Московский государственный технический

университет им. Н. Э. Баумана

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

"Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных"

Выполнил:
Анцифров Н. С.
студент группы ИУ5-61Б

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Подпись и дата:

Москва, 2022 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; масштабирование данных.

2. Ячейки Jupyter-ноутбука

1. Выбор и загрузка данных

В качестве датасета будем использовать набор данных, содержащий данные по продажам автомобилей в США. Данный набор доступен по адресу: https://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

Импорт библиотек

Импортируем библиотеки с помощью команды import:

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка данных

Загрузим набор данных:

In [2]:

```
data = pd.read_csv('Car_sales.csv')
```

2. Первичный анализ данных

Выведем первые 5 строк датасета:

In [3]:

```
data.head()
```

Out[3]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheel
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	1
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	1
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	1
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	1
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	1

Определим размер датасета:

In [4]:

```
data.shape
```

Out[4]:

```
(157, 16)
```

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

In [5]:

```
data.dtypes
```

Out[5]:

```
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
Fuel_efficiency        float64
Latest_Launch           object
Power_perf_factor      float64
dtype: object
```

Проверим наличие пропусков:

In [6]:

```
data.isnull().sum()
```

Out[6]:

```
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.

3. Обработка пропусков данных

Удалим колонки, содержащие пустые значения:

In [7]:

```
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[7]:

```
((157, 16), (157, 5))
```

Выведем первые строки датасета на экран:

In [8]:

```
data_new_1
```

Out[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 × 5 columns

Удалим строки, содержащие пустые значения:

In [9]:

```
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[9]:

```
((157, 16), (117, 16))
```

In [10]:

```
data_new_2.head()
```

Out[10]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheel
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	1
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	1
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	1
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	1
5	Audi	A6	18.780	23.555	Passenger	33.95	2.8	200.0	1

Заполним все пропущенные значения нулями:

In [11]:

```
data_new_3 = data.fillna(0)
```

Выведем на экран:

In [12]:

```
data_new_3.head()
```

Out[12]:

	Manufacturer	Model	Sales_in_thousands	__year_resale_value	Vehicle_type	Price_in_thousands	Engine_size	Horsepower	Wheel
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	1
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	1
2	Acura	CL	14.114	18.225	Passenger	0.00	3.2	225.0	1
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	1
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	1

Импьютация данных

Обработка пропусков в числовых данных

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

In [13]:

```
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_nul
l_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:

In [14]:

```
data_num = data[num_cols]
data_num
```

Out[14]:

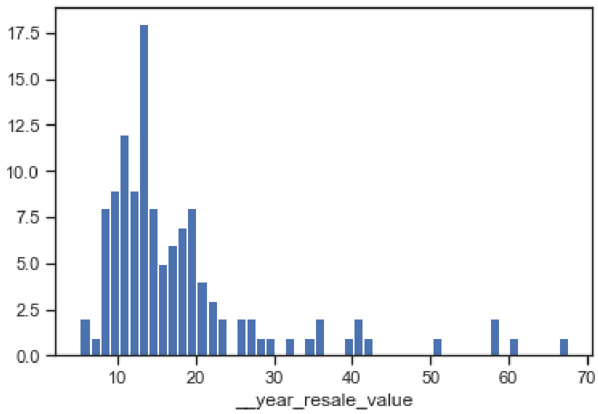
	__year_resale_value	Price_in_thousands	Engine_size	Horsepower	Wheelbase	Width	Length	Curb_weight	Fuel_capacity	Fuel_
0	16.360	21.50	1.8	140.0	101.2	67.3	172.4	2.639	13.2	
1	19.875	28.40	3.2	225.0	108.1	70.3	192.9	3.517	17.2	
2	18.225	NaN	3.2	225.0	106.9	70.6	192.0	3.470	17.2	
3	29.725	42.00	3.5	210.0	114.6	71.4	196.6	3.850	18.0	
4	22.255	23.99	1.8	150.0	102.6	68.2	178.0	2.998	16.4	
...	...	...	...	...	...	...	...	...	...	
152	NaN	24.40	1.9	160.0	100.5	67.6	176.6	3.042	15.8	
153	NaN	27.50	2.4	168.0	104.9	69.3	185.9	3.208	17.9	
154	NaN	28.80	2.4	168.0	104.9	69.3	186.2	3.259	17.9	
155	NaN	45.50	2.3	236.0	104.9	71.5	185.7	3.601	18.5	
156	NaN	36.00	2.9	201.0	109.9	72.1	189.8	3.600	21.1	

