EE 219

Large-Scale Data Mining

Project 4

Regression Analysis

Winter 2018

By Xudong Li (804944940),

Tao Wu (504946672),

Yangyang Mao (504945234),

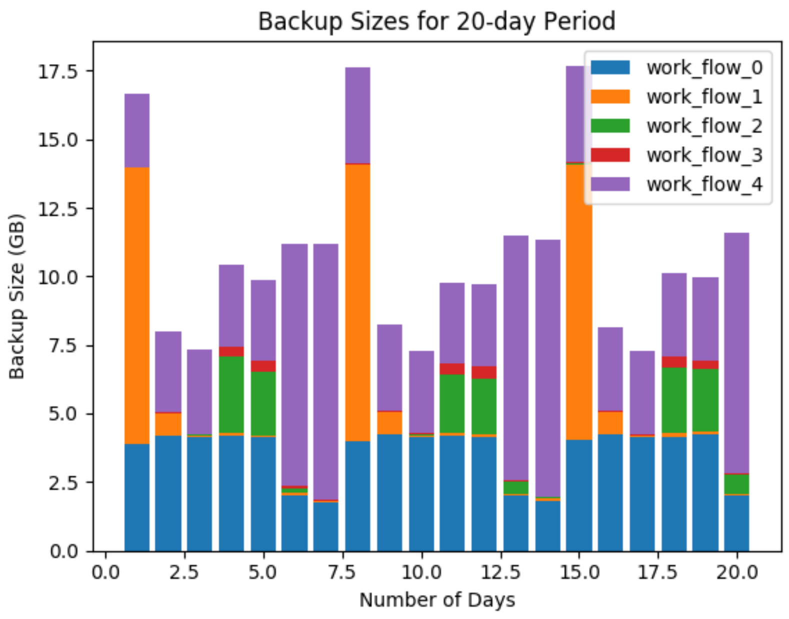
Di Jin (305026178)

March 4, 2018

Introduction

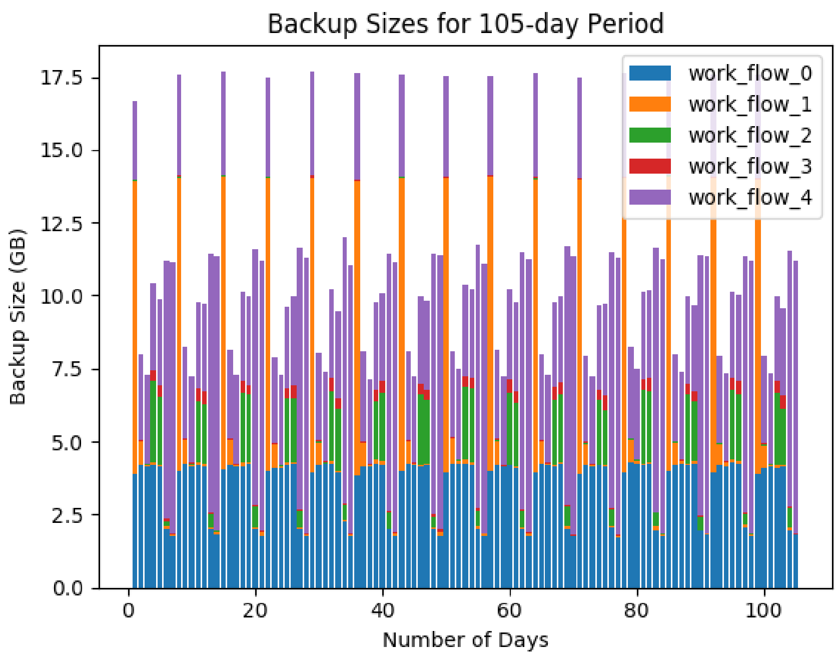
Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, we explore basic regression models on a given dataset, along with basic techniques to handle overfitting; namely cross-validation, and regularization. With cross-validation, we test for over-fitting, while with regularization we penalize overly complex models. The model we used in this project includes Linear Regression, Polynomial Regression, Random Forest Regression, Neural Network, and K-Nearest Neighbor Regression.

Problem 1(a)



In this figure, we plot the first 20-day period of the backup sizes for all workflows, and each workflow is color coded on the y-axis. We can see a pattern that the backup sizes are always highest on Monday, especially the workflow_1 has very large sizes. Meanwhile, workflow_4 has large backup sizes on weekend.

Problem 1(b)



Question: Can you identify any repeating patterns?

Answer: Yes. We can see that the backup size is similar for the same day of week, and the pattern repeats every week. Monday always has the highest backup size while the weekend (i.e. Saturday and Sunday) has the second largest backup size. Wednesday is always the lowest during the whole week, and we can use these patterns to do feature extractions. In addition, different workflows also behavior differently as described in the previous question, and we can see that workflow_0 is quite consistent except on weekends.

Problem 2(a) (i)

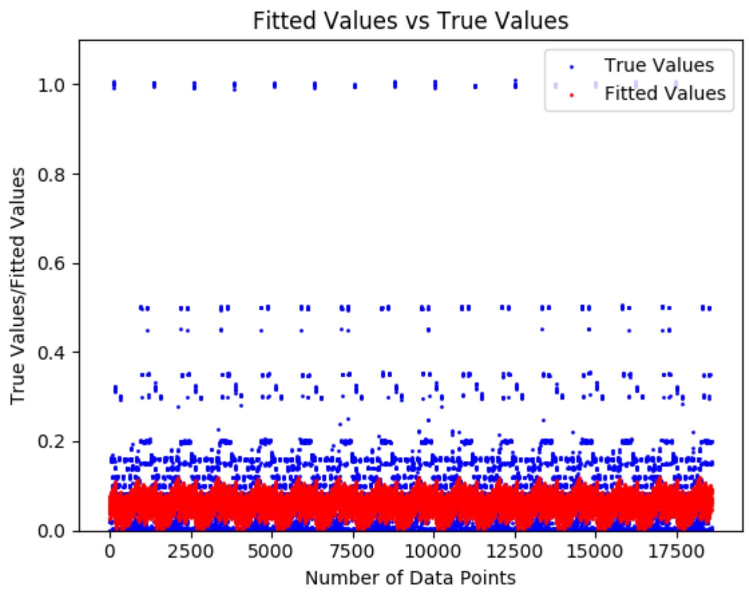
In this part, we use a linear regression model to evaluate our dataset. Each categorical feature is converted into one dimensional numerical values using scalar encoding. We report training and testing RMSE with a plot of a fitted values against true values and a plot of residuals against fitted values over the number of data points.

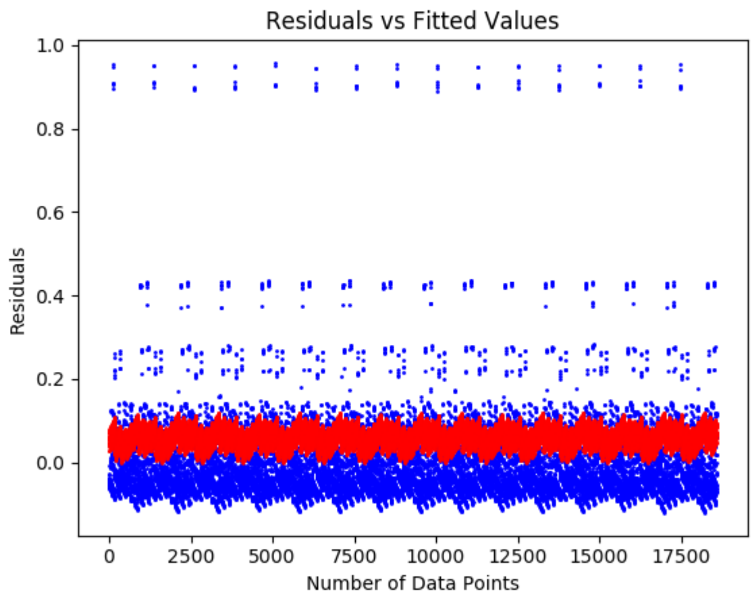
Training RMSE:

0.10154324	0.10215321	0.10153619	0.1021405	0.10150314
0.10212658	0.10151588	0.10212573	0.10151381	0.10218052

Testing RMSE:

0.10452384	0.0990271	0.10457893	0.0991425	0.10486548
0.09927142	0.10475605	0.09928238	0.10477973	0.09878119





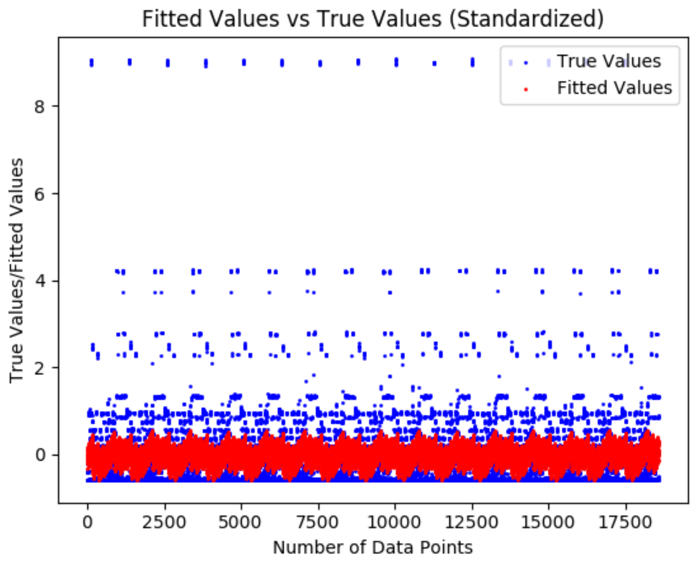
Problem 2(a) (ii)

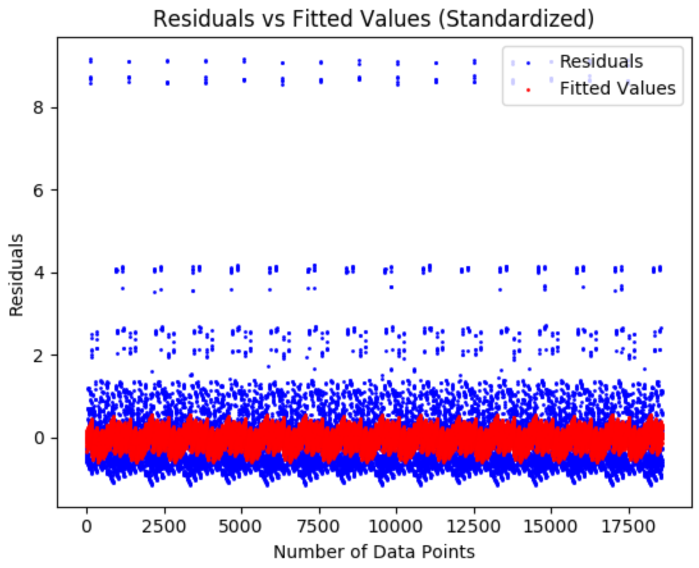
Training RMSE:

0.10154324	0.10215321	0.10153619	0.1021405	0.10150314
0.10212658	0.10151588	0.10212573	0.10151381	0.10218052

Testing RMSE:

0.10452384	0.0990271	0.10457893	0.0991425	0.10486548
0.09927142	0.10475605	0.09928238	0.10477973	0.09878119





From the above RMSE and plots, we can see that after standardization, both the training and testing RMSE stays the same, and the shape of the plots is also the same except the scale changes due to standardization. This is because when we normalize the features, we subtract it from its mean and divided by its standard deviation, and the new linear model with new features will get same result because the relationship between the data points does not change, only the scale changes. Therefore, linear regression model is invariant to normalizations.

