

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

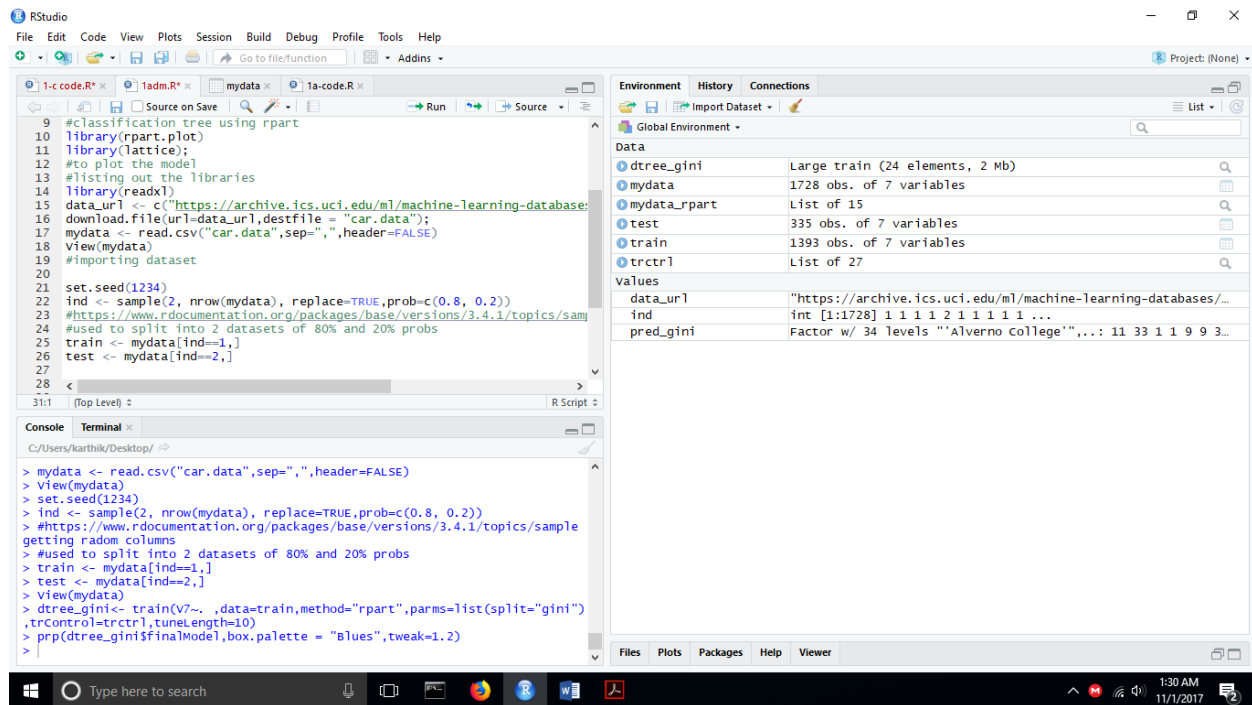
Loading the data

```
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,]
```

