AEDA

Exercise 2

Raúl García Gómez

Marc Monclús

Ali Muhammad

Problem 1

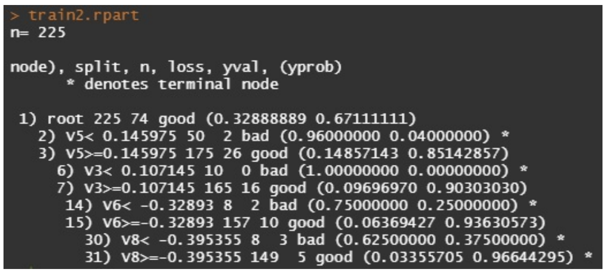
This dataset consists of a series of numerical variables that end up describing the state of the ionosphere and qualifying it as ‘bad’ or ‘good’ in a final qualitative variable.

To begin we load the necessary libraries and set up the work directory. For this problem, the dataset is obtained from one of the packages.

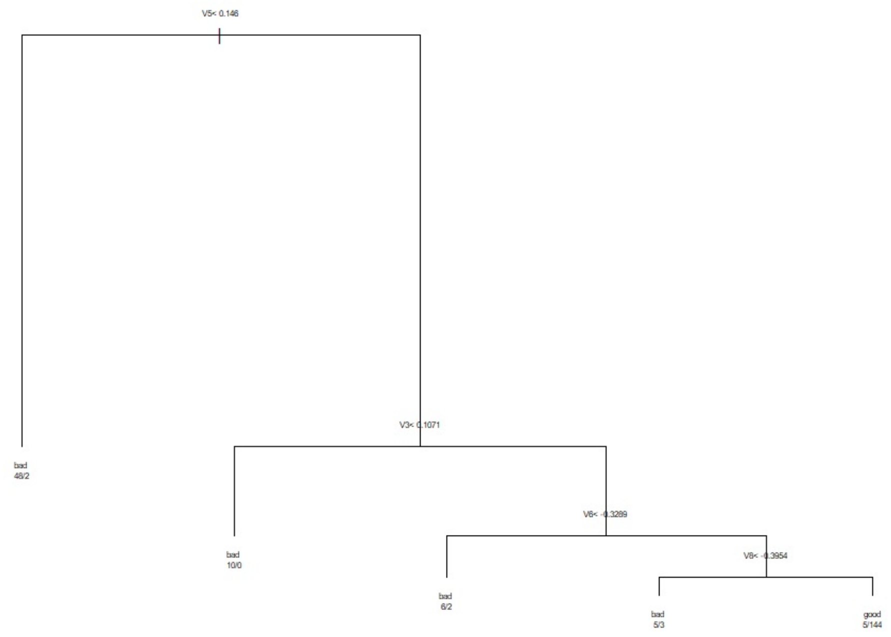
The first step will be cleaning up the dataset. There are parts to this, one is eliminating the column ‘V2’. We do this because when we check the summary of the dataset we can see that this column is just filled with 0s so we eliminate because it will give no information. The other part is eliminating the outliers, we will do this with the quantitative continuous variables, meaning all variables except ‘V1’ which is a binary variable and ‘class’ which is the dependent variable that makes this dataset a supervised one.

After the dataset has been cleaned, we do the final preparatory step by separating the dataset into a training set and a test set in an 80%/20% ratio respectively.

Once done we create the classification tree with the training set and plot it.



