Data Mining **Fall 2020**

Homework 3 By (BAN620: Group-8)

Question 1. Summary on Universal Bank dataset:

- The new record:
  (Age= 40, Experience = 10, Income = 84, Family= 2, CCAvg=2, Education=2, Mortgage=0, Securities.Account=0, CD.Account=0, Online=1, CreditCard=1)

- We have normalized, preprocessed and partitioned (into 2 parts, training and validation) the original data set and new record. Then we have explored the data summary after choosing k= 3, and later in our experiment, we examine the accuracy (of prediction in the validation set) of all the results that came from different values of k between 1 and 14.

Original dataset excluding zip code and ID column

```
> str(bank.copy.dataset)
'data.frame':	5000 obs. of  12 variables:
 $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
 $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
 $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
 $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
 $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
 $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
>
```

After converting it to factors

```
> str(knn.norm.dataset)
'data.frame':   5000 obs. of  12 variables:
 $ Age              : num  -1.7967 -0.0377 -0.5654 -0.9172 -0.9172 ...
 $ Experience       : num  -1.686 -0.105 -0.456 -0.983 -1.071 ...
 $ Income           : num  -0.527 -0.855 -1.358 0.589 -0.614 ...
 $ Family           : Factor w/ 4 levels "1","2","3","4": 4 3 1 1 4 4 2 1 3 1 ...
 $ CCAvg            : num  -0.182 -0.24 -0.528 0.454 -0.528 ...
 $ Education        : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : num  -0.568 -0.568 -0.568 -0.568 -0.568 ...
 $ Personal.Loan    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
 $ Securities.Account: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
 $ CD.Account       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Online           : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
 $ CreditCard       : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
>
```

Prediction category of new data:
The new customer is predicted to not accept the loan offer

```
> knn.fit<-create.knnmodel(knn.train.data,new.norm.df,"Personal.Loan",3)
> #predicted class of the new data
> knn.fit
[1] 0
Levels: 0 1
```

Question 1.a. Scope of the study – Identify the best K:

- We choose k which has the lowest error rate or the highest accuracy in validation data.
- If k decreases, more local structure can be captured but there is also a risk of noise. But higher values of k might provide more smoothing of the structure, but it might not capture a lot of local structure. So, we must be very careful while picking k and find a better trade-off.
- According to the accuracy table given below, best k=3, as the corresponding accuracy (96.25%) is highest for the same

```
> banking.accuracy.df
    k accuracy
1   1   0.9590
2   2   0.9565
3   3   0.9625
4   4   0.9575
5   5   0.9595
6   6   0.9600
7   7   0.9600
8   8   0.9585
9   9   0.9575
10 10   0.9545
11 11   0.9540
12 12   0.9540
13 13   0.9535
14 14   0.9535
> maxk.value=banking.accuracy.df[which.max(banking.accuracy.df[,"accuracy"]),"k"]
> maxk.value
[1] 3
>
```

Question 1.b. The role of Confusion Matrix, Sensitivity and Specificity in Classification:

- Sensitivity for the above best k is 99.89%, which implies we are predicting true positives 99.89% of the times.
- Specificity is 64.39%, which implies we are predicting true negatives of the case 64.39% of the times.
- On K=3, with 2 partition: accuracy level on validation data is 96.25%

```
> knn.fit2 <- create.knnmodel(knn.train.data,knn.valid.data,"Personal.Loan",maxk.value)
> confusionMatrix(knn.fit2,knn.valid.data$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1793   73
         1    2  132

               Accuracy : 0.9625
                 95% CI : (0.9532, 0.9704)
    No Information Rate : 0.8975
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7593

 Mcnemar's Test P-Value : 6.324e-16

            Sensitivity : 0.9989
            Specificity : 0.6439
         Pos Pred Value : 0.9609
         Neg Pred Value : 0.9851
             Prevalence : 0.8975
         Detection Rate : 0.8965
   Detection Prevalence : 0.9330
      Balanced Accuracy : 0.8214

       'Positive' Class : 0
```

