

DATA MINING ASSIGNMENT

AMMANAMANCHI SAI KARTHIK

B150310CS

10

About the dataset:

buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

The dataset has 6 feature variables and 1 class variable

Number of Instances: 1728 (instances completely cover the attribute space)

Number of Attributes: 6

1 a. Decision Tree with Gini index as the impurity measure

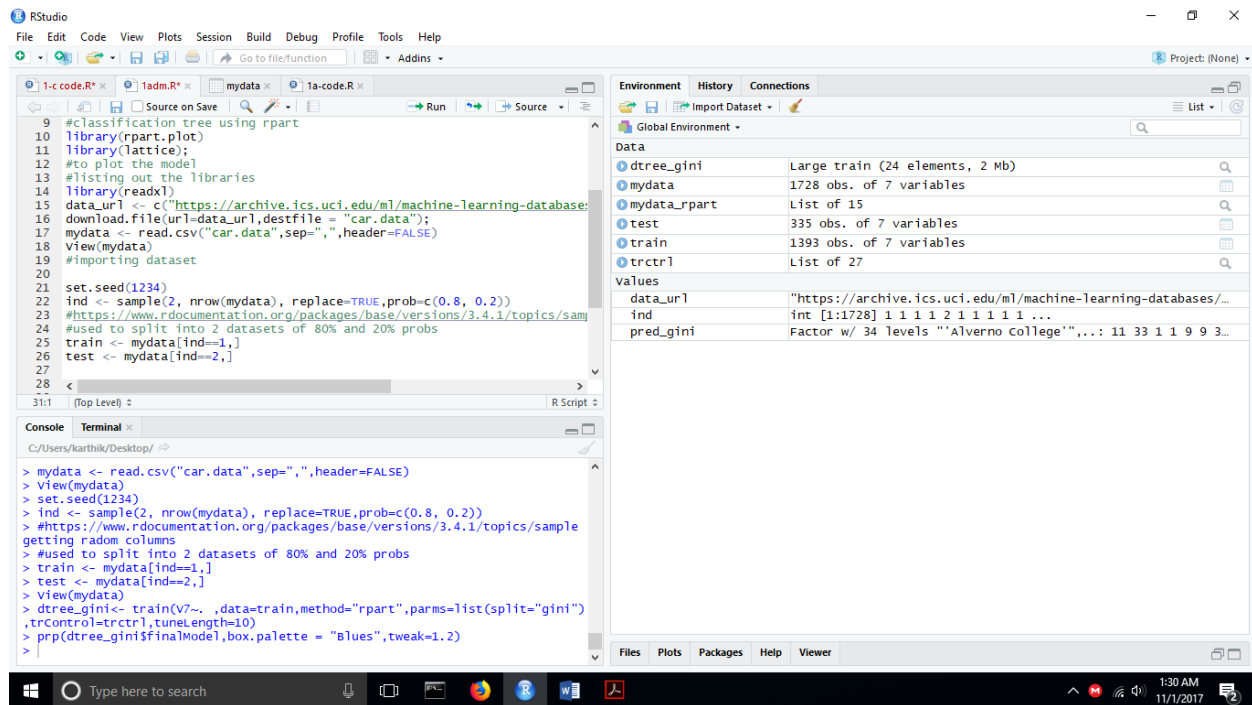
Loading the data

```
data_url <- c("https://archive.ics.uci.edu/ml/machine-learning-  
databases/car/car.data");  
download.file(url=data_url, destfile = "car.data");  
mydata <- read.csv("car.data", sep=",", header=FALSE)
```

Data Slicing

Splitting the data into training set and test set

```
ind <- sample(2, nrow(mydata), replace=TRUE, prob=c(0.8, 0.2))  
train <- mydata[ind==1,]  
test <- mydata[ind==2,]
```

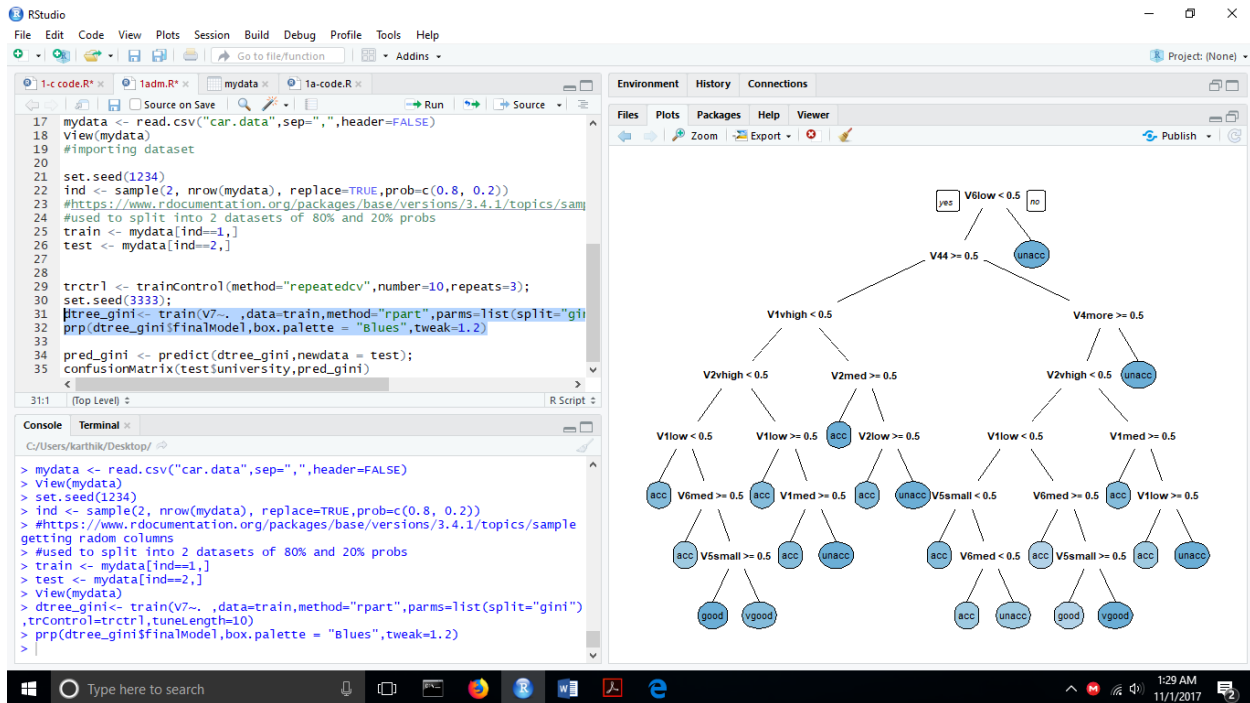


Training the dataset using the caret library based on decision on gini index on university column

```
trctrl <- trainControl(method="repeatedcv", number=10, repeats=3);
set.seed(3333);
dtree_gini<- train(V7 ~.
, data=train, method="rpart", parms=list(split="gini"), trControl=trctrl, tuneLength=10)
```