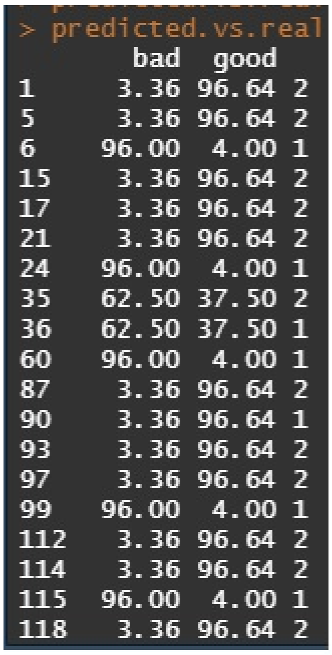
```
> train2.rpart
n= 225

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 225 74 good (0.32888889 0.67111111)
   2) V5< 0.145975 50  2 bad (0.96000000 0.04000000) *
   3) V5>=0.145975 175 26 good (0.14857143 0.85142857)
     6) V3< 0.107145 10  0 bad (1.00000000 0.00000000) *
     7) V3>=0.107145 165 16 good (0.09696970 0.90303030)
      14) V6< -0.32893 8  2 bad (0.75000000 0.25000000) *
      15) V6>=-0.32893 157 10 good (0.06369427 0.93630573)
        30) V8< -0.395355 8  3 bad (0.62500000 0.37500000) *
        31) V8>=-0.395355 149  5 good (0.03355705 0.96644295) *
```



Training Classification tree result and plot

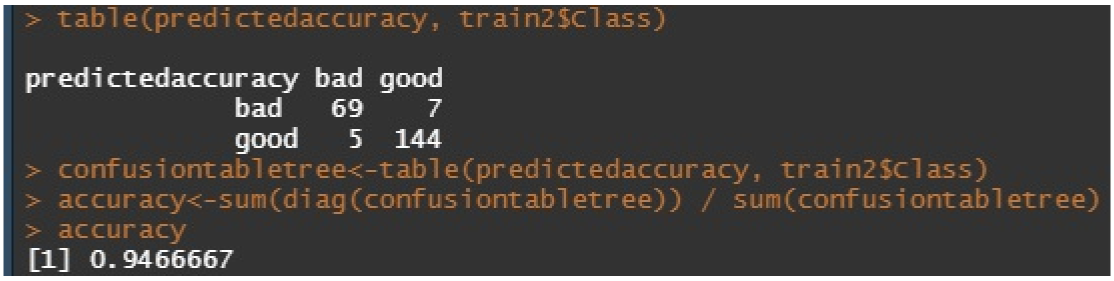
Before extracting conclusions about it we use the 1SD rule to prune the tree. For that we use the 'printcp' function to obtain the deviation errors, find the smallest (0.27027) and add 1 time the standard deviation error of that leaf (0.057686). The result tells us to prune the fourth leaf (4th leaf xerror = 0.32432<0.327956=0.27027+0.057686), but because it already belongs to the last branch of the tree, it remains without being pruned.

Now we can see that the most important variables have been 'V5', 'V3', 'V6' and 'V8' in that order but it remains to be seen if this classification is accurate or not. Predicting the training set give us these values of prediction and accuracy:



```
> predicted.vs.real
      bad  good
1     3.36 96.64 2
5     3.36 96.64 2
6    96.00  4.00 1
15    3.36 96.64 2
17    3.36 96.64 2
21    3.36 96.64 2
24   96.00  4.00 1
35   62.50 37.50 2
36   62.50 37.50 1
60   96.00  4.00 1
87    3.36 96.64 2
90    3.36 96.64 1
93    3.36 96.64 2
97    3.36 96.64 2
99   96.00  4.00 1
112   3.36 96.64 2
114   3.36 96.64 2
115  96.00  4.00 1
118   3.36 96.64 2
```