- Among 2000 customers, our model correctly predicted that 132 customers will accept the loan. Through this confusion matrix, we can also determine what type of misclassification is more frequent.

```
> print(temp.table)
   knn.fit2
       0    1
  0 1793    2
  1   73  132
> #temp.table[1, "0"]-->1st row having column-name as "0"
> sensitivity.value<-temp.table[1, "0"]/(temp.table[1, "0"]+temp.table[1, "1"])
> specificity.value<-temp.table[2, "1"]/(temp.table[2, "1"]+temp.table[2, "0"])
> cat("specificity is :",specificity.value)
specificity is : 0.6439024> cat("sensitivity  is :",sensitivity.value)
sensitivity  is : 0.9988858
>
```

Question 1.c. Observation after partitioning dataset into 3 parts:

- Accuracy of validation data for 2 partitions and 3 partitions are comparable.
- Accuracy of test data (in case of 3 partitions) is lower than that of validation data.

Accuracy on validation data when k=3 and Partitioning into 3 sets

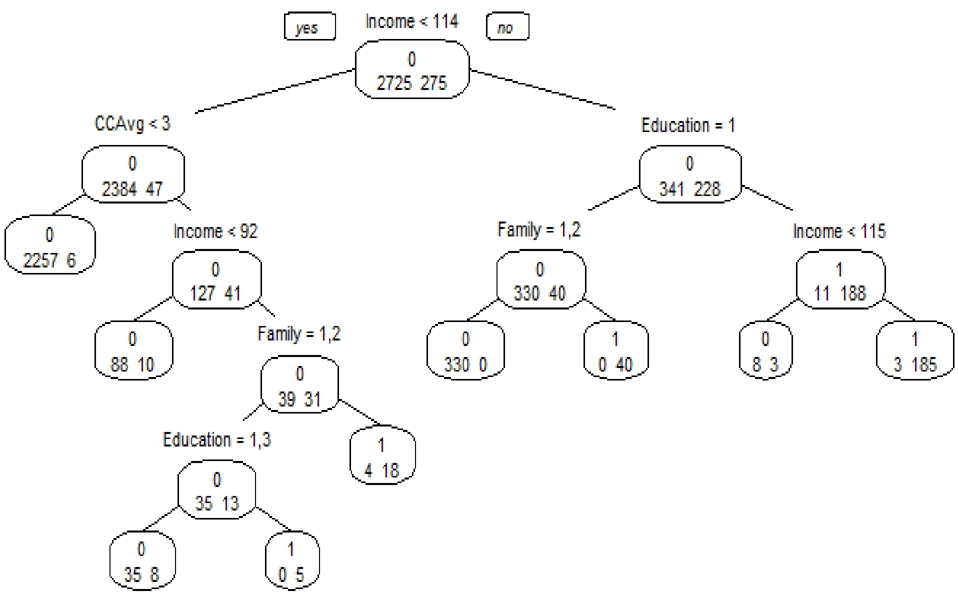
```
> knn.fit2 <- create.knnmodel(knn.train.data,knn.valid.data,"Personal.Loan",maxk.value)
> confusionMatrix(knn.fit2,knn.valid.data$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1363   46
         1    1   90

               Accuracy : 0.9687
                 95% CI : (0.9585, 0.9769)
    No Information Rate : 0.9093
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7767

 Mcnemar's Test P-Value : 1.38e-10

            Sensitivity : 0.9993
            Specificity : 0.6618
         Pos Pred Value : 0.9674
         Neg Pred Value : 0.9890
             Prevalence : 0.9093
         Detection Rate : 0.9087
   Detection Prevalence : 0.9393
      Balanced Accuracy : 0.8305

       'Positive' Class : 0
```

Accuracy on test data when k=3 and Partitioning into 3 sets

```
> knn.fit3 <- create.knnmodel(knn.train.data,knn.test.data,"Personal.Loan",maxk.value)
> confusionMatrix(knn.fit3,knn.test.data$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 886  42
         1   2  70

               Accuracy : 0.956
                 95% CI : (0.9414, 0.9679)
    No Information Rate : 0.888
    P-Value [Acc > NIR] : 1.587e-14

                  Kappa : 0.7379

 Mcnemar's Test P-Value : 4.116e-09

            Sensitivity : 0.9977
            Specificity : 0.6250
         Pos Pred Value : 0.9547
         Neg Pred Value : 0.9722
             Prevalence : 0.8880
         Detection Rate : 0.8860
   Detection Prevalence : 0.9280
      Balanced Accuracy : 0.8114

       'Positive' Class : 0
```

Question 2.a. Summary on Universal Bank dataset with classification tree:

- One of the rules from the following tree:
- if (Income <114, CCAvg < 3, then Personal.Loan =0. This depicts that the majority of clients will not accept the personal loan offer.



