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

Лабораторная работа №1
по дисциплине
«Методы машинного обучения»
на тему
«Разведочный анализ данных. Исследование и визуализация данных»

Выполнил:
студент группы ИУ5-21М
Якубов А. Р.

Москва — 2019 г.

1. Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных. [1]

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style='ticks')

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

```
<IPython.core.display.HTML object>
```

1.1. Загрузка и первичный анализ данных

In [35]:
```
# Используем данные из соревнования House Prices: Advanced Regressior
data = pd.read_csv('data/train.csv', sep=",")
```

In [36]:
```
# Первые 5 строк набора
data.head()
```

Out[36]:

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

	LandContour	Utilities	…	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	Lvl	AllPub	…	0	NaN	NaN	NaN	0
1	Lvl	AllPub	…	0	NaN	NaN	NaN	0
2	Lvl	AllPub	…	0	NaN	NaN	NaN	0
3	Lvl	AllPub	…	0	NaN	NaN	NaN	0
4	Lvl	AllPub	…	0	NaN	NaN	NaN	0

	YrSold	SaleType	SaleCondition	SalePrice
0	2008	WD	Normal	208500
1	2007	WD	Normal	181500
2	2008	WD	Normal	223500
3	2006	WD	Abnorml	140000
4	2008	WD	Normal	250000

[5 rows x 81 columns]

In [37]:
```
# Размер набора данных
data.shape
```

Out[37]: (1460, 81)

```
In [38]: # Типы колонок
         data.dtypes
```

```
Out[38]: Id                 int64
         MSSubClass         int64
         MSZoning          object
         LotFrontage      float64
         LotArea            int64
         Street            object
         Alley             object
         LotShape          object
         LandContour       object
         Utilities         object
         LotConfig         object
         LandSlope         object
         Neighborhood      object
         Condition1        object
         Condition2        object
         BldgType          object
         HouseStyle        object
         OverallQual        int64
         OverallCond        int64
         YearBuilt          int64
         YearRemodAdd       int64
         RoofStyle         object
         RoofMatl          object
         Exterior1st       object
         Exterior2nd       object
         MasVnrType        object
         MasVnrArea       float64
         ExterQual         object
         ExterCond         object
         Foundation        object
                            ...
         BedroomAbvGr       int64
         KitchenAbvGr       int64
         KitchenQual       object
         TotRmsAbvGrd       int64
         Functional        object
         Fireplaces         int64
         FireplaceQu       object
         GarageType        object
         GarageYrBlt      float64
         GarageFinish      object
         GarageCars         int64
         GarageArea         int64
         GarageQual        object
         GarageCond        object
         PavedDrive        object
         WoodDeckSF         int64
         OpenPorchSF        int64
```

```
EnclosedPorch       int64
3SsnPorch           int64
ScreenPorch         int64
PoolArea            int64
PoolQC             object
Fence              object
MiscFeature        object
MiscVal             int64
MoSold              int64
YrSold              int64
SaleType           object
SaleCondition      object
SalePrice           int64
Length: 81, dtype: object
```

In [39]:
```
# Количество пропущенных значений в каждой колонке
data.isnull().sum()
```

Out[39]:
```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
Street             0
Alley           1369
LotShape           0
LandContour        0
Utilities          0
LotConfig          0
LandSlope          0
Neighborhood       0
Condition1         0
Condition2         0
BldgType           0
HouseStyle         0
OverallQual        0
OverallCond        0
YearBuilt          0
YearRemodAdd       0
RoofStyle          0
RoofMatl           0
Exterior1st        0
Exterior2nd        0
MasVnrType         8
MasVnrArea         8
ExterQual          0
ExterCond          0
Foundation         0
                ...
BedroomAbvGr       0
KitchenAbvGr       0
```

```
KitchenQual         0
TotRmsAbvGrd        0
Functional          0
Fireplaces          0
FireplaceQu       690
GarageType         81
GarageYrBlt        81
GarageFinish       81
GarageCars          0
GarageArea          0
GarageQual         81
GarageCond         81
PavedDrive          0
WoodDeckSF          0
OpenPorchSF         0
EnclosedPorch       0
3SsnPorch           0
ScreenPorch         0
PoolArea            0
PoolQC           1453
Fence            1179
MiscFeature      1406
MiscVal             0
MoSold              0
YrSold              0
SaleType            0
SaleCondition       0
SalePrice           0
Length: 81, dtype: int64
```

```
In [40]: total_count = data.shape[0]
         print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 1460
```

