1. Problem 1:-

Build a model to determine the credit worthiness of a customer. Our model predicts whether a customer is ‘credit card defaulter’ or not.

**FLOWCHART:-**

Calculate frequency of repayment

Import data

Implement consistency based feature selection

Build Classification Tree Model

Build Neural Network Model

Build Linear Regression Model

Part 1:

**Algorithm Implementation:-**

* **Pre-processing:**
* Frequency of repayment status:

Calculate the number of each repayment status against individual customer.

out <- sapply(levels,function(x)rowSums(df==x)) #count occurrences of x in each row

colnames(out) <- levels

head(out)

colnames(out)<-c("count2","count\_1","count0", "count\_2","count1","count3","count4","count8","count7","count5","count6")

* **Feature Selection:**

Consistency-based filter

**Package:**

 FSelector

**Version:**

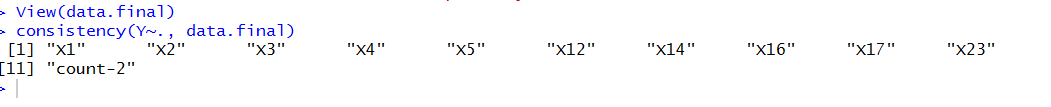
 0.21

**Description**

The algorithm finds attribute subset using consistency measure for continous and discrete data. The alorithm makes use of best.first.search for searching the attribute subset space.

**Usage**

consistency(Y~., data.final)



Important Feature:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE |
|  |  |  |  |  |

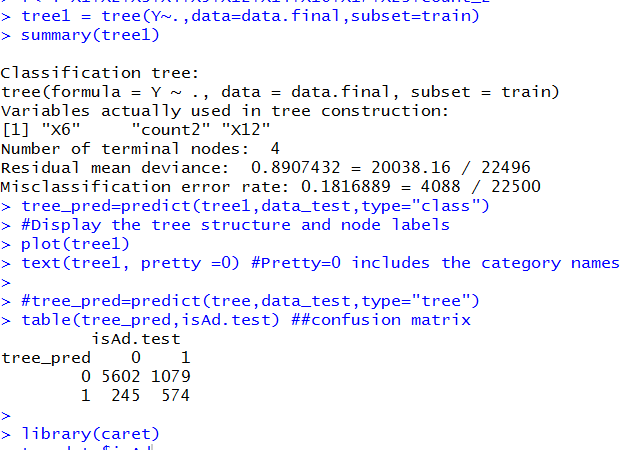
BILL\_AMT1 BILL\_AMT3 BILL\_AMT5 BILL\_AMT6 PAY\_AMT6 Count-2

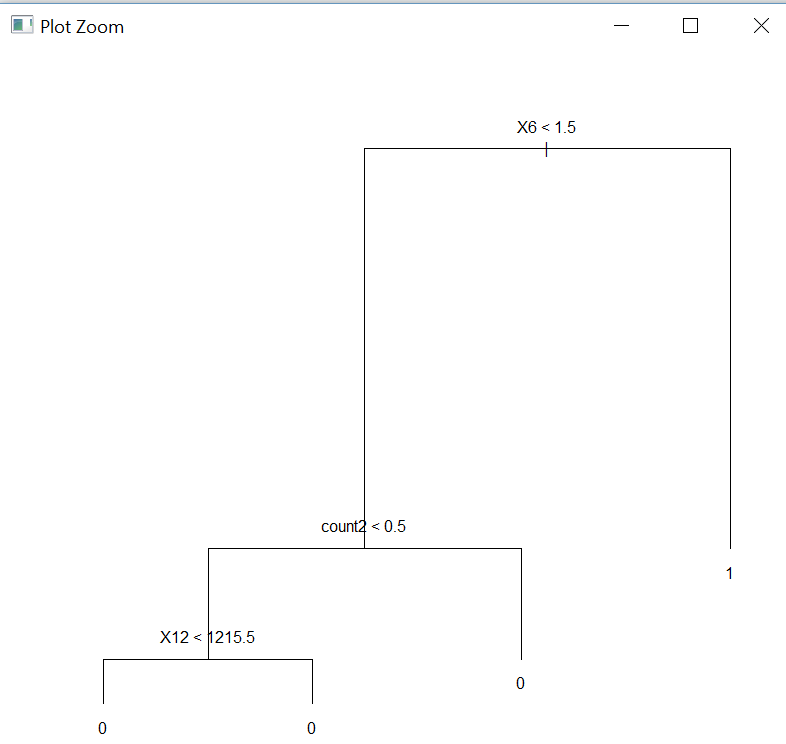
* **Construct a classification tree to calculate the error rate:**

tree1 = tree(Y~.,data=data.final,subset=train)

summary(tree1)

tree\_pred=predict(tree1,data\_test,type="class")





fit.pr = predict(tree1,newdata=data\_test,type="vector")[,2]

fit.pred = prediction(fit.pr,data\_test$Y)

fit.perf = performance(fit.pred,"tpr","fpr")

plot(fit.perf,lwd=2,col="blue", main="ROC: Classification Trees")

abline(a=0,b=1)

test\_prob<-predict(tree1,data\_test,type = "response")

