**EE219 Project 1**

Regression Analysis

Winter 2017

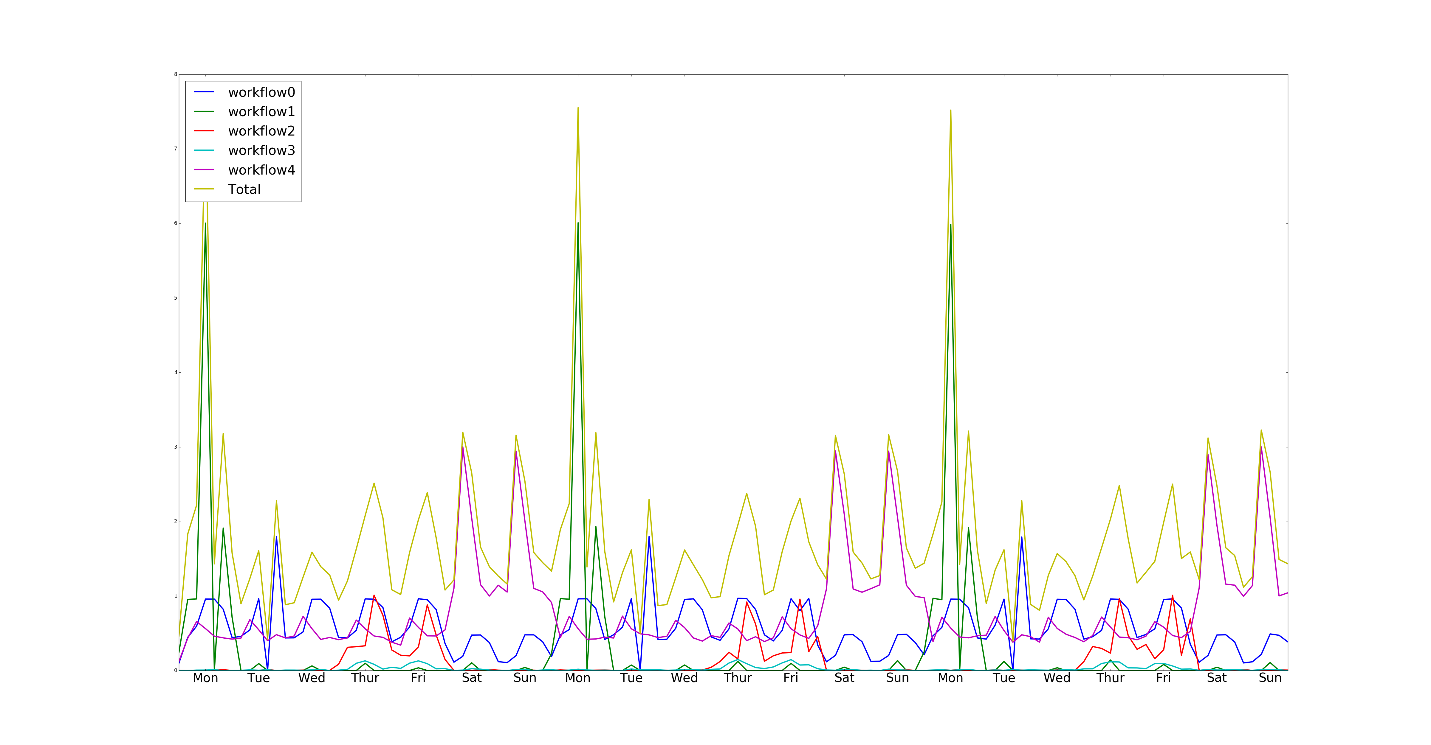
Mojtaba Sahraee 204517437

Kia Karbasi 804617014

Heng Zhao 304436191

1. Plot of the size of the backup vs other features

Size of Backup vs. Day for each workflow and all the workflows:



We plotted the size of backup for 3 weeks and obviously there is periodicity in the plot in which the period is a week. And it can be seen that the backup size has been the highest in Mondays and we also have relatively large backup size on Saturday and Sunday as well.

1. Modeling the Network Backup Dataset
2. Linear Regression

The linear regression minimizes

where is the output vector and is the matrix of feature vectors with each row being the feature vector of one of the input points.

Using the raw numbers in the file as the features we get the following coefficients for each of the features:

Week: 0.000 Day of the week: 0.001 Backup start time: 0.001 Work flow: 0.003 File number: -0.000 Backup Time: 0.071

Therefore, almost the only feature that has been used in the linear regression is the backup time.

In this regression the mean squared error on the training data is 0.07952 and the RMSE for 10 fold cross validation is 0.07949.

Now using StatsModel Package in Python we analyze the significance of each variable. The result of the model is as follows:

OLS Regression Results

==============================================================================

Dep. Variable: y R-squared: 0.417

Model: OLS Adj. R-squared: 0.417

Method: Least Squares F-statistic: 2216.

Date: Wed, 01 Feb 2017 Prob (F-statistic): 0.00

Time: 15:39:02 Log-Likelihood: 20679.

No. Observations: 18588 AIC: -4.134e+04

Df Residuals: 18581 BIC: -4.129e+04

Df Model: 6

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [95.0% Conf. Int.]

------------------------------------------------------------------------------

Week -1.729e-05 0.000 -0.128 0.898 -0.000 0.000

Day 0.0010 0.000 3.253 0.001 0.000 0.002

Start Time 0.0009 8.55e-05 10.616 0.000 0.001 0.001

Work Flow 0.0026 0.000 6.370 0.000 0.002 0.003

File ID 0 0 nan nan 0 0

Backup Time 0.0699 0.001 111.675 0.000 0.069 0.071

const -0.0216 0.002 -10.426 0.000 -0.026 -0.018

==============================================================================

Omnibus: 17815.803 Durbin-Watson: 0.643

Prob(Omnibus): 0.000 Jarque-Bera (JB): 1044097.871

Skew: 4.614 Prob(JB): 0.00

Kurtosis: 38.538 Cond. No. 92.9

The main parameters of the model as can be seen are the T parameter and the P parameter. The t-value measures the size of the difference relative to the variation in your sample data. Put another way, t is simply the calculated difference represented in units of standard error. The greater the magnitude of t (it can be either positive or negative), the greater the evidence against the null hypothesis that there is no significant difference. The closer t is to 0, the more likely there isn't a significant difference. On the other hand, p-value is a metric of rejecting the null hypothesis and concludes that there is a statically significant difference. So, the parameters with high t value and low p values are significant in the model. It can be seen that the most important feature is the backup time (which is consistent with our intuition that backup size must be proportional to the backup time (at least if time was not so roughly quantized)). Also, day and start time and work flow are all significant whereas week and fileID are not. Again, the fact that week is not significant is consistent with our intuition that the backup size is almost periodic with period of a week and thus week must not be important in estimating the backup size.

But since the features like file number, day of the week, are categorical variables, converting them to numbers (let’s say 0 to 6 for days of the week) and using them directly as features for linear regression does not make sense. This is because linear regression implies that if we change a variable by some amount, the output would change proportional to the coefficient corresponding to that feature. But Even though the output is dependent on the day of the week, it is not linearly so. Therefore, another type of encoding is commonly used for encoding categorical variables known as onehot encoding. In this encoding instead of encoding days of the week as one number from 0 to 6, we encode it as a 7-vector with all zeros and only one 1 at the place corresponding to the day of the week (thus the name onehot). Therefor a categorical variable that can be one of the n categories, is encode into an n-vector.

If we use this feature vector and use the least square minimization, the entries of vector of coefficients, , would become very large, indicating that the features in are not independent and there have been numerical problems in calculating . (this happen because in calculating the coefficients we must find the pseudo inverse of the feature matrix and if some of the features which requires the inverse of the cross-correlation matrix of the features. This matrix can be singular if some features are not linearly independent). Therefore, we need to use a regularizer to make the solution well-defined. In this problem, we use Lasso regression which adds a penalty that makes the features used in regression sparse, thus not using the features that are not important. Doing so we will get the following RMSE from 10-fold corss-validation:

which is a 16% improvement over the previous case.

Here’s the summary of the result of this model:

OLS Regression Results

==============================================================================

Dep. Variable: y R-squared: 0.538

Model: OLS Adj. R-squared: 0.536

Method: Least Squares F-statistic: 371.6

Date: Wed, 28 Jan 2017 Prob (F-statistic): 0.00

Time: 19:54:32 Log-Likelihood: 22833.

No. Observations: 18588 AIC: -4.555e+04

Df Residuals: 18529 BIC: -4.509e+04

Df Model: 58

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [95.0% Conf. Int.]

------------------------------------------------------------------------------

const 0.0782 0.001 57.628 0.000 0.076 0.081

x1 0 0 nan nan 0 0

x2 0 0 nan nan 0 0

x3 0 0 nan nan 0 0

x4 0 0 nan nan 0 0

x5 0 0 nan nan 0 0

x6 0 0 nan nan 0 0

x7 0 0 nan nan 0 0

x8 0 0 nan nan 0 0

x9 0 0 nan nan 0 0

x10 0 0 nan nan 0 0

x11 0 0 nan nan 0 0

x12 0 0 nan nan 0 0

x13 0 0 nan nan 0 0

x14 0 0 nan nan 0 0

x15 0 0 nan nan 0 0

x16 0.0284 0.002 16.650 0.000 0.025 0.032

x17 -0.0210 0.002 -11.282 0.000 -0.025 -0.017

x18 0 0 nan nan 0 0

x19 0 0 nan nan 0 0

x20 -0.0195 0.002 -10.591 0.000 -0.023 -0.016

x21 0.0107 0.002 6.356 0.000 0.007 0.014

x22 0.0101 0.002 6.028 0.000 0.007 0.013

x23 -0.0047 0.002 -2.574 0.010 -0.008 -0.001

x24 -0.0006 0.002 -0.331 0.741 -0.004 0.003

x25 0.0116 0.002 6.378 0.000 0.008 0.015

x26 0.0190 0.002 10.420 0.000 0.015 0.023

x27 0 0 nan nan 0 0

x28 0.0065 0.002 3.554 0.000 0.003 0.010

x29 0.0215 0.001 17.724 0.000 0.019 0.024

x30 0.0144 0.001 11.270 0.000 0.012 0.017

x31 -0.0012 0.001 -1.061 0.289 -0.003 0.001

x32 -0.0097 0.001 -8.295 0.000 -0.012 -0.007

x33 0.0532 0.001 43.184 0.000 0.051 0.056

x34 0 0 nan nan 0 0

x35 0 0 nan nan 0 0

x36 0 0 nan nan 0 0

x37 0 0 nan nan 0 0

x38 0 0 nan nan 0 0

x39 0 0 nan nan 0 0

x40 0 0 nan nan 0 0

x41 0 0 nan nan 0 0

x42 0 0 nan nan 0 0

x43 0 0 nan nan 0 0

x44 0 0 nan nan 0 0

x45 0 0 nan nan 0 0

x46 0 0 nan nan 0 0

x47 0 0 nan nan 0 0

x48 0 0 nan nan 0 0

x49 0 0 nan nan 0 0

x50 0 0 nan nan 0 0

x51 0 0 nan nan 0 0

x52 0 0 nan nan 0 0

x53 0 0 nan nan 0 0

x54 0 0 nan nan 0 0

x55 0 0 nan nan 0 0

x56 0 0 nan nan 0 0

x57 0 0 nan nan 0 0

x58 0 0 nan nan 0 0

x59 0 0 nan nan 0 0

x60 0 0 nan nan 0 0

x61 0 0 nan nan 0 0

x62 0 0 nan nan 0 0

x63 0 0 nan nan 0 0

x64 -0.0824 0.001 -58.286 0.000 -0.085 -0.080

x65 -0.0479 0.001 -39.294 0.000 -0.050 -0.045

x66 0.0377 0.001 25.187 0.000 0.035 0.041

x67 0.0154 0.002 6.629 0.000 0.011 0.020

x68 0.1553 0.004 43.941 0.000 0.148 0.162

==============================================================================

Omnibus: 20446.728 Durbin-Watson: 0.715

Prob(Omnibus): 0.000 Jarque-Bera (JB): 1880270.095

Skew: 5.658 Prob(JB): 0.00

Kurtosis: 50.955 Cond. No. nan

X1 to x15 correspond to weekdays and x34 to x63 correspond to file IDs which are again not used in the model.