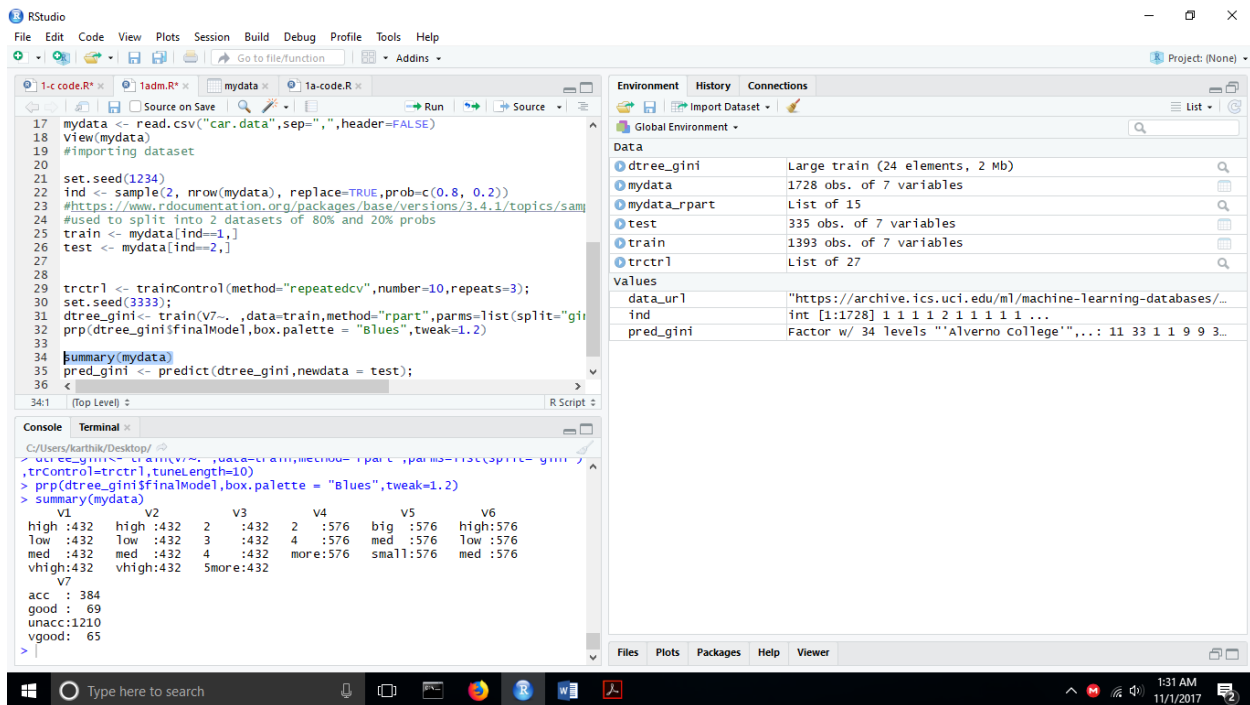
Displaying the dataset

```
prp(dtree_gini$finalModel, box.palette = "Blues", tweak=1.2)
```



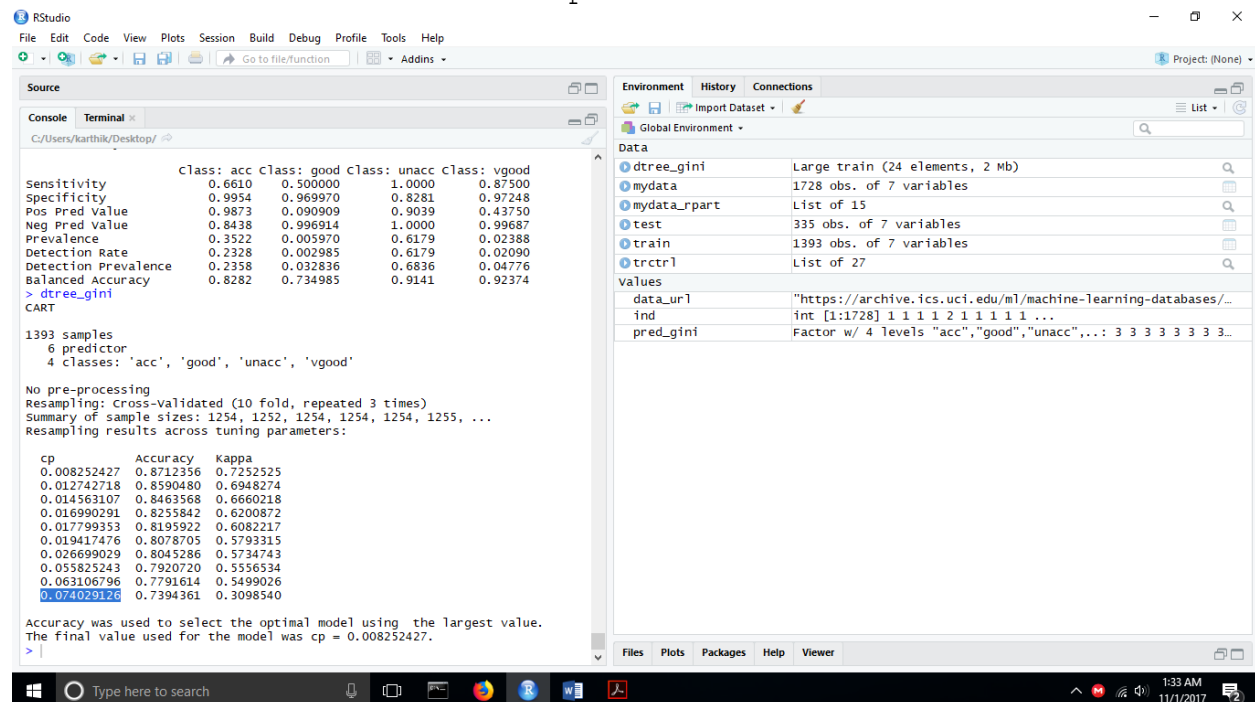
We can check if our dataset contains any missing values using the `anyNA(dataset)` function

