# EE219 Project 4

# Regression Analysis

**Winter 2018**

**Jianfeng He (005025694)**
**Shouhan Gao (304944056)**
**ZhengXu Xia(104250792)**
**Tairan Zhu(605031908)**

**03/04/2018**

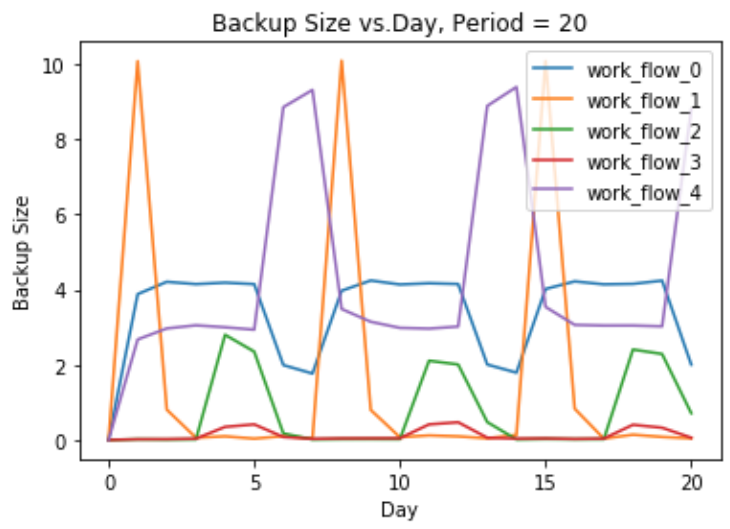
## Introduction and Problem Statement

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, the team was given a dataset which contains simulated traffic data on a backup system over a network. Some basic regression models were used on the given dataset with basic techniques to handle overfitting, which are cross-validation, and regularization. With cross-validation, overfitting can be tested. Also, with regularization, the penalization on overly complex models can be applied.

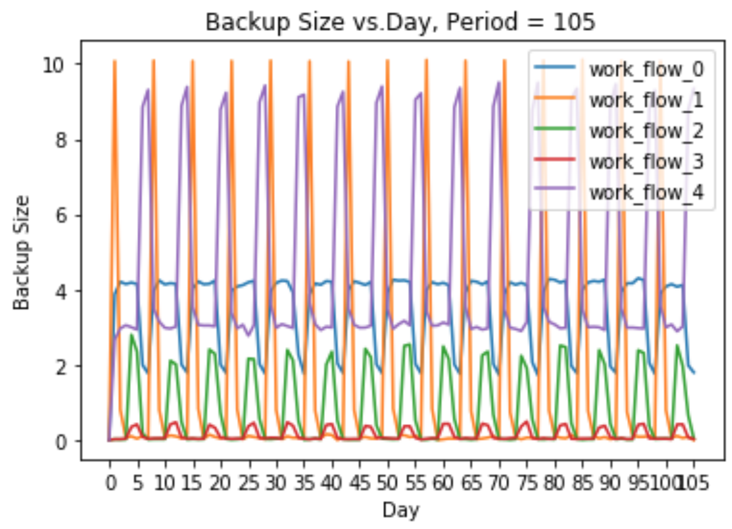
## Solutions

### Problem 1

**a. Twenty-Day Period:**



**b. 105-Day Period**



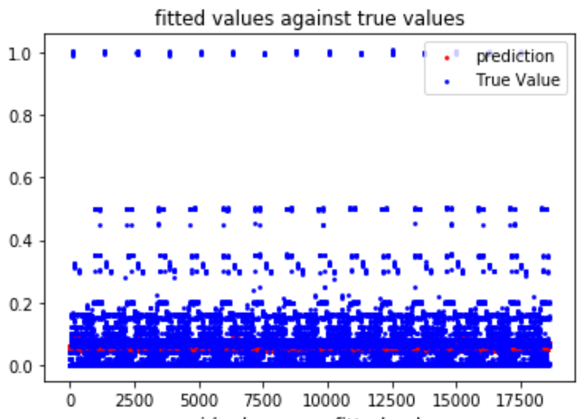
**Observation:**
From the plots, it can be concluded that the backup size for each workflow has a periodical pattern. The backup size for each workflow will reach the maximum and drop to a minimum at certain points every seven days. For example, the workflow 0 will reach a maximum at the third day of each seven-day period and drop to a minimum at the seventh day. This pattern will repeat periodically and it is similar for all workflows.

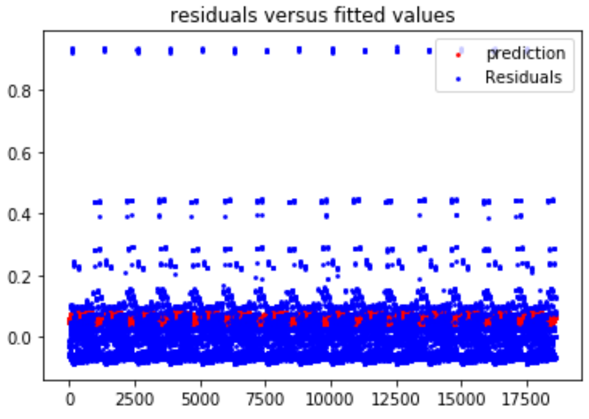
## Problem 2(a)

### (i) Fit a linear regression model

First of all, we converted each of 5 features into integers using scalar encoding method and fitted the data into linear regression model.

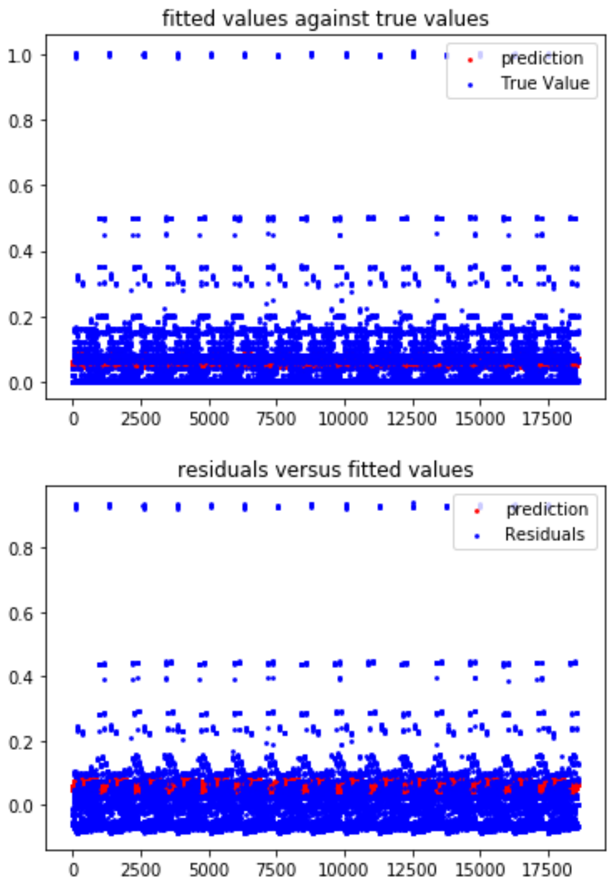
```
Training RMSE is 0.103587856885
Test RMSE is 0.103627896737
```





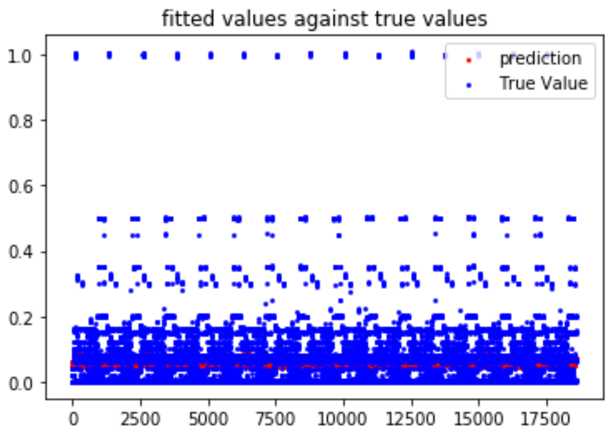
(ii) Then we standardize all these numerical features and observed that there is no difference at all. Training & Testing RMSE as well as the plot are exactly the same.

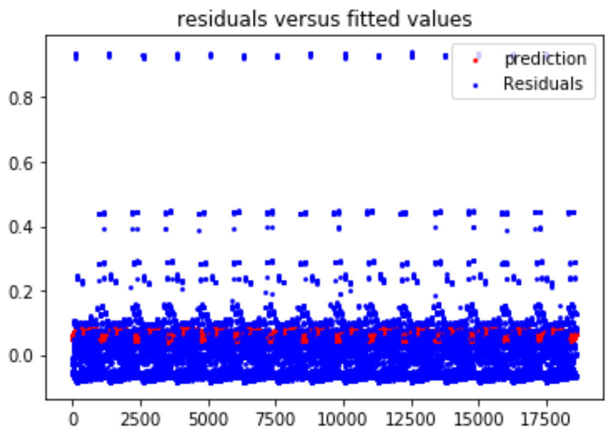
```
Training RMSE is 0.103587856885
Test RMSE is 0.103627896737
```



(iii) Using f_regression, we selected Day of Week, Backup Start Time - Hour of Day, Work-Flow-ID as our 3 most important features. We observed that test RMSE decreased a little, but from the perspective of the plot we couldn't figure out the difference in performance.

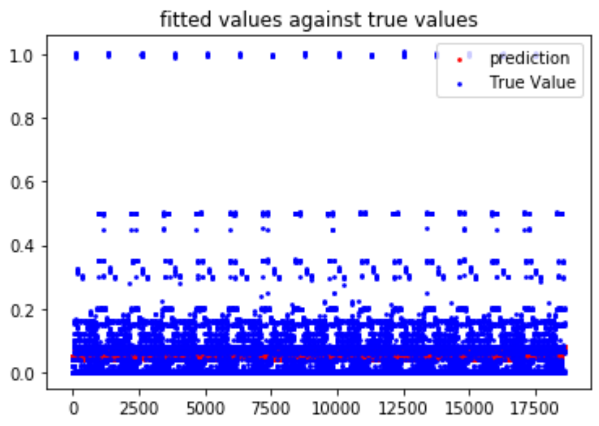
```
Training RMSE is 0.103588792478
Test RMSE is 0.103611218666
```

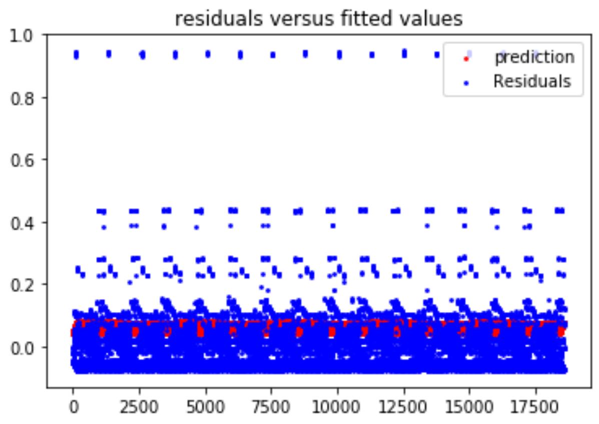




Using mutual information regression, we selected Backup Start Time - Hour of Day, Work-Flow-ID, File Name as our 3 most important features. We observed that there is a little increase for the Test RMSE and still, we couldn't figure out a difference in performance from the plot. Due to the increase of Test RMSE, we say that the performance is a little worse than before by selecting the 3 features using mutual information regression.

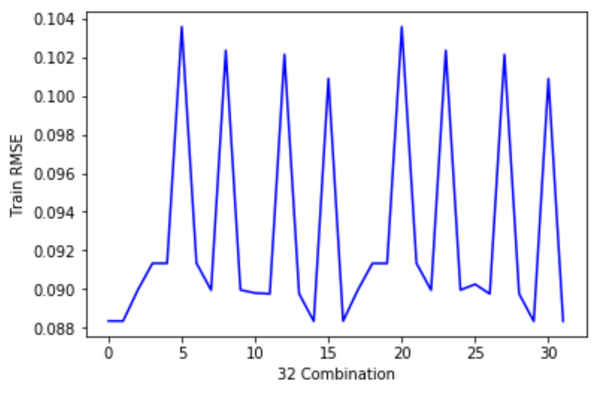
```
Training RMSE is 0.103697228859
Test RMSE is 0.103720626763
```

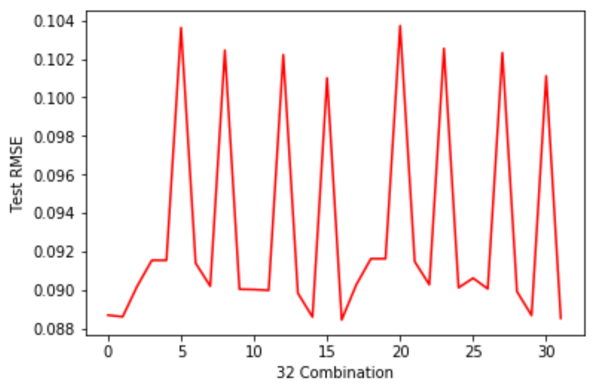




(iv) Since we can apply either scalar encoding or one-hot-encoding to each of the 5 features, there are 32 choices of combinations. Here, we utilized the mask function in which we choose whether one of the scalar-encoded feature needs to be replaced by one-hot-encoding technique. We derived the result that the best result is achieved is when the combination index is 16, which is [False, True, True, True, False], corresponding to "day of the week", "backup start time-hour of day", "work-flow ID". These three important features are consistent with our plotted pattern in question 1 and it makes a lot of sense. Firstly, from our plot in problem 1, it's not hard to find that back-up size for each work-flow has the same pattern for each week, and therefore the week index doesn't make a difference for the prediction of the pattern. Secondly, different work-flow has different back-up size, and the work-flow is from files that has similar change in term of size over time. It means the back-up size depends on work-flow index instead of file name. Of course, each work-flow ID should have different back-up size at a certain time in a day and certain day in a week. Therefore, week index and file name are two unnecessary features in the prediction model.

The training & testing RMSE plot and data, as well as the best combination RMSE and plot are shown below.





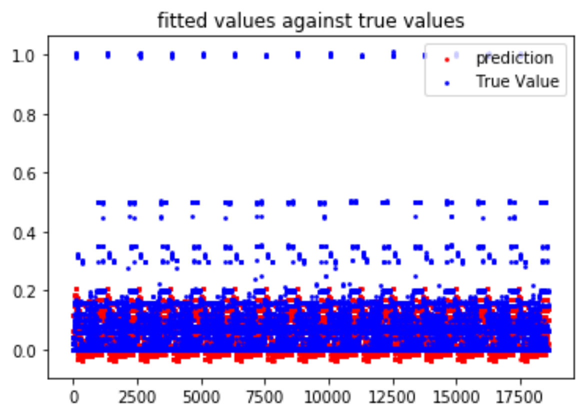
```
print (avg_train_rmse)
print (avg_test_rmse)
```

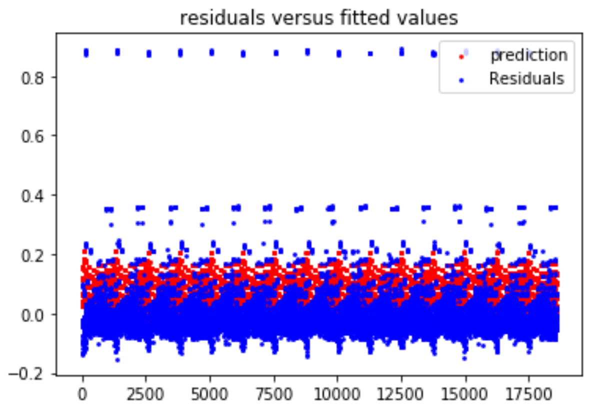
```
[0.08835003322992036, 0.0883454334034843l6, 0.089949398288239238, 0.091335346446430848, 0.091335148315303
677, 0.10358785688545552, 0.091344223729349305, 0.089946228445639098, 0.10236174932859557, 0.0899547875339
18681, 0.089795473497471801, 0.089751920353005693, 0.10215018581773144, 0.08975889667101529, 0.08833368208
1962836, 0.10090327007275081, 0.088340896889768872, 0.089946498878188813, 0.09133236449615563, 0.091331341
66902851, 0.10358349821383288, 0.091340043276389823, 0.089941636806100983, 0.10235712265017191, 0.08995062
6172803097, 0.090249169912603053, 0.089750311657207343, 0.10214804155006828, 0.089756508991091916, 0.08833
1351829880139, 0.10090061286862086, 0.088338269223304663]
[0.088684715423578642, 0.088600321298543672, 0.090196082589475798, 0.091536108253037388, 0.091536504422881
179, 0.1036278967366156, 0.091386853734248893, 0.090192510079847715, 0.10244970466132744, 0.090033705053452
498, 0.090016171876337814, 0.089976263278906496, 0.10221988970281136, 0.089827280569240039, 0.088590252185
599308, 0.10101541988610339, 0.088439534225397270, 0.090268925146982482, 0.091618650728590781, 0.0916174014
3972559, 0.10372798831044303, 0.091470126798306378, 0.090270235413343974, 0.10254806145200702, 0.090113345
4457827, 0.090609723756832711, 0.090051961482817702, 0.10232376156565623, 0.089914693455842076, 0.08867194
3212582455, 0.10111842225242033, 0.088523292588389729]
```

Best Performance Combination: 16

```
Training RMSE is 0.0883408968898
Test RMSE is 0.0884395342254
```

We can observe something from the Test RMSE and scattered value plots shown below. The test RMSE has an obvious drop compared to the previous example, which means it has a better prediction performance. Additionally, the fitted value are more evenly distributed among true values and residuals. The fitted value also has taken a larger portion of the true value or residual value compared to previous examples. This suggest that the model is better fitted in this optimized example (best combination). In other words, our best combination of features achieves an obvious increase in performance.

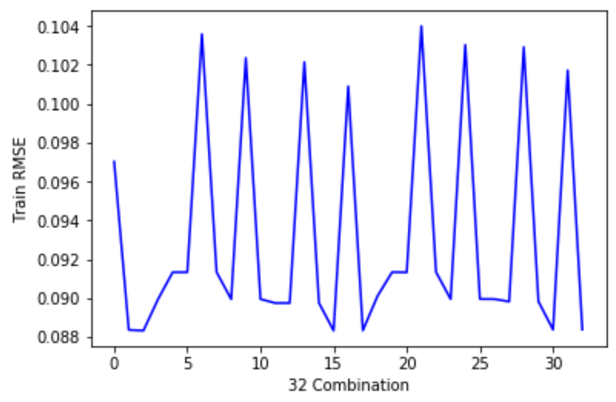




(v) Surprisingly, we didn't observe a big difference in testing and training RMSE in each of the combination. The reason is probably that our data is already well fitted and is already ready to use for the model. Then we applied three regularization methods to see if the performance has improved. In the Ridge Regularizer and Lasso Regularizer, we set a tunable range for alpha in the model, and in Elastic Net Regularizer, we set a tunable range for l1_ratio.

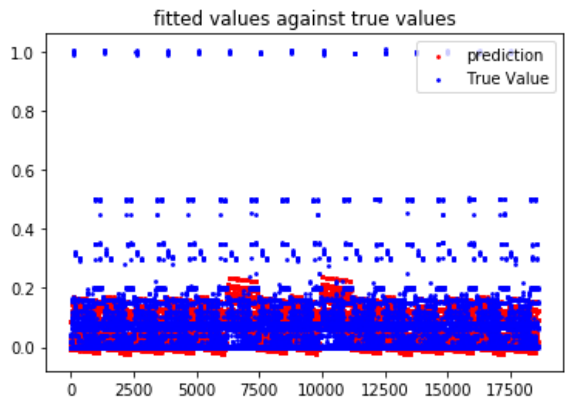
**For Ridge Regularizer:**

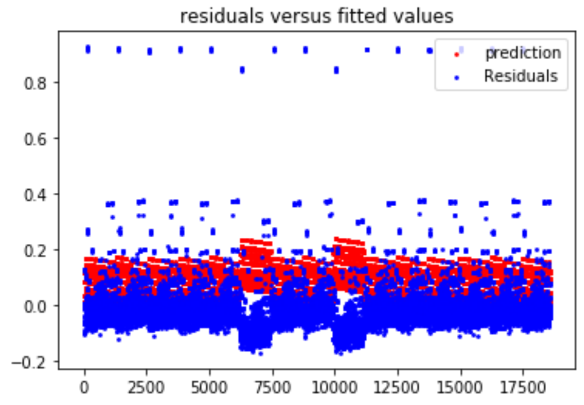
```
Best Combination Index: 17
Optimal Alpha 0.001
Optimal Coefficients
 [  4.01350066e-02  -1.28814112e-02  -2.00696523e-02  -5.55062586e-03
  -5.45763609e-03   2.97438087e-03   1.33123719e-03  -2.02387157e-02
  -2.14722877e-02   7.65400832e-03   3.35264915e-02  -2.23979290e-03
   2.24732228e-03   3.90865010e-02  -1.30613609e-02  -4.06695895e-02
  -5.69478212e-02   7.20127555e-02   4.07057142e-05   4.51866938e-05]
```



Using the best combination:

```
Training RMSE is 0.088340815393676553
Test RMSE is 0.088439524898406036
```

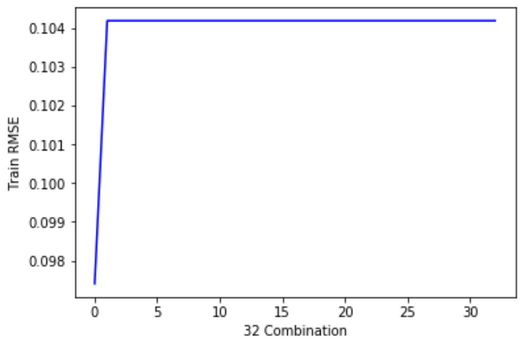


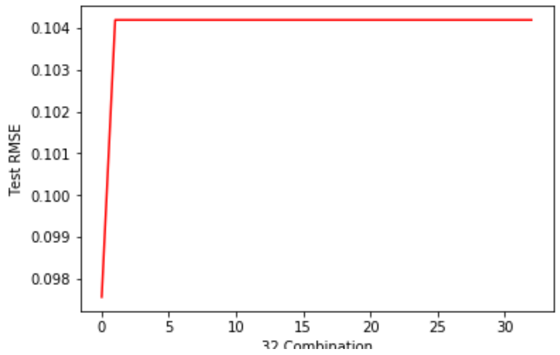


We observe that, with Ridge Regularizer, the test RMSE is almost the same, but the plot is different, the fitted value is no longer evenly distributed, which means that the performance is a little worse than before, but overall it still has a good performance.

**For Lasso Regularizer:**

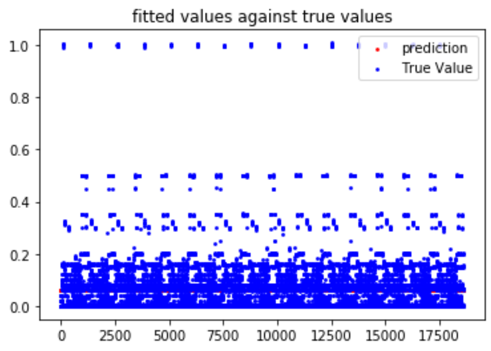
```
Best Combination Index: 0
Optimal Alpha 100.0
Optimal Coefficients
 [  6.01259676e-05  -2.29689838e-03   1.26966630e-03   1.40598425e-03
   1.62594379e-04]
```

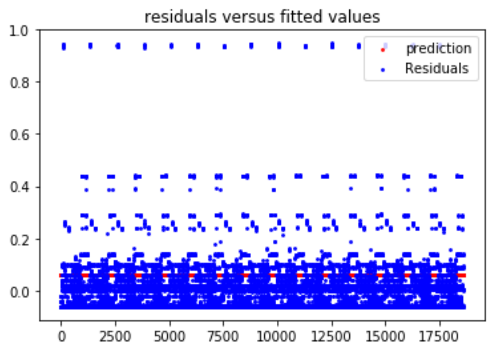




Using the best combination:

```
Training RMSE is 0.097405472897829318
Test RMSE is 0.097564694637076552
```

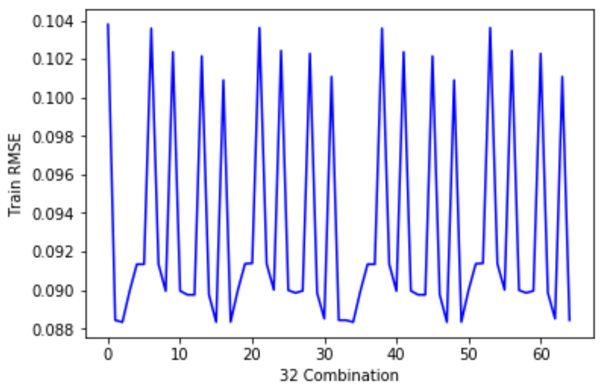


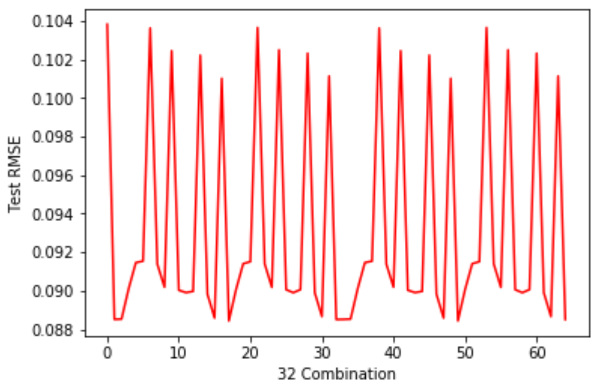


We can observe that Lasso Regularizer is a really bad model to fit those data. Firstly, the average test RMSE achieves a peak value and keep unchanged after the first combination and the test RMSE value is really high. Additionally, the fitted value in the scattered plot only takes a very small portion of true data, which suggests a really bad performance. Therefore, we should not choose Lasso Regularizer.

**For Elastic Net Regularizer:**

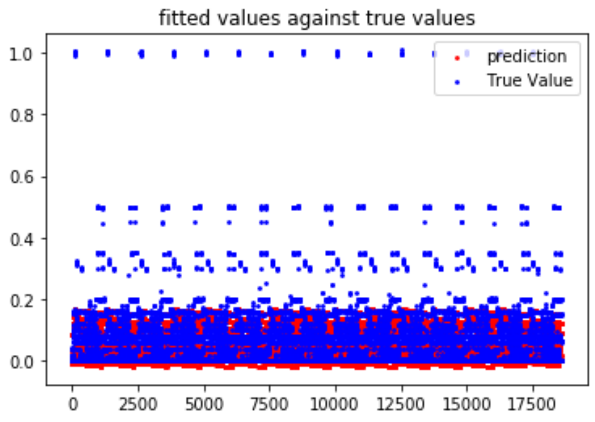
```
Best Combination Index: 17
Optimal l1_ratio 0.9
Optimal Coefficients
 [  4.01350066e-02  -1.28814112e-02  -2.00696523e-02  -5.55062586e-03
  -5.45763609e-03   2.97438087e-03   1.33123719e-03  -2.02387157e-02
  -2.14722877e-02   7.65400832e-03   3.35264915e-02  -2.23979290e-03
   2.24732228e-03   3.90865010e-02  -1.30613609e-02  -4.06695895e-02
  -5.69478212e-02   7.20127555e-02   4.07057142e-05   4.51866938e-05]
```

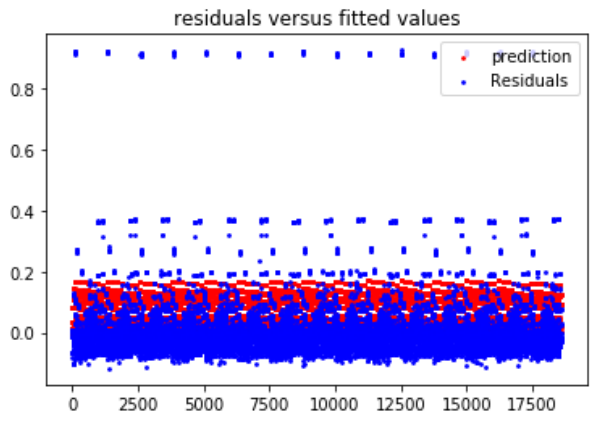




Using the best combination:

```
Training RMSE is 0.08834148219380436
Test RMSE is 0.08843471260844842 9
```



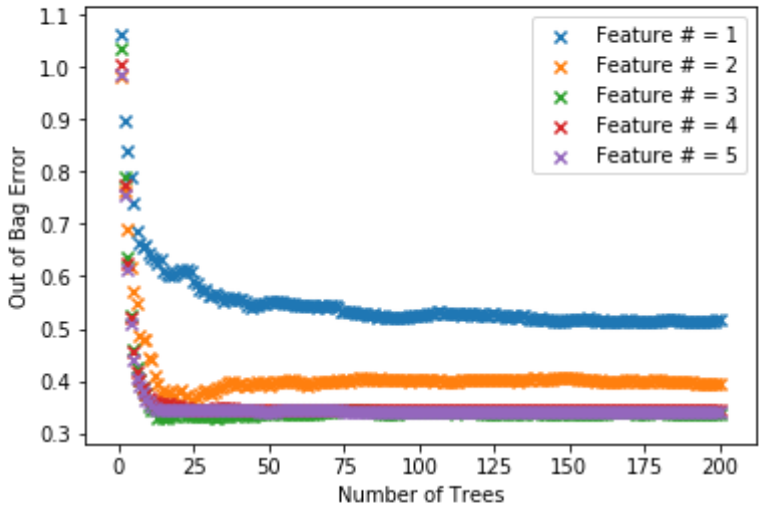


We can observe that for Elastic Net Regularizer, its performance is still approximately the same as Ridge Regularizer or without any regularizer. So overall, its performance is still relatively good.

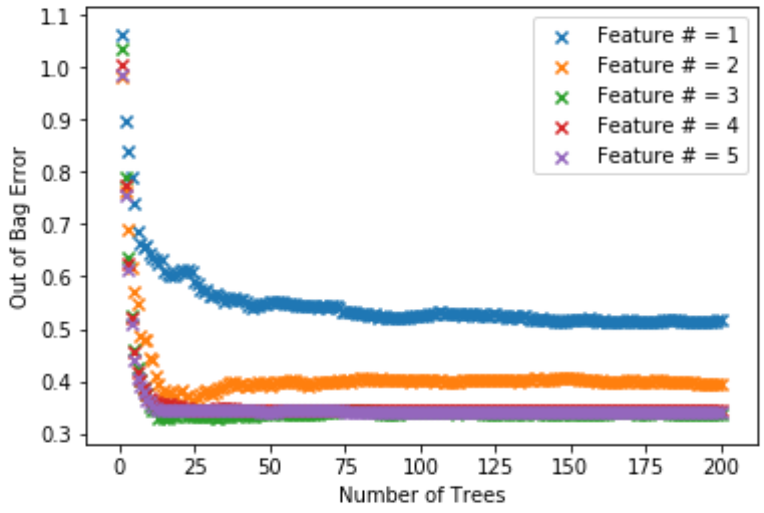
**2(b). Fit a random forest regressor**

**i.** In the initial random forest regression model, we had number of trees equal to 20, depth of each tree equal to 4, number of features considered at each split is 5, and with bootstrap turned on. We firstly converted all features to numerical encodings. After performing 10-fold cross validation, **average training RMSE** is **0.0604832481886** and **average test RMSE** is **0.0607882493747**. **Out of bag error** of the initial model is **0.343369776393**.

**ii.** We swept over number of trees from 1 to 200 and maximum number of features from 1 to 5 by performing 10-fold cross validation on random forest regression model. Then we collected a list of OOBE and average test RMSE. The first figure below is plotted as out of bag error (y axis) against number of trees (x axis) and the second figure is for average Test-RMSE(y axis) against number of trees(x axis). We used colors to represent the maximum number of features considered at each split.



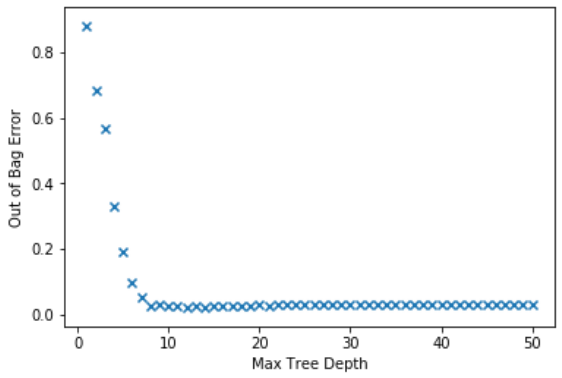
| Minimum OOBE for each feature number | | |
|---|---|---|
| Feature Number | Tree Number | Min OOBE |
| 1 | 195 | 0.5127621362722362 |
| 2 | 25 | 0.36287171124995976 |
| 3 | 15 | 0.3290470385528568 |
| 4 | 156 | 0.3438164482663869 |
| 5 | 154 | 0.34059952477559885 |

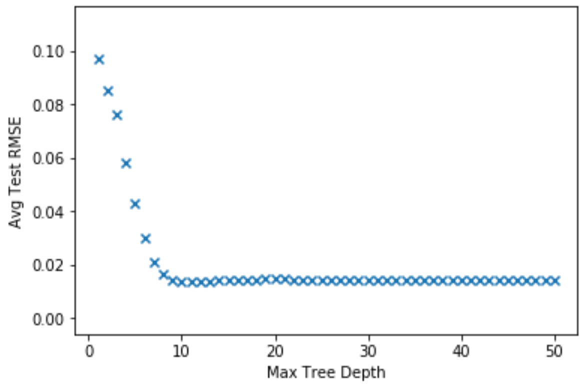


| Minimum average Test RMSE for each feature number | | |
|---|---|---|
| Feature Number | Tree Number | Min Average Test RMSE |
| 1 | 2 | 0.06989289154944507 |
| 2 | 2 | 0.05978983992142693 |
| 3 | 13 | 0.05819036858891585 |
| 4 | 24 | 0.06085225512283422 |
| 5 | 3 | 0.06046859957584057 |

We found that both average test RMSE and OOBE exponentially decrease when the number of tree reaches around 20 and then stabilize at a number no matter how large the tree number is. The number of features considered at each node split can also influence the performance of random forest regressor. If the number of features is small, then the information known at each tree node is limited so that information gain at each node is also limited. However, it does not mean that the more features a node split can consider the better the performance. Our results show that both average test RMSE and OOBE reach their minimum values respectively when the number of features equal to **3** and the number of trees approximately equal to **11**. When the feature number and tree number are too large, the results turn out not be the best.

**iii.** We then set maximum of number of features considered each split to 3 and tree numbers to 11. We swept tree depth from 1 to 50 and generated the relationship between oobe and tree depth and that between average test RMSE and tree depth.



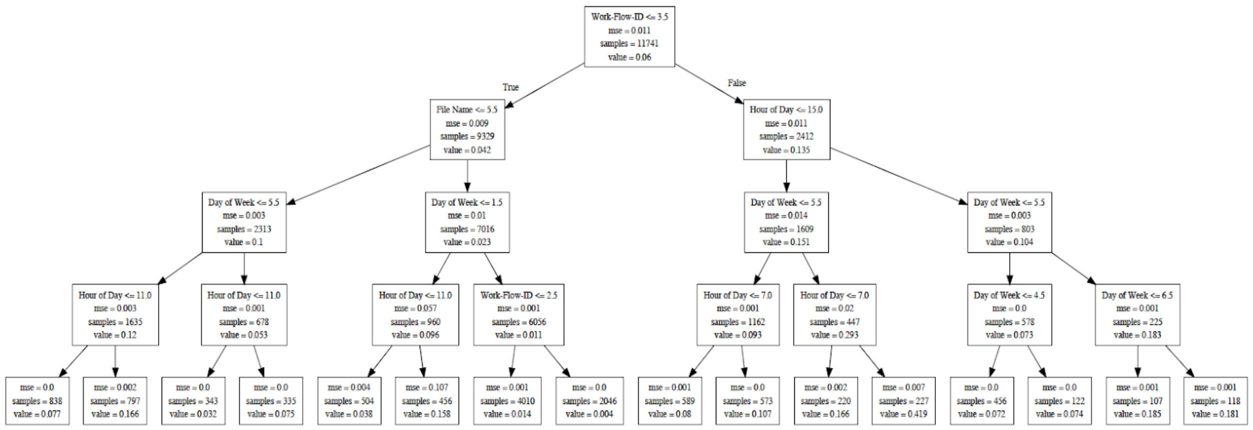


We found that **minimum oobe** is **0.021633894265363884** when **tree depth** is **12** and the **minimum average test RMSE** is **0.013427039563200495** when **tree depth** is **11**. We decided to choose **11** as the **tree depth** in best parameter set.

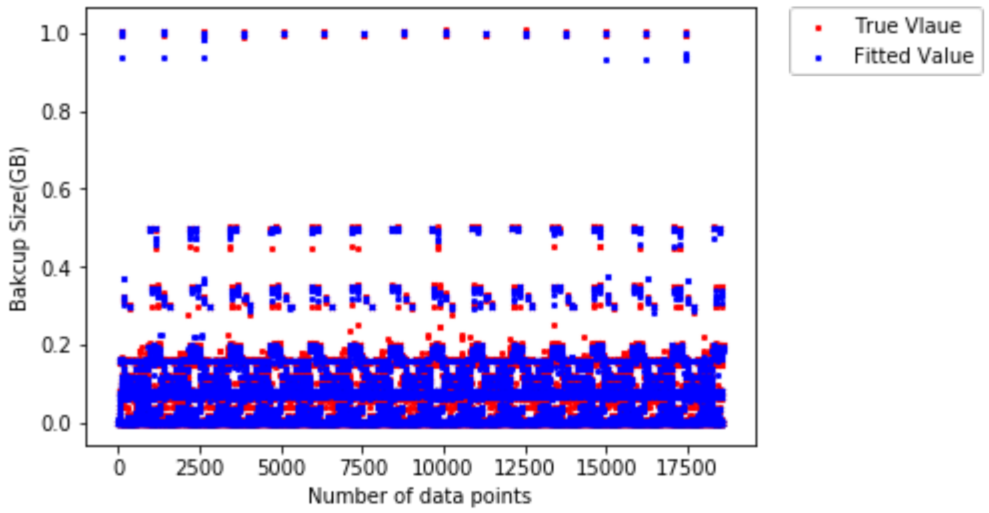
**iv.** We fitted the entire dataset in a random forest regression model with **maximum number of features** equal to **3**, **number of trees** equal to **13**, **tree depth** equal to **11**.

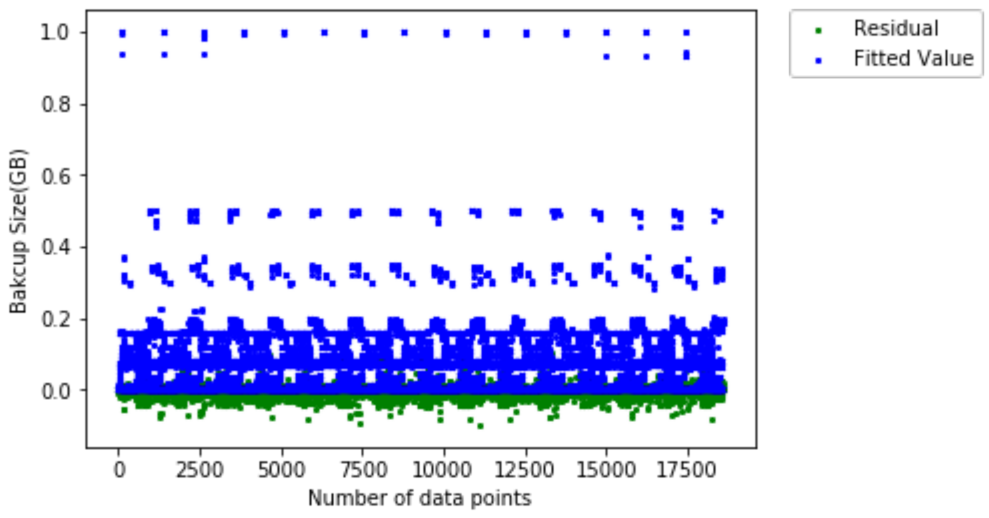
| Feature | Importance |
|---|---|
| Backup Start Time - Hour of Day | 0.35270485 |
| Day of Week | 0.29365164 |
| File Name | 0.19330782 |
| Work-Flow-ID | 0.15492203 |
| Week # | 0.00541366 |

**v.** We fitted the entire dataset in a random forest regression model with maximum number of features equal to 3, number of trees equal to 13, tree depth equal to 4. Then we plotted the first tree in the decision tree list of fitted random forest regressor as figure shown below. However, the root node is **work flow id** instead of **backup start time-hour of day** for the first tree. Then we checked other decision trees we found filename appears 6 times out of 13 at the root node while work flow id appears 5 times out 13 and day of week appears twice out of 13. Then we did feature importance analysis for this model, we found feature importance is **filename > day of week > workflow id > week day > start hour**.
Since the feature importance in sci-kitlearn is calculated by how purely a node separates the classes (Gini index). Also, the feature importance is the weighted average of GINI index over all nodes and among all trees. "The most important feature might not have to be the root node of every tree. Just because a node is lower on the tree does not necessarily mean that it is less important. The relative rank (i.e. depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the target variable. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The **expected fraction of the samples** they contribute to can thus be used as an estimate of the **relative importance of the features**. By **averaging** those expected activity rates over several randomized trees one can **reduce the variance** of such an estimate and use it for feature selection." [1]



With the best parameters we found, max tree number is 13, max feature number is 3, max tree depth is 11 and with bootstrapping. Here are the comparisons between fitted results and true values and between residuals and fitted values:





**2(c). Use a Neural Network Regression Model with All Features**

In this section, a neural network regression is used to examine all features one-hot encoded. The relationship between the number of hidden units and activation functions is investigated in terms of test rmse. The range of hidden units we examine is [5,10,20,50,100,200,400].

According to the figure of Activation Functions Vs Number of Neurons, the test rmse calculated by logistic function and tanh function remains relatively high and stable. The reason is that logistic function outputs either 0 or 1, and relu function returns range from -1 to 1. Since the true values range from 0 to 1, logistic function can only return what classes each data point belong to. In other words, logistic function is well-performed when the input data is binary. For tanh activation function, it returns values from -1 to 1. Therefore, tanh function is good to predict values that contain negative numbers. For relu activation function, its test rmse curve goes down as the number of neurons increase. In conclusion, the best combination of activation function and the number of neurons are **relu function with 200 hidden units.** Its **test rmse** is about **0.038**.

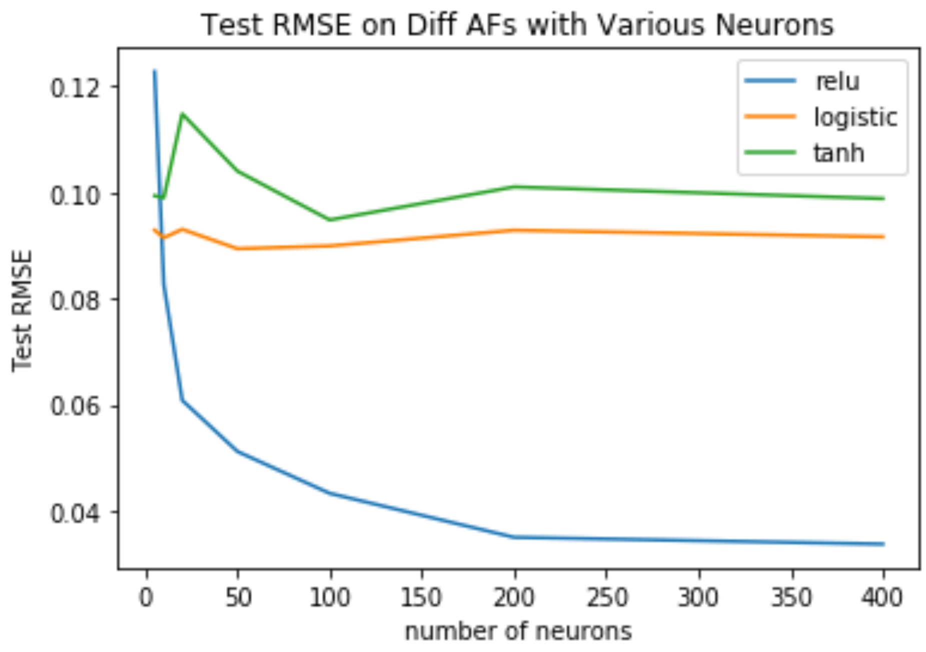
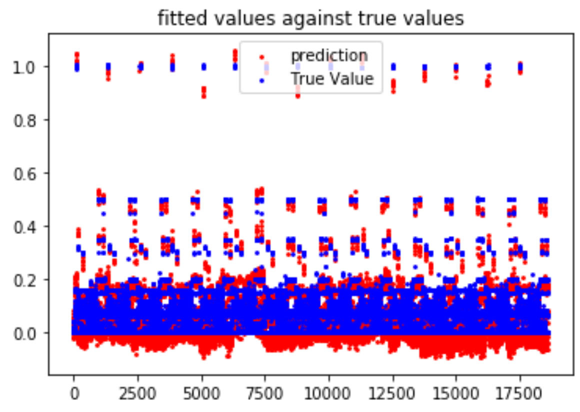


Figure. Activation Functions Vs Number of Neurons

Using Relu activation function and 200 hidden units, one may tell the predicted values in red are very close to true values. Based on the plot of residuals vs fitted values, the residuals values are close to zero, which means the result is accurate.



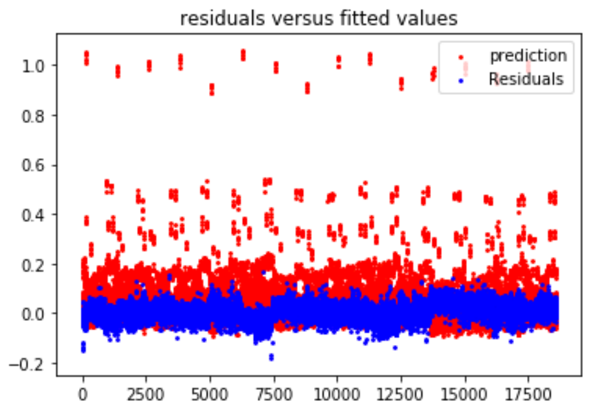


Figure. Fitted Values Vs True Values

Figure. Prediction Vs Residuals

**2(d). Predict the Backup Size for Each of the Work Flow**

**i. Using Linear Regression Model**

In this section, we examine the train rmse and test rmse for each work flow, and check if the fit is improved or not.

In general, the linear regression model is improved in terms of rmse values. According to two tables below, the values of rmse for each work flow are generally smaller than those for all work flow. The lower rmse is, the more accurate the result it returns.

| | Work Flow 0 | Work Flow 1 | Work Flow 2 | Work Flow 3 | Work Flow 4 |
|---|---|---|---|---|---|
| Train RMSE | 0.03584 | 0.1488 | 0.04291 | 0.007244 | 0.08592 |
| Test RMSE | 0.03589 | 0.1489 | 0.04307 | 0.007261 | 0.8599 |

Table: Train and Test RMSE for Each Work Flow

Figures below are plots of Fitted Values Vs True Values and plots of Residuals Vs Fitted Values:

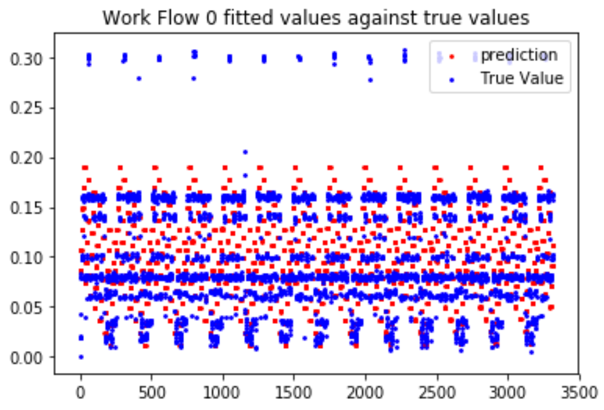


Figure. Work Flow 0 Fitted Values Vs True Values

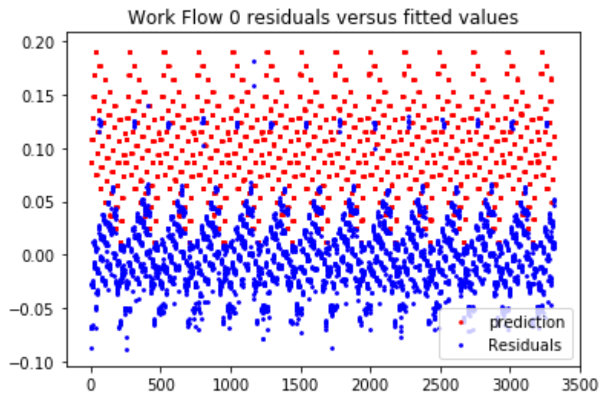


Figure. Work Flow 0 Prediction Vs Residuals

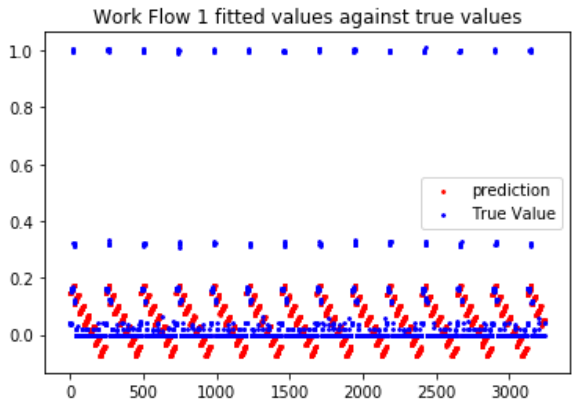


Figure. Work Flow 1 Fitted Values Vs True Values

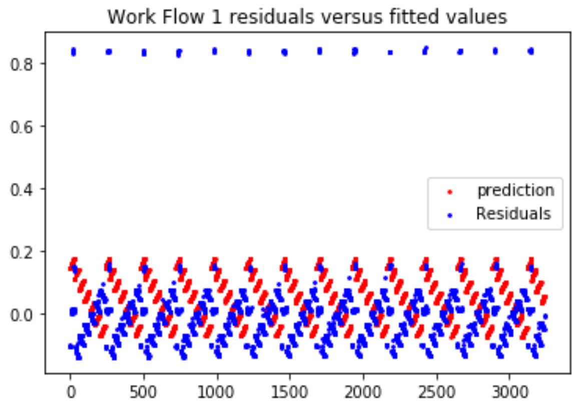


Figure. Work Flow 1 Prediction Vs Residuals

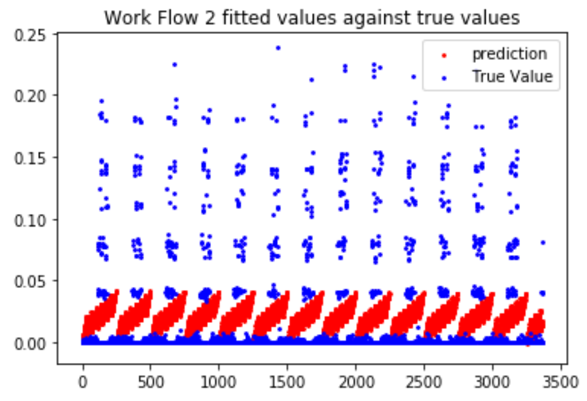


Figure. Work Flow 2 Fitted Values Vs True Values

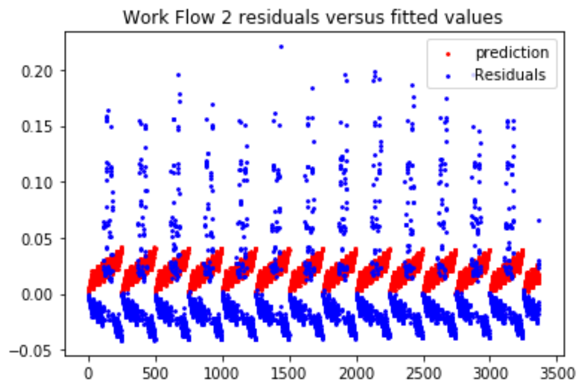


Figure. Work Flow 2 Prediction Vs Residuals

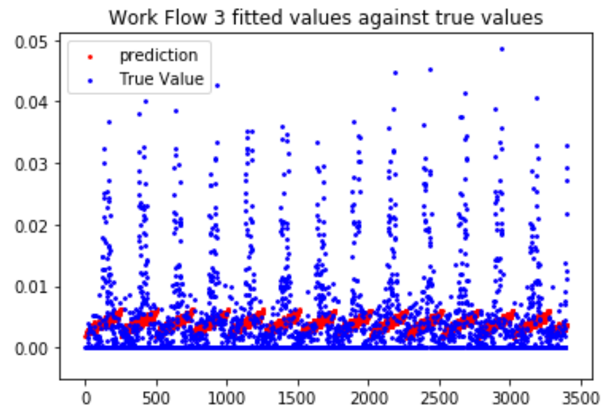


Figure. Work Flow 3 Fitted Values Vs True Values

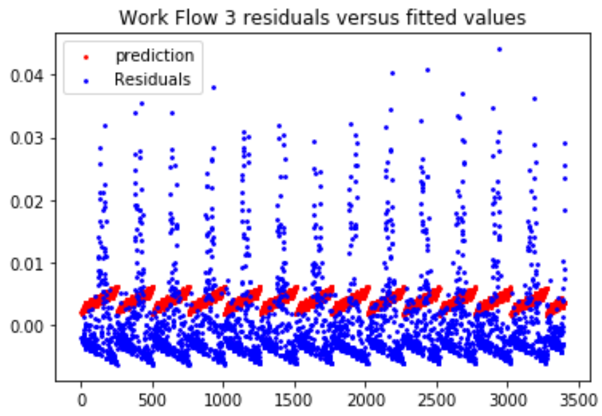


Figure. Work Flow 3 Prediction Vs Residuals

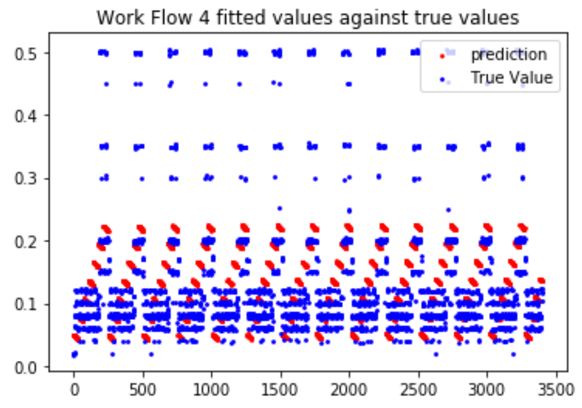


Figure. Work Flow 4 Fitted Values Vs True Values

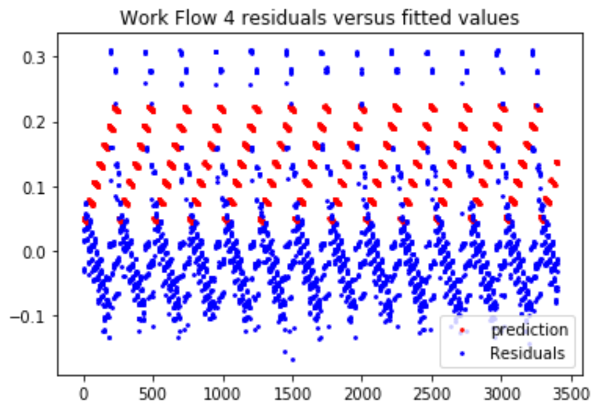


Figure. Work Flow 4 Prediction Vs Residuals

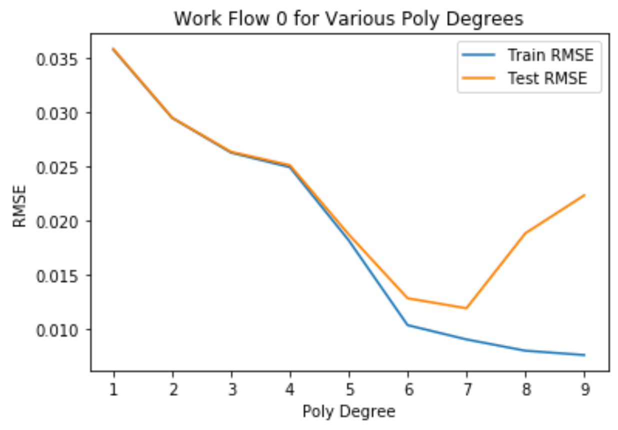
**2(d).ii Use Polynomial Regression to Improve the Fit**

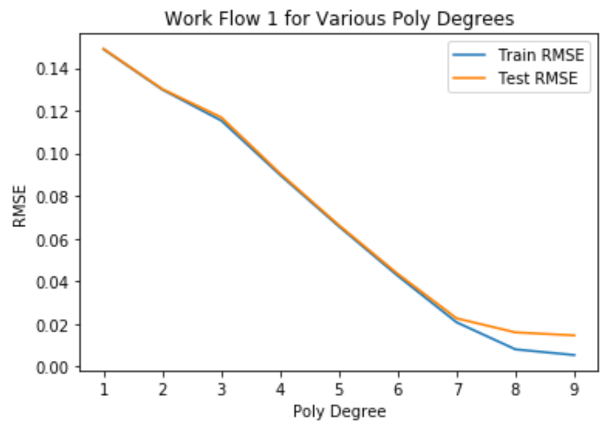
In this section, we apply polynomial regression for each work flow, and investigate the values of rmse as the degree of the polynomial function increases from 1 to 10. In the end, we try to find out the threshold at which the fit starts to turn bad.

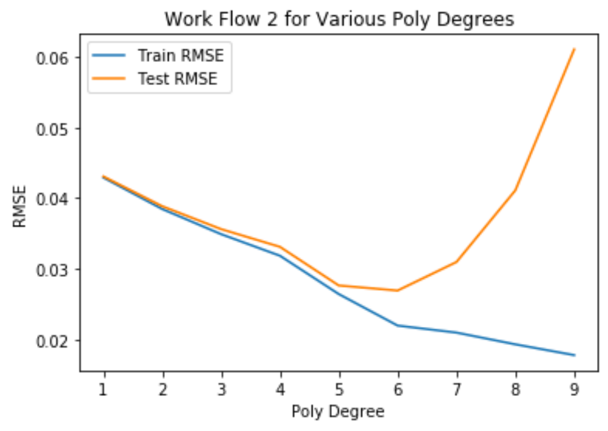
The thresholds for each work flow is shown in the below table. The methodology to find threshold is based on the turning point where test rmse starts to increase. The theory of how cross validation helps controlling the complexity of our model is to check if our model is overfitted based on train and test rmse. The sign of overfitting is a falling training error and a rising test error. If our model uses lots of features to be trained, it will have high probability to be overfitted, and its training rmse gets smaller. In other words, since lots of features are used in training data, less data is used for testing; as a result, the testing rmse will rise. In conclusion, the point where test rmse starts to increase is the balance point of our model.

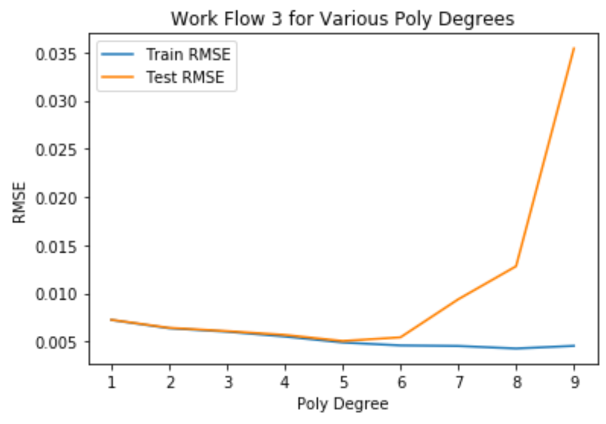
| | Work Flow 0 | Work Flow 1 | Work Flow 2 | Work Flow 3 | Work Flow 4 |
|---|---|---|---|---|---|
| Threshold | 7 | 6 | 6 | 6 | 5 |

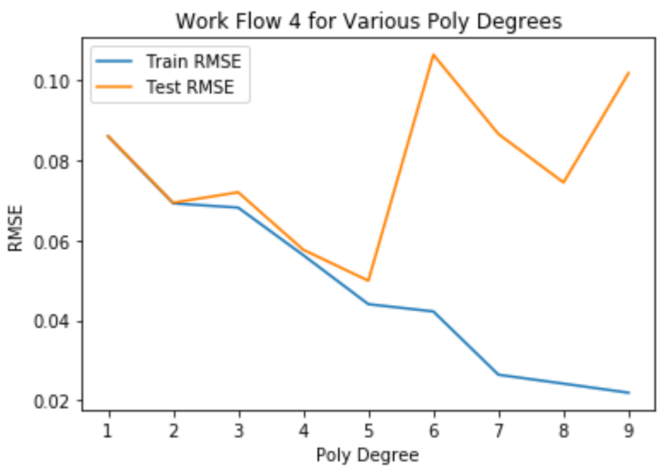
Table: Threshold for each work flow











The training and testing rmse for each work flow is shown below.
For work flow 0, its rmse ranges from 0.010 to 0.035.
For work flow 1, its rmse ranges from 0.010 to 0.14.
For work flow 2, its rmse ranges from 0.010 to 0.06.
For work flow 3, its rmse ranges from 0.005 to 0.035.
For work flow 4, its rmse ranges from 0.002 to 0.010.

Work Flow 0
Train RMSE:  [0.035835520779861095, 0.029518915091185507, 0.026309547124530652, 0.024962238121049168, 0.018209437791092842, 0.010377933173032755, 0.0090600017375540715, 0.0080170610951268706, 0.0076181827132134824]
Test RMSE:  [0.035886970248931206, 0.029540009203662444, 0.026387534977949535, 0.025150666805501116, 0.018754772222202461, 0.01286119329224818, 0.011939365899111198, 0.018864605536721406, 0.022369544365290719]

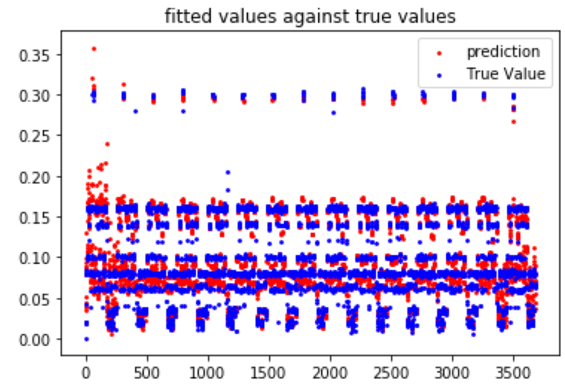
Work Flow 1
Train RMSE:  [0.14876603056260168, 0.12984413828137184, 0.11533001582391803, 0.089831545293320714, 0.065707659350818626, 0.042532372345961175, 0.020756415526454696, 0.0080810322236137717, 0.0054364203943832847]
Test RMSE:  [0.14891860201393806, 0.13009650190777197, 0.11685267028724416, 0.090557237690219602, 0.066301237417921016, 0.043466496234742052, 0.022619779364947669, 0.016050811167531709, 0.014643568483903132]

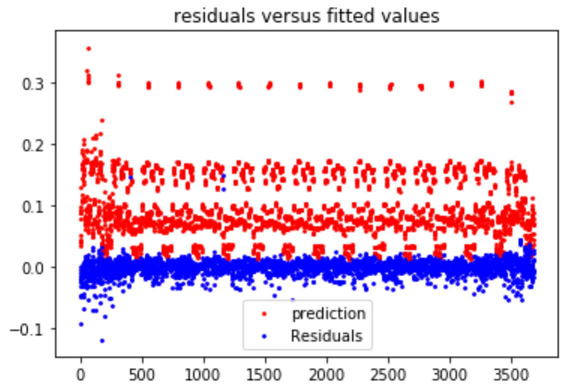
Work Flow 3
Train RMSE:  [0.0072438788738825345, 0.0063798475163659432, 0.0060245160864939281, 0.0055220430886261064, 0.0049049498397625161, 0.0046014427484824712, 0.0045420345518851404, 0.0042813021741649592, 0.0045562057771475883]
Test RMSE:  [0.007260894242099694, 0.0064263656987339551, 0.0060859324174098294, 0.005693827031599567, 0.0050616657599734599, 0.0054393928071319372, 0.0093861747658765416, 0.012811881841530421, 0.035398013095147375]

Work Flow 4
Train RMSE: [0.085921936793271939, 0.069192398447646422, 0.068123526307968404, 0.056254736755275259, 0.044011832283068625, 0.04218488638665472, 0.026385301881806572, 0.024167134839290935, 0.021889831408209905]
Test RMSE: [0.085990614115654451, 0.069340664871216928, 0.071957999214204246, 0.05759587097304518, 0.049870096568613201, 0.10632152524330896, 0.086491071783058801, 0.074409973012360034, 0.10172874026330511]

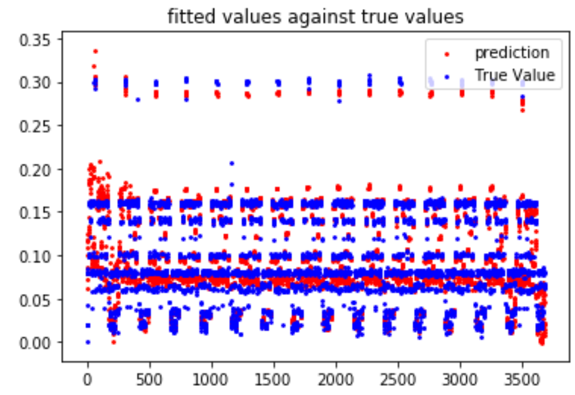
Figures below are plotted for each work flow with its corresponding threshold. Most of the predicted values are close to true values, and their residual values are approximately equal to zero. Therefore, the polynomial regression performs well on each work flow with its threshold

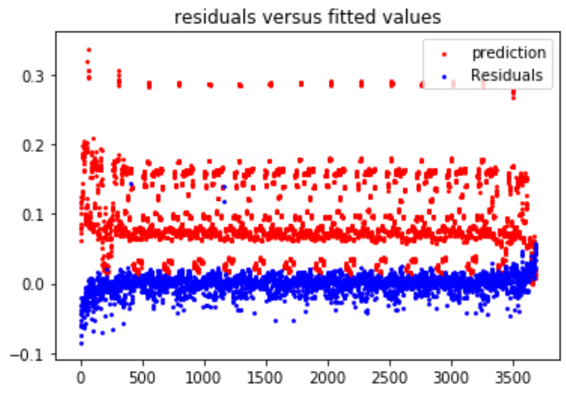
Work Flow 0 with Poly Degree = 7



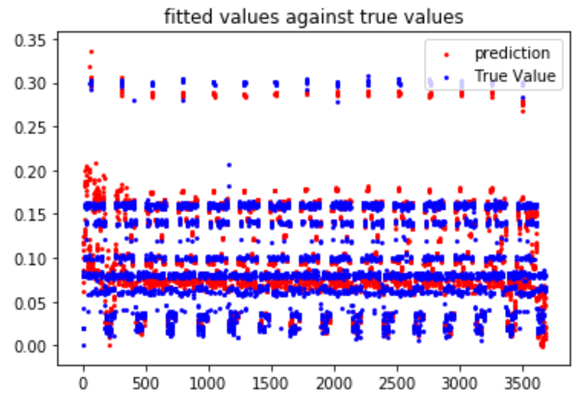


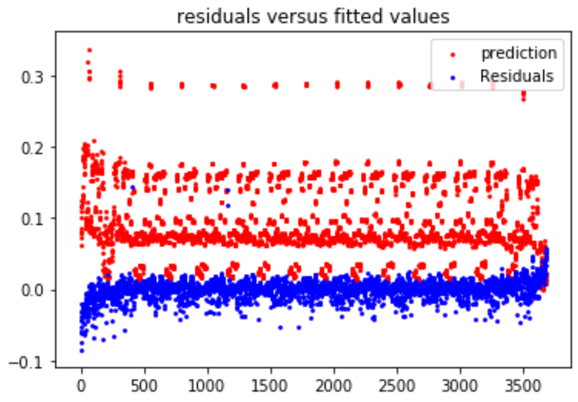
Work Flow 1 with Poly Degree = 6



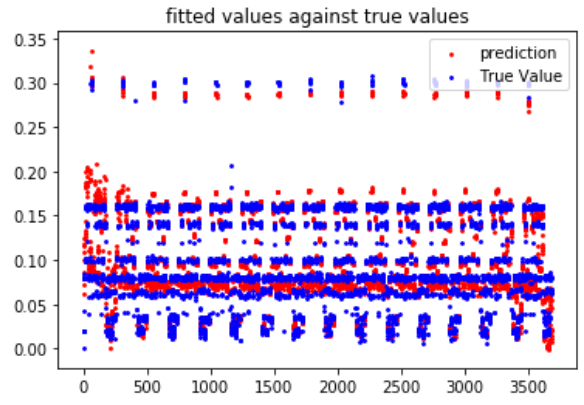


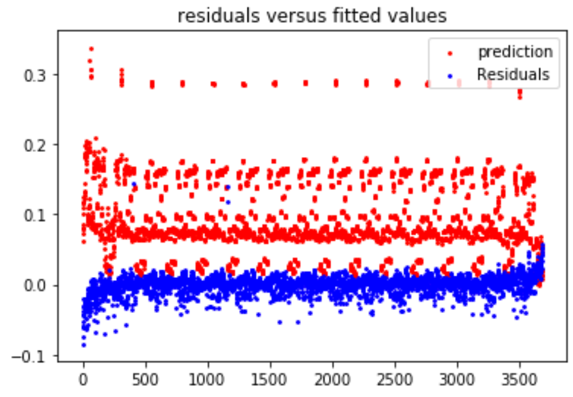
Work Flow 2 with Poly Degree = 6



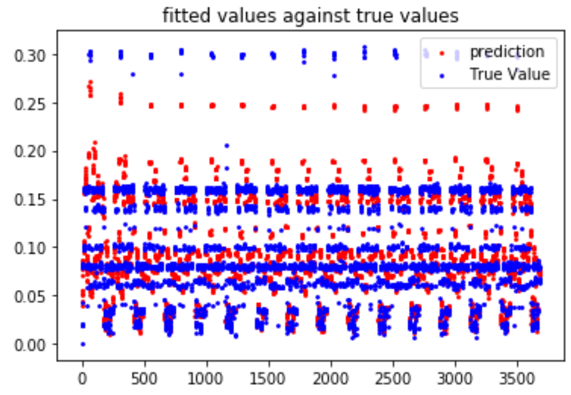


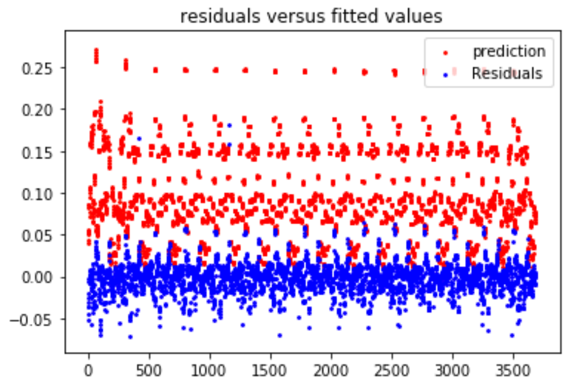
Work Flow 3 with Poly Degree = 6





Work Flow 4 with Poly Degree = 5

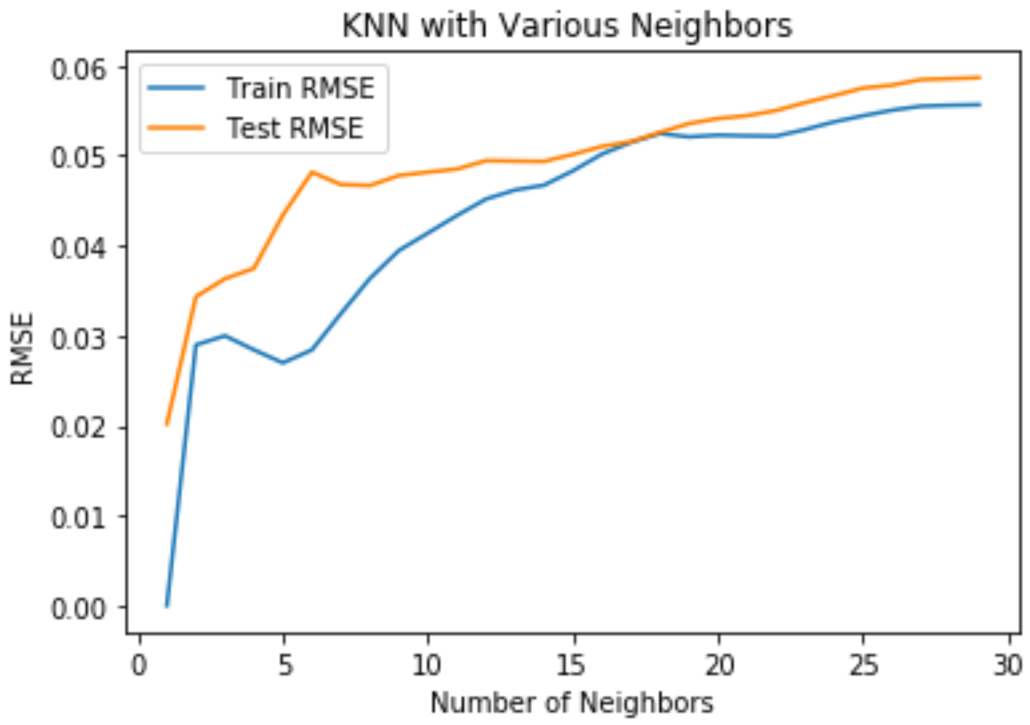




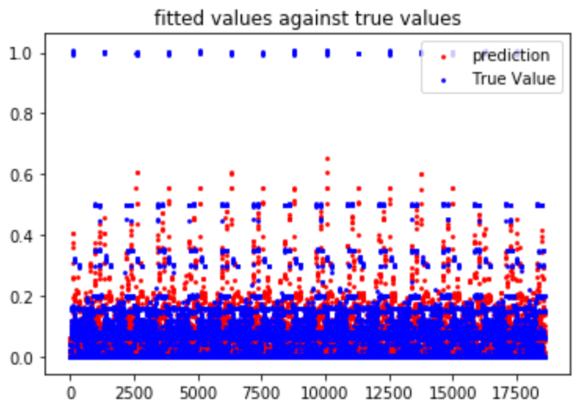
**2(e). User K-Nearest Neighbor Regression and Find the Best Parameter**

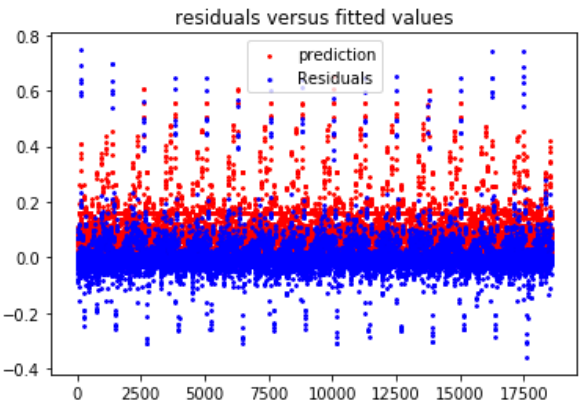
In this section, we investigate the relationship between the number of neighbors in KNN and the values of rmse. In this case, we test the model in the range of 30.

The training and testing rmse both increase as the number of neighbors increase, and they converge to **0.06** as k is greater than **20.** Therefore, increasing the number of neighbors greater than 20 will not return much more satisfying result. In this case, the **best parameter** is **20**.



The figures below are the Fitted Values Vs True Values and Residuals Vs Fitted Values with the number of neighbors = 20.





## Problem 3

Question: Compare these regression models you have used and write some comments, such as which model is best at handling categorical features, which model is good at handling sparse features or not? which model overall generates the best results?

Answer: In this project, we firstly apply linear regression model for all features with various conditions. Implementing standardization on linear regression model does not improve the result, whereas f-regression and mutual information regression have further improved the overall performance, since these methods help select the most important features. In addition, implementing 32 combinations of encoding the five categorical variables also returns satisfying result, because some combinations of important variables can yield better outcome. Ridge Regularizer and Elastic Net Regularizer can slightly improve the result, whereas Lasso Regularizer cannot increase the accuracy. Secondly, implementing random forest regression can handle categorical variables without using one-hot or scaler encoding. The most three important features acquired by random forest are filename, day of week, and workflow id. Thirdly, selecting Relu activation function with 200 hidden units in neural network is the best combination in the implementation of neural network in this case. The testing rmse is greatly reduced compared to linear regression. Fourthly, applying polynomial function on each of the work flow also can yield promising results. In this step, we find out threshold for each of the work flow, and it also improves the overall performance. At last, we apply k-nearest neighbors to visualize the rmse over the number of neighbors. The training and testing rmse stabilizes at k = 20, and the result will not be improved with increasing k.

In conclusion, random forest regression is good at handing categorical features, because it does not need one-hot or scalar encoding; linear regression with regularizers is good at handling sparse features. Random forest regression generates the best results, since it can effectively select the most important features and the result is very satisfying with large value in the number of trees and the depth of the tree. Polynomial function can effectively avoid overfitting and generate a good result. Neural Network with one hidden layer can also improve the performance.

Reference

1. http://scikit-learn.org/stable/modules/ensemble.html#forest