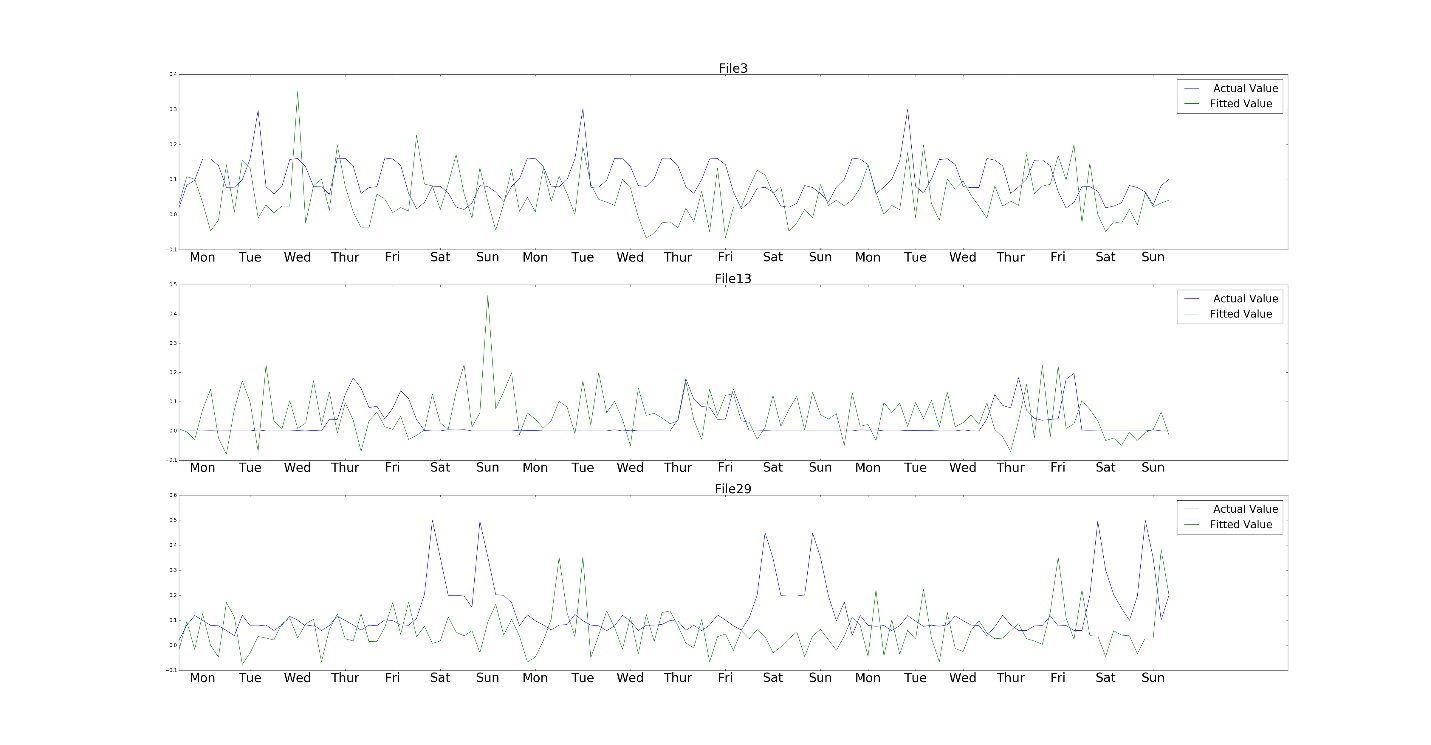


Figure 0- Fitted value vs Actual overtime for linear regression

It is clear from the graph above that estimation error is time dependent.

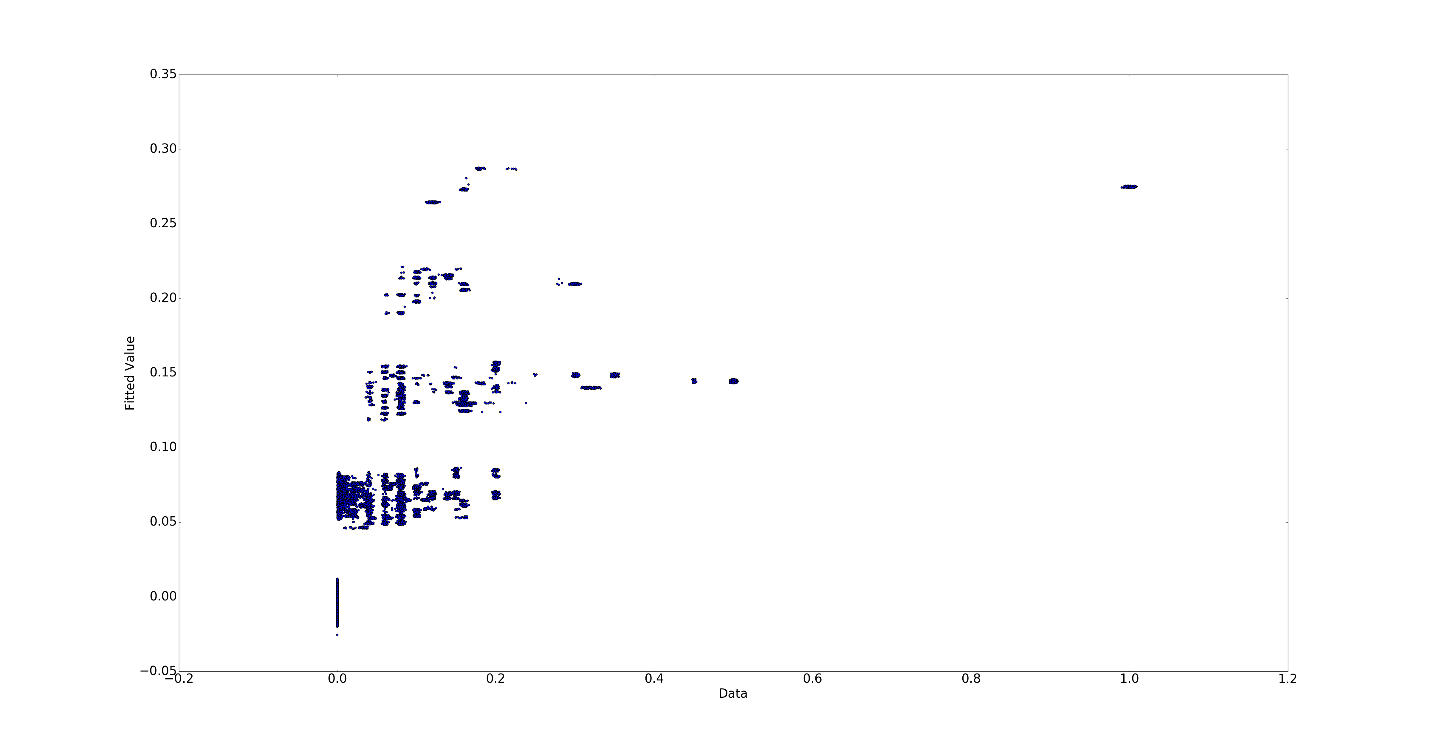
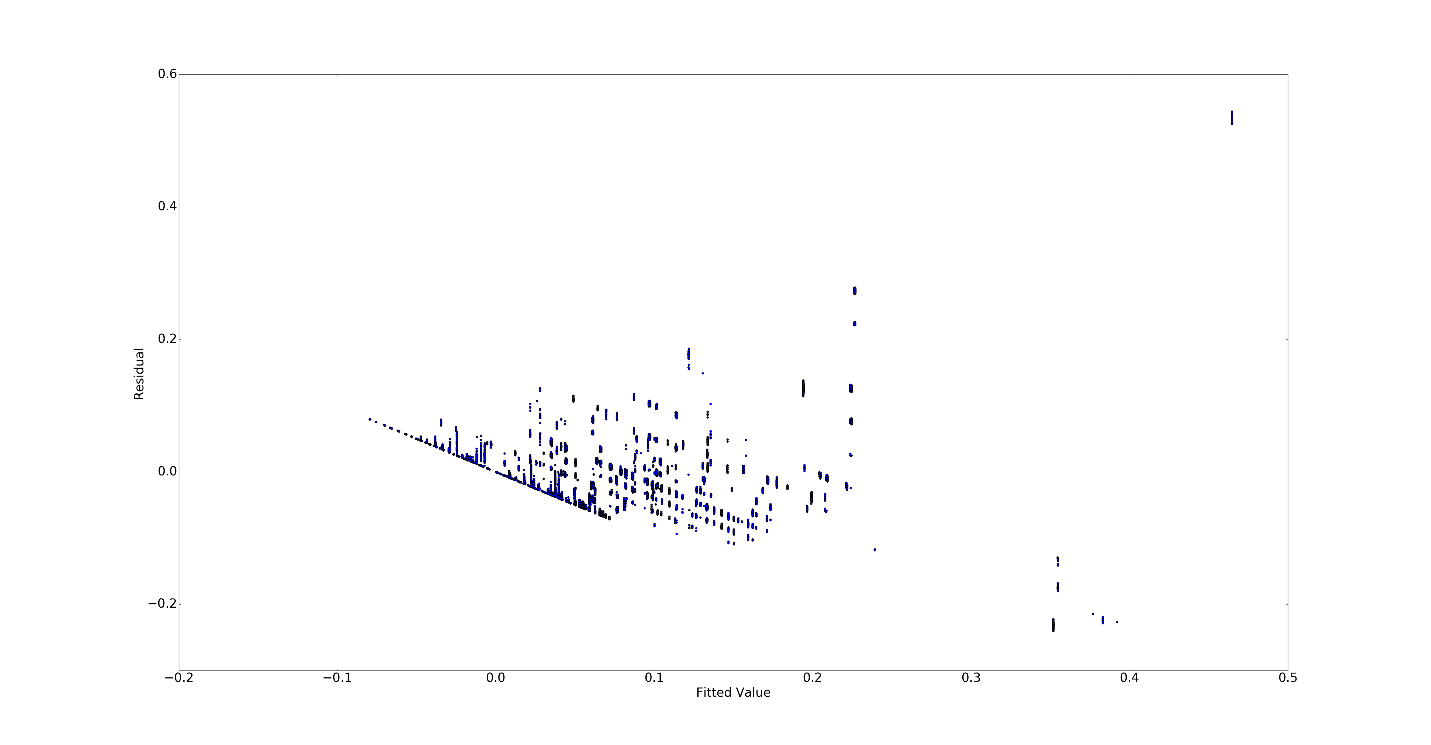


Figure - Fitted value vs Actual for Linear regression

As can be seen the results are fitted on a linear regression line and the next plot shows the residual which gives us better idea how far the predictions are from the actual values:

Figure -Residual vs Fitted value for linear regression

The residual for most of the data points are close to zero which shows that the linear model efficiently predicts the “size of the backup”. However, linear regression might not be the best fit for this dataset.

b) Random Forest:

Using the default parameters provided in the assignment, we know implement the Random Forest method and plot the “fitted vs actual” values for these parameters. We then go ahead and play with these parameter values to observe the changes they make to our predictions. Plots and average RMSE for Random Forest method with different values for the number of trees and depth can be found below.

