<u>Problems on Auctions on eBay.com</u>

1) **Question (1) -** Conversion of 'Duration' column to categorical values and data split

Before conversion:

```
> unique(eBayAuctions$Duration)
[1]  5  7  1  3 10
```

Code and results after conversion:

```
> # Define bin edges
> bin_edges <- c(0, 4, 8, 10)  # Adjust the bin edges based on your data
> # Define bin labels
> bin_labels <- c( "Short", "Medium", "Long")
> # Use cut() to create a categorical variable
> categorical_variable <- cut(eBayAuctions$Duration, breaks = bin_edges, labels = bin_labels, include.lowest =
TRUE)
> # Create a data frame for display
> df <- data.frame(eBayAuctions$Duration, categorical_variable)
> eBayAuctions$Duration <- categorical_variable
> df
  eBayAuctions.Duration categorical_variable
1                     5               Medium
2                     5               Medium
3                     5               Medium
4                     5               Medium
```

eBayAuctions table 'duration' column:

```
> head(eBayAuctions)
                  Category currency sellerRating Duration endDay ClosePrice OpenPrice Competitive
1273 Clothing/Accessories      EUR         4153     Long    Sat      40.58      3.69           0
1271          Photography       US          121    Short    Sun     197.50      0.50           1
823          Toys/Hobbies       US         3277    Short    Sun       5.00      5.00           0
1749          Collectibles      US          985   Medium    Sun      75.00      9.99           1
1034 Clothing/Accessories      EUR          158    Short    Thu       8.49      8.49           0
1183 Clothing/Accessories      EUR         2016   Medium    Fri      18.32      6.02           0
```

We shuffled the data prior to split because the original dataset was not shuffled:

	Category	currency	sellerRating	Duration	endDay	ClosePrice	OpenPrice	Competitive.
1	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
2	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
3	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
4	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
5	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
6	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0
7	Music/Movie/Game	US	3249	Medium	Mon	0.01	0.01	0

Showing 1 to 7 of 1,972 entries, 8 total columns

Shuffling task and resulting data:

```
> set.seed(60)
> shuffle_index <- sample(1:nrow(eBayAuctions))
> head(shuffle_index)
[1] 947 592 258 549  90 116
> eBayAuctions <- eBayAuctions[shuffle_index, ]
```

Commented [1]: For this assignment, I think its easier if we create the entire script first and then create the document with the snapshot. Meanwhile, let's just copy-paste our codes here to follow each other's work. What do you think?

Commented [2]: Sounds good, Usman!

```
> head(eBayAuctions)
                Category currency sellerRating Duration endDay ClosePrice OpenPrice Competitive
1273 Clothing/Accessories      EUR         4153     Long    Sat      40.58      3.69           0
1271         Photography       US          121    Short    Sun     197.50      0.50           1
823          Toys/Hobbies      US         3277    Short    Sun       5.00      5.00           0
1749         Collectibles      US          985   Medium    Sun      75.00      9.99           1
1034 Clothing/Accessories      EUR          158    Short    Thu       8.49      8.49           0
1183 Clothing/Accessories      EUR         2016   Medium    Fri      18.32      6.02           0
> tail(eBayAuctions)
                Category currency sellerRating Duration endDay ClosePrice OpenPrice Competitive
1653          Automotive       US          957   Medium    Tue      19.99     19.99           0
1824          Automotive       US          671   Medium    Sun      39.95     39.95           0
722  Clothing/Accessories      EUR          241   Medium    Sun      23.97      1.23           1
31       Music/Movie/Game      US         3249   Medium    Mon       0.55      0.01           1
155      Music/Movie/Game      GBP         1029   Medium    Thu       1.25      1.25           0
699        SportingGoods       US        27132   Medium    Mon      17.01      0.99           1
```

Splitting the data into training 60% and validation 40% datasets.

```
> # Creating random samples for data preparation
> #Sampling training data with 60% of the total
> train.rows <- sample(rownames(eBayAuctions), dim(eBayAuctions)[1]*0.6)
> #Sampling validation data with 40% of the total
> valid.rows <- sample(setdiff(rownames(eBayAuctions), train.rows),dim(eBayAuctions)[1]*0.4)
> #Populating samples
> train.eBayAuctions <- eBayAuctions[train.rows, ]
> valid.eBayAuctions <- eBayAuctions[valid.rows, ]
```

Dimensions of resulting training and validation datasets:

```
> dim(train.eBayAuctions)
[1] 1183    8
> dim(valid.eBayAuctions)
[1] 788   8
```

Question (1a)

Creating a classification tree using all predictors, using the best-pruned tree with minbucket = 50 and maxdepth = 7.

