hr_naiveBayes_using_caret_package

June 11, 2018

0.1 In this exercise, we will use the HR dataset and understand the following using caret package:

1. Building the naive bayes model
2. What is marked as the positive class by the model when using caret package
3. Writing the model equation and interpreting the model summary
4. Creating the Confusion Matrix and ROC plot on train data
5. Creating the Confusion Matrix and ROC plot on test data

There are bugs/missing code in the entire exercise. The participants are expected to work upon them.

---

1 Code starts here

We are going to use below mentioned libraries for demonstrating logistic regression:

```
In [1]: #install.packages("klaR", "/Users/Rahul/anaconda3/lib/R/library")
```

```
In [2]: library(caret)     #for data partition. Model building
        library(ROCR)      #for ROC plot (other way)
```

```
Loading required package: lattice
Loading required package: ggplot2
Loading required package: gplots

Attaching package: gplots

The following object is masked from package:stats:

    lowess
```

1.1 Data Import and Manipulation

1.1.1 1. Importing a data set

Give the correct path to the data

```
In [3]: raw_df <- read.csv("/Users/Rahul/Documents/Datasets/IMB533_HR_Data_No_Missing_Value.csv

        raw_df[1779,
              ]
```

	SLNO	Candidate.Ref	DOJ.Extended	Duration.to.accept.offer	Notice.period	Offered.band
1779	2653	2328204	Yes	12	30	E1

Note that `echo = FALSE` parameter prevents printing the R code that generated the plot.

1.1.2 2. Structure and Summary of the dataset

```
In [4]: str(raw_df)
        summary(raw_df)
```

```
'data.frame':        8995 obs. of  18 variables:
 $ SLNO                        : int  1 2 3 4 5 6 7 9 11 12 ...
 $ Candidate.Ref               : int  2110407 2112635 2112838 2115021 2115125 2117167 2119124 2
 $ DOJ.Extended                : Factor w/ 2 levels "No","Yes": 2 1 1 1 2 2 2 2 1 1 ...
 $ Duration.to.accept.offer    : int  14 18 3 26 1 17 37 16 1 6 ...
 $ Notice.period               : int  30 30 45 30 120 30 30 0 30 30 ...
 $ Offered.band                : Factor w/ 4 levels "E0","E1","E2",..: 3 3 3 3 3 2 3 2 2 2 ...
 $ Pecent.hike.expected.in.CTC: num  -20.8 50 42.8 42.8 42.6 ...
 $ Percent.hike.offered.in.CTC: num  13.2 320 42.8 42.8 42.6 ...
 $ Percent.difference.CTC      : num  42.9 180 0 0 0 ...
 $ Joining.Bonus               : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Candidate.relocate.actual   : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
 $ Gender                      : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 2 1 1 2 ...
 $ Candidate.Source            : Factor w/ 3 levels "Agency","Direct",..: 1 3 1 3 3 3 3 2 3 3 .
 $ Rex.in.Yrs                  : int  7 8 4 4 6 2 7 8 3 3 ...
 $ LOB                         : Factor w/ 9 levels "AXON","BFSI",..: 5 8 8 8 8 8 8 7 2 3 ...
 $ Location                    : Factor w/ 11 levels "Ahmedabad","Bangalore",..: 9 3 9 9 9 9 9
 $ Age                         : int  34 34 27 34 34 34 32 34 26 34 ...
 $ Status                      : Factor w/ 2 levels "Joined","Not Joined": 1 1 1 1 1 1 1 1 1 1


      SLNO        Candidate.Ref      DOJ.Extended Duration.to.accept.offer
 Min.   :    1   Min.   :2109586   No :4788     Min.   :  0.00
 1st Qu.: 3208   1st Qu.:2386476   Yes:4207     1st Qu.:  3.00
 Median : 5976   Median :2807482                Median : 10.00
 Mean   : 5971   Mean   :2843647                Mean   : 21.43
 3rd Qu.: 8739   3rd Qu.:3300060                3rd Qu.: 33.00
 Max.   :12333   Max.   :3836076                Max.   :224.00

 Notice.period    Offered.band Pecent.hike.expected.in.CTC
```

```
 Min.   :  0.00   E0: 211      Min.   :-68.83
 1st Qu.: 30.00   E1:5568      1st Qu.: 27.27
 Median : 30.00   E2:2711      Median : 40.00
 Mean   : 39.29   E3: 505      Mean   : 43.86
 3rd Qu.: 60.00                3rd Qu.: 53.85
 Max.   :120.00                Max.   :359.77

 Percent.hike.offered.in.CTC Percent.difference.CTC Joining.Bonus
 Min.   :-60.53              Min.   :-67.270        No :8578
 1st Qu.: 22.09              1st Qu.: -8.330        Yes: 417
 Median : 36.00              Median :  0.000
 Mean   : 40.66              Mean   : -1.574
 3rd Qu.: 50.00              3rd Qu.:  0.000
 Max.   :471.43              Max.   :300.000

 Candidate.relocate.actual    Gender             Candidate.Source
 No :7705                  Female:1551   Agency          :2585
 Yes:1290                  Male  :7444   Direct          :4801
                                         Employee Referral:1609



   Rex.in.Yrs          LOB            Location          Age
 Min.   : 0.000   INFRA   :2850   Chennai   :3150   Min.   :20.00
 1st Qu.: 3.000   ERS     :2426   Noida     :2727   1st Qu.:27.00
 Median : 4.000   BFSI    :1396   Bangalore:2230   Median :29.00
 Mean   : 4.239   ETS     : 691   Hyderabad: 341   Mean   :29.91
 3rd Qu.: 6.000   CSMP    : 579   Mumbai    : 197   3rd Qu.:34.00
 Max.   :24.000   AXON    : 568   Gurgaon   : 146   Max.   :60.00
                  (Other): 485   (Other)   : 204
        Status
 Joined    :7313
 Not Joined:1682
```

