HW1

前言：

使用三種不同套件(tree、rpart、randomforest)，對給定資料切成 80%的 train 與 20%的 test 進行訓練(用 bus、mrt_bus、bus_interbus 三個 feature)建模之後測試並分析最後的準確率，為了避免抽樣有所影響，因此總共會做十次比較其平均與標準差。

程式碼：

```
# 導入相關套件
library(tree)
library(rpart)
library(randomForest)

# 讀入資料
setwd('C:/Users/Steven/Desktop/陽交109下/巨量資料分析/課程/單元2： 預測模式（一）/範例程式與資料')
data <- read.csv("MaaS_Data.csv",header=T)
head(data) # 看一下前幾筆資料

# 資料清洗和選取
data[data == ""] <- NA # 將空值以NA取代
head(data) # 確認一下空值是否都用NA補


# 刪除具有NA的資料並加以確認
data<-data[,-1] # ID對判斷是否購買沒幫助，予以刪除
num_na <- function(x){sum(is.na(x))}
sapply(data, num_na) # 對Data Frame的每一行(column)進行num_na函數運算，用來計算NA的數量，發現只有Buy那欄有NA
data <- data[!is.na(data$Buy),] # 將有NA值得資料刪除
sapply(data, num_na) # 確認是否還有空值
head(data) # 確認一下資料現在的樣子


# label轉成factor，用於分類
data$Buy <- as.factor(data$Buy)
```

以下的部分要跑十次

```
# 分割資料(train:80%, test:20%)(這個block要跑10次)
n <- 0.2*nrow(data)
index <- sample(1:nrow(data), n) # 用隨機取的方式
maas_train <- data[-index,]
maas_test <- data[index,]
```

Tree 的部分

```
# 利用tree套件建模
maas.tree <- tree(Buy ~ bus+mrt_bus+bus_interbus, data=maas_train)
plot(maas.tree)
text(maas.tree, cex=0.75)
# tree進行預測
tree.predict <- predict(maas.tree, maas_test, type="class")
# 計算tree預測的正確率
compare.tree <- ifelse(tree.predict == maas_test$Buy, 1, 0)
accuracy.tree <- sum(compare.tree)/ length(compare.tree)
accuracy.tree
```

rpart 的部分

```
# 利用rpart套件建模
maas.rpart <- rpart(Buy ~ bus+mrt_bus+bus_interbus, data=maas_train, cp=0)
plot(maas.rpart)
text(maas.rpart, cex=0.75)
# rpart進行預測
rpart.predict <- predict(maas.rpart, maas_test, type="class")
# 計算rpart預測的正確率
compare.rpart <- ifelse(rpart.predict == maas_test$Buy, 1, 0)
accuracy.rpart <- sum(compare.rpart)/ length(compare.rpart)
accuracy.rpart
```

randomForest 的部分

```
# 利用rf套件建模
maas.rf <- randomForest(Buy ~ bus+mrt_bus+bus_interbus, data=maas_train)
# rf進行預測
rf.predict <- predict(maas.rf, maas_test, type="class")
# 計算rf預測的正確率
compare.rf <- ifelse(rf.predict == maas_test$Buy, 1, 0)
accuracy.rf <- sum(compare.rf)/ length(compare.rf)
accuracy.rf
```

輸出結果：

	tree	rpart	randomForest
第 1 次	0.9403748	0.9369676	0.9344123
第 2 次	0.9531516	0.9505963	0.9497445
第 3 次	0.9386712	0.9267462	0.9344123
第 4 次	0.9361158	0.9352641	0.9369676
第 5 次	0.9241908	0.9241908	0.9224872
第 6 次	0.9318569	0.9241908	0.9224872
第 7 次	0.9403748	0.9361158	0.9335605
第 8 次	0.9429302	0.9429302	0.9412266
第 9 次	0.9378194	0.9352641	0.9344123
第 10 次	0.927598	0.9241908	0.9224872
平均	0.937308	0.933646	0.93322
標準差	0.008172	0.008892	0.008827

分析：

1.從結果來看，tree 最好、rpart 次之、最差的是 rf，跟事先預期的結果差不了多少。

2.因為 tree 是用完一個 feature 之後，往下長的過程中，用過的 feature 不會再用，加上本身無法設 cp=xxx，因此樹不會長的太複雜；

3.而 rpart 用過的 feature 仍有機會重複使用，加上設了 cp=0 沒有剪枝的關係，樹會長到無法再長為止，因此使得模型可能有些 overfitting，泛化能力較差；

4.而 randomForest，他採用的分類方法是 gini index，也就是屬於 rpart 的集成，表現與穩定性通常會比 rpart 好些，而 rf 本身也可以設 cp=xxx，但是本作業中沒有用到，但結果確是 rpart 比 rf 好一點。

5.而事實上本次實驗，三種模型表現的都差不多，因此準確率的高低也有可能

只是抽樣帶來的結果，並無好壞之分。

以下附上某次實驗中 tree 與 rpart 的圖，方便比較：