First we created the default tree:

```
> #Best-pruned decision tree according to the assignment document.
> fit <- rpart(Competitive~., data = train.eBayAuctions, method = 'class', minbucket =50, maxdepth = 7)
> prp(fit, type = 2, extra = 101, main = "Customized Decision Tree Display")
```

See appendix 1 for default tree (1 page). You can zoom into it for better clarity.

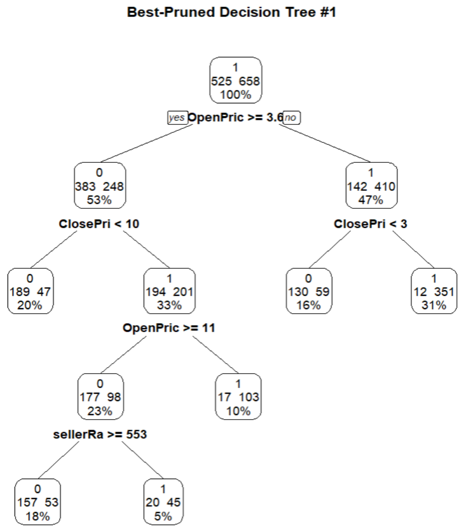
Then, we also produced a classification tree using all predictors which is shown in appendix 2, where all terminal nodes are pure and belong to only one class. This tree did not apply the conditions given in the assignment (minbucket and maxdepth) as it is a deeper one. It produced an error prompting software limitations as shown below:

```
> prp(fit.deeper, type = 2, extra = 1, under = TRUE, split.font = 1, varlen = -10)
Warning message:
labs do not fit even at cex 0.15, there may be some overplotting
```

For this reason, we used the default tree with given conditions in the assignment and produced the best-pruned tree. We noticed that only three predictors were used by the model - openPrice, closePrice and sellerRating which are numeric attributes. They are more suitable to choose as variables.

At the same time, to create the best-prune tree, we created multiple decision trees using different complexity parameters, however the decision tree remained the same. This denoted that either our tree was already optimally pruned or that our dataset was too small (problem was straightforward) due to which the changing cp value did not affect our decision tree.

```
#Testing for best-pruned tree with different complexity parameters
tree_model  <- rpart(Competitive~., data = train.eBayAuctions, method = 'class',  minbucket =50, maxdepth = 7

cv_results <- prune(tree_model, cp = seq(0.01))
cv_results2 <- prune(tree_model, cp = seq(0.0001))
cv_results3 <- prune(tree_model, cp = seq(0.5 , 0.01, by = 0.01))

best_pruned_tree <- cv_results$cptable[which.min(cv_results$cptable[, "xerror"]), "nsplit"]
best_pruned_model <- prune(tree_model, best_pruned_tree)

best_pruned_tree2 <- cv_results2$cptable[which.min(cv_results$cptable[, "xerror"]), "nsplit"]
best_pruned_model2 <- prune(tree_model, best_pruned_tree2)

best_pruned_tree3 <- cv_results2$cptable[which.min(cv_results$cptable[, "xerror"]), "nsplit"]
best_pruned_model3 <- prune(tree_model, best_pruned_tree3)

prp(best_pruned_model, type = 2, extra = 101, main = "Best-Pruned Decision Tree #1")
prp(best_pruned_model2, type = 2, extra = 101, main = "Best-Pruned Decision Tree #2")
prp(best_pruned_model3, type = 2, extra = 101, main = "Best-Pruned Decision Tree #3")
```



Results in terms of the rules (based on best-pruned decision tree above):

- IF openPrice < 3.6, AND closePrice <3, THEN Class = 0 (16 Non Competitive)
- IF openPrice < 3.6, AND closePrice >=3, THEN Class = 1 (31% Competitive)
- IF openPrice >= 3.6, AND closePrice <10, THEN Class = 0 (20% Non Competitive)
- IF openPrice >= 3.6, AND closePrice >=10, AND openPrice < 11, THEN Class = 1 (10% Competitive)
- IF openPrice >= 3.6, AND closePrice >=10, AND openPrice >= 11, AND sellerRating >= 553, THEN Class = 0 (18% Non Competitive)
- IF openPrice >= 3.6, AND closePrice >=10, AND openPrice >= 11, AND sellerRating < 553, THEN Class = 0 (5% Competitive)

