EE219 Project4
Regression Analysis

Qidi Sang 705028670
Hui Wang 205036597
Zhonglin Zhang 005030520

Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables.

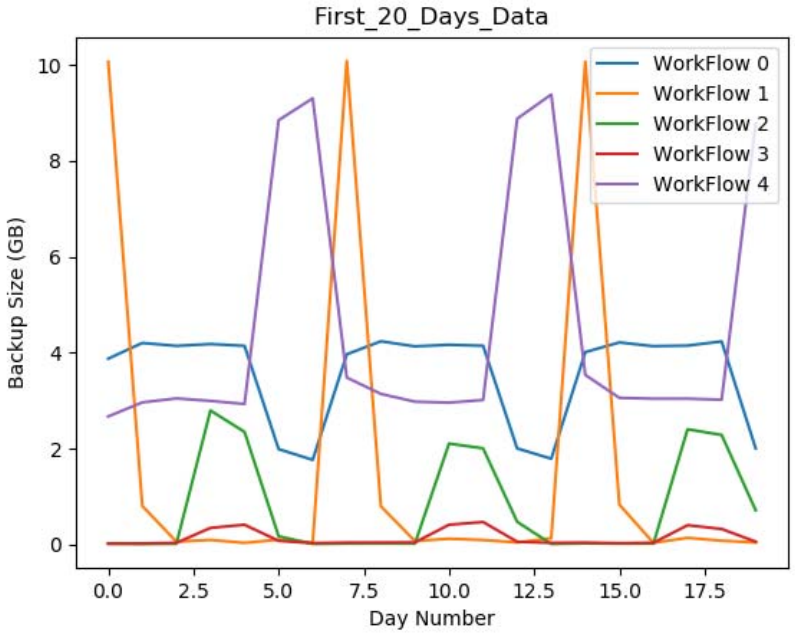
In this project, we analyzed the Network backup Dataset, which is comprised of simulated traffic data on a backup system over a network. This dataset captures simulated traffic data on a backup system and contains information of the size of data moved to the destination as well as the time it took for backup. The dataset has around 18000 data points with the following columns/variables: week index, day of the week at which the file backup has started, backup start time: Hour of the day, workflow ID, file name, backup size: the size of the file that is backed up in that cycle in GB, and backup time: the duration of the backup procedure in hour. Our task is to predict the backup size of the traffic depending on the other variables using different prediction models and evaluate the performances of these models.

We explored basic regression models, such as Linear Regression, Random Forest, Neural Network and Polynomial Regression on the Network backup Dataset, along with basic techniques to handle over-fitting; namely cross-validation, and regularization. With cross-validation, we tested for over-fitting, while with regularization we penalized overly complex models.

1. Load the dataset

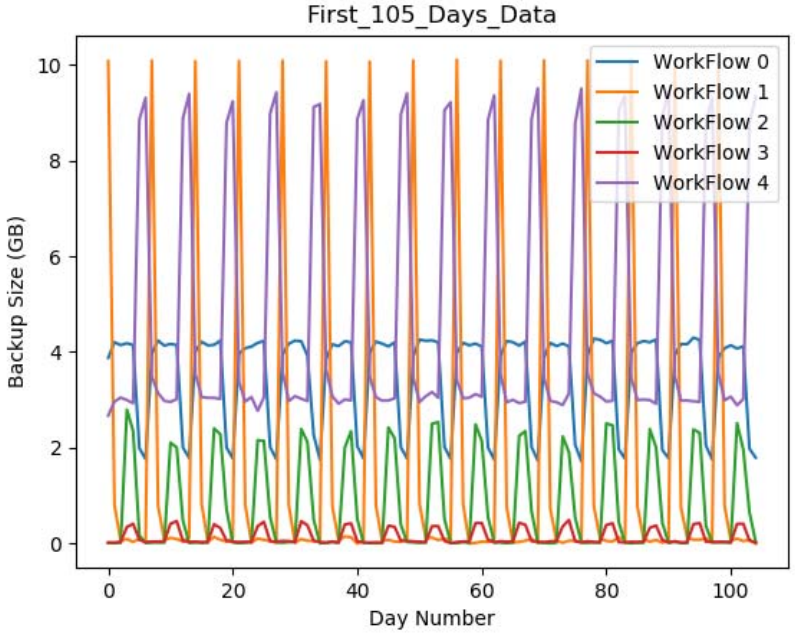
(a) Plot a 20-day period backup sizes for all workflows

We plotted the backup size data in the first 20-day period and the result is shown in the following figure. Meanwhile, we can plot the data in any period by passing different parameters to the function ‘plot_back’ in our codes. According to the plot, we can find that for all the five workflows, their backup sizes are distributed periodically. The period is about 7 days.



(b) Plot a 105-day period backup sizes for all workflows

Like the former question, we set the start date to 0 (the first day in the dataset) and the end date to 104 to plot the accumulative backup size in the first 105 days. The plot is in the following. Similarly, the periodicity remains in the first 150 days.



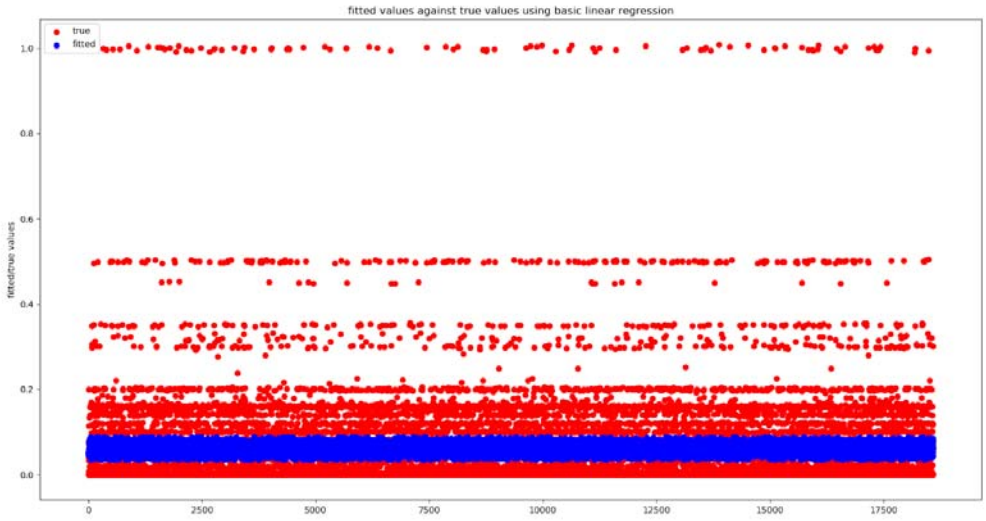
2. Predict

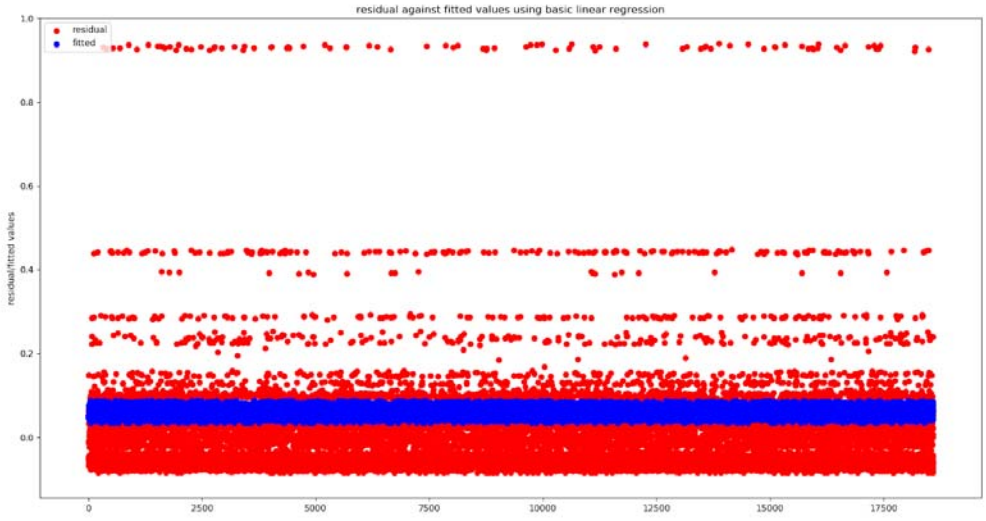
(a) Linear regression model

In the first 3 tasks (scalar encoding), we use shuffled data. And in task 4 and 5, we use non-shuffled data so as to observe a test RMSE increase in one hot encoding and try to solve it.

i. Fit a basic linear regression model using scalar encoding

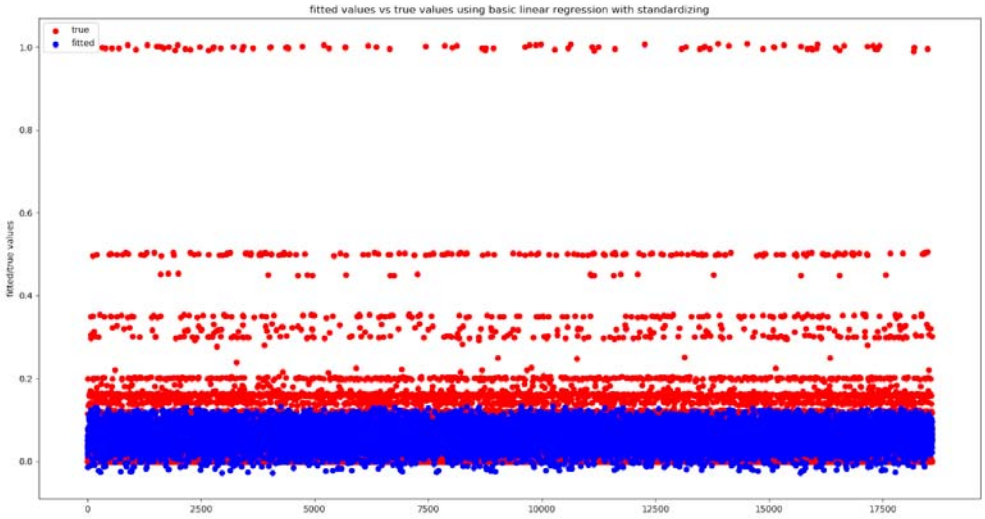
For basic linear regression model with scalar encoding, the average training RMSE is 0.10358765917764905, average test RMSE is 0.10363231433154203.

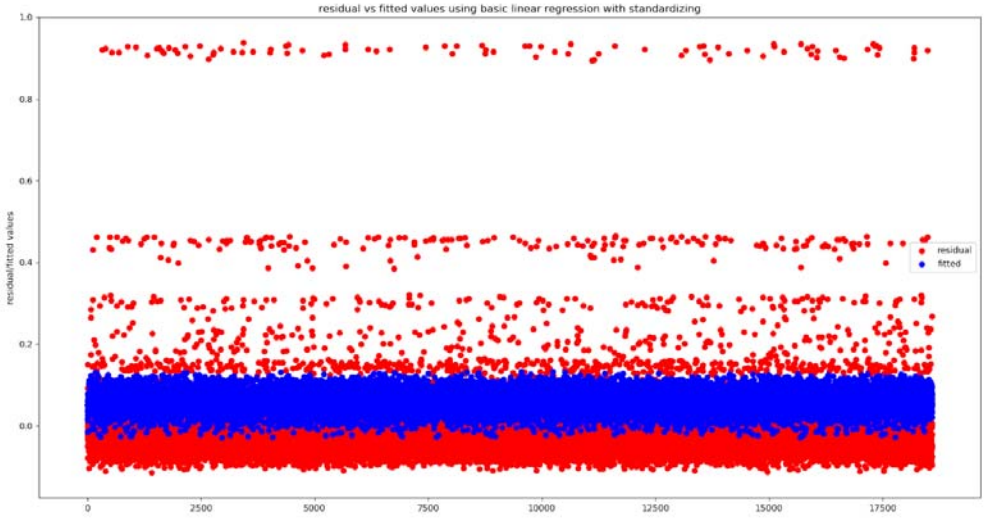




ii. Data preprocessing

For basic linear regression model with Standardized scalar encoding, the average training RMSE is 0.10138534238595381, average test RMSE is 0.10141585039159018.





We can see the test RMSE has slightly decreased compared to no standardizing. And fitted values has a better range. This shows that standardizing can help improve performance because it makes sure each feature has the same mean and variance, thus giving each feature same weight.

iii. Feature selection

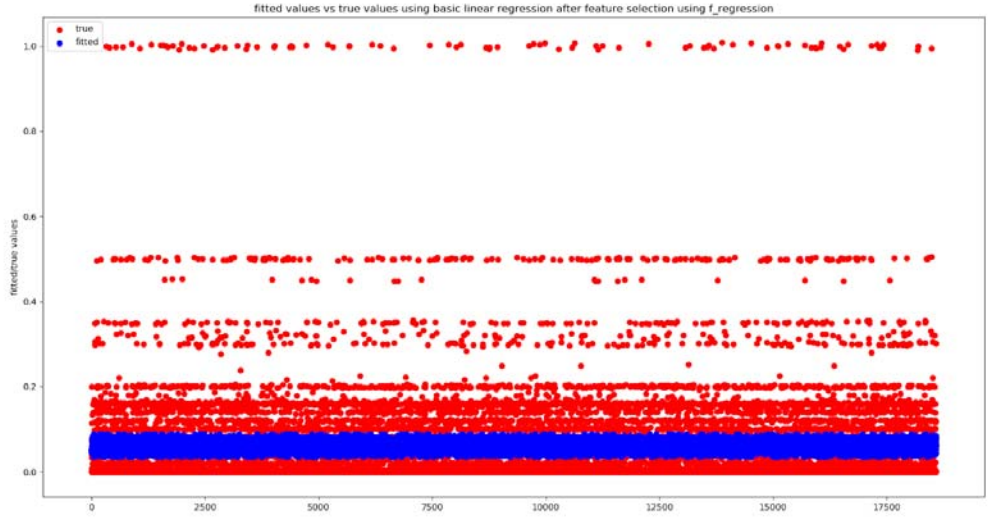
Using f_regression, the F values are:

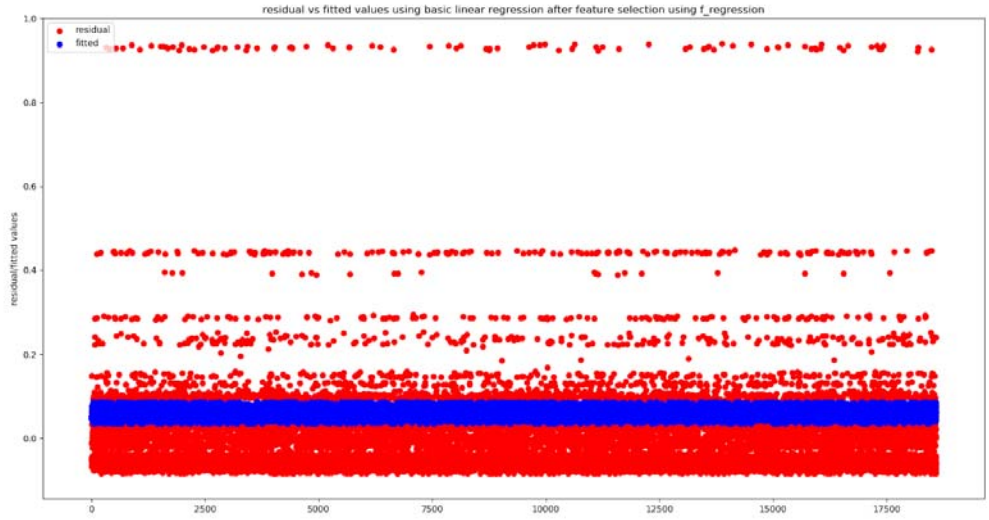
[8.45006257e-03 3.88163798e+01 1.50740934e+02 2.61386654e+01 2.53200943e+01]

And the p values are:

[9.26759237e-01 4.75614169e-10 1.62474985e-34 3.20909922e-07 4.90153868e-07]

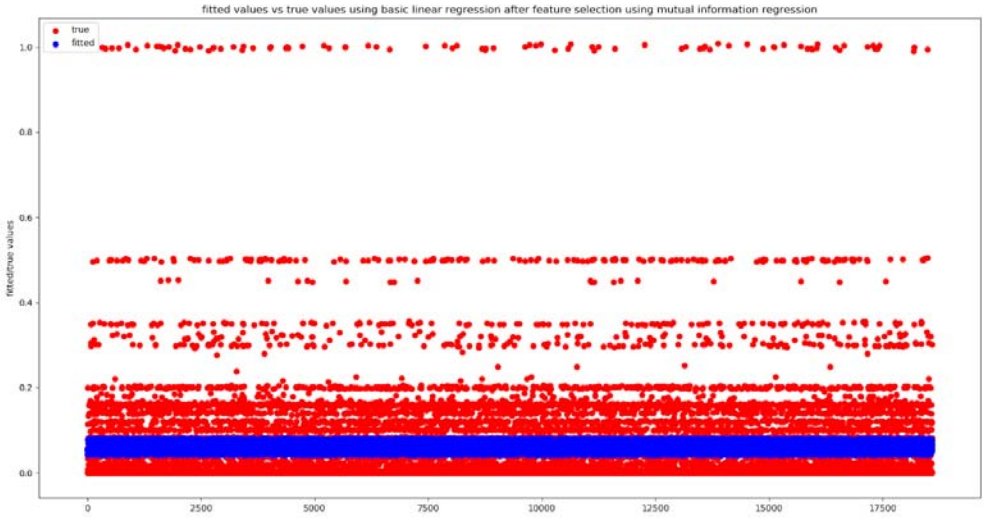
Higher F values and lower p values means feature's value is more related to the variable. Based on f_regression results, we choose Day of Week, Backup Start Time, Work-Flow-ID as three most important variables.
For basic linear regression model after feature selection using f_regression, the average training RMSE is 0.1035885930336633, average test RMSE is 0.10361570856861975

