**EE219 Project1**

**Regression Analysis**

**Winter 2017**

**Team members:**

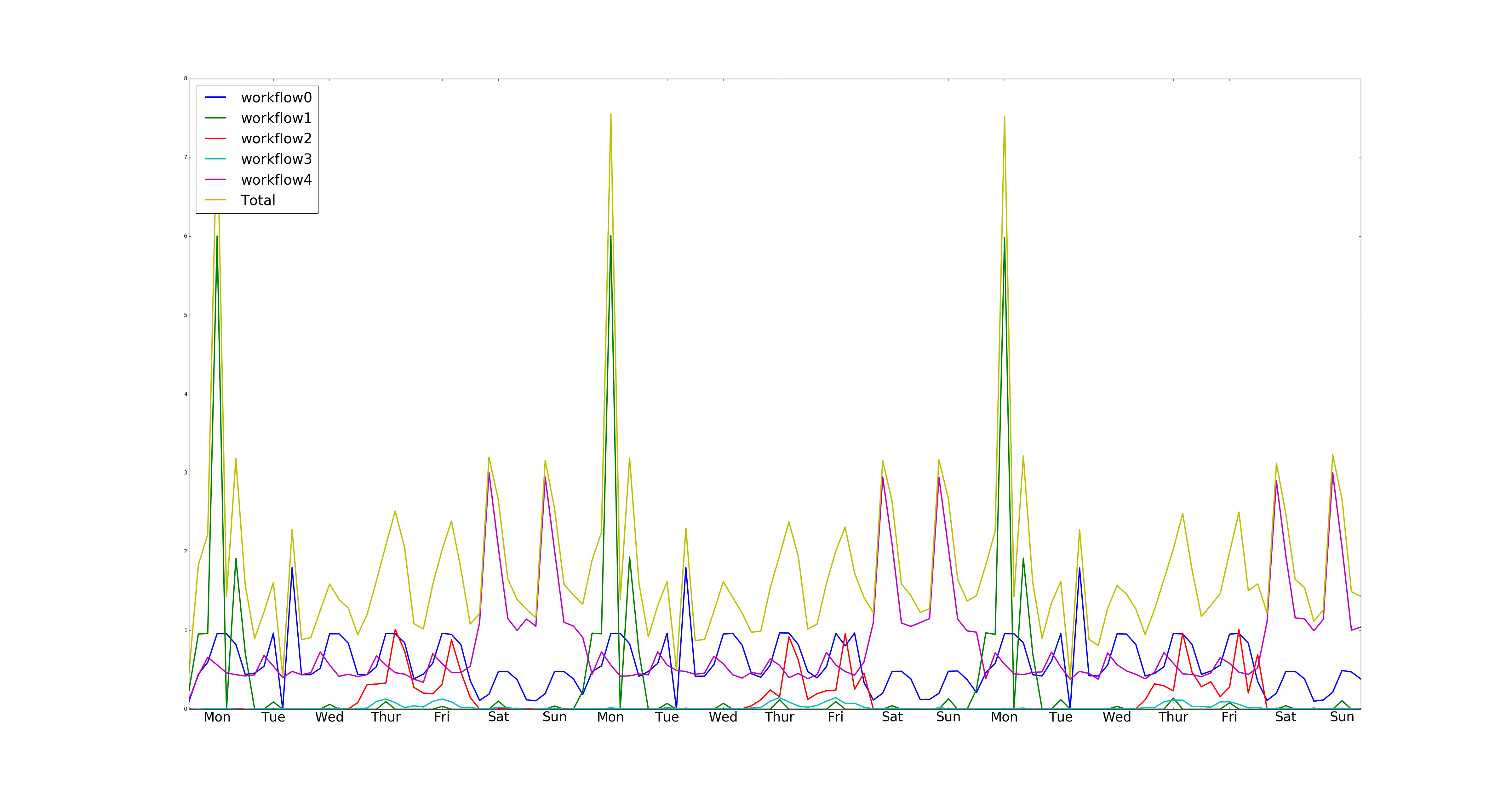
**Mojtaba Sahraee**

**Kia Karbasi 804617014**

**Heng Zhao**

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 0.006332 and the RMSE for 10 fold cross validation is .006334.

