Problem 1 (CART): Competitive auctions on eBay.com:

The file eBayAuctions.csv contains information on 1972 auctions that transacted on eBay.com during May-June in 2004. The goal is to use these data in order to build a model that will classify competitive auctions from non-competitive ones. A competitive auction is defined as an auction with at least 2 bids placed on the auctioned item. The data include variables that describe the auctioned item (auction category), the seller (his/her eBay rating) and the auction terms that the seller selected (auction duration, opening price, currency, day-of-week of auction close). In addition, we have the price that the auction closed at. The goal is to predict whether the auction will be competitive or not.

1. Note that in the dataset, the original variables of Category (11 categories), Currency (USD, nonUS), and EndDay (Weekend, Week) are categorical variables. Therefore, the dataset also contains their corresponding dummy variables.

2. Import the dataset and split the data into training and validation datasets using a 60%-40% ratio.
Ans: In order to import the data and split them into training/validation datasets, we use the following code:

```
HW3.R × | CIS8695_CART.R ×
Source on Save | Run | Source

rm(list = ls())
# Setting the Current Working Directory
setwd("/Users/bhavin/Documents/MSIS - Big Data Analytics/Fall 2021 Semester/Fall 2021 - Seme

# Installing Required Packages
library(rpart)
# install.packages("rpart.plot")
library(rpart.plot)

# Getting Rid of Dummy Variables
ebayAuctions.df <- read.csv("eBayAuctions.csv")
ebayAuctions.df <- ebayAuctions.df[, -c(6,7,8,19,20,22)]

# Partioning the Data
set.seed(1)
train.index <- sample(c(1:dim(ebayAuctions.df)[1]), dim(ebayAuctions.df)[1]*0.6)
train.df <- ebayAuctions.df[train.index, ]
valid.df <- ebayAuctions.df[-train.index, ]
```

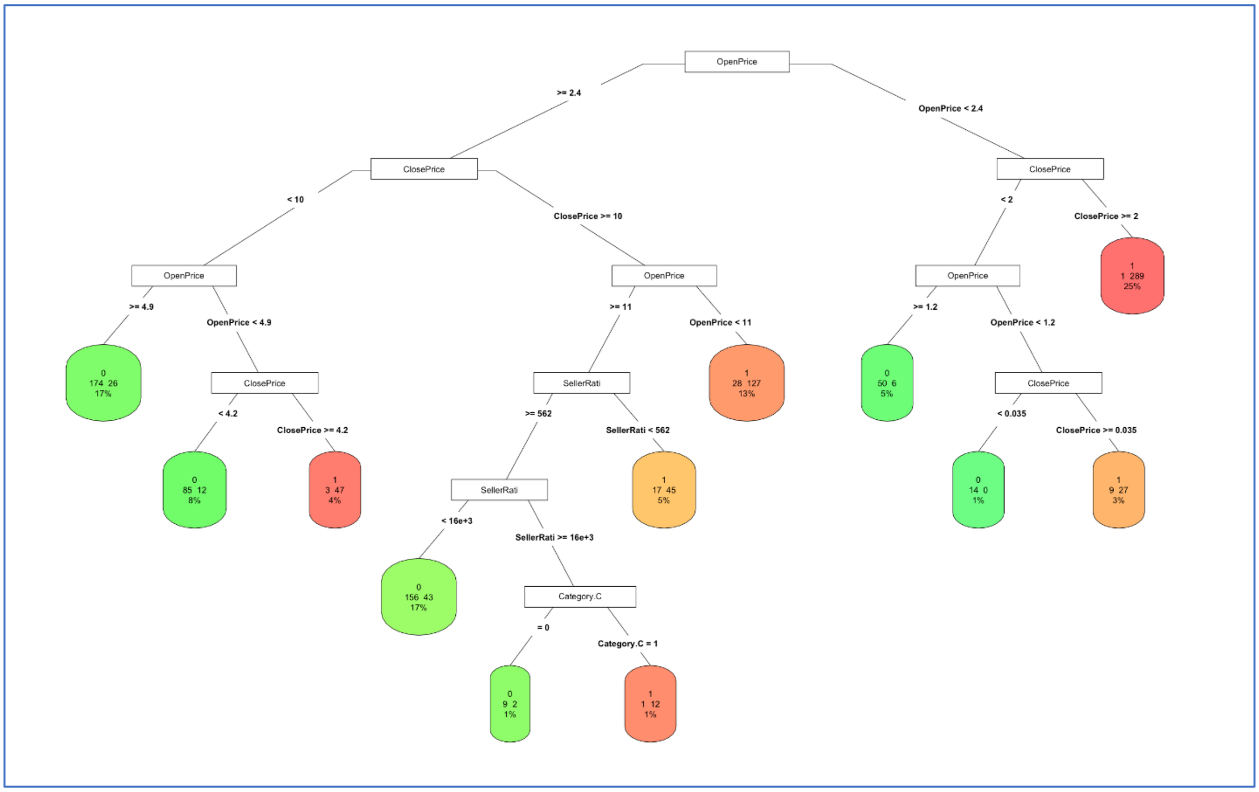
We also exclude one dummy variable from each group of dummy variables (Category_SportingGoods, Currency_nonUS and EndDay_Weekend). We fit the classification tree (setting maxdepth = 6), look at the rules and generate the confusion matrix.

```
HW3.R × | CIS8695_CART.R ×
Source on Save | Run | Source

# Classification Tree
default.ct <- rpart(Competitive ~ ., data = train.df, control = rpart.control(maxdepth=6),me
summary(default.ct)
printcp(default.ct)

# Plotting using prp()
prp(default.ct, type = 5, extra = 101,  clip.right.lab = FALSE,
    box.palette = "GnYlRd", leaf.round = 5,
    branch = .3, varlen = -10, space=0)

# Decision Rules
rpart.rules(default.ct, cover = TRUE)

# Classifying Records
default.ct.pred.train <- predict(default.ct,train.df,type = "class")
default.ct.pred.valid <- predict(default.ct,valid.df,type = "class")
install.packages("caret")
library(caret)
# Generating Confusion Matrix
confusionMatrix(default.ct.pred.valid, as.factor(valid.df$Competitive))
```

