I Linear Regression

1. Simple Linear Regression

a) Divide the dataset into training set and test set; use **sklearn.linear_model. LinearRegression()** to implement a simple linear regression and determine the relation of predictive variables and responding variable.

   Before the division procedure, we have to be familiar with the dataset

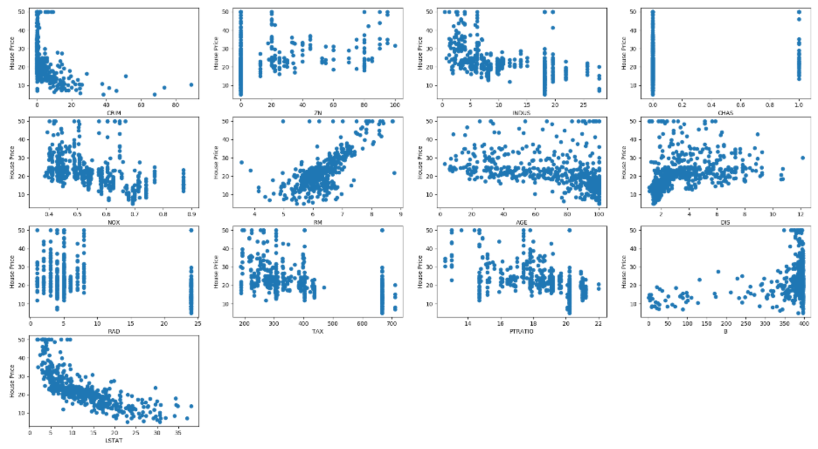


Fig.1 Scatter plot of 13 predictive variables to 1 responding variable

   From figure.1 we can find that some predictive variables obviously have linear relation with the responding variable while others do not. Meanwhile, we find some abnormal data in the dataset, so we eliminate these abnormal data with **np.append().**

   Afterwards, we use **train_test_split()** to divide the dataset and leave 20% as test set. Then we use **LinearRegression,fit()** and **LinearRegression,predict()** to train our model and test the trained linear regression model.

b) Plot the relation between responding variable and predictive variables, draw the least square regression line.

   From the training results in a) we can use **matplotlib.pyplot.plot()** to do this task and the result is shown as below.

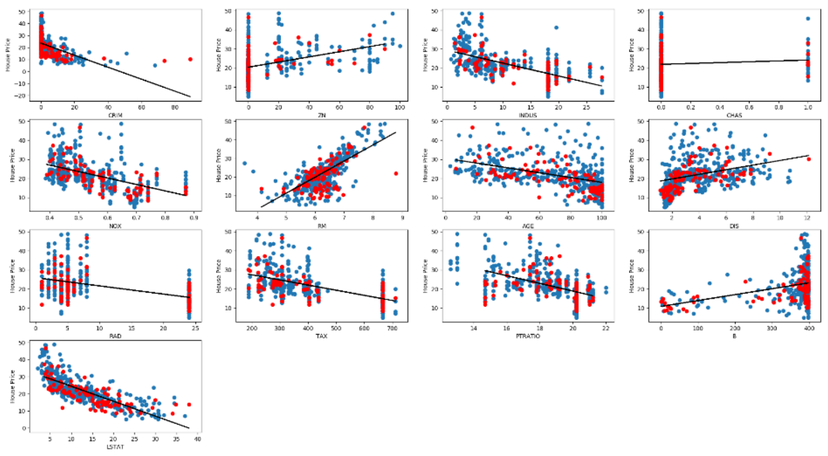
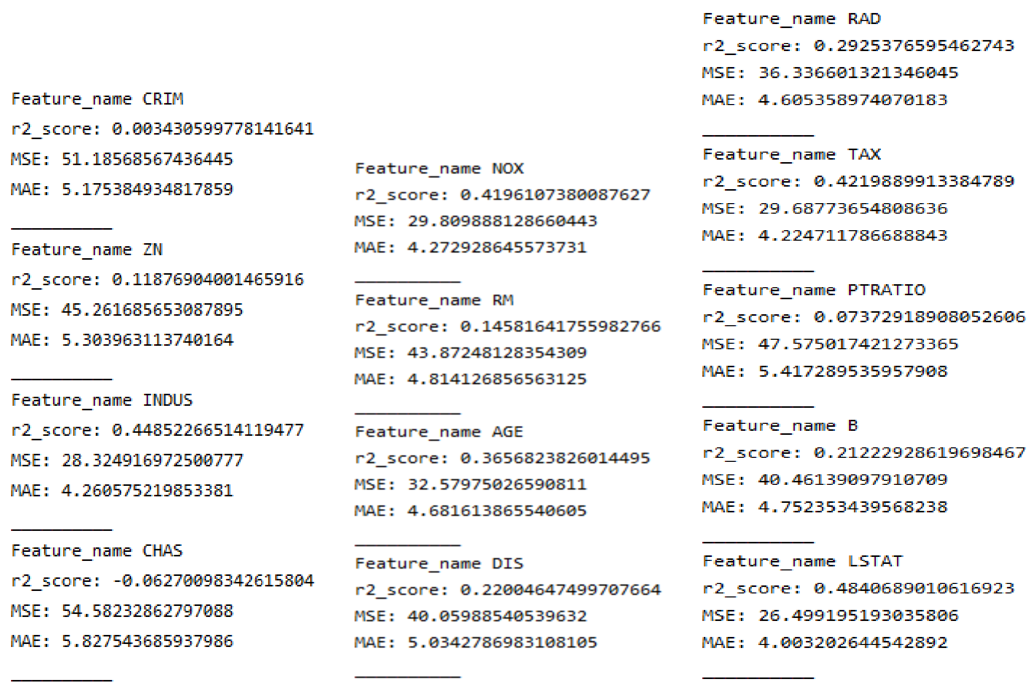