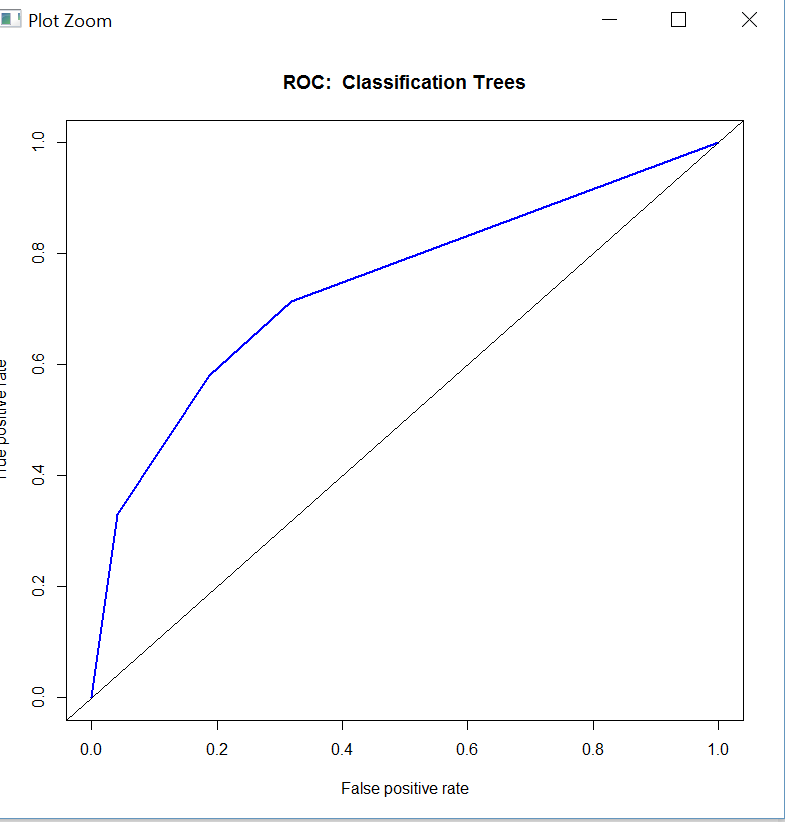
data\_test$probs<-fit.pr

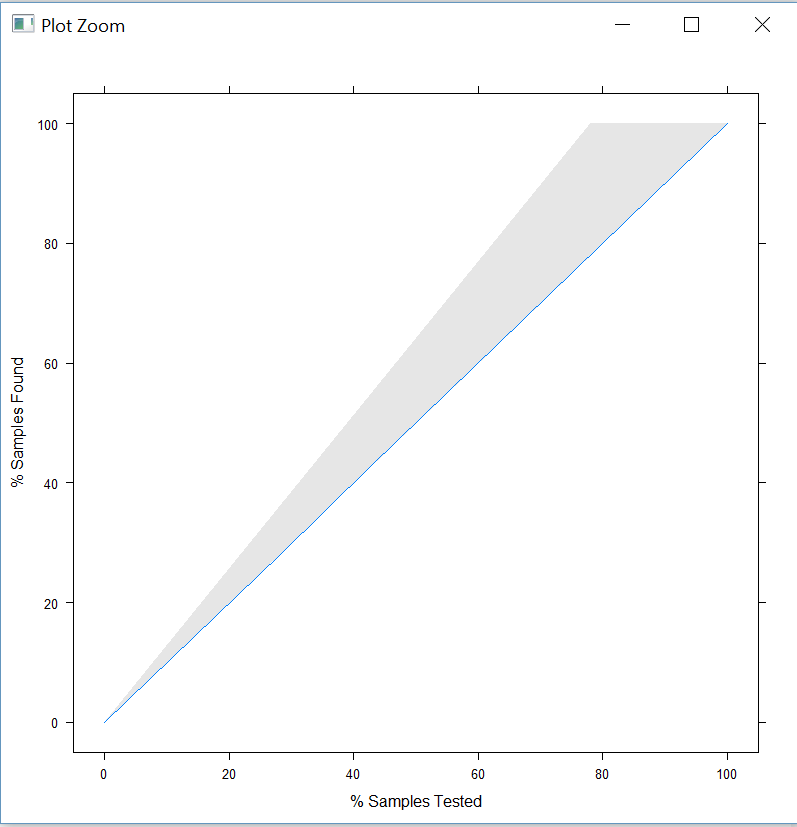
data\_test$prob<-sort(data\_test$probs,decreasing = T)

lift<-lift(Y~prob,data\_test)

lift

xyplot(lift,plot = "gain")





1. **Problem 2:-**

Build a model to predict whether the provided information is for an ‘advertisement’ or ‘non-advertisement’.

**FLOW CHART:-**

Replace the ‘?’ in the data

Import data

Implement backward feature selection

Build Linear Regression Model

Build Classification Tree Model

Build Neural Network Model

Confusion Matrix

ROC Curve

Lift Curve

Part 1:

**Algorithm Implementation:-**

* **Pre-processing:**
* Use package zoo functions to replace the ‘?’:

Remove the trailing and leading whitespaces using trimws and replace all the ‘?’ by ‘NA’. Then use na.approx function of package zoo to replace those values.

* **Feature Selection:**

We implemented feature selection using rfe function of caret package for backward feature selection.

rfe {caret}

**Backwards Feature Selection**

**Package:**

 caret

**Version:**

 6.0-70

**Description**

A simple backwards selection, a.k.a. recursive feature selection (RFE), algorithm

### Details

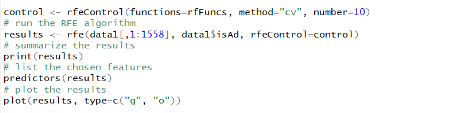
More details on this function can be found at <http://topepo.github.io/caret/featureselection.html>.

This function implements backwards selection of predictors based on predictor importance ranking. The predictors are ranked and the less important ones are sequentially eliminated prior to modeling. The goal is to find a subset of predictors that can be used to produce an accurate model. The web page<http://topepo.github.io/caret/featureselection.html#rfe> has more details and examples related to this function.

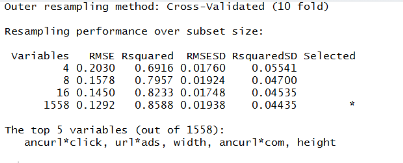
rfe can be used with "explicit parallelism", where different resamples (e.g. cross-validation group) can be split up and run on multiple machines or processors. By default, rfe will use a single processor on the host machine. As of version 4.99 of this package, the framework used for parallel processing uses the foreachpackage. To run the resamples in parallel, the code for rfe does not change; prior to the call to rfe, a parallel backend is registered with foreach (see the examples below).

rfeIter is the basic algorithm while rfe wraps these operations inside of resampling. To avoid selection bias, it is better to use the function rfe than rfeIter.

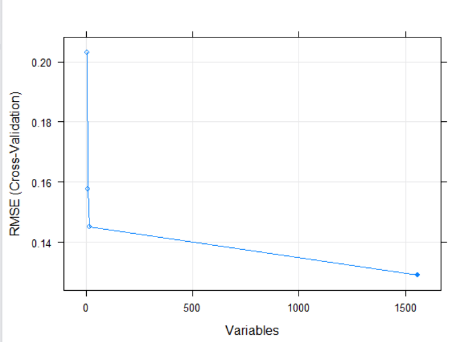
When updating a model, if the entire set of resamples were not saved using rfeControl(returnResamp = "final"), the existing resamples are removed with a warning.

