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

Лабораторная работа №2-3
по дисциплине
«Методы машинного обучения»
на тему
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных. Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»

Выполнил:
студент группы ИУ5-14М
Подопригорова Н. С.

Москва — 2023 г.

1. Лабораторная №2

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

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные 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.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
    - устранение пропусков в данных;
    - кодирование категориальных признаков;
    - нормализация числовых признаков.

2. Melbourne Housing Snapshot

- Rooms: Number of rooms
- Price: Price in dollars
- Method: S - property sold; SP - property sold prior; PI - property passed in; PN - sold prior not disclosed; SN - - sold not disclosed; NB - no bid; VB - vendor bid; W - withdrawn prior to auction; SA - sold after auction; SS - sold after auction price not disclosed. N/A - price or highest bid not available.
- Type: br - bedroom(s); h - house,cottage,villa, semi,terrace; u - unit, duplex; t - townhouse; dev site - - development site; o res - other residential.
- SellerG: Real Estate Agent
- Date: Date sold
- Distance: Distance from CBD
- Regionname: General Region (West, North West, North, North east …etc)
- Propertycount: Number of properties that exist in the suburb.
- Bedroom2 : Scraped # of Bedrooms (from different source)
- Bathroom: Number of Bathrooms
- Car: Number of carspots
- Landsize: Land Size
- BuildingArea: Building Size
- CouncilArea: Governing council for the area

[14]:
```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats

import matplotlib.pyplot as plt
```

[3]:
```
data = pd.read_csv('melb_data.csv')
```

[25]:
```
data.head()
```

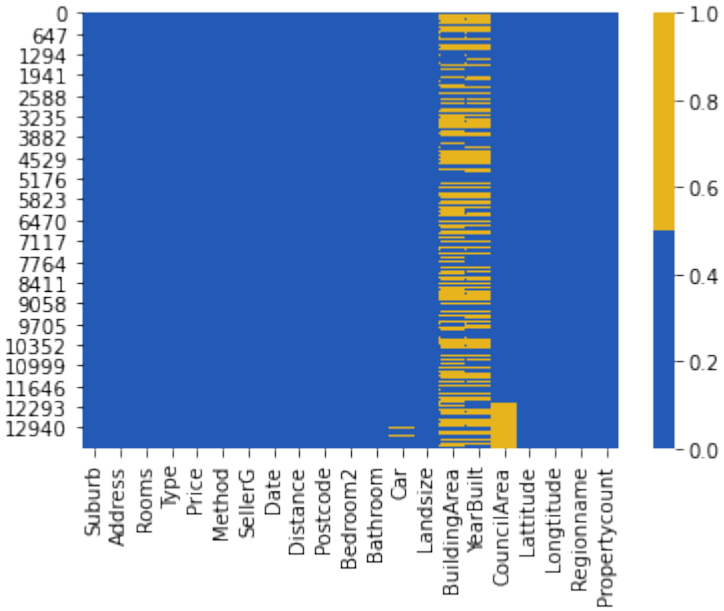
[25]:
```
       Suburb           Address  Rooms Type      Price Method SellerG  \
0  Abbotsford      85 Turner St      2    h  1480000.0      S  Biggin
1  Abbotsford   25 Bloomburg St      2    h  1035000.0      S  Biggin
2  Abbotsford      5 Charles St      3    h  1465000.0     SP  Biggin
3  Abbotsford  40 Federation La      3    h   850000.0     PI  Biggin
4  Abbotsford       55a Park St      4    h  1600000.0     VB  Nelson

        Date  Distance  Postcode  …  Bathroom  Car  Landsize  BuildingArea  \
0  3/12/2016       2.5    3067.0  …       1.0  1.0     202.0           NaN
1  4/02/2016       2.5    3067.0  …       1.0  0.0     156.0          79.0
2  4/03/2017       2.5    3067.0  …       2.0  0.0     134.0         150.0
3  4/03/2017       2.5    3067.0  …       2.0  1.0      94.0           NaN
4  4/06/2016       2.5    3067.0  …       1.0  2.0     120.0         142.0

   YearBuilt  CouncilArea Lattitude  Longtitude             Regionname  \
0        NaN        Yarra  -37.7996    144.9984  Northern Metropolitan
1     1900.0        Yarra  -37.8079    144.9934  Northern Metropolitan
2     1900.0        Yarra  -37.8093    144.9944  Northern Metropolitan
3        NaN        Yarra  -37.7969    144.9969  Northern Metropolitan
4     2014.0        Yarra  -37.8072    144.9941  Northern Metropolitan

  Propertycount
0        4019.0
1        4019.0
2        4019.0
3        4019.0
4        4019.0

[5 rows x 21 columns]
```