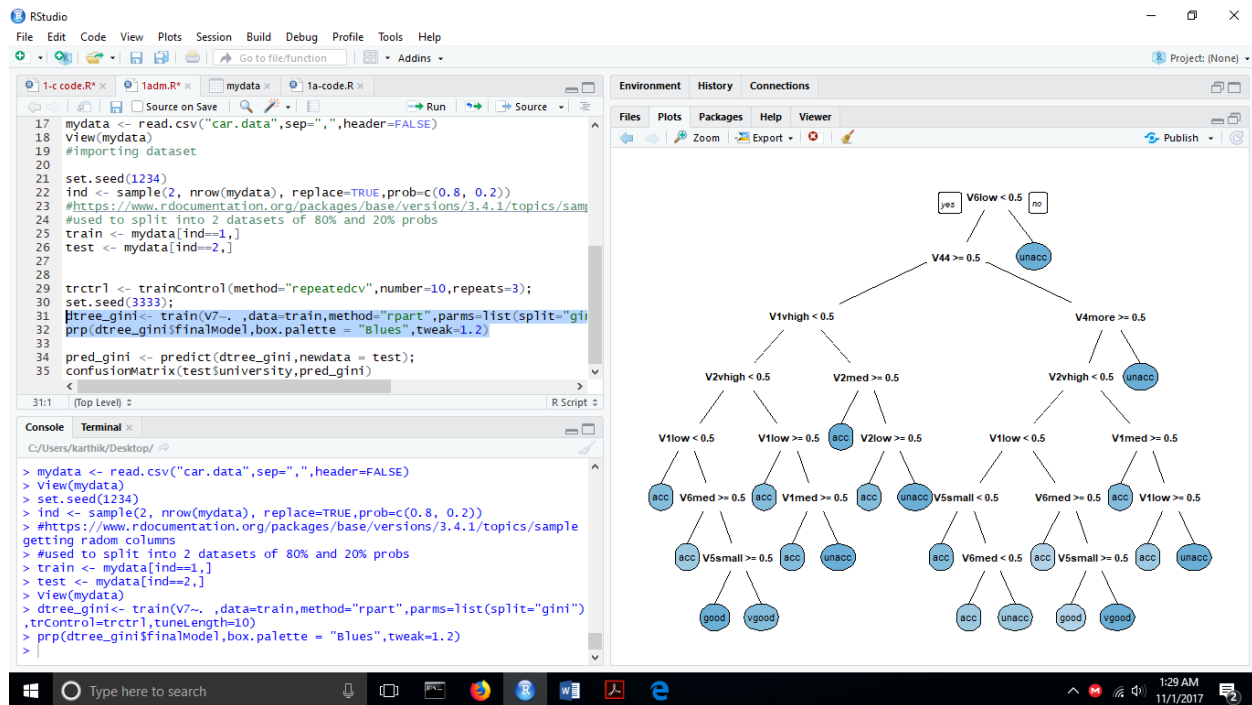


Training the dataset using the caret library based on decision on gini index on university column

```
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)
```

