```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv("C:\\Users\\my pc\\Downloads\\
CarPrice_Assignment.csv")
df.head()
```

```
   car_ID  symboling                   CarName fueltype aspiration
doornumber  \
0       1          3        alfa-romero giulia      gas        std
two
1       2          3       alfa-romero stelvio      gas        std
two
2       3          1  alfa-romero Quadrifoglio      gas        std
two
3       4          2               audi 100 ls      gas        std
four
4       5          2                audi 100ls      gas        std
four

       carbody drivewheel enginelocation  wheelbase  ...
enginesize  \
0  convertible        rwd          front       88.6  ...         130

1  convertible        rwd          front       88.6  ...         130

2    hatchback        rwd          front       94.5  ...         152

3        sedan        fwd          front       99.8  ...         109

4        sedan        4wd          front       99.4  ...         136


   fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm
citympg  \
0        mpfi       3.47    2.68              9.0        111     5000
21
1        mpfi       3.47    2.68              9.0        111     5000
21
2        mpfi       2.68    3.47              9.0        154     5000
19
3        mpfi       3.19    3.40             10.0        102     5500
24
4        mpfi       3.19    3.40              8.0        115     5500
18

   highwaympg    price
0          27  13495.0
```

```
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

```
df=df[['symboling','wheelbase','carlength','carwidth','carheight','cur
bweight','boreratio','stroke','compressionratio','horsepower','peakrpm
','citympg','highwaympg','price']]
```

```
df.head()
```

```
   symboling  wheelbase  carlength  carwidth  carheight  curbweight  \
0          3       88.6      168.8      64.1       48.8        2548
1          3       88.6      168.8      64.1       48.8        2548
2          1       94.5      171.2      65.5       52.4        2823
3          2       99.8      176.6      66.2       54.3        2337
4          2       99.4      176.6      66.4       54.3        2824

   boreratio  stroke  compressionratio  horsepower  peakrpm
citympg  \
0       3.47    2.68               9.0         111     5000       21

1       3.47    2.68               9.0         111     5000       21

2       2.68    3.47               9.0         154     5000       19

3       3.19    3.40              10.0         102     5500       24

4       3.19    3.40               8.0         115     5500       18


   highwaympg    price  price_log  price_reciprocal  price_sqroot  \
0          27  13495.0   9.510075          0.000074    116.167982
1          27  16500.0   9.711116          0.000061    128.452326
2          26  16500.0   9.711116          0.000061    128.452326
3          30  13950.0   9.543235          0.000072    118.110118
4          22  17450.0   9.767095          0.000057    132.098448

   price_exponential
0        2765.736115
1        3270.167165
2        3270.167165
3        2843.228997
4        3426.332661
```

```
df.shape
```

```
(205, 18)
```

Feature Scaling

To bring the data columns in the same unit and range

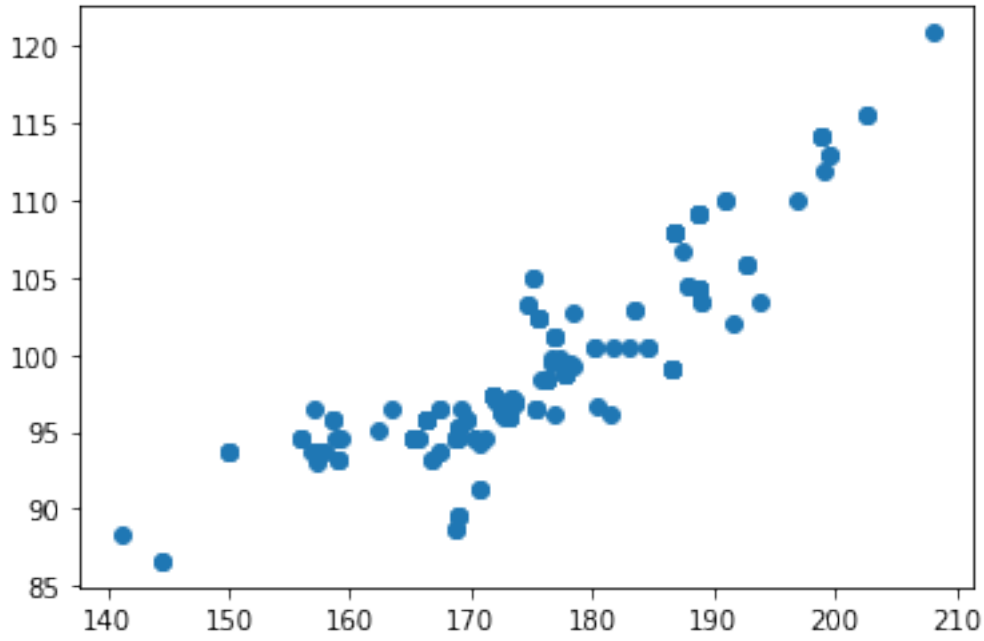
```
scaler = MinMaxScaler()

