 16.360 | Passenger |  |
| 1 | Acura | TL | 39.384 | 19.875 | Passenger |  |
| 2 | Acura | CL | 14.114 | 18.225 | Passenger |  |
| 3 | Acura | RL | 8.588 | 29.725 | Passenger |  |
| 4 | Audi | A4 | 20.397 | 22.255 | Passenger |  |

Price\_in\_thousands Engine\_size Horsepower Wheelbase Width Length \

0 21.50 1.8 140.0 101.2 67.3 172.4

1 28.40 3.2 225.0 108.1 70.3 192.9

2 0.00 3.2 225.0 106.9 70.6 192.0

3 42.00 3.5 210.0 114.6 71.4 196.6

4 23.99 1.8 150.0 102.6 68.2 178.0

Curb\_weight Fuel\_capacity Fuel\_efficiency Latest\_Launch \

0 2.639 13.2 28.0 2/2/2012

1 3.517 17.2 25.0 6/3/2011

2 3.470 17.2 26.0 1/4/2012

3 3.850 18.0 22.0 3/10/2011

4 2.998 16.4 27.0 10/8/2011

Power\_perf\_factor

0 58.280150

1 91.370778

2 0.000000

3 91.389779

4 62.777639

1. :

## Импьютация данных

* + 1. **Обработка пропусков в числовых данных**

Выберем числовые столбцы с пропущенными значениями и посчитаем количество пустых значений:

num\_cols = []

**for** col **in** data.columns:

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 / data.shape[0]) \* 100.0, 2) print('Столбец **{}**. Тип данных **{}**. Количество пустых значений **{}**, **{}**%.'.

*‹→*format(col, dt, temp\_null\_count, temp\_perc))

Столбец year\_resale\_value. Тип данных float64. Количество пустых значений 36, 22.93%.

Столбец Price\_in\_thousands. Тип данных float64. Количество пустых значений 2, 1.27%.

Столбец Engine\_size. Тип данных float64. Количество пустых значений 1, 0.64%. Столбец Horsepower. Тип данных float64. Количество пустых значений 1, 0.64%. Столбец Wheelbase. Тип данных float64. Количество пустых значений 1, 0.64%.

Столбец Width. Тип данных float64. Количество пустых значений 1, 0.64%. Столбец Length. Тип данных float64. Количество пустых значений 1, 0.64%. Столбец Curb\_weight. Тип данных float64. Количество пустых значений 2, 1.27%.

Столбец Fuel\_capacity. Тип данных float64. Количество пустых значений 1, 0.64%. Столбец Fuel\_efficiency. Тип данных float64. Количество пустых значений 3, 1.91%.

Столбец Power\_perf\_factor. Тип данных float64. Количество пустых значений 2, 1.27%.

Отфильтруем по столбцам:

1. :

data\_num = data[num\_cols] data\_num

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [14]: | year\_resale\_value | | | Price\_in\_thousands | | Engine\_size | Horsepower | \ | |
| 0 | 16.360 | | | 21.50 | | 1.8 | 140.0 |  | |
| 1 | 19.875 | | | 28.40 | | 3.2 | 225.0 |  | |
| 2 | 18.225 | | | NaN | | 3.2 | 225.0 |  | |
| 3 | 29.725 | | | 42.00 | | 3.5 | 210.0 |  | |
| 4  .. 152 | 22.255  … NaN | | | 23.99  … 24.40 | | 1.8  …  1.9 | 150.0  …  160.0 |  | |
| 153 | NaN | | | 27.50 | | 2.4 | 168.0 |  | |
| 154 | NaN | | | 28.80 | | 2.4 | 168.0 |  | |
| 155 | NaN | | | 45.50 | | 2.3 | 236.0 |  | |
| 156 | NaN | | | 36.00 | | 2.9 | 201.0 |  | |
|  | Wheelbase | Width | Length | | Curb\_weight | Fuel\_capacity | Fuel\_efficiency | | \ |
| 0 | 101.2 | 67.3 | 172.4 | | 2.639 | 13.2 | 28.0 | |  |
| 1 | 108.1 | 70.3 | 192.9 | | 3.517 | 17.2 | 25.0 | |  |
| 2 | 106.9 | 70.6 | 192.0 | | 3.470 | 17.2 | 26.0 | |  |
| 3 | 114.6 | 71.4 | 196.6 | | 3.850 | 18.0 | 22.0 | |  |
| 4 | 102.6 | 68.2 | 178.0 | | 2.998 | 16.4 | 27.0 | |  |
| .. | … | … | … | | … | … | … | |  |
| 152 | 100.5 | 67.6 | 176.6 | | 3.042 | 15.8 | 25.0 | |  |
| 153 | 104.9 | 69.3 | 185.9 | | 3.208 | 17.9 | 25.0 | |  |
| 154 | 104.9 | 69.3 | 186.2 | | 3.259 | 17.9 | 25.0 | |  |
| 155 | 104.9 | 71.5 | 185.7 | | 3.601 | 18.5 | 23.0 | |  |
| 156 | 109.9 | 72.1 | 189.8 | | 3.600 | 21.1 | 24.0 | |  |

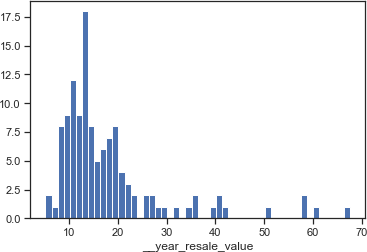
|  |  |
| --- | --- |
|  | Power\_perf\_factor |
| 0 | 58.280150 |
| 1 | 91.370778 |
| 2 | NaN |
| 3 | 91.389779 |
| 4 | 62.777639 |
| .. | … |
| 152 | 66.498812 |
| 153 | 70.654495 |
| 154 | 71.155978 |
| 155 | 101.623357 |
| 156 | 85.735655 |

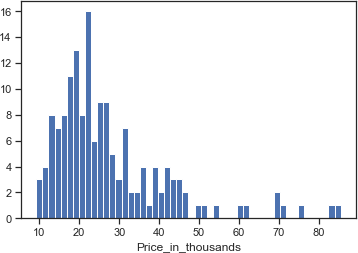
1. :

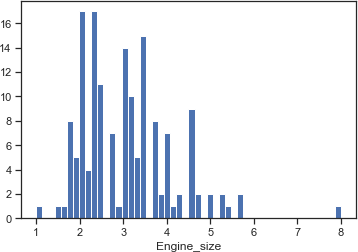
[157 rows x 11 columns]