Fig.2 The least square lines and scatter of datasets

In above figure, red points indicate test set data and blue points indicate training data, red lines are least square regression lines. It is easy to see that features like **NOX, RM, AGE PTRATIO, LSTAT** show better linearity with **House Price** than other features. This fact will help us modify our model and gain better performance.

c) Use the evaluation module in **LinearRegression()** and output the results.



```
Feature_name CRIM
r2_score: 0.003430599778141641
MSE: 51.18568567436445
MAE: 5.175384934817859
__________
Feature_name ZN
r2_score: 0.11876904001465916
MSE: 45.261685653087895
MAE: 5.303963113740164
__________
Feature_name INDUS
r2_score: 0.44852266514119477
MSE: 28.324916972500777
MAE: 4.260575219853381
__________
Feature_name CHAS
r2_score: -0.06270098342615804
MSE: 54.58232862797088
MAE: 5.827543685937986
__________
```

```
Feature_name NOX
r2_score: 0.4196107380087627
MSE: 29.809888128660443
MAE: 4.272928645573731
__________
Feature_name RM
r2_score: 0.14581641755982766
MSE: 43.87248128354309
MAE: 4.814126856563125
__________
Feature_name AGE
r2_score: 0.3656823826014495
MSE: 32.57975026590811
MAE: 4.681613865540605
__________
Feature_name DIS
r2_score: 0.22004647499707664
MSE: 40.05988540539632
MAE: 5.0342786983108105
__________
```

```
Feature_name RAD
r2_score: 0.2925376595462743
MSE: 36.336601321346045
MAE: 4.605358974070183
__________
Feature_name TAX
r2_score: 0.4219889913384789
MSE: 29.68773654808636
MAE: 4.224711786688843
__________
Feature_name PTRATIO
r2_score: 0.07372918908052606
MSE: 47.575017421273365
MAE: 5.417289535957908
__________
Feature_name B
r2_score: 0.21222928619698467
MSE: 40.46139097910709
MAE: 4.752353439568238
__________
Feature_name LSTAT
r2_score: 0.4840689010616923
MSE: 26.499195193035806
MAE: 4.003202644542892
__________
```

Fig.3 Evaluation of 13 simple linear regression models

From above indicators (R2, MSE, MAE) we can intuitively judge the regression models.

d) Train linear regression models **LinearRegression()** and **SGDRegressor()** on bonston dataset and give the evaluation results.
Before training we use **StandarScaler()** to standardize the training data so as to speed up the training procedure. Also we use **r2_score**(), **mean_squared_error()** and **mean_absolute_error()** to evaluate the model.