To check the summary of the dataset we can use `summary(dataset)`



Prediction

Our model is trained with a cp of 0.008252427



```
Source
Console Terminal
C:/Users/karthik/Desktop/

Class: acc Class: good Class: unacc Class: vgood
Sensitivity      0.6610 0.500000 1.0000 0.87500
Specificity      0.9954 0.969970 0.8281 0.97248
Pos Pred Value   0.9873 0.990909 0.9039 0.43750
Neg Pred Value   0.8438 0.996914 1.0000 0.99687
Prevalence       0.3522 0.005970 0.6179 0.02388
Detection Rate   0.2328 0.002985 0.6179 0.02090
Detection Prevalence 0.2358 0.032836 0.6836 0.04776
Balanced Accuracy 0.8282 0.734985 0.9141 0.92374
> dtree_gini
CART

1393 samples
6 predictor
4 classes: 'acc', 'good', 'unacc', 'vgood'

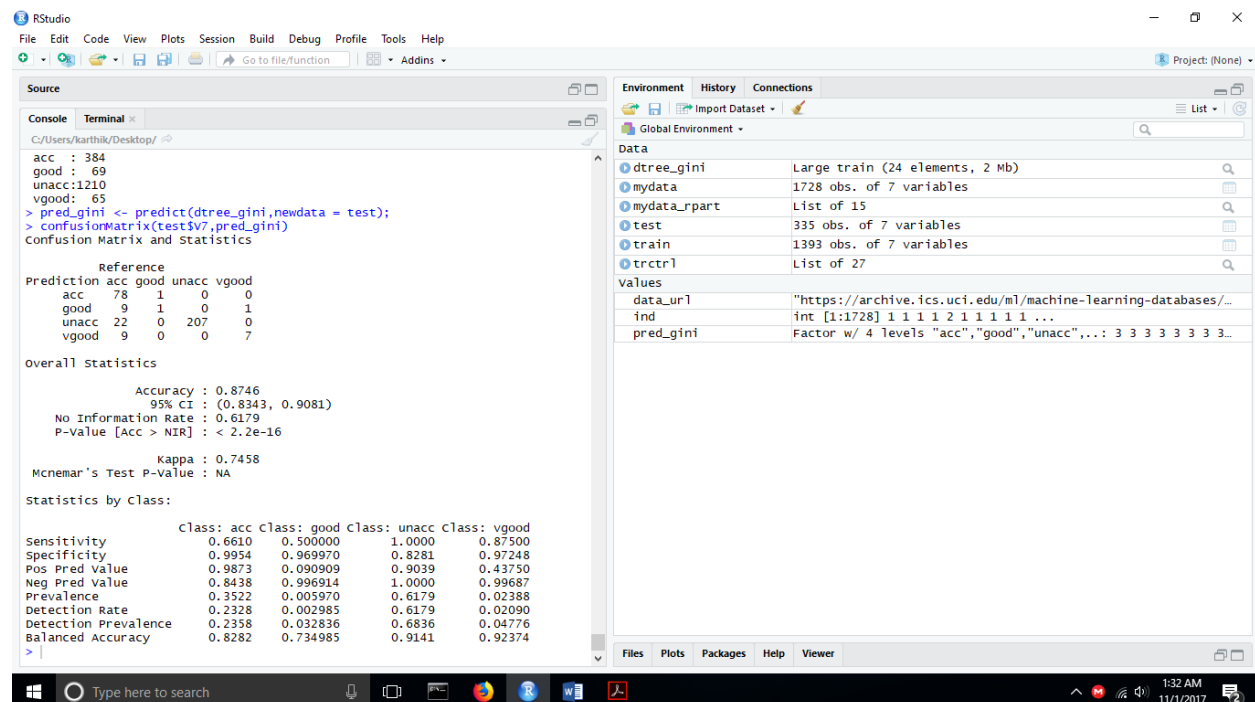
No pre-processing
Resampling: Cross-validated (10 fold, repeated 3 times)
Summary of sample sizes: 1254, 1252, 1254, 1254, 1255, ...
Resampling results across tuning parameters:

cp          Accuracy      Kappa
0.008252427 0.8712356 0.7252525
0.012742718 0.8590480 0.6948274
0.014563107 0.8463568 0.6660218
0.016990291 0.8255842 0.6200872
0.017799353 0.8195922 0.6082217
0.019417476 0.8078705 0.5793315
0.026699029 0.8045286 0.5734743
0.055825243 0.7920720 0.5556534
0.063106796 0.7791614 0.5499026
0.074020146 0.7394361 0.3098540

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.008252427.
> |
```

Confusion Matrix and statistics

```
pred_gini <- predict(dtree_gini,newdata = test);
confusionMatrix(test$V7,pred_gini)
```



```
Source
Console Terminal
C:/Users/karthik/Desktop/

acc : 384
good : 69
unacc:1210
vgood: 65
> pred_gini <- predict(dtree_gini,newdata = test);
> confusionMatrix(test$V7,pred_gini)
Confusion Matrix and Statistics

      Reference
Prediction acc good unacc vgood
acc       78    1    0    0
good      9    1    0    1
unacc     22   20  207    0
vgood      9    0    0    7

Overall Statistics

      Accuracy : 0.8746
      95% CI : (0.8343, 0.9081)
      No Information Rate : 0.6179
      P-value [Acc > NIR] : < 2.2e-16

      Kappa : 0.7458
      McNemar's Test P-value : NA

Statistics by Class:

      Class: acc Class: good Class: unacc Class: vgood
Sensitivity      0.6610 0.500000 1.0000 0.87500
Specificity      0.9954 0.969970 0.8281 0.97248
Pos Pred Value   0.9873 0.990909 0.9039 0.43750
Neg Pred Value   0.8438 0.996914 1.0000 0.99687
Prevalence       0.3522 0.005970 0.6179 0.02388
Detection Rate   0.2328 0.002985 0.6179 0.02090
Detection Prevalence 0.2358 0.032836 0.6836 0.04776
Balanced Accuracy 0.8282 0.734985 0.9141 0.92374
> |
```

Accuracy : 87.46%

The values of fmeasure,recall,precision are as follows:

Precision, Recall and f- measure

```
result<- table(pred_gini,test$V7)

precision <- result[1][1]/(sum(result[1,]));

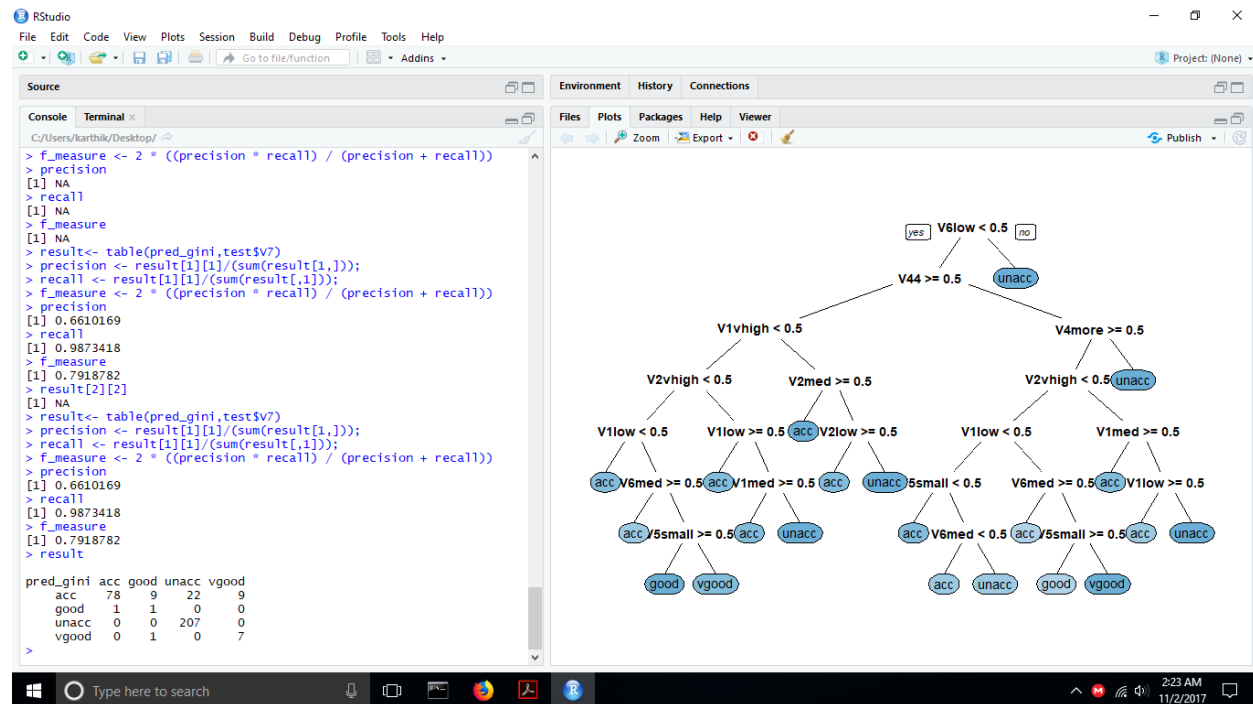
recall <- result[1][1]/(sum(result[,1]));

f_measure <- 2 * ((precision * recall) / (precision + recall))

precision

recall

f_measure
```



10 fold cross validation

```
set.seed(1234)

mydata<-mydata[sample(nrow(mydata)),]

#Create 10 equally size folds
folds <- cut(seq(1,nrow(mydata)),breaks=10,labels=FALSE)

#Perform 10 fold cross validation
for(i in 1:10){
  #Segement your data by fold using the which() function
  testIndexes <- which(folds==i,arr.ind=TRUE)
  test <- mydata[testIndexes, ]
```

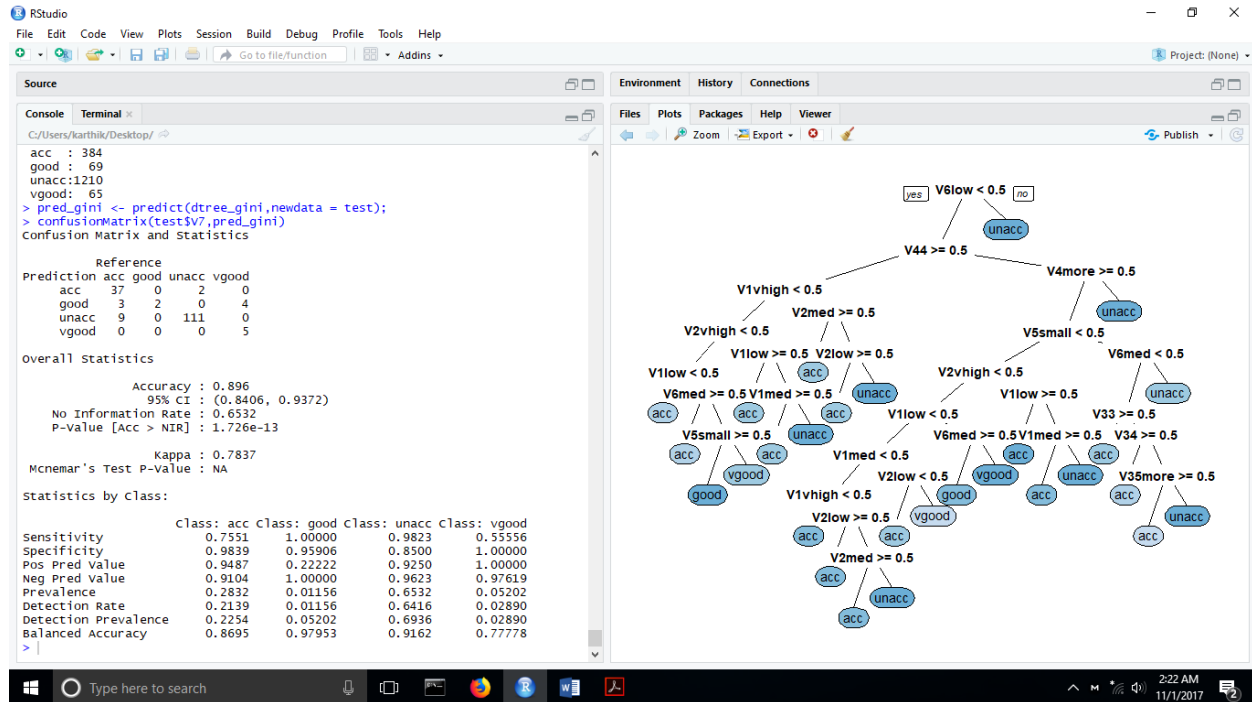
```

train <- mydata[-testIndexes, ]

#Use the test and train data partitions however you desire...
}

```

We use this code for 10 fold cross validation



Now the accuracy is 89.6%

This classifier performs better with case 2 i.e 10 fold cross validation

ROC Curve

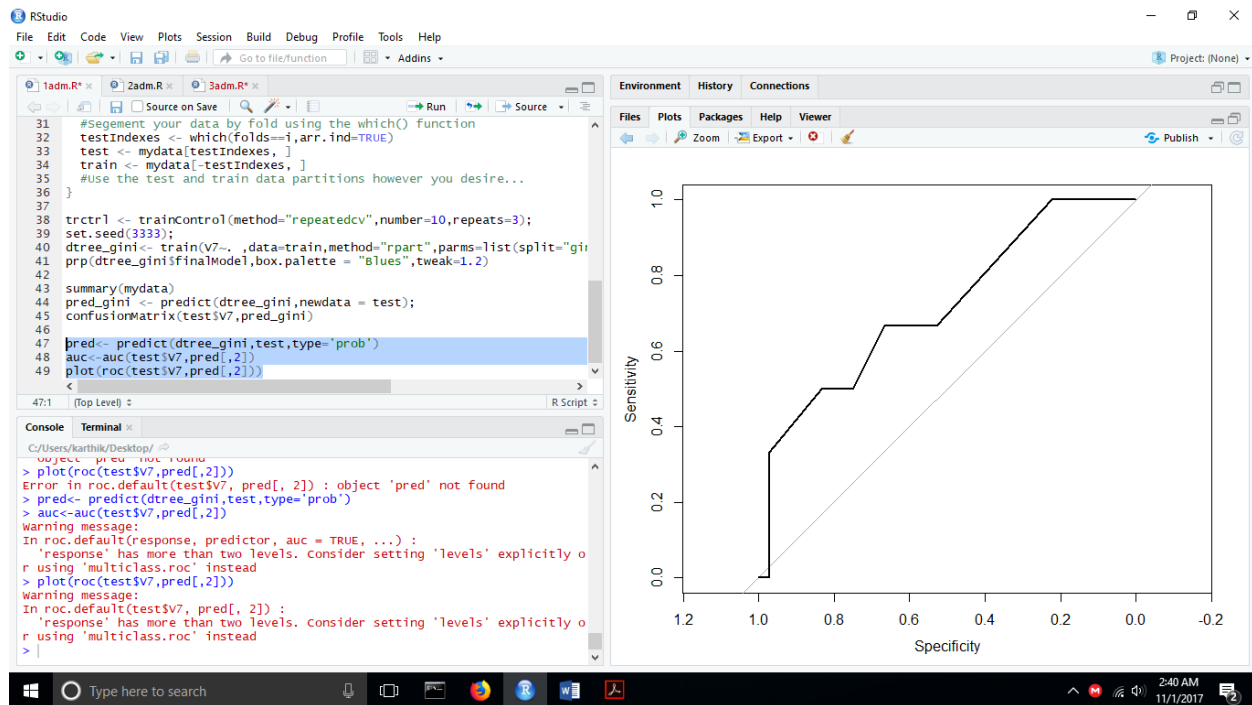
Install pROC library before proceeding

```
pred<- predict(dtree_gini,test,type='prob')
```

```
auc<-auc(test$V7,pred[,2])
```

```
plot(roc(test$V7,pred[,2]))
```

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two class groups.



Area under the curve: 0.8621

1 b. Decision Tree with Entropy as the impurity measure

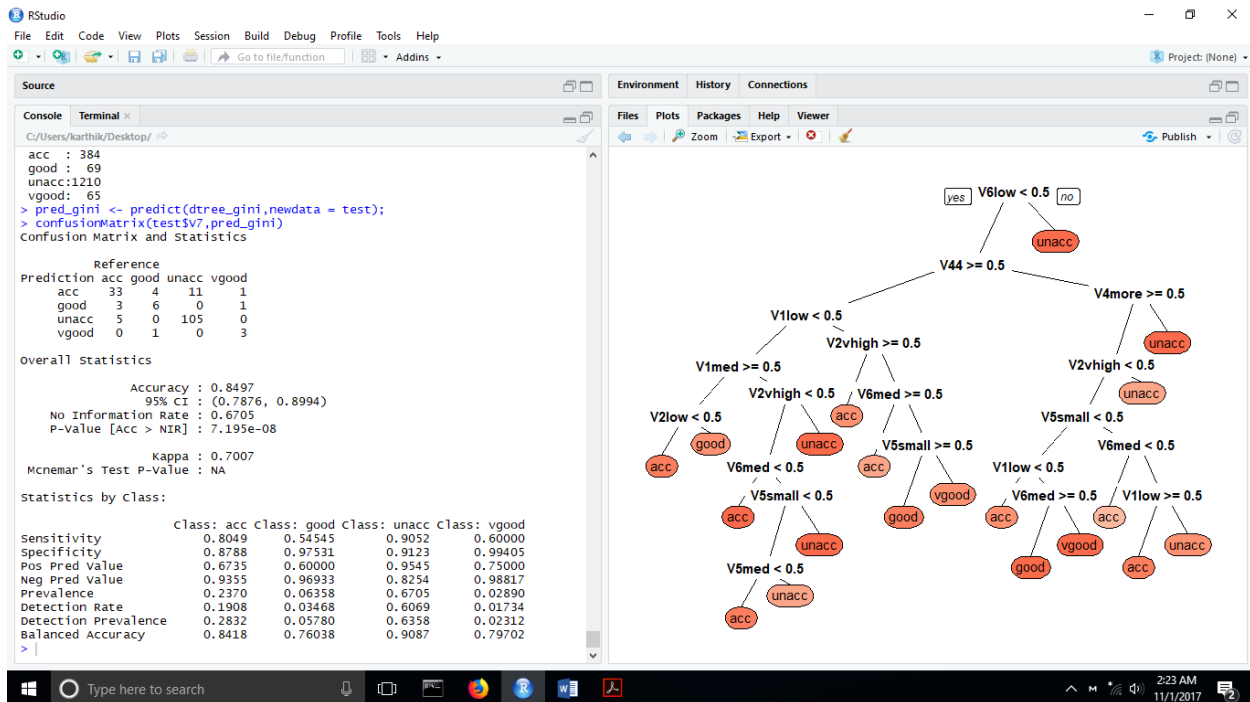
We now simulate the steps in 1 a and change the statement to obtain the impurity measure

```

dtree_gini<- train(university ~.
,data=train,method="rpart",parms=list(split="gini"),trControl=trctrl,tuneLeng
th=10)

```

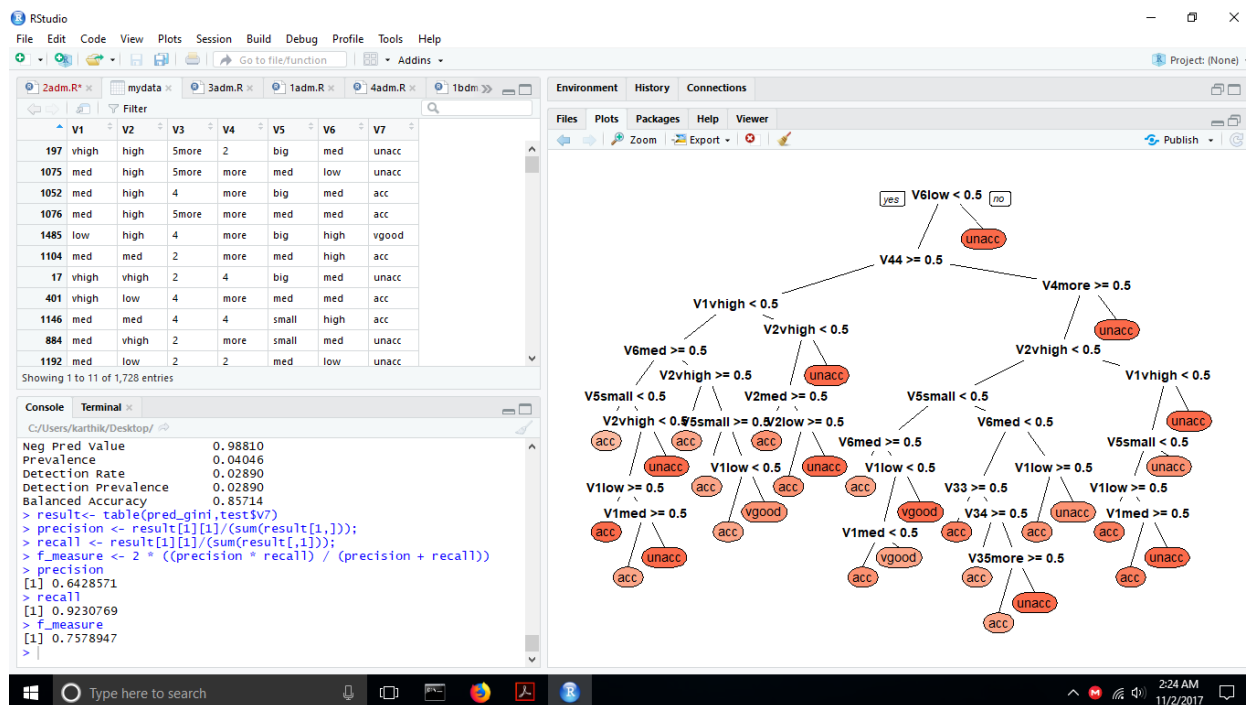

Using 10 fold cross validation similar to 1 a



The accuracy now is 84.97%.

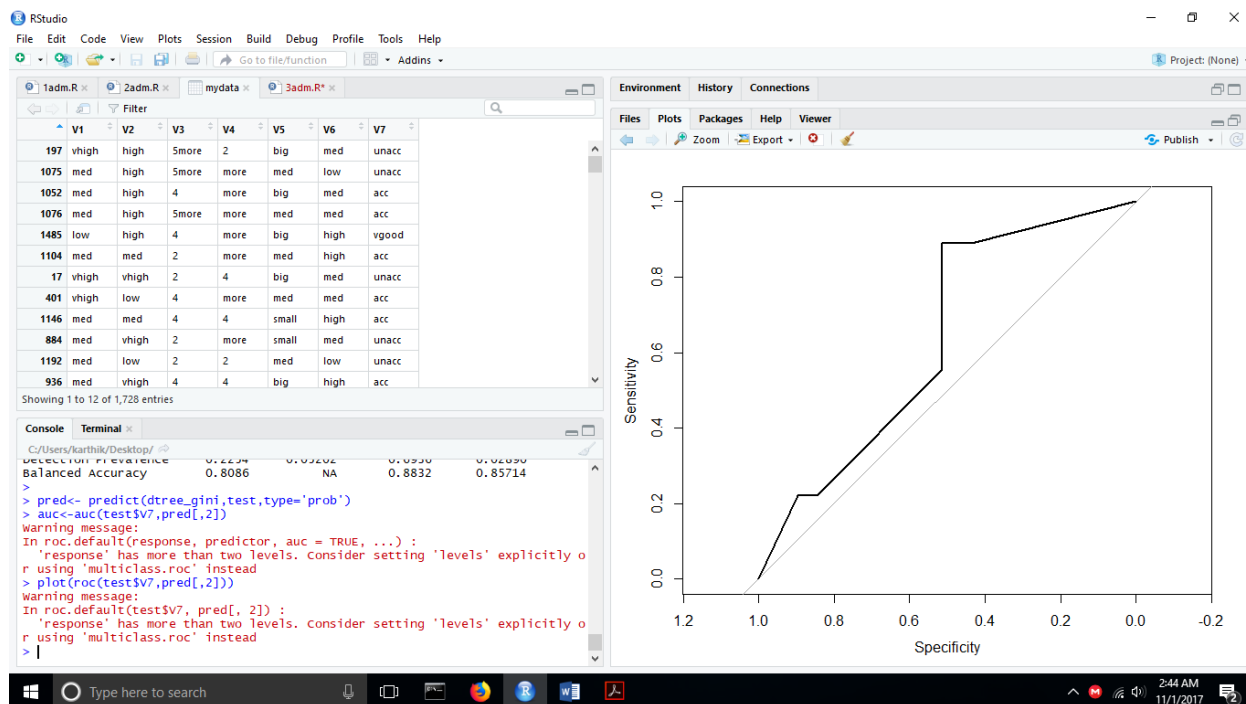
This classifier performs better with case 1 i.e random sampling

