Customer Churning Behavior

Vinh Ha Duc

April 22, 2019

Summary

This is the dataset regarding the Churning behavior of Telco customer. In this report, I will examine the dataset and trying to find the model that would best predict the churning behavior before it actually happen, so Teclco company might be able to reach out to customer with better service/improvement to retain them.

This document is part of the course work for edx Havardx Data Science Capstone

Method/Analysis

There are 7 methods that will be use for our analysis/fitting and the best one will be recommended at the conclusion section of this document.

The details steps include:

Step 1: Downloading Customer Churning Data

Step 2: Examine Looks and Feel of Data/ Factors that Can be used for analysis

Step 3: Data Cleaning/Grouping in Preparation for used in later models

Step 4: Data Visualization

Step 5: Factor To be used

Step 6: Splitting Data into Training set (75% of data) and Test set (25% of Data)

Step 7: Train data with Logistic Regression

Step 8: Traing data with Linear ALgoithm LDA

Step 9: Train data with CART

Step 10: Train data with K-nn

Step 11: Train data with SVM

Step 12: Train data with Random Forest

Step 13: Train data with GBM

Step 14: Looks at Accuracy of each algorithm

Step 15: Looks at the best model that fit our data

Step 16: Conclustion.

Exploration and Results

Step 1: Downloading Customer Churning Data

The data set is taken from Kaggle and stored in Github.

Data downloaded is stored in data frame

```
#Loading required Library to be used
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggplot2
```

```
if(!require(lattice)) install.packages("lattice", repos = "http://cran.us.r-project.org")
```

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

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages -------------------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  2.0.1       v purrr   0.3.1
## v tidyr   0.8.3       v dplyr   0.8.0.1
## v readr   1.3.1       v stringr 1.4.0
## v tibble  2.0.1       v forcats 0.4.0
```

```
## -- Conflicts ----------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```r
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: plyr
```

```
## -------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```r
if(!require(RCurl)) install.packages("RCurl", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

```
##
## Attaching package: 'RCurl'
```

```
## The following object is masked from 'package:tidyr':
##
##     complete
```

```r
if(!require(foreign)) install.packages("foreign", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: foreign
```

```r
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(grid)) install.packages("grid", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: grid
```

```
if(!require(pROC)) install.packages("pROC", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: pROC
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
#Reading data from URL, the data is taken from Kaggle and Stored in Github
url <- 'https://raw.githubusercontent.com/haducvinh/customer_retention/master/WA_Fn-UseC_-Telco-Customer-Churn.
csv'
url_dat <- getURL(url)

all_data <- read.csv('C:/Users/vinh.haduc/Downloads/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.csv',
 stringsAsFactors = FALSE, na.strings = c("NA", ""))
all_data <- read_csv( url_dat)
```

Step 2: Examine Looks and Feel of Data/ Factors that Can be used for analysis

I will first take a look at the first few line of data to have a feeling of how the data looks like

