МГТУ им. Н. Э. Баумана, кафедра ИУ5

курс “Методы машинного обучения”

Лабораторная работа №2

«Обработка признаков (часть 1)»

ВЫПОЛНИЛ:

Акушко А.С.

Группа: ИУ5-21М

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2022

Задание:

- Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.
- Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  1. устранение пропусков в данных;
  2. кодирование категориальных признаков;
  3. нормализацию числовых признаков.

• Сформировать отчет и разместить его в своем репозитории на github.

Выполнение работы:

Импортирование необходимых библиотек

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [68]:
```
data = pd.read_csv("/content/drive/MyDrive/data/house_sales.csv")
```

In [69]:
```
data = data.drop('Id', 1)
data.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:   FutureWar
ning: In a future version of pandas all arguments of DataFrame.drop excep
t for the argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

Out[69]:

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

5 rows × 80 columns

In [33]:
```
data_features = list(zip(
# признаки
[i for i in data.columns],
zip(
    # типы колонок
    [str(i) for i in data.dtypes],
    # проверим есть ли пропущенные значения
    [i for i in data.isnull().sum()]
)))
# Признаки с типом данных и количеством пропусков
data_features
```

Out[33]:
```
[('MSSubClass', ('int64', 0)),
 ('MSZoning', ('object', 0)),
 ('LotFrontage', ('float64', 259)),
 ('LotArea', ('int64', 0)),
 ('Street', ('object', 0)),
 ('Alley', ('object', 1369)),
 ('LotShape', ('object', 0)),
 ('LandContour', ('object', 0)),
 ('Utilities', ('object', 0)),
```

```
('LotConfig', ('object', 0)),
('LandSlope', ('object', 0)),
('Neighborhood', ('object', 0)),
('Condition1', ('object', 0)),
('Condition2', ('object', 0)),
('BldgType', ('object', 0)),
('HouseStyle', ('object', 0)),
('OverallQual', ('int64', 0)),
('OverallCond', ('int64', 0)),
('YearBuilt', ('int64', 0)),
('YearRemodAdd', ('int64', 0)),
('RoofStyle', ('object', 0)),
('RoofMatl', ('object', 0)),
('Exterior1st', ('object', 0)),
('Exterior2nd', ('object', 0)),
('MasVnrType', ('object', 8)),
('MasVnrArea', ('float64', 8)),
('ExterQual', ('object', 0)),
('ExterCond', ('object', 0)),
('Foundation', ('object', 0)),
('BsmtQual', ('object', 37)),
('BsmtCond', ('object', 37)),
('BsmtExposure', ('object', 38)),
('BsmtFinType1', ('object', 37)),
('BsmtFinSF1', ('int64', 0)),
('BsmtFinType2', ('object', 38)),
('BsmtFinSF2', ('int64', 0)),
('BsmtUnfSF', ('int64', 0)),
('TotalBsmtSF', ('int64', 0)),
('Heating', ('object', 0)),
('HeatingQC', ('object', 0)),
('CentralAir', ('object', 0)),
('Electrical', ('object', 1)),
('1stFlrSF', ('int64', 0)),
('2ndFlrSF', ('int64', 0)),
('LowQualFinSF', ('int64', 0)),
('GrLivArea', ('int64', 0)),
('BsmtFullBath', ('int64', 0)),
('BsmtHalfBath', ('int64', 0)),
('FullBath', ('int64', 0)),
('HalfBath', ('int64', 0)),
('BedroomAbvGr', ('int64', 0)),
('KitchenAbvGr', ('int64', 0)),
('KitchenQual', ('object', 0)),
('TotRmsAbvGrd', ('int64', 0)),
('Functional', ('object', 0)),
('Fireplaces', ('int64', 0)),
('FireplaceQu', ('object', 690)),
('GarageType', ('object', 81)),
('GarageYrBlt', ('float64', 81)),
('GarageFinish', ('object', 81)),
('GarageCars', ('int64', 0)),
('GarageArea', ('int64', 0)),
('GarageQual', ('object', 81)),
('GarageCond', ('object', 81)),
('PavedDrive', ('object', 0)),
('WoodDeckSF', ('int64', 0)),
('OpenPorchSF', ('int64', 0)),
('EnclosedPorch', ('int64', 0)),
('3SsnPorch', ('int64', 0)),
('ScreenPorch', ('int64', 0)),
('PoolArea', ('int64', 0)),
('PoolQC', ('object', 1453)),
```