But since the features like file number, day of the week, … are categorical variables

If we use this minimization, the entries of vector of coefficients, , would become very large, indicating that the features in are not independent.

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. For this model as we saw, the days of the week are mostly significant variables as can also be seen in the dataset plot (because clearly there is an obvious data size difference between Monday and Wednesday). But the week number is not significant and its T value is not very large and p value not very small. And that is also reasonable because the we have periodicity in the data with period of a week so week number can not distinguish the results.

Here is the result for RMSE:

**RMSE= 0.07150688**

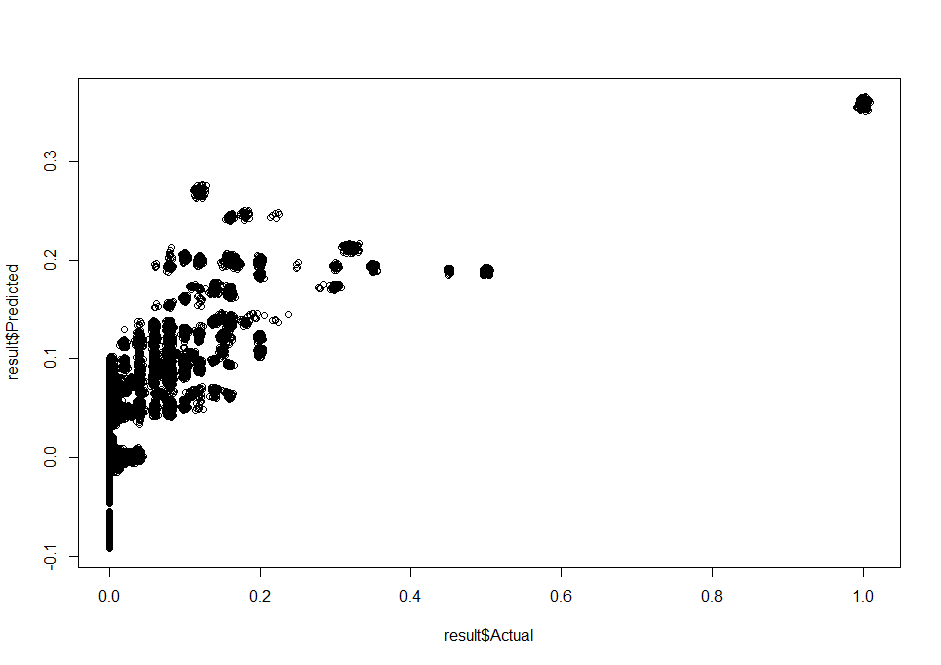


Figure 1- Fitted value vs Actual for Linear regression

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:

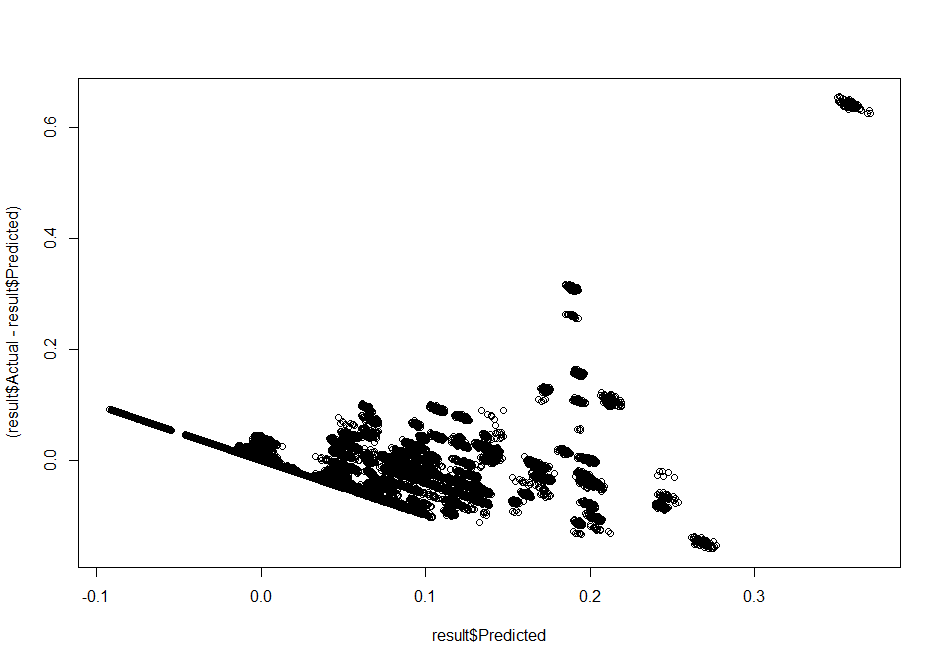


Figure 2-Residual vs Fitted value for Linear regression

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”. Although 7% of RMSE shows that still linear regression might not be the best fit for this dataset.

