Untitled

victor souob

22/08/2020

HAVARDX FINAL PROJECT DATA SCIENCE VICTOR SOUOB

DEMONSTRATION OF THE RMFT MARKETING MODEL BY STUDYING THE BEHAVIOR OF A BLOOD TRANSFUSION SERVICE CENTER DATA SET IN TAIWAN.

INTRODUCTION

In today business world, is important to know the tendencies of clients and customer's, in order for companies or organization to invest accordantly. The regency, frequency, monetary and time (RFMT) method is an approach used to measure costumer's loyalty and segment customers into various group for future personalization services (Yeh, I-Cheng et al, 2008). The data set used for this project come from the UCI machine learning website (https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center).
The initial data set has 748 observations for 5 variables (we created a 6th variable for the purpose of this study). The 6 variables can be describes as follow: R (Regency - months since last donation) F (Frequency - total number of donation) M (Monetary - total blood donated in c.c.) T (Time - months since first donation) A binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood). A 6th variable was created (period) to calculate the difference between the first month of donation and the last month of donation. The aim here is to create and compare 3 different machine learning models by their accuracy of predicting if a random person will be coming back (or not) to donate blood within a period of time. The 3 models used here was K nearest neighbor's, logistic regression and decision trees. By comparing the accuracy of each model will we decided witch model is best to use

2- METHOD AND ANALYSIS

-DATA CLEAN UP We started by loading all needed libraries, renamed all columns, we checked for all missing values and finally adding a sixth variable inside our data frame (data_period).

```
# Let's start by downloading all needed Libraries
if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")

## Loading required package: tidyverse

## ── Attaching packages
─────────────────────────────────────────────────────────────────────
──────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.1     ✓ purrr   0.3.4
## ✓ tibble  3.0.1     ✓ dplyr   1.0.0
## ✓ tidyr   1.1.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 3.6.2

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## ── Conflicts
─────────────────────────────────────────────────────────────────────────
──────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.2

##
## Attaching package: 'caret'

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

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

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

```r
library(tidyverse)
library(caret)
library(dplyr)
library(rpart)

# Download dataset from UCI respiratory website

data <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/blood-transfusion/transfusion.data",fill = TRUE,header = TRUE,sep = ",")
head(data)
```

```
##   Recency..months. Frequency..times. Monetary..c.c..blood. Time..months.
## 1                2                50                 12500            98
## 2                0                13                  3250            28
## 3                1                16                  4000            35
## 4                2                20                  5000            45
## 5                1                24                  6000            77
## 6                4                 4                  1000             4
##   whether.he.she.donated.blood.in.March.2007
## 1                                          1
## 2                                          1
## 3                                          1
## 4                                          1
## 5                                          0
## 6                                          0
```

```r
# Rename columns

tranfusion_dat_names <- c( "last_M_after_don", "Frequency_of_don", "Amount_blood","M_after_first_don",
                           "Are_theyComingBack")



names(data) <- tranfusion_dat_names

head(data)
```

```
##   last_M_after_don Frequency_of_don Amount_blood M_after_first_don
## 1                2               50        12500                98
## 2                0               13         3250                28
## 3                1               16         4000                35
## 4                2               20         5000                45
## 5                1               24         6000                77
## 6                4                4         1000                 4
##   Are_theyComingBack
## 1                  1
## 2                  1
## 3                  1
```

```
## 4                  1
## 5                  0
## 6                  0
```

```
#-looking for missing values NA
sum(is.na(data))
```

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

```
# adding period to the dataset  by soubstrating first month of donation to
last month

data_period<- data %>% mutate(period = c( M_after_first_don -
last_M_after_don))
head(data_period)
```

```
##   last_M_after_don Frequency_of_don Amount_blood M_after_first_don
## 1                2               50        12500                98
## 2                0               13         3250                28
## 3                1               16         4000                35
## 4                2               20         5000                45
## 5                1               24         6000                77
## 6                4                4         1000                 4
##   Are_theyComingBack period
## 1                  1     96
## 2                  1     28
## 3                  1     34
## 4                  1     43
## 5                  0     76
## 6                  0      0
```

- DATA EXPLORATION

```
str(data_period)
```

```
## 'data.frame':    748 obs. of  6 variables:
##  $ last_M_after_don  : num  2 0 1 2 1 4 2 1 2 5 ...
##  $ Frequency_of_don  : int  50 13 16 20 24 4 7 12 9 46 ...
##  $ Amount_blood      : int  12500 3250 4000 5000 6000 1000 1750 3000 2250
11500 ...
##  $ M_after_first_don : num  98 28 35 45 77 4 14 35 22 98 ...
##  $ Are_theyComingBack: int  1 1 1 1 0 0 1 0 1 1 ...
##  $ period            : num  96 28 34 43 76 0 12 34 20 93 ...
```

```
# we are working with a data frame, has 748 rows and 6 variables
# The variable to predict "Are_theyComingBack"  is listed as numeric , and we
will change it to a factor of 2 class(0,1)
```

The structure of the data_period that we are using for our study is a data frame with 748 obs and 6 variables. Also the variable to predict “Are_theyComingBack” is listed as numeric , and we will change it to a factor of 2 class(0 = no,1 = yes). Now we will look at the summary of the data

```r
# changing the variable to predict as a factor (0 = no, 1 = yes)
data_period$Are_theyComingBack <- as.factor(data_period$Are_theyComingBack)
# data summary
summary(data_period$period)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00   19.00   24.78   41.00   96.00
```

```r
summary (data_period$Frequency_of_don)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   5.515   7.000  50.000
```

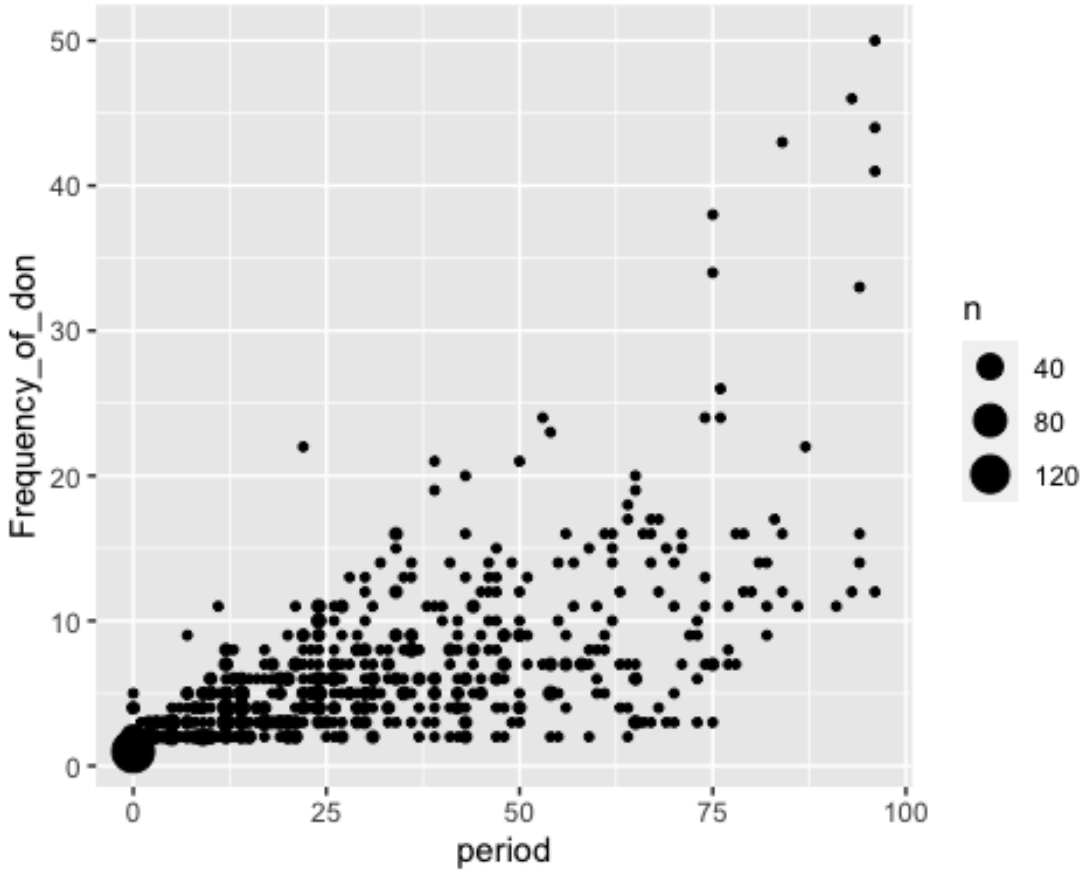
```r
summary(data_period$Amount_blood)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     250     500    1000    1379    1750   12500
```

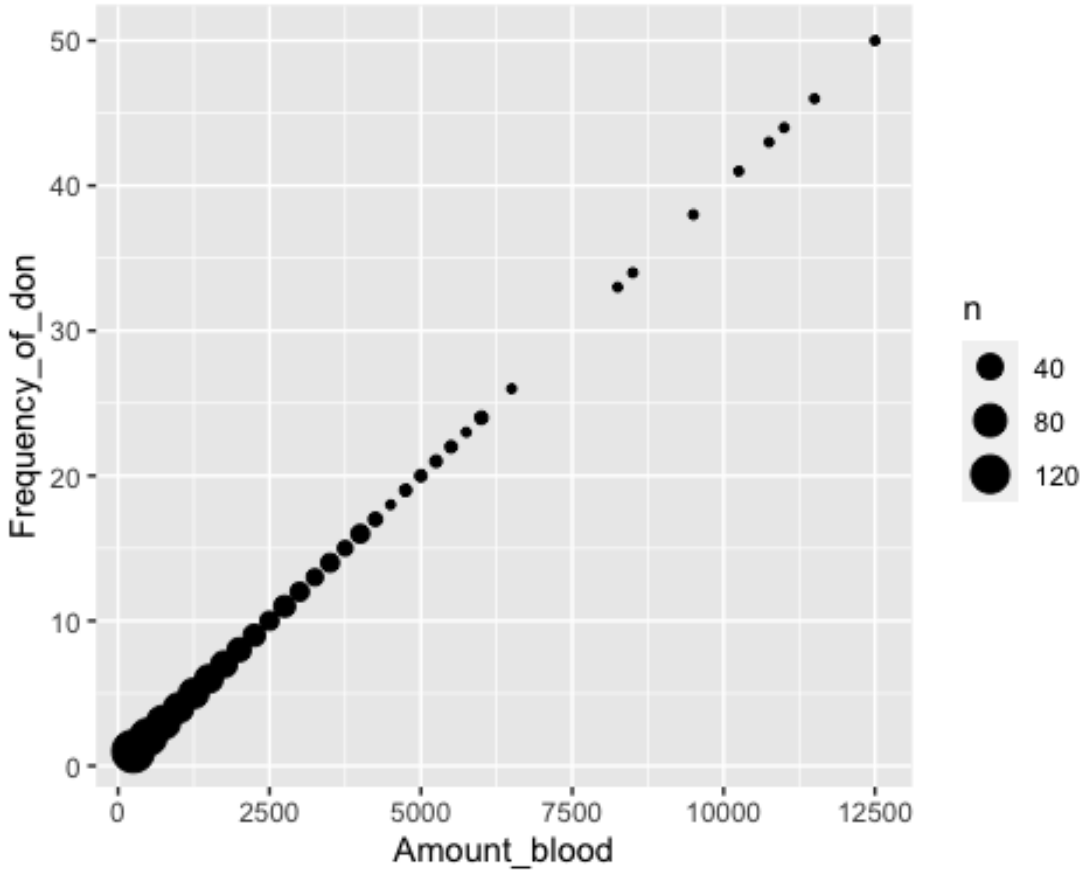
From top to bottom we have period, frequency of donation and amount of donation. We oserved that it tooks only 19 months out 96 months to have half of the data collection. The average total number of donation (frequency_of_don) was 5.515 and the maximum was 50. the maximum amount of blood donated was 12500cc (cc is used here to quantify the amount of blood) , the minimum was 250 cc in less than a month.

DATA VISUALIZATION

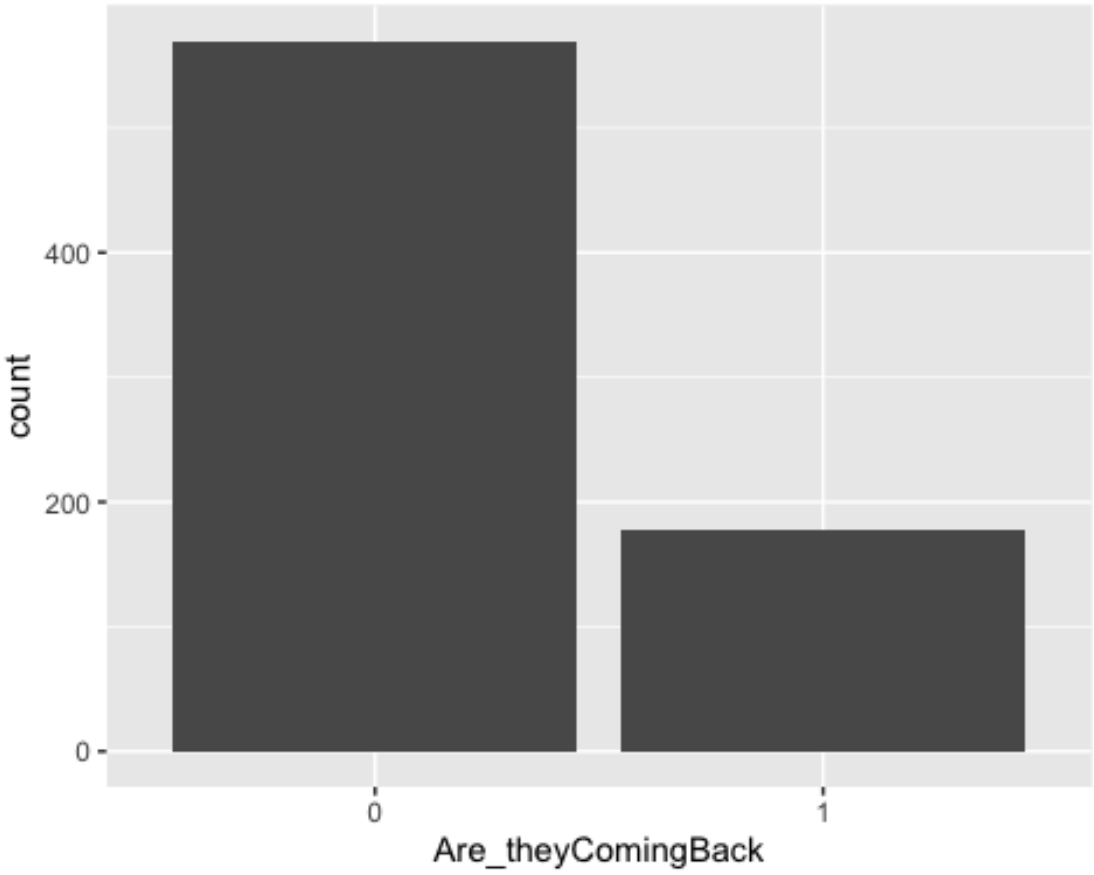
Let's plot the frequeny of donation during the giving period



We observed here that most of the blood donation seems to happen before the 50th month. We also observed that most of the blood donation decreased with time



Here we observe that when the total amount of donation increase , the amount of blood increase as well. At the beginnig the frequency of donaation is hapenning at a higher rate but decreased with time.




```
##
##   0   1
## 570 178
```

In the data set a large proportion of people did not came back to donate . In fact 570 did not came back compared to 178 (during that period) to donate blood. This confirm we have only 2 classes (0,1) in our data set.

-INSIGHT We have a supervised data to predict with known categories. We also have a binary outcome (1 = blood donation, 0 = NO) to classify and predict for accuracy. Here we are trying to reproduce a RTMC model (Regency as months since last donation, Frequency as total number of donation, Monetary as total blood donated, Time as months since last donation) so we will purposely predict the outcome without identifying the best variable (forward selection, correlation etc…) that can generated the best accuracy . KNN,logistic regression and decision trees seems to be good models to explore for accuracy.

-KNN model We will used the KNN method to train the first model, test it and compute for the accuracy of the model. The theorie here is to separated the supervised data (train and test set). Train the known supervised data (using crosss validation) and compare each point of the the test set , to the trained data. By looking at the mojority of classes (or the class with the higher average) surounding the variable to predict, We can then decide the ouput

of a specific class. Prior to that , we will first find the best K (number of neighbor) that optimize the modeol.

-Logistic regresion Logistic regression can also be use to classify sample. By knowing the classes we are trying to predict , the logistic regression fits an S shape (goes from 0 to 1) logistic function through the data (train_sample). By looking at the probability (if X is greater or smaller than 50 % on the S shape curve), we can then classify an unknown variable.

-Decision Trees Are used in prediction problem where the outcome is a classification of data. The decision Tree can de define Is a flow chart of Yes or No question , and the chalenge is to define an algorithm that used data to create tress with prediction at the end.

CREATE DATA PARTITION

We divided the data into a 2 sets train_sample(80 %) and a test_sample(20%)

```
set.seed(1)
data$Are_theyComingBack <- as.factor(data$Are_theyComingBack)
test_index <- createDataPartition(y = data$Are_theyComingBack, times = 1, p =
0.2, list = FALSE)
train_sample <- data_period [-test_index,]
test_sample <- data_period [test_index,]

# lets check if we have the same probabiblity after partition

prop.table(table(train_sample$Are_theyComingBack))

##
##         0         1
## 0.7625418 0.2374582

prop.table(table(test_sample$Are_theyComingBack))

##
##    0    1
## 0.76 0.24

# The probability for train and test set are almost the same.
```

The probability for train and test set are almost the same after partition.

RESULTS

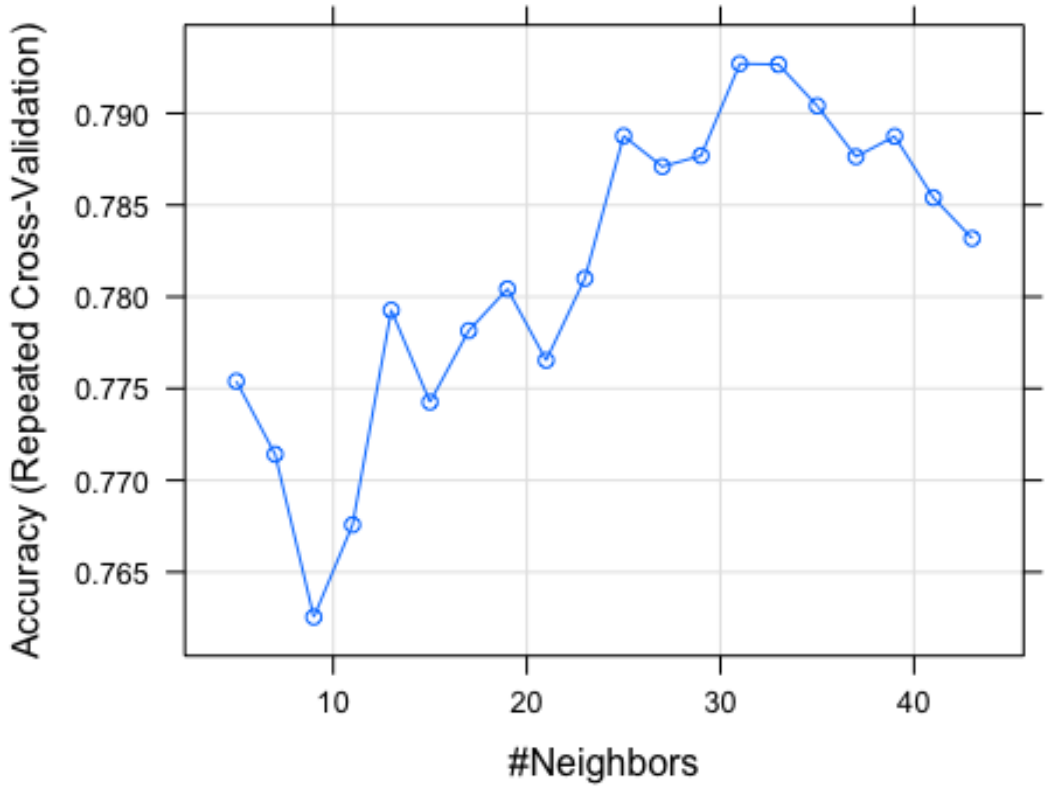
KNN model

```
# lets find the best k for the smallest RMSE

set.seed(1)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knnFit <- train(Are_theyComingBack ~ ., data = train_sample, method = "knn",
```

```
trControl = ctrl, preProcess = c("center","scale"),tuneLength = 20)
knnFit
```

```
## k-Nearest Neighbors
##
## 598 samples
##   5 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (5), scaled (5)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 539, 538, 539, 538, 538, 539, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7754001  0.2924604
##    7  0.7714180  0.2623511
##    9  0.7625479  0.2249935
##   11  0.7675762  0.2453621
##   13  0.7792717  0.2850017
##   15  0.7742626  0.2676763
##   17  0.7781515  0.2676274
##   19  0.7804296  0.2787336
##   21  0.7765404  0.2595661
##   23  0.7810031  0.2682078
##   25  0.7887629  0.2979754
##   27  0.7870963  0.2819296
##   29  0.7876889  0.2832523
##   31  0.7926989  0.3026223
##   33  0.7926615  0.2901348
##   35  0.7904023  0.2762050
##   37  0.7876333  0.2606393
##   39  0.7887447  0.2635870
##   41  0.7854017  0.2463837
##   43  0.7831788  0.2338956
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 31.
```



The final value used for the model was k = 31.

```
# lets predict with the test set
y_hat_knn <- predict(knnFit,test_sample, type = "raw")


# lets find the accuracy

confusionMatrix(y_hat_knn,test_sample$Are_theyComingBack)$overall[["Accuracy"
]]
```

```
## [1] 0.7866667
```

```
## we have an accuracy of 0.78 with KNN
```

KNN model has an accuracy of 0.78

LOGISTIC REGRESSION

```
##### LOGISTIC REGRESSION
# lets fitt the model

glmFit <- train(Are_theyComingBack ~ ., data = train_sample, method = "glm",
family="binomial")
```

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

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

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

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

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

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

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

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

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

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

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

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
```

```
== :
## prediction from a rank-deficient fit may be misleading

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

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

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

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

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

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

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

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

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

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

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
```

```
== :
## prediction from a rank-deficient fit may be misleading

glmFit

## Generalized Linear Model
##
## 598 samples
##   5 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 598, 598, 598, 598, 598, 598, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7768077  0.1310143

# lets predict it using test set and  find the accuracy
y_hat_glmn<- predict(glmFit, test_sample, type = "raw")

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

y_hat_glmn

##   [1] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
## [149] 0 0
## Levels: 0 1

confusionMatrix(data = y_hat_glmn, reference =
test_sample$Are_theyComingBack)$overall[1]

##  Accuracy
## 0.7666667

### final accuracy of our logisic regresion model is 0.76
```

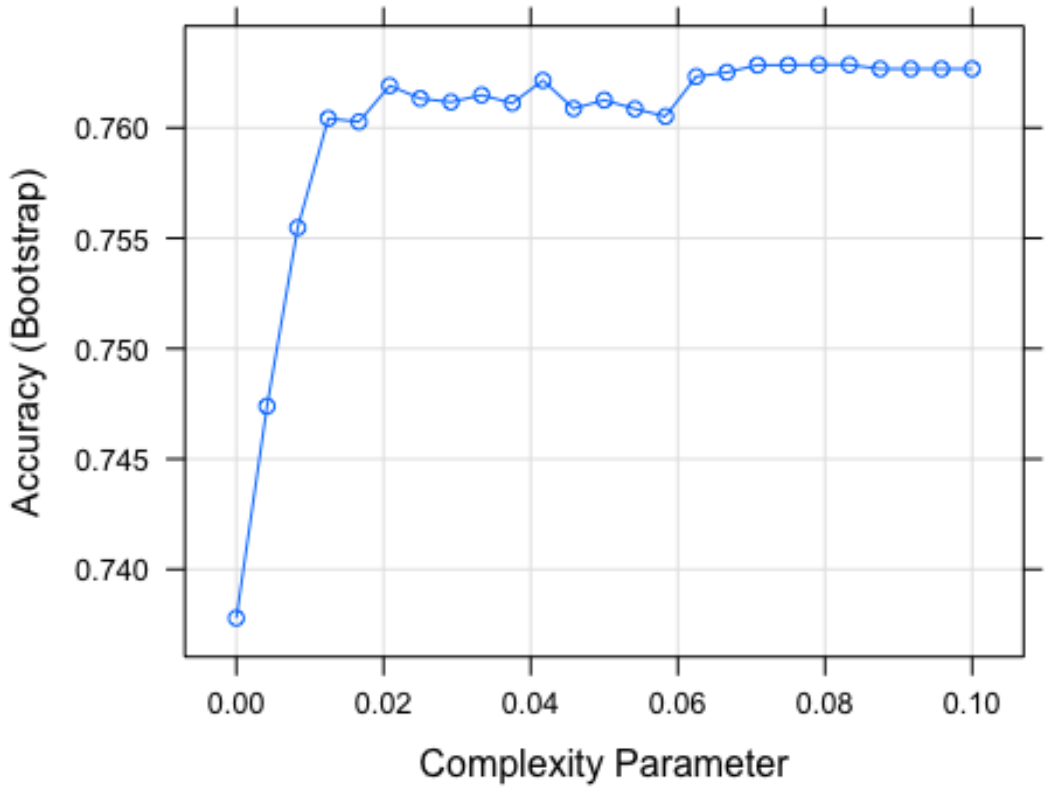
The accuracy of the model with logistic regression is 0.76

DECISSION TREES

```
#### DECISION TREES
```

```
# lets train the rpart and find the best cp
train_rpart <- train(Are_theyComingBack~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),

                                           data = train_sample)
plot(train_rpart)
```



the best cp is around 0.03

```
## lets find the accuracy of the model
confusionMatrix(predict(train_rpart, test_sample),
test_sample$Are_theyComingBack)$overall["Accuracy"]

## Accuracy
##     0.76

### accuracy of 76 percent
```

With decision Tree we have an accuacy of 0.76

```
# lets look at some feature
train_rpart$finalModel
```

```
## n= 598
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 598 142 0 (0.7625418 0.2374582) *
```

This is a summary description of our data (See rmd file for pictures) with decision tree. Here we observe that 9.5 months after the last donation, donors with an averall frequency lower than 4.5 did came back to donate blood. If the period of donation was more than 46.5 months donors with a frequency of donation higher than 4.5 will most likely donate. Also for a period of 46.5 months donors with a frequency of donation lower than 25 will not , most likely comeback.

Findings

Knn with an accuracy of 0.78 is by far the highest of the 3 model we studied here, followed by Logistic regression (0.76) and decision trees (0.76). Althoug The decision Tree give us a very detail summary of data compared to KNN and Logistic regression. Overall the decison tree give us more information but with a lower accuracy compare to KNN.

CONCLUSION

In summary , we have tested 3 different model. KNN is the most accurate of the 3 models but it is less informative compare to Decison Tree. One of the limitation was the sample size . In futur works, we could replicate this study with a larger sample size and predict the accuracy.