Question (1b)

In order to determine the practicality of the model built in section 1a, we used a confusion matrix as shown below:

```
> #To test our decision tree we use a confusion matrix
> default.ct.point.pred.train <- predict(fit,train.eBayAuctions, type= "class")
> deeper.ct.point.pred.train <- predict(fit.deeper,train.eBayAuctions, type= "class")
> train.eBayAuctions$Competitive <- factor(train.eBayAuctions$Competitive, levels = c(0,1), labels = c("0", "1"))
> levels(default.ct.point.pred.train)
[1] "0" "1"
> levels(train.eBayAuctions$Competitive)
[1] "0" "1"
> library(caret)
```

```
> confusionMatrix(default.ct.point.pred.train, train.eBayAuctions$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 494 119
         1  42 528

               Accuracy : 0.8639
                 95% CI : (0.843, 0.8829)
    No Information Rate : 0.5469
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7287

 Mcnemar's Test P-Value : 2.103e-09

            Sensitivity : 0.9216
            Specificity : 0.8161
         Pos Pred Value : 0.8059
         Neg Pred Value : 0.9263
             Prevalence : 0.4531
         Detection Rate : 0.4176
   Detection Prevalence : 0.5182
      Balanced Accuracy : 0.8689

       'Positive' Class : 0
```

The results of the confusion matrix for the default tree shows that our accuracy is 86%. The false positives were119 and false negatives were 42.

We also mapped the confusion matrix for the deeper tree:

```
> fit.deeper <- rpart(Competitive~., data = train.eBayAuctions, method = 'class', cp = 0, minsplit =1)
> prp(fit.deeper, type = 2, extra = 1, under = TRUE, split.font = 1, varlen = -10)
```

```
> confusionMatrix(deeper.ct.point.pred.train, train.eBayAuctions$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 535   8
         1   1 639

               Accuracy : 0.9924
                 95% CI : (0.9856, 0.9965)
    No Information Rate : 0.5469
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9847

 Mcnemar's Test P-Value : 0.0455

            Sensitivity : 0.9981
            Specificity : 0.9876
         Pos Pred Value : 0.9853
         Neg Pred Value : 0.9984
             Prevalence : 0.4531
         Detection Rate : 0.4522
   Detection Prevalence : 0.4590
      Balanced Accuracy : 0.9929

       'Positive' Class : 0
```

We used the training dataset to assess the accuracy of our best-pruned decision tree. We used the following code:

```
> confusionMatrix(test.train, train.eBayAuctions$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 476 159
         1  49 499

               Accuracy : 0.8242
                 95% CI : (0.8013, 0.8455)
    No Information Rate : 0.5562
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6512

 Mcnemar's Test P-Value : 4.1e-14

            Sensitivity : 0.9067
            Specificity : 0.7584
         Pos Pred Value : 0.7496
         Neg Pred Value : 0.9106
             Prevalence : 0.4438
         Detection Rate : 0.4024
   Detection Prevalence : 0.5368
      Balanced Accuracy : 0.8325

       'Positive' Class : 0
```

We used the validation dataset to further assess the accuracy of our best-pruned decision tree. We used the following code:

```
>   valid.eBayAuctions$Competitive <- factor(valid.eBayAuctions$Competitive, levels = c(0,1), labels = c("0", "1"))
>   #To test our decision tree using the validation dataset
>   default.ct.point.pred.valid <- predict(fit,valid.eBayAuctions, type= "class")
>   deeper.ct.point.pred.valid <- predict(fit.deeper,valid.eBayAuctions, type= "class")
>   confusionMatrix(default.ct.point.pred.valid, valid.eBayAuctions$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 331  99
         1  49 309

               Accuracy : 0.8122
                 95% CI : (0.7831, 0.8389)
    No Information Rate : 0.5178
    P-Value [Acc > NIR] : < 2.2e-16
```

The confusion matrix for the validation dataset showed our accuracy to be 81.2%. 99 False negatives were identified, while the number of false positives were 49. This confusion matrix also noted a slight decrease in our accuracy as compared to the confusion matrix of the training dataset.
Based on the accuracy of the validation data set, this model is practical because the accuracy of the validation dataset is closer to the accuracy of the training dataset.