Create a new data frame and store the raw data copy. This is being done to have a copy of the raw data intact for further manipulation if needed.

```
In [5]: filter_df <- na.omit(raw_df) # listwise deletion of missing
```

1.1.3 3. Create train and test dataset

Reserve 80% for *training* and 20% of *test* *Correct the error in the below code chunk*

```
In [6]: set.seed(2341)
        trainIndex <- createDataPartition(filter_df$Status, p = 0.80, list = FALSE)
        train_df <- filter_df[trainIndex,]
        test_df <- filter_df[-trainIndex,]
```

We can pull the specific attribute needed to build the model is another data frame. This agian is more of a hygine practice to not touch the **train** and **test** data set directly.

```
In [7]: naive_train_df <- as.data.frame(train_df[,c("DOJ.Extended",
                                                  "Duration.to.accept.offer",
                                                  #"Notice.period",
                                                  "Offered.band",
                                                  #"Percent.difference.CTC",
                                                  #"Joining.Bonus",
                                                  #"Gender",
                                                  #"Candidate.Source",
                                                  #"Rex.in.Yrs",
                                                  "LOB",
                                                  #"Location",
                                                  #"Age",
                                                  "Status"
        )])
```

Correct the error in the below code chunk

```
In [8]: naive_test_df <- as.data.frame(test_df[,c("DOJ.Extended",
                                                "Duration.to.accept.offer",
                                                #"Notice.period",
                                                "Offered.band",
                                                #"Percent.difference.CTC",
                                                #"Joining.Bonus",
                                                #"Gender",
                                                #"Candidate.Source",
                                                #"Rex.in.Yrs",
                                                "LOB",
                                                #"Location",
                                                #"Age",
                                                "Status"
        )])
```

1.2 Model Building: Using the caret() package

There are a number of models which can be built using caret package. To get the names of all the models possible.

```
In [9]: names(getModelInfo())
```

1. 'ada' 2. 'AdaBag' 3. 'AdaBoost.M1' 4. 'adaboost' 5. 'amdai' 6. 'ANFIS' 7. 'avNNet' 8. 'awnb' 9. 'awtan' 10. 'bag' 11. 'bagEarth' 12. 'bagEarthGCV' 13. 'bagFDA' 14. 'bagFDAGCV' 15. 'bam' 16. 'bartMachine' 17. 'bayesglm' 18. 'binda' 19. 'blackboost' 20. 'blasso' 21. 'blassoAveraged' 22. 'bridge' 23. 'brnn' 24. 'BstLm' 25. 'bstSm' 26. 'bstTree' 27. 'C5.0' 28. 'C5.0Cost' 29. 'C5.0Rules' 30. 'C5.0Tree' 31. 'cforest' 32. 'chaid' 33. 'CSimca' 34. 'ctree' 35. 'ctree2' 36. 'cubist' 37. 'dda' 38. 'deepboost' 39. 'DENFIS' 40. 'dnn' 41. 'dwdLinear' 42. 'dwdPoly' 43. 'dwdRadial' 44. 'earth' 45. 'elm' 46. 'enet' 47. 'evtree' 48. 'extraTrees' 49. 'fda' 50. 'FH.GBML' 51. 'FIR.DM' 52. 'foba' 53. 'FRBCS.CHI' 54. 'FRBCS.W' 55. 'FS.HGD' 56. 'gam' 57. 'gamboost' 58. 'gamLoess' 59. 'gamSpline' 60. 'gaussprLinear' 61. 'gaussprPoly' 62. 'gaussprRadial' 63. 'gbm_h2o' 64. 'gbm' 65. 'gcvEarth' 66. 'GFS.FR.MOGUL' 67. 'GFS.LT.RS' 68. 'GFS.THRIFT' 69. 'glm.nb' 70. 'glm' 71. 'glmboost' 72. 'glmnet_h2o' 73. 'glmnet' 74. 'glmStepAIC' 75. 'gpls' 76. 'hda' 77. 'hdda' 78. 'hdrda' 79. 'HYFIS' 80. 'icr' 81. 'J48' 82. 'JRip' 83. 'kernelpls' 84. 'kknn' 85. 'knn' 86. 'krlsPoly' 87. 'krlsRadial' 88. 'lars' 89. 'lars2' 90. 'lasso' 91. 'lda' 92. 'lda2' 93. 'leapBackward' 94. 'leapForward' 95. 'leapSeq' 96. 'Linda' 97. 'lm' 98. 'lmStepAIC' 99. 'LMT' 100. 'loclda' 101. 'logicBag' 102. 'LogitBoost' 103. 'logreg' 104. 'lssvmLinear' 105. 'lssvmPoly' 106. 'lssvmRadial' 107. 'lvq' 108. 'M5' 109. 'M5Rules' 110. 'manb' 111. 'mda' 112. 'Mlda' 113. 'mlp' 114. 'mlpKerasDecay' 115. 'mlpKerasDecayCost' 116. 'mlpKerasDropout' 117. 'mlpKerasDropoutCost' 118. 'mlpML' 119. 'mlpSGD' 120. 'mlpWeightDecay' 121. 'mlpWeightDecayML' 122. 'monmlp' 123. 'msaenet' 124. 'multinom' 125. 'mxnet' 126. 'mxnetAdam' 127. 'naive_bayes' 128. 'nb' 129. 'nbDiscrete' 130. 'nbSearch' 131. 'neuralnet' 132. 'nnet' 133. 'nnls' 134. 'nodeHarvest' 135. 'null' 136. 'OneR' 137. 'ordinalNet' 138. 'ORFlog' 139. 'ORFpls' 140. 'ORFridge' 141. 'ORFsvm' 142. 'ownn' 143. 'pam' 144. 'parRF' 145. 'PART' 146. 'partDSA' 147. 'pcaNNet' 148. 'pcr' 149. 'pda' 150. 'pda2' 151. 'penalized' 152. 'PenalizedLDA' 153. 'plr' 154. 'pls' 155. 'plsRglm' 156. 'polr' 157. 'ppr' 158. 'PRIM' 159. 'protoclass' 160. 'pythonKnnReg' 161. 'qda' 162. 'QdaCov' 163. 'qrf' 164. 'qrnn' 165. 'randomGLM' 166. 'ranger' 167. 'rbf' 168. 'rbfDDA' 169. 'Rborist' 170. 'rda' 171. 'regLogistic' 172. 'relaxo' 173. 'rf' 174. 'rFerns' 175. 'RFlda' 176. 'rfRules' 177. 'ridge' 178. 'rlda' 179. 'rlm' 180. 'rmda' 181. 'rocc' 182. 'rotationForest' 183. 'rotationForestCp' 184. 'rpart' 185. 'rpart1SE' 186. 'rpart2' 187. 'rpartCost' 188. 'rpartScore' 189. 'rqlasso' 190. 'rqnc' 191. 'RRF' 192. 'RRFglobal' 193. 'rrlda' 194. 'RSimca' 195. 'rvmLinear' 196. 'rvmPoly' 197. 'rvmRadial' 198. 'SBC' 199. 'sda' 200. 'sdwd' 201. 'simpls' 202. 'SLAVE' 203. 'slda' 204. 'smda' 205. 'snn' 206. 'sparseLDA' 207. 'spikeslab' 208. 'spls' 209. 'stepLDA' 210. 'stepQDA' 211. 'superpc' 212. 'svmBoundrangeString' 213. 'svmExpoString' 214. 'svmLinear' 215. 'svmLinear2' 216. 'svmLinear3' 217. 'svmLinearWeights' 218. 'svmLinearWeights2' 219. 'svmPoly' 220. 'svmRadial' 221. 'svmRadialCost' 222. 'svmRadialSigma' 223. 'svmRadialWeights' 224. 'svmSpectrumString' 225. 'tan' 226. 'tanSearch' 227. 'treebag' 228. 'vbmpRadial' 229. 'vglmAdjCat' 230. 'vglmContRatio' 231. 'vglmCumulative' 232. 'widekernelpls' 233. 'WM' 234. 'wsrf' 235. 'xgbDART' 236. 'xgbLinear' 237. 'xgbTree' 238. 'xyf'

To get the info on specific model:

```
In [10]: getModelInfo()$glm$type
```

1. 'Regression' 2. 'Classification'

The below chunk of code is standarized way of building model using caret package. Setting in the control parameters for the model.

```
In [11]: set.seed(1234)
         objControl <- trainControl(method = "cv", number = 2, returnResamp = 'none',
                                    summaryFunction = twoClassSummary,
                                    #summaryFunction = twoClassSummary, defaultSummary
```

```
                                        classProbs = TRUE,
                                        savePredictions = TRUE)
```

The search grid is basically a model fine tuning option. The paramter inside the **expan.grid()** function varies according to model. The **complete** list of tuning paramter for different models.

A useful read on how numeric variables are taken care of by kernal density function is here: http://uc-r.github.io/naive_bayes

- usekernel parameter allows us to use a kernel density estimate for continuous variables versus a guassian density estimate,
- adjust allows us to adjust the bandwidth of the kernel density (larger numbers mean more flexible density estimate),
- fL allows us to incorporate the Laplace smoother

```
In [12]: #This parameter is for glmnet. Need not be executed if method  is glmStepAIC
         searchGrid <-  expand.grid(usekernel=c(TRUE), fL = c(1:5), adjust=c(1:5))
```

The model building starts here. > 1. **metric= "ROC"** uses ROC curve to select the best model.Accuracy, Kappa are other options. To use this change twoClassSummary to defaultSummary in **ObjControl** 2. **verbose = FALSE**: does not show the processing output on console

The factor names at times may not be consistent. R may expect **"Not.Joined"** but the actual level may be **"Not Joined"** This is corrected by using **make.names()** function to give syntactically valid names.

In case of large number of predictors and particularly (numeric predicators), the naive bayes (with kernal density estimation) may end up giving warning messages `Numerical 0 probability for all classes with observation....`

Please refer to the post to know about the issue: https://github.com/topepo/caret/issues/793

```
In [13]: #naive_train_df$StatusFactor <- as.factor(ifelse(naive_train_df$Status == "Joined", 1
         set.seed(766)
         levels(naive_train_df$Status) <- make.names(levels(factor(naive_train_df$Status)))
         naive_caret_model <- train(naive_train_df[,1:4],
                                    naive_train_df[,5],
                                    method = 'nb', #'glm', glmnet
                                    trControl = objControl,
                                    tuneGrid= searchGrid,
                                    metric = "ROC")
```

1.3 Model Evaluation

1.3.1 1. One useful plot from caret package is the variable importance plot

In case you get an error "Invalid Graphic state", uncomment the line below

```
In [14]: naive_caret_model
         summary(naive_caret_model$finalModel)

         #dev.off()
         #plot(varImp(naive_caret_model, scale = TRUE))
```

```
Naive Bayes

7197 samples
   4 predictor
   2 classes: 'Joined', 'Not.Joined'

No pre-processing
Resampling: Cross-Validated (2 fold)
Summary of sample sizes: 3598, 3599
Resampling results across tuning parameters:

  fL  adjust  ROC        Sens       Spec
  1   1       0.6128812  0.9976074  0.026002972
  1   2       0.5966586  0.9984618  0.021545319
  1   3       0.5848701  0.9988036  0.013372957
  1   4       0.5834954  0.9989745  0.006686478
  1   5       0.5852380  0.9993164  0.004457652
  2   1       0.6126230  0.9976074  0.026002972
  2   2       0.5956725  0.9982909  0.021545319
  2   3       0.5836627  0.9988036  0.013372957
  2   4       0.5827498  0.9989745  0.006686478
  2   5       0.5845042  0.9993164  0.004457652
  3   1       0.6125996  0.9976074  0.026002972
  3   2       0.5954838  0.9982909  0.022288262
  3   3       0.5835337  0.9988036  0.014858841
  3   4       0.5825063  0.9989745  0.006686478
  3   5       0.5844353  0.9993164  0.004457652
  4   1       0.6127178  0.9974365  0.026002972
  4   2       0.5957735  0.9982909  0.022288262
  4   3       0.5837206  0.9986327  0.015601783
  4   4       0.5827699  0.9988036  0.008172363
  4   5       0.5846573  0.9991455  0.004457652
  5   1       0.6129023  0.9974365  0.026002972
  5   2       0.5957690  0.9981201  0.022288262
  5   3       0.5841314  0.9986327  0.015601783
  5   4       0.5828613  0.9988036  0.008172363
  5   5       0.5848873  0.9991455  0.005200594

Tuning parameter 'usekernel' was held constant at a value of TRUE
ROC was used to select the optimal model using the largest value.
The final values used for the model were fL = 5, usekernel = TRUE and adjust
 = 1.


            Length Class      Mode
apriori     2      table      numeric
tables      4      -none-     list
levels      2      -none-     character
```

```
call        6      -none-     call
x           4      data.frame list
usekernel   1      -none-     logical
varnames    4      -none-     character
xNames      4      -none-     character
problemType 1      -none-     character
tuneValue   3      data.frame list
obsLevels   2      -none-     character
param       0      -none-     list
```

1.3.2 2. The prediction and confusion Matrix on train data.

The syntax for prediction in caret is almost similar expect the the **type** attribute expects input as **'raw'** or **'prob'**. In case of prob, the predicted value holds the probability of both positive and negative class.

```
In [15]: #Missing code. May result in error
         levels(naive_train_df$Status) <- make.names(levels(factor(naive_train_df$Status)))
         caretPredictedClass <- predict(object = naive_caret_model, naive_train_df[,1:4], type
         confusionMatrix(caretPredictedClass,naive_train_df$Status)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   Joined Not.Joined
  Joined       5838       1310
  Not.Joined     13         36

               Accuracy : 0.8162
                 95% CI : (0.807, 0.8251)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.2487

                  Kappa : 0.039
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.99778
            Specificity : 0.02675
         Pos Pred Value : 0.81673
         Neg Pred Value : 0.73469
             Prevalence : 0.81298
         Detection Rate : 0.81117
   Detection Prevalence : 0.99319
      Balanced Accuracy : 0.51226

       'Positive' Class : Joined
```

1.3.3 4. Confusion Matrix on the test data

The **predict** function is used to get the predicted probability on the new dataset. The probability value along with the optimal cut-off can be used to build confusion matrix

```
In [16]: test_predicted_prob = predict(naive_caret_model, naive_test_df, type = "prob")

         #variable with all the values as joined
         n <- length(naive_test_df$Status)
         predicted_y = rep("Not Joined", n)

         # defining log odds in favor of not joining
         predicted_y[test_predicted_prob[1] > test_predicted_prob[2]] = "Joined"

         #add the model_precition in the data
         naive_test_df$predicted_y <- predicted_y

         ###Create the confusionmatrix###
         addmargins(table(naive_test_df$Status, naive_test_df$predicted_y))
         mean(naive_test_df$predicted_y == naive_test_df$Status)
```

```
Warning message in FUN(X[[i]], ...):
Numerical 0 probability for all classes with observation 1317
```

	Joined	Not Joined	Sum
Joined	1458	4	1462
Not Joined	328	8	336
Sum	1786	12	1798

0.815350389321468

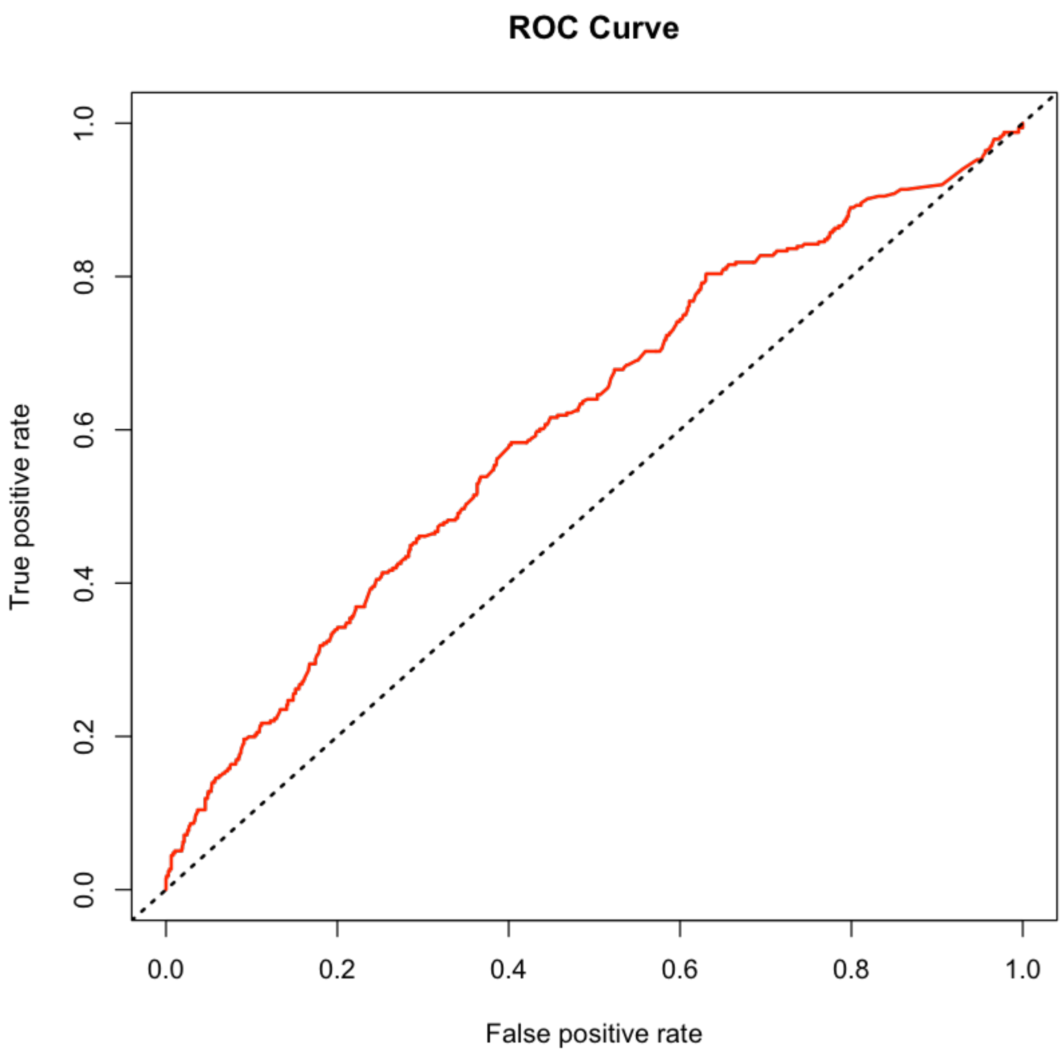
1.3.4 5. ROC Plot on the test data

ROCR package can be used to evaluate the model performace on the test data. The same package can also be used to get the model performace on the test data.

```
In [17]: #error in below line
         lgPredObj <- prediction(test_predicted_prob[2],naive_test_df$Status)
         lgPerfObj <- performance(lgPredObj, "tpr","fpr")
         plot(lgPerfObj,main = "ROC Curve",col = 2,lwd = 2)
         abline(a = 0,b = 1,lwd = 2,lty = 3,col = "black")
         performance(lgPredObj, "auc")
```

```
An object of class "performance"
Slot "x.name":
[1] "None"

Slot "y.name":
[1] "Area under the ROC curve"
```

```
Slot "alpha.name":
[1] "none"

Slot "x.values":
list()

Slot "y.values":
[[1]]
[1] 0.6102809


Slot "alpha.values":
list()
```



End of Document