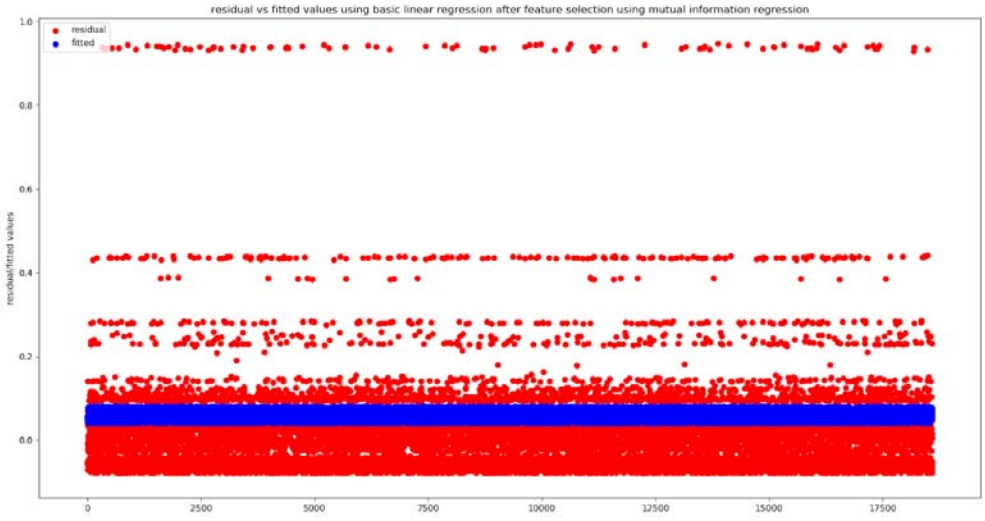




Using mutual information regression, the estimated mutual information between each feature and the target are:
[0.00603987 0.22937479 0.23953897 0.26851717 0.42950362]

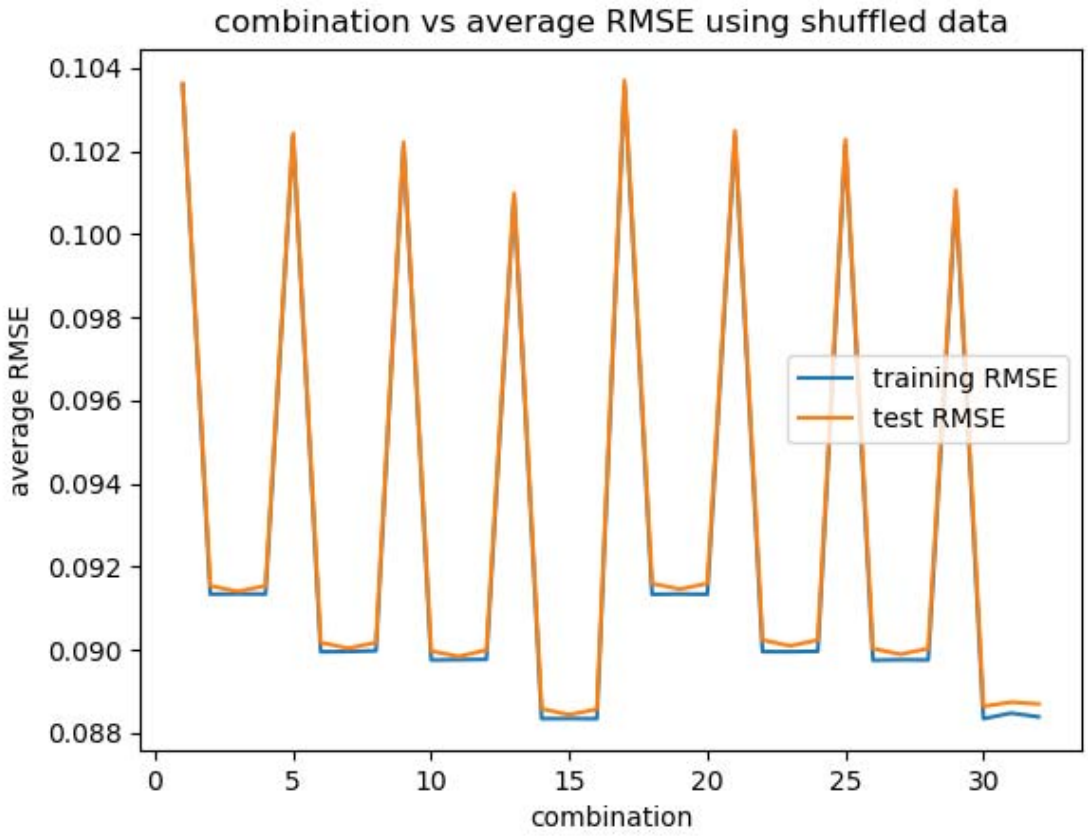
High mutual information means feature and variable are highly dependent. Based on mutual information regression results, we choose Backup Start Time, Work-Flow-ID, File Name as three most important variables.
For basic linear regression model after feature selection using mutual information regression, the average training RMSE is 0.10369750211483769, average test RMSE is 0.10371612473193641.



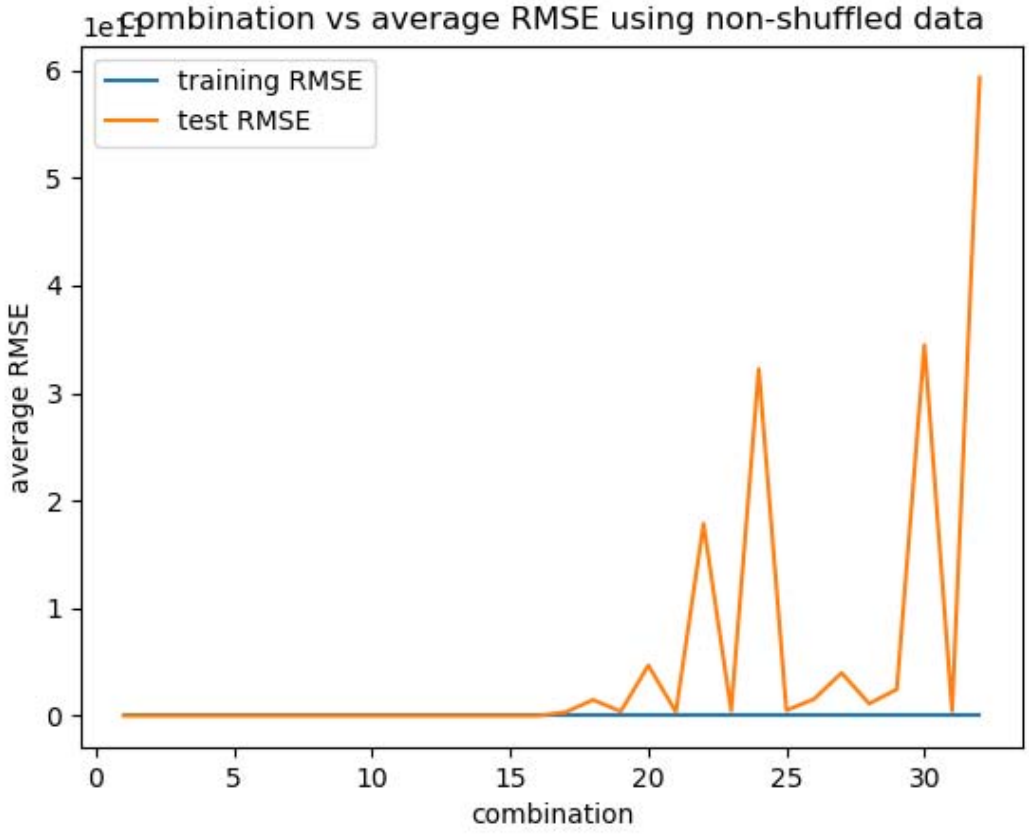


However, the results did not really improve after feature selection. It could be that other unselected features play role in predicting and abandoning them will do harm to accuracy. However, using fewer features speeds up the predicting procedure while only results in a small increase in RMSE.

iv. Feature encoding



Using shuffled data, in 32 one hot encoding combinations, we observe no sudden increase in test RMSE. On the contrary, they are quite close to each other. Then we switched to non-shuffled data.



To note, the y axis in non-shuffled results is in 1e11, which means test RMSE could be over tens of billion, which is quite large. And we can see that those spikes only appear when feature 'week number' is one hot encoded. That's because if we do not shuffle the data, they are ordered from week 1 to week 15. And doing 10-fold cross validation means we will always use data from 13.5 weeks to predict results for the other 1.5 week; and 1 week in our validation set has never showed up in our training set. And through one hot encoding, we are giving feature 'week number ' more weight, and failing to incorporate that 1 week data introduces very large test RMSE.

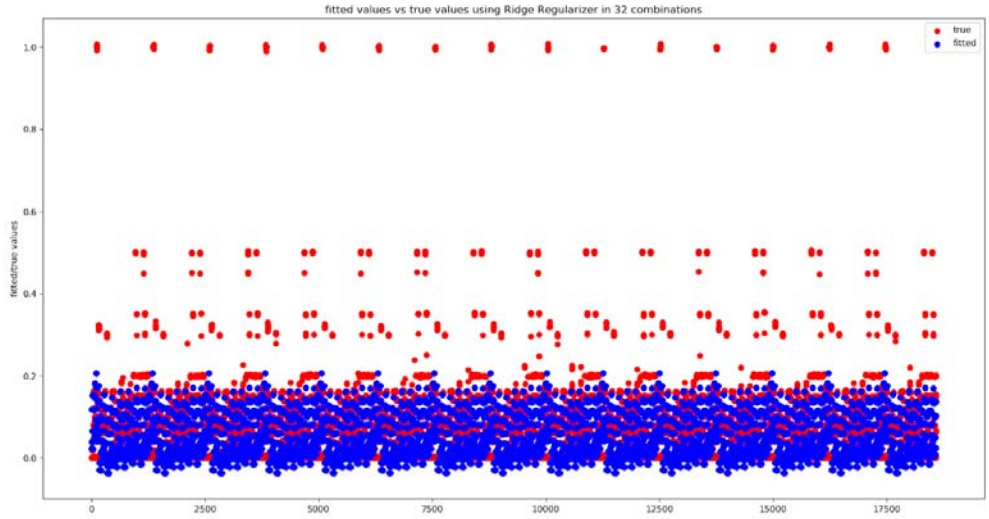
And the best combination is one hot encode only Day of week, Back-up start time and File name. Lowest test RMSE in 32 combination is 0.08850377062577282. An intuitive explanation is that there do not exist a natural ordering in categories from those features and can perform better if we don't use scalar encoding.

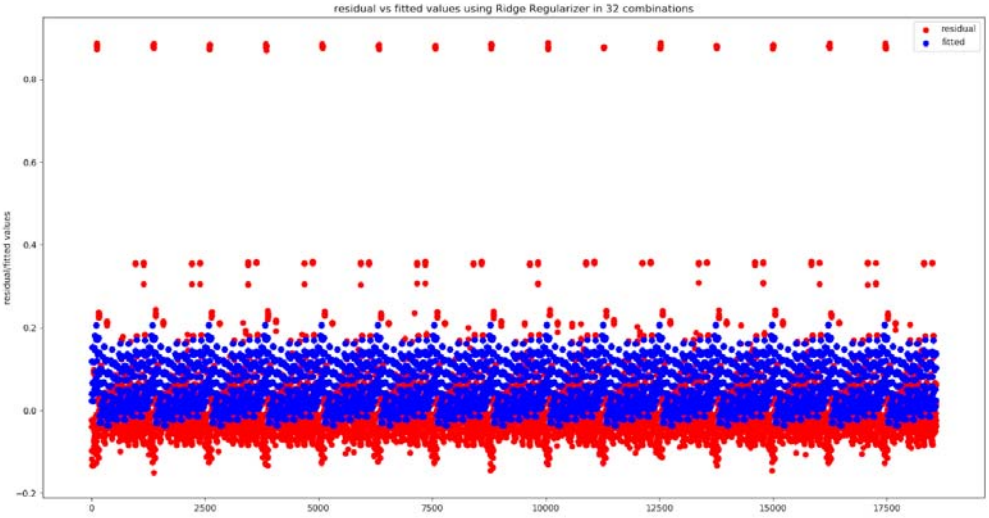
v. Controlling ill-conditioning and over-fitting

1.Ridge Regularizer

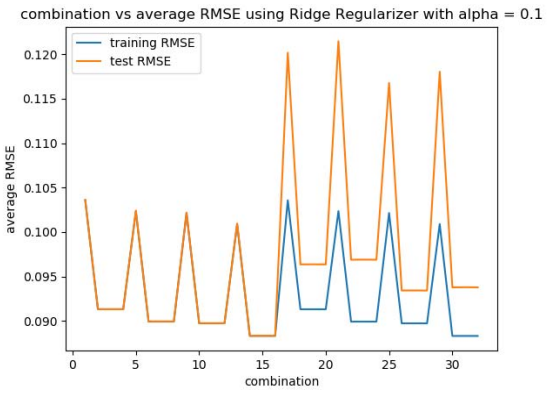
Lowest test RMSE with Ridge Regularizer is 0.08850347786271374, alpha = 10.0.

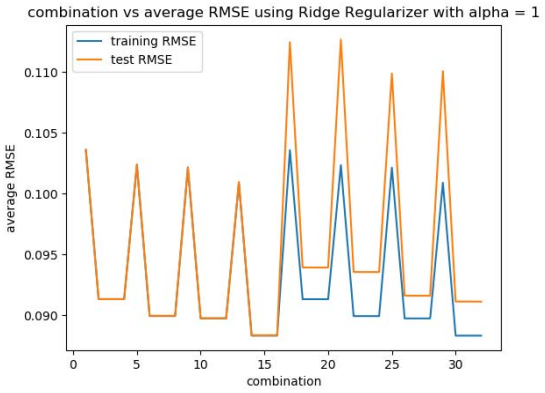
The combination is one hot encode Day of Week, Backup Start Time, and Work-Flow-ID.

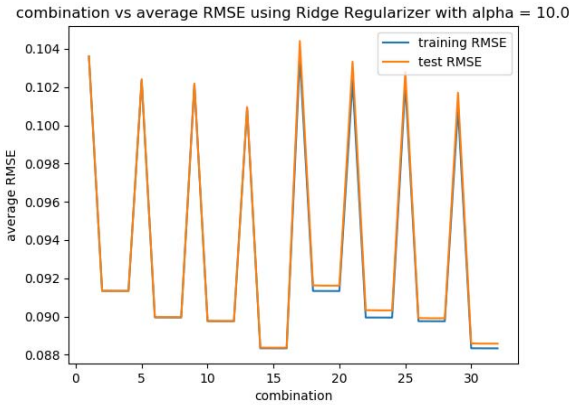


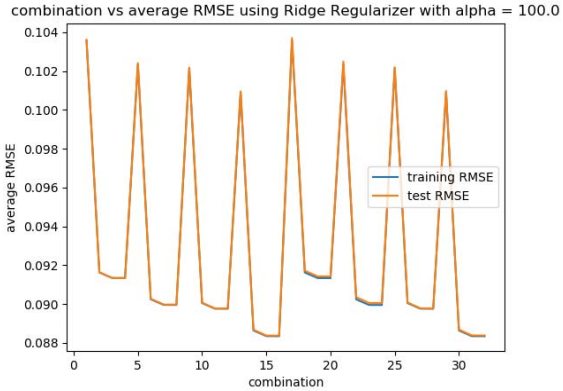


We also plotted training RMSE and test RMSE difference on different alpha.
We can see as alpha increases, training RMSE and test RMSE are getting closer.