df_scaled=df.copy()
col_names=['peakrpm','carlength']
features=df_scaled[col_names]
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

```
     symboling  wheelbase  carlength  carwidth  carheight  curbweight
\
0            3       88.6   0.413433      64.1       48.8        2548

1            3       88.6   0.413433      64.1       48.8        2548

2            1       94.5   0.449254      65.5       52.4        2823

3            2       99.8   0.529851      66.2       54.3        2337

4            2       99.4   0.529851      66.4       54.3        2824

..         ...        ...        ...       ...        ...         ...

200         -1      109.1   0.711940      68.9       55.5        2952

201         -1      109.1   0.711940      68.8       55.5        3049

202         -1      109.1   0.711940      68.9       55.5        3012

203         -1      109.1   0.711940      68.9       55.5        3217

204         -1      109.1   0.711940      68.9       55.5        3062


     boreratio  stroke  compressionratio  horsepower   peakrpm
citympg  \
0         3.47    2.68               9.0         111  0.346939
21
1         3.47    2.68               9.0         111  0.346939
21
2         2.68    3.47               9.0         154  0.346939
19
3         3.19    3.40              10.0         102  0.551020
24
4         3.19    3.40               8.0         115  0.551020
18
```

```
..          ...     ...                ...         ...       ...      ..
.
200       3.78    3.15                9.5         114  0.510204
23
201       3.78    3.15                8.7         160  0.469388
19
202       3.58    2.87                8.8         134  0.551020
18
203       3.01    3.40               23.0         106  0.265306
26
204       3.78    3.15                9.5         114  0.510204
19

     highwaympg    price
0             27  13495.0
1             27  16500.0
2             26  16500.0
3             30  13950.0
4             22  17450.0
..           ...      ...
200           28  16845.0
201           25  19045.0
202           23  21485.0
203           27  22470.0
204           25  22625.0

[205 rows x 14 columns]
```

