EE219 Project4

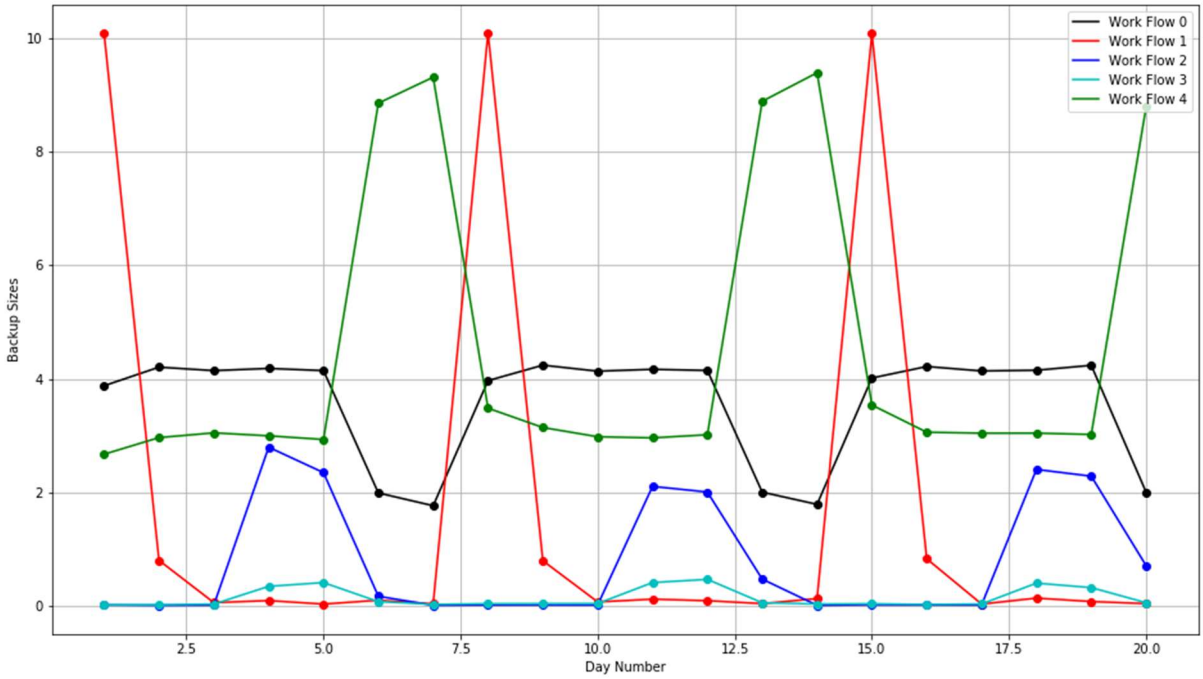
Yuan Tao (305033824), Yi Jia (805033204), Shuojian Ye (904946811), Fangjia Zhu (905036438)
March 3rd, 2018

Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, we explore basic regression models on a given dataset, along with basic techniques to handle overfitting; namely cross-validation, and regularization. With cross-validation, we test for overfitting, while with regularization we penalize overly complex models.

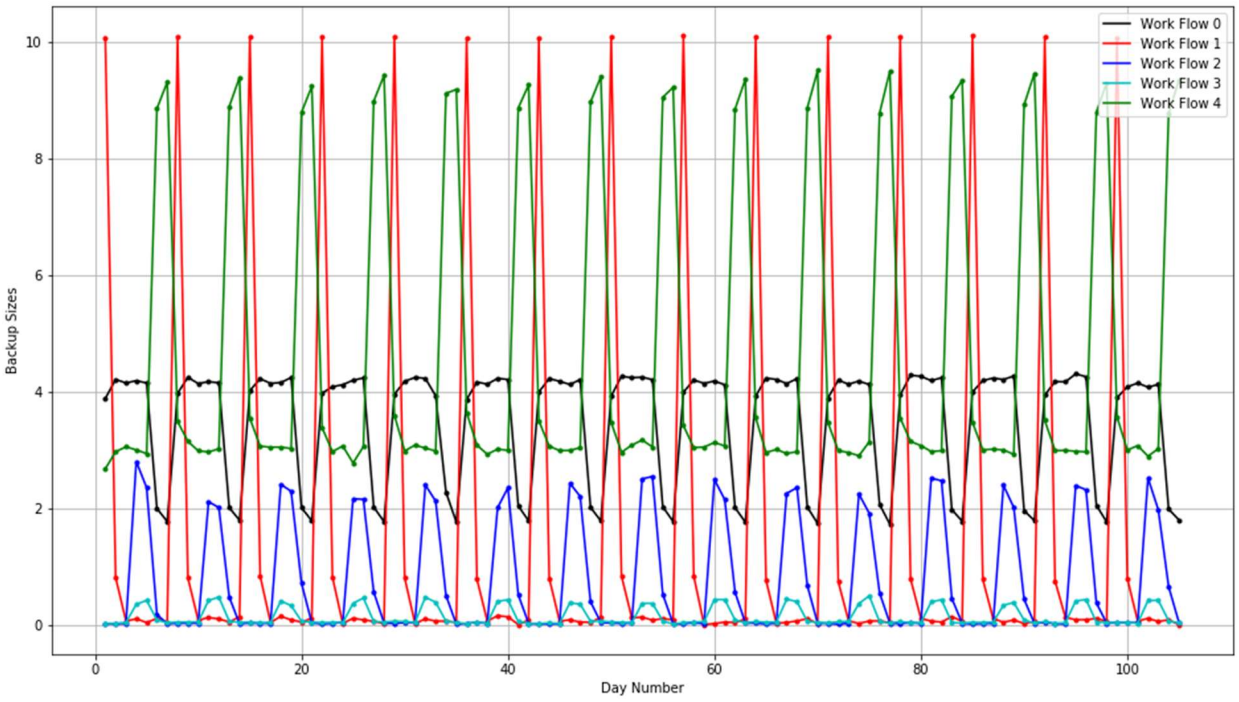
Problem 1

The first step we need to do is to load the dataset and analyze properties of the dataset.
(a) For a twenty-day period (X-axis unit is day number) plot the backup sizes for all workflows (color coded on the Y-axis).



In this part, we found that there are regular patterns with period of 7 days for each workflow. For example, in workflow 0, there is drop in 5.0-7.5, 12.5-15 days, while workflow 0 has the peak at the same time period.

(b) Do the same plot for the first 105-day period.



Can you identify any repeating patterns?

For the 105 days plot, we can see more clearly from the two plots that each workflow is periodic with period of 7 days. However, the amount of change in backup size varies a lot for each workflow. For instance, the workflow 1 and workflow 4 has obviously large change compare to the rest of the workflows, especially to the workflow 3. This means if we trained all workflows together, the model we get would not fit all the values.

Problem 2

Predict the backup size of a file given the other attributes. We use all attributes, except Backup time, as candidate features for the prediction of backup size.

(a) Fit a linear regression model.

i. First convert each categorical feature into one dimensional numerical values using scalar encoding and then directly use them to fit a basic linear regression model.

The training and test RMSE for 10-fold cross validation are listed as follows.

Train RMSE = 0.101834358198

Test RMSE = 0.101939446242

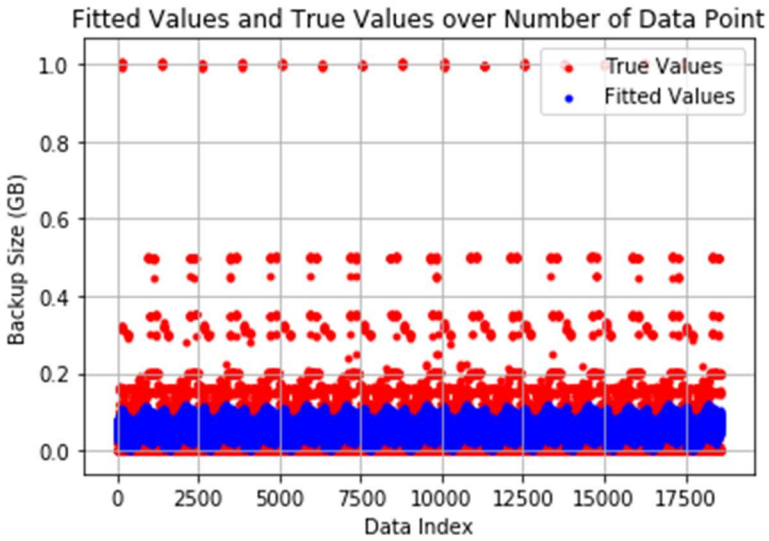


Figure. 2(a)i.1 Fitted values vs true values

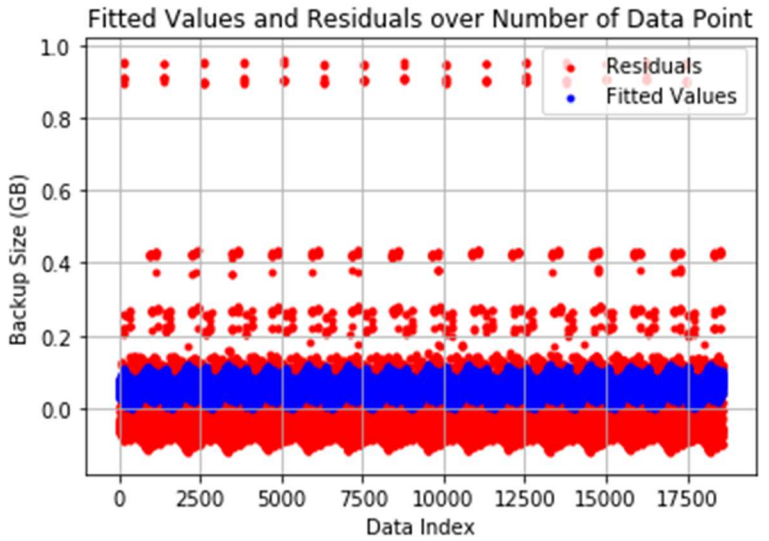


Figure. 2(a)i.2 Fitted values vs residuals

Analysis:

We can see from the 10-fold training and testing RMSE that RMSE is around 0.1. From the figures, we can see that the fitted values of backup size is around 0.0 to 0.15 to for all data point size while the true values range from 0 to 1. Additionally, from the figure of fitted values vs residuals, we can see residuals are around 0 to 1, which is similar to true values. It means that the residuals are similar to true values and the fitted values predicted badly on the dataset.

ii. We applied standardization as data preprocessing to all these numerical features, then fit and test the model.

The training and test RMSE for 10-fold cross validation are listed as follows.

Train RMSE = 0.101834358198
Test RMSE = 0.101939446242

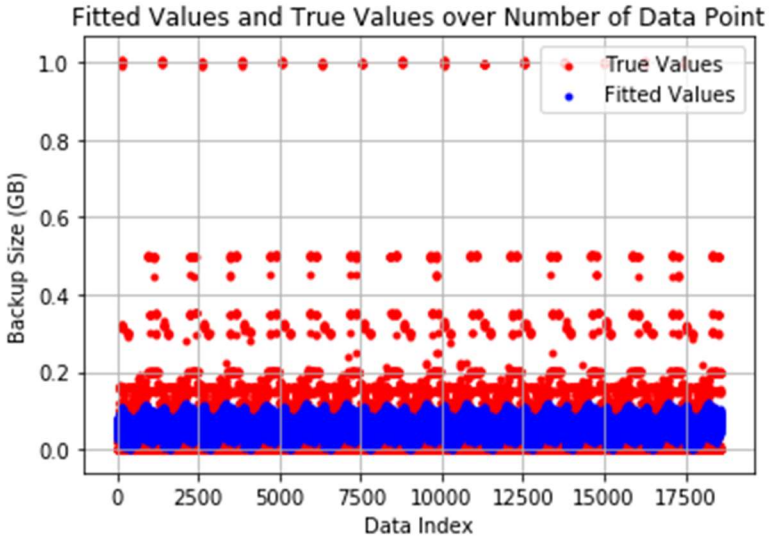


Figure. 2(a)ii.1 Fitted values vs true values

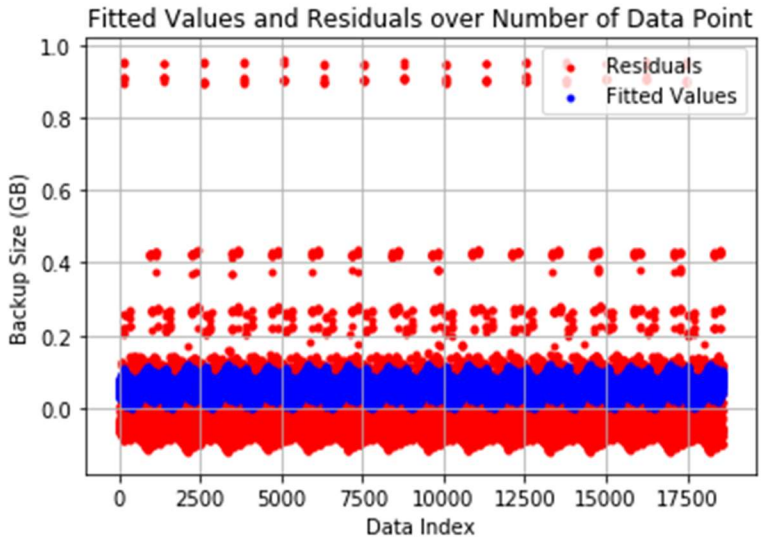


Figure. 2(a)ii.2 Fitted values vs residuals

Analysis:

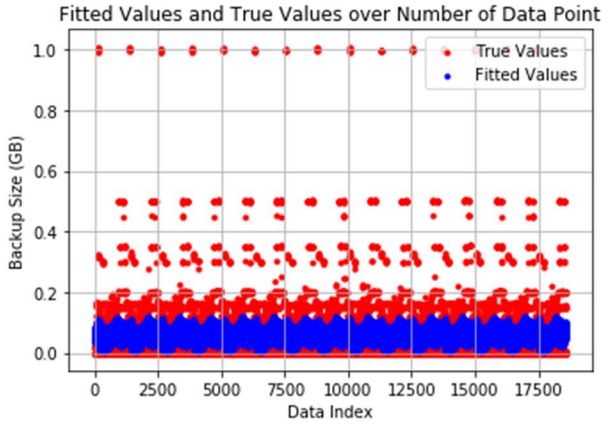
We can see that the predicted results after data standardization are the same as problem 2(a) i. Suppose that for model in question 2(a)(i) is $y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$. After normalization, the features will become like this:

$x' = [x_1' = (x_1 - \mu_1)/\sigma_1, \ldots, x_n' = (x_n - \mu\_n)/\sigma_n]$ Then we can have a new linear model that have $\theta' = [\theta_0', \theta_1', \ldots, \theta_n']$ on the new features, with $\theta_i' = \sigma_i \theta_i$ for $i = 1, \ldots, n$ and $\theta_0' = \theta_0 + \sum_{i=1}^{n} \mu_i \theta$ In this way, we can have exactly the same model and the accuracy will not change.

iii. Feature Selection: Use f regression and mutual information regression measure to select three most important variables respectively.

1). The 3 most important features found by f_regression are "Day of Week", "Backup Start Time - Hour of Day", "File Name". Using these 3 most important features from f_regression to do linear regression, we have 10-fold cross validation results as follows:

Train RMSE = 0.101871974794
Test RMSE = 0.101893426765



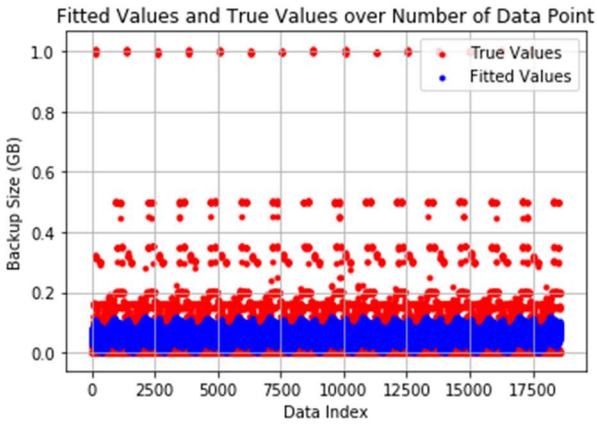
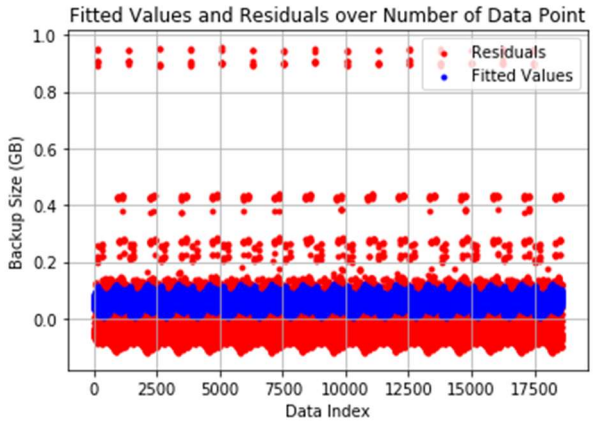


Figure. 2(a)iii.1 Fitted values vs true values i Fitted values vs true values



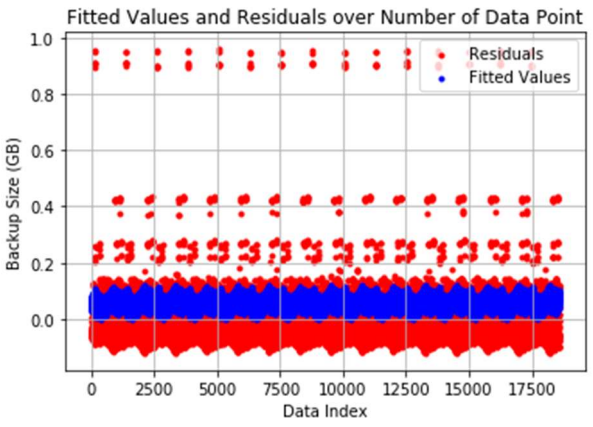


Figure. 2(a)iii.2 Fitted values vs residuals i Fitted values vs residuals

Analysis:
We can see from the figure of fitted values vs true values, the result in problem 2(a) iii covers more true values than result in problem 2(a) i. What's more, the average test RMSE in problem 2(a) i is 0.1019 while that after applying f_regression is 0.1010. In this way, the model

performance is improved after applying 3 most important features by f regression but the model is still not very good.

2). The 3 most important features found by mutual information regression are
"Backup Start Time - Hour of Day", "Work-Flow-ID", "File Name"

Using these 3 most important features from mutual information regression to do linear regression, we have 10-fold cross validation results as follows.

Train RMSE = 0.102465412867
Test RMSE = 0.102547286771

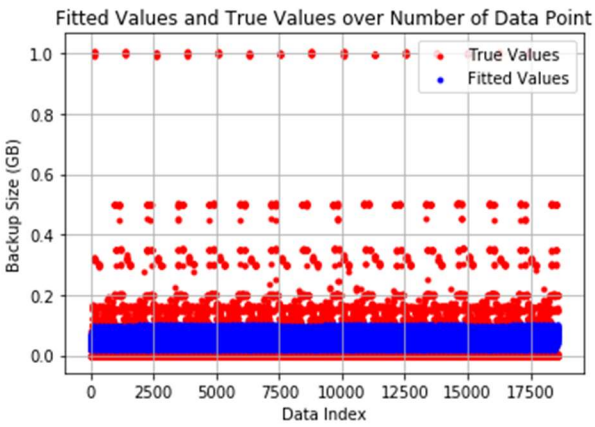
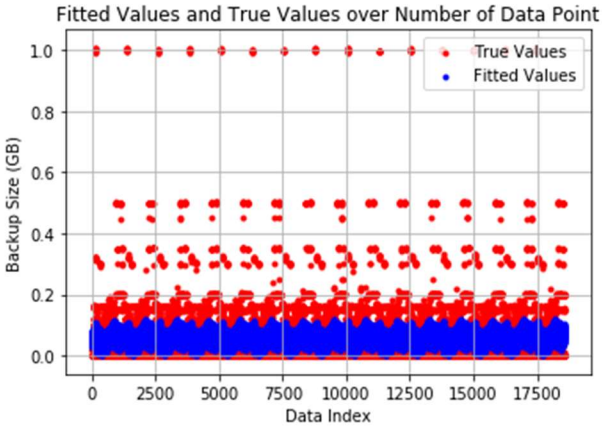


Figure. 2(a)iii.3 Fitted values vs true values



i Fitted values vs true values

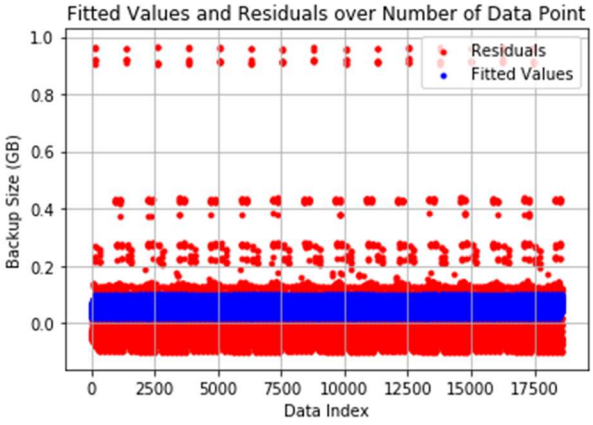
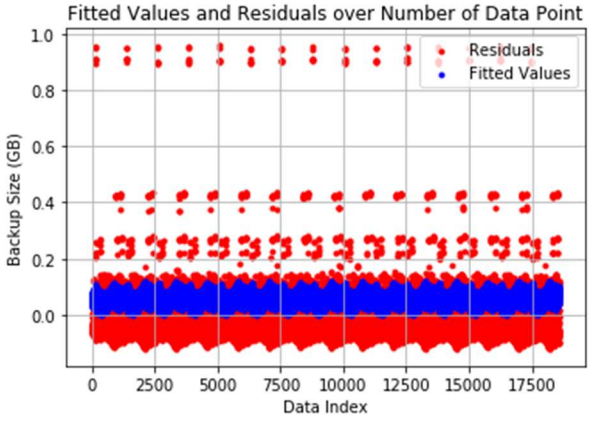


Figure. 2(a)iii.4 Fitted values vs residuals



i Fitted values vs residuals

Analysis:
We can see from the figure of fitted values vs true values, the result in problem 2(a) iii covers more true values than result in problem 2(a) i. What's more, the average test RMSE in problem 2(a) i is **0.1019** while that after applying mutual information regression is **0.1025**. And in

general, all the models above seem to be having very similar training and testing RMSEs while differ in the patterns in fitted values against data index, although intuitively, using most important features would decrease the error. One of the explanations, which can be tell from the above figures, is that the nature of the backup size data is non-linear and 'spiky' with most of the data within a certain range while some being relatively large. In this case, a simple linear regression model should not be expected to perform very good.

iv. Feature Encoding

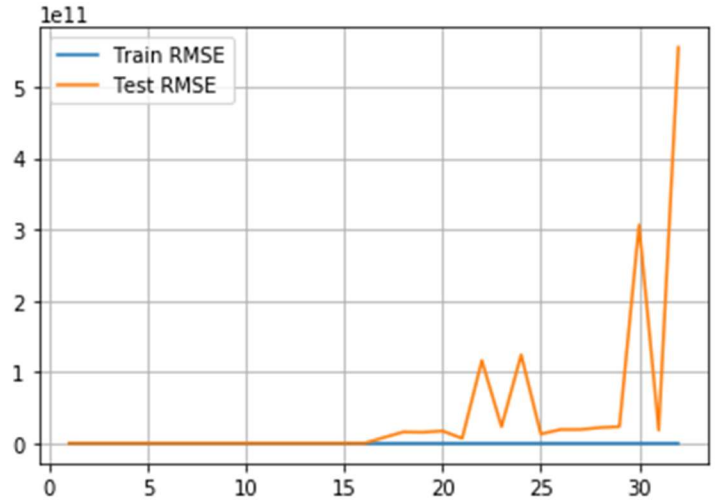


Figure 2(a).iv train RMSE, test RMSE vs encoding combination

For this problem, we used binary representation to aid the enumeration of all possible scalar/one-hot encoding combinations, for example, '01111' means the first feature(week number) is scalar encoded while all the rest features are one-hot encoded. We can see from the plot, the optimal combination is as follows:

Optimal combination = [0 1 1 1 0]
Train RMSE = 0.0883374686358
Test RMSE = 0.0885042308482

It means that for 2nd to 4th features, we use one hot encoding while for others we use scalar encoding. It means that when we use one-hot-encoding for day of the week, hour of the day, workflow id and scalar encoding for week number and file name, the combination is optimal. It is reasonable because day of week (1-7), hour of the day(0-24) and workflow id(0-4) all have limited number of values, which works for one-hot encoding. While week number and file name have unlimited numbers, and scalar encoding is more suitable.

v. Controlling ill-conditioning and over-fitting

Question: You should have found obvious increases in test RMSE compared to training RMSE in some combinations, can you explain why this happens?

Answer: For obvious increases in test RMSE compared to training RMSE in some combinations, the reason is that for combinations with more 1, eg:[11101] and [11111], day number and file name apply one-hot-encoding. In this way, the number of input features will become much larger, which causes overfitting on the training set.

i.Ridge Regulation

The best mode we obtained for this kind of regulation was a linear regularizer with α**= 10** and **Optimal Combination '01110'** (i.e. the first and last features scaler encoded and the rest one-hot encoded). The training RMSE for this best model was around 0.08834 and the test RMSE was around 0.08850. The following array shows the coefficients of the fitted model:

```
[      1.12213829e-05      -5.68405285e-03       3.91627975e-02       3.26491222e-03
 1.49099450e-03       -5.22907536e-03      -1.28131005e-02      -2.01924755e-02
-2.01413839e-02      -2.09905740e-02       7.76660500e-03       3.33486425e-02
-1.98555062e-03       2.00226104e-03       3.81267109e-02      -1.40968949e-02
-3.99758992e-02  -5.67354089e-02   7.26814921e-02   1.48819358e-05]
```

What we also know from the results is that when is less than 100, the best optimal combination exists at 01110, while is beyond 100, the best optimal combination are not stable. The train and test RMSE are around 0.0883 and 0.0885 while is less than 100, when is larger than 100, train and test RMSE increases a lot.

ii. Lasso Regulation

The best mode we obtained for lasso regulation with α**= 0.0001** and **Optimal Combination '11111'** (i.e. all the features are one-hot encoded). The training RMSE for this best model was around 0.08834 and the test RMSE was around 0.08851. The following array shows the coefficients of the fitted model:

```
[ -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
 4.37514320e-02       7.71888574e-03       5.93838483e-03      -0.00000000e+00
-6.96905340e-03      -1.44485431e-02      -2.10345962e-02      -2.18506979e-02
 5.78177614e-03       3.14450312e-02      -2.79292999e-03       2.96227139e-06
 5.17787527e-02      -0.00000000e+00      -2.56788224e-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]

What we also know from the results is that when α is less than 0.01, the best optimal combination exists at 11111, while α is beyond 0.01, the best optimal combinations are mostly 00000. The test and train RMSE increases asα increases. The coefficients become all 0 while $\alpha$is larger than 1.

iii. Elastic Net Regularizer

The best mode we obtained for elastic net regularizer with $\boldsymbol{\alpha}$**= 0.01, l1=0.0001, l2=0.0099** and **Optimal Combination '11111'** (i.e. all features one-hot encoded). The training RMSE for this best model was around 0.0884 and the test RMSE was around 0.08857. The following array shows the coefficients of the fitted model:

[-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 -3.31591581e-03
3.74568807e-02 3.77285401e-03 2.10566916e-03 -2.92017783e-03
-9.92300207e-03 -1.69812443e-02 -1.85667022e-02 -1.92574885e-02
6.77040206e-03 3.10175809e-02 -1.40515899e-03 1.33482689e-03
3.83796333e-02 -1.04741220e-02 -3.54014578e-02 -5.12447905e-02
6.86280237e-02 0.00000000e+00 0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -3.56389719e-05 -0.00000000e+00 -0.00000000e+00
-1.62213113e-04 -0.00000000e+00 -0.00000000e+00 -2.64857111e-04
-1.17037809e-05 0.00000000e+00 -0.00000000e+00 -2.44463200e-04
-2.82070313e-04 -7.37595035e-05 2.81588215e-03 3.25195296e-03
3.39864261e-03 2.57346767e-03 3.07354591e-03 2.98914556e-03
0.00000000e+00 0.00000000e+00 0.00000000e+00 -0.00000000e+00
-0.00000000e+00 -0.00000000e+00 -0.00000000e+00]

What we also know from the results that the test and train RMSE increases asα increases. When α is larger than 1, all the coefficients become 0.

iv. Best Unregularized Model

The best mode we obtained for unregularized model with **Optimal Combination '01110'** (i.e. first and last features scaled while others one-hot encoded). The training RMSE for this best model was around 0.08834 and the test RMSE was around 0.08850. The following array shows the coefficients of the fitted model:

```
[ 6.17610844e+10  6.17610844e+10  6.17610844e+10  6.17610844e+10
6.17610844e+10  6.17610844e+10  6.17610844e+10  6.17610844e+10
6.17610844e+10  6.17610844e+10  6.17610844e+10  6.17610844e+10
6.17610844e+10  6.17610844e+10  6.17610844e+10  8.05820637e+11
8.05820637e+11  8.05820637e+11  8.05820637e+11  8.05820637e+11
8.05820637e+11  8.05820637e+11 -2.13282889e+12 -2.13282889e+12
 -2.13282889e+12 -2.13282889e+12 -2.13282889e+12 -2.13282889e+12
 3.04578314e+11 -6.44865360e+11 -1.06281045e+12 -8.15718077e+11
-2.77005330e+11 -9.59045296e+11 -9.59045296e+11 -9.60162238e+09
 -9.60162238e+09  4.08343472e+11  4.08343472e+11  4.08343472e+11
4.08343472e+11  4.08343472e+11  4.08343472e+11  1.61251094e+11
1.61251094e+11 -9.59045296e+11  1.61251094e+11  1.61251094e+11
1.61251094e+11  1.61251094e+11 -3.77461652e+11 -3.77461652e+11
-3.77461652e+11 -3.77461652e+11 -3.77461652e+11 -3.77461652e+11
-9.59045296e+11 -9.59045296e+11 -9.59045296e+11 -9.60162238e+09
-9.60162238e+09 -9.60162238e+09 -9.60162238e+09]
```

Discussion: Compare the values of the estimated coefficients for these regularized good models, with the un-regularized best model.

Answer:

1) For regularized good models, the coefficients are much smaller than un-regularized best model.

2) Coefficients of regularized models have more zero values than un-regularized model.

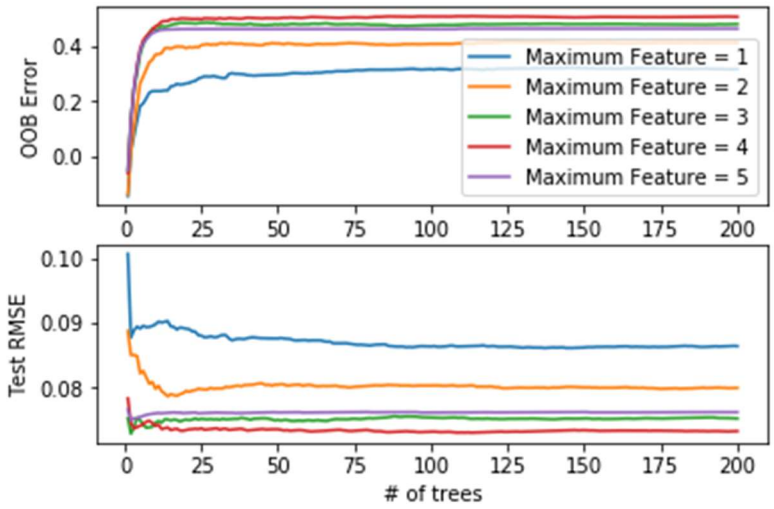
3) For Lasso and Elastic Net Regularized models, with increased α, the number of coefficients decreases to 5 with most of them, if not all, being zero.

(b) i. Report Training and average Test RMSE from 10-fold cross validation and Out Of Bag error from initial model

OOB Error = 0.462183201142
Train RMSE = 0.0758562036039
Test RMSE = 0.0760726397527

Analysis:
According to the data, the train RMSE and test RMSE are both slightly better than the method we have used in the previous section, which means, for each of the model in their tree, the models fits well. Besides, the RMSE for train and test set are approximately the same, which means each set has good predictions on their test sets. For the out-of-bag error, it is much larger than the RMSE since they are the data filtered out from the data we used in training, which could have difference in pattern. Thus, for each model generated from different trees, they have much worse prediction on the unseen data.

(b) ii. Sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5, plot figure 1 for out of bag error (y axis) against number of trees(x axis), figure 2 for average Test-RMSE(y axis) against number of trees(x axis).



The best number of trees for max_features = 1 is 145
Test RMSE = 0.0861250754311
The best number of trees for max_features = 2 is 14
Test RMSE = 0.0786381444334
The best number of trees for max_features = 3 is 2
Test RMSE = 0.0728515885559

The best number of trees for max_features = 4 is 113
Test RMSE = 0.0730184244879
The best number of trees for max_features = 5 is 2
Test RMSE = 0.0748912139753

Analysis:
According to the graph above, we found that the out-of-bag error increases as the number of trees increases, while the test RMSE decrease with increased number of trees.

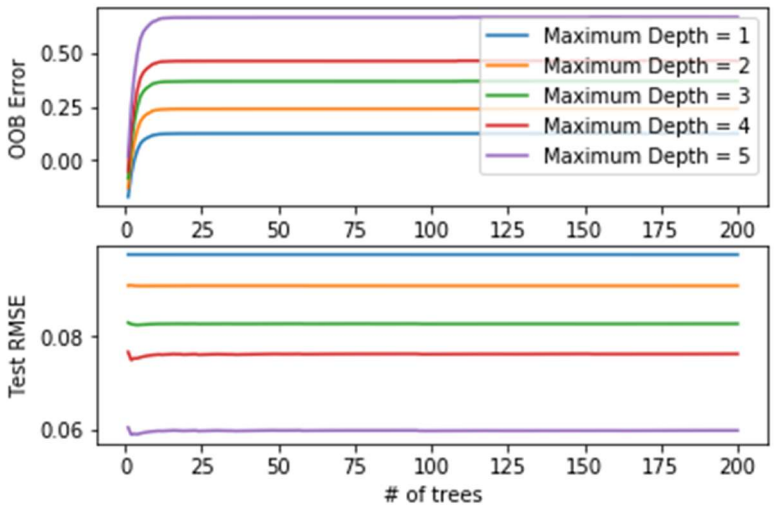
For the increasing OOB error, this may because that increase the number of trees generate more models, where increased number of models induced larger variance among all the trees. And we also found that the less the feature we used, the less the OOB error is, the reason could be the more features cause the prediction hardness for the sample out of bag.
For the decreasing RMSE on test set, it is reasonable that, for more analysis on the different random subset generated from the original sample, the error would be less. And we also notice that the less feature we used, the larger RMSE in each model we have, since more features provide more information which can make our prediction more accurate in each tree.

In conclusion, as we want our model has less RMSE in their own tree and also the less OOB to predict future data, the trade off between these two factors occurs. To averaging these two factors, we found that using depth equal to 3 would be appropriate. In addition, the number of trees being 2 would be appropriate since 2 is the mode in the result.

(b)iii. Pick another parameter you want to experiment on. Plot similar figure 1 and figure 2 as above. What parameters would you pick to achieve the best performance?
In this part, we chose to use the **max_depth** as our parameter. For the random forest tree model, one of the most important parameters is the tree depth. Extended tree depth could cause overfitting, since exact fit in one model tree can cause larger variance and the OOB error could be extremely large. OOB error represent the model's ability to predict future data. The larger the error is, the weaker the model is. Thus we want to find a reasonable tree depth to control our model not to be overfitting.

The best number of trees for max_depth = 1 is 109
Test RMSE = 0.0975340046111
The best number of trees for max_depth = 2 is 5
Test RMSE = 0.0907988589473
The best number of trees for max_depth = 3 is 4
Test RMSE = 0.0824068477978
The best number of trees for max_depth = 4 is 2

Test RMSE = 0.0748912139753
The best number of trees for max_depth = 5 is 2
Test RMSE = 0.0588520145196



Analysis:
In general, we found that the larger the max_depth is, the less the RMSE is for the test in each model. The max_depth represent for the percent fitting within the same tree, the larger the depth, the more fitted the model is. Thus, there would be less error with increasing depth.

We also found that OOB error increases with increasing depth, since as each model fits its own picked data, they would have less ability to predict other unseen data.

In conclusion, as we want our model has less RMSE in their own tree and also the less OOB to predict future data, the tradeoff between these two factors occurs. To averaging these two factors, we found that using depth equal to 4 would be appropriate. In addition, the number of trees being 2 would be appropriate since 2 is the mode in the result.

(b)iv. Report the feature importance from the best random forest regression we got
[2.06464367e-06 3.72203076e-01 7.23952483e-02 5.53855345e-01 1.54426657e-03], which corresponds to 'Week Number', 'Day of Week', 'Start Time', 'Filename', and 'Workflow ID', respectively.
Using this feature importance metric, the importance of the five features are ranked as follows (from most important to least important):

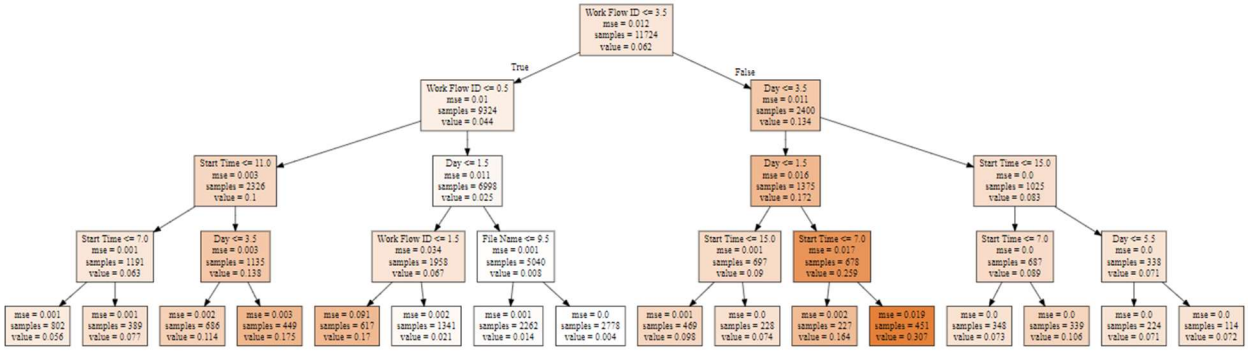
1. Workflow ID
2. Day of Week
3. Backup Start Time
4. Filename
5. Week Number

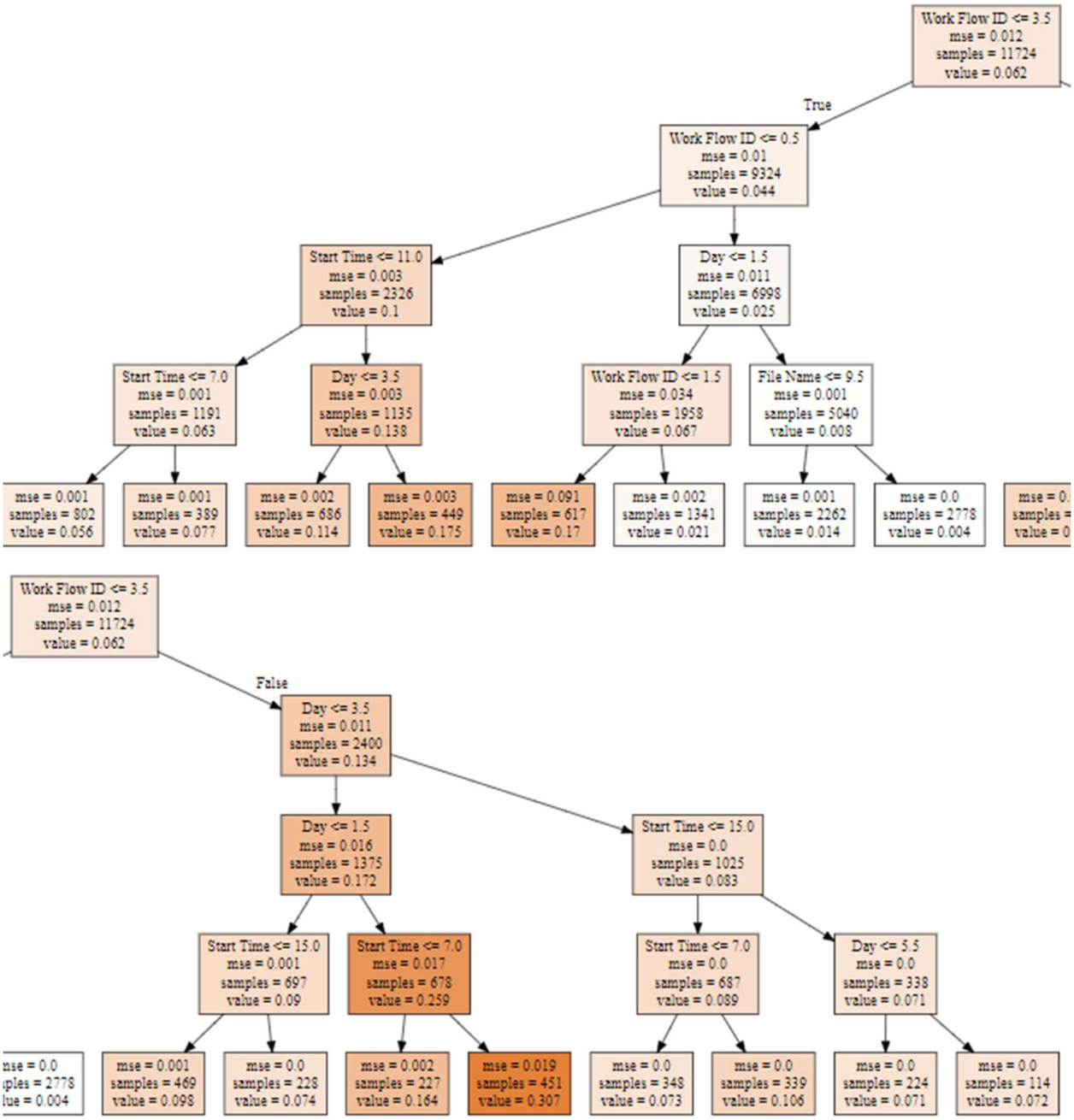
This rank of importance is different from the list of importance we got from using f_regression and mutual_info methods, since the random tree model used different algorithm from those two models. However, the common feature occurs in all three list is the Backup Start Time. This means though models analyze the problem in different ways, we can still pick important data through multiple features to increase the accuracy and decrease the effort.

Also, it is non-trivial to mention that in this part, we choose to use the following numbers:
Max_depth = 4
Max_features = 3
# of trees = 2
Those are the numbers we got from previous analysis.

(b)v. Visualize the decision trees and report the root node with analysis on feature importance

The following figure shows one of the trees in our best random forest regressor. Since the visualization does not quite fit the typeset, in the next two figures we are breaking it down into two sub-trees.

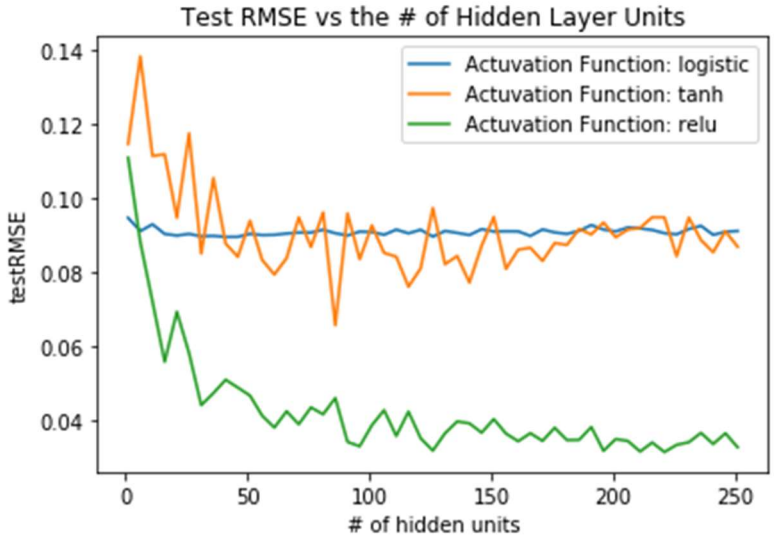




The root node is workflow ID <= 3.5, which accords with our result from (b)iv, where the feature importance for the work_flow_id is the largest. This means using this feature as major deviance in analysis is crucial to get the correct model. Besides, from the pattern we find in part 1, different workflows have different amount of change in some days, which can cause large variance as we have analyzed, so differentiating the data using workflows would be a reasonable choice. The other features in the child of roots that are used to divide the data in sequence also accord with our rank in the previous part, and indeed, all the nodes shown in the figure are using either workflow ID, start time or day of week as criteria.

2(c). Report the best combination on parameters: number of hidden units, activity function(relu,logistic,tanh)

The best number of hidden units for logistic is 41
Test RMSE = 0.0894420767545
The best number of hidden units for tanh is 86
Test RMSE = 0.0656335291798
The best number of hidden units for relu is 221
Test RMSE = 0.0315192585313



Analysis:
As shown in graph, all three functions has test error decreases with increasing number of hidden unit. The point we found to be best is the position where the plots tends to be stable.

For logistic, the plot is nearly flat, which means the function already works well for small number of hidden unit, so we choose the best one at 41 according to the trials.

For the tanh plot, we see that this one goes to steady state from about 90, and we get the actual number to be 86 from the experiment.

For the relu curve, the dropping part is the longest among the three functions, and the decrease of 0.001 is seen as a large improve, so it tends to be stable at about 200, where we choose it to be 221 according to the experiment.

(d) Predict the Backup size for each of the workflows separately.

(i) For workflow 0-4, we have the following prediction results.

Workflow 0

Train RMSE = 0.0429753502075

Test RMSE = 0.0432797948994

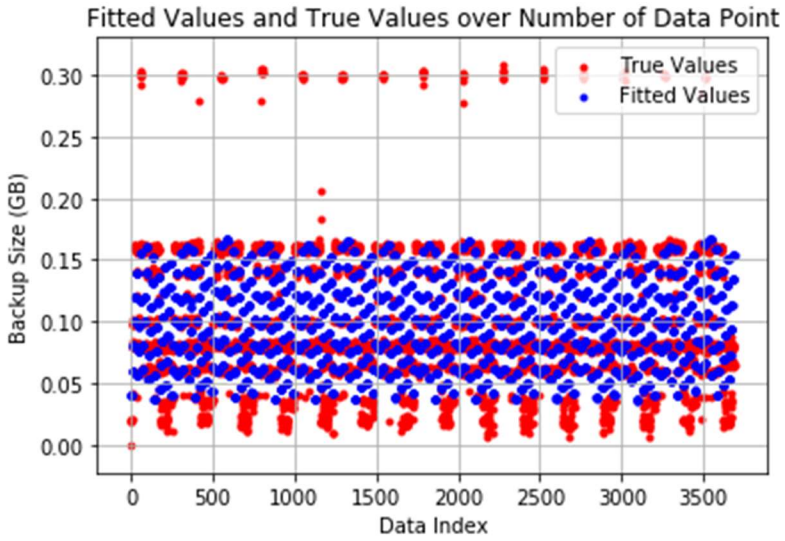


Figure 2(i).1 Workflow 0 fitted values vs true values

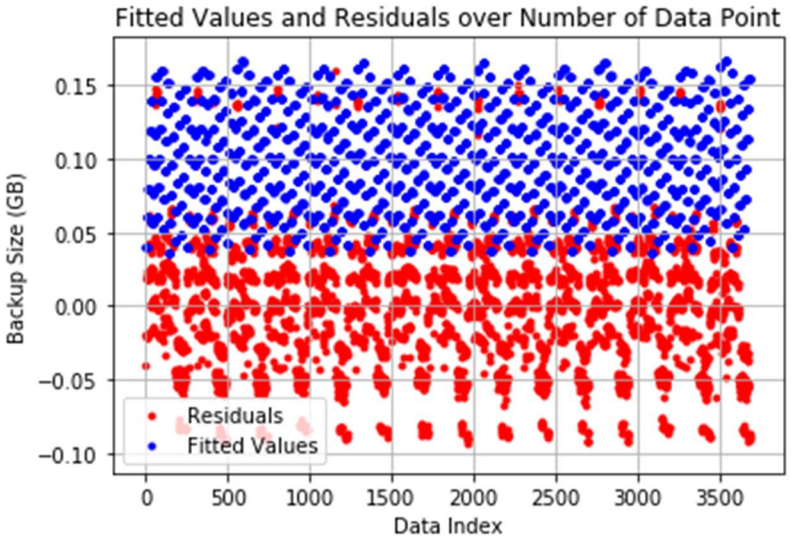


Figure 2(i).2 Workflow 0 fitted values vs residuals

Workflow 1

Train RMSE = 0.159588613416
Test RMSE = 0.160866897874

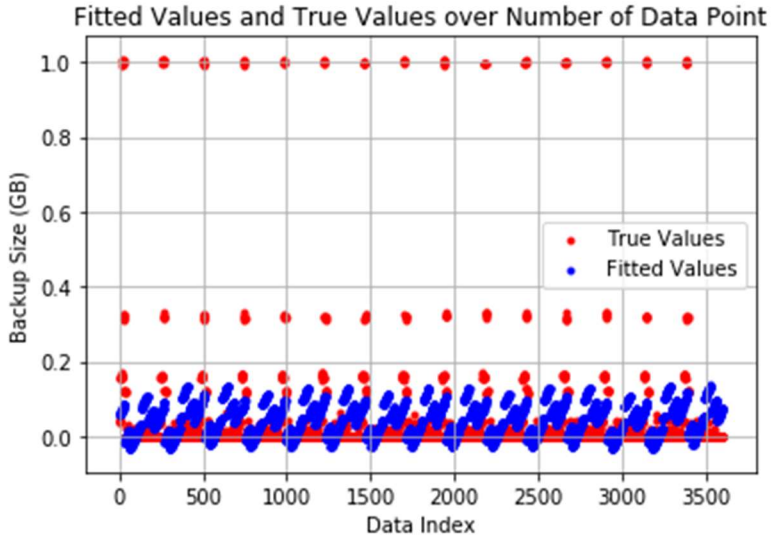


Figure 2(i).3 Workflow 1 fitted values vs true values

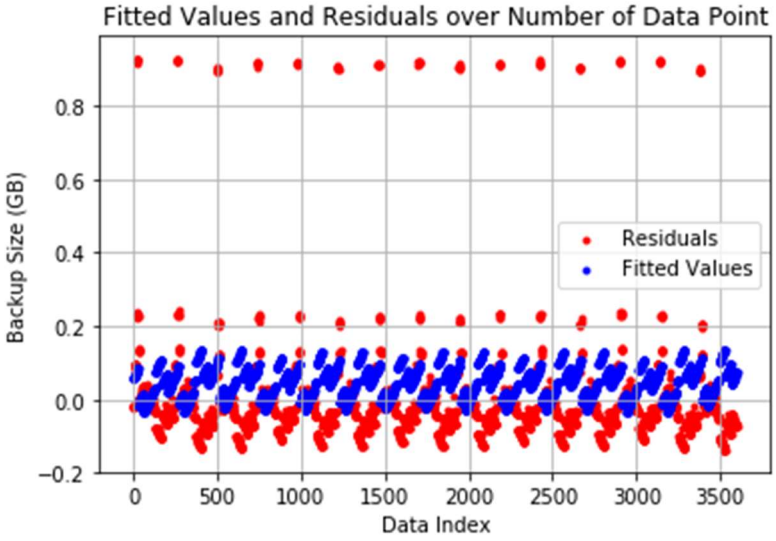


Figure 2(i).4 Workflow 1 fitted values vs residuals

Workflow 2

Train RMSE = 0.0422368374392
Test RMSE = 0.0423899431949

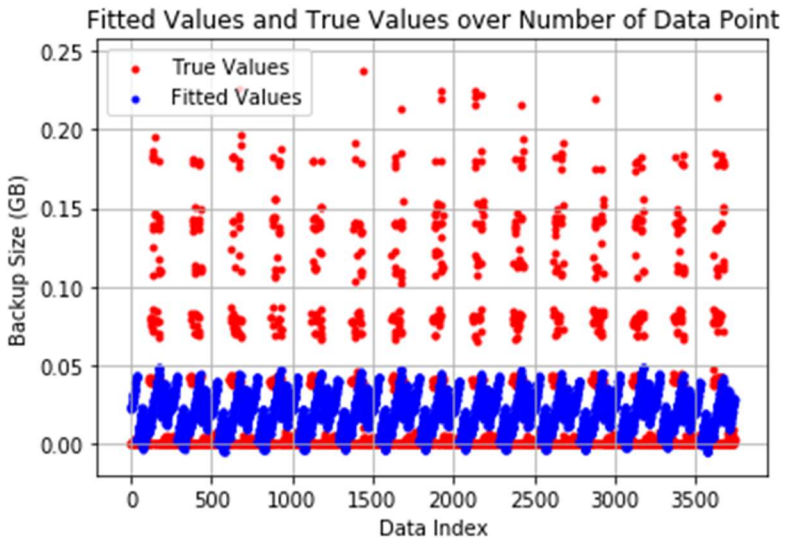


Figure 2(i).5 Workflow 2 fitted values vs true values

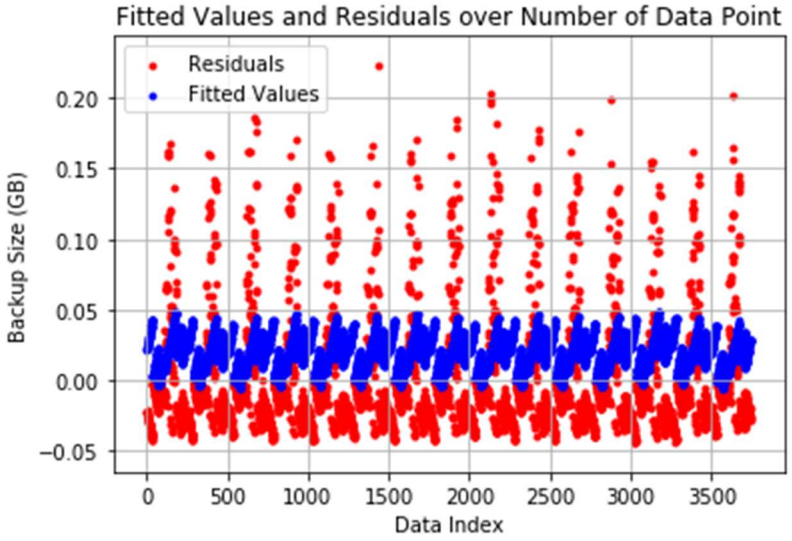


Figure 2(i).6 Workflow 2 fitted values vs residuals

Workflow 3

Train RMSE = 0.00711721308856
Test RMSE = 0.00713018372534

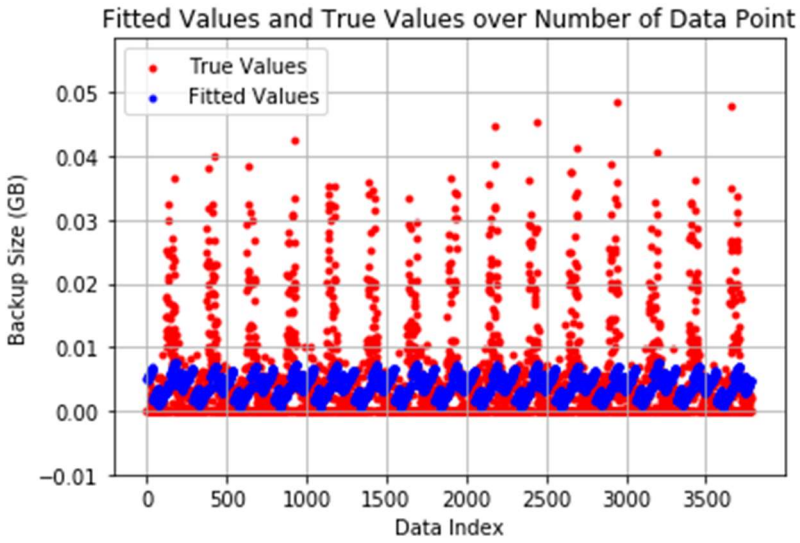


Figure 2(i).7 Workflow 3 fitted values vs true values

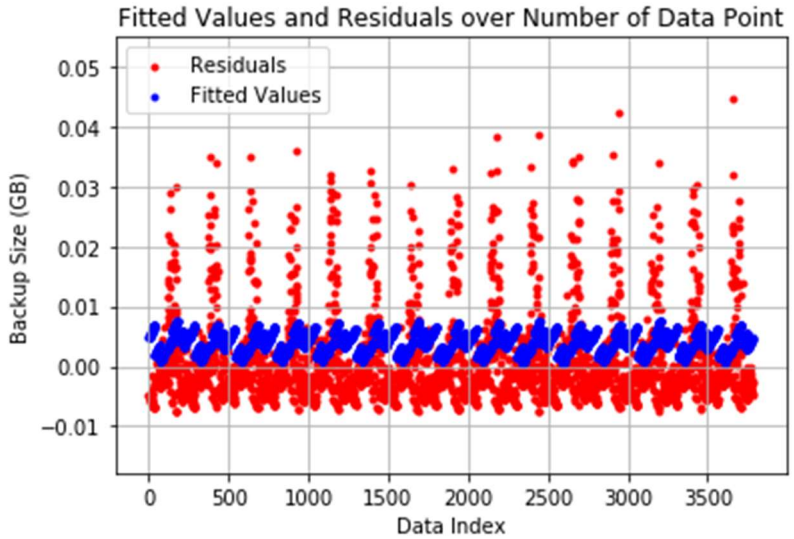


Figure 2(i).8 Workflow 3 fitted values vs residuals

Workflow 4

Train RMSE = 0.102994844158
Test RMSE = 0.103584897179

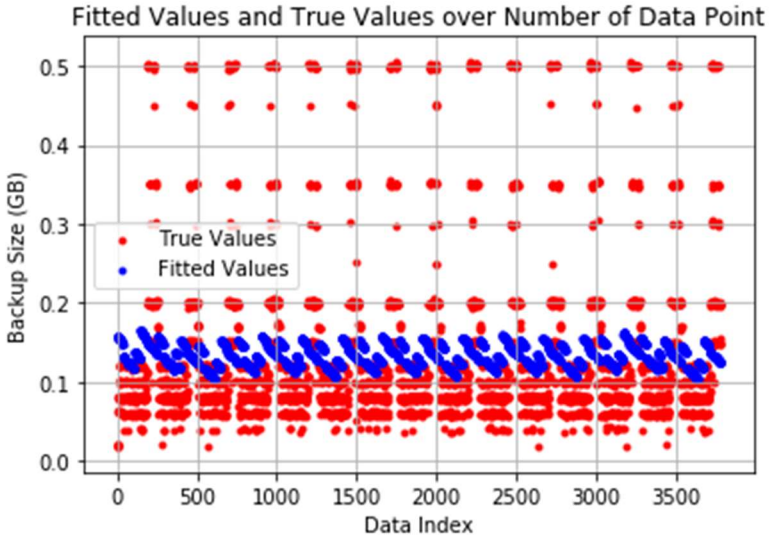


Figure 2(i).9 Workflow 4 fitted values vs true values

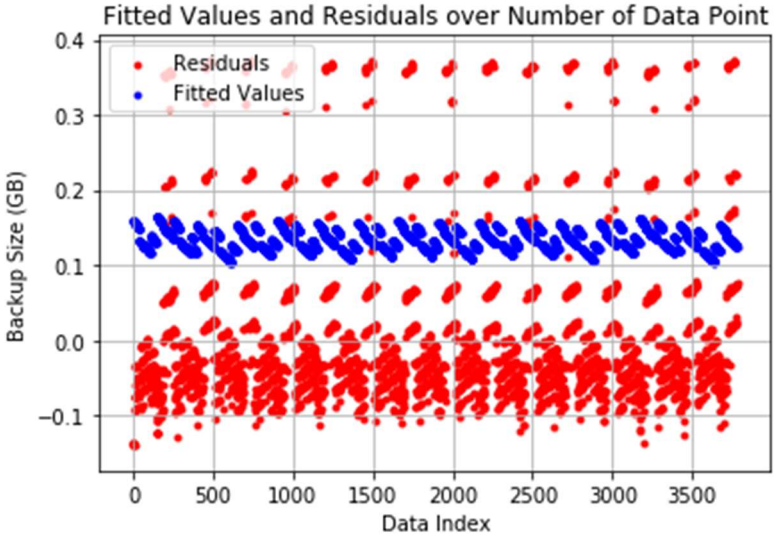
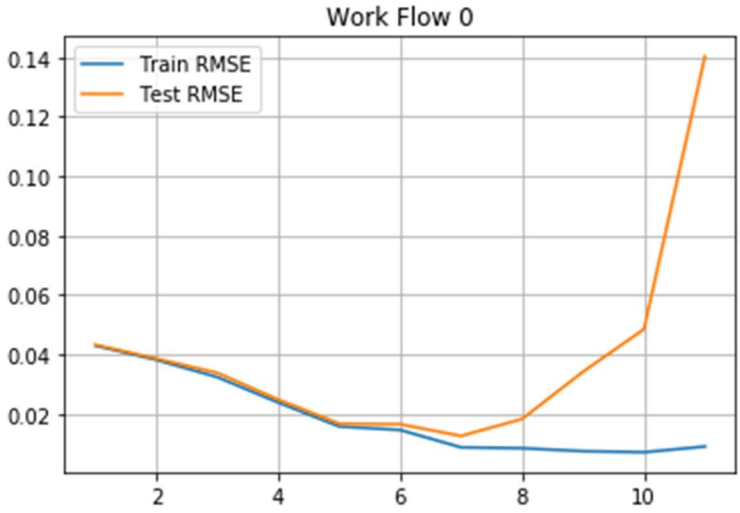


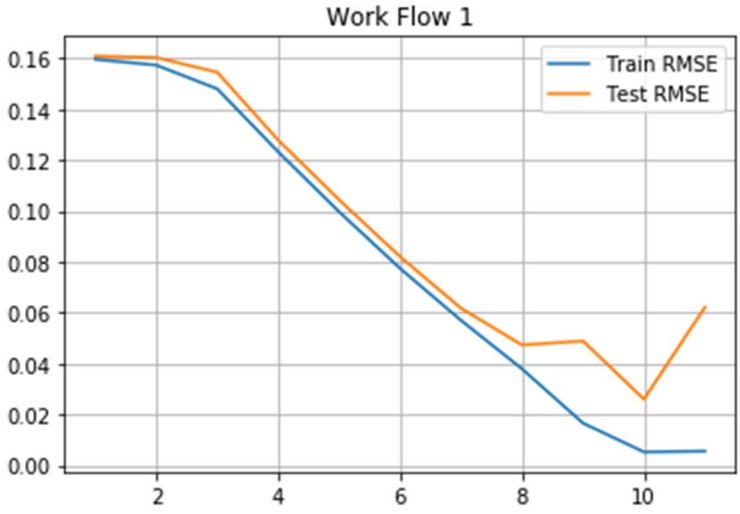
Figure 2(i).10 Workflow 4 fitted values vs residuals

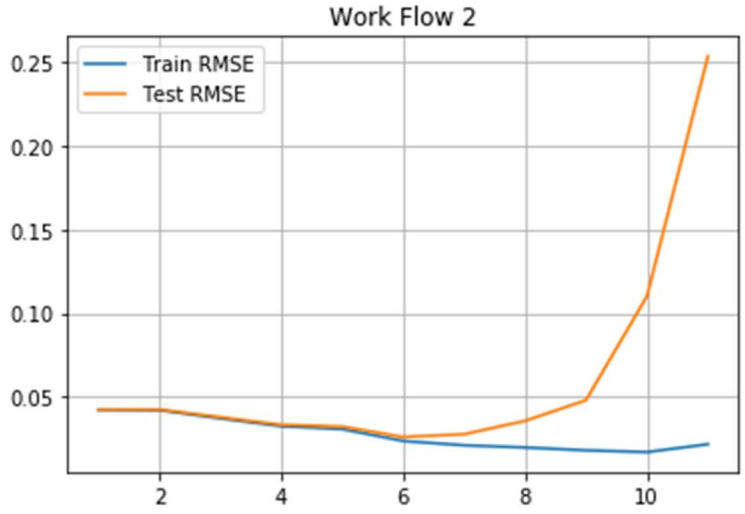
Analysis:

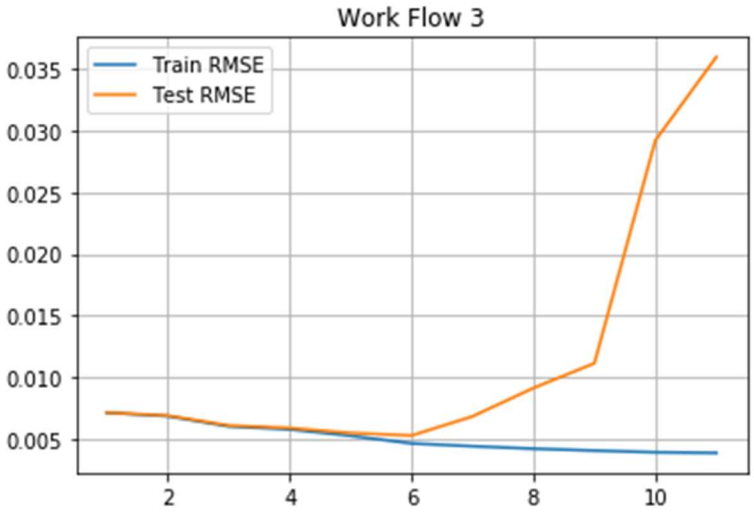
We can see that for workflow 0, 1, 2, 3 separately, the fitted values cover most of true values and the residuals are relatively small, which are from 0.00 to 0.20. For workflow 4, the fitted value cover some of true values and residuals are around 0.5. However, for linear regression model with all workflows fitted together, fitted values cover some true values and the residuals are around 1. In this way, the linear model on separate workflow increases the accuracy and the model improves.

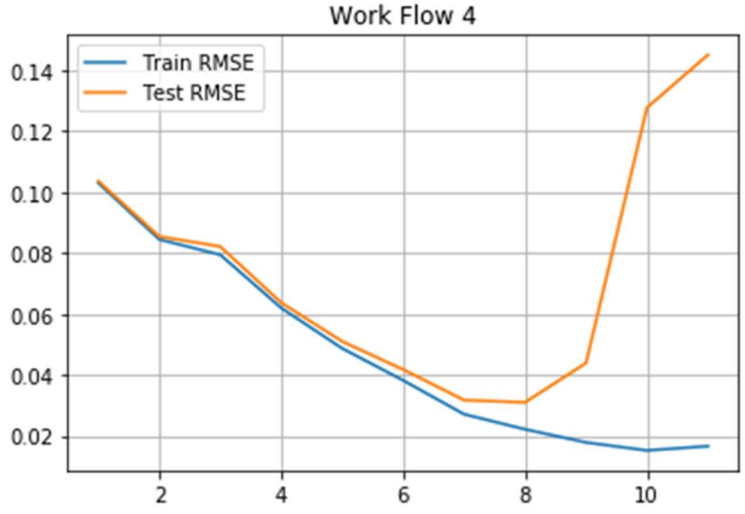
(ii)






Work Flow 2
Train RMSE
Test RMSE
0.25
0.20
0.15
0.10
0.05
2
4
6
8
10


Work Flow 3
Train RMSE
Test RMSE
0.035
0.030
0.025
0.020
0.015
0.010
0.005
2
4
6
8
10


Work Flow 4
Train RMSE
Test RMSE
0.14
0.12
0.10
0.08
0.06
0.04
0.02
2
4
6
8
10

Analysis:

1. From the plots from workflow 0~4, we can see that when the degree of polynomial reaches or is over 8, the test RMSE is increasing sharply. We can conclude that the generation error of model is worse beyond degree=8.
2. Cross validation helps control complexity of the model because this approach, by using a portion of the data to train the model (or a model with a wide range of complexity parameters) and the rest to test and validate, would directly estimate the complexity without knowing the prior information. And for our polynomial regression example, a higher order polynomial would be complex enough to generate overfitting. When doing cross validation, the discrepancies between training and testing error can be easily obtained and right choices of complexity parameters can be made.

The table below list the best test RMSE for linear model and polynomial model on workflow 0 to 4 separately. We can see that polynomial model in all have less test RMSE than linear model, it means that polynomial model better predicts data size than linear model.

Best Test RMSE	Workflow 0	Workflow 1	Workflow 2	Workflow 3	Workflow 4
Linear model	0.0433	0.1595	0.0420	0.0070	0.1028
Polynomial model	0.0126	0.0261	0.0259	0.0053	0.0313

Table.1 linear model vs polynomial on best test RMSE

(e) We applied ball_tree, kd_tree and brute algorithm in k-nearest neighbor regression.
The plot of test RMSE vs number of neighbors is below, we can say that the best model has lowest test RMSE, therefore, we can conclude from the plot:
1)The best number of neighbors for ball_tree algorithm is 6, the lowest Test RMSE = 0.05712
2)The best number of neighbors for kd_tree algorithm is 6, the lowest Test RMSE = 0.05624
3)The best number of neighbors for brute algorithm is 1, the lowest Test RMSE = 0.05306

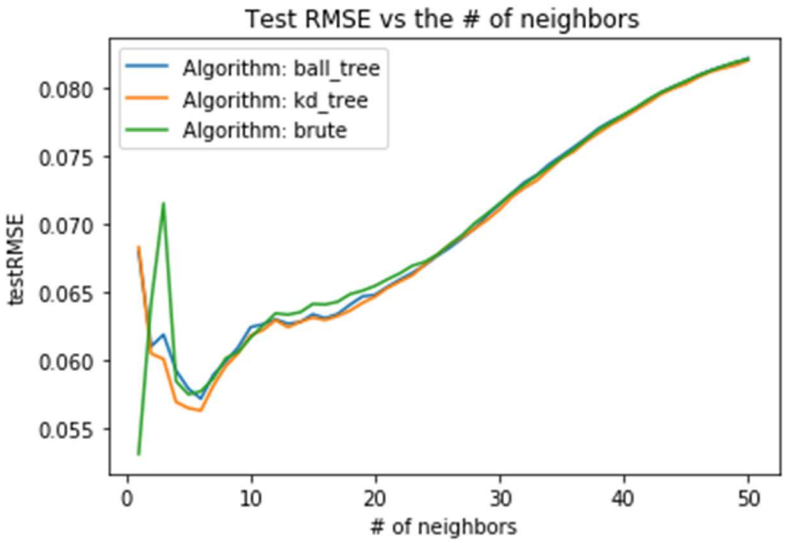


Figure 2(e) Test RMSE vs number of neighbors for three algorithms

From the plot of three algorithm we used in analyzing the k-nearest model, all three curves have almost the same pattern with a drop on about 6 neighbors and then the error increases. This means that all three algorithms tend to perform well in the condition of small number of neighbors, representing the sparse data. However, the brute force seems to work well for only one neighbor. As the algorithm defines the nearest means only the distance of the neighbor to the center means nearest, which is not similar to the other algorithm that also considering the radius and dimension.

3. Compare the different regression models:
Linear regression model
Random forest regression model
Neural network regression model
K-nearest regression model

Linear model is good at dealing with features, since, from the one-hot-coding, linear regression model tell us how many feature would be appropriate in constructing the models and which three features would work, whereas for the rest of the models, none of them specifies the specific use of features. In addition, it is not good at dealing with sparse data. Though different workflow all have the regular pattern, they have different variance and thus sparse. The linear model performs well only on certain workflows. Comparing to unregularized ones, regularized linear regression models would definitely be preferred since the regularization term controls the model complexity and indeed the squared errors are somewhat smaller. Also, all the linear regression models seem to provide a better performance with sparse features since the cross validated best models mostly have a combination of '11111' or '01110'.

The K-nearest regression model is good at dealing with sparse model. As the number of neighbors increases, the RMSE first reaches to a local minimum and then increases, which means if the data are too “crowded”, it would perform bad.

Neural network regression model tends to perform good in general, since the RMSE is generally small if the hidden unit is large enough.

Random forest regression model has better performance on dealing with overfitting, where the OOB represent the error test on the unused data.