```
 ('Fence', ('object', 1179)),
 ('MiscFeature', ('object', 1406)),
 ('MiscVal', ('int64', 0)),
 ('MoSold', ('int64', 0)),
 ('YrSold', ('int64', 0)),
 ('SaleType', ('object', 0)),
 ('SaleCondition', ('object', 0)),
 ('SalePrice', ('int64', 0))]
```

Устранение пропусков

In [34]:

```
# Доля (процент) пропусков
[(c, data[c].isnull().mean()) for c in data.columns]
```

Out[34]:

```
[('MSSubClass', 0.0),
 ('MSZoning', 0.0),
 ('LotFrontage', 0.1773972602739726),
 ('LotArea', 0.0),
 ('Street', 0.0),
 ('Alley', 0.9376712328767123),
 ('LotShape', 0.0),
 ('LandContour', 0.0),
 ('Utilities', 0.0),
 ('LotConfig', 0.0),
 ('LandSlope', 0.0),
 ('Neighborhood', 0.0),
 ('Condition1', 0.0),
 ('Condition2', 0.0),
 ('BldgType', 0.0),
 ('HouseStyle', 0.0),
 ('OverallQual', 0.0),
 ('OverallCond', 0.0),
 ('YearBuilt', 0.0),
 ('YearRemodAdd', 0.0),
 ('RoofStyle', 0.0),
 ('RoofMatl', 0.0),
 ('Exterior1st', 0.0),
 ('Exterior2nd', 0.0),
 ('MasVnrType', 0.005479452054794521),
 ('MasVnrArea', 0.005479452054794521),
 ('ExterQual', 0.0),
 ('ExterCond', 0.0),
 ('Foundation', 0.0),
 ('BsmtQual', 0.025342465753424658),
 ('BsmtCond', 0.025342465753424658),
 ('BsmtExposure', 0.026027397260273973),
 ('BsmtFinType1', 0.025342465753424658),
 ('BsmtFinSF1', 0.0),
 ('BsmtFinType2', 0.026027397260273973),
 ('BsmtFinSF2', 0.0),
 ('BsmtUnfSF', 0.0),
 ('TotalBsmtSF', 0.0),
 ('Heating', 0.0),
 ('HeatingQC', 0.0),
 ('CentralAir', 0.0),
 ('Electrical', 0.0006849315068493151),
 ('1stFlrSF', 0.0),
 ('2ndFlrSF', 0.0),
 ('LowQualFinSF', 0.0),
 ('GrLivArea', 0.0),
 ('BsmtFullBath', 0.0),
```

```
('BsmtHalfBath', 0.0),
('FullBath', 0.0),
('HalfBath', 0.0),
('BedroomAbvGr', 0.0),
('KitchenAbvGr', 0.0),
('KitchenQual', 0.0),
('TotRmsAbvGrd', 0.0),
('Functional', 0.0),
('Fireplaces', 0.0),
('FireplaceQu', 0.4726027397260274),
('GarageType', 0.05547945205479452),
('GarageYrBlt', 0.05547945205479452),
('GarageFinish', 0.05547945205479452),
('GarageCars', 0.0),
('GarageArea', 0.0),
('GarageQual', 0.05547945205479452),
('GarageCond', 0.05547945205479452),
('PavedDrive', 0.0),
('WoodDeckSF', 0.0),
('OpenPorchSF', 0.0),
('EnclosedPorch', 0.0),
('3SsnPorch', 0.0),
('ScreenPorch', 0.0),
('PoolArea', 0.0),
('PoolQC', 0.9952054794520548),
('Fence', 0.8075342465753425),
('MiscFeature', 0.963013698630137),
('MiscVal', 0.0),
('MoSold', 0.0),
('YrSold', 0.0),
('SaleType', 0.0),
('SaleCondition', 0.0),
('SalePrice', 0.0)]
```

In [35]:

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

Out[35]:

	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConfig
0	60	RL	8450	Pave	Reg	Lvl	AllPub	Inside
1	20	RL	9600	Pave	Reg	Lvl	AllPub	FR2
2	60	RL	11250	Pave	IR1	Lvl	AllPub	Inside
3	70	RL	9550	Pave	IR1	Lvl	AllPub	Corner
4	60	RL	14260	Pave	IR1	Lvl	AllPub	FR2
...	...	...	...	...	...	...	...	...
1455	60	RL	7917	Pave	Reg	Lvl	AllPub	Inside
1456	20	RL	13175	Pave	Reg	Lvl	AllPub	Inside
1457	70	RL	9042	Pave	Reg	Lvl	AllPub	Inside
1458	20	RL	9717	Pave	Reg	Lvl	AllPub	Inside
1459	20	RL	9937	Pave	Reg	Lvl	AllPub	Inside

1460 rows × 61 columns

In [36]:

```
# Удаление колонок, содержащих пустые значения
```

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

Out[36]:

	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConfig
0	60	RL	8450	Pave	Reg	Lvl	AllPub	Inside
1	20	RL	9600	Pave	Reg	Lvl	AllPub	FR2
2	60	RL	11250	Pave	IR1	Lvl	AllPub	Inside
3	70	RL	9550	Pave	IR1	Lvl	AllPub	Corner
4	60	RL	14260	Pave	IR1	Lvl	AllPub	FR2
...	...	...	...	...	...	...	...	...
1455	60	RL	7917	Pave	Reg	Lvl	AllPub	Inside
1456	20	RL	13175	Pave	Reg	Lvl	AllPub	Inside
1457	70	RL	9042	Pave	Reg	Lvl	AllPub	Inside
1458	20	RL	9717	Pave	Reg	Lvl	AllPub	Inside
1459	20	RL	9937	Pave	Reg	Lvl	AllPub	Inside

1460 rows × 61 columns

In [37]:

```
# Удаление колонок с высоким процентом пропусков (более 50%)
data.dropna(axis=1, thresh=730)
```

Out[37]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utiliti
0	60	RL	65.0	8450	Pave	Reg	Lvl	AllP
1	20	RL	80.0	9600	Pave	Reg	Lvl	AllP
2	60	RL	68.0	11250	Pave	IR1	Lvl	AllP
3	70	RL	60.0	9550	Pave	IR1	Lvl	AllP
4	60	RL	84.0	14260	Pave	IR1	Lvl	AllP
...	...	...	...	...	...	...	...	
1455	60	RL	62.0	7917	Pave	Reg	Lvl	AllP
1456	20	RL	85.0	13175	Pave	Reg	Lvl	AllP
1457	70	RL	66.0	9042	Pave	Reg	Lvl	AllP
1458	20	RL	68.0	9717	Pave	Reg	Lvl	AllP
1459	20	RL	75.0	9937	Pave	Reg	Lvl	AllP

1460 rows × 76 columns

In [38]:

```
# Заполним пропуски возраста средними значениями
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'LotFrontage', data['LotFrontage'].mean())
```

In [41]:

```
# Убедимся, что признак LotFrontage не имеет пустых значений
data.isnull().sum()
```

Out[41]:
```
MSSubClass        0
MSZoning          0
LotFrontage       0
LotArea           0
Street            0
                 ..
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 80, dtype: int64
```

Кодирование категориальных признаков

In [42]:
```
from sklearn.preprocessing import LabelEncoder
```

In [43]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(data['SaleCondition'])
```

In [44]:
```
data['SaleCondition'].unique()
```

Out[44]:
```
array(['Normal', 'Abnorml', 'Partial', 'AdjLand', 'Alloca', 'Family'],
      dtype=object)
```

In [45]:
```
np.unique(cat_enc_le)
```

Out[45]:
```
array([0, 1, 2, 3, 4, 5])
```

In [46]:
```
le.inverse_transform([0, 1, 2, 3, 4, 5])
```

Out[46]:
```
array(['Abnorml', 'AdjLand', 'Alloca', 'Family', 'Normal', 'Partial'],
      dtype=object)
```

In [47]:
```
data['LotConfig'].unique()
```

Out[47]:
```
array(['Inside', 'FR2', 'Corner', 'CulDSac', 'FR3'], dtype=object)
```

In [58]:
```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.4.0-py2.py3-none-any.whl (86 kB)
     |████████████████████████████████| 86 kB 2.6 MB/s eta 0:00:011
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/d
ist-packages (from category_encoders) (0.5.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from category_encoders) (1.0.2)
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.
7/dist-packages (from category_encoders) (1.3.5)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/pytho
n3.7/dist-packages (from category_encoders) (0.10.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/d
ist-packages (from category_encoders) (1.4.1)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/
```

```
dist-packages (from category_encoders) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/p
ython3.7/dist-packages (from pandas>=0.21.1->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/d
ist-packages (from pandas>=0.21.1->category_encoders) (2018.9)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packa
ges (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/d
ist-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from scikit-learn>=0.20.0->category_encoders) (3.1.
0)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.4.0
```

In [88]:
```
#CountEncoder
from category_encoders.count import CountEncoder as ce_CountEncoder
```

In [100…
```
ce_CountEncoder1 = ce_CountEncoder()
data_COUNT_ENC = ce_CountEncoder1.fit_transform(data[data.columns.differ
```