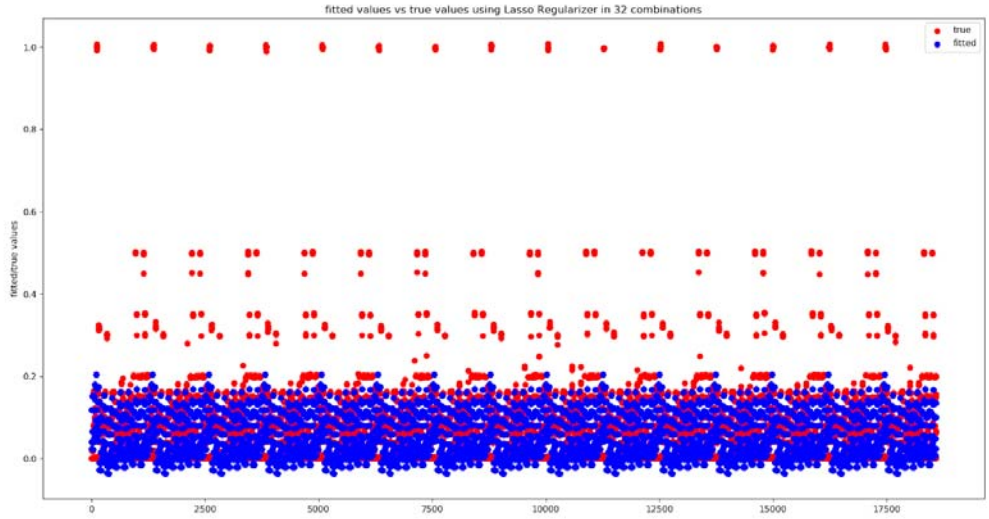


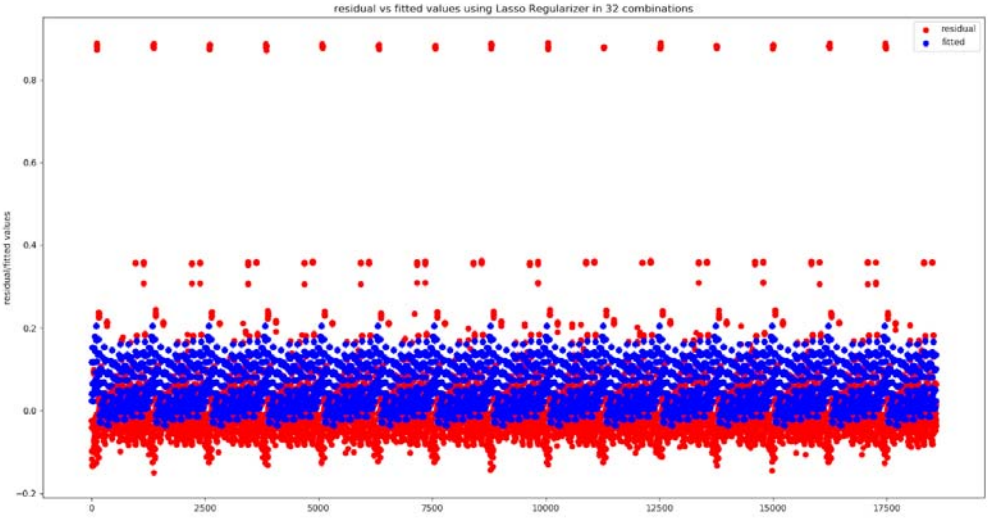


2.Lasso Regularizer

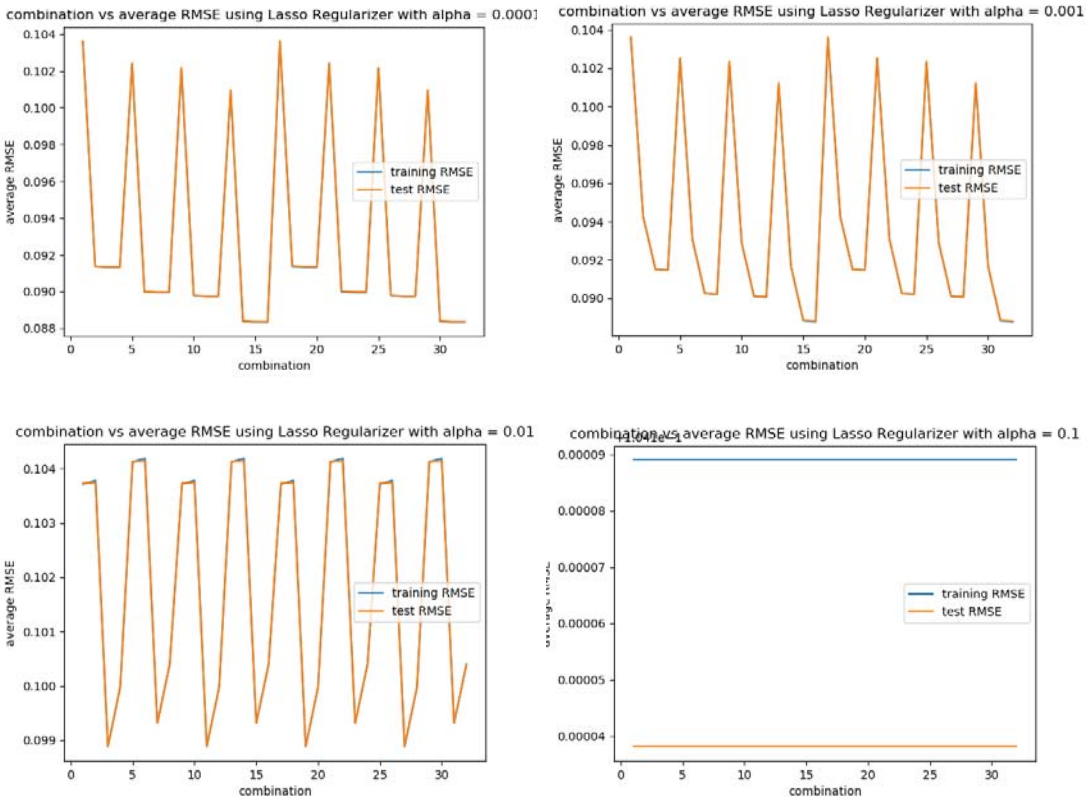
Lowest test RMSE with Lasso Regularizer is 0.08850780525116371, alpha = 0.0001

The combination is one hot encode all features.





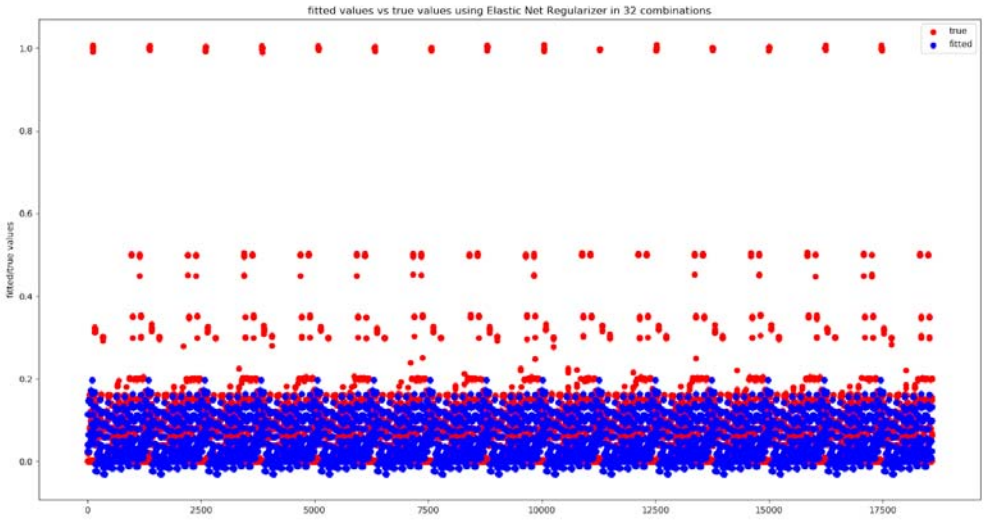
We also plotted training RMSE and test RMSE difference on different alpha.
We can see as alpha decreases, training RMSE and test RMSE are getting closer.

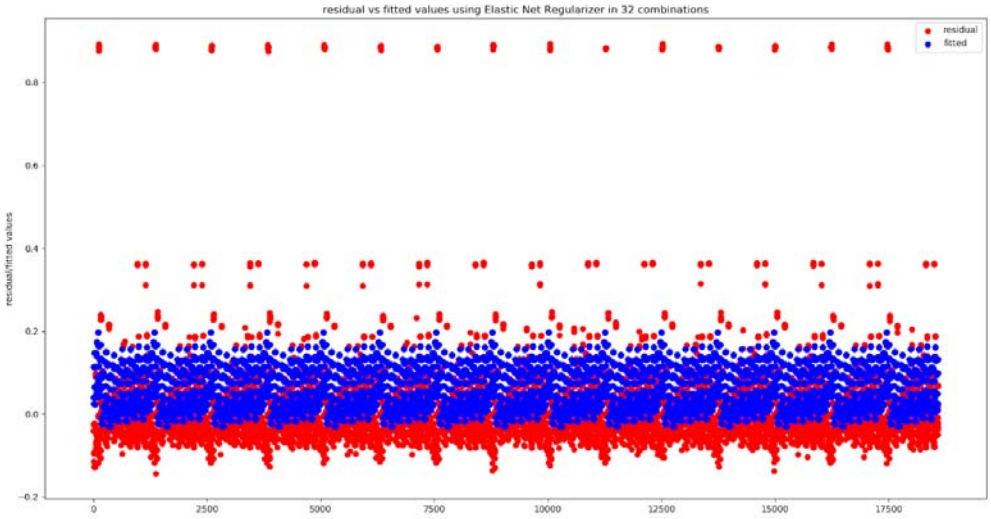


3.Elastic Net Regularizer

Lowest test RMSE with Elastic Net Regularizer is 0.08856704984262825, lambda1 = 0.0001, lambda2 = 0.01

The combination is one hot encode all features.





Coefficients of lowest test RMSE using basic linear regression are:
[3.74168041e-02 -1.47589152e-02 -2.21619298e-02 -7.14310922e-03
 -7.60124374e-03 1.38233784e-03 -3.98150005e-04 -2.24158012e-02
 -2.32722733e-02 5.58078351e-03 3.12440329e-02 -4.19783036e-03
 -2.03117592e-04 4.14278345e-02 4.26766737e-02 4.23476655e-02
  4.23211689e-02 4.19306262e-02 4.21770761e-02 -1.19395294e-02
 -1.24232028e-02 -1.22144859e-02 -1.24980549e-02 -1.26028297e-02
 -1.22749159e-02 -4.13893333e-02 -4.09747879e-02 -4.05005150e-02
 -4.15685192e-02 -3.90491561e-02 -4.03808913e-02 -5.96903124e-02
 -5.93632276e-02 -5.93043550e-02 -5.96639892e-02 -5.97125951e-02
 -5.94434510e-02 6.78815846e-02 6.84449894e-02 6.86344858e-02
  6.75682389e-02 6.82143479e-02 6.81052540e-02 1.12089750e-05
  2.18238777e-03] 0.060983512625039114

Coefficients of lowest test RMSE with Ridge Regularizer are:
[3.87675448e-02 -1.32024376e-02 -2.05928851e-02 -5.62447352e-03
 -6.07318508e-03 2.88345404e-03 1.11468135e-03 -2.05703619e-02
 -2.14437337e-02 7.29313250e-03 3.28880907e-02 -2.42801160e-03
  1.53358284e-03 3.85919916e-02 -1.40759539e-02 -4.06271473e-02
 -5.77994352e-02 7.11832437e-02 1.95767259e-05 9.31227086e-05]
0.060947220148083146

Coefficients of lowest test RMSE with Lasso Regularizer are:
[-0.00000000e+00 -0.00000000e+00 0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 -0.00000000e+00 0.00000000e+00

-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 4.37513766e-02
-6.96907942e-03 -1.44485670e-02 -0.00000000e+00 -0.00000000e+00
7.71880392e-03 5.93830959e-03 -2.10345963e-02 -2.18506978e-02
5.78177598e-03 3.14450311e-02 -2.79293029e-03 2.96261921e-06
5.17787530e-02 -0.00000000e+00 -2.56788220e-02 -4.23859230e-02
8.64849397e-02 -0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
-0.00000000e+00 0.00000000e+00 -0.00000000e+00 -0.00000000e+00
0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
-0.00000000e+00 0.00000000e+00 0.00000000e+00] 0.043128472292066

Coefficients of lowest test RMSE with Elastic Net Regularizer are:

[-0.00000000e+00 -0.00000000e+00 0.00000000e+00 -0.00000000e+00
0.00000000e+00 -0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 -0.00000000e+00 0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 3.74090893e-02
-9.93812289e-03 -1.69921547e-02 -2.94079437e-03 -3.33731222e-03
3.74714101e-03 2.08124577e-03 -1.85554996e-02 -1.92452198e-02
6.76745382e-03 3.10011059e-02 -1.40431416e-03 1.33508573e-03
3.83464192e-02 -1.04840856e-02 -3.53941181e-02 -5.11872025e-02
6.85174464e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -5.35046071e-05 -0.00000000e+00 -0.00000000e+00
-1.79761205e-04 -0.00000000e+00 -0.00000000e+00 -3.21027688e-04
-6.84432048e-05 -2.29648274e-05 -3.00645961e-04 -3.38159614e-04
-1.30315643e-04 2.88259335e-03 3.31766880e-03 3.46402238e-03
2.64072554e-03 3.13966260e-03 3.05545268e-03] 0.056961693297227764

In the regularized good models, the coefficients are sparse compared to un-regularized best model.

(b) Random forest regression model

In the random forest section, we used scalar encoding features. More specifically, we preprocessed the dataset values, converting them into a one dimensional numerical value. For example, Day of the Week variable takes on values 1, … , 7, corresponding to Monday through Sunday; the Hour of the Day is encoded as 1, … , 24; Workflow ID is encoded as 0, … , 4; File name is also encoded as one dimensional numerical value.

i. Evaluate the initial random forest model

In this part, we used RandomForestRegressor from sklearn, there are four initial parameters to be set:

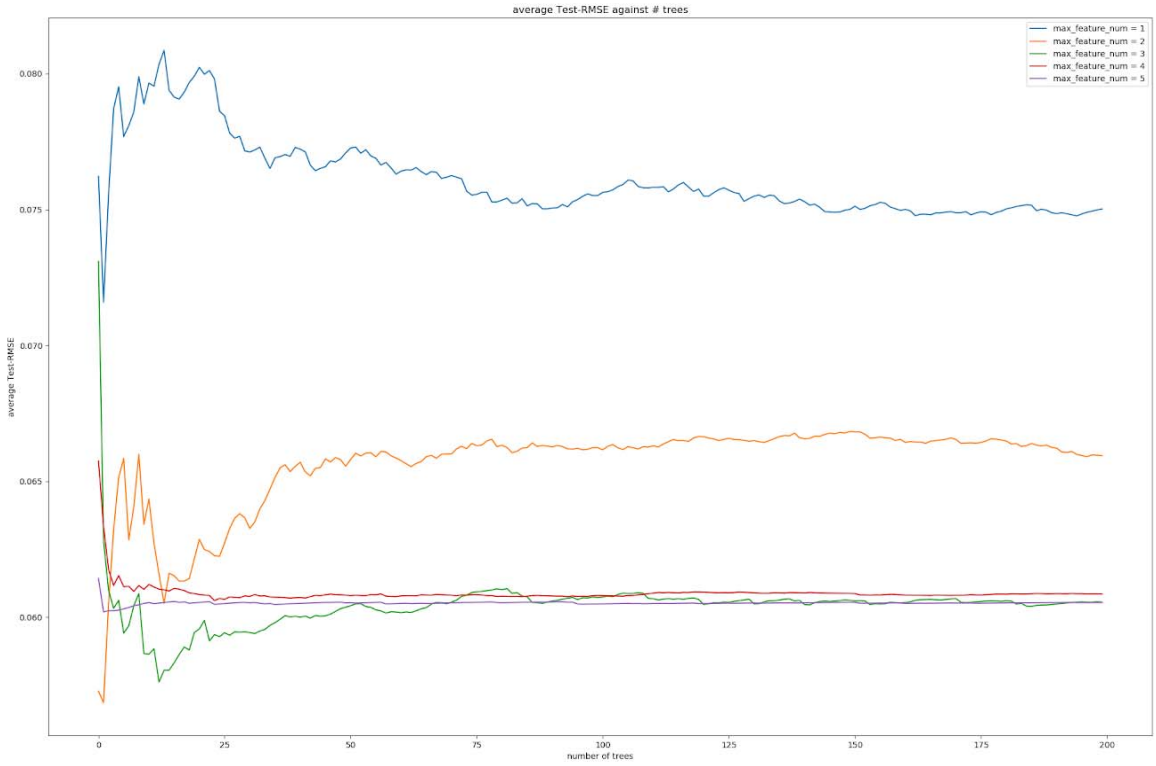
1. n_estimators: The number of trees in the forest. Typically, more trees will decrease the variance of the model.
2. max_depth: The maximum depth of the tree. Typically, deeper trees will decrease the bias of the model.
3. Bootstrap: Whether bootstrap samples are used when building trees.
4. max_features: The number of features to consider when looking for the best split.

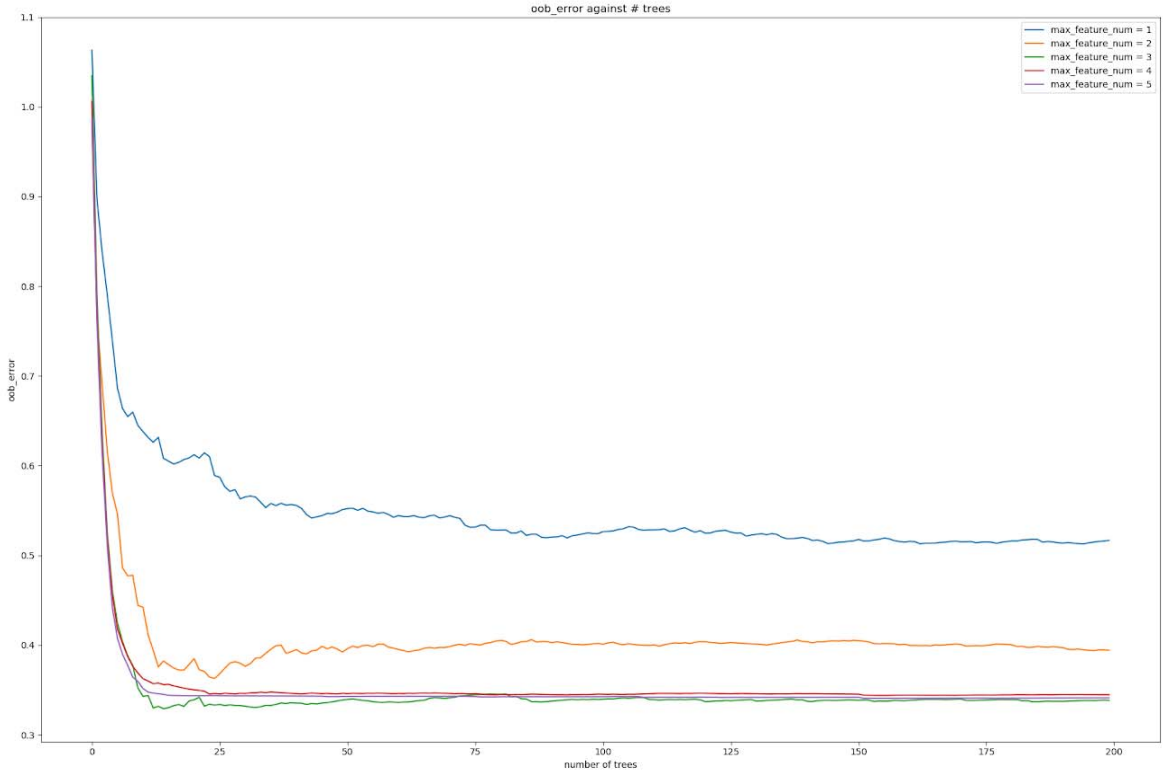
Here we set the parameters as follows: n_estimators = 20, max_depth = 4, Bootstrap = True, and max_features = 5. And then we used 10-fold cross validation.
This is the average training and test RMSE and Out of Bag error from this initial model:

average rmse for trainset	0.06048324818673603
average rmse for testset	0.06078824937279181
Out of Bag error	0.3346098549289712

From the above values we can see that using these initial parameters in the model, the average rmse and out of bag error is quite high, which means not ideally performance. Thus we need to find optimal parameter values to get a best performance using this model. Moreover, the trainset rmse is less than the testset rmse.