Question (1c)

Describe the interesting and uninteresting information that these rules (in section 1a) provide:

- Based on the rules (in section 1a), only 46% of the auctions were competitive which is less than half of the total auctions.
- Almost all the instances, open price and close price were factors determining the competitiveness of the auctions.
- Only three variables - close price, open price and seller rating had an impact on the competitiveness of auctions, and not the other variables.
- The most competitive auctions (31%) occurred when the openPrice started at the lowest price (i.e., less than 3.6) in that respective currency, and the closePrice was more than or equal to 3 which is a purely price based auction.
- Higher seller ratings did not have any impact on the competitiveness of the auctions. Comparatively, only seller ratings lower than 553 had an impact, however, it was a very little impact as it represented only 5% of the competitive auctions.

- The second-most competitive auctions (10%) were also dependent on price only but the open price range was >= 3.6 and <11, whereas closePrice was >=10.

Question (1d)

Based on the above interpretations, only the open price, close price and the seller ratings were considered to determine the tree as it appeared in the previous best-pruned tree. Other columns were dropped.

Reducing the data to only the chosen three columns.

```
> library(readr)
> eBayAuctions <- read.csv("eBayAuctions.csv")
> View(eBayAuctions)
> eBayMin <- eBayAuctions[, -c(1,2,4,5)]
> View(eBayMin)
> head(eBayMin)
  sellerRating ClosePrice OpenPrice Competitive
1         3249       0.01      0.01           0
2         3249       0.01      0.01           0
3         3249       0.01      0.01           0
4         3249       0.01      0.01           0
5         3249       0.01      0.01           0
6         3249       0.01      0.01           0
```

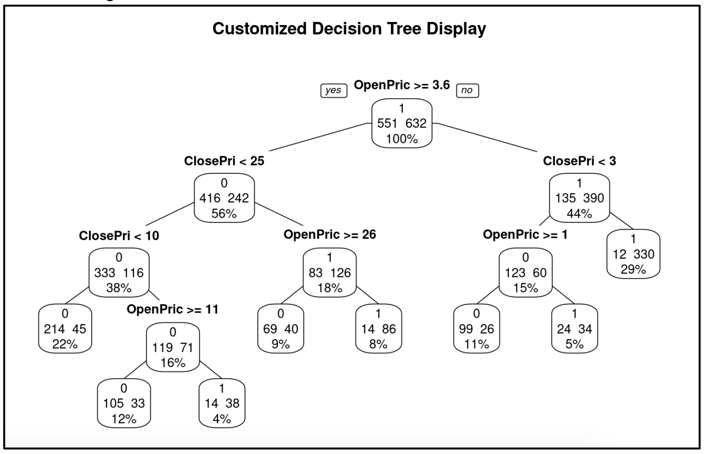
Shuffling the dataset:

```
> #Set Seed
> #To ensure getting same result each time with same seed
> set.seed(60)
> shuffle_index <- sample(1:nrow(eBayMin))
> head(shuffle_index)
[1] 947 592 258 549  90 116
> eBayMin <- eBayMin[shuffle_index, ]
> head(eBayMin)
    sellerRating ClosePrice OpenPrice Competitive
947        10067       6.00      6.00           0
592         2349       3.60      3.60           0
258          533     112.50      0.01           1
549          743       9.72      1.00           1
90          3249       4.90      0.01           1
116         3249       7.50      0.01           1
> tail(eBayMin)
     sellerRating ClosePrice OpenPrice Competitive
973           109      49.80      1.23           1
1742           88      71.00      9.99           1
524          4390       2.99      2.99           1
30           3249       0.21      0.01           1
147          1029       1.07      1.07           0
1494          362     199.00      1.00           1
```

Splitting the dataset into the training data 60% and validation data 40%.
Then we generated the best pruned tree using minbucket = 50 and levels = 7, with the new training dataset and plotted the tree.

