

Лабораторная работа №3
по дисциплине
«Методы машинного обучения»
на тему
«Обработка пропусков в данных,
кодирование категориальных
признаков, масштабирование
данных»

Выполнил:
студент группы ИУ5-21М
Фомин В.Ю.

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")
```

In [2]:

```
# Будем использовать только обучающую выборку
data = pd.read_csv('Building_Permits.csv', sep=",")
```

```
c:\users\innap\miniconda3\lib\site-packages\IPython\core\interactiveshell.
py:3063: DtypeWarning: Columns (22,32) have mixed types. Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

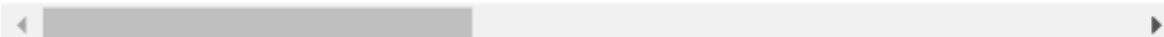
In [3]:

```
data.head()
```

Out[3]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Stre Suff
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	/
3	201611072166	8	etc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	/
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	

5 rows × 43 columns



In [4]:

```
data.shape
```

Out[4]:

(198900, 43)

In [5]:

```
data.dtypes
```

Out[5]:

Permit Number	object
Permit Type	int64
Permit Type Definition	object
Permit Creation Date	object
Block	object
Lot	object
Street Number	int64
Street Number Suffix	object
Street Name	object
Street Suffix	object
Unit	float64
Unit Suffix	object
Description	object
Current Status	object
Current Status Date	object
Filed Date	object
Issued Date	object
Completed Date	object
First Construction Document Date	object
Structural Notification	object
Number of Existing Stories	float64
Number of Proposed Stories	float64
Voluntary Soft-Story Retrofit	object
Fire Only Permit	object
Permit Expiration Date	object
Estimated Cost	float64
Revised Cost	float64
Existing Use	object
Existing Units	float64
Proposed Use	object
Proposed Units	float64
Plansets	float64
TIDF Compliance	object
Existing Construction Type	float64
Existing Construction Type Description	object
Proposed Construction Type	float64
Proposed Construction Type Description	object
Site Permit	object
Supervisor District	float64
Neighborhoods - Analysis Boundaries	object
Zipcode	float64
Location	object
Record ID	int64
dtype:	object

In [6]:

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[6]:

Permit Number	0
Permit Type	0
Permit Type Definition	0
Permit Creation Date	0
Block	0
Lot	0
Street Number	0
Street Number Suffix	196684
Street Name	0
Street Suffix	2768
Unit	169421
Unit Suffix	196939
Description	290
Current Status	0
Current Status Date	0
Filed Date	0
Issued Date	14940
Completed Date	101709
First Construction Document Date	14946
Structural Notification	191978
Number of Existing Stories	42784
Number of Proposed Stories	42868
Voluntary Soft-Story Retrofit	198865
Fire Only Permit	180073
Permit Expiration Date	51880
Estimated Cost	38066
Revised Cost	6066
Existing Use	41114
Existing Units	51538
Proposed Use	42439
Proposed Units	50911
Plansets	37309
TIDF Compliance	198898
Existing Construction Type	43366
Existing Construction Type Description	43366
Proposed Construction Type	43162
Proposed Construction Type Description	43162
Site Permit	193541
Supervisor District	1717
Neighborhoods - Analysis Boundaries	1725
Zipcode	1716
Location	1700
Record ID	0
dtype: int64	

In [7]:

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

Всего строк: 198900

1. Обработка пропусков в данных

1.1. Простые стратегии - удаление или заполнение нулями

In [8]:

```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[8]:

```
((198900, 43), (198900, 12))
```

In [9]:

```
# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[9]:

```
((198900, 43), (0, 43))
```

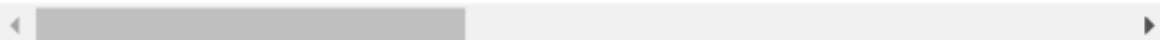
In [10]:

```
data.head()
```

Out[10]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Stre Suff
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	/
3	201611072166	8	etc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	/
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	

5 rows × 43 columns



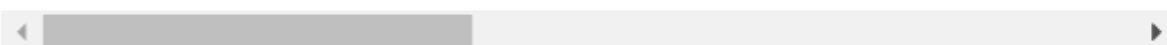
In [11]:

```
# Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том числе категориальны
е колонки
data_new_3 = data.fillna(0)
data_new_3.head()
```

Out[11]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Stre Suff
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	0	Ellis	
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	0	Geary	
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	0	Pacific	/
3	201611072166	8	etc alterations permit	11/07/2016	0156	011	1230	0	Pacific	/
4	201611283529	6	demolitions	11/28/2016	0342	001	950	0	Market	

5 rows × 43 columns



1.2. "Внедрение значений" - импьютация (imputation)

1.2.1. Обработка пропусков в числовых данных

In [12]:

```
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count > 0 and (dt == 'float64' or dt == 'int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(c
ol, dt, temp_null_count, temp_perc))
```

Колонка Unit. Тип данных float64. Количество пустых значений 169421, 85.18%.

Колонка Number of Existing Stories. Тип данных float64. Количество пустых значений 42784, 21.51%.

Колонка Number of Proposed Stories. Тип данных float64. Количество пустых значений 42868, 21.55%.

Колонка Estimated Cost. Тип данных float64. Количество пустых значений 38066, 19.14%.

Колонка Revised Cost. Тип данных float64. Количество пустых значений 6066, 3.05%.

Колонка Existing Units. Тип данных float64. Количество пустых значений 51538, 25.91%.

Колонка Proposed Units. Тип данных float64. Количество пустых значений 50911, 25.6%.

Колонка Plansets. Тип данных float64. Количество пустых значений 37309, 18.76%.

Колонка Existing Construction Type. Тип данных float64. Количество пустых значений 43366, 21.8%.

Колонка Proposed Construction Type. Тип данных float64. Количество пустых значений 43162, 21.7%.

Колонка Supervisor District. Тип данных float64. Количество пустых значений 1717, 0.86%.

Колонка Zipcode. Тип данных float64. Количество пустых значений 1716, 0.86%.

In [13]:

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


Out[13]:

	Unit	Number of Existing Stories	Number of Proposed Stories	Estimated Cost	Revised Cost	Existing Units	Proposed Units	Plansets	Con
0	NaN	6.0	NaN	4000.00	4000.00	143.0	NaN	2.0	
1	0.0	7.0	NaN	1.00	500.00	NaN	NaN	2.0	
2	NaN	6.0	6.0	20000.00	NaN	39.0	39.0	2.0	
3	0.0	2.0	2.0	2000.00	2000.00	1.0	1.0	2.0	
4	NaN	3.0	NaN	100000.00	100000.00	NaN	NaN	2.0	
5	NaN	5.0	5.0	4000.00	4000.00	326.0	326.0	2.0	
6	0.0	3.0	3.0	12000.00	12000.00	5.0	5.0	0.0	
7	NaN	NaN	NaN	NaN	0.00	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	0.00	NaN	NaN	NaN	
10	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
11	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
12	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
13	NaN	2.0	2.0	30000.00	0.00	1.0	1.0	2.0	
14	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
15	301.0	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
16	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
17	0.0	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
18	1.0	4.0	4.0	75000.00	0.00	6.0	6.0	2.0	
19	0.0	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
20	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
21	NaN	2.0	2.0	100000.00	0.00	2.0	2.0	2.0	
22	NaN	3.0	3.0	100000.00	NaN	6.0	6.0	2.0	
23	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
24	NaN	4.0	4.0	64650.00	64650.00	9.0	9.0	0.0	
25	NaN	2.0	2.0	7000.00	7000.00	1.0	1.0	0.0	
26	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
27	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
28	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
29	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
...	
198870	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198871	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198872	NaN	2.0	2.0	10000.00	0.00	NaN	NaN	2.0	
198873	0.0	2.0	2.0	1.00	1.00	1.0	1.0	0.0	

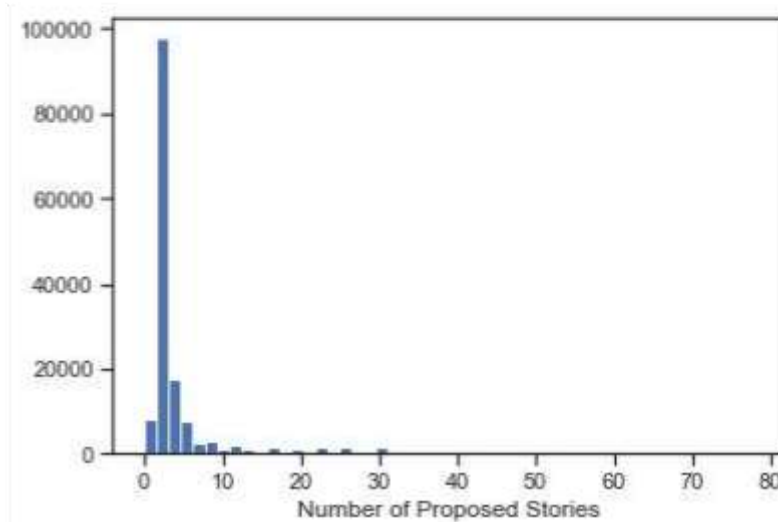
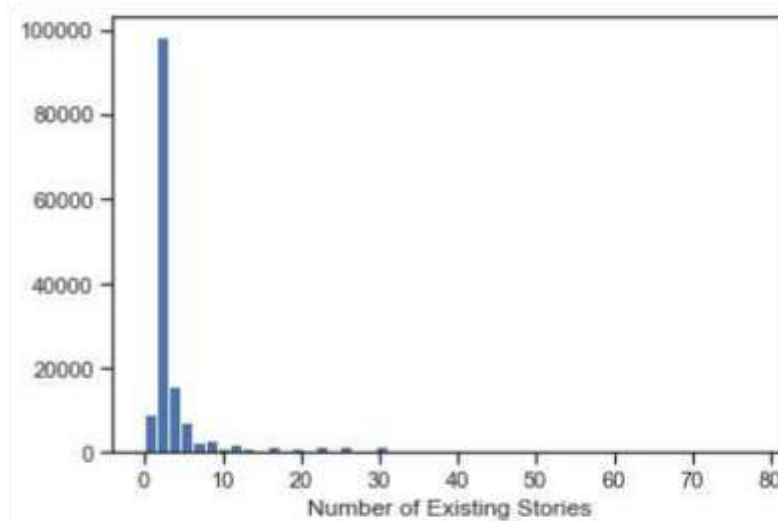
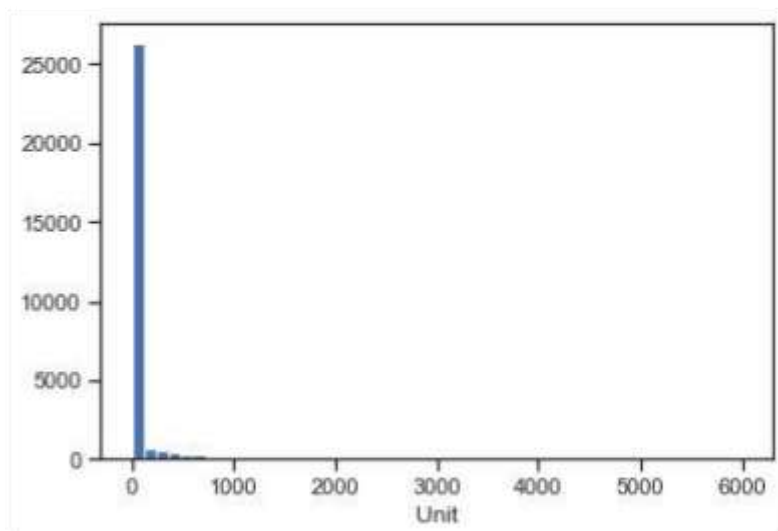
	Unit	Number of Existing Stories	Number of Proposed Stories	Estimated Cost	Revised Cost	Existing Units	Proposed Units	Plansets	Con
198874	0.0	3.0	3.0	10000.00	15000.00	3.0	3.0	0.0	
198875	NaN	NaN	5.0	1.00	0.00	NaN	29.0	2.0	
198876	NaN	NaN	NaN	NaN	0.00	NaN	NaN	NaN	
198877	NaN	2.0	2.0	1000.00	1000.00	1.0	1.0	0.0	
198878	0.0	2.0	2.0	20000.00	0.00	4.0	4.0	2.0	
198879	NaN	2.0	2.0	18453.12	18453.12	1.0	1.0	0.0	
198880	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198881	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198882	NaN	2.0	2.0	1.00	1.00	1.0	1.0	0.0	
198883	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198884	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198885	NaN	2.0	2.0	5000.00	0.00	1.0	1.0	2.0	
198886	NaN	3.0	3.0	65000.00	65000.00	6.0	6.0	2.0	
198887	NaN	4.0	4.0	30000.00	30000.00	191.0	191.0	2.0	
198888	NaN	9.0	9.0	18000.00	30000.00	78.0	78.0	0.0	
198889	NaN	1.0	1.0	20000.00	20000.00	1.0	1.0	0.0	
198890	NaN	6.0	6.0	750000.00	750000.00	NaN	NaN	2.0	
198891	NaN	2.0	2.0	1.00	1.00	0.0	0.0	2.0	
198892	NaN	3.0	3.0	55000.00	38000.00	5.0	5.0	2.0	
198893	NaN	3.0	3.0	2800.00	2800.00	3.0	3.0	0.0	
198894	NaN	2.0	2.0	7400.00	7400.00	2.0	2.0	0.0	
198895	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198896	NaN	4.0	4.0	5000.00	5000.00	4.0	4.0	2.0	
198897	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198898	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	
198899	NaN	NaN	NaN	NaN	1.00	NaN	NaN	NaN	

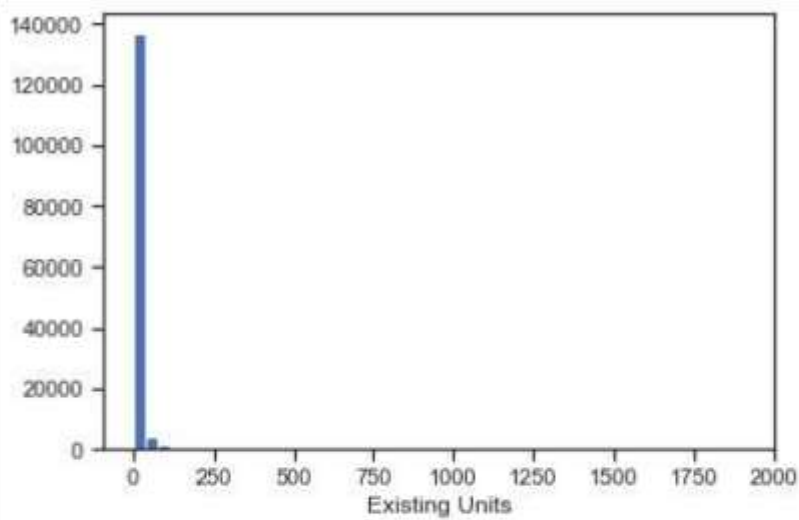
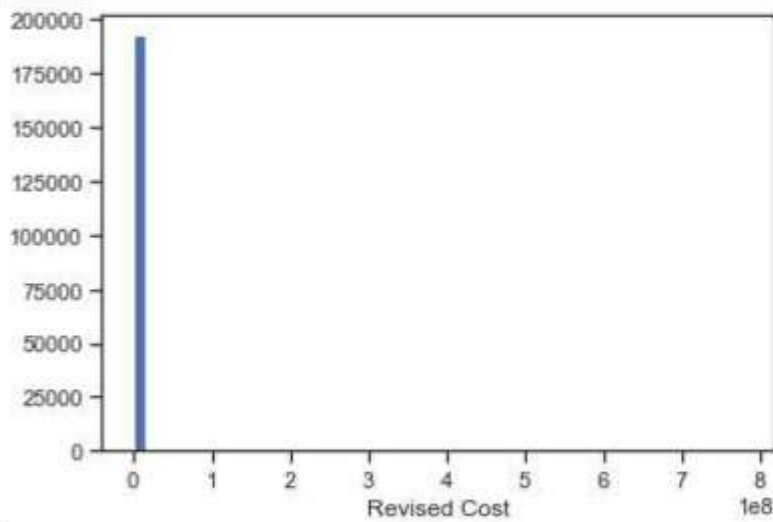
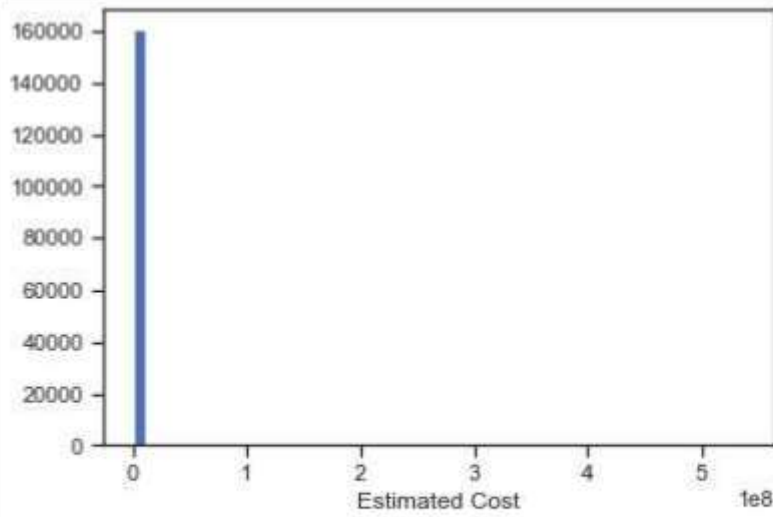
198900 rows × 12 columns

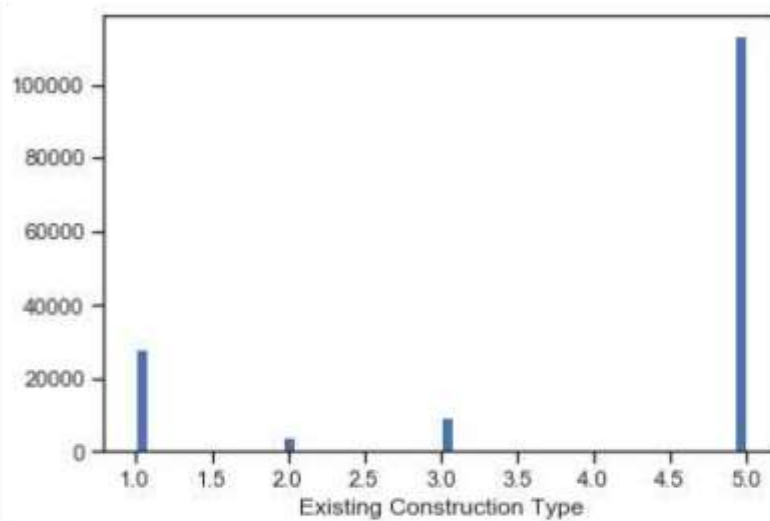
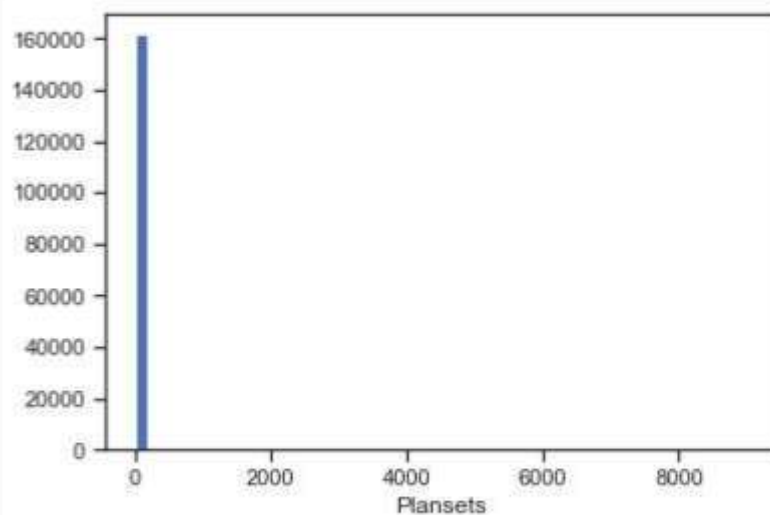
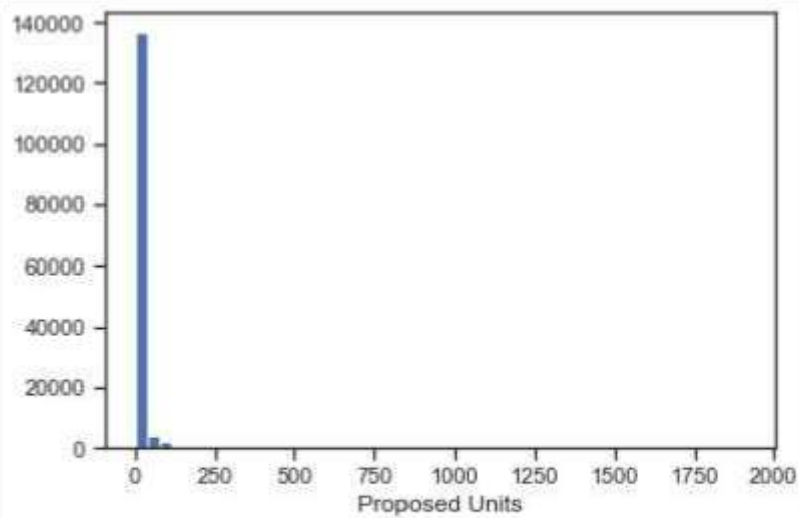
In [14]:

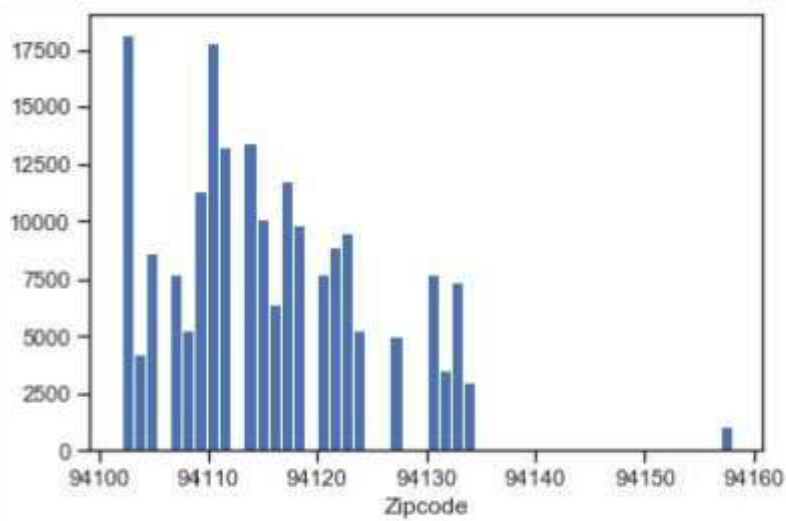
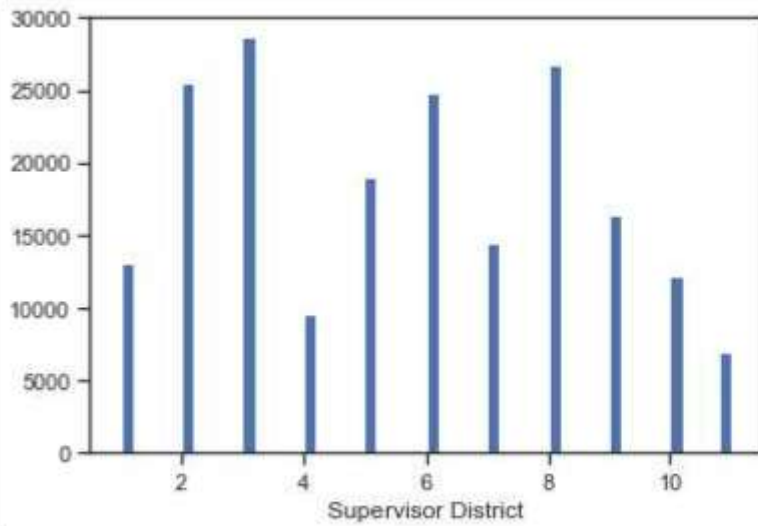
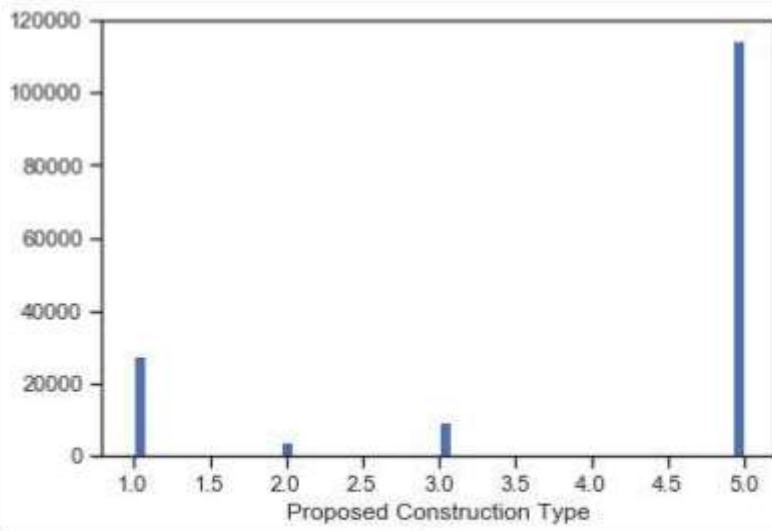
```
# Гистограмма по признакам  
for col in data_num:  
    plt.hist(data[col], 50)  
    plt.xlabel(col)  
    plt.show()
```

```
c:\users\innap\miniconda3\lib\site-packages\numpy\lib\histograms.py:824: R
untimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
c:\users\innap\miniconda3\lib\site-packages\numpy\lib\histograms.py:825: R
untimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```









In [15]:

```
# Фильтр по пустым значениям поля Estimated Cost  
data[data['Estimated Cost'].isnull()]
```


Out[15]:

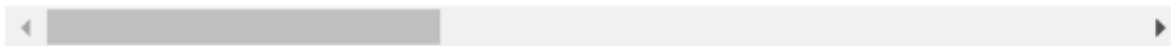
	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
7	M803667	8	otc alterations permit	06/30/2017	4789	014	1465	NaN	R
8	M804227	8	otc alterations permit	07/05/2017	1212	054	2094	NaN	
9	M804767	8	otc alterations permit	07/06/2017	1259	016	89	NaN	/
10	M805287	8	otc alterations permit	07/06/2017	3541	001	106	NaN	Sai
11	M805907	8	otc alterations permit	07/07/2017	0829	021	675	NaN	
12	M806447	8	otc alterations permit	07/10/2017	6537	023	4082	NaN	
14	M813729	8	otc alterations permit	07/26/2017	1049	027	2761	NaN	
15	M813907	8	otc alterations permit	07/27/2017	0243	043	840	NaN	F
16	M813967	8	otc alterations permit	07/27/2017	0268	008	220	NaN	Montgr
17	M814148	8	otc alterations permit	07/27/2017	3621	097	3707	NaN	
19	M814368	8	otc alterations permit	07/28/2017	2659	001	263	NaN	C
20	M814967	8	otc alterations permit	07/31/2017	1176	068	1624	NaN	f
23	M816927	8	otc alterations permit	08/07/2017	0941	012	2824	NaN	f
26	M820728	8	otc alterations permit	08/18/2017	1449	006	331	NaN	
27	M821207	8	otc alterations permit	08/21/2017	1193	024	120	NaN	St
28	M821268	8	otc alterations permit	08/21/2017	1851	012	1512	NaN	
29	M821847	8	otc alterations permit	08/22/2017	0326	001	165	NaN	F

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
30	M522307	8	otc alterations permit	08/23/2017	0182	004	1441	NaN	
31	M522487	8	otc alterations permit	08/23/2017	3580	017	3830	NaN	
32	M522707	8	otc alterations permit	08/23/2017	3610	041	563	NaN	
34	M523847	8	otc alterations permit	08/25/2017	3545	018A	1810	NaN	
35	M524007	8	otc alterations permit	08/25/2017	0749	006A	1445	NaN	
36	M524127	8	otc alterations permit	08/28/2017	6551	023	1383	NaN	C
37	M524927	8	otc alterations permit	08/29/2017	6597	024	681	NaN	Sar
38	M525267	8	otc alterations permit	08/29/2017	1572	017	606	NaN	
39	M528787	8	otc alterations permit	09/07/2017	0177	022	655	NaN	F
40	M529567	8	otc alterations permit	09/08/2017	0490	002	3325	NaN	S
41	M532207	8	otc alterations permit	09/15/2017	1609	021F	700	NaN	
43	M533229	8	otc alterations permit	09/19/2017	0605	028	2379	NaN	Ja
44	M534107	8	otc alterations permit	09/21/2017	7151	014	247	NaN	Cl
...	
198822	M592988	8	otc alterations permit	02/23/2018	5811	042	815	NaN	Mi
198824	M593007	8	otc alterations permit	02/23/2018	5880	007	36	NaN	
198830	M593027	8	otc alterations permit	02/23/2018	6474	014	680	NaN	Brun
198831	M593047	8	otc alterations permit	02/23/2018	4343	029	2587	NaN	

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
198833	M593067	8	otc alterations permit	02/23/2018	1523	031	462	NaN	
198837	M593087	8	otc alterations permit	02/23/2018	1144	001B	222	NaN	Sti
198840	M593107	8	otc alterations permit	02/23/2018	0576	011	1856	NaN	F
198841	M593127	8	otc alterations permit	02/23/2018	0652	032	2215	NaN	Cali
198842	M593147	8	otc alterations permit	02/23/2018	0191	018	1242	NaN	N
198843	M593167	8	otc alterations permit	02/23/2018	0852	042	221	NaN	
198847	M593187	8	otc alterations permit	02/23/2018	1205	035	403	NaN	Bro
198850	M593207	8	otc alterations permit	02/23/2018	3844	015	143	NaN	Sar
198858	M593227	8	otc alterations permit	02/23/2018	0652	005	2007	NaN	Buck
198859	M593247	8	otc alterations permit	02/23/2018	3504	030	1699	NaN	N
198861	M593267	8	otc alterations permit	02/23/2018	0309	009	156	NaN	I
198862	201802232161	8	otc alterations permit	02/23/2018	0670	004	1233	NaN	
198865	M593307	8	otc alterations permit	02/23/2018	1070	001A	2750	NaN	I
198868	M593327	8	otc alterations permit	02/23/2018	5744	021	311	NaN	Cre
198869	M593328	8	otc alterations permit	02/23/2018	0025	024	884	NaN	North
198870	M593328	8	otc alterations permit	02/23/2018	0025	024	888	NaN	North
198871	M593347	8	otc alterations permit	02/23/2018	1253	001	701	NaN	Cl
198876	M593367	8	otc alterations permit	02/23/2018	1213	013	500	NaN	Sti

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
198880	M893387	8	otc alterations permit	02/23/2018	1841	025D	1442	NaN	Fu
198881	M893407	8	otc alterations permit	02/23/2018	0018	004	2300	NaN	Stc
198883	M893427	8	otc alterations permit	02/23/2018	0315	021	377	NaN	i
198884	M893447	8	otc alterations permit	02/23/2018	3504	030	1699	NaN	N
198895	M862628	8	otc alterations permit	12/05/2017	0113	017A	1228	NaN	Montgr
198897	M863507	8	otc alterations permit	12/06/2017	4318	019	1568	NaN	In
198898	M863747	8	otc alterations permit	12/06/2017	0298	029	795	NaN	:
198899	M864287	8	otc alterations permit	12/07/2017	0160	006	838	NaN	F

38066 rows × 43 columns



In [16]:

```
# Запоминаем индексы строк с пустыми значениями
flt_index = data[data['Estimated Cost'].isnull()].index
flt_index
```

Out[16]:

```
Int64Index([    7,    8,    9,   10,   11,   12,   14,    1
 5,
          16,   17,
          ...
198871, 198876, 198880, 198881, 198883, 198884, 198895, 19889
7,
          198898, 198899],
          dtype='int64', length=38066)
```

In [17]:

```
# Проверяем что выводятся нужные строки  
data[data.index.isin(flt_index)]
```

Out[17]:

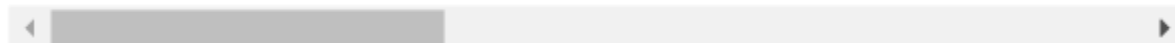
	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
7	M503667	8	otc alterations permit	06/30/2017	4789	014	1465	NaN	R
8	M504227	8	otc alterations permit	07/05/2017	1212	054	2094	NaN	
9	M504767	8	otc alterations permit	07/06/2017	1259	016	89	NaN	/
10	M505287	8	otc alterations permit	07/06/2017	3541	001	106	NaN	Sai
11	M505907	8	otc alterations permit	07/07/2017	0829	021	675	NaN	
12	M506447	8	otc alterations permit	07/10/2017	6537	023	4082	NaN	
14	M513729	8	otc alterations permit	07/26/2017	1049	027	2761	NaN	
15	M513907	8	otc alterations permit	07/27/2017	0243	043	840	NaN	F
16	M513967	8	otc alterations permit	07/27/2017	0268	008	220	NaN	Montgr
17	M514148	8	otc alterations permit	07/27/2017	3621	097	3707	NaN	
19	M514368	8	otc alterations permit	07/28/2017	2659	001	263	NaN	C
20	M514967	8	otc alterations permit	07/31/2017	1176	068	1624	NaN	f
23	M516927	8	otc alterations permit	08/07/2017	0941	012	2824	NaN	f
26	M520728	8	otc alterations permit	08/18/2017	1449	006	331	NaN	
27	M521207	8	otc alterations permit	08/21/2017	1193	024	120	NaN	Str
28	M521268	8	otc alterations permit	08/21/2017	1851	012	1512	NaN	
29	M521847	8	otc alterations permit	08/22/2017	0326	001	165	NaN	F

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
30	M522307	8	otc alterations permit	08/23/2017	0182	004	1441	NaN	
31	M522487	8	otc alterations permit	08/23/2017	3580	017	3830	NaN	
32	M522707	8	otc alterations permit	08/23/2017	3610	041	563	NaN	
34	M523847	8	otc alterations permit	08/25/2017	3545	018A	1810	NaN	
35	M524007	8	otc alterations permit	08/25/2017	0749	006A	1445	NaN	
36	M524127	8	otc alterations permit	08/28/2017	6551	023	1383	NaN	C
37	M524927	8	otc alterations permit	08/29/2017	6597	024	681	NaN	Sar
38	M525267	8	otc alterations permit	08/29/2017	1572	017	606	NaN	
39	M528787	8	otc alterations permit	09/07/2017	0177	022	655	NaN	F
40	M529567	8	otc alterations permit	09/08/2017	0490	002	3325	NaN	S
41	M532207	8	otc alterations permit	09/15/2017	1609	021F	700	NaN	
43	M533229	8	otc alterations permit	09/19/2017	0605	028	2379	NaN	Ja
44	M534107	8	otc alterations permit	09/21/2017	7151	014	247	NaN	Cl
...	
198822	M592988	8	otc alterations permit	02/23/2018	5811	042	815	NaN	Mi
198824	M593007	8	otc alterations permit	02/23/2018	5880	007	36	NaN	
198830	M593027	8	otc alterations permit	02/23/2018	6474	014	680	NaN	Brun
198831	M593047	8	otc alterations permit	02/23/2018	4343	029	2587	NaN	

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
198833	M593067	8	otc alterations permit	02/23/2018	1523	031	462	NaN	
198837	M593087	8	otc alterations permit	02/23/2018	1144	001B	222	NaN	Sta
198840	M593107	8	otc alterations permit	02/23/2018	0576	011	1856	NaN	F
198841	M593127	8	otc alterations permit	02/23/2018	0652	032	2215	NaN	Cali
198842	M593147	8	otc alterations permit	02/23/2018	0191	018	1242	NaN	N
198843	M593167	8	otc alterations permit	02/23/2018	0852	042	221	NaN	
198847	M593187	8	otc alterations permit	02/23/2018	1205	035	403	NaN	Bro
198850	M593207	8	otc alterations permit	02/23/2018	3644	015	143	NaN	Sar
198858	M593227	8	otc alterations permit	02/23/2018	0652	005	2007	NaN	Buck
198859	M593247	8	otc alterations permit	02/23/2018	3504	030	1699	NaN	N
198861	M593267	8	otc alterations permit	02/23/2018	0309	009	156	NaN	I
198862	201802232161	8	otc alterations permit	02/23/2018	0670	004	1233	NaN	
198865	M593307	8	otc alterations permit	02/23/2018	1070	001A	2750	NaN	I
198868	M593327	8	otc alterations permit	02/23/2018	5744	021	311	NaN	Cre
198869	M593328	8	otc alterations permit	02/23/2018	0025	024	884	NaN	North
198870	M593328	8	otc alterations permit	02/23/2018	0025	024	888	NaN	North
198871	M593347	8	otc alterations permit	02/23/2018	1253	001	701	NaN	Cl
198876	M593367	8	otc alterations permit	02/23/2018	1213	013	500	NaN	Sta

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	
198880	M593387	8	otc alterations permit	02/23/2018	1841	025D	1442	NaN	Fu
198881	M593407	8	otc alterations permit	02/23/2018	0018	004	2300	NaN	Stc
198883	M593427	8	otc alterations permit	02/23/2018	0315	021	377	NaN	
198884	M593447	8	otc alterations permit	02/23/2018	3504	030	1699	NaN	N
198895	M562628	8	otc alterations permit	12/05/2017	0113	017A	1228	NaN	Montgr
198897	M563507	8	otc alterations permit	12/06/2017	4318	019	1568	NaN	In
198898	M563747	8	otc alterations permit	12/06/2017	0298	029	795	NaN	
198899	M564287	8	otc alterations permit	12/07/2017	0160	006	838	NaN	F

38066 rows × 43 columns



In [18]:

```
# фильтр по колонке
```

```
data_num[data_num.index.isin(flt_index)]['Estimated Cost']
```

Out[18]:

7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
14	NaN
15	NaN
16	NaN
17	NaN
19	NaN
20	NaN
23	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	NaN
31	NaN
32	NaN
34	NaN
35	NaN
36	NaN
37	NaN
38	NaN
39	NaN
40	NaN
41	NaN
43	NaN
44	NaN
	..
198822	NaN
198824	NaN
198830	NaN
198831	NaN
198833	NaN
198837	NaN
198840	NaN
198841	NaN
198842	NaN
198843	NaN
198847	NaN
198850	NaN
198858	NaN
198859	NaN
198861	NaN
198862	NaN
198865	NaN
198868	NaN
198869	NaN
198870	NaN
198871	NaN
198876	NaN
198880	NaN
198881	NaN
198883	NaN
198884	NaN
198895	NaN
198897	NaN

```
198898    NaN
198899    NaN
Name: Estimated Cost, Length: 38066, dtype: float64
```

In [19]:

```
data_num_EstCost = data_num[['Estimated Cost']]
data_num_EstCost.head()
```

Out[19]:

	Estimated Cost
0	4000.0
1	1.0
2	20000.0
3	2000.0
4	100000.0

In [20]:

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

In [21]:

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

Out[21]:

```
array([[False],
       [False],
       [False],
       ...,
       [ True],
       [ True],
       [ True]])
```

In [22]:

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

In [23]:

```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_EstCost)
    return data_num_imp[mask_missing_values_only]
```

In [24]:

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

Out[24]:

```
('mean',  
 array([168955.44329682, 168955.44329682, 168955.44329682, ...,  
        168955.44329682, 168955.44329682, 168955.44329682]))
```

In [25]:

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

Out[25]:

```
('median', array([11000., 11000., 11000., ..., 11000., 11000., 11000.]))
```

In [26]:

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

Out[26]:

```
('most_frequent', array([1., 1., 1., ..., 1., 1., 1.]))
```

In [27]:

```
# Более сложная функция, которая позволяет задавать колонку и вид импутации  
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]
```

In [28]:

```
data[['Revised Cost']].describe()
```

Out[28]:

	Revised Cost
count	1.928340e+05
mean	1.328562e+05
std	3.584903e+06
min	0.000000e+00
25%	1.000000e+00
50%	7.000000e+03
75%	2.870750e+04
max	7.805000e+08

In [29]:

```
test_num_impute_col(data, 'Revised Cost', strategies[0])
```

Out[29]:

```
('Revised Cost', 'mean', 6066, 132856.1864917494, 132856.1864917494)
```

In [30]:

```
test_num_impute_col(data, 'Revised Cost', strategies[1])
```

Out[30]:

```
('Revised Cost', 'median', 6066, 7000.0, 7000.0)
```

In [31]:

```
test_num_impute_col(data, 'Revised Cost', strategies[2])
```

Out[31]:

```
('Revised Cost', 'most_frequent', 6066, 1.0, 1.0)
```

2. Обработка пропусков в категориальных данных

In [32]:

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

In [33]:

```
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count > 0 and (dt == 'object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(c
ol, dt, temp_null_count, temp_perc))
```

Колонка manufacturer. Тип данных object. Количество пустых значений 22764, 11.44%.

Колонка model. Тип данных object. Количество пустых значений 7989, 4.02%.

Колонка condition. Тип данных object. Количество пустых значений 231934, 116.61%.

Колонка cylinders. Тип данных object. Количество пустых значений 199683, 100.39%.

Колонка fuel. Тип данных object. Количество пустых значений 3985, 2.0%.

Колонка title_status. Тип данных object. Количество пустых значений 3062, 1.54%.

Колонка transmission. Тип данных object. Количество пустых значений 3719, 1.87%.

Колонка vin. Тип данных object. Количество пустых значений 207425, 104.29%.

Колонка drive. Тип данных object. Количество пустых значений 144143, 72.47%.

Колонка size. Тип данных object. Количество пустых значений 342003, 171.95%.

Колонка type. Тип данных object. Количество пустых значений 141531, 71.16%.

Колонка paint_color. Тип данных object. Количество пустых значений 164706, 82.81%.

Колонка image_url. Тип данных object. Количество пустых значений 14, 0.01%.

Колонка description. Тип данных object. Количество пустых значений 16, 0.01%.

In [34]:

```
cat_temp_data = data[['model']]
cat_temp_data.head()
```

Out[34]:

	model
0	golf r
1	f-150
2	sierra 1500
3	f-150
4	f-450

In [35]:

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

Out[35]:

```
array(['golf r', 'f-150', 'sierra 1500', ..., 'Camaro 2-door coupe',  
      'Isuzu VehiCROSS', 'peterbilt 378'], dtype=object)
```

In [36]:

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

Out[36]:

```
(7989, 1)
```

In [37]:

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

Out[37]:

```
array(['golf r',  
      'f-150',  
      'sierra 1500',  
      ...,  
      'cherokee',  
      'Porsche Macan GTS',  
      'f-150'], dtype=object)
```

In [38]:

```
# Пустые значения отсутствуют  
np.unique(data_imp2)
```

Out[38]:

```
array(['#350', '#4', '$1500 DOWN PAYMENT', ..., 'É\x9bÌ\x83fini MS-9',  
      'â\x80\x98 Astro',  
      'ð\x9d\x97\x9fð\x9d\x97¶ð\x9d\x97»ð\x9d\x97°ð\x9d\x97¼ð\x9d\x97¹ð\x9d\x97»  
      ð\x9d\x97\xa0ð\x9d\x97\x9eð\x9d\x97! ð\x9d\x9f".ð\x9d\x9f³ð\x9d\x97\x9f  
      ð\x9d\x97\x94ð\x9d\x97ªð\x9d\x97\x97'],  
      dtype=object)
```


In [39]:

```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='!!!')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[39]:

```
array([[ 'golf r'],
       [ 'f-150'],
       [ 'sierra 1500'],
       ...,
       [ 'cherokee'],
       [ 'Porsche Macan GTS'],
       [ '!!!']], dtype=object)
```

In [40]:

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

Out[40]:

```
array(['!!!', '#350', '#4', ..., 'É\x9bİ\x83fini MS-9',
      'â\x80\x9898 Astro',
      'ð\x9d\x97\x9fð\x9d\x97¶ð\x9d\x97»ð\x9d\x97°ð\x9d\x97¼ð\x9d\x97¹ð\x9d\x97» ð\x9d\x97\xa0ð\x9d\x97\x9eð\x9d\x97! ð\x9d\x9f".ð\x9d\x9f³ð\x9d\x97\x9f ð\x9d\x97\x94ð\x9d\x97³ð\x9d\x97\x97'],
      dtype=object)
```

In [41]:

```
data_imp3[data_imp3=='!!!'].size
```

Out[41]:

7989

2. Преобразование категориальных признаков в числовые

In [42]:

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

Out[42]:

	c1
0	golf r
1	f-150
2	sierra 1500
3	f-150
4	f-450
5	f-150
6	f-350
7	sierra
8	f-250
9	f-150
10	f-250
11	yukon
12	tundra
13	sierra
14	f-150
15	2500
16	sierra
17	silverado
18	s-class
19	f-350
20	sierra
21	wrangler rubicon unlimited
22	excursion limited
23	olet Silverado 2500HD
24	civic
25	sierra 1500
26	ierra 1500
27	3500 hd
28	silverado 1500
29	spark ev
...	...
509547	f-150
509548	express 3500
509549	silverado 1500
509550	f-150
509551	versa note
509552	fusion

	c1
509553	mus:ang
509554	f-350 super duty
509555	canyon crew cab slt
509556	f-150
509557	3500
509558	wheelchair conversion van
509559	silverado 2500hd
509560	olet Silverado 2500HD
509561	gx470
509562	silverado 1500
509563	ierra 1500
509564	pathfinder platinum
509565	super duty f-250 srw
509566	silverado
509567	f-250
509568	F-150
509569	olet Silverado 1500
509570	jetta 2.0l s
509571	peterbilt 378
509572	xterra
509573	3 series 328i
509574	cherokee
509575	Porsche Macan GTS
509576	f-150

509577 rows × 1 columns

Кодирование категорий целочисленными значениями - label encoding¶

In [43]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [44]:

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

In [45]:

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

Out[45]:

```
array(['golf r', 'f-150', 'sierra 1500', ..., 'Camaro 2-door coupe',  
      'Isuzu VehiCROSS', 'peterbilt 378'], dtype=object)
```

In [46]:

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

Out[46]:

```
array([ 0, 1, 2, ..., 35849, 35850, 35851])
```

In [47]:

```
le.inverse_transform([0, 1, 2, 3])
```

Out[47]:

```
array(['#350', '#4', '$1500 DOWN PAYMENT', '$500 DOWN PROGRAMS!'],  
      dtype=object)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

In [48]:

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

In [49]:

```
cat_enc.shape
```

Out[49]:

```
(509577, 1)
```

In [50]:

```
cat_enc_ohe.shape
```

Out[50]:

```
(509577, 35852)
```

In [51]:

```
cat_enc_ohe
```

Out[51]:

```
<509577x35852 sparse matrix of type '<class 'numpy.float64'>'  
  with 509577 stored elements in Compressed Sparse Row format>
```

In [24]:

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

Out[24]:

```
matrix([[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., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [52]:

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

Out[52]:

	c1
0	golf r
1	f-150
2	sierra 1500
3	f-150
4	f-450
5	f-150
6	f-350
7	sierra
8	f-250
9	f-150

Pandas get_dummies - быстрый вариант one-hot кодирования

In [57]:

```
pd.get_dummies(cat_enc[:30000]).head(2)
```

```
-----
-
MemoryError                                Traceback (most recent call las
t)
<ipython-input-57-bc7013508324> in <module>
----> 1 pd.get_dummies(cat_enc).head(2)

c:\users\innap\miniconda3\lib\site-packages\pandas\core\reshape\reshape.py
in get_dummies(data, prefix, prefix_sep, dummy_na, columns, sparse, drop_f
irst, dtype)
    857         dummy = _get_dummies_1d(col[1], prefix=pre, prefix_sep
=sep,
    858                                dummy_na=dummy_na, sparse=spar
se,
--> 859                                drop_first=drop_first, dtype=d
type)
    860         with_dummies.append(dummy)
    861         result = concat(with_dummies, axis=1)

c:\users\innap\miniconda3\lib\site-packages\pandas\core\reshape\reshape.py
in _get_dummies_1d(data, prefix, prefix_sep, dummy_na, sparse, drop_first,
dtype)
    961
    962     else:
--> 963         dummy_mat = np.eye(number_of_cols, dtype=dtype).take(codes
, axis=0)
    964
    965         if not dummy_na:
```

MemoryError:

In [27]:

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

Out[27]:

	model_08 titan	model_1 series	model_117,000	model_122S Amazon	model_124 spider	model_128i	model_135i	mc co
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	

5 rows × 2494 columns

3 Масштабирование данных

In [28]:

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

3.1. MinMax масштабирование

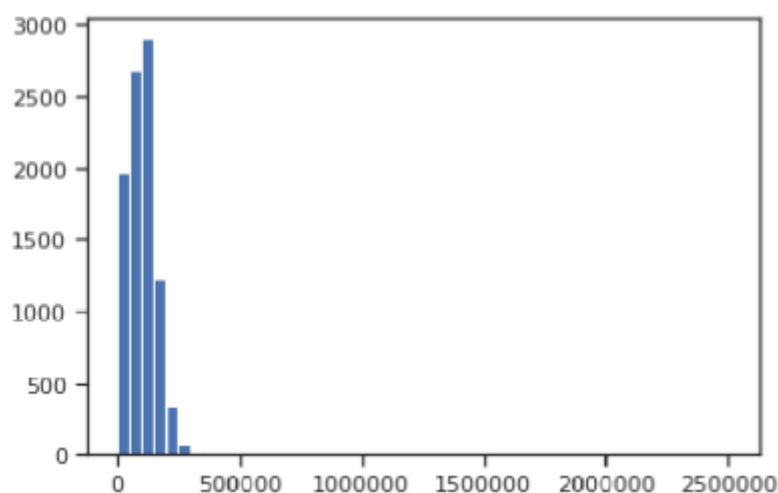
In [34]:

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

In [35]:

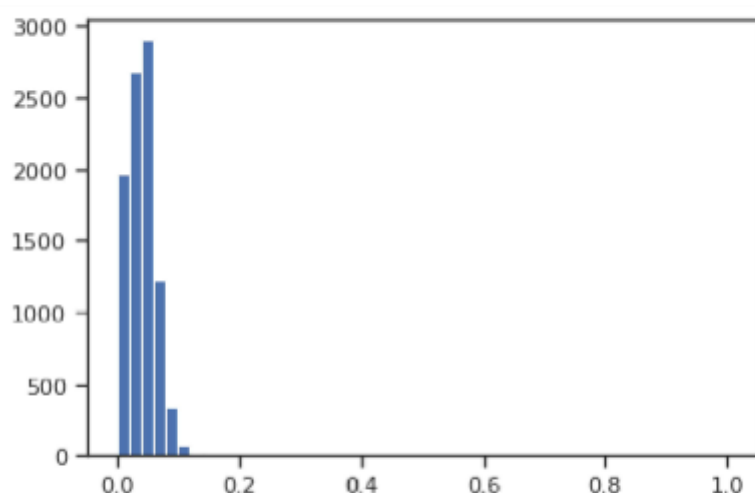
```
plt.hist(data['odometer'], 50)  
plt.show()
```

```
/srv/conda/envs/notebook/lib/python3.7/site-packages/numpy/lib/histograms.  
py:824: RuntimeWarning: invalid value encountered in greater_equal  
    keep = (tmp_a >= first_edge)  
/srv/conda/envs/notebook/lib/python3.7/site-packages/numpy/lib/histograms.  
py:825: RuntimeWarning: invalid value encountered in less_equal  
    keep &= (tmp_a <= last_edge)
```



In [36]:

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



3.2. Масштабирование данных на основе Z-оценки - StandardScaler

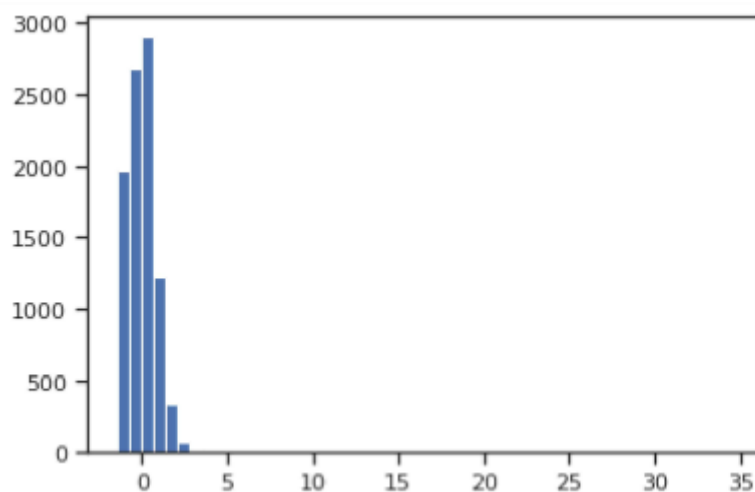
In [37]:

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

In [38]:

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

```
/srv/conda/envs/notebook/lib/python3.7/site-packages/numpy/lib/histograms.  
py:824: RuntimeWarning: invalid value encountered in greater_equal  
    keep = (tmp_a >= first_edge)  
/srv/conda/envs/notebook/lib/python3.7/site-packages/numpy/lib/histograms.  
py:825: RuntimeWarning: invalid value encountered in less_equal  
    keep &= (tmp_a <= last_edge)
```



3.3. Нормализация данных

In [43]:

```
sc3 = Normalizer()  
sc3_data = sc3.fit_transform(data[['odometer']])
```

In [67]:

```
dict = {'odometer': 0}  
data = data.fillna(dict)
```

In [68]:

```
data[data['odometer'].isnull()]
```

Out[68]:

id	url	region	region_url	price	year	manufacturer	model	condition	cylinders	...	drive
----	-----	--------	------------	-------	------	--------------	-------	-----------	-----------	-----	-------

0 rows × 25 columns

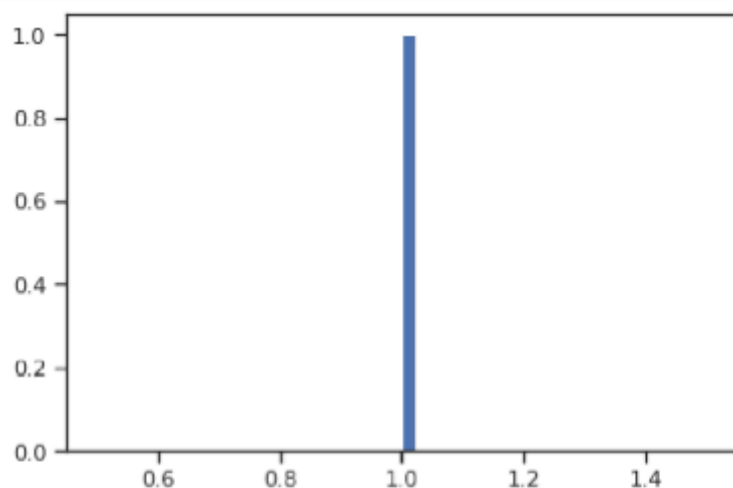


In [69]:

```
sc3 = Normalizer()  
sc3_data = sc3.fit_transform(data[['odometer']])
```

In [70]:

```
plt.hist(sc3_data, 50)  
plt.show()
```



In []:

In []:

In []:

Вывод: в ходе выполнения лабораторной работы были изучены способы обработки пропусков данных, выполнено кодирование категориальных признаков, произведено масштабирование данных.

Обработка пропусков данных может быть выполнена следующими способами:

- Удаление или заполнение нулями недостающих данных;
- Внедрение значений (импьюация).

С помощью импьютации можно обрабатывать числовые и категориальные данные, цель метода – заполнить пропуски в данных усредненными значениями, заданными значениями и т.д. При выполнении лабораторной работы использовался класс SimpleImputer библиотеки sklearn, использовались следующие стратегии: 'mean', 'median', 'most_frequent'.

Кодирование категориальных признаков выполняется при помощи целочисленных значений (label encoding) и бинарных значений (one-hot encoding). Используются классы библиотеки sklearn LabelEncoder и OneHotEncoder соответственно. Также был рассмотрен Pandas get_dummies - быстрый вариант one-hot кодирования.

Масштабирование данных - изменение диапазона измерения величины.

Если признаки лежат в различных диапазонах, то необходимо их нормализовать.

Как правило, применяют два подхода:

- MinMax масштабирование;
- Масштабирование данных на основе Z-оценки.

MinMax масштабирование реализовано при помощи класса MinMaxScaler библиотеки sklearn, а масштабирование данных на основе Z-оценки -

StandardScaler. Нормализация данных предполагает изменение распределения данных. Доступна в sklearn при помощи класса Normalizer.

Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. – 2020. – Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_MISSING (дата обращения: 25.03.2020).
- [2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. – 2019. – Access mode: <https://ipython.readthedocs.io/en/stable/> (online; accessed: 20.02.2019).
- [3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. – 2018. – Access mode: <https://seaborn.pydata.org/> (online; accessed: 20.02.2019).
- [4] pandas 0.24.1 documentation [Electronic resource] // PyData. – 2019. – Access mode: <http://pandas.pydata.org/pandas-docs/stable/> (online; accessed: 20.02.2019).
- [5] Gupta L. Google Play Store Apps [Electronic resource] // Kaggle. – 2019. – Access mode: <https://www.kaggle.com/lava18/google-play-store-apps> (online; accessed: 05.04.2019).