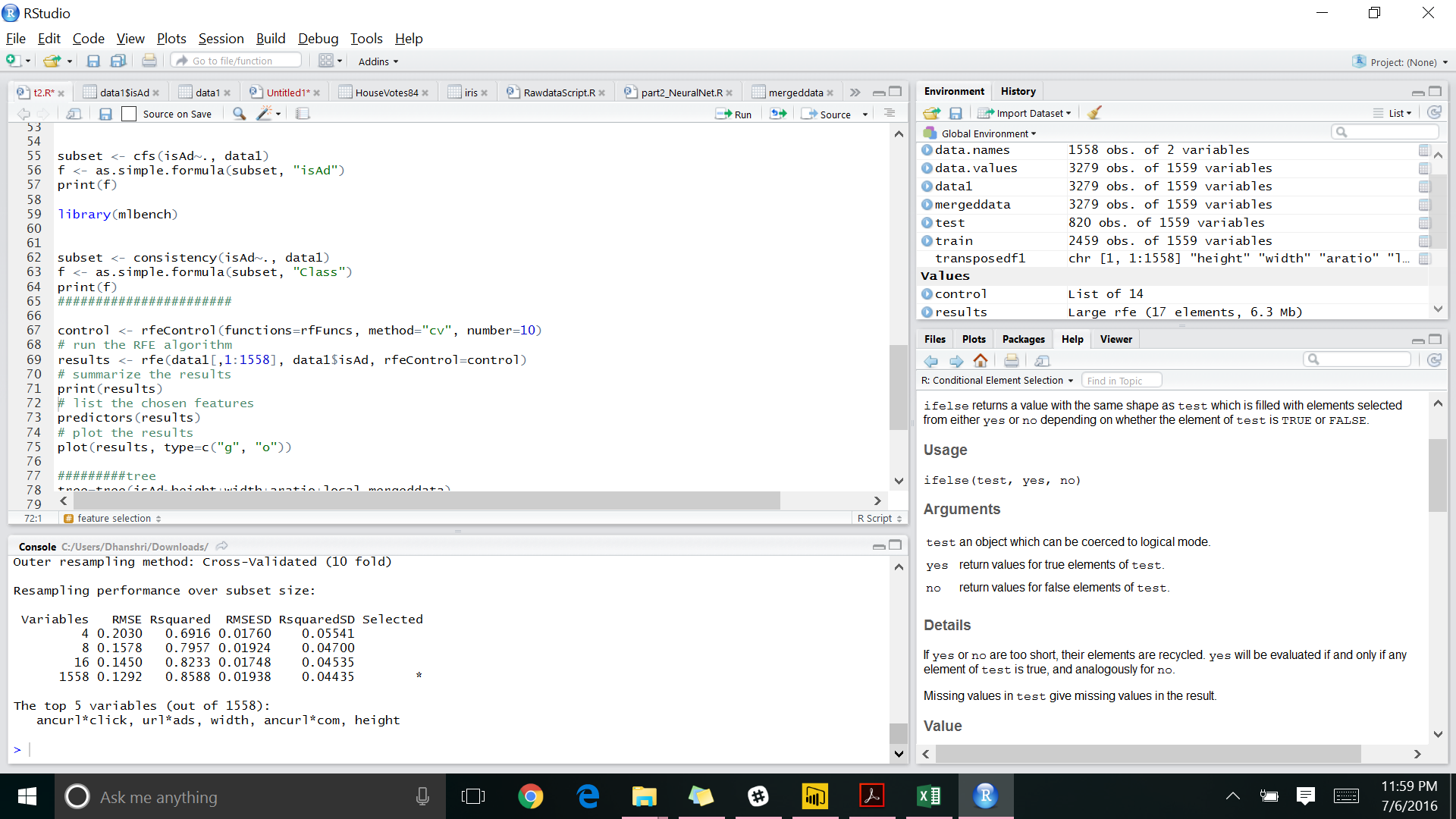


Selected below variable:



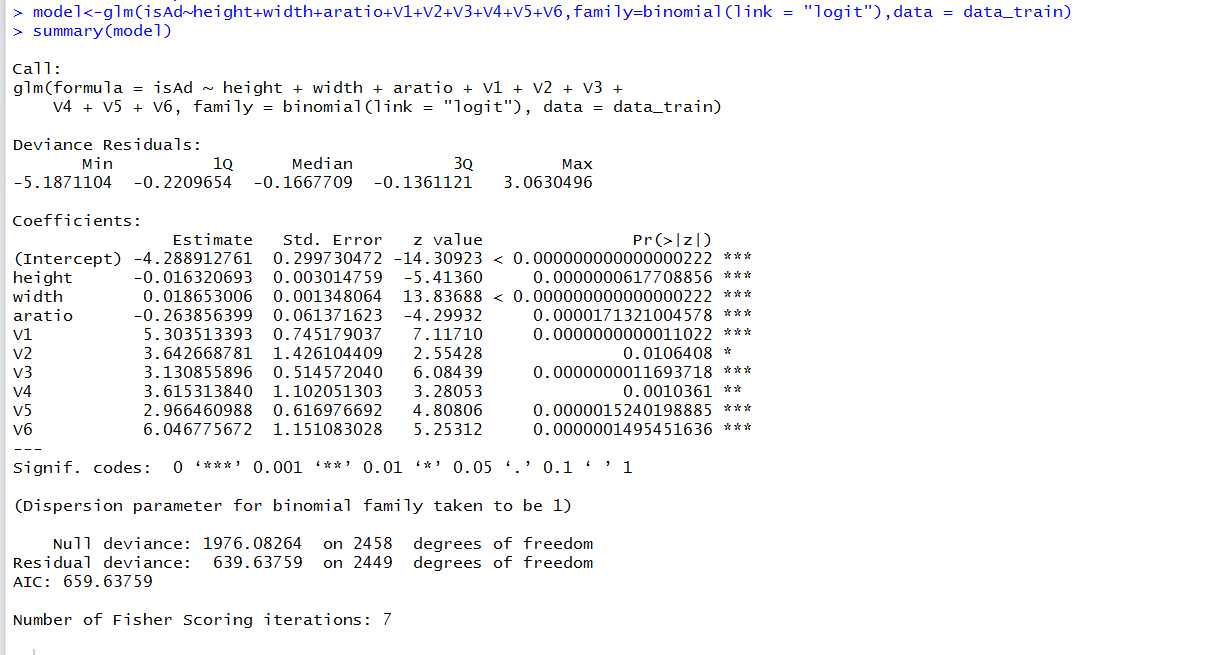
Graph shows variable importance:



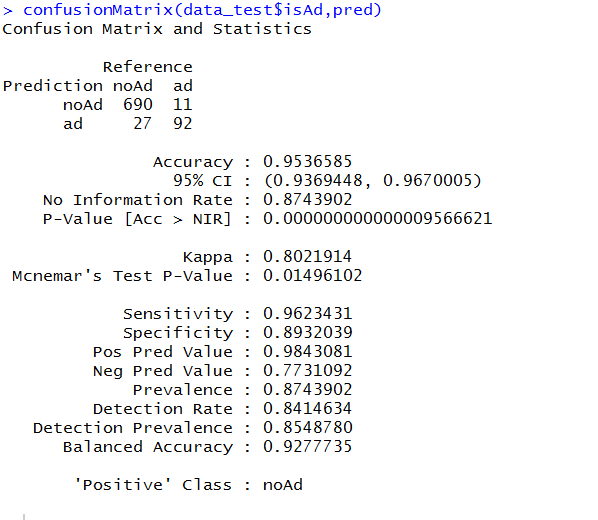


* **Construct a linear regression model:**

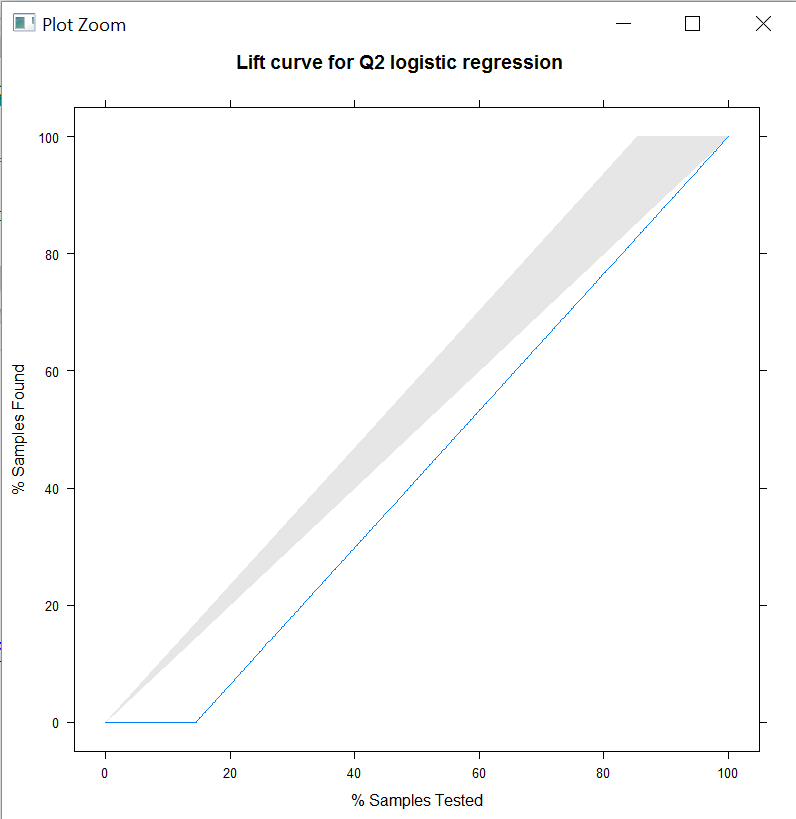
**Model:**



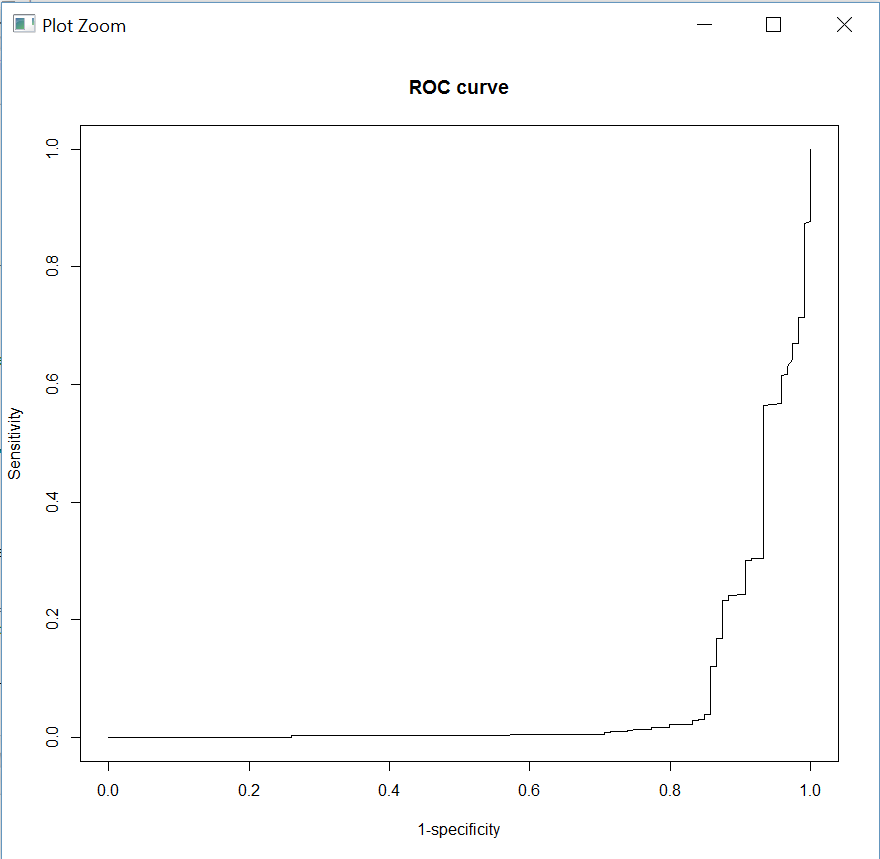
**Confusion Matrix:**



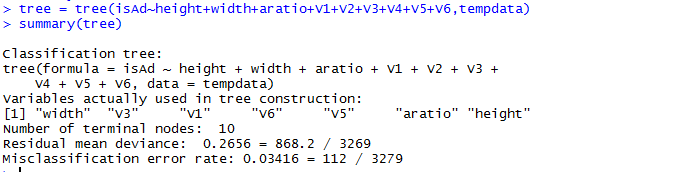