The next step is to examine the columns that data has and the type of data each column contain.

```
#Look And feel of data
head(all_data)
```

```
## # A tibble: 6 x 21
##   customerID gender SeniorCitizen Partner Dependents tenure PhoneService
##   <chr>      <chr>          <dbl> <chr>   <chr>       <dbl> <chr>
## 1 7590-VHVEG Female             0 Yes     No              1 No
## 2 5575-GNVDE Male               0 No      No             34 Yes
## 3 3668-QPYBK Male               0 No      No              2 Yes
## 4 7795-CFOCW Male               0 No      No             45 No
## 5 9237-HQITU Female             0 No      No              2 Yes
## 6 9305-CDSKC Female             0 No      No              8 Yes
## # ... with 14 more variables: MultipleLines <chr>, InternetService <chr>,
## #   OnlineSecurity <chr>, OnlineBackup <chr>, DeviceProtection <chr>,
## #   TechSupport <chr>, StreamingTV <chr>, StreamingMovies <chr>,
## #   Contract <chr>, PaperlessBilling <chr>, PaymentMethod <chr>,
## #   MonthlyCharges <dbl>, TotalCharges <dbl>, Churn <chr>
```

```
sapply(all_data, typeof)
```

```
##       customerID           gender    SeniorCitizen          Partner
##      "character"      "character"         "double"      "character"
##       Dependents           tenure     PhoneService    MultipleLines
##      "character"         "double"      "character"      "character"
##  InternetService   OnlineSecurity     OnlineBackup DeviceProtection
##      "character"      "character"      "character"      "character"
##      TechSupport      StreamingTV  StreamingMovies         Contract
##      "character"      "character"      "character"      "character"
## PaperlessBilling    PaymentMethod   MonthlyCharges     TotalCharges
##      "character"      "character"         "double"         "double"
##            Churn
##      "character"
```

```
str(all_data)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 7043 obs. of  21 variables:
##  $ customerID      : chr  "7590-VHVEG" "5575-GNVDE" "3668-QPYBK" "7795-CFOCW" ...
##  $ gender          : chr  "Female" "Male" "Male" "Male" ...
##  $ SeniorCitizen   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Partner         : chr  "Yes" "No" "No" "No" ...
##  $ Dependents      : chr  "No" "No" "No" "No" ...
##  $ tenure          : num  1 34 2 45 2 8 22 10 28 62 ...
##  $ PhoneService    : chr  "No" "Yes" "Yes" "No" ...
##  $ MultipleLines   : chr  "No phone service" "No" "No" "No phone service" ...
##  $ InternetService : chr  "DSL" "DSL" "DSL" "DSL" ...
##  $ OnlineSecurity  : chr  "No" "Yes" "Yes" "Yes" ...
##  $ OnlineBackup    : chr  "Yes" "No" "Yes" "No" ...
##  $ DeviceProtection: chr  "No" "Yes" "No" "Yes" ...
##  $ TechSupport     : chr  "No" "No" "No" "Yes" ...
##  $ StreamingTV     : chr  "No" "No" "No" "No" ...
##  $ StreamingMovies : chr  "No" "No" "No" "No" ...
##  $ Contract        : chr  "Month-to-month" "One year" "Month-to-month" "One year" ...
##  $ PaperlessBilling: chr  "Yes" "No" "Yes" "No" ...
##  $ PaymentMethod   : chr  "Electronic check" "Mailed check" "Mailed check" "Bank transfer (automatic)" ...
##  $ MonthlyCharges  : num  29.9 57 53.9 42.3 70.7 ...
##  $ TotalCharges    : num  29.9 1889.5 108.2 1840.8 151.7 ...
##  $ Churn           : chr  "No" "No" "Yes" "No" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   customerID = col_character(),
##   ..   gender = col_character(),
##   ..   SeniorCitizen = col_double(),
##   ..   Partner = col_character(),
##   ..   Dependents = col_character(),
##   ..   tenure = col_double(),
##   ..   PhoneService = col_character(),
##   ..   MultipleLines = col_character(),
##   ..   InternetService = col_character(),
##   ..   OnlineSecurity = col_character(),
##   ..   OnlineBackup = col_character(),
##   ..   DeviceProtection = col_character(),
##   ..   TechSupport = col_character(),
##   ..   StreamingTV = col_character(),
##   ..   StreamingMovies = col_character(),
##   ..   Contract = col_character(),
##   ..   PaperlessBilling = col_character(),
##   ..   PaymentMethod = col_character(),
##   ..   MonthlyCharges = col_double(),
##   ..   TotalCharges = col_double(),
##   ..   Churn = col_character()
##   .. )
```

