Machine Learning Modelling

Fanchao(Franco) MENG

14 November, 2023

Abstract

This is a continuation of Project 1 - Exploratory Data Analysis, by utilizing supervised and unsupervised machine learning model to further discover useful insights.

This report uses “Global YouTube Statistics 2023” dataset : (https://www.kaggle.com/datasets/nelgiriyewithana/global-youtube-statistics-2023 (https://www.kaggle.com/datasets/nelgiriyewithana/global-youtube-statistics-2023)), obtained from Kaggle, collected by Nidula Elgiriyewithana.

The dataset has included top 995 Youtubers, based on, and ordered by the number of their subscribers.

Literature Review

Digital platforms, have played a pivotal roles in our current economy, politics, and everyday living. More and more content creators start to build career through their Youtube channels, and successful creators are able to enjoy the lavish profit and rewards Youtube provides. In 2022, Punam Wakel (2022) mentioned ‘over the past five years YouTube has paid out quite $5 billion to YouTube content creators. widespread YouTuber PewDiePie created $5 million in 2016 from YouTube alone’.

According to Forbes’s 2022 list, the highest-paid Youtubers were Mr.Beast, with 150 million subscribers, making a staggering $54 million in 2022.

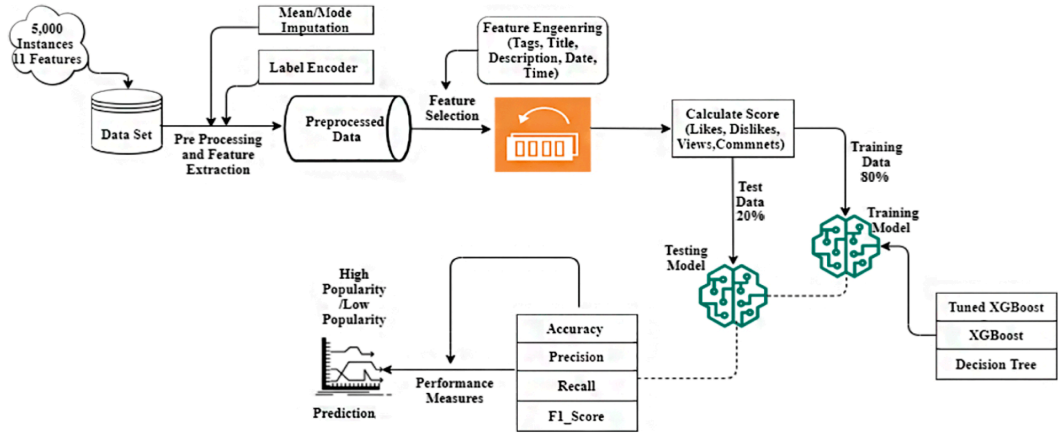
Although people believe the direct relationships between ‘video views’ and ‘earning’, and is hard to verify how YouTube really pay the profit to content creators, but based on many online non-academic resources, earning are not purely on the number of video views, but rather on ‘ad views’, ’each YouTuber has an individual CPM (cost per 1,000 views), and it is correlated with how engaged audiences are, and how much ads have been viewed.

While there are not many research directly investigate on the YouTube income, but many explorations on the YouTube itself, regarding the views, subscribers, and popularity, which are also meaningful for our study on YouTube earnings:

Mathias Bärtl (2018) provided an overall characterization of YouTubes, that ‘a vast majority of on average 85% of all views goes to a small minority of 3% of all channels’. His finding also shows that ’older channels have a significantly higher probability to garner a large viewerships, but also shows that there has always been a small chance for young channels to become successful quickly, depending on whether they choose their genre wisely”.

This finding has provided some insights, of indication that possibly created_year, and genre, may be useful indicators for our income high/low classifications.

Meher UN Nisa (2021) has started his research paper by pointing out, that ‘YouTube is a source of income for many people, and therefore a video’s popularity ultimately becomes the top priority for sustaining a steady income’, his team has performed interesting prediction of YouTube video popularity by using XGBoost. Below graph has shown their workflow and setup, which are extremely useful for our machine learning process.



Meher UN Nisa (2021) concluded that, ‘video quality’, ‘video duration’ play a vital role in for video to become viral. These information are not gathered in out dataset, but maybe informative, or being good indicators on whether the income of YouTubers are high, or low.

Punam Wakel (2022) has also used linear regression, polynomial regression, K-Neighbors Classifier, Decision Tree Regression to predict YouTuber popularity. The model contained features including ‘likes’, ‘dislike’, ‘views’, ‘comments count’, and ‘subscribers’.

1.Data Preparation, Transformation

1.1 Installing Packages, loading libraries.

```
library(ggplot2)
library(wesanderson)
library(corrplot)
library(dplyr)
library(lubridate)
library(ROCR)
library(glmnet)
library(caTools)
library(caret)
library(randomForest)
library(countrycode)
#library(tidyverse)
library(kableExtra)
library(knitr)
library(pander)
library(psych)
library(gridExtra)
library(RColorBrewer)
library(vtreat)
library(hexbin)
library(lime)
library(ROCit)
```

1.2 Loading Data, Target Variable Selection.

```
df.1 <- read.csv("project_2.csv")
#str(raw_data)
#summary(raw_data)
#View(df.1)
```

```
df.2 <- subset(df.1, select = -c(Population,Gross.tertiary.education.enrollment....,Unemployment.rate,Urban_population,Latit
ude,Longitude))
#colnames(df.2)
```

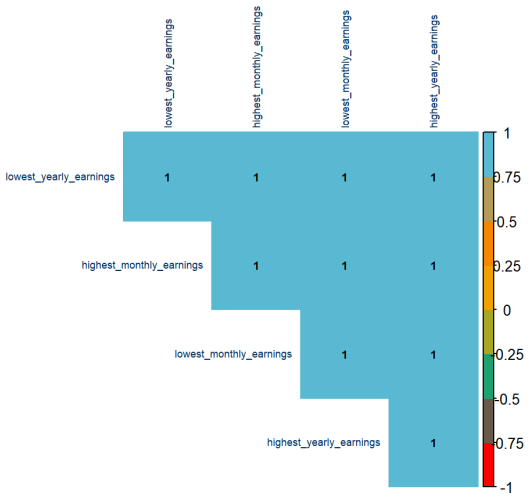
Data were loaded from Project 1 EDA, however I have retrieved the data before all the missing values were filled by Vtreat packages. The missing values will be dealt carefully to suit machine learning model, with new 'variable_no_na' column created, in order to maintain integrity of the original data, similar with "Is_Bad" columns created by VTreat package, as some models are capable of dealing the missing values, and N/A may be an useful category/level in certain instances.

In order to choose appropriate target variable from our 4 "earning" related variables. Below two considerations have been made :

- how correlated are these columns ?
- which column has the least numbers of missing values?

```
earning_subset <- df.2[, which(names(df.2) %in% c("lowest_monthly_earnings","lowest_yearly_earnings","highest_monthly_earnin
gs","highest_yearly_earnings"))]

earning_subset <- na.omit (earning_subset)

M_1 <- cor(earning_subset)
test_Res_1 = cor.mtest(earning_subset, conf.level = 0.95)

corrplot( M_1, p.mat = test_Res_1$p, method = 'color', tl.col="#003366", col=wes_palette(8, name = "Darjeeling1", type = "co
ntinuous"),  type = 'upper', insig='pch', addCoef.col ='black', tl.cex=0.55, number.cex = 0.6,number.font = 2, order = 'hclu
st',diag=TRUE , mar=c(1,1,2,1))
title (main= "Fig.1 Correlation between 4 earning columns",cex.main=0.8, adj = 0.2, line = 3)
```

Fig.1 Correlation between 4 earning columns



```
my_na_function <- function(dataframe, digit, string ) {
  if(missing(digit) & missing(string)) {
    result <- apply(X = is.na(dataframe), MARGIN = 2, FUN = sum)
    round (result/nrow(dataframe),2)
  }
  else if (missing(digit)){
    result <- apply(X = (is.na(dataframe) | dataframe == string), MARGIN = 2, FUN = sum)
    round (result/nrow(dataframe),2)
  }
  else if (missing(string)){
    result <- apply(X = (is.na(dataframe) | dataframe == digit), MARGIN = 2, FUN = sum)
    round (result/nrow(dataframe),2)
  }
  else {
    result <- apply(X = (is.na(dataframe) | dataframe == digit | dataframe == string), MARGIN = 2, FUN = sum)
    round (result/nrow(dataframe),2)
  }
}

#sprintf("%0.1f%%", my_na_function (subset(df.2, select = c(lowest_monthly_earnings,highest_monthly_earnings,        #Lowest
_yearly_earnings,highest_yearly_earnings)) , 0,"Unknown"))

missing_result <- as.data.frame(my_na_function (subset(df.2, select = c(lowest_monthly_earnings,highest_monthly_earnings, lo
west_yearly_earnings,highest_yearly_earnings)) , 0,"Unknown"))
colnames (missing_result) <- c( "Percentage")

kable(missing_result,  booktabs = TRUE, longtable = F, caption = "** Table A: Missing Value Percentages **") %>%
  kable_styling(font_size = 12)%>%
  kable_styling(bootstrap_options = "bordered") %>% row_spec(4, background = "#74A089")
```

** Table A: Missing Value Percentages **

	Percentage
lowest_monthly_earnings	0.11
highest_monthly_earnings	0.08
lowest_yearly_earnings	0.08
highest_yearly_earnings	0.07

From the results above correlation map, and NA percentage calculation function:

- Fig.1 : Due to strong correlation between all 4 columns, it is meaningful to choose any of these as our target variables.
- Table A : Column “Highest_monthly_earnings” is having the least numbers of missing value, therefore has been chosen for the target variable.

*Both 0 (zero) and N/A have been considered as missing values.

1.3 Merging GDP Data.

Two assumptions have been made :

- all the income were recorded in US dollar in the data, it is more meaningful to incorporate **country** in order to classify low/high income.
- the “country” information indicate both “where the YouTube channel originates”, and “where the account holder resides”.

In order to making more meaningful classification, the latest 2022 GDP data was used to normalized the income figures.

For example, an yearly income of $30,000 may be classified as **Low Income** in developed country, but realistically this should be classified as **High Income** in developing / undeveloped countries, due to significant lower living costs.

- In the cases where certain countries having no 2022 gdp data recorded, the most recent record in previous years data will be used.

```
df.gdp <- read.csv("API_NY.GDP.PCAP.CD_DS2_en_csv_v2_5871588.csv", skip = 4)

years_to_remove <- paste0 ('X', seq(1960,2013))
years_to_remove <- c('X',years_to_remove)

df.gdp <- subset(df.gdp, select = ! (colnames(df.gdp) %in% (years_to_remove)))
df.gdp$X2022 <- ifelse(is.na(df.gdp$X2022), df.gdp$X2021, df.gdp$X2022)
df.gdp$X2022 <- ifelse(is.na(df.gdp$X2022), df.gdp$X2020, df.gdp$X2022)  #for cuba
df.gdp$X2022 <- ifelse(is.na(df.gdp$X2022), df.gdp$X2014, df.gdp$X2022)  #for veVenezuela
df.gdp$Country.Name[df.gdp$Country.Name=='Korea, Rep.'] <- 'South Korea'
df.gdp$Country.Name[df.gdp$Country.Name=='Russian Federation'] <- 'Russia'
df.gdp$Country.Name[df.gdp$Country.Name=='Turkiye'] <- 'Turkey'
df.gdp$Country.Name <- sub(",.*", "", df.gdp$Country.Name)

#df.gdp
```

```
df.countrynames <- unique(df.2$Country)
df.countrynames [!(df.countrynames%in% unique(df.gdp$Country.Name))]
```

```
## [1] "Unknown"
```

Above code has checked that all countries have been merged successfully, apart from rows with “Unknown” in Country column. Below code has used the mean value of GDP data, to fill the N/A values where the country is ‘unknown’.

```
df.gdp <- subset(df.gdp, select= c(Country.Name, X2022))
names<-colnames(df.gdp)
df.gdp[nrow(df.gdp)+1,] <- list('Unknown',mean(df.gdp$X2022[!is.na(df.gdp$X2022)]))
#head(df.gdp)
```

```
df.3<-left_join(df.2,df.gdp, by=c("Country"="Country.Name"))
#head(df.3)
```

```
#df.3[ which(is.na(df.3$highest_yearly_earnings )),]
df.3 <- df.3[- which(is.na(df.3$highest_yearly_earnings )),]
```

1.4 Normalizing income data by using merged GDP data, then classifying results into 'High/low' income groups

- All the income figures have been normalized / adjusted as below.
- A new **income** column has been created, where:
  - **1 : High Income**
  - **0 : Low Income**
- The target variable is balanced.

```
Normalise_index <- df.3[df.3$Country == 'United States', 'X2022'][1]
df.3$Normalised_Income <- df.3$highest_yearly_earnings * (Normalise_index/df.3$X2022)


med<-median(df.3$Normalised_Income)

df.3$income <- ifelse(df.3$Normalised_Income>med, 1, 0)

kbl(df.3[,c('highest_yearly_earnings','Normalised_Income','Country', 'income')],longtable = F, booktabs = TRUE,format.args =
list(big.mark = ","),caption = "**Final Results of Nomalised Earning Info**  ")%>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")%>%
  column_spec(5, background = "#78B7C5")%>%
  scroll_box(width = "80%", height = "280px")
```

Final Results of Nomalised Earning Info

	highest_yearly_earnings	Normalised_Income	Country	income
1	1.084e+08	3.467108e+09	India	1
2	5.800e-01	5.800000e-01	United States	0
3	6.470e+07	6.470000e+07	United States	1
4	9.480e+07	9.480000e+07	United States	1
5	8.750e+07	2.798634e+09	India	1

1.5 Merge Continent information.

By using package **countrycode**, the data have been enriched with continent information based on country. Due to many countries have extremely low instances in our data. Having a more broader classification by using continent information may become handy.

```
df.3$continent <- countrycode(sourcevar = df.3$Country, origin = "country.name", destination = "continent")
df.3$continent <- ifelse(is.na(df.3$continent),'Unknown',df.3$continent)
```

Extracting Year information to create a Year column, from Created_Date column.

```
df.4<- subset(df.3, select = -c(lowest_monthly_earnings,highest_monthly_earnings,lowest_yearly_earnings,highest_yearly_earni
ngs,Normalised_Income,rank,Youtuber,Title,Abbreviation,video_views_rank,country_rank,channel_type_rank,X2022))
df.4 <- df.4 %>% dplyr::mutate (year = lubridate::year(df.4$created_date_1),month = lubridate::month(df.4$created_date_1),da
y = lubridate::day(df.4$created_date_1))
df.4 <- subset(df.4, select = -c(created_date_1, month, day))
#colnames(df.4)
```

1.6 Filling the missing values for column: subscribers_for_last_30_days.

Upon checking the data :

- There are still 284 rows with NAs in **subscribers_for_last_30_days** column
- Decided to use **video_views_for_the_last_30_days** to fill in the N/A values.

```
na_index <- which(is.na(df.4$subscribers_for_last_30_days))
ratio_vc <- df.4$video_views_for_the_last_30_days[-na_index] / df.4$subscribers_for_last_30_days[-na_index]
P.3 <- ggplot() + aes(ratio_vc)+ geom_histogram( colour="midnightblue", fill="#00A08A") + labs(title = "Histogram of Ratio, between 'video_views_for_the_last \n _30_days' & 'subscribers_for_last_30_days' ",caption = "Fig 2.1") + theme( plot.title = element_text(size=9, face = "bold"),axis.text=element_text(size=8),axis.title=element_blank())

P.4 <- ggplot() + aes(ratio_vc)+ geom_histogram( colour="midnightblue", fill="#00A08A") +  xlim(c(0, 2500))+ labs(title = "Same histogram as left. \n with outliers excluded ",caption = "Fig 2.2") + theme( plot.title = element_text(size=9, face = "bold"),axis.text=element_text(size=8),axis.title=element_blank())

grid.arrange (P.3,P.4, ncol = 2)
```

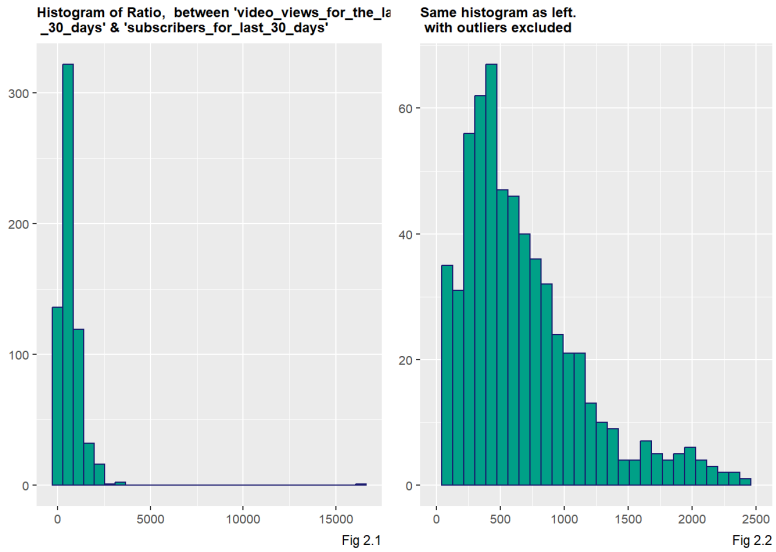


Fig 2.1 Fig 2.2

- Above Fig 2.1 & Fig 2.2, are the histogram of **video_views_for_the_last_30_days / subscribers_for_last_30_days ratio**
- Apart from some extreme outliers, the ratios are concentrated around 500 range.
- Using median ratio to fill in the missing values in **subscribers_for_last_30_days ratio**

```
median_ratio <- median(ratio_vc)
df.4 <- df.4[-which(is.na(df.4$video.views)),]
df.4$subscribers_for_last_30_days_no_na <- 0
df.4<- df.4 %>% mutate(subscribers_for_last_30_days_no_na = if_else(is.na(subscribers_for_last_30_days), video_views_for_the_last_30_days/median_ratio, subscribers_for_last_30_days))
df.4$income <- as.factor(df.4$income)
```

2. Feature Selection

- Training/Testing data splitting, Data type conversion, Building Single Variable Model

```
set.seed(12345)
intrain <- runif(nrow(df.4)) < 0.8
df.train <- df.4[intrain,]
df.test <-df.4[!intrain,]
#df.train
#df.test
```

```
vars <- setdiff(colnames(df.4), c('income'))
catVars <- vars[sapply(df.train[,vars],class) %in% c('factor','character')]
numericVars <- vars[sapply(df.train[,vars],class) %in% c('numeric','integer')]

catVars
```

```
## [1] "category"     "Country"      "channel_type" "continent"
```

```
numericVars
```

```
## [1] "subscribers"                      "video.views"
## [3] "uploads"                          "video_views_for_the_last_30_days"
## [5] "subscribers_for_last_30_days"     "year"
## [7] "subscribers_for_last_30_days_no_na"
```

2.1 Single Variable Model.

Below code create **pred_variable** columns for each column, in order to :

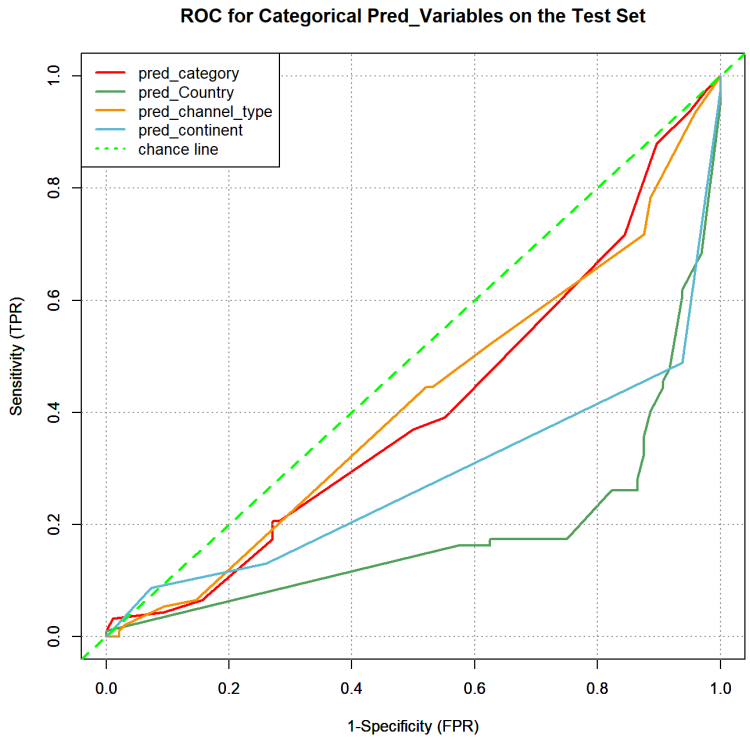
- Transform all categorical variables into numeric variables, with extra **probability** information added.
- Group all numeric variables into pred_numeric variables, in order to normalize the data, which were heavily skewed, which have revealed by the EDA analysis.
- Plot all receiver operating characteristic curve on the same plot to compare the performance.
- The code were adapted from lecture slides.

```
plot_roc <- function(predcol, outcol, colour_id = 2, overlaid =F ) {
  ROCit_obj <- rocit (score = predcol, class = outcol == "1")
  par(new = overlaid ,  cex = 0.85)
  plot (ROCit_obj, col = c(colour_id,"green"), legend = FALSE, YIndex = FALSE, values = FALSE)
}
```

```
pos = 1
mkPredC <- function(outCol, varCol, appCol) {
pPos <- sum(outCol == pos) / length(outCol)
naTab <- table(as.factor(outCol[is.na(varCol)]))
pPosWna <- (naTab/sum(naTab))[pos]
vTab <- table(as.factor(outCol), varCol)
pPosWv <- (vTab[pos, ] + 1.0e-3*pPos) / (colSums(vTab) + 1.0e-3)
pred <- pPosWv[appCol]
pred[is.na(appCol)] <- pPosWna
pred[is.na(pred)] <- pPos
pred
}


for(v in catVars) {
pi <- paste('pred_', v, sep='')
df.train[,pi] <- mkPredC(df.train[,'income'], df.train[,v], df.train[,v])
df.test[,pi] <- mkPredC(df.train[,'income'], df.train[,v], df.test[,v])
}

calcAUC <- function(predcol,outcol) {
perf <- performance(prediction(predcol,outcol==pos),'auc')
as.numeric(perf@y.values)
}


for(v in catVars) {
pi <- paste('pred_', v, sep='')
aucTrain <- calcAUC(df.train[,pi], df.train[,'income'])
if (aucTrain >= 0.1) {
aucCal <- calcAUC(df.test[,pi], df.test[,'income'])
print(sprintf(
"%s: trainAUC: %4.3f; calibrationAUC: %4.3f",
pi, aucTrain, aucCal))
}
}
```

```
## [1] "pred_category: trainAUC: 0.376; calibrationAUC: 0.407"
## [1] "pred_Country: trainAUC: 0.130; calibrationAUC: 0.185"
## [1] "pred_channel_type: trainAUC: 0.396; calibrationAUC: 0.417"
## [1] "pred_continent: trainAUC: 0.267; calibrationAUC: 0.279"
```

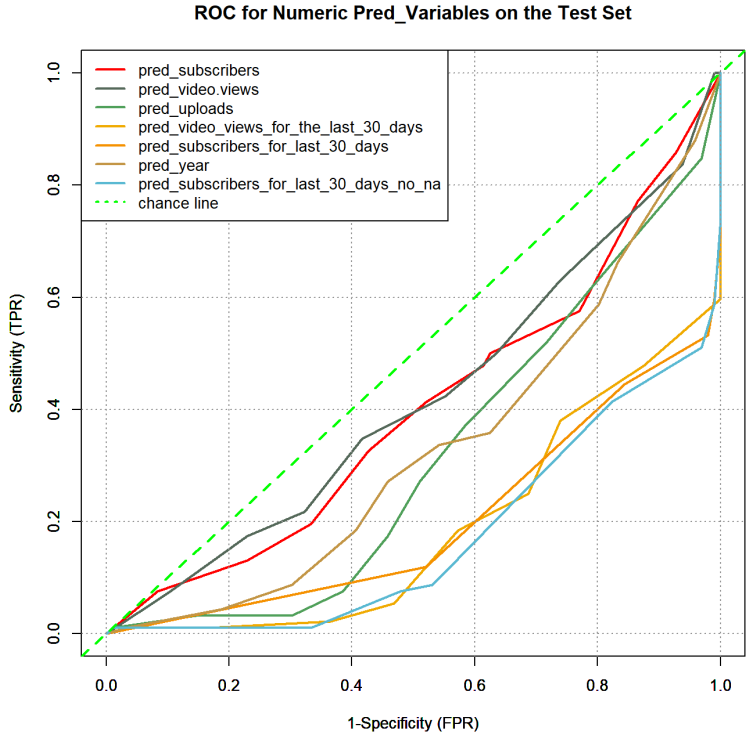
```
wes<- wes_palette(length(catVars), name = "Darjeeling1", type = "continuous")


n=0
pi.list<- ""
colour.list <- ""
for(v in catVars){
pi <- paste('pred_', v, sep='')
n = n+1
plot_roc (df.test[,pi],df.test$income, colour_id = wes[n], overlaid = T)
pi.list <- c(pi.list, pi)
colour.list <- c(colour.list, wes[n])
}
legend(x = "topleft", legend=c(pi.list[2:length(pi.list)], "chance line"),
       lty = c(rep(1, length(pi.list)-1), 3), lwd = 2,  col = c(colour.list[2:length(pi.list)], "green"), text.font = 1)
title("ROC for Categorical Pred_Variables on the Test Set")
```




```
mkPredN <- function(outCol, varCol, appCol) {
cuts <- unique(
quantile(varCol, probs=seq(0, 1, 0.1), na.rm=T))
varC <- cut(varCol,cuts)
appC <- cut(appCol,cuts)
mkPredC(outCol,varC,appC)
}


for(v in numericVars) {
pi <- paste('pred_', v, sep='')
df.train[,pi] <- mkPredN(df.train[,'income'], df.train[,v], df.train[,v])
df.test[,pi] <- mkPredN(df.train[,'income'], df.train[,v], df.test[,v])
aucTrain <- calcAUC(df.train[,pi], df.train[,'income'])
if(aucTrain >= 0.1) {
aucCal <- calcAUC(df.test[,pi], df.test[,'income'])
print(sprintf(
"%s: trainAUC: %4.3f; calibrationAUC: %4.3f",
pi, aucTrain, aucCal))
}
}
```

```
## [1] "pred_subscribers: trainAUC: 0.383; calibrationAUC: 0.402"
## [1] "pred_video.views: trainAUC: 0.342; calibrationAUC: 0.418"
## [1] "pred_uploads: trainAUC: 0.298; calibrationAUC: 0.317"
## [1] "pred_video_views_for_the_last_30_days: trainAUC: 0.149; calibrationAUC: 0.188"
## [1] "pred_subscribers_for_last_30_days: trainAUC: 0.206; calibrationAUC: 0.201"
## [1] "pred_year: trainAUC: 0.374; calibrationAUC: 0.331"
## [1] "pred_subscribers_for_last_30_days_no_na: trainAUC: 0.178; calibrationAUC: 0.173"
```

```
wes<- wes_palette(length(numericVars), name = "Darjeeling1", type = "continuous")


n=0
pi.list<- ""
colour.list <- ""
for(v in numericVars){
pi <- paste('pred_', v, sep='')
n = n+1
plot_roc (df.test[,pi],df.test$income, colour_id = wes[n], overlaid = T)
pi.list <- c(pi.list, pi)
colour.list <- c(colour.list, wes[n])
}
legend(x = "topleft", legend=c(pi.list[2:length(pi.list)], "chance line"),
       lty = c(rep(1, length(pi.list)-1), 3), lwd = 2,  col = c(colour.list[2:length(pi.list)], "green"), text.font = 1)
title("ROC for Numeric Pred_Variables on the Test Set")
```



From above two ROC plots, all the single variable models by using **pred_variables** are performing poorly on the testing dataset, worse than random guess. This may also due to the fact that the target variable is balanced, which would require more features rather than single variables to predict accurately.

2.2 Building Null Model.

```
(Npos <- sum(df.train[,'income'] == 1))
```

```
## [1] 365
```

```
pred.Null <- Npos / nrow(df.train)
cat("Proportion of outcome == 1 in df.train:", pred.Null)
```

```
## Proportion of outcome == 1 in df.train: 0.5034483
```

2.3 Log Likelihood / Deviance

```
logLikelyhood <- function(ytrue, ypred, epsilon = 1e-6) {
  sum(ifelse(ytrue==1, log(ypred+epsilon), log(1-ypred-epsilon)),na.rm = T)
}



logNull <- logLikelyhood (df.train[,'income'], sum(df.train[,'income']==1)/nrow(df.train))
logNull
```

```
## [1] -502.5145
```

```
selCatVars <- c()
selPredCatVars <- c()
minDrop<- 0

for(v in catVars) {
  pi <- paste('pred_',v,sep='')
  devDrop <- logLikelyhood(df.train[,'income'],df.train[,pi])
 # if(devDrop>=minDrop) {
    print(sprintf("%s, likelihood: %g",pi,devDrop))

 # }
}
```

```
## [1] "pred_category, likelihood: -590.204"
## [1] "pred_Country, likelihood: -1560.58"
## [1] "pred_channel_type, likelihood: -642.439"
## [1] "pred_continent, likelihood: -819.075"
```

```
selCatVars <- c()
selPredCatVars <- c()
minDrop<- 0

for(v in catVars) {
  pi <- paste('pred_',v,sep='')
  devDrop <- 2*(logLikelyhood(df.train[,'income'],df.train[,pi]) - logNull)
 # if(devDrop>=minDrop) {
    print(sprintf("%s, deviance reduction: %g",pi,devDrop))
    selCatVars <- c(selCatVars, v)
    selPredCatVars <- c(selPredCatVars,pi)

 # }
}
```

```
## [1] "pred_category, deviance reduction: -175.379"
## [1] "pred_Country, deviance reduction: -2116.14"
## [1] "pred_channel_type, deviance reduction: -279.85"
## [1] "pred_continent, deviance reduction: -633.122"
```

```
selNumVars <- c()
selPredNumVars <- c()
#minDrop<- 200

for(v in numericVars) {
  pi <- paste('pred_',v,sep='')
  devDrop <- 2*(logLikelyhood(df.train[,'income'],df.train[,pi]) - logNull)
 # if(devDrop>minDrop) {
    print(sprintf("%s, deviance reduction: %g",pi,devDrop))
    selNumVars <- c(selNumVars, v)
    selPredNumVars <- c(selPredNumVars,pi)
  #}
}
```

```
## [1] "pred_subscribers, deviance reduction: -100.858"
## [1] "pred_video.views, deviance reduction: -223.035"
## [1] "pred_uploads, deviance reduction: -359.678"
## [1] "pred_video_views_for_the_last_30_days, deviance reduction: -4926.46"
## [1] "pred_subscribers_for_last_30_days, deviance reduction: -900.346"
## [1] "pred_year, deviance reduction: -126.189"
## [1] "pred_subscribers_for_last_30_days_no_na, deviance reduction: -2197.57"
```

Conclusion :

- Above drop deviance test shows negative results, which matching the results of ROC curve.
- In comparison, the current Null Model is a good model which yields the smallest deviance, and largest log likelihood., as the target variable is perfectly balanced.

2.4 Chi-square Test

- All the categorical variable, along with “pred_variables” were used for Chi-Square Test.
- The Chi-Square Test is for testing the independence between categorical variables, however it will also automatically treat each different numeric value as a new ‘categorical level’ for testing as well. Therefore the **Pred_Numerical_Variables** can be used for testing as each column only has 10 unique values / Probabilities

```
variable <- c()
chi_Statistic <- c()
pred_numericVars <- paste('pred_', numericVars, sep='')
pred_catVars <- paste('pred_', catVars, sep='')


for(v in c(catVars,pred_catVars,pred_numericVars)) {
  test<-chisq.test(df.train[,v], df.train[,'income'],simulate.p.value = TRUE)
  variable <- c(variable, v)
  chi_Statistic<- c(chi_Statistic,test$statistic)
}

chi.df <-data.frame(variable, chi_Statistic)
kbl(chi.df[order(-chi.df$chi_Statistic),],longtable = F, booktabs = TRUE,  digits = 0,format.args = list(big.mark = ","),cap
tion = "**Table B: Chi-Square Test Results**  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")%>%
  row_spec(c(1:7), background = "#78B7C5")%>%
  scroll_box(width = "80%", height = "380px")
```

Table B: Chi-Square Test Results

	variable	chi_Statistic
2	Country	331
6	pred_Country	331
12	pred_video_views_for_the_last_30_days	287
15	pred_subscribers_for_last_30_days_no_na	238
13	pred_subscribers_for_last_30_days	204
4	continent	169

	variable	chi_Statistic
8	pred_continent	169

Conclusion:

- Result above shows **Country**, **pred_video_views_for_the_last_30_days**, **pred_subscribers_for_last_30_days_no_na**, **continent** are having the highest Chi-Statistics, which means these variables could be **important features**, as they are not independent with target variable, and some sort of relationship exist.
- Caveat : however, chi-square test are sensitive to the sample size, there are situations, especially in Country, only one observation in the dataset, such as **cuba / Afghanistan / Andorra**, these would violate the chi-square assumption where **all expected number should all be over 5**.
- When the expected cell counts are lower than 5, it is recommended to use Fisher's Exact Test instead.

2.4 Fisher's Exact Test

A guideline when to use Fisher's Exact Test : https://statisticsbyjim.com/hypothesis-testing/fishers-exact-test/ (https://statisticsbyjim.com/hypothesis-testing/fishers-exact-test/)

- Cell counts are smaller than 20
- A cell has an expected value 5 or less.
- The column or row marginal values are extremely uneven.

```
variable <- c()
Statistic <- c()
table_result <- c()
for(v in c(catVars)) {
  table_test <- table(df.train[,v], df.train[,'income'])
  test<-fisher.test(table_test,alternative = 'two.sided',simulate.p.value=TRUE)
  table_result <- rbind(table_result,table_test)
  variable <- c(variable, v)
  Statistic<- c(Statistic,test)
}

fish.df <-data.frame(variable, Statistic)


kbl(table_result,longtable = F, booktabs = TRUE,  digits = 0,format.args = list(big.mark = ","),caption = "**Table C: Contin
gency Tabels for Categorical Variables, 0 : Low Income, 1 : High Income**  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")%>%
  row_spec(c(1:16), background = "#78B7C5")%>%
  row_spec(c(17:63), background = "#74A089")%>%
  row_spec(c(64:77), background = "lightgrey") %>%
  row_spec(c(78:83), background = "#DC5C5C") %>%
  scroll_box(width = "50%", height = "280px")
```

Table C: Contingency Tabels for Categorical Variables, 0 : Low Income, 1 : High Income

	0	1
Autos & Vehicles	0	2
Comedy	17	31
Education	14	20
Entertainment	74	87
Film & Animation	11	21

Conclusion:

- Fisher Exact Test shows all categorical variables are not independent from the target income high/low variables.
- Above **Table C** is the summary of contingency table for references.

The rows are color coded as below:

- **Blue - Category**
- **Green - Country**
- **Grey - Channel Type**
- **Red - Continent**

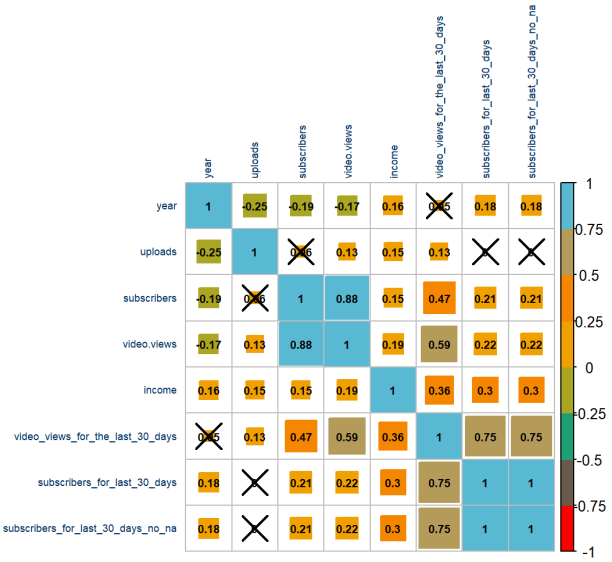
2.5 Correlation

Above tests were mainly used for categorical variables, correlation matrix can be used for numeric variables.

Below are the two correlation map.

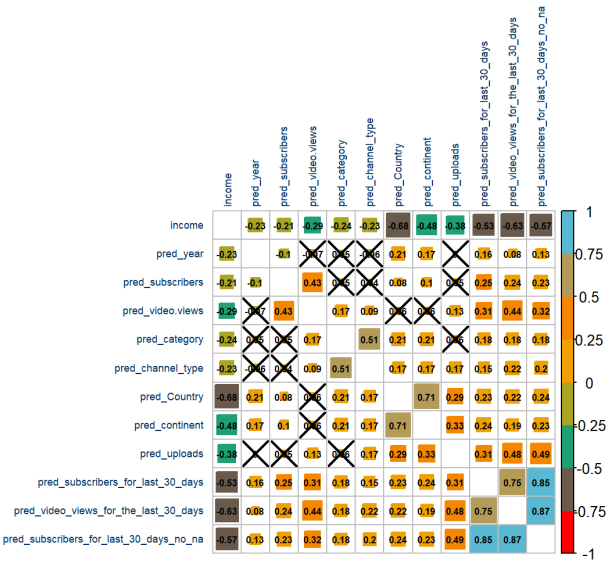
1. **Correlation between income and numerical variables**

```
numeric_subset <- df.train[, which(names(df.train) %in% c(numericVars,'income'))]
numeric_subset$income <- as.numeric(numeric_subset$income)
numeric_subset <- na.omit (numeric_subset)

M_1 <- cor(numeric_subset)
test_Res_1 = cor.mtest(numeric_subset, conf.level = 0.95)

corrplot( M_1, p.mat = test_Res_1$p, method = 'square', tl.col="#003366", col=wes_palette(8, name = "Darjeeling1", type = "c
ontinuous"), insig='pch', addCoef.col ='black', tl.cex=0.55, number.cex = 0.6,number.font = 2, order = 'hclust',diag=TRUE)
```



2. **Correlation between income and Pred_variables**

```
allpredvar <- paste('pred_',vars,sep='')
numeric_subset_2 <- df.train[, which(names(df.train) %in% c(allpredvar,'income'))]
numeric_subset_2$income <- as.numeric(numeric_subset_2$income)
numeric_subset_2 <- na.omit (numeric_subset_2)

M_1 <- cor(numeric_subset_2)
test_Res_1 = cor.mtest(numeric_subset_2, conf.level = 0.95)

corrplot( M_1, p.mat = test_Res_1$p, method = 'square', tl.col="#003366", col=wes_palette(8, name = "Darjeeling1", type = "c
ontinuous"), insig='pch', addCoef.col ='black',  tl.cex=0.55, number.cex = 0.5,number.font = 2, order = 'hclust',diag=FALSE)
```



Conclusion :

From above Correlation matrices, below are the variables may be significant features.

- Video_views_for_the_last_30_days
- Subscribers_for_last_30_days
- Subscribers_for_last_30_days_no_na
- Pred_Country
- Pred_subscribers_for_last_30_days
- Pred_subscribers_for_last_30_days_no_na
- Pred_video_views_for_the_last_30_days

2.6 Information Gain

By definition, information gain is part of the **Filter Methods** for feature selection, it calculate the reduction in entropy from the dataset.

```
library(FSelectorRcpp)
result <- as.data.frame(information_gain(formula = income~., data = df.test) )
colnames(result) <- c("attributes","importance")
newdata <- result[order(-result$importance),]
kbl(newdata,longtable = F, booktabs = TRUE, format.args = list(big.mark = ","),caption = "**Table D: Information Gain Results**  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")%>%
  row_spec(c(1:7), background = "#78B7C5")%>%
  scroll_box(width = "80%", height = "380px")
```

Table D: Information Gain Results

	attributes	importance
5	Country	0.2971657
7	video_views_for_the_last_30_days	0.2735040
11	subscribers_for_last_30_days_no_na	0.2680452
19	pred_video_views_for_the_last_30_days	0.2341067
22	pred_subscribers_for_last_30_days_no_na	0.2205002
8	subscribers_for_last_30_days	0.2018984
13	pred_Country	0.2001807
20	pred_subscribers_for_last_30_days	0.1604751

Conclusion :

From above Information Gain results, it shows again, below features are possible important features.

- Country
- Video_views_for_the_last_30_days
- Subscribers_for_last_30_days
- Subscribers_for_last_30_days_no_na
- Pred_Country
- Pred_subscribers_for_last_30_days
- Pred_subscribers_for_last_30_days_no_na
- Pred_video_views_for_the_last_30_days

Two sets of selected features.

- From the single variable model AUC / Log likelihood / Deviance : Pred_category, pred_channel_type, pred_subscribers, pred_year
- From Chi-Square, Correlation, Information Gain : Country / Video_views_for_the_last_30_days / Subscribers_for_last_30_days

However above two sets of features are more for the project requirements purpose. Due to the relatively small number of features in the data, and also due to different model may require different combination of features for test. The below model training process will involve a lot more feature combination for the performance results.

Some models also incorporate the cross validation, as well as feature selection process.

3. Modelling

3.1 Penalized Logistic Regression with LASSO penalty

Regularization methods are also called penalization method

F Shofiyah (2018) summarized the logistic regression as “The logistic regression model is a model that describes the relationship between several factors(predictor variables) with dichotomous (binary) response variables”.

In simple words, logit function has been used, log(odds) on the left hand side of the regression formular.

$$\mathbf{Logit}(\pi_i) = \mathbf{Log}(\frac{\pi_i}{1-\pi_i})$$

Usually in logistic regression, **Maximum Likelihood Estimation / MLE** is used for parameter estimation. However there is another method, which will be used here for the parameter estimation : **Least Absolute Shrinkage and Selection Operator (LASSO)**

The LASSO penalty function is :

$$\mathbf{P}(\beta) = \sum_{j=1}^{p} |\beta_j|$$

LASSO can shrink some coefficients towards zero and even exactly zero, in order to perform selection of the predictor variables as well.

The regulation purpose is to balance the accuracy and simplicity of the model, the ideal goal is by using smallest number of features, to return with a good accuracy.

Building Model : model_glmnet

Some code below are adapted from : Link (http://www.sthda.com/english/articles/36-classification-methods-essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/)

```
df.train$income <- as.factor(df.train$income)

x<- model.matrix (income ~ pred_category + pred_Country + pred_channel_type + pred_continent + pred_year + pred_subscribers
+ pred_video.views + pred_uploads+pred_video_views_for_the_last_30_days+pred_subscribers_for_last_30_days_no_na,df.train)[,-
1]
y<- df.train$income


cv.lasso <- cv.glmnet(x,y,alpha=1, family = 'binomial')

model_glmnet <- glmnet(x,y,alpha=1,family = 'binomial',lambda = cv.lasso$lambda.min)

Predict_Train.1 <- model_glmnet %>% predict(newx = x, type='response')

coef(model_glmnet)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                              s0
## (Intercept)                          18.2934633
## pred_category                        -0.9843484
## pred_Country                        -12.3758000
## pred_channel_type                     .
## pred_continent                        .
## pred_year                            -1.9021369
## pred_subscribers                     -2.4152851
## pred_video.views                     -2.5435309
## pred_uploads                         -0.5357085
## pred_video_views_for_the_last_30_days -13.8521041
## pred_subscribers_for_last_30_days_no_na -1.0704056
```

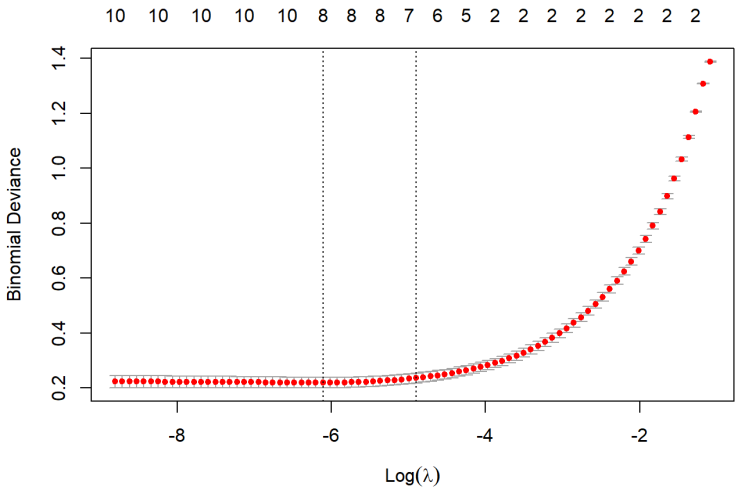
Making Prediction on the Testing Data

```
x.test <- model.matrix(income ~ pred_category+ pred_Country + pred_channel_type + pred_continent + pred_year + pred_subscrib
ers + pred_video.views+pred_uploads+pred_video_views_for_the_last_30_days+pred_subscribers_for_last_30_days_no_na,df.test)[,
-1]
probabilities <- model_glmnet %>% predict(newx = x.test, type='response')
predicted.classes <- ifelse(probabilities > 0.5, "1", "0")
# Model accuracy
observed.classes <- df.test$income


table(observed.classes,predicted.classes)
```

```
##                predicted.classes
## observed.classes  0  1
##                0 91  5
##                1  9 83
```

Find the optimal value of lambda that minimizes the cross-validation error

```
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")
plot(cv.lasso)
```



- Above plot displays the cross-validation error, the above plot shows when **Log(λ)** is approximately -6, it will minimize the prediction error.
- this optimal value is:

```
cv.lasso$lambda.min
```

```
## [1] 0.002222252
```

In oder to reduce the model complexity, cv.glmnet() also finds the λ which will give the simplest model but also lies within 1 standard error of above optimal λ, this best balanced value is :

```
cv.lasso$lambda.1se
```

```
## [1] 0.007448106
```

Below shows the model when λ being replace by the most balance one, rather than most optimal one.

```
model_glmnet.2 <- glmnet(x, y, alpha = 1, family = "binomial",
                    lambda = cv.lasso$lambda.1se)
coef(model_glmnet.2)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                              s0
## (Intercept)                           12.7966952
## pred_category                         -0.1130141
## pred_Country                          -9.9162425
## pred_channel_type                      .
## pred_continent                         .
## pred_year                             -0.9022496
## pred_subscribers                      -1.1268302
## pred_video.views                      -1.2333648
## pred_uploads                           .
## pred_video_views_for_the_last_30_days -11.2431367
## pred_subscribers_for_last_30_days_no_na  -0.4721509
```

- As we can see from above, two more features have now been further eliminated. Left with 6 features in total in the logistic regression.
- Below validated, the prediction is still as good as the first model, when the optimal λ was used.

```
probabilities.2<- model_glmnet.2 %>% predict(newx = x.test, type='response')
predicted.classes.2 <- ifelse(probabilities.2 > 0.5, "pos", "neg")

table(observed.classes,predicted.classes.2)
```

```
##                predicted.classes.2
## observed.classes neg pos
##                0  93   3
##                1   9  83
```

```
performanceMeasures <- function(train_true, train_pred,test_true, test_pred, data.name.1 = "Training",data.name.2 = "Testin
g" , threshold=0.5) {

cmat <- table(actual = train_true, predicted = train_pred >= threshold)
accuracy <- sum(diag(cmat)) / sum(cmat)
precision <- cmat[2, 2] / sum(cmat[, 2])
recall <- cmat[2, 2] / sum(cmat[2, ])
f1 <- 2 * precision * recall / (precision + recall)
trainperf_df<-data.frame(model = data.name.1, precision = precision,
recall = recall, f1 = f1)
tmat <- table(actual = test_true, predicted = test_pred >= threshold)
accuracy <- sum(diag(tmat)) / sum(tmat)
precision <- tmat[2, 2] / sum(tmat[, 2])
recall <- tmat[2, 2] / sum(tmat[2, ])
f1 <- 2 * precision * recall / (precision + recall)
testperf_df <- data.frame(model = data.name.2, precision = precision,
recall = recall, f1 = f1)
perftable <- rbind(trainperf_df, testperf_df)
perftable
}
test.1 <- performanceMeasures(  y, Predict_Train.1,data.name.2 = "Testing ( with Optimal Lambda lambda.min ) " , observed.cl
asses,probabilities)
test.2 <- performanceMeasures(  y, Predict_Train.1,data.name.2 = "Testing ( with Balanced Lambda lambda.1se ) ", observed.cl
asses,probabilities.2)
test.3 <- rbind(test.1, test.2)
```

```
kbl(test.3,longtable = F, booktabs = TRUE, format.args = list(big.mark = ","),caption = "**Table E.2: Penalised Logistic Reg
ression **")%>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")
```

Table E.2: Penalised Logistic Regression

model	precision	recall	f1
Training	0.9666667	0.9534247	0.9600000
Testing (with Optimal Lambda lambda.min)	0.9431818	0.9021739	0.9222222
Training	0.9666667	0.9534247	0.9600000
Testing (with Balanced Lambda lambda.1se)	0.9651163	0.9021739	0.9325843

Conclusion:

- The Penalized Logistic Regression with LASSO penalty has performed quite well in the testing data set.

- The simpler model by using cv.lasso$lambda.1se, rather than using the optimal lambda, is still returning exceptional results on testing dataset.

3.2 Random Forest / Decision Tree

Bagging is an ensemble algorithm that fits multiple models on different subsets of a training dataset, then combines the predictions from all models.

Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample.

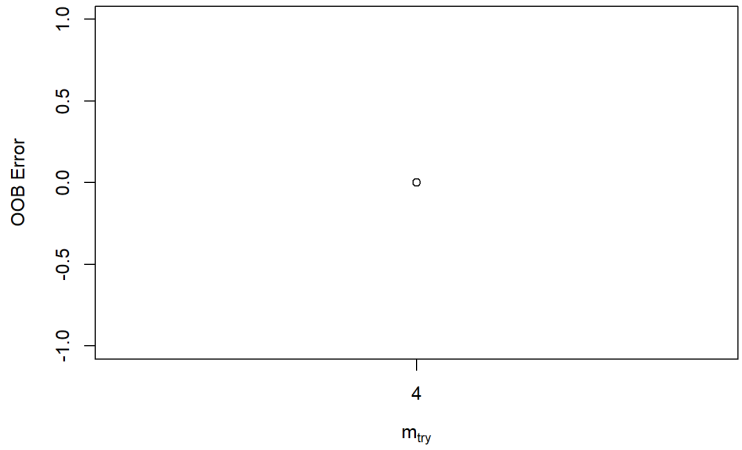
Random forest also have an out-of-bag (OOB) sample that provides a built-in validation set.

- How does TrunRF function work:

tuneRF function based on Rdocument description : Starting with the default value of mtry, search for the optimal value (with respect to Out-of-Bag error estimate) of mtry for randomForest.

There are 22 variables in the first attempt, so the mtry default value would be **sqrt(22)** , 4 or 5 Then mtry default will divided by, or times stepFactor settings **1.1**, to calculate the OBB error , **OBB_Left** ,**OBB_Right**

```
df.train.1 <- subset(df.train, select = -c(subscribers_for_last_30_days))
df.train.1$income <- as.factor(df.train.1$income)
bestmtry <- tuneRF(df.train.1,df.train.1$income,stepFactor = 1.1, improve = 0.01, trace=F, plot= T)
```



As seem from above, the initial Mtry as 4, has OOB Error 0. Therefore it stopped at 4.

- **First attempt with all the original variables** (Formula as below)

```
vars.1 <- c("subscribers","video.views","category","uploads","Country" ,"channel_type","video_views_for_the_last_30_days","c
ontinent","year"
          ,"subscribers_for_last_30_days_no_na")

formula_rf <- paste('income',' ~ ', paste(vars.1, collapse=' + '), sep='')
formula_rf
```

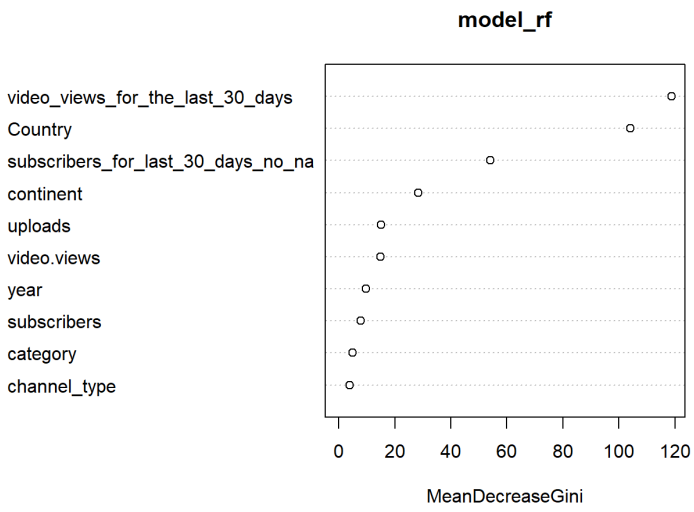
```
## [1] "income ~ subscribers + video.views + category + uploads + Country + channel_type + video_views_for_the_last_30_days
+ continent + year + subscribers_for_last_30_days_no_na"
```

```
formula_rf <- as.formula(formula_rf)
model_rf <- randomForest(formula_rf,data= df.train.1, mtry=4)
model_rf$confusion
```

```
##     0   1 class.error
## 0 339  21  0.05833333
## 1  11 354  0.03013699
```

- Above shows model_rf did a good job on the training dataset.
- Below shows importance of all the variables in Random Forest Model_rf

```
#importance(model_rf)
varImpPlot(model_rf)
```




```
df.test.1 <- subset(df.test, select = -c(subscribers_for_last_30_days))

pred_test <- predict(model_rf, newdata = df.test.1, type= "class")

#pred_test
#df.test.1$income
#head(predict(model_rf, newdata = df.test.1, type = "Prob"))

confusionMatrix(table(pred_test,df.test.1$income))
```

```
## Confusion Matrix and Statistics
##
##
## pred_test  0  1
##         0 49  3
##         1 47 89
##
##                Accuracy : 0.734
##                  95% CI : (0.6648, 0.7957)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : 3.230e-10
##
##                   Kappa : 0.4731
##
##  Mcnemar's Test P-Value : 1.193e-09
##
##             Sensitivity : 0.5104
##             Specificity : 0.9674
##          Pos Pred Value : 0.9423
##          Neg Pred Value : 0.6544
##              Prevalence : 0.5106
##          Detection Rate : 0.2606
##    Detection Prevalence : 0.2766
##       Balanced Accuracy : 0.7389
##
##        'Positive' Class : 0
##
```

- Above shows by using the model_rf, the model performed poorly, especially on negative class, which leads to just around 50% of sensitivity .

- **Discussion on the first rf model**
  - The over fitting issue presents in above experiment, when all variables were used for the model.
- **Other experiments with different combination of the features**
  - By using less variables from above selected features, multiple experiments have been conducted, not all code demonstrated here .

```
vars.2<- c("pred_Country", "pred_video_views_for_the_last_30_days","pred_subscribers","pred_year")
vars.3<- c("continent", "video_views_for_the_last_30_days","subscribers")
formula_rf.2 <- as.formula(paste('income',' ~ ', paste(vars.2, collapse=' + '), sep=''))
formula_rf.3 <- as.formula(paste('income',' ~ ', paste(vars.3, collapse=' + '), sep=''))
model_rf.2 <- randomForest(formula_rf.2,data= df.train.1)
model_rf.3 <- randomForest(formula_rf.3,data= df.train.1)
model_rf.2$confusion
```

```
##     0   1 class.error
## 0 348  12  0.03333333
## 1  19 346  0.05205479
```

```
model_rf.3$confusion
```

```
##     0   1 class.error
## 0 342  18   0.0500000
## 1  66 299   0.1808219
```

- Above training confusion matrix shows reduced performance when the feature numbers are dropped.
- Below test confusion matrix has proven the over-fitting issues have been reduced.

```
pred_test.2 <- predict(model_rf.2, newdata = df.test.1, type= "class")
pred_test.3 <- predict(model_rf.3, newdata = df.test.1, type= "class")

confusionMatrix(table(pred_test.2,df.test.1$income))
```

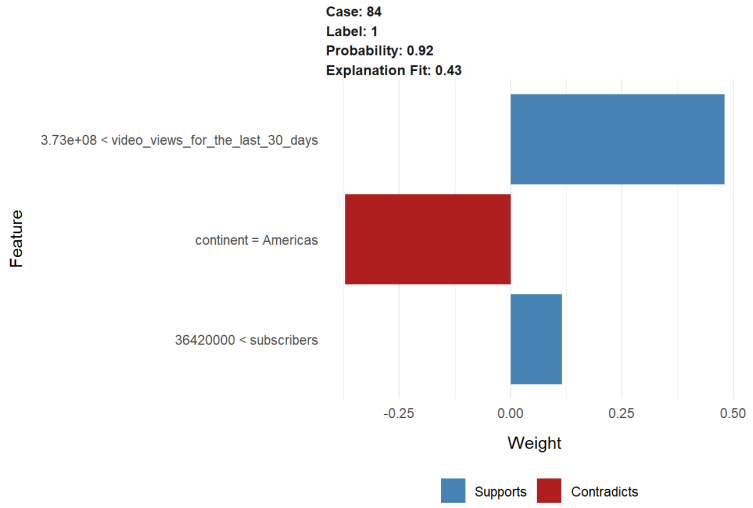
```
## Confusion Matrix and Statistics
##
##
## pred_test.2  0  1
##           0 94  8
##           1  2 84
##
##                Accuracy : 0.9468
##                  95% CI : (0.9044, 0.9742)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8934
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9792
##             Specificity : 0.9130
##          Pos Pred Value : 0.9216
##          Neg Pred Value : 0.9767
##              Prevalence : 0.5106
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5426
##       Balanced Accuracy : 0.9461
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(table(pred_test.3,df.test.1$income))
```

```
## Confusion Matrix and Statistics
##
##
## pred_test.3  0  1
##           0 92 19
##           1  4 73
##
##                Accuracy : 0.8777
##                  95% CI : (0.8221, 0.9208)
##     No Information Rate : 0.5106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7544
##
##  Mcnemar's Test P-Value : 0.003509
##
##             Sensitivity : 0.9583
##             Specificity : 0.7935
##          Pos Pred Value : 0.8288
##          Neg Pred Value : 0.9481
##              Prevalence : 0.5106
##          Detection Rate : 0.4894
##    Detection Prevalence : 0.5904
##       Balanced Accuracy : 0.8759
##
##        'Positive' Class : 0
##
```

- **LIME Ex-plainer, by using the model_rf.3 model**

```
explainer <- lime(df.train[,vars.3], model=as_classifier( model_rf.3),
    bin_continuous=TRUE, n_bins=10)


example.1 <- df.test[15,vars.3]
explanation.1 <- lime::explain(example.1, explainer, n_labels = 1, n_features = 3)
plot_features(explanation.1)
```



Explanation 1:

Above result shows

- **video_views_for_the_last_30_days** was the main strong evidence for income = 1 (high) classification
- **Continent = Americas** contradict the predicted classification.

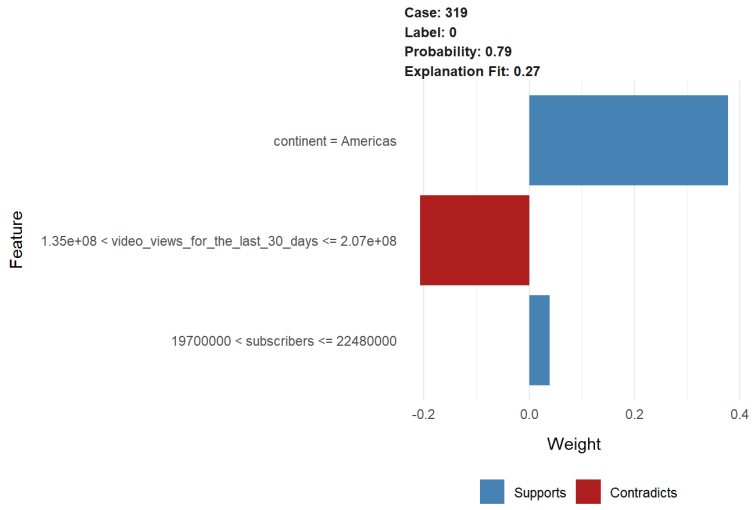
The table below shows the income situation for Continent = Americas, which validates the contradict explaining above.

- If only based on the continent, the predicted class most likely should be 0 (low)

```
table(df.test[df.test$continent == 'Americas', 'income'])
```

```
##
##  0  1
## 65 33
```

```
example.2 <- df.test[69,vars.3]
explanation.2 <- lime::explain(example.2, explainer, n_labels = 1, n_features = 3)
plot_features(explanation.2)
```

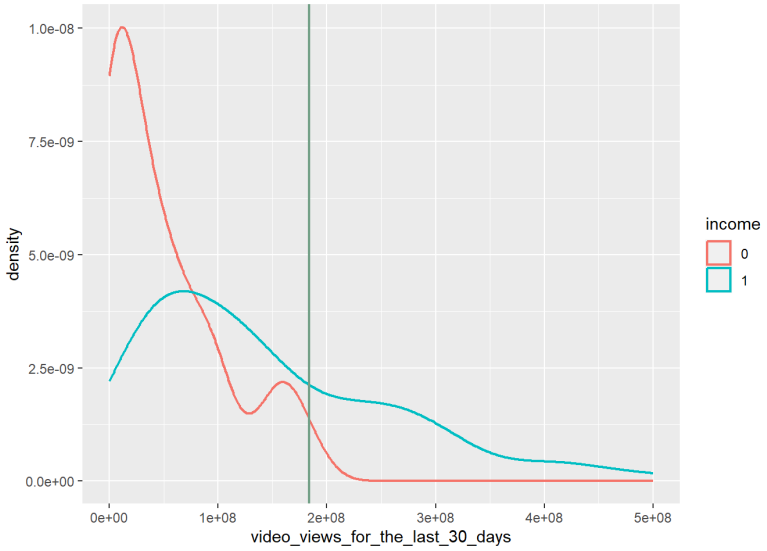


Explanation 2:

Above result shows the different story comparing with explanation 1:

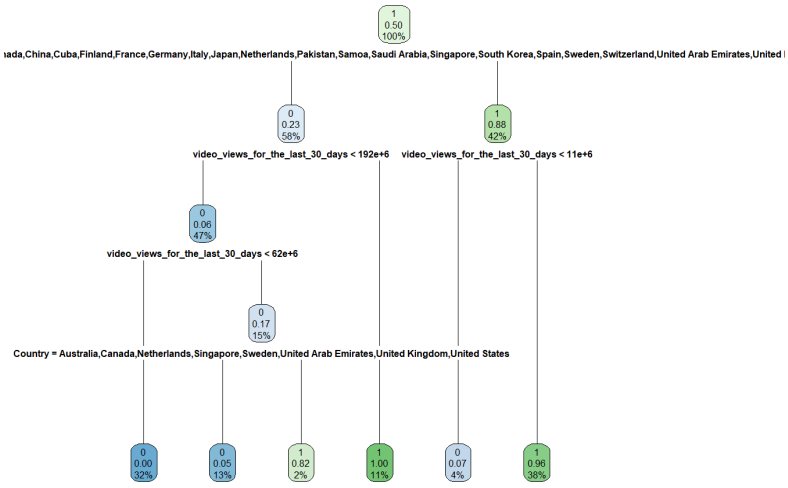
- **Continent = Americas** has become the strong evidence for the classification of income = 0 (low).
- **video_views_for_the_last_30_days** now contradicts the classification
- In the **density plot** below, the green vertical line indicates the case above. If the classification is only based on the **video_views_for_the_last_30_days**, the predicted class most likely should be 1 (High), which match the contradict explanation above.

```
ggplot(df.test)+ geom_density (aes(x=video_views_for_the_last_30_days, color = income), linewidth=0.8) + geom_vline(xinterce
pt = example.2[,'video_views_for_the_last_30_days'], color="#74A089", linewidth=0.8) + xlim(0, 500000000)
```



- **For practice purpose, below is a demo of simple decision tree plot**

```
library(rpart)
dt <- rpart(formula = formula_rf, data = df.train.1)
library(rpart.plot)
rpart.plot(dt)
```




```
#predict(dt, newdata=df.test.1)
```

3.3 RFE : Recursive Feature Elimination / Cross Validation

Recursive feature elimination is a popular feature selection method, it recursively eliminates one feature or small set of feature at a time, by using cross-validation.

In order to better select features for above random forest model, below will use RFE to perform best features,by fitting a random Forrest model within.

Some of code below are adapted from : Link (https://towardsdatascience.com/effective-feature-selection-recursive-feature-elimination-using-r-148ff998e4f7)

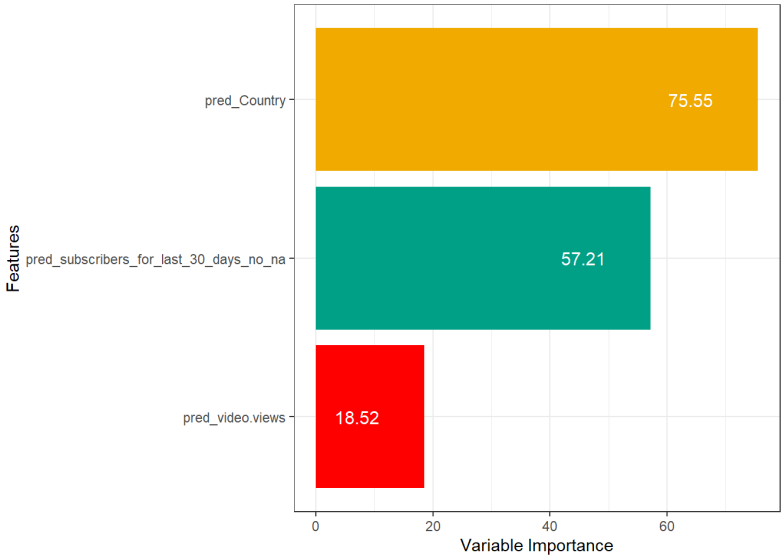
- **rfFuncs** was called to use 'random forest' for importance calculation
- **10-fold cross-validation with 5 repeats** are the settings for cross-validation.

However this time , **the video_view_for_last_30_days** has been removed from the candidates. As it is shown from above models, and also a known knowledge, where **views** is a strong predictor of **earning**, regardless whether it is actual **video viewership**, or **ad viewership**.

```
rfe_train <- df.train %>% select (c(pred_catVars, pred_numericVars))  %>% select(-c(pred_subscribers_for_last_30_days,pred_v
ideo_views_for_the_last_30_days))
rfe_train.1 <- df.train %>% select (c(catVars, numericVars))  %>% select(-c(subscribers_for_last_30_days,video_views_for_the
_last_30_days))

rfe_income <- as.factor(df.train$income)
rfe_train<-  mutate_if(rfe_train, is.character, as.factor)
#summary(rfe_train)
#Length(rfe_income)

control <- rfeControl(functions = rfFuncs,
                      method = "repeatedcv",
                      repeats = 5,
                      number = 10)

result_rfe1 <- rfe(x = rfe_train,
                   y = rfe_income,
                   sizes = c(1:ncol(rfe_train)),
                   rfeControl = control)

result_rfe1.1 <- rfe(x = rfe_train.1,
                   y = rfe_income,
                   sizes = c(1:ncol(rfe_train.1)),
                   rfeControl = control)
```

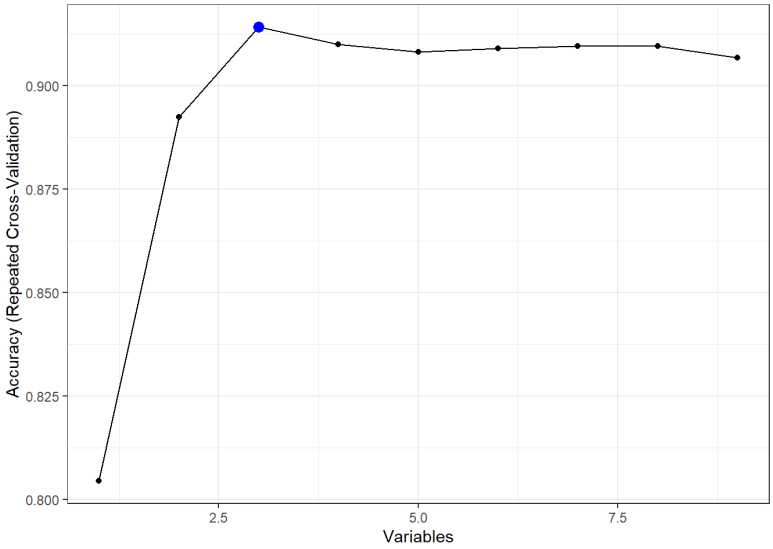
- RFE has been performed twice in above code. one for all the original variables, another one for pre_variables. **the video_view_for_last_30_days** has been excluded from RFE.
- For pred_variables, the recommended variables are :

```
#pander(predictors(result_rfe1))

#kbl(varImp(result_rfe1))



varimp_data <- data.frame(feature = row.names(varImp(result_rfe1)),
                          importance = varImp(result_rfe1)[, 1])

ggplot(data = varimp_data,
       aes(x = reorder(feature, importance), y = importance, fill = feature)) +
  geom_bar(stat="identity") + labs(x = "Features", y = "Variable Importance") +
  geom_text(aes(label = round(importance, 2)), hjust=2, color="white", size=4) +
  theme_bw() + theme(legend.position = "none")+ coord_flip()   + scale_fill_manual(values = rev(wes_palette(length(varimp_da
ta$feature), name = "Darjeeling1", type = "discrete")), name = "")
```



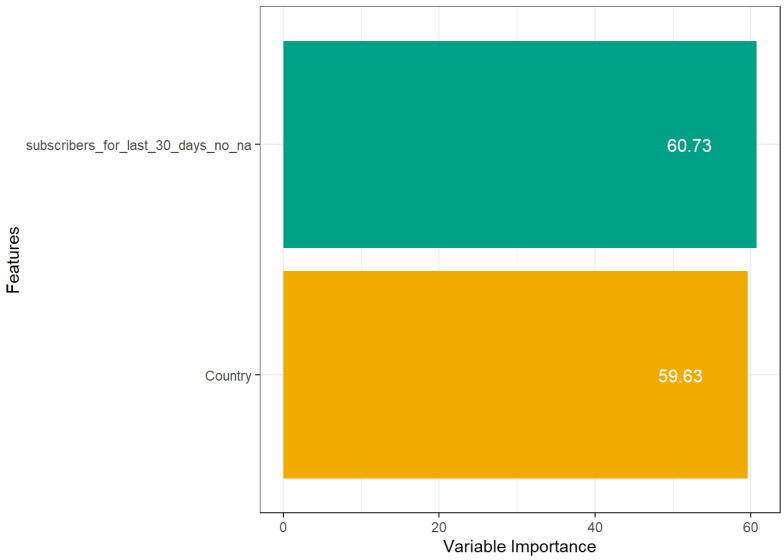
- These three variables reached the best performance (Accuracy over 0.95) as the plot shows below.

```
ggplot(data = result_rfe1, metric = "Accuracy") + theme_bw()
```



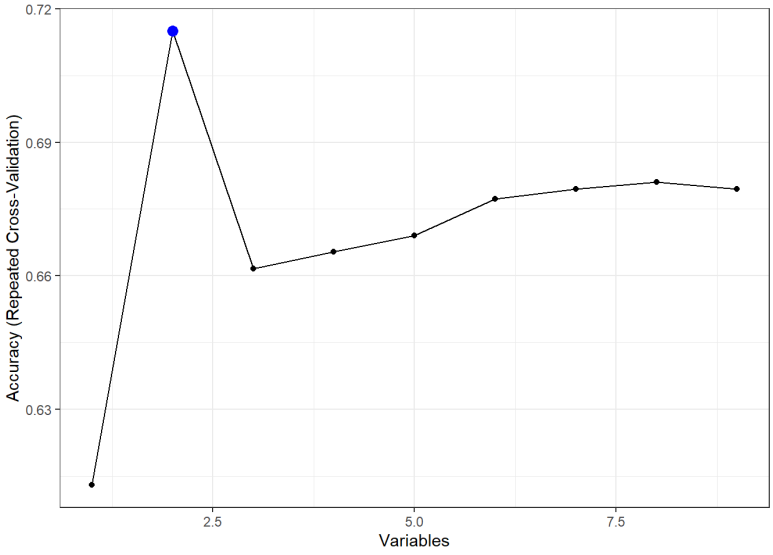
- For original variables, the recommended variables are :

```
#pander(predictors(result_rfe1.1))


varimp_data.1 <- data.frame(feature = row.names(varImp(result_rfe1.1)),
                          importance = varImp(result_rfe1.1)[, 1])

ggplot(data = varimp_data.1,
       aes(x = reorder(feature, importance), y = importance, fill = feature)) +
  geom_bar(stat="identity") + labs(x = "Features", y = "Variable Importance") +
  geom_text(aes(label = round(importance, 2)), hjust=2, color="white", size=4) +
  theme_bw() + theme(legend.position = "none")+ coord_flip()   + scale_fill_manual(values = rev(wes_palette(length(varimp_da
ta$feature), name = "Darjeeling1", type = "discrete")), name = "")
```



- These two variables reached the best performance (Accuracy over 0.69) as the plot shows below.

```
ggplot(data = result_rfe1.1, metric = "Accuracy") + theme_bw()
```



- Using the first 3 pred_variables recommended in the first REF cross validation feature selection process, below is the model result on the testing data.

```
rfe_test <- df.test %>% select (c("pred_Country","pred_subscribers_for_last_30_days_no_na","pred_video.views"))
rfe_test_income <- as.factor(df.test$income)
rfe_test<-  mutate_if(rfe_test, is.character, as.factor)

postResample(predict(result_rfe1, rfe_test), rfe_test_income)
```

```
##  Accuracy     Kappa
## 0.8829787 0.7655329
```

```
ref_predict <- predict(result_rfe1, rfe_test)
kbl(ref_predict,longtable = F, booktabs = TRUE, format.args = list(big.mark = ","),caption = "**Table F: Predicted Result on
testing data, with predicted probablity for each class,  by using RFE**  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A") %>%
  scroll_box(width = "50%", height = "480px")
```

Table F: Predicted Result on testing data, with predicted probablity for each class, by using RFE

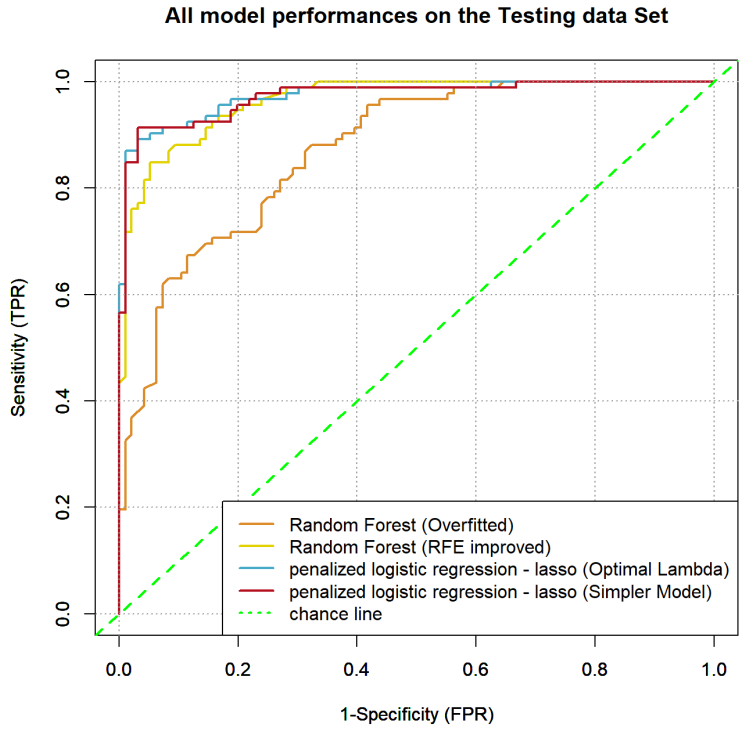
	pred	0	1
3	0	0.836	0.164
5	1	0.000	1.000
12	1	0.038	0.962
25	1	0.000	1.000
28	0	0.672	0.328
42	1	0.020	0.980
43	1	0.476	0.524
48	1	0.000	1.000
56	1	0.012	0.988
57	1	0.018	0.982

```
performanceMeasures.1 <- function(train_true, train_pred,test_true, test_pred, data.name.1 = "Original Random Forest (with o
verfitting issue)",data.name.2 = "New Random Forest (with REF recommended variables)" ) {

cmat <- table(actual = train_true, predicted = train_pred )
accuracy <- sum(diag(cmat)) / sum(cmat)
precision <- cmat[2, 2] / sum(cmat[, 2])
recall <- cmat[2, 2] / sum(cmat[2, ])
f1 <- 2 * precision * recall / (precision + recall)
trainperf_df<-data.frame(model = data.name.1, precision = precision,
recall = recall, f1 = f1)
tmat <- table(actual = test_true, predicted = test_pred)
accuracy <- sum(diag(tmat)) / sum(tmat)
precision <- tmat[2, 2] / sum(tmat[, 2])
recall <- tmat[2, 2] / sum(tmat[2, ])
f1 <- 2 * precision * recall / (precision + recall)
testperf_df <- data.frame(model = data.name.2, precision = precision,
recall = recall, f1 = f1)
perftable <- rbind(trainperf_df, testperf_df)
perftable
}


test.final <- performanceMeasures.1(  df.test.1$income, pred_test, df.test$income,ifelse(predict(result_rfe1, rfe_test)
[,"1"]>0.5,1,0))

test.final <- as.data.frame(test.final)
kbl(test.final,longtable = F, booktabs = TRUE, format.args = list(big.mark = ","),caption = "**Table G: Random Forrest on Te
sting Data **  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")
```

Table G: Random Forrest on Testing Data

model	precision	recall	f1
Original Random Forest (with overfitting issue)	0.6544118	0.9673913	0.7807018
New Random Forest (with REF recommended variables)	0.9069767	0.8478261	0.8764045

```
wes<- wes_palette(5, name = "FantasticFox1", type = "discrete")

calcAUC <- function(predcol, outcol) {
  perf <- performance(prediction(predcol, outcol == '1'),'auc')
  as.numeric(perf@y.values)
}


pred_rf<-predict(model_rf, newdata = df.test.1, type = "Prob")


plot_roc <- function(predcol, outcol, colour_id = 2, overlaid =F ) {
  ROCit_obj <- rocit (score = predcol, class = outcol == "1")
  par(new = overlaid)
  plot (ROCit_obj, col = c(colour_id,"green"), legend = FALSE, YIndex = FALSE, values = FALSE)
}


pred_glmnet <- model_glmnet %>% predict(newx = x.test, type='response')
pred_glmnet.2 <- model_glmnet.2 %>% predict(newx = x.test, type='response')

plot_roc (pred_rf[,"1"],df.test.1$income,colour_id = wes[1])  #original random forest
plot_roc (ref_predict[,"1"],df.test$income,colour_id = wes[2], overlaid = T) #updated random forest after RFE
plot_roc (pred_glmnet[,1],df.test$income,colour_id = wes[3], overlaid = T) #penalized lasso regression with lambda$min
plot_roc (pred_glmnet.2[,1],df.test$income,colour_id = wes[5], overlaid = T) #penalized lasso regression with lambda$1se


legend(x = "bottomright", legend=c("Random Forest (Overfitted)","Random Forest (RFE improved)","penalized logistic regressio
n - lasso (Optimal Lambda)","penalized logistic regression - lasso (Simpler Model)", "chance line"),
       lty = c(rep(1,4), 3), lwd = 2,  col = c(wes[1],wes[2],wes[3],wes[5], "green"), text.font = 1)
title("All model performances on the Testing data Set")
```



3.4 Supervised Modeling Summary

- As shown above in the ROC plot, it is proven that with careful data transformation and feature selections, both logistic regression model and random forest model can be tuned to perform quite well on the testing data:
  - Random Forest: The REF, with built in **cross validations** functionality, has proven to be extrememly useful to improve the orginal Random forest model.
  - After removing **the video_view_for_last_30_days** from the feature options, the **subscribers_for_last_30_days** has become the most important predictor for the model.
  - The orginal rf model, performed well on the training data, but not the testing data, due to **over fitting** issues.
  - In logistic regression, **Maximum Likelihood Estimation / MLE** was not used for parameter estimation, instead, the **Least Absolute Shrinkage and Selection Operator (LASSO)** parameter estimation was used.
  - By not using the Optimal Lambda, and using the Lambda within 1se, it has simplified model by panelised / dropping out 2 more parameters, but still maintained the model performance.

3.5 Supervised Modeling Discussion

- Apart from building models with good performance on the training data, there are still many areas worth considered, and discussed :
  - Below table has listed all the levels in categorical variables, with artifacts of two few observations, and no observation in one of the target classes. These levels may not be good predictors for any generalisations, e.g if Youtuber county is Bangladesh, or channel type is Nonprofits/Activism, there will be 100% of chances of being in the low income group.
  - In order to deal with the **Complete/Perfect Separation** issues in the dataset, the most recommended approach is using **regularization / continuity corrections**, which was used above in the logistic regression model.
  - Due to the nature of our data structure, ordered by top 900s YouTuber by the number of subscribers. The model may have limited predicting power if applied on random selected youtube sampels. For example: the nature, and characteristics of world's top 1000 richest individuals maybe fundenmentally different to the general public.
  - As described in literature reviews, **youtube ad viewships, membership type, number of likes, video duration**, could all be useful features to determine the popularities of YouTuber. These information were not included in our dataset.

```
table_result.1 <- as.data.frame(table_result)
colnames(table_result.1) = c("Low_Income","High_Income" )


kbl(table_result.1 [table_result.1$Low_Income == 0 | table_result.1$High_Income == 0,],longtable = F, booktabs = TRUE,  digi
ts = 0,format.args = list(big.mark = ","),caption = "**Contingency Tabels for Categorical Variables, with complete separatio
n**  ") %>%
  kable_classic(full_width = T, html_font = "Cambria") %>%
  kable_styling(latex_options = c( "hold_position","striped","scale_down"),font_size = 12)%>%
  row_spec(0,background="#EBCC2A")%>%
  row_spec(c(1:2), background = "#78B7C5")%>%
  row_spec(c(3:25), background = "#74A089")%>%
  row_spec(c(26:28), background = "lightgrey") %>%
  row_spec(c(29), background = "#DC5C5C") %>%
  scroll_box(width = "50%", height = "480px")
```

Contingency Tabels for Categorical Variables, with complete separation

	Low_Income	High_Income
Autos...Vehicles	0	2

	Low_Income	High_Income
Nonprofits...Activism	1	0
Andorra	1	0
Bangladesh	1	0
Barbados	0	1
China	1	0
Colombia	0	10
Cuba	1	0

```
calcAUC(pred_rf[,"1"], df.test.1$income)
```

```
## [1] 0.8724524
```

```
calcAUC(pred_glmnet[,1], df.test$income)
```

```
## [1] 0.9737319
```

```
calcAUC(pred_glmnet.2[,1], df.test$income)
```

```
## [1] 0.9723732
```

```
calcAUC(ref_predict[,"1"], df.test$income)
```

```
## [1] 0.9650702
```

4. Clustering

For the below clustering part, only the numerical variables were selected for clustering unsupervised model.

4.1 Data Preparation

Before performing clustering analysis, the data needs to be :

- **Scaling** : the data need to be scaled, centering the data with mean of 0 for all the variables
- **Principal Component Analysis (PCA)** to be conducted, in order to transform high dimensional data into two dimension for easier visualization .

```
df.combine <- rbind(df.train, df.test)
cluster.df <- df.combine %>% select(c(numericVars))  %>% select (-c(subscribers_for_last_30_days))
#character_vars <- lapply(cluster.df, class) == "character"
#cluster.df[, character_vars] <- lapply(cluster.df[, character_vars], as.factor)
#summary(cluster.df)
```

```
scaled_df <- scale(cluster.df)
attributes(scaled_df)$`scaled:center`
```

```
##                      subscribers                      video.views
##                     2.273724e+07                     1.136375e+10
##                          uploads   video_views_for_the_last_30_days
##                     9.864250e+03                     1.607874e+08
##                             year subscribers_for_last_30_days_no_na
##                     2.012698e+03                     2.664637e+05
```

```
attributes(scaled_df)$`scaled:scale`
```

```
##                      subscribers                      video.views
##                     1.689160e+07                     1.454988e+10
##                          uploads   video_views_for_the_last_30_days
##                     3.543267e+04                     2.964999e+08
##                             year subscribers_for_last_30_days_no_na
##                     4.274384e+00                     5.397331e+05
```

```
#summary(scaled_df)
```

4.2 Hierarchical Clustering

- Below utilize the dist function, to calculate ‘distance’ for the dataset.
- Here the method = “Euclidean” distance were used.
- The Hierarchical Clustering is performed, with K set to be 2, or 8. (Please refer section below for how 8 was determined)
- The print_cluster function is to print each cluster, with specific columns information added from original data frame . The specific columns can be any column. Even the categorical columns which were not used for clustering analysis .

```
d <- dist(scaled_df, method="euclidean")
pfit <- hclust(d, method="ward.D2")
hgroups <- cutree(pfit, k=8)
hgroups.2 <- cutree(pfit, k=2)


print_clusters <- function(df, groups, cols_to_print) {
Ngroups <- max(groups)
for (i in 1:Ngroups) {
print(paste("cluster", i))
print(df[groups == i, cols_to_print])
}
}

#print_clusters(df.combine, hgroups, c("Country","category"))
```

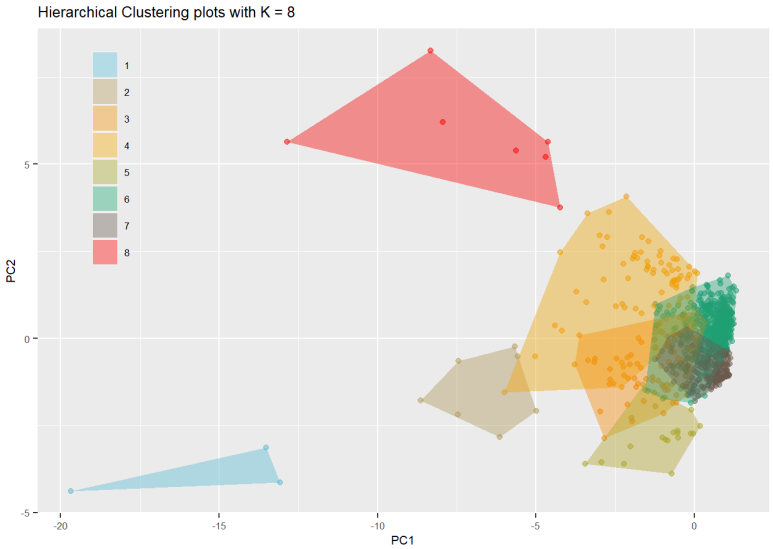
Combining the clustering results to each row of the project2D dataframe.

```
hclust.project2D <- cbind(project2D, cluster=as.factor(hgroups), country=df.combine$Country)

#hclust.project2D
```

Below function was using grDevices to calculate the data end point for each clusters, in order to be able to plot polygons.

```
library('grDevices')
find_convex_hull <- function(proj2Ddf, groups) {
do.call(rbind,
lapply(unique(groups),
FUN = function(c) {
f <- subset(proj2Ddf, cluster==c);
f[chull(f),]
}
)
)
}
```

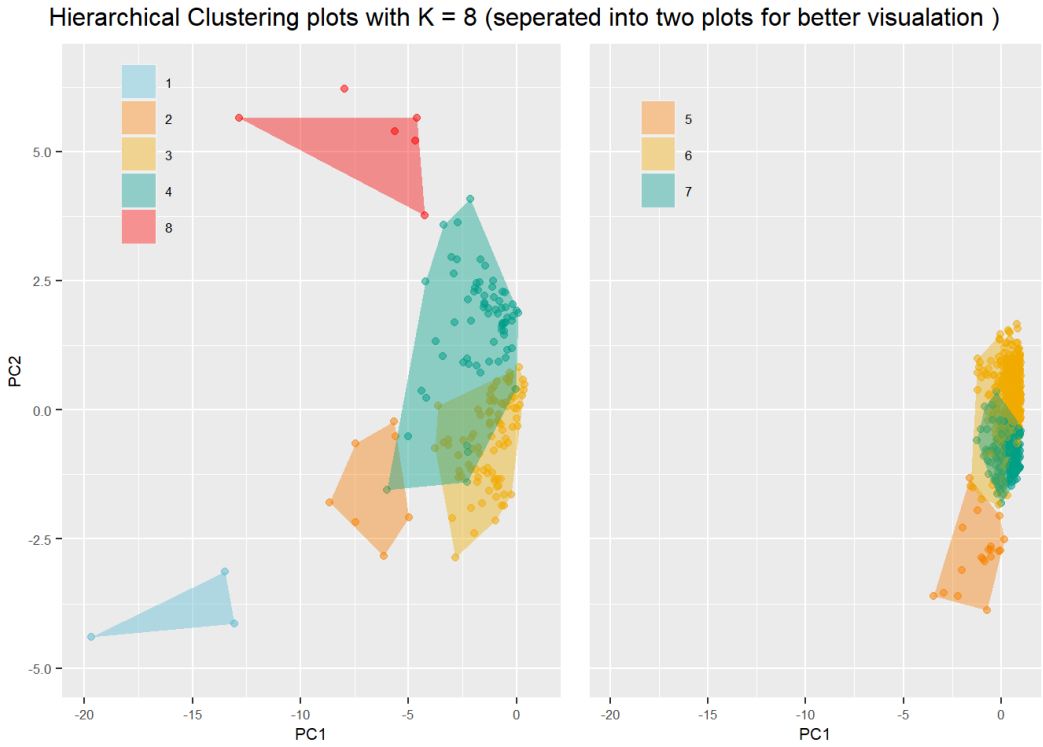
```
hclust.hull <- find_convex_hull(hclust.project2D, hgroups)

ggplot(hclust.project2D, aes(x=PC1, y=PC2)) +
geom_point(aes(colour=cluster), alpha=0.5)+ guides(colour="none") +
geom_polygon(data=hclust.hull, aes(group=cluster, fill=as.factor(cluster)),
alpha=0.4, linetype=0) + theme(text=element_text(size=8),legend.position = c(0.1, 0.75),legend.background = element_rect(fil
l = NA)) + scale_fill_manual(values = rev(wes_palette(8, name = "Darjeeling1", type = "continuous")), name = "") + scale_col
or_manual(values= rev(wes_palette("Darjeeling1", n = 8, type = "continuous"))) + ggtitle("Hierarchical Clustering plots with
K = 8 ")
```




```
# +scale_y_continuous(limits = c(-5, 6.5))+
# scale_x_continuous(limits = c(-20, 1))
```

- clustering plots (seperated into two plots for better visualation) with K = 8

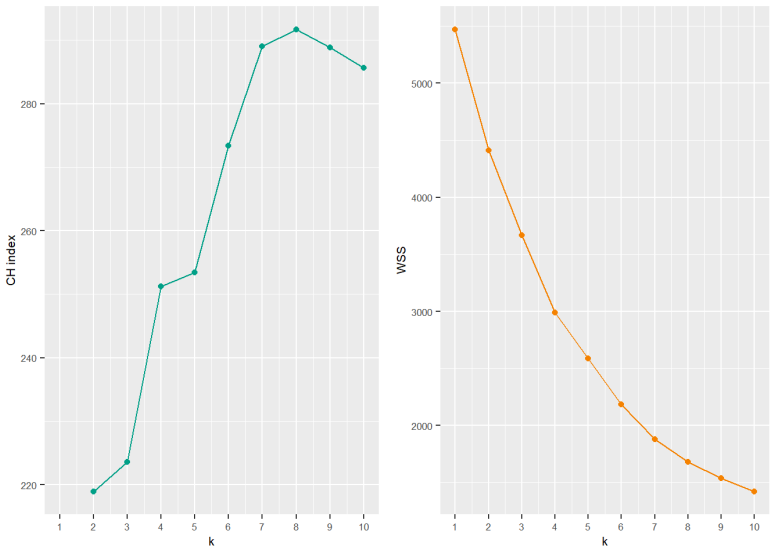
```
cl.1 <- ggplot(hclust.project2D[hclust.project2D$cluster %in% c(1,2,3,4,8),], aes(x=PC1, y=PC2)) +
geom_point(aes(colour=cluster), alpha=0.5)+ guides(colour="none") +
geom_polygon(data=hclust.hull[hclust.hull$cluster %in% c(1,2,3,4,8),], aes(group=cluster, fill=as.factor(cluster)),
alpha=0.4, linetype=0) + theme(text=element_text(size=8),legend.position = c(0.17, 0.85),legend.background = element_rect(fi
ll = NA)) + scale_fill_manual(values = rev(wes_palette(5, name = "Darjeeling1", type = "discrete")), name = "") + scale_colo
r_manual(values= rev(wes_palette("Darjeeling1", n = 5, type = "discrete"))) +scale_y_continuous(limits = c(-5, 6.5))+ scale_
x_continuous(limits = c(-20, 1))



cl.2 <- ggplot(hclust.project2D[hclust.project2D$cluster %in% c(5,6,7),], aes(x=PC1, y=PC2)) +
geom_point(aes(colour=cluster), alpha=0.5) + guides(colour="none") +
geom_polygon(data=hclust.hull[hclust.hull$cluster %in% c(5,6,7),], aes(group=cluster, fill=as.factor(cluster)),
alpha=0.4, linetype=0) + theme(text=element_text(size=8),legend.position = c(0.17, 0.85),legend.background = element_rect(fi
ll = NA), axis.text.y = element_blank(),axis.ticks.y = element_blank(), axis.title.y = element_blank()) + scale_fill_manual
(values = rev(wes_palette(4, name = "Darjeeling1", type = "discrete")), name = "") + scale_color_manual(values= rev(wes_pale
tte("Darjeeling1", n = 4, type = "discrete")))+scale_y_continuous(limits = c(-5, 6.5))+ scale_x_continuous(limits = c(-20,
1))

#grid.arrange(cl.1,cl.2,ncol = 2)

grid.arrange(arrangeGrob (cl.1, ncol = 1, nrow =1 ), arrangeGrob (cl.2, ncol=1, nrow = 1), widths = c(1.2,1),top = "Hierarch
ical Clustering plots with K = 8 (seperated into two plots for better visualation )")
```



- By using Clusterboot to check stability of all the 8 clusters.
- As shown below, the cluster 2, 3 were not stable, the rest performed well and quite stable.

```
sqr_euDist <- function(x, y) {
    sum((x - y)^2)}

wss <- function(clustermat) {
    c0 <- colMeans(clustermat)
    sum(apply( clustermat, 1, FUN=function(row) {sqr_euDist(row, c0)} ))
}


wss_total <- function(scaled_df, labels) {
    wss.sum <- 0
    k <- length(unique(labels))
    for (i in 1:k)
        wss.sum <- wss.sum + wss(subset(scaled_df, labels == i))
    wss.sum
}



tss <- function(scaled_df) {
   wss(scaled_df)
}


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

    npts <- nrow(scaled_df)
    wss.value <- numeric(kmax)
    wss.value[1] <- wss(scaled_df)

    if (method == "kmeans") {
        # kmeans
        for (k in 2:kmax) {
            clustering <- kmeans(scaled_df, k, nstart=10, iter.max=100)
            wss.value[k] <- clustering$tot.withinss
        }
    } else {
        # hclust
        d <- dist(scaled_df, method="euclidean")
        pfit <- hclust(d, method="ward.D2")
        for (k in 2:kmax) {
            labels <- cutree(pfit, k=k)
            wss.value[k] <- wss_total(scaled_df, labels)
        }
    }
    bss.value <- tss(scaled_df) - wss.value   # this is a vector
    B <- bss.value / (0:(kmax-1))             # also a vector
    W <- wss.value / (npts - 1:kmax)          # also a vector

    data.frame(k = 1:kmax, CH_index = B/W, WSS = wss.value)
}
```

```
crit.df <- CH_index(scaled_df, 10, method="hclust")
fig1 <- ggplot(crit.df, aes(x=k, y=CH_index)) +
geom_point(colour="#00A08A") + geom_line(colour="#00A08A") +
scale_x_continuous(breaks=1:10, labels=1:10) +
labs(y="CH index") + theme(text=element_text(size=8))
fig2 <- ggplot(crit.df, aes(x=k, y=WSS)) +
geom_point(colour="#F98400") + geom_line(colour="#F98400") +
scale_x_continuous(breaks=1:10, labels=1:10) +
theme(text=element_text(size=8))
grid.arrange(fig1, fig2, nrow=1)
```



- Above are CH index and WSS plots, based on the CH index, the K was decided to set to 8 for above Hierarchical Clustering.

4.3 K-MEANS Clustering

```
kbest.p <- 3
kmClusters <- kmeans(scaled_df, kbest.p, nstart=100, iter.max=100)
#kmClusters$centers
#kmClusters$size
```

```
groups<- kmClusters$cluster
#print_clusters(df.4, groups, "Country")
```

```
library(fpc)
kmClustering.ch <- kmeansruns(scaled_df, krange=1:10, criterion="ch")
kmClustering.ch$bestk
```

```
## [1] 8
```

```
kmClustering.asw <- kmeansruns(scaled_df, krange=1:10, criterion="asw")
kmClustering.asw$bestk
```

```
## [1] 2
```

```
# Compare the CH values for kmeans() and hclust().
print("CH index from kmeans for k=1 to 10:")
```

```
## [1] "CH index from kmeans for k=1 to 10:"
```

```
print(kmClustering.ch$crit)
```

```
##  [1]   0.0000 287.3402 269.2049 282.3477 309.2285 325.1524 347.0239 347.5415
##  [9] 335.9985 325.8409
```

```
print("ASW index from kmeans for k=1 to 10:")
```

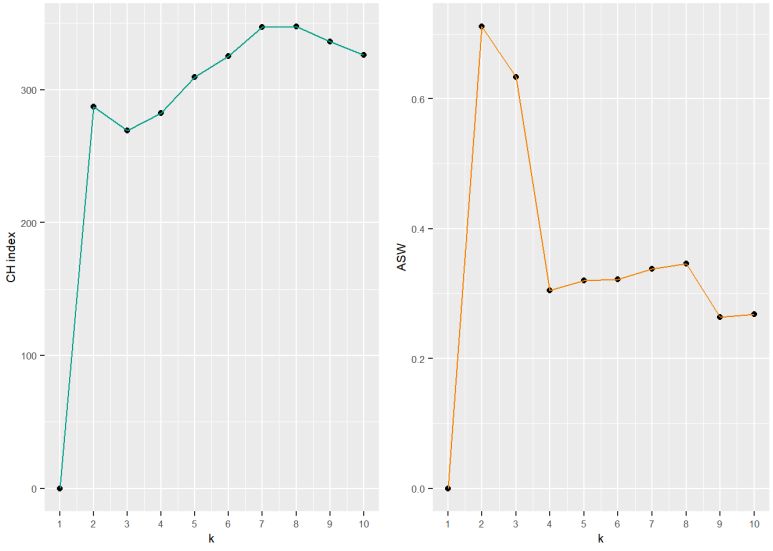
```
## [1] "ASW index from kmeans for k=1 to 10:"
```

```
print(kmClustering.asw$crit)
```

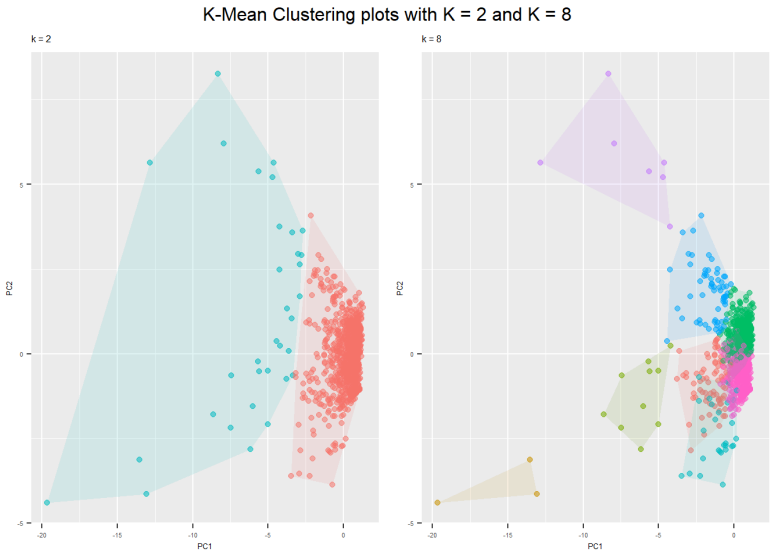
```
##  [1] 0.0000000 0.7113237 0.6330261 0.3050028 0.3202695 0.3219660 0.3382698
##  [8] 0.3459202 0.2632967 0.2683665
```

- Based on above recommendations from CH and ASW index, the K were chosen to be 8 or 2.
- The plot below are the visual representations of best K for K-Mean clusterings.

```
library(gridExtra)
kmCritframe <- data.frame(k=1:10, ch=kmClustering.ch$crit,
asw=kmClustering.asw$crit)
fig1 <- ggplot(kmCritframe, aes(x=k, y=ch)) +
geom_point() + geom_line(colour="#00A08A") +
scale_x_continuous(breaks=1:10, labels=1:10) +
labs(y="CH index") + theme(text=element_text(size=8))
fig2 <- ggplot(kmCritframe, aes(x=k, y=asw)) +
geom_point() + geom_line(colour="#F98400") +
scale_x_continuous(breaks=1:10, labels=1:10) +
labs(y="ASW") + theme(text=element_text(size=8))
grid.arrange(fig1, fig2, nrow=1)
```



```
fig <- c()
kvalues <- seq(2,8)
for (k in kvalues) {
groups <- kmeans(scaled_df, k, nstart=100, iter.max=100)$cluster
kmclust.project2D <- cbind(project2D, cluster=as.factor(groups),
country=df.4$Country)
kmclust.hull <- find_convex_hull(kmclust.project2D, groups)
assign(paste0("fig", k),
ggplot(kmclust.project2D, aes(x=PC1, y=PC2)) +
geom_point(aes( color=cluster, alpha = 0.1)) +
geom_polygon(data=kmclust.hull, aes(group=cluster, fill=cluster),
alpha=0.1, linetype=0) +
labs(title = sprintf("k = %d", k)) +
theme(legend.position="none", text=element_text(size=5))
)
}



grid.arrange(arrangeGrob (fig2, ncol = 1, nrow =1 ), arrangeGrob (fig8, ncol=1, nrow = 1), widths = c(1,1),top = "K-Mean Clu
stering plots with K = 2 and K = 8")
```



- Above has demonstrated the K mean clustering did a better job when K set to 2, comparing the the hierarchy clustering when K set to 2.
- Clustering can be used during the EDA process, to discover intrinsic relationship within the dataset.

```
require("cluster")
```

```
## Loading required package: cluster
```

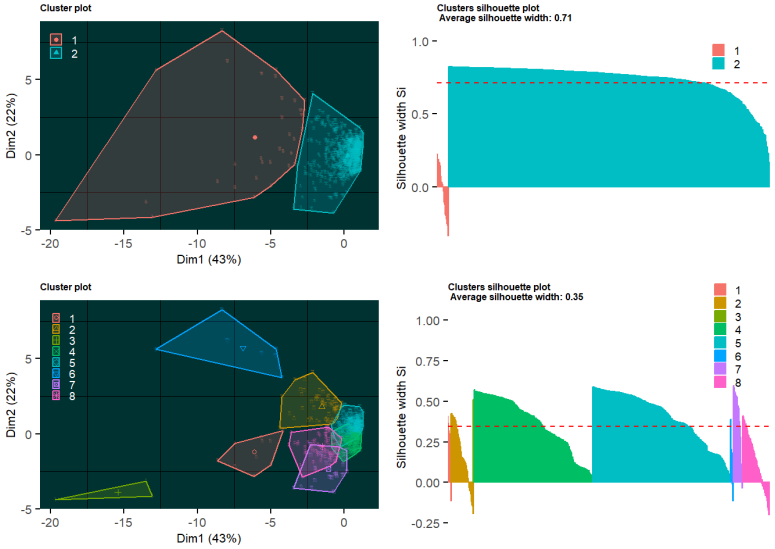
```
sil <- silhouette(km$cluster, dist(scaled_df))
kv.1 <- fviz_cluster(km, cluster.df, labelsize = 1,pointsize = 0.5 ,alpha = 0.2,  ellipse.type = "convex") + theme(panel.bac
kground = element_rect(fill = "#003333"),panel.grid.major = element_blank(),panel.grid.minor = element_line(colour = "blac
k"),legend.position = c(0.07, 0.9), legend.key = element_rect(fill = NA),legend.key.size = unit(1, 'cm'), legend.key.height
= unit(0.3, 'cm'), legend.key.width = unit(0.3, 'cm'), legend.title = element_blank(), legend.text =element_text(size=7, col
or = "white"),
    plot.title = element_text(size=6, face = "bold"),axis.text=element_text(size=8),axis.title=element_text(size=8),legend.b
ackground = element_rect(fill = NA))

kv.2 <- fviz_silhouette(sil,abel = TRUE) + theme(legend.position = c(0.87, 0.9), legend.key = element_rect(fill = NA),legen
d.key.size = unit(1, 'cm'), legend.key.height = unit(0.3, 'cm'), legend.key.width = unit(0.3, 'cm'), legend.title = element_
blank(), legend.text =element_text(size=7, color = "Black"),
    plot.title = element_text(size=6, face = "bold"),axis.text=element_text(size=8),axis.title=element_text(size=8),legend.b
ackground = element_rect(fill = NA))
```

```
##   cluster size ave.sil.width
## 1       1   33         -0.02
## 2       2  880          0.74
```

```
sil.8 <- silhouette(km.8$cluster, dist(scaled_df))
kv.3 <- fviz_cluster(km.8, cluster.df,labelsize = 1, pointsize = 0.5 ,alpha = 0.2)+ theme(panel.background = element_rect(fi
ll = "#003333"),panel.grid.major = element_blank(),panel.grid.minor = element_line(colour = "black"),legend.position = c(0.0
7, 0.75), legend.key = element_rect(fill = NA),legend.key.size = unit(0.8, 'cm'), legend.key.height = unit(0.2, 'cm'), legen
d.key.width = unit(0.2, 'cm'), legend.title = element_blank(), legend.text =element_text(size=7, color = "white"),
    plot.title = element_text(size=6, face = "bold"),axis.text=element_text(size=8),axis.title=element_text(size=8),legend.b
ackground = element_rect(fill = NA))
kv.4 <- fviz_silhouette(sil.8,abel = TRUE)+ theme(legend.position = c(0.87, 0.9), legend.key = element_rect(fill = NA),legen
d.key.size = unit(1, 'cm'), legend.key.height = unit(0.3, 'cm'), legend.key.width = unit(0.3, 'cm'), legend.title = element_
blank(), legend.text =element_text(size=7, color = "Black"),
    plot.title = element_text(size=6, face = "bold"),axis.text=element_text(size=8),axis.title=element_text(size=8),legend.b
ackground = element_rect(fill = NA))
```

```
##   cluster size ave.sil.width
## 1       1   10          0.21
## 2       2   63          0.17
## 3       3    3          0.41
## 4       4  336          0.36
## 5       5  392          0.40
## 6       6    7          0.12
## 7       7   27          0.39
## 8       8   75          0.14
```

```
grid.arrange( kv.1,kv.2,kv.3,kv.4, nrow=2)
```



- **Above plots by using fviz_cluster and fviz_silhouette, to further investigating the clustering results.**
- **The silhouette value is a measure of how similar an object is to its own cluster (cohesion), compared to other clusters (separation).**
- **Silouette analysis can be used to visualize the separation distances. It can also be used to choose the optimal number of clusters.**
- **Based on the Silouette plot for 2 clusters. Cluster 2 is below the average silhouette scores, and maybe a bad K pick.**

References

F Shofiyah, A Sofro. 2018. “Split and Conquer Method in Penalized Logistic Regression with Lasso (Application on Credit Scoring Data).” *Journal of Physics: Conference Series* 1108. doi :10.1088/1742-6596/1108/1/012107 (https://doi :10.1088/1742-6596/1108/1/012107).

Mathias Bärtl. 2018. “YouTube Channels, Uploads and Views: A Statistical Analysis of the Past 10 Years.” *Convergence* 24(1):Special issue: YouTube. https://doi.org/10.1177/1354856517736979 (https://doi.org/10.1177/1354856517736979).

Meher UN Nisa, Ghufran Ahmed, Danish Mahmood. 2021. “Optimizing Prediction of YouTube Video Popularity Using XGBoost.” *Electronics 2021* 10(23): 2962. https://doi.org/10.3390/electronics10232962 (https://doi.org/10.3390/electronics10232962).

Punam Wakel, Swati Dakhore, Nita Borkar. 2022. “Predictive Analysis of YouTube Using Machine Learning.” *International Research Journal of Modernization in Engineering Technology and Science* 2022-04: 2398–2404. https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/21964/final/fin_irjmets1651658132.pdf (https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2022/21964/final/fin_irjmets1651658132.pdf).