Question 2.b. Performance checking classification tree and kNN on the same data set:

- In kNN model (k=3) the accuracy is 96.25 % on the validation data, and sensitivity is 99.89%, and specificity is 64.39%.
- In classification tree accuracy level is 98.15%, which is greater than that of KNN model. Here observed sensitivity value is 99.55%, and specificity value is 85.85%.
- Considering above metrics Classification Tree model is recommended over KNN to get more accurate prediction for the new customer. Some of the other advantages are:
- A decision Tree model is very intuitive as all the decision paths are clearly formulated.
- It does not require lot of data-preprocessing as compared to KNN and handles missing data better than KNN models.
- It also determines variable importance and place the most important one as the top node of tree. Combining this information with the domain knowledge can be helpful in creating more accurate models.

```
> bank.ctmodel$variable.importance
 Education     Income     Family      CCAvg CD.Account   Mortgage        Age Experience
189.133578 160.495243 128.634847  75.500615  20.974594  19.374508   3.131641   1.494987
>
```

Performance of above tree on validation data

```
[1] "....................................Performance on the validation data..................................."
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1787   29
         1    8  176

               Accuracy : 0.9815
                 95% CI : (0.9746, 0.9869)
    No Information Rate : 0.8975
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8947

 Mcnemar's Test P-Value : 0.001009

            Sensitivity : 0.9955
            Specificity : 0.8585
         Pos Pred Value : 0.9840
         Neg Pred Value : 0.9565
             Prevalence : 0.8975
         Detection Rate : 0.8935
   Detection Prevalence : 0.9080
      Balanced Accuracy : 0.9270
```

Performance of kNN on validation data

```
> knn.fit2 <- create.knnmodel(knn.train.data,knn.valid.data,"Personal.Loan",maxk.value)
> confusionMatrix(knn.fit2,knn.valid.data$Personal.Loan)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1793   73
         1    2  132

               Accuracy : 0.9625
                 95% CI : (0.9532, 0.9704)
    No Information Rate : 0.8975
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7593

 Mcnemar's Test P-Value : 6.324e-16

            Sensitivity : 0.9989
            Specificity : 0.6439
         Pos Pred Value : 0.9609
         Neg Pred Value : 0.9851
             Prevalence : 0.8975
         Detection Rate : 0.8965
   Detection Prevalence : 0.9330
      Balanced Accuracy : 0.8214

       'Positive' Class : 0

```

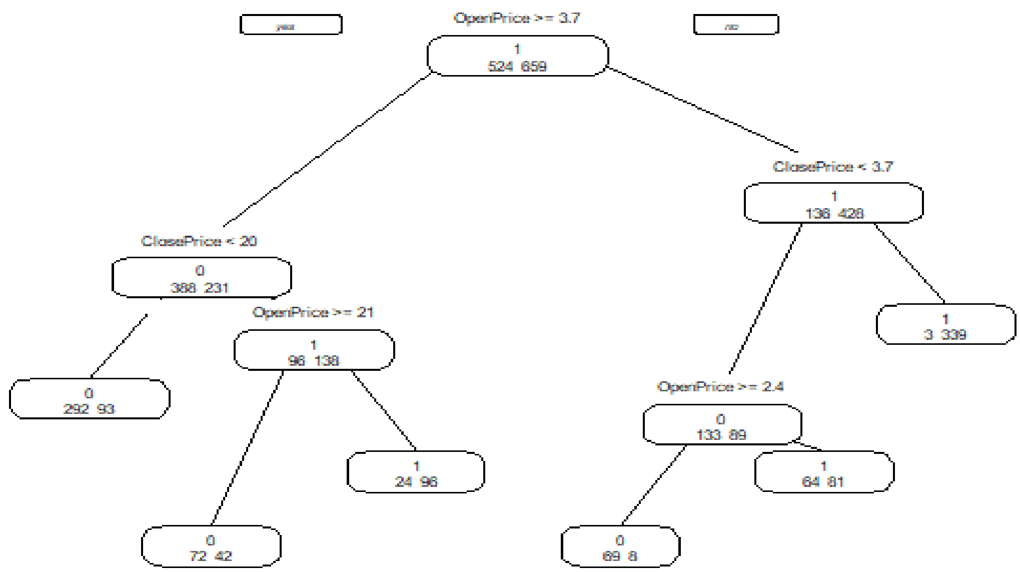
Question 3.a. Summary on eBayAuctions dataset:

- The eBay Auction data set contains information of 1972 auctions during May-June 2004. The following model classifies auctions as competitive or non- competitive.
- It is given that the seller selected few variables (duration, opening price, currency, day-of-week) and trying to verify sellers.

Data set details

```
> str(ebay.dataset)
'data.frame':   1972 obs. of  8 variables:
 $ Category    : Factor w/ 18 levels "Antique/Art/Craft",..: 14 14 14 14 14 14 14 14 14 14 ...
 $ currency    : Factor w/ 3 levels "EUR","GBP","US": 3 3 3 3 3 3 3 3 3 3 ...
 $ sellerRating: int  3249 3249 3249 3249 3249 3249 3249 3249 3249 3249 ...
 $ Duration    : Factor w/ 5 levels "1","3","5","7",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ endDay      : Factor w/ 7 levels "Fri","Mon","Sat",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ ClosePrice  : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ...
 $ OpenPrice   : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ...
 $ Competitive : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
> dim(ebay.ct.train.df)
[1] 1183    8
> dim(ebay.ct.valid.df)
[1] 789   8
>
```

Tree with all the predictor



- This above tree can be converted into a set of rules, as: - IF (OpenPrice >= 3.7) AND (ClosePrice < 20) THEN Personal.Loan =0. It conveys that majority of the bid is not competitive.

Performance of above tree on training data

```
> #checking performance of the above model
> check.ct.performance(ebay.ct.train.df,ebay.ct.valid.df,"Competitive",ebay.ctmodel)
[1] ".....................................Performance on the training data................................."
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 433 143
         1  91 516

               Accuracy : 0.8022
                 95% CI : (0.7783, 0.8245)
    No Information Rate : 0.5571
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6032

 Mcnemar's Test P-Value : 0.0008561

            Sensitivity : 0.8263
            Specificity : 0.7830
         Pos Pred Value : 0.7517
         Neg Pred Value : 0.8501
             Prevalence : 0.4429
         Detection Rate : 0.3660
   Detection Prevalence : 0.4869
      Balanced Accuracy : 0.8047

       'Positive' Class : 0
```

Performance on the validation data

```
[1] "..................................Performance on the validation data................................."
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 294 114
         1  88 293

               Accuracy : 0.744
                 95% CI : (0.712, 0.7741)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4885

 Mcnemar's Test P-Value : 0.07858

            Sensitivity : 0.7696
            Specificity : 0.7199
         Pos Pred Value : 0.7206
         Neg Pred Value : 0.7690
             Prevalence : 0.4842
         Detection Rate : 0.3726
   Detection Prevalence : 0.5171
      Balanced Accuracy : 0.7448

       'Positive' Class : 0
```

Question 3.b. Recommendation for prediction of a New Auction:

- The above tree in **part a** is not helpful for prediction of a new auction since the seller will not have any information on closing price before the auction; though it is an important variable for our model.