Step 3: Data Cleaning/Grouping in Preparation for used in later models

The data cleaning would involve removing not relevant NA data, creating grouping for later analysis in data training and chaning data to factor for processing.

```r
#Data Cleaning/Grouping/ Data preparation

#Remove NA/ Show the percentage of NA data
sapply(all_data[,-c(2)], function(x) round((sum(is.na(x))/length(x)*100),2))
```

```
##       customerID    SeniorCitizen          Partner       Dependents 
##             0.00             0.00             0.00             0.00 
##           tenure     PhoneService    MultipleLines  InternetService 
##             0.00             0.00             0.00             0.00 
##   OnlineSecurity     OnlineBackup DeviceProtection      TechSupport 
##             0.00             0.00             0.00             0.00 
##      StreamingTV  StreamingMovies         Contract PaperlessBilling 
##             0.00             0.00             0.00             0.00 
##    PaymentMethod   MonthlyCharges     TotalCharges            Churn 
##             0.00             0.00             0.16             0.00
```

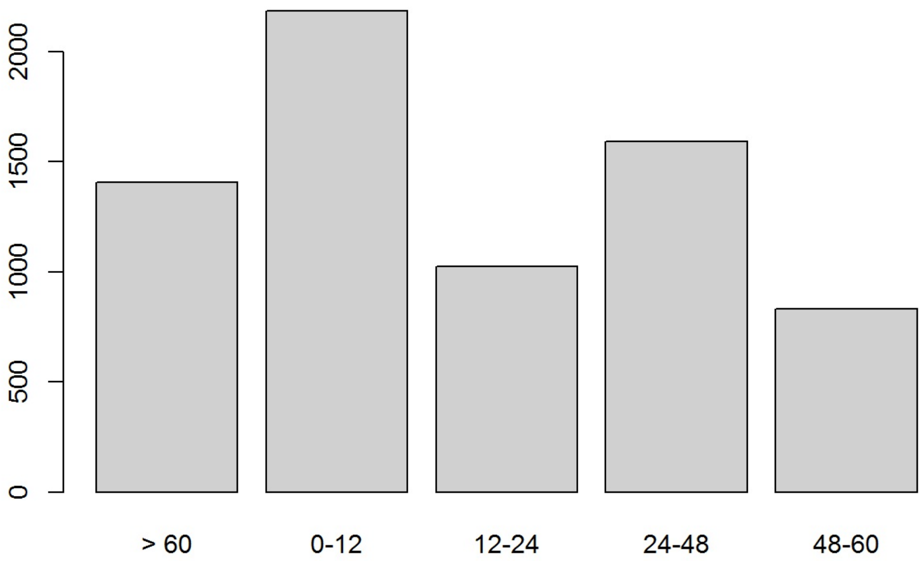
```r
#Probing data values for grouping
paste("The Minimum value is: ",min(all_data$tenure)," and the Maximum value is: ",max(all_data$tenure))
```

```
## [1] "The Minimum value is:  0  and the Maximum value is:  72"
```

```r
# create the goupping function based on maxium and minimum value discovered above
F_Grouping <- function(tn){
  if (tn >= 0 & tn <= 12){
    return('0-12')
  }else if(tn > 12 & tn <= 24){
    return('12-24')
  }else if (tn > 24 & tn <= 48){
    return('24-48')
  }else if (tn > 48 & tn <=60){
    return('48-60')
  }else if (tn > 60){
    return('> 60')
  }
}

# Apply grouping function
all_data$GrpTenure <- sapply(all_data$tenure,F_Grouping)
# Set the new column as factor
all_data$GrpTenure <- as.factor(all_data$GrpTenure)
# Checking the grouping bins frequency
table(all_data$GrpTenure)
```

```
## 
##  > 60  0-12 12-24 24-48 48-60 
##  1407  2186  1024  1594   832
```

```r
barplot(table(all_data$GrpTenure), col=c("#d0d0d0"))
```