**b) Random Forest:**

In this part Random Forest algorithm is been used instead of linear regression which might be good for not a linear dataset. The default has been set in the problem to be:

Tree number= 20 , Tree depth= 4

We need to add the randomForest package and use “randomForest” function to model the data:

mymodel <- randomForest(trainingset$Size.of.Backup..GB. ~ ., data = trainingset[,-8], ntree = 40, nodesize=12)

Now we plot the residual and fitted value.

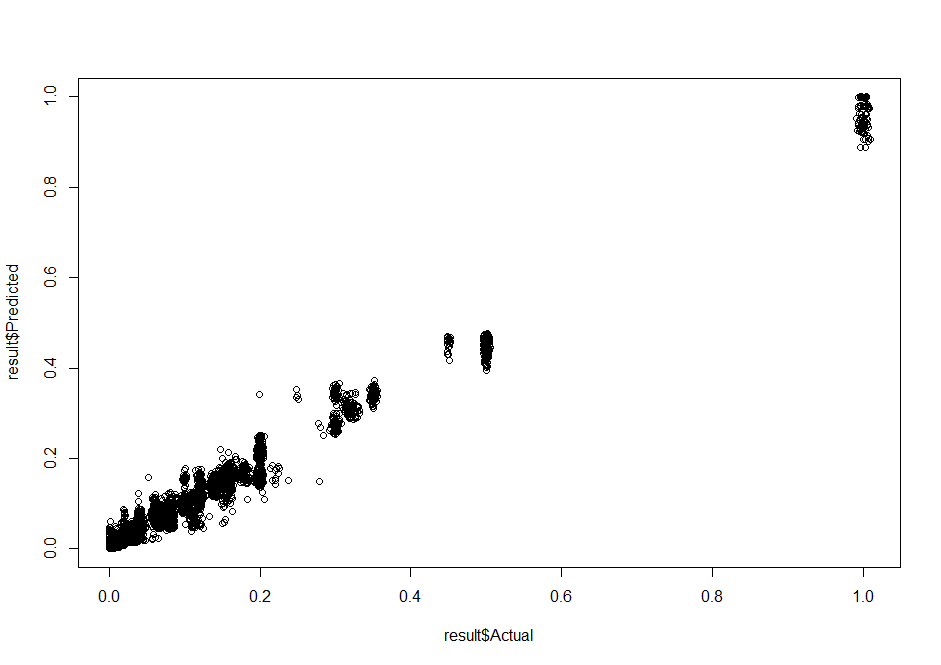


Figure 3-Fitted value vs Actual for Random Forest ntree=20 depth=4

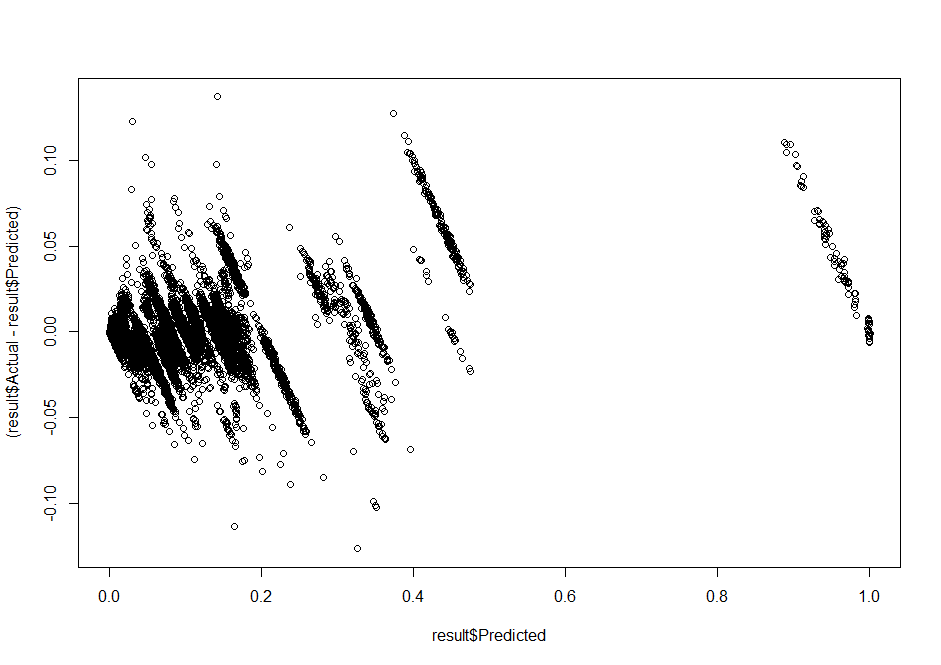


Figure 4- Residual vs Fitted value for Random Forest ntree=20 depth=4

The RMSE found to be: RMSE = 0.01327423

Now tuning the parameters hopefully are giving us better results:

**Tree number= 20 , Tree depth= 8 RMSE= 0.01326013**

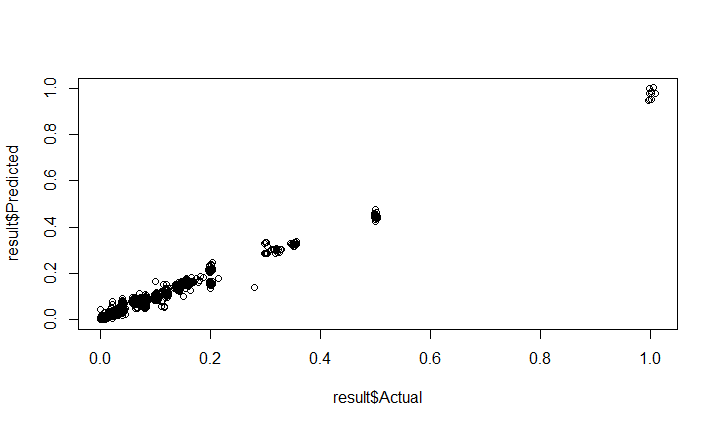


Figure 5-Fitted value vs Actual for Random Forest ntree=20 depth=8

**Tree number= 10 , Tree depth= 4 RMSE= 0.01414224**

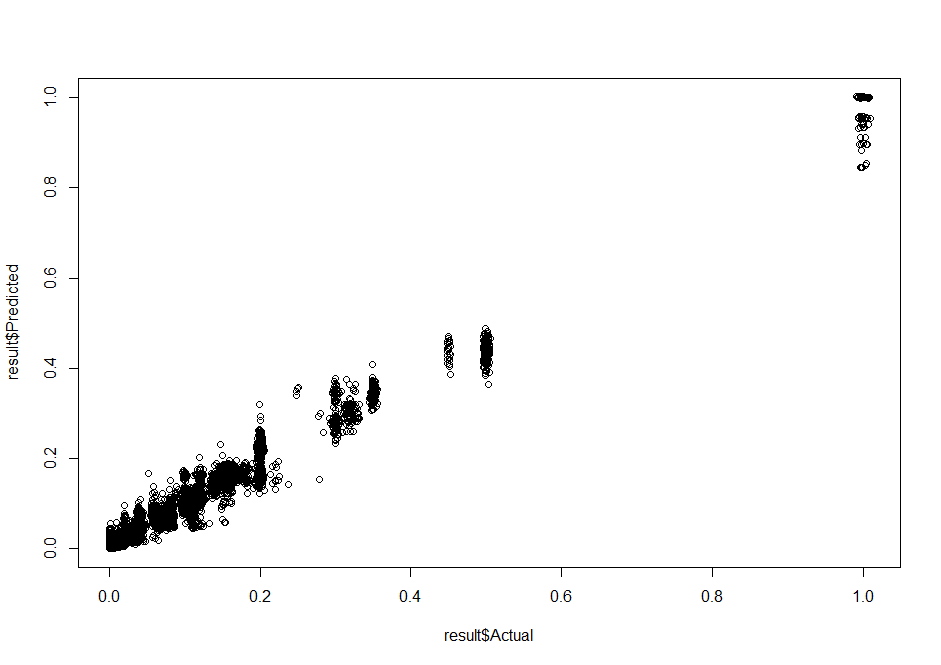


Figure 5-Fitted value vs Actual for Random Forest ntree=10 depth=4

**Tree number= 10 , Tree depth= 8 RMSE= 0.01433228**

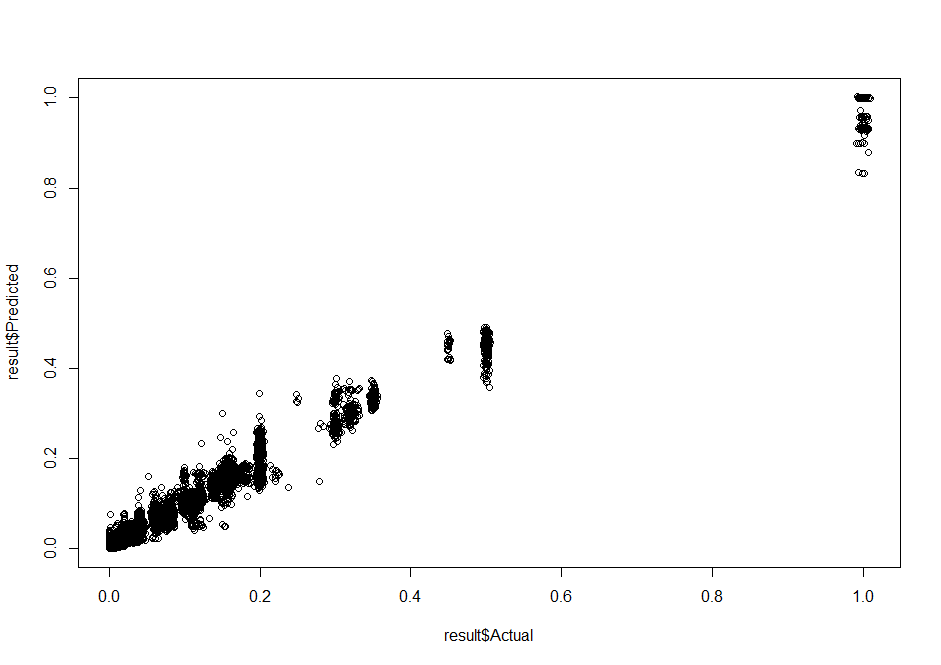


Figure 6-Fitted value vs Actual for Random Forest ntree=10 depth=8

So more 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. But after certain amount of increasing these parameters the curve of error rate will flatten as there is an obvious point of diminishing returns. After that increasing the depth and number of trees does not make sense because it will add the cost and memory and processing time. We found this to be about the optimum tuning:

Tree number= 40 , Tree depth= 12 RMSE= 0.01240899

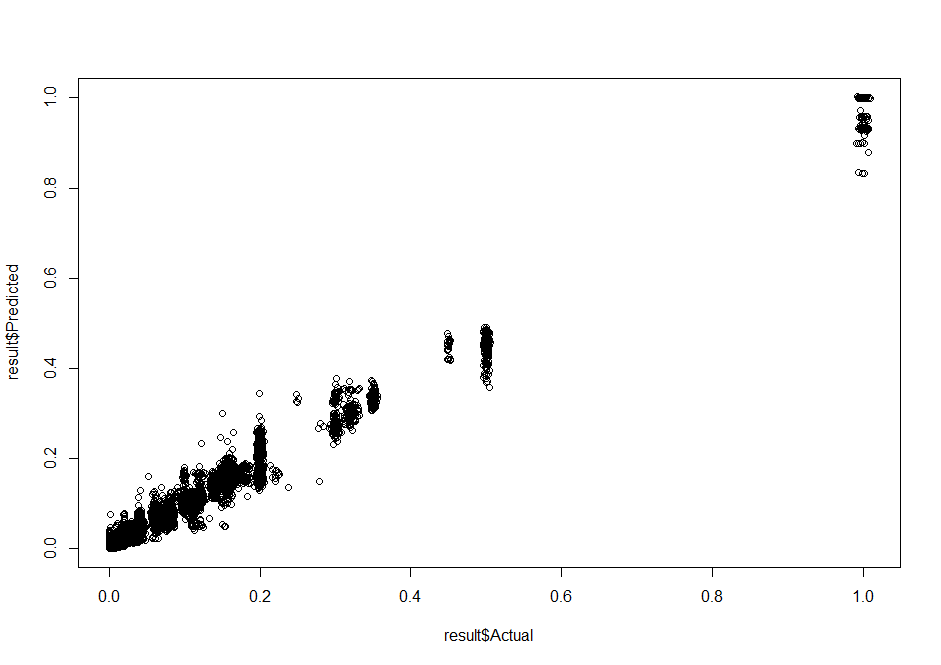


Figure 6-Fitted value vs Actual for Random Forest ntree=40 depth=12

For ntree =60 and depth of 12: RMSE = 0.01453088

For ntree =40 and depth of 20: RMSE = 0.01288931

So the results for 40 and 12 (highlighted seems to be a good stopping point where fairly after that RMSE gets worse)

**c) Neural Network:**

The next step is to model the data with neural network. That has been done with the “nnet” function:

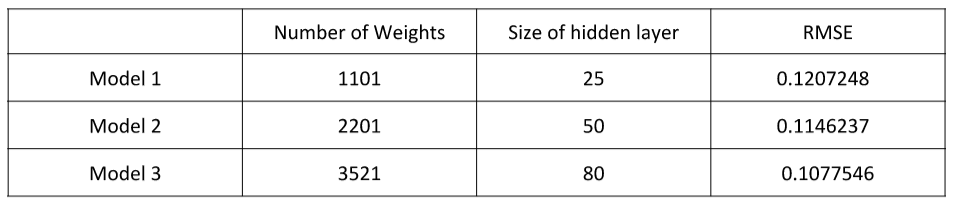
mymodel <- nnet(trainingset$Size.of.Backup..GB. ~ ., data = trainingset[,-8], size = 20)

The result can be seen here for the size of 5 for the network:

Neural network, size=5 RMSE= 0.1207248

In neural network, by increasing size of hidden layer in order to have convergence you should increase the maximum number of iteration as well.

by assigning “*maxint*” in “nnet” object in R you can increase the maximum number of iteration of neural network algorithm.



1. **polynomial**

In this part we try to model the data with a polynomial. So the first step is to train with all the features separately and find the most informative features. Here is the result:

As can be seen the “backup time” and “work flow ID” are the most informative features.

backup time RMSE= 0.07998068

week RMSE= 0.1042106

day of week RMSE= 0.1026971

Backup Start Time Hour of Day RMSE= 0.1037807

File name RMSE= 0.09215041

Work flow id RMSE= 0.0919794

So now we tried to fit the data with polynomial using only these features and see how increasing the degree of polynomial improves the accuracy:

**poly (size ~ work flow id and backup time)**

|  |  |
| --- | --- |
| degree | RMSE |
| 2 | 0.07824315 |
| 3 | 0.07370568 |
| 4 | 0.06595698 |
| 5 | 0.06581353 |
| 6 | 0.06584575 |

As can be seen by the RMSE plot, increasing the degree of the polynomial helps us improve the fit and reduce the error rate and this is because the “backup size” is far from linear so by increasing the complexity of the model we would be able to catch its variation. But as can be seen from the plot after certain amount of poly degree the accuracy tends to saturate and this is because of the overfitting and more coefficient after that will have very limited effect on improving the accuracy and comes with the trade-off of memory and processing time and unit.

1. **Boston data set**

Now we do the same thing we did in part 2 here:

Here are the results

Model : least square error – linear regression

RMSE= 4.861832

Model : Random Forest ntree=20

RMSE= 3.411698

Model : Neural Network

RMSE= 23.41194

Feature Ranking:

CRIM RMSE=8.603689

ZN: RMSE=8.615033

INDUS RMSE= 8.080178

CHAS: RMSE= 9.098518

NOX: RMSE= 8.36158

RM: RMSE= 6.65742

AGE: RMSE=8.534641

DIS: RMSE=8.951091

RAD: RMSE=8.563641

TAX: RMSE= 8.160803

PTRATIO RMSE= 7.966722

B: RMSE= 8.704385

LSTAT RMSE= 6.228161

Now we model the dataset in polynomial only using the RM and LSTAT which are the most informative features. Here you can see how RMSE changes with the degree of the polynomial:

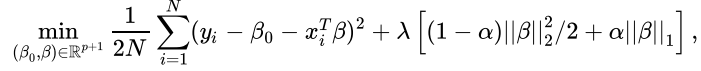
|  |  |
| --- | --- |
| degree | RMSE |
| 2 | 4.696783 |
| 3 | 4.999404 |
| 4 | 4.913379 |
| 5 | 8.356929 |
| 6 | 4.979818 |

Again as can be seen the RMSE saturates for about 4.7 as the degree of poly goes up and this means that for this dataset lower degrees of polynomial fits better. And after 3 and 4 degrees we have overfitting occur.

1. **Regularization of the Parameters:**

In this part, we use glmnet package which is designed in Stanford university. Glmnet is a package that fits a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso or elasticnet penalty at a grid of values for the regularization parameter lambda.

Gaussian is the default family option in the function glmnet . Suppose we have observations and the xi ∈ Rp , responses yi  ∈ R, i = 1, … , N. The objective function for the Gaussian family is



where l> 0 is a complexity parameter and 0 < α< 1 is a compromise between ridge (α=0 ) and lasso (α=1 ).

The regularization parameter lis a control on your fitting parameters. As the magnitudes of the fitting parameters increase, there will be an increasing penalty on the cost function.

In other words, l is a parameter to control overfitting and underfitting. As illustration, if you chose l=0 you have over fitting, because the function solve the least mean square error and fit a model either linear or polynomial to reach least RMSE. Although, if you chose l too large, underfitting occurs, and the curve does not follow the direction of the points as well is it did before. Therefore, in order to have a good fitting, the l should be chosen not too large or not too small.

