#1. Import the data files

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

customer\_id= read.csv("C:\\Users\\HP\\Desktop\\customer\_ID.csv")

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

print(internet)

print(customer\_id)

print(churn)

head(internet)

head(customer\_id)

head(churn)

#2. Check the structure of the data files and show the common column

str(internet)

str(customer\_id)

str(churn)

print('no. of unique Customer IDs in:')

paste('internet',length(unique(internet$customer\_ID)))

paste('customer\_id',length(unique(customer\_id$customer\_ID)))

paste('churn',length(unique(churn$customer\_ID)))

intersect(names(churn), intersect(names(internet), names(customer\_id)))

#3. Create combined data by merging the three data files

merged = merge(customer\_id,internet, by ="customer\_ID")

merged\_2 = merge(merged,churn, by ="customer\_ID")

data = merged\_2

rm(merged\_2)

rm(merged)

head(data)

#4. Do a preliminary statistical summary of this combined data

str(data)

summary(data)

#5. Identify all categorical columns and convert them to factor type

char\_cols = names(data)[(sapply(data, class) == "character")]

data[char\_cols] = lapply(data[char\_cols], factor)

sapply(data, class)

data$Senior\_Citizen = as.factor(data$Senior\_Citizen)

summary(data)

#6. Check for missing and trivial values

sapply(data, function(x) sum(is.na(x)))

#7. Remove the rows with missing values

data[is.na(data$TotalCharges),]

data\_no\_miss <- na.omit(data)

#8. Check for outliers in the numerical variables using a box plot and if any,

#remove them

install.packages("RColorBrewer")

library(RColorBrewer)

numerical\_cols =

names(data\_no\_miss)[(sapply(data\_no\_miss, class) != "factor")]

numerical\_cols

colors = brewer.pal(length(numerical\_cols), 'Set3')

i = 1

for (var in numerical\_cols){ boxplot(data[,var], main = var, horizontal = T,

col = colors[i] )

i = i+1 }

summary(data\_no\_miss[numerical\_cols])

#9. Study the overall distribution of the Churn variable

names(data\_no\_miss)[sapply(data\_no\_miss, class) == 'factor']

cnt = table(data\_no\_miss$Churn)

perc = round(cnt/sum(cnt) \* 100, 2)

colors = brewer.pal(length(cnt), 'Set3')

pie(cnt, labels = paste(names(cnt), '-', perc), col = colors,

main = 'Churn Distribution')

#10. Study the distribution of categorical variables

names(data\_no\_miss)[sapply(data\_no\_miss, class) == 'factor']

cat\_cols = c("gender", "Senior\_Citizen", "partner", "dependents",

"MultipleLines", "InternetService" , "OnlineSecurity",

"OnlineBackup", "DeviceProtection", "TechSupport", "StreamingTV",

"StreamingMovies", "PhoneService", "Contract", "PaperlessBilling"

,"PaymentMethod" )

#data\_churn <- data\_no\_miss[data\_no\_miss$Churn == 1,]

for (col in cat\_cols){

cnt = table(data\_no\_miss[,col], data\_no\_miss$Churn)

n = nrow(cnt)

barplot(cnt, main = col,

col = suppressWarnings(brewer.pal(n, 'Set3')),

legend = row.names(cnt))

}

#11. Remove unwanted variables, like Customer ID

data\_no\_miss = data\_no\_miss[,-1]

#12. In the variables related to internet service, there are three categories:

#Yes, No, and No Internet Service. Transform them to binary variables with No

#as no and No internet service and Yes otherwise. Do the same with the variable

#“Multiline”

for (col in intersect(cat\_cols, names(internet))){

temp = as.character(data\_no\_miss[,col])

ts = ifelse(temp == 'No internet service' | temp=="No phone service" ,

"No", temp)

data\_no\_miss[,col] = as.factor(ts) }

summary(data\_no\_miss)

#13. Follow the pointers below for model building and prediction:

#a. Divide the data into train and test as 80 – 20

'set.seed(12) tr\_indices' = sample(1:nrow(data\_no\_miss),

as.integer(0.8 \* nrow(data\_no\_miss)))

train = data\_no\_miss[tr\_indices,]

test = data\_no\_miss[-tr\_indices,]

names(train) = train$Churn

#b.Use the train data to develop a logistic regression model with

#cross-validation

fitctrl =

trainControl(method = "cv", number = 5,

classProbs = TRUE,)

logit\_mod = train(Churn~., data = train,

trControl = fitctrl, method = 'glm',

family = binomial("logit"))

#c.Check the summary of the Logit model and calculate the marginal effects of

#the dependent variables

best\_model\_logit = logit\_mod$finalModel

summary(best\_model\_logit)

margins(best\_model\_logit, data= best\_model\_logit$model )

#d.Evaluate the model using a test dataset with a confusion matrix and ROCR

#curve and discuss the results

pred\_logit = predict(logit\_mod, test, type = 'prob')

pred\_class = predict(logit\_mod, test, type = 'raw')

confusionMatrix(pred\_class, test$Churn, positive = 'Yes')

roc\_data = roc(test$Churn, pred\_logit[, 2], positive = 'Yes')

fpr\_tpr = data.frame(TPR = roc\_data$sensitivities, FPR = 1 - roc\_data$specificities)

plot(fpr\_tpr$FPR, fpr\_tpr$TPR, lwd = 3, type = 'l',

col = 'steelblue',

main = paste0('AUC Score = ',

round(auc(test$Churn, pred\_logit[, 2]),4)),

xlab = 'FPR', ylab = 'TPR')

abline(0, 1, col = 'orange', lwd = 2)

#e.Plot the fitted values from logit regression and the linear probability model

#and compare the results. Discuss the dependent and independent variables for

#regression and their relationship

plot(best\_model\_logit$fitted.values,

type="l", xlab="", ylab="",

main="Fitted vs Actual for Logit",

ylim=c(0,1), col="green")

churn\_no = ifelse(train$Churn == "Yes", 1, 0)

lines(churn\_no, type="p", col="red", pch=20)

legend("topright", c("Actual", "Fitted"),

fill=c("red", "Green"))

library(dplyr)

train\_linear = train %>% mutate(Churn = churn\_no)

linear\_mod = lm(Churn~., data = train\_linear)

summary(linear\_mod)

plot(linear\_mod$fitted.values,

type="l", xlab="", ylab="",

main="Fitted vs Actual for Linear Probability Model",

ylim=c(0,1), col="green")

lines(churn\_no, type="p", col="red", pch=20)

legend("topright", c("Actual", "Fitted"),

fill=c("red", "Green"))

#f.. Compare the model performance of the logit model and linear probability

#model with classification accuracy using threshold values of zero

prob\_Yes = pred\_logit[,2]

thresholds = c(0.2, 0.4, 0.6, 0.8) churn\_test = ifelse(test$Churn== "Yes", 1,0)

test\_linear = test %>% mutate(Churn = churn\_test)

linear\_pred = predict(linear\_mod, test\_linear)acc\_logit <- c()

acc\_linear = c()

for (thresh in thresholds){

class\_data = as.factor(ifelse(prob\_Yes >thresh, "Yes", "No"))

cm = confusionMatrix(class\_data, test$Churn, positive = 'Yes')

acc\_logit = c(acc\_logit,cm$overall["Accuracy"])

class\_lin = as.factor(ifelse(linear\_pred > thresh, "Yes", "No"))

cm\_lin = confusionMatrix(class\_lin, test$Churn,

positive = 'Yes')

acc\_linear = c(acc\_linear,cm\_lin$overall["Accuracy"]) }

data.frame(thresholds, acc\_logit, acc\_linear)