**Lift Curve:**

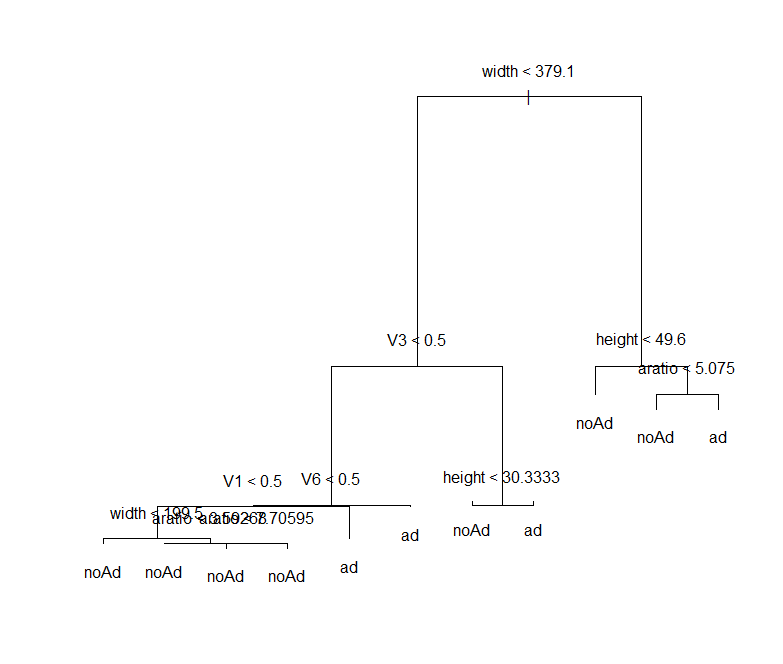


**ROC Curve:**

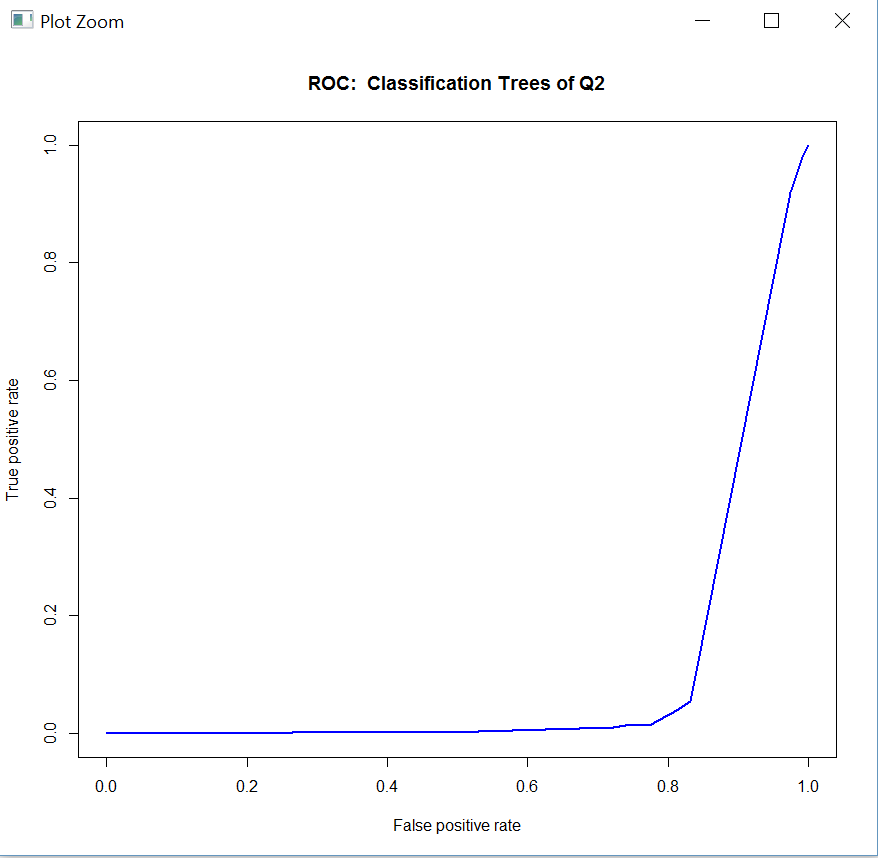


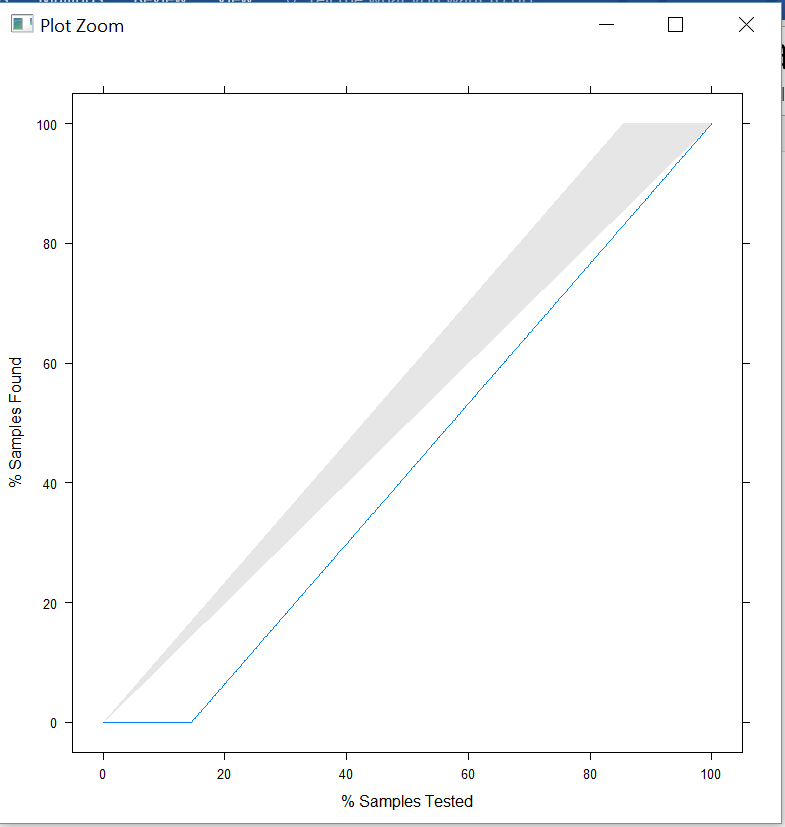
* **Build a classification tree model:**





