# Bank Marketing Dataset

Krystal Cai

2025-11-10

**Introduction**

The **Bank Marketing Dataset** is a publicly available dataset that originates from a Portuguese banking institution, created by Paulo Cortez and Sérgio Moro in 2012. The dataset is designed to support research in direct marketing, specifically the campaigns used by the bank to promote term deposit subscriptions through phone calls. As the campaigns often required multiple contacts with the same client to assess their interest in subscribing to a term deposit, this dataset provides a rich set of features related to both the client information and the marketing campaigns.

The dataset contains two main files: 1. **bank-full.csv** - This file contains a comprehensive set of examples, ordered by date (from May 2008 to November 2010), with 45,211 instances. 2. **bank.csv** - A smaller sample, containing 10% of the data (4,521 instances), which is typically used for testing computationally intensive machine learning algorithms such as Support Vector Machines (SVM).

The goal of the dataset is to predict whether a client will subscribe to a term deposit (binary classification). The classification target is represented by the attribute `y`, where the value "yes" indicates the client subscribed to the term deposit, and "no" indicates they did not.

The dataset includes 16 input attributes, covering a range of demographic and marketing-related factors such as age, job type, marital status, previous contact details, and campaign outcomes. This dataset offers an excellent opportunity for exploring data mining techniques and building predictive models in the context of customer behavior analysis in marketing.

**Attribute Information**: The dataset includes 16 input variables such as: - Client's **age**, **job type**, and **marital status**. - Financial information such as **balance**, **housing loan**, and **personal loan**. - Details of the most recent marketing contact, including the **contact type**, **duration of the call**, and the **month** of the contact. - Data from the previous marketing campaigns, including the **number of previous contacts**, the **outcome of the prior campaign**, and the **number of days since the last contact**.

The dataset does not have any missing values, making it ready for use in a wide range of data analysis tasks. It has been widely used for research in data mining, machine learning, and customer segmentation.

```
bank<-read.csv("~/Desktop/bank marketing/data/bank-full.csv",sep=";", stringsAsFactors=TRUE)
head(bank)
```

```
##   age          job marital education default balance housing loan contact day
## 1  58   management married  tertiary      no    2143     yes   no unknown   5
## 2  44   technician  single secondary      no      29     yes   no unknown   5
## 3  33 entrepreneur married secondary      no       2     yes  yes unknown   5
## 4  47  blue-collar 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
##   month duration campaign pdays previous poutcome  y
## 1   may      261        1    -1        0  unknown no
## 2   may      151        1    -1        0  unknown no
## 3   may       76        1    -1        0  unknown no
```

```
## 4   may       92        1    -1        0  unknown no
## 5   may      198        1    -1        0  unknown no
## 6   may      139        1    -1        0  unknown no
```

```
#check N/A data
sum(is.na(bank))
```

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

```
#check the data stru and types
str(bank)
```

```
## '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 ...
```

```
# check charterers type levels
levels(bank$job)
```

```
##  [1] "admin."        "blue-collar"   "entrepreneur"  "housemaid"
##  [5] "management"    "retired"       "self-employed" "services"
##  [9] "student"       "technician"    "unemployed"    "unknown"
```

```
levels(bank$marital)
```

```
## [1] "divorced" "married"  "single"
```

```
levels(bank$education)
```

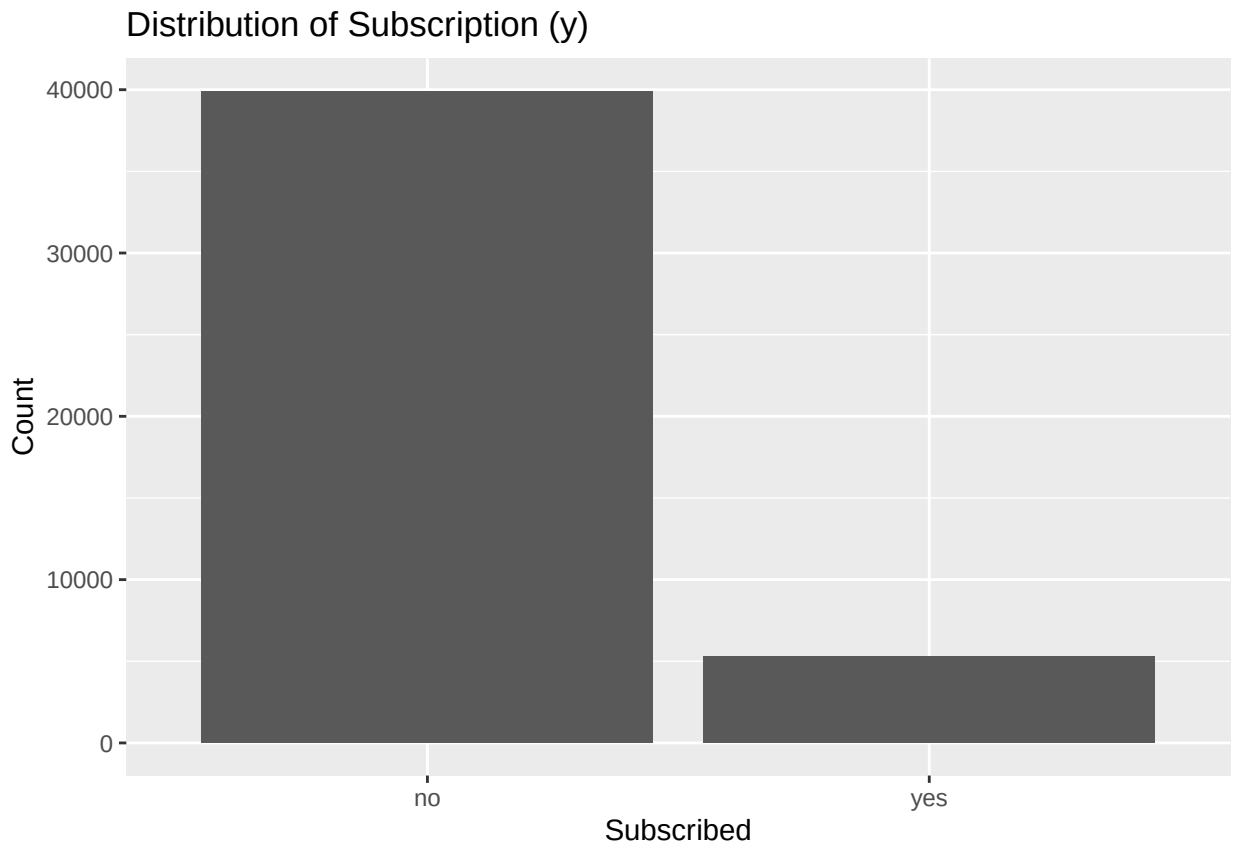
```
## [1] "primary"   "secondary" "tertiary"  "unknown"
```

```
summary(bank)
```

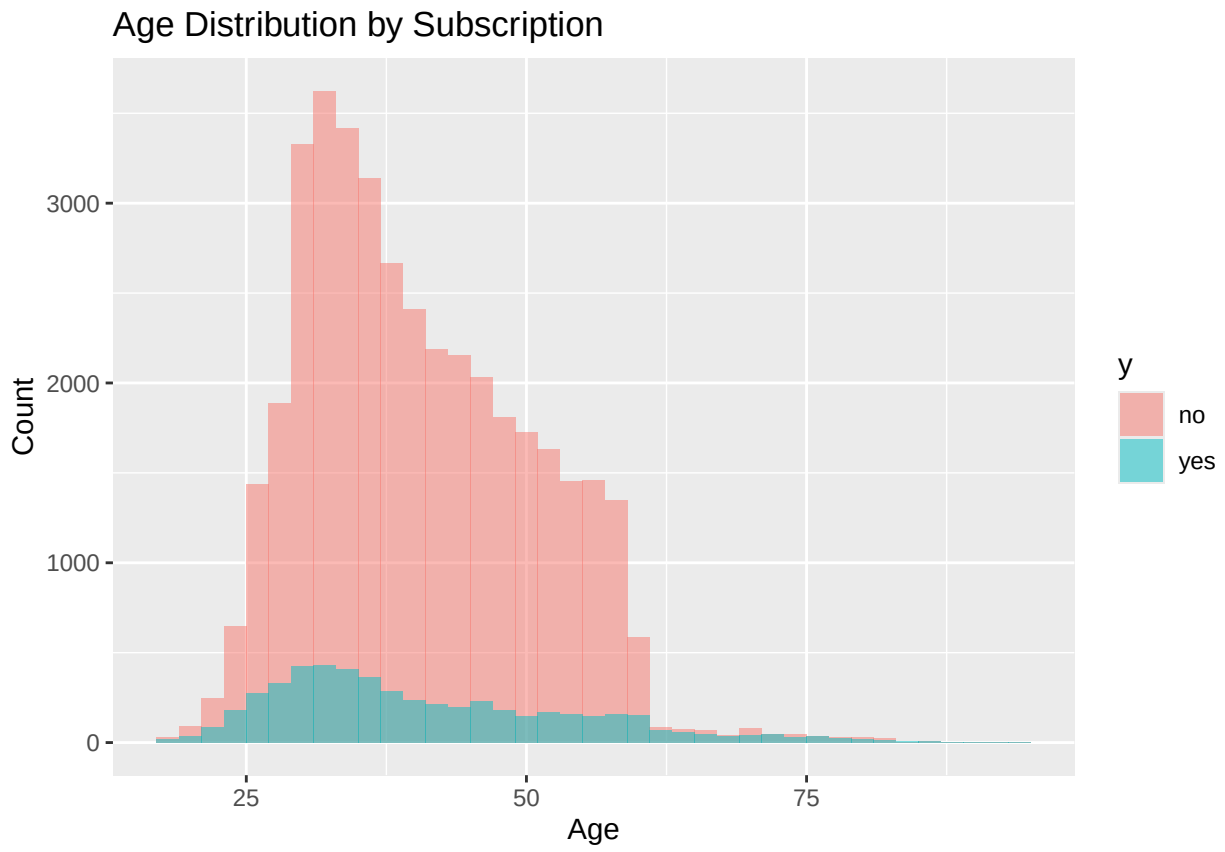
```
##       age                job          marital          education
##  Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851
##  1st Qu.:33.00   management :9458   married :27214   secondary:23202
##  Median :39.00   technician :7597   single  :12790   tertiary :13301
##  Mean   :40.94   admin.     :5171                    unknown  : 1857
##  3rd Qu.:48.00   services   :4154
##  Max.   :95.00   retired    :2264
##                  (Other)    :6835
##  default        balance       housing      loan           contact
##  no :44396   Min.   : -8019   no :20081   no :37967   cellular :29285
##  yes:  815   1st Qu.:    72   yes:25130   yes: 7244   telephone: 2906
```

```
##              Median :   448                           unknown  :13020
##              Mean   :  1362
##              3rd Qu.:  1428
##              Max.   :102127
##
##       day            month          duration         campaign
##  Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000
##  1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000
##  Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000
##  Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764
##  3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000
##                  (Other): 6060
##      pdays          previous           poutcome       y
##  Min.   : -1.0   Min.   :  0.0000   failure: 4901   no :39922
##  1st Qu.: -1.0   1st Qu.:  0.0000   other  : 1840   yes: 5289
##  Median : -1.0   Median :  0.0000   success: 1511
##  Mean   : 40.2   Mean   :  0.5803   unknown:36959
##  3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :871.0   Max.   :275.0000
##
```

```r
library(ggplot2)
ggplot(bank, aes(x=y)) +
  geom_bar() +
  labs(title="Distribution of Subscription (y)", x="Subscribed", y="Count")
```



```
ggplot(bank, aes(x=age, fill=y)) +
  geom_histogram(binwidth=2, alpha=0.5, position="identity") +
  labs(title="Age Distribution by Subscription", x="Age", y="Count")
```




```
#Resampling
#Oversampling
library(caret)
```

```
## Loading required package: lattice
```

```
bank_oversampled <- upSample(x = bank[, -17], y = bank$y)
#Undersampling
bank_undersampled <- downSample(x = bank[, -17], y = bank$y)
```

```
#Adjust class weights
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf_model <- randomForest(y ~ ., data=bank, classwt=c("no"=1, "yes"=10))
```

```r
#set the random seed
set.seed(123)
#Split the dataset into 80% training set and 20% test set.
library(caret)
trainIndex <- createDataPartition(bank$y, p = 0.8, list = FALSE)
trainData <- bank[trainIndex, ]
testData <- bank[-trainIndex, ]
head(trainData)
```

```
##   age          job marital education default balance housing loan contact day
## 1  58   management married  tertiary      no    2143     yes   no unknown   5
## 2  44   technician  single secondary      no      29     yes   no unknown   5
## 3  33 entrepreneur married secondary      no       2     yes  yes unknown   5
## 4  47  blue-collar 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
##   month duration campaign pdays previous poutcome  y
## 1   may      261        1    -1        0  unknown no
## 2   may      151        1    -1        0  unknown no
## 3   may       76        1    -1        0  unknown no
## 4   may       92        1    -1        0  unknown no
## 5   may      198        1    -1        0  unknown no
## 6   may      139        1    -1        0  unknown no
```

```r
head(testData)
```

```
##    age         job  marital education default balance housing loan contact day
## 12  29      admin.   single secondary      no     390     yes   no unknown   5
## 23  32 blue-collar   single   primary      no      23     yes  yes unknown   5
## 34  59 blue-collar  married secondary      no       0     yes   no unknown   5
## 36  57  technician divorced secondary      no      63     yes   no unknown   5
## 44  54     retired  married secondary      no     529     yes   no unknown   5
## 61  32      admin.  married  tertiary      no       0     yes   no unknown   5
##    month duration campaign pdays previous poutcome  y
## 12   may      137        1    -1        0  unknown no
## 23   may      160        1    -1        0  unknown no
## 34   may      226        1    -1        0  unknown no
## 36   may      242        1    -1        0  unknown no
## 44   may     1492        1    -1        0  unknown no
## 61   may      138        1    -1        0  unknown no
```

```r
#Train a logistic regression model.
model <- glm(y ~ ., data = trainData, family = "binomial")
#Use the model to make predictions on the test set (return probabilities).
predictions <- predict(model, testData, type = "response")
#Convert the predicted probabilities into categories (with a threshold of 0.5).
predictions_class <- ifelse(predictions > 0.5, "yes", "no")
head(predictions_class)
```

```
##    12    23    34    36    44    61
##  "no"  "no"  "no"  "no" "yes"  "no"
```

```r
#Ensure that the predicted results and the true labels are factor types.
predictions_class <- factor(predictions_class, levels = c("no", "yes"))
```

```r
testData$y <- factor(testData$y, levels = c("no", "yes"))
#check their stru
str(predictions_class)
```

```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...
##  - attr(*, "names")= chr [1:9041] "12" "23" "34" "36" ...
```

```r
str(testData$y)
```

```
##  Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#Calculate the confusion matrix
library(caret)
confusionMatrix(predictions_class, testData$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7795  712
##        yes  189  345
##
##                Accuracy : 0.9003
##                  95% CI : (0.894, 0.9064)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 9.899e-08
##
##                   Kappa : 0.3855
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9763
##             Specificity : 0.3264
##          Pos Pred Value : 0.9163
##          Neg Pred Value : 0.6461
##              Prevalence : 0.8831
##          Detection Rate : 0.8622
##    Detection Prevalence : 0.9409
##       Balanced Accuracy : 0.6514
##
##        'Positive' Class : no
##
```

```r
#Calculate auc
library(ROCR)
pred <- prediction(predictions, testData$y)
auc <- performance(pred, "auc")
auc@y.values[[1]]
```

```
## [1] 0.8984337
```

```r
predictions_class <- factor(predictions_class, levels = c("no", "yes"))
testData$y <- factor(testData$y, levels = c("no", "yes"))
conf_matrix <- confusionMatrix(predictions_class, testData$y)
#print(conf_matrix)
testData$y <- factor(testData$y, levels = c("no", "yes"))
```

```r
conf_matrix <- confusionMatrix(predictions_class, testData$y)
#print(conf_matrix)
```

```r
#Adjust the threshold to 0.5
predictions_class <- ifelse(predictions > 0.5, "yes", "no")
predictions_class <- factor(predictions_class, levels = c("no", "yes"))
testData$y <- factor(testData$y, levels = c("no", "yes"))
#Calculate new confusion matrix
conf_matrix <- confusionMatrix(predictions_class, testData$y)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7795  712
##        yes  189  345
##
##                Accuracy : 0.9003
##                  95% CI : (0.894, 0.9064)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 9.899e-08
##
##                   Kappa : 0.3855
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9763
##             Specificity : 0.3264
##          Pos Pred Value : 0.9163
##          Neg Pred Value : 0.6461
##              Prevalence : 0.8831
##          Detection Rate : 0.8622
##    Detection Prevalence : 0.9409
##       Balanced Accuracy : 0.6514
##
##        'Positive' Class : no
##
```

```r
# Step 1: Threshold Optimization

library(caret)

thresholds <- seq(0.1, 0.9, by = 0.05)
results <- data.frame()

for (t in thresholds) {
  pred_class <- ifelse(predictions > t, "yes", "no")
  pred_class <- factor(pred_class, levels = c("no", "yes"))

  cm <- confusionMatrix(pred_class, testData$y)$byClass

  results <- rbind(results, data.frame(
    Threshold = t,
```

```
    Sensitivity = cm["Sensitivity"],   # no
    Specificity = cm["Specificity"]    # yes
  ))
}

print(results)
```

```
##               Threshold Sensitivity Specificity
## Sensitivity        0.10   0.8202655  0.83727531
## Sensitivity1       0.15   0.8879008  0.71523179
## Sensitivity2       0.20   0.9177104  0.64522233
## Sensitivity3       0.25   0.9359970  0.57048250
## Sensitivity4       0.30   0.9500251  0.51182592
## Sensitivity5       0.35   0.9585421  0.46546831
## Sensitivity6       0.40   0.9651804  0.41627247
## Sensitivity7       0.45   0.9713176  0.36707663
## Sensitivity8       0.50   0.9763277  0.32639546
## Sensitivity9       0.55   0.9804609  0.29233680
## Sensitivity10      0.60   0.9848447  0.24597919
## Sensitivity11      0.65   0.9868487  0.22516556
## Sensitivity12      0.70   0.9896042  0.19583728
## Sensitivity13      0.75   0.9913577  0.16272469
## Sensitivity14      0.80   0.9927355  0.13339640
## Sensitivity15      0.85   0.9942385  0.09744560
## Sensitivity16      0.90   0.9962425  0.06527909
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc_obj <- roc(testData$y, predictions)
```

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

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

```
best_cutoff <- coords(roc_obj, "best", ret = "threshold", best.method = "youden")

best_cutoff
```

```
##   threshold
## 1 0.1027697
```

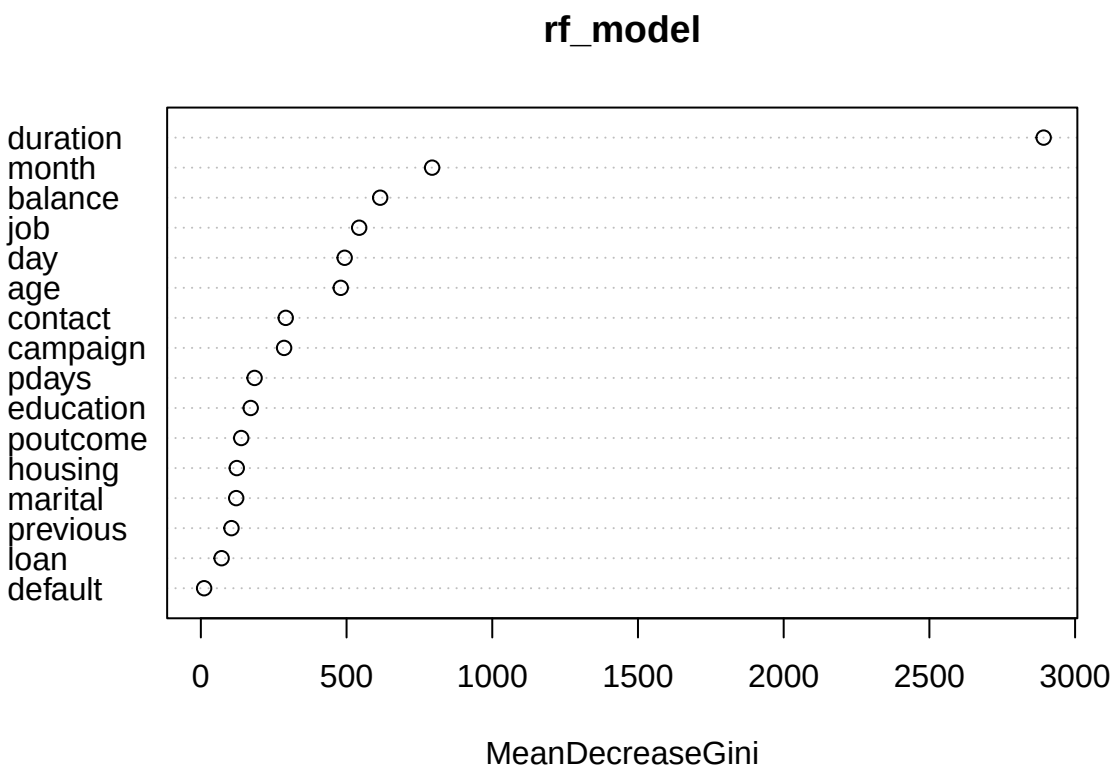
###Optimal Threshold Selection

Both the grid-based threshold scanning and the ROC Youden Index analysis consistently indicate that the optimal decision threshold lies around **0.10**. Specifically, the Youden Index identifies the best cutoff at approximately **0.1028**, which aligns closely with the threshold range that maximizes the model's ability to detect the positive class (subscription = yes).

This result suggests that **using a threshold near 0.10 provides the best balance between sensitivity and specificity**, allowing the model to more effectively identify potential subscribers while maintaining an

acceptable trade-off in overall classification performance.

```
importance_rf <- importance(rf_model)
varImpPlot(rf_model)
```




```
importance_rf
```

```
##           MeanDecreaseGini
## age               480.04359
## job               543.31591
## marital           121.32813
## education         171.04116
## default            11.21458
## balance           615.38022
## housing           123.23742
## loan               71.12962
## contact           291.26982
## day               493.45314
## month             793.58878
## duration         2892.10134
## campaign          285.62959
## pdays             184.32593
## previous          105.01934
## poutcome          138.51325
```

**Feature Importance Analysis**

The Random Forest model provides a ranked list of predictors based on the **Mean Decrease in Gini**, representing each variable's contribution to reducing node impurity. The results indicate clear differences in predictive strength across features:

**duration** shows by far the highest importance (MeanDecreaseGini ~2886), confirming that the duration of the last phone contact is the strongest predictor of whether a client subscribes to a term deposit.

