# ASSIGNMENT-2

1. Import the data to R. Copy the R code used below. (0.1)

organics.df <-read\_excel("organics.xlsx")

(b) Examine the distribution of the target variable: (1) plot a bar chart to show the number of

observations in each category, and (2) plot a bar chart to show the frequency of observations in each

category. Copy the code used and the resulting plots below. How many individuals purchased organic

products? What is the approximate proportion of individuals who purchased organic products? (0.1+0.1+0.1+0.1+0.1)

f <- factor(organics.df$TargetBuy)

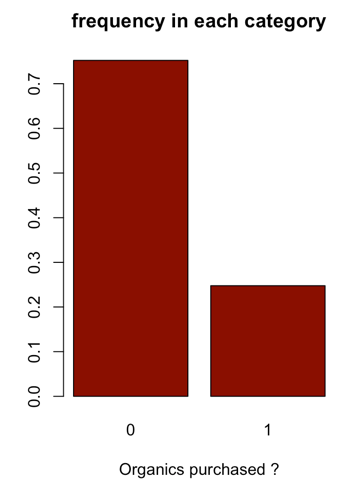
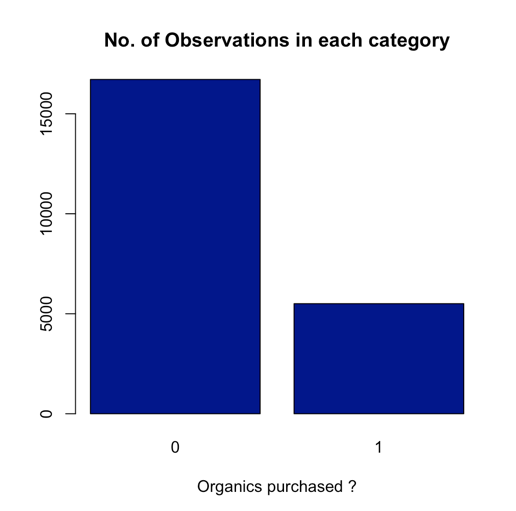
counts <- table(f)

barplot(counts, main="No. of Observations in each category",xlab="Organics purchased

?",col="darkblue")

h <- table(f) / length(f)

barplot(h, main="frequency in each category",xlab="Organics purchased ?",col="darkred")



We observe that 5505 customers purchased organic products and 16717 customers did

Not purchase the organic products. ~75% of customers did not purchase organic

products and 25% purchased.

It is an imbalanced data, which needs up sampling or down sampling for balancing the

Classes of the response variable.

(c) The variable DemClusterGroup contains collapsed levels of the variable DemCluster. Presume

that, based on previous experience, you believe that DemClusterGroup is sufficient for this type of

modeling effort. Exclude the variable DemCluster for the analysis. Copy the R code used below. (0.1)

organics.df<- select(organics.df,-c(3)

(d) As noted above, only TargetBuy will be used for this analysis and should have a role of target.

Can TargetAmt be used as an input for a model used to predict TargetBuy? Why or why not? (0.2+0.1)

# Drop variable DemCluster

organics.df<- select(organics.df,-c(12))

No, TargetAmt cannot be used as an input for a model used to predict the TargetBuy.

Because TargetAmt is itself one of the possible response variables, which is strongly correlated with TargetBuy. If the TargetAmt is zero (Number of organic products purchased) it intuitively means TargetBuy is 0 and if the TargetAmt is not zero it means TargetBuy is 1.

This will lead a completely wrong analysis.

(e) Partition the data: set records 1, 3, 5, … (the rows with odd numbers) as the training data, and set

records 2, 4, 6, … (the rows with even numbers) as the validation data, which results in 50%/50%

partition for training/validation. Copy the code used below. (0.5)

num <- c(1:nrow(organics.df))

numodd <- c()

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

if(num[i]%%2 != 0)

numodd[i] <- num[i]}

sample<- numodd[!is.na(numodd)]

train <- organics.df[sample, ]

validate <- organics.df[-sample, ]

(f) Implement a decision tree on the Training data to predict “TargetBuy” status. Plot the tree. Copy