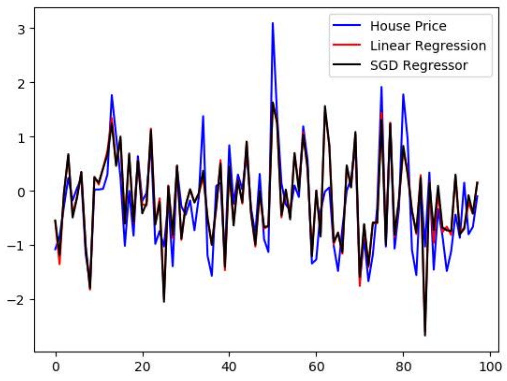
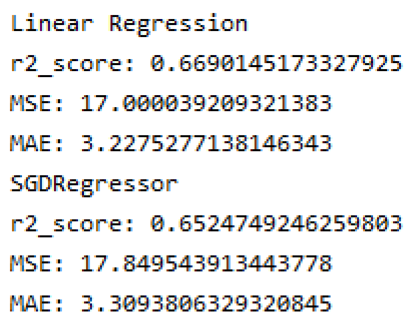


Fig.4 Comparison of Linear regression and SGD Regressor

It can be seen from Fig.4 that linear regression outperform SGD regressor but only a little bit, this may result from the preprocessing of **StandarScaler().**



```
Linear Regression
r2_score: 0.6690145173327925
MSE: 17.000039209321383
MAE: 3.2275277138146343
SGDRegressor
r2_score: 0.6524749246259803
MSE: 17.849543913443778
MAE: 3.3093806329320845
```

Fig.5 Evaluation of Linear Regression and SGD Regressor

From Fig.5 we can reassure the statement above that there is little difference between the performance of these two model. And I do think it is because I standardized the training data before training my model.

2. Multiple Linear Regression

a) Draw the scatter plot of all variables in the dataset.

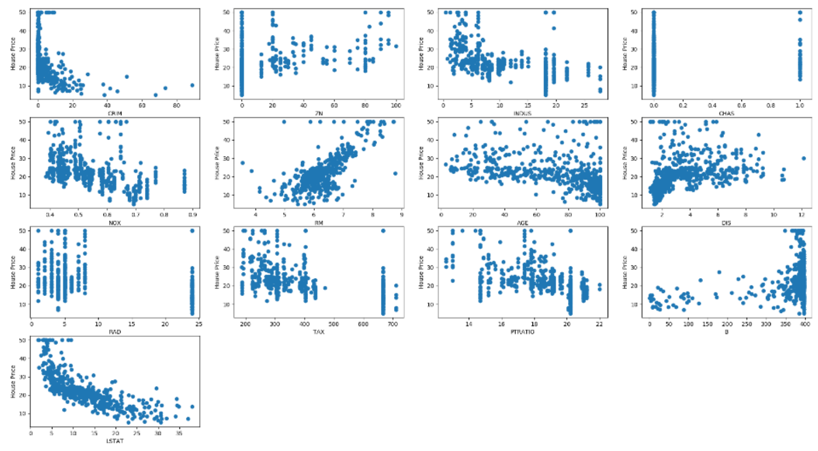


Fig.6 Scatter plot of all variables in the dataset

This has been done in 1. Simple Linear Regression, but here we do it again.

b) Compute the Correlation matrix of all variables.

Here we use **numpy.corrcoef()** to compute the correlation matrix and **matshow()** to visualize this matrix.

Due to the large dimension of this correlation matrix, we do not show it here and we just show the mat figure of it.

