EE219 PROJECT 1
Regression Analysis

Guanchu Ling 904590047
Yingchao Tang
Yang Guo

Introduction

In this project, we have done data analysis on two data sets - Network Backup dataset and Boston Housing dataset. The datasets are analyzed by using the data mining approaches of regression. Regression analysis is a statistical process for estimating the relationships among variables. We have studied the concepts and methods of cross-validation, regularization, random forest and neural network regression in order to predict a dependent variable present in the datasets.

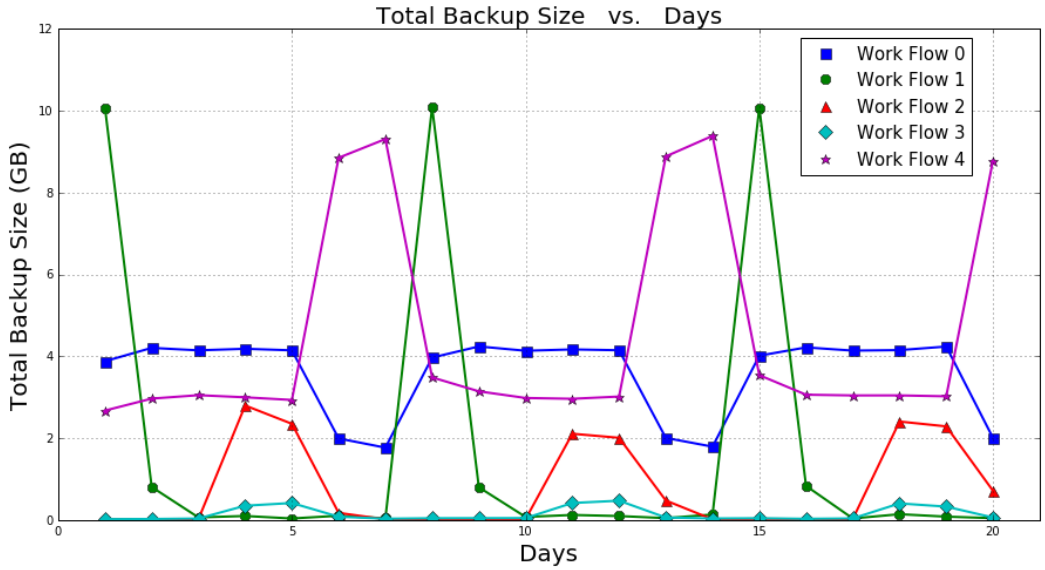
The "network_backup_dataset" has captured simulated traffic data on a backup system and contains information of the size of the data moved to the destination as well as the time it took for backup. Our mission was to predict the backup size of the traffic depending on the file-name, day/time of backup. Prediction models have been using Linear, Random Forest, Neural Network and Polynomial Regression.

The "housing dataset" that contained the housing values of suburbs. We have employed Linear and Polynomial Regression along with other algorithms to overcome overfitting to create a predictive model that was used to estimate the value of owner-occupied homes.

Network backup dataset analysis

1. Repeating pattern

To get a whole sense of the patterns existing in this dataset, we tried to split the pattern into five parts, according to different work flows that they belong to. We plotted the actual copy sizes of all the files on the time period of 20 days to see if any repeating patterns exist. The plot is shown below:



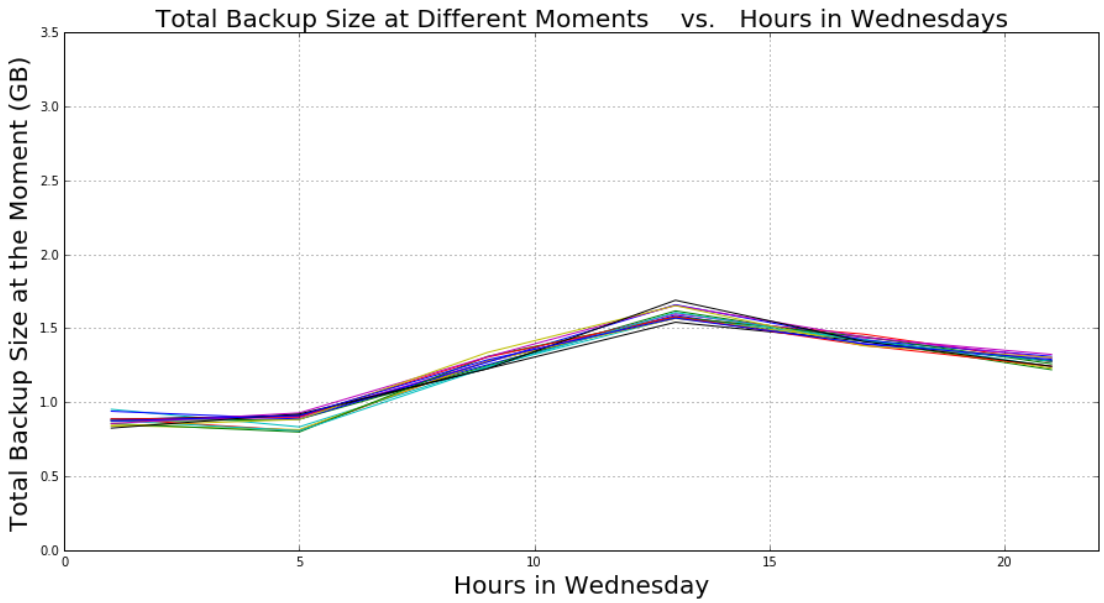
Total backup size in time variation

As can be clearly seen from the plot above, the total backup size of each work flow has an approximately periodic relation. The same pattern appears every 7 days, with very slight variations. We can make speculation that "week" attribute is not important, because the total backup size is strongly related to the "day of week" attribute. If we are given a specific day of week, we can make very good approximation on the total backup data size of this day.

If we look at the five different work flows respectively, we can get more information from this plot:

- For Work Flow 0, it periodically backs up more data on workdays (Mondays to Fridays), and less data on weekends (Saturdays and Sundays). This could possibly be a daily routine for the network system that typically has higher data change rate during workdays.
- For Work Flow 1, it backs up a large amount of data only on every Mondays, but far less data backup on other days in a week. This work flow could possibly be a weekly backup.
- For Work Flow 2, Work Flow 3 and Work Flow 4, the most data backup happens on Thursdays and Fridays. On other days, only a little data is backed up. These three work flows have similar patterns of backup size, but the peak backup size is variant.

