Predicting Client Subscriptions Decision, a Potential Business Solution

Liu, Donghan; Shen, Xiaolong

Nov 18, 2018

Objective

The main task here is predicting whether the client will subscribe to a term deposit. To achieve that we have several small tasks, and they are as follow,
1. Preliminary analysis to get an overlook of the data
2. Preparing data for building model
3. Model fiting and tunning
4. Model validation

Data Preparation

Reading Raw Data

```
raw_data = read.csv("bank-additional-train.csv")
```

Reading Data with H2o

```
df = h2o.importFile(path = normalizePath("bank-additional-train.csv"))
```

The implemented statistical package are listed above and the data was successfully read

Preliminary Analysis

$- - - - - - - $ *Table for statistics of y* $- - - - - - -$

```
table(raw_data$y)
```

```
##
##    no   yes
## 32908  4161
```

$- - - - - - - $ *Summary of each predictor* $- - - - - - -$

```
summary(raw_data)
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
##       age                  job            marital     
##  Min.   :17.00   admin.     :9380   divorced: 4159  
##  1st Qu.:32.00   blue-collar:8331   married :22426  
##  Median :38.00   technician :6075   single  :10410  
##  Mean   :40.04   services   :3553   unknown :   74  
##  3rd Qu.:47.00   management :2646                   
##  Max.   :98.00   retired    :1536                   
##                  (Other)    :5548                   
##                education        default          housing     
##  university.degree  :10931   no     :29301   no     :16753  
##  high.school        : 8555   unknown: 7767   unknown:  869  
##  basic.9y           : 5449   yes    :    1   yes    :19447  
##  professional.course: 4709                                  
##  basic.4y           : 3790                                  
##  basic.6y           : 2060                                  
##  (Other)            : 1575                                  
##       loan            contact           month       day_of_week
##  no     :30595   cellular :23506   may    :12351   fri:7030   
##  unknown:  869   telephone:13563   jul    : 6474   mon:7673   
##  yes    : 5605                     aug    : 5576   thu:7757   
##                                    jun    : 4787   tue:7269   
##                                    nov    : 3701   wed:7340   
##                                    apr    : 2379              
##                                    (Other): 1801              
##     duration         campaign          pdays          previous     
##  Min.   :   0.0   Min.   : 1.000   Min.   :  0.0   Min.   :0.0000  
##  1st Qu.: 102.0   1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.0000  
##  Median : 180.0   Median : 2.000   Median :999.0   Median :0.0000  
##  Mean   : 258.4   Mean   : 2.574   Mean   :962.5   Mean   :0.1726  
##  3rd Qu.: 320.0   3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.0000  
##  Max.   :4918.0   Max.   :56.000   Max.   :999.0   Max.   :7.0000  
##                                                                    
##         poutcome      emp.var.rate      cons.price.idx  cons.conf.idx  
##  failure    : 3820   Min.   :-3.40000   Min.   :92.20   Min.   :-50.8  
##  nonexistent:32018   1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7  
##  success    : 1231   Median : 1.10000   Median :93.75   Median :-41.8  
##                      Mean   : 0.08276   Mean   :93.58   Mean   :-40.5  
##                      3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4  
##                      Max.   : 1.40000   Max.   :94.77   Max.   :-26.9  
##                                                                        
##    euribor3m      nr.employed     y        
##  Min.   :0.634   Min.   :4964   no :32908  
##  1st Qu.:1.344   1st Qu.:5099   yes: 4161  
##  Median :4.857   Median :5191              
##  Mean   :3.622   Mean   :5167              
##  3rd Qu.:4.961   3rd Qu.:5228              
##  Max.   :5.045   Max.   :5228              
##
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

$- - - - - - - \textit{The existence of NA value is}: - - - - - - - -$

```
any(is.na(raw_data))
```

```
## [1] FALSE
```

In this step, we are trying to make a summary table of data in order to bring the data statistics in an intuitional way.

From the result above we know that the data is highly imbalanced, which suggests that we might need to do oversampling or undersampling later. Also from the summary of the data we can see that we have 10 categorical variables and 10 numeric variables, which suggest that we might want to recode categorical data into dummy variables in the future for some model fitting. Besides, we don't have any missing value in the dataset.

$- - - - - - - \textit{Data Explore} - - - - - - -$

```
yes_idx=which(raw_data$y=="yes")
yes_data=raw_data[yes_idx,]

(table_job=sort(table(yes_data$job),decreasing=T))
```

```
## 
##        admin.    technician   blue-collar       retired    management 
##          1221           645           576           395           294 
##      services       student self-employed    unemployed  entrepreneur 
##           282           246           128           124           116 
##     housemaid       unknown 
##            98            36
```

```
(table_marital=sort(table(yes_data$marital),decreasing=T))
```

```
## 
##  married   single divorced  unknown 
##     2262     1456      431       12
```

```
(table_education=sort(table(yes_data$education),decreasing=T))
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
##
##   university.degree         high.school professional.course
##                1486                 910                 537
##             basic.9y            basic.4y             unknown
##                 436                 381                 235
##             basic.6y          illiterate
##                 172                   4
```

```
(table_default=sort(table(yes_data$default),decreasing=T))
```

```
##
##      no unknown     yes
##    3765     396       0
```

```
(table_housing=sort(table(yes_data$housing),decreasing=T))
```

```
##
##     yes      no unknown
##    2243    1825      93
```

```
(table_contact=sort(table(yes_data$contact),decreasing=T))
```

```
##
##  cellular telephone
##      3449       712
```

```
(table_month=sort(table(yes_data$month),decreasing=T))
```

```
##
## may aug jul jun apr nov oct mar sep dec
## 797 588 583 493 480 382 281 256 224  77
```

