#1. Import the data files

train\_data= read.csv("C:\\Users\\HP\\Desktop\\train.csv")

print(train\_data)

#2. Data inspection and cleaning:

#1. For initial inspection of data, look at the structure of the data,

#check data types, and share your observations.

stuc\_td= str(train\_data)

#print(stuc\_td)

typ\_td= typeof(train\_data)

print(typ\_td)

#2. Some of the variables seem to have improper data types to identify the

#reason for the same and correct them.

#Because of data entry error R shows the wrong data type

train\_data$Age= as.numeric(train\_data$Age)

print(train\_data$Age)

print(class(train\_data$Age))

train\_data$SSN= as.integer(train\_data$SSN)

print(class(train\_data$SSN))

train\_data$Annual\_Income = as.numeric(train\_data$Annual\_Income)

print(class(train\_data$Annual\_Income))

train\_data$Num\_of\_Loan= as.integer(train\_data$Num\_of\_Loan)

print(class(train\_data$Num\_of\_Loan))

train\_data$Num\_of\_Delayed\_Payment= as.integer(train\_data$Num\_of\_Delayed\_Payment)

print(class(train\_data$Num\_of\_Delayed\_Payment))

train\_data$Changed\_Credit\_Limit= as.numeric(train\_data$Changed\_Credit\_Limit)

print(class(train\_data$Changed\_Credit\_Limit))

train\_data$Outstanding\_Debt = as.numeric(train\_data$Outstanding\_Debt)

print(class(train\_data$Outstanding\_Debt))

train\_data$Amount\_invested\_monthly= as.numeric(train\_data$Amount\_invested\_monthly)

print(class(train\_data$Amount\_invested\_monthly))

train\_data$Monthly\_Balance= as.numeric(train\_data$Monthly\_Balance)

print(class(train\_data$Monthly\_Balance))

sucu=str(train\_data)

print(sucu)

#3. Handling missing and trivial values in numerical columns

#1. Identify the outliers in the data using box plots. The outliers in the data

#can be considered trivial entries. These entries can be replaced with ‘NA’

#to be filled with group (Customer ID) specific values.

install.packages("ggplot2")

library(ggplot2)

for (x in train\_data) {

xd= train\_data[,x]

print(xd)

p= ggplot(train\_data, aes(x= group, y= value))+ geom\_boxplot()

}

print(p)

q1= quantile(xd, 0.25, na.rm= TRUE)

q3= quantile(xd, 0.75, na.rm= TRUE)

iqr = IQR(xd, na.rm= TRUE)

print(iqr)

ll= q1-(1.5\*iqr)

ul= q3+(1.5\*iqr)

train\_data[train\_data[,x]< ll & !is.na(train\_data[,x]) , x] = NA

train\_data[train\_data[ ,x]> ul & !is.na(train\_data[,x]) , x] = NA

#2. Fill in the missing values for numerical data with the median for

#each customer.

train\_data= train\_data %>%

group\_by(Customer\_ID) %>% mutate

(across(all\_of(train\_data),~ replace\_na

(.x, median(.x, na.rm = TRUE))))

print(train\_data)

#4. Handling missing and trivial values in character columns

char\_cols = names(which(sapply(train\_data, is.character)))

names(which(sapply(train\_data[char\_cols],

function(x)sum(trimws(x) == '')) >0))

#1. Character missing values may also be represented as an empty string or

#blank space or any other format. Identify the formats in each variable

#and replace them.

ad= sort(unique(train\_data$Occupation))

print(ad)

assumed\_missing = sort(unique(train\_data$Occupation))[1]

print(assumed\_missing)

train\_data[train\_data$Occupation == assumed\_missing,'occupation']= NA

train\_data = train\_data %>%

group\_by(Customer\_ID) %>%

mutate(Occupation =replace\_na(Occupation, min(Occupation, na.rm = TRUE) ))

print(unique(credit$Occupation))

#b. Credit history: split into years and months and create a column

#‘credit\_history\_age’ as the total months.

sep\_cr\_h= separate(Credit\_History\_Age, into = paste0('Credit\_History\_Age',

c('years', 'months')),

remove = TRUE, convert = TRUE, sep = "and",fill = 'right') %>%

mutate(across(starts\_with('Credit\_History\_Age'), parse\_number)) %>%

mutate(credit\_history\_age = if\_else(is.na(Credit\_History\_months ), 0,

Credit\_History\_months) +

if\_else(is.na(Credit\_History\_years),0, Credit\_History\_years))

#5. Feature engineering:

#1. Identify the unique loan types.

types\_of\_loan = paste(train\_data$Type\_of\_Loan, collapse = ", ")

t1 = trimws(strsplit(trimws(types\_of\_loan), "and")[[1]])

t2 = paste0(t1, collapse = '')

t3 = trimws(strsplit(t2, ",")[[1]])

unique\_loans = unique(ifelse(t3 =='' , "Not Specified", t3))

train\_data = train\_data %>% as.data.frame()

train\_data[train\_data$Type\_of\_Loan == "", "Type\_of\_Loan"] = 'Not Specified'

print(unique\_loans)

#2. Create a variable corresponding to each loan type with values of 1

#if the loan type is applicable and 0 if not.

string\_conv = function(x){

s1 = strsplit(x, 'and')[[1]]

s2 = trimws(paste(s1, collapse = ''))

s3 = trimws(strsplit(s2, ',')[[1]])

res = var %in% s3

names(res) = NULLreturn( as.numeric(res) )}

for (var in unique\_loans){

results = sapply(train\_data$Type\_of\_Loan, string\_conv)

names(results) = NULL

var = gsub(' ', '\_', var)

var = gsub('-', '\_', var)

train\_data[var] = results

}head(train\_data)

#6. Drop all rows with any missing values

train\_data = na.omit(train\_data)

#7. Exploratory data analysis:

#1. Study the probability distribution of numerical variables such as age,

#annual income, monthly balance, and monthly investments.

vars = c("Age", "Annual\_Income", "Monthly\_Balance",

"Outstanding\_Debt", "Amount\_invested\_monthly", "Monthly\_Balance")

for (var in vars){

title = gsub('\_', ' ', var)

p = ggplot(train\_data,aes\_string(x = var) ) +

geom\_histogram(aes(y = ..density..), color = 'white',

fill = 'steelblue')+

geom\_density(color = 'maroon' , lwd = 1)+

ggtitle(title)+

theme(plot.title = element\_text(hjust = 0.5, size = 20,

face = 'bold', color = 'Hotpink3',

family = 'Times New Roman'))

print(p) }

#2. What is the maximum number of delayed payments for customers?

train\_data %>%

group\_by(Customer\_ID) %>%

summarise(max\_delayed = max(Num\_of\_Delayed\_Payment)) %>%

arrange(desc(max\_delayed))

#3. Study the relationship between credit history age across credit score

#categories for customers.

ggplot(train\_data)+

geom\_density(aes(x = credit\_history\_age, color = Credit\_Score),

alpha = 0.05, lwd = 1.5)

#4. Study the mean annual income vs. outstanding debt for each customer,

#then share your observations.

train\_data%>%

group\_by(Customer\_ID)%>%

summarise(Annual\_Income= mean(Annual\_Income, na.rm= TRUE), Outstanding\_Debt = mean(Outstanding\_Debt, na.rm = TRUE)) %>%

ggplot(aes(x = reorder(Customer\_ID, -Annual\_Income),

y = Annual\_Income, fill = Outstanding\_Debt)) +

geom\_col() + scale\_fill\_gradient2(low = 'skyblue',

mid = 'green',

high = 'red', midpoint = 500) +

scale\_x\_discrete(breaks = NULL) +

labs(x = 'Customer', y = 'Income ($)', fill = 'Avg\nDebt ($)') +

theme\_bw()

#8. Data preparation for modeling:

#1. Identify and drop all unnecessary columns to create file data

names(train\_data)

#2. Identify the ‘Credit\_Score’ variable as a target and rename it for its

#further use.

final = train\_data%>%

select(-ID, -Customer\_ID, -Name, -SSN,-Credit\_Mix, -Credit\_History\_years,

-Credit\_History\_months,-Type\_of\_Loan, -Monthly\_Inhand\_Salary) %>%

rename(Target = Credit\_Score ) %>%

as.data.frame()

names(final)

#3. Transform the categorical columns into factor types.

fact\_cols = c("Month", "Occupation", "Payment\_Behaviour","Auto\_Loan" , "Credit\_Builder\_Loan", "Personal\_Loan", "Home\_Equity\_Loan",

"Not Specified", "Mortgage Loan","Student\_Loan", "Debt Consolidation Loan" ,

"Payday Loan", "Target")

char\_cols = names(which(sapply(final, is.character)))

vars = gsub(' ', '\_', unique\_loans)

loan\_cols = gsub('-', '\_', vars)

fact\_cols = c(char\_cols, loan\_cols)

final[fact\_cols] = lapply(final[,fact\_cols], as.factor)

row.names(final) = 1:nrow(final)

sapply(final, class)

#9. Predictive analytics:

#1. Split the data into training and testing in a 70:30 ratio.

set.seed(12)

train\_indices = sample(1:nrow(final),0.70 \*nrow(final))

train = final[train\_indices,]

test = final[-train\_indices,]

#2. Develop a base model using a decision tree classifier.

ctrl = rpart.control(minsplit = 2, minbucket = 1, cp = 0.001, xval = 5)

tree = rpart(Target~., data = train, control = ctrl)

rpart.plot(tree)

pred = predict(tree, test, type = 'class')

caret::confusionMatrix(pred, test$Target)

#3. Optimize the model by pruning. (Hint: Try to find the optimized control

#hyperparameters (minsplit, minibucket, cp, and others) and develop a pruned

#tree using them.

plotcp(tree)

print(tree$cptable)

ctrl = rpart.control(minsplit = 150, minbucket = 30, cp = 0.001)

tree\_prune = rpart(Target~., data = train, control = ctrl)

rpart.plot(tree\_prune)

#4. Evaluate the model using the confusion matrix and share your interpretation.

pred\_prune = predict(tree\_prune, test, type = 'class')

caret::confusionMatrix(pred\_prune, test$Target)