HW5: Classification and Representation (due 1/9 23:30)

110024521 邱婉如 110024516 邱繼賢

Problem 1: Fault Classification

This problem concerns about the quality of a casting item. This data set collects the images (100x100 pixels) of 6633 items, together with their quality labels (defective or ok).

The objective is to predict the label (defect or ok) of each image.

Your analysis should include the following:

- The training procedures of the classifiers (at least 2 different classifiers), and their performance comparisons.
- A brief analysis summary.
- (Bonus) Can defect items (image data) be clustered with patterns (sub-types)? Explore your findings.

```
#install required packages
library("jpeg")
library("plyr")
library(randomForest)
library(e1071)
library(caret)
library(ROSE)
library(NbClust)
library(factoextra)
library(tsne)
library(knitr)
```

Read image into data frame

```
setwd("casting_100x100")

def_paths <- dir("def_front")
ok_paths <- dir("ok_front")

data_def = sapply(1:length(def_paths), function(x){
  tmp = readJPEG(paste0("def_front/", def_paths[x]))[,,1]
  tmp = as.vector(tmp)
  return(tmp)
})
```

```
data_ok = sapply(1:length(ok_paths), function(x){
  tmp = readJPEG(paste0("ok_front/", ok_paths[x]))[,,1]
  tmp = as.vector(tmp)
  return(tmp)
})

data = as.data.frame(rbind(t(data_def), t(data_ok)))
data$y = as.factor(c(rep(1,length(def_paths)), rep(0,length(ok_paths))))
dim(data)
```

```
## [1]  6633 10001
```

我們將每張圖片轉換成一個長度為 10000 的向量，存成一個資料結構。

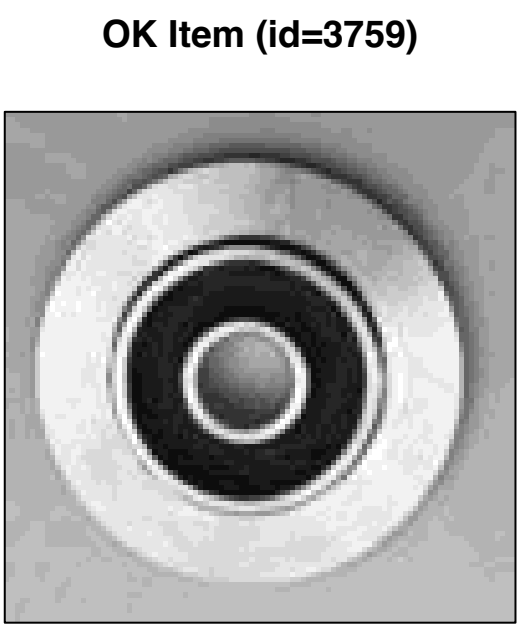
此資料共有 6633 筆，每一筆皆為長度為 10000 的向量再加上一個二元的類別變數：1 表示有鑄造缺陷；而 0 表示通過。

```
## Data Preprocessing

show_image = function(img_cast, col = gray(1:20/20), ...){
  image(matrix(as.matrix(img_cast), ncol = 100, byrow = TRUE)[, 100:1], col = col, ...)
}

## show images for a defect item and a good (OK) item
par(mfrow = c(1,1), pty="s")
show_image(data[1,-10001], col = gray(1:20/20), axes=F); box(); title("Defect Item (id=1)")
```

```
show_image(data[length(def_paths)+1,-10001], col = gray(1:20/20), axes=F); box(); title("OK Item (id=375
```



在開始分析前，我們先建構一個將數據轉回圖片的函數，取兩筆資料確認函數執行無誤，並且肉眼觀察了一下所謂有鑄造缺陷或是通過的圖片有何異同，進而意識到人工檢查既耗時又不一定準確，我想這也是我們希望可以將檢查過程自動化，藉由模型來分類的主要原因。

Your analysis starts here...

Preprocess

```
range(data[,-10001])
```

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

前面將灰階圖片轉成的向量數值皆介於 0 到 1 之間。

```
summary(data$y)
```

```
##    0    1
## 2875 3758
```

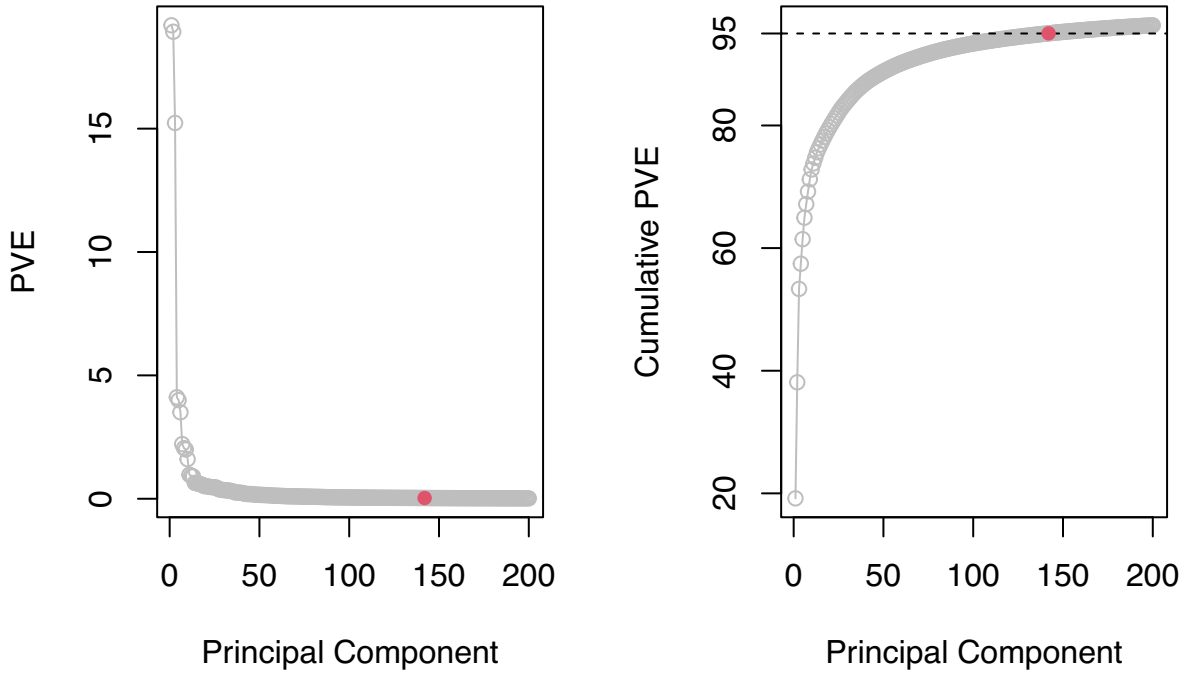
共 3758 筆有鑄造缺陷；2875 為通過。

首先將資料以 8 : 2 的比例切分為 training data 以及 testing data。

```
set.seed(1)
train.idx = sample(1:nrow(data), 0.8*nrow(data))
dat_train = data[train.idx,]
dat_test = data[-train.idx,]
```

處理這筆資料最大的問題就是變數很多，所以我想先利用 PCA 做降維的動作。

```
pca.out = prcomp(dat_train[,-10001], scale=TRUE)
pve = 100*pca.out$sdev^2/sum(pca.out$sdev^2)
cutpt = which((cumsum(pve[1:1000])>95)==T)[1] #142
par(mfcol=c(1,2))
plot(pve[1:200], type="o", ylab="PVE",
    xlab="Principal Component", col="gray")
points(cutpt, pve[cutpt], pch=16, col=2)
plot(cumsum(pve[1:200]), type = "o", ylab = "Cumulative PVE",
    xlab="Principal Component", col="gray")
abline(h=95, lty=2); axis(2,at=95)
points(cutpt, cumsum(pve)[cutpt], pch=16, col=2)
```



保險起見我有對變數做 scaling，然後畫出 PC 的解釋比例以及累積解釋比例，由於個數較多，為了呈現清楚，我只畫出前 200 個的變化。

上圖可見我們可以只使用 142 個 PC 以保留 95% 的累積解釋比例，並將降維後的新資料儲存如下：

```
loading = as.matrix(pca.out$rotation[,1:cutpt])
new_train = data.frame(as.matrix(dat_train[,-10001])%*%loading,
                       y=dat_train$y)
new_test = data.frame(as.matrix(dat_test[,-10001])%*%loading,
                      y=dat_test$y)
```