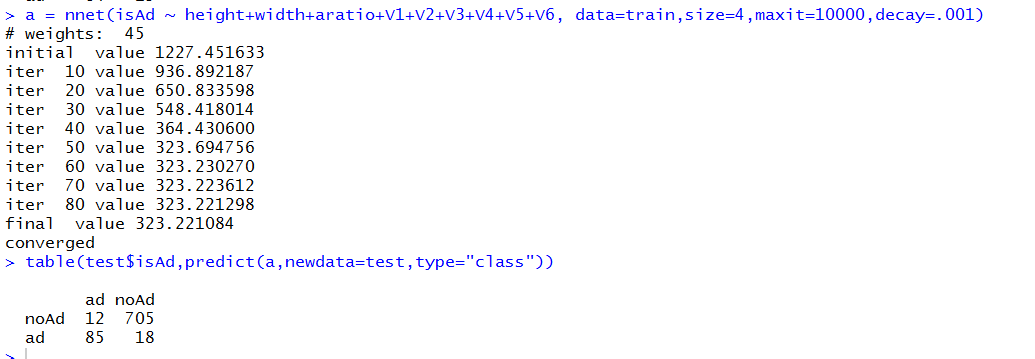
**ROC Curve:**





* **Build a neural network model** based on the variable combination we received in feature selection to predict whether a given information is for ‘ad’ or ‘non-ad’

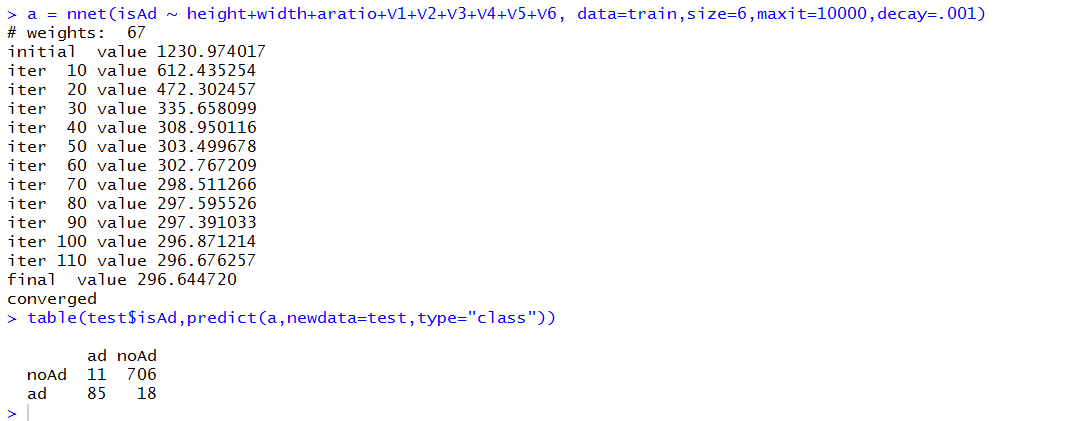
Less than 5 nodes was giving less accuracy… 0.037



