ECE 219
Large Scale Data Mining: Models and Algorithm

Project 4
Regression Analysis

Haitao Wang (UID: 504294402)
Weiqian Xu (UID: 404297854)
Xinyi Gu (UID: 205034975)
Yutan Gu(UID: 005034976)

1. Data Interpretation

1.1 Twenty-day period

Plot the backup sizes against day numbers. Different workflow is marked with different color. The plot is demonstrated in Figure 1.

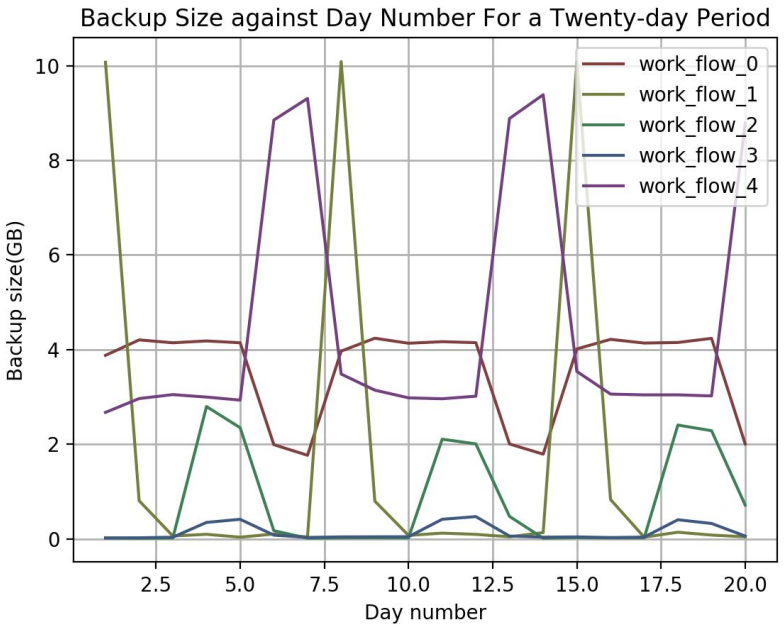

Figure 1: Backup size against day number for a twenty-day period.

1.2 105-day period

Plot the backup sizes against day numbers. Different workflow is marked with different color. The plot is demonstrated in Figure 2.

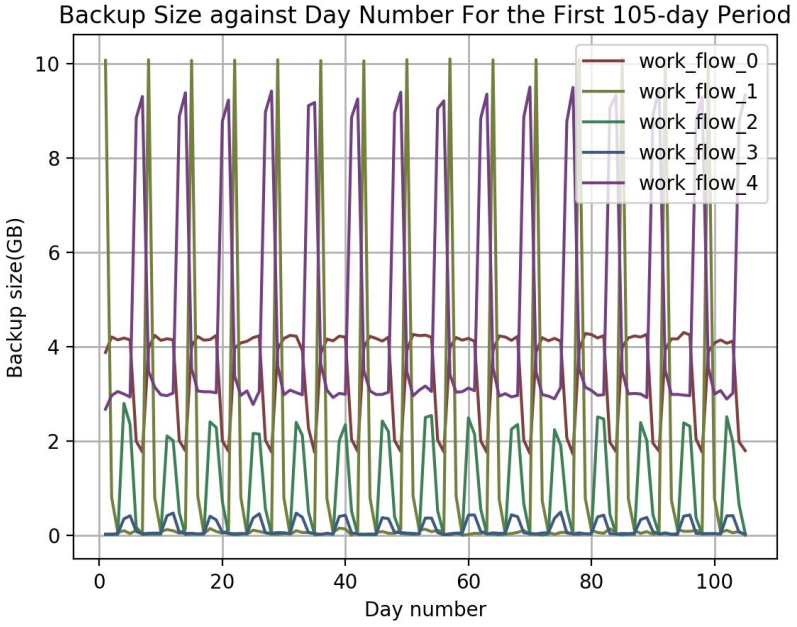

Figure 2: Backup size against day number for a 105-day period.

There are different repeating patterns for different workflows. But, they have the same period of about 7 days, equivalently a week. And, the repeating patterns are similar: a dramatic fall/rise followed by a dramatic rise/fall. In other words, all workflow have a repeating pattern in backup size of period of 7 days.

2. Prediction

2.1 Linear regression

2.1.1 Basic linear regression model

First convert each categorical feature into one-dimensional numerical values using scalar encoding (e.g. Monday to Sunday can be mapped to 1-7). In detail, we convert 'Monday' to 'Sunday' into 1-7, 'work_flow_0' to 'work_flow_4' into 0-4, 'File_1' to 'File_29' into 0-29. Then directly use them to fit a basic linear regression model. The training and testing RMSE are listed in Table 1. The plot of fitted values against true values and the plot of residuals against fitted values are shown in Figure 3.

# fold	Training RMSE	Testing RMSE
1	0.10324	0.10672
2	0.10297	0.10018
3	0.10323	0.10685
4	0.10395	0.10037
5	0.10320	0.10712
6	0.10394	0.10045
7	0.10320	0.10705
8	0.10394	0.10047
9	0.10320	0.10707
10	0.10340	0.99995

Table 1: Training and testing RMSE of each fold in cross-validation

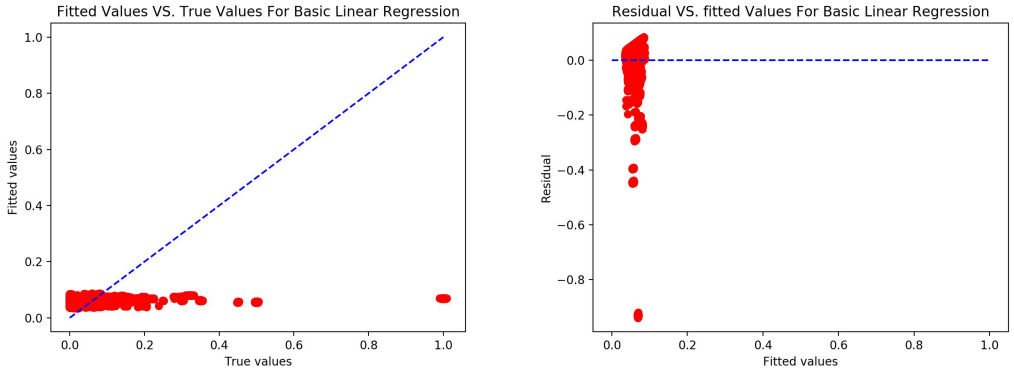

Figure 3: Fitted values versus true values; residuals versus fitted values.

2.1.2 Standardization

Standardize all these numerical features, then fit and test the model. The training and testing RMSE are listed in Table 2. The plot of fitted values against true values and the

plot of residuals against fitted values are shown in Figure 4. Because we apply the same 10-fold cross-validation without shuffling, the results are exactly the same as in 2.1.1. This means that standardization doesn't improving the performance of fitting the data into linear regression.

# fold	Training RMSE	Testing RMSE
1	0.10324	0.10672
2	0.10297	0.10018
3	0.10323	0.10685
4	0.10395	0.10037
5	0.10320	0.10712
6	0.10394	0.10045
7	0.10320	0.10705
8	0.10394	0.10047
9	0.10320	0.10707
10	0.10340	0.09995

Table 2: Training and testing RMSE of each fold in cross-validation

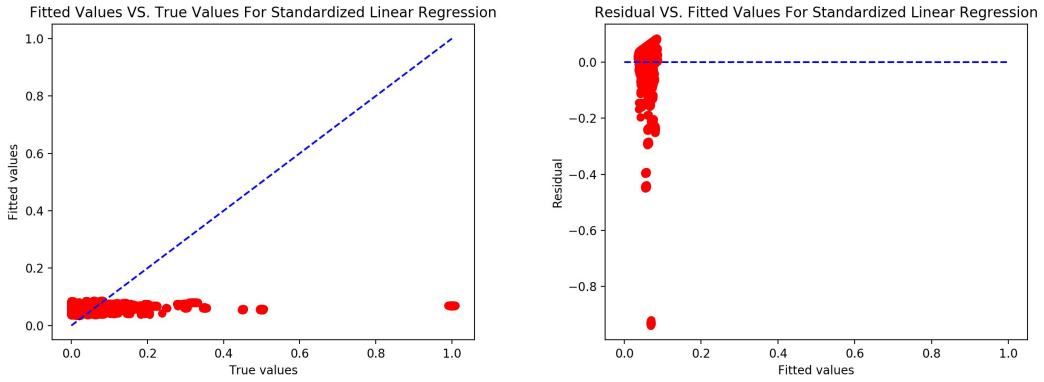

Figure 4: Fitted values versus true values; residuals versus fitted values.

2.1.3 Feature selection

Use f regression and mutual information regression measure to select three most important variables respectively. Report the three most important variables you find. Use those three most important variables to train a new linear model. The f-scores and mutual information scores are reported in Table 3. Taking into account all factors, we decide to choose the best combination of feature 2,3,5 as the best choice. The resulting plots of fitted values against true values and of residuals against fitted values are shown in Figure 5. The training and testing RMSE for each fold is shown in Table 4. Using the best three features instead of the whole set of features doesn't improve the performance at all.

