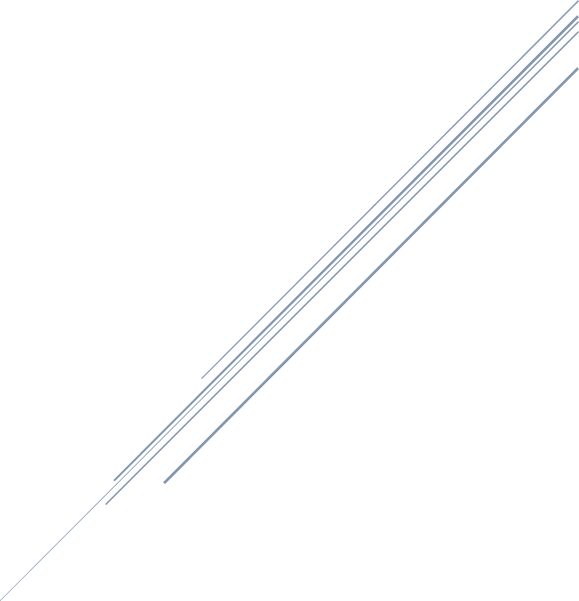
University of California, Los Angeles

EE 219 Winter 2018

*Project 4: Regression Analysis*

**

Group Member: Yunhao Ba & Xiaocheng Zhang & Shuangyu Li & Shengrong Wu

Date: March 4, 2018

Table of Contents

[**Introduction**](#_qw1pxhevoyfz) **2**

[**Question 1: Load the Dataset**](#_8tbieocfygok) **2**

[**Question 2: Predict**](#_mtzh6cbpzwh4) **3**

[a. Linear Regression Model](#_b164ny57br5k) 3

[b. Random Forest Regression Model](#_4vajm3cf0yvm) 32

[c. Neural Network Regression Model](#_xo3s8189xumc) 40

[d. Prediction of Backup Size for each Workflow](#_giyd39may3iq) 42

[e. K-nearest Neighbor Regression](#_c137pt4qb2e1) 54

[**Question 3: Comparison of Regression**](#_jan8c6k27cqq) **56**

## Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, the program will test and compare several regression models using 'Network backup Dataset', with utilization of cross-validation and regularization.

## Question 1: Load the Dataset

The plots of work flows of 20 days and 105 days are presented below.

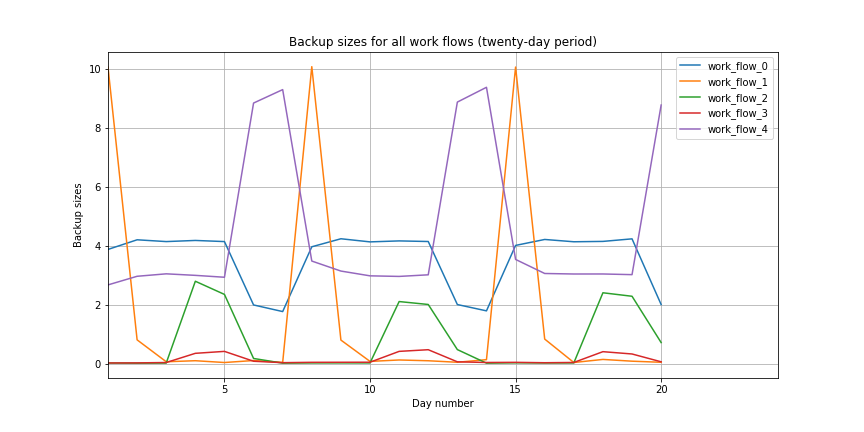


Figure: Work Flows of 20 Days

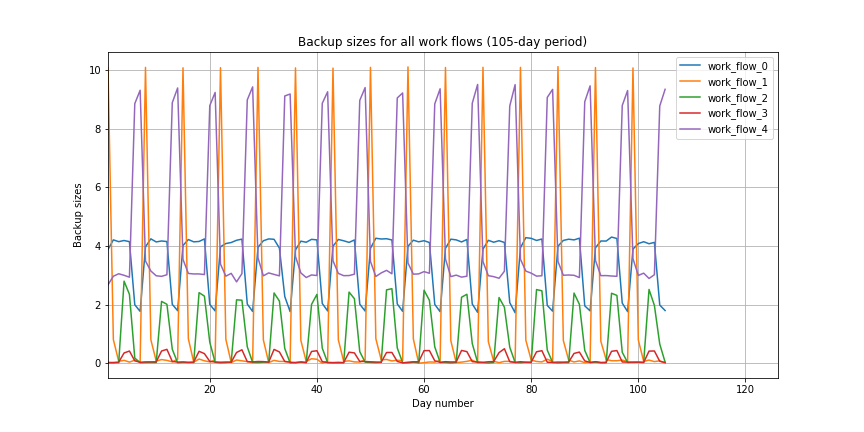


Figure: Work Flows of 105 Days

From the two figures, we could tell that there is a repeating period of 7 days for all 5 workflows. The dataset does present an obvious weekly pattern on backup activities.

## Question 2: Predict

#### a. Linear Regression Model

i. Modeling using raw data.

|  |  |
| --- | --- |
| Train RMSE | 0.103585393643 |
| Test RMSE | 0.103675847676 |

Table: Train and Test RMSE

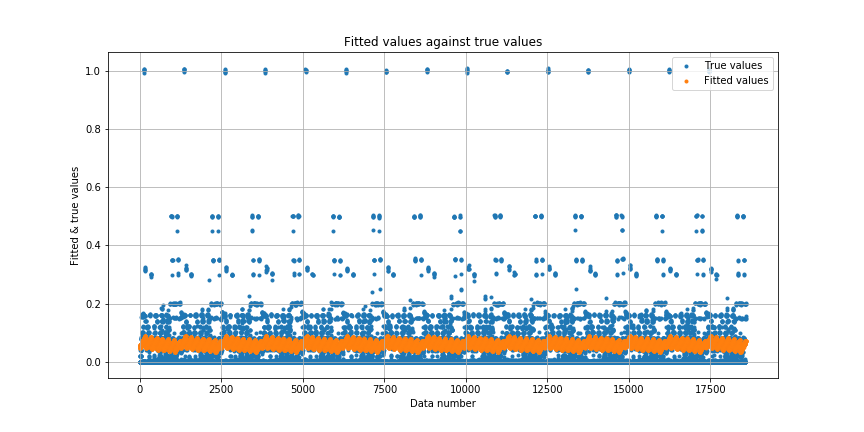


Figure: Fitted Values vs. True Values

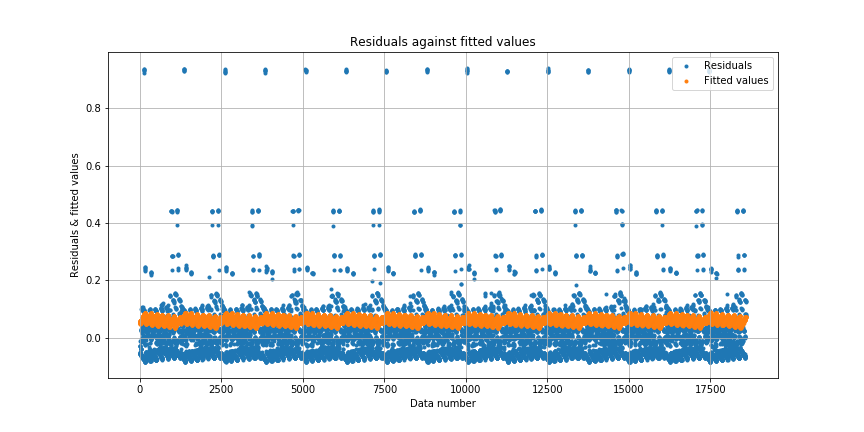


Figure: Residuals vs. Fitted Values

ii. Data Processing (Standardized)

|  |  |
| --- | --- |
| Train RMSE | 0.103585393643 |
| Test RMSE | 0.103675847676 |

Table: Train and Test RMSE

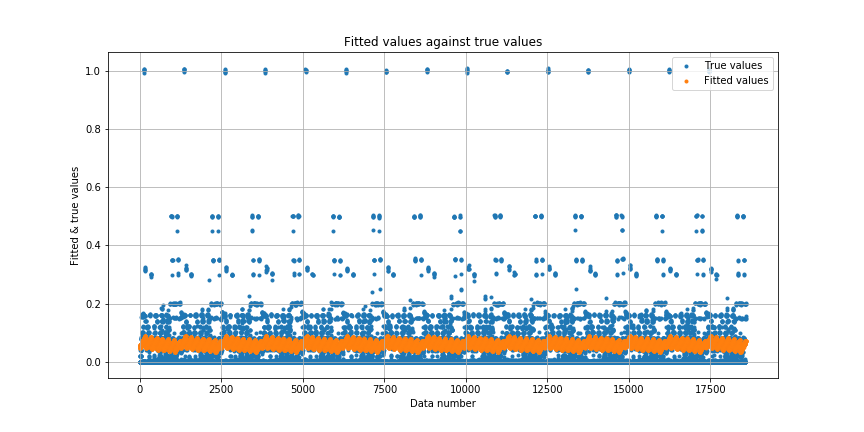


Figure: Fitted Values vs. True Values

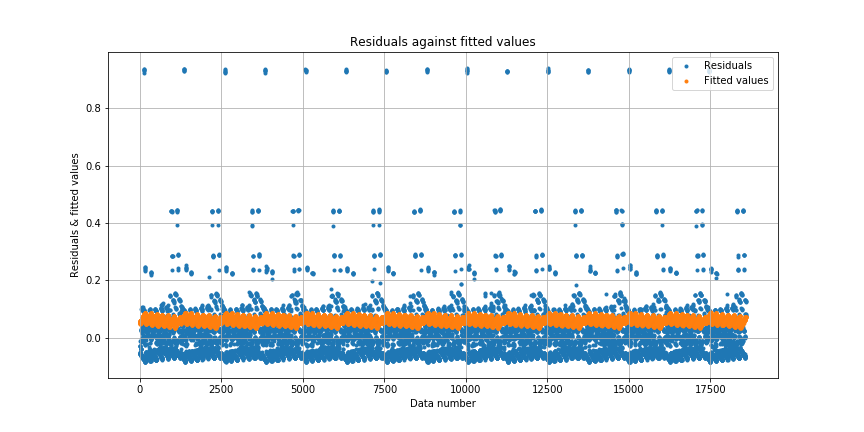


Figure: Residuals vs. Fitted Values

The train and test RMSE and figures are exactly the same as the results without standardization. It is plausible since linear regression is invariant under normalization, and the standardization does nothing but shift the points in space. Therefore the new model with normalized data should have the same parameters as the original model without normalization.

iii. Feature Selection

**F regression:**

The 3 most important variables are:

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

Results using the most important three features from F regression:

|  |  |
| --- | --- |
| Train RMSE | 0.103585682142 |
| Test RMSE | 0.103670661831 |

Table: Train and Test RMSE

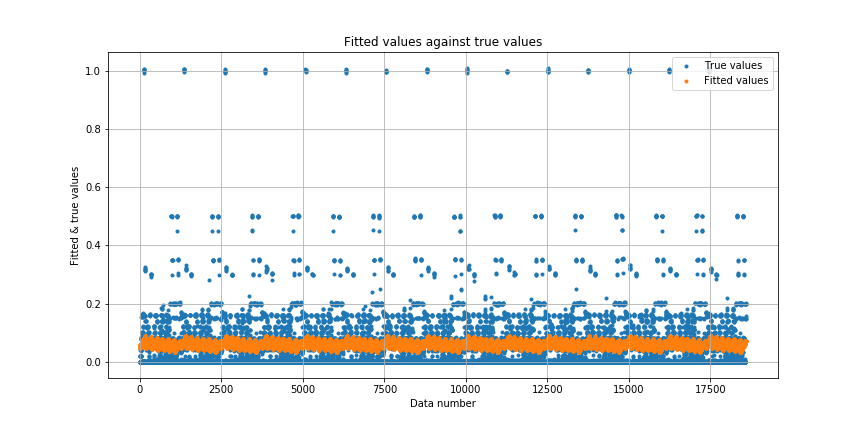


Figure: Fitted Values vs. True Values

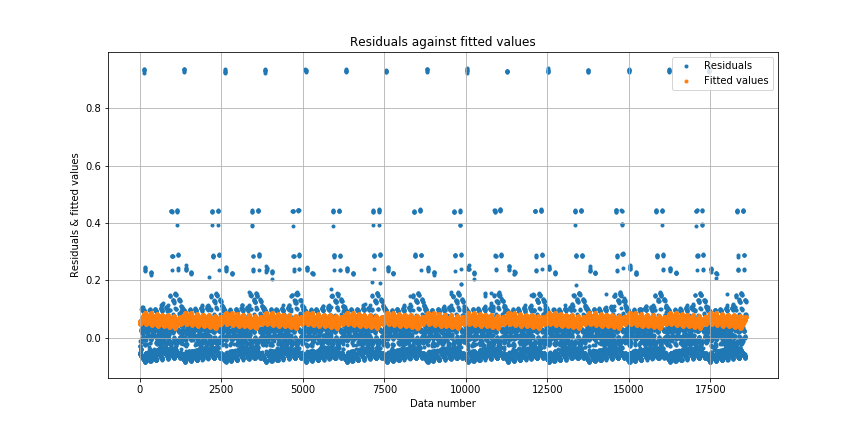


Figure: Residuals vs. Fitted Values

It can be observed that there is only very slightly change in the resulted train and test RMSE (in the level of 1e-6) after utilizing the most important three features. It might be resulted from the limitation of the simple linear regression model. Since the linear model is too naive, underfitting is a significant issue. No matter what features we utilized, the resulted RMSE cannot be improved significantly. The data points from the two plots above behave as such.