1. lamda=0.001 RMSE= 4.873139

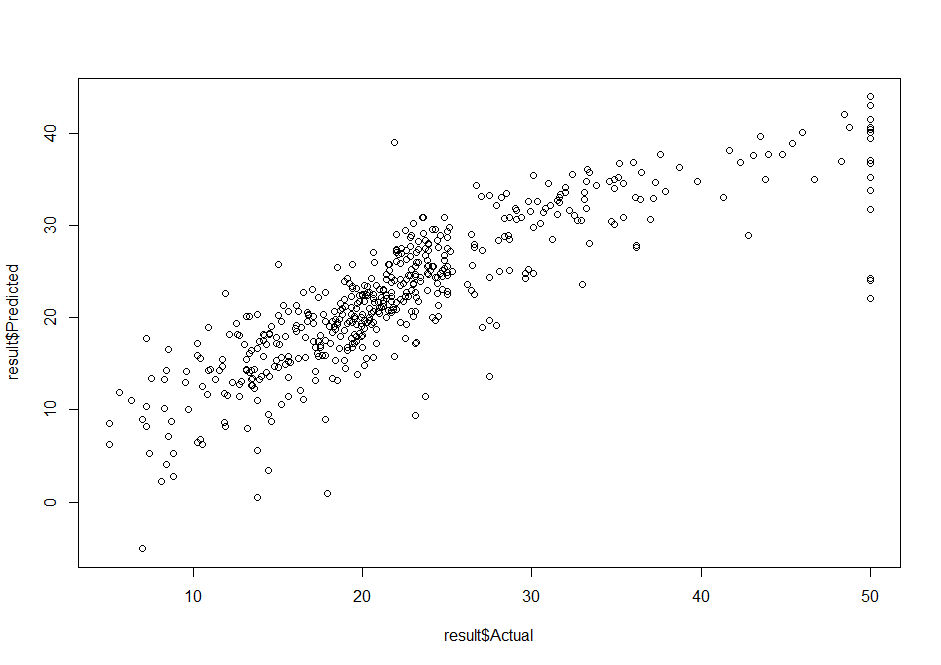


Figure 7-predicted vs actual - lamda=0.001

lamda=0.01 RMSE=4.890635

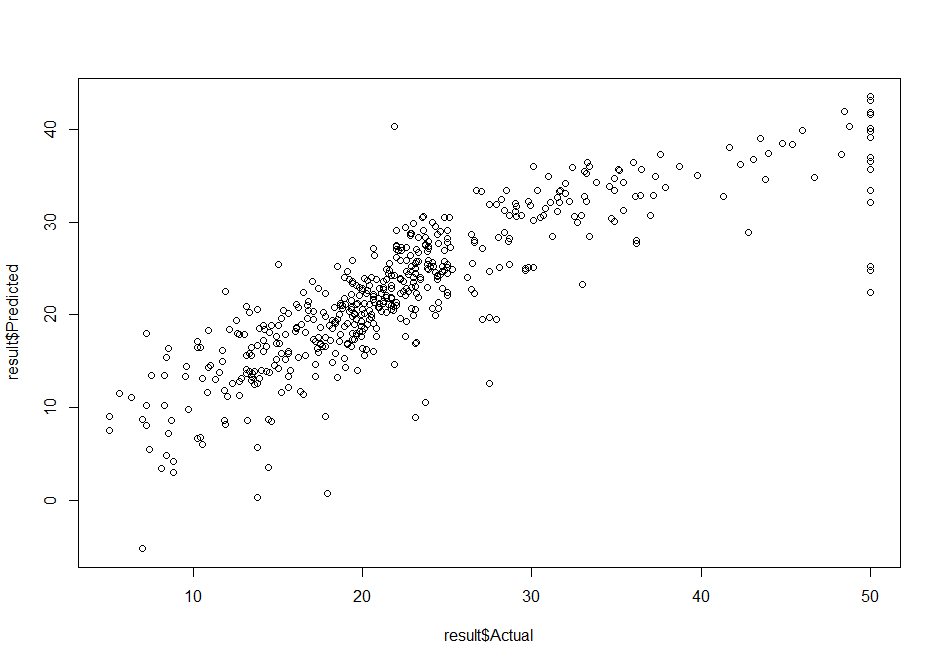


Figure 8-predicted vs actual - lamda=0.01