Problem 2(a) (iii)

Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

MI:

0	0. 23328747	0. 3051186	0. 76380209	0. 76745863

F-score reveals the discriminative power of each feature independently from others. One score is computed for the first feature, and another score is computed for the second feature. The larger the F-score is, the more likely this feature is more discriminative.

F-score:

8.45006257e-03	2.20612122e+02	1.50740934e+02	2.61386654e+01	4.60786474e+02

The p-value of a feature selection score indicates the probability that this score or a higher score would be obtained if this variable showed no interaction with the target. So, the lower the P value, the stronger the correlation between the feature and the target.

P-val:

9.26759e-001	1.28086e-049	1.62474e-034	3.20909e-007	5.44826e-101

Feature Selection

From the p-values, we can see that the first feature ("Week #") has a very high value. According to the p-value definition above, the first feature does not have much interaction with the target, and therefore we can eliminate this feature. From the F-scores, we can see that the first and fourth features ("Week #" and "Workflow ID") are not very discriminative. In addition, the mutual information score shows that there is a strong dependence between the “Workflow ID” and “File Name”, so we only need to choose one of them. From all three scores, we conclude the three most important features are “Day of Week”, “Start Time”, and “File Name”.

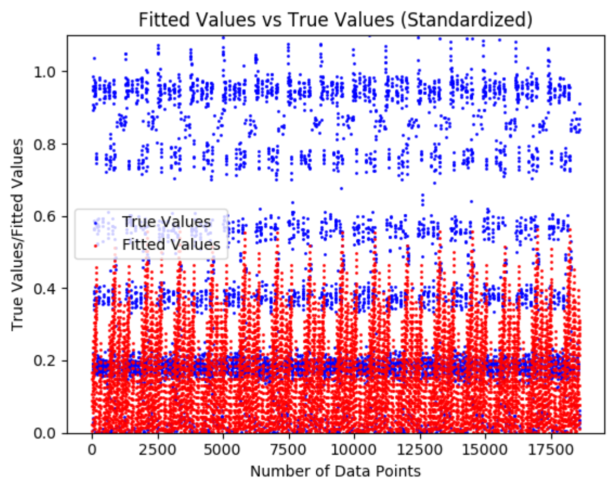
Training RMSE:

0. 97520332	0. 98059805	0. 9751078	0. 980474	0. 97479154
0. 98033417	0. 97490715	0. 98032638	0. 97490239	0. 98086721

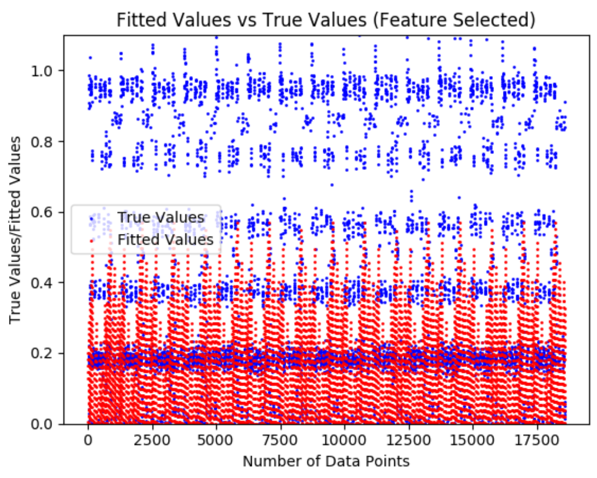
Testing RMSE:

1.00061066	0. 95200077	1. 00147201	0. 95315837	1. 00422706
0. 95443248	1. 00324183	0. 95451631	1. 00327069	0. 94949069

Before

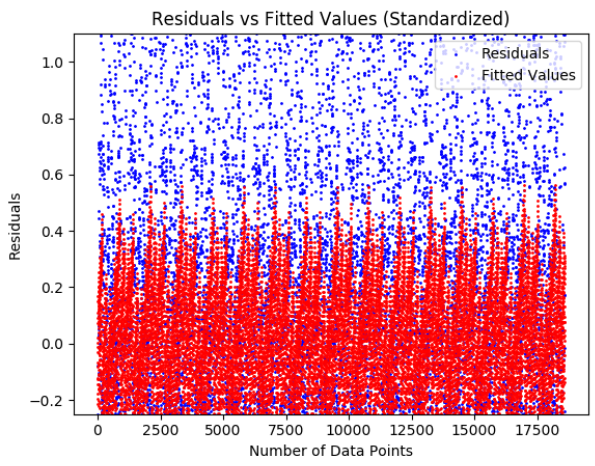


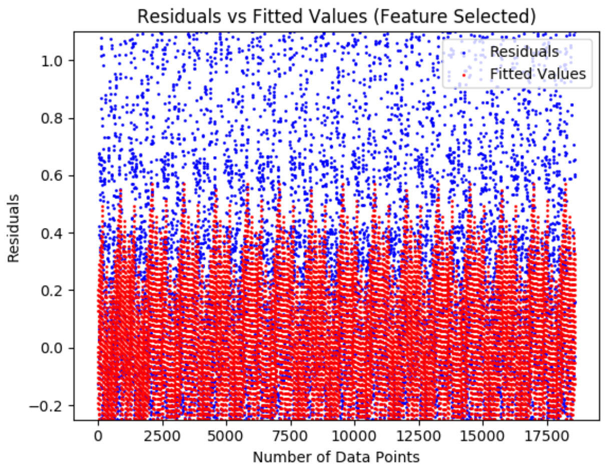
After



Note: the above plot is zoomed-in to show the difference

Before **After**

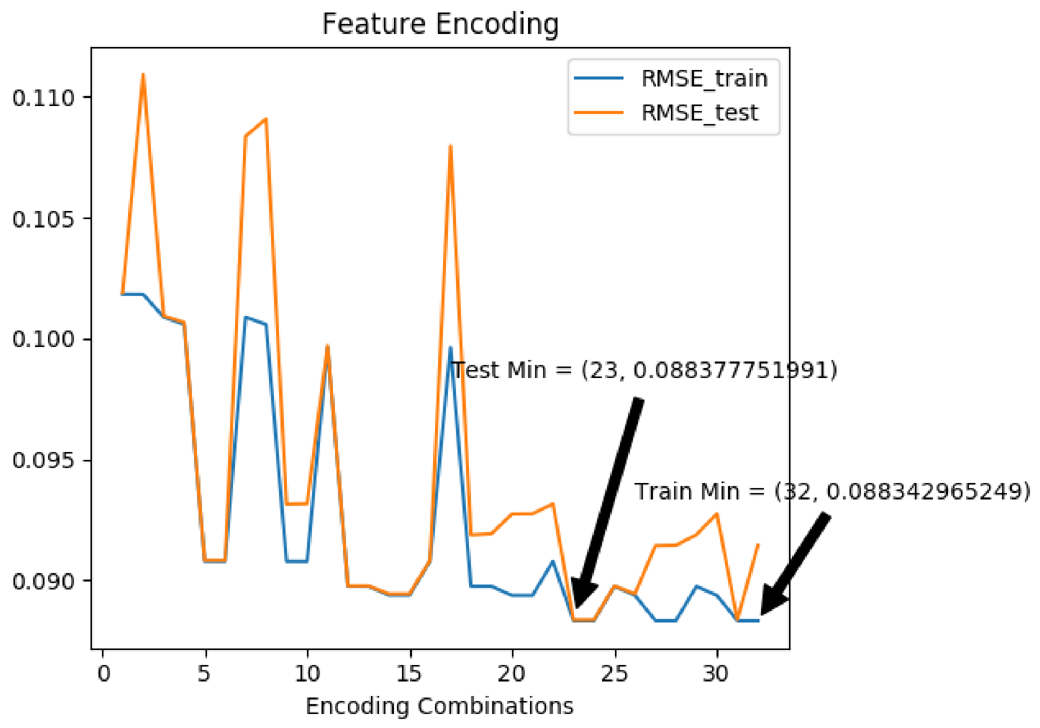




Note: the above plot is zoomed-in to show the difference

After the feature selection, we can see that the performance of the model increases. Both the training and testing RMSE decreases comparing to the linear regression model without any feature selection. From the plots, we can see the red fitted values are more uniform than before. It is easy to see the dots are organized into small dashed lines instead of a group of dense points before the feature selection. This is because we removed the unimportant features like week number, which basically acted as noise of the dataset. When we remove it, the performance becomes better and pattern in the dataset shows up.