**Mutual Information Regression:**

Three most important variables are:

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

Results using the most important three features from F regression:

|  |  |
| --- | --- |
| Train RMSE | 0.103694528194 |
| Test RMSE | 0.103772293071 |

Table: Train and Test RMSE

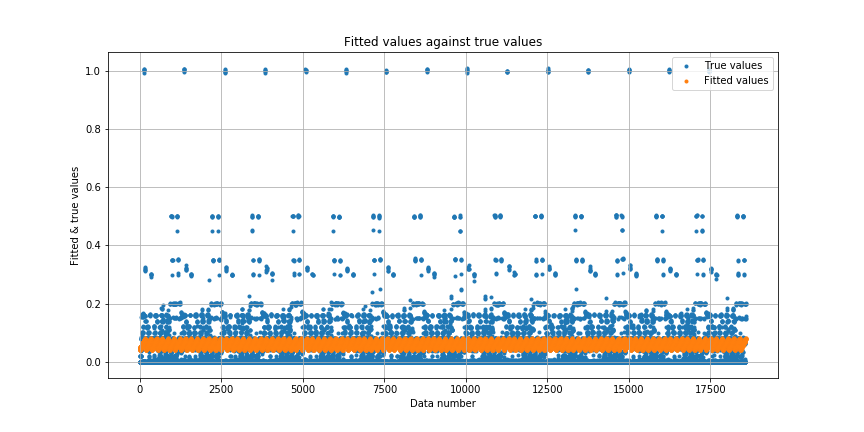


Figure: Fitted Values vs. True Values

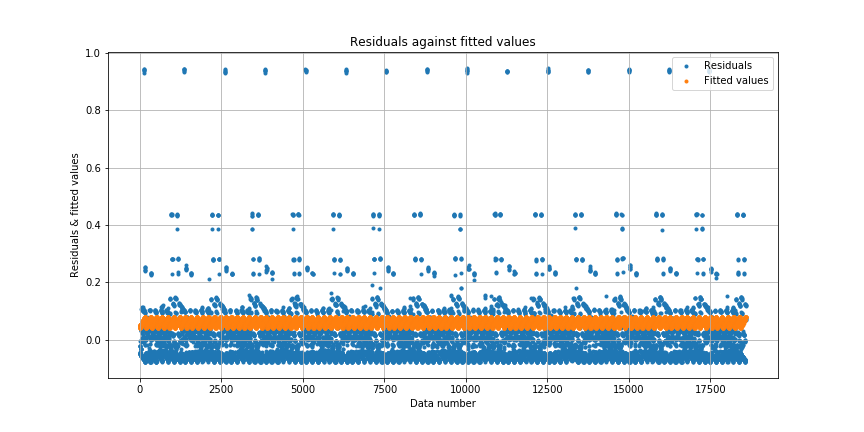


Figure: Residuals vs. Fitted Values

Again, for mutual information scores, due to the limitation of the simple linear regression model, the resulted RMSE does not change significantly. However, using important features from mutual information to train a new linear model will lead to a more even and flat fitted values as shown in the figure, while there is not much visual change for f-regression. The distribution of the different features results in this interesting observation.

iv. Feature Encoding

Performance of 32 encoding schemes with linear regression model is shown below. Higher index mean more colums are encoded in general.

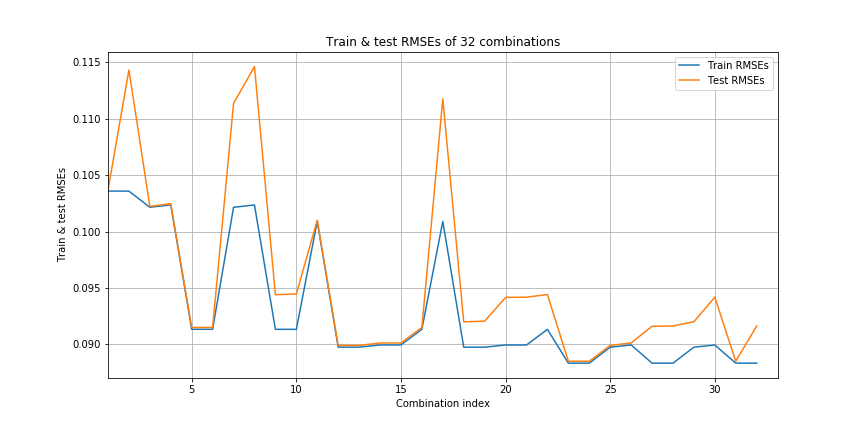


Figure: Train and Test RMSE's of 32 combinations

In general, the trend of RMSE is decreasing as the combination index increases, while there are some sudden increases followed by sudden decrease (peaks) along the way (when combination of index is 2, 7, 8 and 16). The best result among the data points above is shown below:

The minimum test RMSE is 0.0885042609364.

The best combination of features is:

One-Hot features: ['Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID']

Scalar features: ['Week #', 'File Name']

Performance and plots under the best combination:

|  |  |
| --- | --- |
| Train RMSE | 0.0883374486188 |
| Test RMSE | 0.0885042609364 |

Table: Train and Test RMSE

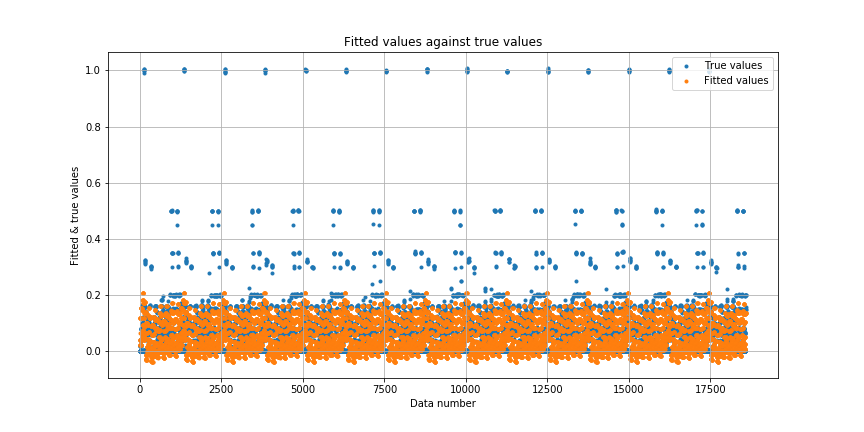


Figure: Fitted Values vs. True Values

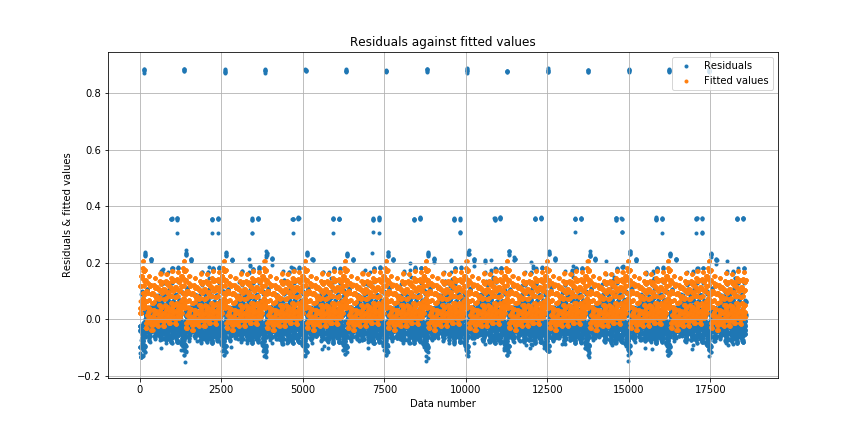


Figure: Residuals vs. Fitted Values

Intuitive explanation: The general trend is that when more features are encoded in One-Hot scheme, the performance is improved in terms of test RMSE (the combination indices are arranged in ascending order of the number of One-Hot features, e.g. All features are One-Hot encoded in Combination 32). One-Hot encoding assumes same distance between different feature, and utilizes more dimensions to represent a single feature. According to the results, we can observe that the best combination encodes the three most important features in the F regression into One-Hot scheme. It is intuitive since if we use more dimensions to represent the important features, the error will decrease accordingly. More specifically, One-Hot encoding encodes variables in orthogonal spaces. This advantage works well for features "Day of Week", "Work-flow ID" as different values in the feature could have large effect on the predicted backup size. While scalar encoding does not work too bad for features "Week index" and "File name". Possible reasons include that the features are not as important as the others, and that the scalar encoding captures the categorical relationship among feature values as well as One-Hot encoding.

v. Controlling ill-conditioning and over-fitting

In the above figure, there is a relatively large difference between the train and test RMSE's in certain combinations, e.g. Combination 2, Combination 7, Combination 8, etc. In these combinations, trained model fits the training data “too well”, and does not generalize well. That is the overfitting issue, and to avoid the overfitting, we need to introduce regularizers. The following regularizers all constrain the values of parameters and prevent from “cheating to overfit the data”.

**Ridge Regularizer:**

Overview:

When increasing the alpha, there is only slight change in the minimum test RMSE. However, the overfitting issue is alleviated significantly when we increase the alpha, especially when the alpha is very large. As shown in the figures below, when alpha is 100, there is almost no difference between train and test RMSE. Detailed results with different alpha are listed below, and the best model in terms of test RMSE is illustrated at the end.

|  |  |
| --- | --- |
| Value of Alpha | Minimum test RMSE |
| 0.0001 | 0.0885030338634 |
| 0.01 | 0.088502586986 |
| 1 | 0.0885052783239 |
| 100 | 0.0885145549195 |

Table: Minimum test RMSE at different values of Alpha

* Alpha = 0.0001

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0885030338634 |
| One-Hot features | 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID' |
| Scalar features | 'Week #', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.0001

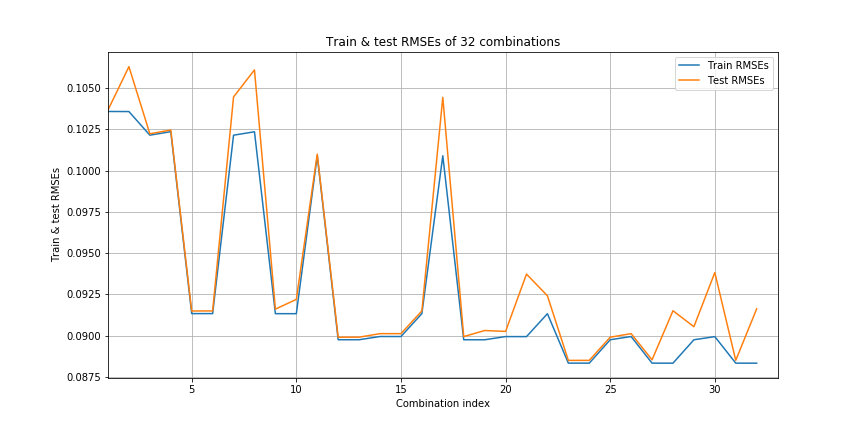


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 0.01

|  |  |
| --- | --- |
| Minimum test RMSE | 0.088502586986 |
| One-Hot features | 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID' |
| Scalar features | 'Week #', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.01

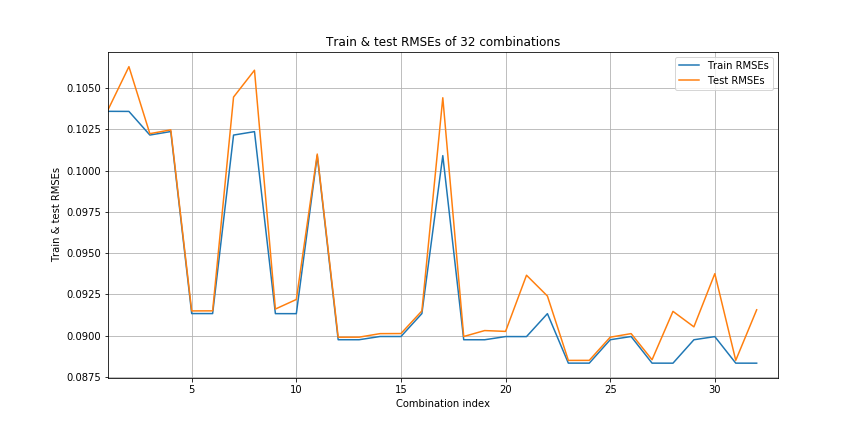


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 1

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0885052783239 |
| One-Hot features | 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |
| Scalar features | 'Week #' |

Table: Minimum test RMSE and Coding Scheme when alpha = 1

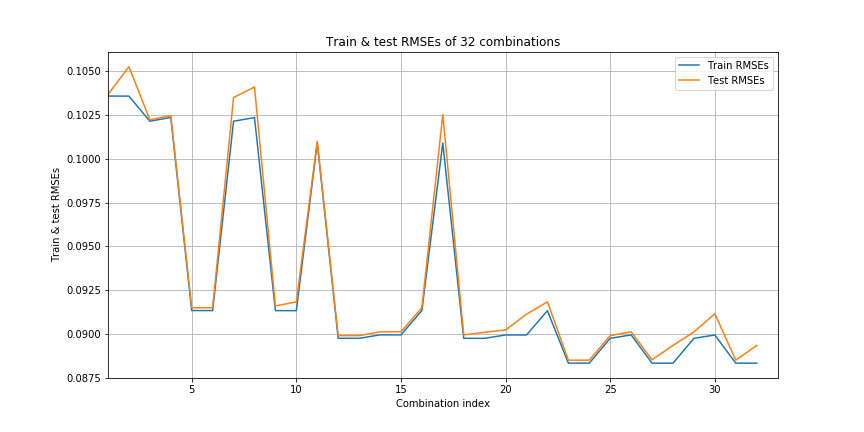


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 100

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0885145549195 |
| One-Hot features | 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |
| Scalar features | 'Week #' |

Table: Minimum test RMSE and Coding Scheme when alpha =100

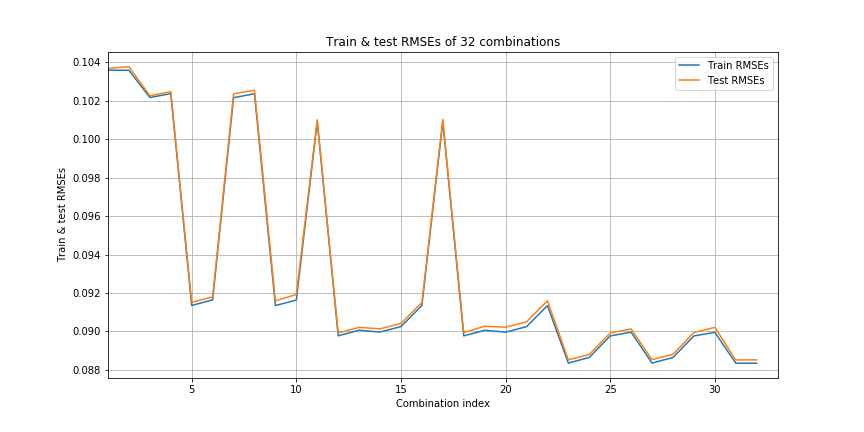


Figure: Train and Test RMSE's of 32 combinations

As we could see, the best minimum test RMSE occurs at alpha = 0.01.

