HarvardX: PH125.9x Data Science
Portugal Banking term loan prediction

Mano Krishnan

02/04/2020

Contents

1 Introduction

We chose Bank Marketing Data set for the final project on Data science. This is Portuguese Banking institutional data. It has many attributes including a client's subscription to term deposit or not. The aim is to build models that can predict whether client will subscribe to term deposit or not.

1.1 Dataset and Data Loading

This dataset is downloaded from UCI Machi Learning Repository. This is related to direct marketing campaigns of the Portuguese Banking institution. This dataset is available at http://archive.ics.uci.edu/ml/datasets/Bank+Marketing. There were 4 datasets in it from which bank-full.csv is used that has all examples (45211) and 17 inputs ordered by date. There are 16 input variables and 1 output variable (desired target).

This had different categories of client data like job, age, marital, education, default, housing, loan, contact, month, balance, day_of_week, duration, campaign, pdays, previous, poutcome and one output variable y that denotes if client subscribed to term deposit or not. These data denote telemarketing data, customer data and some other data. Here, many attributes are numerical and some are categorical. We loaded the dataset in the R studio and checked for any missing values using is.na fucntion and found no missing data. Hence, we have clean data.

1.1.1 Libraries

The following libraries were used in this report:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(rattle)) install.packages("rattle", repos = "http://cran.us.r-project.org")
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")
if(!require(descr)) install.packages("descr", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

1.1.2 Dataset loading

```
Portdata <- read_delim("bank-full.csv",
                       ";", escape_double = FALSE, trim_ws = TRUE)
## Duplicate row check
sum(duplicated(Portdata))
```

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

```
## Missing data check
sum(!complete.cases(Portdata))
```

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

```
all.empty = rowSums(is.na(Portdata))==ncol(Portdata)
sum(all.empty)
```

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

```
Portdata.clean = Portdata[!all.empty,]

Portdata.clean = Portdata.clean %>% distinct

nrow(Portdata.clean)
```

```
## [1] 45211
```

```
Portdata.clean$missing = !complete.cases(Portdata.clean)

sum(is.na(Portdata))
```

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

1.1.3 Attribute Information:

1. age – Client Age- (numeric)
2. job – Type of Job - (categorical) ('admin.','blue-collar','entrepreneur','housemaid','management', 'retired','selfemployed','services', 'student','technician','unemployed','unknown')
3. marital - Client's marital status - (categorical) (divorced, married, single, unknown, note: divorced means divorced or widowed)
4. education - Client's education - (categorical) (basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown)
5. default - has credit in default? - (categorical) (no, yes, unknown)
6. housing - Has housing loan? - (categorical) (no, yes, unknown)
7. loan - has personal loan? - (categorical) (no, yes, unknown')
8. contact – last contact month of year - (categorical) (cellular, telephone)
9. month - Month of last contact with client - (categorical) (January - December)
10. day - last contact day of the month - (categorical) (1-31)
11. duration - last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no').

12. campaign: number of contacts performed during this campaign and for this client (numeric)

13. pdays - number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means clients were not previously contacted)

14. previous - Number of client contacts performed before this campaign - (numeric)
15. poutcome - outcome of the previous marketing campaign - (categorical) (failure, nonexistent, success)
16. balance - balance of the saving account - (numeric)

Output variable (desired target) –
17. Term Deposit - has the client subscribed a term deposit? - (binary: 'yes','no')

1.1.4 Aim & Objectives

The provided dataset will be divided into training set and validation set. We are training the first set with the machine learning algorithms and to predict subscription of term deposit by the client.

Data visualization and data exploration is used to find the interesting trends and the factors affecting the term deposit subscription by the client. We are creating many models based on their resulting accuracy and other attributes and finalizing the optimal model to client's subscription.

2 Methodology & Analysis

2.1 Data Pre-processing

For better analysis, we have mutated many attributes based on the logical segregation. The below mentioned attributes were mutated.

1. Age_group - Grouped age data into 4 categories.

If age is less than 21, we consider it as 'Below 20' group. If age is between 21 and 40, we consider it as 'Between 21 and 40' group. If age is between 41 and 60, we consider it as 'Between 41 and 60' group. If age is greater than 60, we consider it as 'Above 61' group.

2. Cust_group - Grouped balance data into 4 categories

If balance is less than 0, we consider it as 'Below zero' group. If balance is between 1 and mean of the total balance amount , we consider it as 'below average' group. If balance is between mean of the total balance amount and 50000 , we consider it as 'Above average' group. If balance is greater than 50000, we consider it as 'HNI' group.

3. day_group - Grouped day data into 3 categories

If day is between 1 and 10 , we consider it as 'Early Month' group. If day is between 11 and 20 , we consider it as 'Mid month' group. If day is between 21 and 31, we consider it as 'Month end' group.

4. Duration_group - Grouped duration data into 4 categories

If duration is between 1 and mean of the duration , we consider it as 'Less Duration' group. If duration is between mean of the duration and 750 , we consider it as 'Good Duration' group. If duration is between 750 and 1500, we consider it as 'High Duration' group. If duration is above 1500, we consider it as 'Very High Duration' group.

5. Campaign_group - Grouped campaign data into 4 categories

If campaign is between 1 and mean of the campaign , we consider it as 'Average Campaign' group. If campaign is between mean of the campaign and 10, we consider it as 'Average Campaign' group. If campaign is between 10 and 20, we consider it as 'Ample Campaign' group. If campaign is above 20, we consider it as 'Heavy Campaign' group.

6. pdays_group - Grouped pdays data into 4 categories

If pdays is between -1 and mean of the pdays , we consider it as 'Less gap Pdays' group. If pdays is between mean of the pdays and 100, we consider it as 'Medium gap Pdays' group. If pdays is greater 100, we consider it as 'Large gap Pdays' group.

7. previous_group - Grouped previous data into 3 categories

If previous is 0 , we consider it as 'Zero' group. If previous is 1, we consider it as 'One' group. If previous is more than 1, we consider it as 'More than once' group.

8. y-output is "no", then 0 and if it is "yes", then 1.

```
Portdata <- as.data.frame(Portdata)

Portdata <- Portdata %>% mutate(Age_group = ifelse(age < 21,
            "Below 20", ifelse(between(age, 21,40),
            "Between 21 and 40", ifelse(between(age,41,60),
            "Between 41 and 60", "Above 61"))))

Portdata <- Portdata %>% mutate(Cust_group = ifelse(balance <0,
            "below zero", ifelse(between(balance,1,mean(balance)),
            "below average",
            ifelse(between(balance,mean(balance),50000),
            "Above average", "HNI" ))))

Portdata <- Portdata %>% mutate(day_group =
            ifelse(between(day,1,10),"Early Month",
            ifelse(between(day,11,20),"Mid Month", "Month End")))

Portdata <- Portdata %>% mutate(Duration_group =
            ifelse(between(duration,0,mean(duration)),"Less
            Duration", ifelse(between(duration,mean(duration),750),
            "Good Duration", ifelse(between(duration, 750, 1500),
            "High Duration", "Very High Duration" ))))


Portdata <- Portdata %>% mutate(Campaign_group =
            ifelse(between(campaign,1,mean(campaign)),
            "Lean Campaign", ifelse(between(campaign,
            mean(campaign),10),"Average Campaign",
            ifelse(between(campaign, 10, 20),
            "Ample Campaign", "Heavy Campaign" ))))

Portdata <- Portdata %>% mutate(pdays_group =
            ifelse(between(pdays,-1,mean(pdays)), "Less gap
            Pdays", ifelse( between(pdays,mean(pdays), 100),
            "Medium gap Pdays", "Large gap Pdays")))

Portdata <- Portdata %>% mutate(previous_group =
            ifelse(previous == 0, "Zero", ifelse(previous==1,
            "One", "More than once")))

Portdata <- Portdata %>% mutate(y_output = ifelse(y == "no", 0, 1))
```

```
Portdata <- Portdata %>% mutate(y_output = as.factor(y_output))

## Removing original columns
Portdata <- Portdata %>% select(-age,-balance,-day,-duration,-campaign,-pdays,-previous,-y)

## converting into factor
Portdata <- Portdata %>% mutate(Age_group = as.factor(Age_group),
                                Cust_group = as.factor(Cust_group),
                                day_group = as.factor(day_group),
                                Duration_group = as.factor(Duration_group),
                                Campaign_group = as.factor(Campaign_group),
                                previous_group = as.factor(previous_group),
                                Duration_group = as.factor(Duration_group),
                                y_output = as.factor(y_output) )

## Converting all characters into factors
Portdata=Portdata %>% mutate_if(is.character, as.factor)
```

2.2 Data Visualization and Data Exploration

2.2.1 General Data Information

```
# The first 6 rows of the dataset will be displayed to review the dataset.
head(Portdata)
```

```
##            job marital education default housing loan contact month poutcome
## 1   management married  tertiary      no     yes   no unknown   may  unknown
## 2   technician  single secondary      no     yes   no unknown   may  unknown
## 3 entrepreneur married secondary      no     yes  yes unknown   may  unknown
## 4  blue-collar married   unknown      no     yes   no unknown   may  unknown
## 5      unknown  single   unknown      no      no   no unknown   may  unknown
## 6   management married  tertiary      no     yes   no unknown   may  unknown
##           Age_group    Cust_group   day_group              Duration_group
## 1 Between 41 and 60 Above average Early Month               Good Duration
## 2 Between 41 and 60 below average Early Month Less \n            Duration
## 3 Between 21 and 40 below average Early Month Less \n            Duration
## 4 Between 41 and 60 Above average Early Month Less \n            Duration
## 5 Between 21 and 40 below average Early Month Less \n            Duration
## 6 Between 21 and 40 below average Early Month Less \n            Duration
##   Campaign_group                  pdays_group previous_group y_output
## 1  Lean Campaign Less gap \n            Pdays           Zero        0
## 2  Lean Campaign Less gap \n            Pdays           Zero        0
## 3  Lean Campaign Less gap \n            Pdays           Zero        0
## 4  Lean Campaign Less gap \n            Pdays           Zero        0
## 5  Lean Campaign Less gap \n            Pdays           Zero        0
## 6  Lean Campaign Less gap \n            Pdays           Zero        0
```

```
# Summary Statistics of Portdata which explains the overview of the all columns.
summary(Portdata)
```

```
##           job           marital          education     default     housing
```

```
##  blue-collar:9732   divorced: 5207   primary  : 6851   no :44396   no :20081
##  management :9458   married :27214   secondary:23202   yes:  815   yes:25130
##  technician :7597   single  :12790   tertiary :13301
##  admin.     :5171                    unknown  : 1857
##  services   :4154
##  retired    :2264
##  (Other)    :6835
##   loan            contact          month          poutcome
##  no :37967   cellular :29285   may    :13766   failure: 4901
##  yes: 7244   telephone: 2906   jul    : 6895   other  : 1840
##              unknown  :13020   aug    : 6247   success: 1511
##                                jun    : 5341   unknown:36959
##                                nov    : 3970
##                                apr    : 2932
##                                (Other): 6060
##              Age_group             Cust_group          day_group
##  Above 61          : 1188   Above average:11730   Early Month:13725
##  Below 20          :   97   below average:26183   Mid Month  :18389
##  Between 21 and 40:24620   below zero   : 3766   Month End  :13097
##  Between 41 and 60:19306   HNI          : 3532
##
##
##
##                      Duration_group           Campaign_group
##  Good Duration               :12770   Ample Campaign  :  952
##  High Duration               : 2035   Average Campaign:13966
##  Less \n            Duration:30179   Heavy Campaign  :  244
##  Very High Duration          :  227   Lean Campaign   :30049
##
##
##
##                         pdays_group           previous_group  y_output
##  Large gap Pdays              : 6820   More than once: 5485   0:39922
##  Less gap \n           Pdays:37180   One           : 2772   1: 5289
##  Medium gap Pdays             : 1211   Zero          :36954
##
##
##
##
```

2.3 Data exploration

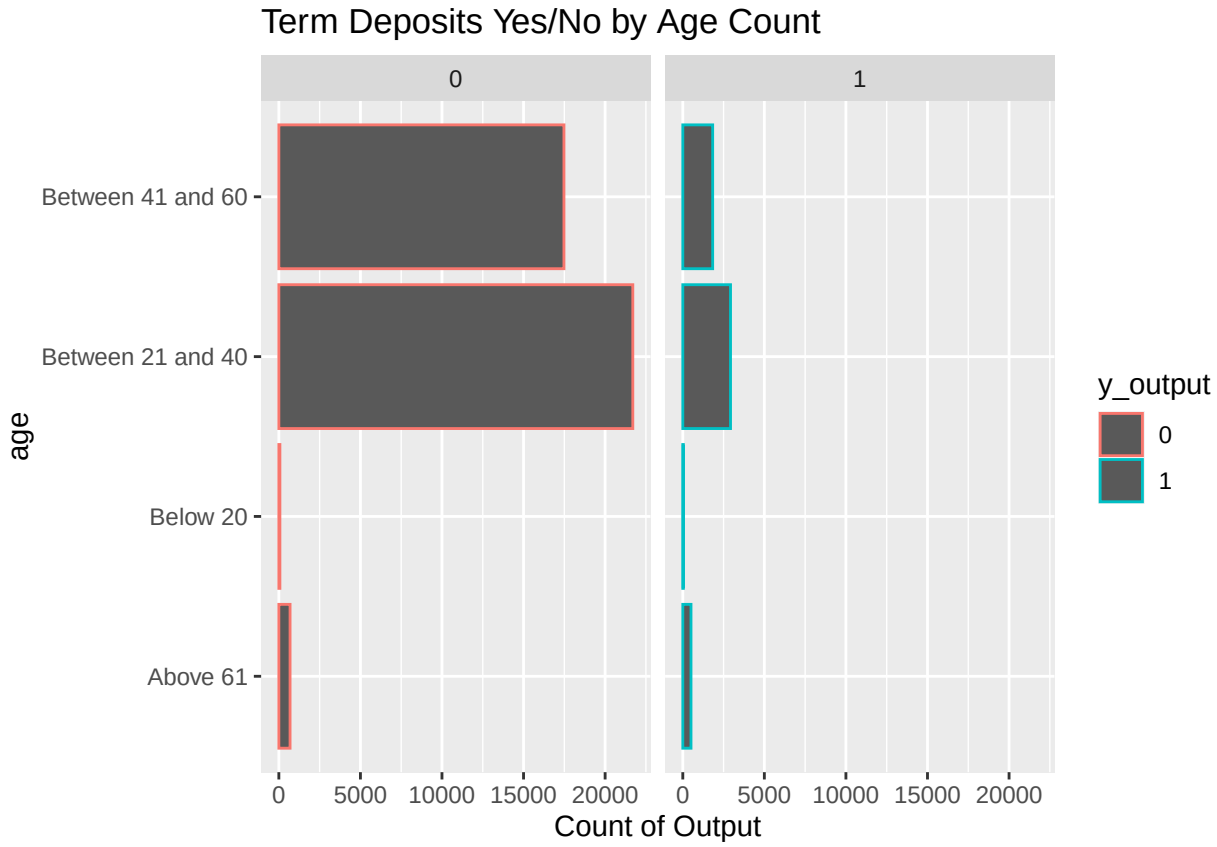
Each attribute of the dataset has been explored properly by comparing the count of the y_output.

```
### Exploration of Age attribute

Age_count <- data.frame(Portdata %>% group_by(Age_group, y_output) %>% summarise(Count = n()))

Age_count <- Age_count %>% spread(y_output,Count) %>%
rename(No = `0`, Yes = `1` ) %>%
mutate(Percent_of_yes = Yes /(Yes + No)) %>%
arrange(desc(Percent_of_yes))
```

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

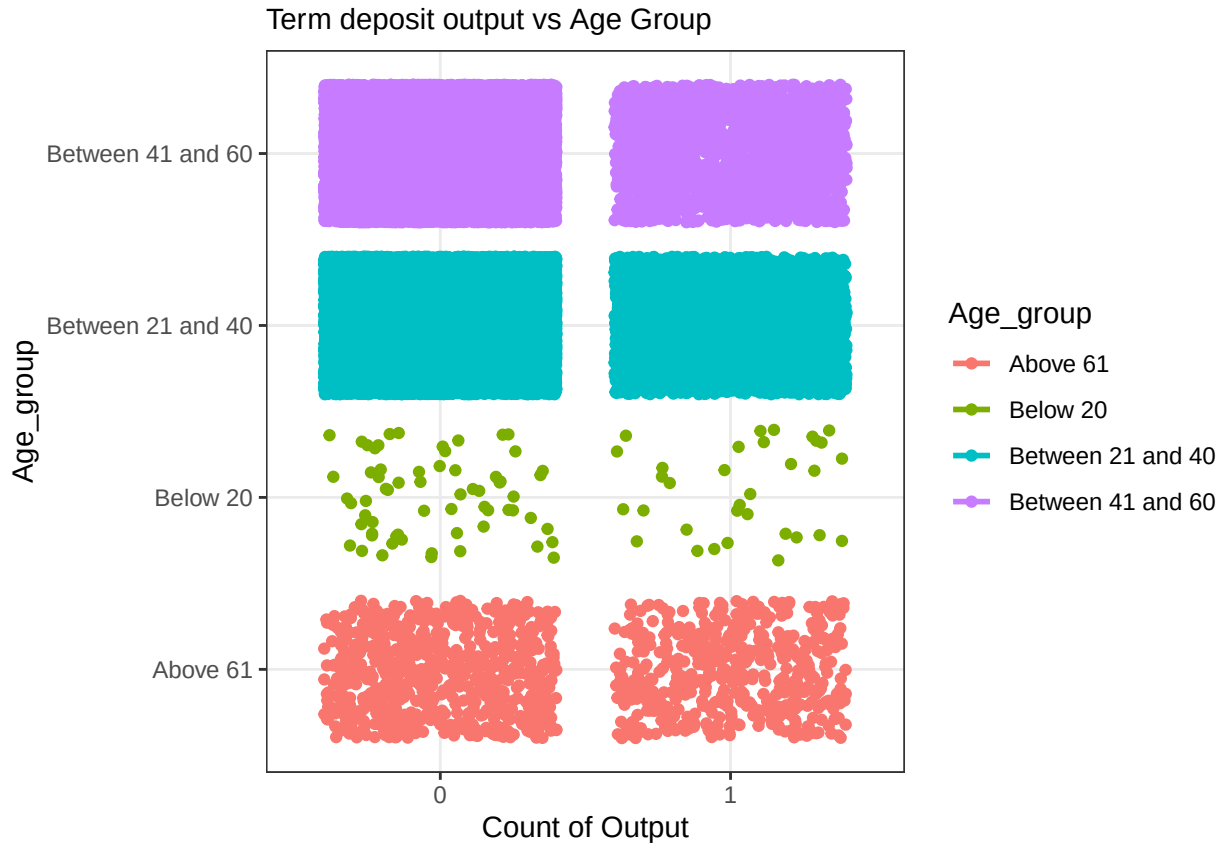
Age_group	No	Yes	Percent_of_yes
Above 61	686	502	0.4225589
Below 20	64	33	0.3402062
Between 21 and 40	21696	2924	0.1187652
Between 41 and 60	17476	1830	0.0947892

```
Portdata %>% ggplot(aes(Age_group, color = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by Age Count", x="age",
  y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
theme_set(theme_bw())  # pre-set the bw theme.
g <- Portdata %>% ggplot( aes(y_output, Age_group)) +
labs(subtitle="Term deposit output vs Age Group",
Y="Age Group", x="Count of Output")

g + geom_jitter(aes(col=Age_group)) +
  geom_smooth(aes(col=Age_group), method="lm", se=F)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



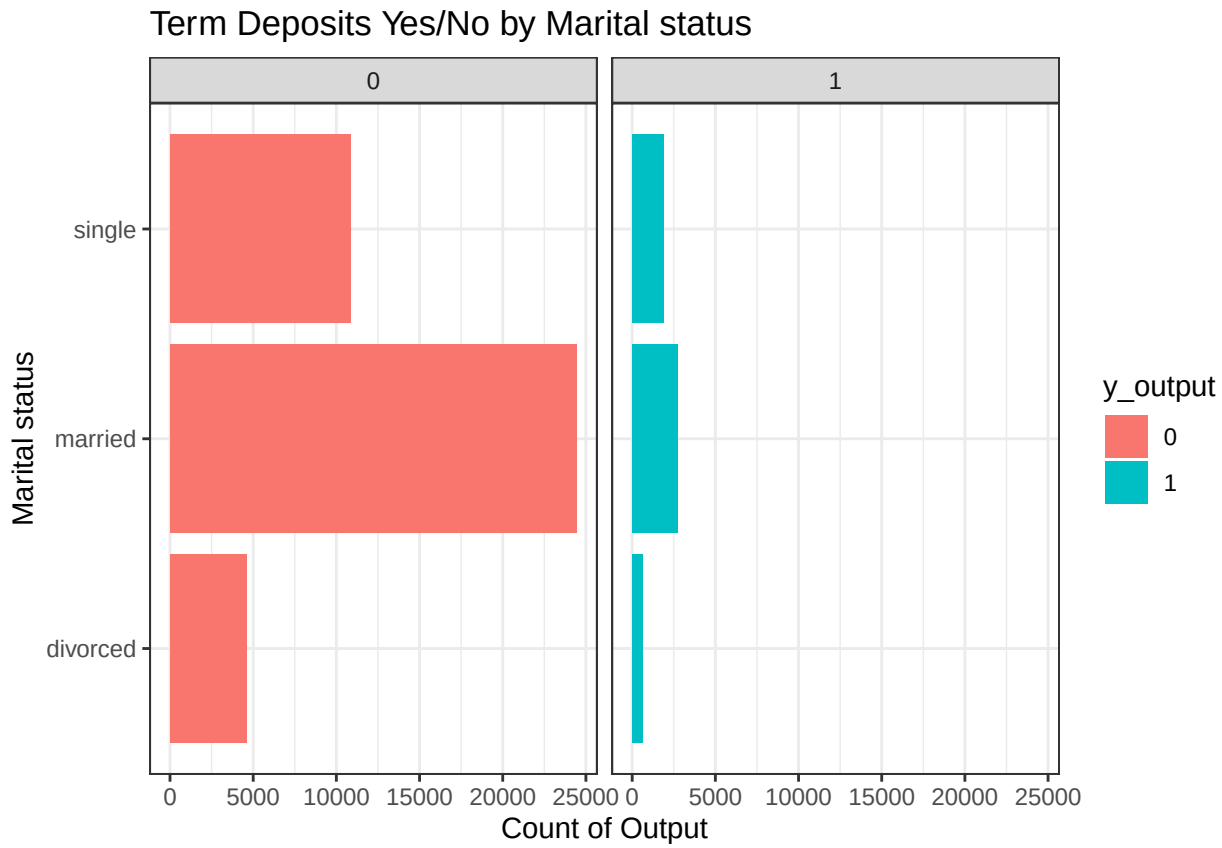

```
### Exploration of Job atrribute

job_count <- data.frame(Portdata %>% group_by(job,y_output) %>% summarise(Count = n()))

job_count <- job_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

job	No	Yes	Percent_of_yes
student	669	269	0.2867804
retired	1748	516	0.2279152
unemployed	1101	202	0.1550269
management	8157	1301	0.1375555
admin.	4540	631	0.1220267
self-employed	1392	187	0.1184294
unknown	254	34	0.1180556
technician	6757	840	0.1105700
services	3785	369	0.0888300
housemaid	1131	109	0.0879032
entrepreneur	1364	123	0.0827169
blue-collar	9024	708	0.0727497

```
### Exploration of Marital status

Marital_count <- data.frame(Portdata %>% group_by(marital,y_output) %>% summarise(Count = n()))

Marital_count <- Marital_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

marital	No	Yes	Percent_of_yes
single	10878	1912	0.1494918
divorced	4585	622	0.1194546
married	24459	2755	0.1012347

```
Portdata %>% ggplot(aes(marital, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by Marital status", x="Marital status", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of Education attribute

Education_count <- data.frame(Portdata %>% group_by(education,y_output) %>% summarise(Count = n()))
```

```
Education_count <- Education_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

education	No	Yes	Percent_of_yes
tertiary	11305	1996	0.1500639
unknown	1605	252	0.1357027
secondary	20752	2450	0.1055943
primary	6260	591	0.0862648

```
### Exploration of Default attribute

Default_count <- data.frame(Portdata %>% group_by(default,y_output) %>% summarise(Count = n()))

Default_count <- Default_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

default	No	Yes	Percent_of_yes
no	39159	5237	0.1179611
yes	763	52	0.0638037

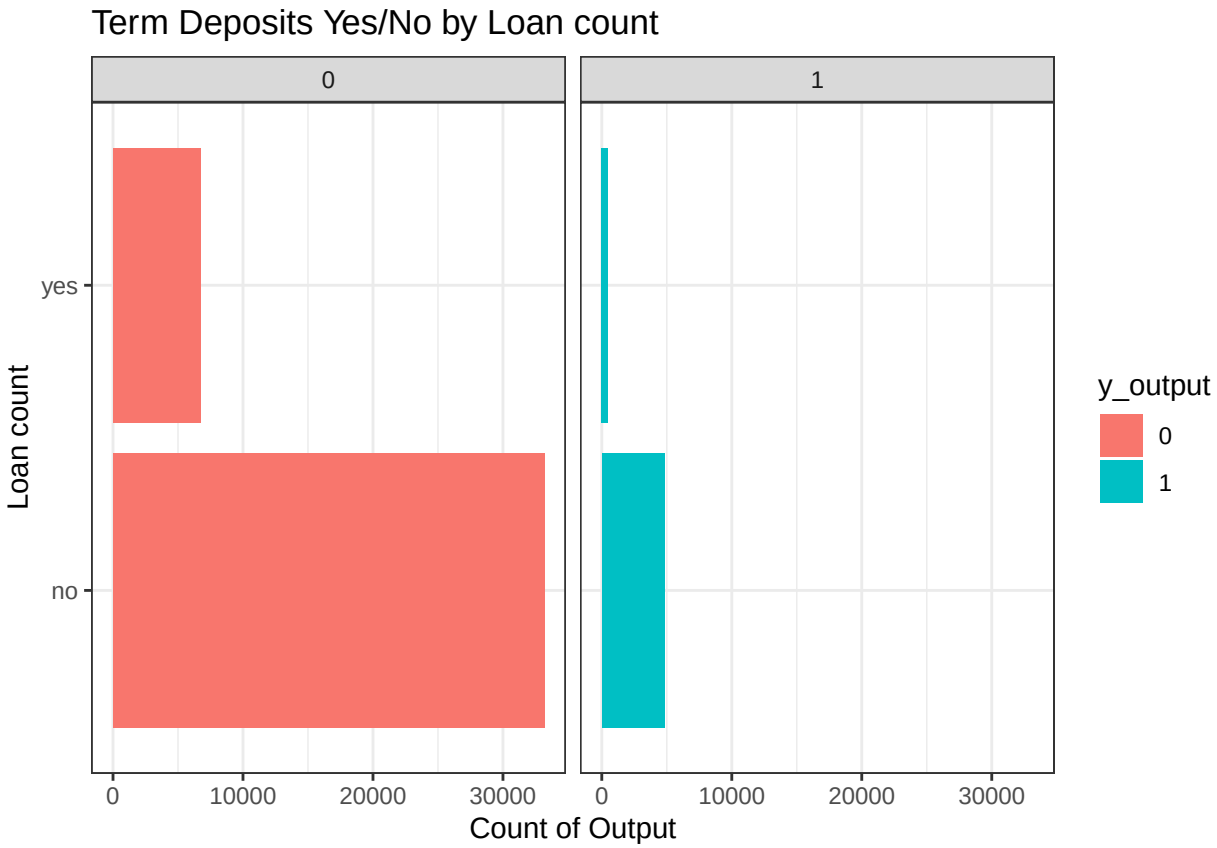
```
### Exploration of Housing attribute

Housing_count <- data.frame(Portdata %>% group_by(housing,y_output) %>% summarise(Count = n()))

Housing_count <- Housing_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

housing	No	Yes	Percent_of_yes
no	16727	3354	0.1670236
yes	23195	1935	0.0769996

```
### Exploration of Loan attribute

Loan_count <- data.frame(Portdata %>% group_by(loan,y_output) %>% summarise(Count = n()))

Loan_count <- Loan_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

loan	No	Yes	Percent_of_yes
no	33162	4805	0.1265573
yes	6760	484	0.0668139

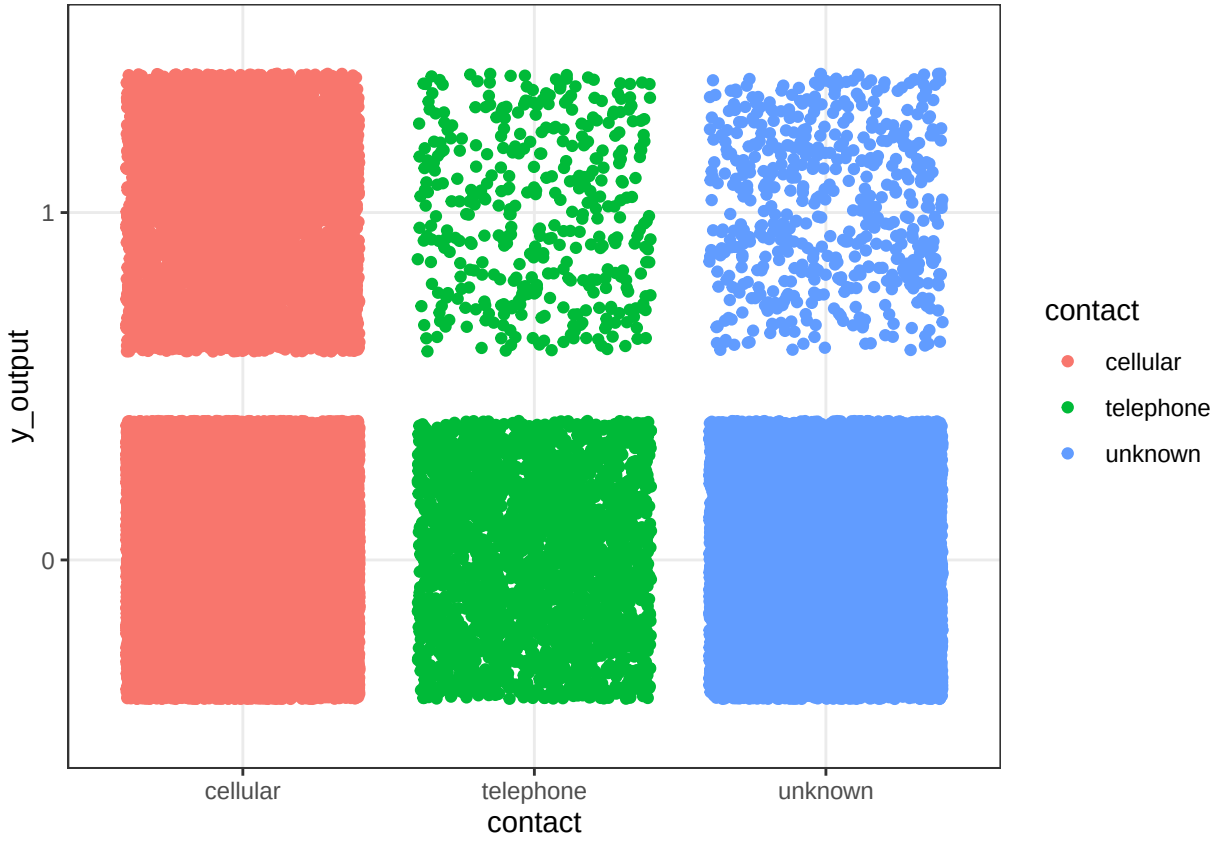
```
Portdata %>% ggplot(aes(loan, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by Loan count", x="Loan count", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of Contact attribute

Contact_count <- data.frame(Portdata %>% group_by(contact,y_output) %>% summarise(Count = n()))

Contact_count <- Contact_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

contact	No	Yes	Percent_of_yes
cellular	24916	4369	0.1491890
telephone	2516	390	0.1342051
unknown	12490	530	0.0407066

```
Portdata %>% ggplot(aes(contact, y_output, color = contact)) +
  geom_jitter()
```

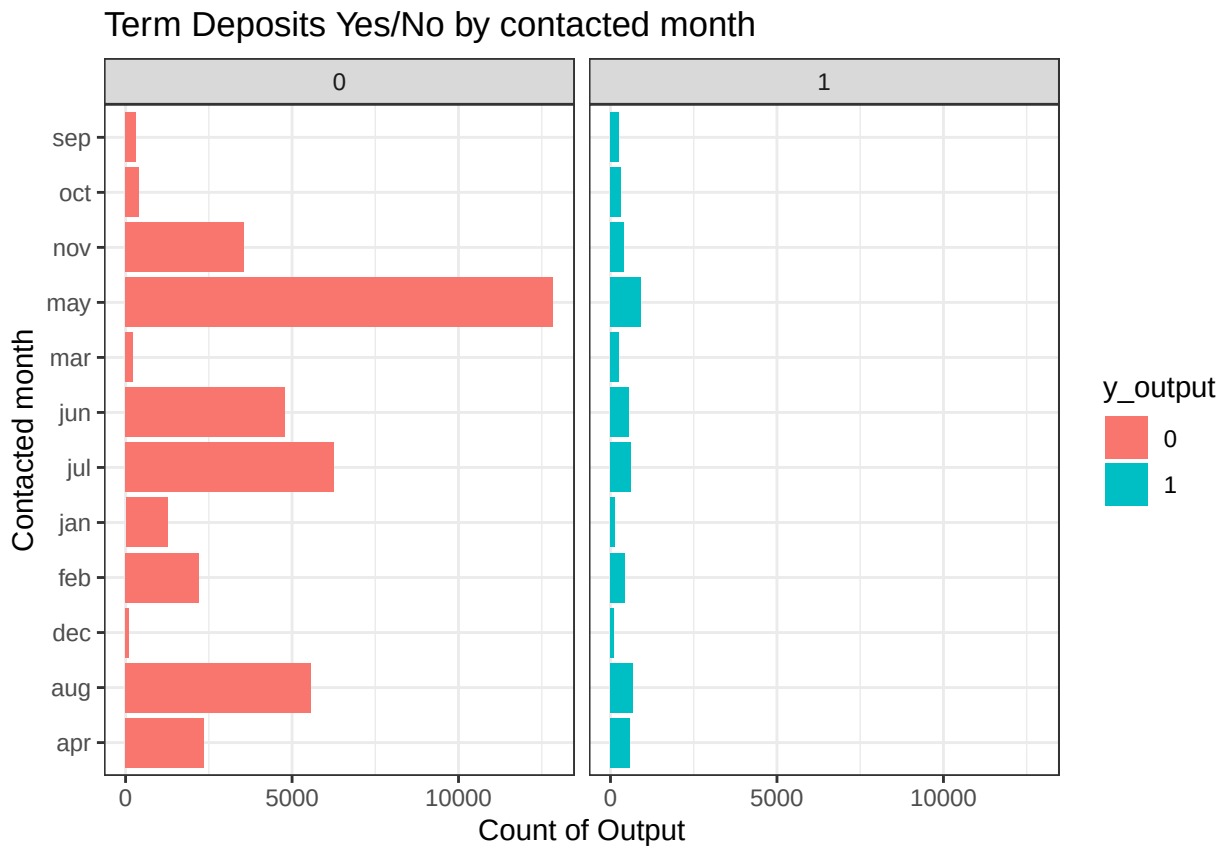



```
### Exploration of Month attribute

Month_count <- data.frame(Portdata %>% group_by(month,y_output) %>% summarise(Count = n()))

Month_count <- Month_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

month	No	Yes	Percent_of_yes
mar	229	248	0.5199161
dec	114	100	0.4672897
sep	310	269	0.4645941
oct	415	323	0.4376694
apr	2355	577	0.1967940
feb	2208	441	0.1664779
aug	5559	688	0.1101329
jun	4795	546	0.1022280
nov	3567	403	0.1015113
jan	1261	142	0.1012117
jul	6268	627	0.0909355

month	No	Yes	Percent_of_yes
may	12841	925	0.0671945

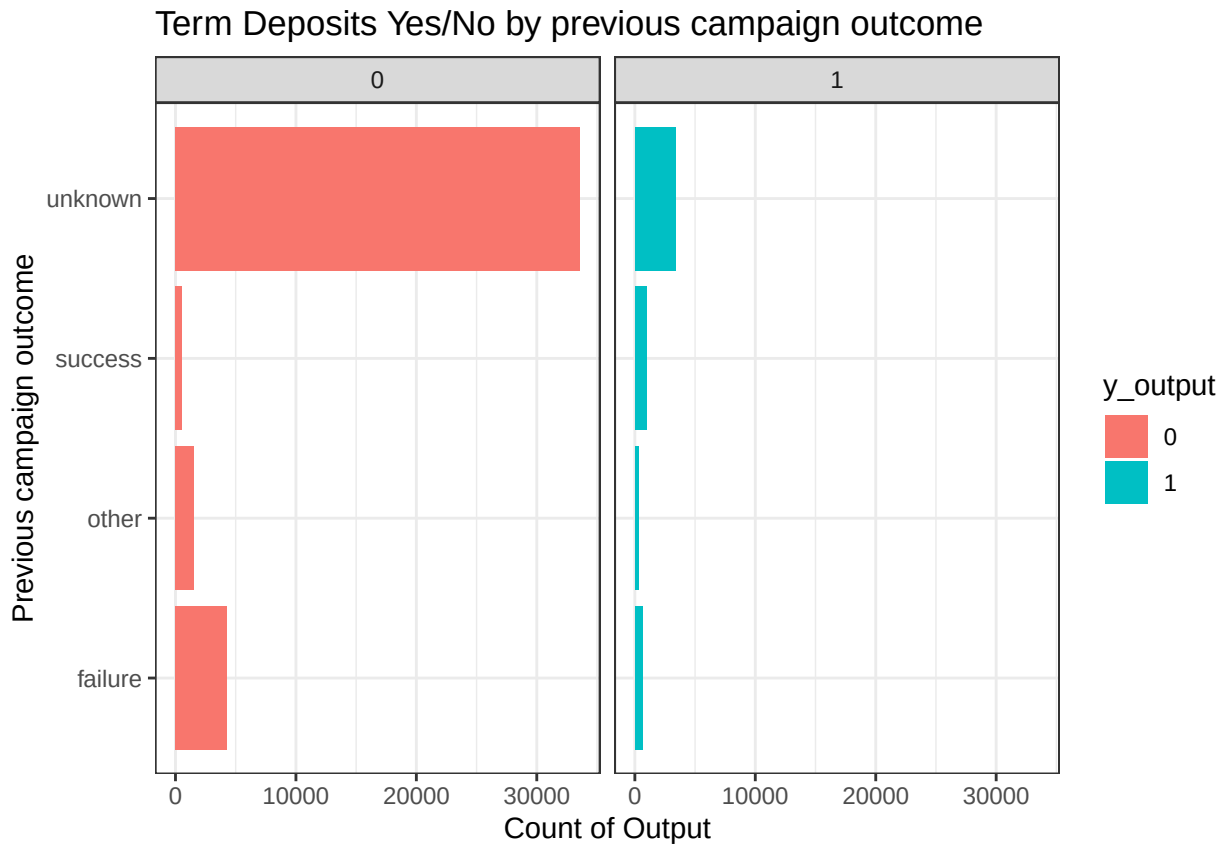
```
Portdata %>% ggplot(aes(month, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by contacted month", x="Contacted month", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of Poutcome attribute

Pout_count <- data.frame(Portdata %>% group_by(poutcome,y_output) %>% summarise(Count = n()))

Pout_count <- Pout_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

poutcome	No	Yes	Percent_of_yes
success	533	978	0.6472535
other	1533	307	0.1668478
failure	4283	618	0.1260967
unknown	33573	3386	0.0916150

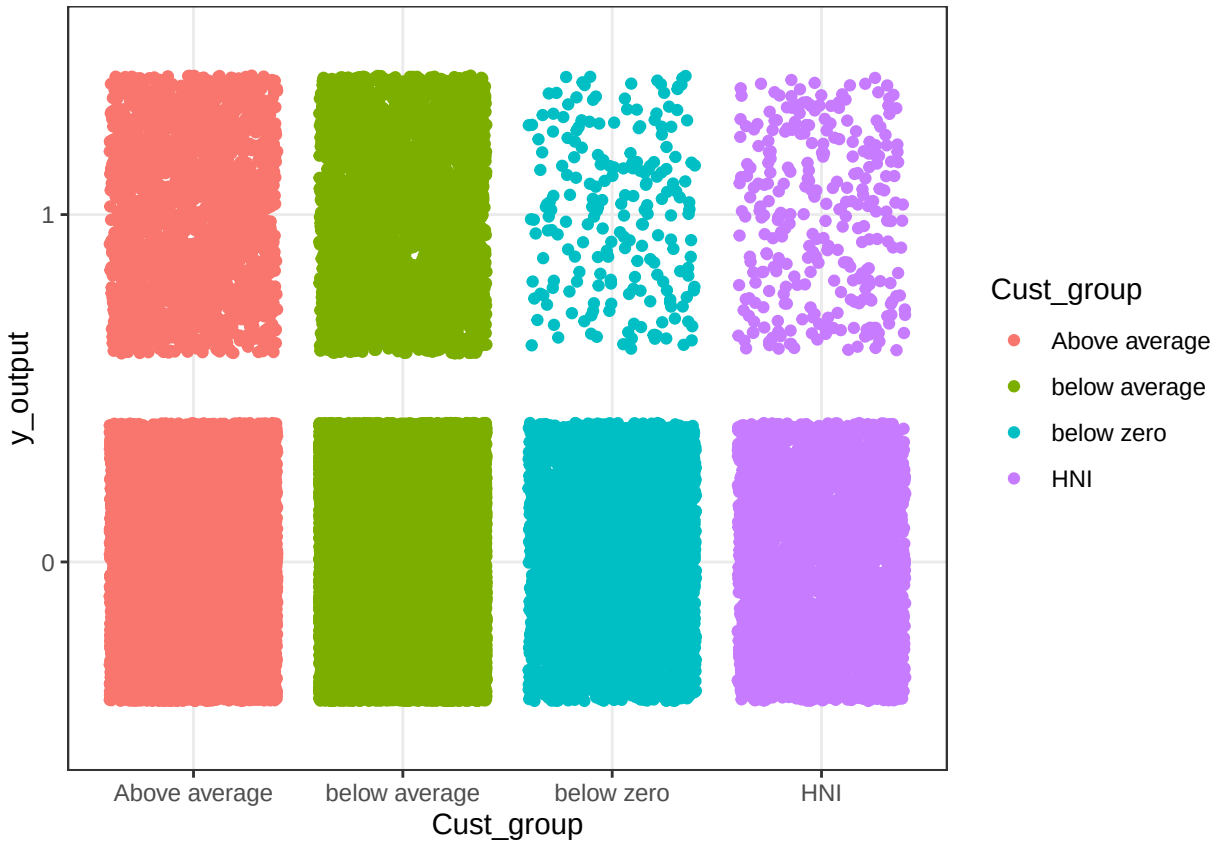
```
Portdata %>% ggplot(aes(poutcome, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by previous campaign outcome",
  x="Previous campaign outcome", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of Customer attribute

Cust_count <- data.frame(Portdata %>% group_by(Cust_group,y_output) %>% summarise(Count = n()))

Cust_count <- Cust_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

Cust_group	No	Yes	Percent_of_yes
Above average	9855	1875	0.1598465
below average	23275	2908	0.1110644
HNI	3236	296	0.0838052
below zero	3556	210	0.0557621

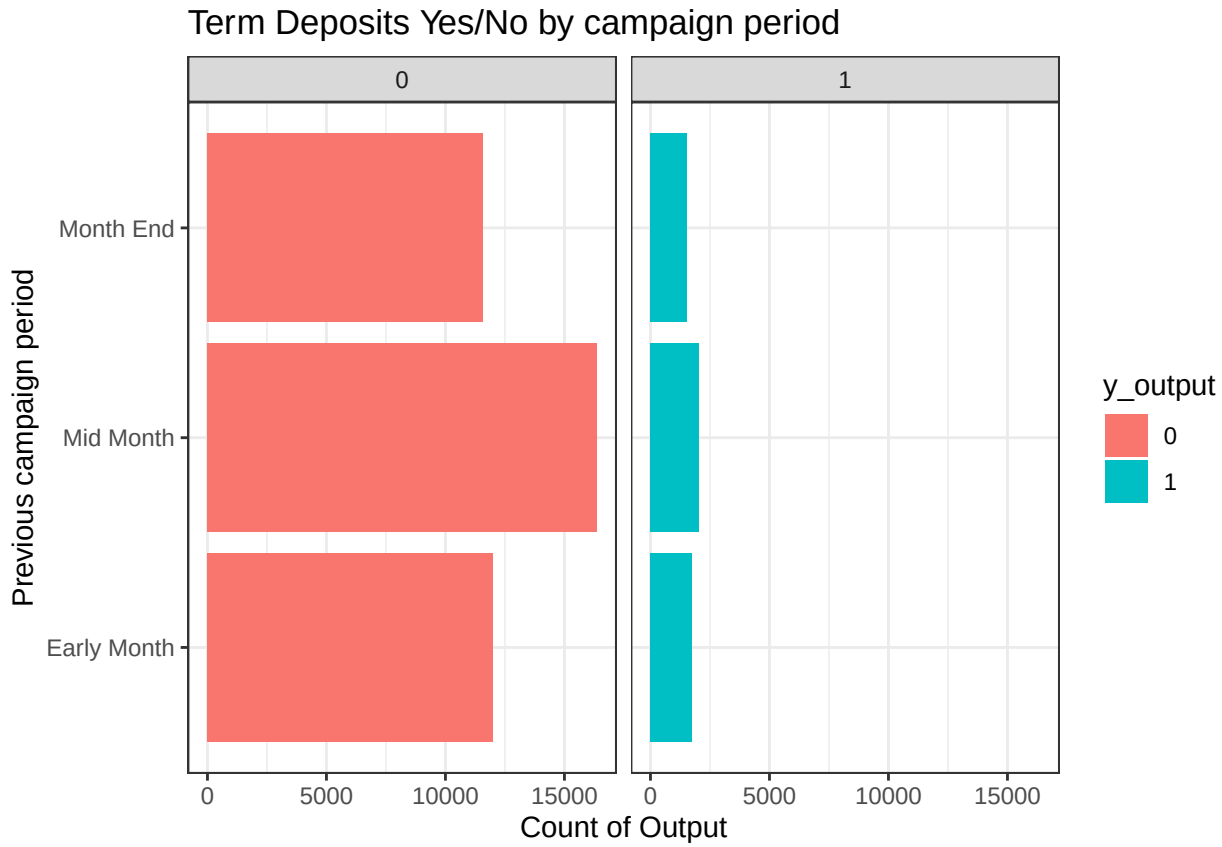
```
Portdata %>% ggplot(aes(Cust_group, y_output, color = Cust_group)) +
  geom_jitter()
```




```
### Exploration of day_group attributes

Day_count <- data.frame(Portdata %>% group_by(day_group,y_output) %>% summarise(Count = n()))

Day_count <- Day_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

day_group	No	Yes	Percent_of_yes
Early Month	11991	1734	0.1263388
Month End	11566	1531	0.1168970
Mid Month	16365	2024	0.1100658

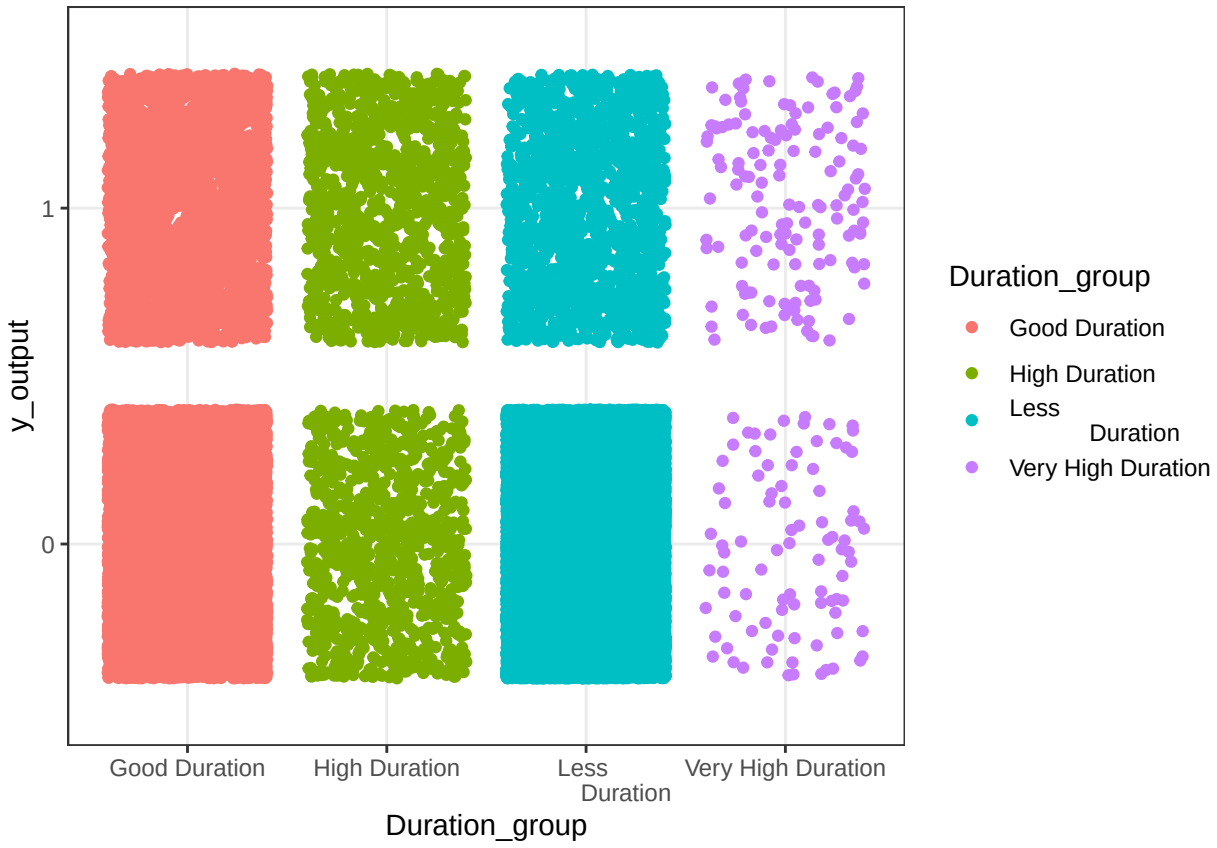
```
Portdata %>% ggplot(aes(day_group, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by campaign period",
  x="Previous campaign period", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of Duration attributes

Duration_count <- data.frame(Portdata %>% group_by(Duration_group,y_output) %>% summarise(Count = n()))

Duration_count <- Duration_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

Duration_group	No	Yes	Percent_of_yes
Very High Duration	89	138	0.6079295
High Duration	940	1095	0.5380835
Good Duration	10176	2594	0.2031323
Less Duration 28717	1462		0.0484443

```
Portdata %>% ggplot(aes(Duration_group, y_output, color = Duration_group)) +
  geom_jitter()
```



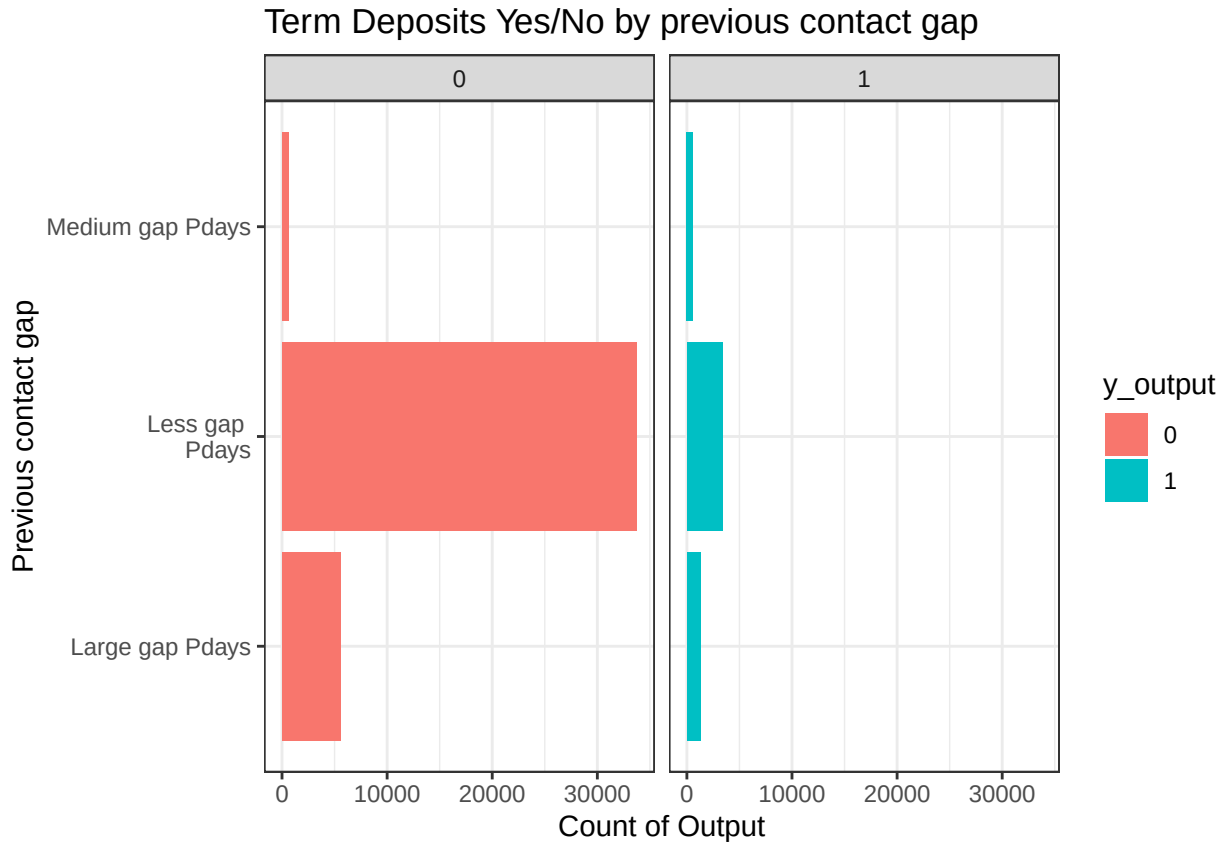

```
### Exploration of Campaign_group attributes

Campaign_count <- data.frame(Portdata %>% group_by(Campaign_group,y_output) %>% summarise(Count = n()))

Campaign_count <- Campaign_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

Campaign_group	No	Yes	Percent_of_yes
Lean Campaign	26087	3962	0.1318513
Average Campaign	12686	1280	0.0916512
Ample Campaign	909	43	0.0451681
Heavy Campaign	240	4	0.0163934

```
### Exploration of pdays_group attributes

pdays_count <- data.frame(Portdata %>% group_by(pdays_group,y_output) %>% summarise(Count = n()))

pdays_count <- pdays_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

pdays_group	No	Yes	Percent_of_yes
Medium gap Pdays	630	581	0.4797688
Large gap Pdays	5537	1283	0.1881232
Less gap Pdays 33755 3	425	0.09	21194

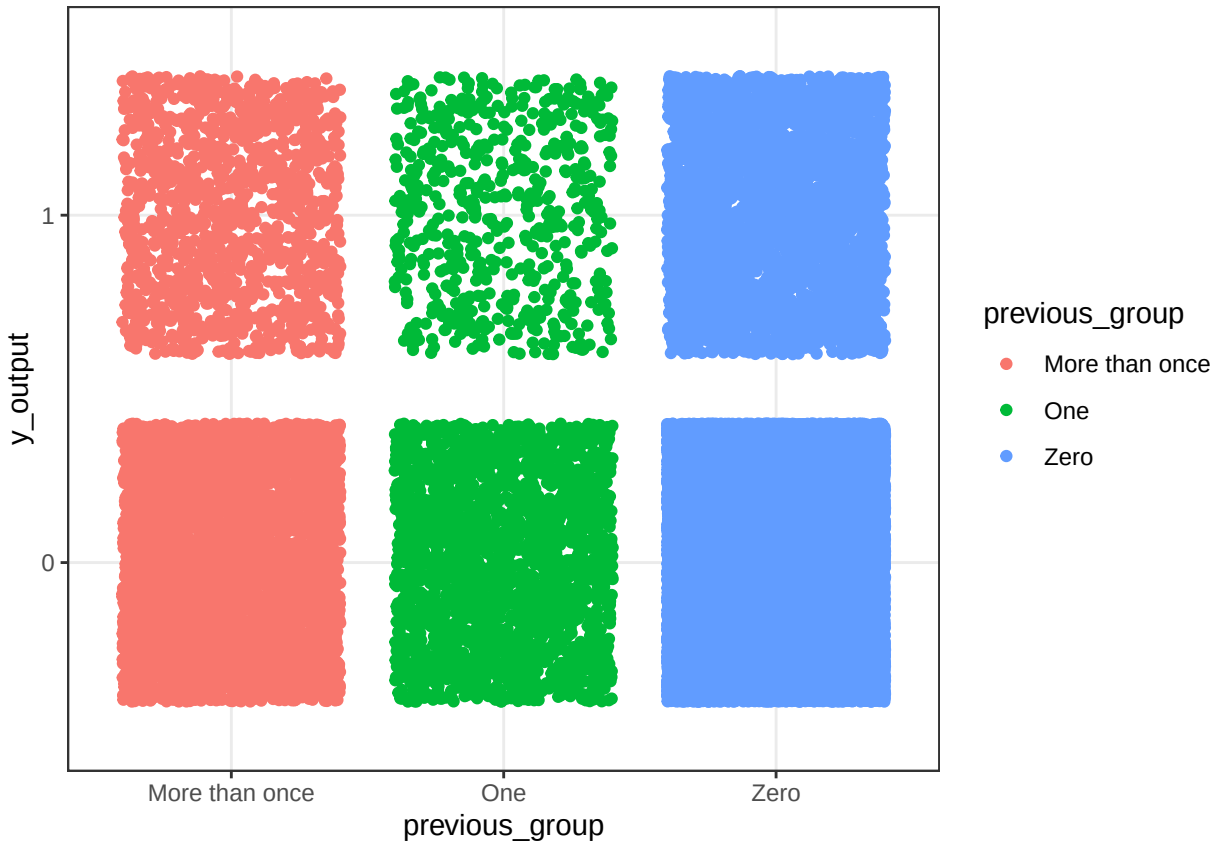
```
Portdata %>% ggplot(aes(pdays_group, fill = y_output)) +
  geom_bar() +
  labs(title = "Term Deposits Yes/No by previous contact gap",
  x="Previous contact gap", y="Count of Output") +
  facet_wrap(y_output ~ .) +
  coord_flip()
```




```
### Exploration of previous_group attributes

previous_count <- data.frame(Portdata %>% group_by(previous_group,y_output) %>% summarise(Count = n()))

previous_count <- previous_count %>% spread(y_output,Count) %>% rename(No = `0`, Yes = `1` ) %>%
  mutate(Percent_of_yes = Yes /(Yes + No)) %>% arrange(desc(Percent_of_yes))

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

previous_group	No	Yes	Percent_of_yes
More than once	4163	1322	0.2410210
One	2189	583	0.2103175
Zero	33570	3384	0.0915733

```
Portdata %>% ggplot(aes(previous_group, y_output, color = previous_group)) +
  geom_jitter()
```



Based on the above exploration, we can understand that some of the attributes contribute more to the clients subscription decision. We have mentioned below the list of attributes in the order of high impact.

1. Duration_group - last contact duration
2. Month - Month of last contact with client
3. Poutcome - outcome of the previous marketing campaign
4. Age_group - Client Age
5. pdays_group - number of days that passed by after the client was last contacted from a previous campaign

2.3.1 Splitting of dataset

We had partitioned the banking dataset into 2 sets[main dataset and validation dataset in the 90:10 ratio]. The main datset is further split into train and test dataset[in the 80:20 ratio]. We were training the dataset with the train set and testing with the test dataset.Then, based on the test results, we had finalised the model. Finally, we implemented the model in the validation dataset for the conclusion.

```r
# Validation set will be 10% of bank marketing data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
Valid_index <- createDataPartition(y = Portdata$y, times = 1, p = 0.1, list = FALSE)
Portdata_main <- Portdata[-Valid_index,]
Portdata_validation <- Portdata[Valid_index,]

# Splitting Portdata_main into 2 sets for initial testing
# Test dataset is 20% of the Portdata_main dataset
set.seed(2, sample.kind="Rounding")
```

```
## Warning in set.seed(2, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = Portdata_main$y, times = 1, p = 0.2, list = FALSE)
Portdata_main_train <- Portdata_main[-test_index,]
Portdata_main_test <- Portdata_main[test_index,]
```

2.4 Data Analysis and modelling

2.4.1 GLM:

Logistic regression is useful when you are predicting a binary outcome from a set of continuous predictor variables. It is frequently preferred over discriminant function analysis because of its less restrictive assumptions. In the logit model the log odds of the outcome is modeled as a linear combination of the predictor variables.

```r
glm_fit <- Portdata_main_train %>%
    glm(y_output ~ ., data=., family = binomial(link='logit'))

p_hat_logit <- predict(glm_fit, newdata = Portdata_main_test, type = "response")
y_hat_logit <- ifelse(p_hat_logit > 0.25,1, 0) %>% factor

### Cross table of GLM - All attributes
CrossTable(Portdata_main_test$y_output, y_hat_logit,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =====================================
```

```
##                   predicted default
## actual default        0       1   Total
## ---------------------------------------
## 0                  6660     526    7186
##                   0.818   0.065
## ---------------------------------------
## 1                   387     565     952
##                   0.048   0.069
## ---------------------------------------
## Total              7047    1091    8138
## =======================================
```

```
### Confusion matrix of GLM - All attributes
conf_glm <- confusionMatrix(y_hat_logit,
Portdata_main_test$y_output, mode = "prec_recall", positive = '1')

conf_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6660  387
##          1  526  565
##
##                Accuracy : 0.8878
##                  95% CI : (0.8808, 0.8946)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.09154
##
##                   Kappa : 0.4893
##
##  Mcnemar's Test P-Value : 4.944e-06
##
##               Precision : 0.51787
##                  Recall : 0.59349
##                      F1 : 0.55311
##              Prevalence : 0.11698
##          Detection Rate : 0.06943
##    Detection Prevalence : 0.13406
##       Balanced Accuracy : 0.76014
##
##        'Positive' Class : 1
##
```

```
accuracy_glm = conf_glm$overall[["Accuracy"]]
Sensitivity_glm = conf_glm$byClass[["Sensitivity"]]
Specificity_glm = conf_glm$byClass[["Specificity"]]
Precision_glm = conf_glm$byClass[["Precision"]]
F1_glm = conf_glm$byClass[["F1"]]

## creation of confusion table for GLM - All attributes
Confusion_table <- data_frame(method = "GLM - All attributes",
```

```
      Accuracy =accuracy_glm,SensitivityorRecall = Sensitivity_glm,
      Specificity=Specificity_glm,Precision = Precision_glm,
      F1= F1_glm  )
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
Confusion_table <- as.data.frame(Confusion_table)

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082

```
# selected attributes

glm_fit_all <- Portdata_main_train %>%
glm(y_output ~ Duration_group+month+poutcome+Age_group+pdays_group,
data=., family = binomial(link='logit'))
p_hat_logit_all <- predict(glm_fit_all, newdata = Portdata_main_test, type = "response")
y_hat_logit_all <- ifelse(p_hat_logit_all > 0.25,1, 0) %>% factor

### Cross table of GLM - Selected attributes
CrossTable(Portdata_main_test$y_output, y_hat_logit_all,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## ========================================
##                   predicted default
## actual default        0       1   Total
## ----------------------------------------
## 0                  6705     481    7186
##                   0.824   0.059
## ----------------------------------------
## 1                   449     503     952
##                   0.055   0.062
## ----------------------------------------
## Total              7154     984    8138
## ========================================
```

```
### Confusion matrix of GLM - Selected attributes
conf_glm_all <- confusionMatrix(y_hat_logit_all,
Portdata_main_test$y_output, mode = "prec_recall", positive = '1')
```

```
conf_glm_all
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6705  449
##          1  481  503
##
##                Accuracy : 0.8857
##                  95% CI : (0.8786, 0.8926)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.2298
##
##                   Kappa : 0.4548
##
##  Mcnemar's Test P-Value : 0.3094
##
##               Precision : 0.51118
##                  Recall : 0.52836
##                      F1 : 0.51963
##              Prevalence : 0.11698
##          Detection Rate : 0.06181
##    Detection Prevalence : 0.12091
##       Balanced Accuracy : 0.73071
##
##        'Positive' Class : 1
##
```

```
accuracy_glm_all = conf_glm_all$overall[["Accuracy"]]
Sensitivity_glm_all = conf_glm_all$byClass[["Sensitivity"]]
Specificity_glm_all = conf_glm_all$byClass[["Specificity"]]
Precision_glm_all = conf_glm_all$byClass[["Precision"]]
F1_glm_all = conf_glm_all$byClass[["F1"]]


## Binding confusion table for GLM - All attributes
Confusion_table <- bind_rows(Confusion_table,
    data_frame(method = "GLM - Selected attributes",
    Accuracy =accuracy_glm_all,SensitivityorRecall =
    Sensitivity_glm_all, Specificity=Specificity_glm_all,
    Precision = Precision_glm_all, F1= F1_glm_all))


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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281

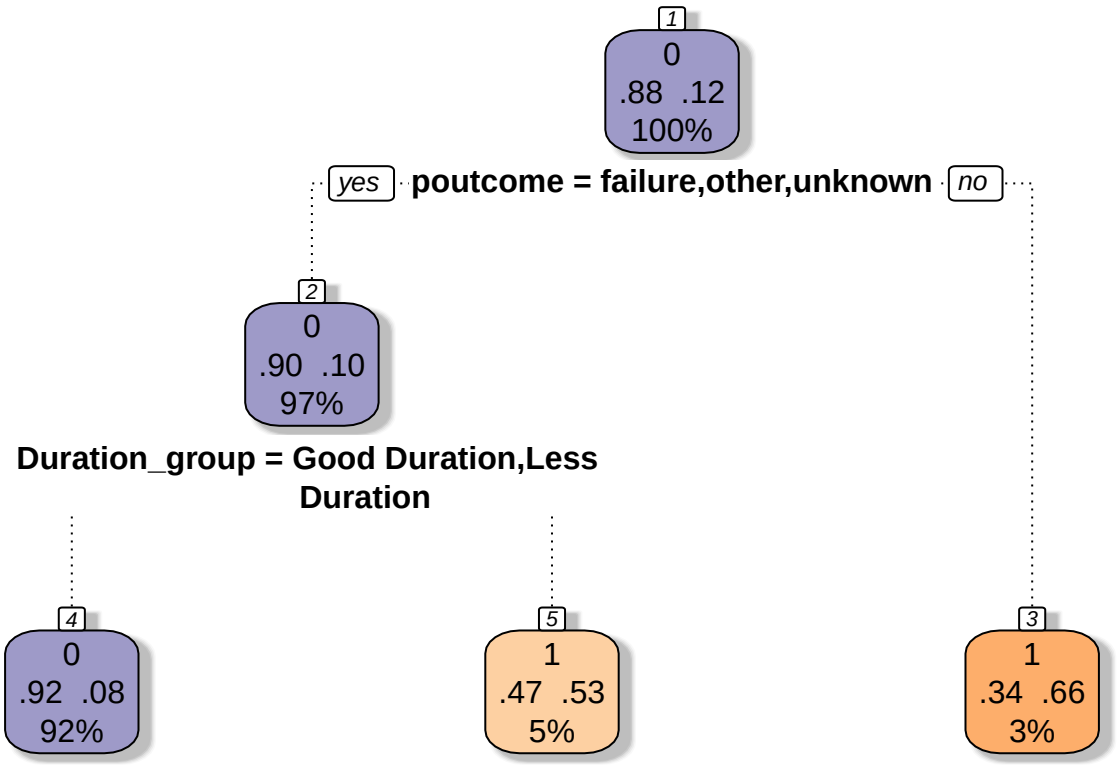
2.4.2 RPART:

The rpart algorithm works by splitting the dataset recursively, which means that the subsets that arise from a split are further split until a predetermined termination criterion is reached. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the dependent (predicted) variable.

```
# All attributes

model_rpart = rpart(formula = y_output ~ .,
                    data = Portdata_main_train, method = "class")
p_hat_logit <- predict(model_rpart, newdata = Portdata_main_test, type = "class")

### Cross table of Rpart - All attributes
CrossTable(Portdata_main_test$y_output, p_hat_logit,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =======================================
##                   predicted default
## actual default        0       1   Total
## ---------------------------------------
## 0                  6896     290    7186
##                   0.847   0.036
## ---------------------------------------
## 1                   575     377     952
##                   0.071   0.046
## ---------------------------------------
## Total              7471     667    8138
## =======================================
```

```
## RPLOT for the model
fancyRpartPlot(model_rpart ,digits=2, type=2,palettes = c("Purples", "Oranges"))
```




```
### Confusion matrix of Rpart - All attributes
conf_rpart <- confusionMatrix(p_hat_logit,
Portdata_main_test$y_output, mode = "prec_recall", positive = '1')

conf_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6896  575
##          1  290  377
##
##                Accuracy : 0.8937
##                  95% CI : (0.8868, 0.9003)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.001265
##
##                   Kappa : 0.4087
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.56522
##                  Recall : 0.39601
##                      F1 : 0.46572
##              Prevalence : 0.11698
```

```
##          Detection Rate : 0.04633
##    Detection Prevalence : 0.08196
##       Balanced Accuracy : 0.67783
##
##        'Positive' Class : 1
##
```

```
accuracy_rpart = conf_rpart$overall[["Accuracy"]]
Sensitivity_rpart = conf_rpart$byClass[["Sensitivity"]]
Specificity_rpart = conf_rpart$byClass[["Specificity"]]
Precision_rpart = conf_rpart$byClass[["Precision"]]
F1_rpart = conf_rpart$byClass[["F1"]]

## Binding confusion table for Rpart - All attributes
Confusion_table <- bind_rows(Confusion_table,
       data_frame(method = "Rpart - All attributes",
       Accuracy =accuracy_rpart,SensitivityorRecall =
       Sensitivity_rpart, Specificity=Specificity_rpart,
       Precision = Precision_rpart, F1= F1_rpart))

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

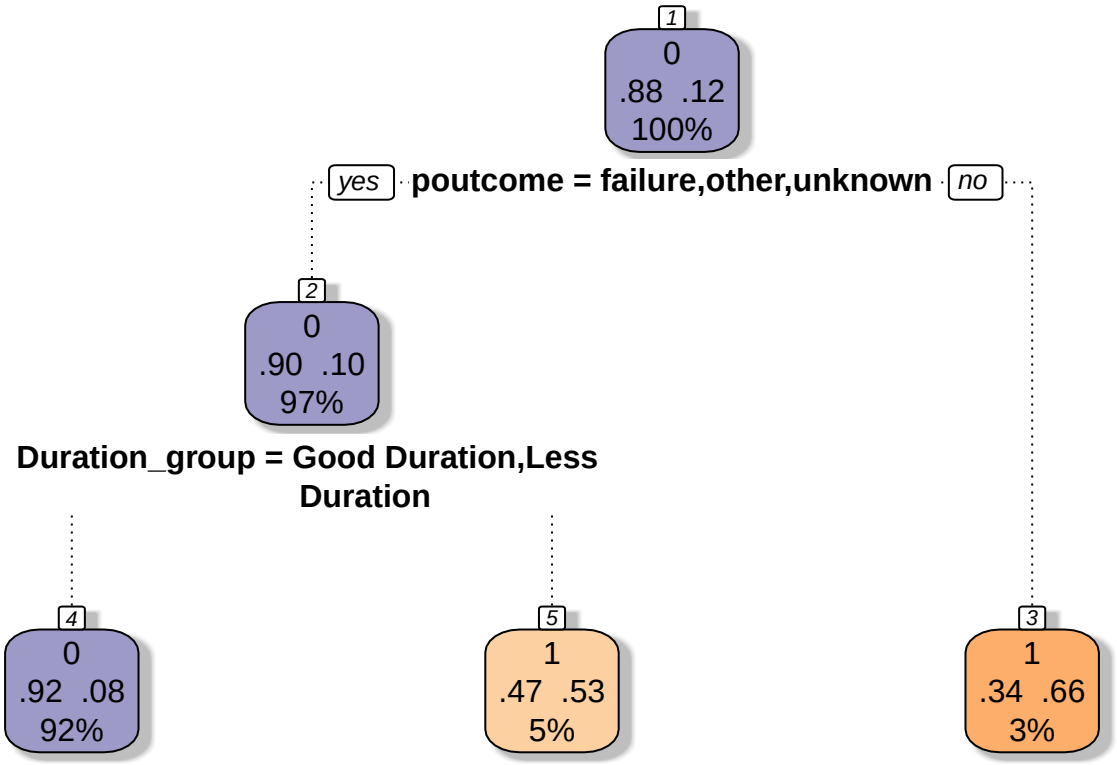
method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196

```
# Selected attributes

model_rpart_select = rpart(formula = y_output ~ Duration_group+month+poutcome+Age_group+pdays_group,
                   data = Portdata_main_train, method = "class")

p_hat_logit_select <- predict(model_rpart_select, newdata = Portdata_main_test, type = "class")

### Cross table of Rpart - Selected attributes
CrossTable(Portdata_main_test$y_output, p_hat_logit_select,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =======================================
##                  predicted default
## actual default        0       1   Total
## ---------------------------------------
## 0                  6896     290    7186
##                   0.847   0.036
```

```
## ----------------------------------------
## 1                   575     377     952
##                   0.071   0.046
## ----------------------------------------
## Total              7471     667    8138
## ========================================
```

```
### RPLOT for the model
fancyRpartPlot(model_rpart_select ,digits=2, type=2,palettes = c("Purples", "Oranges"))
```




```
### Confusion matrix of Rpart - Selected attributes
conf_rpart_select <- confusionMatrix(p_hat_logit_select,
Portdata_main_test$y_output,mode = "prec_recall", positive = '1')

conf_rpart_select
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6896  575
##          1  290  377
##
##                Accuracy : 0.8937
##                  95% CI : (0.8868, 0.9003)
```

```
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.001265
##
##                   Kappa : 0.4087
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.56522
##                  Recall : 0.39601
##                      F1 : 0.46572
##              Prevalence : 0.11698
##          Detection Rate : 0.04633
##    Detection Prevalence : 0.08196
##       Balanced Accuracy : 0.67783
##
##        'Positive' Class : 1
##
```

```
accuracy_rpart_select = conf_rpart_select$overall[["Accuracy"]]
Sensitivity_rpart_select = conf_rpart_select$byClass[["Sensitivity"]]
Specificity_rpart_select = conf_rpart_select$byClass[["Specificity"]]
Precision_rpart_select = conf_rpart_select$byClass[["Precision"]]
F1_rpart_select = conf_rpart_select$byClass[["F1"]]

## Binding confusion table for Rpart - Selected attributes
Confusion_table <- bind_rows(Confusion_table,
      data_frame(method = "Rpart - Selected attributes",
      Accuracy =accuracy_rpart_select,SensitivityorRecall =
      Sensitivity_rpart_select,Specificity=Specificity_rpart_select
      ,Precision = Precision_rpart_select, F1= F1_rpart_select))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
Rpart - Selected attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196

2.4.3 Randomforest:

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

```
# All attributes

model_rf = randomForest(y_output ~ .,
                   data = Portdata_main_train)

pred_rf <- predict(model_rf, newdata = Portdata_main_test, type = "class")
```

```
### Cross table of RF - All attributes
CrossTable(Portdata_main_test$y_output, pred_rf,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## ========================================
##                   predicted default
## actual default        0       1   Total
## ----------------------------------------
## 0                  6933     253    7186
##                   0.852   0.031
## ----------------------------------------
## 1                   547     405     952
##                   0.067   0.050
## ----------------------------------------
## Total              7480     658    8138
## ========================================
```

```
### Confusion matrix of RF - All attributes
conf_rf <- confusionMatrix(pred_rf, Portdata_main_test$y_output, mode = "prec_recall", positive = '1')

conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6933  547
##          1  253  405
##
##                Accuracy : 0.9017
##                  95% CI : (0.895, 0.9081)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 4.394e-08
##
##                   Kappa : 0.4506
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.61550
##                  Recall : 0.42542
##                      F1 : 0.50311
##              Prevalence : 0.11698
##          Detection Rate : 0.04977
##    Detection Prevalence : 0.08086
##       Balanced Accuracy : 0.69511
```

```
##
##        'Positive' Class : 1
##
```

```
accuracy_rf = conf_rf$overall[["Accuracy"]]
Sensitivity_rf = conf_rf$byClass[["Sensitivity"]]
Specificity_rf = conf_rf$byClass[["Specificity"]]
Precision_rf = conf_rf$byClass[["Precision"]]
F1_rf = conf_rf$byClass[["F1"]]

## Binding confusion table for RF - All attributes
Confusion_table <- bind_rows(Confusion_table,
  data_frame(method = "RF - All attributes", Accuracy =accuracy_rf,
  SensitivityorRecall = Sensitivity_rf,
  Specificity=Specificity_rf,Precision = Precision_rf, F1= F1_rf))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
Rpart - Selected attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
RF - All attributes	0.9016957	0.4254202	0.9647927	0.6155015	0.5031056

```
# Selected attributes

model_rf_select = randomForest(y_output ~ Duration_group+month+poutcome+Age_group+pdays_group,
                        data = Portdata_main_train)

pred_rf_select <- predict(model_rf_select, newdata = Portdata_main_test, type = "class")

### Cross table of RF - Selected attributes
CrossTable(Portdata_main_test$y_output, pred_rf_select,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## ========================================
##                   predicted default
## actual default        0       1   Total
## ----------------------------------------
## 0                  6985     201    7186
##                   0.858   0.025
## ----------------------------------------
## 1                   620     332     952
```

```
##                   0.076   0.041
## ---------------------------------------
## Total              7605     533    8138
## =======================================
```

```
### Confusion matrix of RF - Selected attributes
conf_rf_select <- confusionMatrix(pred_rf_select,
Portdata_main_test$y_output, mode = "prec_recall", positive = '1')

conf_rf_select
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6985  620
##          1  201  332
##
##                Accuracy : 0.8991
##                  95% CI : (0.8924, 0.9056)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 2.201e-06
##
##                   Kappa : 0.3965
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.6229
##                  Recall : 0.3487
##                      F1 : 0.4471
##              Prevalence : 0.1170
##          Detection Rate : 0.0408
##    Detection Prevalence : 0.0655
##       Balanced Accuracy : 0.6604
##
##        'Positive' Class : 1
##
```

```
accuracy_rf_select = conf_rf_select$overall[["Accuracy"]]
Sensitivity_rf_select = conf_rf_select$byClass[["Sensitivity"]]
Specificity_rf_select = conf_rf_select$byClass[["Specificity"]]
Precision_rf_select = conf_rf_select$byClass[["Precision"]]
F1_rf_select = conf_rf_select$byClass[["F1"]]

## Binding confusion table for RF - Selected attributes
Confusion_table <- bind_rows(Confusion_table,
      data_frame(method = "RF - Selected attributes",
      Accuracy =accuracy_rf_select,SensitivityorRecall =
      Sensitivity_rf_select, Specificity=Specificity_rf_select,
      Precision = Precision_rf_select, F1= F1_rf_select))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
Rpart - Selected attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
RF - All attributes	0.9016957	0.4254202	0.9647927	0.6155015	0.5031056
RF - Selected attributes	0.8991153	0.3487395	0.9720289	0.6228893	0.4471380

2.4.4 KNN:

K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase.

```
# ALL Attributes

model_knn <- train(y_output ~ ., data = Portdata_main_train, method = "knn",
                   maximize = TRUE,
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess=c("center", "scale"))

pred_kNN <- predict(model_knn , Portdata_main_test)

### Cross table of KNN-All attributes
CrossTable(Portdata_main_test$y_output, pred_kNN,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =======================================
##                   predicted default
## actual default        0       1   Total
## ---------------------------------------
## 0                  6981     205    7186
##                   0.858   0.025
## ---------------------------------------
## 1                   651     301     952
##                   0.080   0.037
## ---------------------------------------
## Total              7632     506    8138
## =======================================
```

```
### Confusion matrix of KNN-All attributes
conf_knn <- confusionMatrix(pred_kNN , Portdata_main_test$y_output,
```

```
            mode = "prec_recall", positive = '1')
conf_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6981  651
##          1  205  301
##
##                Accuracy : 0.8948
##                  95% CI : (0.8879, 0.9014)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.0004199
##
##                   Kappa : 0.361
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.59486
##                  Recall : 0.31618
##                      F1 : 0.41289
##              Prevalence : 0.11698
##          Detection Rate : 0.03699
##    Detection Prevalence : 0.06218
##       Balanced Accuracy : 0.64382
##
##        'Positive' Class : 1
##
```

```
accuracy_knn = conf_knn$overall[["Accuracy"]]
Sensitivity_knn = conf_knn$byClass[["Sensitivity"]]
Specificity_knn = conf_knn$byClass[["Specificity"]]
Precision_knn = conf_knn$byClass[["Precision"]]
F1_knn = conf_knn$byClass[["F1"]]

## Binding confusion table for KNN-All attributes
Confusion_table <- bind_rows(Confusion_table,
      data_frame(method = "KNN-All attributes",
      Accuracy =accuracy_knn,SensitivityorRecall = Sensitivity_knn,
      Specificity=Specificity_knn,Precision = Precision_knn,
      F1= F1_knn))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
Rpart - Selected attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
RF - All attributes	0.9016957	0.4254202	0.9647927	0.6155015	0.5031056
RF - Selected attributes	0.8991153	0.3487395	0.9720289	0.6228893	0.4471380

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
KNN-All attributes	0.8948145	0.3161765	0.9714723	0.5948617	0.4128944

2.4.5 Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

As there were many parameters present in the confusion matrix, we were concentrating on the below parameters for our analysis.

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

Specificity - The proportion of observed negatives that were predicted to be negatives. In other words, of all the transactions that were legitimate, what percentage did we predict to be so?

Specificity = TN/FP+TN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

F1 Score = 2*(Recall* Precision) / (Recall + Precision)

Final confusion matrix table as below:

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - All attributes	0.8878103	0.5934874	0.9268021	0.5178735	0.5531082
GLM - Selected attributes	0.8857213	0.5283613	0.9330643	0.5111789	0.5196281
Rpart - All attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
Rpart - Selected attributes	0.8937085	0.3960084	0.9596438	0.5652174	0.4657196
RF - All attributes	0.9016957	0.4254202	0.9647927	0.6155015	0.5031056
RF - Selected attributes	0.8991153	0.3487395	0.9720289	0.6228893	0.4471380
KNN-All attributes	0.8948145	0.3161765	0.9714723	0.5948617	0.4128944

3 Results and Discussion

Based on the above confusion matrix details of all the analysed models, we shortlisted GLM - All attributes and GLM - Selected attributes as the F1 and Recall values are good, and RF - All atrributes and Rf - Selected atrributes as the Accuracy, Precision and Specificity are good.

Hence, we implemented the above 4 models into the Validation set for the final conclusion.

```
#GLM:

glm_fit_final <- Portdata_main %>%
  glm(y_output ~ ., data=., family = binomial(link='logit'))
p_hat_logit_final <- predict(glm_fit_final, newdata = Portdata_validation, type = "response")
y_hat_logit_final <- ifelse(p_hat_logit_final > 0.25,1, 0) %>% factor

### Cross table of GLM - Final - All
CrossTable(Portdata_validation$y_output, y_hat_logit_final,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =======================================
##                   predicted default
## actual default        0       1   Total
## ---------------------------------------
## 0                  3702     291    3993
##                   0.819   0.064
## ---------------------------------------
## 1                    197     332     529
##                   0.044   0.073
## ---------------------------------------
## Total              3899     623    4522
## =======================================
```

```
### Confusion matrix of GLM - Final - All
conf_glm_final <- confusionMatrix(y_hat_logit_final,
Portdata_validation$y_output, mode = "prec_recall", positive = '1')

conf_glm_final
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3702  197
##          1  291  332
##
##                Accuracy : 0.8921
```

```
##                  95% CI : (0.8827, 0.901)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.02942
##
##                   Kappa : 0.515
##
##  Mcnemar's Test P-Value : 2.555e-05
##
##               Precision : 0.53291
##                  Recall : 0.62760
##                      F1 : 0.57639
##              Prevalence : 0.11698
##          Detection Rate : 0.07342
##    Detection Prevalence : 0.13777
##       Balanced Accuracy : 0.77736
##
##        'Positive' Class : 1
##
```

```
accuracy_glm_final = conf_glm_final$overall[["Accuracy"]]
Sensitivity_glm_final = conf_glm_final$byClass[["Sensitivity"]]
Specificity_glm_final = conf_glm_final$byClass[["Specificity"]]
Precision_glm_final = conf_glm_final$byClass[["Precision"]]
F1_glm_final = conf_glm_final$byClass[["F1"]]

## Creating confusion table final for GLM - Final - All
Confusion_table_final <- data_frame(method = "GLM - Final - All",
          Accuracy =accuracy_glm_final,
          SensitivityorRecall = Sensitivity_glm_final,
          Specificity=Specificity_glm_final,
          Precision = Precision_glm_final, F1= F1_glm_final  )

Confusion_table_final <- as.data.frame(Confusion_table_final)

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - Final - All	0.8920831	0.6275992	0.9271225	0.5329053	0.5763889

```
# Selected attributes

glm_fit_select_final <- Portdata_main %>%
glm(y_output ~ Duration_group+month+poutcome+Age_group+pdays_group,
data=., family = binomial(link='logit'))
p_hat_logit_select_final <- predict(glm_fit_select_final, newdata =
      Portdata_validation, type = "response")
y_hat_logit_select_final <- ifelse(p_hat_logit_select_final > 0.25,1, 0) %>% factor

### Cross table of GLM - Final - selected
CrossTable(Portdata_validation$y_output, y_hat_logit_select_final,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
```

```
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =====================================
##                  predicted default
## actual default        0       1   Total
## -------------------------------------
## 0                  3673     320    3993
##                   0.812   0.071
## -------------------------------------
## 1                   204     325     529
##                   0.045   0.072
## -------------------------------------
## Total              3877     645    4522
## =====================================
```

```
### Confusion matrix of GLM - Final - selected
conf_glm_select_final <- confusionMatrix(y_hat_logit_select_final,
Portdata_validation$y_output, mode = "prec_recall", positive = '1')

conf_glm_select_final
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3673  204
##          1  320  325
##
##                Accuracy : 0.8841
##                  95% CI : (0.8744, 0.8933)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.4197
##
##                   Kappa : 0.4878
##
##  Mcnemar's Test P-Value : 5.066e-07
##
##               Precision : 0.50388
##                  Recall : 0.61437
##                      F1 : 0.55366
##              Prevalence : 0.11698
##          Detection Rate : 0.07187
##    Detection Prevalence : 0.14264
##       Balanced Accuracy : 0.76711
##
##        'Positive' Class : 1
##
```

```
accuracy_glm_select_final = conf_glm_select_final$overall[["Accuracy"]]
Sensitivity_glm_select_final = conf_glm_select_final$byClass[["Sensitivity"]]
Specificity_glm_select_final = conf_glm_select_final$byClass[["Specificity"]]
Precision_glm_select_final = conf_glm_select_final$byClass[["Precision"]]
F1_glm_select_final = conf_glm_select_final$byClass[["F1"]]

## Binding confusion table final for GLM - Final - selected
Confusion_table_final <- bind_rows(Confusion_table_final,
      data_frame(method = "GLM - Final - selected",
      Accuracy =accuracy_glm_select_final,SensitivityorRecall =
      Sensitivity_glm_select_final,
      Specificity=Specificity_glm_select_final,Precision =
      Precision_glm_select_final, F1= F1_glm_select_final))


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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - Final - All	0.8920831	0.6275992	0.9271225	0.5329053	0.5763889
GLM - Final - selected	0.8841221	0.6143667	0.9198598	0.5038760	0.5536627

```
##Randomforest

# All attributes

model_rf_final = randomForest(y_output ~ .,
                        data = Portdata_main)

pred_rf_final <- predict(model_rf_final, newdata = Portdata_validation, type = "class")

### Cross table of RF - Final - All
CrossTable(Portdata_validation$y_output, pred_rf_final,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## =====================================
##                   predicted default
## actual default        0       1   Total
## -------------------------------------
## 0                  3851     142    3993
##                   0.852   0.031
## -------------------------------------
## 1                   276     253     529
##                   0.061   0.056
## -------------------------------------
```

```
## Total              4127     395    4522
## ======================================
```

```
### Confusion matrix of RF - Final - All
conf_rf_final <- confusionMatrix(pred_rf_final,
Portdata_validation$y_output, mode = "prec_recall", positive = '1')

conf_rf_final
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3851  276
##          1  142  253
##
##                Accuracy : 0.9076
##                  95% CI : (0.8987, 0.9159)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 6.569e-08
##
##                   Kappa : 0.4973
##
##  Mcnemar's Test P-Value : 7.757e-11
##
##               Precision : 0.64051
##                  Recall : 0.47826
##                      F1 : 0.54762
##              Prevalence : 0.11698
##          Detection Rate : 0.05595
##    Detection Prevalence : 0.08735
##       Balanced Accuracy : 0.72135
##
##        'Positive' Class : 1
##
```

```
accuracy_rf_final = conf_rf_final$overall[["Accuracy"]]
Sensitivity_rf_final = conf_rf_final$byClass[["Sensitivity"]]
Specificity_rf_final = conf_rf_final$byClass[["Specificity"]]
Precision_rf_final = conf_rf_final$byClass[["Precision"]]
F1_rf_final = conf_rf_final$byClass[["F1"]]

## Binding confusion table final for RF - Final - All
Confusion_table_final <- bind_rows(Confusion_table_final,
    data_frame(method = "RF - Final - All",
    Accuracy =accuracy_rf_final,SensitivityorRecall =
    Sensitivity_rf_final,Specificity=Specificity_rf_final,
    Precision = Precision_rf_final, F1= F1_rf_final))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - Final - All	0.8920831	0.6275992	0.9271225	0.5329053	0.5763889
GLM - Final - selected	0.8841221	0.6143667	0.9198598	0.5038760	0.5536627
RF - Final - All	0.9075630	0.4782609	0.9644378	0.6405063	0.5476190

```
# Selected attributes

model_rf_select_final = randomForest(y_output ~ Duration_group+month+poutcome+Age_group+pdays_group,
                              data = Portdata_main)

pred_rf_select_final <- predict(model_rf_select_final, newdata = Portdata_validation, type = "class")

### Cross table of RF - Final - Selected
CrossTable(Portdata_validation$y_output, pred_rf_select_final,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
## ========================================
##                   predicted default
## actual default        0       1   Total
## ----------------------------------------
## 0                  3881     112    3993
##                   0.858   0.025
## ----------------------------------------
## 1                   329     200     529
##                   0.073   0.044
## ----------------------------------------
## Total              4210     312    4522
## ========================================
```

```
### Confusion matrix of RF - Final - Selected
conf_rf_select_final <- confusionMatrix(pred_rf_select_final,
Portdata_validation$y_output, mode = "prec_recall", positive = '1')

conf_rf_select_final
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3881  329
##          1  112  200
##
##                Accuracy : 0.9025
##                  95% CI : (0.8935, 0.911)
##     No Information Rate : 0.883
```

```
##     P-Value [Acc > NIR] : 1.685e-05
##
##                   Kappa : 0.4258
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.64103
##                  Recall : 0.37807
##                      F1 : 0.47562
##              Prevalence : 0.11698
##          Detection Rate : 0.04423
##    Detection Prevalence : 0.06900
##       Balanced Accuracy : 0.67501
##
##        'Positive' Class : 1
##
```

```
accuracy_rf_select_final = conf_rf_select_final$overall[["Accuracy"]]
Sensitivity_rf_select_final = conf_rf_select_final$byClass[["Sensitivity"]]
Specificity_rf_select_final = conf_rf_select_final$byClass[["Specificity"]]
Precision_rf_select_final = conf_rf_select_final$byClass[["Precision"]]
F1_rf_select_final = conf_rf_select_final$byClass[["F1"]]

## Binding confusion table final for RF - Final - Selected
Confusion_table_final <- bind_rows(Confusion_table_final,
      data_frame(method = "RF - Final - Selected",
      Accuracy =accuracy_rf_select_final,
      SensitivityorRecall = Sensitivity_rf_select_final,
      Specificity=Specificity_rf_select_final,Precision =
      Precision_rf_select_final, F1= F1_rf_select_final))

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

method	Accuracy	SensitivityorRecall	Specificity	Precision	F1
GLM - Final - All	0.8920831	0.6275992	0.9271225	0.5329053	0.5763889
GLM - Final - selected	0.8841221	0.6143667	0.9198598	0.5038760	0.5536627
RF - Final - All	0.9075630	0.4782609	0.9644378	0.6405063	0.5476190
RF - Final - Selected	0.9024768	0.3780718	0.9719509	0.6410256	0.4756243

The best accuracy(0.9075) came for the RandomForest model with all attributes. The second best accuracy(0.9024) came for the RandomForest model with selected attributes.

The best sensitivity or Recall(0.6275) came for the GLM model with all attributes. The second best sensitivity(0.6143) came for the GLM model with selected attributes.

The best specificity(0.9719) came for the RandomForest model with selected attributes. The second best specificity(0.9644) came for the RandomForest model with all attributes.

The best Precision(0.64102) came for the RandomForest model with selected attributes. The second best Precision (0.6405) came for the RandomForest model with all attributes.

The best F1 (0.5763) came for the GLM model with all attributes. The second best F1(0.5536) came for the GLM model with selected attributes.

4 Conclusion

We have built a machine learning algorithm with different models to predict whether a client will subscribe the term deposit or not. We ran through different algorithms like GLM, RPART, RandomForest and Knn with all attributes and selected attributes. Finally, we have concentrated on the GLM and RandonForest algorithms based on the initial testing results. GLM all attributes and selected attributes had good results on Sensitivity or Recall and F1 values whereas RandomForest all attributes and selected attributes had good values of Accuracy, Specificity and Precision. Based on the collective observations, we can conclude that RandonForest model is a better fit for this Portugues marketing dataset analysis. Instead of selecting 5 attributes, we can run all the algorithms with different sets of attributes in a series. But this will become a lengthy analysis and difficult to conclude in a better way. It is also time-consuming and does not give a desirable outcome.

5 Appendix - Environment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          6.3
## year           2020
## month          02
## day            29
## svn rev        77875
## language       R
## version.string R version 3.6.3 (2020-02-29)
## nickname       Holding the Windsock
```