Precision, Recall and f- measure



ROC Curve

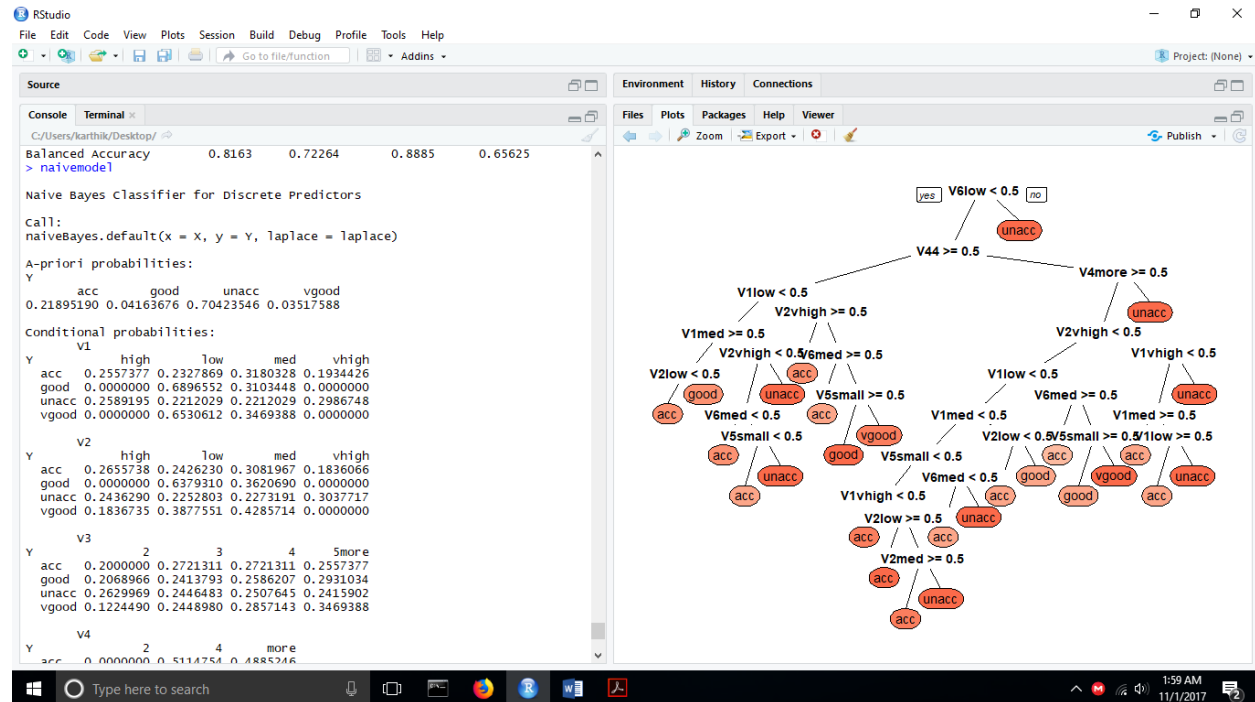
Find the ROC curve similar to 1 a



1 c. Naïve Bayesian Classifier

We now install and include the libraries `rminer` and `e1071`.

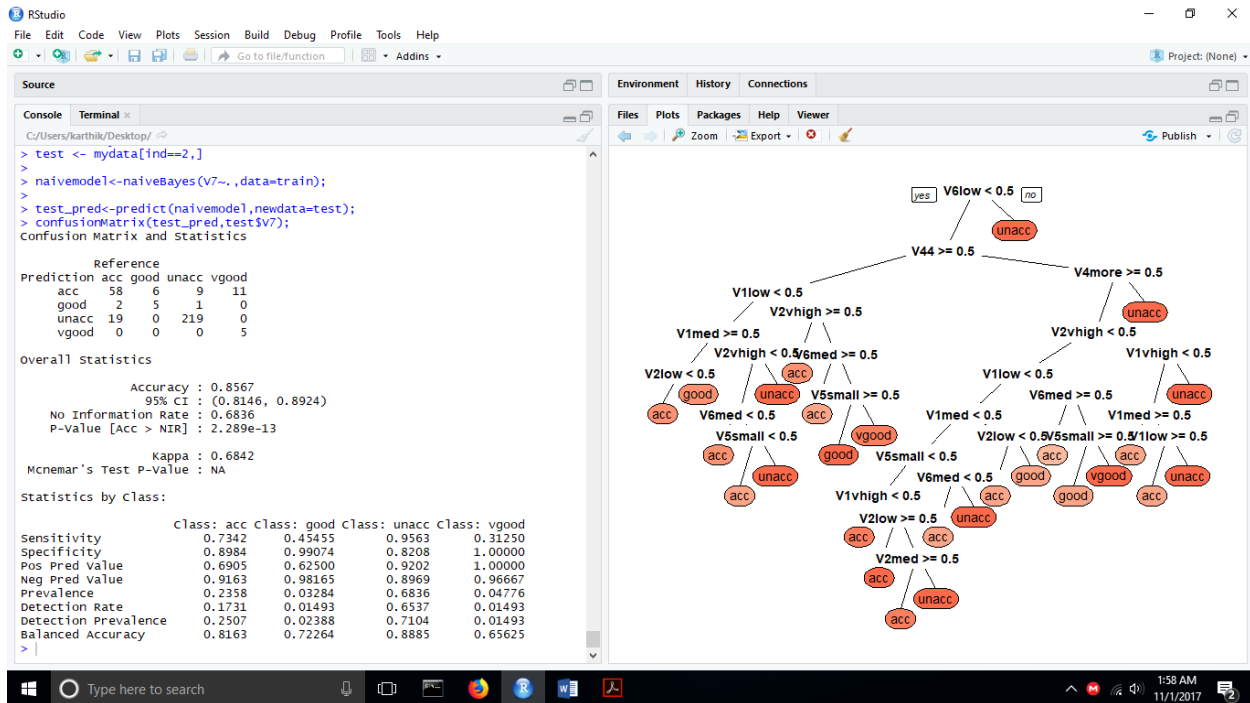
```
naivemodel<-naiveBayes(V7~.,data=train);
```



```
test_pred<-predict(naivemodel,newdata=test);
```

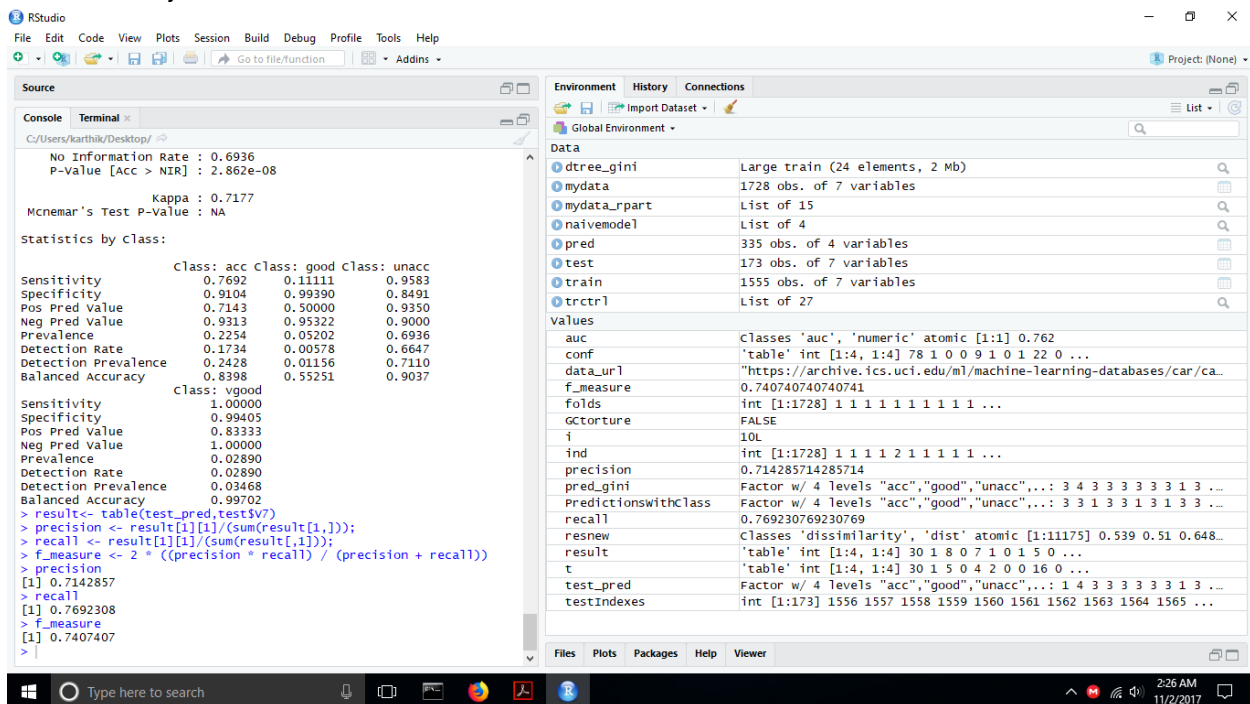
```
confusionMatrix(test_pred, test$V7);
```

Confusion Matrix and statistics:

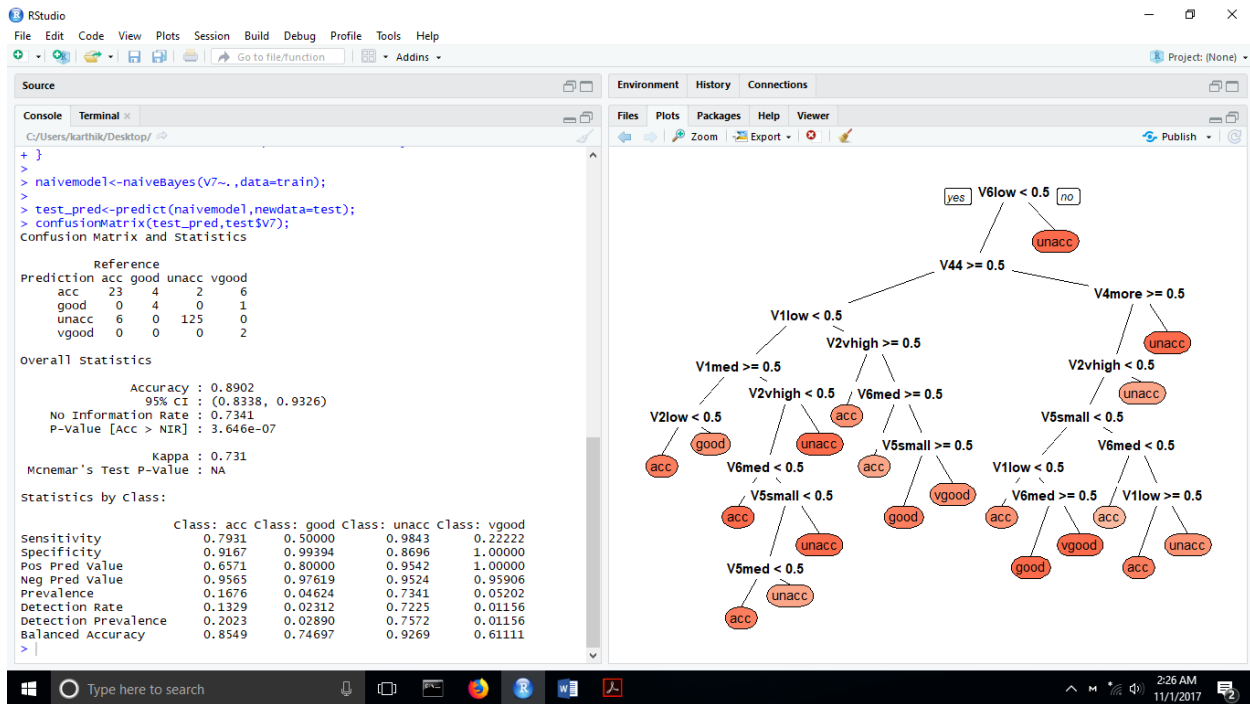


Accuracy: 85.6%

Precision, Recall and f- measure



Using 10 fold cross validation similar to 1 a



Accuracy now : 89.02%

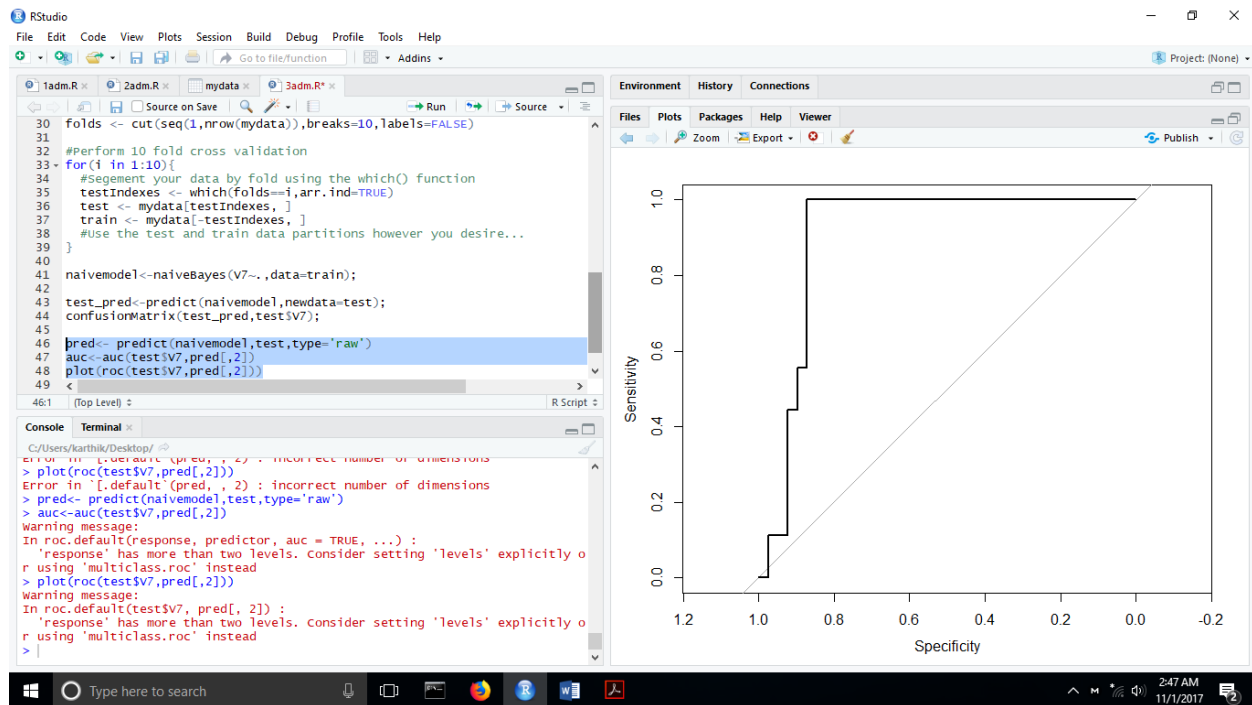
This classifier performs better with case 2 i.e 10 fold cross validation