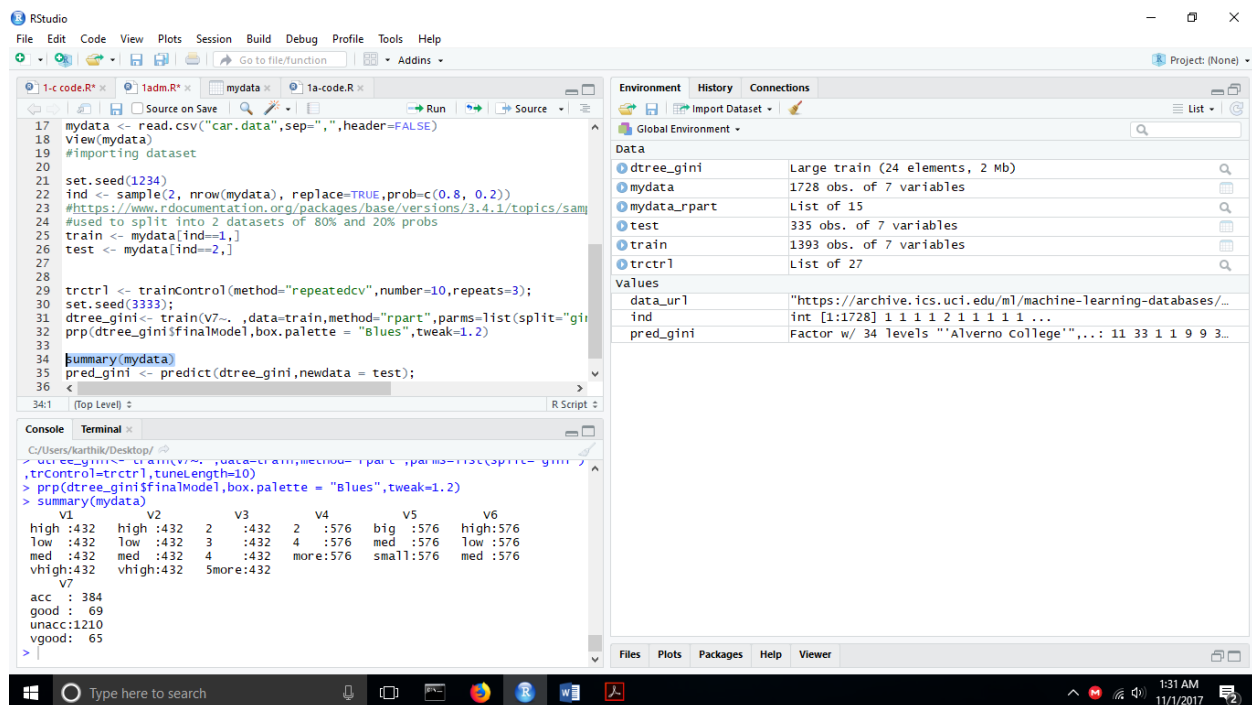
Displaying the dataset

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



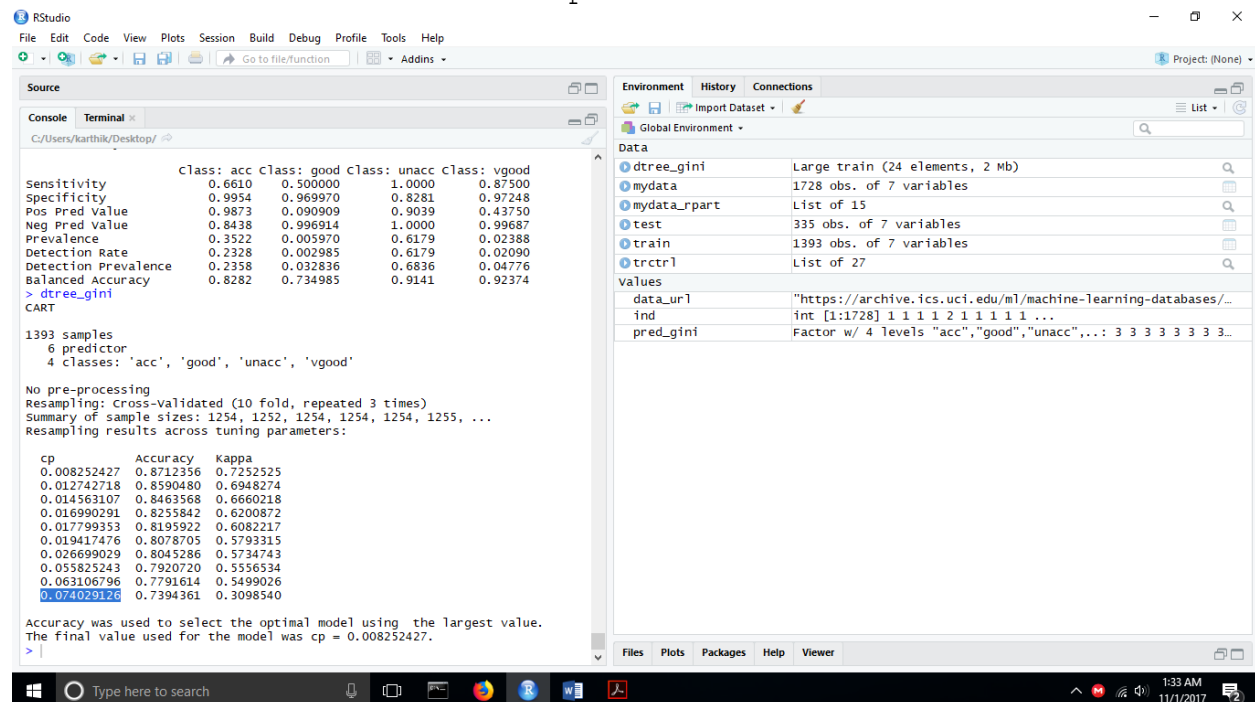
We can check if our dataset contains any missing values using the `anyNA(dataset)` function

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 x
C:/Users/karthik/Desktop/

Sensitivity      Class: acc Class: good Class: unacc Class: vgood
Specificity      0.6610 0.500000 1.0000 0.87500
Pos Pred Value   0.9954 0.969970 0.8281 0.97248
