Laboratory #1 Classification

Table of Contents

The steps for you to follow to do this include:

1. Using the datasets provided run each of the three different machine learning techniques as described in this laboratory documentation. This will provide you with familiarity of each technique. This counts for roughly 50% of the points available for this laboratory.
2. Next, understanding how each of these techniques works use each of the three techniques to evaluate the online news popularity dataset. Note that this will involve some significant data "munging" or pre-processing to get the dataset in a form that will work. That is, each technique may require you to pre-process the dataset in a different way.
3. Generate your written laboratory report on your work. Use R markdown to generate the result and submit the knitted file (html, word or pdf) or upload it on R pubs.

Method #1. Tree-based classification

(MLR pg 128)

Step 1: Collecting the data

The data set used for this method is the credit.csv file. This data set is available for you on Moodle. In addition, you can check it out at the UCI Data Repository at http://archive.ics.uci.edu/ml/. Before you use a data set you should check on its description to see how big the data set is, what format it is in, what processing has already been done, and to determine if there is anything "quirky" about it that you need to know in order to conduct your

analysis. For example, the class variable "default" is the 17th column, or variable, out of the 17 in this dataset. We'll use this knowledge when we create our model later. For now, use the following command to read it into R.

```
> credit <- read.csv("credit.csv")
```

Use the structure command to check what is in the credit object created to hold the data in R. The str command is shown below with the first few lines returned.

```
> str(credit)
'data.frame':  1000 obs. of  21 variables:
 $ Creditability                   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Account.Balance                 : int  1 1 2 1 1 1 1 1 4 2 ...
 $ Duration.of.Credit..month.      : int  18 9 12 12 12 10 8 6 18 24 ...
 $ Payment.Status.of.Previous.Credit: int  4 4 2 4 4 4 4 4 4 2 ...
 $ Purpose                         : int  2 0 9 0 0 0 0 0 3 3 ...
 $ Credit.Amount                   : int  1049 2799 841 2122 2171 2241 3398
1361 1098 3758 ...
 $ Value.Savings.Stocks            : int  1 1 2 1 1 1 1 1 1 3 ...
 $ Length.of.current.employment    : int  2 3 4 3 3 2 4 2 1 1 ...
 $ Instalment.per.cent             : int  4 2 2 3 4 1 1 2 4 1 ...
 $ Sex...Marital.Status            : int  2 3 2 3 3 3 3 3 2 2 ...
 $ Guarantors                      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Duration.in.Current.address     : int  4 2 4 2 4 3 4 4 4 4 ...
 $ Most.valuable.available.asset   : int  2 1 1 1 2 1 1 1 3 4 ...
 $ Age..years.                     : int  21 36 23 39 38 48 39 40 65 23 ...
 $ Concurrent.Credits              : int  3 3 3 3 1 3 3 3 3 3 ...
 $ Type.of.apartment               : int  1 1 1 1 2 1 2 2 2 1 ...
 $ No.of.Credits.at.this.Bank      : int  1 2 1 2 2 2 2 1 2 1 ...
 $ Occupation                      : int  3 3 2 2 2 2 2 2 1 1 ...
 $ No.of.dependents                : int  1 2 1 2 1 2 1 2 1 1 ...
 $ Telephone                       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Foreign.Worker                  : int  1 1 1 2 2 2 2 2 1 1 ...
```

Step 2: Exploring the data

There are many ways you can further check the data in this object, e.g. summary or table. For example, you can use the summary() command to get information about loan amounts requested as follows (keep in mind that the monetary unit is Deutsch Marks (DM)). The table lists the number of loans that were defaulted or not.

```
> summary(credit$Credit.Amount)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    250    1366    2320    3271    3972   18420
> table(credit$Creditability)

 No(1)   yes(0)
700       300
```

In order to develop the tree-based classification model you need to split the data into training and test records. But before we do that this particular data set has not been randomized so we can do a good analysis with it. Use the following to randomize the 1000 observations (records) in this data set.

```
> set.seed(12345)
> credit_rand <- credit[order(runif(1000)), ]
```