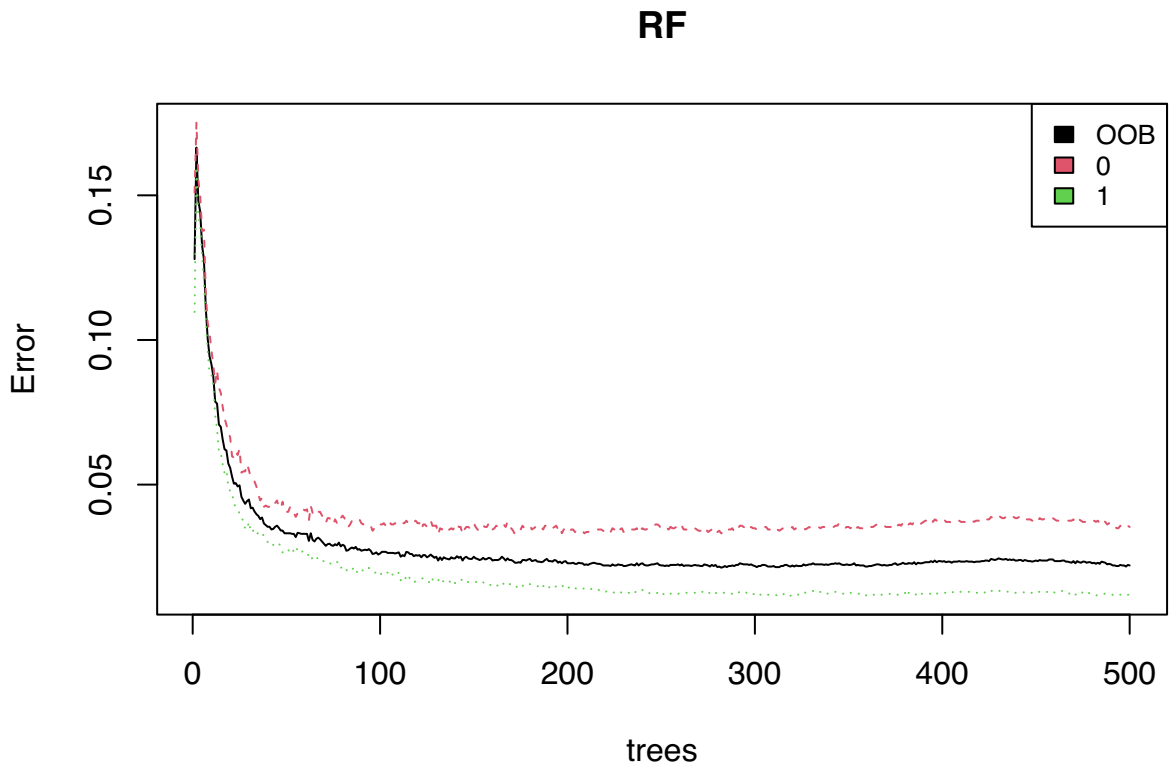
Classifiers

接下來我會 train 兩個分類器：Random Forest 以及 SVM。

RF

首先調整參數：*ntree* 以及 *mtry*。

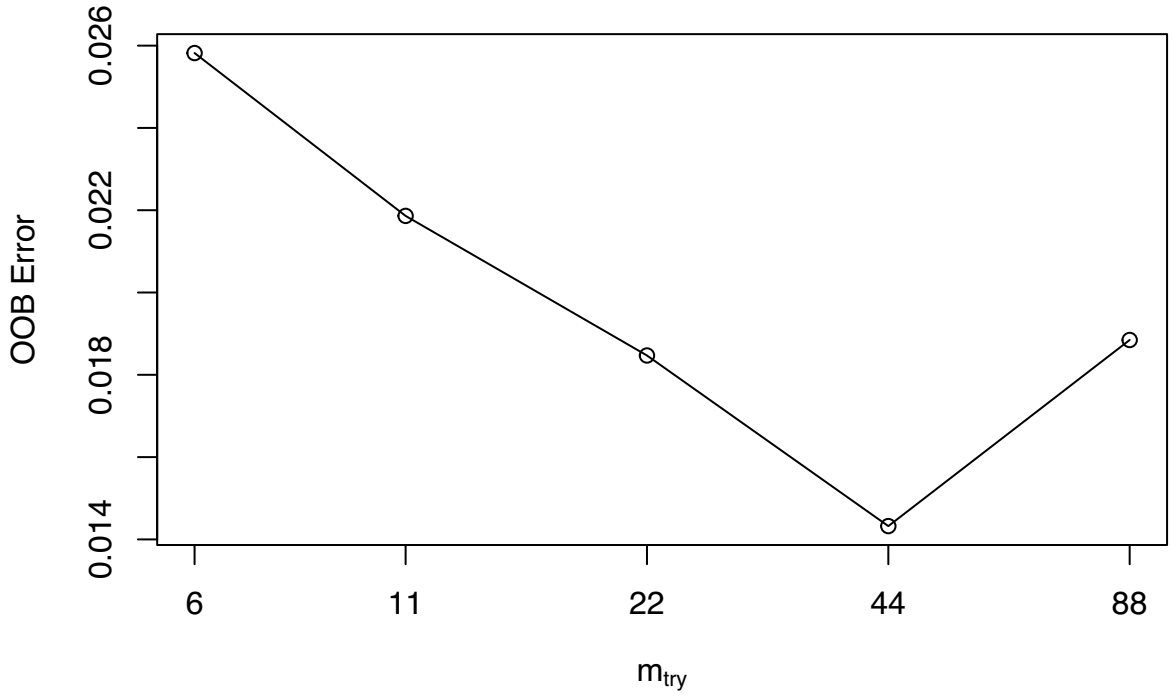
```
set.seed(1)
RF = randomForest(y~., new_train)
plot(RF)
legend("topright", colnames(RF$err.rate),
       col=1:4,cex=0.8,fill=1:4)
```



ntree 大概在 300 之後就趨於穩定。

```r
set.seed(1)
tuneRF(new_train[,-143], new_train[,143], ntreeTry=300)
```

```
## mtry = 11  OOB error = 2.19%
## Searching left ...
## mtry = 6     OOB error = 2.58%
## -0.1810345 0.05
## Searching right ...
## mtry = 22    OOB error = 1.85%
## 0.1551724 0.05
## mtry = 44    OOB error = 1.43%
## 0.2244898 0.05
## mtry = 88    OOB error = 1.88%
## -0.3157895 0.05
```




```
##        mtry   OOBError
## 6.OOB     6 0.02581983
## 11.OOB   11 0.02186204
## 22.OOB   22 0.01846966
## 44.OOB   44 0.01432341
## 88.OOB   88 0.01884659
```

mtry 選定 44 使得 OOB error 較小。

調整完參數後建模：

```
fit_RF = randomForest(y~., new_train, ntree=300, mtry=44, importance=T)
```

SVM

對此資料我選擇直接嘗試做 nonlinear 的 SVM，那麼常用的 kernel 有 radial 以及 polynomial 兩種。

其實我有先做過 radial 但預測表現太差 (ACC 大概 0.6 左右)，所以以下只呈現 kernel 使用 polynomial 的 SVM。

首先利用 10-fold CV 調整參數：*cost* 以及 *gamma*。

```
set.seed(1)
tune.out = tune(svm, y~., data=new_train, kernel="polynomial",
                ranges=list(cost=seq(5,10,length=10),
                            gamma=c(10:20)*0.01))
tune.out$best.parameters
```

```
##   cost gamma
## 1    5   0.1
```

選定上述參數後建模：

```
fit_SVM = svm(y~., data=new_train, kernel="polynomial",
              cost=tune.out$best.parameters[1,1],
              gamma=tune.out$best.parameters[1,2],
              probability=T)
```

Performance comparison

以上我 fit 了兩種模型，接下來利用這兩種模型對 testing data 做預測，並將預測表現整理如下：

```
RF.pred = predict(fit_RF, new_test)
#confusionMatrix(RF.pred, new_test$y, positive="1")
ACC_RF = mean(new_test$y==RF.pred)
RF.prob = predict(fit_RF, new_test, type="prob")[,2]
AUC_RF = roc.curve(new_test$y, RF.prob, plotit=F)$auc
SVM.pred = predict(fit_SVM, new_test, probability=T)
#confusionMatrix(SVM.pred, new_test$y, positive="1")
ACC_SVM = mean(new_test$y==SVM.pred)
SVM.prob= attr(SVM.pred,"probabilities")[,1]
AUC_SVM = roc.curve(new_test$y, SVM.prob, plotit=F)$auc
tab = data.frame(Model=c("RF","SVM"),
                 ACC=c(ACC_RF,ACC_SVM),
                 AUC=c(AUC_RF,AUC_SVM))
kable(tab)
```