Feature	1	2	3	4	5
F-score	0.00845	38.816	150.74	26.139	25.320
MI	0	0.23010	0.23543	0.27735	0.42778

Table 3: F-scores and mutual information scores of each feature

# fold	Training RMSE	Testing RMSE
1	0.10325	0.10670
2	0.10397	0.10018
3	0.10323	0.10684
4	0.10395	0.10038
5	0.10320	0.10711
6	0.10394	0.10045
7	0.10321	0.10704
8	0.10394	0.10047
9	0.10321	0.10705
10	0.10399	0.09995

Table 4: Training and testing RMSE of each fold in cross-validation

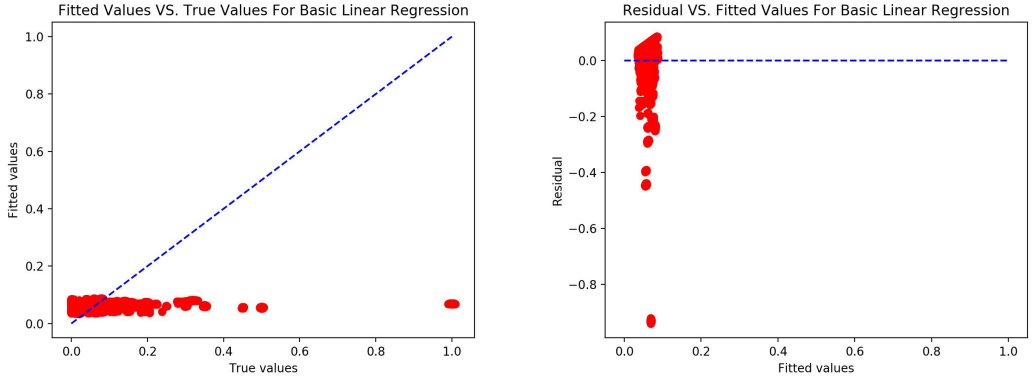

Figure 5: Fitted values versus true values; residuals versus fitted values.

2.1.4 Feature Encoding

There are 32 combination of encoding the five categorical features, since each sort of feature is possibly encoded as one-hot vector. The plot of average training RMSE and test RMSE of each combination is shown in Figure 5. The best feature representation is the combination '01101', where '0' indicates scalar encoding and '1' indicated one-hot encoding. Therefore, '01101' indicates that the first and forth features are represented by scalar encoding, and the remaining features are represented by one-hot encoding. In other words, 'day of week', 'hour of day' and 'file name' are encoded in the one-hot representation, while 'week number' and 'workflow ID' are encoded in the scalar representation. One-hot encoding makes more sense from machine learning perspective. We can't say that 'Monday' is smaller or greater in magnitude than 'Sunday'. The categorical features are just ordinal values, not nominal. One-hot encoding is to generate one boolean row for each sample. Only one of elements in the row could take on the

value 1 for each sample. Then, there will not be comparison in the magnitudes of samples for a specific feature. Also, each category in the feature is the same length away from each other, this making more sense than different distances in between each pair of categories in scalar encoding. The reason why this combination performs the best is that 'day of week' and 'hour of day' are more correlated with backup size, according to 1.1. 'file name' is also more related to backup size, in comparison with 'week number' and 'workflow ID'. The backup size alters as a repeating pattern of a period of 7 days, namely a week. Therefore, 'week number' is not related at all. 'workflow ID' has higher f-score but lower MI score than 'file name'. It's hard to judge whether it is a more significant or not. Therefore, 'day of week', 'hour of day' and 'file name' are temporarily picked as the three most significant features. Only representing the significant features in more correct way(one-hot encoding) can improve the performance of the model.

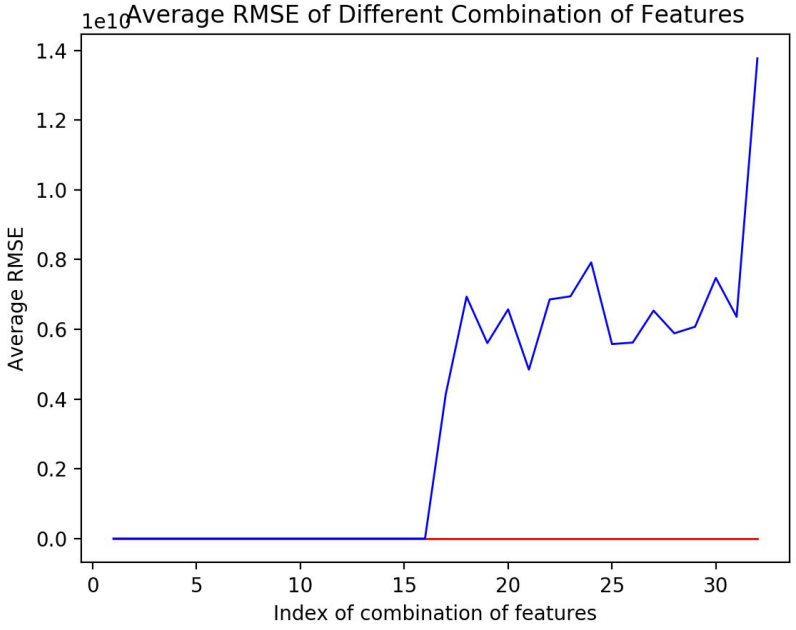

Figure 5: Average training and testing RMSE over each combination

2.1.5 Controlling ill-conditioning and over-fitting

As shown in Figure 5, there are obvious increases in the test RMSE in some combination, compared to the train RMSE. From the 17th combination to the 32th combination, the feature 'week number' is encoded in one-hot format. The difference between the training and testing RMSE happen to be relatively large since the 17th combination. We can interpret that if the wrong feature is represented in the correct way in the model, it will hurt the overall performance of the model to a significant extent. In other way, it proves that the first type of feature 'week number' is a bad feature.

The reason of the big gap between training and testing RMSE is that the model is likely to be overfitting, since the training RMSE of the 'bad' combinations are still kept as low as those of the 'good' combinations. Then, we need to use regularization to correct the model.

For Lasso model, the best combination is '01110', as shown in Figure 6. After tuning the parameter of the Lasso model, we set 'alpha' as 0.0001. The plots of the optimum model are demonstrated in Figure 7. The training and testing RMSE of the chosen Lasso model are listed in Table 5.

# fold	Training RMSE	Testing RMSE
1	0.08821	0.09385
2	0.08933	0.08371
3	0.08825	0.09355
4	0.08931	0.08392
5	0.08825	0.09361
6	0.08928	0.08426
7	0.08821	0.09389
8	0.08932	0.08390
9	0.08824	0.09363
10	0.08933	0.08373

Table 5: Training and testing RMSE for each fold in cross-validation

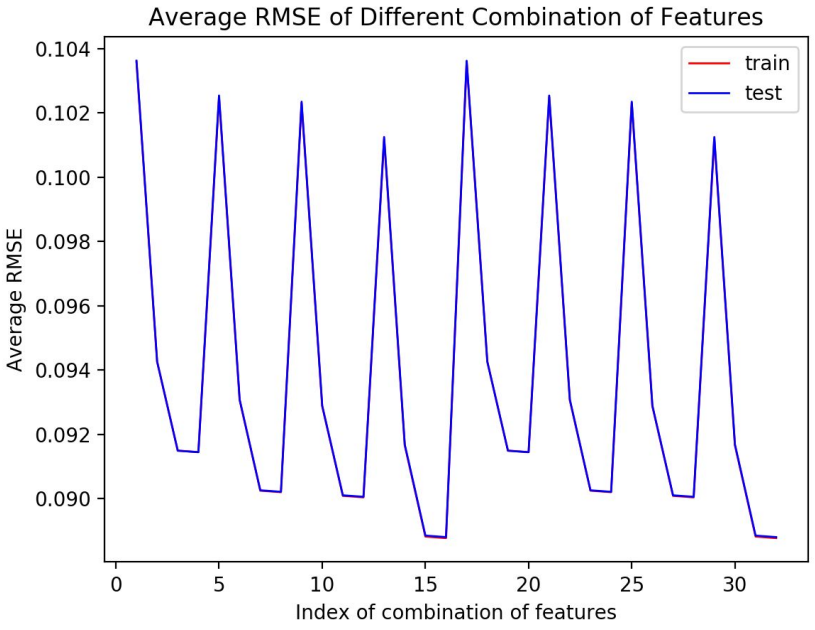

Figure 6: Average RMSE of different combinations of feature encoding.

