Sowmya Vijayakumar
StudentID: 15233539

Machine Learning Assignment 3

Design, implement and evaluate implementation of machine learning algorithm.

The classification algorithm chosen is C4.5 mysteriously called J48. The programming scripting language used is R programming, Tool used is RStudio. C4.5 is an extension of ID3 algorithm developed by Ross Quinlan. It deals with the given continuous data and pruning trees thereby improving the ID3 algorithm.

The given data has 4 attributes taken in the code as X1-body-length, X2- wing-length,X3- body-width and X4- wing-width . Owl is the target value. The data is divided into training data by 2/3rd and testing data by 2/3rd. Basic rule is to add conditions to the algorithm to improve the accuracy. The algorithm calculates the entropy and Information gain values for the training data for all the attributes.

Entropy is a measure of the uncertainty of a random variable and the formula is:

$$E(S) = \sum_{i=1}^{n} -Pr(C_i) * log_2 Pr(C_i)$$

Since the data is continuous, threshold value is found in the code by taking the median on the sorted attribute values where classification changes. The information gain for each attribute is calculated using the formula:

$$G(S, A) = E(S) - \sum_{i=1}^{m} Pr(A_i) E(S_{A_i})$$

Then select the attribute that has the highest information gain and this attribute is made the tree node. First calculating the Entropy for three classes of owl using E(S) = -Plog(P) . Then calculate Entropy for each attribute using two threshold values 0.6 and 1.6. Information gain gives “wing-width” as the root node as it has the highest gain. This is implemented by calling the function Infogain for all attributes.

Gain obtained for X1-body-length, X2- wing-length,X3- body-width and X4- wing-width

```
[1] 0.29161213 0.02729866 0.58198243 0.95135020
```

Now subset the data that contains only the objects containing wing-width > 0.6 and find the threshold, then calculated entropy and Information gain for this subset using the above procedure. Once again wing-width has the highest information gain and so is considered as the second node. Repeated the next iteration for the subset that contains data with attribute wing-width <= 1.6. After calculating the Information gain, “body-width” is considered as the third node. Subset the data containing only the objects with attribute body-width > 4.8 and find the threshold. This time wing-width has the highest information gain. Subset the data with attribute wing-width <= 1.5.

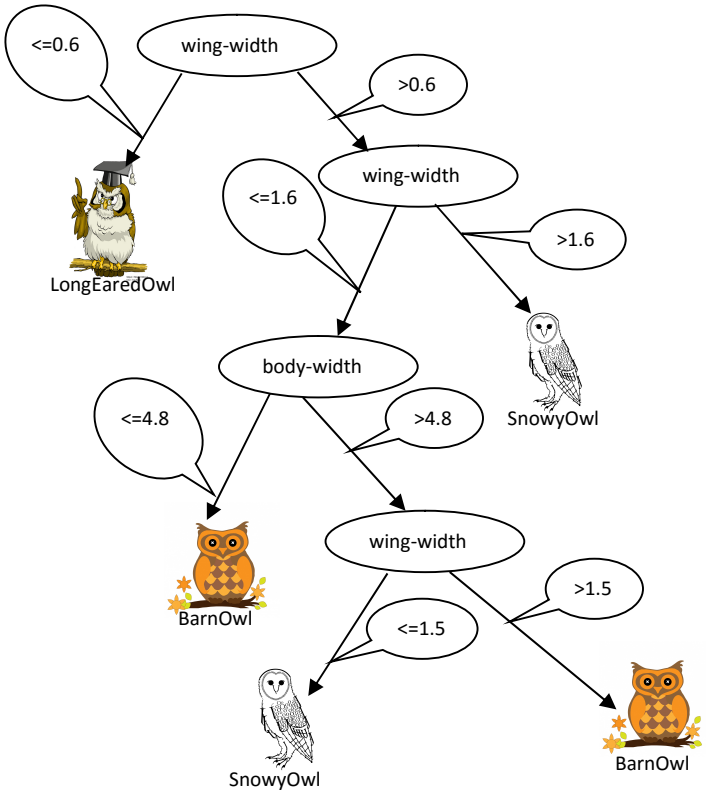
Attribute wing-width is the one that contains much more information and for this reason it has been selected as the first split criteria.