Neg Pred Value   0.9873 0.990909 0.9039 0.43750
Prevalence       0.8438 0.996914 1.0000 0.99687
Detection Rate   0.3522 0.005970 0.6179 0.02388
Detection Rate   0.2328 0.002985 0.6179 0.02090
Detection Rate   0.2358 0.032836 0.6836 0.04776
Balanced Accuracy 0.2882 0.734985 0.9141 0.92374

> dtree_gini
CART

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

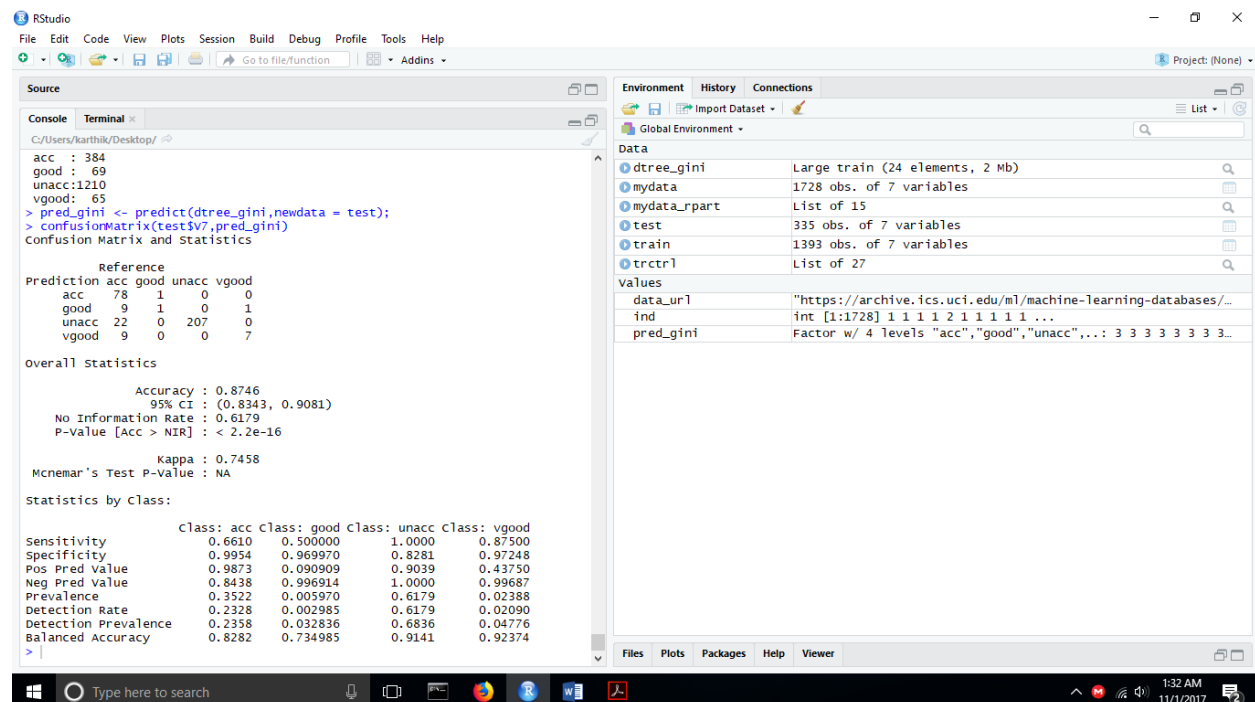
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1254, 1252, 1254, 1254, 1255, ...
Resampling results across tuning parameters:

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 x
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    0    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 Rate   0.2358 0.032836 0.6836 0.04776
Balanced Accuracy 0.2882 0.734985 0.9141 0.92374
> |
```

Accuracy : 87.46%

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

Precision, Recall and f- measure

They are calculated using:

```
result <- confusionMatrix(data = test_pred, test$PAID_FINE)
precision <- result$byClass['Pos Pred Value']
recall <- result$byClass['Sensitivity']
f_measure <- 2 * ((precision * recall) / (precision + recall))
```

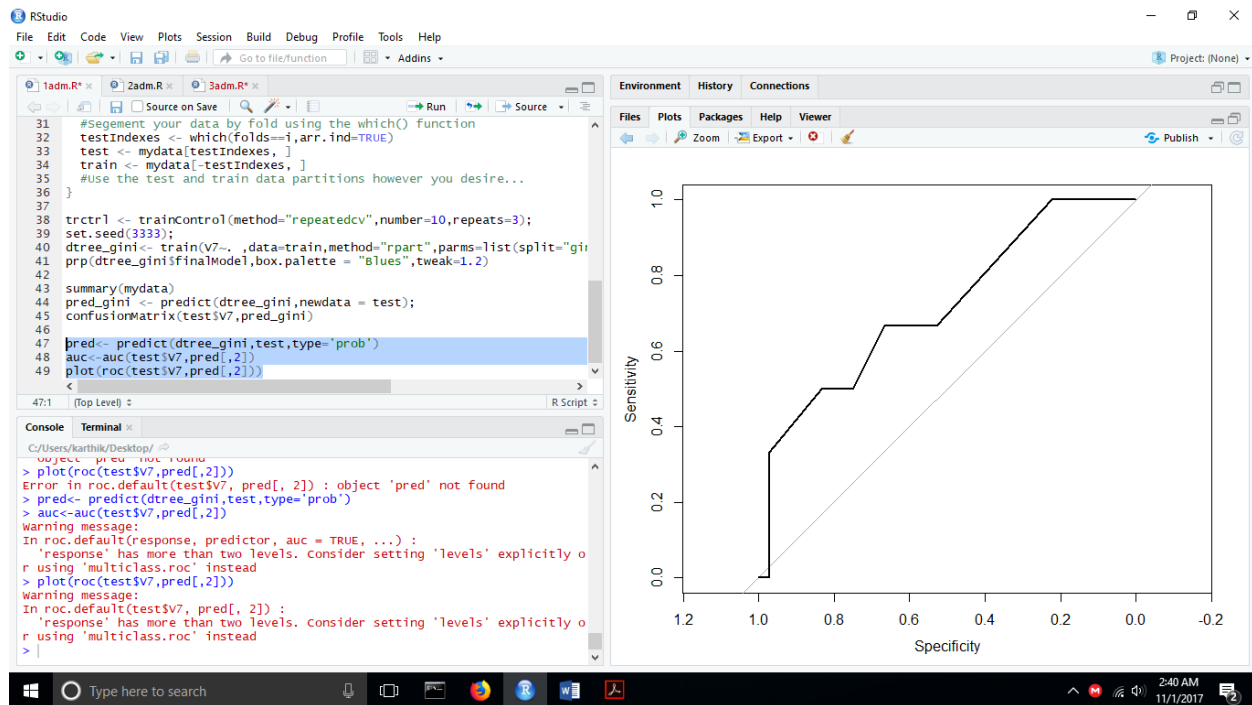
10 fold cross validation

```
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){
  #Segment 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