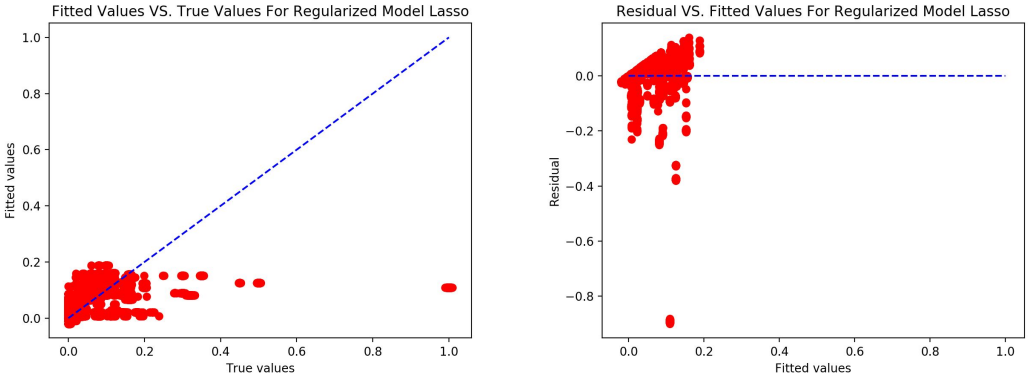

Figure 7: Fitted values versus true values; residuals versus fitted values.

For Ridge model, the best combination is '01110', as shown in Figure 8. After tuning the parameter of the Ridge model, we set 'alpha' as 0.1. The plots of the optimum model are demonstrated in Figure 9. The training and testing RMSE of the chosen Ridge model are listed in Table 6.

# fold	Training RMSE	Testing RMSE
1	0.08777	0.099348
2	0.08890	0.08326
3	0.08781	0.09312
4	0.08889	0.08344
5	0.08780	0.09316
6	0.08884	0.08385
7	0.08777	0.09345
8	0.08888	0.08344
9	0.08780	0.09318
10	0.08933	0.08330

Table 6: Training and testing RMSE for each fold in cross-validation

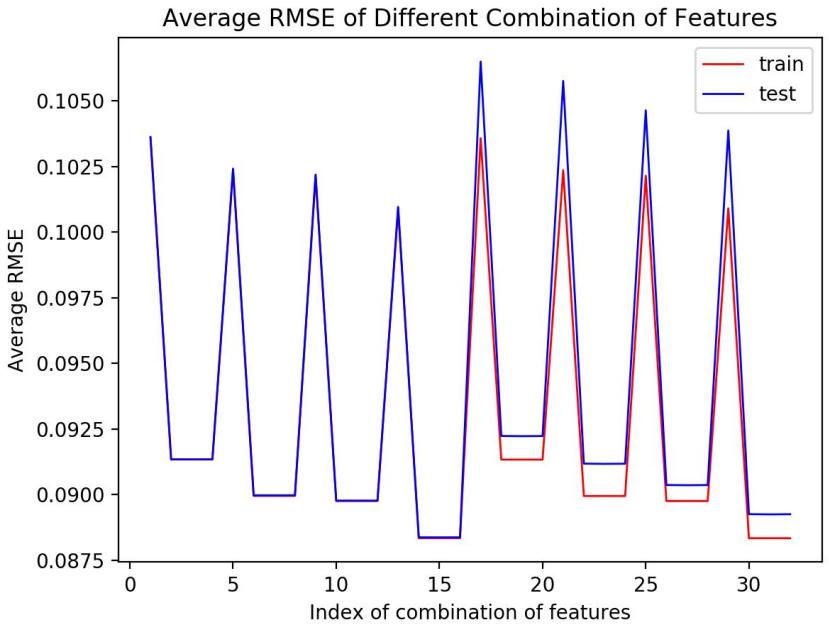

Figure 8: Average RMSE of different combinations of feature encoding.

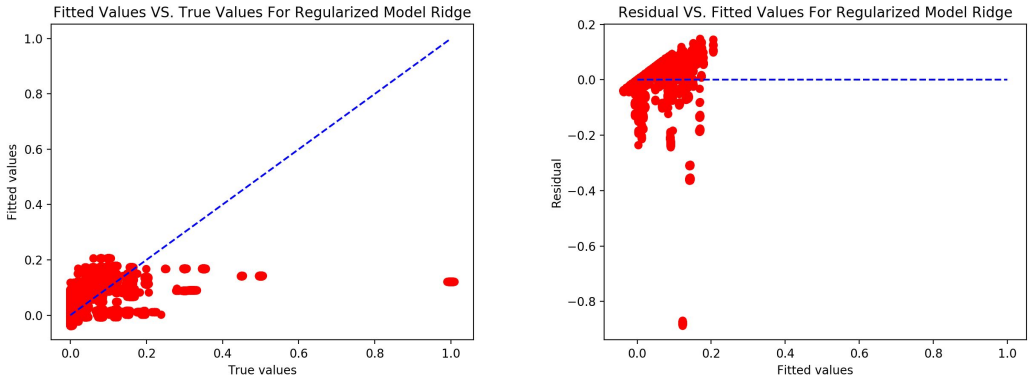

Figure 9: Fitted values versus true values; residuals versus fitted values.

In comparison with the un-regularized model, there is significant improvement in the regularized model. The training and testing RMSE drop by about 20%. The Figure 7 and 9 shows better results, compared to Figure 4: the fitted data points in the regularized model are distributed in a range as twice wide as that in the case of the un-regularized model. Thus, the coefficients for the regularized model make the data points fit into the trained model better, than those for the un-regularized model.

2.2 Random Forest Regression Model

In this part, we will evaluate the random forest regression model.

2.2.1 Initial Test

We use RandomForestRegressor function in the sklearn package to implement this model. One thing that is worth to mention is that although the random forest regression model itself can handle categorical variables without using encoding method, we find that the sklearn package can only handle encoded data. Thus, we first apply scalar encoding on the original data. We manually assign values 1 to 7 corresponding to Monday through Friday, value 0 to 4 corresponding to workflow_0 to workflow_4, etc.

As the random forest regression model uses bagging method, we introduce a new estimation of this model which is the out of bag error. We also evaluate the 10-fold cross validation result as what we did in the last part.

For this first subsection, we evaluate the model using initial hyperparameters, which is:

- *Number of trees: 20*
- *Depth of each tree: 4*
- *Maximum number of features: 5*

The results are shown as follows:

oob_error: 0.341496260413

# fold	Training RMSE	Testing RMSE
1	0.06080392	0.06204569
2	0.06100628	0.0601497
3	0.05945914	0.07293733
4	0.06160476	0.05359858
5	0.06121703	0.05827642
6	0.06111271	0.05260626
7	0.06015277	0.06604414
8	0.05968983	0.06664805
9	0.06107385	0.05970672
10	0.06170821	0.05375136

Table 7: Training and testing RMSE for each fold in cross-validation

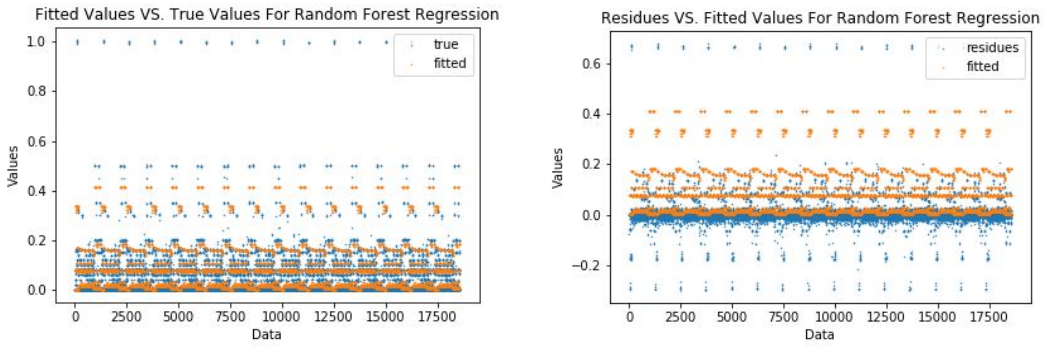

Figure 10: Fitted values versus true values; residuals versus fitted values for random forest regression

These results could be seen as the baseline when we tune the parameters in the following sections. As we can see in the results, the average training RMSE and testing RMSE of each fold is around 0.6, which is not low enough. In the scatter plot, the fitted result is not good especially when the true value is larger than 0.2.

2.2.2 Tune Number of Trees and Max Number of Features

In this part, we will tune two hyperparameters which are number of trees and max number of features. We will sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5.

