Bank Term deposit Subscription Prediction

Juilee Salunkhe

5/2/2020

Introduction:
The objective of the assignment is to predict whether or not the client will subscribe a bank term deposit.It is a binary classification problem of a Bank Marketing domain.Multiple machine learning algorithms,including logistic regression,Gradient Boosting,Naive Bayes,K-nearest neighbors and Random Forest to build a subscription checking model will be used. Dataset Link: https://www.mldata.io/dataset-details/bank_marketing/

Loading packages

```
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(caretEnsemble))
suppressPackageStartupMessages(library(rpart))
suppressPackageStartupMessages(library(randomForest))
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(klaR))
suppressPackageStartupMessages(library(naivebayes))
suppressPackageStartupMessages(library(doParallel))
suppressPackageStartupMessages(library(Amelia))
suppressPackageStartupMessages(library(pROC))
suppressPackageStartupMessages(library(gridExtra))
suppressPackageStartupMessages(library(grid))
suppressPackageStartupMessages(library(resample))
suppressPackageStartupMessages(library(ggpubr))
suppressWarnings(suppressPackageStartupMessages(library(raster)))
```

Loading training dataset.

```
d_train<- read.csv("/Users/juilee81/Desktop/bank_marketing_dataset.csv",header=TRUE)
```

Checking for missing values column-wise and visualize the data:

```
colSums(is.na(d_train))
```

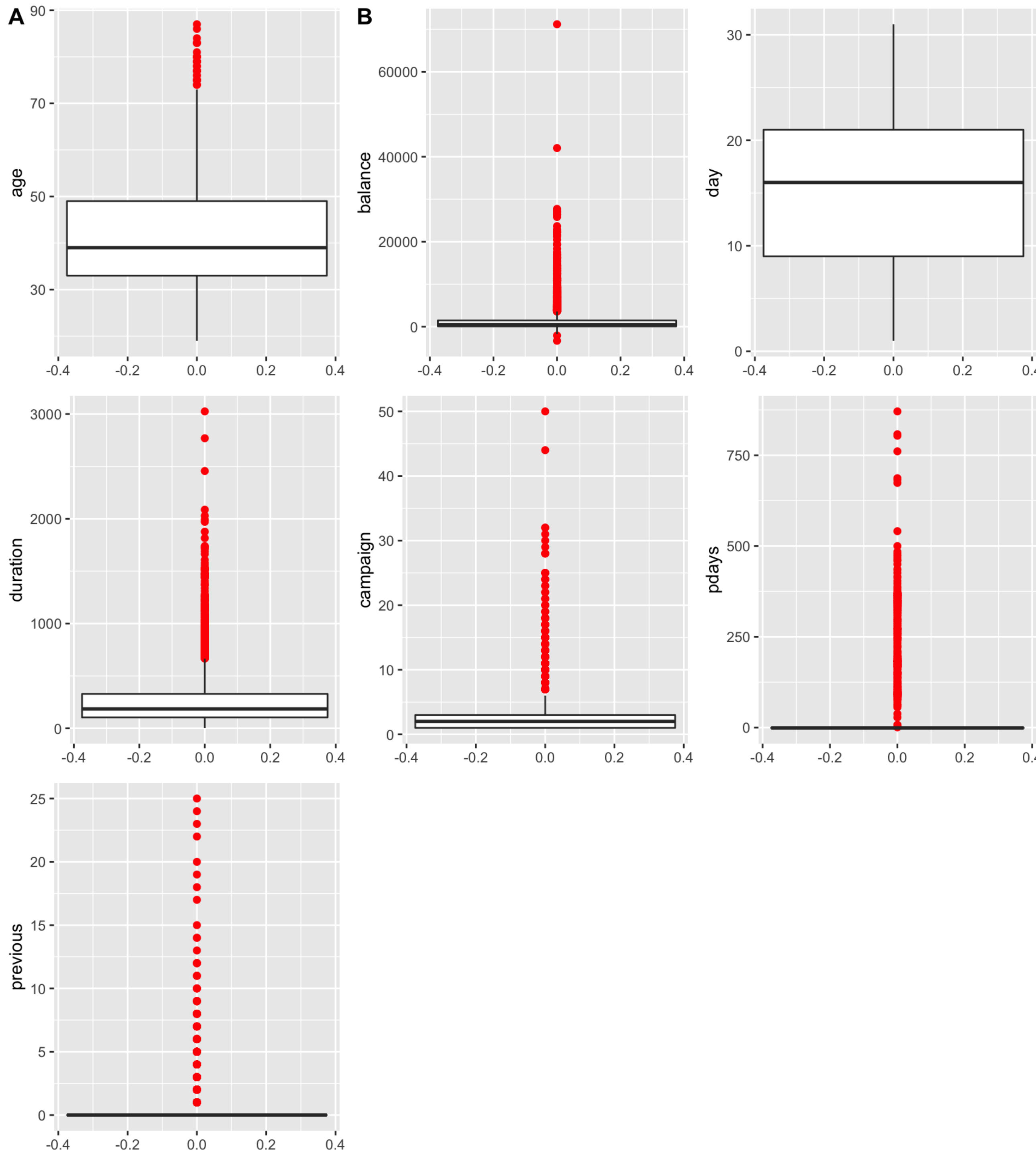
```
##       age       job   marital education   default   balance   housing
##         0         0         0         0         0         0         0
##      loan   contact       day     month  duration  campaign     pdays
##         0         0         0         0         0         0         0
##  previous  poutcome         y
##         0         0         0
```

```
summary(d_train)
```

```
##       age                  job          marital         education
##  Min.   :19.00   management :969   divorced: 528   primary  : 678
##  1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306
##  Median :39.00   technician :768   single  :1196   tertiary :1350
##  Mean   :41.17   admin.     :478                   unknown  : 187
##  3rd Qu.:49.00   services   :417
##  Max.   :87.00   retired    :230
##                  (Other)    :713
##  default       balance      housing     loan           contact
##  no :4445   Min.   :-3313   no :1962   no :3830   cellular :2896
##  yes:  76   1st Qu.:   69   yes:2559   yes: 691   telephone: 301
##             Median :  444                         unknown  :1324
##             Mean   : 1423
##             3rd Qu.: 1480
##             Max.   :71188
##
##       day             month         duration       campaign
##  Min.   : 1.00   may    :1398   Min.   :   4   Min.   : 1.000
##  1st Qu.: 9.00   jul    : 706   1st Qu.: 104   1st Qu.: 1.000
##  Median :16.00   aug    : 633   Median : 185   Median : 2.000
##  Mean   :15.92   jun    : 531   Mean   : 264   Mean   : 2.794
##  3rd Qu.:21.00   nov    : 389   3rd Qu.: 329   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 293   Max.   :3025   Max.   :50.000
##                  (Other): 571
##      pdays           previous          poutcome      y
##  Min.   : -1.00   Min.   : 0.0000   failure: 490   no :4000
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 197   yes: 521
##  Median : -1.00   Median : 0.0000   success: 129
##  Mean   : 39.77   Mean   : 0.5426   unknown:3705
##  3rd Qu.: -1.00   3rd Qu.: 0.0000
##  Max.   :871.00   Max.   :25.0000
##
```