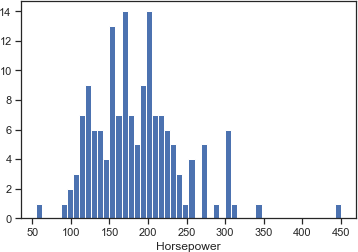
Гистограмма по признакам:

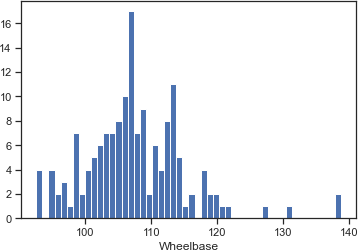
**for** col **in** data\_num: plt.hist(data[col], 50) plt.xlabel(col) plt.show()

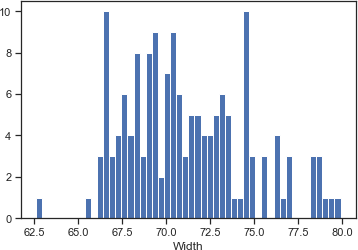


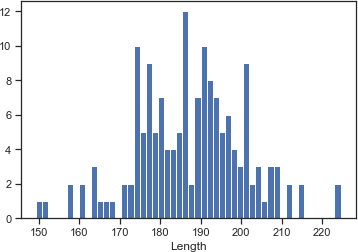


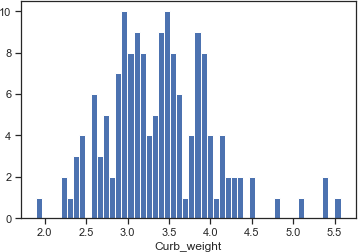


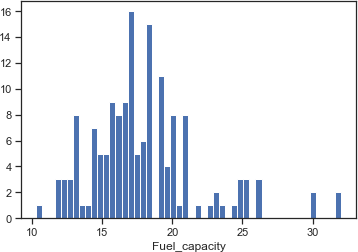


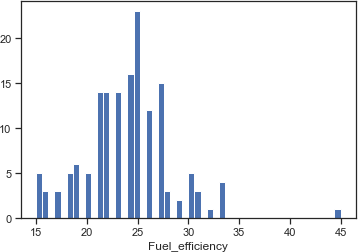


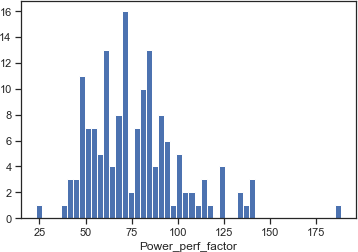












1. :

Будем использовать встроенные средства импьютации библиотеки scikit-learn, доступные по адресу: https://scikit-learn.org/stable/modules/impute.html

1. :

data\_num\_pit = data\_num[['Price\_in\_thousands']]

**from sklearn.impute import** SimpleImputer

**from sklearn.impute import** MissingIndicator

Фильтр для проверки заполнения пустых значений:

1. :

indicator = MissingIndicator()

mask\_missing\_values\_only = indicator.fit\_transform(data\_num\_pit) mask\_missing\_values\_only

1. : array([[False],

[False],

[ True],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[ True],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

1. :

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False],

[False]])

Проведем импьютацию различными показателями центра распределения:

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

1. :

**def** test\_num\_impute(strategy\_param):

imp\_num = SimpleImputer(strategy=strategy\_param) data\_num\_imp = imp\_num.fit\_transform(data\_num\_pit)

**return** data\_num\_imp[mask\_missing\_values\_only]

1. :

strategies[0], test\_num\_impute(strategies[0])

1. : ('mean', array([27.39075484, 27.39075484]))
2. :

strategies[1], test\_num\_impute(strategies[1])

1. : ('median', array([22.799, 22.799]))
2. :

strategies[2], test\_num\_impute(strategies[2])

1. : ('most\_frequent', array([12.64, 12.64]))

Создадим функцию, позволяющую задавать столбец и вид импьютации:

1. :



**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]

Проверим работу функции по продажам автомобилей:

1. :

data[[' year\_resale\_value']].describe()

|  |  |  |
| --- | --- | --- |
| [25]: |  | year\_resale\_value |
|  | count | 121.000000 |
|  | mean | 18.072975 |
|  | std | 11.453384 |
|  | min | 5.160000 |
|  | 25% | 11.260000 |
|  | 50% | 14.180000 |
|  | 75% | 19.875000 |
|  | max | 67.550000 |

1. :

test\_num\_impute\_col(data, ' year\_resale\_value', strategies[0])

1. : (' year\_resale\_value',

'mean', 36, 18.07297520661157, 18.07297520661157)

1. :

test\_num\_impute\_col(data, ' year\_resale\_value', strategies[1])

1. : (' year\_resale\_value',

'median', 36, 14.18, 14.18)

1. :

test\_num\_impute\_col(data, ' year\_resale\_value', strategies[2])

1. : (' year\_resale\_value',

'most\_frequent',

36, 7.75, 7.75)

1. :
2. :

## Обработка пропусков в категориальных данных

Так как в датасете нет пропусков среди столбца “Производитель”, то искуственно подправим датасет и загрузим его:

data\_mod = pd.read\_csv('Car\_sales\_mod.csv')

Проверим категориальный признак:

cat\_cols = []

**for** col **in** data.columns:

temp\_null\_count = data\_mod[data\_mod[col].isnull()].shape[0] dt = str(data\_mod[col].dtype)

**if** temp\_null\_count>0 **and** (dt=='object'): cat\_cols.append(col)

temp\_perc = round((temp\_null\_count / data.shape[0]) \* 100.0, 2) print('Столбец **{}**. Тип данных **{}**. Количество пустых значений **{}**, **{}**%.'.

*‹→*format(col, dt, temp\_null\_count, temp\_perc))

Столбец Manufacturer. Тип данных object. Количество пустых значений 15, 9.55%.

