Health Insurance Cross Sell

```r
getwd();
```

```
## [1] "/Users/jeanbai/Desktop/ML_YorkU/groupC #1"
```

```r
data=read.csv("train.csv", header = TRUE, na.strings = c("NA","","#NA"))
```

Data preparation for prediction models

Remove "id" feature.

```r
data$id = NULL
```

###Encoding categorical data ####Convert Gender, Vehicle_Age, Vehicle_Damage from categorical variables to factors

```r
data$Gender = factor(data$Gender,
                         levels = c('Male', 'Female'),
                         labels = c(1, 2))
data$Vehicle_Age = factor(data$Vehicle_Age,
                          levels = c('> 2 Years', '1-2 Year', '< 1 Year'),
                          labels = c(2,1,0))
data$Vehicle_Damage = factor(data$Vehicle_Damage,
                             levels = c("Yes", "No"),
                             labels = c(1,0))
```

A categorical variable can be divided into nominal categorical variable and ordinal categorical variable.Continuous class variables are the default value in R. They are stored as numeric or integer.

Driving_License and Previously_Insured are nominal cateforical variables but labeled as intergers. We need to convert them into factors.

```r
data$Driving_License = as.factor(data$Driving_License)
data$Previously_Insured = as.factor(data$Previously_Insured)
```

Convert numeric variables to levels of factors

"Region_code's variables and Policy_Sales_Channel's variables are in the format of numeric. However those numbers are characters. Region_Code are the unique code for the region of the customer; PolicySalesChannel are the anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc. So we need to convert those numerics to charactors and then group them by the frequency.

```r
data$Region_Code = as.factor(data$Region_Code)
data$Policy_Sales_Channel = as.factor(data$Policy_Sales_Channel)
```

Check how many levels of Region_Code

```r
levels(data$Region_Code)
```

```
##  [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [16] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29"
## [31] "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42" "43" "44"
## [46] "45" "46" "47" "48" "49" "50" "51" "52"
```

There are 53 levels(0 - 52) in Region_Code. We need check the order of the frequency and group them into less levels to avoid overfitting issues when we do the modeling.

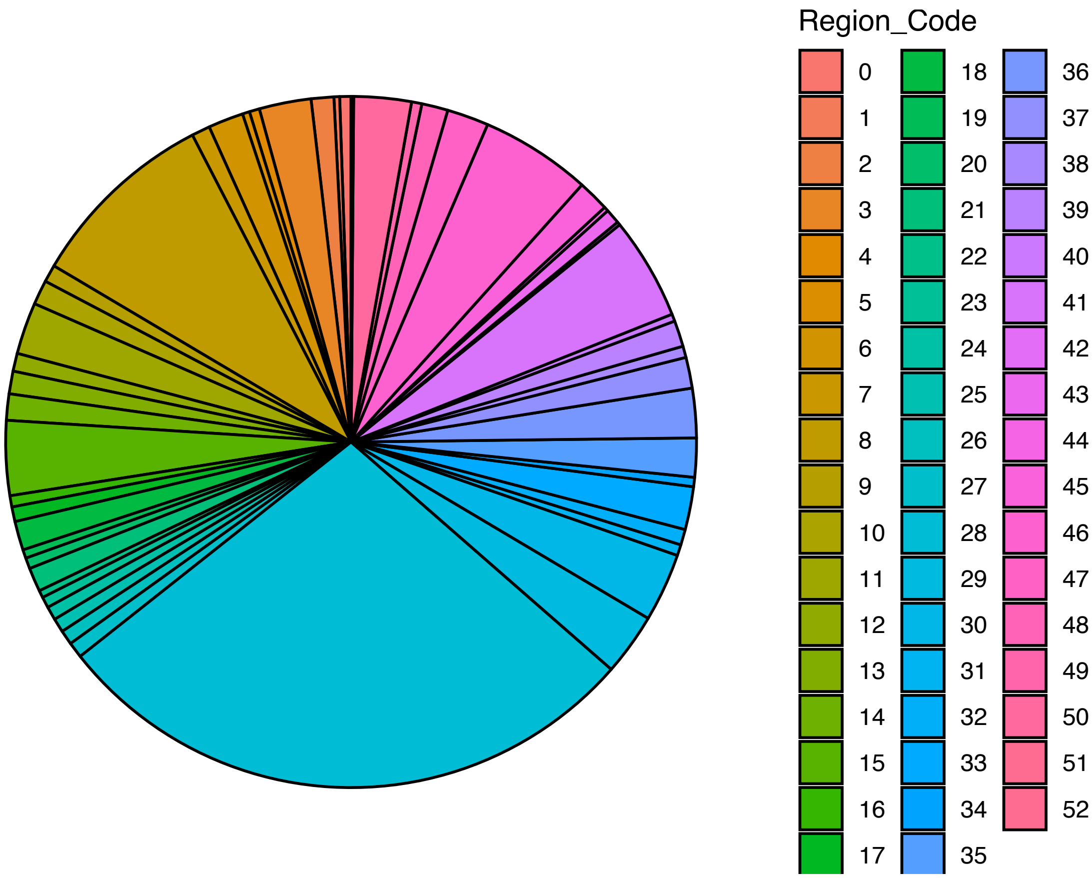
```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

Check the frequency of each level in Region_Code

```r
g1 = ggplot(data, aes(x=character(1), fill=Region_Code))+
   geom_bar(width=1, colour="black")+
   coord_polar(theta="y")+
   theme_void()
print(g1)
```




```
sort(table(data$Region_Code), decreasing = TRUE)
```

```
##
##     28     8    46    41    15    30    29    50     3    11    36
## 106415 33877 19749 18263 13308 12191 11065 10243  9251  9232  8797
##     33    47    35     6    45    37    18    48    14    39    10
##   7654  7436  6942  6280  5605  5501  5153  4681  4678  4644  4374
##     21     2    13     7    12     9    27    32    43    17    26
##   4266  4038  4036  3279  3198  3101  2823  2787  2639  2617  2587
##     25    24    38     0    16    23    31    20    49     4    34
##   2503  2415  2026  2021  2007  1960  1960  1935  1832  1801  1664
##     19    22    40     5     1    44    42    52    51
##   1535  1309  1295  1279  1008   808   591   267   183
```

The top 8 frequency Region_Code are “28”, “8”, “46”, “41”, “15”, “30”,“29”,“50”. Base on above plot and sort table we can group the Region_Code by the frequency into 9 groups including "other’ group.