The corresponding best combination of feature is:

One-Hot features: ['Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID']

Scalar features: ['Week #', 'File Name']

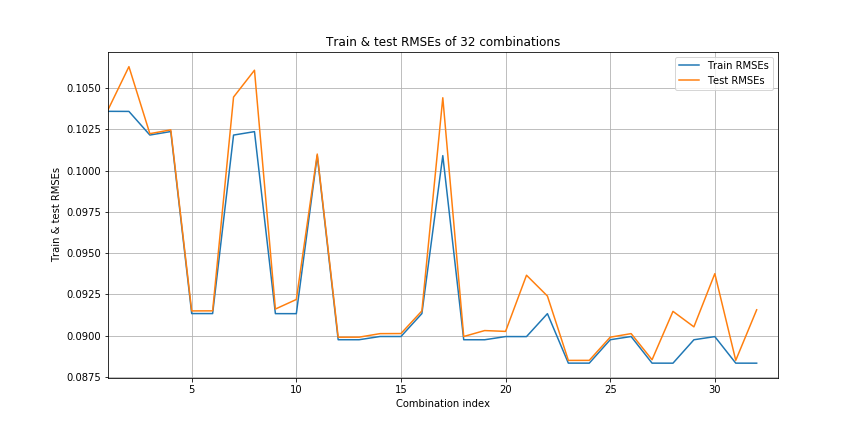


Figure: Train and Test RMSE's of 32 combinations

Best model using alpha = 0.01 and its best combination:

|  |  |
| --- | --- |
| Train RMSE | 0.0883375937521 |
| Test RMSE | 0.0885047267501 |

Table: Train and Test RMSE

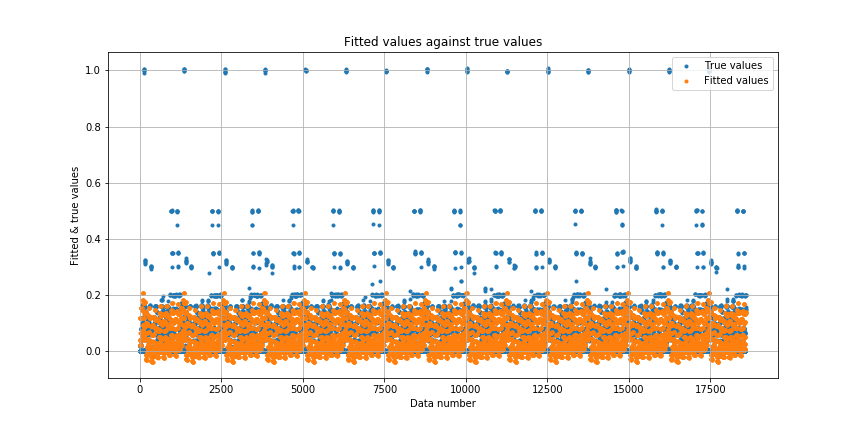


Figure: Fitted Values vs. True Values

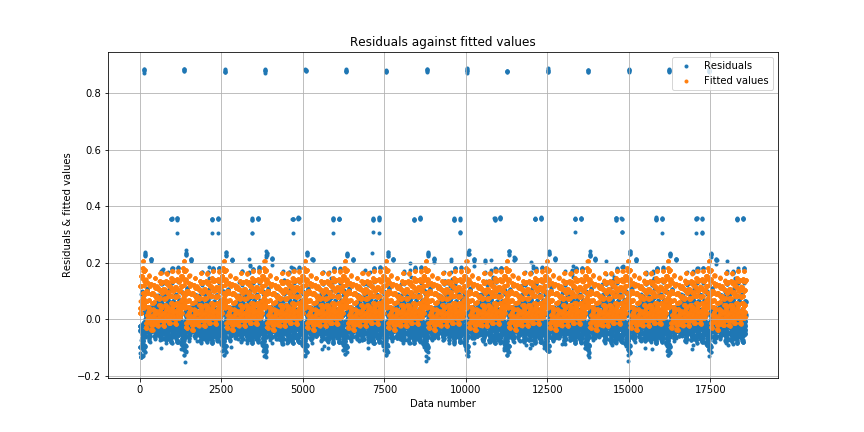


Figure: Residuals vs. Fitted Values

Conclusion: Ridge Regularizer can help with the overfitting issue, however, it can not improve the performance significantly in terms of the test RMSE. The shape of fitted values and residuals is roughly the same as the naive linear regression model. The prediction is good when the true values are small, and the prediction for samples with relatively larger true values is not so precise. The nature of linear regression model prevents it to fit data with extremely nonlinear distributions, which leads to this result.

**Lasso Regularizer:**

Overview:

Unlike the Ridge regularizer, when increasing the alpha, there is significant difference in the minimum test RMSE. When alpha is smaller than 1, the minimum test RMSE will decrease when alpha decreases. When alpha is bigger than 1, the test RMSE does not change when alpha increases, and the model becomes invariant under different encoding schemes. Besides, Lasso Regularizer can solve overfitting even when alpha is very small (e.g. 0.001). The difference between test RMSE and train RMSE remains very small with different alpha values. Detailed results with different alpha are listed below, and the best model in terms of test RMSE is illustrated at the end.

|  |  |
| --- | --- |
| Value of Alpha | Minimum test RMSE |
| 0.0001 | 0.0885082492502 |
| 0.01 | 0.0989736268589 |
| 1 | 0.104190237886 |
| 100 | 0.104190237886 |

Table: Minimum test RMSE at different values of Alpha

* Alpha = 0.0001

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0885082492502 |
| One-Hot features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |
| Scalar features | None |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.0001

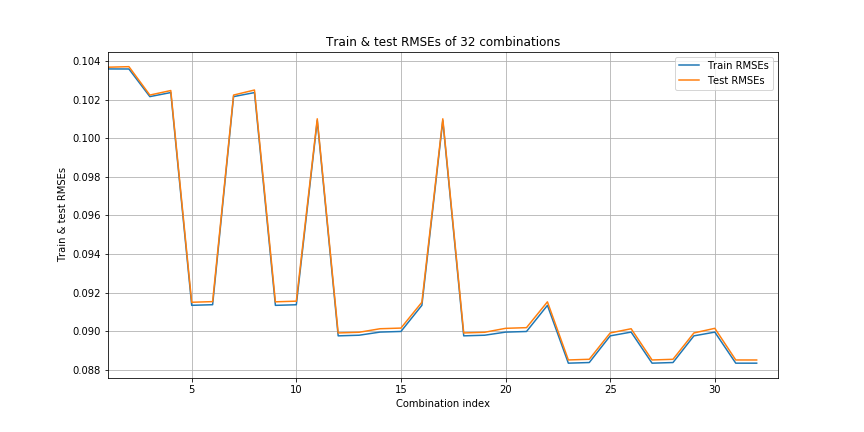


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 0.01

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0989736268589 |
| One-Hot features | 'Work-Flow-ID' |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.01

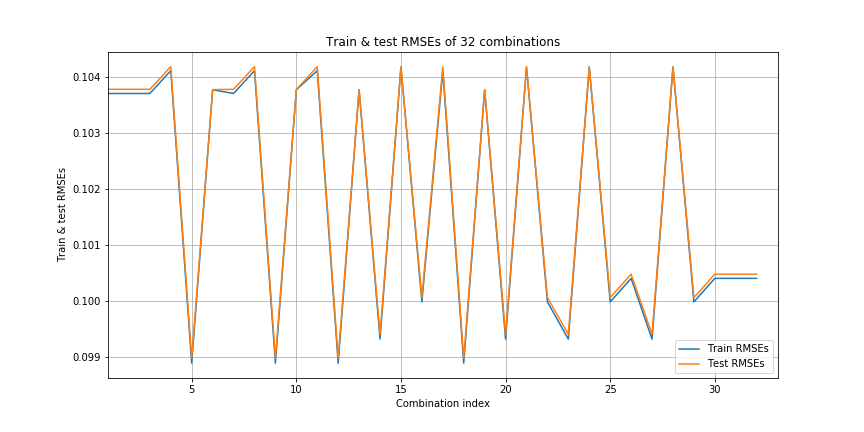


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 1

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 1

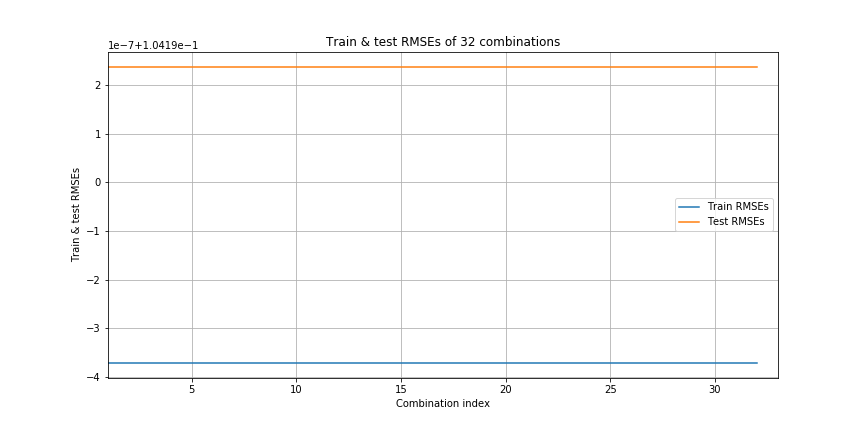


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 100

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 100

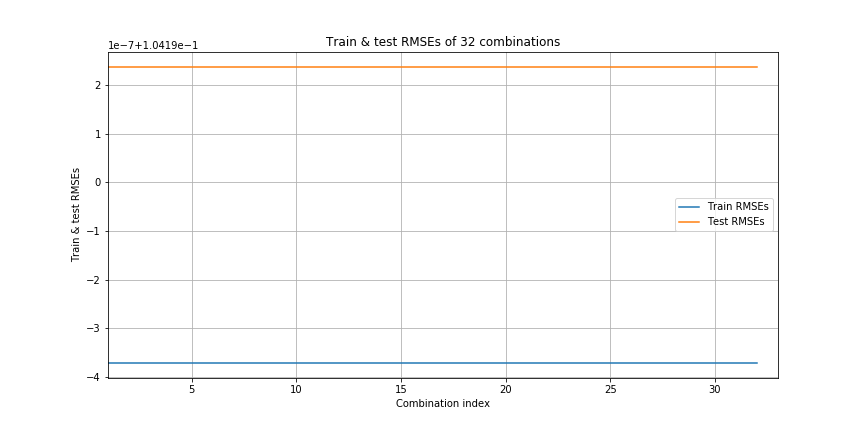


Figure: Train and Test RMSE's of 32 combinations

As we could see, the best minimum test RMSE occurs at alpha = 0.0001.

The corresponding best combination of feature is:

One-Hot features: ['Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File

Name']

Scalar features: []

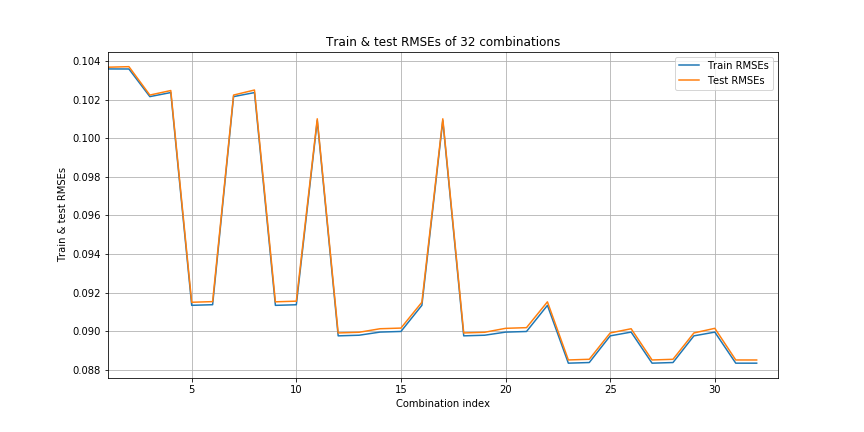


Figure: Train and Test RMSE's of 32 combinations

Best model using alpha = 0.0001 and its best combination:

|  |  |
| --- | --- |
| Train RMSE | 0.0883429584855 |
| Test RMSE | 0.0885082492502 |

Table: Train and Test RMSE

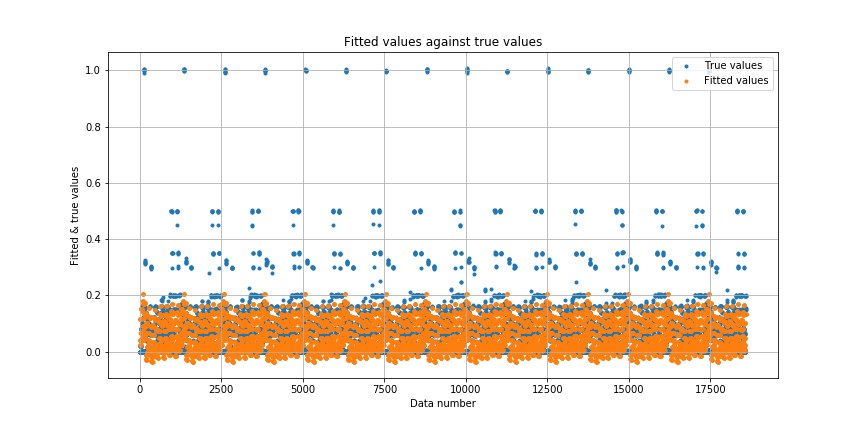


Figure: Fitted Values vs. True Values

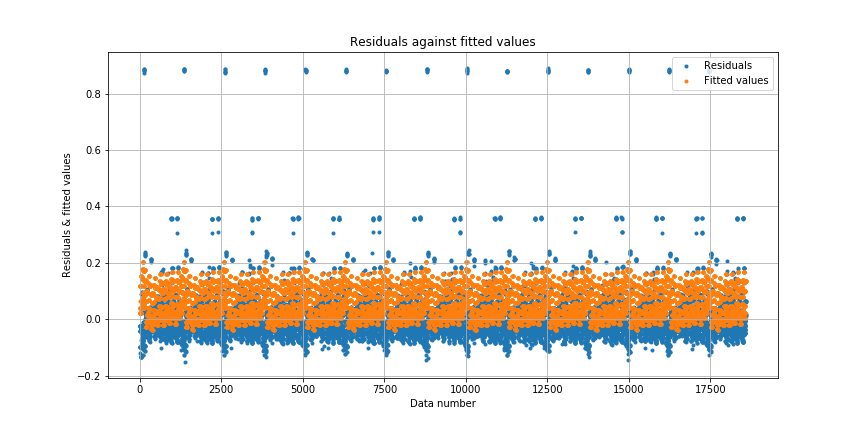


Figure: Residuals vs. Fitted Values

Conclusion: Lasso Regularizer can help with the overfitting issue, however, it can not improve the performance significantly in terms of the test RMSE as well. Similar conclusion to the Ridge Regularizer can be obtained: the shape of fitted values and residuals is roughly the same as the naive linear regression model. The prediction is good when the true values are small, and the prediction for samples with relatively larger true values is not so precise. The nature of linear regression model prevents it to fit data with extremely nonlinear distributions, which leads to this result.

