HARVARDX-PH125.9x:CYO Capstone Project: Bank Marketing Campaign for Opening a Term Deposit

Deepika Dittakavi

6/21/2020

Contents

1. Introduction

Marketing Campaigns are often performed to create value for customers and build strong customer relationships by focusing on the customer needs and capture value from customers in return.

Marketing Campaigns usually focus on certain variables to devise a strategy. It mainly considers, who - the segment of population the campaign is trying to target, where/how - determines how to reach the customers, for example, by means of Telephone/TV/Radio or geographic location and finally what - the promotional offer which decides the best price to capture the customer.

For bank marketing campaigns the main interest is to open a checking/savings account or term deposit to keep up the regular operative cash flow activities. This project primarily focuses on marketing for Term Deposit accounts.

A term deposit account is an account that the bank or financial institution offers with a fixed rate, and the money will be returned after the maturity time specified in the offer. The term deposit account often offers better rate than a regular account.

1.1 Data Source

This dataset is a direct marketing campaign of a Portugese banking institution. The marketing campaigns were based on phone calls where multiple contacts were made to the same client. The dataset was downloaded from the Kaggle website. However, the website mentions that data was originally uploaded in the UCI Machine Learning Repository. The dataset gives information to analyze and find ways for strategies to improve future marketing campaigns for the bank.

Kaggle Dataset: https://www.kaggle.com/janiobachmann/bank-marketing-dataset

1.2 Project Goal

The primary goal of the project is to analyze data, identify patterns and predict which clients will open a term deposit account. The dependent variable is the outcome with two labels: “yes” or “no” to opening term deposit, hence the project uses Supervised Learning Classification.

1.3 Process Methodology

The process followed in the data analysis can be broadly classified into the following categories and this document follows a similar order.

-Problem Definition

-Identify Process

-Data Wrangling

-Data Exploration

-Machine Learning methods

-Results Comparison

-Conclusion

-Future Work

2. Data Wrangling

This section outlines the process followed in obtaining the data, initial setup and understanding the data.

2.1 Data Collection

The Kaggle website was used to download the data for the analysis. The CSV data file has been uploaded into this project's GitHub Repository and downloaded from there through the R code as shown below.

```
fileurl<-"https://raw.githubusercontent.com/adeepikaa/bankdeposit/master/bankdeposit.csv"
download.file(fileurl, "bank_data.csv")

bank_data<-read.csv("bank_data.csv")
```

The dataset can be found at : https://www.kaggle.com/janiobachmann/bank-marketing-dataset

2.2 Data Understanding

The dataset has 17 columns and 11162 rows. The data dictionary is given below.

```
# Data Dictionary:

#  "age"       : Age of the person 18-95
#  "job"       : Job, 22 types
#  "marital"   : Married, Single, Divorced
#  "education" : primary, secondary, tertiary, unknown
#  "default"   : has credit in default? yes/no
#  "balance"   : bank balance
#  "housing"   : has housing loan? yes/no
#  "loan"      : has personal load? yes/no
#  "contact"   : cell, telephone, unknown
#  "day"       : days of the month
#  "month"     : 12 months
#  "duration"  : duration of call, not to be used for predictions
#  "campaign"  : no. of contacts made to this person during this campaign
#  "pdays"     : number of days passed since last contact in previous campaign
#  "previous"  : number of contacts made before this campaign
#  "poutcome"  : outcome of previous marketing campaign
#  "deposit"   : Term Deposited, yes/no

head(bank_data)
```

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	cam
59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	
56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	
41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	
55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	
54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	
42	management	single	tertiary	no	0	yes	yes	unknown	5	may	562	

```
str(bank_data)
```

```
## 'data.frame':    11162 obs. of  17 variables:
##  $ age      : int  59 56 41 55 54 42 56 60 37 28 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 1 1 10 8 1 5 5 6 10 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 2 2 2 3 2 1 2 3 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 2 2 2 2 3 3 3 2 2 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  2343 45 1270 2476 184 0 830 545 1 5090 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 1 2 2 1 2 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 2 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ day      : int  5 5 5 5 5 5 6 6 6 6 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ duration : int  1042 1467 1389 579 673 562 1201 1030 608 1297 ...
##  $ campaign : int  1 1 1 1 2 2 1 1 1 3 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ deposit  : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

The dataset structure shows that age, job, marital status, education, default, housing, loan, contact, month, poutcome and deposit are all categorical variables and balance, day, duration, campaign, pdays, previous are numerical continuous variables.

2.3 Data Tidying

This dataset did not have any missing values or duplicated rows. The data is in tidy format and ready for further analysis.

```
sum(is.na(bank_data))
```

```
## [1] 0
```

```
nrow(unique(bank_data))
```

```
## [1] 11162
```

```
summary(bank_data)
```

```
##       age                 job           marital         education
##  Min.   :18.00   management :2566   divorced:1293   primary  :1500
##  1st Qu.:32.00   blue-collar:1944   married :6351   secondary:5476
##  Median :39.00   technician :1823   single  :3518   tertiary :3689
##  Mean   :41.23   admin.     :1334                   unknown  : 497
##  3rd Qu.:49.00   services   : 923
##  Max.   :95.00   retired    : 778
##                  (Other)    :1794
##  default        balance      housing     loan           contact
##  no :10994   Min.   :-6847   no :5881   no :9702   cellular :8042
```

```
##  yes:  168   1st Qu.:  122   yes:5281   yes:1460   telephone: 774
##              Median :  550                         unknown  :2346
##              Mean   : 1529
##              3rd Qu.: 1708
##              Max.   :81204
##
##       day            month         duration       campaign
##  Min.   : 1.00   may    :2824   Min.   :   2   Min.   : 1.000
##  1st Qu.: 8.00   aug    :1519   1st Qu.: 138   1st Qu.: 1.000
##  Median :15.00   jul    :1514   Median : 255   Median : 2.000
##  Mean   :15.66   jun    :1222   Mean   : 372   Mean   : 2.508
##  3rd Qu.:22.00   nov    : 943   3rd Qu.: 496   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 923   Max.   :3881   Max.   :63.000
##                  (Other):2217
##      pdays          previous          poutcome    deposit
##  Min.   : -1.00   Min.   : 0.0000   failure:1228   no :5873
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 537   yes:5289
##  Median : -1.00   Median : 0.0000   success:1071
##  Mean   : 51.33   Mean   : 0.8326   unknown:8326
##  3rd Qu.: 20.75   3rd Qu.: 1.0000
##  Max.   :854.00   Max.   :58.0000
##
```

A summary on all the columns of the dataset shows a preliminary spread of the data. It can be seen that the balance is negative for few customers. The pdays column has negative values at -1 for more than half of the customers indicating that more than half the customers were not contacted in the previous campaign. By running the following commands this has been verified through the previous column.

```
table(bank_data$pdays)
table(bank_data$previous)
```

2.4 Installing Packages and Libraries

The different packages and libraries needed to run the R code for data analysis will be loaded if the user does not have them already. The analysis uses the $tidyverse, gridExtra, caret, knitr, randomForest, MLmetrics$ and $ROCR $ packages.

3 Data Exploration & Visualization

The first step to data exploration is to analyze the dependent variable across different predictors. The dependent variable is called "deposit" which indicates if the customer has ended up making a deposit or not with a "yes" or "no".

The deposit column has an almost even spread in the data.

```
table(bank_data$deposit)
```

```
##
##   no  yes
## 5873 5289
```

3.1 Age

The customers were aged between 18 and 95 years with a median age of 39years. The below plot shows the spread of Age and deposit percent at different ages.

```
median(bank_data$age)
```

```
## [1] 39
```

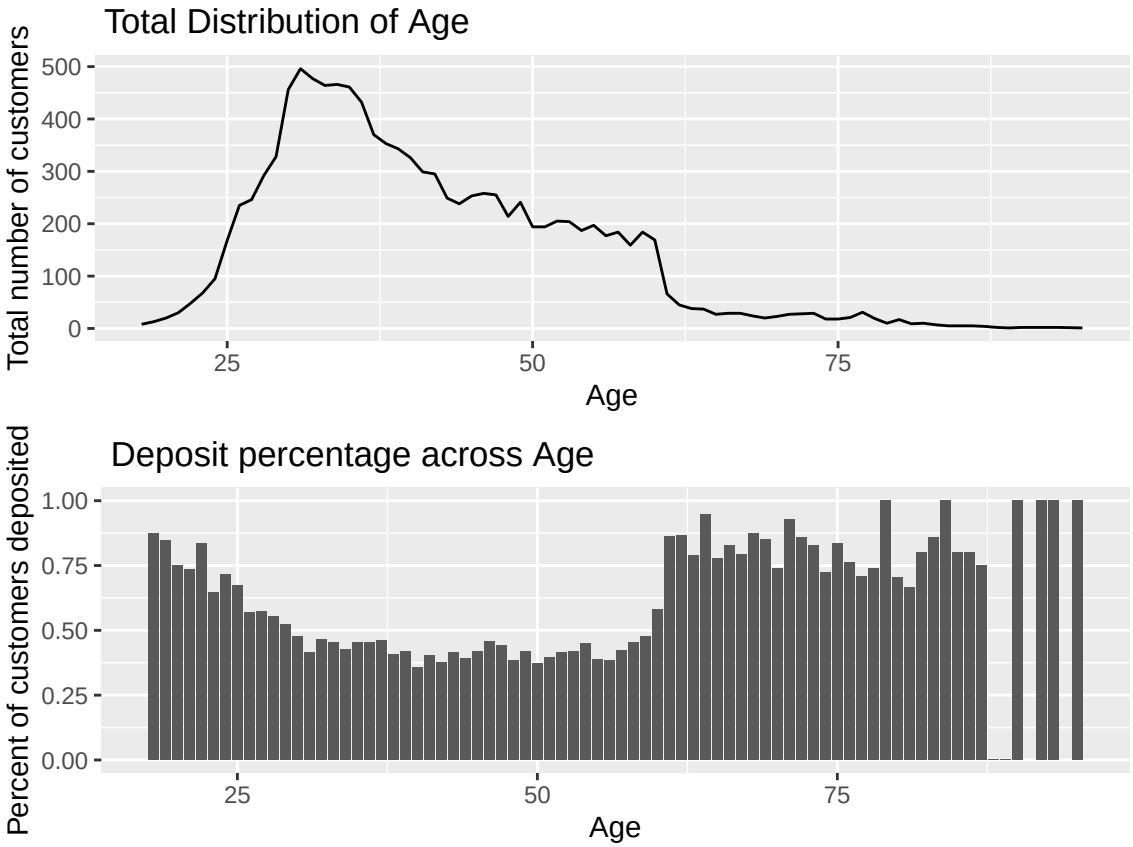
```
min(bank_data$age)
```

```
## [1] 18
```

```
max(bank_data$age)
```

```
## [1] 95
```

```
dist<-bank_data%>%
  group_by(age)%>%
  summarize(n=n(), .groups='drop')%>%
  ggplot(aes(x=age, y=n))+
  geom_line()+
  xlab("Age")+
  ylab("Total number of customers")+
  ggtitle(" Total Distribution of Age")

dist_dep<-bank_data%>%
  group_by(age)%>%
  summarize(n=n(), deposit_pct=sum(deposit=="yes")/n, .groups="drop")%>%
  ggplot(aes(x=age, y=deposit_pct))+
  geom_bar(stat="identity")+
  xlab("Age")+
  ylab("Percent of customers deposited")+
  ggtitle(" Deposit percentage across Age")

grid.arrange(dist, dist_dep, ncol=1)
```



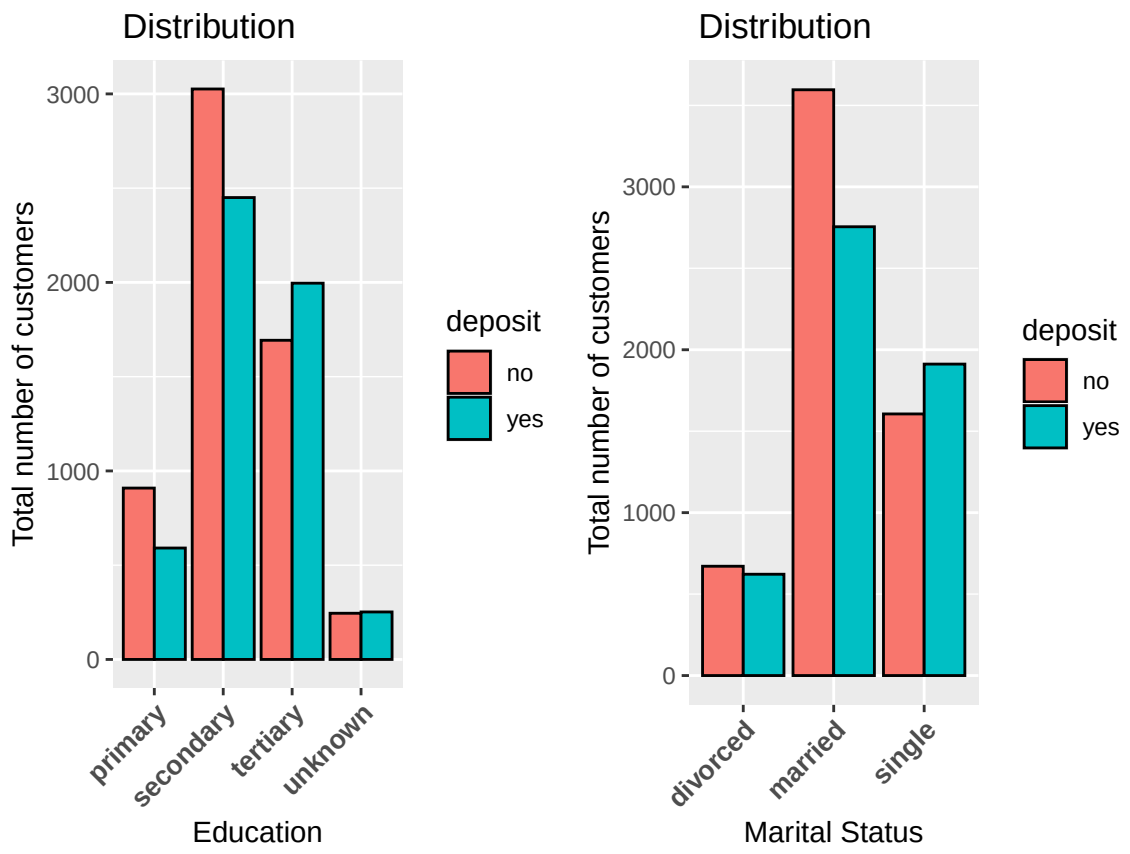
Most customers above 60 years and less than 25 years prefer to deposit. It can also be seen that though there are more customers around 30-50 years of age, a less percentage of them choose to deposit. Marketing campaigns should try to target this age group to increase customer base and continue to support the above 60 years and below 25 years age groups to retain customers.

3.2 Education and Marital Status

Data Visualizations of different educational backgrounds and marital status revealed that customers with secondary and tertiary education have shown most interest in depositing. The tertiary education group has a higher percent than the secondary education group of deposits.

```
e<-bank_data%>%
  ggplot(aes(x=education,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle(" Distribution")+
  xlab("Education")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

m<-bank_data%>%
  ggplot(aes(x=marital,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle(" Distribution")+
  xlab("Marital Status")+
```

```
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

grid.arrange(e,m, nrow=1)
```



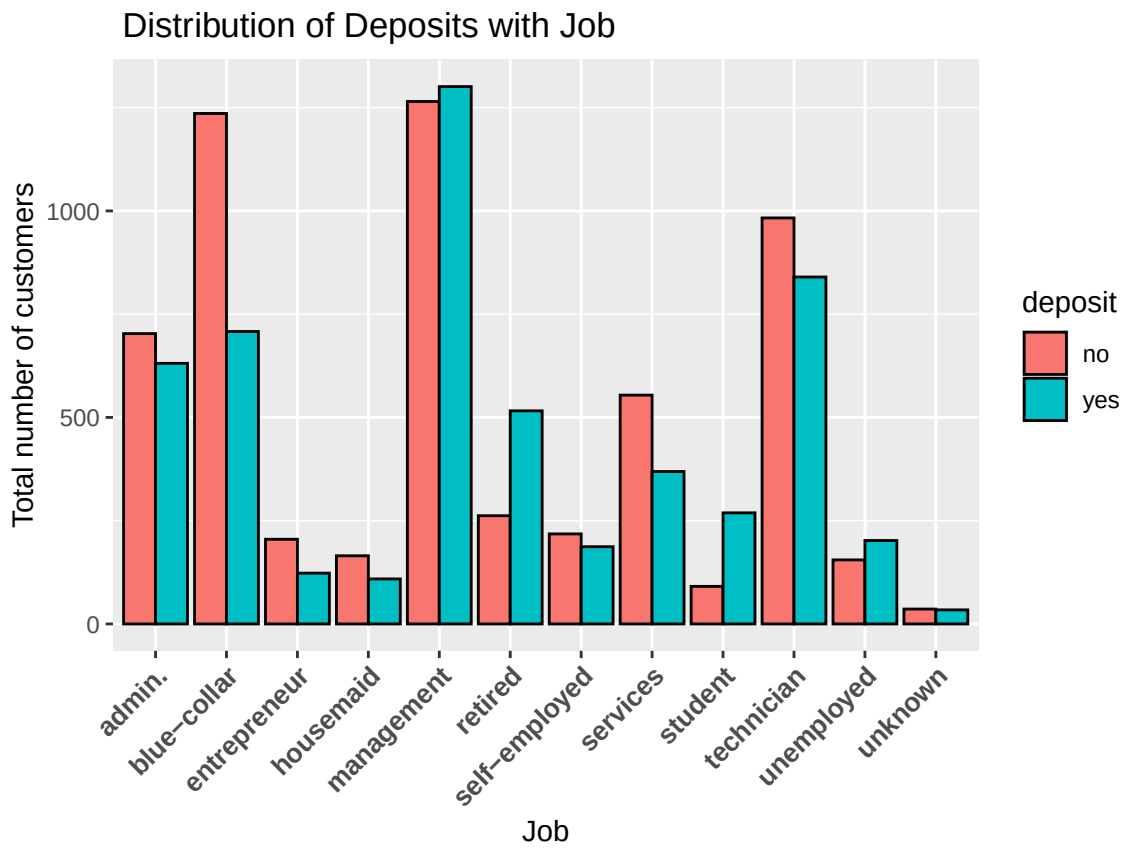
The number of deposits made by married customers though higher than the others has a lower percent as more number of married customers chose not to deposit.

Marketing campaigns should hence target the secondary education group and married group of customers to increase customers.

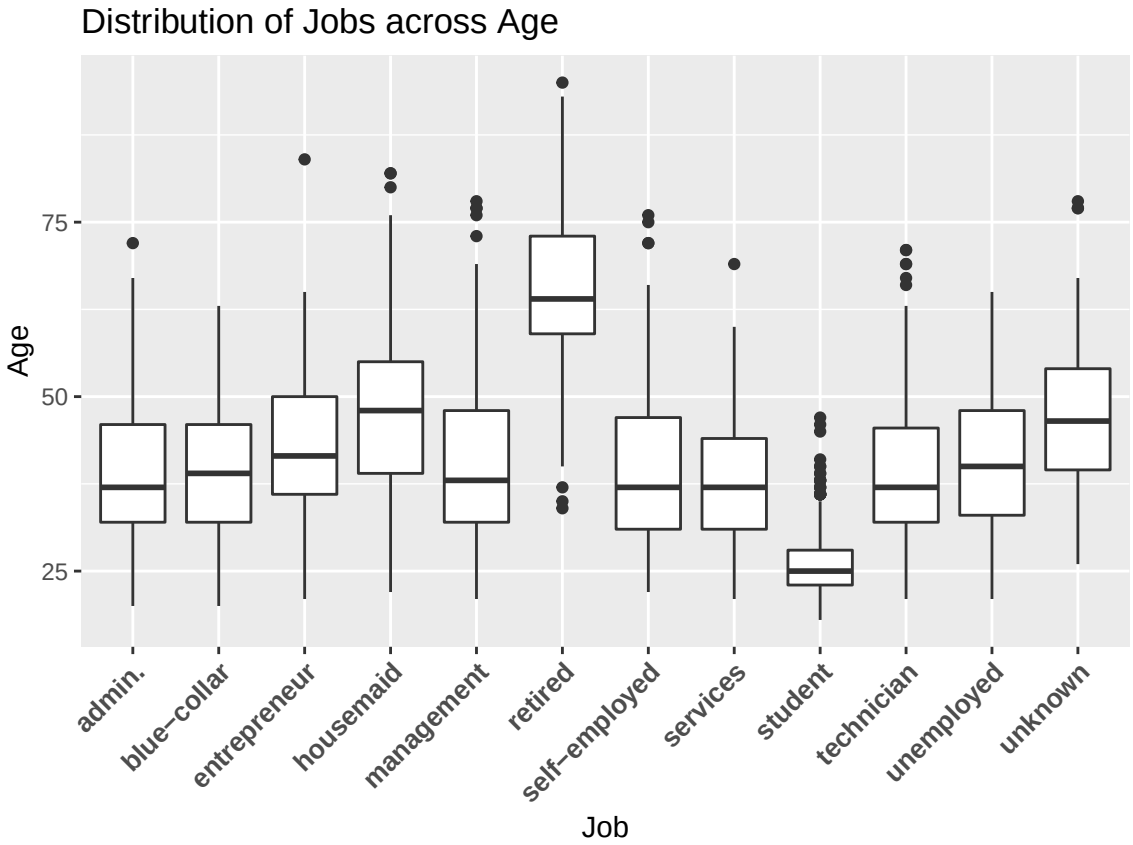
3.3 Job

Customers with Management jobs and customers who have retired are two groups that show higher interest in depositing though technicians and admin jobs are not far behind.

```
bank_data%>%
  ggplot(aes(x=job,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle(" Distribution of Deposits with Job")+
  xlab("Job")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
```




```
bank_data%>%
  ggplot()+
  geom_boxplot(aes(job, age))+
  ggtitle("Distribution of Jobs across Age")+
  xlab("Job")+
  ylab("Age")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
```



The above plot shows the distribution of jobs across different ages. Based on the two graphs, retired customers and students have shown lot of interest in doing deposits.

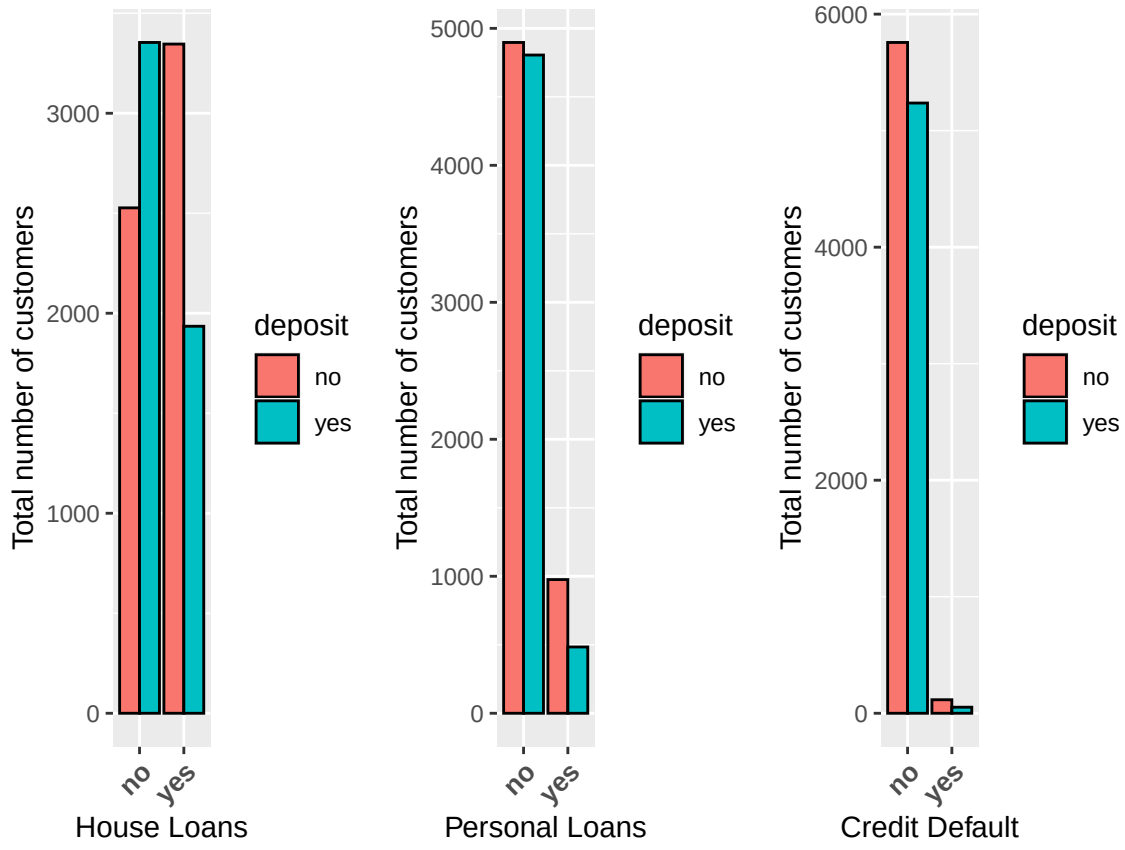
3.4 Loans and Credit default

The dataset consists of three financial categories - the home loan, the personal loan and credit default - that can potentially help in understanding if a person will do a deposit. Since most customers do not have credit defaults, even though it shows that customers with defaults almost do not deposit, this variable is not the strongest predictor. However, having a housing loan can give more insights. A higher percent of customers without housing loan, tend to deposit than the ones having a housing loan. Also, when having a personal loan lesser number of customers tend to deposit.

The Marketing Campaign should specifically segment the population to take these prior financial situations into consideration to approach the customers.

```
h<-bank_data%>%
  ggplot(aes(x=housing,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  xlab("House Loans")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

l<-bank_data%>%
  ggplot(aes(x=loan,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
```

```
  xlab("Personal Loans")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
d<-bank_data%>%
  ggplot(aes(x=default,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  xlab("Credit Default")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
grid.arrange(h,l,d, nrow=1)
```

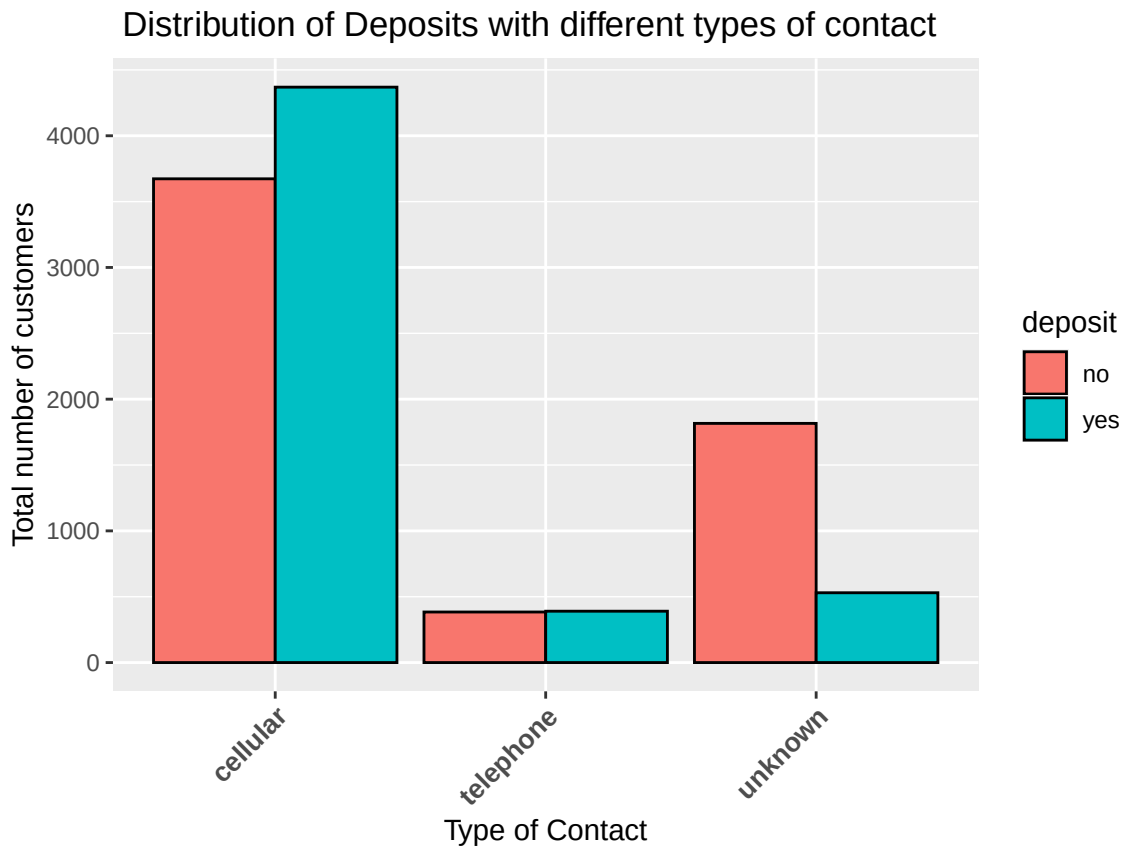


3.5 Contact type

The *typeofcontact* chart shows that mostly cellular contact was used and had better success than the others.

```
bank_data%>%
  ggplot(aes(x=contact,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle(" Distribution of Deposits with different types of contact")+
  xlab("Type of Contact")+
  ylab("Total number of customers")+
```

```
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
```

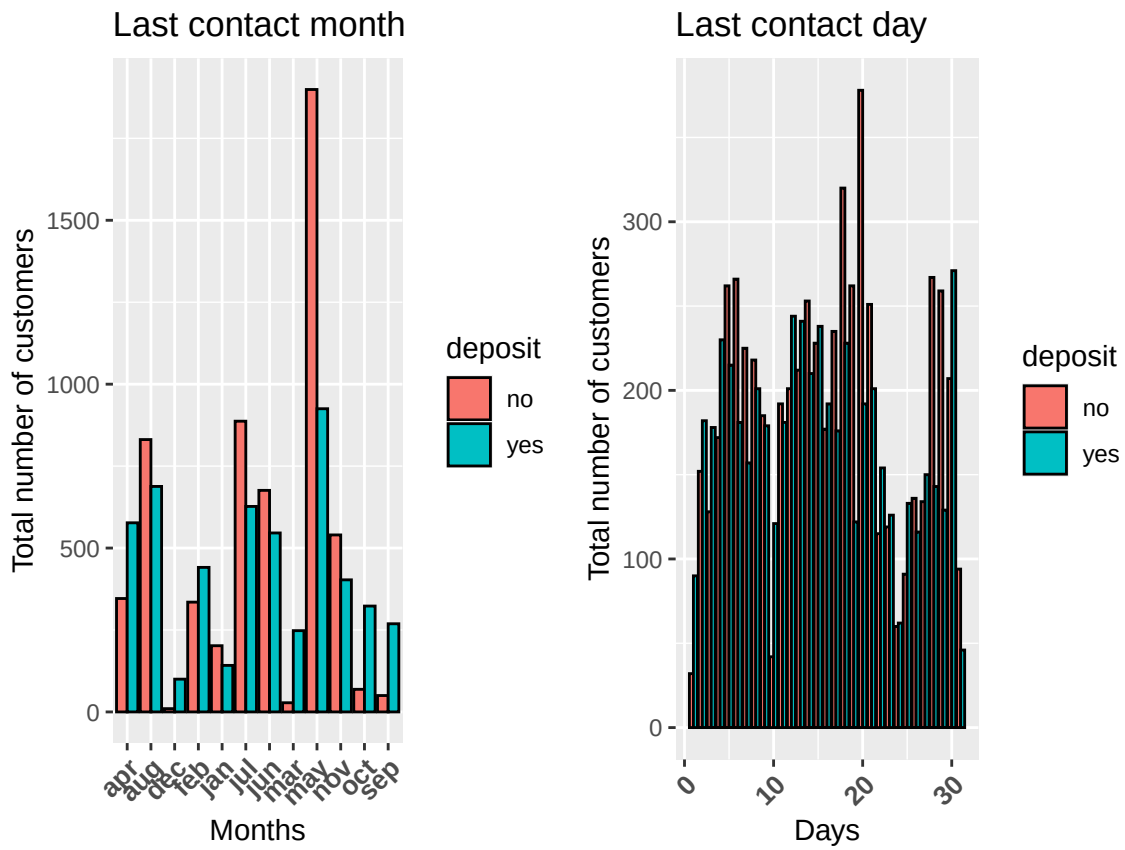


3.6 Contacts for current campaign

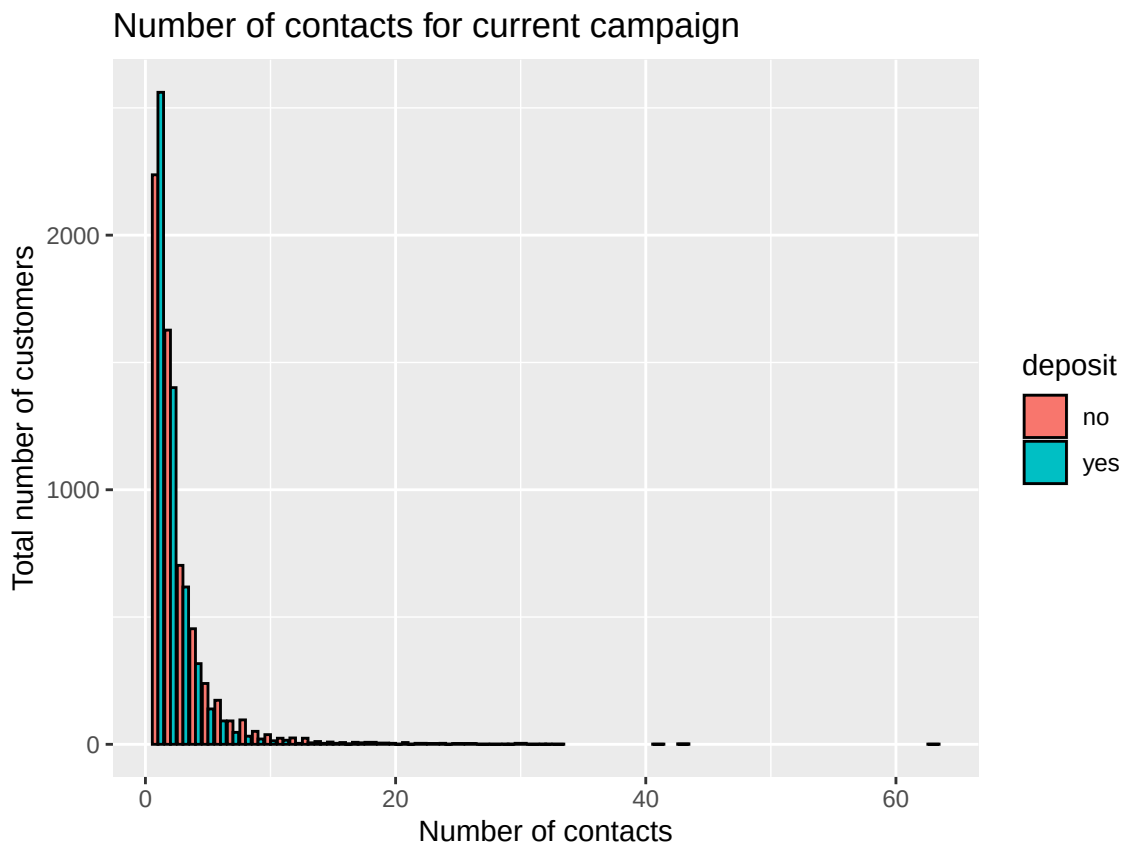
May seems to be the month with maximum contacts to customers, though it resulted in lesser percent of deposits compared to October, September, April, December and February.

```
mon<-bank_data%>%
  ggplot(aes(x=month,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle("Last contact month")+
  xlab("Months")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
day<-bank_data%>%
  ggplot(aes(x=day,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle("Last contact day")+
  xlab("Days")+
  ylab("Total number of customers")+
  theme(axis.text.x = element_text(face = "bold",
```

```
                                    size = 10, angle = 45, hjust = 1, vjust = 1))
grid.arrange(mon, day, nrow=1)
```




```
bank_data%>%
  ggplot(aes(x=campaign,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  xlab("Number of contacts")+
  ylab("Total number of customers")+
  ggtitle("Number of contacts for current campaign")
```

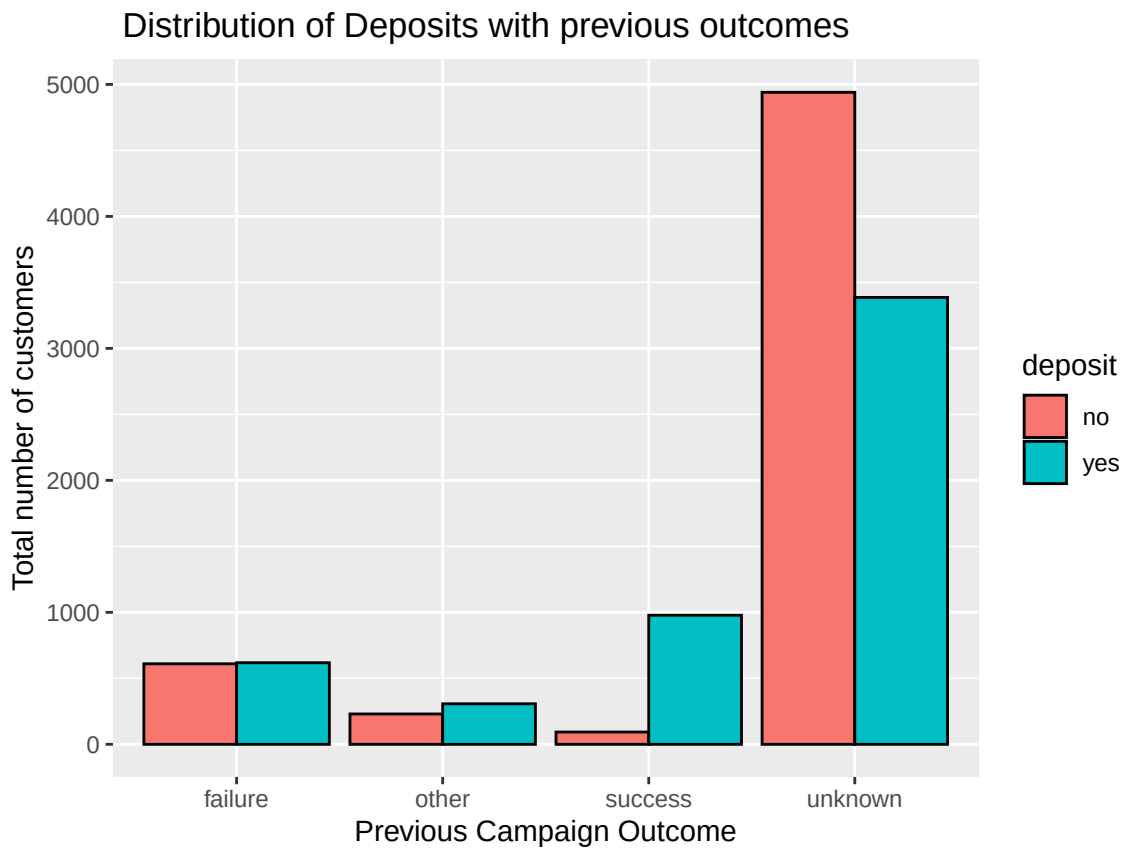


As the number of contacts increased, the chances of getting deposits has decreased. Therefore, campaigns should focus on lesser number of contacts but be effective.

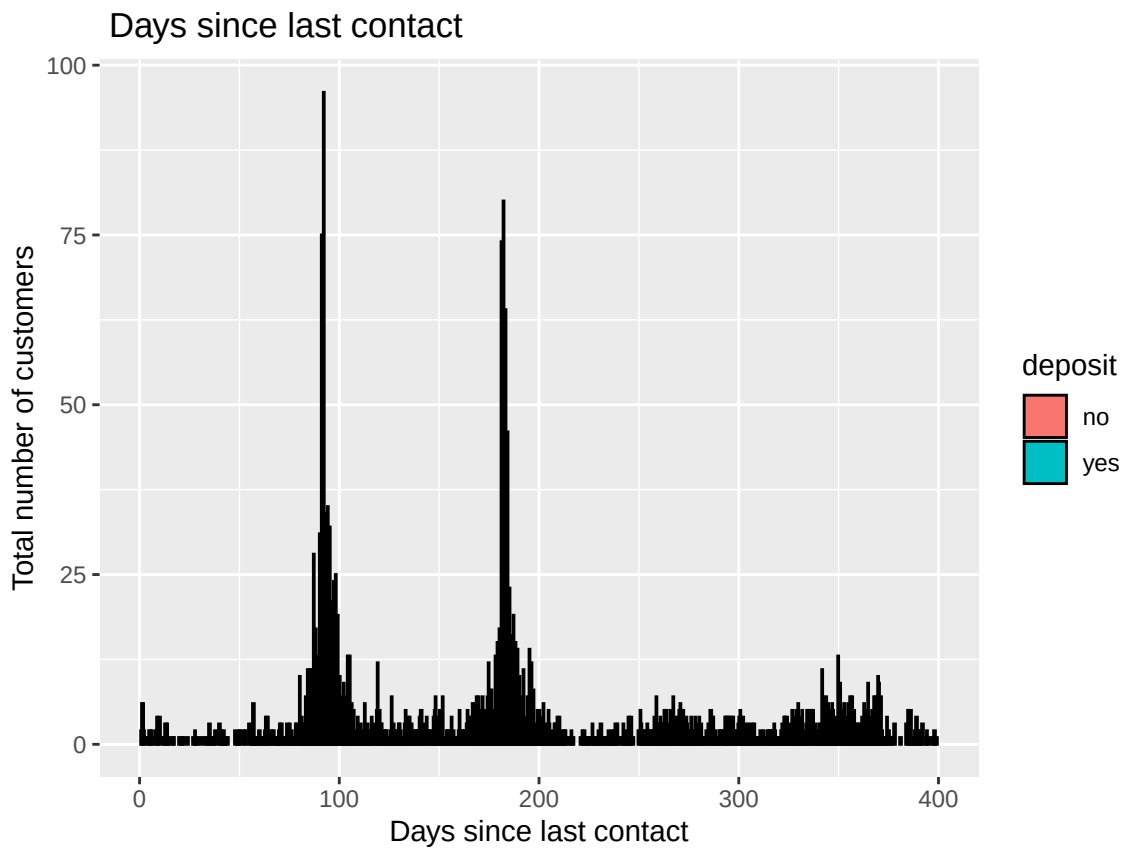
3.7 Contacts for previous campaign

The chart with previous outcomes shows that most of the previous outcomes are unknown. However, about 1000 deposits that were successful in the last campaign have deposited in the current campaign too. About 500 customers chose not deposit either times.

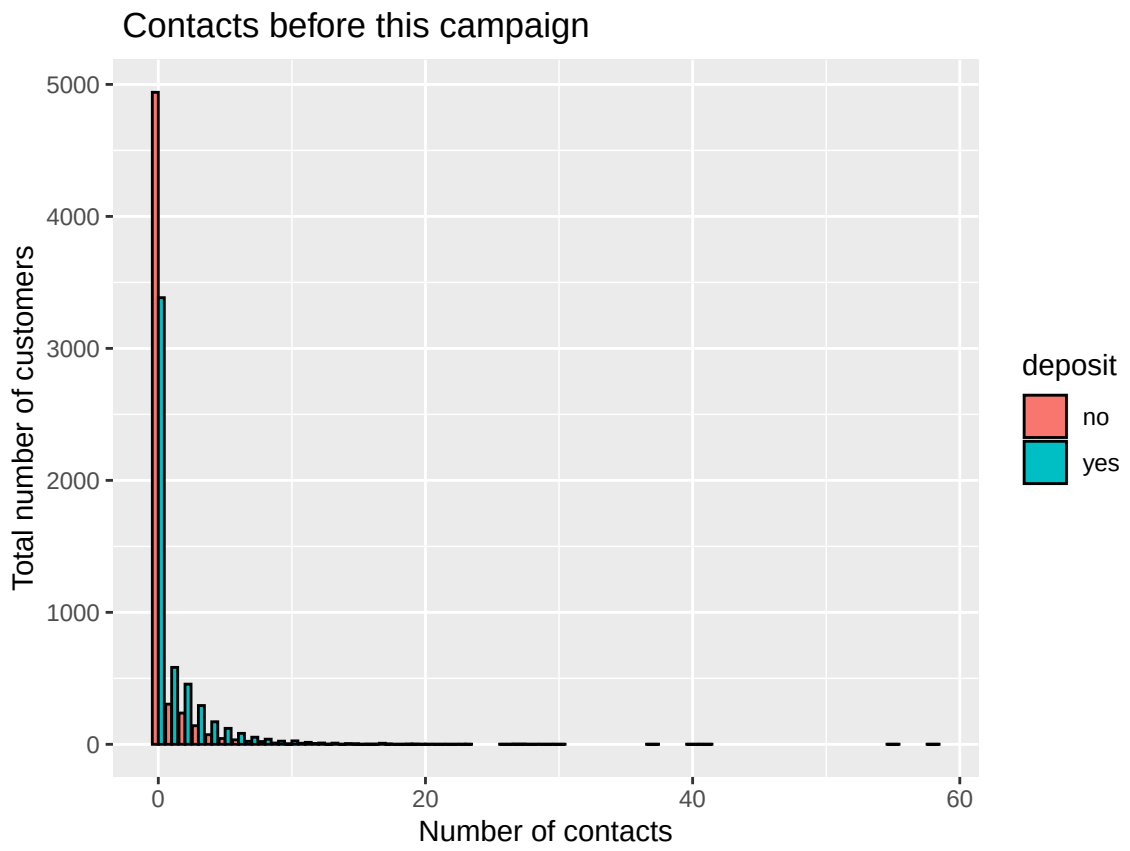
```
bank_data%>%
  ggplot(aes(x=poutcome,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  xlab("Previous Campaign Outcome")+
  ylab("Total number of customers")+
  ggtitle(" Distribution of Deposits with previous outcomes")
```




```
bank_data%>%
  ggplot(aes(x=pdays,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  ggtitle(" Days since last contact")+
  xlab("Days since last contact")+
  ylab("Total number of customers")+
  xlim(c(0,400))
```




```
bank_data%>%
  ggplot(aes(x=previous,  group=deposit, fill=deposit))+
  geom_histogram(stat="count", bins=20, col="black", position="dodge")+
  xlab("Number of contacts")+
  ylab("Total number of customers")+
  ggtitle(" Contacts before this campaign")
```



The number of days since last contact shows local maximums indicating that contacts were made every three months. Though the number of deposits made was higher during first contact, higher percent of deposits were made in subsequent contacts.

3.8 Bank Balance

Bank balance is an important factor to be able to do a deposit. It varied from -6,847 to 81,204 with median balance of 550. The below chart shows that the most balances were from 100 to 1,100 and the median balance of customers who deposited was slightly higher than the customers who did not deposit.

```
median(bank_data$balance)
```

```
## [1] 550
```

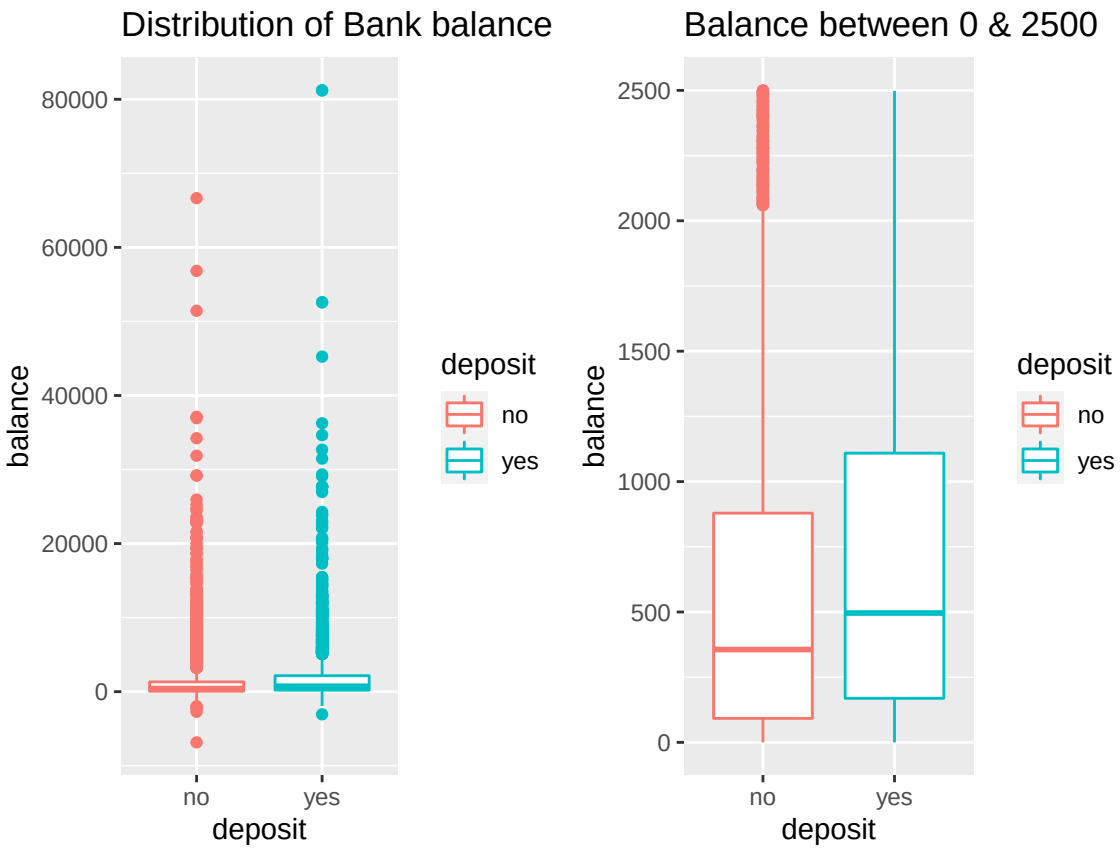
```
min(bank_data$balance)
```

```
## [1] -6847
```

```
max(bank_data$balance)
```

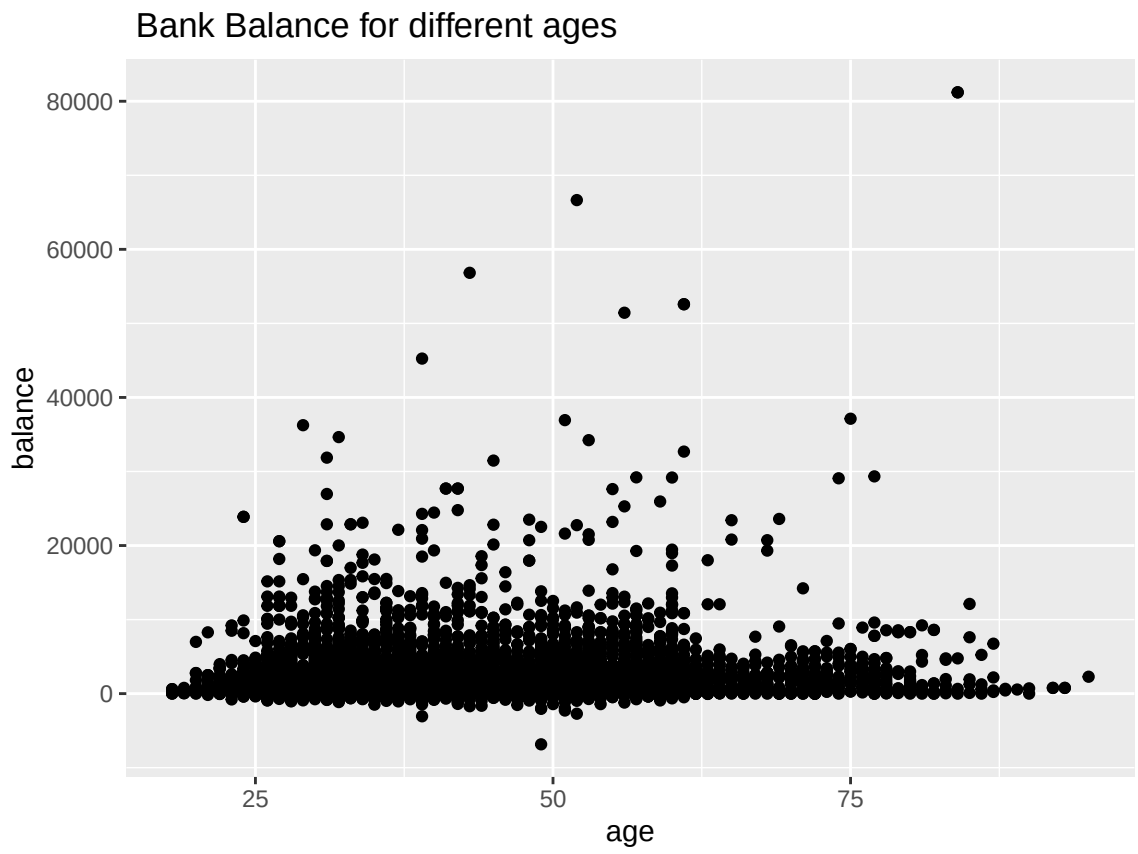
```
## [1] 81204
```

```
bal1<-bank_data%>%
  ggplot(aes(x=deposit, y=balance, col=deposit))+
  geom_boxplot()+
  ggtitle("Distribution of Bank balance")

bal2<-bank_data%>%
  ggplot(aes(x=deposit, y=balance, col=deposit))+
  geom_boxplot()+
  ggtitle("Balance between 0 & 2500")+
  ylim(0, 2500)

grid.arrange(bal1, bal2, nrow=1)
```

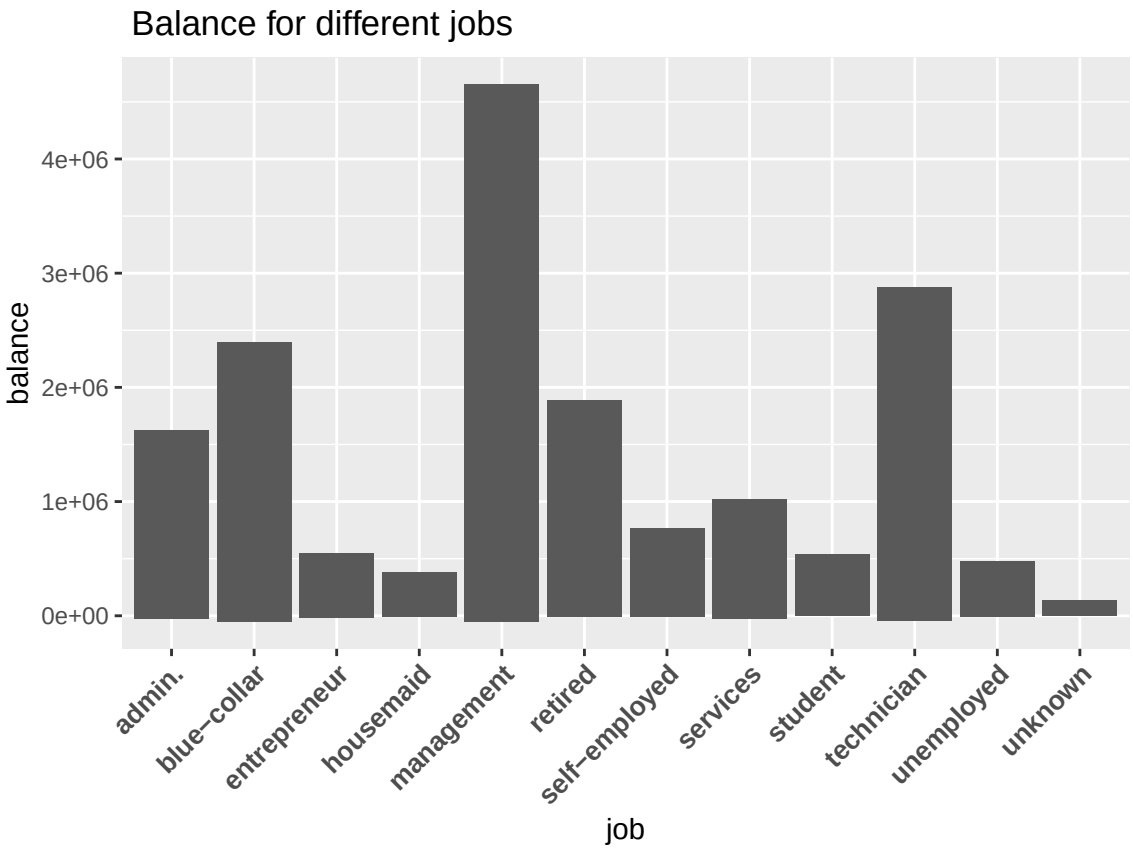


The below charts show how balances vary with different other factors.

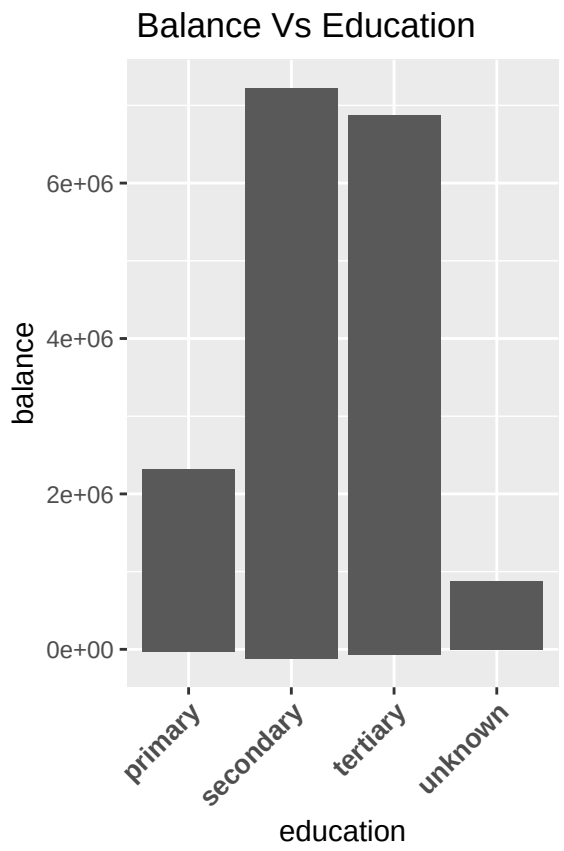
```
bank_data%>%
  ggplot(aes(x=age, y=balance))+
  geom_point()+
  ggtitle(" Bank Balance for different ages")
```

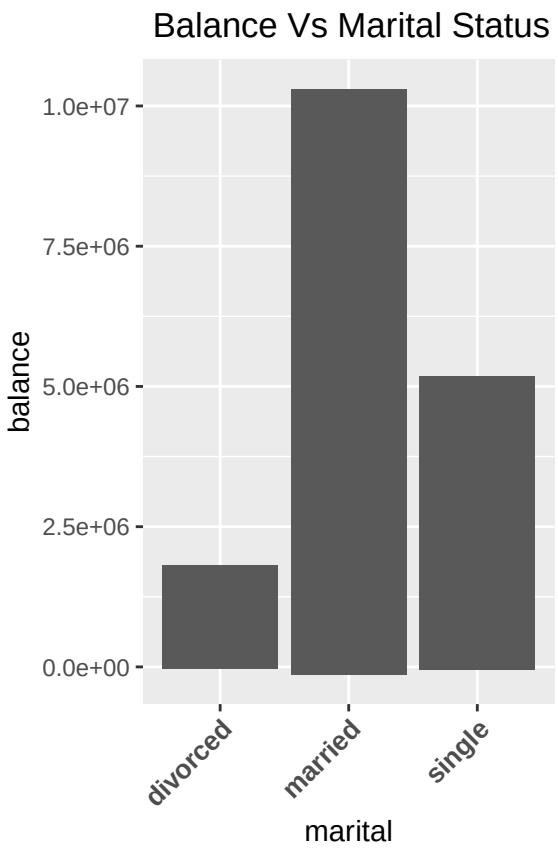



```
bank_data%>%
  ggplot(aes(x=job,  y=balance))+
  geom_bar(stat="identity")+
  ggtitle(" Balance for different jobs")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
```

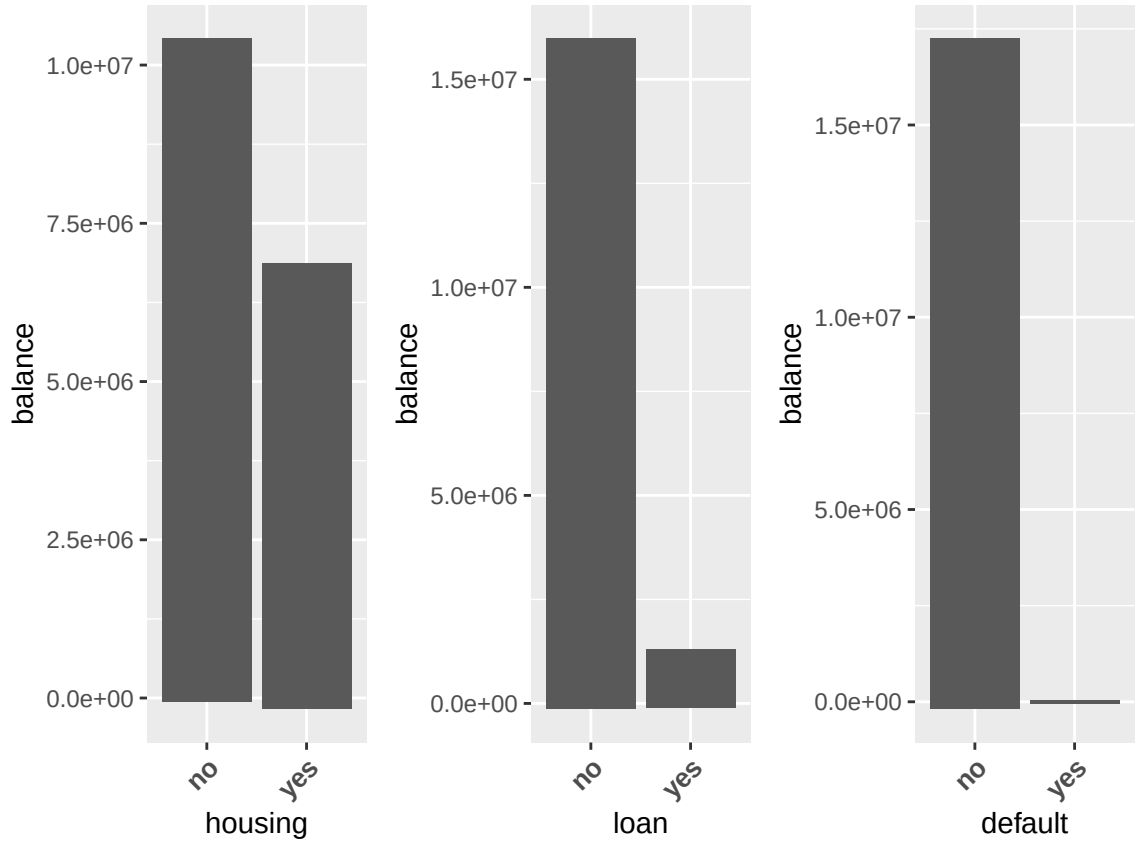



```
b1<-bank_data%>%
  ggplot(aes(x=education,  y=balance))+
  geom_bar(stat="identity")+
  ggtitle(" Balance Vs Education")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

b2<-bank_data%>%
  ggplot(aes(x=marital,  y=balance))+
  geom_bar(stat="identity")+
  ggtitle(" Balance Vs Marital Status")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
grid.arrange(b1, b2, nrow=1)
```






```
h_b<-bank_data%>%
  ggplot(aes(x=housing,  y=balance))+
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

l_b<-bank_data%>%
  ggplot(aes(x=loan,  y=balance))+
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))
d_b<-bank_data%>%
  ggplot(aes(x=default,  y=balance))+
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(face = "bold",
                                   size = 10, angle = 45, hjust = 1, vjust = 1))

grid.arrange(h_b, l_b, d_b, nrow=1)
```



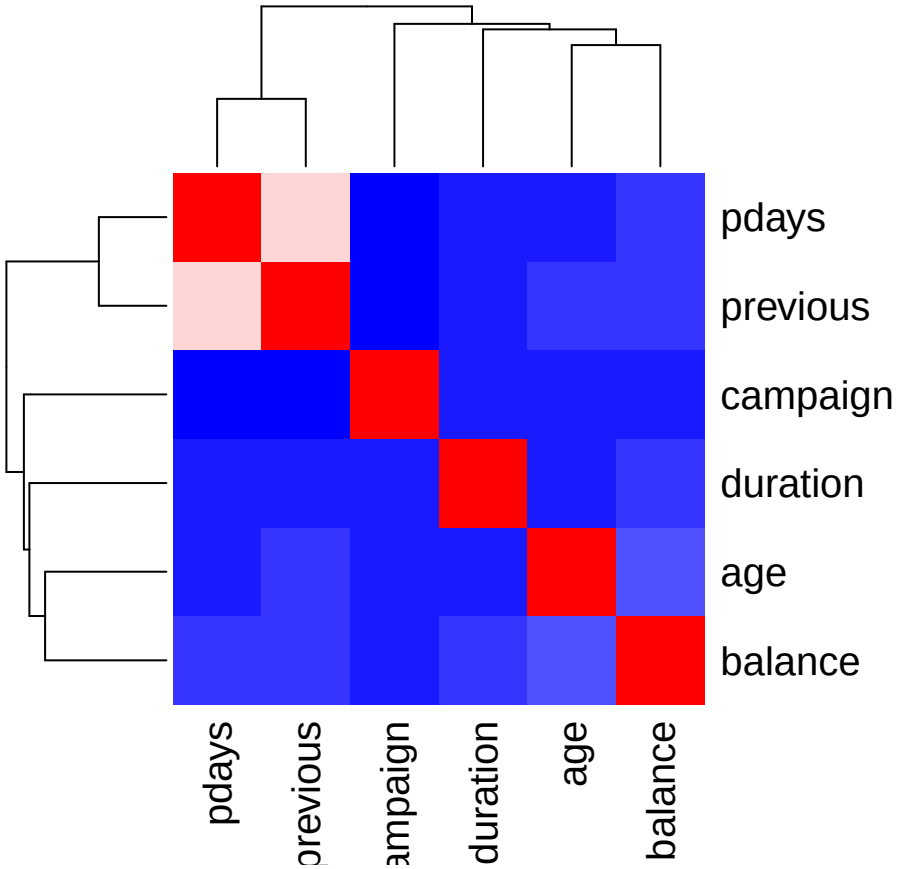
3.9 Correlation

The correlation analysis was performed with the duration column as it is heavily correlated with successful deposit. As per suggestion from the Kaggle and UCI Ml websites, the duration column must be excluded from running predictions because the duration of the call will be longer after it is decided that the customer wants to make a deposit.

```
cor(bank_data[,c("previous", "campaign","pdays", "age", "balance")], bank_data$duration)
```

```
##                   [,1]
## previous -0.0267161713
## campaign -0.0415574588
## pdays    -0.0273915532
## age       0.0001892281
## balance   0.0224361313
```

```
cortable<-cor(bank_data[,c("age", "balance", "duration", "campaign", "pdays", "previous")])
col<- colorRampPalette(c("blue", "white", "red"))(20)
heatmap(cortable, col=col, symm=TRUE)
```




```
cor(bank_data[,c("previous", "campaign","pdays", "age", "balance")], bank_data$duration)
```

```
##                   [,1]
## previous -0.0267161713
## campaign -0.0415574588
## pdays    -0.0273915532
## age       0.0001892281
## balance   0.0224361313
```

The corelation heatmap also shows that most variables are correlated with each other. Though some correlations were not very significant.

4 Data Analysis & Methods

The dataset has been divided into train, test and evaluation sets to predict if a customer will do a term deposit.

4.1 Data Splitting

The train set was used for training the model and test set to fine tune and compare different models. Once a final model has been chosen, the evaluation set can be used to validate the model. The data has been split with a 80%-20% ratio.

The *pdays* and *duration* column have been removed to run predictive analysis. The *duration* column has been removed because it heavily correlates with depositing an account and will be higher after the decision to deposit has been made. The *pdays* column has been removed as more than half of the customers have not been contacted before.

```
bank_data_ml<-bank_data[, -c(12, 14)]

set.seed(123, sample.kind="Rounding")

test_index <- createDataPartition(bank_data_ml$deposit, times = 1, p = 0.2, list = FALSE)
temp <- bank_data_ml[-test_index,]
evalset <- bank_data_ml[test_index,]

set.seed(123, sample.kind="Rounding")

test_index <- createDataPartition(temp$deposit, times = 1, p = 0.2, list = FALSE)
trainset <- temp [-test_index,]
testset <- temp [test_index,]
```

4.2 Output Measuring method

This dataset uses Supervised Classification algorithms to run different models. It is important to define the correct output measurements for comparison. The classification models usually use accuracy to measure the quality of a model. However, other measures like precision, recall, F score and Area under the ROC curve can be important to describe how well the model predicts the correct values.

The table giving the correct and wrong values is called the Confusion Matrix as shown below. True Positives and True Negatives are the correct predictions. False Negatives are the actual positives that got wrongly got predicted negative whereas False positives are the actual negatives that got wrongly predicted positive. The other measures can be defined as shown below.

	Actual 0	Actual 1
Prediction 0	True Negative (TN)	False Negative (FN)
Prediction 1	False Positive (FP)	True Positive (TP)

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}$$

$$False\ Positive\ Rate(FPR) = \frac{FP}{FP + TN}$$

$$Precision\ or\ True\ Positive\ Rate(TPR) = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$FScore = 2.\frac{Precison.Recall}{Precision + Recall}$$

The area under the curve AUC is the area under the ROC(Receiver Operating Characteristic) curve which is a probability curve. The AUC tells how much the model is capable of distinguishing between the classes. The higher the AUC better the model is at predicting the classes. This curve is plotted with TPR(True Positive Rate) on the y-axis against the FPR(False Positive Rate) on the x-axis. The different points on the curve are obtained by using different cutoff values to determine the class labels.

There are many functions already available in R that give these metrics however the below user-defined function was used in this project. Different models are compared to find the highest Area under the curve. The chosen model is then fine tuned with cut-off analysis to apply on the evaluation set.

```
get_result_stats<-function(x,y){
  cm<-table(Predict=x, Reference=y)
  acc<-(cm[1,1]+cm[2,2])/sum(cm)
  precision<-cm[2,2]/(cm[1,2]+cm[2,2])
  recall<-cm[2,2]/(cm[2,1]+cm[2,2])
  f1score<-2*precision*recall/(precision+recall)

  list(cm=cm, acc=acc, precision=precision, recall=recall, f1score=f1score)
}
```

4.3 Machine Learning Models

Classification algorithms have been modeled on the train data and tested using the test data. Cross-validation has been used with k-fold=5 to make the models more robust.

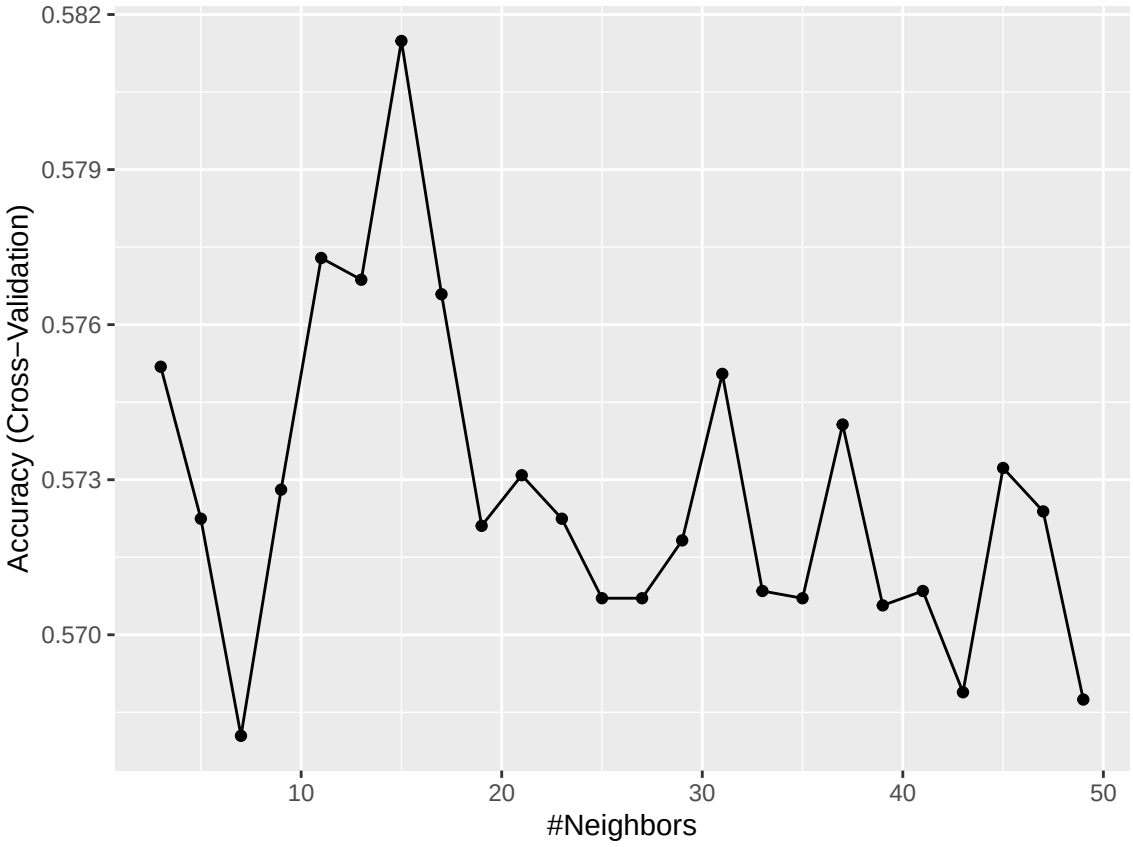
4.3.1 K-Nearest Neighbors (KNN)

The KNN model is a classification model that assigns to a class that is most common among its k nearest neighbors. The value of k can be tuned for best performance. Most models in this project have used the train function in the caret package to model.

```
ctrl <- trainControl(method = "cv",
                     number = 5,
                     allowParallel = TRUE)
set.seed(123, sample.kind="Rounding")
model_knn<-  train(deposit~., data=trainset, method = "knn",
                   trControl = ctrl, tuneGrid = data.frame(k = seq(3, 50, 2)))
model_knn$bestTune
```

	k
7	15

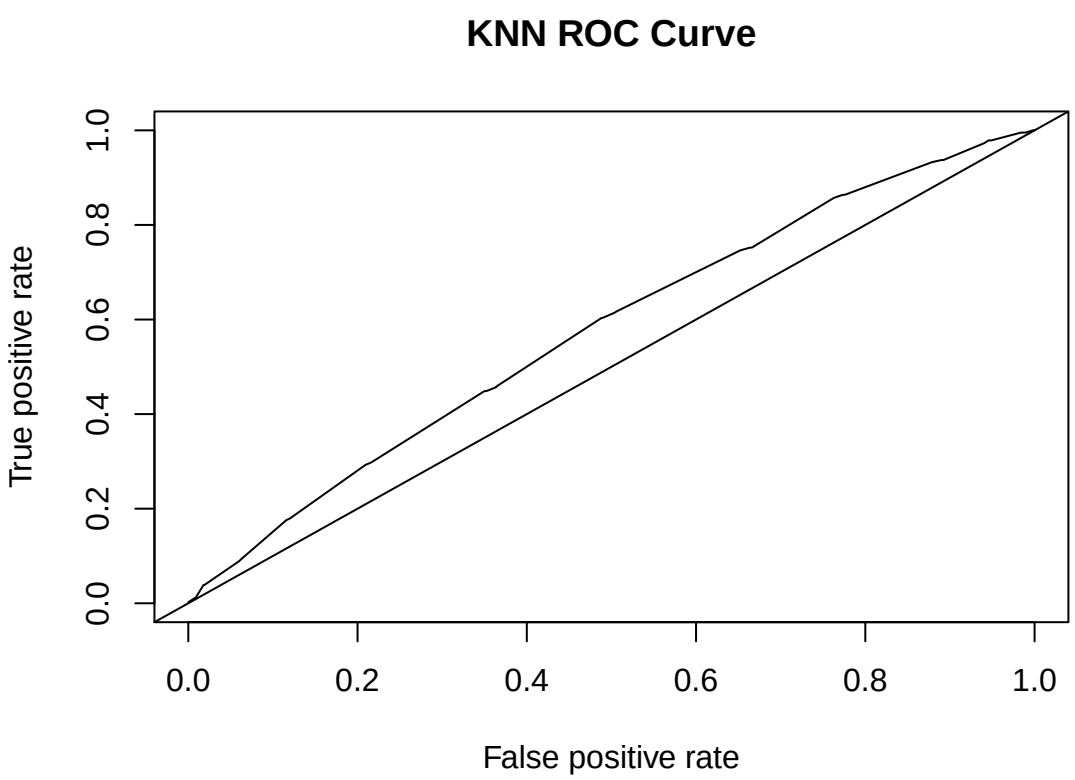
```
ggplot(model_knn)
```




```
knn_dep<-predict(model_knn, testset)
knn_summary<-get_result_stats(knn_dep, testset$deposit)

knn_y<- predict(model_knn, testset, type="prob")[,2]
pred.knn <- prediction(knn_y, testset$deposit)
knn_auc = performance(pred.knn, measure = "auc")

all_results<-data.frame(method="Baseline: KNN", f1score=knn_summary$f1score,
                                                accuracy=knn_summary$acc,
                                                precision=knn_summary$precision,
                                                recall=knn_summary$recall,
                                                AUC=round(knn_auc@y.values[[1]],6))
all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.531856	0.576293

```
roc.perf.knn = performance(pred.knn, measure = "tpr", x.measure = "fpr")
plot(roc.perf.knn, main="KNN ROC Curve")
abline(a=0, b= 1)
```



As can be seen above, the optimum k value can be obtained from the chart. The above plot shows the area under the curve. Ideally, it is desired to see the curve in the upper left corner. This model gave an accuracy of 0.552, AUC of 0.576 and Fscore of 0.489. This model has been used as baseline model to compare improvement and performance of other models.

4.3.2 Naive- Bayes (NB)

Naive-Bayes algorithm is a probabilistic machine learning algorithm used for classification. It assumes that the predictors or features are independent (hence called naive). The algorithm is based on the Bayes theorem which states that the probability of event A happening given event B is happening can be calculated.

Bayes Theorem:

$$P(A/B) = \frac{P(B/A).P(A)}{P(B)}$$

The Naive Bayes algorithm is fast and easy to implement but the biggest disadvantage is that in real life cases the predictors can be dependent which impacts the classification.

```
set.seed(123, sample.kind="Rounding")
model_nb<-  train(deposit~., data=trainset, method = "naive_bayes",
                  trControl = ctrl)

nb_dep<-predict(model_nb, testset)

nb_summary<-get_result_stats(nb_dep, testset$deposit)
```

```
nb_y<- predict(model_nb, testset, type="prob")[,2]
pred <- prediction(nb_y, testset$deposit)
nb_auc = performance(pred, measure = "auc")

all_results<-rbind(all_results, data.frame(method="Naive Bayes",
                                           f1score=nb_summary$f1score,
                                           accuracy=nb_summary$acc,
                                           precision=nb_summary$precision,
                                           recall=nb_summary$recall,
                                           AUC=round(nb_auc@y.values[[1]],6)))
all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027

The accuracy, Fscore and area under the curve(AUC) have all shown improvement.

4.3.3 Logistic Regression (GLM)

Logistic Regression is a classification algorithm used to assign observations to a discrete set of classes. The algorithm is based on the concept of probability and can be called linear regression model which uses a more complex function like the sigmoid or logistic function that takes an S shaped curve to classify.

```
set.seed(123, sample.kind="Rounding")
model_glm<-  train(deposit~., data=trainset, method = "glm",
                   trControl = ctrl)

model_glm$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
##        (Intercept)                 age    `jobblue-collar`     jobentrepreneur
##          6.092e-01           3.982e-03           1.093e-02          -1.178e-01
##       jobhousemaid       jobmanagement          jobretired  `jobself-employed`
##         -3.247e-01          -1.115e-01           3.276e-01          -6.314e-02
##        jobservices          jobstudent       jobtechnician       jobunemployed
##         -9.130e-02           7.419e-01          -1.304e-02           2.533e-01
##         jobunknown      maritalmarried       maritalsingle  educationsecondary
##         -6.404e-01          -2.113e-01           9.823e-02           6.578e-02
##  educationtertiary    educationunknown          defaultyes             balance
##          3.067e-01           7.772e-02          -4.190e-01           2.593e-05
##         housingyes             loanyes    contacttelephone      contactunknown
##         -3.938e-01          -4.212e-01          -2.313e-01          -1.156e+00
##                day            monthaug            monthdec            monthfeb
##         -2.856e-03          -7.542e-01           1.502e+00          -4.156e-01
##           monthjan            monthjul            monthjun            monthmar
##         -1.232e+00          -6.192e-01           8.090e-02           1.534e+00
```

```
##           monthmay           monthnov           monthoct           monthsep
##         -5.532e-01         -9.131e-01          6.597e-01          6.646e-01
##           campaign           previous       poutcomeother    poutcomesuccess
##         -8.414e-02          5.468e-04          1.472e-01          2.310e+00
##    poutcomeunknown
##         -6.957e-03
##
## Degrees of Freedom: 7141 Total (i.e. Null);  7101 Residual
## Null Deviance:       9881
## Residual Deviance: 8093  AIC: 8175
```

The coefficients for the logistic regression analysis can be seen above. From the coefficients, those that contribute the most can be seen to have higher coefficients like the *jobretired*, *jobstudent*, *poutcomesuccess*, and few months.

```
glm_dep<-predict(model_glm, testset)
glm_summary<-get_result_stats(glm_dep, testset$deposit)

glm_y<- predict(model_glm, testset, type="prob")[,2]
pred <- prediction(glm_y, testset$deposit)
glm_auc = performance(pred, measure = "auc")

all_results<-rbind(all_results, data.frame(method="Logistic Regression",
                                           f1score=glm_summary$f1score,
                                           accuracy=glm_summary$acc,
                                           precision=glm_summary$precision,
                                           recall=glm_summary$recall,
                                           AUC=round(glm_auc@y.values[[1]],6)))

all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319

The Logistic regression algorithm though has dropped the recall rate, overall has further improved the classification as all other metrics have shown improvement.

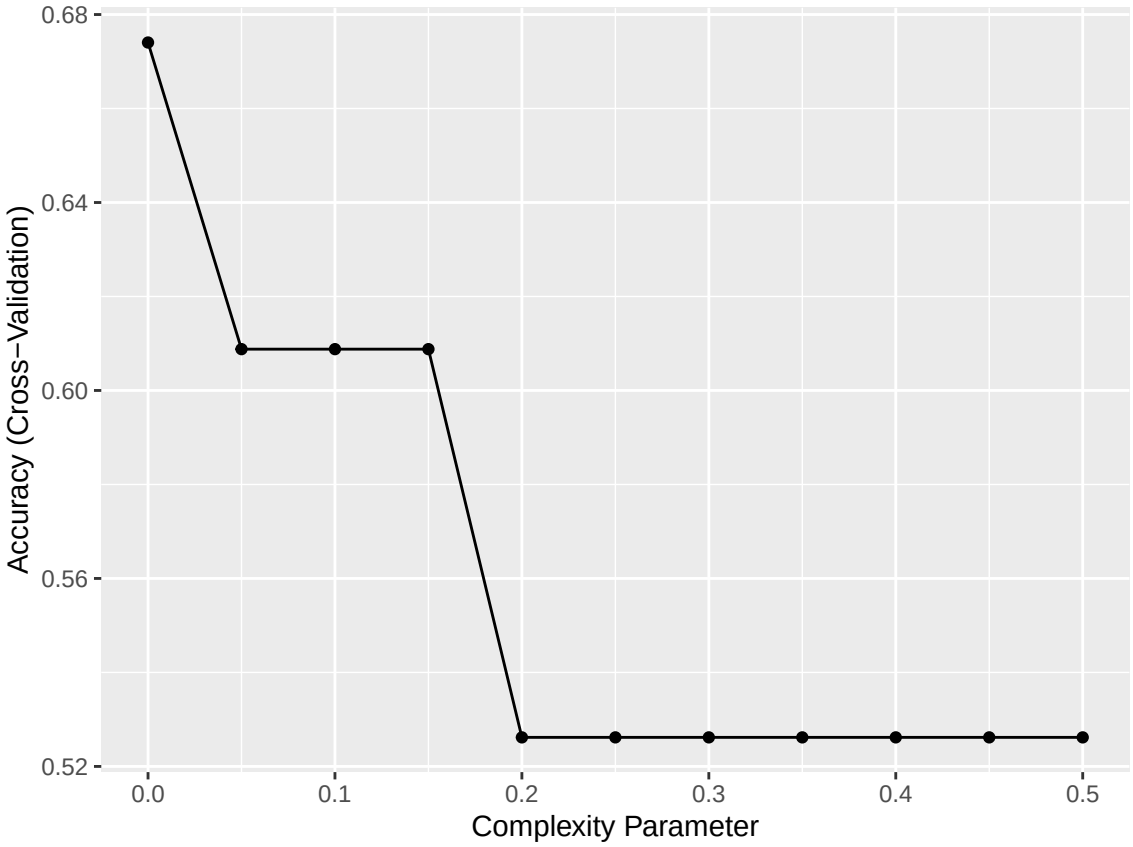
4.3.4 Classification and Regression Trees (CART)

CART is classification and regression trees for machine learning. It uses a decision tree as a predictive model to change observations to conclusions based on the binary outcome at each node in the tree.

```
set.seed(123, sample.kind="Rounding")
model_tree<-  train(deposit~., data=trainset, method="rpart",
                    tuneGrid= data.frame(cp=seq(0, 0.5, 0.05)),
                    trControl = ctrl)
model_tree$bestTune
```

cp
0

```
ggplot(model_tree)
```



Fine tuning of hyper-parameter called complexity paramater *cp* has shown that the cp=0 yielded best results.

```
tree_dep<-predict(model_tree, testset)
tree_summary<-get_result_stats(tree_dep, testset$deposit)

tree_y<- predict(model_tree, testset, type="prob")[,2]
pred <- prediction(tree_y, testset$deposit)
tree_auc = performance(pred, measure = "auc")

all_results<-rbind(all_results, data.frame(method="CART", f1score=tree_summary$f1score,
                                           accuracy=tree_summary$acc,
                                           precision=tree_summary$precision,
                                           recall=tree_summary$recall,
                                           AUC=round(tree_auc@y.values[[1]],6)))
all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319
CART	0.6501211	0.6765529	0.6340024	0.6670807	0.733119

Based on the results it can be seen that though the Fscore has improved, the drop in the recall rate has impacted the area under the curve. This also indicates that probably one tree is not enough and having multiple trees may even out the classification.

4.3.5 Random Forest (RF)

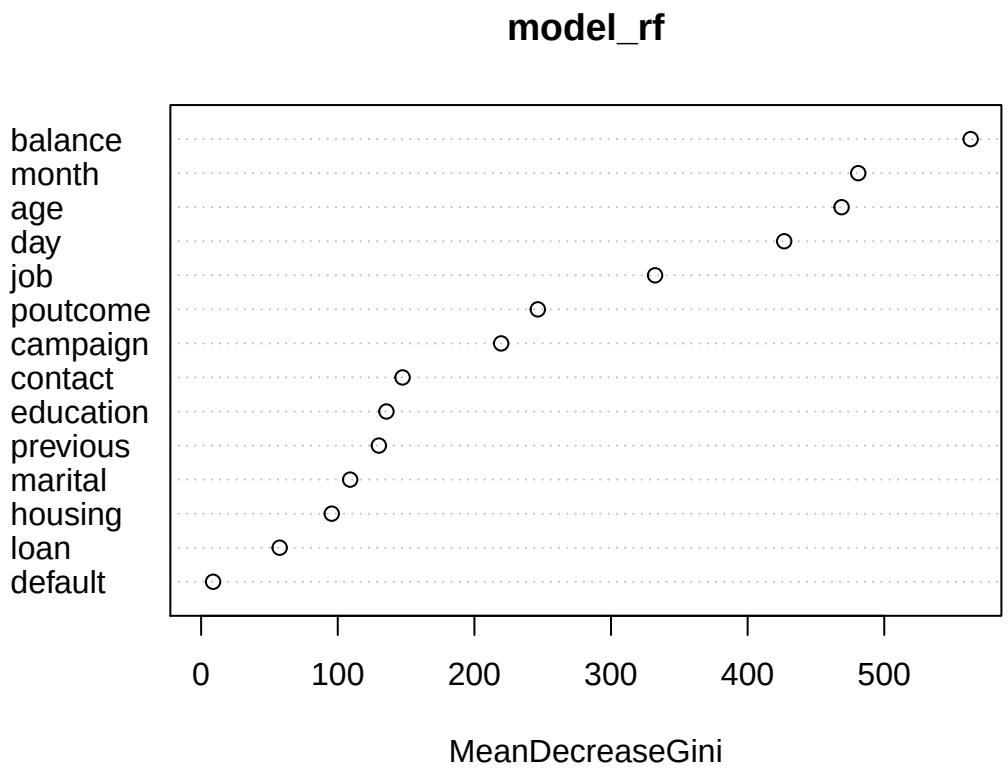
Random Forest is a flexible machine learning algorithm that even without hyper-parameter tuning yields a great result most times. It is an example of ensemble model, which is a collection of multiple models to predict best outcomes that individual models fail to do by themselves. The "forest" is an ensemble of multiple "trees" that are combined together and increase overall results. This method is also called "bagging" method.

```
set.seed(123, sample.kind="Rounding")
model_rf<-  randomForest(deposit~., data=trainset,  mtry=4,
                  trControl = ctrl)

rf_dep<-predict(model_rf, testset)

rf_summary<-get_result_stats(rf_dep, testset$deposit)

rf_y<- predict(model_rf, testset, type="prob")[,2]
pred <- prediction(rf_y, testset$deposit)
rf_auc = performance(pred, measure = "auc")

all_results<-rbind(all_results, data.frame(method="Random Forest",
                                           f1score=rf_summary$f1score,
                                           accuracy=rf_summary$acc,
                                           precision=rf_summary$precision,
                                           recall=rf_summary$recall,
                                           AUC=round(rf_auc@y.values[[1]],6)))
all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319
CART	0.6501211	0.6765529	0.6340024	0.6670807	0.733119
Random Forest	0.7015113	0.7347510	0.6576151	0.7516869	0.787963

The results summary shows improvement in all measures and that the classification has evened out the loss seen in the CART algorithm.

```
varImpPlot(model_rf)
```



The variable importance plot shows the order in which the different predictors have influenced the classification. The balance in a person's account and what time the contact was made were some important predictors other than age and job among others.

4.3.6 Gradient Boosting (GBM)

Gradient Boosting algorithm is method of converting weak predictions to strong. The algorithm begins by training a decision tree and use the residual loss function to assign weights to observations that are difficult to classify. The next tree improves predictions on these difficult observations and re-computes loss function for subsequent trees. Prediction of the final ensemble model is therefore the weighted sum of predictions made by the previous models. The caret package's train function is used for modeling with hyper-parameters *interaction.depth* and *n.trees*. Interaction depth specifies the maximum depth of each tree i.e. that highest level of variable interactions allowed while training the model. *n.trees* is the number of trees used for classification.

gbmGrid <- expand.grid(interaction.depth = c(1, 5, 9), n.trees = (1:30)*50, shrinkage = 0.1, n.minobsinnode = 20) set.seed(123, sample.kind="Rounding") model_gbm<- train(deposit~., data=trainset, method="gbm", trControl = ctrl, tuneGrid = gbmGrid)

```
model_gbm$bestTune
```

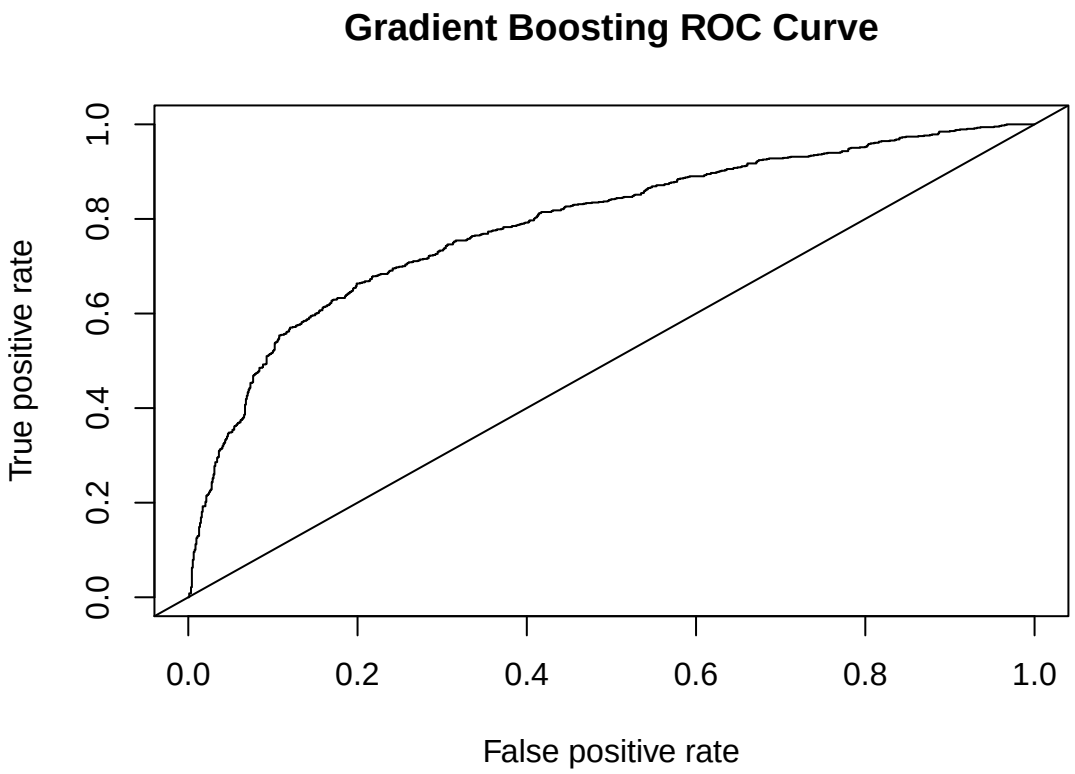
	n.trees	interaction.depth	shrinkage	n.minobsinnode
63	150	9	0.1	20

```
gbm_dep<-predict(model_gbm, testset)

gbm_summary<-get_result_stats(gbm_dep, testset$deposit)

gbm_y<- predict(model_gbm, testset, type="prob")[,2]
pred.gbm <- prediction(gbm_y, testset$deposit)
gbm_auc = performance(pred.gbm, measure = "auc")

all_results<-rbind(all_results, data.frame(method="Gradient Boost",
                                           f1score=gbm_summary$f1score,
                                           accuracy=gbm_summary$acc,
                                           precision=gbm_summary$precision,
                                           recall=gbm_summary$recall,
                                           AUC=round(gbm_auc@y.values[[1]],6)))

all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319
CART	0.6501211	0.6765529	0.6340024	0.6670807	0.733119
Random Forest	0.7015113	0.7347510	0.6576151	0.7516869	0.787963
Gradient Boost	0.6842801	0.7325126	0.6115702	0.7766117	0.789384

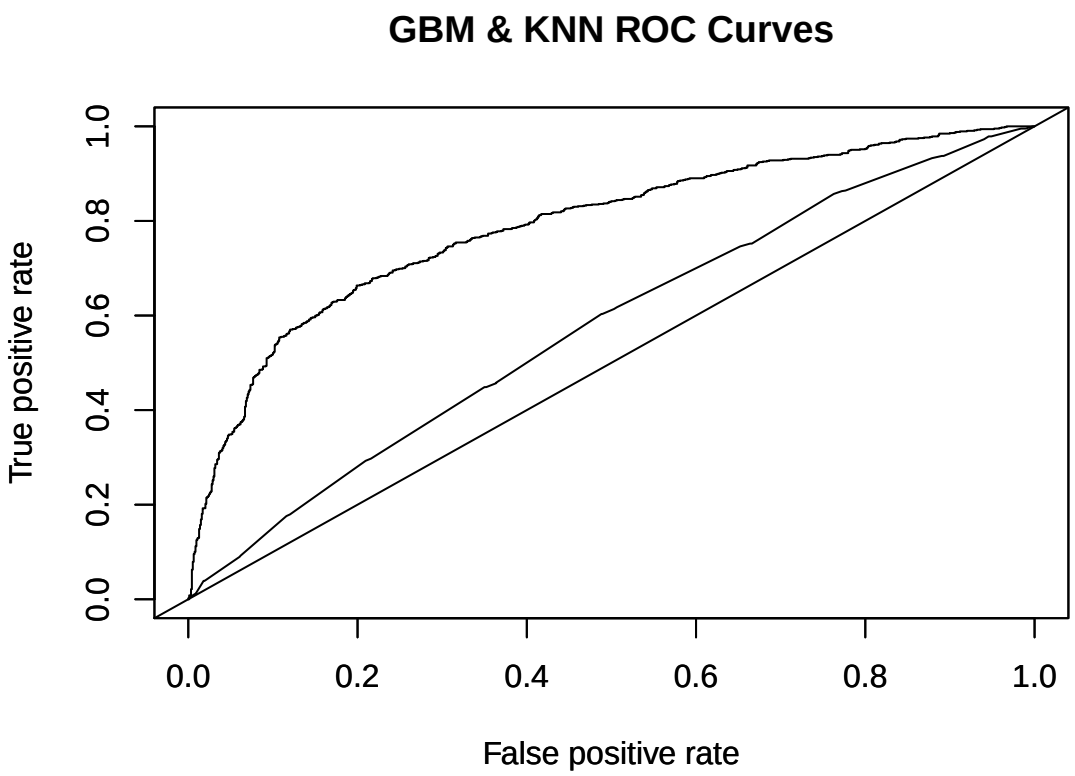
The gradient boosting algorithm which is also an ensemble model shows slight improvement in the area under the curve over the Random Forest model.

```
# plot ROC curve for different cutoff values
roc.perf.gbm = performance(pred.gbm, measure = "tpr", x.measure = "fpr")
plot(roc.perf.gbm, main="Gradient Boosting ROC Curve")
abline(a=0, b= 1)
```



The ROC curve has moved into the upper left quadrant. The below graph overlays the KNN and GBM model ROC curves to clearly show the improvement.

```
# plot of KNN ROC curve to compare performance
plot(roc.perf.gbm, main="GBM & KNN ROC Curves")
par(new=TRUE)
plot(roc.perf.knn)
abline(a=0, b= 1)
```



4.3.6 Model Results

The final summary of all models shows that overall the Gradient Boost model shows the best performance with respect to area under the curve. Hence, this model has been chosen as the final model.

```
all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319
CART	0.6501211	0.6765529	0.6340024	0.6670807	0.733119
Random Forest	0.7015113	0.7347510	0.6576151	0.7516869	0.787963
Gradient Boost	0.6842801	0.7325126	0.6115702	0.7766117	0.789384

To fine tune the model cut-off analysis has been performed. Cut-off Analysis basically analyzes the different cut-off levels that determine the class labels. In this case, there are two labels, hence default cutoff is 0.5. By sweeping different values of cut-off and measuring maximum Fscore the optimum cut-off value can be obtained. The user-defined function can be written as shown below.

```
get_cutoff<-function(x){

  cutoff<-seq(0.1, 0.9, 0.025)
```

```
  f1s<-sapply(cutoff, function(z){
    y<-ifelse(x>z,1,0)
    scores<-get_result_stats(y, testset$deposit)
    scores$f1score
  })
  return(cutoff[which.max(f1s)])
}
```

```
final_cutoff<-get_cutoff(gbm_y)

final_y<-ifelse(gbm_y>final_cutoff,1,0)

final_stats<-get_result_stats(final_y, testset$deposit)

all_results<-rbind(all_results,
                   data.frame(method="Gradient Boost(Cut-off analysis)",
                              f1score=final_stats$f1score,
                              accuracy=final_stats$acc,
                              precision=final_stats$precision,
                              recall=final_stats$recall,
                              AUC=round(gbm_auc@y.values[[1]],6)))

all_results%>%knitr::kable()
```

method	f1score	accuracy	precision	recall	AUC
Baseline: KNN	0.4894837	0.5517627	0.4533648	0.5318560	0.576293
Naive Bayes	0.5654135	0.6765529	0.4439197	0.7784679	0.740027
Logistic Regression	0.6483957	0.7056519	0.5726092	0.7473035	0.765319
CART	0.6501211	0.6765529	0.6340024	0.6670807	0.733119
Random Forest	0.7015113	0.7347510	0.6576151	0.7516869	0.787963
Gradient Boost	0.6842801	0.7325126	0.6115702	0.7766117	0.789384
Gradient Boost(Cut-off analysis)	0.7119777	0.7106883	0.7544274	0.6740506	0.789384

```
final_cutoff
```

```
## [1] 0.375
```

The above table shows that the FScore has improved to 0.712. A cut-off value of 0.375 has given best Fscore. The GBM model with cutoff 0.375 is the final model that can be applied to new data to determine if a term deposit will be made.

4.3.7 Evaluation Data Results

The GBM model with cut-off 0.375 has been applied to the evaluation dataset and shown an accuracy of 0.7138 with FScore of 0.7136. These values are very close to the training/testing model and hence prove that the model is not over-trained or under-trained.

```
eval_y<- predict(model_gbm, evalset, type="prob")[,2]
```

```r
# Use cutoff obtained from GBM to define class labels
evalset_deposit<-ifelse(eval_y>final_cutoff,1,0)

# Get final summary for evaluation set
final_summary<-get_result_stats(evalset_deposit, evalset$deposit)

# Final summary of the evaluation of the model put in easy to read format
data.frame(F1Score=final_summary$f1score,
           Accuracy=final_summary$acc,
           Precision=final_summary$precision,
           Recall=final_summary$recall)
```

F1Score	Accuracy	Precision	Recall
0.7135814	0.7138379	0.7523629	0.6786019

5 Conclusion

This section outlines the final summary of the project and future work.

5.1 Summary

Data analysis of the Portugese banking institution's dataset obtained from Kaggle website has shown that the marketing campaign can leverage the key insights generated to optimally target the customer base and effectively increase the term deposits.

The Gradient Boosting algorithm has shown the best results with Fscore of 0.7136 for predicting if a person will do a term deposit.

5.2 Limitations

One limitation of this analysis is that even though the model uses ensemble techniques, the accuracy is still around 72%. This may be primarily due to using a subset of the full dataset. The dataset on Kaggle website seems to be a subset of the original data uploaded to UCI Machine Learning Repository. This potentially impacts the robustness of the analysis.

5.3 Future Work

In order to further understand the behavior of customer choices for success and failure, competitive benchmarking data can also be included along with demographics. Also, to address the limitation of the analysis mentioned above, extending the analysis to include the full-set can increase robustness, predictive power and give further insights.

To increase the predictive power of the model, machine learning concepts like the SVM, Neural Networks and Ensemble model techniques like Stacking and Blending can be evaluated.

6 References

https://rafalab.github.io/dsbook/

https://topepo.github.io/caret/available-models.html

https://towardsdatascience.com/

https://www.analyticsvidhya.com/

https://machinelearningmastery.com/