In [103…
```
data_COUNT_ENC.head()
```

Out[103…

	1stFlrSF	2ndFlrSF	3SsnPorch	Alley	BedroomAbvGr	BldgType	BsmtCond	BsmtExpos
0	856	854	0	1369	3	1220	1311	
1	1262	0	0	1369	3	1220	1311	
2	920	866	0	1369	3	1220	1311	
3	961	756	0	1369	3	1220	65	
4	1145	1053	0	1369	4	1220	1311	

5 rows × 79 columns

In [104…
```
data['MSZoning'].unique()
```

Out[104…
```
array(['RL', 'RM', 'C (all)', 'FV', 'RH'], dtype=object)
```

In [105…
```
data_COUNT_ENC['MSZoning'].unique()
```

Out[105…
```
array([1151,  218,   10,   65,   16])
```

In [106…
```
ce_CountEncoder2 = ce_CountEncoder(normalize=True)
data_FREQ_ENC = ce_CountEncoder2.fit_transform(data[data.columns.differe
```

In [107…
```
data_FREQ_ENC['MSZoning'].unique()
```

Out[107…
```
array([0.78835616, 0.14931507, 0.00684932, 0.04452055, 0.0109589 ])
```

In [117…
```
from category_encoders.helmert import HelmertEncoder as ce_HelmertEncode
```

In [118…

```
#HelmetEncoder
ce_HelmertEncoder1 = ce_HelmertEncoder()
data_HELM_ENC = ce_HelmertEncoder1.fit_transform(data[data.columns.diffe
```

In [119…

```
data_HELM_ENC.head()
```

Out[119…

	intercept	1stFlrSF	2ndFlrSF	3SsnPorch	Alley_0	Alley_1	BedroomAbvGr	BldgType_0
0	1	856	854	0	-1.0	-1.0	3	-1.0
1	1	1262	0	0	-1.0	-1.0	3	-1.0
2	1	920	866	0	-1.0	-1.0	3	-1.0
3	1	961	756	0	-1.0	-1.0	3	-1.0
4	1	1145	1053	0	-1.0	-1.0	4	-1.0

