house-price-prediction-part1

March 19, 2019

```
In [12]: import pandas as pd
         import numpy as np
         %matplotlib inline
         import matplotlib.pyplot as plt
         import seaborn as sns
         import warnings
         warnings.filterwarnings('ignore')
         from scipy import stats
         from scipy.stats import norm, skew
         from scipy.special import boxcox1p
         #inv - inv_boxcox1p(y, 2.5)
```

```
In [13]: df = pd.read_csv('housing.data', sep='\s+', header=None, names=['CRIM', 'ZN', 'INDUS'
```

```
1. CRIM      per capita crime rate by town

2. ZN        proportion of residential land zoned for lots over
             25,000 sq.ft.

3. INDUS     proportion of non-retail business acres per town

4. CHAS      Charles River dummy variable (= 1 if tract bounds
             river; 0 otherwise)

5. NOX       nitric oxides concentration (parts per 10 million)

6. RM        average number of rooms per dwelling

7. AGE       proportion of owner-occupied units built prior to 1940

8. DIS       weighted distances to five Boston employment centres

9. RAD       index of accessibility to radial highways

10. TAX      full-value property-tax rate per $10,000

11. PTRATIO  pupil-teacher ratio by town
```

```
12. B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks
             by town

13. LSTAT    % lower status of the population

14. MEDV     Median value of owner-occupied homes in $1000's
```

1. CRIM: 1
2. ZN: 25,000 sq.ft. .
3. INDUS: non-retail business(t.v stations, radio stations, internet and telephone businesses, advertising campaigns) ()
4. CHAS: (1, 0)
5. NOX: (1000)
6. RM:
7. AGE: 1940
8. DIS: 5
9. RAD:
10. TAX: $ 10,000
11. PTRATIO: -
12. B:
13. LSTAT:
14. MEDV: 1000

```
In [14]: df.head()
```

```
Out[14]:       CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
         0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
         1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
         2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
         3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
         4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

            PTRATIO       B  LSTAT  MEDV
         0     15.3  396.90   4.98  24.0
         1     17.8  396.90   9.14  21.6
         2     17.8  392.83   4.03  34.7
         3     18.7  394.63   2.94  33.4
         4     18.7  396.90   5.33  36.2
```

1 Part 1. Data Exploration

-

1.0.1 target

-

1.0.2 features

-

1.0.3 including feature engineering

1.1 Target (house price)

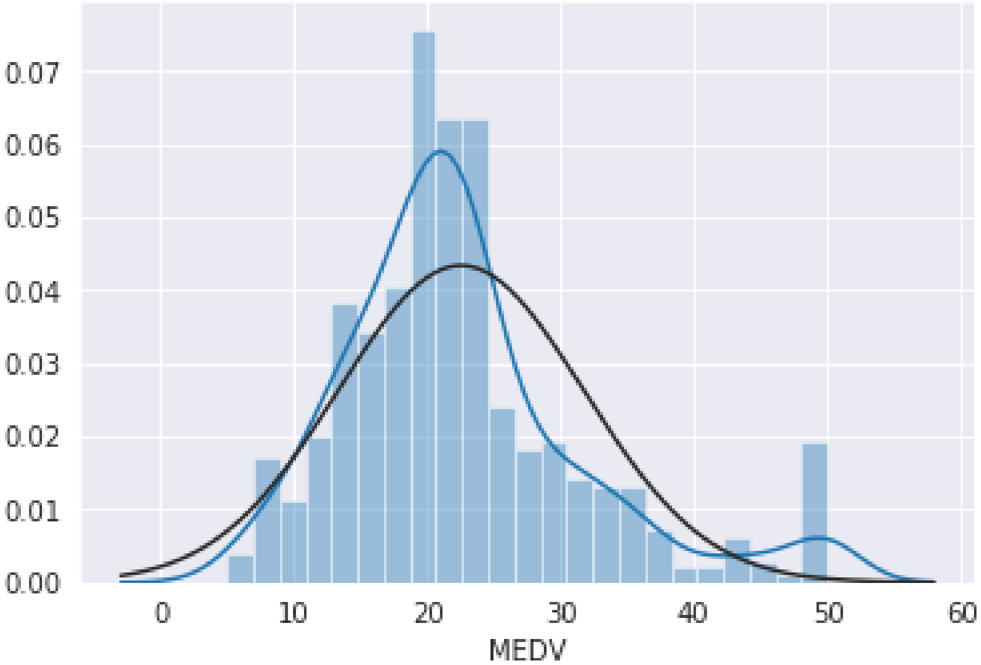
```
In [15]: df['MEDV'].describe()
```

```
Out[15]: count    506.000000
         mean      22.532806
         std        9.197104
         min        5.000000
         25%       17.025000
         50%       21.200000
         75%       25.000000
         max       50.000000
         Name: MEDV, dtype: float64
```

There is no zero price in here.

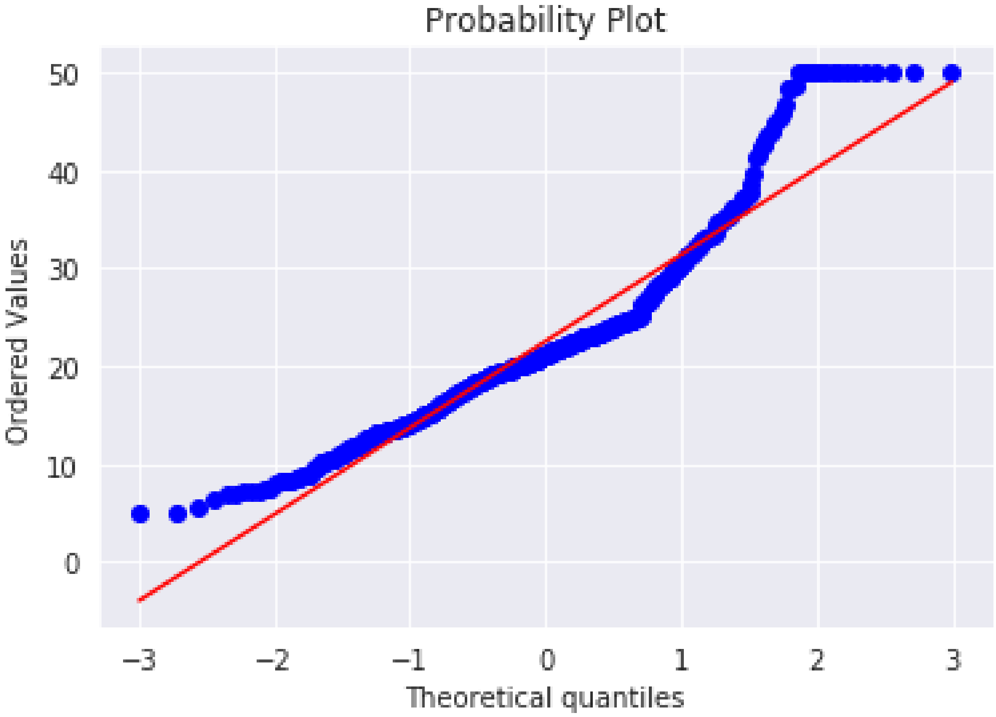
```
In [16]: sns.distplot(df['MEDV'], fit=norm)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f497de41160>
```



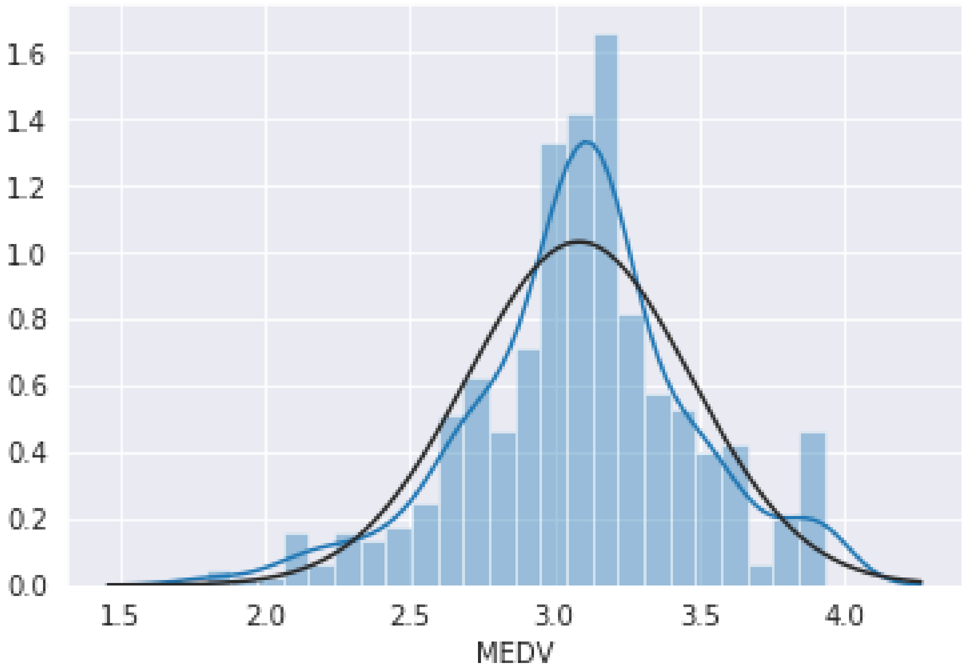
Overall, it is normally distributed. There may be some outliers with around $50,000 price houses.

```
In [17]: qq = stats.probplot(df['MEDV'], plot=plt)
```