You can and should check to make sure that randomizing your data has not made any substantive changes to it, i.e. the means should still be the same and so on. You can check the summary on the original data to the summary using the new object you created to do this.

```
> summary(credit$ Credit.Amount)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    250    1366    2320    3271    3972   18424
```

You'll need to subset the observations (records) to establish the training set and the test set. Sort of a rule of thumb is to use between 75% and 90% of the records for the training set. There are many ways to do this in R. For example, you can use:

```
> credit_train <- credit_rand[1:900, ]
> credit_test <- credit_rand[901:1000, ]
```

If the randomization went well then the percentages between splits should be close. Before we checked on the number of defaults using the table() command. You can check the percentages rather than actual numbers for both the training set and the test set using:

```
> prop.table(table(credit_train$ Creditability))

        0         1
0.3088889 0.6911111
 > prop.table(table(credit_test$ Creditability))

   0    1
0.22 0.78
```

Step 3: Training a model on the data

You'll need the C5.0 algorithm to complete this step. If you haven't already you'll need to install the C50 package. To start you'll need to attach it to this session using:

```
> library(C50)
```

You only need one command line to create the decision tree model. However, now you need to set-up this command so that the algorithm knows that the class or response variable is in the 1st column.

```
> credit_model <- C5.0(x = credit_train[-1], y = credit_train$Creditability)
```