lamda=0.1 RMSE= 4.83065

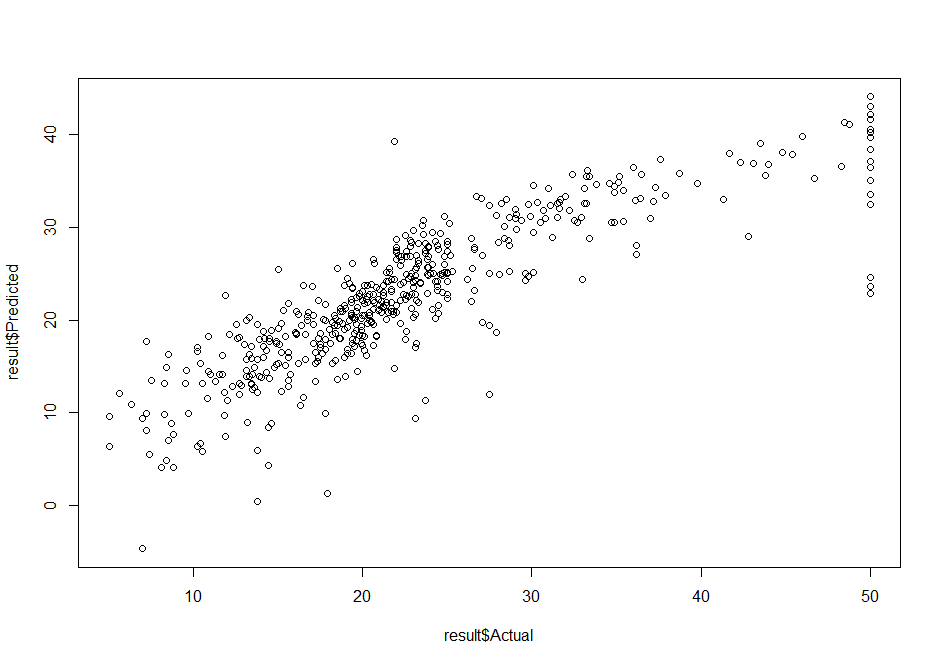


Figure 9-predicted vs actual - lamda=0.1

b)

|  |  |  |  |
| --- | --- | --- | --- |
|  | l=0.1 | l=0.01 | l=0.01 |
| RMSE | 4.928247 | 4.866313 | 4.888762 |