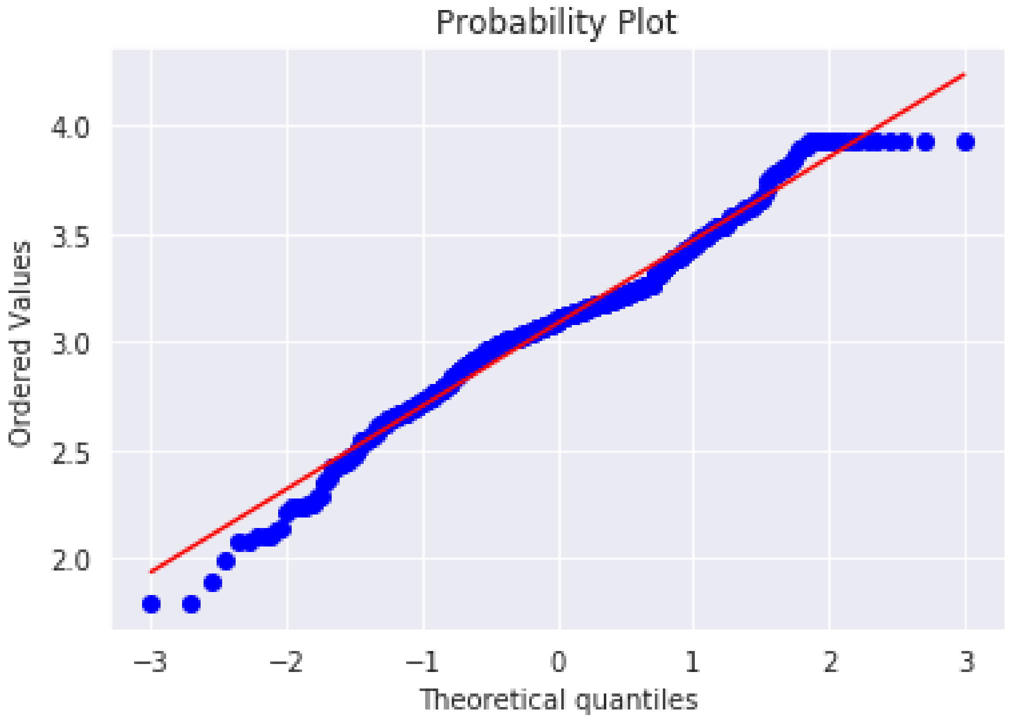


In a QQ plot, It looks the target variable is skewed, so it needs transformation to make it more normally distributed.

```
In [18]: # to reverse : np.expm1()
         test1 = np.log1p(df['MEDV'])

In [19]: sns.distplot(test1 , fit=norm)

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f497dd0e978>
```



```
In [20]: new_qq = stats.probplot(test1, plot=plt)
```



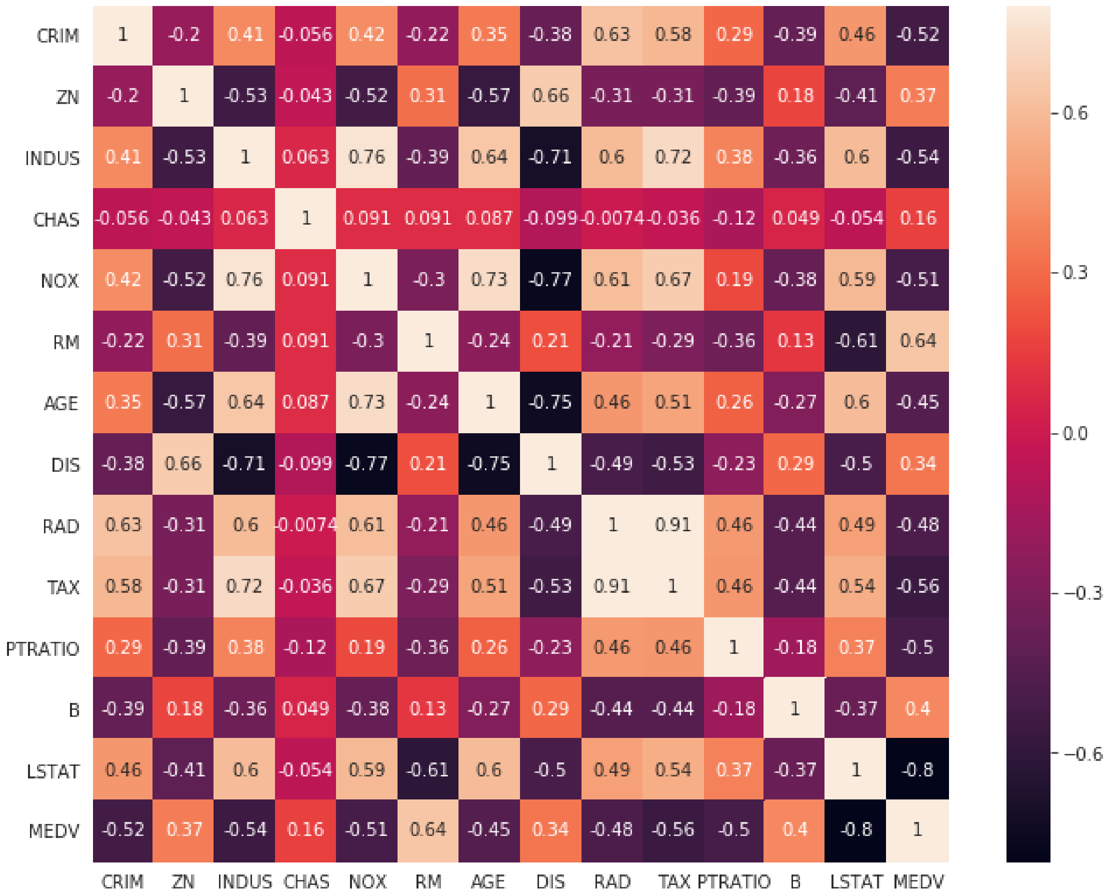
```
In [21]: print(df['MEDV'].skew(), df['MEDV'].kurt())
         print(test1.skew(), test1.kurt())

1.1080984082549072 1.495196944165818
-0.2412435236076338 0.6740533238972457
```

After log transformation, the skewness and kurtosis are decreased.

```
In [22]: df['MEDV'] = np.log1p(df['MEDV'])
```

1.2 Features

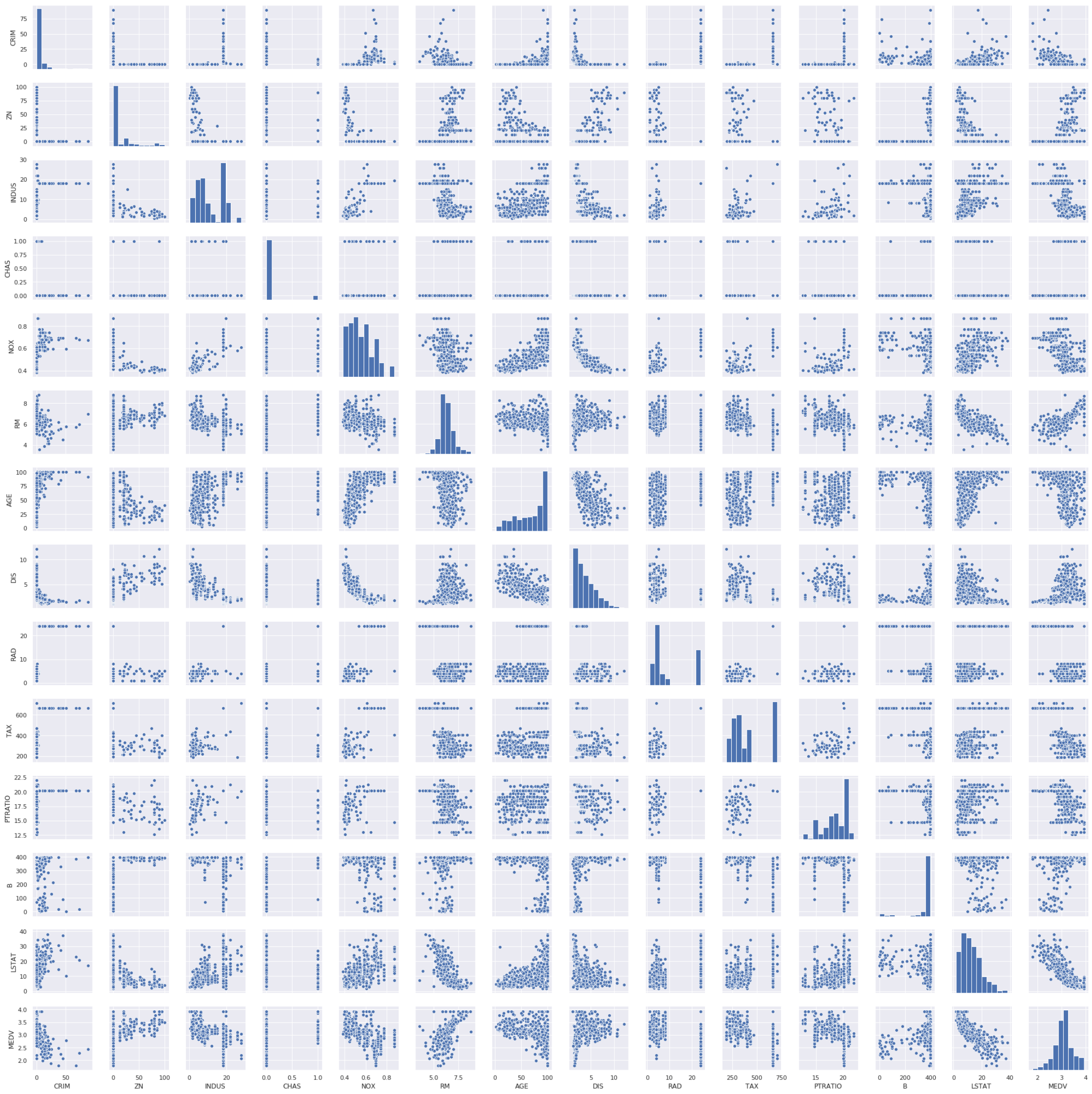
```
In [23]: #correlation matrix
         corrmat = df.corr()
         f, ax = plt.subplots(figsize=(12, 9))
         sns.heatmap(corrmat, annot=True, vmax=.8, square=True);
```



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.41	-0.056	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.52
ZN	-0.2	1	-0.53	-0.043	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.37
INDUS	0.41	-0.53	1	0.063	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.54
CHAS	-0.056	-0.043	0.063	1	0.091	0.091	0.087	-0.099	-0.0074	-0.036	-0.12	0.049	-0.054	0.16
NOX	0.42	-0.52	0.76	0.091	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.51
RM	-0.22	0.31	-0.39	0.091	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.64
AGE	0.35	-0.57	0.64	0.087	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.45
DIS	-0.38	0.66	-0.71	-0.099	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.34
RAD	0.63	-0.31	0.6	-0.0074	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.48
TAX	0.58	-0.31	0.72	-0.036	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.56
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.5
B	-0.39	0.18	-0.36	0.049	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.4
LSTAT	0.46	-0.41	0.6	-0.054	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.8
MEDV	-0.52	0.37	-0.54	0.16	-0.51	0.64	-0.45	0.34	-0.48	-0.56	-0.5	0.4	-0.8	1

It seems like 'LSTAT' and 'RM' have a strong relationship with the house prices.

```
In [30]: #sns.set()
         #sns.pairplot(df,)
         sns.pairplot(df, size=2.0)

Out[30]: <seaborn.axisgrid.PairGrid at 0x7f49754e8128>
```