The R command and the model created is:

```
> credit_model

Call:
C5.0.default(x = credit_train[-1], y = credit_train$Creditability)


## Account.Balance > 2:
## :...Concurrent.Credits > 2:
## :   :...Age..years. > 33: 1 (179/11)
## :   :   Age..years. <= 33:
## :   :   :...Credit.Amount > 6681:
## :   :       :...Length.of.current.employment <= 2: 0 (4)
## :   :       :   Length.of.current.employment > 2:
## :   :       :   :...Payment.Status.of.Previous.Credit <= 3: 1 (4)
## :   :       :       Payment.Status.of.Previous.Credit > 3: 0 (3/1)
## :   :       Credit.Amount <= 6681:
## :   :       :...Occupation > 2:
## :   :           :...Occupation <= 3: 1 (120/12)
## :   :           :   Occupation > 3:
## :   :           :   :...Duration.of.Credit..month. <= 33: 1 (9)
## :   :           :       Duration.of.Credit..month. > 33: 0 (3)
## :   :           Occupation <= 2:
## :   :           :...No.of.Credits.at.this.Bank > 1: 1 (6)
## :   :               No.of.Credits.at.this.Bank <= 1:
## :   :               :...Most.valuable.available.asset > 1: 0 (3)
## :   :                   Most.valuable.available.asset <= 1:
## :   :                   :...Credit.Amount <= 1987: 1 (8/1)
## :   :                       Credit.Amount > 1987: 0 (2)
## :   Concurrent.Credits <= 2:
## :   :...Guarantors > 1: 1 (4)
## :       Guarantors <= 1:
## :       :...Purpose <= 0:
## :           :...Most.valuable.available.asset <= 2: 0 (5)
## :           :   Most.valuable.available.asset > 2:
## :           :   :...No.of.dependents <= 1: 1 (7/1)
## :           :       No.of.dependents > 1: 0 (2)
## :           Purpose > 0:
## :           :...Purpose <= 4: 1 (35/2)
## :               Purpose > 4:
## :               :...Length.of.current.employment <= 2: 0 (4)
## :                   Length.of.current.employment > 2:
## :                   :...No.of.dependents > 1: 0 (3/1)
## :                       No.of.dependents <= 1:
## :                       :...Length.of.current.employment > 3: 1 (4)
## :                           Length.of.current.employment <= 3:
## :                           :...Instalment.per.cent <= 2: 1 (2)
## :                               Instalment.per.cent > 2: 0 (2)
## Account.Balance <= 2:
## :...Payment.Status.of.Previous.Credit <= 1:
##     :...Value.Savings.Stocks <= 2: 0 (49/10)
##     :   Value.Savings.Stocks > 2:
```

```
##     :   :...Credit.Amount <= 2064: 0 (3)
##     :       Credit.Amount > 2064: 1 (9/1)
##     Payment.Status.of.Previous.Credit > 1:
##     :...Credit.Amount > 7980:
##         :...Value.Savings.Stocks > 4:
##         :   :...Payment.Status.of.Previous.Credit <= 2: 0 (4/1)
##         :   :   Payment.Status.of.Previous.Credit > 2: 1 (3)
##         :   Value.Savings.Stocks <= 4:
##         :   :...Account.Balance > 1: 0 (15)
##         :       Account.Balance <= 1:
##         :       :...Concurrent.Credits <= 2: 0 (2)
##         :           Concurrent.Credits > 2:
##         :           :...Credit.Amount <= 10297: 0 (6)
##         :               Credit.Amount > 10297: 1 (3)
##         Credit.Amount <= 7980:
##         :...Duration.of.Credit..month. <= 11:
##             :...Occupation > 3:
##             :   :...Concurrent.Credits <= 2: 1 (3)
##             :   :   Concurrent.Credits > 2:
##             :   :   :...Payment.Status.of.Previous.Credit <= 2: 1 (4/1)
##             :   :       Payment.Status.of.Previous.Credit > 2: 0 (3)
##             :   Occupation <= 3:
##             :   :...Age..years. > 32: 1 (34)
##             :       Age..years. <= 32:
##             :       :...Most.valuable.available.asset <= 1: 1 (13/1)
##             :           Most.valuable.available.asset > 1:
##             :           :...Instalment.per.cent <= 3: 1 (6/1)
##             :               Instalment.per.cent > 3: 0 (6/1)
##             Duration.of.Credit..month. > 11:
##             :...Duration.of.Credit..month. > 36:
##                 :...Length.of.current.employment <= 1: 1 (3)
##                 :   Length.of.current.employment > 1:
##                 :   :...No.of.dependents > 1: 1 (5/1)
##                 :       No.of.dependents <= 1:
##                 :       :...Duration.in.Current.address <= 1: 1 (4/1)
##                 :           Duration.in.Current.address > 1: 0 (23)
##                 Duration.of.Credit..month. <= 36:
##                 :...Guarantors > 2:
##                     :...Foreign.Worker <= 1: 1 (23/1)
##                     :   Foreign.Worker > 1: 0 (2)
##                     Guarantors <= 2:
##                     :...Credit.Amount <= 1381:
##                         :...Telephone > 1:
##                         :   :...Sex...Marital.Status > 3: 0 (2)
##                         :   :   Sex...Marital.Status <= 3:
##                         :   :   :...Duration.of.Credit..month. <= 16: 1 (7)
##                         :   :       Duration.of.Credit..month. > 16: 0 (3/1)
##                         :   Telephone <= 1:
##                         :   :...Concurrent.Credits <= 2: 0 (9)
##                         :       Concurrent.Credits > 2:
##                         :       :...Account.Balance <= 1: 0 (29/6)
##                         :           Account.Balance > 1: [S1]
##                         Credit.Amount > 1381:
##                         :...Guarantors > 1:
##                             :...Foreign.Worker > 1: 1 (2)
##                             :   Foreign.Worker <= 1:
##                             :   :...Instalment.per.cent > 2: 0 (5)
##                             :       Instalment.per.cent <= 2: [S2]
##                             Guarantors <= 1:
##                             :...Payment.Status.of.Previous.Credit > 3:
##                                 :...Age..years. > 33: 1 (22)
##                                 :   Age..years. <= 33:
##                                 :   :...Purpose > 3: 1 (7)
```

```
##                                   :       Purpose <= 3: [S3]
##                                   Payment.Status.of.Previous.Credit <= 3:
##                                   :...Instalment.per.cent <= 2:
##                                       :...No.of.dependents > 1:
##                                       :   :...Purpose <= 0: 1 (2)
##                                       :   :   Purpose > 0: 0 (3)
##                                       :   No.of.dependents <= 1: [S4]
##                                       Instalment.per.cent > 2:
##                                       :...Concurrent.Credits <= 1: 1 (8/1)
##                                           Concurrent.Credits > 1:
##                                           :...Sex...Marital.Status <= 1: 0 (6/1)
##                                               Sex...Marital.Status > 1:
##                                               :...Account.Balance > 1: [S5]
##                                                   Account.Balance <= 1: [S6]
##
## SubTree [S1]
##
## Duration.in.Current.address > 3: 1 (8/1)
## Duration.in.Current.address <= 3:
## :...Purpose > 2: 0 (5)
##     Purpose <= 2:
##     :...Type.of.apartment <= 1: 0 (2)
##         Type.of.apartment > 1: 1 (5/1)
##
## SubTree [S2]
##
## Duration.in.Current.address <= 2: 1 (2)
## Duration.in.Current.address > 2: 0 (4/1)
##
## SubTree [S3]
##
## Duration.of.Credit..month. <= 16: 1 (4)
## Duration.of.Credit..month. > 16:
## :...Length.of.current.employment <= 3: 0 (8)
##     Length.of.current.employment > 3: 1 (6/1)
##
## SubTree [S4]
##
## Duration.in.Current.address > 1: 1 (41/6)
## Duration.in.Current.address <= 1:
## :...Value.Savings.Stocks > 3: 0 (2)
##     Value.Savings.Stocks <= 3:
##     :...Length.of.current.employment > 2: 1 (4)
##         Length.of.current.employment <= 2:
##         :...Instalment.per.cent <= 1: 0 (3)
##             Instalment.per.cent > 1: 1 (3/1)
##
## SubTree [S5]
##
## Sex...Marital.Status > 3: 0 (2)
## Sex...Marital.Status <= 3:
## :...Length.of.current.employment > 3: 1 (10)
##     Length.of.current.employment <= 3:
##     :...Duration.in.Current.address <= 1: 1 (5)
##         Duration.in.Current.address > 1:
##         :...Length.of.current.employment <= 2: 0 (4)
##             Length.of.current.employment > 2:
##             :...Value.Savings.Stocks <= 1: 0 (3)
##                 Value.Savings.Stocks > 1: 1 (5)
##
## SubTree [S6]
##
## Payment.Status.of.Previous.Credit > 2: 0 (3)
```

```
## Payment.Status.of.Previous.Credit <= 2:
## :...Purpose <= 0: 0 (7/1)
##     Purpose > 0:
##     :...Most.valuable.available.asset <= 1: 0 (5/1)
##         Most.valuable.available.asset > 1:
##         :...Sex...Marital.Status <= 2: 1 (6)
##             Sex...Marital.Status > 2:
##             :...Length.of.current.employment > 4: 0 (5)
##                 Length.of.current.employment <= 4:
##                 :...Telephone > 1: 1 (3)
##                     Telephone <= 1:
##                     :...Length.of.current.employment <= 2: 0 (2)
##                         Length.of.current.employment > 2:
##                         :...Age..years. <= 28: 1 (4)
##                             Age..years. > 28: 0 (2)
##
##
## Evaluation on training data (900 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      85   70( 7.8%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     233    45    (a): class 0
##      25   597    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% Account.Balance
##   67.11% Credit.Amount
##   63.11% Concurrent.Credits
##   55.33% Payment.Status.of.Previous.Credit
##   50.33% Age..years.
##   45.44% Duration.of.Credit..month.
##   40.11% Guarantors
##   24.44% Occupation
##   18.33% Instalment.per.cent
##   15.56% Purpose
##   14.22% Length.of.current.employment
##   13.67% Duration.in.Current.address
##   12.67% Value.Savings.Stocks
##   12.22% No.of.dependents
##    9.33% Sex...Marital.Status
##    9.00% Telephone
##    8.78% Most.valuable.available.asset
##    4.22% Foreign.Worker
##    2.11% No.of.Credits.at.this.Bank
##    0.78% Type.of.apartment

Time: 0.0 secs
```

