```
##################################################################################
# Project:     CredX identify the right customers using predictive models
# Description:   Data processing and cleaning
# Data:        Creditbureau_data.csv& demographic_data
# By:          Jyothi,Avinash,Shiva,Shail
##################################################################################


#INSTALLING  REQUIRED PACKAGES AND LOAD LIBRARIES
install.packages("mlbench", dependencies = TRUE)
install.packages("Information", dependencies = TRUE)
install.packages("e1071", dependencies = TRUE)
install.packages("caret", dependencies = TRUE)
install.packages("installr", dependencies = TRUE)
install.packages("rattle", dependencies = TRUE)
library(mlbench)
library(Hmisc)
library(Information)
library(dplyr)
library(caret)
library(e1071)
library(installr)
library(corrplot)
library(rattle)

##

View(demographic_data)
#LOAD DATASETS
getwd()
demographic_data<-read.csv("E:/Learning Workdirectory/CAPSTONE/Demographic data.csv")
```

```
View(demographic_data)



dim(demographic_data)

# there are 71295 rows and 12 varaibles of demographic_data dataset



#point of view to think:

#  we must think what factors/features  influence the  creidt risk


# Standards ----------------------------------------------------------------

# Barplot --- Univariate Analysis for Categorical  Variables

# BoxPlot --  Univariate TO spot Outliers


## DEMOGRAPHIC  Exploratory Data Analysis----------------------------
============================================================================


## Removing Duplicate Values

demographic_data<-unique(demographic_data)


##Data Summarization

summary(demographic_data)

sapply(demographic_data,class)


##Dimension

dim(demographic_data)


##-----------------------------------------------------------------------------------------------------------------

##Skewness  for all Numeric vairbales       Negative means Mean is less than Median, and Left Skewed,

                                   ##  Positive means Mean is More than Median and Right Skewed

## Numeric variables in Demographic dataset are :
```

```
skewness(demographic_data$Age)    #-0.009038358

skewness(demographic_data$Application.ID)  #0.002931744

skewness(demographic_data$Income)   # 0.1883371

skewness(demographic_data$No.of.months.in.current.residence) #0.9880599

skewness(demographic_data$No.of.months.in.current.company)  #0.1208541

##----------------------------------------------------------------------------------------------------------------


names(demographic_data)

##----------------------------------------------------------------------------------------------------------------


#1. "Application.ID"

describe(demographic_data$Application.ID)

# ""Application.ID" it has no impact on the final model to decide credit risk so we remove the first column.

demographic_data <- demographic_data[-1]

names(demographic_data)

##----------------------------------------------------------------------------------------------------------------


#2. Age:   -- -3 to 65, Integer

# credit card below 19 years need not to be considered. so such kind of data can be invalid and can be removed.

hist(demographic_data[,1])

plot(table(demographic_data$Age))

#Removing -3 row, removing rows where age less than 19 no credit card company offers card below 19 years

demographic_data<- demographic_data[-16316,]

demographic_data<- demographic_data[!(demographic_data$Age <= 19), ]

dim(demographic_data)

names(demographic_data)

##----------------------------------------------------------------------------------------------------------------
```

```
#3.Gender    76.4% Male , 23.6% Female

plot(table(demographic_data$Gender))

#interpretaiton:  There are More Males than Females.

describe(demographic_data$Gender)

which(demographic_data$Gender == '')     #-- 39404

#removing invalid Gender value

which(demographic_data$Gender == '')

demographic_data<- demographic_data[-39404,]

##---------------------------------------------------------------------------------------------------------------


#4.Marital.Status..at.the.time.of.application   14.8% single  10516   and 85.2% Married -- 60661

plot(table(demographic_data$Marital.Status..at.the.time.of.application.))

describe(demographic_data$Marital.Status..at.the.time.of.application.)

summary(demographic_data$Marital.Status..at.the.time.of.application.)

