title: "Predictive Analysis and Model Evaluation"

author: "Adedeji Ogundipe"

output:

  word_document: default

  pdf_document: default

```{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

```

```{r}

rm(list = ls())

library(readr)

test_data <- read.csv("C:/Users/User/Desktop/PaPA/CW-test.csv")

train_data <- read.csv("C:/Users/User/Desktop/PaPA/CW-train.csv")

```

Test_data Variables

```{r}

names(test_data)

```

```
 [1] "age"       "job"       "marital"   "education" "default"   "balance"
 [7] "housing"   "loan"      "contact"   "day"       "month"     "duration"
[13] "campaign"  "pdays"     "previous"  "poutcome"  "y"
```

Train_data Variables

```{r}

names(train_data)
```

```
 [1] "age"       "job"       "marital"   "education" "default"   "balance"
 [7] "housing"   "loan"      "contact"   "day"       "month"     "duration"
[13] "campaign"  "pdays"     "previous"  "poutcome"  "y"
```

Number of subscribers based on train_data

```{r}

table(train_data$y)

```
    0   1
27920 3728
```

As 0 = "no" and 1 = "y", there are 3,728 subscribers and 27,920 non-subscribers.

Indicating "job" and other variables as factors in R

```{r}

glm(y ~ as.factor(job) + as.factor(housing)+ as.factor(education), family='binomial',

data = train_data)

```

```
Call:  glm(formula = y ~ as.factor(job) + as.factor(housing) + as.factor(education),
    family = "binomial", data = train_data)

Coefficients:
                  (Intercept)      as.factor(job)blue-collar
                     -1.82069                       -0.39093
   as.factor(job)entrepreneur        as.factor(job)housemaid
                     -0.53131                       -0.49765
     as.factor(job)management          as.factor(job)retired
                     -0.22831                        0.52241
  as.factor(job)self-employed         as.factor(job)services
                     -0.34587                       -0.33802
        as.factor(job)student       as.factor(job)technician
                      0.76392                       -0.20853
     as.factor(job)unemployed          as.factor(job)unknown
                      0.03381                       -0.54155
        as.factor(housing)yes  as.factor(education)secondary
                     -0.74823                        0.23250
 as.factor(education)tertiary    as.factor(education)unknown
                      0.60389                        0.29681

Degrees of Freedom: 31647 Total (i.e. Null);  31632 Residual
Null Deviance:      22950
Residual Deviance: 21930        AIC: 21960
```

Obtaining the standard errors and p-values of the train_data

```{r}

Model1logit = glm(y ~ as.factor(job) + as.factor(housing)+ as.factor(education), family='binomial', data = train_data)

summary(Model1logit)

```
Call:
glm(formula = y ~ as.factor(job) + as.factor(housing) + as.factor(education),
    family = "binomial", data = train_data)

Deviance Residuals:
    Min   1Q   Median   3Q   Max
-0.9921 -0.5540 -0.4333 -0.3559   2.5116

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.82069   0.08030 -22.675 < 2e-16 ***
as.factor(job)blue-collar   -0.39093   0.07209 -5.423 5.87e-08 ***
as.factor(job)entrepreneur   -0.53131   0.12389 -4.289 1.80e-05 ***
as.factor(job)housemaid   -0.49765   0.13365 -3.723 0.000197 ***
as.factor(job)management   -0.22831   0.07185 -3.178 0.001485 **
as.factor(job)retired   0.52241   0.08193   6.376 1.81e-10 ***
as.factor(job)self-employed   -0.34587   0.11072 -3.124 0.001785 **
as.factor(job)services   -0.33802   0.08374 -4.037 5.42e-05 ***
as.factor(job)student   0.76392   0.10296   7.419 1.18e-13 ***
as.factor(job)technician   -0.20853   0.06763 -3.083 0.002048 **
as.factor(job)unemployed   0.03381   0.10789   0.313 0.754005
as.factor(job)unknown   -0.54155   0.23571 -2.298 0.021587 *
as.factor(housing)yes   -0.74823   0.03771 -19.844 < 2e-16 ***
as.factor(education)secondary 0.23250   0.06295   3.693 0.000221 ***
as.factor(education)tertiary   0.60389   0.07203   8.384 < 2e-16 ***
as.factor(education)unknown   0.29681   0.10267   2.891 0.003840 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 22945 on 31647 degrees of freedom
Residual deviance: 21926 on 31632 degrees of freedom
AIC: 21958

Number of Fisher Scoring iterations: 5
```

```{r}

predict_model1 <- predict(Model1_train, train_data, type = "response")

```

Building a logistic model to predict using a probability cut-off of 0.5

```{r}

install.packages("ggplot2", repos = 'http://cran.us.r-project.org')

installed.packages("lattice")
```

install.packages("caret")

```

```{r}

library(ggplot2)

library(lattice)

library(caret)

```

```{r}

predict_model1_class <- ifelse(predict_model1 >0.5, 1, 0)

train_data$predict_model1_class = predict_model1_class

confusionMatrix(as.factor(train_data$predict_model1_class), as.factor(train_data$y))

```

```
Warning: Levels are not in the same order for reference and data. Refactoring data to match.Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27920  3728
         1     0     0

               Accuracy : 0.8822
                 95% CI : (0.8786, 0.8857)
    No Information Rate : 0.8822
    P-Value [Acc > NIR] : 0.5044

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8822
         Neg Pred Value :    NaN
             Prevalence : 0.8822
         Detection Rate : 0.8822
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

The report above suggests that job is not a sufficiently good predictor or determinant for subscription.

Building a model for "y" with more variables

```{r}

Model2_train = glm(y ~ age + as.factor(job) + as.factor(marital) + balance + as.factor(housing) , family='binomial', data = train_data ) predict_model2 <- predict(Model2_train, train_data, type = "response")

predict_model2_class <- ifelse(predict_model2 > 0.5, 1, 0)

train_data$predict_model2_class = predict_model2_class

confusionMatrix(as.factor(train_data$predict_model2_class), as.factor(train_data$y))

```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27916  3727
         1     4     1
                                          
               Accuracy : 0.8821          
                 95% CI : (0.8785, 0.8856)
    No Information Rate : 0.8822          
    P-Value [Acc > NIR] : 0.5252          
                                          
                  Kappa : 2e-04           
                                          
 Mcnemar's Test P-Value : <2e-16          
                                          
            Sensitivity : 0.9998567       
            Specificity : 0.0002682       
         Pos Pred Value : 0.8822172       
         Neg Pred Value : 0.2000000       
             Prevalence : 0.8822042       
         Detection Rate : 0.8820779       
   Detection Prevalence : 0.9998420       
      Balanced Accuracy : 0.5000625       
                                          
       'Positive' Class : 0               
```

Even when more variables are considered, the prediction seems to be more at variance with reality. This may be an indictment on the model.

By adjusting the cut-off to 0.25,

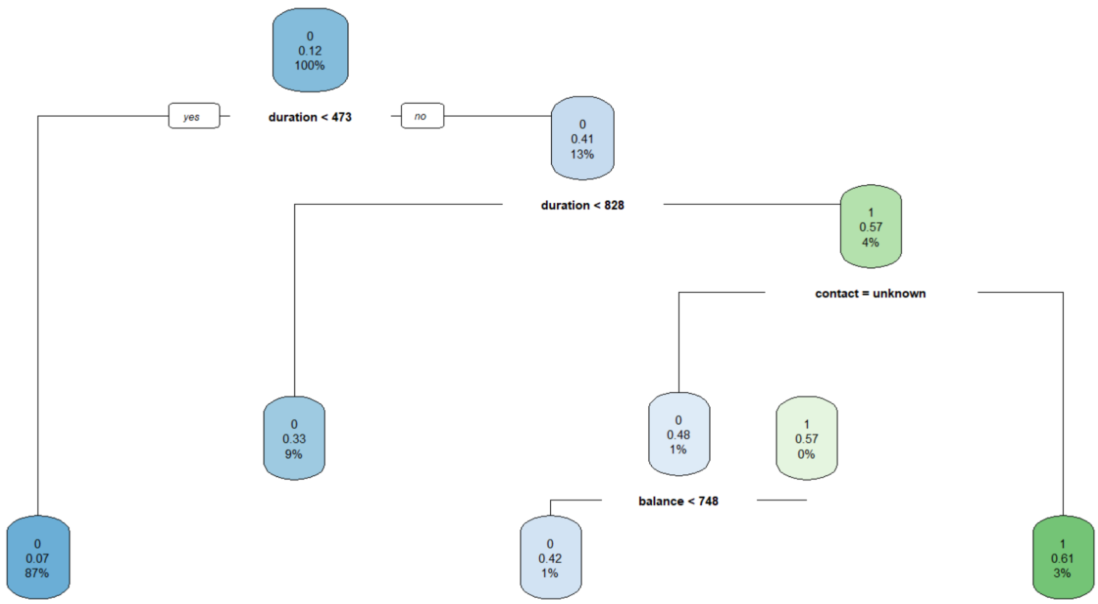
```{r}

Model2_train = glm(y ~ age + as.factor(job) + as.factor(marital) + balance + as.factor(housing) , family='binomial', data = train_data )

predict_model2 <- predict(Model2_train, train_data, type = "response") predict_model2_class <- ifelse(predict_model2 > 0.25, 1, 0)

train_data$predict_model2_class = predict_model2_class

confusionMatrix(as.factor(train_data$predict_model2_class), as.factor(train_data$y))

```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27170  3366
         1   750   362

               Accuracy : 0.8699
                 95% CI : (0.8662, 0.8736)
    No Information Rate : 0.8822
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1009

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9731
            Specificity : 0.0971
         Pos Pred Value : 0.8898
         Neg Pred Value : 0.3255
             Prevalence : 0.8822
         Detection Rate : 0.8585
   Detection Prevalence : 0.9649
      Balanced Accuracy : 0.5351

       'Positive' Class : 0
```

From the foregoing, the true positive is about 32.55% whilst the false positive takes about 64.45 of the total number of subscribers predicted. On the other hand, the prediction for non-subscribers has 88.98 true negative and 11.02 false negative. Consequently this model seems more reliable for predicting the number of potential non-subscribers than for predicting potential subscribers.

Decision trees model

As an alternative to the logistic model, the decision-trees model can be tried and its accuracy assessed accordingly.
```

````
```{r}
install.packages("rpart")
install.packages("rpart.plot")
```
```{r}
library(rpart)
library(rpart.plot)
```
```{r}
tree <- rpart(formula = y ~ job +age + marital+ balance+ duration + campaign + contact
+ loan + default ,
data = train_data,
# minsplit =5,
# minbucket =10,
# control = list(minbucket=10, maxdepth=4),
method = 'class', cp =0.004
)
```
```{r}
rpart.plot(tree)
```
````




```{r}

PredictCART_train = predict(tree, data = train_data, type = "class")

train_data$PredictCART_train = PredictCART_train

confusionMatrix(train_data$PredictCART_train, as.factor(train_data$y))

```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27499  3091
         1   421   637

               Accuracy : 0.889
                 95% CI : (0.8855, 0.8925)
    No Information Rate : 0.8822
    P-Value [Acc > NIR] : 7.604e-05

                  Kappa : 0.2259

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9849
            Specificity : 0.1709
         Pos Pred Value : 0.8990
         Neg Pred Value : 0.6021
             Prevalence : 0.8822
         Detection Rate : 0.8689
   Detection Prevalence : 0.9666
      Balanced Accuracy : 0.5779

       'Positive' Class : 0
```

Based on the decision tree model, the true positive records 60.21% as against the 39.79% recorded by false positive for the number of subscribers predicted. Contrarily, the prediction for non-subscribers has 89.90% true negative and 10.10% for false negative. This shows that the model still predicts the number of potential non-subscribers fairly better than that of the subscribers although there's an improvement when compared to the logistic model.

KNN Model

Using the KNN model, a new data set is created from the train_data set with 5000 observations and renamed "small_train_data". Another one is also created from the test_data set and renamed small_test_data.

```{r}
```

small_train_data = train_data[1:1000,]

small_test_data = train_data[1:1000,]

```

```{r}

knn.fit <- train(y ~ age + balance , data = small_train_data,

method = "knn",

# tuneLength = 17,

preProcess=c("center", "scale"))

```

```{r}

knn.fit

```

```
k-Nearest Neighbors

1000 samples
   2 predictor
   2 classes: '0', '1'

Pre-processing: centered (2), scaled (2)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9725655  -0.007521545
  7  0.9762180  -0.002869700
  9  0.9783961   0.000000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

Predicting with KNN using the small train_data set

```{r}

predictions = predict(knn.fit, small_train_data)

confusionMatrix(predictions, small_train_data$y)

```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 980  20
         1   0   0

               Accuracy : 0.98
                 95% CI : (0.9693, 0.9877)
    No Information Rate : 0.98
    P-Value [Acc > NIR] : 0.5591

                  Kappa : 0

 Mcnemar's Test P-Value : 2.152e-05

            Sensitivity : 1.00
            Specificity : 0.00
         Pos Pred Value : 0.98
         Neg Pred Value :  NaN
             Prevalence : 0.98
         Detection Rate : 0.98
   Detection Prevalence : 1.00
      Balanced Accuracy : 0.50

       'Positive' Class : 0
```

Predicting with KNN using the original train_data set

```{r}

knn.fit1 <- train(y ~ age + balance , data = train_data,

method = "knn",

# tuneLength = 17,

preProcess=c("center", "scale"))

```

```{r}

predictions = predict(knn.fit1, train_data)

confusionMatrix(predictions, train_data$y)

```
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27735  3419
         1   185   309

               Accuracy : 0.8861
                 95% CI : (0.8826, 0.8896)
    No Information Rate : 0.8822
    P-Value [Acc > NIR] : 0.01532

                  Kappa : 0.1222

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.99337
            Specificity : 0.08289
         Pos Pred Value : 0.89025
         Neg Pred Value : 0.62551
             Prevalence : 0.88220
         Detection Rate : 0.87636
   Detection Prevalence : 0.98439
      Balanced Accuracy : 0.53813

       'Positive' Class : 0
```

Using the original train_data, KNN predicts the number of potential subscribers at 62.63% which appears marginally better than the position recorded with the decision-trees model.

Random Forest Model

```{r}

install.packages("randomForest")

```

```{r}

library(randomForest)

```

```{r}

random_forest_class = randomForest(y ~.,

data = train_data, #train_data data set

importance = T)

```

```{r}

p1 <- predict(random_forest_class, train_data)

confusionMatrix(p1, train_data$y)

```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 27920    78
         1     0  3650

               Accuracy : 0.9975
                 95% CI : (0.9969, 0.9981)
    No Information Rate : 0.8822
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.988

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.9791
         Pos Pred Value : 0.9972
         Neg Pred Value : 1.0000
             Prevalence : 0.8822
         Detection Rate : 0.8822
   Detection Prevalence : 0.8847
      Balanced Accuracy : 0.9895

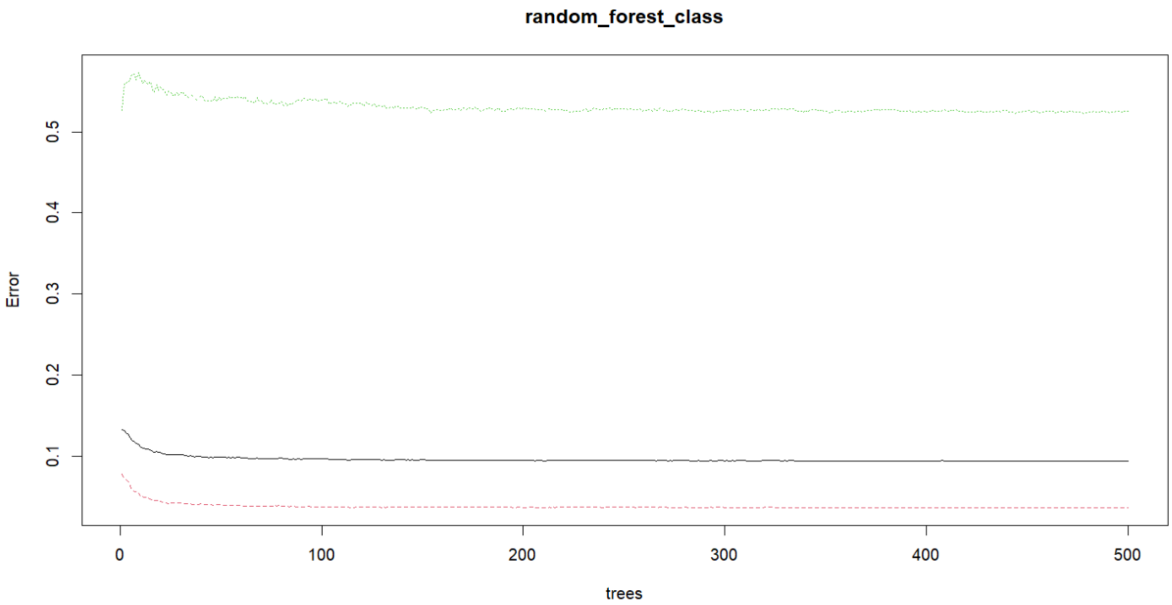
       'Positive' Class : 0
```

This model records 99.51% accuracy and 100% true positive of the prediction for potential subscribers. So far, it appears the best of the models tested.

```{r}

plot(random_forest_class)

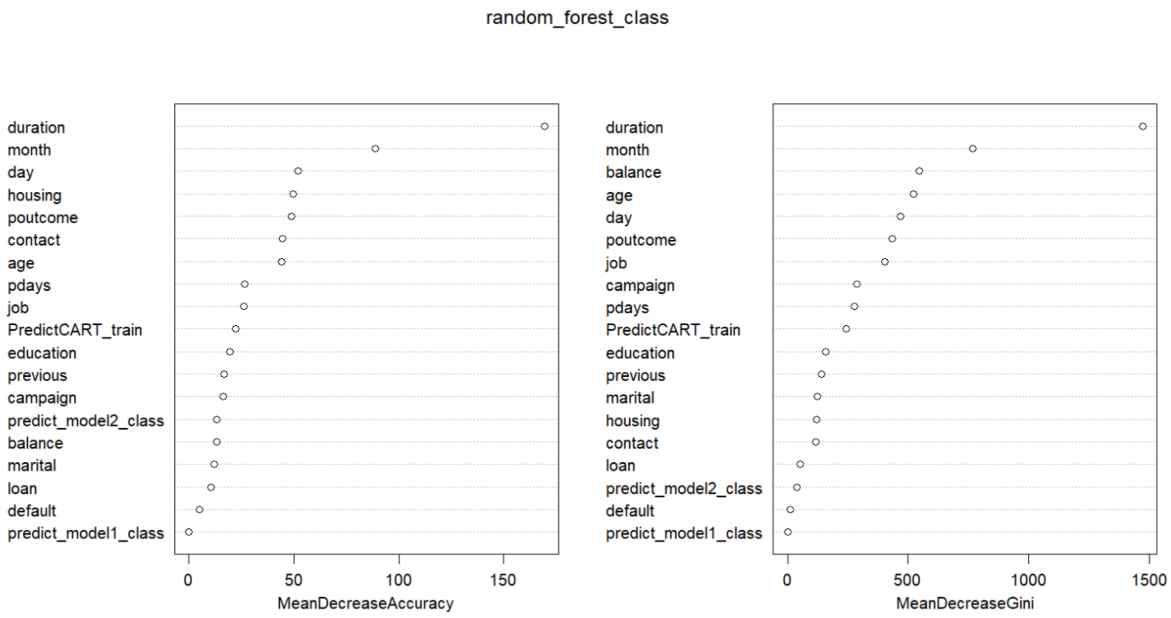
```




```{r}

varImpPlot(random_forest_class)

```



Applying the models to the test data

Logistic model

```{r}

predict_model2 <- predict(Model2_train, test_data, type = "response")

predict_model2_class <- ifelse(predict_model2 > 0.25, 1, 0)

test_data$predict_model2_class = predict_model2_class

confusionMatrix(as.factor(test_data$predict_model2_class), as.factor(test_data$y))

```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11678  1404
         1   324   157
                                          
               Accuracy : 0.8726          
                 95% CI : (0.8669, 0.8782)
    No Information Rate : 0.8849          
    P-Value [Acc > NIR] : 1               
                                          
                  Kappa : 0.1053          
                                          
 Mcnemar's Test P-Value : <2e-16          
                                          
            Sensitivity : 0.9730          
            Specificity : 0.1006          
         Pos Pred Value : 0.8927          
         Neg Pred Value : 0.3264          
             Prevalence : 0.8849          
         Detection Rate : 0.8610          
   Detection Prevalence : 0.9645          
      Balanced Accuracy : 0.5368          
                                          
       'Positive' Class : 0               
```

Model accuracy: 87.26%

Decision trees Model

```{r}

tree1 <- rpart(formula = y ~ job +age + marital+ balance+ duration + campaign +

contact + loan + default ,

data = test_data,

# minsplit =5,

# minbucket =10,

# control = list(minbucket=10, maxdepth=4),

method = 'class', cp =0.004

)

```

```{r}

PredictCART_test = predict(tree1, test_data, type = "class")

test_data$PredictCART_test = PredictCART_test

confusionMatrix(test_data$PredictCART_test, as.factor(test_data$y))
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11807  1196
         1   195   365

               Accuracy : 0.8974
                 95% CI : (0.8922, 0.9025)
    No Information Rate : 0.8849
    P-Value [Acc > NIR] : 1.807e-06

                  Kappa : 0.3017

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9838
            Specificity : 0.2338
         Pos Pred Value : 0.9080
         Neg Pred Value : 0.6518
             Prevalence : 0.8849
         Detection Rate : 0.8705
   Detection Prevalence : 0.9587
      Balanced Accuracy : 0.6088

       'Positive' Class : 0
```

Model accuracy: 89.74%

KNN Model

```{r}

predictions = predict(knn.fit, small_test_data)

confusionMatrix(predictions, small_test_data$y)
```

```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 980 20
         1   0   0

               Accuracy : 0.98
                 95% CI : (0.9693, 0.9877)
    No Information Rate : 0.98
    P-Value [Acc > NIR] : 0.5591

                  Kappa : 0

 Mcnemar's Test P-Value : 2.152e-05

            Sensitivity : 1.00
            Specificity : 0.00
         Pos Pred Value : 0.98
         Neg Pred Value : NaN
             Prevalence : 0.98
         Detection Rate : 0.98
   Detection Prevalence : 1.00
      Balanced Accuracy : 0.50

       'Positive' Class : 0
```

Model accuracy: 98.00%

**Random forest**

```{r}

random_forest_class1 = randomForest(y ~.,

data = test_data, importance = T)

```
```

```
Warning: The response has five or fewer unique values.  Are you sure you want to do regression?
```

```{r}

p2 <- predict(random_forest_class1, test_data)

confusionMatrix(p2, test_data$y)

```

The decision tree is hereby adopted as the choice model for the assignment

Assuming all the variables are included in the model

```{r}

tree2 <- rpart(formula = y ~ age + job + marital + education + default + balance + housing + loan + contact + day + month + duration + campaign + pdays + previous + poutcome ,

data = train_data,

# minsplit =5,

# minbucket =10,

# control = list(minbucket=10, maxdepth=4),

method = 'class', cp =0.004

)

```

Identifying the importance of each variable in the model (Goman, 2014a)

```{r}

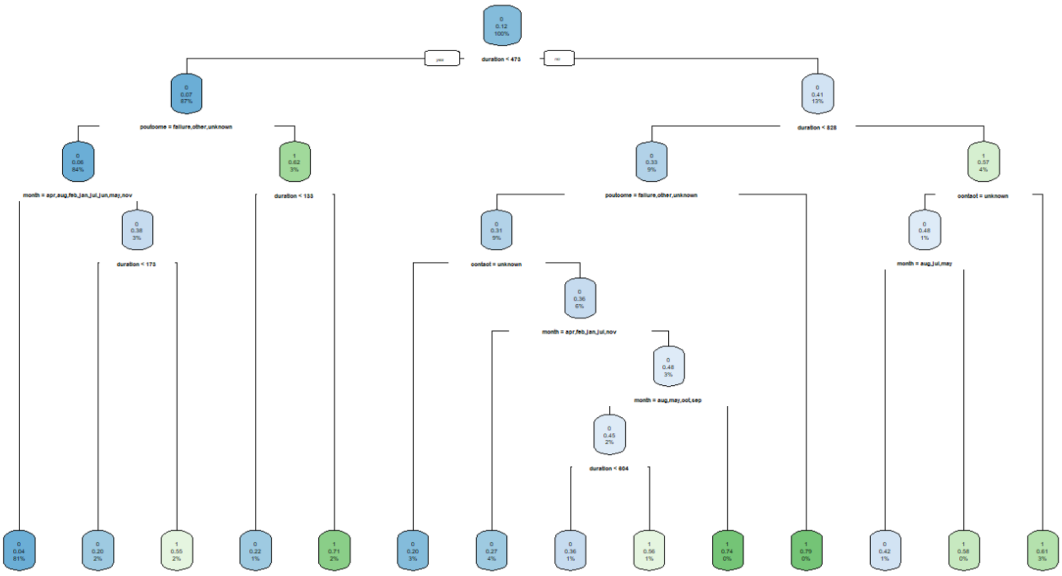
tree2$variable.importance

```

```
    duration     poutcome        month      contact          day 
1037.9492602  588.2552789  295.5053183   47.8173947   15.3466371 
     housing          job        pdays          age      balance 
   8.6261065    8.5196710    5.5441366    5.2497339    4.4469251 
    campaign     previous    education      default 
   2.0859253    1.8792353    0.4759936    0.3829057 
```

From the above, "duration" has the highest sum of the goodness of split measures and can therefore be considered as the most significant of the variables (Goman, 2014a; Therneau et al, 2022).

Plotting the decision tree for tree2



Checking the cross-validation results

```{r}

printcp(tree2)

```

```
Classification tree:
rpart(formula = y ~ age + job + marital + education + default +
    balance + housing + loan + contact + day + month + duration +
    campaign + pdays + previous + poutcome, data = train_data,
    method = "class", cp = 0.004)

Variables actually used in tree construction:
[1] contact  duration month    poutcome

Root node error: 3728/31648 = 0.1178

n= 31648

         CP nsplit rel error  xerror     xstd
1 0.0346924      0   1.00000 1.00000 0.015383
2 0.0249464      3   0.89592 0.93079 0.014910
3 0.0217275      4   0.87098 0.88948 0.014615
4 0.0073766      5   0.84925 0.86347 0.014424
5 0.0054319      7   0.83450 0.85354 0.014350
6 0.0042918     11   0.81277 0.84523 0.014288
7 0.0040000     13   0.80418 0.84469 0.014284
```

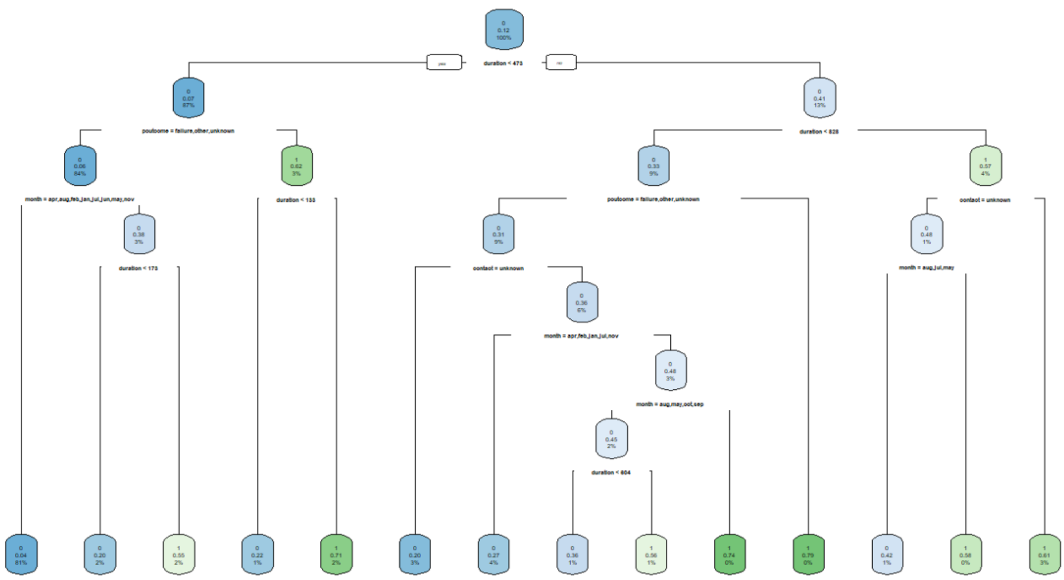
From the above, the best xerror is 0.84281 with xstd of 0.014270; hence, we look for the smallest tree with xerror less than 0.85708 i.e., (0.84281+0.014270). Therefore, the tree with cp=0.0040000 is selected and the tree will be pruned with a cp slightly greater than 0.004, say, 0.0041.

```{r}

tree2 <- prune(tree2, cp = 0.0041)

rpart.plot(tree2)

```



Using the model to predict with test_data

tree3 is created from tree1 with an adjustment from cp=0.004 to cp=0.0041

tree3 <- rpart(formula = y ~ job +age + marital+ balance+ duration + campaign + contact + loan + default ,

data = test_data,

minsplit =5,

minbucket =10,

control = list(minbucket=10, maxdepth=4),

method = 'class', cp =0.0041

)

```

```{r}

PredictCART_test = predict(tree3, test_data, type = "class")

test_data$PredictCART_test = PredictCART_test

confusionMatrix(test_data$PredictCART_test, as.factor(test_data$y))

`

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 11807 1196
         1   195   365

               Accuracy : 0.8974
                 95% CI : (0.8922, 0.9025)
    No Information Rate : 0.8849
    P-Value [Acc > NIR] : 1.807e-06

                  Kappa : 0.3017

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9838
            Specificity : 0.2338
         Pos Pred Value : 0.9080
         Neg Pred Value : 0.6518
             Prevalence : 0.8849
         Detection Rate : 0.8705
   Detection Prevalence : 0.9587
      Balanced Accuracy : 0.6088

       'Positive' Class : 0
```