Area under the curve: 0.8621

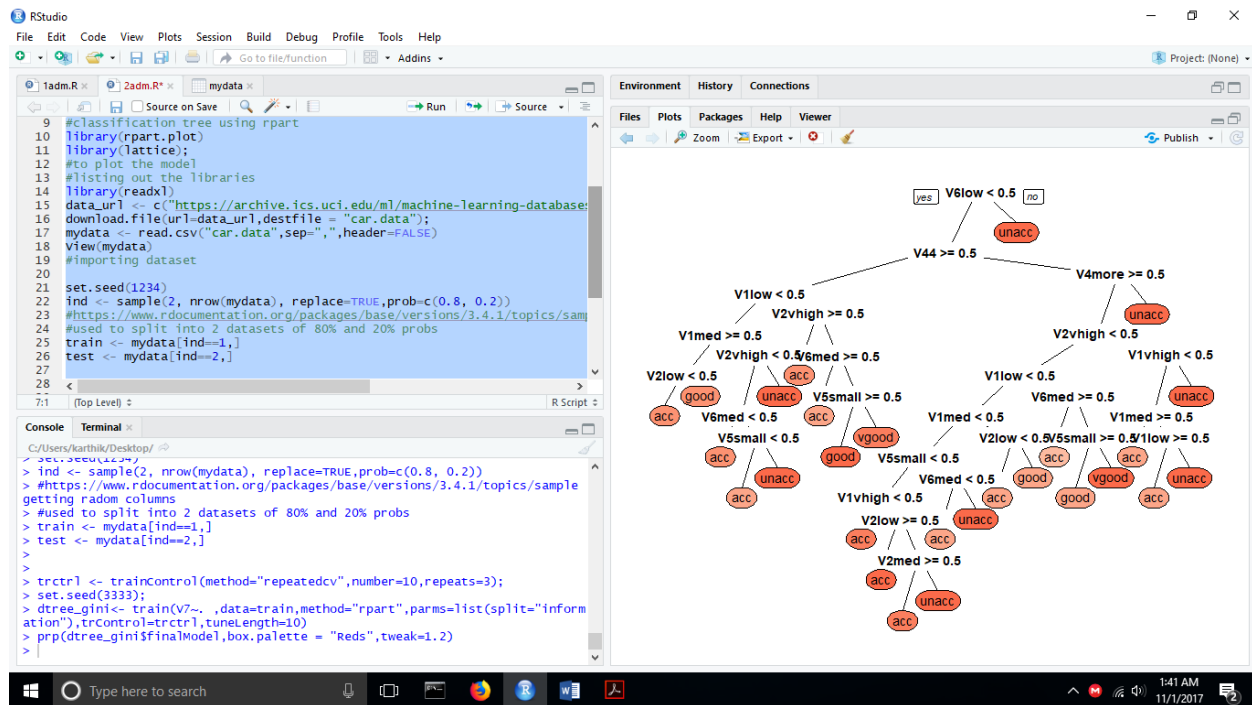
1 b. Decision Tree with Entropy as the impurity measure

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

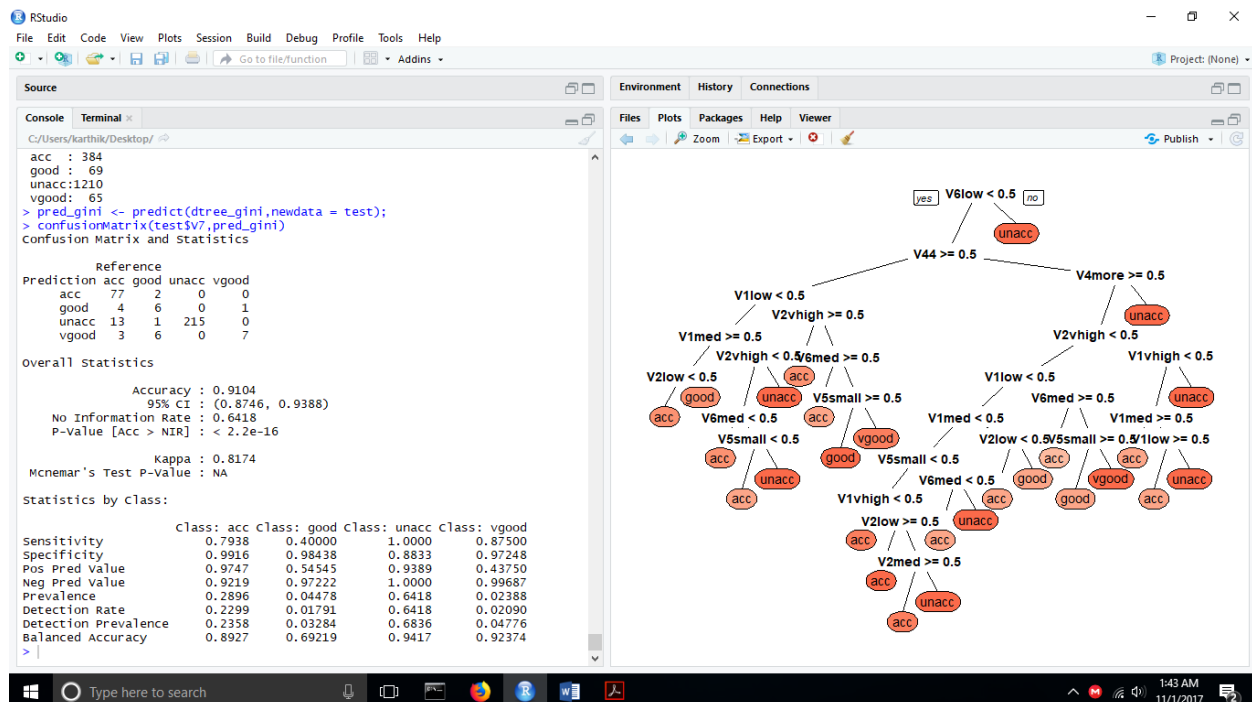
```

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

```

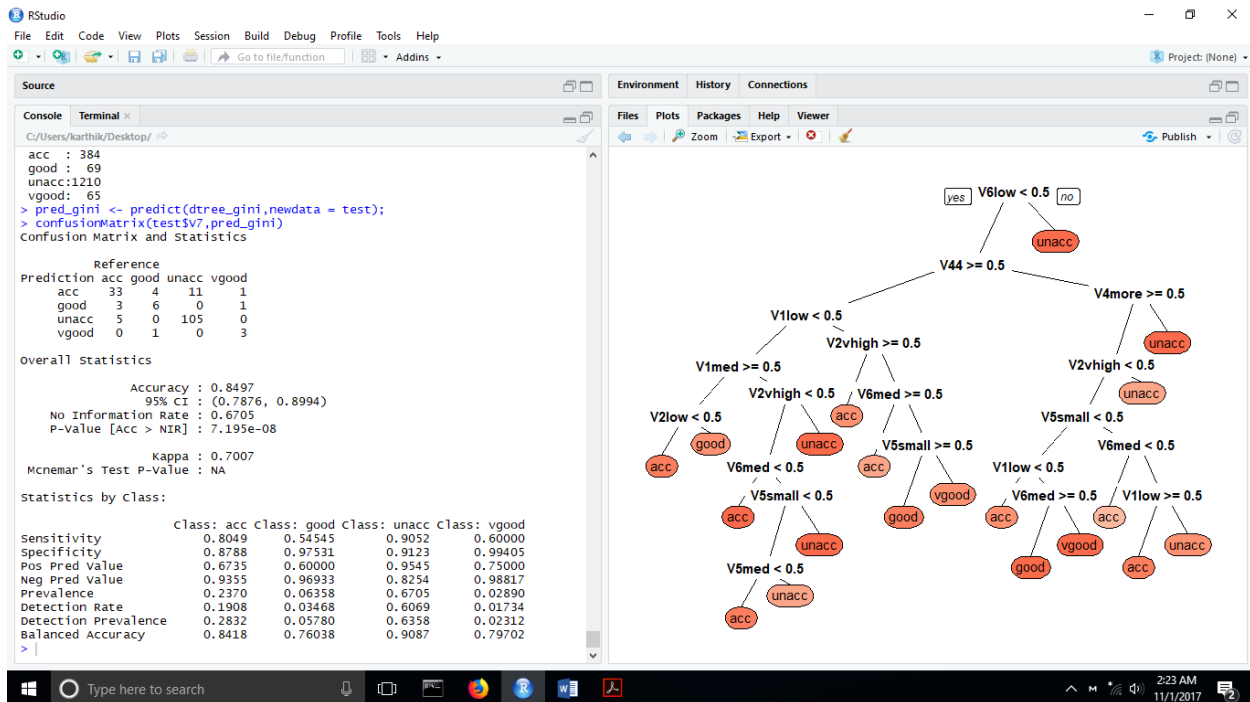


Confusion matrix and Statistics



Accuracy: 91.04%

Using **10 fold cross validation** similar to 1 a

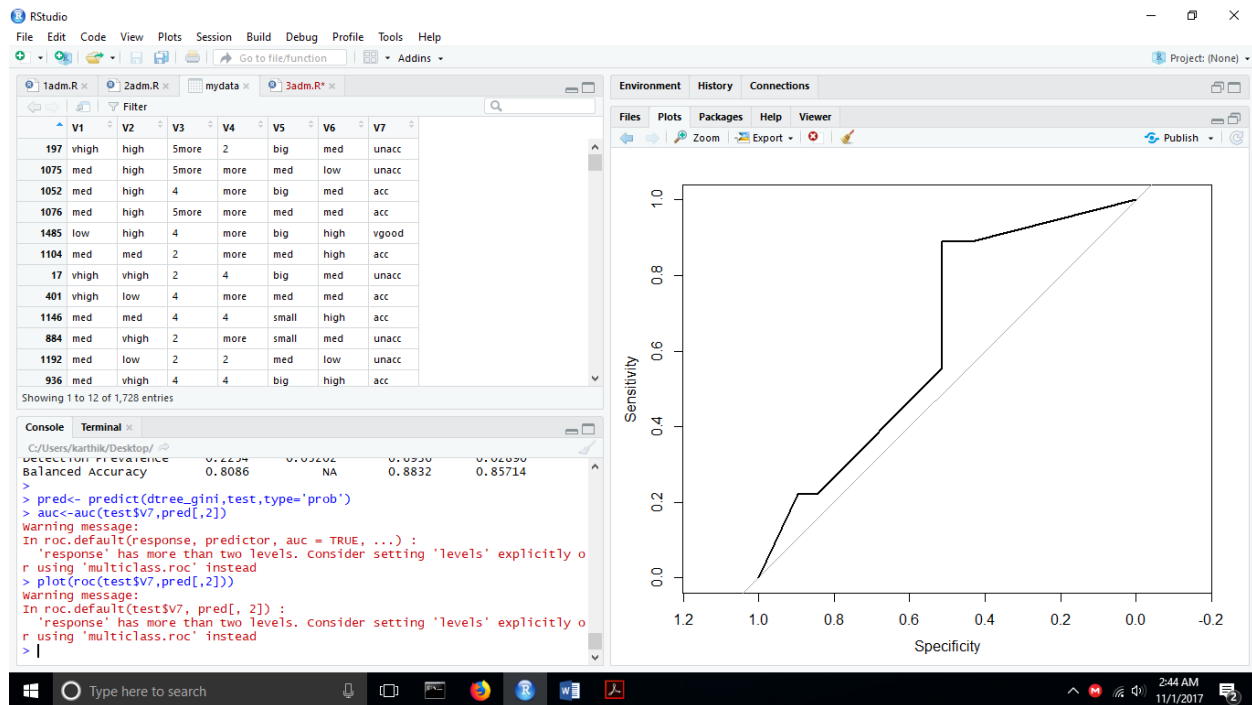


The accuracy now is 84.97%.