2.1. Обработка пропусков в данных

Желтый - пропущенные данные, синий - не пропущенные

[10]:
```
cols = data.columns
colours = ['#235AB5', '#E8B41E']
sns.heatmap(data[cols].isnull(), cmap=sns.color_palette(colours))
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad56bcb940>
```



[9]:
```
data.info()
```

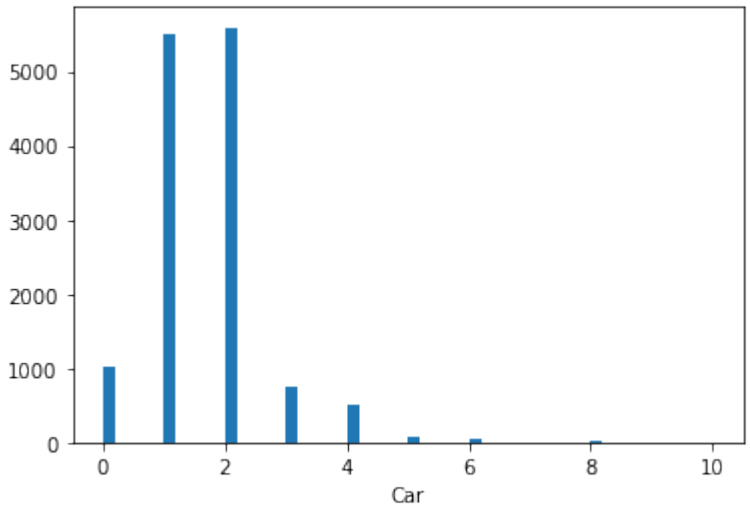
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  object
 1   Address        13580 non-null  object
 2   Rooms          13580 non-null  int64
 3   Type           13580 non-null  object
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object
 6   SellerG        13580 non-null  object
 7   Date           13580 non-null  object
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13518 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   7130 non-null   float64
 15  YearBuilt      8205 non-null   float64
 16  CouncilArea    12211 non-null  object
 17  Lattitude      13580 non-null  float64
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

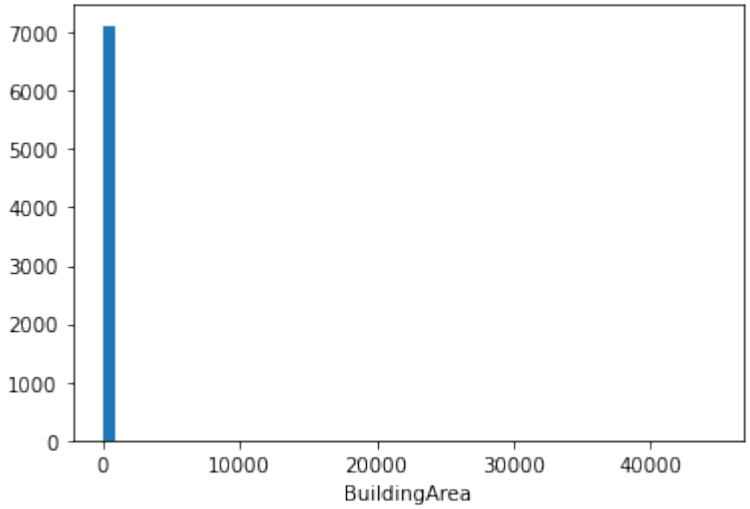
Рассмотрим числовые колонки с пропущенными значениями