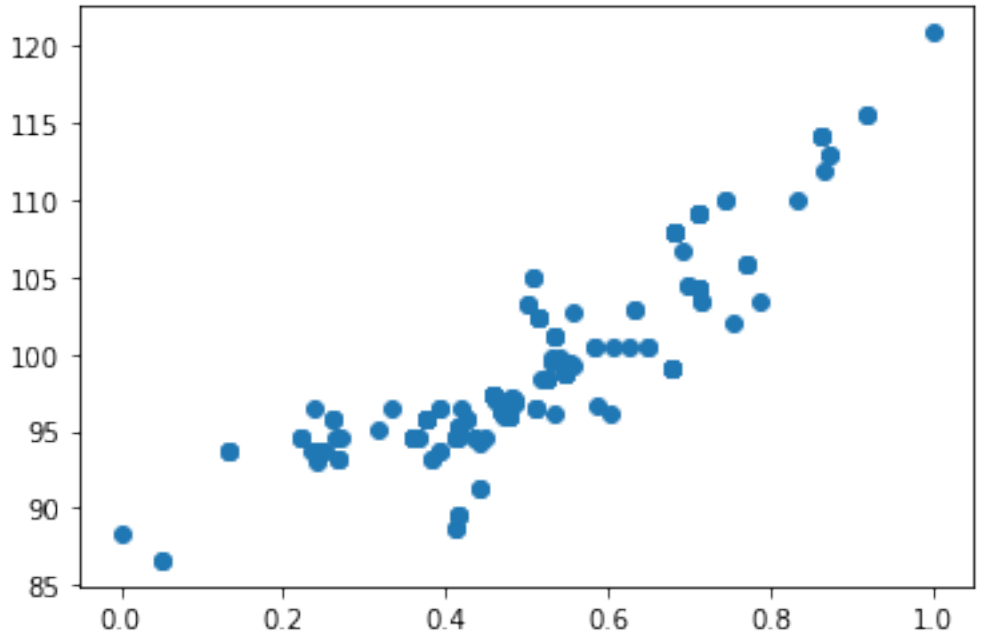
```
y = df['wheelbase']
x = df['carlength']

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x762f6d8760>
```




```
y = df_scaled['wheelbase']
x = df_scaled['carlength']

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7629066130>
```




```
scaler = MinMaxScaler(feature_range=(5, 10))
```

```
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

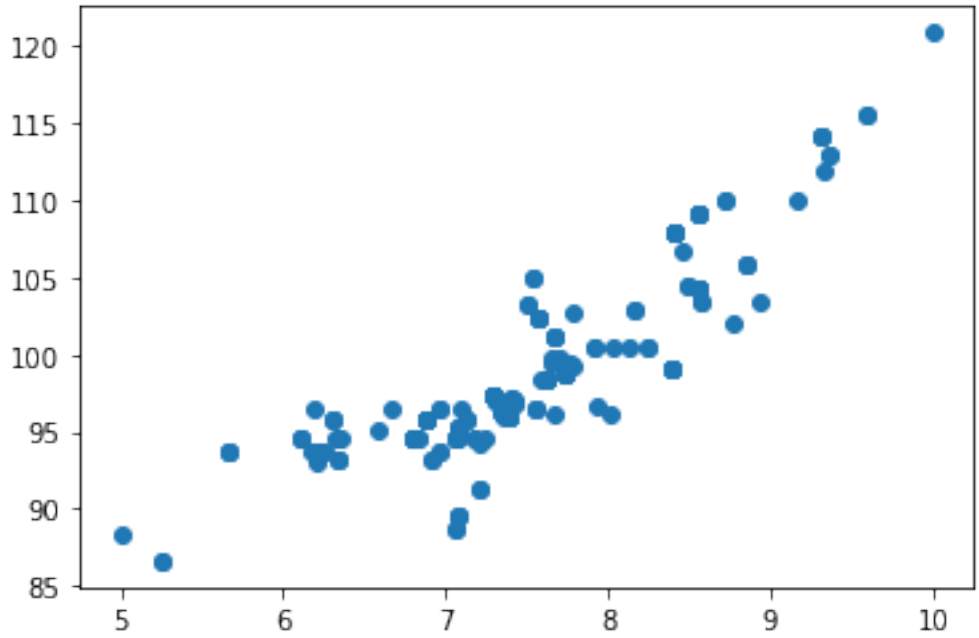
```
     symboling  wheelbase  carlength  carwidth  carheight  curbweight
\
0            3       88.6   7.067164      64.1       48.8        2548

1            3       88.6   7.067164      64.1       48.8        2548

2            1       94.5   7.246269      65.5       52.4        2823

3            2       99.8   7.649254      66.2       54.3        2337

4            2       99.4   7.649254      66.4       54.3        2824

..         ...        ...        ...       ...        ...         ...

200         -1      109.1   8.559701      68.9       55.5        2952

201         -1      109.1   8.559701      68.8       55.5        3049

202         -1      109.1   8.559701      68.9       55.5        3012

203         -1      109.1   8.559701      68.9       55.5        3217

204         -1      109.1   8.559701      68.9       55.5        3062


     boreratio  stroke  compressionratio  horsepower   peakrpm
citympg  \
0         3.47    2.68               9.0         111  6.734694
21
1         3.47    2.68               9.0         111  6.734694
21
2         2.68    3.47               9.0         154  6.734694
19
3         3.19    3.40              10.0         102  7.755102
24
4         3.19    3.40               8.0         115  7.755102
18
..         ...     ...               ...         ...       ...      ..
.
200       3.78    3.15               9.5         114  7.551020
23
201       3.78    3.15               8.7         160  7.346939
19
202       3.58    2.87               8.8         134  7.755102
18
203       3.01    3.40              23.0         106  6.326531
```

```
26
204       3.78    3.15                9.5         114  7.551020
19
```

```
     highwaympg    price
0             27  13495.0
1             27  16500.0
2             26  16500.0
3             30  13950.0
4             22  17450.0
..           ...      ...
200           28  16845.0
201           25  19045.0
202           23  21485.0
203           27  22470.0
204           25  22625.0

[205 rows x 14 columns]
```

```
y = df_scaled['wheelbase']
x = df_scaled['carlength']

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x762909fb20>
```



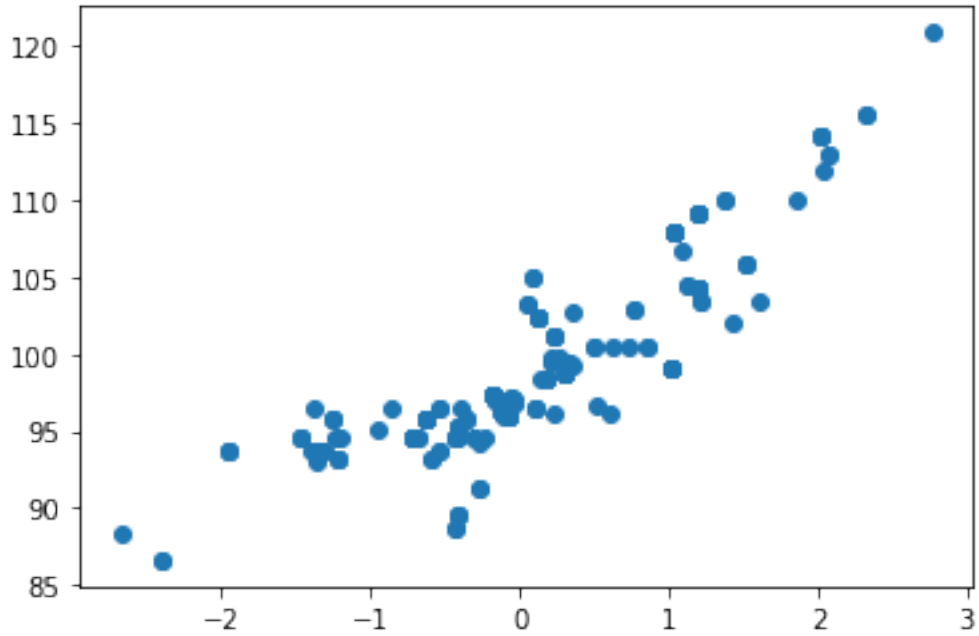

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled[col_names] = scaler.fit_transform(features.values)
df_scaled
```

```
     symboling  wheelbase  carlength  carwidth  carheight  curbweight
\
0            3       88.6  -0.426521      64.1       48.8        2548

1            3       88.6  -0.426521      64.1       48.8        2548

2            1       94.5  -0.231513      65.5       52.4        2823

3            2       99.8   0.207256      66.2       54.3        2337

4            2       99.4   0.207256      66.4       54.3        2824

..         ...        ...        ...       ...        ...         ...

200         -1      109.1   1.198549      68.9       55.5        2952

201         -1      109.1   1.198549      68.8       55.5        3049

202         -1      109.1   1.198549      68.9       55.5        3012

203         -1      109.1   1.198549      68.9       55.5        3217

204         -1      109.1   1.198549      68.9       55.5        3062


     boreratio  stroke  compressionratio  horsepower   peakrpm
citympg  \
0         3.47    2.68               9.0         111 -0.262960
21
1         3.47    2.68               9.0         111 -0.262960
21
2         2.68    3.47               9.0         154 -0.262960
19
3         3.19    3.40              10.0         102  0.787855
24
4         3.19    3.40               8.0         115  0.787855
18
..         ...     ...               ...         ...       ...      ..
.
200       3.78    3.15               9.5         114  0.577692
23
201       3.78    3.15               8.7         160  0.367529
19
202       3.58    2.87               8.8         134  0.787855
18
203       3.01    3.40              23.0         106 -0.683286
26
204       3.78    3.15               9.5         114  0.577692
19
```

```
     highwaympg    price
0             27  13495.0
1             27  16500.0
2             26  16500.0
3             30  13950.0
4             22  17450.0
..           ...      ...
200           28  16845.0
201           25  19045.0
202           23  21485.0
203           27  22470.0
204           25  22625.0

[205 rows x 14 columns]
```

```
y = df_scaled['wheelbase']
x = df_scaled['carlength']

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x762f3157c0>
```



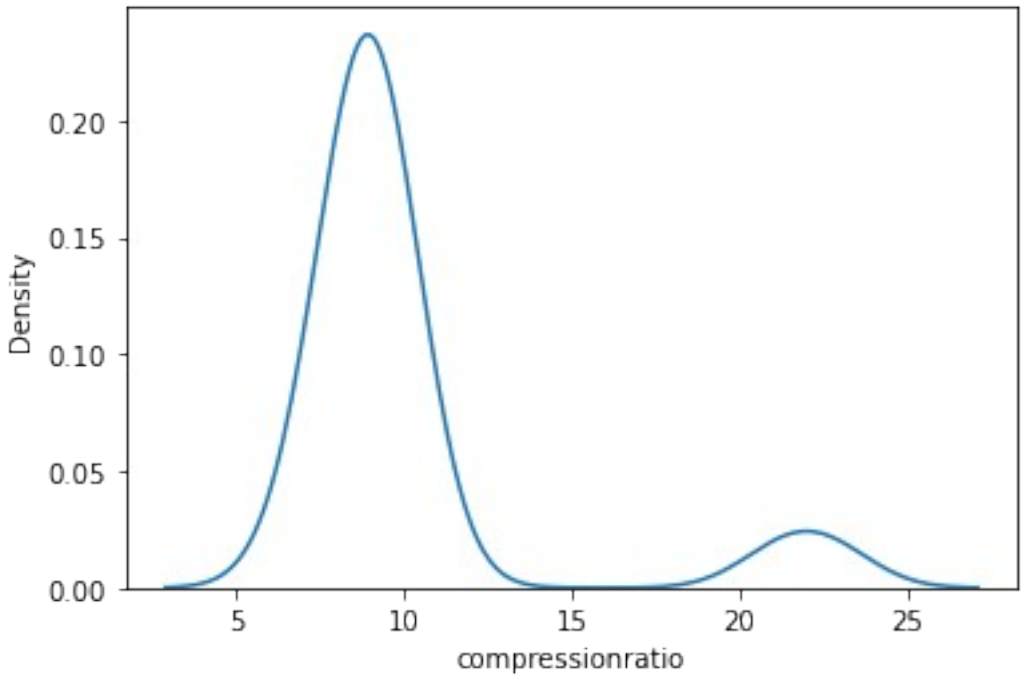

```
#skewness in the data
df.skew()
```

```
symboling           0.211072
wheelbase           1.050214
carlength           0.155954
carwidth            0.904003
carheight           0.063123
```

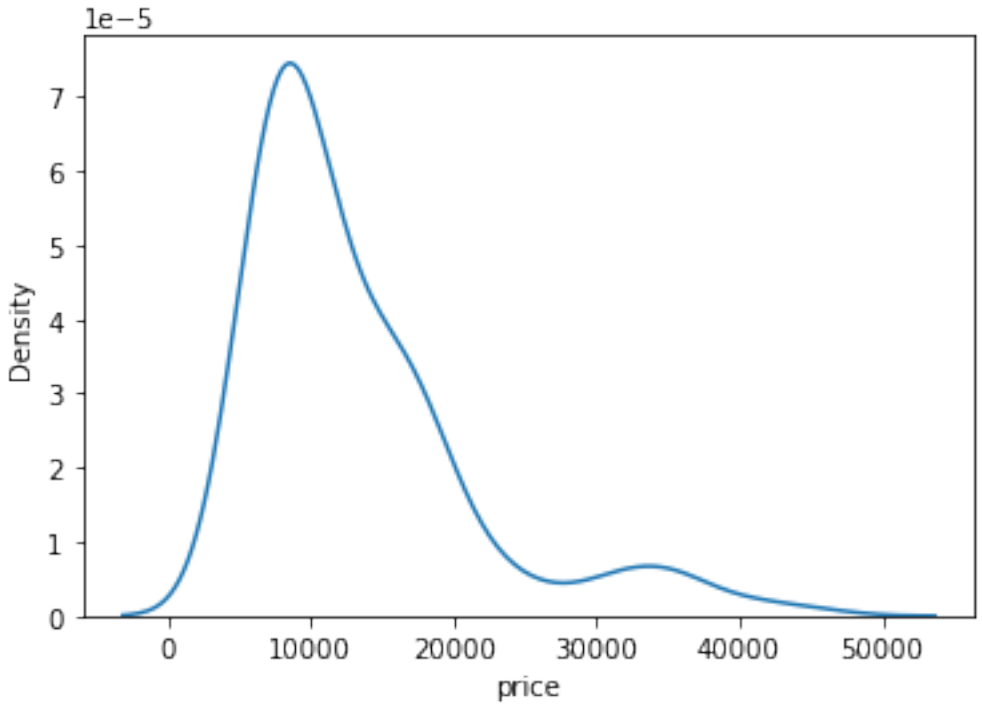
```
curbweight           0.681398
boreratio            0.020156
stroke              -0.689705
compressionratio     2.610862
horsepower           1.405310
peakrpm              0.075159
citympg              0.663704
highwaympg           0.539997
price                1.777678
dtype: float64
```

```
#The variables with skewness > 1 such as wheelbase, compressionratio,
horsepower, price are highly positively skewed.
#The variables with skewness < -1 are highly negat ively skewed.
#The variables with 0.5 < skewness < 1 such as carwidth, curbweight,
citympg are moderately positively skewed.
#The variables with -0.5 < skewness < -1 such as stroke are moderately
negatively skewed.
#And, the variables with -0.5 < skewness < 0.5 are symmetric i.e
normally distributed such as symboling, carheight, boreration,
peakrpm, highwaympg.
```