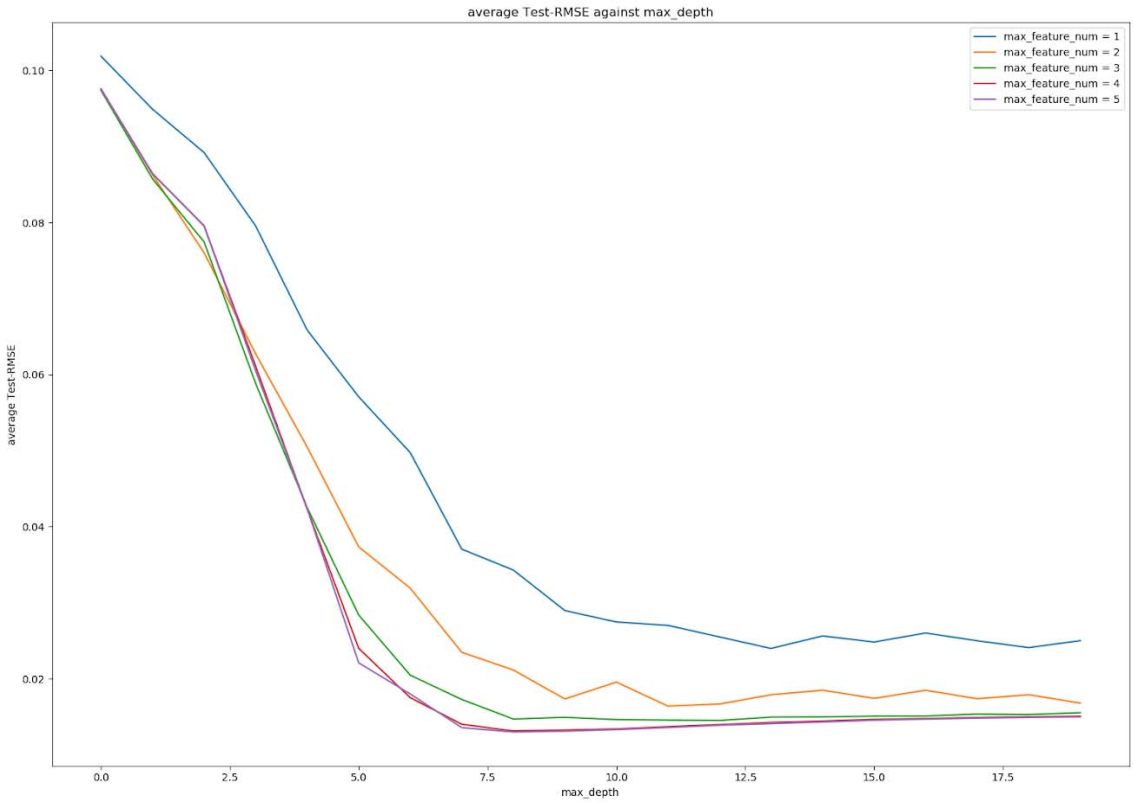
ii. Find best number of trees and maximum number of features
In this part, we experimented on the number of trees (n_estimators parameter) and maximum number of features (max_features parameter) in the RandomForestRegressor. We swept over number of trees from 1-200 and maximum number of features from 1-5 and plotted two figures as below:

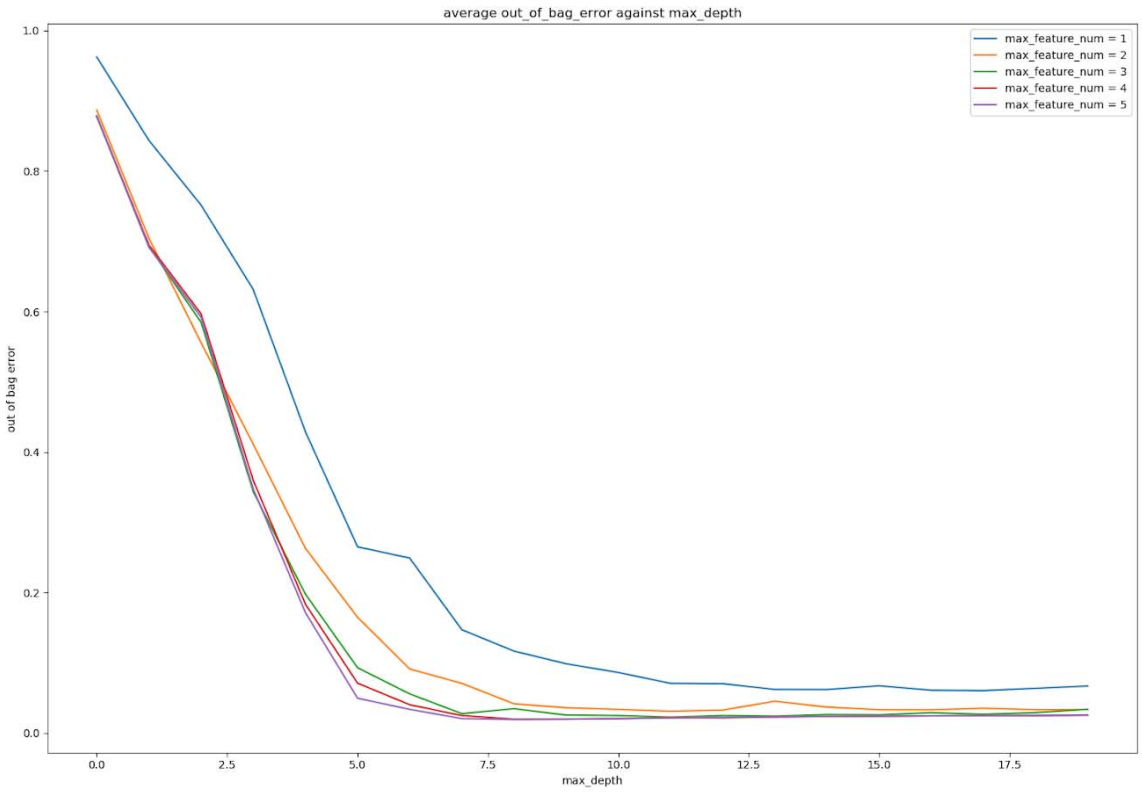

average Test-RMSE against # trees
max_feature_num = 1
max_feature_num = 2
max_feature_num = 3
max_feature_num = 4
max_feature_num = 5
average Test-RMSE
number of trees


oob_error against # trees
max_feature_num = 1
max_feature_num = 2
max_feature_num = 3
max_feature_num = 4
max_feature_num = 5
oob_error
number of trees

The first figure is the average Test-RMSE against number of trees. We can see that the rmse is very high when using only one feature. And the lowest rmse occurs when max_features = 3 and number of tree = 12, which means better performance.
The second figure is the out of bag error against number of trees. Similarly, the oob error is very high when using just one feature. And also, this model has better performance when max_feature = 3 and number of tree = 12.

iii. Find best max depth and maximum number of features
In this part, we experimented on the max_depth parameter in the RandomForestRegressor. We used the best number of tree (= 12) that we found out in question ii and then swept over max_depth from 1 to 20 and maximum number of features from 1 to 5. We plotted two figures as below:





The first figure is the average Test-RMSE against maximum depth. We can see that the average rmse decreases as max_depth increases, and then becomes steady when max_depth is larger than 9. For different number of max_features used, using 3, 4, 5 max_features have similar performances, which is better than using just 1 and 2 max_features.
The second figure is the out of bag error against maximum depth. Similarly, the oob error decreases as max_depth increases, and then becomes steady when max_depth reaches 9. Using 3, 4, 5 max_features have similar performances, which is better than using just 1 and 2 max_features.

Therefore in combination with question ii, we can conclude that when setting the following parameters, the random forest regressor achieves the best performance:
n_estimators (number of trees) = 12;
max_depth (maximum depth of the tree) = 9;
max_features (number of features for the best split) = 3.

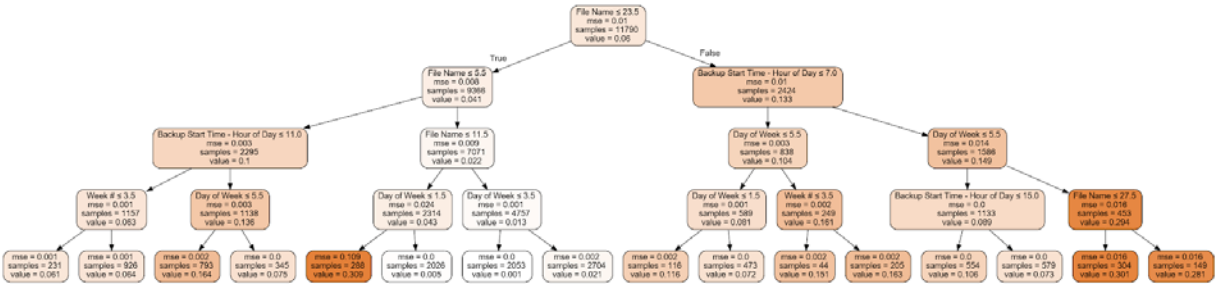
iv. Evaluate the feature importances
In this part, we evaluated the importance of each feature by ignoring this feature, and then calculate the average RMSE and Out of Bag error using 10-fold cross validation. If one certain feature is ignored in training and fitting and then the RMSE (or Out of Bag error) of predicted values increases, it means that this features is important in predicting the backup size. The

higher the rmse, more important is the feature. We used the parameters we found in previous parts (n_estimators = 12, max_depth = 9).
The results are as below:

	avg rmse	oob_error
Ignore 'Week #' only	0.06052898871973059	0.34337072235636745
Ignore 'Day of Week' only	0.08804281708277198	0.7065716330803488
Ignore 'Backup Start Time - Hour of Day' only	0.06926313724281526	0.44521301952386616
Ignore 'Work-Flow-ID' only	0.06052899257692829	0.3433697763782868
Ignore 'File Name' only	0.06049158785330684	0.34289249480739925

From the above rmse and oob error values, we can see that feature 'Day of Week' influenced the model prediction most, then is feature 'Backup Start Time - Hour of Day', the rest three has approximately the same influence on the model.
Thus the feature importances are:
Day of Week > Backup Start Time - Hour of Day > Work-Flow-ID > Week # > File Name

v. Visualize the decision trees
In this part we used the best parameters we found in the previous parts, which is number of trees = 12, max number of features = 3, and set the max depth = 4 as required. Then we randomly picked a trees in random forest. Below is the visualization of tree5.



Given a bunch of sample data points, at the first stage feature 'File Name' is chosen to split the data by condition 'File Name <= 23.5', and there are 9366 samples satisfying this condition and 2424 samples not satisfied. Among the 9366 samples, at the second stage feature 'File Name' is chosen again to split the data by 'File Name <= 5.5', data points are then split into two parts. And then it follows the same decision pattern. At the fourth stage, the data points are finally well-split with a relatively low mse value, and then the fitting process is completed.

From the structure of decision tree, we can see that the root node of this tree is ‘File Name’. Below is the importances of each feature. From the table we can see that the most important feature is ‘File Name’, which is same as root node.

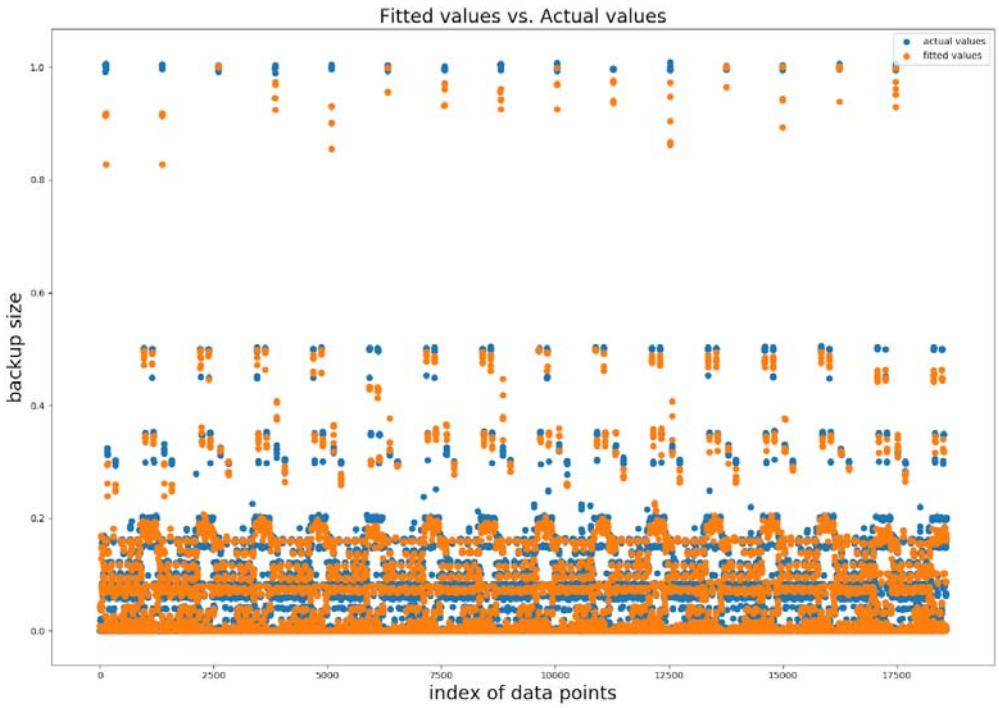
Feature Name	Feature Importance
Week #	0.00042164
Day of Week	0.35497636
Backup Start Time - Hour of Day	0.07424227
Work-Flow-ID	0.21378391
File Name	0.35657582

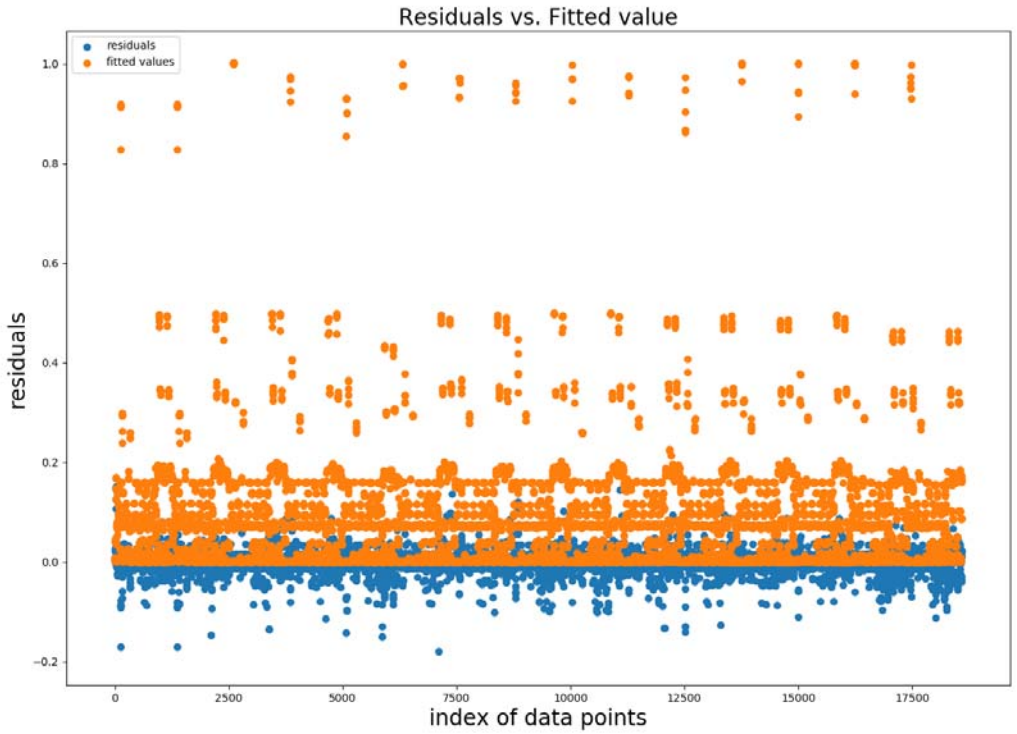
Visualize the random forest regressor model

In this part, we plotted two figures using the best parameters found in previous questions to visualize the random forest regressor model:

(i) Fitted values against true values scattered over the number of data points;

(ii) Residuals versus fitted values scattered over the number of data points using the whole dataset. Residuals means (actual values - fitted values).