ROC Curve

```
pred<- predict(naivemodel,test,type='raw')
```

```
auc<-auc(test$V7,pred[,2])
```

```
plot(roc(test$V7,pred[,2]))
```



Area under the curve: 0.9031

1.d Artificial Neural Network – with and without hidden layers

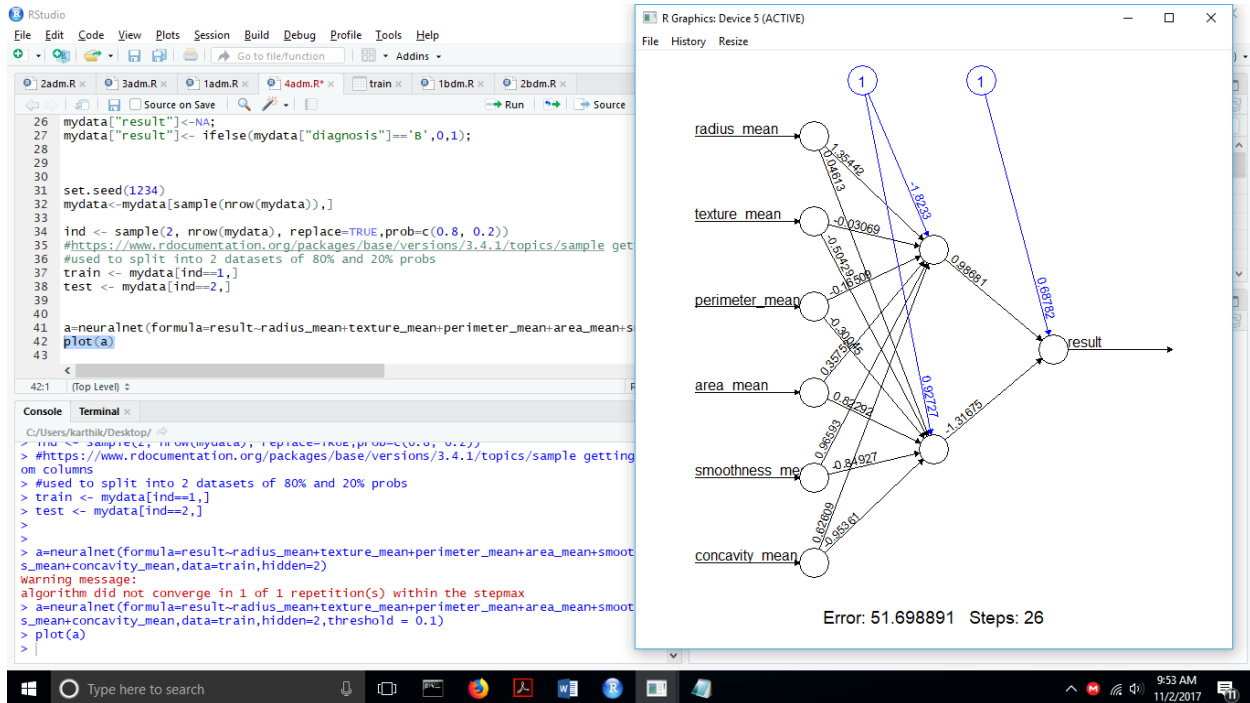
A change in dataset as artificial neural network requires numerical data to compute weights;

Dataset-url: <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/data>

Include the library neuralnet

Neural Network with two hidden layer:

```
a=neuralnet(formula=result~radius_mean+texture_mean+perimeter_mean+area_mean+smoot
hness_mean+concavity_mean,data=train,hidden=2)
```

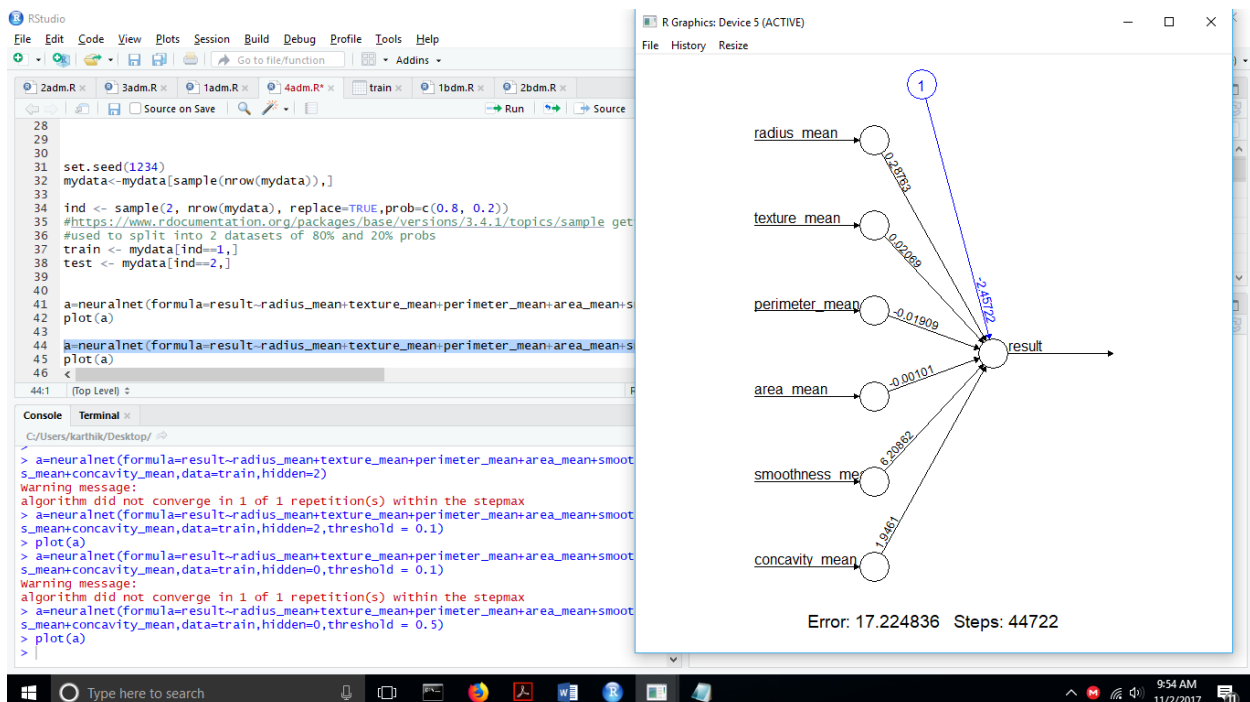


Neural Network with no hidden layer:

```

a=neuralnet(formula=result~radius_mean+texture_mean+perimeter_mean+area_mean+smoothness_mean+concavity_mean,data=train,hidden=0,threshold = 0.1)

```



Neural Network with two hidden layer:

