Business Data Mining – Predicting Net Promotor Score

1. **What is the business problem in this case and how is this business problem converted into an analytics problem?**

Solution:

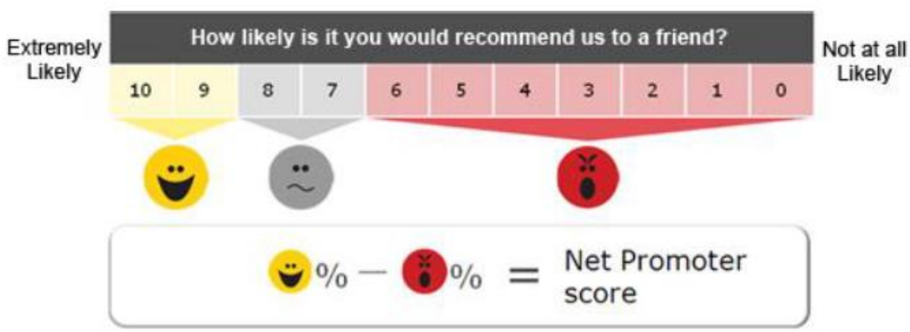
The main problem / objectives of the management at Manipal Health Enterprises is:

- **To enhance customer experience & satisfaction and build customer loyalty through continuous & real time feedback from customers.**

How business problem was converted into Analytics problem:

- Collecting feedback in a structured manner & translate it into meaningful information in real time.
- **Use of Net Promoter Score:** They used the measure called "Net Promoter Score" or NPS. This score is based on a single question: "How likely are you to recommend this product/service to your friend/colleagues". The customers respond on a scale of 0 to 10. Loyal customers are likely to provide a score of 9 or 10, passive customers a score of 7 or 8, while people who score 6 or less are detractors. Subtracting the percentage of detractors from percentage of promoters yield a figure called Net Promoter Score.

NPS question – On a scale of 0 to 10, how likely is it that you would recommend our hospital to a friend or family member?



Source: http://www.netpromotersystem.com/about/measuring-your-net-promoter-score.aspx

- **Patients were asked to give the hospital an overall rating for the services, value for money & accessibility, and the NPS question. Apart from mandatory questions, patients could also provide feedback for specific departments.**
- **The data from this survey could help them deep dive into numerous opportunities like in-depth analysis of department performance, staff or services offered, improving the in-room experience or food & beverages section.**
- **The data from survey could be pivotal as it provides:**
    - **Understanding of the deficiencies in the system and ways of improving them.**
    - **The significant factors influencing the detractors**
    - **The significant factors that improved the Net Promoter Score**
    - **Improvement opportunities within the departments, using NPS**

2) How can we estimate sensitivity and specificity of three-class problem? Provide the formulas.
Solution:
Suppose there are 3 classes, A, B & C. We will use One Vs. All approach to calculate the recall and specificity.

		True Class		
		A	B	C
Predicted Class	A	TP_A	E_{BA}	E_{CA}
	B	E_{AB}	TP_B	E_{CB}
	C	E_{AC}	E_{BC}	TP_C

From the above confusion matrix, for class A:

a. The number of True Positives is TP_A.
b. The number of False Negatives is samples from A that were incorrectly classified as B or C.
   So, $FN_A = E_{AB} + E_{AC}$
c. The number of False Positives is samples that were incorrectly classified as A.
   So, $FP_A = E_{BA} + E_{CA}$
d. The number of True Negatives can be calculated by adding all columns and rows excluding the row and column of class A. So $TN_A = TP_B + E_{CB} + E_{BC} + TP_C$

The sensitivity and specificity are calculated for each class. For example, the sensitivity of A is:

$$\text{Sensitivity or Recall of A} = \frac{TP_A}{TP_A + FN_A} = \frac{TP_A}{TP_A + E_{AB} + E_{AC}}$$

$$\text{Specificity of A} = 1 - \text{False Positive Rate (A)} = \frac{TN_A}{TN_A + FP_A} = \frac{TP_B + E_{CB} + E_{BC} + TP_C}{TP_B + E_{CB} + E_{BC} + TP_C + E_{BA} + E_{CA}}$$

$$\text{Sensitivity or Recall of B} = \frac{TP_B}{TP_B + FN_B} = \frac{TP_B}{TP_B + E_{BA} + E_{BC}}$$

$$\text{Specificity of B} = 1 - \text{False Positive Rate (B)} = \frac{TN_B}{TN_B + FP_B} = \frac{TP_A + E_{CA} + E_{AC} + TP_C}{TP_A + E_{CA} + E_{AC} + TP_C + E_{AB} + E_{CB}}$$

$$\text{Sensitivity or Recall of C} = \frac{TP_C}{TP_C + FN_C} = \frac{TP_C}{TP_C + E_{CA} + E_{CB}}$$

$$\text{Specificity of B} = 1 - \text{False Positive Rate (C)} = \frac{TN_C}{TN_C + FP_C} = \frac{TP_A + E_{AB} + E_{BA} + TP_B}{TP_A + E_{AB} + E_{BA} + TP_B + E_{AC} + E_{BC}}$$

With multi-class classification, we do Micro-averaging or Macro-averaging of these evaluation measures. The formulae are as follows:

The macro average has its name from the fact that it averages over larger groups, namely over the performance for individual classes rather than observations:

$$\text{Macro Recall} = \frac{Recall_A + Recall_B + Recall_C}{3}$$

$$\text{Macro Specificity} = \frac{Specificity_A + Specificity_B + Specificity_C}{3}$$

The micro average has its name from the fact that it pools the performance over the smallest possible unit (i.e. over all samples):

$$\text{Micro Recall} = \frac{TP_A + TP_B + TP_C}{TP_A + TP_B + TP_C + FN_A + FN_B + FN_C}$$

$$\text{Micro Specificity} = \frac{TN_A + TN_B + TN_C}{TN_A + TN_B + TN_C + FP_A + FP_B + FP_C}$$

```
library(MASS)
library(class)
library(dplyr)

library(ggplot2)
library(randomForest)

library(tidyr)
library(LiblineaR)
library(ROCR)

library(DMwR)

library(caret)
library(tidyverse)

library(readxl)



#Importing Multi Class classification data:

MultiTrain <- read_xlsx("MultiTraining.xlsx")
MultiTest <- read_xlsx("MultiTest.xlsx")
```

Pre-processing of Multi-class Data:

```
#Removing Serial number variable SN and HospitalNo2, as they do not contribute to determi
ning the response variable prediction:
MultiTrain <- MultiTrain %>% select(-SN)
MultiTest <- MultiTest %>% select(-SN)

MultiTrain <- MultiTrain %>% select(-HospitalNo2)
MultiTest <- MultiTest %>% select(-HospitalNo2)

MultiTrain <- MultiTrain %>% select(-AdmissionDate)
MultiTest <- MultiTest %>% select(-AdmissionDate)

MultiTrain <- MultiTrain %>% select(-DischargeDate)
MultiTest <- MultiTest %>% select(-DischargeDate)

#Converting categorical variables into factor

MultiTrain$NPS_Status<- as.factor(MultiTrain$NPS_Status)
MultiTest$NPS_Status<- as.factor(MultiTest$NPS_Status)

MultiTrain$MaritalStatus <- as.factor(MultiTrain$MaritalStatus)
MultiTest$MaritalStatus <- as.factor(MultiTest$MaritalStatus)

MultiTrain$Sex <- as.factor(MultiTrain$Sex)
MultiTest$Sex <- as.factor(MultiTest$Sex)

MultiTrain$BedCategory <- as.factor(MultiTrain$BedCategory)
MultiTest$BedCategory <- as.factor(MultiTest$BedCategory)

MultiTrain$Department <- as.factor(MultiTrain$Department)
MultiTest$Department <- as.factor(MultiTest$Department)
```

```r
MultiTrain$InsPayorcategory <- as.factor(MultiTrain$InsPayorcategory)
MultiTest$InsPayorcategory <- as.factor(MultiTest$InsPayorcategory)

MultiTrain$State <- as.factor(MultiTrain$State)
MultiTest$State <- as.factor(MultiTest$State)

MultiTrain$Country <- as.factor(MultiTrain$Country)
MultiTest$Country <- as.factor(MultiTest$Country)

MultiTrain$STATEZONE <- as.factor(MultiTrain$STATEZONE)
MultiTest$STATEZONE <- as.factor(MultiTest$STATEZONE)
```

Importing Binary Data:

```r
binaryTrain <- read_xlsx("BinaryTraining.xlsx")
binaryTest <- read_xlsx("BinaryTest.xlsx")
#binaryTrain <- rbind(binaryTrain, binaryTest)
```

3. What is quasi-complete separation? Which variables in the Manipal Hospital dataset are leading to quasi-complete separation?

Quasi complete separation in logistic regression happens when the target variables separates a predictor variable or a set of predictor variables almost completely. For example:

Gender	Marital Status	Target
Male	Married	Yes
Male	Single	Yes
Female	Single	No
Male	Single	Yes
Male	Married	Yes
Male	Married	Yes
Female	Single	No

In this example, the variable **Marital Status is causing Quasi Complete Separation** as, the class Married is predicting the Target class completely as Yes. For class Single, The Target takes both Yes and No values.

The variable **Gender is causing Complete separation** in this example, because for all instances of Male, the Target class is Yes and for all Female instances the target class is No.

Quasi-Complete Separation in Manipal Health Enterprises Data:

```r
library(brglm2)
library(brglm)

## Loading required package: profileModel

## 'brglm' will gradually be superseded by 'brglm2' (https://cran.r-project.org/package=b
rglm2), which provides utilities for mean and median bias reduction for all GLMs and meth
ods for the detection of infinite estimates in binomial-response models.

binaryTrain$NPS_Status<- as.factor(binaryTrain$NPS_Status)

glm(NPS_Status ~ .-CE_NPS, data = binaryTrain, family='binomial',
    method = "detect_separation", linear_program = "dual")
```

```
## Separation: FALSE
## Existence of maximum likelihood estimates
##                       (Intercept)                                 SN
##                               Inf                                  0
##                       HospitalNo2             MaritalStatusMarried
##                                 0                                Inf
##            MaritalStatusSeparated               MaritalStatusSingle
##                               Inf                                Inf
##              MaritalStatusWidowed                            AgeYrs
##                               Inf                                  0
##                              SexM                 BedCategoryDAYCARE
##                                 0                               -Inf
##                BedCategoryGENERAL              BedCategoryGENERAL HD
##                              -Inf                               -Inf
##                    BedCategoryITU              BedCategoryRenal ICU
##                               Inf                                Inf
##            BedCategorySEMISPECIAL          BedCategorySEMISPECIAL HD
##                              -Inf                               -Inf
##                BedCategorySPECIAL              BedCategoryULTRA DLX
##                              -Inf                               -Inf
##              BedCategoryULTRA SPL                     DepartmentGEN
##                              -Inf                                  0
##                  DepartmentGYNAEC                   DepartmentORTHO
##                                 0                                  0
##               DepartmentPEDIATRIC                   DepartmentRENAL
##                                 0                                  0
##                 DepartmentSPECIAL                     Estimatedcost
##                                 0                                  0
##         InsPayorcategoryEXEMPTION          InsPayorcategoryINSURANCE
##                                 0                                  0
##     InsPayorcategoryINTERNATIONAL            InsPayorcategoryPATIENT
##                                 0                                  0
##          StateAndaman And Nicobar               StateAndhra Pradesh
##                               Inf                               -Inf
##                       StateAssam                    StateBangladesh
##                              -Inf                                Inf
##                 StateBhubaneshwar                        StateBihar
##                               Inf                               -Inf
##                   StateChandigarh                 StateChhattisgarh
##                               Inf                               -Inf
##                   StateDarjeeling                        StateDelhi
##                               Inf                               -Inf
##                         StateDoha                      StateGermany
##                               Inf                                Inf
##                          StateGoa                      StateGujarat
##                              -Inf                                Inf
##                      StateHaryana                 StateInternational
##                              -Inf                                Inf
##                         StateIraq                     StateJharkand
##                              -Inf                                Inf
##                    StateJharkhand                    StateKarnataka
##                              -Inf                               -Inf
##                        StateKenya                       StateKerala
##                               Inf                               -Inf
##                      StateKolkata                     StateKolkatta
```

```
##                                     -Inf                                     -Inf
##                     StateMadhya Pradesh                        StateMaharashtra
##                                     -Inf                                     -Inf
##                           StateMaldives                            StateManipur
##                                     -Inf                                     -Inf
##                          StateMauritius                          StateMeghalaya
##                                     -Inf                                      Inf
##                            StateMizoram                           StateMongolia
##                                      Inf                                     -Inf
##                             StateMumbai                             StateMuscat
##                                      Inf                                      Inf
##                              StateNepal                        StateNew Zealand
##                                     -Inf                                     -Inf
##                            StateNigeria                               StateOman
##                                      Inf                                     -Inf
##                            StateOntario                             StateOrissa
##                                     -Inf                                     -Inf
##                          StateRajasthan                             StateRanchi
##                                     -Inf                                     -Inf
##                             StateRWANDA                       StateSaudi Arabia
##                                        0                                      Inf
##                             StateSikkim                         StateTamil Nadu
##                                      Inf                                     -Inf
##                           StateTanzania                            StateTripura
##                                     -Inf                                     -Inf
##                                StateUAE                                 StateUK
##                                      Inf                                     -Inf
##                            StateUnknown                                StateUSA
##                                     -Inf                                      Inf
##                     StateUttar Pradesh                        StateUttarakhand
##                                     -Inf                                     -Inf
##                        StateWest Bengal                           StateZimbabwe
##                                     -Inf                                     -Inf
##                           CountryANGOLA                       CountryBANGLADESH
##                                     -Inf                                      Inf
##                           CountryCANADA                             CountryFIJI
##                                       NA                                      Inf
##                          CountryGERMANY                            CountryINDIA
##                                       NA                                        0
##                             CountryIRAQ    CountryISLAMIC REPUBLIC OF IRAN
##                                        0                                      Inf
##                            CountryKENYA                         CountryMALDIVES
##                                       NA                                       NA
##                       CountryMAURITIUS                         CountryMONGOLIA
##                                        0                                       NA
##                      CountryMOZAMBIQUE                            CountryNEPAL
##                                      Inf                                        0
##                     CountryNEW ZEALAND                          CountryNIGERIA
##                                       NA                                      Inf
##                             CountryOMAN                            CountryQATAR
##                                        0                                       NA
##                     CountrySaudi Arabia                     CountrySAUDI ARABIA
##                                      Inf                                       NA
##                            CountrySUDAN                           CountryUGANDA
##                                     -Inf                                       NA
```

```
##       CountryUNITED ARAB EMIRATES              CountryUNITED KINGDOM
##                                NA                                 NA
## CountryUNITED REPUBLIC OF TANZANIA    CountryUNITED STATES OF AMERICA
##                                NA                                 NA
##                      CountryYEMEN                    CountryZIMBABWE
##                                NA                                 NA
##                     STATEZONEEAST           STATEZONEINTERNATIONAL
##                                NA                                 NA
##                    STATEZONENORTH                    STATEZONESOUTH
##                                NA                                 NA
##                  STATEZONEUnknown                     STATEZONEWEST
##                                NA                                 NA
##                  CE_ACCESSIBILITY                           CE_CSAT
##                                 0                                  0
##                  CE_VALUEFORMONEY             EM_IMMEDIATEATTENTION
##                                 0                                  0
##                        EM_NURSING                         EM_DOCTOR
##                                 0                                  0
##                        EM_OVERALL                           AD_TIME
##                                 0                                  0
##     AD_TARRIFFPACKAGESEXPLAINATION                  AD_STAFFATTITUDE
##                                 0                                  0
##               INR_ROOMCLEANLINESS                     INR_ROOMPEACE
##                                 0                                  0
##                 INR_ROOMEQUIPMENT                  INR_ROOMAMBIENCE
##                                 0                                  0
##                   FNB_FOODQUALITY               FNB_FOODDELIVERYTIME
##                                 0                                  0
##                     FNB_DIETICIAN                 FNB_STAFFATTITUDE
##                                 0                                  0
##                   AE_ATTENDEECARE              AE_PATIENTSTATUSINFO
##                                 0                                  0
##                   AE_ATTENDEEFOOD          DOC_TREATMENTEXPLAINATION
##                                 0                                  0
##                      DOC_ATTITUDE                        DOC_VISITS
##                                 0                                  0
##        DOC_TREATMENTEFFECTIVENESS                NS_CALLBELLRESPONSE
##                                 0                                  0
##                NS_NURSESATTITUDE              NS_NURSEPROACTIVENESS
##                                 0                                  0
##                 NS_NURSEPATIENCE          OVS_OVERALLSTAFFATTITUDE
##                                 0                                  0
##        OVS_OVERALLSTAFFPROMPTNESS             OVS_SECURITYATTITUDE
##                                 0                                  0
##                  DP_DISCHARGETIME               DP_DISCHARGEQUERIES
##                                 0                                  0
##               DP_DISCHARGEPROCESS                     AdmissionDate
##                                 0                                  0
##                     DischargeDate                      LengthofStay
##                                 0                                 NA
## 0: finite value, Inf: infinity, -Inf: -infinity
```

The variables which are causing Quasi Complete Separation are: NPS_Status, MaritalStatus, BedCategory, State & Country.

Pre-processing of Binary-class Data:

```
#Removing Serial number variable SN and HospitalNo2 from Binary data-set, as they do not
contribute to determining the response variable prediction:
binaryTrain <- binaryTrain %>% select(-SN)
binaryTest <- binaryTest %>% select(-SN)

binaryTrain <- binaryTrain %>% select(-HospitalNo2)
binaryTest <- binaryTest %>% select(-HospitalNo2)

binaryTrain <- binaryTrain %>% select(-AdmissionDate)
binaryTest <- binaryTest %>% select(-AdmissionDate)

binaryTrain <- binaryTrain %>% select(-DischargeDate)
binaryTest <- binaryTest %>% select(-DischargeDate)

#Converting categorical variables into factor

binaryTrain$NPS_Status<- as.factor(binaryTrain$NPS_Status)
binaryTest$NPS_Status<- as.factor(binaryTest$NPS_Status)

binaryTrain$MaritalStatus <- as.factor(binaryTrain$MaritalStatus)
binaryTest$MaritalStatus <- as.factor(binaryTest$MaritalStatus)

binaryTrain$Sex <- as.factor(binaryTrain$Sex)
binaryTest$Sex <- as.factor(binaryTest$Sex)

binaryTrain$BedCategory <- as.factor(binaryTrain$BedCategory)
binaryTest$BedCategory <- as.factor(binaryTest$BedCategory)

binaryTrain$Department <- as.factor(binaryTrain$Department)
binaryTest$Department <- as.factor(binaryTest$Department)

binaryTrain$InsPayorcategory <- as.factor(binaryTrain$InsPayorcategory)
binaryTest$InsPayorcategory <- as.factor(binaryTest$InsPayorcategory)

binaryTrain$State <- as.factor(binaryTrain$State)
binaryTest$State <- as.factor(binaryTest$State)

binaryTrain$Country <- as.factor(binaryTrain$Country)
binaryTest$Country <- as.factor(binaryTest$Country)

binaryTrain$STATEZONE <- as.factor(binaryTrain$STATEZONE)
binaryTest$STATEZONE <- as.factor(binaryTest$STATEZONE)
```

5. What is orthogonal polynomial coding and how is it implemented in contrasting ordinal variables?

Orthogonal polynomial coding is a form of trend analysis in that it is looking for the linear, quadratic and cubic trends in the categorical variable. This type of coding tries to find out the impact of variable transformations on the target variable. The transformations include Linear, Quadratic & Cubic transformation of predictor.

This coding system should be used only with an ordinal variable in which the levels are equally spaced. Examples of such a variable might be survey questions like satisfaction scale (Extremely Unhappy, Okay , Extremely happy) or clothes Size chart (XS,S,M,L,XL).

The table below shows the contrast coefficients for the linear, quadratic and cubic trends for the four levels. In R it is not necessary to compute these values since this contrast can be obtained for any categorical variable by using the contr.poly function. This is also the default contrast used for ordered factor variables.

Implementation of Orthogonal Polynomial Coding in factor variables in the Manipal Health Data: (Variables used - CE_Accessibility, CE_CSAT)

```
orth.poly <- binaryTrain

contr.poly(4)

##                 .L   .Q          .C
## [1,] -0.6708204  0.5 -0.2236068
## [2,] -0.2236068 -0.5  0.6708204
## [3,]  0.2236068 -0.5 -0.6708204
## [4,]  0.6708204  0.5  0.2236068

orth.poly$Ord.ACCESSIBILITY<- factor(orth.poly$CE_ACCESSIBILITY, order = TRUE, levels = c
("1", "2", "3", "4"))

contrasts(orth.poly$Ord.ACCESSIBILITY) = contr.poly(4)
summary(glm(NPS_Status ~ Ord.ACCESSIBILITY, orth.poly, family = "binomial"))

##
## Call:
## glm(formula = NPS_Status ~ Ord.ACCESSIBILITY, family = "binomial",
##     data = orth.poly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6019  -1.1445   0.8057   0.8057   2.0074
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -0.6694     0.1982  -3.378 0.000731 ***
## Ord.ACCESSIBILITY.L    2.2582     0.5126   4.405 1.06e-05 ***
## Ord.ACCESSIBILITY.Q    0.4255     0.3964   1.074 0.283027
## Ord.ACCESSIBILITY.C   -0.4461     0.2268  -1.967 0.049228 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6578.3  on 4988  degrees of freedom
```

```
## Residual deviance: 6142.6  on 4985  degrees of freedom
## AIC: 6150.6
##
## Number of Fisher Scoring iterations: 4
```

For variable CE_ACCESSIBILITY, the linear and cubic coefficients have P-values less than 0.05 and thus are significant in determining the target variable NPS_Status. But the Quadratic coefficient has a larger p-value and thus is not significant.

```
orth.poly$Ord.CSAT <- factor(orth.poly$CE_CSAT, order = TRUE, levels = c("1", "2", "3", "
4"))

contrasts(orth.poly$Ord.CSAT) = contr.poly(4)
summary(glm(NPS_Status ~ Ord.CSAT, orth.poly, family = "binomial"))

##
## Call:
## glm(formula = NPS_Status ~ Ord.CSAT, family = "binomial", data = orth.poly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8000  -1.1073   0.6641   0.6641   2.4506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.0405     0.2089  -4.982 6.30e-07 ***
## Ord.CSAT.L    3.1998     0.5032   6.359 2.04e-10 ***
## Ord.CSAT.Q    1.0381     0.4177   2.485  0.01295 *
## Ord.CSAT.C   -1.0089     0.3095  -3.260  0.00111 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6578.3  on 4988  degrees of freedom
## Residual deviance: 5692.8  on 4985  degrees of freedom
## AIC: 5700.8
##
## Number of Fisher Scoring iterations: 5

LRTrain <- MultiTrain
LRTest <- MultiTest
```

For variable CE_CSAT, the linear, quadratic and cubic coefficients have P-values less than 0.05 and thus are significant in determining the target variable NPS_Status.

6.

Convert multi-class problem to binary class problem:

```
# Converting multi class to binary class problem: by converting individual classes of tar
get variable into variables Detractors, Passive & Promoters

LRTrain <- LRTrain %>%
  mutate(Detractors = ifelse(NPS_Status == "Detractor", "Yes", "No") )

LRTrain <- LRTrain %>% select(-NPS_Status)

LRTest <- LRTest %>%
  mutate(Detractors = ifelse(NPS_Status == "Detractor", "Yes", "No") )

LRTest <- LRTest %>% select(-NPS_Status)

LRTrain$Detractors <- as.factor(LRTrain$Detractors)
LRTest$Detractors <- as.factor(LRTest$Detractors)
```

Removing variables that cause Quasi-complete Separation:

```
#Removing variables found out in Quasi Complete separation from Train & Test: MaritalStat
us,BedCategory,State, Country, CE_NPS

remove.vars <- names(LRTrain) %in% c("MaritalStatus", "BedCategory", "State", "Country",
"CE_NPS")
LRTrain <- LRTrain[!remove.vars]
LRTest <- LRTest[!remove.vars]
```

Convert survey questionnaire responses to Ordinal:

```
#Converting survey responses into ordinal variables by normalizing
options(digits=2)
normalize <- function(x) {
    return ((x - min(x)) / (max(x) - min(x)))
}

LRordinalTr <- as.data.frame(lapply(LRTrain[7:41], normalize))
LRTrain[7:41] <- LRordinalTr[1:35]

LRordinalTst <- as.data.frame(lapply(LRTest[7:41], normalize))
LRTest[7:41] <- LRordinalTst[1:35]
```

Step-wise Logistic Regression for Binary Class problem:

```
# Logistic Regression on binary classification


FullGlmModel <- glm(Detractors ~. , data = LRTrain, family = "binomial")

StepGlmModel <- glm(Detractors ~. , data = LRTrain, family = "binomial")%>%
  stepAIC(trace = FALSE)

summary(FullGlmModel)

##
## Call:
## glm(formula = Detractors ~ ., family = "binomial", data = LRTrain)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.973  -0.413  -0.185  -0.109   3.541
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    7.48e+00   1.15e+00    6.52  7.0e-11 ***
## AgeYrs                         5.40e-03   3.38e-03    1.60   0.1105
## SexM                           1.60e-01   1.22e-01    1.31   0.1895
## DepartmentGEN                  6.29e-02   2.60e-01    0.24   0.8089
## DepartmentGYNAEC              -2.79e-02   3.44e-01   -0.08   0.9352
## DepartmentORTHO               -3.00e-01   3.30e-01   -0.91   0.3644
## DepartmentPEDIATRIC            1.06e-01   3.38e-01    0.31   0.7541
## DepartmentRENAL               -8.29e-01   4.19e-01   -1.98   0.0479 *
## DepartmentSPECIAL             -3.14e-01   2.86e-01   -1.10   0.2725
## Estimatedcost                  2.14e-07   7.77e-07    0.27   0.7834
## InsPayorcategoryEXEMPTION     -2.92e-01   3.33e-01   -0.88   0.3808
## InsPayorcategoryINSURANCE     -1.71e-02   2.45e-01   -0.07   0.9444
## InsPayorcategoryINTERNATIONAL  3.00e-01   8.05e-01    0.37   0.7089
## InsPayorcategoryPATIENT        2.23e-01   2.43e-01    0.92   0.3586
## STATEZONEEAST                 -1.94e+00   9.64e-01   -2.01   0.0445 *
## STATEZONEINTERNATIONAL        -1.71e+00   1.21e+00   -1.42   0.1564
## STATEZONENORTH                -1.37e+00   1.22e+00   -1.12   0.2640
## STATEZONESOUTH                -1.46e+00   9.38e-01   -1.55   0.1203
## STATEZONEUnknown              -1.91e+00   1.09e+00   -1.76   0.0792 .
## STATEZONEWEST                 -1.74e+00   1.25e+00   -1.39   0.1647
## CE_ACCESSIBILITY              -1.68e+00   3.25e-01   -5.17  2.4e-07 ***
## CE_CSAT                       -2.69e+00   3.94e-01   -6.85  7.6e-12 ***
## CE_VALUEFORMONEY              -3.20e+00   3.33e-01   -9.60  < 2e-16 ***
## EM_IMMEDIATEATTENTION         -2.76e-01   4.75e-01   -0.58   0.5609
## EM_NURSING                    -9.54e-01   5.57e-01   -1.71   0.0868 .
## EM_DOCTOR                     -4.88e-01   5.51e-01   -0.89   0.3759
## EM_OVERALL                     1.93e-01   5.76e-01    0.33   0.7381
## AD_TIME                        7.85e-01   3.36e-01    2.34   0.0195 *
## AD_TARRIFFPACKAGESEXPLAINATION -9.61e-01   3.63e-01   -2.65   0.0081 **
## AD_STAFFATTITUDE              -1.18e-01   3.86e-01   -0.30   0.7607
## INR_ROOMCLEANLINESS           -5.43e-01   3.62e-01   -1.50   0.1333
## INR_ROOMPEACE                 -1.84e-01   3.37e-01   -0.55   0.5842
```

```
## INR_ROOMEQUIPMENT                4.07e-01   4.05e-01    1.01   0.3148
## INR_ROOMAMBIENCE                -2.41e-01   4.49e-01   -0.54   0.5917
## FNB_FOODQUALITY                 -3.61e-01   3.31e-01   -1.09   0.2752
## FNB_FOODDELIVERYTIME            -6.93e-01   3.57e-01   -1.94   0.0524 .
## FNB_DIETICIAN                   -4.44e-02   3.88e-01   -0.11   0.9090
## FNB_STAFFATTITUDE                3.94e-01   4.15e-01    0.95   0.3417
## AE_ATTENDEECARE                 -2.46e-01   4.22e-01   -0.58   0.5607
## AE_PATIENTSTATUSINFO            -3.18e-01   4.51e-01   -0.70   0.4812
## AE_ATTENDEEFOOD                  3.77e-01   3.52e-01    1.07   0.2836
## DOC_TREATMENTEXPLAINATION       -3.00e-01   5.64e-01   -0.53   0.5946
## DOC_ATTITUDE                     1.48e-01   5.94e-01    0.25   0.8028
## DOC_VISITS                      -1.16e+00   4.36e-01   -2.67   0.0077 **
## DOC_TREATMENTEFFECTIVENESS       2.04e-01   5.12e-01    0.40   0.6900
## NS_CALLBELLRESPONSE              2.77e-01   4.40e-01    0.63   0.5282
## NS_NURSESATTITUDE                1.96e-01   5.82e-01    0.34   0.7361
## NS_NURSEPROACTIVENESS            4.25e-01   3.87e-01    1.10   0.2722
## NS_NURSEPATIENCE                 7.57e-01   5.73e-01    1.32   0.1868
## OVS_OVERALLSTAFFATTITUDE        -5.99e-01   5.92e-01   -1.01   0.3121
## OVS_OVERALLSTAFFPROMPTNESS      -2.77e-01   4.63e-01   -0.60   0.5497
## OVS_SECURITYATTITUDE             1.16e+00   4.42e-01    2.62   0.0089 **
## DP_DISCHARGETIME                -9.65e-01   4.40e-01   -2.19   0.0283 *
## DP_DISCHARGEQUERIES             -1.66e+00   4.47e-01   -3.72   0.0002 ***
## DP_DISCHARGEPROCESS              1.20e+00   5.54e-01    2.17   0.0301 *
## LengthofStay                    -1.65e-02   1.55e-02   -1.06   0.2871
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3257.3  on 4988  degrees of freedom
## Residual deviance: 2163.0  on 4933  degrees of freedom
## AIC: 2275
##
## Number of Fisher Scoring iterations: 6
```

```r
summary(StepGlmModel)
```

```
##
## Call:
## glm(formula = Detractors ~ AgeYrs + Sex + Department + CE_ACCESSIBILITY +
##     CE_CSAT + CE_VALUEFORMONEY + EM_NURSING + AD_TIME + AD_TARRIFFPACKAGESEXPLAINATION
+
##     INR_ROOMCLEANLINESS + FNB_FOODDELIVERYTIME + DOC_VISITS +
##     NS_NURSEPATIENCE + OVS_SECURITYATTITUDE + DP_DISCHARGETIME +
##     DP_DISCHARGEQUERIES + DP_DISCHARGEPROCESS, family = "binomial",
##     data = LRTrain)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.927  -0.418  -0.189  -0.112   3.440
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     5.54770    0.49367   11.24  < 2e-16 ***
## AgeYrs                          0.00535    0.00327    1.64  0.10173
## SexM                            0.17610    0.12050    1.46  0.14391
```

```
## DepartmentGEN                       0.18584    0.24420    0.76  0.44666
## DepartmentGYNAEC                   -0.00693    0.33858   -0.02  0.98368
## DepartmentORTHO                    -0.16489    0.31546   -0.52  0.60118
## DepartmentPEDIATRIC                 0.22910    0.32279    0.71  0.47785
## DepartmentRENAL                    -0.70727    0.40345   -1.75  0.07960 .
## DepartmentSPECIAL                  -0.21639    0.27505   -0.79  0.43145
## CE_ACCESSIBILITY                   -1.63379    0.31412   -5.20  2.0e-07 ***
## CE_CSAT                            -2.59318    0.38630   -6.71  1.9e-11 ***
## CE_VALUEFORMONEY                   -3.34362    0.31950  -10.47  < 2e-16 ***
## EM_NURSING                         -1.38326    0.35258   -3.92  8.7e-05 ***
## AD_TIME                             0.70016    0.31471    2.22  0.02609 *
## AD_TARRIFFPACKAGESEXPLAINATION -0.95133    0.33956   -2.80  0.00508 **
## INR_ROOMCLEANLINESS                -0.66341    0.27799   -2.39  0.01701 *
## FNB_FOODDELIVERYTIME               -0.67535    0.28527   -2.37  0.01791 *
## DOC_VISITS                         -1.27024    0.31645   -4.01  6.0e-05 ***
## NS_NURSEPATIENCE                    1.21256    0.39549    3.07  0.00217 **
## OVS_SECURITYATTITUDE                0.94859    0.37971    2.50  0.01248 *
## DP_DISCHARGETIME                   -0.87462    0.43095   -2.03  0.04241 *
## DP_DISCHARGEQUERIES                -1.66469    0.43312   -3.84  0.00012 ***
## DP_DISCHARGEPROCESS                 1.12824    0.54310    2.08  0.03776 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3257.3  on 4988  degrees of freedom
## Residual deviance: 2187.5  on 4966  degrees of freedom
## AIC: 2233
##
## Number of Fisher Scoring iterations: 6
```

In the stepwise Regression model, the number of variables were reduced to 17.

The variables used in the Step-wise Regression Model are: AgeYrs + Sex + Department + CE_ACCESSIBILITY + CE_CSAT + CE_VALUEFORMONEY + EM_NURSING + AD_TIME + AD_TARRIFFPACKAGESEXPLAINATION + INR_ROOMCLEANLINESS + FNB_FOODDELIVERYTIME + DOC_VISITS + NS_NURSEPATIENCE + OVS_SECURITYATTITUDE + DP_DISCHARGETIME + DP_DISCHARGEQUERIES + DP_DISCHARGEPROCESS

Prediction and Confusion Matrix for Full Logistic Regression Model on Test Data

```
#Prediction accuracy of the Full logistic regression model:

pred <- predict(FullGlmModel, LRTest, type = "response")
pred.model <- rep("No", length(pred))
pred.model[pred > 0.5] <- "Yes"
confusionMatrix(table(pred.model, LRTest$Detractors), positive = "Yes")

## Confusion Matrix and Statistics
##
##
## pred.model  No Yes
##        No  309  25
##        Yes  11  19
##
##                Accuracy : 0.901
##                  95% CI : (0.866, 0.93)
##     No Information Rate : 0.879
##     P-Value [Acc > NIR] : 0.1119
##
##                   Kappa : 0.461
##
##  Mcnemar's Test P-Value : 0.0303
##
##             Sensitivity : 0.4318
##             Specificity : 0.9656
##          Pos Pred Value : 0.6333
##          Neg Pred Value : 0.9251
##              Prevalence : 0.1209
##          Detection Rate : 0.0522
##    Detection Prevalence : 0.0824
##       Balanced Accuracy : 0.6987
##
##        'Positive' Class : Yes
##
```

Prediction Accuracy for Full Logistic Regression model comes out to be 90.1%

Prediction and Confusion Matrix for Step-wise Logistic Regression Model on Test Data

```
# Prediction accuracy of the stepwise logistic regression model:

pred <- predict(StepGlmModel, LRTest, type = "response")
pred.model <- rep("No", length(pred))
pred.model[pred > 0.5] <- "Yes"
confusionMatrix(table(pred.model, LRTest$Detractors), positive = "Yes")

## Confusion Matrix and Statistics
##
##
## pred.model  No Yes
##        No  311  25
##        Yes   9  19
```

```
##
##                Accuracy : 0.907
##                  95% CI : (0.872, 0.934)
##     No Information Rate : 0.879
##     P-Value [Acc > NIR] : 0.0596
##
##                   Kappa : 0.479
##
##  Mcnemar's Test P-Value : 0.0101
##
##             Sensitivity : 0.4318
##             Specificity : 0.9719
##          Pos Pred Value : 0.6786
##          Neg Pred Value : 0.9256
##              Prevalence : 0.1209
##          Detection Rate : 0.0522
##    Detection Prevalence : 0.0769
##       Balanced Accuracy : 0.7018
##
##        'Positive' Class : Yes
##
```

Prediction Accuracy for Step-wise Logistic Regression model is 90.7%.

The accuracy comes out to be 90.7%. This is slightly better than full model(90.1%), but it has very few variables as compared to the full model. Thus, the Step wise model avoids overfitting to the training data.

7.

With the help of ensemble methods we want to identify the Detractors and Promotors among the customers. And we also want to understand why a customer is falling in any of the 3 profiles of Promotor, Detractor or Passive.

The variable CE_NPS is removed from the predictor variables, because in a sense it is our target variable, since we determine the NPS_Status of a customer on the basis of their NPS Score, i.e., NPS Score of 0-6 is Detractor, 7-8 is Passive and 9-10 is Promotor.

Note: I tried running the models with the variable CE_NPS, and the accuracy of those model came to be 1. On checking the important variables through importance function, I found that CE_NPS has a very high Mean Decrease Gini (~2000). But the model was not much informative of other important variables responsible for the given NPS Score / Status. Hence I decided to remove CE_NPS from the predictors.

Random Forest for Multi-class Classification:

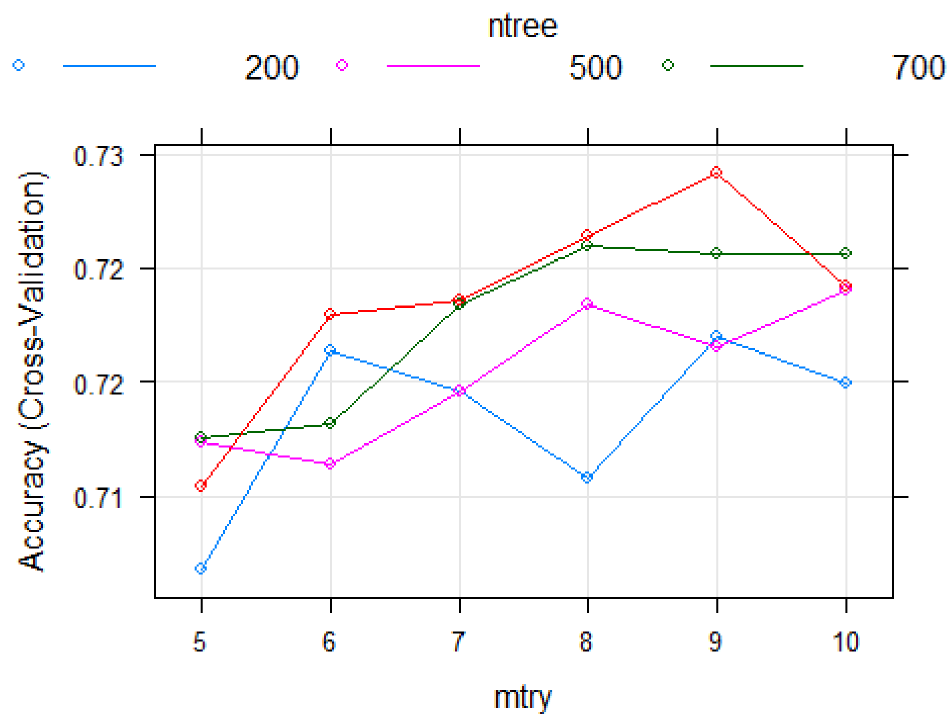
```
multirfTrain <- MultiTrain %>% select(-CE_NPS)
multirfTest <- MultiTest %>% select(-CE_NPS)
RFdata <- rbind(multirfTrain,multirfTest)

rfData_cat <- dplyr::select_if(RFdata, is.factor)
sapply(rfData_cat, function(x) length(unique(x)))

##    MaritalStatus              Sex       BedCategory        Department
##                5                2                11                 7
## InsPayorcategory            State           Country         STATEZONE
##                5               58                29                 7
##       NPS_Status
##                3

#We remove variables that have number of classes more than 53.

RFdata <- RFdata %>%
  select(-State)

multirfTrain <- multirfTrain %>%
  select(-State)

multirfTest <- multirfTest %>%
  select(-State)

# Setting the number of levels of factor variables in Training & Test data as same

common <- intersect(names(multirfTrain), names(multirfTest))
for (p in common) {
  if (class(multirfTrain[[p]]) == "factor") {
    levels(multirfTest[[p]]) <- levels(multirfTrain[[p]]) } }
```

Random Forest for Multi Class Classification:

Step 1: Cross Validation for Parameter Tuning best mtry and ntree

```
library(randomForest)
library(caret)

set.seed(101)
MHE_rf <- list(type = "Classification", library = "randomForest", loop = NULL)

MHE_rf$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric", 2)
, label = c("mtry", "ntree"))

MHE_rf$grid <- function(x, y, len = NULL, search = "grid") {}
MHE_rf$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...)
  {
  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
  }

MHE_rf$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL) predict(m
odelFit, newdata)
```

```
MHE_rf$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)

  predict(modelFit, newdata, type = "prob")

MHE_rf$sort <- function(x) x[order(x[,1]),]

MHE_rf$levels <- function(x) x$classes

control <- trainControl(method="cv", number=3)
tunegrid <- expand.grid(.mtry=c(5:10), .ntree=c(100,200,500,700))

set.seed(111)

multiRF <- train(NPS_Status ~.-NPS_Status, data=multirfTrain,
                   method=MHE_rf, metric="Accuracy",
                   tuneGrid=tunegrid, trControl=control)

plot(multiRF)
```




```
multiRF

## 4989 samples
##   45 predictor
##    3 classes: 'Detractor', 'Passive', 'Promotor'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3326, 3326, 3326
## Resampling results across tuning parameters:
##
##   mtry  ntree  Accuracy  Kappa
##    5    100    0.71      0.35
##    5    200    0.71      0.36
```

```
##    5    500    0.71      0.36
##    5    700    0.71      0.36
##    6    100    0.72      0.38
##    6    200    0.71      0.37
##    6    500    0.71      0.37
##    6    700    0.72      0.38
##    7    100    0.71      0.38
##    7    200    0.71      0.38
##    7    500    0.72      0.39
##    7    700    0.72      0.39
##    8    100    0.71      0.38
##    8    200    0.72      0.39
##    8    500    0.72      0.40
##    8    700    0.72      0.40
##    9    100    0.72      0.40
##    9    200    0.72      0.39
##    9    500    0.72      0.40
##    9    700    0.72      0.41
##   10    100    0.71      0.39
##   10    200    0.72      0.40
##   10    500    0.72      0.40
##   10    700    0.72      0.40

## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 9 and ntree = 700.
```

Accuracy is highest for mtry = 9 and ntree = 700, with corresponding accuracy around ~72%.

Random Forest for Multi Class: Cross Validation for Model Evaluation –

I conducted a 10-fold cross validation and find out the accuracy of Random Forest for Multi-class classification problem, with the best mtry = 9 and best ntree = 700.

```
set.seed(111)
library(e1071)
library(pROC)
library(randomForest)
library(caret)
library(AUC)

k1 = 10

n = floor(nrow(RFdata)/k1)
accuracy.vect = rep(NA,k1)

for (i in 1:k1) {


  s1 = ((i-1) * n+1)
  s2 = (i*n)
  subset = s1:s2

  multirfcv.train = RFdata[-subset,]
  multirfcv.test = RFdata[subset,]
```

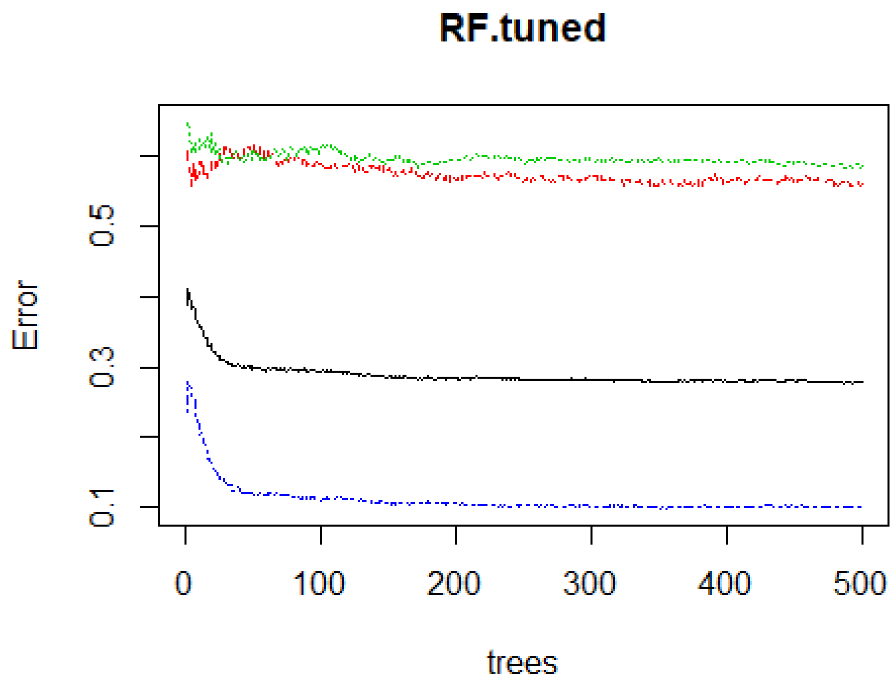
```
  tuned.RandForest <- randomForest(NPS_Status~.-NPS_Status, data = multirfcv.train, mtry
= 9, ntree = 700 )

  tuned.RF.pred <- predict(tuned.RandForest,
                           newdata = multirfcv.test, type = "class")

  accuracy.vect[i] <- (confusionMatrix(tuned.RF.pred, multirfcv.test$NPS_Status))$overall
[1]

 print(paste("Accuracy for fold", i, ":", accuracy.vect[i]))

    }

## [1] "Accuracy for fold 1 : 0.723364485981308"
## [1] "Accuracy for fold 2 : 0.738317757009346"
## [1] "Accuracy for fold 3 : 0.71588785046729"
## [1] "Accuracy for fold 4 : 0.753271028037383"
## [1] "Accuracy for fold 5 : 0.685981308411215"
## [1] "Accuracy for fold 6 : 0.700934579439252"
## [1] "Accuracy for fold 7 : 0.74018691588785"
## [1] "Accuracy for fold 8 : 0.730841121495327"
## [1] "Accuracy for fold 9 : 0.725233644859813"
## [1] "Accuracy for fold 10 : 0.695327102803738"

print(paste(" Average Accuracy for multiclass Random Forest :", mean(accuracy.vect)))

## [1] "Average Accuracy for multiclass Random Forest : 0.720934579439252"
```

Average Accuracy from 10-Fold cross Validation for multiclass Random Forest : 0.720934579439252

Retrain the model with best parameters obtained from Cross Validation and checking the performance measure - Accuracy on the test data.

```
set.seed(123)

#Retraining the model with best values of mtry and ntree

RF.tuned <- randomForest(NPS_Status ~. -NPS_Status,
                      data=multirfTrain,
                      importance = TRUE,
                      mtry = 9,
                      ntree = 500)
print(RF.tuned)
## Call:
##  randomForest(formula = NPS_Status ~ . - NPS_Status, data = multirfTrain,      importa
nce = TRUE, mtry = 9, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 28%
## Confusion matrix:
##           Detractor Passive Promotor class.error
## Detractor       221     148      133        0.56
## Passive          70     560      717        0.58
## Promotor         19     300     2821        0.10
```

```r
plot(RF.tuned)
```



Prediction and Confusion Matrix for Multi-class Random Forest on Test Data

```r
# Making final prediction on test data

RFtest.pred <- predict(RF.tuned, multirfTest, type = "prob")

confusionMatrix(predict(RF.tuned, newdata= multirfTest, type = "class"),
                multirfTest$NPS_Status)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  Detractor Passive Promotor
##   Detractor        20       8        1
##   Passive          15      46       18
##   Promotor          9      63      184
##
## Overall Statistics
##
##                Accuracy : 0.687
##                  95% CI : (0.636, 0.734)
##     No Information Rate : 0.558
##     P-Value [Acc > NIR] : 3.12e-07
##
##                   Kappa : 0.407
##
##  Mcnemar's Test P-Value : 2.49e-07
##
## Statistics by Class:
##
```

```
##                          Class: Detractor Class: Passive Class: Promotor
## Sensitivity                        0.4545          0.393           0.906
## Specificity                        0.9719          0.866           0.553
## Pos Pred Value                     0.6897          0.582           0.719
## Neg Pred Value                     0.9284          0.751           0.824
## Prevalence                         0.1209          0.321           0.558
## Detection Rate                     0.0549          0.126           0.505
## Detection Prevalence               0.0797          0.217           0.703
## Balanced Accuracy                  0.7132          0.630           0.730
```

Accuracy for multiclass Random Forest on Test Data : 68.7%

Important variables from multi-class Random Forest Model:

```
multirfImportant <- importance(RF.tuned, type = 2)

multiRFImportance <- data.frame(Variables = row.names(multirfImportant),
            Importance = round(multirfImportant[ ,'MeanDecreaseGini'],2))

multiRFImportance <- multiRFImportance[order((multiRFImportance$Importance),decreasing =
TRUE), ]

head(multiRFImportance, 20)

##                                                      Variables Importance
## AgeYrs                                                  AgeYrs        213
## Estimatedcost                                    Estimatedcost        170
## CE_VALUEFORMONEY                              CE_VALUEFORMONEY        160
## CE_CSAT                                                CE_CSAT        158
## LengthofStay                                      LengthofStay        137
## BedCategory                                        BedCategory        128
## Department                                          Department        120
## InsPayorcategory                              InsPayorcategory         81
## CE_ACCESSIBILITY                              CE_ACCESSIBILITY         77
## AE_ATTENDEEFOOD                                AE_ATTENDEEFOOD         66
## AD_TARRIFFPACKAGESEXPLAINATION AD_TARRIFFPACKAGESEXPLAINATION         57
## DP_DISCHARGETIME                              DP_DISCHARGETIME         56
## FNB_FOODDELIVERYTIME                      FNB_FOODDELIVERYTIME         55
## DP_DISCHARGEPROCESS                        DP_DISCHARGEPROCESS         55
## FNB_FOODQUALITY                                FNB_FOODQUALITY         53
## INR_ROOMCLEANLINESS                        INR_ROOMCLEANLINESS         47
## DP_DISCHARGEQUERIES                        DP_DISCHARGEQUERIES         46
## AD_TIME                                                AD_TIME         46
## INR_ROOMAMBIENCE                              INR_ROOMAMBIENCE         45
## FNB_DIETICIAN                                    FNB_DIETICIAN         42
```

Final Results for Random Forest for Multi Class Classification:

CV for Parameter Tuning	10-fold CV for model evaluation	Accuracy on Test Data
mtry = 9, ntree = 700	**Average Accuracy = 72.09%**	**Accuracy = 68.7%**

AdaBoost for Multi Class Classification

```
multiAdaTrain <- MultiTrain %>% select(-CE_NPS)
multiAdaTest <- MultiTest %>% select(-CE_NPS)

multiAdaData <- rbind(multiAdaTrain,multiAdaTest)

# Creating dummy variables for categorical variables

library(caret)
multiAda_nums <- dplyr::select_if(multiAdaData, is.numeric)
multiAda_cat <- dplyr::select_if(multiAdaData, is.factor)

var_onehot <- c('MaritalStatus','Sex','BedCategory','Department', "InsPayorcategory", "St
ate", "Country", "STATEZONE")

# One Hot Encoding

dummys <- dummyVars(" ~ .", data = multiAda_cat[,var_onehot])

dummy_cats <- data.frame(predict(dummys, newdata = multiAda_cat[,var_onehot]))

 new.multiAdaData <- cbind(multiAda_nums,dummy_cats,multiAda_cat$NPS_Status)

names(new.multiAdaData)[names(new.multiAdaData) =="multiAda_cat$NPS_Status"] <- "NPS_Stat
us"

multiAdaTrain <- new.multiAdaData[1:4989,]

multiAdaTest <- new.multiAdaData[4990:5353,]

# Setting the number of levels of factor variables in Training & Test data as same

common <- intersect(names(multiAdaTrain), names(multiAdaTest))
for (p in common) {
  if (class(multiAdaTrain[[p]]) == "factor") {
    levels(multiAdaTest[[p]]) <- levels(multiAdaTrain[[p]]) } }
```

Ada Boost for Multi Class Classification:

Cross Validation for Parameter Tuning - Cross validation is conducted to find the best value of mfinal (no. of iterations), Complexity parameter (cp) and maxdepth.

```
set.seed(111)
library("adabag")

## Loading required package: rpart

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel
```

```r
# Find the best model with the best mfinal, cp and maxdepth, via cross-validations

multi.best.mfinal <- NA
multi.best.cp <- NA
multi.best.maxdepth <- NA

highest.accuracy <- 0
for (m.final in c(10,20)) {
  for (comp.p in c(0.005,0.001,0.01)) {
    for (maxdepth in c(10, 20,30)){
    multiAdaBoost <- boosting(NPS_Status ~ ., data = multiAdaTrain,
                              mfinal = m.final,
                              control = rpart.control(maxdepth =maxdepth,
                                                      cp=comp.p))

    multipred.best <- as.factor((predict.boosting(multiAdaBoost,multiAdaTrain))$class)

    #levels(multipred.best)
    #levels(multiAdaTrain$NPS_Status)

    fold.accuracy <- (confusionMatrix(multipred.best, multiAdaTrain$NPS_Status)$overall)[
1]

    cat("Results for mfinal=",m.final," : ", "Complexity parameter = ",comp.p,":", "and m
axdepth = ",maxdepth,":", "Accuracy = ",fold.accuracy,"\n",sep="")

    if(fold.accuracy > highest.accuracy){
      highest.accuracy <- fold.accuracy
      multi.best.mfinal <- m.final
      multi.best.cp <- comp.p
      multi.best.maxdepth <- maxdepth

    }
  }
 }
}
```

```
## Results for mfinal=10 : Complexity parameter = 0.005:and maxdepth = 10:Accuracy = 0.72
## Results for mfinal=10 : Complexity parameter = 0.005:and maxdepth = 20:Accuracy = 0.73
## Results for mfinal=10 : Complexity parameter = 0.005:and maxdepth = 30:Accuracy = 0.73
## Results for mfinal=10 : Complexity parameter = 0.001:and maxdepth = 10:Accuracy = 0.78
## Results for mfinal=10 : Complexity parameter = 0.001:and maxdepth = 20:Accuracy = 0.86
## Results for mfinal=10 : Complexity parameter = 0.001:and maxdepth = 30:Accuracy = 0.87
## Results for mfinal=10 : Complexity parameter = 0.01:and maxdepth = 10:Accuracy = 0.71
## Results for mfinal=10 : Complexity parameter = 0.01:and maxdepth = 20:Accuracy = 0.7
## Results for mfinal=10 : Complexity parameter = 0.01:and maxdepth = 30:Accuracy = 0.72
## Results for mfinal=20 : Complexity parameter = 0.005:and maxdepth = 10:Accuracy = 0.73
## Results for mfinal=20 : Complexity parameter = 0.005:and maxdepth = 20:Accuracy = 0.73
## Results for mfinal=20 : Complexity parameter = 0.005:and maxdepth = 30:Accuracy = 0.73
## Results for mfinal=20 : Complexity parameter = 0.001:and maxdepth = 10:Accuracy = 0.8
```

```
## Results for mfinal=20 : Complexity parameter = 0.001:and maxdepth = 20:Accuracy = 0.94
## Results for mfinal=20 : Complexity parameter = 0.001:and maxdepth = 30:Accuracy = 0.95
## Results for mfinal=20 : Complexity parameter = 0.01:and maxdepth = 10:Accuracy = 0.71
## Results for mfinal=20 : Complexity parameter = 0.01:and maxdepth = 20:Accuracy = 0.71
## Results for mfinal=20 : Complexity parameter = 0.01:and maxdepth = 30:Accuracy = 0.71
```

```
cat("For Multi-class Classification:", "\n")
```

```
## For Multi-class Classification:
```

```
cat("Best mfinal (number of iterations) is:",m.final,"\n")
```

```
## Best mfinal (number of iterations) is: 20
```

```
cat("Best complexity parameter is:",comp.p,"\n")
```

```
## Best complexity parameter is: 0.001
```

```
cat("Best maxdepth is:",maxdepth,"\n")
```

```
## Best maxdepth is: 30
```

```
cat("Best accuracy is:",highest.accuracy,"\n")
```

```
## Best accuracy is: 0.95
```

For Adaboost Multi-class Classification:
Best mfinal = 20
Best Complexity Parameter = 0.001
Best maxdepth = 30
Best accuracy = 0.95

I am using complexity parameter as 0.005, as a lower complexity parameter may result in overfitting and cause greater test error.

Retraining the model with best parameters obtained from cross validation

```
set.seed(111)
  library("adabag")

bestmulti.adaboost <- boosting(NPS_Status ~ ., data = multiAdaTrain, mfinal = 20, control
= rpart.control(maxdepth = 30, cp=0.005 ))
```

Prediction and Confusion Matrix for Multi-class Adaboost on Training Data

```
multi.predboosting.tr <- as.factor(predict.boosting(bestmulti.adaboost,
                                      newdata = multiAdaTrain)$class)
confusionMatrix(multi.predboosting.tr, multiAdaTrain$NPS_Status)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  Detractor Passive Promotor
##   Detractor       218      64       15
##   Passive         124     543      208
##   Promotor        160     740     2917
##
## Overall Statistics
##
##                Accuracy : 0.737
##                  95% CI : (0.725, 0.749)
##     No Information Rate : 0.629
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.435
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Detractor Class: Passive Class: Promotor
## Sensitivity                    0.4343          0.403           0.929
## Specificity                    0.9824          0.909           0.513
## Pos Pred Value                 0.7340          0.621           0.764
## Neg Pred Value                 0.9395          0.805           0.810
## Prevalence                     0.1006          0.270           0.629
## Detection Rate                 0.0437          0.109           0.585
## Detection Prevalence           0.0595          0.175           0.765
## Balanced Accuracy              0.7083          0.656           0.721
```

Accuracy for multiclass Ada-Boost on Training Data : 73.7%

Prediction and Confusion Matrix for Multi-class Adaboost Model on Test Data

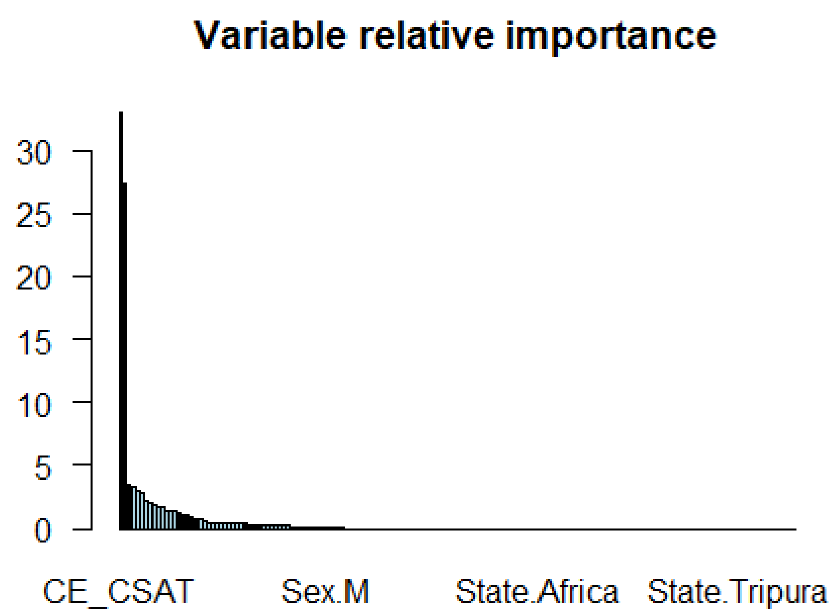
```
multi.predboosting <- as.factor(predict.boosting(bestmulti.adaboost,
                                      newdata = multiAdaTest)$class)
#length(multi.predboosting)
#length(multiAdaTest$NPS_Status)

confusionMatrix(multi.predboosting, multiAdaTest$NPS_Status)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  Detractor Passive Promotor
```

```
##   Detractor        20       9        2
##   Passive           8      41       17
##   Promotor         16      67      184
##
## Overall Statistics
##
##                Accuracy : 0.673
##                  95% CI : (0.622, 0.721)
##     No Information Rate : 0.558
##     P-Value [Acc > NIR] : 4.56e-06
##
##                   Kappa : 0.374
##
##  Mcnemar's Test P-Value : 7.54e-09
##
## Statistics by Class:
##
##                      Class: Detractor Class: Passive Class: Promotor
## Sensitivity                    0.4545          0.350           0.906
## Specificity                    0.9656          0.899           0.484
## Pos Pred Value                 0.6452          0.621           0.689
## Neg Pred Value                 0.9279          0.745           0.804
## Prevalence                     0.1209          0.321           0.558
## Detection Rate                 0.0549          0.113           0.505
## Detection Prevalence           0.0852          0.181           0.734
## Balanced Accuracy              0.7101          0.625           0.695
```

Accuracy for multiclass Ada-Boost on Test Data : 67.3%

Important variables from multi-class Adaboost Model:

```
importanceplot(bestmulti.adaboost)
```



```
MultiadaImportant <- bestmulti.adaboost$importance

head(sort(MultiadaImportant, decreasing = TRUE),30)

##                      CE_CSAT                CE_VALUEFORMONEY
##                      32.9684                         27.3753
##          DP_DISCHARGEQUERIES             BedCategory.GENERAL
##                       3.4338                          3.3482
##             CE_ACCESSIBILITY                          AgeYrs
##                       2.9343                          2.7836
##         FNB_FOODDELIVERYTIME                 AE_ATTENDEEFOOD
##                       2.1733                          2.0390
##              FNB_FOODQUALITY     AD_TARRIFFPACKAGESEXPLAINATION
##                       1.9280                          1.7477
##             INR_ROOMAMBIENCE             DP_DISCHARGEPROCESS
##                       1.7392                          1.3804
##                   DOC_VISITS               FNB_STAFFATTITUDE
##                       1.3470                          1.3373
##             DP_DISCHARGETIME            AE_PATIENTSTATUSINFO
##                       1.2698                          1.0963
##          INR_ROOMCLEANLINESS                   Estimatedcost
##                       1.0272                          0.9344
##              AE_ATTENDEECARE                       EM_DOCTOR
##                       0.6974                          0.6705
##     OVS_OVERALLSTAFFATTITUDE         BedCategory.SEMISPECIAL
##                       0.5532                          0.4574
##             AD_STAFFATTITUDE        OVS_OVERALLSTAFFPROMPTNESS
##                       0.4468                          0.4432
##       NS_NURSEPROACTIVENESS       DOC_TREATMENTEFFECTIVENESS
##                       0.4401                          0.4252
##                FNB_DIETICIAN                    DOC_ATTITUDE
##                       0.4241                          0.4202
##                   EM_NURSING           EM_IMMEDIATEATTENTION
##                       0.3875                          0.3858
##           NS_NURSESATTITUDE             BedCategory.SPECIAL
##                       0.3574                          0.3164
##               STATEZONE.EAST       InsPayorcategory.INSURANCE
##                       0.2560                          0.2461
##                   EM_OVERALL              NS_CALLBELLRESPONSE
##                       0.2304                          0.2230
##               Department.GEN                 STATEZONE.SOUTH
##                       0.2230                          0.2230
##                INR_ROOMPEACE               NS_NURSEPATIENCE
##                       0.1959                          0.1779
```

Final Results from Multi-Class Ada-Boost:

Parameter Tuning	Training accuracy with best parameters	Accuracy on Test Data
mfinal = 20, cp = 0.001, maxdepth = 30	**Average Accuracy = 73.7%**	**Accuracy = 67.3%**

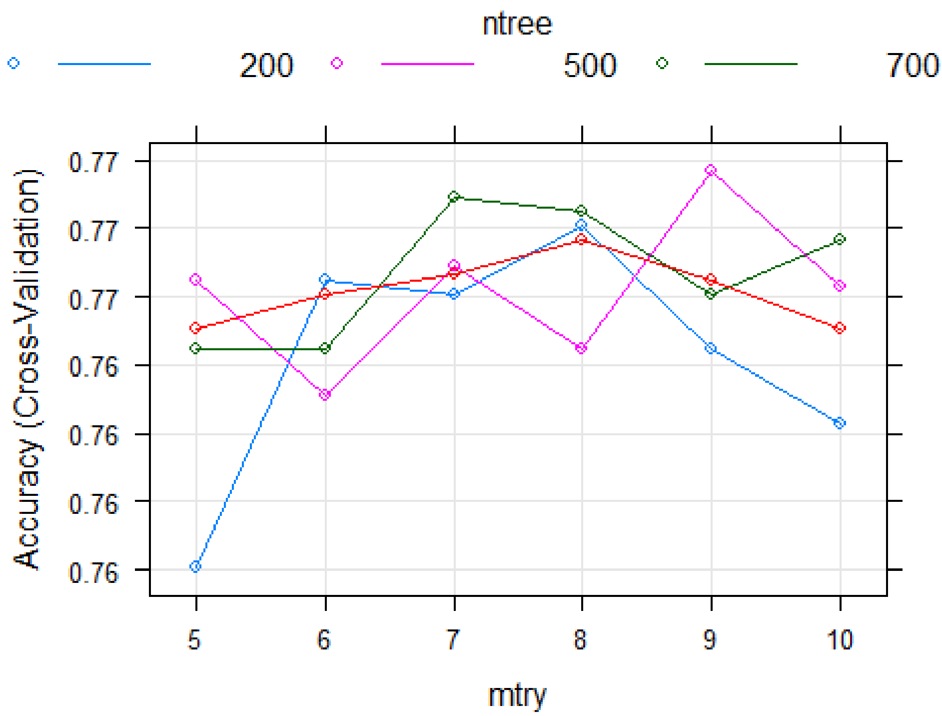
Random Forest for Binary classification:

```
binaryrf.train <- binaryTrain %>% select(-CE_NPS)

binaryrf.test <- binaryTest %>% select(-CE_NPS)

binaryrf.data <- rbind(binaryrf.train,binaryrf.test)


#We remove variables that have number of classes more than 53.

binaryrf.data <- binaryrf.data %>%
  select(-State)

binaryrf.train <- binaryrf.train %>%
  select(-State)

binaryrf.test <- binaryrf.test %>%
  select(-State)

# Setting the number of levels of factor variables in Training & Test data as same

common <- intersect(names(binaryrf.train), names(binaryrf.test))
for (p in common) {
  if (class(binaryrf.train[[p]]) == "factor") {
    levels(binaryrf.test[[p]]) <- levels(binaryrf.train[[p]]) } }
```

Random Forest for Binary classification:

Parameter Tuning for best mtry and ntree:

```
#Tuning Parameters for Binary class Random Forest

set.seed(111)

BinaryRF <- train(NPS_Status ~.-NPS_Status, data=binaryrf.train,
                  method=MHE_rf, metric="Accuracy",
                  tuneGrid=tunegrid, trControl=control)

plot(BinaryRF)
```




```
BinaryRF

## 4989 samples
##   45 predictor
##    2 classes: 'Detractor', 'Promotor'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3326, 3326, 3326
## Resampling results across tuning parameters:
##
##   mtry  ntree  Accuracy  Kappa
##    5    100    0.76      0.45
##    5    200    0.77      0.47
##    5    500    0.76      0.47
##    5    700    0.77      0.47
##    6    100    0.77      0.47
##    6    200    0.76      0.47
##    6    500    0.76      0.47
##    6    700    0.77      0.47
##    7    100    0.77      0.48
##    7    200    0.77      0.47
##    7    500    0.77      0.48
##    7    700    0.77      0.47
##    8    100    0.77      0.48
##    8    200    0.76      0.47
##    8    500    0.77      0.48
##    8    700    0.77      0.48
##    9    100    0.76      0.47
##    9    200    0.77      0.48
##    9    500    0.77      0.48
##    9    700    0.77      0.48
```

```
##   10    100    0.76      0.47
##   10    200    0.77      0.48
##   10    500    0.77      0.48
##   10    700    0.77      0.47
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 9 and ntree = 200.
```

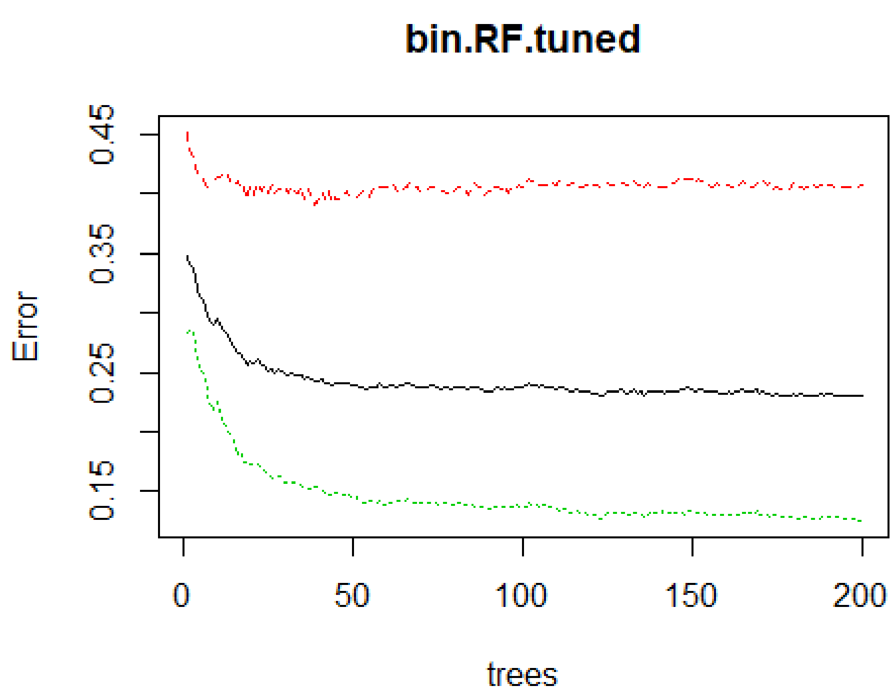
Binary Random Forest: Accuracy is highest for mtry = 9 and ntree = 200, with corresponding accuracy around ~77%.

```
set.seed(111)

k2 = 10

n = floor(nrow(binaryrf.data)/k2)
accuracy.vect.bin = rep(NA,k2)

for (i in 1:k2) {

  s3 = ((i-1) * n+1)
  s4 = (i*n)
  subset = s3:s4

  binrfcv.train = binaryrf.data[-subset,]
  binrfcv.test = binaryrf.data[subset,]

  Bin.tuned.RandForest <- randomForest(NPS_Status~.-NPS_Status, data = binrfcv.train, mtr
y = 9, ntree = 200 )

  binRF.pred <- predict(Bin.tuned.RandForest,
                           newdata = binrfcv.test, type = "class")

  accuracy.vect.bin[i] <- (confusionMatrix(binRF.pred, binrfcv.test$NPS_Status))$overall[
1]

 print(paste("Accuracy for fold", i, ":", accuracy.vect.bin[i]))

    }

## [1] "Accuracy for fold 1 : 0.786915887850467"
## [1] "Accuracy for fold 2 : 0.779439252336449"
## [1] "Accuracy for fold 3 : 0.758878504672897"
## [1] "Accuracy for fold 4 : 0.794392523364486"
## [1] "Accuracy for fold 5 : 0.734579439252336"
## [1] "Accuracy for fold 6 : 0.753271028037383"
## [1] "Accuracy for fold 7 : 0.777570093457944"
## [1] "Accuracy for fold 8 : 0.790654205607477"
## [1] "Accuracy for fold 9 : 0.779439252336449"
## [1] "Accuracy for fold 10 : 0.747663551401869"

print(paste(" Average Accuracy for binary Random Forest :", mean(accuracy.vect.bin)))

## [1] " Average Accuracy for binary Random Forest : 0.770280373831776"
```

Average Accuracy from 10-Fold cross Validation for binary Random Forest : 0.770280373831776

Retrain the model with best parameters

```
set.seed(123)

#Retraining the model with best values of mtry and ntree

bin.RF.tuned <- randomForest(NPS_Status ~. -NPS_Status,
                      data=binaryrf.train,
                      importance = TRUE,
                      mtry = 9,
                      ntree = 200)
print(bin.RF.tuned)

##
## Call:
##  randomForest(formula = NPS_Status ~ . - NPS_Status, data = binaryrf.train,      impor
tance = TRUE, mtry = 9, ntree = 200)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 23%
## Confusion matrix:
##           Detractor Promotor class.error
## Detractor      1095      754        0.41
## Promotor        394     2746        0.13

plot(bin.RF.tuned)
```



Prediction & Confusion Matrix for Binary class Random Forest on Test Data

```
# Making final prediction on test data

RFtest.pred <- predict(bin.RF.tuned, binaryrf.test, type = "prob")

confusionMatrix(predict(bin.RF.tuned, newdata= binaryrf.test,
                        type = "class"),
                binaryrf.test$NPS_Status)

## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Detractor Promotor
##   Detractor        86       20
##   Promotor         75      183
##
##                Accuracy : 0.739
##                  95% CI : (0.691, 0.783)
##     No Information Rate : 0.558
##     P-Value [Acc > NIR] : 6.40e-13
##
##                   Kappa : 0.452
##
##  Mcnemar's Test P-Value : 3.02e-08
##
##             Sensitivity : 0.534
##             Specificity : 0.901
##          Pos Pred Value : 0.811
##          Neg Pred Value : 0.709
##              Prevalence : 0.442
##          Detection Rate : 0.236
##    Detection Prevalence : 0.291
##       Balanced Accuracy : 0.718
##
##        'Positive' Class : Detractor
##
```

Accuracy for Binary class Random Forest on Test Data : 73.9%

Important variables from binary-class Random Forest Model:

```
BinrfImportant <- importance(bin.RF.tuned, type = 2)

BinRFImportance <- data.frame(Variables = row.names(BinrfImportant),
            Importance = round(BinrfImportant[ ,'MeanDecreaseGini'],2))

BinRFImportance <- BinRFImportance[order((BinRFImportance$Importance),decreasing = TRUE),
]

head(BinRFImportance, 20)

##                                                      Variables Importance
## AgeYrs                                                  AgeYrs        186
## CE_VALUEFORMONEY                              CE_VALUEFORMONEY        163
## CE_CSAT                                                CE_CSAT        161
## Estimatedcost                                    Estimatedcost        149
## BedCategory                                        BedCategory        121
## LengthofStay                                      LengthofStay        118
## Department                                          Department        107
## InsPayorcategory                              InsPayorcategory         72
## AE_ATTENDEEFOOD                                AE_ATTENDEEFOOD         66
## DP_DISCHARGETIME                              DP_DISCHARGETIME         64
## DP_DISCHARGEPROCESS                        DP_DISCHARGEPROCESS         61
## CE_ACCESSIBILITY                              CE_ACCESSIBILITY         60
## FNB_FOODDELIVERYTIME                      FNB_FOODDELIVERYTIME         48
## AD_TARRIFFPACKAGESEXPLAINATION AD_TARRIFFPACKAGESEXPLAINATION         48
## FNB_FOODQUALITY                                FNB_FOODQUALITY         47
## DP_DISCHARGEQUERIES                        DP_DISCHARGEQUERIES         44
## INR_ROOMAMBIENCE                              INR_ROOMAMBIENCE         43
## AD_TIME                                                AD_TIME         40
## INR_ROOMPEACE                                    INR_ROOMPEACE         37
## INR_ROOMCLEANLINESS                        INR_ROOMCLEANLINESS         36
```

Final Results for Random Forest for Binary Class Classification:

CV for Parameter Tuning	10-fold CV for model evaluation	Accuracy on Test Data
mtry = 9, ntree = 200	**Average Accuracy = 77.02%**	**Accuracy = 73.9%**

Ada Boost for Binary Classification:

```
binAdaTrain <- binaryTrain %>% select(-CE_NPS)

binAdaTest <- binaryTest %>% select(-CE_NPS)

binAdaData <- rbind(binAdaTrain,binAdaTest)

library(caret)
binAda_nums <- dplyr::select_if(binAdaData, is.numeric)
binAda_cat <- dplyr::select_if(binAdaData, is.factor)

# Creating dummy variables for categorical variables

var_onehot <- c('MaritalStatus','Sex','BedCategory','Department', "InsPayorcategory", "St
ate", "Country", "STATEZONE")

# One Hot Encoding

dummy <- dummyVars(" ~ .", data = binAda_cat[,var_onehot])

dummy_cat <- data.frame(predict(dummy, newdata = binAda_cat[,var_onehot]))

 new.binAdaData <- cbind(binAda_nums,dummy_cat,binAda_cat$NPS_Status)

names(new.binAdaData)[names(new.binAdaData) =="binAda_cat$NPS_Status"] <- "NPS_Status"

binAdaTrain <- new.binAdaData[1:4989,]

binAdaTest <- new.binAdaData[4990:5353,]

# Setting the number of levels of factor variables in Training & Test data as same

common <- intersect(names(binAdaTrain), names(binAdaTest))
for (p in common) {
  if (class(binAdaTrain[[p]]) == "factor") {
    levels(binAdaTest[[p]]) <- levels(binAdaTrain[[p]]) } }
```

Ada Boost for Binary Class Classification:

Cross Validation for Parameter Tuning - Cross validation is conducted to find the best value of mfinal (no. of iterations), Complexity parameter (cp) and maxdepth.

```
# Find the best model with the best mfinal, cp and maxdepth, via cross-validations
set.seed(111)
bin.best.mfinal <- NA
bin.best.cp <- NA
bin.best.maxdepth <- NA

highestbin.accuracy <- 0
for (m.final1 in c(10,20)) {
  for (comp.p1 in c(0.005,0.001,0.01)) {
    for (maxdepth1 in c(10, 20,30)){
    binaryAdaBoost <- boosting(NPS_Status ~ ., data = binAdaTrain,
                            mfinal = m.final1,
```

```
                          control = rpart.control(maxdepth =maxdepth1,
                                                   cp=comp.p1))

    binpred.best <- as.factor((predict.boosting(binaryAdaBoost,binAdaTrain))$class)

    #levels(multipred.best)
    #levels(multiAdaTrain$NPS_Status)

    binfold.accuracy <- (confusionMatrix(binpred.best, binAdaTrain$NPS_Status)$overall)[1
]

    cat("Results in the Binary classification for mfinal=",m.final1," : ", "Complexity pa
rameter = ",comp.p1,":", "and maxdepth = ",maxdepth1,":", "Accuracy = ",binfold.accuracy,
"\n",sep="")

    if(binfold.accuracy > highest.accuracy){
      highestbin.accuracy <- binfold.accuracy
      multi.best.mfinal <- m.final1
      multi.best.cp <- comp.p1
      multi.best.maxdepth <- maxdepth1

    }
  }
 }
}

## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.005:and
maxdepth = 10:Accuracy = 0.78
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.005:and
maxdepth = 20:Accuracy = 0.79
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.005:and
maxdepth = 30:Accuracy = 0.79
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.001:and
maxdepth = 10:Accuracy = 0.83
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.001:and
maxdepth = 20:Accuracy = 0.93
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.001:and
maxdepth = 30:Accuracy = 0.93
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.01:and m
axdepth = 10:Accuracy = 0.76
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.01:and m
axdepth = 20:Accuracy = 0.77
## Results in the Binary classification for mfinal=10 : Complexity parameter = 0.01:and m
axdepth = 30:Accuracy = 0.77
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.005:and
maxdepth = 10:Accuracy = 0.79
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.005:and
maxdepth = 20:Accuracy = 0.8
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.005:and
maxdepth = 30:Accuracy = 0.8
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.001:and
maxdepth = 10:Accuracy = 0.86
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.001:and
maxdepth = 20:Accuracy = 0.98
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.001:and
maxdepth = 30:Accuracy = 1
```

```
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.01:and m
axdepth = 10:Accuracy = 0.77
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.01:and m
axdepth = 20:Accuracy = 0.78
## Results in the Binary classification for mfinal=20 : Complexity parameter = 0.01:and m
axdepth = 30:Accuracy = 0.77

cat("For Binary Classification:", "\n")

## For Binary Classification:

cat("Best mfinal (number of iterations) is:",m.final,"\n")

## Best mfinal (number of iterations) is: 20

cat("Best complexity parameter is:",comp.p,"\n")

## Best complexity parameter is: 0.001

cat("Best maxdepth is:",maxdepth,"\n")

## Best maxdepth is: 30

cat("Best accuracy is:",highestbin.accuracy,"\n")

## Best accuracy is: 1
```

For Adaboost Binary-class Classification:
Best mfinal = 20
Best Complexity Parameter = 0.001
Best maxdepth = 30
Best accuracy = 1

I am using complexity parameter as 0.005, as a lower complexity parameter may result in overfitting and cause greater test error.

Retraining the model with best parameters obtained from cross validation

```
set.seed(111)
  library("adabag")

bestbinary.adaboost <- boosting(NPS_Status ~ ., data = binAdaTrain, mfinal = 20, control
= rpart.control(maxdepth = 30, cp=0.005 ))
```

Prediction & Confusion Matrix for Binary class Ada-boost on Training Data

```
binary.predboosting.tr <- as.factor(predict.boosting(bestbinary.adaboost,
                                      newdata = binAdaTrain)$class)
#length(multi.predboosting)
#length(multiAdaTest$NPS_Status)

confusionMatrix(binary.predboosting.tr, binAdaTrain$NPS_Status)

## Confusion Matrix and Statistics
## 
##             Reference
## Prediction  Detractor Promotor
##   Detractor      1134      258
##   Promotor        715     2882
## 
##                Accuracy : 0.805
##                  95% CI : (0.794, 0.816)
##     No Information Rate : 0.629
##     P-Value [Acc > NIR] : <2e-16
## 
##                   Kappa : 0.56
## 
##  Mcnemar's Test P-Value : <2e-16
## 
##             Sensitivity : 0.613
##             Specificity : 0.918
##          Pos Pred Value : 0.815
##          Neg Pred Value : 0.801
##              Prevalence : 0.371
##          Detection Rate : 0.227
##    Detection Prevalence : 0.279
##       Balanced Accuracy : 0.766
## 
##        'Positive' Class : Detractor
## 
```

Accuracy for Binary class Ada-Boost on Training Data : 80.5%

Prediction & Confusion Matrix for Binary class Ada-boost on Test Data

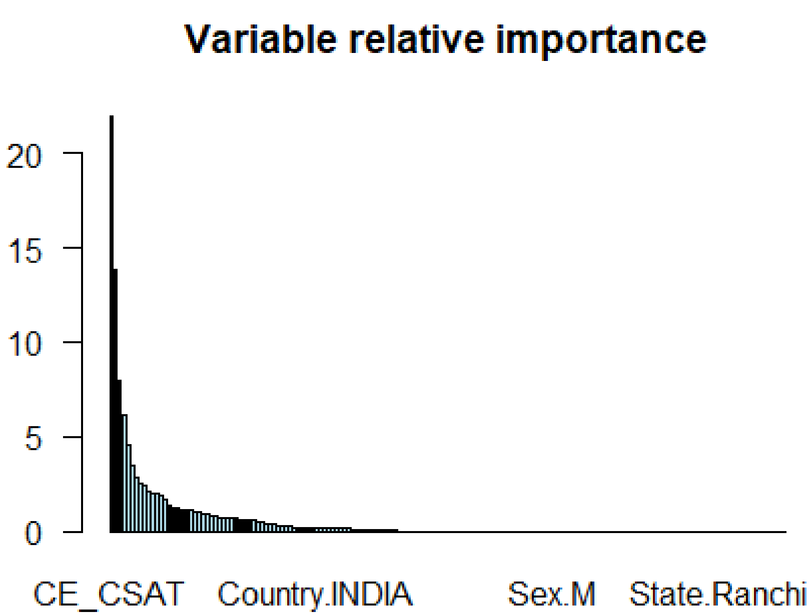
```
binary.predboosting <- as.factor(predict.boosting(bestbinary.adaboost,
                                      newdata = binAdaTest)$class)
#length(multi.predboosting)
#length(multiAdaTest$NPS_Status)

confusionMatrix(binary.predboosting, binAdaTest$NPS_Status)

## Confusion Matrix and Statistics
## 
##             Reference
## Prediction  Detractor Promotor
##   Detractor        87       29
##   Promotor         74      174
## 
```

```
##                Accuracy : 0.717
##                  95% CI : (0.668, 0.763)
##     No Information Rate : 0.558
##     P-Value [Acc > NIR] : 2.83e-10
##
##                   Kappa : 0.409
##
##  Mcnemar's Test P-Value : 1.45e-05
##
##             Sensitivity : 0.540
##             Specificity : 0.857
##          Pos Pred Value : 0.750
##          Neg Pred Value : 0.702
##              Prevalence : 0.442
##          Detection Rate : 0.239
##    Detection Prevalence : 0.319
##       Balanced Accuracy : 0.699
##
##        'Positive' Class : Detractor
##
```

Accuracy for Binary class Random Forest on Test Data : 71.7%

Important Variables in Binary class Ada-boost Model

```
importanceplot(bestbinary.adaboost)
```




```
BinadaImportant <- bestbinary.adaboost$importance

head(sort(BinadaImportant, decreasing = TRUE),30)

##                            CE_CSAT                     CE_VALUEFORMONEY
##                             21.934                               13.826
##                     AE_ATTENDEEFOOD                               AgeYrs
##                              7.926                                6.190
##                   CE_ACCESSIBILITY       AD_TARRIFFPACKAGESEXPLAINATION
##                              4.502                                3.466
```

```
##                      Estimatedcost                      LengthofStay
##                              2.831                             2.547
##                         DOC_VISITS                BedCategory.GENERAL
##                              2.369                             2.057
##                    FNB_FOODQUALITY                  AD_STAFFATTITUDE
##                              2.018                             1.990
##                   INR_ROOMAMBIENCE               FNB_FOODDELIVERYTIME
##                              1.851                             1.636
##               INR_ROOMCLEANLINESS                DP_DISCHARGEPROCESS
##                              1.388                             1.279
##                AE_PATIENTSTATUSINFO            NS_NURSEPROACTIVENESS
##                              1.239                             1.154
##                    AE_ATTENDEECARE                      DOC_ATTITUDE
##                              1.143                             1.120
##                      INR_ROOMPEACE                  DP_DISCHARGETIME
##                              1.045                             0.998
##                     Department.GEN               NS_CALLBELLRESPONSE
##                              0.959                             0.880
##                 INR_ROOMEQUIPMENT             EM_IMMEDIATEATTENTION
##                              0.857                             0.841
##               DP_DISCHARGEQUERIES                 FNB_STAFFATTITUDE
##                              0.718                             0.717
##              OVS_SECURITYATTITUDE                             Sex.F
##                              0.700                             0.670
##      OVS_OVERALLSTAFFPROMPTNESS                           AD_TIME
##                              0.659                             0.633
##                        EM_NURSING                NS_NURSESATTITUDE
##                              0.629                             0.590
```

Final Results from Binary Class Ada-Boost:

Parameter Tuning	Training accuracy with best parameters	Accuracy on Test Data
mfinal = 20, cp = 0.001, maxdepth = 30	**Average Accuracy = 80.5%**	**Accuracy = 71.7%**

8. Check the effect of balancing methods (under-sampling, over-sampling, and SMOTE (Synthethic Minority Oversampling) on the performance of ensemble methods.

Balancing Data using SMOTE:

Balancing the train data using SMOTE function from DMwR library. SMOTE uses K-nearest neighbour method to generate new samples, as to increase the minority class rows and decrease the majority class rows in the data.

```
Samplingtrain <- binaryTrain
Samplingtest <- binaryTest

Samplingtrain %>%
  group_by(NPS_Status) %>%
  summarise(count = n())

## # A tibble: 2 x 2
##   NPS_Status count
```

```
##   <fct>      <int>
## 1 Detractor   1849
## 2 Promotor    3140
```

```
library(DMwR)

## Smote : Synthetic Minority Oversampling Technique To Handle Class Imbalance In Binary
Classification

SMOTE.balanced <- SMOTE(NPS_Status ~., as.data.frame(Samplingtrain),
                       perc.under = 170,
                       perc.over = 180 , k = 5)
as.data.frame(table(SMOTE.balanced$NPS_Status))
```

```
##        Var1 Freq
## 1 Detractor 3698
## 2  Promotor 3143
```

Balancing Data using Under Sampling:

```
library(ROSE)
```

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

```
underSample <- ovun.sample(NPS_Status ~., as.data.frame(Samplingtrain),
                           method = "under", N=4000)$data
underSample %>% group_by(NPS_Status) %>% count()
```

```
## # A tibble: 2 x 2
## # Groups:   NPS_Status [2]
##   NPS_Status     n
##   <fct>      <int>
## 1 Promotor    2151
## 2 Detractor   1849
```

Balancing Data using Over Sampling:

```
library(ROSE)

overSample <- ovun.sample(NPS_Status ~., as.data.frame(Samplingtrain),
                          method = "over", N=6000)$data
overSample %>% group_by(NPS_Status) %>% count()
```

```
## # A tibble: 2 x 2
## # Groups:   NPS_Status [2]
##   NPS_Status     n
##   <fct>      <int>
## 1 Promotor    3140
## 2 Detractor   2860
```

Random Forest with SMOTE data:

```
rftrain.smote <- SMOTE.balanced %>% select(-CE_NPS)
rftest.smote <- binaryTest %>% select(-CE_NPS)

rf.smote <- rbind(rftrain.smote,rftest.smote)

rf.smote <- rf.smote %>% select(-State)

#rfsmote_cat <- dplyr::select_if(rf.smote, is.factor)
#sapply(rfsmote_cat, function(x) length(unique(x)))

set.seed(111)
k2 = 10
n = floor(nrow(rf.smote)/k2)
accuracy.vect.smote = rep(NA,k2)

for (i in 1:k2) {

  s5 = ((i-1) * n+1)
  s6 = (i*n)
  subset = s5:s6

  smoterfcv.train = rf.smote[-subset,]
  smoterfcv.test = rf.smote[subset,]

  smote.tuned.RandForest <- randomForest(NPS_Status~.-NPS_Status, data = smoterfcv.train,
mtry = 9, ntree = 200 )

  smoteRF.pred <- predict(smote.tuned.RandForest,
                          newdata = smoterfcv.test, type = "class")

  accuracy.vect.smote[i] <- (confusionMatrix(smoteRF.pred, smoterfcv.test$NPS_Status))$ov
erall[1]

 print(paste("Accuracy for fold", i, ":", accuracy.vect.smote[i]))

    }

## [1] "Accuracy for fold 1 : 0.891666666666667"
## [1] "Accuracy for fold 2 : 0.875"
## [1] "Accuracy for fold 3 : 0.894444444444444"
## [1] "Accuracy for fold 4 : 0.902777777777778"
## [1] "Accuracy for fold 5 : 0.766666666666667"
## [1] "Accuracy for fold 6 : 0.595833333333333"
## [1] "Accuracy for fold 7 : 0.716666666666667"
## [1] "Accuracy for fold 8 : 0.977777777777778"
## [1] "Accuracy for fold 9 : 0.970833333333333"
## [1] "Accuracy for fold 10 : 0.843055555555556"

print(paste(" Average Accuracy for Smote Random Forest :", mean(accuracy.vect.smote)))

## [1] " Average Accuracy for Smote Random Forest : 0.843472222222222"
```

Accuracy for Random Forest through Cross Validation, with SMOTE: 84.34%

Important Variables for Random Forest with SMOTE

```
smoterfImportant <- importance(smote.tuned.RandForest, type = 2)

smoteRFImportance <- data.frame(Variables = row.names(smoterfImportant),
            Importance = round(smoterfImportant[ ,'MeanDecreaseGini'],2))

smoteRFImportance <- smoteRFImportance[order((smoteRFImportance$Importance),decreasing =
TRUE), ]

head(smoteRFImportance, 20)

##                                                    Variables Importance
## CE_CSAT                                              CE_CSAT        306
## CE_VALUEFORMONEY                            CE_VALUEFORMONEY        249
## AgeYrs                                                AgeYrs        201
## Estimatedcost                                  Estimatedcost        173
## DP_DISCHARGETIME                            DP_DISCHARGETIME        139
## LengthofStay                                    LengthofStay        137
## BedCategory                                      BedCategory        130
## CE_ACCESSIBILITY                            CE_ACCESSIBILITY        130
## Department                                        Department        127
## DP_DISCHARGEPROCESS                      DP_DISCHARGEPROCESS        118
## AE_ATTENDEEFOOD                              AE_ATTENDEEFOOD        108
## AD_TARRIFFPACKAGESEXPLAINATION AD_TARRIFFPACKAGESEXPLAINATION         98
## InsPayorcategory                            InsPayorcategory         86
## DP_DISCHARGEQUERIES                      DP_DISCHARGEQUERIES         71
## FNB_FOODQUALITY                              FNB_FOODQUALITY         67
## AD_TIME                                              AD_TIME         65
## STATEZONE                                          STATEZONE         63
## FNB_FOODDELIVERYTIME                    FNB_FOODDELIVERYTIME         56
## INR_ROOMPEACE                                  INR_ROOMPEACE         54
## FNB_DIETICIAN                                  FNB_DIETICIAN         49
```

Random Forest with Under Sampled data:

```
rftrain.us <- underSample %>% select(-CE_NPS)
rftest.us <- binaryTest %>% select(-CE_NPS)

rf.us <- rbind(rftrain.us,rftest.us)

rf.us <- rf.us %>% select(-State)

#rfsmote_cat <- dplyr::select_if(rf.smote, is.factor)
#sapply(rfsmote_cat, function(x) length(unique(x)))

set.seed(111)
k2 = 10

n = floor(nrow(rf.us)/k2)
accuracy.vect.us = rep(NA,k2)

for (i in 1:k2) {


  s7 = ((i-1) * n+1)
```

```
  s8 = (i*n)
  subset = s7:s8

  USrfcv.train = rf.us[-subset,]
  USrfcv.test = rf.us[subset,]

  US.tuned.RandForest <- randomForest(NPS_Status~.-NPS_Status, data = USrfcv.train, mtry
= 9, ntree = 200 )

  usRF.pred <- predict(US.tuned.RandForest,
                            newdata = USrfcv.test, type = "class")

  accuracy.vect.us[i] <- (confusionMatrix(usRF.pred, USrfcv.test$NPS_Status))$overall[1]

 print(paste("Accuracy for fold", i, ":", accuracy.vect.us[i]))

    }

## [1] "Accuracy for fold 1 : 0.782110091743119"
## [1] "Accuracy for fold 2 : 0.779816513761468"
## [1] "Accuracy for fold 3 : 0.756880733944954"
## [1] "Accuracy for fold 4 : 0.73394495412844"
## [1] "Accuracy for fold 5 : 0.786697247706422"
## [1] "Accuracy for fold 6 : 0.600917431192661"
## [1] "Accuracy for fold 7 : 0.630733944954128"
## [1] "Accuracy for fold 8 : 0.582568807339449"
## [1] "Accuracy for fold 9 : 0.658256880733945"
## [1] "Accuracy for fold 10 : 0.717889908256881"

print(paste(" Average Accuracy for Under Sampled Random Forest :", mean(accuracy.vect.us)
))

## [1] " Average Accuracy for Under Sampled Random Forest : 0.702981651376147"
```

Accuracy for Random Forest through Cross Validation, with Under Sampled Data: 70.29%

Important Variables for Random Forest with Under Sampled Data

```
usrfImportant <- importance(US.tuned.RandForest, type = 2)

usRFImportance <- data.frame(Variables = row.names(usrfImportant),
            Importance = round(usrfImportant[ ,'MeanDecreaseGini'],2))

usRFImportance <- usRFImportance[order((usRFImportance$Importance),decreasing = TRUE), ]

head(usRFImportance, 20)

##                                                          Variables Importance
## AgeYrs                                                      AgeYrs        158
## CE_CSAT                                                    CE_CSAT        141
## Estimatedcost                                        Estimatedcost        126
## CE_VALUEFORMONEY                                  CE_VALUEFORMONEY        118
## LengthofStay                                          LengthofStay         99
## BedCategory                                            BedCategory         97
## Department                                              Department         92
## AE_ATTENDEEFOOD                                    AE_ATTENDEEFOOD         70
## InsPayorcategory                                  InsPayorcategory         63
```

```
## DP_DISCHARGETIME                                      DP_DISCHARGETIME         53
## DP_DISCHARGEPROCESS                                DP_DISCHARGEPROCESS         51
## AD_TARRIFFPACKAGESEXPLAINATION AD_TARRIFFPACKAGESEXPLAINATION         51
## CE_ACCESSIBILITY                                      CE_ACCESSIBILITY         51
## FNB_FOODQUALITY                                        FNB_FOODQUALITY         45
## FNB_FOODDELIVERYTIME                              FNB_FOODDELIVERYTIME         42
## DP_DISCHARGEQUERIES                                DP_DISCHARGEQUERIES         37
## AD_TIME                                                        AD_TIME         33
## FNB_DIETICIAN                                            FNB_DIETICIAN         32
## STATEZONE                                                    STATEZONE         30
## INR_ROOMCLEANLINESS                                INR_ROOMCLEANLINESS         29
```

Random Forest with Over Sampled data:

```
rftrain.os <- overSample %>% select(-CE_NPS)
rftest.os <- binaryTest %>% select(-CE_NPS)

rf.os <- rbind(rftrain.os,rftest.os)

rf.os <- rf.os %>% select(-State)

#rfsmote_cat <- dplyr::select_if(rf.smote, is.factor)
#sapply(rfsmote_cat, function(x) length(unique(x)))

set.seed(111)
k2 = 10
n = floor(nrow(rf.os)/k2)
accuracy.vect.os = rep(NA,k2)

for (i in 1:k2) {

  s9 = ((i-1) * n+1)
  s10 = (i*n)
  subset = s9:s10

  OSrfcv.train = rf.os[-subset,]
  OSrfcv.test = rf.os[subset,]

  OS.tuned.RandForest <- randomForest(NPS_Status~.-NPS_Status, data = OSrfcv.train, mtry
= 9, ntree = 500 )

  osRF.pred <- predict(OS.tuned.RandForest,
                            newdata = OSrfcv.test, type = "class")

  accuracy.vect.os[i] <- (confusionMatrix(osRF.pred, OSrfcv.test$NPS_Status))$overall[1]

 print(paste("Accuracy for fold", i, ":", accuracy.vect.os[i]))

    }

## [1] "Accuracy for fold 1 : 0.861635220125786"
## [1] "Accuracy for fold 2 : 0.845911949685535"
## [1] "Accuracy for fold 3 : 0.842767295597484"
## [1] "Accuracy for fold 4 : 0.814465408805031"
## [1] "Accuracy for fold 5 : 0.844339622641509"
## [1] "Accuracy for fold 6 : 0.860062893081761"
```

```
## [1] "Accuracy for fold 7 : 0.841194968553459"
## [1] "Accuracy for fold 8 : 0.828616352201258"
## [1] "Accuracy for fold 9 : 0.844339622641509"
## [1] "Accuracy for fold 10 : 0.808176100628931"
```

```
print(paste(" Average Accuracy for Over Sampled Random Forest :", mean(accuracy.vect.os))
)
```

```
## [1] " Average Accuracy for Over Sampled Random Forest : 0.839150943396226"
```

Accuracy for Random Forest through Cross-Validation, with Over Sampled Data: 83.92%

Important Variables for Random Forest with Over Sampled Data

```
osrfImportant <- importance(OS.tuned.RandForest, type = 2)

osRFImportance <- data.frame(Variables = row.names(osrfImportant),
            Importance = round(osrfImportant[ ,'MeanDecreaseGini'],2))

osRFImportance <- osRFImportance[order((osRFImportance$Importance),decreasing = TRUE), ]

head(osRFImportance, 20)
```

```
##                                                      Variables Importance
## AgeYrs                                                  AgeYrs        232
## CE_CSAT                                                CE_CSAT        211
## Estimatedcost                                    Estimatedcost        183
## CE_VALUEFORMONEY                              CE_VALUEFORMONEY        168
## BedCategory                                        BedCategory        152
## LengthofStay                                      LengthofStay        144
## Department                                          Department        136
## InsPayorcategory                              InsPayorcategory         96
## AD_TARRIFFPACKAGESEXPLAINATION AD_TARRIFFPACKAGESEXPLAINATION         91
## DP_DISCHARGETIME                              DP_DISCHARGETIME         90
## CE_ACCESSIBILITY                              CE_ACCESSIBILITY         80
## AE_ATTENDEEFOOD                                AE_ATTENDEEFOOD         76
## DP_DISCHARGEPROCESS                        DP_DISCHARGEPROCESS         68
## DP_DISCHARGEQUERIES                        DP_DISCHARGEQUERIES         62
## FNB_FOODQUALITY                                FNB_FOODQUALITY         62
## FNB_FOODDELIVERYTIME                      FNB_FOODDELIVERYTIME         53
## STATEZONE                                            STATEZONE         47
## AD_TIME                                                AD_TIME         46
## INR_ROOMAMBIENCE                              INR_ROOMAMBIENCE         43
## Sex                                                        Sex         43
```

Ada Boost with SMOTE data:

```
Adatrain.smote <- SMOTE.balanced %>% select(-CE_NPS)
Adatest.smote <- binaryTest %>% select(-CE_NPS)

Ada.smote <- rbind(Adatrain.smote,Adatest.smote)

#Converting categorical variables into dummy numerica variables

Ada.smote_nums <- dplyr::select_if(Ada.smote, is.numeric)
Ada.smote_cat <- dplyr::select_if(Ada.smote, is.factor)
```

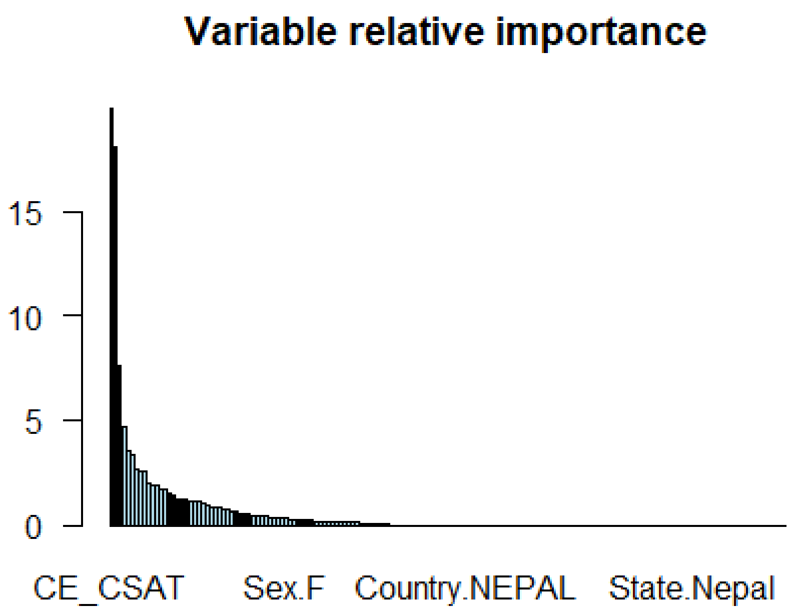
```
var_onehot <- c('MaritalStatus','Sex','BedCategory','Department', "InsPayorcategory", "St
ate", "Country", "STATEZONE")

# One Hot Encoding

dummys1 <- dummyVars(" ~ .", data = Ada.smote_cat[,var_onehot])
dummy_cats1 <- data.frame(predict(dummys1, newdata = Ada.smote_cat[,var_onehot]))
 new.Ada.smote <- cbind(Ada.smote_nums,dummy_cats1,Ada.smote_cat$NPS_Status)
names(new.Ada.smote)[names(new.Ada.smote) =="Ada.smote_cat$NPS_Status"] <- "NPS_Status"
Adatrain.smote <- new.Ada.smote[1:6841,]
Adatest.smote <- new.Ada.smote[6842:7205,]
Ada.smote <- rbind(Adatrain.smote,Adatest.smote)

set.seed(111)
library(adabag)

k3 = 10

x = floor(nrow(Ada.smote)/k3)
accuracy.Ada.smote = rep(NA,k3)

for (i in 1:k3) {

  p1 = ((i-1) * x+1)
  p2 = (i*x)
  subset = p1:p2

  smoteAdacv.train = Ada.smote[-subset,]
  smoteAdacv.test = Ada.smote[subset,]

 smote.tuned.Ada <- boosting(NPS_Status ~ ., data = smoteAdacv.train, mfinal = 20, contro
l = rpart.control(maxdepth = 30, cp=0.005 ))

smote.predboosting <- as.factor(predict.boosting(smote.tuned.Ada,
                                      newdata = smoteAdacv.test)$class)
#length(multi.predboosting)
#length(multiAdaTest$NPS_Status)

accuracy.Ada.smote[i] <- (confusionMatrix(smote.predboosting, smoteAdacv.test$NPS_Status)
)$overall[1]

 print(paste("Accuracy for fold", i, ":", accuracy.Ada.smote[i]))

    }

## [1] "Accuracy for fold 1 : 0.802777777777778"
## [1] "Accuracy for fold 2 : 0.793055555555556"
## [1] "Accuracy for fold 3 : 0.783333333333333"
## [1] "Accuracy for fold 4 : 0.823611111111111"
## [1] "Accuracy for fold 5 : 0.683333333333333"
## [1] "Accuracy for fold 6 : 0.55"
## [1] "Accuracy for fold 7 : 0.604166666666667"
## [1] "Accuracy for fold 8 : 0.951388888888889"
## [1] "Accuracy for fold 9 : 0.963888888888889"
## [1] "Accuracy for fold 10 : 0.836111111111111"
```

```r
print(paste(" Average Accuracy for SMOTE Sampled Ada Boost Model :", mean(accuracy.Ada.smote)))

## [1] " Average Accuracy for SMOTE Sampled Ada Boost Model : 0.779166666666667"
```

Accuracy for Ada Boost through Cross-Validation with SMOTE Data: 77.91%

Important Variables for Ada Boost with SMOTE Data

```r
importanceplot(smote.tuned.Ada)
```




```r
smoteadaImportant <- smote.tuned.Ada$importance

head(sort(smoteadaImportant, decreasing = TRUE),20)

##                      CE_CSAT                CE_VALUEFORMONEY
##                         19.9                            18.1
##                 LengthofStay                          AgeYrs
##                          7.6                             4.8
##            AD_STAFFATTITUDE                CE_ACCESSIBILITY
##                          3.6                             3.4
##                Estimatedcost AD_TARRIFFPACKAGESEXPLAINATION
##                          2.7                             2.6
##              FNB_FOODQUALITY            INR_ROOMCLEANLINESS
##                          2.5                             2.0
##            DP_DISCHARGETIME               INR_ROOMAMBIENCE
##                          1.9                             1.9
##         DP_DISCHARGEQUERIES                  FNB_DIETICIAN
##                          1.7                             1.7
##           INR_ROOMEQUIPMENT               AE_ATTENDEECARE
##                          1.5                             1.4
##         DP_DISCHARGEPROCESS           NS_NURSEPROACTIVENESS
##                          1.2                             1.2
##        AE_PATIENTSTATUSINFO                      DOC_VISITS
##                          1.2                             1.1
```

Ada Boost with under sampled data:

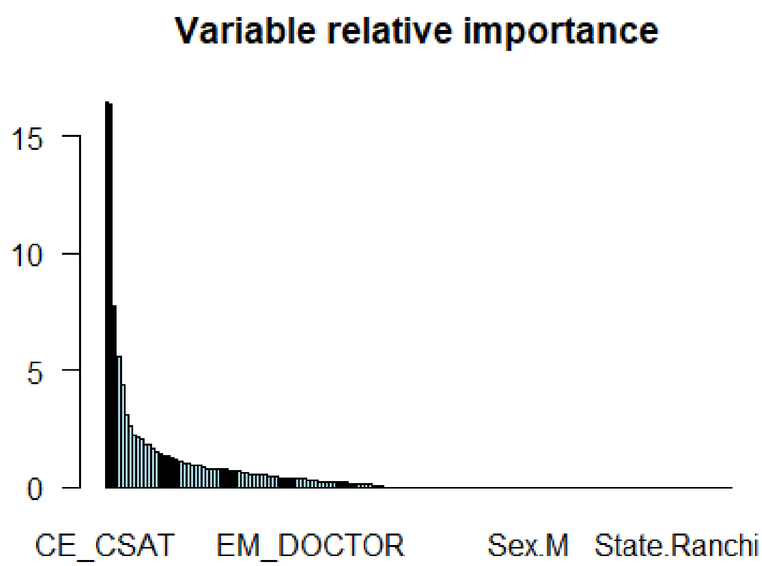
```
Adatrain.us <- underSample %>% select(-CE_NPS)
Adatest.us <- binaryTest %>% select(-CE_NPS)

Ada.us <- rbind(Adatrain.us,Adatest.us)

#Converting categorical variables into dummy numerica variables
Ada.us_nums <- dplyr::select_if(Ada.us, is.numeric)
Ada.us_cat <- dplyr::select_if(Ada.us, is.factor)

var_onehot <- c('MaritalStatus','Sex','BedCategory','Department', "InsPayorcategory", "St
ate", "Country", "STATEZONE")

# One Hot Encoding
dummys2 <- dummyVars(" ~ .", data = Ada.us_cat[,var_onehot])
dummy_cats2 <- data.frame(predict(dummys2, newdata = Ada.us_cat[,var_onehot]))
 new.Ada.us <- cbind(Ada.us_nums,dummy_cats2,Ada.us_cat$NPS_Status)
names(new.Ada.us)[names(new.Ada.us) =="Ada.us_cat$NPS_Status"] <- "NPS_Status"
Adatrain.us <- new.Ada.us[1:4000,]
Adatest.us <- new.Ada.us[4001:4364,]
Ada.us <- rbind(Adatrain.us,Adatest.us)

library(adabag)

k3 = 10

y = floor(nrow(Ada.us)/k3)
accuracy.Ada.us = rep(NA,k3)

for (i in 1:k3) {

  p3 = ((i-1) * y+1)
  p4 = (i*y)
  subset = p3:p4

  USAdacv.train = Ada.us[-subset,]
  USAdacv.test = Ada.us[subset,]


 US.tuned.Ada <- boosting(NPS_Status ~ ., data = USAdacv.train, mfinal = 20, control = rp
art.control(maxdepth = 30, cp=0.005 ))

US.predboosting <- as.factor(predict.boosting(US.tuned.Ada,
                                      newdata = USAdacv.test)$class)
#length(multi.predboosting)
#length(multiAdaTest$NPS_Status)

accuracy.Ada.us[i] <- (confusionMatrix(US.predboosting, USAdacv.test$NPS_Status))$overall
[1]

 print(paste("Accuracy for fold", i, ":", accuracy.Ada.us[i]))

    }

## [1] "Accuracy for fold 1 : 0.76605504587156"
```

```
## [1] "Accuracy for fold 2 : 0.729357798165138"
## [1] "Accuracy for fold 3 : 0.731651376146789"
## [1] "Accuracy for fold 4 : 0.731651376146789"
## [1] "Accuracy for fold 5 : 0.740825688073395"
## [1] "Accuracy for fold 6 : 0.564220183486238"
## [1] "Accuracy for fold 7 : 0.594036697247706"
## [1] "Accuracy for fold 8 : 0.594036697247706"
## [1] "Accuracy for fold 9 : 0.642201834862385"
## [1] "Accuracy for fold 10 : 0.715596330275229"
print(paste(" Average Accuracy for Under Sampled Ada Boosted model :", mean(accuracy.Ada.
us)))

## [1] " Average Accuracy for Under Sampled Ada Boosted model : 0.680963302752294"
```

Accuracy for Ada Boost through Cross-Validation with Under Sampled Data: 68.09%

Important Variables for Ada Boost with Under Sampled Data

```
importanceplot(US.tuned.Ada)
```




```
usadaImportant <- US.tuned.Ada$importance

head(sort(usadaImportant, decreasing = TRUE),20)

##                   CE_CSAT                  CE_VALUEFORMONEY
##                      16.4                              16.4
##                    AgeYrs                     Estimatedcost
##                       7.7                               5.6
##              LengthofStay                  CE_ACCESSIBILITY
##                       4.4                               3.1
##          FNB_FOODQUALITY AD_TARRIFFPACKAGESEXPLAINATION
##                       2.6                               2.2
##       FNB_FOODDELIVERYTIME                 DP_DISCHARGETIME
##                       2.1                               2.1
##          AE_ATTENDEEFOOD                             Sex.F
##                       1.8                               1.8
##       AE_PATIENTSTATUSINFO       OVS_OVERALLSTAFFPROMPTNESS
##                       1.7                               1.5
##             FNB_DIETICIAN               BedCategory.GENERAL
```

```
##                          1.4                           1.3
##          DP_DISCHARGEQUERIES          NS_NURSEPROACTIVENESS
##                          1.3                           1.3
##             INR_ROOMAMBIENCE                    DOC_VISITS
##                          1.1                           1.1
```

Ada Boost with Over Sample data:

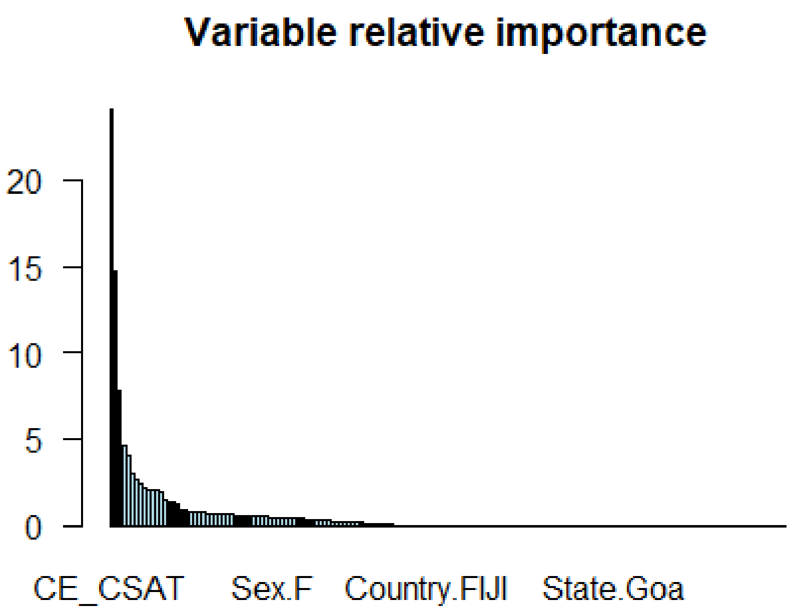
```
Adatrain.os <- overSample %>% select(-CE_NPS)
Adatest.os <- binaryTest %>% select(-CE_NPS)

Ada.os <- rbind(Adatrain.os,Adatest.os)

#Converting categorical variables into dummy numerica variables
Ada.os_nums <- dplyr::select_if(Ada.os, is.numeric)
Ada.os_cat <- dplyr::select_if(Ada.os, is.factor)

var_onehot <- c('MaritalStatus','Sex','BedCategory','Department', "InsPayorcategory", "St
ate", "Country", "STATEZONE")

# One Hot Encoding
dummys3 <- dummyVars(" ~ .", data = Ada.smote_cat[,var_onehot])
dummy_cats3 <- data.frame(predict(dummys3, newdata = Ada.os_cat[,var_onehot]))
 new.os.smote <- cbind(Ada.os_nums,dummy_cats3,Ada.os_cat$NPS_Status)
names(new.os.smote)[names(new.os.smote) =="Ada.os_cat$NPS_Status"] <- "NPS_Status"
Adatrain.os <- new.os.smote[1:6000,]
Adatest.os <- new.os.smote[6001:6364,]
Ada.os <- rbind(Adatrain.os,Adatest.os)

library(adabag)

k3 = 10

z = floor(nrow(Ada.os)/k3)
accuracy.Ada.os = rep(NA,k3)

for (i in 1:k3) {

  p5 = ((i-1) * z+1)
  p6 = (i*z)
  subset = p5:p6

  OSAdacv.train = Ada.os[-subset,]
  OSAdacv.test = Ada.os[subset,]

 OS.tuned.Ada <- boosting(NPS_Status ~ ., data = OSAdacv.train, mfinal = 20, control = rp
art.control(maxdepth = 30, cp=0.005 ))

OS.predboosting <- as.factor(predict.boosting(OS.tuned.Ada,
                                      newdata = OSAdacv.test)$class)
#levels(OS.predboosting)
#length(multiAdaTest$NPS_Status)

accuracy.Ada.os[i] <- (confusionMatrix(OS.predboosting, OSAdacv.test$NPS_Status)$overall)
[1]
```

```
 print(paste("Accuracy for fold", i, ":", accuracy.Ada.os[i]))

    }

## [1] "Accuracy for fold 1 : 0.764150943396226"
## [1] "Accuracy for fold 2 : 0.754716981132076"
## [1] "Accuracy for fold 3 : 0.754716981132076"
## [1] "Accuracy for fold 4 : 0.726415094339623"
## [1] "Accuracy for fold 5 : 0.773584905660377"
## [1] "Accuracy for fold 6 : 0.64622641509434"
## [1] "Accuracy for fold 7 : 0.660377358490566"
## [1] "Accuracy for fold 8 : 0.627358490566038"
## [1] "Accuracy for fold 9 : 0.638364779874214"
## [1] "Accuracy for fold 10 : 0.712264150943396"
print(paste(" Average Accuracy for Over Sampled Ada Boost Model :", mean(accuracy.Ada.os)
))

## [1] " Average Accuracy for Over Sampled Ada Boost Model : 0.705817610062893"
```

Accuracy for Ada Boost through Cross-Validation with Over Sampled Data: 70.58%

Important Variables for Ada Boost with Over Sampled Data

```
importanceplot(OS.tuned.Ada)
```




```
osadaImportant <- OS.tuned.Ada$importance

head(sort(osadaImportant, decreasing = TRUE),20)

##                      CE_CSAT                CE_VALUEFORMONEY
##                        24.07                           14.70
##                       AgeYrs                   Estimatedcost
##                         7.84                            4.64
##             CE_ACCESSIBILITY                  FNB_FOODQUALITY
##                         4.08                            3.03
##             INR_ROOMAMBIENCE                  AE_ATTENDEEFOOD
##                         2.61                            2.46
```

```
##           BedCategory.GENERAL            FNB_FOODDELIVERYTIME
##                          2.14                            2.09
## AD_TARRIFFPACKAGESEXPLAINATION                   LengthofStay
##                          2.07                            2.01
##           DP_DISCHARGEQUERIES             INR_ROOMCLEANLINESS
##                          1.89                            1.42
##              DP_DISCHARGETIME                AD_STAFFATTITUDE
##                          1.41                            1.37
##                    DOC_VISITS      DOC_TREATMENTEFFECTIVENESS
##                          1.25                            0.91
##                       AD_TIME                NS_NURSEPATIENCE
##                          0.88                            0.77
```

9.
Summarizing all Models:

Logistic Regression – Full Model	Logistic Regression – Stepwise Model	Random Forest – Multi Class	Ada-Boost Multi Class	Random Forest Binary Class	Ada-Boost Binary Class
Test Data Accuracy = 90.1%	**Test Data Accuracy = 90.7%**	**Cross Validation Accuracy = 72.09% Test Data Accuracy = 68.7%**	**Training Data Accuracy = 73.7% Test Data Accuracy = 67.8**	**Cross Validation Accuracy = 77.02% Test Data Accuracy = 73.9%**	**Training Data Accuracy = 80.5% Test Data Accuracy = 71.7**
48 variables used, may lead to over-fitting	**17 variables used, with higher accuracy.**				

SMOTE – Random Forest	SMOTE – Ada-Boost	Under Sampling – Random Forest	Under Sampling – Ada-Boost	Over Sampling – Random Forest	Over Sampling – Ada-Boost
84.34%	**77.91%**	**70.92%**	**68.09%**	**83.92**	**70.58%**

Objective of Manipal Health Enterprises: Reduce the proportion of Detractors.

Strategy: The constructed models help in finding out the reasons why a customer is a Detractor. The important variables identified in the models are the ones that lead to a customer becoming Promotor or Detractor. Manipal Health Enterprises can use these findings to develop a strategy that addresses those problem areas and reduce the proportion of Detractors.

To detect whether a customer is Detractor or not, and why is he/she a Detractor, Manipal Health can use Binary Class Classification.

Logistic Regression gives a good accuracy of 90.7%, with the following variables:

AgeYrs + Sex + Department + CE_ACCESSIBILITY + CE_CSAT + CE_VALUEFORMONEY + EM_NURSING + AD_TIME + AD_TARRIFFPACKAGESEXPLAINATION + INR_ROOMCLEANLINESS + FNB_FOODDELIVERYTIME +

DOC_VISITS + NS_NURSEPATIENCE + OVS_SECURITYATTITUDE + DP_DISCHARGETIME + DP_DISCHARGEQUERIES + DP_DISCHARGEPROCESS + DischargeDate

Both Ensemble methods, Random Forest and Ada-Boost provide similar results, with a Training accuracy in the range of 77-80% and Test Accuracy in the range of 71-74%.

Random Forest does a bit better on Unseen data, with an accuracy of 73.9%.

On applying SMOTE and Over Sampling, on Random Forest the results are improved quite a bit, with a 10-Fold Cross Validation average accuracy of 84.34% & 83.92% respectively

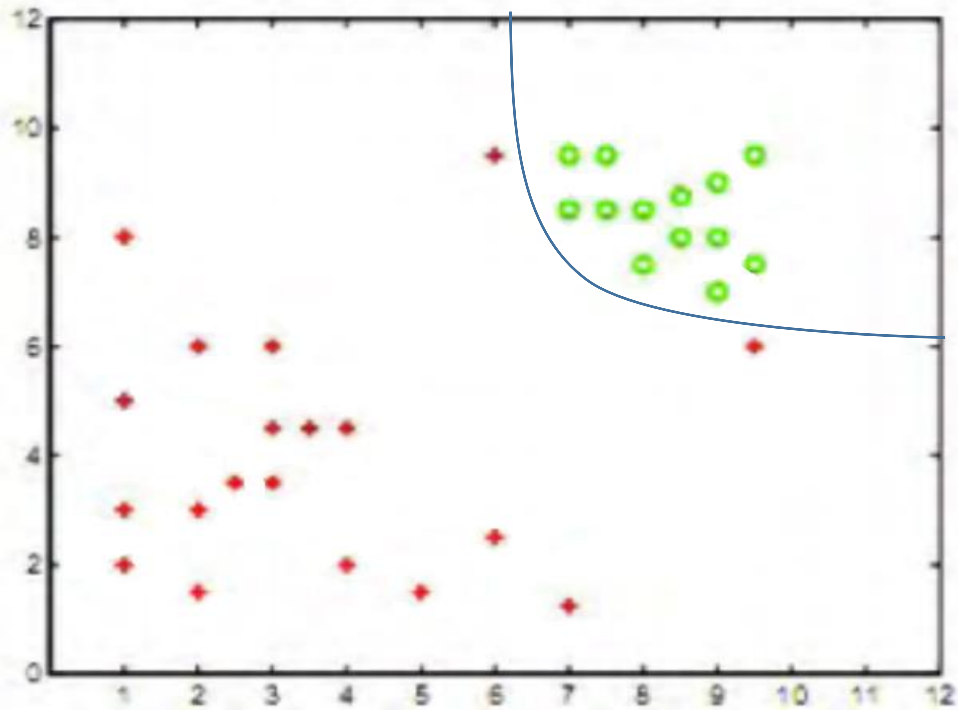
The important variables identified from all the models are quite similar.
From Random Forest with SMOTE data, important variables are:

CE_CSAT *- Overall, were you Satisfied by the service you recieved*
CE_VALUEFORMONEY *– Did you receive overall value for money?*
AgeYrs
Estimatedcost
DP_DISCHARGETIME – *Time Taken for Discharge Process*
BedCategory
CE_ACCESSIBILITY – *Did you find us when you need us?*
Department
DP_DISCHARGEPROCESS - *Overall Discharge Process*
AE_ATTENDEEFOOD – *Food options for your Attendee*
AD_TARRIFFPACKAGESEXPLAINATION – Explanation of Tarrif & Packages available
InsPayorcategory
DP_DISCHARGEQUERIES – *Communication & handling of queries*
FNB_FOODQUALITY – *Overall Quality & Taste of Food*
AD_TIME – *Time Taken for Admission*
STATEZONE
FNB_FOODDELIVERYTIME – *Timliness of Service*
INR_ROOMPEACE- *Peace & Quite in the Room*
FNB_DIETICIAN – *Regular Diet Counselling*

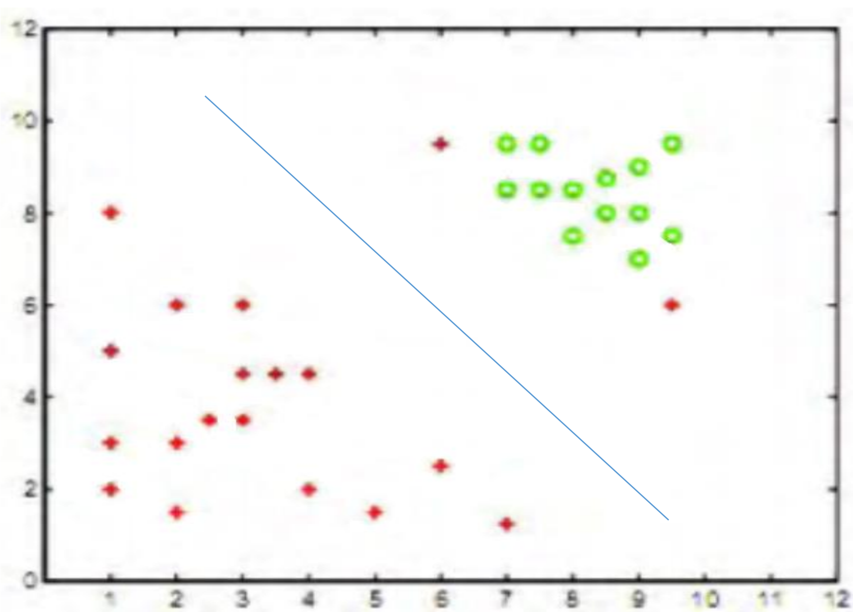
From the above, we understand that Value for Money, Overall Satisfaction, Food Quality, Timliness of Services, time taken for Dischare process, explanation of tarrif packages, peace & quite in Room, are major contributors for the given NPS Score. Thus MHE should maintain these features in order to reduce the proportion of Detractors.

Problem 2:

a. A large value of C means higher cost of misclassification. In this case the decision boundary will be such that it reduces the number of misclassifications, i.e, generate low training error. This may result in overfitting. We also have to minimize ||w||, so we choose the parabola with least curvature.



b. A small value of C means lower cost of misclassification. In this case the decision boundary will be such that it allows misclassification to a greater extent than the first case. This model will generate comparatively higher training error.



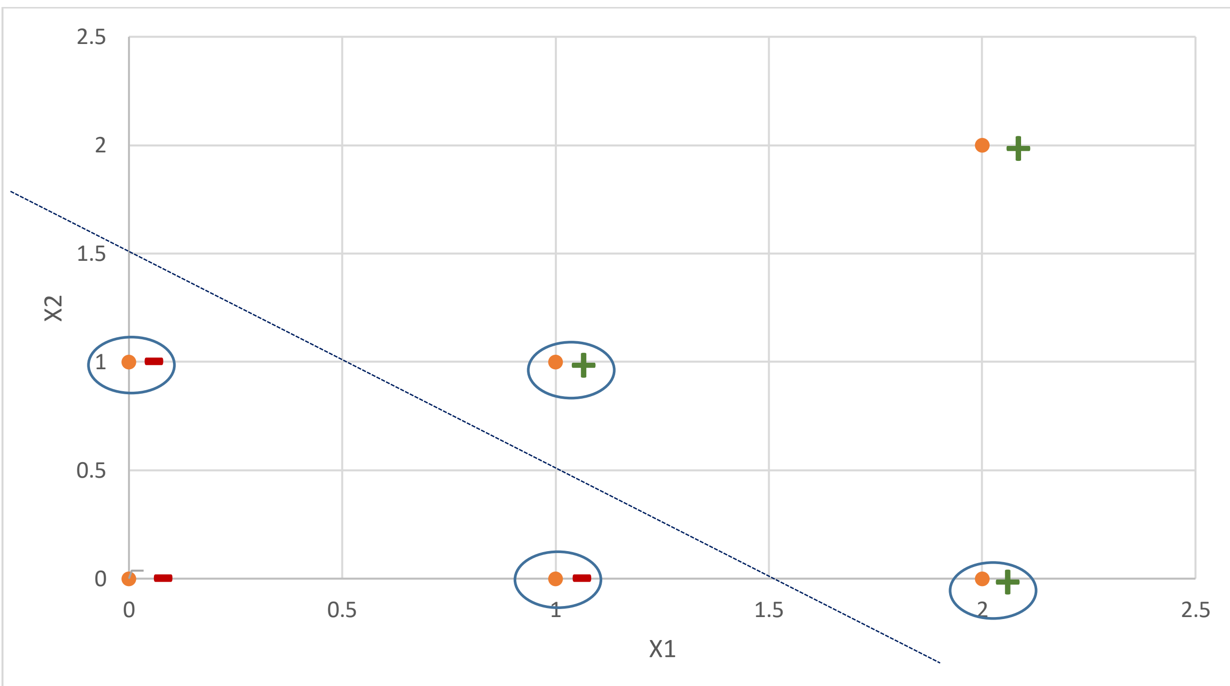
c. The SVM suffers from Overfitting, as it is too specific to the training data. This model might have low bias, but with new test points, the model may not perform as well and thus have high variance.

Problem 3:

class	x_1	x_2
+	1	1
+	2	2
+	2	0
−	0	0
−	1	0
−	0	1

a. Plot these six training points. Are the classes {+,-} linearly separable?

Solution: **Yes, the classes {+,-} are linearly separable**



b.) Construct the weight vector of the maximum margin hyperplane by inspection and identify the support vectors.

Solution: The line separating two classes is given by:

$$w_1x_1 + w_2x_2 + b = 0$$

We have to find w_1, w_2 and b, using the formula:

$$y_i\,(w.x_i + b) = 1 \qquad \text{--------------} \qquad \textbf{Equation 2}$$

We have 4 support vectors and thus 4 points that satisfy the Equation 2. The points are:
(1,1), (1,0), (2,0), (0,1)
Substituting the values of X_i :

$$\boldsymbol{Point\ (1,1):}\qquad +1\left[(w_1,w_2)\begin{pmatrix}1\\1\end{pmatrix}+b\right]=1$$

$$\equiv\ 1.w_1+1.w_2+b=1$$

$$\boldsymbol{Point\ (1,0):}\qquad -1\left[(w_1,w_2)\begin{pmatrix}1\\0\end{pmatrix}+b\right]=1$$

$$\equiv\ 1.w_1+0.w_2+b=-1$$

$$\boldsymbol{Point\ (2,0):}\qquad +1\left[(w_1,w_2)\begin{pmatrix}2\\0\end{pmatrix}+b\right]=1$$

$$\equiv\ 2.w_1+0.w_2+b=1$$

Solving the above three equations, we get:
$w_1 = 2$,
$w_2 = 2$,
$b = -3$,
Equation of the line separating two classes is:

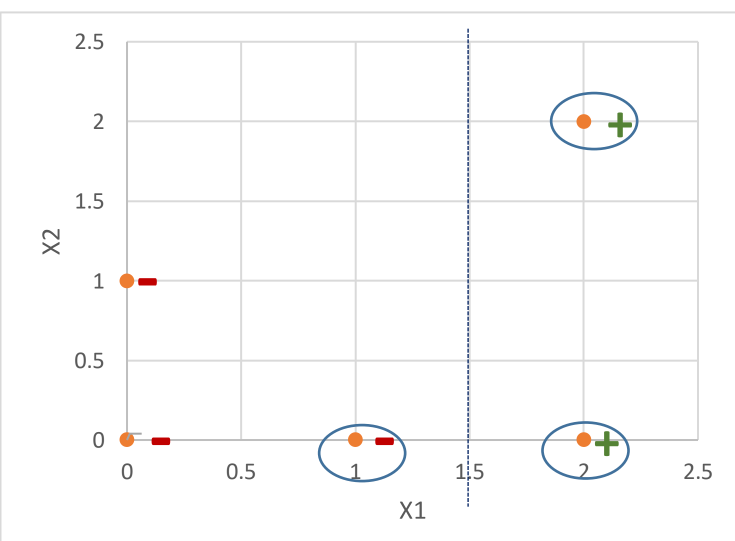
$$2x_1 + 2x_2 - 3 = 0$$

Margin is given by:

$$Margin = \frac{2}{\| w \|}$$

Margin = $\frac{1}{2\sqrt{2}} = 0.707$

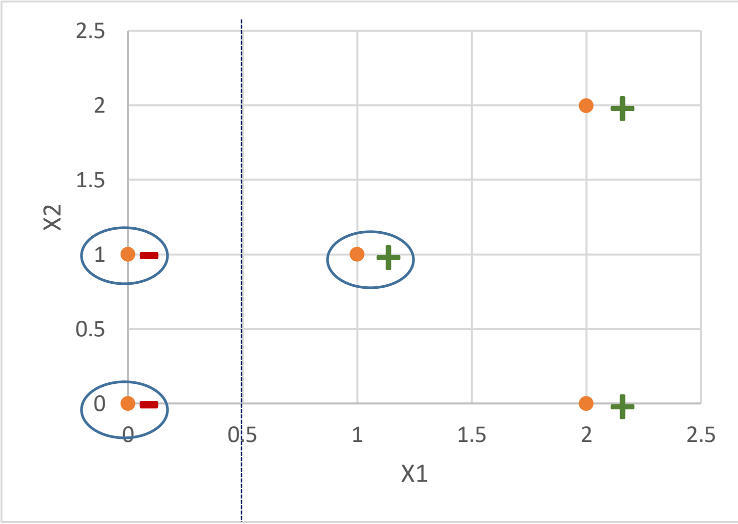
b.) If you remove one of the support vectors does the size of the optimal margin decrease, stay the same, or increase? Explain.



Case 1: If support vector (1,1) is removed. New support vectors are:
(2,2), (2,0) and (1,0)

$$2.w_1+2.w_2+b=1$$
$$2.w_1+0.w_2+b=1$$
$$1.w_1+0.w_2+b=-1$$

Solving these equations, we get:
$w_1 = 2$,
$w_2 = 0$,
$b = -3$,
Margin = $\frac{2}{\sqrt{4}} = 1$



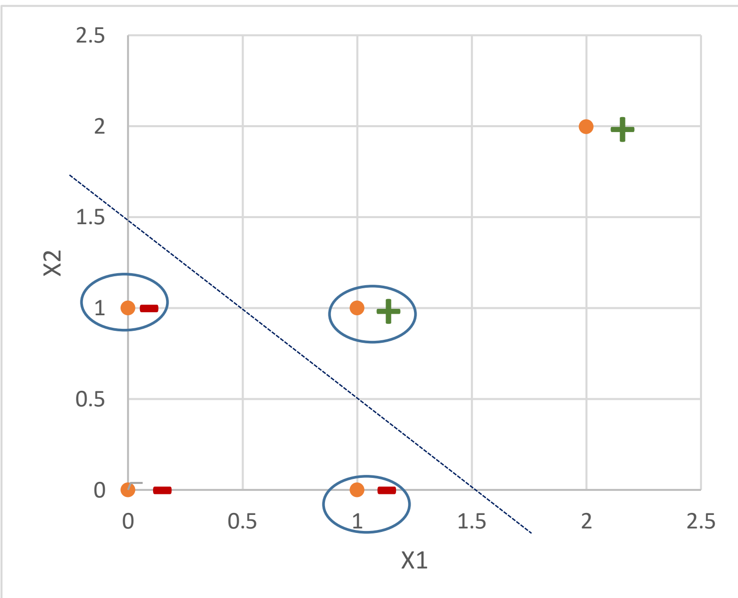
Case 2: If support vector (1,0) is removed. New support vectors are:
(0,0), (0,1) and (1,1)

$$\mathbf{0.w_1 + 0.w_2 + b = -1}$$
$$\mathbf{0.w_1 + 1.w_2 + b = -1}$$
$$\mathbf{1.w_1 + 1.w_2 + b = 1}$$

Solving these equations, we get:

$\mathbf{w_1 = 2}$,
$\mathbf{w_2 = 0}$,
$\mathbf{b = -1}$,

Margin = $\mathbf{\frac{2}{\sqrt{4}} = 1}$



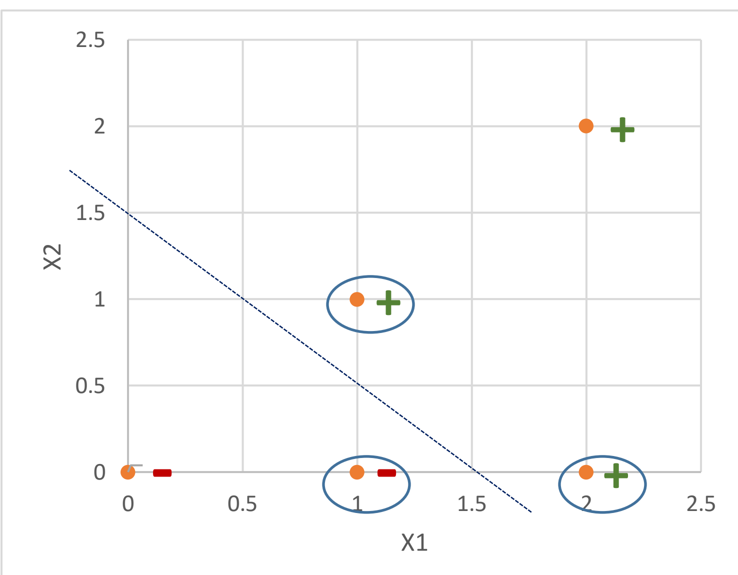
Case 3: If support vector (2,0) is removed. New support vectors are:
(1,1), (0,1) and (1,0)

$$\mathbf{1.w_1 + 1.w_2 + b = 1}$$
$$\mathbf{0.w_1 + 1.w_2 + b = -1}$$
$$\mathbf{1.w_1 + 0.w_2 + b = -1}$$

Solving these equations, we get:

$\mathbf{w_1 = 2}$,
$\mathbf{w_2 = 2}$,
$\mathbf{b = -3}$,

Margin = $\mathbf{\frac{2}{2\sqrt{2}} = 0.707}$



Case 4: If support vector (0,1) is removed. New support vectors are:
(1,1), (1,0) and (2,0)

$$\mathbf{1.w_1 + 1.w_2 + b = 1}$$
$$\mathbf{0.w_1 + 1.w_2 + b = -1}$$
$$\mathbf{1.w_1 + 0.w_2 + b = -1}$$

Solving these equations, we get:

$\mathbf{w_1 = 2}$,
$\mathbf{w_2 = 2}$,
$\mathbf{b = -3}$,

Margin = $\mathbf{\frac{2}{2\sqrt{2}} = 0.707}$

If we remove support vectors (1,1) or (1,0), the margin increases to 1. If we remove support vectors (2,0) or (0,1), the margin remains the same at 0.707.