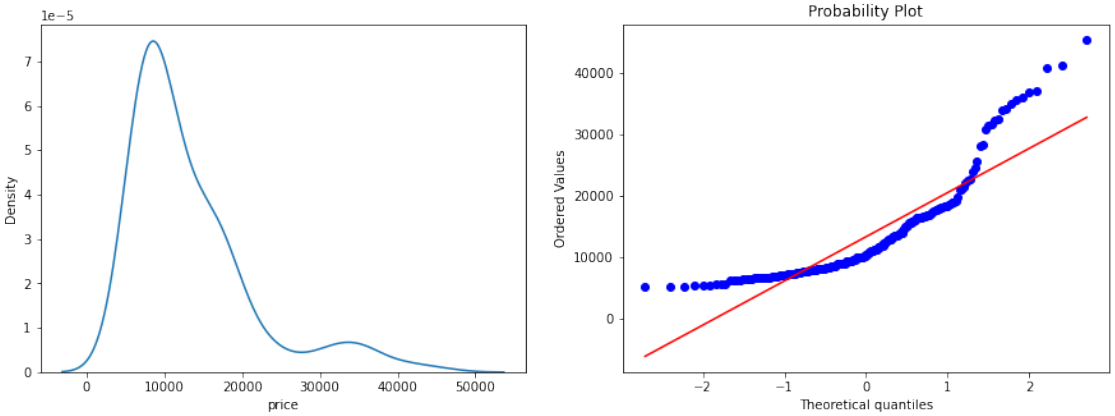
```
sns.kdeplot(df.compressionratio);
```




```
sns.kdeplot(df.price);
```




```
#importing necessary libraries
import scipy.stats as stats
import pylab
def normality(data,feature):
    plt.figure(figsize=(15,5))
    plt.subplot(1,2,1)
    sns.kdeplot(data[feature])
    plt.subplot(1,2,2)
    stats.probplot(data[feature],plot=pylab)
    plt.show()

normality(df,'price')
```



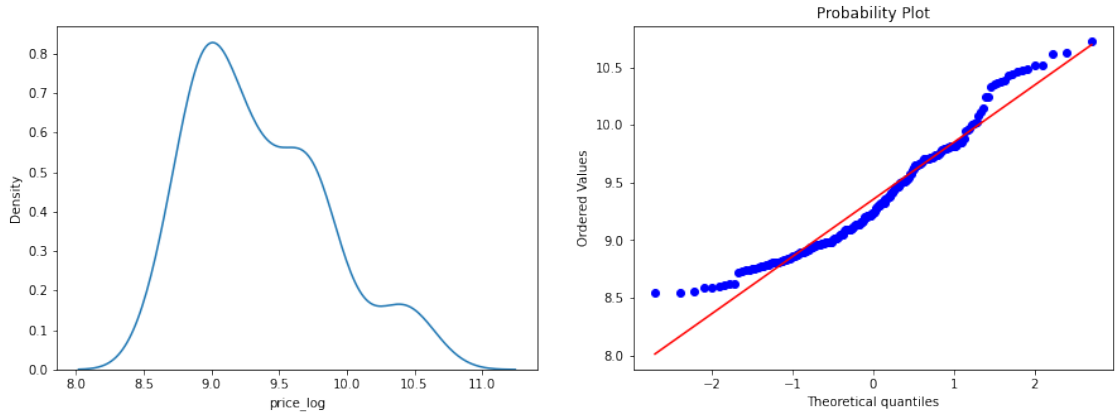
Data Transformation

Skewed to normal

Logarithmic Transformation

This will convert the Price value to its log value i.e log(Price)