Его и будем использовать:

1. :

cat\_temp\_data = data\_mod[['Manufacturer']] cat\_temp\_data.head()

|  |  |  |
| --- | --- | --- |
| [31]: |  | Manufacturer |
|  | 0 | Acura |
|  | 1 | Acura |
|  | 2 | Acura |
|  | 3 | Acura |
|  | 4 | Audi |

1. :

cat\_temp\_data['Manufacturer'].unique()

1. : array(['Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', nan,

'Dodge', 'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep',

'Lexus', 'Mitsubishi', 'Mercury', 'Mercedes-B', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

1. :

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

[33]: (15, 1)

Импьютация наиболее частыми значениями:

1. :

imp2 = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent') data\_imp2 = imp2.fit\_transform(cat\_temp\_data)

data\_imp2

1. : array([['Acura'],

['Acura'],

['Acura'],

['Acura'],

['Audi'],

['Audi'],

['Audi'],

['BMW'],

['BMW'],

['BMW'],

['Buick'],

['Buick'],

['Buick'],

['Buick'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Honda'],

['Honda'],

['Honda'],

['Honda'],

['Honda'],

['Hyundai'],

['Hyundai'],

['Hyundai'],

['Infiniti'],

['Jaguar'],

['Jeep'],

['Jeep'],

['Jeep'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'],

['Dodge'],

['Dodge'],

['Dodge'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Plymouth'],

['Plymouth'],

['Plymouth'],

['Plymouth'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Porsche'],

['Porsche'],

['Porsche'],

['Saab'],

['Saab'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Subaru'],

['Subaru'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo']], dtype=object)

1. :

np.unique(data\_imp2)

1. : array(['Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge',

'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep', 'Lexus', 'Mercedes-B', 'Mercury', 'Mitsubishi', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

1. :

Наблюдаем отсутствие пустых значений. Импьютация константой:

1. : array([['Acura'],

imp3 = SimpleImputer(missing\_values=np.nan, strategy='constant', fill\_value='???') data\_imp3 = imp3.fit\_transform(cat\_temp\_data)

data\_imp3

['Acura'],

['Acura'],

['Acura'],

['Audi'],

['Audi'],

['Audi'],

['BMW'],

['BMW'],

['BMW'],

['Buick'],

['Buick'],

['Buick'],

['Buick'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Cadillac'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'],

['Chevrolet'], ['???'],

['???'],

['???'],

['???'],

['???'],

['???'],

['???'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Dodge'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Ford'],

['Honda'],

['Honda'],

['Honda'],

['Honda'],

['Honda'],

['Hyundai'],

['Hyundai'],

['Hyundai'],

['Infiniti'],

['Jaguar'],

['Jeep'],

['Jeep'],

['Jeep'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'],

['Lexus'], ['???'],

['???'],

['???'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mitsubishi'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercury'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Mercedes-B'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Nissan'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Oldsmobile'],

['Plymouth'],

['Plymouth'],

['Plymouth'],

['Plymouth'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Pontiac'],

['Porsche'],

['Porsche'],

['Porsche'],

['Saab'],

['Saab'], ['???'],

['???'],

['???'],

['???'],

['???'],

['Subaru'],

['Subaru'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Toyota'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volkswagen'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo'],

['Volvo']], dtype=object)

1. :

np.unique(data\_imp3)

1. : array(['???', 'Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet',

'Dodge', 'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep',

'Lexus', 'Mercedes-B', 'Mercury', 'Mitsubishi', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

1. :

data\_imp3[data\_imp3==0].size

[38]: 0

Значения были заменены на “???”.

## Преобразование категориальных признаков в числовые

1. :

cat\_enc = pd.DataFrame({'c1':data\_imp2.T[0]}) cat\_enc

|  |  |  |
| --- | --- | --- |
| [39]: |  | c1 |
|  | 0 | Acura |
|  | 1 | Acura |
|  | 2 | Acura |
|  | 3 | Acura |
|  | 4 | Audi |
|  | .. | … |
|  | 152 | Volvo |
|  | 153 | Volvo |
|  | 154 | Volvo |
|  | 155 | Volvo |
|  | 156 | Volvo |
|  | [157 | rows x 1 columns] |

# Кодирование категорий целочисленными значениями

## LabelEncoder

1. :

**from sklearn.preprocessing import** LabelEncoder

1. :

cat\_enc['c1'].unique()

1. : array(['Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge',

'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep', 'Lexus', 'Mitsubishi', 'Mercury', 'Mercedes-B', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

1. :

le = LabelEncoder()

1. :

cat\_enc\_le = le.fit\_transform(cat\_enc['c1'])

1. :

le.classes\_

[44]: array(['Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge',

'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep', 'Lexus', 'Mercedes-B', 'Mercury', 'Mitsubishi', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [45]: | cat\_enc\_le |  | | | | | | | | | | | | | | | |
| [45]: | array([ 0, | 0, | 0, | 0, | 1, | 1, | 1, | 2, | 2, | 2, | 3, | 3, | 3, | 3, | 4, | 4, | 4, |
|  | 4, | 4, | 5, | 5, | 5, | 5, | 5, | 5, | 5, | 5, | 5, | 6, | 6, | 6, | 6, | 6, | 6, |
|  | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 6, | 7, | 7, | 7, | 7, | 7, |
|  | 7, | 7, | 7, | 7, | 7, | 7, | 8, | 8, | 8, | 8, | 8, | 9, | 9, | 9, | 10, | 11, | 12, |
|  | 12, | 12, | 13, | 13, | 13, | 13, | 13, | 13, | 6, | 6, | 6, | 16, | 16, | 16, | 16, | 16, | 16, |
|  | 16, | 15, | 15, | 15, | 15, | 15, | 15, | 14, | 14, | 14, | 14, | 14, | 14, | 14, | 14, | 14, | 17, |
|  | 17, | 17, | 17, | 17, | 17, | 17, | 18, | 18, | 18, | 18, | 18, | 18, | 19, | 19, | 19, | 19, | 20, |
|  | 20, | 20, | 20, | 20, | 20, | 21, | 21, | 21, | 22, | 22, | 6, | 6, | 6, | 6, | 6, | 23, | 23, |
|  | 24, | 24, | 24, | 24, | 24, | 24, | 24, | 24, | 24, | 25, | 25, | 25, | 25, | 25, | 25, | 26, | 26, |
|  | 26, | 26, | 26, | 26]) |  |  |  |  |  |  |  |  |  |  |  |  |  |

[46]:

np.unique(cat\_enc\_le)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [46]: | array([ 0, 1, 2, 3, | 4, | 5, | 6, | 7, | 8, | 9, | 10, | 11, | 12, | 13, | 14, | 15, | 16, | | |
|  | 17, 18, 19, 20, | 21, | 22, | 23, | 24, | 25, | 26]) | |  |  |  | | | |  |  |
| [47]: | le.inverse\_transform([ | 0, | 1, | 2, | 3, | 4, | 5, 6, | | 7, | 8, | 9, 10, 11, 12, | | | | 13, | 14, |
|  | *‹→*15, 16,  17, 18, 19, 20, 21, 22, 23, 24, 25, 26]) | | | | | | | | | | | | | | | |

1. : array(['Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge',

'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep', 'Lexus', 'Mercedes-B', 'Mercury', 'Mitsubishi', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object)

## OrdinalEncoder

1. :