male_married<-subset(demographic_data,demographic_data$Gender == 'M'   &
demographic_data$Marital.Status..at.the.time.of.application. == 'Married')

male_single<-subset(demographic_data,demographic_data$Gender == 'M'    &
demographic_data$Marital.Status..at.the.time.of.application. == 'Single')

female_single<-subset(demographic_data,demographic_data$Gender == 'F'  &
demographic_data$Marital.Status..at.the.time.of.application. == 'Single')

female_married<-subset(demographic_data,demographic_data$Gender == 'F' &
demographic_data$Marital.Status..at.the.time.of.application. == 'Married')

which(demographic_data$Marital.Status..at.the.time.of.application.!= 'Married' &
demographic_data$Marital.Status..at.the.time.of.application. != 'Single')

#removing invalid marital status at the time of application data

demographic_data <- demographic_data[-c(6289, 35423, 48305, 50634, 59192),]

dim(demographic_data)

##---------------------------------------------------------------------------------------------------------------


#5. demographic_data$No.of.dependents

plot(table(demographic_data$No.of.dependents)  )

summary(demographic_data$No.of.dependents)
```

```
describe(demographic_data$No.of.dependents)

#checking for NA's and Removing

which(is.na(demographic_data$No.of.dependents))

demographic_data <- demographic_data[-c(4587,43315),]

##--------------------------------------------------------------------------------------------------------------

View(demographic_data)

#6. demographic_data$Income

summary(demographic_data$Income)

plot(table(demographic_data$Income))

#-0.5, 0 values are invalid so we can remove them


##--------------------------------------------------------------------------------------------------------------


#7. demographic_data$Education

plot(table(demographic_data$Education))

table(demographic_data$Profession)

##--------------------------------------------------------------------------------------------------------------


#8. demographic_data$Type.of.residence

plot(table(demographic_data$Type.of.residence))

describe(demographic_data$Type.of.residence)

##--------------------------------------------------------------------------------------------------------------


#9.demographic_data$No.of.months.in.current.residence

describe(demographic_data$No.of.months.in.current.residence)

#-- 6 months, 34 Months as Median

plot(table(demographic_data$No.of.months.in.current.residence))

##--------------------------------------------------------------------------------------------------------------


#10.No.of.months.in.current.company

describe(demographic_data$No.of.months.in.current.company)
```

```
plot(table(demographic_data$No.of.months.in.current.company))
##------------------------------------------------------------------------------------------------------------------

#11.Performance.Tag
describe(demographic_data$Performance.Tag)
plot(table(demographic_data$Performance.Tag))

#Missing Values Treatment of Demographic Variables
dim(demographic_data)
demo1<- unique(demographic_data)
dim(demo1)
##------------------------------------------------------------------------------------------------------------------


## Missing Values Analysis and Imputation If Required---- Start------




## Missing Values  ---------------------------------------End--------

# Outliers Treatment ---------------------------Outliers start------
# Demographic _Data
#step1  sort variable
#step2  calculate q1,q2,q3
#step3  calculate lower_threshold, upper_threshold
sorted_demographc_age<- sort(demographic_data$Age)
q1 <- as.numeric(quantile(sorted_demographc_age)[2])
q2 <- as.numeric(quantile(sorted_demographc_age)[3])
q3 <- as.numeric(quantile(sorted_demographc_age)[4])
lower_threshold <- q1 - (1.5 * IQR(sorted_demographc_age))
upper_threshold <- q3 + (1.5 + IQR(sorted_demographc_age))
```

```
out_liers_age<-sorted_demographc_age[which(sorted_demographc_age < lower_threshold | sorted_demographc_age > upper_threshold)]

print(out_liers_age)

boxplot(demographic_data$Age)



plot(demographic_data$Marital.Status..at.the.time.of.application.)

boxplot(demographic_data$No.of.dependents)

boxplot(demographic_data$Income)

describe(demographic_data$Income)



#income:

sorted_demographc_income<- sort(demographic_data$Income)

q1 <- as.numeric(quantile(sorted_demographc_income)[2])

q2 <- as.numeric(quantile(sorted_demographc_income)[3])

q3 <- as.numeric(quantile(sorted_demographc_income)[4])

lower_threshold <- q1 - (1.5 * IQR(sorted_demographc_income))

upper_threshold <- q3 + (1.5 + IQR(sorted_demographc_income))

out_liers_income<-sorted_demographc_income[which(sorted_demographc_income < lower_threshold | sorted_demographc_income > upper_threshold)]

print(out_liers_income)


#

plot(demographic_data$Education)

plot(demographic_data$Profession)

plot(demographic_data$Type.of.residence)



#No.of.months.in.current.residence

boxplot(demographic_data$No.of.months.in.current.residence)
```

```
sorted_No.of.months.in.current.residence<-
sort(demographic_data$No.of.months.in.current.residence)

q1 <- as.numeric(quantile(sorted_No.of.months.in.current.residence)[2])

q2 <- as.numeric(quantile(sorted_No.of.months.in.current.residence)[3])

q3 <- as.numeric(quantile(sorted_No.of.months.in.current.residence)[4])

lower_threshold <- q1 - (1.5 * IQR(sorted_No.of.months.in.current.residence))

upper_threshold <- q3 + (1.5 + IQR(sorted_No.of.months.in.current.residence))

out_liers_at_residence<-
sorted_No.of.months.in.current.residence[which(sorted_No.of.months.in.current.residence <
lower_threshold | sorted_No.of.months.in.current.residence > upper_threshold)]

print(out_liers_at_residence)

#removing outliers

demographic_data1<- demographic_data

dim(demographic_data)

demographic_data<-
filter(demographic_data,demographic_data$No.of.months.in.current.residence < 115.5)


plot(demographic_data$Performance.Tag)


# Outliers Treatment  ----------------------------Outliers End---------------

View(demographic_data)




## Feature Selection

## Response Varaible  : demographic_data$Performance.Tag

describe(demographic_data$Performance.Tag)

sapply(demographic_data,class)

## Data Transformation


## Taking Backup

backup<-demographic_data
```

```
View(backup)

### Making all column values to Uppercase for future data transformations

## Converting Data types
## Gender
demographic2<-demographic_data
View(demographic2)
View(demographic_data)
demographic2$Gender<- as.numeric(demographic2$Gender)
# 2 -- Female
# 3 -- Male

##
demographic2$Marital.Status..at.the.time.of.application.<-
as.numeric(demographic_data$Marital.Status..at.the.time.of.application.)
View(demographic2)
View(demographic_data)
# 2 -- Married
# 3 -- Single

describe(demographic2$Education)
#Null     1
#Bachelor 2
#Masters  3
#Others   4
#Phd      5
#Professional  6
```

```
demographic2$Education<-as.numeric(demographic2$Education)

demographic2$Profession<-as.numeric(demographic2$Profession)

demographic2$Type.of.residence<-as.numeric(demographic2$Type.of.residence)

sapply(demographic2, class)



## Before we work on Response variable lets look at mIssing values Treatment of Response variable

describe(demographic2$Performance.Tag)

demographic2$Performance.Tag[which(is.na(demographic2$Performance.Tag) == T)]<-9

demographic2$Performance.Tag[which(is.na(demographic2$Performance.Tag) == 9)]<-0

describe(demographic2$Performance.Tag)



# calculate the pre-process parameters from the dataset

preprocessParams <- preProcess(demographic2, method=c("center", "scale", "pca"))

# summarize transform parameters

print(preprocessParams)

# transform the dataset using the parameters

transformed <- predict(preprocessParams, demographic2)

# summarize the transformed dataset

summary(transformed)


## correlations

correlations <- cor(demographic2[,1:10])

corrplot(correlations, method="circle")
```

```
# demographic2$Performance.Tag[which(is.na(demographic2$Performance.Tag) == T)]<-9

# SOUTH1 <- subset(demographic2, demographic2$Performance.Tag == 9)

# demographic2$Performance.Tag<-demographic2$Performance.Tag[-SOUTH1]

# View(demographic2)
```

```
## we have no idea which algorith will best fit for this problem

## Generally glm, glmnet ( for logistic regression because reponse variable is of integer binomial)

## we use RMSE, R2 as Accurary Evaluation Metrics

## Decision Trees and SVM may also best fit algorithms in this case

## RMSE --> RMSE will give a gross idea of how wrong all predictions are (0 is perfect)

## R2   --> R2 will give an idea of how well the model has ???t the data (1 is perfect, 0 is worst).
```

```
## Evaluation Metrics : RMSE, R2

## Resampling Method  : Repeated CV

## Machine Learning Algorithms Used:
```

```
demographic3<-demographic2

View(demographic3)

# Split out validation dataset

# create a list of 80% of the rows in the original dataset we can use for training
```

```
set.seed(7)

validationIndex <- createDataPartition(demographic3$Performance.Tag, p=0.80, list=FALSE)

# select 20% of the data for validation

validation <- demographic3[-validationIndex,]

# use the remaining 80% of data to training and testing the models

dataset <- demographic3[validationIndex,]


### validation is our Test Dataset

###  dataset  is our Train Dataset



# Run algorithms using 10-fold cross validation

trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)

metric <- "RMSE"


# Lets divide the problem into Linear Regression and Non Linear Regression

# Linear Regression Algorithms      : Linear Regression (LR), Generalized Linear Regression (GLM) and Penalized Linear Regression (GLMNET)

# Non Linear Regression Algorithms  : Classification and Regression Trees (CART), Support Vector Machines (SVM) with a radial basis function and k-Nearest Neighbors (KNN)



######==============================================================================================================


# LM

set.seed(7)

fit.lm <- train(Performance.Tag~., data=dataset, method="lm", metric=metric, preProc=c("center", "scale"), trControl=trainControl)

plot(fit.lm$finalModel)

text(fit.lm$finalModel)
```

```
# GLM

set.seed(7)

fit.glm <- train(Performance.Tag~., data=dataset, method="glm", metric=metric, preProc=c("center", "scale"), trControl=trainControl)

plot(fit.glm$finalModel)

text(fit.glm$finalModel)

# GLMNET

set.seed(7)

fit.glmnet <- train(Performance.Tag~., data=dataset, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl=trainControl)

plot(fit.glmnet$finalModel)

text(fit.glmnet$finalModel)

# SVM

#set.seed(7)

#fit.svm <- train(Performance.Tag~., data=dataset, method="svmRadial", metric=metric, preProc=c("center", "scale"), trControl=trainControl)

# CART

set.seed(7)

grid <- expand.grid(.cp=c(0, 0.05, 0.1))

fit.cart <- train(Performance.Tag~., data=dataset, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center", "scale"), trControl=trainControl)

##fancyRpartPlot(fit.cart$finalModel)

plot(fit.cart$finalModel)

text(fit.cart$finalModel)

# KNN

##set.seed(7)

##fit.knn <- train(Performance.Tag~., data=dataset, method="knn", metric=metric, preProc=c("center", "scale"), trControl=trainControl)



#Lets compare algorithms by uisng a simple table which shows all results
```

```
results <- resamples(list(LM=fit.lm, GLM=fit.glm, GLMNET=fit.glmnet, CART=fit.cart))

summary(results)

dotplot(results)




## Transformation  using Box Cox

# LM

set.seed(7)

fit.lm <- train(Performance.Tag~., data=dataset, method="lm", metric=metric, preProc=c("center", "scale","boxcox"), trControl=trainControl)

plot(fit.lm$finalModel)

text(fit.lm$finalModel)

# GLM

set.seed(7)

fit.glm <- train(Performance.Tag~., data=dataset, method="glm", metric=metric, preProc=c("center", "scale","boxcox"), trControl=trainControl)

plot(fit.glm$finalModel)

text(fit.glm$finalModel)

# GLMNET

set.seed(7)

fit.glmnet <- train(Performance.Tag~., data=dataset, method="glmnet", metric=metric, preProc=c("center", "scale","boxcox"), trControl=trainControl)

plot(fit.glmnet$finalModel)

text(fit.glmnet$finalModel)

# SVM

#set.seed(7)

#fit.svm <- train(Performance.Tag~., data=dataset, method="svmRadial", metric=metric, preProc=c("center", "scale"), trControl=trainControl)

# CART

set.seed(7)

grid <- expand.grid(.cp=c(0, 0.05, 0.1))
```

```
fit.cart <- train(Performance.Tag~., data=dataset, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center", "scale","boxcox"), trControl=trainControl)

plot(fit.cart$finalModel)

text(fit.cart$finalModel)

# KNN

##set.seed(7)

##fit.knn <- train(Performance.Tag~., data=dataset, method="knn", metric=metric, preProc=c("center", "scale","boxcox"), trControl=trainControl)




#Lets compare algorithms by uisng a simple table which shows all results

results <- resamples(list(LM=fit.lm, GLM=fit.glm, GLMNET=fit.glmnet, CART=fit.cart))

summary(results)

boxplot(results)


View(demographic3)

install.packages("MASS")

library(MASS)

library(car)

demographic3$Performance.Tag

# Divide you data in 70:30


set.seed(101)

indices= sample(1:nrow(demographic3), 0.7*nrow(demographic3))


train=demographic3[indices,]

test = demographic3[-indices,]


#------------------------------Multiple Linear regression-----------------------------------

#------------------------------------------------------------------------------------------
```

```
# Develop the first model

model_1 <-lm(Performance.Tag~.,data=train[,-1])
summary(model_1)

#-------------------------------------------------------------------------------------------

# Apply the stepwise approach

step <- stepAIC(model_1, direction="both")

#-------------------------------------------------------------------------------------------


# Run the step object

step

#-------------------------------------------------------------------------------------------

# create a new model_2 after stepwise method
model_2 <-lm(formula = Performance.Tag ~ Marital.Status..at.the.time.of.application. +
             No.of.dependents + Income + Education + No.of.months.in.current.residence +
             No.of.months.in.current.company, data = train[, -1])

#-------------------------------------------------------------------------------------------

# summary of model_2

summary(model_2)
```

```
vif(model_2)

#Remove the variables from the model whose VIF is more than 2


# But check the maximum VIF and then the significance value of that variable, and then take the call of removing this variable

# Remove the "Marital.Status..at.the.time.of.application." variable


model_3 <-lm(formula = Performance.Tag ~  No.of.dependents + Income + Education + No.of.months.in.current.residence +

             No.of.months.in.current.company, data = train[, -1])

summary(model_3)

vif(model_3)



# But check the maximum VIF and then the significance value of that variable, and then take the call of removing this variable

# Remove the "Education" variable



model_4 <-lm(formula = Performance.Tag ~  No.of.dependents + Income  + No.of.months.in.current.residence +

             No.of.months.in.current.company, data = train[, -1])

summary(model_4)

vif(model_4)

#-------------------------------------------------------------------------------------------


# Test the model on test dataset


Predict_1 <- predict(model_4,test[,-c(1,20)])


#-------------------------------------------------------------------------------------------
```

```
# Add a new column "test_predict" into the test dataset

test$Performance.Tag <- Predict_1

#------------------------------------------------------------------------------------------

# calculate the test R2

cor(test$Performance.Tag,test$Performance.Tag)
cor(test$Performance.Tag,test$Performance.Tag)^2


# # ---------------------------------------------------------------------



## Model Building

##---------Logistic Regression----------#

# Required Packages
install.packages("caret")
install.packages("caTools")
install.packages("dummies")

library(caret)
library(caTools)
library(dummies)

#-------------------------------------------------------
```

```
# splitting into train and test data

set.seed(1)

split_indices <- sample.split(demographic_data$Performance.Tag, SplitRatio = 0.70)

train <- demographic_data[split_indices, ]

test <- demographic_data[!split_indices, ]

nrow(train)/nrow(demographic_data)

nrow(test)/nrow(demographic_data)

#---------------------------------------------------------

### Model 1: Logistic Regression

install.packages("MASS")

library(MASS)

library(car)

logistic_1 <- glm(Performance.Tag~ ., family = "binomial", data = train)

summary(logistic_1)

#---------------------------------------------------------
```

```
# Using stepwise algorithm for removing insignificant variables

# stepAIC has removed some variables and only the following ones remain

sapply(train,class)

logistic_2 <- glm(formula = Performance.Tag ~ Age+Gender+Marital.Status..at.the.time.of.application.+No.of.dependents+Education+No.of.months.in.current.company +No.of.months.in.current.residence+Profession+Type.of.residence +Income + Education, family = "binomial", data = train)

# checking vif for logistic_2

vif(logistic_2)

summary(logistic_2)

#---------------------------------------------------------
#---------------------------------------------------------

# removing "Age"since vif is high and also the variable is not significant
logistic_3 <- glm(formula = Performance.Tag ~ Gender+Marital.Status..at.the.time.of.application.+No.of.dependents+Education+No.of.months.in.current.company +No.of.months.in.current.residence+Profession+Type.of.residence +Income + Education, family = "binomial", data = train)

# checking vif for logistic_3

vif(logistic_3)

summary(logistic_3)
#removing gender and Martial status of the applicant dur to the variables are insignificant
```

logistic_4 <- glm(formula = Performance.Tag ~ No.of.dependents+Education+No.of.months.in.current.company +No.of.months.in.current.residence+Profession+Type.of.residence +Income + Education, family = "binomial", data = train)

checking vif for logistic_4

vif(logistic_4)

summary(logistic_4)

#removing No of dependents and educationBachelor to the variables are insignificant

logistic_5 <- glm(formula = Performance.Tag ~ No.of.months.in.current.company +No.of.months.in.current.residence+Profession+Type.of.residence +Income, family = "binomial", data = train)

checking vif for logistic_5

vif(logistic_5)

summary(logistic_5)

#removing No.of.months.in.current.company to the variables are insignificant

logistic_6 <- glm(formula = Performance.Tag ~ No.of.months.in.current.residence+Profession+Type.of.residence +Income, family = "binomial", data = train)

checking vif for logistic_6

vif(logistic_6)

summary(logistic_6)

#removing No.of.months.in.current.residence to the variables are insignificant

logistic_7 <- glm(formula = Performance.Tag ~ +Profession+Type.of.residence +Income, family = "binomial", data = train)

checking vif for logistic_7

```
vif(logistic_7)

summary(logistic_7)
logistic_final <- logistic_7
#-------------------------------------------------------

# Predicting probabilities of responding for the test data

predictions_logit <- predict(logistic_final, newdata = test[, -61], type = "response")
summary(predictions_logit)

#-------------------------------------------------------


## Model Evaluation: Logistic Regression

# Let's use the probability cutoff of 50%.

predicted_response <- factor(ifelse(predictions_logit >= 0.50, "yes", "no"))

# Creating confusion matrix for identifying the model evaluation.

conf <- confusionMatrix(predicted_response, test$Performance.Tag, positive = "yes")

conf

#-------------------------------------------------------

# Let's find out the optimal probalility cutoff

perform_fn <- function(cutoff)
```

```
{
  predicted_response <- factor(ifelse(predictions_logit >= cutoff, "yes", "no"))
  conf <- confusionMatrix(predicted_response, test$Performance.Tag, positive = "yes")
  acc <- conf$overall[1]
  sens <- conf$byClass[1]
  spec <- conf$byClass[2]
  out <- t(as.matrix(c(sens, spec, acc)))
  colnames(out) <- c("sensitivity", "specificity", "accuracy")
  return(out)
}

#---------------------------------------------------------

# Creating cutoff values from 0.01 to 0.99 for plotting and initiallizing a matrix of 1000 X 4.

s = seq(.01,.99,length=100)

OUT = matrix(0,100,3)


for(i in 1:100)
{
  OUT[i,] = perform_fn(s[i])
}

#---------------------------------------------------------

# plotting cutoffs
plot(s, OUT[,1],xlab="Cutoff",ylab="Value",cex.lab=1.5,cex.axis=1.5,ylim=c(0,1),type="l",lwd=2,axes=FALSE,col=2)
axis(1,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)
```

```
axis(2,seq(0,1,length=5),seq(0,1,length=5),cex.lab=1.5)

lines(s,OUT[,2],col="darkgreen",lwd=2)

lines(s,OUT[,3],col=4,lwd=2)

box()

legend(0,.50,col=c(2,"darkgreen",4,"darkred"),lwd=c(2,2,2,2),c("Sensitivity","Specificity","Accuracy"))



#---------------------------------------------------------


cutoff <- s[which(abs(OUT[,1]-OUT[,2])<0.01)]



# Let's choose a cutoff value of 12% for final model


predicted_response <- factor(ifelse(predictions_logit >= 0.128, "yes", "no"))


conf_final <- confusionMatrix(predicted_response, test$Performance.Tag, positive = "yes")


acc <- conf_final$overall[1]


sens <- conf_final$byClass[1]


spec <- conf_final$byClass[2]


acc


sens


spec
```

#The End --