Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

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

Выполнил:
студент группы ИУ5-23М
Богомолов Д.Н.

Москва — 2020 г.

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

```
1  import numpy as np
2  import pandas as pd
3  pd.set_option('display.max.columns', 100)
4  %matplotlib inline
5  import matplotlib.pyplot as plt
6  import seaborn as sns
7  import warnings
8  warnings.filterwarnings('ignore')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnin
  import pandas.util.testing as tm
```

Загрузка и первичный анализ данных

```
1  data = pd.read_csv("/content/drive/My Drive/train.csv", sep=',')
2  data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

```
1  data.shape
```

```
(1460, 81)
```

```
1  data.dtypes
```

```
Id                int64
MSSubClass        int64
MSZoning         object
LotFrontage     float64
LotArea           int64
                  ...
MoSold            int64
YrSold            int64
SaleType         object
SaleCondition    object
SalePrice         int64
Length: 81, dtype: object
```

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

```
Id                  0
MSSubClass          0
MSZoning            0
LotFrontage       259
LotArea             0
                   ...
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
Length: 81, dtype: int64
```

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

```
Всего строк: 1460
```

1. Обработка пропусков в данных

1.1. Простые стратегии - удаление или заполнение нуля

Удаление колонок, содержащих пустые значения

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

```
((1460, 81), (1460, 62))
```

Удаление строк с пустыми значениями

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

```
((1460, 81), (0, 81))
```

```
data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

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

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	RL	65.0	8450	Pave	0	Reg	
1	2	20	RL	80.0	9600	Pave	0	Reg	
2	3	60	RL	68.0	11250	Pave	0	IR1	
3	4	70	RL	60.0	9550	Pave	0	IR1	
4	5	60	RL	84.0	14260	Pave	0	IR1	

1.2. "Внедрение значений" - импьютация (imputation)

1.2.1. 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=='int64' or dt=='float64'):
```

```
7           num_cols.append(col)
8           temp_perc = round((temp_null_count / total_count) * 100.0, 2)
9           print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format
```

