Project4 Report

Cheng Ma 105033453, **Jinxi Zou** 605036454, **Shuo Bai** 505032786, **Xiaoxi Gong** 705034355

March 5, 2018

1 Introduction

In this project, we work with a dataset which is network backup dataset. For this project, we use data mining approaches of regression. Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In our project, we explore basic regression model along with basic techniques to handle over-fitting, such as cross-validation, regularization. The network backup dataset includes the simulated traffic data. It also includes the week index, day of the weeks which the file backup has started, backup start time, work flow ID, file name, backup size, and backup time. Our goal is predicting the backup size with other five features. We use different model of regression, linear regression, random forest regression, neural network regression, and k-nearest neighbor regression model.

2 Problem1

For the problem 1, our goal is to load the dataset and get idea on the type of relationships in the network backup dataset.

2.1 part a

This part requires us to plot the backup sizes for all workflows, which is quite simple, just use the *groupby* method and plot the figures.

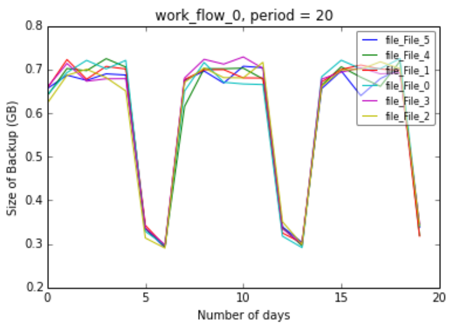


Figure 1: size of Backup (GB) versus number of days

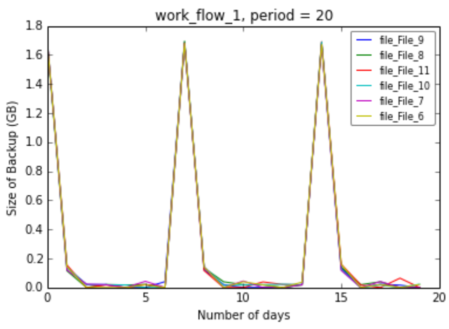


Figure 2: size of Backup (GB) versus number of days

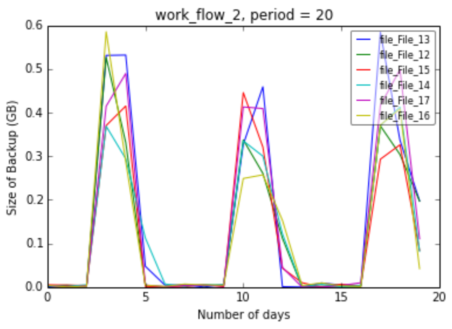


Figure 3: size of Backup (GB) versus number of days

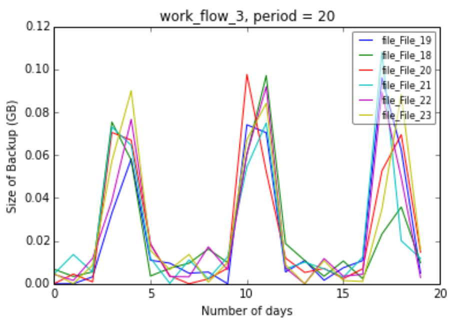


Figure 4: size of Backup (GB) versus number of days

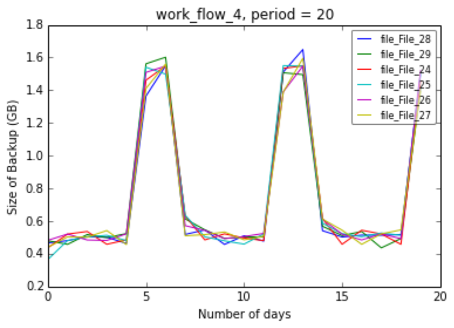


Figure 5: size of Backup (GB) versus number of days

2.2 part b

This part we just change a parameter from 20 to 105 and we can get the following graphs.

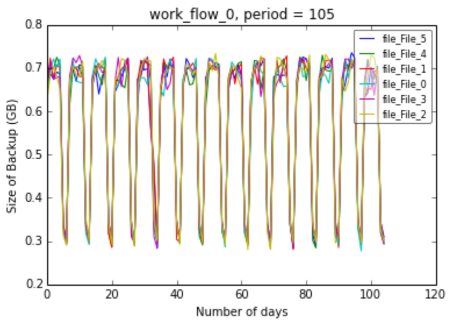


Figure 6: size of Backup (GB) versus number of days

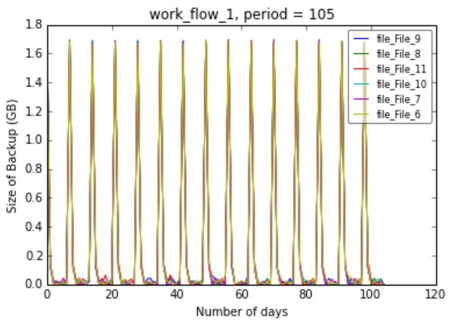


Figure 7: size of Backup (GB) versus number of days

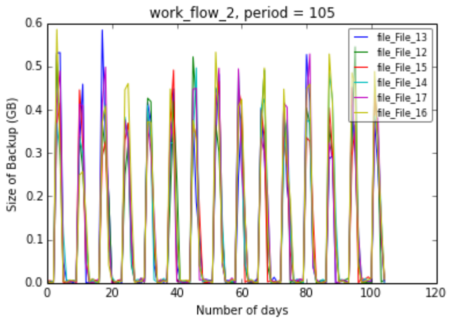


Figure 8: size of Backup (GB) versus number of days

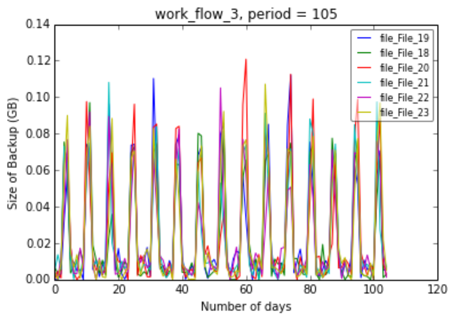


Figure 9: size of Backup (GB) versus number of days

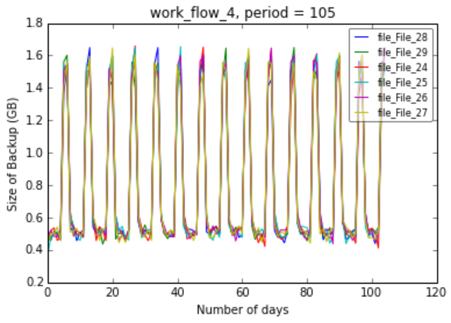


Figure 10: size of Backup (GB) versus number of days

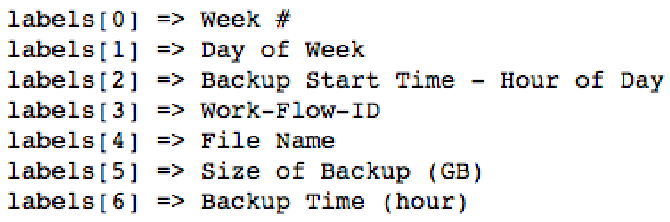
And we can easily find the repeating patterns from the graph. The plot repeat every 7 days with slight variations. We could know that "week" is not a important feature. And from the graph, obviously we could conclude that weekday and weekend have difference. For the work_flow_0, the backup size is large during the weekday, and drop dramatically at the weekend. For the work_flow_1, the backup size is small during the weekday and increase a lot when the weekend. For the work_flow_2, it only have large backup size on wednessday and other days have small backup size. For the work_flow_3, the backup size on Thursday is large and other days is small. For work_flow_4, the backup size is relatively small during the weekday, and increase on the weekend.

