A Data-Driven Approach to Improve Telemarketing in Banking

DATA 606: Project Final Report

Group 4 - Daksh Patel, Jacques Botha, Shrivarshini Balaji, Tomasz Szymczyk

June 17, 2023

Contents

```r
#Loading required packages:
library(readr)
library(dplyr)
library(regclass)
library(MASS)
library(ISLR)
library(klaR)
library(tree)
library(VGAM)
library(ggplot2)
library(ggsci)
library(corrplot)
library(grid)
library(gridExtra)
library(caret)
library(car)
library(tidyverse)
library(ROSE)
library(Rfast)
library(nnet)
library(vcdExtra)
library(energy)
library(QuantPsyc)
```

```r
options(repos = c(CRAN = "https://cloud.r-project.org/"))
```

1. Introduction

Marketing plays a key role in business, banks included. It allows businesses to raise awareness of the products or services they offer, differentiate their products from competitors by educating their customers or potential customers. Most importantly it can allow a business to grow their customer base and earn higher profits.

Our data set which was obtained through direct marketing (telemarketing) of a Portuguese banking institution contains a variety of information on those who participated in the calls. The information gathered ranges from things such as; age, types of loans the person has (home or auto), marital status and education level among many others. There is immense value in being able to predict what demographic is more likely to subscribe to a certain product or service a business may offer. It would allow business to target marketing campaigns more efficiently for certain groups of individuals.

Our group's goal is to build a binary classification model with our data set to gain a better understanding of what type of demographic is more likely to open a term deposit than not. We also want to evaluate how accurate our models are at making these predictions.

2. Data set

The data set was obtained from the UCI Machine Learning Repository. It is a public data set that can be used for research [1][2]. The data set consists of 45,211 observations with 16 explanatory variables (5 numeric and 11 categorical variables).

Table 1: Dataset Column Names and Descriptions

Column Name	Description
age	Age of the person being called (years)
job	Type of job
marital	Marital Status
education	Level of education achieved
default	Has credit default? (no, yes, unknown)
balance	Average yearly balance in euros
housing	Has housing loan? (no, yes, unknown)
loan	Has personal loan? (no, yes, unknown)
contact	Contact communication type (cellular, telephone)
day	Last contact day of the month
month	Last contact month of year
duration	Last contact duration (seconds)
campaign	Number of contacts performed during this campaign and for this client
pdays	Number of days that passed by after the client was last contacted from a previous campaign (-1 means client was not previously contacted)
previous	Number of contacts performed before this campaign and for this client
poutcome	Outcome of the previous marketing campaign (failure, success, unknown, other)
y	**Response Variable** - Has the client subscribed to a term deposit (yes, no)

```
#Import the data set
banking_marketing_dataset<-read_delim("~/Desktop/banking_marketing_dataset.csv",
delim=";",escape_double=FALSE,trim_ws=TRUE)
```

```
## Rows: 45211 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (10): job, marital, education, default, housing, loan, contact, month, p...
## dbl  (7): age, balance, day, duration, campaign, pdays, previous
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Basic descriptive statistics for each:
summary(banking_marketing_dataset)
```

```
##       age            job              marital           education
##  Min.   :18.00   Length:45211       Length:45211       Length:45211
##  1st Qu.:33.00   Class :character   Class :character   Class :character
##  Median :39.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :40.94
##  3rd Qu.:48.00
##  Max.   :95.00
##    default            balance         housing              loan
##  Length:45211       Min.   : -8019   Length:45211       Length:45211
##  Class :character   1st Qu.:    72   Class :character   Class :character
##  Mode  :character   Median :   448   Mode  :character   Mode  :character
##                     Mean   :  1362
##                     3rd Qu.:  1428
```

```
##                     Max.   :102127
##    contact               day           month              duration
##  Length:45211       Min.   : 1.00   Length:45211       Min.   :   0.0
##  Class :character   1st Qu.: 8.00   Class :character   1st Qu.: 103.0
##  Mode  :character   Median :16.00   Mode  :character   Median : 180.0
##                     Mean   :15.81                      Mean   : 258.2
##                     3rd Qu.:21.00                      3rd Qu.: 319.0
##                     Max.   :31.00                      Max.   :4918.0
##     campaign          pdays          previous          poutcome
##  Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000   Length:45211
##  1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000   Class :character
##  Median : 2.000   Median : -1.0   Median :  0.0000   Mode  :character
##  Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
##  3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :63.000   Max.   :871.0   Max.   :275.0000
##       y
##  Length:45211
##  Class :character
##  Mode  :character
##
##
##
```

2.1 Data Wrangling

This data set at first glance is very easy to work with and doesn't require much processing to work with. We will check for null data and scrub it from the data frame prior to doing any analysis.

```
#Remove nulls where appropriate
banking_marketing_dataset<-na.omit(banking_marketing_dataset)
```

```
#In order to help some some of our visuals in the Exploratory Data Analysis,
#we binned our age groups:
age_breaks<-c(17,25, 30, 40, 50, 60, Inf)
age_labels<-c("18-25", "26-30", "31-40", "41-50", "51-60", "61+")
banking_marketing_dataset$age_group<-cut(banking_marketing_dataset$age,
breaks=age_breaks,labels=age_labels)

#We also binned the "duration" of calls:
duration_breaks<-c(-1,180,360,600,900,1200,1500,Inf)
duration_labels<-c("0-3 min","3-6 min","6-10 min","10-15 min",
"15-20 min", "20-25 min", "25+ min")
banking_marketing_dataset$duration_group<-cut(banking_marketing_dataset$duration,
breaks=duration_breaks,labels=duration_labels)
```

3. Guiding Question

Based on the data set, the objective of the project is to analyze the factors affecting a customer's choice to open a bank account. The best model will be chosen from the candidate models constructed by different statistical approaches. This model is aimed to detect if the person approached by the bank will be becoming their permanent customer. Moreover, the data set contains the person's personal past banking choices as

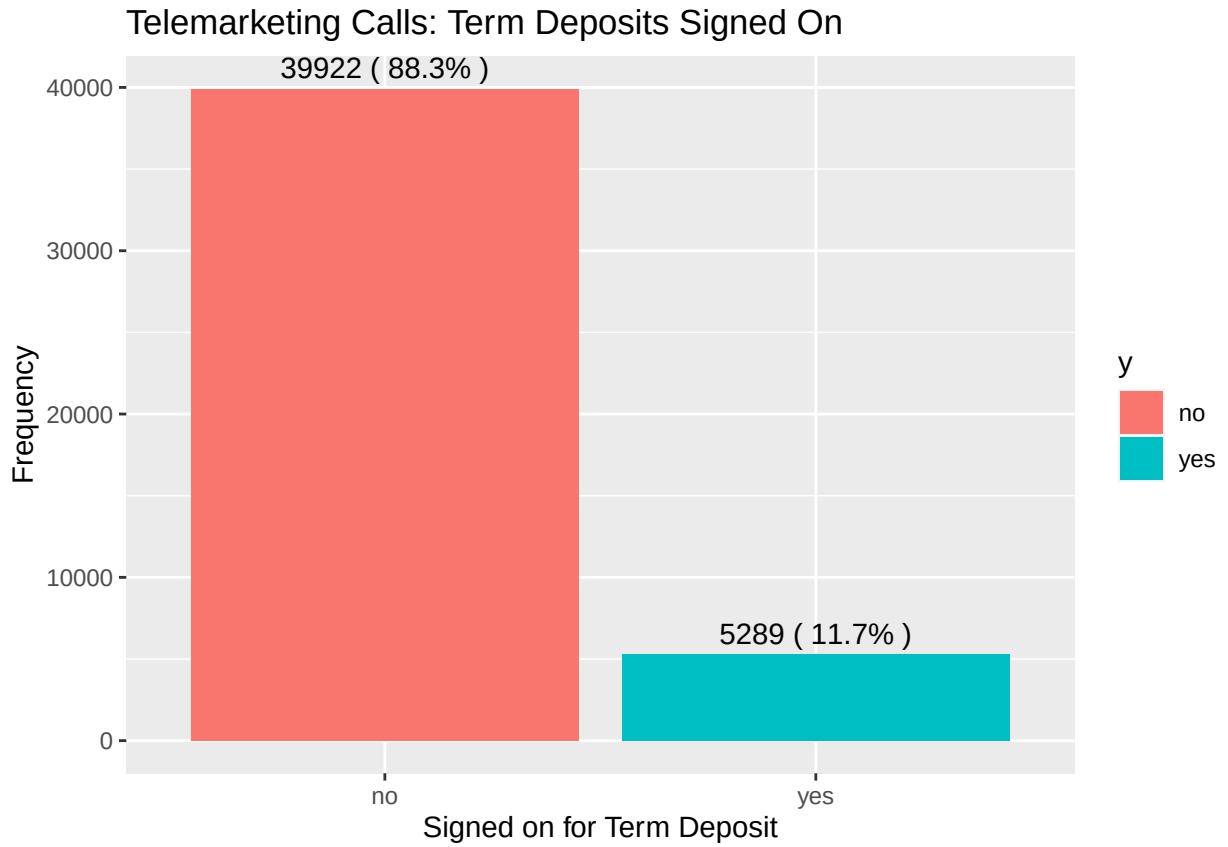
well as the marketing approach taken by the bank at the time. This will give us a broader view towards the variables correlating with our response variable. This study will help the banking institution to devise a customer tailored marketing experience.

4. Exploratory Data Analysis

4.1 EDA on Individual Features

In order to better understand the data, some preliminary exploratory data analysis was completed to get a better understanding of what this data set contains and potential pitfalls within it.

```
ggplot(banking_marketing_dataset, aes(x = y, fill=y))+geom_bar()+
geom_text(stat='count',aes(label= paste(after_stat(count),
"(",scales::percent(after_stat(count/sum(count)),accuracy=0.1),")")),
vjust=-0.5)+labs(x="Signed on for Term Deposit",y="Frequency",
title="Telemarketing Calls: Term Deposits Signed On")
```



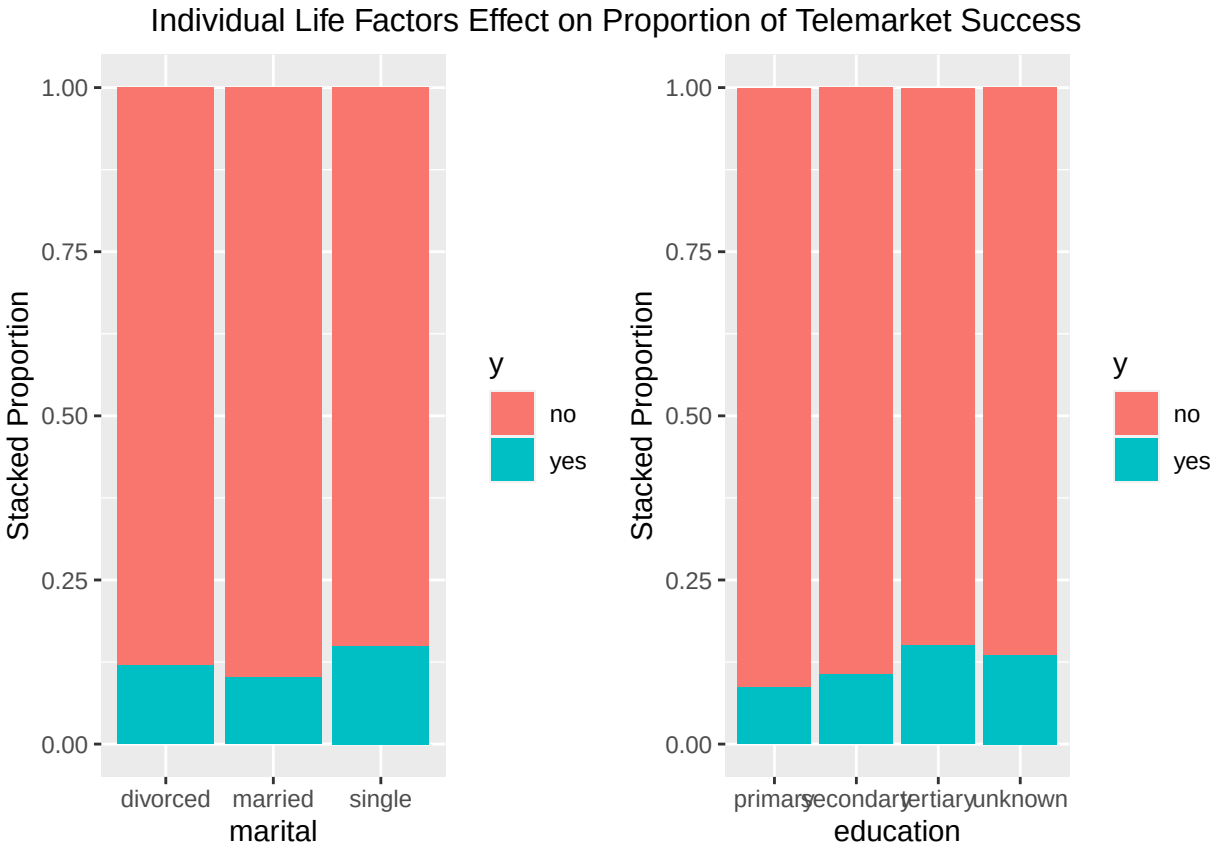
First we look into just how often a call is actually successful. Within our data set, 11.7% of calls result in a client signing up for a term deposit. Given that, we would expect our models to be highly accurate considering this project's response variable is binary.

```
#Grouping and Aggregating Data (Calculating mean balance by job):
mean_balance_by_job<- banking_marketing_dataset%>%group_by(job)%>%
  summarize(mean_balance = mean(balance))
arrange(mean_balance_by_job,desc(mean_balance))
```

```
## # A tibble: 12 x 2
##    job           mean_balance
##    <chr>                <dbl>
##  1 retired              1984.
##  2 unknown              1772.
##  3 management           1764.
##  4 self-employed        1648.
##  5 unemployed           1522.
##  6 entrepreneur         1521.
##  7 housemaid            1392.
##  8 student              1388.
##  9 technician           1253.
## 10 admin.               1136.
## 11 blue-collar          1079.
## 12 services              997.
```

The mean of balances across jobs is rather surprising, containing a spread of about 1,000 euros from the highest mean balance to lowest mean balance. As such, initially we do not expect it to play a big factor in determining whether or not a person signs up for a term deposit.

```
#Compare factors related to an individuals personal life situation

#Create individual ggplot objects
plot1<-ggplot(banking_marketing_dataset,aes(x=marital,fill=y))+
geom_bar(position= "fill")+labs(y="Stacked Proportion")

plot2<-ggplot(banking_marketing_dataset,aes(x=education,fill=y))+
geom_bar(position="fill")+labs(y="Stacked Proportion")

#Arrange the plots using grid.arrange and add a title
grid.arrange(plot1,plot2,nrow=1,
top=textGrob("Individual Life Factors Effect on Proportion of Telemarket Success"))
```



Personal life situation:

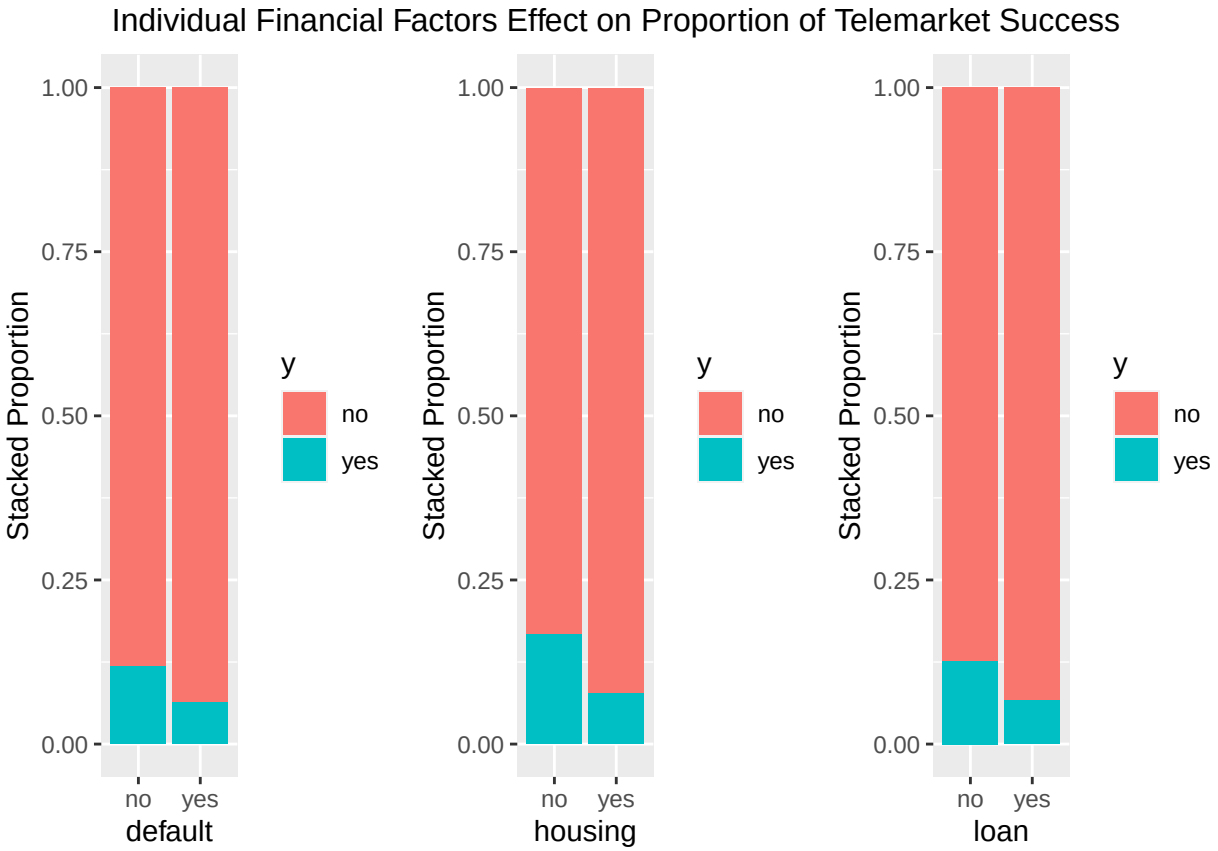
At a high level, for the main categorical variables the order of proportions (from highest to lowest) that people sign up for term deposits are:

a) Marital status - single > divorced > married

b) Education - tertiary > unknown > secondary > primary

These seem reasonable given that single people and more educated people likely have more disposable income to sign up for term deposits with.

```
# Compare factors related to an individuals financial situation

# Create individual ggplot objects
plot1<-ggplot(banking_marketing_dataset,aes(x=default,fill=y))+
geom_bar(position="fill")+labs(y="Stacked Proportion")

plot2<-ggplot(banking_marketing_dataset,aes(x=housing,fill=y))+
geom_bar(position="fill")+labs(y="Stacked Proportion")

plot3<-ggplot(banking_marketing_dataset,aes(x=loan,fill=y))+
geom_bar(position="fill")+labs(y="Stacked Proportion")

#Arrange the plots using grid.arrange and add a title
grid.arrange(plot1,plot2,plot3,nrow=1,
top=textGrob("Individual Financial Factors Effect on Proportion of Telemarket Success"))
```



Financial standing categories:

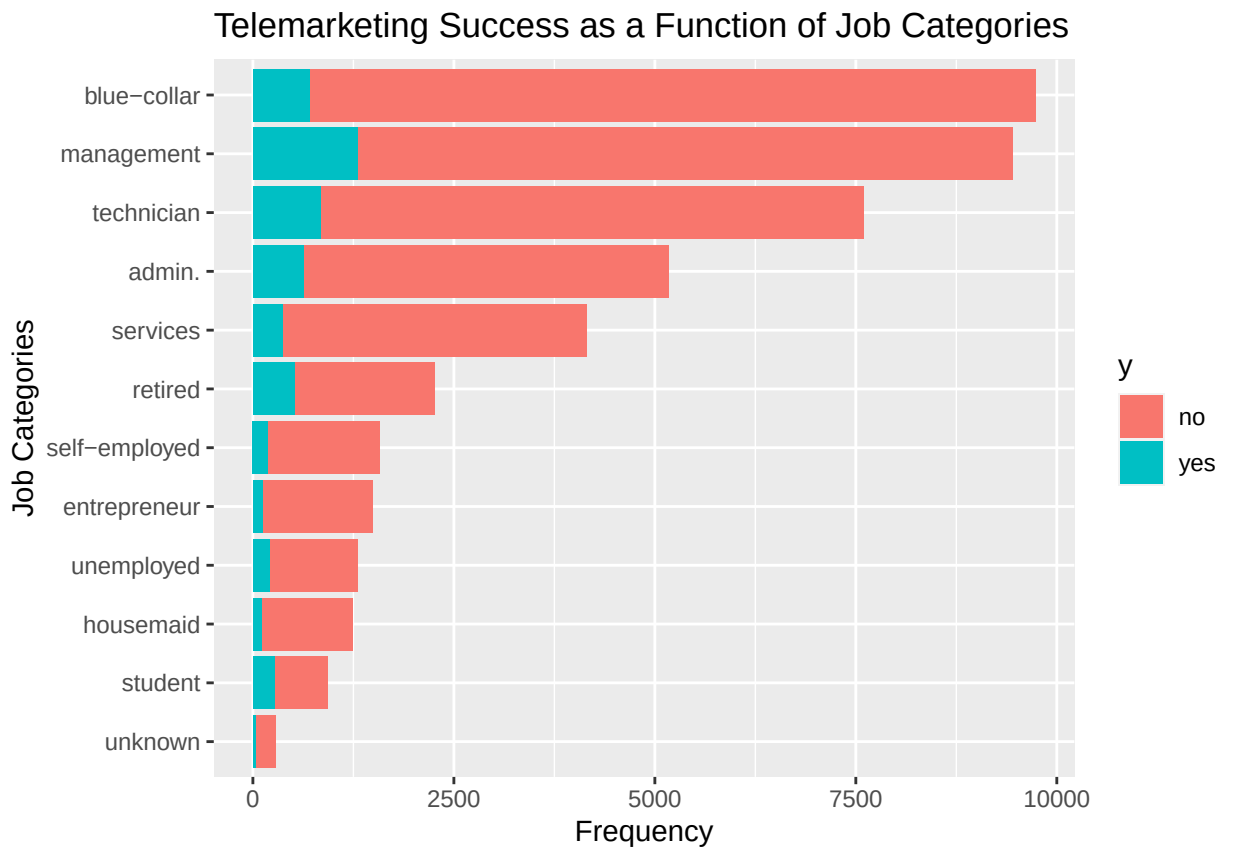
a) Has previously defaulted? Unsurprisingly, if someone defaulted before, they're less likely to sign up for a term deposit.

b) Has a housing loan? If someone has a housing loan, they're less likely to sign up for a term deposit. Intuitively this makes sense, as those who have a mortgage typically will prioritize paying the mortgage off before investing the money in a savings account.

c) Has a personal loan? if someone has a personal loan, they're again less likely to sign up for a term deposit. For very a very similar reason we would expect someone with a mortgage to not sign up for a savings account we would expect the same for someone with a loan.

The Job Category has a lot of classes in it, so that will be looked at separately for better visibility.

```
#Compare job types and to see what is the most represented

banking_marketing_dataset %>%count(job, y) %>%mutate(job = reorder(job, n)) %>%
ggplot(aes(x=job,y=n,fill=y)) +geom_bar(stat="identity")+coord_flip()+
labs(x="Job Categories",y="Frequency",
title="Telemarketing Success as a Function of Job Categories",fill="y")
```



Roughly 70% of our data set consists of people with that work blue-collar, management, and technician jobs. Generally speaking, it might be reasonable to assume that banks are targeting individuals with stables jobs that have a moderate to good income. Something else worth mentioning is that it would a appear that those in management positions seem much more likely to open a savings account more so than those working a blue collar job.

```
# Investigate whether or not the duration of the call appears to have any impact
#on the term deposit sign up rate

percent_by_duration <- banking_marketing_dataset %>%group_by(duration_group)%>%
summarize(total_count = n(),yes_count = sum(ifelse(y == "yes", 1, 0))) %>%
mutate(percentage_yes = round((yes_count / total_count) * 100, 1))
percent_by_duration
```

```
## # A tibble: 7 x 4
##   duration_group total_count yes_count percentage_yes
##   <fct>                <int>     <dbl>          <dbl>
## 1 0-3 min              22660       709            3.1
## 2 3-6 min              13197      1534           11.6
## 3 6-10 min              5564      1213           21.8
## 4 10-15 min             2372       993           41.9
## 5 15-20 min              858       494           57.6
## 6 20-25 min              333       208           62.5
## 7 25+ min                227       138           60.8
```

The result above shows that the longer a telemarketer kept someone on the phone, the more successful they were in getting that person to sign up for a savings account.

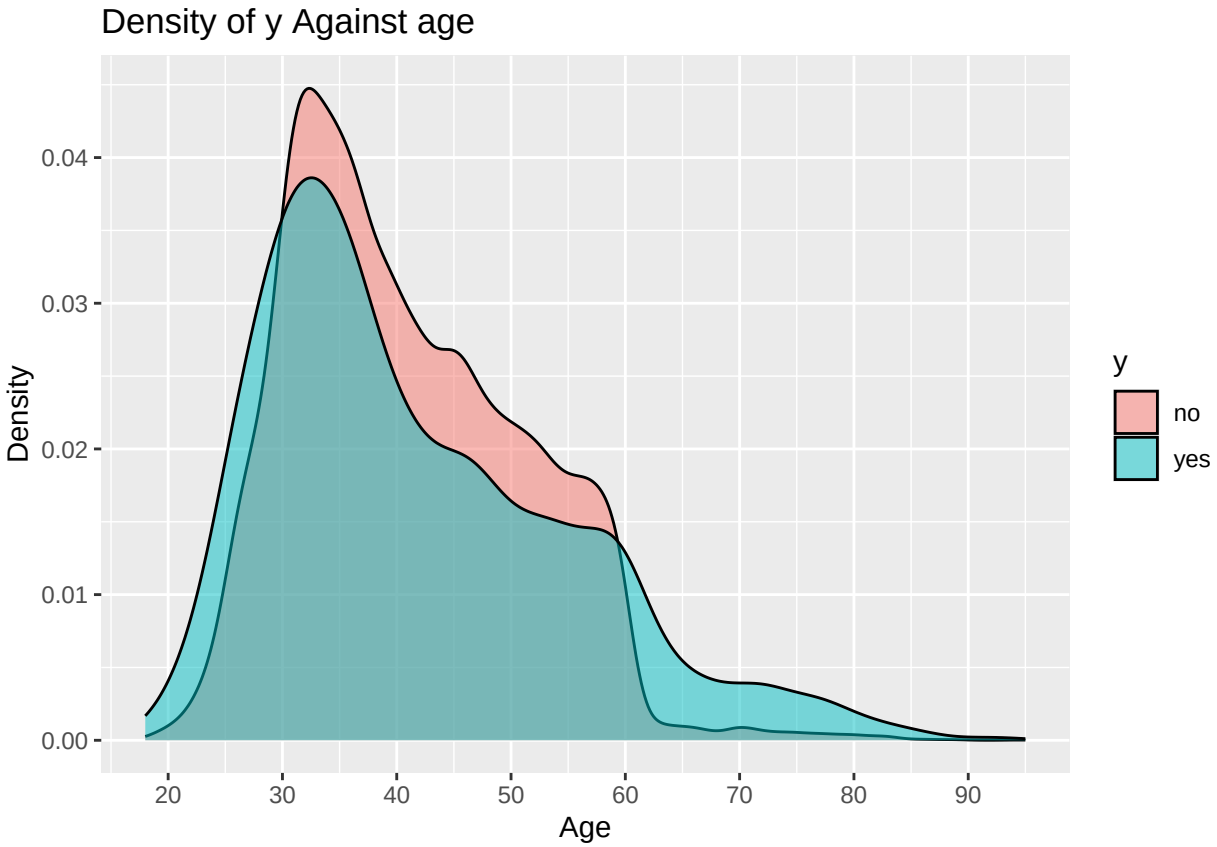
If we had to look at the distribution around when calls are made by month and the success rate of those calls, we can see below that a large percentage of calls are made in the month of May. However, even though less calls were made in the months of December and March, they appear to have a higher success rate during these months.

```
percent_by_month <- banking_marketing_dataset %>%group_by(month)%>%
summarize(total_count = n(),yes_count = sum(ifelse(y == "yes", 1, 0))) %>%
mutate(percentage_yes = round((yes_count / total_count) * 100, 1))
percent_by_month
```

```
## # A tibble: 12 x 4
##    month total_count yes_count percentage_yes
##    <chr>       <int>     <dbl>          <dbl>
##  1 apr          2932       577           19.7
##  2 aug          6247       688           11
##  3 dec           214       100           46.7
##  4 feb          2649       441           16.6
##  5 jan          1403       142           10.1
##  6 jul          6895       627            9.1
##  7 jun          5341       546           10.2
##  8 mar           477       248           52
##  9 may         13766       925            6.7
## 10 nov          3970       403           10.2
## 11 oct           738       323           43.8
## 12 sep           579       269           46.5
```

The next two plots will investigate the impact of age on the response variable y:

```
# Create density plots
ggplot(banking_marketing_dataset,aes(x=age,fill=y))+geom_density(alpha=0.5)+
labs(x="Age",y="Density",title = "Density of y Against age")+
scale_x_continuous(breaks=seq(0,100,by=10))
```



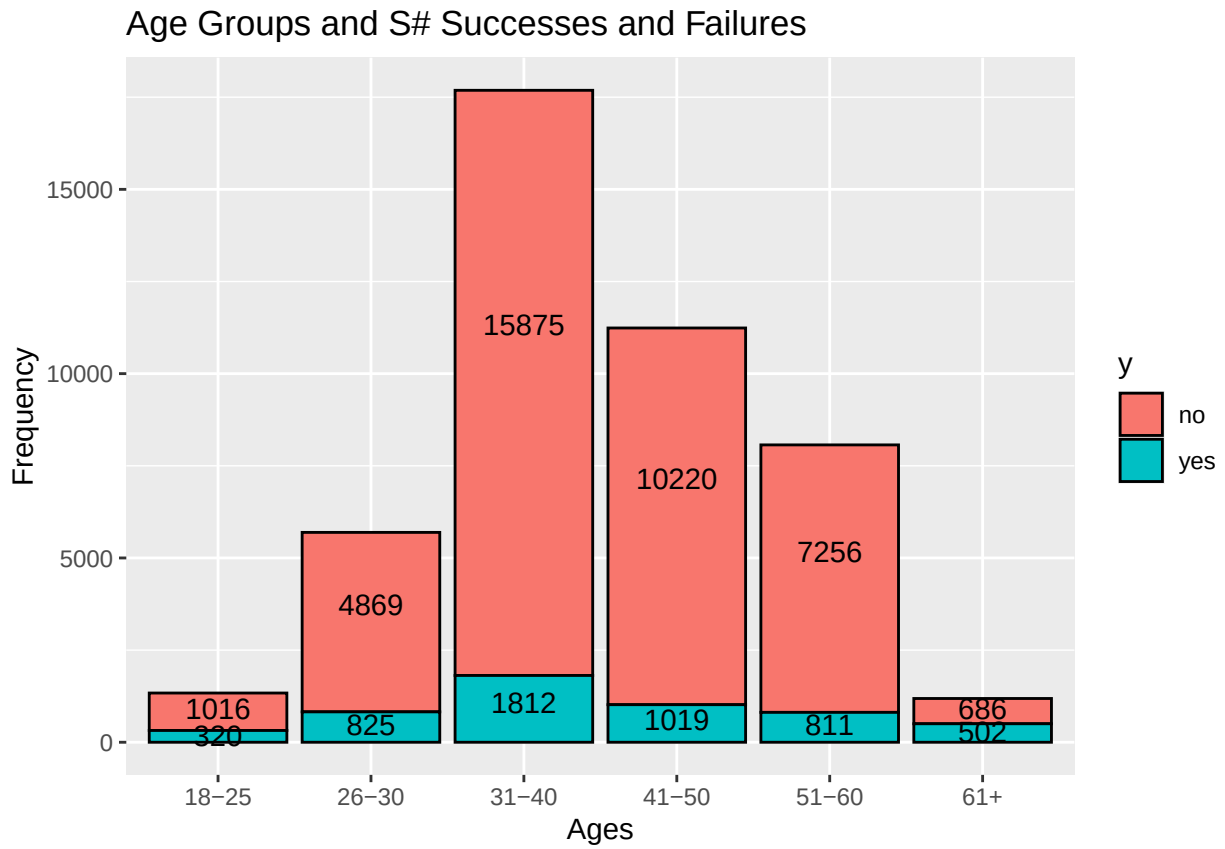
This plot highlights that the extremities where people are <25 years old and >60 years old tend to sign up for term deposits more often than those in between. This is also corroborated in the probability table below.

```
#Create fixed segments for age
age_segments<-cut(banking_marketing_dataset$age,
breaks=c(18,26,31,41,51,61,max(banking_marketing_dataset$age)+1),
labels=c("18-25","26-30","31-40","41-50","51-60","61+"),include.lowest = TRUE)

#Create a contingency table with the variables "age segments" and "y"
contingency_table<-table(age_segments,banking_marketing_dataset$y)

#Calculate the row proportions as percentages
row_proportions<-prop.table(contingency_table,margin=1)*100

#Calculate the counts of "yes" and "no" instances
count_table<-addmargins(contingency_table,margin=2)

#Combine row proportions and count table
combined_table<-cbind(row_proportions,count_table)
combined_table
```

```
##             no       yes    no  yes   Sum
## 18-25 78.79496 21.205044  1687  454  2141
## 26-30 86.97168 13.028322  5988  897  6885
## 31-40 89.83630 10.163703 15256 1726 16982
## 41-50 91.03271  8.967291  9908  976 10884
```

```
## 51-60 89.13163 10.868370  6487  791  7278
## 61+   57.25264 42.747358   596  445  1041
```

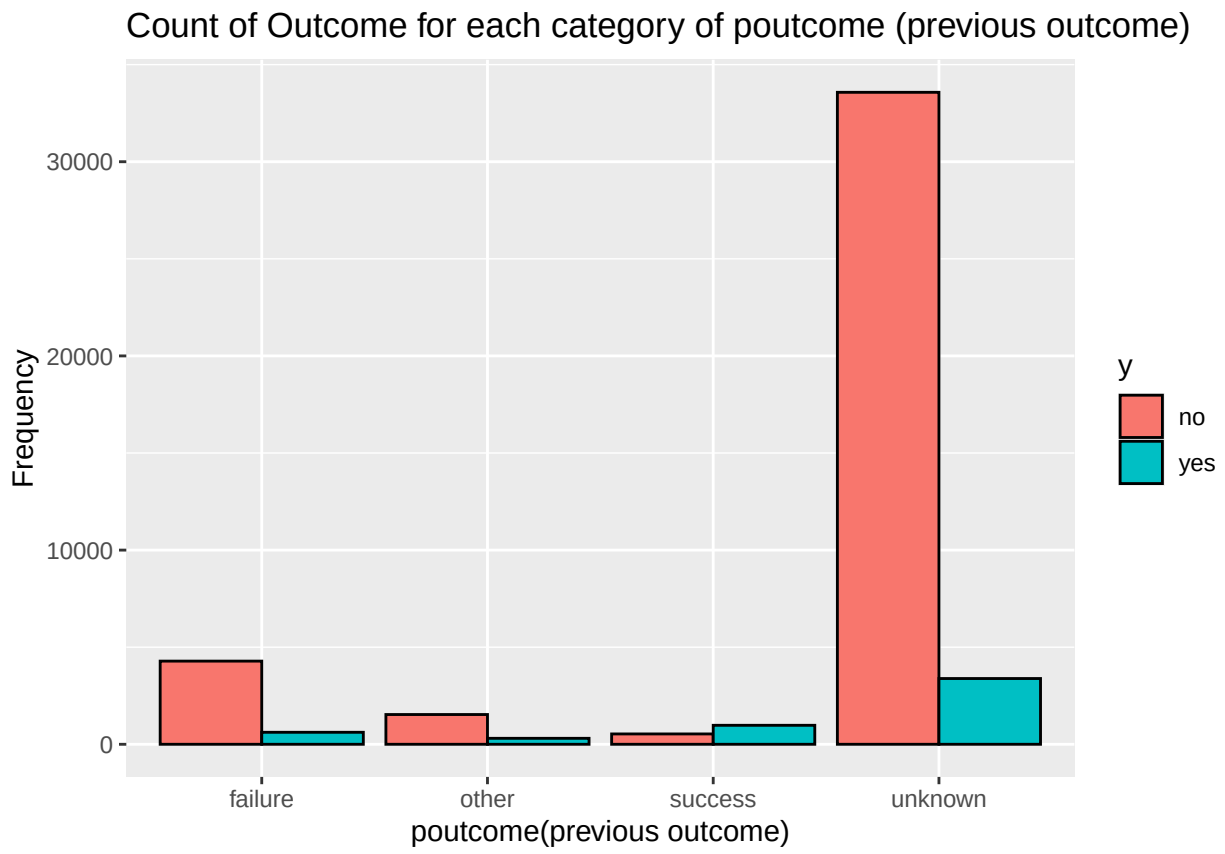
```
# Create a histogram of custom age group categories:

ggplot(banking_marketing_dataset,aes(x=age_group,fill=y))+
geom_bar(color="black")+geom_text(stat='count',aes(label = after_stat(count)),
position = position_stack(vjust = 0.6),color="black")+labs(x="Ages",
y="Frequency",title="Age Groups and S# Successes and Failures")
```



The most targeted age groups are roughly middle aged people (31-40, 41-50). From a logical perspective, it makes sense to target given that these individuals are in the middle of their careers and more established. These individuals likely have disposable income. However, as shown by the density plot above and contingency/probability table the most successful calls are actually to the youngest and oldest individuals.

Perhaps the missing feature in this data set is how much or how long of a term deposit an individual signed up for. It's possible that young/old people sign up for low amounts or times, while middle-aged people get higher value term deposits that make the bank more profit.

```
# Investigate poutcome feature to see how prevalent it is:

ggplot(banking_marketing_dataset,aes(x=poutcome,fill=y))+
geom_bar(color="black",position="dodge")+
labs(x="poutcome(previous outcome)",
y="Frequency",
title="Count of Outcome for each category of poutcome (previous outcome)")
```

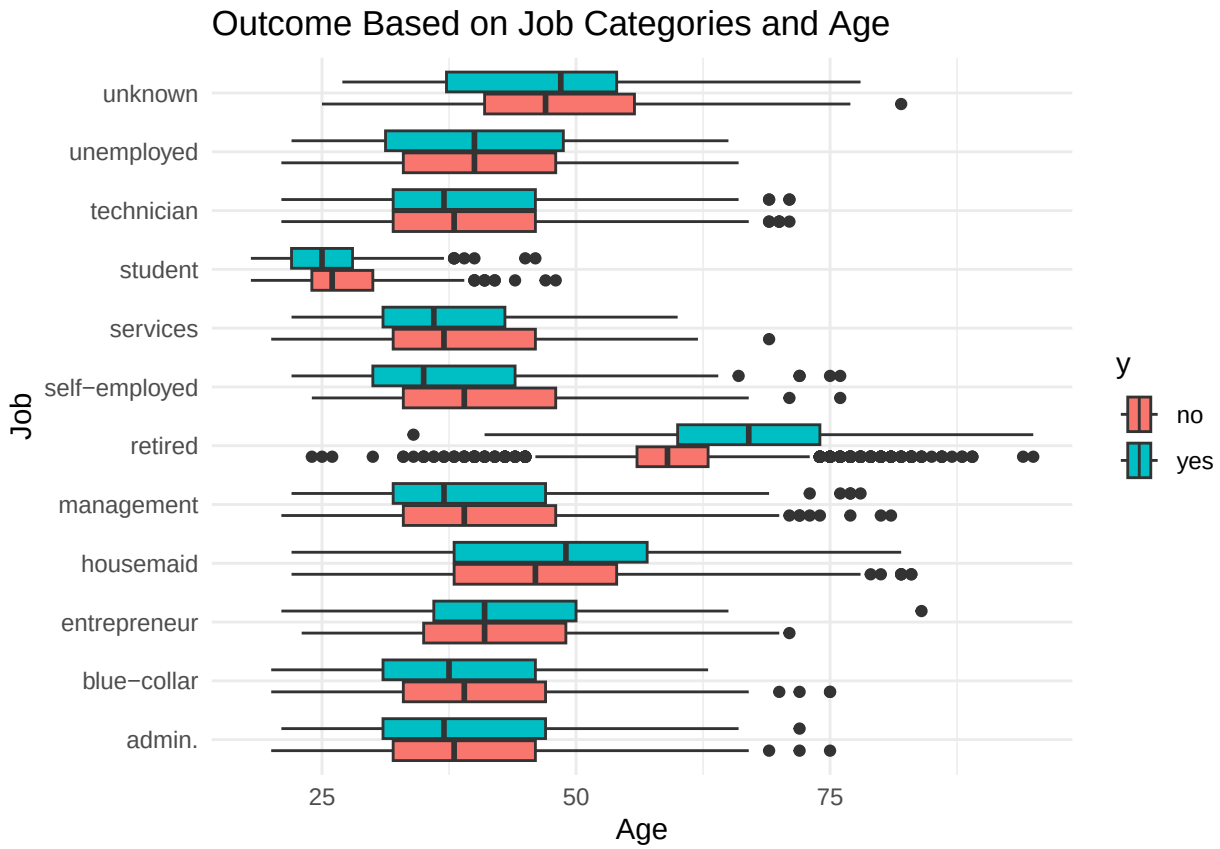


In this plot we intentionally wanted to highlight the disparity in the number of unknowns to the "failure, success, other" poutcomes feature classes. The vast majority of these calls have an unknown previous outcome, hinting that it might potentially be a poor variable to model with.

4.2 EDA on Potential Relationships With Variables

The boxplot below hints that there could be some messy data within the retired category, where we see outliers of 25 year olds that are retired. It seems that within a job category, the age of the person has no material impact in their willingness to sign up for a term deposit. Interestingly though, older retired people (ages ~ 55-75) seem to sign up for term deposits more often than younger retired people (ages ~ 50-55).

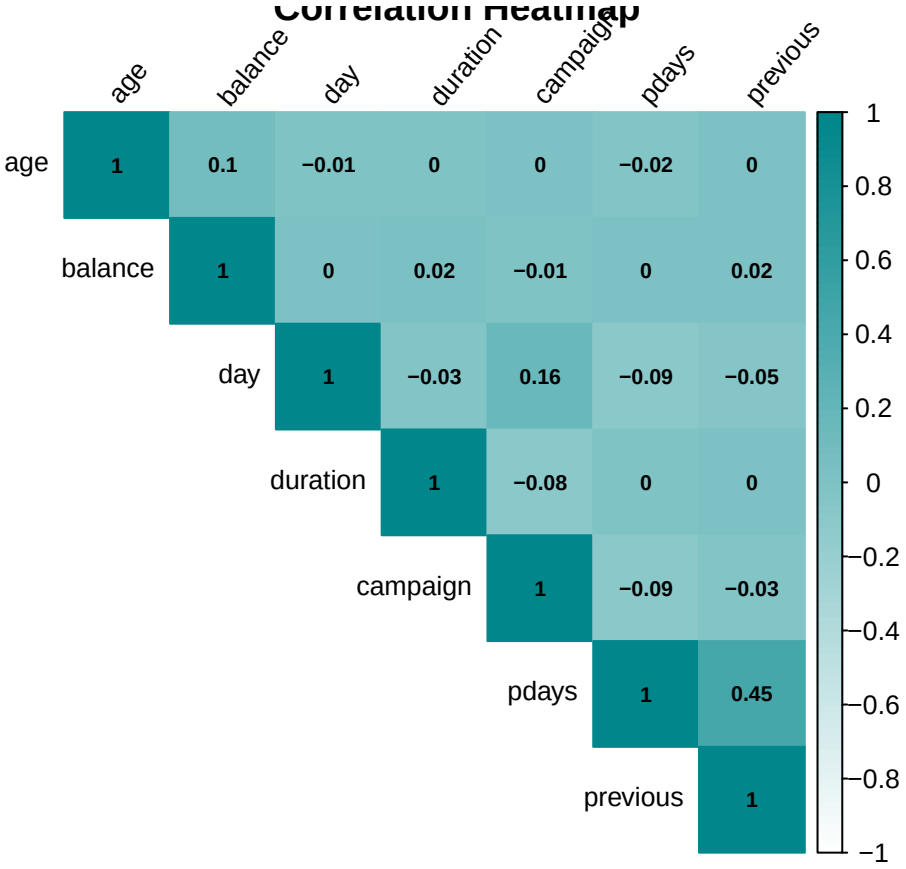
```
#Compare job categories, age, and y response variable to see if there's a story here:
ggplot(banking_marketing_dataset,aes(x=age,y=job,fill=y))+
geom_boxplot()+labs(x="Age",y ="Job",
title="Outcome Based on Job Categories and Age")+theme_minimal()
```



4.3 Relationships between the Quantitative Variables

It is important to understand if any sort of relationships (i.e. multicollinearity) exists between dependent variables as it can cause issues when building out our models. In order to investigate whether any sort of relationships exists between the "quantitative" variables we can use a correlation matrix. During our model building we will also calculate the VIF (Variance Inflation Factor) for the variables present in our model.

```
# Selecting numeric columns from the data set:
numeric_cols<-sapply(banking_marketing_dataset, is.numeric)
numeric_data<-banking_marketing_dataset[, numeric_cols]

#Computing the correlation matrix:
cor_matrix <- cor(numeric_data)

#Creating a custom color scheme:
my_colors <- colorRampPalette(c("white", "#00868B"))(100)

#Creating the correlation heatmap with the custom color scheme:
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.8,
tl.col = "black", col = my_colors,
addCoef.col = "black", number.cex = 0.7, tl.srt = 50,title = "Correlation Heatmap")
```



The correlation plot shows that the there are no significant relationships detected between the quantitative variables in our data set. The closest one would be pdays and previous, which makes sense since pdays is the number of days since last contact and previous is the number of contacts performed before this campaign and for this client. The driver behind this relationship is likely that customers being contacted the first time have no previous data.

4.4 Relationships between the various Quantitative and Qualitative Variables

In order to compare relationships between the "quantitative" and "quantitative" variables we can use a few tools, we can use box plots, bar plots, perform T-test (ANOVA), scatter plots etc. Below we will use the ANOVA function and print the results of only the relationships between quantitative and qualitative variables that are significant (i.e. p-value < 0.05).

It is evident from below that numerous "significant" relationships exists between many of the quantitative and quantitative variables in our data set. Again, we will need to be cognizant of this during our model building as this will pose a challenge when attempting to only select the best variables which may have an impact on the dependent variable.

```
#Specify the qualitative variables and quantitative variables from the data
#set and store them in a vector
qual_variables<-c("job","marital","education","default","housing",
                  "loan","contact","day","month","poutcome")
quant_variables<-c("age","balance","duration","campaign","previous")

#Iterate over the qualitative variables
for (qual_var in qual_variables) {
```

```r
  #Fit ANOVA model for the current qualitative variable against
  #each quantitative variable
  for (quant_var in quant_variables) {
    model<-aov(as.formula(paste(quant_var,"~",qual_var)),
    data=banking_marketing_dataset)
    anova_table<-summary.aov(model)

    #Get the p-value from the ANOVA table
    p_value<-anova_table[[1]]$Pr[1]

    #Check if the p-value is significant (i.e. p-value < 0.05) and if so print the results
    if (p_value<0.05) {
      cat("\nANOVA Results for", qual_var, "vs", quant_var, "\n")
      print(anova_table)
    }
  }
}
```

```
##
## ANOVA Results for job vs age
##                Df  Sum Sq Mean Sq F value Pr(>F)
## job            11 1280211  116383    1378 <2e-16 ***
## Residuals   45199 3817583      84
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for job vs balance
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## job            11 4.341e+09 394673996   43.01 <2e-16 ***
## Residuals   45199 4.148e+11   9176804
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for job vs duration
##                Df    Sum Sq Mean Sq F value   Pr(>F)
## job            11 4.985e+06  453172   6.843 1.23e-11 ***
## Residuals   45199 2.993e+09   66226
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for job vs campaign
##                Df Sum Sq Mean Sq F value Pr(>F)
## job            11   1314  119.48   12.48 <2e-16 ***
## Residuals   45199 432599    9.57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for job vs previous
##                Df Sum Sq Mean Sq F value   Pr(>F)
## job            11    442   40.21   7.591 3.18e-13 ***
## Residuals   45199 239435    5.30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## ANOVA Results for marital vs age
##                Df  Sum Sq Mean Sq F value Pr(>F)
## marital         2  957685  478843    5229 <2e-16 ***
## Residuals   45208 4140109      92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for marital vs balance
##                Df    Sum Sq   Mean Sq F value   Pr(>F)
## marital         2 3.326e+08 166322537   17.95 1.61e-08 ***
## Residuals   45208 4.188e+11   9263651
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for marital vs duration
##                Df    Sum Sq Mean Sq F value  Pr(>F)
## marital         2 1.601e+06  800669   12.08 5.7e-06 ***
## Residuals   45208 2.997e+09   66288
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for marital vs campaign
##                Df Sum Sq Mean Sq F value   Pr(>F)
## marital         2    428  214.18   22.34 2.01e-10 ***
## Residuals   45208 433485    9.59
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for marital vs previous
##                Df Sum Sq Mean Sq F value  Pr(>F)
## marital         2     69   34.74    6.55 0.00143 **
## Residuals   45208 239808    5.30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for education vs age
##                Df  Sum Sq Mean Sq F value Pr(>F)
## education       3  236087   78696   731.8 <2e-16 ***
## Residuals   45207 4861707     108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for education vs balance
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## education       3 3.220e+09 1.073e+09   116.7 <2e-16 ***
## Residuals   45207 4.159e+11 9.200e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for education vs campaign
##                Df Sum Sq Mean Sq F value   Pr(>F)
## education       3    190   63.49   6.618 0.000182 ***
## Residuals   45207 433723    9.59
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for education vs previous
##               Df Sum Sq Mean Sq F value   Pr(>F)
## education      3    165   54.95   10.36 8.19e-07 ***
## Residuals   45207 239712    5.30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for default vs age
##               Df  Sum Sq Mean Sq F value   Pr(>F)
## default        1    1630  1629.6   14.46 0.000144 ***
## Residuals   45209 5096164   112.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for default vs balance
##               Df    Sum Sq   Mean Sq F value Pr(>F)
## default        1 1.867e+09 1.867e+09   202.3 <2e-16 ***
## Residuals   45209 4.173e+11 9.230e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for default vs duration
##               Df    Sum Sq Mean Sq F value Pr(>F)
## default        1 3.011e+05  301124   4.541 0.0331 *
## Residuals   45209 2.998e+09   66315
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for default vs campaign
##               Df Sum Sq Mean Sq F value   Pr(>F)
## default        1    123   122.8    12.8 0.000348 ***
## Residuals   45209 433791     9.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for default vs previous
##               Df Sum Sq Mean Sq F value   Pr(>F)
## default        1     81   80.59   15.19 9.72e-05 ***
## Residuals   45209 239796    5.30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for housing vs age
##               Df  Sum Sq Mean Sq F value Pr(>F)
## housing        1  175441  175441    1611 <2e-16 ***
## Residuals   45209 4922353     109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for housing vs balance
##               Df    Sum Sq   Mean Sq F value Pr(>F)
```

```
## housing         1 1.982e+09 1.982e+09   214.8 <2e-16 ***
## Residuals   45209 4.171e+11 9.227e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for housing vs campaign
##                Df Sum Sq Mean Sq F value   Pr(>F)
## housing         1    242  241.65   25.19 5.21e-07 ***
## Residuals   45209 433672    9.59
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for housing vs previous
##                Df Sum Sq Mean Sq F value   Pr(>F)
## housing         1    330   329.7   62.23 3.12e-15 ***
## Residuals   45209 239547     5.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for loan vs age
##                Df  Sum Sq Mean Sq F value   Pr(>F)
## loan            1    1249  1249.4   11.08 0.000872 ***
## Residuals   45209 5096545   112.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for loan vs balance
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## loan            1 2.982e+09 2.982e+09     324 <2e-16 ***
## Residuals   45209 4.161e+11 9.205e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for loan vs duration
##                Df    Sum Sq Mean Sq F value  Pr(>F)
## loan            1 4.619e+05  461917   6.966 0.00831 **
## Residuals   45209 2.998e+09   66312
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for loan vs campaign
##                Df Sum Sq Mean Sq F value Pr(>F)
## loan            1     43   43.22   4.503 0.0338 *
## Residuals   45209 433870    9.60
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for loan vs previous
##                Df Sum Sq Mean Sq F value Pr(>F)
## loan            1     29  29.255   5.514 0.0189 *
## Residuals   45209 239848   5.305
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## ANOVA Results for contact vs age
##                Df  Sum Sq Mean Sq F value Pr(>F)
## contact         2  148290   74145   677.2 <2e-16 ***
## Residuals   45208 4949504     109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for contact vs balance
##                Df    Sum Sq   Mean Sq F value Pr(>F)
## contact         2 1.019e+09 509688184   55.11 <2e-16 ***
## Residuals   45208 4.181e+11   9248461
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for contact vs duration
##                Df    Sum Sq Mean Sq F value   Pr(>F)
## contact         2 2.641e+06 1320386   19.93 2.24e-09 ***
## Residuals   45208 2.996e+09   66265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for contact vs campaign
##                Df Sum Sq Mean Sq F value Pr(>F)
## contact         2   1346   672.9   70.33 <2e-16 ***
## Residuals   45208 432568     9.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for contact vs previous
##                Df Sum Sq Mean Sq F value Pr(>F)
## contact         2   5702  2851.2   550.4 <2e-16 ***
## Residuals   45208 234175     5.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for day vs duration
##                Df    Sum Sq Mean Sq F value   Pr(>F)
## day             1 2.736e+06 2735767   41.29 1.33e-10 ***
## Residuals   45209 2.996e+09   66262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for day vs campaign
##                Df Sum Sq Mean Sq F value Pr(>F)
## day             1  11457   11457    1226 <2e-16 ***
## Residuals   45209 422457       9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for day vs previous
##                Df Sum Sq Mean Sq F value Pr(>F)
## day             1    641   641.4   121.2 <2e-16 ***
## Residuals   45209 239236     5.3
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for month vs age
##               Df  Sum Sq Mean Sq F value Pr(>F)
## month          11  130859   11896   108.3 <2e-16 ***
## Residuals   45199 4966935     110
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for month vs balance
##               Df    Sum Sq   Mean Sq F value Pr(>F)
## month          11 1.018e+10 925370276   102.3 <2e-16 ***
## Residuals   45199 4.089e+11   9047649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for month vs duration
##               Df    Sum Sq Mean Sq F value Pr(>F)
## month          11 1.382e+07 1256469   19.03 <2e-16 ***
## Residuals   45199 2.985e+09   66031
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for month vs campaign
##               Df Sum Sq Mean Sq F value Pr(>F)
## month          11  23281  2116.4     233 <2e-16 ***
## Residuals   45199 410633     9.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for month vs previous
##               Df Sum Sq Mean Sq F value Pr(>F)
## month          11   7375   670.5   130.3 <2e-16 ***
## Residuals   45199 232502     5.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for poutcome vs age
##               Df  Sum Sq Mean Sq F value Pr(>F)
## poutcome       3    8909  2969.8   26.38 <2e-16 ***
## Residuals   45207 5088885   112.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for poutcome vs balance
##               Df    Sum Sq   Mean Sq F value   Pr(>F)
## poutcome       3 6.546e+08 218183948   23.57 3.09e-15 ***
## Residuals   45207 4.185e+11   9256735
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for poutcome vs duration
##               Df    Sum Sq Mean Sq F value Pr(>F)
## poutcome       3 6.183e+06 2060881   31.14 <2e-16 ***
```

```
## Residuals   45207 2.992e+09   66188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for poutcome vs campaign
##                Df Sum Sq Mean Sq F value Pr(>F)
## poutcome        3   5482  1827.5   192.8 <2e-16 ***
## Residuals   45207 428431     9.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ANOVA Results for poutcome vs previous
##                Df Sum Sq Mean Sq F value Pr(>F)
## poutcome        3  69761   23254    6180 <2e-16 ***
## Residuals   45207 170116       4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.5 Relationships between the various Qualitative Variables

In order to compare relationships between the various "qualitative" variables we will conduct a Chi-square test and print the results of only the relationships between qualitative variables that are significant (i.e. p-value < 0.05).

Again similar to our analysis above numerous "significant" relationships exists between many of the qualitative variables in our data set. Again, caution will be used when building out our models.

5. Sampling Methodology

An assumption we have made with out data set is that it is a good "representation" of the population from which it was taken. Meaning that all relevant subgroups or "strata" that are characteristically different from one another in the population have been captured in our sample data set. We have also made the assumption that the subgroups or strata in the data set are represented "proportionally" to the population from which they were taken.

That being said we plan to train and evaluate our models with two different sampling strategies each with a 75-25 train-test sample size. The two difference sampling strategies we will employ on our data set are found below.

1) Simple Random Sampling
2) Undersampling

We will use the simple random sampling data sets as our "baseline" in order to evaluate and compare our models performance and accuracy. We will then use the undersampling in order to draw a equal proportion of our dependent variable (y) in our train and test sample sets to evaluate the impact on our models accuracy and performance.

5.1 Create Train and Test Simple Random Sample Data Subsets with 75-25 Split

Below we will create "Train" and "Test" data subsets from our original data set using simple random sampling. The split will be; 75% of the original subset observations will be assigned to the "Train" subset and the remaining 25% will be assigned to the "Test" subset.

```
#original split
  set.seed(1)
  idx=sample(dim(banking_marketing_dataset)[1],
             round(nrow(banking_marketing_dataset)*0.75,0))
  Train1=banking_marketing_dataset[idx,]
  Test1=banking_marketing_dataset[-idx,]
```

```
#Train data sub set
table_1 = as.table(table(Train1$y))
prop.table(table_1)
```

```
##
##        no       yes
## 0.8825941 0.1174059
```

```
#Test data sub set
table_2 = as.table(table(Test1$y))
prop.table(table_2)
```

```
##
##        no       yes
## 0.8842785 0.1157215
```

```
#Entire data set
prop.table(table(banking_marketing_dataset$y))
```

```
##
##        no       yes
## 0.8830152 0.1169848
```

We can see above that the proportion of "yes" and "no" data points in our "Train" and "Test" data sub sets roughly follows the same proportion in our entire sample. For this reason we have decided not to include a "stratified" sampling approach that would allocate the same proportions to the Test and Train data set as it would produce a similar result.

5.2 Create Train and Test Undersampled Data Subsets with 75-25 Split

Below we will create "Train" and "Test" data subsets from our original data set using "undersampling." The split will be; 75% of the original subset observations will be assigned to the "Train" subset and the remaining 25% will be assigned to the "Test" subset. The resultant or "dependent" variable "y" will have proportional allocation of 50% - yes and 50% - no in both the Train and Test data subsets. Whether utilizing these Test or Train data subsets will have a positive or negative impact on our models accuracy is to be investigated.

```
#Create the "undersampled" data set and display the top few lines
undersampling_df<- ovun.sample(y ~ .,
data = banking_marketing_dataset, method = "under", p = 0.5, seed = 1)$data
head(undersampling_df)
```

```
##   age         job  marital education default balance housing loan  contact day
## 1  35  technician divorced secondary      no    2047     yes   no cellular   1
```

```
## 2  26 blue-collar   single   primary      no      93     yes   no cellular  14
## 3  35  management  married  tertiary      no    2123     yes   no cellular  21
## 4  43  technician   single secondary      no     326     yes   no cellular  15
## 5  37  technician divorced secondary      no     695     yes   no  unknown  26
## 6  36 blue-collar  married secondary      no     604     yes   no  unknown  30
##   month duration campaign pdays previous poutcome  y age_group duration_group
## 1   oct       66        1    -1        0  unknown no     31-40        0-3 min
## 2   may       43        3    -1        0  unknown no     26-30        0-3 min
## 3   jul      249        3    -1        0  unknown no     31-40        3-6 min
## 4   jul       87        8    -1        0  unknown no     41-50        0-3 min
## 5   may       80        2    -1        0  unknown no     31-40        0-3 min
## 6   may      235        1    -1        0  unknown no     31-40        3-6 min
```

```
#Check the dimensions of the under sampled data set
dim(undersampling_df)
```

```
## [1] 10549    19
```

```
#Create under sampled Test and Train Data Subsets
  set.seed(1)
  idx=sample(dim(undersampling_df)[1],round(nrow(undersampling_df)*0.75,0))
  Train2=undersampling_df[idx,]
  Test2=undersampling_df[-idx,]
```

```
#Confirm 50-50 proportional based on "y" for Test and Train data subsets and Undersampled data set
table_2 = as.table(table(Train2$y))
prop.table(table_2)
```

```
##
##        no       yes
## 0.4984833 0.5015167
```

```
table_2
```

```
##
##   no  yes
## 3944 3968
```

```
table_3 = as.table(table(Test2$y))
prop.table(table_3)
```

```
##
##       no      yes
## 0.499052 0.500948
```

```
table_3
```

```
##
##   no  yes
## 1316 1321
```

```
prop.table(table(undersampling_df$y))
```

```
##
##        no       yes
## 0.4986255 0.5013745
```

As we can see above we have created a 50-50 proportional under sampled data set along with the respected 75-25 Test-Train data subsets in which we will be using to train and test our various models.

6 Model Fitting

In this section we will look to train, test and evaluate the following types of models for our two sampling methods discussed earlier.

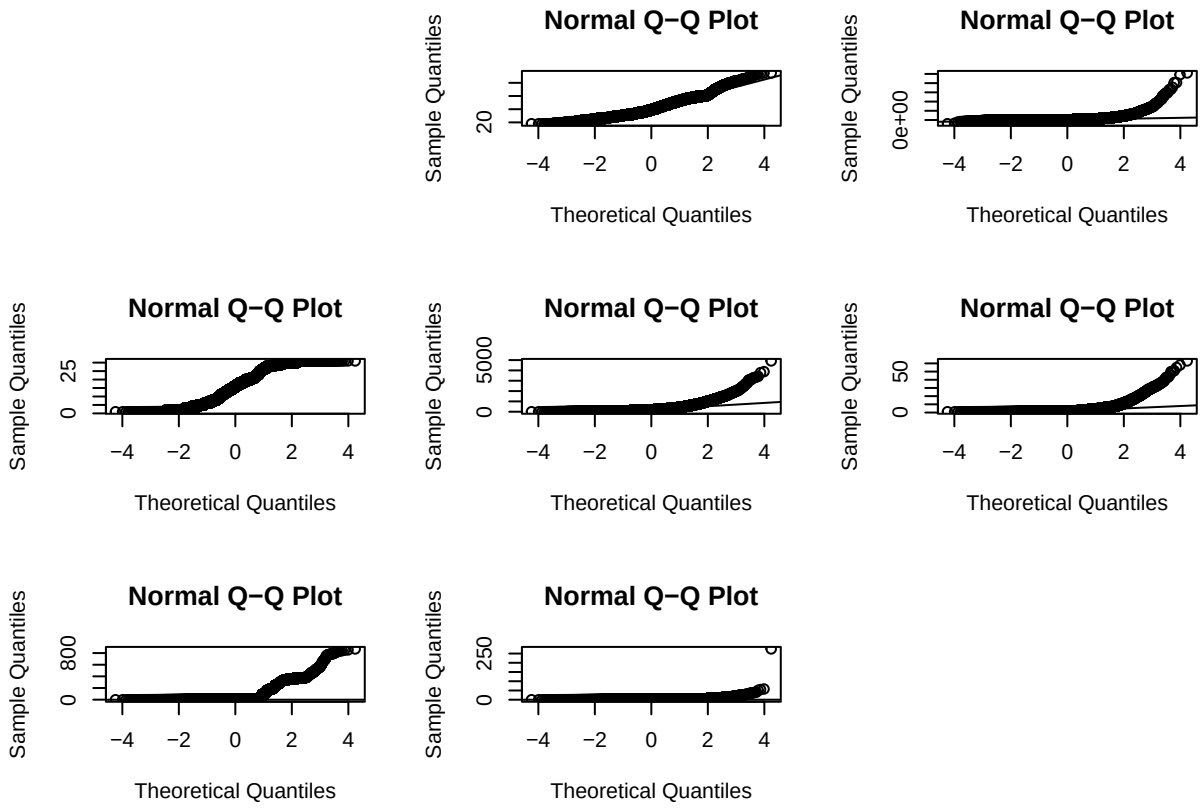
-Linear Discriminant Analysis -Quadratic Discriminant Analysis -Logistic Regression -Classification Tree

6.1 Linear Discriminant Analysis (LDA) (Simple Random Sampling)

In order to fit our data to a LDA model we need to ensure the following assumptions are met.

Independence: Our observations are phone calls to various individuals and we have no reason to suspect they are NOT independent. Normality: Does not appear to hold for certain variables as can be seen in the QQ-plots below.
Linear Separability: The data can be separated by a linear bounder, TBD. We will investigate QDA as well.

```
#Create a subset of the dataset to include only numeric/quantitative variables
numeric_data<-banking_marketing_dataset[,numeric_cols]

#Fit to one page
num_plots<-length(colnames(numeric_data))
num_cols<-3
num_rows <- ceiling(num_plots / num_cols)
par(mfrow = c(num_rows, num_cols))

plot.new()

#Create QQ plots for each numeric/quantitative variable
for (i in 1:num_plots) {
  col <- colnames(numeric_data)[i]
  plot_num <- (i - 1) %% (num_cols * num_rows) + 1  # Calculate plot position
  par(mfg = c(plot_num %/% num_cols + 1, plot_num %% num_cols + 1))

  qqnorm(numeric_data[[col]])
  qqline(numeric_data[[col]])
}
```



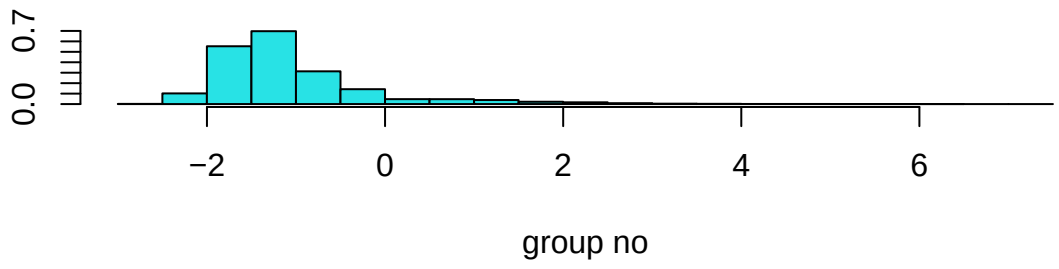

```
#Train a LDA model using the SRS sample Train data subset
lda.fit1<-lda(y~., data=Train1)
lda.fit1
```

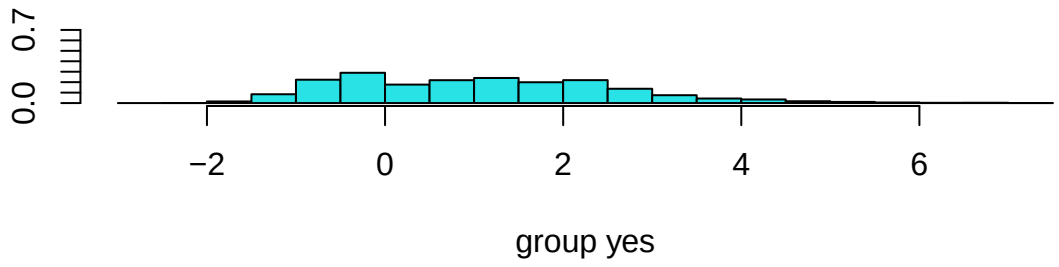
```
## Call:
## lda(y ~ ., data = Train1)
##
## Prior probabilities of groups:
##        no       yes
## 0.8825941 0.1174059
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.81401      0.2256491      0.03411635   0.02897049     0.2038293
## yes 41.62849      0.1296157      0.02361216   0.02059784     0.2461693
##     jobretired jobself-employed jobservices jobstudent jobtechnician
## no  0.04380660       0.03508537  0.09556588 0.01610586     0.1696796
## yes 0.09796534       0.03566943  0.07460437 0.05149460     0.1595077
##     jobunemployed  jobunknown maritalmarried maritalsingle educationsecondary
## no     0.02699903 0.006649514      0.6110201     0.2740335          0.5203328
## yes    0.03566943 0.006279829      0.5192163     0.3599598          0.4664657
##     educationtertiary educationunknown  defaultyes  balance housingyes
## no          0.2830888       0.03983025 0.019246834 1297.643  0.5848231
## yes         0.3760362       0.04747551 0.009796534 1766.910  0.3705099
##       loanyes contacttelephone contactunknown      day  monthaug    monthdec
## no  0.16760785       0.06348782      0.3134962 15.89013 0.1372339 0.003107562
```

```
## yes 0.08791761       0.07435318      0.1019844 15.19945 0.1250942 0.016578749
##       monthfeb   monthjan  monthjul  monthjun    monthmar  monthmay   monthnov
## no  0.05339660 0.03147659 0.1583520 0.1193905 0.005880977 0.3229191 0.08911685
## yes 0.08540568 0.02461693 0.1223311 0.1034916 0.047224315 0.1783471 0.07410198
##       monthoct    monthsep duration campaign    pdays  previous poutcomeother
## no  0.01119391 0.007752197 221.2483 2.844388 36.59080 0.5110435    0.03812611
## yes 0.06330068 0.051494599 536.2871 2.129364 69.14142 1.1441849    0.05601608
##     poutcomesuccess poutcomeunknown age_group26-30 age_group31-40
## no        0.0132322       0.8400775      0.1226652      0.3984028
## yes       0.1848782       0.6405426      0.1547350      0.3411203
##     age_group41-50 age_group51-60 age_group61+ duration_group3-6 min
## no       0.2557891      0.1799044   0.01770976             0.2906740
## yes      0.1939211      0.1537302   0.09394624             0.2906305
##     duration_group6-10 min duration_group10-15 min duration_group15-20 min
## no               0.1092325              0.03485147             0.009021953
## yes              0.2275810              0.18638533             0.094197438
##     duration_group20-25 min duration_group25+ min
## no              0.002973903            0.00237244
## yes             0.040190907            0.02612409
##
## Coefficients of linear discriminants:
##                                   LD1
## age                     -2.618647e-03
## jobblue-collar          -1.554730e-01
## jobentrepreneur         -1.626389e-01
## jobhousemaid            -2.480480e-01
## jobmanagement           -6.458597e-02
## jobretired              -1.251786e-01
## jobself-employed        -1.383698e-01
## jobservices             -7.095138e-02
## jobstudent               3.330930e-01
## jobtechnician           -7.288803e-02
## jobunemployed           -8.570372e-02
## jobunknown              -2.655228e-01
## maritalmarried          -9.664154e-02
## maritalsingle           -1.034899e-02
## educationsecondary       3.739353e-02
## educationtertiary        1.641985e-01
## educationunknown         3.326289e-02
## defaultyes              -3.067287e-03
## balance                  6.035924e-06
## housingyes              -3.189041e-01
## loanyes                 -1.744586e-01
## contacttelephone        -9.397042e-02
## contactunknown          -5.541958e-01
## day                      7.518088e-03
## monthaug                -4.180093e-01
## monthdec                 6.208123e-01
## monthfeb                 2.177671e-02
## monthjan                -6.837200e-01
## monthjul                -4.138469e-01
## monthjun                 1.767941e-01
## monthmar                 1.859966e+00
## monthmay                -1.634779e-01
```

```
## monthnov                -4.520949e-01
## monthoct                 1.023641e+00
## monthsep                 1.100505e+00
## duration                 1.500010e-03
## campaign                -1.440651e-02
## pdays                   -2.076923e-04
## previous                 3.786562e-03
## poutcomeother            1.108940e-01
## poutcomesuccess          2.736370e+00
## poutcomeunknown         -8.577414e-02
## age_group26-30          -3.852185e-01
## age_group31-40          -4.486981e-01
## age_group41-50          -4.061455e-01
## age_group51-60          -3.786184e-01
## age_group61+             4.977943e-01
## duration_group3-6 min    2.089412e-01
## duration_group6-10 min   5.493260e-01
## duration_group10-15 min  1.507151e+00
## duration_group15-20 min  2.177246e+00
## duration_group20-25 min  2.286777e+00
## duration_group25+ min    9.675507e-01
```

```
plot(lda.fit1)
```





```
#Using the trained LDA model predict using the SRS Test Data Subset
set.seed(1)
lda.pred1=predict(lda.fit1, Test1)
table(lda.pred1$class, Test1$y)
```

```
##
##         no  yes
##   no  9513  660
##   yes  482  648
```

```
paste('Accuracy rate for full LDA model: ',
      100-round(100*mean(lda.pred1$class!=Test1$y),2),'%',sep='')
```

```
## [1] "Accuracy rate for full LDA model: 89.9%"
```

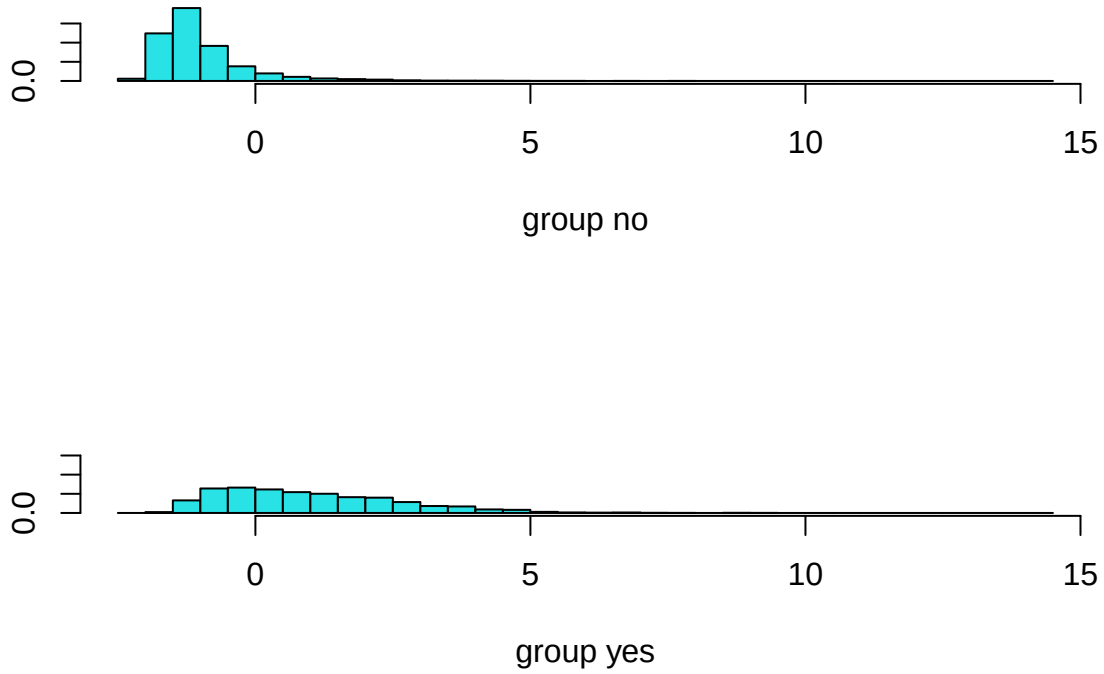
In an attempt to improve the LDA models accuracy we will reduce our initial LDA model by reducing the model based on positive discriminant values.

```
#Train a reduced LDA model base on the SRS sample Train data subset
lda.fit.2<-lda(y~age+marital+education+balance+day+month+duration+
                previous+poutcome, data=Train1)
lda.fit.2
```

```
## Call:
## lda(y ~ age + marital + education + balance + day + month + duration +
##     previous + poutcome, data = Train1)
##
## Prior probabilities of groups:
##        no       yes
## 0.8825941 0.1174059
##
## Group means:
##          age maritalmarried maritalsingle educationsecondary educationtertiary
## no  40.81401      0.6110201     0.2740335          0.5203328         0.2830888
## yes 41.62849      0.5192163     0.3599598          0.4664657         0.3760362
##     educationunknown  balance      day  monthaug    monthdec   monthfeb
## no        0.03983025 1297.643 15.89013 0.1372339 0.003107562 0.05339660
## yes       0.04747551 1766.910 15.19945 0.1250942 0.016578749 0.08540568
##       monthjan  monthjul  monthjun    monthmar  monthmay   monthnov   monthoct
## no  0.03147659 0.1583520 0.1193905 0.005880977 0.3229191 0.08911685 0.01119391
## yes 0.02461693 0.1223311 0.1034916 0.047224315 0.1783471 0.07410198 0.06330068
##        monthsep duration  previous poutcomeother poutcomesuccess
## no  0.007752197 221.2483 0.5110435    0.03812611       0.0132322
## yes 0.051494599 536.2871 1.1441849    0.05601608       0.1848782
##     poutcomeunknown
## no        0.8400775
## yes       0.6405426
##
## Coefficients of linear discriminants:
##                               LD1
## age                  4.390804e-03
## maritalmarried      -1.171814e-01
```

```
## maritalsingle       1.220568e-01
## educationsecondary  7.872238e-02
## educationtertiary   2.327281e-01
## educationunknown    1.373344e-01
## balance             1.160906e-05
## day                 4.769923e-03
## monthaug           -2.962904e-01
## monthdec            8.033104e-01
## monthfeb            5.801366e-02
## monthjan           -6.231668e-01
## monthjul           -4.187146e-01
## monthjun           -2.556200e-01
## monthmar            2.151051e+00
## monthmay           -5.854350e-01
## monthnov           -4.778601e-01
## monthoct            1.240689e+00
## monthsep            1.339782e+00
## duration            3.259027e-03
## previous            2.194664e-03
## poutcomeother       1.631046e-01
## poutcomesuccess     2.996161e+00
## poutcomeunknown    -1.580115e-01
```

```
plot(lda.fit.2)
```



```
#Using the trained "reduced" LDA model predict using the SRS Test Data Subset
set.seed(1)
lda.pred.2=predict(lda.fit.2, Test1)
table(lda.pred.2$class, Test1$y)
```

```
##
##         no  yes
##   no  9634  763
##   yes  361  545
```

```
paste('Acccuracy rate for reduced LDA model: ',
      100-round(100*mean(lda.pred.2$class!=Test1$y),2),'%',sep='')
```

```
## [1] "Acccuracy rate for reduced LDA model: 90.06%"
```

We can see from above that we have now managed to increase our LDA models accuracy ever so slightly from 89.9% to 90.0%, by removing certain variables. We will use 10 fold Cross Validation now on our "reduced" model in order to evaluate its performance. We have elected to go with the reduced LDA because of its slightly higher accuracy but also because it is a simpler (less variables) model to interpret.

Evaluating the LDA Models Performance using 10 Fold Cross Validation (Simple Random Sampling)

We can see from below that the accuracy of the reduced LDA model is 89.96%. This aligns very closely with the accuracy we found above (90.06%). his gives us good confidence in our models performance.

```
#Creating folds
set.seed(1)
folds1 = createFolds(factor(banking_marketing_dataset$y), k=10)
```

```
#Create a formula for calculating the classifications rate
misclassification_lda<-function(idx){
  Train<-banking_marketing_dataset[-idx,]
  Test<-banking_marketing_dataset[idx,]
  fit<-lda(y~age+marital+education+balance+day+month+duration+previous+poutcome,
           data=Train)
  pred<-predict(fit,Test)
  return(mean(pred$class==Test$y))
}
```

```
mis_rate1 = lapply(folds1, misclassification_lda)
```

```
paste('Accuracy rate for full LDA model: ',
      round(100*mean(as.numeric(mis_rate1)),2),'%',sep='')
```

```
## [1] "Accuracy rate for full LDA model: 89.96%"
```

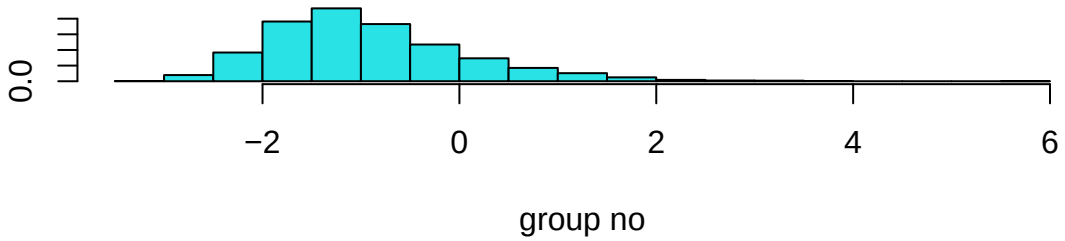
6.2 Linear Discriminant Analysis (LDA) (Undersampling)

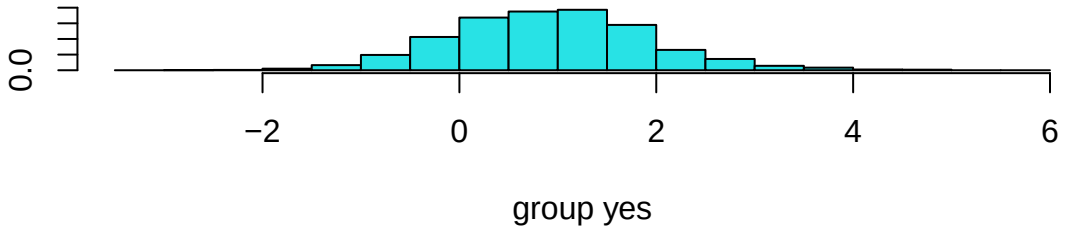
```r
#Train a LDA model base on the undersampled Train data subset
lda.fit1<-lda(y~., data=Train2)
lda.fit1
```

```
## Call:
## lda(y ~ ., data = Train2)
##
## Prior probabilities of groups:
##        no       yes
## 0.4984833 0.5015167
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.88235      0.2223631      0.03042596   0.03093306     0.1975152
## yes 41.62198      0.1353327      0.02242944   0.02116935     0.2414315
##     jobretired jobself-employed jobservices jobstudent jobtechnician
## no  0.04994929       0.03144016  0.09355984 0.01495943     0.1744422
## yes 0.09727823       0.03301411  0.07031250 0.04863911     0.1633065
##     jobunemployed  jobunknown maritalmarried maritalsingle educationsecondary
## no     0.02712982 0.007860041      0.6179006     0.2687627          0.5195233
## yes    0.03931452 0.005544355      0.5183972     0.3623992          0.4662298
##     educationtertiary educationunknown defaultyes  balance housingyes
## no           0.275355       0.03752535 0.01698783 1326.535  0.5834178
## yes          0.375252       0.04763105 0.01008065 1848.906  0.3707157
##        loanyes contacttelephone contactunknown      day  monthaug    monthdec
## no  0.16379310       0.06896552      0.3108519 15.80629 0.1364097 0.003549696
## yes 0.09072581       0.07510081      0.1003024 15.08443 0.1335685 0.019153226
##       monthfeb   monthjan  monthjul   monthjun    monthmar  monthmay   monthnov
## no  0.05400609 0.02535497 0.1577079 0.12119675 0.007606491 0.3166836 0.09051724
## yes 0.08492944 0.02444556 0.1154234 0.09929435 0.047379032 0.1738911 0.07686492
##       monthoct   monthsep duration campaign    pdays previous poutcomeother
## no  0.01090264 0.01191684 223.3702 2.787272 37.33342 0.525355    0.03727181
## yes 0.05947581 0.05241935 535.7694 2.125252 68.97681 1.179940    0.05695565
##     poutcomesuccess poutcomeunknown age_group26-30 age_group31-40
## no       0.01419878       0.8369675      0.1184077      0.3985801
## yes      0.18447581       0.6396169      0.1532258      0.3460181
##     age_group41-50 age_group51-60 age_group61+ duration_group3-6 min
## no       0.2654665      0.1736815   0.01926978             0.2958925
## yes      0.1932964      0.1504536   0.09450605             0.2847782
##     duration_group6-10 min duration_group10-15 min duration_group15-20 min
## no               0.1090264              0.03625761             0.007606491
## yes              0.2295867              0.18775202             0.093497984
##     duration_group20-25 min duration_group25+ min
## no              0.003296146           0.002281947
## yes             0.040826613           0.025201613
##
## Coefficients of linear discriminants:
##                                    LD1
## age                     -6.359641e-03
## jobblue-collar          -2.093488e-01
## jobentrepreneur         -2.223219e-01
## jobhousemaid            -3.313485e-01
## jobmanagement           -1.622933e-01
```

```
## jobretired              -4.016389e-01
## jobself-employed        -3.157275e-01
## jobservices             -1.939653e-01
## jobstudent               1.390706e-01
## jobtechnician           -1.875362e-01
## jobunemployed           -1.214688e-01
## jobunknown              -4.597799e-01
## maritalmarried          -1.231400e-01
## maritalsingle           -1.768251e-02
## educationsecondary       1.811047e-01
## educationtertiary        3.541622e-01
## educationunknown         3.035585e-01
## defaultyes              -9.009398e-02
## balance                  9.928605e-06
## housingyes              -4.055069e-01
## loanyes                 -2.340652e-01
## contacttelephone        -2.049357e-01
## contactunknown          -7.754942e-01
## day                      2.983090e-03
## monthaug                -4.802499e-01
## monthdec                 2.719510e-01
## monthfeb                 3.615734e-03
## monthjan                -7.126313e-01
## monthjul                -6.274204e-01
## monthjun                 6.357917e-02
## monthmar                 9.871637e-01
## monthmay                -3.999950e-01
## monthnov                -5.107743e-01
## monthoct                 6.070852e-01
## monthsep                 2.584455e-01
## duration                 1.330491e-03
## campaign                -3.622585e-02
## pdays                    2.689516e-05
## previous                 5.345251e-03
## poutcomeother            1.692367e-01
## poutcomesuccess          1.182083e+00
## poutcomeunknown         -1.051598e-01
## age_group26-30          -2.793044e-01
## age_group31-40          -4.430822e-01
## age_group41-50          -3.850727e-01
## age_group51-60          -2.273082e-01
## age_group61+             6.110799e-01
## duration_group3-6 min    5.791258e-01
## duration_group6-10 min   1.135802e+00
## duration_group10-15 min  1.674389e+00
## duration_group15-20 min  1.709024e+00
## duration_group20-25 min  1.378581e+00
## duration_group25+ min    6.733279e-01
```

```
plot(lda.fit1)
```






```
#Using the trained LDA model predict using the undersampled Test data subset
set.seed(1)
lda.pred1=predict(lda.fit1, Test2)
table(lda.pred1$class, Test2$y)
```

```
##
##         no  yes
##   no  1112  237
##   yes  204 1084
```

```
paste('Accuracy for full LDA model: ',
      100-round(100*mean(lda.pred1$class!=Test2$y),2),'%',sep='')
```

```
## [1] "Accuracy for full LDA model: 83.28%"
```

We can see from above that we have now decreased our LDA models accuracy from 89% to 83% when we utilized the under sampled data sets. We have gained a larger number of "yes" predictions; however, for the purpose of this project we want to be able to accurately model what factors or variables may influence a person decision to open a term account.

Producing a model that gives false "hope" to bankers would be cruel...

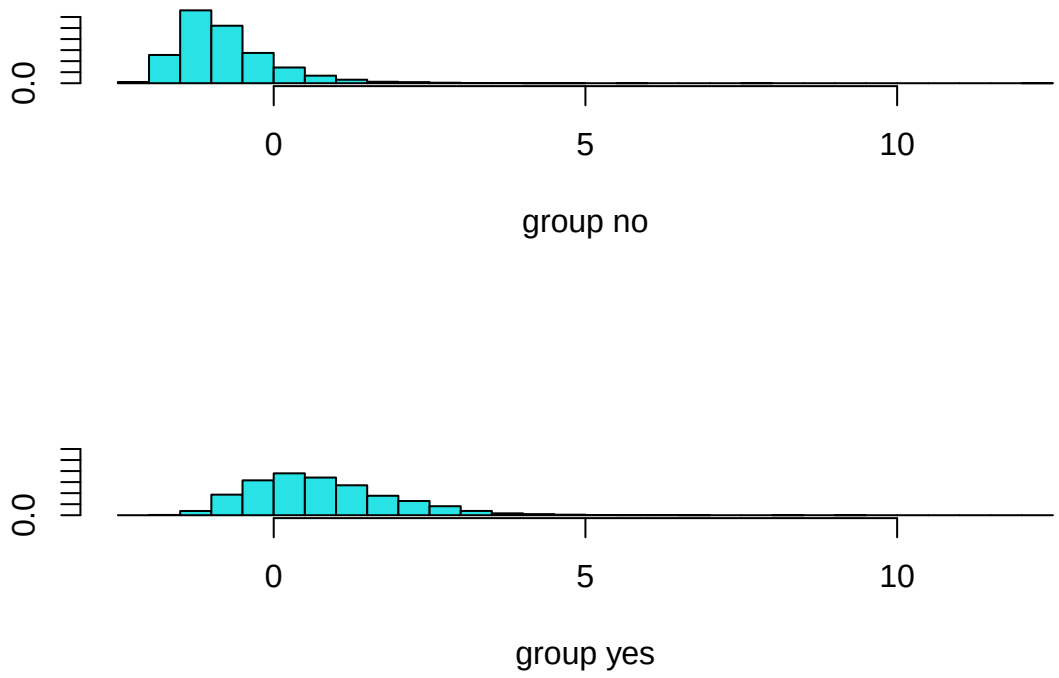
Again just as we carried out earlier, we will look to improve on our LDA models accuracy by reducing the model based on positive discriminant values.

```r
#Train a reduced LDA model base on the under sampled Train data subset
lda.fit.2<-lda(y~age+marital+education+default+balance+day+month+
                 duration+previous+poutcome, data=Train2)
lda.fit.2
```

```
## Call:
## lda(y ~ age + marital + education + default + balance + day +
##     month + duration + previous + poutcome, data = Train2)
##
## Prior probabilities of groups:
##        no       yes
## 0.4984833 0.5015167
##
## Group means:
##          age maritalmarried maritalsingle educationsecondary educationtertiary
## no  40.88235      0.6179006     0.2687627          0.5195233          0.275355
## yes 41.62198      0.5183972     0.3623992          0.4662298          0.375252
##     educationunknown defaultyes  balance      day  monthaug    monthdec
## no        0.03752535 0.01698783 1326.535 15.80629 0.1364097 0.003549696
## yes       0.04763105 0.01008065 1848.906 15.08443 0.1335685 0.019153226
##       monthfeb   monthjan  monthjul   monthjun    monthmar  monthmay   monthnov
## no  0.05400609 0.02535497 0.1577079 0.12119675 0.007606491 0.3166836 0.09051724
## yes 0.08492944 0.02444556 0.1154234 0.09929435 0.047379032 0.1738911 0.07686492
##       monthoct   monthsep duration previous poutcomeother poutcomesuccess
## no  0.01090264 0.01191684 223.3702 0.525355    0.03727181      0.01419878
## yes 0.05947581 0.05241935 535.7694 1.179940    0.05695565      0.18447581
##     poutcomeunknown
## no        0.8369675
## yes       0.6396169
##
## Coefficients of linear discriminants:
##                              LD1
## age                 6.879030e-03
## maritalmarried     -1.251808e-01
## maritalsingle       2.555522e-01
## educationsecondary  2.544335e-01
## educationtertiary   4.501488e-01
## educationunknown    4.552106e-01
## defaultyes         -1.613922e-01
## balance             1.554202e-05
## day                -3.088067e-03
## monthaug           -5.057124e-01
## monthdec            3.163812e-01
## monthfeb           -1.245662e-01
## monthjan           -7.287829e-01
## monthjul           -7.661916e-01
## monthjun           -5.620098e-01
## monthmar            1.004712e+00
## monthmay           -1.002085e+00
## monthnov           -6.704810e-01
## monthoct            6.461639e-01
## monthsep            3.231804e-01
## duration            2.726737e-03
## previous            2.428907e-03
```

```
## poutcomeother       2.736259e-01
## poutcomesuccess     1.442282e+00
## poutcomeunknown    -2.874090e-01
```

```
plot(lda.fit.2)
```




```
set.seed(1)
lda.pred2=predict(lda.fit.2, Test2)
table(lda.pred2$class, Test2$y)
```

```
##
##         no  yes
##   no  1147  359
##   yes  169  962
```

```
paste('Accuracy rate for reduced LDA model: ',
      100-round(100*mean(lda.pred2$class!=Test2$y),2),'%',sep='')
```

```
## [1] "Accuracy rate for reduced LDA model: 79.98%"
```

We can see from above that we have managed to decrease our LDA models accuracy from 83.31% to 79.98%, by removing certain variables. We will use 10 fold Cross Validation now on our "full" model in order to evaluate its performance based on the higher accuracy.

Evaluating the LDA Models Performance using 10 Fold Cross Validation (Undersampling)

We can see from below that the accuracy of the full LDA model is 83.67% using 10 Fold Cross Validation. This aligns very closely with the accuracy we found above (83.28%). This gives us good confidence in our models performance.

```
#Creating folds
set.seed(1)
folds2 = createFolds(factor(undersampling_df$y), k=10)
```

```
#Create a formula for calculating the classifications rate
misclassification_lda_2<-function(idx){
  Train<-undersampling_df[-idx,]
  Test<-undersampling_df[idx,]
  fit<-lda(y~., data=Train)
  pred<-predict(fit,Test2)
  return(mean(pred$class==Test2$y))
}
```

```
mis_rate2 = lapply(folds2, misclassification_lda_2)
```

```
paste('Accuracy rate for full LDA model: ',
      round(100*mean(as.numeric(mis_rate2)),2),'%',sep='')
```

```
## [1] "Accuracy rate for full LDA model: 83.67%"
```

6.3 Quadratic Discriminant Analysis (QDA) (Simple Random Sampling)

Much like LDA the assumptions for QDA also include "independence" and "normality". However, there is no assumption for "linear separability".

```
#Train a QDA model base on the Simplified Random Sampling Train data subset
qda.fit.1<-qda(y~., data=Train1)
qda.fit.1
```

```
## Call:
## qda(y ~ ., data = Train1)
##
## Prior probabilities of groups:
##        no       yes
## 0.8825941 0.1174059
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.81401      0.2256491      0.03411635   0.02897049     0.2038293
## yes 41.62849      0.1296157      0.02361216   0.02059784     0.2461693
##     jobretired jobself-employed jobservices jobstudent jobtechnician
## no  0.04380660       0.03508537  0.09556588 0.01610586     0.1696796
## yes 0.09796534       0.03566943  0.07460437 0.05149460     0.1595077
##     jobunemployed  jobunknown maritalmarried maritalsingle educationsecondary
## no     0.02699903 0.006649514      0.6110201     0.2740335          0.5203328
## yes    0.03566943 0.006279829      0.5192163     0.3599598          0.4664657
```

```
##     educationtertiary educationunknown  defaultyes  balance housingyes
## no          0.2830888       0.03983025 0.019246834 1297.643  0.5848231
## yes         0.3760362       0.04747551 0.009796534 1766.910  0.3705099
##        loanyes contacttelephone contactunknown      day  monthaug    monthdec
## no  0.16760785       0.06348782      0.3134962 15.89013 0.1372339 0.003107562
## yes 0.08791761       0.07435318      0.1019844 15.19945 0.1250942 0.016578749
##       monthfeb   monthjan  monthjul  monthjun    monthmar  monthmay   monthnov
## no  0.05339660 0.03147659 0.1583520 0.1193905 0.005880977 0.3229191 0.08911685
## yes 0.08540568 0.02461693 0.1223311 0.1034916 0.047224315 0.1783471 0.07410198
##       monthoct    monthsep duration campaign    pdays  previous poutcomeother
## no  0.01119391 0.007752197 221.2483 2.844388 36.59080 0.5110435    0.03812611
## yes 0.06330068 0.051494599 536.2871 2.129364 69.14142 1.1441849    0.05601608
##     poutcomesuccess poutcomeunknown age_group26-30 age_group31-40
## no        0.0132322       0.8400775      0.1226652      0.3984028
## yes       0.1848782       0.6405426      0.1547350      0.3411203
##     age_group41-50 age_group51-60 age_group61+ duration_group3-6 min
## no       0.2557891      0.1799044   0.01770976             0.2906740
## yes      0.1939211      0.1537302   0.09394624             0.2906305
##     duration_group6-10 min duration_group10-15 min duration_group15-20 min
## no               0.1092325              0.03485147             0.009021953
## yes              0.2275810              0.18638533             0.094197438
##     duration_group20-25 min duration_group25+ min
## no              0.002973903            0.00237244
## yes             0.040190907            0.02612409
```

```
#Using the trained QDA model predict using the Simplified Random Sampling Test data subset
set.seed(1)
qda.pred.1=predict(qda.fit.1, Test1)
table(qda.pred.1$class, Test1$y)
```

```
##
##         no  yes
##   no  9128  523
##   yes  867  785
```

```
paste('Misclassification rate for full QDA model: ',
      100-round(100*mean(qda.pred.1$class!=Test1$y),2),'%',sep='')
```

```
## [1] "Misclassification rate for full QDA model: 87.7%"
```

We can see from above that the accuracy of our full QDA model is 87.1%. Which is close but not as accurate as our LDA model's accuracy of 89.9%.

Evaluating the QDA Models Performance using 10 Fold Cross Validation (Simple Random Sampling)

We will again use 10 Fold Cross Validation in order to evaluate our QDA models performance.

```
#Create folds
set.seed(1)
folds.qda = createFolds(factor(banking_marketing_dataset$y), k=10)
```

```
#Create formula for calculating the misclassification rate
misclassification_qda_1<-function(idx){
  Train<-banking_marketing_dataset[-idx,]
  Test<-banking_marketing_dataset[idx,]
  fit<-qda(y~., data=Train)
  pred<-predict(fit,Test)
  return(mean(pred$class==Test$y))
}
```

```
mis_rate_qda = lapply(folds.qda, misclassification_qda_1)
```

```
paste('Accuracy rate for k-fold QDA model: ',
      round(100*mean(as.numeric(mis_rate_qda)),2),'%',sep='')
```

```
## [1] "Accuracy rate for k-fold QDA model: 87.47%"
```

We can see from below that the accuracy of the QDA model is 87.47% using 10 Fold Cross Validation. This aligns very closely with the accuracy we found above (87.7%). This gives us good confidence in our models performance.

6.4 Quadratic Discriminant Analysis (QDA) (Undersampling)

```
#Fit a QDA model to the under sampled Train data set.
qda.fit.2<-qda(y~., data=Train2)
qda.fit.2
```

```
## Call:
## qda(y ~ ., data = Train2)
##
## Prior probabilities of groups:
##        no       yes
## 0.4984833 0.5015167
##
## Group means:
##          age jobblue-collar jobentrepreneur jobhousemaid jobmanagement
## no  40.88235      0.2223631      0.03042596   0.03093306     0.1975152
## yes 41.62198      0.1353327      0.02242944   0.02116935     0.2414315
##     jobretired jobself-employed jobservices jobstudent jobtechnician
## no  0.04994929       0.03144016  0.09355984 0.01495943     0.1744422
## yes 0.09727823       0.03301411  0.07031250 0.04863911     0.1633065
##     jobunemployed  jobunknown maritalmarried maritalsingle educationsecondary
## no     0.02712982 0.007860041      0.6179006     0.2687627          0.5195233
## yes    0.03931452 0.005544355      0.5183972     0.3623992          0.4662298
##     educationtertiary educationunknown defaultyes  balance housingyes
## no           0.275355       0.03752535 0.01698783 1326.535  0.5834178
## yes          0.375252       0.04763105 0.01008065 1848.906  0.3707157
##        loanyes contacttelephone contactunknown      day  monthaug    monthdec
## no  0.16379310       0.06896552      0.3108519 15.80629 0.1364097 0.003549696
## yes 0.09072581       0.07510081      0.1003024 15.08443 0.1335685 0.019153226
##       monthfeb   monthjan  monthjul   monthjun    monthmar  monthmay   monthnov
```

```
## no  0.05400609 0.02535497 0.1577079 0.12119675 0.007606491 0.3166836 0.09051724
## yes 0.08492944 0.02444556 0.1154234 0.09929435 0.047379032 0.1738911 0.07686492
##       monthoct   monthsep duration campaign    pdays previous poutcomeother
## no  0.01090264 0.01191684 223.3702 2.787272 37.33342 0.525355    0.03727181
## yes 0.05947581 0.05241935 535.7694 2.125252 68.97681 1.179940    0.05695565
##     poutcomesuccess poutcomeunknown age_group26-30 age_group31-40
## no       0.01419878       0.8369675      0.1184077      0.3985801
## yes      0.18447581       0.6396169      0.1532258      0.3460181
##     age_group41-50 age_group51-60 age_group61+ duration_group3-6 min
## no       0.2654665      0.1736815   0.01926978             0.2958925
## yes      0.1932964      0.1504536   0.09450605             0.2847782
##     duration_group6-10 min duration_group10-15 min duration_group15-20 min
## no               0.1090264              0.03625761             0.007606491
## yes              0.2295867              0.18775202             0.093497984
##     duration_group20-25 min duration_group25+ min
## no              0.003296146           0.002281947
## yes             0.040826613           0.025201613
```

```
#Using the trained QDA model predict using the undersampled Test data subset
set.seed(1)
qda.pred.2=predict(qda.fit.2, Test2)
table(qda.pred.2$class, Test2$y)
```

```
##
##         no  yes
##   no  1176  478
##   yes  140  843
```

```
paste('Accuracy rate for full QDA model: ',
      100-round(100*mean(qda.pred.2$class!=Test2$y),2),'%',sep='')
```

```
## [1] "Accuracy rate for full QDA model: 76.56%"
```

We can see from above that the accuracy of our full QDA model is 72.7% using the undersampled data sets.

6.4.1 Evaluating the QDA Models Performance using 10 Fold Cross Validation (Undersampling)

We will again use 10 Fold Cross Validation in order to evaluate our QDA models performance.

```
#Create folds
set.seed(1)
folds.qda.2 = createFolds(factor(undersampling_df$y), k=10)
```

```
#Create a formula for calculating the misclassificaiton rate
misclassification_qda_2<-function(idx){
  Train<-undersampling_df[-idx,]
  Test<-undersampling_df[idx,]
  fit<-qda(y~., data=Train)
  pred<-predict(fit,Test)
  return(mean(pred$class==Test$y))
}
```

```
mis_rate_qda_2 = lapply(folds.qda.2, misclassification_qda_2)
```

```
paste('Accuracy rate for k-fold QDA model: ',
      round(100*mean(as.numeric(mis_rate_qda_2)),2),'%',sep='')
```

```
## [1] "Accuracy rate for k-fold QDA model: 76.57%"
```

We can see from above that the accuracy of the QDA model is 72.46% using 10 Fold Cross Validation. This aligns very closely with the accuracy we found above (72.70%).

However, when we used the simple random sample set we found the accuracy of the QDA model to be 86.94% using 10 Fold Cross Validation. We can see that again, using under sampling data sets has again resulted in a inferior models vs simple random sampling. The same phenomenon was observed with respect to our accuracy in the LDA models.

6.6 Logistic Regression Model (Simple Random Sampling)

In order to fit our data to a logistic regression model we need to ensure the following assumptions are met: "binary" outcome, "independence", no "multicollinearity" exists. As discussed earlier we have our dependent or response variable "y" of which there are only two possible outcomes "yes" and "no" which represent a person's willingness to sign up for a term account. We have already discussed how we have no reason to suspect our data points are NOT independent and we will investigate whether any multicollinearity exists in our model.

```
#Train a Logistic Regression model base on the simple random sample Train data subset
logistic.fit.1<-glm(factor(y)~., family = binomial, data = Train1)
summary(logistic.fit.1)
```

```
##
## Call:
## glm(formula = factor(y) ~ ., family = binomial, data = Train1)
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -2.026e+00  2.827e-01  -7.167 7.64e-13 ***
## age                      -8.326e-03  7.223e-03  -1.153 0.249060
## jobblue-collar           -4.110e-01  8.493e-02  -4.840 1.30e-06 ***
## jobentrepreneur          -3.640e-01  1.455e-01  -2.502 0.012342 *
## jobhousemaid             -5.195e-01  1.573e-01  -3.302 0.000960 ***
## jobmanagement            -1.636e-01  8.634e-02  -1.894 0.058185 .
## jobretired               -2.891e-01  1.296e-01  -2.230 0.025734 *
## jobself-employed         -3.217e-01  1.298e-01  -2.479 0.013180 *
## jobservices              -1.780e-01  9.589e-02  -1.856 0.063387 .
## jobstudent                1.649e-01  1.355e-01   1.217 0.223788
## jobtechnician            -1.859e-01  8.059e-02  -2.307 0.021075 *
## jobunemployed            -2.596e-01  1.332e-01  -1.949 0.051327 .
## jobunknown               -5.756e-01  2.758e-01  -2.087 0.036905 *
## maritalmarried           -2.503e-01  6.778e-02  -3.693 0.000222 ***
## maritalsingle            -7.154e-02  7.872e-02  -0.909 0.363478
## educationsecondary        1.678e-01  7.605e-02   2.206 0.027387 *
## educationtertiary         4.075e-01  8.884e-02   4.587 4.49e-06 ***
## educationunknown          1.744e-01  1.234e-01   1.413 0.157718
```

```
## defaultyes              -3.584e-02  1.906e-01  -0.188 0.850873
## balance                  1.065e-05  6.539e-06   1.628 0.103492
## housingyes              -6.540e-01  5.149e-02 -12.701  < 2e-16 ***
## loanyes                 -4.663e-01  6.984e-02  -6.677 2.45e-11 ***
## contacttelephone        -1.480e-01  8.914e-02  -1.660 0.096952 .
## contactunknown          -1.471e+00  8.291e-02 -17.747  < 2e-16 ***
## day                      1.297e-02  2.921e-03   4.439 9.04e-06 ***
## monthaug                -5.661e-01  9.361e-02  -6.048 1.47e-09 ***
## monthdec                 4.906e-01  2.143e-01   2.289 0.022076 *
## monthfeb                 9.427e-02  1.061e-01   0.888 0.374326
## monthjan                -1.210e+00  1.451e-01  -8.341  < 2e-16 ***
## monthjul                -7.203e-01  9.024e-02  -7.982 1.44e-15 ***
## monthjun                 5.662e-01  1.106e-01   5.119 3.07e-07 ***
## monthmar                 1.835e+00  1.449e-01  12.661  < 2e-16 ***
## monthmay                -3.233e-01  8.507e-02  -3.801 0.000144 ***
## monthnov                -7.777e-01  9.967e-02  -7.803 6.03e-15 ***
## monthoct                 9.729e-01  1.280e-01   7.599 2.98e-14 ***
## monthsep                 9.880e-01  1.438e-01   6.869 6.45e-12 ***
## duration                 2.435e-03  2.937e-04   8.291  < 2e-16 ***
## campaign                -8.006e-02  1.159e-02  -6.906 4.99e-12 ***
## pdays                   -6.438e-05  3.610e-04  -0.178 0.858445
## previous                 6.162e-03  6.408e-03   0.962 0.336297
## poutcomeother            1.333e-01  1.070e-01   1.246 0.212779
## poutcomesuccess          2.311e+00  9.902e-02  23.340  < 2e-16 ***
## poutcomeunknown         -1.487e-01  1.096e-01  -1.357 0.174909
## age_group26-30          -5.328e-01  1.181e-01  -4.512 6.41e-06 ***
## age_group31-40          -6.782e-01  1.365e-01  -4.969 6.72e-07 ***
## age_group41-50          -6.070e-01  1.909e-01  -3.179 0.001477 **
## age_group51-60          -5.060e-01  2.510e-01  -2.016 0.043826 *
## age_group61+             4.999e-01  3.464e-01   1.443 0.148928
## duration_group3-6 min    9.755e-01  7.464e-02  13.070  < 2e-16 ***
## duration_group6-10 min   1.452e+00  1.218e-01  11.923  < 2e-16 ***
## duration_group10-15 min  2.086e+00  1.949e-01  10.703  < 2e-16 ***
## duration_group15-20 min  2.239e+00  2.879e-01   7.775 7.55e-15 ***
## duration_group20-25 min  1.954e+00  3.873e-01   5.046 4.51e-07 ***
## duration_group25+ min    3.042e-01  5.264e-01   0.578 0.563401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24531  on 33907  degrees of freedom
## Residual deviance: 15539  on 33854  degrees of freedom
## AIC: 15647
##
## Number of Fisher Scoring iterations: 6
```

We can see below that the GVIF values of our logistic regression model are all less than 1.5. This is indicative of low multicollinearity.

```
#Checking the multicollinearity of our LR model
vif(logistic.fit.1)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
```

```
## age            17.335967  1        4.163648
## job             6.202055 11        1.086487
## marital         1.509486  2        1.108427
## education       2.322671  3        1.150795
## default         1.018357  1        1.009137
## balance         1.049689  1        1.024543
## housing         1.465241  1        1.210471
## loan            1.068506  1        1.033686
## contact         2.031302  2        1.193833
## day             1.336890  1        1.156239
## month           4.192069 11        1.067314
## duration       21.647092  1        4.652644
## campaign        1.109135  1        1.053155
## pdays           3.714435  1        1.927287
## previous        1.198481  1        1.094752
## poutcome        4.240216  3        1.272227
## age_group      29.232062  5        1.401477
## duration_group 23.109011  6        1.299118
```

```
#Using the trained LR model predict using the simple random sample Test data subset
set.seed(1)
Prob.predict.1<-predict(logistic.fit.1, Test1, type="response")
Predict<-rep("no",dim(Test1)[1])
Predict[Prob.predict.1>=0.5]="yes"
Actual<-Test1$y
table(Predict, Actual)
```

```
##        Actual
## Predict   no  yes
##     no  9728  820
##     yes  267  488
```

```
paste('Accuracy rate for full logistic regression model: ',
      100-round(100*mean(Predict!=Actual),2),'%',sep='' )
```

```
## [1] "Accuracy rate for full logistic regression model: 90.38%"
```

The accuracy of our full Logistic regression model is 90.15%.

We will now attempt to improve on the accuracy of our model by removing any insignificant variables below in order to come up with a "reduced" logistic regression model.

```
#Train a reduced LR model on the simple random sampling Train data set
logistic.fit.reduced.1<-glm(factor(y)~job+marital+education+housing+loan+
                              contact+day+month+duration+campaign+poutcome,
                            family = binomial, data = Train1)
summary(logistic.fit.reduced.1)
```

```
##
## Call:
## glm(formula = factor(y) ~ job + marital + education + housing +
##     loan + contact + day + month + duration + campaign + poutcome,
```

```
##     family = binomial, data = Train1)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.520e+00  1.496e-01 -16.837  < 2e-16 ***
## jobblue-collar     -3.754e-01  8.446e-02  -4.445 8.80e-06 ***
## jobentrepreneur    -3.090e-01  1.426e-01  -2.167  0.03024 *
## jobhousemaid       -4.877e-01  1.554e-01  -3.139  0.00169 **
## jobmanagement      -1.520e-01  8.459e-02  -1.797  0.07232 .
## jobretired          2.035e-01  1.008e-01   2.019  0.04351 *
## jobself-employed   -2.663e-01  1.278e-01  -2.083  0.03723 *
## jobservices        -1.626e-01  9.545e-02  -1.703  0.08849 .
## jobstudent          4.208e-01  1.237e-01   3.402  0.00067 ***
## jobtechnician      -1.740e-01  7.938e-02  -2.192  0.02837 *
## jobunemployed      -2.653e-01  1.329e-01  -1.997  0.04583 *
## jobunknown         -3.891e-01  2.705e-01  -1.438  0.15041
## maritalmarried     -2.051e-01  6.663e-02  -3.079  0.00208 **
## maritalsingle       4.024e-02  7.195e-02   0.559  0.57599
## educationsecondary  1.829e-01  7.489e-02   2.442  0.01460 *
## educationtertiary   3.936e-01  8.672e-02   4.539 5.66e-06 ***
## educationunknown    1.931e-01  1.211e-01   1.594  0.11091
## housingyes         -6.877e-01  5.015e-02 -13.714  < 2e-16 ***
## loanyes            -4.980e-01  6.973e-02  -7.143 9.16e-13 ***
## contacttelephone   -1.208e-01  8.474e-02  -1.426  0.15388
## contactunknown     -1.595e+00  8.384e-02 -19.026  < 2e-16 ***
## day                 1.224e-02  2.871e-03   4.263 2.02e-05 ***
## monthaug           -7.083e-01  9.112e-02  -7.773 7.67e-15 ***
## monthdec            4.732e-01  2.092e-01   2.262  0.02371 *
## monthfeb            3.223e-03  1.028e-01   0.031  0.97499
## monthjan           -1.260e+00  1.426e-01  -8.836  < 2e-16 ***
## monthjul           -7.688e-01  8.843e-02  -8.694  < 2e-16 ***
## monthjun            5.149e-01  1.081e-01   4.764 1.89e-06 ***
## monthmar            1.623e+00  1.373e-01  11.824  < 2e-16 ***
## monthmay           -3.517e-01  8.328e-02  -4.224 2.40e-05 ***
## monthnov           -8.706e-01  9.741e-02  -8.937  < 2e-16 ***
## monthoct            9.093e-01  1.215e-01   7.485 7.13e-14 ***
## monthsep            9.772e-01  1.377e-01   7.094 1.30e-12 ***
## duration            4.178e-03  7.422e-05  56.297  < 2e-16 ***
## campaign           -9.284e-02  1.169e-02  -7.941 2.01e-15 ***
## poutcomeother       1.790e-01  1.034e-01   1.731  0.08343 .
## poutcomesuccess     2.308e+00  9.204e-02  25.082  < 2e-16 ***
## poutcomeunknown    -1.053e-01  6.586e-02  -1.598  0.10995
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24531  on 33907  degrees of freedom
## Residual deviance: 16217  on 33870  degrees of freedom
## AIC: 16293
##
## Number of Fisher Scoring iterations: 6
```

We can see below that the GVIF values of our "reduced" logistic regression model are all less than 1.2. This

is indicative of low multicollinearity.

```
#Checking the multicollinearity of our LR model
vif(logistic.fit.reduced.1)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## job       2.956930 11        1.050514
## marital   1.179055  2        1.042038
## education 2.225484  3        1.142626
## housing   1.408700  1        1.186887
## loan      1.052056  1        1.025698
## contact   1.867491  2        1.169000
## day       1.340996  1        1.158014
## month     3.606075 11        1.060034
## duration  1.130414  1        1.063209
## campaign  1.101037  1        1.049303
## poutcome  1.230088  3        1.035117
```

```
#Using the trained reduced LR model predict using the simple random sample Test data subset
set.seed(1)
Prob.predict.reduced.1<-predict(logistic.fit.reduced.1, Test1, type="response")
Predict<-rep("no",dim(Test1)[1])
Predict[Prob.predict.reduced.1>=0.5]="yes"
Actual<-Test1$y
table(Predict, Actual)
```

```
##        Actual
## Predict   no  yes
##     no  9744  859
##     yes  251  449
```

```
paste('Accuracy rate for our reduced logistic regression model: ', 100-round(100*mean(Predict!=Actual),
```

```
## [1] "Accuracy rate for our reduced logistic regression model: 90.18%"
```

We have managed to only slightly improve on the accuracy of our full logistic regression model by removing some insignificant variables. We will now use 10 Fold Cross Validation in order to re-enforce the +/- 90% accuracy of our logistic regression model.

Evaluating the Logistic Regression Model Performance using 10 Fold Cross Validation (Simple Random Sampling)

We can see from below that the accuracy of the logistic regression model is 90.16% using 10 Fold Cross Validation. This aligns very closely with the accuracy we found above (90.18%). This gives us good confidence in our logistic regression model. The logistic regression model has so far given the highest accuracy (90.16%) vs linear discriminant analysis (89.96%) and quadratic discriminant analysis (87.47%) all with using simple random sampling.

```
#Create folds
set.seed(1)
folds.log.1 = createFolds(factor(banking_marketing_dataset$y), k=10)
```

```r
#Create a function that calculates the misclassification rate of the logistic regression model
misclassification_log_1<-function(idx){
  Train<-banking_marketing_dataset[-idx,]
  Test<-banking_marketing_dataset[idx,]
  logistic.fit<-glm(factor(y)~job+marital+education+housing+loan+contact+
                      day+month+duration+campaign+poutcome,
                    family = binomial, data = Train)
  accuracy = calc_misclass_log_1(logistic.fit, Test)
  return(accuracy)
}
```

```r
#Create a function that will calculate the misclassificaiton rate of a logistic regression model.
calc_misclass_log_1<-function(model,Test){
  Prob.predict<-predict(model, Test, type="response")
  Predict<-rep("no",dim(Test)[1])
  Predict[Prob.predict>=0.5]="yes"
  Actual<-Test$y
  #table(Predict, Actual)
  accuracy = 100-round(100*mean(Predict!=Actual),2)
  paste('Accuracy rate for logistic model fold: ', accuracy,'%',sep='' )
  return(accuracy)
}
```

```r
mis_rate_log_1 = lapply(folds.log.1, misclassification_log_1)
```

```r
paste('Accuracy rate for k-fold logistic regression model: ', round(mean(as.numeric(mis_rate_log_1)),2)
```

```
## [1] "Accuracy rate for k-fold logistic regression model: 90.16%"
```

We will now evaluate how using the undersampled data sets will affect the accuracy our our logistic regression model.

6.6 Logistic Regression Model (Undersampling)

```r
#Train a logistic regression model on the undersample data sets
logistic.fit.2<-glm(factor(y)~., family = binomial, data = Train2)
summary(logistic.fit.2)
```

```
##
## Call:
## glm(formula = factor(y) ~ ., family = binomial, data = Train2)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.495e-01  4.277e-01  -0.583 0.559651
## age                     -5.439e-03  1.105e-02  -0.492 0.622462
## jobblue-collar          -3.532e-01  1.247e-01  -2.834 0.004603 **
## jobentrepreneur         -2.591e-01  2.153e-01  -1.203 0.228806
## jobhousemaid            -5.465e-01  2.285e-01  -2.392 0.016751 *
## jobmanagement           -2.515e-01  1.302e-01  -1.931 0.053459 .
```

```
## jobretired              -7.512e-01  2.036e-01  -3.690 0.000224 ***
## jobself-employed        -4.874e-01  2.058e-01  -2.369 0.017856 *
## jobservices             -2.940e-01  1.439e-01  -2.043 0.041085 *
## jobstudent               1.732e-01  2.277e-01   0.760 0.447043
## jobtechnician           -3.129e-01  1.190e-01  -2.629 0.008552 **
## jobunemployed           -1.327e-01  2.005e-01  -0.662 0.508121
## jobunknown              -8.080e-01  4.097e-01  -1.972 0.048581 *
## maritalmarried          -2.180e-01  1.030e-01  -2.118 0.034214 *
## maritalsingle           -6.181e-02  1.206e-01  -0.512 0.608345
## educationsecondary       3.762e-01  1.109e-01   3.392 0.000693 ***
## educationtertiary        6.747e-01  1.320e-01   5.113 3.18e-07 ***
## educationunknown         5.427e-01  1.896e-01   2.862 0.004204 **
## defaultyes              -1.317e-01  2.750e-01  -0.479 0.632118
## balance                  2.481e-05  1.047e-05   2.369 0.017836 *
## housingyes              -7.152e-01  7.526e-02  -9.503  < 2e-16 ***
## loanyes                 -4.274e-01  1.001e-01  -4.271 1.95e-05 ***
## contacttelephone        -2.193e-01  1.349e-01  -1.626 0.103907
## contactunknown          -1.415e+00  1.148e-01 -12.327  < 2e-16 ***
## day                      7.057e-03  4.287e-03   1.646 0.099755 .
## monthaug                -7.503e-01  1.354e-01  -5.542 2.98e-08 ***
## monthdec                 6.123e-01  3.702e-01   1.654 0.098143 .
## monthfeb                 8.909e-02  1.575e-01   0.566 0.571547
## monthjan                -1.219e+00  2.227e-01  -5.474 4.40e-08 ***
## monthjul                -1.045e+00  1.367e-01  -7.646 2.08e-14 ***
## monthjun                 2.152e-01  1.604e-01   1.341 0.179761
## monthmar                 1.706e+00  2.430e-01   7.021 2.21e-12 ***
## monthmay                -7.031e-01  1.296e-01  -5.426 5.77e-08 ***
## monthnov                -8.730e-01  1.468e-01  -5.947 2.73e-09 ***
## monthoct                 1.294e+00  2.259e-01   5.729 1.01e-08 ***
## monthsep                 4.042e-01  2.191e-01   1.845 0.065062 .
## duration                 4.385e-03  5.437e-04   8.065 7.31e-16 ***
## campaign                -9.298e-02  1.624e-02  -5.726 1.03e-08 ***
## pdays                    1.306e-04  4.992e-04   0.262 0.793680
## previous                 1.566e-02  1.620e-02   0.967 0.333628
## poutcomeother            3.034e-01  1.644e-01   1.845 0.065010 .
## poutcomesuccess          2.439e+00  1.785e-01  13.663  < 2e-16 ***
## poutcomeunknown         -1.621e-01  1.628e-01  -0.995 0.319613
## age_group26-30          -4.730e-01  1.902e-01  -2.487 0.012900 *
## age_group31-40          -8.379e-01  2.151e-01  -3.895 9.84e-05 ***
## age_group41-50          -7.687e-01  2.949e-01  -2.607 0.009147 **
## age_group51-60          -5.534e-01  3.872e-01  -1.429 0.152954
## age_group61+             9.192e-01  5.453e-01   1.686 0.091878 .
## duration_group3-6 min    6.032e-01  1.119e-01   5.390 7.06e-08 ***
## duration_group6-10 min   9.302e-01  2.117e-01   4.394 1.11e-05 ***
## duration_group10-15 min  1.366e+00  3.459e-01   3.948 7.87e-05 ***
## duration_group15-20 min  1.139e+00  5.284e-01   2.155 0.031148 *
## duration_group20-25 min -2.006e-01  7.166e-01  -0.280 0.779533
## duration_group25+ min   -2.005e+00  9.124e-01  -2.198 0.027974 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10968  on 7911  degrees of freedom
```

```
## Residual deviance:  6260  on 7858  degrees of freedom
## AIC: 6368
##
## Number of Fisher Scoring iterations: 5
```

```
#Train a logistic regression model on the undersample data sets
set.seed(1)
Prob.predict.2<-predict(logistic.fit.2, Test2, type="response")
Predict<-rep("no",dim(Test2)[1])
Predict[Prob.predict.2>=0.5]="yes"
Actual<-Test2$y
table(Predict, Actual)
```

```
##        Actual
## Predict   no  yes
##     no  1120  214
##     yes  196 1107
```

```
paste('Accuracy rate for full logistic model: ',
      100-round(100*mean(Predict!=Actual),2),'%',sep='' )
```

```
## [1] "Accuracy rate for full logistic model: 84.45%"
```

The accuracy of the full logistic regression model using the under sampled data set is 83.96%, this pales in comparison to the 90.16% when we used the simple random sampled data sets. We will again attempt to improve on this accuracy by removing any insignificant variables from our full model.

```
#Train a reduced linear regression model on the under sampled Train data subset
logistic.fit.reduced.2<-glm(factor(y)~job+education+balance+housing+loan+
                              contact+month+duration+campaign+poutcome,
                            family = binomial, data = Train2)
summary(logistic.fit.reduced.2)
```

```
##
## Call:
## glm(formula = factor(y) ~ job + education + balance + housing +
##     loan + contact + month + duration + campaign + poutcome,
##     family = binomial, data = Train2)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -8.961e-01  1.897e-01  -4.723 2.32e-06 ***
## jobblue-collar     -3.728e-01  1.215e-01  -3.069 0.002149 **
## jobentrepreneur    -3.253e-01  2.116e-01  -1.537 0.124207
## jobhousemaid       -5.253e-01  2.194e-01  -2.394 0.016651 *
## jobmanagement      -2.680e-01  1.266e-01  -2.116 0.034336 *
## jobretired          3.006e-02  1.521e-01   0.198 0.843324
## jobself-employed   -4.339e-01  1.995e-01  -2.175 0.029663 *
## jobservices        -2.829e-01  1.411e-01  -2.005 0.044985 *
## jobstudent          6.369e-01  2.051e-01   3.105 0.001903 **
## jobtechnician      -3.031e-01  1.162e-01  -2.608 0.009108 **
## jobunemployed      -1.625e-01  1.955e-01  -0.831 0.406023
```

```
## jobunknown         -6.793e-01  3.903e-01  -1.740 0.081786 .
## educationsecondary  3.885e-01  1.070e-01   3.631 0.000283 ***
## educationtertiary   6.678e-01  1.263e-01   5.288 1.24e-07 ***
## educationunknown    5.705e-01  1.817e-01   3.140 0.001689 **
## balance             2.322e-05  1.010e-05   2.298 0.021541 *
## housingyes         -7.684e-01  7.246e-02 -10.604  < 2e-16 ***
## loanyes            -4.615e-01  9.814e-02  -4.703 2.56e-06 ***
## contacttelephone   -1.048e-01  1.254e-01  -0.836 0.403435
## contactunknown     -1.479e+00  1.119e-01 -13.212  < 2e-16 ***
## monthaug           -8.597e-01  1.294e-01  -6.645 3.04e-11 ***
## monthdec            6.415e-01  3.560e-01   1.802 0.071609 .
## monthfeb           -3.226e-02  1.452e-01  -0.222 0.824176
## monthjan           -1.090e+00  2.120e-01  -5.142 2.72e-07 ***
## monthjul           -1.064e+00  1.317e-01  -8.076 6.67e-16 ***
## monthjun            1.225e-01  1.497e-01   0.818 0.413238
## monthmar            1.577e+00  2.352e-01   6.706 2.00e-11 ***
## monthmay           -6.766e-01  1.239e-01  -5.461 4.74e-08 ***
## monthnov           -9.304e-01  1.407e-01  -6.614 3.73e-11 ***
## monthoct            1.220e+00  2.162e-01   5.642 1.68e-08 ***
## monthsep            4.693e-01  2.111e-01   2.223 0.026236 *
## duration            5.656e-03  1.529e-04  36.991  < 2e-16 ***
## campaign           -9.818e-02  1.604e-02  -6.123 9.21e-10 ***
## poutcomeother       3.416e-01  1.566e-01   2.181 0.029202 *
## poutcomesuccess     2.427e+00  1.701e-01  14.269  < 2e-16 ***
## poutcomeunknown    -1.917e-01  9.443e-02  -2.030 0.042316 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10968.3  on 7911  degrees of freedom
## Residual deviance:  6509.5  on 7876  degrees of freedom
## AIC: 6581.5
##
## Number of Fisher Scoring iterations: 6
```

```
#Using the trained reduced LR model predict using the undersampled Test data subset
set.seed(1)
Prob.predict.reduced.2<-predict(logistic.fit.reduced.2, Test2, type="response")
Predict<-rep("no",dim(Test2)[1])
Predict[Prob.predict.reduced.2>=0.5]="yes"
Actual<-Test2$y
table(Predict, Actual)
```

```
##        Actual
## Predict   no  yes
##     no  1129  236
##     yes  187 1085
```

```
paste('Accuracy rate for reduced logistic model: ',
      100-round(100*mean(Predict!=Actual),2),'%',sep='' )
```

```
## [1] "Accuracy rate for reduced logistic model: 83.96%"
```

As you can see from above, we have not managed to increase the accuracy of our full logistic regression model, as it is stagnant at 83.96%. We will attempt to validate this accuracy by using 10 Fold Cross Validation next.

Evaluating the Logistic Regression Model Performance using 10 Fold Cross Validation (Undersampling)

We can see below that the accuracy of the logistic regression model using 10 Fold Cross Validation does align again with out initial accuracy calculation of roughly 83%.

```
#Create folds
set.seed(1)
folds.log.2 = createFolds(factor(undersampling_df$y), k=10)
```

```
#Create a function that calculates the misclassificaiton rate
misclassification_log_2<-function(idx){
  Train<-undersampling_df[-idx,]
  Test<-undersampling_df[idx,]
  logistic.fit<-glm(factor(y)~job+education+balance+housing+loan
                    +contact+month+duration+campaign+poutcome,
                    family = binomial, data = Train)
  accuracy = calc_misclass_log_2(logistic.fit, Test)
  return(accuracy)
}
```

```
calc_misclass_log_2<-function(model,Test){
  Prob.predict<-predict(model, Test, type="response")
  Predict<-rep("no",dim(Test)[1])
  Predict[Prob.predict>=0.5]="yes"
  Actual<-Test$y
  #table(Predict, Actual)
  accuracy = 100-round(100*mean(Predict!=Actual),2)
  paste('Accuracy rate for logistic model fold: ', accuracy,'%',sep='' )
  return(accuracy)
}
```

```
mis_rate_log_2 = lapply(folds.log.2, misclassification_log_2)
```

```
paste('Accuracy rate for k-fold logistic regression model: ', round(mean(as.numeric(mis_rate_log_2)),2)
```

```
## [1] "Accuracy rate for k-fold logistic regression model: 83.1%"
```

6.7 Classification Tree Model (Simple Random Sampling)

Lastly, we will train a classification tree model. Classification tree models are considered very robust and do well with outliers and do not require any sort of linear relationships. The assumption of independence still applies however.

```
# Fit a classification tree based on the entire data set
tree_model<-tree(factor(y)~.,data=Train1, method="class")
```

```
## Warning in tree(factor(y) ~ ., data = Train1, method = "class"): NAs introduced
## by coercion
```

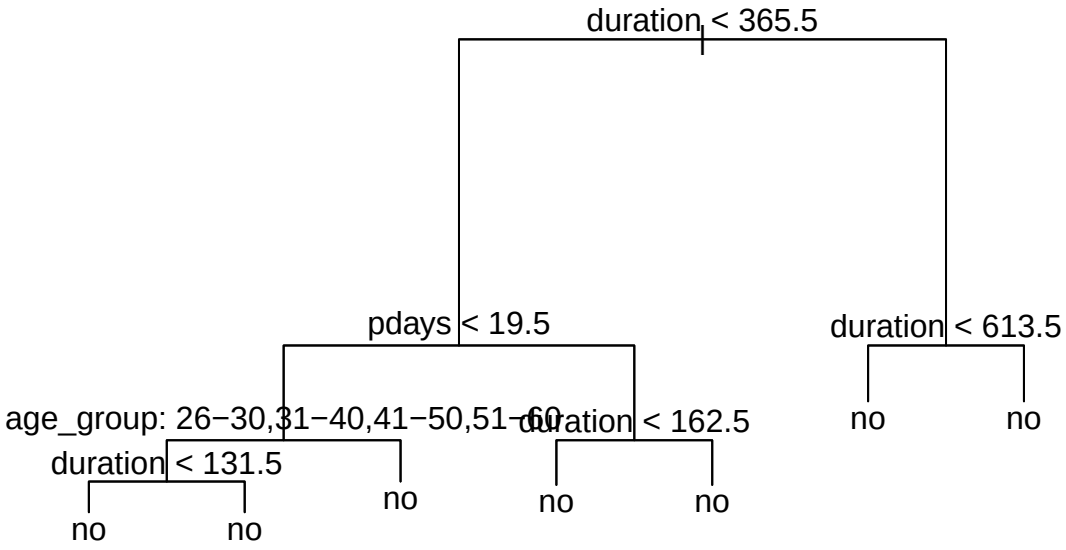
```
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = factor(y) ~ ., data = Train1, method = "class")
## Variables actually used in tree construction:
## [1] "duration"  "pdays"     "age_group"
## Number of terminal nodes:  7
## Residual mean deviance:  0.5554 = 18830 / 33900
## Misclassification error rate: 0.1174 = 3981 / 33908
```

```
tree_model
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 33908 24530 no ( 0.882594 0.117406 )
##    2) duration < 365.5 27037 12770 no ( 0.936605 0.063395 )
##      4) pdays < 19.5 22177  7346 no ( 0.960725 0.039275 )
##        8) age_group: 26-30,31-40,41-50,51-60 21149  5903 no ( 0.968604 0.031396 )
##         16) duration < 131.5 9643  1067 no ( 0.990148 0.009852 ) *
##         17) duration > 131.5 11506  4531 no ( 0.950548 0.049452 ) *
##        9) age_group: 18-25,61+ 1028  1033 no ( 0.798638 0.201362 ) *
##      5) pdays > 19.5 4860  4484 no ( 0.826543 0.173457 )
##       10) duration < 162.5 2486  1193 no ( 0.935237 0.064763 ) *
##       11) duration > 162.5 2374  2847 no ( 0.712721 0.287279 ) *
##    3) duration > 365.5 6871  8714 no ( 0.670063 0.329937 )
##      6) duration < 613.5 4160  4401 no ( 0.778365 0.221635 ) *
##      7) duration > 613.5 2711  3758 no ( 0.503873 0.496127 ) *
```

```
#Plot the Tree Classification Model
plot(tree_model)
text(tree_model ,pretty=0)
```




```
set.seed(1)

Predict<-predict(tree_model,Test1,type = "class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
Actual<-Test1$y
table(Predict,Actual)
```

```
##         Actual
## Predict   no  yes
##     no  9995 1308
##     yes    0    0
```

```
paste('Accuracy rate for Tree Classification model: ',
      100-round(100*mean(Predict!=Actual),2),'%',sep='' )
```

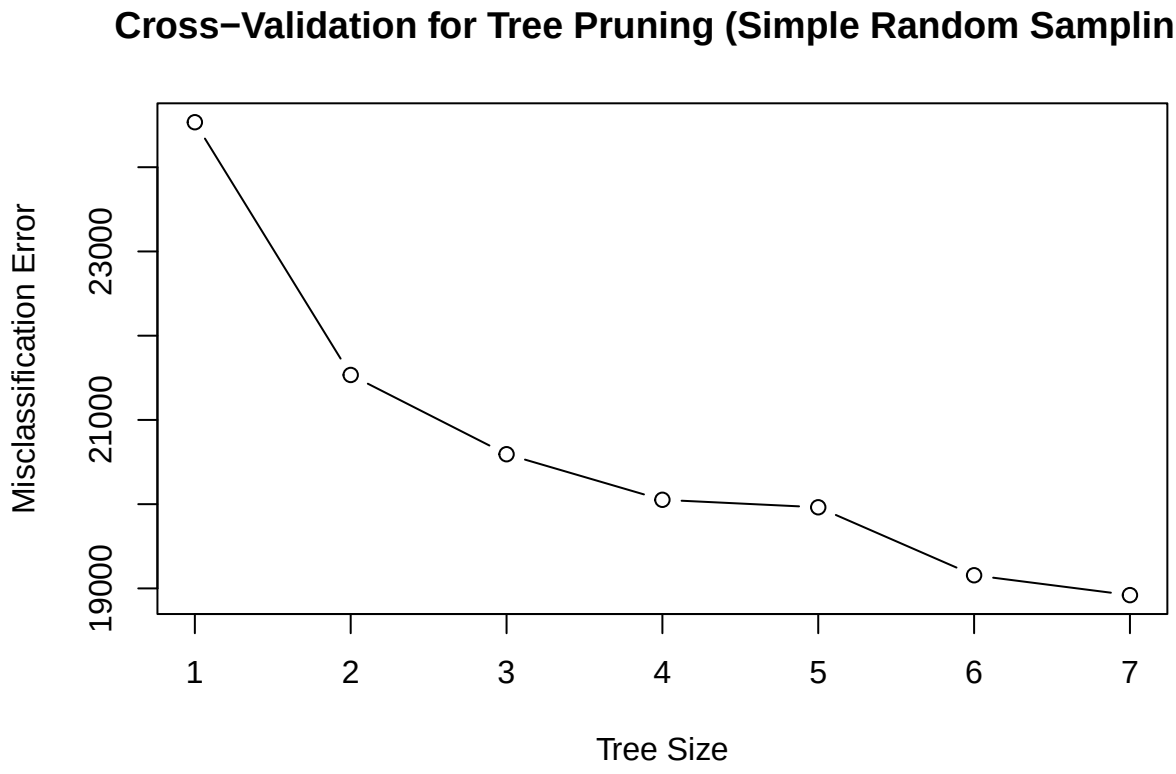
```
## [1] "Accuracy rate for Tree Classification model: 88.43%"
```

We can see the accuracy rate for our Tree Classification model is around 88.4%. We will look to prune our tree using 10 Fold Cross Validation to see if we can further increase its accuracy.

```r
# Perform cross-validation on the classification tree model
cv_tree<-cv.tree(tree_model)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
# Plot the cross-validation results
plot(cv_tree$size, cv_tree$dev,type="b",xlab="Tree Size",
     ylab="Misclassification Error",
     main = "Cross-Validation for Tree Pruning (Simple Random Sampling)")
```



From the plot above the optimum tree size appears to be the tree with 7 terminal nodes or our original unpruned tree.

Evaluating the Classification Tree Model Model Performance using 10 Fold Cross Validation (Simple Random Sampling)

```
#Create folds
set.seed(1)
folds.tc.1 = createFolds(factor(banking_marketing_dataset$y), k=10)
```

```
#Create a formula for calculating the accuracy of the classification tree
misclassification_tc_1<-function(idx){
  Train<-banking_marketing_dataset[-idx,]
  Test<-banking_marketing_dataset[idx,]
  fit<-tree(factor(y)~duration,data=Train, method="class")
  pred<-predict(fit, Test,type = "class")
  return(mean(pred==Test$y))
}
```

```
accuracy_tc_1 = lapply(folds.tc.1, misclassification_tc_1)
```

```
paste('Accuracy rate for k-fold Tree Classification model: ', round(100*mean(as.numeric(accuracy_tc_1))
```

```
## [1] "Accuracy rate for k-fold Tree Classification model: 88.34%"
```

The 10 Fold Cross Validation accuracy for our classification tree using the simple random sample data set is 88.34%. Next we will find the accuracy of a classification tree based on the undersampled data set.

6.8 Classification Tree Model (Undersampling)

```
# Fit a classification tree based on the entire data set
tree_model_us<-tree(factor(y)~.,data=Train2, method="class")
```

```
## Warning in tree(factor(y) ~ ., data = Train2, method = "class"): NAs introduced
## by coercion
```

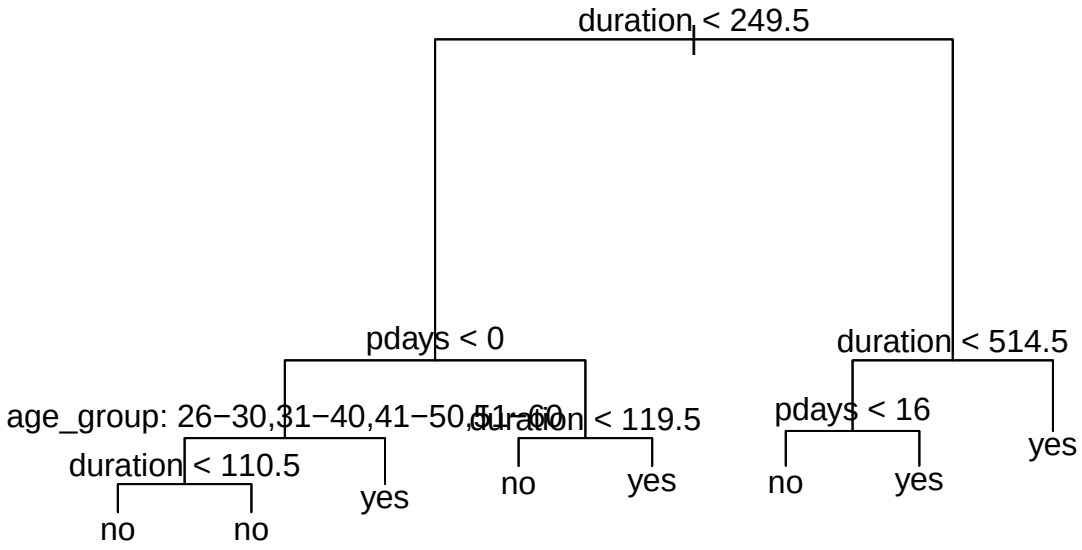
```
summary(tree_model_us)
```

```
##
## Classification tree:
## tree(formula = factor(y) ~ ., data = Train2, method = "class")
## Variables actually used in tree construction:
## [1] "duration"  "pdays"     "age_group"
## Number of terminal nodes:  8
## Residual mean deviance:  0.9968 = 7879 / 7904
## Misclassification error rate: 0.2414 = 1910 / 7912
```

```
tree_model_us
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 7912 10970.0 yes ( 0.49848 0.50152 )
##    2) duration < 249.5 3806  4437.0 no ( 0.73043 0.26957 )
##      4) pdays < 0 2870  2746.0 no ( 0.81533 0.18467 )
##        8) age_group: 26-30,31-40,41-50,51-60 2650  2221.0 no ( 0.85208 0.14792 )
##         16) duration < 110.5 1027   411.6 no ( 0.94937 0.05063 ) *
##         17) duration > 110.5 1623  1666.0 no ( 0.79051 0.20949 ) *
##        9) age_group: 18-25,61+ 220   290.6 yes ( 0.37273 0.62727 ) *
##      5) pdays > 0 936  1294.0 yes ( 0.47009 0.52991 )
##       10) duration < 119.5 275   298.4 no ( 0.76727 0.23273 ) *
##       11) duration > 119.5 661   853.0 yes ( 0.34644 0.65356 ) *
##    3) duration > 249.5 4106  4896.0 yes ( 0.28349 0.71651 )
##      6) duration < 514.5 2147  2904.0 yes ( 0.40848 0.59152 )
##       12) pdays < 16 1417  1964.0 no ( 0.50741 0.49259 ) *
##       13) pdays > 16 730   762.6 yes ( 0.21644 0.78356 ) *
##      7) duration > 514.5 1959  1632.0 yes ( 0.14650 0.85350 ) *
```

```
#Plot the Tree Classification Model
plot(tree_model_us)
text(tree_model_us ,pretty=0)
```



Here we can see a tree that has been trained with the undersampled data that the resulting tree contains 8 terminal nodes verses the tree trained on the simple random sample data sets only have 7 terminal nodes. Next we will evaluate the accuracy of this initial tree classification model.

```
set.seed(1)

Predict<-predict(tree_model_us,Test2,type = "class")
```

```
## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion
```

```
Actual<-Test2$y
table(Predict,Actual)
```

```
##         Actual
## Predict   no  yes
##     no  1080  400
##     yes  236  921
```

```r
paste('Accuracy rate for Tree Classification model: ',
      100-round(100*mean(Predict!=Actual),2),'%',sep='' )
```

```
## [1] "Accuracy rate for Tree Classification model: 75.88%"
```

We can see the accuracy rate for our Tree Classification model the accuracy is 75.73% when training and testing of the tree is done on the undersampled data sets. This again is much worse than the accuracy of 88.4% we achieved by using the simple random sampling data sets.

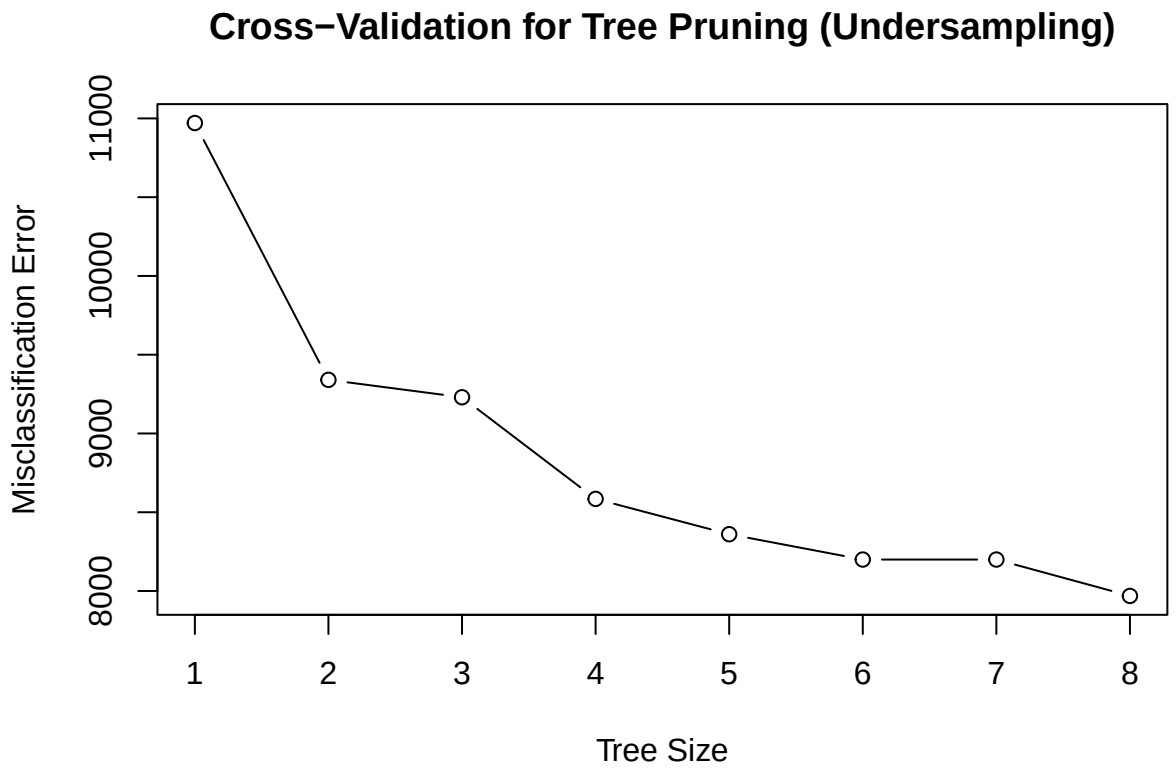
We will look to prune our tree using 10 Fold Cross Validation to see if we can further increase its accuracy.

```r
# Perform cross-validation on the classification tree model
cv_tree<-cv.tree(tree_model_us)
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```r
# Plot the cross-validation results
plot(cv_tree$size, cv_tree$dev,type="b",xlab="Tree Size",
     ylab="Misclassification Error",
     main = "Cross-Validation for Tree Pruning (Undersampling)")
```



Again much like before, the optimum classification tree would appear to have 8 terminal nodes. Therefore we will stick with our original unpruned tree and use 10 Fold Cross Validation to determine its accuracy.

Evaluating the Classification Tree Model Model Performance using 10 Fold Cross Validation (Undersampling)

```r
#Create folds
set.seed(1)
folds.tc.2 = createFolds(factor(undersampling_df$y), k=10)
```

```r
#Create a formula for calculating the accuracy of the classification tree
misclassification_tc_2<-function(idx){
  Train<-undersampling_df[-idx,]
  Test<-undersampling_df[idx,]
  fit<-tree(factor(y)~duration,data=Train, method="class")
  pred<-predict(fit, Test,type = "class")
  return(mean(pred==Test$y))
}
```

```r
accuracy_tc_2 = lapply(folds.tc.2, misclassification_tc_2)
```

```r
paste('Accuracy rate for k-fold Tree Classification model: ', round(100*mean(as.numeric(accuracy_tc_2))
```

```
## [1] "Accuracy rate for k-fold Tree Classification model: 71.74%"
```

The 10 Fold Cross Validation accuracy for our classification tree using the undersampled data set is 71.74%. Again, this pales in comparison to the 88.34% we achieved by using the simple random sample data set.

7. Results

7.1 Model Accuracy

The summary table below highlights the various model accuracy for the two sampling approaches taken. The accuracy of the Linear Discriminant Analysis and Linear Regression models both appear to be the best at around 90% when we have utilized a simple random sampling strategy and about 83% when we used the under sampling strategy.

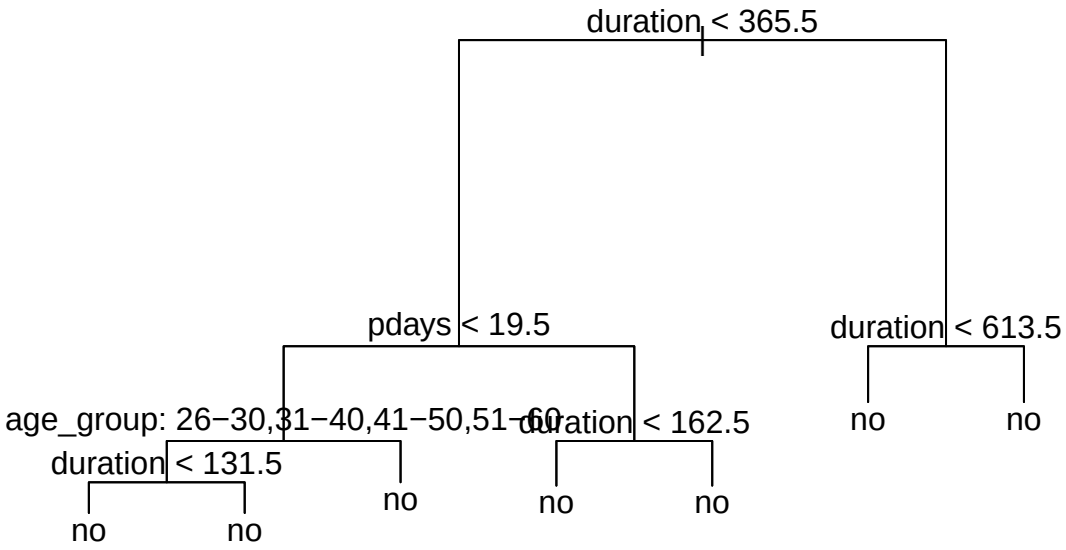
The worst model appears to be the Quadratic Discriminant Analysis model with an accuracy of around 87% but not far ahead is the Tree Classification mode at around 88%. All of our models appear to be well fitted (not over fitted) to the data as the 10 Fold CV accuracy is very close to the initial or model accuracy. The Tree Classification model does shoe some discrepancy between the 10 Fold CV accuracy and initial accuracy for the under sampled data set. It has likely been overfit to the under sampled Train data set as this data set is quit a bit smaller than our simple random sampling data set and these models can be more vulnerable to overfitting, hence the need for pruning.

	Simple Random		Undersampling	
Model / Accuracy	**Accuracy**	**10 Fold CV Accuracy**	**Accuracy**	**10 Fold CV Accuracy**
Linear Discriminant Analysis	90.20%	89.96%	83.31%	83.63%
Quadratic Descriminant Analysis	87.14%	86.94%	72.70%	72.46%
Logistic Regression	90.18%	90.16%	83.96%	83.10%
Tree Classification	88.43%	88.34%	75.73%	71.74%

7.2 Model Interpretations

The Tree Classification model is the easiest model to interpret. However, the shortfall it has is that it is a model built off of only 3 variables "duration", "pdays" and "age". There isn't a lot of information that can be pulled from the Tree Model and shared with a marketing team at a bank that may help increase the number of people who would sign up for a term deposit from a phone call.

```
#Plot the Tree Classification Model
plot(tree_model)
text(tree_model ,pretty=0)
```




```
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = factor(y) ~ ., data = Train1, method = "class")
## Variables actually used in tree construction:
## [1] "duration"  "pdays"     "age_group"
## Number of terminal nodes:  7
## Residual mean deviance:  0.5554 = 18830 / 33900
## Misclassification error rate: 0.1174 = 3981 / 33908
```

In looking at the summary of the Logistic Regression below, we are given quite a bit more information on what variables tend to have positive effect (i.e. push people more towards signing up for a term deposit) like

if they happen to be "retired" (positive co-efficient) or negative effect (i.e. make people less likely to signing up for a term deposit) such as if they have a "blue-collar" job.

The strength of these relationships (i.e. magnitude of the effect they have in swaying a person one way or another) is tied to the variables p-value. A smaller p-value indicates a stronger relationship with the resultant or dependent variable.

The table below gives a lot more information that can be shared with a marketing team at a bank in order to more effectively target the correct people and how to target them.

```
#Summary of the Logistic Regression model
summary(logistic.fit.1)
```

```
##
## Call:
## glm(formula = factor(y) ~ ., family = binomial, data = Train1)
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.026e+00  2.827e-01  -7.167 7.64e-13 ***
## age                     -8.326e-03  7.223e-03  -1.153 0.249060
## jobblue-collar          -4.110e-01  8.493e-02  -4.840 1.30e-06 ***
## jobentrepreneur         -3.640e-01  1.455e-01  -2.502 0.012342 *
## jobhousemaid            -5.195e-01  1.573e-01  -3.302 0.000960 ***
## jobmanagement           -1.636e-01  8.634e-02  -1.894 0.058185 .
## jobretired              -2.891e-01  1.296e-01  -2.230 0.025734 *
## jobself-employed        -3.217e-01  1.298e-01  -2.479 0.013180 *
## jobservices             -1.780e-01  9.589e-02  -1.856 0.063387 .
## jobstudent               1.649e-01  1.355e-01   1.217 0.223788
## jobtechnician           -1.859e-01  8.059e-02  -2.307 0.021075 *
## jobunemployed           -2.596e-01  1.332e-01  -1.949 0.051327 .
## jobunknown              -5.756e-01  2.758e-01  -2.087 0.036905 *
## maritalmarried          -2.503e-01  6.778e-02  -3.693 0.000222 ***
## maritalsingle           -7.154e-02  7.872e-02  -0.909 0.363478
## educationsecondary       1.678e-01  7.605e-02   2.206 0.027387 *
## educationtertiary        4.075e-01  8.884e-02   4.587 4.49e-06 ***
## educationunknown         1.744e-01  1.234e-01   1.413 0.157718
## defaultyes              -3.584e-02  1.906e-01  -0.188 0.850873
## balance                  1.065e-05  6.539e-06   1.628 0.103492
## housingyes              -6.540e-01  5.149e-02 -12.701  < 2e-16 ***
## loanyes                 -4.663e-01  6.984e-02  -6.677 2.45e-11 ***
## contacttelephone        -1.480e-01  8.914e-02  -1.660 0.096952 .
## contactunknown          -1.471e+00  8.291e-02 -17.747  < 2e-16 ***
## day                      1.297e-02  2.921e-03   4.439 9.04e-06 ***
## monthaug                -5.661e-01  9.361e-02  -6.048 1.47e-09 ***
## monthdec                 4.906e-01  2.143e-01   2.289 0.022076 *
## monthfeb                 9.427e-02  1.061e-01   0.888 0.374326
## monthjan                -1.210e+00  1.451e-01  -8.341  < 2e-16 ***
## monthjul                -7.203e-01  9.024e-02  -7.982 1.44e-15 ***
## monthjun                 5.662e-01  1.106e-01   5.119 3.07e-07 ***
## monthmar                 1.835e+00  1.449e-01  12.661  < 2e-16 ***
## monthmay                -3.233e-01  8.507e-02  -3.801 0.000144 ***
## monthnov                -7.777e-01  9.967e-02  -7.803 6.03e-15 ***
## monthoct                 9.729e-01  1.280e-01   7.599 2.98e-14 ***
## monthsep                 9.880e-01  1.438e-01   6.869 6.45e-12 ***
```

```
## duration                 2.435e-03  2.937e-04   8.291  < 2e-16 ***
## campaign                -8.006e-02  1.159e-02  -6.906 4.99e-12 ***
## pdays                   -6.438e-05  3.610e-04  -0.178 0.858445
## previous                 6.162e-03  6.408e-03   0.962 0.336297
## poutcomeother            1.333e-01  1.070e-01   1.246 0.212779
## poutcomesuccess          2.311e+00  9.902e-02  23.340  < 2e-16 ***
## poutcomeunknown         -1.487e-01  1.096e-01  -1.357 0.174909
## age_group26-30          -5.328e-01  1.181e-01  -4.512 6.41e-06 ***
## age_group31-40          -6.782e-01  1.365e-01  -4.969 6.72e-07 ***
## age_group41-50          -6.070e-01  1.909e-01  -3.179 0.001477 **
## age_group51-60          -5.060e-01  2.510e-01  -2.016 0.043826 *
## age_group61+             4.999e-01  3.464e-01   1.443 0.148928
## duration_group3-6 min    9.755e-01  7.464e-02  13.070  < 2e-16 ***
## duration_group6-10 min   1.452e+00  1.218e-01  11.923  < 2e-16 ***
## duration_group10-15 min  2.086e+00  1.949e-01  10.703  < 2e-16 ***
## duration_group15-20 min  2.239e+00  2.879e-01   7.775 7.55e-15 ***
## duration_group20-25 min  1.954e+00  3.873e-01   5.046 4.51e-07 ***
## duration_group25+ min    3.042e-01  5.264e-01   0.578 0.563401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 24531  on 33907  degrees of freedom
## Residual deviance: 15539  on 33854  degrees of freedom
## AIC: 15647
##
## Number of Fisher Scoring iterations: 6
```

8. Conclusion

The models that our group would use to determine if a person is more likely to sign up for term deposit is a Logistic Regression model due to its accuracy and its ease to interpret. The table below is sorted based on the magnitude that each independent variable has the dependent variable (whether someone will sign up for a term deposit). The variables that have a greater effect near the top. The variables that are highlighted in red have a negative effect on a persons decision.

Factor	Estimat	Std. Erro	z value	Pr(>|z|)
housingyes	-6.88E-01	5.06E-02	-13.591	2.00E-16
contactunknown	-1.59E+00	8.40E-02	-18.972	2.00E-16
monthjan	-1.25E+00	1.43E-01	-8.771	2.00E-16
monthjul	-7.65E-01	8.86E-02	-8.634	2.00E-16
monthmar	1.62E+00	1.37E-01	11.816	2.00E-16
monthnov	-8.77E-01	9.80E-02	-8.948	2.00E-16
duration	4.18E-03	7.42E-05	56.251	2.00E-16
poutcomesuccess	2.31E+00	9.50E-02	24.333	2.00E-16
campaign	-9.30E-02	1.17E-02	-7.949	1.88E-15
monthaug	-7.04E-01	9.13E-02	-7.713	1.22E-14
monthoct	9.06E-01	1.22E-01	7.445	9.71E-14
monthsep	9.78E-01	1.38E-01	7.096	1.29E-12
loanyes	-4.92E-01	7.01E-02	-7.02	2.21E-12
monthjun	5.13E-01	1.08E-01	4.741	2.13E-06
jobblue-collar	-3.76E-01	8.45E-02	-4.444	8.81E-06
educationtertiary	3.87E-01	8.74E-02	4.427	9.53E-06
day	1.22E-02	2.87E-03	4.259	2.06E-05
monthmay	-3.51E-01	8.34E-02	-4.215	2.50E-05
jobstudent	4.09E-01	1.26E-01	3.25	0.00115
maritalmarried	-2.12E-01	6.70E-02	-3.17	0.00153
jobhousemaid	-4.82E-01	1.56E-01	-3.09	0.002
educationsecondary	1.81E-01	7.53E-02	2.402	0.0163
monthdec	4.70E-01	2.10E-01	2.244	0.02483
jobtechnician	-1.74E-01	7.94E-02	-2.191	0.02846
jobentrepreneur	-3.06E-01	1.43E-01	-2.141	0.03229
jobself-employed	-2.68E-01	1.28E-01	-2.092	0.03647
jobunemployed	-2.68E-01	1.33E-01	-2.013	0.04409
jobretired	2.19E-01	1.12E-01	1.957	0.05029
jobmanagement	-1.54E-01	8.47E-02	-1.823	0.06836
jobservices	-1.62E-01	9.55E-02	-1.697	0.08961
poutcomeother	1.72E-01	1.04E-01	1.651	0.09883
balance	1.03E-05	6.32E-06	1.633	0.10242
educationunknown	1.94E-01	1.21E-01	1.597	0.1103
jobunknown	-3.86E-01	2.71E-01	-1.427	0.15366
contacttelephone	-1.21E-01	8.58E-02	-1.406	0.15971
previous	6.37E-03	6.36E-03	1.001	0.31678
poutcomeunknown	-7.17E-02	1.06E-01	-0.676	0.49923
age	-1.15E-03	2.54E-03	-0.452	0.65104
maritalsingle	2.68E-02	7.68E-02	0.349	0.72723
pdays	6.52E-05	3.48E-04	0.187	0.8515
defaultyes	-1.10E-02	1.88E-01	-0.059	0.95319
monthfeb	4.29E-03	1.03E-01	0.042	0.96678

9. Task Division

The tasks for our group project were divided up as such.

Project Proposal: Daksh Patel, Jacques Botha and Tomasz Szymczyk\ Exploratory Data Analysis: Shrivarshini Balaji and Tomasz Szymczyk\ Sampling Strategy: Daksh Patel and Jacques Botha\ Modelling: - Daksh Patel and Jacques Botha\ Project Presentation: Daksh Patel, Jacques Botha and Tomasz Szymczyk\ Compiling Final Report: Daksh Patel, Jacques Botha and Tomasz Szymczyk

10. References

1. S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014.

2. "Bank Marketing Data Set." UCI Machine Learning Repository: Bank Marketing Data Set, archive.ics.uci.edu/ml/datasets/bank+marketing. Accessed 24 May 2023.