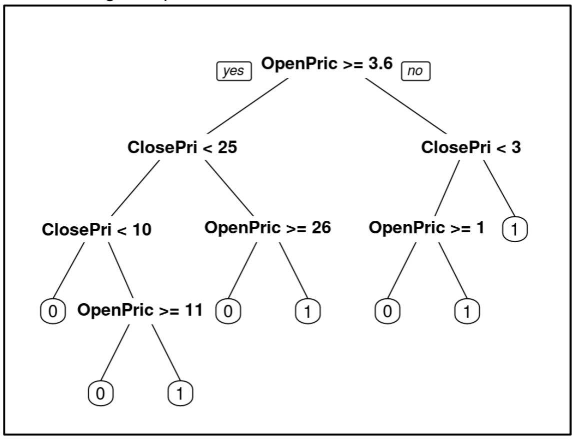
```
> # Creating random samples for data preparation
> #Sampling training data with 60% of the total
> train.rows <- sample(rownames(eBayMin), dim(eBayMin)[1]*0.6)
> #Sampling validation data with 40% of the total
> valid.rows <- sample(setdiff(rownames(eBayMin), train.rows),dim(eBayMin)[1]*0.4)
> #Populating samples
> train.eBayMin <- eBayMin[train.rows, ]
> valid.eBayMin <- eBayMin[valid.rows, ]
> #Best-pruned decision tree according to the assignment document.
> fit <- rpart(Competitive~., data = train.eBayMin, method = 'class', minbucket =50, maxdepth = 7)
> prp(fit, type = 2, extra = 101, main = "Customized Decision Tree Display")
> prp(fit, type = 1, extra = 101, main = "Customized Decision Tree Display")
```

The resulting default decision tree:




```
> cv_eBayMin1 <- prune(fit, cp = eBayMin$cptable[which.min(eBayMin$cptable[, "xerror"]), "CP"])
> prp(cv_eBayMin1)
```

The resulting best pruned decision tree:



Important Note: Although we used three variables to draw the best-pruned tree - open price, close price and the seller rating, the resulting best-pruned tree was only produced using two of the variables which are open price and close price.

The rules of the above best-pruned tree are as follows:

- IF openPrice < 3.6, AND closePrice >=3, THEN Class =1 (Competitive)
- IF openPrice < 3.6, AND closePrice <3, AND openPrice >=1, THEN Class =0 (Non Competitive)
- IF openPrice < 3.6, AND closePrice <3, AND openPrice <1, THEN Class =1 (Competitive)
- IF openPrice >= 3.6, AND closePrice <25, AND closePrice <10, THEN Class = 0 (Non Competitive)
- IF openPrice >= 3.6, AND closePrice <25, AND closePrice >=10, AND openPrice >=11, THEN Class = 0 (Non Competitive)
- IF openPrice >= 3.6, AND closePrice <25, AND closePrice >=10, AND openPrice <11, THEN Class = 1 (Competitive)

- IF openPrice >= 3.6, AND closePrice >=25, AND openPrice >=26, THEN Class = 0 (Non Competitive)
- IF openPrice >= 3.6, AND closePrice >=25, AND openPrice <26, THEN Class = 1 (Competitive)

Smallest set of rules for classification (based on above rules):

- IF openPrice < 3.6, AND closePrice >=3, THEN Class =1 (Competitive)
- IF openPrice < 3.6, AND closePrice <3, THEN Class =0 (Non Competitive)
- IF closePrice <3, AND openPrice <1,THEN Class =1 (Competitive)
- IF openPrice >= 3.6, AND closePrice <10, THEN Class = 0 (Non Competitive)
- IF closePrice <25, AND closePrice >=10, AND openPrice >=11, THEN Class = 0 (Non Competitive)
- IF closePrice <25, AND closePrice >=10, AND openPrice <11, THEN Class = 1 (Competitive)
- IF closePrice >=25, AND openPrice >=26, THEN Class = 0 (Non Competitive)
- IF AND closePrice >=25, AND openPrice <26, THEN Class = 1 (Competitive)

Description based on smallest rules for classification:
There are four rules which classify the auctions as competitive (and four for non-competitive) as written above, and all of them are classified based only on two predictors openPrice and closePrice. In competitive auction rules, often the difference between the open price and the close price is relatively significant than those in non-competitive auctions. The rules do not show the use of sellerRatings for classification in comparison to the previous best-pruned model, although we used it as a predictor here.

Question (1e)
Following is the scatterplot for the pruned decision tree.

```
# Scatter plot of the data points
plot(
  train.eBayAuctions$ClosePrice,
  train.eBayAuctions$OpenPrice,
  col = ifelse(train.eBayAuctions$Competitive == 1, "red", "blue"),
  pch = 19,
  main = "Decision Tree on Scatter Plot",
  xlab = "Close Price",  # Adding x-axis label
  ylab = "Open Price"    # Adding y-axis label
)
```

The red points denote the competitive auctions and the blue points represent the non-competitive auctions.

When open price and close price are increasing with little to no margin, it is a non-competitive auction (blue). When the open price is significantly lower and the close price is increasing relative to the open price (with a higher margin) it becomes a competitive auction (red).