```
(table_day_of_week=sort(table(yes_data$day_of_week),decreasing=T))
```

```
##
## thu tue wed mon fri
## 937 858 847 767 752
```

```
(table_poutcome=sort(table(yes_data$poutcome),decreasing=T))
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
##
## nonexistent     success     failure
##        2810         805         546
```

This is a data exploration for people who subscripe. From the result above, we can see that people who subscript are more likely to have an administration job, married, have an university degree, has home load, do not have credit in default, and was contacted by cellphone.

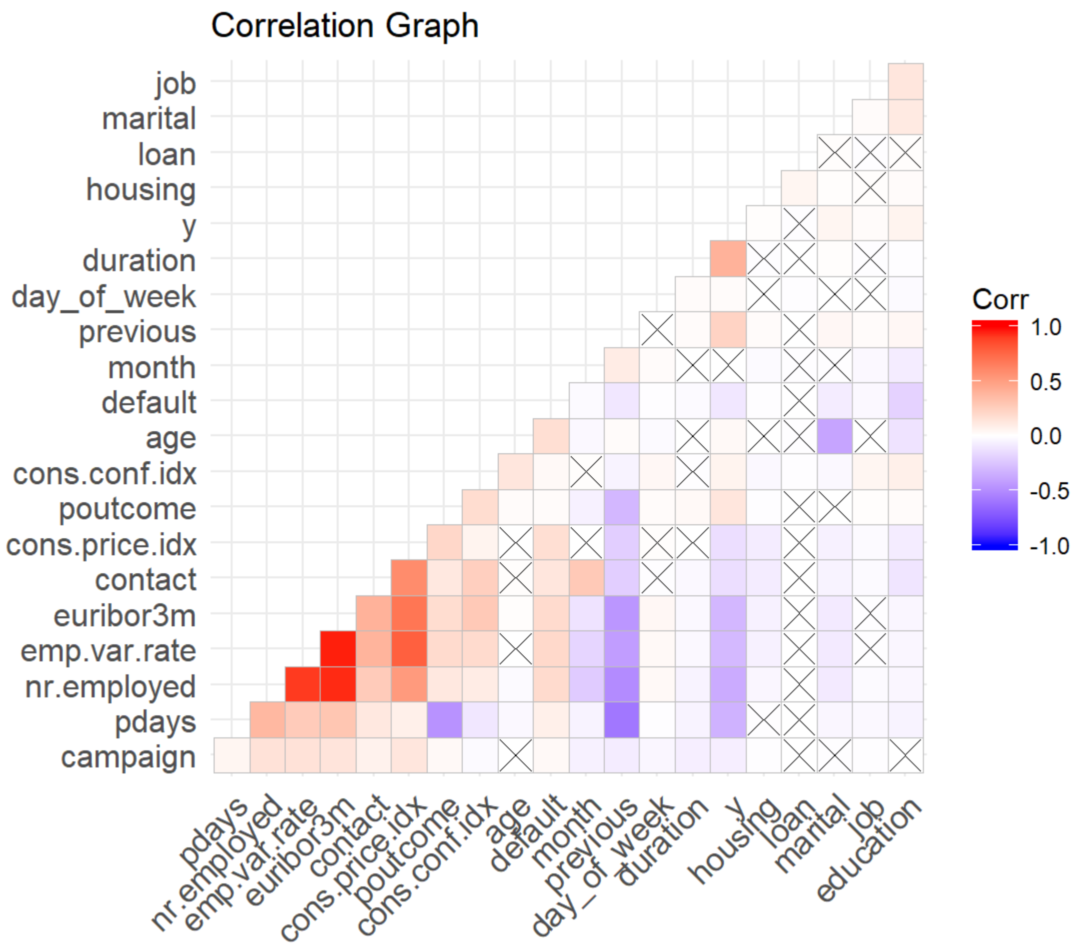
Data Type Conversion

```
#Convert non-numeric predictor to numeric for the process of redundant data removal

fac_var = c("job","marital","education","default","housing","loan","contact","month","day_of_week"
,"poutcome","y")
convert = function(dat){
  t = as.numeric(raw_data[,dat])
  return (t)
}

data = raw_data
for (i in fac_var){
  data[,i] = convert(i)
}
data[,"y"] = factor(data[,"y"])
```

Since the predictors are potentially highly correlated with each other, there might be some redundant information that needs to be removed regarding the elimination of noise. We decided to make the correlation table to see what might be the redundant variables. In order to do so, we need to transform our variables as numeric variables.

Correlation Graph

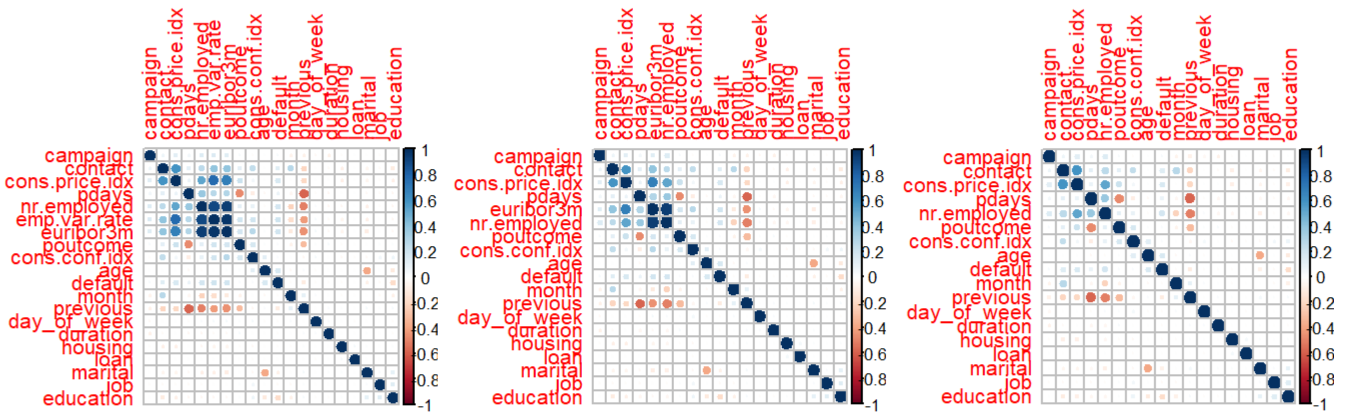
Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



Correlation Demonstration

```
#Eliminate highly correlated data
par(mfrow=c(1,3))

corr = corrplot(cor(data[,-21]), order = "hclust")

highlyCorrelated <- findCorrelation(corr, cutoff=0.75, name = TRUE)

corr1 = corrplot(cor(subset( data, select = -c(16, 21 ) )), order = "hclust")

highlyCorrelated1 <- findCorrelation(corr1, cutoff=0.75, name = TRUE)

corr2 = corrplot(cor(data[,c(-16,-19,-21)]), order = "hclust")
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



```
highlyCorrelated2 <- findCorrelation(corr2, cutoff=0.75, verbose = TRUE, name = TRUE)
```

```
## All correlations <= 0.75
```

```
bank_new = data[,c(-16,-19)]
```

After the data type transformation, we generated the correlation plot and tried to find the highly correlated predictors. The cutoff is setted as 0.75, which is a common level for identifying high correlation. Later, we removed the predictor once at a time. Column number 16 and 19 are considered as redudant columns.