3. Fit a classification tree. Use Competitive as the output variable and the rest of variables as predictors. In the model, make sure that you exclude one dummy variable from each group of dummy variables (e.g., exclude Category_SportingGoods, Currency_nonUS and EndDay_Weekend). To avoid overfitting, set the maxdepth=6.

a. Report the tree (copy and paste the tree diagram).

Ans: The Classification Tree upon plotting is:



b. Report the prediction Confusion Matrix of Validation Data.

Ans: The Confusion Matrix generated is follows, accuracy observed is 86.31 %.

```
Console   Jobs ×
R 4.1.1 · ~/Documents/MSIS - Big Data Analytics/Fall 2021 Semester/Fall 2021 - Semester (M1)/CIS 8695 - Big Data Analytics (Ling Xue)/Homework Assignments/
> library(caret)
> # Generating Confusion Matrix
> confusionMatrix(default.ct.pred.valid, as.factor(valid.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 301  50
         1  58 380

               Accuracy : 0.8631
                 95% CI : (0.8371, 0.8863)
    No Information Rate : 0.545
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7235

 Mcnemar's Test P-Value : 0.5006

            Sensitivity : 0.8384
            Specificity : 0.8837
         Pos Pred Value : 0.8575
         Neg Pred Value : 0.8676
             Prevalence : 0.4550
         Detection Rate : 0.3815
   Detection Prevalence : 0.4449
```

c. What predictors are used by the tree?

Ans: The predictors used by the model were ClosedPrice, OpenPrice and SellerRating:

```
Console   Jobs ×
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big Data Analytics (Ling Xue)/Homework Assignments/

Classification tree:
rpart(formula = Competitive ~ ., data = train.df, method = "class",
    control = rpart.control(maxdepth = 6))

Variables actually used in tree construction:
[1] Category.Clothing.Toys ClosePrice            OpenPrice
[4] SellerRating

Root node error: 547/1183 = 0.46238

n= 1183

        CP nsplit rel error  xerror     xstd
1 0.290676      0   1.00000 1.00000 0.031350
2 0.090494      1   0.70932 0.73309 0.029764
3 0.073126      3   0.52834 0.54662 0.027326
4 0.051188      4   0.45521 0.48263 0.026181
5 0.040219      5   0.40402 0.43693 0.025247
6 0.016453      7   0.32358 0.34918 0.023136
7 0.010055      9   0.29068 0.33455 0.022738
8 0.010000     11   0.27057 0.32176 0.022376
>
```

d. List the decision rules. For example, if variable1<0 AND variable2<2, class=0.