Model	ACC	AUC
RF	0.9841748	0.9944286
SVM	0.9849284	0.9947063

可以看到 RF 跟 SVM 在這邊表現真的非常相當，都預測得極好！

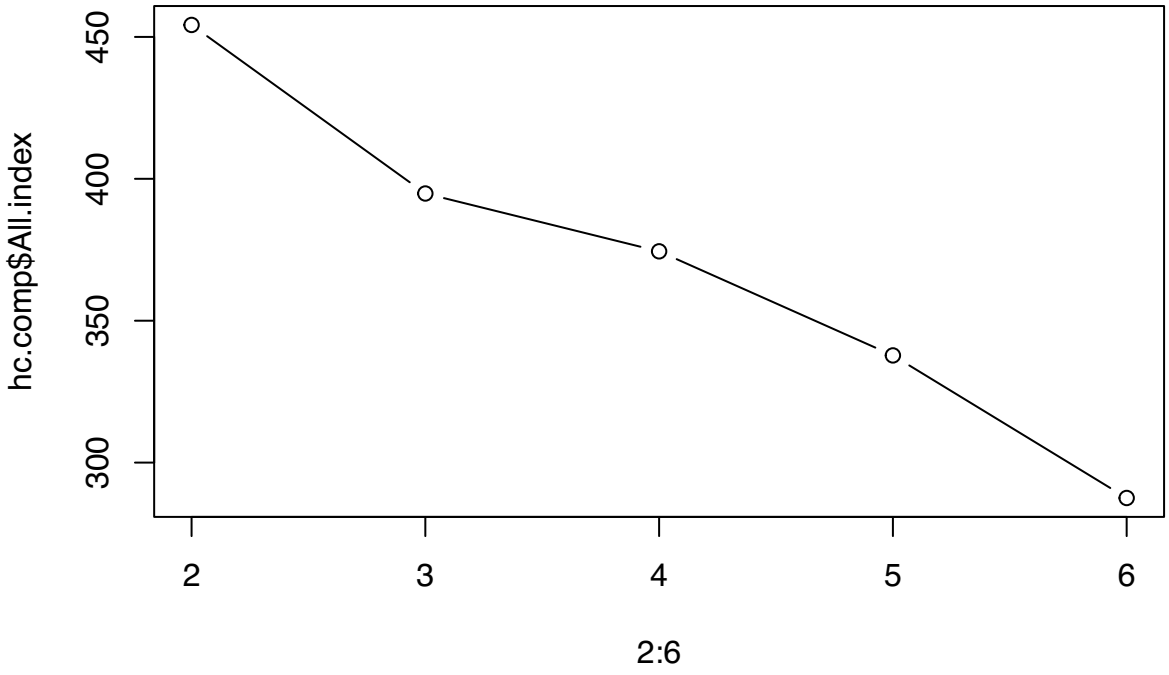
Clustering

前面利用建構分類器來判別有無鑄造缺陷，那接下來我們還想處理另一個議題，也就是能否在有鑄造缺陷的那些圖片中歸納出所謂的鑄造缺陷有哪些種類。

首先對於有鑄造缺陷的 3758 張圖片，我們同樣只使用前面那 142 個 PC 的資訊，然後做 complete linkage 的 hierarchical clustering。

先利用 CH index 來決定要分成幾群：

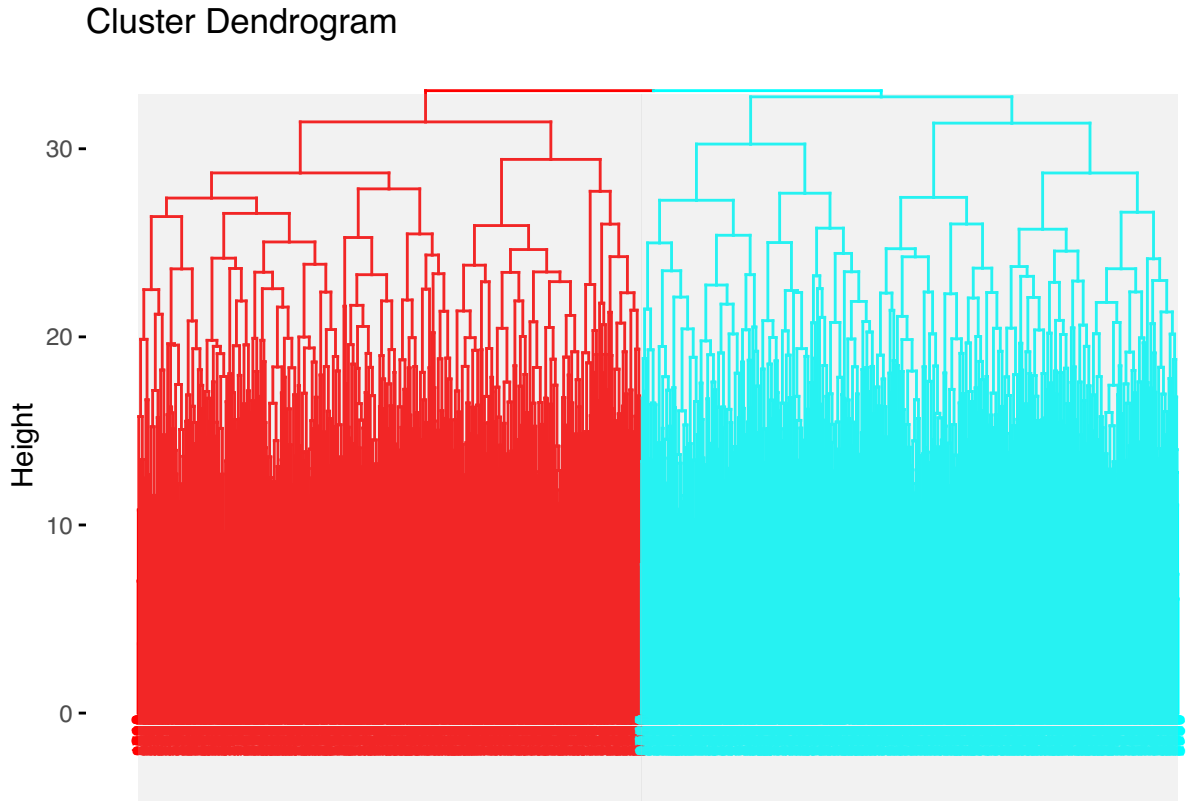
```
defect_dat = data[data$y==1,-10001]
defect_pcdat = data.frame(as.matrix(defect_dat)%*%loading)
hc.comp = NbClust(defect_pcdat, distance="euclidean",
                  min.nc=2, max.nc=6, method="complete", index="ch")
plot(2:6, hc.comp$All.index, type="b")
```



由圖可見我們可以分成兩群就好了。

接著來看看分成兩群的 Dendrogram：

```
dist.defect_pcdat = dist(defect_pcdat)
fviz_dend(x=hclust(dist.defect_pcdat), cex=0.5, lwd=0.5,
          k=2, k_colors=rainbow(2),
          rect=TRUE, rect_border="gray", rect_fill=T)
```



可以看到兩類的比例蠻平衡的。

```
cluster_dat = data.frame(defect_dat, "cluster"=hc.comp$Best.partition)
cluster_dat$cluster = factor(cluster_dat$cluster)
summary(cluster_dat$cluster)
```