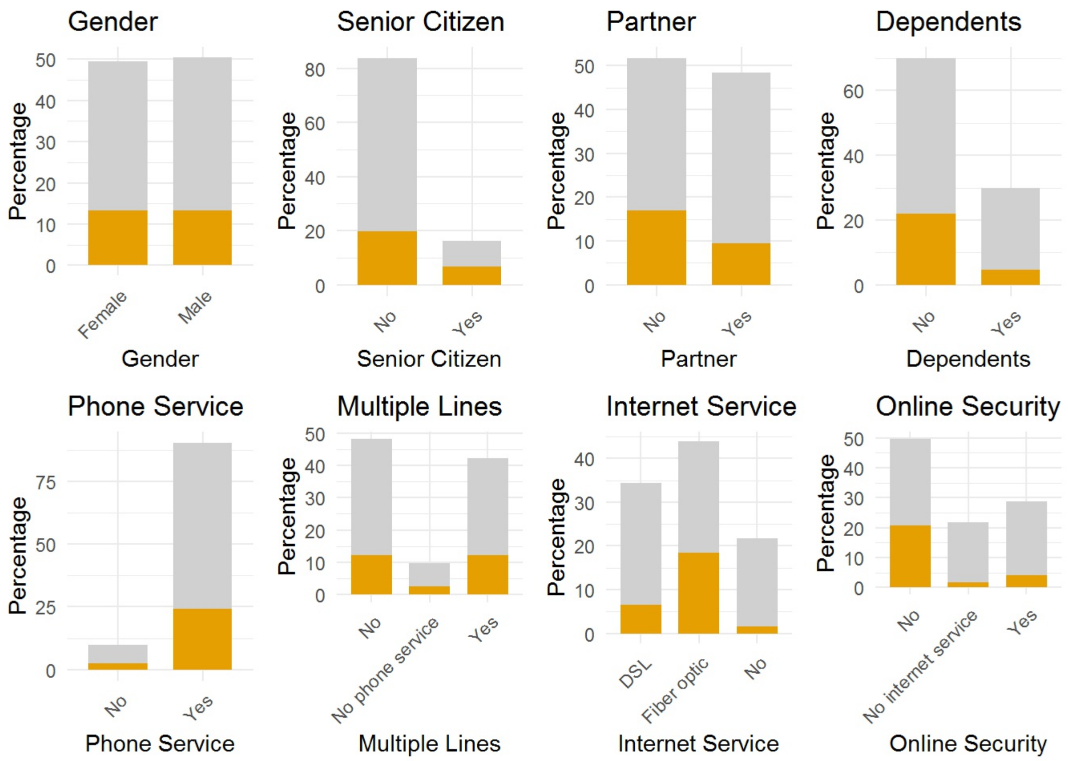

```
# Change value to factor for "Senior Citizen"
all_data$SeniorCitizen <- as.factor(
  mapvalues(all_data$SeniorCitizen,
            from=c("0","1"),
            to=c("No", "Yes"))
)

# Change value to factor for "Churn"
all_data$Churn  <- factor(all_data$Churn)
```