**Elastic Net Regularizer:**

Overview:

There are two parameters in this regularizer, alpha is the constant that multiplies the penalty terms and the control the overall regularization strength, and l1\_ratio is the term control the relative strength between L1 term and L2 term (l2\_ratio is just 1 - l1\_ratio). The penalty is a combination of L1 and L2. Similar to Lasso Regularizer, when increasing the alpha, there is significant difference in the minimum test RMSE. When alpha is smaller than 1, the minimum test RMSE will decrease when alpha decreases. When alpha is bigger than 1, the test RMSE does not change when alpha increases, and the model becomes invariant under different encoding schemes. Besides, Elastic Net Regularizer can also solve overfitting even when alpha is very small (e.g. 0.01). The difference between test RMSE and train RMSE remains very small with different alpha values. Detailed results with different alpha and l1\_ratio are listed below, and the best model in terms of test RMSE is illustrated at the end. We observed that the best result is achieved when both alpha and l1\_ratio are small.

|  |  |  |
| --- | --- | --- |
| Value of Alpha | Value of l1\_ratio | Minimum test RMSE |
| 0.01 | 0.1 | 0.089110136077 |
| 0.5 | 0.0940302613713 |
| 0.9 | 0.0977983901844 |
| 1 | 0.1 | 0.104190237886 |
| 0.5 | 0.104190237886 |
| 0.9 | 0.104190237886 |
| 100 | 0.1 | 0.104190237886 |
| 0.5 | 0.104190237886 |
| 0.9 | 0.104190237886 |

Table: Minimum test RMSE at different values of Alpha and l1\_ratio

* Alpha = 0.01, l1\_ratio = 0.1

|  |  |
| --- | --- |
| Minimum test RMSE | 0.089110136077 |
| One-Hot features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |
| Scalar features |  |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.01, l1\_ratio = 0.1

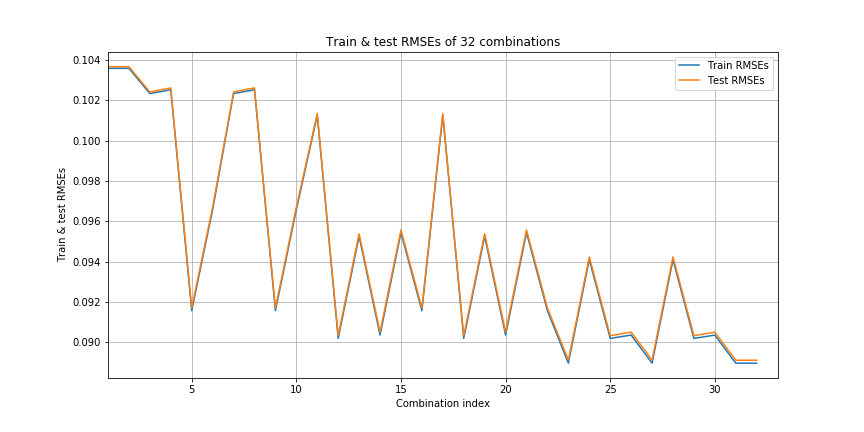


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 0.01, l1\_ratio = 0.5

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0940302613713 |
| One-Hot features | 'Day of Week', 'Work-Flow-ID' |
| Scalar features | 'Week #', 'Backup Start Time - Hour of Day', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.01, l1\_ratio = 0.5

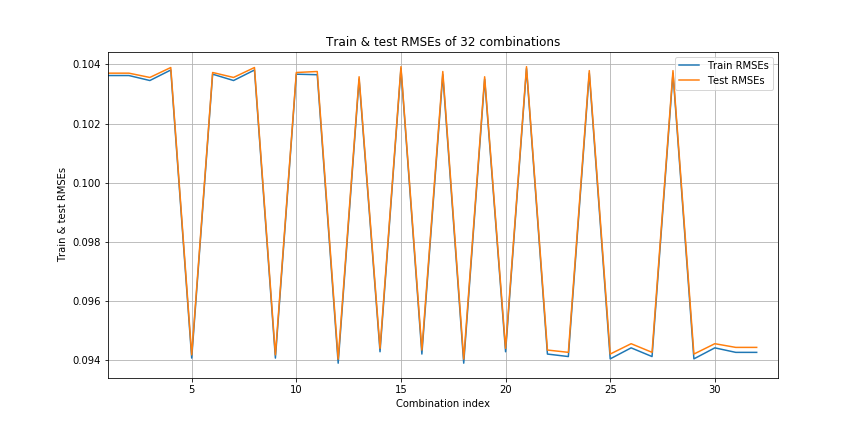


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 0.01, l1\_ratio = 0.9

|  |  |
| --- | --- |
| Minimum test RMSE | 0.0977983901844 |
| One-Hot features | 'Work-Flow-ID' |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 0.01, l1\_ratio = 0.9

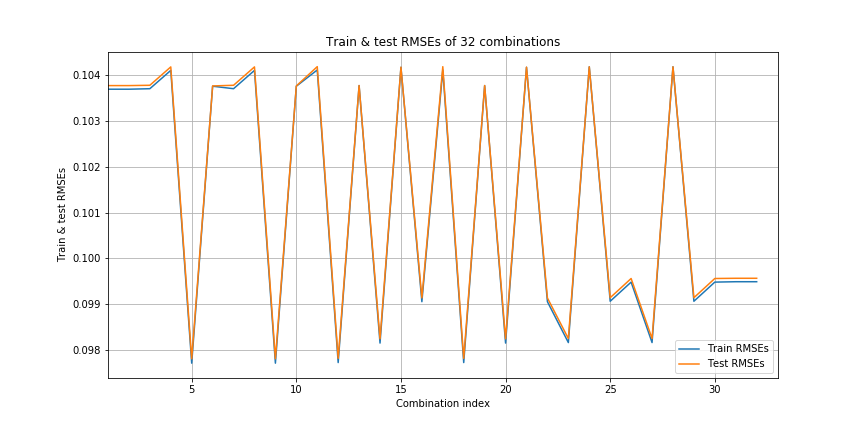


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 1, l1\_ratio = 0.1

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 1, l1\_ratio = 0.1

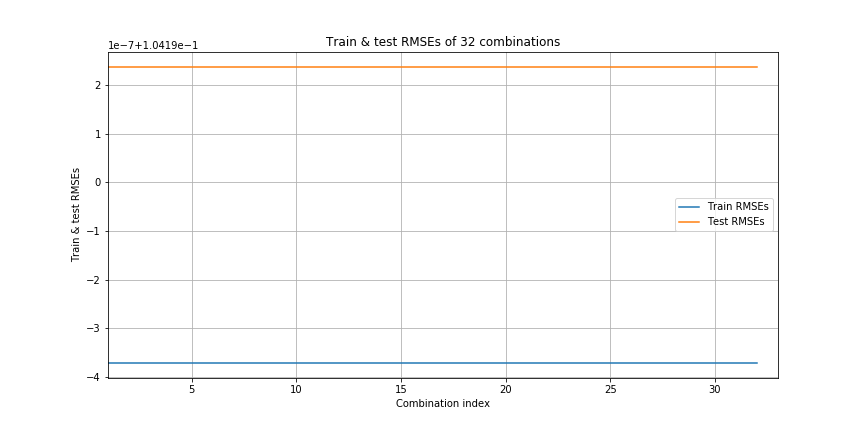


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 1, l1\_ratio = 0.5

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 1, l1\_ratio = 0.5

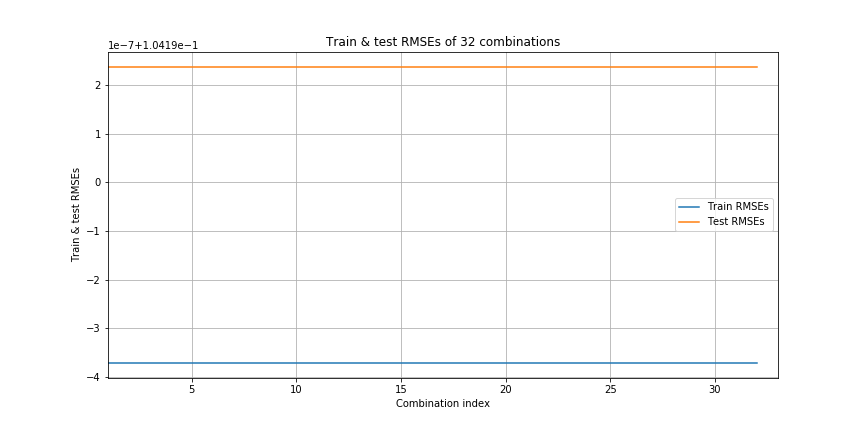


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 1, l1\_ratio = 0.9

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 1, l1\_ratio = 0.9

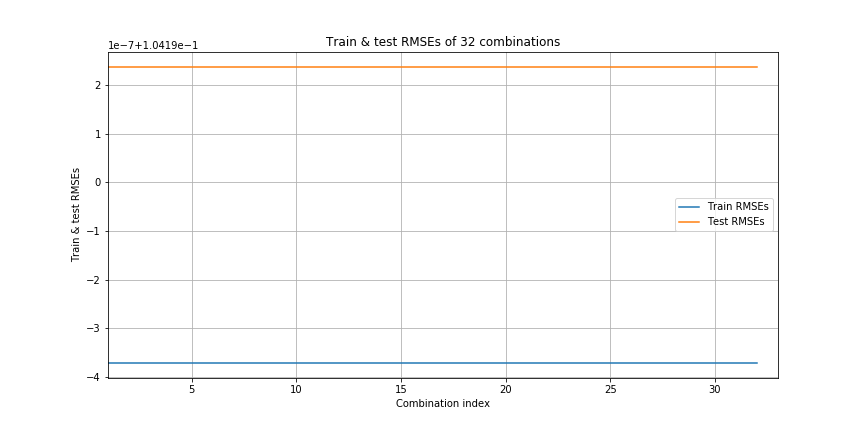


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 100, l1\_ratio = 0.1

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 100, l1\_ratio = 0.1

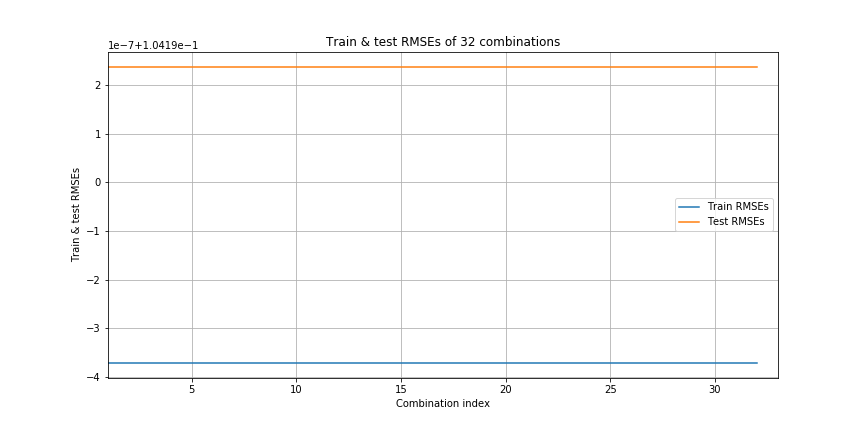


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 100, l1\_ratio = 0.5

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 100, l1\_ratio = 0.5

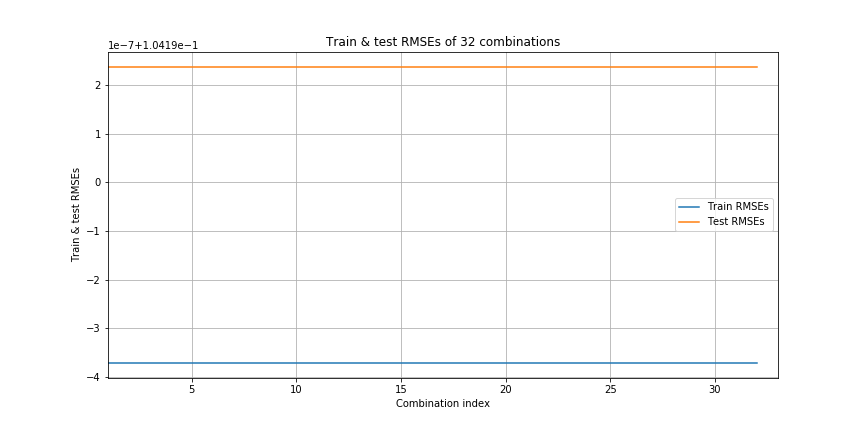


Figure: Train and Test RMSE's of 32 combinations

* Alpha = 100, l1\_ratio = 0.9

|  |  |
| --- | --- |
| Minimum test RMSE | 0.104190237886 |
| One-Hot features |  |
| Scalar features | 'Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File Name' |

Table: Minimum test RMSE and Coding Scheme when alpha = 100, l1\_ratio = 0.9

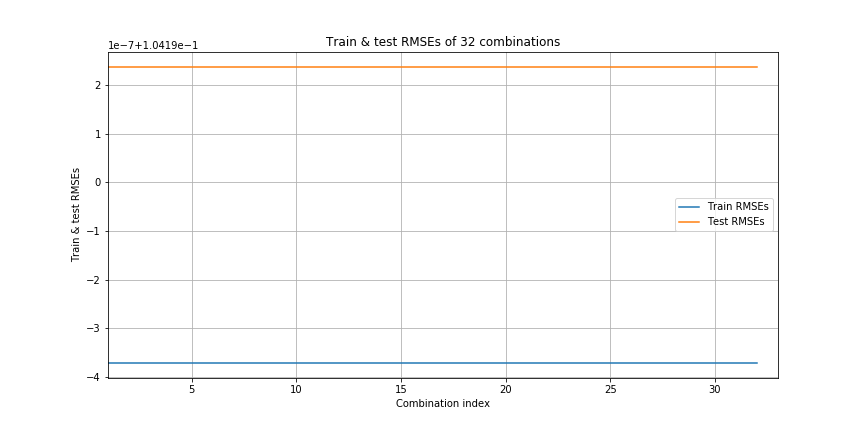


Figure: Train and Test RMSE's of 32 combinations

As we could see, the best minimum test RMSE occurs at alpha = 0.01 and l1\_ratio = 0.1.

