HW2

前言：

使用五種不同套件(tree、rpart、randomforest、svm、logistic regression)，對給定資料切成 80%的 train 與 20%的 test 進行訓練(這次採用除了 Buy 以外的所有 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)
library(e1071)

# 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（二）/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 ~ ., data=maas_train)
tree.predict <- predict(maas.tree, maas_test, type="class")
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 ~ ., data=maas_train)
rpart.predict <- predict(maas.rpart, maas_test, type="class")
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 ~ ., data=maas_train)
rf.predict <- predict(maas.rf, maas_test, type="class")
compare.rf <- ifelse(rf.predict == maas_test$Buy, 1, 0)
accuracy.rf <- sum(compare.rf)/ length(compare.rf); accuracy.rf
```

svm 的部分

```
# 利用svm套件建模
maas.svm <- svm(Buy ~ ., data=maas_train)
svm.predict <- predict(maas.svm, maas_test)
compare.svm <- ifelse(svm.predict == maas_test$Buy, 1, 0)
accuracy.svm <- sum(compare.svm)/ length(compare.svm); accuracy.svm
```

logistic regression 的部分

```
# 利用Logistics Regression套件建模
maas.logit <- glm(Buy ~ ., family=binomial(link='logit'), data=maas_train)
logit.predict <- predict(maas.logit, maas_test, type="response")
logit.results <- ifelse(logit.predict > 0.5, 2, 1)
compare.logit <- ifelse(logit.results == as.numeric(maas_test$Buy), 1, 0)
accuracy.logit <- sum(compare.logit)/ length(compare.logit); accuracy.logit
```

輸出結果：

	tree	rpart	randomForest	svm	logit
第 1 次	0.9412266	0.9412266	0.9369676	0.9412266	0.9412266
第 2 次	0.9250426	0.9250426	0.9156729	0.9241908	0.923339
第 3 次	0.9250426	0.9250426	0.9182283	0.923339	0.9216354
第 4 次	0.9403748	0.9403748	0.927598	0.939523	0.9378194
第 5 次	0.939523	0.939523	0.9224872	0.9386712	0.9378194
第 6 次	0.9344123	0.9344123	0.9267462	0.9335605	0.9318569
第 7 次	0.9369676	0.9369676	0.9250426	0.9369676	0.9369676
第 8 次	0.9369676	0.9369676	0.9284497	0.9369676	0.9361158
第 9 次	0.9386712	0.9386712	0.9250426	0.9386712	0.9378194
第 10 次	0.9412266	0.9412266	0.9293015	0.939523	0.9386712
平均	0.935945	0.935945	0.925554	0.935264	0.934327
標準差	0.006127	0.006127	0.005958	0.006399	0.006676

分析：

1.從結果來看，在準確率方面 tree 和 rpart 最好、而在變動程度方面，rf 最好。

2.因為本次打算所有模型都不設 cp=XXX，方便比較，發現十次下來，tree 跟 rpart 的表現完全一樣，可以合理推斷，雖然用的演算法不同，但他們生成的樹的模樣應該是完全一樣的，代表 gini 跟 entropy 都是可以拿來判斷亂度的。

3.而 rf 的穩定度最高也是合理的，因為它屬於多棵樹的生成，有用到平均的概念，因此每次的表現得差異應該都差不了太多。

4.而至於為什麼 tree 和 rpart 表現最好，我認為可能還是因為 data 本身不算太

複雜，所以用樹就可以表現得很好，如果 data 整個 feature 維度加 10 倍、資料量也加 10 倍，那可能這時 svm 或 logistic regression 表現可能會更佳。
5.而事實上本次實驗，雖然看似有高低，但五種模型的表現不會差太多，因此也有可能只是抽樣帶來的結果，可能需要做更多次來考證此想法。