Step 4: Data Visualization

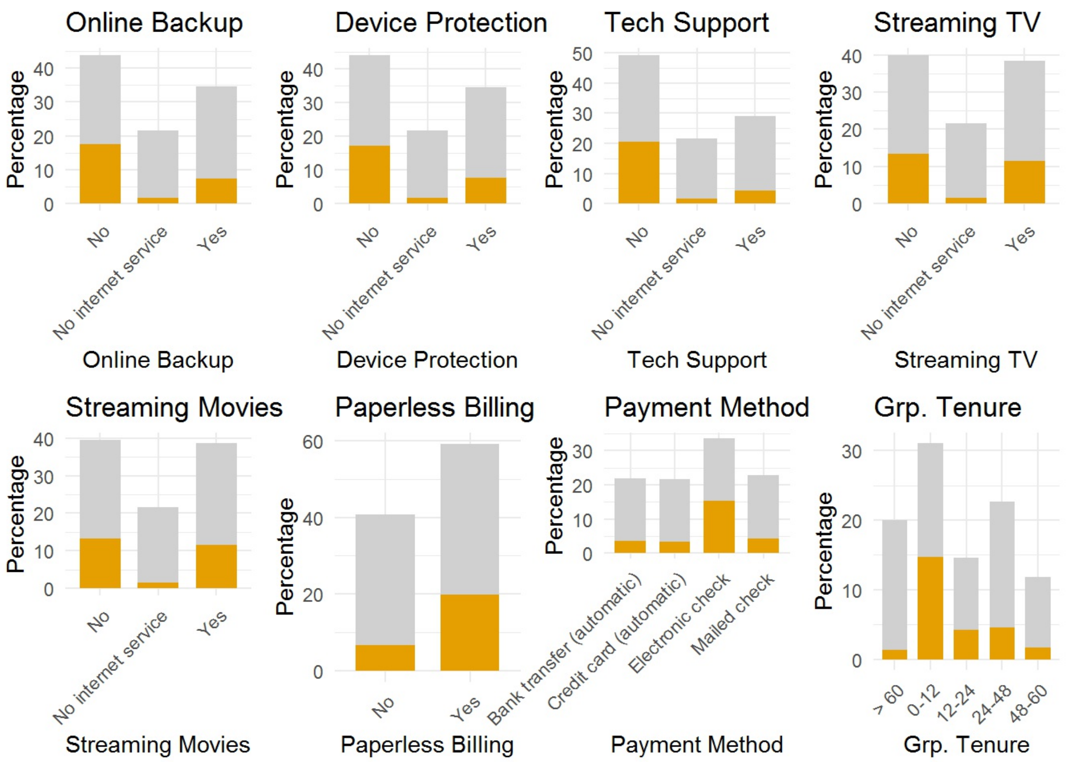
Create the Plotting fucntion to be re-used in plotting different factors that will be retained and use in training/fitting model.

The data visulized include its effect on churning behavior of customers.

```r
# Function for Plotting
F_Plot <- function(dst, column, name) {
  plt <- ggplot(dst, aes(x=column, fill=(Churn))) +
    ggtitle(name) +
    xlab(name) +
    ylab("Percentage")  +
    geom_bar(aes(y = 100*(..count..)/sum(..count..)), width = 0.7) +
    theme_minimal() +
    theme(legend.position="none", axis.text.x = element_text(angle = 45, hjust = 1)) +
    scale_fill_manual(values=c("#d0d0d0", "#E69F00"))
  return(plt)
}


# Plot 1 by gender
p1 <- F_Plot(all_data, all_data$gender, "Gender")
# plot 2 by Senior Citizen
p2 <- F_Plot(all_data, all_data$SeniorCitizen, "Senior Citizen")
# plot 3 by Partner
p3 <- F_Plot(all_data, all_data$Partner, "Partner")
# plot 4 by Dependents
p4 <- F_Plot(all_data, all_data$Dependents, "Dependents")
# plot 5 by Phone Service
p5 <- F_Plot(all_data, all_data$PhoneService, "Phone Service")
# plot 6 by Multiple Lines
p6 <- F_Plot(all_data, all_data$MultipleLines, "Multiple Lines")
# plot 7 by Internet Service
p7 <- F_Plot(all_data, all_data$InternetService, "Internet Service")
# plot 8 by Online Security
p8 <- F_Plot(all_data, all_data$OnlineSecurity, "Online Security")

# draw the plot grid
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol=4)
```



```
# Plot 1 by OnlineBackup
p1 <- F_Plot(all_data, all_data$OnlineBackup, "Online Backup")
# plot 2 by DeviceProtection
p2 <- F_Plot(all_data, all_data$DeviceProtection, "Device Protection")
# plot 3 by TechSupport
p3 <- F_Plot(all_data, all_data$TechSupport, "Tech Support")
# plot 4 by StreamingTV
p4 <- F_Plot(all_data, all_data$StreamingTV, "Streaming TV")
# plot 5 by StreamingMovies
p5 <- F_Plot(all_data, all_data$StreamingMovies, "Streaming Movies")
# plot 6 by PaperlessBilling
p6 <- F_Plot(all_data, all_data$PaperlessBilling, "Paperless Billing")
# plot 7 by PaymentMethod
p7 <- F_Plot(all_data, all_data$PaymentMethod, "Payment Method")
# plot 8 by GrpTenure
p8 <- F_Plot(all_data, all_data$GrpTenure, "Grp. Tenure")

# draw the plot grid
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol=4)
```