Step 4: Evaluating Model Performance

We still need to use our test set to evaluate/validate the model's overall performance. To do this we'll use the predict() command as follows:

```
> cred_pred <- predict(credit_model, credit_test)
```

Last, we'll use the gmodels package to create a confusion table looking at the predicted and actual values using the training and test sets for our credit data. So, first make sure gmodels is installed and attached using library(gmodels). Then use the CrossTable() command as follows:

```
> CrossTable(credit_test$Creditability, cred_pred, prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE, dnn = c(
'Actual Creditability', 'Predicted Creditability'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                      | Predicted Creditability
## Actual Creditability |         0 |         1 | Row Total |
## ---------------------|-----------|-----------|-----------|
##                    0 |         8 |        14 |        22 |
##                      |     0.080 |     0.140 |           |
## ---------------------|-----------|-----------|-----------|
##                    1 |        17 |        61 |        78 |
##                      |     0.170 |     0.610 |           |
## ---------------------|-----------|-----------|-----------|
##         Column Total |        25 |        75 |       100 |
## ---------------------|-----------|-----------|-----------|
```

The table indicates that for the 100 records in our test set 12 cases were misclassified, i.e. false negatives or a Type II error, and 15 actual defaults were misclassified as not careditable, i.e. false positives or a Type I error.

Q1- If you see an accuracy of 100%, what does it mean? Does this mean that we design a perfect model? This is some thing that needs more discussion. Write a few sentences about accuracy of 100%.

Method #2. Random forest

Let's use the same dataset but this time for random forest. Use the similar strategy described in course to see the accuracy.

```
> credit_train$Creditability <- as.factor(credit_train$Creditability)
> random_model <- randomForest(Creditability ~ . , data= credit_train)
> summary(random_model)
```

To evaluate the model:

```
> cred_pred <- predict(random_model, credit_test)
> (p <- table(cred_pred, credit_test$Creditability))
```

And the accutracy is:

```
> (Accuracy <- sum(diag(p))/sum(p)*100)
```

79%

Q2- What are the three most important features in this model.

Now, Change the random seed to 23458 and find the new accuracy of random forest.

Method #3. Adding regression to trees

(MLR pg 190)

Step 1: Collecting the Data

Our last method will use the whitewines.csv file available for you on Moodle. There is nothing quirky about reading this data into R. The read command and structure of the wine object are shown below.

```
> wine <- read.csv("whitewines.csv")
> str(wine)
'data.frame':          4898 obs. of  12 variables:
 $ fixed.acidity       : num  6.7 5.7 5.9 5.3 6.4 7 7.9 6.6 7 6.5 ...
 $ volatile.acidity    : num  0.62 0.22 0.19 0.47 0.29 0.14 0.12 0.38 0.16 0.37 ...
 $ citric.acid         : num  0.24 0.2 0.26 0.1 0.21 0.41 0.49 0.28 0.3 0.33 ...
 $ residual.sugar      : num  1.1 16 7.4 1.3 9.65 0.9 5.2 2.8 2.6 3.9 ...
 $ chlorides           : num  0.039 0.044 0.034 0.036 0.041 0.037 0.049 0.043 0.043 0.027 ...
 $ free.sulfur.dioxide : num  6 41 33 11 36 22 33 17 34 40 ...
 $ total.sulfur.dioxide: num  62 113 123 74 119 95 152 67 90 130 ...
 $ density             : num  0.993 0.999 0.995 0.991 0.993 ...
 $ pH                  : num  3.41 3.22 3.49 3.48 2.99 3.25 3.18 3.21 2.88 3.28 ...
 $ sulphates           : num  0.32 0.46 0.42 0.54 0.34 0.43 0.47 0.47 0.47 0.39 ...
 $ alcohol             : num  10.4 8.9 10.1 11.2 10.9 ...
 $ quality             : int  5 6 6 4 6 6 6 6 6 7 ...
```

Again, there isn't anything particularly quirky about this data on the surface. However, because we will use regression we should check to see if the class variable, quality, follows a normal distribution or is nearly normal. We can do that with the hist() function as follows:

```
> hist(wine$quality)
```

This command will automatically generate a plot in "Plots". I will leave it up to you to answer the question of whether or not this data are normal.

