Импортирование необходимых библиотек

```
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
```

```
data = pd.read_csv("/content/house_sales.csv")
```

```
data = data.drop('Id', 1)
data.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: In a fut
  """Entry point for launching an IPython kernel.
```

	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

```
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
```

```
      ('YearRemodAdd', ('int64', 0)),
      ('RoofStyle', ('object', 0)),
      ('RoofMatl', ('object', 0)),
      ('Exterior1st', ('object', 0)),
      ('Exterior2nd', ('object', 0)),
      ('MasVnrType', ('object', 8)),
      ('MasVnrArea', ('float64', 8)),
```

Сохранено

```
('ExterQual', ('object', 0)),
('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)),
```

Устранение пропусков

Сохранено

in data.columns]

```
('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),
...0137),
```

Сохранено

```
('YrSold', 0.0),
('SaleType', 0.0),
```

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

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

1460 rows × 61 columns

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

Сохранено

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilit
			65.0	8450	Pave	Reg	Lvl	All
1	20	RL	80.0	9600	Pave	Reg	Lvl	All
2	60	RL	68.0	11250	Pave	IR1	Lvl	All
3	70	RL	60.0	9550	Pave	IR1	Lvl	All
4	60	RL	84.0	14260	Pave	IR1	Lvl	All
...	...	...	...	...	...	...	...	
1455	60	RL	62.0	7917	Pave	Reg	Lvl	All

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

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

```
    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

```
from sklearn.preprocessing import LabelEncoder
```

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

```
data['SaleCondition'].unique()
```

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

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



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

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

```
data['LotConfig'].unique()
```

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

```
pip install category_encoders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting category_encoders
  Downloading category_encoders-2.4.1-py2.py3-none-any.whl (80 kB)
     |████████████████████████████████| 80 kB 5.6 MB/s
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from pats
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pa
Installing collected packages: category-encoders
Successfully installed category-encoders-2.4.1
```

```
#CountEncoder
from category_encoders.count import CountEncoder as ce_CountEncoder
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
ce_CountEncoder1 = ce_CountEncoder()
data_COUNT_ENC = ce_CountEncoder1.fit_transform(data[data.columns.difference(['SaleType'])])
```

```
data_COUNT_ENC.head()
```

Сохранено

	1stFlrSF	2ndFlrSF	3SsnPorch	Alley	BedroomAbvGr	BldgType	BsmtCond	BsmtExposure
			0	1369	3	1220	1311	953
1	1262	0	0	1369	3	1220	1311	134
2	920	866	0	1369	3	1220	1311	114
3	961	756	0	1369	3	1220	65	953
4	1145	1053	0	1369	4	1220	1311	221

5 rows × 79 columns

```
data['MSZoning'].unique()
```

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

```
data_COUNT_ENC['MSZoning'].unique()
```

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

```
ce_CountEncoder2 = ce_CountEncoder(normalize=True)
data_FREQ_ENC = ce_CountEncoder2.fit_transform(data[data.columns.difference(['SaleType'])])
```

```
data_FREQ_ENC['MSZoning'].unique()
```

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

```
from category_encoders.helmert import HelmertEncoder as ce_HelmertEncoder
```

```
#HelmetEncoder
ce_HelmertEncoder1 = ce_HelmertEncoder()
data_HELM_ENC = ce_HelmertEncoder1.fit_transform(data[data.columns.difference(['SaleType'])],
```