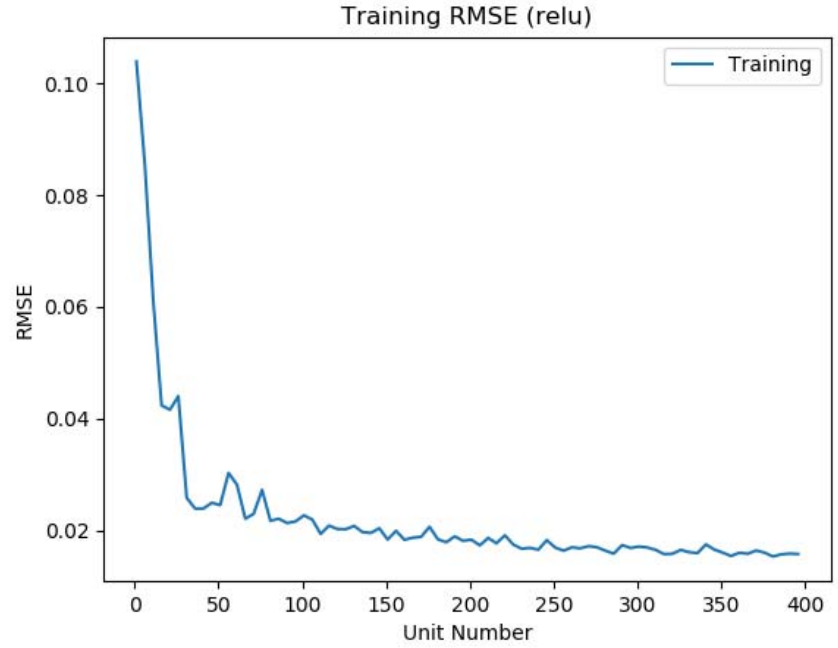


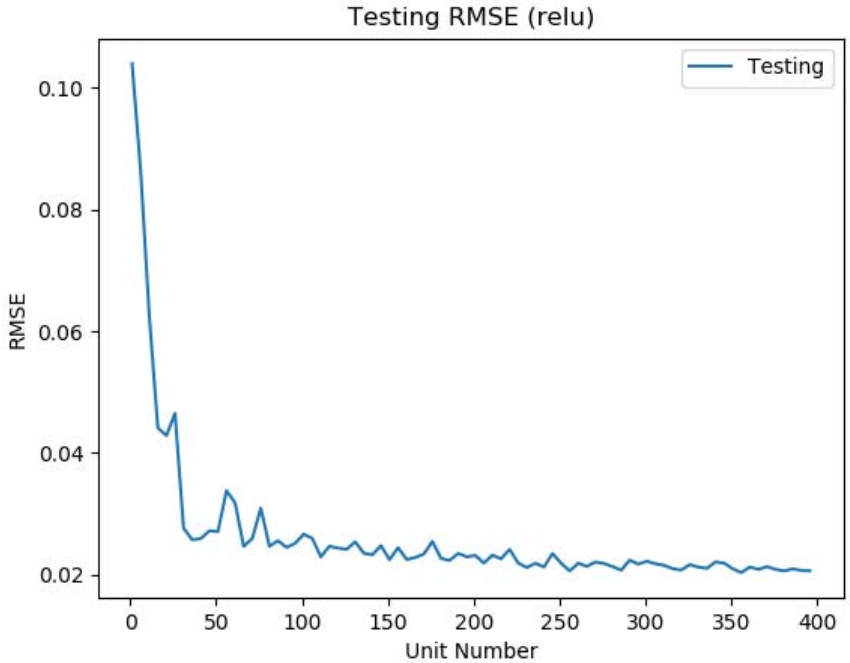


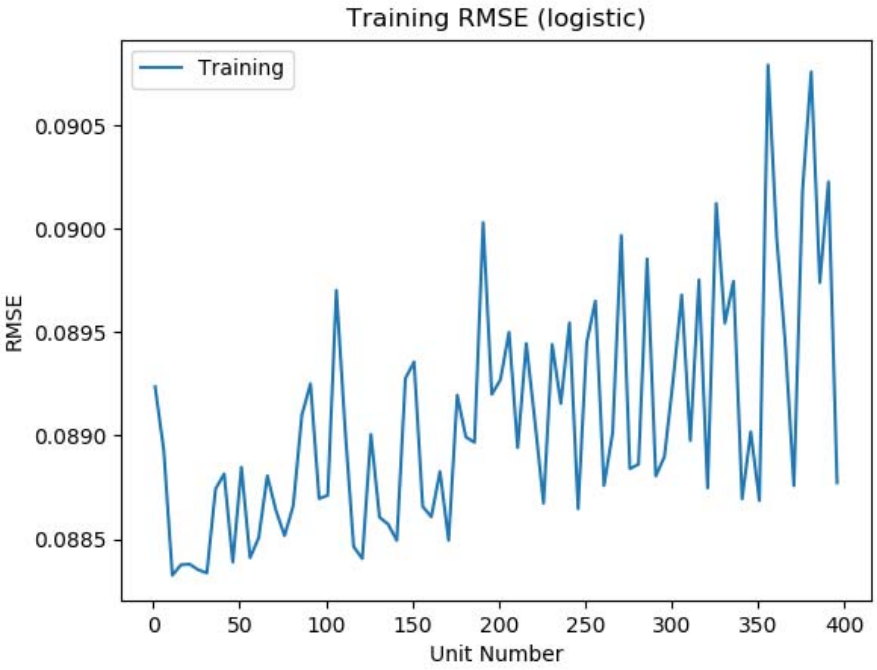
In the first plot, the X-axis is the index of all data points(from 1 to over 18000) and Y-axis is the backup size. Actual values are labeled as blue and fitted values are labeled as orange. From the plot we can see that fitted values and actual values are quite close, which means this model has good performance.
In the second plot, the X-axis is the index of all data points(from 1 to over 18000) and Y-axis is residuals, which are actual values minus fitted values. Residuals are labeled as blue and fitted values are labeled as orange. From the plot we can see that the majority of residuals are around 0, which means a good performance of the model.

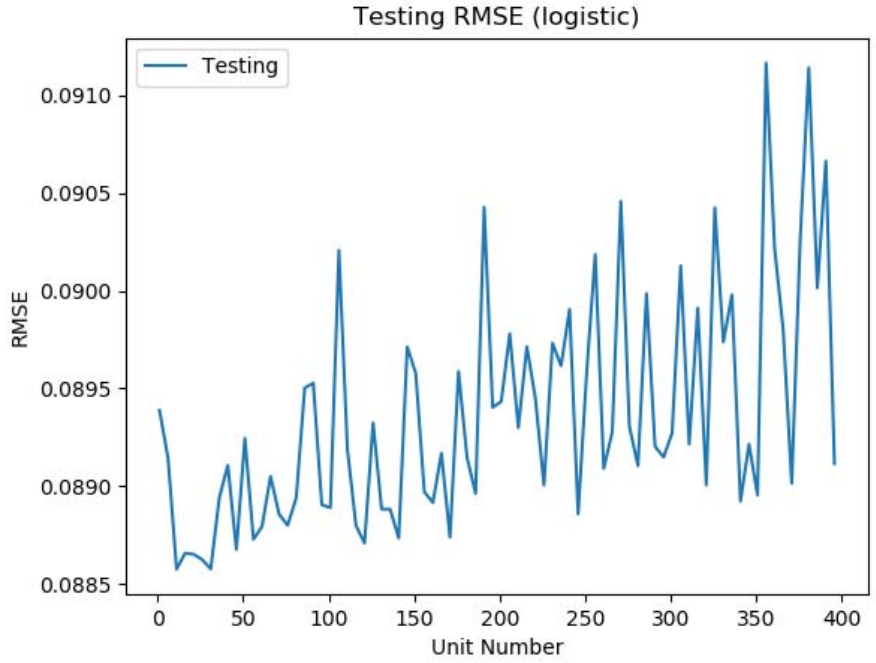
(c) Neural network regression model

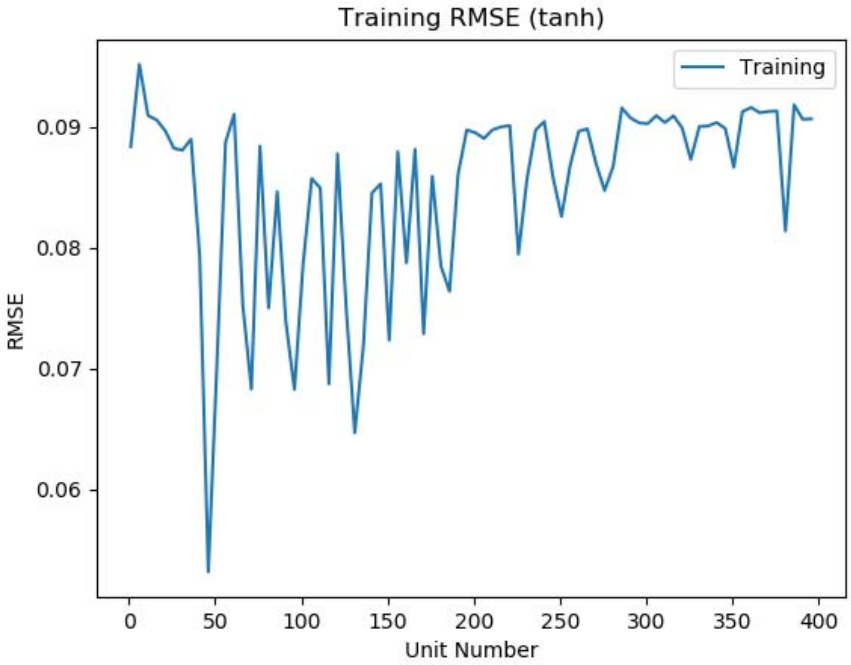
For this part, we use neural network regression to predict the backup size. We took advantage of the 'MLPRegressor' model in sklearn package to realize this function. In order to find the best prediction result, we need to train different models with different parameters. They include the number of hidden units and activity function. In our project, we swept the number of units from 1 to 400 with the step of 5. The activity function was selected from 'relu', 'logistic' and 'tanh'. Their training and testing RMSE results are plotted in the following figures. According to the testing RMSE results, we concluded that the neural network learner performs best when unit number is 356 and activity function is 'relu'. The minimum RMSE is 0.0203.

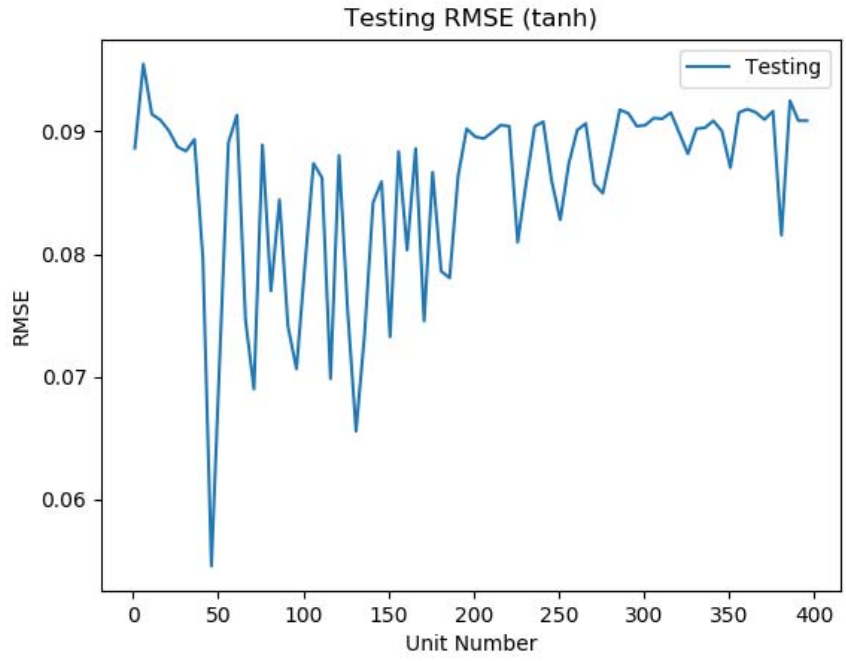

Training RMSE (relu)
Training
RMSE
0.10
0.08
0.06
0.04
0.02
0
50
100
150
200
250
300
350
400
Unit Number


Testing RMSE (relu)
Testing
RMSE
0.10
0.08
0.06
0.04
0.02
0
50
100
150
200
250
300
350
400
Unit Number


Training RMSE (logistic)
Training
RMSE
0.0905
0.0900
0.0895
0.0890
0.0885
0
50
100
150
200
250
300
350
400
Unit Number


Testing RMSE (logistic)
Testing
RMSE
0.0910
0.0905
0.0900
0.0895
0.0890
0.0885
0
50
100
150
200
250
300
350
400
Unit Number

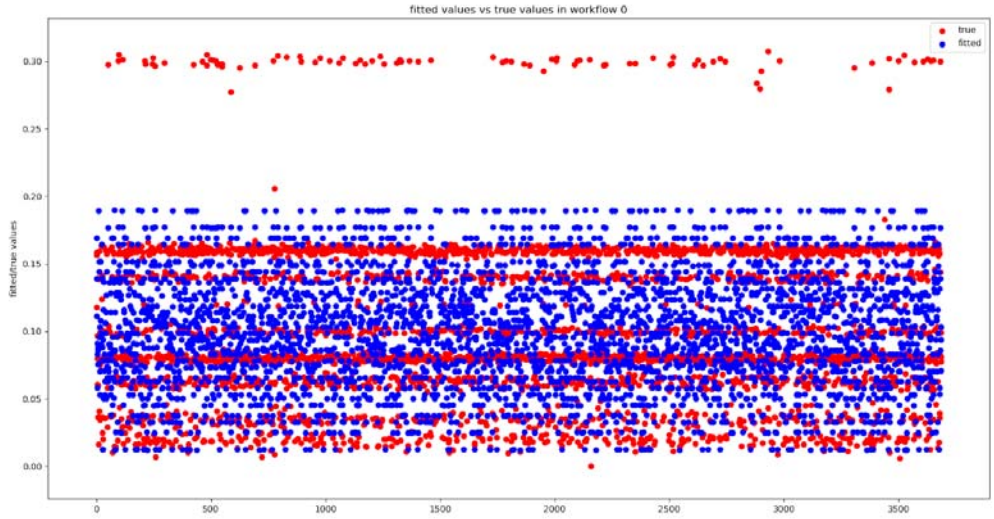


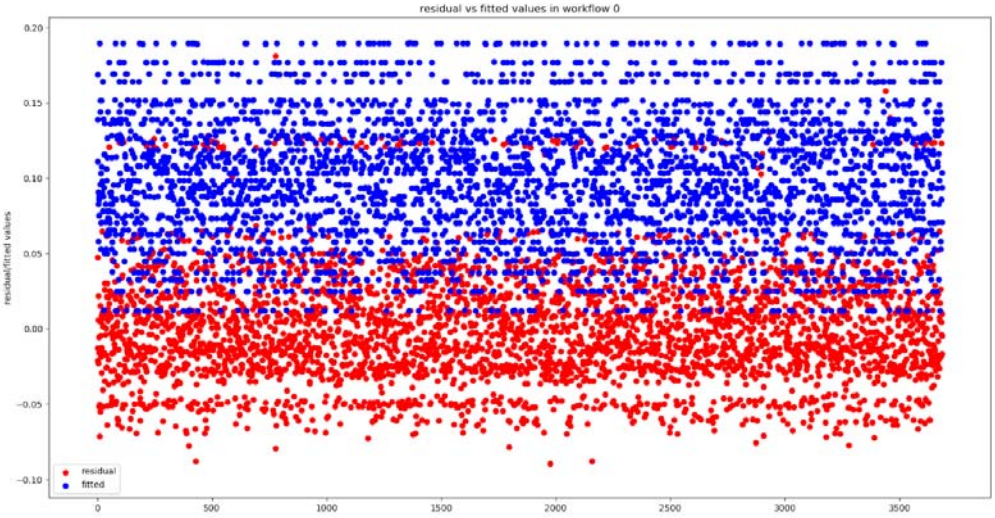


(d) Predict the backup size for each of the workflows separately

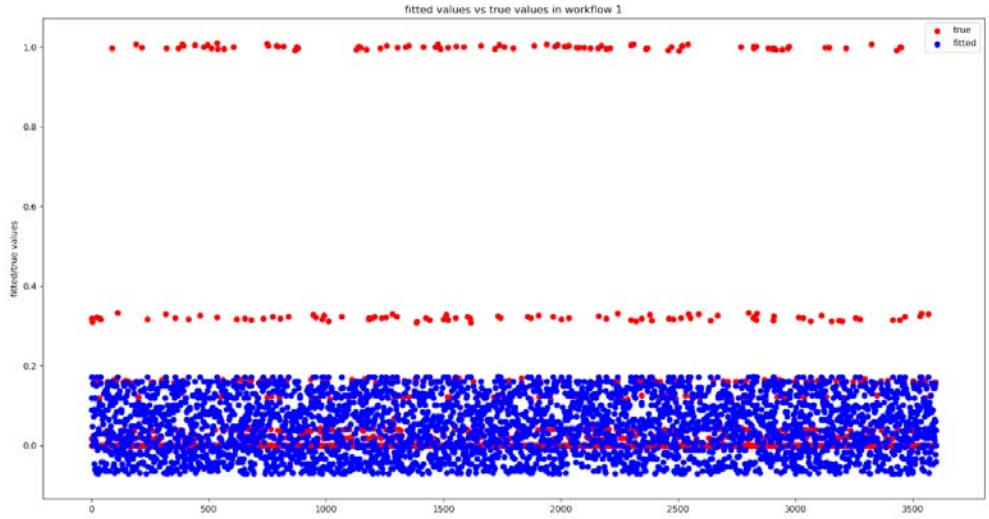
i. Using linear regression model

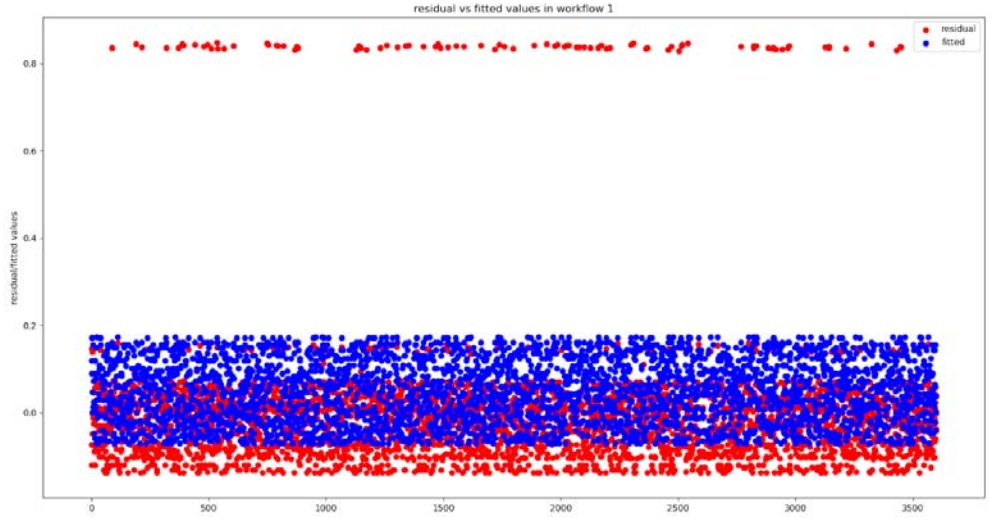
For workflow 0, using basic linear regression, the average training RMSE is 0.035834901179050554, average test RMSE is 0.03589927770403469.



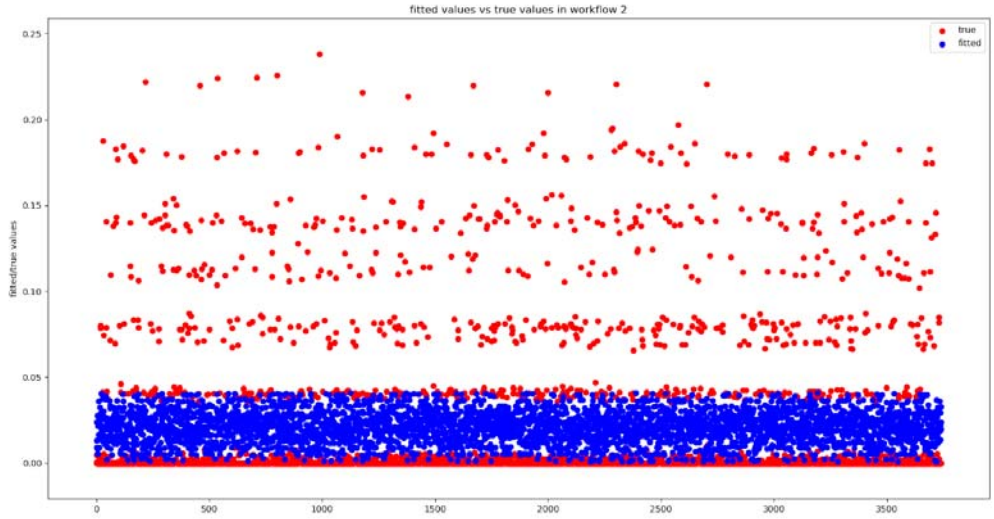


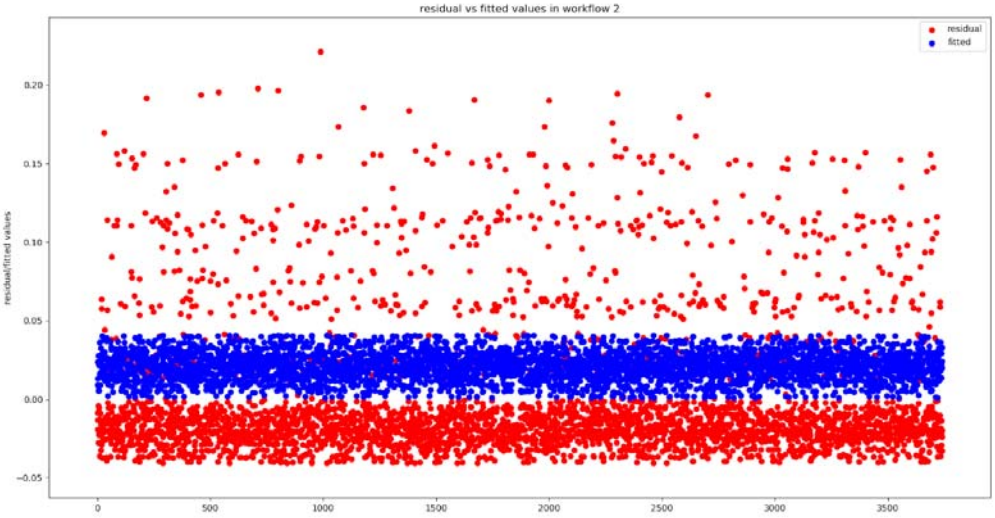
For workflow 1, using basic linear regression, the average training RMSE is 0.1487594946225272, average test RMSE is 0.14903824415933326.