Testing the data:

The test samples that is (45 objects) 1/3rd of the given data is passed into the function Predict for the prediction of the class. If wing width is less than 0.6, it is classified as LongEaredOwl, otherwise if wing-width is greater than 1.6, then that fall in SnowyOwl. If wing- width is less than equal to 1.6, body-width

attribute is tested. If body-width is less than wqual to 4.8 it belongs to BarnOwl, otherwise if wing-width is greater than 1.5, it is classified as BarnOwl, else it is SnowyOwl.

The figure below shows the tree and the nodes for the given data:



The confusion matrix is evaluated that shows out of 45 testing samples 18 are LongEaredOwl, 13 are BarnOwl and 11 are SnowyOwl. However 1 sample is misclassified as both BarnOwl and SnowyOwl.

So Error rate is 1/45 = 0.022*100 = 2.2%.

However this result is changing every time I run the code as the training samples are taken in random. Error rate varies between 2% to 8%.

```
ConfusionMatrix
              Predict
               BarnOwl LongEaredOwl SnowyOwl
  BarnOwl           13            0        1
  LongEaredOwl       0           18        0
  SnowyOwl           0            0       11
```

Conclusion:

- The decision tree has classified 18 LongEaredOwl objects as LongEaredOwl, 13 BarnOwl objects as BarnOwl and 1 as SnowyOwl, leading in 1 misclassification.
- Average Accuracy is 95% (1-Error Rate).
- The algorithm tries to classify the training data so well and it becomes too specific to correctly classify the test data

Reference:

- http://data-mining.business-intelligence.uoc.edu/home/j48-decision-tree
- Lecture 2 Notes from BlackBoard http://datamining.it.nuigalway.ie/