```
data_HELM_ENC.head()
```

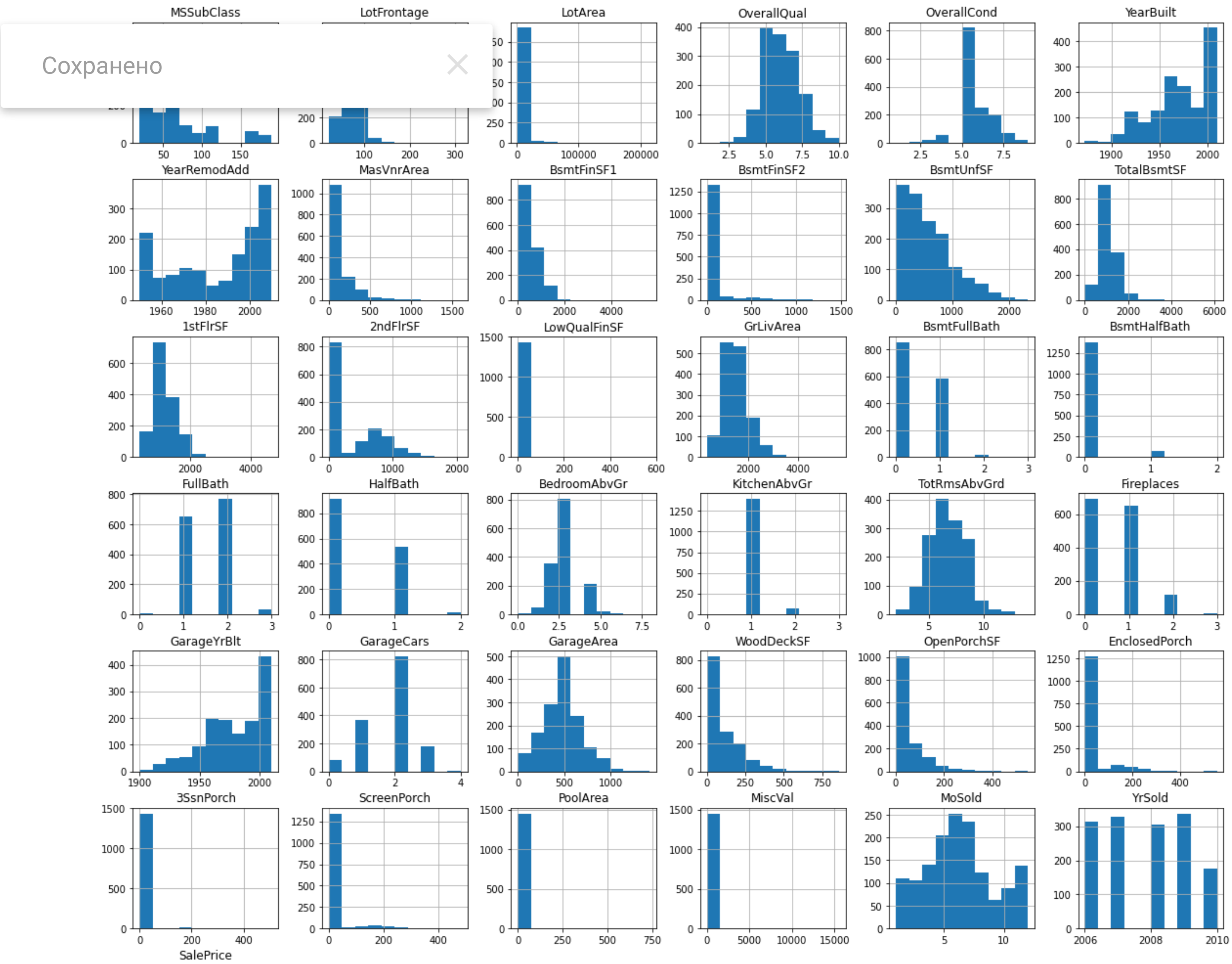
Сохранено

intercept	1stFlrSF	2ndFlrSF	3SsnPorch	Alley_0	Alley_1	BedroomAbvGr	BldgType_0
[illegible]	[illegible]	[illegible]54	0	-1.0	-1.0	3	-1.0

Нормализация числовых признаков

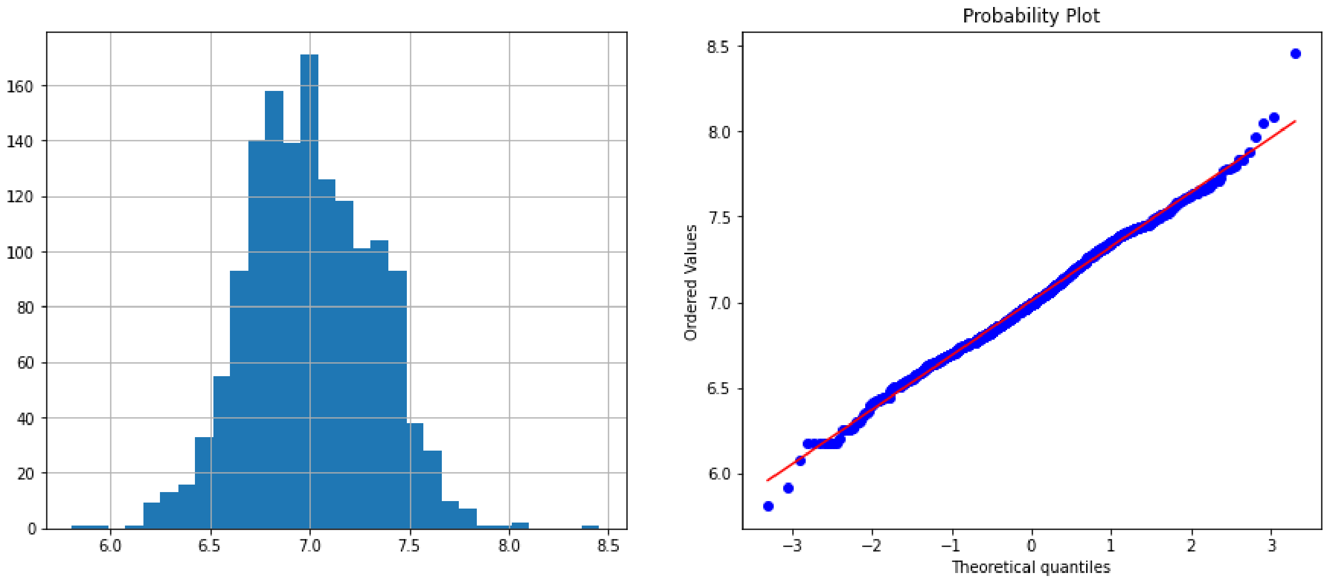
```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()