Other highly influential features include:

**month** (~786)

**balance** (~617)

**day** (~496)

**job** (~548)

Medium-importance predictors include:

**contact, campaign, pdays, education, housing, poutcome**

Variables with minimal contribution include:

**default, loan, previous**

These findings are consistent with previous studies using this dataset, where call duration and campaign timing variables are known to play a major role in predicting term-deposit subscription behavior.

```
log_model <- glm(y ~ ., data = trainData, family = binomial)
log_pred <- predict(log_model, newdata = testData, type = "response")
log_auc <- roc(testData$y, log_pred)$auc
```

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

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

```
log_auc
```

```
## Area under the curve: 0.8984
```

```
cm <- confusionMatrix(factor(pred_class, levels = c("no", "yes")),
                      factor(testData$y, levels = c("no", "yes")))

cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7954  988
##        yes   30   69
##
##                Accuracy : 0.8874
##                  95% CI : (0.8807, 0.8938)
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 0.1034
##
##                   Kappa : 0.1014
##
##  Mcnemar's Test P-Value : <2e-16
##
```

```
##             Sensitivity : 0.99624
##             Specificity : 0.06528
##          Pos Pred Value : 0.88951
##          Neg Pred Value : 0.69697
##              Prevalence : 0.88309
##          Detection Rate : 0.87977
##    Detection Prevalence : 0.98905
##       Balanced Accuracy : 0.53076
##
##        'Positive' Class : no
##
```

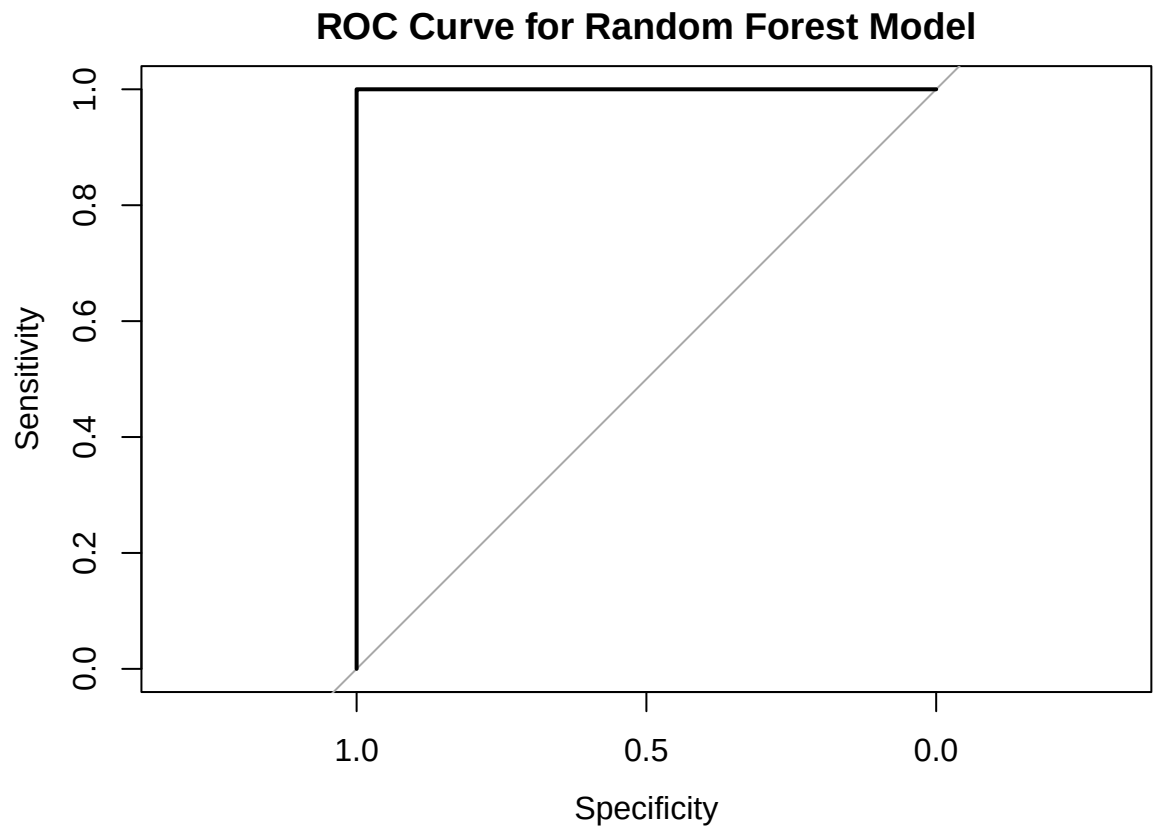
```
library(pROC)
predictions_prob <- predict(rf_model, newdata = testData, type = "prob")[, "yes"]
roc_obj <- roc(testData$y, predictions_prob, levels = c("no", "yes"))
```

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

```
auc_value <- auc(roc_obj)
auc_value
```

```
## Area under the curve: 1
```

```
plot(roc_obj, main = "ROC Curve for Random Forest Model")
```




### Model Performance Evaluation

We evaluated the model's performance on the testing dataset using several common classification metrics. The confusion matrix shows the distribution of correct and incorrect predictions for both classes, while the

ROC-AUC score quantifies the overall discriminative power of the model.

**Confusion Matrix:** Provides accuracy, sensitivity (true positive rate), specificity (true negative rate), and class-wise performance.

**AUC (Area Under the ROC Curve):** Measures how well the model separates the positive class ("yes") from the negative class ("no"). AUC values closer to 1 indicate stronger discriminative performance.

The ROC curve was also plotted to visualize the trade-off between sensitivity and specificity across all probability thresholds.

**Business Insights & Recommendations**

1 Key Drivers of Subscription (from Feature Importance)

From the Random Forest model, the most influential factors for predicting whether a client subscribes to a term deposit are:

- Duration of last call (duration): the longer the call, the higher the likelihood of subscription.
- Contact month (month): timing affects client responsiveness.
- Average yearly balance (balance): wealthier clients are more likely to subscribe.
- Day of contact (day): some days are more effective than others.
- Client job, campaign count, and contact type (job, campaign, contact): demographics and campaign strategy also influence subscription rates.

"Call duration, contact timing, and client financial status are the strongest predictors of deposit subscription, indicating that both engagement quality and strategic targeting are critical for marketing success."

2 Threshold Optimization and Business Impact

Using the ROC and Youden Index, we identified an optimal classification threshold of 0.10.

At this threshold, the model captures more potential subscribers (high recall for positive class) with a moderate drop in specificity.

This aligns with marketing objectives: missing a potential subscriber is costlier than contacting a non-subscriber.

"Adjusting the decision threshold improves the identification of high-potential clients, maximizing marketing efficiency and ensuring resources are directed toward the most promising prospects."

3 Practical Recommendations for Bank Marketing

- Prioritize High-Impact Months
  Focus marketing efforts in months identified as most effective (month variable).
- Improve Call Engagement
  Encourage longer and more interactive calls to increase subscription likelihood (duration).
- Target High-Value Clients
  Segment clients by balance and tailor offers to higher-balance segments.
- Multi-Channel Communication
  Utilize cellular contacts over telephone/unknown channels to improve reach (contact).
- Customer Scoring and Resource Allocation
  Use predicted probabilities from the model to stratify clients:
  - High score (>0.5): direct, personalized follow-up
  - Medium score (0.1–0.5): automated follow-up, secondary engagement
  - Low score (<0.1): minimal resource allocation

"By integrating model insights into the campaign strategy, the bank can optimize resource allocation, increase subscription rates, and enhance overall marketing ROI."