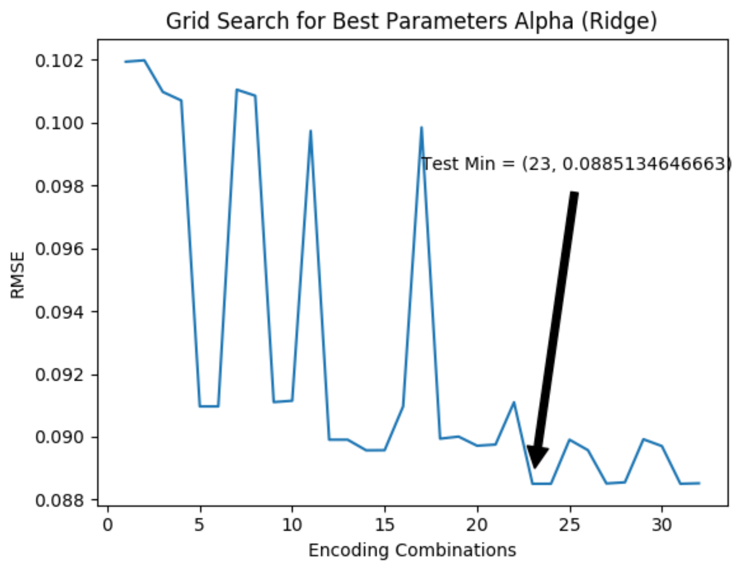
Problem 2(a) (iv)

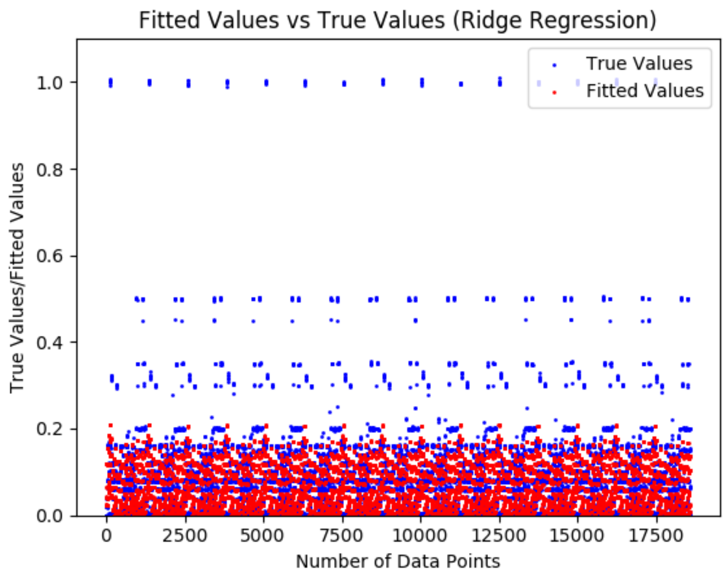


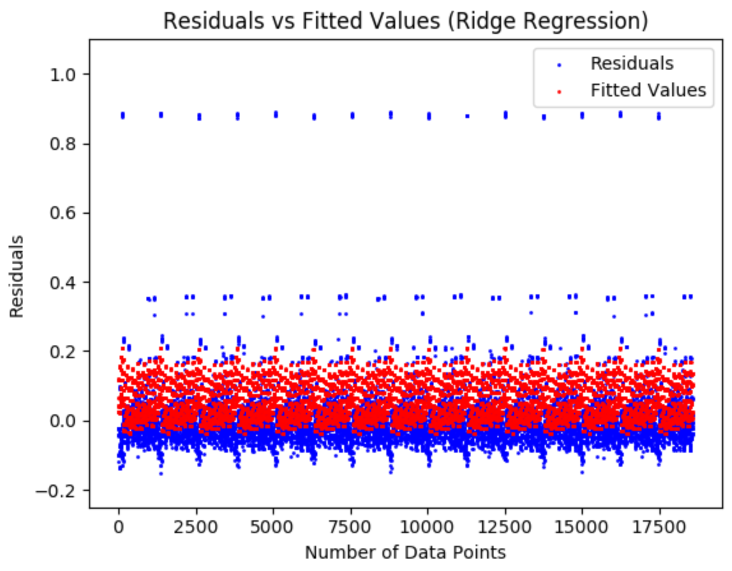
From our implementation of feature encoding, the higher the number in encoding combinations, the more One-Hot-Encoding we used. i.e. 1 represents all Scalar Encoding, and while 32 represents all One-Hot-Encoding. The number in between follows an order that we one-hot-encode features from left to right, starting from a single feature then to all five features. Surprisingly, the plot shows that the combinations that achieve the best performance is one-hot-encoding feature columns [1,2,3] ("Day of Week", "Backup Start Time", "Work-Flow-ID"), and [1,2,4] ("Day of Week", "Backup Start Time", "File Name"). This exactly corresponds to the features that we want to select in the previous feature selection part. Since column 3 and 4 ("Work-Flow-ID" and "File Name") have strong dependence, and it is very reasonable to see that these two combinations produce similar result. In addition, from the general trend of the graph, we can see that as we one-hot-encode more features, the RMSE decreases. This is because one-hot-encoding removes the correlations that induced by the standard scaler. For example, we scalar encode the "File Name", some file name will get a value of 24 while some will get 1. However, this implicates that 24 is larger than 1, and therefore means that File_24 is inherently greater than File_1, but this is not the case. By one-hot-encoding such feature, we remove the correlation so that each variable weight the same.

Problem 2(a) (v)

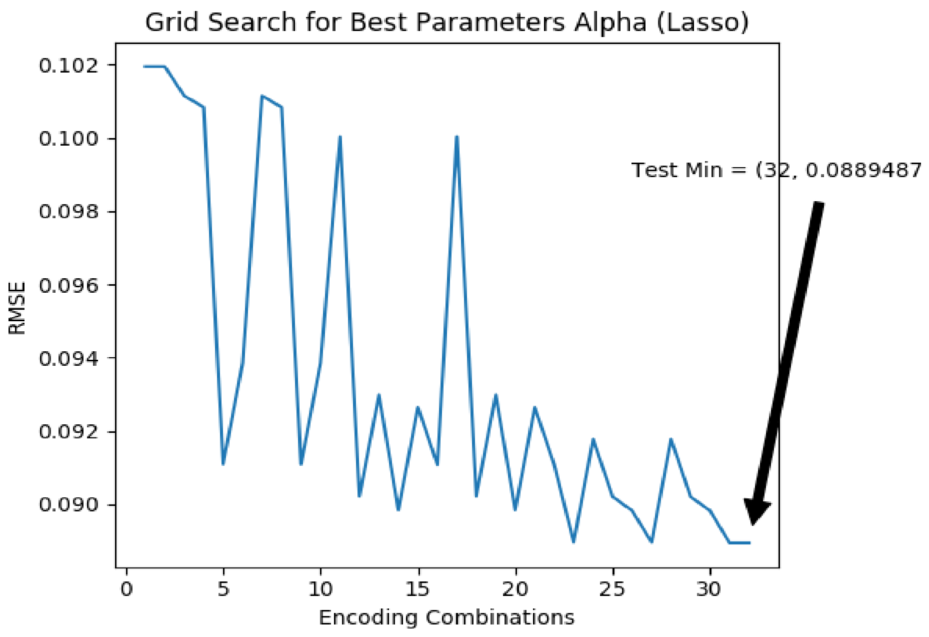
The test RMSE is greater than the training RMSE in some combinations is because the effect of overfitting. Without any regularizations, the model can learn the training too well that it overfits the training data. When new data is given (i.e. the testing data), since the model has not seen the data before, it fails to make good prediction based on the testing data. A possible solution to this problem is to add a penalty term at our model. We can use L1 or L2 regularization, i.e. the Ridge Regularizer and Lasso Regularizer to penalize any overfitting. Therefore, our model will follow the general trend of the data while not fit it too well.

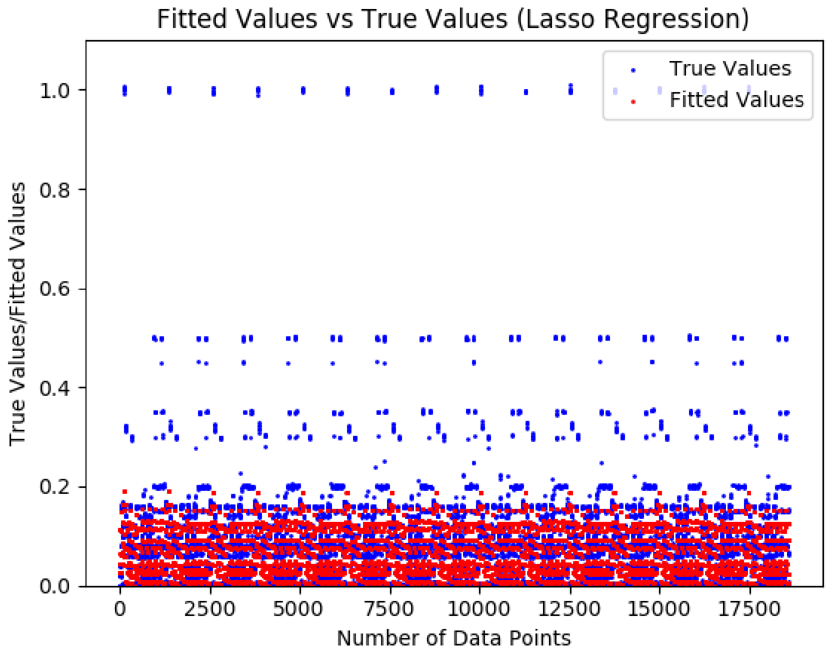


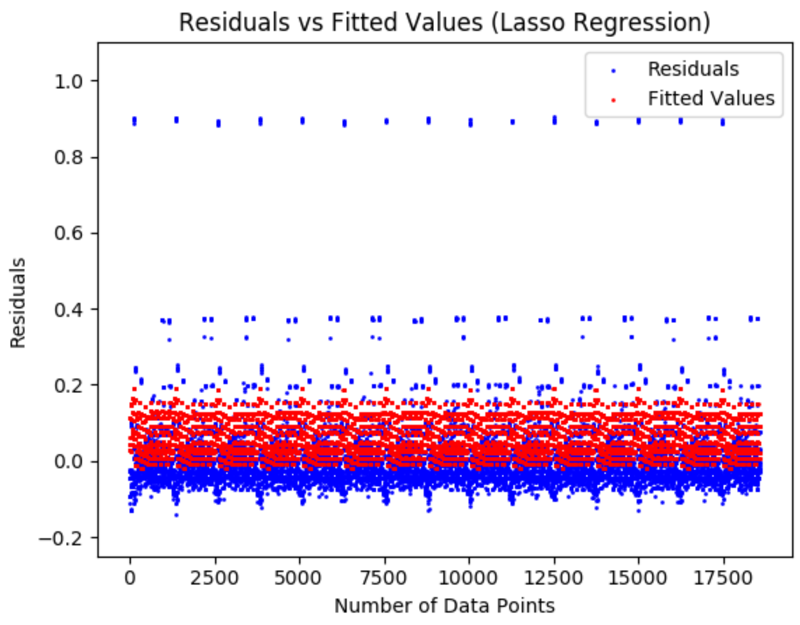




We use Grid Search to find the alpha for the Ridge Regularizer, and the regularization coefficient is 1, which is a normal regularization with the coefficient neither too strong or too weak. The lowest testing RMSE is 0.08851346 when we one-hot-encoding the columns [1,2,4] ("Day of Week", "Backup Start Time", "File Name"), and scalar encoding the rest of the columns. This is very similar to the case when we do linear regression, where we also encode the columns like this, except we add a regularization term. The estimated coefficient becomes a little bit larger than the coefficient from the best un-regularized linear model, because the regularization coefficient is 1 that it penalizes the overfitted coefficient. From the above plot, we can see that the testing RMSE is almost the same to the training RMSE in the linear regression, which makes sense because the added regularization term prevents overfitting, and therefore decreases the testing RMSE to the training level. Because the testing error should always be greater than the training error, our model produces a quite good result.


