```
{
  "authors": [
    {
      "name": "Алексеев Андрей Сергеевич"
    }
  ],
  "group": "ИУ5-62Б",
  "kernelspec": {
    "name": "python3",
    "display_name": "Python 3 (ipykernel)",
    "language": "python"
  },
  "language_info": {
    "name": "python",
    "version": "3.9.7",
    "mimetype": "text/x-python",
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "pygments_lexer": "ipython3",
    "nbconvert_exporter": "python",
    "file_extension": ".py"
  },
  "title": "Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных"
}
```

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

```python
# Будем использовать только обучающую выборку
data = pd.read_csv('winemag-data_first150k.csv', sep=",")
```

```python
# размер набора данных
data.shape
```

```
(150930, 11)
```

```python
# типы колонок
data.dtypes
```

```
Unnamed: 0       int64
country         object
description     object
designation     object
points           int64
```

```
price          float64
province        object
region_1        object
region_2        object
variety         object
winery          object
dtype: object
```

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

```
Unnamed: 0         0
country            5
description        0
designation    45735
points             0
price          13695
province           5
region_1       25060
region_2       89977
variety            0
winery             0
dtype: int64
```

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

```
   Unnamed: 0 country
description  \
0           0      US  This tremendous 100% varietal wine hails
from ...
1           1   Spain  Ripe aromas of fig, blackberry and cassis
are ...
2           2      US  Mac Watson honors the memory of a wine once
ma...
3           3      US  This spent 20 months in 30% new French oak,
an...
4           4  France  This is the top wine from La Bégude, named
aft...

                            designation  points  price        province
\
0                     Martha's Vineyard      96  235.0      California

1  Carodorum Selección Especial Reserva      96  110.0  Northern Spain

2         Special Selected Late Harvest      96   90.0      California

3                               Reserve      96   65.0          Oregon

4                            La Brûlade      95   66.0        Provence
```

```
            region_1           region_2             variety  \
0        Napa Valley               Napa  Cabernet Sauvignon   
1               Toro                NaN       Tinta de Toro   
2     Knights Valley             Sonoma     Sauvignon Blanc   
3  Willamette Valley  Willamette Valley          Pinot Noir   
4             Bandol                NaN  Provence red blend   

                    winery  
0                    Heitz  
1  Bodega Carmen Rodríguez  
2                 Macauley  
3                    Ponzi  
4     Domaine de la Bégude  
```

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

```
Всего строк: 150930
```

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

```
((150930, 11), (150930, 5))
```

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

```
((150930, 11), (39241, 11))
```

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

```
   Unnamed: 0 country                                        
description  \
0           0      US  This tremendous 100% varietal wine hails 
from ...   
1           1   Spain  Ripe aromas of fig, blackberry and cassis 
are ...   
2           2      US  Mac Watson honors the memory of a wine once 
ma...   
3           3      US  This spent 20 months in 30% new French oak, 
an...   
4           4  France  This is the top wine from La Bégude, named 
aft...   
```

```
                            designation  points  price        province  \
0                     Martha's Vineyard      96  235.0      California   

1  Carodorum Selección Especial Reserva      96  110.0  Northern Spain   

2         Special Selected Late Harvest      96   90.0      California   

3                               Reserve      96   65.0          Oregon   

4                            La Brûlade      95   66.0        Provence   


            region_1           region_2             variety  \
0        Napa Valley               Napa  Cabernet Sauvignon   
1               Toro                  0       Tinta de Toro   
2     Knights Valley             Sonoma     Sauvignon Blanc   
3  Willamette Valley  Willamette Valley          Pinot Noir   
4             Bandol                  0  Provence red blend   

                    winery  
0                    Heitz  
1  Bodega Carmen Rodríguez  
2                 Macauley  
3                    Ponzi  
4     Domaine de la Bégude  
```

```
# 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))
```

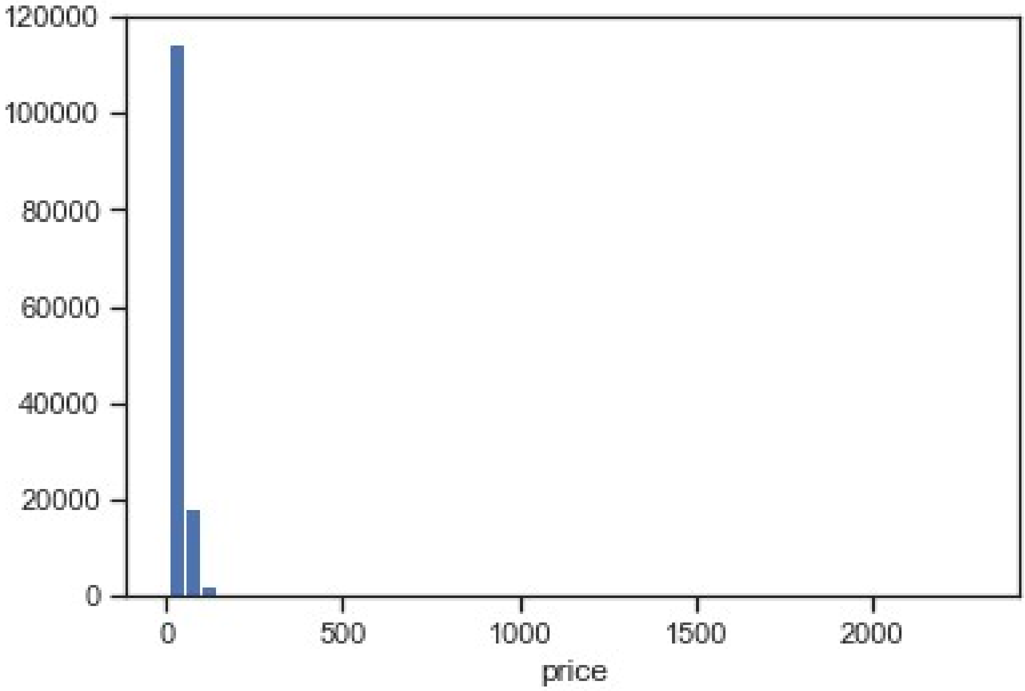
```
Колонка price. Тип данных float64. Количество пустых значений 13695, 
9.07%.
```

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

```
        price
0       235.0
```

```
1        110.0
2         90.0
3         65.0
4         66.0
...        ...
150925    20.0
150926    27.0
150927    20.0
150928    52.0
150929    15.0

[150930 rows x 1 columns]
```

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



```
data_num_price = data_num[['price']]
data_num_price.head()
```

```
   price
0  235.0
1  110.0
2   90.0
3   65.0
4   66.0
```

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

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

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

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

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

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

```
('mean',
 array([33.13148249, 33.13148249, 33.13148249, ..., 33.13148249,
        33.13148249, 33.13148249]))
```

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

```
('median', array([24., 24., 24., ..., 24., 24., 24.]))
```

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

```
('most_frequent', array([20., 20., 20., ..., 20., 20., 20.]))
```

```
# Более сложная функция, которая позволяет задавать колонку и вид
импьютации
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

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

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

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0],
filled_data[filled_data.size-1]
```

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

```
               price
count  137235.000000
mean       33.131482
std        36.322536
min         4.000000
25%        16.000000
50%        24.000000
75%        40.000000
max      2300.000000
```

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

```
('price', 'mean', 13695, 33.13148249353299, 33.13148249353299)
```

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

```
('price', 'median', 13695, 24.0, 24.0)
```

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

```
('price', 'most_frequent', 13695, 20.0, 20.0)
```

```
# 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 country. Тип данных object. Количество пустых значений 5,
0.0%.
Колонка designation. Тип данных object. Количество пустых значений
45735, 30.3%.
Колонка province. Тип данных object. Количество пустых значений 5,
0.0%.
Колонка region_1. Тип данных object. Количество пустых значений 25060,
16.6%.
Колонка region_2. Тип данных object. Количество пустых значений 89977,
59.62%.
```

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

```
  country
0      US
```

```
1   Spain
2      US
3      US
4  France
```

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

```
array(['US', 'Spain', 'France', 'Italy', 'New Zealand', 'Bulgaria',
       'Argentina', 'Australia', 'Portugal', 'Israel', 'South Africa',
       'Greece', 'Chile', 'Morocco', 'Romania', 'Germany', 'Canada',
       'Moldova', 'Hungary', 'Austria', 'Croatia', 'Slovenia', nan,
       'India', 'Turkey', 'Macedonia', 'Lebanon', 'Serbia', 'Uruguay',
       'Switzerland', 'Albania', 'Bosnia and Herzegovina', 'Brazil',
       'Cyprus', 'Lithuania', 'Japan', 'China', 'South Korea',
'Ukraine',
       'England', 'Mexico', 'Georgia', 'Montenegro', 'Luxembourg',
       'Slovakia', 'Czech Republic', 'Egypt', 'Tunisia', 'US-France'],
      dtype=object)
```

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

```
(5, 1)
```

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

```
array([['US'],
       ['Spain'],
       ['US'],
       ...,
       ['Italy'],
       ['France'],
       ['Italy']], dtype=object)
```

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

```
array(['Albania', 'Argentina', 'Australia', 'Austria',
       'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada',
'Chile',
       'China', 'Croatia', 'Cyprus', 'Czech Republic', 'Egypt',
'England',
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
       'Israel', 'Italy', 'Japan', 'Lebanon', 'Lithuania',
'Luxembourg',
       'Macedonia', 'Mexico', 'Moldova', 'Montenegro', 'Morocco',
       'New Zealand', 'Portugal', 'Romania', 'Serbia', 'Slovakia',
       'Slovenia', 'South Africa', 'South Korea', 'Spain',
'Switzerland',
       'Tunisia', 'Turkey', 'US', 'US-France', 'Ukraine', 'Uruguay'],
      dtype=object)
```

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

```
array([['US'],
       ['Spain'],
       ['US'],
       ...,
       ['Italy'],
       ['France'],
       ['Italy']], dtype=object)
```

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

```
array(['Albania', 'Argentina', 'Australia', 'Austria',
       'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada',
'Chile',
       'China', 'Croatia', 'Cyprus', 'Czech Republic', 'Egypt',
'England',
       'France', 'Georgia', 'Germany', 'Greece', 'Hungary', 'India',
       'Israel', 'Italy', 'Japan', 'Lebanon', 'Lithuania',
'Luxembourg',
       'Macedonia', 'Mexico', 'Moldova', 'Montenegro', 'Morocco',
       'New Zealand', 'Portugal', 'Romania', 'Serbia', 'Slovakia',
       'Slovenia', 'South Africa', 'South Korea', 'Spain',
'Switzerland',
       'Tunisia', 'Turkey', 'US', 'US-France', 'Ukraine', 'Uruguay'],
      dtype=object)
```

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

```
62402
```

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

```
            c1
0           US
1        Spain
2           US
3           US
4       France
...        ...
150925   Italy
150926  France
150927   Italy
150928  France
```

```
150929   Italy

[150930 rows x 1 columns]
```

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

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

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

```
array(['US', 'Spain', 'France', 'Italy', 'New Zealand', 'Bulgaria',
       'Argentina', 'Australia', 'Portugal', 'Israel', 'South Africa',
       'Greece', 'Chile', 'Morocco', 'Romania', 'Germany', 'Canada',
       'Moldova', 'Hungary', 'Austria', 'Croatia', 'Slovenia',
'India',
       'Turkey', 'Macedonia', 'Lebanon', 'Serbia', 'Uruguay',
       'Switzerland', 'Albania', 'Bosnia and Herzegovina', 'Brazil',
       'Cyprus', 'Lithuania', 'Japan', 'China', 'South Korea',
'Ukraine',
       'England', 'Mexico', 'Georgia', 'Montenegro', 'Luxembourg',
       'Slovakia', 'Czech Republic', 'Egypt', 'Tunisia', 'US-France'],
      dtype=object)
```

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

```
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])
```

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

```
array(['Albania', 'Argentina', 'Australia', 'Austria'], dtype=object)
```

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

```
cat_enc.shape
```

```
(150930, 1)
```

```
cat_enc_ohe.shape
```

```
(150930, 48)
```

```
cat_enc_ohe
```

```
<150930x48 sparse matrix of type '<class 'numpy.float64'>'
     with 150930 stored elements in Compressed Sparse Row format>
```

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

```
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., 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., 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., 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., 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.,
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., 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.,
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., 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., 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., 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., 0., 0., 0., 0., 0., 0., 0.,
0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0.]])
```

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

```
       c1
0      US
1   Spain
2      US
3      US
4  France
5   Spain
6   Spain
7   Spain
8      US
9      US
```

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

```
   c1_Albania  c1_Argentina  c1_Australia  c1_Austria  \
0           0             0             0           0
1           0             0             0           0
2           0             0             0           0
3           0             0             0           0
4           0             0             0           0

   c1_Bosnia and Herzegovina  c1_Brazil  c1_Bulgaria  c1_Canada
c1_Chile  \
0                          0          0            0          0
0
1                          0          0            0          0
0
2                          0          0            0          0
0
3                          0          0            0          0
0
4                          0          0            0          0
0

   c1_China  ...  c1_South Africa  c1_South Korea  c1_Spain
c1_Switzerland  \
0         0  ...                0               0         0
0
```

```
1         0  ...                0               0         1
0
2         0  ...                0               0         0
0
3         0  ...                0               0         0
0
4         0  ...                0               0         0
0

   c1_Tunisia  c1_Turkey  c1_US  c1_US-France  c1_Ukraine  c1_Uruguay

0           0          0      1             0           0           0

1           0          0      0             0           0           0

2           0          0      1             0           0           0

3           0          0      1             0           0           0

4           0          0      0             0           0           0


[5 rows x 48 columns]
```

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

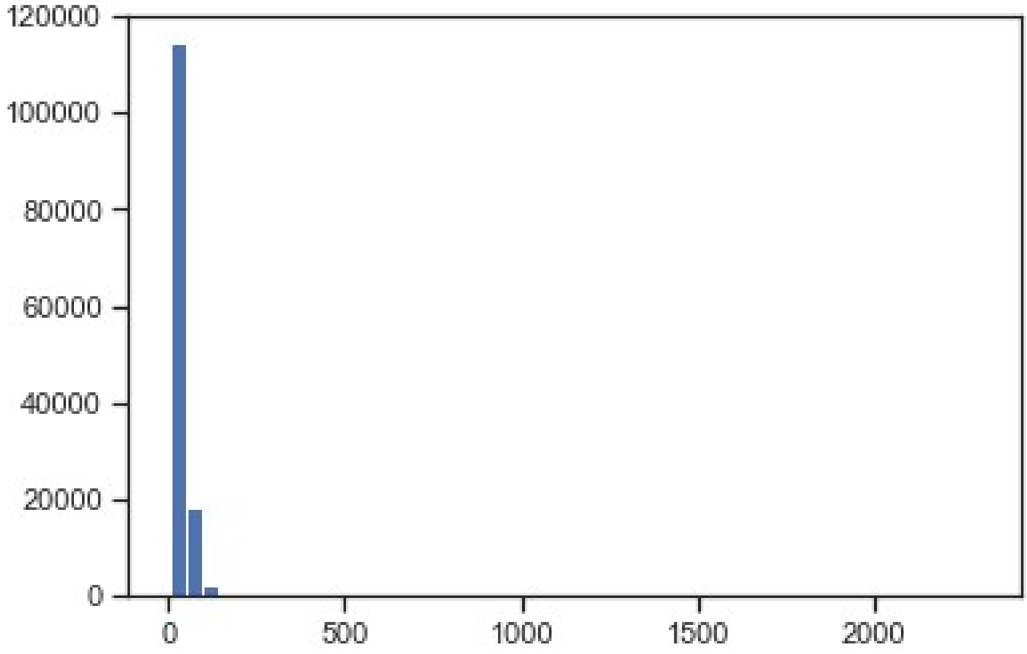
```
   country_Albania  country_Argentina  country_Australia  
country_Austria  \
0                0                  0                  0
0
1                0                  0                  0
0
2                0                  0                  0
0
3                0                  0                  0
0
4                0                  0                  0
0

   country_Bosnia and Herzegovina  country_Brazil  country_Bulgaria  \
0                               0               0                 0
1                               0               0                 0
2                               0               0                 0
3                               0               0                 0
4                               0               0                 0

   country_Canada  country_Chile  country_China  ...  country_South 
Korea  \
0               0              0              0  ...
0
```

```
1               0              0              0  ...
0
2               0              0              0  ...
0
3               0              0              0  ...
0
4               0              0              0  ...
0

   country_Spain  country_Switzerland  country_Tunisia  country_Turkey
\
0              0                    0                0               0

1              1                    0                0               0

2              0                    0                0               0

3              0                    0                0               0

4              0                    0                0               0


   country_US  country_US-France  country_Ukraine  country_Uruguay  \
0           1                  0                0                0
1           0                  0                0                0
2           1                  0                0                0
3           1                  0                0                0
4           0                  0                0                0

   country_nan
0            0
1            0
2            0
3            0
4            0

[5 rows x 49 columns]
```

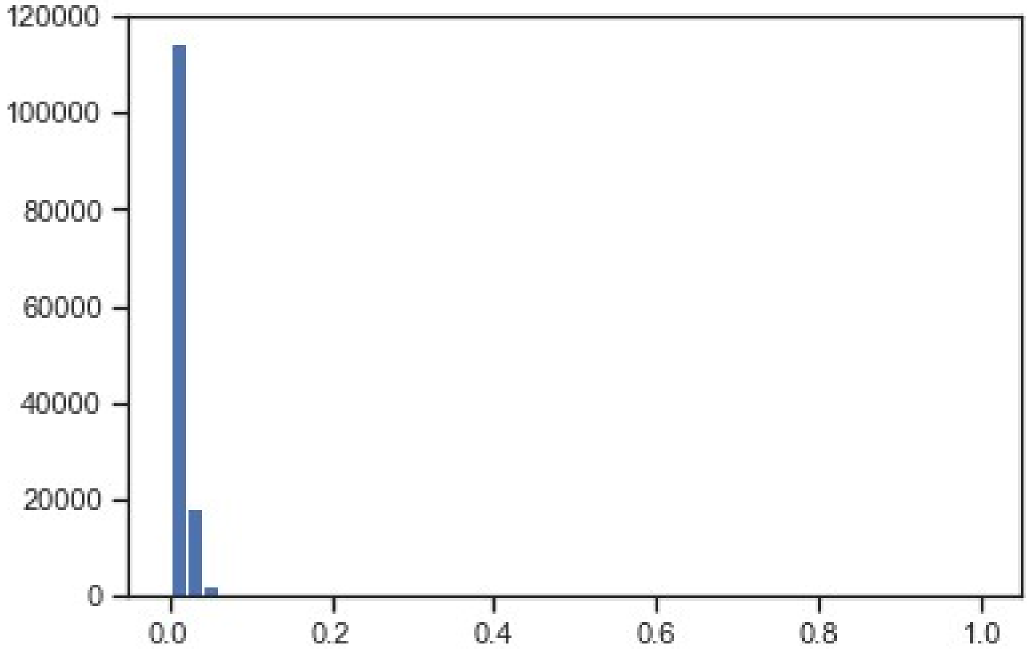
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
Normalizer
```

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

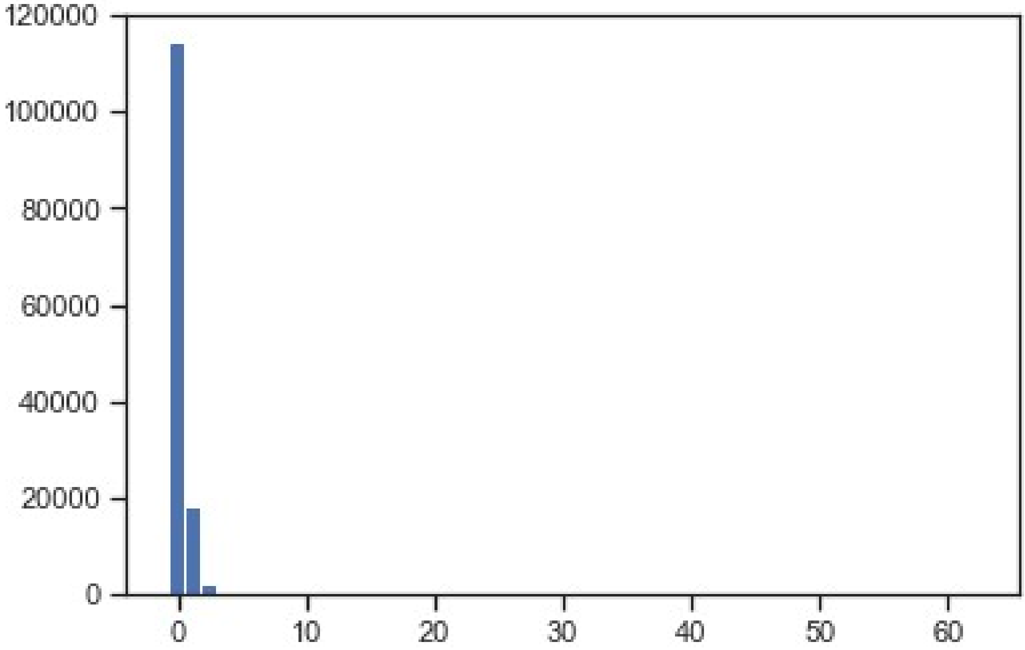
```
plt.hist(data['price'], 50)
plt.show()
```



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



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

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


120000
100000
80000
60000
40000
20000
0
0
10
20
30
40
50
60