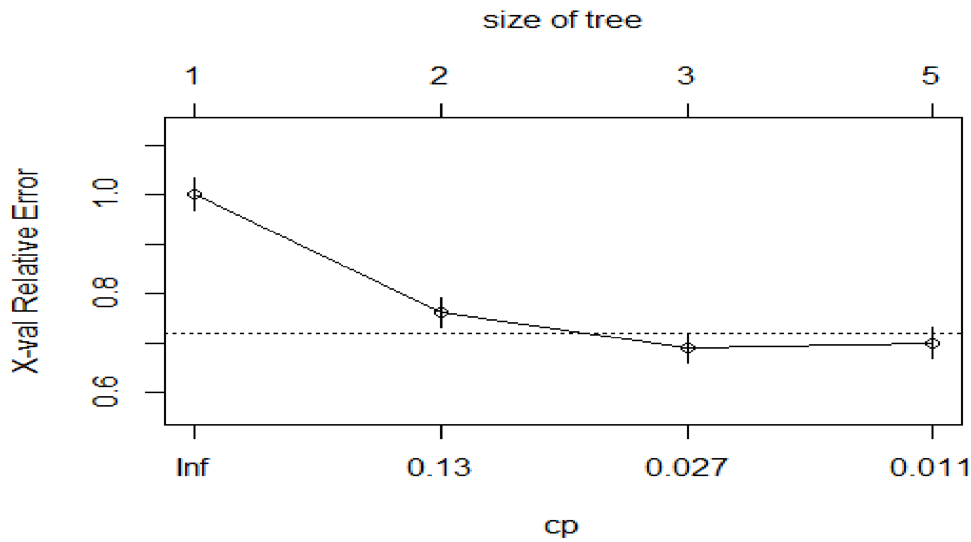
- To change the approach, we should change the variables that are chosen by seller to predict a new auction and also go for pruning to check if the accuracy can be increased.

- Dropped ClosePrice from the predictors list and performed tree pruning

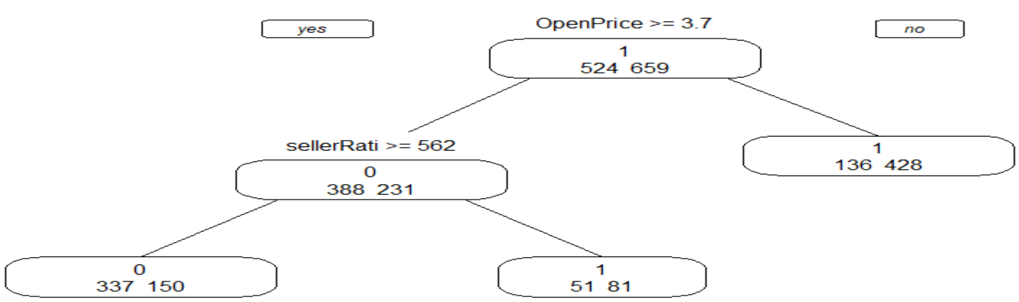
Cross validation:(Cp selected-0.01)

```
> printcp(ebay.ct)

Classification tree:
rpart(formula = Competitive ~ . - ClosePrice, data = ebay.ct.train.df,
    method = "class", control = rpart.control(minbucket = 50,
        maxdepth = 7), cp = 1e-05, minsplit = 10, xval = 5)

Variables actually used in tree construction:
[1] Category     endDay       OpenPrice    sellerRating

Root node error: 524/1183 = 0.44294

n= 1183

        CP nsplit rel error  xerror     xstd
1 0.299618      0   1.00000 1.00000 0.032605
2 0.057252      1   0.70038 0.76145 0.031033
3 0.012405      2   0.64313 0.68893 0.030225
4 0.010000      4   0.61832 0.70038 0.030364
```

- As no of splits increases, CP value decreases.



Tree after pruning



Performance of above tree on training data

```
> check.ct.performance(ebay.ct.train.df,ebay.ct.valid.df,"Competitive",ebaypruned.ct)
[1] "....................................Performance on the training dat
a..................................."
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 337 150
         1 187 509

               Accuracy : 0.7151
                 95% CI : (0.6885, 0.7407)
    No Information Rate : 0.5571
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4185

 Mcnemar's Test P-Value : 0.04987

            Sensitivity : 0.6431
            Specificity : 0.7724
         Pos Pred Value : 0.6920
         Neg Pred Value : 0.7313
             Prevalence : 0.4429
         Detection Rate : 0.2849
   Detection Prevalence : 0.4117
      Balanced Accuracy : 0.7078

       'Positive' Class : 0
```

Performance of above tree on validation data

```
[1] "....................................Performance on the validation dat
a..................................."
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 238  97
         1 144 310

               Accuracy : 0.6946
                 95% CI : (0.6611, 0.7265)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3862

 Mcnemar's Test P-Value : 0.003045

            Sensitivity : 0.6230
            Specificity : 0.7617
         Pos Pred Value : 0.7104
         Neg Pred Value : 0.6828
             Prevalence : 0.4842
         Detection Rate : 0.3016
   Detection Prevalence : 0.4246
      Balanced Accuracy : 0.6924

       'Positive' Class : 0
```

- The accuracy decreases from 74.4% to 69.46% after pruning (all predictors except closing price), So Random Forest and boosting algorithms can be executed to check if the accuracy can be improved.