3 Problem2

3.1 part a

3.1.1 problem i

In this part, we are asked to take the most simple task which is just try to use Linear regression to train a basic model, our dataset is composited as following:



```
labels[0] => Week #
labels[1] => Day of Week
labels[2] => Backup Start Time - Hour of Day
labels[3] => Work-Flow-ID
labels[4] => File Name
labels[5] => Size of Backup (GB)
labels[6] => Backup Time (hour)
```

Figure 11: dataset composition

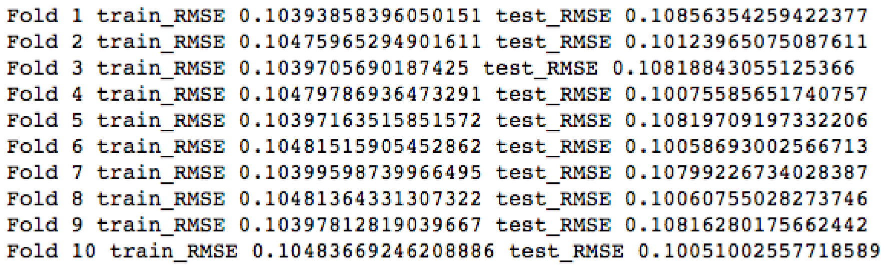
From the requirement of this project, we should discard the backup time feature. Therefore, in order to train our model, we should make some transformation, there are two kinds of them:

One is scalar encoding in which we transform the value of a feature directly into a specific number(Monday to 1. e.g). Another is One-Hot-Encoding in which we set an binary array to set a specific position as 0,1 to indicate the value(Monday to [1,0,0,0,0,0,0] e.g).

For the first part, we apply all feature with scalar encoding to our Linear regression model. The penalty function is shown as following:

$$min_{\beta}||Y - X\beta||^2 \quad (1)$$

For every folder, we should train a model with train set and then compute the Training RMSE and Test RMSE. The result is shown as following:



```
Fold 1 train_RMSE 0.10393858396050151 test_RMSE 0.10856354259422377
Fold 2 train_RMSE 0.10475965294901611 test_RMSE 0.10123965075087611
Fold 3 train_RMSE 0.1039705690187425 test_RMSE 0.10818843055125366
Fold 4 train_RMSE 0.10479786936473291 test_RMSE 0.10075585651740757
Fold 5 train_RMSE 0.10397163515851572 test_RMSE 0.10819709197332206
Fold 6 train_RMSE 0.10481515905452862 test_RMSE 0.10058693002566713
Fold 7 train_RMSE 0.10399598739966495 test_RMSE 0.10799226734028387
Fold 8 train_RMSE 0.10481364331307322 test_RMSE 0.10060755028273746
Fold 9 train_RMSE 0.10397812819039667 test_RMSE 0.10816280175662442
Fold 10 train_RMSE 0.10483669246208886 test_RMSE 0.10051002557718589
```

Figure 12: dataset composition

From the result, we can see that in most folders. The test RMSE is higher than the training RMSE and they distributes among 1.02, it shows that the model is not so good because the RMSE is comparatively high.

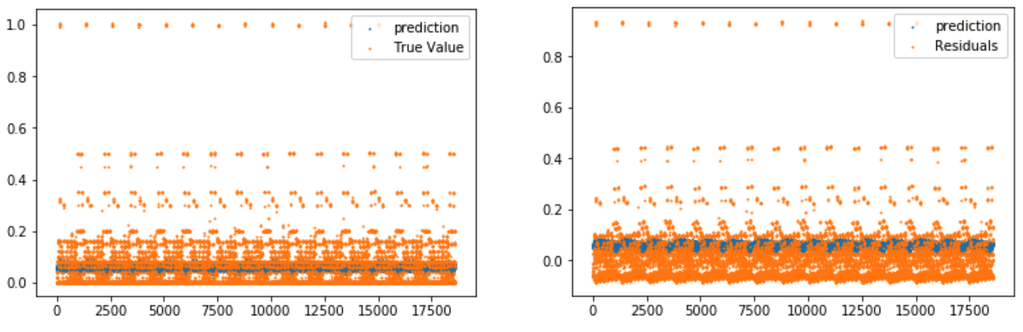
We can see the result from scatter picture more directly. In this part we should plot two kinds of scatter: fitted-true and fitted-residual:

From the result, we can the linear regression for this kind of result really does bad work. Because the result distribution is not seriously distributed along the line and in the residual scatter figure some values are extremely far away. Therefore, we should make some improvement.

3.1.2 problem ii

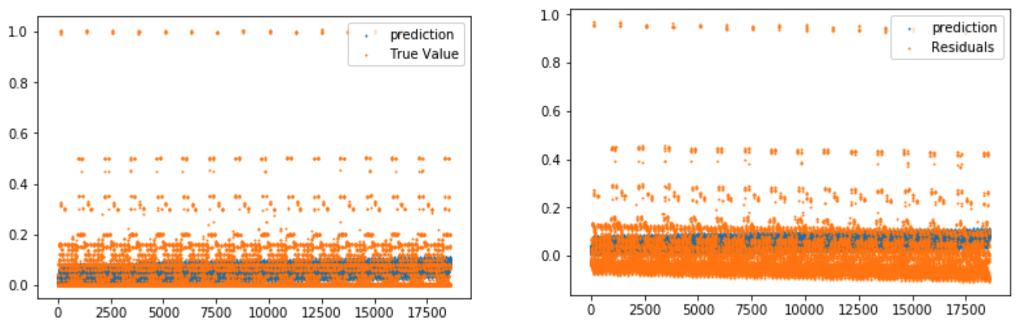
In this problem, we make a simple improvement which is to standardize the data. After a few tries, we find that we just need only scale the data and will gain a higher performance than what we apply scale and centering both. Therefore, we get the result as following:

From the result we can observe that we really have improve the training performance. The data distributed more linearly. However, it still looks bad because there are still many data points distributed far away. Therefore, we should introduce more useful way.



(a) (b)

Figure 13: fitted over true and residual over fitted



(a) (b)

Figure 14: fitted over true and residual over fitted

3.1.3 problem iii

Before we apply the one-hot-encoding in to our training, we check some important features in our model. We separately take f_regression and mutual information regression measure to select.

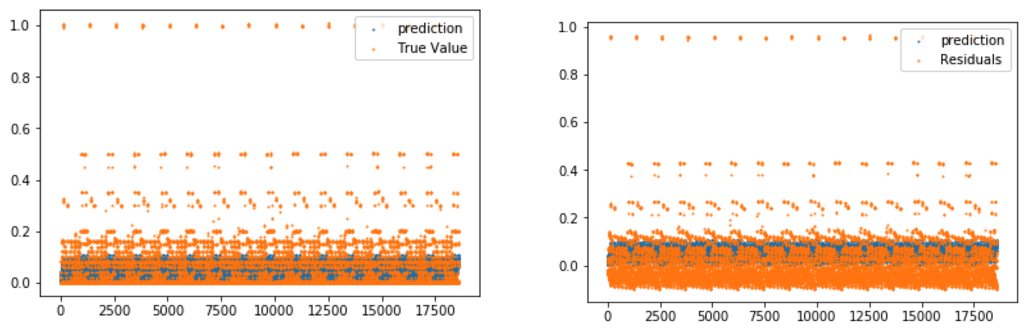
Using the f_regression measure, we can calculate three most important feature:

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

Backup Start Time - Hour of Day
Day of Week
Work-Flow-ID
```

Figure 15: f_regression three most important feature

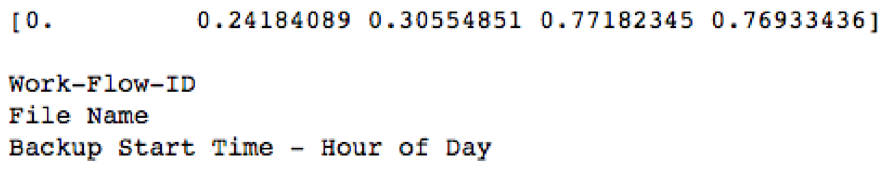
we use the three features to train our model and get the similar result as following:



(a) (b)

Figure 16: fitted over true and residual over fitted

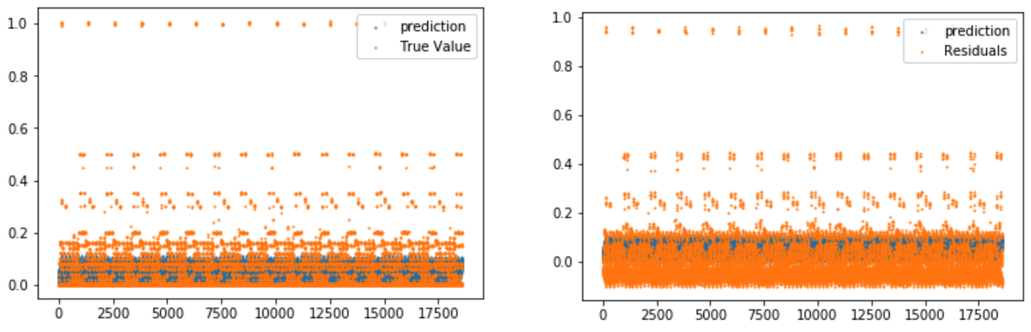
From the result shown, we get a similar result as before, may be it has improved a little. However the improvement is not quite obvious. The following is the same way we use with mutual information regression:



```
[0.         0.24184089 0.30554851 0.77182345 0.76933436]

Work-Flow-ID
File Name
Backup Start Time - Hour of Day
```

Figure 17: mutual information regression three most important feature



(a) (b)

Figure 18: fitted over true and residual over fitted

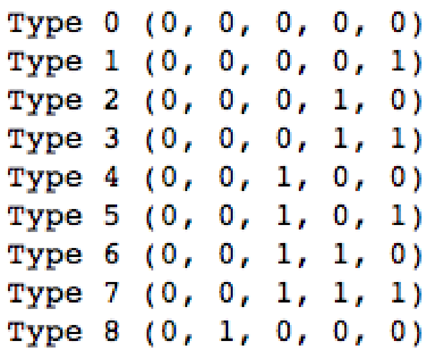
The result shows comparatively better performance. However, through fitted value over true value. The performance is still bad. Therefore, we will introduce one-hot-encoding.

3.1.4 problem iv

We use a combination of 32 types data. They are indicated as the following way. The position 1 means we apply one-hot-encoding in this label and the 0 means we apply scalar-encoding in this label.

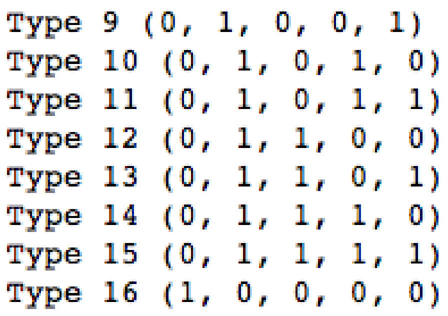
For this part, it is hard for us to make a single result plot for every RMSE in every folder. Therefore we take another way to calculate an average Test and Training RMSE as the standard for our evaluation.

$$RMSE_{a}verage = \sqrt{\frac{MSE_1 + MSE_2 + \ldots + MSE_{10}}{10}} \tag{2}$$



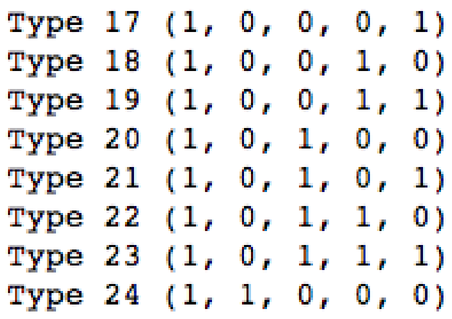
```
Type 0 (0, 0, 0, 0, 0)
Type 1 (0, 0, 0, 0, 1)
Type 2 (0, 0, 0, 1, 0)
Type 3 (0, 0, 0, 1, 1)
Type 4 (0, 0, 1, 0, 0)
Type 5 (0, 0, 1, 0, 1)
Type 6 (0, 0, 1, 1, 0)
Type 7 (0, 0, 1, 1, 1)
Type 8 (0, 1, 0, 0, 0)
```

(a)



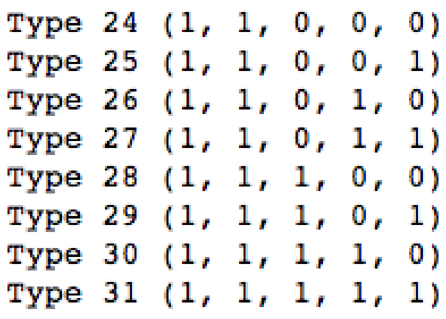
```
Type 9 (0, 1, 0, 0, 1)
Type 10 (0, 1, 0, 1, 0)
Type 11 (0, 1, 0, 1, 1)
Type 12 (0, 1, 1, 0, 0)
Type 13 (0, 1, 1, 0, 1)
Type 14 (0, 1, 1, 1, 0)
Type 15 (0, 1, 1, 1, 1)
Type 16 (1, 0, 0, 0, 0)
```

(b)



```
Type 17 (1, 0, 0, 0, 1)
Type 18 (1, 0, 0, 1, 0)
Type 19 (1, 0, 0, 1, 1)
Type 20 (1, 0, 1, 0, 0)
Type 21 (1, 0, 1, 0, 1)
Type 22 (1, 0, 1, 1, 0)
Type 23 (1, 0, 1, 1, 1)
Type 24 (1, 1, 0, 0, 0)
```

(c)