**from sklearn.preprocessing import** OrdinalEncoder

1. :

data\_oe = data\_mod[['Manufacturer', 'Model']] data\_oe.head()

1. : Manufacturer Model
   1. Acura Integra
   2. Acura TL
   3. Acura CL
   4. Acura RL
   5. Audi A4
2. :

imp4 = SimpleImputer(missing\_values=np.nan, strategy='constant', fill\_value='???') data\_oe\_filled = imp4.fit\_transform(data\_oe)

data\_oe\_filled

1. : array([['Acura', 'Integra'],

['Acura', 'TL'],

['Acura', 'CL'],

['Acura', 'RL'],

['Audi', 'A4'],

['Audi', 'A6'],

['Audi', 'A8'],

['BMW', '323i'],

['BMW', '328i'],

['BMW', '528i'],

['Buick', 'Century'],

['Buick', 'Regal'],

['Buick', 'Park Avenue'],

['Buick', 'LeSabre'],

['Cadillac', 'DeVille'],

['Cadillac', 'Seville'],

['Cadillac', 'Eldorado'],

['Cadillac', 'Catera'],

['Cadillac', 'Escalade'],

['Chevrolet', 'Cavalier'],

['Chevrolet', 'Malibu'],

['Chevrolet', 'Lumina'], ['Chevrolet', 'Monte Carlo'], ['Chevrolet', 'Camaro'],

['Chevrolet', 'Corvette'],

['Chevrolet', 'Prizm'],

['Chevrolet', 'Metro'],

['Chevrolet', 'Impala'],

['???', 'Sebring Coupe'],

['???', 'Sebring Conv.'],

['???', 'Concorde'],

['???', 'Cirrus'],

['???', 'LHS'],

['???', 'Town & Country'],

['???', '300M'],

['Dodge', 'Neon'],

['Dodge', 'Avenger'],

['Dodge', 'Stratus'],

['Dodge', 'Intrepid'],

['Dodge', 'Viper'],

['Dodge', 'Ram Pickup'],

['Dodge', 'Ram Wagon'],

['Dodge', 'Ram Van'],

['Dodge', 'Dakota'],

['Dodge', 'Durango'],

['Dodge', 'Caravan'],

['Ford', 'Escort'],

['Ford', 'Mustang'],

['Ford', 'Contour'],

['Ford', 'Taurus'],

['Ford', 'Focus'],

['Ford', 'Crown Victoria'],

['Ford', 'Explorer'],

['Ford', 'Windstar'],

['Ford', 'Expedition'],

['Ford', 'Ranger'],

['Ford', 'F-Series'],

['Honda', 'Civic'],

['Honda', 'Accord'],

['Honda', 'CR-V'],

['Honda', 'Passport'],

['Honda', 'Odyssey'],

['Hyundai', 'Accent'],

['Hyundai', 'Elantra'],

['Hyundai', 'Sonata'],

['Infiniti', 'I30'],

['Jaguar', 'S-Type'],

['Jeep', 'Wrangler'],

['Jeep', 'Cherokee'],

['Jeep', 'Grand Cherokee'],

['Lexus', 'ES300'],

['Lexus', 'GS300'],

['Lexus', 'GS400'],

['Lexus', 'LS400'],

['Lexus', 'LX470'],

['Lexus', 'RX300'],

['???', 'Continental'],

['???', 'Town car'],

['???', 'Navigator'],

['Mitsubishi', 'Mirage'],

['Mitsubishi', 'Eclipse'],

['Mitsubishi', 'Galant'],

['Mitsubishi', 'Diamante'],

['Mitsubishi', '3000GT'],

['Mitsubishi', 'Montero'], ['Mitsubishi', 'Montero Sport'], ['Mercury', 'Mystique'],

['Mercury', 'Cougar'],

['Mercury', 'Sable'], ['Mercury', 'Grand Marquis'], ['Mercury', 'Mountaineer'],

['Mercury', 'Villager'],

['Mercedes-B', 'C-Class'],

['Mercedes-B', 'E-Class'],

['Mercedes-B', 'S-Class'],

['Mercedes-B', 'SL-Class'],

['Mercedes-B', 'SLK'],

['Mercedes-B', 'SLK230'],

['Mercedes-B', 'CLK Coupe'],

['Mercedes-B', 'CL500'],

['Mercedes-B', 'M-Class'],

['Nissan', 'Sentra'],

['Nissan', 'Altima'],

['Nissan', 'Maxima'],

['Nissan', 'Quest'],

['Nissan', 'Pathfinder'],

['Nissan', 'Xterra'],

['Nissan', 'Frontier'],

['Oldsmobile', 'Cutlass'],

['Oldsmobile', 'Intrigue'],

['Oldsmobile', 'Alero'],

['Oldsmobile', 'Aurora'],

['Oldsmobile', 'Bravada'], ['Oldsmobile', 'Silhouette'], ['Plymouth', 'Neon'],

['Plymouth', 'Breeze'],

['Plymouth', 'Voyager'],

['Plymouth', 'Prowler'],

['Pontiac', 'Sunfire'],

['Pontiac', 'Grand Am'],

['Pontiac', 'Firebird'],

['Pontiac', 'Grand Prix'],

['Pontiac', 'Bonneville'],

['Pontiac', 'Montana'],

['Porsche', 'Boxter'], ['Porsche', 'Carrera Coupe'], ['Porsche', 'Carrera Cabrio'], ['Saab', '5-Sep'],

['Saab', '3-Sep'],

['???', 'SL'],

['???', 'SC'],

['???', 'SW'],

['???', 'LW'],

['???', 'LS'],

['Subaru', 'Outback'],

['Subaru', 'Forester'],

['Toyota', 'Corolla'],

['Toyota', 'Camry'],

['Toyota', 'Avalon'],

['Toyota', 'Celica'],

['Toyota', 'Tacoma'],

['Toyota', 'Sienna'],

['Toyota', 'RAV4'],

['Toyota', '4Runner'],

['Toyota', 'Land Cruiser'],

['Volkswagen', 'Golf'],

['Volkswagen', 'Jetta'],

['Volkswagen', 'Passat'],

['Volkswagen', 'Cabrio'],

['Volkswagen', 'GTI'],

['Volkswagen', 'Beetle'],

['Volvo', 'S40'],

['Volvo', 'V40'],

['Volvo', 'S70'],

['Volvo', 'V70'],

['Volvo', 'C70'],

['Volvo', 'S80']], dtype=object)

1. :

oe = OrdinalEncoder()