Grid Search for Best Parameters Alpha (Lasso)
Test Min = (32, 0.0889487
RMSE
0.102
0.100
0.098
0.096
0.094
0.092
0.090
0
5
10
15
20
25
30
Encoding Combinations

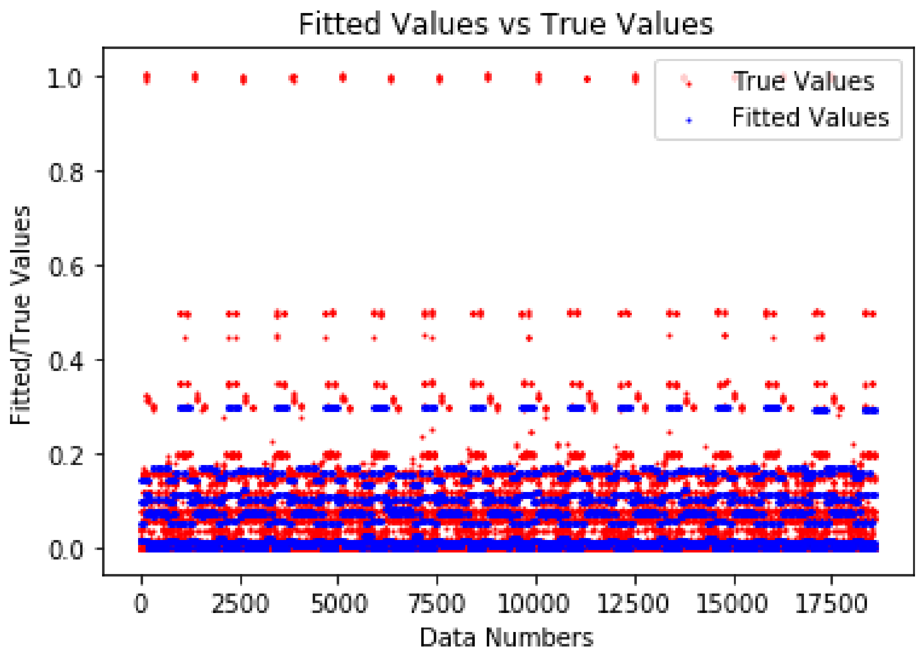

Fitted Values vs True Values (Lasso Regression)
True Values
Fitted Values
True Values/Fitted Values
1.0
0.8
0.6
0.4
0.2
0.0
0
2500
5000
7500
10000
12500
15000
17500
Number of Data Points

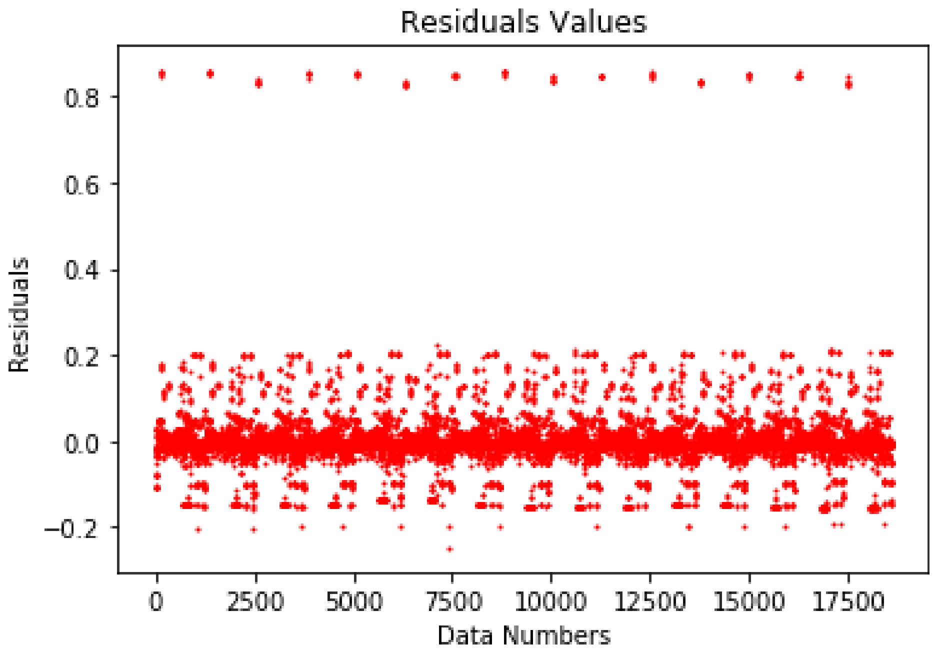


We use Grid Search to find the alpha for the Lasso Regularizer, and the regularization coefficient is 0.001, which is a very weak regularization with the coefficient being small. The lowest testing RMSE is 0.0.0889487 when we one-hot-encoding the columns [0,1,2,3,4] ("Week #", "Day of Week", "Backup Start Time", "WorkFlow ID", "File Name"), and there is no scalar encoding. Compare to the Ridge Regularizer above, the Lasso Regularizer does not perform that well. We can see many the local peaks that does not exist in Ridge. However, compare to the linear regression, the testing RMSE is still better in general. It is also very close to the training RMSE in the linear case. The estimated coefficient is almost the same as un-regularized best model since the regularization strength alpha is very small, and therefore little penalty is added to the estimated coefficient.

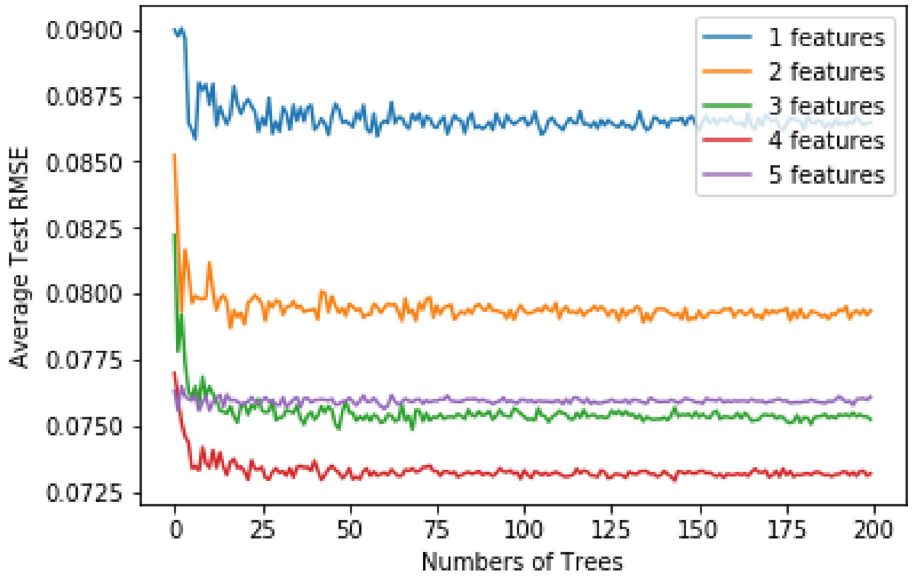
Problem 2(b) (i)

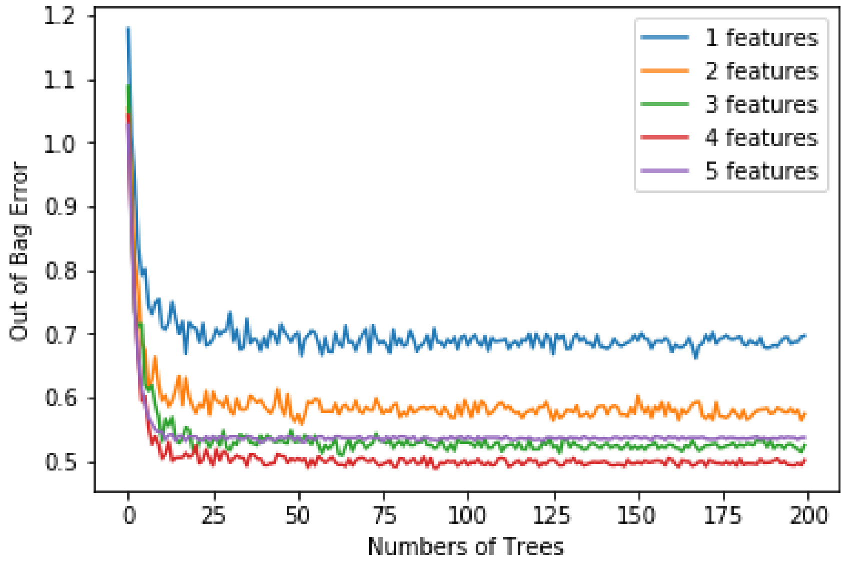
```
RMSE_Train: [ 0.07580245  0.07626825  0.07584806  0.07584232  0.07581633
0.075191
  0.07581551  0.07617055  0.07582936  0.07629597]
RMSE_Test: [ 0.08143949  0.07104488  0.08095383  0.0702578   0.08111626
0.07063382
  0.08093771  0.07100256  0.0808438   0.07123764]
Out of bag error: 0.537443568059
```





Problem 2(b) (ii)

