Decision Tree, SVM, Boosting

Data Set 1:

<http://archive.ics.uci.edu/ml/datasets/Image+Segmentation>

The instances were drawn randomly from a database of 7 outdoor images. The images were hand segmented to create a classification for every pixel.

Attribute Information:

1. region-centroid-col: the column of the center pixel of the region.

2. region-centroid-row: the row of the center pixel of the region.

3. region-pixel-count: the number of pixels in a region = 9.

4. short-line-density-5: the results of a line extractoin algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.

5. short-line-density-2: same as short-line-density-5 but counts lines of high contrast, greater than 5.

6. vedge-mean: measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.

7. vegde-sd: (see 6)

8. hedge-mean: measures the contrast of vertically adjacent pixels. Used for horizontal line detection.

9. hedge-sd: (see 8).

10. intensity-mean: the average over the region of (R + G + B)/3

11. rawred-mean: the average over the region of the R value.

12. rawblue-mean: the average over the region of the B value.

13. rawgreen-mean: the average over the region of the G value.

14. exred-mean: measure the excess red: (2R - (G + B))

15. exblue-mean: measure the excess blue: (2B - (G + R))

16. exgreen-mean: measure the excess green: (2G - (R + B))

17. value-mean: 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)

18. saturatoin-mean: (see 17)

19. hue-mean: (see 17)

Classification is used here to identify the image which the pixel belongs to. This seems to be interesting as the data consists of real values and will help me identify the difficulties in image recognition.

Data Set 2:

<http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice>

This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

Attribute Information:

1. Wife's age (numerical)

2. Wife's education (categorical) 1=low, 2, 3, 4=high

3. Husband's education (categorical) 1=low, 2, 3, 4=high

4. Number of children ever born (numerical)

5. Wife's religion (binary) 0=Non-Islam, 1=Islam

6. Wife's now working? (binary) 0=Yes, 1=No

7. Husband's occupation (categorical) 1, 2, 3, 4

8. Standard-of-living index (categorical) 1=low, 2, 3, 4=high

9. Media exposure (binary) 0=Good, 1=Not good

10. Contraceptive method used (class attribute) 1=No-use, 2=Long-term, 3=Short-term {Class}

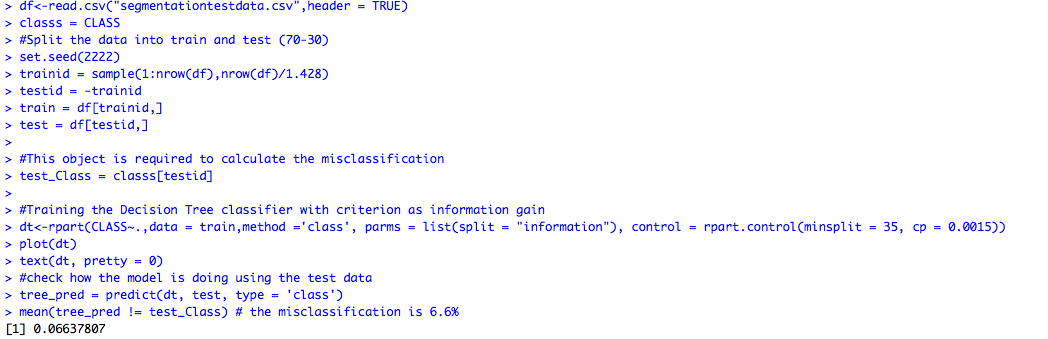
Classification method is used to identify the contraceptive method used. This dataset is useful as the data consists of both continuous as categorical data.

## Classification Method 1: Decision Trees

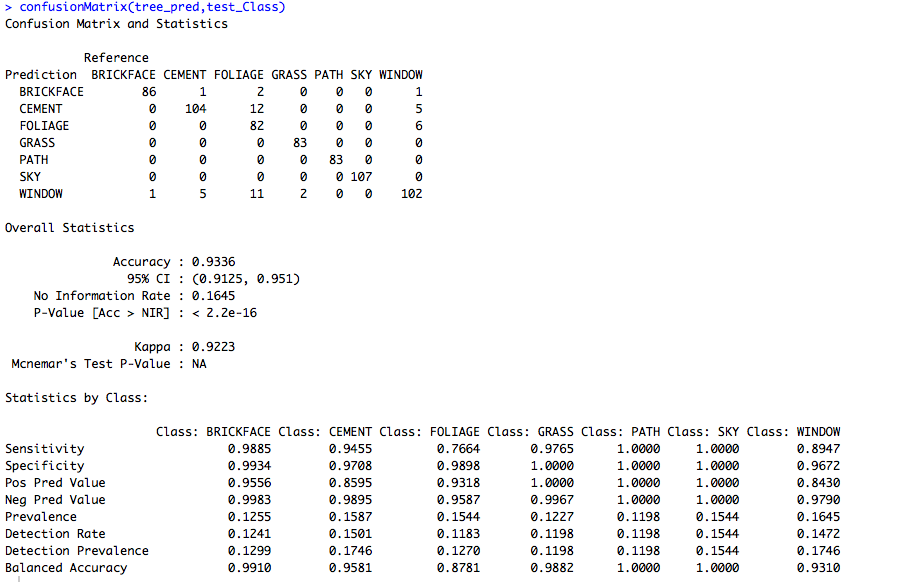
Package used: rpart

Results:

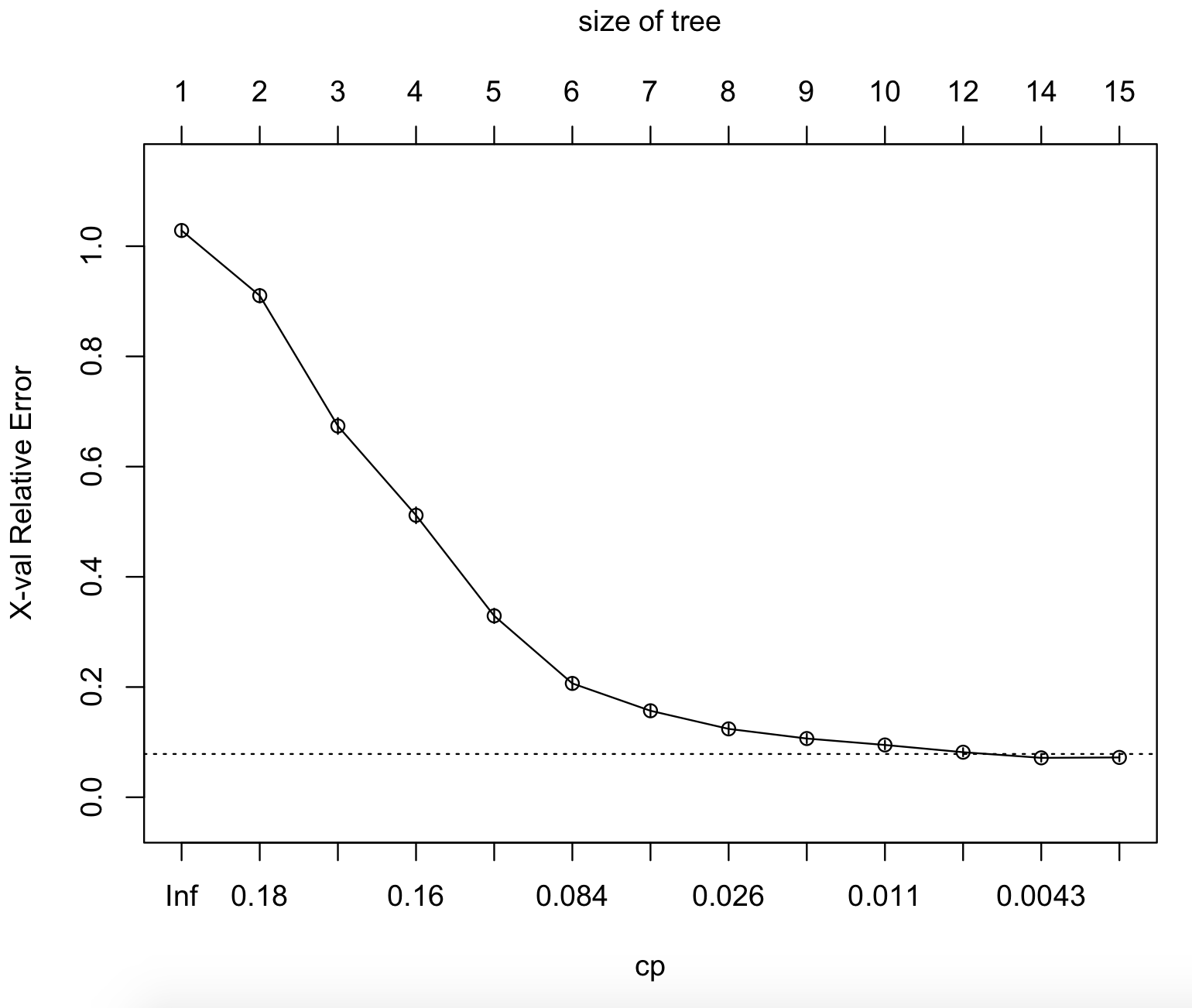
Data set 1 has given an accuracy of 93.4% with the split method as Information Gain.



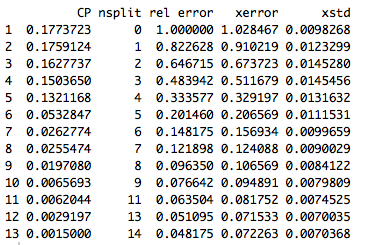
Below is the confusion matrix:



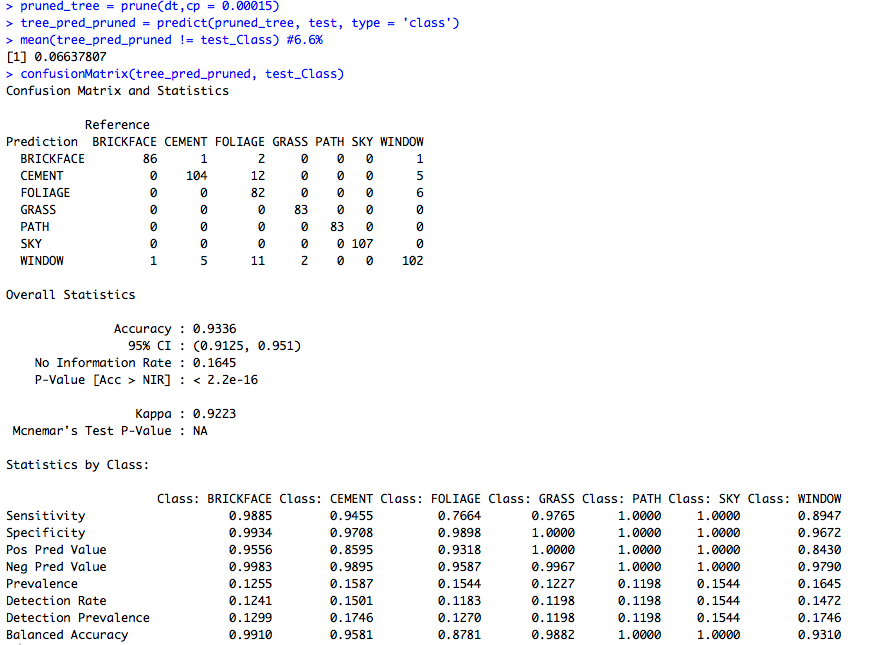
Below is the complex parameter curve obtained on the model:



From the graph above, the best CP has been chosen. Below is the table for the same.

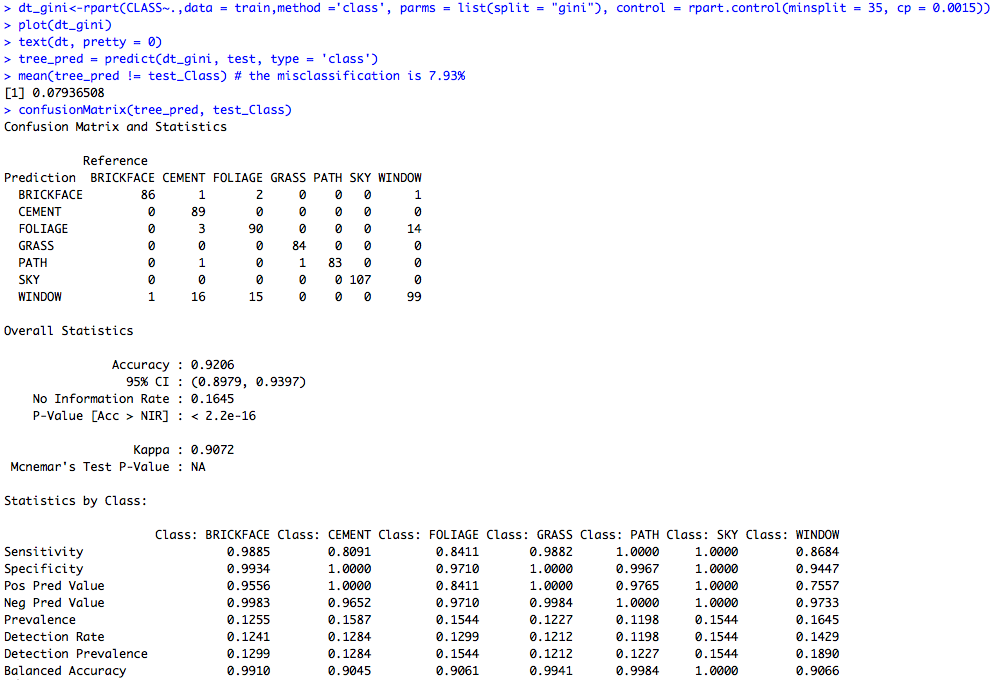


The model has been pruned and identified that the accuracy is still the same. This shows that the model does not have the problem of over fitting.



Therefore, the Gini index is used to run the decision tree algorithm on the same dataset.

Below is the implementation:



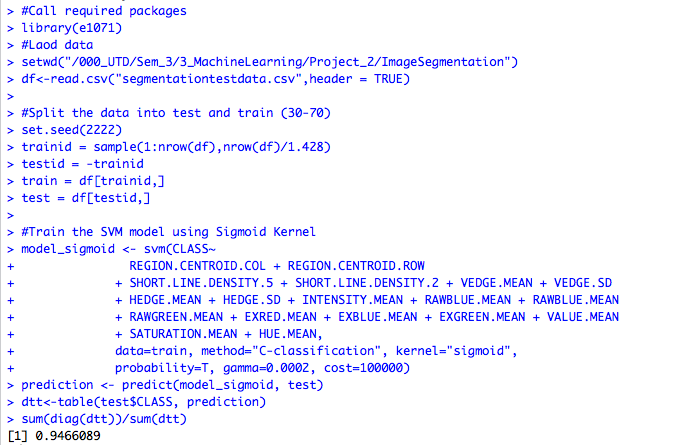
As you can see the accuracy is 92.06 % Upon pruning the same accuracy is obtained. So, no pruning required.

Hence for the given data set information gain is a better splitting method.

This is because the no of elements under each node needed weightage while allocating the entropy for the node. Thereby choosing the best split to train the model this way has provided better accuracy over the gini index method.

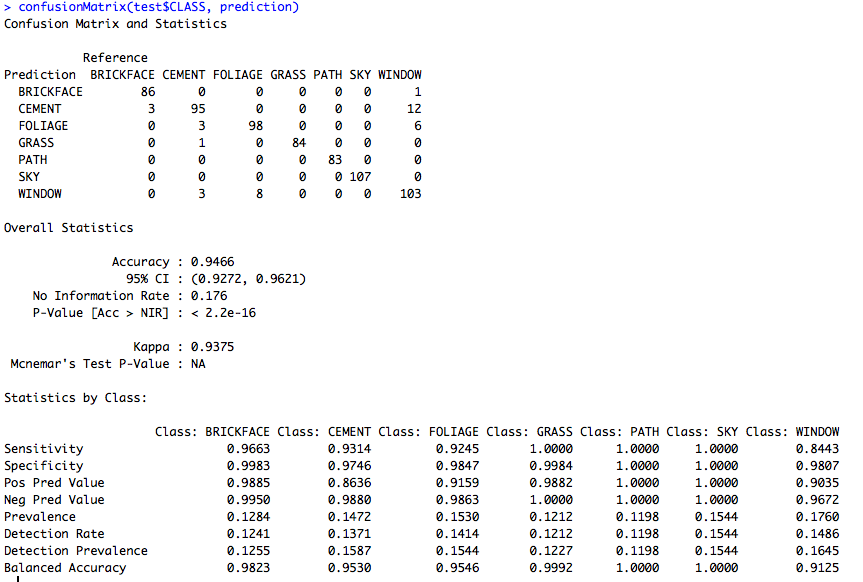
For the second dataset, the same procedure has been followed and Gini index has given the better accuracy over information gain. Several pruning values are tested and reached the best cp value for the pruning the tree.

## Classification Method 2: SVM



As you can see the Kernel sigmoid was able to train a model which has an accuracy of 94.66%

Below is the confusion matrix for the same:



Below are the results:

Sigmoid Kernel has given an accuracy of: 94.66%

Radial Kernel has given an accuracy of: 95.67%

Linear Kernel has given an accuracy of: 89.89%

Polynomial Kernel has given an accuracy of: 38.09%

Therefore, radial kernel has the highest accuracy for this dataset.

Below are the results for second data set:

Sigmoid Kernel has given an accuracy of: 50.45%

Radial Kernel has given an accuracy of: 57.46%

Linear Kernel has given an accuracy of: 41.62%

Polynomial Kernel has given an accuracy of: 44.34%

Therefore, radial kernel has the highest accuracy for this dataset.

## Classification Method 3: XGBOOST

Purpose / description

Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine.  It supports various objective functions, including regression, classification and ranking.  It also has additional features for doing cross validation and finding important variables. There are many parameters which needs to be controlled to optimize the model.  XGBoost only works with numeric vectors. So, the training data needs to be converted into matrix with numeric type only. The following R code performs the function of converting the training data into numeric matrix.

The parameters used for tuning the algorithm are as follows:

param <- list("objective" = "multi:softprob",

"num\_class" = 100,

"eval\_metric" = "merror",

"nthread" = 8,

"max\_depth" =8,

"eta" = 0.3,

"gamma" = 0,

"subsample" = 1,

"colsample\_bytree" = 1

)

1.objective : multi:softprob

We chose multi:softprob as it used to do multi class classification problem.

2.num\_class : 7

As the number of cluster are 7, we specify classification into 7 classes.

3. eval\_metric : merror

This is the evaluation metric used by the algorithm to assess the performance of the algorithm. We use merror which is exact matching error, used to evaluate multi-class classification.

4.nthread : 8

This is used to control the number of threads used in the training. To make best use of the processing power of the machine at our disposal we specified the number of threads to 8.

5. max\_depth : 8

The maximum depth of the tree to be generated by the model.

6. eta : 0.3

This is the step size of each boosting step.

7. gamma: 0

This controls the minimum loss reduction required to make a further partition on a leaf node of the tree node the tree. The larger, the more conservative the algorithm will be. So we chose to assign a value of 0 for this control parameter.

8.subsample : 1

This parameter controls the subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this will prevent overfitting. It makes computation shorter. It is advised to use this parameter with eta and increase nround. So, we use the default value of 1 for this parameter.

9.colsample\_bytree

This parameter controls subsample ratio of columns when constructing each tree. The default value of 1 is used for this parameter.

Algorithm implementation:

R Code:

>set.seed(42) # set random seed to make model reproducible.

>bst.cv <- xgb.cv(param=param, data=trainMatrix, label=y, nfold=4, nrounds=50, prediction=TRUE, verbose=TRUE)

Explanation:

1.param= param

tuning parameters for the algorithm passed through parameter “param”.

2.Data=trainMatrix

trainMatrix is the data used for training the algorithm.

3.label=y

Y is the variable containing the label for the classification.

4.nfold=4

We made use of 4-fold cross validation to judge the performance of the classification power of the algorithm for the training data set.

5.nrounds=50

The max number of iterations to be used for growing the forest. We used 50 iterations for training the algorithm.

6.Verbose=TRUE

If 0, xgboost will stay silent. If 1, xgboost will print information of performance. We chose to print the performance at each step of the iteration.

As the algorithm only accepts numerical data and we have tag the classes from 0 to 6 the below function is written for the same:

vlookup<-function(fact,vals,x) {

#probably should do an error checking to make sure fact

# and vals are the same length

out<-rep(vals[1],length(x))

for (i in 1:length(x)) {

out[i]<-vals[levels(fact)==x[i]]

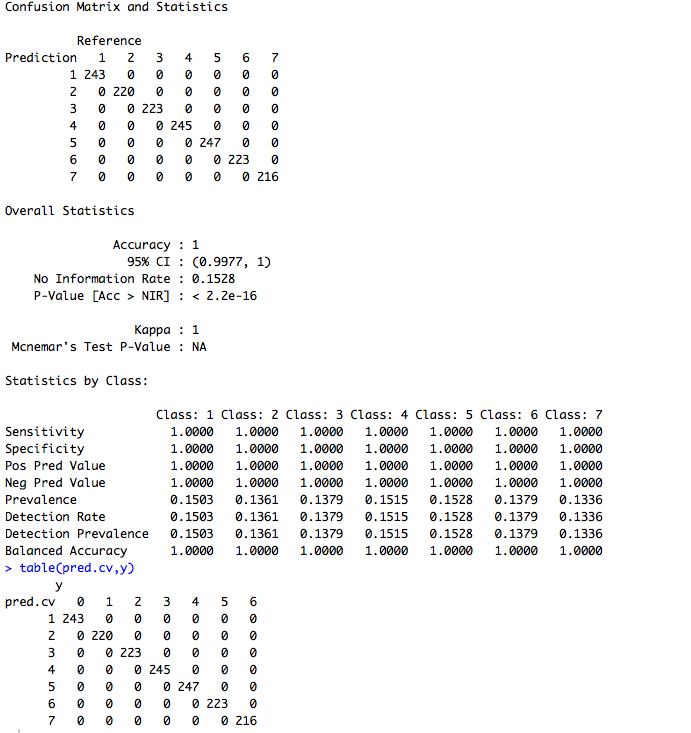
}

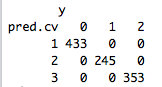
return(out)

}

## XGBOOST Output

XGBoost gave an accuracy of 100% and the tree depth of 8, 7,6, 5 are tested and received the same results.





As you can see the prediction are in place with the target class values. This will show that the accuracy is 100%

The same results were obtained for the second data set as well.