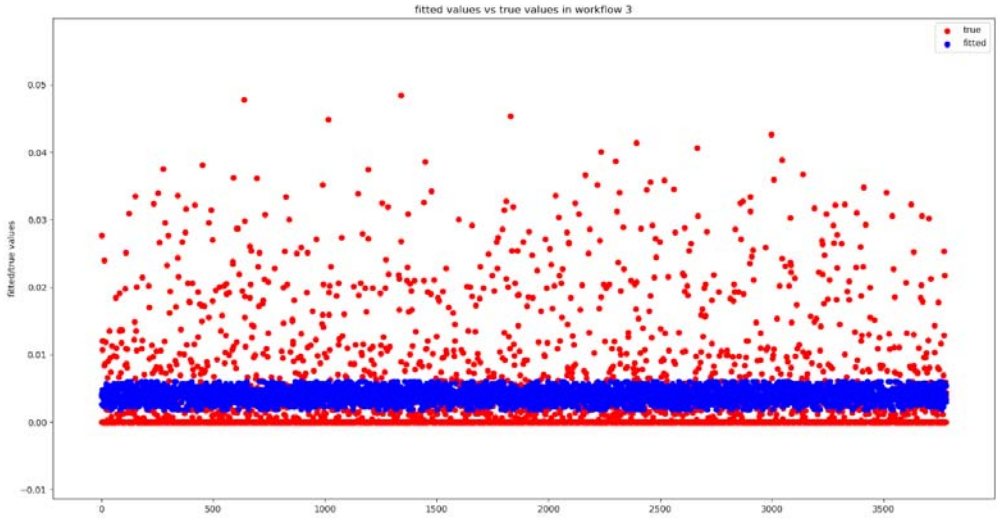


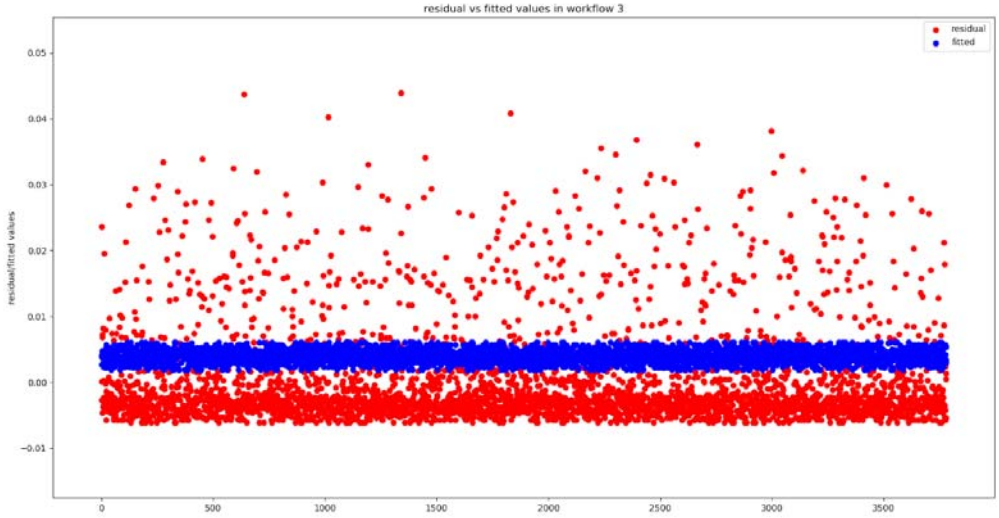
For workflow 2, using basic linear regression, the average training RMSE is 0.042914065189547004, average test RMSE is 0.042965667936516565.



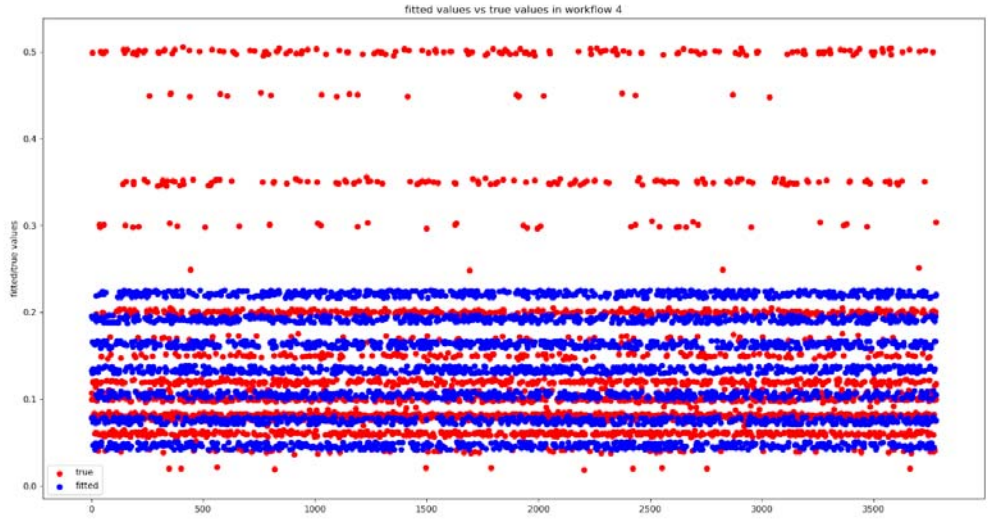


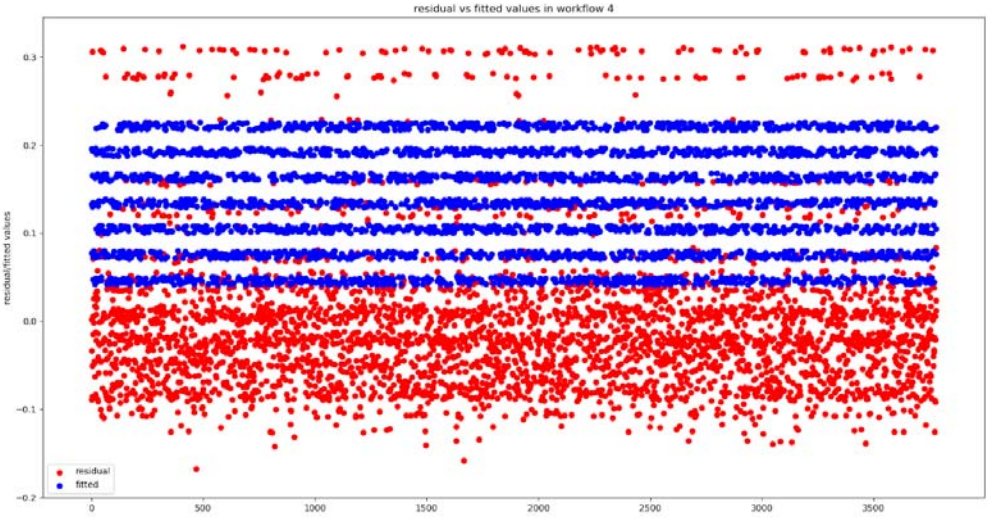
For workflow 3, using basic linear regression, the average training RMSE is 0.0072440909416008874, average test RMSE is 0.007255959565207277.





For workflow 4, using basic linear regression, the average training RMSE is 0.08592076058855774, average test RMSE is 0.08601209740162333.





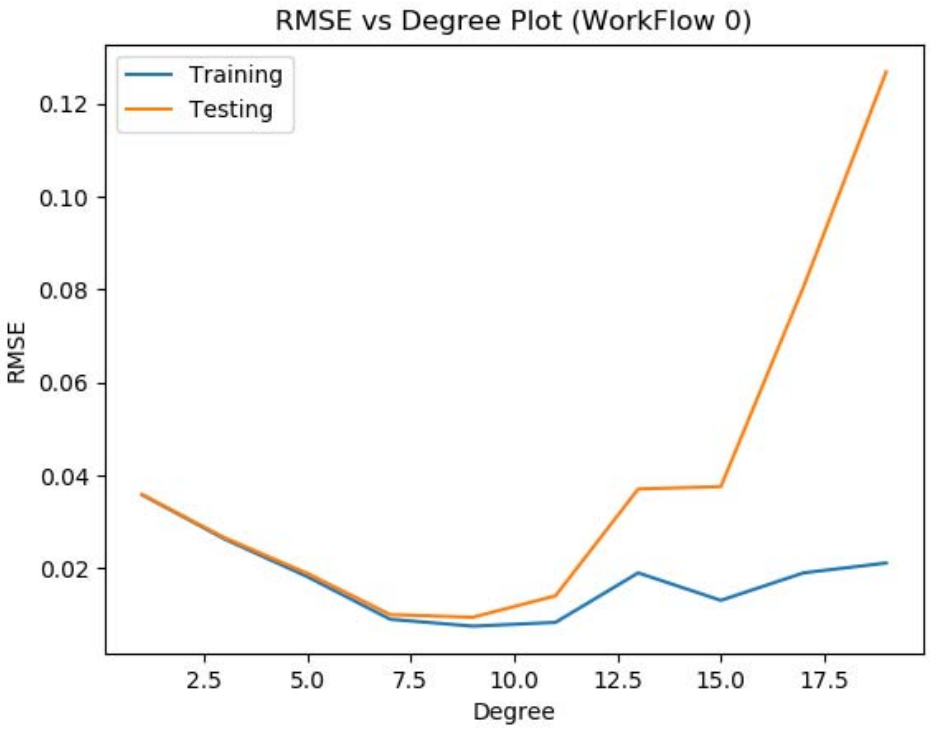
4 out of 5 workflow have been improved, the best test RMSE is as small as 0.007. This improvement shows that inside each workflow, the data are more linearly dependent thus we can get a better results.

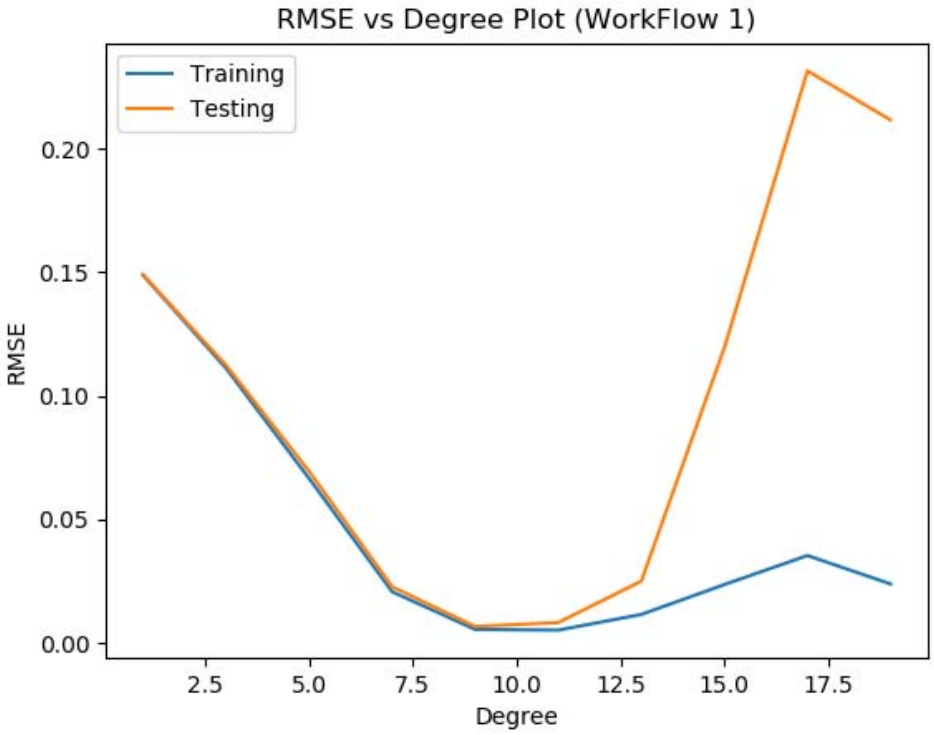
ii. Polynomial regression function

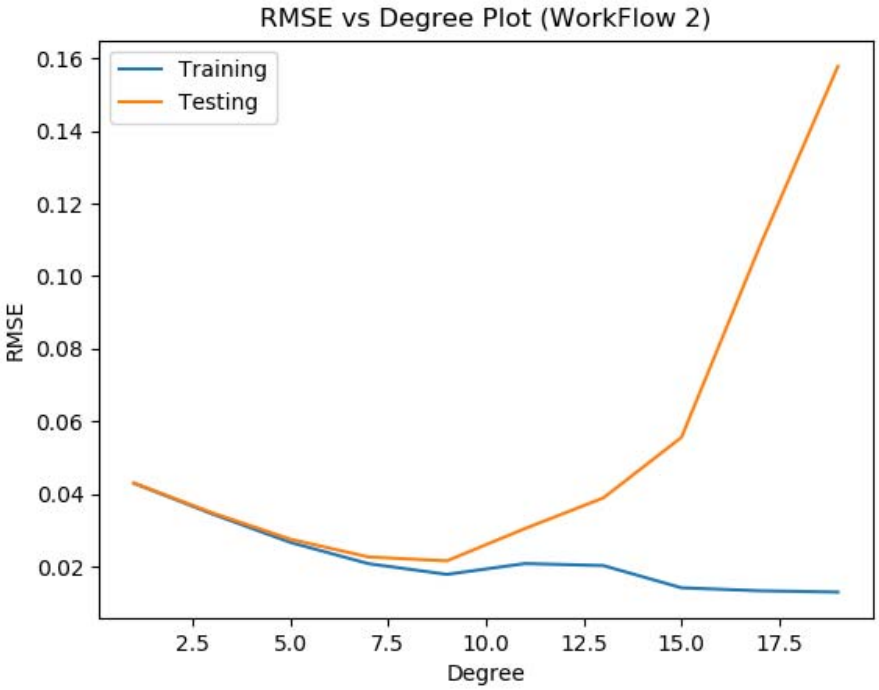
In this part, we used a more complex regression model than before. The 'PolynomialFeatures' method can generate a polynomial function with the assigned degree as the training model. The degree of the polynomial function in our experiment ranges from 1 to 21 with the step of 2. The training and testing RMSE plots for 5 workflows are shown in the following. It is notable that

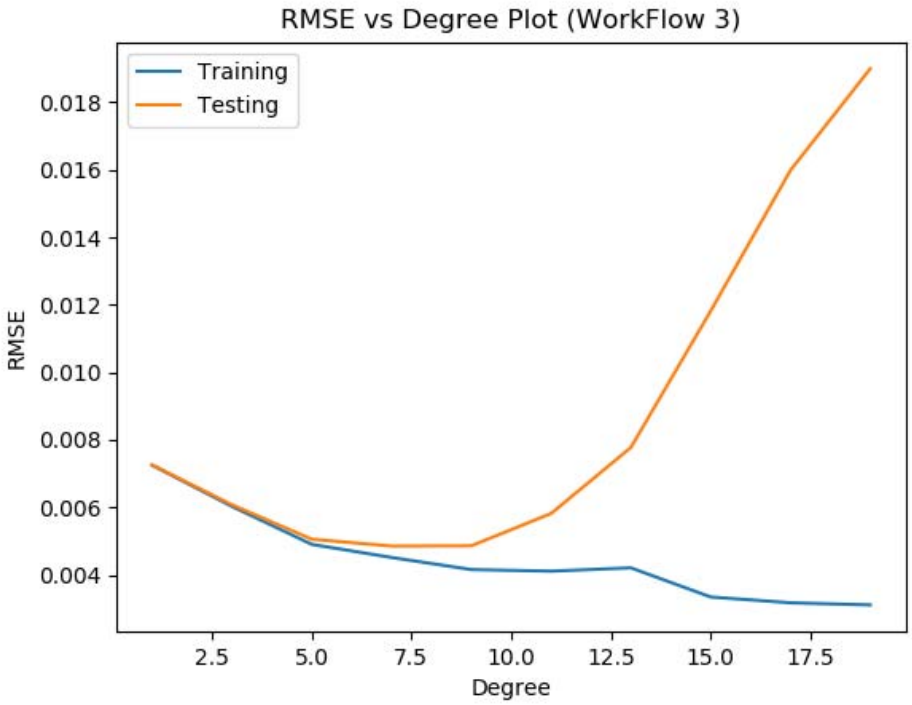
there exists a threshold in each workflow case. Before the threshold, the RMSE decreases while the model gets worse after the threshold. For workflow 0, 1, 2, 4, the threshold of degree is 9 and for workflow 3, it is 7.