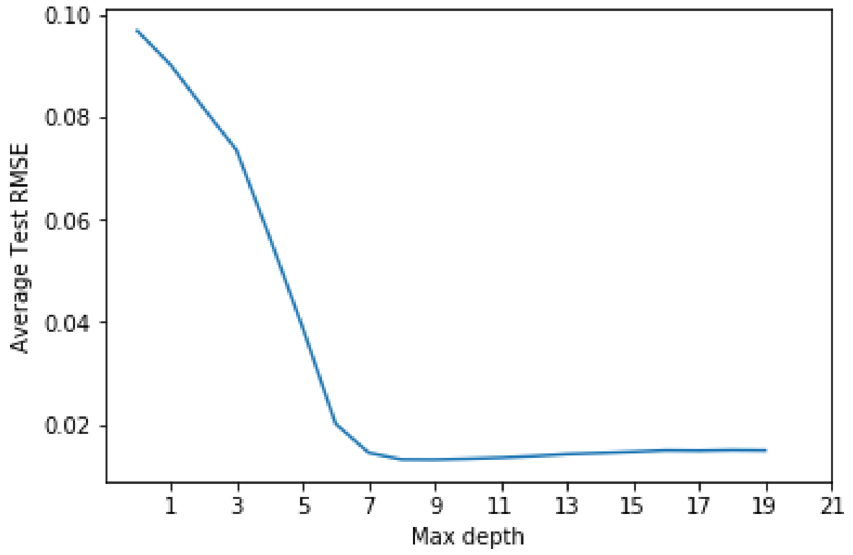


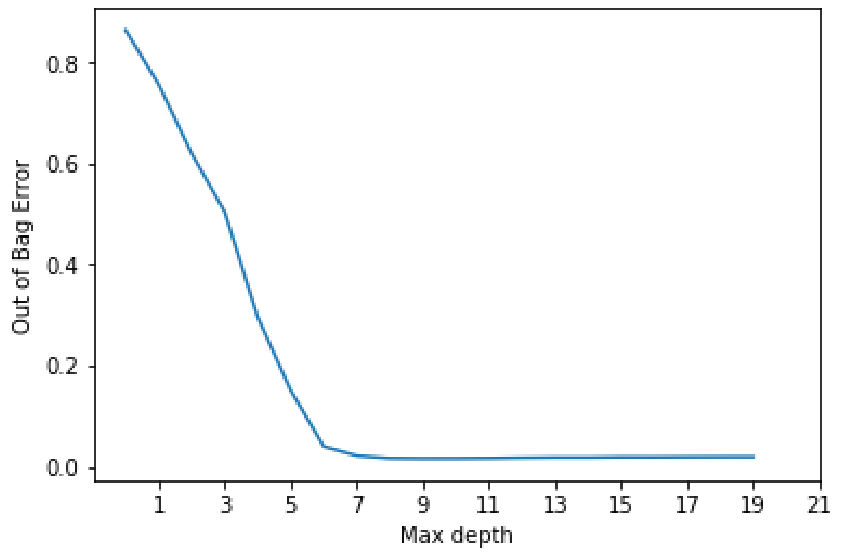


Analysis:

From above graphs we can find that the out of bag error and average RMSE decease with number of trees. When number of trees increase to 10, both RMSE and out of bag error are going to fluctuations around the minimum. As for the number of features, RMSE and out of bag error are smallest when the feature number is 4.

Problem 2(b) (iii)





Analysis:

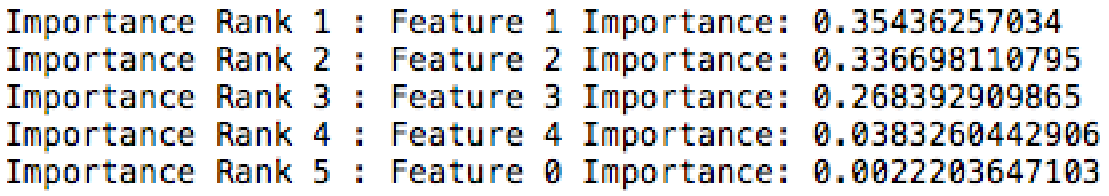
From above result, we choose tree number to be 25 and max feature number to be 4 and sweep over max depth from 1 to 20. We can find that the OOB error and RMSE decease sharply with the max depth and reach the minimum around 7 and 8 and then keep stable around the minimum. To get the best performance, I will choose following parameters:

Tree number:25

Maximum number of features: 4

Max depth: 8

Problem 2(b) (iv)



```
Importance Rank 1 : Feature 1 Importance: 0.35436257034
Importance Rank 2 : Feature 2 Importance: 0.336698110795
Importance Rank 3 : Feature 3 Importance: 0.268392909865
Importance Rank 4 : Feature 4 Importance: 0.0383260442906
Importance Rank 5 : Feature 0 Importance: 0.0022203647103
```

Analysis:

Feature Importance Rank:

Rank 1: Day of Week

Rank 2: Backup Start Time - Hour of Day

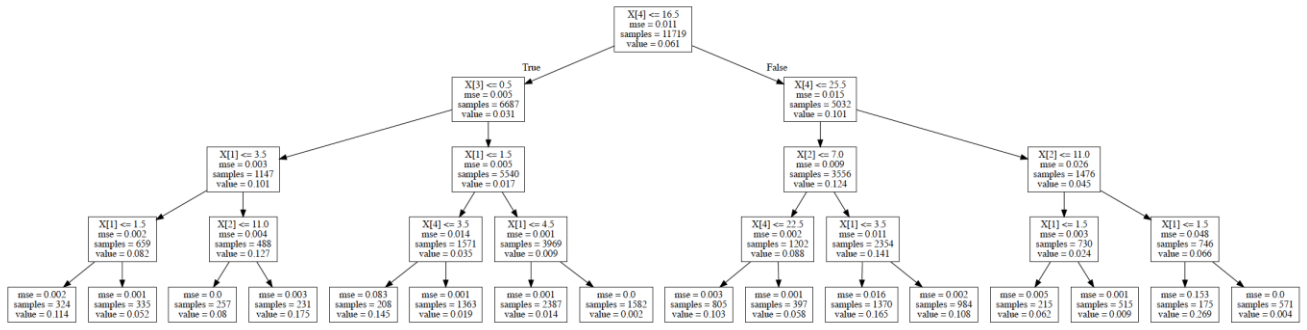
Rank 3: Work-Flow-ID

Rank 4: File Name

Rank 5: Week #

Day of week, backup start time and workflow ID are the top 3 in the feature importance. And the importance for the first two features are similar and followed by workflow with 26.8. Compared to above three features, file name and week number are not so important and only get 0.03 and 0.002 in feature importance.

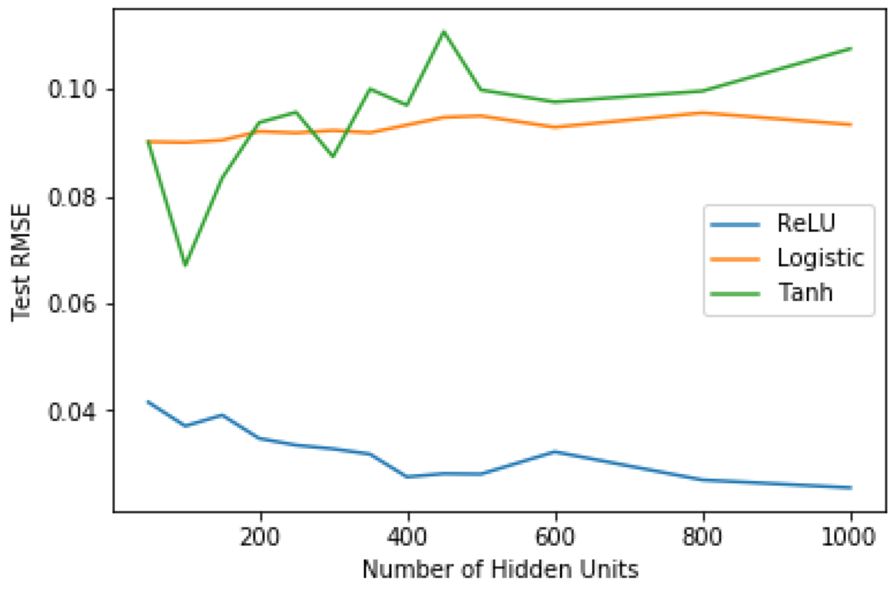
Problem 2(b) (v)

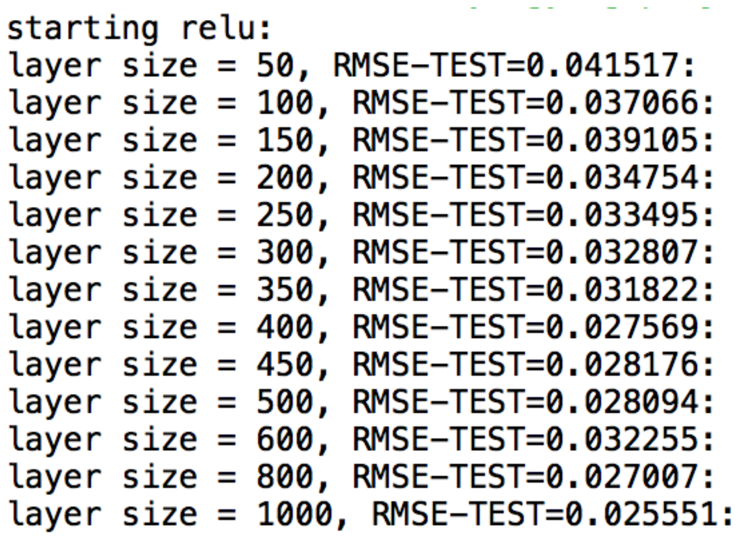

X[4] <= 16.5 mse = 0.011 samples = 11719 value = 0.061
True
False
X[3] <= 0.5 mse = 0.005 samples = 6687 value = 0.031
X[4] <= 25.5 mse = 0.015 samples = 5032 value = 0.101
X[1] <= 3.5 mse = 0.003 samples = 1147 value = 0.101
X[1] <= 1.5 mse = 0.005 samples = 5540 value = 0.017
X[2] <= 7.0 mse = 0.009 samples = 3556 value = 0.124
X[2] <= 11.0 mse = 0.026 samples = 1476 value = 0.045
X[1] <= 1.5 mse = 0.002 samples = 659 value = 0.082
X[2] <= 11.0 mse = 0.004 samples = 488 value = 0.127
X[4] <= 3.5 mse = 0.014 samples = 1571 value = 0.035
X[1] <= 4.5 mse = 0.001 samples = 3969 value = 0.009
X[4] <= 22.5 mse = 0.002 samples = 1202 value = 0.088
X[1] <= 3.5 mse = 0.011 samples = 2354 value = 0.141
X[1] <= 1.5 mse = 0.003 samples = 730 value = 0.024
X[1] <= 1.5 mse = 0.048 samples = 746 value = 0.066
mse = 0.002 samples = 324 value = 0.114
mse = 0.001 samples = 335 value = 0.052
mse = 0.0 samples = 257 value = 0.08
mse = 0.003 samples = 231 value = 0.175
mse = 0.083 samples = 208 value = 0.145
mse = 0.001 samples = 1363 value = 0.019
mse = 0.001 samples = 2387 value = 0.014
mse = 0.0 samples = 1582 value = 0.002
mse = 0.003 samples = 805 value = 0.103
mse = 0.001 samples = 397 value = 0.058
mse = 0.016 samples = 1370 value = 0.165
mse = 0.002 samples = 984 value = 0.108
mse = 0.005 samples = 215 value = 0.062
mse = 0.001 samples = 515 value = 0.009
mse = 0.153 samples = 175 value = 0.269
mse = 0.0 samples = 571 value = 0.004