Information Value Comparison

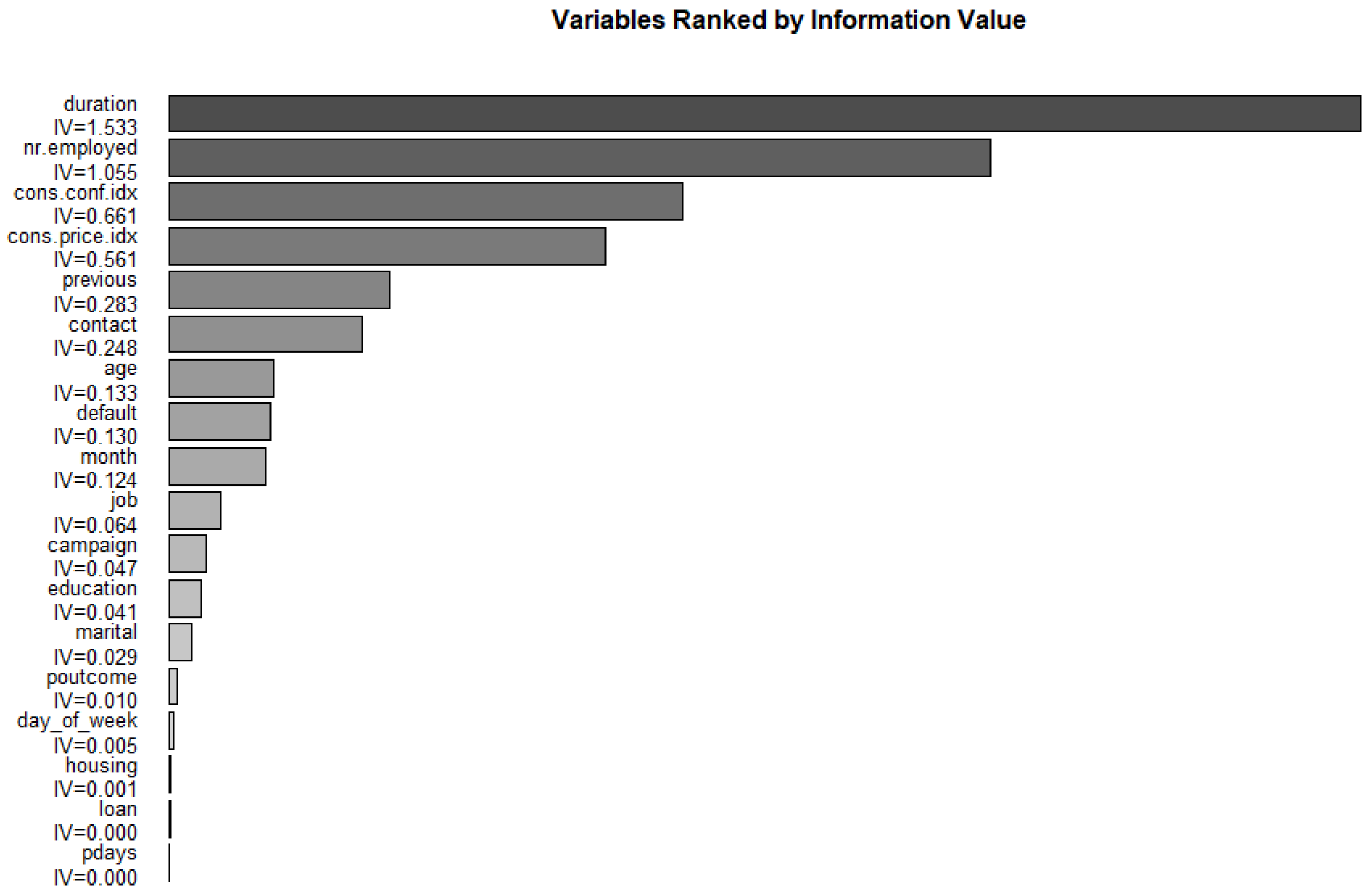
The information value are calculated based on the following formula

$$IV = \Sigma(\%\ of\ \ nonevents - \%\ of\ events) * WOE$$

Information value is one of the most useful technique to select important variables in a predictive model. It helps to rank variables on the basis of their importance, by way of example, higher information value is equivalent to higher predictive power.

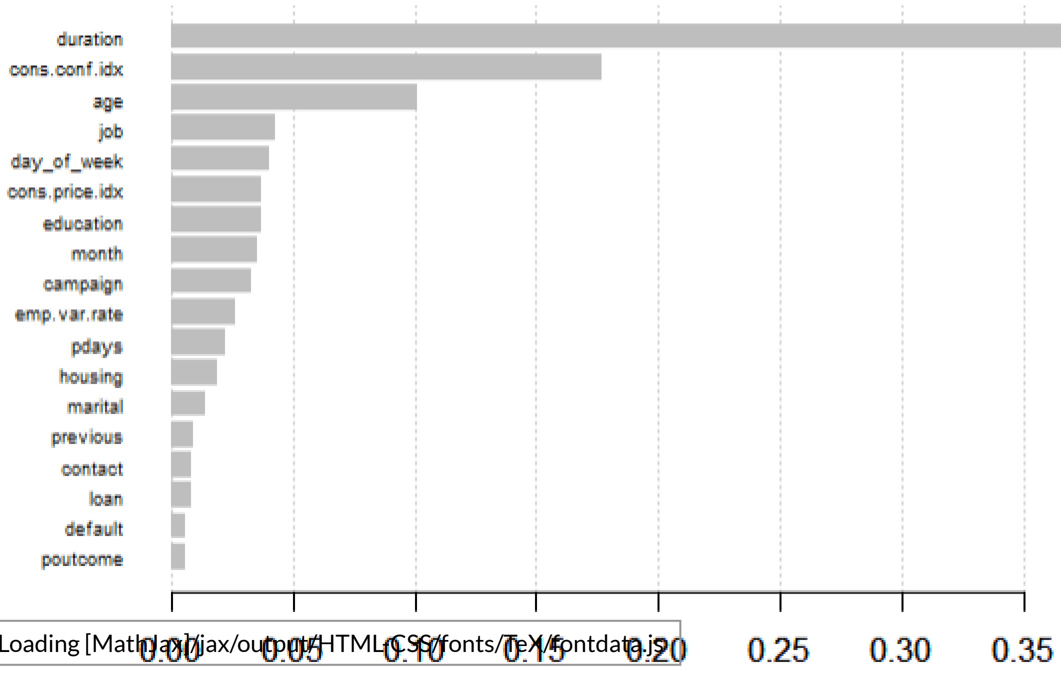
Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

The information values are shown above, the “Duration” variable is obviously the most influential variable and pdays does not perform well in the potential model with the lowest score. However, “duration” won’t help us much in the prediction task as mentioned in the data description of the competition, and we will only use it for benchmark and remove it for actual model fitting.



Information value comparison

Importance



Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

Other than the information value, in order to be certain about the importance of predictors, this importance plot indicates that duration also has the most significant impact. However, the rank of some predictors, such as pdays, has changed. Regarding the ranking changes, it is not surprising because a different algorithm will generate a relatively different result. Notably, their position of ranking does not change much, namely, they stay at the same level. For instance, the pdays is still the least important variable and will be removed due to its meaningless.

```
#Comparison and integration
#Remove noise for better training
data_new = bank_new[,-c(13,7,6,15,3)]

df = as.h2o(data_new)
```

Take the previous statement into account, we decided to discard columns of 3,6,7,13,15.

From here, we completed the process of data preparation.

Model training

Before actually deploying different algorithms, we first define several terminologies.

TN / True Negative: case was negative and predicted negative TP / True Positive: case was positive and predicted positive FN / False Negative: case was positive but predicted negative FP / False Positive: case was negative but predicted positive

In this specific case, the negative regarding y = “no”, while the positive refers to y = “yes”

$$Precision = \frac{TP}{FP + TP}$$

$$Accuracy = \frac{TN + TP}{FN + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Performance Metrics Selection

As we discovered before, the dataset is highly imbalanced, so simply looking at accuracy won¡¯t tell us too much about how good the model is performing. In our case, we choose to choose the model based on the precision of prediction yes. And we will display ROC and AUC for our final model as well.

Including “Duration” for Benchmark

$$-------Benchmarked\ Random\ Forest\ model\ summary-------$$

```
rf_ben@model$validation_metrics
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
## H2OBinomialMetrics: drf
## ** Reported on validation data. **
##
## MSE:  0.06516738
## RMSE:  0.255279
## LogLoss:  0.2242763
## Mean Per-Class Error:  0.1814323
## AUC:  0.9292477
## Gini:  0.8584954
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1    2    Error       Rate
## 1      6052  525 0.079824  =525/6577
## 2       242  613 0.283041   =242/855
## Totals 6294 1138 0.103202  =767/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.323070 0.615153 219
## 2                       max f2  0.117577 0.735997 317
## 3                 max f0point5  0.410511 0.582557 180
## 4                 max accuracy  0.450922 0.903660 166
## 5                max precision  0.982323 1.000000   0
## 6                   max recall  0.000005 1.000000 399
## 7              max specificity  0.982323 1.000000   0
## 8             max absolute_mcc  0.262491 0.565677 247
## 9   max min_per_class_accuracy  0.152803 0.859510 299
## 10 max mean_per_class_accuracy  0.077047 0.873773 342
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>, xval=<T/F>)`
```