Outlier Detection

```r
bxp_Age<- ggplot(d_train, aes(y=age))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_balance<- ggplot(d_train, aes(y=balance))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_day<- ggplot(d_train, aes(y=day))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_duration<- ggplot(d_train, aes(y=duration))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_campaign<- ggplot(d_train, aes(y=campaign))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)
bxp_pdays <- ggplot(d_train, aes(y=pdays))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)
bxp_previous <- ggplot(d_train, aes(y=previous))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)




figure <- ggarrange(bxp_Age,bxp_balance,bxp_day,bxp_duration,bxp_campaign,bxp_pdays,bxp_previous,
                    labels = c("A", "B"),
                    ncol = 3, nrow = 3)
figure
```



Using Inter-Quartile Range method to remove outliers.

```
out<- function(q1,q3) {
  l<- q1-1.5*(q3-q1)
  u<- q3+1.5*(q3-q1)
  print(paste(l,u))
}
```

```
#Clamping using inter-quartile range
out(33,49)
```

```
## [1] "9 73"
```

```
d_train$age<-clamp(d_train$age, lower=9, upper=73)
out(69,1480)
```

```
## [1] "-2047.5 3596.5"
```

```
d_train$balance<-clamp(d_train$balance, lower=-2047.5, upper=3596.5)
out(104,329)
```

```
## [1] "-233.5 666.5"
```

```
d_train$duration<-clamp(d_train$duration, lower=-233.5, upper=666.5)
out(1,3)
```

```
## [1] "-2 6"
```

```
d_train$campaign<-clamp(d_train$campaign, lower=-2, upper=6)
out(-1,-1)
```

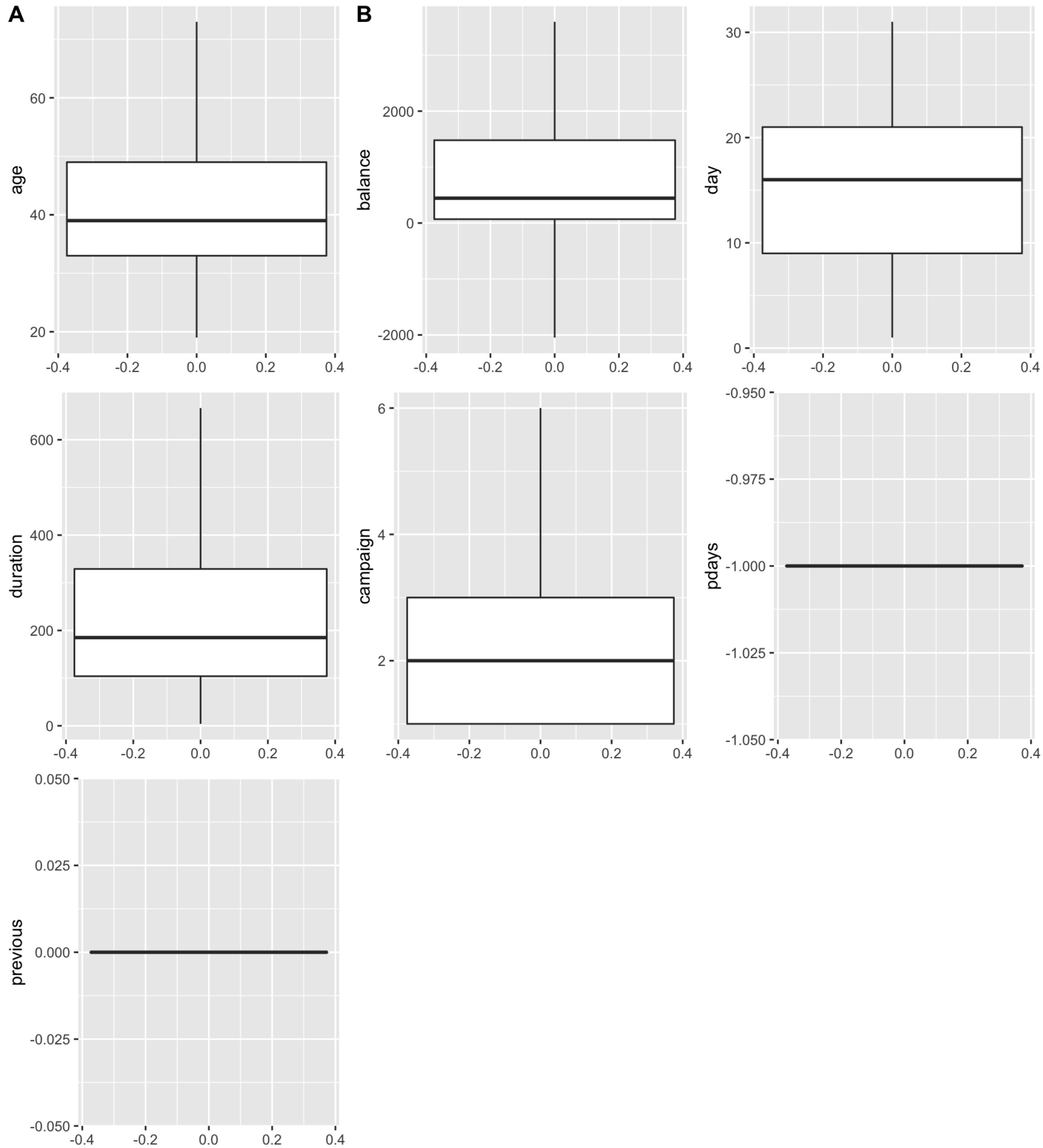
```
## [1] "-1 -1"
```

```
d_train$pdays<-clamp(d_train$pdays, lower=-1, upper=-1)
out(0,0)
```

```
## [1] "0 0"
```

```
d_train$previous<-clamp(d_train$previous, lower=0, upper=0)
```

```
bxp_Age<- ggplot(d_train, aes(y=age))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_balance<- ggplot(d_train, aes(y=balance))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_day<- ggplot(d_train, aes(y=day))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_duration<- ggplot(d_train, aes(y=duration))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)

bxp_campaign<- ggplot(d_train, aes(y=campaign))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)
bxp_pdays <- ggplot(d_train, aes(y=pdays))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)
bxp_previous <- ggplot(d_train, aes(y=previous))+
              geom_boxplot(outlier.colour="red",
              outlier.shape=16,
              outlier.size=2, notch=FALSE)




figure <- ggarrange(bxp_Age,bxp_balance,bxp_day,bxp_duration,bxp_campaign,bxp_pdays,bxp_previous,
                    labels = c("A", "B"),
                    ncol = 3, nrow = 3)
figure
```