Step 5: Factor To be used

After Examining the data, only relevant factors which affect churning behavior are retained

```
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"


# create the test dataset with only the testing columns
factor_to_keep <- c('Churn','SeniorCitizen','Partner','Dependents','GrpTenure','PhoneService',
                'MultipleLines','InternetService','OnlineSecurity','OnlineBackup','DeviceProtection',
                'TechSupport','StreamingTV','StreamingMovies','Contract','PaperlessBilling',
                'PaymentMethod','MonthlyCharges')
```

Step 6: Splitting Data into Training set (75% of data) and Test set (25% of Data)

The training set represent 75% of the data set

The test set represent 25% of the data set

The data is being scaled up (to be used for K-nn)

```
# let's split the dataset into two
DstTrainTest <- all_data[,factor_to_keep]
idxSplit <- createDataPartition(all_data$Churn, p = 0.75, list=FALSE)
DstTrainModel <- DstTrainTest[idxSplit,]
DstTestModel <- DstTrainTest[-idxSplit,]

trainX <- DstTrainModel[,names(DstTrainModel) != "Churn"]
preProcValues <- preProcess(x = trainX,method = c("center", "scale"))
preProcValues
```

```
## Created from 5283 samples and 17 variables
##
## Pre-processing:
##   - centered (1)
##   - ignored (16)
##   - scaled (1)
```

Step 7: Train data with Logistic Regression

Train data with Caret package, the algorithm is Logistic Regression

```
# logistic regression
set.seed(7)
fit.glm <- train(Churn ~ ., data=DstTrainModel, method="glm", metric=metric, trControl=control)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

Step 8: Traing data with Linear ALgoithm LDA

Train data with Caret package, the algorithm is LDA

```
# linear algorithms
set.seed(7)
fit.lda <- train(Churn ~ ., data=DstTrainModel, method="lda", metric=metric, trControl=control)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

Step 9: Train data with CART

Train data with Caret package, the algorithm is CART

```
# CART
set.seed(7)
fit.cart <- train(Churn ~ ., data=DstTrainModel, method="rpart", metric=metric, trControl=control)
```

Step 10: Train data with K-nn

Train data with Caret package, the algorithm is K-nn

```
# kNN
set.seed(7)
fit.knn <- train(Churn ~ ., data=DstTrainModel, method="knn",
                 metric=metric, trControl=control, preProcess = c("center","scale"), tuneLength = 5)
```

Step 11: Train data with SVM

Train data with Caret package, the algorithm is SVM

```
# SVM
set.seed(7)
fit.svm <- train(Churn ~ ., data=DstTrainModel, method="svmRadial", metric=metric, trControl=control)
```

Step 12: Train data with Random Forest

Train data with Caret package, the algorithm is RF

```
# Random Forest
set.seed(7)
fit.rf <- train(Churn ~ ., data=DstTrainModel, method="rf", metric=metric, trControl=control)
```

Step 13: Train data with GBM

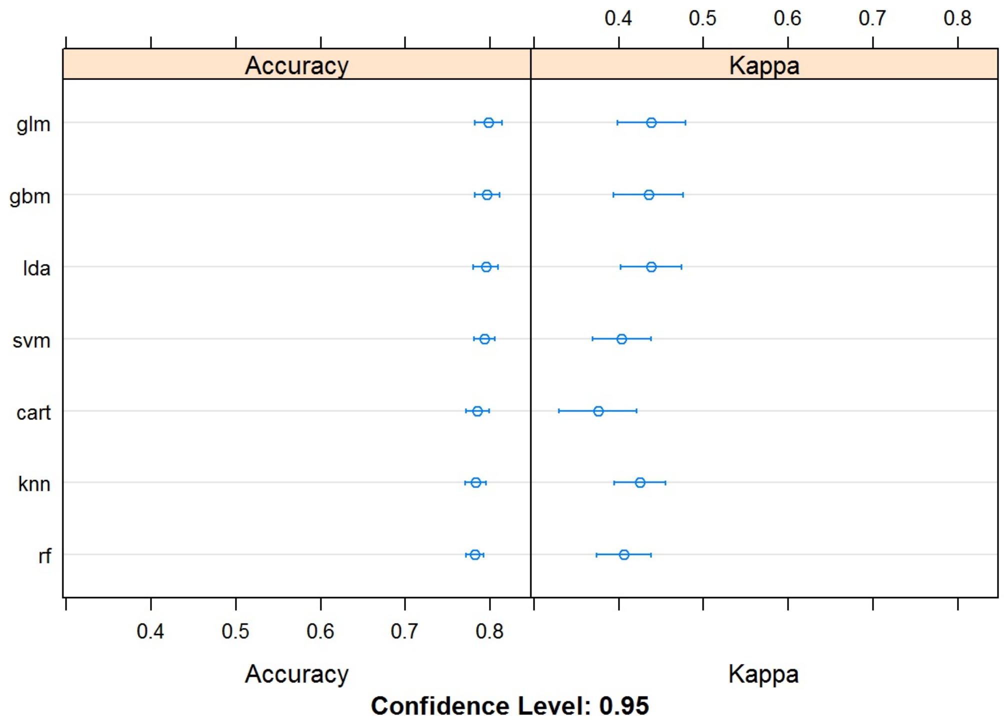
Train data with Caret package, the algorithm is GBM

```
# Gradient Boost Machine (GBM)
set.seed(7)
fit.gbm <- train(Churn ~ ., data=DstTrainModel, method="gbm",
                 metric=metric, trControl=control, verbose=FALSE)
```

Step 14: Looks at Accuracy of each algorithm

Plot accuracy of each algorithm being used to see if it is effective in predicting churning behavior

```
# summarize accuracy of models
results <- resamples(list(
  glm=fit.glm,
  lda=fit.lda,
  cart=fit.cart,
  knn=fit.knn,
  svm=fit.svm,
  rf=fit.rf,
  gbm=fit.gbm
))
#summary(results)
# compare accuracy of models
dotplot(results)
```



Step 15: Looks at the best model that fit our data

Apply the algorithm to our test data and show the accuracy of the algorithm in predicting churning behavior on test data.

```
calculate_accuracy <- function(TestFit, name) {
  # prediction
  DstTestModelClean <- DstTestModel
  DstTestModelClean$Churn <- NA
  predictedval <- predict(TestFit, newdata=DstTestModelClean)

  # summarize results with confusion matrix
  cm <- confusionMatrix(predictedval, DstTestModel$Churn)

  # calculate accuracy of the model
  Accuracy<-round(cm$overall[1],2)
  acc <- as.data.frame(Accuracy)

  roc_obj <- roc(DstTestModel$Churn, as.numeric(predictedval))
  acc$Auc <- auc(roc_obj)

  acc$FitName <- name
  return(acc)
}



accuracy_all <- calculate_accuracy(fit.glm, "glm")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.lda, "lda"))
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.cart, "cart"))
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.knn, "knn"))
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.svm, "svm"))
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.rf, "rf"))
accuracy_all <- rbind(accuracy_all, calculate_accuracy(fit.gbm, "gbm"))
rownames(accuracy_all) <- c()
arrange(accuracy_all,desc(Accuracy))
```

```
##   Accuracy       Auc FitName
## 1     0.81 0.7086826     glm
## 2     0.81 0.7206677     lda
## 3     0.81 0.7157905     gbm
## 4     0.80 0.6865274     svm
## 5     0.79 0.6610103    cart
## 6     0.78 0.6969318     knn
## 7     0.78 0.6802483      rf
```

Conclusion

From above 4 algorithm, the algorithm that perform best is the GLM, LDA,SVM,GBM, which has highest accuracy of around 0.8
