**Data Preprocessing and Understanding**

Author’s Name

Institutional Affiliation

Course Name and Number

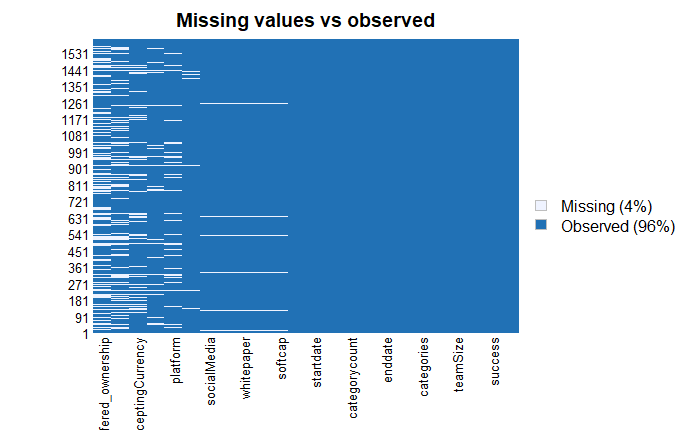
Professor’s Name

Assignment Due Date

Data Preprocessing and Understanding

After data collection, the next step is preprocessing. Data preprocessing as highlighted earlier is the gateway to building models and fitting data to models. It is a set of procedures that create, modify, delete, mutate and even change existing datapoints to create the necessary variables. Similar to a lot of real-life data our data had a lot of mishaps in it. The whole purpose of this particular phase in the lifecycle of analysis is to prepare the data for model building. Key structures are enacted that will clean, format, modify, update and nutate the data points in targeted variables into a ready dataset. To achieve the most optimal results, our preprocessing was done in two phases. For the data understanding model, preprocessing was done in a variable basis rather than across the entire dataset. This was to ensure only necessary steps were enacted for a certain variable and the maximum information can be learnt from it. However, for the model, the data had to be uniform and preprocessing was done across the entire model.

The first step of the preprocessing stage was visualization. In each variable understanding the unique terms is key. This helps give a rough sketch of out of the total how many are unique under each column. The next step is visualizing the NA. These are the empty slots under each row. The figure below shows the total number under each particular variable. The following is the graph showing the same. The understanding shows out of the 1606 possible columns, where do the missing lie. This brief graph can help make an understanding of how the preprocessing can be done.



**Figure 1.1 Missing Values**

The next step after visualizing missing values, is dealing with them/ This shall be done in two phases, for variables and this being mostly numerical variables with distributions, the mean of that column shall be used to fill the variables. This is done for variables that are not unique and fall under a certain distribution. The rest of the variables including logistic ones and factor variables are eliminated. This is done in descending order, whereby the variable with most missing values is used first. Visualization is done subsequently to ensure the most minimum number of rows are omitted from the dataset.

Some variables require scaling. This is done for two main purposes. The first purpose of scaling is ensuring that the distribution remains the same. The second goal is to ensure the mismatch between different variables is avoided in calculations. Smoothening out the biggest and tiniest numbers will ensure the model does not run into infinites and its log time is short and manageable. This set of procedures is done in the following sections.

#Install packages here

#import libraries  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.7  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidyr)  
library(dplyr)  
library(forecast)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(broom)  
library(ggplot2)  
library(stargazer)

##   
## Please cite as:

## Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

## R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

library(mltools)

##   
## Attaching package: 'mltools'

## The following object is masked from 'package:tidyr':  
##   
## replace\_na

library(data.table)

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

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

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

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(fitdistrplus)

## Loading required package: MASS

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

library("Hmisc")

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

#Import dataset  
library(readr)  
data <- read\_csv("data.csv", col\_types = cols(enddate = col\_date(format = "%d/%m/%Y"),   
 startdate = col\_date(format = "%d/%m/%Y")))  
head(data)

## # A tibble: 6 x 20  
## id success tokenNum teamSize country categories overallrating  
## <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 1 0 1.27e7 14 Estonia Charity,Education,Heal~ 2.6  
## 2 2 0 0 13 Singap~ Infrastructure 2.1  
## 3 3 1 2.22e9 9 Singap~ Platform,Business serv~ 2.5  
## 4 4 0 4 e7 0 USA Cryptocurrency,Busines~ 1.5  
## 5 5 1 4.5 e7 7 Mexico Internet,Infrastructur~ 2.1  
## 6 6 1 8 e7 20 Gibral~ Platform,Infrastructur~ 4.3  
## # ... with 13 more variables: offered\_ownership <dbl>, enddate <date>,  
## # startdate <date>, tokenName <chr>, tokenPrice <chr>, tokenType <chr>,  
## # platform <chr>, acceptingCurrency <chr>, softcap <dbl>, hardcap <dbl>,  
## # whitepaper <dbl>, video <dbl>, socialMedia <dbl>

#Create new variables that will be used  
#Log form of token number  
#Count of categories  
#Count of supported currecies  
#Duration of ICO  
data$tokenNum = ifelse(is.na(data$tokenNum),ave(data$tokenNum, FUN = function(x) mean(x, na.rm = 'TRUE')),data$tokenNum)  
  
data\_x = data %>%  
 mutate(logtokennumber = log(tokenNum),   
 categorycount = sapply(as.list(strsplit(categories,",")),length),   
 duration = as.numeric(enddate - startdate),  
 currencycount = sapply(as.list(strsplit(acceptingCurrency,",")),length)  
 )  
#data\_x = na.omit(data\_x)  
#data\_x = data\_x  
  
head(data\_x)

## # A tibble: 6 x 24  
## id success tokenNum teamSize country categories overallrating  
## <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 1 0 1.27e7 14 Estonia Charity,Education,Heal~ 2.6  
## 2 2 0 0 13 Singap~ Infrastructure 2.1  
## 3 3 1 2.22e9 9 Singap~ Platform,Business serv~ 2.5  
## 4 4 0 4 e7 0 USA Cryptocurrency,Busines~ 1.5  
## 5 5 1 4.5 e7 7 Mexico Internet,Infrastructur~ 2.1  
## 6 6 1 8 e7 20 Gibral~ Platform,Infrastructur~ 4.3  
## # ... with 17 more variables: offered\_ownership <dbl>, enddate <date>,  
## # startdate <date>, tokenName <chr>, tokenPrice <chr>, tokenType <chr>,  
## # platform <chr>, acceptingCurrency <chr>, softcap <dbl>, hardcap <dbl>,  
## # whitepaper <dbl>, video <dbl>, socialMedia <dbl>, logtokennumber <dbl>,  
## # categorycount <int>, duration <dbl>, currencycount <int>

#Create one hot encoding for some variables  
hotdata = data.frame(  
 Outcome = seq(1, 1606, by=1),  
 Variable = data$country  
)  
dummy <- dummyVars(" ~ .", data=hotdata)  
newdata <- data.frame(predict(dummy, newdata = hotdata))   
#newdata

#Get unique variables and their count  
cat("The platofrms used:", length(unique(data$platform)))

## The platofrms used: 64

cat("\n")

cat("The total number of countries:", length(unique(data$country)))

## The total number of countries: 117

cat("\n")

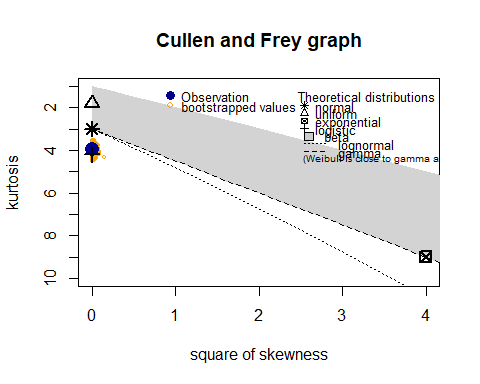
cat("The total number of token types is: ", length(unique(data$tokenType)))

## The total number of token types is: 44

#Getting Summary descriptive statistics of some key variables   
summarydata = data\_x %>%  
 dplyr::select(teamSize, overallrating, offered\_ownership, duration, logtokennumber, categorycount,)  
summary(summarydata)

## teamSize overallrating offered\_ownership duration   
## Min. : 0.00 Min. :0.80 Min. : 0.010 Min. :-61.00   
## 1st Qu.: 6.00 1st Qu.:2.30 1st Qu.: 0.450 1st Qu.: 27.00   
## Median :10.00 Median :2.80 Median : 0.600 Median : 31.00   
## Mean :11.17 Mean :2.85 Mean : 1.144 Mean : 40.19   
## 3rd Qu.:15.00 3rd Qu.:3.50 3rd Qu.: 0.740 3rd Qu.: 50.00   
## Max. :67.00 Max. :4.80 Max. :601.250 Max. :382.00   
## NA's :541 NA's :5   
## logtokennumber categorycount   
## Min. : -Inf Min. : 1.00   
## 1st Qu.:16.12 1st Qu.: 1.00   
## Median :18.42 Median : 2.00   
## Mean : -Inf Mean : 2.54   
## 3rd Qu.:20.37 3rd Qu.: 3.00   
## Max. :37.66 Max. :17.00   
##

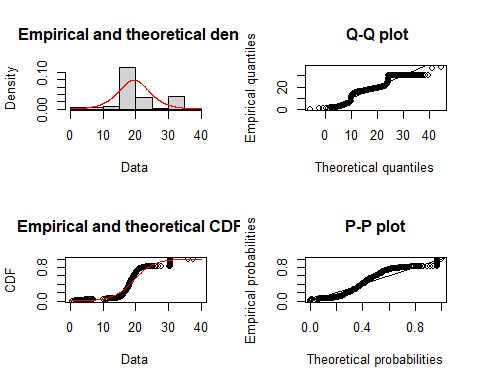
#Fitting and plotting distributions on some variables using Maximum Likelihood Estimation (MLE)  
# Log Normal distribution would be mostly targetted towards   
  
descdistribution <- function(data){  
 x = na.omit(data)  
 x = x[!is.infinite((x))]  
 x <- x[x > 0]  
 desc = descdist(x, boot = 1000)  
 print(summary(desc))  
}  
  
descdistribution(data\_x$logtokennumber)



## Length Class Mode   
## min 1 -none- numeric   
## max 1 -none- numeric   
## median 1 -none- numeric   
## mean 1 -none- numeric   
## sd 1 -none- numeric   
## skewness 1 -none- numeric   
## kurtosis 1 -none- numeric   
## method 1 -none- character

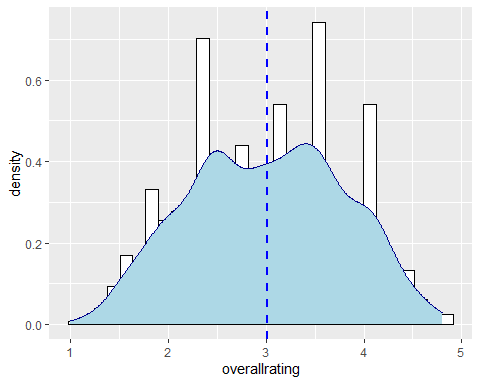
range01 <- function(x){(x-min(x))/(max(x)-min(x))}  
  
fitdistribution <- function(data){  
 x <- data[data > 0]  
 x = na.omit(x)  
 x = x[!is.infinite((x))]  
 fit <- fitdist(x, "logis")  
 print(summary(fit))  
 plot(fit)  
   
}  
  
fitdistribution(data\_x$logtokennumber)

## $start.arg  
## $start.arg$location  
## [1] 19.87259  
##   
## $start.arg$scale  
## [1] 3.373556  
##   
##   
## $fix.arg  
## NULL  
##   
## Fitting of the distribution ' logis ' by maximum likelihood   
## Parameters :   
## estimate Std. Error  
## location 19.443457 0.14151447  
## scale 3.171912 0.07450316  
## Loglikelihood: -4543.12 AIC: 9090.24 BIC: 9100.761   
## Correlation matrix:  
## location scale  
## location 1.0000000 0.1037904  
## scale 0.1037904 1.0000000



rand\_x = na.omit(summarydata)  
rand\_x = rand\_x[!is.infinite(rowSums(rand\_x)),]  
  
# Basic density  
ggplot(rand\_x, aes(x = overallrating)) + geom\_histogram(aes(y=..density..), colour="black", fill="white")+  
 geom\_density(color="darkblue", fill="lightblue") + geom\_vline(aes(xintercept=mean(overallrating)),  
 color="blue", linetype="dashed", size=1)

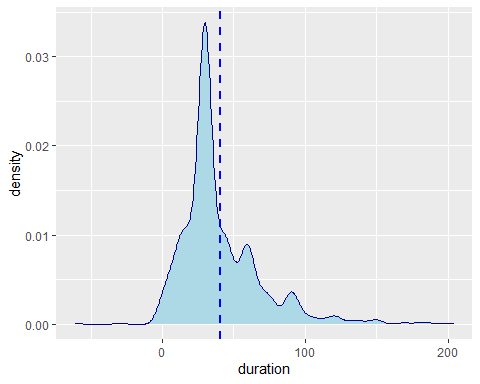
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



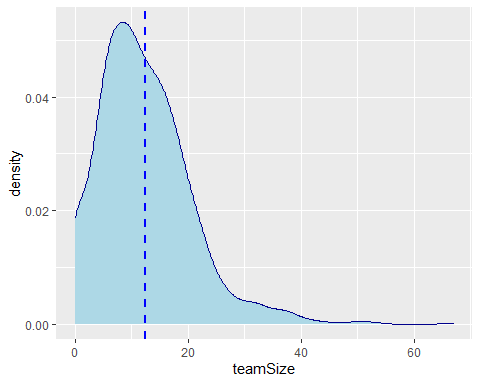
ggplot(rand\_x, aes(x = logtokennumber)) +   
 geom\_density(color="darkblue", fill="lightblue") + geom\_vline(aes(xintercept = 17.55),  
 color="blue", linetype="dashed", size=1)

## 

ggplot(rand\_x, aes(x = duration)) +   
 geom\_density(color="darkblue", fill="lightblue") + geom\_vline(aes(xintercept=mean(duration)),  
 color="blue", linetype="dashed", size=1)



ggplot(rand\_x, aes(x = teamSize)) +   
 geom\_density(color="darkblue", fill="lightblue") + geom\_vline(aes(xintercept=mean(teamSize)),  
 color="blue", linetype="dashed", size=1)



corr\_matrix <- rcorr(as.matrix(rand\_x))  
corr\_matrix

## teamSize overallrating offered\_ownership duration  
## teamSize 1.00 0.55 0.02 -0.04  
## overallrating 0.55 1.00 -0.01 -0.08  
## offered\_ownership 0.02 -0.01 1.00 -0.01  
## duration -0.04 -0.08 -0.01 1.00  
## logtokennumber 0.15 0.13 0.01 -0.04  
## categorycount 0.07 0.16 -0.01 0.00  
## logtokennumber categorycount  
## teamSize 0.15 0.07  
## overallrating 0.13 0.16  
## offered\_ownership 0.01 -0.01  
## duration -0.04 0.00  
## logtokennumber 1.00 -0.02  
## categorycount -0.02 1.00  
##   
## n= 987   
##   
##   
## P  
## teamSize overallrating offered\_ownership duration  
## teamSize 0.0000 0.5890 0.2090   
## overallrating 0.0000 0.8724 0.0097   
## offered\_ownership 0.5890 0.8724 0.7578   
## duration 0.2090 0.0097 0.7578   
## logtokennumber 0.0000 0.0000 0.8484 0.2383   
## categorycount 0.0199 0.0000 0.6704 0.9739   
## logtokennumber categorycount  
## teamSize 0.0000 0.0199   
## overallrating 0.0000 0.0000   
## offered\_ownership 0.8484 0.6704   
## duration 0.2383 0.9739   
## logtokennumber 0.5670   
## categorycount 0.5670

#table(data\_x$tokenType)  
cat("The rato of successful ICOs:")

## The rato of successful ICOs:

cat("\n")

table(data$success)

##   
## 0 1   
## 721 885

cat("The Ratio of ICOs with video marketing")

## The Ratio of ICOs with video marketing

cat("\n")

table(data$video)

##   
## 0 1   
## 516 1061

cat("The Ratio of ICOs that had set a minimum target")

## The Ratio of ICOs that had set a minimum target

cat("\n")

table(data$softcap)

##   
## 0 1   
## 891 686

cat("The Ratio of ICOs that had set a maximum target")

## The Ratio of ICOs that had set a maximum target

cat("\n")

table(data$hardcap)

##   
## 0 1   
## 486 1091

cat("The Ratio of ICOs with a whitepaper")

## The Ratio of ICOs with a whitepaper

cat("\n")

table(data$whitepaper)

##   
## 0 1   
## 77 1500

cat("The Unique values of the Platforms used for the fundraising")

## The Unique values of the Platforms used for the fundraising

cat("\n")

#table(data$platform)  
cat("The Countries")

## The Countries

cat("\n")

#table(data$country)