1.2. 1. Обработка пропусков в данных

1.2.1. 1.1. Простые стратегии - удаление или заполнение нулями

```
In [41]: # Удаление колонок, содержащих пустые значения
         data_new_1 = data.dropna(axis=1, how='any')
         (data.shape, data_new_1.shape)
```

```
Out[41]: ((1460, 81), (1460, 62))
```

```
In [42]: # Удаление строк, содержащих пустые значения
         data_new_2 = data.dropna(axis=0, how='any')
         (data.shape, data_new_2.shape)
```

```
Out[42]: ((1460, 81), (0, 81))
```

```
In [43]: data.head()
```

```
Out[43]:    Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShap
         0   1          60       RL         65.0     8450   Pave   NaN      Re
         1   2          20       RL         80.0     9600   Pave   NaN      Re
         2   3          60       RL         68.0    11250   Pave   NaN      IR
         3   4          70       RL         60.0     9550   Pave   NaN      IR
         4   5          60       RL         84.0    14260   Pave   NaN      IR

           LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal
         0         Lvl    AllPub  …        0    NaN   NaN         NaN       0
         1         Lvl    AllPub  …        0    NaN   NaN         NaN       0
         2         Lvl    AllPub  …        0    NaN   NaN         NaN       0
         3         Lvl    AllPub  …        0    NaN   NaN         NaN       0
         4         Lvl    AllPub  …        0    NaN   NaN         NaN       0

           YrSold  SaleType  SaleCondition  SalePrice
         0   2008        WD         Normal     208500
         1   2007        WD         Normal     181500
         2   2008        WD         Normal     223500
         3   2006        WD        Abnorml     140000
         4   2008        WD         Normal     250000

         [5 rows x 81 columns]
```

```
In [44]: # Заполнение всех пропущенных значений нулями
         # В данном случае это некорректно, так как нулями заполняются в том ч
         data_new_3 = data.fillna(0)
         data_new_3.head()
```

```
Out[44]:    Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShap
         0   1          60       RL         65.0     8450   Pave     0      Re
         1   2          20       RL         80.0     9600   Pave     0      Re
         2   3          60       RL         68.0    11250   Pave     0      IR
         3   4          70       RL         60.0     9550   Pave     0      IR
         4   5          60       RL         84.0    14260   Pave     0      IR

           LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscVal
         0         Lvl    AllPub  …        0      0     0           0       0
         1         Lvl    AllPub  …        0      0     0           0       0
         2         Lvl    AllPub  …        0      0     0           0       0
         3         Lvl    AllPub  …        0      0     0           0       0
         4         Lvl    AllPub  …        0      0     0           0       0

           YrSold  SaleType  SaleCondition  SalePrice
         0   2008        WD         Normal     208500
         1   2007        WD         Normal     181500
         2   2008        WD         Normal     223500
         3   2006        WD        Abnorml     140000
         4   2008        WD         Normal     250000

         [5 rows x 81 columns]
```

1.2.2. 1.2. “Внедрение значений” - импьютация (imputation)

1.2.1. Обработка пропусков в числовых данных

```
In [45]: # 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
```

```
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%.
```

```
In [46]: # Фильтр по колонкам с пропущенными значениями
         data_num = data[num_cols]
         data_num
```