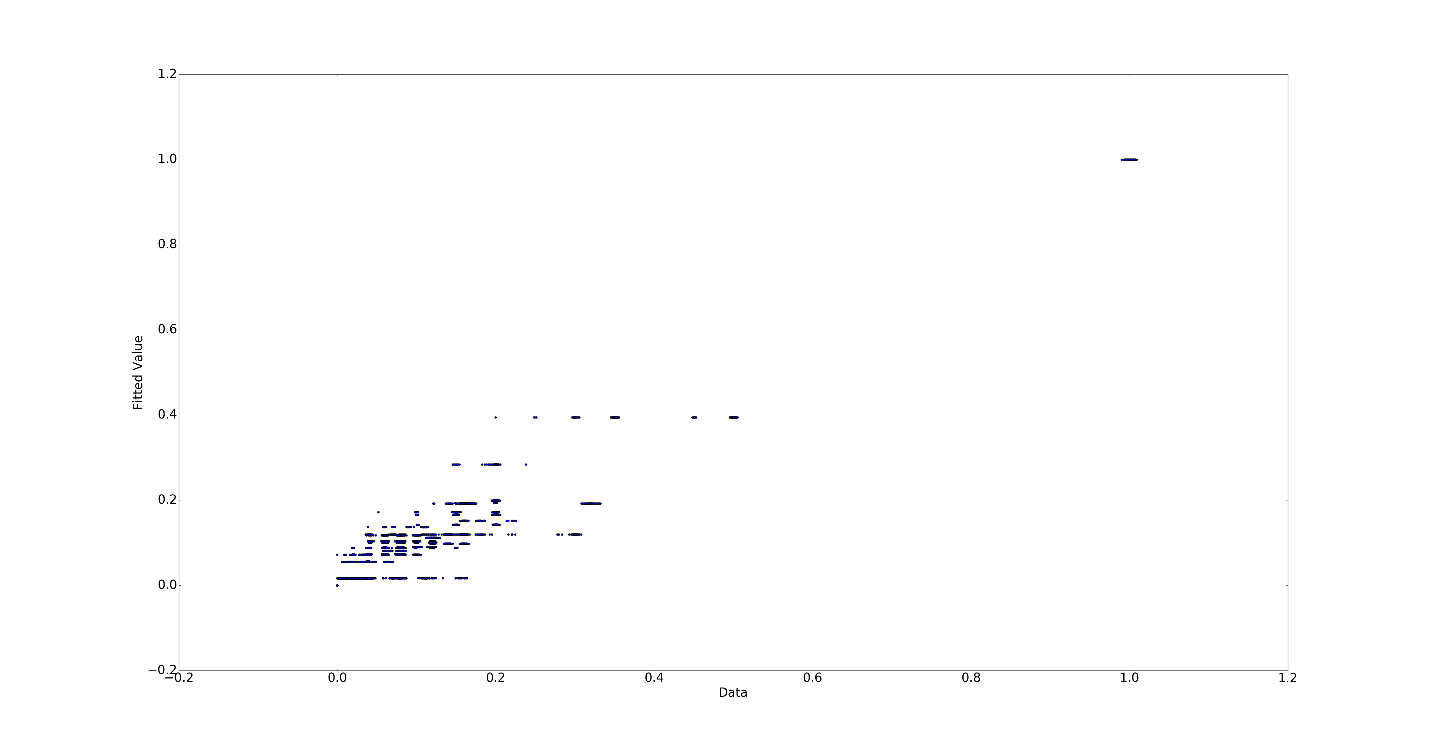


Figure -Fitted value vs Actual for Random Forest ntree=20 depth=4

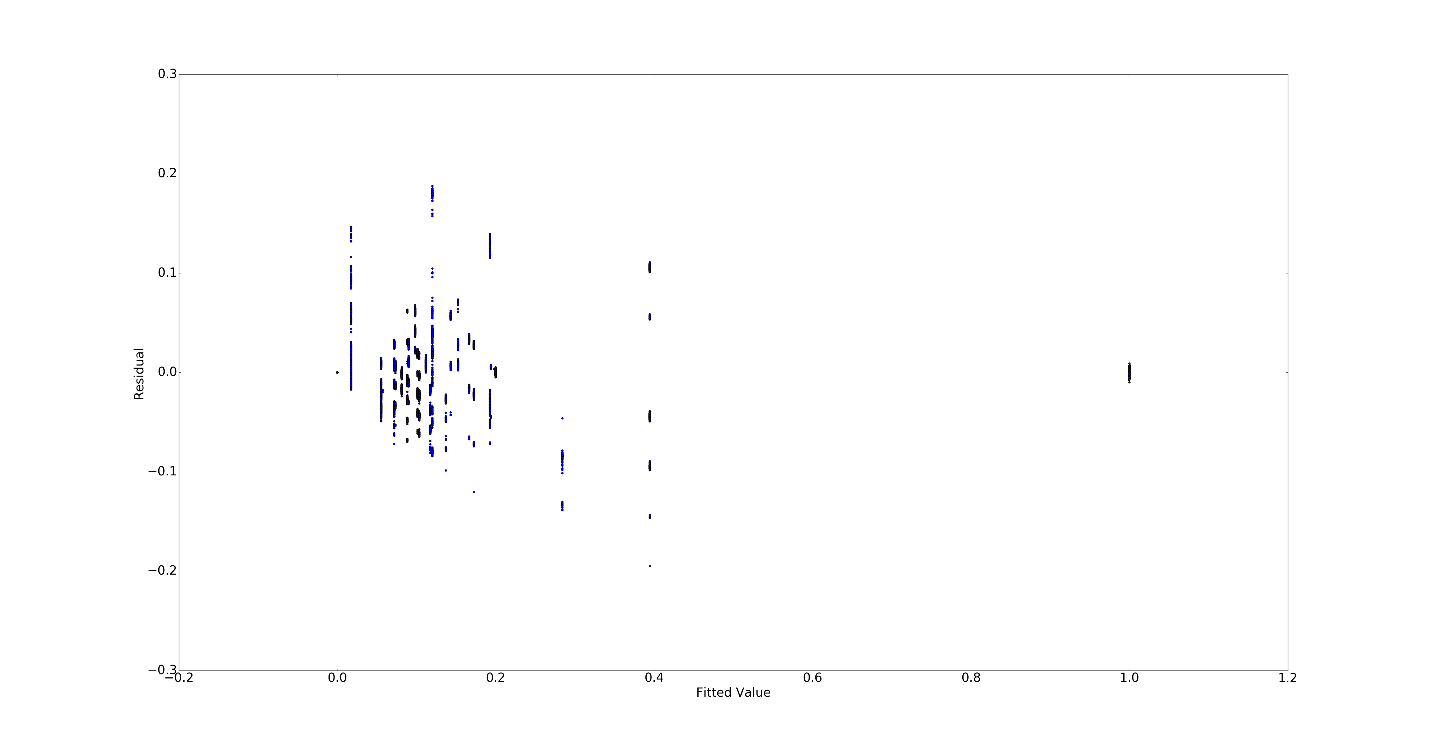


Figure - Residual vs Fitted value for Random Forest ntree=20 depth=4

For the default values, the RMSE found to be :

Training MSE =0.0295413394014

Test MSE mean (**RMSE**) = 0.0296965905821526

Test\_MSE std = 0.007865175

Now we tune the parameters to see if we can obtain better results:

Tree number= 60, Tree depth= 4 RMSE= 0.0298891368

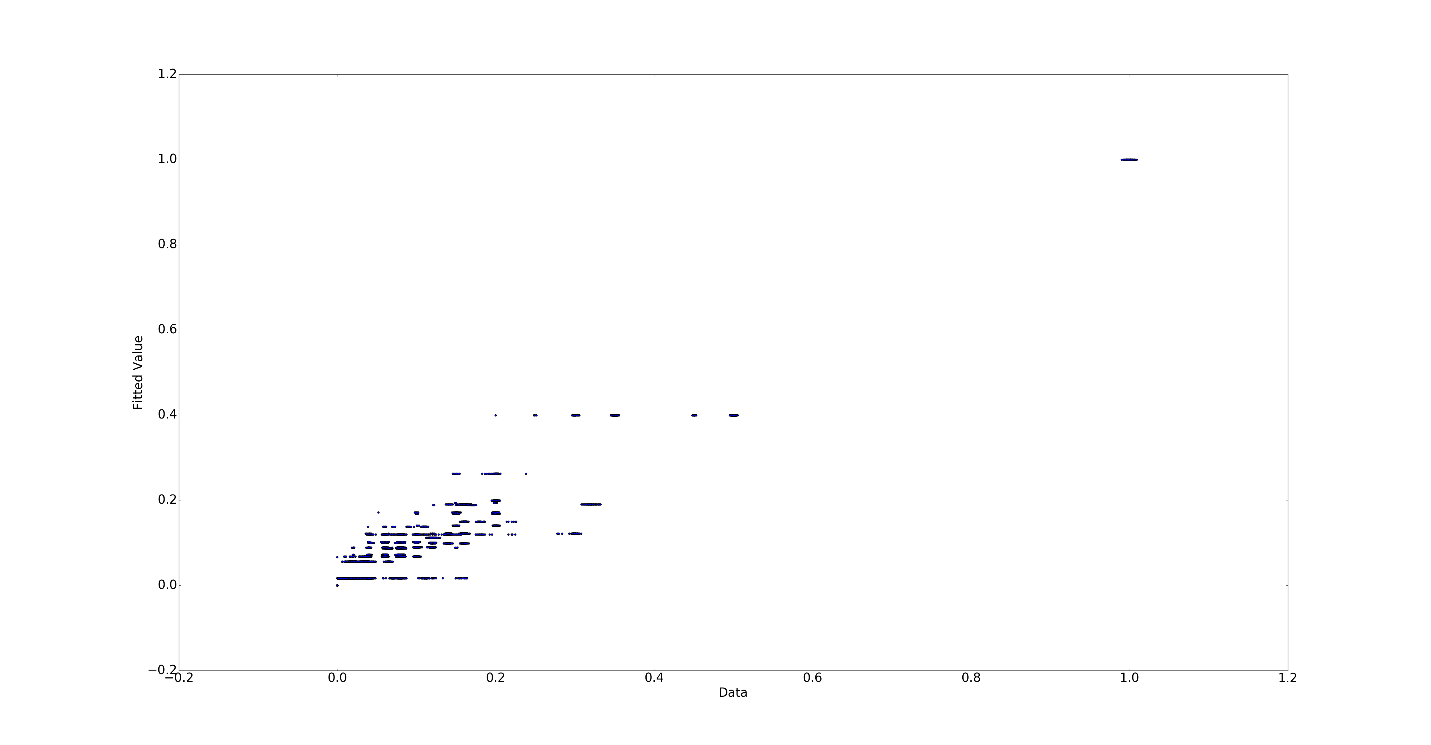


Figure -Fitted value vs Actual for Random Forest ntree=60 depth=4

Tree number= 40 , Tree depth= 10 RMSE= 0.00934777805

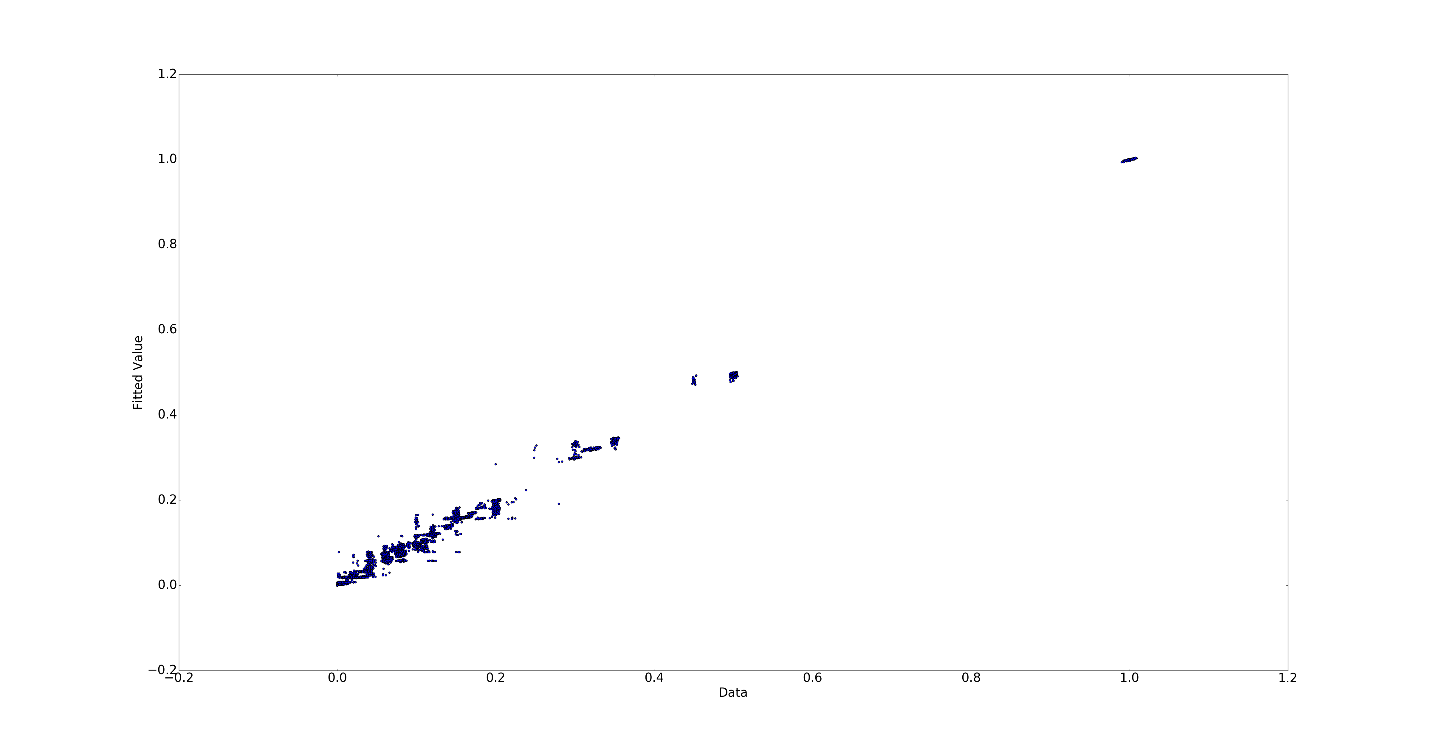


Figure -Fitted value vs Actual for Random Forest ntree=40 depth=10

Tree number= 40 , Tree depth= 12 RMSE= 0.00960552584

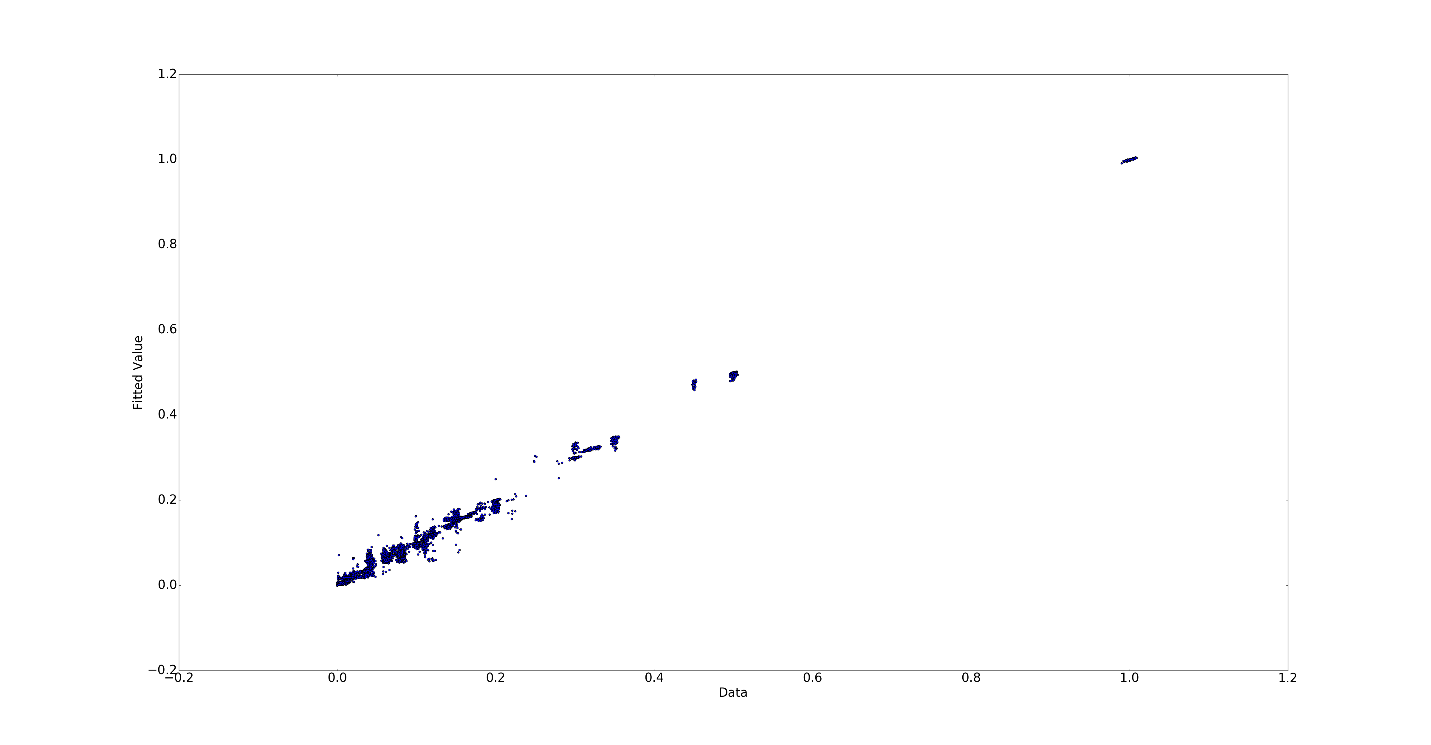


Figure -Fitted value vs Actual for Random Forest ntree=40 depth=12

Tree number= 100 , Tree depth= 10 RMSE= 0.00932060402

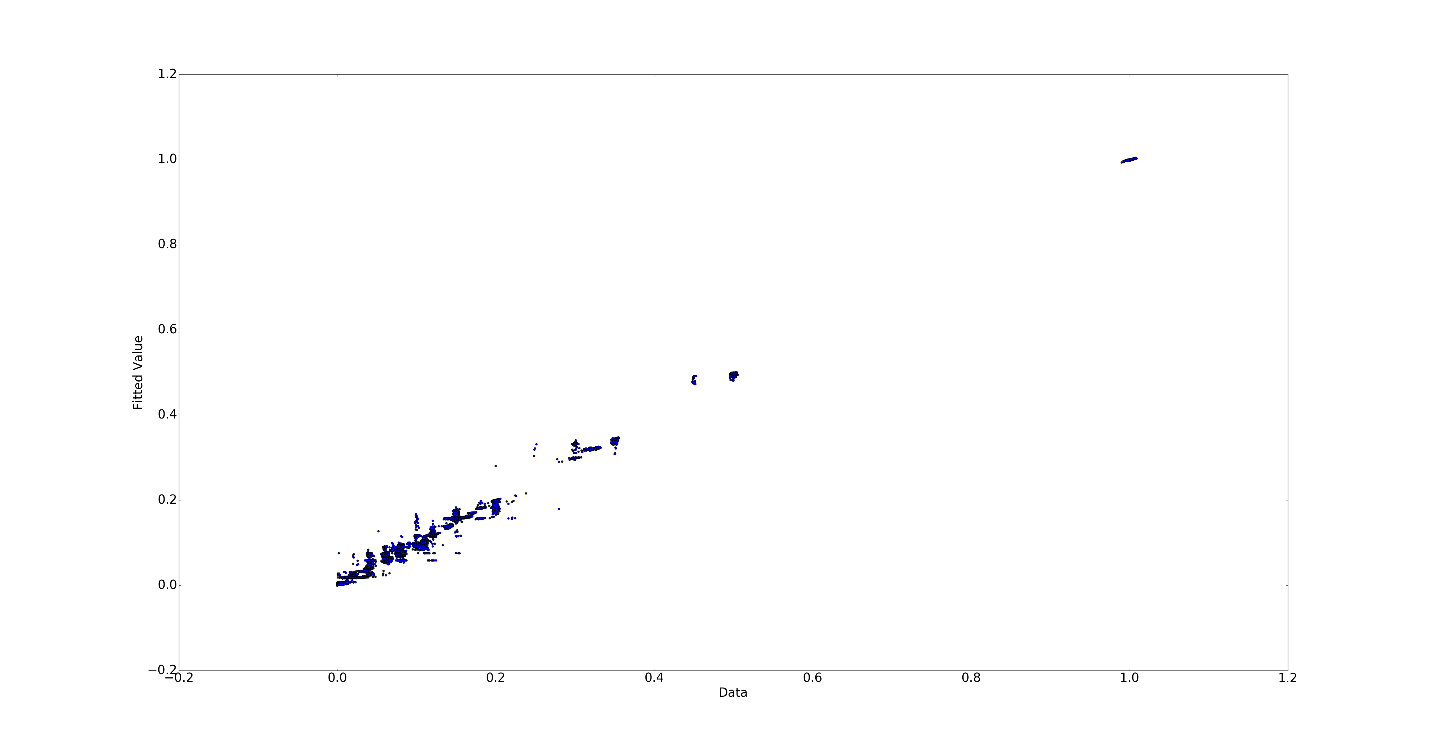


Figure 7-Fitted value vs Actual for Random Forest ntree=100 depth=10

So increasing number of trees and more depth will help us get better result and less RMSE. This is expected because more trees help diminishing returns and reduces the variance. Also deeper trees reduces the bias. However, too much depth will result in overfitting because doing so in reality increases the number of parameters while the number of data points stays constant. Moreover, the computation becomes expensive as we increase these parameters. Using a utilitarian approach, we can assume that the following result is practically optimum:

Tree number= 40 , Tree depth= 10 RMSE= 0.00934777805

c) Neural Network:

For neural networks, the major parameter is the number of hidden layers. Also, for weight optimization, number of iterations are important parameters.