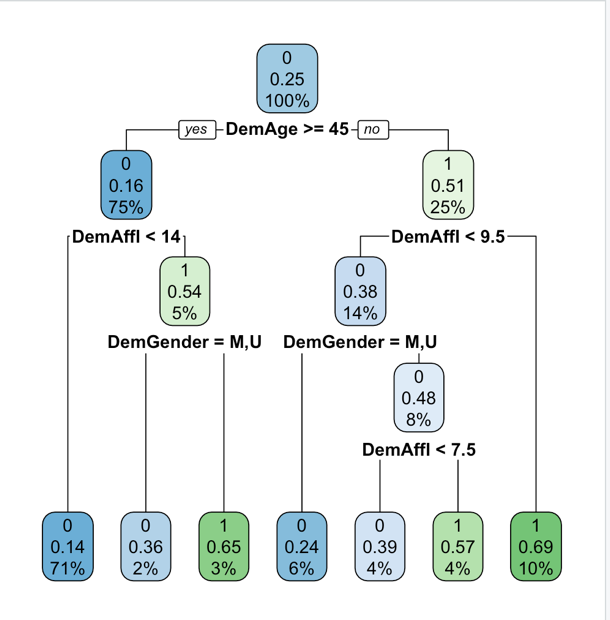
the code used and the result below. How many leaves are in the tree? Which variable was used for the

first split? Create a confusion matrix which shows the accuracy rate of your classification. Copy the

code used and the result below. (0.5+0.1+0.1+0.2)

Ttree <- rpart(TargetBuy ~ ., data = train, method = "class",control = rpart.control(cp=0.01))

rpart.plot(Ttree)

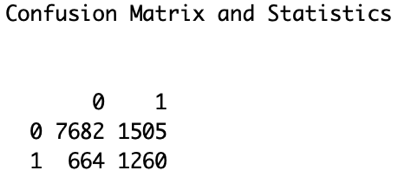


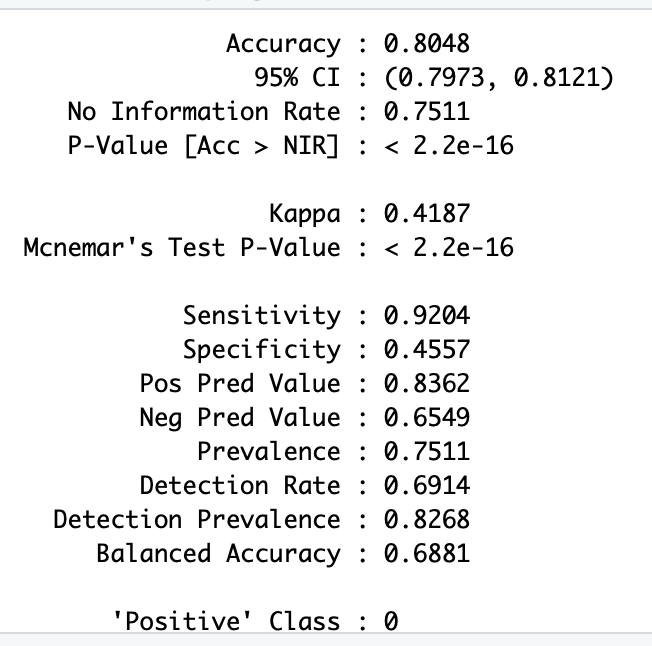
There are 7 leaf nodes and the variable used for first split is DemAge.

train$PredictClass <- predict(Ttree, newdata = train, type="class")

traintab <- table(train$PredictClass, train$TargetBuy)

confusionMatrix(traintab)





(g) Apply your decision tree from the training data to the validation data, and compare the accuracy

of classification of your validation and training data sets. Show the confusion matrix. Copy the code

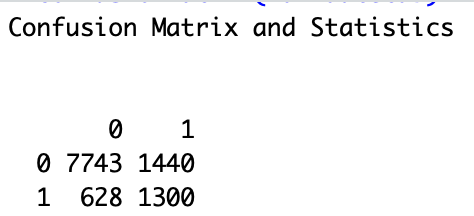
used and the results below. How is the accuracy using validation data different from that using

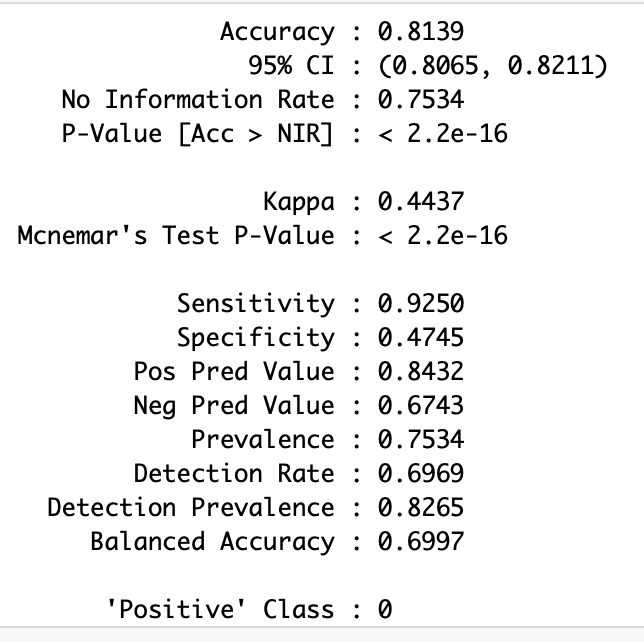
training data? Is this what you expected? Why? (0.5+0.5)