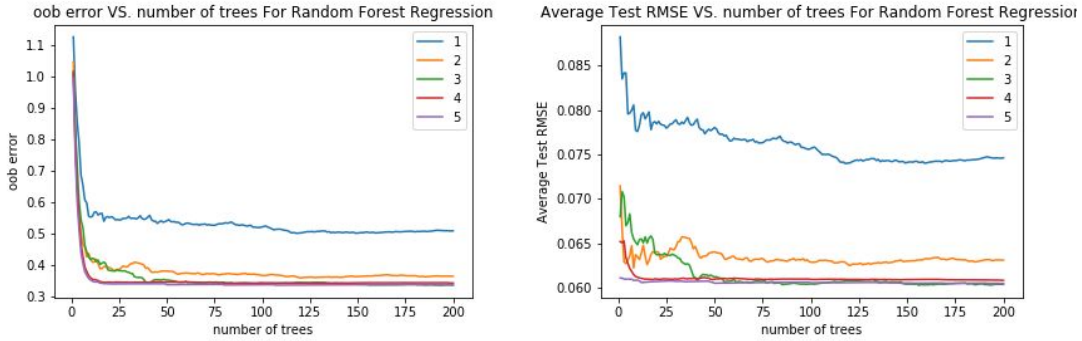

Figure 11a: OOB errors versus number of trees; Average Test RMSE versus number of trees(legend stands for max number of features)

best number of trees for 1 = 118
the best oob error is 0.501312711555
best number of trees for 2 = 120
the best oob error is 0.36045657773
best number of trees for 3 = 193
the best oob error is 0.336963724688
best number of trees for 4 = 81
the best oob error is 0.343292899263
best number of trees for 5 = 107
the best oob error is 0.337602353109

best number of trees for 1 = 118
the best average test rmse is 0.0739607135277
best number of trees for 2 = 8
the best average test rmse is 0.0621930219729
best number of trees for 3 = 156
the best average test rmse is 0.0602364588954
best number of trees for 4 = 200
the best average test rmse is 0.0608003903707
best number of trees for 5 = 200
the best average test rmse is 0.060440264212

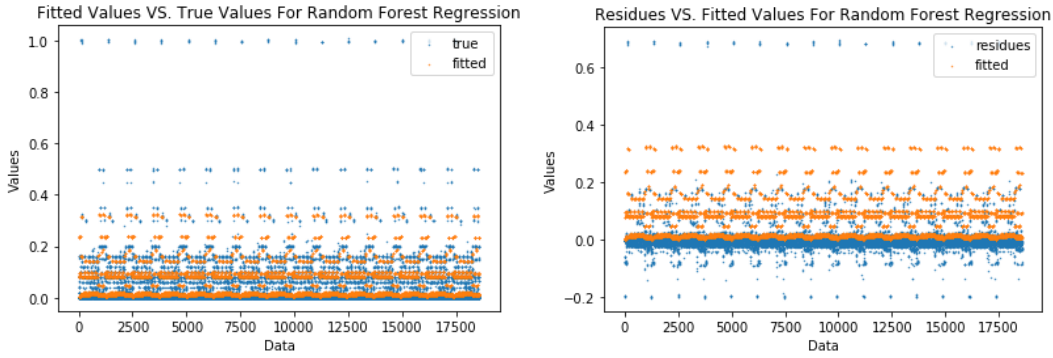

Figure 11b: Fitted values versus true values; residuals versus fitted values for random forest regression after tuning number of trees and max number of features

As for the number of trees, we can see that for each of the experiment, the model performs relatively well after n>50. After that, the curves become stable which means there is only slightly improvement or even no improvement if we continuously increase the number of trees. At the meantime, increasing the number of trees may also result in the increasing of computation time.

As for the max number of features, intuitively, increasing this hyperparameter will let each tree in the forest to take more account more kind of features into consideration, which should give a better performance. The plot we generate proves this intuitive sense. We can see that as we increase the parameter, both of the errors will decrease. We also notice that there is only slight difference between parameter 3, 4 and 5. This indicates that we do not need to take all the features into account. Some features may be highly irrelevant to the prediction result. Adding them into the model will increase the computation time but will not benefit the performance. For instance, the "Week #" feature is not closely related to the predicted output for this dataset .

From these above, we choose max number of features=3, number of tree=156 as the best hyperparameter. We then get the scatter plot figure 11b using these tuned parameters, without changing the max depth(remains 4). We can see that the fitted result is still not very good by tuning only these two hyperparameters.

2.2.3 Tune Max Depth

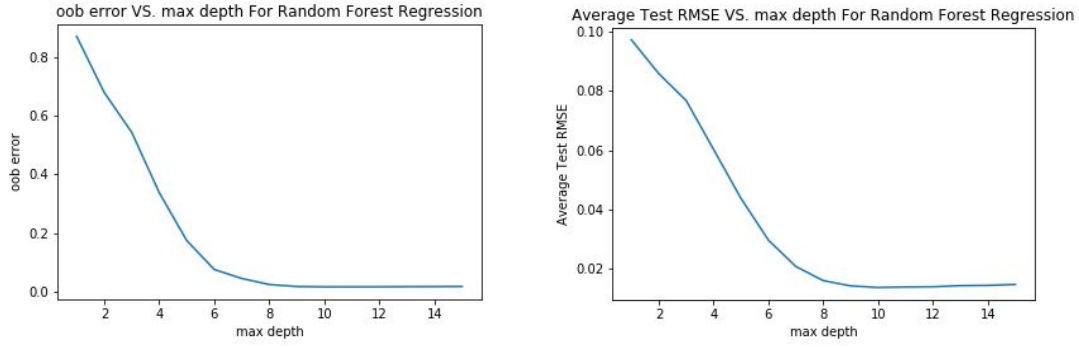

Figure 12a: OOB errors versus max_depth; Average Test RMSE versus max_depth

best max depth = 10
the best oob error is 0.0157867432076

best max depth for = 10
the best average test rmse is 0.0135270653852

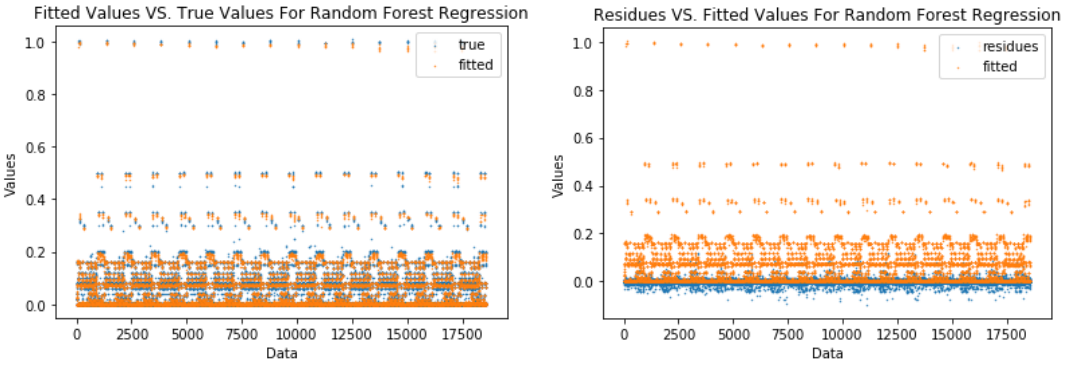

Figure 12b: Fitted values versus true values; residuals versus fitted values for random forest regression after tuning max depth

Similarly as the above section, we then tune the max depth parameter. We sweep the max depth from 1 to 15. Both of the best oob error and test rmse appears at max depth=10. As we increase the max depth value, the performance becomes better and better. After max depth>10, the oob error becomes relatively stable while the test RMSE becomes even slightly worse and worse. We also notice that tuning this parameter will significantly improve the performance than the initial test we discussed above. It seems that the model is still underfitting the data with max depth<10, while it may overfitting the data with max depth>10. Thus we will choose max depth=10 as the best parameter to achieve the best performance. Using this best parameter, from the scatter plot we can see that the fitted value is much more closer to the true value. And the residues are closely distributed around zero. The performance of this model improved significantly after tuning the max depth.

2.2.4 Feature Importance

From the previous sections, we get our best parameters:

- *Number of trees: 156*
- *Depth of each tree: 10*
- *Maximum number of features: 3*

The features are:

features = ['Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name']

Their relevant feature importances are:

[0.00447815 0.2884866 0.33478316 0.17474622 0.19750587]

The most important feature here is Backup Start Time - Hour of Day.

The order is:

Backup Start Time - Hour of Day > Day of Week > File Name > Work-Flow-ID > Week #

The feature importance of 'Week #' is greatly smaller than the other 4 features, which indicates that it is almost irrelevant to the prediction result.

2.2.5 Visualize Decision Tree

Finally, we use the export_graphviz function to visualize one of the decision tree in the forest. Notice that if we use our best max depth=10 to visualize the tree, it will be to large for the figure. Thus, we set max depth to 4 to see the visualization result. After, we

set max depth to 4, the most important feature becomes *Day of Week*. Their relevant feature importances are:

[3.18892323e-04 3.27476258e-01 1.16147139e-01 2.61900735e-01 2.94156977e-01]

One of the decision tree is shown in figure 13. In this decision tree, the root node feature is *File name*, which is not the most important feature we get - Day of Week. This phenomenon is actually reasonable since we get this most important feature from the average of the whole forest, instead of focusing on only one tree. That is to say, this most important feature we get from the whole forest may not be the same most important feature for one specific tree in this forest. Thus, it is definitely reasonable if we find that the root feature of one of the trees is not the most important feature.