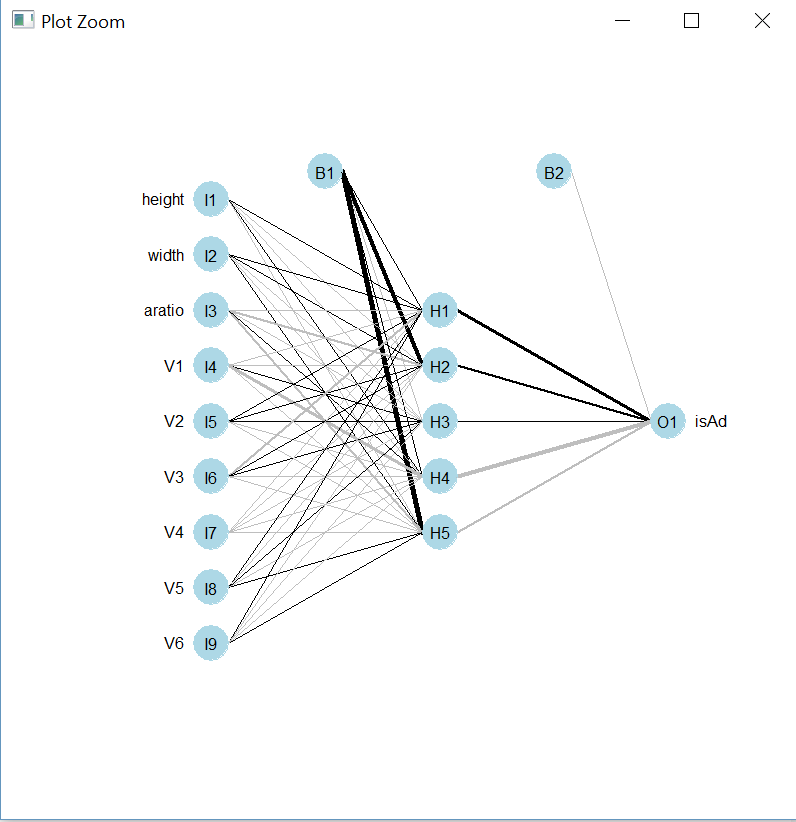
So we selected 5 nodes



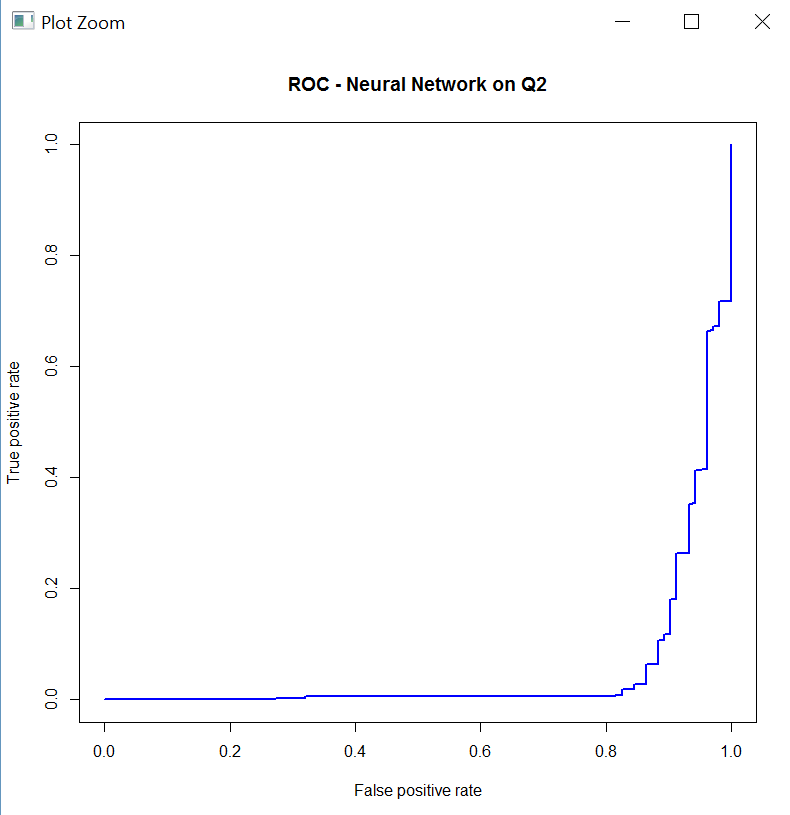
After 6 nodes, again error started increasing



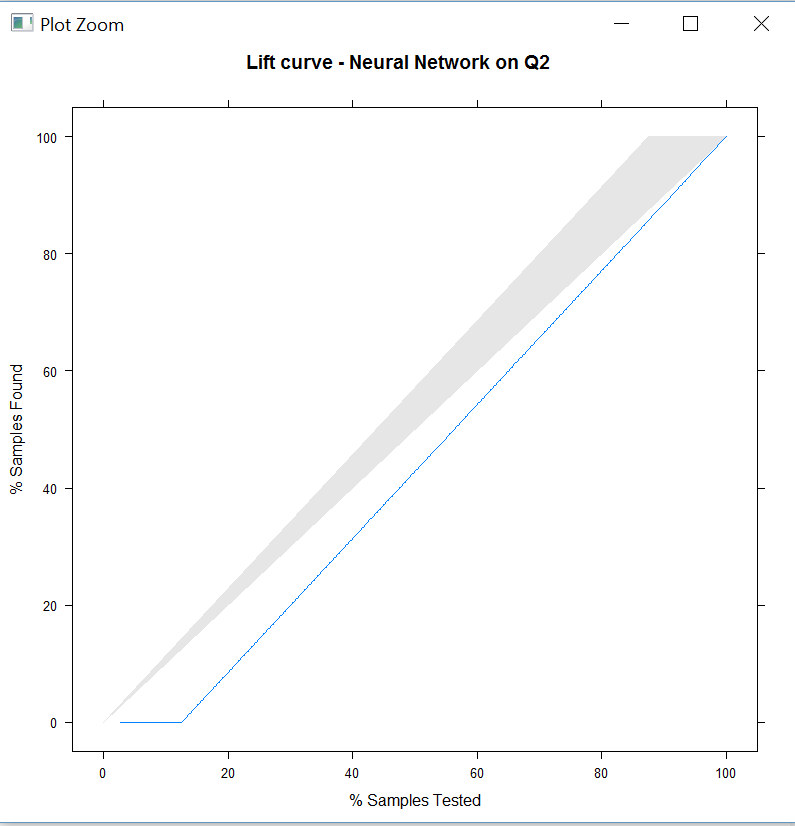
**Neural Network:**



**ROC Curve:**



**Lift curve:**



1. Problem 3:-

Build a model to predict whether the hourly power generation in 7 wind farms for time period 2011/1/1 to 2012/6/28.

**FLOW CHART:-**

Merge files to get hourly consistent data.

Import data

Implement Time Series Analysis to predict power generation

Build Linear Regression Model

Build Classification Tree Model

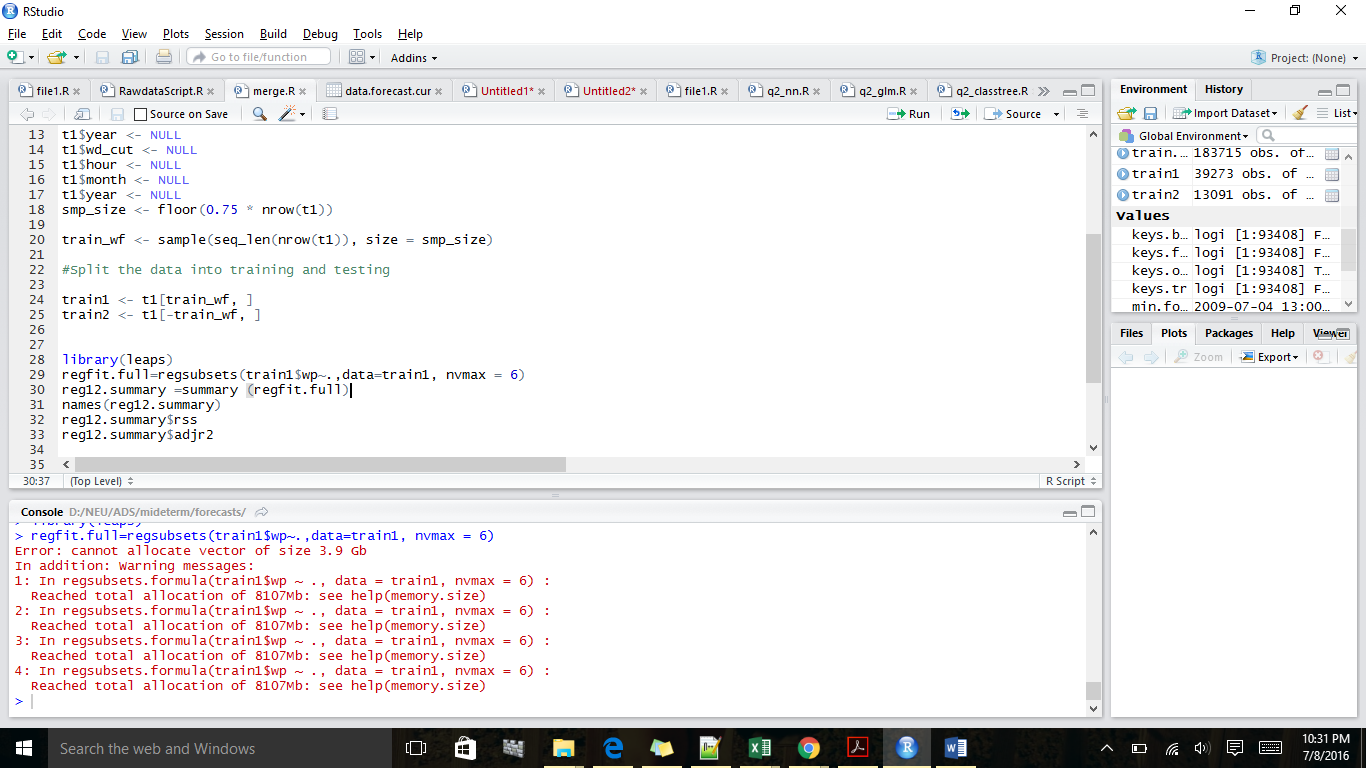
Build a feed forward Neural Network Model