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

|  |  |  |  |
| --- | --- | --- | --- |
| [51]: | array([[ | 1., | 79.], |
|  | [ | 1., | 143.], |
|  | [ | 1., | 25.], |
|  | [ | 1., | 115.], |
|  | [ | 2., | 8.], |
|  | [ | 2., | 9.], |
|  | [ | 2., | 10.], |
|  | [ | 3., | 3.], |
|  | [ | 3., | 4.], |
|  | [ | 3., | 7.], |
|  | [ | 4., | 38.], |
|  | [ | 4., | 121.], |
|  | [ | 4., | 107.], |
|  | [ | 4., | 89.], |
|  | [ | 5., | 51.], |
|  | [ | 5., | 137.], |
|  | [ | 5., | 58.], |
|  | [ | 5., | 35.], |
|  | [ | 5., | 59.], |
|  | [ | 6., | 36.], |
|  | [ | 6., | 92.], |
|  | [ | 6., | 90.], |
|  | [ | 6., | 97.], |
|  | [ | 6., | 30.], |
|  | [ | 6., | 46.], |
|  | [ | 6., | 111.], |
|  | [ | 6., | 94.], |
|  | [ | 6., | 78.], |
|  | [ | 0., | 135.], |
|  | [ | 0., | 134.], |
|  | [ | 0., | 42.], |
|  | [ | 0., | 40.], |
|  | [ | 0., | 83.], |
|  | [ | 0., | 146.], |
|  | [ | 0., | 2.], |
|  | [ | 7., | 104.], |
|  | [ | 7., | 17.], |
|  | [ | 7., | 141.], |
|  | [ | 7., | 80.], |
|  | [ | 7., | 151.], |
|  | [ | 7., | 117.], |
|  | [ | 7., | 119.], |
|  | [ | 7., | 118.], |
|  | [ | 7., | 50.], |
|  | [ | 7., | 53.], |
|  | [ | 7., | 32.], |
|  | [ | 8., | 60.], |
|  | [ | 8., | 101.], |
|  | [ | 8., | 44.], |

|  |  |  |
| --- | --- | --- |
| [ | 8., | 145.], |
| [ | 8., | 65.], |
| [ | 8., | 48.], |
| [ | 8., | 62.], |
| [ | 8., | 153.], |
| [ | 8., | 61.], |
| [ | 8., | 120.], |
| [ | 8., | 63.], |
| [ | 9., | 41.], |
| [ | 9., | 12.], |
| [ | 9., | 28.], |
| [ | 9., | 109.], |
| [ | 9., | 105.], |
| [ | 10., | 11.], |
| [ | 10., | 57.], |
| [ | 10., | 140.], |
| [ | 11., | 77.], |
| [ | 12., | 123.], |
| [ | 13., | 154.], |
| [ | 13., | 39.], |
| [ | 13., | 74.], |
| [ | 14., | 55.], |
| [ | 14., | 68.], |
| [ | 14., | 69.], |
| [ | 14., | 85.], |
| [ | 14., | 87.], |
| [ | 14., | 116.], |
| [ | 0., | 43.], |
| [ | 0., | 147.], |
| [ | 0., | 103.], |
| [ | 17., | 95.], |
| [ | 17., | 56.], |
| [ | 17., | 71.], |
| [ | 17., | 52.], |
| [ | 17., | 1.], |
| [ | 17., | 98.], |
| [ | 17., | 99.], |
| [ | 16., | 102.], |
| [ | 16., | 47.], |
| [ | 16., | 133.], |
| [ | 16., | 75.], |
| [ | 16., | 100.], |
| [ | 16., | 150.], |
| [ | 15., | 23.], |
| [ | 15., | 54.], |
| [ | 15., | 122.], |
| [ | 15., | 129.], |
| [ | 15., | 130.], |
| [ | 15., | 131.], |
| [ | 15., | 27.], |
| [ | 15., | 26.], |
| [ | 15., | 91.], |
| [ | 18., | 136.], |

|  |  |  |
| --- | --- | --- |
| [ | 18., | 14.], |
| [ | 18., | 93.], |
| [ | 18., | 113.], |
| [ | 18., | 110.], |
| [ | 18., | 155.], |
| [ | 18., | 67.], |
| [ | 19., | 49.], |
| [ | 19., | 81.], |
| [ | 19., | 13.], |
| [ | 19., | 15.], |
| [ | 19., | 21.], |
| [ | 19., | 139.], |
| [ | 20., | 104.], |
| [ | 20., | 22.], |
| [ | 20., | 152.], |
| [ | 20., | 112.], |
| [ | 21., | 142.], |
| [ | 21., | 73.], |
| [ | 21., | 64.], |
| [ | 21., | 76.], |
| [ | 21., | 19.], |
| [ | 21., | 96.], |
| [ | 22., | 20.], |
| [ | 22., | 34.], |
| [ | 22., | 33.], |
| [ | 23., | 6.], |
| [ | 23., | 0.], |
| [ | 0., | 128.], |
| [ | 0., | 127.], |
| [ | 0., | 132.], |
| [ | 0., | 86.], |
| [ | 0., | 84.], |
| [ | 24., | 106.], |
| [ | 24., | 66.], |
| [ | 25., | 45.], |
| [ | 25., | 31.], |
| [ | 25., | 16.], |
| [ | 25., | 37.], |
| [ | 25., | 144.], |
| [ | 25., | 138.], |
| [ | 25., | 114.], |
| [ | 25., | 5.], |
| [ | 25., | 88.], |
| [ | 26., | 72.], |
| [ | 26., | 82.], |
| [ | 26., | 108.], |
| [ | 26., | 29.], |
| [ | 26., | 70.], |
| [ | 26., | 18.], |
| [ | 27., | 124.], |
| [ | 27., | 148.], |
| [ | 27., | 125.], |
| [ | 27., | 149.], |

1. :

[ 27., 24.],

[ 27., 126.]])