validate$PredictClass <- predict(Ttree, newdata = validate, type="class")

validatetab <- table(validate$PredictClass, validate$TargetBuy)

confusionMatrix(validatetab)





We observe the accuracy in the train data is 0.8048 and that of validation data it is 0.8139. They are almost same, in fact the validation dataset has little higher accuracy compared to train data.

This could be rare cases of observations, could be because of the data selection and data partition. Usually the validation data has less accuracy compared to test. We perform pruning to handle any overfitting issues in ideal case.

(h) Imposing maxdepth = 2, create another decision tree on the training data to predict TargetBuy

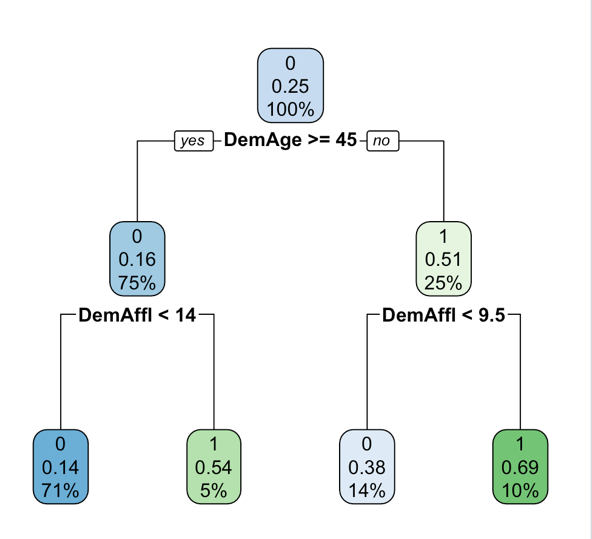
status. Plot the tree. Create a confusion matrix which shows the accuracy rate of your classification.

Copy the code used and the result below. How many leaves are in the tree? Compared with the tree in

(f), which one appears to be better? Is this what you expected? Why? (0.3+0.3+0.1+0.1+0.1)

Dtree <- rpart(train$TargetBuy ~ ., data = train, method = "class", maxdepth = 2)

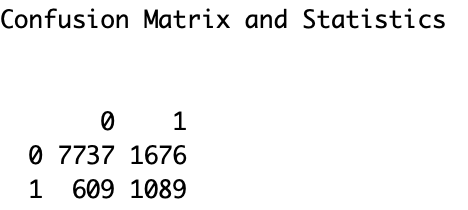
rpart.plot(Dtree)

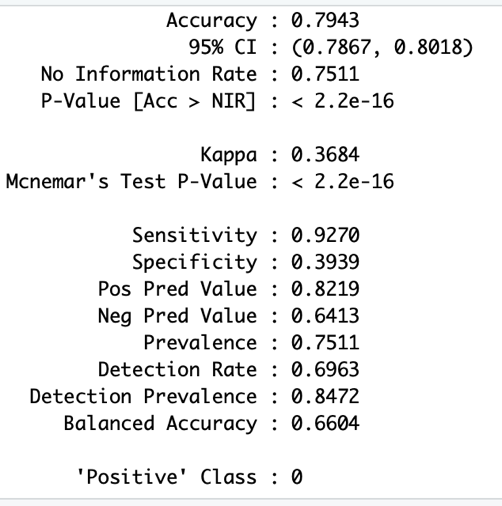


train$PredictClass <- predict(Dtree, newdata = train, type="class")

Dtraintab <- table(train$PredictClass, train$TargetBuy)

confusionMatrix(Dtraintab)





We observe 4 leaves in the tree above. The accuracy is 0.7943, which is bit less than (f). Taking accuracy into consideration we can choose the (f) one. We expect after pruning the accuracy might increases, but the goal is to choose the right depth (hyper parameter) to avoid both underfitting and overfitting. It appears depth = 2 is not the right value.