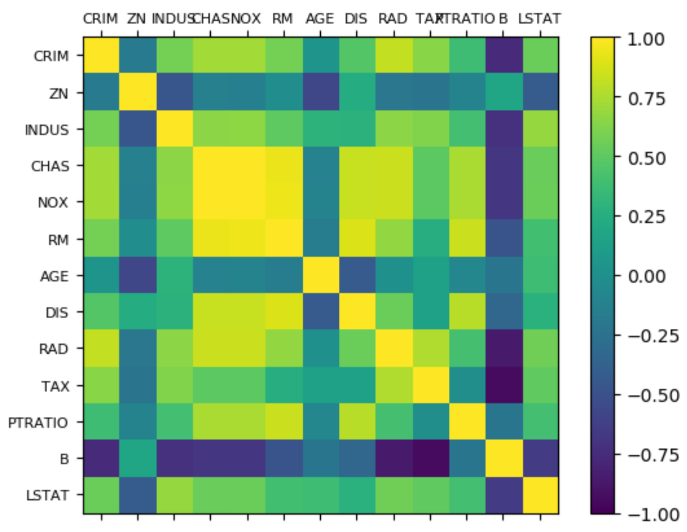
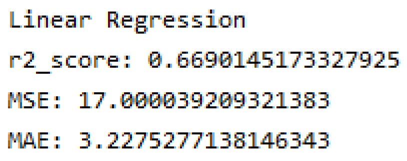


Fig.7 Mat plot of correlation matrix

From Fig.7 we can easily find out which variables have strong relation and which do not. This information will also help us choose precise features and build a better model.

c) Use **sklearn.preprocessing.PolynomialFeatures(degree=2).fit_transform(boston.data)** to implement multiple linear regression and give its evaluation.



```
Linear Regression
r2_score: 0.6690145173327925
MSE: 17.000039209321383
MAE: 3.2275277138146343
```

Fig.8 Evaluation of Multiple Linear Regression

By contrasting the above evaluation indicators with those of simple linear regression we can be sure that multiple linear regression shows better performance than simple linear regression.

d) Run Cross-Validation.

Following the instruction in the given website, we can get the following figure

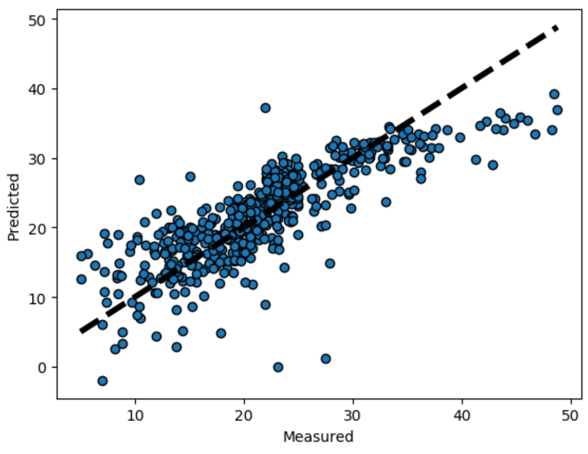


Fig.9 Cross-Validation

Hence we can analyze the prediction errors from the abve figure. In fact, this multiple linear regression does show better performance than simple linear regression.

e) How will the evaluation scores change with the increasing of training data size? Draw a curve to visualize the changes.

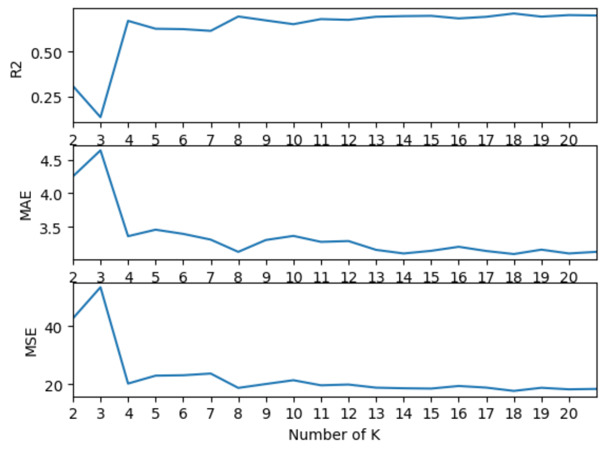


Fig.10 The change of R2, MAE and MSE

Still, we choose R2, MAE and MSE as our evaluation indicators. As the training dataset increases, those these indicators rise clearly. But as we know, if we divide the training set much larger than the test set, the **generalization ability will degenerate**. So every time we train a model, we need to think carefully about the size of training set.