The corresponding best combination is:

One-Hot features: ['Week #', 'Day of Week', 'Backup Start Time - Hour of Day', 'Work-Flow-ID', 'File

Name']

Scalar features: []

Using alpha = 0.01, l1\_ratio\_ratio = 0.1 and its best combination:

|  |  |
| --- | --- |
| Train RMSE | 0.0889641150629 |
| Test RMSE | 0.089110136077 |

Table: Train and Test RMSE

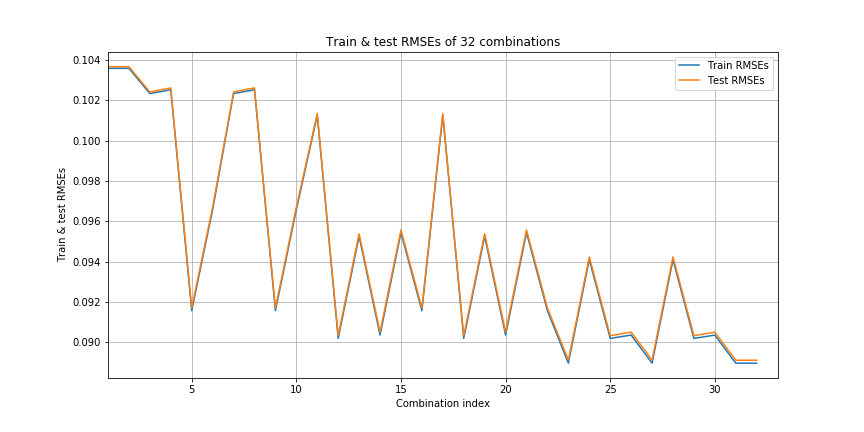


Figure: Train and Test RMSE's of 32 combinations

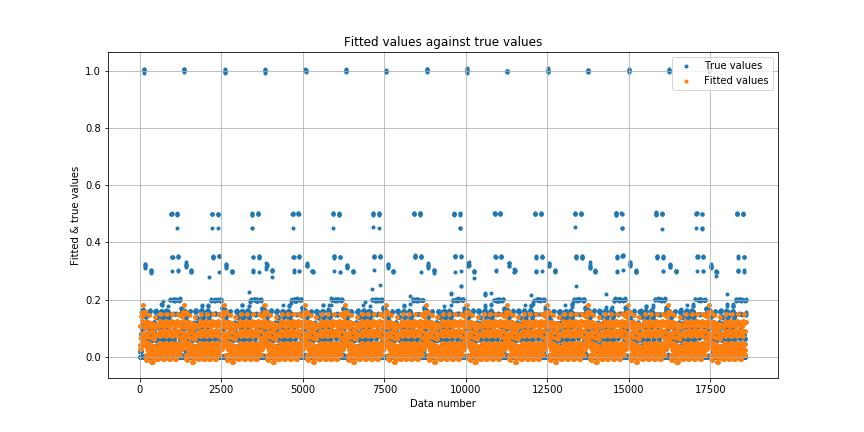


Figure: Fitted Values vs. True Values

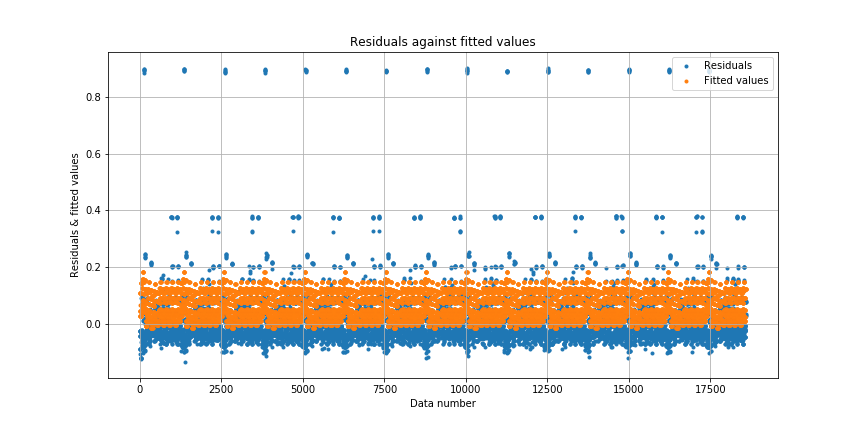


Figure: Residuals vs. Fitted Values

Conclusion: Elastic Net Regularizer can help with the overfitting issue, however, it can not improve the performance significantly in terms of the test RMSE as well. Similar conclusion to other regularizers can be obtained: the shape of fitted values and residuals is roughly the same as the naive linear regression model. The prediction is good when the true values are small, and the prediction for samples with relatively larger true values is not so precise. The nature of linear regression model prevents it to fit data with extremely nonlinear distributions, which leads to this result.

**Compare the fitted coefficients:**

Unregularized model with scalar encoding:

Model coefficients:

[ 1.71674482e-05 -2.37717728e-03 1.36734990e-03 2.39021236e-03 4.56647178e-05]  
Mean value: 0.00028864343052

Unregularized model with best combination:

Model coefficients:

[ 3.90732847e-02 -1.31024360e-02 -2.05054552e-02 -5.48662926e-03  
 -5.94496601e-03 3.03882082e-03 1.25833235e-03 -2.04832495e-02  
 -2.13397211e-02 7.51333516e-03 3.31765845e-02 -2.26524291e-03  
 1.72924529e-03 3.84426612e-02 -1.41233527e-02 -4.05355434e-02  
 -5.75150822e-02 7.20622686e-02 1.12114502e-05 4.60289450e-05]  
Mean value: -0.000247495267672

Ridge model with best combination:

Model coefficients:

[ 3.88702891e-02 -1.33364259e-02 -2.07295136e-02 -5.67202203e-03  
 -6.16543957e-03 2.84362915e-03 1.06420641e-03 -2.07449954e-02  
 -2.15534228e-02 7.26357646e-03 3.29376532e-02 -2.48764947e-03  
 1.45956152e-03 3.87936939e-02 -1.40863470e-02 -4.08302026e-02  
 -5.81141154e-02 7.11116947e-02 2.02963891e-05 8.62647419e-05]  
Mean value: -0.000463463419308

Lasso model with best combination:

Model coefficients:

[ 0 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0 4.37513766e-02  
 -6.96907942e-03 -1.44485670e-02 0 0  
 7.71880392e-03 5.93830959e-03 -2.10345963e-02 -2.18506978e-02  
 5.78177598e-03 3.14450311e-02 -2.79293029e-03 2.96261921e-06  
 5.17787530e-02 0 -2.56788220e-02 -4.23859230e-02  
 8.64849397e-02 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0 0  
 0 0 0]  
Mean value: 0.00155144978689

Elastic Net Model with best combination:

Model coefficients:

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.03314429 -0.00224642 -0.00985427 0. 0. 0. 0. -0.01361543 -0.01395416 0.00176253 0.02613158 0. 0. 0.04271847 0. -0.02295918 -0.03901929 0.07591347 0. 0.

0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]  
Mean value: 0.00123843792922

|  |  |
| --- | --- |
| Model | Mean of coefficients |
| Unregularized (scalar encoding) | 0.00028864343052 |
| Unregularized (best encoding combination) | -0.000247495267672 |
| Ridge (best encoding combination) | -0.000463463419308 |
| Lasso (best encoding combination) | 0.00155144978689 |
| Elastic Net (best encoding combination) | 0.00123843792922 |

Table: Mean coefficients of different models

Conclusion:

The model coefficients using Lasso and Elastic Net models is relatively more sparse compared with the unregularized one (with a lot of zeros). It is interpretable since both Lasso and Elastic Net models incorporate the L1 norm as the penalty. L1 regularization will ensure the sparsity, which leads to the models with sparse coefficients. As to the Ridge model, we can observe that the resulted coefficients are more close to the origin compared with the unregularized one, since L2 regularization will cause the coefficients shifting towards zero. Similar results can be observed in Elastic Net Model as well, since Elastic Net Model utilizes both L1 and L2 regularization.

#### b. Random Forest Regression Model

i. 10 fold validation

|  |  |
| --- | --- |
| Train RMSE | 0.060514412777 |
| Test RMSE | 0.0605990304804 |
| Out Of Bag error | 0.34311621073 |

Table: Minimum Train and Test RMSE and Out of Bag Error

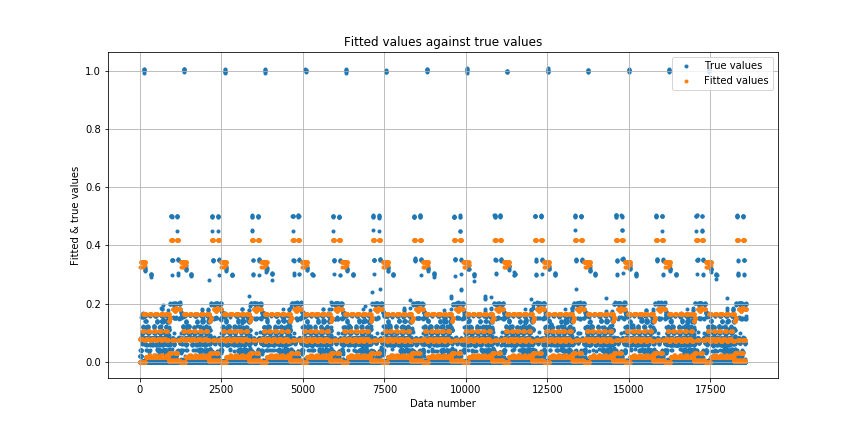


Figure: Fitted Values vs. True Values

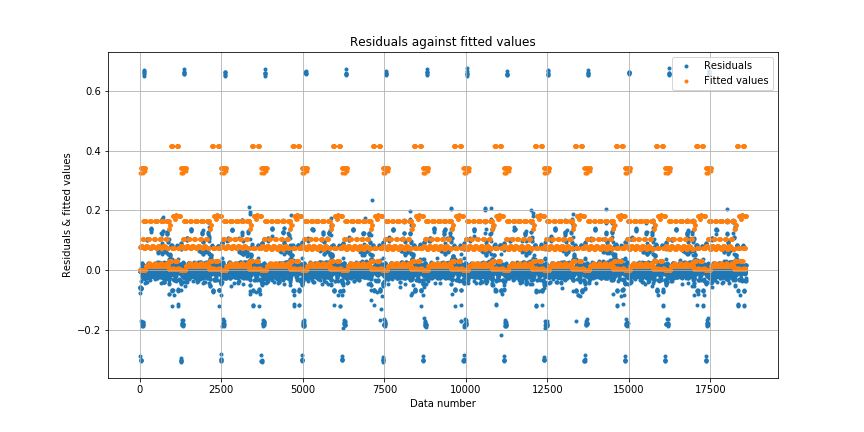


Figure: Residuals vs. Fitted Values

Random forest with default setting can achieve better performance than the linear regression model. That is because random forest can provide more nonlinearity to fit the training data.

ii.

Find best tree number and best feature number to minimize oob error:

|  |  |
| --- | --- |
| Minimum Out Of Bag Error | 0.316476649167 |
| Best tree number | 49 |
| Best feature number | 3 |

Table: Minimum oob error, best tree number, and best feature number

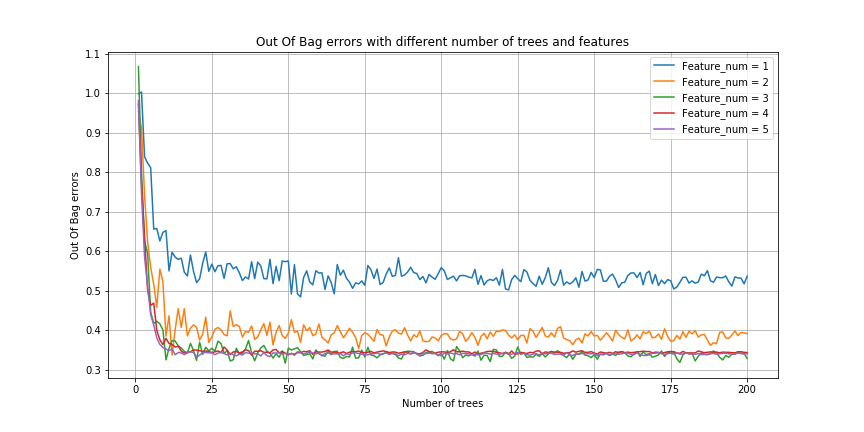


Figure: oob error vs. number of trees and features

Find best tree number and best feature number to minimize RMSE:

|  |  |
| --- | --- |
| Minimum RMSE | 0.0594710982746 |
| Best tree number | 7 |
| Best feature number | 3 |

Table: Minimum RMSE, best tree number, and best feature number

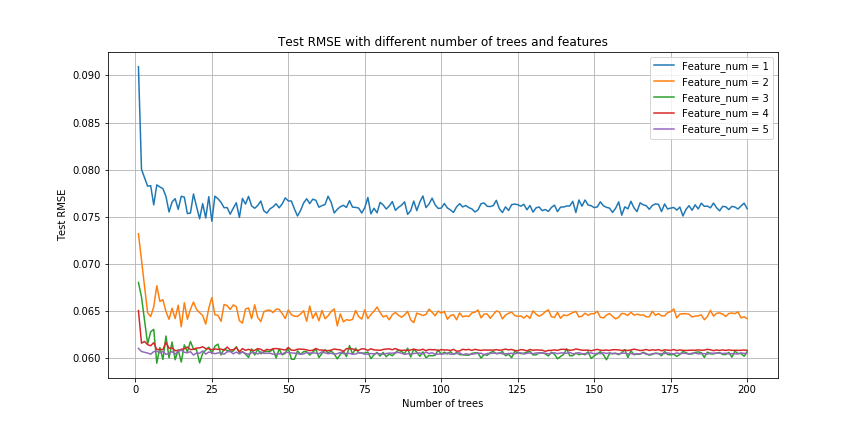


Figure: Test RMSE vs. number of trees and features

From both test RMSE and the Out-of-Bag error, their values drop dramatically at the beginning when the number of trees increases for feature numbers from 1 to 5. As the number of trees gets larger, the curves become more even and flat. There is a difference when feature number is smaller than 3, where we could improve the performance by increasing the number of features. The performance for feature number of 3, 4, 5 are roughly the same, especially when the tree number is large. The best test RMSE is achieved when the number of features is 3, so we choose as our best parameter.

The actual results change from time to time due to the randomness. Here only exhibits the result for one trial. Train and test RMSE, along with oob error under best model using test RMSE as the criterion (tree number = 7 and feature number = 3):

|  |  |
| --- | --- |
| Train RMSE | 0.0622698079894 |
| Test RMSE | 0.0625002197091 |
| Out of Bag Error | 0.501978409403 |

Table: Train RMSE, Test RMSE, and OOB under best model

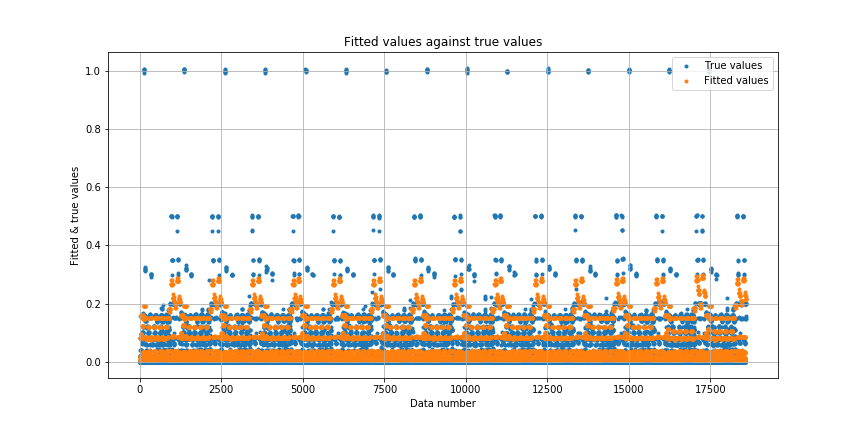


Figure: Fitted Values vs. True Values

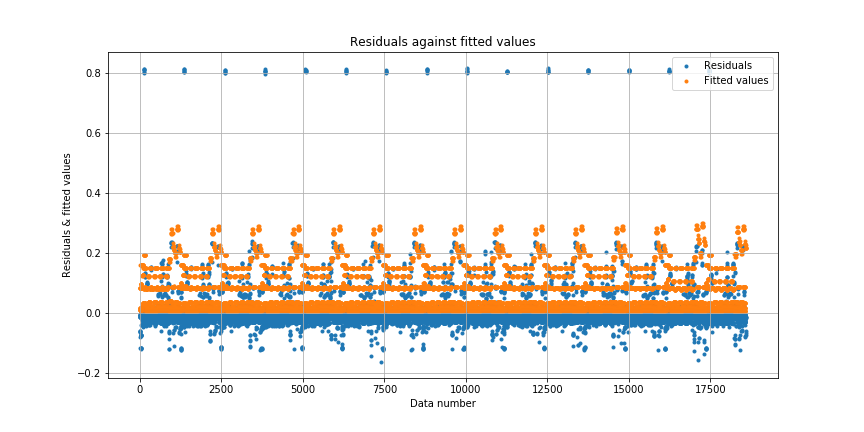


Figure: Residuals vs. Fitted Values

iii.

