**Running notes, packages, codes that work for me**

Running list of packages and libraries I use and add any others I might need:

install.packages("caret")

library(caret)

library(dplyr)

install.packages("tidyverse")

library(tidyverse)

install.packages("cluster")

library(cluster)

install.packages("factoextra")

library(factoextra)

install.packages("cowplot")

library(cowplot)

library(ggplot2)

install.packages("tidyr")

library(tidyr)

library(dplyr)

install.packages("tidyverse")

library(tidyverse)

install.packages("cluster")

library(cluster)

library(readr)

library(tidyr)

install.packages("devtools")

library(devtools)

library(cluster)

install.packages("fpc")

library(fpc)

library(readr)

library(dplyr)

library(ggplot2)

install.packages("ggcorrplot")

library(ggcorrplot)

library(tidyr)

library(fastDummies)

library(caret)

install.packages("VIM")

library(VIM)

library(readr)

library(tidyverse)

library(caret)

library(pROC)

library(ggcorrplot)

library(gmodels)

library(rpart)

Code notes:

Apply past learned codes

glimpse(train)

summary(train)

converting character variables to factors

train %>% mutate(cut = as.factor(cut), color = as.factor(color), clarity <- as.factor(clarity))

bar plots

ggplot(train, aes(x=cut, fill = cut)) + geom\_bar() + theme\_classic() + labs(title="Various types of diamond cuts", x="Cut categories", y = "Count")

ggplot(train, aes(x=clarity, fill = clarity)) + geom\_bar() + theme\_classic() + labs(title="Various types of diamond clarity levels", x="diamond clarity levels", y = "Count")

histogram plots

ggplot(train, aes(x = depth)) + geom\_histogram(fill = 'blue', bins=100) + labs(x="depth", y="Count",title = "Probability Distribution of depth") + theme\_classic()

ggplot(train, aes(x = log(carat))) + geom\_histogram(fill = 'blue', bins=100) + labs(x="carat", y="Count",title = "Probability Distribution of carat") + theme\_classic()

is.na?

apply(train,2,function(x){any(is.na(x))})

is there any correlation to the variables?

train\_cor <- round(cor(train %>% select\_if(is.numeric)), 1)

ggcorrplot(train\_cor, title = "Correlation", type = "lower") + theme(plot.title = element\_text(hjust = 0.5), axis.text.x = element\_text(angle = 90))

x, y, z is very correlated to each other and also it's very correlated with caret variable --- remove from dataset

train <- train %>% select(-c(x,y,z))

box plot for all numeric variables

train %>% select\_if(is.numeric) %>% mutate\_all(scale) %>% gather("features","values") %>% na.omit() %>%

ggplot(aes(x = features, y = values)) + geom\_boxplot(show.legend = FALSE) + stat\_summary(fun = mean, geom = "point", pch = 1) +

scale\_y\_continuous(name = "Variable values", minor\_breaks = NULL) + scale\_fill\_brewer(palette = "Set1") + coord\_flip() + theme\_minimal() + labs(x = "Variable names") + ggtitle(label = "Distribution of numeric variables in diamond train dataset")

creating a dummy

train\_d <- dummy\_cols(train)

train\_d <- train\_d %>% select(-c(cut, color, clarity))

Splitting dataset into training (60%) and validation (40%) sets

set.seed(23)

index <- createDataPartition(train\_d$price, p=0.6, list = FALSE)

train\_df <- train\_d[index,]

test\_df <- train\_d[-index,]

Defining a function to normalize the data

scale\_fun <- preProcess(train\_df %>% select(-price), method = c("center", "scale"))

train\_norm <- predict(scale\_fun, train\_df)

test\_norm <- predict(scale\_fun, test\_df)

Building a model to estimate the diamond price value

diamond\_train\_model <- lm(price ~ . , data = train\_norm)

Performance metrics on test data

RMSE on test data

(linear\_base\_rsme <- sqrt( mean(( test\_norm$price - predict( diamond\_train\_model, test\_norm))^2)))

R squared on test data

(linear\_base\_rsquare <- cor( test\_norm$price, predict( diamond\_train\_model, test\_norm))^2)

**Customer Churn Group Project**

# Loading the churn dataset

Churn\_Train <- read\_csv("Churn\_Train.csv")

# Creating a copy of the file for working

Churn\_Data <- read\_csv("Churn\_Train.csv")

# Inspecting data

head(Churn\_Data)

# Examining the dataset

glimpse(Churn\_Data)

# Summary statistics of dataset

summary(Churn\_Data)

# From glimpse we can see that, Some of the character variables can be converted into factors, So Converting character variables to factors.

Churn\_Data <- Churn\_Data %>% mutate\_if(is.character, as.factor)

# Checking NULL values in the dataset at column level.

colSums(is.na(Churn\_Data))

# imputation of missing values - median imputation technique – The model being built using only numeric values. – impute applies the median value of the column to any NA values – this is instead of the KNN approach – both approaches will work, I’ve only used it because I could never get the KNN to work for me

imputation\_model <- preProcess(Churn\_Data %>% select\_if(is.numeric),method = "medianImpute")

data <- predict(imputation\_model, Churn\_Data %>% select\_if(is.numeric))

Churn\_Data <- Churn\_Data %>% select(setdiff(names(Churn\_Data), names(data))) %>% cbind(data)

# Box plot - to detect the outliers

Churn\_Data %>% select\_if(is.numeric) %>% mutate\_all(scale) %>% gather("features","values") %>% na.omit() %>%

ggplot(aes(x = features, y = values)) +

geom\_boxplot(show.legend = FALSE) +

stat\_summary(fun = mean, geom = "point", pch = 1) + # Add average to the boxplot

scale\_y\_continuous(name = "Variable values", minor\_breaks = NULL) +

scale\_fill\_brewer(palette = "Set1") +

coord\_flip() +

theme\_minimal() +

labs(x = "Variable names") +

ggtitle(label = "Distribution of numeric variables in Churn dataset")

# Visualizing distribution of Churn categorical variable.

ggplot(Churn\_Data, aes(x=churn, y=..prop..,group = 1)) +

geom\_bar(fill="light blue") +

theme\_classic() +

geom\_text(aes(label=round(..prop..,2)),stat = "count",

position = position\_stack(vjust=0.5)) +

labs(y = 'Proportion', title = "Proportion of churn") +

scale\_x\_discrete(labels = c("No","Yes"))

# finding correlation between variables

Churn\_Data\_cor <- round(cor(Churn\_Data %>% select\_if(is.numeric)), 1)

ggcorrplot(Churn\_Data\_cor, title = "Correlation", type = "lower") +

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

axis.text.x = element\_text(angle = 90))

# Total minutes and total charge for the day, evening, night, and international are strongly linked, we can deduce. -- change categorical variable from factor to numeric – no probabilities with state column so dropping the state column

Churn\_Data <- Churn\_Data %>% select(-state, -churn) %>%

fastDummies::dummy\_cols(., remove\_selected\_columns = TRUE) %>% mutate(state = Churn\_Data$state, churn = Churn\_Data$churn)

# Pre-Processing of data

# Splitting dataset into training (80%) and validation (20%) sets

# Set seed is random number it makes it consistent so the code won’t change with each time it is run – basically standardizes it for future iterations of the same data otherwise each time you run it, it will have different results

set.seed(12)

index <- createDataPartition(Churn\_Data$churn, p=0.8, list=FALSE)

Churn\_Data\_train\_df <- Churn\_Data[index,]

Churn\_Data\_test\_df <- Churn\_Data[-index,]

# scaling the data

scaling <- preProcess(Churn\_Data\_train\_df %>% select\_if(is.numeric), method = c("center", "scale"))

Churn\_Data\_train\_norm <- predict(scaling, Churn\_Data\_train\_df %>% select\_if(is.numeric))

Churn\_Data\_test\_norm <- predict(scaling, Churn\_Data\_test\_df %>% select\_if(is.numeric))

Churn\_Data\_train\_norm$churn <- Churn\_Data\_train\_df$churn

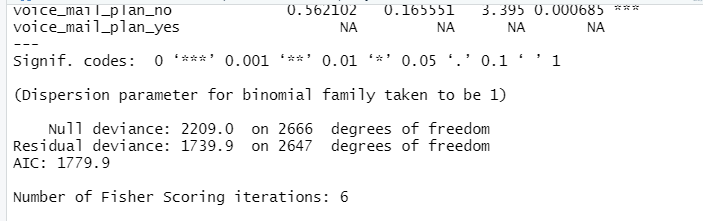
Churn\_Data\_test\_norm$churn <- Churn\_Data\_test\_df$churn

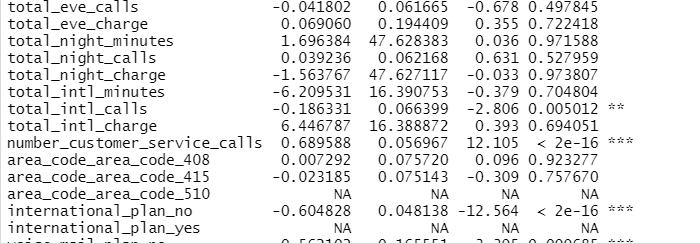
# Model Construction

Model\_1 <- glm(churn ~ ., data = Churn\_Data\_train\_norm , family= "binomial")

summary(Model\_1)

# Number of Fisher Scoring iterations: 6 – is indicated by the asterisk - there are 6 variables with the asterisk - these indicate that there is statistical significance to the probability of churn – the number of asterisks is the degree of significance – see screenshot below





# Predict values using based on Model\_1.

pred\_probs <- predict(object = Model\_1,Churn\_Data\_test\_norm, type = "response")

--- 

# warning message is calling out the Outliers – can ignore the warning

# Assigning labels based on probability prediction

Model\_Pre\_lables <- as.factor(ifelse(pred\_probs>0.6 ,"yes","no"))

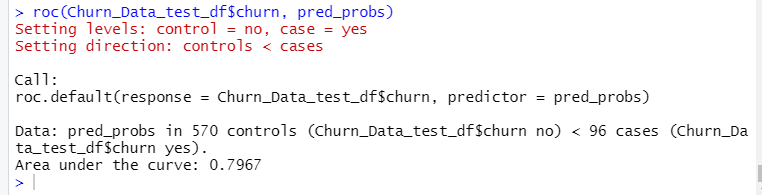
# Performance Metrics

# Confusion matrix for significant variable model.

confusionMatrix(Model\_Pre\_lables,Churn\_Data\_test\_norm$churn)

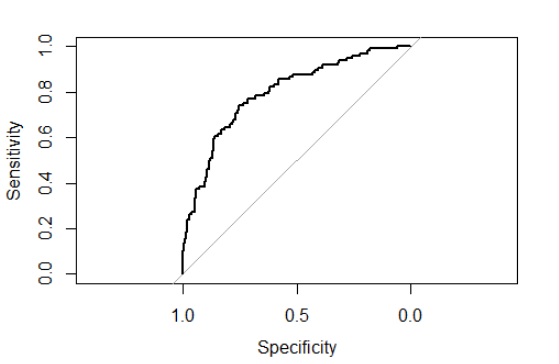
# AUC of the churn model ---- Area under the curve – this model indicates an 80% accuracy of the model

roc(Churn\_Data\_test\_df$churn, pred\_probs)



# This visualizes the sensitivity of the model

plot.roc(roc(Churn\_Data\_test\_df$churn, pred\_probs))



**Applying the model to the Customers to Predict data file**

# Load the data file

# the below address is specific to me and where I housed the file

load("C:/Users/xlamo/Desktop/XanLamoreux/Group Project/Customers\_To\_Predict.RData")

# creating a copy to work with

customer\_predict <- Customers\_To\_Predict

# removing the state column as it is not necessary

customer\_predict <- customer\_predict %>% select(-state) %>%

fastDummies::dummy\_cols(., remove\_selected\_columns = TRUE)

# Transformation for scaling the data (Z score transformation)

customer\_predict <- as.data.frame(scale(customer\_predict))

#predicting the model with the test data --- using the Model\_1 file created earlier

predict\_labels <- predict(object=Model\_1,customer\_predict,type="response")

# applies the probability ratio if under 60% customer will not churn

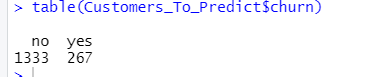
Model\_Pre\_lables\_2 <- as.factor(ifelse(predict\_labels>0.6 ,"yes","no"))

# adding chrun column and attaching the predictor from the model

Customers\_To\_Predict <- Customers\_To\_Predict %>% mutate(churn=Model\_Pre\_lables\_2)

# visual of the results which shows that 267 will churn

table(Customers\_To\_Predict$churn)



View(Customers\_To\_Predict)

# The Customers\_To\_Predict file can be exported as the final results of our model