5 rows × 255 columns

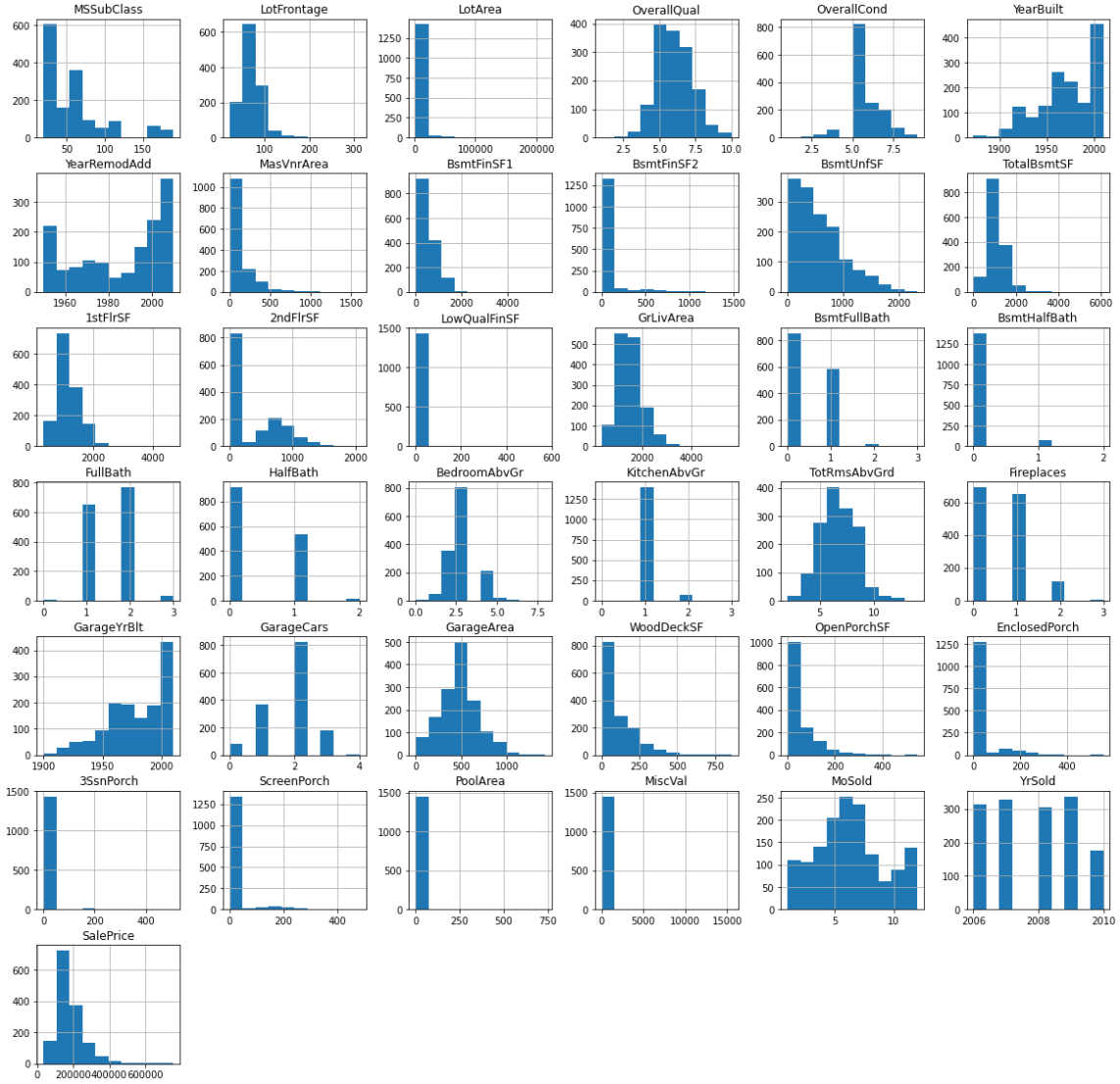
Нормализация числовых признаков

In [120…

```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ##  Q-Q  plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

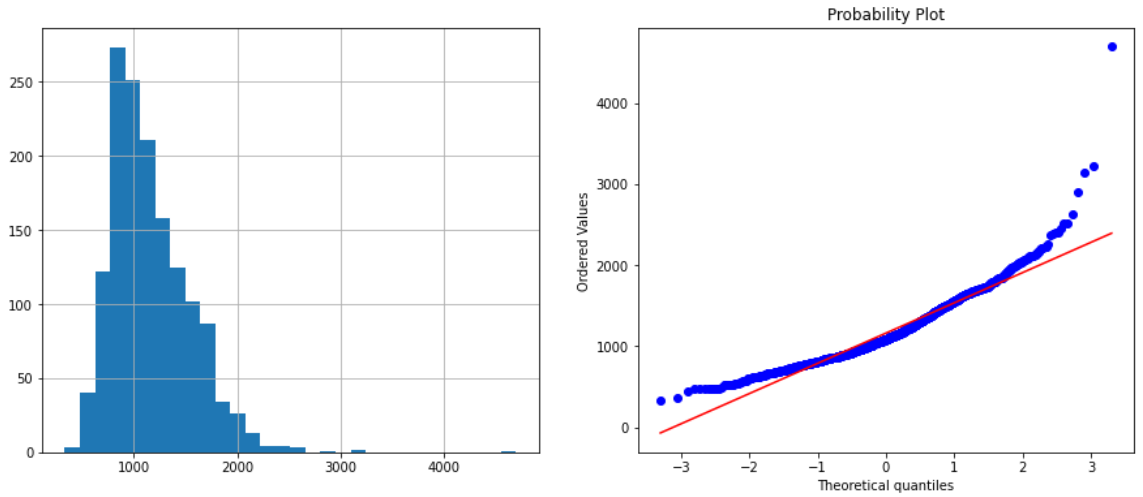
In [121…