Analysis:

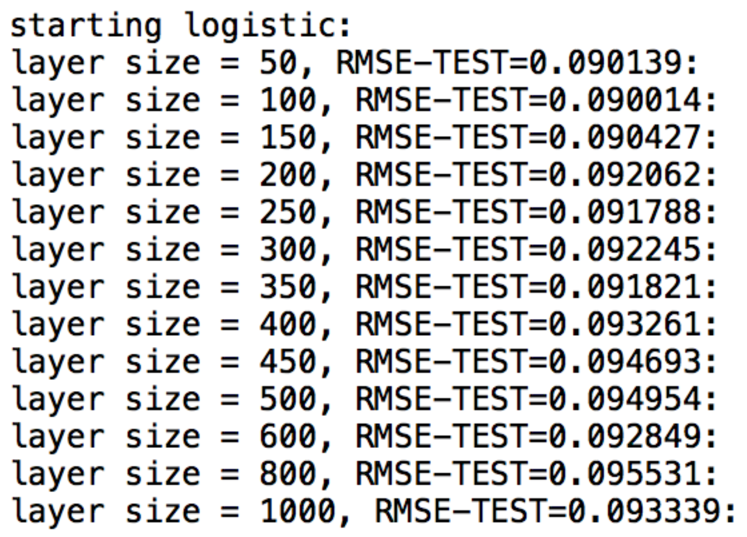
In this tree the root node is not same with the most important feature we got in previous work. This is because in random forest algorithm, it will try several times to get different trees and vote the most important feature from those result. So for a particular tree, its root node may not equal to our most important feature but in general, the most importance feature is the root node which appears most in all trees.

Problem 2(c)

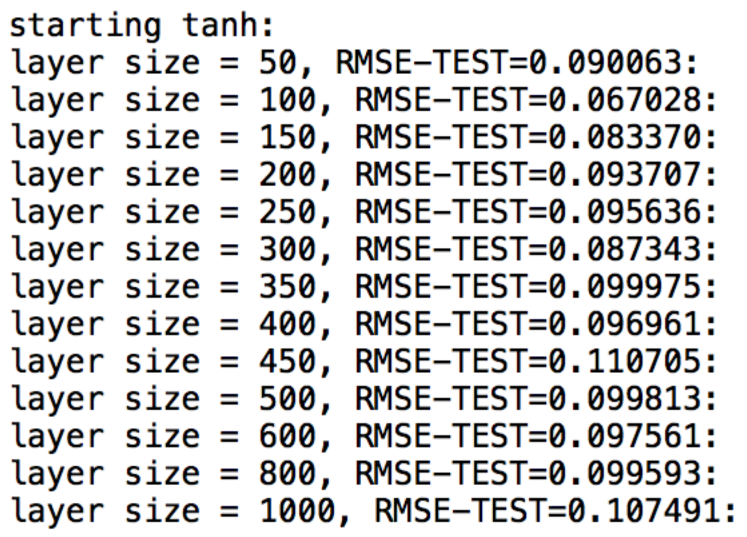




```
starting relu:
layer size = 50, RMSE-TEST=0.041517:
layer size = 100, RMSE-TEST=0.037066:
layer size = 150, RMSE-TEST=0.039105:
layer size = 200, RMSE-TEST=0.034754:
layer size = 250, RMSE-TEST=0.033495:
layer size = 300, RMSE-TEST=0.032807:
layer size = 350, RMSE-TEST=0.031822:
layer size = 400, RMSE-TEST=0.027569:
layer size = 450, RMSE-TEST=0.028176:
layer size = 500, RMSE-TEST=0.028094:
layer size = 600, RMSE-TEST=0.032255:
layer size = 800, RMSE-TEST=0.027007:
layer size = 1000, RMSE-TEST=0.025551:
```



```
starting logistic:
layer size = 50, RMSE-TEST=0.090139:
layer size = 100, RMSE-TEST=0.090014:
layer size = 150, RMSE-TEST=0.090427:
layer size = 200, RMSE-TEST=0.092062:
layer size = 250, RMSE-TEST=0.091788:
layer size = 300, RMSE-TEST=0.092245:
layer size = 350, RMSE-TEST=0.091821:
layer size = 400, RMSE-TEST=0.093261:
layer size = 450, RMSE-TEST=0.094693:
layer size = 500, RMSE-TEST=0.094954:
layer size = 600, RMSE-TEST=0.092849:
layer size = 800, RMSE-TEST=0.095531:
layer size = 1000, RMSE-TEST=0.093339:
```



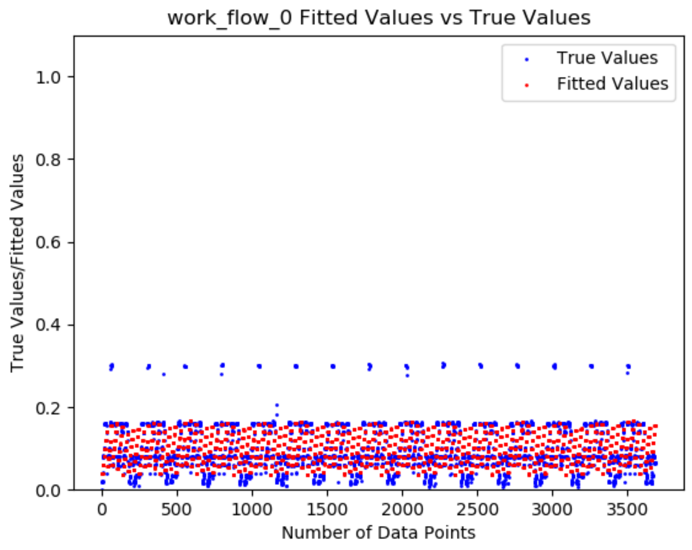
```
starting tanh:
layer size = 50, RMSE-TEST=0.090063:
layer size = 100, RMSE-TEST=0.067028:
layer size = 150, RMSE-TEST=0.083370:
layer size = 200, RMSE-TEST=0.093707:
layer size = 250, RMSE-TEST=0.095636:
layer size = 300, RMSE-TEST=0.087343:
layer size = 350, RMSE-TEST=0.099975:
layer size = 400, RMSE-TEST=0.096961:
layer size = 450, RMSE-TEST=0.110705:
layer size = 500, RMSE-TEST=0.099813:
layer size = 600, RMSE-TEST=0.097561:
layer size = 800, RMSE-TEST=0.099593:
layer size = 1000, RMSE-TEST=0.107491:
```

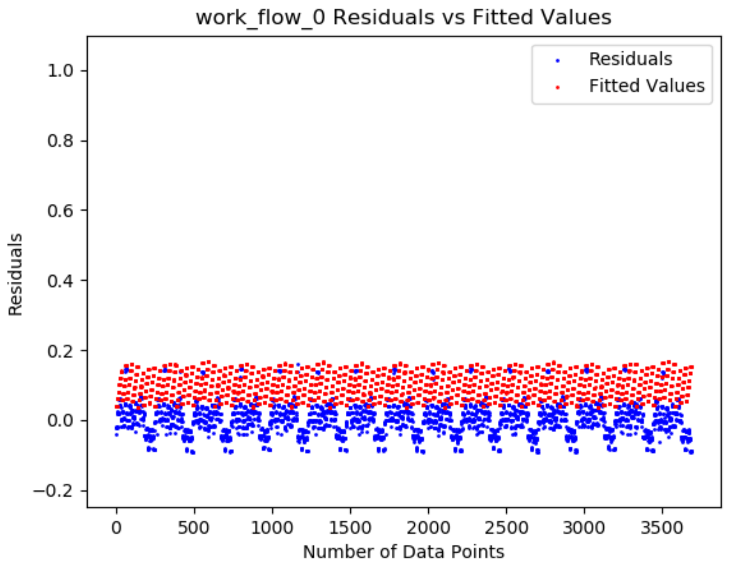
Analysis:

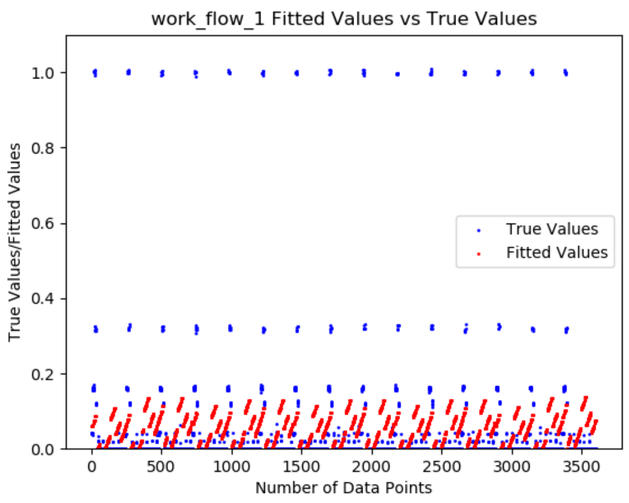
We use a neural network regression model (one hidden layer) with all features one-hot encoded. Parameters includes: Number of hidden units ranging from 50 to 1000 with step 50, and Activity Function (relu, logistic, tanh). As it was shown in the figure, the test RMSE with logistic and tanh activity function are close, they are around [0.08, 0.11]. The test RMSE with activity function Relu is lower comparing to others, they are around [0.03,0. 04], and it is decreasing with number of hidden units increasing. The best combination is using Relu activity function with number of hidden units of 400 with testing RMSE of 0.3025984

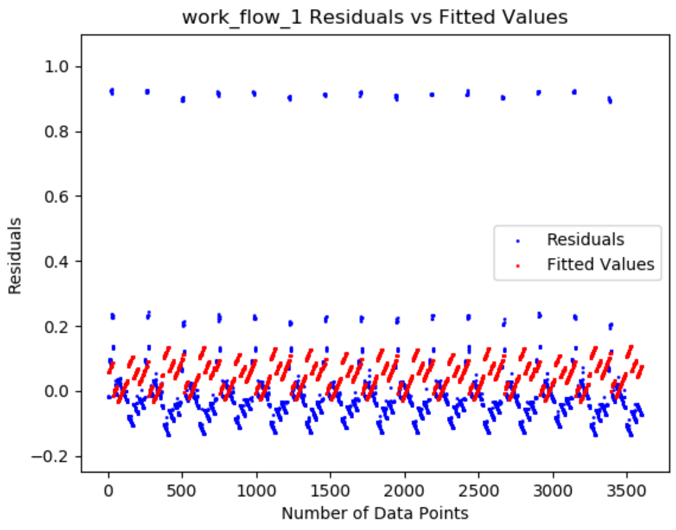
Problem 2(d)(i)

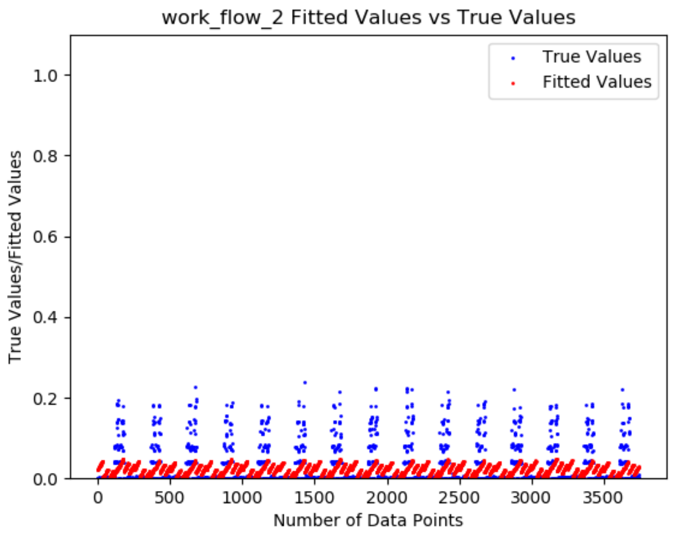
In this part, we try to predict each of the workflows separately. First, we use linear regression model.

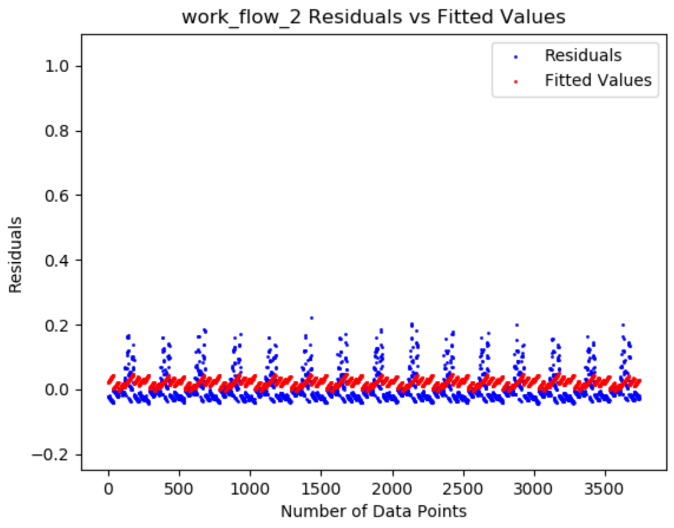




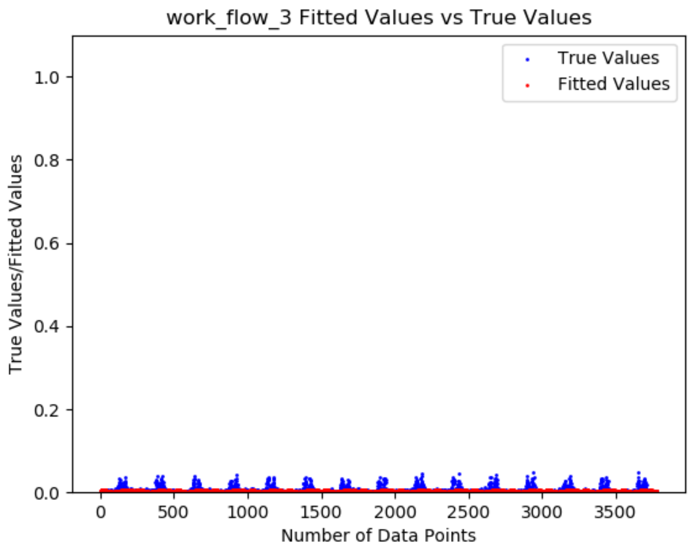

work_flow_1 Fitted Values vs True Values
True Values
Fitted Values
True Values/Fitted Values
Number of Data Points


work_flow_1 Residuals vs Fitted Values
Residuals
Fitted Values
Residuals
Number of Data Points

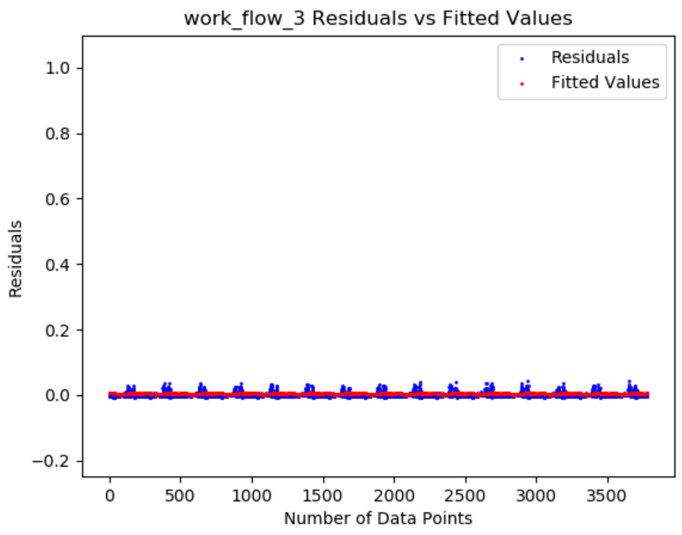

work_flow_2 Fitted Values vs True Values
True Values
Fitted Values
True Values/Fitted Values
Number of Data Points



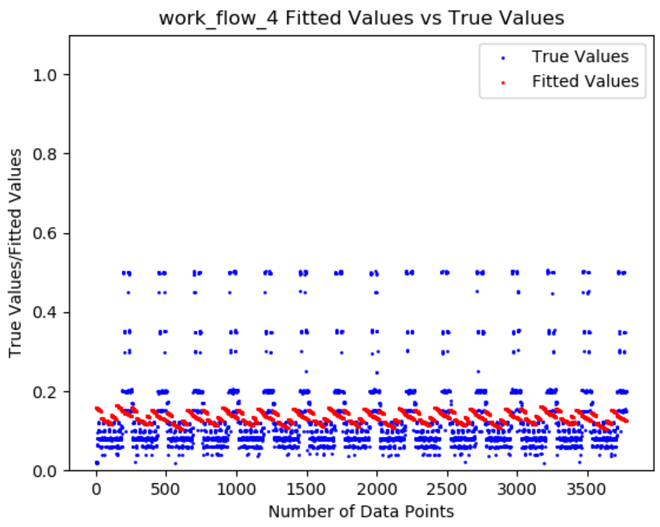
work_flow_2 Residuals vs Fitted Values
Residuals
Fitted Values
Residuals
Number of Data Points

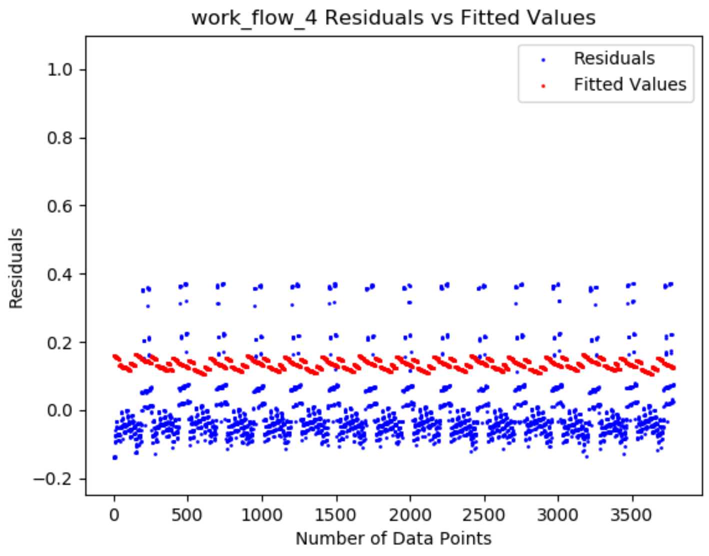


work_flow_3 Fitted Values vs True Values
True Values
Fitted Values
True Values/Fitted Values
Number of Data Points



work_flow_3 Residuals vs Fitted Values
Residuals
Fitted Values
Residuals
Number of Data Points



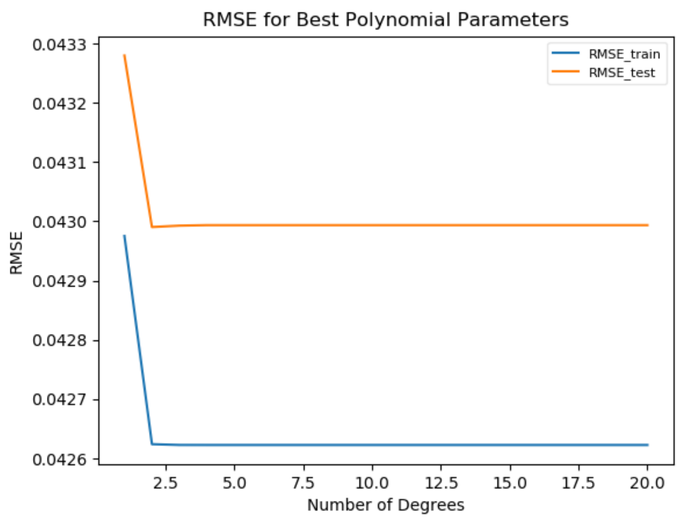


Analysis:

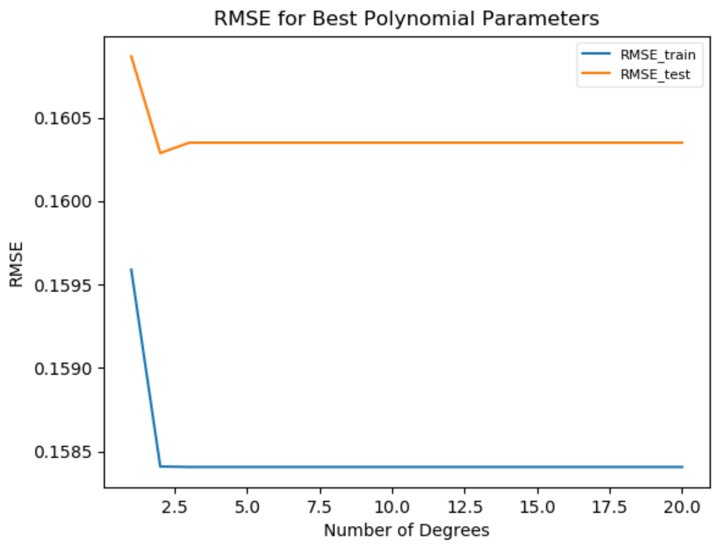
Based on the dataset provided in the question, we consider to perform the 10-fold cross validated linear regression on each workflow. To do this, we need firstly separate the original dataset into 5 different datasets, each of which contains all the same attributes for each work flow. From each part we get above, we can see that the fit is improved. Clearly, most of our fitted values and true values are overlapped which means our data fit well. The result improved when comparing to the linear regression of the whole dataset, because each workflow has its own pattern and we make better predictions by separating them.

