Decision Trees & Random Forest

Niti Wattanasirichaigoon (A20406934)

12/4/2020

Load Data

```
df_train <- read.csv(file="H1.csv",sep=",",stringsAsFactors = FALSE, na.strings = "NULL")
df_test <- read.csv(file="H2.csv",sep=",",stringsAsFactors = FALSE, na.strings = "NULL")
```

Data Transformation for training and testing

```
df_train$Country <- as.numeric(factor(df_train$Country))
df_train$MarketSegment <- as.numeric(factor(df_train$MarketSegment))
df_train$DepositType <- as.numeric(factor(df_train$DepositType))
df_train$CustomerType <- as.numeric(factor(df_train$CustomerType))
df_train$IsCanceled <- factor(df_train$IsCanceled)

df_train <- df_train[c('IsCanceled','LeadTime', 'Country', 'MarketSegment',
                       'DepositType', 'CustomerType', 'RequiredCarParkingSpaces',
                       'ArrivalDateWeekNumber')]

df_test$Country <- as.numeric(factor(df_test$Country))
df_test$MarketSegment <- as.numeric(factor(df_test$MarketSegment))
df_test$DepositType <- as.numeric(factor(df_test$DepositType))
df_test$CustomerType <- as.numeric(factor(df_test$CustomerType))
df_test$IsCanceled <- factor(df_test$IsCanceled)

df_test <- df_test[c('IsCanceled','LeadTime', 'Country', 'MarketSegment',
                       'DepositType', 'CustomerType', 'RequiredCarParkingSpaces',
                       'ArrivalDateWeekNumber')]

addmargins(table(df_train$IsCanceled))
```

```
##
##     0     1   Sum
## 28938 11122 40060
```

Training and validation split

```
library(caret)
```

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

```
## Loading required package: ggplot2
```

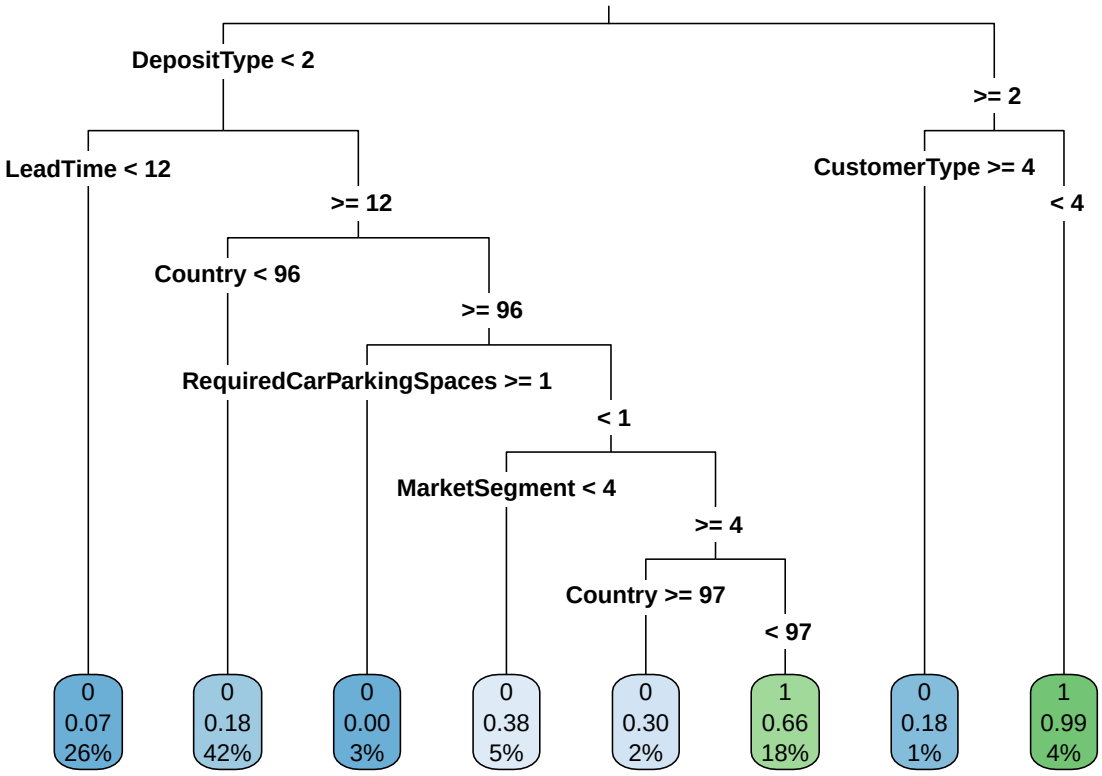
```r
set.seed(111)
trainIndex = createDataPartition(df_train$IsCanceled, p = .8, list = FALSE)
val_x = df_train[-trainIndex, -1]
val_y = df_train[-trainIndex, 1]
```

Decision Tree

Build simple decision tree, plot it, and calculate misclassification rate.

```r
library(rpart)
library(rpart.plot)

# using all attributes
fit = rpart(IsCanceled ~ ., data = df_train[trainIndex,])
rpart.plot(fit, type=3)
```



Training misclassification rate and confusion matrix

```r
tree_pred = predict(fit, df_train[trainIndex,-1], type = 'class')
miss_rate = sum(tree_pred != df_train[trainIndex,1])/length(tree_pred)
miss_rate
```

```
## [1] 0.1807545
```

```r
library(e1071)
confusionMatrix(tree_pred, df_train[trainIndex,1])
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction     0     1
##          0 21205  3847
##          1  1946  5051
##
##                Accuracy : 0.8192
##                  95% CI : (0.815, 0.8234)
##     No Information Rate : 0.7224
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5176
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9159
##             Specificity : 0.5677
##          Pos Pred Value : 0.8464
##          Neg Pred Value : 0.7219
##              Prevalence : 0.7224
##          Detection Rate : 0.6616
##    Detection Prevalence : 0.7817
##       Balanced Accuracy : 0.7418
##
##        'Positive' Class : 0
##
```

Validation misclassification rate and confusion matrix

```
tree_pred = predict(fit, val_x, type = 'class')
miss_rate = sum(tree_pred != val_y)/length(tree_pred)
miss_rate
```

```
## [1] 0.1815004
```

```
confusionMatrix(tree_pred, val_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5304  971
##          1  483 1253
##
##                Accuracy : 0.8185
##                  95% CI : (0.8099, 0.8269)
##     No Information Rate : 0.7224
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5147
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9165
##             Specificity : 0.5634
```

```
##          Pos Pred Value : 0.8453
##          Neg Pred Value : 0.7218
##              Prevalence : 0.7224
##          Detection Rate : 0.6621
##    Detection Prevalence : 0.7833
##       Balanced Accuracy : 0.7400
##
##        'Positive' Class : 0
##
```

Test set misclassification rate and confusion matrix

```
tree_pred = predict(fit, df_test[,-1], type = 'class')
miss_rate = sum(tree_pred != df_test[,1])/length(tree_pred)
miss_rate
```

```
## [1] 0.2680953
```

```
confusionMatrix(tree_pred, df_test[,1])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 46214 21254
##          1    14 11848
##
##                Accuracy : 0.7319
##                  95% CI : (0.7288, 0.735)
##     No Information Rate : 0.5827
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3935
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9997
##             Specificity : 0.3579
##          Pos Pred Value : 0.6850
##          Neg Pred Value : 0.9988
##              Prevalence : 0.5827
##          Detection Rate : 0.5826
##    Detection Prevalence : 0.8505
##       Balanced Accuracy : 0.6788
##
##        'Positive' Class : 0
##
```

Random Forest

```
library(randomForest)
set.seed(123)
rf_fit = randomForest(IsCanceled ~ ., data = df_train[trainIndex,], na.action=na.omit)
```

Training confusion matrix

```
rf_pred = predict(rf_fit, df_train[trainIndex,-1], type = 'class')
confusionMatrix(rf_pred, df_train[trainIndex,1])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21463  3451
##          1  1351  5411
##
##                Accuracy : 0.8484
##                  95% CI : (0.8444, 0.8523)
##     No Information Rate : 0.7202
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5944
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9408
##             Specificity : 0.6106
##          Pos Pred Value : 0.8615
##          Neg Pred Value : 0.8002
##              Prevalence : 0.7202
##          Detection Rate : 0.6776
##    Detection Prevalence : 0.7865
##       Balanced Accuracy : 0.7757
##
##        'Positive' Class : 0
##
```

Validation confusion matrix

```
rf_pred = predict(rf_fit, val_x)
confusionMatrix(rf_pred, val_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5340  886
##          1  365 1329
##
##                Accuracy : 0.842
##                  95% CI : (0.8338, 0.85)
##     No Information Rate : 0.7203
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5776
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.9360
##             Specificity : 0.6000
##          Pos Pred Value : 0.8577
##          Neg Pred Value : 0.7845
##              Prevalence : 0.7203
##          Detection Rate : 0.6742
##    Detection Prevalence : 0.7861
##       Balanced Accuracy : 0.7680
##
##        'Positive' Class : 0
##
```

Test set misclassification rate and confusion matrix

```
# make predictions
rf_pred = predict(rf_fit, df_test[,-1], type = 'class')
confusionMatrix(rf_pred, df_test[,1])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 45449 19537
##          1   777 13543
##
##                Accuracy : 0.7439
##                  95% CI : (0.7408, 0.7469)
##     No Information Rate : 0.5829
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.427
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9832
##             Specificity : 0.4094
##          Pos Pred Value : 0.6994
##          Neg Pred Value : 0.9457
##              Prevalence : 0.5829
##          Detection Rate : 0.5731
##    Detection Prevalence : 0.8194
##       Balanced Accuracy : 0.6963
##
##        'Positive' Class : 0
##
```