The splitting does seem reasonable since it reflects the nature of both the predictors. However, the two classes are not separated very cleanly. Some of the non-competitive data points are misclassified and thus the classification seems slightly unclean.

Question (1f)

Confusion matrix

Confusion matrix for training data was 82.42% accurate. There are 144 false positives, and 64 false negatives.

```
> confusionMatrix(pruned.train, train.eBayMin$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 487 144
         1  64 488

               Accuracy : 0.8242
                 95% CI : (0.8013, 0.8455)
    No Information Rate : 0.5342
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.65

 Mcnemar's Test P-Value : 4.31e-08

            Sensitivity : 0.8838
            Specificity : 0.7722
         Pos Pred Value : 0.7718
         Neg Pred Value : 0.8841
             Prevalence : 0.4658
         Detection Rate : 0.4117
   Detection Prevalence : 0.5334
      Balanced Accuracy : 0.8280

       'Positive' Class : 0
```

Confusion matrix for validation data was 82.87% accurate.

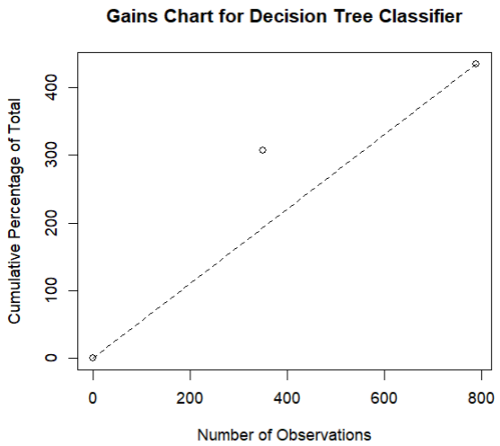
```
> confusionMatrix(pruned.valid, valid.eBayMin$Competitive)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 303  84
         1  51 350

               Accuracy : 0.8287
                 95% CI : (0.8005, 0.8544)
    No Information Rate : 0.5508
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6567

 Mcnemar's Test P-Value : 0.005885

            Sensitivity : 0.8559
            Specificity : 0.8065
         Pos Pred Value : 0.7829
         Neg Pred Value : 0.8728
             Prevalence : 0.4492
         Detection Rate : 0.3845
   Detection Prevalence : 0.4911
      Balanced Accuracy : 0.8312

       'Positive' Class : 0
```

The accuracy of the training dataset is 82.4% and the validation dataset is 82.8%. This is almost the same. Based on the above two confusion matrices' accuracy percentage, we identify that the predictive performance of this model is strong and excellent.

Lift Chart

For plotting the lift chart, the following code was used. We used the gains library to plot the number of observations predicted against the cumulative percentage of total.

```
valid.eBayMin$Competitive <- as.numeric(as.character(valid.eBayMin$Competitive))

library(gains)
gain <- gains(train.eBayAuctions$Competitive,test.train, groups = 200 )

plot(c(0,gain$cume.pct.of.total*sum(valid.eBayMin$Competitive))~
       c(0,gain$cume.obs),
     xlab="Number of Observations", ylab = "Cumulative Percentage of Total", main = "Lift Chart for Decision Tree Classifier" )
lines(c(0,sum(valid.eBayMin$Competitive))~c(0,dim(valid.eBayMin)[1]),lty=2)
```



When our Number of Observations are approximately 600, the decision tree model performs better than the baseline. However, for a higher (1200) or lower(0) number of observations, our model does not perform well at all, and predicts same as a random model.

We identified 3 points in our lift chart.

The point (600,340) indicates that, when using the model, after considering the first 600 observations (cumulative), our model has identified 320 (cumulative) positive outcomes. This suggests that the model is performing better than random chance, as it's above the baseline, since the higher the point, the better the model is at identifying positive outcomes.

The origin (0, 0) on the chart and (1200,500) point, represent the cumulative percentage of the total dataset versus the cumulative percentage of the positive outcomes (e.g., successes, events). Since they're both on the baseline, it indicates the performance of a model that doesn't provide any advantage over random chance.

In summary, the model shows reasonable overall accuracy, but the insights from the lift/gain chart suggest potential areas for improvement, particularly in later portions of the dataset. Further analysis and potentially refining the model or exploring additional evaluation metrics could be beneficial in improving the overall accuracy and performance of predicting competitive auctions.

List of Appendices for section 1a

Appendix 1: Default Decision Tree

Appendix 2: Decision Tree using all predictors - with pure terminal nodes/ leaves