3. Ridge Regression and Lasso Regression

a) Use **sklearn.linear_model.Ridge()** and **sklearn.linear_model.Lasso()** to realize Ridge regression and Lasso regression. Analyze the relation between different input variables and the output variable.
Ridge regression adds a punishment term of l2-norm while Lasso regression adds a term of l1-norm.

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2,$$

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|$$

Both terms are used to diminish the number of features.

b) Run test on the test sets and compare them with multiple linear regression.multiple linear regression.
Here we just implement Ridge regression and Lasso regression as instructed.

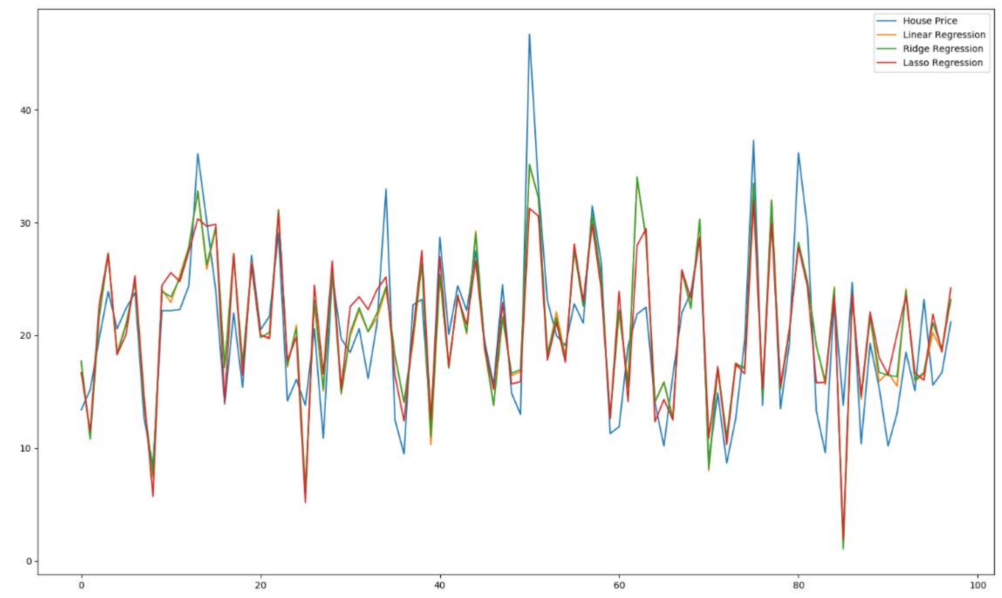
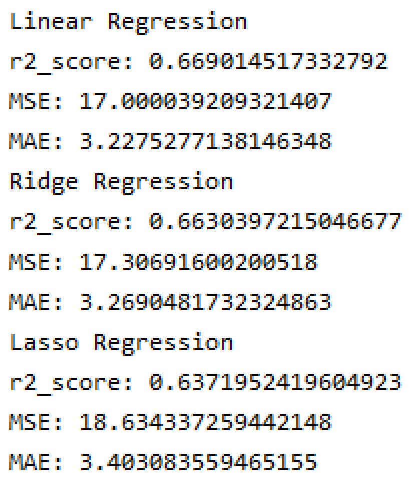


Fig.11 Comparison between three different regression models



```
Linear Regression
r2_score: 0.669014517332792
MSE: 17.000039209321407
MAE: 3.2275277138146348
Ridge Regression
r2_score: 0.6630397215046677
MSE: 17.30691600200518
MAE: 3.2690481732324863
Lasso Regression
r2_score: 0.6371952419604923
MSE: 18.634337259442148
MAE: 3.403083559465155
```

Fig.12 Evaluation of three different regression models

From Fig.11 and Fig.12 we can find that multiple linear regression outperfom Ridge regression and Lasso regression a little bit.

c) Change α in Ridge regression and Lasso rgression and draw the change curve of regression coefficients. Predict the impact of the changes.
Following the instruction in the given website, we can get the change curve as follow.