**Approach 1:**

Algorithm Implementation:-

* Merging of files:

1. While analyzing the given data we observed that the data provided in train.csv was missing in benchmark.csv and vice-versa. Hence, it was of crucial importance to merge these files on the basis of ‘date’.
2. To predict the wind forecast for farm 1 based on windforecasts\_wf1.csv, we tried to implement feature selection technique which provided the following error



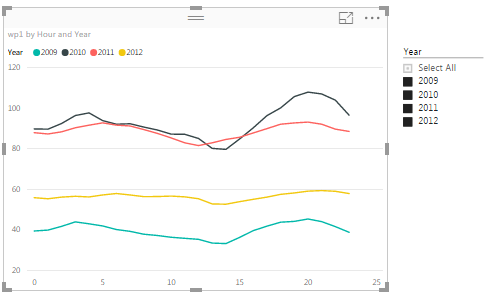
**Approach 2:**

Algorithm Implementation:-

* Merging of files:

1. While analyzing the given data we observed that the data provided in train.csv was missing in benchmark.csv and vice-versa. Hence, it was of crucial importance to merge these files on the basis of ‘date’.
2. We used Time Series Analysis to predict the power generation for a particular wind farm:

Wind generation data has seasonality on hourly basis which repeats every year, so we did neural net time series model to forecast 48 hours ahead power generation



Wind power generation drops around 10 to 15 hours and increase during 17 to 22 hours

We used Neural Network Time Series for forecasting wind power generation

**Neural Network Time Series Forecasts**

**Package:**

forecast

**Version:**

 7.1

**Description**

Feed-forward neural networks with a single hidden layer and lagged inputs for forecasting univariate time series.

### Details

A feed-forward neural network is fitted with lagged values of x as inputs and a single hidden layer with sizenodes. The inputs are for lags 1 to p, and lags m to mP where m=[**frequency**](http://inside-r.org/r-doc/stats/frequency)(x). If there are missing values in x or xreg), the corresponding rows (and any others which depend on them as lags) are omitted from the fit. A total of repeats networks are fitted, each with random starting weights. These are then averaged when computing forecasts. The network is trained for one-step forecasting. Multi-step forecasts are computed recursively.

For non-seasonal data, the fitted model is denoted as an NNAR(p,k) model, where k is the number of hidden nodes. This is analogous to an AR(p) model but with nonlinear functions. For seasonal data, the fitted model is called an NNAR(p,P,k)[m] model, which is analogous to an ARIMA(p,0,0)(P,0,0)[m] model but with nonlinear functions.

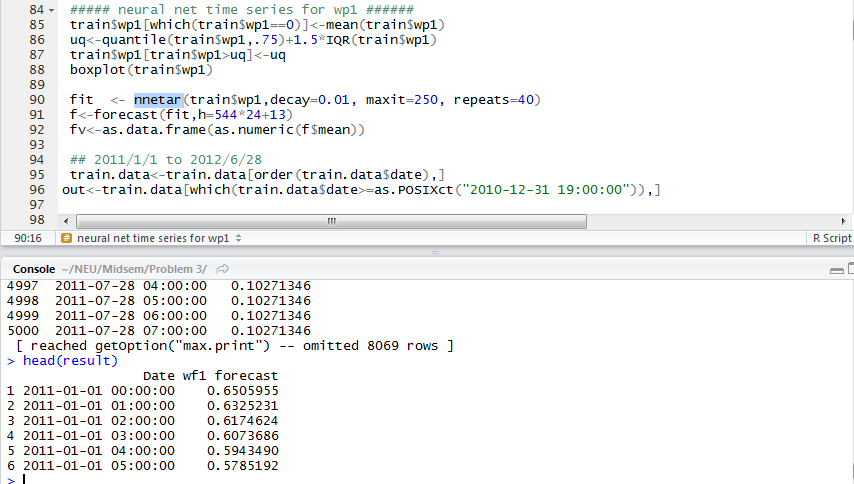
### Values

nnetar returns an object of class "nnetar". forecast.nnetar returns an object of class "forecast".

The function [**summary**](http://inside-r.org/r-doc/base/summary) is used to obtain and print a summary of the results, while the

[1]

Screenshot for Neural net time series model for wp1:



References:

[1] <http://www.inside-r.org/packages/cran/forecast/docs/nnetar>

[2] <http://www.inside-r.org/packages/cran/caret/docs/update.rfe>

[3] <http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/>

[4] <http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/>

[5] [https://en.m.wikibooks.org/wiki/Data\_Mining\_Algorithms\_In\_R/Dimensionality\_Reduction](https://en.m.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Dimensionality_Reduction" \t "_blank)