```
Колонка LotFrontage. Тип данных float64. Количество пустых значений 259, 17.74%.
Колонка MasVnrArea. Тип данных float64. Количество пустых значений 8, 0.55%.
Колонка GarageYrBlt. Тип данных float64. Количество пустых значений 81, 5.55%.
```

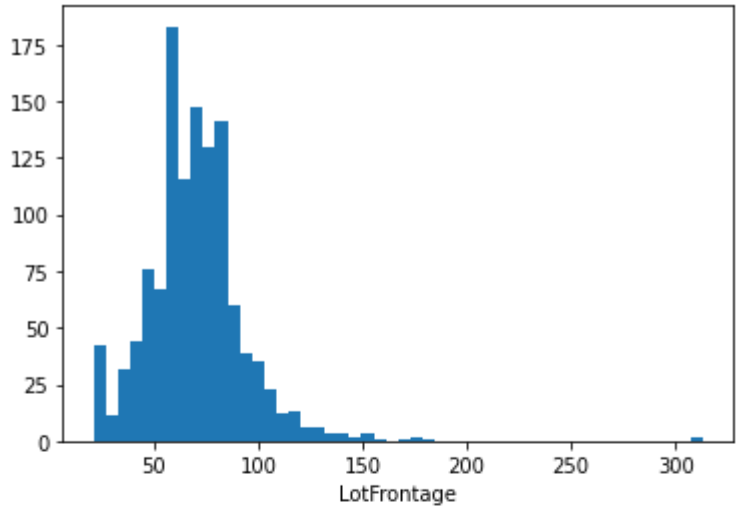
```
1   data_num = data[num_cols]
2   data_num
```

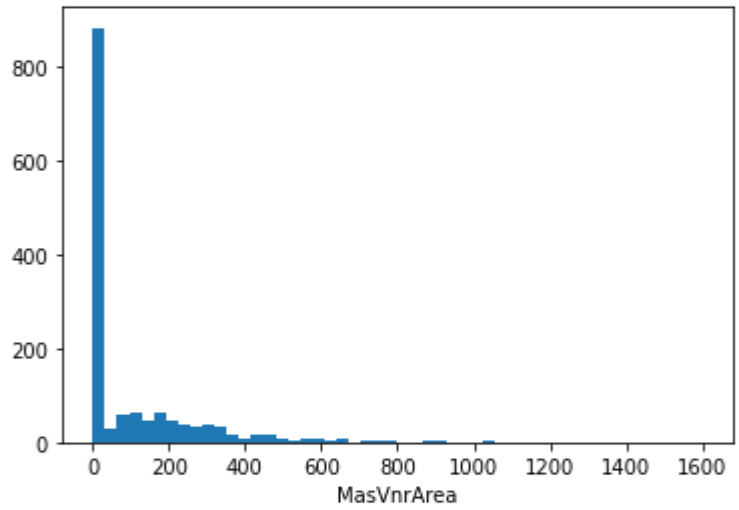
	LotFrontage	MasVnrArea	GarageYrBlt
0	65.0	196.0	2003.0
1	80.0	0.0	1976.0
2	68.0	162.0	2001.0
3	60.0	0.0	1998.0
4	84.0	350.0	2000.0
...	...	...	...
1455	62.0	0.0	1999.0
1456	85.0	119.0	1978.0
1457	66.0	0.0	1941.0
1458	68.0	0.0	1950.0
1459	75.0	0.0	1965.0

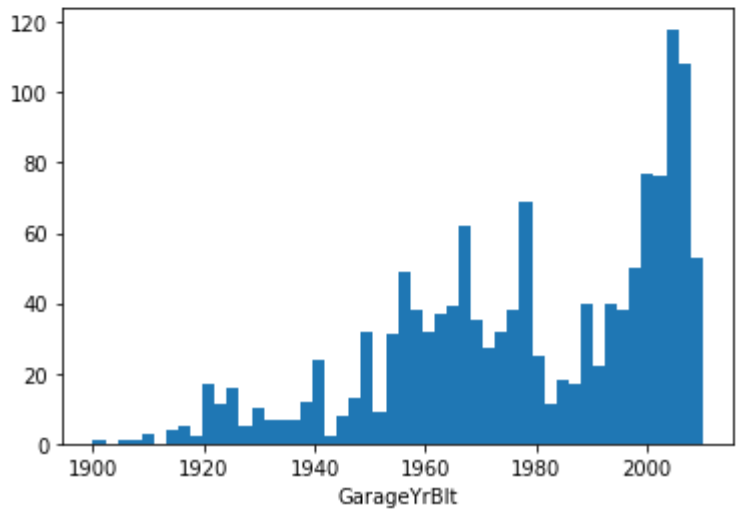
1460 rows × 3 columns

Гистограмма по признакам

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








```
1   data[data['LotFrontage'].isnull()]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Land
7	8	60	RL	NaN	10382	Pave	NaN	IR1	
12	13	20	RL	NaN	12968	Pave	NaN	IR2	
14	15	20	RL	NaN	10920	Pave	NaN	IR1	
16	17	20	RL	NaN	11241	Pave	NaN	IR1	
24	25	20	RL	NaN	8246	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	
1429	1430	20	RL	NaN	12546	Pave	NaN	IR1	
1431	1432	120	RL	NaN	4928	Pave	NaN	IR1	
1441	1442	120	RM	NaN	4426	Pave	NaN	Reg	
1443	1444	30	RL	NaN	8854	Pave	NaN	Reg	
1446	1447	20	RL	NaN	26142	Pave	NaN	IR1	

259 rows × 81 columns

Запоминание индексы строк с пустыми значениями

```
1   flt_index = data[data['LotFrontage'].isnull()].index
2   flt_index
```

```
Int64Index([   7,   12,   14,   16,   24,   31,   42,   43,   50,   64,
            ...
            1407, 1417, 1419, 1423, 1424, 1429, 1431, 1441, 1443, 1446],
           dtype='int64', length=259)
```