Уникальные значения столбца “Производитель”:

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [52]: | array([ 0., | 1., 2., 3., 4., 5., 6., 7., 8., 9., | 10., | 11., | 12., |
|  | 13., | 14., 15., 16., 17., 18., 19., 20., 21., 22., | 23., | 24., | 25., |
|  | 26., | 27.]) |  |  |  |

Уникальные значения столбца “Модель”:

1. :

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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [53]: | array([ 0., | 1., | 2., | 3., | 4., | 5., | 6., | 7., | 8., | 9., | 10., |
|  | 11., | 12., | 13., | 14., | 15., | 16., | 17., | 18., | 19., | 20., | 21., |
|  | 22., | 23., | 24., | 25., | 26., | 27., | 28., | 29., | 30., | 31., | 32., |
|  | 33., | 34., | 35., | 36., | 37., | 38., | 39., | 40., | 41., | 42., | 43., |
|  | 44., | 45., | 46., | 47., | 48., | 49., | 50., | 51., | 52., | 53., | 54., |
|  | 55., | 56., | 57., | 58., | 59., | 60., | 61., | 62., | 63., | 64., | 65., |
|  | 66., | 67., | 68., | 69., | 70., | 71., | 72., | 73., | 74., | 75., | 76., |
|  | 77., | 78., | 79., | 80., | 81., | 82., | 83., | 84., | 85., | 86., | 87., |
|  | 88., | 89., | 90., | 91., | 92., | 93., | 94., | 95., | 96., | 97., | 98., |
|  | 99., | 100., | 101., | 102., | 103., | 104., | 105., | 106., | 107., | 108., | 109., |
|  | 110., | 111., | 112., | 113., | 114., | 115., | 116., | 117., | 118., | 119., | 120., |
|  | 121., | 122., | 123., | 124., | 125., | 126., | 127., | 128., | 129., | 130., | 131., |
|  | 132., | 133., | 134., | 135., | 136., | 137., | 138., | 139., | 140., | 141., | 142., |
|  | 143., | 144., | 145., | 146., | 147., | 148., | 149., | 150., | 151., | 152., | 153., |
|  | 154., | 155.]) |  |  |  |  |  |  |  |  |  |

Все значения:

1. :

oe.categories\_

1. : [array(['???', 'Acura', 'Audi', 'BMW', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge', 'Ford', 'Honda', 'Hyundai', 'Infiniti', 'Jaguar', 'Jeep',

'Lexus', 'Mercedes-B', 'Mercury', 'Mitsubishi', 'Nissan', 'Oldsmobile', 'Plymouth', 'Pontiac', 'Porsche', 'Saab', 'Subaru', 'Toyota', 'Volkswagen', 'Volvo'], dtype=object),

array(['3-Sep', '3000GT', '300M', '323i', '328i', '4Runner', '5-Sep',

'528i', 'A4', 'A6', 'A8', 'Accent', 'Accord', 'Alero', 'Altima',

'Aurora', 'Avalon', 'Avenger', 'Beetle', 'Bonneville', 'Boxter',

'Bravada', 'Breeze', 'C-Class', 'C70', 'CL', 'CL500', 'CLK Coupe',

'CR-V', 'Cabrio', 'Camaro', 'Camry', 'Caravan', 'Carrera Cabrio', 'Carrera Coupe', 'Catera', 'Cavalier', 'Celica', 'Century', 'Cherokee', 'Cirrus', 'Civic', 'Concorde', 'Continental', 'Contour', 'Corolla', 'Corvette', 'Cougar', 'Crown Victoria', 'Cutlass', 'Dakota', 'DeVille', 'Diamante', 'Durango', 'E-Class',

'ES300', 'Eclipse', 'Elantra', 'Eldorado', 'Escalade', 'Escort', 'Expedition', 'Explorer', 'F-Series', 'Firebird', 'Focus', 'Forester', 'Frontier', 'GS300', 'GS400', 'GTI', 'Galant', 'Golf', 'Grand Am', 'Grand Cherokee', 'Grand Marquis', 'Grand Prix', 'I30', 'Impala', 'Integra', 'Intrepid', 'Intrigue', 'Jetta', 'LHS', 'LS',

'LS400', 'LW', 'LX470', 'Land Cruiser', 'LeSabre', 'Lumina',

'M-Class', 'Malibu', 'Maxima', 'Metro', 'Mirage', 'Montana', 'Monte Carlo', 'Montero', 'Montero Sport', 'Mountaineer',

'Mustang', 'Mystique', 'Navigator', 'Neon', 'Odyssey', 'Outback', 'Park Avenue', 'Passat', 'Passport', 'Pathfinder', 'Prizm', 'Prowler', 'Quest', 'RAV4', 'RL', 'RX300', 'Ram Pickup', 'Ram Van',

'Ram Wagon', 'Ranger', 'Regal', 'S-Class', 'S-Type', 'S40', 'S70',

'S80', 'SC', 'SL', 'SL-Class', 'SLK', 'SLK230', 'SW', 'Sable',

'Sebring Conv.', 'Sebring Coupe', 'Sentra', 'Seville', 'Sienna', 'Silhouette', 'Sonata', 'Stratus', 'Sunfire', 'TL', 'Tacoma',

'Taurus', 'Town & Country', 'Town car', 'V40', 'V70', 'Villager', 'Viper', 'Voyager', 'Windstar', 'Wrangler', 'Xterra'], dtype=object)]

1. :

oe.inverse\_transform(cat\_enc\_oe)