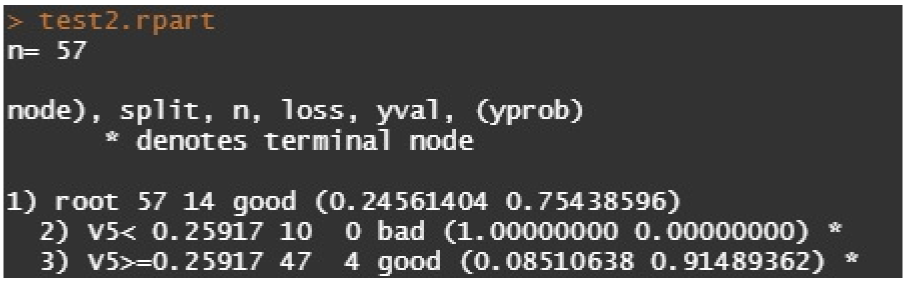
Training classification tree prediction



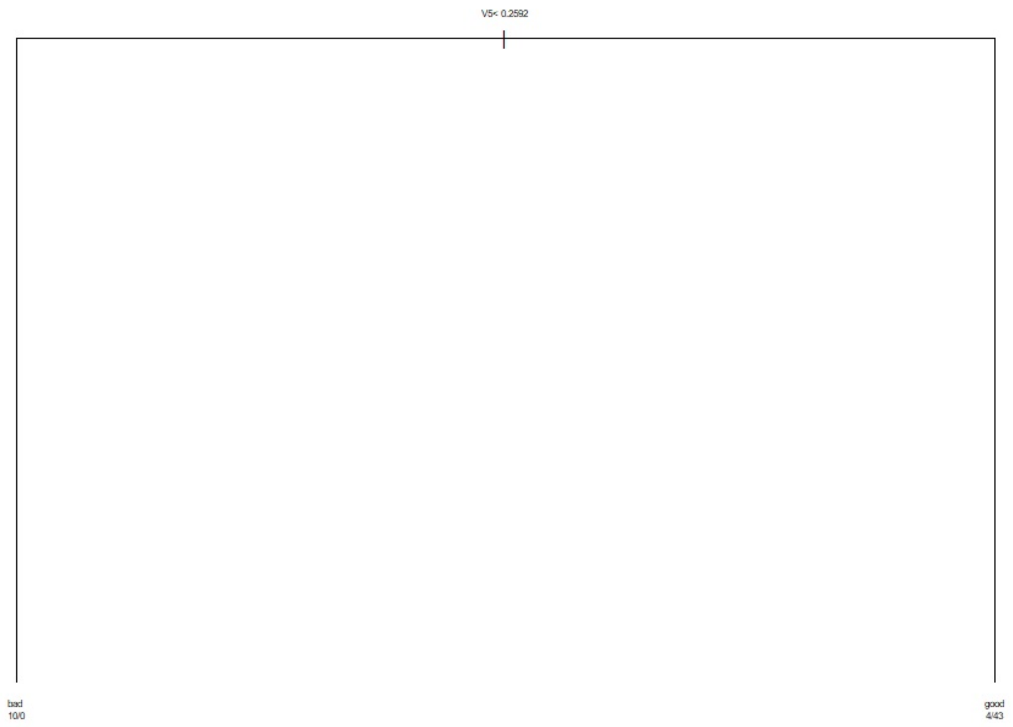
```
> table(predictedaccuracy, train2$Class)

predictedaccuracy bad good
             bad   69    7
             good   5  144
> confusiontabletree<-table(predictedaccuracy, train2$Class)
> accuracy<-sum(diag(confusiontabletree)) / sum(confusiontabletree)
> accuracy
[1] 0.9466667
```

Confusion matrix and accuracy rate

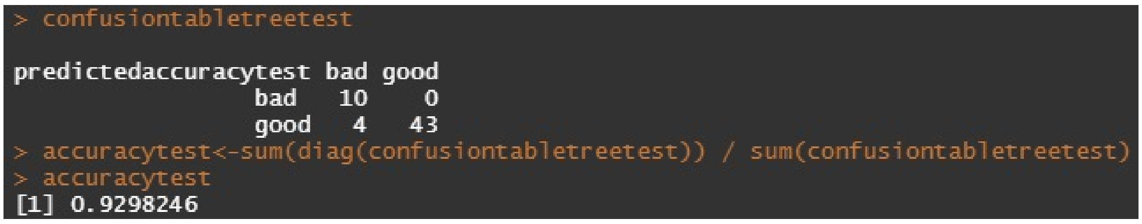
The training set seems to have a 94,6% accuracy rate but to properly check we need to run the prediction with the test set:



```
> test2.rpart
n= 57

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 57 14 good (0.24561404 0.75438596)
  2) V5< 0.25917 10  0 bad (1.00000000 0.00000000) *
  3) V5>=0.25917 47  4 good (0.08510638 0.91489362) *
```