Ans: The decision rules are shown as below:

```
Console   Jobs ×
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big Data Analytics (Ling Xue)/Homework Assignments/
> # Decision Rules
> rpart.rules(default.ct, cover = TRUE)
 Competitive
       cover
        0.00 when OpenPrice <  1.2        & ClosePrice <   0.035
          1%
        0.11 when OpenPrice is 1.2 to  2.4 & ClosePrice <   2.025
          5%
        0.12 when OpenPrice is 2.4 to  4.9 & ClosePrice <   4.195
          8%
        0.13 when OpenPrice >=        4.9 & ClosePrice <  10.000
         17%
        0.18 when OpenPrice >=       11.1 & ClosePrice >=          10.000 & SellerRating >=        16284 & Category.Clothing.Toys
 is 0      1%
        0.22 when OpenPrice >=       11.1 & ClosePrice >=          10.000 & SellerRating is  562 to 16284
         17%
        0.73 when OpenPrice >=       11.1 & ClosePrice >=          10.000 & SellerRating <   562
          5%
        0.75 when OpenPrice <  1.2        & ClosePrice is  0.035 to  2.025
          3%
        0.82 when OpenPrice is 2.4 to 11.1 & ClosePrice >=          10.000
         13%
        0.92 when OpenPrice >=       11.1 & ClosePrice >=          10.000 & SellerRating >=        16284 & Category.Clothing.Toys
 is 1      1%
        0.94 when OpenPrice is 2.4 to  4.9 & ClosePrice is  4.195 to 10.000
          4%
```

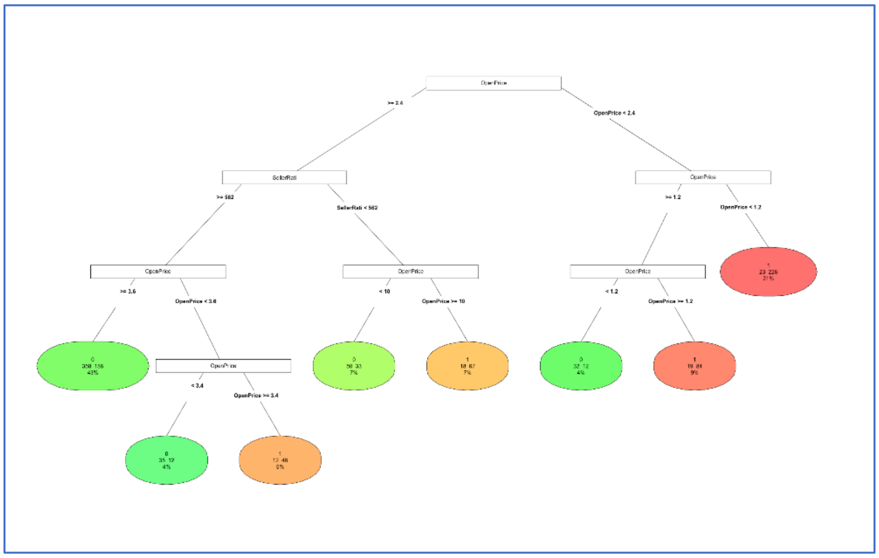
4. Are the rules practical for predicting the outcome of a new auction? Explain why (Hint: are you able to use the rules to classify a new auction before the auction ends? Do you know the values of all predictors in the rules before the auction ends? Some of them may not be known before the end of auction. What are them?). What variables should NOT be included in the predictor set? Explain why.

Ans: The rules are not practical for predicting the outcome of a new auction because the auction uses "ClosePrice" as a predictor for classifying the auction, which cannot be known before the auction ends. Hence, this classification is not possible and the predictor "ClosePrice" must be removed from the list of predictors to predict the outcome.

5. Fit another classification tree using the same setting in question 3. This time only use the predictors that can be used for predicting the outcome of a new auction. 1

a. Report the tree (copy and paste the tree diagram).

Ans: Now, we remove the ClosePrice as predictor and repeat the steps for Classification Tree:



b. Report the prediction Confusion Matrix of Validation Data.