```
data.hist(figsize=(20,20))
plt.show()
```



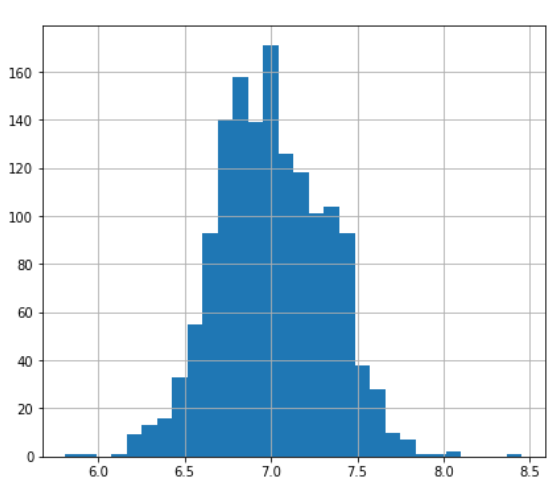
In [126…

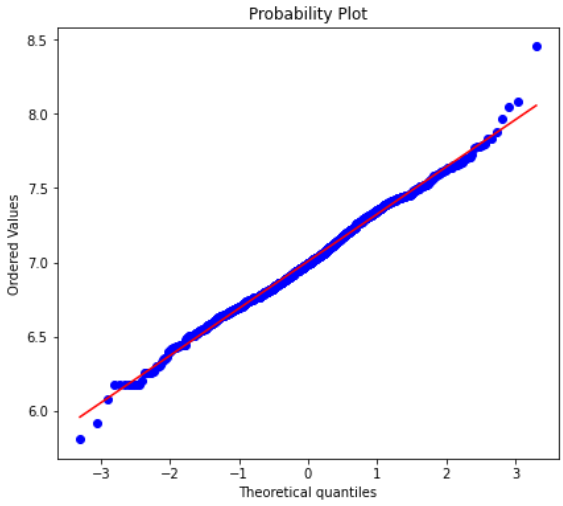
```
diagnostic_plots(data, '1stFlrSF')
```



In [127…

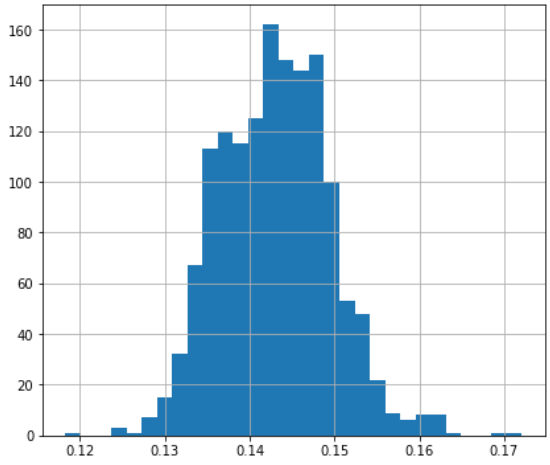
```
#Логарифмическое преобразование
data['1stFlrSF'] = np.log(data['1stFlrSF'])
diagnostic_plots(data, '1stFlrSF')
```

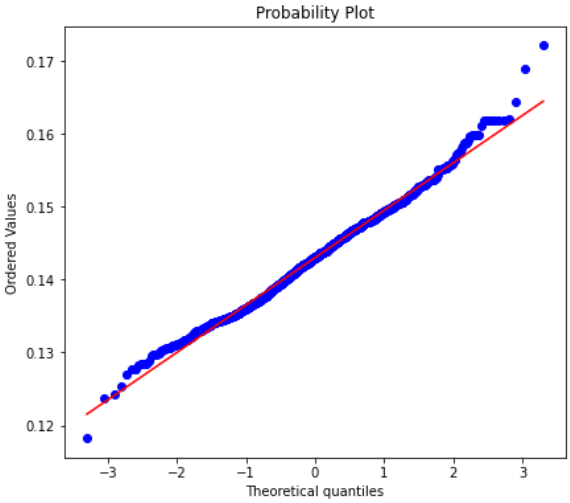




In [128…

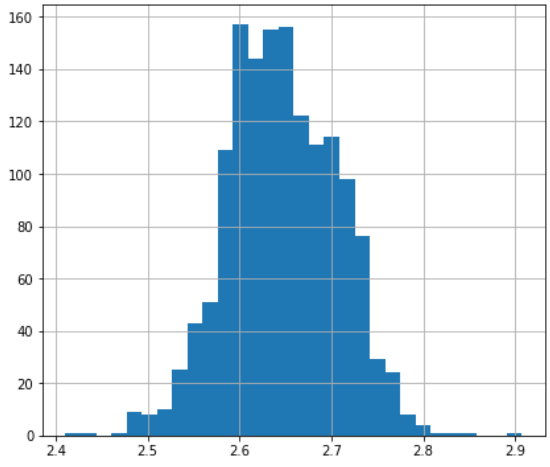
```
#Обратное  преобразование
data['1stFlrSF_reciprocal'] = 1 / (data['1stFlrSF'])
diagnostic_plots(data, '1stFlrSF_reciprocal')
```

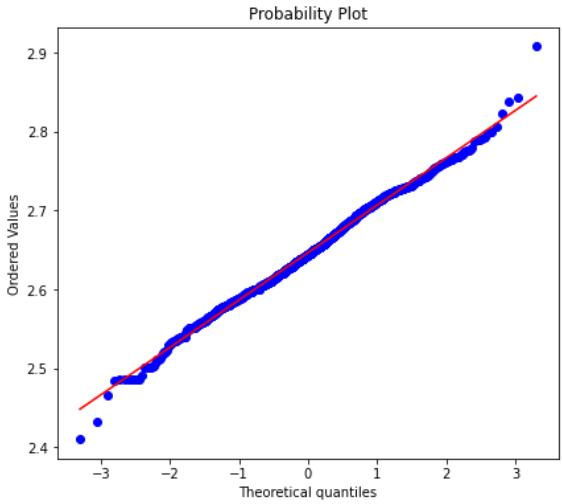




In [129…