To fix the test result, we set the randon\_state to 1 for all test results. We discover several algorithms and their performance with different number of hidden layers. We test the RMSE with different hidden layer sizes, and we plotted the following:

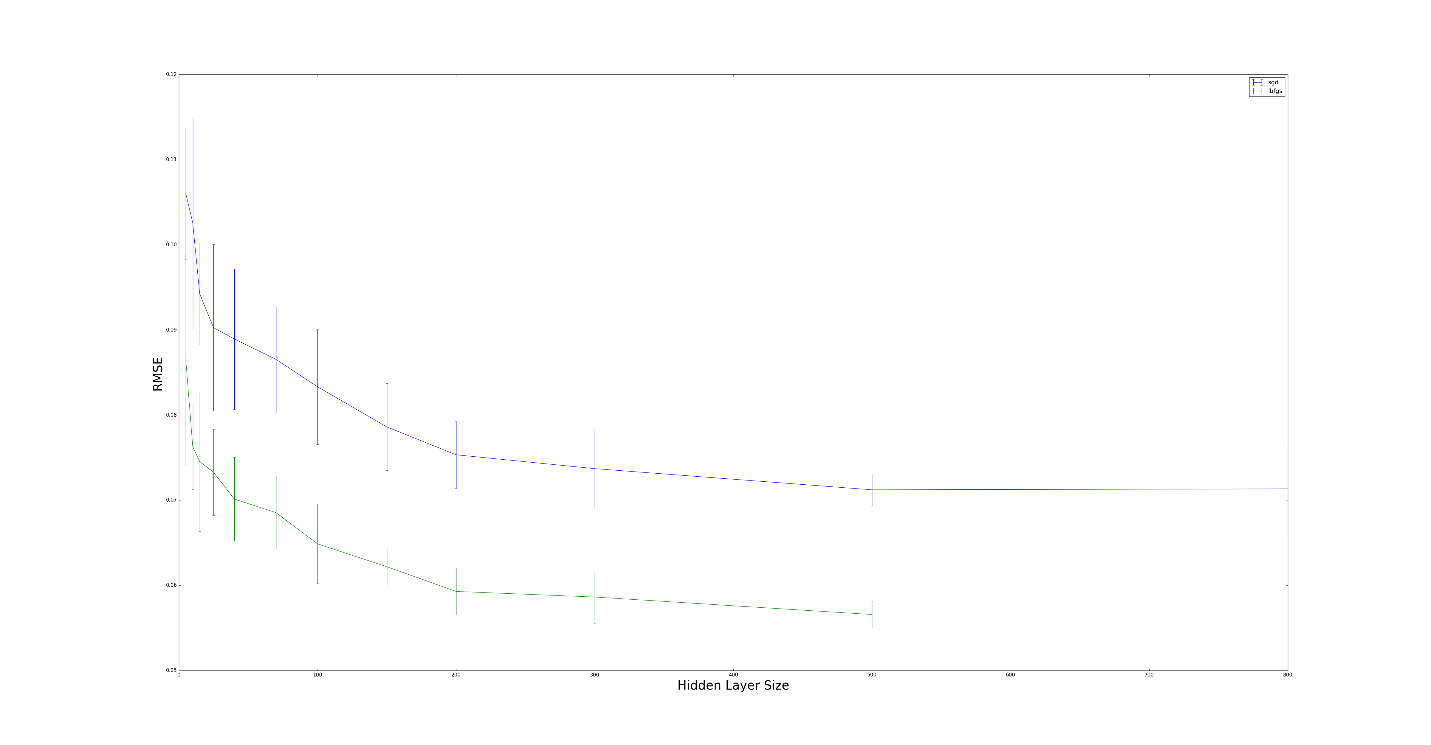


Figure 8- RMSE of SGD and L-BFGS vs Hidden Layer Size

We realized that for size of the hidden greater than 100, the RMSE flattens. Also, the L-BFGS is much more computationally expensive since it took much longer to plot the L-BFGS portion of the above graph than the SGD. However, the L-BFGS provides about %30 less RMSE. For that reason, the L-BFGS is recommended for smaller data sets while the SGD works better and quicker for larger data sets.



First, we calculated the **RMSE** for the different workflows using linear regression. The following graph was plotted:

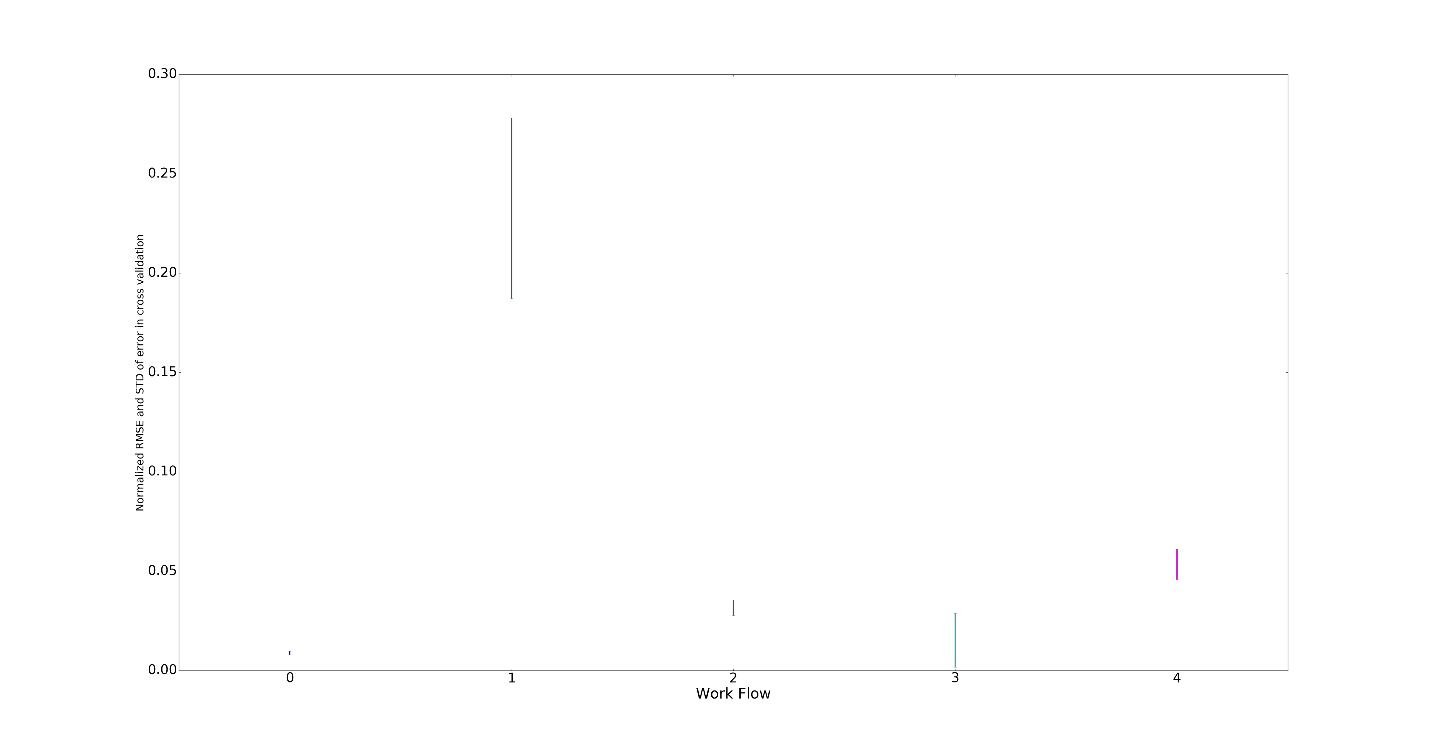
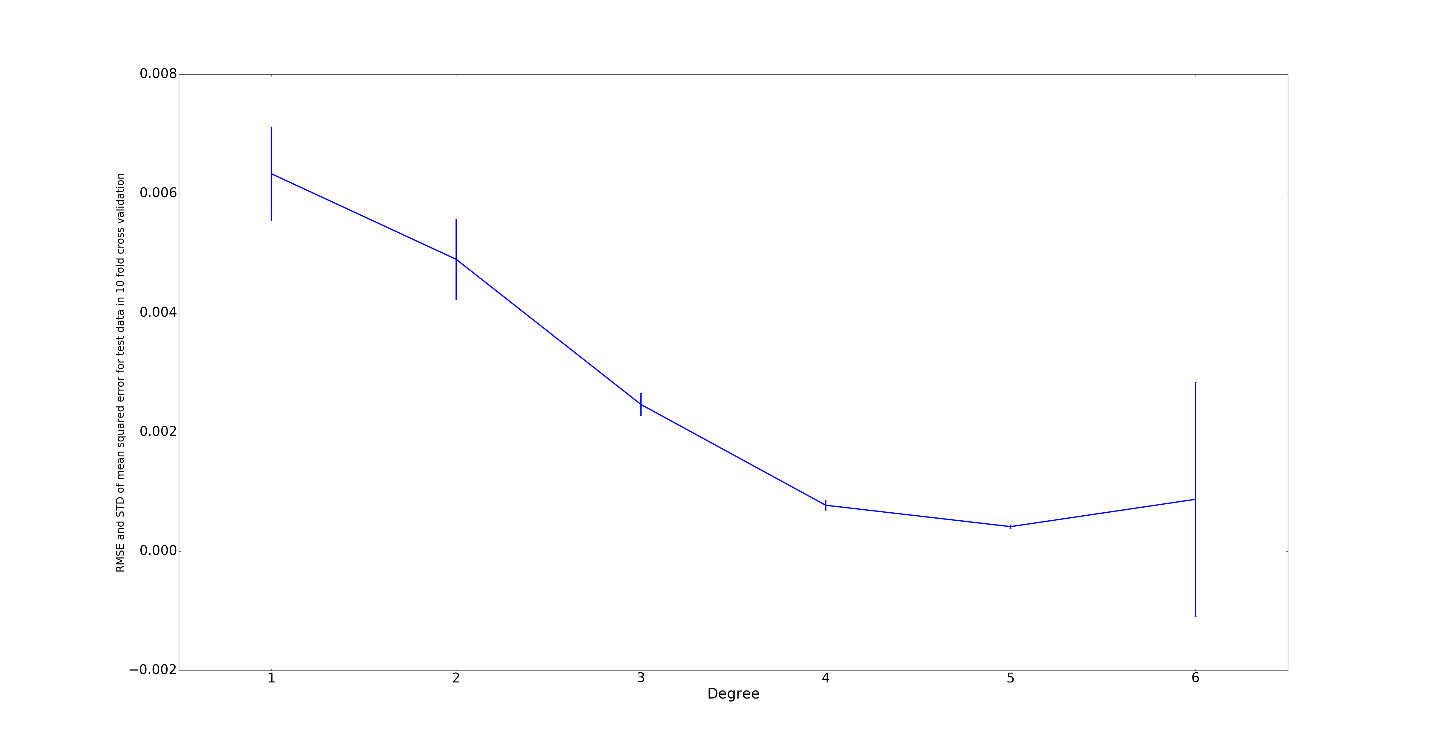