```
library(forcats)
```

```
library(dplyr)
```

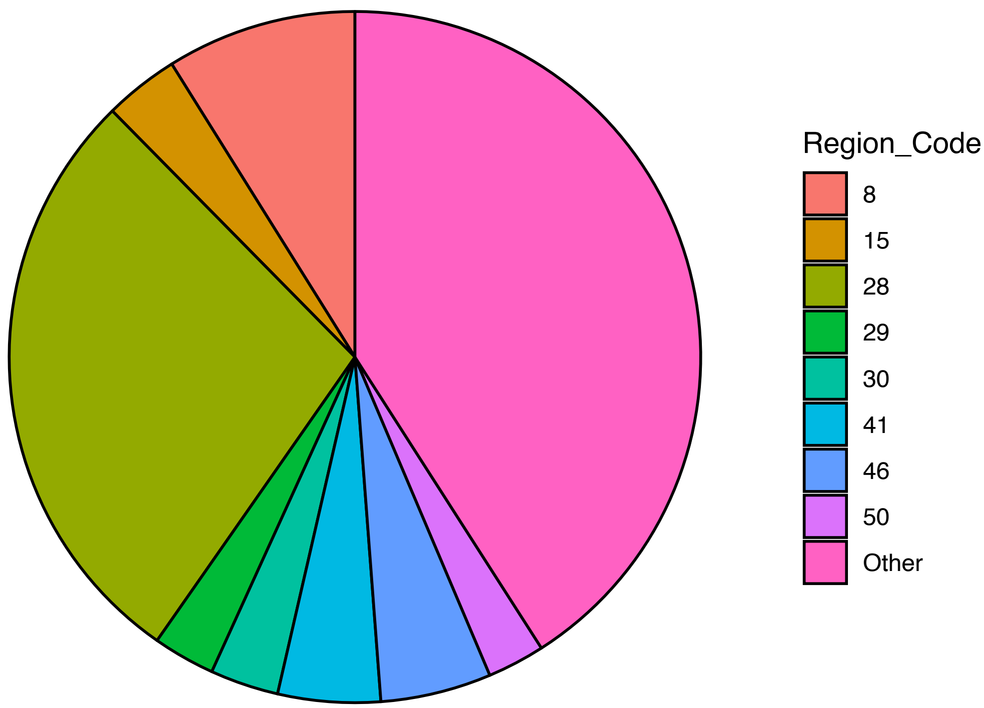
```
data$Region_Code =forcats::fct_lump_n(data$Region_Code,8, other_level = "Other")
```

```
levels(data$Region_Code)
```

```
## [1] "8"     "15"    "28"    "29"    "30"    "41"    "46"    "50"    "Other"
```

We get 9 levels of Region_Code.

```
g1 = ggplot(data, aes(x=factor(1), fill=Region_Code))+
   geom_bar(width=1, colour="black")+
   coord_polar(theta="y")+
   theme_void()
print(g1)
```



Relabel the factor levers of Region_Code

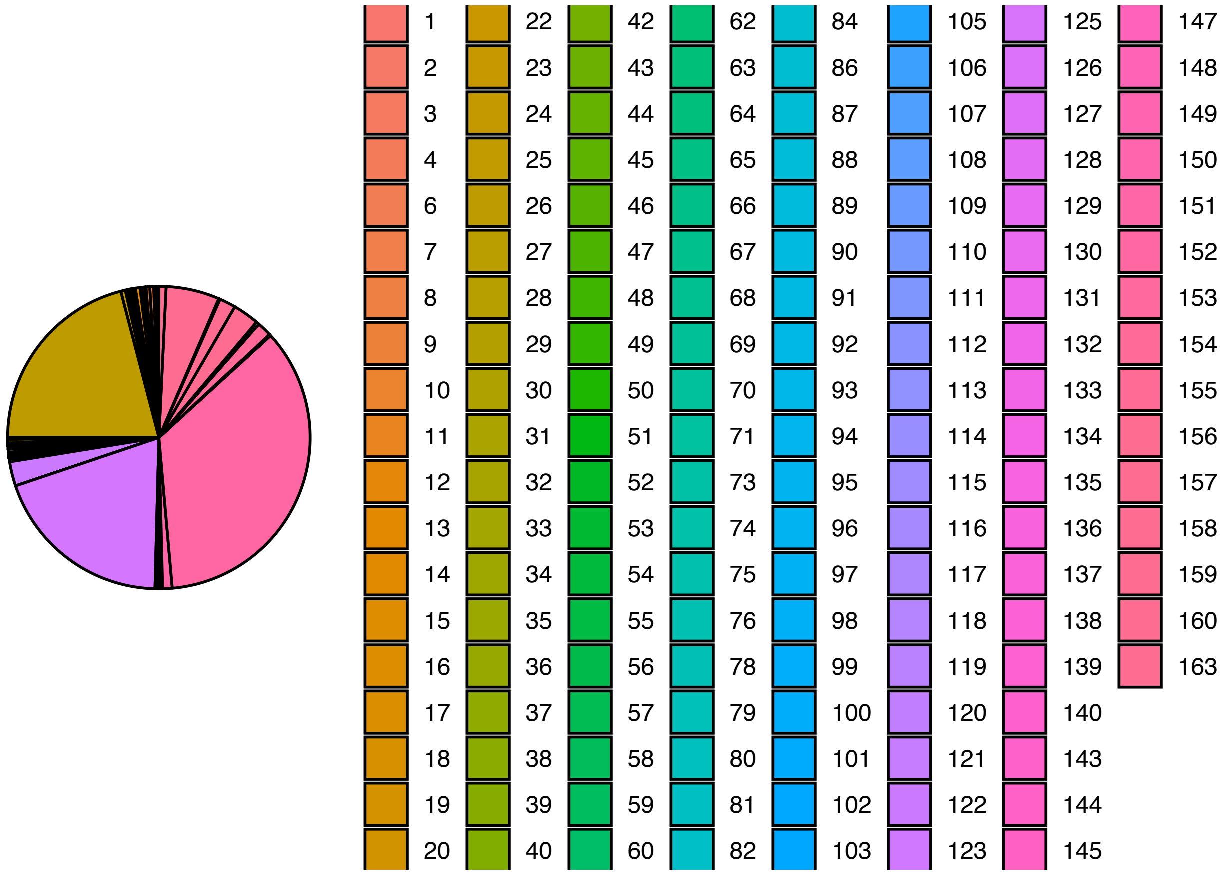
```
data$Region_Code = factor(data$Region_Code,
                          levels = c('15','28','29','30','41','46','50', '8', 'Other'),
                          labels = c(1, 2,3,4,5,6,7,8,9))
```

```
levels(data$Region_Code)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

Using forcats method check the order of frequency in Policy_Sales_Channel

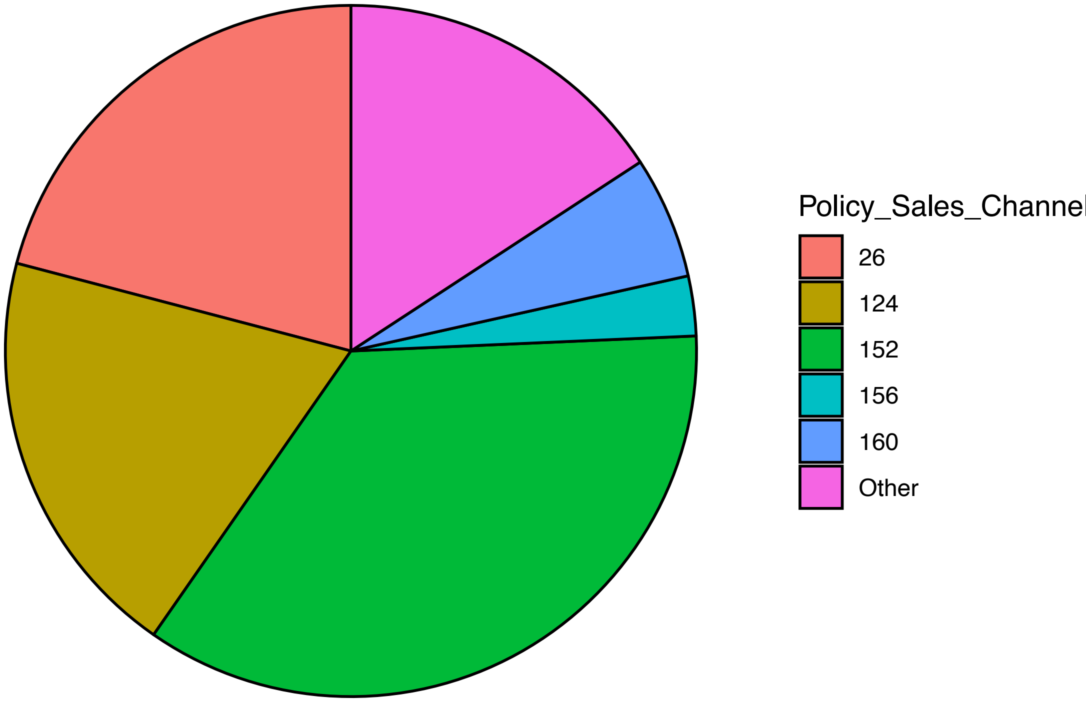
```r
g2 = ggplot(data, aes(x=character(1), fill=Policy_Sales_Channel))+
   geom_bar(width=1, colour="black")+
   coord_polar(theta="y")+
   theme_void()
print(g2)
```



Base on above plot,that we can group the Policy_Sales_Channel by the frequency into 6 groups including one "Other" group.

```r
data$Policy_Sales_Channel =forcats::fct_lump_n(data$Policy_Sales_Channel,5, other_level = "Other")
```

```r
g2 = ggplot(data, aes(x=factor(1), fill=Policy_Sales_Channel))+
   geom_bar(width=1, colour="black")+
   coord_polar(theta="y")+
   theme_void()
print(g2)
```



Relabel the levels of Policy_Sales_Channel

```
data$Policy_Sales_Channel = factor(data$Policy_Sales_Channel,
                         levels = c('124', '152','156','160', '26','Other'),
                         labels = c(1,2,3,4,5,6))
```

```
levels(data$Policy_Sales_Channel)
```

```
## [1] "1" "2" "3" "4" "5" "6"
```

Using Capping method to treat the Annual_Premium outliers issue.

```
pcap <- function(x){
  for (i in which(sapply(x, is.numeric))) {
    quantiles <- quantile( x[,i], c(.05, .95 ), na.rm =TRUE)
    x[,i] = ifelse(x[,i] < quantiles[1] , quantiles[1], x[,i])
    x[,i] = ifelse(x[,i] > quantiles[2] , quantiles[2], x[,i])}
  x}
```

```
data = pcap(data)
summary(data$Annual_Premium)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2630   24405   31669   29898   39400   55176
```

(There is an article in a website “If you choose too large of a training set you run the risk of overfitting your model. Overfitting is a classic mistake people make when first entering the field of machine learning.”)

We have 381,109.00 observations we will going to only use 10% of the raw data as a model data and split the 10% into train/test datasets.

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(caTools)
```

Using the Partition method to get a new dataset and use the new data as a sample data to do the medolling.We will use the1% observations to do the data modeling

```
set.seed(198)
sample_split = createDataPartition(data$Response, p = 0.1, list=FALSE)
sampleData = data[sample_split,]
remainData = data[-sample_split,]
```

```
dim(sampleData)
```

```
## [1] 38111    11
```

```
dim(remainData)
```

```
## [1] 342998     11
```

```
library(data.table)
library(dplyr)
```

convert all sampleDate factor levels to numeric so that we can scale the data to do the modelling.

```
indx <- sapply(sampleData[], is.factor)
sampleData[indx] <- lapply(sampleData[indx], function(x) as.numeric(as.factor(x)))
```

```
str(sampleData)
```

```
## 'data.frame':    38111 obs. of  11 variables:
##  $ Gender              : num  2 2 2 2 1 2 2 1 1 2 ...
##  $ Age                 : num  32 21 25 62 39 27 39 69 50 21 ...
##  $ Driving_License     : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Region_Code         : num  9 2 2 2 1 9 8 2 2 9 ...
##  $ Previously_Insured  : num  2 2 2 1 1 2 2 1 1 1 ...
##  $ Vehicle_Age         : num  3 3 3 1 2 3 2 2 2 3 ...
##  $ Vehicle_Damage      : num  2 2 2 1 1 2 2 1 1 1 ...
##  $ Annual_Premium      : num  28771 55176 55176 33830 37849 ...
##  $ Policy_Sales_Channel: num  2 2 2 5 1 2 2 6 6 2 ...
##  $ Vintage             : num  80 72 107 130 24 111 131 158 285 79 ...
##  $ Response            : int  0 0 0 0 0 0 0 0 0 0 ...
```

convert Response to factor variables.

```
sampleData$Response = as.factor(sampleData$Response)
```

```
is.factor(sampleData$Response)
```

```
## [1] TRUE
```

Split the sampleDate to generate train and test dataset.We only use 20% of the sampleData as the training set.

```
set.seed(198)
split = sample.split(sampleData$Response, SplitRatio = 0.2)
train = subset(sampleData, split == TRUE)
test = subset(sampleData, split == FALSE)
```

```
dim(train)
```

```
## [1] 7622   11
```

```
dim(test)
```

```
## [1] 30489    11
```

Comparing the train dataset and original dataset

```
table(data$Response)
```

```
##
##      0      1
## 334399  46710
```

```
prop.table(table(data$Response))
```

```
##
##         0         1
## 0.8774366 0.1225634
```

```
table(train$Response)
```

```
##
##    0    1
## 6701  921
```

```
prop.table(table(train$Response))
```

```
##
##         0         1
## 0.8791656 0.1208344
```

The Percentage of customer who have positive response“1” are simily, which is 12%. So that the small sample of train set can represent the original data. We will use the train dataset to do our model.

Features scaling

```
train[,c(2,8,10) ] = scale(train[, c(2,8,10)])
```

```
str(train)
```

```
## 'data.frame':    7622 obs. of  11 variables:
##  $ Gender              : num  1 1 1 1 2 1 1 1 1 1 ...
##  $ Age                 : num  0.0325 0.7735 -1.18 0.5041 0.4367 ...
##  $ Driving_License     : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Region_Code         : num  1 4 9 2 2 9 2 2 2 1 ...
##  $ Previously_Insured  : num  1 1 2 1 1 1 1 1 2 2 ...
##  $ Vehicle_Age         : num  2 2 3 2 2 2 2 2 2 3 ...
##  $ Vehicle_Damage      : num  1 1 2 1 1 2 1 1 2 2 ...
##  $ Annual_Premium      : num  0.518 0.281 -0.112 1.663 -0.292 ...
##  $ Policy_Sales_Channel: num  1 5 4 5 5 6 1 1 1 6 ...
##  $ Vintage             : num  -1.554 0.919 -0.903 -0.3 -0.819 ...
##  $ Response            : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 2 1 1 ...
```

```
test[,c(2,8,10) ] = scale(test[, c(2,8,10)])
```

```
str(test)
```

```
## 'data.frame':    30489 obs. of  11 variables:
##  $ Gender              : num  2 2 2 2 2 2 1 1 2 1 ...
##  $ Age                 : num  -0.446 -1.179 -0.912 1.552 -0.779 ...
##  $ Driving_License     : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Region_Code         : num  9 2 2 2 9 8 2 2 9 9 ...
##  $ Previously_Insured  : num  2 2 2 1 2 2 1 1 1 2 ...
##  $ Vehicle_Age         : num  3 3 3 1 3 2 2 2 3 3 ...
##  $ Vehicle_Damage      : num  2 2 2 1 2 2 1 1 1 2 ...
##  $ Annual_Premium      : num  -0.0774 1.6802 1.6802 0.2594 0.3713 ...
##  $ Policy_Sales_Channel: num  2 2 2 5 2 2 6 6 2 2 ...
##  $ Vintage             : num  -0.894 -0.991 -0.566 -0.286 -0.517 ...
##  $ Response            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Create Models

Logistic regression classifier model

```
glmModel = glm(Response ~., train, family = binomial)
```

```
summary(glmModel)
```

```
##
## Call:
## glm(formula = Response ~ ., family = binomial, data = train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.1878  -0.6613  -0.0483  -0.0357   3.7690
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -17.965674 469.686804  -0.038   0.9695
## Gender                -0.124185   0.079017  -1.572   0.1160
## Age                   -0.283487   0.057069  -4.967 6.78e-07 ***
## Driving_License       12.298205 234.843223   0.052   0.9582
## Region_Code           -0.056904   0.012874  -4.420 9.86e-06 ***
## Previously_Insured    -3.793828   0.526371  -7.208 5.70e-13 ***
## Vehicle_Age           -0.752726   0.096435  -7.806 5.93e-15 ***
## Vehicle_Damage        -1.924101   0.243165  -7.913 2.52e-15 ***
## Annual_Premium        -0.013471   0.038248  -0.352   0.7247
## Policy_Sales_Channel   0.036249   0.019827   1.828   0.0675 .
## Vintage                0.001889   0.038468   0.049   0.9608
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5618.7  on 7621  degrees of freedom
## Residual deviance: 4190.6  on 7611  degrees of freedom
## AIC: 4212.6
##
## Number of Fisher Scoring iterations: 13
```

Features selection

Gender, Driving_License, Annual_Premium , Policy_Sales_Channel and Vintage have P_valua are much more than 0.05. We remove these four features from both the train dataset and test dataset.

```r
train$Gender = NULL
train$Driving_License = NULL
train$Annual_Premium = NULL
train$Policy_Sales_Channel = NULL
train$Vintage = NULL
```

```r
test$Gender = NULL
test$Driving_License = NULL
test$Annual_Premium = NULL
test$Policy_Sales_Channel = NULL
test$Vintage = NULL
```

```r
dim(train)
```

```
## [1] 7622    6
```

```r
dim(test)
```

```
## [1] 30489     6
```

New GLM model

```
glmNew = glm(Response ~., train, family = binomial)
```

Use the new glm model to do the probability prediction.

```
prob_pred = predict(glmNew, type = 'response', test[-6])
```

Change prob_pred percentage of probability to "1", "0" binimial number.

```
y_pred = ifelse(prob_pred >0.5, 1, 0)
```

```
is.vector(y_pred)
```

```
## [1] TRUE
```

```
is.atomic(test$Response)
```

```
## [1] TRUE
```

Convert "y_pred" list vector to atomic vector matching with the test$Response for comparison

```
y_pred = as.character(as.numeric(as.integer(y_pred)))
```

```
is.atomic(y_pred)
```

```
## [1] TRUE
```

```
cm = table(test[,6], y_pred)
```

```
cm
```

```
##    y_pred
##          0     1
##   0 26780    24
##   1  3669    16
```

The model predict customer responce "0", which is not interested. There is an imbalanced classification we need to adjuster the imbalance

```
levels(as.factor(y_pred))
```

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

```
library(caret)
```

```r
confusionMatrix(as.factor(y_pred), test$Response, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 26780  3669
##          1    24    16
##
##                Accuracy : 0.8789
##                  95% CI : (0.8752, 0.8825)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 0.5602
##
##                   Kappa : 0.006
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0043419
##             Specificity : 0.9991046
##          Pos Pred Value : 0.4000000
##          Neg Pred Value : 0.8795034
##              Prevalence : 0.1208633
##          Detection Rate : 0.0005248
##    Detection Prevalence : 0.0013119
##       Balanced Accuracy : 0.5017233
##
##        'Positive' Class : 1
##
```

Althogh we got 0.8789 accuracy , however the Sensitivity is only 0.004. That means the model detect customer did not respons very well, however did not detect those customers who are interested in the cross sell. There is strong imbalance clissificaton issues .

Solve the imbalance classification

```r
library(ROSE)
```

Generate new balanced data by ROSE. Use Over sampling for better sensitivity

```r
table(train$Response)
```

```
##
##    0    1
## 6701  921
```

```r
6701*2
```

```
## [1] 13402
```

```r
over = ovun.sample(Response~., data=train, method = "over", N=13402)$data
```

```r
table(over$Response)
```

```
##
##    0    1
## 6701 6701
```

```r
summary(over)
```

```
##       Age             Region_Code    Previously_Insured  Vehicle_Age
##  Min.   :-1.1800   Min.   :1.000   Min.   :1.000      Min.   :1.000
##  1st Qu.:-0.7758   1st Qu.:2.000   1st Qu.:1.000      1st Qu.:2.000
##  Median : 0.1673   Median :6.000   Median :1.000      Median :2.000
##  Mean   : 0.1343   Mean   :5.585   Mean   :1.263      Mean   :2.249
##  3rd Qu.: 0.7735   3rd Qu.:9.000   3rd Qu.:2.000      3rd Qu.:3.000
##  Max.   : 2.0534   Max.   :9.000   Max.   :2.000      Max.   :3.000
##  Vehicle_Damage  Response
##  Min.   :1.000   0:6701
##  1st Qu.:1.000   1:6701
##  Median :1.000
##  Mean   :1.287
##  3rd Qu.:2.000
##  Max.   :2.000
```

```r
glm_over = glm(Response~., over, family = binomial)
```

```r
dim(test)
```

```
## [1] 30489     6
```

```r
over_pred = predict(glm_over, type = 'response', test[-6])
```

```r
y_over_pred = ifelse(over_pred >0.5, 1, 0)
```

```r
y_over_pred = as.factor(y_over_pred)
```

```r
levels(y_over_pred)
```

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

```r
levels(test$Response)
```

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

```r
cm = table(test[,6], y_over_pred)
```

```r
cm
```

```
##    y_over_pred
##         0     1
##   0 15820 10984
##   1    80  3605
```

```r
library(caret)
```

```r
confusionMatrix(as.factor(y_over_pred), test$Response, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15820    80
##          1 10984  3605
##
##                Accuracy : 0.6371
##                  95% CI : (0.6317, 0.6425)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2498
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9783
##             Specificity : 0.5902
##          Pos Pred Value : 0.2471
##          Neg Pred Value : 0.9950
##              Prevalence : 0.1209
##          Detection Rate : 0.1182
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.7843
##
##        'Positive' Class : 1
##
```

0.97 Sensitivity rate. That means this model can predict 97% of those customer who are intersted the cross sell. So far we got a good model. Let try other models to see which one is fit the data most. We will focus on the model Sensitivity value, which indicate how much the percentage accuracy the model catched for those customer who is interested in the cross sell.

Apply the treated training set to other models

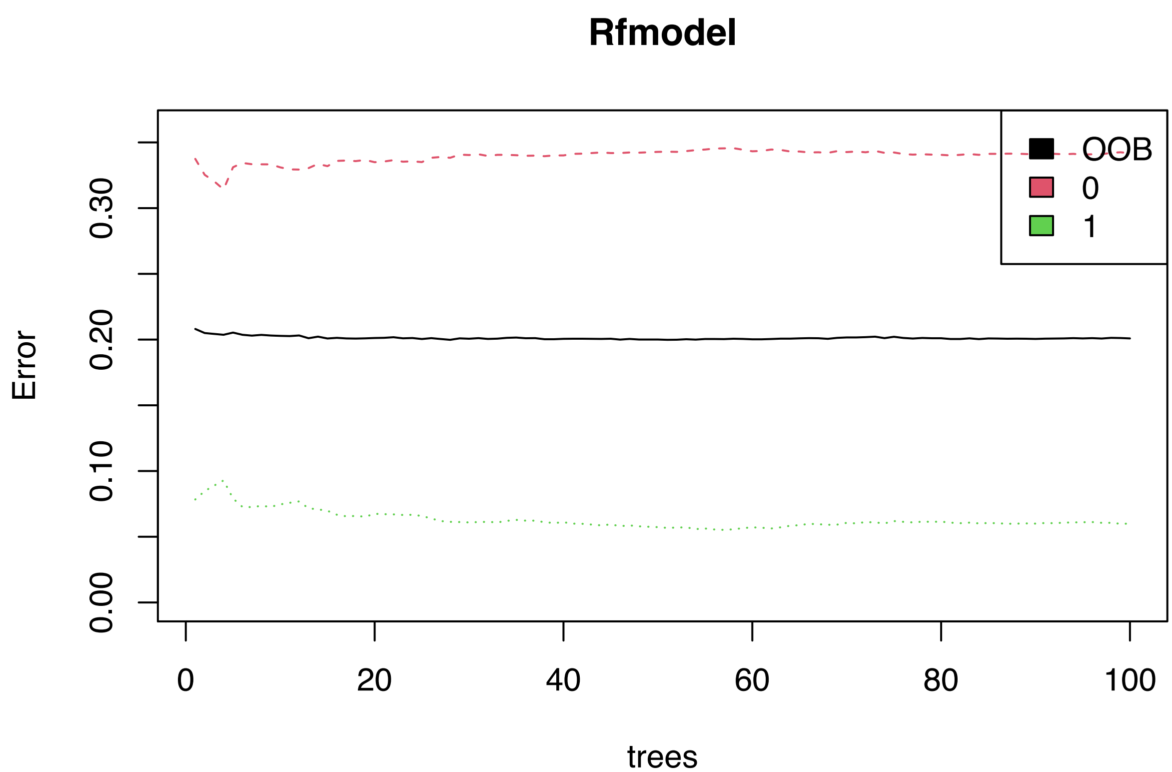
Random Forest Prediction

```r
library(randomForest)
```

```r
set.seed(123)
Rfmodel <- randomForest(Response ~ ., method= "anova",data=over, importance= TRUE, ntree = 100)
```

Predict using the test set

```r
plot(Rfmodel, ylim=c(0,0.36))
legend('topright', colnames(Rfmodel$err.rate), col=1:3, fill=1:3)
```



The black line shows the overall error rate which falls around 20%%. The red and green lines show the error rate for 'not responce' and 'repsonce' respectively. Less error in prediction the "Responce" rate.

```r
set.seed(123)
confusionMatrix(predict(Rfmodel, test), test$Response, positive = "1")
```

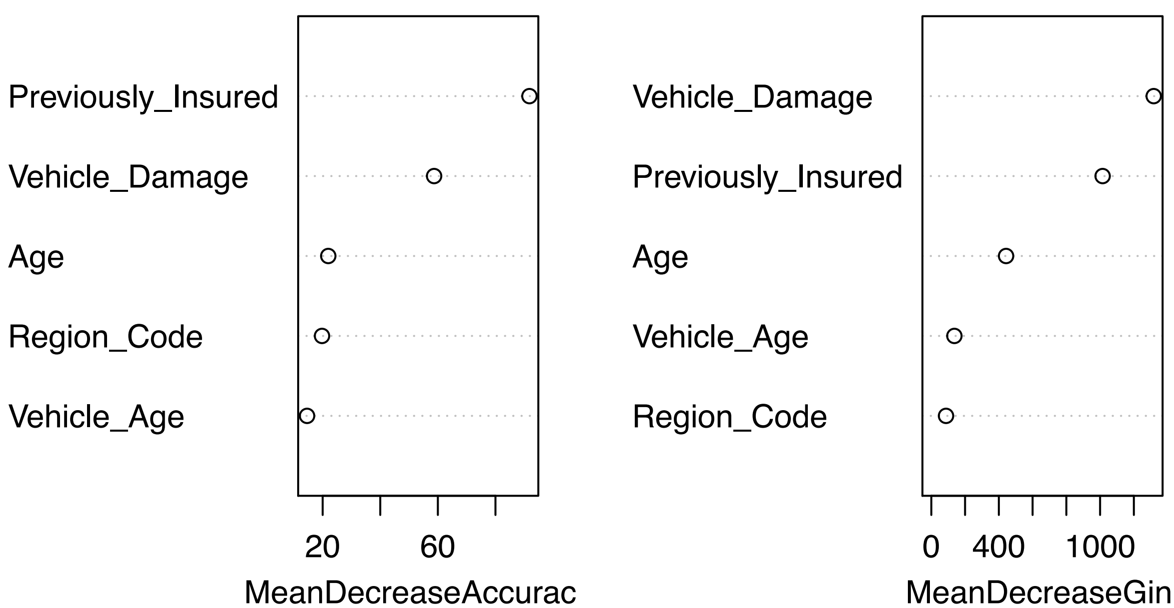
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17640   281
##          1  9164  3404
##
##                Accuracy : 0.6902
```

```
##                  95% CI : (0.685, 0.6954)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2853
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9237
##             Specificity : 0.6581
##          Pos Pred Value : 0.2708
##          Neg Pred Value : 0.9843
##              Prevalence : 0.1209
##          Detection Rate : 0.1116
##    Detection Prevalence : 0.4122
##       Balanced Accuracy : 0.7909
##
##        'Positive' Class : 1
##
```

Sensitivity is 0.9237.

Get features importance

```r
varImpPlot(Rfmodel, main="")
```



The left figure above, is the important features order of Random Forest. Previously_Insured and Vehicle_Damage would be categorized as the most important features when predicting response. Age, Vehicle_Age and Region_code would fall under moderate importance. The right figure is the important features order of the model of logistic regression which using the Gini importance method while the Vehicle_Damage is the most importanct features.

Support Vector Classification (SVM_Classification)

```r
library(e1071)
```

```r
set.seed(123)
svm_model = svm(Response ~ ., data=over, type = 'C-classification', kernel = 'radial')
```

```r
predSVM <- predict(svm_model, test[-6])
```

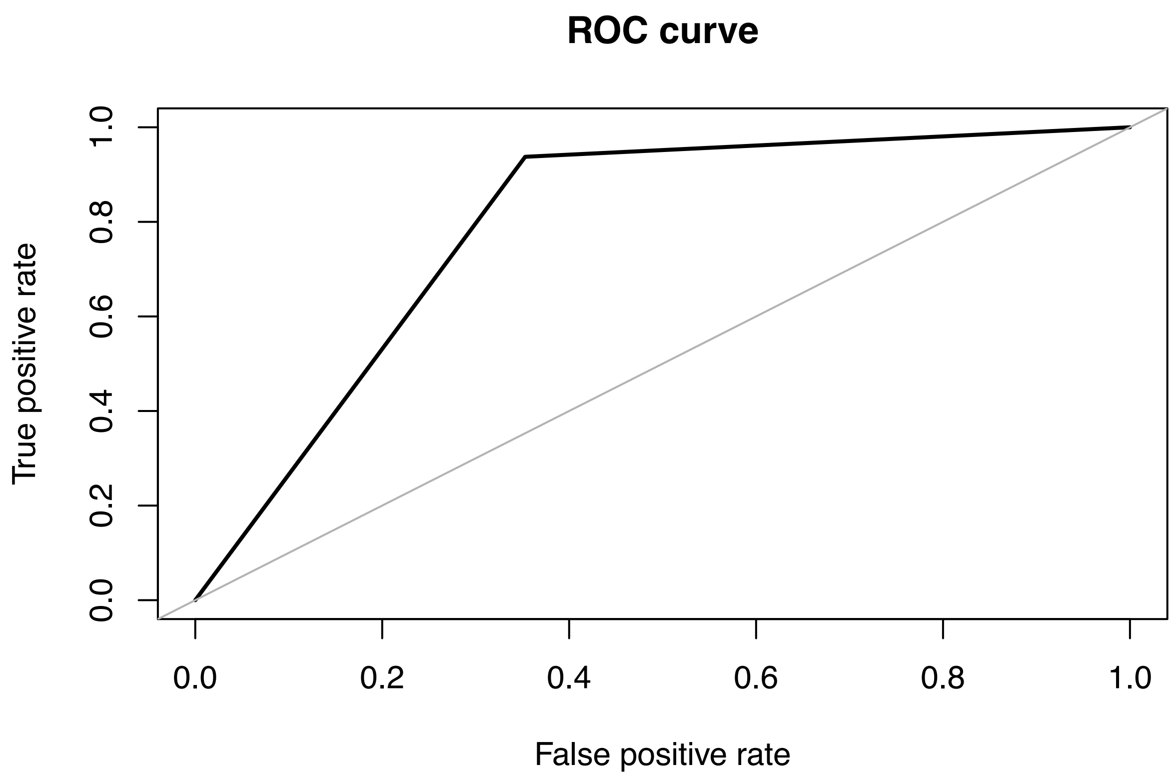
```r
set.seed(123)
confusionMatrix(as.factor(predSVM), test$Response, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 17342   230
##          1  9462  3455
##
##                Accuracy : 0.6821
##                  95% CI : (0.6769, 0.6873)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.281
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9376
##             Specificity : 0.6470
##          Pos Pred Value : 0.2675
##          Neg Pred Value : 0.9869
##              Prevalence : 0.1209
##          Detection Rate : 0.1133
##    Detection Prevalence : 0.4237
##       Balanced Accuracy : 0.7923
##
##        'Positive' Class : 1
##
```

Sensitivity is 0.93, close to the one of Random Forest.

```r
library(pROC)
```

```r
roc.curve(test$Response, predSVM,plotit= TRUE, add.roc = FALSE)
```




```
## Area under the curve (AUC): 0.792
```

Naive Bayes Model

```r
library(e1071)
```

```r
set.seed(123)
naive_model=naiveBayes(Response~.,
  data=over)
```

```r
pred_nb = predict(naive_model, test[-6])
```

```r
confusionMatrix(pred_nb, test$Response, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction     0     1
##          0 15812    76
##          1 10992  3609
##
##                Accuracy : 0.637
##                  95% CI : (0.6316, 0.6424)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.25
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9794
##             Specificity : 0.5899
##          Pos Pred Value : 0.2472
##          Neg Pred Value : 0.9952
##              Prevalence : 0.1209
##          Detection Rate : 0.1184
##    Detection Prevalence : 0.4789
##       Balanced Accuracy : 0.7846
##
##        'Positive' Class : 1
##
```

Sensitivity score is 0.9794.

Decision Tree

```r
library(rpart)
```

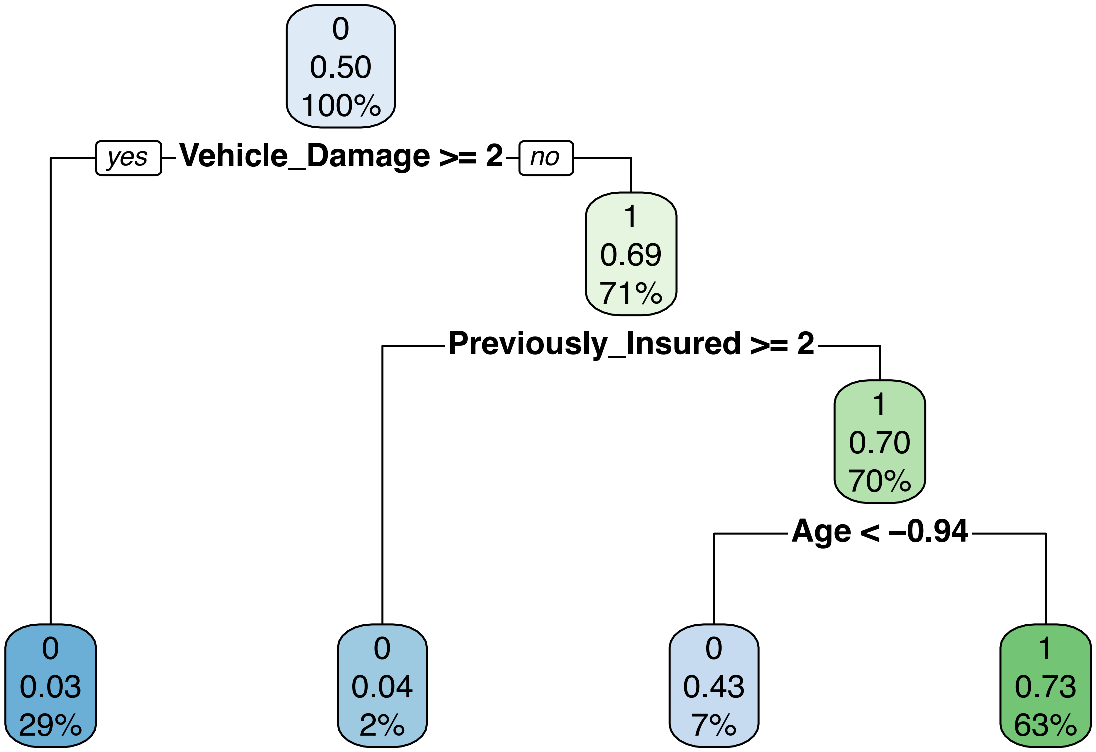
```r
set.seed(123)
treeModel = rpart(Response~., over  )
```

```r
predTree = predict(treeModel, test[-6])
```

```r
y_predTree = ifelse(over_pred >0.5, 1, 0)
```

```r
library(rpart.plot)
```

```r
rpart.plot(treeModel)
```



Decision Tree Model Evaluation

Making the Confusion Matrix

```
set.seed(123)
confusionMatrix(as.factor(y_predTree), test$Response, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15820    80
##          1 10984  3605
##
##                Accuracy : 0.6371
##                  95% CI : (0.6317, 0.6425)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2498
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9783
##             Specificity : 0.5902
##          Pos Pred Value : 0.2471
```

```
##          Neg Pred Value : 0.9950
##              Prevalence : 0.1209
##          Detection Rate : 0.1182
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.7843
##
##        'Positive' Class : 1
##
```

Sensitivity is 0.978.

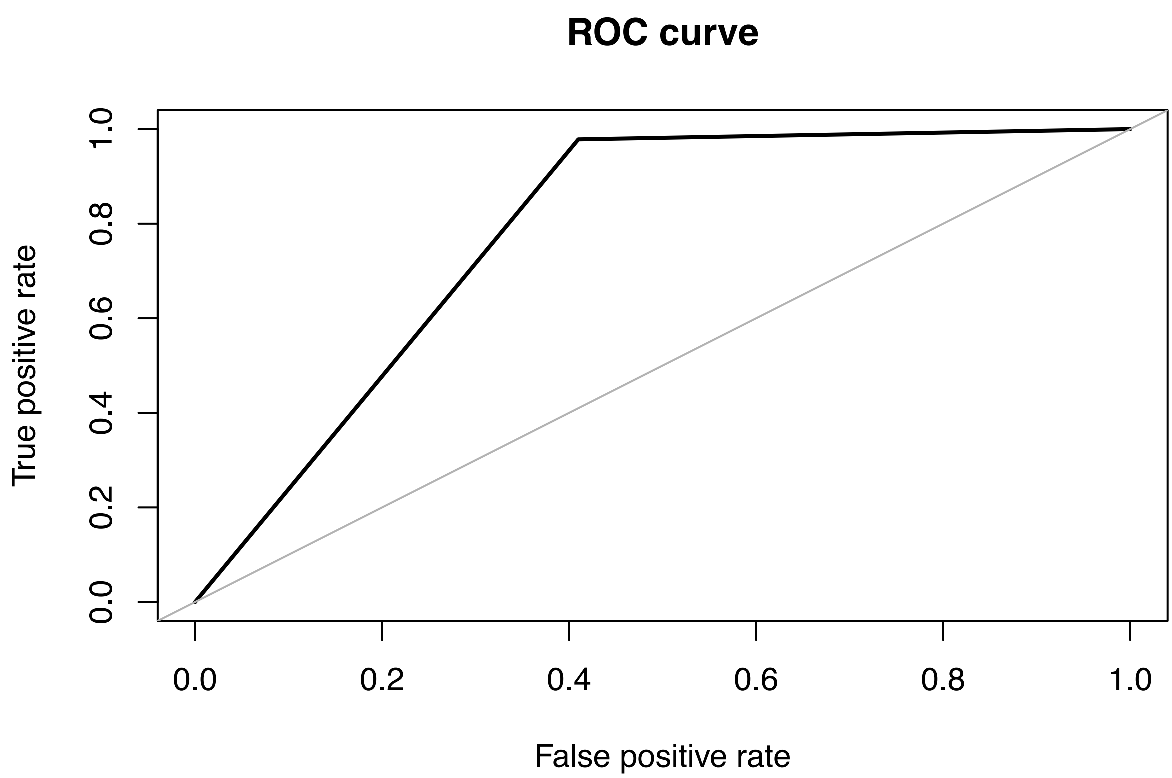
```
accuracy.meas(test$Response, y_predTree)
```

```
##
## Call:
## accuracy.meas(response = test$Response, predicted = y_predTree)
##
## Examples are labelled as positive when predicted is greater than 0.5
##
## precision: 0.247
## recall: 0.978
## F: 0.197
```

These metrics provide an interesting interpretation. With threshold value as 0.5, Precision = 0.247 says there are no false positives. Recall = 0.978 is very much high and indicates that we have lower number of false negatives as well. Threshold values can be altered also. F = 0.197 measn we have very accuracy of this model.

Recall in this context is also referred to as the true positive rate or sensitivity, and precision is also referred to as positive predictive value (PPV); other related measures used in classification include true negative rate and accuracy.True negative rate is also called specificity.

```
roc.curve(test$Response, y_predTree)
```




```
## Area under the curve (AUC): 0.784
```

```
library(class)
```

Knn model

```
set.seed(198)
knn_pred = knn(train = over[,-6],
               test = test[,-6],
               cl = over[,6],
                k = 4)
```

```
cm = table(test[, 6], knn_pred)
cm
```

```
##    knn_pred
##         0     1
##   0 18305  8499
##   1   611  3074
```

```
confusionMatrix(knn_pred,test$Response)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction     0     1
##          0 18305   611
##          1  8499  3074
##
##                Accuracy : 0.7012
##                  95% CI : (0.696, 0.7063)
##     No Information Rate : 0.8791
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2689
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6829
##             Specificity : 0.8342
##          Pos Pred Value : 0.9677
##          Neg Pred Value : 0.2656
##              Prevalence : 0.8791
##          Detection Rate : 0.6004
##    Detection Prevalence : 0.6204
##       Balanced Accuracy : 0.7586
##
##        'Positive' Class : 0
##
```

Conclusion

I have done the data exploration and visulization to have a basic statistc backgroud information of the data. Then did some data preparation for modeling, including check missing data, convert data variables for modeling, treat outliers issues. When do the first model, logistic regression, I found out that the model had overfitting issues and imbalanced classification. After solving these two big issues, would be able to generate several applicable models which all have more the 93% Sensitivity rate(recall rate, true positive). Decision tree, Naive bayes and logistic Regresion have the highest True Positive Rate (Sensitivity rate). I recommend the Insurance company use the logistic regression model due to the other two models may cost more on the daily usage in the field of business management and technical maintaining.