# Data Processing ----  
  
# set the seed for reproducibility ----  
set.seed(123)  
  
# libraries ---  
library("mice")

library(tm)

library(SnowballC)  
library(tidyr)  
library(dplyr)

library(caret)

library(ggplot2)  
library(ggcorrplot)

# import data ----  
data <- read.csv("Data/data\_cleaned.csv")  
  
# structure of data ----  
str(data)

# Loading data into dataframe ----  
df <- data  
  
# IDENTIFYING MISSING DATA ----  
## replace blank or missing with NA ----  
df[df == "" | df == " "] <- NA  
  
summary(df)

## Count the number of missing values in each column ----  
miss\_count <- df %>%   
 summarise\_all(~ sum(is.na(.))) %>%   
 pivot\_longer(everything(), names\_to = "variable", values\_to = "count") %>%   
 arrange(desc(count))  
  
## Create a horizontal bar plot of missing values ----  
ggplot(miss\_count, aes(x = 0, xend = count, y = variable, yend = variable)) +  
 geom\_segment(size = 2, color = "skyblue") +  
 geom\_text(aes(x = count + 1, label = ifelse(count > 0, count, "")), hjust = 0) +  
 labs(title = "Number of missing values in each variable",   
 x = "Count", y = "Variable") +  
 theme\_minimal() +  
 scale\_x\_continuous(limits = c(0, max(miss\_count$count) + 5))

## Visualizing observations with missing values ----  
# aggr(df, numbers = TRUE, prop = FALSE, combined = TRUE, cex.axis = 0.6)  
  
## observations with missing values ----  
missing\_data <- df[rowSums(is.na(df)) > 0, ]  
count(missing\_data)

# HANDLING MISSING DATA ----  
  
## Imputation of missing data ----  
### SI method ----  
#### priceUSD ----  
si <- df  
si$priceUSD[is.na(si$priceUSD)] <- mean(si$priceUSD, na.rm = TRUE)  
  
#### teamSize ----  
si$teamSize[is.na(si$teamSize)] <- round(mean(si$teamSize, na.rm = TRUE))  
  
### MI method ----  
colnames(df)

summary(df)

summary(df$teamSize)

summary(df$priceUSD)

mi\_df <- df %>%  
 mutate("is\_success" = ifelse(success == "Yes", 1, 0))  
  
imi <- mice(subset(mi\_df, select = c('hasVideo', 'rating', "teamSize", 'priceUSD',   
 "hasGithub","hasReddit", "coinNum",   
 "minInvestment", "distributedPercentage",   
 'is\_success', 'ico\_duration')), m = 5,   
 maxit = 50, method = "cart", seed = 500)

mi <- complete(imi)  
  
colnames(mi)

colnames(df)

# Summary of teamSize  
summary(si$teamSize)

summary(mi$teamSize)

# Summary of priceUSD  
summary(si$priceUSD)

summary(mi$priceUSD)

# Saving data from the two imputation methods  
df\_si <- si  
df\_mi <- cbind(df[!names(df) %in% names(mi)], mi)  
  
summary(df\_mi)

# Histograms of teamSize  
df %>%  
 ggplot(aes(x = teamSize)) +  
 geom\_histogram(binwidth = 3) +  
 labs(title = "Distribution of teamSize before imputation",  
 subtitle = "binwidth = 3")

df\_si %>%  
 ggplot(aes(x = teamSize)) +  
 geom\_histogram(binwidth = 3) +  
 labs(title = "Distribution of teamSize after simple imputation method",  
 subtitle = "binwidth = 3")

df\_mi %>%  
 ggplot(aes(x = teamSize)) +  
 geom\_histogram(binwidth = 3) +  
 labs(title = "Distribution of teamSize after multiple imputation method",  
 subtitle = "binwidth = 3")

# Boxplot of priceUSD  
df %>%  
 ggplot(aes(x = 1.0, y = log(priceUSD))) +  
 geom\_boxplot() +  
 labs(title = "Distribution of priceUSD before imputation") +  
 xlab("")

df\_si %>%  
 ggplot(aes(x = 1.0, y = log(priceUSD))) +  
 geom\_boxplot() +  
 labs(title = "Distribution of priceUSD after simple imputation method") +  
 xlab("")

df\_mi %>%  
 ggplot(aes(x = 1.0, y = log(priceUSD))) +  
 geom\_boxplot() +  
 labs(title = "Distribution of priceUSD after multiple imputation method") +  
 xlab("")

### replace missing country with Unknown ----  
df\_2 <- df\_mi %>%  
 mutate(countryRegion = ifelse(is.na(countryRegion),"Unknown", countryRegion))  
  
### replace blank cells with Unknown ----  
df\_2 <- df\_2 %>%  
 mutate(platform = ifelse(is.na(platform), "Unknown", platform))  
  
# HANDLING OUTLIERS ----  
summary(df\_2)

boxplot(df\_2$ico\_duration, main = "Boxplot of ICO duration",   
 ylab = "ICO duration")

z\_score <- scale(df\_2$ico\_duration)  
  
# identify the data points with Z-score greater than 3 or less than -3 ----  
outliers <- which(abs(z\_score) > 3)  
length(outliers)

df\_outliers <- df\_2[outliers, ]  
  
# changing start date of outlier   
df\_2[1092, "startDate"] <- "2020-01-10"  
  
# calculating new ico\_duration  
df\_2[1092, "ico\_duration"] <-   
 as.numeric(as.Date.character(df\_2[1092, "endDate"]) -   
 as.Date.character(df\_2[1092, "startDate"]))  
  
# exploring the dependent variable ----  
table(df\_2$success)

summary(df\_2$success)

# Factoring target variable ----  
df\_2$success <- factor(df\_2$success, levels = c("Yes", "No"))  
  
# Dropping irrelevant columns ----  
df\_2 <- df\_2 %>%  
 select(-startDate, -endDate, -is\_success)  
  
summary(df\_2)

# separate categorical, text, and numerical columns  
df\_numerical <- df\_2 %>%  
 select(-brandSlogan, -countryRegion, -platform, -success)  
  
df\_categorical <- df\_2 %>%  
 select(countryRegion, platform)  
  
df\_text <- df\_2 %>%  
 select(brandSlogan)  
  
df\_target <- df\_2 %>%  
 select(success)  
  
# correlation plot of numerical variables  
cor(df\_numerical)

# corrplot(cor(df\_numerical, method = "number")   
ggcorrplot(cor(df\_numerical)) +  
 labs(title = "Correlation Plot of Numerical Variables") +  
 theme(plot.title = element\_text(hjust = 0, size = 10, face = "bold"))

# Dummy coding categorical variables  
df\_categorical\_encoded <- as.data.frame(model.matrix(~.-1, df\_categorical))  
  
# testing  
sum(df\_categorical\_encoded[1, ])

# processing text column ----  
corpus <- Corpus(VectorSource(df\_text$brandSlogan))  
corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- tm\_map(corpus, removePunctuation)

corpus <- tm\_map(corpus, removeNumbers)

corpus <- tm\_map(corpus, removeWords, stopwords("english"))

corpus <- tm\_map(corpus, stemDocument)

dtm <- DocumentTermMatrix(corpus)  
  
# Convert the dtm to a data frame and bind it with the response variable  
df\_text\_tokenized <- as.data.frame(as.matrix(dtm))  
  
# just cleaned data  
df\_cleaned <- cbind(df\_target, df\_numerical, df\_categorical\_encoded,  
 df\_text\_tokenized)  
  
# save data  
write.csv(df\_cleaned, file = "Data/data\_preprocessed.csv", row.names = FALSE)