Figure 9- RMSE vs Workflow

Each **RMSE** has been normalized. The length of the bars indicate the standard deviation of the **RMSE** for each workflow. We can see that predicting workflow\_0 has the smallest “overall” error and thus best prediction since the error is almost always small and mostly the same. We can also see that for workflow\_3, the error is also almost always small while it could deviate to zero or 0.05. We can also see that our prediction of file sizes for workflow\_1 would be the worst. Next we try to use polynomial fits for our prediction model. We calculated the RMSE using a 10-fold cross validation polynomial fits with different degrees. The result is plotted below:

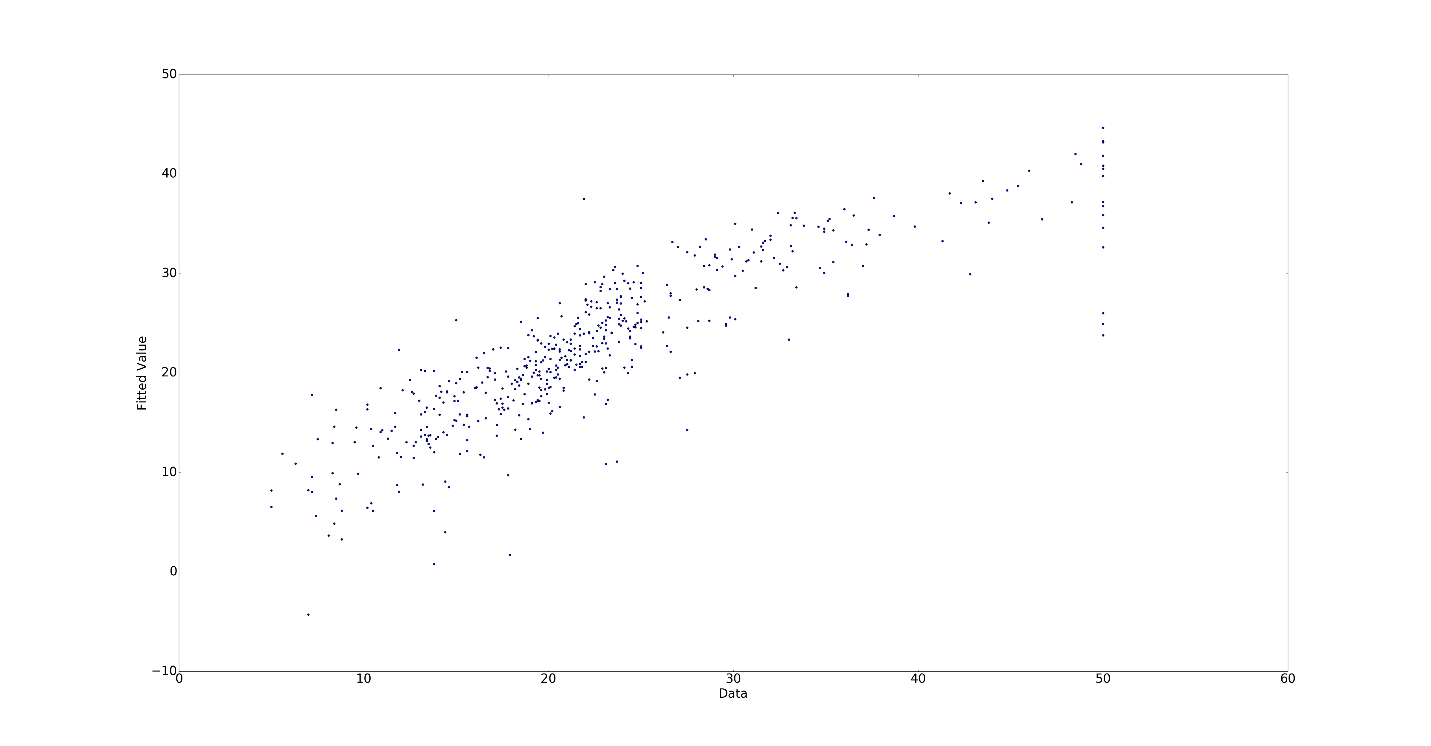
Figure 10- RMSE vs Polynomial Degree

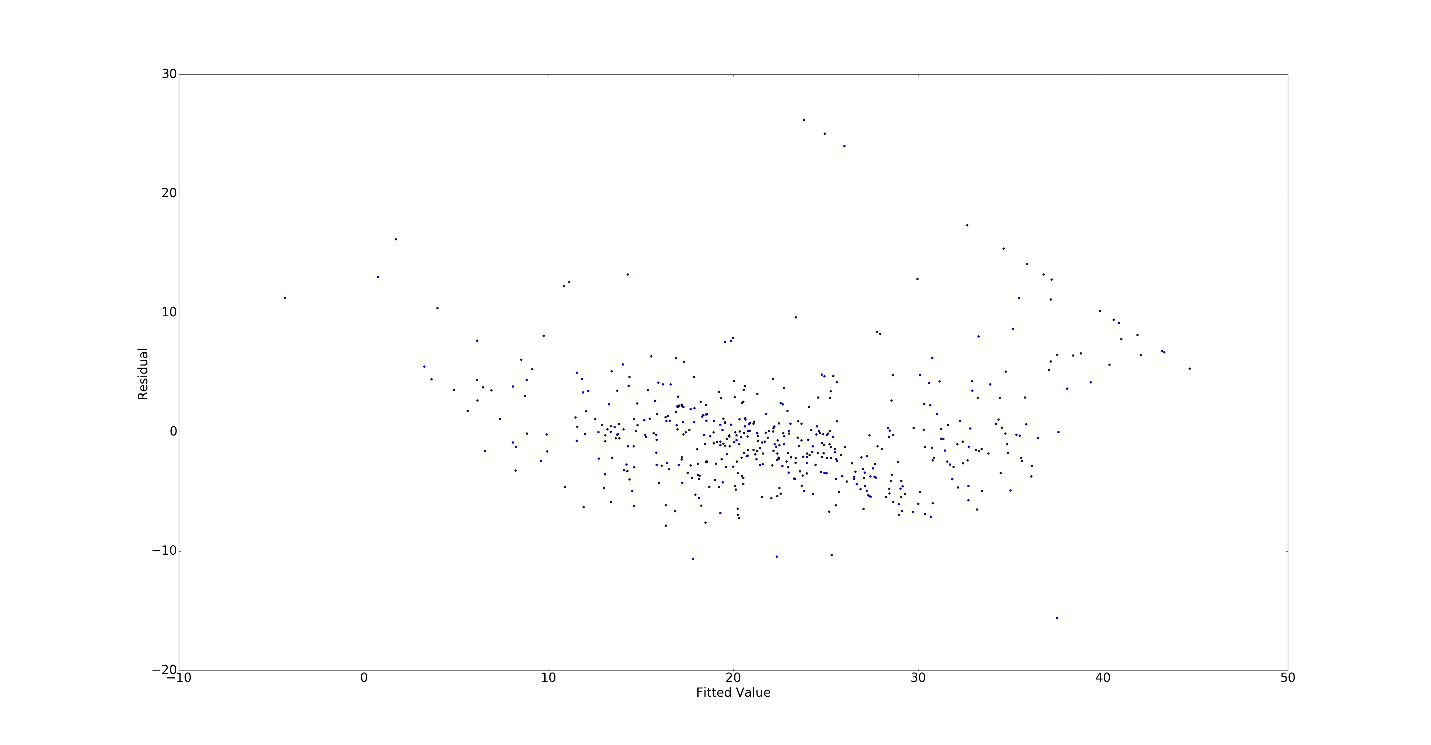
The length of each bar represents the standard deviation of **RMSE** (there are 10 values for 10 different training sets). We can see that initially, increasing the degree of the polynomial would decrease the **RMSE** and therefore better prediction. While increasing the degree of the polynomial results in computational cost, it doesn’t always result in better prediction. We can see that fitting to a 6th degree polynomial worsens our prediction as **RMSE** grows, making a 5th degree polynomial fit as the optimum. The reason is because of overfitting as we saw that during our 10-fold cross validation, the training error was almost zero while the testing error had become significant.