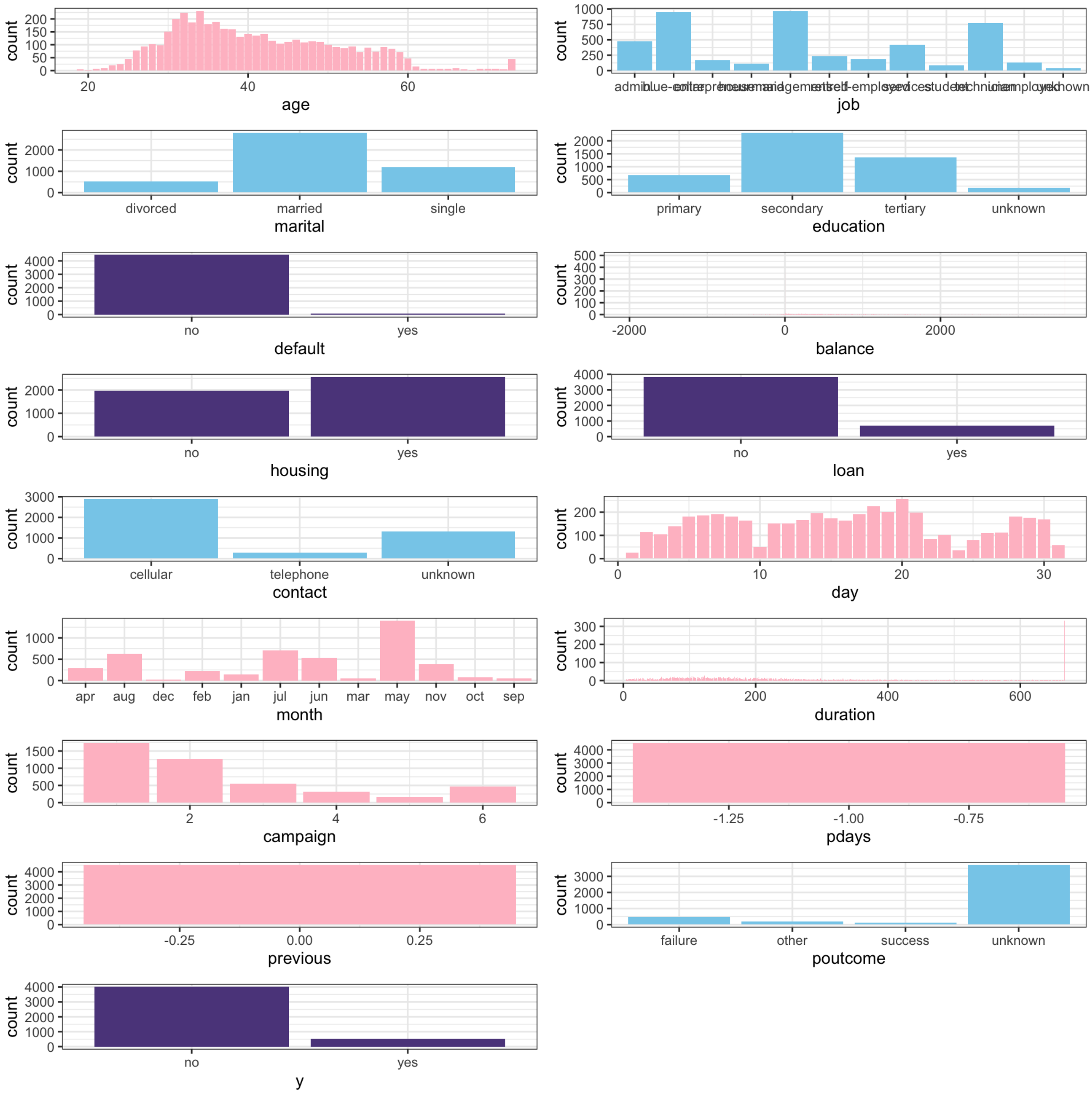


Outliers are removed.

Exploratory Data Analysis:

Univariate Analysis

```r
p1<-ggplot(d_train, aes(x=age)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p2<-ggplot(d_train, aes(x=job)) +
  geom_bar(stat="count",position="dodge",fill="skyblue")+theme_bw()
p3<-ggplot(d_train, aes(x=marital)) +
  geom_bar(stat="count",position="dodge",fill="skyblue")+theme_bw()
p4<-ggplot(d_train, aes(x=education)) +
  geom_bar(stat="count",position="dodge",fill="skyblue")+theme_bw()
p5<-ggplot(d_train, aes(x=default)) +
  geom_bar(stat="count",position="dodge",fill="mediumpurple4")+theme_bw()
p6<-ggplot(d_train, aes(x=balance)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p7<-ggplot(d_train, aes(x=housing)) +
  geom_bar(stat="count",position="dodge",fill="mediumpurple4")+theme_bw()
p8<-ggplot(d_train, aes(x=loan)) +
  geom_bar(stat="count",position="dodge",fill="mediumpurple4")+theme_bw()
p9<-ggplot(d_train, aes(x=contact)) +
  geom_bar(stat="count",position="dodge",fill="skyblue")+theme_bw()
p10<-ggplot(d_train, aes(x=day)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p11<-ggplot(d_train, aes(x=month)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p12<-ggplot(d_train, aes(x=duration)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p13<-ggplot(d_train, aes(x=campaign)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p14<-ggplot(d_train, aes(x=pdays)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p15<-ggplot(d_train, aes(x=previous)) +
  geom_bar(stat="count",position="dodge",fill="pink")+theme_bw()
p16<-ggplot(d_train, aes(x=poutcome)) +
  geom_bar(stat="count",position="dodge",fill="skyblue")+theme_bw()
p17<-ggplot(d_train, aes(x=y)) +
  geom_bar(stat="count",position="dodge",fill="mediumpurple4")+theme_bw()



grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,ncol=2, nrow= 9)
```



Variables in skyblue are categorical, in purple are binary whereas in pink are continous. Binary variable y is the target variable whether or not the client has subscribed a term deposit.

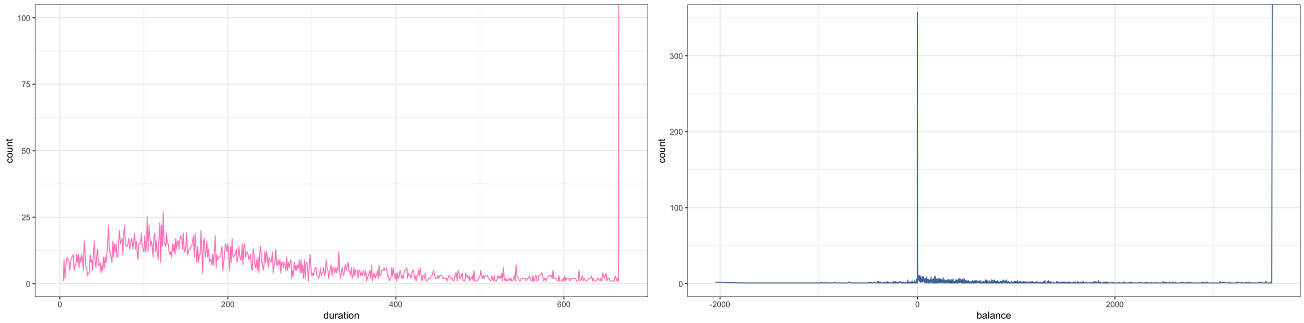
About Customers:

Greater number of customers have a blue-collar job,are from management and are technicians.Majority of customers are not defaulters(has credit in default), have housing loans and are married. Very few have personal loans. Majority of the clients fall in the age group of 26-58, education level upto secondary, tertiary and conmmunication type registered is cellular.

About Bank:

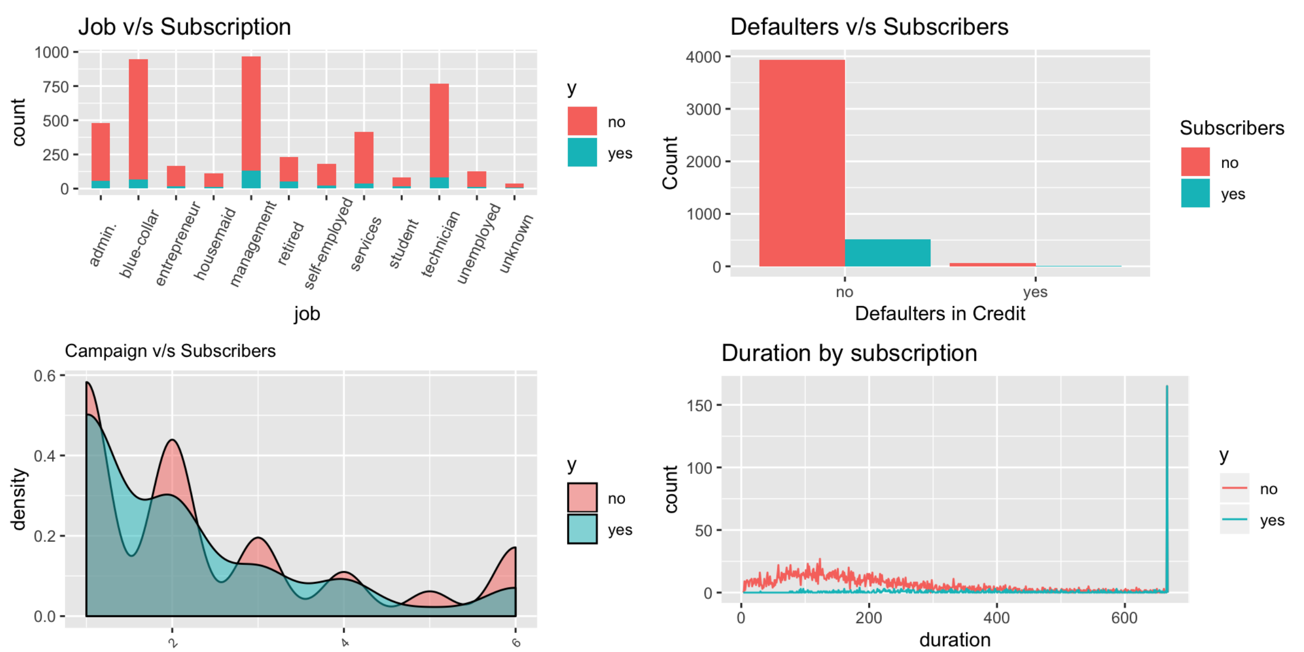
Campaign tells us how many times the client was contacted. Majorly the number of contacts made to the customers were once or twice and many were contacted in the month of May (highest).The outcome of the previous marketing campaign is unknown for most of the cases. Let us see duration and balance features more clearly.

```
q1<-ggplot(d_train, aes(x=duration)) +
  geom_line(stat="count",position=position_dodge(width = 0.9), alpha = 0.8,color="hotpink")+theme_bw()+coord
_cartesian(ylim=c(0,100))

q2<- ggplot(d_train, aes(x=balance)) +
  geom_line(stat="count",position=position_dodge(width = 0.9), alpha = 0.8,color="dodgerblue4")+theme_bw()+c
oord_cartesian(ylim=c(0,350))
grid.arrange(q1,q2,ncol=2, nrow= 1)
```



Majority of the contacts had a contact duration of 100-130 seconds and Average yearly balance is mostly in the range of 0-500 Euros.

```
#Bivariate Analysis
r1 <- ggplot(d_train, aes(job)) +
  geom_bar(aes(fill=y), width = 0.5) +
  theme(axis.text.x = element_text(angle=65, vjust=0.6)) +
  labs(title="Job v/s Subscription")

r2 <- ggplot(d_train, aes(x = default, fill = y)) +
                    geom_bar(position = "dodge") +
                    ggtitle("Defaulters v/s Subscribers") +
                    labs(x = "Defaulters in Credit", y = "Count", fill = "Subscribers")

r3 <- ggplot(d_train,aes(x = campaign, fill = y)) +
               geom_density(alpha = 0.5) +
               ggtitle("Campaign v/s Subscribers") +
               theme(plot.title = element_text(size =10),
                     axis.text.x = element_text(size =7,angle = 45,hjust=1),
                     axis.title.x=element_blank())
r4<- ggplot(d_train, aes(duration, colour = y)) +
geom_freqpoly(binwidth = 1) + labs(title="Duration by subscription")


grid.arrange(r1,r2,r3,r4,ncol=2,nrow= 2)
```



Clients belonging to the management field have subscribed more as compared to others.More number of clients subscribe if they are contacted once or twice. When the duration of the contact is less than 200 secs there are high chances that the client do not subscribe.

Creating dummy variables of all the categorical variables

```
library(fastDummies)
d_dummies <- fastDummies::dummy_cols(d_train,select_columns=c('job','marital','education','default','housing
','loan','contact','month','poutcome'))
d_dummies <- d_dummies[,c(-2,-3,-4,-5,-7,-8,-9,-11,-16)]
```

```
d_dummies$y <-factor(d_dummies$y,labels = c("False", "True"))
```

Split data into training and test data sets

```
indxTrain <- createDataPartition(y =d_dummies$y,p = 0.75,list = FALSE)
training <- d_train[indxTrain,]
testing <- d_train[-indxTrain,]
testing$y <- factor(testing$y,labels = c("False", "True"))
table(training$y)
```

```
##
##   no  yes
## 3000  391
```

Since the data is highly imbalanced we will use package ROSE to over and under sample the data.The ROSE package provides functions to deal with binary classification problems in the presence of imbalanced classes. Artificial balanced samples are generated according to a smoothed bootstrap approach and allow for aiding both the phases of estimation and accuracy evaluation of a binary classifier in the presence of a rare class.Here N is the desired sample size of the dataset.

```
#install.packages("ROSE")
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
data_balanced_both <- ovun.sample(y ~ ., data = training, method = "both", p=0.5,N=3391, seed = 1)$data
table(data_balanced_both$y)
```

```
##
##   no  yes
## 1751 1640
```

```
data_balanced_both$y <-factor(data_balanced_both$y,labels = c("False", "True"))
```

Trying Linear Algorithms:
Naive Bayes and Logistic Regression Base models
Naive Bayes

```
set.seed(124)
Naive_Bayes_Model <- naiveBayes(y ~.,data=data_balanced_both)
```

```
predict_nb <- predict(Naive_Bayes_Model, newdata = testing)
confusionMatrix(predict_nb,testing$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   819   30
##      True    181  100
##
##                Accuracy : 0.8133
##                  95% CI : (0.7893, 0.8356)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3908
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8190
##             Specificity : 0.7692
##          Pos Pred Value : 0.9647
##          Neg Pred Value : 0.3559
##              Prevalence : 0.8850
##          Detection Rate : 0.7248
##    Detection Prevalence : 0.7513
##       Balanced Accuracy : 0.7941
##
##        'Positive' Class : False
##
```

Logistic Regression

```
set.seed(123)
model_lr <- train(y ~ ., data = data_balanced_both,
                   method = "glm", family = "binomial")
```

```
suppressWarnings(predict_lr <- predict(model_lr, newdata = testing))
suppressWarnings(confusionMatrix(predict_lr, testing$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   831   31
##      True    169   99
##
##                Accuracy : 0.823
##                  95% CI : (0.7995, 0.8448)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4054
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8310
##             Specificity : 0.7615
##          Pos Pred Value : 0.9640
##          Neg Pred Value : 0.3694
##              Prevalence : 0.8850
##          Detection Rate : 0.7354
##    Detection Prevalence : 0.7628
##       Balanced Accuracy : 0.7963
##
##        'Positive' Class : False
##
```

Trying Nonlinear Algorithm:

K-Nearest Neighbor Base Model

```r
set.seed(124)
model_knn <- train(y ~ ., data = data_balanced_both, method = "knn")

predict_knn <- predict(model_knn, newdata = testing)
confusionMatrix(predict_knn, testing$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   657   40
##      True    343   90
##
##                Accuracy : 0.6611
##                  95% CI : (0.6326, 0.6887)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1734
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6570
##             Specificity : 0.6923
##          Pos Pred Value : 0.9426
##          Neg Pred Value : 0.2079
##              Prevalence : 0.8850
##          Detection Rate : 0.5814
##    Detection Prevalence : 0.6168
##       Balanced Accuracy : 0.6747
##
##        'Positive' Class : False
##
```

Trying Ensemble Algorithm:

Random Forest and Gradient Boosting

Building a model : Random forest

```r
set.seed(124)
model_rf <- train(y ~ ., data =data_balanced_both, method = "rf")
model_rf
```

```
## Random Forest
##
## 3391 samples
##   16 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3391, 3391, 3391, 3391, 3391, 3391, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8617461  0.7229768
##   22    0.9472866  0.8946675
##   42    0.9428528  0.8858202
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 22.
```

```r
predict_rf <- predict(model_rf, newdata = testing)
confusionMatrix(predict_rf, testing$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   905   49
##      True     95   81
##
##                Accuracy : 0.8726
##                  95% CI : (0.8517, 0.8915)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 0.9103164
##
##                   Kappa : 0.4576
##
##  Mcnemar's Test P-Value : 0.0001768
##
##             Sensitivity : 0.9050
##             Specificity : 0.6231
##          Pos Pred Value : 0.9486
##          Neg Pred Value : 0.4602
##              Prevalence : 0.8850
##          Detection Rate : 0.8009
##    Detection Prevalence : 0.8442
##       Balanced Accuracy : 0.7640
##
##        'Positive' Class : False
##
```

Hyperparameter Tuning for Random forest:

```
control <- trainControl(method="cv", number=3, repeats=1)
mtry <- c(1,22,34,10)
tunegrid <- expand.grid(.mtry=mtry)
```

Each axis of the grid is an algorithm parameter, and points in the grid are specific combinations of parameters. Because we are only tuning one parameter, the grid search is a linear search through a vector of candidate values.mtry parameter is available in caret for tuning.
1. **mtry:** Number of variables randomly sampled as candidates at each split.

```
set.seed(124)
model_rf_tune <- train(y~., data=data_balanced_both, method="rf",metric='Accuracy',tuneGrid=tunegrid, trCont
rol=control)
model_rf_tune
```

```
## Random Forest
##
## 3391 samples
##   16 predictor
##    2 classes: 'False', 'True'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 2260, 2260, 2262
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7272175  0.4479492
##   10    0.9507497  0.9015883
##   22    0.9501613  0.9004217
##   34    0.9492755  0.8986528
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.
```

```
predict_rf_tune <- predict(model_rf_tune, newdata = testing)
a<-confusionMatrix(predict_rf_tune, testing$y)
a
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   910   55
##      True     90   75
##
##                Accuracy : 0.8717
##                  95% CI : (0.8508, 0.8906)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 0.92407
##
##                   Kappa : 0.4359
##
##  Mcnemar's Test P-Value : 0.00475
##
##             Sensitivity : 0.9100
##             Specificity : 0.5769
##          Pos Pred Value : 0.9430
##          Neg Pred Value : 0.4545
##              Prevalence : 0.8850
##          Detection Rate : 0.8053
##    Detection Prevalence : 0.8540
##       Balanced Accuracy : 0.7435
##
##        'Positive' Class : False
##
```

Building a model : Gradient Boosting

```
set.seed(124)
unwantedoutput <- suppressWarnings(capture.output(model_gbm <- train(y ~ ., data = data_balanced_both, metho
d = "gbm")))
```

```
predict_gbm <- predict(model_gbm, newdata = testing)
confusionMatrix(predict_gbm, testing$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False   845   26
##      True    155  104
##
##                Accuracy : 0.8398
##                  95% CI : (0.8171, 0.8607)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4505
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8450
##             Specificity : 0.8000
##          Pos Pred Value : 0.9701
##          Neg Pred Value : 0.4015
##              Prevalence : 0.8850
##          Detection Rate : 0.7478
##    Detection Prevalence : 0.7708
##       Balanced Accuracy : 0.8225
##
##        'Positive' Class : False
##
```

Hyperparameter Tuning for Gradient Boosting:

```
#install.packages("doParallel")
control <- trainControl(method="repeatedcv", number=3, repeats=3)
tgrid<- expand.grid(n.trees =c(1080:1100),
                    interaction.depth=c(1:3),
                    shrinkage=c(0.001,0.2,0.3),
                    n.minobsinnode=15)
```

1.**n.trees:** The total number of trees in the sequence or ensemble.Since they can easily overfit if there are many number of trees,we must find the optimal number of trees that minimize the loss function of interest with cross validation. 2.**shrinkage:** Determines the contribution of each tree on the final outcome and controls how quickly the algorithm proceeds down the gradient descent.Generally, the smaller this value, the more accurate the model can be but also will require more trees in the sequence. 3.**interaction.depth:** Controls the depth of the individual trees.Higher depth trees allow the algorithm to capture unique interactions but also increase the risk of over-fitting. 4.**n.minobsinnode:** Controls the complexity of each tree.Higher values help prevent a model from learning relationships which might be highly specific to the particular sample selected for a tree (overfitting) but smaller values can help with imbalanced target classes in classification problems.

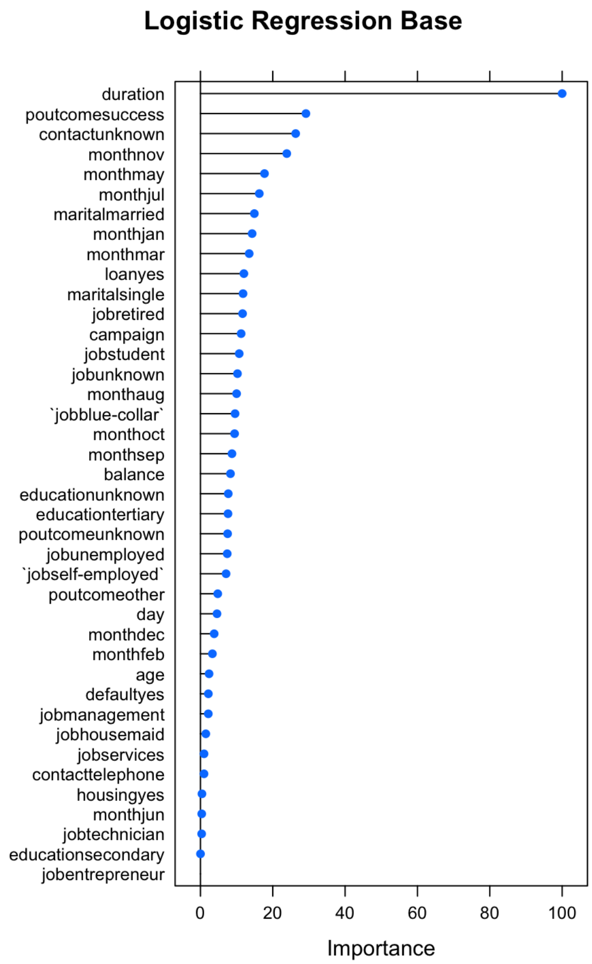
```
set.seed(124) #for reproducability
unwantedoutput <- suppressWarnings(capture.output(model_gbm_tune <- train(y ~ ., data = data_balanced_both,
method = "gbm",metric="Accuracy",tuneGrid=tgrid,trControl=control)))
```

```
predict_gbm_tune <- predict(model_gbm_tune, newdata = testing)
b<-confusionMatrix(predict_gbm_tune,testing$y)
b
```

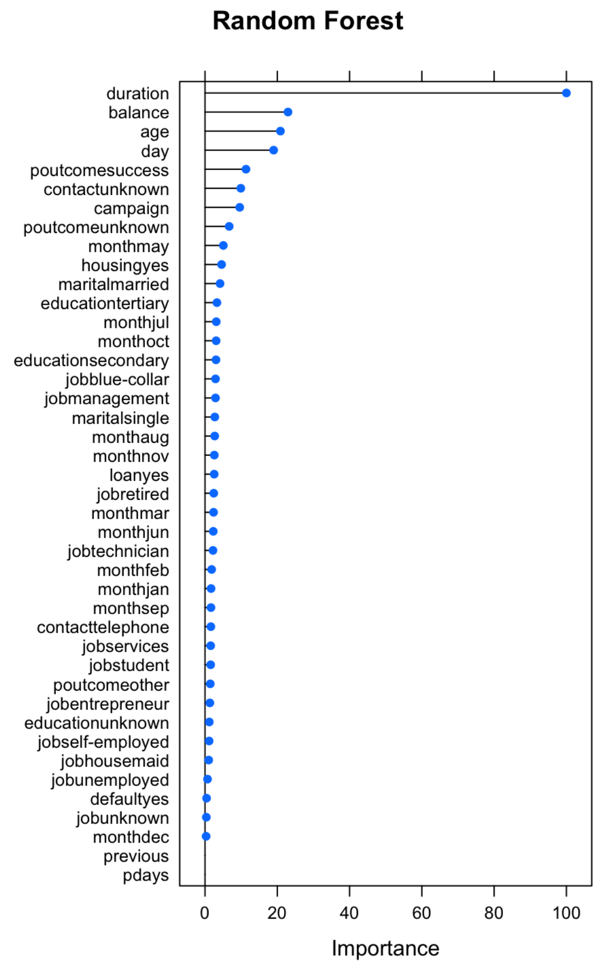
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction False True
##      False   888   54
##      True    112   76
##
##                Accuracy : 0.8531
##                  95% CI : (0.8311, 0.8732)
##     No Information Rate : 0.885
##     P-Value [Acc > NIR] : 0.9995
##
##                   Kappa : 0.3958
##
##  Mcnemar's Test P-Value : 9.686e-06
##
##             Sensitivity : 0.8880
##             Specificity : 0.5846
##          Pos Pred Value : 0.9427
##          Neg Pred Value : 0.4043
##              Prevalence : 0.8850
##          Detection Rate : 0.7858
##    Detection Prevalence : 0.8336
##       Balanced Accuracy : 0.7363
##
##        'Positive' Class : False
##
```

Feature Importance of three models

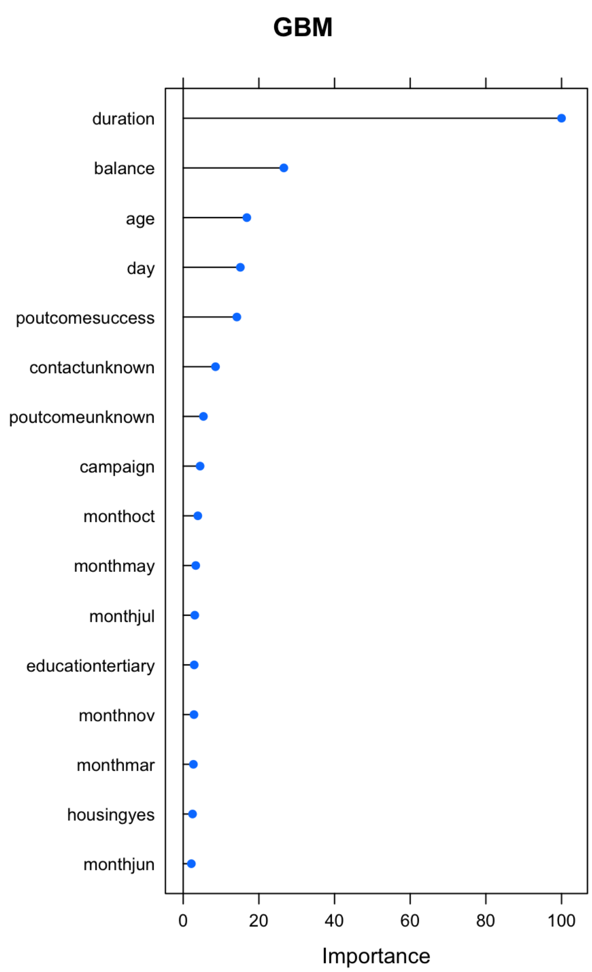
```
library(gbm)
grid.arrange(plot(varImp(model_lr),main="Logistic Regression Base"),
             plot(varImp(model_rf_tune),main="Random Forest"),
             plot(varImp(model_gbm_tune),
                  top=16 ,main="GBM"),nrow = 2, ncol = 2)
```



Logistic Regression Base
duration
poutcomesuccess
contactunknown
monthnov
monthmay
monthjul
maritalmarried
monthjan
monthmar
loanyes
maritalsingle
jobretired
campaign
jobstudent
jobunknown
monthaug
`jobblue-collar`
monthoct
monthsep
balance
educationunknown
educationtertiary
poutcomeunknown
jobunemployed
`jobself-employed`
poutcomeother
day
monthdec
monthfeb
age
defaultyes
jobmanagement
jobhousemaid
jobservices
contacttelephone
housingyes
monthjun
jobtechnician
educationsecondary
jobentrepreneur
0 20 40 60 80 100
Importance



Random Forest
duration
balance
age
day
poutcomesuccess
contactunknown
campaign
poutcomeunknown
monthmay
housingyes
maritalmarried
educationtertiary
monthjul
monthoct
educationsecondary
jobblue-collar
jobmanagement
maritalsingle
monthaug
monthnov
loanyes
jobretired
monthmar
monthjun
jobtechnician
monthfeb
monthjan
monthsep
contacttelephone
jobservices
jobstudent
poutcomeother
jobentrepreneur
educationunknown
jobself-employed
jobhousemaid
jobunemployed
defaultyes
jobunknown
monthdec
previous
pdays
0 20 40 60 80 100
Importance



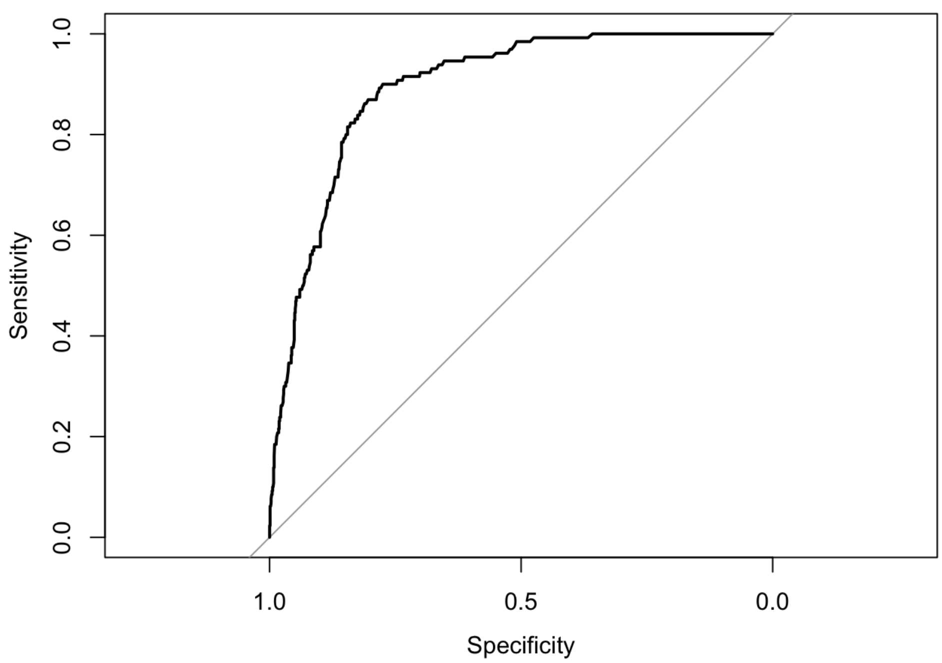
GBM
duration
balance
age
day
poutcomesuccess
contactunknown
poutcomeunknown
campaign
monthoct
monthmay
monthjul
educationtertiary
monthnov
monthmar
housingyes
monthjun
0 20 40 60 80 100
Importance

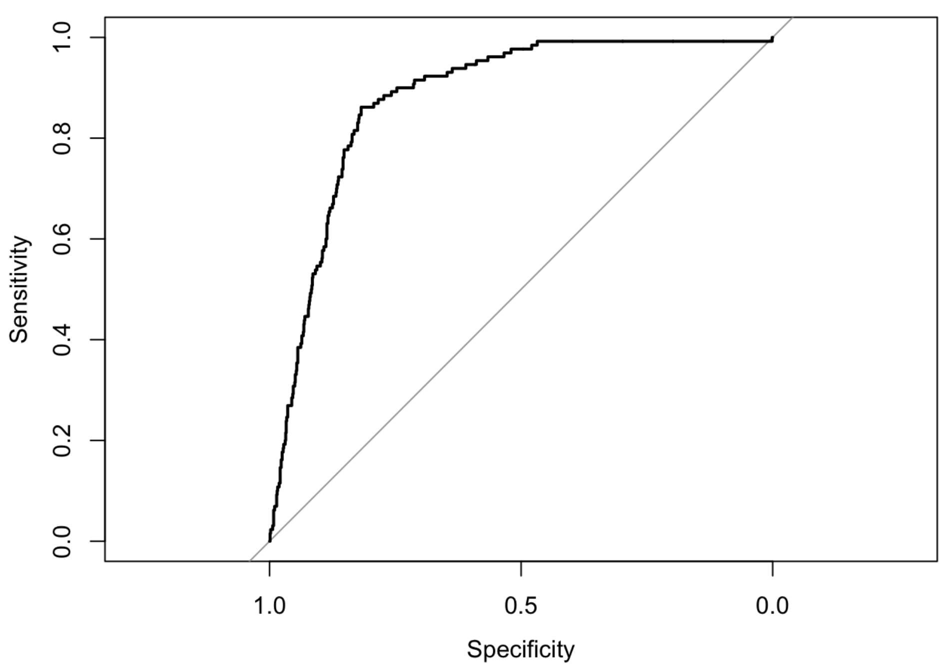
Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature.The most prominent feature came out to be duration followed by age and balance.

ROC AND AUC

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds.

```
roc_func<- function(mo){
probs <- predict(mo,testing,type="prob")
head(probs)
colnames(probs)[1]="False"
colnames(probs)[2]="True"
roc_curve <- roc(testing$y,probs$True)
plot(roc_curve)
}

listee<-c( roc_func(model_rf_tune),roc_func(model_gbm_tune))
```





Area under the curve AUC provides an aggregate measure of performance across all possible classification thresholds. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

```
auc_func<- function(mo){
probs <- predict(mo,testing,type="prob")
head(probs)
colnames(probs)[1]="False"
colnames(probs)[2]="True"
auc_mo<-auc(testing$y,probs$True)
auc_mo
}
recall_rf<- a$byClass['Sensitivity']
recall_gbm<- b$byClass['Sensitivity']
precision_rf<- a$byClass['Pos Pred Value']
precision_gbm <- b$byClass['Pos Pred Value']
auc_rf<-auc_func(model_rf_tune)
auc_gbm<-auc_func(model_gbm_tune)
Accuracy_rf<-a$overall['Accuracy']
Accuracy_gbm<-b$overall['Accuracy']
Model <- c("Random_Forest","Gradient Boosting")
Accuracy<- c(Accuracy_rf,Accuracy_gbm)
AUC <- c(auc_rf,auc_gbm)
Precision<- c(precision_rf,precision_gbm)
Recall<-c(recall_rf,recall_gbm)
auc<-data.frame(Model,Accuracy,AUC,Precision,Recall)
auc<-auc[with(auc, order(-AUC)), ]
auc
```

```
##                Model  Accuracy       AUC Precision Recall
## 1     Random_Forest 0.8716814 0.8945000 0.9430052  0.910
## 2 Gradient Boosting 0.8530973 0.8792538 0.9426752  0.888
```

Conclusion
In this study we have explored the data of Banking subscription dataset and gain insights about the key factors that decide the whether or not the client will subscribe a bank term deposit using multiple machine learning algorithms and data analysis.We have compared various machine learning models in terms of Accuracy and the two best performing models were further tuned. On the basis of performance metrics AUC, Precision and Accuracy found that Random Forest had the highest AUC and Precision.