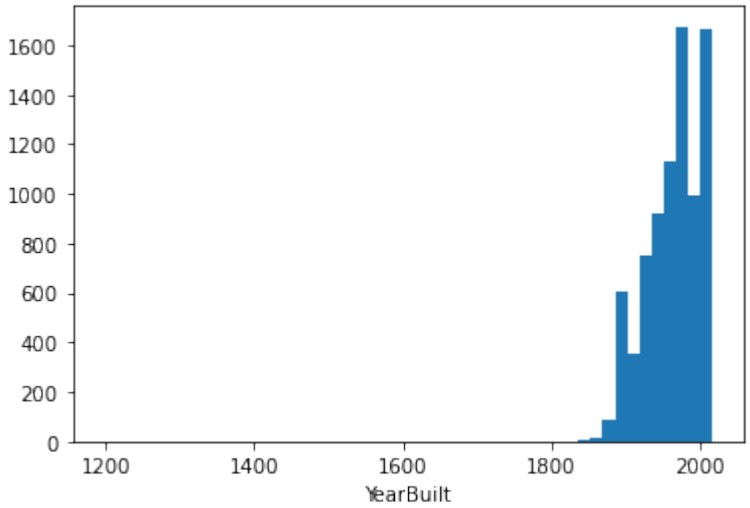
[4]:
```
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('     {}.        {}.                 {}, {}%.'.format(col, dt,␣
 ↪temp_null_count, temp_perc))
```

```
    Car.        float64.                62, 0.46%.
    BuildingArea.       float64.                6450,
47.5%.
    YearBuilt.      float64.                5375, 39.58%.
```

[5]:
```
data_num = data[num_cols]
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```







Выбросов нет, распределения одномодальные

[6]:
```
data = data.fillna(data.mode())
```

Рассмотрим пропуски в категориальных данных

[7]:
```
cat_cols = []
for col in data.columns:
    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('     {}.         {}.                  {}, {}%.'.format(col, dt,␣
 ↪temp_null_count, temp_perc))
```

```
    CouncilArea.         object.                  1369, 10.08%.
```

[8]:
```
data[:] = SimpleImputer(missing_values=np.nan, strategy='most_frequent').
↪fit_transform(data)
```

[17]:
```
data.isnull().sum()
```

[17]:
```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
YearBuilt        0
CouncilArea      0
Lattitude        0
Longtitude       0
Regionname       0
Propertycount    0
dtype: int64
```

Все пропуски в данных заполнены

2.2. Кодирование категориальных признаков

Рассмотрим количество категорий в признаках типа object

[25]:
```
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    dt = str(data[col].dtype)
    if (dt=='object'):
        num_cols.append(col)
        print('     {}.            : {}'.format(col, data[col].unique().
↪shape[0]))
```

```
    Suburb.            : 314
    Address.            : 13378
    Type.            : 3
    Method.            : 5
    SellerG.            : 268
    Date.            : 58
    CouncilArea.            : 33
    Regionname.            : 8
```

В признаках Suburb, Address, SellerG, CouncilArea слишком много категорий для OneHotEncoder, так что используем LabelEncoder.

[18]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
categorical1 = ['Suburb', 'Address', 'SellerG', 'CouncilArea']
for col in categorical1:
    data[col] = le.fit_transform(data[col])
```

Для остальных признаков используем OneHotEncoder

[19]:
```
categorical2 = ['Type', 'Method', 'Regionname']

data = pd.concat([data, pd.get_dummies(data[categorical2],␣
 ↪columns=categorical2, drop_first=True)],axis=1)
data.drop(categorical2, axis=1, inplace=True)
```

Дату обработаем отдельно

[20]:
```
import datetime as dt

data['Date'] = pd.to_datetime(data['Date'])
data['Date'] = data['Date'].map(dt.datetime.toordinal)
```

[35]:
```
data.shape
```

[35]:
```
(13580, 31)
```

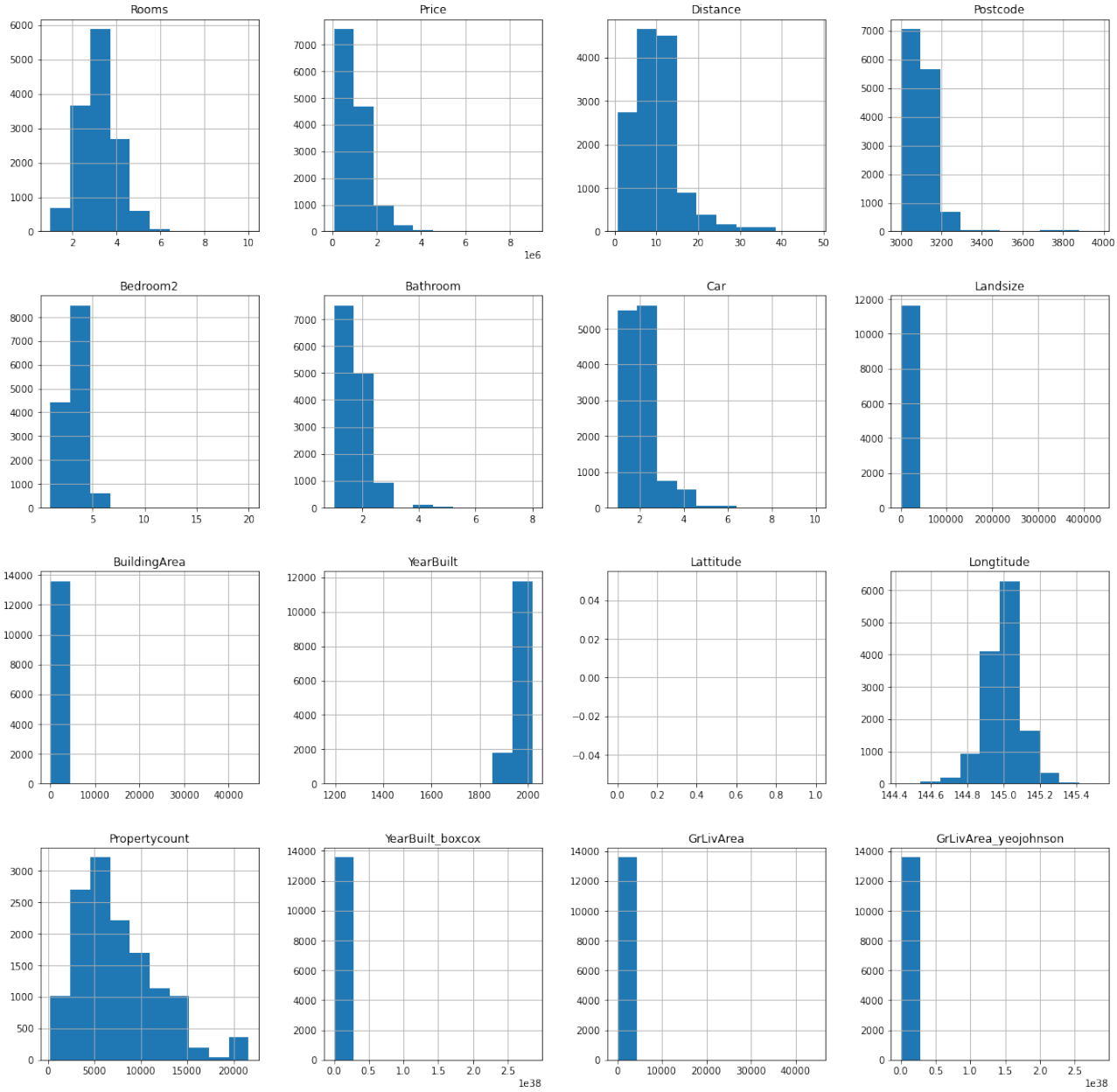
2.3. Нормализация числовых признаков

[52]:
```
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    dt = str(data[col].dtype)
    if dt=='float64' or dt=='int64':
        num_cols.append(col)
```

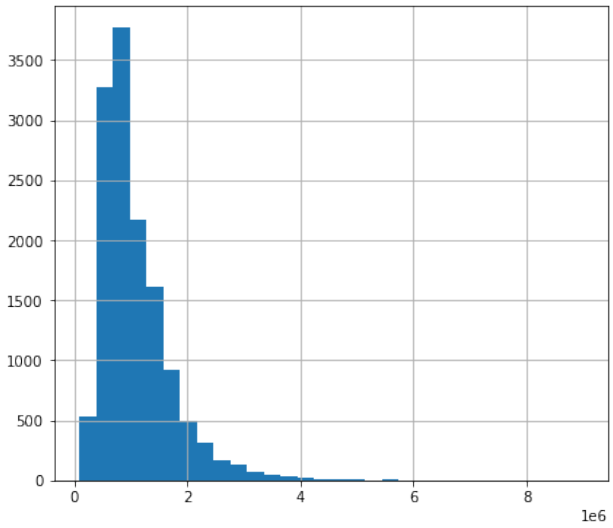
[53]:
```
data[num_cols] = data[data[num_cols] > 0][num_cols]
```

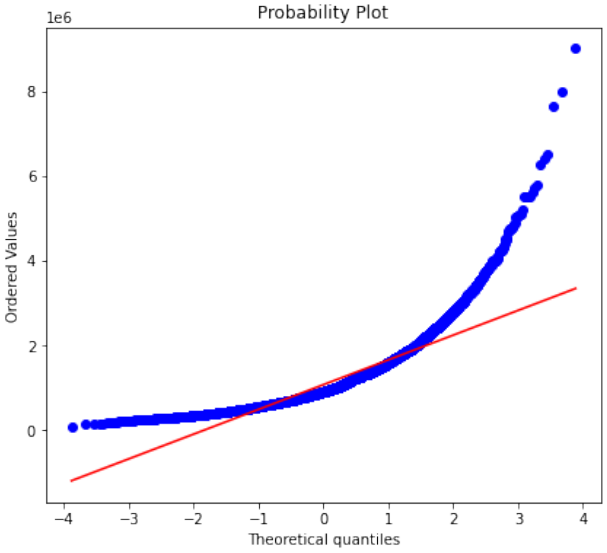
[50]:
```
def diagnostic_plots_data(df):
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df.hist(bins=30)
    plt.subplot(1, 2, 2)
    stats.probplot(df, dist="norm", plot=plt)
    plt.show()
```

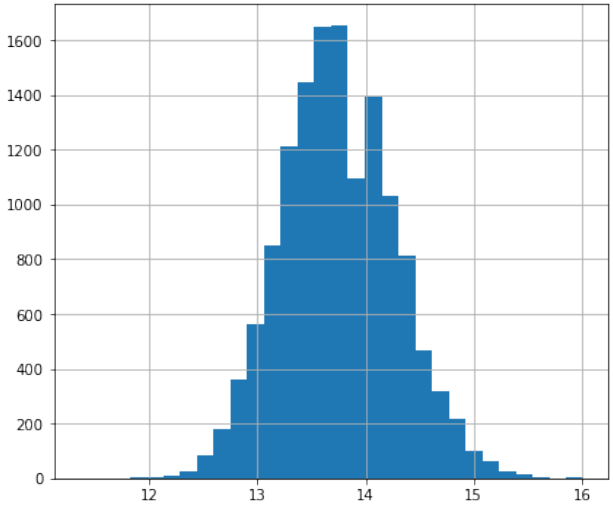
[54]:
```
data[num_cols].hist(figsize=(20,20))
plt.show()
```

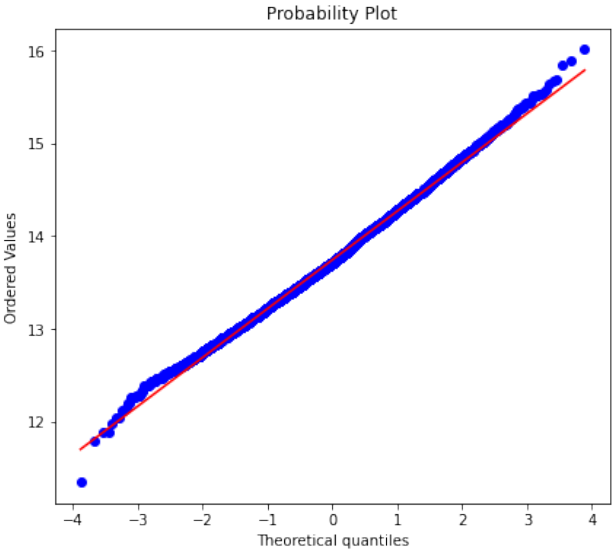


[59]:
```
diagnostic_plots_data(data['Price'])
diagnostic_plots_data( np.log(data['Price']) )
```

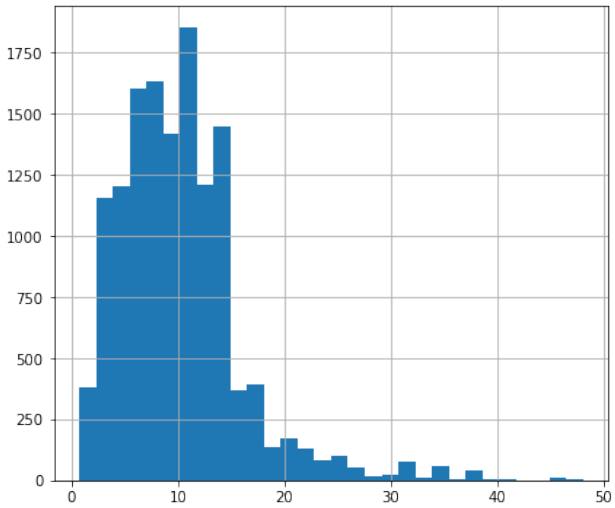


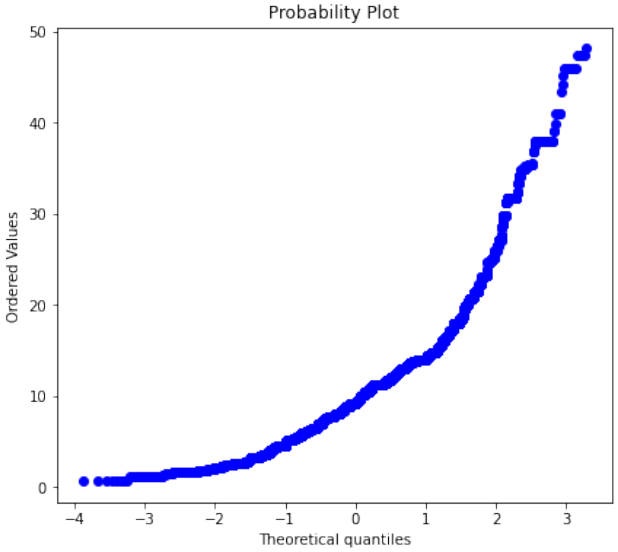


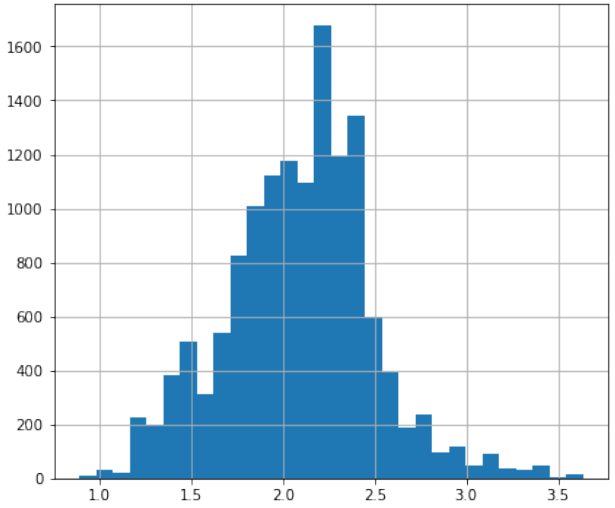


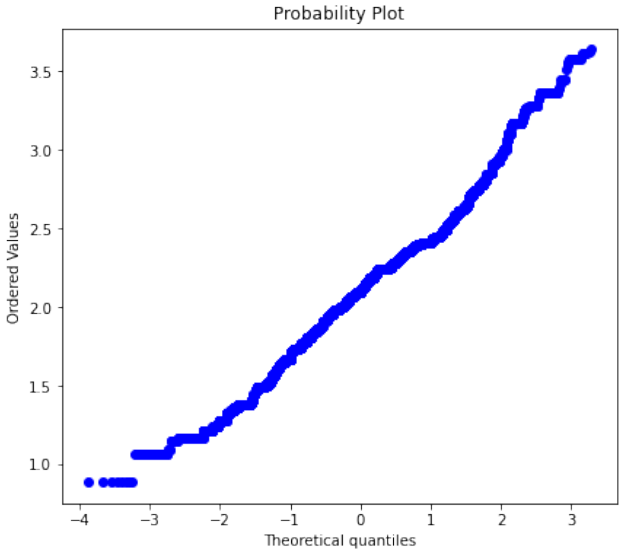


[65]:
```
diagnostic_plots_data(data['Distance'])
diagnostic_plots_data( data['Distance']**(1/3) )
```

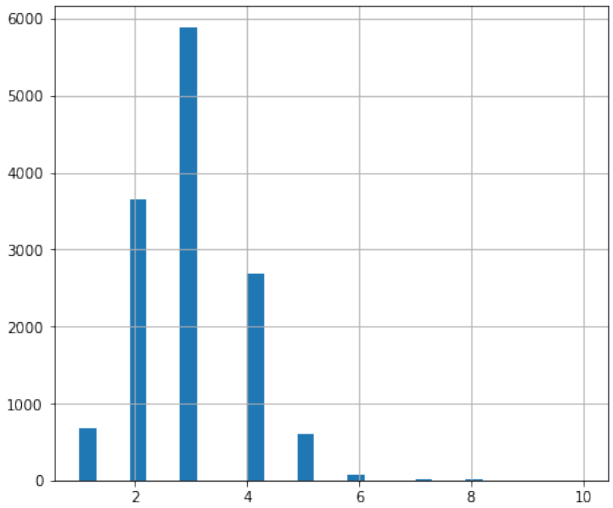


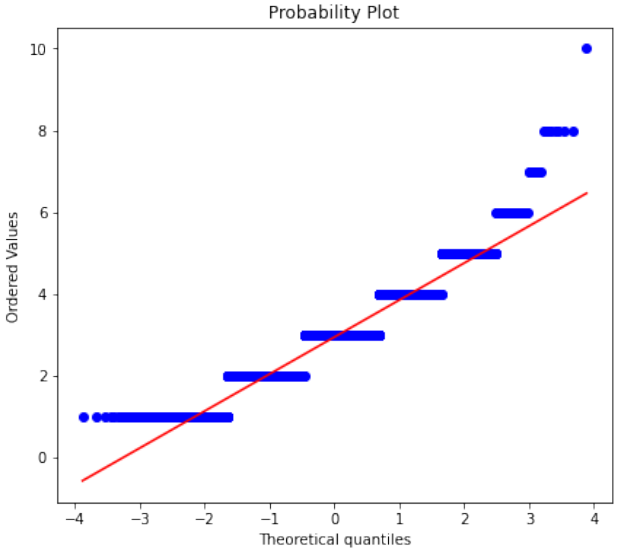


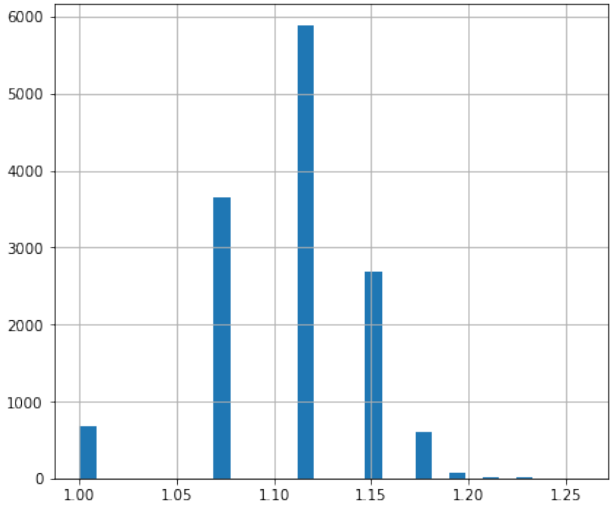


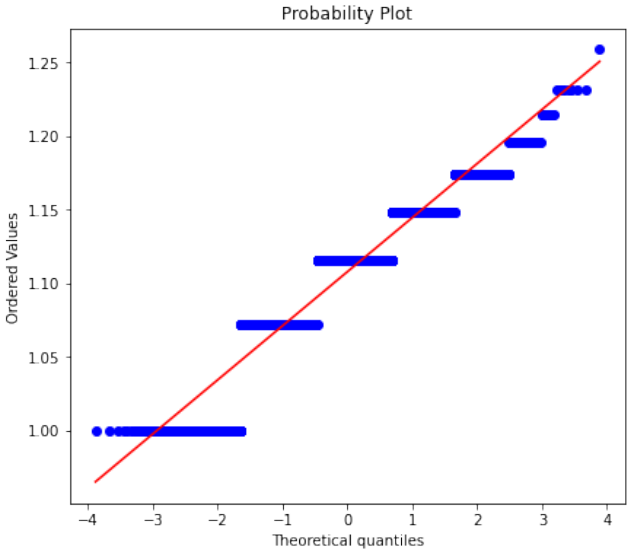


[57]:
```
diagnostic_plots_data(data['Rooms'])
diagnostic_plots_data( data['Rooms']**(1/10)  )
```

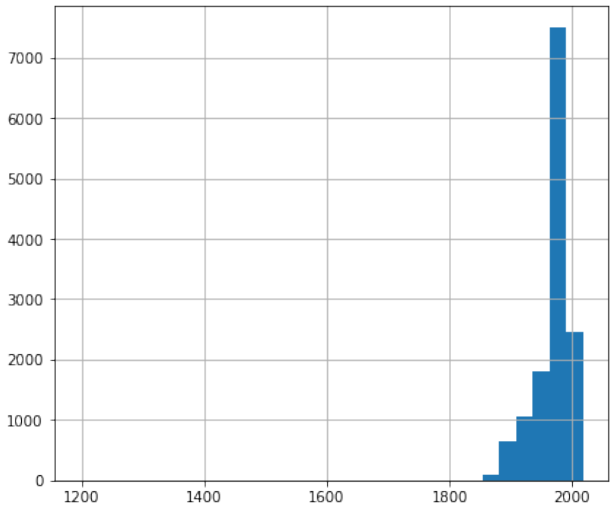