Going over three algorithms and the same steps as part 2, we obtained the following results:

Linear regression

Figure 11- Fitted value vs Actual for Linear regression

Figure 12- Residual vs fitted value for linear regression

**RMSE** : 4.87275227866

Random Forest (n\_trees=20, depth=10)

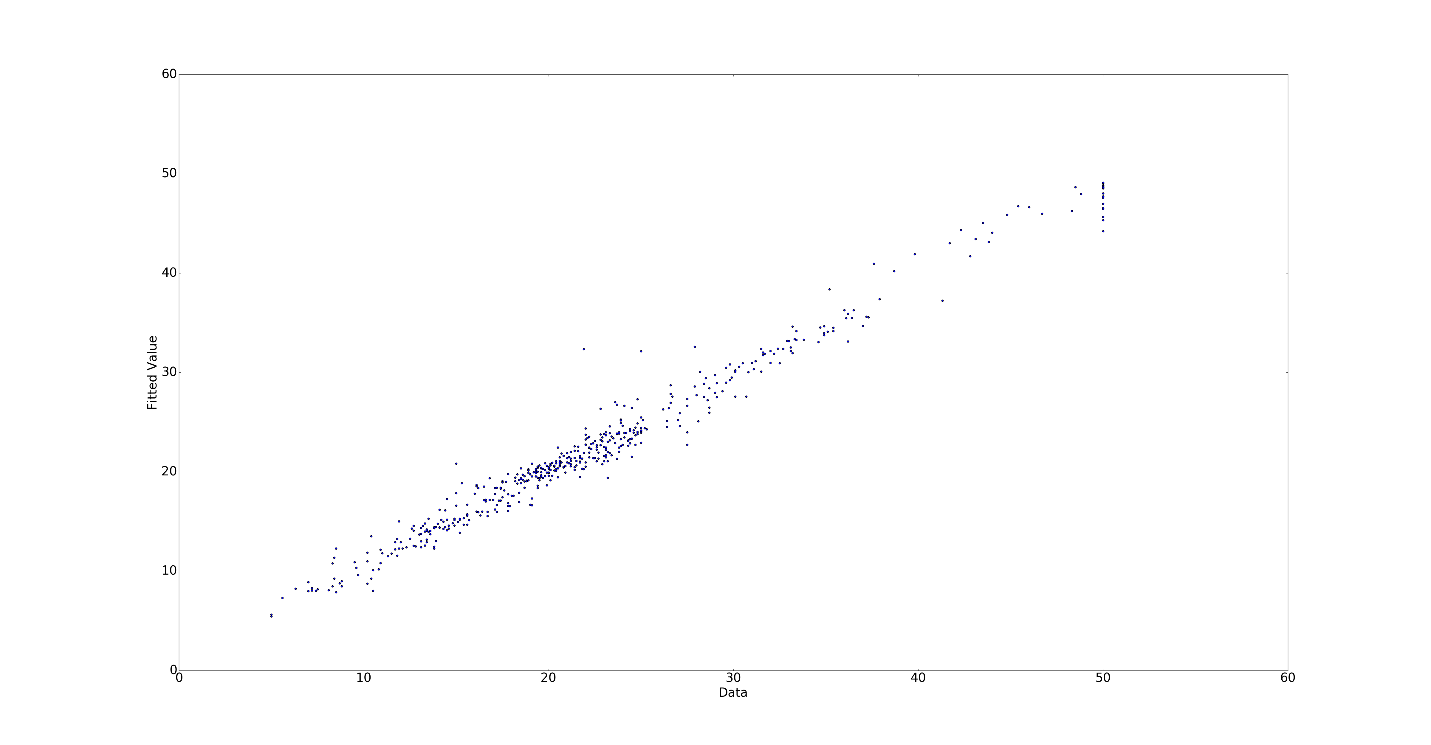


Figure 13- Fitted value vs Actual for Linear regression

**RMSE** : 3.45046852949

Neural Network

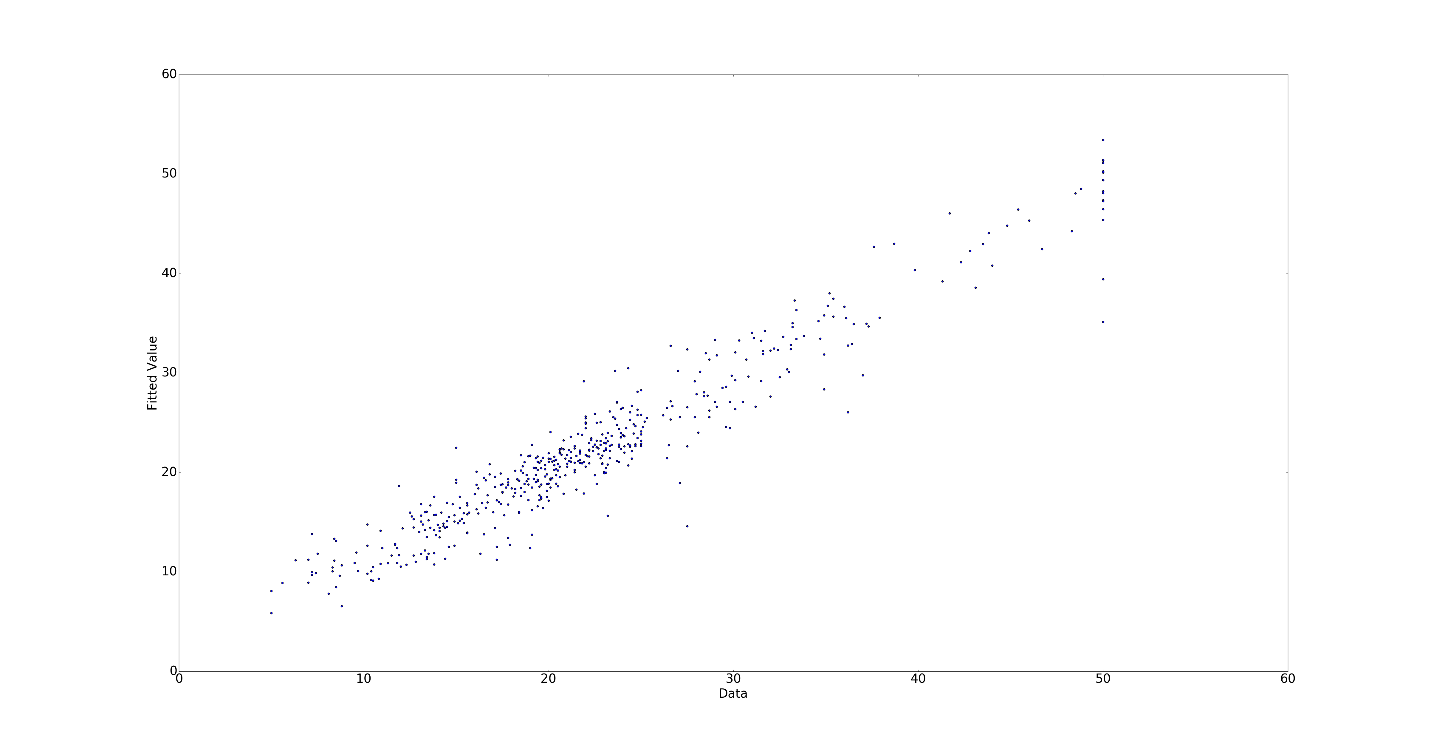


Figure 14- Fitted value vs Actual for Linear regression

**RMSE:** 3.38905446929

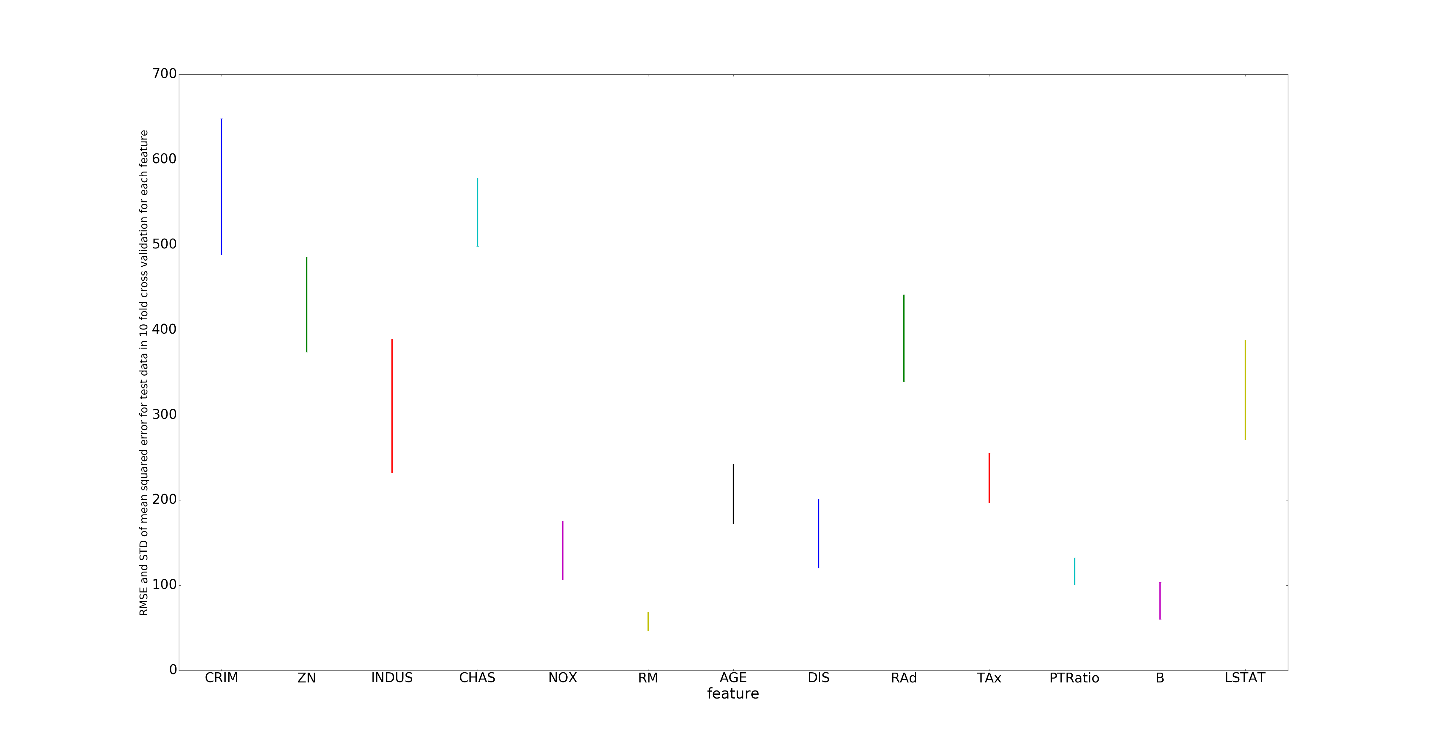
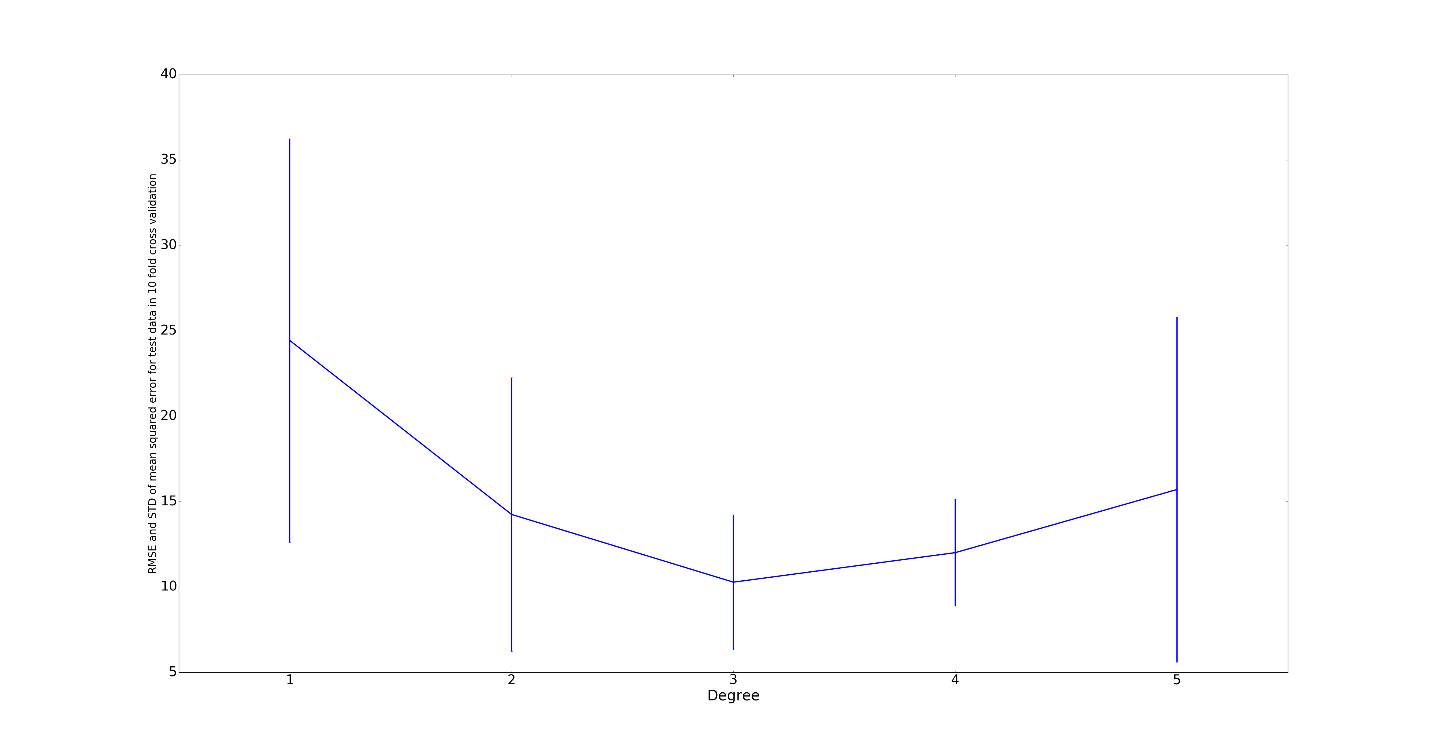
In order to measure the significance of different features, we calculated the RMSE for test data in 10 fold cross validation for each feature. The results are plotted below:  


