Bank Marketing - Classification Project

Ryan Reuther

12/24/2019

Introduction

For banks, the ability to generate more money is at the heart of their business. Cash allows the banks to further invest in capital in order to raise more capital. For retail banks, a prime way to raise capital is to receive money in the form of bank term deposits from individuals. A bank term deposit is a cash investment in which an individual provides a financial institution with money. The bank invests this money, and in exchange the bank pays the individual an agreed upon percentage of their balance in the form of interest rates over a period of time, or term.

This dataset contains direct marketing campaigns from a Portuguese banking institution. The campaigns are completed through direct phone calls to users in order to generate business in the form of submitting bank term deposits. The more bank term deposits are created, the more capital the bank has to invest in.

Therein lies the goal of this project: using the dataset provided, train an algorithm to predict whether or not a use will submit a bank term deposit. Ultimately, a good model can be used by the bank in the future to make more informed decisions regarding which clients to contact in order to recieve further funding.

The original dataset contains 45211 records with a total of 17 variables. The goal of this project is to predict when a user has (y==‘yes’) or has not (y =‘no’) submitted a bank term deposit. Considering the outcome is binary and categorical, Several methods were employed in order to determine an ideal classification model:

- Tree-based
- Logistic Regression
- Quantitative Discriminant Analysis
- Linear Discriminant Analysis
- K-Nearest Neighbors
- Random Forest
- Ensemble methods

In the end, due to its individual performance and relative quality in predicting if a user submitted a bank deposit, a Random Forest method was used for the final model, providing an Accuracy of around 83.3%, a Sensitivity (True Positive Rate) or 57.54%, and a Specificity (True Negative Rate) of 90.9%

Method and Analysis

Data Exploration and Examining Data Cleaning Methods Several different models were developed for this classifiation model. Considering the outcome is binary (yes or no), evaluating accuracy alone is not a good enough metric of the performance of the algorithm; the models were evaluated based off of a confusion matrix in order to review their Accuracy, True Positive Rate, and True Negative Rate. Using these values allows us to more accurately understand when the model is accurately predicting whether or not a user has submitted a bank deposit.

Accuracy is the proportion predicted y values that equal the actual value of y. True Positive Rate, or sensitivity, is the conditional probability that the model predicts "yes" when y actually equals "yes". The True Negative Rate, or specificity, is the probability that the model predicts "no" when y actually is "no". While maximizing all three metrics would be ideal, generating a model to optimize the true positive rate should be prioritized, as a model that does this would be good at identifying the best customers to campaign for.

To begin, we'll load in the libaries and raw csv file. We'll examine the structure of the raw dataset.

```
# Automatically downloads data from url
file_name <- "bank-full.csv"
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip"
temp_file <-  tempfile()
download.file(url, temp_file)
raw_data <- read.csv2(unz(temp_file, file_name))
str(raw_data)
```

```
## 'data.frame':    45211 obs. of  17 variables:
##  $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 5 10 3 2 12 5 5 3 6 10 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 3 2 2 3 2 3 1 2 3 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 2 2 4 4 3 3 3 1 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 2 1 1 ...
##  $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 2 2 2 1 2 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

A decent amount of predictors have a value of "unknown", so those values were set to NA.

```
raw_data2 <- read.csv2(unz(temp_file, file_name), na = c("unknown"))
unlink(temp_file)
```

The initial review of the dataset showed several predictors contained NA's. I reviewed how many records were missing one or more predictors (records with one or more NA's):

```
# dataset containing records with NA values
a <- raw_data2[!complete.cases(raw_data2),]

# Proportion of data was missing one or more records
count(a) / count(raw_data2)
```

```
##           n
## 1 0.8265466
```

More than 82% of records contain one or more NA's. Additionally, upon exploring the column we're predicting, y, I noticed most y records were set to 0/no:

```
#prevalence of y==1 or y=="yes"
prop <- data.frame(Method = "Original", Probability = prop.table(table(raw_data2$y))[2])
prop
```

```
##       Method Probability
## yes Original   0.1169848
```

This shows that there is a major class imbalance in this dataset. Considering this, and the presence of NA's in the dataset, the dataset needed to be cleaned up. We have two options for doing this:

1 - Impute random categorical values based on each categorical predictor's probability (probability-based random assignment) 2 - Removing all NA values.

```
# IMPUTATION METHOD
# Find columns that have NA's
colnames(raw_data)[colSums(is.na(raw_data2)) > 0]
```

```
## [1] "job"       "education" "contact"   "poutcome"
```

```
# Use data without NA's to gather Medians/Modes for those columns
cleaned <- raw_data2[complete.cases(raw_data2),]
N <- count(cleaned)$n

#Gather the prevalence of each unique element for job, education, contact, and poutcome
# JOB
pctjobs <- prop.table(table(cleaned$job))
jobnames <- sort(as.character(unique(cleaned$job)))

# EDUCATION
pcteducation <- prop.table(table(cleaned$education))
educationnames <- sort(as.character(unique(cleaned$education)))

# CONTACT
pctcontact <- prop.table(table(cleaned$contact))
contactnames <- sort(as.character(unique(cleaned$contact)))

#POUTCOME
pctpoutcome <- prop.table(table(cleaned$poutcome))
poutcomenames <- sort(as.character(unique(cleaned$poutcome)))

cleaned_data <- raw_data2 %>% mutate(
  job = factor(ifelse(is.na(job), sample(jobnames, 1, replace = TRUE, prob = pctjobs),
                      as.character(job))),
  education = factor(ifelse(is.na(education), sample(educationnames, 1, replace = TRUE,
                                                     prob = pcteducation),as.character(education))),
  contact = factor(ifelse(is.na(contact), sample(contactnames, 1, replace = TRUE,
                                                 prob = pctcontact), as.character(contact))),
  poutcome = factor(ifelse(is.na(poutcome), sample(poutcomenames, 1, replace = TRUE,
                                                   prob = pctpoutcome), as.character(poutcome)))
)
```

The imputation method preserves all records from the original dataset. Preserving this amount could be useful in training a more accurate algorithm. However, the issue with this approach is that because no records were removed, the class imbalance remains.

```
# imputation method... cleaned_data => train and test
imp_cleaned <- cleaned_data %>%
  mutate(y = factor(ifelse(y=="no",0,1)),
         poutcome = factor(ifelse(poutcome=="success",1,0)),campaign = as.integer(campaign))

# Add the prevalence of y == yes/1 to the prevalence table
prop <- bind_rows(prop,
                  data_frame(Method = "Imputation Method",
                             Probability = prop.table(table(imp_cleaned$y))[2]))
prop
```

```
##              Method Probability
## 1          Original   0.1169848
## 2 Imputation Method   0.1169848
```

Less than 12% of records are subscribers. This method did not alleviate the concerns about class imbalance. This makes sense because we did not remove records nor change y values. The removal method, on the other hand, slightly fixed the class imbalance.

```
# Create a dataset with all records not containing NA's
rem_cleaned <- raw_data2[complete.cases(raw_data2),]

prop <- bind_rows(prop,
                  data_frame(Method = "Removal Method",
                             Probability = prop.table(table(rem_cleaned$y))[2])
)
prop
```

```
##              Method Probability
## 1          Original   0.1169848
## 2 Imputation Method   0.1169848
## 3    Removal Method   0.2277480
```

The removal method reduced the number of records in the dataset yet improved the class imbalance, as more than 22% of records in the removal dataset are subscribers. This indicates that deposit accounts (y == yes/1) are more likely to put in all their information than non-deposit accounts.

To determine which data cleaning method was best for the model, the two separate models were trained through Binary Logistic Regression. The accuracy, specificity and selectivity between the predicted values and the test set was then used to evaluate the performance of the two cleaning methods.

```
# IMPUTATION Algo Testing
imp_test_index <- createDataPartition(y = imp_cleaned$y ,
                                      times = 1, p = 0.1, list = FALSE)
imp_train <- imp_cleaned[-imp_test_index,]
imp_test <- imp_cleaned[imp_test_index,]
imp_fit <- train(y~., data = imp_train, method = "glm", family="binomial")
imp_y_hat <- predict(imp_fit, imp_test)
imp_con <- confusionMatrix(imp_y_hat,imp_test$y, positive = "1")
```

```r
comp <- data.frame(Type = "Imputation",
                   Accuracy = imp_con$overall["Accuracy"],
                   Sensitivity = imp_con$byClass["Sensitivity"],
                   Specificity= imp_con$byClass["Specificity"]
)

# REMOVAL Algo Testing
rem_cleaned <- raw_data2[complete.cases(raw_data2),]
rem_cleaned <- rem_cleaned %>% mutate(y = factor(ifelse(y=="no",0,1)))
rem_test_index <- createDataPartition(y = rem_cleaned$y , times = 1, p = 0.1, list = FALSE)
rem_train <- rem_cleaned[-rem_test_index,]
rem_test <- rem_cleaned[rem_test_index,]
rem_fit <- train(y~., data = rem_train,method = "glm", family="binomial")
rem_y_hat <- predict(rem_fit, rem_test)
rem_con <- confusionMatrix(rem_y_hat, rem_test$y, positive = "1")
comp <- bind_rows(comp,
                  data.frame(Type = "Removal",
                             Accuracy = rem_con$overall["Accuracy"],
                             Sensitivity = rem_con$byClass["Sensitivity"],
                             Specificity= rem_con$byClass["Specificity"])
)
```

The confusion matrices for the two cleaning methods have been complete. The tables will now be reviewed.

```r
imp_con
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3898  340
##          1   95  189
##
##                Accuracy : 0.9038
##                  95% CI : (0.8948, 0.9122)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 4.497e-06
##
##                   Kappa : 0.4173
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3573
##             Specificity : 0.9762
##          Pos Pred Value : 0.6655
##          Neg Pred Value : 0.9198
##              Prevalence : 0.1170
##          Detection Rate : 0.0418
##    Detection Prevalence : 0.0628
##       Balanced Accuracy : 0.6667
##
##        'Positive' Class : 1
##
```

The sensitivity, or True Positive Rate, is very low. This can be attributed to the class imbalance, where there was a low prevalence of y == "yes" records.

```
comp
```

```
##         Type  Accuracy Sensitivity Specificity
## 1 Imputation 0.9038036   0.3572779   0.9762084
## 2    Removal 0.8420382   0.5530726   0.9273927
```

The imputation method had a higher accuracy and specificity, which at first glance would indicate this was the best data cleaning method. However, with regard to the Bank, they would be better off knowing when a user is a good fit for the subscription plan, so the sensitivity is most ideal metric to base the model's true effectiveness on.

Althought the removal method has fewer records to train on, the removal method has an improved sensitivity rating. This can be attributed the removal set having a greater prevalence of subscription users (y = yes/1), which indicates less class imbalance. Therefore, the removal method will be used to clean the data.

Final Data Cleaning The dataset was split up twice. The first time was to separate it into a training subset and a validation subset, the latter of which will be used to evaluate the final model.

```
set.seed(123)
data <- raw_data2[complete.cases(raw_data2),] %>%
  mutate(y= factor(ifelse(y == "yes",1,0)), poutcome = factor(poutcome))
test_index <- createDataPartition(y = data$y , times = 1, p = 0.1, list = FALSE)
algotraining <- data[-test_index,]
validation <- data[test_index,]

set.seed(123)
test_index <- createDataPartition(y = algotraining$y , times = 1, p = 0.1, list = FALSE)
train <- algotraining[-test_index,]
test <- algotraining[test_index,]
```

Further Data Exploration Now that the data is cleaned, we will continue exploring the new set to uncover new findings. To get a general understanding of the data, correlations between numercial features and the predictor were examined.

```
h <- hetcor(data)
col <- colnames(h$correlations[])
ycorr <- data.frame("y-Correlation" = h$correlations[,"y"])
ycorr[order(ycorr$y.Correlation), , drop = FALSE]
```

```
##           y.Correlation
## housing     -0.51712054
## loan        -0.29153072
## default     -0.24429370
## pdays       -0.20160548
## campaign    -0.15654211
## contact     -0.03117624
## day          0.04015413
## month        0.05809197
```

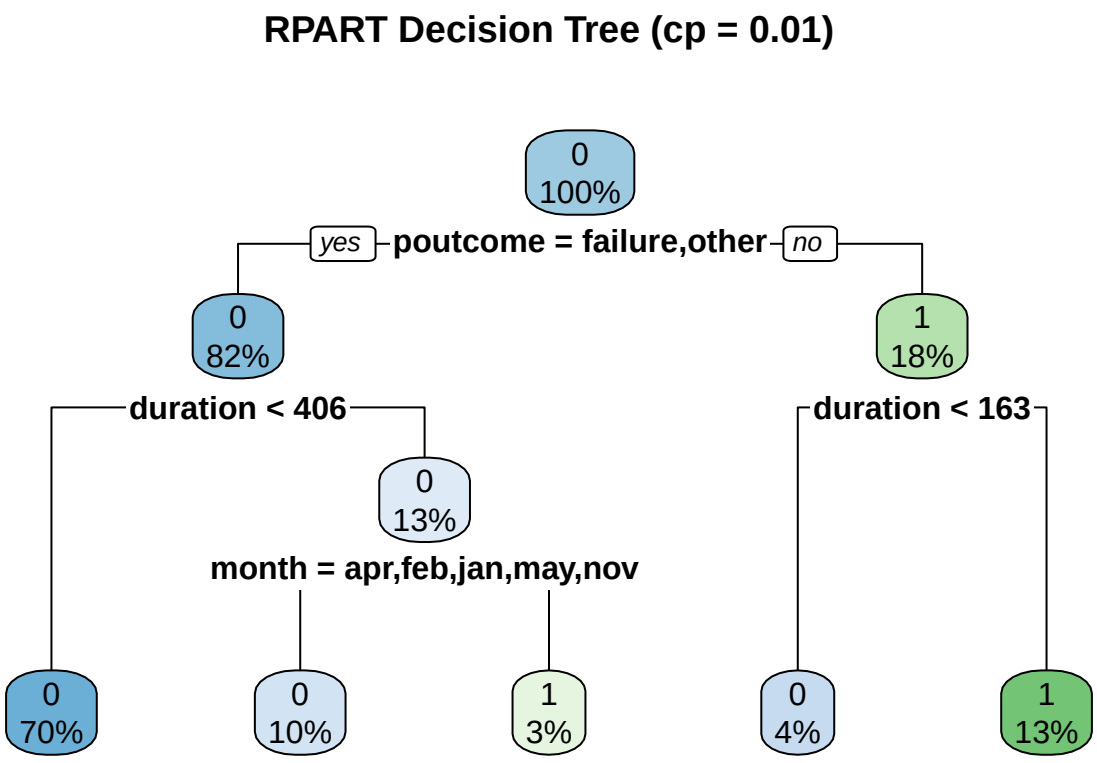
```
## marital      0.08135119
## age          0.08639151
## balance      0.10318587
## job          0.10343975
## education    0.17273520
## duration     0.40740556
## poutcome     0.58885652
## y            1.00000000
## previous             NA
```

Housing, duration, and poutcome, in terms of absolute values, had notable correlations with y. However, one important note regarding the duration column is that it should not be included when training predictive models. This field, which details the last contact duration in seconds, is not known until after a call is completed. Since one cannot predict how long a call will last, using this field to predict whether a user will submit a bank note is not possible. Therefore, the duration column will not be used in predicting y. This field will be incorporated into the decision tree to gather an understanding of it's importance in analyzing a potential customer after a call, however it does not hold weight in generating the predictive model.

Further information on the duration column and all 16 other predictors can be found at the following website: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing.

This dataset consists of both numeric and categorical data, so a decision tree was useful in visualizing which users traditionally sign on as subscribers.

```
rpart_fit1 <- rpart(y~., data = train, method = "class")
rpart_y_hat1 <- predict(rpart_fit1, test, type = "class")
rpart.plot(rpart_fit1,  main="RPART Decision Tree (cp = 0.01)", extra=100)
```

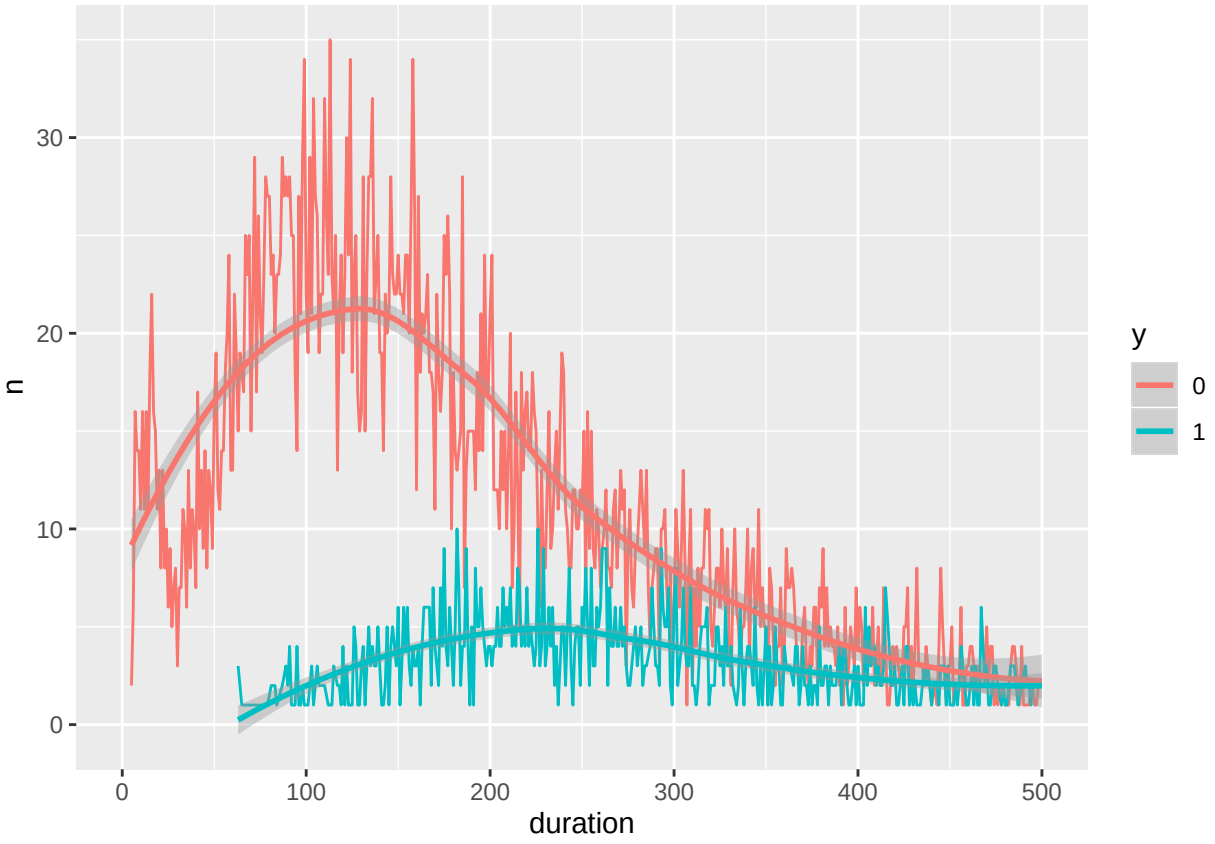


Duration, which will not be used in the algorithm, is a defining characteristic of the decision tree. Providing a proper order to provide a better understanding of how duration operates would be helpful for the bank.

```
dur <- data %>% group_by(y,duration) %>%summarize(n = n())
ggplot(dur, aes(duration, n,colour=y)) +
  geom_line()+ geom_smooth() + scale_x_continuous(limits = c(0,500))
```

```
## Warning: Removed 622 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 622 rows containing missing values (geom_path).
```



As shown above, there are many calls for non-subscribers that last a very short period of time. This is to be expected given the brief nature of certain marketing calls for non-interested parties. In other words, it is common for people that do not want a company's business to keep a call very short. In some scenarios, an individual might hang up within seconds of answering the phone. The inflection point around 30 seconds for the y==0 line further proves this point, and offers insight that if a user remains on the phone for longer than 30 seconds, further business could be conducted.

On the other hand, calls in which a user has submitted a deposit prove to be longer, which could be an indication of a worthwhile conversation regarding future business endeavors or selling the bank's services.

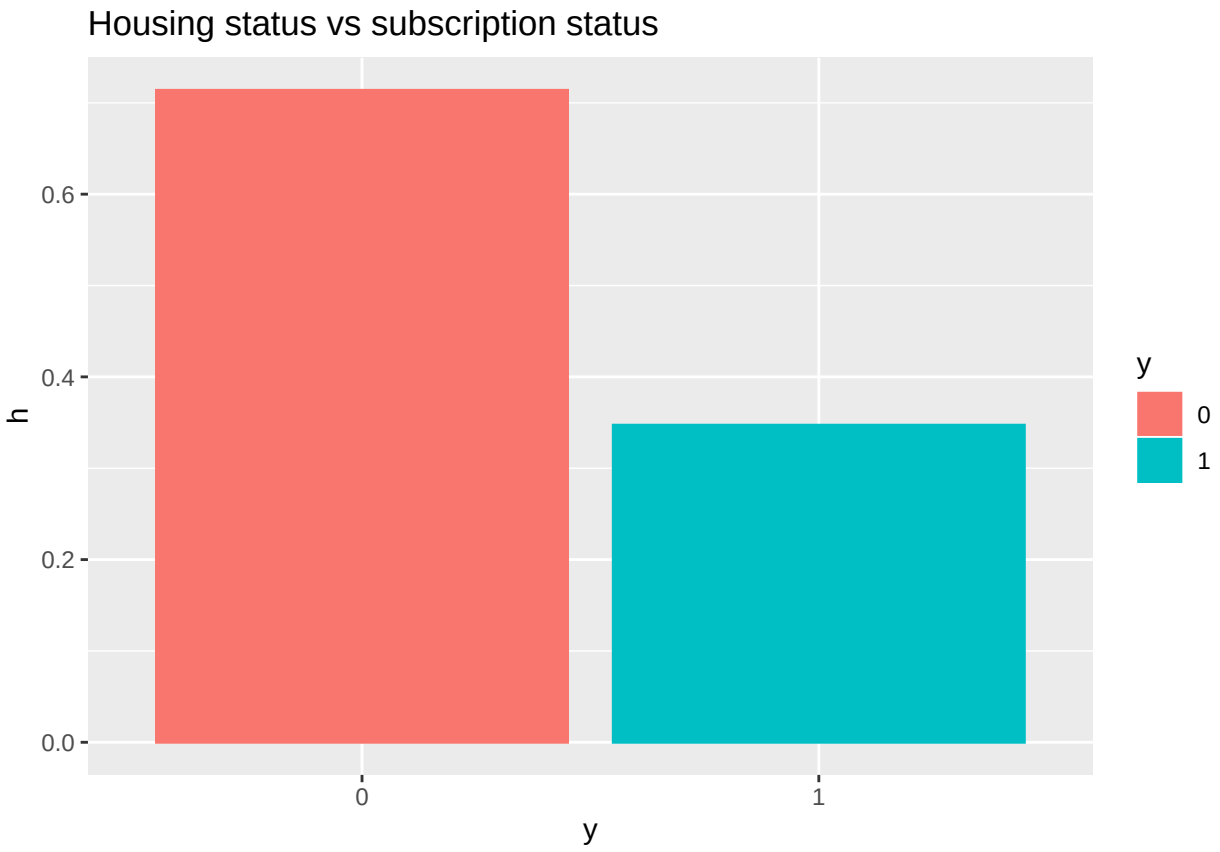
The importance of the predictors in the decision tree were reviewed, as this helps interpret which predictors could be best for developing the model.

```r
varImp(rpart_fit1)
```

```
##              Overall
## campaign    6.012301
## duration  493.085212
## housing   296.492575
## job        40.795345
## month     438.569482
## pdays     161.276826
## poutcome  484.555144
## age         0.000000
## marital     0.000000
## education   0.000000
## default     0.000000
## balance     0.000000
## loan        0.000000
## contact     0.000000
## day         0.000000
## previous    0.000000
```

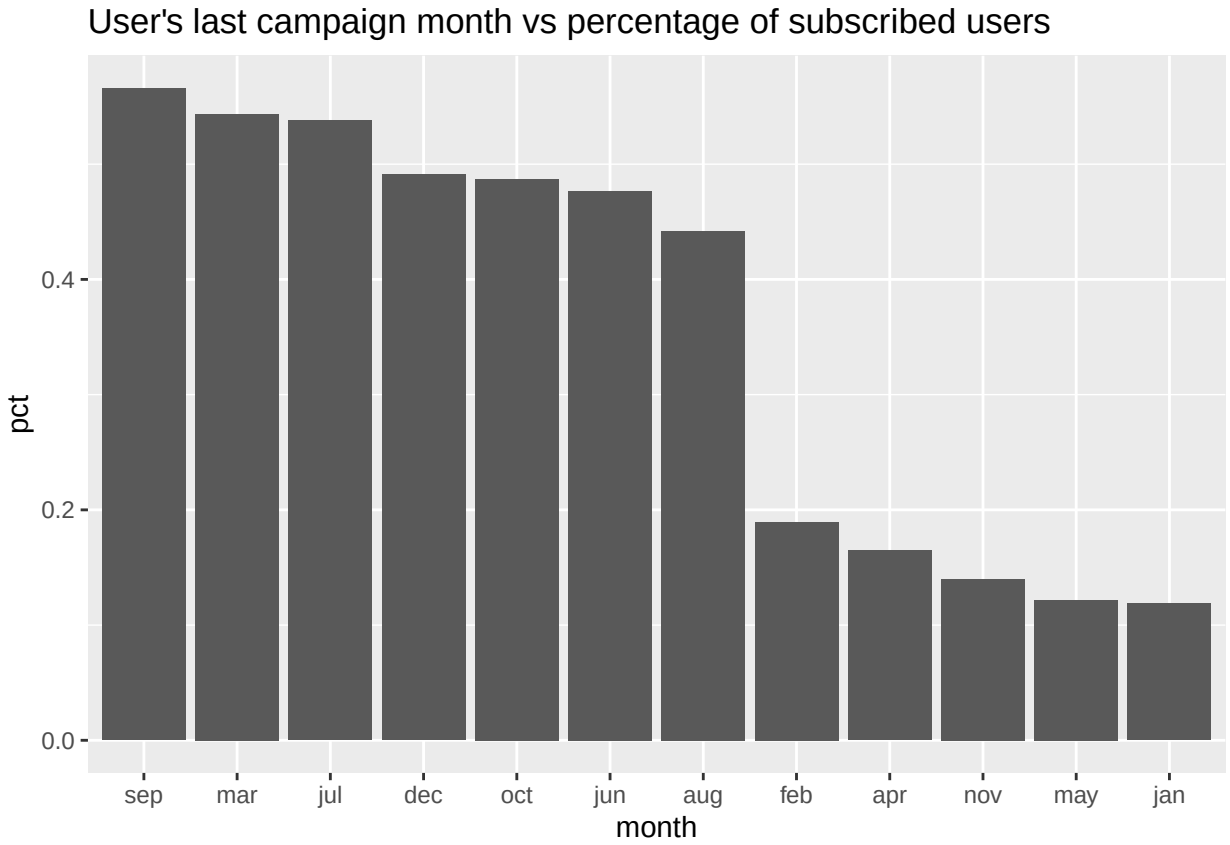
Housing was another predictor with significance:

```r
data %>% mutate(housing = ifelse(housing=="yes",1,0)) %>%group_by(y) %>%
  summarize(h = mean(housing), sd = sd(housing), n = n()) %>%
  ggplot(aes(y, h, fill=y,color = y)) + geom_bar(stat = "identity") +
  ggtitle("Housing status vs subscription status")
```



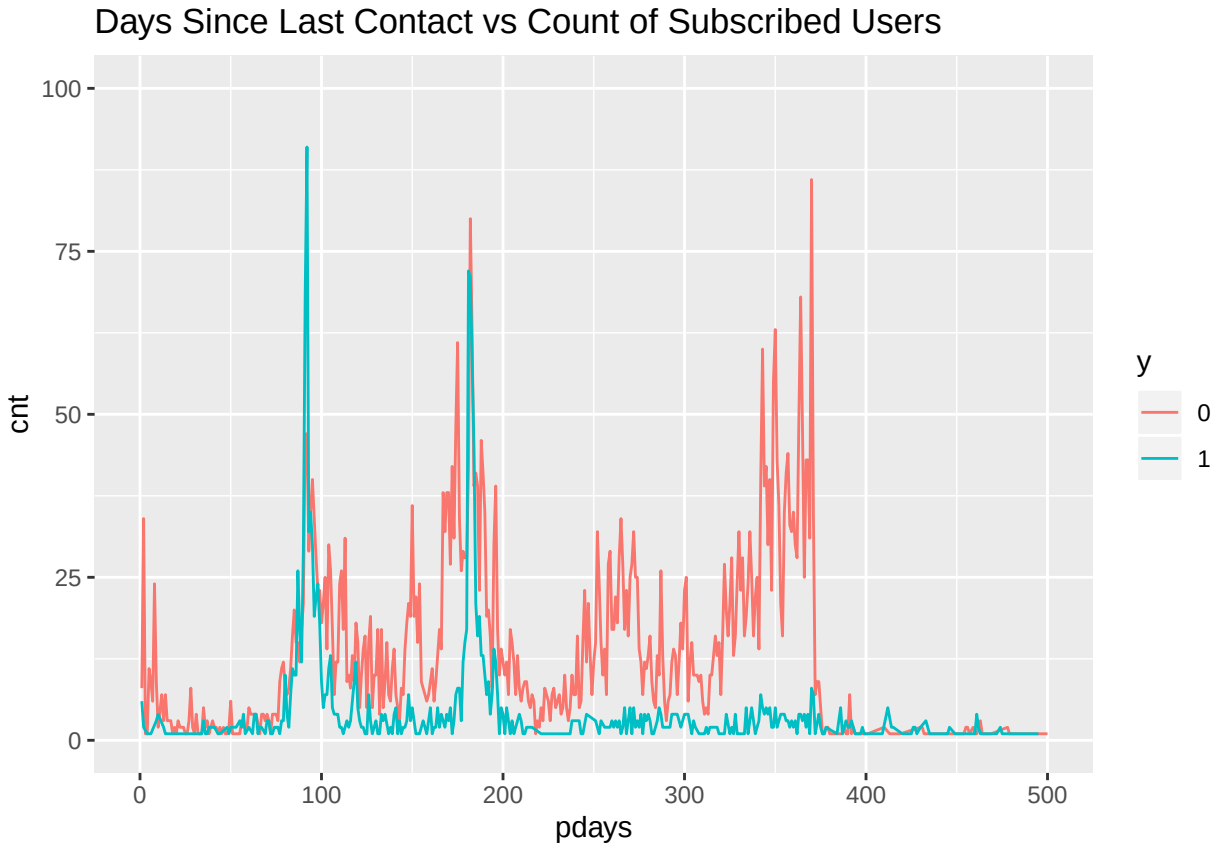
Finally, the last contact month was determined to be a defining characteristic. Certain months are proven to have a higher rate of subscribed users, or users that have submitted a deposit.

```
data %>% group_by(month) %>% summarize(pct = mean(y==1)) %>%
  mutate(month = reorder(month,desc(pct))) %>%
  ggplot(aes(month,pct)) +
  geom_bar(position = "dodge2", stat = "identity") +
  ggtitle("User's last campaign month vs percentage of subscribed users")
```



This indicates there is a discrepancy between the months in which users sign on for the paid subscription. Furthermore, this lines up with the cadence in which users are contacted or advertised to.

```
data %>% group_by(pdays,y) %>% summarize(cnt = n()) %>%
  ggplot(aes(x = pdays, y = cnt, group = y, color = y)) +
  geom_line() +  scale_x_continuous(limits = c(0, 500)) +
  scale_y_continuous(limits = c(0,100))+
  ggtitle("Days Since Last Contact vs Count of Subscribed Users")
```



This graph illustrates that customers seem to be contacted in waves. This could indicate that the marketing team already has

It is apparent that there are deviations in user prevalence for multiple predictors. Considering there are several important predictors, the training models will deal with high-dimensionality, which must be considered when determining the most efficient model.

Algorithm Training Similar to before, an RPart method was trained and added to a data frame for final review. The following models were created:

- RPART
- Logistic Regression
- Quantitative Discriminant Analysis
- Linear Discriminant Analysis
- K-Nearest Neighbors
- Random Forest
- Ensemble Methods

```
rpart_fit <- rpart(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                   data = train, method = "class")
rpart_y_hat <- predict(rpart_fit, test, type = "class")
rpart_con <- confusionMatrix(rpart_y_hat, test$y, positive = "1")

results <- data.frame(Type = "RPART",
                      Accuracy = rpart_con$overall["Accuracy"],
                      Sensitivity = rpart_con$byClass["Sensitivity"],
```

```
                      Specificity= rpart_con$byClass["Specificity"]
)
```

For Logistic Regression,The variable to predict is a is a 0 or 1, so it is Binary Logistic Regression.

```
glm_fit <- train(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                 data = train, method = "glm", family="binomial")
glm_p_hat <- predict(glm_fit, test, type = "prob")
glm_y_hat <- factor(ifelse(glm_p_hat[1] > glm_p_hat[2], 0,1))
glm_con <- confusionMatrix(glm_y_hat, test$y, positive = "1")

results <- bind_rows(results,
                     data.frame(Type = "Logistic Regression",
                                Accuracy = glm_con$overall["Accuracy"],
                                Sensitivity = glm_con$byClass["Sensitivity"],
                                Specificity= glm_con$byClass["Specificity"]
                     )
)
```

Next, a Naive Bayes model was considered. This model assumes we can estimate conditional distributions of the predictors. However, because there are several features, so we will not be using Naive Bayes model in the final algorithm. Instead, Quantitative Discriminant Analysis and Linear Discriminant Analysis, which are variants of the Naive Bayes were utilized.

Quantitative Discriminant Analysis is a form of the Naive Bayes model, and assumes the the distribution of the variable we're predicting is bivariate normal. Considering the multidimensionality of the dataset, and the class imbalance, this is not the best assumption to make.

```
qda_fit <- train(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                 data = train, method = "qda")
qda_y_hat <- predict(qda_fit, test)
qda_con <- confusionMatrix(qda_y_hat,test$y, positive = "1")
results <- bind_rows(results,
                     data.frame(Type = "Quantitative Discriminant Analysis",
                                Accuracy = qda_con$overall["Accuracy"],
                                Sensitivity = qda_con$byClass["Sensitivity"],
                                Specificity= qda_con$byClass["Specificity"]
                     )
)
```

Linear Discriminant Analysis is another form of the Naive Bayes model, but it asuems that that the the correlation structure is the same for all classes, which reduces the number of parameters we need to estimate. Again, considering the multidimensionality of the dataset, this is not the best assuption to make.

```
lda_fit <- train(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                 data = train, method = "lda")
lda_y_hat <- predict(lda_fit, test)
lda_con <- confusionMatrix(lda_y_hat,test$y, positive = "1")
results <- bind_rows(results,
                     data.frame(Type = "Linear Discriminant Analysis",
                                Accuracy = lda_con$overall["Accuracy"],
                                Sensitivity = lda_con$byClass["Sensitivity"],
                                Specificity= lda_con$byClass["Specificity"]
```

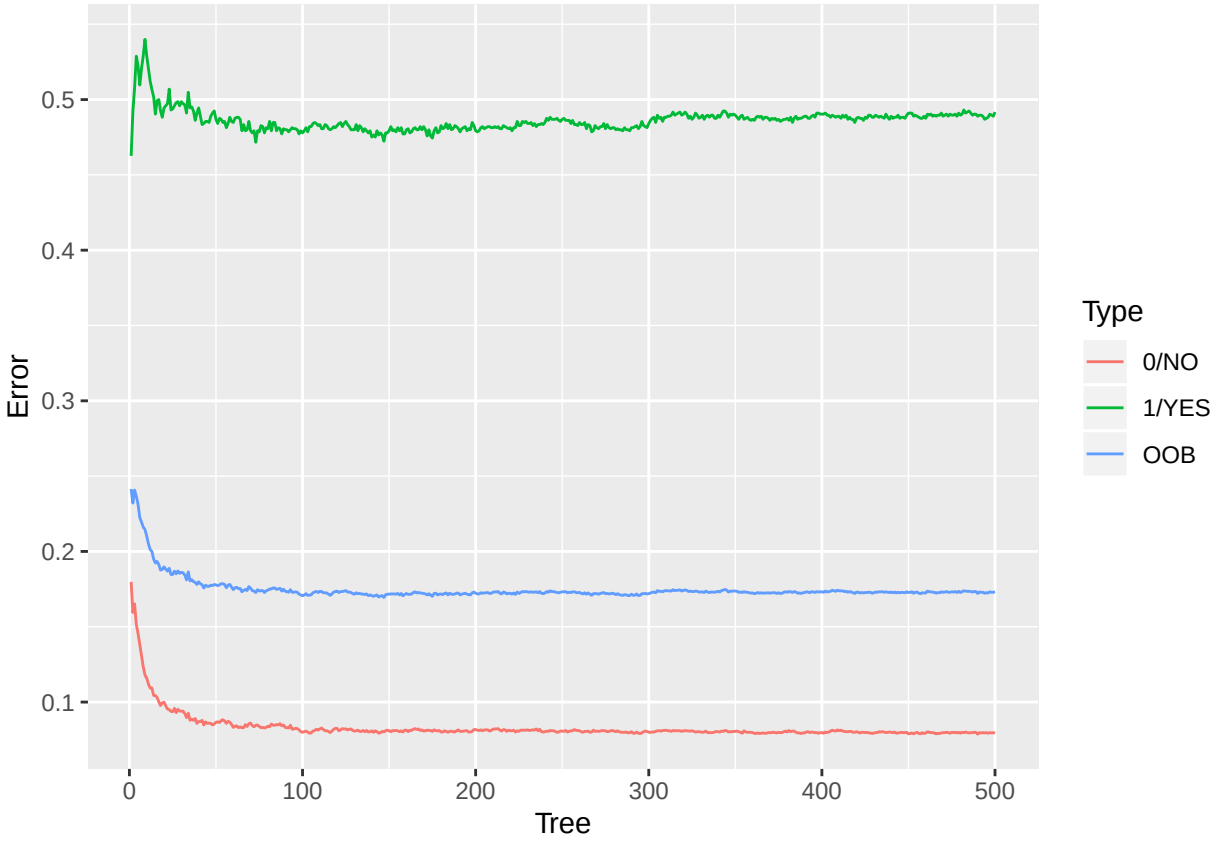
```
                    )
)
```

K-Nearest neighbors makes the assumption that similar outcomes exist where the predictors are in close proximity to each other. This proximity value is defined as distance. Similar to QDA and LDA, KNN struggles when the number of predictors increases.

```
knn_fit <- train(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                 data = train, method = "knn")
# final KNN algo uses 9 nearest-neighbors
knn_y_hat <- predict(knn_fit, test)
knn_con <- confusionMatrix(knn_y_hat,test$y, positive = "1")
results <- bind_rows(results,
                     data.frame(Type = "K-Nearest Neighbors",
                                Accuracy = knn_con$overall["Accuracy"],
                                Sensitivity = knn_con$byClass["Sensitivity"],
                                Specificity= knn_con$byClass["Specificity"]
                     )
)
```

Considering the data set has several useful predictors, which are a mix of factors and numeric values, the Random Forest model was the first model originally considered. In order to develop a more efficient model, the optimal tree count was determined, and then used to identify the optimal amount of nodes for the final Random Forest model.

```
# How many trees are optimal?
rf_fit <- randomForest(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                       data = train, proximity = TRUE)
rf_fit.df <- data.frame(
  Tree = rep(1:nrow(rf_fit$err.rate), times = 3),
  Type = rep(c("OOB","0/NO", "1/YES"),each=nrow(rf_fit$err.rate)),
  Error = c(rf_fit$err.rate[,"OOB"],
            rf_fit$err.rate[,"0"],
            rf_fit$err.rate[,"1"]
  )
)
rf_fit.df %>% ggplot(aes(Tree,Error)) + geom_line(aes(color = Type))
```



Error rates level off around 100 trees, so this will be the number of trees created in the model.

```
# How many nodes are optimal?
oob <- vector(length = 10)
for(i in 1:10){
  rf <- randomForest(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                     data = train, proximity = TRUE, ntree = 100, mtry = i)
  oob[i] <- rf$err.rate[nrow(rf$err.rate),1]
}
oob[which.min(oob)]
```

```
## [1] 0.1675327
```

```
node.min <-which.min(oob)
```

Three nodes are deal, so now the final Random Forest model can be developed.

```
rf_fit <- randomForest(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                       data = train, proximity = TRUE, ntree = 100, mtry = node.min)
rf_y_hat <- predict(rf_fit, test)
rf_con <- confusionMatrix(rf_y_hat,test$y, positive = "1")

results <- bind_rows(results,
                     data.frame(Type = "Random Forest",
                                Accuracy = rf_con$overall["Accuracy"],
```

```
                                Sensitivity = rf_con$byClass["Sensitivity"],
                                Specificity= rf_con$byClass["Specificity"]
                    )
)
```

Finally, three different Ensemble methods were developed. Ensemble methods take the y predictions and tally them on on a by-record basis. Whichever outcome has more votes becomes the predicted outcome for that record.

```
df <- data.frame( glm = ifelse(as.numeric(glm_y_hat)==1,0,1),
  knn = ifelse(as.numeric(knn_y_hat)==1,0,1),
  lda = ifelse(as.numeric(lda_y_hat)==1,0,1),
  qda = ifelse(as.numeric(qda_y_hat)==1,0,1),
  rPart = ifelse(as.numeric(rpart_y_hat)==1,0,1),
  rF = ifelse(as.numeric(rf_y_hat)==1,0,1)
)

p <- mean(test$y == 1)
votes <- rowSums(df)


ensemble_y_hat <- ifelse(votes >= 3,1,0)
ens_con1 <- confusionMatrix(factor(ensemble_y_hat), test$y, positive = "1")
results <- bind_rows(results,
                     data.frame(Type = "Ensemble v1",
                                Accuracy = ens_con1$overall["Accuracy"],
                                Sensitivity = ens_con1$byClass["Sensitivity"],
                                Specificity= ens_con1$byClass["Specificity"]
                     )
)

results %>% arrange(Sensitivity)
```

```
##                                 Type  Accuracy Sensitivity Specificity
## 1               K-Nearest Neighbors 0.7563739   0.2173913   0.9155963
## 2               Logistic Regression 0.8286119   0.4285714   0.9467890
## 3      Linear Discriminant Analysis 0.8271955   0.5031056   0.9229358
## 4                      Ensemble v1 0.8342776   0.5031056   0.9321101
## 5                            RPART 0.8243626   0.5341615   0.9100917
## 6                     Random Forest 0.8512748   0.5527950   0.9394495
## 7 Quantitative Discriminant Analysis 0.7804533   0.6459627   0.8201835
```

Another ensemble model was created to remove one model from the voting system. The model returns the model with the highest sensitivity. This is robust method to remove the worst performing algorithm.

```
ensemble_loop <- data.frame(MissingModel = character(),
                            Accuracy = numeric(),
                            Sensitivity = numeric(),
                            Specificity= numeric()
)

for (i in 1:length(df)){
```

```
  loop.df <- df[,-i]
  loop.votes <- rowSums(loop.df)
  loop.y <- ifelse(loop.votes >= 3,1,0)
  loop.conf <- confusionMatrix(factor(loop.y),test$y, positive = "1")
  ensemble_loop <- bind_rows(ensemble_loop,
                             data.frame(MissingModel = colnames(df[i]),
                                        Accuracy = loop.conf$overall["Accuracy"],
                                        Sensitivity = loop.conf$byClass["Sensitivity"],
                                        Specificity= loop.conf$byClass["Specificity"]
                             ))
}
ensemble_loop
```

```
##   MissingModel  Accuracy Sensitivity Specificity
## 1          glm 0.8371105   0.4906832   0.9394495
## 2          knn 0.8356941   0.4844720   0.9394495
## 3          lda 0.8399433   0.4720497   0.9486239
## 4          qda 0.8371105   0.4844720   0.9412844
## 5        rPart 0.8356941   0.4720497   0.9431193
## 6           rF 0.8328612   0.4658385   0.9412844
```

```
max_ensemble <- ensemble_loop[which.max(ensemble_loop$Sensitivity),]

results <- bind_rows(results,
                     data.frame(Type = "Ensemble v2 (remove worst performer)",
                                Accuracy = max_ensemble$Accuracy,
                                Sensitivity = max_ensemble$Sensitivity,
                                Specificity= max_ensemble$Specificity
                     )
)

results
```

```
##                                  Type  Accuracy Sensitivity Specificity
## 1                               RPART 0.8243626   0.5341615   0.9100917
## 2                 Logistic Regression 0.8286119   0.4285714   0.9467890
## 3   Quantitative Discriminant Analysis 0.7804533   0.6459627   0.8201835
## 4         Linear Discriminant Analysis 0.8271955   0.5031056   0.9229358
## 5                 K-Nearest Neighbors 0.7563739   0.2173913   0.9155963
## 6                       Random Forest 0.8512748   0.5527950   0.9394495
## 7                          Ensemble v1 0.8342776   0.5031056   0.9321101
## 8 Ensemble v2 (remove worst performer) 0.8371105   0.4906832   0.9394495
```

After reviewing the results of the models, the KNN method was the worst method in terms of sensitivity. This model was also removed from Ensemble Version 2. While the two Ensemble methods capture votes from a good deal of models, one final ensemble method was produced using the three best methods: LDA, QDA, and Random Forest.

```
# Top three are as LDA (3), QDA (4), RandomForest (6)
df3 <- df[c(3,4,6)]
votes3 <- rowSums(df3)
ensemble_y_hat3 <- ifelse(votes3 >= 2,1,0)
```

```
ens_con3 <- confusionMatrix(factor(ensemble_y_hat3), test$y, positive = "1")


results <- bind_rows(results,
                     data.frame(Type = "Ensemble v3 (Top 3 Performers)",
                                Accuracy = ens_con3$overall["Accuracy"],
                                Sensitivity = ens_con3$byClass["Sensitivity"],
                                Specificity= ens_con3$byClass["Specificity"]
                     )
)
results
```

```
##                                   Type  Accuracy Sensitivity Specificity
## 1                                RPART 0.8243626   0.5341615   0.9100917
## 2                  Logistic Regression 0.8286119   0.4285714   0.9467890
## 3   Quantitative Discriminant Analysis 0.7804533   0.6459627   0.8201835
## 4         Linear Discriminant Analysis 0.8271955   0.5031056   0.9229358
## 5                  K-Nearest Neighbors 0.7563739   0.2173913   0.9155963
## 6                        Random Forest 0.8512748   0.5527950   0.9394495
## 7                          Ensemble v1 0.8342776   0.5031056   0.9321101
## 8 Ensemble v2 (remove worst performer) 0.8371105   0.4906832   0.9394495
## 9       Ensemble v3 (Top 3 Performers) 0.8427762   0.5341615   0.9339450
```

After reviewing the logic, it appears QDA provided the best sensitivity but the lowest accuracy. While I initially aimed for the best sensitivity, the low accuracy is concerning. Additionally, the inherent assumptions made for this model, regarding the distribution of the predicting variable being bivariate normal, would be difficult to uphold considering this dataset is multidimensinoal. So this will not be the final method we use.

The next two models considered were the Random Forest and first Ensemble methods. The two models shared very similar accuracies, yet the Random Forest model beat out the ensemble method on the Sensitivity and Specificity scores. Considering this, and the fact that the ensemble methods requre 5 or more methods to train (which affects computer performance), the Random Forest method is the best model to use to predict whether or not a user will be a subscription user or not.

```
fit <- randomForest(y~poutcome+month+job+pdays+balance+day+education+age+marital+campaign+housing,
                    data = train, proximity = TRUE, ntree = 100, mtry = node.min)

y_hat <- predict(fit, validation)
final_con <- confusionMatrix(y_hat,validation$y, positive = "1")
results <- bind_rows(results,
                     data.frame(Type = "Random Forest - FINAL (validation)",
                                Accuracy = final_con$overall["Accuracy"],
                                Sensitivity = final_con$byClass["Sensitivity"],
                                Specificity= final_con$byClass["Specificity"]
                     ))
```

Results

In the end, the final model used to predict subscribers was the Random Forest model. As shown in the results table below, the Random Forest model did not have the highest sensitivity, however the combination that value and its high Accuracy and Sensitivity makes this a quality algorithm to leverage for this business case. The final Random Forest Model was tested against an untouched portion of the dataset, called the validation set, and returned an 83.3% Accuracy, a sensitivity of 57.5%, and a specificity of 90.9%.

```
##                                    Type  Accuracy Sensitivity Specificity
## 1                                 RPART 0.8243626   0.5341615   0.9100917
## 2                   Logistic Regression 0.8286119   0.4285714   0.9467890
## 3    Quantitative Discriminant Analysis 0.7804533   0.6459627   0.8201835
## 4          Linear Discriminant Analysis 0.8271955   0.5031056   0.9229358
## 5                   K-Nearest Neighbors 0.7563739   0.2173913   0.9155963
## 6                         Random Forest 0.8512748   0.5527950   0.9394495
## 7                           Ensemble v1 0.8342776   0.5031056   0.9321101
## 8  Ensemble v2 (remove worst performer) 0.8371105   0.4906832   0.9394495
## 9        Ensemble v3 (Top 3 Performers) 0.8427762   0.5341615   0.9339450
## 10   Random Forest - FINAL (validation) 0.8331210   0.5754190   0.9092409
```

Another additional benefit from not using an ensemble method was that we only need to train the dataset for a single model– the Random Forest model. This improves the performance of the model as less time is needed to predict whether or nor a user will ultimately sign up for the subscription-based account.

Conclusion

Ultimately, the model was successful in predicting if a user would become a paid subscriber through a Random Forest model that was optimized for tree and node count.

The main difficulty in this model was handling the class imbalance, which in turn affects the sensitivity of the model. With regard to the bank and their marketing campaigns, a low sensitivity means the model does not effectively predict that a user is a subscriber, given that they are an actual subscriber. The imbalanced set is demonstrated by the low prevalence of y=="yes" records in the dataset (12% in the original, 22% in the set after NA's were removed). There two potential resampling methods that could be used to improve the dataset's class imbalance:

- Undersampling the y=="no" records.
- Oversampling the y=="yes" records.

Considering there were not a great deal of records that did not contain at least one NA, oversampling would be a better idea, as adding more records would decrease inaccuracy and variation.

Another potential improvement would be to gather more information about the user's behavior. Metrics on the type of account would be a quick way to improve the model. Checking account members might be more the more active users, as checking accounts are generally meant for short-term finances, whereas savings is more meant for a long-term financial plan. Moreover, more information on account activity would be helpful, as a user that is consistently reviewing finances or sending payments might benefit from the subscription model.