Appendix

```
#Reading the data from csv file
ml <- read.csv("C:\\Users\\Sowmya\\Desktop\\NUI GALWAY\\Machine Learning\\Assignment 3\\owls15.csv")

#Running for 10 times
for(i in 1:10){

 #Dividing the data into 2/3 and 1/3 for training and testing
 index <- 1:nrow(ml)
trainindex <- sample(index, trunc(length(index)*(2/3)))
training_data <- ml[trainindex, ]
testing_data <- ml[-trainindex, ]


#Funtion to find Information Gain
Infogain<- function(x){
 X4<-sort(x)

 Y<-NULL
 for(i in 1:length(X4)-1){
  Y[i]<-X4[i+1]-X4[i]
 }

 #Finding the Threshold value
 #Sorting the attributes to find Threshold value

 #Threshold for  X4
 Threshold<-X4[which.max(Y)]

 #Count of LongEaredOwl(LEO), SnowyOwl(SO), BarnOwl(BO)

 Length_LEO<-length(which(training_data$owl=="LongEaredOwl"))
 Length_SO<-length(which(training_data$owl=="SnowyOwl"))
 Length_BO<-length(which(training_data$owl=="BarnOwl"))
 Length_Owl<-length(training_data$owl)
  #Calculating Entropy E(S) = -Plog(P)
 Entropy_S <- (-(Length_LEO/Length_Owl)*(log2(Length_LEO/Length_Owl)))+
  (-(Length_SO/Length_Owl)*(log2(Length_SO/Length_Owl)))+
  (-(Length_BO/Length_Owl)*(log2(Length_BO/Length_Owl)))
 #Calculating Entropy X4<=Threshold  E(X4)= (-P_LOlog(P_LO))+(-P_BOlog(P_BO))+(-P_SOlog(P_SO))
X4_ThLess<-which(training_data$X4<=Threshold)
 Count_Less<-length(X4_ThLess)
 Count_LO<-0
 Count_SO<-0
 Count_BO<-0
for(i in 1:length(x)){
  if(x[i]<=Threshold){
    if(training_data$owl[i]=="LongEaredOwl"){Count_LO=Count_LO+1}else
    {if(training_data$owl[i]=="SnowyOwl"){Count_SO=Count_SO+1}else
    {Count_BO=Count_BO+1}}

  }}

 #Calculating Entropy
 S1<-0
 S2<-0
 S3<-0
 if(Count_LO!=0){S1<-(Count_LO/Count_Less)*(log2(Count_LO/Count_Less))}
 if(Count_SO!=0){S2<-(Count_SO/Count_Less)*(log2(Count_SO/Count_Less))}
```

```
 if(Count_BO!=0){S3<-(Count_BO/Count_Less)*(log2(Count_BO/Count_Less))}

 Entropy_Less<-(-S1)+(-S2)+(-S3)

 #Calculating Entropy X4<in between>Threshold  E(X4)= (-P_LOlog(P_LO))+(-P_BOlog(P_BO))+(-P_SOlog(P_SO))

 subset_index<-which.max(Y)
 Threshold_median <- X4[((subset_index + 1) + length(X3))/2]
 count_l<-count_middle<-count_s<-count_b<-0
 for(i in 1:length(x)){
  if(x[i]>Threshold && x[i]<=Threshold_median){
   {
    count_middle<-count_middle+1
    if(training_data$owl[i]=="LongEaredOwl"){
     count_l=count_l+1
    }else if(training_data$owl[i]=="SnowyOwl"){
     count_s=count_s+1
    }else if(training_data$owl[i]=="BarnOwl"){
     count_b=count_b+1
    }
   }
  }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_middle)*(log2(count_l/count_middle)))
 E2<-ifelse(count_s==0,0,(count_s/count_middle)*(log2(count_s/count_middle)))
 E3<-ifelse(count_b==0,0,(count_b/count_middle)*(log2(count_b/count_middle)))
 Entropy_Middle<- -E1-E2-E3


 #Calculating Entropy X4>Threshold  E(X4)= (-P_LOlog(P_LO))+(-P_BOlog(P_BO))+(-P_SOlog(P_SO))
count_l<-count_more<-count_s<-count_b<-0
 for(i in 1:length(x)){
  if(x[i]>Threshold_median){
   {
    count_more<-count_more+1
    if(training_data$owl[i]=="LongEaredOwl"){
     count_l=count_l+1
    }else if(training_data$owl[i]=="SnowyOwl"){
     count_s=count_s+1
    }else if(training_data$owl[i]=="BarnOwl"){
     count_b=count_b+1
    }
   }
  }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_more)*(log2(count_l/count_more)))
 E2<-ifelse(count_s==0,0,(count_s/count_more)*(log2(count_s/count_more)))
 E3<-ifelse(count_b==0,0,(count_b/count_more)*(log2(count_b/count_more)))
 Entropy_greater<- -E1-E2-E3
Information_gain <- Entropy_S - ((Count_Less/Length_Owl)*(Entropy_Less)) -
  ((count_middle/Length_Owl)*(Entropy_Middle)) -
  ((count_more/Length_Owl)*(Entropy_greater))
 return(Information_gain)
}


#Calling functions to calculate Information gain for 4 attributes

GainX4<-Infogain(training_data$X4)