Step 2: Exploring and Preparing the Data

Essentially all we need to do this time is to subset our data into training and test sets. Let's use an approximately 75% for training and 25% for testing split of the observations.

```
> wine_train <- wine[1:3750, ]
> wine_test <- wine[3751:4898, ]
```

Step 3: Training a Model on the Data

To create this model we'll use the rpart package. "rpart" stands for recursive partitioning. Using the rpart function we'll set-up our command with the formula syntax and the class variable quality as follows:

```
> m.rpart <- rpart(quality ~ ., data=wine_train)
```

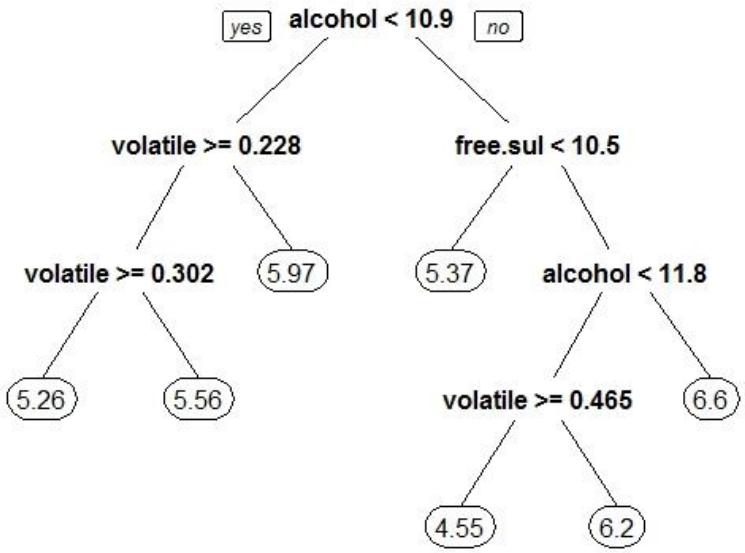
To get the basic information about the tree just type in the object name m.rpart at the command prompt:

```
> m.rpart
n= 3750

node), split, n, deviance, yval
      * denotes terminal node

 1) root 3750 2945.53200 5.870933
   2) alcohol< 10.85 2372 1418.86100 5.604975
     4) volatile.acidity>=0.2275 1611  821.30730 5.432030
       8) volatile.acidity>=0.3025 688  278.97670 5.255814 *
       9) volatile.acidity< 0.3025 923  505.04230 5.563380 *
     5) volatile.acidity< 0.2275 761  447.36400 5.971091 *
   3) alcohol>=10.85 1378 1070.08200 6.328737
     6) free.sulfur.dioxide< 10.5 84   95.55952 5.369048 *
     7) free.sulfur.dioxide>=10.5 1294  892.13600 6.391036
      14) alcohol< 11.76667 629  430.11130 6.173291
        28) volatile.acidity>=0.465 11   10.72727 4.545455 *
        29) volatile.acidity< 0.465 618  389.71680 6.202265 *
      15) alcohol>=11.76667 665  403.99400 6.596992 *
```

Note that the rpart function automatically determined that the most important predictor was the variable "alcohol". The root split between <10.85 and >=10.85 as shown above. Nodes with an * are terminal or leaf nodes.