```
Type 24 (1, 1, 0, 0, 0)
Type 25 (1, 1, 0, 0, 1)
Type 26 (1, 1, 0, 1, 0)
Type 27 (1, 1, 0, 1, 1)
Type 28 (1, 1, 1, 0, 0)
Type 29 (1, 1, 1, 0, 1)
Type 30 (1, 1, 1, 1, 0)
Type 31 (1, 1, 1, 1, 1)
```

(d)

Figure 19: 32 types of data

To be noticed, we should set the shuffle parameter to false to observe obvious difference between $RMSE_test$ and $RMSE_train$. The 32 type comparison is shown as following:

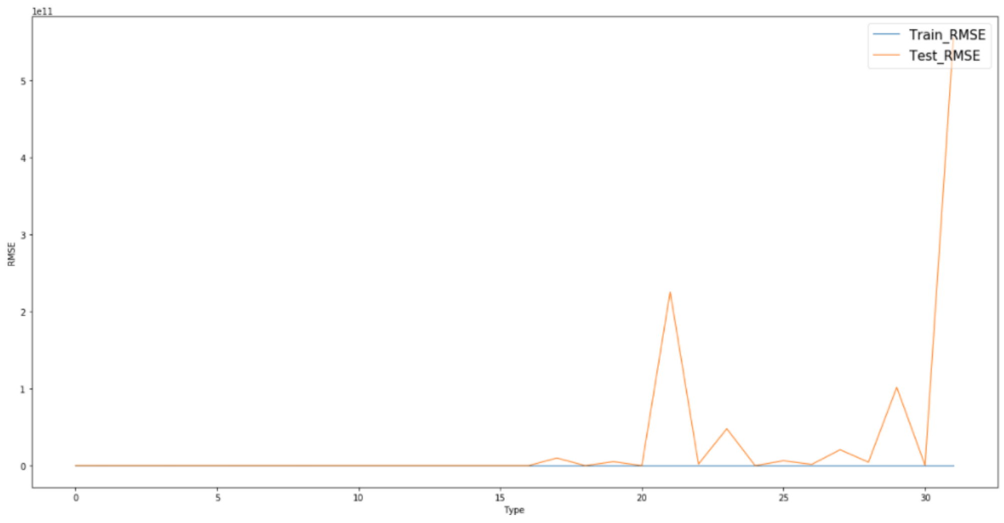


Figure 20: mutual information regression three most important feature

We find that the smallest Test RMSE is **0.0885042607046** which appears in Type **14**. We apply the one-hot-encoding to label_1, label_2 and label_3. These three features are three most important features we use f_regression to get.

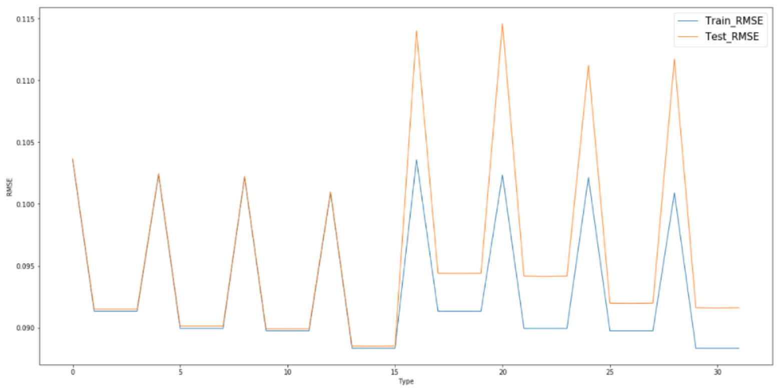
However, when we check the plot we find that we apply type **21, 23, 29**, the test RMSE is quite higher than train RMSE. It indicates that in this condition the training model is quite bad. When we observe these three types. They all apply one-hot-encode in label_0, label_2 and label_4. This will cause bad prediction. May be the reason is that label_0 is not so important in predicting the size. As we have calculated before. label_2 and label_4 have higher importance. If we apply one-hot-encode in week number and two other feature together. It may cause conflict(a unexpected sparse matrix) and affect the important feature.

3.1.5 problem v

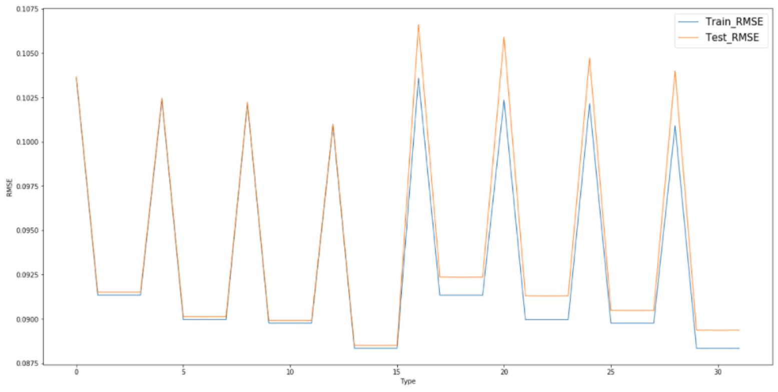
In this part, we use two kinds of Regularizers to make comparison with the linear regression.

$$RidgeRegularizers : \min_{\beta} ||Y - X\beta||^2 + \alpha||\beta||_2^2 \tag{3}$$

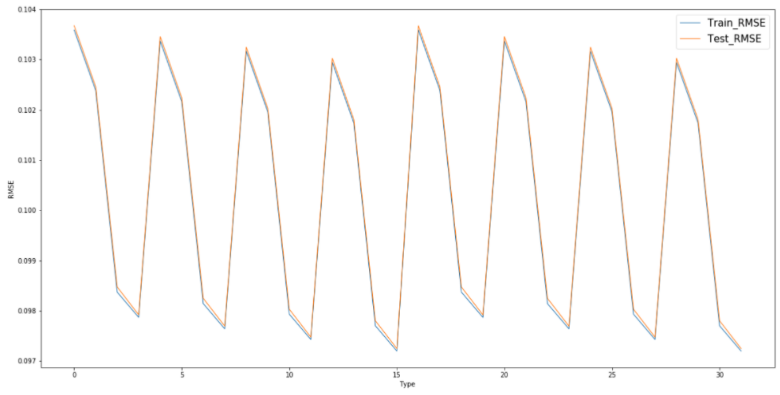
We try α as 0.001,1,10000 to observe the result:



(a)



(b)



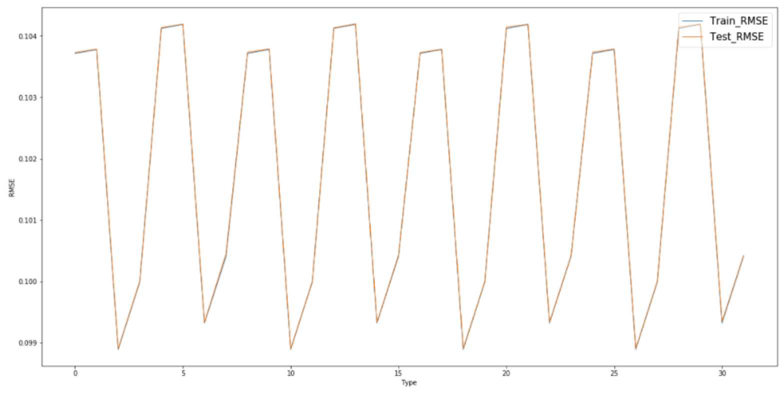
(c)

Figure 21: 32 type of α = 0.001,1,10000(top-bottom)

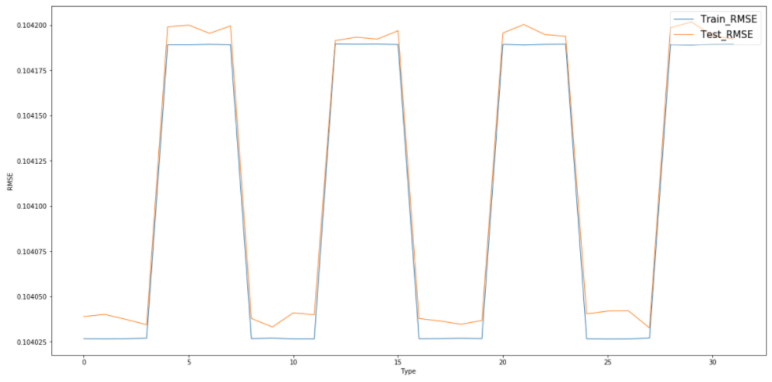
We can observe that when α is very low, the test RMSE and train RMSE have a large gap and as the α increases, the gap becomes smaller. At the same time, when α locates around 1, the test RMSE will have a comparatively lower value which is better than Linear Regression.

$$LassoRegularizers : \min_{\beta} ||Y - X\beta||^2 + \alpha||\beta||_1^2 \qquad (4)$$

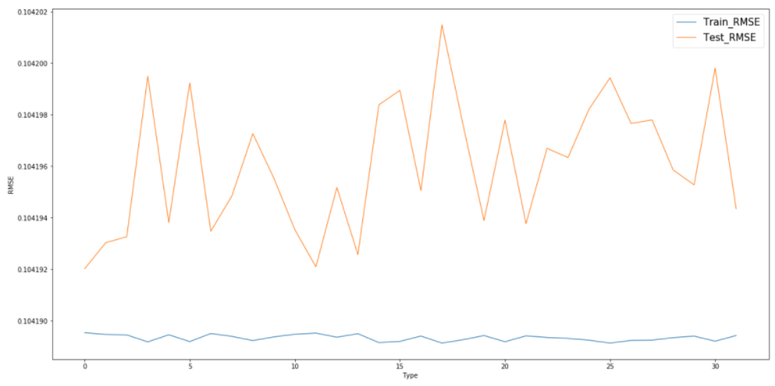
In the Lasso Regularizers, we can not set the α as before. Because when α is very large here, we can't see any difference. Finally, after many tries, we set the α as 0.01, 0.05, 0.1 so that we can observe an obvious change.



(a)



(b)



(c)

Figure 22: 32 type of α = 0.01,0.05,0.1(top-bottom)

In this part, we gain a reversed conclusion as the α increases, the gap between train RMSE and test RMSE becomes larger and we get a lower performance than Linear regression because the RMSE locates around 0.100.

3.2 part b

3.2.1 problem i

In this part, I used Random Forest model to regression and predict backup size. The number of trees in Random Forest model is 20, the maximum depth is 4, features number is 5. I transformed data to scalar data and used k-fold to split data into ten folds to observe regression performance. After The result Average Train RMSE was 0.0603416698099, the result Average Test RMSE was 0.0605420221758. Test RMSE was a little larger than train error as I thought. Then I used whole set of data to calculate the out of bag error, the result was 0.343369776393.

3.2.2 problem ii

I swept over number of trees from 1 to 200 and number of features from 1 to 5. After cross-validation, the result of average Test RMSE is shown in figure 23 and figure 24.

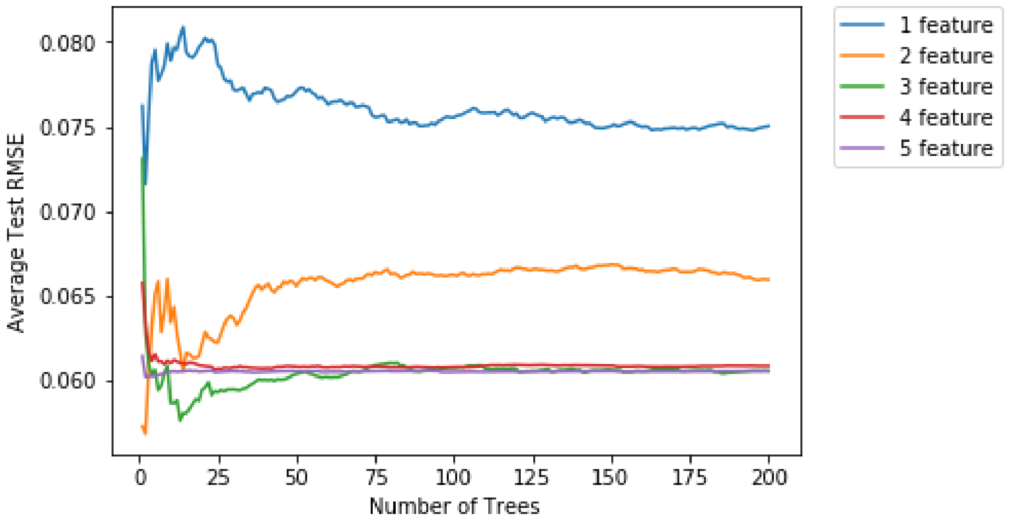


Figure 23: Average Test RMSE against tree number

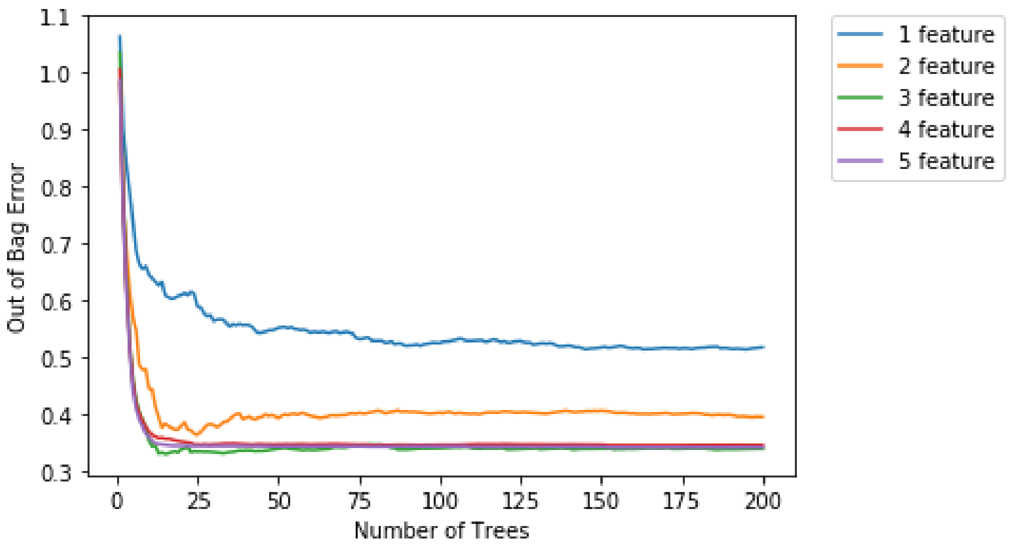


Figure 24: Out of bag error against tree number

From figure 23 and figure 24, we knew that increasing number of trees would reduce RMSE and out of bag error in the beginning, then RMSE and out of bag error kept approximately stable. The RMSE and out of bag error both decreased when number of features increased but stayed around 0.34 when number of feature is larger than 3.

3.2.3 problem iii

I picked depth of trees to experiment on. I swept maximum depth from 1 to 20, the I got figure 25.

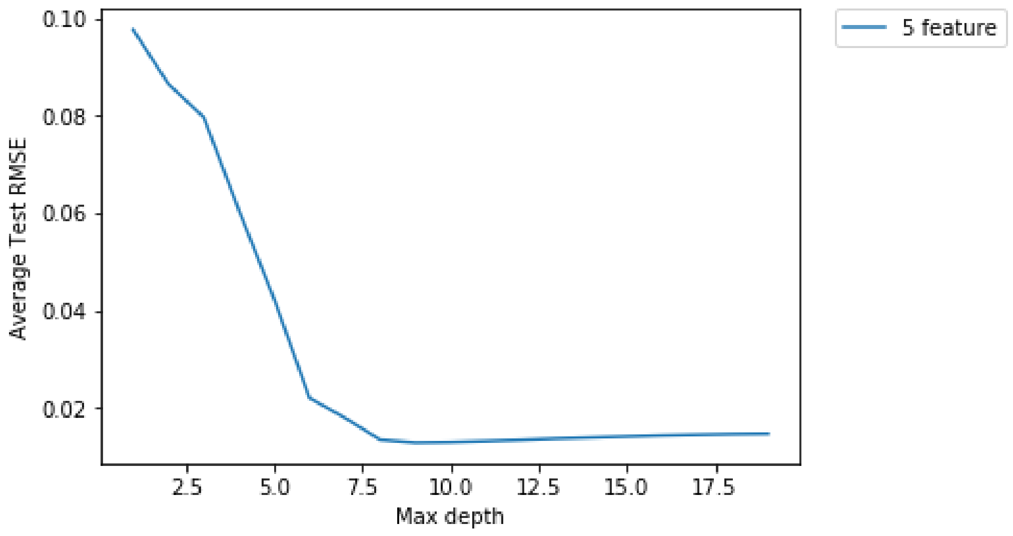


Figure 25: Average Test RMSE against maximum depth

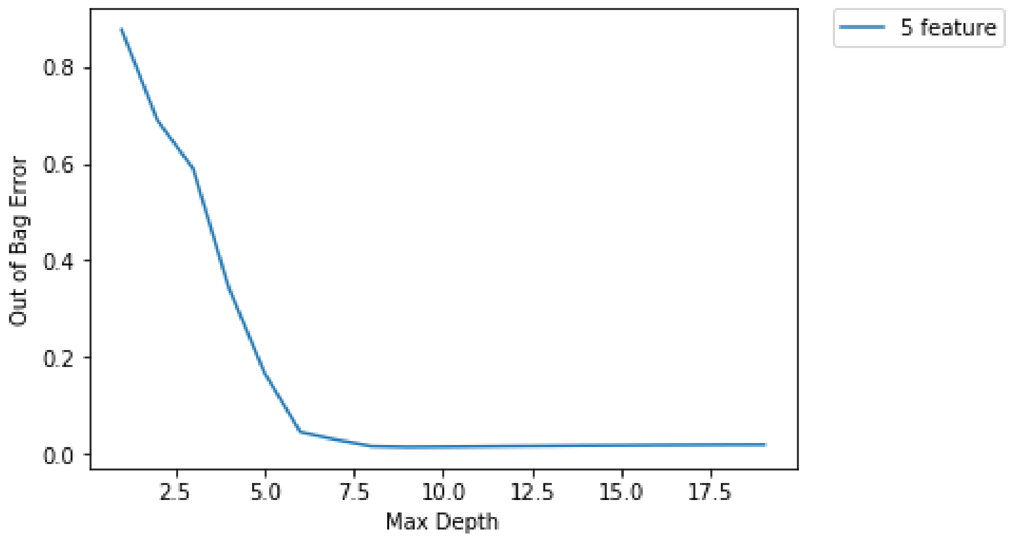


Figure 26: Out of bag error against maximum depth

The trend of test RMSE and out of bag error were quite similar to number of trees and number of features. Increasing maximum depth can decrease RMSE and out of bag error, but when maximum number is larger than 5, Test RMSE and out of bag error kept stable. Since increasing depth will increase computation complexity, I picked maximum depth 5 to be the best model's parameter, whose error was small and computation complexity was fine.

3.2.4 problem vi

I set number of trees to be 25, features to be 5 and maximum depth to be 5 as the best random forest regression model. Then I output the importance of features, and the result is in table 1.

Number of Week	3.71415156e-05
Day of week	2.24652734e-01
Hour of Day	3.16439781e-01
Work-Flow-ID	1.82947681e-01
File Name	2.75922662e-01

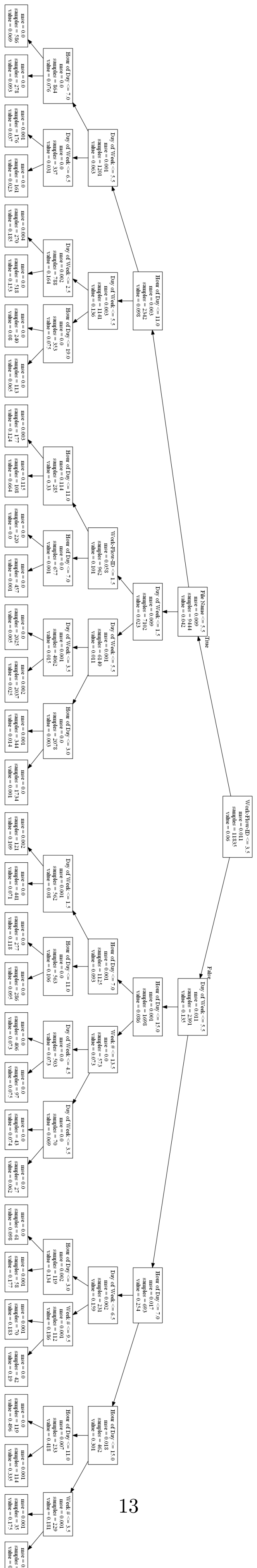
From table 1, I knew that the most important feature is hour of day, Which is the consent with Problem 1 result suggested.

3.2.5 problem v

I visualize one tree in best random forest model with number of tree 25, number of feature 5 and maximum depth 5. The result is shown in figure 27.

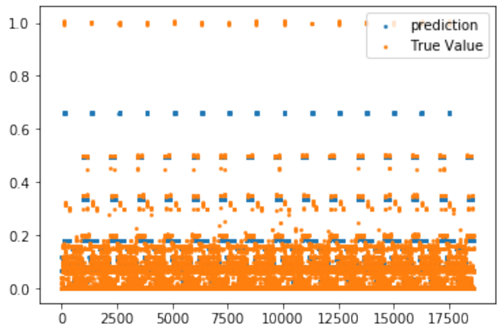
From figure 27, the root node of decision tree is work-flow -id, which is not the most important feature.

The reason why the root note of tree we plot was different from the most important feature is that the importance of features were avearage decreased variance of all decision trees in forest rather than the only tree we plot. The root node of one tree in forest was selected randomly, so it's quite normal that the root node of one tree in forest isn't the most important feature.



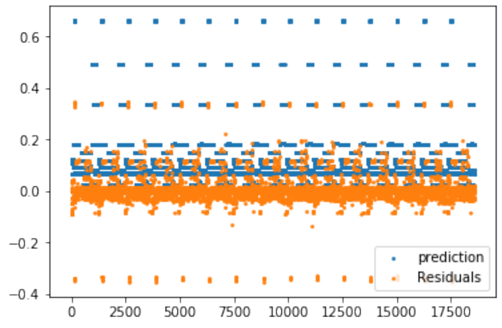
3.2.6 Scatter diagram

Finally, I drew the scatter diagram of fitted value and true value as well as residual value of each sample. The results are shown in following figure.



(a)

Figure 28: True and Fitted values against data point



(a)

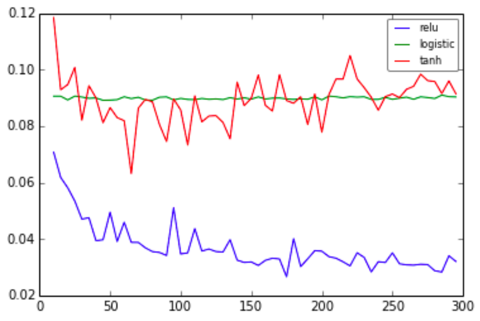
Figure 29: Residual and Fitted values against data point

From figure of True and Fitted values against data point, it's can be seen that most True values were overlapped by prediction values, proving the model was good for predicting backup size.

3.3 part c

In this part, we predict the backup size with different hidden units and different activation. And we aim to find the best parameters.

The plot of RMSE versus number of units is as follows.

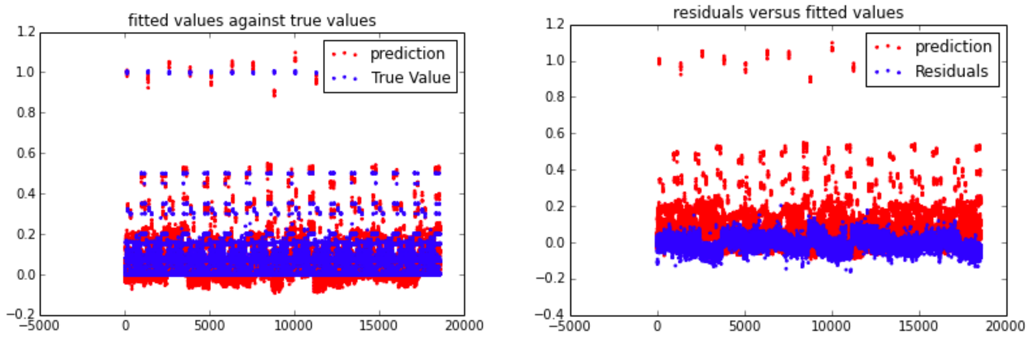


(a)

Figure 30: RMSE versus number of units

From the graph shown above we found that 'relu' is the best activation. And we choose 175 hidden units as the best parameters.

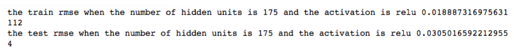
According to the following graphs, we can conclude that the neural network regression model fits the data quite well.



(a) (b)

Figure 31: fitted over true and residual over fitted

And the final test and train rmse are 0.031 and 0.019 accordingly.



```
the train rmse when the number of hidden units is 175 and the activation is relu 0.01888731697563l
112
the test rmse when the number of hidden units is 175 and the activation is relu 0.030501659221295
4
```

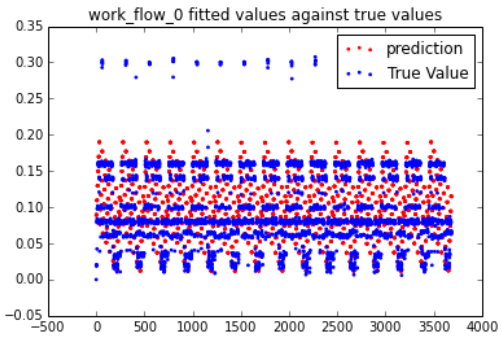
(a)

Figure 32: test and train rmse

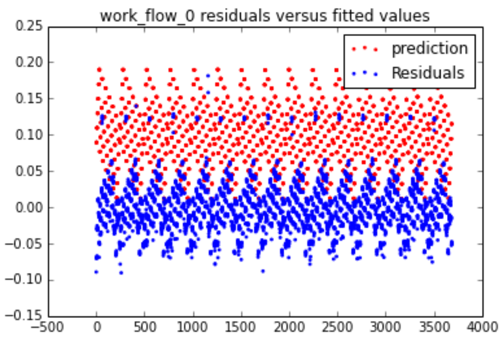
3.4 part d

3.4.1 problem i

In this part, we predict the Backup size for each of the workflows, and from the graphs we can conclude that some workflows have a quite good performance than the previous linear regression model, while others don't have such good improvement. The graphs are lists as follows.

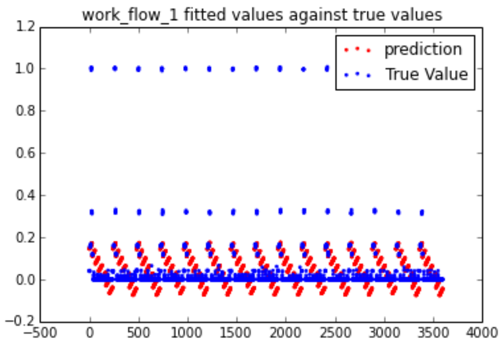


(a)

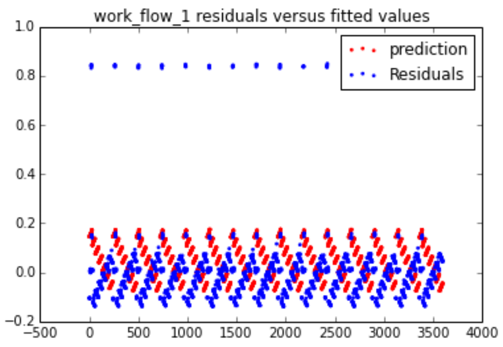


(b)

Figure 33: fitted over true and residual over fitted

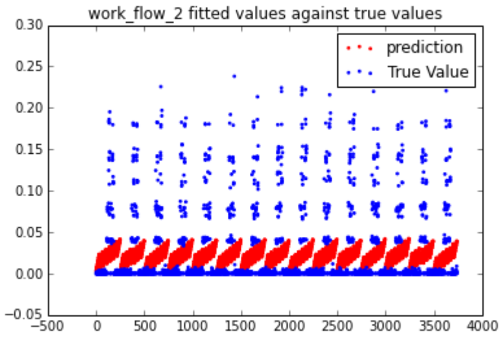


(a)

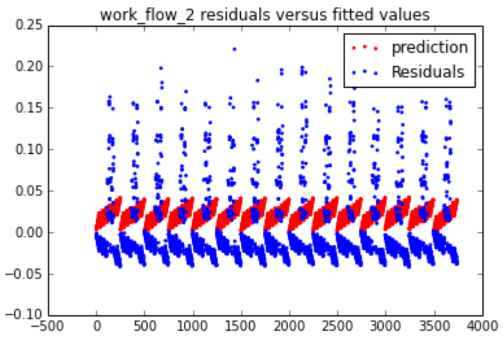


(b)

Figure 34: fitted over true and residual over fitted

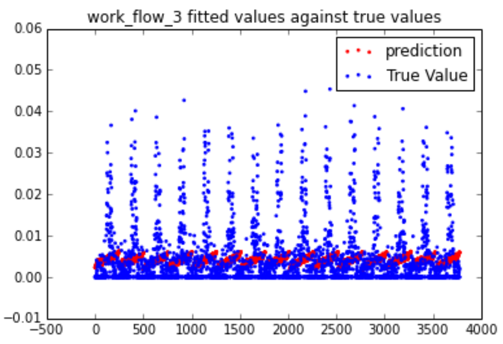


(a)

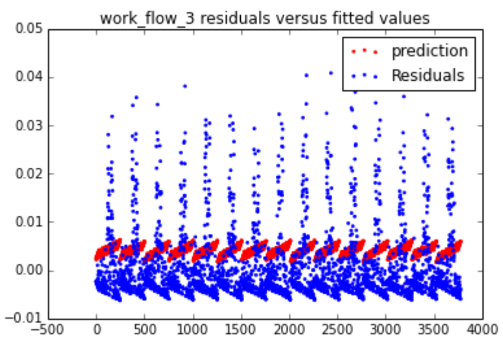


(b)

Figure 35: fitted over true and residual over fitted

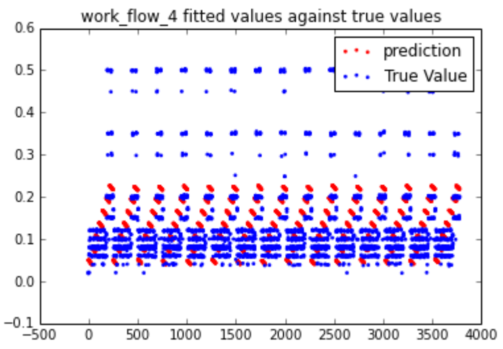


(a)

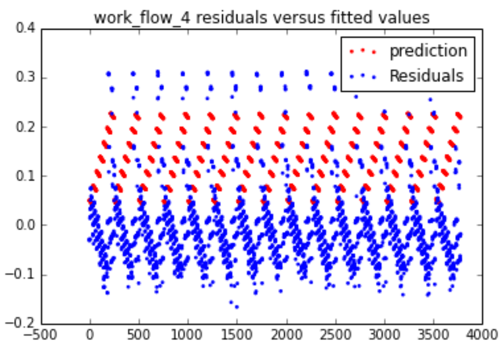


(b)

Figure 36: fitted over true and residual over fitted



(a)



(b)

Figure 37: fitted over true and residual over fitted

3.4.2 problem ii

When using polynomial function, we could certainly get a better result than using the basic linear regression model. To achieve this, we use the function *PolynomialFeatures* to get the polynomial combinations of the features, then use the linear regression to fit these features. And by this way we can get the polynomial regression of the data.

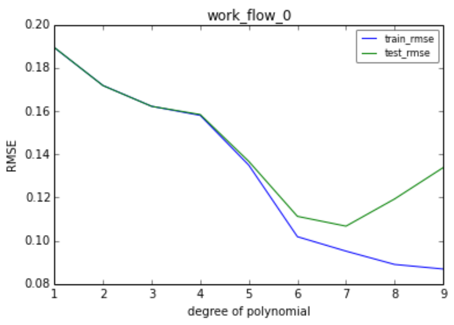


Figure 38: RMSE versus degree for workflow0

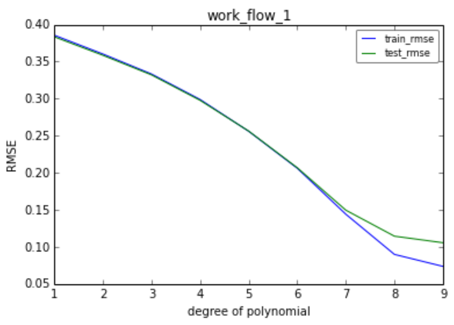


Figure 39: RMSE versus degree for workflow1

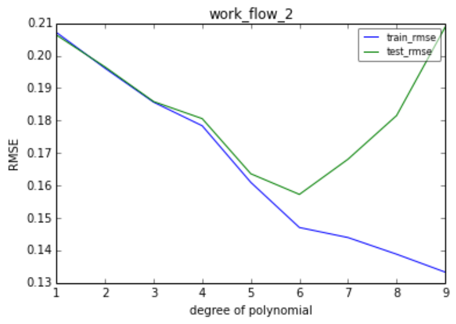


Figure 40: RMSE versus degree for workflow2

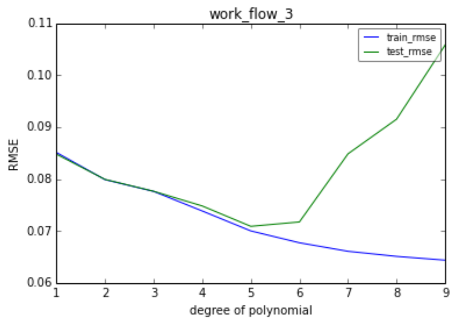


Figure 41: RMSE versus degree for workflow3

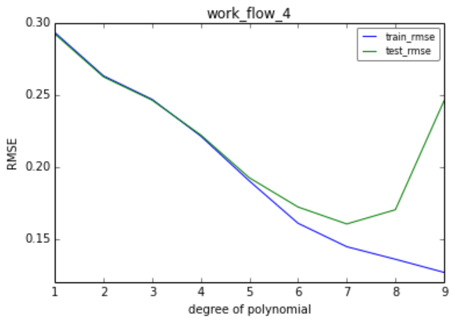


Figure 42: RMSE versus degree for workflow4

From the above figures we can conclude that different workflows have different thresholds beyond which the error will get worse. The typical threshold is around 6. And this is as expected, because high polynomial function will result in overfitting, and thus the performance on the test set will get worse.

Cross validation allows us to use the same data set to do the training for multiple times, so it improve the utilization of the data. By dividing the data set into training set and test set, it helps us to detect the overfitting.

3.5 part e

In this section, we use k-nearest neighbor regression to fit the data. For simplicity, we range the k from 1 to 30, and the result shows that the algorithm gets the best rmse when the number of neighbor is 1.

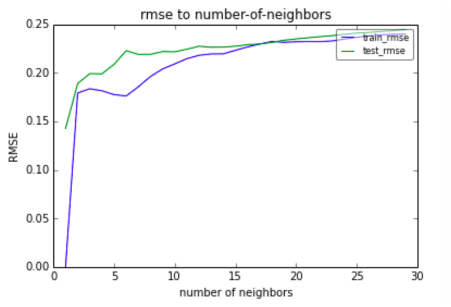
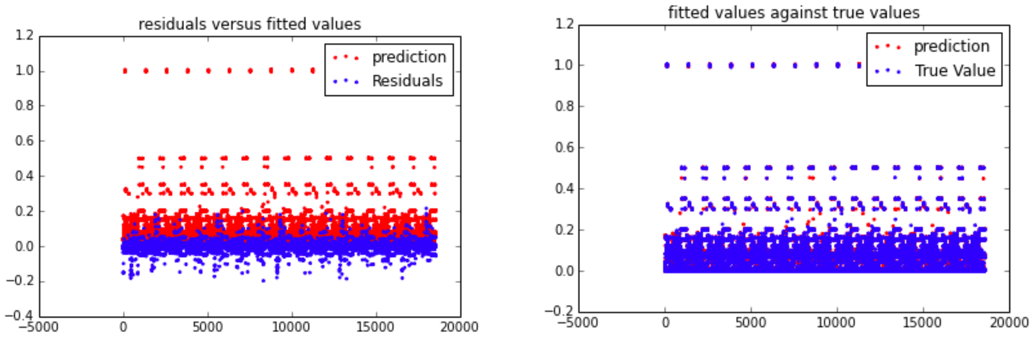


Figure 43: RMSE versus number of neighbors

From the figure we can see that the train rmse is 0 when the number of neighbor is 1. And this is not beyond our expectation, because under this situation, the one neighbor that one nodes will find is exactly itself, thus there is no deviation.

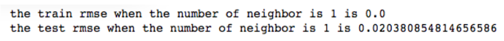
And we also plot the fitted values against true values and residuals versus fitted values scattered over the number of data points using the whole dataset when number is 1, and the result we get is surprisingly good.



(a) (b)

Figure 44: fitted over true and residual over fitted

Also, we get the train and test rmse, and the values are 0 and 0.02, accordingly.



```
the train rmse when the number of neighbor is 1 is 0.0
the test rmse when the number of neighbor is 1 is 0.020380854814656586
```

(a)

Figure 45: train and test rmse

4 Problem 3

After comparing with different regression models, we found that k-nearest neighbor regression over all generates the best results when handling categorical features, the test rmse reached 0.02. Linear regression model behaved better when handling sparse features than categorical features and the lowest average test rmse can be as low as 0.0885 when label 1, 2 and 3 were one hot encoding, However, Nerual network regression had lower test rmse which was 0.031 when handling sparse features than linear regression handling sparse features. Nerual network regression was the best model for sparse features among the models we used in this experiment. The test rmse of Random forest tree regression model when handling categorical features was 0.060, which was worse than the result of k-nn. When analyzing test rmse in each work-flow, the linear regression model improved compared with analying with all work-flow, test rmse reached 0.007 in work-flow 3. We also compared the result of polinomial regression model analyzing each work-flow, polinomial regression did improve performance in each work-flow and had a lower test rmse than linear regression model, whose test rmse reached 0.0051 in work-flow 3. In conclusion, when analyzing each work-flow, polinomial regression model had a better performance than linear regression model.