Problem 2(d)(ii)

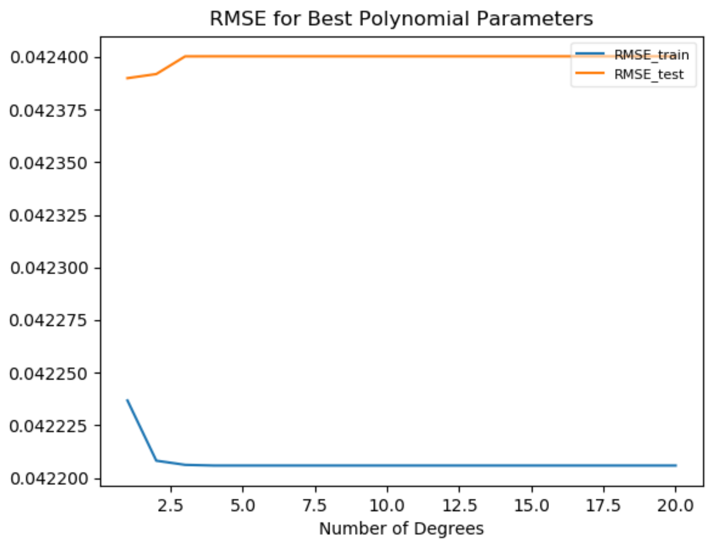
Workflow 0



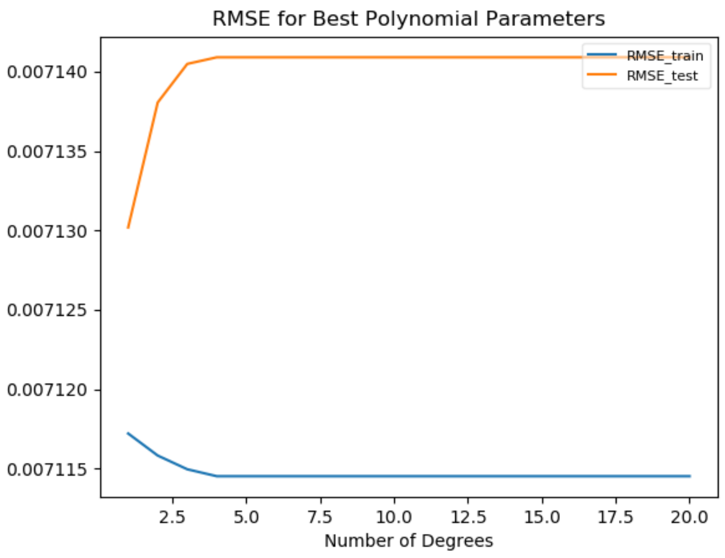
Workflow 1



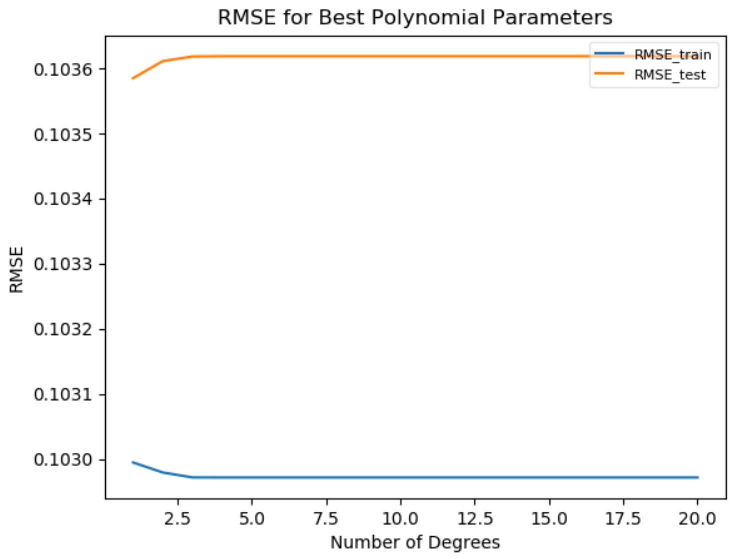
Workflow 2



Workflow 3

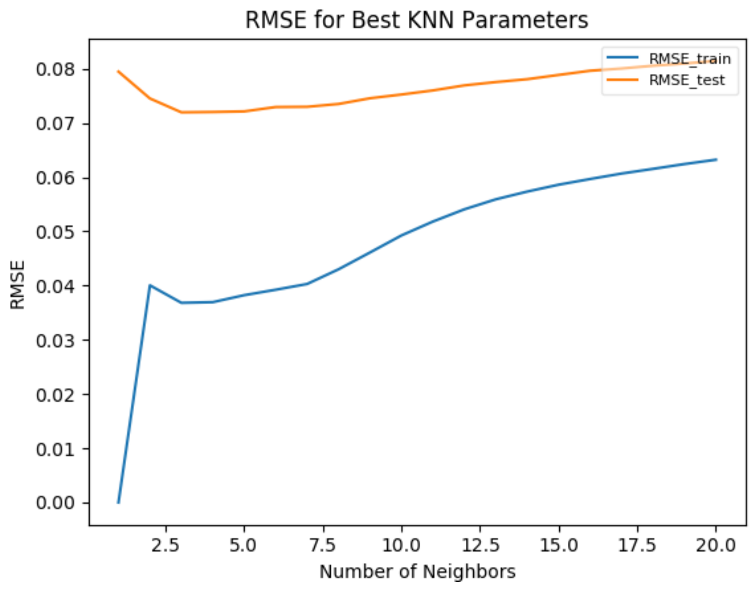


Workflow 4



We can see that as the degree of the polynomial 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. In general, the threshold on the degree of the fitted polynomial is about 3. 3-degree 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.

Problem 2(e)



It is obvious from the figure that the best parameter is around 3. It is the case when both the training RMSE and the testing RMSE have the lowest point (except for the initial condition). The reason that this happens is that although high number of neighbors gather more data points when calculating the estimator, some neighbors are too far away and not significant to the data point that we interested in. On the other hand, if the number of neighbors is too low, then we do not have enough information to make good predictions.

Problem 3

In this project, we used many regression models, such as Linear Regression, Polynomial Regression, Random Forest Regression, Neural Network, and K-Nearest Neighbor Regression. All these regression models behave differently when facing various kinds of dataset. We will make a conclusion on each model.

First, the Linear Regression is a simple linear function and tries to minimize the sum of squared residuals when making predictions. The max degree of such model is 1, and we can combine multiple linear regressions with different estimator from various features into a single one, but each estimator is still linear. This type of regression works great if the dataset is linear. However, such situation is very rare in reality, most of the dataset are non-linear. When doing linear regression, it is very important to one-hot-encode the categorical data instead of using the label encoder. This is because standard scaler adds an unnecessary information to the dataset. For example, if we label encode the week days with 1 represents Monday and 7 represents Sunday, then this will make the linear model think that Sunday is greater than Monday because 7 is greater than 1. However, in reality, this is not necessarily the case. Monday and Sunday are equal except their labels are different. From this project, we think that linear regression is not a very good model because the dataset we have is not linear, it also handles the categorical features poorly. It makes bad predictions when comparing with other models.

Second, the Polynomial Regression is very similar to the Linear Regression except that each term can have a degree higher than 1. It is also possible for each term has different degrees based on the features, and we can combine several polynomials into a single polynomial model. Similar to linear regression, we need to handle the categorical data carefully. However, one of the problem of the polynomial regression is that it is very easy to overfit the training data. A polynomial degree of 100 will usually fits the training dataset better than a degree of 3, but such degree will perform the poor prediction. Therefore, it is important to find the best degree with both training and testing RMSE low. Since the polynomial model is number based regression, it makes poor decision on the categorical features.

Third, the Random Forest Regression is a kind of ensemble learning that combines the results of many randomly selected decision trees. The decision tree is made based on the feature that minimize a variance in the regression model. Such split will continue to happen until a threshold is reached. From our experimental results, this type of regression performs better than both the linear regression and polynomial regression. It can handle the categorical values very well because it is not a numerical based regression. The decisions and regression results are based on information gain or impurity, so the categorical data will not affect the random forest results. From the results in random forest regression, it is a good predictor in general.

Fourth, the Neural Network is a completely different model in contrast of the previous three. This model is based on deep learning theories by trying to mimic the neural network in human brains. The interpretability of neural network is quite low because the whole model is somewhat a black box. We only split the dataset and pass it to the network, then the network will train itself based on the back propagation and activity functions. One of the biggest advantage of using neural network is its accuracy and works great for sparse features. We can specify the number of layers and hidden units to best fit our dataset. From our experimental results, the neural network with relu layer out performs all other models. Therefore, it is the best model for this project.

Finally, the K-Nearest Neighbor Regression calculates the k-nearest data points and make predictions based on the average and voting of these points. This model works great if the feature dimension is low and the matrix is not sparse. If the dimension is too high or the matrix is sparse, the Euclidean distance in high dimension space will converge to 1, and it is very difficult to identity the k-nearest points. From our result, we can see that it makes good prediction when the number of neighbor is 3, and as we increase the number of neighbors, the RMSE increases. For handling categorical features, it is quite effective because the model is not numerical based like linear regression or polynomial regression, but the results from the neural network seems a little bit better than K-Nearest Neighbor.