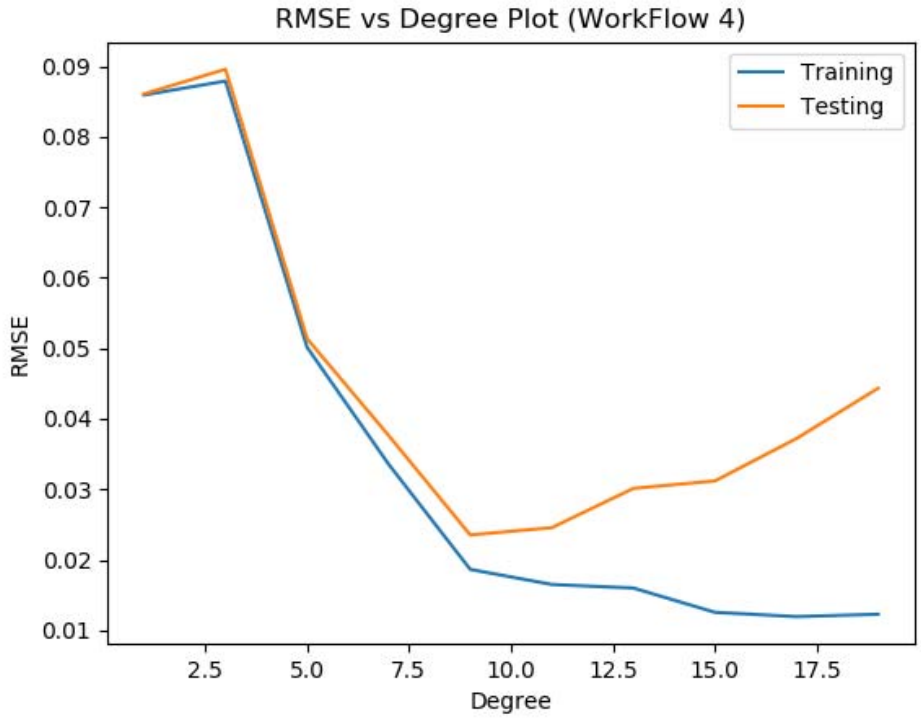
By using cross validation, the noise in the data was eliminated to some extent. Hence, the model is less likely to suffer from overfitting which would increase the function complexity. In this way, cross validation is very helpful in controlling the complexity of our model.




RMSE vs Degree Plot (WorkFlow 1)
Training
Testing
RMSE
0.00
0.05
0.10
0.15
0.20
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Degree

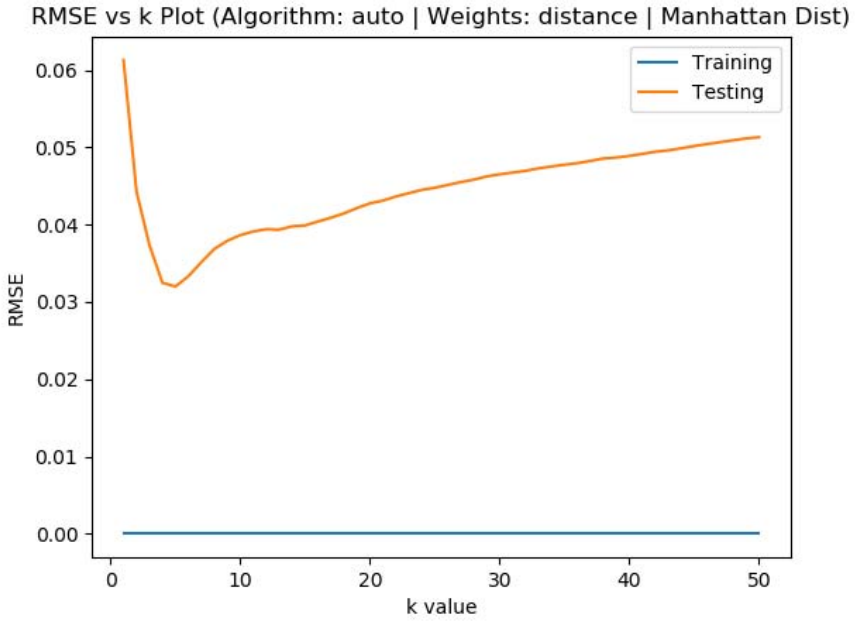

RMSE vs Degree Plot (WorkFlow 2)
Training
Testing
RMSE
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Degree

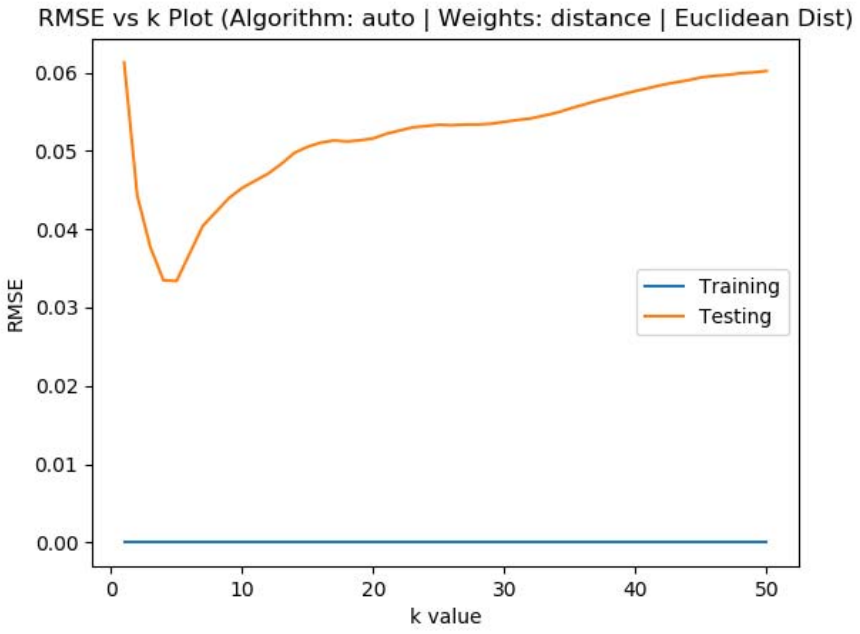


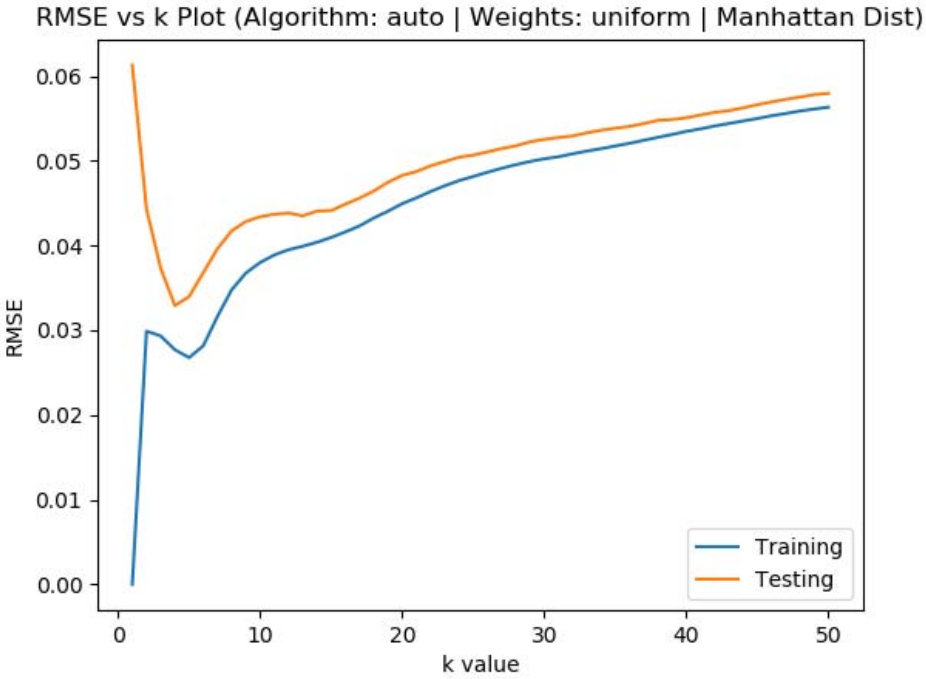


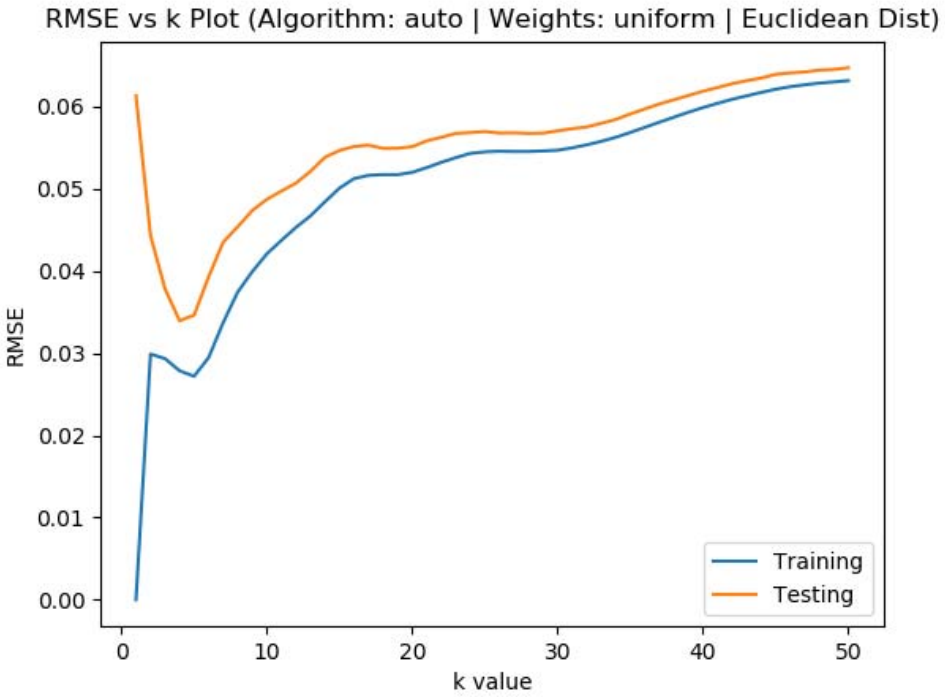
(e) K-nearest neighbor regression model

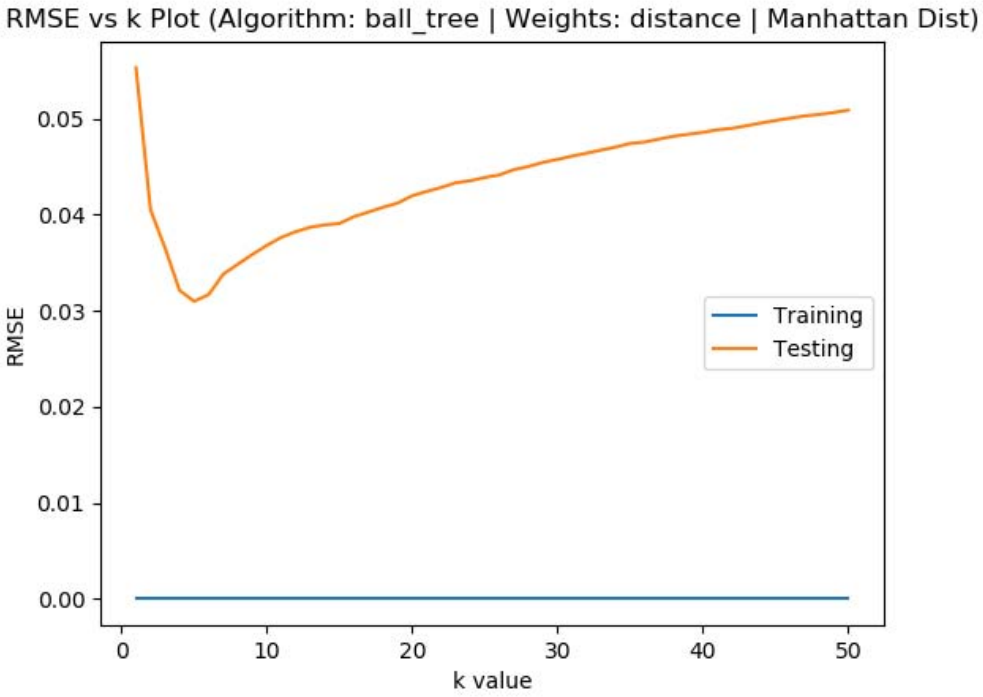
For the last model, we used K-nearest neighbor (KNN) regression model to make predictions. The corresponding function is 'KNeighborsRegressor' in sklearn. In order to find the best combination of parameters for this model, we selected 3 coefficients as our research objects. They are 'algorithm', 'weights' and 'p' parameters in the 'KNeighborsRegressor' model and their candidate values are separately ('ball_tree', 'kd_tree', 'brute', 'auto'), ('uniform', 'distance'), (1, 2). When p = 1, that means the model uses Manhattan distance as the distance metric and when p = 2, it uses Euclidean distance. Finally, we set k value from 1 to 50 and plotted the RMSE against k value for all these combinations.

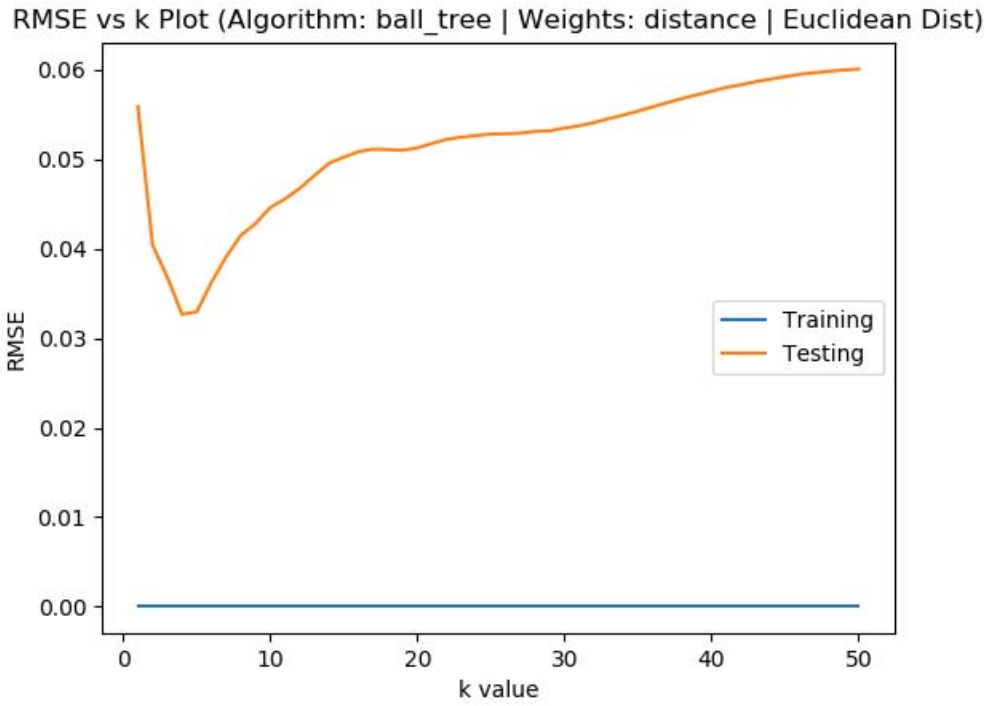


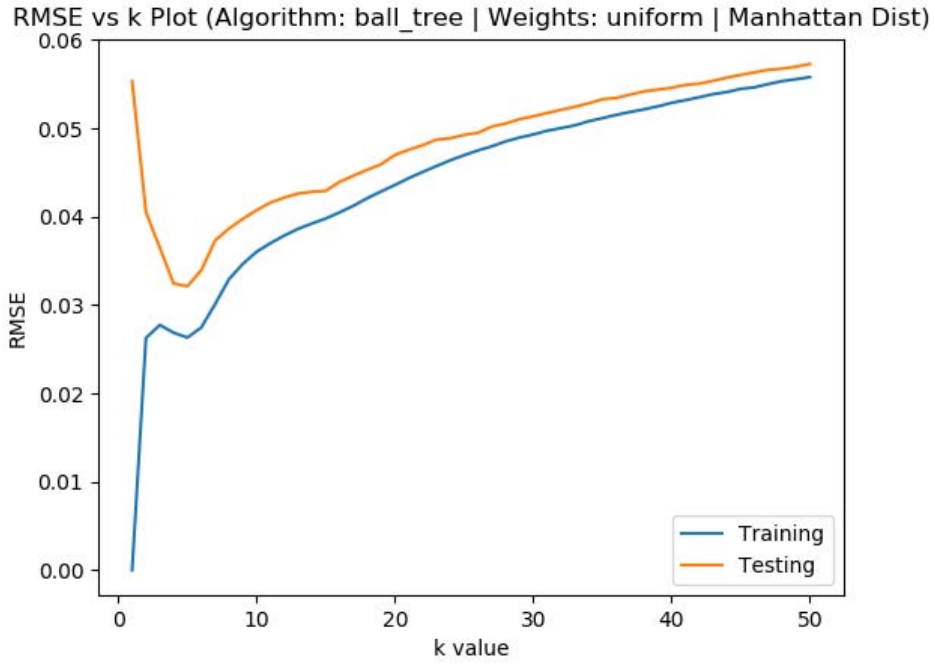

RMSE vs k Plot (Algorithm: auto | Weights: distance | Euclidean Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50

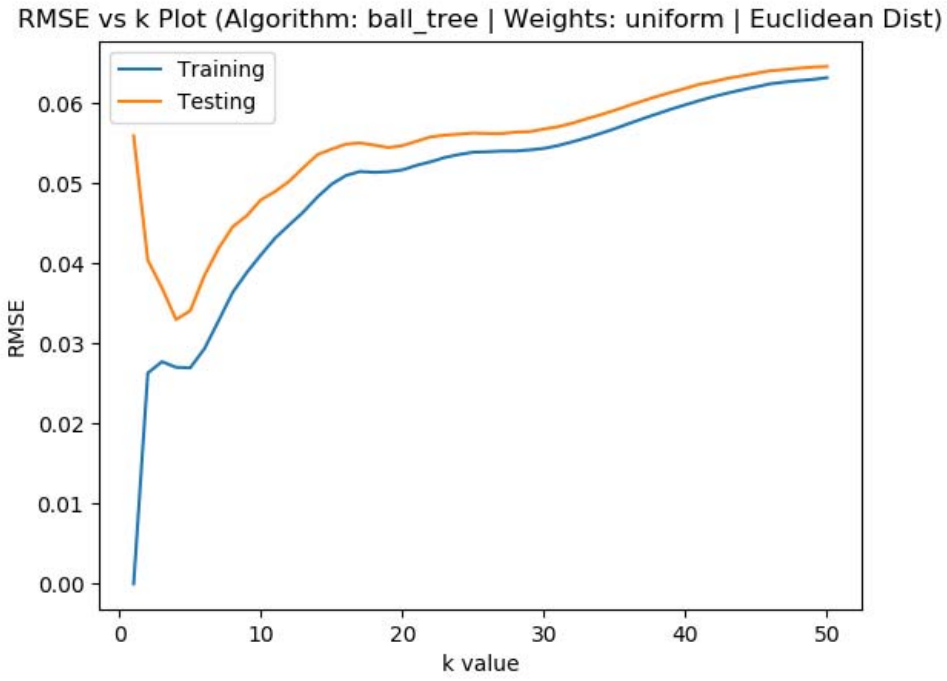

RMSE vs k Plot (Algorithm: auto | Weights: uniform | Manhattan Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50

