ASSIGNMENT – 1 (Report)

of

MACHINE LEARNING | CSPC-204

ACADEMIC YEAR: 2019-2020

SUBMITTED BY:

RAJAN KATARIA
18103076
CSE/4TH SEMESTER
GROUP: G4

SUBMITTED TO:

Dr. KULDEEP KUMAR
ASSISTANT PROFESSOR
CSE DEPARTMENT

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DR. B.R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY, JALANDHAR

CONTENTS

ATTACHMENTS

The following files have been submitted along with this report in the .zip format named as “18103076.zip” :

1. R notebook implementation of code (Assign_18103076.nb.html)
2. R notebook webpage (Assign_18103076.html)
3. Balance-scale Dataset (balance_scale.csv)
4. Balance-scale Dataset (balance_scale.xlsx)

1.1 Objective

The objective of this assignment is to implement the decision tree using the two splitting criterion i.e. Information Gain and Gini index. After that we would compare the accuracy of both these algorithms. This document will provide a brief introduction towards the Decision tree algorithms and then describe the steps followed to implement the above two algorithms.

1.2 Introduction to Decision Trees

The decision-tree algorithms are widely used methods for the inductive inference. They approximate discrete functions being robust to the data with noise. The decision tree learning algorithms include ID3, CART, C4.5, etc. These are the supervised learning algorithms used for both type of tasks, classification and regression.

1.2.1 ID3 Algorithm

The full form of ID3 is Iterative Dichotomiser 3. The ID3 algorithm was developed by Ross Quinlan in the year 1986. This algorithm makes a multiway decision-tree, after finding the categorical feature for each node via Greedy approach that will result in the largest information gain for the categorical target. The decision-trees are grown to the maximum size i.e. to the greatest depth and then a pruning step is done to improve the ability of the tree to generalize to the unseen data and also to decrease the depth of the tree. To find the node with maximum of information gain, we first split the dataset along the values of descriptive features and then calculate the entropy of the dataset. This gives us the remaining entropy once we have split the dataset along the feature values. After that, information gain of a feature is calculated as:

$$\text{Information Gain (feature)} = \text{Entropy (Dataset)} - \text{Entropy (feature)}$$

1.2.2 C4.5 Algorithm

C4.5 Algorithm is a successor to ID3 algorithm and does not have any restriction that the features or attributes must be in categorical values by defining a discrete attribute dynamically based on numerical variables that factors the value of continuous variables into the discrete set of intervals. C4.5 algorithm converts the trained trees of the ID3 algorithm into the sets of if-then rules. Then the accuracy of each rule is evaluated in order to determine the order in which they should be applied. Here also, the pruning is done by removing a precondition of if-then rule if the accuracy of the rule improves without it. The splitting criteria used for classification algorithm in C4.5 is Gain Ratio.

1.2.3 CART Algorithm

CART stands for Classification and Regression Trees. This algorithm is very similar to C4.5 algorithm, but the only difference is that it supports numerical target variables i.e. regression. The binary trees are constructed in CART algorithm using this feature and the threshold yields the largest information gain at each node which meets the actual condition of selecting the node as in ID3 algorithm. The splitting criterion used in this algorithm is Gini index. The node with the least Gini index is selected as the next node while drawing a decision tree. The Gini index is calculated by subtracting sum of squared probabilities of each class from one. The larger partitions are favored by Gini Index and easy to implement whereas information gain favors smaller partitions with distinct values.

1.3 Exploring Dataset

To implement the decision tree classification algorithms, the Balance-scale dataset has been taken in consideration. The link to the dataset is https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/.

Balance-scale

In this dataset, we have the classified attribute as Class.Name which has three classes in itself, i.e. 'B', 'L' and 'R'. The classification is decided depending upon four other attributes namely, Left.Weight, Left.Distance, Right,Weight and Right.Distance where Left.Weight and Right.Weight equals the weight on the left and right side of the balance-scale respectively, and Left.Distance and Right.Distance signify the distance of left weight and right weight from the centre of the balance-scale respectively . These four variables or attributes collectively particiopate to determine the side towards which the balance-scale would be bent.