From the experiments above, we set the optimal maximum number of feature as 3, and iterate over different tree numbers from 1 to 200 and tree depths from 4 to 10. We pick the max depth as the tested parameters, since it is another very important hyperparameters affecting the overall results in the random forest algorithm. Generally speaking, the deeper the tree is, the more nonlinear the decision boundaries are. This nonlinearity will lead to better performance. The plots are shown below.

Find Best model for best OOB error

|  |  |
| --- | --- |
| Minimum out of bag error | 0.0155004048689 |
| Best tree number | 199 |
| Best max depth | 10 |

Table: Best OOB error, tree number and max depth

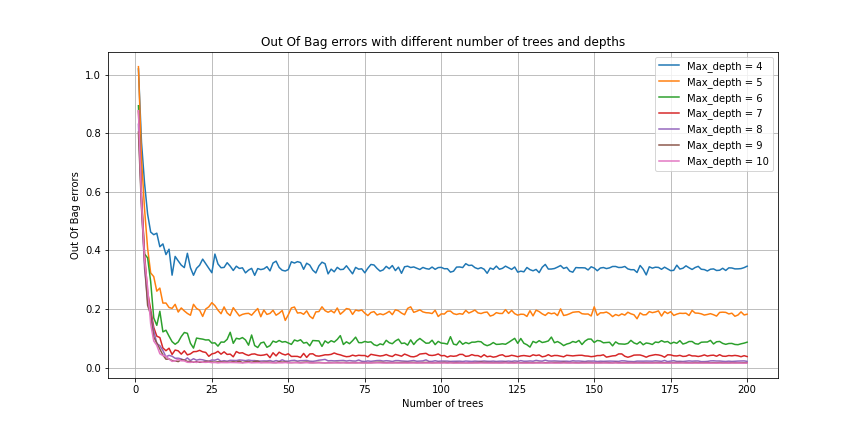


Figure: oob error vs. number of trees and depths

Find Best model for best minimum RMSE

|  |  |
| --- | --- |
| Min RMSE | 0.0135348089485 |
| Best tree number | 112 |
| Best max depth | 10 |

Table: Best OOB error, tree number and max depth

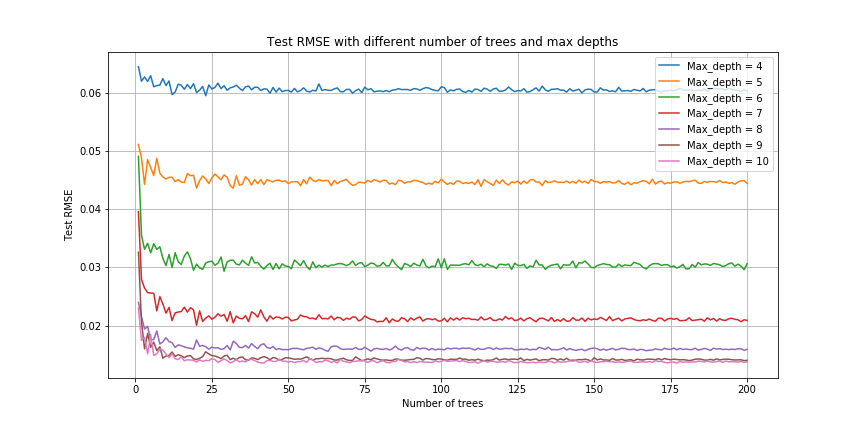


Figure: RMSE vs. number of trees and max depths

We could see that the performance improves rapidly, increasing tree depth from 4 to 9, and tree number from 1 to 20. The error stays in a plateau when these parameters rise further. Hence to achieve minimal out of bag error, we'd choose tree number of 199 and max depth of 10; to achieve minimized RMSE, we'd choose tree number of 112 and max depth of 10.

iv. Feature importance

The actual results change from time to time due to the randomness. Here only exhibits the result for one trial.

Best model results when choosing tree number of 112 and max depth of 10 (using test RMSE as the criterion):

The feature importances from the best random regression is reported below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train RMSE | 0.0113901185016 | | | | |
| Test RMSE | 0.0136852107899 | | | | |
| Out Of Bag error | 0.0159808562862 | | | | |
| Feature name | Week | Day of Week | Backup Start Time - Hour of Day | Work-Flow-ID | File Name |
| Feature importances | 0.00456987 | 0.29402742 | 0.3405605 | 0.1683513 | 0.19249091 |

Table: Train RMSE, Test RMSE, OOB error, Feature name, and feature importances

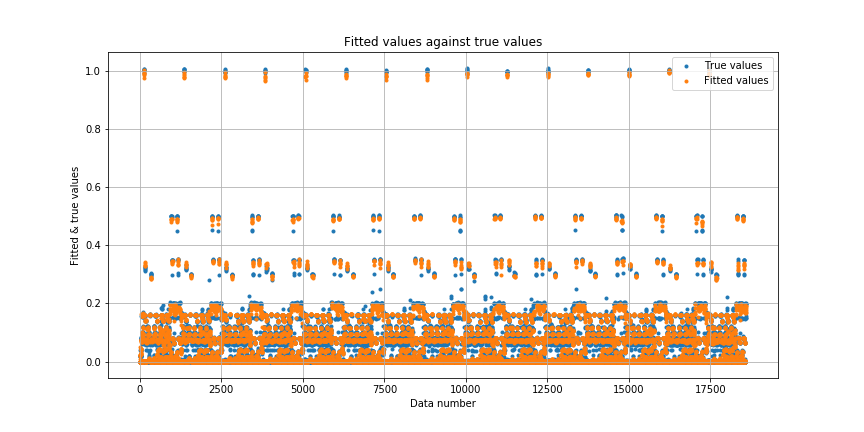


Figure: Fitted Values vs. True Values

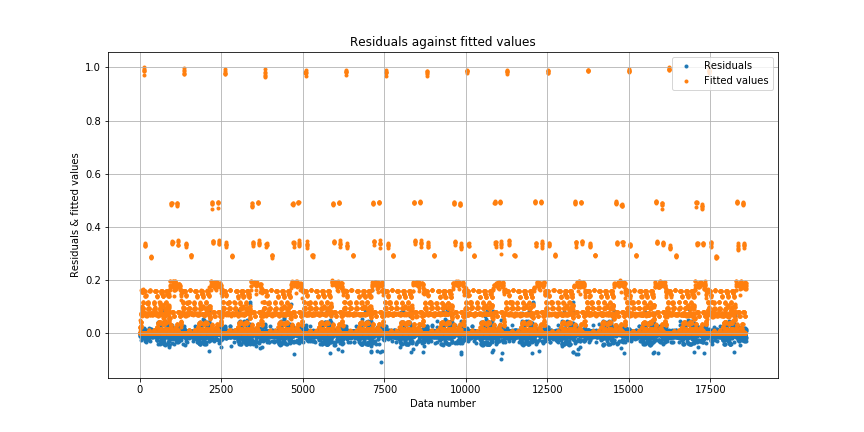


Figure: Residuals vs. Fitted Values

Conclusion: From the figure above, we can observe that the prediction is rather precise. The difference between the fitted values and true values is small even when the true values are very large, and the residuals are very small for all the samples. Random forest model could provide a lot of nonlinearity to fit the data, which leads to this remarkable performance.

v. Visualize the tree structure

The visualized result indicates that the root node is “Backup Start Time - Hour of Day”, and it is the most important feature in our best model. However, it does not mean all the trees in the random forest must use this most important feature as the roots.

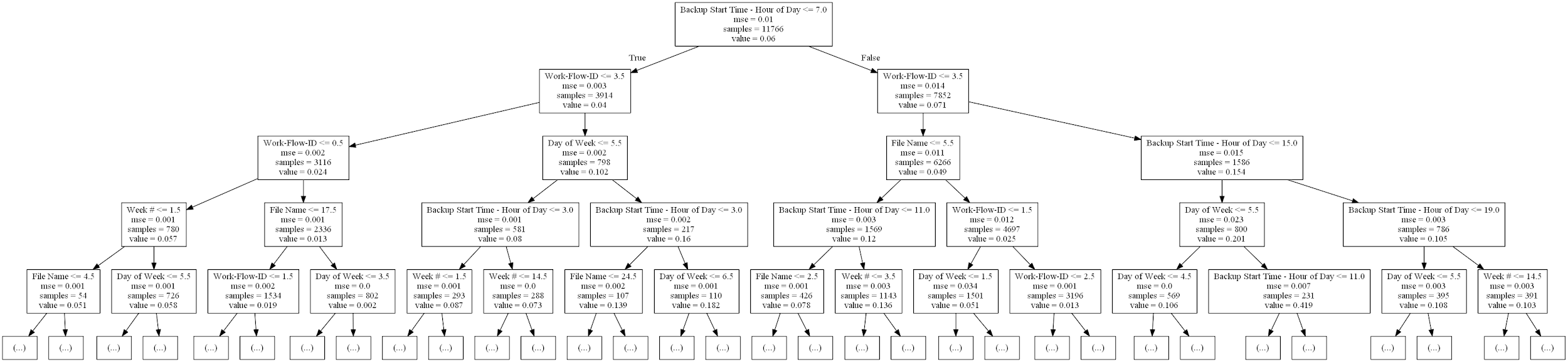


Figure: Decision Tree

Here shows another tree whose root is not the most important feature. The root of the below tree is “File Name”.

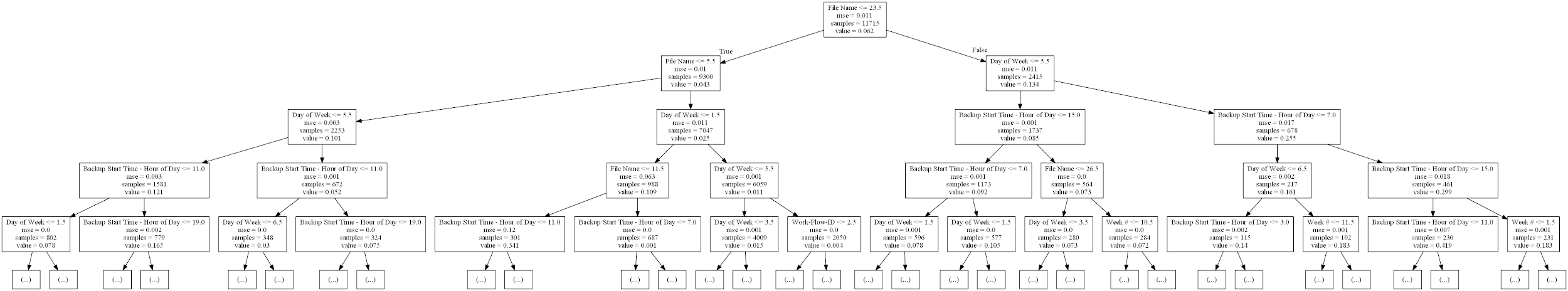


Figure: Decision Tree

As a conclusion, the root of a tree largely depends on the actual training data used for each tree. Even though the most important feature is “Backup Start Time - Hour of Day” in our best model, it does not necessarily mean each tree must utilize this feature as the root. For a specific tree, the feature that could achieve the best separation in the root node is affected by the random process during training.

#### c. Neural Network Regression Model

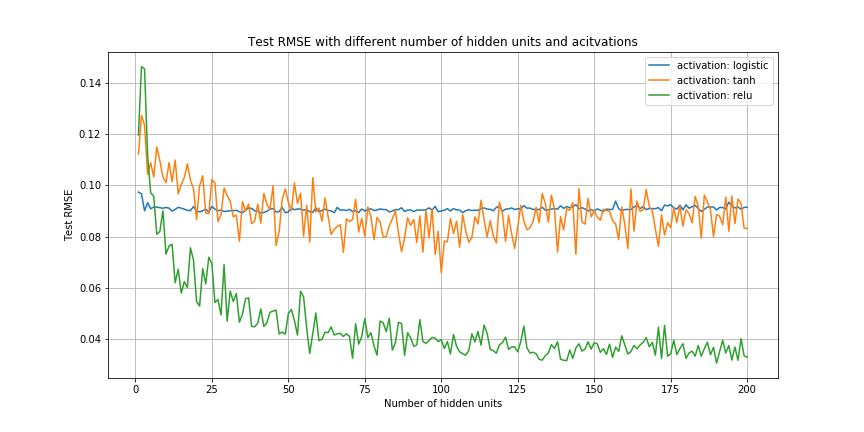


Figure: Test RMSE vs. number of hidden units and activations

From the figure above, we could tell that relu activation method has significantly lower RMSE when the number of hidden units is larger than 25.