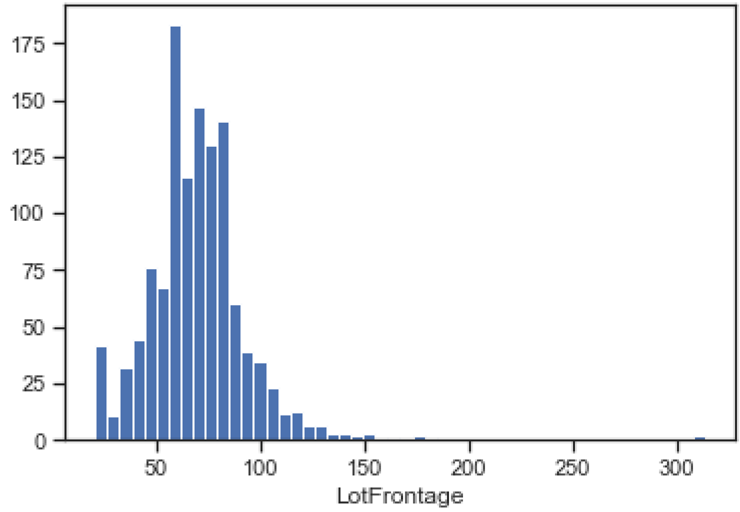
Out[46]:

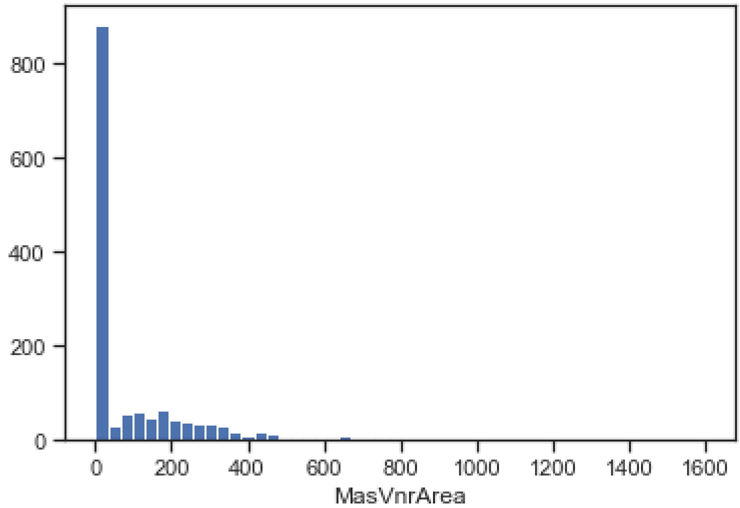
	LotFrontage	MasVnrArea	GarageYrBlt
0	65.0	196.0	2003.0
1	80.0	0.0	1976.0
2	68.0	162.0	2001.0
3	60.0	0.0	1998.0
4	84.0	350.0	2000.0
5	85.0	0.0	1993.0
6	75.0	186.0	2004.0
7	NaN	240.0	1973.0
8	51.0	0.0	1931.0
9	50.0	0.0	1939.0
10	70.0	0.0	1965.0
11	85.0	286.0	2005.0
12	NaN	0.0	1962.0
13	91.0	306.0	2006.0
14	NaN	212.0	1960.0
15	51.0	0.0	1991.0
16	NaN	180.0	1970.0
17	72.0	0.0	1967.0
18	66.0	0.0	2004.0
19	70.0	0.0	1958.0
20	101.0	380.0	2005.0
21	57.0	0.0	1930.0
22	75.0	281.0	2002.0
23	44.0	0.0	1976.0
24	NaN	0.0	1968.0
25	110.0	640.0	2007.0

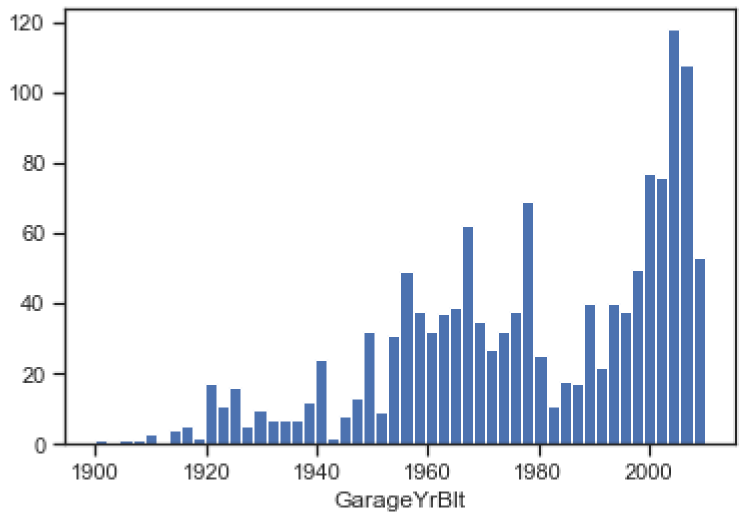
```
26          60.0      0.0    2005.0
27          98.0    200.0    2008.0
28          47.0      0.0    1957.0
29          60.0      0.0    1920.0
…           …       …       …
1430        60.0      0.0    2005.0
1431         NaN      0.0    1976.0
1432        60.0      0.0    1928.0
1433        93.0    318.0    2000.0
1434        80.0      0.0    1977.0
1435        80.0    237.0    1962.0
1436        60.0      0.0    1974.0
1437        96.0    426.0    2008.0
1438        90.0      0.0    1957.0
1439        80.0     96.0    1979.0
1440        79.0      0.0    1993.0
1441         NaN    147.0    2004.0
1442        85.0    160.0    2008.0
1443         NaN      0.0    1916.0
1444        63.0    106.0    2004.0
1445        70.0      0.0    1990.0
1446         NaN    189.0    1962.0
1447        80.0    438.0    1995.0
1448        70.0      0.0    1950.0
1449        21.0      0.0       NaN
1450        60.0      0.0       NaN
1451        78.0    194.0    2008.0
1452        35.0     80.0    2005.0
1453        90.0      0.0       NaN
1454        62.0      0.0    2004.0
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 x 3 columns]
```

```
In [47]: # Гистограмма по признакам
         for col in data_num:
             plt.hist(data[col], 50)
             plt.xlabel(col)
             plt.show()
```

```
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-pac
  keep = (tmp_a >= mn)
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-pac
  keep &= (tmp_a <= mx)
```


175
150
125
100
75
50
25
0
50
100
150
200
250
300
LotFrontage


800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600
MasVnrArea


120
100
80
60
40
20
0
1900
1920
1940
1960
1980
2000
GarageYrBlt

```
In [48]: # Фильтр по пустым значениям поля MasVnrArea
         data[data['MasVnrArea'].isnull()]

Out[48]:         Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley Lo
         234    235          60       RL          NaN     7851   Pave   NaN
         529    530          20       RL          NaN    32668   Pave   NaN
         650    651          60       FV         65.0     8125   Pave   NaN
         936    937          20       RL         67.0    10083   Pave   NaN
         973    974          20       FV         95.0    11639   Pave   NaN
         977    978         120       FV         35.0     4274   Pave  Pave
         1243  1244          20       RL        107.0    13891   Pave   NaN
         1278  1279          60       RL         75.0     9473   Pave   NaN

              LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscV
         234          Lvl    AllPub  …        0    NaN   NaN         NaN
         529          Lvl    AllPub  …        0    NaN   NaN         NaN
         650          Lvl    AllPub  …        0    NaN   NaN         NaN
         936          Lvl    AllPub  …        0    NaN   NaN         NaN
         973          Lvl    AllPub  …        0    NaN   NaN         NaN
         977          Lvl    AllPub  …        0    NaN   NaN         NaN
         1243         Lvl    AllPub  …        0    NaN   NaN         NaN
         1278         Lvl    AllPub  …        0    NaN   NaN         NaN

              MoSold YrSold  SaleType  SaleCondition  SalePrice
         234       5   2010        WD         Normal     216500
         529       3   2007        WD         Alloca     200624
         650       5   2008        WD         Normal     205950
         936       8   2009        WD         Normal     184900
         973      12   2008       New        Partial     182000
         977      11   2007       New        Partial     199900
         1243      9   2006       New        Partial     465000
         1278      3   2008        WD         Normal     237000

         [8 rows x 81 columns]

In [49]: # Запоминаем индексы строк с пустыми значениями
         flt_index = data[data['MasVnrArea'].isnull()].index
         flt_index

Out[49]: Int64Index([234, 529, 650, 936, 973, 977, 1243, 1278], dtype='int64')

In [50]: # Проверяем что выводятся нужные строки
         data[data.index.isin(flt_index)]

Out[50]:         Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley Lo
         234    235          60       RL          NaN     7851   Pave   NaN
         529    530          20       RL          NaN    32668   Pave   NaN
         650    651          60       FV         65.0     8125   Pave   NaN
```

```
936    937          20       RL        67.0    10083    Pave    NaN
973    974          20       FV        95.0    11639    Pave    NaN
977    978         120       FV        35.0     4274    Pave   Pave
1243   1244         20       RL       107.0    13891    Pave    NaN
1278   1279         60       RL        75.0     9473    Pave    NaN

     LandContour Utilities  … PoolArea PoolQC Fence MiscFeature MiscV
234          Lvl    AllPub  …        0    NaN   NaN         NaN
529          Lvl    AllPub  …        0    NaN   NaN         NaN
650          Lvl    AllPub  …        0    NaN   NaN         NaN
936          Lvl    AllPub  …        0    NaN   NaN         NaN
973          Lvl    AllPub  …        0    NaN   NaN         NaN
977          Lvl    AllPub  …        0    NaN   NaN         NaN
1243         Lvl    AllPub  …        0    NaN   NaN         NaN
1278         Lvl    AllPub  …        0    NaN   NaN         NaN

     MoSold YrSold  SaleType  SaleCondition  SalePrice
234       5   2010        WD         Normal     216500
529       3   2007        WD         Alloca     200624
650       5   2008        WD         Normal     205950
936       8   2009        WD         Normal     184900
973      12   2008       New        Partial     182000
977      11   2007       New        Partial     199900
1243      9   2006       New        Partial     465000
1278      3   2008        WD         Normal     237000

[8 rows x 81 columns]
```

```
In [51]: # Фильтр по колонке
         data_num[data_num.index.isin(flt_index)]['MasVnrArea']
```

```
Out[51]: 234    NaN
         529    NaN
         650    NaN
         936    NaN
         973    NaN
         977    NaN
         1243   NaN
         1278   NaN
         Name: MasVnrArea, dtype: float64
```

Будем использовать встроенные средства импьютации библиотеки scikit-learn - https://scikit-learn.org/stable/modules/impute.html#impute

```
In [52]: data_num_MasVnrArea = data_num[['MasVnrArea']]
         data_num_MasVnrArea.head()
```

```
Out[52]:    MasVnrArea
         0       196.0
         1         0.0
         2       162.0
         3         0.0
         4       350.0
```

```
In [53]: from sklearn.impute import SimpleImputer
         from sklearn.impute import MissingIndicator

In [54]: # Фильтр для проверки заполнения пустых значений
         indicator = MissingIndicator()
         mask_missing_values_only = indicator.fit_transform(data_num_MasVnrAre
         mask_missing_values_only

Out[54]: array([[False],
                [False],
                [False],
                …,
                [False],
                [False],
                [False]], dtype=bool)
```

С помощью класса SimpleImputer можно проводить импьютацию различными показателями центра распределения

```
In [55]: strategies=['mean', 'median','most_frequent']

In [56]: def test_num_impute(strategy_param):
             imp_num = SimpleImputer(strategy=strategy_param)
             data_num_imp = imp_num.fit_transform(data_num_MasVnrArea)
             return data_num_imp[mask_missing_values_only]

In [57]: for strat in strategies:
             print(strat, test_num_impute(strat))

mean [ 103.68526171  103.68526171  103.68526171  103.68526171  103.68526171
  103.68526171  103.68526171  103.68526171]
median [ 0.   0.   0.   0.   0.   0.   0.   0.]
most_frequent [ 0.   0.   0.   0.   0.   0.   0.   0.]


In [58]: # Более сложная функция, которая позволяет задавать колонку и вид имп
         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],

In [59]: data[['GarageYrBlt']].describe()
```

```
Out[59]:        GarageYrBlt
         count  1379.000000
         mean   1978.506164
         std      24.689725
         min    1900.000000
         25%    1961.000000
         50%    1980.000000
         75%    2002.000000
         max    2010.000000
```

```
In [60]: for strat in strategies:
             print(test_num_impute_col(data, 'GarageYrBlt', strat))

('GarageYrBlt', 'mean', 81, 1978.5061638868744, 1978.5061638868744)
('GarageYrBlt', 'median', 81, 1980.0, 1980.0)
('GarageYrBlt', 'most_frequent', 81, 2005.0, 2005.0)
```

1.2.2. Обработка пропусков в категориальных данных

```
In [61]: # 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

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%
```

Класс SimpleImputer можно использовать для категориальных признаков со стратегиями “most_frequent” или “constant”.

```
In [62]: cat_temp_data = data[['MasVnrType']]
         cat_temp_data.head()

Out[62]:   MasVnrType
         0    BrkFace
         1       None
         2    BrkFace
         3       None
         4    BrkFace

In [63]: cat_temp_data['MasVnrType'].unique()

Out[63]: array(['BrkFace', 'None', 'Stone', 'BrkCmn', nan], dtype=object)

In [64]: cat_temp_data[cat_temp_data['MasVnrType'].isnull()].shape

Out[64]: (8, 1)

In [65]: # Импьютация наиболее частыми значениями
         imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
         data_imp2 = imp2.fit_transform(cat_temp_data)
         data_imp2

Out[65]: array([['BrkFace'],
                ['None'],
                ['BrkFace'],
                …,
                ['None'],
                ['None'],
                ['None']], dtype=object)

In [66]: # Пустые значения отсутствуют
         np.unique(data_imp2)

Out[66]: array(['BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)

In [67]: # Импьютация константой
         imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill
         data_imp3 = imp3.fit_transform(cat_temp_data)
         data_imp3

Out[67]: array([['BrkFace'],
                ['None'],
                ['BrkFace'],
                …,
                ['None'],
                ['None'],
                ['None']], dtype=object)

In [68]: np.unique(data_imp3)

Out[68]: array(['!!!', 'BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)

In [69]: data_imp3[data_imp3=='!!!'].size

Out[69]: 8
```

1.3. 2. Преобразование категориальных признаков в числовые

```
In [70]: cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
         cat_enc
```

Out[70]:

	c1
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace
5	None
6	Stone
7	Stone
8	None
9	None
10	None
11	Stone
12	None
13	Stone
14	BrkFace
15	None
16	BrkFace
17	None
18	None
19	None
20	BrkFace
21	None
22	BrkFace
23	None
24	None
25	Stone
26	None
27	Stone
28	None
29	None
…	…
1430	None
1431	None
1432	None
1433	BrkFace
1434	None
1435	BrkFace
1436	None
1437	Stone
1438	None
1439	BrkFace
1440	None
1441	BrkFace
1442	Stone
1443	None

```
1444   BrkFace
1445      None
1446   BrkFace
1447   BrkFace
1448      None
1449      None
1450      None
1451     Stone
1452   BrkFace
1453      None
1454      None
1455      None
1456     Stone
1457      None
1458      None
1459      None

[1460 rows x 1 columns]
```

1.3.1. 2.1. Кодирование категорий целочисленными значениями - label encoding

```
In [71]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder

In [72]: le = LabelEncoder()
         cat_enc_le = le.fit_transform(cat_enc['c1'])

In [73]: cat_enc['c1'].unique()

Out[73]: array(['BrkFace', 'None', 'Stone', 'BrkCmn'], dtype=object)

In [74]: np.unique(cat_enc_le)

Out[74]: array([0, 1, 2, 3])

In [75]: le.inverse_transform([0, 1, 2, 3])

Out[75]: array(['BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

1.3.2. 2.2. Кодирование категорий наборами бинарных значений - one-hot encoding

```
In [76]: ohe = OneHotEncoder()
         cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])

In [77]: cat_enc.shape

Out[77]: (1460, 1)

In [78]: cat_enc_ohe.shape

Out[78]: (1460, 4)

In [79]: cat_enc_ohe
```

```
Out[79]: <1460x4 sparse matrix of type '<class 'numpy.float64'>'
                 with 1460 stored elements in Compressed Sparse Row format>
```

```
In [80]: cat_enc_ohe.todense()[0:10]
```

```
Out[80]: 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.]])
```

```
In [81]: cat_enc.head(10)
```

```
Out[81]:        c1
         0  BrkFace
         1     None
         2  BrkFace
         3     None
         4  BrkFace
         5     None
         6    Stone
         7    Stone
         8     None
         9     None
```

1.3.3. 2.3. Pandas get_dummies - быстрый вариант one-hot кодирования

```
In [82]: pd.get_dummies(cat_enc).head()
```

```
Out[82]:    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
```

```
In [83]: pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

```
Out[83]:    MasVnrType_BrkCmn  MasVnrType_BrkFace  MasVnrType_None  MasVnrType
         0                  0                   1                0
         1                  0                   0                1
         2                  0                   1                0
         3                  0                   0                1
         4                  0                   1                0

            MasVnrType_nan
         0               0
```

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

1.4. 3. Масштабирование данных

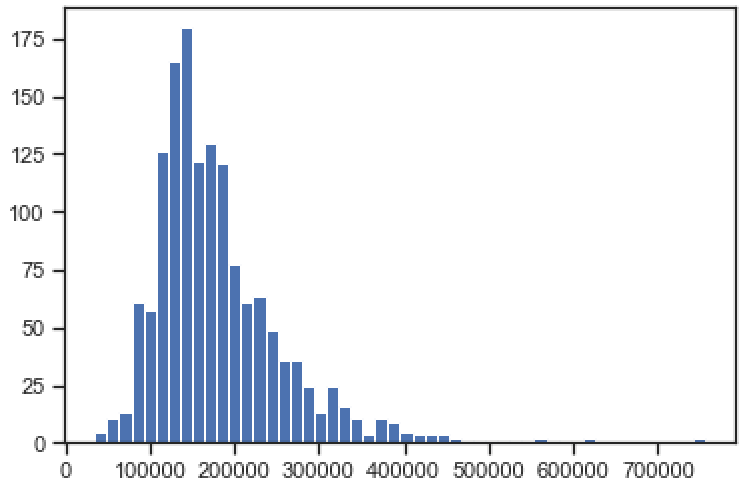
```
In [84]: from sklearn.preprocessing import MinMaxScaler, StandardScaler, Norma
```

1.4.1. 3.1. MinMax масштабирование

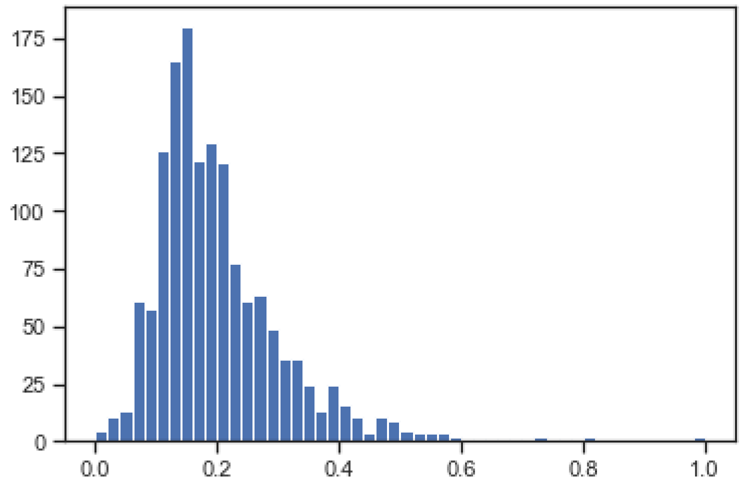
```
In [85]: sc1 = MinMaxScaler()
         sc1_data = sc1.fit_transform(data[['SalePrice']])
```

```
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-pac
  return self.partial_fit(X, y)
```

```
In [86]: plt.hist(data['SalePrice'], 50)
         plt.show()
```