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

ROC Curve

Find the ROC curve similar to 1 a

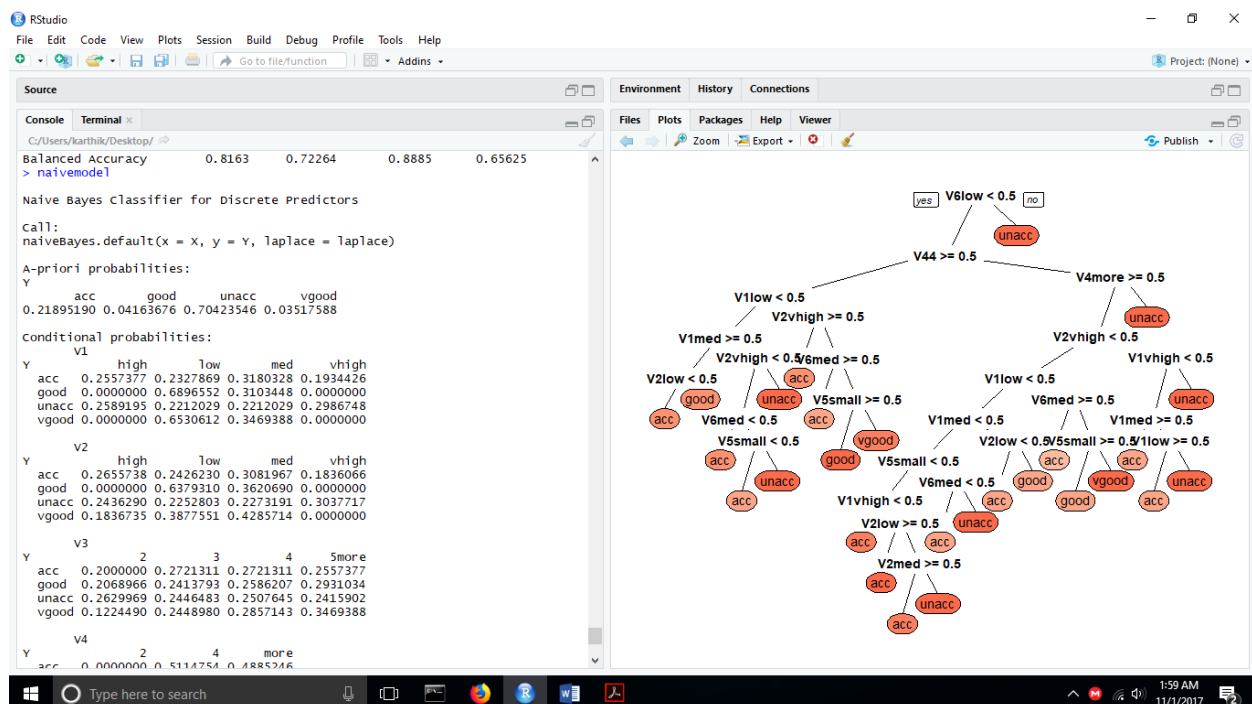


Area under the curve: 0.8694

1 c. Naïve Bayesian Classifier

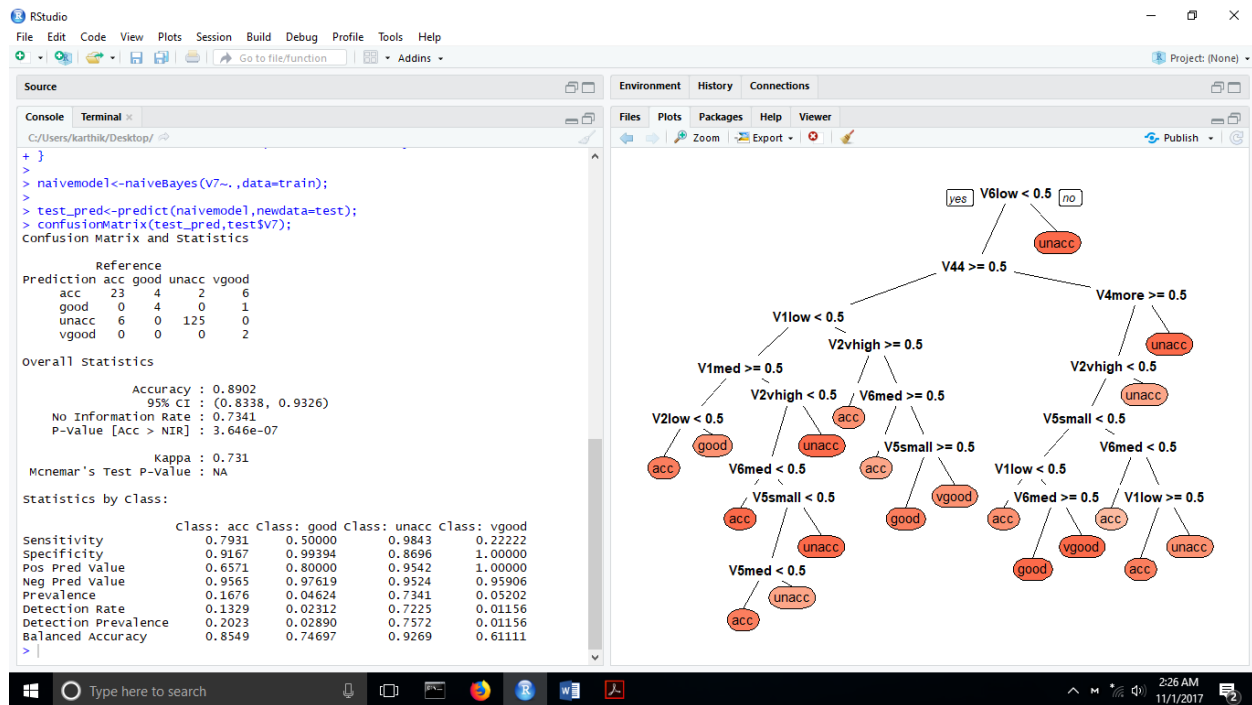
We now install and include the libraries rminer and e1071.

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



Confusion Matrix and statistics:





Accuracy now : 89.02%

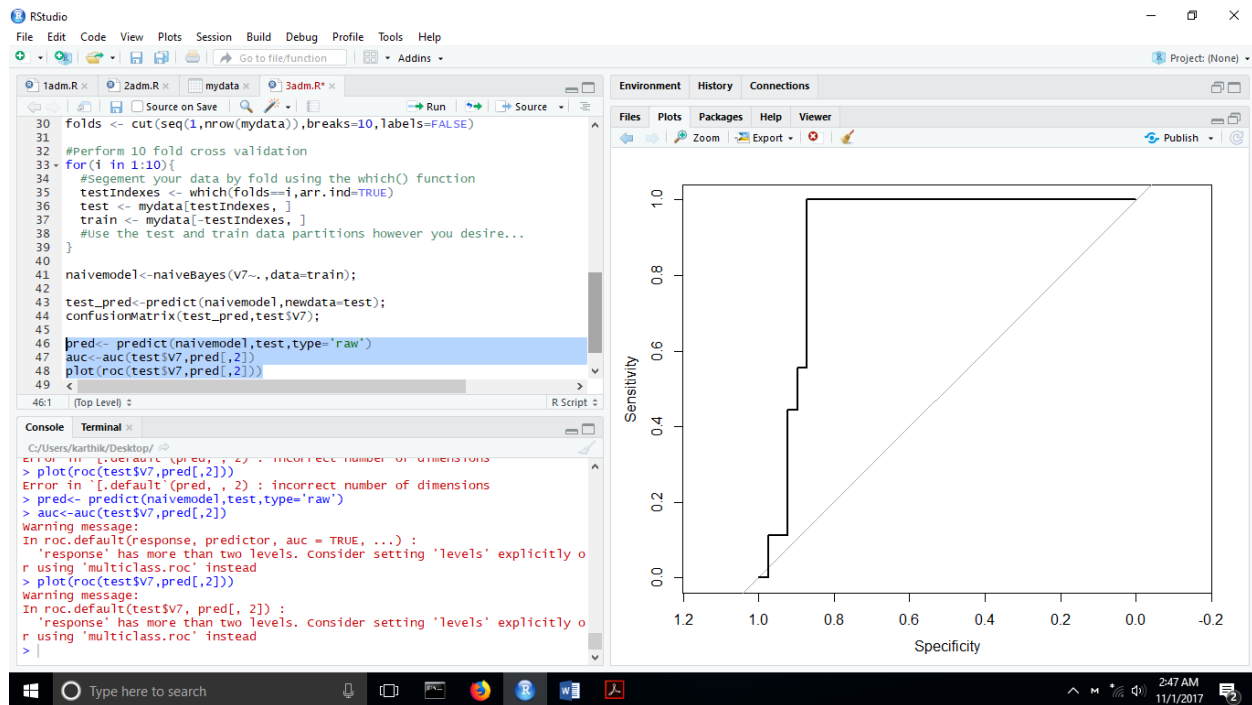
This classifier performs better with case 2 i.e 10 fold cross validation

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

Neural Network with two hidden layer: