Data Science and Business Intel Project

```
library(tidyverse)
library(ggplot2)
library(ggROC)
library(cowplot)
library(reshape2)
library(car)
library(caret)
library(leaps)
library(bestglm)
library(plotly)
library(webshot)
library(DataExplorer)
library(purrr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(e1071)
library(pROC)
source("./[5]Script/Confusion_matrix.R")
source("./[5]Script/cutoff.R")
```

```
# Read in raw data
ds <- (read.csv("./[4]source/WA_Fn-UseC_-Telco-Customer-Churn.csv"))
ds$SeniorCitizen <- as.factor(ds$SeniorCitizen)
#########################################################################################
# Metadata
######################################################################################### Customer
# Service that each customer has signed up for
# Demographic information
# Customer Account Information
#########################################################################################
# Data Type:
#########################################################################################
# 16 Categorical Variables:
# - 6 Binary Variables (Gender, Senior Citizen, Partner, Dependents, Phone Service, Paperless 
# - 9 3-Factor level Variable (Multiple Lines, Internet Service, Online Security, Online Backu
# - 1 4-Factor level Variable (Payment Method)
#########################################################################################
# 3 Continious Variables:
# - Tenure, Monthly Charge, Total Charge
#########################################################################################
# 1 Target Variables:
# - Churn
#########################################################################################
```

Data Cleaning

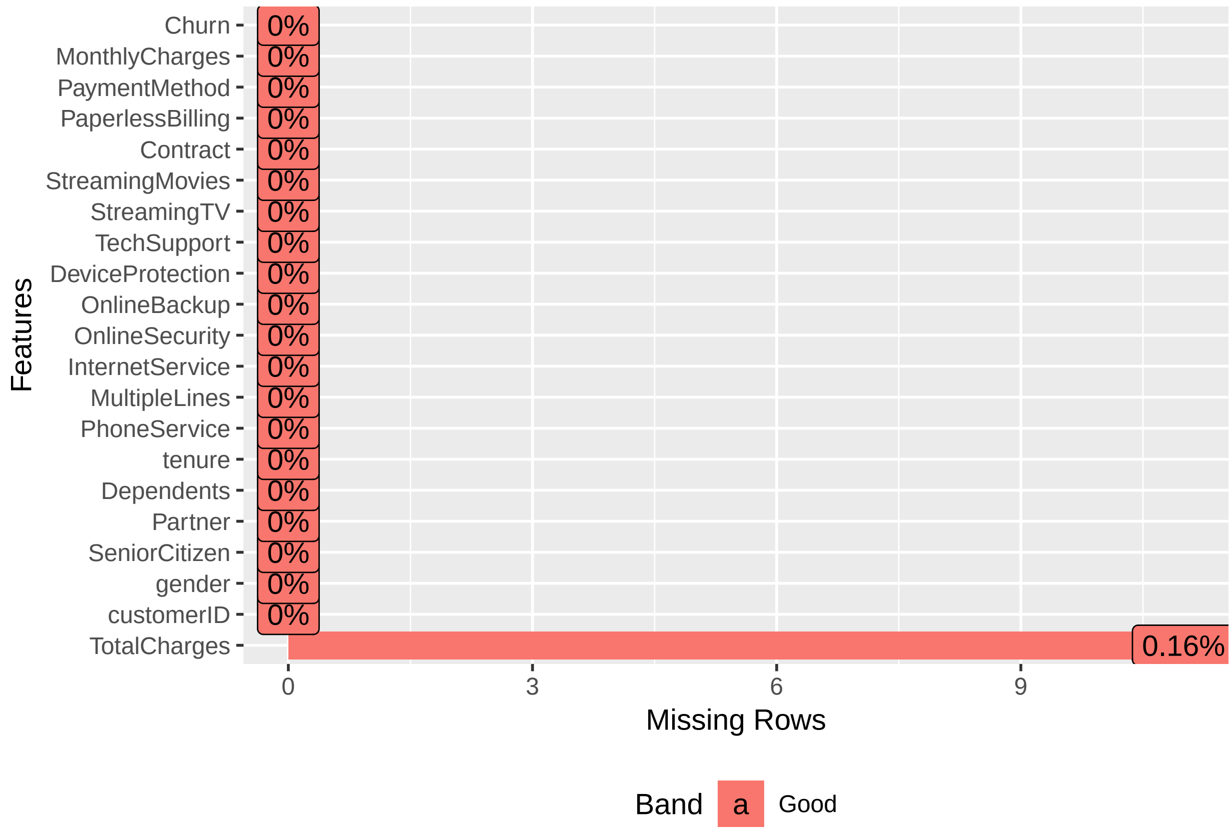
```r
# missing data analysis
# 1) We first check if missing exist within our dataset
print(paste0("The dataset contains missing data: ", any(is.na(ds))))
```

```
## [1] "The dataset contains missing data: TRUE"
```

```r
if (any(is.na(ds)) == "TRUE"){
      print(paste0("The total number of missing data(s) are: ", sum(is.na(ds))))
      print(paste0("The variable(s) with missing data(s) are: ", colnames(ds)[colSums(is.na(ds
}
```

```
## [1] "The total number of missing data(s) are: 11"
## [1] "The variable(s) with missing data(s) are: TotalCharges"
```

```r
plot_missing(ds)
```




```r
# 2) Filter the missing data into a its own dataset for further analysis
df_na <- ds[rowSums(is.na(ds))>0,]
df_na[c("gender","tenure","PhoneService","InternetService","Contract","MonthlyCharges","TotalCh
```

```
##      gender tenure PhoneService InternetService Contract MonthlyCharges
## 489  Female      0           No             DSL Two year          52.55
## 754    Male      0          Yes              No Two year          20.25
```

```
## 937  Female      0           Yes             DSL Two year          80.85
## 1083   Male      0           Yes              No Two year          25.75
## 1341 Female      0            No             DSL Two year          56.05
## 3332   Male      0           Yes              No Two year          19.85
## 3827   Male      0           Yes              No Two year          25.35
## 4381 Female      0           Yes              No Two year          20.00
## 5219   Male      0           Yes              No One year          19.70
## 6671 Female      0           Yes             DSL Two year          73.35
## 6755   Male      0           Yes             DSL Two year          61.90
##      TotalCharges Churn
## 489            NA    No
## 754            NA    No
## 937            NA    No
## 1083           NA    No
## 1341           NA    No
## 3332           NA    No
## 3827           NA    No
## 4381           NA    No
## 5219           NA    No
## 6671           NA    No
## 6755           NA    No
```

From the above missing data analysis, we are able to see out of the 7043 observation of 21 variables there are only 11 missing values and they are belong to the TOTAL CHARGES column(.16%), hence we are working with a pretty clean dataset.

An possible explaination for this mssing values is: (1) These customer never paid anything to the company (2) Tenure for all these customer are 0, thus meaning that this may be their first month with the company and thus the company hasn't charged them.

For these 11 missing data, we can either: (1) Impute the total charge value (2) Set total charge value to be zero (3) Remove them from the data set

Since we have a relatively large dataset, and that none of the customer with missing value have churn, thus for convience of the analysis, we will drop the 11 observation with missing TOTAL CHARGE. ## Data Exploration

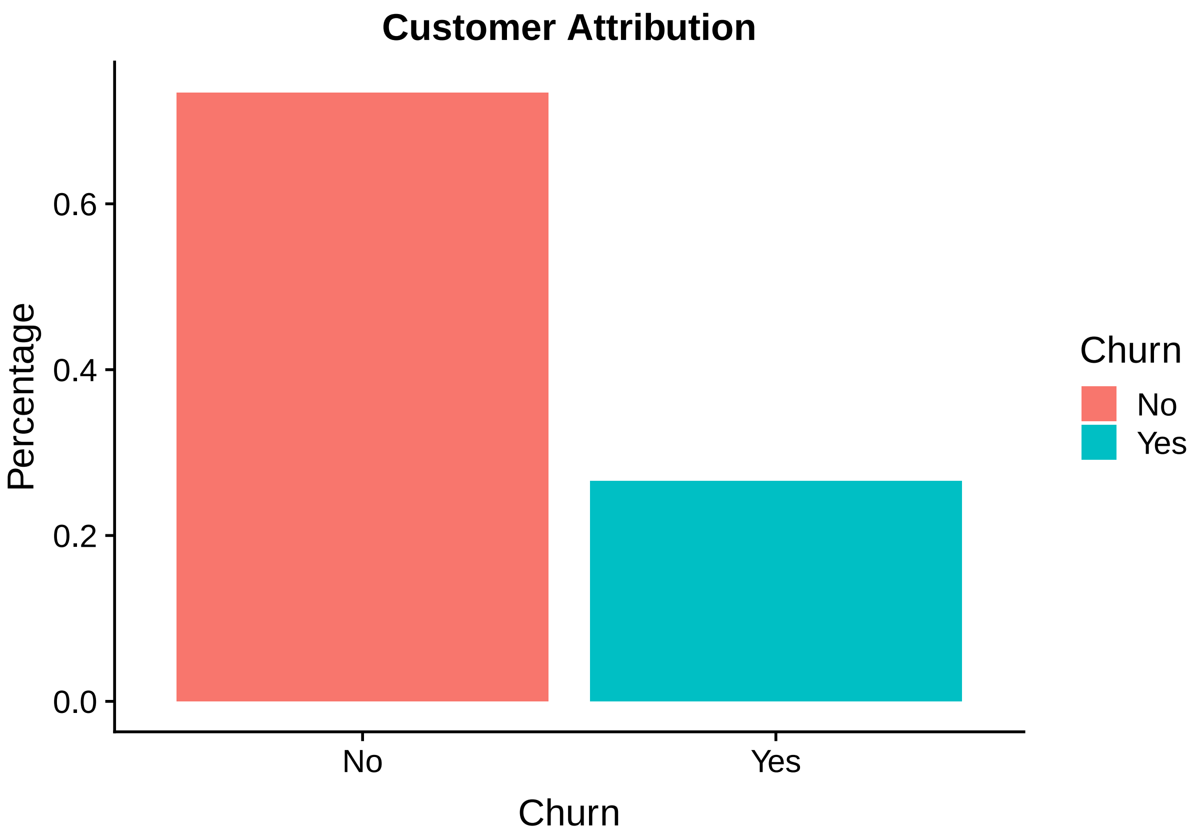
```r
df_clean <- ds %>%
            na.omit() %>%
            select(-1)

summary(df_clean)
```

```
##     gender     SeniorCitizen Partner    Dependents     tenure
##  Female:3483   0:5890        No :3639   No :4933   Min.   : 1.00
##  Male  :3549   1:1142        Yes:3393   Yes:2099   1st Qu.: 9.00
##                                                    Median :29.00
##                                                    Mean   :32.42
##                                                    3rd Qu.:55.00
##                                                    Max.   :72.00
```

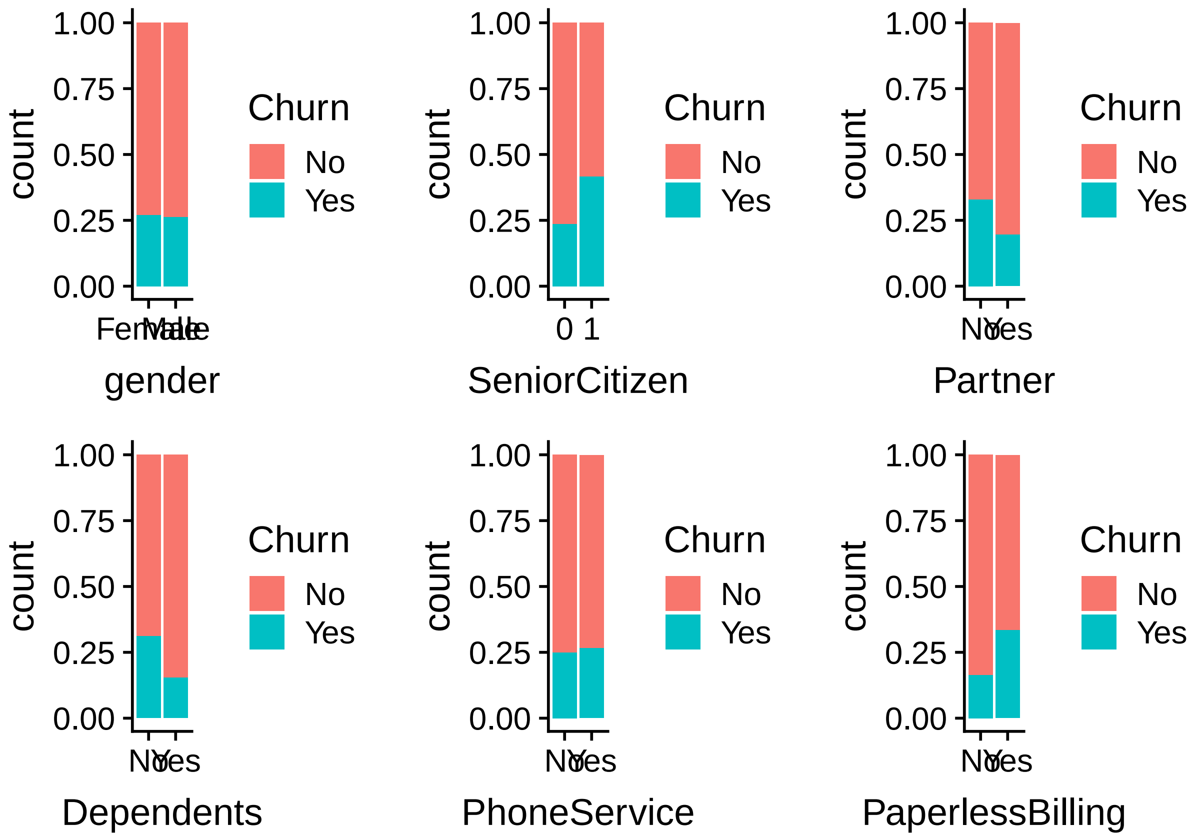
```
##  PhoneService          MultipleLines     InternetService
##  No : 680     No              :3385   DSL        :2416
##  Yes:6352     No phone service: 680   Fiber optic:3096
##               Yes             :2967   No         :1520
##
##
##
##              OnlineSecurity              OnlineBackup
##  No                 :3497   No                 :3087
##  No internet service:1520   No internet service:1520
##  Yes                :2015   Yes                :2425
##
##
##
##             DeviceProtection              TechSupport
##  No                 :3094    No                 :3472
##  No internet service:1520    No internet service:1520
##  Yes                :2418    Yes                :2040
##
##
##
##               StreamingTV              StreamingMovies
##  No                 :2809   No                 :2781
##  No internet service:1520   No internet service:1520
##  Yes                :2703   Yes                :2731
##
##
##
##            Contract     PaperlessBilling                   PaymentMethod
##  Month-to-month:3875   No :2864         Bank transfer (automatic):1542
##  One year      :1472   Yes:4168         Credit card (automatic)  :1521
##  Two year      :1685                    Electronic check         :2365
##                                         Mailed check             :1604
##
##
##  MonthlyCharges    TotalCharges    Churn
##  Min.   : 18.25   Min.   :  18.8   No :5163
##  1st Qu.: 35.59   1st Qu.: 401.4   Yes:1869
##  Median : 70.35   Median :1397.5
##  Mean   : 64.80   Mean   :2283.3
##  3rd Qu.: 89.86   3rd Qu.:3794.7
##  Max.   :118.75   Max.   :8684.8
```

```
# Binary variable distribution in Customer attribution
ggplot(data = df_clean, aes(x = Churn, y = (..count..)/sum(..count..), fill = Churn))+
      geom_bar()+
      ggtitle("Customer Attribution")+
      ylab("Percentage")
```

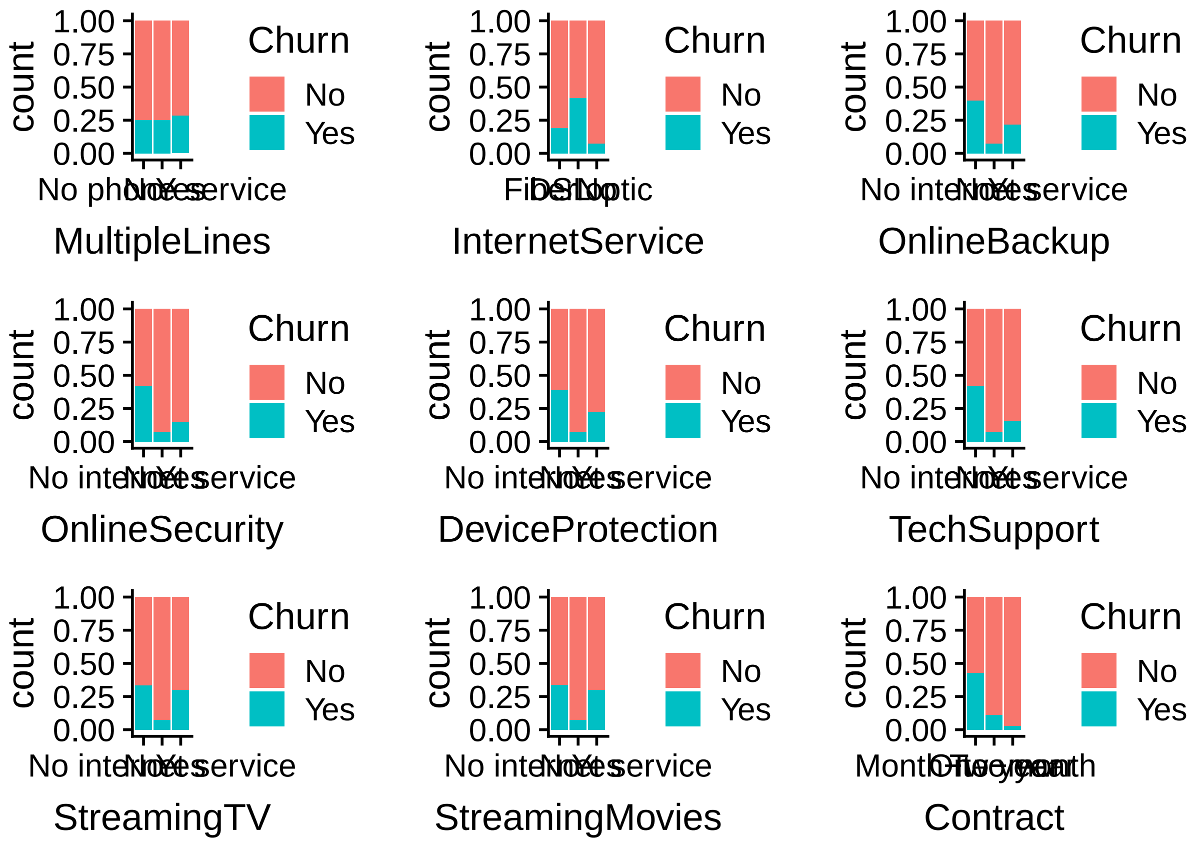


Of our dataset, 26% of the customer has left the platform within the past month

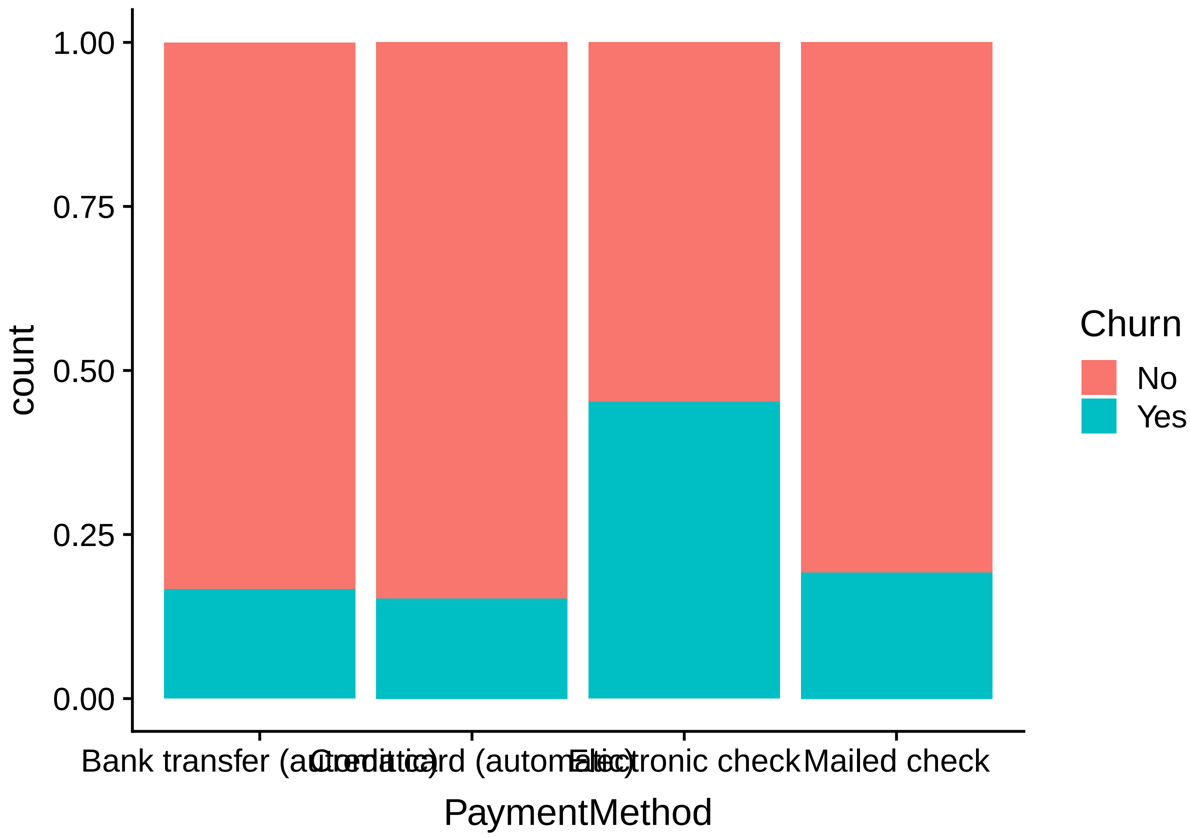
```
# Categorical Variable Analysis
# Binary binary variables Analysis
options(repr.plot.width = 12, repr.plot.height = 8)
plot_grid(
  ggplot(data = df_clean, aes(gender, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(SeniorCitizen, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(Partner, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(Dependents, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(PhoneService, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(PaperlessBilling, fill = Churn))+geom_bar(position = "fill")
)
```




```r
plot_grid(
  ggplot(data = df_clean, aes(MultipleLines, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(InternetService, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(OnlineBackup, fill = Churn))+geom_bar(position = "fill"),
   ggplot(data = df_clean, aes(OnlineSecurity, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(DeviceProtection, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(TechSupport, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(StreamingTV, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(StreamingMovies, fill = Churn))+geom_bar(position = "fill"),
  ggplot(data = df_clean, aes(Contract, fill = Churn))+geom_bar(position = "fill")
)
```



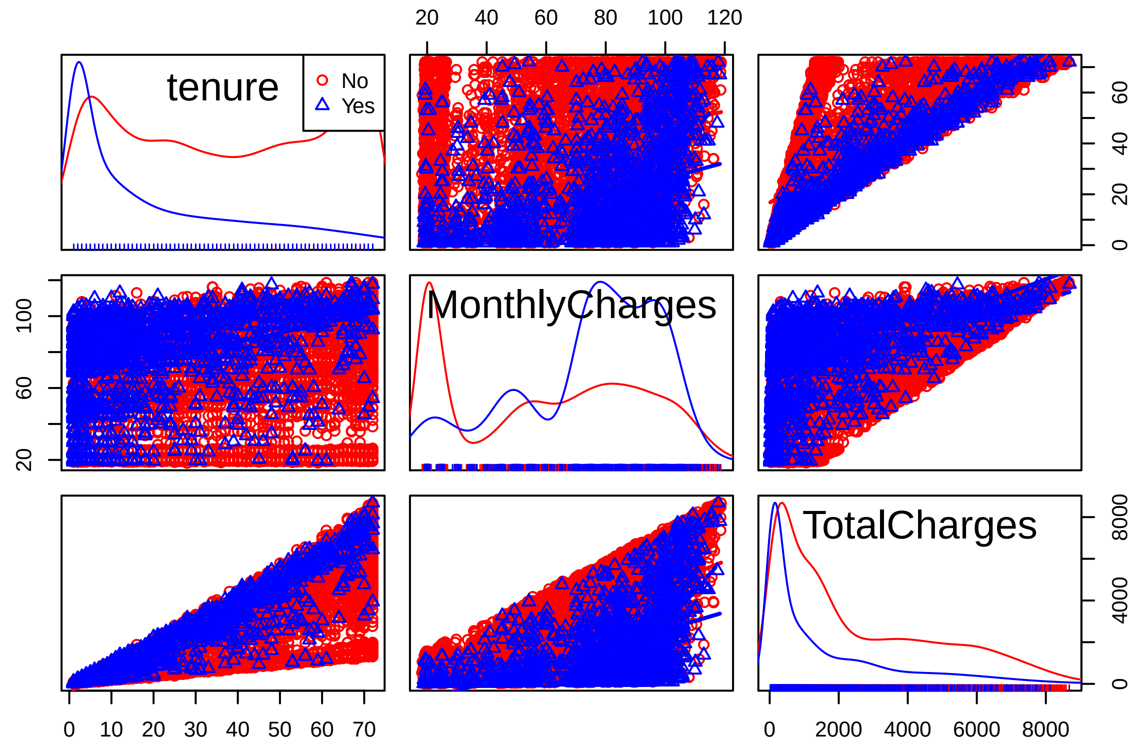

```r
ggplot(data = df_clean, aes(x=PaymentMethod, fill=Churn))+
  geom_bar(position = "fill")
```



Some Trends - Senior Citizens churn percentage are higher - Customer with dependents or partners tend to have lower churn rate compared to counterparts - Customer with paperless billing have higher churn rate - Customer with Fiber Optic Internet Service have significant higher churn rate - Customer with No online security, or online backup or tech support have higher churn rate - Customer with monthly subscription are more likely to churn compared to customer with one- or two-year contract - Customer with Electronic Check payment method tend to leave our client more compared to other options.

```
# # Continous Variable Analysis
# p <- plot_ly(df_clean,
#              x = ~MonthlyCharges,
#              y = ~TotalCharges,
#              z = ~tenure,
#              color = ~Churn,
#              marker = list(
#                size = 2)) %>%
#              add_markers() %>%
#              layout(scene = list(
#                xaxis = list(title = "Monthly Charges"),
#                yaxis = list(title = "Total Charges"),
#                zaxis = list(title = "Tenure")
#              ))
# p
```

```
# Correlation matrix of continous variable analysis (thank you very much)
scatterplotMatrix(~ tenure + MonthlyCharges + TotalCharges|Churn, data = df_clean, col = c("red
```



This appears to follow simple intuition, customer that are with the telecommunication firm for only a short period would in general have no loyalty compared to long time customer which are comfortable with the service provided and thus less willing to switch. Also, customer with higher monthly charges, will in general wish to reduce cost by seeking alternative service provider that may provide the same level of service for lower cost. From Figure 6, the scatter plots between continuous variables in general follows the trend described above and allow us to visually see if there are any obvious outliers which in this case seem to be none.

```
# Data Cleaning and Standardization
df_clean2 <- df_clean %>%
             mutate(MultipleLines=replace(MultipleLines,MultipleLines=="No phone service", "No
             mutate(OnlineSecurity=replace(OnlineSecurity,OnlineSecurity=="No internet service
             mutate(DeviceProtection=replace(DeviceProtection,DeviceProtection=="No internet se
             mutate(TechSupport=replace(TechSupport,TechSupport=="No internet service","No")) %
             mutate(StreamingTV=replace(StreamingTV,StreamingTV=="No internet service","No")) %
             mutate(StreamingMovies=replace(StreamingMovies,StreamingMovies=="No internet servi
             mutate(OnlineBackup=replace(OnlineBackup,OnlineBackup=="No internet service","No")
             mutate(tenure=scale(tenure)) %>%
             mutate(MonthlyCharges=scale(MonthlyCharges)) %>%
             mutate(TotalCharges=scale(TotalCharges))
```

```r
# Split data into training and validation split
set.seed(1994)
training <- sample(2,nrow(df_clean2),replace=TRUE,prob=c(.8,.2))
```

```r
# GLM Analysis
# Still need to look at threshold analysis
# Full GLM
df_clean.fulllogit <- glm(Churn~.,
                          family = binomial,
                          data = df_clean2[training==1,])

getinfo(df_clean.fulllogit,df_clean2)[c("confusion_matrix", "accuracy","sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed   0   1
##      No  770 251
##      Yes  84 278
##
## $accuracy
## [1] 0.757773
##
## $sensitivity
## [1] 0.7679558
```

```r
# Using Forward Approach to search for GLM model with lowest BIC

# tmp.modelsearch <- bestglm(df_clean2[training==1,],IC = "BIC", family = binomial, method = "


# Takes a long while (>= 4 to 6 hours)
# tmp.modelsearch$BestModels
# tmp.modelsearch$BestModel

# Best GLM Model
df_clean.bestlogit <- glm(Churn~
                            SeniorCitizen +
                            tenure +
                            PhoneService +
                            InternetService +
                            OnlineSecurity +
                            Contract +
                            PaperlessBilling +
                            PaymentMethod +
                            TotalCharges,
                          family = binomial,
                          data = df_clean2[training==1,])
```

```r
summary(df_clean.bestlogit)
```

```
##
## Call:
## glm(formula = Churn ~ SeniorCitizen + tenure + PhoneService +
##     InternetService + OnlineSecurity + Contract + PaperlessBilling +
##     PaymentMethod + TotalCharges, family = binomial, data = df_clean2[training ==
##     1, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7854  -0.6901  -0.2898   0.7639   3.5287
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           -1.15482    0.16519  -6.991 2.74e-12
## SeniorCitizen1                         0.36347    0.09087   4.000 6.33e-05
## tenure                                -1.60719    0.16355  -9.827  < 2e-16
## PhoneServiceYes                       -0.57854    0.14494  -3.991 6.57e-05
## InternetServiceFiber optic             0.84120    0.10752   7.824 5.13e-15
## InternetServiceNo                     -0.72024    0.14942  -4.820 1.43e-06
## OnlineSecurityYes                     -0.49034    0.09413  -5.209 1.90e-07
## ContractOne year                      -0.67067    0.11803  -5.682 1.33e-08
## ContractTwo year                      -1.39744    0.19416  -7.198 6.13e-13
## PaperlessBillingYes                    0.36526    0.08213   4.447 8.71e-06
## PaymentMethodCredit card (automatic) -0.08679    0.12595  -0.689  0.49081
## PaymentMethodElectronic check          0.33125    0.10477   3.162  0.00157
## PaymentMethodMailed check             -0.17305    0.12792  -1.353  0.17612
## TotalCharges                           0.87972    0.15954   5.514 3.50e-08
##
## (Intercept)                          ***
## SeniorCitizen1                       ***
## tenure                               ***
## PhoneServiceYes                      ***
## InternetServiceFiber optic           ***
## InternetServiceNo                    ***
## OnlineSecurityYes                    ***
## ContractOne year                     ***
## ContractTwo year                     ***
## PaperlessBillingYes                  ***
## PaymentMethodCredit card (automatic)
## PaymentMethodElectronic check        **
## PaymentMethodMailed check
## TotalCharges                         ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6553.1  on 5648  degrees of freedom
## Residual deviance: 4715.9  on 5635  degrees of freedom
## AIC: 4743.9
##
## Number of Fisher Scoring iterations: 6
```

```r
vif(df_clean.bestlogit)
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## SeniorCitizen     1.080908  1        1.039667
## tenure           14.617958  1        3.823344
## PhoneService      1.401480  1        1.183841
## InternetService   2.402199  2        1.244951
## OnlineSecurity    1.127523  1        1.061849
## Contract          1.521245  2        1.110580
## PaperlessBilling  1.113757  1        1.055347
## PaymentMethod     1.343729  3        1.050474
## TotalCharges     16.492869  1        4.061141
```

```r
getinfo(df_clean.bestlogit,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed   0   1
##      No  773 248
##      Yes  82 280
##
## $accuracy
## [1] 0.7613883
##
## $sensitivity
## [1] 0.7734807
```

```r
# Remove Total Charges due to high VIF value (>2, thus multi-colinearity effect)

df_clean.bestlogit2 <- glm(Churn~
                            SeniorCitizen +
                            tenure +
                            PhoneService +
                            InternetService +
                            OnlineSecurity +
                            Contract +
                            PaperlessBilling +
                            PaymentMethod,
                          family = binomial,
                          data = df_clean2[training==1,])
summary(df_clean.bestlogit2)
```

```
##
## Call:
## glm(formula = Churn ~ SeniorCitizen + tenure + PhoneService +
##     InternetService + OnlineSecurity + Contract + PaperlessBilling +
##     PaymentMethod, family = binomial, data = df_clean2[training ==
##     1, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8218  -0.6731  -0.3068   0.7653   3.1154
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           -1.31487    0.15882  -8.279  < 2e-16
## SeniorCitizen1                         0.37046    0.09132   4.057 4.98e-05
## tenure                                -0.78160    0.05612 -13.927  < 2e-16
## PhoneServiceYes                       -0.40597    0.13731  -2.957 0.003111
## InternetServiceFiber optic             1.07104    0.09861  10.862  < 2e-16
## InternetServiceNo                     -0.85564    0.14626  -5.850 4.91e-09
## OnlineSecurityYes                     -0.44134    0.09379  -4.706 2.53e-06
## ContractOne year                      -0.63739    0.11659  -5.467 4.58e-08
## ContractTwo year                      -1.30311    0.19059  -6.837 8.07e-12
## PaperlessBillingYes                    0.37475    0.08173   4.585 4.53e-06
## PaymentMethodCredit card (automatic) -0.08768    0.12574  -0.697 0.485626
## PaymentMethodElectronic check          0.35246    0.10472   3.366 0.000763
## PaymentMethodMailed check             -0.11915    0.12660  -0.941 0.346609
##
## (Intercept)                           ***
## SeniorCitizen1                        ***
## tenure                                ***
## PhoneServiceYes                       **
## InternetServiceFiber optic            ***
## InternetServiceNo                     ***
## OnlineSecurityYes                     ***
## ContractOne year                      ***
## ContractTwo year                      ***
## PaperlessBillingYes                   ***
## PaymentMethodCredit card (automatic)
## PaymentMethodElectronic check         ***
## PaymentMethodMailed check
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6553.1  on 5648  degrees of freedom
## Residual deviance: 4748.6  on 5636  degrees of freedom
## AIC: 4774.6
```

```
##
## Number of Fisher Scoring iterations: 6
```

```r
vif(df_clean.bestlogit2)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## SeniorCitizen    1.081676  1        1.040037
## tenure           1.662398  1        1.289340
## PhoneService     1.358571  1        1.165577
## InternetService  1.924459  2        1.177815
## OnlineSecurity   1.113718  1        1.055328
## Contract         1.481751  2        1.103300
## PaperlessBilling 1.108524  1        1.052865
## PaymentMethod    1.318029  3        1.047098
```

```r
getinfo(df_clean.bestlogit2,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed   0   1
##      No  780 241
##      Yes  87 275
##
## $accuracy
## [1] 0.7628344
##
## $sensitivity
## [1] 0.7596685
```

```r
# Decision Tree Analysis
df_clean.fulltree <- rpart(Churn ~.,
                       data = df_clean2[training==1,], method = "class",
                       control = rpart.control(cp=0))

getinfo(df_clean.fulltree,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed   0   1
##      No  811 210
##      Yes 127 235
##
## $accuracy
## [1] 0.7563268
##
## $sensitivity
## [1] 0.6491713
```

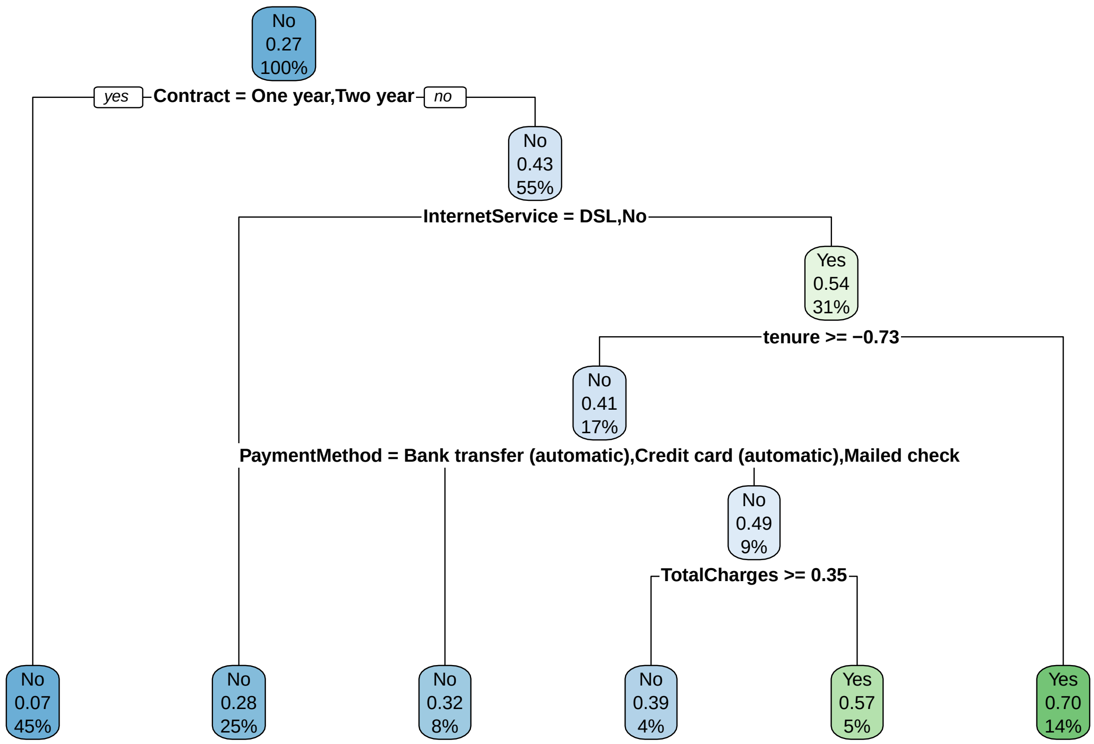
```r
# Hyperparameter Tuning
```

```r
# plotcp(df_clean.fulltree)
printcp(df_clean.fulltree)
```

```
##
## Classification tree:
## rpart(formula = Churn ~ ., data = df_clean2[training == 1, ],
##     method = "class", control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
##  [1] Contract         Dependents       DeviceProtection gender
##  [5] InternetService  MonthlyCharges   MultipleLines    OnlineBackup
##  [9] OnlineSecurity   PaperlessBilling Partner          PaymentMethod
## [13] PhoneService     SeniorCitizen    StreamingMovies  StreamingTV
## [17] TechSupport      tenure           TotalCharges
##
## Root node error: 1507/5649 = 0.26677
##
## n= 5649
##
##            CP nsplit rel error  xerror     xstd
## 1  0.07011723      0   1.00000 1.00000 0.022058
## 2  0.01360319      3   0.78965 0.79761 0.020412
## 3  0.00398142      5   0.76244 0.81287 0.020553
## 4  0.00265428     10   0.73723 0.80027 0.020437
## 5  0.00248839     18   0.71267 0.79429 0.020381
## 6  0.00232250     22   0.70272 0.79429 0.020381
## 7  0.00199071     31   0.67750 0.79496 0.020387
## 8  0.00176952     32   0.67551 0.79695 0.020406
## 9  0.00165893     49   0.63504 0.79695 0.020406
## 10 0.00149303     61   0.61447 0.79628 0.020400
## 11 0.00132714     65   0.60849 0.80226 0.020455
## 12 0.00110595     75   0.59456 0.80624 0.020492
## 13 0.00099536     81   0.58792 0.80823 0.020510
## 14 0.00088476     91   0.57664 0.80956 0.020523
## 15 0.00082946    107   0.55740 0.80956 0.020523
## 16 0.00079628    113   0.55209 0.81221 0.020547
## 17 0.00066357    120   0.54612 0.82681 0.020679
## 18 0.00049768    126   0.54214 0.82681 0.020679
## 19 0.00044238    137   0.53417 0.84472 0.020838
## 20 0.00033179    140   0.53285 0.84472 0.020838
## 21 0.00026543    142   0.53218 0.85468 0.020924
## 22 0.00022119    147   0.53086 0.86662 0.021027
## 23 0.00016589    150   0.53019 0.86662 0.021027
## 24 0.00013271    159   0.52820 0.86662 0.021027
## 25 0.00000000    164   0.52754 0.86662 0.021027
```

```r
tmp <- df_clean.fulltree$cptable[which.min(df_clean.fulltree$cptable[,"xerror"]),]
```

```r
# Prune the tree
df_clean.besttree <- prune(df_clean.fulltree,cp = 0.01)
rpart.plot(df_clean.besttree)
```




```r
getinfo(df_clean.besttree,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##          predicted
## observed   0   1
##       No  844 177
##       Yes 144 218
##
## $accuracy
## [1] 0.7678959
##
## $sensitivity
## [1] 0.6022099
```

```r
# Random Forest
set.seed(1994)
df_clean.rforest <- randomForest(Churn~.,
                                 data = df_clean2[training==1,],
                                 ntree=500,                        # dataset
                                 cutoff=c(0.5,0.5),
```

```r
                                    mtry=2,
                                    importance=TRUE)
df_clean.rforest
```

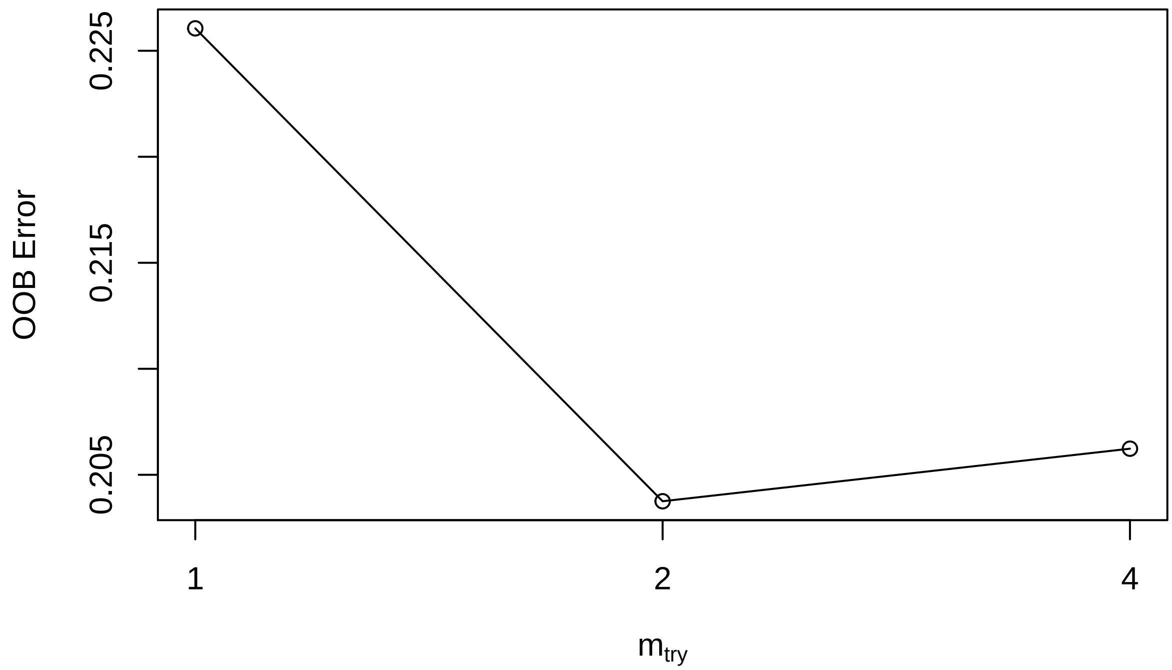
```
##
## Call:
##  randomForest(formula = Churn ~ ., data = df_clean2[training ==      1, ], ntree = 500, cuto
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 20.46%
## Confusion matrix:
##       No Yes class.error
## No  3789 353  0.08522453
## Yes  803 704  0.53284672
```

```r
# Confusion Matrix Test
getinfo(df_clean.rforest,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed   0   1
##      No  837 184
##      Yes 119 243
##
## $accuracy
## [1] 0.7809111
##
## $sensitivity
## [1] 0.6712707
```

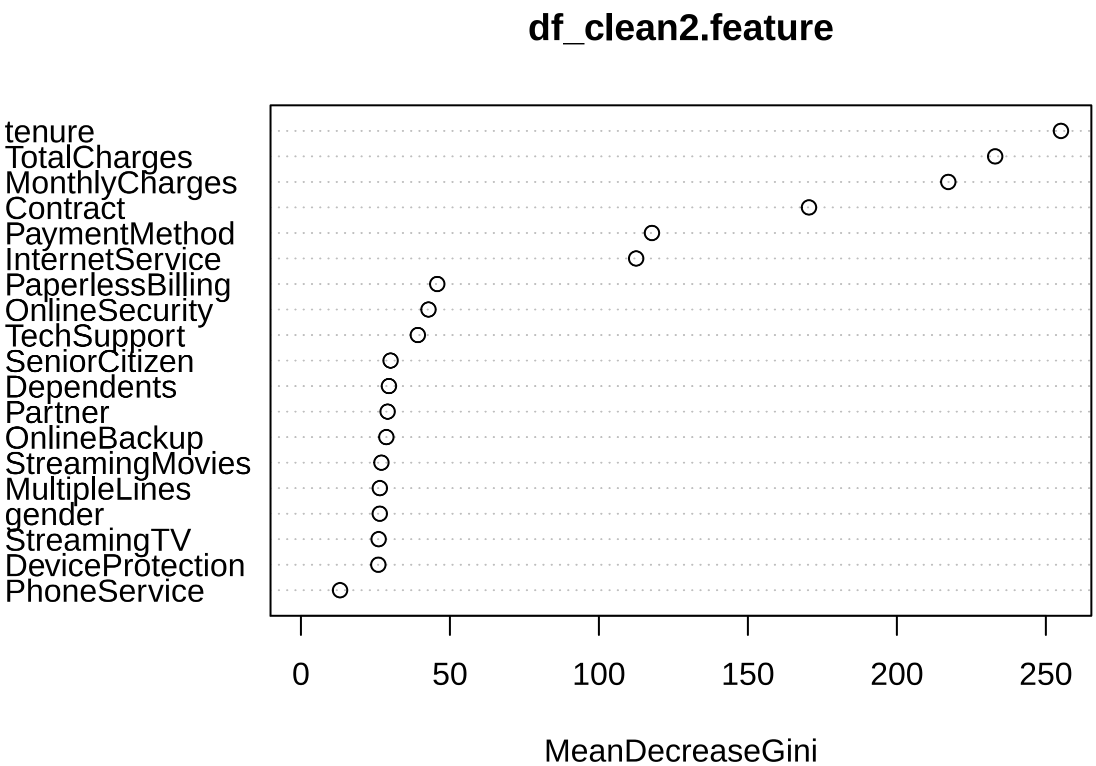
```r
# Hyperparameter Tuning
set.seed(1994)
rforest.tune <- tuneRF(x = df_clean2[training==1,]%>%select(-Churn),
                       y = df_clean2[training==1,]$Churn,mtryStart=2,
                       ntreeTry = 500)
```

```
## mtry = 2  OOB error = 20.38%
## Searching left ...
## mtry = 1     OOB error = 22.61%
## -0.10947 0.05
## Searching right ...
## mtry = 4     OOB error = 20.62%
## -0.01216334 0.05
```




```r
# Feature Importance Analysis
# generateFilterValuesData(task, "randomForest.importance")
df_clean2.feature <- randomForest(Churn~., data = df_clean2, importance = FALSE, ntree = 500, m

varImpPlot(df_clean2.feature)
```




```r
# SVM
df_clean.svm <- svm(Churn~.,
                    data = df_clean2[training==1,],
                    kernel = "linear",
                    cost = 0.01,
                    proability = TRUE)

getinfo(df_clean.svm,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##          predicted
## observed  No Yes
##      No  919 102
##      Yes 179 183
##
## $accuracy
## [1] 0.7968185
##
## $sensitivity
## [1] 0.5055249
```

```r
# Hyperparameter Tuning
svm.tune <- tune(svm,
                 Churn~.,
```

```r
                 data = df_clean2[training==1,],
                 kernel = "linear",
                 ranges = list(cost = 10^(-5:0)))

print(svm.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1993228
```

```r
svm.tune$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = Churn ~ ., data = df_clean2[training ==
##     1, ], ranges = list(cost = 10^(-5:0)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.03225806
##
## Number of Support Vectors:  2589
```

```r
df_clean.bestsvm <- svm(Churn~.,
                        data = df_clean2[training==1,],
                        kernel = "linear",
                        cost = 0.1,
                        probaility = TRUE)
summary(df_clean.bestsvm)
```

```
##
## Call:
## svm(formula = Churn ~ ., data = df_clean2[training == 1, ], kernel = "linear",
##     cost = 0.1, probaility = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  0.1
##       gamma:  0.03225806
##
## Number of Support Vectors:  2608
##
##  ( 1308 1300 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

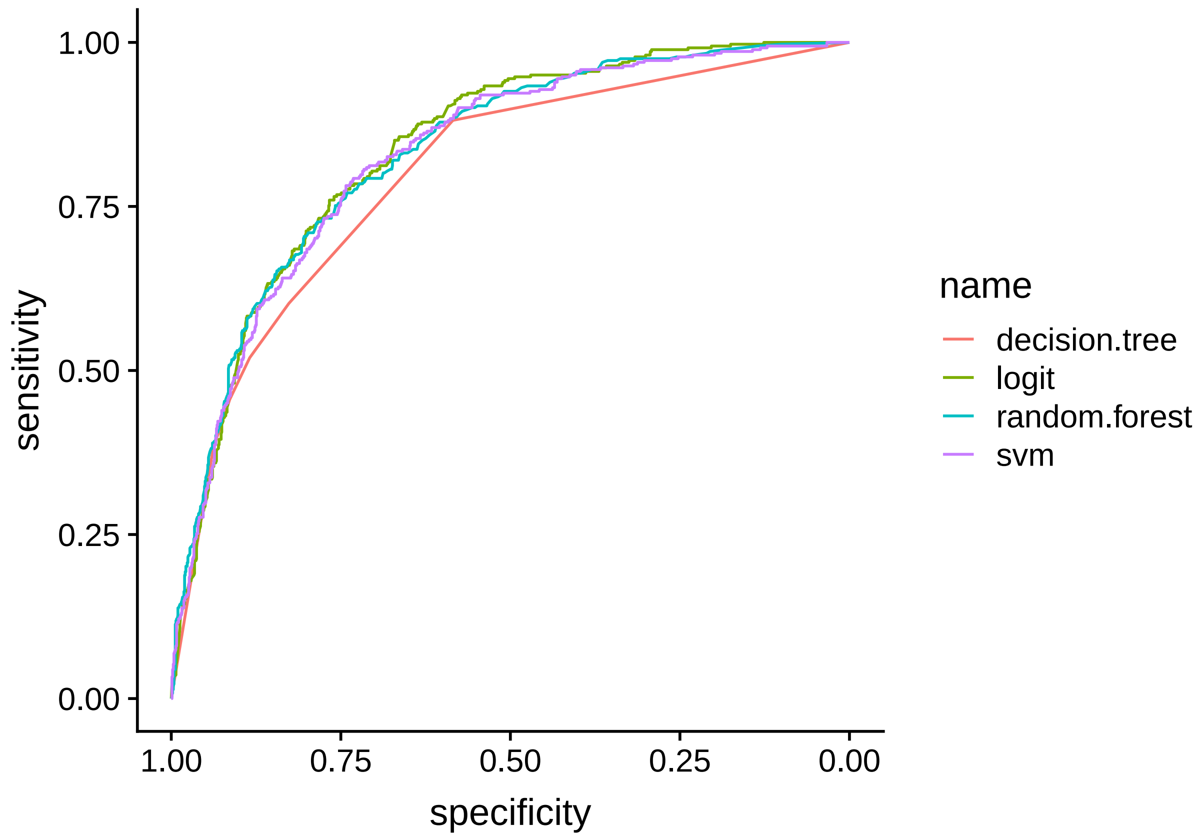
```r
getinfo(df_clean.bestsvm,df_clean2)[c("confusion_matrix", "accuracy", "sensitivity")]
```

```
## $confusion_matrix
##         predicted
## observed  No Yes
##      No  915 106
##      Yes 178 184
##
## $accuracy
## [1] 0.7946493
##
## $sensitivity
## [1] 0.5082873
```

```r
# Performance evaluation - Learning Curves and Fitted Graphs
# AUC Curve
# First assemble the probability matrix
prob_matrix <- data.frame(
               "logit" = predict(df_clean.bestlogit2,df_clean2[training==2,],type = "response")
               "d_tree" = predict(df_clean.besttree, df_clean2[training==2,],type="prob")[,2],
               "r_forest" = predict(df_clean.rforest, df_clean2[training==2,], type = "prob")[
               "svm" = as.numeric(attr(predict(df_clean.bestsvm, df_clean2[training==2,], decis
               )
# Create the ROC Varible

logit.roc <- roc(df_clean2$Churn[training==2],prob_matrix$logit)
d_tree.roc <- roc(df_clean2$Churn[training==2],prob_matrix$d_tree)
r_forest.roc <- roc(df_clean2$Churn[training==2],prob_matrix$r_forest)
svm.roc <- roc(df_clean2$Churn[training==2],prob_matrix$svm)

ggroc(list(logit=logit.roc,decision.tree=d_tree.roc,random.forest=r_forest.roc,svm=svm.roc),leg
```




```
tmp4 <- c(logit.roc$auc,d_tree.roc$auc,r_forest.roc$auc,svm.roc$auc)

tmp5 <- data.frame(
  "AUC" = tmp4
)
row.names(tmp5)<- c("Logit","Decision Tree", "Random Forest","SVM")

tmp5
```

```
##                     AUC
## Logit         0.8358193
## Decision Tree 0.7981680
## Random Forest 0.8314863
## SVM           0.8270775
```

A small discussion about cutoff point:

As we are attempting to identify customer that are going to churn, we thus need to focus on sensitivity metric compared to accuracy. As it is comparitively more expensive to acquire customer than retain customer, thus we are not as concern with false positive, but rather concerned with false negative. We would idealy like a model that is able to successful target all customer that are going to churn, and it should matter less if we have a higher number of false postive to us a telcommunication company. Thus we should have a lower threshold value than 0.5, though the actual value often require domain knowledge which we lack, thus we are going to use a more objective method to set out threshold value.

```r
# Lets use logistic regression as it has the largest AUC out of all three method

# output <- matrix(0,100,3)
# x_axis <- seq(0.01,0.8,length=100)
#
# for (i in 1:100)
# {
#   output[i,]=threshold(x_axis[i])
# }
#
# plot(x_axis,output[,1], type = "l", col = "darkgreen", xlab = "Threshold Value", ylab = "Val
# lines(x_axis,output[,2],col = "red")
# lines(x_axis,output[,3], col = "blue")
# legend("bottom",col=c(2,"darkgreen",4,"darkred"),text.font =3,inset = 0.02,
#        box.lty=0,cex = 0.8,
#        lwd=c(2,2,2,2),c("Specificity","Senitivity","Accuracy"))
#
#
# x_axis[which(abs(output[,1]-output[,2])<0.01)]
```