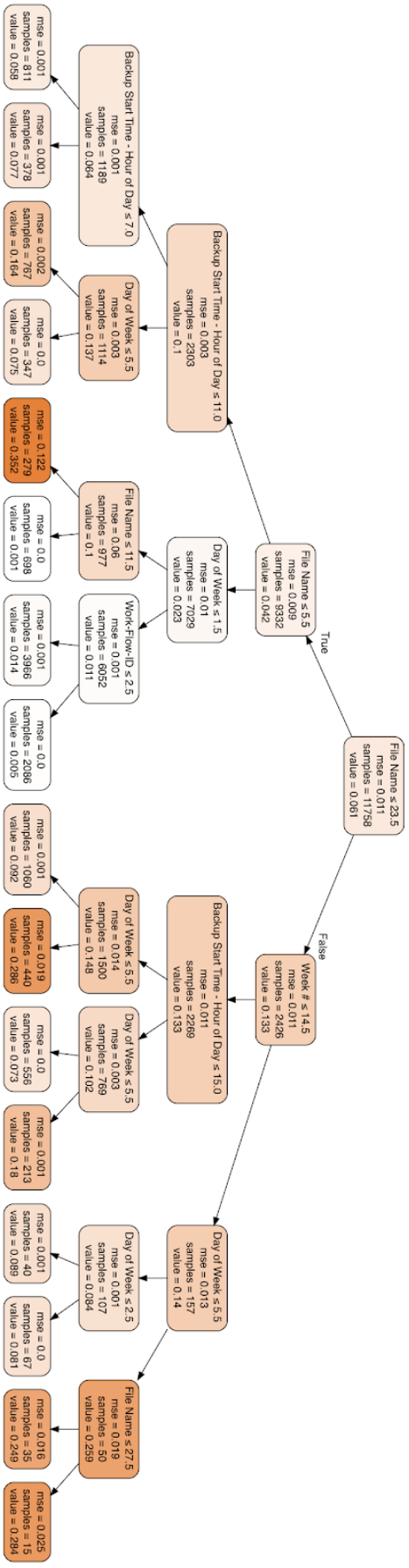

Figure 13: Visualization of one decision tree

2.3 Neural Network Regression Model

In this part, we are going to use a neural network regression model with all input data one-hot encoded. We are then going to sweep the number of hidden units using only one hidden layer with the choice of one of the activity functions including relu, logistic and tanh. We are going to use the *MLPRegressor* from the *sklearn.neural_network* library.

2.3.1 Relu Activity Function

We first use the relu activity function to train and test the neural network using the 10-fold cross validation. We sweep the number of hidden neurons from 1 to 200 and each time we perform the 10-fold cross validation and calculate the mean test RMSE. The result is shown in the graph below. As we can discover, the RMSE begins at 0.12 and decreases in a steep slope at first but slow down as the number of hidden neurons increases. When the number of neurons reaches 100, the test RMSE start decreasing with a very slow speed. The lowest RMSE is reached when the number of hidden neurons is 199 with a test rmse value as low as 0.01813 (The graph looks not precise enough).

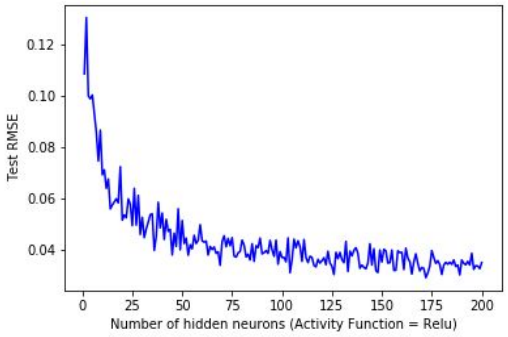

Figure 14: Test RMSE vs number of hidden neurons using relu activity function

We then plotted the graphs of true values vs. fitted values and true values vs. residues as shown below. One can discover that the red area on the true values vs. fitted values is surrounding the blue dotted line that represents equal fitted values and true values. We can conclude that the prediction is very good in additional to the low test RMSE value. Similarly, we can discover that for the true values vs. residuals graph, the points are all surrounding the horizontal line. We can make the conclusion that using neural network regressor with 199 hidden neurons and relu activity function can generate a fairly good result.

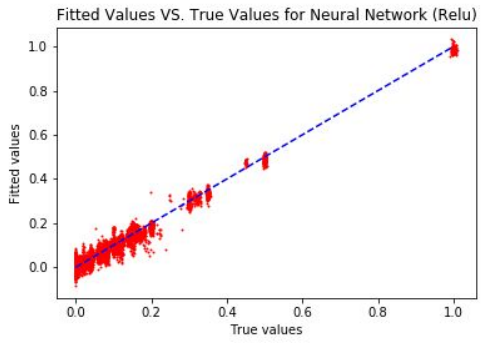

Figure 15: True values vs. fitted values

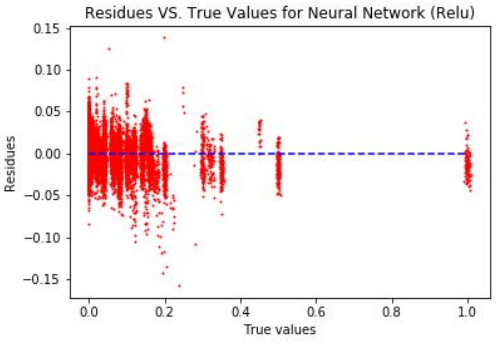

Figure 16: True values vs. residuals

2.3.2 Logistic Activity Function

The result using logistic activity function is presented below. As one can discover, the Test RMSE has a sharp decrease at first and then slightly goes up with steep fluctuations. The best number of neurons we got is 13 with the lowest RMSE value to be 0.08835. By analyzing the true values vs. fitted values graph and true values vs. residuals graph, we can discover that for all true values higher than 0.2, our neural network is not able to provide correct prediction compared to the one using the relu activity function.

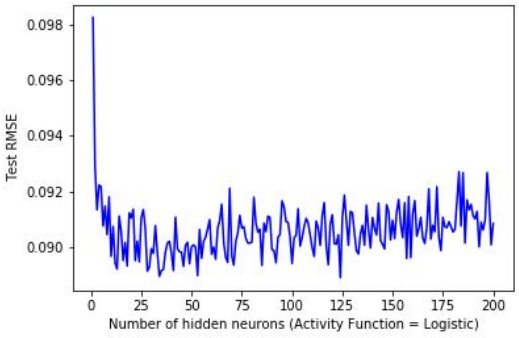

Figure 17: Test RMSE vs number of hidden neurons using logistic activity function

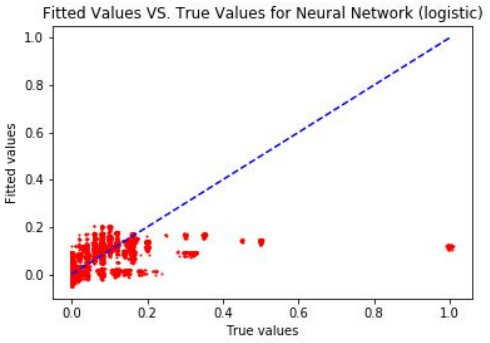

Figure 18: True values vs. fitted values

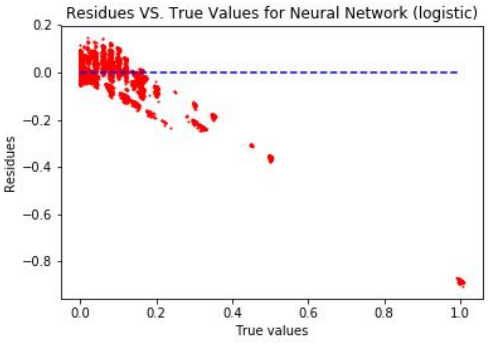

Figure 19: True values vs. residuals

2.3.3 Tanh Activity Function

We then switch to use the tanh activity function and the results are presented below. As we can see the Test RMSE does not decrease as smooth as using the relu activity function with strong fluctuations. The best number of neurons to use is 111 and the associated RMSE value is as low as 0.06224. Again, by inspecting the true values vs. fitted values graph and true values vs. residuals graph, we can discover that when the true values are high, the neural network is not able to predict correctly.

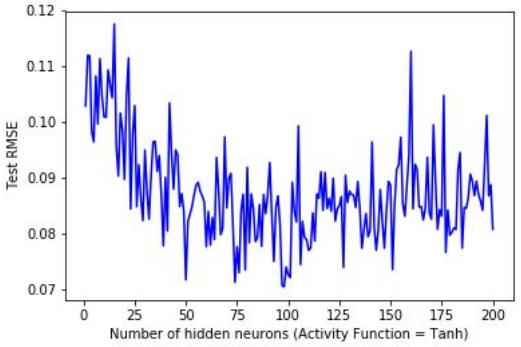

Figure 20: Test RMSE vs number of hidden neurons using tanh activity function

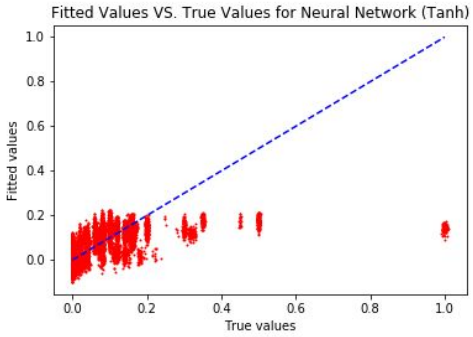

Figure 21: True values vs. fitted values

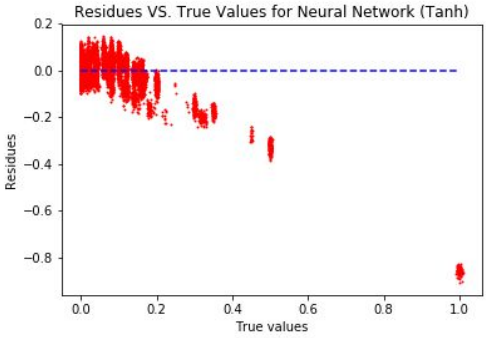

Figure 22: True values vs. residuals

By comparing the results using relu, logistic and tanh activity functions, we can make the conclusion that using relu function can achieve the best result with RMSE value as low as 0.01813. The next best activity function is the tanh activity function that achieve lowest RMSE of 0.0622. Then the last one is the logistic activity function that is 0.08835, much higher than the results of relu and tanh activity functions.