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

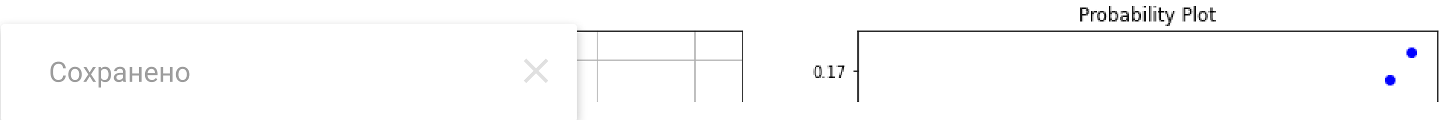



```
diagnostic_plots(data, '1stFlrSF')
```

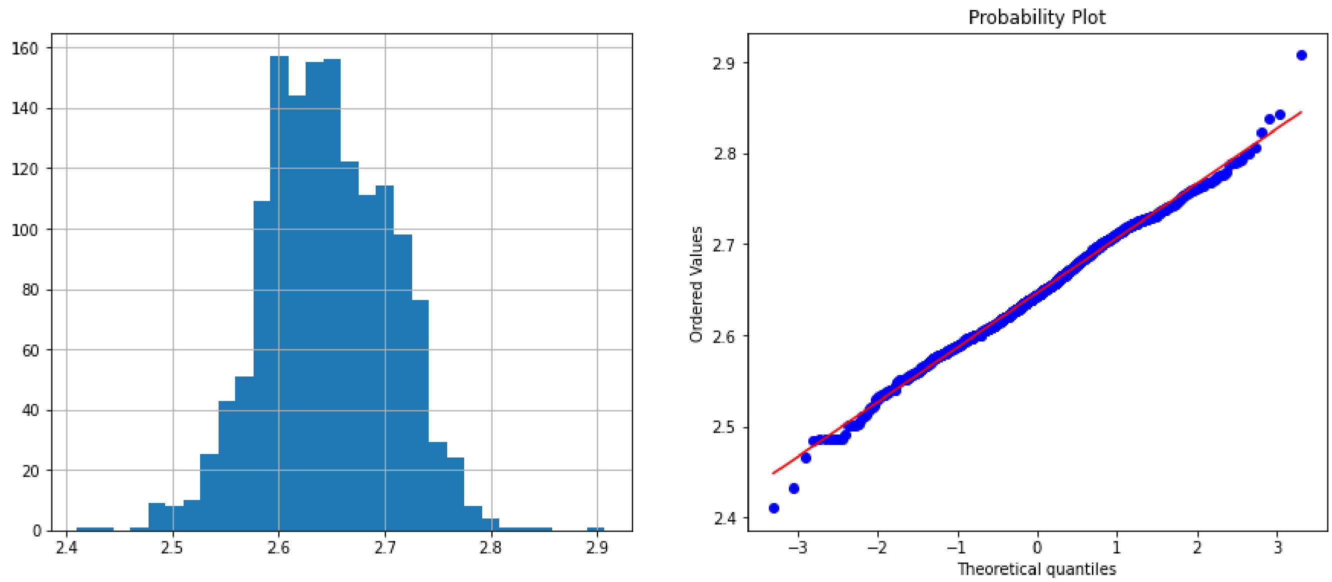

```
#Логарифмическое преобразование
data['1stFlrSF'] = np.log(data['1stFlrSF'])
diagnostic_plots(data, '1stFlrSF')
```



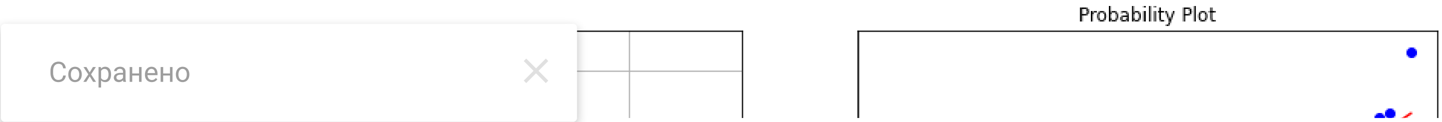

```
#Обратное преобразование
data['1stFlrSF_reciprocal'] = 1 / (data['1stFlrSF'])
diagnostic_plots(data, '1stFlrSF_reciprocal')
```



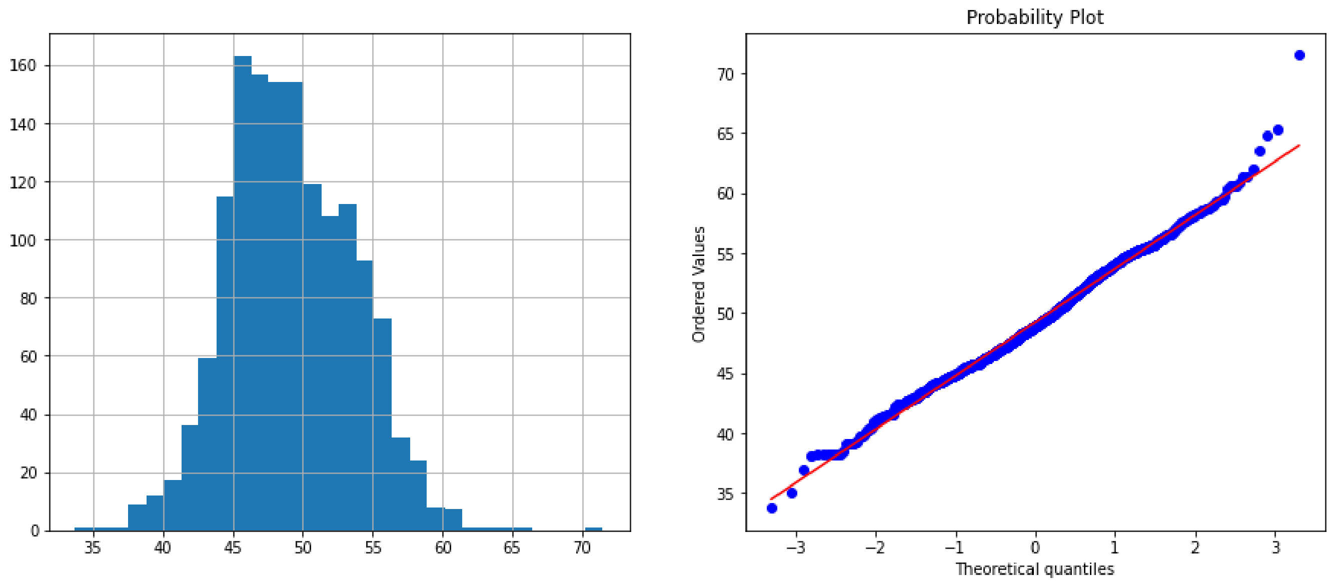

```
#Квадратный корень
data['1stFlrSF_sqr'] = data['1stFlrSF']**(1/2)
diagnostic_plots(data, '1stFlrSF_sqr')
```




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




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




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

Сохранено

```
data['1stFlrSF_boxcox'], param = stats.boxcox(data['1stFlrSF'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, '1stFlrSF_boxcox')
```

```
Оптимальное значение λ = 0.46304765872484194
```