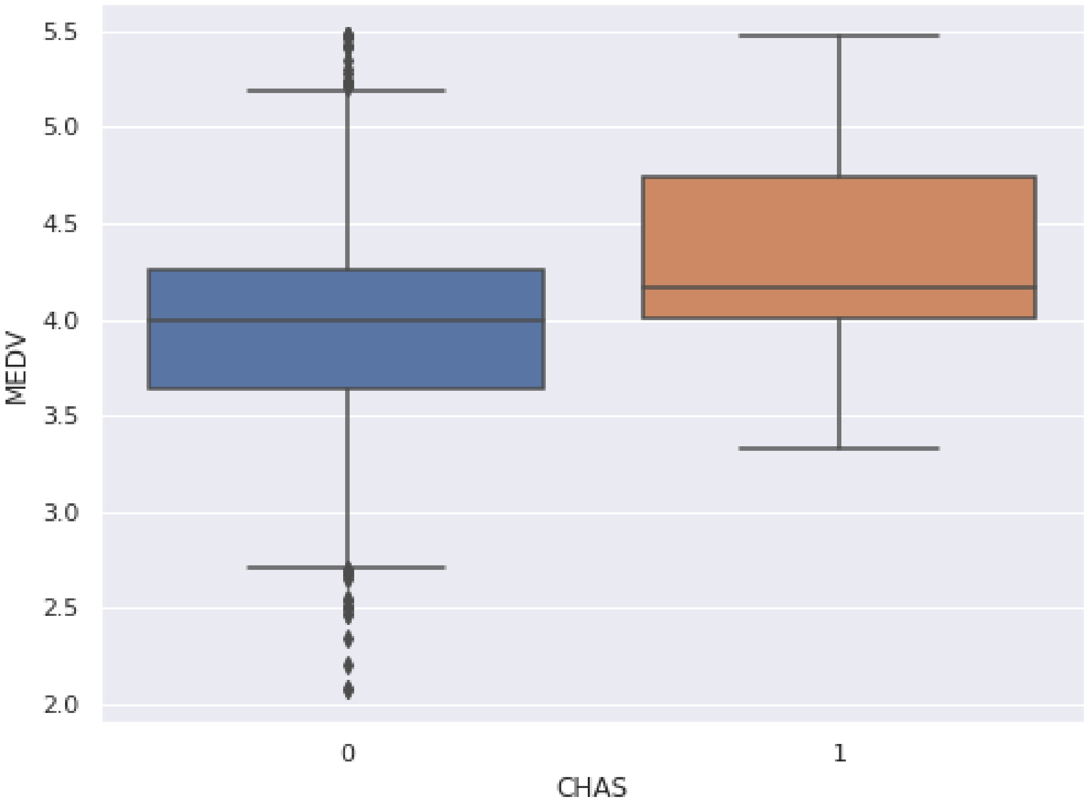


```
In [32]: plt.show()
```

In a pair plot, 'NOX' and 'ZN' got my attention. 'ZN' has linear relationship with the house price and 'NOX' also has a linear relationship with the house price('MEDV').

1.3 Relationship with categorical features(CHAS)

```
In [45]: #box plot CHAS/MEDV
         data = pd.concat([df['MEDV'], df['CHAS']], axis=1)
         f, ax = plt.subplots(figsize=(8, 6))
         fig = sns.boxplot(x='CHAS', y='MEDV', data=data)
```



It seems that it is not that beautiful with our target MEDV, but I can live with that.

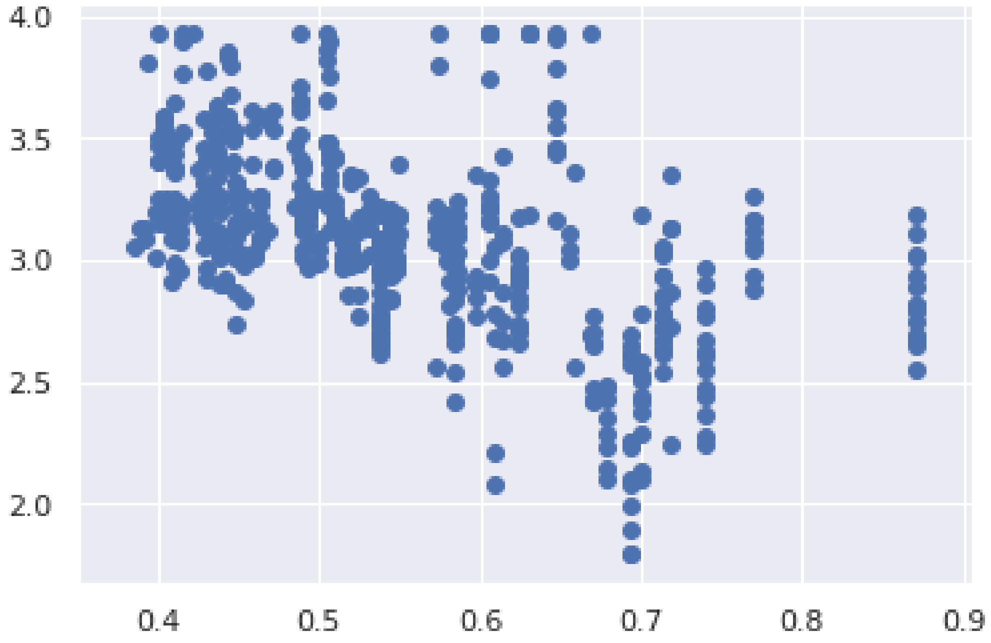
1.4 Relationship with numerical features

-

1.4.1 NOX(nitric oxides concentration (parts per 10 million))

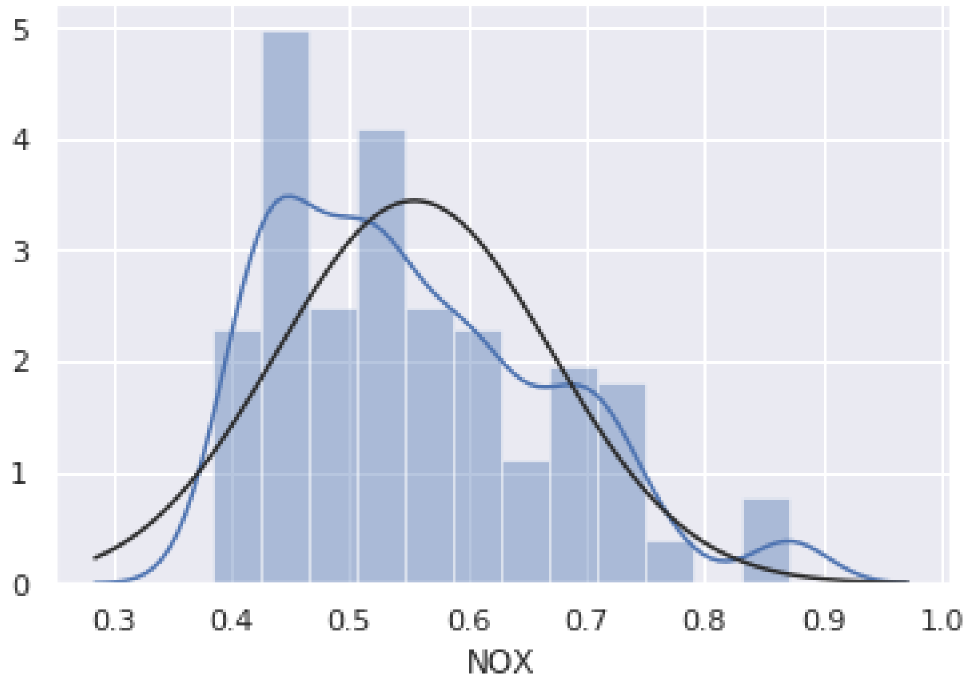
```
In [33]: plt.scatter(df['NOX'], df['MEDV'])
```

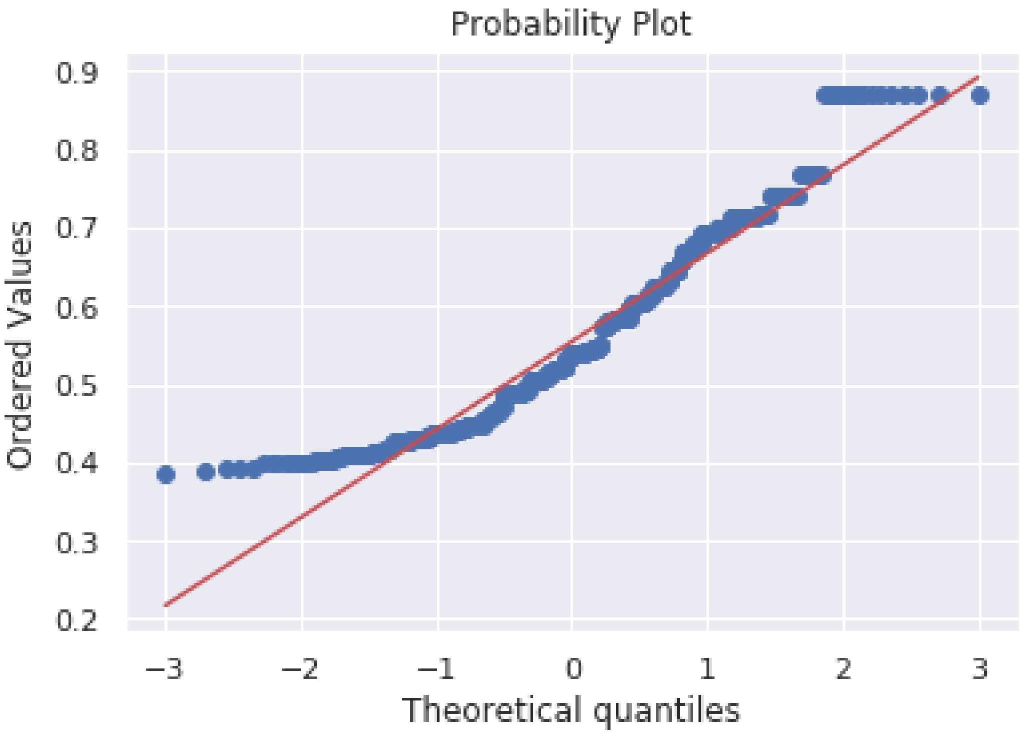
```
Out[33]: <matplotlib.collections.PathCollection at 0x7f496b3c0cf8>
```



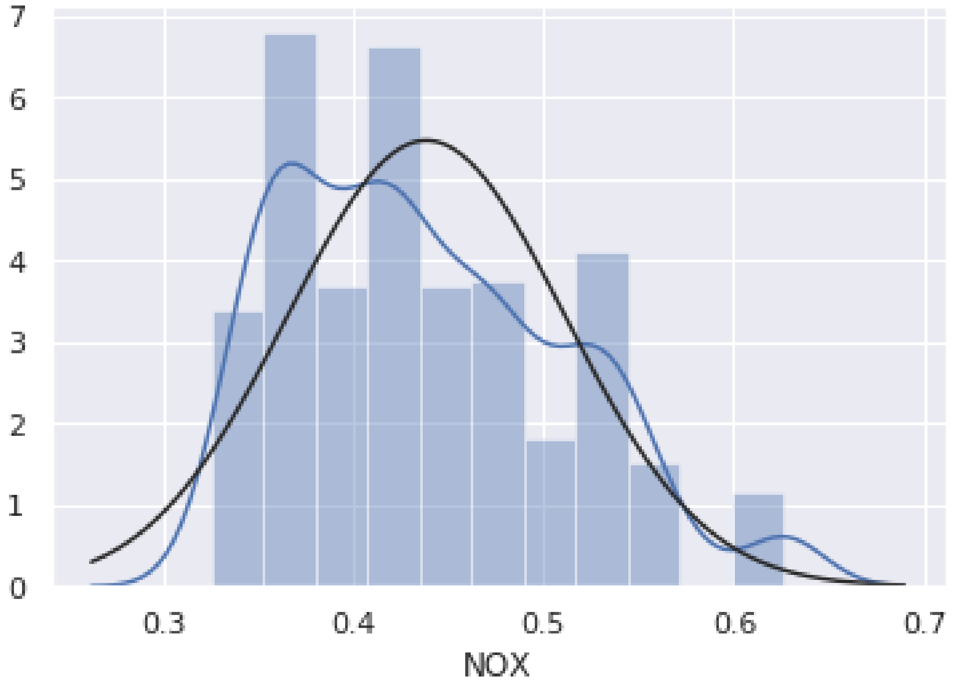
This is scatter plot between 'NOX' and house price. It looks like it got linear relationship.