Проверка вывода нужных строк

```
1   data[data.index.isin(flt_index)]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Land
7	8	60	RL	NaN	10382	Pave	NaN	IR1	
12	13	20	RL	NaN	12968	Pave	NaN	IR2	
14	15	20	RL	NaN	10920	Pave	NaN	IR1	
16	17	20	RL	NaN	11241	Pave	NaN	IR1	
24	25	20	RL	NaN	8246	Pave	NaN	IR1	
...	...	...	...	...	...	...	...	...	...
1429	1430	20	RL	NaN	12546	Pave	NaN	IR1	
1431	1432	120	RL	NaN	4928	Pave	NaN	IR1	
1441	1442	120	RM	NaN	4426	Pave	NaN	Reg	
1443	1444	30	RL	NaN	8854	Pave	NaN	Reg	
1446	1447	20	RL	NaN	26142	Pave	NaN	IR1	

259 rows × 81 columns

Фильтр по колонке

```
1   data_num[data_num.index.isin(flt_index)]['LotFrontage']
```

```
7      NaN
12     NaN
14     NaN
16     NaN
24     NaN
       ..
1429   NaN
1431   NaN
1441   NaN
1443   NaN
1446   NaN
Name: LotFrontage, Length: 259, dtype: float64
```

```
1   data_num_LotFrontage = data_num[['LotFrontage']]
2   data_num_LotFrontage.head()
```

```
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_LotFrontage)
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_LotFrontage)
    return data_num_imp[mask_missing_values_only]
```

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

```
('mean',
 array([70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837, 70.04995837,
        70.04995837, 70.04995837, 70.04995837, 70.04995837]))
```

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

```
('median',
 array([69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.,
        69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69., 69.]))
```

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

```
('most_frequent',
 array([60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.,
        60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60., 60.]))
```

Задание колонки и видаимпьютации

```
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[fill
```

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

	MasVnrArea
count	1452.000000
mean	103.685262
std	181.066207
min	0.000000
25%	0.000000
50%	0.000000
75%	166.000000
max	1600.000000

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

```
('MasVnrArea', 'mean', 8, 103.68526170798899, 103.68526170798899)
```

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

```
('MasVnrArea', 'median', 8, 0.0, 0.0)
```

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

```
('MasVnrArea', 'most_frequent', 8, 0.0, 0.0)
```

1.2.2. Обработка пропусков в категориальных данных

Выбор категориальных колонок 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
```

```
Колонка Alley. Тип данных object. Количество пустых значений 1369, 93.77%.
Колонка MasVnrType. Тип данных object. Количество пустых значений 8, 0.55%.
Колонка BsmtQual. Тип данных object. Количество пустых значений 37, 2.53%.
Колонка BsmtCond. Тип данных object. Количество пустых значений 37, 2.53%.
Колонка BsmtExposure. Тип данных object. Количество пустых значений 38, 2.6%.
Колонка BsmtFinType1. Тип данных object. Количество пустых значений 37, 2.53%.
Колонка BsmtFinType2. Тип данных object. Количество пустых значений 38, 2.6%.
Колонка Electrical. Тип данных object. Количество пустых значений 1, 0.07%.
Колонка FireplaceQu. Тип данных object. Количество пустых значений 690, 47.26%.
Колонка GarageType. Тип данных object. Количество пустых значений 81, 5.55%.
Колонка GarageFinish. Тип данных object. Количество пустых значений 81, 5.55%.
Колонка GarageQual. Тип данных object. Количество пустых значений 81, 5.55%.
Колонка GarageCond. Тип данных object. Количество пустых значений 81, 5.55%.
Колонка PoolQC. Тип данных object. Количество пустых значений 1453, 99.52%.
Колонка Fence. Тип данных object. Количество пустых значений 1179, 80.75%.
Колонка MiscFeature. Тип данных object. Количество пустых значений 1406, 96.3%.
```

```
1   cat_temp_data = data[['MasVnrType']]
2   cat_temp_data.head()
```

	MasVnrType
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace

```
1   cat_temp_data['MasVnrType'].unique()
```

```
array(['BrkFace', 'None', 'Stone', 'BrkCmn', nan], dtype=object)
```

```
1   cat_temp_data[cat_temp_data['MasVnrType'].isnull()].shape
```

```
(8, 1)
```