GainX3<-Infogain(training_data$X3)
```

```
GainX2<-Infogain(training_data$X2)
GainX1<-Infogain(training_data$X1)

#To find the the attribute with highest Informain gain

Gain<-c(GainX1,GainX2,GainX3,GainX4)
name<-names(ml)
Information_Gain_Max_1<-name[which.max(Gain)]

#Second Iteration depending on the Information gain: X4 is higher , hence X4 is the root node
#Subsetting the data for those values of X4 greater than the Threshold

Second_Set<-subset(training_data, training_data$X4 > Threshold)

#Finding the second Threshold
Threshold2<- median(Second_Set$X4)

# Calculating Information gain for the second iteration
InfoGain <-function(x){
  count_l<-count_total_greater<-count_s<-count_b<-0
 for(i in 1:length(x)){
   if(x[i]>Threshold){
     {
       count_total_greater<-count_total_greater+1
       if(training_data$owl[i]=="LongEaredOwl"){
         count_l=count_l+1
       }else if(training_data$owl[i]=="SnowyOwl"){
         count_s=count_s+1
       }else if(training_data$owl[i]=="BarnOwl"){
         count_b=count_b+1
       }
     }
   }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_total_greater)*(log2(count_l/count_total_greater)))
 E2<-ifelse(count_s==0,0,(count_s/count_total_greater)*(log2(count_s/count_total_greater)))
 E3<-ifelse(count_b==0,0,(count_b/count_total_greater)*(log2(count_b/count_total_greater)))
 entropy_greater<--E1-E2-E3

 Total_Owl<-length(training_data$owl)
 Total_L<-length(which(training_data$owl=="LongEaredOwl"))
 Total_S<-length(which(training_data$owl=="SnowyOwl"))
 Total_B<-length(which(training_data$owl=="BarnOwl"))
 #Calculating Entropy E(S) = -Plog(P)
 Entropy_S <- (-(Total_S/Total_Owl)*(log2(Total_S/Total_Owl)))+
   (-(Total_B/Total_Owl)*(log2(Total_B/Total_Owl)))

 #Calculating Entropy X4<in between>Threshold  E(X4)= (-P_LOlog(P_LO))+(-P_BOlog(P_BO))+(-P_SOlog(P_SO))

 count_l<-count_total_middle<-count_s<-count_b<-0
 for(i in 1:length(x)){
   if(x[i]>Threshold & x[i]<Threshold2){
     {

       count_total_middle<-count_total_middle+1
       if(training_data$owl[i]=="LongEaredOwl"){
         count_l=count_l+1
       }else if(training_data$owl[i]=="SnowyOwl"){
         count_s=count_s+1
       }else if(training_data$owl[i]=="BarnOwl"){
```

```
      count_b=count_b+1
     }
    }
   }
  }

 E1<-ifelse(count_l==0,0,(count_l/count_total_middle)*(log2(count_l/count_total_middle)))
 E2<-ifelse(count_s==0,0,(count_s/count_total_middle)*(log2(count_s/count_total_middle)))
 E3<-ifelse(count_b==0,0,(count_b/count_total_middle)*(log2(count_b/count_total_middle)))
 entropy_middle<--E1-E2-E3

 #entropy of x4 for thres1hold greater than situation
 count_l<-count_total_lesser<-count_s<-count_b<-0
 for(i in 1:length(x)){
  if(x[i]<=Threshold){
   {
    count_total_lesser<-count_total_lesser+1
    if(training_data$owl[i]=="LongEaredOwl"){
     count_l=count_l+1
    }else if(training_data$owl[i]=="SnowyOwl"){
     count_s=count_s+1
    }else if(training_data$owl[i]=="BarnOwl"){
     count_b=count_b+1
    }
   }
  }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_total_lesser)*(log2(count_l/count_total_lesser)))
 E2<-ifelse(count_s==0,0,(count_s/count_total_lesser)*(log2(count_s/count_total_lesser)))
 E3<-ifelse(count_b==0,0,(count_b/count_total_lesser)*(log2(count_b/count_total_lesser)))
 entropy_lesser<--E1-E2-E3

 #Information Gain
 Information_Gain_2<-Entropy_S-(count_total_greater/length(Second_Set$owl))*entropy_greater-
(count_total_lesser/length(Second_Set$owl))*entropy_inbet-(count_total_inbet/length(Second_Set$owl))*entropy_inbet
 return(Information_Gain_2)
}
GainX1<-InfoGain(Second_Set$X1)
GainX2<-InfoGain(Second_Set$X2)
GainX3<-InfoGain(Second_Set$X3)
GainX4<-InfoGain(Second_Set$X4)


#To find the the attribute with highest Informain gain

