Boston_Housing_Price_Prediction

February 27, 2019

1

- **OS**: Ubuntu 18.04.2 LTS
- **Language**: Python 3.7.2
- **Library**:
- numpy 1.16.1
- pandas 0.24.1
- sklearn 0.0.9
- matplotlib 3.0.2

```
In [1]: from datetime import datetime
        import time

        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import tensorflow as tf

        from sklearn import linear_model
        from sklearn import datasets
        from sklearn.svm import l1_min_c
        from sklearn.linear_model import LassoCV
        import sklearn.model_selection
```

2

- raw data : rawdata
- : txt file
- :

```
In [4]: ##
        rawdata = pd.read_csv("./housing_data.txt", sep="\s+", header=None)
        ## (column)
        rawdata.columns = ["CRIM", "ZN", "INDUS", "CHAS","NOX","RM","AGE","DIS","RAD","TAX","P
```

```
In [5]: #
        rawdata.head(10)
```

Out[5]:
```
       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
5  0.02985   0.0   2.18     0  0.458  6.430   58.7  6.0622    3  222.0
6  0.08829  12.5   7.87     0  0.524  6.012   66.6  5.5605    5  311.0
7  0.14455  12.5   7.87     0  0.524  6.172   96.1  5.9505    5  311.0
8  0.21124  12.5   7.87     0  0.524  5.631  100.0  6.0821    5  311.0
9  0.17004  12.5   7.87     0  0.524  6.004   85.9  6.5921    5  311.0

   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
5     18.7  394.12   5.21  28.7
6     15.2  395.60  12.43  22.9
7     15.2  396.90  19.15  27.1
8     15.2  386.63  29.93  16.5
9     15.2  386.71  17.10  18.9
```

In [6]:
```python
#     n = 506
#     p = 14
n, p = rawdata.shape
```

2.1 ,

- Training data : 404 row(80%)
- Test data : 102 row(20%)

In [72]:
```python
# X, Y data
x_raw = rawdata.iloc[:, 0:13]
y_raw = rawdata.iloc[:, 13]

train_x, test_x, train_y, test_y = sklearn.model_selection.train_test_split(x_raw, y_
```

2.1.1 null , null

In [108]:
```python
train_x.isnull().sum() # train_x
```

Out[108]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
```

```
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
dtype: int64
```

In [109]:
```
train_y.isnull().sum() # train_y
```

Out[109]:
```
0
```

In [110]:
```
test_x.isnull().sum() # text_x
```

Out[110]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
dtype: int64
```

In [111]:
```
test_y.isnull().sum() # test_y
```

Out[111]:
```
0
```

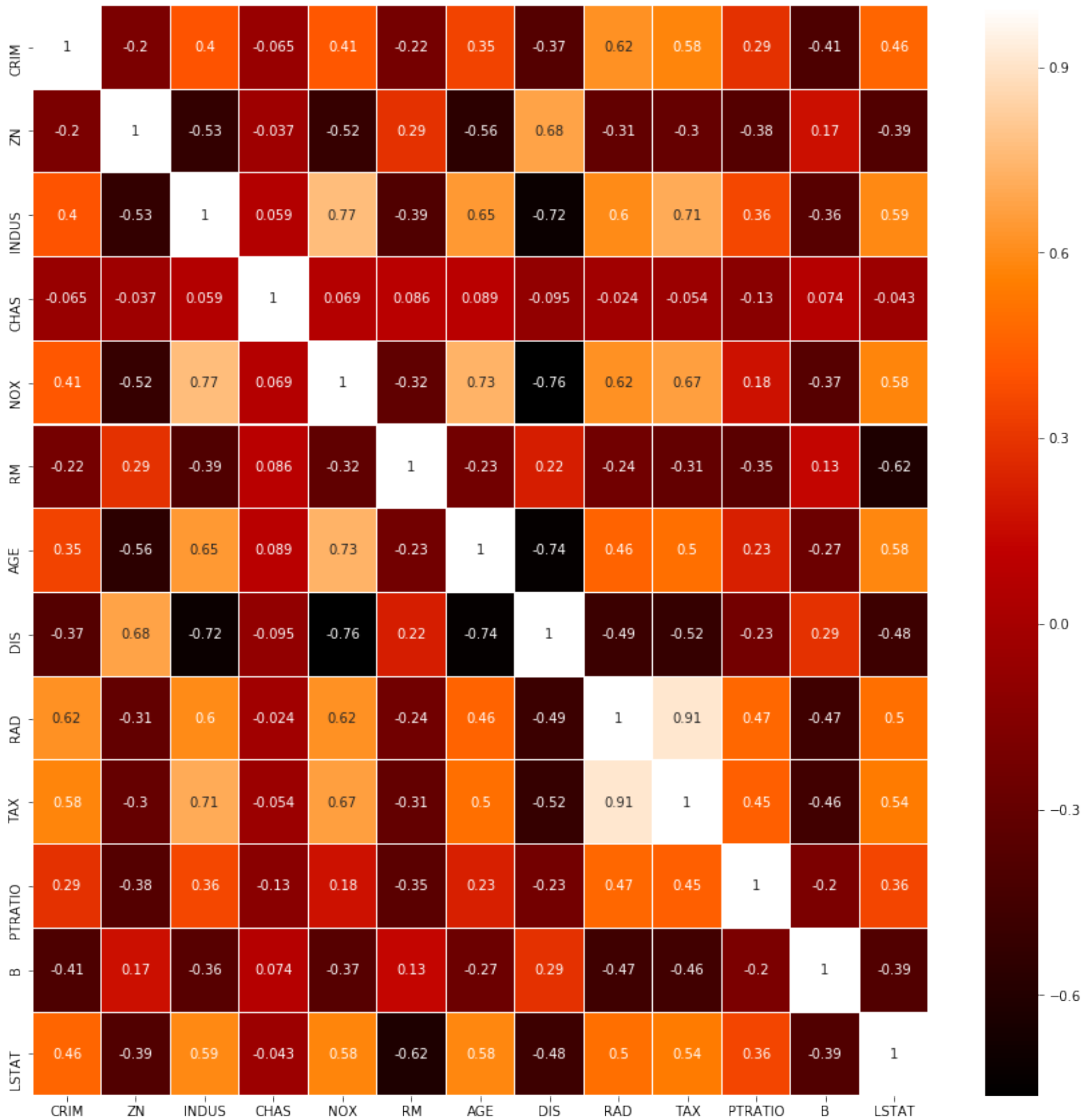
3

In [112]:
```
train_x.describe() #
```

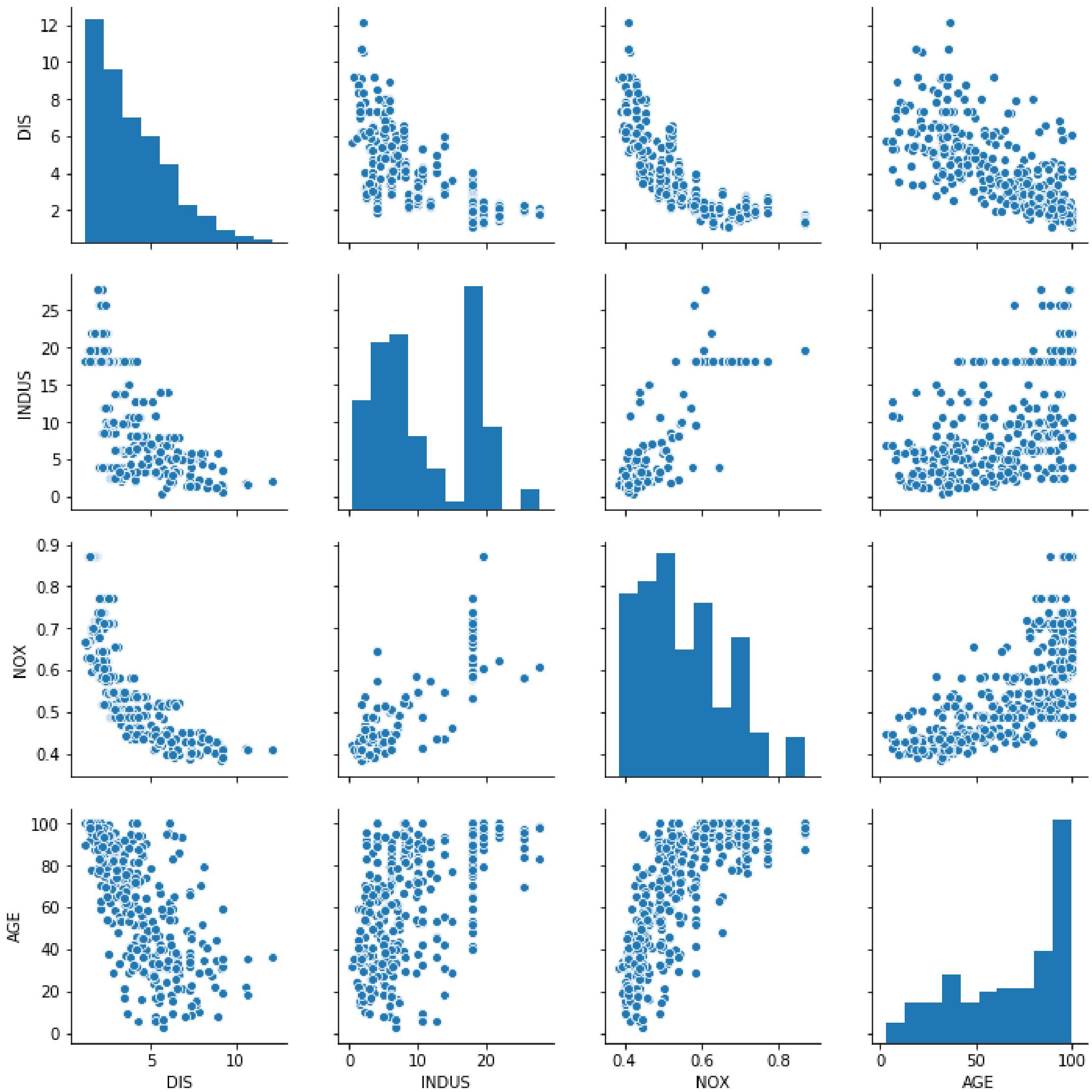
Out[112]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  404.000000  404.000000  404.000000  404.000000  404.000000  404.000000
mean     3.262956   11.733911   10.954356    0.076733    0.552274    6.301433
std      8.052195   23.710472    6.878348    0.266497    0.116741    0.726747
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.071615    0.000000    4.950000    0.000000    0.448000    5.888000
50%      0.227290    0.000000    8.560000    0.000000    0.524000    6.229500
75%      2.904685   12.500000   18.100000    0.000000    0.624000    6.635000
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000
```

```
              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  404.000000  404.000000  404.000000  404.000000  404.000000  404.000000
mean    67.677723    3.831537    9.269802  401.368812   18.383416  359.642599
std     28.435612    2.099385    8.636812  167.482335    2.157794   86.938206
min      2.900000    1.129600    1.000000  187.000000   12.600000    3.500000
25%     42.050000    2.111750    4.000000  277.000000   17.225000  376.745000
50%     76.500000    3.298600    5.000000  329.000000   18.700000  391.565000
75%     93.950000    5.218725   12.000000  666.000000   20.200000  396.307500
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT
count  404.000000
mean    12.356881
std      7.275672
min      1.730000
25%      6.720000
50%     10.530000
75%     16.457500
max     37.970000
```

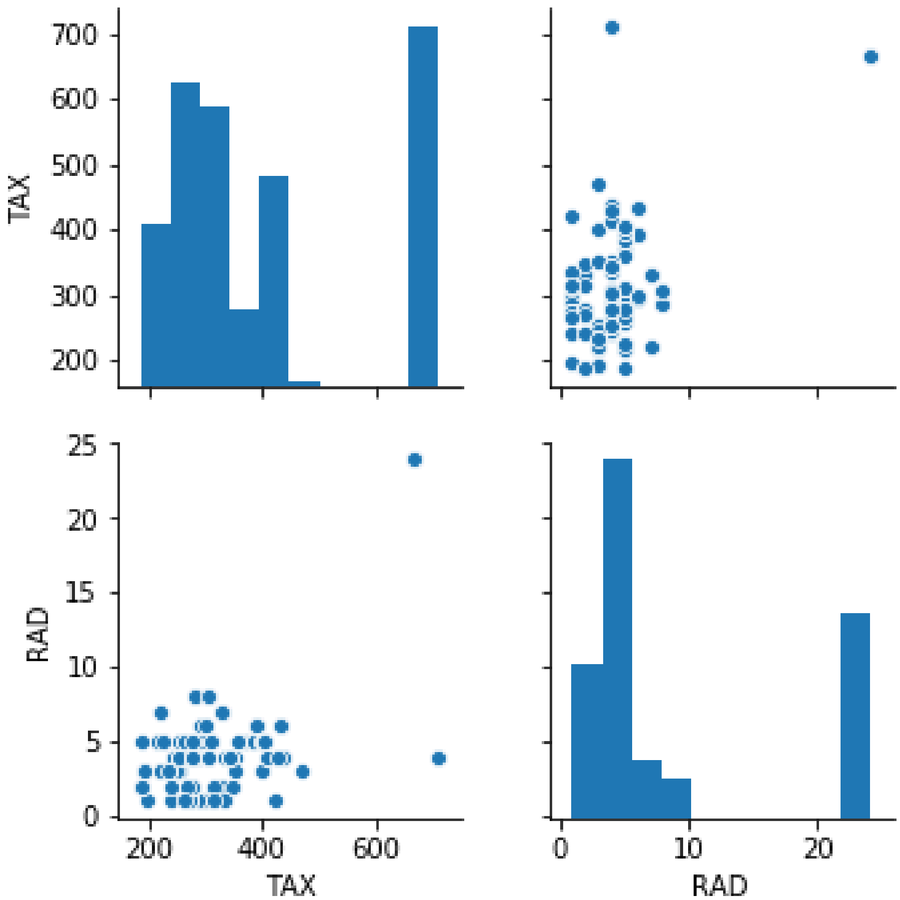
In [176]:
```
plt.figure(figsize=(15, 15))
sns.heatmap(train_x.corr(), linewidths=0.1, vmax=1, cmap=plt.cm.gist_heat, linecolor=
plt.show()
```



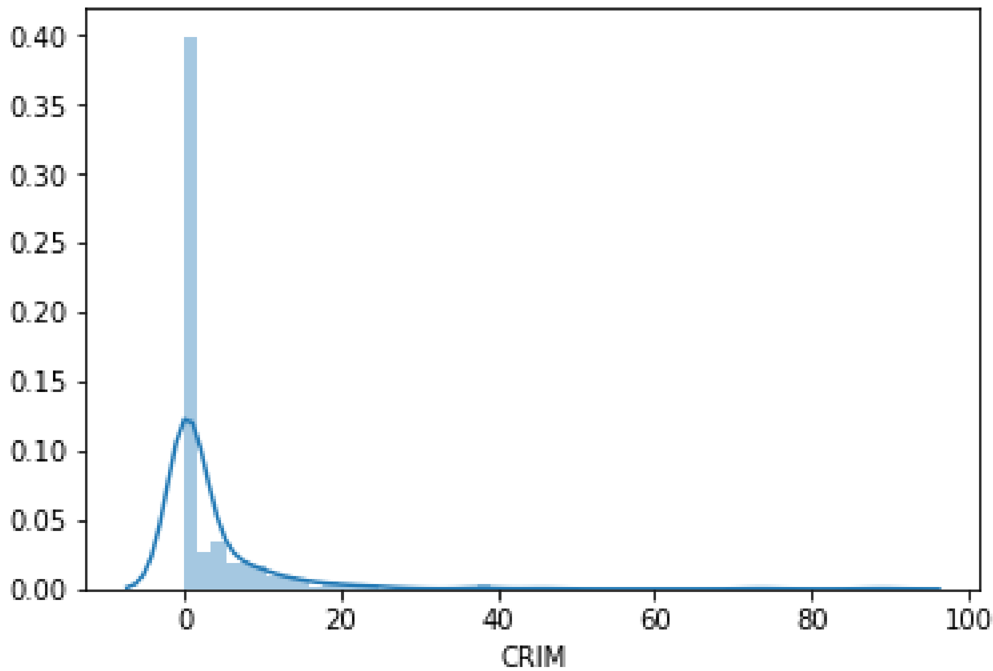

```
In [177]: sns.pairplot(train_x, vars=["DIS", "INDUS","NOX","AGE"])

Out[177]: <seaborn.axisgrid.PairGrid at 0x7f4f17bd78d0>
```




```
In [178]: sns.pairplot(train_x, vars=["TAX", "RAD"])

Out[178]: <seaborn.axisgrid.PairGrid at 0x7f4f17e83e10>
```

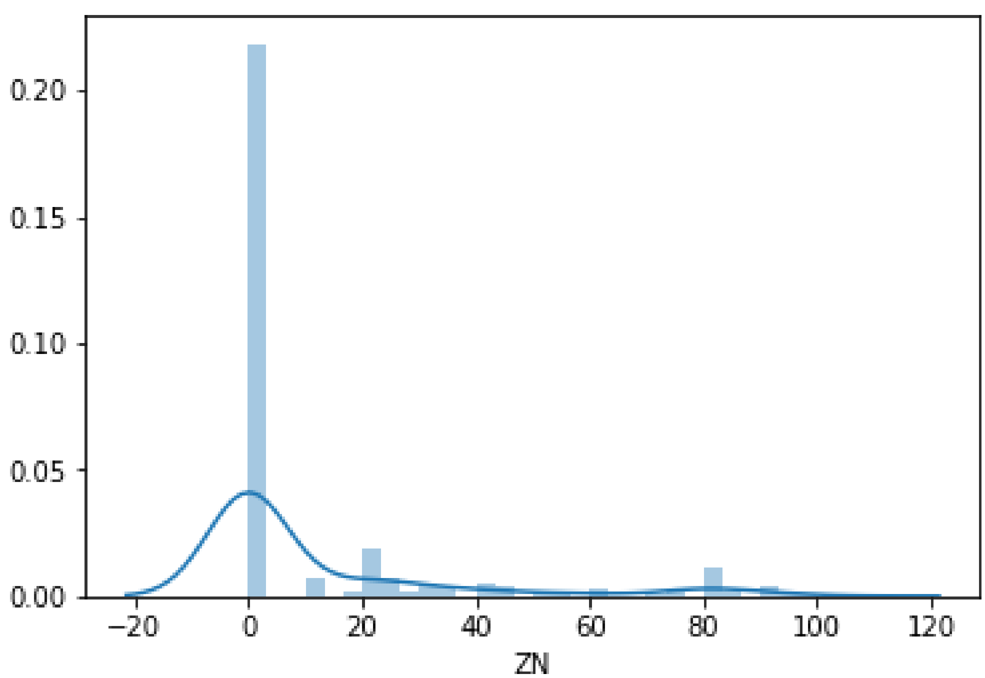



```
In [142]: #     1
          sns.distplot(train_x['CRIM'])

Out[142]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dcbe860>
```

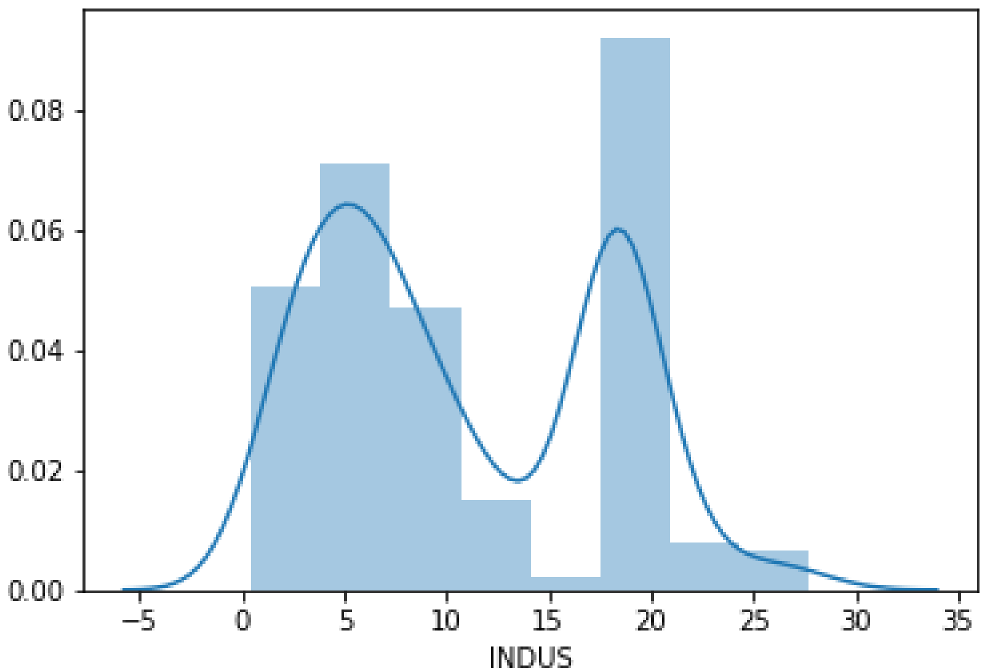



```
In [143]: # 25000
          sns.distplot(train_x['ZN'])
```

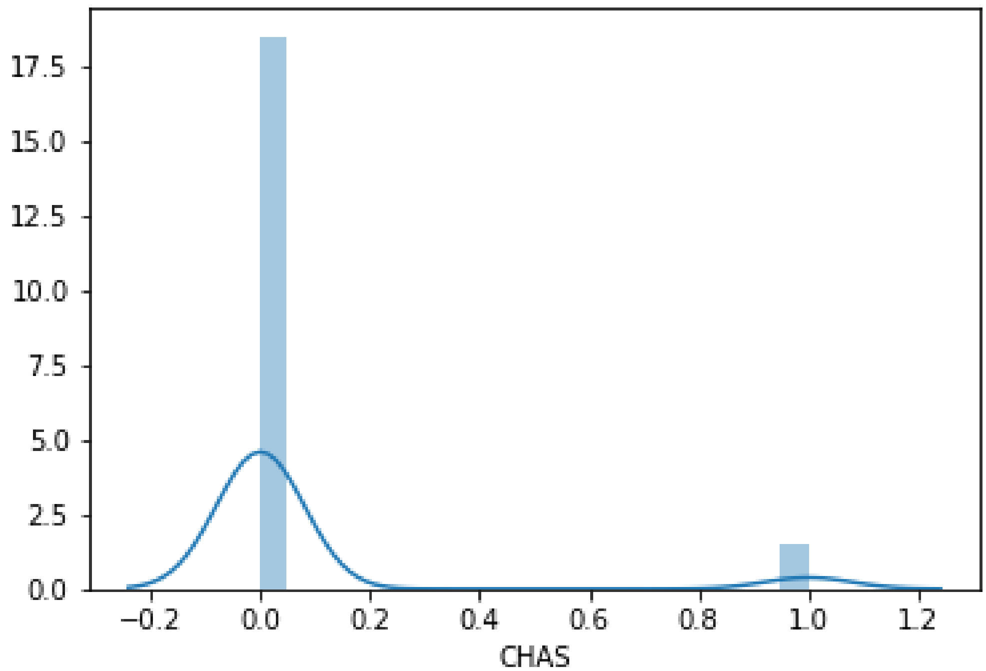
Out[143]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1e010860>



```
In [146]: #
          sns.distplot(train_x['INDUS'])

Out[146]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dcd4668>
```

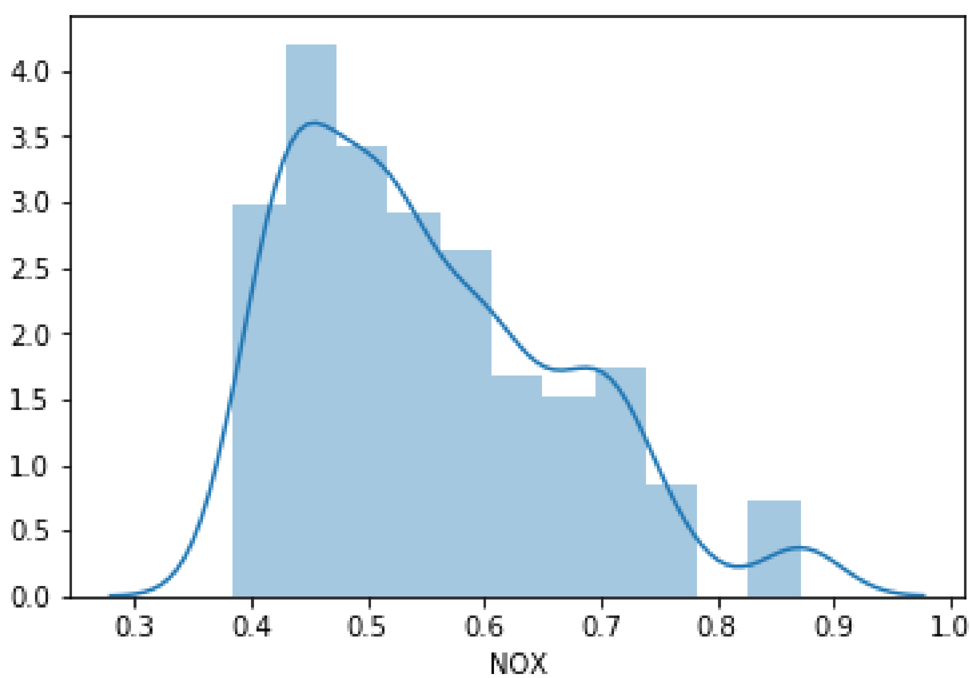



```
In [145]: #    (    1,   0)
          sns.distplot(train_x['CHAS'])

Out[145]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dd24ef0>
```



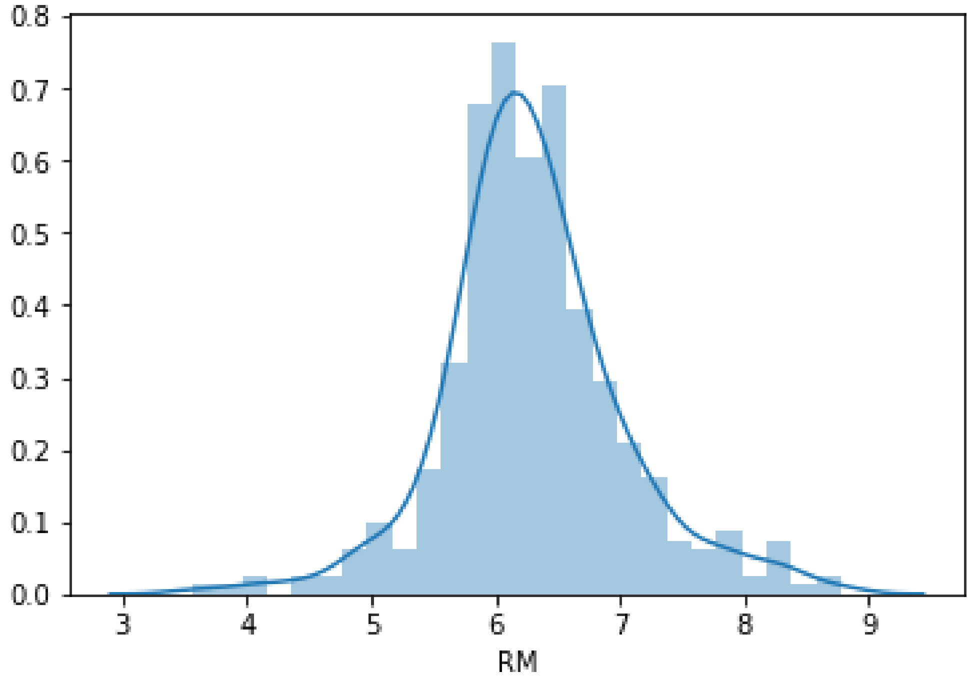

```
In [147]: # 10ppm
          sns.distplot(train_x['NOX'])
```

```
Out[147]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1e0692b0>
```



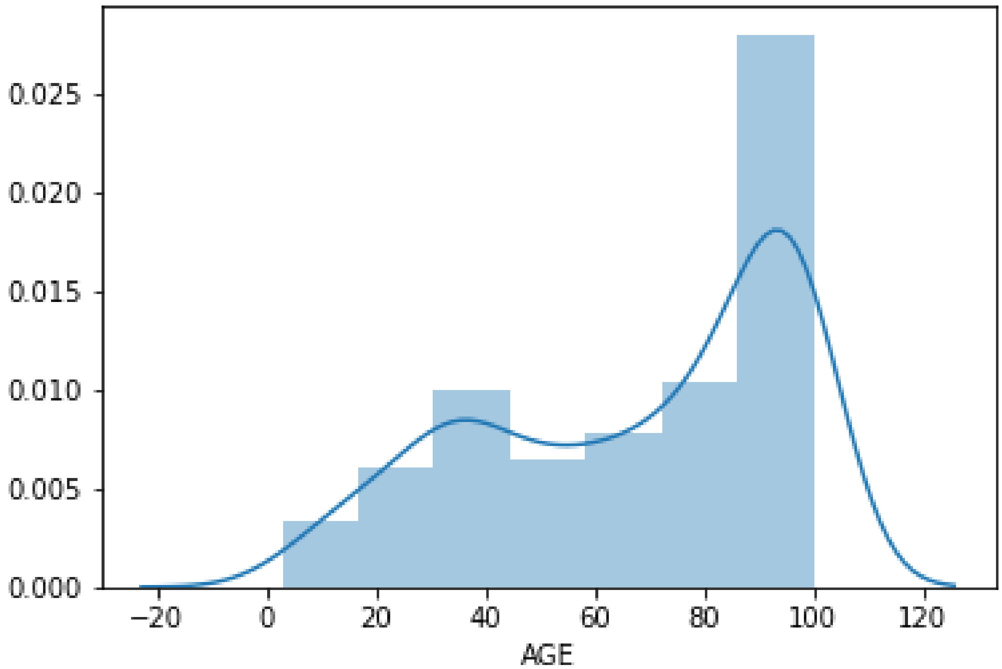
```
In [148]: #  1
          sns.distplot(train_x['RM'])
```

```
Out[148]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1de393c8>
```

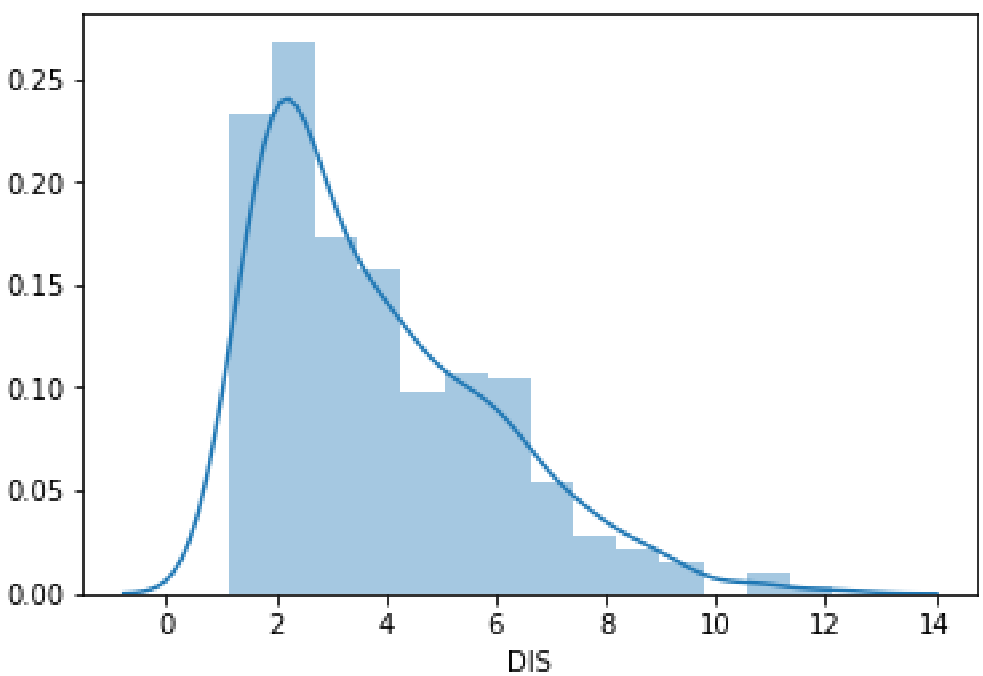



```
In [149]: # 1940
          sns.distplot(train_x['AGE'])
```

```
Out[149]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1df8ef28>
```

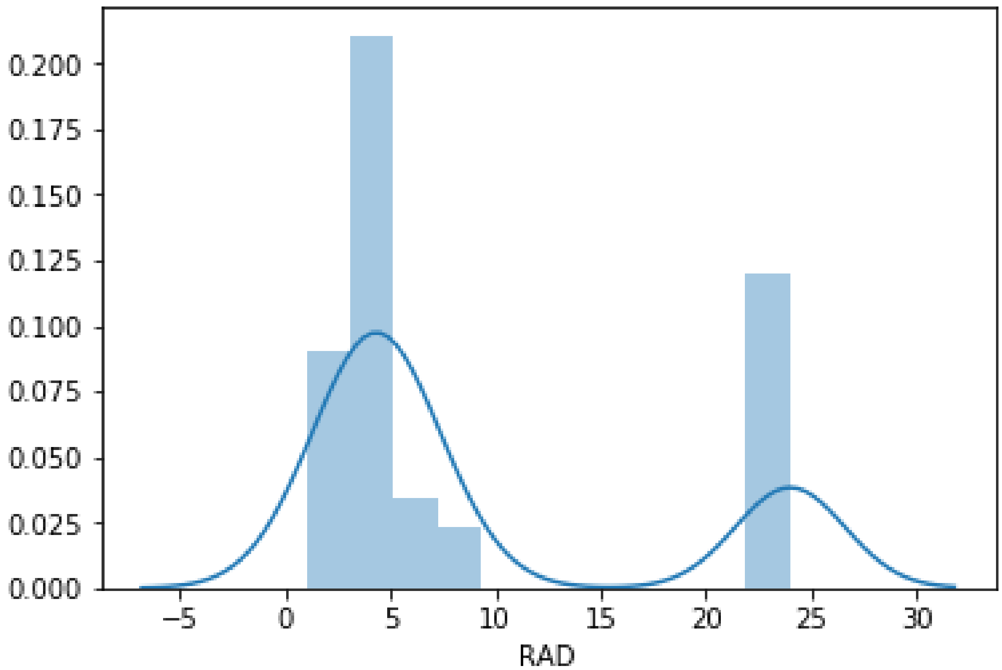



```
In [150]: # 5
          sns.distplot(train_x['DIS'])

Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1df5eac8>
```



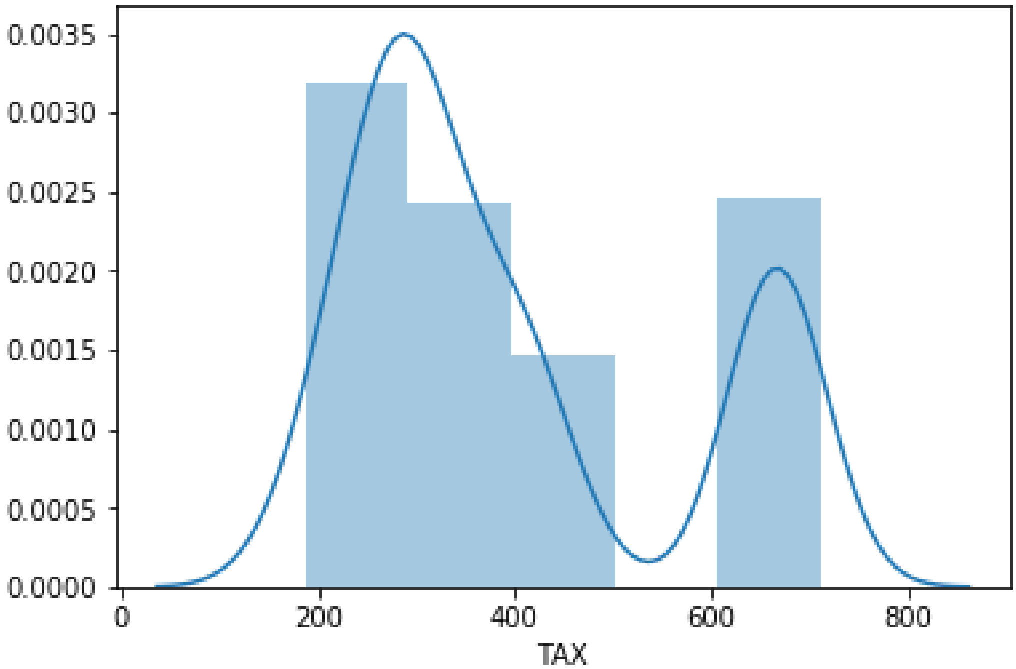
In [151]: #
```
sns.distplot(train_x['RAD'])
```

Out[151]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1e0780b8>

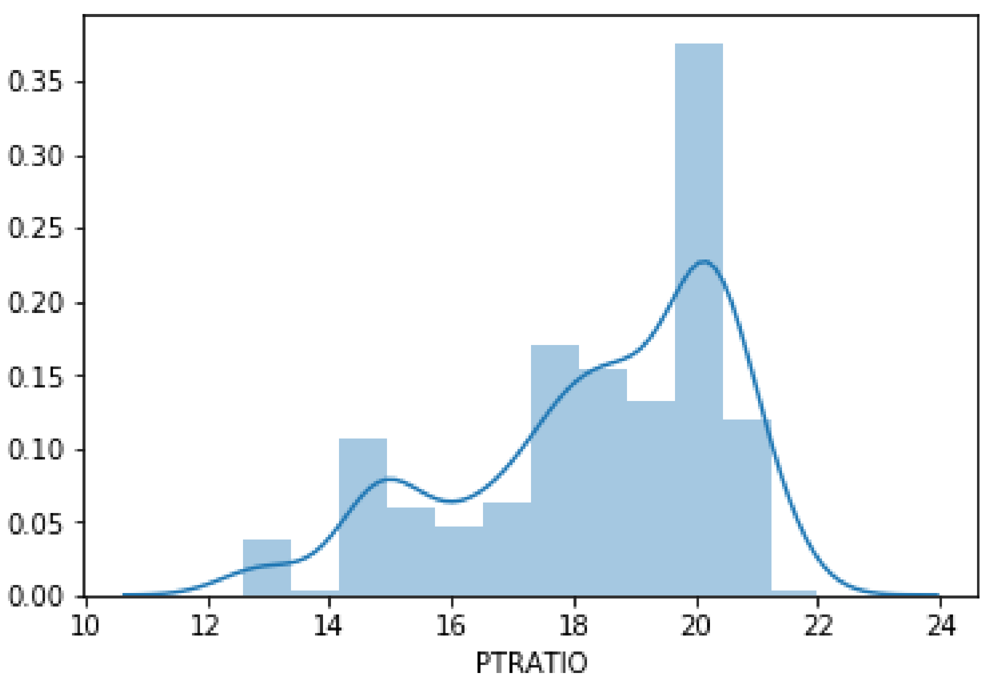


In [152]: # 10,000
```
sns.distplot(train_x['TAX'])
```

Out[152]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dbe6be0>

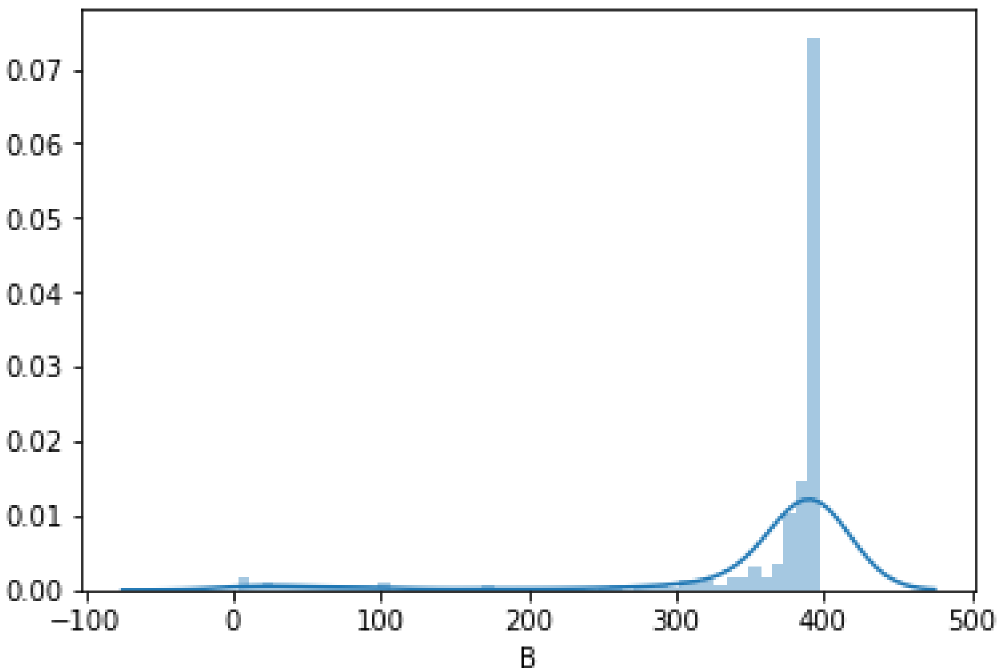



```
In [153]: #  /
          sns.distplot(train_x['PTRATIO'])

Out[153]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1db5a128>
```



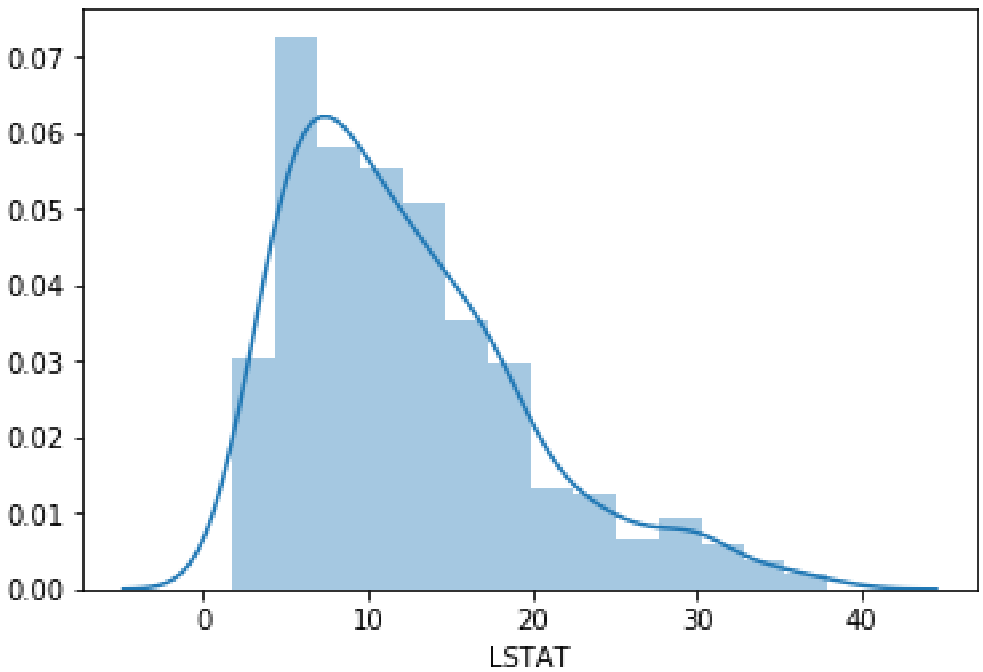
In [154]: #
```
sns.distplot(train_x['B'])
```

Out[154]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dad8fd0>



In [155]: #
```
sns.distplot(train_x['LSTAT'])
```

Out[155]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4f1dc70ef0>



4

4.1 Model1 = Lasso regression(L1 regularization)

In [164]:
```
# Lasso Regression
clf_lasso = linear_model.Lasso(alpha=1.0, fit_intercept=False, normalize=True, tol=1e
clf_lasso.fit(train_x, train_y)
```

Out[164]:
```
Lasso(alpha=1.0, copy_X=True, fit_intercept=False, max_iter=1000,
   normalize=True, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=1e-06, warm_start=False)
```

In [165]:
```
#
coef_lasso = clf_lasso.coef_
print(coef_lasso)
```

```
[-0.03670378  0.04121776 -0.          0.          0.          4.03558104
  0.03271015 -0.17000814  0.07389608 -0.00641813 -0.09949015  0.01624475
 -0.53412134]
```

4.2 Model2 = Ridge regression(L2 regularization)

In [166]:
```
# Ridge Regression
clf_ridge = linear_model.Ridge(alpha=0.0, fit_intercept=False, normalize=True, tol=1e
clf_ridge.fit(train_x, train_y)
```

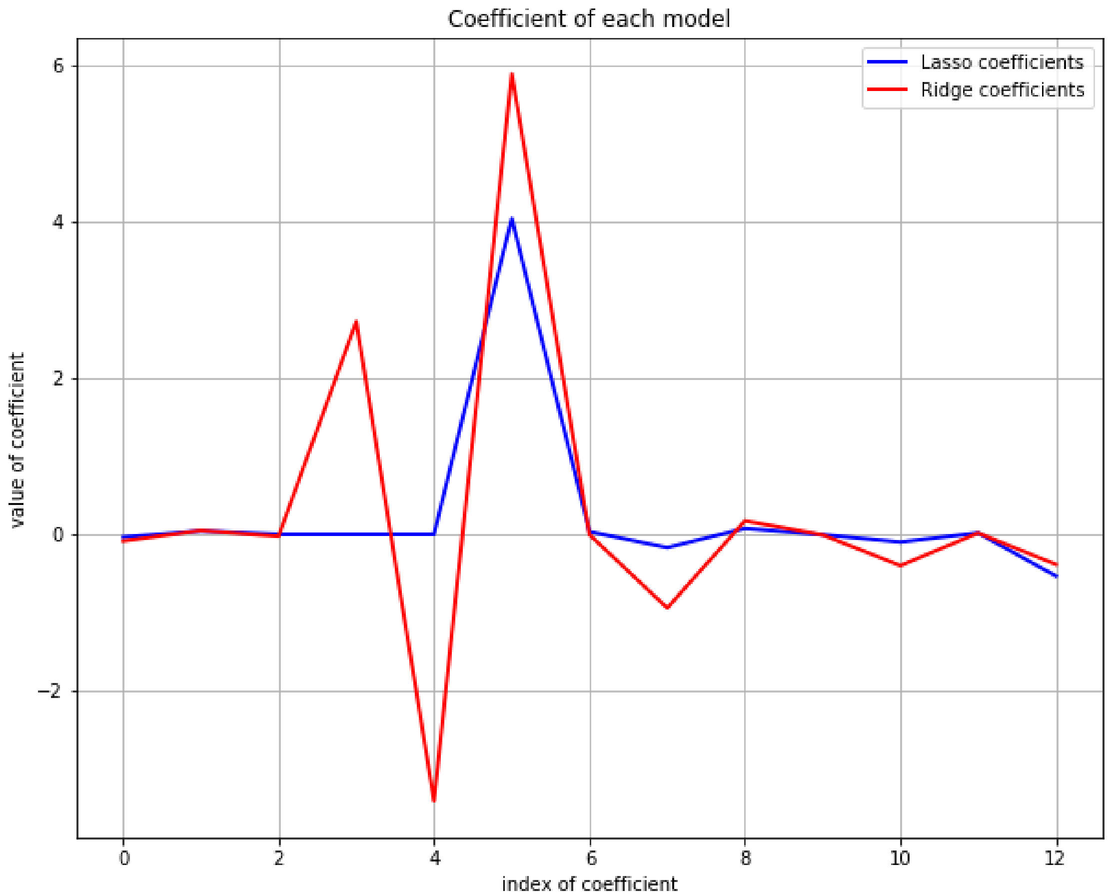
```
Out[166]: Ridge(alpha=0.0, copy_X=True, fit_intercept=False, max_iter=None,
             normalize=True, random_state=None, solver='auto', tol=1e-06)
```

```
In [167]: #
          coef_ridge = clf_ridge.coef_
          print(coef_ridge)
```

```
[-8.61158712e-02  4.64502631e-02 -2.65007636e-02  2.71886515e+00
 -3.40834404e+00  5.88156430e+00 -5.06153133e-03 -9.42950569e-01
  1.70793544e-01 -8.68774357e-03 -4.00698002e-01  1.53968223e-02
 -3.85297737e-01]
```

```
In [168]: # Lasso vs Ridge   plot
          plt.figure(figsize=(10, 8))
          plt.title("Coefficient of each model")
          plt.grid()
          plt.plot(coefficient, color='blue', linewidth=2, label='Lasso coefficients')
          plt.plot(coefficient_ridge, color='red', linewidth=2, label='Ridge coefficients')
          plt.xlabel("index of coefficient")
          plt.ylabel("value of coefficient")
          plt.legend()
          plt.show
```

```
Out[168]: <function matplotlib.pyplot.show(*args, **kw)>
```




```
In [ ]:
```

5

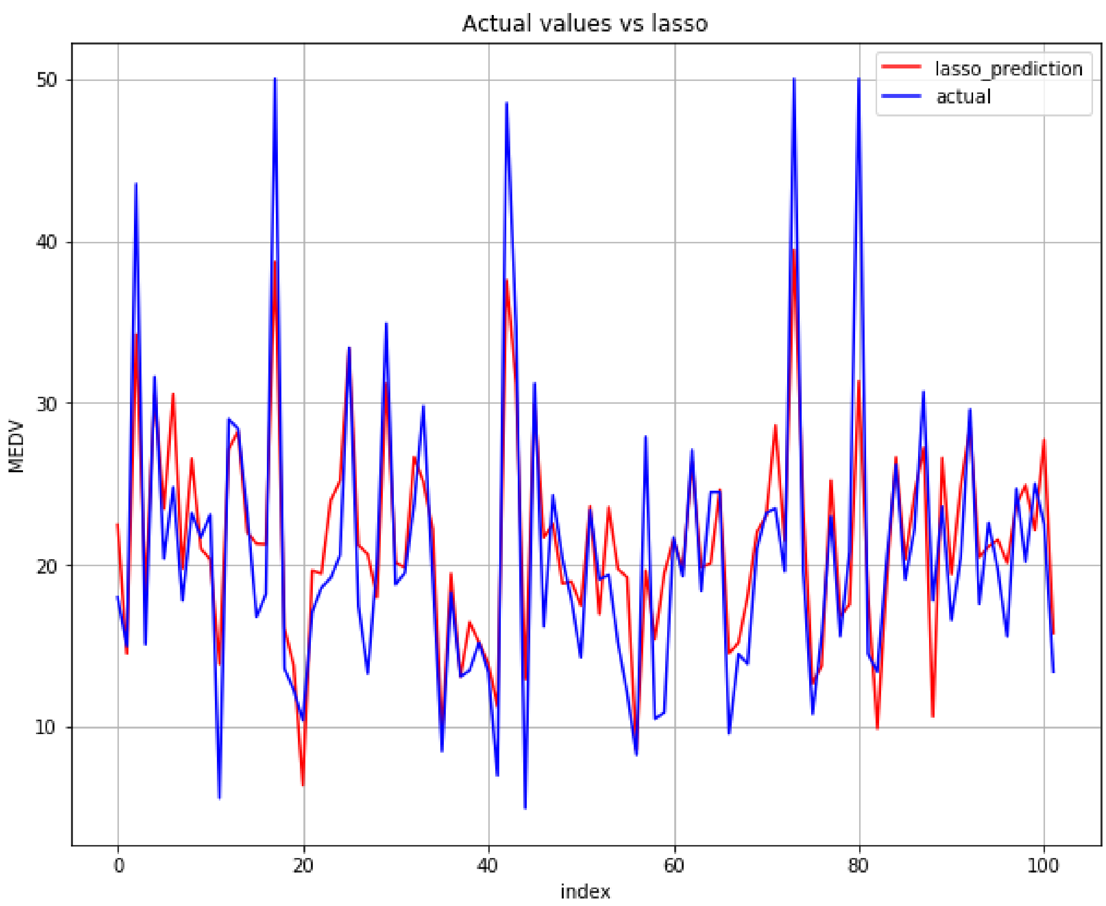
```
In [169]: pred_y_lasso = clf_lasso.predict(test_x) # lasso
          pred_y_ridge = clf_ridge.predict(test_x) # ridge
```

```
In [170]: # plot
          pred_y_col_lasso = pd.Series(pred_y_lasso) # lasso  Series
          pred_y_col_ridge = pd.Series(pred_y_ridge) # ridge  Series

          test_y_col = pd.Series(test_y).reset_index(drop=True) #
```

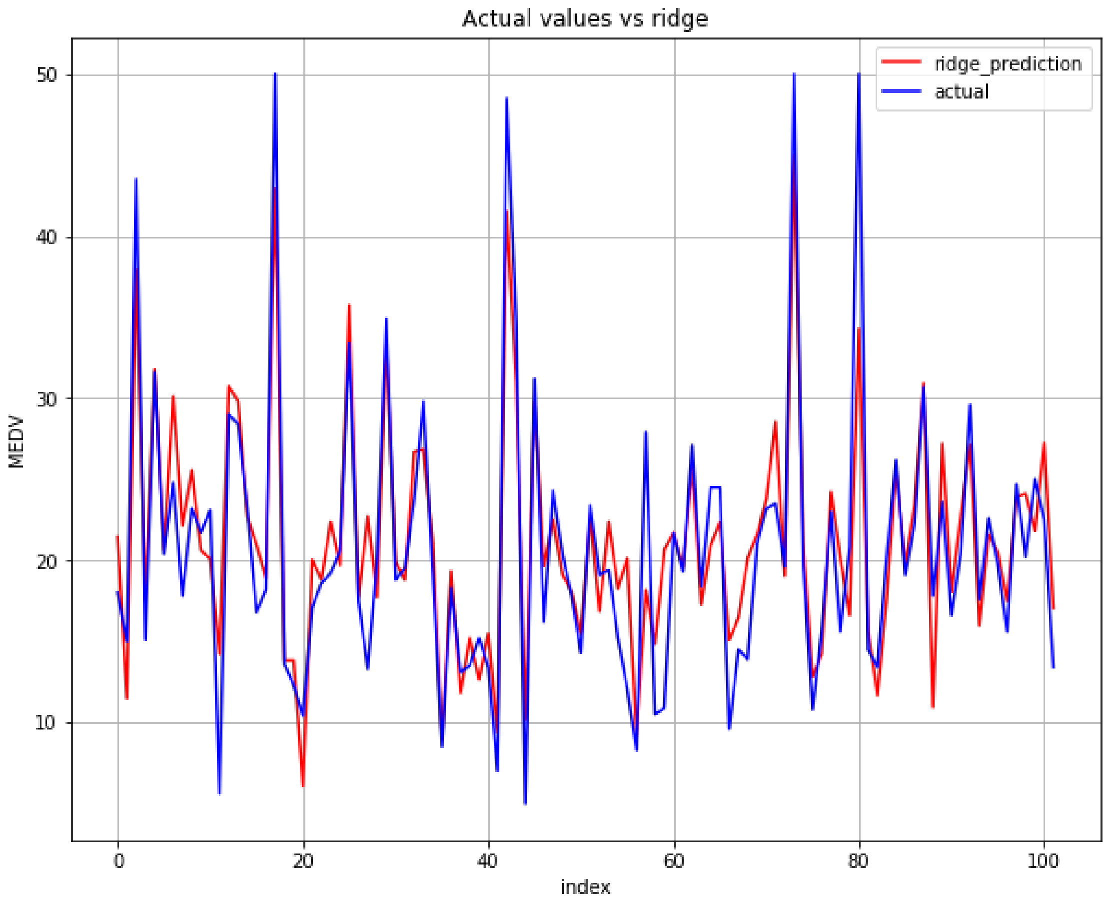
```
In [171]: compare_y = pd.concat([ pred_y_col_lasso, pred_y_col_ridge, test_y_col] , axis=1) #
          compare_y.columns = ["predict_lasso", "predict_ridge","real_MEDV"] # column
          compare_y['index'] = compare_y.index #
```

```
In [172]: #  () vs Lasso () plot
          plt.figure(figsize=(10, 8))
```

```
plt.title("Actual values vs lasso")
plt.grid()
plt.plot(compare_y['index'], compare_y['predict_lasso'], 'r-', label='lasso_predictio
plt.plot(compare_y['index'], compare_y['real_MEDV'], 'b-', label='actual')
plt.xlabel("index")
plt.ylabel("MEDV")
plt.legend()
plt.show()
```




```
In [173]: #  () vs Ridge () plot
          plt.figure(figsize=(10, 8))
          plt.title("Actual values vs ridge")
          plt.grid()
          plt.plot(compare_y['index'], compare_y['predict_ridge'], 'r-', label='ridge_predicti
          plt.plot(compare_y['index'], compare_y['real_MEDV'], 'b-', label='actual')
          plt.xlabel("index")
          plt.ylabel("MEDV")
          plt.legend()
          plt.show()
```




```
In [174]: # Lasso R^2 value
          clf_lasso.score(test_x, test_y)
```

```
Out[174]: 0.7486259621705256
```

```
In [175]: # Ridge R^2 value
          clf_ridge.score(test_x, test_y)
```

```
Out[175]: 0.8145234131230182
```

```
In [ ]:
```