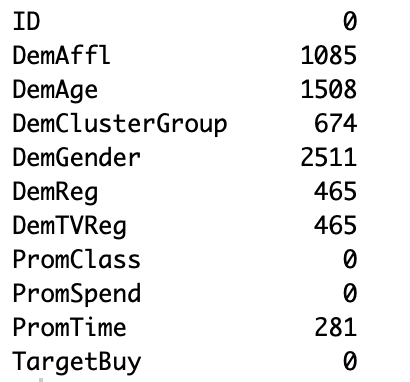
(i) Next, consider using a logistic regression model. First, are there any missing values? If so, is any

missing values imputation needed for logit model? Is imputation required before generating the

decision tree models and why? (0.3+0.1+0.1)

data.frame(miss.val=sapply(organics.df, function(x)

(sum(length(which(is.na(x)))))))



Yes, there are missing values. We observe good amount of missing information in many variables. Simply, ignoring data with missing values can lead to insufficient data being left to train a good model. As ‘missingness’ of attributes increases, it also affects the prediction accuracy. A blank response variable might also have meaning, implying some useful information. Accounting all these reasons both logistic and decision trees require missing values imputation. We can consider the missing values into a new category as “NA”, while performing logistic regression and decision trees.

(j) Impute: impute “U” for unknown class variable values and the overall mean for unknown interval

variable values. Copy the code used below. (0.5)

x <- data.frame(datatype = sapply(organics.df, is.numeric))

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

if ( x$datatype[i] == TRUE)

{

organics.df[is.na(organics.df[,i]), i] <- mean(as.vector(unlist(array(as.data.frame(organics.df[,i])))), na.rm = TRUE)

}

else{

organics.df[is.na(organics.df[,i]), i] <- 'U'

}}

(k) Use a logistic regression model to classify the data set using the same dependent variable,

TargetBuy. Copy the code used and the result below. (0.5)

model1 <- glm(TargetBuy~.,family=binomial(link='logit'),data=train)

summary(model1)

model2 <- glm(TargetBuy~.,family=binomial(link='logit'),data=validate)

summary(model2)

(l) Compare the performance of the logit model on the training and validation data sets by creating

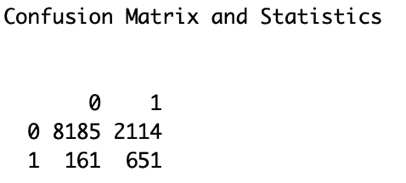
confusion matrixes which show the accuracy rates. Copy the code used and the result below. Which

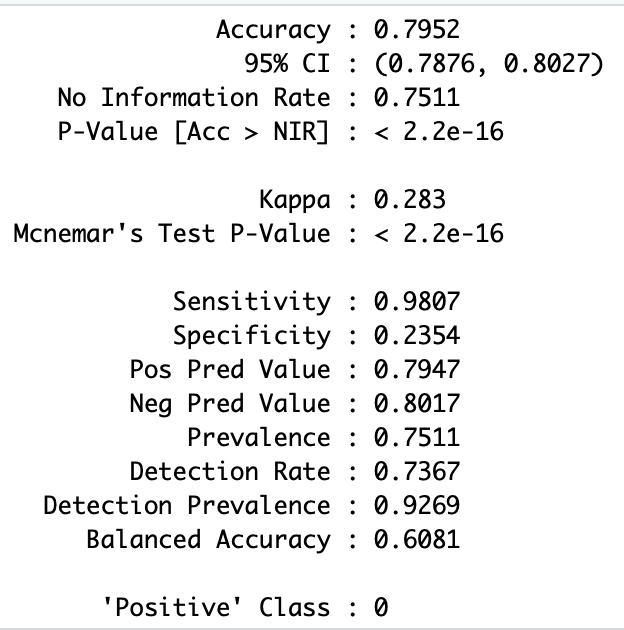
one appears to be better? (0.5+0.3+0.2)

train$pdata1 <- ifelse(predict(model1, train)> 0.5,1,0)

traintab <- table(train$pdata1, train$TargetBuy)

confusionMatrix(traintab)