Gain<-c(GainX1,GainX2,GainX3,GainX4)
name<-names(ml)
Information_Gain_Max_2<-name[which.max(abs(Gain))]

#Third Iteration depending on the Information gain: X4 is higheragain , hence X4 is the next node
#Subsetting the data for those values of X3 greater than the Threshold3 and X4 less than Threshold 2


Threshold3<-median(Second_Set$X3)
Third_Subset<-subset(training_data, training_data$X4>Threshold & training_data$X4 <= Threshold2)

InfoGain<-function(x){
 count_l<-count_total_greater<-count_s<-count_b<-0
 for(i in 1:length(x)){
  if(x[i]>Threshold3){
```

```
    {
     count_total_greater<-count_total_greater+1
     if(training_data$owl[i]=="LongEaredOwl"){
      count_l=count_l+1
     }else if(training_data[i]=="SnowyOwl"){
      count_s=count_s+1
     }else if(training_data[i]=="BarnOwl"){
      count_b=count_b+1
     }
    }
   }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_total_greater)*(log2(count_l/count_total_greater)))
 E2<-ifelse(count_s==0,0,(count_s/count_total_greater)*(log2(count_s/count_total_greater)))
 E3<-ifelse(count_b==0,0,(count_b/count_total_greater)*(log2(count_b/count_total_greater)))
 entropy_greater<--E1-E2-E3

 Total_Owl<-length(training_data$owl)
 Total_L<-length(which(training_data$owl=="LongEaredOwl"))
 Total_S<-length(which(training_data$owl=="SnowyOwl"))
 Total_B<-length(which(training_data$owl=="BarnOwl"))


 #Calculating Entropy E(S) = -Plog(P)
 Entropy_S <- (-(Total_L/Total_Owl)*(log2(Total_L/Total_Owl)))+
  (-(Total_S/Total_Owl)*(log2(Total_S/Total_Owl)))+
  (-(Total_B/Total_Owl)*(log2(Total_B/Total_Owl)))

#entropy of x for Threshold greater than X3 situation
 count_l<-count_total_lesser<-count_s<-count_b<-0
 for(i in 1:length(x)){
  if(x[i]<=Threshold3){
    {
    count_total_lesser<-count_total_lesser+1
    if(training_data$owl[i]=="LongEaredOwl"){
     count_l=count_l+1
    }else if(training_data$owl[i]=="SnowyOwl"){
     count_s=count_s+1
    }else if(training_data$owl[i]=="BarnOwl"){
     count_b=count_b+1
    }
  }
  }
 }
 E1<-ifelse(count_l==0,0,(count_l/count_total_lesser)*(log2(count_l/count_total_lesser)))
 E2<-ifelse(count_s==0,0,(count_s/count_total_lesser)*(log2(count_s/count_total_lesser)))
 E3<-ifelse(count_b==0,0,(count_b/count_total_lesser)*(log2(count_b/count_total_lesser)))
 entropy_lesser<--E1-E2-E3

 #Information Gain
 IG<-Entropy_S-(count_total_greater/length(training_data$owl))*entropy_lesser-
(count_total_lesser/length(training_data$owl))*entropy_lesser
 return(IG)
}
GainX1<-InfoGain(Third_Subset$X1)
#GainX2<-InfoGain(Third_Subset$X2)
#GainX3<-InfoGain(Third_Subset$X3)
GainX4<-InfoGain(Third_Subset$X4)
```

```
#To find the the attribute with highest Informain gain

Gain<-c(GainX1,GainX2,GainX3,GainX4)
name<-names(ml)
Information_Gain_Max_3<-name[which.max(abs(Gain))]

#Fourth Subset

Fourth_Subset<-subset(training_data,  training_data$X4<=Threshold2 & training_data$X3 > Threshold3)

Threshold4<-median(Fourth_Subset$X4)

}

#Predicting the testing samples

Prediction<-function(testing_data)
{
 Predict<-NULL
 for(i in 1:nrow(testing_data))
 {
   if(testing_data$X4[i]<=Threshold){
     Predict[i]<-"LongEaredOwl"
   }else if(testing_data$X4[i]>Threshold & testing_data$X4[i]>Threshold2){
     Predict[i]<-"SnowyOwl"
   }else if(testing_data$X4[i]>Threshold & testing_data$X4[i]<=Threshold2){
     if(testing_data$X3[i]<=Threshold3){
       Predict[i]<-"BarnOwl"
     }else if(testing_data$X4[i]>Threshold3 & testing_data$X4[i]>Threshold4){
       Predict[i]<-"BarnOwl"
     }else if(testing_data$X4[i]>Threshold3 & testing_data$X4[i]<=Threshold4){
       Predict[i]<-"SnowyOwl"
     }
   }
 }
 return(Predict)
}
Predict<-Prediction(testing_data)

ConfusionMatrix<-table(testing_data$owl,Predict
```