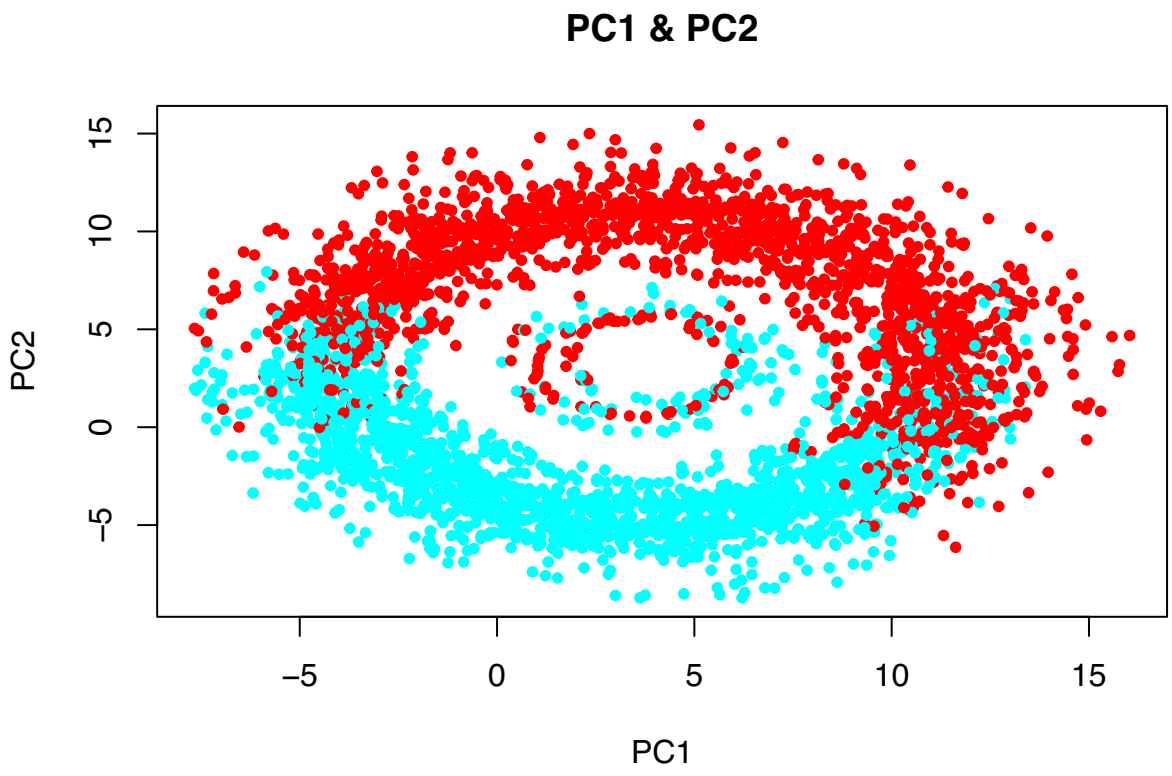
```
##    1    2
## 1940 1818
```

有一類包含 1940 張的圖片，而另一類包含 1818 張。

接著我們把資料降到二維平面上來看能不能呈現出我們的分群效果。

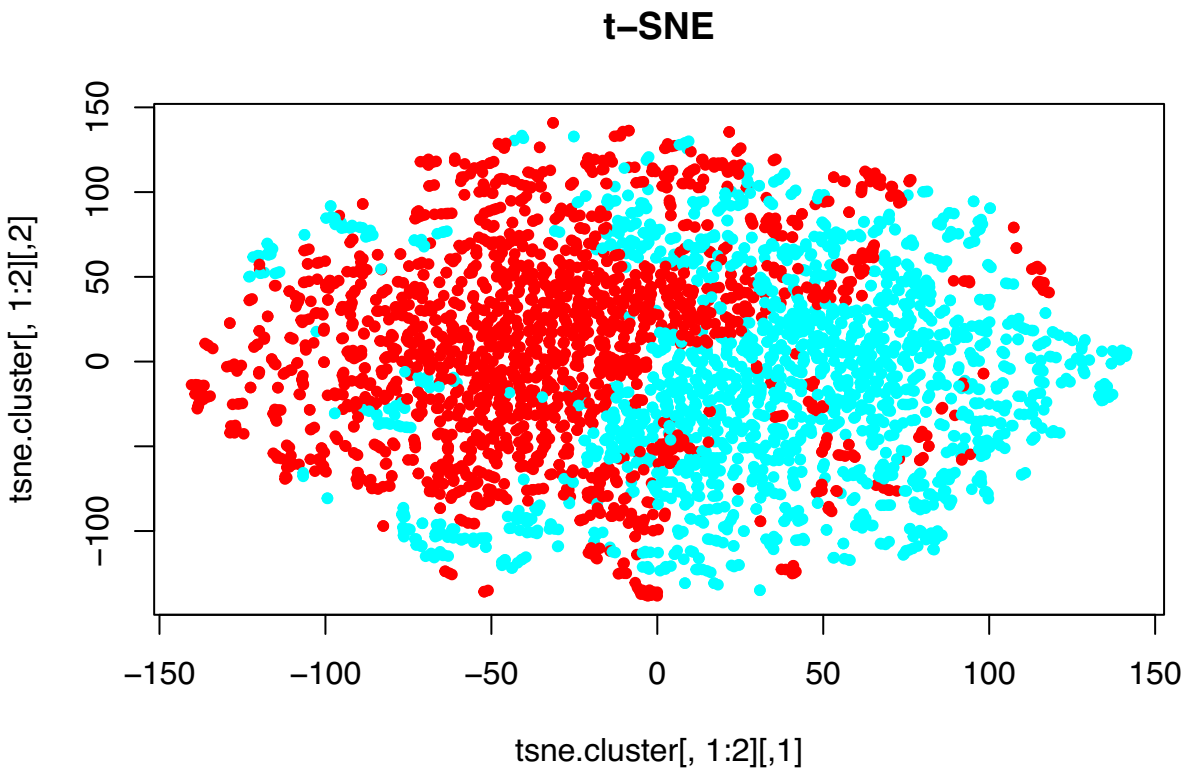
這是在 PC1 跟 PC2 的平面：

```
cluster_pcdat = data.frame(defect_pcdat, "cluster"=hc.comp$Best.partition)
colors = rainbow(2); names(colors) = 1:2
par(mfcol=c(1,1))
plot(cluster_pcdat[,1:2], t='n', main="PC1 & PC2")
points(cluster_pcdat[,1:2], col=colors[cluster_pcdat$cluster], pch=20)
```



這是用 t-SNE 降到二維的呈現：

```
tsne.cluster = tsne(cluster_pcdat[,-143], k=2,
            perplexity=100, initial_dims=50, max_iter=2000)
par(mfcol=c(1,1))
plot(tsne.cluster[,1:2], t='n', main="t-SNE")
points(tsne.cluster[,1:2], col=colors[cluster_pcdat$cluster], pch=20)
```