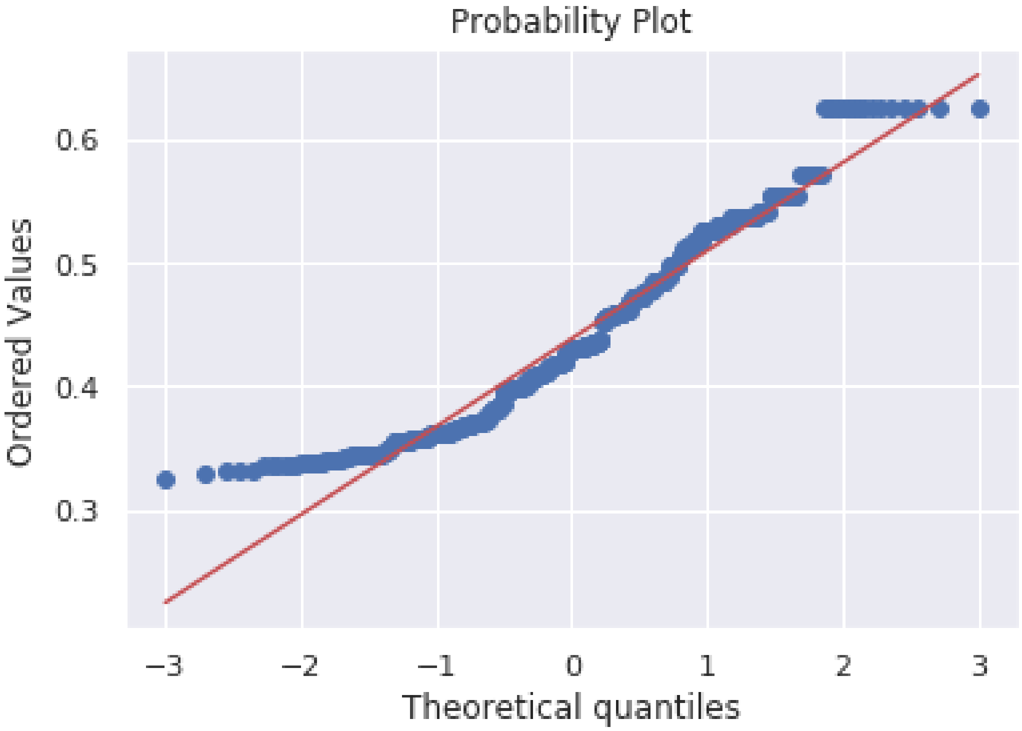
```
In [34]: #histogram and normal probability plot
         sns.distplot(df['NOX'], fit=norm);
         fig = plt.figure()
         res = stats.probplot(df['NOX'], plot=plt)
```



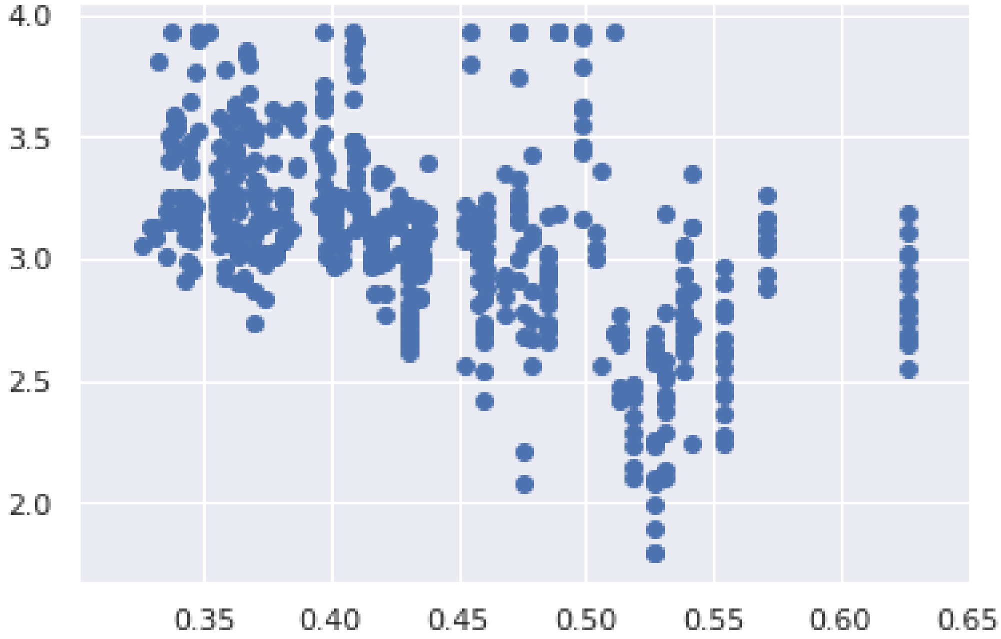


```
In [40]: test1 = np.log1p(df['NOX'])

In [41]: #histogram and normal probability plot
         sns.distplot(test1, fit=norm);
         fig = plt.figure()
         res = stats.probplot(test1, plot=plt)
```


7
6
5
4
3
2
1
0
0.3
0.4
0.5
0.6
0.7
NOX


Probability Plot
Ordered Values
0.6
0.5
0.4
0.3
−3
−2
−1
0
1
2
3
Theoretical quantiles

```
In [42]: plt.scatter(test1, df['MEDV'])

Out[42]: <matplotlib.collections.PathCollection at 0x7f496afe26d8>
```



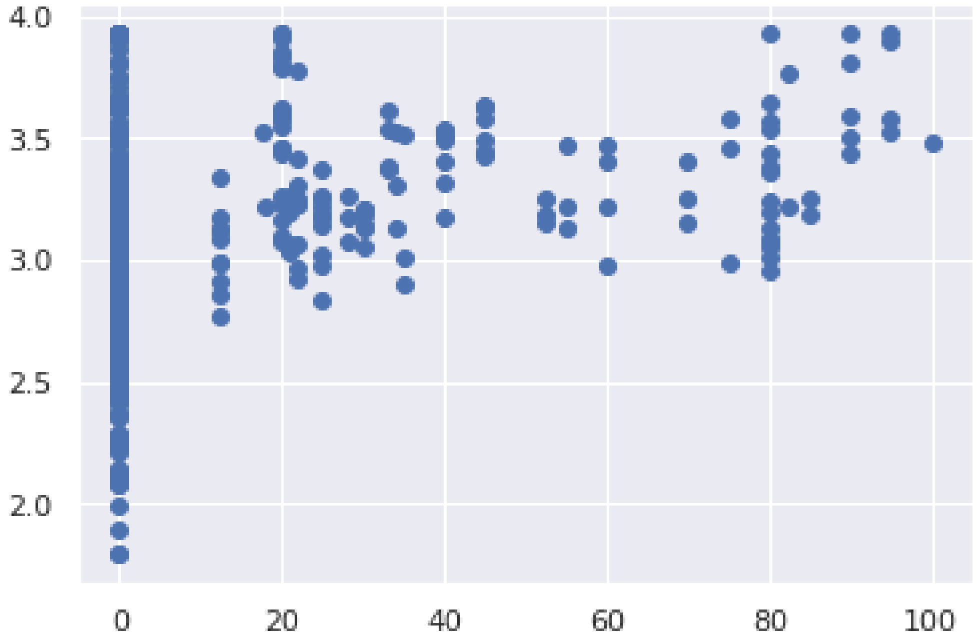
```
In [43]: print(df['NOX'].skew(), df['NOX'].kurt())
         print(test1.skew(), test1.kurt())

0.7293079225348787 -0.06466713336542629
0.5843471124349947 -0.3770771732950444
```

After log transformation, skewness is decreased, but kurtosis is increased. Log transformation did not help to make this more normally distributed. In the last of jupyter notebook, we are going to apply different transform to make it better.

-

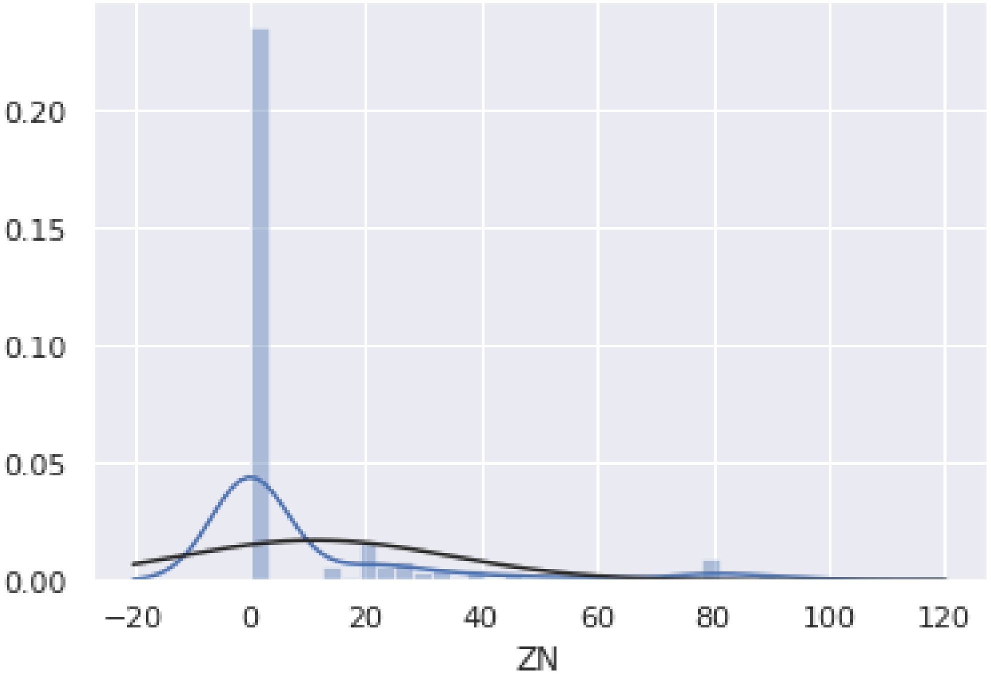
1.4.2 ZN (proportion of residential land zoned for lots over 25,000 sq.ft.)

