Term Deposit Model Development

Wilmar Mangapot

2024-05-03

Project Details

This project demonstrates on the classification model using different machine learning techniques in R. The dataset used in this project contains information about a bank's marketing campaigns, and is commonly used for binary classification tasks to predict whether a customer will subscribe to a term deposit.

Data

The Bank Marketing dataset is available from the UCI Machine Learning Repository, which is a public repository for machine learning datasets. You can access the dataset and its information from the following link:

https://archive.ics.uci.edu/dataset/222/bank+marketing

Input variables:

1. age (numeric)
2. job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur", "student", "blue-collar","self-employed","retired","technician","services")
3. marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
4. education (categorical: "unknown","secondary","primary","tertiary")
5. default: has credit in default? (binary: "yes","no")
6. balance: average yearly balance, in euros (numeric)
7. housing: has housing loan? (binary: "yes","no")
8. loan: has personal loan? (binary: "yes","no") # related with the last contact of the current campaign:
9. contact: contact communication type (categorical: "unknown","telephone","cellular")
10. day: last contact day of the month (numeric)
11. month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12. duration: last contact duration, in seconds (numeric) # other attributes:
13. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15. previous: number of contacts performed before this campaign and for this client (numeric)
16. poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

    Output variable (desired target):

17. y - has the client subscribed a term deposit? (binary: "yes","no")

Libraries

```
library(easypackages)
libraries("readr","tidyverse","ggplot2","dplyr","rpart","caret","class",
          "KernSmooth","ROSE","rpart.plot","randomForest","MASS","pROC",
          "car")
```

Dataset

```
bank_df <- read_delim("C:/Users/wpman/Desktop/GROWSARI/bank-full.csv",
                      delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

EXPLANATORY DATA ANALYSIS

```
# Making a copy of the data set for Explanatory Data Analysis
bank <- bank_df

# Checking the dimensions of the dataset
dim(bank)
```

```
## [1] 45211    17
```

```
# Checking the contents of the dataset
head(bank)
```

```
## # A tibble: 6 x 17
##     age job        marital education default balance housing loan  contact   day
##   <dbl> <chr>      <chr>   <chr>     <chr>     <dbl> <chr>   <chr> <chr>   <dbl>
## 1    58 management married tertiary  no         2143 yes     no    unknown     5
## 2    44 technician single  secondary no           29 yes     no    unknown     5
## 3    33 entrepren~ married secondary no            2 yes     yes   unknown     5
## 4    47 blue-coll~ married unknown   no         1506 yes     no    unknown     5
## 5    33 unknown    single  unknown   no            1 no      no    unknown     5
## 6    35 management married tertiary  no          231 yes     no    unknown     5
## # i 7 more variables: month <chr>, duration <dbl>, campaign <dbl>, pdays <dbl>,
## #   previous <dbl>, poutcome <chr>, y <chr>
```

The dataset has 45,211 records with 17 columns.

```
str(bank) # To check the data types of each variables in the dataset
```

```
## spc_tbl_ [45,211 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age      : num [1:45211] 58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : chr [1:45211] "management" "technician" "entrepreneur" "blue-collar" ...
##  $ marital  : chr [1:45211] "married" "single" "married" "married" ...
##  $ education: chr [1:45211] "tertiary" "secondary" "secondary" "unknown" ...
##  $ default  : chr [1:45211] "no" "no" "no" "no" ...
##  $ balance  : num [1:45211] 2143 29 2 1506 1 ...
##  $ housing  : chr [1:45211] "yes" "yes" "yes" "yes" ...
##  $ loan     : chr [1:45211] "no" "no" "yes" "no" ...
##  $ contact  : chr [1:45211] "unknown" "unknown" "unknown" "unknown" ...
##  $ day      : num [1:45211] 5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : chr [1:45211] "may" "may" "may" "may" ...
##  $ duration : num [1:45211] 261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : num [1:45211] 1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : num [1:45211] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : num [1:45211] 0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : chr [1:45211] "unknown" "unknown" "unknown" "unknown" ...
##  $ y        : chr [1:45211] "no" "no" "no" "no" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   job = col_character(),
##   ..   marital = col_character(),
##   ..   education = col_character(),
##   ..   default = col_character(),
##   ..   balance = col_double(),
##   ..   housing = col_character(),
##   ..   loan = col_character(),
##   ..   contact = col_character(),
##   ..   day = col_double(),
##   ..   month = col_character(),
##   ..   duration = col_double(),
##   ..   campaign = col_double(),
##   ..   pdays = col_double(),
##   ..   previous = col_double(),
##   ..   poutcome = col_character(),
##   ..   y = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

In R, converting categorical variables to factors is very important because it enables R to recognize the variable as a categorical variable, rather than a numerical variable. This is important because categorical variables have unique properties that require special treatment when performing statistical analyses or machine learning algorithms.

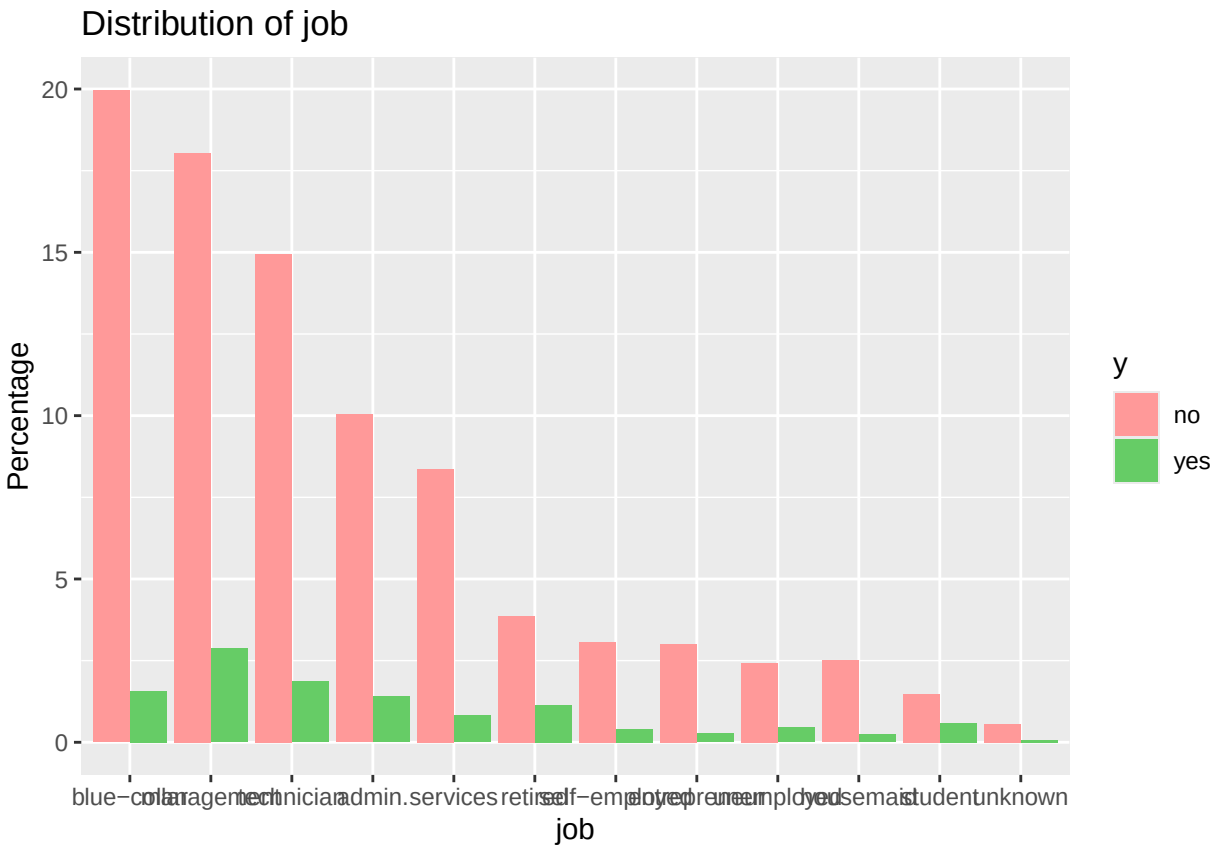
Categorical Variables

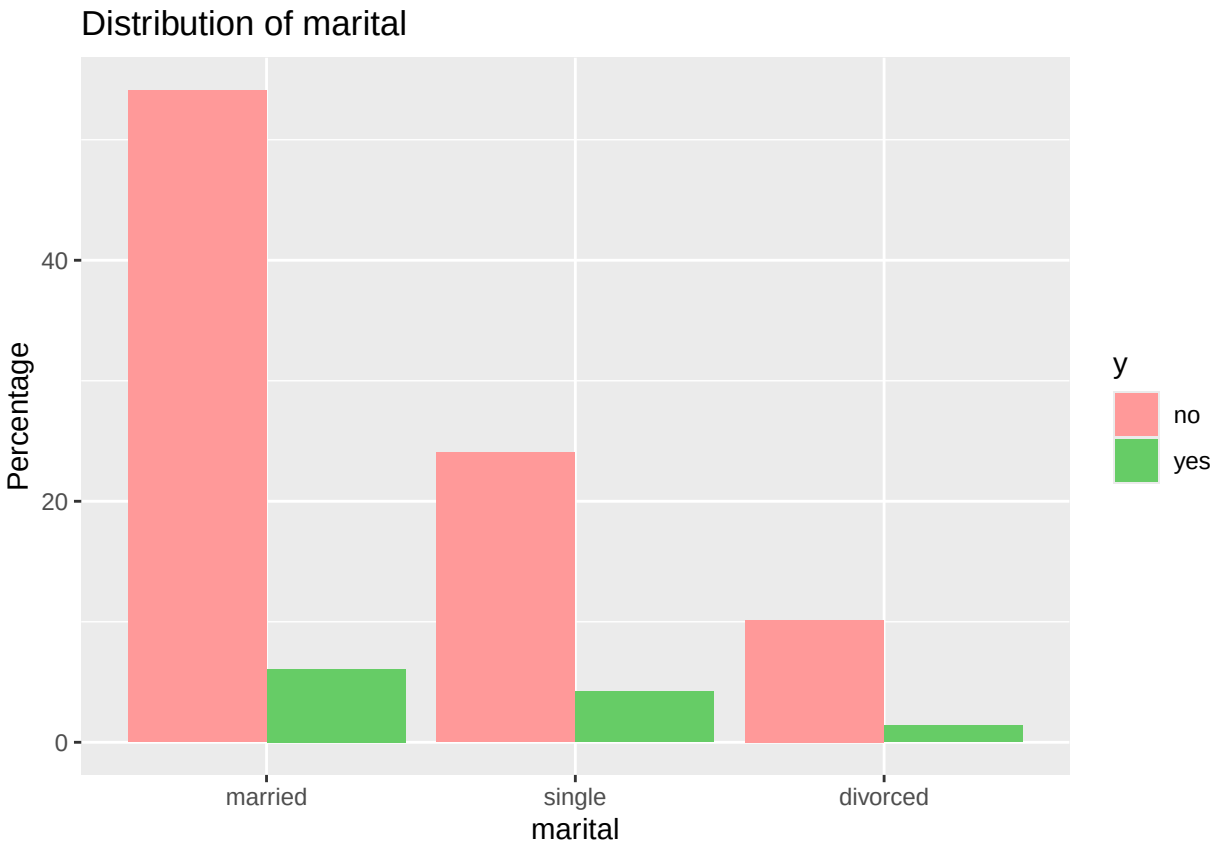
```r
# List of columns to convert to factors
cat_vars <- c("job", "marital", "education", "default", "housing", "loan",
              "contact", "month", "poutcome","y")

convert_to_factors <- function(data, columns) {
  for (column in columns) {
    data[[column]] <- as.factor(data[[column]])
  }
  return(data)
}

# Convert specified columns to factors
bank <- convert_to_factors(bank, cat_vars)

# Find the categorical variables
cat_vars <- sapply(bank, is.factor)

# Find the unique values of each categorical variable
unique_vals <- lapply(bank[, cat_vars], unique)
print(unique_vals)
```

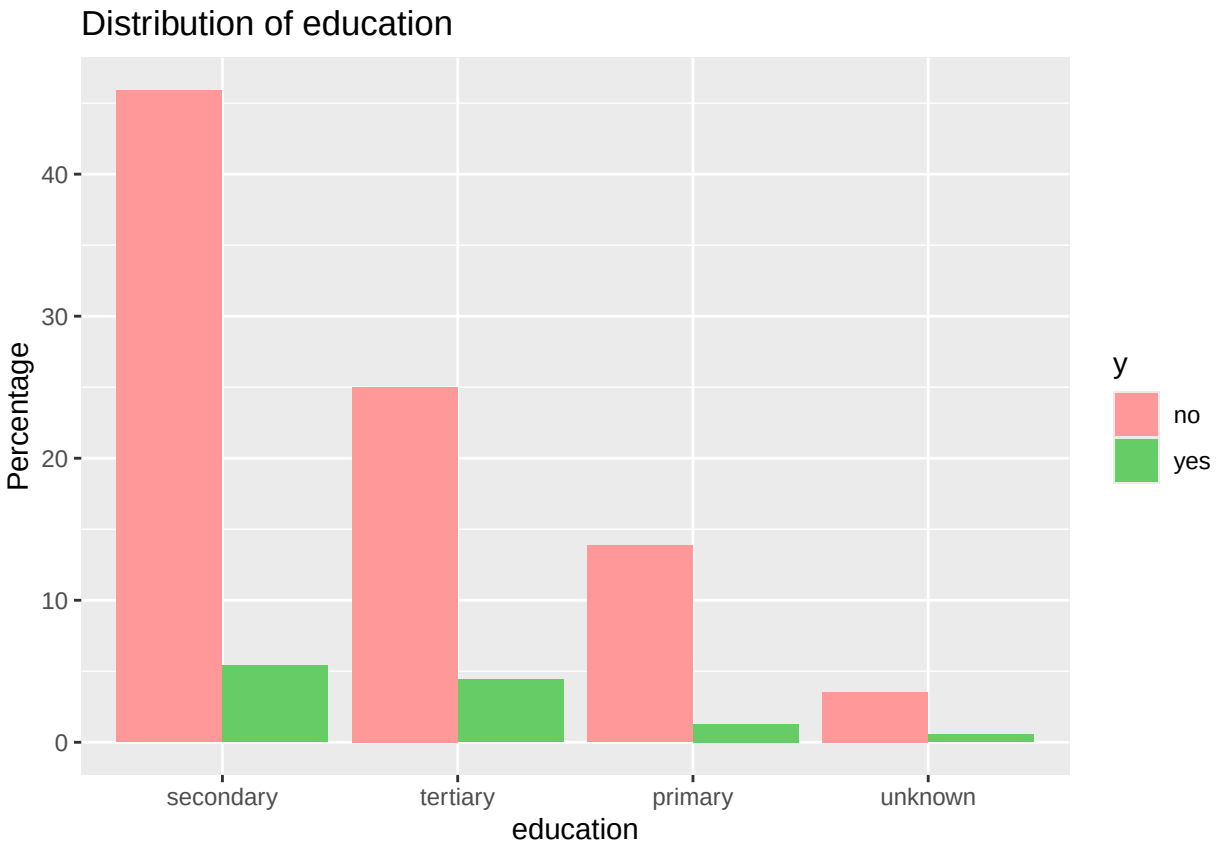
```
## $job
##  [1] management    technician    entrepreneur  blue-collar   unknown
##  [6] retired       admin.        services      self-employed unemployed
## [11] housemaid     student
## 12 Levels: admin. blue-collar entrepreneur housemaid management ... unknown
##
## $marital
## [1] married  single   divorced
## Levels: divorced married single
##
## $education
## [1] tertiary  secondary unknown   primary
## Levels: primary secondary tertiary unknown
##
## $default
## [1] no  yes
## Levels: no yes
##
## $housing
## [1] yes no
## Levels: no yes
##
## $loan
## [1] no  yes
## Levels: no yes
##
## $contact
## [1] unknown   cellular  telephone
## Levels: cellular telephone unknown
##
## $month
##  [1] may jun jul aug oct nov dec jan feb mar apr sep
```

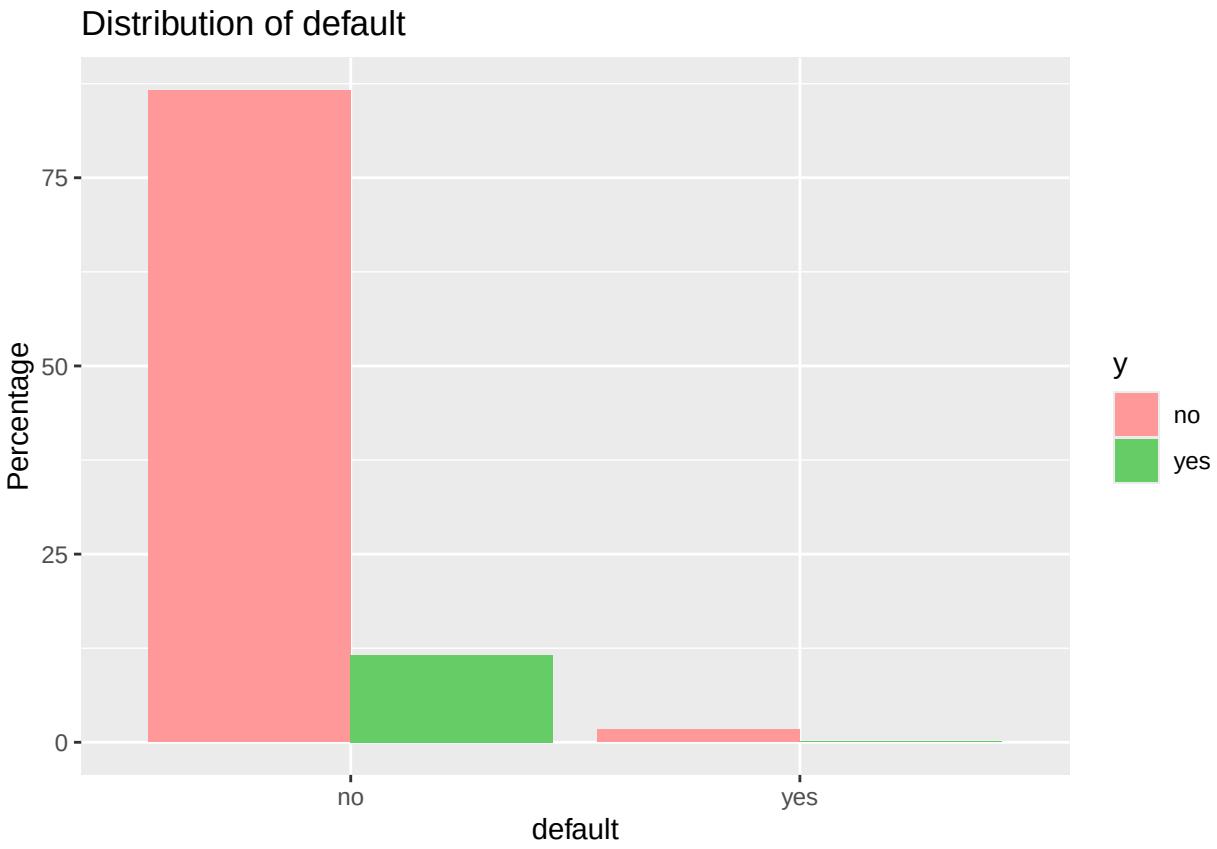
```
## Levels: apr aug dec feb jan jul jun mar may nov oct sep
##
## $poutcome
## [1] unknown failure other   success
## Levels: failure other success unknown
##
## $y
## [1] no  yes
## Levels: no yes
```

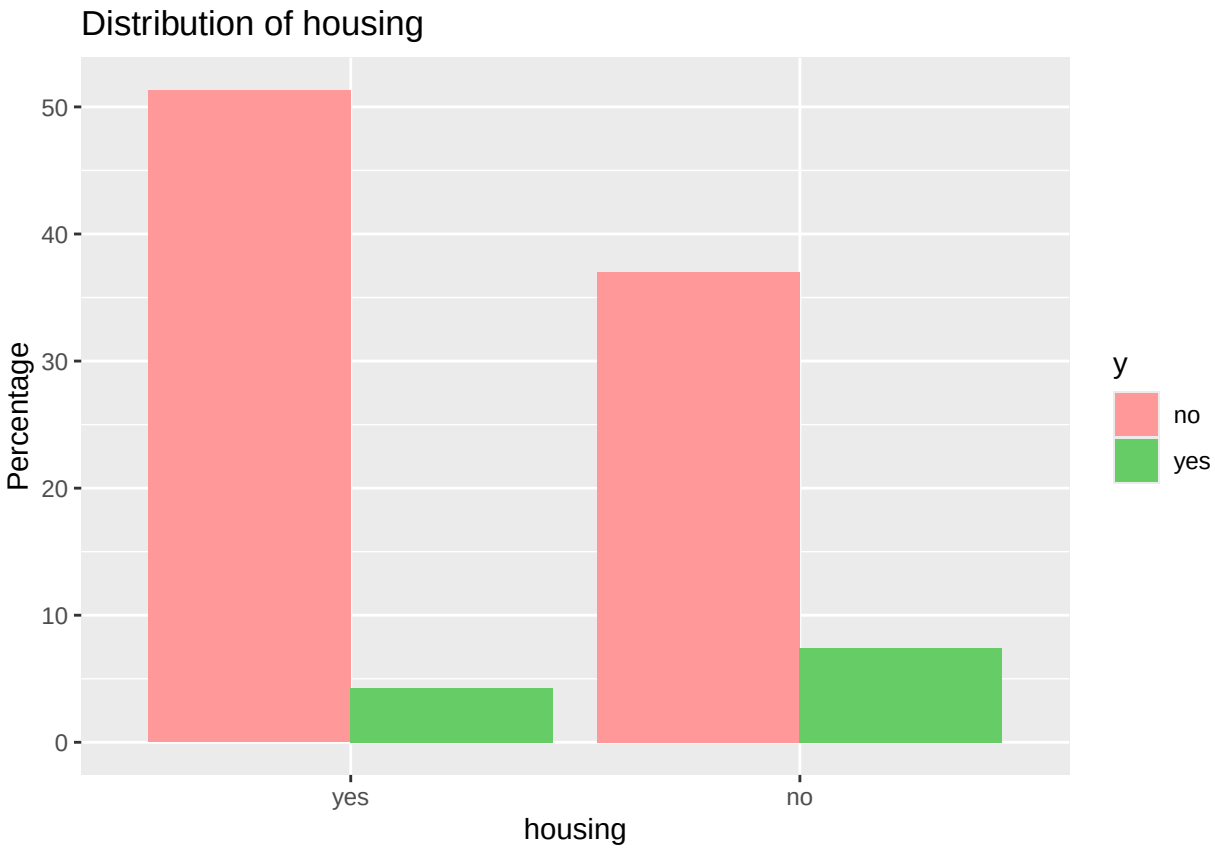
```
cat_vars <- c("job", "marital", "education", "default", "housing", "loan",
              "contact", "month", "poutcome")

# Create a function to plot bar charts for categorical variables
plot_cat_var <- function(data, var_name) {

  # Prepare data for plotting
  freq_table <- data %>%
    group_by(.data[[var_name]], y) %>%
    summarise(freq = n(), .groups = 'drop') %>%
    mutate(percentage = (freq / sum(freq)) * 100)

  # Create plot
  p <- ggplot(freq_table, aes(x = reorder(.data[[var_name]], -freq), y = percentage, fill = y)) +
    geom_bar(stat = "identity", position = "dodge") +
    scale_fill_manual(values = c("#FF9999", "#66CC66")) +
    labs(title = paste("Distribution of", var_name), x = var_name, y = "Percentage")

  # Return the plot object
  return(p)
}

# Plot bar charts for all categorical variables using a loop
for (var in cat_vars) {
  print(plot_cat_var(bank, var))  # Explicitly print each plot
}
```

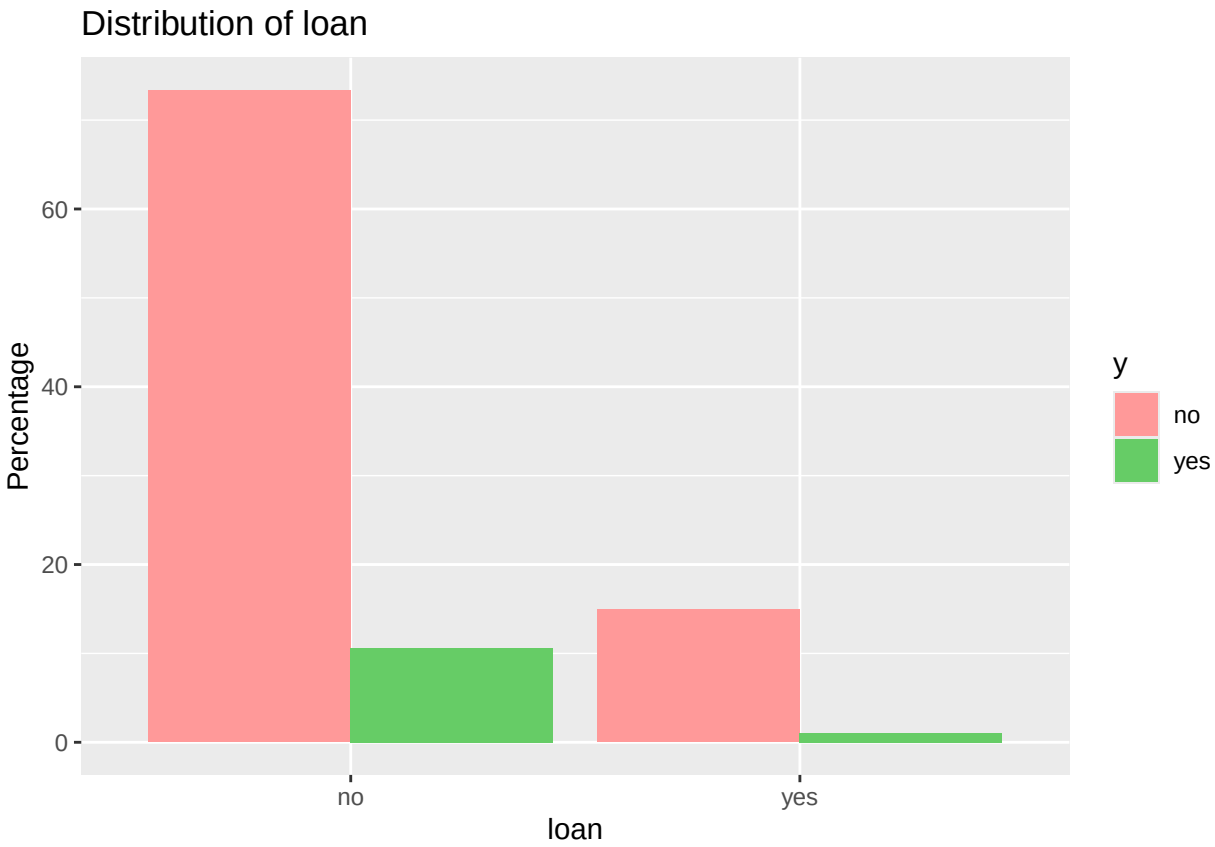

Distribution of job
Percentage
20
15
10
5
0
blue−collar
management
technician
admin.
services
retired
self−employed
entrepreneur
unemployed
housemaid
student
unknown
job
y
no
yes

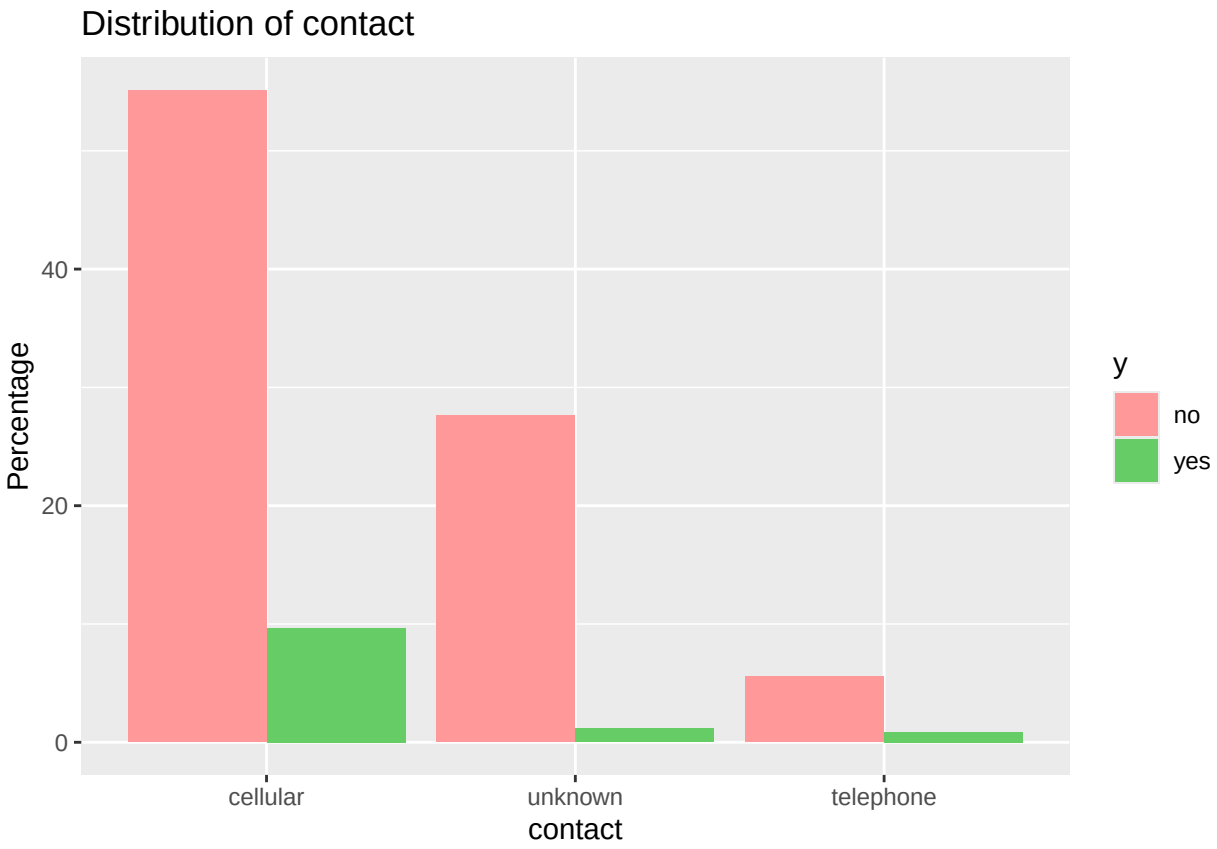

Distribution of marital
Percentage
40
20
0
married
single
divorced
marital
y
no
yes

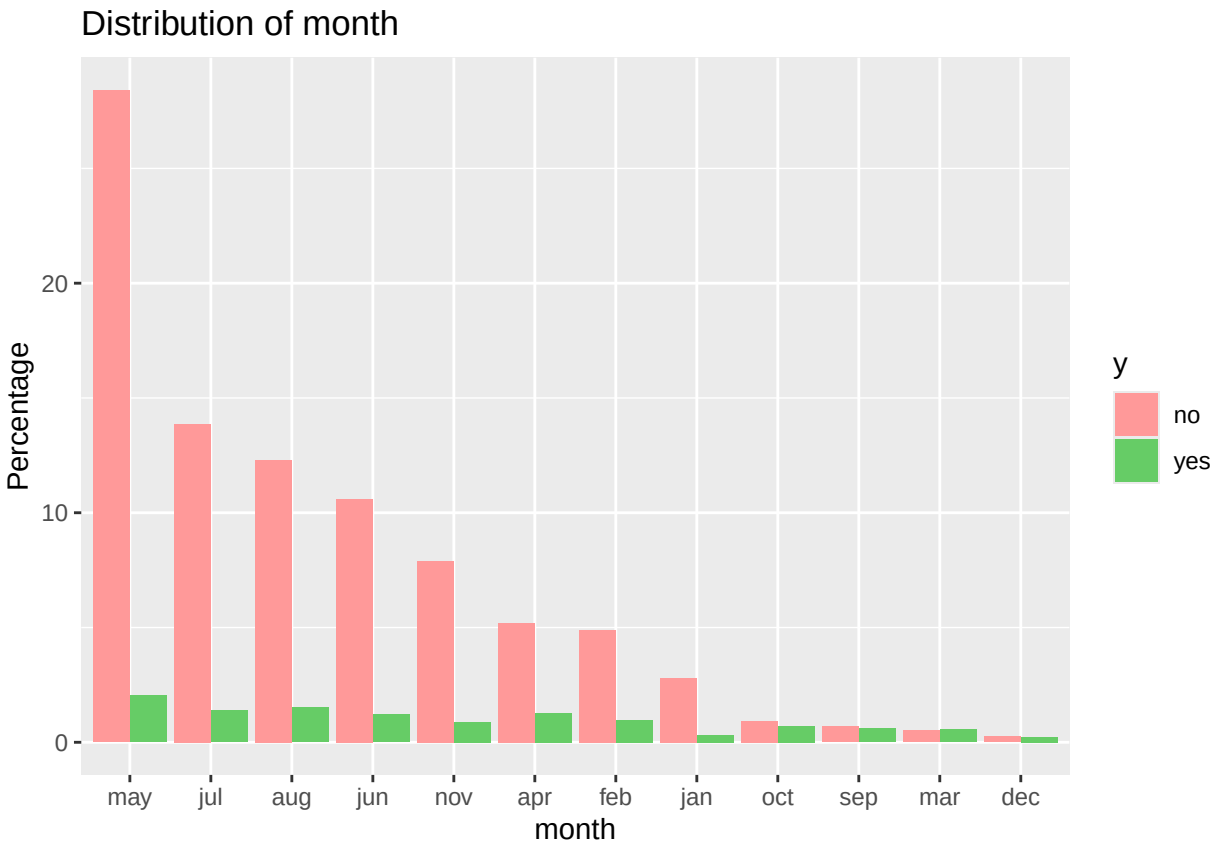

Distribution of education
Percentage
40
30
20
10
0
secondary
tertiary
primary
unknown
education
y
no
yes

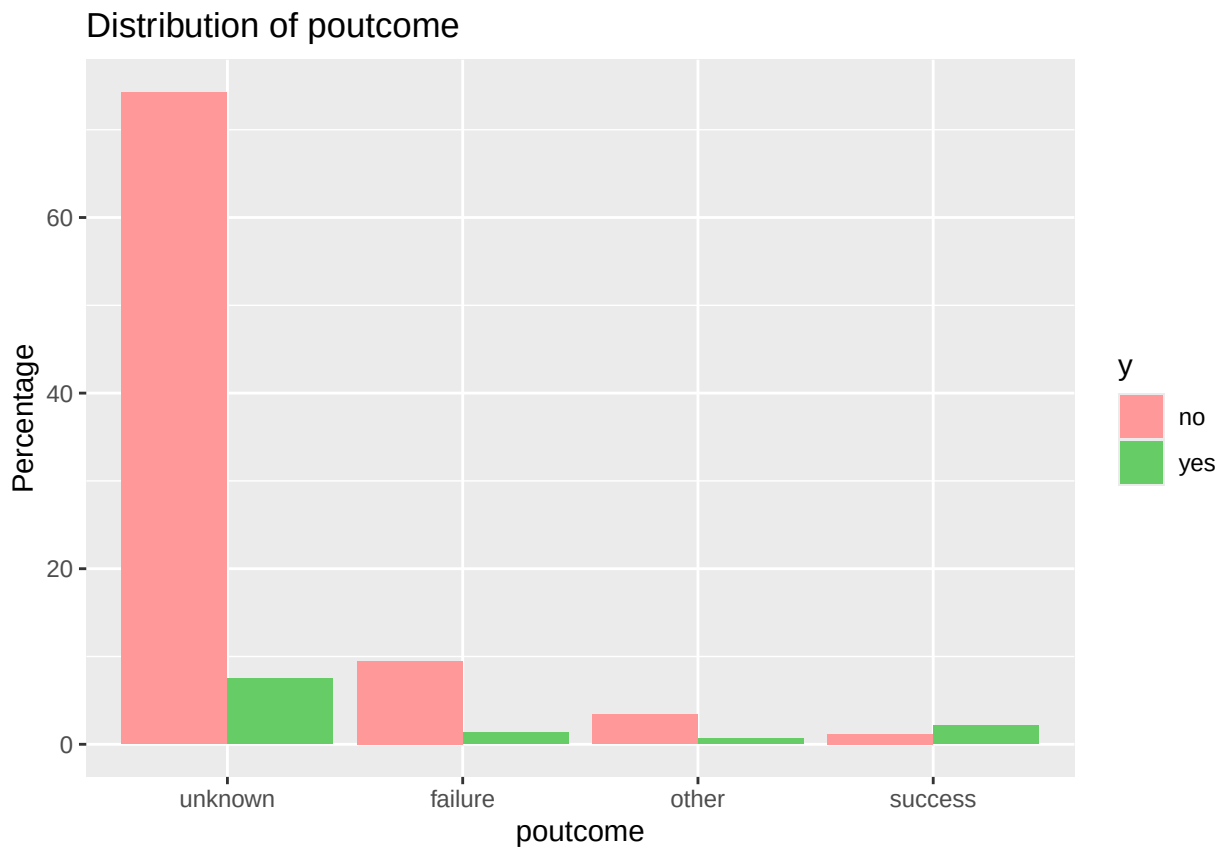

Distribution of default
Percentage
75
50
25
0
no
yes
default
y
no
yes


Distribution of housing
Percentage
50
40
30
20
10
0
yes
no
housing
y
no
yes


Distribution of loan
Percentage
60
40
20
0
no
yes
loan
y
no
yes


Distribution of contact
Percentage
40
20
0
cellular
unknown
telephone
contact
y
no
yes


Distribution of month
Percentage
20
10
0
may
jul
aug
jun
nov
apr
feb
jan
oct
sep
mar
dec
month
y
no
yes



Here we are the few other observations made.

Job: Maximum number of clients work in Blue-color job.

Maritial: Maximum number of clients are married. Minimum number of clients are divorced.

Education: Maximum number of clients have completed Secondary education.

Default: Almost all clients in the dataset have no credit default.

Loan: Maximum number of clients do not have a housing.

Loan: Maximum number of clients do not have a personal loan.

Contact: Maximum number of clients has been contacted through cellular.

Month: Most number of contacts were carried out in the month of May. Least number of contacts were carried out in the month of December.

Poutcome: For maximum number of clients outcome of previous marketing campaign is unknown. Number of failures are higher when compared to success in the results of previous marketing campaign.

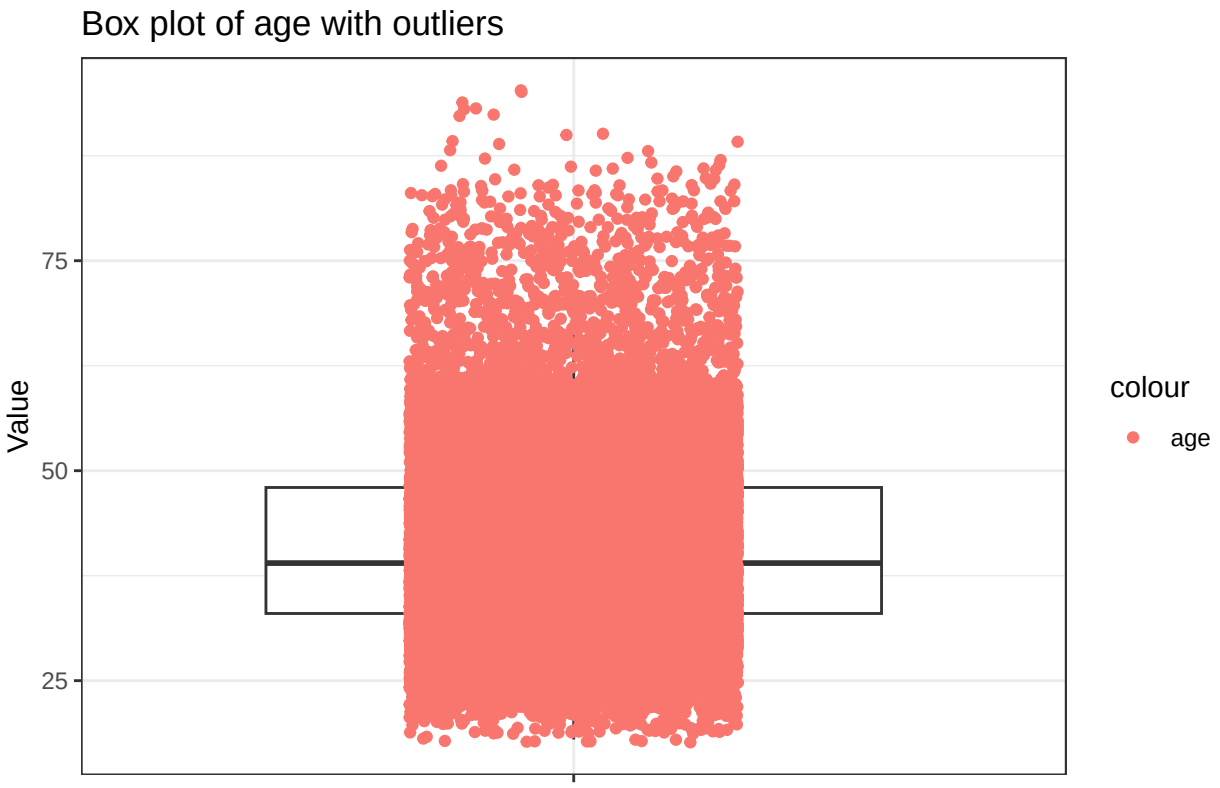
Numerical Variables

```
# Display summary statistics for numeric variables
num_vars <- c("age", "balance", "day", "duration", "campaign", "pdays", "previous")
summary(bank[,num_vars])
```

```
##       age            balance             day           duration
##  Min.   :18.00   Min.   : -8019   Min.   : 1.00   Min.   :   0.0
##  1st Qu.:33.00   1st Qu.:    72   1st Qu.: 8.00   1st Qu.: 103.0
##  Median :39.00   Median :   448   Median :16.00   Median : 180.0
##  Mean   :40.94   Mean   :  1362   Mean   :15.81   Mean   : 258.2
##  3rd Qu.:48.00   3rd Qu.:  1428   3rd Qu.:21.00   3rd Qu.: 319.0
##  Max.   :95.00   Max.   :102127   Max.   :31.00   Max.   :4918.0
##     campaign          pdays          previous
##  Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000
##  1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000
##  Median : 2.000   Median : -1.0   Median :  0.0000
##  Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
##  3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :63.000   Max.   :871.0   Max.   :275.0000
```

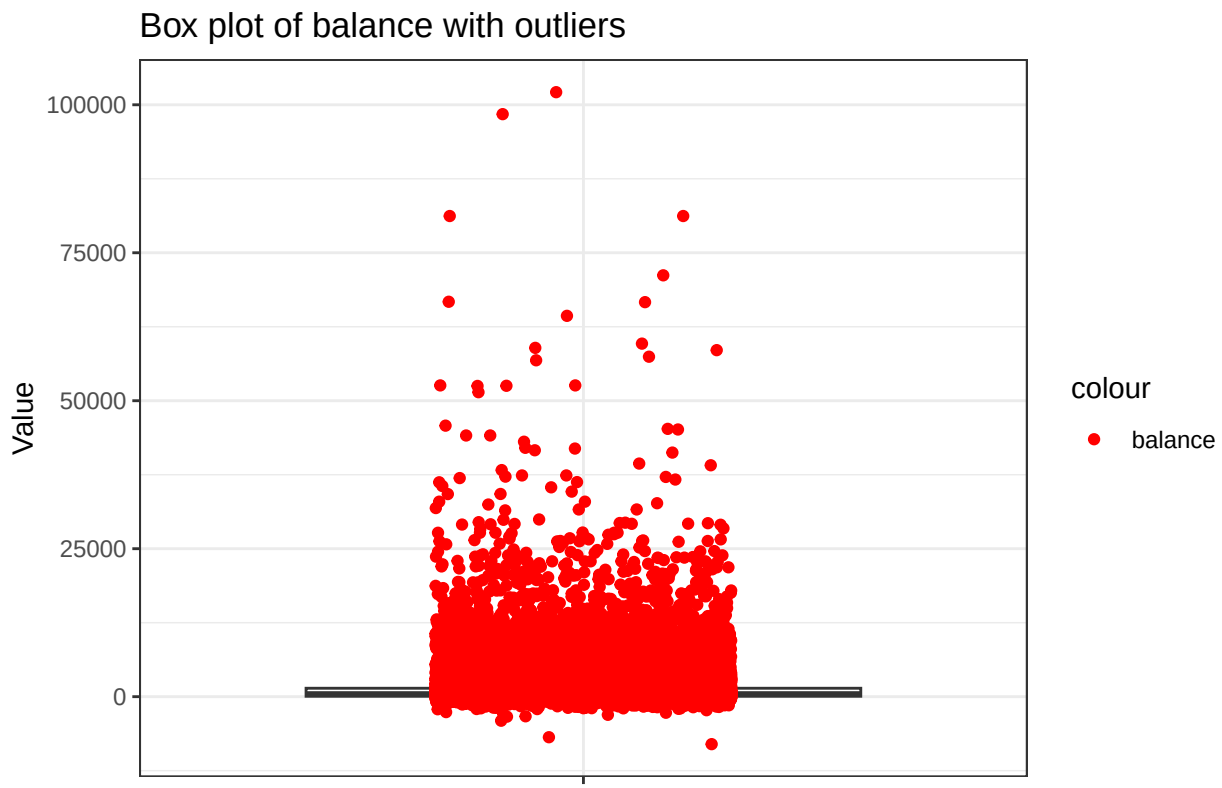
From the above statistical summary, the minimum age in the dataset is 18 years old while the maximum age is 95 years old. Lets see the distribution of it using box plot.

```
# Check for the outliers in the numerical variables in the dataset
# Create a box plot of age
ggplot(data = bank, aes(x = "", y = age)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(aes(color = "age"), width = 0.2) +
  labs(x = "", y = "Value") +
  ggtitle("Box plot of age with outliers") +
  theme_bw()
```

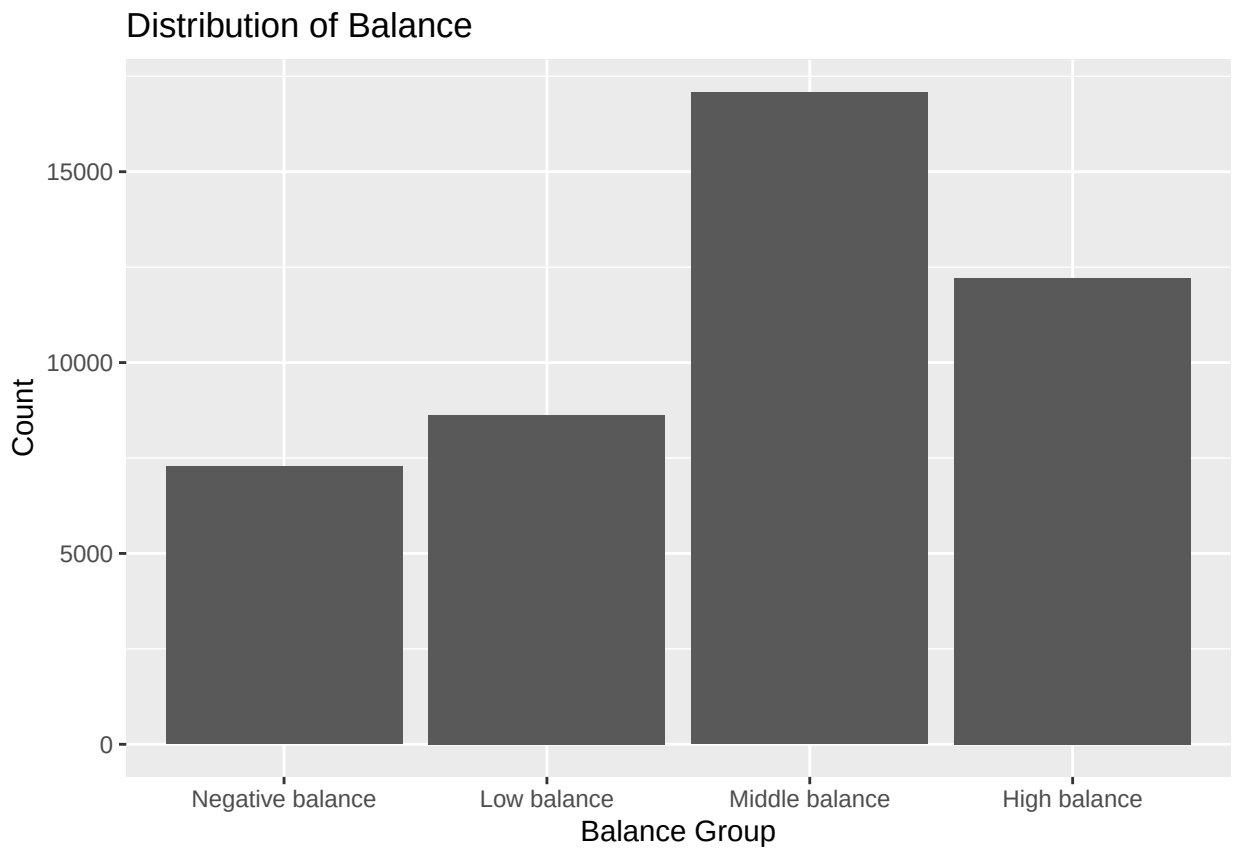


Age has outliers.

```
#Create a box plot of balance with outliers
ggplot(data = bank, aes(x = "", y = balance)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(aes(color = "balance"), width = 0.2) +
  labs(x = "", y = "Value") +
  scale_color_manual(values = c("red", "blue")) +
  ggtitle("Box plot of balance with outliers") +
  theme_bw()
```



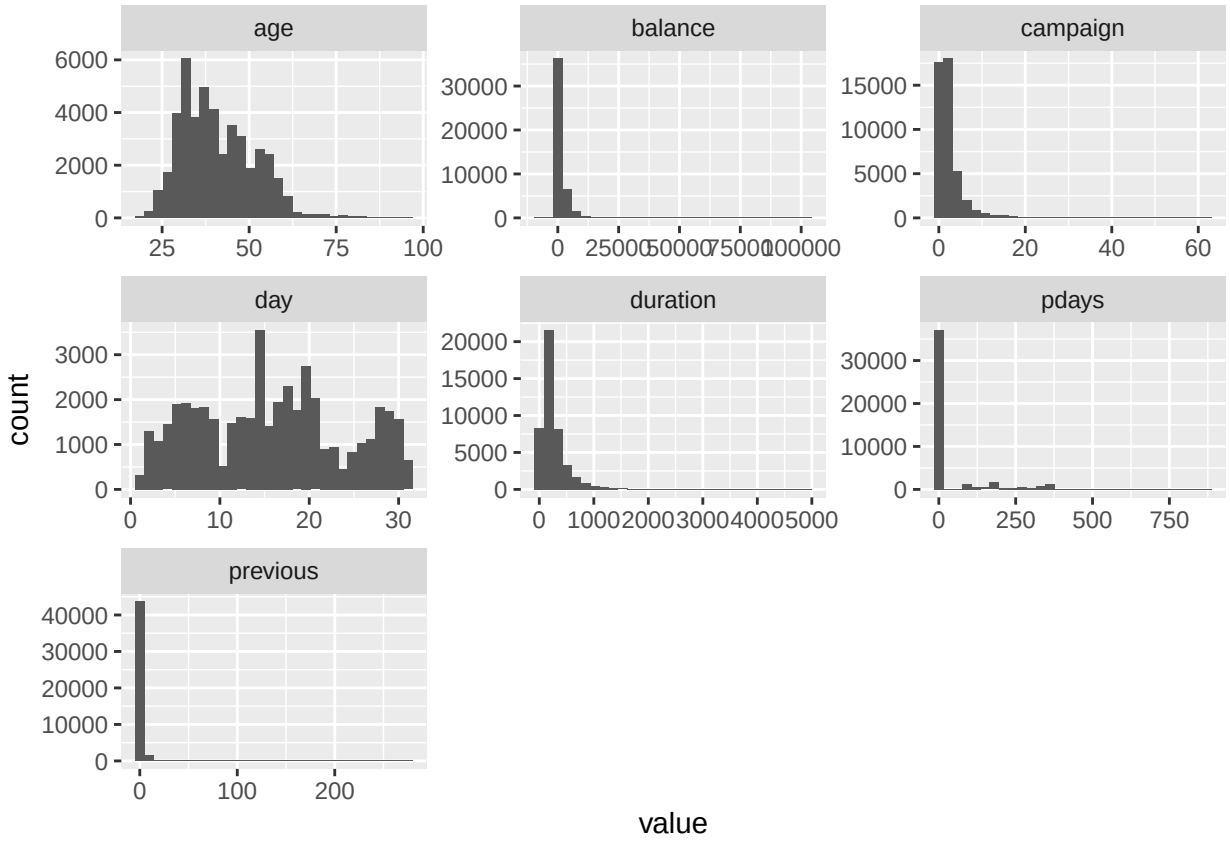

```
# Create bins for different balance levels
bins <- c(-Inf, 0, 200, 1300, Inf)
labels <- c("Negative balance", "Low balance", "Middle balance", "High balance")

# Group the balance values into the bins and add labels
bank_group <- bank
bank_group$balance_group <- cut(bank$balance, breaks = bins, labels = labels)

# Create a bar graph of balance by balance_group
ggplot(bank_group, aes(x = balance_group)) +
  geom_bar() +
  ggtitle("Distribution of Balance") +
  xlab("Balance Group") +
  ylab("Count")
```



From the above summary statistics of balance data, its distribution of bins and box plot, we can infer that the balance variable has outliers. The negative balance here is assumed to be allowed for term deposits and hence not removed. More than 7000 customers hold negative balance and more than 12000 customers hold high balance greater than 1500 euros, and rest hold average balance between 500 t0 1500 euros.

Lets see the distribution of all other numeric variables.

```
# select numeric columns
bank_numeric <- bank %>%
  select_if(is.numeric)

# gather data into long format for plotting
bank_gathered <- bank_numeric %>%
  gather()

# plot the distribution of numeric variables
ggplot(bank_gathered, aes(x=value)) +
  geom_histogram(bins=30) +
  facet_wrap(~key, scales="free")
```




```
# Check for missing values
sum(is.na(bank))
```

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

Fortunately, there are no missing values. If there were missing values we will have to decide whether to remove them or do imputation technique.

Before we proceed to the model development, we need to check the distribution of our dependent variable "y"

```
# Check for class imbalance
ggplot(bank, aes(x = y, fill = y)) +
  geom_bar() +
  ggtitle("Distribution of Target Variable") +
  xlab("Target Variable") +
  ylab("Count")
```

According ot the box plot, the dataset is highly imbalance in terms of the dependent variable. Understanding the distribution of the target variable can help guide the development of machine learning models. In particular, it can help determine the appropriate evaluation metrics to use (e.g. accuracy, precision, recall, F1-score) and the techniques that can be used to handle class imbalance (e.g. resampling methods, cost-sensitive learning, ensemble methods).

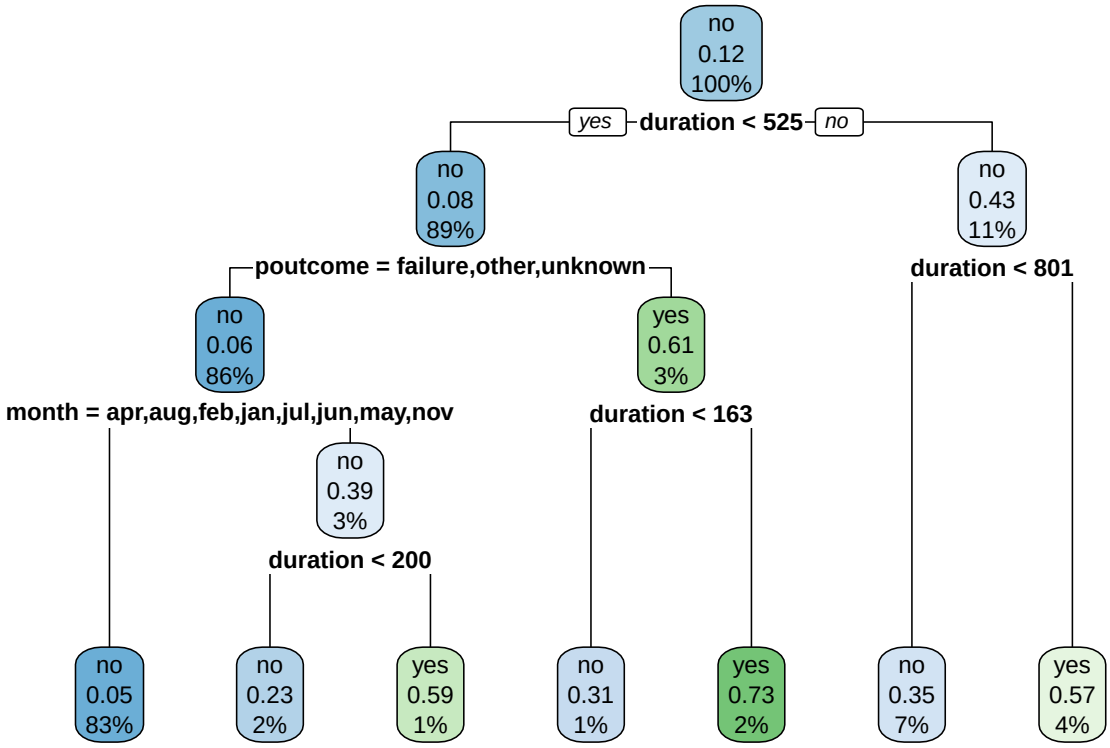
MODEL DEVELOPMENT

Classification using Decision Tree

Training and Test data set

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

```
tree_model <- rpart(y ~ ., data = training, method = "class") # Training the model
rpart.plot(tree_model) # Plotting the model
```




```
tree_model
```

```
## n= 31649
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 31649 3703 no (0.88299788 0.11700212)
##    2) duration< 524.5 28199 2207 no (0.92173481 0.07826519)
##      4) poutcome=failure,other,unknown 27291 1651 no (0.93950387 0.06049613)
##        8) month=apr,aug,feb,jan,jul,jun,may,nov 26271 1255 no (0.95222869 0.04777131) *
##        9) month=dec,mar,oct,sep 1020  396 no (0.61176471 0.38823529)
##         18) duration< 199.5 575  132 no (0.77043478 0.22956522) *
##         19) duration>=199.5 445  181 yes (0.40674157 0.59325843) *
##      5) poutcome=success 908  352 yes (0.38766520 0.61233480)
##       10) duration< 162.5 256   79 no (0.69140625 0.30859375) *
##       11) duration>=162.5 652  175 yes (0.26840491 0.73159509) *
##    3) duration>=524.5 3450 1496 no (0.56637681 0.43362319)
##      6) duration< 800.5 2093  723 no (0.65456283 0.34543717) *
##      7) duration>=800.5 1357  584 yes (0.43036109 0.56963891) *
```

After training the model, we plot the decision tree using the plot function. This produces a graphical representation of the tree that can help us interpret the model.

Model Prediction

```r
# Make predictions on the testing set
predictions <- predict(tree_model, newdata = testing, type = "class")
```

```r
# Evaluate the model
conf_matrix <- confusionMatrix(predictions, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11574   929
##        yes   402   657
##
##                Accuracy : 0.9019
##                  95% CI : (0.8967, 0.9068)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 1.619e-12
##
##                   Kappa : 0.4448
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9664
##             Specificity : 0.4142
##          Pos Pred Value : 0.9257
##          Neg Pred Value : 0.6204
##              Prevalence : 0.8831
##          Detection Rate : 0.8534
##    Detection Prevalence : 0.9219
##       Balanced Accuracy : 0.6903
##
##        'Positive' Class : no
##
```

```r
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.902
```

```r
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.966
```

```r
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.414
```

```r
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.926
```

```r
predictions <- predict(tree_model, newdata = testing, type = "prob")[,2]

# Calculate AUC for Decision Tree
tree_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```r
print(paste("AUC for Decision Tree:", round(auc(tree_auc),3)))
```

```
## [1] "AUC for Decision Tree: 0.803"
```

Classification using Decision Tree with SMOTE

Training and Test data set

```r
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

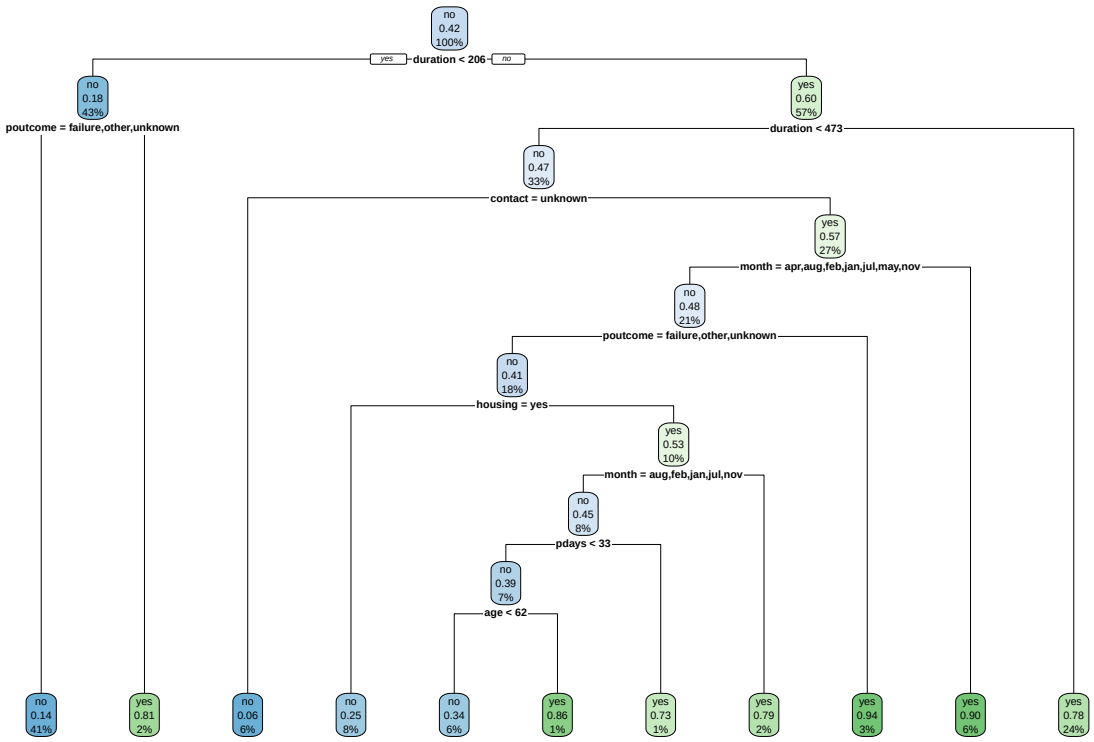
To preserve the accuracy of the testing data set, we will only apply SMOTE to the training data set

```r
training_balance <- ovun.sample(y ~ ., data = training, method = "over", N = 48000)$data
prop.table(table(training_balance$y))
```

```
##
##        no       yes
## 0.5822083 0.4177917
```

After using SMOTE, the distribution of the dependent variable are now became stable. Now, we will build the model again.

```r
tree_model <- rpart(y ~ ., data = training_balance, method = "class") # Training the model
rpart.plot(tree_model) # Plotting the model
```




```
tree_model
```

```
## n= 48000
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 48000 20054 no (0.58220833 0.41779167)
##     2) duration< 205.5 20766  3647 no (0.82437638 0.17562362)
##       4) poutcome=failure,other,unknown 19596  2699 no (0.86226781 0.13773219) *
##       5) poutcome=success 1170   222 yes (0.18974359 0.81025641) *
##     3) duration>=205.5 27234 10827 yes (0.39755453 0.60244547)
##       6) duration< 472.5 15799  7473 no (0.52699538 0.47300462)
##        12) contact=unknown 2927   176 no (0.93987017 0.06012983) *
##        13) contact=cellular,telephone 12872  5575 yes (0.43311063 0.56688937)
##          26) month=apr,aug,feb,jan,jul,may,nov 10086  4794 no (0.52468769 0.47531231)
##            52) poutcome=failure,other,unknown 8805  3594 no (0.59182283 0.40817717)
##             104) housing=yes 3854   980 no (0.74571873 0.25428127) *
##             105) housing=no 4951  2337 yes (0.47202585 0.52797415)
##               210) month=aug,feb,jan,jul,nov 3772  1681 no (0.55434783 0.44565217)
##                 420) pdays< 32.5 3157  1229 no (0.61070637 0.38929363)
##                   840) age< 61.5 2873   986 no (0.65680473 0.34319527) *
##                   841) age>=61.5 284    41 yes (0.14436620 0.85563380) *
##                 421) pdays>=32.5 615   163 yes (0.26504065 0.73495935) *
##               211) month=apr,may 1179   246 yes (0.20865140 0.79134860) *
##            53) poutcome=success 1281    81 yes (0.06323185 0.93676815) *
```

```
##          27) month=dec,jun,mar,oct,sep 2786   283 yes (0.10157933 0.89842067) *
##       7) duration>=472.5 11435  2501 yes (0.21871447 0.78128553) *
```

Model Prediction

```
# Make predictions on the testing set
predictions <- predict(tree_model, newdata = testing, type = "class")
```

```
# Evaluate the model
conf_matrix <- confusionMatrix(predictions, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10515   369
##        yes  1461  1217
##
##                Accuracy : 0.8651
##                  95% CI : (0.8592, 0.8708)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4969
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8780
##             Specificity : 0.7673
##          Pos Pred Value : 0.9661
##          Neg Pred Value : 0.4544
##              Prevalence : 0.8831
##          Detection Rate : 0.7753
##    Detection Prevalence : 0.8025
##       Balanced Accuracy : 0.8227
##
##        'Positive' Class : no
##
```

```
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.865
```

```
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.878
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.767
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.966
```

```
predictions <- predict(tree_model, newdata = testing, type = "prob")[,2]

# Calculate AUC for Decision Tree
tree_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
print(paste("AUC for Decision Tree:", round(auc(tree_auc),3)))
```

```
## [1] "AUC for Decision Tree: 0.861"
```

Classification using Random Forest

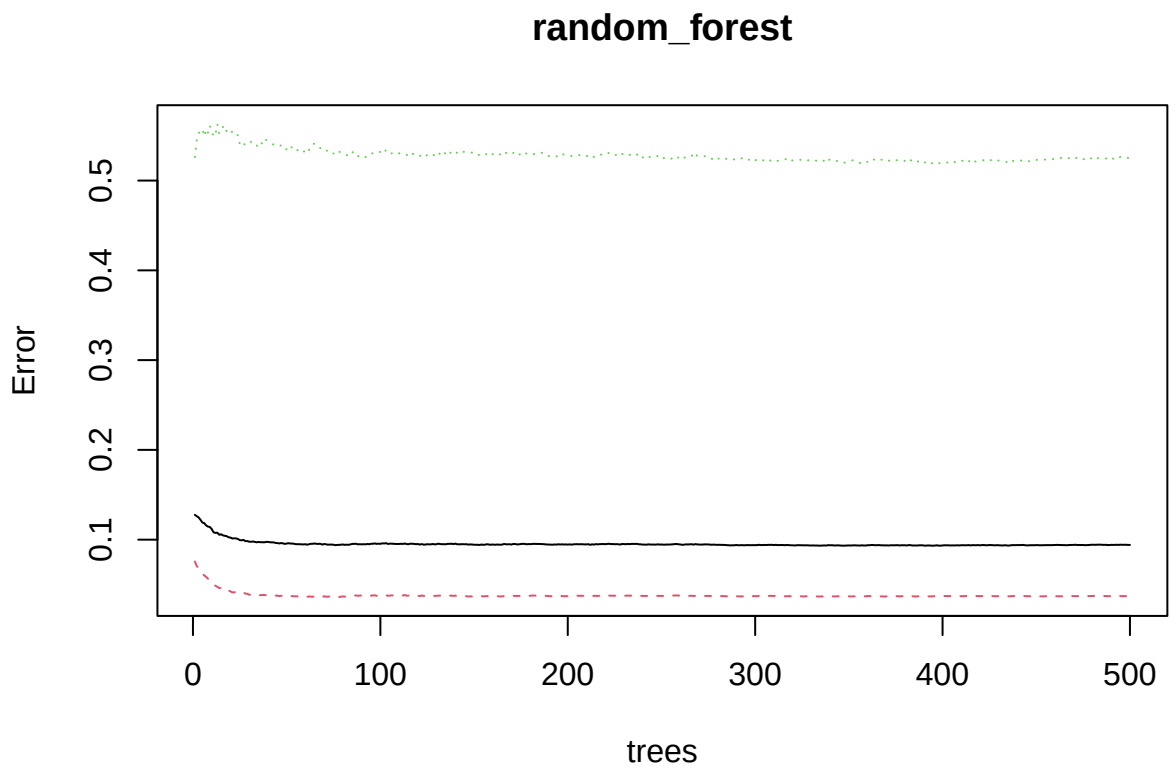
Training and Test data set

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

```
# Training the model
random_forest <- randomForest(y ~ .,data = training, ntree = 500)
random_forest
```

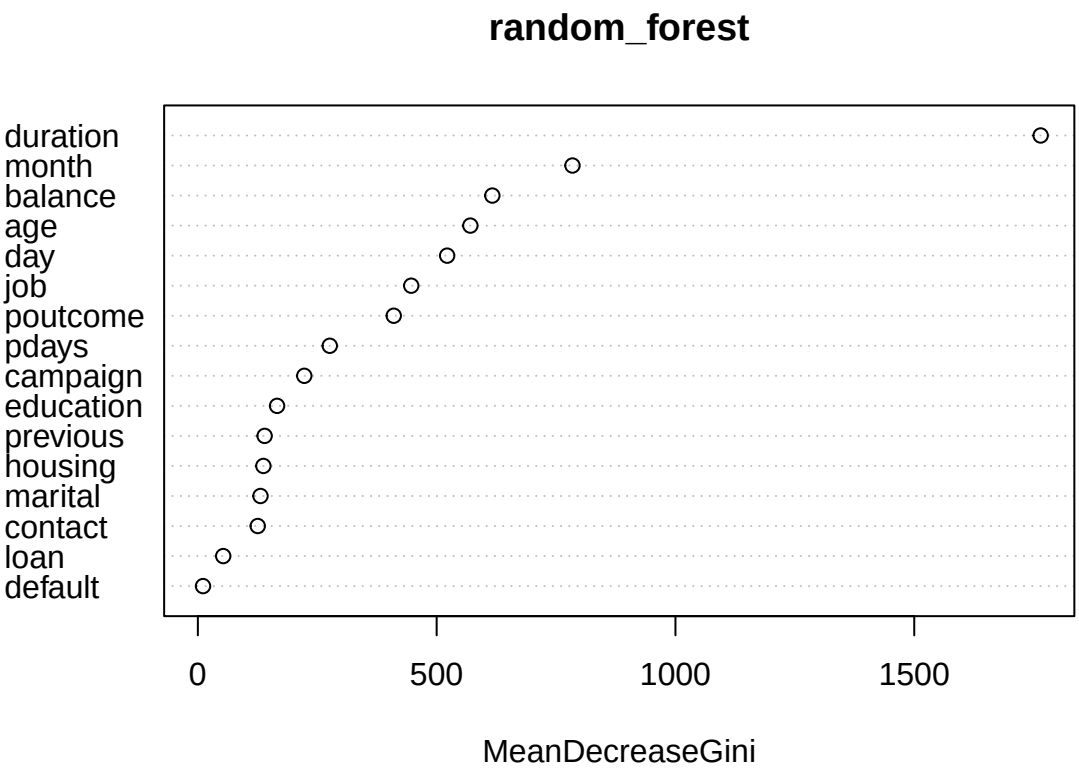
```
##
## Call:
##  randomForest(formula = y ~ ., data = training, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.41%
## Confusion matrix:
##        no  yes class.error
## no  26912 1034  0.03699993
## yes  1945 1758  0.52524980
```

```r
plot(random_forest) # Plotting the model
```



After training the model, we plot the random forest using the plot function. This produces a graphical representation of the forest that can help us interpret the model.

```r
# Variable importance plot
varImpPlot(random_forest)
```



Based on the Variance important plot, duration, month and balance has the highest importance in the model while contact, loan and default has the least importance.

Model Prediction

```
# Make predictions on the testing set
predictions <- predict(random_forest, newdata = testing, type = "class")
```

```
# Evaluate the model
conf_matrix <- confusionMatrix(predictions, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11581   807
##        yes   395   779
##
##                Accuracy : 0.9114
##                  95% CI : (0.9065, 0.9161)
##     No Information Rate : 0.8831
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5164
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9670
##             Specificity : 0.4912
##          Pos Pred Value : 0.9349
##          Neg Pred Value : 0.6635
##              Prevalence : 0.8831
##          Detection Rate : 0.8539
##    Detection Prevalence : 0.9134
##       Balanced Accuracy : 0.7291
##
##        'Positive' Class : no
##
```

```
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.911
```

```
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.967
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.491
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.935
```

```
predictions <- predict(random_forest, newdata = testing, type = "prob")[,2]

# Calculate AUC for Random Forest
forest_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
print(paste("AUC for Random Forest:", round(auc(forest_auc),3)))
```

```
## [1] "AUC for Random Forest: 0.935"
```

Classification using Random Forest with SMOTE

Training and Test data set

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

To preserve the accuracy of the testing data set, we will only apply SMOTE to the training data set

```
training_balance <- ovun.sample(y ~ ., data = training, method = "over", N = 48000)$data
prop.table(table(training_balance$y))
```

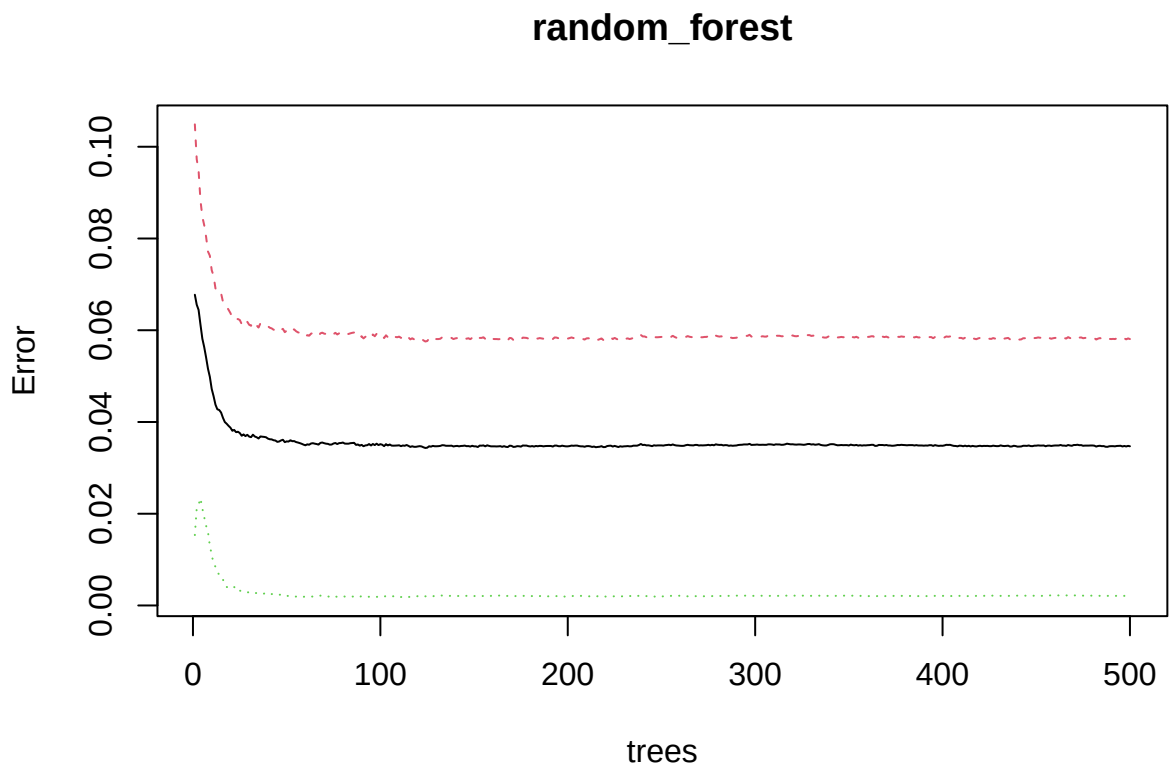
```
##
##        no       yes
## 0.5822083 0.4177917
```

After using SMOTE, the distribution of the dependent variable are now became stable. Now, we will build the model again.

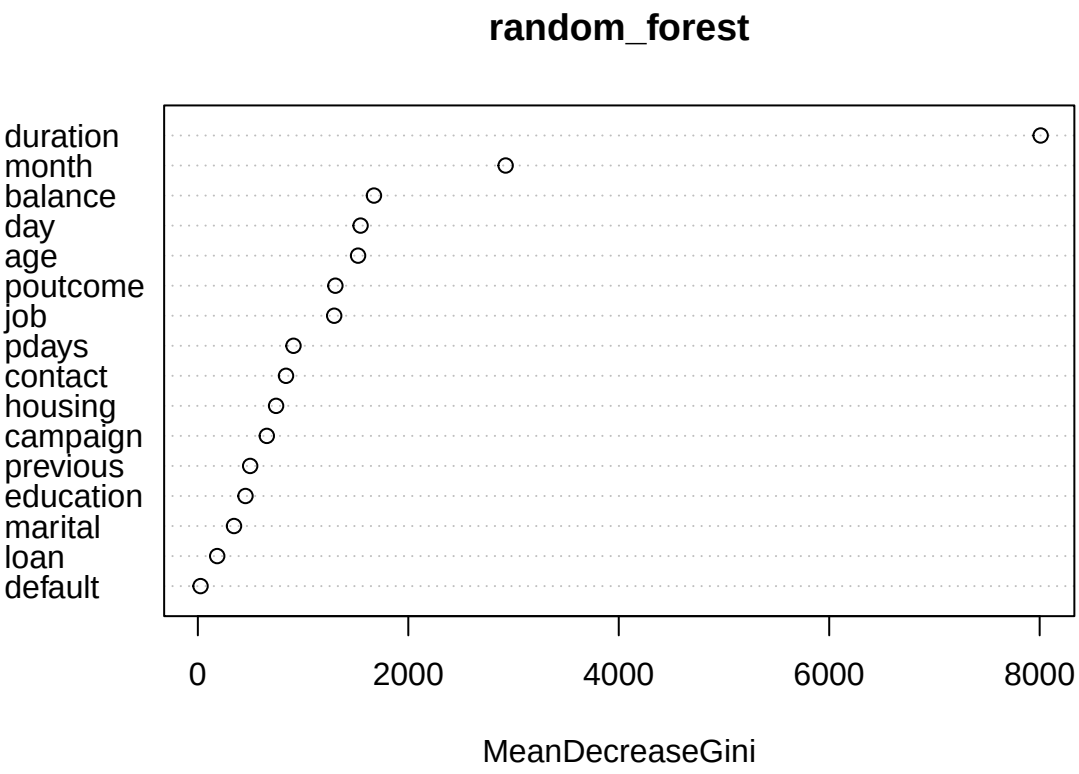
```
random_forest <- randomForest(y ~ .,data = training_balance, ntree = 500)
random_forest
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = training_balance, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 3.47%
## Confusion matrix:
##        no   yes class.error
## no  26323  1623 0.058076290
## yes    43 20011 0.002144211
```

```
plot(random_forest) # Plotting the model
```




```r
# Variable importance plot
varImpPlot(random_forest)
```



Based on the Variance important plot, duration, month and balance has the highest importance in the model while marital, loan and default has the least importance.

Model Prediction

```
# Make predictions on the testing set
predictions <- predict(random_forest, newdata = testing, type = "class")
```

```
# Evaluate the model
conf_matrix <- confusionMatrix(predictions, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11360   594
##        yes   616   992
##
##                Accuracy : 0.9108
##                  95% CI : (0.9059, 0.9155)
##     No Information Rate : 0.8831
```

```
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5706
##
##  Mcnemar's Test P-Value : 0.546
##
##             Sensitivity : 0.9486
##             Specificity : 0.6255
##          Pos Pred Value : 0.9503
##          Neg Pred Value : 0.6169
##              Prevalence : 0.8831
##          Detection Rate : 0.8376
##    Detection Prevalence : 0.8814
##       Balanced Accuracy : 0.7870
##
##        'Positive' Class : no
##
```

```
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.911
```

```
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.949
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.625
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.95
```

```
predictions <- predict(random_forest, newdata = testing, type = "prob")[,2]

# Calculate AUC for Random Forest
forest_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
print(paste("AUC for Random Forest:", round(auc(forest_auc),3)))
```

```
## [1] "AUC for Random Forest: 0.936"
```

Classification using Logistic Regression

Training and Test data set

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

```
# Training the model
logistic <- glm(y ~ ., data = training, family = "binomial") # Using backward stepwise regression
summary(logistic)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6617  -0.3788  -0.2544  -0.1492   3.5052
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.267e+00  2.178e-01 -10.410  < 2e-16 ***
## age                -3.181e-03  2.617e-03  -1.216 0.224049
## jobblue-collar     -3.203e-01  8.664e-02  -3.696 0.000219 ***
## jobentrepreneur    -4.593e-01  1.494e-01  -3.075 0.002107 **
## jobhousemaid       -5.971e-01  1.648e-01  -3.623 0.000292 ***
## jobmanagement      -2.452e-01  8.721e-02  -2.811 0.004938 **
## jobretired          2.761e-01  1.163e-01   2.375 0.017545 *
## jobself-employed   -3.994e-01  1.356e-01  -2.945 0.003226 **
## jobservices        -1.647e-01  9.865e-02  -1.670 0.094991 .
## jobstudent          4.502e-01  1.284e-01   3.506 0.000455 ***
## jobtechnician      -1.670e-01  8.146e-02  -2.050 0.040393 *
## jobunemployed      -1.788e-01  1.349e-01  -1.325 0.185064
## jobunknown         -2.535e-01  2.656e-01  -0.954 0.339955
## maritalmarried     -1.746e-01  7.107e-02  -2.456 0.014033 *
## maritalsingle       1.098e-01  8.066e-02   1.361 0.173484
## educationsecondary  1.508e-01  7.733e-02   1.950 0.051166 .
## educationtertiary   4.029e-01  8.966e-02   4.494 7.00e-06 ***
## educationunknown    2.684e-01  1.217e-01   2.205 0.027438 *
## defaultyes         -5.163e-02  1.998e-01  -0.258 0.796130
## balance             1.419e-05  6.288e-06   2.257 0.024001 *
## housingyes         -7.559e-01  5.218e-02 -14.486  < 2e-16 ***
## loanyes            -4.385e-01  7.092e-02  -6.184 6.25e-10 ***
## contacttelephone   -5.792e-02  8.727e-02  -0.664 0.506884
## contactunknown     -1.607e+00  8.765e-02 -18.329  < 2e-16 ***
## day                 9.199e-03  2.964e-03   3.104 0.001912 **
## monthaug           -7.023e-01  9.347e-02  -7.514 5.73e-14 ***
## monthdec            6.653e-01  2.075e-01   3.205 0.001349 **
## monthfeb           -1.013e-01  1.058e-01  -0.957 0.338398
```

```
## monthjan           -1.262e+00  1.447e-01  -8.720  < 2e-16 ***
## monthjul           -7.333e-01  9.122e-02  -8.039 9.07e-16 ***
## monthjun            4.284e-01  1.109e-01   3.863 0.000112 ***
## monthmar            1.603e+00  1.412e-01  11.354  < 2e-16 ***
## monthmay           -3.506e-01  8.533e-02  -4.108 3.98e-05 ***
## monthnov           -7.839e-01  9.906e-02  -7.913 2.51e-15 ***
## monthoct            9.515e-01  1.272e-01   7.483 7.27e-14 ***
## monthsep            6.967e-01  1.438e-01   4.843 1.28e-06 ***
## duration            4.085e-03  7.639e-05  53.474  < 2e-16 ***
## campaign           -9.407e-02  1.220e-02  -7.709 1.26e-14 ***
## pdays              -8.017e-05  3.615e-04  -0.222 0.824505
## previous            7.343e-03  6.423e-03   1.143 0.252900
## poutcomeother       1.330e-01  1.058e-01   1.257 0.208884
## poutcomesuccess     2.167e+00  9.688e-02  22.370  < 2e-16 ***
## poutcomeunknown    -1.613e-01  1.090e-01  -1.480 0.138794
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22845  on 31648  degrees of freedom
## Residual deviance: 15242  on 31606  degrees of freedom
## AIC: 15328
##
## Number of Fisher Scoring iterations: 6
```

```r
# Checking of Variance Inflation Factor (VIF) for Multicollinearity
vif_values <- vif(logistic)
print(vif_values)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## age       2.189009  1        1.479530
## job       4.139507 11        1.066702
## marital   1.440637  2        1.095566
## education 2.256879  3        1.145297
## default   1.018526  1        1.009220
## balance   1.044967  1        1.022236
## housing   1.424391  1        1.193478
## loan      1.066380  1        1.032657
## contact   1.884716  2        1.171687
## day       1.343237  1        1.158981
## month     3.705411 11        1.061344
## duration  1.122949  1        1.059693
## campaign  1.109249  1        1.053209
## pdays     3.689718  1        1.920864
## previous  1.199988  1        1.095440
## poutcome  4.150199  3        1.267685
```

Based on the initial result above, all VIF are within the acceptable level. However, there are some factors that is not significant with the model. Thus, we will be using the step wise backward regression technique to determine the best model for logistic regression.

```r
# Using stepwise backward regression technique
logistic_model <- stepAIC(logistic, direction = "backward", trace = FALSE)
summary(logistic_model)
```

```
##
## Call:
## glm(formula = y ~ job + marital + education + balance + housing +
##     loan + contact + day + month + duration + campaign + poutcome,
##     family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6698  -0.3783  -0.2547  -0.1493   3.5002
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.419e+00  1.549e-01 -15.612  < 2e-16 ***
## jobblue-collar     -3.187e-01  8.655e-02  -3.683 0.000231 ***
## jobentrepreneur    -4.679e-01  1.492e-01  -3.135 0.001718 **
## jobhousemaid       -6.133e-01  1.645e-01  -3.727 0.000194 ***
## jobmanagement      -2.487e-01  8.712e-02  -2.855 0.004309 **
## jobretired          2.138e-01  1.046e-01   2.044 0.040931 *
## jobself-employed   -4.014e-01  1.356e-01  -2.961 0.003067 **
## jobservices        -1.637e-01  9.859e-02  -1.660 0.096835 .
## jobstudent          4.811e-01  1.260e-01   3.817 0.000135 ***
## jobtechnician      -1.676e-01  8.142e-02  -2.058 0.039604 *
## jobunemployed      -1.820e-01  1.349e-01  -1.349 0.177277
## jobunknown         -2.695e-01  2.653e-01  -1.016 0.309816
## maritalmarried     -1.654e-01  7.076e-02  -2.338 0.019409 *
## maritalsingle       1.440e-01  7.576e-02   1.900 0.057372 .
## educationsecondary  1.603e-01  7.691e-02   2.084 0.037193 *
## educationtertiary   4.167e-01  8.896e-02   4.683 2.82e-06 ***
## educationunknown    2.661e-01  1.217e-01   2.186 0.028803 *
## balance             1.367e-05  6.265e-06   2.181 0.029152 *
## housingyes         -7.493e-01  5.170e-02 -14.495  < 2e-16 ***
## loanyes            -4.380e-01  7.071e-02  -6.194 5.86e-10 ***
## contacttelephone   -7.289e-02  8.627e-02  -0.845 0.398170
## contactunknown     -1.609e+00  8.758e-02 -18.367  < 2e-16 ***
## day                 9.126e-03  2.963e-03   3.080 0.002070 **
## monthaug           -7.050e-01  9.336e-02  -7.551 4.32e-14 ***
## monthdec            6.665e-01  2.075e-01   3.212 0.001319 **
## monthfeb           -9.944e-02  1.056e-01  -0.942 0.346234
## monthjan           -1.259e+00  1.447e-01  -8.703  < 2e-16 ***
## monthjul           -7.312e-01  9.111e-02  -8.025 1.02e-15 ***
## monthjun            4.297e-01  1.109e-01   3.875 0.000107 ***
## monthmar            1.602e+00  1.411e-01  11.348  < 2e-16 ***
## monthmay           -3.490e-01  8.525e-02  -4.094 4.24e-05 ***
## monthnov           -7.839e-01  9.850e-02  -7.959 1.74e-15 ***
## monthoct            9.492e-01  1.270e-01   7.473 7.84e-14 ***
## monthsep            6.970e-01  1.438e-01   4.848 1.25e-06 ***
## duration            4.085e-03  7.638e-05  53.481  < 2e-16 ***
## campaign           -9.391e-02  1.219e-02  -7.701 1.35e-14 ***
## poutcomeother       1.448e-01  1.054e-01   1.374 0.169342
```

```
## poutcomesuccess     2.172e+00  9.405e-02  23.097  < 2e-16 ***
## poutcomeunknown    -1.617e-01  6.743e-02  -2.399 0.016462 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22845  on 31648  degrees of freedom
## Residual deviance: 15245  on 31610  degrees of freedom
## AIC: 15323
##
## Number of Fisher Scoring iterations: 6
```

The best model determine by Stepwise Backward Regression consist of factors such as job, marital status, education, balance, housing, loan, contact, day, month, duration, campaign and poutcome

Model Prediction

```
# Make predictions on the testing set
predictions <- predict(logistic_model, newdata = testing, type = "response")
```

```
# Predicting in the test dataset
pred_prob <- predict(logistic_model, testing, type = "response")

# Converting from probability to actual output
testing$pred_class <- ifelse(pred_prob >= 0.5, "yes", "no")
testing$pred_class <- as.factor(testing$pred_class)
# Generating the classification table

conf_matrix <- confusionMatrix(testing$pred_class, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  11699  1032
##        yes   277   554
##
##                Accuracy : 0.9035
##                  95% CI : (0.8984, 0.9084)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 1.639e-14
##
##                   Kappa : 0.4111
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9769
```

```
##             Specificity : 0.3493
##          Pos Pred Value : 0.9189
##          Neg Pred Value : 0.6667
##              Prevalence : 0.8831
##          Detection Rate : 0.8626
##    Detection Prevalence : 0.9387
##       Balanced Accuracy : 0.6631
##
##        'Positive' Class : no
##
```

```
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.903
```

```
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.977
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.349
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.919
```

```
# Calculate AUC for Logistic Regression
logistic_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
logistic_auc_value <- auc(logistic_auc)
print(paste("AUC for Logistic Regression:", round(logistic_auc_value,3)))
```

```
## [1] "AUC for Logistic Regression: 0.909"
```

Classification using Logistic Regression with SMOTE

Training and Test data set

```r
# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bank$y, p = 0.7, list = FALSE)
training <- bank[trainIndex, ]
testing <- bank[-trainIndex, ]
```

To preserve the accuracy of the testing data set, we will only apply SMOTE to the training data set

```r
training_balance <- ovun.sample(y ~ ., data = training, method = "over", N = 48000)$data
prop.table(table(training_balance$y))
```

```
##
##        no       yes
## 0.5822083 0.4177917
```

After using SMOTE, the distribution of the dependent variable are now became stable. Now, we will build the model again.

```r
# Training the model
logistic <- glm(y ~ ., data = training_balance, family = "binomial") # Using backward stepwise regressi
summary(logistic)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = training_balance)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.9478  -0.5842  -0.2647   0.5745   3.0389
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -7.227e-01  1.305e-01  -5.538 3.07e-08 ***
## age                -3.521e-03  1.563e-03  -2.253 0.024271 *
## jobblue-collar     -4.322e-01  5.093e-02  -8.485  < 2e-16 ***
## jobentrepreneur    -4.232e-01  8.441e-02  -5.013 5.35e-07 ***
## jobhousemaid       -5.447e-01  9.292e-02  -5.862 4.57e-09 ***
## jobmanagement      -3.190e-01  5.237e-02  -6.092 1.12e-09 ***
## jobretired          3.225e-01  7.167e-02   4.499 6.82e-06 ***
## jobself-employed   -5.217e-01  8.107e-02  -6.435 1.24e-10 ***
## jobservices        -1.996e-01  5.782e-02  -3.452 0.000556 ***
## jobstudent          5.640e-01  8.275e-02   6.816 9.38e-12 ***
## jobtechnician      -1.842e-01  4.823e-02  -3.818 0.000134 ***
## jobunemployed      -1.907e-01  8.243e-02  -2.313 0.020723 *
## jobunknown         -2.245e-01  1.579e-01  -1.421 0.155217
## maritalmarried     -1.334e-01  4.242e-02  -3.145 0.001658 **
## maritalsingle       1.358e-01  4.848e-02   2.800 0.005110 **
## educationsecondary  1.820e-01  4.516e-02   4.031 5.56e-05 ***
## educationtertiary   4.834e-01  5.307e-02   9.109  < 2e-16 ***
## educationunknown    3.213e-01  7.346e-02   4.374 1.22e-05 ***
## defaultyes         -7.779e-02  1.106e-01  -0.704 0.481669
```

```
## balance             2.288e-05  4.193e-06   5.456 4.88e-08 ***
## housingyes         -8.127e-01  3.047e-02 -26.669  < 2e-16 ***
## loanyes            -5.879e-01  4.100e-02 -14.338  < 2e-16 ***
## contacttelephone    3.595e-02  5.230e-02   0.687 0.491847
## contactunknown     -1.564e+00  4.758e-02 -32.863  < 2e-16 ***
## day                 3.912e-03  1.736e-03   2.253 0.024252 *
## monthaug           -9.179e-01  5.459e-02 -16.815  < 2e-16 ***
## monthdec            6.386e-01  1.459e-01   4.377 1.21e-05 ***
## monthfeb           -1.358e-01  6.251e-02  -2.173 0.029814 *
## monthjan           -1.272e+00  8.230e-02 -15.452  < 2e-16 ***
## monthjul           -9.638e-01  5.410e-02 -17.816  < 2e-16 ***
## monthjun            2.359e-01  6.428e-02   3.670 0.000243 ***
## monthmar            1.740e+00  1.009e-01  17.244  < 2e-16 ***
## monthmay           -5.856e-01  5.136e-02 -11.400  < 2e-16 ***
## monthnov           -9.071e-01  5.882e-02 -15.422  < 2e-16 ***
## monthoct            1.269e+00  8.773e-02  14.461  < 2e-16 ***
## monthsep            7.538e-01  9.770e-02   7.715 1.21e-14 ***
## duration            5.452e-03  6.044e-05  90.207  < 2e-16 ***
## campaign           -1.093e-01  6.871e-03 -15.913  < 2e-16 ***
## pdays              -2.050e-04  2.137e-04  -0.959 0.337466
## previous            1.490e-02  7.384e-03   2.018 0.043610 *
## poutcomeother       1.129e-01  6.429e-02   1.756 0.079136 .
## poutcomesuccess     2.292e+00  6.881e-02  33.305  < 2e-16 ***
## poutcomeunknown    -3.104e-01  6.797e-02  -4.567 4.95e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 65239  on 47999  degrees of freedom
## Residual deviance: 38515  on 47957  degrees of freedom
## AIC: 38601
##
## Number of Fisher Scoring iterations: 5
```

```r
# Checking of Variance Inflation Factor (VIF) for Multicollinearity
vif_values <- vif(logistic)
print(vif_values)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## age        2.134241  1        1.460904
## job        4.015170 11        1.065224
## marital    1.434412  2        1.094381
## education  2.309782  3        1.149728
## default    1.021867  1        1.010874
## balance    1.052482  1        1.025905
## housing    1.402452  1        1.184252
## loan       1.061782  1        1.030428
## contact    1.868139  2        1.169102
## day        1.289008  1        1.135345
## month      3.167132 11        1.053798
## duration   1.201867  1        1.096297
## campaign   1.100048  1        1.048832
## pdays      3.640781  1        1.908083
```

```
## previous  1.648922  1        1.284104
## poutcome  4.676257  3        1.293152
```

Based on the initial result above, all VIF are within the acceptable level. However, there are some factors that is not significant with the model. Thus, we will be using the step wise backward regression technique to determine the best model for logistic regression.

```
# Using stepwise backward regression technique
logistic_model <- stepAIC(logistic, direction = "backward", trace = FALSE)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logistic_model)
```

```
##
## Call:
## glm(formula = y ~ age + job + marital + education + balance +
##     housing + loan + contact + day + month + duration + campaign +
##     previous + poutcome, family = "binomial", data = training_balance)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.9483  -0.5849  -0.2645   0.5749   3.0415
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -7.730e-01  1.198e-01  -6.451 1.11e-10 ***
## age                -3.548e-03  1.562e-03  -2.271 0.023161 *
## jobblue-collar     -4.328e-01  5.091e-02  -8.502  < 2e-16 ***
## jobentrepreneur    -4.254e-01  8.436e-02  -5.043 4.58e-07 ***
## jobhousemaid       -5.429e-01  9.287e-02  -5.846 5.03e-09 ***
## jobmanagement      -3.197e-01  5.236e-02  -6.107 1.02e-09 ***
## jobretired          3.241e-01  7.164e-02   4.524 6.07e-06 ***
## jobself-employed   -5.218e-01  8.106e-02  -6.437 1.22e-10 ***
## jobservices        -2.001e-01  5.783e-02  -3.461 0.000539 ***
## jobstudent          5.664e-01  8.271e-02   6.848 7.50e-12 ***
## jobtechnician      -1.837e-01  4.822e-02  -3.810 0.000139 ***
## jobunemployed      -1.932e-01  8.241e-02  -2.344 0.019071 *
## jobunknown         -2.255e-01  1.580e-01  -1.428 0.153347
## maritalmarried     -1.321e-01  4.240e-02  -3.115 0.001839 **
## maritalsingle       1.362e-01  4.848e-02   2.810 0.004952 **
## educationsecondary  1.818e-01  4.515e-02   4.027 5.65e-05 ***
## educationtertiary   4.849e-01  5.305e-02   9.140  < 2e-16 ***
## educationunknown    3.212e-01  7.346e-02   4.373 1.23e-05 ***
## balance             2.314e-05  4.187e-06   5.525 3.29e-08 ***
## housingyes         -8.149e-01  3.040e-02 -26.808  < 2e-16 ***
## loanyes            -5.897e-01  4.090e-02 -14.418  < 2e-16 ***
## contacttelephone    3.571e-02  5.229e-02   0.683 0.494686
## contactunknown     -1.562e+00  4.755e-02 -32.856  < 2e-16 ***
## day                 3.933e-03  1.736e-03   2.265 0.023513 *
## monthaug           -9.164e-01  5.455e-02 -16.799  < 2e-16 ***
## monthdec            6.416e-01  1.459e-01   4.397 1.10e-05 ***
## monthfeb           -1.324e-01  6.243e-02  -2.121 0.033930 *
```

```
## monthjan           -1.270e+00  8.226e-02 -15.435  < 2e-16 ***
## monthjul           -9.644e-01  5.406e-02 -17.838  < 2e-16 ***
## monthjun            2.373e-01  6.424e-02   3.694 0.000220 ***
## monthmar            1.742e+00  1.008e-01  17.276  < 2e-16 ***
## monthmay           -5.860e-01  5.136e-02 -11.409  < 2e-16 ***
## monthnov           -9.010e-01  5.839e-02 -15.430  < 2e-16 ***
## monthoct            1.272e+00  8.769e-02  14.503  < 2e-16 ***
## monthsep            7.544e-01  9.767e-02   7.724 1.13e-14 ***
## duration            5.452e-03  6.043e-05  90.216  < 2e-16 ***
## campaign           -1.094e-01  6.871e-03 -15.925  < 2e-16 ***
## previous            1.520e-02  7.384e-03   2.058 0.039572 *
## poutcomeother       1.154e-01  6.420e-02   1.797 0.072364 .
## poutcomesuccess     2.306e+00  6.734e-02  34.242  < 2e-16 ***
## poutcomeunknown    -2.621e-01  4.481e-02  -5.848 4.99e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 65239  on 47999  degrees of freedom
## Residual deviance: 38516  on 47959  degrees of freedom
## AIC: 38598
##
## Number of Fisher Scoring iterations: 5
```

The best model determine by Stepwise Backward Regression consist of factors such as job, marital status, education, balance, housing, loan, contact, day, month, duration, campaign and poutcome

Model Prediction

```r
# Make predictions on the testing set
predictions <- predict(logistic_model, newdata = testing, type = "response")
```

```r
# Predicting in the test dataset
pred_prob <- predict(logistic_model, testing, type = "response")

# Converting from probability to actual output
testing$pred_class <- ifelse(pred_prob >= 0.5, "yes", "no")
testing$pred_class <- as.factor(testing$pred_class)
# Generating the classification table

conf_matrix <- confusionMatrix(testing$pred_class, testing$y)

# Print the confusion matrix and the overall accuracy
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10594   381
```

```
##        yes  1382  1205
##
##                Accuracy : 0.87
##                  95% CI : (0.8642, 0.8756)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.5059
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8846
##             Specificity : 0.7598
##          Pos Pred Value : 0.9653
##          Neg Pred Value : 0.4658
##              Prevalence : 0.8831
##          Detection Rate : 0.7812
##    Detection Prevalence : 0.8092
##       Balanced Accuracy : 0.8222
##
##        'Positive' Class : no
##
```

```
cat("Overall Accuracy:", round(conf_matrix$overall[1], 3), "\n")
```

```
## Overall Accuracy: 0.87
```

```
# Calculate precision, recall, and F1 score
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- conf_matrix$byClass[3]
cat("Precision:", round(precision, 3), "\n")
```

```
## Precision: 0.885
```

```
cat("Recall:", round(recall, 3), "\n")
```

```
## Recall: 0.76
```

```
cat("F1 Score:", round(f1_score, 3), "\n")
```

```
## F1 Score: 0.965
```

```
# Calculate AUC for Logistic Regression
logistic_auc <- roc(testing$y, predictions)
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```r
logistic_auc_value <- auc(logistic_auc)
print(paste("AUC for Logistic Regression:", round(logistic_auc_value,3)))
```

```
## [1] "AUC for Logistic Regression: 0.912"
```

SUMMARY

```r
summary<- matrix(NA, nrow = 6, ncol = 5)
# Set column names
colnames(summary) <- c("Accuracy", "Precision", "Recall", "F1 Score", "AUC")

# Set row names
rownames(summary) <- c("Decision Tree", "Decision Tree with SMOTE",
                       "Random Forest", "Random Forest with SMOTE",
                       "Logistic Regression","Logistic Regression with SMOTE")

# For Decision Tree
summary[1,1] <- 0.902
summary[1,2] <- 0.966
summary[1,3] <- 0.414
summary[1,4] <- 0.926
summary[1,5] <- 0.803

# For Decision Tree with Smote
summary[2,1] <- 0.865
summary[2,2] <- 0.878
summary[2,3] <- 0.767
summary[2,4] <- 0.966
summary[2,5] <- 0.861

# For Random Forest
summary[3,1] <- 0.911
summary[3,2] <- 0.967
summary[3,3] <- 0.491
summary[3,4] <- 0.935
summary[3,5] <- 0.935

# For Random Forest with Smote
summary[4,1] <- 0.911
summary[4,2] <- 0.949
summary[4,3] <- 0.625
summary[4,4] <- 0.950
summary[4,5] <- 0.936

# For Logistic Regression
summary[5,1] <- 0.903
summary[5,2] <- 0.977
summary[5,3] <- 0.349
summary[5,4] <- 0.919
summary[5,5] <- 0.909
```

```r
# For Logistic Regression with Smote
summary[6,1] <- 0.870
summary[6,2] <- 0.885
summary[6,3] <- 0.760
summary[6,4] <- 0.965
summary[6,5] <- 0.912

summary
```

```
##                                Accuracy Precision Recall F1 Score   AUC
## Decision Tree                     0.902     0.966  0.414    0.926 0.803
## Decision Tree with SMOTE          0.865     0.878  0.767    0.966 0.861
## Random Forest                     0.911     0.967  0.491    0.935 0.935
## Random Forest with SMOTE          0.911     0.949  0.625    0.950 0.936
## Logistic Regression               0.903     0.977  0.349    0.919 0.909
## Logistic Regression with SMOTE    0.870     0.885  0.760    0.965 0.912
```

Based on the summarized table of all the performances of the different models, Random Forest, Random Forest with SMOTE and Logistic Regression are the top 3 models with highest accuracy. Thus, we will be choosing the best model among the three. While Random Forest with SMOTE has higher accuracy, ovarsampling method sometimes leads to better performance. Since we are interested in targeting customers most likely to avail the term deposit, using the original data set will lead to more acceptable and reliable results. Thus, we will remove that to out options and now have two options: Random Forest and Logistic Regresion. Since both of them have accuracy greater than 90%, this impliease that these two models has excellent classification.

Analysis of Metrics

Precision vs. Recall: Precision measures the accuracy of positive predictions, while recall (sensitivity) measures the ability to find all positive instances. Logistic Regression shows a higher precision but significantly lower recall than Random Forest. This suggests that while Logistic Regression is more accurate when it predicts a positive class, it fails to capture a substantial number of actual positives compared to Random Forest.

F1 Score: The F1 Score is the harmonic mean of precision and recall. It is particularly useful when you need a balance between precision and recall. Random Forest has a higher F1 Score indicating a better balance between precision and recall compared to Logistic Regression.

Considerations for Model Selection

Model Complexity and Interpretability:

1. Random Forest is a more complex model as it builds multiple trees and aggregates their results. This complexity can lead to better performance but at the cost of interpretability.
2. Logistic Regression is inherently simpler and more interpretable. The coefficients of the model can be directly interpreted in terms of odds ratios, making it easier to explain to stakeholders. Computational Efficiency:
3. Random Forest can be computationally intensive, particularly with a large number of trees and deep tree structures. It requires more memory and processing power, especially during training.
4. Logistic Regression is generally faster to train and requires less computational resources, making it suitable for environments with limited computational capacity.

Conclusion and Recommendation

Given that **Logistic Regression** demonstrates the highest precision among the evaluated models, it becomes the preferable choice when the priority is minimizing costs associated with incorrect predictions. This model ensures that marketing efforts are directed only towards those most likely to avail the term deposit, thereby optimizing resource allocation and reducing waste. High precision is crucial in scenarios where the cost of targeting non-customers (false positives) outweighs the potential loss from not identifying every possible customer. This approach not only conserves budget but also enhances the effectiveness of marketing campaigns by focusing on high-probability leads.

Justification

1. Precision and Cost Minimization Logistic Regression has demonstrated the highest precision among the models evaluated. This implies that it has the highest likelihood of correctly identifying only those customers who are most likely to avail the term deposit. In scenarios where marketing resources are limited or costly, maximizing precision helps in reducing wasteful expenditure by targeting only the most promising leads.

2. Interpretability and Stakeholder Communication One of the strongest points in favor of Logistic Regression is its interpretability. The model coefficients can be easily translated into odds ratios, providing clear insights into how each predictor influences the probability of a customer availing the term deposit. This makes it easier to communicate the model's decision-making process to non-technical stakeholders, facilitating better business decisions.

3. Model Simplicity and Deployment Logistic Regression is not only easier to implement but also generally faster and less resource-intensive compared to more complex models like Random Forest. This simplicity can be advantageous when deploying the model in production environments where resources are a constraint or where real-time decision-making is required.

Final Model for reporting

```
summary(logistic_model)
```

```
##
## Call:
## glm(formula = y ~ age + job + marital + education + balance +
##     housing + loan + contact + day + month + duration + campaign +
##     previous + poutcome, family = "binomial", data = training_balance)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.9483  -0.5849  -0.2645   0.5749   3.0415
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -7.730e-01  1.198e-01  -6.451 1.11e-10 ***
## age                -3.548e-03  1.562e-03  -2.271 0.023161 *
## jobblue-collar     -4.328e-01  5.091e-02  -8.502  < 2e-16 ***
## jobentrepreneur    -4.254e-01  8.436e-02  -5.043 4.58e-07 ***
## jobhousemaid       -5.429e-01  9.287e-02  -5.846 5.03e-09 ***
## jobmanagement      -3.197e-01  5.236e-02  -6.107 1.02e-09 ***
```

```
## jobretired          3.241e-01  7.164e-02   4.524 6.07e-06 ***
## jobself-employed   -5.218e-01  8.106e-02  -6.437 1.22e-10 ***
## jobservices        -2.001e-01  5.783e-02  -3.461 0.000539 ***
## jobstudent          5.664e-01  8.271e-02   6.848 7.50e-12 ***
## jobtechnician      -1.837e-01  4.822e-02  -3.810 0.000139 ***
## jobunemployed      -1.932e-01  8.241e-02  -2.344 0.019071 *
## jobunknown         -2.255e-01  1.580e-01  -1.428 0.153347
## maritalmarried     -1.321e-01  4.240e-02  -3.115 0.001839 **
## maritalsingle       1.362e-01  4.848e-02   2.810 0.004952 **
## educationsecondary  1.818e-01  4.515e-02   4.027 5.65e-05 ***
## educationtertiary   4.849e-01  5.305e-02   9.140  < 2e-16 ***
## educationunknown    3.212e-01  7.346e-02   4.373 1.23e-05 ***
## balance             2.314e-05  4.187e-06   5.525 3.29e-08 ***
## housingyes         -8.149e-01  3.040e-02 -26.808  < 2e-16 ***
## loanyes            -5.897e-01  4.090e-02 -14.418  < 2e-16 ***
## contacttelephone    3.571e-02  5.229e-02   0.683 0.494686
## contactunknown     -1.562e+00  4.755e-02 -32.856  < 2e-16 ***
## day                 3.933e-03  1.736e-03   2.265 0.023513 *
## monthaug           -9.164e-01  5.455e-02 -16.799  < 2e-16 ***
## monthdec            6.416e-01  1.459e-01   4.397 1.10e-05 ***
## monthfeb           -1.324e-01  6.243e-02  -2.121 0.033930 *
## monthjan           -1.270e+00  8.226e-02 -15.435  < 2e-16 ***
## monthjul           -9.644e-01  5.406e-02 -17.838  < 2e-16 ***
## monthjun            2.373e-01  6.424e-02   3.694 0.000220 ***
## monthmar            1.742e+00  1.008e-01  17.276  < 2e-16 ***
## monthmay           -5.860e-01  5.136e-02 -11.409  < 2e-16 ***
## monthnov           -9.010e-01  5.839e-02 -15.430  < 2e-16 ***
## monthoct            1.272e+00  8.769e-02  14.503  < 2e-16 ***
## monthsep            7.544e-01  9.767e-02   7.724 1.13e-14 ***
## duration            5.452e-03  6.043e-05  90.216  < 2e-16 ***
## campaign           -1.094e-01  6.871e-03 -15.925  < 2e-16 ***
## previous            1.520e-02  7.384e-03   2.058 0.039572 *
## poutcomeother       1.154e-01  6.420e-02   1.797 0.072364 .
## poutcomesuccess     2.306e+00  6.734e-02  34.242  < 2e-16 ***
## poutcomeunknown    -2.621e-01  4.481e-02  -5.848 4.99e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 65239  on 47999  degrees of freedom
## Residual deviance: 38516  on 47959  degrees of freedom
## AIC: 38598
##
## Number of Fisher Scoring iterations: 5
```

Key Insights:

1. Age: The negative coefficient for age (-0.003548) suggests that as clients get older, the likelihood of subscribing decreases slightly. Given the p-value (<0.05), this effect is statistically significant.
2. Job: Different job categories have different impacts. For instance, retirees (jobretired) have a positive coefficient (0.3241), indicating they are more likely to subscribe compared to the baseline job category. Conversely, blue-collar workers (jobblue-collar) have a negative impact, being less likely to subscribe.

3. Marital Status: Being single (maritalsingle) positively impacts the subscription likelihood compared to being divorced (the baseline), whereas being married (maritalmarried) slightly decreases the likelihood.

4. Education: Higher education levels (secondary, tertiary) increase the likelihood of subscription. This indicates that clients with more education are more inclined to subscribe to term deposits.

5. Housing and Loan: Having a housing loan (housingyes) or a personal loan (loanyes) significantly decreases the likelihood of subscribing, possibly due to financial constraints.

6. Contact: Clients who were contacted via unknown methods (contactunknown) have a significantly lower likelihood of subscribing, possibly indicating the effectiveness of more personalized contact methods.

7. Month of Contact: Months like March (monthmar), October (monthoct), and December (monthdec) show a higher likelihood of subscription. These could be strategic months for campaigns.

8. Duration of Last Contact: A very strong predictor with duration having a high positive coefficient (0.005452), meaning longer calls significantly increase the probability of a client subscribing.

9. Campaign and Previous Contacts: More contacts in the current campaign (campaign) decrease the likelihood, possibly due to contact fatigue. However, more contacts from previous campaigns (previous) slightly increase the likelihood, suggesting that repeated contact over time may build trust or recognition.

10. Outcome of Previous Marketing Campaign: The success of previous marketing (poutcomesuccess) dramatically increases the likelihood of a subscription, highlighting the impact of positive past experiences.

Recommendations for Strategic Focus

1. Target younger clients and those in specific job categories like retirees and students for marketing term deposits.

2. Tailor the contact strategy to use more direct and personal methods rather than unknown methods.

3. Focus marketing efforts during months with historically higher success rates.

4. Consider lengthening the duration of contact calls as it has a strong positive impact on subscription likelihood.

5. Re-evaluate and potentially reduce the number of contacts per campaign to avoid contact fatigue while maintaining or increasing contacts from previous campaigns to reinforce recognition.

Next Steps with Logistic Regression

1. Model Optimization Before final deployment, consider tuning the model to enhance its performance further:

a. Feature Engineering: Refine existing features or create new ones that could help improve model accuracy.
b. Threshold Adjustment: Adjust the classification threshold to balance precision and recall according to business needs.
c. Cross-Validation: Use cross-validation to ensure the model's robustness and avoid overfitting.

2. Model Validation It's essential to validate the model with a fresh dataset or through techniques like cross-validation to ensure its performance holds up with new, unseen data. This will help confirm the model's generalizability and reliability.

3. Deployment and Monitoring Deploy the model into a production environment where it can start scoring actual customers: a. Integration: Integrate the model with existing customer databases and marketing systems. b.Monitoring: Regularly monitor the model's performance and make adjustments as needed based on feedback and changing data patterns.

4. Stakeholder Reporting Prepare reports and presentations for stakeholders detailing the model's predictive performance, the insights gained from model coefficients, and the expected impact on marketing strategies.