Bank Marketing Project

John Vincent Landingin

11/15/2021

Contents

Introduction

The goal in this project is to create an algorithm that will predict whether a client will subscribe to a bank term deposit. The data used is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. The dataset was collected from the following website - https://archive.ics.uci.edu/ml/datasets/bank+marketing.

In the dataset, there are 45211 instances and 16 attributes/variables. The output variable is binary which is if the client subscribed or not to a term deposit in the marketing campaign.

Below are the variables.

Input variables:
1 - age (numeric)
2 - job : type of job (categorical:“admin.”,“unknown”,“unemployed”,“management”,“housemaid”,“entrepreneur”,“student”,“bl
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”)
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)
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”)

Data exploration and visualization is done to check for each input variable on whether or how they may be used for building the algorithm. This gives a general idea for the dataset. Then for the modeling approach, both logistics regression and random forest is done. The former is done for benchmark purposes to compare whether a more complex model like Random Forest is necessary.

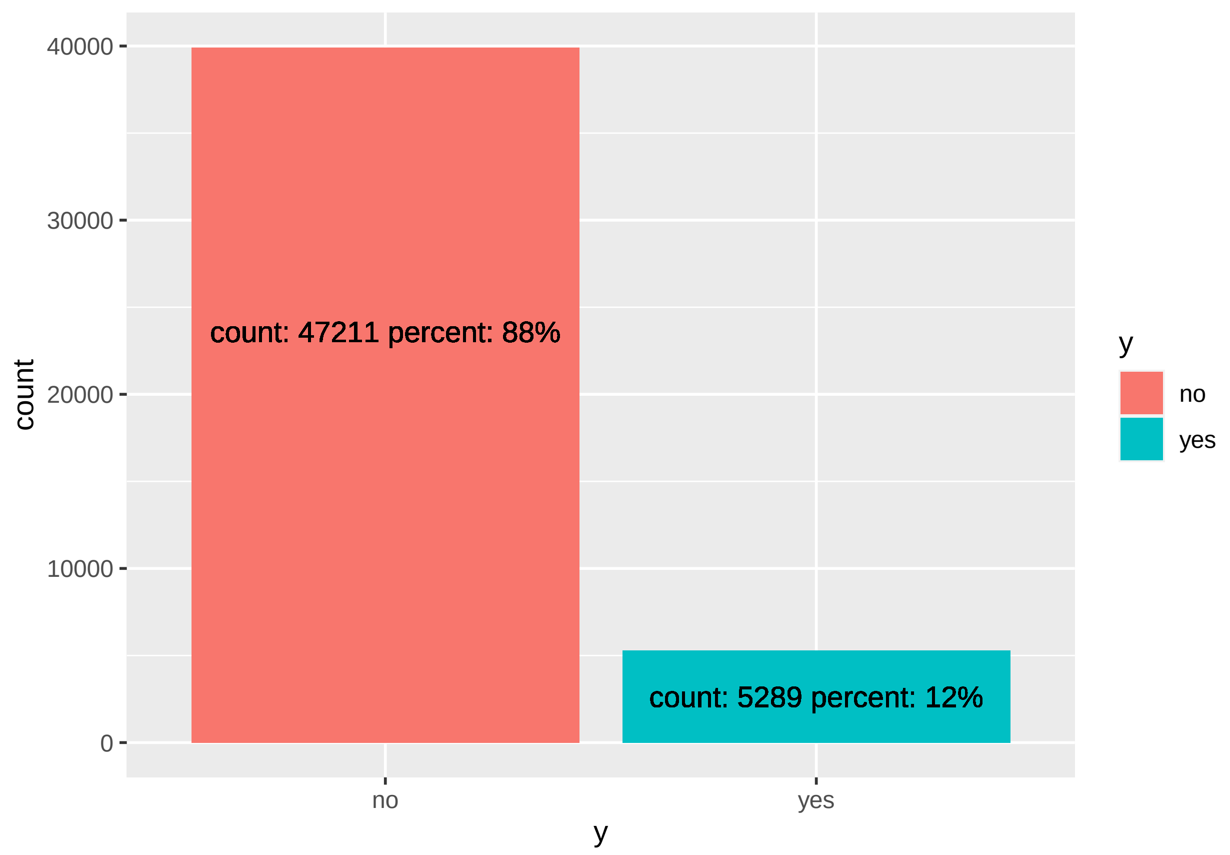
Analysis

Data Exploration and Visualization

For this section, we explore our variables and look for insights that could help in creating our model in predicting whether a client will subscribe to a term deposit after a campaign.

Demographic Profile

Subscribers (y) To get an initial idea, below is the distribution of clients who subscribed and did not subscribed.

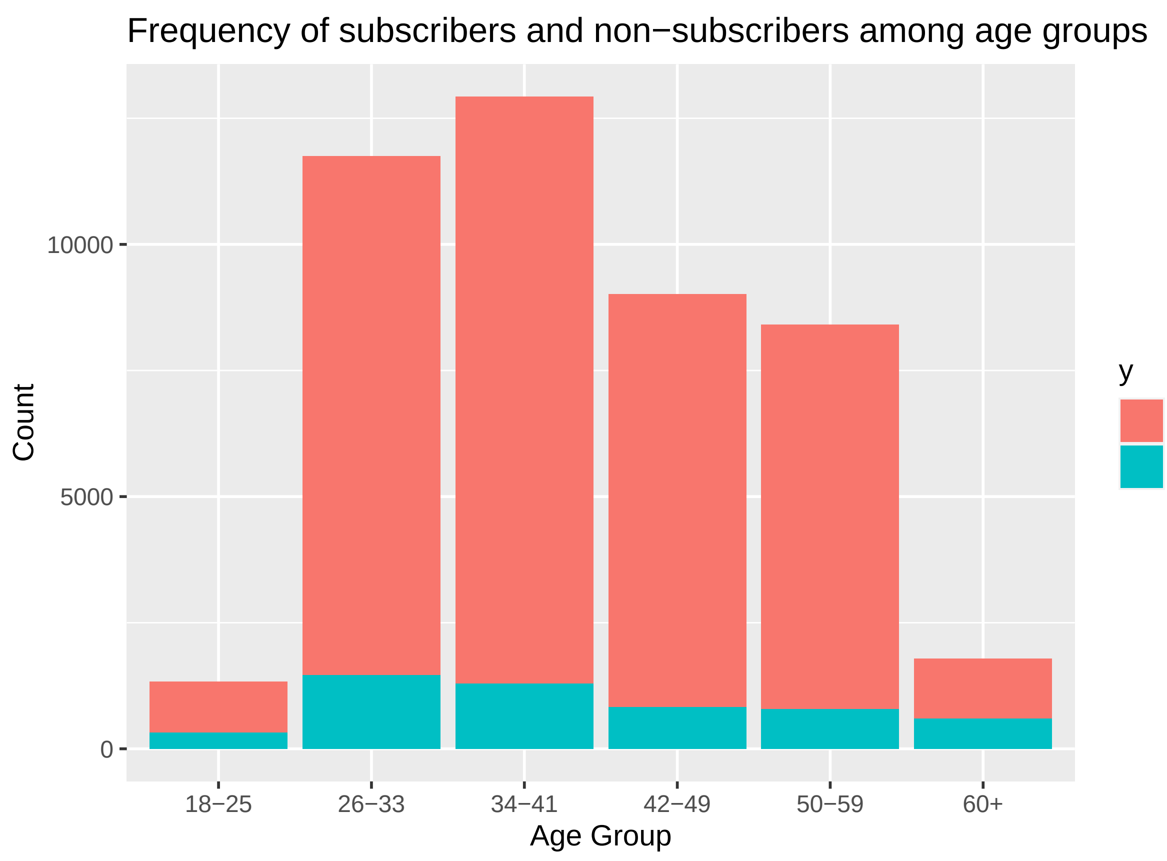


Overall, 5289 clients subscribed from the campaign, which is 12% of the total clients contacted during the campaign.

Age

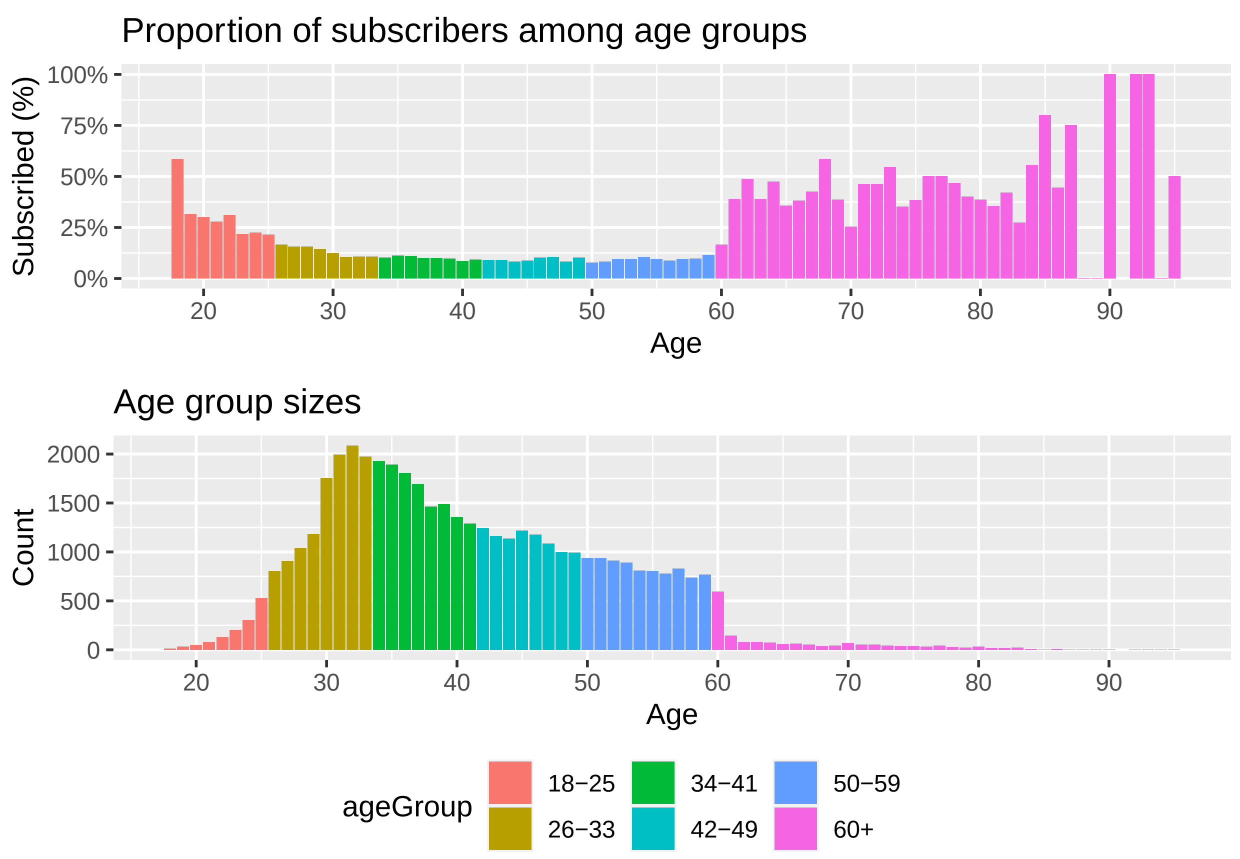
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   33.00   39.00   40.94   48.00   95.00
```

The minimum age in the data is 18, while the maximum age is 95. The Median age is 39.

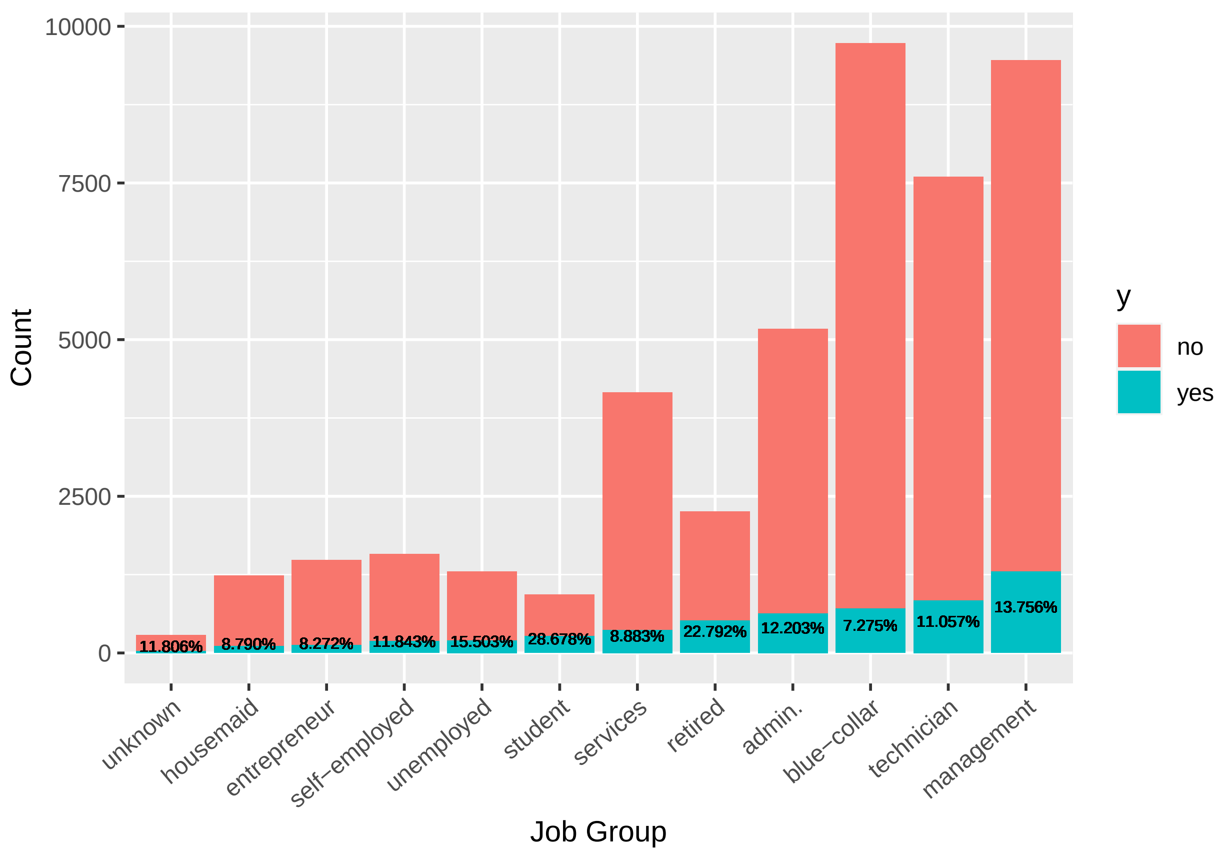


Based from the plot, most of the subscribers are aged between 26-33 and 34-41, while least of the subscribers are 18-25 and 60+.

```
## ‘summarise()‘ has grouped output by ’age’. You can override using the ‘.groups‘ argument.
## ‘summarise()‘ has grouped output by ’age’. You can override using the ‘.groups‘ argument.
```

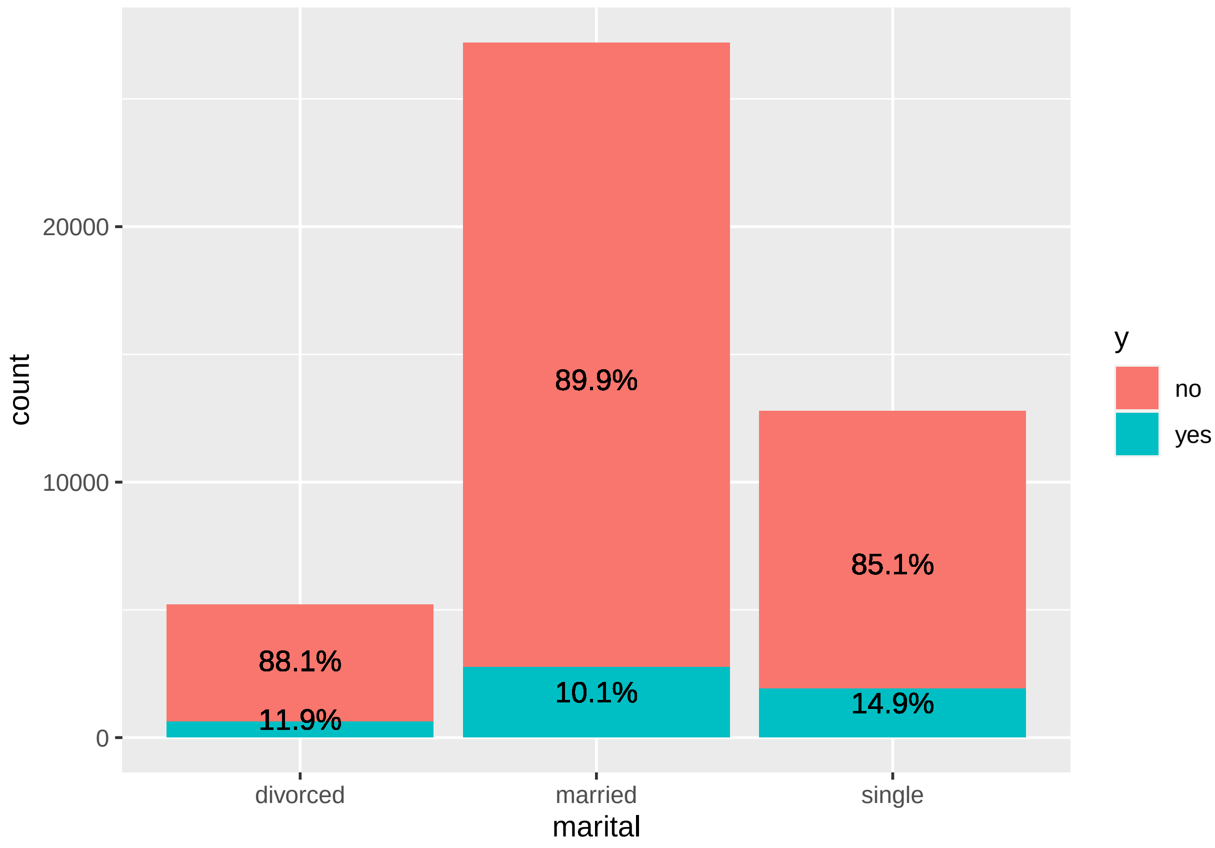


It can be seen from the plot that clients aged around 18-25 and 60+ are more likely to subscribe from a campaign but their group sizes are few which is why most subscribers are still aged between 26-33 as seen from the previous plot.



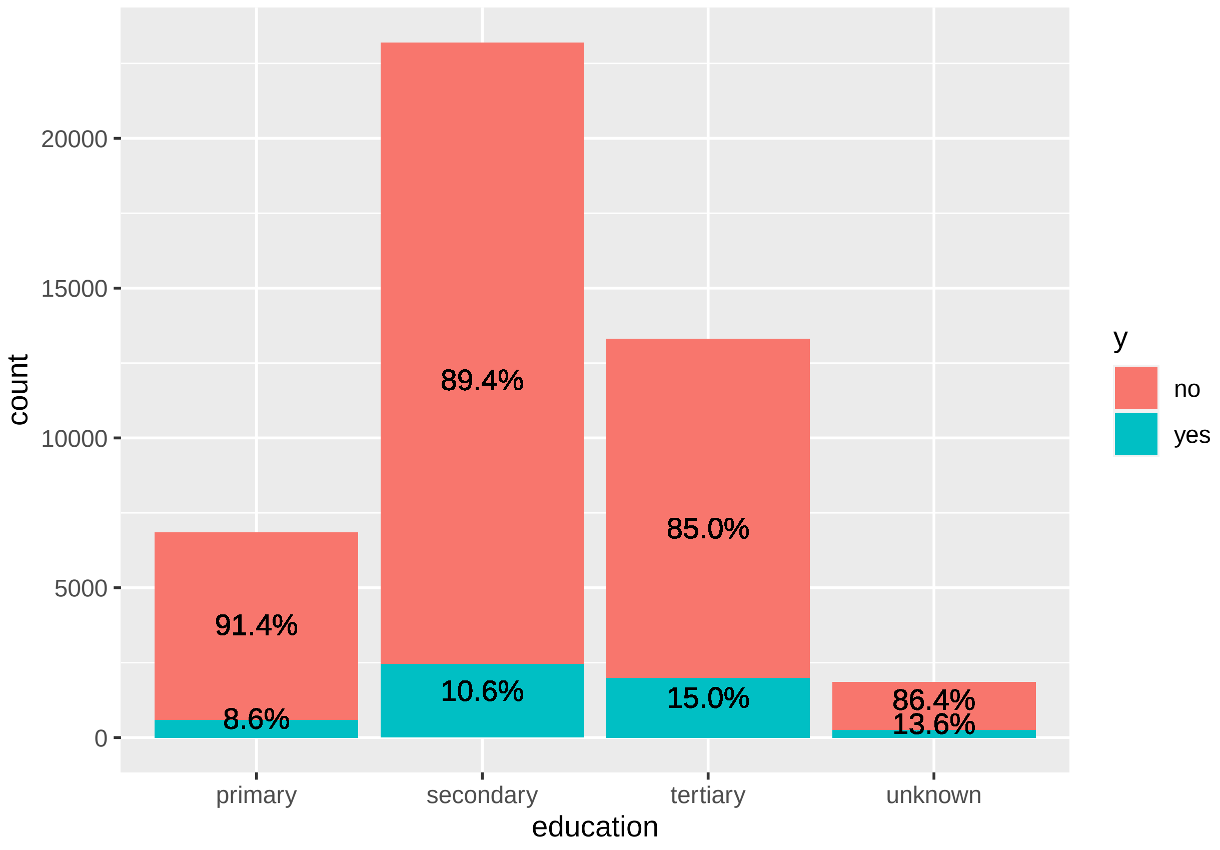
Job Group

Based from the plot, most of the clients called were blue-collar workers, but majority who subscribed were working with management. Meanwhile, least of the subscribers were housemaids. With regards to proportion of subscribers per job group, the student and retired were most likely to be subscribed with 28.7% and 22.8% of the group size, while blue-collar workers were least likely to be subscribed, with only 7.28% of the group.



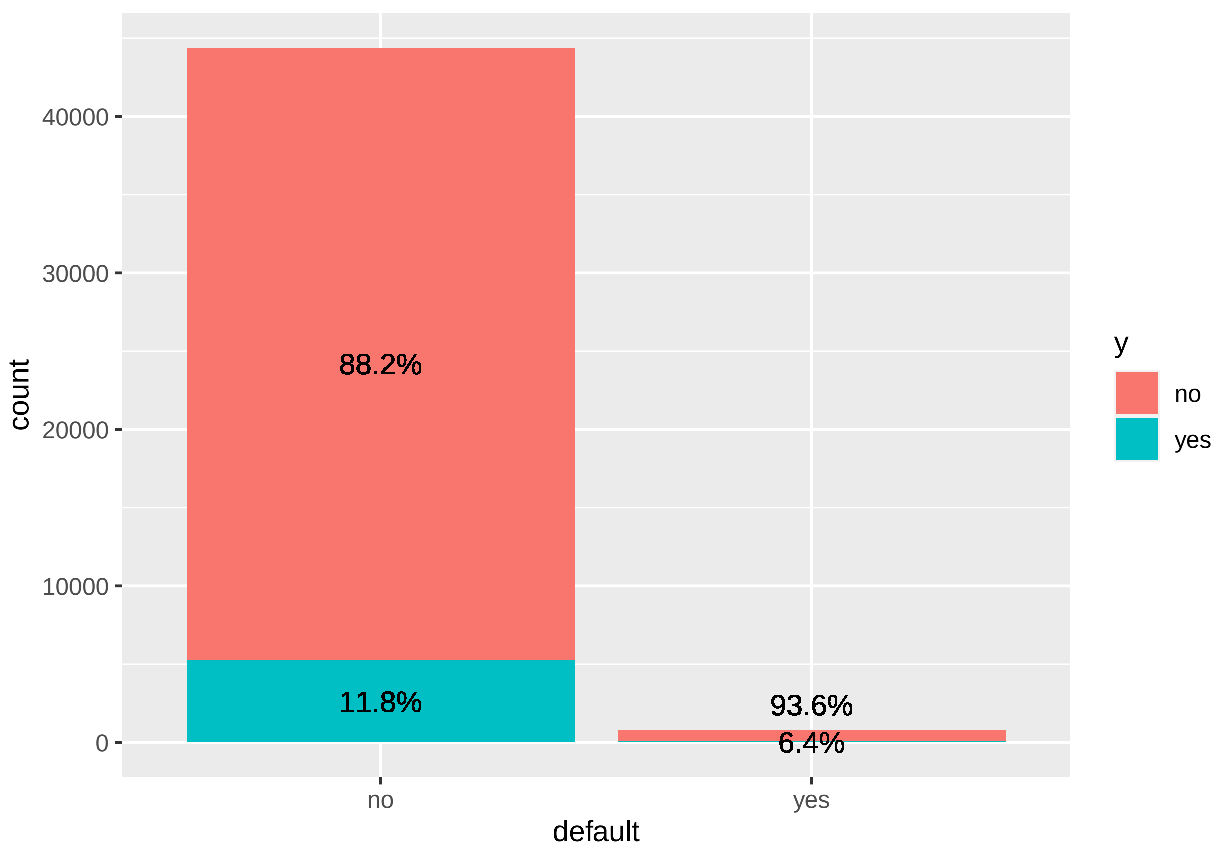
Marital Status

Based from the plot, majority of the subscribers are married, but only 10.1% of those married are subscribed. Moreover, single clients are more likely to be subscribers, comprising 14.9% of the single clients.

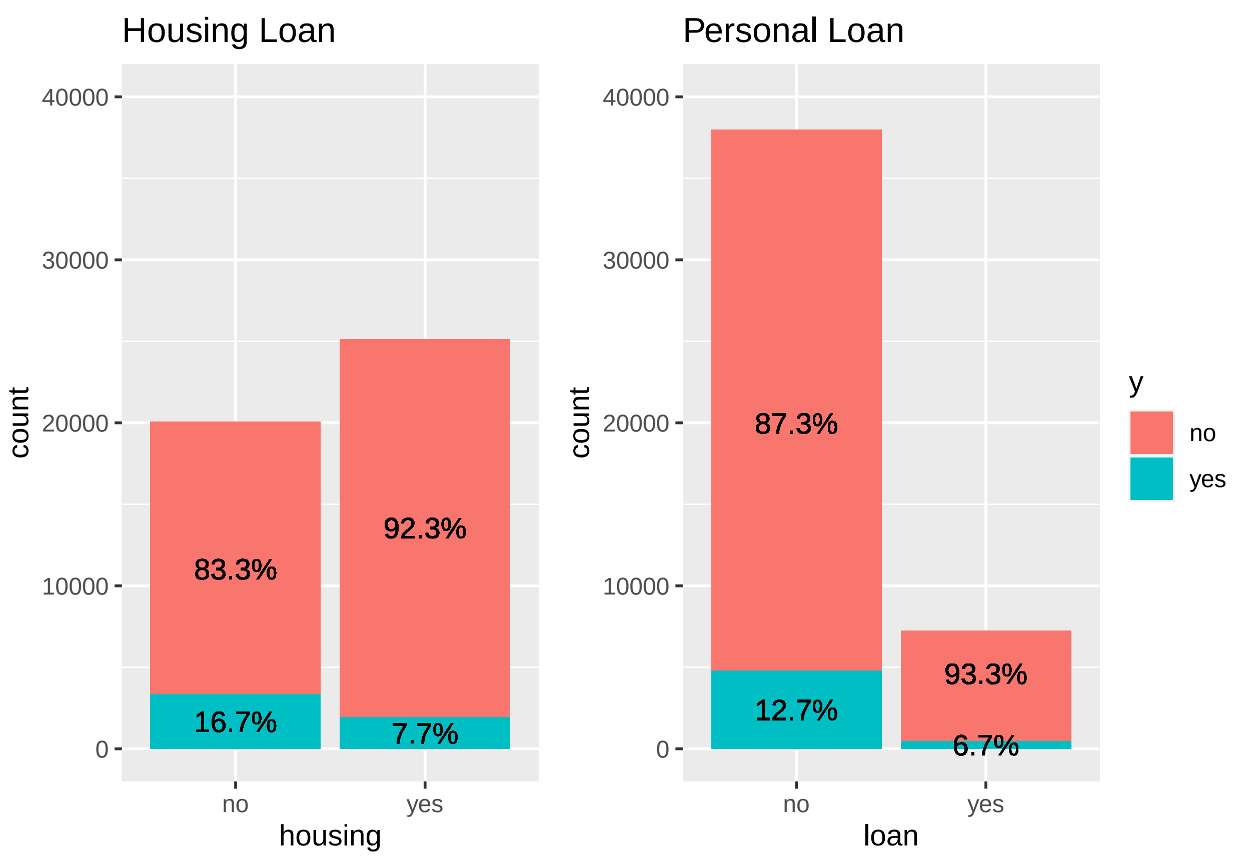


Education

Based from the plot, majority of the subscribers have secondary education level, but only 10.6% of those are subscribed. Furthermore, tertiary education level clients are more likely to be subscribers, comprising 15.0% of their group.



Default
Based from the plot, majority of the subscribers do not have credit in default, and are also more likely to be subscribed comprising 11.8% of their group.



Loan

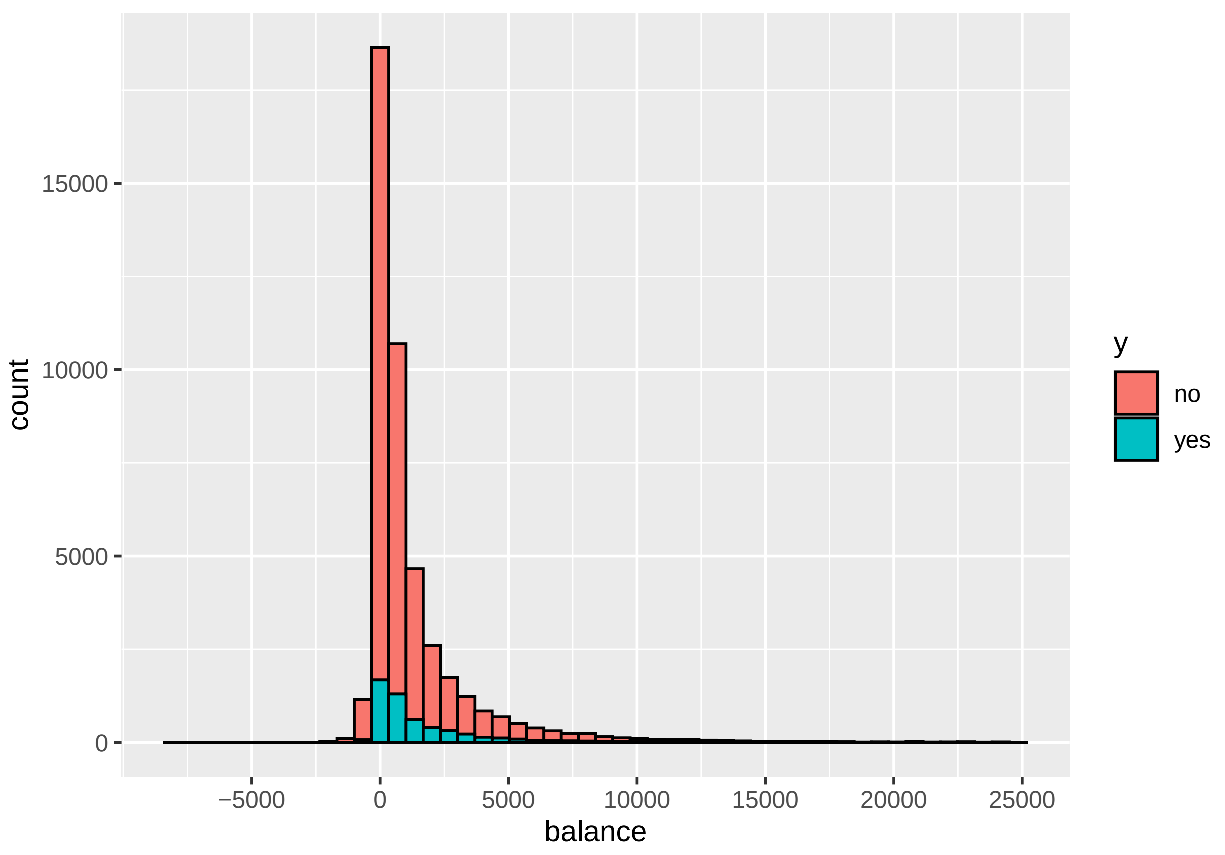
Based from the plot, majority of the clients had a housing loan but majority of the clients that subscribed did not have a housing loan. Furthermore, majority of the clients do not have a personal loan. Majority of the clients who also subscribed do not have a personal loan.

```
summary(dat$balance)
```

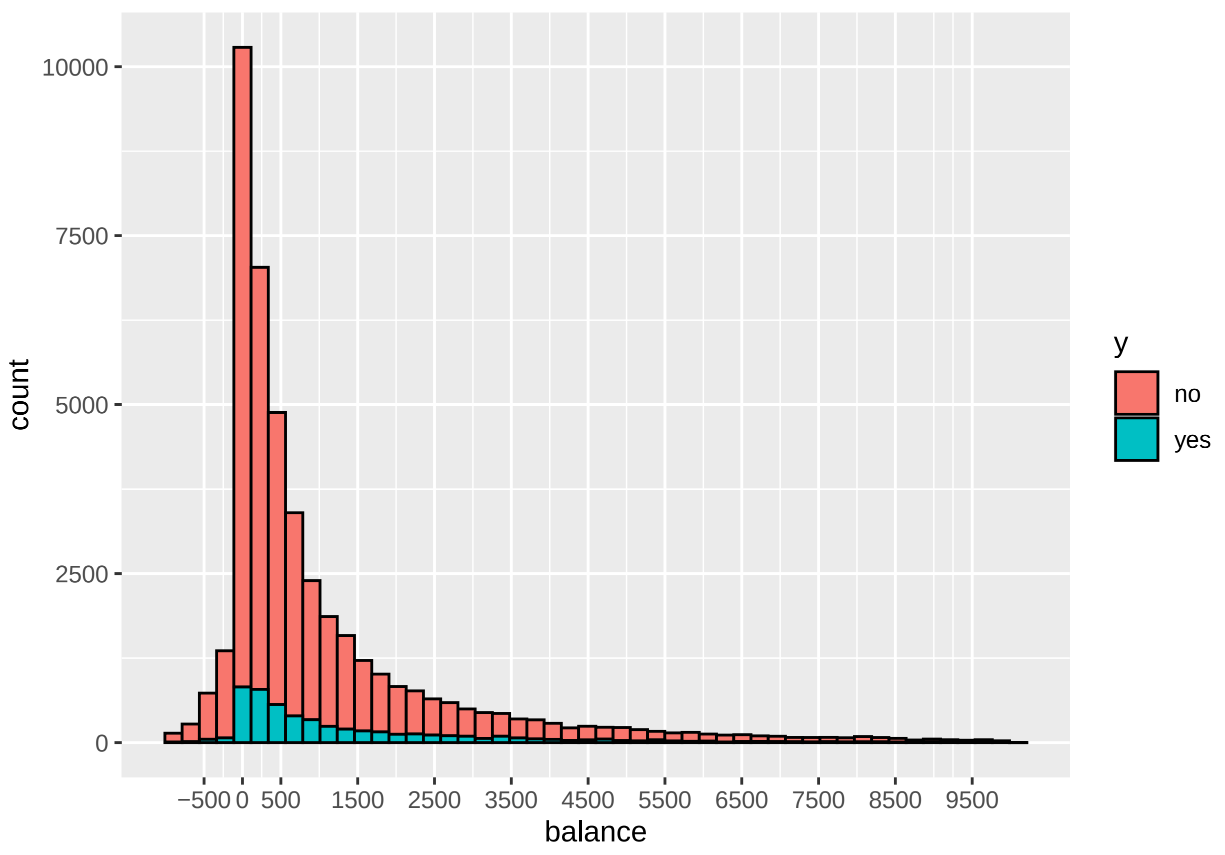
Balance

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8019      72     448    1362    1428  102127
```

The minimum balance in the data is -8019 euros, while the maximum balance is 102127 euros.

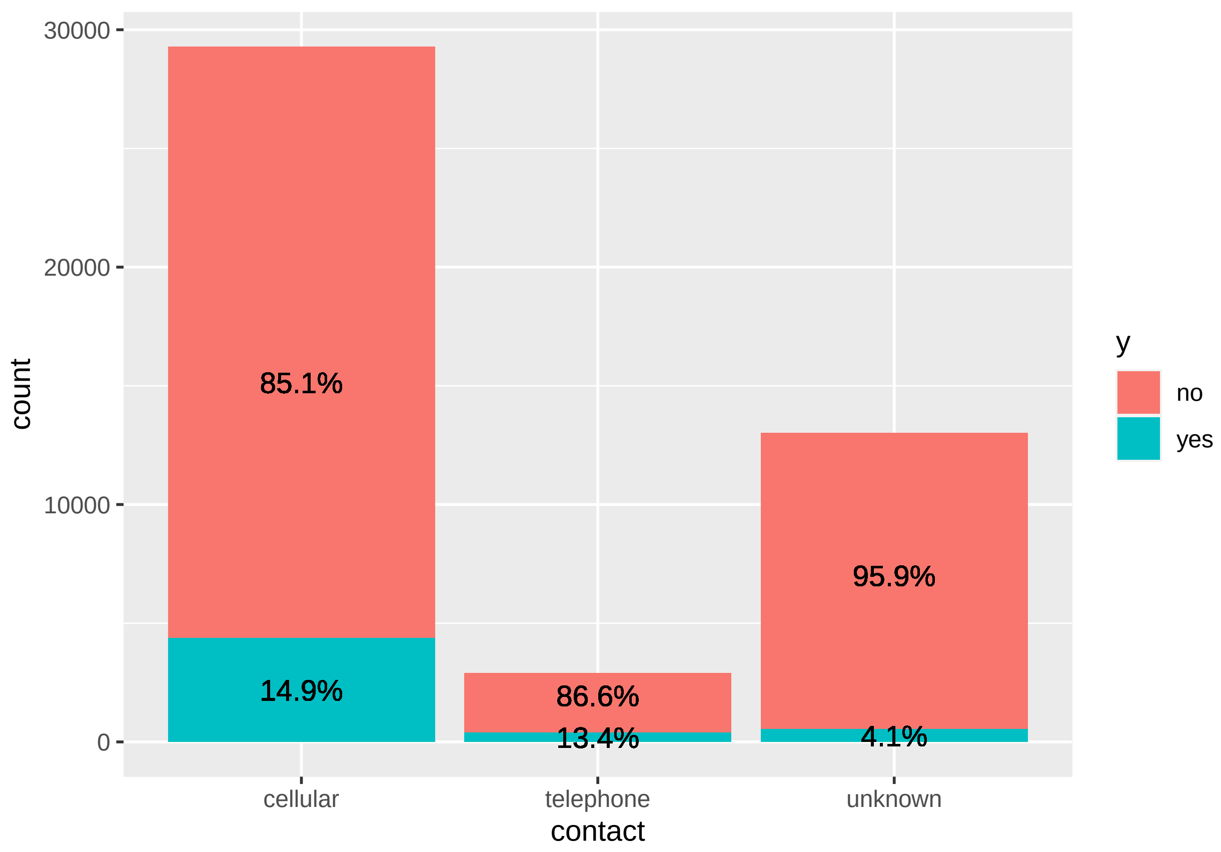


We observe that most balances are between around -1000 euros to 10000 euros so we zoom a bit more in the graph below.

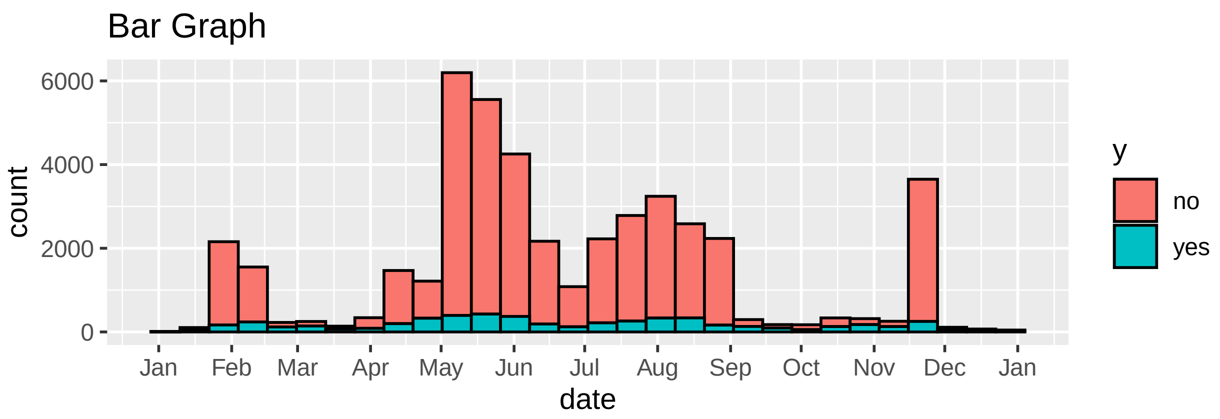


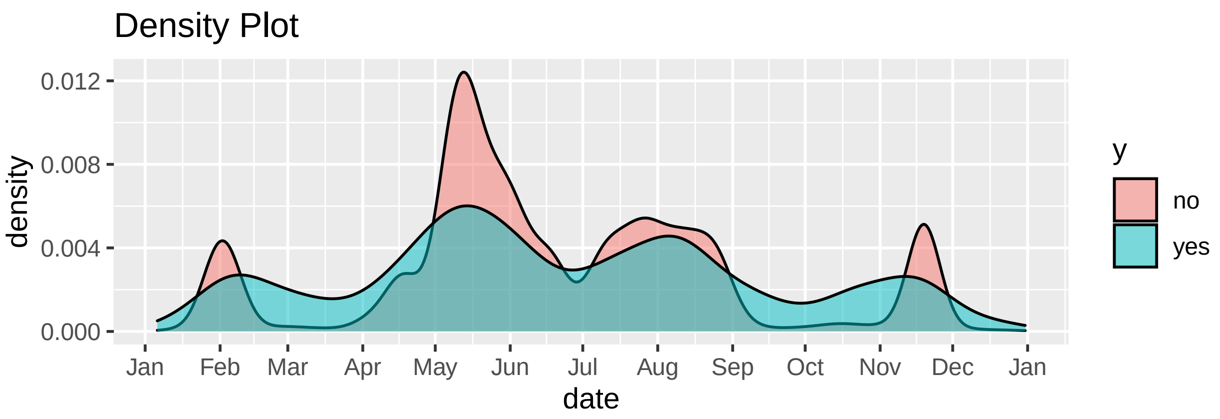
As observed from the graph, balanced of the clients are mostly between -500 euros to 1500 euros. Summary statistic tells us that the median is 448 euros.

Contact Details



Contact
Based from the graph, most of the clients contacted was by cellular. Most clients who also subscribed were also contacted by cellular, having as well the highest proportion of the group size that subscribed (14.9%).





Date

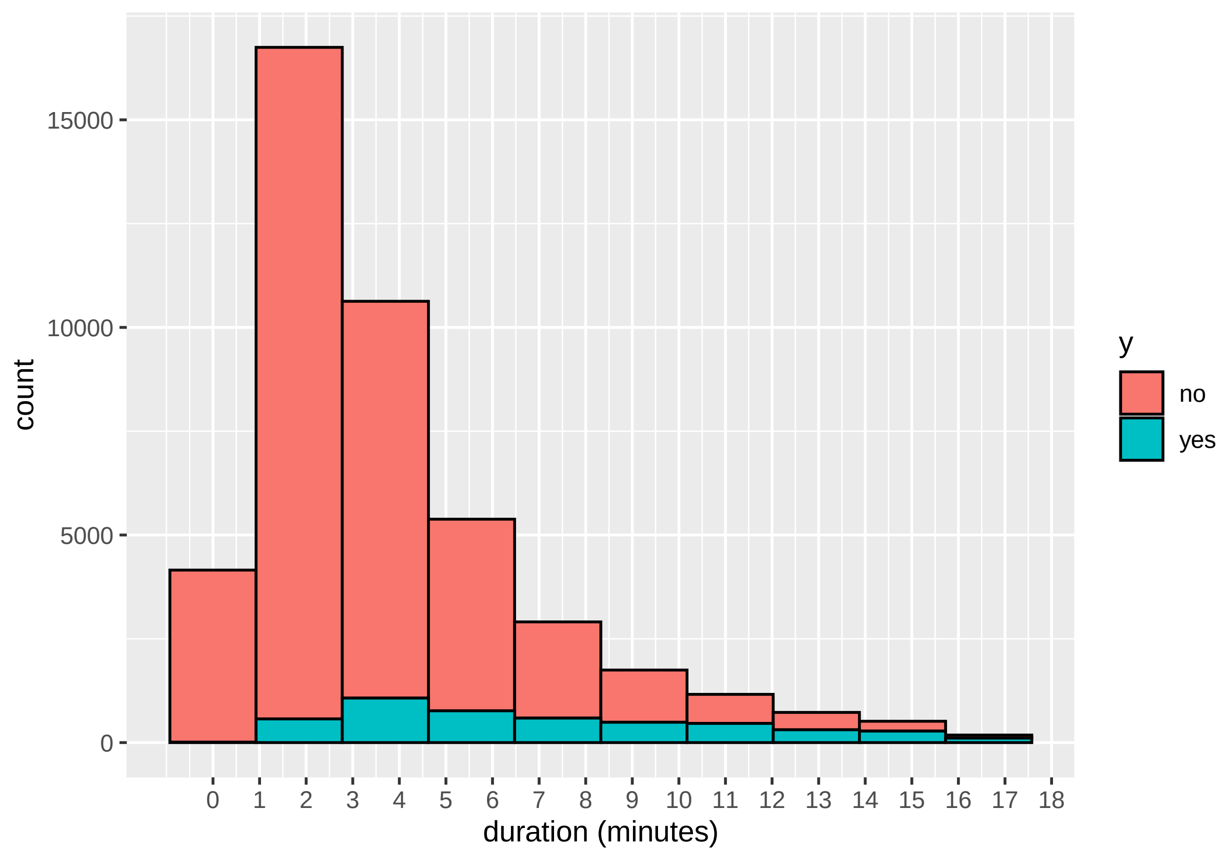
It seems that most of the clients are contacted between May to June. From the density plot, it can be seen that most subscribers are also within those months, followed by from July to September.

```
# Below we add date to the dataset
dat <- dat %>%
  mutate(date = as.Date(paste0(month, day), "%B%d")) %>%
  select(-month, -day)
```

```
summary(dat$duration)
```

Duration

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     0.0   103.0   180.0   258.2   319.0  4918.0
```



From the bar graph, it can be observed that most calls are between 2 minutes. But call above 3 minutes look to have higher rate of client subscription than below 2 mins.

Campaign Details

```
summary(dat$campaign)
```

Campaign

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.764   3.000  63.000
```

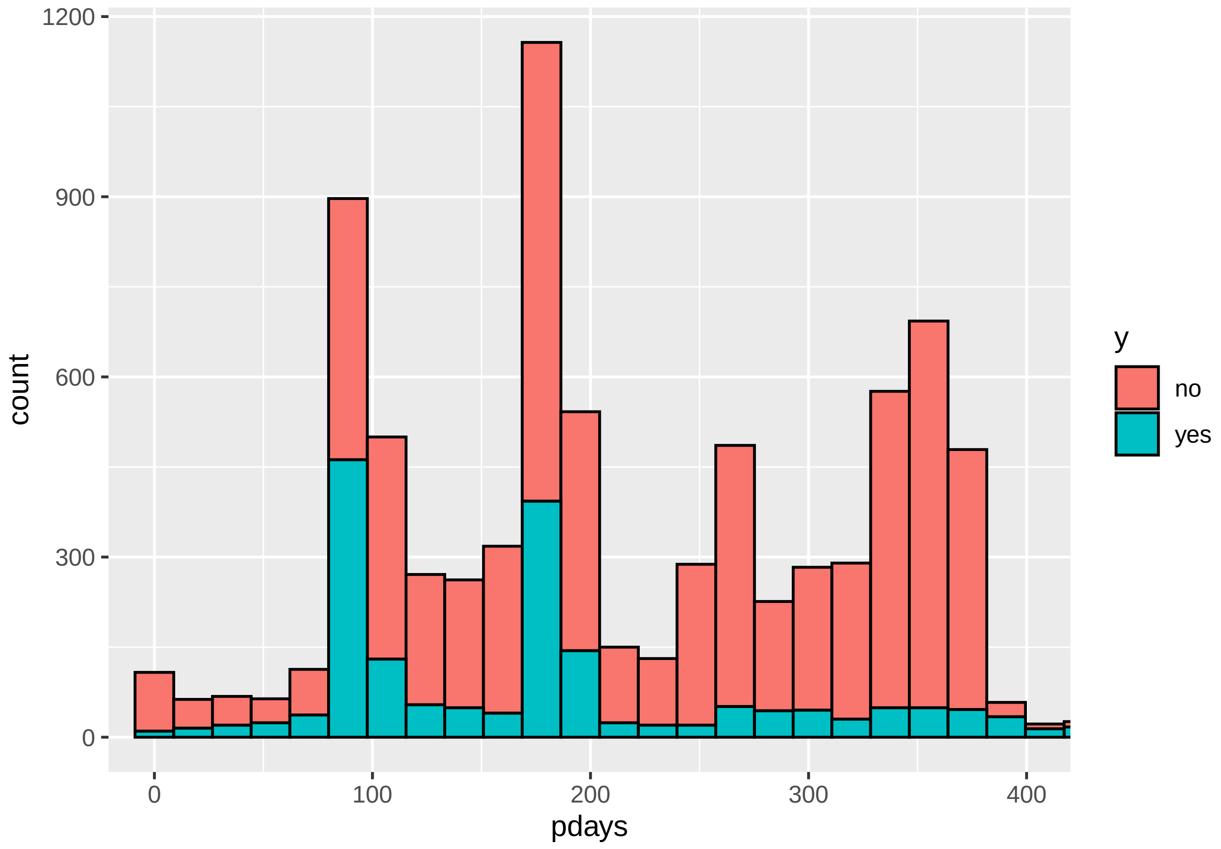
```
sum(dat$campaign)
```

```
## [1] 124956
```

A total of 124956 contacts were made. The median number of contacts per client is 2.

####pdays

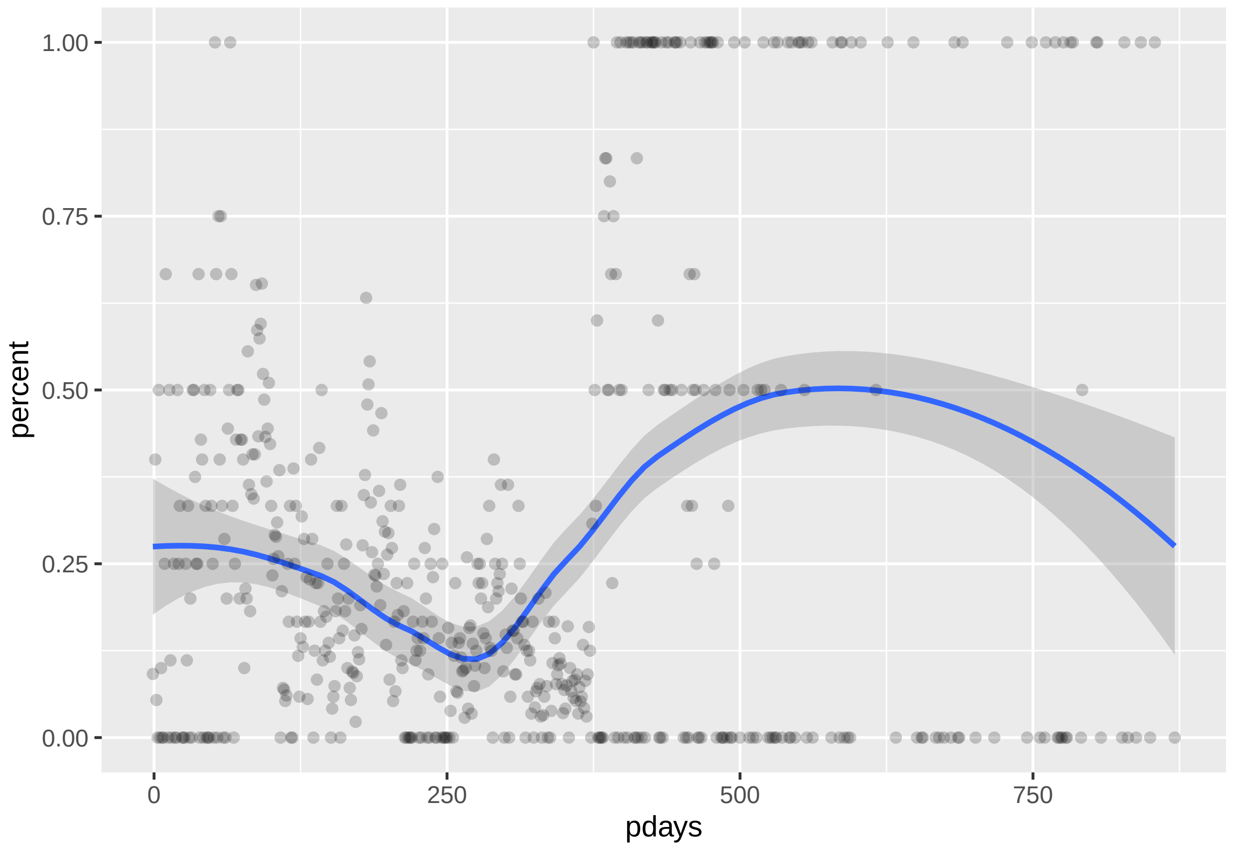
```
dat %>%
  filter(pdays > 0) %>%
  ggplot(aes(x = pdays, fill =y)) +
  geom_histogram(color = "black", bins = 50) +
  scale_x_continuous(breaks = c(0,100,200,300,400)) +
  coord_cartesian(xlim = c(-1, 400))
```



Clients contacted just around less than 100 days since last contact were the most common among clients that were previously contacted. They also look to have highest subscriber to non-subscriber ratio.

It does not look like there is a significant pattern shown in pdays so we explore further the proportion of subscribers in the below graph.

```
dat %>%
  group_by(pdays) %>%
  summarize(percent = mean(y == "yes")) %>%
  ggplot(aes(pdays, percent)) +
  geom_smooth() +
  geom_point(alpha = 0.2)
```

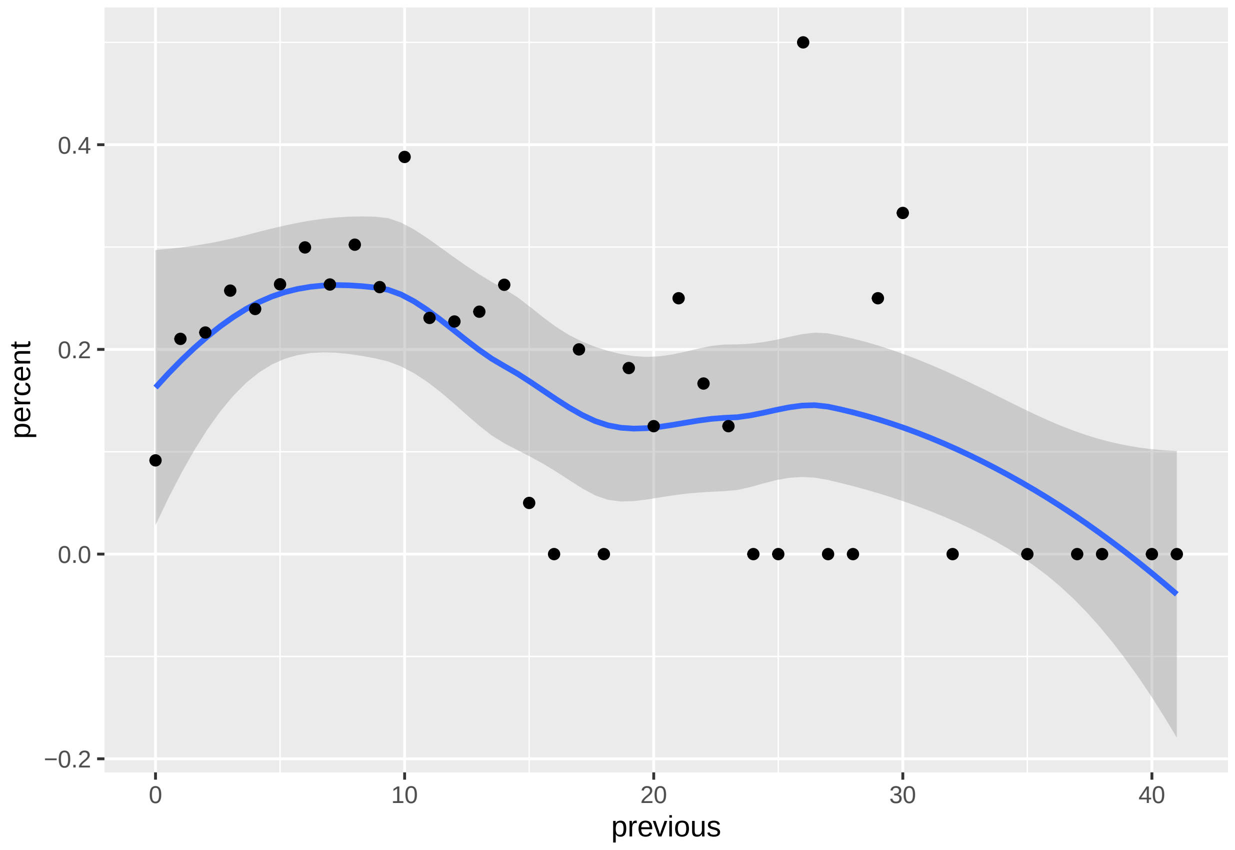


Upon inspection, pdays do not seem to uncover a much meaningful insight or possible generalization, thus, this will not be used in building the algorithm.

```
dat %>%
  group_by(previous) %>%
  filter(previous < 50) %>%
  summarize(percent = mean(y == "yes")) %>%
  ggplot(aes(previous, percent)) +
  geom_smooth() +
  geom_point()
```

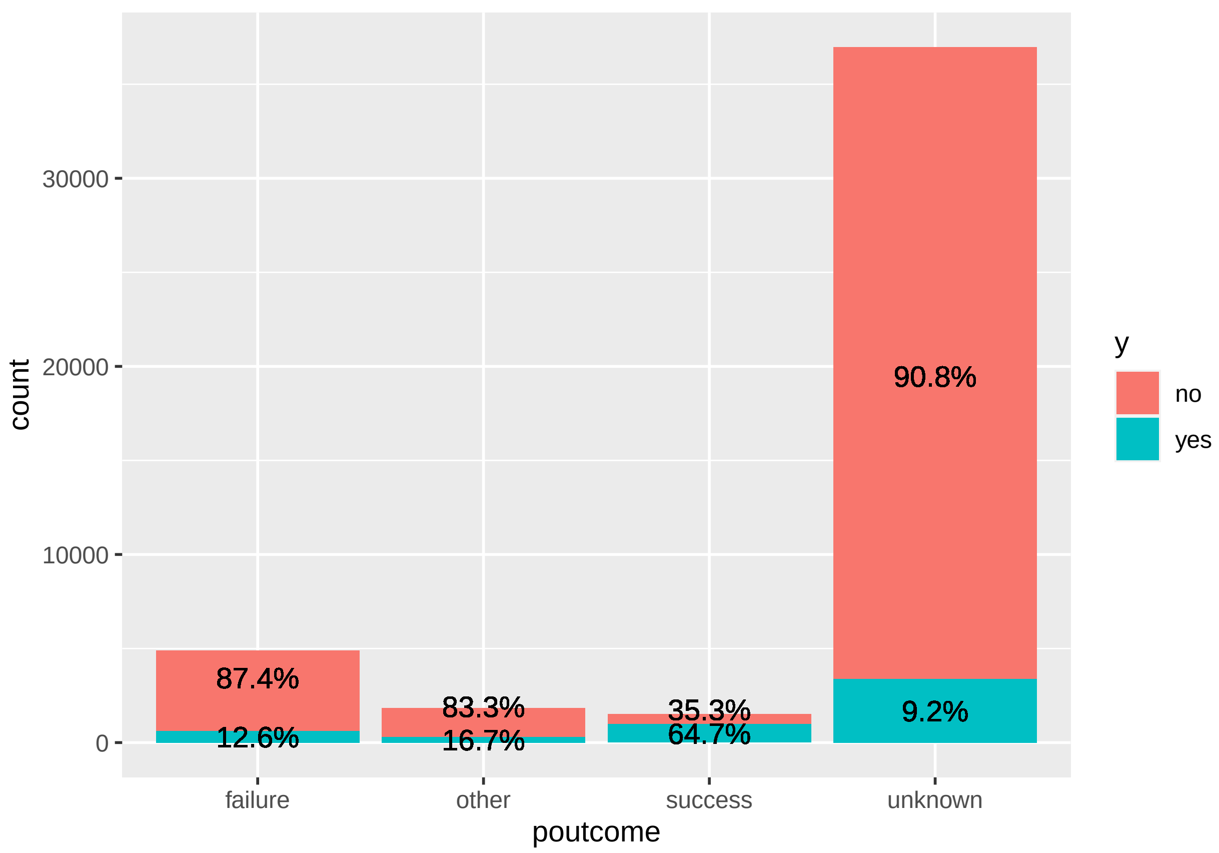
previous

```
## ‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’
```



Upon inspection, it looks like the more number of contacts performed before this campaign and for this client, the less likely they will subscribe. But the larger previous numbers vary very wisdely and looks to show a lot of noise. Thus, this will not be used.

```
dat %>%
  group_by(poutcome) %>%
  mutate(yesnum = sum(y == "yes"),
         prop = ifelse(y == "yes", sum(y == "yes")/n(), sum(y == "no")/n()),
         count = ifelse(y == "yes", sum(y == "yes"), sum(y == "no") + sum(y == "yes") + 2000)) %>%
  ungroup() %>%
  ggplot(aes(x = poutcome, fill =y)) +
  geom_bar() +
  geom_text(aes(label = scales::percent(prop),
                y = count/2,
                group = y),
            vjust = 0.5,
            size = 4) +
  ylab("count")
```



poutcome
Based from the graph, most of the clients had unknown outcome from previous marketing campaign. But upon inspection, a successful outcome from previous marketing campaign shows a better chance of their subscription for a new campaign. Also,a previous failure shows the worst chance that they will subscribe.

To finalize, based from the data exploration, all input variables will be used to train the algorithm except for pdays and previous.

Modeling

In this section, we create an algorithm to predict if the client will subscribe a term deposit. The input variables will be used to train the algorithm.

First we divide the dataset into training and test sets with the code below. 20% of the data is the test set and the other 80% is the training set. The ratio 80/20 ratio was chosen, as opposed to 70/30 or 60/40, since the dataset is fairly large.

```
test_index <- createDataPartition(y = dat$y, times = 1, p = 0.2, list = F)
train_set <- dat[-test_index,]
test_set <- dat[test_index,]
```

First, logistics regression is done below as a benchmark to compare as to whether a more complex model will be necessary.

```
train_glm <- train(y ~ age + job + marital + education + default + balance + housing + loan + date +
                     contact + duration + campaign + poutcome, method = "glm", data = train_set,
                   trControl = trainControl(method="cv", number = 5))
train_glm
```

Logistics Regression

```
## Generalized Linear Model
##
## 36168 samples
##    13 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 28934, 28935, 28935, 28933, 28935
## Resampling results:
##
##   Accuracy   Kappa
##   0.9008239  0.3896622
```

The accuracy is fairly good with 90.08% accuracy.

Random Forest Below we run train a random forest algorithm. First we see what variables give highest importance and see if we can reduce the predictors. Also we use ntree = 150 and nsamp = 5000 to speed up runnnig the code for productivity purposes in tuning. 5-fold cross validation is also used.

```
train_rf <- train(y ~ age + job + marital + education + default + balance + housing + loan + date +
                           contact + duration + campaign + poutcome, method = "rf", data = train_set,
                         ntree = 150,
                         trControl = trainControl(method="cv", number = 5),
                         tuneGrid = data.frame(mtry = c(1, 5, 10, 25, 50, 100)),
                         nsamp = 5000)
train_rf
```

```
## Random Forest
##
## 36168 samples
##    13 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 28935, 28935, 28935, 28934, 28933
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     1   0.8830181  0.0000000
##     5   0.9035059  0.4226907
##    10   0.9041141  0.4682051
##    25   0.9056625  0.4962750
##    50   0.9054412  0.4973678
##   100   0.9050541  0.4953621
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 25.
```

Below is the variable importance values:

```
varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 29)
##
##                    Overall
## duration           100.000
## date                60.794
## balance             39.384
## age                 34.332
## poutcomesuccess     30.938
## campaign            12.128
## housingyes           8.085
## poutcomeunknown      5.288
## contactunknown       4.404
## maritalmarried       3.374
## jobtechnician        3.302
## educationtertiary    2.966
## educationsecondary   2.953
## jobmanagement        2.937
## jobblue-collar       2.478
## maritalsingle        2.475
## loanyes              2.386
## jobservices          1.899
## contacttelephone     1.755
## educationunknown     1.486
```

varImp shows that only duration, date, balance, age, campaign, housing, and contact have importance values above 4. Below we rerun the code with variables having importance values above 4 and see how it affects accuracy.

```
train_rf <- train(y ~ age + balance + housing  + date + contact + duration + campaign,
                        method = "rf",
                        data = train_set,
                        ntree = 150,
                        trControl = trainControl(method="cv", number = 5),
                        tuneGrid = data.frame(mtry = c(1, 5, 10, 25, 50, 100)),
                        nsamp = 5000)
train_rf
```

```
## Random Forest
##
## 36168 samples
##     7 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 28934, 28934, 28935, 28934, 28935
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     1   0.8829905  -5.525206e-05
##     5   0.9002157   4.616313e-01
##    10   0.8997180   4.657978e-01
##    25   0.8992480   4.626309e-01
##    50   0.8994692   4.633287e-01
##   100   0.8990544   4.614380e-01
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

```
varImp(train_rf)
```

```
## rf variable importance
##
##                  Overall
## duration         100.000
## date              72.755
## balance           48.612
## age               41.369
## campaign          12.564
## housingyes         5.860
## contactunknown     4.547
## contacttelephone   0.000
```

Accuracy is a bit worse compared to using all variables. Below we try with variables having importances above 10 if accuracy is better:

```
train_rf <- train(y ~ age + balance  + date + duration + campaign,
                        method = "rf",
                        data = train_set,
                        ntree = 150,
                        trControl = trainControl(method="cv", number = 5),
                        tuneGrid = data.frame(mtry = c(1, 5, 10, 25, 50, 100)),
                        nsamp = 5000)
train_rf
```

```
## Random Forest
##
## 36168 samples
##     5 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 28934, 28934, 28935, 28934, 28935
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     1   0.8888243  0.1995285
##     5   0.8956259  0.4262144
##    10   0.8964830  0.4317574
##    25   0.8958194  0.4298505
##    50   0.8954323  0.4251958
##   100   0.8961789  0.4265875
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.
```

Accuracy is worse now with lesser predictors, thus we stick with using all predictors in the original algorithm.

```r
train_rf <- train(y ~ age + job + marital + education + default + balance + housing + loan + date +
                           contact + duration + campaign + poutcome, method = "rf", data = train_set,
                         ntree = 150,
                         trControl = trainControl(method="cv", number = 5),
                         tuneGrid = data.frame(mtry = c(1, 5, 10, 25, 50, 100)),
                         nsamp = 5000)
train_rf
```

```
## Random Forest
##
## 36168 samples
##    13 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 28934, 28935, 28934, 28935, 28934
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     1   0.8830181  0.0000000
##     5   0.9039206  0.4227932
##    10   0.9058285  0.4785839
##    25   0.9048055  0.4953314
##    50   0.9040313  0.4919336
##   100   0.9048055  0.4963004
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 10.
```

```r
data.frame(Method = "Logistic Regression", Accuracy = train_glm$results$Accuracy) %>%
  bind_rows(data.frame(Method = "Random Forest", Accuracy = max(train_rf$results$Accuracy)))
```

```
##                Method  Accuracy
## 1 Logistic Regression 0.9008239
## 2       Random Forest 0.9058285
```

Upon checking, using random forest has a better accuracy than just using logistics regression. Thus we fit the final model using random forest.

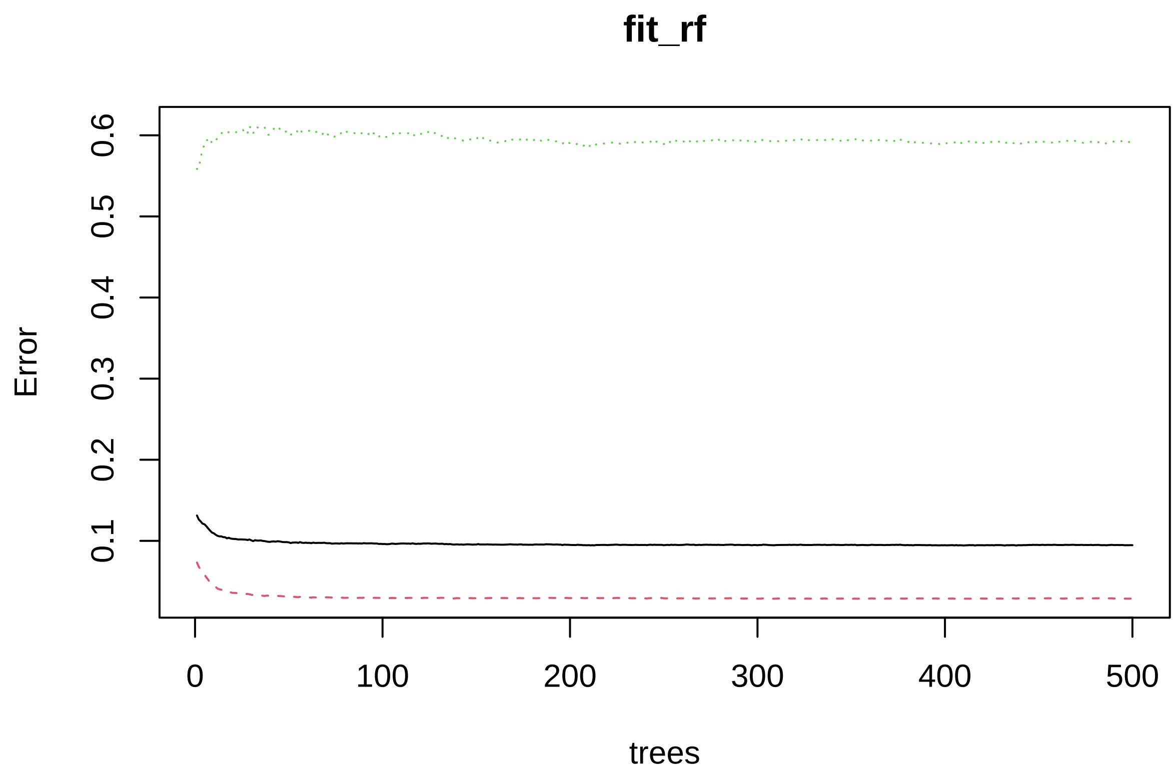
Fitting the Final Model

```
col_index_remove <- which(colnames(train_set) %in% c("pdays", "previous", "y"))
y_col_index <- which(colnames(train_set) == "y")

fit_rf <- randomForest(train_set[,-col_index_remove], factor(train_set[,y_col_index]),
                       minNode = train_rf$bestTune$mtry)
y_hat_fitrf <- predict(fit_rf, test_set[,-col_index_remove])
confusionMatrix(y_hat_fitrf, factor(test_set[,y_col_index]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7759  632
##        yes  226  426
##
##                Accuracy : 0.9051
##                  95% CI : (0.8989, 0.9111)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 9.269e-12
##
##                   Kappa : 0.4491
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9717
##             Specificity : 0.4026
##          Pos Pred Value : 0.9247
##          Neg Pred Value : 0.6534
##              Prevalence : 0.8830
##          Detection Rate : 0.8580
##    Detection Prevalence : 0.9279
##       Balanced Accuracy : 0.6872
##
##        'Positive' Class : no
##
```

```
plot(fit_rf)
```



Plot also shows that it look like we have used enough trees to get the best result out of the random forest

Results

Random Forest performed better than logistic regression during algorithm training, thus random forest was used to train the final model. All input variables were used except for pdays and previous. The final model performed with 90.51% accuracy.

Conclusion

The goal in this project was to create an algorithm that will predict if a client will subscribe to a bank term deposit. Using all the variables except for pdays and previous, we were able to train an algorithm that can predict if a client will subscribe to a bank term deposit that performs with 90.51% accuracy when evaluated with a test set. This model may be used for improving the marketing of the bank's products. For the limitations, the algorithm may be improved with more fine tuning with the parameters which can be done with faster computing power. Testing many other algorithm models then creating an Ensemble may also yield better results.