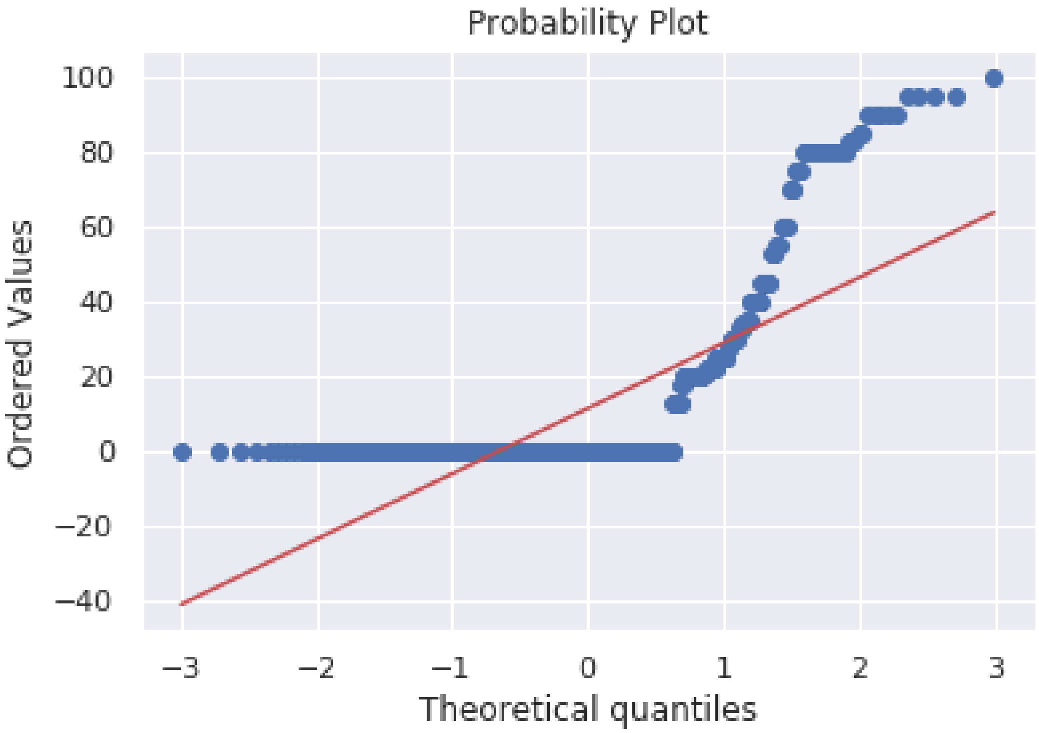
```
In [44]: plt.scatter(df['ZN'], df['MEDV'])

Out[44]: <matplotlib.collections.PathCollection at 0x7f496b075dd8>
```



'ZN' has some linear relationship with 'MEDV',but it has many zeros.

```
In [202]: #histogram and normal probability plot
          sns.distplot(df['ZN'], fit=norm);
          fig = plt.figure()
          res = stats.probplot(df['ZN'], plot=plt)
```





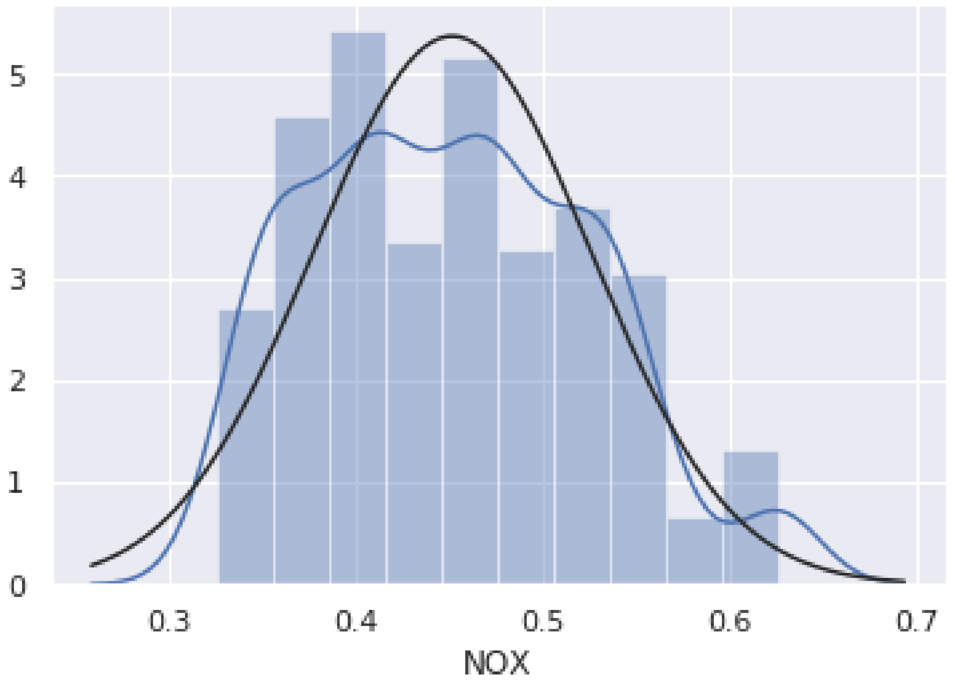
```
In [203]: df['ZN'].describe()
```

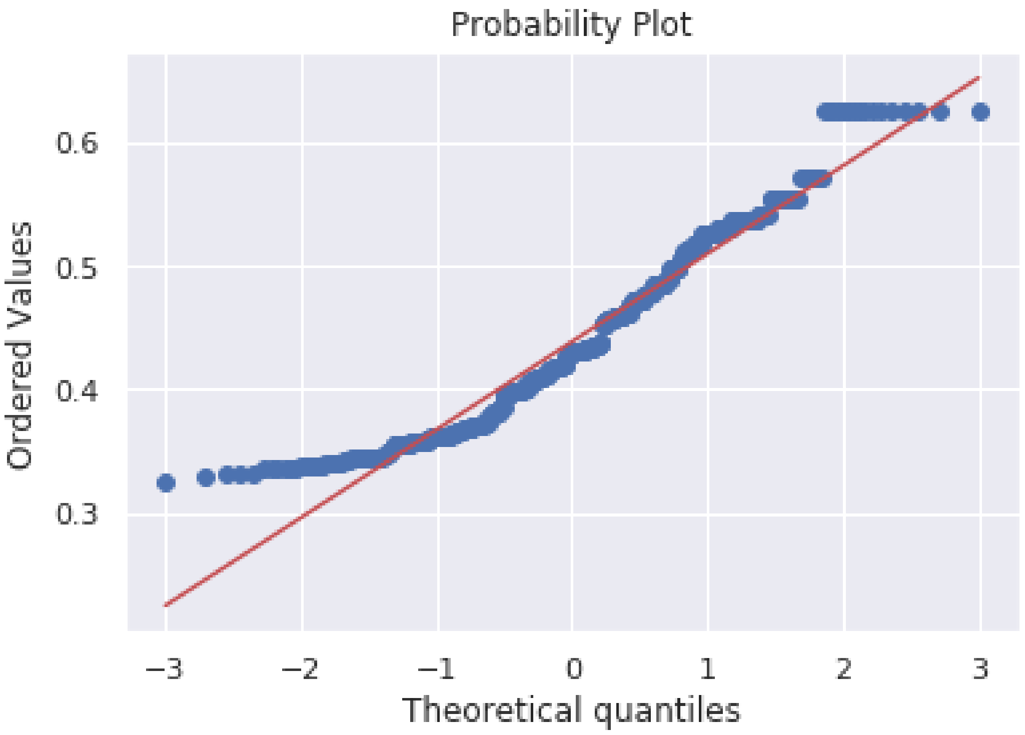
```
Out[203]: count    506.000000
          mean      11.363636
          std       23.322453
          min        0.000000
          25%        0.000000
          50%        0.000000
          75%       12.500000
          max      100.000000
          Name: ZN, dtype: float64
```

As we can see, there are over 50% of zeros here, we can make is categorical values, but we don't want to lose other numerical values, so I will keep the zeros.

```
In [204]: test1 = np.log1p(df['ZN'])
```

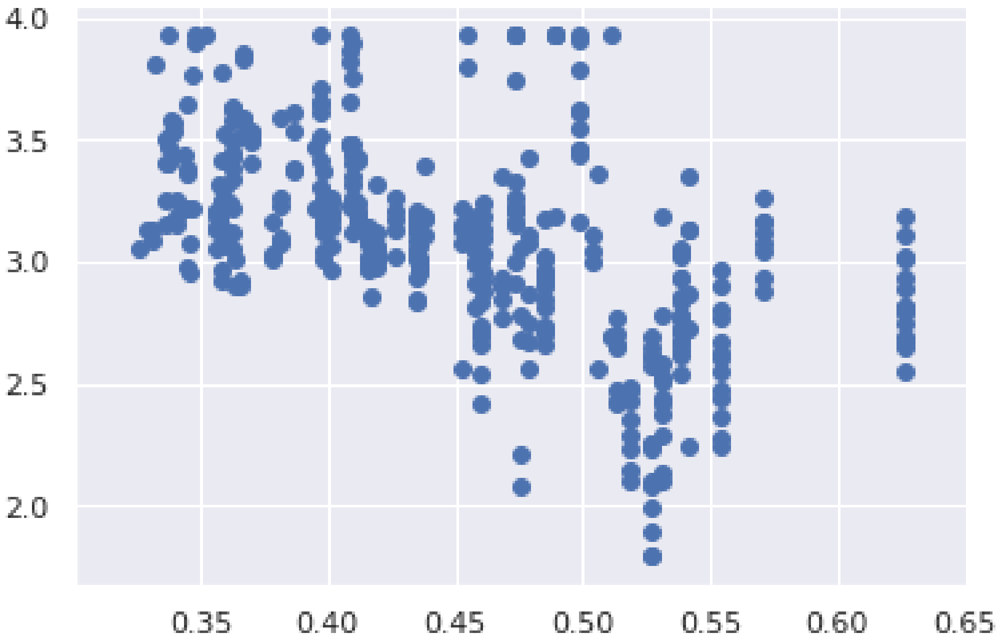
```
In [47]: #histogram and normal probability plot
         sns.distplot(test1[int(len(test1)/5):], fit=norm);
         fig = plt.figure()
         res = stats.probplot(test1, plot=plt)
```


5
4
3
2
1
0
0.3
0.4
0.5
0.6
0.7
NOX


Probability Plot
Ordered Values
0.6
0.5
0.4
0.3
−3
−2
−1
0
1
2
3
Theoretical quantiles

```
In [50]: plt.scatter(test1[int(len(test1)/5):], df['MEDV'][int(len(test1)/5):])
```

```
Out[50]: <matplotlib.collections.PathCollection at 0x7f496ae6b518>
```



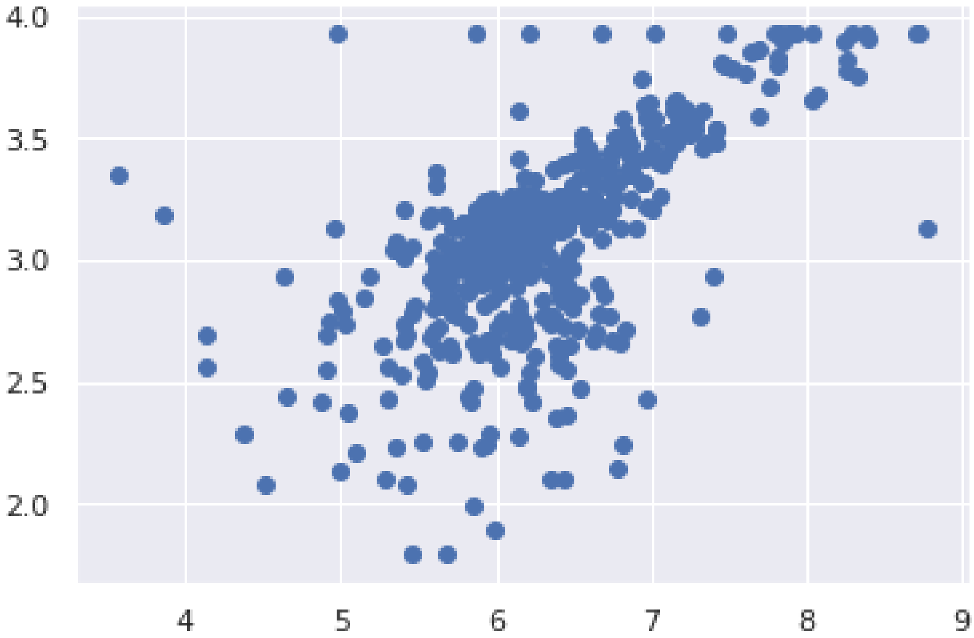
After log transformation, except zeros, numerical values now have some linear relationship with the house price.

-

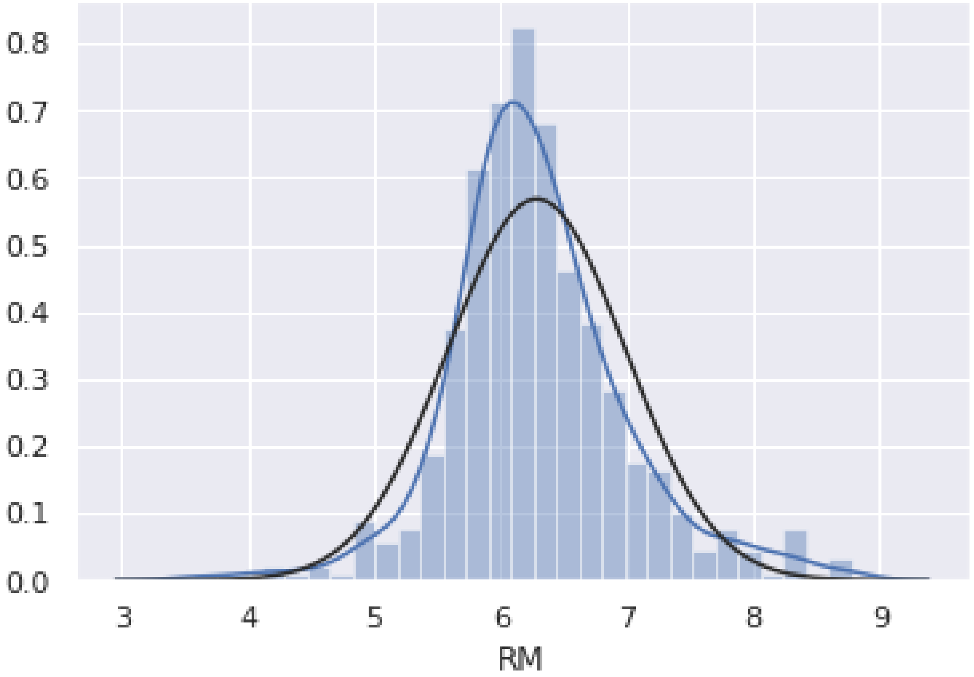
1.4.3 RM (average number of rooms per dwelling)

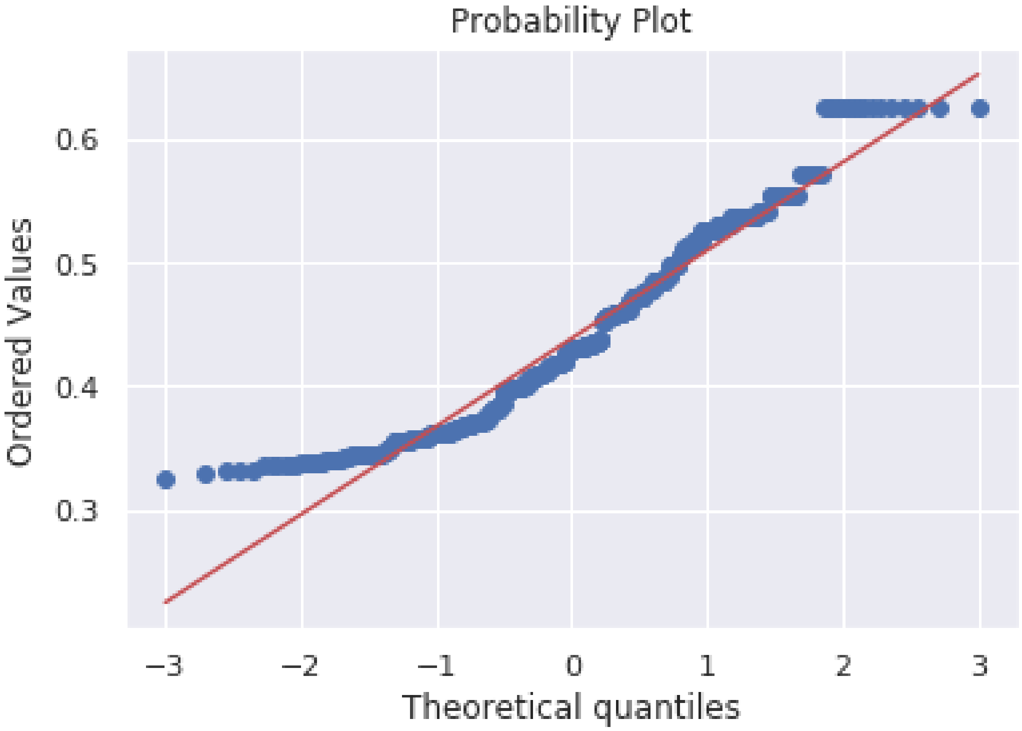
```
In [51]: plt.scatter(df['RM'], df['MEDV'])
```

```
Out[51]: <matplotlib.collections.PathCollection at 0x7f496ac55588>
```

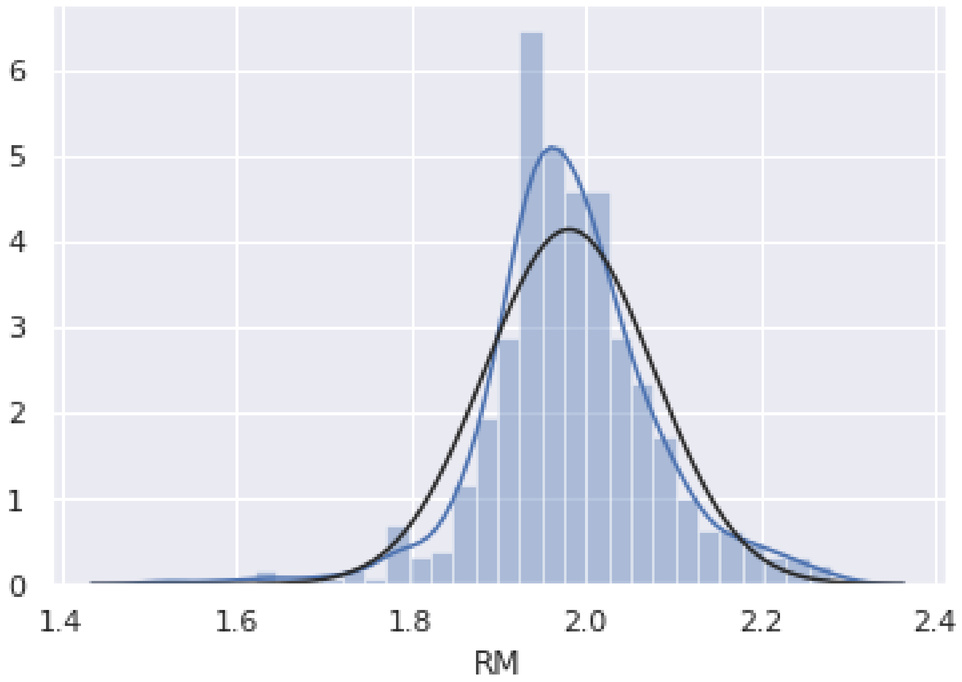


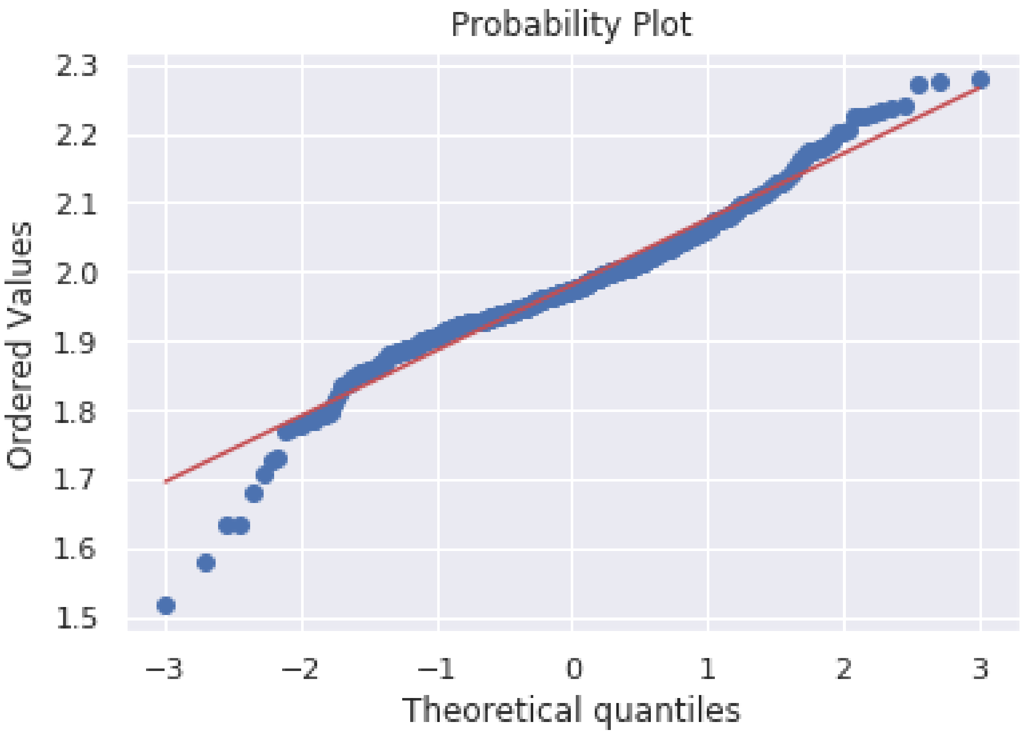
```
In [52]: #histogram and normal probability plot
         sns.distplot(df['RM'], fit=norm);
         fig = plt.figure()
         res = stats.probplot(test1, plot=plt)
```



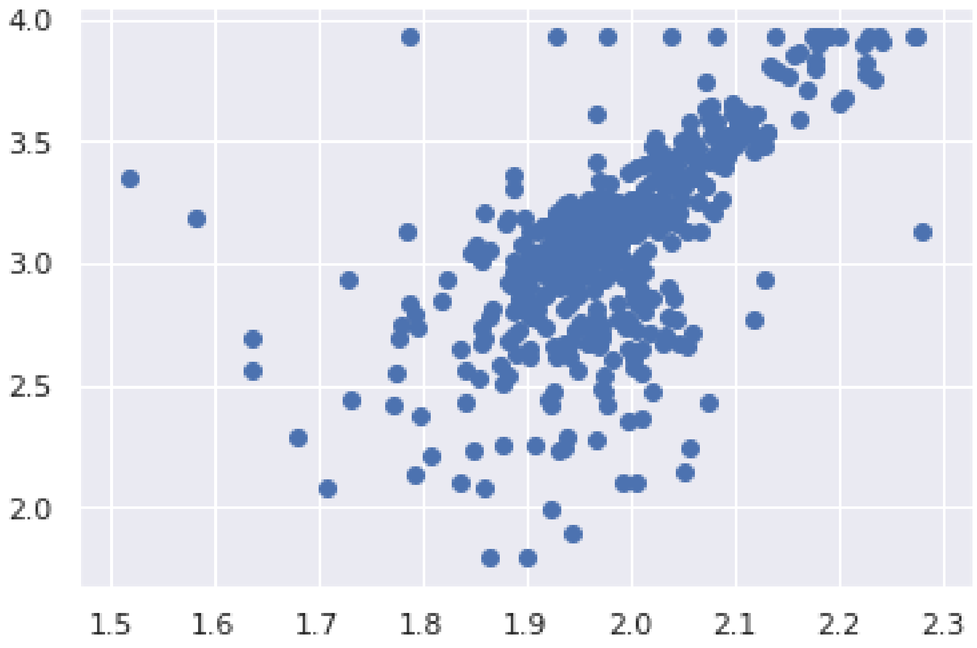


```
In [53]: test1 = np.log1p(df['RM'])

In [54]: #histogram and normal probability plot
         sns.distplot(test1, fit=norm);
         fig = plt.figure()
         res = stats.probplot(test1, plot=plt)
```


6
5
4
3
2
1
0
1.4
1.6
1.8
2.0
2.2
2.4
RM


Probability Plot
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.6
1.5
Ordered Values
−3
−2
−1
0
1
2
3
Theoretical quantiles

```
In [55]: plt.scatter(test1, df['MEDV'])

Out[55]: <matplotlib.collections.PathCollection at 0x7f496aa22e10>
```

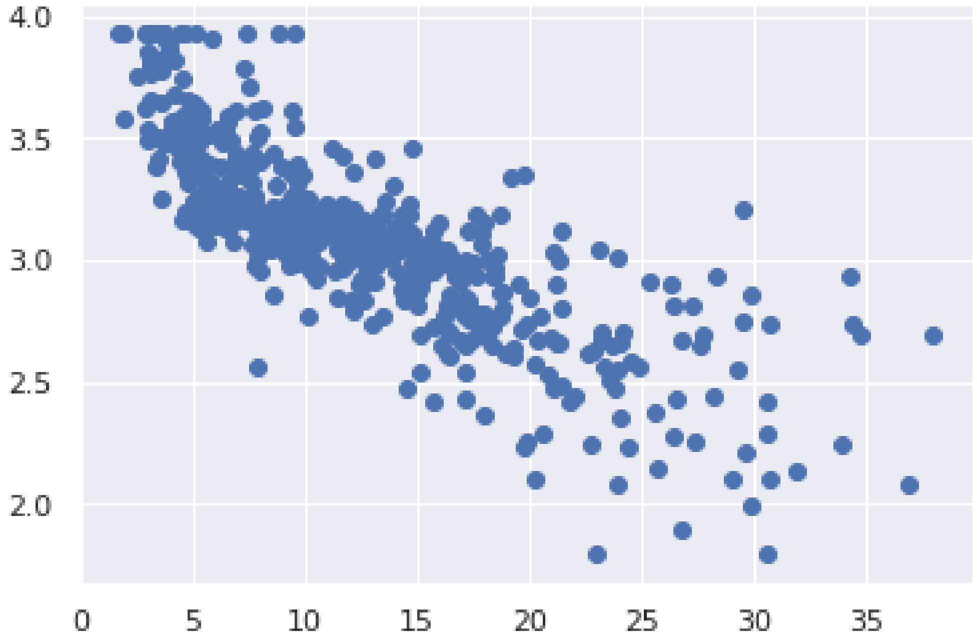


'RM' certainly has a strong relationship with the house price and it is normally distributed. After log transformation, we can see it fits more than before in a QQ plot.

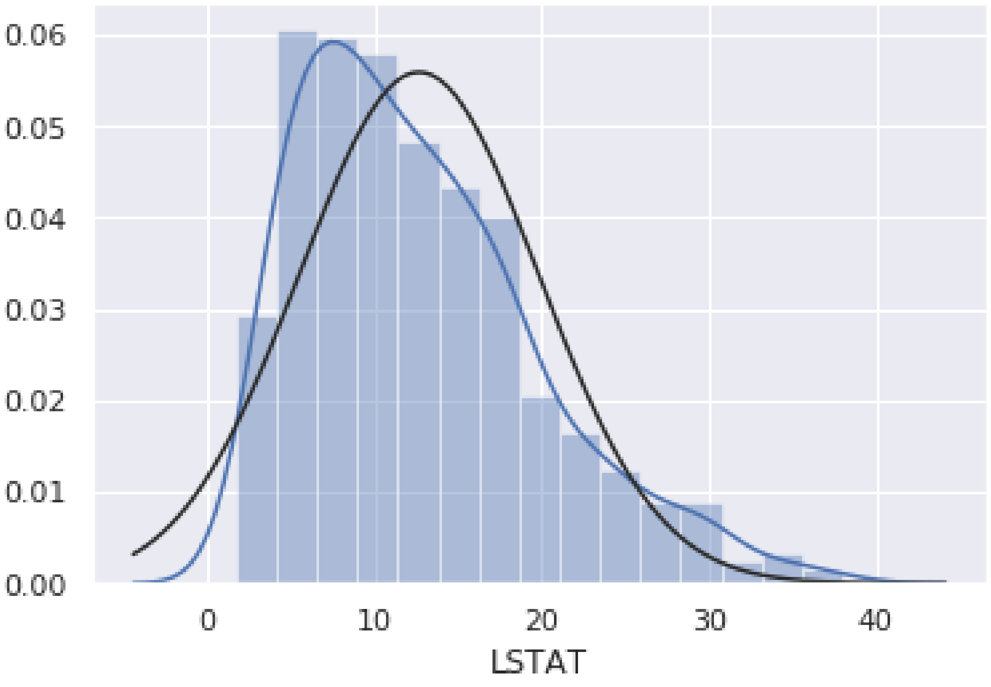
-

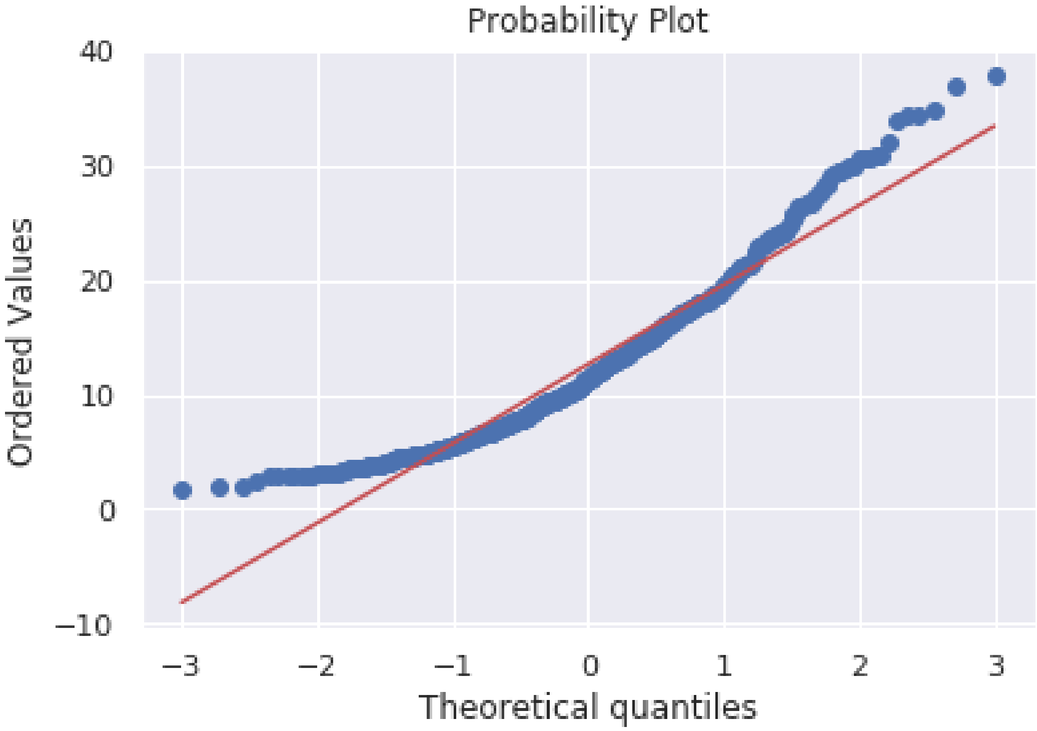
1.4.4 LSTAT (% lower status of the population)

```
In [56]: plt.scatter(df['LSTAT'], df['MEDV'])

Out[56]: <matplotlib.collections.PathCollection at 0x7f496aa017f0>
```



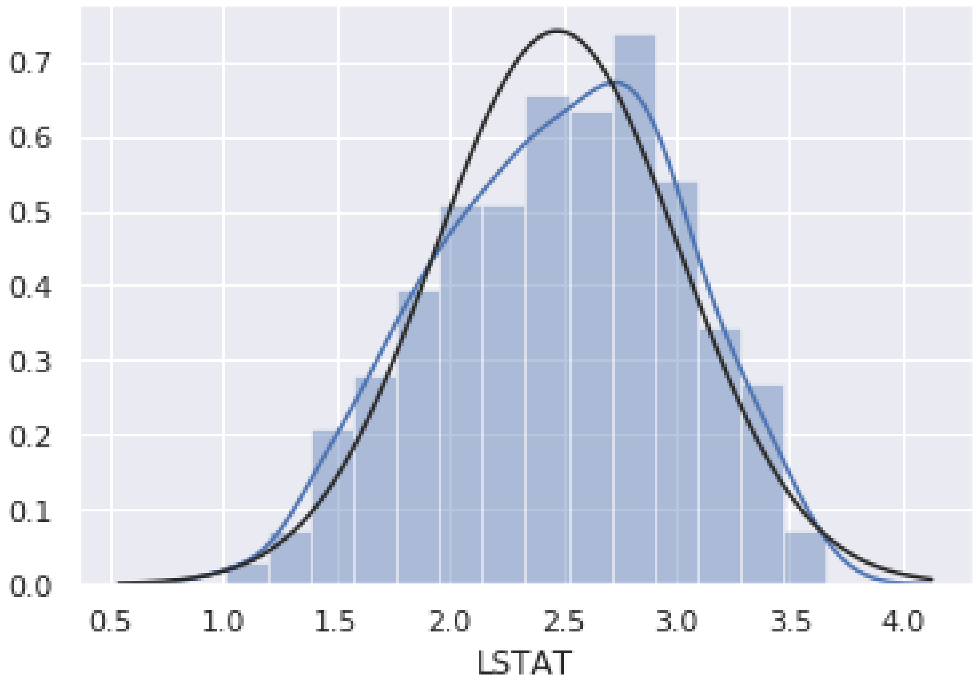
```
In [57]: sns.distplot(df['LSTAT'], fit=norm);
         fig = plt.figure()
         res = stats.probplot(df['LSTAT'], plot=plt)
```

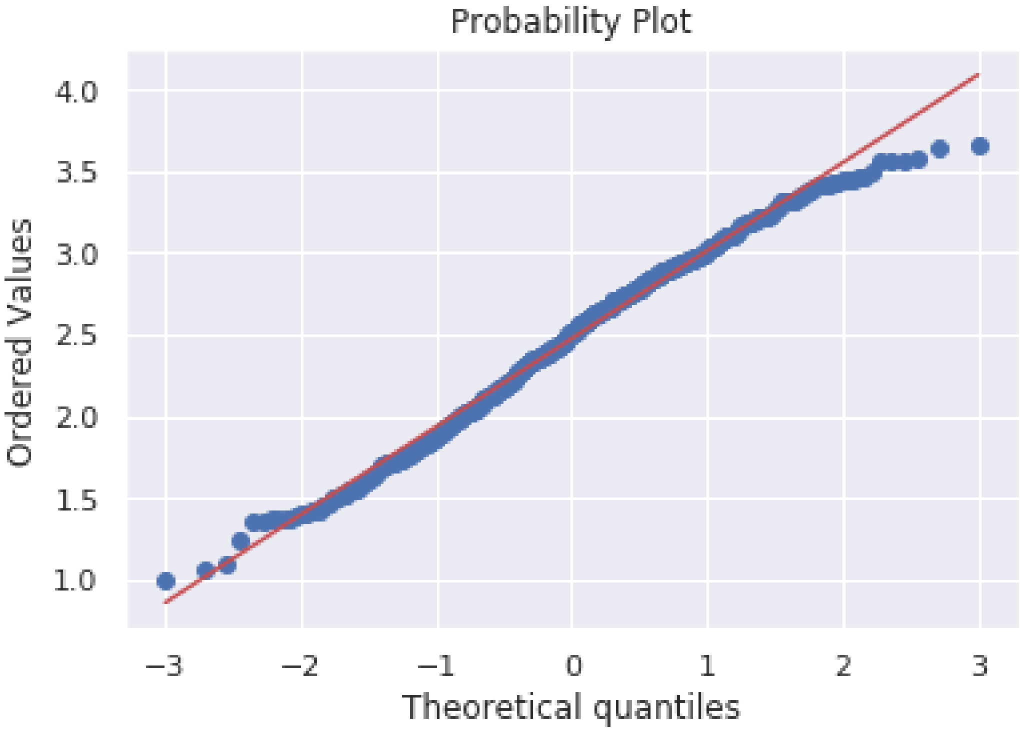




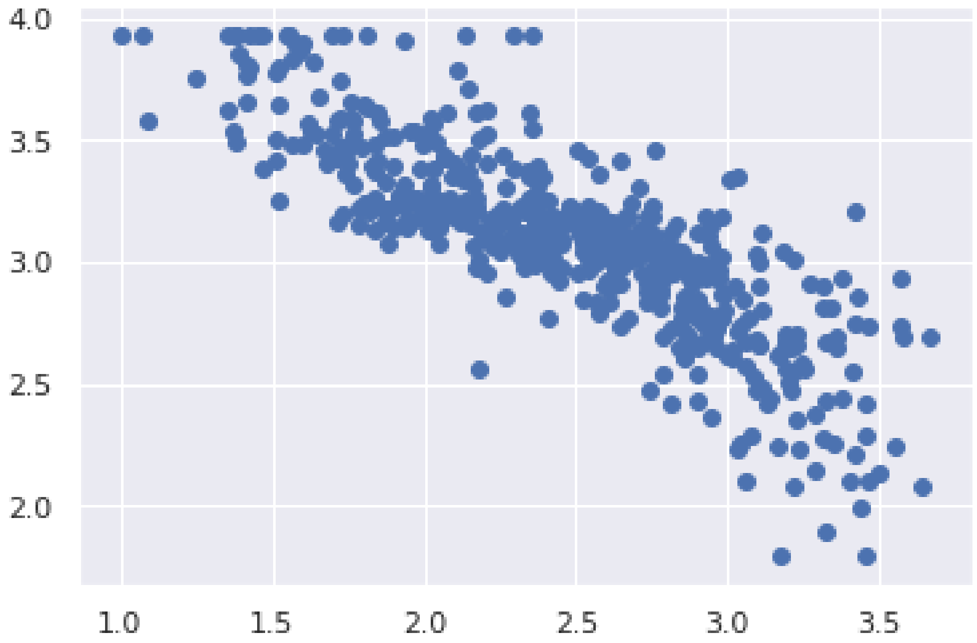
We already have seen this strong relationship in a heatmap. As we can see, it has a strong relationship with the house price and QQ plot looks okay.

```
In [59]: test1 = np.log1p(df['LSTAT'])

In [60]: #histogram and normal probability plot
         sns.distplot(test1, fit=norm);
         fig = plt.figure()
         res = stats.probplot(test1, plot=plt)
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
LSTAT


Probability Plot
Ordered Values
4.0
3.5
3.0
2.5
2.0
1.5
1.0
−3
−2
−1
0
1
2
3
Theoretical quantiles

```
In [61]: plt.scatter(test1, df['MEDV'])

Out[61]: <matplotlib.collections.PathCollection at 0x7f496a9575c0>
```



Clearly, log transformation made 'LSTAT' more normaly distributed and got sronger relationship with the house price.

1.5 Feature Engineering

In our dataset, we don't have any missing values and just have one categorical value. What I can do is to apply log transformation to make our data more normally distributed, but there is another good transformation method like log named boxcox transformation. We will just pick highly skewed data and apply boxcox transformation.

```
In [62]: df['MEDV'] = np.log1p(df['MEDV'])
         for i in df.columns[:-1]:
             print(i)
             print(df[i].skew(), df[i].kurt())
             if i != 'CHAS' and (abs(df[i].skew()) > 0.75):
                 df[i] = boxcox1p(df[i], 0.15)
                 print(df[i].skew(), df[i].kurt())

             print('--------------------------')
```

```
CRIM
5.223148798243851 37.13050912952203
1.5087823936419944 1.6054359340291966
```

```
--------------------------
ZN
2.2256663227354307 4.031510083739155
1.2734919266490703 -0.08987364189668012
--------------------------
INDUS
0.29502156787351164 -1.2335396011495188
--------------------------
CHAS
3.405904172058746 9.638263777819526
--------------------------
NOX
0.7293079225348787 -0.06466713336542629
--------------------------
RM
0.40361213328874385 1.8915003664993404
--------------------------
AGE
-0.5989626398812962 -0.9677155941626912
--------------------------
DIS
1.0117805793009007 0.4879411222443908
0.4244094298892784 -0.7983866388790837
--------------------------
RAD
1.0048146482182057 -0.8672319936034931
0.6621327207579614 -0.9106709679989304
--------------------------
TAX
0.669955941795016 -1.1424079924768082
--------------------------
PTRATIO
-0.8023249268537809 -0.28509138330538875
-0.9715716337361822 0.12043864877940402
--------------------------
B
-2.8903737121414492 7.226817549260753
-3.986986233095778 16.27898690430474
--------------------------
LSTAT
0.9064600935915367 0.49323951739272776
-0.021989638291353485 -0.6365237855998052
--------------------------
```

```
In [ ]:
```