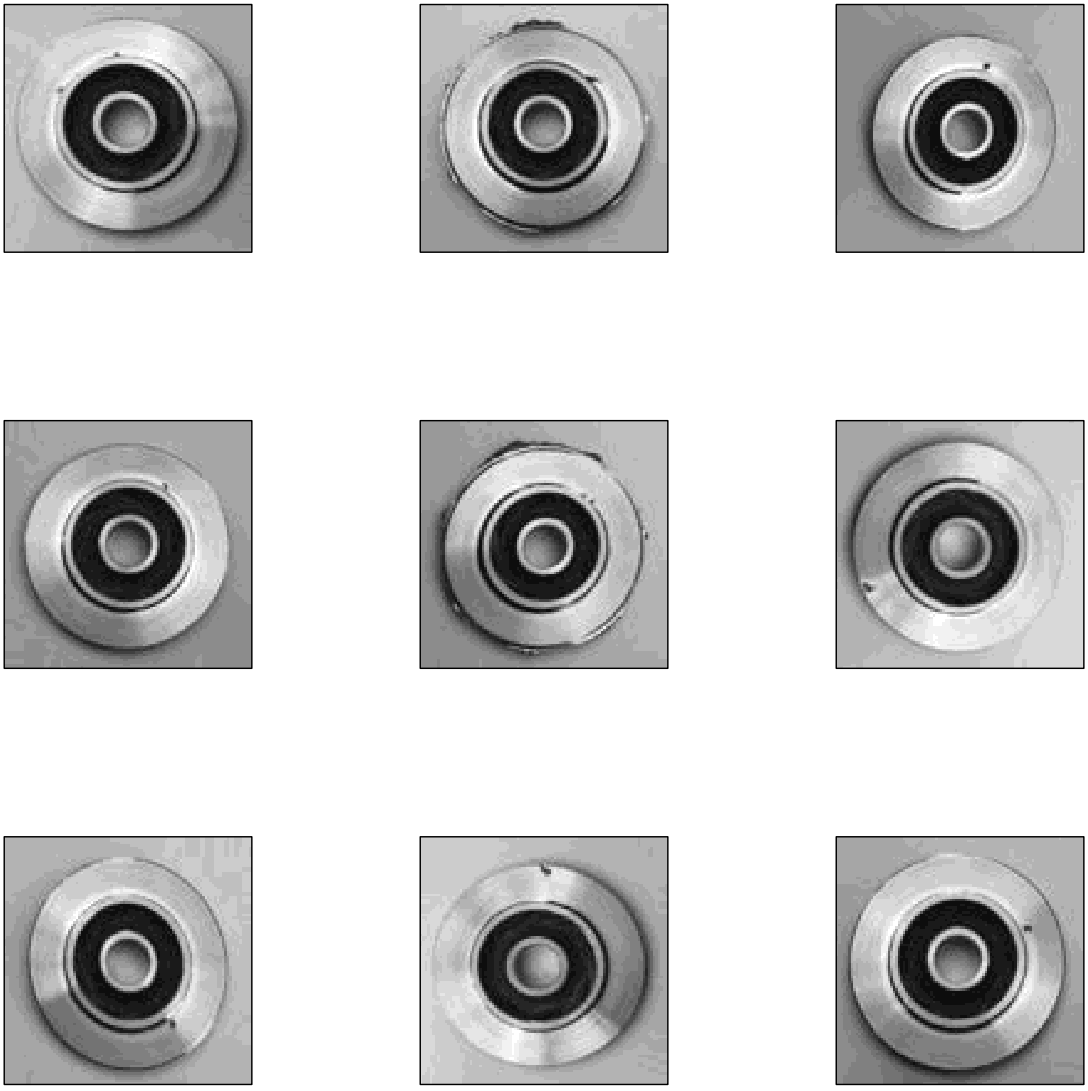


不論是在 biplot 或 t-SNE 視覺化的結果都可以看到，儘管只是在二維平面上呈現，我們的分群效果貌似是蠻不錯的。

接下來我們再從這兩類分別抽出 9 張圖片來觀察有什麼不一樣的特徵。

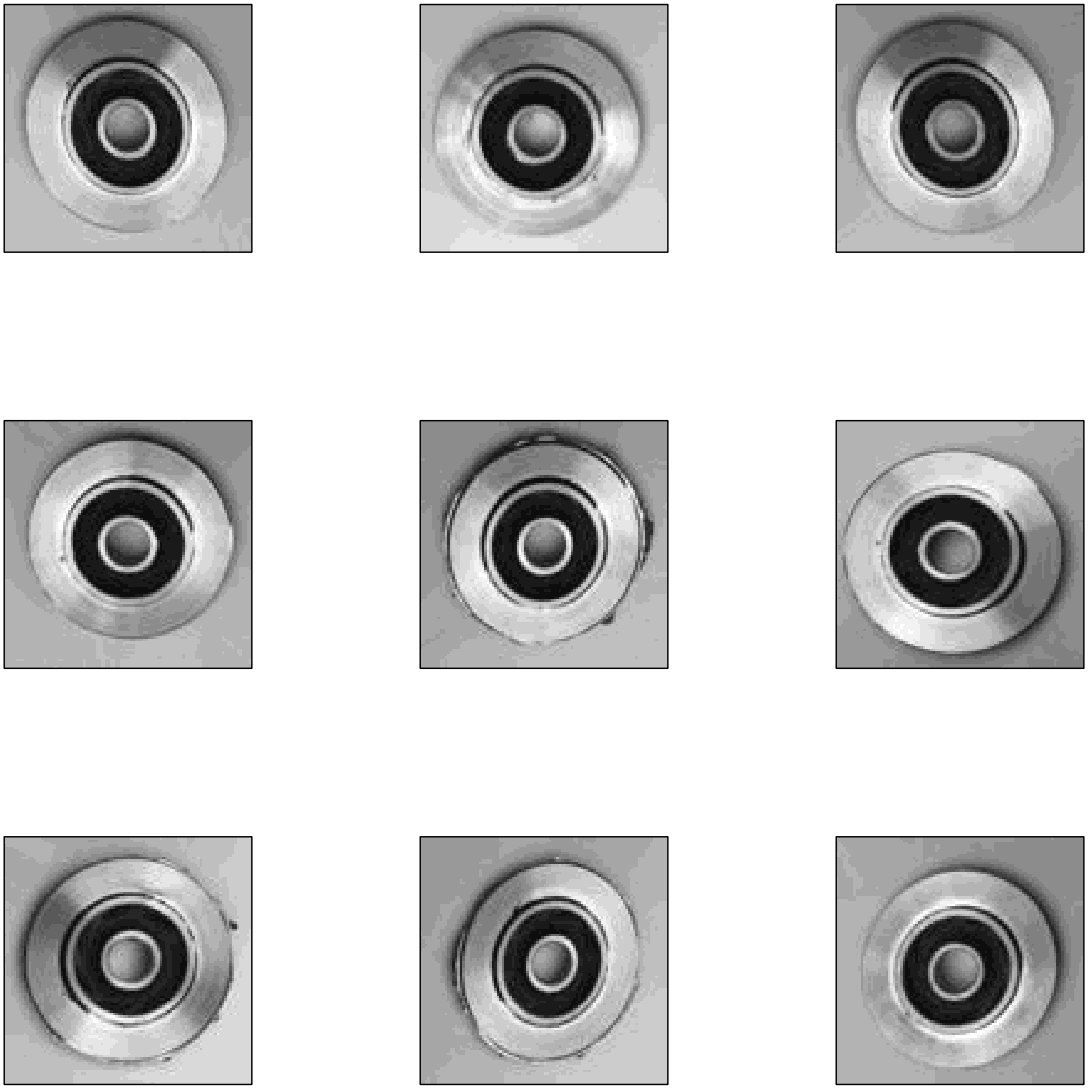
這是較大那一類：

```
cluster1 = cluster_dat[cluster_dat$cluster==1,-10001]
set.seed(731)
samp.1 = sample(1:nrow(cluster1), 9)
par(mfrow = c(3,3), pty="s")
for (i in samp.1) {
  show_image(cluster1[i,], col=gray(1:20/20), axes=F); box()
}
```



這是另一類：

```
cluster2 = cluster_dat[cluster_dat$cluster==2,-10001]
set.seed(18)
samp.2 = sample(1:nrow(cluster2), 9)
par(mfrow = c(3,3), pty="s")
for (i in samp.2) {
  show_image(cluster2[i,], col=gray(1:20/20), axes=F); box()
}
```



好...我看不出明顯的差異...硬要說的話，較大那一類表面都有明顯的破洞，而另一類則以盤面有凹折為主。

不過我們還是認為這兩群的解釋需要交給專業人員來判斷...

Problem 2: Forecast for SP500 returns

This problem concerns about 2 tasks on forecasting the return for SP500:

- Task 1 (regression): forecast the daily return value;
- Task 2 (classification): forecast the trend ("up" if return greater than zero; or "down" otherwise).

The data can be retrieved from yahoo into R via the function `getSymbols` in `quantmod` package. In this homework, we take the data before 2021/12/31 as the training data; and the data after 2022/1/1 as the testing data.

You may replace the SP500 returns by any other return series of your interest for this problem.

Your analysis should include the following:

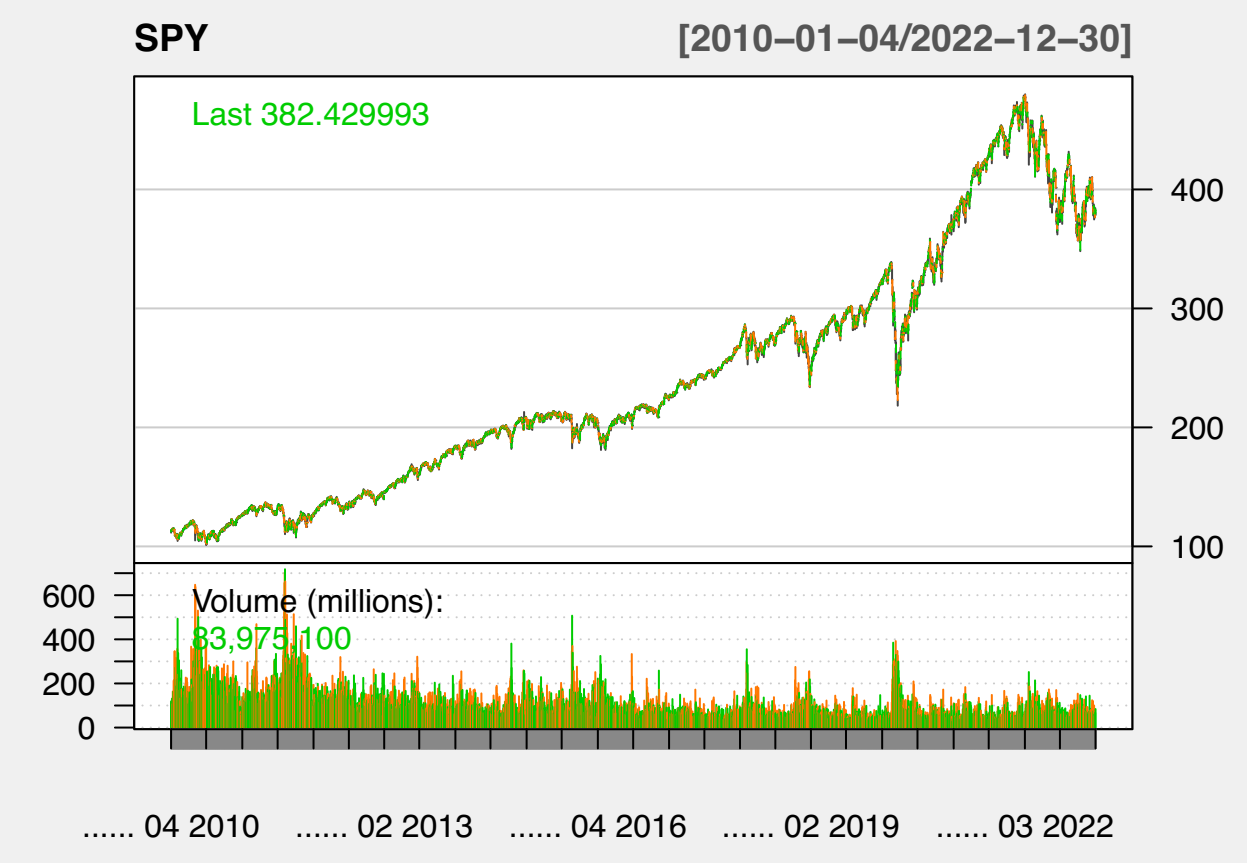
- The training procedures of the predictive models for both tasks. (training period: before 2022)
- Summarize the prediction performance in the testing period (2022/1/1-2022/12/31).

```
library(quantmod)
library(randomForest)
library(caret)
library(gbm)
library(latex2exp)
library(e1071)
library(ROSE)
```

```
set.seed(01025)
getSymbols("SPY", src = "yahoo", from = as.Date("2010-01-01"),
           to = as.Date("2022-12-31"))
```

```
## [1] "SPY"
```

```
chartSeries(SPY, theme = chartTheme("white"))
```