Minimum RMSE is achieved using the following parameter:

|  |  |
| --- | --- |
| Best hidden unit number | 190 |
| Best activation | relu |

Table: Minimum RMSE with best hidden unit number and best activation

Results with the best model:

|  |  |
| --- | --- |
| Minimum Train RMSE | 0.0191107607826 |
| Minimum Test RMSE | 0.0343930517408 |

Table: Minimum Train RMSE and Minimum Test RMSE

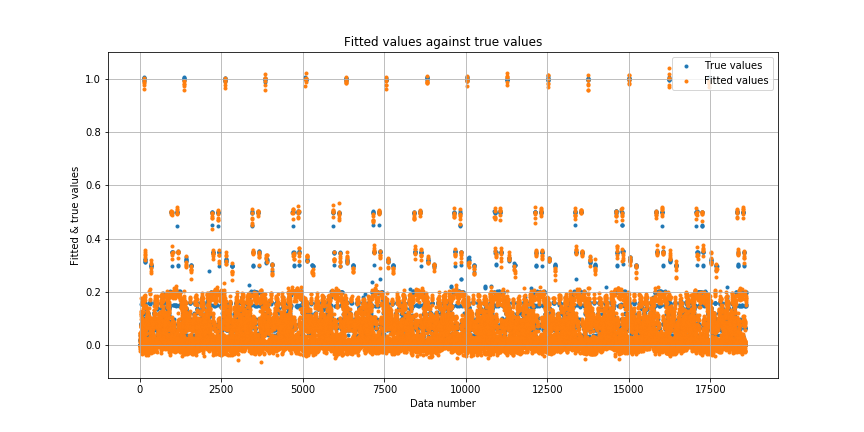


Figure: Fitted Values vs. True Values

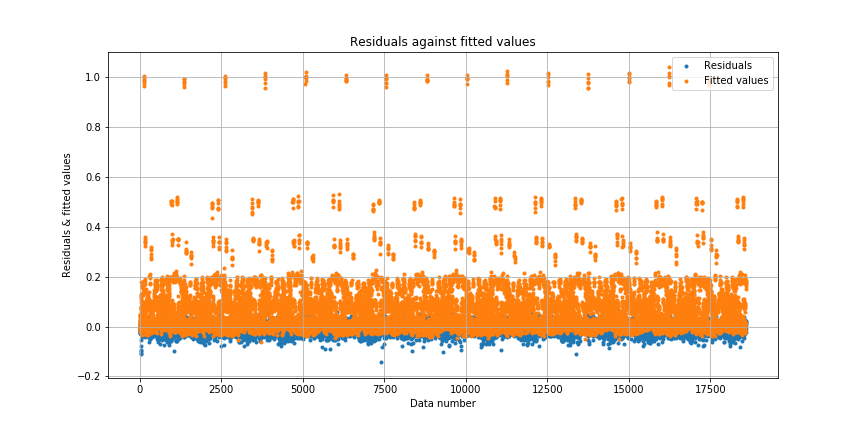


Figure: Residuals vs. Fitted Values

Conclusion: From the figure above, we can observe that the prediction is rather precise. The difference between the fitted values and true values is small even when the true values are very large, and the residuals are very small for all the samples. Neural Network with ReLU activation model could provide a lot of nonlinearity to fit the data, which leads to this relatively good performance. For here, we can see the potential of the Neural Network. Even with only one hidden layer, the regression result is still much better than the linear regression model.

#### **d. Prediction of Backup Size for each Workflow**

i. Linear Regression Model

Plots of fitted values against true values and residuals against fitted values of different workflow are shown below.

The linear regression without isolation of workflows has

train RMSE = 0.103585393643 and Test RMSE = 0.103675847676

After separating the data into workflows we could have a clear comparison. The average RMSE of different workflows decreases obviously, especially the workflow\_3 is as low as 0.007. It is very impressive for a simple linear regression. However, the error of the workflow\_1 increases to 0.148. Probably originally the error caused by workflow\_1 was relative high and was average down by other workflows’ results. However, due to the average of RMSE does drops after separating the workflows, we could conclude it improves the fitting.

**Workflow\_0:**

|  |  |
| --- | --- |
| Train RMSE | 0.03585857176071605 |
| Test RMSE | 0.03586198881981263 |

Table: Train and Test RMSE of workflow\_0 using linear regression



Figure: Fitted Values vs. True Values of workflow\_0 using linear regression

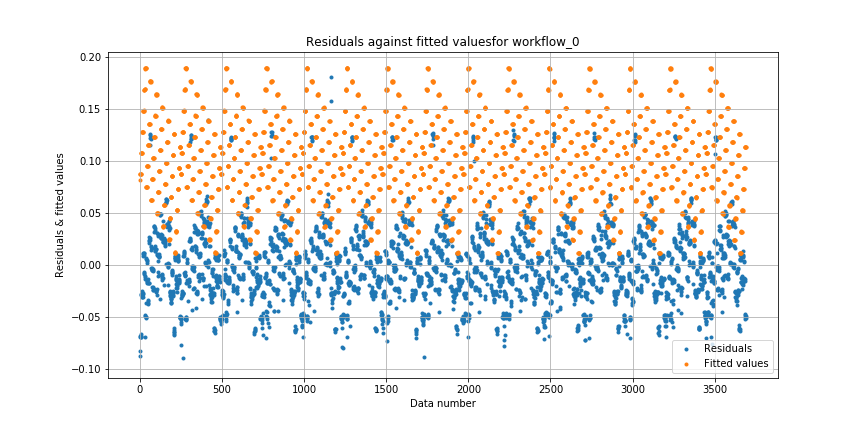


Figure: Residuals vs. Fitted Values of workflow\_0 using linear regression

**Workflow\_1:**

|  |  |
| --- | --- |
| Train RMSE | 0.14875275424136503 |
| Test RMSE | 0.14877231295247528 |

Table: Train and Test RMSE of workflow\_1 using linear regression

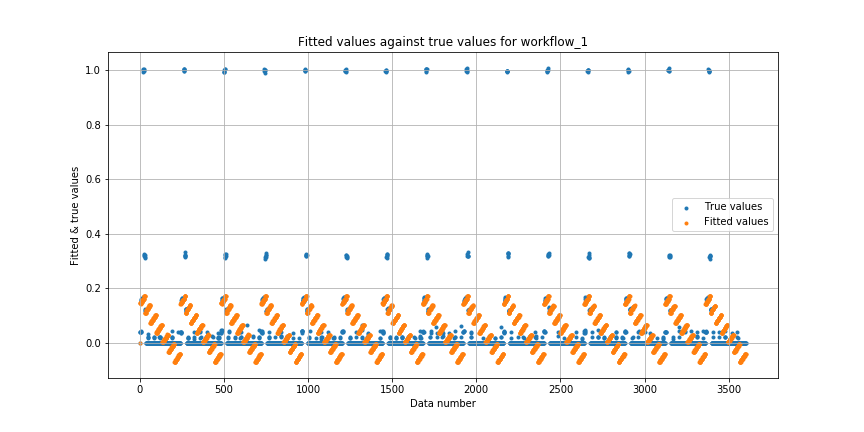


Figure: Fitted Values vs. True Values of workflow\_1 using linear regression

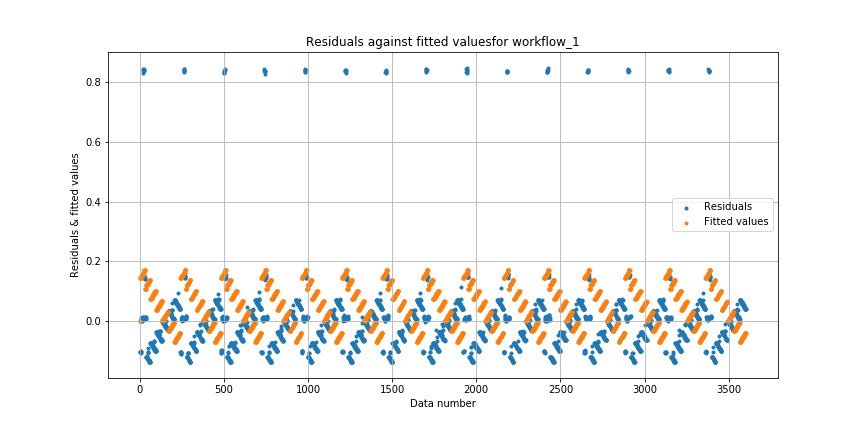


Figure: Residuals vs. Fitted Values of workflow\_1 using linear regression

**Workflow\_2:**

|  |  |
| --- | --- |
| Train RMSE | 0.04290977640154784 |
| Test RMSE | 0.042921141140436825 |

Table: Train and Test RMSE of workflow\_2 using linear regression



Figure: Fitted Values vs. True Values of workflow\_2 using linear regression

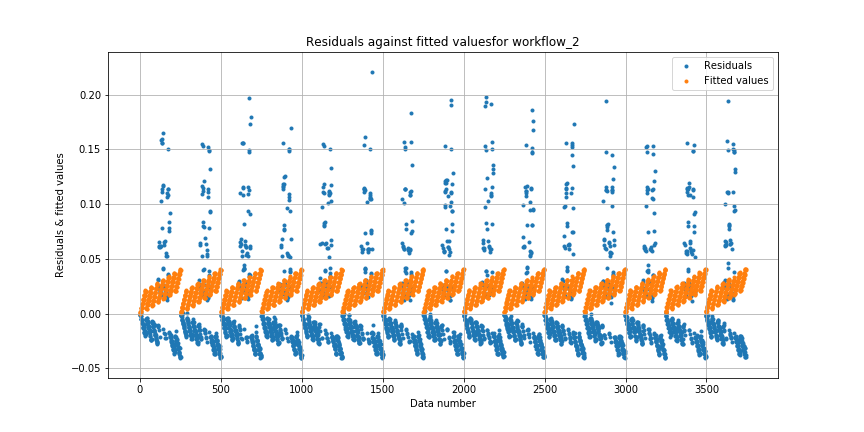


Figure: Residuals vs. Fitted Values of workflow\_2 using linear regression

**Workflow\_3:**

|  |  |
| --- | --- |
| Train RMSE | 0.007243571732649243 |
| Test RMSE | 0.007245350275393028 |

Table: Train and Test RMSE of workflow\_3 using linear regression



Figure: Fitted Values vs. True Values of workflow\_3 using linear regression

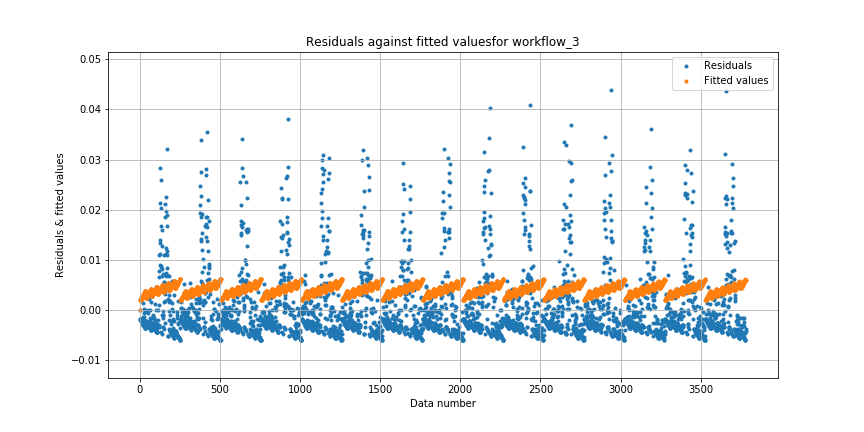


Figure: Residuals vs. Fitted Values of workflow\_3 using linear regression

**Workflow\_4:**

|  |  |
| --- | --- |
| Train RMSE | 0.08591156984175183 |
| Test RMSE | 0.08593821680923319 |

Table: Train and Test RMSE of workflow\_4 using linear regression



Figure: Fitted Values vs. True Values of workflow\_4 using linear regression

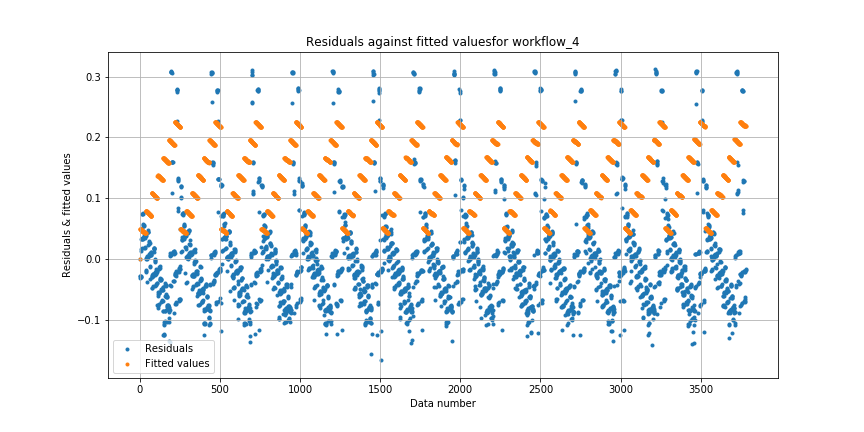


Figure: Residuals vs. Fitted Values of workflow\_4 using linear regression

ii. Polynomial Regression Models with different orders

The plot of test and train RMSE vs polynomial degree for different workflows are shown below:

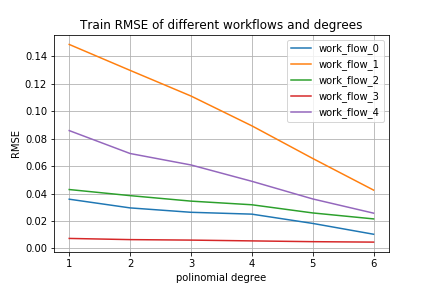


Figure: Train RMSE vs polynomial degrees for different workflows

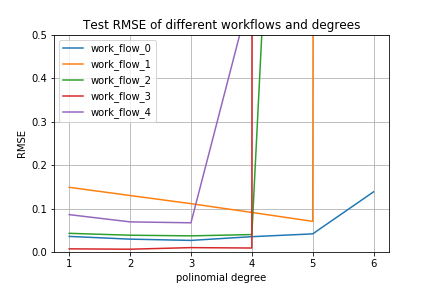


Figure: Test RMSE vs polynomial degrees for different workflows

The best polynomial orders of degrees which result in best minimum test and train RMSE for different workflows are presented below:

|  |  |  |  |
| --- | --- | --- | --- |
| Workflow | Best polynomial degree | Min Train RMSE | Min Test RMSE |
| Workflow\_0 | 3 | 0.010295558081478275 | 0.026673117261153338 |
| Workflow\_1 | 5 | 0.0424956178789225 | 0.0704479992574137 |
| Workflow\_2 | 3 | 0.021485893199284116 | 0.03711667514264777 |
| Workflow\_3 | 2 | 0.0045530456777495175 | 0.006412684458661681 |
| Workflow\_4 | 3 | 0.025680915343315236 | 0.06713983905481843 |

Table: Best polynomial degree and min train RMSE and min test RMSE of different workflows

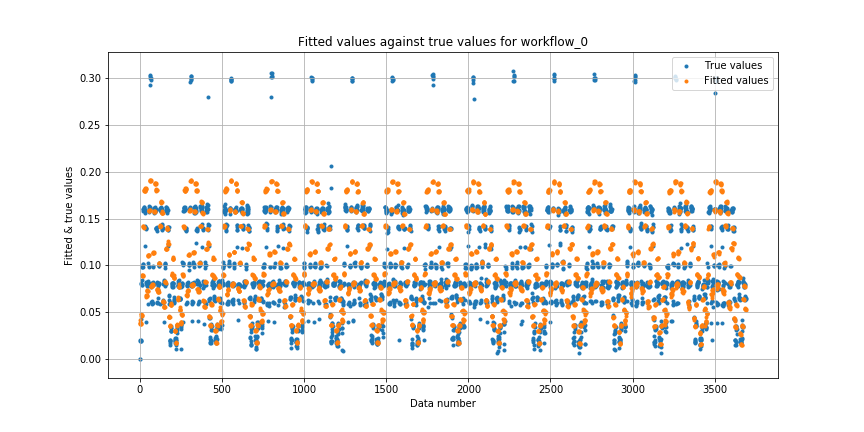


Figure: Fitted Values vs True Values for Workflow\_0

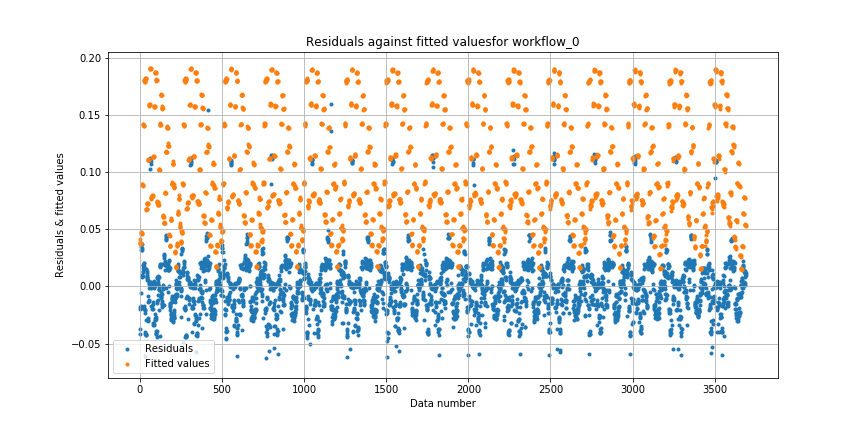


Figure: Residual Values vs Fitted Values for Workflow\_0



Figure: Fitted Values vs True Values for Workflow\_1

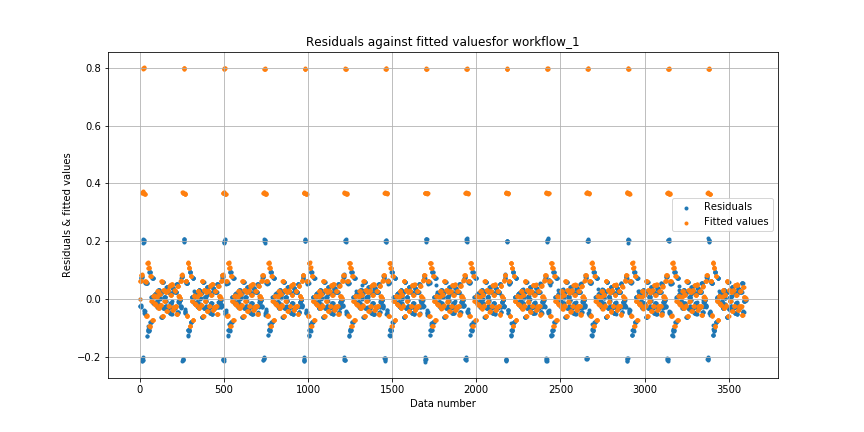


Figure: Residual Values vs Fitted Values for Workflow\_1



Figure: Fitted Values vs True Values for Workflow\_2

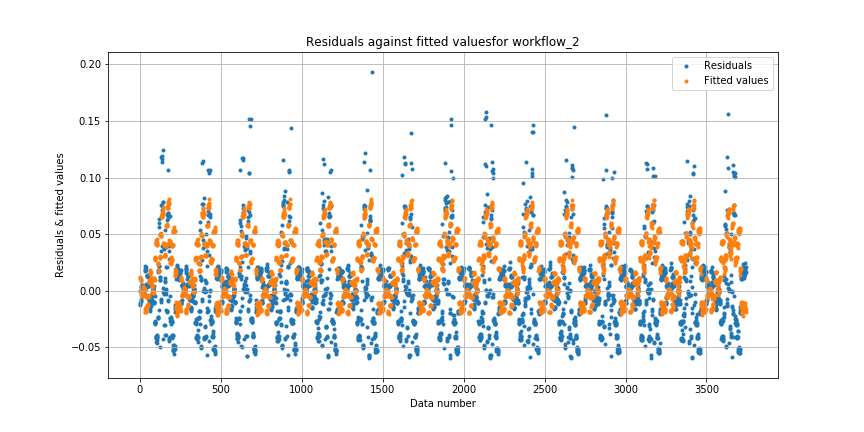


Figure: Residual Values vs Fitted Values for Workflow\_2



Figure: Fitted Values vs True Values for Workflow\_3

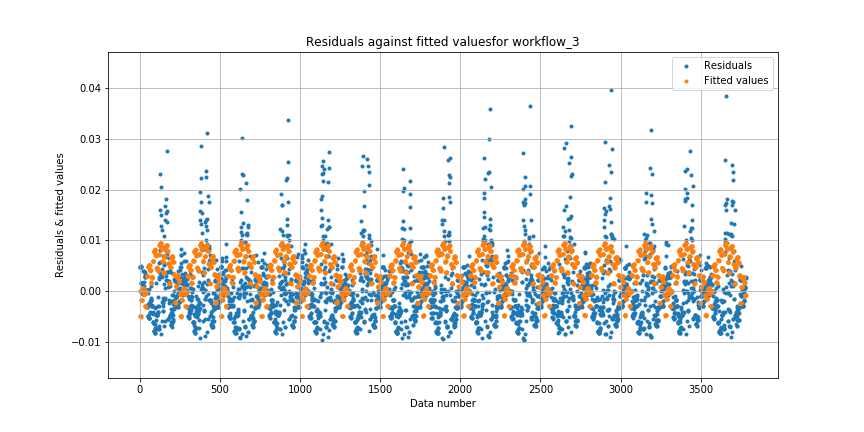


Figure: Residual Values vs Fitted Values for Workflow\_3



Figure: Fitted Values vs True Values for Workflow\_4

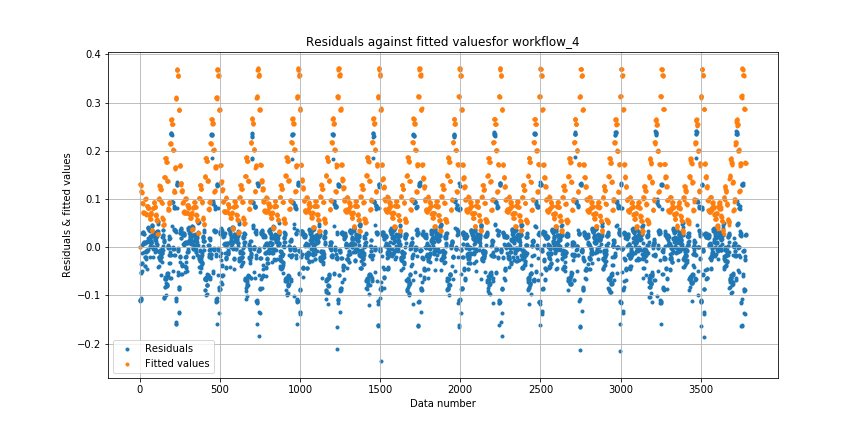


Figure: Residual Values vs Fitted Values for Workflow\_4

Generally speaking, the threshold on the degree of the fitted polynomial is **3**. If the degree of fitted polynomial is larger than 3, the generalization error would get worse.

To be specific on each one of the workflow, the threshold is 3 for workflow\_4; the threshold is 4 for workflow\_2 and workflow\_3; the threshold is 5 for workflow\_0 and workflow\_1:

|  |  |
| --- | --- |
| Workflows | Thresholds |
| workflow\_0 | 5 |
| workflow\_1 | 5 |
| workflow\_2 | 4 |
| workflow\_3 | 4 |
| workflow\_4 | 3 |
| General | 3 |

Table: Thresholds on the degree of the fitted polynomial for different thresholds

How cross validation helps controlling the complexity of the model?

The cross validation process provides a way for us to tune the hyperparameters in our models. As shown in the figures above, the train RMSE always decreases when the model complexity is increased, since complex models can utilize more training parameters to fit the training set. This may leads to the overfitting problem. By introducing the cross validation, we can evaluate the performance of our model in a set which is not used for training. We can thus control the model complexity by analyzing the performance on these “clean” data to avoid overfitting.

#### **e. K-nearest Neighbor Regression**

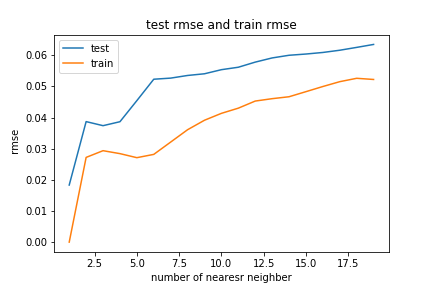


Figure: Test and Train RMSE vs Number of Nearest Neighbor with scalar encoding

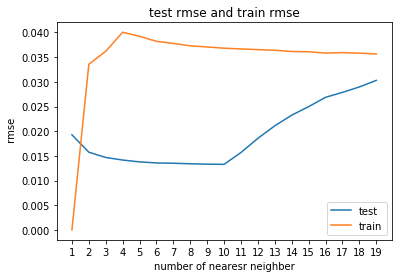


Figure: Test and Train RMSE vs Number of Nearest Neighbor with one hot encoding

Generally speaking, KNN regression would have better result when using fewer nearest neighbor. As we could tell from the plot, test and train RMSE is lower when the number of nearest neighbor is lower, which is

RMSE\_scalarcoding = 0.018331464302376928

RMSE\_onehotcoding = 0.01385914869123501

For KNN regression, the one hot encoding method looks a little bit better in training although its train and test result looks pretty strange because test error is even smaller than training result.

The corresponding results are presented below:

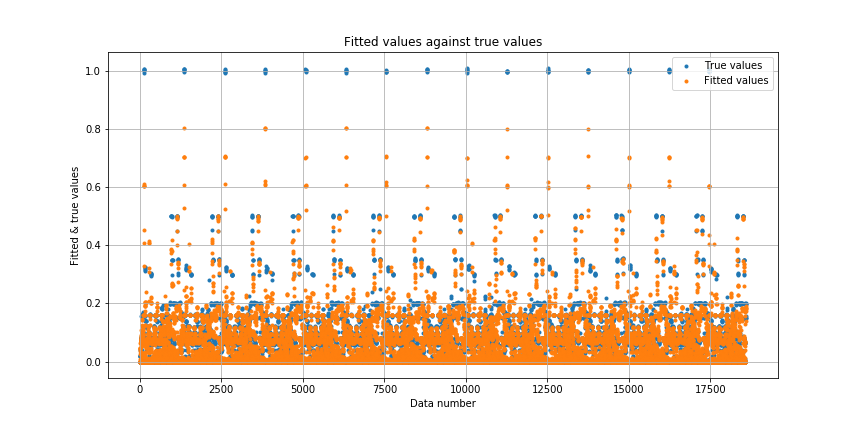


Figure: Fitted Values vs True Values

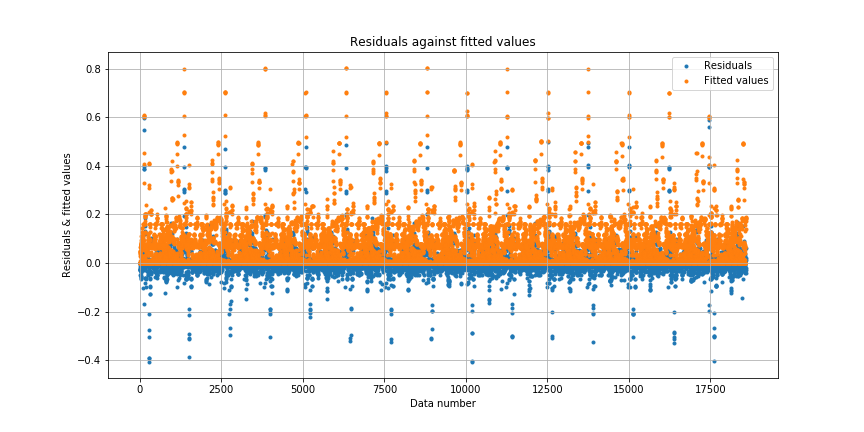


Figure: Residuals against Fitted Values

## Question 3: Comparison of Regression

|  |  |  |
| --- | --- | --- |
|  | One hot Encoding (sparse) | Scalar Encoding |
| Linear Regression | Test: 0.0885042609364  Train: 0.0883374486188 | Test: 0.103675847676  Train: 0.103585393643 |
| Ridge | Test: 0.0885047267501  Train: 0.0883374486188 | None |
| Lasso | Test: 0.0885082492502  Train: 0.0883429584855 | None |
| Elastic | Test: 0.089110136077  Train: 0.0889641150629 | None |
| Random Forest | None | Test: 0.0113901185016  Train: 0.0136852107899 |
| Neural Network | Test: 0.0343930517408  Train:0.0191107607826 | None |
| Polynomial(separate workflow) | None | Test: 0.04155806  Train: 0.02090221 |
| KNN | Test: 0.0138591486912350 | Test: 0.01833146430237 |

From all the experiments above tuning hyperparameters of each model to its best, the random forest regression model generates the best result for the smallest RMSE. Random Forest Regression Model actually handles the categorical features really well as it does not require either scalar or one-hot encoding. This advantage could prevent information loss during the encoding process.

For scalar encoding, KNN behaves the best compared to polynomial and linear regression, but it takes much more time to test than the other two models.

For one hot encoding, all the linear models perform similar no matter what the regularization. KNN is also the best model to predict the result compared to NN and linear models. However, NN’s performance could probably improve in further with deeper structure and finer tuning.

For all the model we deal with in this project, linear models (including Ridges, Lasso and Elastic Net) is great when the relationship is “linear” related which will over simplifies many real world problems. Therefore underfitting happens a lot and regularization in Ridges, Lasso and Elastic Net are trivial to improve the performance. Polynomial models have the advantage similar to a linear model. It is computationally friendly and easy to interpret however it works poorly with interpolatory and asymptotic properties

KNN is the one of the most straightforward model we tried in this project. It is easy to implement and sometimes has great effect on multi-variables regression. However, it takes long time to do every prediction and hard to deal with extreme huge dataset in the real world. What’s more the “meaningful” distance function for KNN is also very subtle to find.

Ensemble decision tree model like random forest could effectively reduce the variance of result and handle any type of input and evaluate the importance of each models however due to its nature, it is comparatively hard to interpret the result visually.

Neural Network works like a black box. It works exceptionally good sometimes with fine tuning. However it also suggests its disadvantage. It takes great time, memory and computation to get a set of decent hyper parameters. What’s more people cannot interpret the prediction process of neural network intuitively.