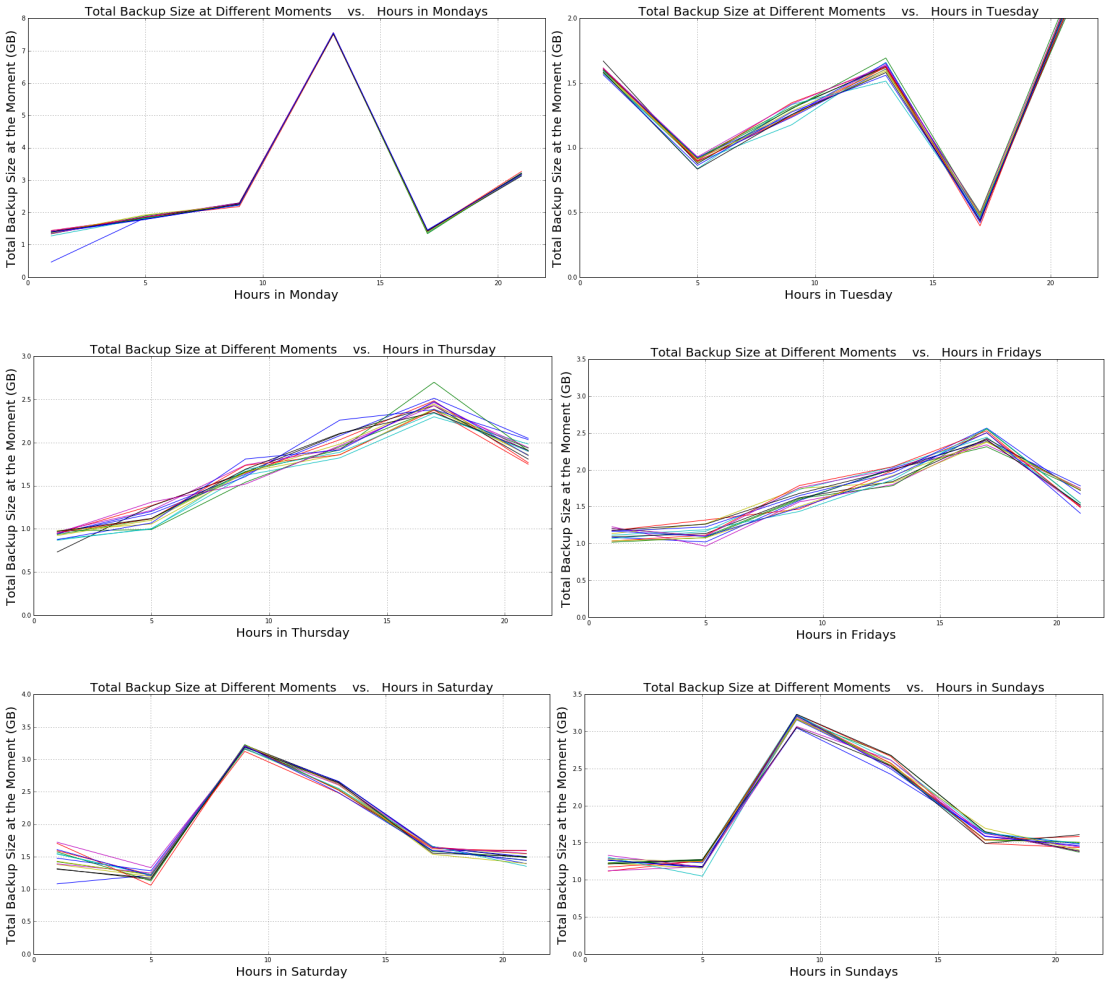
In order to verify the pattern we have discovered so far, we tried to plot the backup data size of every four hours cycle on fixed 'day of week' (i.e. fixed on Mondays, fixed on Tuesdays, etc.). One of the plots are shown below:



Total backup size at different moments for all Wednesdays

In the plot shown above, every line in a different color represents a Wednesday of a different week. For example, Wednesday of the 1st week, Wednesday of the 2nd week, etc. As we know that the backup occurs every four hours, so there are totally 6 backups every day at 1:00, 5:00, 9:00, 13:00, 17:00 and 21:00, respectively. By connecting these six dots for every day, the repeating pattern can be clearly recognized: all lines tend to overlap together, which means that all Wednesdays have the same backup pattern and that the variation of week number does not have too much influence on the pattern. This verifies the speculation from the previous plot that 'week' attribute is not important.

More plots on other days in a week are also shown below, and we can draw the same conclusion which again verifies our speculation.



Total backup size at different moments

2a. Linear Regression

The original dataset contains 7 attributes:

- Week #: This attribute records in which week the backup occurs;
- Day of Week: This attribute records on which day of that week the backup occurs;
- Backup Start Time: This attribute records the exact moment when a backup occurs;
- Work-flow-ID: This attribute records to which work flow the backup belongs;
- File Name: This attribute records the name of the file being backed up;
- Size of Backup: This attribute records the size of backup data;
- Backup Time: This attribute records the duration that the backup spends.

Among these 7 attributes, some are numerical-valued while others are string-valued. In order for the regression algorithm to calculate, all string values must be converted to numerical values.

A straightforward approach is to assign every string-valued item a corresponding numerical-valued number. However, by doing this, these attributes will become ordinal values rather than categorical values, and this is not correct. For example, if we just simply convert the "week" attribute like below:

Before Conversion	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
After Conversion	1	2	3	4	5	6	7

Then it does not make sense that "Sunday" (converted to 7) is greater than "Monday" (converted to 1).

In fact, what we want to do is denote every value differently while keep their values treated equally. Therefore, One-Hot Encoding is utilized for this purpose. For every categorical attribute, all possible values will first be gathered by iterating the whole dataset; then for every possible value of this attribute, a new Boolean attribute will be added to the dataset to denote whether or not the item belongs to this category. For example, the attribute "Backup Occurs on Monday" will be added to the dataset. If a backup occurs on Monday, then the value under this attribute will be 1; otherwise it will be 0.

In this sense, for all 7 possible values in "Day of Week", there will be totally 7 new attributes added to the dataset. Similarly, there are 6 possible values in "Backup Start Time", 5 possible values in "Work-Flow-ID" and 30 possible values in "File name". These attributes will also be converted by One-Hot Encoding. A screenshot of part of the dataset after One-Hot Encoding is shown below:

Wednesday	Thursday	Friday	Saturday	Sunday	start_time_1	start_time_5	start_time_9	start_time_1	start_time_1	start_time_2	work_flow_0	work_flow_1	work_flow_2	work_flow_3	work_flow_4	File_19	File_18
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0

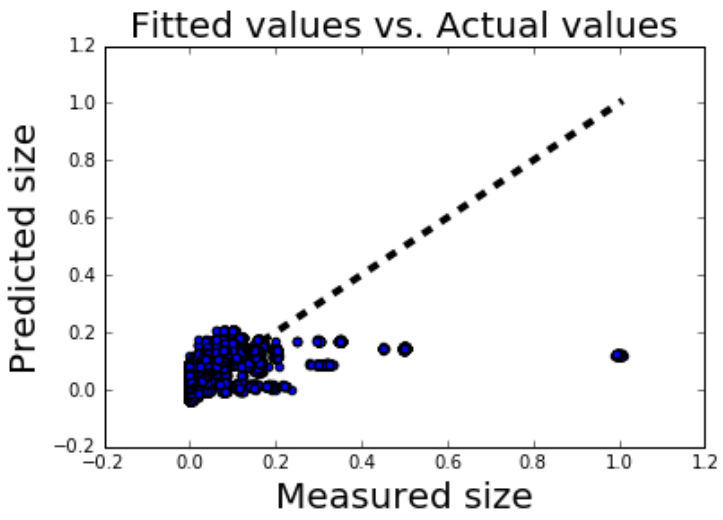
Part of the One-Hot encoded dataset

Now the dataset is ready for linear regression. Because the attributes have been converted into One-Hot Encoding, the original attributes are no longer needed and are dropped from the dataset. Additionally, the attribute “Backup Time” could also been dropped because it is a “consequence” attribute which cannot be used for determine the size of backup.

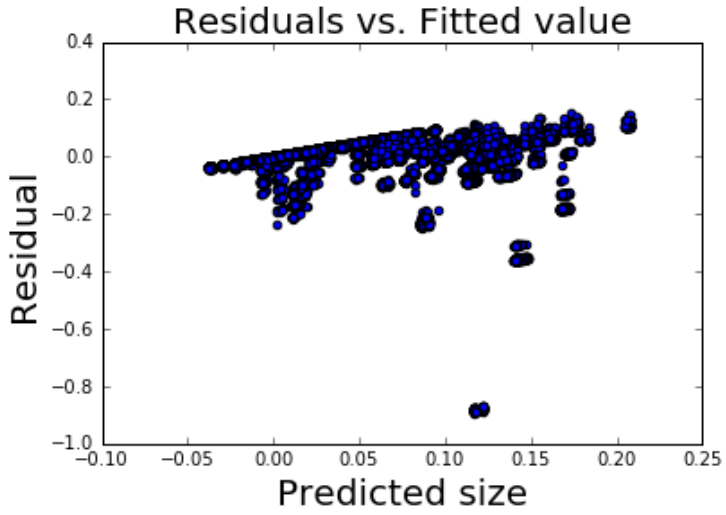
The linear model is then fitted based on the training data (the One-Hot encoded dataset) and the target data (the attribute “Backup Size”). After all the coefficients are calculated, the model is then tested in a 10-fold cross validation and the RMSE is calculated. The coefficients of attributes are listed below:

1.02827829e-05	7.39038123e+09	7.39038123e+09	7.39038123e+09
7.39038123e+09	7.39038123e+09	7.39038123e+09	7.39038123e+09
-5.27680545e+09	-5.27680545e+09	-5.27680545e+09	-5.27680545e+09
-5.27680545e+09	-5.27680545e+09	-1.23453802e+09	-9.92849985e+08
-1.66813454e+09	-6.80892140e+08	-2.08031685e+09	-1.43268364e+09
-1.43268364e+09	-1.12072580e+09	-1.12072580e+09	-4.45441239e+08
-4.45441239e+08	-4.45441239e+08	-4.45441239e+08	-4.45441239e+08
-4.45441239e+08	-3.32589303e+07	-3.32589303e+07	-3.32589303e+07
-3.32589303e+07	-3.32589303e+07	-3.32589303e+07	-1.43268364e+09
-1.43268364e+09	-1.43268364e+09	-1.43268364e+09	-1.12072580e+09
-1.12072580e+09	-8.79037761e+08	-8.79037761e+08	-1.12072580e+09
-1.12072580e+09	-8.79037761e+08	-8.79037761e+08	-8.79037761e+08
-8.79037761e+08			

After 10-fold cross validation, all predicted target data was obtained. The scattered plot of predicted values and actual values is shown below:



Scattered plot of predicted values vs. actual values



Scattered plot of residuals vs.predicted values

From the first plot, we can see that most data are well fitted and fall within the range of 0-0.2 GB. Only a very small portion of data deviate from the ideal linear model (denoted as the dashed line with slope 1).

The second plot shows the residuals for every data point. Most predicted data have the

error less than 0.2 GB, and only a small portion of predicted data have relatively larger error greater than 0.2 GB.

In order to evaluate the importance of different attributes, the RMSE are calculated and listed in the chart below:

Dataset	***RMSE value***
Including all attributes	0.0885051678528
Ignoring attribute “week” only	0.0885365666582
Ignoring attribute “day of week” only	0.0902339057945
Ignoring attribute “backup start time” only	0.0904156660514
Ignoring attribute “work flow” only	0.0885054600665
Ignoring attribute “file name” only	0.0885042223827

In this chart, the original RMSE is 0.0885051678528, it is calculated based on the whole training dataset. To evaluate the importance of every attribute separately, we drop one attributes every time and compare the RMSE with the original value. If the new RMSE becomes smaller, then we can say that the dropped attribute is not important because it causes the model to be less accurate.

- By ignoring attribute “week” only, the RMSE becomes slightly larger. But the variation is very small and can be neglected. This corresponds to our speculation that the attribute “week” is not very important and it contributes little to the accuracy of the model.

- By ignoring attribute “day of week” only or “backup start time” only, the new RMSE becomes significantly larger, which means these two attributes are very important. Once they are excluded from the dataset, the model becomes inaccurate.

- By ignoring attribute “work flow” only, the new RMSE basically remains the same. This means that this attribute is possibly to be irrelevant.

- By ignoring attribute “file name” only, the new RMSE becomes smaller so the accuracy of the model is improved. We can conclude that the attribute “file name” causes overfitting and damages the accuracy. Therefore, it can be safely excluded from the dataset.

2b. Random Forest Regression

For random forest regression, there are three initial values to be set:

- n_estimators: The number of trees in a forest. Typically, more trees will decrease the variance of the model.
- max_depth: The maximum depth of the tree. Typically, deeper trees will decrease the bias of the model.
- max_features: The number of features to consider when looking for the best split.

In order to find the optimal number of trees in a forest, we tried to tune the parameter “n_estimators”. The max_dapth is set to be 4, and the max_feature is set to be default value (the number of features in the dataset). The results are shown below:

```
# of trees: 10   RMSE is: 0.0627186209979
# of trees: 20   RMSE is: 0.0628000680311
# of trees: 30   RMSE is: 0.0627247759653
# of trees: 40   RMSE is: 0.0627352799792
# of trees: 50   RMSE is: 0.0627554545522
# of trees: 60   RMSE is: 0.0627619684082
# of trees: 70   RMSE is: 0.062700378652
# of trees: 80   RMSE is: 0.0627477490526
# of trees: 90   RMSE is: 0.0626499876516
# of trees: 100   RMSE is: 0.062741280354
# of trees: 110   RMSE is: 0.0627719819882
# of trees: 120   RMSE is: 0.0627148743873
# of trees: 130   RMSE is: 0.0627287569682
```

```
# of trees: 140   RMSE is: 0.0627188179933
# of trees: 150   RMSE is: 0.0627191681006
# of trees: 160   RMSE is: 0.0627342994665
# of trees: 170   RMSE is: 0.0626974660608
# of trees: 180   RMSE is: 0.0627183933339
# of trees: 190   RMSE is: 0.0627156249662
```

We can see from the results that as the number of trees increases, the RMSE remains basically unchanged. In other words, the increase of number of trees does not contribute to the accuracy of the model, so we will use the initial value 20.

In order to find the optimal value for the maximum depth of the tree, we let the number of trees = 20, and max_features = the number of features in the dataset. By tuning the value of the depth of tree, we have the following results:

```
Depth of trees: 1   RMSE is: 0.0975424365973
Depth of trees: 2   RMSE is: 0.0895559136165
Depth of trees: 3   RMSE is: 0.0803349596555
Depth of trees: 4   RMSE is: 0.0627817660769
Depth of trees: 5   RMSE is: 0.0309338009904
Depth of trees: 6   RMSE is: 0.0222333067614
Depth of trees: 7   RMSE is: 0.0163998024619
Depth of trees: 8   RMSE is: 0.0143897713028
Depth of trees: 9   RMSE is: 0.0129912064174
Depth of trees: 10   RMSE is: 0.0131377230913
Depth of trees: 11   RMSE is: 0.0134151040999
Depth of trees: 12   RMSE is: 0.0136313868924
Depth of trees: 13   RMSE is: 0.0139226206851
Depth of trees: 14   RMSE is: 0.01412995415
Depth of trees: 15   RMSE is: 0.0143695862366
Depth of trees: 16   RMSE is: 0.01458485574
Depth of trees: 17   RMSE is: 0.0147100985907
Depth of trees: 18   RMSE is: 0.014888449559
Depth of trees: 19   RMSE is: 0.0149435206172
```

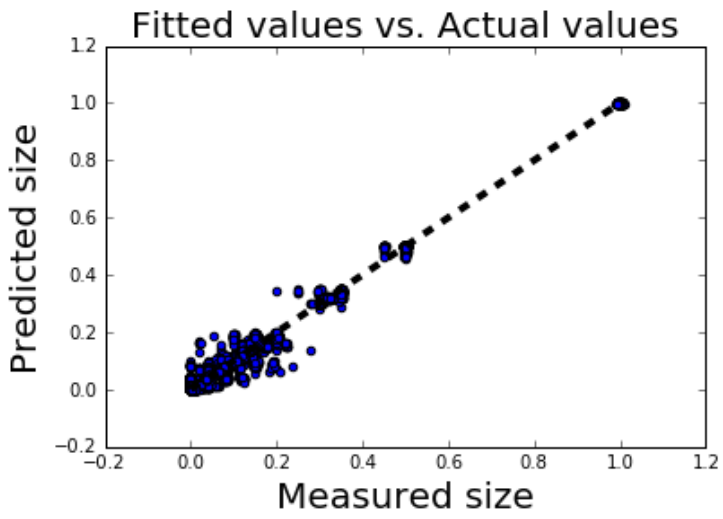
We can see from the results that when the depth of trees is less than 9, the RMSE keeps decreasing as the depth of trees increase; when the depth of tree is greater than 9, the RMSE becomes increasing as the depth of tree increase. Therefore, we should fix the depth of tree at 9, which is the optimal value.

In order to find the optimal value for the number of features, we let the number of trees to be 20, and depth of trees to be 9. By tuning the value of "max_features", we have the following results:

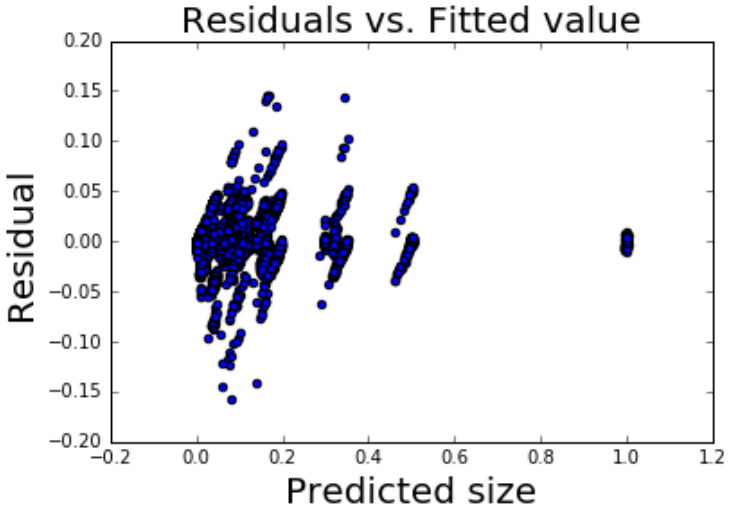
```
Max # of features: 30   RMSE is: 0.013977847822
Max # of features: 31   RMSE is: 0.013785422128
Max # of features: 32   RMSE is: 0.0136143448951
Max # of features: 33   RMSE is: 0.0134885736576
Max # of features: 34   RMSE is: 0.0135386444129
Max # of features: 35   RMSE is: 0.0132511118831
Max # of features: 36   RMSE is: 0.0133781969353
Max # of features: 37   RMSE is: 0.0132338447772
Max # of features: 38   RMSE is: 0.0131569572615
Max # of features: 39   RMSE is: 0.013046025145
Max # of features: 40   RMSE is: 0.0130268977989
Max # of features: 41   RMSE is: 0.0129949786104
Max # of features: 42   RMSE is: 0.0129768260321
Max # of features: 43   RMSE is: 0.0129754655541
Max # of features: 44   RMSE is: 0.0129010933883
Max # of features: 45   RMSE is: 0.012961583313
Max # of features: 46   RMSE is: 0.0129474118697
Max # of features: 47   RMSE is: 0.0129453737211
Max # of features: 48   RMSE is: 0.0129678330613
Max # of features: 49   RMSE is: 0.0129910915896
```

From these results we can see that as the "max_features" increases, the RMSE shows a decreasing trend, but at a very slow rate. For the purpose of lowering RMSE as much as possible, we choose to use 49 as the number of features (the maximum value we can use).

We have now explored all three parameters of this random forest model, so we use the optimal values (n_estimators = 20, max_depth = 9, max_features = 49) to train the model, and get the scattered plots shown below:



Scattered plot of predicted values vs. actual values



Scattered plot of residuals vs. predicted values

From the first plot we can see that all data points fall in the vicinity of the ideal dashed line. There is no data point deviating far away. So this is a better regression model compared to linear regression.

The second plot further supports the accuracy of random forest model. All predicted data now have the error less than 0.2 GB, with no exceptions. Compared to the linear model we did earlier, where some predicted data have error value larger than 0.8 GB, the advantage of random forest model is obvious.

