Computational Modeling and Cluster Analysis

Name: Clariza Look
Subject: Computational Data Modeling and Clustering Techniques

0. Task

This project is to understand the entire process of creating machine learning models and evaluation. The task was to create simple statistical learning models from scratch, evaluating the models and testing the chosen model. Models used in this project include Null Model, Single Variable Model, Logistic Regression Model and Support Vector machine. These models are evaluated through techniques such as confusing matrix, and Receiver Operating Characteristic (ROC) curve. All are done in RStudio.

Other tasks involved in this project include splitting of data to 3 sets: Train, Calibration and Test; and feature engineering, one hot encoding and cluster analysis.

1. Description of Data

This data was made available by GovHack Employment Fund Dataset (https://data.gov.au/data/dataset/6fd24a3b-6028-473e-985a-b863bac25521/resource/60fe1efd-8032-41d2-995d-68f178118dc9/download/govhack-employment-fund-dataset_20150701-to-20170630.csv).

Business Understanding

The data includes jobactive provider information, job seeker characteristics at the time of the EF expenditure transaction, and transaction details and expenditure. The dataset was extracted on 5 August 2018, however is based on EF expenditure transactions that occurred between 1 July 2015 and 30 June 2017.

Below are some of the business terms that will be used in the data analysis:

Jobactive

jobactive is a Government program to aid Australians to get into work. It acts a bridge between job seekers (participants) and employers through a network of jobactive providers around the country. jobactive help job seekers acquire knowledge, skills, and other resources to meet the needs for employment.

Employment Fund

Employment Fund (EF) refers to available funds provided to jobactive agency as credits, which is used as a claim reimbursement for goods and services that helped to support job seekers to gain the tools, skills and experience needed to get and keep a job. Goods and services may include training, food, clothing, transport, tools and other resources.

The original dataset has 1,231,535 rows and 20 variables as refered in Section 1: Column Definition of Appendix for full details. But only selected columns are needed for creating our classification models later.

After going through EDA process in the previous task, the “cleaned data” has been exported and stored as csv file. So, read.csv function is used to read csv file and store the data into R.

2. Load cleaned data

```
ef_data=read.csv(file="employfund_cleaned.csv",header=TRUE,sep=",")

#Load Libraries
library(dplyr)
library(kableExtra)
library(ggplot2)
library(magrittr)
library(rapportools)
library('ROCR')
library(ROCit)
library(e1071)
library(caret)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.0.3
```

```r
library(mltools)
library(data.table)
library(ape)
```

```
## Warning: package 'ape' was built under R version 4.0.3
```

```r
library(fpc)
library(plyr)
library(pROC)
```

```r
#View dataset
dim(ef_data)
```

```
## [1] 1231535      29
```

```r
kable(ef_data[1:5,1:ncol(ef_data)]) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), full_width = F, font_size = 9) %>%
  scroll_box(width = "910px", height = "200px")
```

X	ï..DATA_AS_AT	TRANSACTION_ID	EXPENDITURE	STATE	JSKR_LINKAGE_KEY	STREAM_PLACEMENT_DESC	EF_CATEGORY_DESCRIPTION	DERIVED_DATE_OF_SERVICE	ACTIVITY_TY
1	2018-08-05	303887J	150	QLD	1	Stream A	Professional Services	2016-01-04	NULL
2	2018-08-05	283232J	300	QLD	1	Stream A	Professional Services	2015-12-21	NULL
3	2018-08-05	412011J	150	QLD	1	Stream A	Professional Services	2016-01-11	NULL
4	2018-08-05	376042J	150	QLD	1	Stream A	Professional Services	2016-02-08	NULL
5	2018-08-05	415778J	225	VIC	2	Stream C	Professional Services	2016-02-18	NULL

3. Choosing Only Relevant Columns

The variables that we are going to use for creating the classification models are TRANSACTION_ID, EXPENDITURE, STATE, STREAM_PLACEMENT_DESC, GENDER, INDIGENOUS,HOMELESS, PWD, CALD, REFUGEE, EX_OFFENDER, AGE_GROUP, UE_GROUP, EDU_LVL_SIM, EF_CAT_SIM, year_month.

```r
#Choosing only selected columns
employfund_datamodel <- dplyr::select (ef_data,
                                       TRANSACTION_ID,
                                       EXPENDITURE,
                                       STATE,
                                       STREAM_PLACEMENT_DESC,
                                       GENDER,
                                       INDIGENOUS,
                                       HOMELESS,
                                       PWD,
                                       CALD, REFUGEE,
                                       EX_OFFENDER,
                                       AGE_GROUP,
                                       UE_GROUP,
                                       EDU_LVL_SIM,
                                       EF_CAT_SIM,
                                       year_month)


#View dataset
dim(employfund_datamodel)
```

```
## [1] 1231535      16
```

```
#Format table using Kable
kable(employfund_datamodel[1:5,1:ncol(employfund_datamodel)]) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"), full_width = F, font_size = 9) %>%
  scroll_box(width = "910px", height = "200px")
```

TRANSACTION_ID	EXPENDITURE	STATE	STREAM_PLACEMENT_DESC	GENDER	INDIGENOUS	HOMELESS	PWD	CALD	REFUGEE	EX_OFFENDER	AGE_GROUP	UE_GROUP	EDU_LV
303887J	150	QLD	Stream A	Female	NULL	NULL	NULL	1	NULL	NULL	50 to 54 years	Under 12 Months	BacDeg
283232J	300	QLD	Stream A	Female	NULL	NULL	NULL	1	NULL	NULL	50 to 54 years	Under 12 Months	BacDeg
412011J	150	QLD	Stream A	Female	NULL	NULL	NULL	1	NULL	NULL	50 to 54 years	Under 12 Months	BacDeg
376042J	150	QLD	Stream A	Female	NULL	NULL	NULL	1	NULL	NULL	50 to 54 years	Under 12 Months	BacDeg
415778J	225	VIC	Stream C	Female	NULL	1	NULL	NULL	NULL	NULL	25 to 29 years	24 to 59	Diploma

- Before proceeding to do the modeling, we will need to clean and transform the dataset and convert the data types into proper format for modeling. Then, we will choose the target variable then split entire dataset into 3 sets: train, test data and validation data. Train data is used to train the classification model while test data is used to validate the model. The proportions will be divided into 50% train, 25% test and 25% calibration.

4. Data Wrangling

Part 1 - Binary Classification

Choose the response (target) variable

The chosen response variable is the "STREAM_PLACEMENT_DESC" as described below:

- STREAM_PLACEMENT_DESC: The placement type (Stream) of the job seeker at the time of the EF transaction.
  - **Stream A** - Job seekers are the most job ready. They will receive services to help them understand what employers want and how to navigate the local labour market, build a resume, look for jobs and learn how to access self-help facilities
  - **Stream B** - Job seekers need their jobactive provider to play a greater role to help them become job ready and will be referred for case management support.
  - **Stream C** - Job seekers have a combination of work capacity and personal issues that need to be addressed and will get case management support so that they can take up and keep a job.
  - **Stream Volunteer** - Volunteer job seekers can access jobactive services for up to six months.

However, from the above, we have 4 categories (as a response/target variable) of STREAM_PLACEMENT: (Stream A, B, C and Volunteer) in the dataset. Since we are doing a binary classification model, we would classify between "Stream C" and other streams that are "Non-Stream C".

*[**Important Note**: The reason of choosing "Stream C" is because we can use it a an indicator to ensure that this particular stream's threshold can still be manageable or do the job active providers needs to add more support services].*

We will focus on this as a target variable to help classify if the employment fund transaction belongs to "Stream C" and not. Having this, we will be creating an additional column categorizing each row between "Stream C" and "Non-Stream C". The labels we assigned to this variable are "STREAM_relabel_binary" with values "yes" if value is Stream C and "No" if value is non-stream C. Our hope is to create a classification model that, with a variety of explanatory variables, will be able to assess if a certain employment fund transaction belongs to Stream C or not.

Create a categorical "target" variable. Is transaction Stream C or other type of stream?

```
#Create a new column to categorize STREAM C and non STREAM C with values Yes & No
employfund_datamodel <- employfund_datamodel %>% mutate(STREAM_relabel_binary = case_when(
                          STREAM_PLACEMENT_DESC=="Stream B" ~ "No",
                          STREAM_PLACEMENT_DESC=="Stream A" ~ "No",
                          STREAM_PLACEMENT_DESC=="Stream Volunteer" ~ "No",
                          STREAM_PLACEMENT_DESC=="Stream C" ~ "Yes"))

#Check NA in STREAM_relabel_binary
summary(employfund_datamodel$STREAM_relabel_binary)
```

```
##    Length     Class      Mode 
##   1231535 character character
```

```
#Exclude "NA" in employfund_datamodel$STREAM_relabel_binary. The cause of NA is because that STREAM_PLACEMENT_DESC
#is NULL and this is a very small proportion of the entire dataset
employfund_datamodel <- employfund_datamodel %>% filter(!is.na(employfund_datamodel$STREAM_relabel_binary))
```

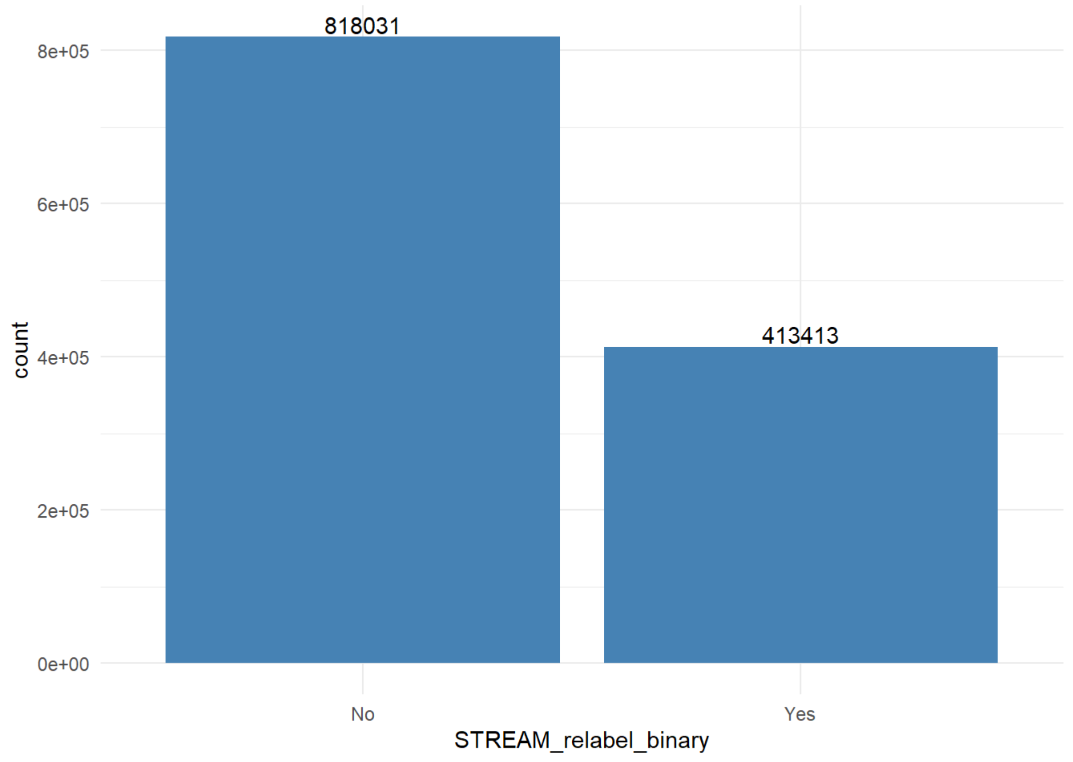
Convert variables to their desired data type format

```
#Convert to categorical data for binary classification
temp_categorical <- employfund_datamodel %>% mutate_at(vars(
                                  TRANSACTION_ID,
                                  STATE,
                                  STREAM_PLACEMENT_DESC,
                                  INDIGENOUS,
                                  HOMELESS,
                                  PWD, CALD,
                                  EF_CAT_SIM,
                                  EDU_LVL_SIM,
                                  REFUGEE,
                                  EX_OFFENDER,
                                  GENDER,
                                  AGE_GROUP,
                                  UE_GROUP,
                                  EDU_LVL_SIM,
                                  EF_CAT_SIM,
                                  STREAM_relabel_binary),list(factor))

employfund_datamodel <- temp_categorical

#Check data types
str(employfund_datamodel)
```

```
## 'data.frame':    1231444 obs. of  17 variables:
##  $ TRANSACTION_ID       : Factor w/ 1231444 levels "100000J","100001J",..: 805400 792748 863929 842124 866598 916309 91
8551 893510 1216022 1214189 ...
##  $ EXPENDITURE          : num  150 300 150 150 225 300 225 300 150 150 ...
##  $ STATE                : Factor w/ 8 levels "ACT","NSW","NT ",..: 4 4 4 4 7 7 7 7 7 7 ...
##  $ STREAM_PLACEMENT_DESC: Factor w/ 4 levels "Stream A","Stream B",..: 1 1 1 1 3 3 3 3 3 3 ...
##  $ GENDER               : Factor w/ 3 levels "Female","Male",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ INDIGENOUS           : Factor w/ 2 levels "1","NULL": 2 2 2 2 2 2 2 2 2 2 ...
##  $ HOMELESS             : Factor w/ 2 levels "1","NULL": 2 2 2 2 1 1 1 1 1 1 ...
##  $ PWD                  : Factor w/ 2 levels "1","NULL": 2 2 2 2 2 2 2 2 2 2 ...
##  $ CALD                 : Factor w/ 2 levels "1","NULL": 1 1 1 1 2 2 2 2 2 2 ...
##  $ REFUGEE              : Factor w/ 2 levels "1","NULL": 2 2 2 2 2 2 2 2 2 2 ...
##  $ EX_OFFENDER          : Factor w/ 2 levels "1","NULL": 2 2 2 2 2 2 2 2 2 2 ...
##  $ AGE_GROUP            : Factor w/ 9 levels "22 to 24 years",..: 5 5 5 5 2 2 2 2 2 2 ...
##  $ UE_GROUP             : Factor w/ 5 levels "12 to 23 Months",..: 5 5 5 5 2 2 2 2 2 2 ...
##  $ EDU_LVL_SIM          : Factor w/ 14 levels "<Y10","BacDeg",..: 2 2 2 2 3 3 3 3 3 3 ...
##  $ EF_CAT_SIM           : Factor w/ 20 levels "AccredTraining",..: 12 12 12 12 12 12 12 12 12 12 ...
##  $ year_month           : chr  "Jan 2016" "Dec 2015" "Jan 2016" "Feb 2016" ...
##  $ STREAM_relabel_binary: Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 2 2 2 ...
```

Quick Viz for Stream C vs. Non Stream C

```
#Bar plot for Stream C vs. Non Stream C distribution
ggplot(employfund_datamodel, aes(x=STREAM_relabel_binary)) +
  geom_bar(fill="steelblue")+
  geom_text(stat = 'count',aes(label =..count..,
                              vjust = -0.2)) +
  theme_minimal()
```



Merging Sub-Categories Of The Categorical Data

The columns such as EF_CAT_SIM, EDU_LVL_SIM and AGE_GROUP have many sub categories which will require to merge to reduce the number of categorical values.

- Merge EF_CAT_SIM sub-categories:
  - “AccredTraining”, “EmployerTraining”, “PreEmployPrep” & “PostPlacementAssist” to “Training”
  - “ProfService”, “MedicalExps” to “Medical/Mental Health”
  - “WorkItems”, “WorkLicense”, “PathInternship”, “DrivingLessons” to “Work Requirement/Purchase”
  - “Clothing”, “Transport”, “Vouchers” to “Essentials”
  - “RelocateAssist”,“RentAccomm”, “StreamCAssist” to “Accommodation/Financial Aid”
  - “WorkTrials”, “Non-WfDCosts”,“Non-GovProg” to “Other incentive”)
- Merge EDU_LVL_SIM sub-categories:
  - “Y12/13”, “Y10/11” to “Secondary”
  - “NTVocEdu”,“Trade/TAFE” to “Vocational”
  - “Diploma”, “BacDeg”, “GradDip/Cert”, “PostGrad” to “HigherEdu”
  - “<Y10” to “Primary”
  - “UKN”, “NotVol” to “Others”
- Merge AGE_GROUP sub-categories:
  - “Under 22 years”, “22 to 24 years” to “Below 25”
  - “55 to 59 years”, “60+ years” to “55 and above”)

```
#Merging EF_CAT_SIM sub-categories
employfund_datamodel <- employfund_datamodel %>% mutate(EF_CAT_MERGED = case_when(
                          EF_CAT_SIM =="AccredTraining" ~ "Training",
                          EF_CAT_SIM =="EmployerTraining" ~ "Training",
                          EF_CAT_SIM =="PreEmployPrep" ~ "Training",
                          EF_CAT_SIM =="PostPlacementAssist" ~ "Training",
                          EF_CAT_SIM =="ProfService" ~ "Medical/Mental Health",
                          EF_CAT_SIM =="MedicalExps" ~ "Medical/Mental Health",
                          EF_CAT_SIM =="WorkItems" ~ "Work Requirement/Purchase",
                          EF_CAT_SIM =="WorkLicense" ~ "Work Requirement/Purchase",
                          EF_CAT_SIM =="PathInternship" ~ "Work Requirement/Purchase",
                          EF_CAT_SIM =="DrivingLessons" ~ "Work Requirement/Purchase",
                          EF_CAT_SIM =="Clothing" ~ "Essentials",
                          EF_CAT_SIM =="Transport" ~ "Essentials",
                          EF_CAT_SIM =="Vouchers" ~ "Essentials",
                          EF_CAT_SIM =="RelocateAssist" ~ "Accommodation/Financial Aid",
                          EF_CAT_SIM =="RentAccomm" ~ "Accommodation/Financial Aid",
                          EF_CAT_SIM =="StreamCAssist" ~ "Accommodation/Financial Aid",
                          EF_CAT_SIM =="WorkTrials" ~ "Other incentive",
                          EF_CAT_SIM =="Non-WfDCosts" ~ "Other incentive",
                          EF_CAT_SIM =="Non-GovProg" ~ "Other incentive"))

#Merging EDU_LVL_SIM sub-categories
employfund_datamodel <- employfund_datamodel %>% mutate(EDU_LVL_MERGED = case_when(
                          EDU_LVL_SIM =="Y12/13" ~ "Secondary",
                          EDU_LVL_SIM =="Y10/11" ~ "Secondary",
                          EDU_LVL_SIM =="NTVocEdu" ~ "Vocational",
                          EDU_LVL_SIM =="Trade/TAFE" ~ "Vocational",
                          EDU_LVL_SIM =="PostGrad" ~ "HigherEdu",
                          EDU_LVL_SIM =="GradDip/Cert" ~ "HigherEdu",
                          EDU_LVL_SIM =="BacDeg" ~ "HigherEdu",
                          EDU_LVL_SIM =="Diploma" ~ "HigherEdu",
                          EDU_LVL_SIM =="<Y10" ~ "Primary",
                          EDU_LVL_SIM =="NotVol" ~ "Others",
                          EDU_LVL_SIM =="UKN" ~ "Others"))



#Merging AGE_GROUP sub-categories
employfund_datamodel <- employfund_datamodel %>% mutate(AGE_GROUP_MERGED = case_when(
                          AGE_GROUP =="Under 22 years" ~ "Below 25",
                          AGE_GROUP =="22 to 24 years" ~ "Below 25",
                          AGE_GROUP =="55 to 59 years" ~ "55 and above",
                          AGE_GROUP =="60+ years" ~ "55 and above",
                          AGE_GROUP =="25 to 29 years" ~ "25 to 29 years",
                          AGE_GROUP =="30 to 39 years" ~ "30 to 39 years",
                          AGE_GROUP =="40 to 49 years" ~ "40 to 49 years",
                          AGE_GROUP =="50 to 54 years " ~ "50 to 54 years "))

#Factor the newly added columns
employfund_datamodel <- employfund_datamodel %>% mutate_at(vars(
                          EF_CAT_MERGED,
                          EDU_LVL_MERGED,
                          AGE_GROUP_MERGED),
                          list(factor))

#dim(employfund_datamodel)
```

5. Splitting train, Calibration and Test dataset

Next, the data is split into 3 sets: Train data is used to train the model while calibrate data is used to validate the model for evaluation, then test data is to give the model another validation. The proportions will be divided into 50% train, 25% calibration and 25% test.

Before splitting, due to large dataset (about 1.2M rows) and limited computational power, a dataset close to 24,000 observations has been selected using random number uniformly generated via runif function.

```
set.seed(4848)

#Select random numbers for testtrain dataset and put in a new column of employfund_datamodel
employfund_datamodel <- employfund_datamodel %>% mutate(testtrain = runif(nrow(employfund_datamodel),0,1))

#Take the 24,000 samples/observations
ef_data_s <- employfund_datamodel[employfund_datamodel$testtrain<0.02,]

#Split train and test datasets from the chosen observations
ef_traindata <- ef_data_s[ef_data_s$testtrain < 0.005,]
ef_caldata <- ef_data_s[ef_data_s$testtrain >= 0.005&ef_data_s$testtrain < 0.01, ]
ef_testdata <- ef_data_s[ef_data_s$testtrain >= 0.01,]

#To check that its approximately 24,000 observations
nrow(ef_traindata)
```

```
## [1] 6084
```

```
nrow(ef_caldata)
```

```
## [1] 6145
```

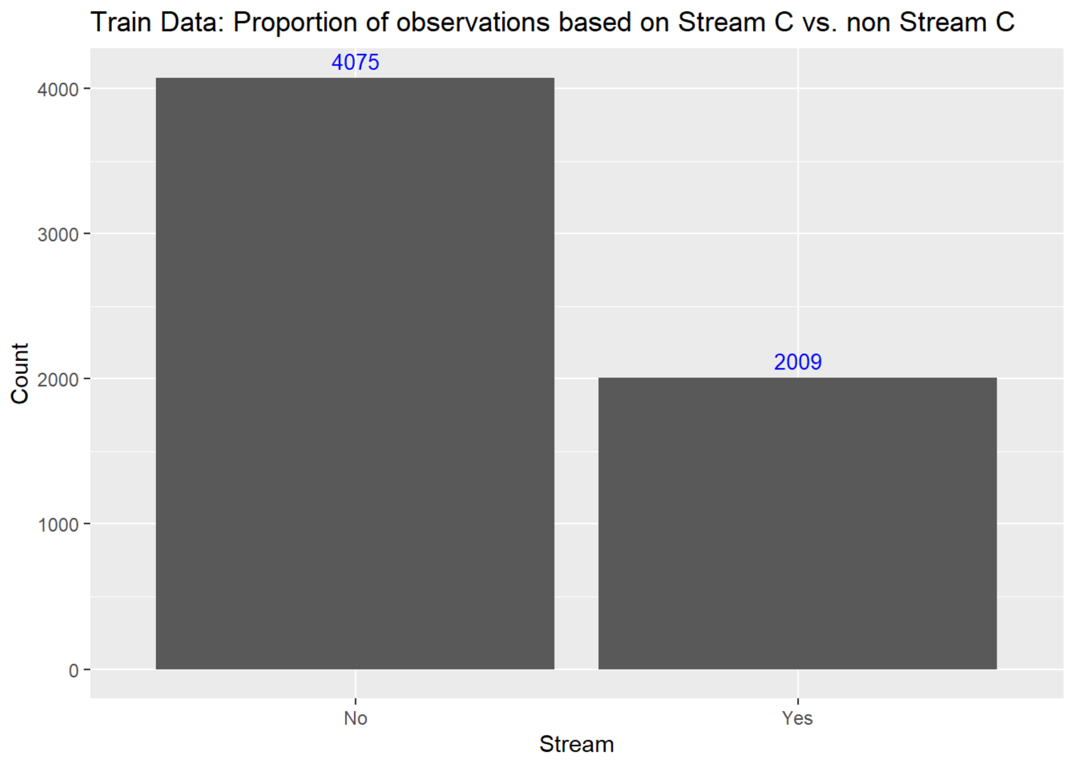
```
nrow(ef_testdata)
```

```
## [1] 12414
```

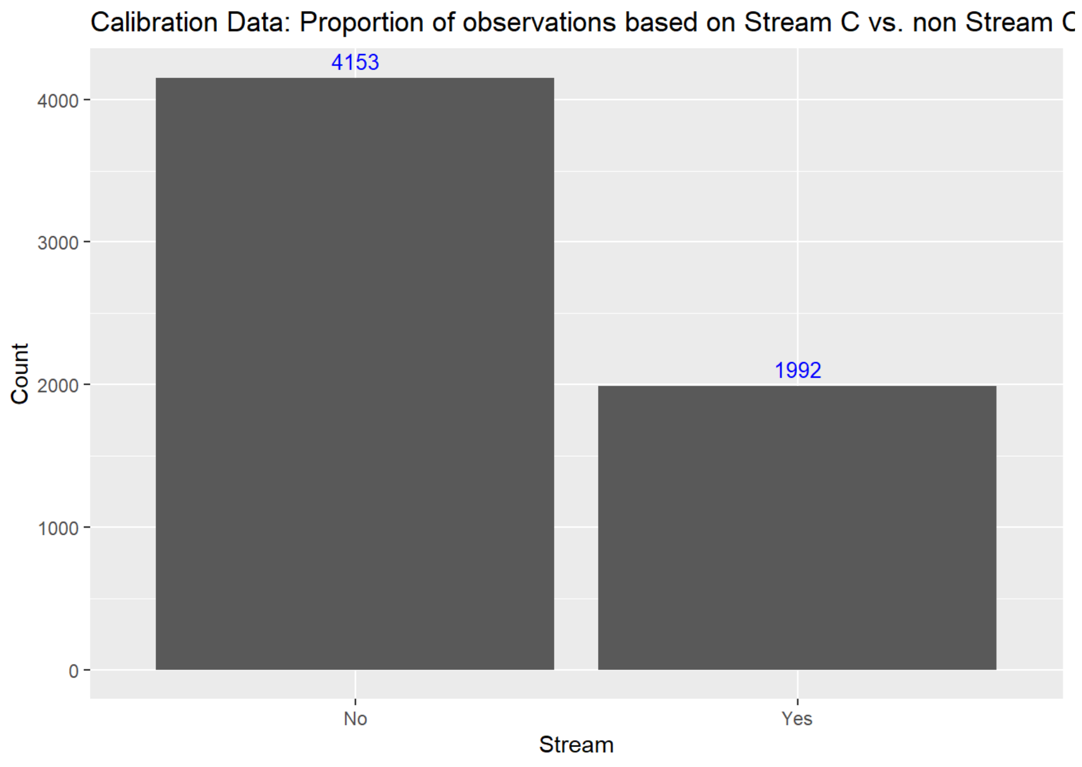
```
#Calculating their proportions
prop_traindata <- round(nrow(ef_traindata)/nrow(ef_data_s)*100,3)

prop_caldata <- round(nrow(ef_caldata)/nrow(ef_data_s)*100,3)

prop_testdata <- round(nrow(ef_testdata)/nrow(ef_data_s)*100,3)
```

To ensure the data is split evenly, a table based on Stream is tabulated for test and train data.

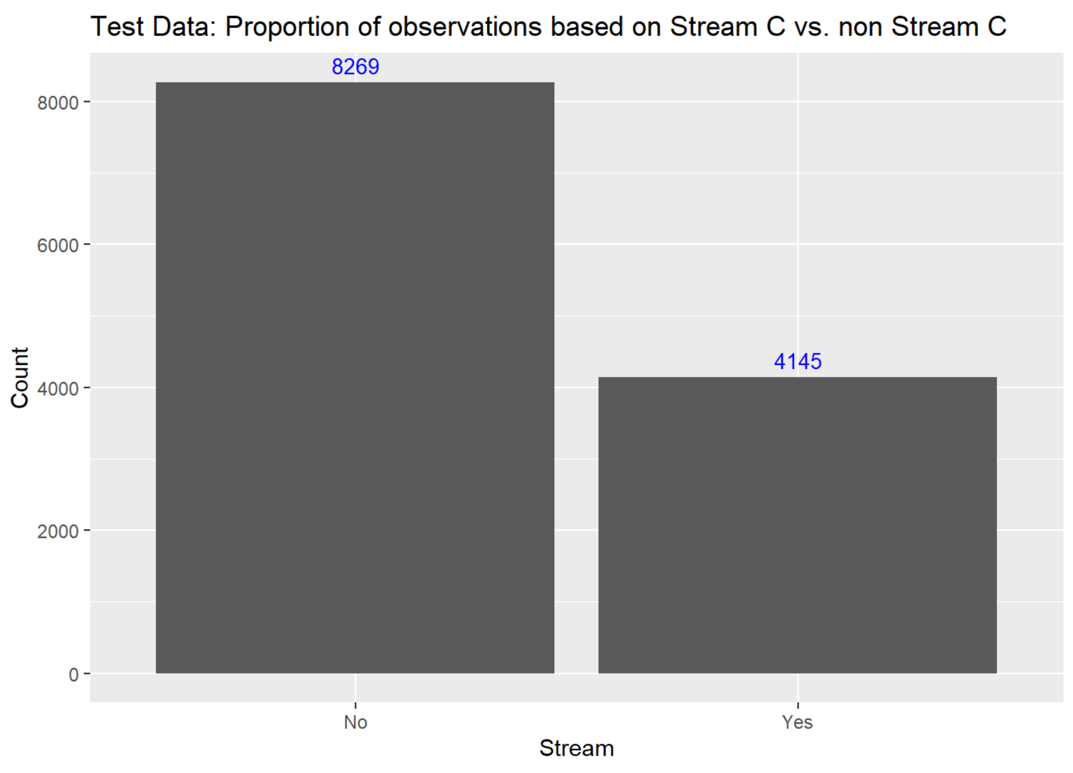
```
traincheck_stream <- data.frame(with(ef_traindata,table(STREAM_relabel_binary)))

ggplot(data=traincheck_stream,aes(y=Freq,x=STREAM_relabel_binary)) +
  geom_col() +
  ggplot2::labs(
            title = "Train Data: Proportion of observations based on Stream C vs. non Stream C ",
            x="Stream",y="Count") +
            geom_text(aes(label=Freq,vjust=-0.5),
                size=3.5,colour="blue")
```



```
calcheck_stream <- data.frame(with(ef_caldata,table(STREAM_relabel_binary)))

ggplot(data=calcheck_stream,aes(y=Freq,x=STREAM_relabel_binary)) +
  geom_col() +
  ggplot2::labs(
          title = "Calibration Data: Proportion of observations based on Stream C vs. non Stream C ",
          x="Stream",y="Count") +
          geom_text(aes(label=Freq,vjust=-0.5),
              size=3.5,colour="blue")
```



```
testcheck_stream <- data.frame(with(ef_testdata,table(STREAM_relabel_binary)))

ggplot(data=testcheck_stream,aes(y=Freq,x=STREAM_relabel_binary)) +
  geom_col() +
  ggplot2::labs(
            title = "Test Data: Proportion of observations based on Stream C vs. non Stream C ",
            x="Stream",y="Count") +
            geom_text(aes(label=Freq,vjust=-0.5),
                size=3.5,colour="blue")
```



6. Modeling

A. NULL Model

```
#Choosing the candidate variables
var_predictors <- dplyr::select(ef_traindata,
                         EXPENDITURE,
                         STATE,
                         GENDER,
                         INDIGENOUS,
                         HOMELESS,
                         PWD,
                         CALD,
                         REFUGEE,
                         EX_OFFENDER,
                         AGE_GROUP_MERGED,
                         UE_GROUP,
                         EDU_LVL_MERGED,
                         EF_CAT_MERGED, STREAM_relabel_binary)

var_dep="STREAM_relabel_binary"

vars_pred_names <- colnames(var_predictors)

#Convert selected variables to categorical data
catVars_pred <- vars_pred_names[sapply(var_predictors
                                       [,vars_pred_names], class)
                                        %in%
                                        c('factor','character')]

pos="Yes"

null_model_predict <- function(y,x){
  p_pos = sum(y==pos)/length(y)
  p_neg = 1-p_pos
  pred=p_pos[x]
  pred[x!=pos]=p_neg
}
```

B. Single Variable Model

```r
#Function for Single Variable Model: Categorical variables
single_catVar_predict <- function(dep_col,var_col,pred_col){
  p_pos = sum(dep_col==pos)/length(dep_col)
  na_Tab = table(as.factor(dep_col[is.empty(var_col)]))
  p_pos_na = (na_Tab/sum(na_Tab))[pos]
  v_Tab = table(as.factor(dep_col),var_col)
  p_pos_v = (v_Tab[pos,]+1.0e-3*p_pos)/(colSums(v_Tab)+1.0e-3)
  pred = p_pos_v[pred_col]
  pred[is.empty(pred_col)] = p_pos_na
  pred[is.empty(pred)] = p_pos
  pred
}

#Predict using the Single Variable Model
for(v in catVars_pred) {

  pred_text=paste('pred',v,sep='_')

  ef_traindata[,pred_text] <- single_catVar_predict(
                        ef_traindata[,var_dep],
                        ef_traindata[,v],
                        ef_traindata[,v])

  ef_caldata[,pred_text] <- single_catVar_predict(
                        ef_traindata[,var_dep],
                        ef_traindata[,v],
                        ef_caldata[,v])


  ef_testdata[,pred_text] <- single_catVar_predict(
                        ef_traindata[,var_dep],
                        ef_traindata[,v],
                        ef_testdata[,v])
}

#Model evaluation for the single-variable model: categorical



calcAUC <- function(pred_col,dep_col) {

  perf <- performance(prediction(pred_col,dep_col==pos),'auc')
  as.numeric(perf@y.values)

}

for (v in catVars_pred) {

  pred_text <- paste('pred',v,sep='_')

  #AUC for train data
  aucTrain <- calcAUC(ef_traindata[,pred_text],ef_traindata[,var_dep])

  #AUC for cal data
  aucCal<- calcAUC(ef_caldata[,pred_text],ef_caldata[,var_dep])

  #AUC for test data
  aucTest<- calcAUC(ef_testdata[,pred_text],ef_testdata[,var_dep])

  print(sprintf("%s, trainAUC: %4.3f ,calAUC: %4.3f , testAUC: %4.3f" ,
                v, aucTrain, aucCal, aucTest))
}
```

```
## [1] "STATE, trainAUC: 0.533 ,calAUC: 0.528 , testAUC: 0.528"
## [1] "GENDER, trainAUC: 0.538 ,calAUC: 0.536 , testAUC: 0.533"
## [1] "INDIGENOUS, trainAUC: 0.504 ,calAUC: 0.504 , testAUC: 0.501"
## [1] "HOMELESS, trainAUC: 0.571 ,calAUC: 0.578 , testAUC: 0.579"
## [1] "PWD, trainAUC: 0.695 ,calAUC: 0.701 , testAUC: 0.704"
## [1] "CALD, trainAUC: 0.514 ,calAUC: 0.516 , testAUC: 0.517"
## [1] "REFUGEE, trainAUC: 0.509 ,calAUC: 0.505 , testAUC: 0.506"
## [1] "EX_OFFENDER, trainAUC: 0.561 ,calAUC: 0.563 , testAUC: 0.566"
## [1] "AGE_GROUP_MERGED, trainAUC: 0.548 ,calAUC: 0.554 , testAUC: 0.552"
## [1] "UE_GROUP, trainAUC: 0.706 ,calAUC: 0.704 , testAUC: 0.695"
## [1] "EDU_LVL_MERGED, trainAUC: 0.576 ,calAUC: 0.584 , testAUC: 0.569"
## [1] "EF_CAT_MERGED, trainAUC: 0.638 ,calAUC: 0.628 , testAUC: 0.624"
## [1] "STREAM_relabel_binary, trainAUC: 1.000 ,calAUC: 1.000 , testAUC: 1.000"
```

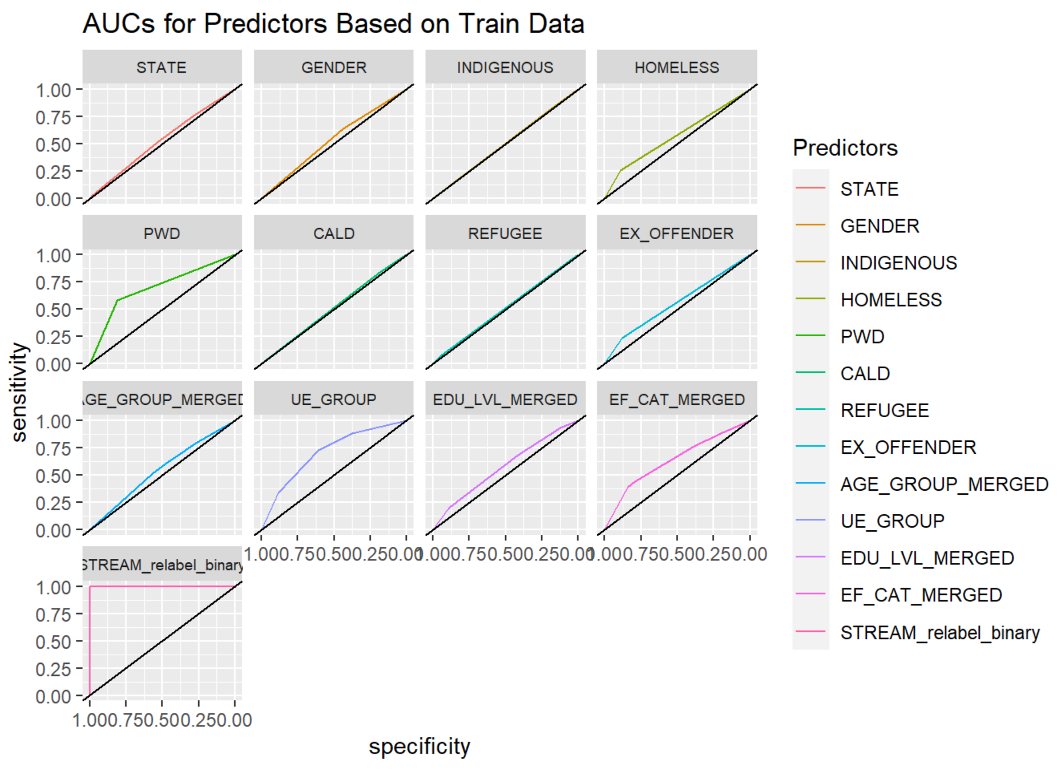
```r
#Model evaluation for the single-variable model: numerical
numVars_pred=vars_pred_names[sapply(var_predictors[,vars_pred_names],class) %in%
c('numeric')]

single_numVar_predict <- function(dep_col,var_col,pred_col) {
  cuts <- unique(as.numeric(
  quantile(var_col, probs=seq(0, 1, 0.1),na.rm=T)))
  varC <- cut(var_col,cuts)
  predC <- cut(pred_col,cuts)
  single_catVar_predict(dep_col,varC,predC)
}

for(v in numVars_pred) {

  pred_text=paste('pred',v,sep='_')

  ef_traindata[,pred_text]=single_numVar_predict(ef_traindata
                                                [,var_dep],ef_traindata[,v],ef_traindata[,v])

  ef_caldata[,pred_text]=single_numVar_predict(ef_traindata
                                              [,var_dep],ef_traindata[,v],ef_caldata[,v])

  ef_testdata[,pred_text]=single_numVar_predict(ef_traindata
                                               [,var_dep],ef_traindata[,v],ef_testdata[,v])
}

#Calculate AUC using calcAUC function above
for (v in numVars_pred) {

  pred_text=paste('pred',v,sep='_')

  aucTrain=calcAUC(ef_traindata[,pred_text],ef_traindata[,var_dep])

  aucCal=calcAUC(ef_caldata[,pred_text],ef_caldata[,var_dep])

  aucTest=calcAUC(ef_testdata[,pred_text],ef_testdata[,var_dep])

  print(sprintf("%s, trainAUC: %4.3f , calAUC: %4.3f , testAUC: %4.3f",v, aucTrain, aucCal, aucTest))
}
```

```
## [1] "EXPENDITURE, trainAUC: 0.596 , calAUC: 0.584 , testAUC: 0.582"
```

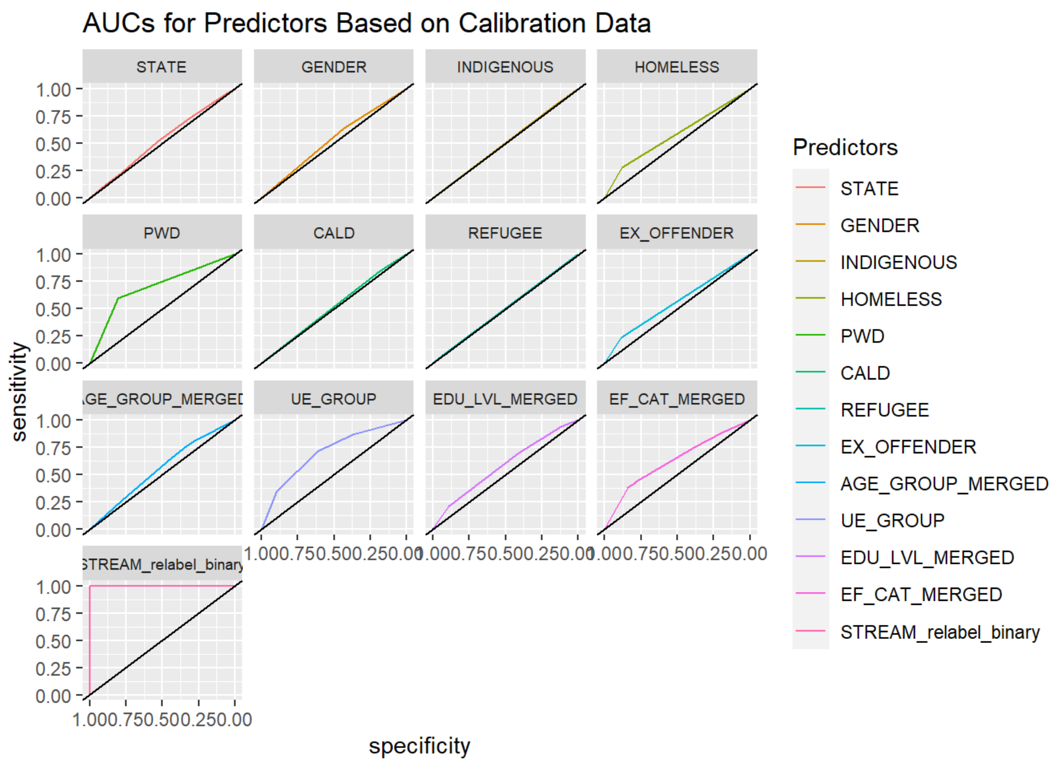
In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test (Zweig & Campbell, 1993).

An AUC of 0.5 represents a test with no discriminating ability (ie, no better than chance), while an AUC of 1.0 represents a test with perfect discrimination.

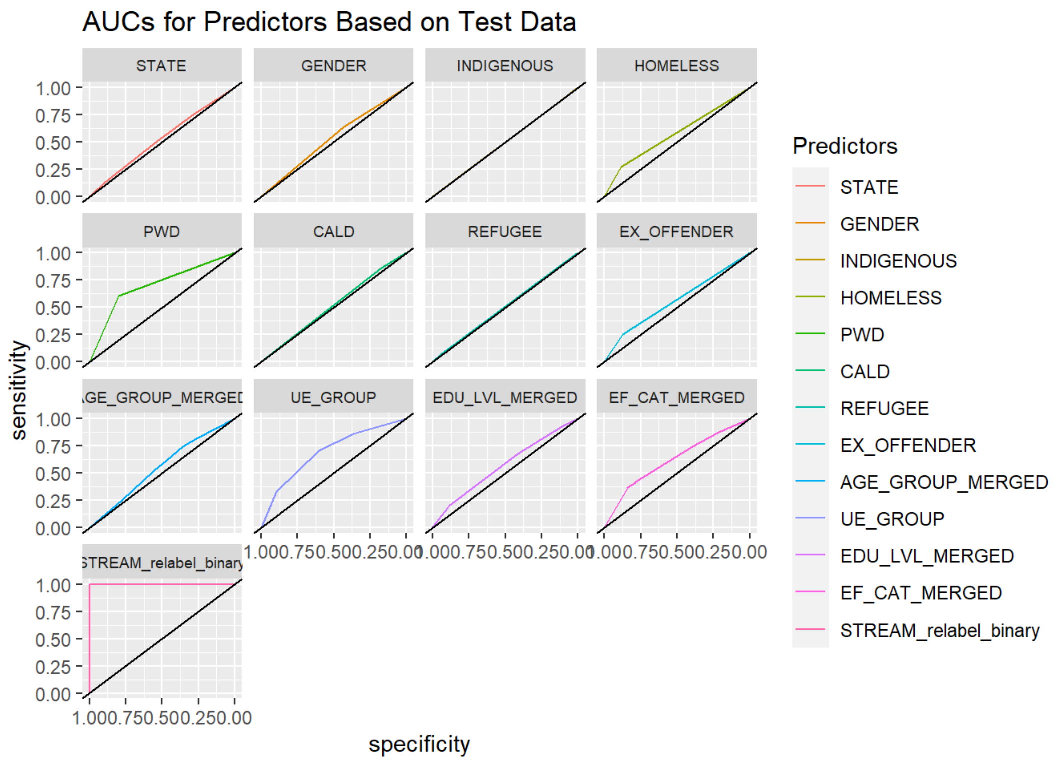
```
#Plotting the UACs from Single Variable Model to compare

roc.sv.train <- list()
roc.sv.cal<- list()
roc.sv.test<- list()

#Train Data
for (v in c(1:length(catVars_pred))){
  pred_text <- paste('pred',catVars_pred[v],sep='_')

  roc_temp <- roc(ef_traindata[,var_dep],ef_traindata[,pred_text])

  roc.sv.train[[v]]=roc_temp

}

names(roc.sv.train) <- c(catVars_pred)

ggroc(roc.sv.train) +
  geom_abline(intercept = 1, slope = 1,colour="black")+
  facet_wrap(.~name)+
  labs(title="AUCs for Predictors Based on Train Data",colour="Predictors")+
  theme(strip.text.x = element_text(size = 7))
```



```
#Calibration Data
for (v in c(1:length(catVars_pred))){
  pred_text <- paste('pred',catVars_pred[v],sep='_')

  roc_temp <- roc(ef_caldata[,var_dep],ef_caldata[,pred_text])

  roc.sv.cal[[v]]=roc_temp

}

names(roc.sv.cal) <- c(catVars_pred)

ggroc(roc.sv.cal)+
  geom_abline(intercept = 1, slope = 1,colour="black")+
  facet_wrap(.~name)+labs(title="AUCs for Predictors Based on Calibration Data",colour="Predictors")+
  theme(strip.text.x = element_text(size = 7))
```



```
#Test Data
for (v in c(1:length(catVars_pred))){
  pred_text <- paste('pred',catVars_pred[v],sep='_')

  roc_temp <- roc(ef_testdata[,var_dep],ef_testdata[,pred_text])

  roc.sv.test[[v]]=roc_temp

}

names(roc.sv.test) <- c(catVars_pred)

ggroc(roc.sv.test)+
  geom_abline(intercept = 1, slope = 1,colour="black")+
  facet_wrap(.~name)+labs(title="AUCs for Predictors Based on Test Data",colour="Predictors")+
  theme(strip.text.x = element_text(size = 7))
```



From the list of single categorical variables, it can be seen that there are a few promising variables. E.g. EF_CAT_MERGED, UE_GROUP, PWD's test and training AUCs scored above 0.60.

Feature Selection

Feature Selection is performed to reduce the number of input variables when developing a predictive model. It is preferable to conduct this process in a way to minimize the input variables then reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

Statistical-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting those input variables that have the strongest relationship with the target variable. These methods can be fast and effective, although the choice of statistical measures depends on the data type of both the input and output variables.

We compute for deviance as a base feature selection method. But to get the deviance, we need to compute for log likelihood of the predicted values from Single Variable model.

```
#Function to compute log likelihood.
logLikelihood <- function(outCol,predCol) {

      sum(ifelse(outCol==pos,log(predCol),log(1-predCol)))
}

#Compute the base rate of a NULL model
baseRateCheck <- logLikelihood(
                    ef_caldata[,var_dep],
                    sum(ef_caldata[,var_dep]==pos)/
                    length(ef_caldata[,var_dep]))


#Pick feature based on a deviance improvement.
selVars <- c()

predictor_val <- c(catVars_pred[catVars_pred!=var_dep],numVars_pred)

deviance_table <- data.frame()
for(v in predictor_val) {
    pi <- paste('pred',v,sep="_")
    liCheck <- 2*((logLikelihood(ef_caldata[,var_dep],ef_caldata[,pi]) - baseRateCheck))
    print(sprintf("%s, calibrationScore: %g",pi,liCheck))
    selVars <- c(selVars,pi)
    deviance_temp=data.frame(predictors=pi,deviance=liCheck)
    deviance_table=rbind(deviance_table,deviance_temp)
}
```

```
## [1] "pred_STATE, calibrationScore: 10.2711"
## [1] "pred_GENDER, calibrationScore: 28.3182"
## [1] "pred_INDIGENOUS, calibrationScore: -0.389181"
## [1] "pred_HOMELESS, calibrationScore: 215.083"
## [1] "pred_PWD, calibrationScore: 970.002"
## [1] "pred_CALD, calibrationScore: 8.99433"
## [1] "pred_REFUGEE, calibrationScore: 0.0895349"
## [1] "pred_EX_OFFENDER, calibrationScore: 150.612"
## [1] "pred_AGE_GROUP_MERGED, calibrationScore: 53.8485"
## [1] "pred_UE_GROUP, calibrationScore: 753.95"
## [1] "pred_EDU_LVL_MERGED, calibrationScore: 149.323"
## [1] "pred_EF_CAT_MERGED, calibrationScore: 369.47"
## [1] "pred_EXPENDITURE, calibrationScore: 141.704"
```

```
#Sort
deviance_table <- deviance_table[order(-deviance_table$deviance),]
deviance_table
```

```
##                predictors     deviance
## 5                pred_PWD 970.00186445
## 10          pred_UE_GROUP 753.95016223
## 12     pred_EF_CAT_MERGED 369.47010996
## 4           pred_HOMELESS 215.08263054
## 8        pred_EX_OFFENDER 150.61233091
## 11    pred_EDU_LVL_MERGED 149.32316423
## 13       pred_EXPENDITURE 141.70386745
## 9  pred_AGE_GROUP_MERGED  53.84847656
## 2             pred_GENDER  28.31815676
## 1              pred_STATE  10.27108763
## 6               pred_CALD   8.99432522
## 7            pred_REFUGEE   0.08953493
## 3         pred_INDIGENOUS  -0.38918071
```

Based on the predictors vs. deviance above, there seems to be quite a few features that we can use such as those having deviance above 5.

B. Logistic Regression Model

Perform Logistic Model

The predictors (variables) selected in deviance test above will be used to fit into logistic regression. To further identify which best set of predictors to be used in the regression, the predictors will be added one by one.

Each model will be compared using Drop-In Deviance test and AIC to determine which set of predictors are significant based on p-value of Drop-In Deviance test and with smallest prediction error based on the value of AIC.

The best model will have the smallest p-value of Drop-In Deviance test and smallest AIC.

```
var_model <- deviance_table[deviance_table$deviance>5,]$predictors

fV=paste(var_dep,'~ .')

#Forward selection logistic model for train data based on deviance in descending order
logmodel_train=list()
for(i in 1:length(var_model)){

  logmodel_train_temp <- glm(fV,data=ef_traindata[,c(var_model[1:i],var_dep)], family = binomial)
  logmodel_train[[i]] <- logmodel_train_temp
}

#Store anova test results to compare the model
model_test=data.frame()

for (i in 2:length(logmodel_train)){
  model_test_temp=data.frame()
  anv_temp <- anova(logmodel_train[[i]],logmodel_train[[1]], test = "Chisq")
  model_test_temp=
    cbind(Model=i,Deviance_reduction=anv_temp$Deviance[2],Pval=anv_temp$`Pr(>Chi)`[2],AIC=logmodel_train[[i]]$aic)
  model_test=rbind(model_test,model_test_temp)
}

model_test
```

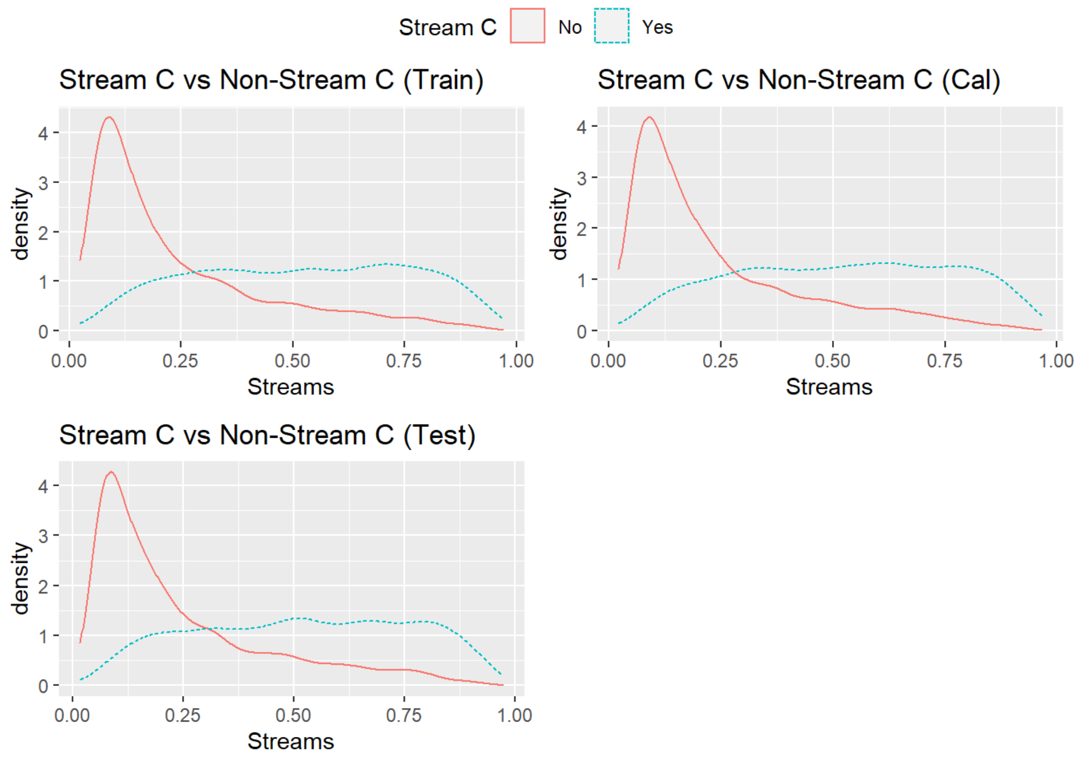
```
##    Model Deviance_reduction          Pval      AIC
## 1      2          -396.7655  2.786397e-88 6405.890
## 2      3          -587.1354 3.200184e-128 6217.520
## 3      4          -713.1147 3.007665e-154 6093.541
## 4      5          -781.6303 7.315200e-168 6027.025
## 5      6          -836.9966 1.145395e-178 5973.659
## 6      7          -838.4945 7.399714e-178 5974.161
## 7      8          -840.5859 3.228340e-177 5974.070
## 8      9          -865.3531 1.676171e-181 5951.302
## 9     10          -888.8088 1.594506e-185 5929.847
## 10    11          -890.7118 6.352201e-185 5929.944
```

Based on Drop-In-Deviance test and AIC, “model 10” has the smallest AIC and p-value. Therefore, model 10 will be used to do prediction. Density plots will be used to identify the probability value that Stream C and non-Stream C intersect at.

```
#Predict the outcomes using logistic regression based on train dataset
ef_traindata$predict_glm_train <- predict(logmodel_train[[10]],ef_traindata[,var_model],type="response")
ef_caldata$predict_glm_cal <- predict(logmodel_train[[10]],ef_caldata[,var_model],type="response")
ef_testdata$predict_glm_test <- predict(logmodel_train[[10]],ef_testdata[,var_model],type="response")


#Model evaluation for the logistic regression model using AUC
glm.calcAUC <- function(pred_col,dep_col) {
    perf <- performance(prediction(pred_col,dep_col==pos),'auc')
    as.numeric(perf@y.values)

}

  glm.aucTrain <- calcAUC(ef_traindata$predict_glm_train,ef_traindata[,var_dep])
  glm.aucCal<- calcAUC(ef_caldata$predict_glm_cal,ef_caldata[,var_dep])
  glm.aucTest<- calcAUC(ef_testdata$predict_glm_test,ef_testdata[,var_dep])

  print(sprintf("glm.trainAUC: %4.3f , glm.calAUC: %4.3f , glm.testAUC: %4.3f",
              glm.aucTrain, glm.aucCal, glm.aucTest))
```

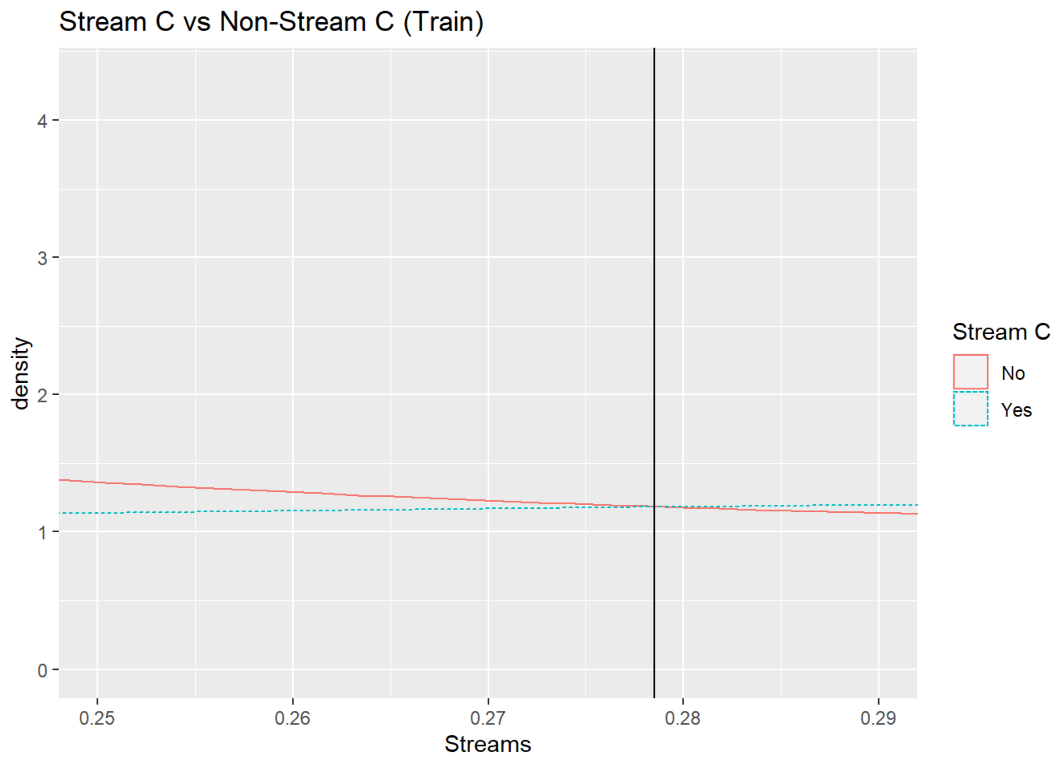
```
## [1] "glm.trainAUC: 0.821 , glm.calAUC: 0.823 , glm.testAUC: 0.816"
```

```
#Density plot to determine the likelihood treshold
train_glm_des<- ggplot(ef_traindata,aes(x=predict_glm_train,color=STREAM_relabel_binary,linetype=STREAM_relabel_binary))+g
eom_density()+
  labs(title="Stream C vs Non-Stream C (Train)",x="Streams",
       linetype="Stream C",colour="Stream C")

cal_glm_des<- ggplot(ef_caldata,aes(x=predict_glm_cal,color=STREAM_relabel_binary,linetype=STREAM_relabel_binary))+geom_de
nsity()+
  labs(title="Stream C vs Non-Stream C (Cal)",x="Streams",
       linetype="Stream C",colour="Stream C")

test_glm_des<- ggplot(ef_testdata,aes(x=predict_glm_test,color=STREAM_relabel_binary,linetype=STREAM_relabel_binary))+geom
_density()+
  labs(title="Stream C vs Non-Stream C (Test)",x="Streams",
       linetype="Stream C",colour="Stream C")

ggarrange(train_glm_des,cal_glm_des,test_glm_des,
          font.label=list(size=5),
          vjust=0.5,
          ncol=2,nrow=2,
          common.legend = TRUE)
```



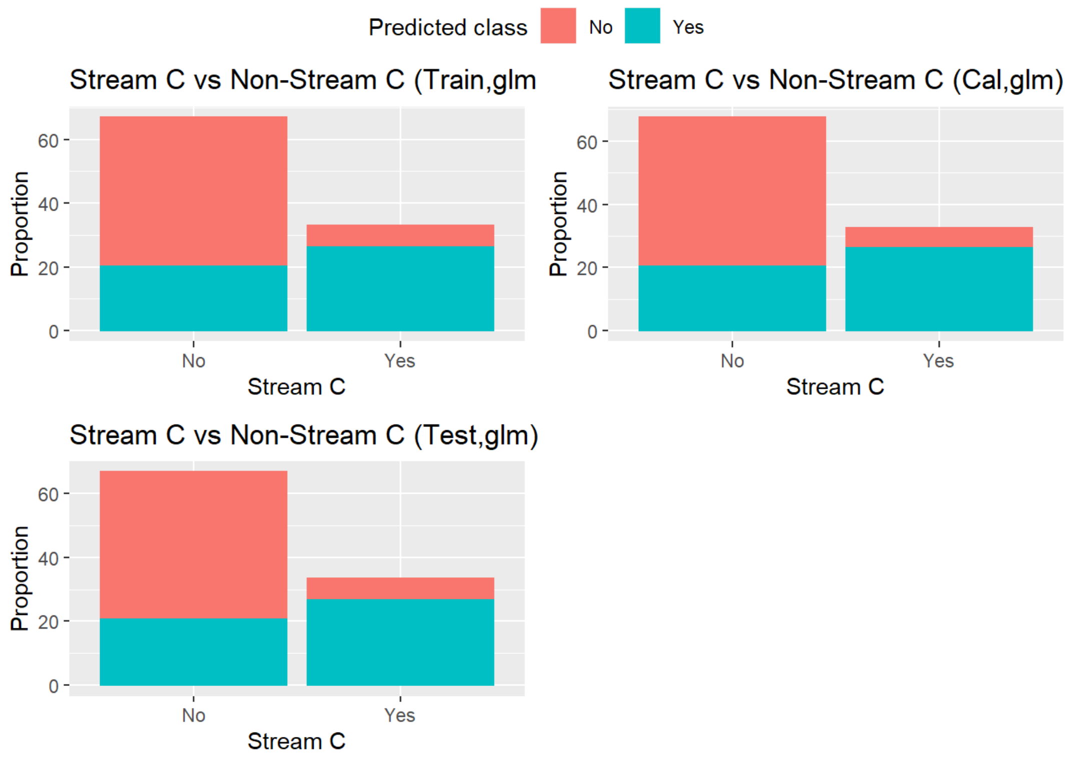
Based on the graphs above, the density curves for Stream C and Non-Stream C for all 3 datasets intersect at one point. Next, the density curves for train data will be zoomed in to look at the intersection point.

```
train_glm_des+ coord_cartesian(xlim = c(0.25, 0.29))+geom_vline(xintercept=0.2785)
```



Based on the graph generated from train data, the intersection point is at 0.2785. Therefore, if the predicted probability value is more than 0.2785, then the applicant will be under Stream C. Else, the applicant will be under non-Stream C. Next, the proportion of predicted Streams versus actual Streams will be visualise using bar chart to look at the proportion of correct classification.

```r
glm_train_df <- data.frame(Actual_cls = ef_traindata$STREAM_relabel_binary,
                           Pred_cls = as.factor(ifelse
                                    (ef_traindata$predict_glm_train>0.2785,"Yes","No")))
glm_train_tab <- data.frame(with(glm_train_df,prop.table(table(Actual_cls,Pred_cls))))


glm_cal_df <- data.frame(Actual_cls=ef_caldata$STREAM_relabel_binary,
                        Pred_cls=
                          as.factor (ifelse(ef_caldata$predict_glm_cal>0.2785,"Yes","No")))

glm_cal_tab <- data.frame(with(glm_cal_df,prop.table(table(Actual_cls,Pred_cls))))

glm_test_df <- data.frame(Actual_cls=ef_testdata$STREAM_relabel_binary,
                        Pred_cls=
                          as.factor(ifelse(ef_testdata$predict_glm_test>0.2785,"Yes","No")))

glm_test_tab <- data.frame(with(glm_test_df,prop.table(table(Actual_cls,Pred_cls))))

train_glm <- ggplot(data=glm_train_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Train,glm)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

cal_glm<- ggplot(data=glm_cal_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Cal,glm)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

test_glm<- ggplot(data=glm_test_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Test,glm)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

ggarrange(train_glm,cal_glm,test_glm,
          font.label=list(size=3),
          vjust=0.5,
          ncol=2,nrow=2,
          common.legend = TRUE)
```



Based on the charts above, logistic regression seem to be better in identifying Stream C applicants. Next, confusion matrix is calculated to look at recall and specificity of the model for each dataset.

Confusion Matrix

```
cf_glm_train <- confusionMatrix(glm_train_df[,2],glm_train_df[,1],
                                positive="Yes",mode="everything")
cf_glm_cal <- confusionMatrix(glm_cal_df[,2],glm_cal_df[,1],
                              positive="Yes",mode="everything")
cf_glm_test <- confusionMatrix(glm_test_df[,2],glm_test_df[,1],
                               positive="Yes",mode="everything")

cf_glm <- list(cf_glm_train,cf_glm_cal,cf_glm_test)

sp_glm <- data.frame()
dataset_name <- c("Train","Calibration","Test")

for(i in c(1:length(cf_glm))){
  sp_glm_temp <- data.frame(dataset=dataset_name[i],Recall=cf_glm[[i]]$byClass["Recall"],Specificity=cf_glm[[i]]$byClass[
"Specificity"])
  sp_glm <- rbind(sp_glm,sp_glm_temp)
}

sp_glm
```

```
##             dataset    Recall Specificity
## Recall        Train 0.7954206   0.6984049
## Recall1 Calibration 0.8087349   0.7009391
## Recall2        Test 0.7961399   0.6919821
```

Based on the table above, logistic regression is able to identify around 80% of actual Stream C applicants and 70% of actual non-Stream C applicants. Logistic regression can be an adequate model to classify Steam C and non-Stream C applicants.

```
#Plotting the UACs from glm model to compare
#Train Data

roc.glm.train <- roc(ef_traindata$STREAM_relabel_binary,ef_traindata$predict_glm_train)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

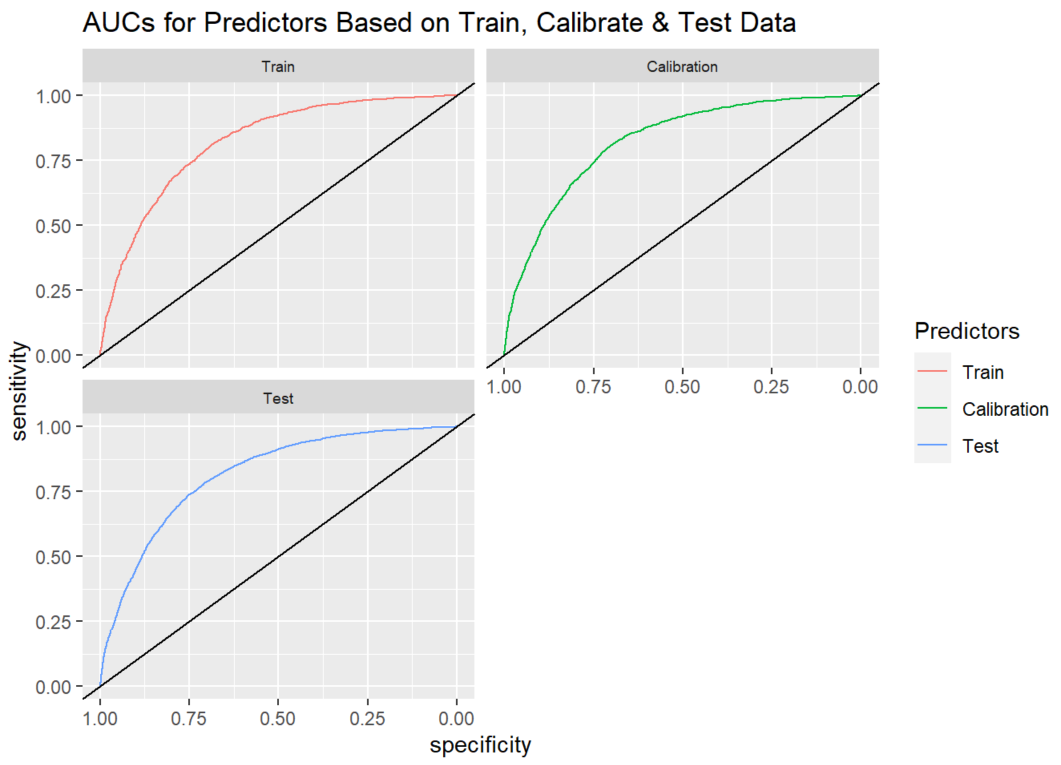
```
roc.glm.cal <- roc(ef_caldata$STREAM_relabel_binary,ef_caldata$predict_glm_cal)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.glm.test <- roc(ef_testdata$STREAM_relabel_binary,ef_testdata$predict_glm_test)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.glm <- list(roc.glm.train,roc.glm.cal,roc.glm.test)

names(roc.glm)=c("Train","Calibration","Test")

ggroc(roc.glm)+
  geom_abline(intercept = 1, slope = 1,colour="black")+
  labs(title="AUCs for Predictors Based on Train, Calibrate & Test Data",colour="Predictors")+
  theme(strip.text.x = element_text(size = 7))+facet_wrap(.~name,nrow=2,ncol=2)
```



The 3 graphs from the splitted datasets used the glm model resulted to a very similar shape of curve away from the AUC diagonal line, this means that the model predicts good.

C. Support vector machine

Perform Support Vector Machine Similar predictors will be used to fit into Support Vector Machine (SVM). To further identify which best set of predictors to be used in the regression, the predictors will be added one by one. From there, AUC for each SVM will be calculated.

SVM with the biggest AUC will be the best SVM model.

```r
#Forward selection SVM and store the models in a list
svmmodel_train <- list()


for(i in 1:length(var_model)){

  svmmodel_train_temp=svm(STREAM_relabel_binary~.,data=ef_traindata[,c(var_model[1:i],var_dep)], kernel="linear",probabili
ty=TRUE)
  svmmodel_train[[i]]=svmmodel_train_temp
}

#Store predicted probability values from SVM models in a data frame
predict_svm_train <- data.frame(ef_traindata[,1])

colnames(predict_svm_train)=colnames(ef_traindata)[1]

for(i in 1:length(svmmodel_train)){
  predict_svm_temp=predict(svmmodel_train[[i]],ef_traindata[,var_model],probability = TRUE)
  predict_prob_temp=data.frame(attr(predict_svm_temp, "probabilities"))[,1]
  predict_svm_train=cbind(predict_svm_train,predict_prob_temp)
  colnames(predict_svm_train)[i+1]=paste("svm_train",i,sep="_")
}

ef_traindata <- cbind(ef_traindata,predict_svm_train[,-1])

#Calculate AUC to identify the best train model
svmtrain_auc=data.frame()

for(i in colnames(predict_svm_train)[-1]){
  pred_text=paste(i)
  svm_temp_auc=data.frame(SVM=pred_text,
               AUC=calcAUC(ef_traindata[,pred_text],ef_traindata[,var_dep]))
  svmtrain_auc=rbind(svmtrain_auc,svm_temp_auc)

}

svmtrain_auc
```

```
##             SVM       AUC
## 1   svm_train_1 0.6952283
## 2   svm_train_2 0.6144591
## 3   svm_train_3 0.6246480
## 4   svm_train_4 0.7988561
## 5   svm_train_5 0.7837589
## 6   svm_train_6 0.8085959
## 7   svm_train_7 0.8068971
## 8   svm_train_8 0.8083319
## 9   svm_train_9 0.8104792
## 10 svm_train_10 0.8136069
## 11 svm_train_11 0.8118262
```

Based on AUC value, SVM model 10 has the highest AUC. Therefore, SVM model 10 will be used to predict Streams in calibration and test data.

```
predict_svm_cal_temp <- predict(svmmodel_train[[10]],ef_caldata[,var_model],probability = TRUE)

ef_caldata$predict_svm_cal <- data.frame(attr(predict_svm_cal_temp, "probabilities"))[,1]

predict_svm_test_temp <- predict(svmmodel_train[[10]],ef_testdata[,var_model],probability = TRUE)

ef_testdata$predict_svm_test <- data.frame(attr(predict_svm_test_temp, "probabilities"))[,1]

  svm.aucTrain <- calcAUC(ef_traindata$svm_train_10,ef_traindata[,var_dep])
  svm.aucCal<- calcAUC(ef_caldata$predict_svm_cal,ef_caldata[,var_dep])
  svm.aucTest<- calcAUC(ef_testdata$predict_svm_test,ef_testdata[,var_dep])

  print(sprintf(" svm.trainAUC: %4.3f , svm.calAUC: %4.3f , svm.testAUC: %4.3f",
               svm.aucTrain, svm.aucCal, svm.aucTest))
```

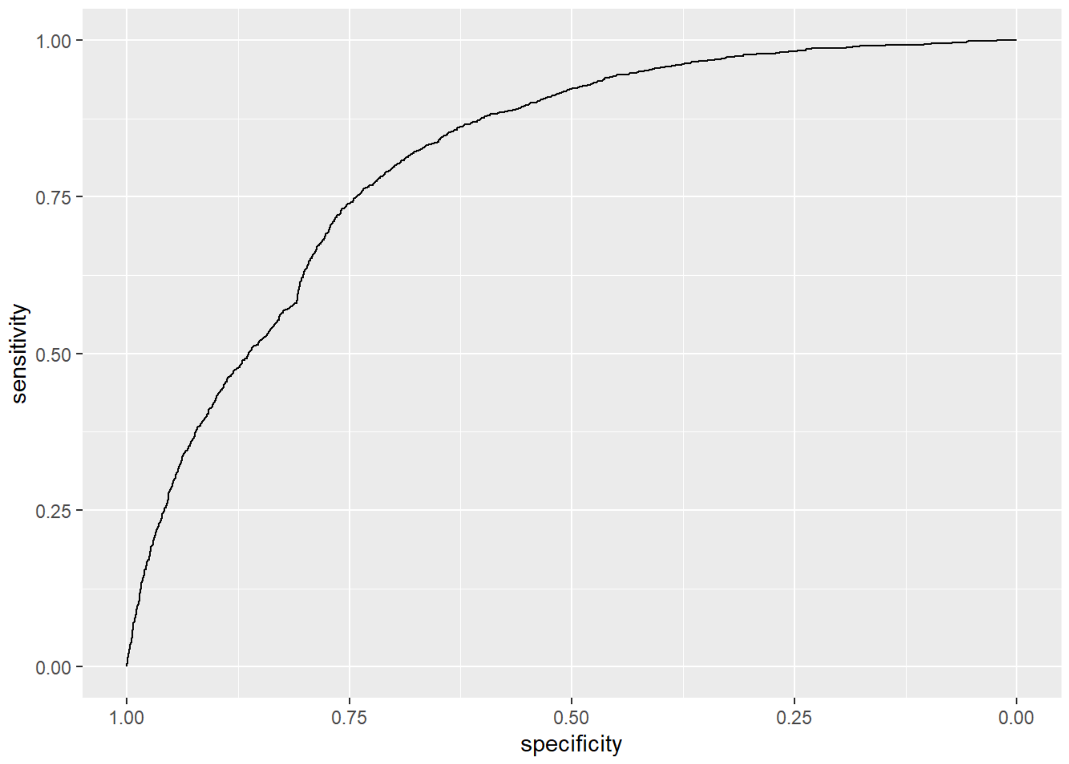
```
## [1] " svm.trainAUC: 0.814 , svm.calAUC: 0.816 , svm.testAUC: 0.811"
```

```
svmtrain_roc <- roc(ef_traindata[,var_dep],ef_traindata$svm_train_10)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

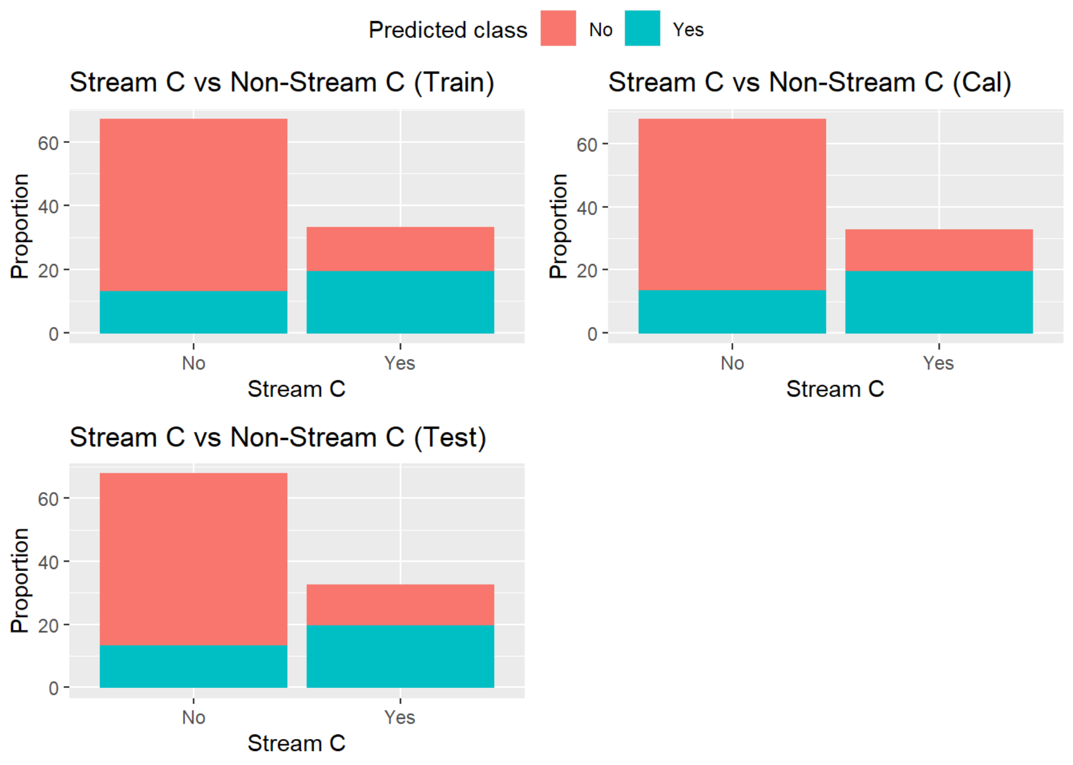
```
ggroc(svmtrain_roc)
```



Based on the values of AUC for train, calibration and test data, all three are approximately the same and above 80%. SVM model 10 seem to be a good model to classify applicants whether they are Stream C or non-Stream C.

Next, the proportion of actual streams will be compared with the proportion of predicted streams using SVM model 10 to look at the proportion of correct classification using bar charts.

```
pred_svm_train <- predict(svmmodel_train[[10]],ef_traindata[,var_model],probability = FALSE)

svm_train_df <- data.frame(Actual_cls=ef_traindata$STREAM_relabel_binary,Pred_cls=pred_svm_train)

svm_train_tab <- data.frame(with(svm_train_df,prop.table(table(Actual_cls,Pred_cls))))


svm_cal_df <- data.frame(Actual_cls=ef_caldata$STREAM_relabel_binary,Pred_cls=predict_svm_cal_temp)

svm_cal_tab <- data.frame(with(svm_cal_df,prop.table(table(Actual_cls,Pred_cls))))

svm_test_df <- data.frame(Actual_cls=ef_testdata$STREAM_relabel_binary,Pred_cls=predict_svm_test_temp)

svm_test_tab <- data.frame(with(svm_cal_df,prop.table(table(Actual_cls,Pred_cls))))

train_svm <- ggplot(data=svm_train_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Train)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

cal_svm=ggplot(data=svm_cal_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Cal)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

test_svm=ggplot(data=svm_test_tab,aes(x=Actual_cls,y=Freq*100,group=Pred_cls,
                              colour=Pred_cls,fill=Pred_cls))+
  geom_col()+
  labs(title="Stream C vs Non-Stream C (Test)",x="Stream C",y="Proportion",
       fill="Predicted class",colour="Predicted class")

ggarrange(train_svm,cal_svm,test_svm,
          font.label=list(size=5),
          vjust=0.5,
          ncol=2,nrow=2,
          common.legend = TRUE)
```



Based on the graphs above, SVM model 10 is better in identifying non-Stream C applicants compared to logistic regression. Next, confusion matrix will be tabulated to look at recall and specificity of the model based on each dataset.

```
cf_svm_train <- confusionMatrix(svm_train_df[,2],svm_train_df[,1],
                             positive="Yes",mode="everything")
cf_svm_cal <- confusionMatrix(svm_cal_df[,2],svm_cal_df[,1],
                           positive="Yes",mode="everything")
cf_svm_test <- confusionMatrix(svm_test_df[,2],svm_test_df[,1],
                            positive="Yes",mode="everything")

cf_svm <- list(cf_svm_train,cf_svm_cal,cf_svm_test)

sp_svm <- data.frame()
dataset_name=c("Train","Calibration","Test")

for(i in c(1:length(cf_svm))){
  sp_svm_temp=data.frame(dataset=dataset_name[i],Recall=cf_svm[[i]]$byClass["Recall"],Specificity=cf_svm[[i]]$byClass["Spe
cificity"])
  sp_svm=rbind(sp_svm,sp_svm_temp)
}

sp_svm
```

```
##              dataset    Recall Specificity
## Recall         Train 0.5808860   0.8095706
## Recall1 Calibration 0.5973896   0.8054418
## Recall2         Test 0.6074789   0.8014270
```

Based on the table above, SVM model 10 is able to identify close to 81% of actual non-Stream C applicants and 60% of actual Stream C applicants.

Findings:

As a conclusion, logistic regression has a better performance in identifying Stream C applicants compared to SVM.

7. Part 2 - Clustering

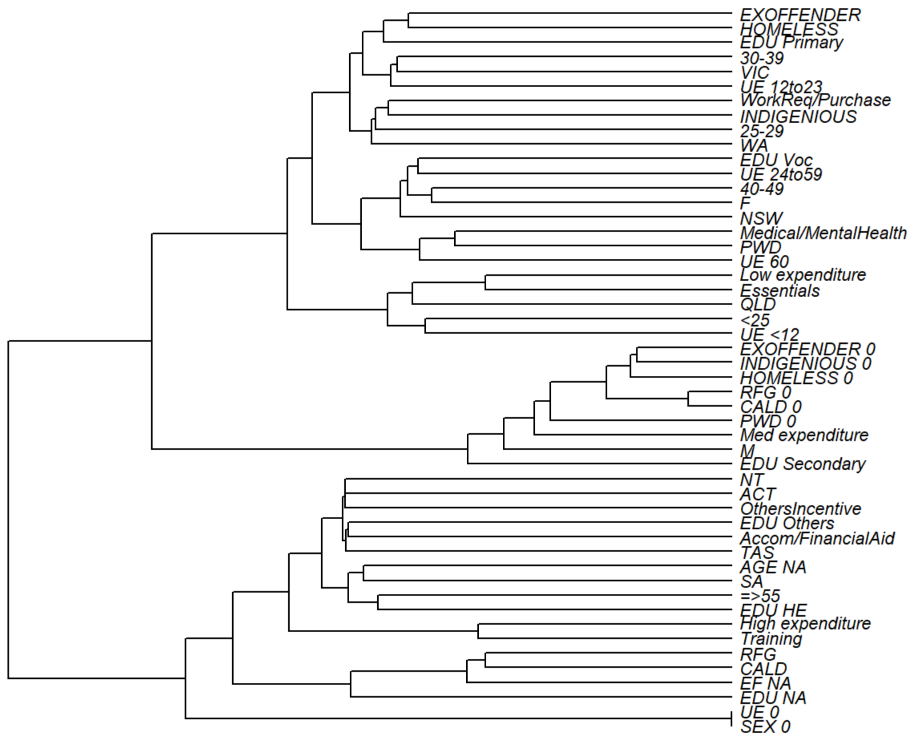
This part will be performing a cluster analysis by “groups” that relates to their distance to each other. Since we are working with a distance matrix, we use hierarchical clustering. For our dataset, since we have mostly categorical variables, then we transformed these variables into their predicted probability figures, that means that the figures are already scaled.We will use the probabilities our clustering analysis particularly calculating the distance matrix. This time we will use the entire combined dataset “ef_data_s” that was used before the splitting of the 3 datasets.

We preferred to use One Hot Encoding values for the clustering to make it more easier for the cluster function to classify data groups or data patterns.

```
#One hot encoding to do hierarchical clustering for entire EF dataset
#Subset data that only merged from the EF dataset
ef_data_s.for.clustering <- ef_data_s[ ,c(2,3,5,6,7,8,9,10,11,13,18,19,20)]



#get the quartile of the expenditure to set low, medium and high
boxplot.stats(ef_data_s.for.clustering$EXPENDITURE)$stats
```

```
## [1]   0.92  44.00 100.00 158.60 330.00
```

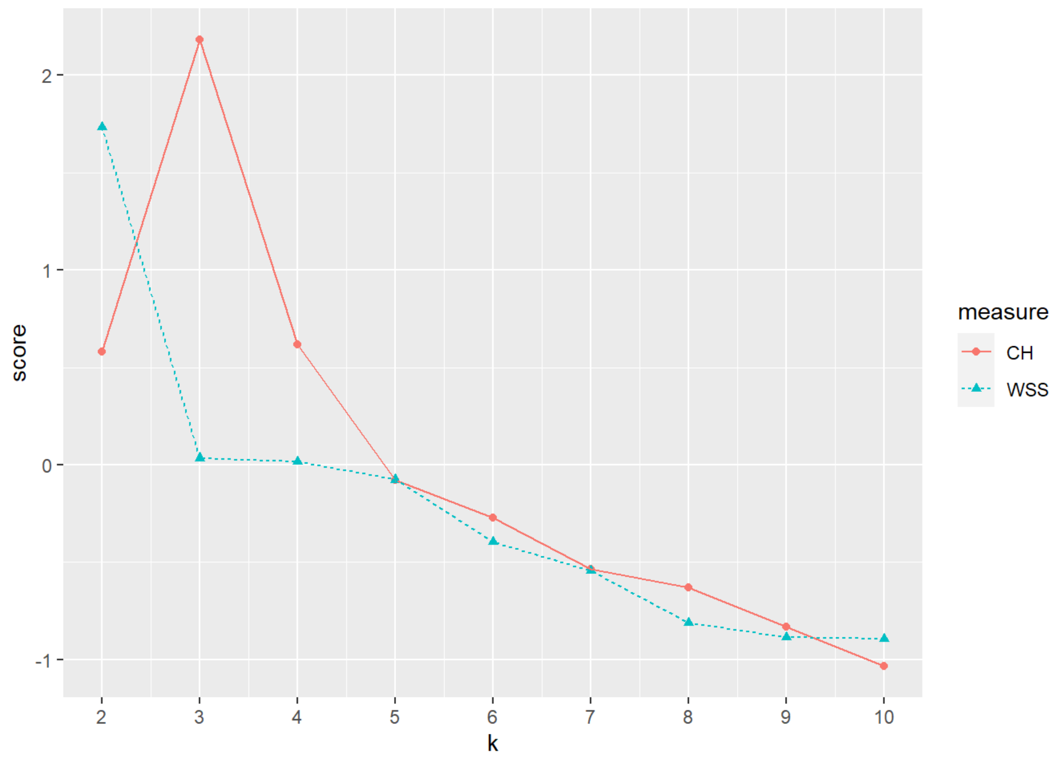
```
ef_data_s.for.clustering$EXP_CAT <- cut(ef_data_s.for.clustering$EXPENDITURE,
                                     breaks=c(-Inf,44,330,Inf),
                                     labels=c("Low","Med","High"))


ef_data_s.for.clustering <- ef_data_s.for.clustering[,-1]


#Perform OHE for these selected vars, separate NA values and put to another column
ohe.ef_data_s <- one_hot(as.data.table(ef_data_s.for.clustering),naCols=TRUE)


#Rename columns to make it tidy
names(ohe.ef_data_s) <- c("ACT","NSW","NT","QLD","SA","TAS","VIC","WA","F","M","SEX_0",
                       "INDIGENIOUS","INDIGENIOUS_0","HOMELESS","HOMELESS_0","PWD","PWD_0",
                       "CALD",
                       "CALD_0","RFG","RFG_0","EXOFFENDER","EXOFFENDER_0","UE_12to23",
                       "UE_24to59",
                       "UE_60","UE_0","UE_<12","EF_NA","Accom/FinancialAid",
                       "Essentials",
                       "Medical/MentalHealth","OthersIncentive","Training",
                       "WorkReq/Purchase","EDU_NA",
                       "EDU_HE","EDU_Others","EDU_Primary",
                       "EDU_Secondary","EDU_Voc","AGE_NA","25-29","30-39","40-49",
                       "=>55","<25","Low_expenditure","Med_expenditure","High_expenditure")

#transpose OHE converted dataset
ohe.ef_data_s_t <- t(ohe.ef_data_s)

#Perform distance matrix calculation
dist.matrix <- dist(ohe.ef_data_s_t, method="binary")

#Clustering using hclust()
ef_hclus <- hclust(dist.matrix, method="ward.D2")
par(mar = c(0, 0, 0, 0))
plot(as.phylo(ef_hclus), cex = 0.7, label.offset = 0.01)
```



Next, we will determine the optimum number of clusters by using WSS and CH index as indicators. Based on CH index, it will help evaluate the most optimal cluster/s.[xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx add here xxxxxxxxxx]

```r
# Function to calculate squared distance
# between two vectors x and y
sqr_edist <- function(x, y) {
sum((x-y)^2)
}
## Function to calculate WSS of a cluster
wss.cluster <- function(clustermat) {

  c0 <- apply(clustermat, 2, FUN=mean)
  sum(apply(clustermat, 1,
  FUN=function(row){sqr_edist(row,c0)}))
}

wss.total <- function(dmatrix, labels) {
  wsstot <- 0
  k <- length(unique(labels))
  for(i in 1:k) {
    wsstot <- wsstot +
    wss.cluster(subset(dmatrix, labels==i))
  }
  wsstot
}

totss <- function(dmatrix) {
  grandmean <- apply(dmatrix, 2, FUN=mean)
  sum(apply(dmatrix, 1,
  FUN=function(row){
    sqr_edist(row, grandmean) }))
}

ch_criterion <- function(dmatrix, kmax, method="kmeans") {
  if(!(method %in% c("kmeans", "hclust"))){
    stop("method must be one of c('kmeans', 'hclust')")
  }

  npts <- dim(dmatrix)[1] # number of rows.
  totss <- totss(dmatrix)
  wss <- numeric(kmax)
  crit <- numeric(kmax)

  wss[1] <- (npts-1)*sum(apply(dmatrix, 2, var))

  for(k in 2:kmax) {
    if(method=="kmeans") {
      clustering<-kmeans(dmatrix, k, nstart=10, iter.max=100)
      wss[k] <- clustering$tot.withinss
    }
    else { # hclust
      d <- dist(dmatrix, method="binary")
      pfit <- hclust(d, method="ward.D2")
      labels <- cutree(pfit, k=k)
      wss[k] <- wss.total(dmatrix, labels)
     }
  }
  bss <- totss - wss
  crit.num <- bss/(0:(kmax-1))
  crit.denom <- wss/(npts - 1:kmax)
  list(crit = crit.num/crit.denom, wss = wss, totss = totss)
}

clustcrit <- ch_criterion(ohe.ef_data_s_t, 10, method="hclust")

CH_index <- data.frame(k=2:10,measure="CH",score=scale(clustcrit$crit[2:10]))

wss_index <- data.frame(k=2:10,measure="WSS",score=scale(clustcrit$wss)[2:10])

wv_index <- data.frame(k=2:10)
```

```
clus_criteria <- rbind(CH_index,wss_index)

ggplot(clus_criteria, aes(x=k, y=score, color=measure)) +
geom_point(aes(shape=measure)) +
geom_line(aes(linetype=measure)) +
scale_x_continuous(breaks=2:10, labels=2:10)
```



Based on the chart above, the number of clusters can be 3 and 4 as they have high scaled CH indices compared to other clusters.

```
ef_hclus3 <- as.factor(cutree(ef_hclus,k=3))

ef_hclus4 <- as.factor(cutree(ef_hclus,k=4))

print_clusters <- function(clusters){
  for(i in 1:length(levels(clusters))){
    cat(paste0('Cluster ',i,':\n'))
    cat(paste(names(clusters[clusters == i]), collapse = "\t"))
    cat('\n\n')
  }
}

print_clusters(ef_hclus3)
```

```
## Cluster 1:
## ACT   NT   SA   TAS SEX_0    CALD     RFG UE_0     EF_NA    Accom/FinancialAid  OthersIncentive Training     EDU_NA   EDU_HE   ED
U_Others   AGE_NA   =>55     High_expenditure
##
## Cluster 2:
## NSW   QLD VIC WA   F    INDIGENIOUS HOMELESS     PWD EXOFFENDER  UE_12to23    UE_24to59    UE_60    UE_<12   Essentials   Medica
l/MentalHealth     WorkReq/Purchase     EDU_Primary EDU_Voc 25-29    30-39    40-49    <25 Low_expenditure
##
## Cluster 3:
## M     INDIGENIOUS_0    HOMELESS_0  PWD_0    CALD_0   RFG_0    EXOFFENDER_0     EDU_Secondary    Med_expenditure
```

```
table(ef_hclus3)
```

```
## ef_hclus3
##  1  2  3
## 18 23  9
```

Interpretation: - Cluster 1: This cluster tells us about the cities mentioned - Cluster 2: - Cluster 3:

```
print_clusters(ef_hclus4)
```

```
## Cluster 1:
## ACT  NT  SA  TAS CALD    RFG EF_NA   Accom/FinancialAid  OthersIncentive Training    EDU_NA  EDU_HE  EDU_Others  AGE_NA
=>55    High_expenditure
##
## Cluster 2:
## NSW  QLD VIC WA  F   INDIGENIOUS HOMELESS    PWD EXOFFENDER  UE_12to23   UE_24to59   UE_60   UE_<12  Essentials  Medica
l/MentalHealth    WorkReq/Purchase    EDU_Primary EDU_Voc 25-29   30-39   40-49   <25 Low_expenditure
##
## Cluster 3:
## M    INDIGENIOUS_0   HOMELESS_0  PWD_0   CALD_0  RFG_0   EXOFFENDER_0    EDU_Secondary   Med_expenditure
##
## Cluster 4:
## SEX_0    UE_0
```

```
table(ef_hclus4)
```

```
## ef_hclus4
##  1  2  3  4
## 16 23  9  2
```

Using 3 clusters, applicants without gender or unemployment period are classified into 3rd cluster instead of 1st cluster.

Now, we validate how strong for each cluster using clusterboot function.

```
cboot.hclust3 <- clusterboot(
ohe.ef_data_s_t, clustermethod=hclustCBI,
method="ward.D2", k=3)
```

```
## boot 1
## boot 2
## boot 3
## boot 4
## boot 5
## boot 6
## boot 7
## boot 8
## boot 9
## boot 10
## boot 11
## boot 12
## boot 13
## boot 14
## boot 15
## boot 16
## boot 17
## boot 18
## boot 19
## boot 20
## boot 21
## boot 22
## boot 23
## boot 24
## boot 25
## boot 26
## boot 27
## boot 28
## boot 29
## boot 30
## boot 31
## boot 32
## boot 33
## boot 34
## boot 35
## boot 36
## boot 37
## boot 38
## boot 39
## boot 40
## boot 41
## boot 42
## boot 43
## boot 44
## boot 45
## boot 46
## boot 47
## boot 48
## boot 49
## boot 50
## boot 51
## boot 52
## boot 53
## boot 54
## boot 55
## boot 56
## boot 57
## boot 58
## boot 59
## boot 60
## boot 61
## boot 62
## boot 63
## boot 64
## boot 65
## boot 66
## boot 67
## boot 68
## boot 69
## boot 70
## boot 71
## boot 72
## boot 73
## boot 74
## boot 75
## boot 76
## boot 77
## boot 78
## boot 79
## boot 80
## boot 81
## boot 82
## boot 83
## boot 84
## boot 85
## boot 86
## boot 87
## boot 88
## boot 89
## boot 90
## boot 91
## boot 92
## boot 93
## boot 94
## boot 95
## boot 96
## boot 97
## boot 98
## boot 99
## boot 100
```

```
cboot.hclust4 <- clusterboot(
ohe.ef_data_s_t, clustermethod=hclustCBI,
method="ward.D2", k=4)
```

```
## boot 1
## boot 2
## boot 3
## boot 4
## boot 5
## boot 6
## boot 7
## boot 8
## boot 9
## boot 10
## boot 11
## boot 12
## boot 13
## boot 14
## boot 15
## boot 16
## boot 17
## boot 18
## boot 19
## boot 20
## boot 21
## boot 22
## boot 23
## boot 24
## boot 25
## boot 26
## boot 27
## boot 28
## boot 29
## boot 30
## boot 31
## boot 32
## boot 33
## boot 34
## boot 35
## boot 36
## boot 37
## boot 38
## boot 39
## boot 40
## boot 41
## boot 42
## boot 43
## boot 44
## boot 45
## boot 46
## boot 47
## boot 48
## boot 49
## boot 50
## boot 51
## boot 52
## boot 53
## boot 54
## boot 55
## boot 56
## boot 57
## boot 58
## boot 59
## boot 60
## boot 61
## boot 62
## boot 63
## boot 64
## boot 65
## boot 66
## boot 67
## boot 68
## boot 69
## boot 70
## boot 71
## boot 72
## boot 73
## boot 74
## boot 75
## boot 76
## boot 77
## boot 78
## boot 79
## boot 80
## boot 81
## boot 82
## boot 83
## boot 84
## boot 85
## boot 86
## boot 87
## boot 88
## boot 89
## boot 90
## boot 91
## boot 92
## boot 93
## boot 94
## boot 95
## boot 96
## boot 97
## boot 98
## boot 99
## boot 100
```

```
cboot.hclust6 <- clusterboot(
ohe.ef_data_s_t, clustermethod=hclustCBI,
method="ward.D2", k=6)
```

```
## boot 1
## boot 2
## boot 3
## boot 4
## boot 5
## boot 6
## boot 7
## boot 8
## boot 9
## boot 10
## boot 11
## boot 12
## boot 13
## boot 14
## boot 15
## boot 16
## boot 17
## boot 18
## boot 19
## boot 20
## boot 21
## boot 22
## boot 23
## boot 24
## boot 25
## boot 26
## boot 27
## boot 28
## boot 29
## boot 30
## boot 31
## boot 32
## boot 33
## boot 34
## boot 35
## boot 36
## boot 37
## boot 38
## boot 39
## boot 40
## boot 41
## boot 42
## boot 43
## boot 44
## boot 45
## boot 46
## boot 47
## boot 48
## boot 49
## boot 50
## boot 51
## boot 52
## boot 53
## boot 54
## boot 55
## boot 56
## boot 57
## boot 58
## boot 59
## boot 60
## boot 61
## boot 62
## boot 63
## boot 64
## boot 65
## boot 66
## boot 67
## boot 68
## boot 69
## boot 70
## boot 71
## boot 72
## boot 73
## boot 74
## boot 75
## boot 76
## boot 77
## boot 78
## boot 79
## boot 80
## boot 81
## boot 82
## boot 83
## boot 84
## boot 85
## boot 86
## boot 87
## boot 88
## boot 89
## boot 90
## boot 91
## boot 92
## boot 93
## boot 94
## boot 95
## boot 96
## boot 97
## boot 98
## boot 99
## boot 100
```

Perform bootstrapping to get the average error of the samples to validate the cluster strength.

```
#Perform bootstraping for the chosen clusters
1-cboot.hclust3$bootbrd/100
```

```
## [1] 1.0 0.8 1.0
```

```
1-cboot.hclust4$bootbrd/100
```

```
## [1] 1.00 0.53 0.57 0.97
```

Based on the result above, 2 clusters: both clusters are quite solid despite the number of features in each cluster is split unequally with 1st cluster having 38 features.

3 clusters: 3 clusters are quite solid but 2nd cluster is slightly weaker as its stability is at 73%.

8 clusters: 1st and 5th clusters are strong indicating that these two clusters are solid real clusters. There are 3 clusters, 2nd, 7th and 8th, managed to identify pattern but with low certainty. The rest are considered unstable.

8. Conclusion

We've used the Employment Fund dataset to fit a predictive classifier that, given a STREAM_PLACEMENT_DESC information, it can decide if a transaction is part of "Stream C" or "not Stream C". We split the data in 3 parts to cross validate the performance of the predictive models created. In addition to the modeling task, we performed a clustering analysis to groups from the different features in the dataset. The next step here would be testing out new cluster analysis while removing some features for classifier from the dataset such as "Gender".

9. Appendix

Section 1: Column Definition

- DATA_AS_AT: Extract date of information from the Employment Services Systems.

- TRANSACTION_ID: Unique identifier of the EF transaction.
- EXPENDITURE: Indicates the dollar amount of the EF expenditure (including GST) transaction.
- STATE: Three-character code that identifies the Australian State of the provider site
- JSKR_LINKAGE_KEY: Unique identifier assigned to the Job seeker by the Department.
- STREAM_PLACEMENT_DESC: The placement type (Stream) of the job seeker at the time of the EF transaction.
  - **Stream A** - Job seekers are the most job ready. They will receive services to help them understand what employers want and how to navigate the local labour market, build a resume, look for jobs and learn how to access self-help facilities
  - **Stream B** - Job seekers need their jobactive provider to play a greater role to help them become job ready and will be referred for case management support.
  - **Stream C** - Job seekers have a combination of work capacity and personal issues that need to be addressed and will get case management support so that they can take up and keep a job.
- EF_CATEGORY_DESCRIPTION: Description of the category of the goods or services purchased including Accredited Training, Clothing and Presentation, Driving Lessons, Employer Required Training, Food, Phone and Petrol Cards/Vouchers, Job Seeker Transport, Medical Expenses, PaTH Internship Placement Costs, Relocation Assistance, Rent and Crisis Accommodation, Stream C only Assistance, Targeted Pre-Employment Preparation, Tools, Books, Equipment and Mobile Phones, Work Related Items, Work Related Licensing, Work Trials, etc
- DERIVED_DATE_OF_SERVICE: identified as the closest date available to the job seeker receiving the service
- ACTIVITY_TYPE_DESC: activity associated with the expenditure.
- ACTIVITY_SUB_DESC: Description of the sub-type of the activity associated with the expenditure.
- EDUCATION_LEVEL_DESC: Description of the job seeker's highest level of education at the time of the EF transaction.
- GENDER: Indicates the job seeker's gender
- INDIGENOUS: Indicates whether the job seeker identified as Indigenous
- HOMELESS: Indicates whether the job seeker identified as having housing instability/homelessness
- PWD: Indicates whether the job seeker identified as having a disability
- CALD: Indicates whether the job seeker has a Culturally and Linguistically Diverse background
- REFUGEE: Indicates whether the job seeker identified as a refugee
- EX_OFFENDER: Indicates whether the job seeker identified as an ex-offender
- AGE_GROUP: Indicates the group of the job seeker's age by category: Under 22 years, 22 to 24 years, 25 to 29 years, 30 to 39 years, 40 to 49 years, 50 to 54 years, 55 to 59 years, 60+ years
- UE_GROUP: Description of the unemployment group, based on the total months the job seeker has been participating in employment services grouped by Under 12 Months, 12 to 23 Months, 24 to 59 Months, 60+ Months