$- - - - - - - -$ *Benchmarked Random Forest model percision* $- - - - - - - -$

```
##prediciont
finalRf_predictions<-h2o.predict(
  object = rf_ben
  ,newdata = test)

pred=finalRf_predictions[,1]
test_label=test$y
(precision=sum(pred[test_label=="2"]=="2")/(sum(pred[test_label=="2"]=="2")+sum(pred[test_label=="1"]=="2")))
```

```
## [1] 0.5288809
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

From the result above we know that duration is highly influential for the prediction task, give with that variable, we are able to get 0.52888 precision for predicting yes even without tuning the model. However, as the task mentioned, we shouldn't use "duration" for prediction. The following are models we used without variable "duration"

Excluding Duration and Training Model

```
##excluding duration
data_new=df
data_new=data_new[,-which(names(data_new) == "duration")]
## train valid test split, 60% trianing, 20% validation, 20% testing
splits <- h2o.splitFrame(
  data_new,
  c(0.6,0.2), seed=1234)

train <- h2o.assign(splits[[1]], "train.hex")
valid <- h2o.assign(splits[[2]], "valid.hex")
test <- h2o.assign(splits[[3]], "test.hex")
```

The processed and manipulated data are prepared, then we move forward to slip the data into train and test for cross-validation, a popular method that often use in predicting of machine learning problem, we test-train divide the data with 60% of them are training data, 20% of them are validation data and the rest are testing data.

Oversampling

As discussed before, the approach of oversampling or undersampling will be implemented below because of the imbalanced label. Eventually, We decided to oversample the response variable in our training set, so that we have balanced counts of "yes" and "no" as our response in order to build a stronger classifier.

```
##doing oversample for training set
train=as.data.frame(train)
train = SMOTE(y ~ ., train, perc.over = 700,perc.under=100)
temp_trn = train
train=as.h2o(train)
table(temp_trn$y)
```

```
##
##     1     2
## 17318 19792
```

From the result above we can see that the training set is balanced now.

Model Training Without "Duration"

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

— — — *Random Forest Evaluation Results* — — — — — —

```
rf1@model$validation_metrics
```

```
## H2OBinomialMetrics: drf
## ** Reported on validation data. **
##
## MSE:  0.1140303
## RMSE:  0.3376837
## LogLoss:  0.3887436
## Mean Per-Class Error:  0.3163855
## AUC:  0.7480486
## Gini:  0.4960972
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1   2    Error        Rate
## 1      6023 554 0.084233   =554/6577
## 2       469 386 0.548538    =469/855
## Totals 6492 940 0.137648  =1023/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.504455 0.430084 128
## 2                       max f2  0.353242 0.496568 181
## 3                 max f0point5  0.618561 0.440559  96
## 4                 max accuracy  0.872615 0.889128  30
## 5                max precision  0.940327 0.641975  14
## 6                   max recall  0.013165 1.000000 397
## 7              max specificity  0.999745 0.999696   0
## 8             max absolute_mcc  0.504455 0.352518 128
## 9   max min_per_class_accuracy  0.234646 0.691228 239
## 10 max mean_per_class_accuracy  0.353242 0.708504 181
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, vali
d=<T/F>, xval=<T/F>)`
```

After getting out first random forest model, we have several ways to tune to model
1: Adding trees will help. The default is 50.
2: changing the depth will help.Changing the depth means you are adjusting the “weakness” of each learner. Adding depth makes each tree fit the data closer, and vice versa.

$-------Random\ Forest\ Evaluation\ Results-------$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
## H2OBinomialMetrics: drf
## ** Reported on validation data. **
##
## MSE:  0.1481489
## RMSE:  0.3849012
## LogLoss:  0.4791892
## Mean Per-Class Error:  0.2824849
## AUC:  0.775447
## Gini:  0.5508941
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1    2    Error       Rate
## 1      5992  585 0.088946  =585/6577
## 2       407  448 0.476023   =407/855
## Totals 6399 1033 0.133477  =992/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.577412 0.474576 123
## 2                       max f2  0.413085 0.532078 184
## 3                 max f0point5  0.752670 0.472557  71
## 4                 max accuracy  0.858239 0.890474  33
## 5                max precision  0.945057 0.807692   6
## 6                   max recall  0.083229 1.000000 396
## 7              max specificity  0.969877 0.999848   0
## 8             max absolute_mcc  0.577412 0.401246 123
## 9   max min_per_class_accuracy  0.353920 0.704094 217
## 10 max mean_per_class_accuracy  0.474219 0.732165 156
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, vali
d=<T/F>, xval=<T/F>)`
```

From the result above, we can see that error rate for predicting yes is 0.476, which is getting smaller, indicating that we are getting to the right direction

$- - - - - - - \textit{Gradient Boosting Machine } 1 \textit{ Evaluation Results} - - - - - - -$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
## H2OBinomialMetrics: gbm
## ** Reported on validation data. **
##
## MSE:  0.1115826
## RMSE:  0.3340398
## LogLoss:  0.3804041
## Mean Per-Class Error:  0.2764556
## AUC:  0.778049
## Gini:  0.5560979
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1    2    Error       Rate
## 1      5979  598 0.090923  =598/6577
## 2       395  460 0.461988   =395/855
## Totals 6374 1058 0.133611  =993/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.480005 0.480920 141
## 2                       max f2  0.306846 0.528967 208
## 3                 max f0point5  0.730919 0.468924  58
## 4                 max accuracy  0.800946 0.892223  33
## 5                max precision  0.923709 1.000000   0
## 6                   max recall  0.087950 1.000000 392
## 7              max specificity  0.923709 1.000000   0
## 8             max absolute_mcc  0.480005 0.408265 141
## 9   max min_per_class_accuracy  0.236474 0.709942 248
## 10 max mean_per_class_accuracy  0.426733 0.730410 158
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, vali
d=<T/F>, xval=<T/F>)`
```

The default gbm model has 0.46 validation error for prediction yes

$- - - - - - - Gradient\ Boosting\ Machine\ 2\ Evaluation\ Results - - - - - - -$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
## H2OBinomialMetrics: gbm
## ** Reported on validation data. **
##
## MSE:  0.1189377
## RMSE:  0.3448734
## LogLoss:  0.402636
## Mean Per-Class Error:  0.2779527
## AUC:  0.7672464
## Gini:  0.5344928
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1    2    Error        Rate
## 1      5967  610 0.092747   =610/6577
## 2       396  459 0.463158    =396/855
## Totals 6363 1069 0.135361  =1006/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.487883 0.477131 142
## 2                       max f2  0.366679 0.527721 186
## 3                 max f0point5  0.681073 0.467147  83
## 4                 max accuracy  0.796475 0.891146  37
## 5                max precision  0.904070 1.000000   0
## 6                   max recall  0.081086 1.000000 398
## 7              max specificity  0.904070 1.000000   0
## 8             max absolute_mcc  0.487883 0.403787 142
## 9   max min_per_class_accuracy  0.259983 0.697076 248
## 10 max mean_per_class_accuracy  0.366679 0.729428 186
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, vali
d=<T/F>, xval=<T/F>)`
```

The tuned gbm model has 0.47 validation error for prediction yes

The deep learning algorithms are getting popular nowsdays, and we applied a simple neurons network model with activation function of Tanh and dropout (avoid overfitting).

$-------Deep\ Learning\ Evaluation\ Results-------$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

```
## H2OBinomialMetrics: deeplearning
## ** Reported on validation data. **
## ** Metrics reported on full validation frame **
##
## MSE:  0.09394164
## RMSE:  0.306499
## LogLoss:  0.3252457
## Mean Per-Class Error:  0.2816653
## AUC:  0.7188072
## Gini:  0.4376143
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##           1    2    Error        Rate
## 1      5072 1505 0.228828  =1505/6577
## 2       286  569 0.334503    =286/855
## Totals 5358 2074 0.240985  =1791/7432
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold    value idx
## 1                       max f1  0.248342 0.388529  17
## 2                       max f2  0.248342 0.517838  17
## 3                 max f0point5  0.248342 0.310895  17
## 4                 max accuracy  0.248358 0.759957   2
## 5                max precision  0.248358 0.274405   2
## 6                   max recall  0.096618 1.000000 197
## 7              max specificity  0.248358 0.772997   0
## 8             max absolute_mcc  0.248342 0.310630  17
## 9   max min_per_class_accuracy  0.106204 0.676023 152
## 10 max mean_per_class_accuracy  0.248342 0.718335  17
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, vali
d=<T/F>, xval=<T/F>)`
```

$-------$ *Overall Models Evaluation Results* $-------$

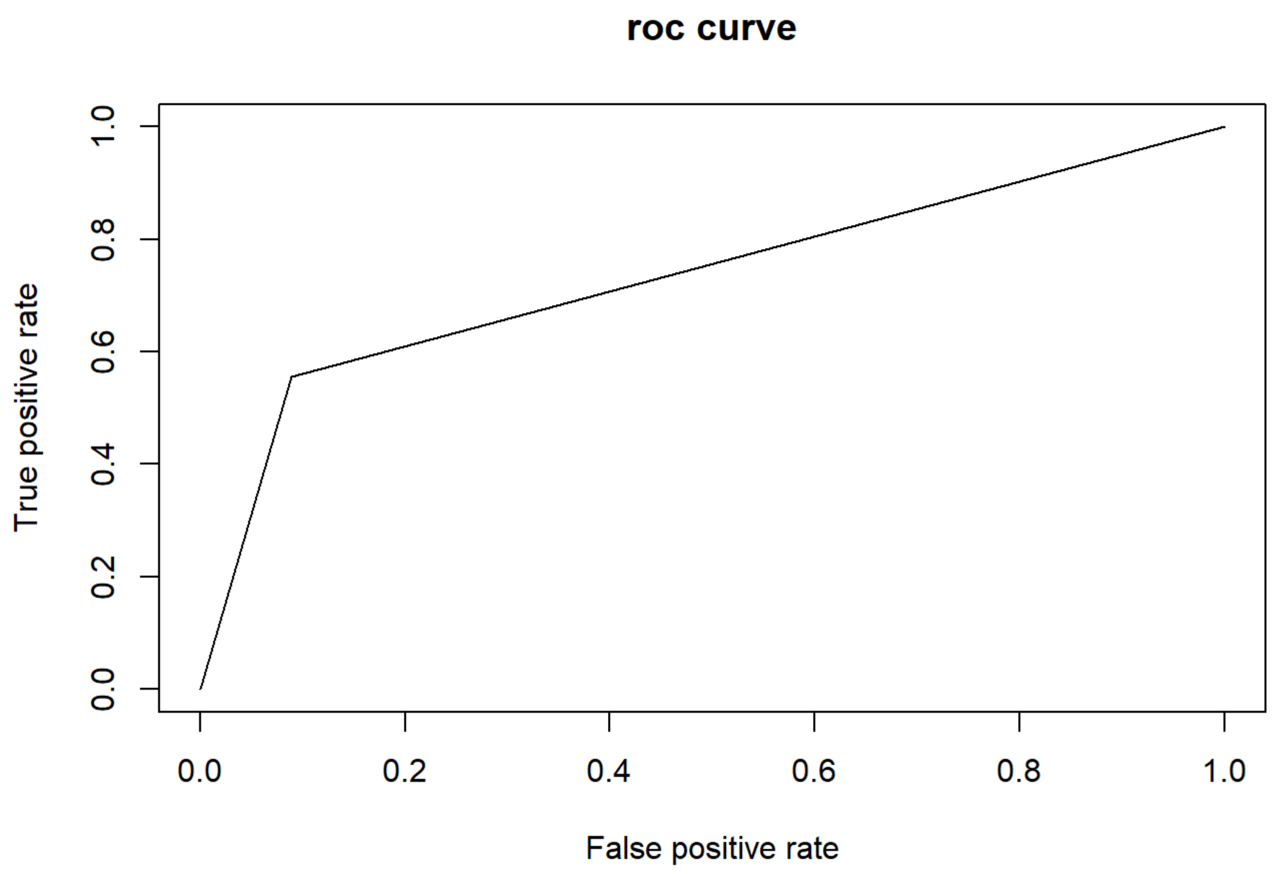
Models	'Yes' prediction error	'No' prediction error	Total error	Precision	Accuracy	AUC
RF1	0.555550	0.083473	0.133477	0.4090420	0.8622174	0.7482088
RF2	0.476023	0.088946	0.133611	0.4336883	0.8665231	0.7754470
GBM1	0.461988	0.090923	0.135361	0.4347826	0.8663886	0.7780490
GBM2	0.463158	0.092747	0.135361	0.4293732	0.8646394	0.7672464
Deep Learning (neurons network)	0.474751	0.110841	0.152718	0.3811545	0.8472820	0.7412937

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

Best Model Performance in Test Data

Based on the validation error, we choose the tuned gbm model as our best model, the following is the performance of tuned gmb in test set.

```
## [1] 0.4463768
```



```
## [1] 0.7333903
```

From the result above, we can see that the tuned GBM model has 0.446 precision for predicting yes in test set, with auc value of 0.733.

Discussion

Even though the final model was chosen, the further conversation about this project is not end. The previous experiences illustrate that the factors of predictors might take the different contribution to the model, say, let’s take a virtual example, we want to predict the number of transactions in a store based on transaction dates. Here transaction dates may not have direct correlation with number of transactions, but if we look at the day of a week, it may have a higher correlation. In this case, the information about day of a week is hidden. We need to extract it to make the model better.

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

As the data exploration part shows, there are intuitively significant different statistics of each factor variables, such as month. From an overlook, December has the lowest amount of subscriptions month, comparison with other months. Another example would be the poutcome, the amount of nonexistent is more than five times than failure and approximately 3 times than the success. We cannot say there MUST be a relationship between these factors and dependent variable, rather, it means that the data at least brings us a new way to understand the connections. We believe that the potential connection between each factor of categorical predictors will be investigated as long as the time is sufficient (we were invited after one week of competition starts).

In addition to the undiscovered relationship of hidden information of predictors and y, a possible technique that could be used to improve the performance is WoE, which refers to Weight of Evidence.

$$WoE = ln\frac{DistributionGood}{DistributionBad}$$

It helps analysts make a optimal bins to divide the data into different group. An example could be the age. Different age group might have different choice of subscriptions of term deposits, such as, younger people need ongoing cash flow but elders may not. By using WoE, this algorithm will bin the data with statistical optimal separators, for instance, age of 16-25, 26-35, 36-50, etc. Making a group of people together will tell the model that they actually have similar features and more likely to do the similar things.

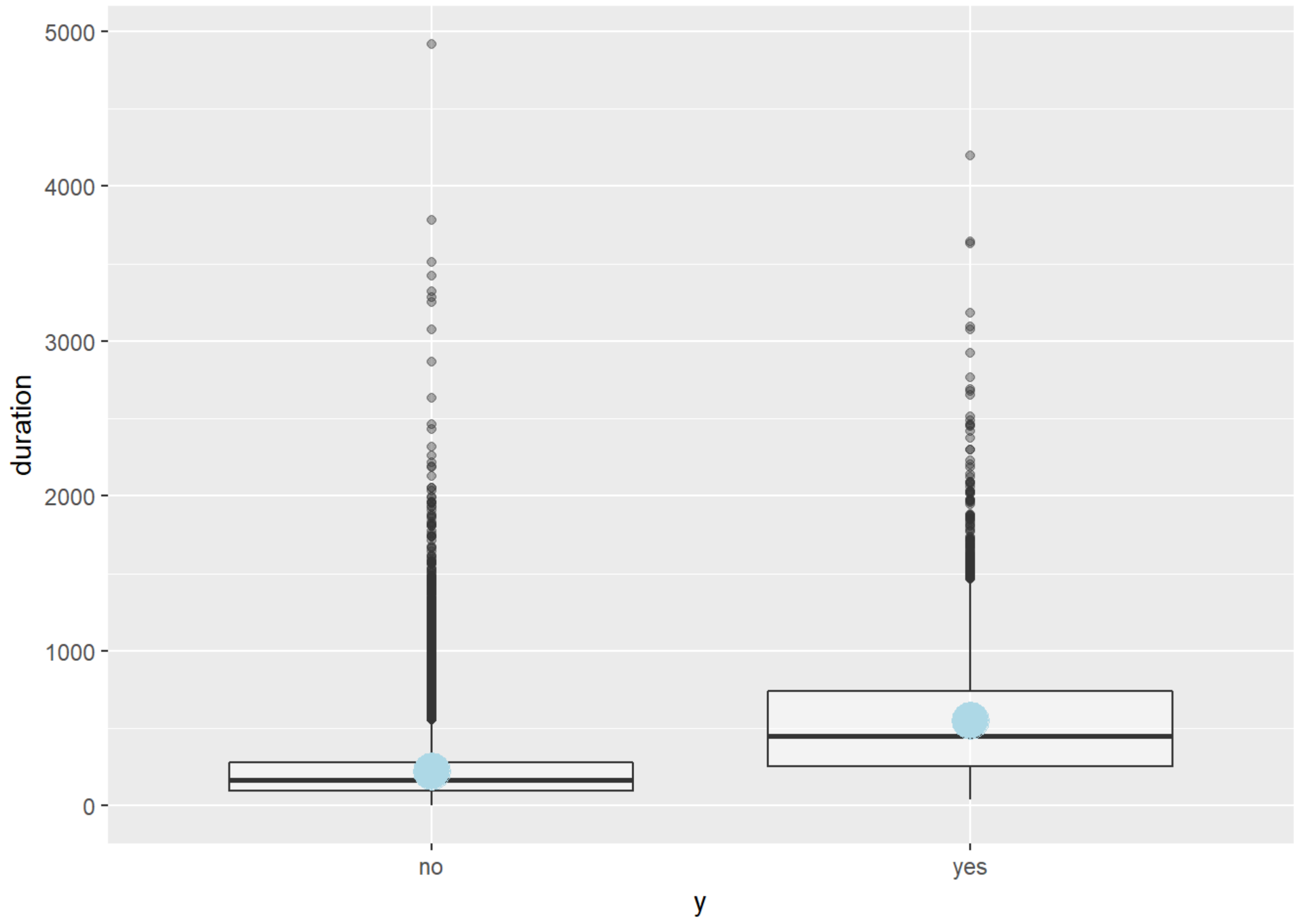
We notice that the model performance has a significantly decrease while removing the duration variable, which indirectly proves its importance. Based on our analysis, the higher duration lasts, the better model's performance they have. We therefore come up with one thought, along with the consideration of high importance of previous, we believe that the clients who have larger amount of contacts will have higher opportunities to the subscriptions and finally are in a favor to "yes" decision. Basically, through the analysis, in order to have higher rate of subscriptions, **we can make the contact frequently and interactively to make sure our clients can still hear from us and remind them our great services.**

Additional Business Solution

Boxplot

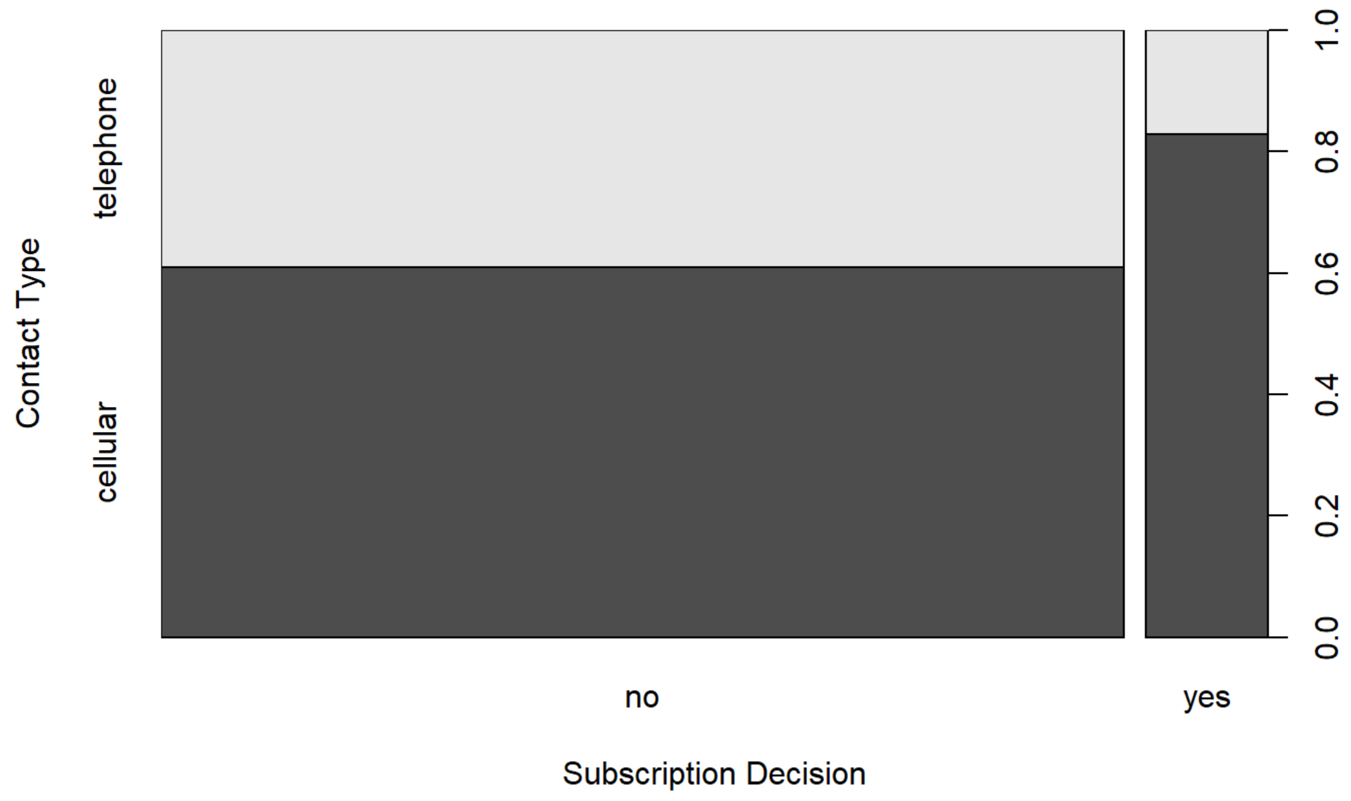
$- - - - - - - Duration\ v.s.\ y - - - - - - -$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



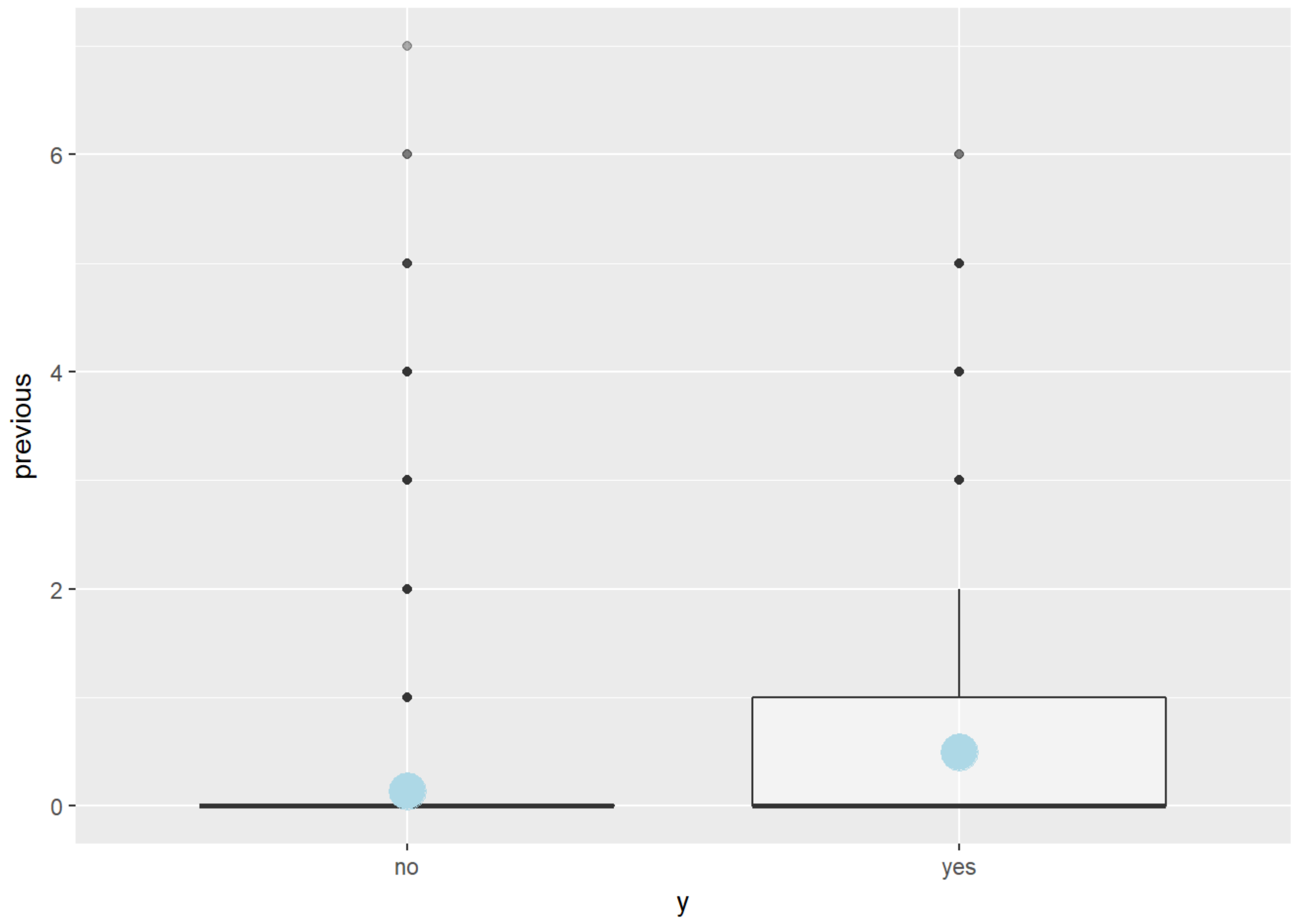
$- - - - - - - ContactType\ v.s.\ y - - - - - - -$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



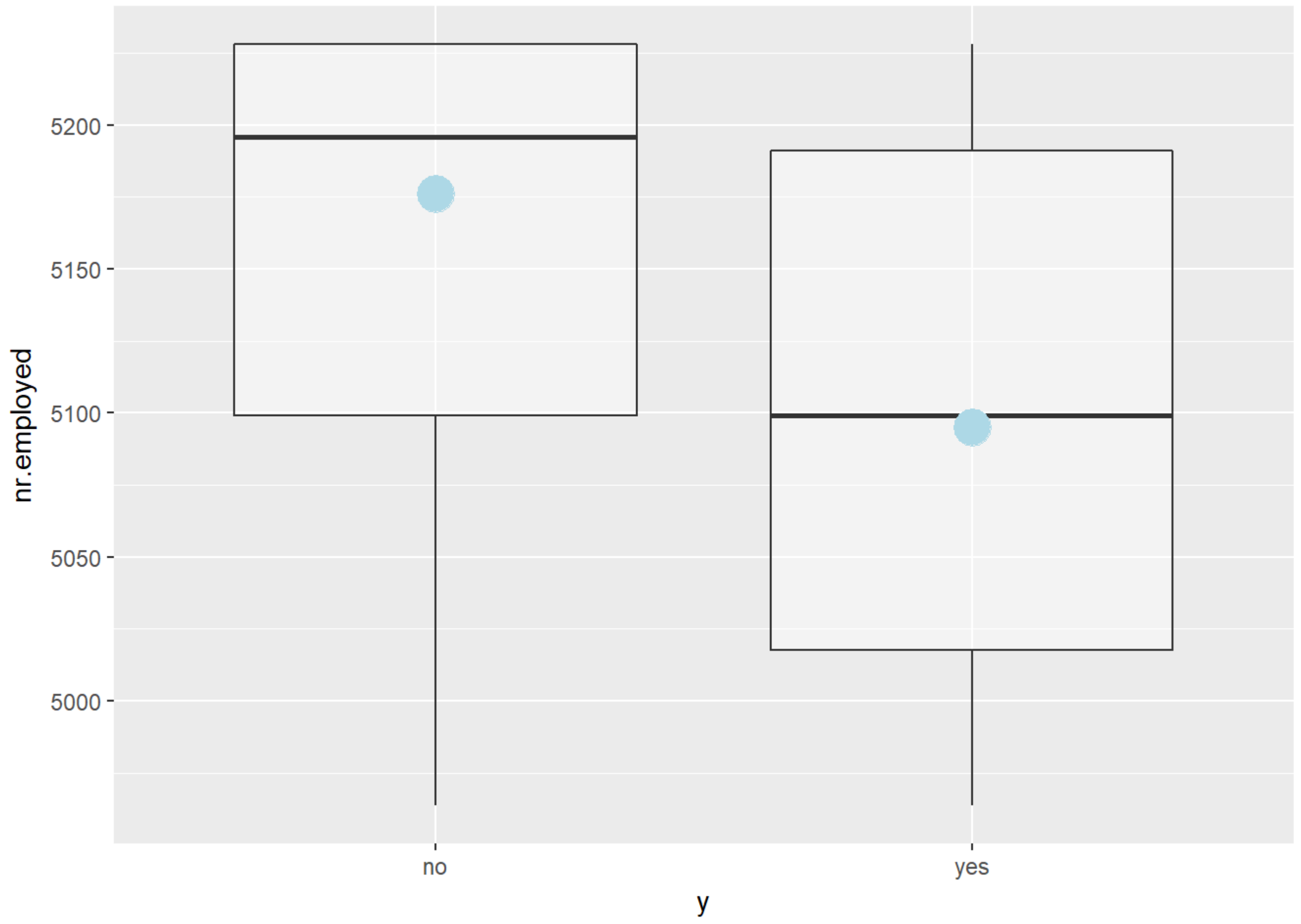
$- - - - - - - PreviousNumberofContact\ v.s.\ y - - - - - - -$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



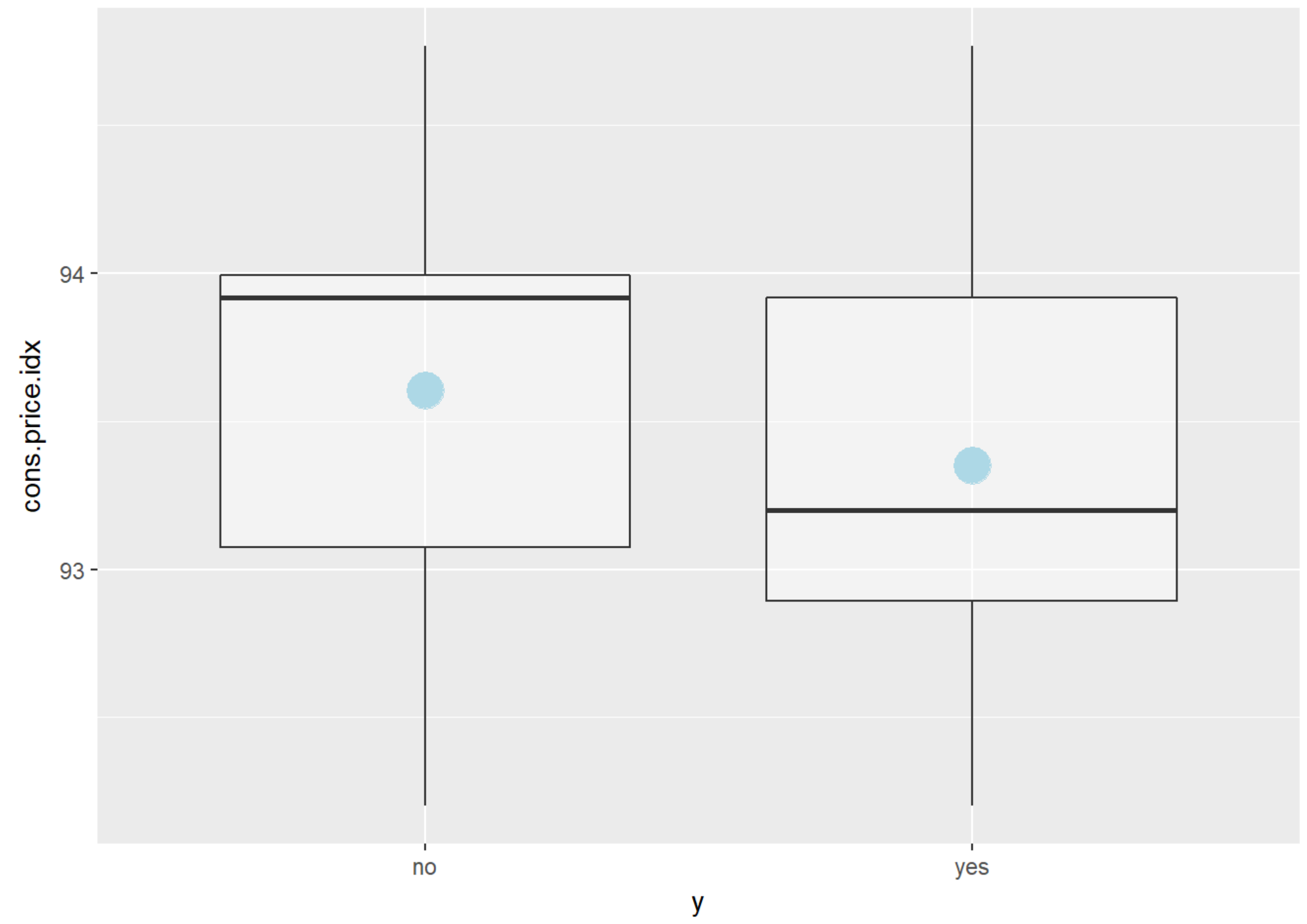
$$- - - - - - - Number\ of\ Employees.\ \ v.s.\ \ y - - - - - - -$$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



$-------\ Consumer\ Price\ Index\ v.s.\ \ y -------$

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js



Interpretation

The boxplot constructed above are the indicators of the confidence and average level for different decision of y. Since the number of people made "no" decision are much larger than those who say "yes", so the imbalanced data might have bias. But according to the training data, there is no way to gather more "yes" data to make the data balanced, so we can only take these visualizations as non-priority reference, and this is also the reason I put them at end of the report.

Take the importance and IV(information value) into account, we choose duration, contact, previous, nr.employed, and cons.price.idx as template. Regarding the duration, people say "yes" has a higher average value than the answer of "no", which states that clients who spend more time on the phone contact will more likely to subscripe the term deposit, same situation as previous (more previous contacts, higher chances to agree term deposit). If time allows, we would definitely build a statistical model to test the significance.The contact type is a special one since it has telephone and cell phone, again, similar results with duration, "yes" consumers use more on the cell phone while the "no" consumers take the telephone call.

The last two box comparison, nr.employed and cons.price.idx have the similar pattern, which is that "yes" prefers to have lower value, particularly, nr.employed with "no" has average of ~5175, whereas "yes" is around 5100. That is, clients who have less number of employee and lower consumer price index will have opportunity to subscripe.

In general, by analyzing the boxplot, we can see a big picture from a real world perspective. In reality, for instance, if a person is interested in the term deposit, the duration of call and number of previous contacts will be higher than those who are not interested since he/she intend to hear more information. Likely, samller

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

company does not have huge cash flow needs, deposit in the bank would at least get interests. When CPI increases, the purchasing power is getting down, the CPI simultaneously turn out to be lower than the term deposit interests (positive interest rate). However, interest cut will take place soon so people are taking money out of bank, this is a possible reason why the average between these two factors is not big.

Again, because of the imbalanced data and limited competition time, any analysis accomplished in this section will not have a guarantee statistical conclusion.

Conclusion

For this data competition, we first discovered that the response variable is imbalance, and there is 10 categorical variables and 10 numeric variables with no missing value. After that we converted the data type for creating correlation plot and variable importance plot. Based on the correlation plot, we found that nr.employed, emp.var.rate, euribor3m are highly correlated with each other. The variable importance graph shows us that variable “duration” is most important, and following by “nr.employed”, “cons.conf.idx” “age” right after. With those information, we exclude columns that are hightly correlated, and test train split our data for model fitting.

As for model selection, we decided to choose model based on precision for prediction yes. After building several model, we finally decided that the best model for prediction is tuned GBM model. It has 0.446 precision for prediction yes in test set, with auc value equals to 0.733

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js