2.4 Prediction for Each of the Workflows

2.4.1 Linear Regression Model

In part 2a, we divide the dataset into 5 pieces: 'work_flow_0' to 'work_flow_4'. In this part, we will apply the linear regression model to each of the workflows separately. Since in part 2a, we already finish the feature selection of the dataset. We just need divide the preprocessed dataset into 5 groups based on their index of workflow. Then for dataset of each workflow, we apply the linear regression model and record the result. The training RMSE and testing RMSE of each workflow is shown in the following tables. The plots of predicted values vs true values and residual values vs predicted values of each workflow is also shown in the following pictures. Compare the RMSE of each workflows and the RMSE table shown in part 2a, we can see that the RMSE of work_flow_0, work_flow_2, work_flow_3 is much smaller than the general RMSE. In other words, the fit of model in these workflows is greatly improved. However, for work_flow_1, the RMSE is larger than the general one, and for work_flow_4, the RMSE does not have much difference than the general RMSE. Therefore, the fit of model for these two workflows do not have improvements.

Work_flow_0

# fold	Training RMSE	Testing RMSE
1	0.03571	0.03705
2	0.03595	0.03487
3	0.03572	0.03687
4	0.03571	0.03701
5	0.03578	0.03640
6	0.03602	0.03417
7	0.03567	0.03737
8	0.03602	0.03416
9	0.03575	0.03664
10	0.03603	0.03408

Table 8. Training and testing RMSE for work_flow_0

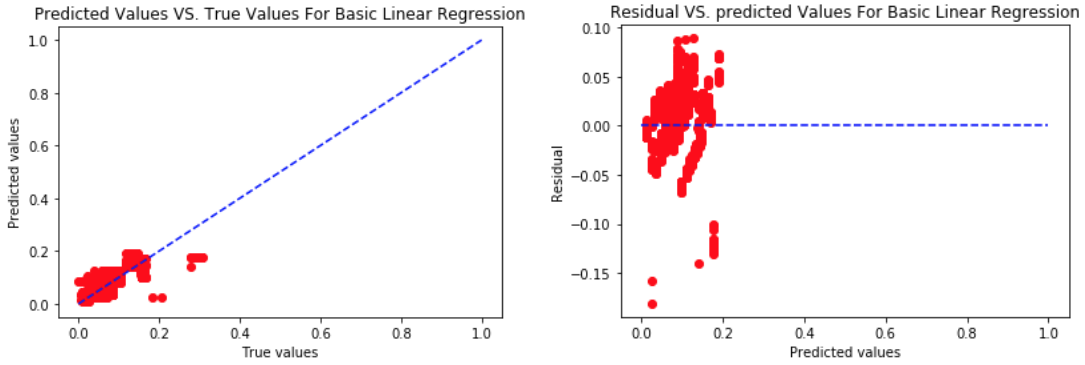

Figure 23: Predicted values versus true values; residuals versus predicted values for work_flow_0

Work_flow_1

# fold	Training RMSE	Testing RMSE
1	0.14618	0.17043
2	0.15131	0.12380
3	0.14621	0.17017
4	0.15131	0.12371
5	0.14621	0.17015
6	0.15126	0.12420
7	0.14618	0.17038
8	0.15130	0.12378
9	0.14618	0.17043
10	0.15130	0.12381

Table 9. Training and testing RMSE for work_flow_1

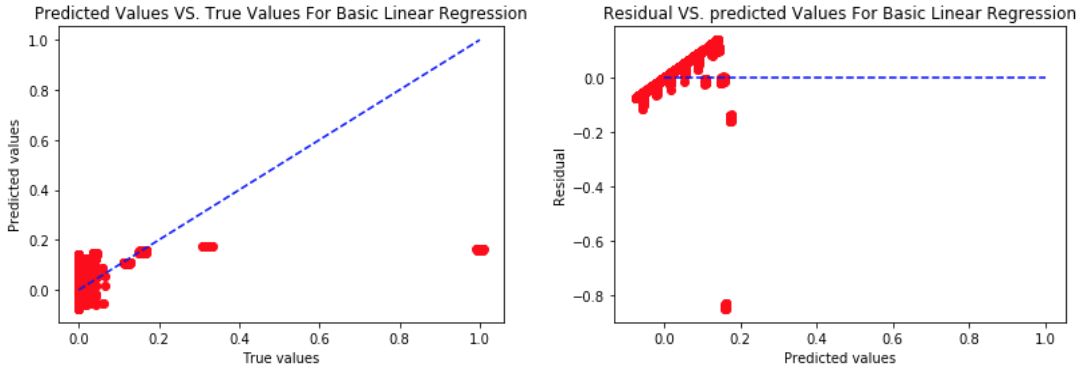

Figure 24: Predicted values versus true values; residuals versus predicted values for work_flow_1

Work_flow_2

# fold	Training RMSE	Testing RMSE
1	0.04359	0.03646
2	0.04228	0.04838
3	0.04366	0.03560
4	0.04249	0.04660
5	0.04366	0.03564
6	0.04168	0.05279
7	0.04340	0.03835
8	0.04236	0.04764

9	0.04368	0.03549
10	0.04223	0.04880

Table 10. Training and testing RMSE for work_flow_2

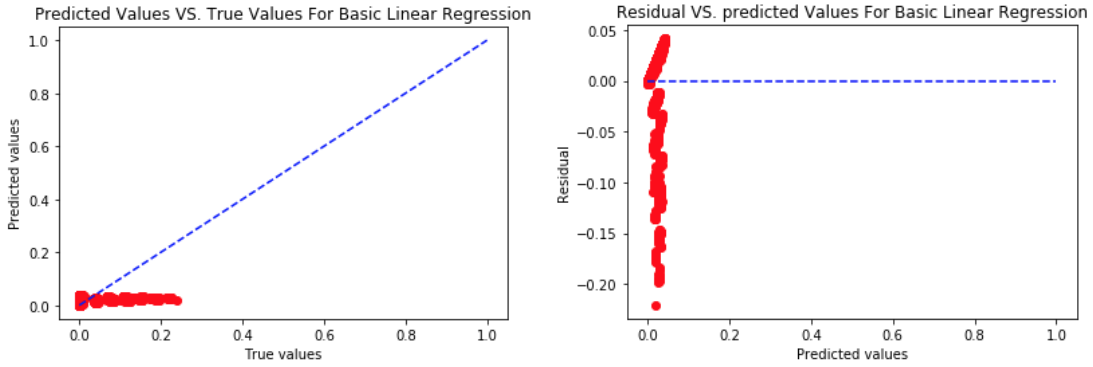

Figure 25: Predicted values versus true values; residuals versus Predicted values for work_flow_2

Work_flow_3

# fold	Training RMSE	Testing RMSE
1	0.00737	0.00604
2	0.00718	0.00783
3	0.00734	0.00636
4	0.00714	0.00816
5	0.00740	0.00568
6	0.00712	0.00832
7	0.00731	0.00661
8	0.00708	0.00857
9	0.00735	0.00627
10	0.00715	0.00802

Table 11. Training and testing RMSE for work_flow_3

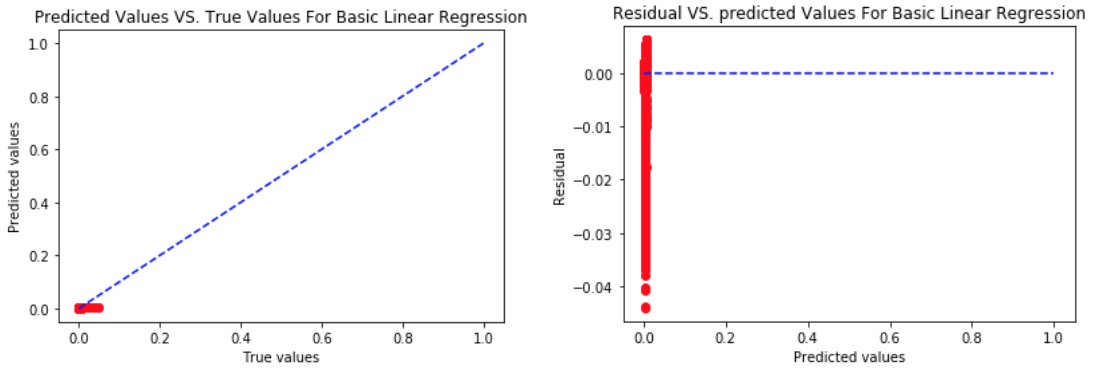

Figure 26: Predicted values versus true values; residuals versus predicted values for work_flow_3