Figure 15- RMSE vs Feature

The length of each bar represents the standard deviation of the **RMSE** for each instance. We can see that features **RM** and **B** since they both have low error with low deviation from the mean.

Now using the same method as part 3, we fit the data with polynomials of different degrees. The results are plotted below:

Figure 16- RMSE vs Polynomial Degree

We can see that the optimum polynomial degree is 3 and maybe 4. However, for higher degree polynomials, overfitting will start to occur and our prediction won’t be accurate.

1. Regularization of the Parameters:

In ridge regression and lasso regression, we choose to use dense-sparse vector because we think regularization can help us to make the coefficient of each feature reasonable.

a)

We use 4 different alphas to train ridge mode. As can be seen from the figure, as the alpha is set large, the RMSE becomes large too. It is because the large step length will make the solution jump back and forth around the optimal value easily, cause big deviation after each jump.

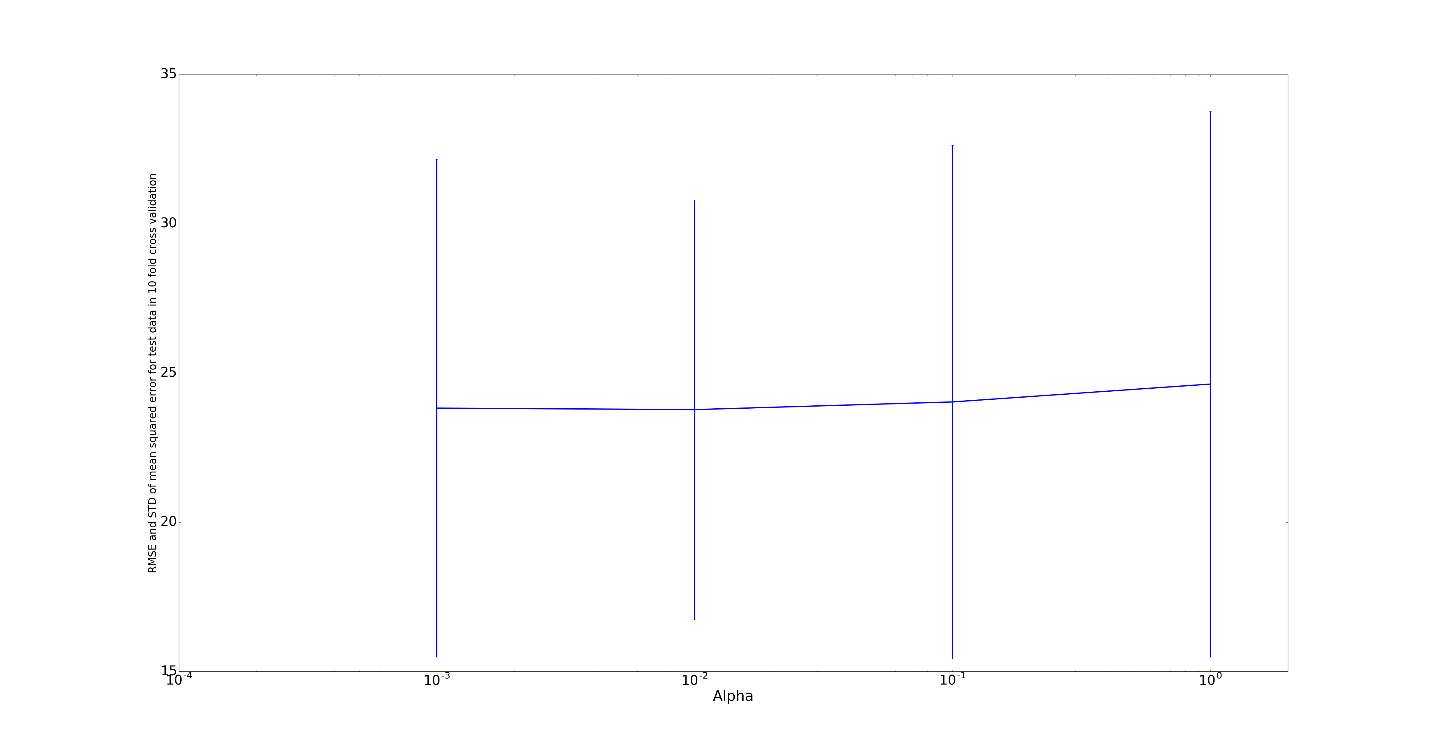


Figure 17- RMSE vs α (ridge model)

The best value for **RMSE** in this case would be 4.83138926511

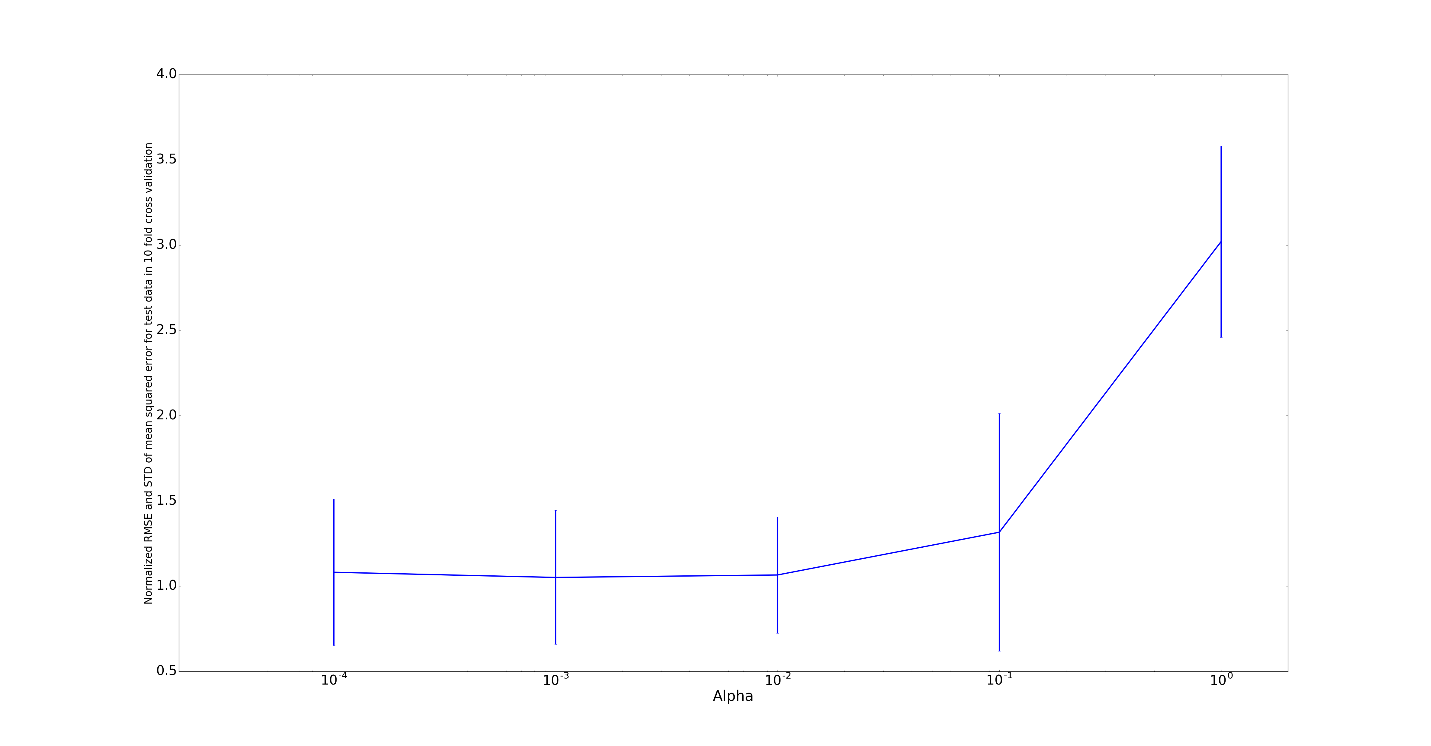
1. For the Lasso model, we used 6 different values of alphas.   
     
    ****

Figure 18- RMSE vs α (Lasso model)

While we can see that large alphas could make our error much larger, we found the best value for **RMSE** to be 4.83451272

Furthermore, the coefficients for the three methods can also be found below:

|  |  |  |
| --- | --- | --- |
| Un-Regularized | Ridge | Lasso |
| -1.080113578367960375e-01 | -0.108 | -0.108 |
| 4.642045836688092619e-02 | 0.046 | 0.046 |
| 2.055862636707226493e-02 | 0.021 | 0.020 |
| 2.686733819344891749e+00 | 2.687 | 2.685 |
| -1.776661122830009276e+01 | -17.767 | -17.733 |
| 3.809865206809212257e+00 | 3.810 | 3.810 |
| 6.922246403444549424e-04 | 0.001 | 0.001 |
| -1.475566845600247534e+00 | -1.476 | -1.475 |
| 3.060494789851759823e-01 | 0.306 | 0.306 |
| -1.233459391657458666e-02 | -0.012 | -0.012 |
| -9.527472317072913643e-01 | -0.953 | -0.952 |
| 9.311683273793820367e-03 | 0.009 | 0.009 |
| -5.247583778554899547e-01 | -0.525 | -0.525 |

Since the optimum alphas are very small, the second term in Ridge and Lasso methods are virtually insignificant. Therefore, we can expect that the coefficients to be almost identical. Which they are!