Test classification tree and plot

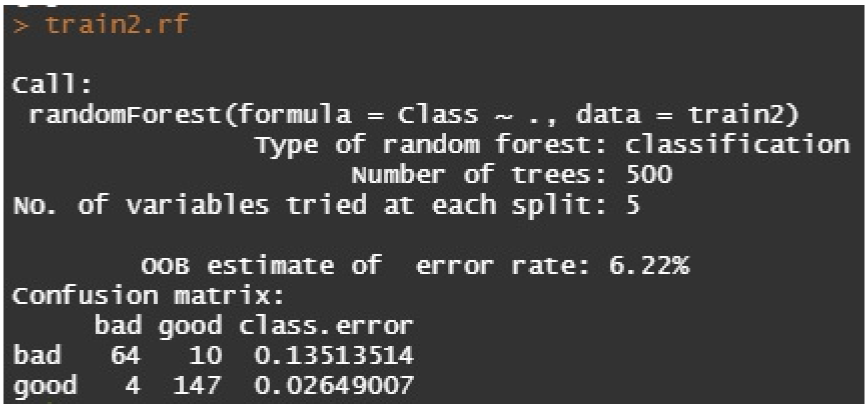


```
> confusiontabletreetest

predictedaccuracytest bad good
                 bad   10    0
                 good   4   43
> accuracytest<-sum(diag(confusiontabletreetest)) / sum(confusiontabletreetest)
> accuracytest
[1] 0.9298246
```

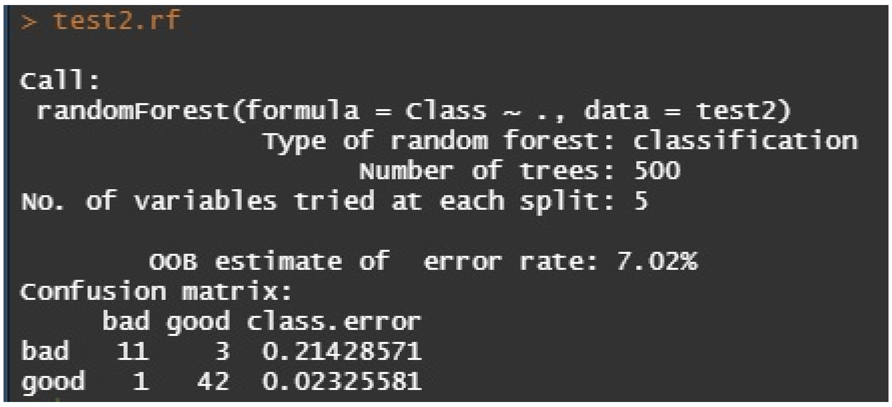
Test confusion matrix and accuracy rate

From the test set we obtain deep confirmation that the most important variable to classify data as ‘bad’ or ‘good’ is ‘V5’ and we reach this conclusion with an accuracy rate of 92,9%.

To end the problem we use random forest to try and obtain better results of the classification tree. Using the default 500 trees with 5 variables each split bring Out Of Bag rates of 6,22% for the training set and 7,02% for the test set. The training set gives a slightly lower accuracy value but the test result is slightly better in the test set, but they're so minor differences that both the classification tree and the random forest seem equally good models for prediction.



```
> train2.rf

Call:
 randomForest(formula = Class ~ ., data = train2)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 6.22%
Confusion matrix:
     bad good class.error
bad   64   10  0.13513514
good   4  147  0.02649007
```



```
> test2.rf

Call:
 randomForest(formula = Class ~ ., data = test2)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 7.02%
Confusion matrix:
     bad good class.error
bad   11    3  0.21428571
good   1   42  0.02325581
```

Random forest result for the training and test sets respectively