Work_flow_4

# fold	Training RMSE	Testing RMSE
1	0.08716	0.07390
2	0.08473	0.09605
3	0.08710	0.07451
4	0.08460	0.09712
5	0.08701	0.07551
6	0.08491	0.09463
7	0.08704	0.07516
8	0.08476	0.09584
9	0.08707	0.07493
10	0.08477	0.09575

Table 12. Training and testing RMSE for work_flow_4

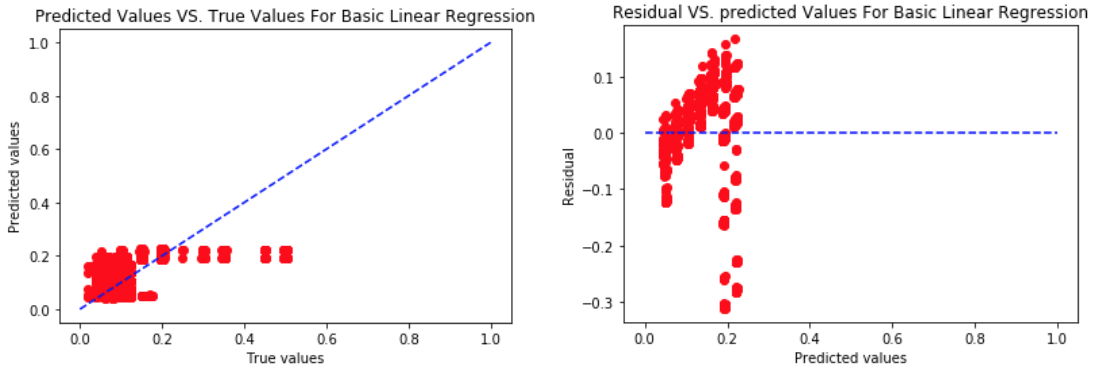

Figure 27: Predicted values versus true values; residuals versus predicted values for work_flow_4

2.4.2 Polynomial regression model

In this part, we try to fit a more complex model to the dataset. We add a polynomial function before the linear regression model, and increase the complexity of model by increasing the degree of polynomial. For each workflow, we try degree from 1 to 10, and plot the train and test RMSE vs the degree of polynomial. These plots are shown in the following pictures. For workflow_0, degree 7 is the threshold where the error is lowest. For workflow_1, degree 9 is the best. For workflow_2, degree 6 gives the smallest test error. For workflow_3, degree 5 is the threshold where the test RMSE is the lowest. For workflow_4, degree 5 is the best. Cross validation helps controlling the complexity of the model because it uses a part of the training data for the model estimation. Cross validation allows the model to traverse all data in the training model with validation process, which will help reduce the overfitting possibility and increase the generalization ability of the model, and then reduce the complexity of the model.

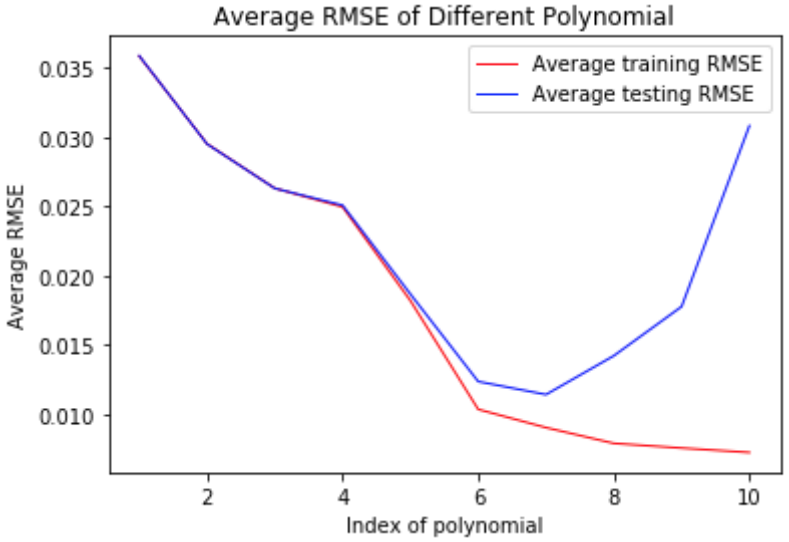

Figure 28. Average RMSE vs different polynomial for workflow_0

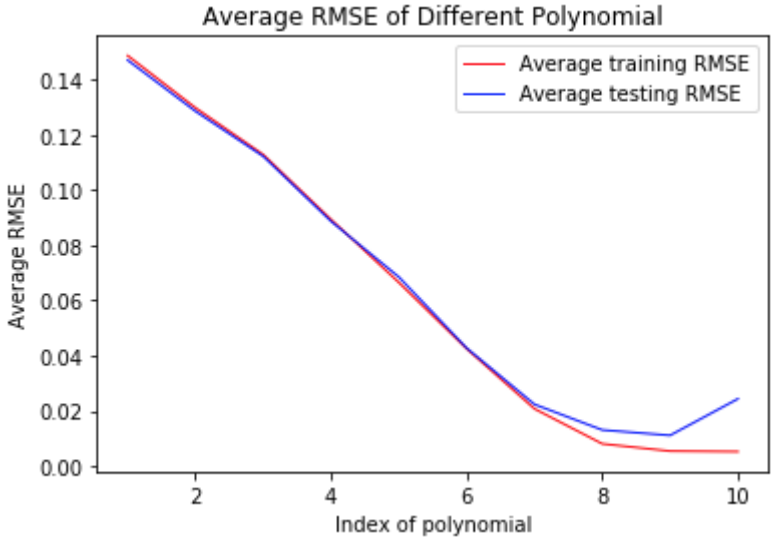

Figure 29. Average RMSE vs different polynomial for workflow_1

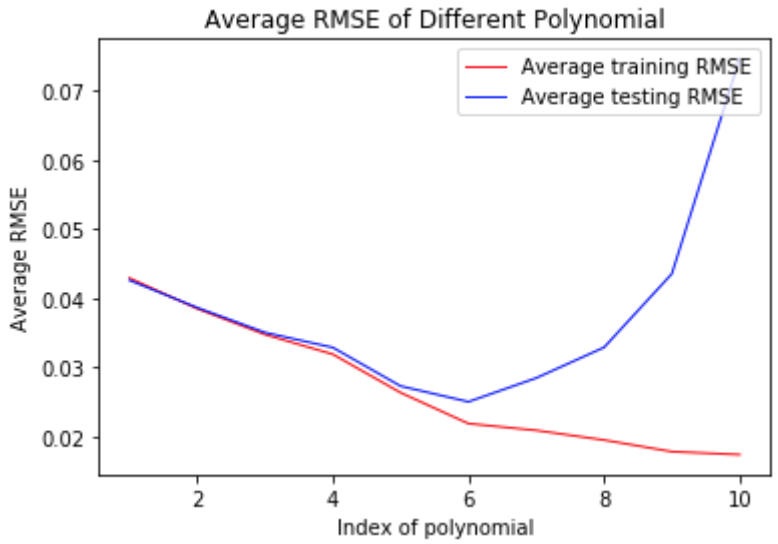

Figure 30. Average RMSE vs different polynomial for workflow_2

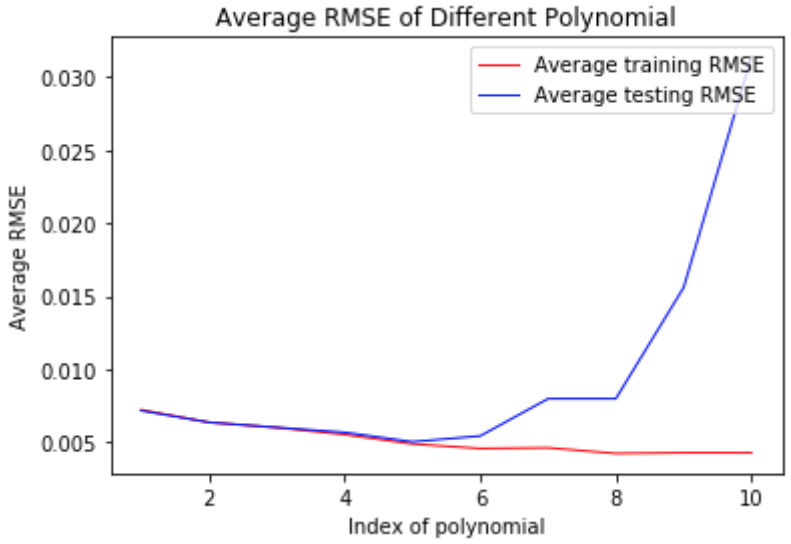

Figure 31. Average RMSE vs different polynomial for workflow_3

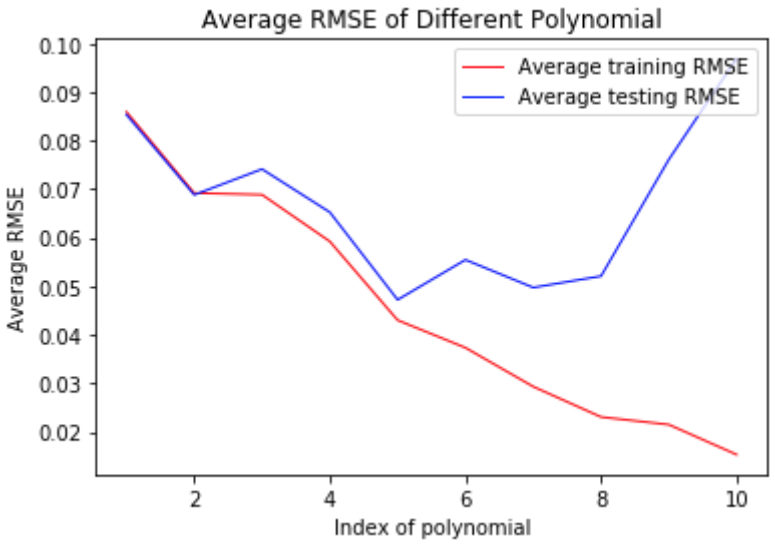

Figure 32. Average RMSE vs different polynomial for workflow_4

2.5 Prediction using KNN Regressor

In this part, we apply the k-nearest neighbor regression model to the preprocessed data. neighbor index are chosen from 1 to 8. The plot of average of train and test RMSE vs neighbor index is shown below. From the picture, it is obviously that k = 1 is the best parameter for this dataset. Then, the plots of predicted values vs true values, and residual values vs predicted values are recorded and shown below.