載入 SPY 從 2010-01-01 到 2022-12-31 的全部資料，並將 2022 年以前的資料當作 train data 訓練預測模型，2022 年以後的資料當作 test data 用來測試模型表現：

Task 1 (regression) : forecast the daily return value

```
# compute daily returns:
rr <- exp(diff(log(SPY$SPY.Close)))-1
Direction <- ifelse(rr >= 0, "Up", "Down")
names(Direction) <- "Direction"

# create lag variables (you may use more lags or other info from SPY object)
rr.Lag <- Lag(rr, 1:5)
Volume.Lag <- Lag(SPY$SPY.Volume,1:5)
Adjust.Lag <- Lag(SPY$SPY.Adjusted,1:5)

#For Task 1 (respones variable: rr):
SPY.return <- data.frame(rr, rr.Lag, Volume.Lag/10^9, Adjust.Lag/10^2) #rescale Volume such that all sc
names(SPY.return) <- c("r", "r.Lag.1", "r.Lag.2", "r.Lag.3", "r.Lag.4", "r.Lag.5",
                           "v.Lag.1", "v.Lag.2", "v.Lag.3", "v.Lag.4", "v.Lag.5",
                           "a.Lag.1", "a.Lag.2", "a.Lag.3", "a.Lag.4", "a.Lag.5")

head(SPY.return)
```

```
##                      r     r.Lag.1     r.Lag.2     r.Lag.3     r.Lag.4 r.Lag.5
```

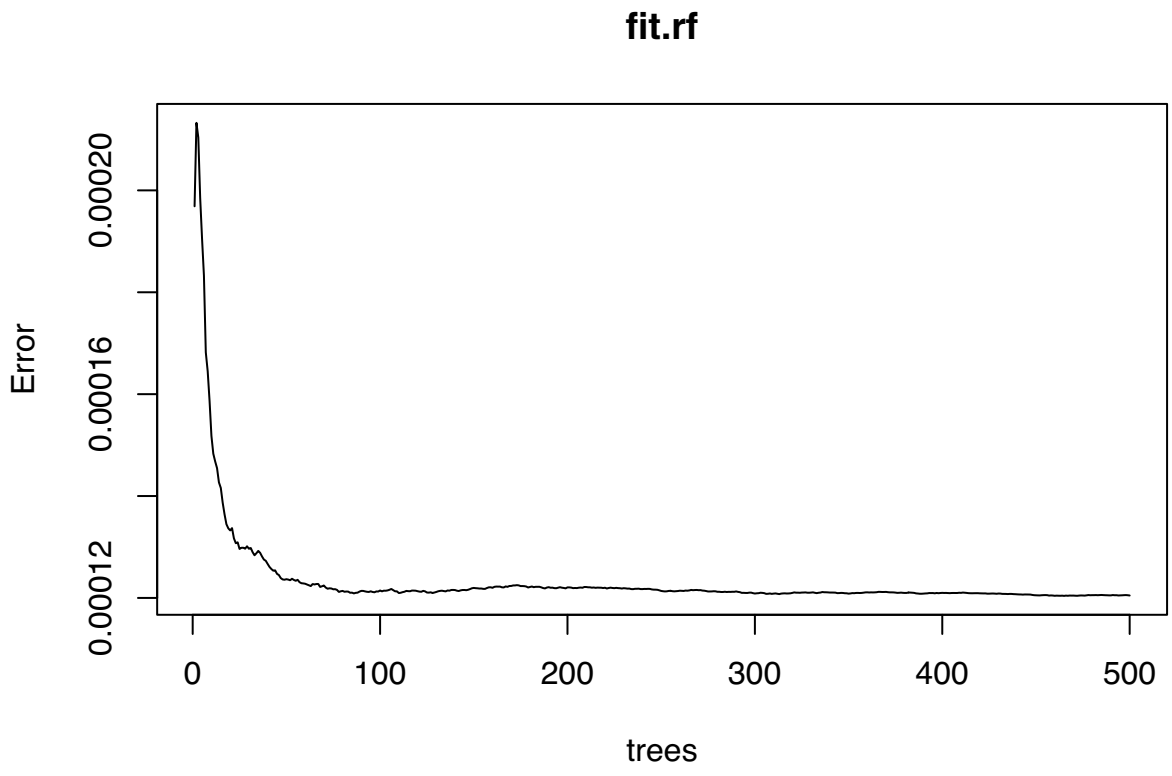
```
## 2010-01-04          NA          NA          NA          NA          NA      NA
## 2010-01-05 0.002647093          NA          NA          NA          NA      NA
## 2010-01-06 0.000704057 0.002647093          NA          NA          NA      NA
## 2010-01-07 0.004221291 0.000704057 0.002647093          NA          NA      NA
## 2010-01-08 0.003327769 0.004221291 0.000704057 0.002647093          NA      NA
## 2010-01-11 0.001396552 0.003327769 0.004221291 0.000704057 0.002647093      NA
##              v.Lag.1   v.Lag.2   v.Lag.3   v.Lag.4   v.Lag.5   a.Lag.1
## 2010-01-04        NA        NA        NA        NA        NA        NA
## 2010-01-05 0.1189446        NA        NA        NA        NA 0.8845421
## 2010-01-06 0.1115799 0.1189446        NA        NA        NA 0.8868836
## 2010-01-07 0.1160744 0.1115799 0.1189446        NA        NA 0.8875082
## 2010-01-08 0.1310911 0.1160744 0.1115799 0.1189446        NA 0.8912544
## 2010-01-11 0.1264028 0.1310911 0.1160744 0.1115799 0.1189446 0.8942203
##              a.Lag.2   a.Lag.3   a.Lag.4   a.Lag.5
## 2010-01-04        NA        NA        NA        NA
## 2010-01-05        NA        NA        NA        NA
## 2010-01-06 0.8845421        NA        NA        NA
## 2010-01-07 0.8868836 0.8845421        NA        NA
## 2010-01-08 0.8875082 0.8868836 0.8845421        NA
## 2010-01-11 0.8912544 0.8875082 0.8868836 0.8845421
```

計算每日 return 當作預測模型的 response，1~5 天前的 return, Volume, Adjusted 都當作預測模型中的 predictors

```
idx_before_2022 = (rownames(SPY.return) < strptime("2022-01-01","%Y-%m-%d"))
SPY.return_train = SPY.return[idx_before_2022,]
SPY.return_train = na.omit(SPY.return_train)
SPY.return_test = SPY.return[!idx_before_2022,]
```

在 train data 上訓練 randomforest model

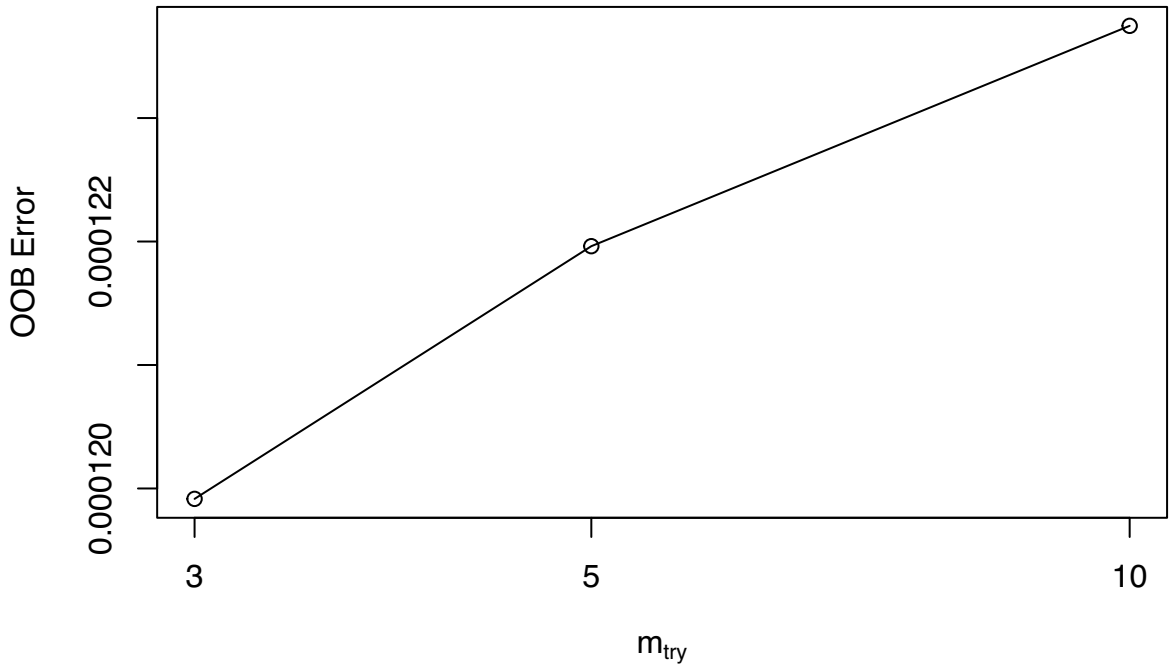
```
set.seed(0102)
fit.rf = randomForest(r ~ ., SPY.return_train)
plot(fit.rf) # ntrees = 250
```



$\Rightarrow$ $n.tree$ $>$ 250 後 error 趨於穩定

```
set.seed(1)
tune_RF1 = tuneRF(SPY.return_train[,-1],SPY.return_train[,1], ntreeTry = 250)
```

```
## mtry = 5  OOB error = 0.0001219623
## Searching left ...
## mtry = 3     OOB error = 0.0001199158
## 0.01677966 0.05
## Searching right ...
## mtry = 10    OOB error = 0.0001237467
## -0.01463133 0.05
```

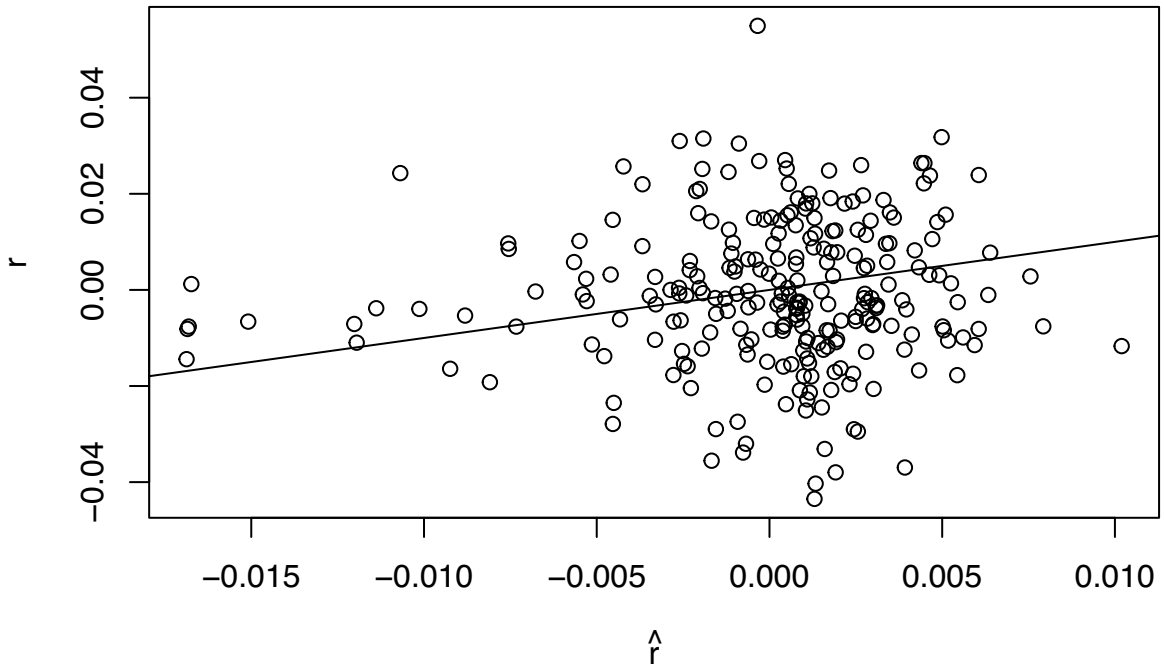


⇒ 在 OOB error 最小時決定參數 *mtry*

利用上述設定的參數建構 randomforest model，然後對 test data 進行預測

```
set.seed(0101)
m = tune_RF1[which.min(tune_RF1[,2]),1]
fit.rf_ = randomForest(r ~ ., SPY.return_train, ntree = 250, mtry = m, importance = T)
pred_return = predict(fit.rf_, SPY.return_test)
```

```
plot(pred_return, SPY.return_test$r, xlab = TeX("$\\hat{r}$"), ylab = "r")
abline(0,1)
```



預測效果看起來並不是很好

```
mean((pred_return-SPY.return_test$r)^2)
```

```
## [1] 0.0002440315
```

MSE 會受到 return 的單位所影響，因此計算出數值才看似很小，但實際預測效果並不理想

Task 2 (classification) : forecast the trend

```
#For Task 2 (response variable: Direction):
SPY.trend <- data.frame(factor(Direction), rr.Lag, Volume.Lag/10^9, Adjust.Lag/10^2)
names(SPY.trend) <- c("Direction", "r.Lag.1", "r.Lag.2", "r.Lag.3",  "r.Lag.4", "r.Lag.5",
                                   "v.Lag.1", "v.Lag.2", "v.Lag.3",  "v.Lag.4", "v.Lag.5",
                                   "a.Lag.1", "a.Lag.2", "a.Lag.3", "a.Lag.4", "a.Lag.5")

head(SPY.trend)
```

```
##            Direction     r.Lag.1     r.Lag.2     r.Lag.3     r.Lag.4 r.Lag.5
## 2010-01-04      <NA>          NA          NA          NA          NA      NA
## 2010-01-05        Up          NA          NA          NA          NA      NA
## 2010-01-06        Up 0.002647093          NA          NA          NA      NA
## 2010-01-07        Up 0.000704057 0.002647093          NA          NA      NA
```

```
## 2010-01-08        Up 0.004221291 0.000704057 0.002647093          NA      NA
## 2010-01-11        Up 0.003327769 0.004221291 0.000704057 0.002647093      NA
##              v.Lag.1   v.Lag.2   v.Lag.3   v.Lag.4   v.Lag.5   a.Lag.1
## 2010-01-04        NA        NA        NA        NA        NA        NA
## 2010-01-05 0.1189446        NA        NA        NA        NA 0.8845421
## 2010-01-06 0.1115799 0.1189446        NA        NA        NA 0.8868836
## 2010-01-07 0.1160744 0.1115799 0.1189446        NA        NA 0.8875082
## 2010-01-08 0.1310911 0.1160744 0.1115799 0.1189446        NA 0.8912544
## 2010-01-11 0.1264028 0.1310911 0.1160744 0.1115799 0.1189446 0.8942203
##              a.Lag.2   a.Lag.3   a.Lag.4   a.Lag.5
## 2010-01-04        NA        NA        NA        NA
## 2010-01-05        NA        NA        NA        NA
## 2010-01-06 0.8845421        NA        NA        NA
## 2010-01-07 0.8868836 0.8845421        NA        NA
## 2010-01-08 0.8875082 0.8868836 0.8845421        NA
## 2010-01-11 0.8912544 0.8875082 0.8868836 0.8845421
```

將每日 return 以 0 為分界區分為兩類：$Direction = Up$ or Down，當作 classification model response，一樣以 1~5 日前的 return, Volume, Adjusted 當作分類模型的 predictors

```
SPY.trend_train = SPY.trend[idx_before_2022,]
SPY.trend_train = na.omit(SPY.trend_train)
SPY.trend_test = SPY.trend[!idx_before_2022,]
```

嘗試利用 SVM (with radial kernel function) 建構 classification model，並利用 10-fold CV 選取參數 *cost*, *gamma*

```
set.seed(01021)
tune_svm_radial = tune(svm, Direction ~ ., data = SPY.trend_train,
                kernel = "radial",
                ranges = list(cost = c(0.1, 1, 10, 100, 1000),
                              gamma = c(0.5, 1, 2, 3, 4)
                            ))
```

```
tune_svm_radial$best.parameters
```

```
##   cost gamma
## 1  0.1   0.5
```