Ans: The prediction Confusion Matrix of Validation data is as follows:

```
Console   Jobs
R 4.1.1 · ~/Documents/MSIS - Big Data Analytics/Fall 2021 Semester/Fall 2021 - Semester (M1)/CIS 8695 - Big I
> library(caret)
> # Generating Confusion Matrix
> confusionMatrix(defaultExclude.ct.pred.valid, as.factor(validExclude.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 288 130
         1  71 300

               Accuracy : 0.7452
                 95% CI : (0.7133, 0.7753)
    No Information Rate : 0.545
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4932

 Mcnemar's Test P-Value : 4.295e-05

            Sensitivity : 0.8022
            Specificity : 0.6977
         Pos Pred Value : 0.6890
         Neg Pred Value : 0.8086
             Prevalence : 0.4550
         Detection Rate : 0.3650
   Detection Prevalence : 0.5298
      Balanced Accuracy : 0.7500

       'Positive' Class : 0
```

c. What predictors are used by the tree?

Ans: The predictors used by the Classification Tree are OpenPrice and SellerRating.

```
Console   Jobs
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big I
Classification tree:
rpart(formula = Competitive ~ ., data = trainExclude.df, method = "class",
    control = rpart.control(maxdepth = 6))

Variables actually used in tree construction:
[1] OpenPrice    SellerRating

Root node error: 547/1183 = 0.46238

n= 1183

        CP nsplit rel error  xerror     xstd
1 0.290676      0   1.00000 1.00000 0.031350
2 0.058501      1   0.70932 0.73309 0.029764
```

d. List the decision rules.

Ans: The decision rules are as follows:

```
Console   Jobs
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big
> # Decision Rules
> rpart.rules(defaultExclude.ct, cover = TRUE)
 Competitive                                                   cover
        0.26 when OpenPrice is 2.4 to  3.4 & SellerRating >= 562   4%
        0.27 when OpenPrice is 1.2 to  1.2                         4%
        0.30 when OpenPrice >=         3.6 & SellerRating >= 562  43%
        0.40 when OpenPrice is 2.4 to 10.0 & SellerRating <  562   7%
        0.79 when OpenPrice >=        10.0 & SellerRating <  562   7%
        0.79 when OpenPrice is 3.4 to  3.6 & SellerRating >= 562   5%
        0.82 when OpenPrice is 1.2 to  2.4                         9%
        0.91 when OpenPrice <  1.2                                21%
>
```

6. Examine and compare the summary reports in questions 3 and 5. Compare the overall performance (e.g., accuracy or error rates) between these two decision trees. Which model has better predictive performance? Explain why.

Ans: The Confusion Matrix for the first model and second model is shown. From the aforementioned information, we can conclude that the First Model has an accuracy of 0.8631 (86.31%), while the second model has an accuracy of 0.7452 (74.52%). The first model has better predictive performance as it includes the addition of ClosingPrice variable. However, the impact of this variable has no effect on the outcome, as it cannot be used for newer auctions.