1. : array([['Acura', 'Integra'],

['Acura', 'TL'],

['Acura', 'CL'],

['Acura', 'RL'],

['Audi', 'A4'],

['Audi', 'A6'],

['Audi', 'A8'],

['BMW', '323i'],

['BMW', '328i'],

['BMW', '528i'],

['Buick', 'Century'],

['Buick', 'Regal'],

['Buick', 'Park Avenue'],

['Buick', 'LeSabre'],

['Cadillac', 'DeVille'],

['Cadillac', 'Seville'],

['Cadillac', 'Eldorado'],

['Cadillac', 'Catera'],

['Cadillac', 'Escalade'],

['Chevrolet', 'Cavalier'],

['Chevrolet', 'Malibu'],

['Chevrolet', 'Lumina'], ['Chevrolet', 'Monte Carlo'], ['Chevrolet', 'Camaro'],

['Chevrolet', 'Corvette'],

['Chevrolet', 'Prizm'],

['Chevrolet', 'Metro'],

['Chevrolet', 'Impala'],

['???', 'Sebring Coupe'],

['???', 'Sebring Conv.'],

['???', 'Concorde'],

['???', 'Cirrus'],

['???', 'LHS'],

['???', 'Town & Country'],

['???', '300M'],

['Dodge', 'Neon'],

['Dodge', 'Avenger'],

['Dodge', 'Stratus'],

['Dodge', 'Intrepid'],

['Dodge', 'Viper'],

['Dodge', 'Ram Pickup'],

['Dodge', 'Ram Wagon'],

['Dodge', 'Ram Van'],

['Dodge', 'Dakota'],

['Dodge', 'Durango'],

['Dodge', 'Caravan'],

['Ford', 'Escort'],

['Ford', 'Mustang'],

['Ford', 'Contour'],

['Ford', 'Taurus'],

['Ford', 'Focus'],

['Ford', 'Crown Victoria'],

['Ford', 'Explorer'],

['Ford', 'Windstar'],

['Ford', 'Expedition'],

['Ford', 'Ranger'],

['Ford', 'F-Series'],

['Honda', 'Civic'],

['Honda', 'Accord'],

['Honda', 'CR-V'],

['Honda', 'Passport'],

['Honda', 'Odyssey'],

['Hyundai', 'Accent'],

['Hyundai', 'Elantra'],

['Hyundai', 'Sonata'],

['Infiniti', 'I30'],

['Jaguar', 'S-Type'],

['Jeep', 'Wrangler'],

['Jeep', 'Cherokee'],

['Jeep', 'Grand Cherokee'],

['Lexus', 'ES300'],

['Lexus', 'GS300'],

['Lexus', 'GS400'],

['Lexus', 'LS400'],

['Lexus', 'LX470'],

['Lexus', 'RX300'],

['???', 'Continental'],

['???', 'Town car'],

['???', 'Navigator'],

['Mitsubishi', 'Mirage'],

['Mitsubishi', 'Eclipse'],

['Mitsubishi', 'Galant'],

['Mitsubishi', 'Diamante'],

['Mitsubishi', '3000GT'],

['Mitsubishi', 'Montero'], ['Mitsubishi', 'Montero Sport'], ['Mercury', 'Mystique'],

['Mercury', 'Cougar'],

['Mercury', 'Sable'], ['Mercury', 'Grand Marquis'], ['Mercury', 'Mountaineer'],

['Mercury', 'Villager'],

['Mercedes-B', 'C-Class'],

['Mercedes-B', 'E-Class'],

['Mercedes-B', 'S-Class'],

['Mercedes-B', 'SL-Class'],

['Mercedes-B', 'SLK'],

['Mercedes-B', 'SLK230'],

['Mercedes-B', 'CLK Coupe'],

['Mercedes-B', 'CL500'],

['Mercedes-B', 'M-Class'],

['Nissan', 'Sentra'],

['Nissan', 'Altima'],

['Nissan', 'Maxima'],

['Nissan', 'Quest'],

['Nissan', 'Pathfinder'],

['Nissan', 'Xterra'],

['Nissan', 'Frontier'],

['Oldsmobile', 'Cutlass'],

['Oldsmobile', 'Intrigue'],

['Oldsmobile', 'Alero'],

['Oldsmobile', 'Aurora'],

['Oldsmobile', 'Bravada'], ['Oldsmobile', 'Silhouette'], ['Plymouth', 'Neon'],

['Plymouth', 'Breeze'],

['Plymouth', 'Voyager'],

['Plymouth', 'Prowler'],

['Pontiac', 'Sunfire'],

['Pontiac', 'Grand Am'],

['Pontiac', 'Firebird'],

['Pontiac', 'Grand Prix'],

['Pontiac', 'Bonneville'],

['Pontiac', 'Montana'],

['Porsche', 'Boxter'], ['Porsche', 'Carrera Coupe'], ['Porsche', 'Carrera Cabrio'], ['Saab', '5-Sep'],

['Saab', '3-Sep'],

['???', 'SL'],

['???', 'SC'],

['???', 'SW'],

['???', 'LW'],

['???', 'LS'],

['Subaru', 'Outback'],

['Subaru', 'Forester'],

['Toyota', 'Corolla'],

['Toyota', 'Camry'],

['Toyota', 'Avalon'],

['Toyota', 'Celica'],

['Toyota', 'Tacoma'],

['Toyota', 'Sienna'],

['Toyota', 'RAV4'],

['Toyota', '4Runner'],

['Toyota', 'Land Cruiser'],

['Volkswagen', 'Golf'],

['Volkswagen', 'Jetta'],

['Volkswagen', 'Passat'],

['Volkswagen', 'Cabrio'],

['Volkswagen', 'GTI'],

['Volkswagen', 'Beetle'],

['Volvo', 'S40'],

['Volvo', 'V40'],

['Volvo', 'S70'],

['Volvo', 'V70'],

['Volvo', 'C70'],

['Volvo', 'S80']], dtype=object)

1. :

## Кодирование шкал порядка

Для кодирования шкал порядка воспользуемся функцией map:



sizes = ['small', 'medium', 'large', 'small', 'medium', 'large', 'small',

*‹→*'medium', 'large']

1. :

pd\_sizes = pd.DataFrame(data={'sizes':sizes}) pd\_sizes

|  |  |  |
| --- | --- | --- |
| [57]: |  | sizes |
|  | 0 | small |
|  | 1 | medium |
|  | 2 | large |
|  | 3 | small |
|  | 4 | medium |
|  | 5 | large |
|  | 6 | small |
|  | 7 | medium |
|  | 8 | large |

1. :

pd\_sizes['sizes\_codes'] = pd\_sizes['sizes'].map({'small':1, 'medium':2, 'large':

*‹→*3})

pd\_sizes

|  |  |  |  |
| --- | --- | --- | --- |
| [58]: |  | sizes | sizes\_codes |
|  | 0 | small | 1 |
|  | 1 | medium | 2 |
|  | 2 | large | 3 |
|  | 3 | small | 1 |
|  | 4 | medium | 2 |
|  | 5 | large | 3 |
|  | 6 | small | 1 |
|  | 7 | medium | 2 |
|  | 8 | large | 3 |

1. :

pd\_sizes['sizes\_decoded'] = pd\_sizes['sizes\_codes'].map({1:'small', 2:'medium', 3:

*‹→*'large'})

pd\_sizes

1. : sizes sizes\_codes sizes\_decoded
   1. small 1 small
   2. medium 2 medium

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | large | 3 | large |
| 3 | small | 1 | small |
| 4 | medium | 2 | medium |
| 5 | large | 3 | large |
| 6 | small | 1 | small |
| 7 | medium | 2 | medium |
| 8 | large | 3 | large |

1. :

## Кодирование категорий наборами бинарных значений - one-hot encoding

Каждое уникальное значение признака становится новым отдельным признаком:

**from sklearn.preprocessing import** OneHotEncoder

1. :

ohe = OneHotEncoder()

cat\_enc\_ohe = ohe.fit\_transform(cat\_enc[['c1']])

1. :

cat\_enc.shape

[62]: (157, 1)

1. :

cat\_enc\_ohe.shape

[63]: (157, 27)

1. :

cat\_enc\_ohe

1. : <157x27 sparse matrix of type '<class 'numpy.float64'>'

with 157 stored elements in Compressed Sparse Row format>

1. :

cat\_enc\_ohe.todense()[0:10]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [65]: | matrix([[1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |
|  | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0.], |  |  |  |  |  |
|  | [0., | 0., | 1., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., | 0., |