In order to evaluate the importance of different attributes, the RMSE are calculated and listed in the chart below:

Dataset	***RMSE value***
Including all attributes	0.0129720739002
Ignoring attribute “week” only	0.0128756065322
Ignoring attribute “day of week” only	0.0950903227489
Ignoring attribute “backup start time” only	0.0685206586222
Ignoring attribute “work flow” only	0.0443982440183
Ignoring attribute “file name” only	0.012903577696
Ignoring both “file name” and “week”	0.0127203076007

From the chart we can see that by ignoring either “day of week”, “backup start time” or “work flow”, the RMSE is significantly increased, which means that these attributes are very important for building the model. By ignoring either “week” or “file name”, the RMSE basically remains unchanged, which means that these two attributes are irrelevant and could be dropped. If we drop both “week” and “file name”, we get even better RMSE = 0.0127203076007, which is by far the best result.

Finally, we get the optimal RMSE = 0.0127203076007. This is a substantial improvement compared to the linear model used earlier, which has the best RMSE = 0.0885051678528.

2c. Neural Network Regression

For neural network regression, there are three parameters to consider:

- learning_rate: This parameter determines the self-adjustment rate of the neural

network.

- units (Number of units): This parameter determines the number of units in the specific layer used for the neural network learning process.
- n_iter (Number of iteration): This parameter determines how many iterations will be employed for the regression process.

In order to obtain optimal parameter settings, we tune only one parameter at a time and observe the influence on RMSE.

For the number of units, we first set a fixed value for learning_rate = 0.1 and n_iter = 10. Then by tuning the number of units, we have the following results:

```
Number of units:10   RMSE is: 0.0723871903412
Number of units:20   RMSE is: 0.0605212261116
Number of units:30   RMSE is: 0.053503439395
Number of units:40   RMSE is: 0.271147599651
Number of units:50   RMSE is: 0.0588774076669
Number of units:60   RMSE is: 0.0670426458246
Number of units:70   RMSE is: 0.0797532612479
Number of units:80   RMSE is: 0.111812552422
Number of units:90   RMSE is: 0.0909362592338
Number of units:100   RMSE is: 0.103875617837
```

From the results, the RMSE does not display a good relationship with respect to the number of units, but we can see that the lowest RMSE appears when the number of units is 30. Thus, we can set the optimal value to be 30.

Next, we set fixed learning rate and number of units while tuning the number of iteration. The results are as below:

```
Number of iteration: 1   RMSE is: 0.0904206625882
Number of iteration: 4   RMSE is: 0.0850844318337
Number of iteration: 7   RMSE is: 0.0813957192392
Number of iteration: 10   RMSE is: 0.0401769604659
```

```
Number of iteration: 13   RMSE is: 0.0896282284815
Number of iteration: 16   RMSE is: 0.0425545530575
Number of iteration: 19   RMSE is: 0.0295881753885
Number of iteration: 22   RMSE is: 0.0458304195632
Number of iteration: 25   RMSE is: 0.0330277931062
Number of iteration: 28   RMSE is: 0.0421404053665
```

From the results, when n_iter is less than 19, the RMSE has a decreasing trend; when n_iter is greater than 19, RMSE becomes increasing. This means that the optimal value occurs at n_iter = 19. The most likely reason is because too many iterations would cause the model to overfit the training data, which has a negative impact on the quality of the model.

Next, we set fixed number of units and number of iteration while tuning the learning rate to find the optimal value. The results are as below:

```
Learning rate: 0.01   RMSE is: 0.091304039986
Learning rate: 0.05   RMSE is: 0.0494674449483
Learning rate: 0.1   RMSE is: 0.0356532825909
Learning rate: 0.2   RMSE is: 0.0289742282022
Learning rate: 0.3   RMSE is: 0.026621180691
Learning rate: 0.4   RMSE is: 0.0749748755125
Learning rate: 0.5   RMSE is: 0.0716080231692
Learning rate: 0.6   RMSE is: 0.118665103198
Learning rate: 0.7   RMSE is: 0.163764576446
```

As we can see from the results that the best RMSE occurs at learning_rate = 0.3, and RMSE = 0.026621180691. Theoretically, the RMSE would decrease as the learning rate increases, because the model will fit better for higher learning rate. However, as learning rate becomes reasonably high, the RMSE increases after some point (when learning rate is higher than 0.3). This is likely to be caused by overfitting that ruins the model.

Finally, we get the optimal neural network model for learning_rate = 0.3, n_iter = 19, number of units = 30. The optimal RMSE produced by this model is 0.026621180691.

3. Piece-wise Linear Regression and Polynomial Regression

Piece-wise linear regression

Based on the dataset provided in the question, we consider to perform the 10-fold cross validated linear regression on each workflow. To do this, we need firstly separate the original dataset into 5 different datasets, each of which contains all the same attributes for each work flow. And after the pre-processing, we do the same linear regression on each of the five datasets and evaluate the performance based on the RMSE. Here are the results:

```
Linear Regression RMSE for work flow 0 is: 0.0256013725565
Linear Regression RMSE for work flow 1 is: 0.113691601844
Linear Regression RMSE for work flow 2 is: 0.0311597035712
Linear Regression RMSE for work flow 3 is: 0.00532256935747
Linear Regression RMSE for work flow 4 is: 0.0544202232538
```

And also giving the RMSE for linear regression without piece-wise separation which is 0.0885. We can see that the fitting accuracy is not necessarily improved for all the workflows, e.g. for workflow 3, the piece-wise separation seems to improve the fitting and reduce the RMSE to a very low level, while for workflow 1, the RMSE even becomes larger after the piece-wise separation. But except for the workflow 1, the fitting of linear regression of each workflow are improved somehow.

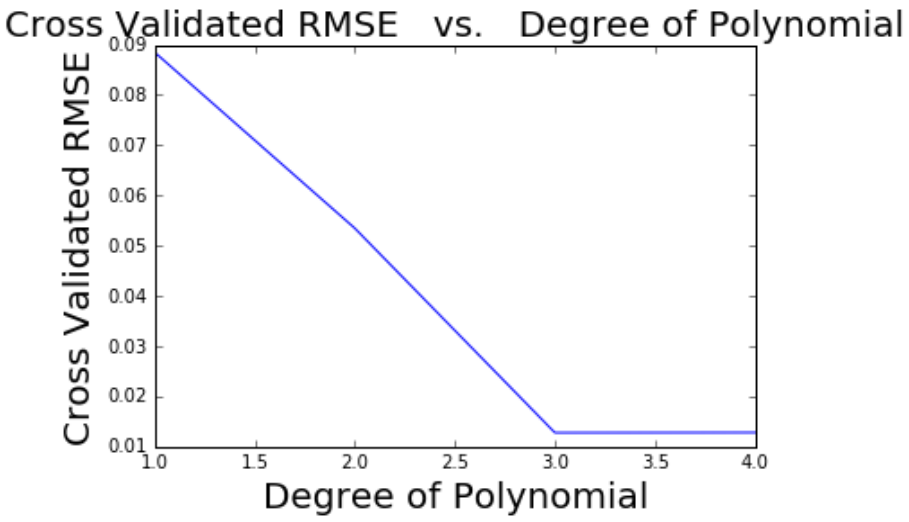
So based on this case, we cannot assume that the piece-wise separation will necessarily optimize the performance of the linear regression for all work flows.

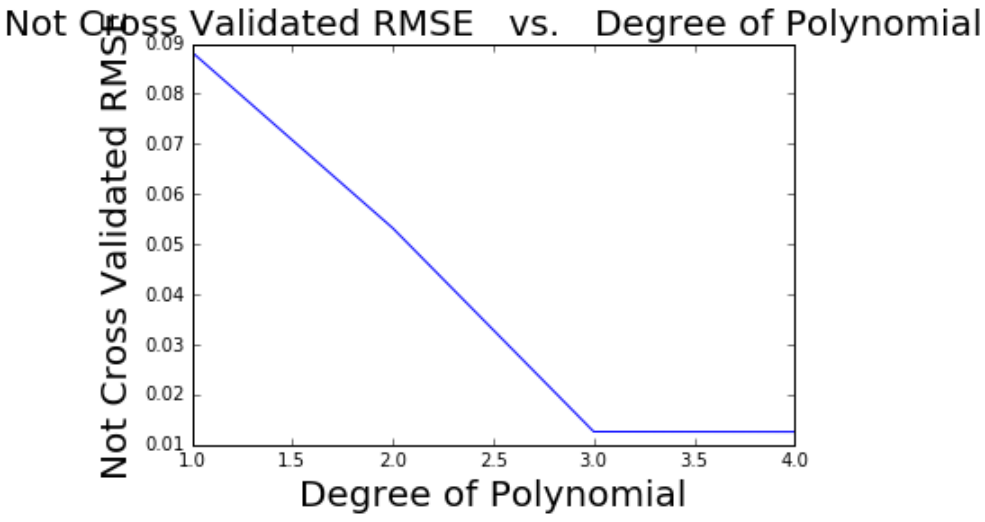
Polynomial regression with/without cross validation

In this part, we explored the fitting performance of the polynomial regression. And we tried to apply the polynomial regression on the fixed original dataset first, changed the degree of the polynomial penalty function from 1 to 4 to see how the RMSE changes. When the degree is larger than 4, we found that the computational complexity was too large for our laptops. So we stopped at degree 4 to see if there will be any pattern. Then

we applied the 10-fold cross validation on the original dataset and did the same process. Here are the results:

```
degree: 1  cross validated RMSE:0.0885150973468
degree: 1  normal RMSE:0.0883519249765
degree: 2  cross validated RMSE:0.0535299681796
degree: 2  normal RMSE:0.053444494396
degree: 3  cross validated RMSE:0.0133242851982
degree: 3  normal RMSE:0.0129790438827
```





RMSE of polynomial regression as function of polynomial degree with/without cross validation

We can see that as the degree of the polynomial penalty function increases, the RMSE of our fitting model decreases almost linearly, so we suppose that in this case, the RMSE

will decrease as the degree of polynomial penalty function increases. And among the 1 to 3 degree we explored. 3-degree penalty function has better performance as for the RMSE of the fitting model. And we can also see that after applying the 10-fold cross validation onto the dataset. The fitting RMSE is slightly reduced but not in a considerable manner.

Cross validation helps controlling the complexity of the model because it separates model selection from testing, and thus it performs estimate of generalization more conservatively. For over-fitting, obtaining more training data will help. For over-fitting degrees of complexity, training errors will be low and testing errors will be high. So cross validation will use the parameters obtained from the training data to test other data. In this process the complexity of the model could be controlled in some degree.

Boston housing price dataset

4. Linear Regression model

Based on the Boston housing price dataset provided, we split the dataset into two parts: the first one as the design matrix contains 500 samples of all 13 attributes except MEDV, and the other one as the target data which contains 500 samples of MEDV. And for this part, we perform basic linear regression, 10-fold cross validation based linear and polynomial regression on the dataset. We evaluate the performance of different regression methods based on the average Rooted Mean Square Error, i.e. lower RMSE means lower training error.

Basic linear regression

We know the penalty function of basic linear regression is:

$$\hat{\beta} = \arg\min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|_{\ell_2}^2 = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}$$

In Python, it is very easy to perform the linear regression on the practical dataset. We can use the methods provided in the scikit-learn package. And will get the results:

```
Beta = [ -1.07416993e-01, 4.61207350e-02,1.42689710e-02,
2.67110782e+00, -1.76336414e+01, 3.79430680e+00,
1.07616131e-03, -1.47917943e+00, 3.01534125e-01,
-1.20534342e-02, -9.58873592e-01, 9.30540151e-03,
-5.27599617e-01]
RMSE = 21.8655799334
Mean price = 22.5299009901
```

We can see the array consists of 13 float number stands for the model we obtained from the linear regression based on the dataset. And the RMSE of this basic regression is 21.8655. Giving the average housing price 22.53, this error is literally unacceptable. And we will explore some other optimized regression method to reduce it.

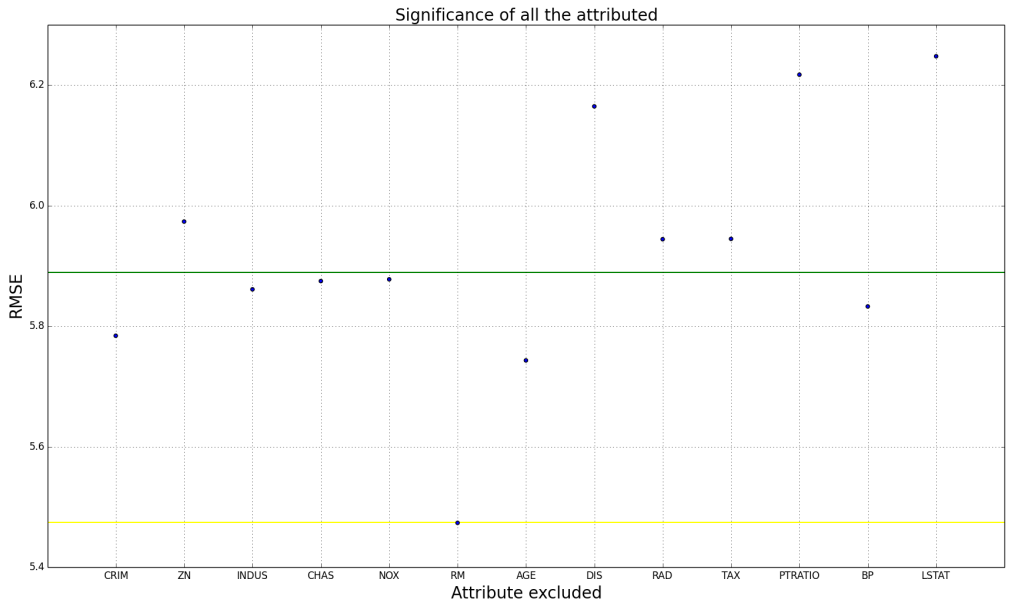
Analysis on the significance of different attributes

For this part, we analyze the significance of the 13 attributes with the 10-fold cross validation linear regression. The method is simple: we just perform the validated linear regression on the original dataset multiple times, and we exclude a certain attribute each time to see if the RMSE will be reduced or not when excluding it. Here are the results:

```
Mean price = 22.5299009901
RMSE for the 10-fold CV = 5.88895135442
RMSE(CRIM excluded) = 5.78403459227
RMSE(ZN excluded) = 5.97351320533
RMSE(INDUS excluded) = 5.86104292233
RMSE(CHAS excluded) = 5.8747764894
RMSE(NOX excluded) = 5.87764054869
RMSE(RM excluded) = 5.47353927356
RMSE(AGE excluded) = 5.7431036997
RMSE(DIS excluded) = 6.1646566271
RMSE(RAD excluded) = 5.94418959532
RMSE(TAX excluded) = 5.94478878545
RMSE(PTRATIO excluded) = 6.21721610524
```

```
RMSE(BP excluded) = 5.83265559215
RMSE(LSTAT excluded) = 6.24776085155
```

The RMSE after optimization according to significance of attributes = 5.47461875538



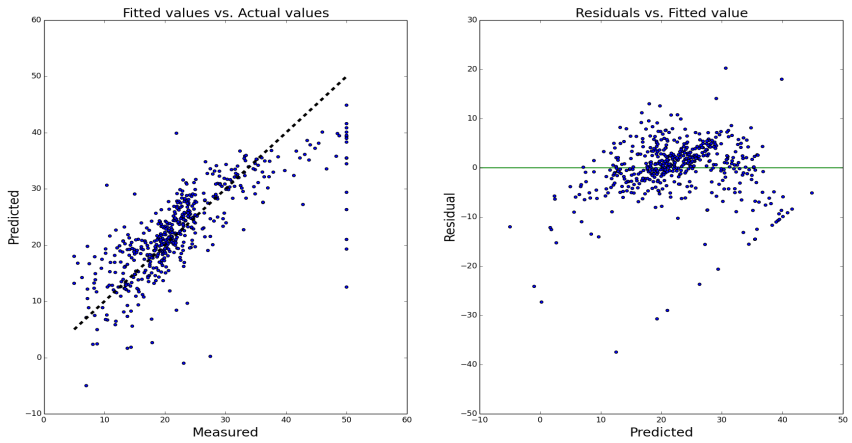
RMSE after excluding each of the attributes

The green horizontal line in the plot shows the RMSE when there is no attribute excluded. And we can see that after excluding the attribute DIS, PTRATIO and LSTAT, the RMSE increased dramatically. So these three attributes shall have a significant impact on the housing price. On the contrary, the RMSE dropped a lot after excluding the attribute RM, which means this attribute have a trivial impact on the housing price, or it could even bring the noise for our prediction somehow. The yellow line in the plot shows the RMSE without the RM contribution. So for the optimization, we could exclude this attribute for the following project.

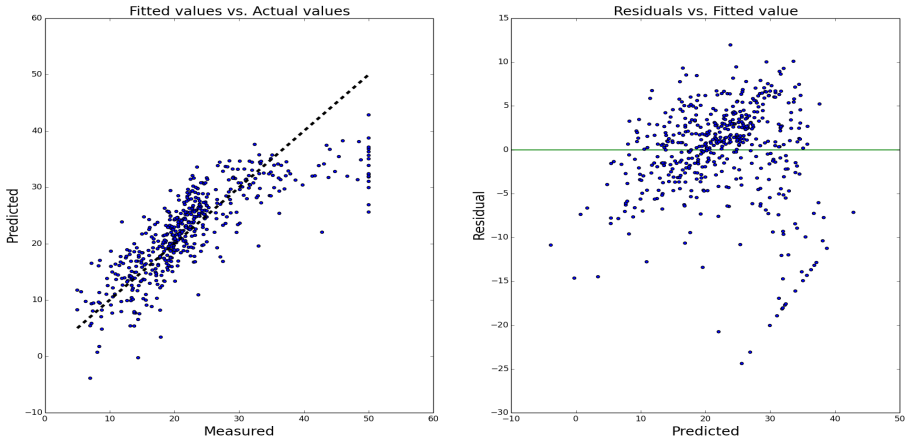
More detailed analysis of the regression

To obtain a more detailed evaluation of the performance, we plotted the scattered

predicted price and the residual verse the true data. The first one is based on the model trained with 10-fold validated linear regression, and the second one is the optimized version based on the analysis of attribute significance.



Before excluding the RM attribute



After excluding the RM attribute

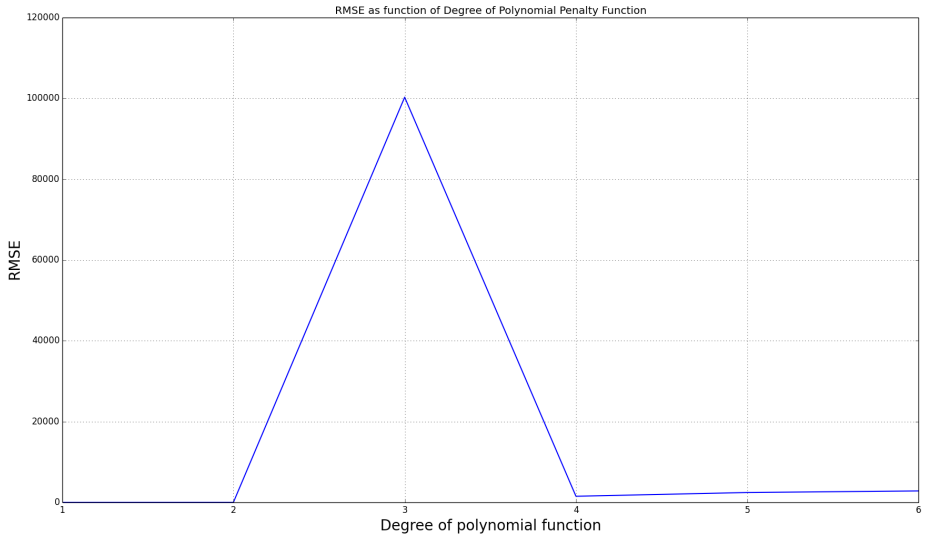
We can see that compared with the first one, the predicted-measured spots in the second plot is more concentrated along the y=x line and the residual spots are also more concentrated along the horizontal zero line, which means the model in the second plot has reduced training error compared with the first one.

Polynomial regression

In this part, we analyzed the performance of polynomial regression. The basic method

of polynomial regression is expanding the training design matrix by adding some higher polynomial terms. The methods provided in the scikit-learn package could help us do the job. And we found that the RMSE is slightly reduced if we include the interactional terms. So the following results are all based on interactional-term-included polynomial regression:

```
RMSE(with 1 degree polynomial penalty function) = 5.88895135442
RMSE(with 2 degree polynomial penalty function) = 9.31942783266
RMSE(with 3 degree polynomial penalty function) = 100296.242904
RMSE(with 4 degree polynomial penalty function) = 1518.76386302
RMSE(with 5 degree polynomial penalty function) = 2452.33130962
RMSE(with 6 degree polynomial penalty function) = 2838.18151266
```

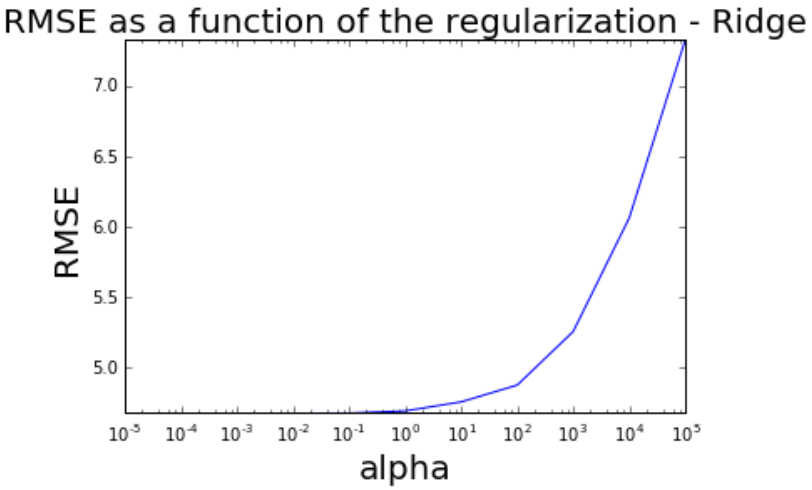


RMSE of polynomial regression of different function degree

We can see that as the degree of penalty function increases, the RMSE increases as well at first, especially when degree increases from 2 to 3, the RMSE increases dramatically to some unacceptable level. So in our case, degree 2 is a threshold for the polynomial regression, and the optimized degree should be 1, i.e. it's better to not do polynomial expanding of the design matrix.

5. Regularization of parameters

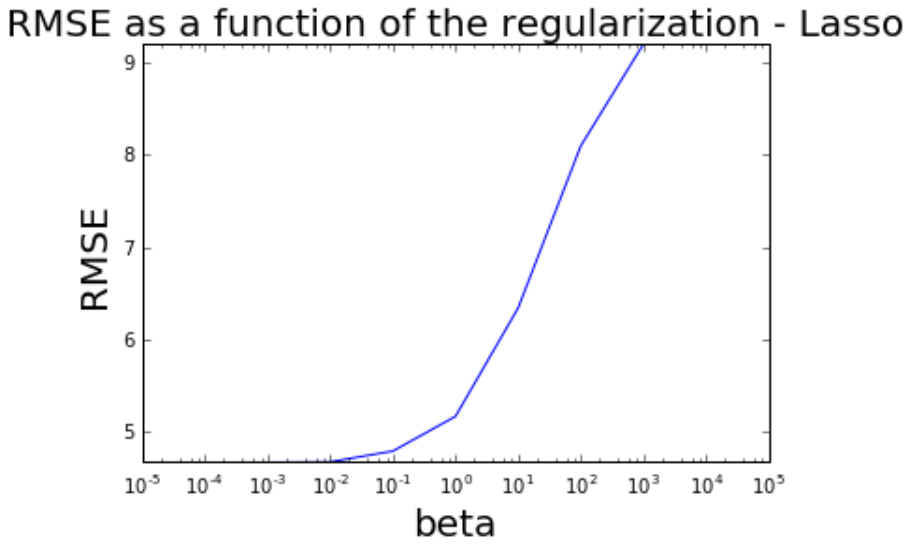
Ridge Regression is similar to least squares but shrinks the estimated coefficients towards zero. Ridge Cross Validation implements Ridge Regression with built-in cross-validation of the alpha parameter. Ridge Cross Validation generates the best alpha after performance tuning on the given dataset. We applied ridge regression based on the different alpha values. We got the result table as follow.




```
RMSE of Ridge(tuning factor: alpha = 1e-05) = 4.67606457755
RMSE of Ridge(tuning factor: alpha = 0.0001) = 4.67606457797
RMSE of Ridge(tuning factor: alpha = 0.001) = 4.67606462032
RMSE of Ridge(tuning factor: alpha = 0.01) = 4.67606880589
RMSE of Ridge(tuning factor: alpha = 0.1) = 4.67644236648
RMSE of Ridge(tuning factor: alpha = 1.0) = 4.69183331215
RMSE of Ridge(tuning factor: alpha = 10.0) = 4.75615797025
RMSE of Ridge(tuning factor: alpha = 100.0) = 4.87523024131
RMSE of Ridge(tuning factor: alpha = 1000.0) = 5.25551694461
RMSE of Ridge(tuning factor: alpha = 10000.0) = 6.06097529981
RMSE of Ridge(tuning factor: alpha = 100000.0) = 7.3235264498
```

Therefore, the best Alpha value for Ridge Regression is 0.01. The best RMSE for corresponding alpha is 4.676.

The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. For this reason, the Lasso and its variants are fundamental to the field of compressed sensing. The library used for performing lasso regression is lasso. After performance tuning, the alpha obtained is given below. We applied lasso regression based on the different alpha values. We got the result table as follow.




```
RMSE of Lasso(tuning factor: beta = 1e-05) = 4.67606458146
RMSE of Lasso(tuning factor: beta = 0.0001) = 4.67606496906
RMSE of Lasso(tuning factor: beta = 0.001) = 4.67610373666
RMSE of Lasso(tuning factor: beta = 0.01) = 4.67996666603
RMSE of Lasso(tuning factor: beta = 0.1) = 4.79636890881
RMSE of Lasso(tuning factor: beta = 1.0) = 5.1719578334
RMSE of Lasso(tuning factor: beta = 10.0) = 6.34376377669
RMSE of Lasso(tuning factor: beta = 100.0) = 8.09611465143
RMSE of Lasso(tuning factor: beta = 1000.0) = 9.1968718848
RMSE of Lasso(tuning factor: beta = 10000.0) = 9.1968718848
RMSE of Lasso(tuning factor: beta = 100000.0) = 9.1968718848
```

Therefore, the best Alpha value for Lasso Regression is 0.01. The best RMSE for corresponding alpha is 4.6796.