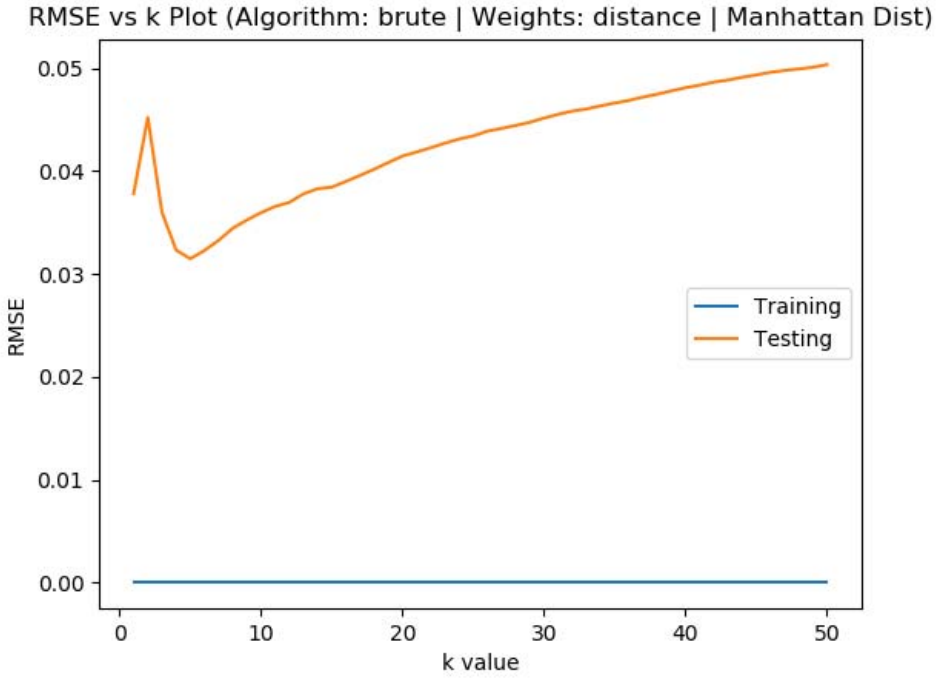

RMSE vs k Plot (Algorithm: auto | Weights: uniform | Euclidean Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50

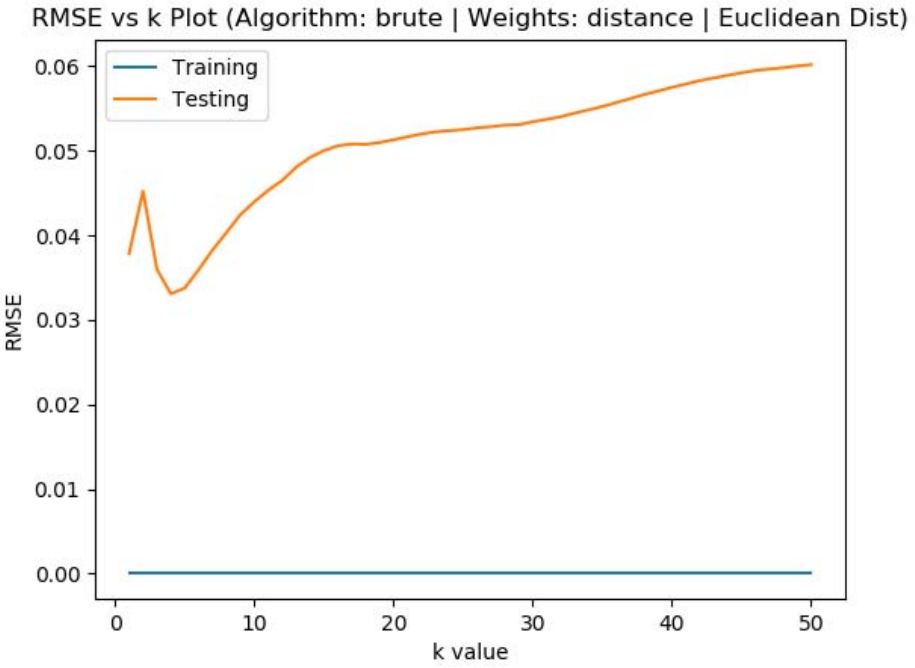

RMSE vs k Plot (Algorithm: ball_tree | Weights: distance | Manhattan Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0
10
20
30
40
50

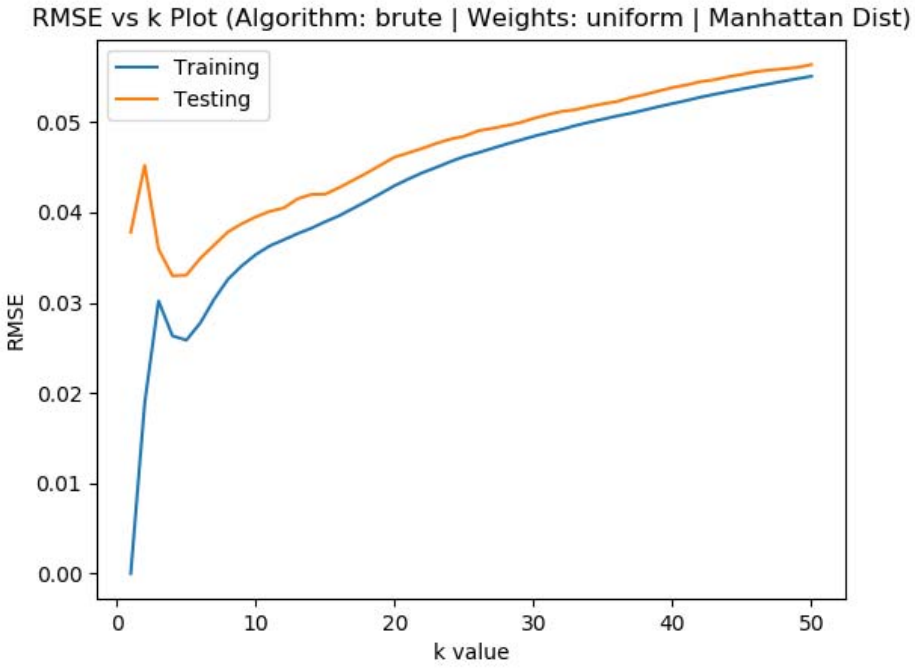

RMSE vs k Plot (Algorithm: ball_tree | Weights: distance | Euclidean Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50

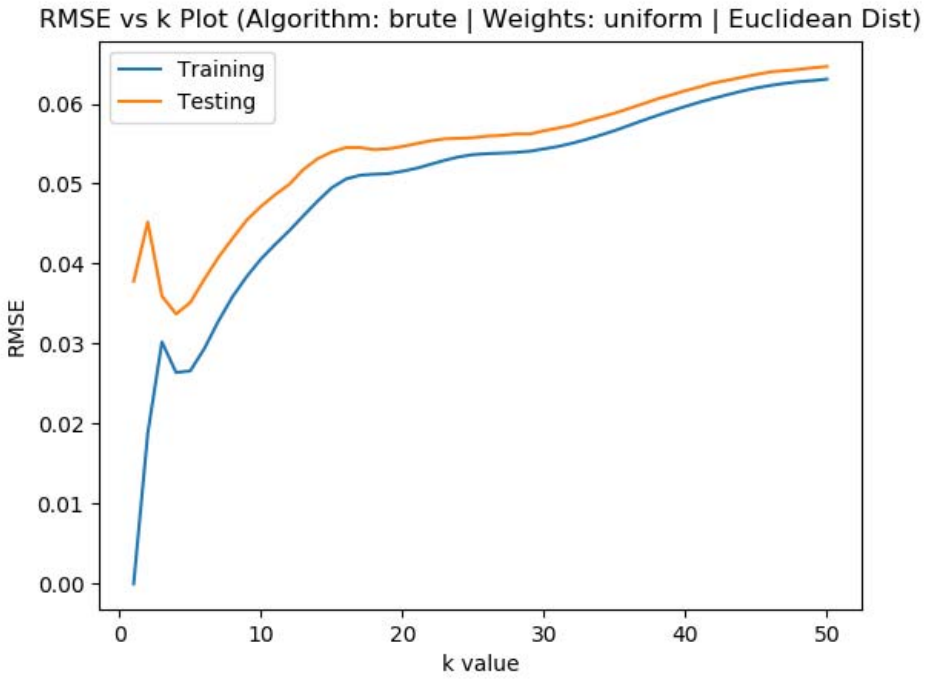

RMSE vs k Plot (Algorithm: ball_tree | Weights: uniform | Manhattan Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50

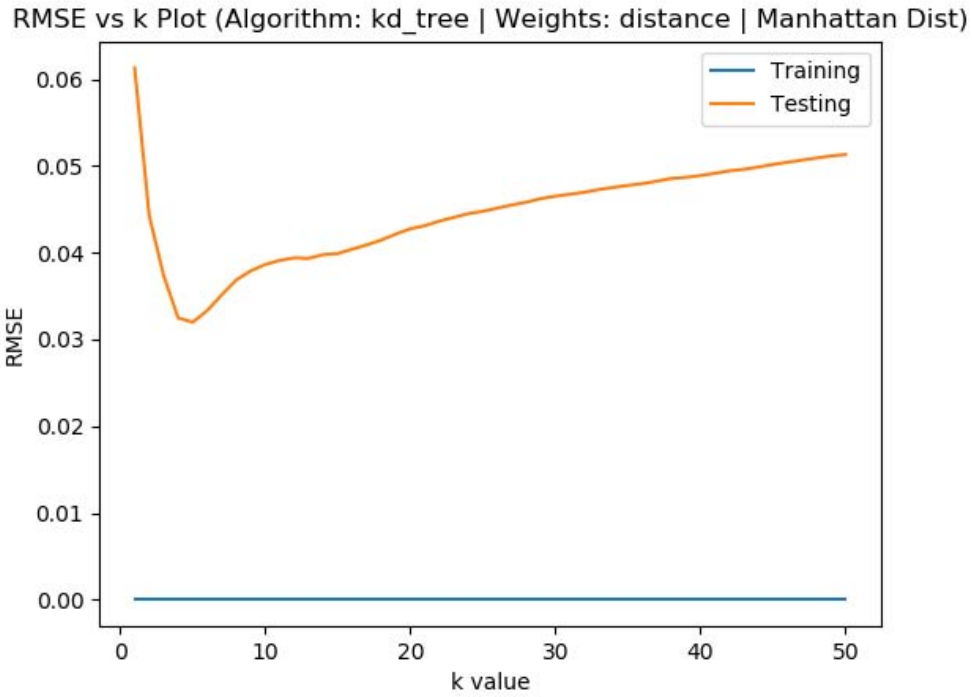

RMSE vs k Plot (Algorithm: ball_tree | Weights: uniform | Euclidean Dist)
Training
Testing
RMSE
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50
k value


RMSE vs k Plot (Algorithm: brute | Weights: distance | Manhattan Dist)
Training
Testing
RMSE
0.00
0.01
0.02
0.03
0.04
0.05
0
10
20
30
40
50
k value

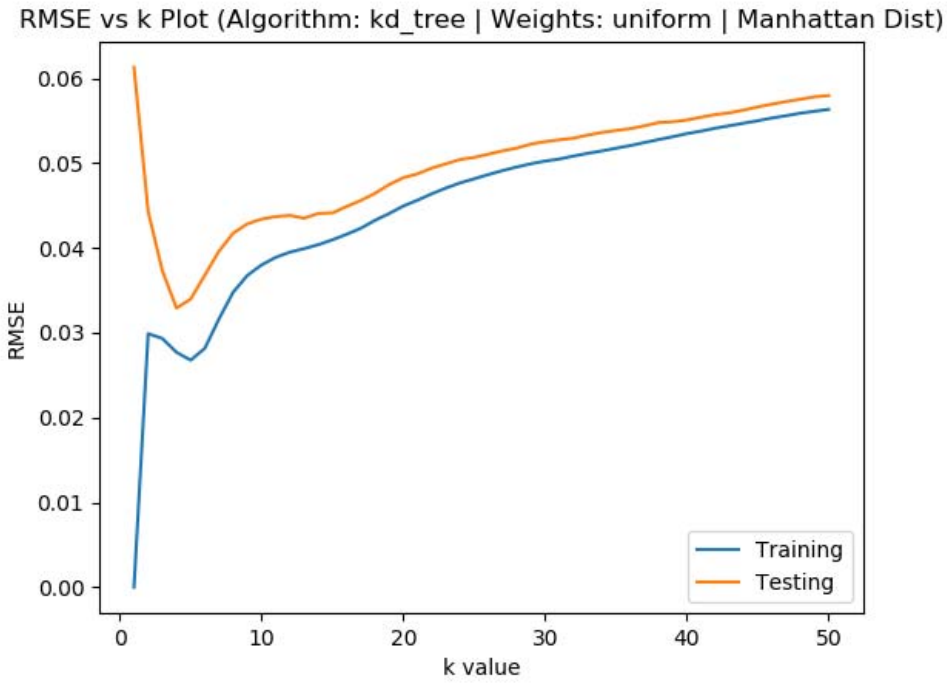

RMSE vs k Plot (Algorithm: brute | Weights: distance | Euclidean Dist)
Training
Testing
RMSE
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
k value

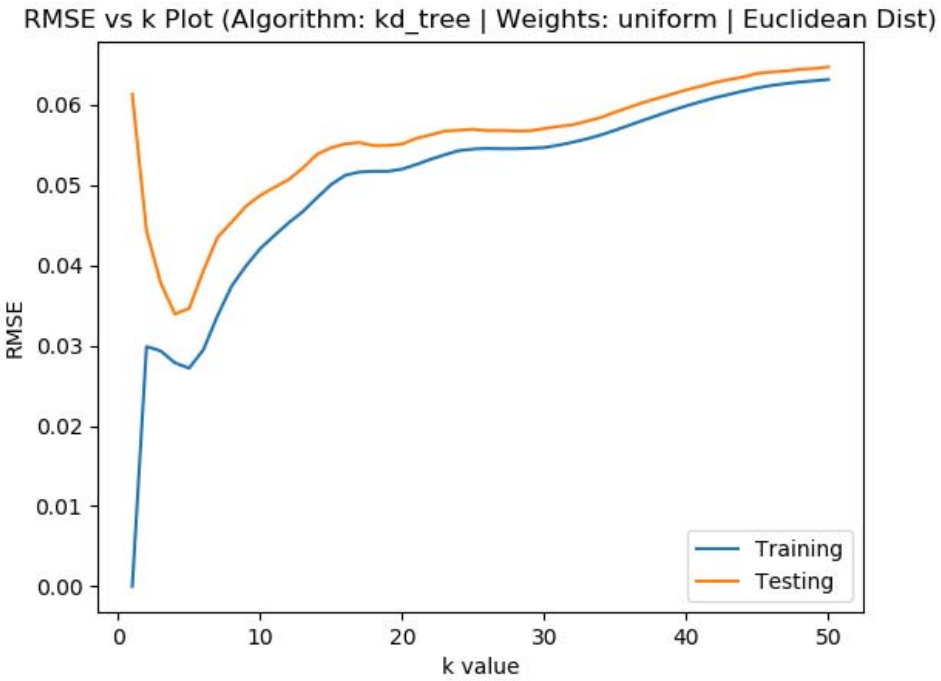

RMSE vs k Plot (Algorithm: brute | Weights: uniform | Manhattan Dist)
Training
Testing
RMSE
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
k value


RMSE vs k Plot (Algorithm: brute | Weights: uniform | Euclidean Dist)
Training
Testing
RMSE
k value
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50


RMSE vs k Plot (Algorithm: kd_tree | Weights: distance | Manhattan Dist)
Training
Testing
RMSE
k value
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50


RMSE vs k Plot (Algorithm: kd_tree | Weights: distance | Euclidean Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50


RMSE vs k Plot (Algorithm: kd_tree | Weights: uniform | Manhattan Dist)
RMSE
k value
Training
Testing
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
10
20
30
40
50



According to the above plots, we found that for all the combinations, the testing RMSE always decreases first and then increases after a specific k value. The best RMSE is 0.03099 when 'algorithm' = 'ball_tree', 'weights' = 'distance', 'p' = 1(Manhattan distance) and k = 5.

3. Compare regression models

(1) Linear Regression Model:
Best test RMSE is 0.08850347786271374.
Its performance is the worst in terms of test RMSE in all models, but it performs fastest and took much less time than other models.

(2) Random Forest Regressor Model:
avg rmse with best parameters is 0.014920238583627442.
From the results we can see that random forest regressor generates best overall results among these regression models, and is best at handling categorical features. It can handle high-dimensional data with many features, and do not have to select features. It also has some advantages: 1) OOB (out of bag) data can be used to evaluate the error rate of the algorithm; 2) The importance of each feature can be calculated using OOB and permutation test.

(3) Neural Network Regression Model:
Its best RMSE is 0.02031. From this result, we can find that neural network model is very good at dealing with sparse features because the neural network model mainly handled the sparse matrix encoded by One-Hot-Encoding in our project and it performed well with just one hidden layer. Among these models, only random forest regression model's result is better than neural

network model. Thus, we can say the neural network model is relatively great and reliable to complete regression analysis.

(4) Polynomial Function Regression Model:
This model is used to make predictions for the backup size of each workflow. Its performance is much better than that of linear regression. According to the test results, we know that for polynomial model, we need to select the correct threshold as the degree of the model to achieve a great performance. It is more suitable for handling numerical features.

(5) K-Nearest Neighbor Regression Model:
The best RMSE is 0.03099 for this model. Its performance does not outstand among all the models. But if we can select the suitable k value and other parameters for this model, it can enjoy a pretty satisfying result. It is also more suitable for handling numerical features instead of sparse features.