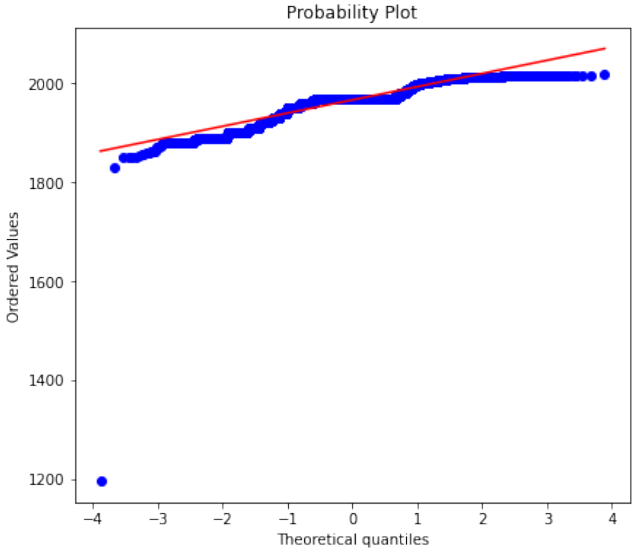






[23]:
```
diagnostic_plots_data(data['YearBuilt'])

data['YearBuilt_boxcox'], param = stats.boxcox(data['YearBuilt'])
print('                = {}'.format(param))
diagnostic_plots_data(data['YearBuilt_boxcox'])
```





= 18.95465909623817