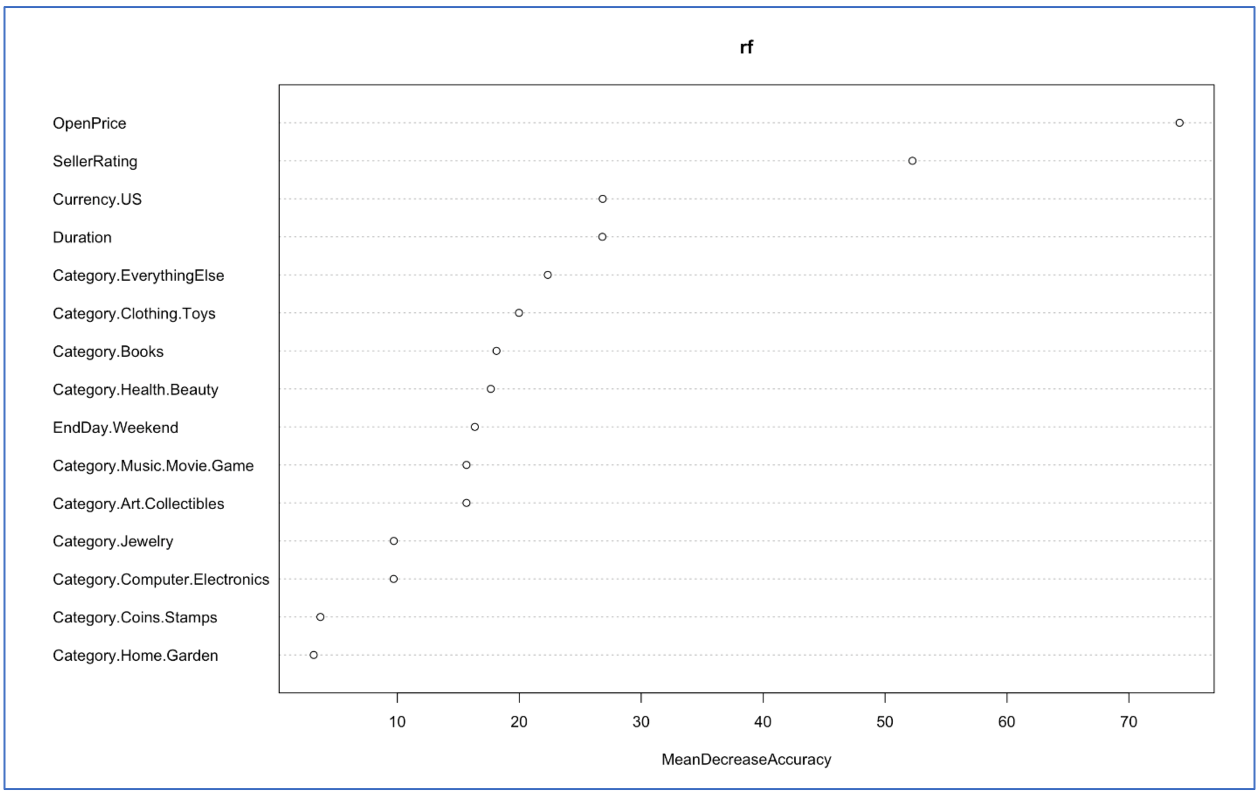
```
Console   Jobs
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big
> confusionMatrix(default.ct.pred.valid, as.factor(valid.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 301  50
         1  58 380

               Accuracy : 0.8631
                 95% CI : (0.8371, 0.8863)
    No Information Rate : 0.545
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7235

 Mcnemar's Test P-Value : 0.5006

            Sensitivity : 0.8384
            Specificity : 0.8837
         Pos Pred Value : 0.8575
         Neg Pred Value : 0.8676
             Prevalence : 0.4550
         Detection Rate : 0.3815
   Detection Prevalence : 0.4449
      Balanced Accuracy : 0.8611

       'Positive' Class : 0
```

```
Console    Jobs
R 4.1.1 · ~/Documents/MSIS - Big Data Analytics/Fall 2021 Semester/Fall 2021 - Semester (M1)/CIS 8695 - Big
> confusionMatrix(defaultExclude.ct.pred.valid, as.factor(validExclude.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 288 130
         1  71 300

               Accuracy : 0.7452
                 95% CI : (0.7133, 0.7753)
    No Information Rate : 0.545
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4932

 Mcnemar's Test P-Value : 4.295e-05

            Sensitivity : 0.8022
            Specificity : 0.6977
         Pos Pred Value : 0.6890
         Neg Pred Value : 0.8086
             Prevalence : 0.4550
         Detection Rate : 0.3650
   Detection Prevalence : 0.5298
      Balanced Accuracy : 0.7500

       'Positive' Class : 0
```

7. Build a random forest model for this prediction problem. Report:

a. Variable importance.

Ans: The variables of importance for this prediction problem are:



b. The prediction Confusion Matrix of Validation Data

Ans: The prediction Confusion Matrix can be seen below and the accuracy attained is 0.7731 (77.31%).

```
Console   Jobs
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big
> confusionMatrix(rf.pred, as.factor(validExclude.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 287 107
         1  72 323

               Accuracy : 0.7731
                 95% CI : (0.7423, 0.8019)
    No Information Rate : 0.545
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5462

 Mcnemar's Test P-Value : 0.01104

            Sensitivity : 0.7994
            Specificity : 0.7512
         Pos Pred Value : 0.7284
         Neg Pred Value : 0.8177
             Prevalence : 0.4550
         Detection Rate : 0.3638
   Detection Prevalence : 0.4994
```

Problem 2 (Naïve Bayes Classifier):

The file accidentsFull.csv contains information on over 42,183 actual automobile accidents in 2001 in the United States that involved one of three levels of injury: NO INJURY, INJURY, or FATALITY. For each accident, additional information is recorded, such as day of week, weather conditions, and road type. A firm might be interested in developing a system for quickly classifying the severity of an accident based on initial reports and associated data in the system (some of which rely on GPS-assisted reporting).