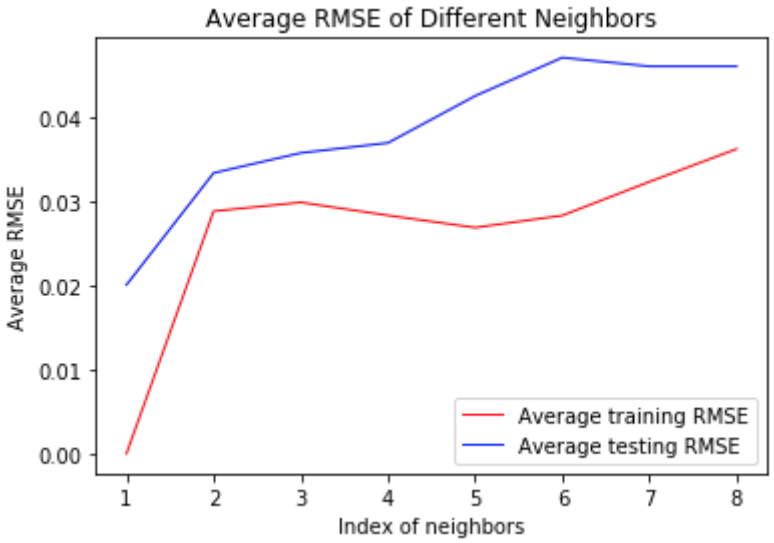

Figure 33. average RMSE of different neighbors vs index of neighbors

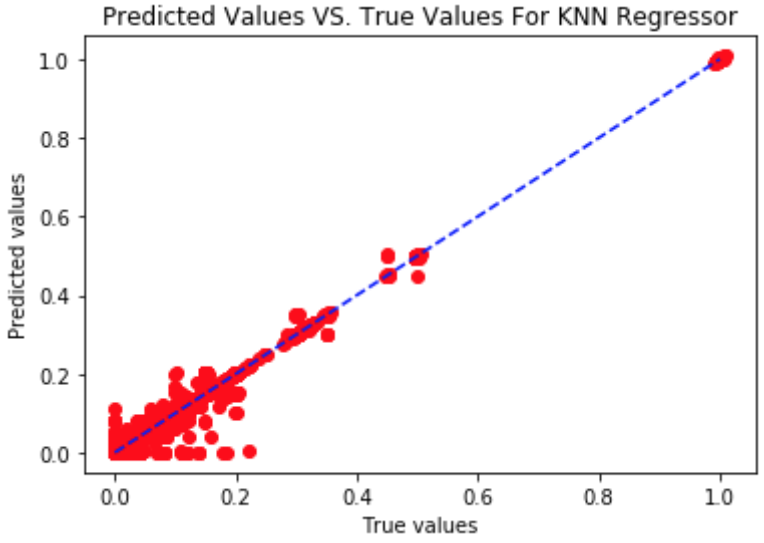

Figure 34. predicted values vs true values for k=1

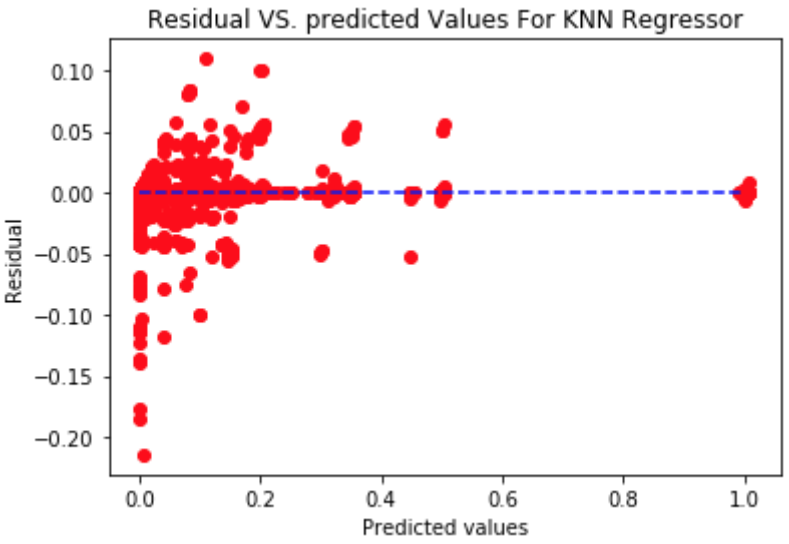

Figure 35. residual values vs predicted values for k=1

3. Comparisons among the models

There are in total four models that are used in this project for comparison purpose: **linear regression model, random forest regression model, neural network regression model and k-nearest neighbor regression model**. The models are analyzed in several aspects: whether the model is good at handling categorical features by itself or scalar encoding, whether the model is good at handling sparse features which are obtained by one-hot encoding, whether the residuals are randomly distributed around the boundary x=0, or there is a pattern for the distribution of the residuals, whether the model has comparable RMSE in both training and testing data, what hyperparameters work best for the model, whether different hyperparameters (or different number or types of layers in the case of the NN model) affect the result significantly.

For the linear regression model, it can be treated as the baseline model, since it is the most straightforward and the most fundamental among all the models implemented in this project. Basically, the linear regression model is finding the best linear regression line which is able to fit most of the data points. This model is expected to be outperformed by the other models, because the relation between the categorical features and output(backup size) is not necessarily interpretable by a simple linear model. According to the plots of residuals versus fitted values in 2.1, the residuals in the plots are not distributed randomly as what a good model behaves. There is no obvious improvement in the performance from 2.1.1 to 2.1.3, but a relatively evident improvement from 2.1.3 to 2.1.5. As what is expected, one-hot encoding is also empirically a better means for feature representation in the linear regression model than scaler encoding. The data points in the plot in 2.1.5 is more spread than those in 2.1.3. Moreover, regularization influence the model to a significant extent. The regularized model can make better use of one-hot feature representation, and hence obtain lower RMSE in both training and testing sets.

For the random forest model, it is able to handle the categorical features without encoding. However, due to the limitation of the sklearn package(as we discussed in part 2.2), we use scalar encoding during the experiment. We tuned three parameters of the model, which are number of trees, max number of features and max depth. Increasing all the hyperparameters will improve the performance but resulting in the increasing of computation time, while it may also cause some overfitting issue. The best model we derive gets testing RMSE as low as 0.0135, which is much better compared to the linear regression model. Another pro of random forest regression model is that since it uses bagging method, we can use the out of bag error to evaluate the model instead of using cross validation method which takes much more time.

For the neural network model, we are encoding all of the features using the one hot encoding. In other words, we are testing whether the model can handle sparse features. We are also analyzing the best number of hidden neurons in the hidden layer to find the best hyperparameter for each model using different activity functions. By comparing the RMSE results of the three activity functions, we can make the conclusion that using relut activity function can let us achieve the lowest RMSE value with smooth decrease of RMSE when the number of hidden neurons increase. Furthermore, the true values vs fitted values graph and true values vs residuals graph show that using the relu activity function, the results are a pattern surrounding the ideal line. Overall, the combination of using neural network regressor with relu activity function can achieve a fairly good result.

For the KNN regression model, it is a non-parametric model so it provides a more

flexible approach to the model fit. KNN regression model identifies the k neighborhoods of each input data, and estimate the predicted value as the average of all training result in the neighborhoods. By tuning the parameter of the model, which is the index of neighbors, we are able to optimize the fit of model and get a lower RMSE. As discussed in 2.5, choosing k = 1 gives the best fit, and the lowest RMSE value is 0.02012. This RMSE is much lower compared to the result in linear regression model.

Overall, the random forest regression model is the best at handling categorical features, since it is the only model which can handle the features without converting it into scalar or one-hot format. The KNN regression model is the best at handling sparse one-hot features, because it performs the best in using one-hot feature representation during the training and testing time. Besides, the linear regression model has a small fall in performance with all feature encoded in one-hot form. The random forest model overall generates the best output, it has the lowest test and train RMSE and the most reasonable distribution of the data points in reference to the true labels.