EE219 Project 4
Regression Analysis
Winter 2018

Shaoming Cheng, 505034686
Yao Xie, 105036239
Jiahui Li, 004356402
Ruiyi Wu, 304615036

1 Introduction

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationship among variables. More specifically, regression analysis helps one understand how the typical value of a dependent variable changes when any of the independent variables is varied, while the other independent variables are held fixed. In this project, we applied different types of regression models for analysis on a given dataset, namely linear regression model, random forest regression model, polynomial regression model and piecewise linear regression model. We also solved overfitting problem by applying parameter regularization. With cross-validation, we examined and compared the performance of all regression models.

2 Problems

1. Load the dataset

(a) For a twenty-day period (X-axis unit is day number), we plot the backup sizes for all workflows (color coded on the Y-axis).

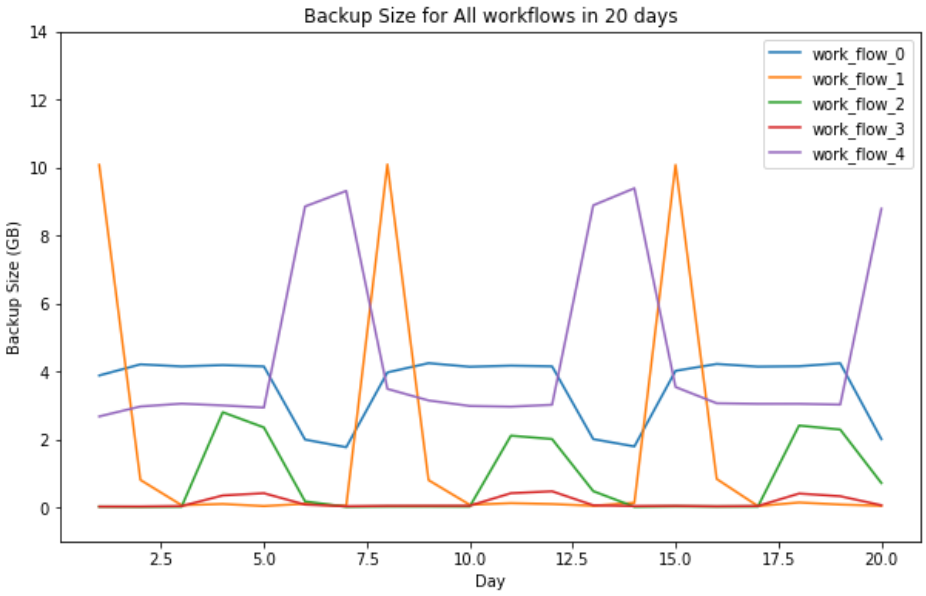


Figure 1. Backup size for all workflows in 20 days

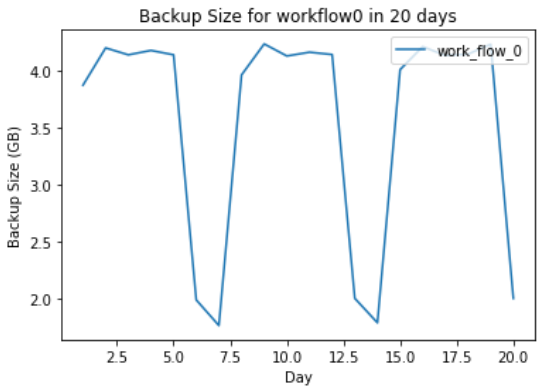


Figure 2. Backup size for workflow 0 in 20 days

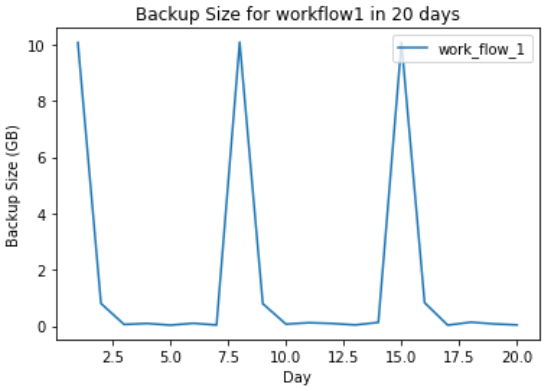


Figure 3. Backup size for workflow 1 in 20 days

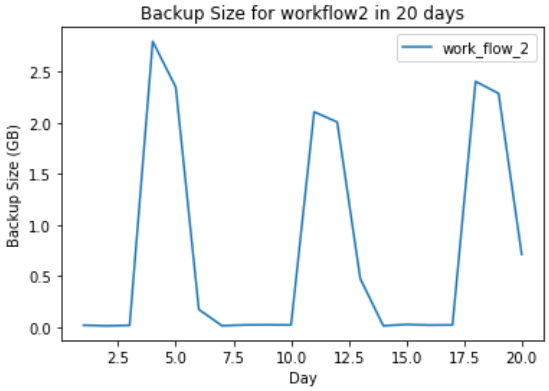


Figure 4. Backup size for workflow 2 in 20 days

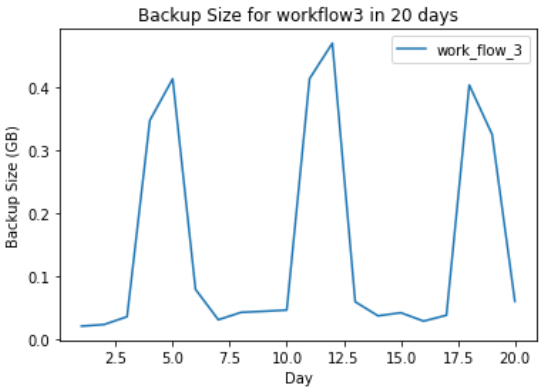


Figure 5. Backup size for workflow 3 in 20 days

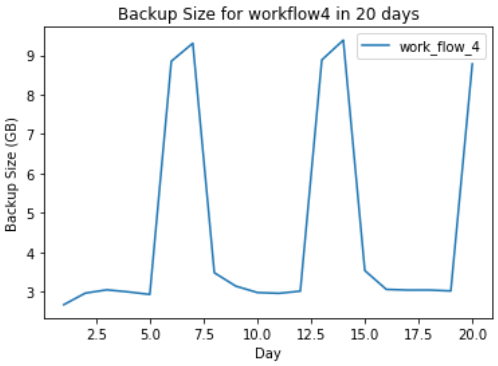


Figure 6. Backup size for workflow 4 in 20 days

Result:

From figures above, we can know that the backup size in each workflow repeats about every 7.5 days. We can regard them as a periodic variation and the maximum size for each workflow appears in different days.

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

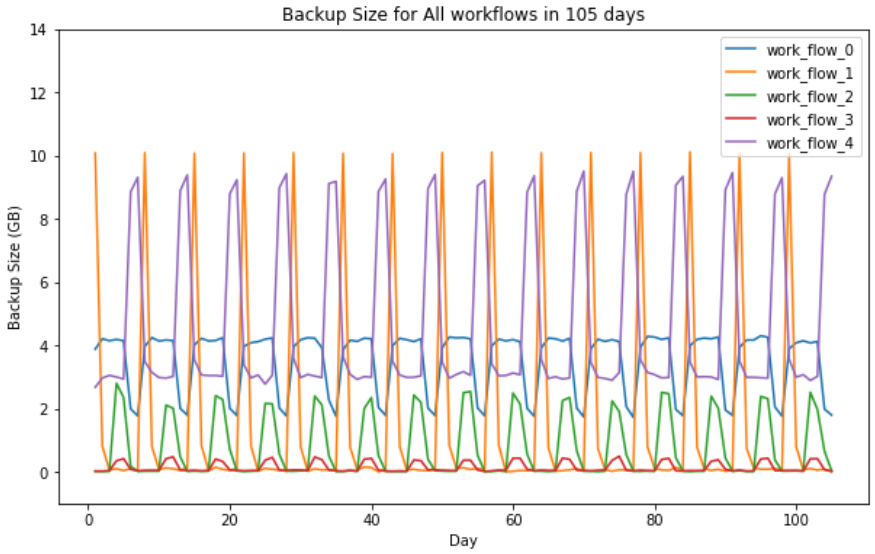


Figure 7. Backup size for all workflows in 105 days

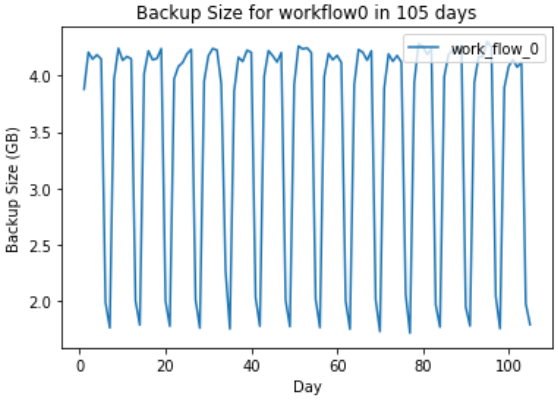


Figure 8. Backup size for workflow 0 in 105 days

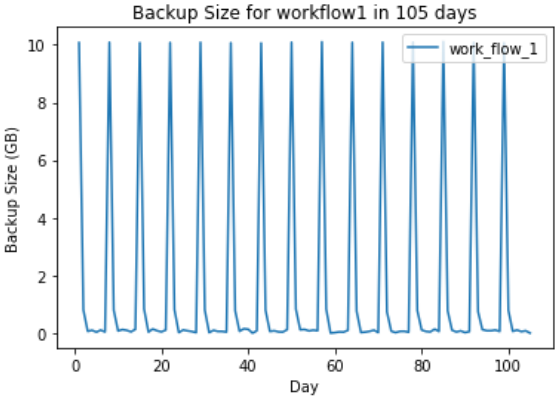


Figure 9. Backup size for workflow 1 in 105 days

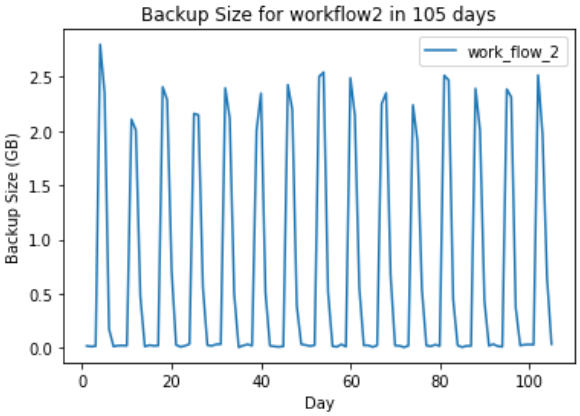


Figure 10. Backup size for workflow 2 in 105 days

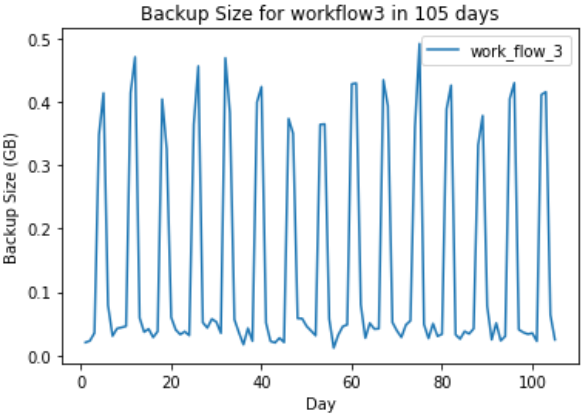


Figure 11. Backup size for workflow 3 in 105 days

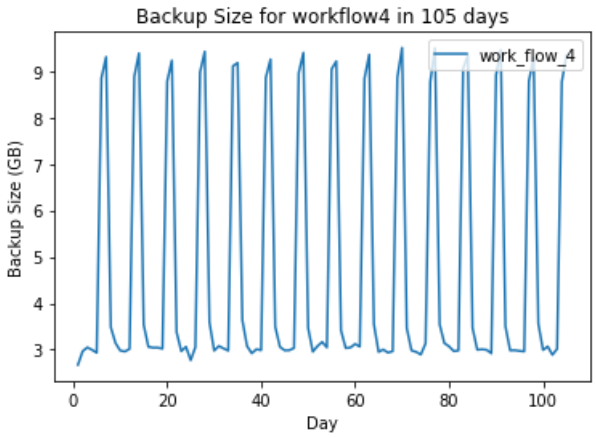


Figure 12. Backup size for workflow 4 in 105 days

Result:

From figures above, we again observe that the backup size in each workflow repeats about every 7.5 days, with periodic variation and maximum size appearing in different days. For each workflow, the numbers of each maximum backup size are not exactly the same but are closed to one another. The maximum sizes for workflow 0, 1, 2, 3 and 4 are about 4 GB, 10 GB, 2.3 GB, 0.4 GB and 9GB, respectively.

2. Predict

(a) In this section, we fit a linear regression model and use ordinary least square as the penalty function:

$$\min\|Y - X\beta\|^2 \text{ , where the minimization is on the coefficient vector } \beta.$$

i) We first apply scalar encoding to data and fit it to linear regression model. The method we use to calculate training and test RMSEs is provided on Piazza as:

> For a model, when doing 10-fold cross-validation, you need to calculate both train and test MSE for each fold, and let's call them $mse_{train,1}, \ldots, mse_{train,10}$, and $mse_{test,1}, \ldots, mse_{test,10}$.
>
> **What you are asked to report is**
>
> $$\text{RMSE}_{train} = \sqrt{\frac{mse_{train,1}+\cdots+mse_{train,10}}{10}} = \sqrt{\frac{sse_{train,1}+\cdots+sse_{train,10}}{10\times(\#\text{ of training data points in each fold})}}$$
>
> and
>
> $$\text{RMSE}_{test} = \sqrt{\frac{mse_{test,1}+\cdots+mse_{test,10}}{10}} = \sqrt{\frac{sse_{test,1}+\cdots+sse_{test,10}}{10\times(\#\text{ of test data points in each fold})}}$$

So with 10-fold cross validation, we obtain the following training and test RMSEs:

Average training RMSE: 0.10358539364277801
Average testing RMSE: 0.1036758476759903

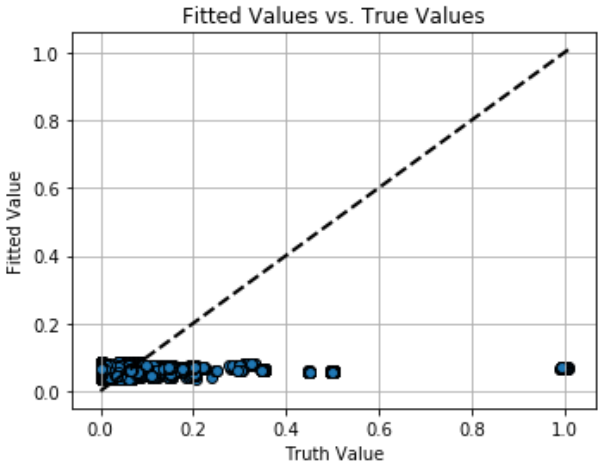


Figure 13. Fitted values vs. True values

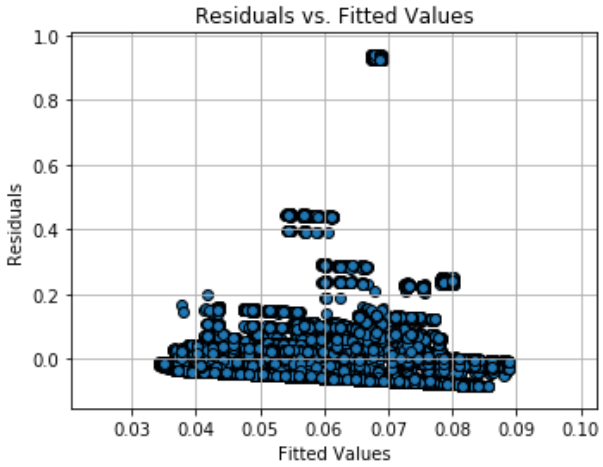


Figure 14. Residual values vs. fitted values

---

ii) We then fit the standardized data to linear regression model and obtain:

Average training RMSE: 0.10358539364277801
Average testing RMSE: 0.1036758476759903

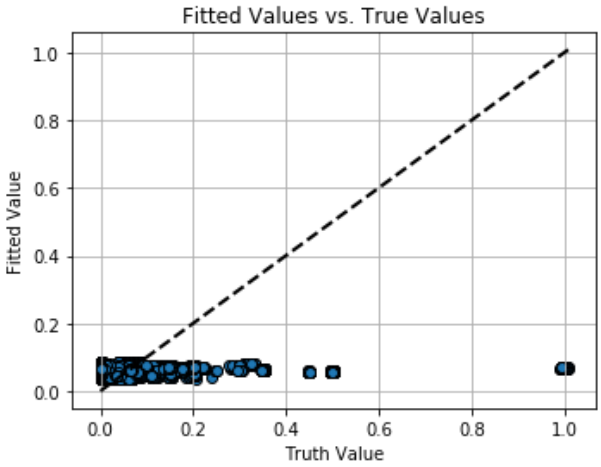


Figure 15. Fitted vs. true after standardize

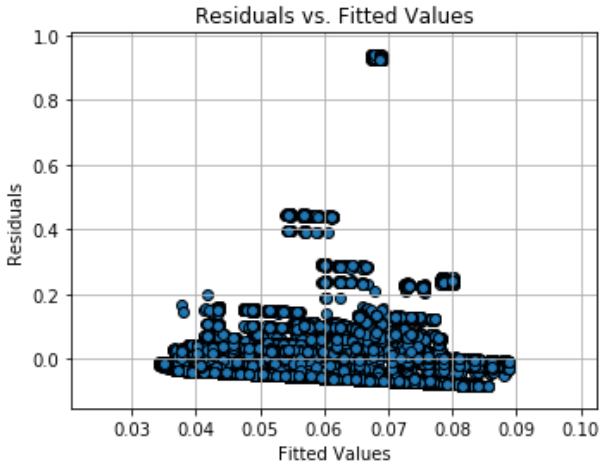


Figure 16. Residual vs. fitted after standardize

Result:

The results after standardization are exactly the same as that without standardization. We can then know that standardization will not change RMSE and fitting performances at all. Standardization only changes the scales of data.

(iii) Fitting select three most important features by f_regression and mutual information regression to linear regression model, we obtain the following table:

	F-test	MI
Feature 1 (Week #)	0.00006	0.00718
Feature 2 (Day of Week)	0.25750	0.30175
Feature 3 (Start Time)	1.00000	0.38828
Feature 4 (WorkFlowID)	0.17340	1.00000
Feature 5 (File Name)	0.16797	0.99286

Result:

The result above shows the score of f-regression and mutual information regression corresponding to features 1 to 5, respectively. Comparing F-test score, feature 3 (backup start time) has the highest score; while when considering mi score, feature 4 and feature 5 get the similar high scores, which corresponding to workflow ID and file name. Therefore, based on F-test score and mi score, we choose feature 3,4 and 5 as the three most important features, and the performance of the new linear model is shown.

Average training RMSE: 0.10369452819355413
Average test RMSE: 0.10377229307066825

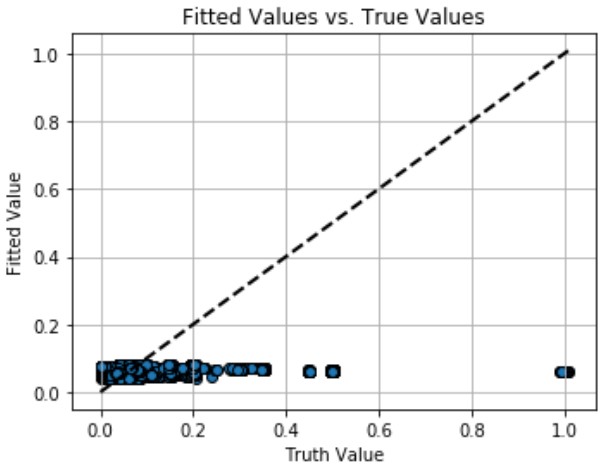


Figure 18. Fitted vs. true after feature selection

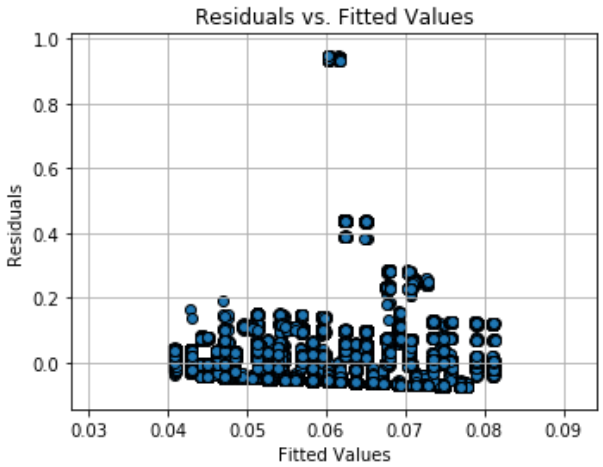


Figure 19. Residual vs. fitted after feature selection

Result:

The average training RMSE before feature selection is 0.1035853936427780, and that of the testing RMSE is 0.10362189439542643. However, the average training and testing RMSEs after feature selection are 0.10369452819355413 and 0.10377229307066825, respectively, which are a little larger than the previous values. Thus, we can see that the performance is not improved after feature selection. The possible explanation is that even though these three features are the top 3 most important features among the 5 features, the information they contains is not enough for fitting a regression better than which containing all 5 features. At least in this regression, the model with 5 features is better than that with only 3, if the average RMSE values are compared. However, in other conditions, having more features may not always result in better performance because too many features may cause overfitting, leading to the increase of RMSE score in testing dataset.

(iv) Feature Encoding

In this section, we apply the 32 possible combinations of encoding the five categorical variables, and the plot of average training and test RMSEs of all 32 feature encoding combinations are shown as:

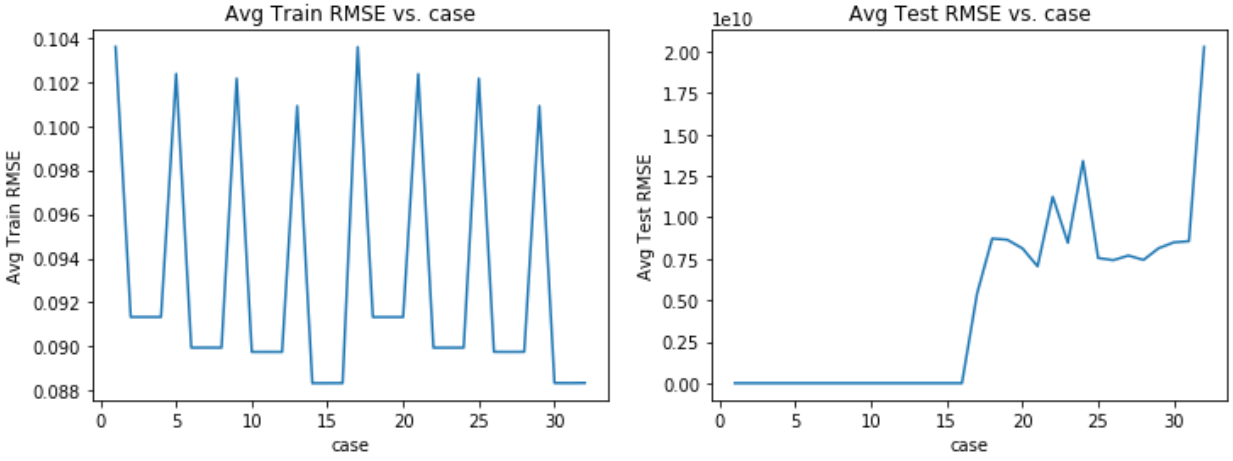


Figure 20. Avg training and testing RMSE of all 32 feature encoding combinations

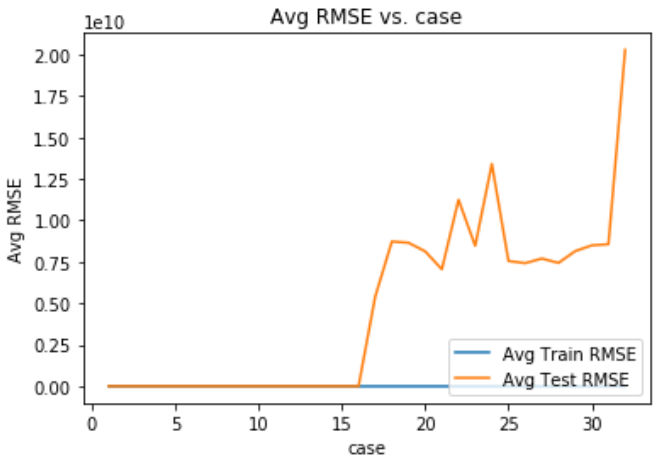


Figure 21. Avg training and testing RMSE of all 32 feature encoding combinations

```
Minimum Train RMSE is 0.08833732602898674 at index 13
Minimum Testing RMSE is 0.08850438637220652 at index 14
best case: index 14
Combination: 01110, scalar | onehot | onehot | onehot | scalar
```

Result:

As we can see, the average value of training RMSE is very small for about 0.1. However, those RMSEs of testing data are smaller in the former 16 combinations while increase rapidly in the latter 16 combinations. The minimum values of both training and testing data appear at index 13. The minimum average training RMSE is 0.08833732602898674 at index 13 and the minimum test RMSE is 0.08850438637220652 at index 14.

Index 14 is the combination: 01110, which is scalar | onehot | onehot | onehot | scalar, corresponding to the order of feature 1 to 5. In other words, the fitting model has the best performance when the numbers of "weeks" and "file names" are encoded in scalar representation, while those of "day of week", "backup start time" and "workflow ID" are encoded in one-hot representation. Since one-hot representation increases the number of columns in the dataset, the datasets with 32 combinations have different sizes.

The intuitive explanation for this (01110) best combination is that for feature 1 (# of week) and and feature 5 (file name), the numbers of them have no meaningful information related to the backup sizes; instead, they are just representations to identify weeks and files with each other. However, for feature 2 (day of week), feature 3 (backup start time) and feature 4

(workflow ID), if they were represented in scalar representation, it would be hard for us to compare and identify whether one component is larger than another. Now since they are in one-hot representation, we don't need to compare the values of them. Each 1 value in the feature corresponds to which subcomponent of the feature it is in for this condition. In addition, the lengths of items in the same features are the same. Concluding from the last problem, we already know that the three most important features are backup start time, workflow ID and file name, and it is obvious that backup start time is closely related with the backup size. However, the backup size has a periodic property, so the number of week is irrelevant here; and file name has a set of number distributed randomly across the whole dataset, not distributing to the result of backup size, either. Therefore, (01110) combination has the best performance.

The results of (01110) combination are shown below:

Average training RMSE: 0.08833837427395831
Average test RMSE: 0.08850438672956973

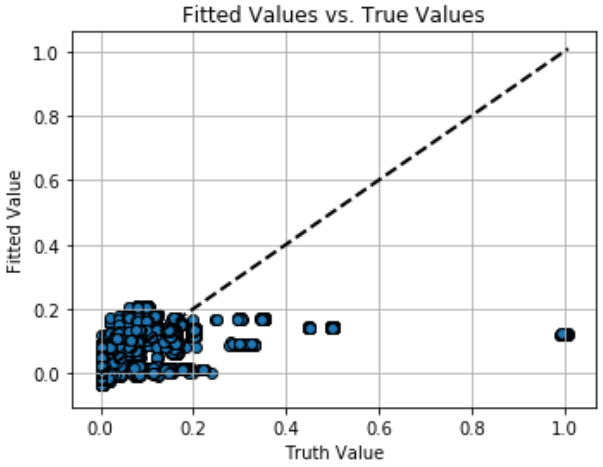


Figure 22. Fitted vs. true of 01110 combination

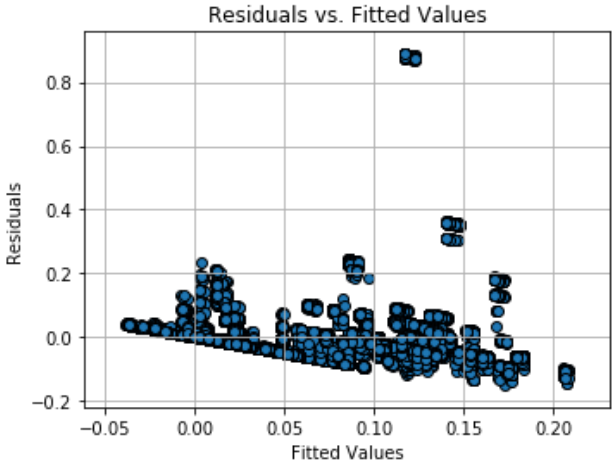


Figure 23. Residual vs. fitted of 01110 combination

(v) Controlling ill-conditioning and over-fitting

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

Explanation:

In some combination, we found larger increment in test RMSE compared to training RMSE. According to the plot, when the number of week is one-hot encoded, the training RMSE is extremely large, which is more than e^10, because the number of week is not related to backup size, thus causing overfitting in the model.

There are mainly two reasons for high RMSE. The first is ill-conditioning and the other is overfitting. Ill conditioning is that the information of a dataset contains is too simple and not adequate enough for a model to analyze and fit, leading to low precision of the predicted results and high RMSE. In contrast, overfitting refers to the condition in which the dataset is fitted too well. It arises because the model is learnt 'too closely' from the training set, which also leads to high RMSE because of the lack of general meaning in test dataset. It does fit the training dataset very well with the values of RMSE extremely low, but it cannot fit the test data well.

In some combinations, the training RMSE values are pretty low while testing RMSE values are high, and it is caused by overfitting. Therefore, we need to apply regularization to solve the overfitting problem. Regularization is a technique used in an attempt to solve the overfitting problem in statistical models.

In the following part, three different regularizers will be applied.

Ridge regularizer:

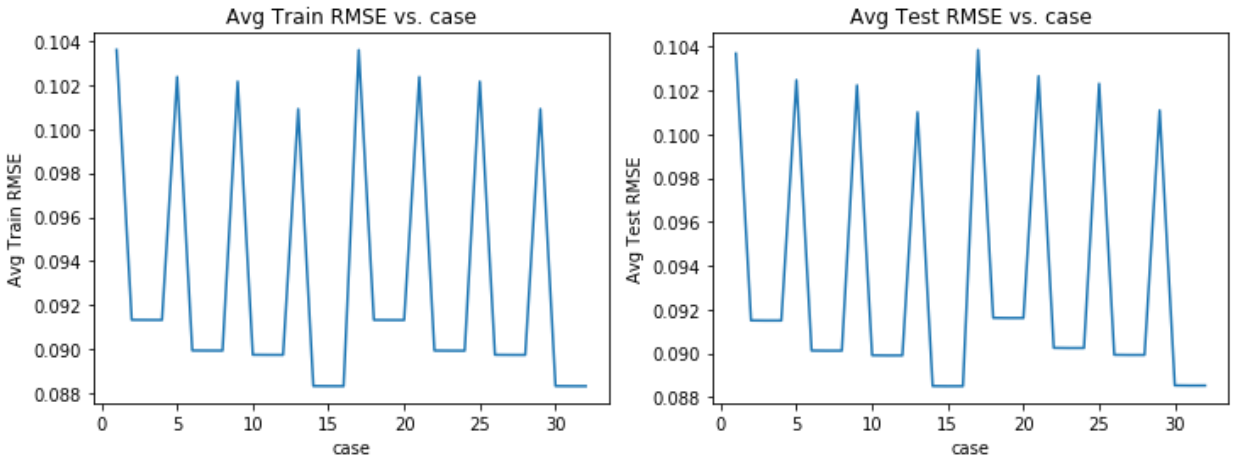


Figure 24. Avg training and testing RMSE of ridge regularizer

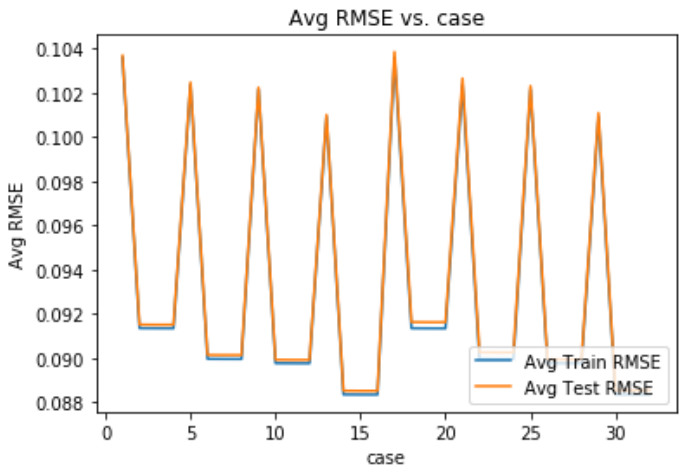


Figure 25. Avg training and testing RMSE of all 32 feature encoding combinations of ridge regularizer

```
Minimum Train RMSE is 0.08833509560395517 at index 31
Minimum Test RMSE is 0.088503990820505 at index 14
best case: index 14
Combination: 01110, scalar | onehot | onehot | onehot | scalar
-----------------------------------------------
Average training RMSE: 0.0883377153893992
Average test RMSE: 0.088503990820505
```

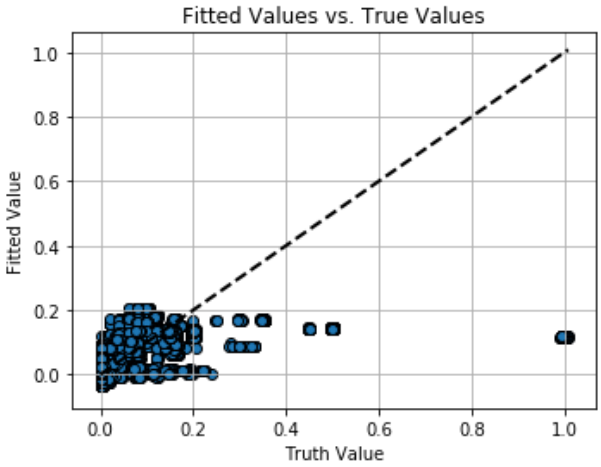


Figure 26. Fitted vs. true of ridge regularizer

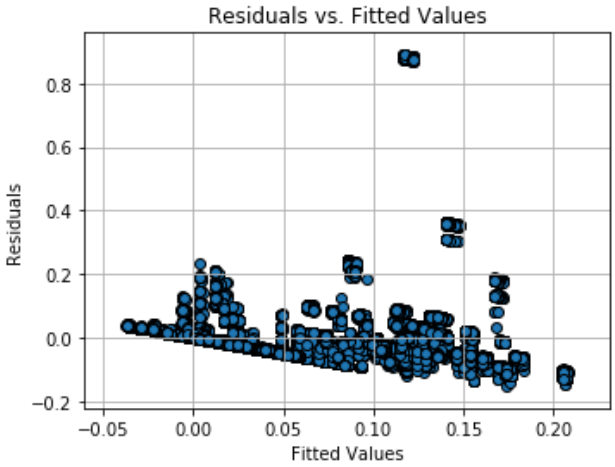


Figure 27. Residual vs. fitted of ridge regularizer

Ridge regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multicollinearity (correlations between predictor variables).

In ridge regularizer, we've found that the performance is best when alpha = 12.

Lasso Regularizer:

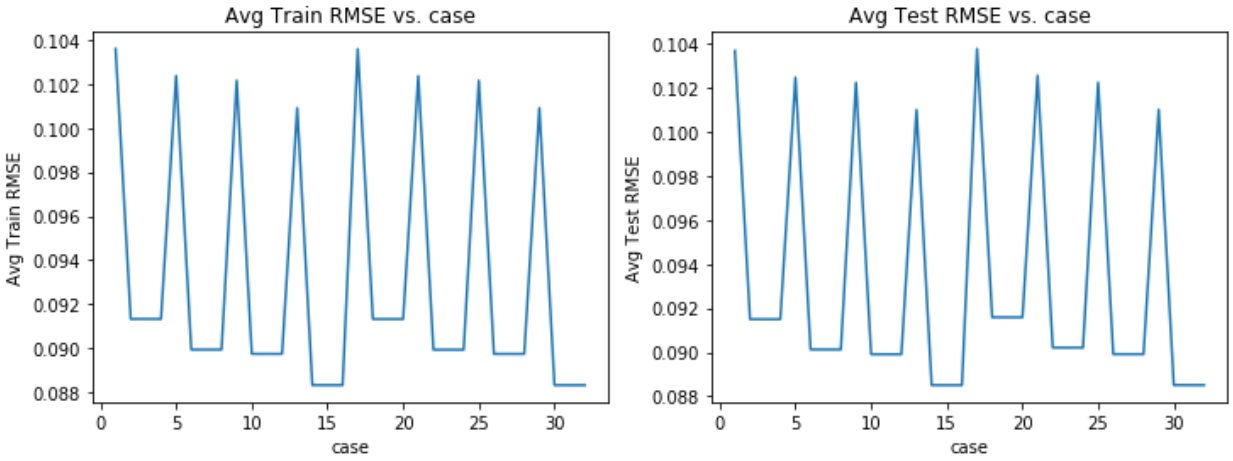


Figure 28. Avg training and testing RMSE of lasso regularizer

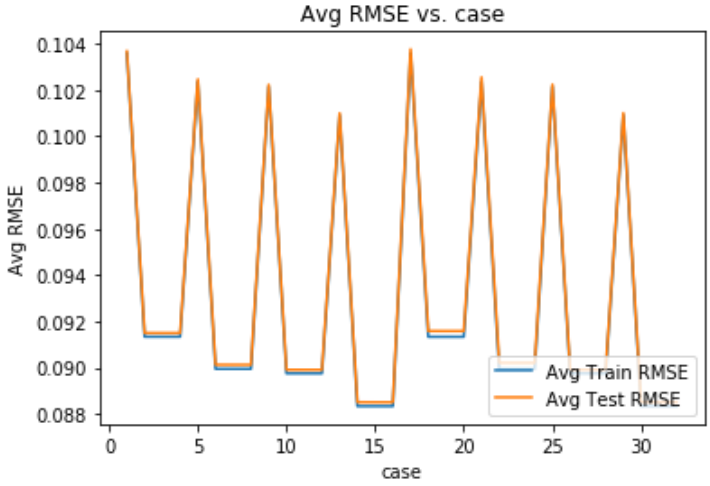


Figure 29. Avg training and testing RMSE of lasso regularizer

```
Minimum Train RMSE is 0.08833570788509812 at index 31
Minimum Test RMSE is 0.08850437612625646 at index 14
best case: index 14
Combination: 01110, scalar | onehot | onehot | onehot | scalar
-----------------------------------------------
Average training RMSE: 0.08833759236962922
Average test RMSE: 0.08850437612625646
```

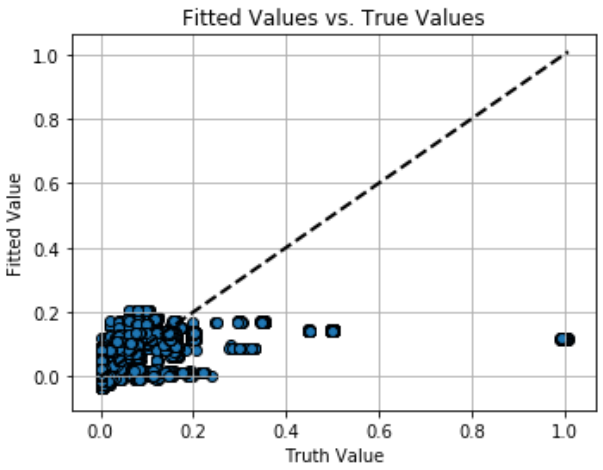


Figure 30. Fitted vs. true of lasso regularizer

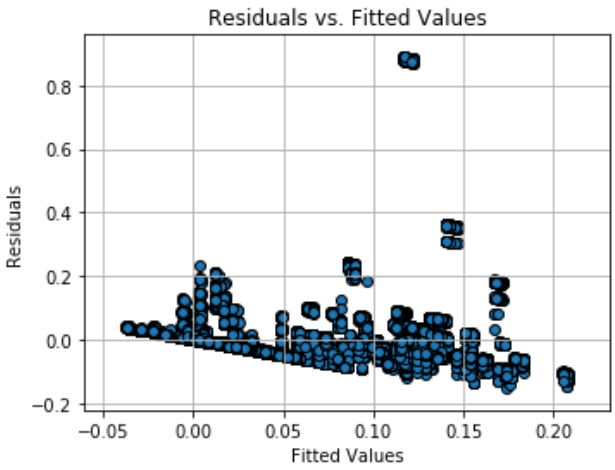


Figure 31. Residual vs. fitted of lasso regularizer

Lasso regression is a type of linear regression that uses shrinkage, which implies that data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when one wants to automate certain parts of model selection, like variable selection/parameter elimination.

In lasso regularizer, we found that the performance is best when alpha = 0.000015.

Elastic Net Regularizer:

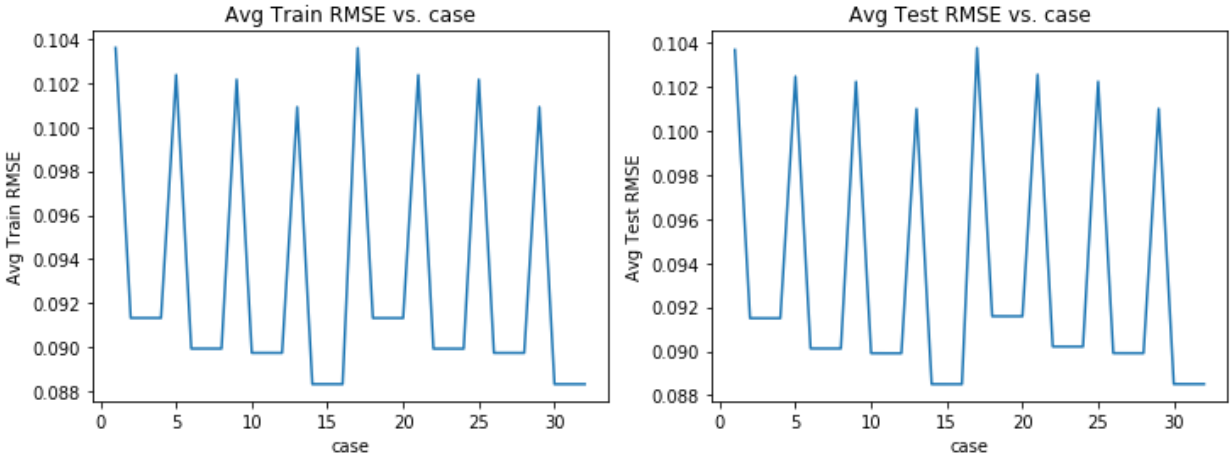


Figure 32. Avg training and testing RMSE of Elastic net regularizer

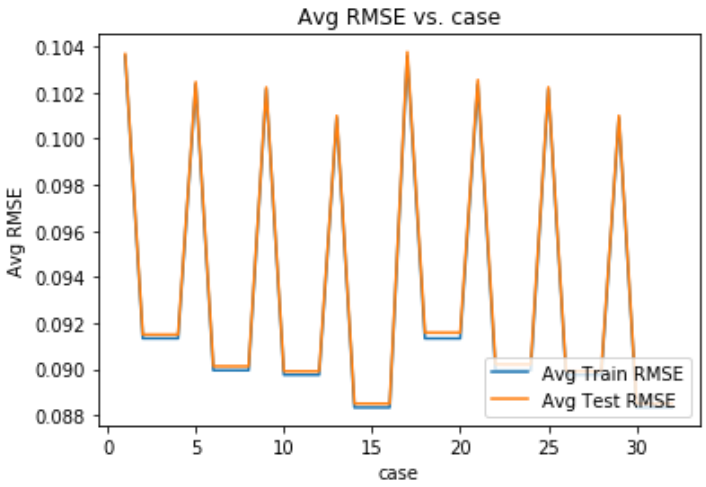


Figure 33. Avg training and testing RMSE of Elastic net regularizer

```
Minimum Train RMSE is 0.08833555355045793 at index 31
Minimum Test RMSE is 0.08850435959342434 at index 14
best case: index 14
Combination: 01110, scalar | onehot | onehot | onehot | scalar
-----------------------------------------------
Average training RMSE: 0.08833755660346942
Average test RMSE: 0.08850435959342434
```

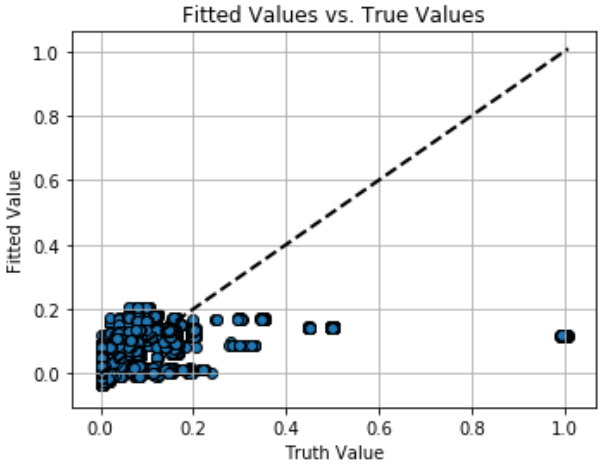


Figure 34. Fitted vs. true of Elastic net regularizer

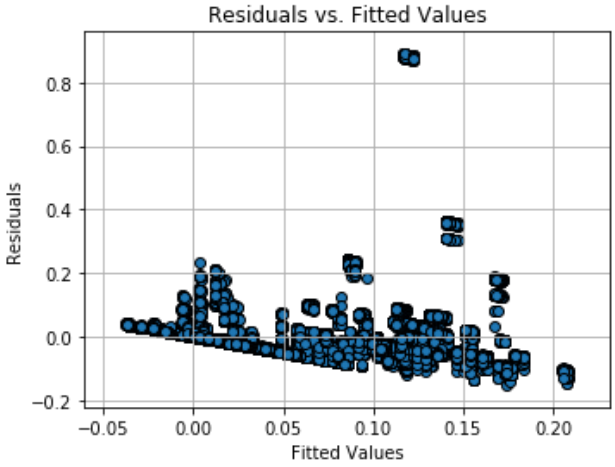


Figure 35. Residual vs. fitted of Elastic net regularizer

Elastic net is the same as lasso when $\alpha = 1$. As α shrinks toward 0, elastic net approaches ridge regression. For other values of α, the penalty term $P_\alpha(\beta)$ interpolates between the L_1 norm of β and the squared L_2 norm of β.

In Elastic Net regularizer, we found that the performance is best when alpha =0.00001 and l1_ratio = 0.1. According to the relationship `alpha = a + b and l1_ratio = a / (a + b)`, we can know that a = 0.000001 and b =0.000009.

Result:

The performance after applying each regularizer is better than before. We found that the smaller the value of alpha is, the better the performance of regularization will be. In addition, the test RMSE largely decreases and is similar to the train RMSE after regularization. Thus we can conclude that regularization will solve the problem of overfitting and improve the prediction performance.

(b) Use a random forest regression model for this same task.

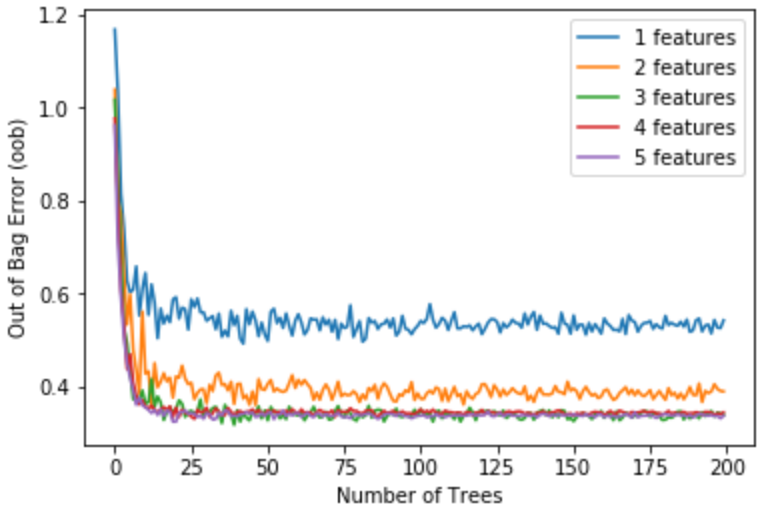
i. In this section, we use a random forest regression model to determine the feature importance with the following parameters:

A. Number of trees: 20
B. Depth of each tree: 4
C. Bootstrap: True
D. Maximum number of features: 5

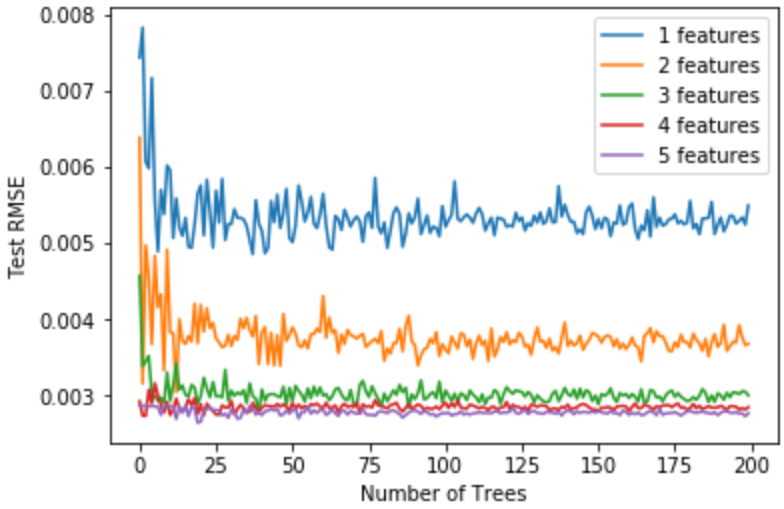
```
The average Train RMSE is 0.06018651686460464
The average Test RMSE is 0.06036674812138954
The Out Of Bag error is 0.34752379258876986
```

---

ii. After sweeping number of trees from 1 to 200 and maximum number of features from 1 to 4, we obtain the following plot of out of bag error (y axis) against number of trees(x axis):



And the plot of average Test-RMSE (y axis) against number of trees(x axis) is:



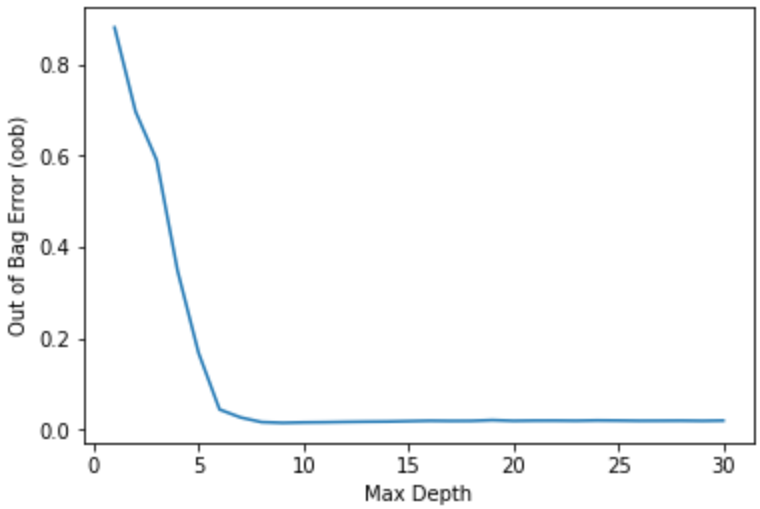
Result:

We can see from the plot that both the out of bag error and average test RMSE get stable after the number of tree becomes large than 4. Thus, the optimal number of tree should be set to four in terms of prediction efficiency.

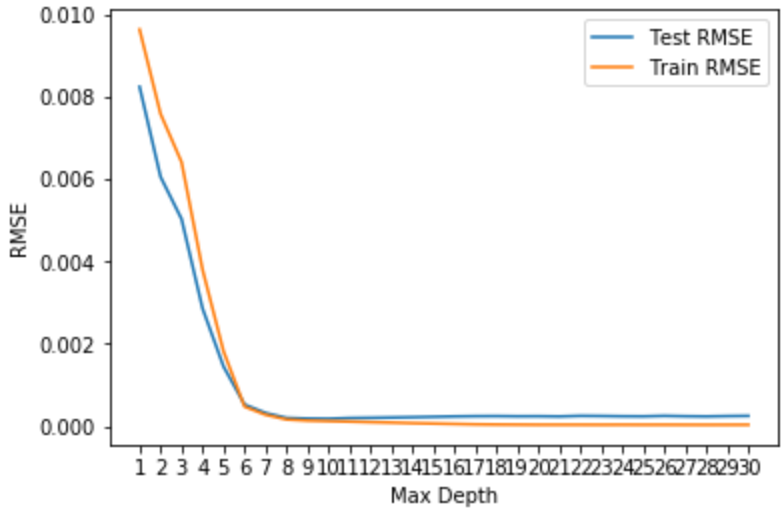
---

iii. In the part, we pick max_depth as another parameter to exam. Our hypothesis is that increasing this value to a certain point will cause the random forest to overfit the data or enter a performance floor region. This will be shown in the plots and is denoted by progressively worse and worse test errors.

Plot of out of bag error (y axis) against number of trees(x axis):



Plot of average train and test RMSE (y axis) against number of trees(x axis):



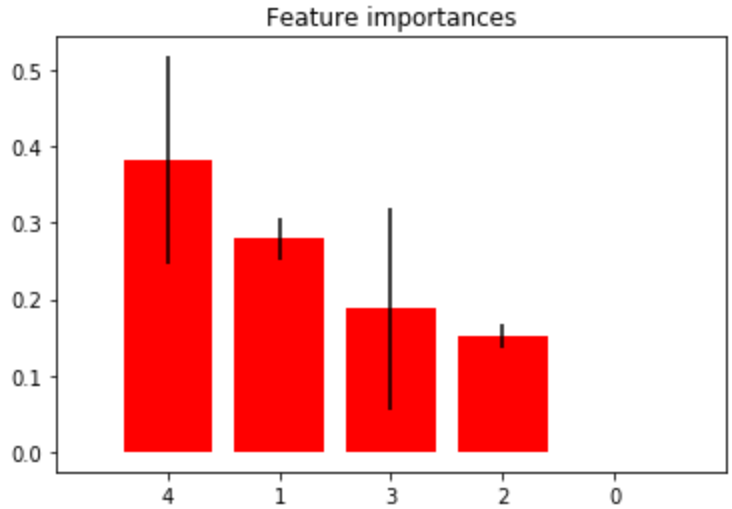
Result:

As shown in the plot, both the out of bag error and average test RMSE get stable after number of tree large than 8. Thus, the max_depth should not exceed 8 for best performance and efficiency.

---

iv. We report the feature importances from the best random forest regression as follows:
Feature ranking:

```
1. feature 4 (0.381779)
2. feature 1 (0.278946)
3. feature 3 (0.187456)
4. feature 2 (0.151817)
5. feature 0 (0.000002)
```

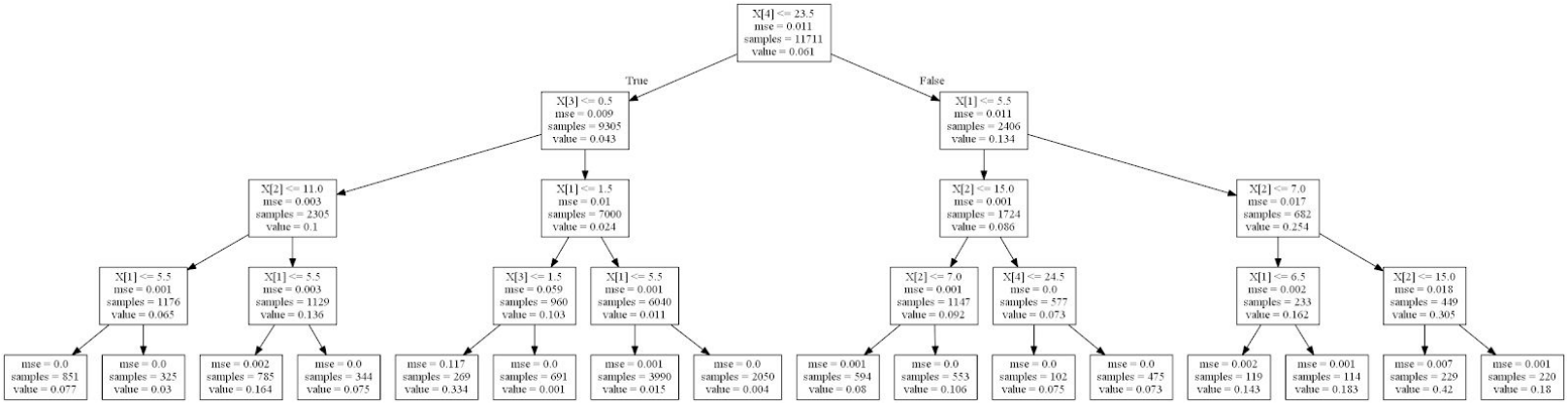


Result:

The plot above shows which features are the most informative and which are not. Clearly, feature 4 is the most informative, followed by features 1, 3, and 2. Feature 0 contains the least information. Thus, an optimal model in terms of efficiency and performance would use features 1 to 4.

---

v. With max depth as 4, we can visualize decision trees as:



Result:

The root is exactly feature 4, and it is the same as the most important feature according to the feature importance reported by the regressor. Thus, the picture verify the result in part iv.

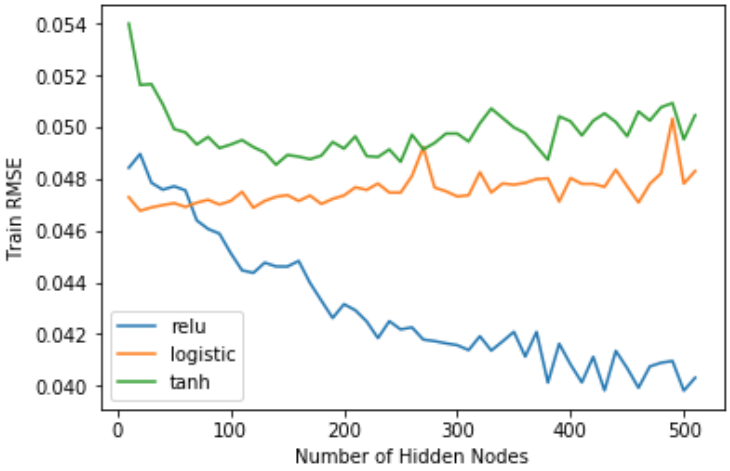
(c) Neural Network Regression Model

In this section, we use a neural network regression model with one hidden layer to analyze the data set. First we convert all features to be one-hot encoded, then we need to determine the best combination of parameters resulting in the optimal performance. The two parameters considered are:

• Activity Functions: (relu, logistic, tanh)

• Number of hidden units: from 10 to 520 with every increment of 10 units.

After sweeping over the range of hidden units for each activity functions, we've obtained the following plot of Test-RMSE as a function of the number of hidden units for different activity functions. Detailed data of each parameter combination is included in code.



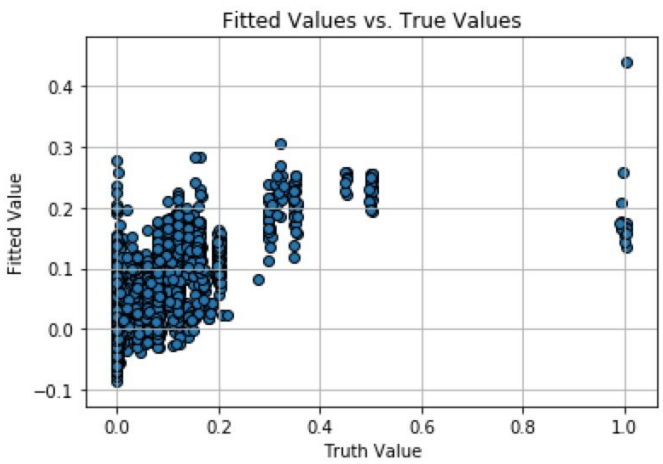
Result:

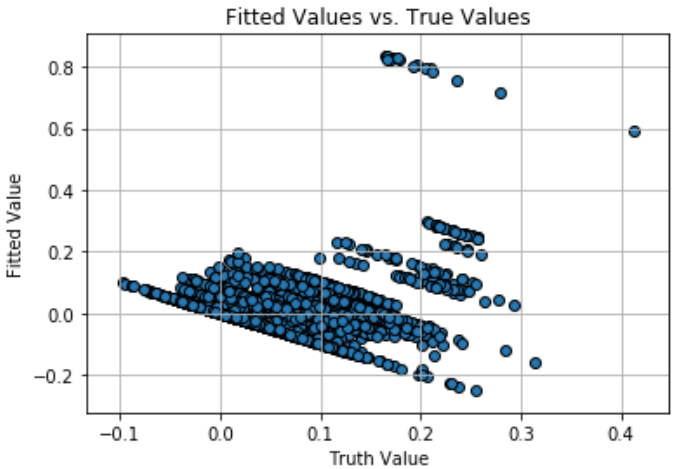
As we can from the graph, activity function of relu gives the best overall performance, which is also commonly used in practice. And the best combination of the two parameters is activity function as relu with number of hidden layer units as 470. For neural network regression model, increasing the number of hidden layers/neurons or using more complicated functions increases the network capacity. However, it is easier to overfit the training data when the model with high capacity starts to fit noise; in our case it would occur after 470 hidden units.

With these two parameters applied, the training and test RMSEs from 10-fold cross validation and scatter plots of residuals vs. fitted values and of fitted values vs. true values are shown:

```
---------------------------------------------
For activity function relu with hidden layer units 470


Training RMSE: 0.03312617243040347
Testing RMSE: 0.04009008137482515
---------------------------------------------
```



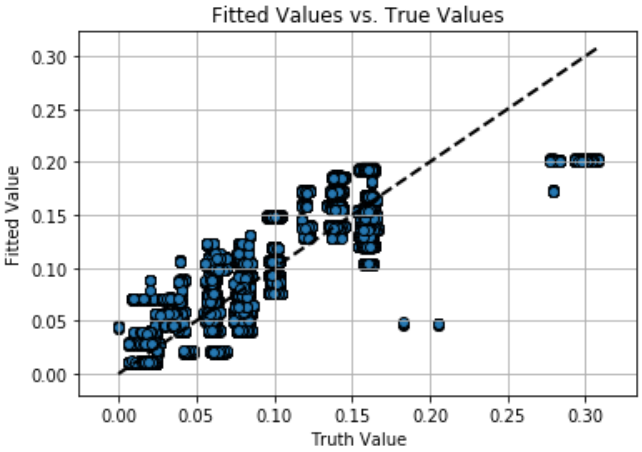


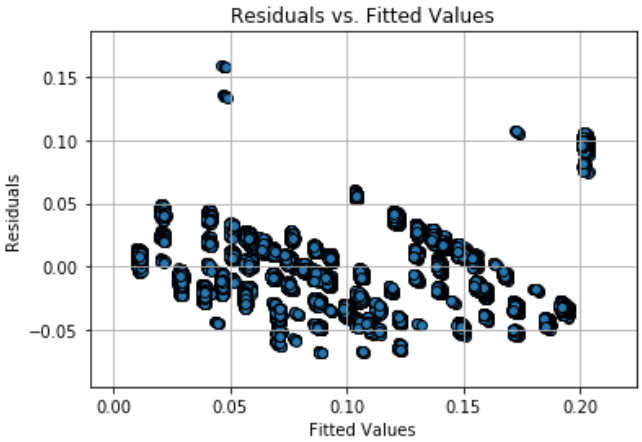
(d) Prediction of backup size for each workflow separately.

i) In this section, for each workflow, we use a linear regression model to predict the backup size and apply 10-fold cross validation to obtain the following results:

For workflow = 0:

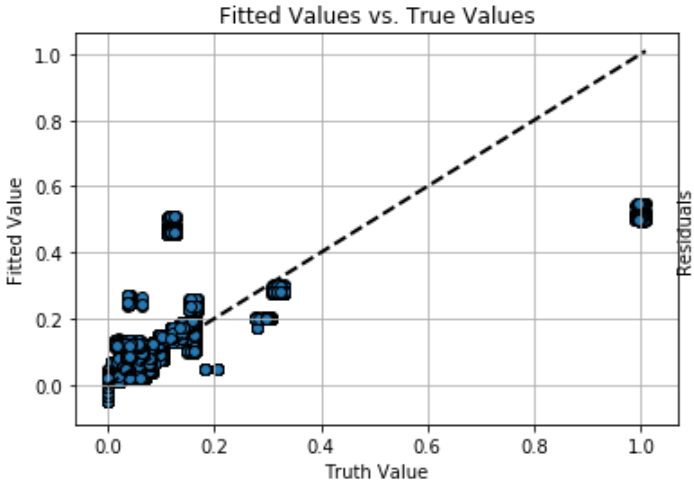
Training RMSE: 0.0292766274507745
Test RMSE: 0.029938276791619266

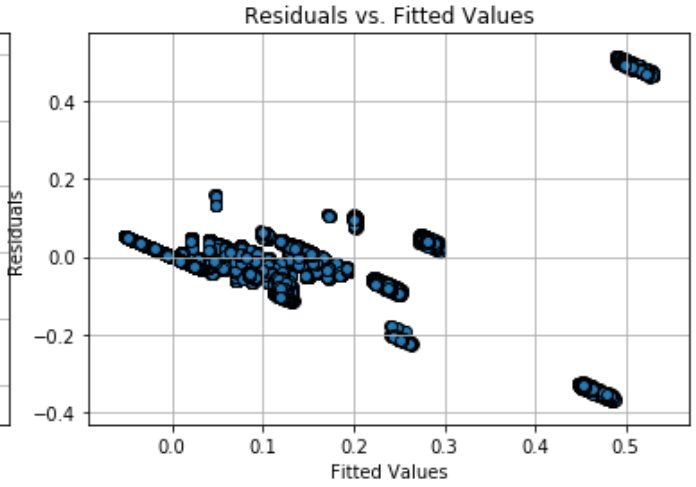




For workflow = 1:

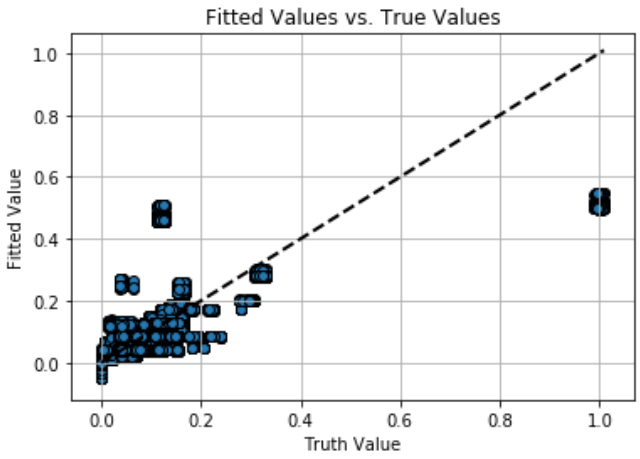
Training RMSE: 0.10326157281833645
Test RMSE: 0.10531924059087508

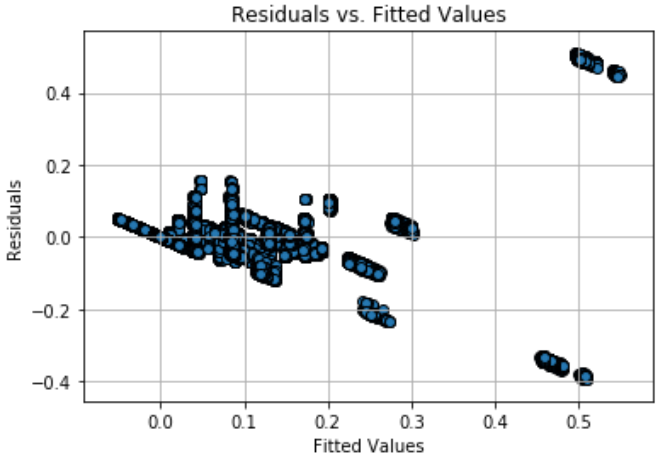




For workflow = 2:

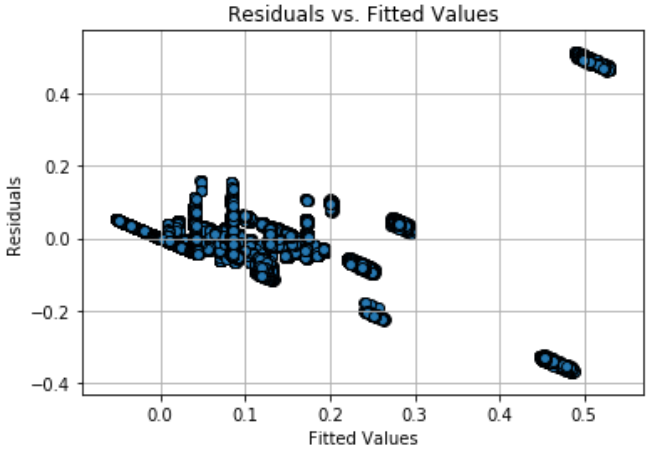
Training RMSE: 0.025503876580813403
Test RMSE: 0.025486196860638622





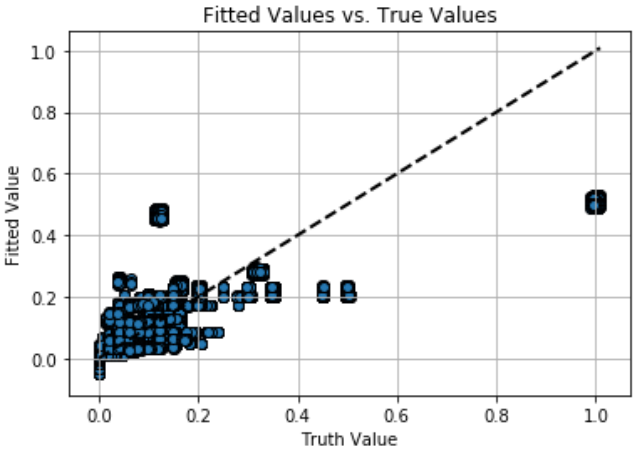
For workflow = 3:

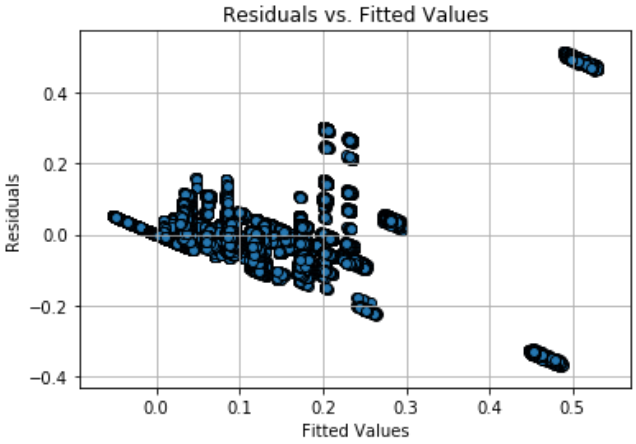
Training RMSE: 0.005901770232587948
Test RMSE: 0.005897143648670121



For workflow = 4:

Training RMSE: 0.08461446489459504
Test RMSE: 0.08235437823949608





Result:

As we can see from the data above, the values of training and testing RMSEs vary considerably for different workflows, with workflow_3 has the smallest testing RMSE as 0.00589 and workflow_0 has the largest as 0.105. However, compared with the simple linear regression model with test RMSE as 0.1037, the fit with piecewise linear regression model does generally improv. Since different workflows have different patterns of backup size changing, such a model with five different workflows will obtain five times more coefficients than the simple linear model, resulting in fit improvement.

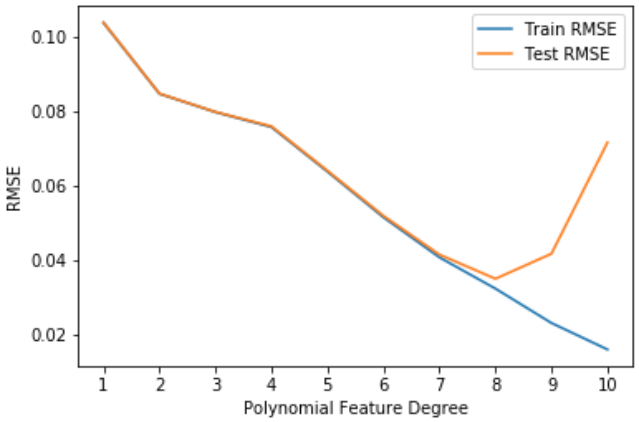
---

ii) In this section, we fit a polynomial regression model to each workflow to improve the fit and increase the degree of the polynomial from 1 to 10 to determine the degree threshold beyond which the generalization error starts to increase.

The training and test RMSEs from 10-fold cross validation of different degrees for each workflow are shown as below, and the plots of the average train and test RMSEs against the degree of the polynomial are shown accompanied with the data. The table of threshold degree, testing RMSE and training RMSE for each workflow is also included.

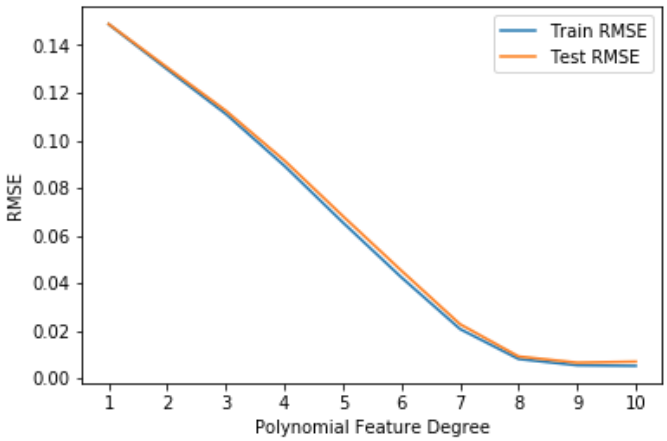
For workflow 0:

```
Work Flow 0
For degree  = 1:
10-fold Training RMSE =  0.10358539364277801
10-fold Average Test RMSE =  0.10367584767599029
For degree  = 2:
10-fold Training RMSE =  0.08452305583333208
10-fold Average Test RMSE =  0.08457614933103919
For degree  = 3:
10-fold Training RMSE =  0.0796057022993115
10-fold Average Test RMSE =  0.07966702189379106
For degree  = 4:
10-fold Training RMSE =  0.07558054210111323
10-fold Average Test RMSE =  0.07579198988964912
For degree  = 5:
10-fold Training RMSE =  0.0636238137091734
10-fold Average Test RMSE =  0.06387294820867767
For degree  = 6:
10-fold Training RMSE =  0.05135471537966795
10-fold Average Test RMSE =  0.051754385237645584
For degree  = 7:
10-fold Training RMSE =  0.040599764223435766
10-fold Average Test RMSE =  0.041344923649762715
For degree  = 8:
10-fold Training RMSE =  0.03222076513235551
10-fold Average Test RMSE =  0.034866125086937004
For degree  = 9:
10-fold Training RMSE =  0.022989339085887916
10-fold Average Test RMSE =  0.041588630651792025
For degree  = 10:
10-fold Training RMSE =  0.0158839531905154
10-fold Average Test RMSE =  0.07144635177518636
```



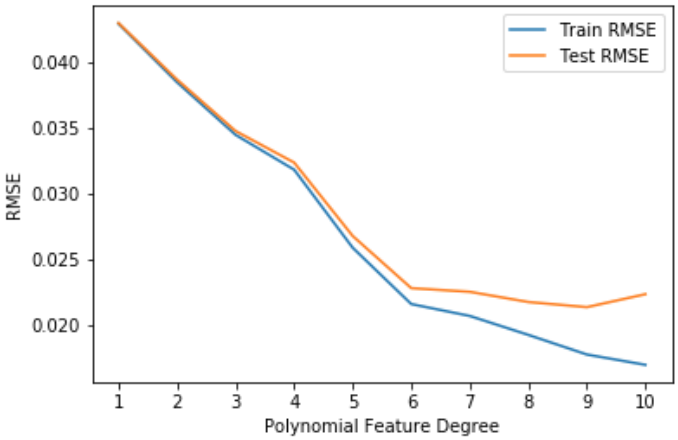
For workflow 1:

```
Work Flow 1
For degree  = 1:
10-fold Training RMSE =  0.14876681840843545
10-fold Average Test RMSE =  0.14889921561405156
For degree  = 2:
10-fold Training RMSE =  0.12981443575391663
10-fold Average Test RMSE =  0.13063838195079863
For degree  = 3:
10-fold Training RMSE =  0.11107961675604595
10-fold Average Test RMSE =  0.11247304155835404
For degree  = 4:
10-fold Training RMSE =  0.08929009514055145
10-fold Average Test RMSE =  0.09153791938753897
For degree  = 5:
10-fold Training RMSE =  0.065445900777658
10-fold Average Test RMSE =  0.06812665485430466
For degree  = 6:
10-fold Training RMSE =  0.042360941742868516
10-fold Average Test RMSE =  0.045171724762173224
For degree  = 7:
10-fold Training RMSE =  0.020673936981560448
10-fold Average Test RMSE =  0.02270766171061567
For degree  = 8:
10-fold Training RMSE =  0.008054658401203743
10-fold Average Test RMSE =  0.009154438024922662
For degree  = 9:
10-fold Training RMSE =  0.005429511461488692
10-fold Average Test RMSE =  0.006556321536727263
For degree  = 10:
10-fold Training RMSE =  0.005206430619277462
10-fold Average Test RMSE =  0.006980778085985342
```



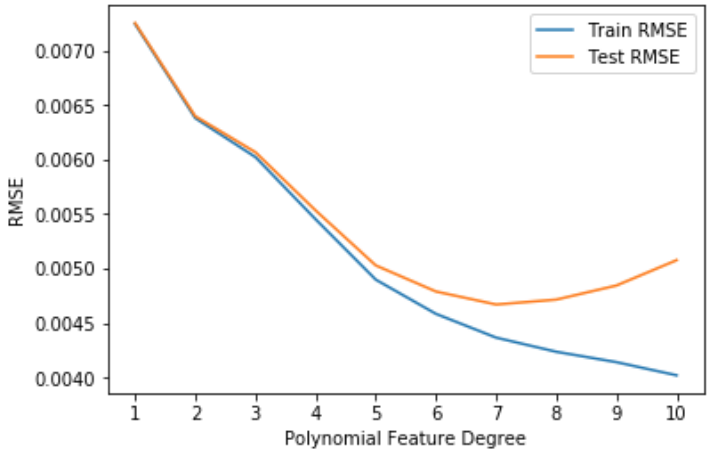
For workflow 2:

```
Work Flow 2
For degree  = 1:
10-fold Training RMSE =  0.04291387646902605
10-fold Average Test RMSE =  0.04297016681002815
For degree  = 2:
10-fold Training RMSE =  0.03846662825889786
10-fold Average Test RMSE =  0.03867051083457821
For degree  = 3:
10-fold Training RMSE =  0.03446833168482608
10-fold Average Test RMSE =  0.03476001053133154
For degree  = 4:
10-fold Training RMSE =  0.03182213515921353
10-fold Average Test RMSE =  0.032362980867411184
For degree  = 5:
10-fold Training RMSE =  0.02589094302975755
10-fold Average Test RMSE =  0.026784485064965433
For degree  = 6:
10-fold Training RMSE =  0.02161173158380393
10-fold Average Test RMSE =  0.022820936743652806
For degree  = 7:
10-fold Training RMSE =  0.0207078991062505
10-fold Average Test RMSE =  0.022543373974041107
For degree  = 8:
10-fold Training RMSE =  0.01926856638256666
10-fold Average Test RMSE =  0.021767109879913826
For degree  = 9:
10-fold Training RMSE =  0.017776227696104065
10-fold Average Test RMSE =  0.021377398914473798
For degree  = 10:
10-fold Training RMSE =  0.016992910590794354
10-fold Average Test RMSE =  0.022362750723112685
```



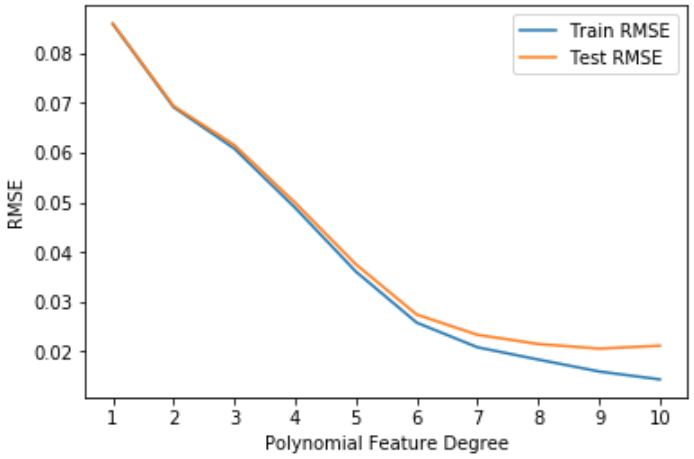
For workflow 3:

```
Work Flow 3
For degree  = 1:
10-fold Training RMSE =  0.007244249830388539
10-fold Average Test RMSE =  0.007252909461700565
For degree  = 2:
10-fold Training RMSE =  0.006380925874088572
10-fold Average Test RMSE =  0.006398808647268262
For degree  = 3:
10-fold Training RMSE =  0.006023617266102906
10-fold Average Test RMSE =  0.006068174754080074
For degree  = 4:
10-fold Training RMSE =  0.005452825537047303
10-fold Average Test RMSE =  0.005533901594925469
For degree  = 5:
10-fold Training RMSE =  0.004899040576456771
10-fold Average Test RMSE =  0.005029623714702724
For degree  = 6:
10-fold Training RMSE =  0.004585398350609503
10-fold Average Test RMSE =  0.004790364188306786
For degree  = 7:
10-fold Training RMSE =  0.004367346215748943
10-fold Average Test RMSE =  0.004671194834985293
For degree  = 8:
10-fold Training RMSE =  0.0042376179958458484
10-fold Average Test RMSE =  0.004716213168603012
For degree  = 9:
10-fold Training RMSE =  0.0041422779964885095
10-fold Average Test RMSE =  0.004845584267452483
For degree  = 10:
10-fold Training RMSE =  0.0040216664823179
10-fold Average Test RMSE =  0.0050782234354472084
```



For workflow 4:

```
Work Flow 4
For degree  = 1:
10-fold Training RMSE =  0.08591961471176292
10-fold Average Test RMSE =  0.08603409489651456
For degree  = 2:
10-fold Training RMSE =  0.06918799177684047
10-fold Average Test RMSE =  0.06942075768128025
For degree  = 3:
10-fold Training RMSE =  0.060835918943146335
10-fold Average Test RMSE =  0.06155186683478309
For degree  = 4:
10-fold Training RMSE =  0.048912172213522516
10-fold Average Test RMSE =  0.04992985200783331
For degree  = 5:
10-fold Training RMSE =  0.036070132968210614
10-fold Average Test RMSE =  0.037542159412282644
For degree  = 6:
10-fold Training RMSE =  0.02579720006894594
10-fold Average Test RMSE =  0.027433285181930417
For degree  = 7:
10-fold Training RMSE =  0.020839404313551406
10-fold Average Test RMSE =  0.023348621118164638
For degree  = 8:
10-fold Training RMSE =  0.018367561485411032
10-fold Average Test RMSE =  0.021503626283425815
For degree  = 9:
10-fold Training RMSE =  0.015970434823599364
10-fold Average Test RMSE =  0.020572917643981628
For degree  = 10:
10-fold Training RMSE =  0.014374297080255252
10-fold Average Test RMSE =  0.021152940969050286
```



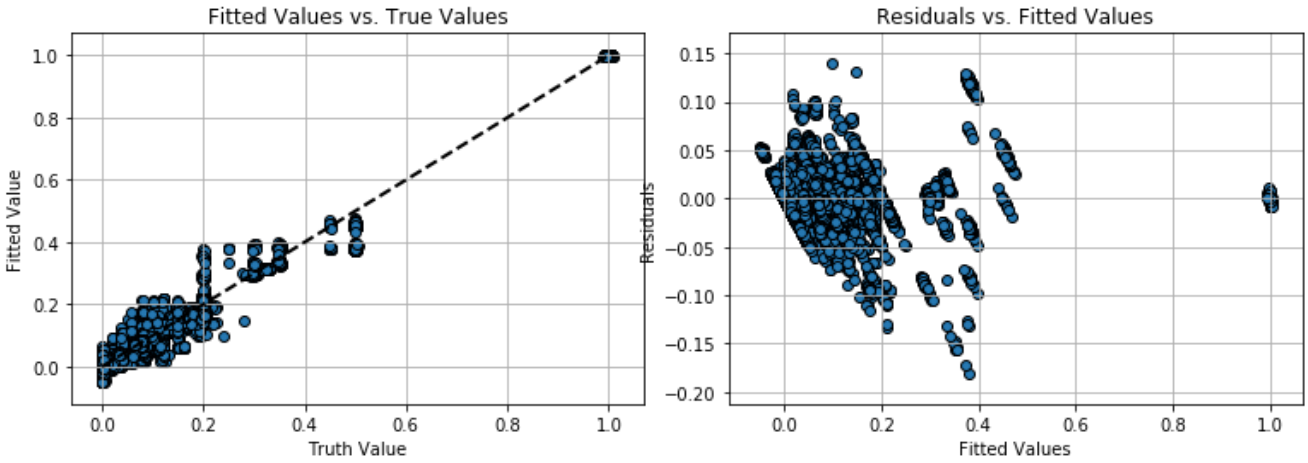
Workflow ID	0	1	2	3	4
Threshold Degree	8	9	9	7	9
Training RMSE	0.03222	0.00543	0.02254	0.00437	0.02057
Test RMSE	0.03487	0.00656	0.02138	0.00467	0.02150

Result:

Despite of concluding the general threshold degree to be 7~9 based from the table, we can see from the plots that roughly speaking the generalization errors start to increase after passing the threshold degree of 6 for each workflow. We again confirm that different workflows have different backup size changing patterns hence different threshold degrees of polynomial fitting.

Also, the results suggest that different dataset can generate different best-fit polynomial; therefore, cross validation for any particular dataset allows the model traverse dall data in model training, which helps to reduce model complexity and overfitting possibility and make the model more general for prediction.

An example of scatter plots of workflow 0 with degree 5 is shown:



(e) K-Nearest Neighbor Regression Model

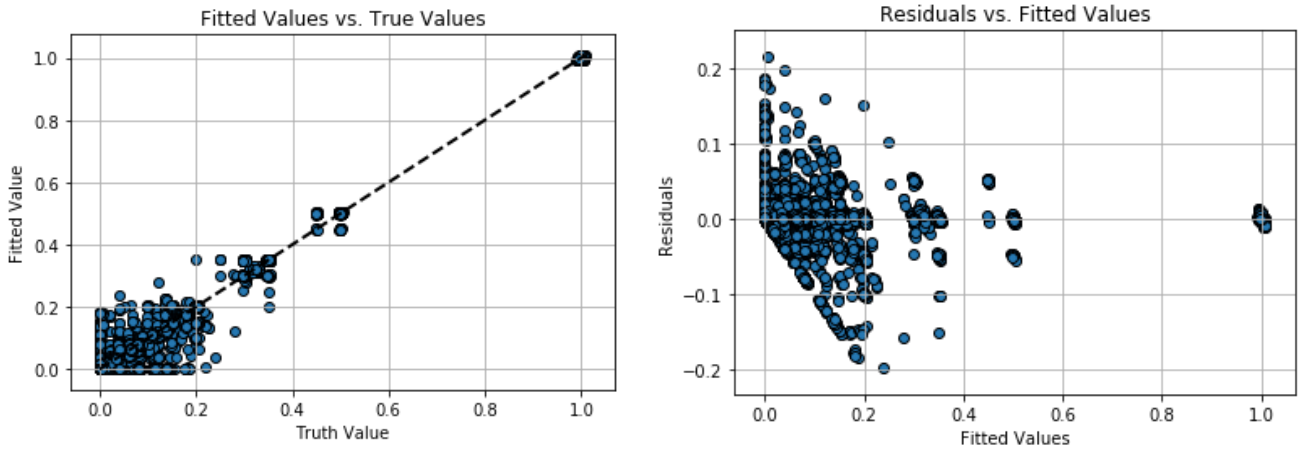
We use two parameters for the k-nearest neighbor regression model in this section: (1) n_neighbors: [1,2,5,8] and (2) weights: [uniform, distance] to determine the best combination of parameters that give the performance of KNR model. The results are obtained as follows:

For weights as uniform:

- n_neighbors = 1:

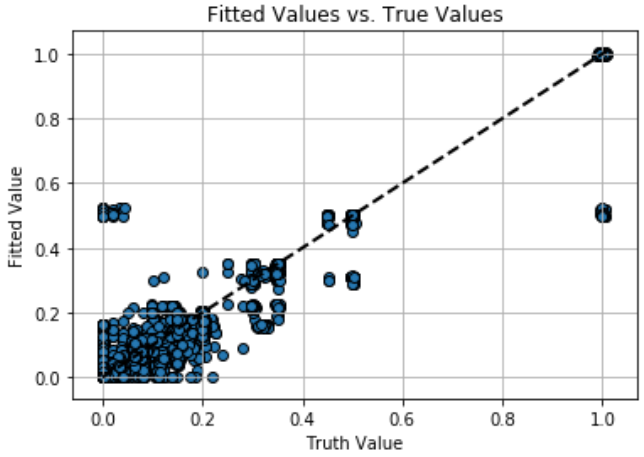
10-Fold Training RMSE: 0.0

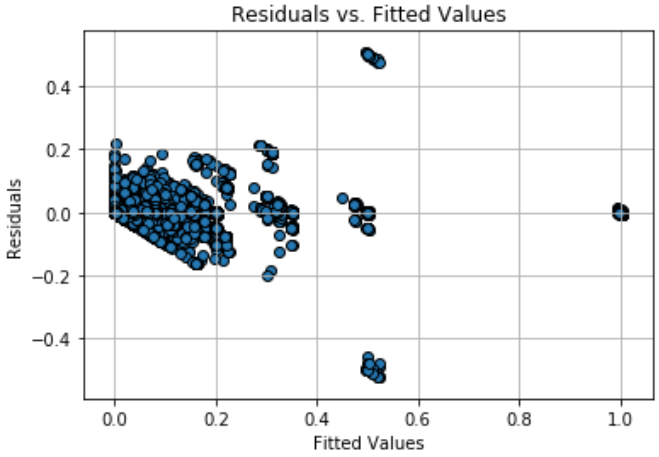
10-Fold Test RMSE: 0.020165775873018654



- n_neighbors = 2:

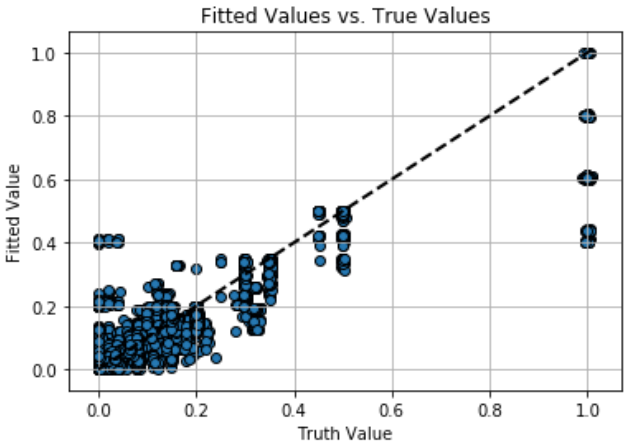
10-Fold Training RMSE: 0.02895622065867378
10-Fold Test RMSE: 0.03431224796735933

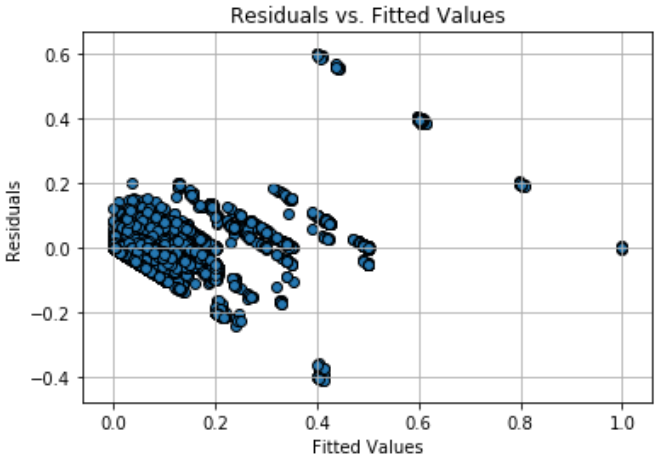




- n_neighbors = 5:

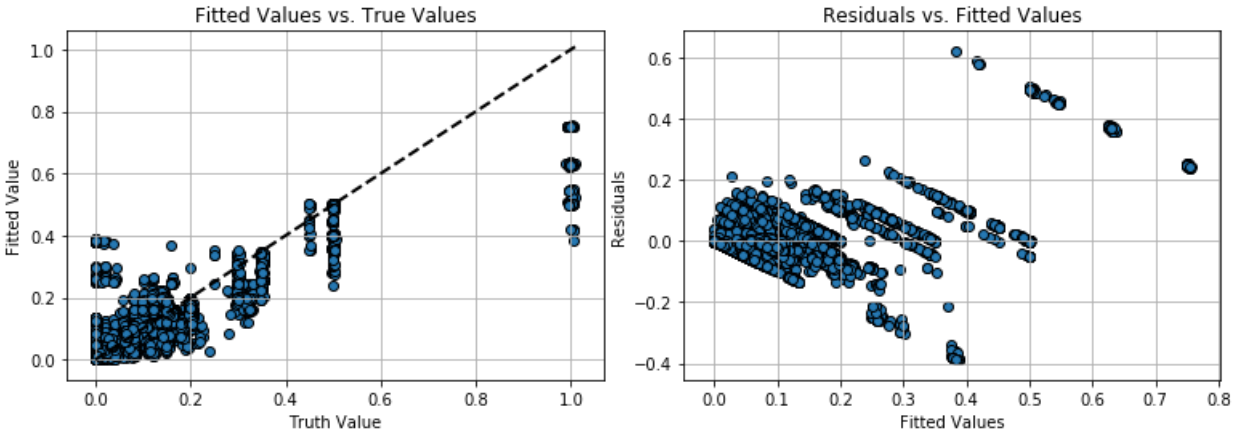
10-Fold Training RMSE: 0.026962640874196808
10-Fold Test RMSE: 0.04339364995943993





- n_neighbors = 8:

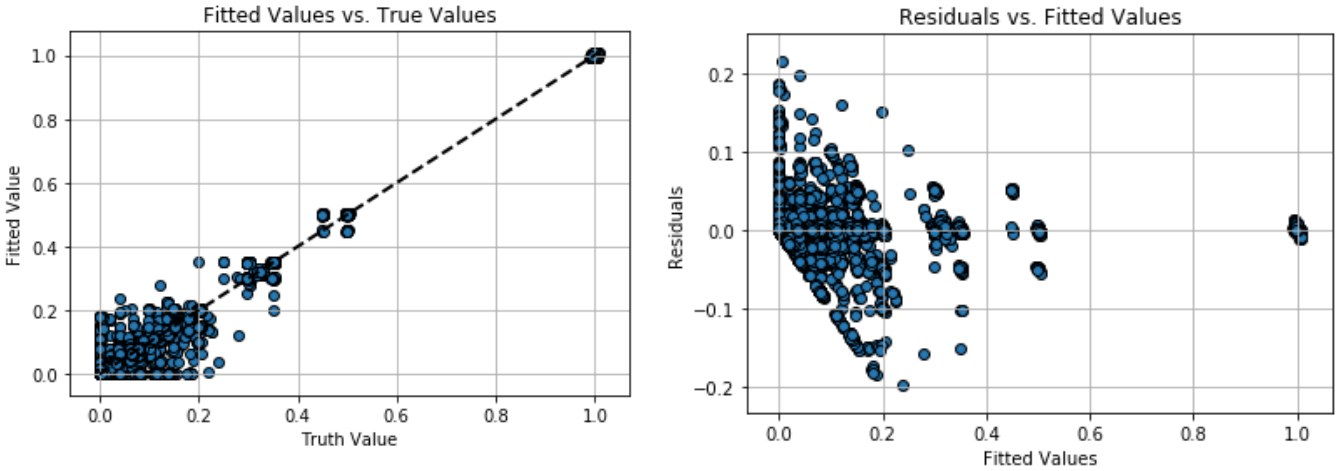
10-Fold Training RMSE: 0.03634385396366641
10-Fold Test RMSE: 0.046666173359154406



For weights as distance:

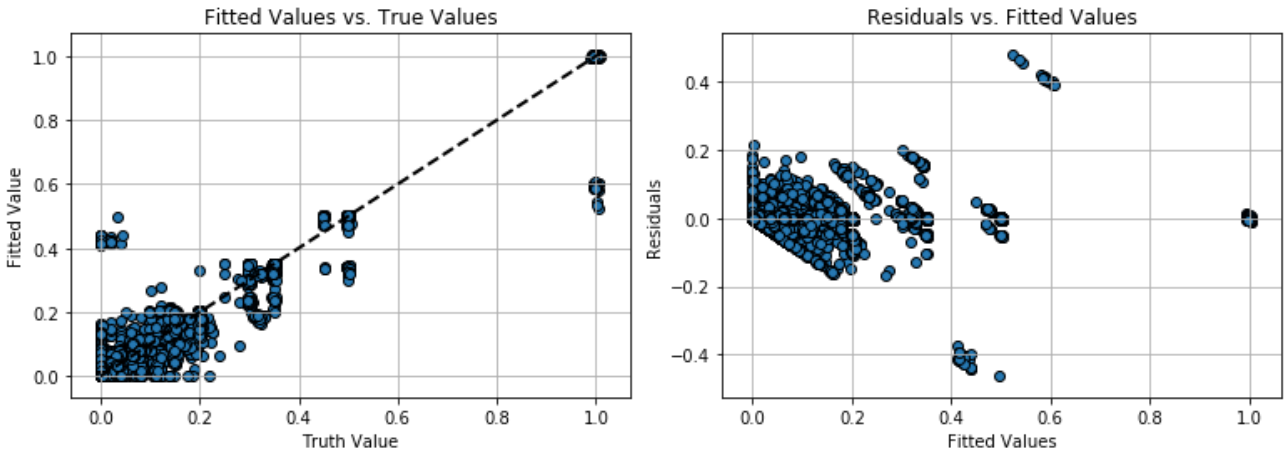
- n_neighbors = 1:

10-Fold Training RMSE: 0.0
10-Fold Test RMSE: 0.020165775873018654



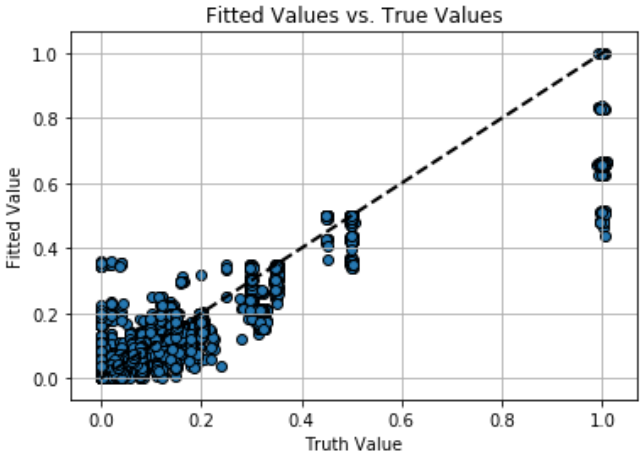
- n_neighbors = 2:

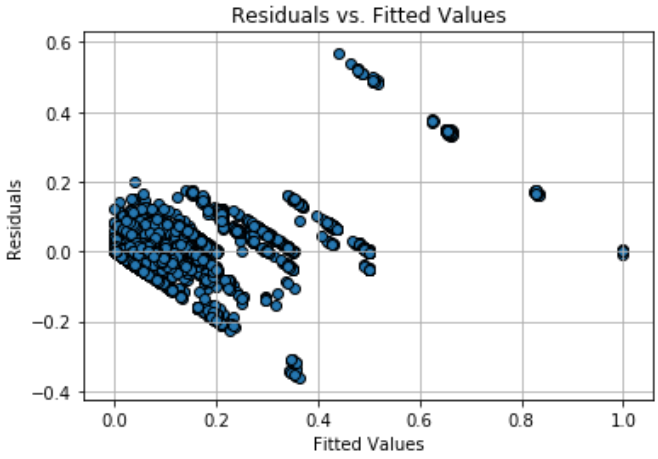
10-Fold Training RMSE: 0.0
10-Fold Testing RMSE: 0.030247962508845788



- n_neighbors = 5:

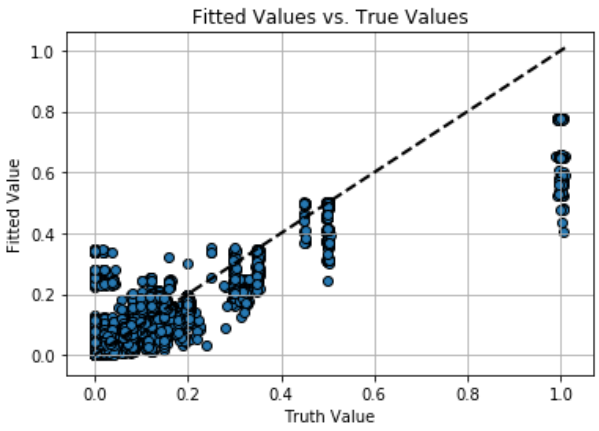
10-Fold Training RMSE: 0.0
10-Fold Test RMSE: 0.03884182477240271

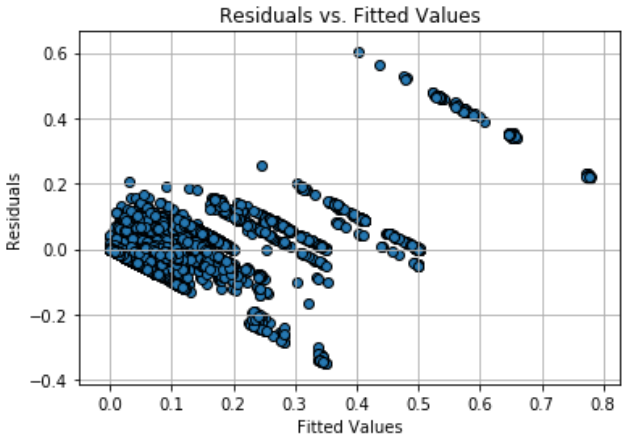




- n_neighbors = 8:

10-Fold Training RMSE: 0.0
10-Fold Test RMSE: 0.04303738774058584





Results:

As we can see from the average RMSEs and graphs, the smaller the number of n_neighrbors is, the better the performance KNR model will obtain. When n_neighbors = 1, both weights as distance and uniform give the best performance with minimum RMSEs as 0.0201; while overall, weight as uniform has better performance than that as distance when the number of n_neighbors increases.

3. Comparison:

In this section we compare these regression models we have used and have determined the best model at handling categorical features, at handling sparse features and at generating the overall best results. The training and testing RMSEs for each model are summarized as:

	Training RMSE	**Testing RMSE**
Linear regression + (01110) encoding	0.08834	0.08850
Random forest regression	0.06019	0.06037
Neural network regression	0.03313	0.04009
Multinomial regression (predict separately)	Wk0: 0.03222 Wk1: 0.00543 Wk2: 0.02254 Wk3: 0.00437 Wk4: 0.02057	Wk0: 0.03487 Wk1: 0.00656 Wk2: 0.02138 Wk3: 0.00467 Wk4: 0.02150
K-nearest neighbor regression	K = 1: 0.0 K = 2: 0.02896 K = 5: 0.02696 K = 8: 0.03634	K = 1: 0.02017 K = 2: 0.03431 K = 5: 0.04339 K = 8: 0.04667

Analysis:

From this project, we have learnt different regression models and their performance on dataset. it is obvious that more complex models yield better fitting results. In this case as the data is nonlinear, the fitting results of linear regression is not satisfying. The models of neural network, multinomial and k-nearest neighbor have relatively better results, comparing to random forest regression and linear regression since these two are simpler and cannot fit the data well. The k-nearest neighbor regression model is robust to noise elimination and effective for large dataset. Multinomial regression model is more robust to violations of assumptions of multivariate normality and equal variance-covariance matrices across groups. Neural network regression has the ability to implicitly detect complex nonlinear relationships between dependent and independent variables, to detect all possible interactions between predictor variables and the availability of multiple training algorithms. However, we should also prevent overfitting. Regularization will improve the performance of fitting and encoding methods of features will also influence the results.

(a) which model is best at handling categorical features?

Categorical encoding refers to transforming a categorical feature into one or multiple numeric features. Comparing to other regression algorithms, decision tree, KNN and random forest algorithms are all pretty good for category encoding features. For categorical variables, we need to calculate the proportion of positive targets for each individual value of the categorical variable, order them from large to small and calculate the optimal split in terms of target proportion. With a lot of values in categorical variable, we need to divide data into al lot of small samples, resulting in lots of noise compared to the real pattern. As the algorithm uses all the information (target proportion), the more values we obtain, the more noise will be incorporated into the model. When features are all categorical variables or contain both categorical and numerical variables, decision tree can perform well since it can handle both categorical and numerical variable. In general, random forest and decision tree are best for categorical features.

(b) which model is good at handling sparse features or not?

A generally safe model for sparse features is to perform a penalized regression. Lasso based penalties ensures a sparse solution, but an Elastic Net solution introduces some Ridge penalty and ensures a stable solution. The balance between Lasso and Ridge should be determined by the importances of gathered features. If they are all believed to be important, then Ridge might be the better while if only a few are important, Lasso might be better. We can try to tune the balance, but it usually may result in overfitting. For Lasso, L1 penalty term yields a sparse coefficient vector, which can be viewed as a feature selection method. However, there are some limitations for the Lasso: if the features have high correlation, the Lasso will only select one of them; and for problems where pp > nn, the Lasso will select at most nn parameters (nn and pp are the number of observations and parameters, respectively). These drawbacks make the Lasso empirically a suboptimal method in terms of predictability compared to ridge regression. For ridge regression, it offers better predictability in general; however, its interpretability is not as nice as the Lasso, as we can see from our results in this project.

(c) which model overall generates the best results?

From the table, we can know that K-NN regression when k = 1 has the best results, since KNN model is effective when the dataset is large and it is robust to noise training data. KNN' s decision boundary can take on any form because KNN is non-parametric and makes no assumption about the data distribution. Also, it doesn't assume an explicit form for f (X), providing a more flexible approach. As a result, the flexibility of KNN' s decision boundary takes a huge advantage. In this project, the KNN regression model suits the dataset well and makes the average RMSE here very low. Since when k = 1, meaning the model only calculates the nearest one neighbor when fitting the dataset, the other neighbors and factors will not have impact in 1-NN regression model, which makes the RMSE smaller.