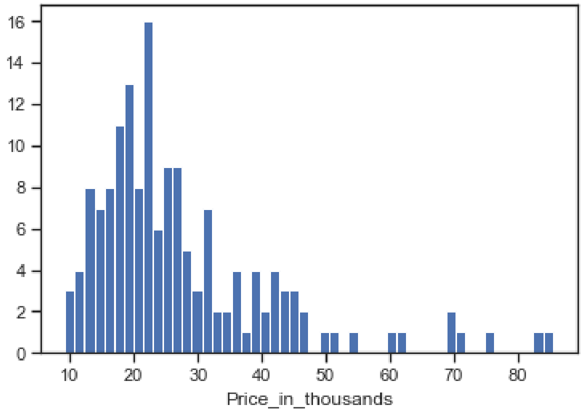
157 rows × 11 columns

Гистограмма по признакам:

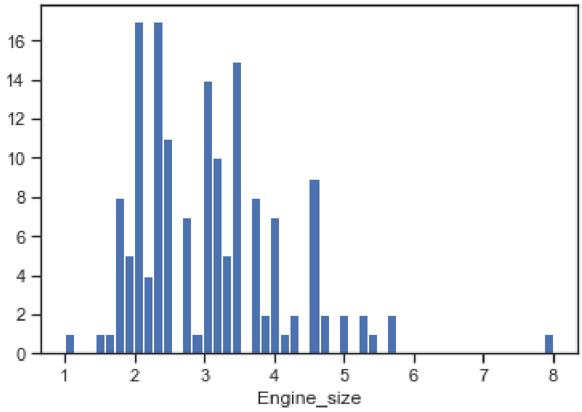
In [15]:

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

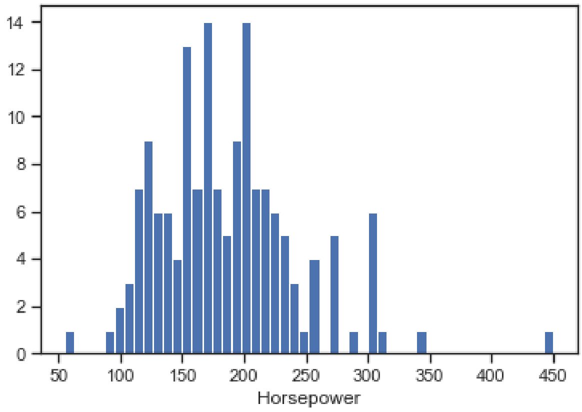




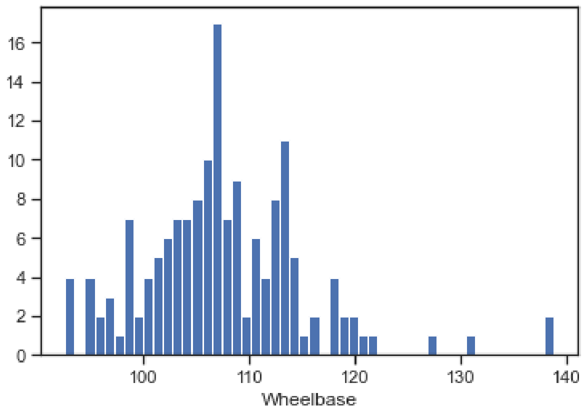
16
14
12
10
8
6
4
2
0
10
20
30
40
50
60
70
80
Price_in_thousands



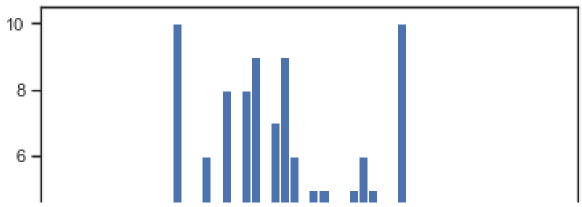
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
Engine_size



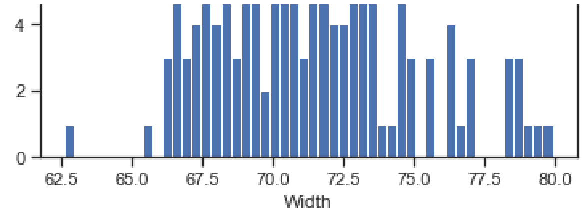
14
12
10
8
6
4
2
0
50
100
150
200
250
300
350
400
450
Horsepower



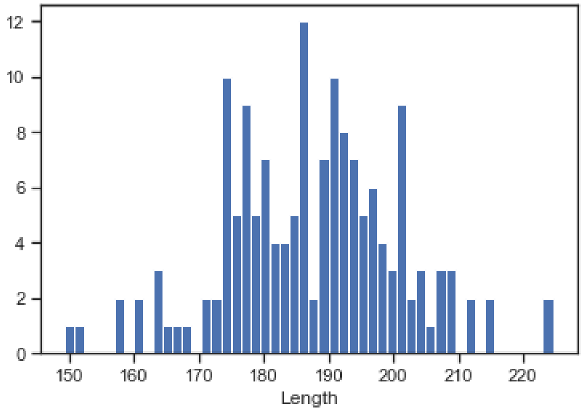
16
14
12
10
8
6
4
2
0
100
110
120
130
140
Wheelbase



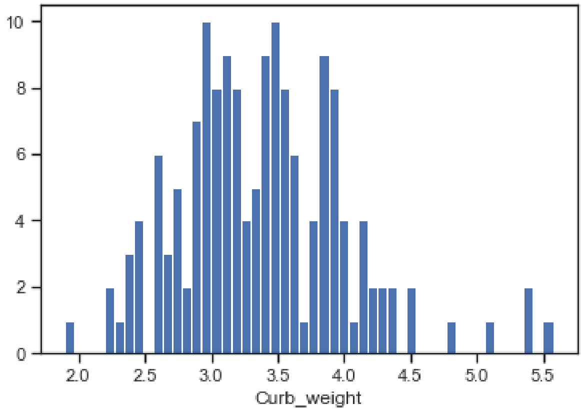
10
8
6



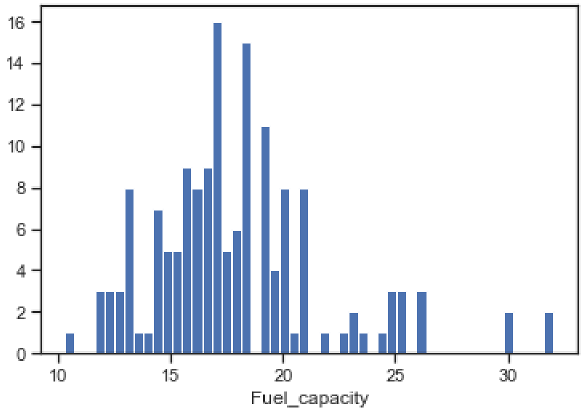
Width



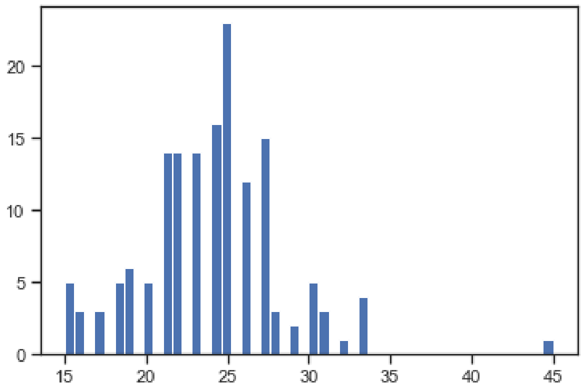
Length



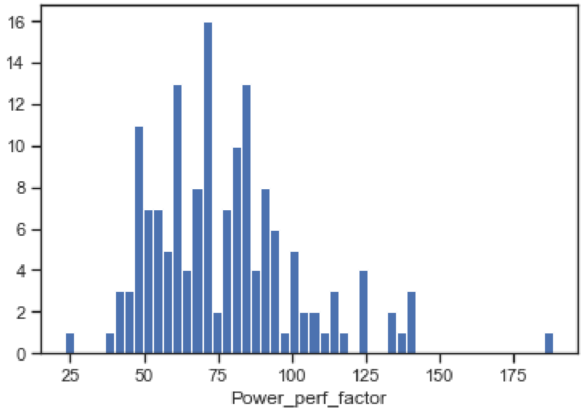
Curb_weight



Fuel_capacity



Fuel_efficiency



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

In [16]:

```
data_num_pit = data_num[['Price_in_thousands']]
```

In [17]:

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

Фильтр для проверки заполнения пустых значений:

In [18]:

```
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_pit)
mask_missing_values_only
```

Out[18]:

```
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],
       [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]])
```

Проведем импьютацию различными показателями центра распределения:

In [19]:

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

In [20]:

```
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]
```

In [21]:

```
strategies[0], test_num_impute(strategies[0])
```

Out[21]:

```
('mean', array([27.39075484, 27.39075484]))
```

In [22]:

```
strategies[1], test_num_impute(strategies[1])
```

Out[22]:

```
('median', array([22.799, 22.799]))
```

In [23]:

```
strategies[2], test_num_impute(strategies[2])
```

Out[23]:

```
('most_frequent', array([12.64, 12.64]))
```

Создадим функцию, позволяющую задавать столбец и вид импьютации:

In [24]:

```
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:

In [25]:

```
data[['__year_resale_value']].describe()
```

Out[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

In [26]:

```
test_num_impute_col(data, '__year_resale_value', strategies[0])
```

Out[26]:

```
('__year_resale_value', 'mean', 36, 18.07297520661157, 18.07297520661157)
```

In [27]:

```
test_num_impute_col(data, '__year_resale_value', strategies[1])
```

Out[27]:

```
('__year_resale_value', 'median', 36, 14.18, 14.18)
```

In [28]:

```
test_num_impute_col(data, '__year_resale_value', strategies[2])
```

Out[28]:

```
('__year_resale_value', 'most_frequent', 36, 7.75, 7.75)
```

Обработка пропусков в категориальных данных

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

In [29]:

```
data_mod = pd.read_csv('Car_sales_mod.csv')
```

Проверим категориальный признак:

In [30]:

```
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_nul
l_count, temp_perc))
```

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

Его и будем использовать:

In [31]:

```
cat_temp_data = data_mod[['Manufacturer']]
cat_temp_data.head()
```

Out[31]:

	Manufacturer
0	Acura
1	Acura
2	Acura
3	Acura
4	Audi

In [32]:

```
cat_temp_data['Manufacturer'].unique()
```

Out[32]:

```
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)
```

In [33]:

```
cat_temp_data[cat_temp_data['Manufacturer'].isnull()].shape
```

Out[33]:

```
(15, 1)
```

Импьютация наиболее частыми значениями:

In [34]:

```
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Out[34]:

```
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)
```

In [35]:

```
np.unique(data_imp2)
```

Out[35]:

```
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)
```

Наблюдаем отсутствие пустых значений.

Импьютация константой:

In [36]:

```
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='???')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[36]:

```
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'],
['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)
```

In [37]:

```
np.unique(data_imp3)
```

Out[37]:

```
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)
```

In [38]:

```
data_imp3[data_imp3==0].size
```

Out[38]:

```
0
```

Значения были заменены на "???".

Преобразование категориальных признаков в числовые

In [39]:

```
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

Out[39]:

	c1
0	Acura
1	Acura
2	Acura
3	Acura
4	Audi
...	...
152	Volvo
153	Volvo
154	Volvo
155	Volvo
156	Volvo

157 rows × 1 columns

4. Кодирование категорий целочисленными значениями

LabelEncoder

In [40]:

```
from sklearn.preprocessing import LabelEncoder
```

In [41]:

```
cat_enc['c1'].unique()
```

Out[41]:

```
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)
```

'Volkswagen', 'Volvo'], dtype=object)

In [42]:

```
le = LabelEncoder()
```

In [43]:

```
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

In [44]:

```
le.classes_
```

Out[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)
```

In [45]:

```
cat_enc_le
```

Out[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])
```

In [46]:

```
np.unique(cat_enc_le)
```

Out[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])
```

In [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])
```

Out[47]:

```
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

In [48]:

```python
from sklearn.preprocessing import OrdinalEncoder
```

In [49]:

```python
data_oe = data_mod[['Manufacturer', 'Model']]
data_oe.head()
```

Out[49]:

	Manufacturer	Model
0	Acura	Integra
1	Acura	TL
2	Acura	CL
3	Acura	RL
4	Audi	A4

In [50]:

```python
imp4 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='???')
data_oe_filled = imp4.fit_transform(data_oe)
data_oe_filled
```

Out[50]:

```
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)
```

In [51]:

```
oe = OrdinalEncoder()
cat_enc_oe = oe.fit_transform(data_oe_filled)
cat_enc_oe
```

Out[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.],
       [ 27.,  24.],
       [ 27., 126.]])
```

Уникальные значения столбца "Производитель":

In [52]:

```
np.unique(cat_enc_oe[:, 0])
```

Out[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":

In [53]:

```
np.unique(cat_enc_oe[:, 1])
```

Out[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:

In [54]:

```
oe.categories_
```

Out[54]:

```
[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)]
```

In [55]:

```
oe.inverse_transform(cat_enc_oe)
```

Out[55]:

```
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', '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', '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)
```

Кодирование шкал порядка

Для кодирования шкал порядка воспользуемся функцией map:

In [56]:

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

In [57]:

```
pd_sizes = pd.DataFrame(data={'sizes':sizes})
pd_sizes
```

Out[57]:

	sizes
0	small
1	medium
2	large
3	small
4	medium
5	large
6	small
7	medium
8	large

In [58]:

```
pd_sizes['sizes_codes'] = pd_sizes['sizes'].map({'small':1, 'medium':2, 'large':3})
pd_sizes
```

Out[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

In [59]:

```
pd_sizes['sizes_decoded'] = pd_sizes['sizes_codes'].map({1:'small', 2:'medium', 3:'large'})
pd_sizes
```

Out[59]:

	sizes	sizes_codes	sizes_decoded
0	small	1	small
1	medium	2	medium
2	large	3	large
3	small	1	small
4	medium	2	medium
5	large	3	large
6	small	1	small

	sizes	sizes_codes	sizes_decoded
6	small	1	small
7	medium	2	medium
8	large	3	large

Кодирование категорий наборами бинарных значений - one-hot encoding

Каждое уникальное значение признака становится новым отдельным признаком:

In [60]:

```
from sklearn.preprocessing import OneHotEncoder
```

In [61]:

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

In [62]:

```
cat_enc.shape
```

Out[62]:

```
(157, 1)
```

In [63]:

```
cat_enc_ohe.shape
```

Out[63]:

```
(157, 27)
```

In [64]:

```
cat_enc_ohe
```

Out[64]:

```
<157x27 sparse matrix of type '<class 'numpy.float64'>'
 with 157 stored elements in Compressed Sparse Row format>
```

In [65]:

```
cat_enc_ohe.todense()[0:10]
```

Out[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.]])
```

In [66]:

```
cat_enc.head(10)
```

Out[66]:

	c1
0	Acura
1	Acura
2	Acura
3	Acura
4	Audi
5	Audi
6	Audi
7	BMW
8	BMW
9	BMW

In [67]:

```
pd.get_dummies(cat_enc).head()
```

Out[67]:

	c1_Acura	c1_Audi	c1_BMW	c1_Buick	c1_Cadillac	c1_Chevrolet	c1_Dodge	c1_Ford	c1_Honda	c1_Hyundai	...	c1_Nissan	c1_(
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	1	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	
4	0	1	0	0	0	0	0	0	0	0	...	0	

5 rows × 27 columns

In [68]:

```
pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

Out[68]:

	Manufacturer_Acura	Manufacturer_Audi	Manufacturer_BMW	Manufacturer_Buick	Manufacturer_Cadillac	Manufacturer_Chevrolet	Ma
0	1	0	0	0	0	0	
1	1	0	0	0	0	0	
2	1	0	0	0	0	0	
3	1	0	0	0	0	0	
4	0	1	0	0	0	0	

5 rows × 28 columns

5. Масштабирование данных

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

In [69]:

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

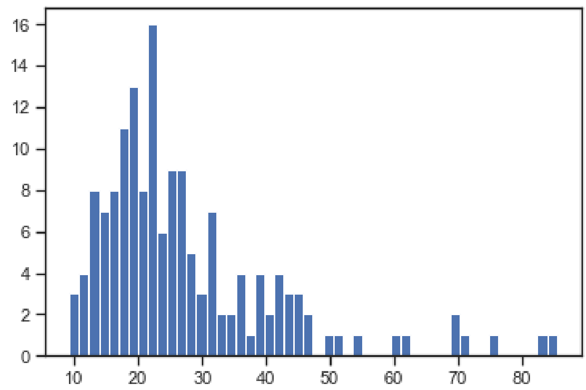
MinMax масштабирование

In [70]:

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

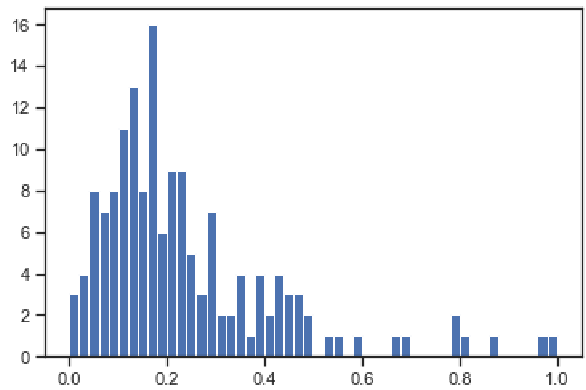
In [71]:

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



In [72]:

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



Масштабирование данных на основе Z-оценки

In [73]:

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

In [74]:

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