Random Forest

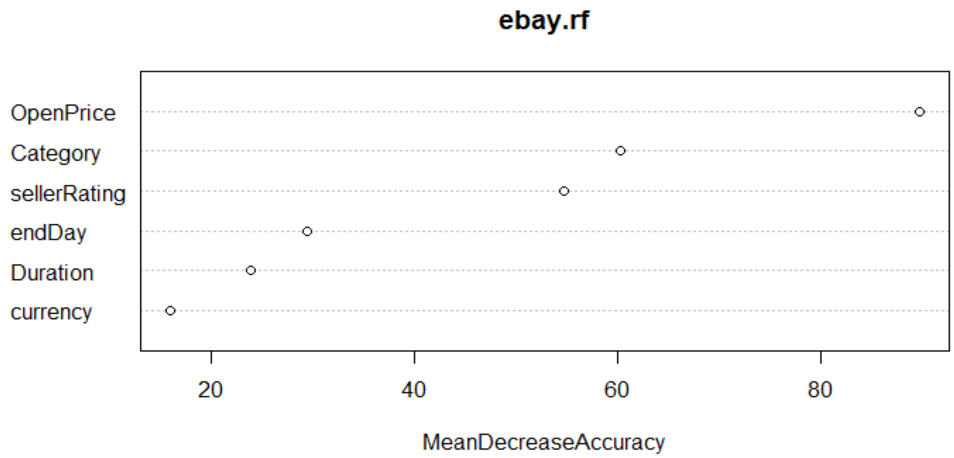
Performance on training data

```
> #checking performance of random forest model:
> check.ct.performance(ebay.ct.train.df,ebay.ct.valid.df,"Competitive",ebay.rf)
[1] "....................................Performance on the training data................................
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 459  39
         1  65 620

               Accuracy : 0.9121
                 95% CI : (0.8945, 0.9276)
    No Information Rate : 0.5571
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8209

 Mcnemar's Test P-Value : 0.01423

            Sensitivity : 0.8760
            Specificity : 0.9408
         Pos Pred Value : 0.9217
         Neg Pred Value : 0.9051
             Prevalence : 0.4429
         Detection Rate : 0.3880
   Detection Prevalence : 0.4210
      Balanced Accuracy : 0.9084

       'Positive' Class : 0
```

(Performance on validation data)

```
[1] "..................................Performance on the validation data.................................."
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 236  85
         1 146 322

               Accuracy : 0.7072
                 95% CI : (0.6741, 0.7388)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.411

 Mcnemar's Test P-Value : 7.89e-05

            Sensitivity : 0.6178
            Specificity : 0.7912
         Pos Pred Value : 0.7352
         Neg Pred Value : 0.6880
             Prevalence : 0.4842
         Detection Rate : 0.2991
   Detection Prevalence : 0.4068
      Balanced Accuracy : 0.7045

       'Positive' Class : 0
```

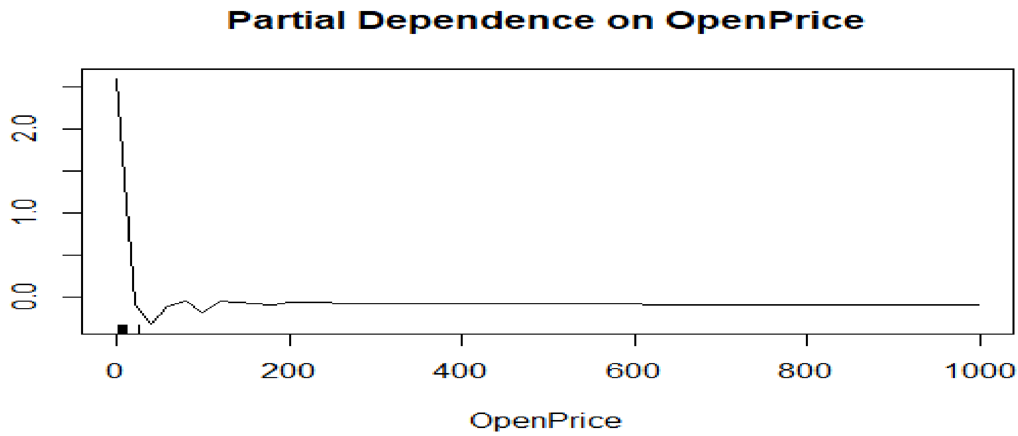
- Accuracy level increased to 70.72 % with random forest method.
- From the below plot, we can draw the conclusion that OpenPrice, Category, SellerRating are the most important predictors.



- Below plots can give the marginal effect of a predictor on a specific classification response.
- We are checking the effect of predictors on the competitive auction. (Response is “1”)

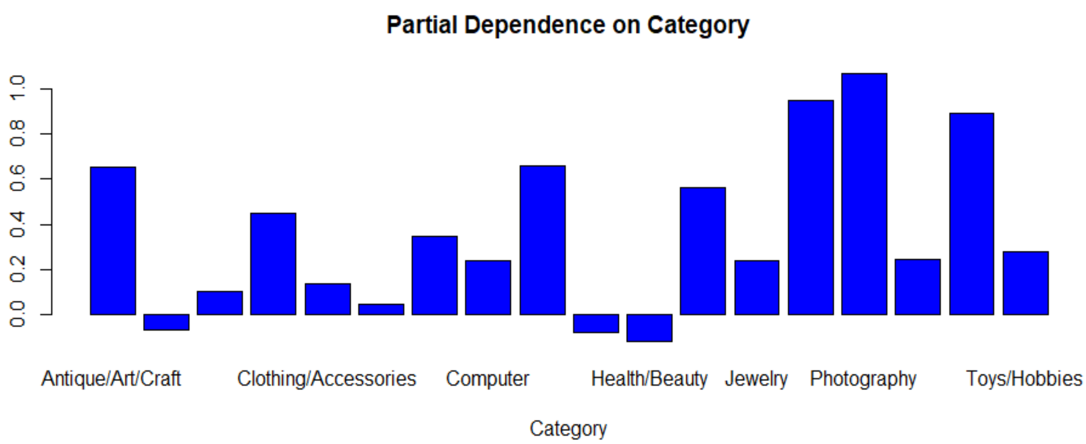
```
varImpPlot(ebay.defaultrf,type=1)

#Relation between outcome and predictors
partialPlot(ebay.defaultrf,ebay.dataset,OpenPrice,"1")
partialPlot(ebay.defaultrf,ebay.dataset,Category,"1")
partialPlot(ebay.defaultrf,ebay.dataset,currency,"1")
```

- Below plot shows that if open price is low, there is higher probability that auction will be of competitive nature. As open Price increases there is less chance that the auction will be classified as competitive.



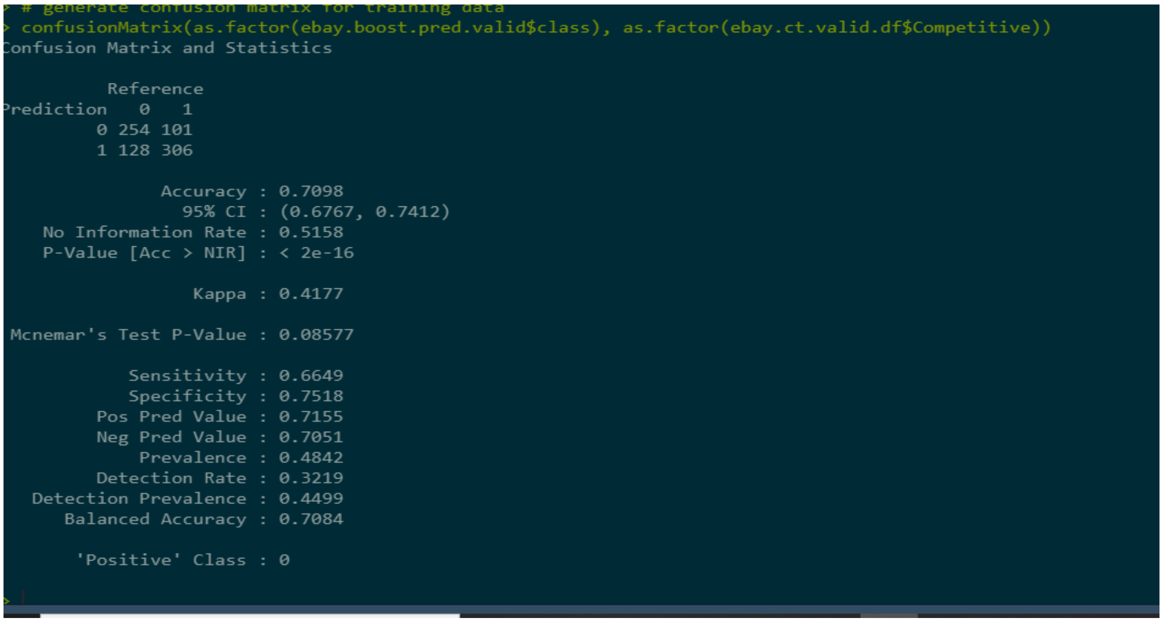
.

- Below plot shows that Photography category has the higher probability to have competitive auction as compared to other categories.



Boosting trees for the same set of predictors.

Performance on valid data



```
> # generate confusion matrix for training data
> confusionMatrix(as.factor(ebay.boost.pred.valid$class), as.factor(ebay.ct.valid.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 254 101
         1 128 306

               Accuracy : 0.7098
                 95% CI : (0.6767, 0.7412)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4177

 Mcnemar's Test P-Value : 0.08577

            Sensitivity : 0.6649
            Specificity : 0.7518
         Pos Pred Value : 0.7155
         Neg Pred Value : 0.7051
             Prevalence : 0.4842
         Detection Rate : 0.3219
   Detection Prevalence : 0.4499
      Balanced Accuracy : 0.7084

       'Positive' Class : 0

>
```

- There is slight improvement in accuracy (70.98%) of the boosted trees as compared to accuracy obtained from random forest (70.72%).
- Boosted trees can be recommended if data for all predictors is available

Summary of Boosted tress with below set of predictors chosen by the seller: Competitive ~ Duration+OpenPrice+endDay+currency

```
> confusionMatrix(as.factor(ebay.selected.boost.pred.valid$class), as.factor(ebay.ct.valid.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 258 133
         1 124 274

               Accuracy : 0.6743
                 95% CI : (0.6403, 0.7069)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.3484

 Mcnemar's Test P-Value : 0.6178

            Sensitivity : 0.6754
            Specificity : 0.6732
         Pos Pred Value : 0.6598
         Neg Pred Value : 0.6884
             Prevalence : 0.4842
         Detection Rate : 0.3270
   Detection Prevalence : 0.4956
      Balanced Accuracy : 0.6743

       'Positive' Class : 0
```

- Accuracy dropped by 3%. Therefore, Seller can add other Predictors like Category, sellerRating and drop currency to increase the accuracy.

Summary of Boosted tress with below set of predictors : Competitive ~ Duration+OpenPrice+endDay+Category+sellerRating

```
> confusionMatrix(as.factor(ebay.selected.boost.pred.valid2$class), as.factor(ebay.ct.valid.df$Competitive))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 261 104
         1 121 303

               Accuracy : 0.7148
                 95% CI : (0.6819, 0.7461)
    No Information Rate : 0.5158
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4283

 Mcnemar's Test P-Value : 0.2861

            Sensitivity : 0.6832
            Specificity : 0.7445
         Pos Pred Value : 0.7151
         Neg Pred Value : 0.7146
             Prevalence : 0.4842
         Detection Rate : 0.3308
   Detection Prevalence : 0.4626
      Balanced Accuracy : 0.7139
```

- In the above matrix table, the accuracy level increased by 1% (compared to Random Forest and boosted trees with other predictors) after including above predictors
- 74% of the times the model can predict correctly that the auction is competitive
- 68% of the times model can predict correctly that the auction is not competitive.
- The model may improve (accuracy metrics) by increasing data size and including more predictors.

Question 3.c. Recommendation for Seller Friend

- Seller can keep the opening prices low to have more competitive auctions.
- Sellers can set up items from Photography category to auction to attract more bidders.
  A seller with more rating would make more people interested in their auctions, hence may lead to increased competitive bidding.