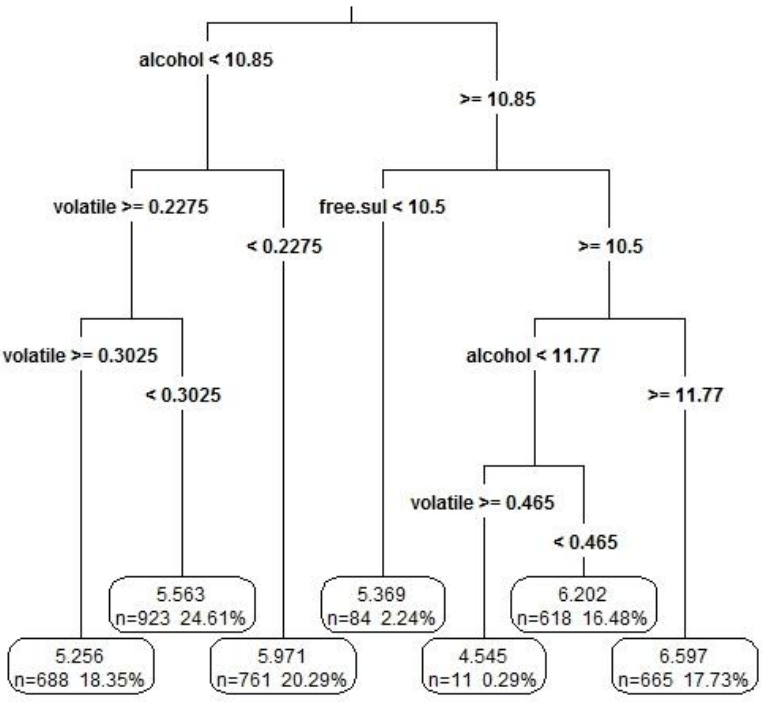
If we want to actually plot the tree we need to install and attach the rpart.plot package. Once that is finished the command is (I've also pasted a picture of the tree below the command):

```
> rpart.plot(m.rpart, digits=3)
```



Another, very different way to visualize the tree is using the command:

```
> rpart.plot(m.rpart, digits=4, fallen.leaves = TRUE, type = 3, extra = 101)
```



Step 4: Evaluating Model Performance

In this case we'll look at summary statistics to evaluate our model's performance. First compute the *p.rpart* using *predict()* function (see step 1).

```
> summary(p.rpart)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.545   5.563   5.971   5.893   6.202   6.597
> summary(wine_test$quality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   5.000   6.000   5.901   6.000   9.000
```

The fact that the model's range is much smaller than the data's actual range suggests that the model is not adequately capturing the extremes of the range of quality, i.e. the very good or very bad wines. We can also check the correlation:

```
> cor(p.rpart, wine_test$quality)
[1] 0.5369525
```

A 54% correlation is ok, but not great. Last thing that we can try is to see the amount of RMSE (Root Mean Square Error) for all the test instances. You will find out an RMSE of around 0.84.

Q3- What is your interpretation about this amount of RMSE?

Method #4. News Popularity

The Online News Popularity data set from the University of California – Irvine Machine Learning Data Repository is provided for your use as well as two related journal articles, "Predicting and Evaluating the Popularity of Online News" by He Ren and Quan Yang, and "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News" by Kelwin Fernandes, Pedro Vinagre and Paulo Cortez. The URL for the data set description is: http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity#. Ultimately, your assignment is to use this one dataset and compare the results of these three different machine learning techniques to evaluate which of those provides the best results. Because you are assessing the "market share" based on different measures of popularity, in this case the best results will be the results that provide the best understanding of what drives market share. This is different than determining exactly what the market share is.

Step 1: Collecting the Data

Again start with loading the data and see how the output looks like

The goal is to see how popular the articles are. There are 61 features here but this makes analysis so time consuming while we don't need some of the features anyways.

Step 2: Pre-processing

We want to make this problem a classification one. One approach is to make any piece of article more than 1400 likes as a favorite one.

Step 3: Modeling and evaluation

Q4- Try decision tree and random forest and evaluate the model.