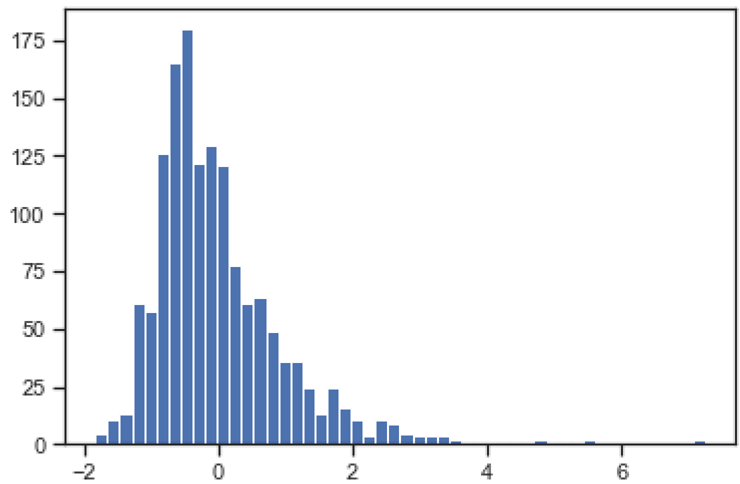

```
In [87]: plt.hist(sc1_data, 50)
         plt.show()
```



1.4.2. 3.2. Масштабирование данных на основе Z-оценки - StandardScaler

```
In [88]: sc2 = StandardScaler()
         sc2_data = sc2.fit_transform(data[['SalePrice']])

c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-pac
  return self.partial_fit(X, y)
c:\program files (x86)\microsoft visual studio\shared\python36_64\lib\site-pac
  return self.fit(X, **fit_params).transform(X)
```

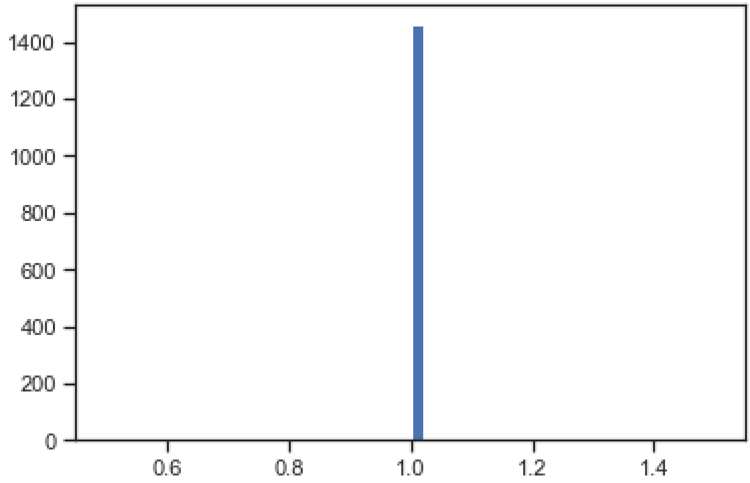
```
In [89]: plt.hist(sc2_data, 50)
         plt.show()
```



1.4.3. 3.3. Нормализация данных

```
In [90]: sc3 = Normalizer()
         sc3_data = sc3.fit_transform(data[['SalePrice']])
```

```
In [91]: plt.hist(sc3_data, 50)
         plt.show()
```



Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: `https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION` (дата обращения: 13.02.2019).