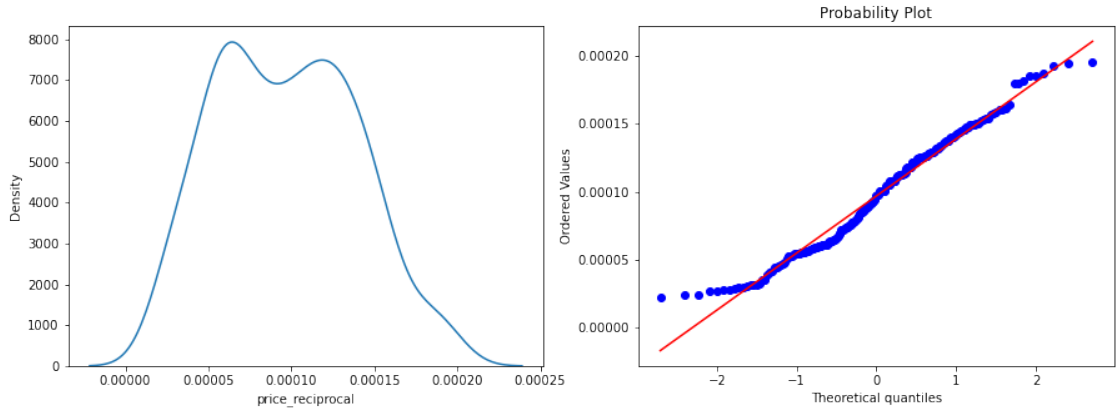
```
#performing logarithmic transformation on the feature
df['price_log']=np.log(df['price'])
#plotting to check the transformation
normality(df,'price_log')
```



Reciprocal Transformation

This will inverse values of Price i.e 1/Price

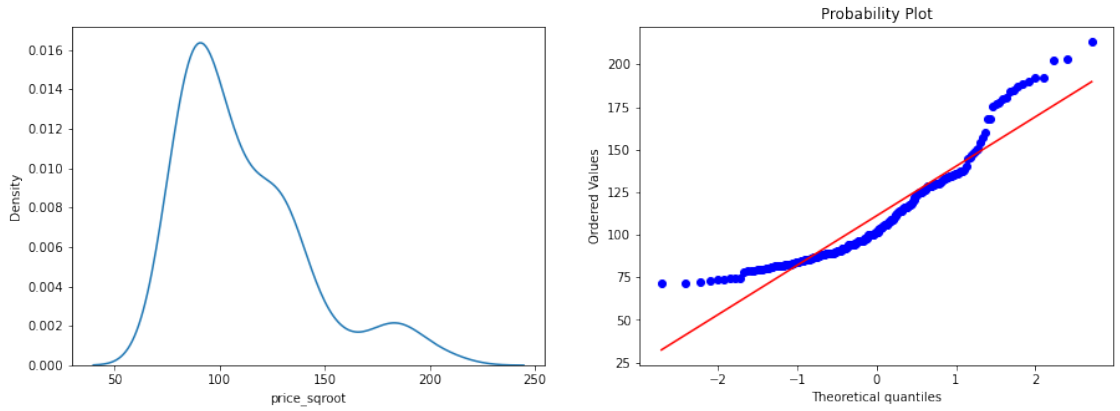
```
df['price_reciprocal']=1/df.price
normality(df,'price_reciprocal')
```



Square Root Transformation

This transformation will take the square root of the Price column i.e sqrt(Price).

```
df['price_sqroot']=np.sqrt(df.price)
normality(df,'price_sqroot')
```



Exponential Transformation:

The exponential value of the Price variable will be taken.

```
df['price_exponential']=df.price**(1/1.2)
normality(df,'price_exponential')
```