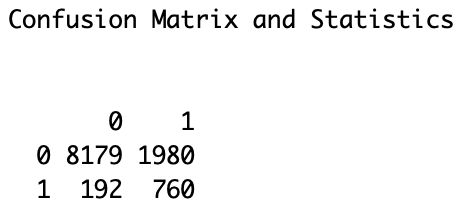


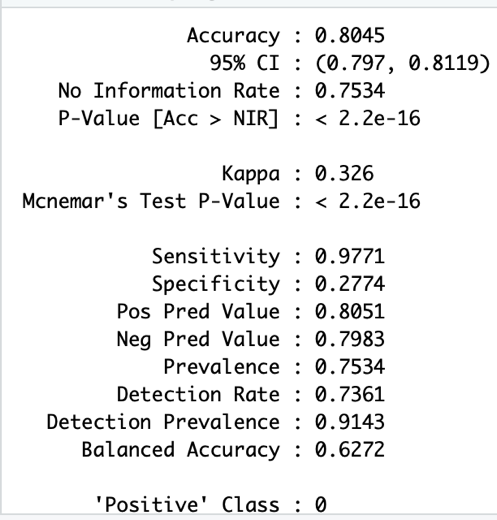


validate$pdata1 <- ifelse(predict(model2, validate)> 0.5,1,0)

validtab <- table(validate$pdata1, validate$TargetBuy)

confusionMatrix(validtab)





Here we set the threshold as 0.5(hyper parameter which can be tuned to get the best value).

We see validate accuracy is bit higher (0.8045) compared to train (0.7952). Again, a rare case could be an issue with sampling of data. K- fold cross validation should be adopted for better results.

(m) Plot ROC curves for the decision tree in (f) and the logit model using validation data. Summarize

each curve by its ROC index (“area under the curve (AUC)”). Copy the code used and the result

below. In terms of ROC index, which model is better? (0.5+0.5)

#Logistic Regression

# Compute AUC for predicting Class with the model

validate <- select(validate,-c(11))

prob <- predict(model1, newdata=validate, type="response")

pred <- prediction(prob, validate$TargetBuy)

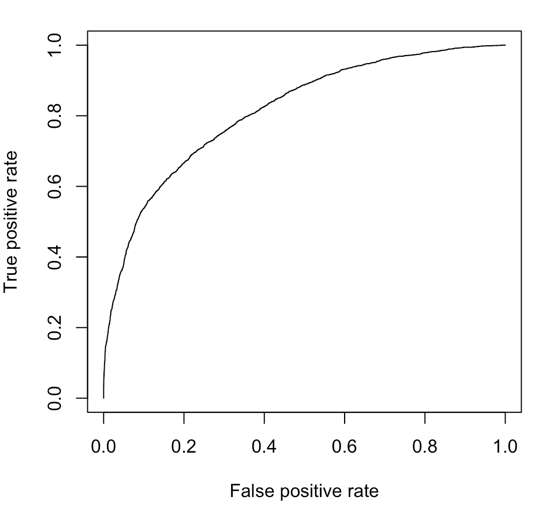
perf <- performance(pred, measure = "tpr", x.measure = "fpr")

plot(perf)

auc <- performance(pred, measure = "auc")

auc

## [1] 0.8148634



We observe ROC Index to be of 0.81.

#Decision Trees

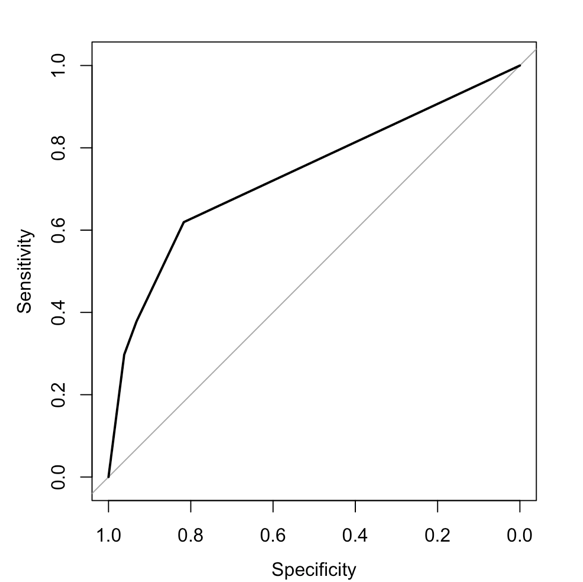
validate <- select(validate,-c(11))

prob1<- predict(Dtree, newdata = validate, type="prob")

auc <- auc(validate$TargetBuy, prob1[,2])

plot(roc(validate$TargetBuy, prob1[,2]))

# Area under the curve: 0.7346



We observe ROC Index to be of 0.73.

In terms of ROC Index, the metric ranges from 0.50 to 1.00, and values above 0.80 indicate that the model does a good job in discriminating between the two categories. We observe logistic regression does a better job.