Our goal here is to predict whether an accident just reported will involve an injury (MAX_SEV_IR = 1 or 2) or will not (MAX_SEV_IR = 0). For this purpose, create a dummy variable called INJURY that takes the value "yes" if MAX_SEV_IR = 1 or 2, and otherwise "no."

a. Using the information in this dataset, if an accident has just been reported and no further information is available, what should the prediction be? (INJURY = Yes or No?) Why

Ans: From the dataset, we can observe that 20,721 accidents resulted without any injury, whereas 21,462 accidents involved minor and fatal injuries. We can calculate the probability around 51%. Using this information alone, the given probability calculates the prediction of an accident that is resulting in injury.:

```
Console   Jobs ×
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big
> injury.tb <- table(accidents.df$INJURY)
> show(injury.tb)

   no   yes
20721 21462
```

```
Console   Jobs ×
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big I

   no   yes
20721 21462
>
> # Calculating the Prediction
> injury.prob =  scales::percent(injury.tb["yes"]/(injury.tb["yes"]+injury.tb["no"]),0.0
1)
> injury.prob
     yes
"50.88%"
```

c. Let us now return to the entire dataset. Partition the data into training (60%) and validation (40%).

(i) Assuming that no information or initial reports about the accident itself are available at the time of prediction (only location characteristics, weather conditions, etc.), which predictors can we include in the analysis? (Use the Data_Codes sheet.)

Ans: i) If there is no initial report about the accident, then the predictors that we can use for analysis are: WEATHER_R (Weather Report), WKDY_I_R (Weekday or Weekend), HOUR_I_R (Rush Hour), INJURY_CRASH (Injury Crash)

(iv) What is the percent improvement relative to the naive rule (using the validation set)?

Ans: Comparing the Accuracies, we can see the percent improvement:

Training Set: 0.9902

Validation Set: 0.9913

Percent Improvement = (0.9913 – 0.9902) / 0.9902 * 100 = -0.11108%

```
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Bi
> # Predicting the Probabilities
> pred.prob <- predict(injury.nb, newdata = train.df, type = "raw")
> ## predict class membership
> pred.class <- predict(injury.nb, newdata = train.df)
> confusionMatrix(as.factor(pred.class), as.factor(train.df$INJURY))
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  12400   237
       yes    11 12661

               Accuracy : 0.9902
                 95% CI : (0.9889, 0.9914)
    No Information Rate : 0.5096
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9804

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9991
            Specificity : 0.9816
         Pos Pred Value : 0.9812
         Neg Pred Value : 0.9991
             Prevalence : 0.4904
         Detection Rate : 0.4899
   Detection Prevalence : 0.4993
      Balanced Accuracy : 0.9904

       'Positive' Class : no
```

```
R 4.1.1 · ~/Documents/MSIS – Big Data Analytics/Fall 2021 Semester/Fall 2021 – Semester (M1)/CIS 8695 – Big
       'Positive' Class : no

>
> ## predict probabilities: Validation
> pred.prob <- predict(injury.nb, newdata = valid.df, type = "raw")
> ## predict class membership
> pred.class <- predict(injury.nb, newdata = valid.df)
> confusionMatrix(as.factor(pred.class), as.factor(valid.df$INJURY))
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  8306  142
       yes    4 8422

               Accuracy : 0.9913
                 95% CI : (0.9898, 0.9927)
    No Information Rate : 0.5075
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9827

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9995
            Specificity : 0.9834
         Pos Pred Value : 0.9832
         Neg Pred Value : 0.9995
             Prevalence : 0.4925
         Detection Rate : 0.4922
   Detection Prevalence : 0.5007
      Balanced Accuracy : 0.9915

       'Positive' Class : no

>
```