0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])

1. :

cat\_enc.head(10)

|  |  |  |
| --- | --- | --- |
| [66]: |  | c1 |
|  | 0 | Acura |
|  | 1 | Acura |
|  | 2 | Acura |
|  | 3 | Acura |
|  | 4 | Audi |
|  | 5 | Audi |
|  | 6 | Audi |
|  | 7 | BMW |
|  | 8 | BMW |
|  | 9 | BMW |

1. :

pd.get\_dummies(cat\_enc).head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [67]: | c1\_Acura | c1\_Audi | c1\_BMW | c1\_Buick | c1\_Cadillac | c1\_Chevrolet | c1\_Dodge | \ |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | c1\_Ford | c1\_Honda | | c1\_Hyundai | | … | c1\_Nissan | c1\_Oldsmobile | | c1\_Plymouth | | \ |
| 0 | 0 | 0 | | 0 | | … | 0 | 0 | | 0 | |  |
| 1 | 0 | 0 | | 0 | | … | 0 | 0 | | 0 | |  |
| 2 | 0 | 0 | | 0 | | … | 0 | 0 | | 0 | |  |
| 3 | 0 | 0 | | 0 | | … | 0 | 0 | | 0 | |  |
| 4 | 0 | 0 | | 0 | | … | 0 | 0 | | 0 | |  |
|  | c1\_Pontiac | | c1\_Porsche | | c1\_Saab | c1\_Subaru | | c1\_Toyota | c1\_Volkswagen | | \ | |
| 0 | 0 | | 0 | | 0 | 0 | | 0 | 0 | |  | |
| 1 | 0 | | 0 | | 0 | 0 | | 0 | 0 | |  | |
| 2 | 0 | | 0 | | 0 | 0 | | 0 | 0 | |  | |
| 3 | 0 | | 0 | | 0 | 0 | | 0 | 0 | |  | |
| 4 | 0 | | 0 | | 0 | 0 | | 0 | 0 | |  | |

c1\_Volvo

0 0

1 0

2 0

3 0

4 0

[5 rows x 27 columns]

1. :

pd.get\_dummies(cat\_temp\_data, dummy\_na=**True**).head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [68]: | Manufacturer\_Acura | Manufacturer\_Audi | Manufacturer\_BMW | \ |
| 0 | 1 | 0 | 0 |  |
| 1 | 1 | 0 | 0 |  |
| 2 | 1 | 0 | 0 |  |
| 3 | 1 | 0 | 0 |  |
| 4 | 0 | 1 | 0 |  |

Manufacturer\_Buick Manufacturer\_Cadillac Manufacturer\_Chevrolet \

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 |  | | 0 | | |  | 0 | | |
| 1 | 0 |  | | 0 | | |  | 0 | | |
| 2 | 0 |  | | 0 | | |  | 0 | | |
| 3 | 0 |  | | 0 | | |  | 0 | | |
| 4 | 0 |  | | 0 | | |  | 0 | | |
|  | Manufacturer\_Dodge | Manufacturer\_Ford | | Manufacturer\_Honda | | | \ |  | | |
| 0 | 0 | 0 | | 0 | | |  |  | | |
| 1 | 0 | 0 | | 0 | | |  |  | | |
| 2 | 0 | 0 | | 0 | | |  |  | | |
| 3 | 0 | 0 | | 0 | | |  |  | | |
| 4 | 0 | 0 | | 0 | | |  |  | | |
|  | Manufacturer\_Hyundai | … | Manufacturer\_Oldsmobile | | | Manufacturer\_Plymouth | | | | \ |
| 0 | 0 | … | 0 | | | 0 | | | |  |
| 1 | 0 | … | 0 | | | 0 | | | |  |
| 2 | 0 | … | 0 | | | 0 | | | |  |
| 3 | 0 | … | 0 | | | 0 | | | |  |
| 4 | 0 | … | 0 | | | 0 | | | |  |
|  | Manufacturer\_Pontiac | Manufacturer\_Porsche | | | Manufacturer\_Saab \ | | | |  | |
| 0 | 0 | 0 | | | 0 | | | |  | |
| 1 | 0 | 0 | | | 0 | | | |  | |
| 2 | 0 | 0 | | | 0 | | | |  | |
| 3 | 0 | 0 | | | 0 | | | |  | |
| 4 | 0 | 0 | | | 0 | | | |  | |
|  | Manufacturer\_Subaru | Manufacturer\_Toyota | | | Manufacturer\_Volkswagen | | | | \ | |
| 0 | 0 | 0 | | | 0 | | | |  | |
| 1 | 0 | 0 | | | 0 | | | |  | |
| 2 | 0 | 0 | | | 0 | | | |  | |
| 3 | 0 | 0 | | | 0 | | | |  | |
| 4 | 0 | 0 | | | 0 | | | |  | |
|  | Manufacturer\_Volvo | Manufacturer\_nan | | |  | | | |  | |
| 0 | 0 | 0 | | |  | | | |  | |
| 1 | 0 | 0 | | |  | | | |  | |
| 2 | 0 | 0 | | |  | | | |  | |
| 3 | 0 | 0 | | |  | | | |  | |
| 4 | 0 | 0 | | |  | | | |  | |
| [5 | rows x 28 columns] |  | | |  | | | |  | |

1. :

# Масштабирование данных

Масштабирование предполагает изменение диапазона измерения величины. Применяют MinMax масштабирование и масштабирование данных на основе Z-оценки.

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

## MinMax масштабирование

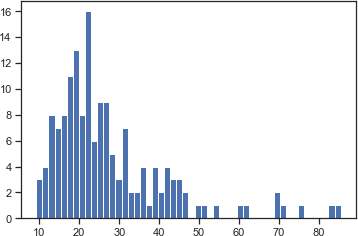
1. :

sc1 = MinMaxScaler()

sc1\_data = sc1.fit\_transform(data[['Price\_in\_thousands']])

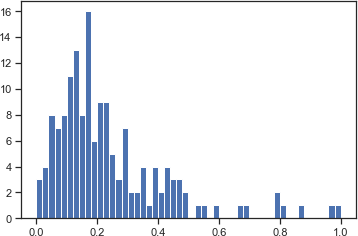
1. :

plt.hist(data['Price\_in\_thousands'], 50) plt.show()



1. :

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



## Масштабирование данных на основе Z-оценки

1. :

sc2 = StandardScaler()

sc2\_data = sc2.fit\_transform(data[['Price\_in\_thousands']])

1. :

plt.hist(sc2\_data, 50) plt.show()