Импьютация наиболее частыми значениями

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

```
array([['BrkFace'],
       ['None'],
       ['BrkFace'],
       ...,
       ['None'],
       ['None'],
       ['None']], dtype=object)
```

```
1   # Пустые значения отсутствуют
```

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

```
array(['BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

Импьютация константой

```
1   imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='!!!')
2   data_imp3 = imp3.fit_transform(cat_temp_data)
3   data_imp3
```

```
array([['BrkFace'],
       ['None'],
       ['BrkFace'],
       ...,
       ['None'],
       ['None'],
       ['None']], dtype=object)
```

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

```
array(['!!!', 'BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

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

```
8
```

2. Преобразование категориальных признаков в числов

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

	c1
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace
...	...
1455	None
1456	Stone
1457	None
1458	None
1459	None

1460 rows × 1 columns

2.1. Кодирование категорий целочисленными значения

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

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

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

```
array(['BrkFace', 'None', 'Stone', 'BrkCmn'], dtype=object)
```

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

```
array([0, 1, 2, 3])
```

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

```
array(['BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

2.2. Кодирование категорий наборами бинарных значен

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

```
1  cat_enc.shape
```

```
(1460, 1)
```

```
1  cat_enc_ohe.shape
```

```
(1460, 4)
```

```
1  cat_enc_ohe
```

```
<1460x4 sparse matrix of type '<class 'numpy.float64'>'
        with 1460 stored elements in Compressed Sparse Row format>
```

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

```
matrix([[0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.],
        [0., 0., 0., 1.],
        [0., 0., 1., 0.],
        [0., 0., 1., 0.]])
```

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

	c1
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace
5	None
6	Stone
7	Stone
8	None
9	None

2.3. Pandas get_dummies - быстрый вариант one-hot коди

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

	c1_BrkCmn	c1_BrkFace	c1_None	c1_Stone
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	1	0	0

```
1
2  pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

	MasVnrType_BrkCmn	MasVnrType_BrkFace	MasVnrType_None	MasVnrType_Stone	MasVnrTy
0	0	1	0	0	
1	0	0	1	0	
2	0	1	0	0	
3	0	0	1	0	
4	0	1	0	0	

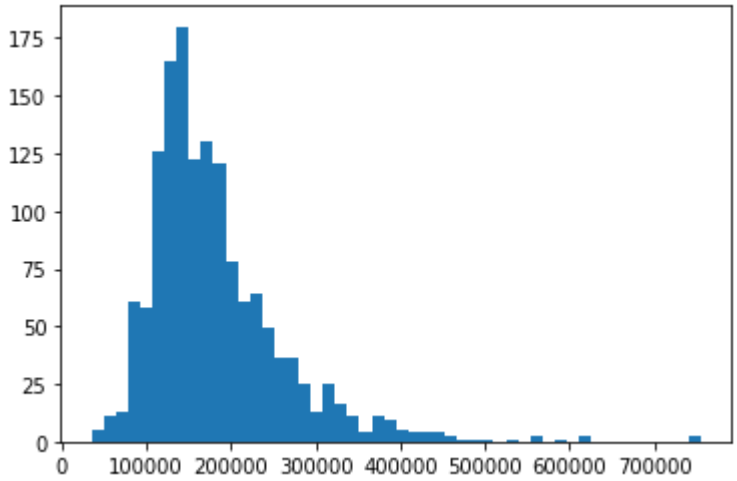
3. Масштабирование данных

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

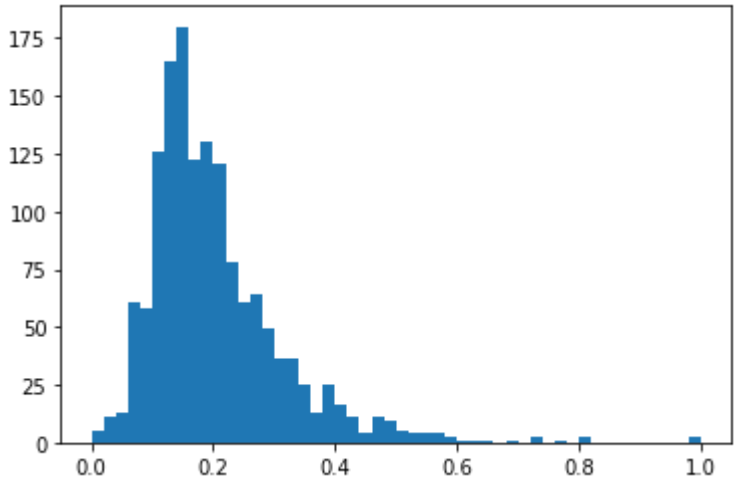
3.1. MinMax масштабирование

```
1  sc1 = MinMaxScaler()
2  sc1_data = sc1.fit_transform(data[['SalePrice']])
```

```
1  plt.hist(data['SalePrice'], 50)
2  plt.show()
```



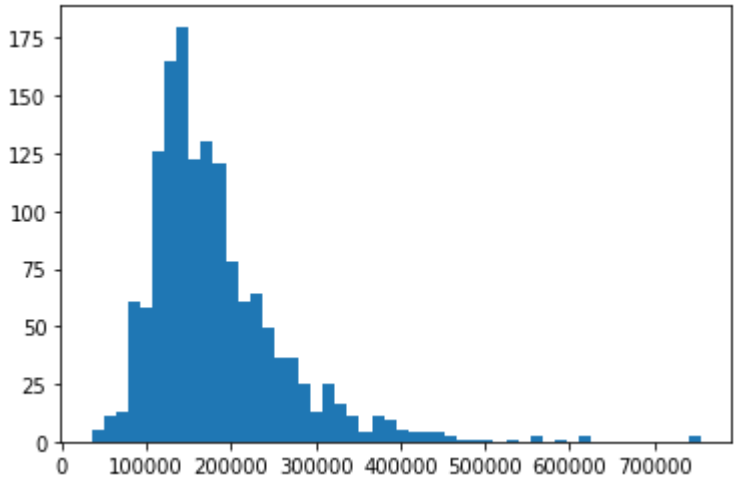

```
plt.hist(sc1_data, 50)
plt.show()
```



3.2. Масштабирование данных на основе Z-оценки - Stan

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

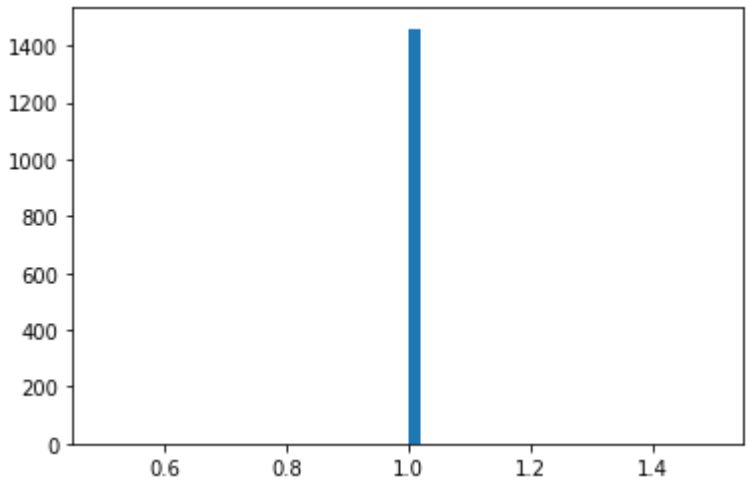
```
plt.hist(sc2_data, 50)
plt.show()
```



3.3. Нормализация данных

```
1   sc3 = Normalizer()
2   sc3_data = sc3.fit_transform(data[['SalePrice']])
```

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



Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // GitHub. – 2019. – Режим досту- па: https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION (дата обращения: 15.02.2020).

[2] Scikit-learn. Boston house-prices dataset [Electronic resource] // Scikit-learn. — 2018. — Access mode: https://scikit- learn.org/0.20/modules/generated/sklearn.datasets.load_boston.html (online;accessed: 18.02.2020).

[3] 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: 20.02.2020).

[4] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: https://seaborn.pydata.org/(online; accessed: 20.02.2020).

[5] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/(online; accessed: 20.02.2020).