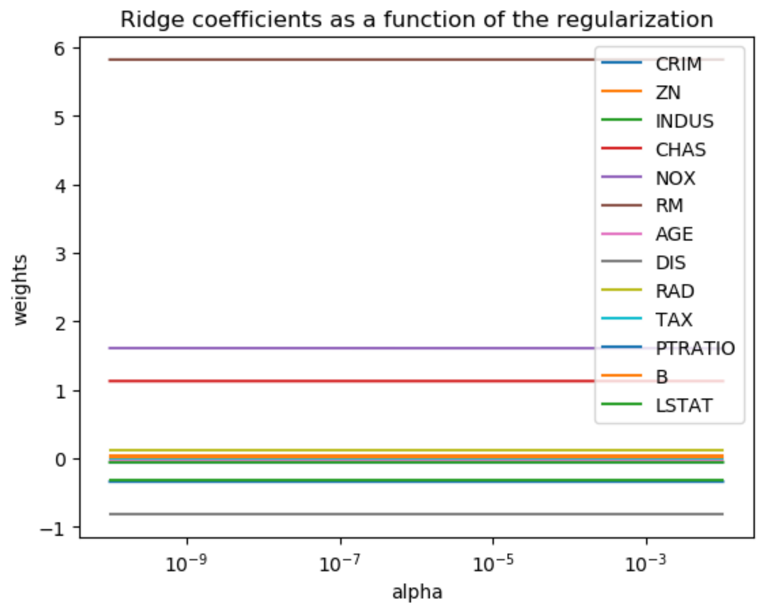


Fig.13 Ridge coefficients (max_iteration = 4000)

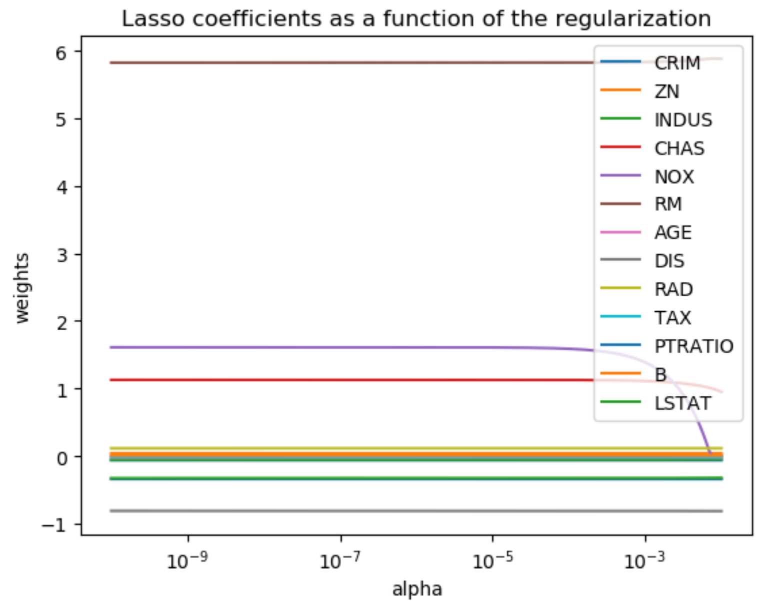


Fig.14 Lasso coefficients (max_iteration = 4000)

At first Fig.14 and Fig.15 did not show as I expected and an error of convergence happened. So I set the max_iteration larger to 4000, then the error disappeared.
From the above figures we can find that Ridge regression has almost no impact on the features while Lasso shows impact on 2 features.

Conclusion

From above simple linear regression to multiple linear regression and Ridge regression and Lasso regression, we can find that in this Boston House Price example, the 13 features have some tight interrelations, that is to say, these 13 features are not independent. That is why the linear regression cannot predict perfectly on this dataset. Therefore in future projects, before choosing a model and train it without any thoughts, we have to dig into

the data and reveal the deeper relations of features. Only by this can we gain deep recognition of the data and models.
Meanwhile, my inexpertness in Python made some trouble in this experiment and decreased my efficiency. Even though I know the basic theorem of these models, I cannot quickly accomplish the tasks. Therefore, I have to gain proficency in Python by more coding training.