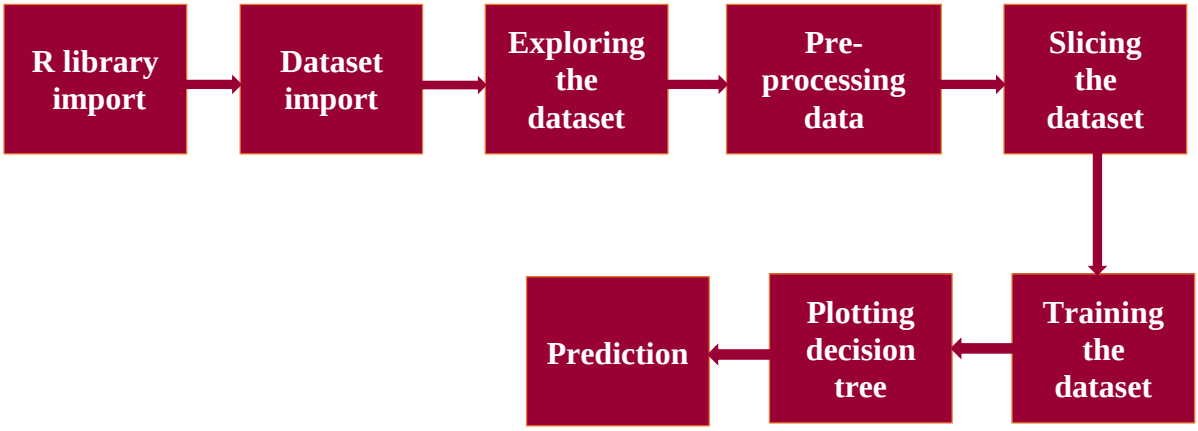
In Class.Name attribute, 'B' stands for Balanced, i.e. the balance-scale would not be either to either side and will remain balanced. 'L' stads for Left, i.e. the balance-scale would be bent towards the left side. 'R' stands for Right, i.e. the balance-scale will be bent towards the right side.

The attributes Left.Weight, Left.Distance, Right,Weight and Right.Distance have the unique values as 1, 2, 3, 4 and 5 considered as 5 levels but are given as integers in the .data file downloaded from the above mentioned link. Hence, as these attributes are not distance and weight in themselves, but are their levels, we have converted these attributes as factor while implementing the decision tree algorithms so that they can add a real meaning to the classifier outputs.

To load the file into R, we have manually converted the .data file to .csv file in MS Excel using the attribute names mentioned in the balance-scale.names file in the same link.

1.4 Implementation of Decision Trees

The flowchart of the steps followed while performing the decision tree algorithms is mentioned as follows:



The step by step procedure showing how the decision-tree has been implemented in the R Notebook attached along with this project is explained as follows. The implementation code in R Notebook is in the zip file submitted in the submission package. Some of the screenshots of the code chunks are also shown here.

1.4.1 R library import

The libraries of different packages installed in R are imported in order to use the functions provided in those packages.

```
library(ggplot2)
library(lattice)
library(lava)
library(purrr)
library(caret)
library(rpart)
library(rpart.plot)
library(ggthemes)
```

1.4.2 Dataset import

The dataset which we have converted from .data file to .csv file is imported into R using read.csv() method.

```
df<- read.csv("C:/Users/Asus/Desktop/ML_18103076/balance_scale.csv")
```

1.4.3 Exploring and visualizing the dataset

After importing the dataset, we need to check which variables and attributes take which type of values and what is the type of the target variable, whether it is classified or need to be converted into classes using some functions like as.factor() etc. Apart from checking the structure or summary of the dataset we can also visualize it using graphs or other visualization techniques. We also check the unique values of the variables and the target variable using the function unique().

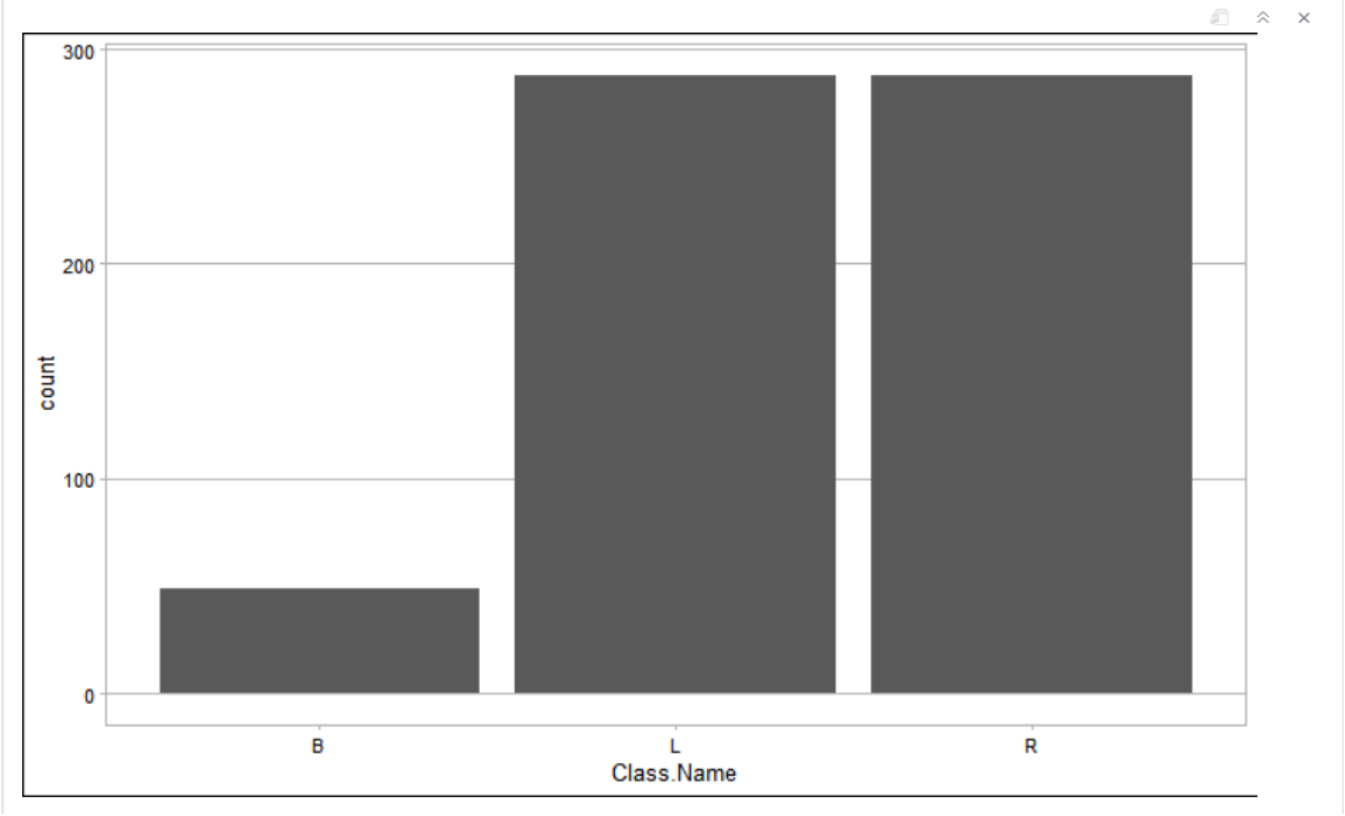
```
str(df)
head(df)
unique(df$Left.Weight)
unique(df$ Left.Distance)
unique(df$Right.Weight)
unique(df$Right.Distance)
unique(df$Class.Name)
```

Also we have converted the int data type into factor data types as mentioned, through this all the variables have got converted into factor data type.

```
df.copy<-df
df.copy$Left.Weight= as.factor(df.copy$Left.Weight)
df.copy$Left.Distance= as.factor(df.copy$Left.Distance)
df.copy$Right.Weight= as.factor(df.copy$Right.Weight)
df.copy$Right.Distance= as.factor(df.copy$Right.Distance)
str(df.copy)
```

The bar graph of the count of target species is mentioned as follows:

```
pl<-ggplot(df.copy, aes(x=Class.Name))
pl+ geom_bar()+theme_calc()
```



1.4.4 Pre-processing of the data

In this step, the NA values are handled normally by replacing them with the mean of the the values of other instances in that column. But, after checking, we found that our dataset is having zero unassigned values.

```
sum(is.na(df.copy))
mean(is.na(df.copy))

#If missing values would have been found, we would have replaced them with the mean of the
values in the respective cloumn
#for(i in 1:ncol(df.copy))
#{df.copy[is.na(df.copy[,i]), i] <- mean(df.copy[,i], na.rm = TRUE)}
```

```
[1] 0
[1] 0
```

1.4.5 Slicing the dataset

Data slicing is a step to split data into train and test set. Training data set can be used specifically for our model building. Test dataset should not be mixed up while building model. We have sliced the total instances into two parts comprising 70% of them in train_data and 30% of them in test_data.

```
pod= sample(2, nrow(df.copy), replace=TRUE, c(0.7, 0.3))
#putting the first index portion of pod in train_data
train_data=df.copy[pod==1,]
#putting the second index portion of pod in test_data
test_data=df.copy[pod==2,]
```

1.4.6 Training the dataset

After pre-processing, the dataset is trained according to the algorithms specified in train() function provided by the rpart package. Here, we are performing 10-fold cross validation on the train_data. Before training, the dataset is passed into trainControl() function which takes the parameters “method”, “number” and “repeats” as arguments. The “method” parameter holds the details about resampling method. We can set “method” with many values like “boot”, “boot632”, “cv”, “repeatedcv”, “LOOCV”, “LGOCV” etc. We are using repeatedcv i.e, repeated cross-validation. The “number” parameter holds the number of resampling iterations. The “repeats ” parameter contains the complete sets of folds to compute for our repeated cross-validation. We are using setting number =10 and repeats =3. This trainControl() methods returns a list.

```
tree.ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

We are going to pass this on our train() method. The only difference between information gain and gini index comes in the split argument of train() method as shown below:

1.4.6.1 Training dataset using Information Gain Criterion

```
ig.dtree <- train(Class.Name ~., data = train_data, method = "rpart",
                  parms = list(split = "information"),
                  trControl=tree.ctrl,
                  tuneLength = 10)
```

1.4.6.2 Training dataset using Gini Index Criterion

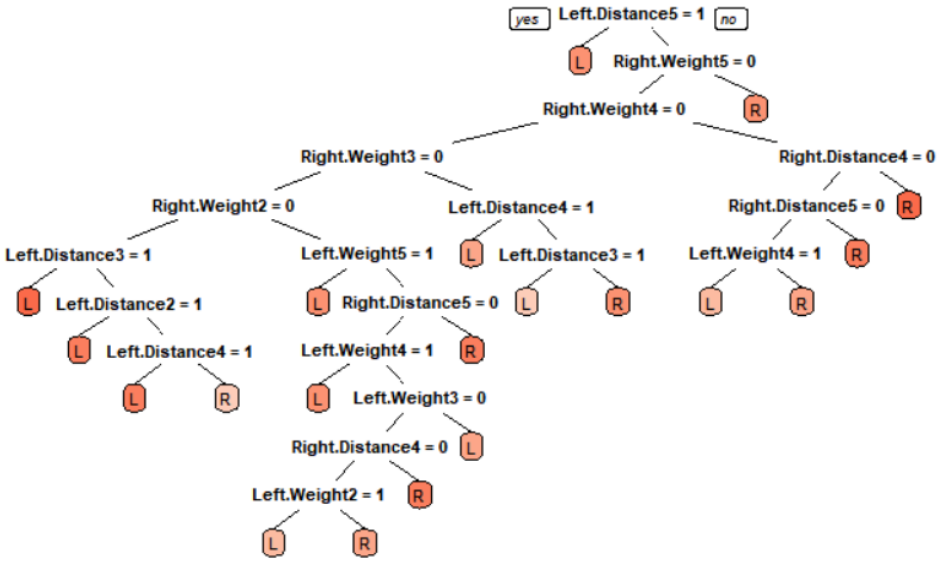
```
gini.dtree <- train(Class.Name~., data = train_data, method = "rpart",
                  parms = list(split = "gini"),
                  trControl=tree.ctrl,
                  tuneLength = 10)
```

1.4.7 Plotting decision tree

The decision tree is being plotted using prp() function below.

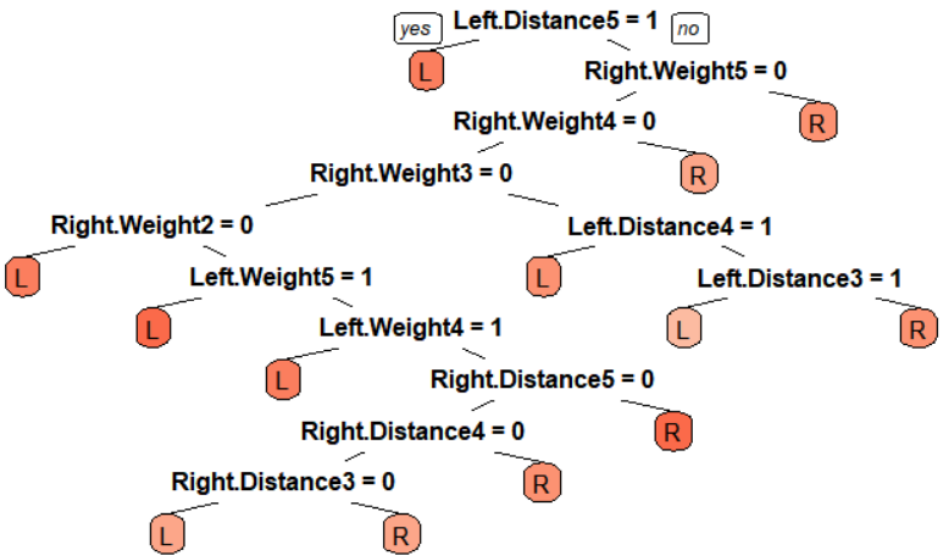
1.4.7.1 Plotting decision tree using Information Gain Criterion

```
prp(ig.dtree$finalModel, box.palette = "Reds", tweak = 1.0)
```



1.4.7.2 Plotting decision tree using Gini Index Criterion

```
prp(gini.dtree$finalModel, box.palette = "Reds", tweak = 1.0)
```



1.4.8 Prediction of classes of test dataset

The prediction is made on the test_data using predict() and confusion matrix is created using confsionMatrix() method. The accuracy and other statistics are also shown when we find the confusion matrix using the above function.

1.4.8.1 Prediction for Information Gain Criterion

```
ig.dtree.predict<- predict(ig.dtree, test_data)
confusionMatrix(ig.dtree.predict, test_data$Class.Name )
```

```
Confusion Matrix and Statistics

          Reference
Prediction  B  L  R
         B  0  0  0
         L  7 56 28
         R  5 23 58

Overall Statistics

               Accuracy : 0.6441
                 95% CI : (0.5688, 0.7145)
    No Information Rate : 0.4859
    P-Value [Acc > NIR] : 1.617e-05

                  Kappa : 0.334

 Mcnemar's Test P-Value : 0.005879

Statistics by Class:

                     Class: B Class: L Class: R
Sensitivity            0.0000   0.7089   0.6744
Specificity            1.0000   0.6429   0.6923
Pos Pred Value            NaN   0.6154   0.6744
Neg Pred Value         0.9322   0.7326   0.6923
Prevalence             0.0678   0.4463   0.4859
Detection Rate         0.0000   0.3164   0.3277
Detection Prevalence   0.0000   0.5141   0.4859
Balanced Accuracy      0.5000   0.6759   0.6834
```

1.4.8.2 Prediction for Gini Index Criterion

```
gini.dtree.predict<- predict(gini.dtree, test_data)
confusionMatrix(gini.dtree.predict, test_data$Class.Name )
```

```
Confusion Matrix and Statistics

          Reference
Prediction  B  L  R
         B  0  0  0
         L  8 56 30
         R  4 23 56

Overall Statistics

               Accuracy : 0.6328
                 95% CI : (0.5572, 0.7038)
    No Information Rate : 0.4859
    P-Value [Acc > NIR] : 5.897e-05

                  Kappa : 0.3137

 Mcnemar's Test P-Value : 0.004803

Statistics by Class:

                     Class: B Class: L Class: R
Sensitivity            0.0000   0.7089   0.6512
Specificity            1.0000   0.6122   0.7033
Pos Pred Value            NaN   0.5957   0.6747
Neg Pred Value         0.9322   0.7229   0.6809
Prevalence             0.0678   0.4463   0.4859
Detection Rate         0.0000   0.3164   0.3164
Detection Prevalence   0.0000   0.5311   0.4689
Balanced Accuracy      0.5000   0.6606   0.6772
```

1.4.9 Comparing accuracy of both splitting criterions

As it is seen from Confusion matrix and statistics in 1.4.8.1 and 1.4.8.2, the accuracy of decision tree classifier made using Information gain as splitting criterion is 64.41% and that made using Gini index is 63.28%. There is not much large difference but for the dataset used above, the information gain splitting criterion is more efficient.

*****