```
#Квадратный корень
data['1stFlrSF_sqr'] = data['1stFlrSF']**(1/2)
diagnostic_plots(data, '1stFlrSF_sqr')
```

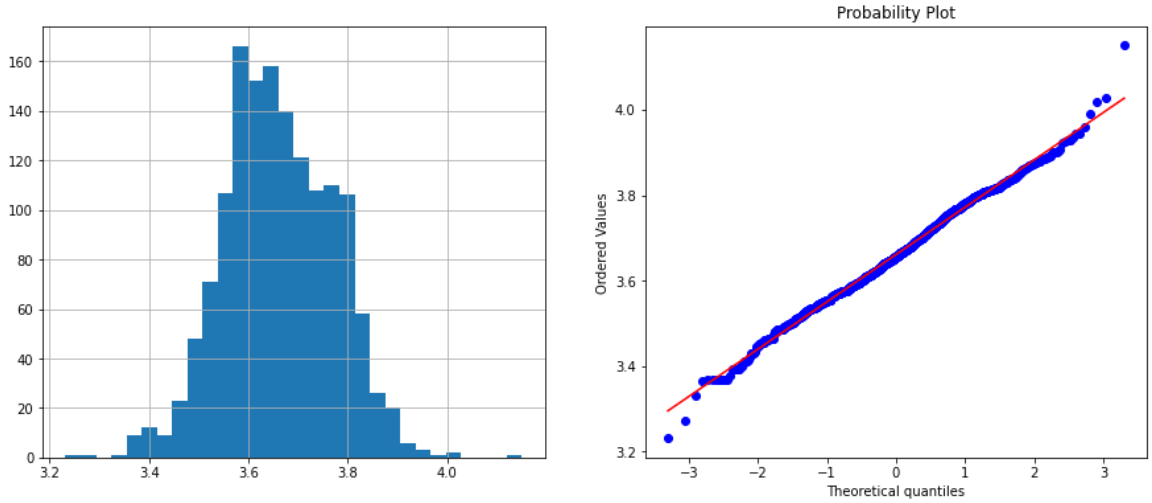




In [130…

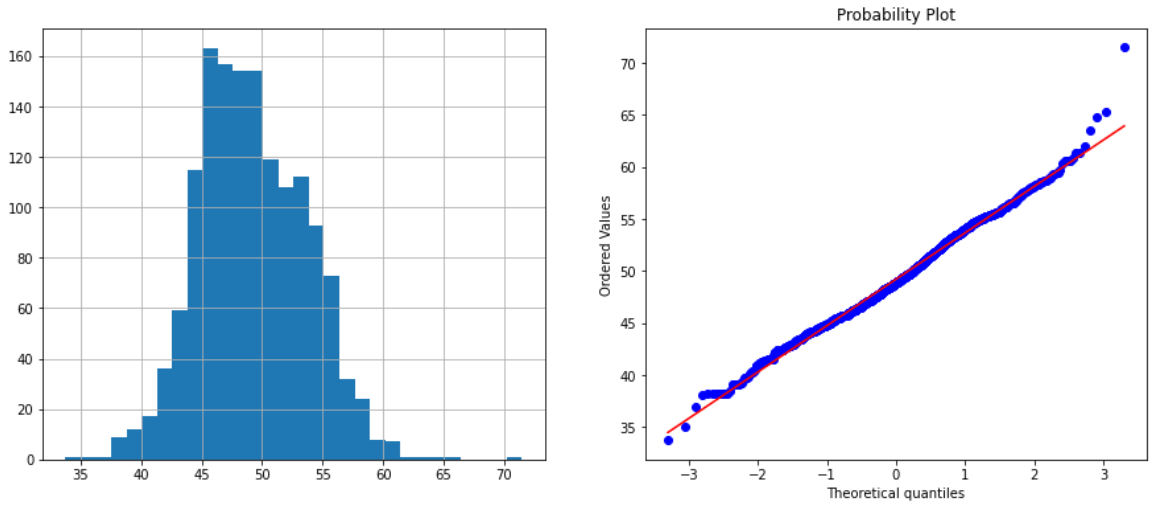
```
#Возведение в степень
data['1stFlrSF_exp1'] = data['1stFlrSF']**(1/1.5)
```

```
diagnostic_plots(data, '1stFlrSF_exp1')
```



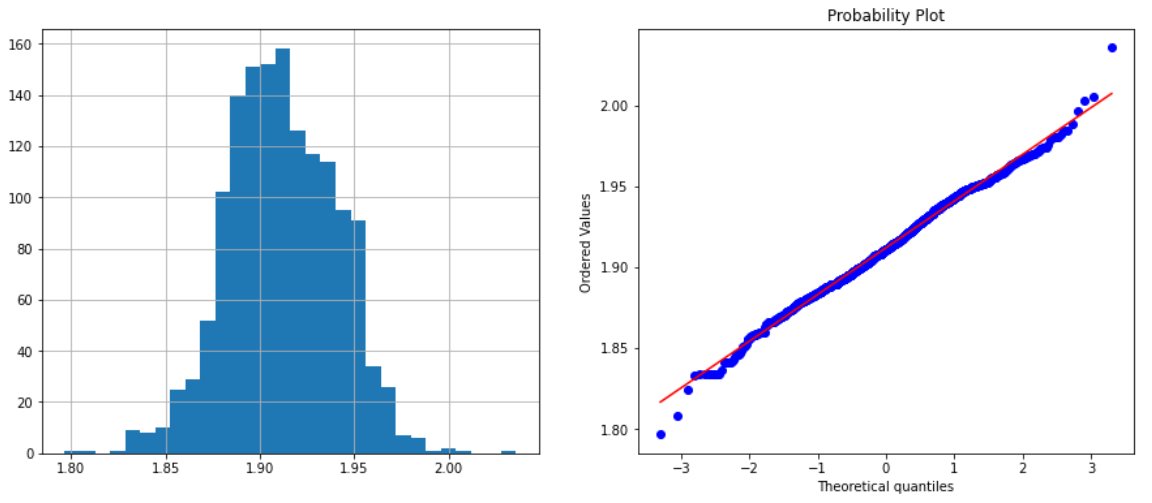
In [131…

```
data['1stFlrSF_exp2'] = data['1stFlrSF']**(2)
diagnostic_plots(data, '1stFlrSF_exp2')
```



In [132…

```
data['1stFlrSF_exp3'] = data['1stFlrSF']**(0.333)
diagnostic_plots(data, '1stFlrSF_exp3')
```

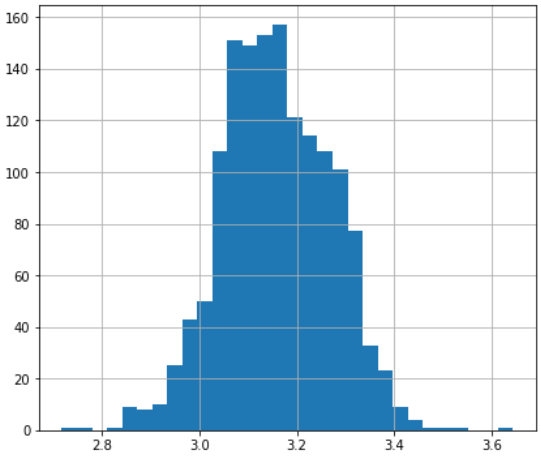


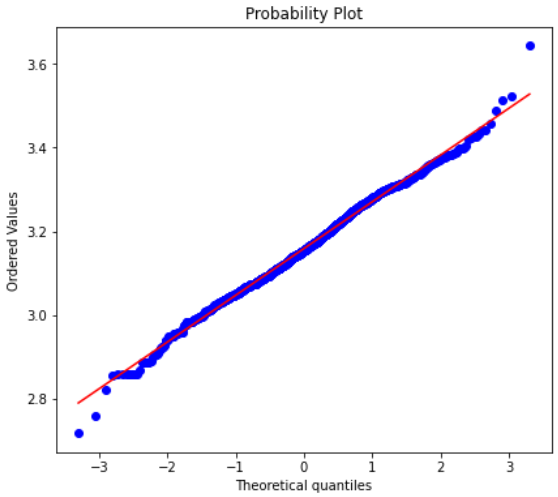
In [133…

```
#Преобразованиея Бокса-Кокса
data['1stFlrSF_boxcox'], param = stats.boxcox(data['1stFlrSF'])
```

```
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, '1stFlrSF_boxcox')
```

```
Оптимальное значение λ = 0.46304765872484194
```





In [ ]: