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

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

ОТЧЁТ ПО

Рубежному контролю №1

Выполнил:

студент группы ИУ5-65Б

Кулешова И. А.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва

2024

РК 1 Кулешова Ирина ИУ5-65Б

Вариант 11

Задача 2

Для заданного набора данных проведите обработку пропусков в данных для одного категориального и одного количественного признака. Какие способы обработки пропусков в данных для категориальных и количественных признаков Вы использовали? Какие признаки Вы будете использовать для дальнейшего построения моделей машинного обучения и почему?

Дополнительные требования по группам: для набора данных построить "парные диаграммы".

Датасет 3

Текстовое описание набора данных

В качестве набора данных мы будем использовать вымышленный набор данных игрушек для исследовательского анализа данных (EDA) и тестирования простых моделей прогнозирования - https://www.kaggle.com/carlolepelaars/toy-dataset

Датасет содержит следующие колонки:

- **Number** - простой индексный номер для каждой строки.
- **City** - местоположение человека (Даллас, Нью-Йорк, Лос-Анджелес, Маунтин-Вью, Бостон, Вашингтон, Сан-Диего и Остин).
- **Gender** - пол человека (мужской или женский).
- **Age** - возраст человека (от 25 до 65 лет).
- **Income** - годовой доход человека (в диапазоне от -674 до 177175).
- **Illness** - болен ли человек? (Да или нет).

Импорт библиотек

```
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

```
data = pd.read_csv('data/toy_dataset.csv', sep=",")
```

Основные характеристики датасета

```
# Первые 5 строк датасета
data.head()
```

```
   Number    City Gender  Age   Income Illness
0       1  Dallas   Male   41  40367.0      No
1       2  Dallas   Male   54  45084.0      No
2       3  Dallas   Male   42  52483.0      No
3       4  Dallas   Male   40  40941.0      No
4       5  Dallas   Male   46  50289.0      No
```

```
# Размер датасета
data.shape
```

```
(150000, 6)
```

```
# Список колонок с типами данных
data.dtypes
```

```
Number       int64
City        object
Gender      object
Age          int64
Income     float64
Illness     object
dtype: object
```

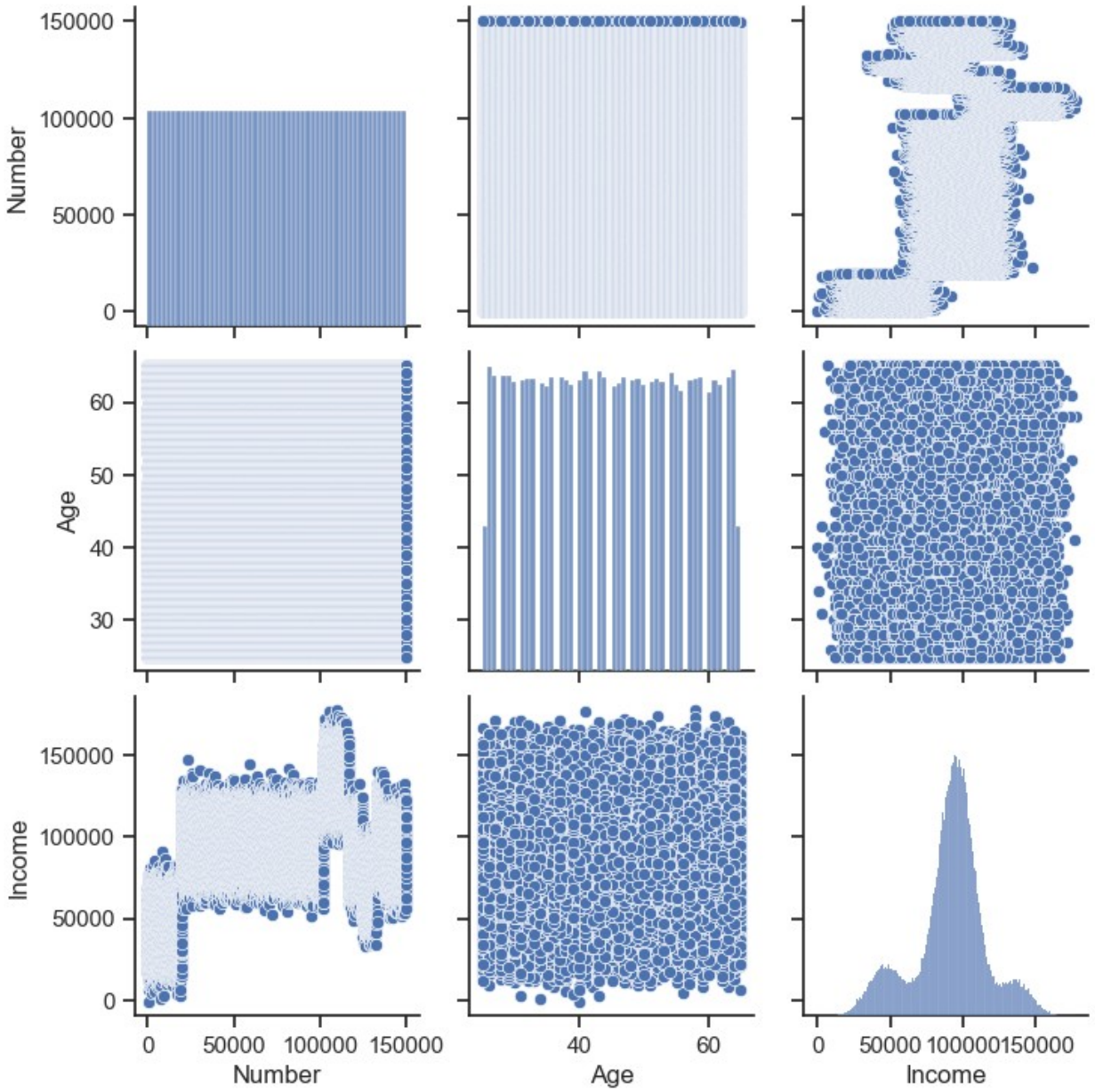
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
Number - 0
City - 0
Gender - 0
Age - 0
Income - 0
Illness - 0
```

Парные диаграммы

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x222b1c2e850>
```

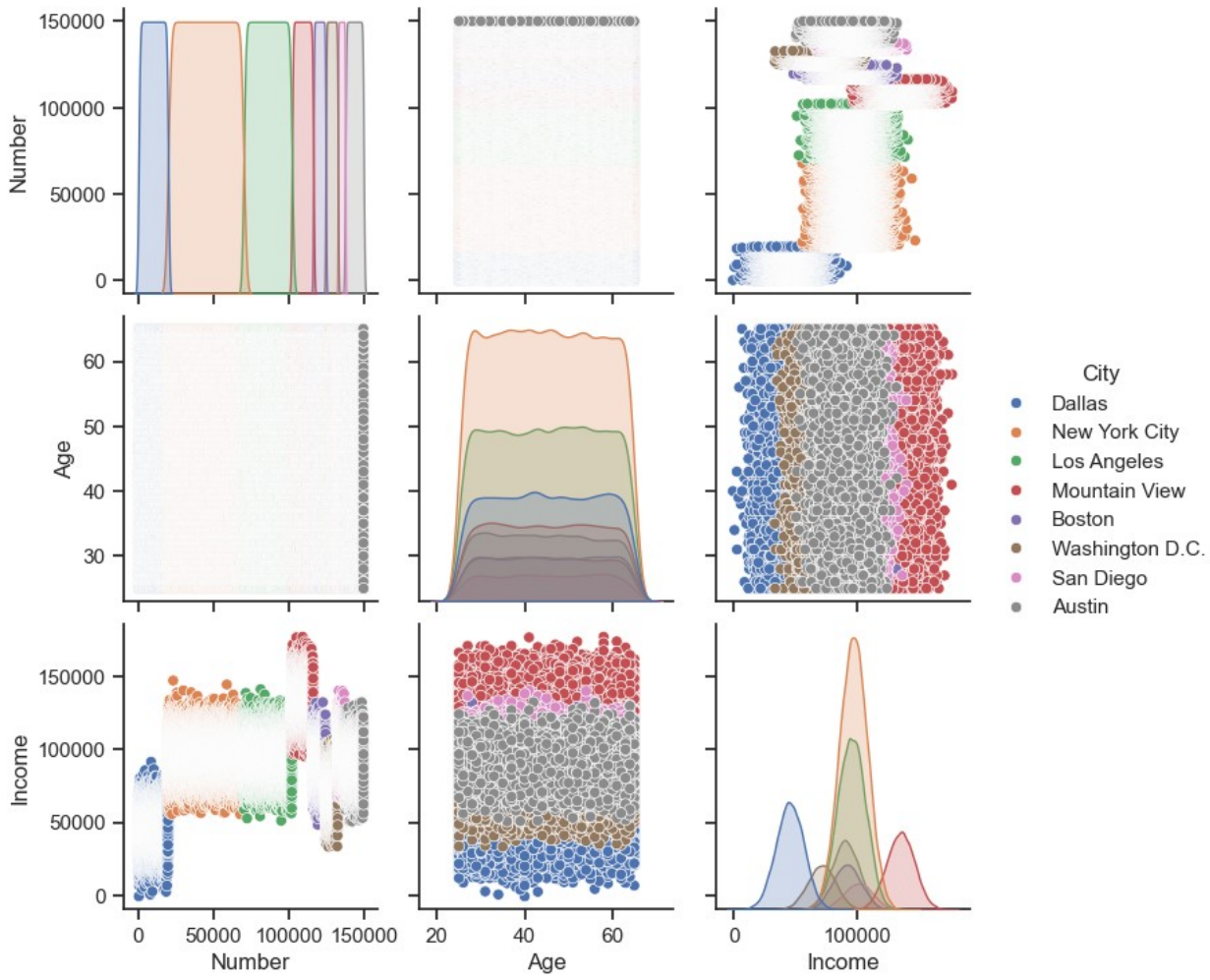


Группировка по значениям какого-либо признака:

Группировать по признаку Number бессмысленно, так как каждое значение данного столбца уникально.

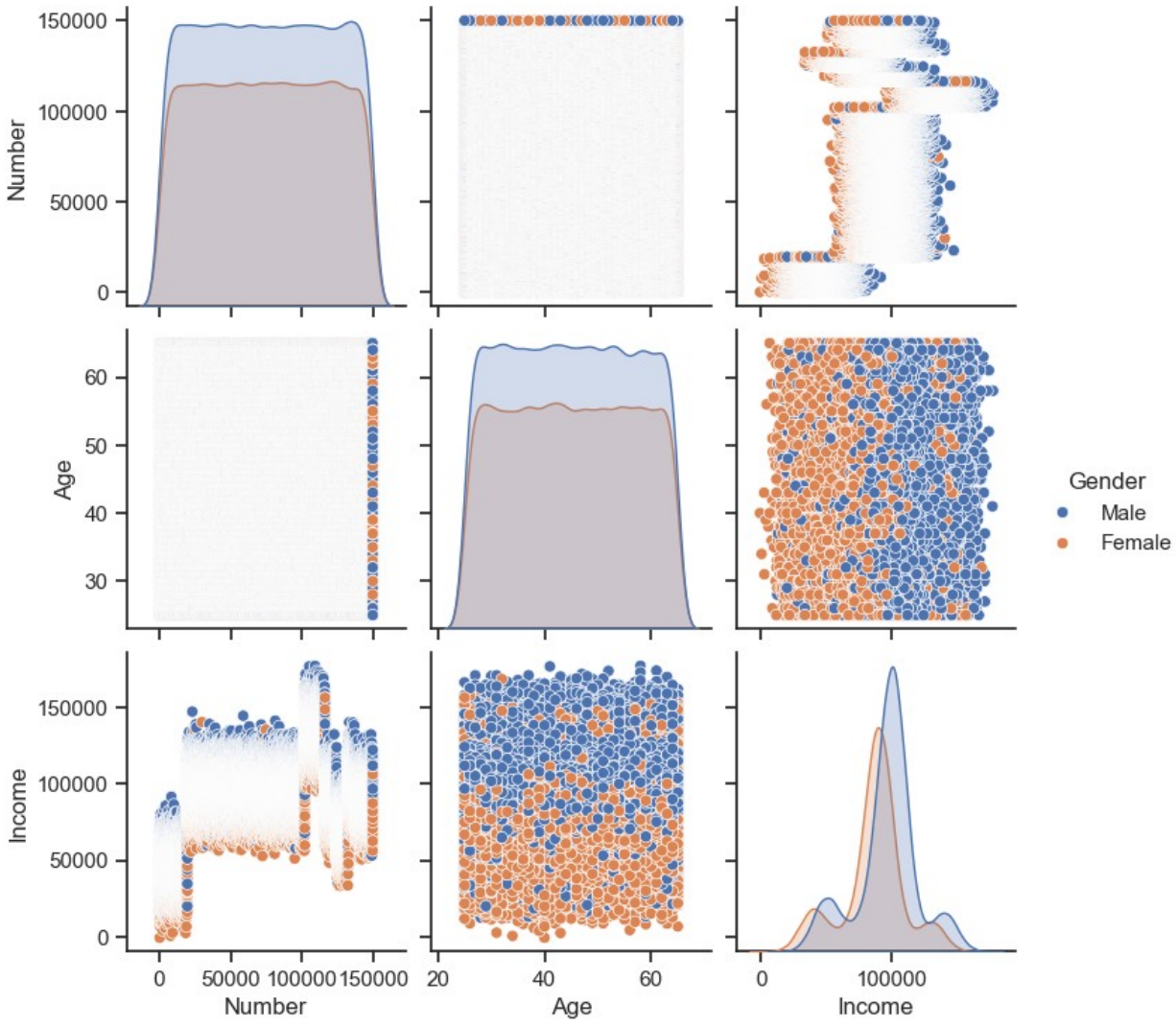
```
sns.pairplot(data, hue="City")
```

```
<seaborn.axisgrid.PairGrid at 0x222b36709d0>
```




```
sns.pairplot(data, hue="Gender")
```

```
<seaborn.axisgrid.PairGrid at 0x222b4383710>
```



Обработка пропусков данных

В данном датасете нет пропусков данных, что видно из первичного его анализа, поэтому используем дополнительный датасет, который содержит пропуски данных.

Датасет - car_prices.csv

Датасет содержит следующие колонки:

- **year** - год выпуска автомобиля.
- **make** - марка или производитель автомобиля.
- **model** - конкретная модель автомобиля.
- **trim** - дополнительное обозначение модели автомобиля.
- **body** - тип кузова автомобиля (например, внедорожник, седан).
- **transmission** - тип коробки передач в автомобиле (например, автомат).
- **vin** - идентификационный номер транспортного средства — уникальный код для каждого транспортного средства.

- **state** - государство, в котором зарегистрировано транспортное средство.
- **condition** - состояние автомобиля, оцененное по шкале.
- **odometer** - пробег или расстояние, пройденное транспортным средством.

```
data = pd.read_csv('data/car_prices.csv', sep=",")
```

```
data.shape
```

```
(558837, 16)
```

```
data.dtypes
```

```
year              int64
make             object
model            object
trim             object
body             object
transmission     object
vin              object
state            object
condition       float64
odometer        float64
color            object
interior         object
seller           object
mmr             float64
sellingprice    float64
saledate         object
dtype: object
```

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

```
year                0
make            10301
model           10399
trim            10651
body            13195
transmission    65352
vin                 4
state               0
condition       11820
odometer           94
color             749
interior          749
seller              0
mmr                38
sellingprice       12
saledate           12
dtype: int64
```

```
# Первые 5 строк датасета
data.head()
```

```
   year   make                model        trim   body transmission  \
0  2015    Kia              Sorento          LX    SUV    automatic
1  2015    Kia              Sorento          LX    SUV    automatic
2  2014    BMW             3 Series  328i SULEV  Sedan    automatic
3  2015  Volvo                  S60          T5  Sedan    automatic
4  2014    BMW  6 Series Gran Coupe        650i  Sedan    automatic

                 vin state  condition  odometer  color interior  \
0  5xyktca69fg566472    ca        5.0   16639.0  white    black
1  5xyktca69fg561319    ca        5.0    9393.0  white    beige
2  wba3c1c51ek116351    ca       45.0    1331.0   gray    black
3  yv1612tb4f1310987    ca       41.0   14282.0  white    black
4  wba6b2c57ed129731    ca       43.0    2641.0   gray    black

                                seller      mmr  sellingprice  \
0                kia motors america  inc  20500.0       21500.0
1                kia motors america  inc  20800.0       21500.0
2  financial services remarketing (lease)  31900.0       30000.0
3                volvo na rep/world omni  27500.0       27750.0
4  financial services remarketing (lease)  66000.0       67000.0

                                saledate
0  Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
1  Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2  Thu Jan 15 2015 04:30:00 GMT-0800 (PST)
3  Thu Jan 29 2015 04:30:00 GMT-0800 (PST)
4  Thu Dec 18 2014 12:30:00 GMT-0800 (PST)
```

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

```
Всего строк: 558837
```

Обработка пропусков в числовых данных

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

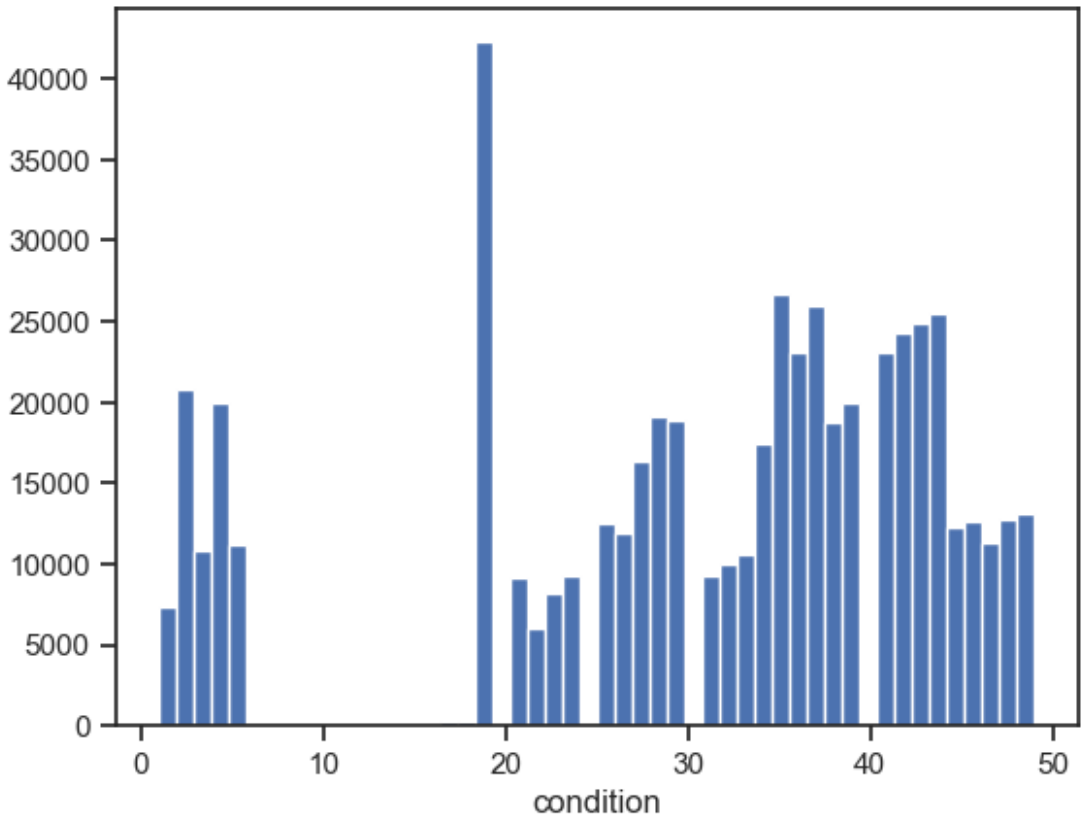
```
Колонка condition. Тип данных float64. Количество пустых значений 11820, 2.12%.
Колонка odometer. Тип данных float64. Количество пустых значений 94, 0.02%.
Колонка mmr. Тип данных float64. Количество пустых значений 38, 0.01%.
Колонка sellingprice. Тип данных float64. Количество пустых значений 12, 0.0%.
```

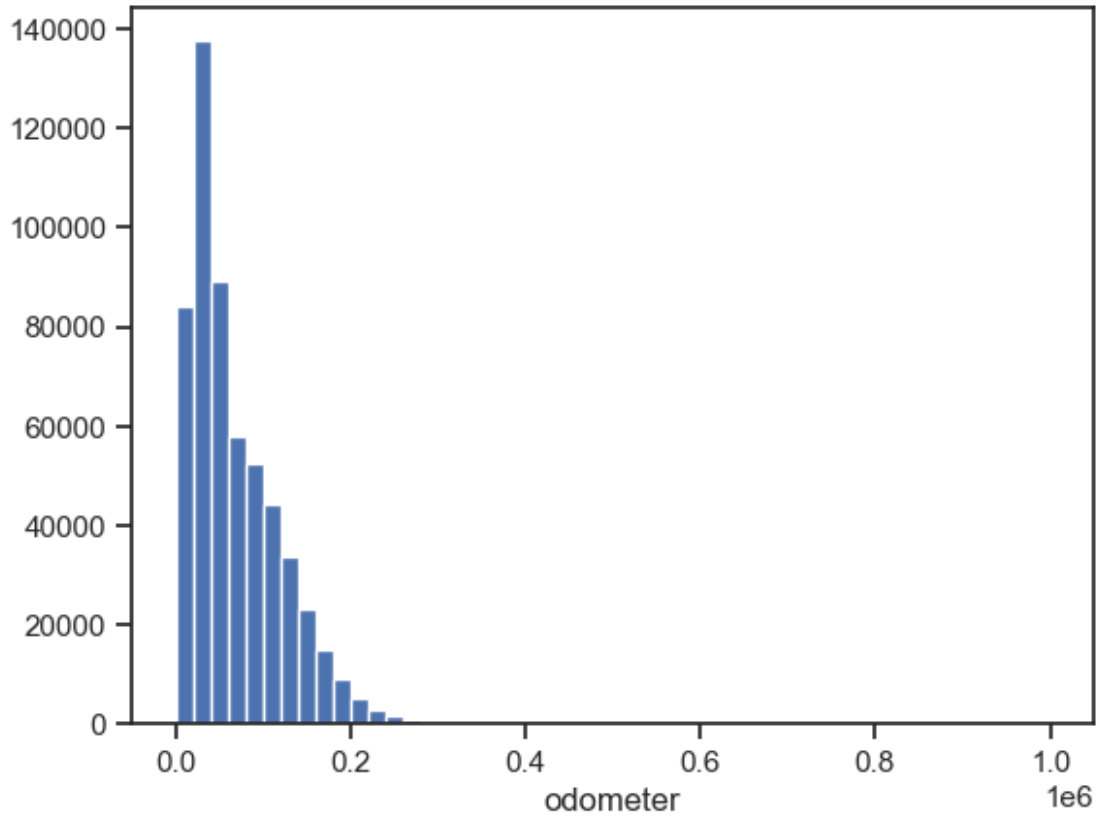
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

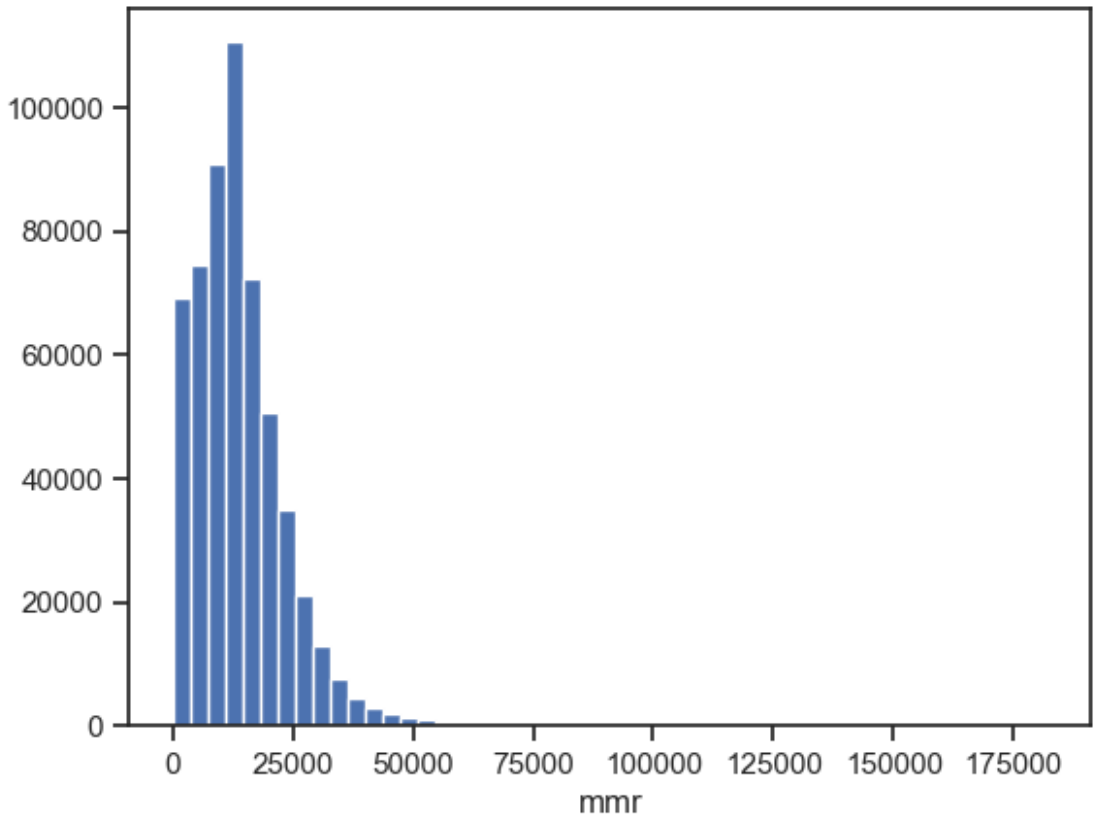
	condition	odometer	mmr	sellingprice
0	5.0	16639.0	20500.0	21500.0
1	5.0	9393.0	20800.0	21500.0
2	45.0	1331.0	31900.0	30000.0
3	41.0	14282.0	27500.0	27750.0
4	43.0	2641.0	66000.0	67000.0
...	...	...	...	...
558832	45.0	18255.0	35300.0	33000.0
558833	5.0	54393.0	30200.0	30800.0
558834	48.0	50561.0	29800.0	34000.0
558835	38.0	16658.0	15100.0	11100.0
558836	34.0	15008.0	29600.0	26700.0

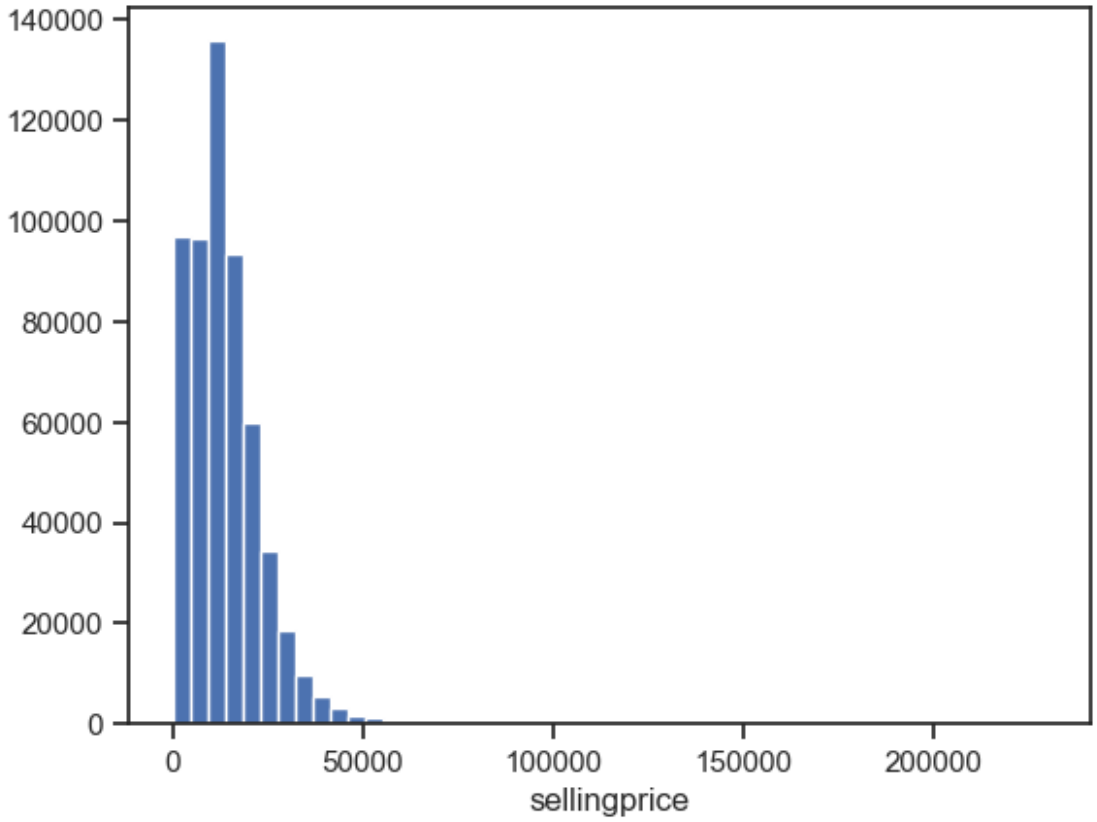
```
[558837 rows x 4 columns]
```

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


40000
35000
30000
25000
20000
15000
10000
5000
0
0
10
20
30
40
50
condition


140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
odometer
1e6


100000
80000
60000
40000
20000
0
0
25000
50000
75000
100000
125000
150000
175000
mmr


140000
120000
100000
80000
60000
40000
20000
0
0
50000
100000
150000
200000
sellingprice

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

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

def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]
    return column, strategy_param, filled_data.size, filled_data[0],
filled_data[filled_data.size-1]

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

```
('odometer', 'mean', 94, 68320.01776666554, 68320.01776666554)
```

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

```
('odometer', 'median', 94, 52254.0, 52254.0)
```

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

```
('odometer', 'most_frequent', 94, 1.0, 1.0)
```

Вывод: Посмотрев на диаграмму, можно сказать, что замена средним наиболее предпочтительна, так как значение, получаемое с помощью среднего наиболее близкое к пику гистограммы, хотя по сравнению с общим числом строк число строк с пропусками настолько мало, что данными 94 строками можно пренебречь.

Обработка пропусков в категориальных данных

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

```
Колонка make. Тип данных object. Количество пустых значений 10301,
1.84%.
```

```
Колонка model. Тип данных object. Количество пустых значений 10399, 1.86%.
Колонка trim. Тип данных object. Количество пустых значений 10651, 1.91%.
Колонка body. Тип данных object. Количество пустых значений 13195, 2.36%.
Колонка transmission. Тип данных object. Количество пустых значений 65352, 11.69%.
Колонка vin. Тип данных object. Количество пустых значений 4, 0.0%.
Колонка color. Тип данных object. Количество пустых значений 749, 0.13%.
Колонка interior. Тип данных object. Количество пустых значений 749, 0.13%.
Колонка saledate. Тип данных object. Количество пустых значений 12, 0.0%.
```

Колонки vin, color, inerior, saledate рассматривать не будем, так как количество пустых значений в них пренебрежимо мало по сравнению с общим количеством строк в таблице.

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

```
    body
0    SUV
1    SUV
2  Sedan
3  Sedan
4  Sedan
```

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

```
(13195, 1)
```

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

```
array([['SUV'],
       ['SUV'],
       ['Sedan'],
       ...,
       ['SUV'],
       ['sedan'],
       ['SuperCrew']], dtype=object)
```

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

```
array(['Access Cab', 'Beetle Convertible', 'CTS Coupe', 'CTS Wagon',
       'CTS-V Coupe', 'CTS-V Wagon', 'Cab Plus', 'Cab Plus 4', 'Club
Cab',
       'Convertible', 'Coupe', 'Crew Cab', 'CrewMax Cab', 'Double
Cab',
       'E-Series Van', 'Elantra Coupe', 'Extended Cab', 'G
Convertible',
       'G Coupe', 'G Sedan', 'G37 Convertible', 'G37 Coupe',
       'Genesis Coupe', 'GranTurismo Convertible', 'Hatchback',
       'King Cab', 'Koup', 'Mega Cab', 'Minivan', 'Navitgation',
       'Promaster Cargo Van', 'Q60 Convertible', 'Q60 Coupe', 'Quad
Cab',
       'Ram Van', 'Regular Cab', 'SUV', 'Sedan', 'SuperCab',
'SuperCrew',
       'TSX Sport Wagon', 'Transit Van', 'Van', 'Wagon', 'Xtracab',
       'access cab', 'beetle convertible', 'cab plus 4', 'club cab',
       'convertible', 'coupe', 'crew cab', 'crewmax cab', 'cts coupe',
       'cts wagon', 'cts-v coupe', 'double cab', 'e-series van',
       'elantra coupe', 'extended cab', 'g convertible', 'g coupe',
       'g sedan', 'g37 convertible', 'g37 coupe', 'genesis coupe',
       'granturismo convertible', 'hatchback', 'king cab', 'koup',
       'mega cab', 'minivan', 'promaster cargo van', 'q60
convertible',
       'q60 coupe', 'quad cab', 'regular cab', 'regular-cab', 'sedan',
       'supercab', 'supercrew', 'suv', 'transit van', 'tsx sport
wagon',
       'van', 'wagon', 'xtracab'], dtype=object)
```

Для обработки пропусков в числовых данных я хотела использовать импьютацию и заменила пропуски на значения среднего выборки. А пропуски в категориальных данных я заполняю с помощью стратегии 'most_frequent'.

Для дальнейшего построения моделей машинного обучения, если количество пропусков пренережимо мало по сравнению с общим количеством объектов в датасете, то можно просто удалить строки с пропусками, однако если это не так, то нужно заменить пропуски такими числами, которые не изменят зависимости между признаками. Для числовых данных это будет замена на значение среднего, медианы или моды. Для категориальных данных это замена наиболее частым значением или константой (при дальнейшей кластеризации с целью выявления для каких объектов значение данного признака не определяется/не определено).