$\Rightarrow cost = 0.1$, $gamma = 0.5$

以此參數設定建構 SVM model，然後對 test data 進行分類預測

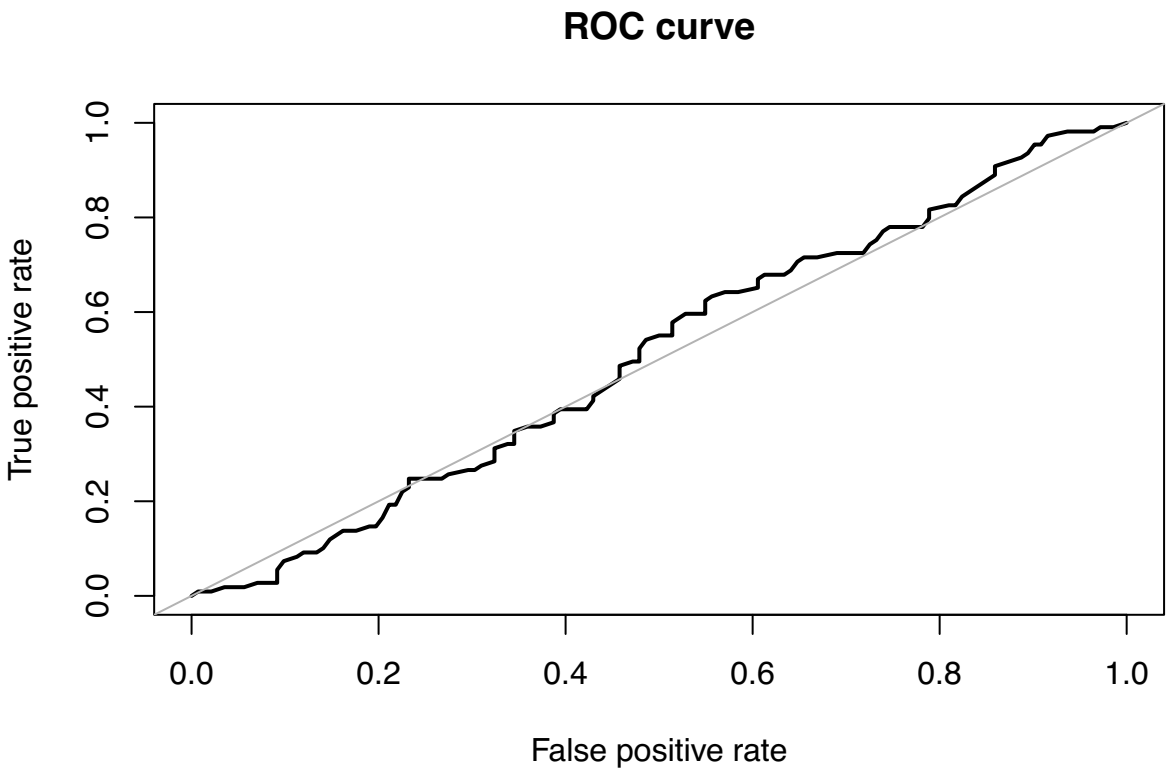
```
set.seed(01022)
fit.svm_radial = svm(Direction ~., data = SPY.trend_train, kernel = "radial",
              cost = 0.1, gamma = 0.5, probability = T, scale = F)
pred_trend.svm_radial = predict(fit.svm_radial, SPY.trend_test)
confusionMatrix(pred_trend.svm_radial, SPY.trend_test$Direction, positive = "Up")
```

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

```
## Prediction Down  Up
##       Down    0   0
##       Up    142 109
##
##                Accuracy : 0.4343
##                  95% CI : (0.3721, 0.498)
##     No Information Rate : 0.5657
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.4343
##          Neg Pred Value :    NaN
##              Prevalence : 0.4343
##          Detection Rate : 0.4343
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Up
##
```

```
pred_trend_prob = predict(fit.svm_radial, SPY.return_test, probability = T)
pred_prob = attr(pred_trend_prob, "probabilities")[,1]
```

```
roc.curve(SPY.trend_test$Direction, pred_prob, plotit = T)
```



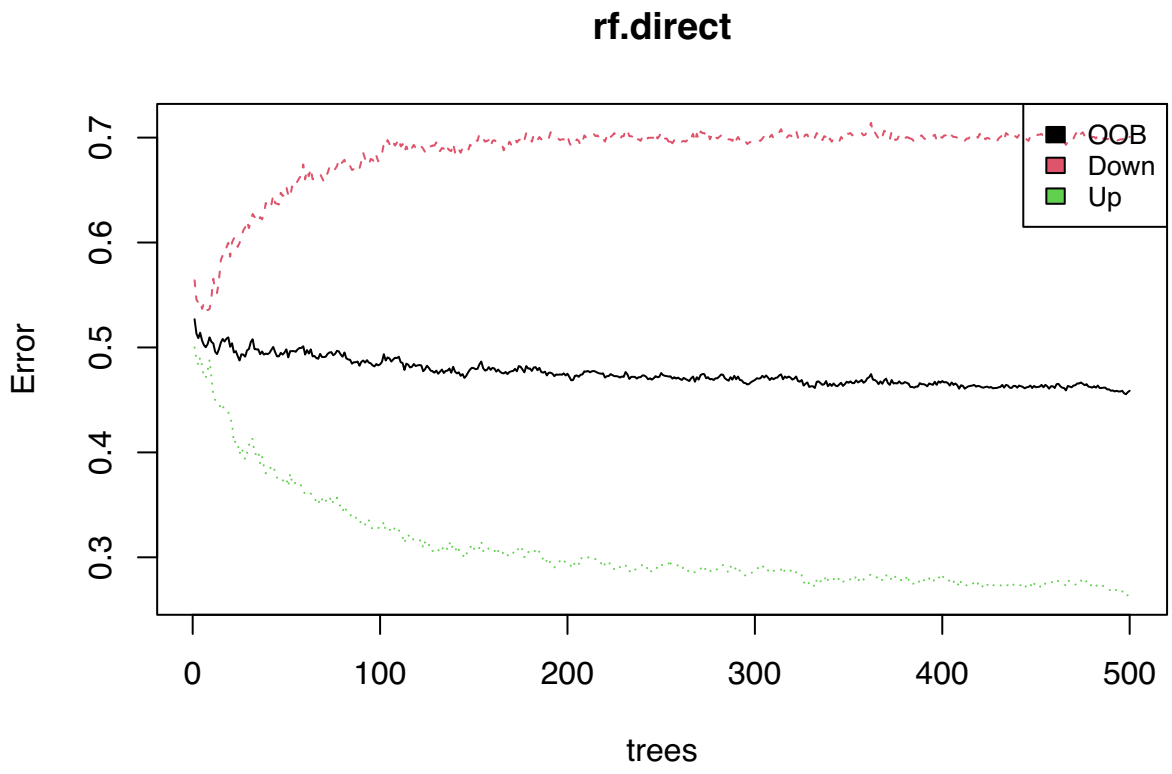

```
## Area under the curve (AUC): 0.510
```

觀察 confusion matrix 和 ROC curve 以及計算出的 ACC, AUC，模型預測表現並不好，而且 SVM 模型預測所有 $Direction = Up$

接下來嘗試 randomforest model

```
set.seed(01023)
rf.direct = randomForest(Direction ~ ., SPY.trend_train)
```

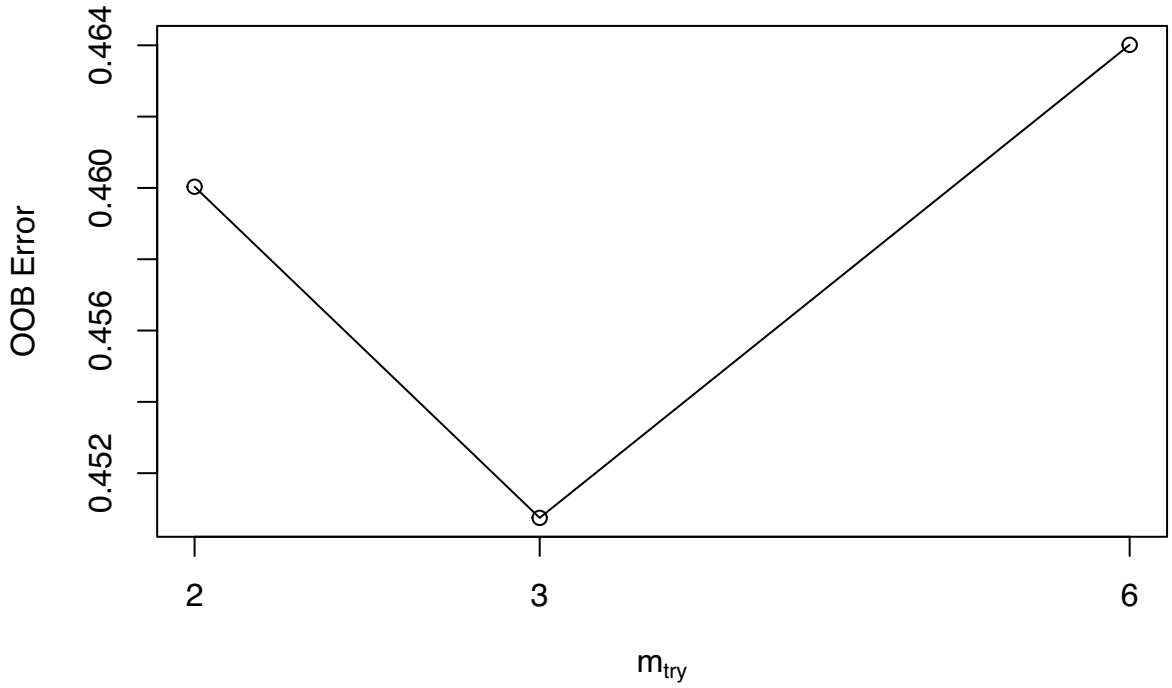
```
plot(rf.direct) # n.tree = 300
legend("topright", colnames(rf.direct$err.rate),col=1:4,cex=0.8,fill=1:4)
```



$\Rightarrow n.tree > 300$ 後趨於穩定

```
set.seed(2)
tune_RF2 = tuneRF(SPY.trend_train[,-1],SPY.trend_train[,1], ntreeTry = 300)
```

```
## mtry = 3  OOB error = 45.07%
## Searching left ...
## mtry = 2     OOB error = 46%
## -0.02060338 0.05
## Searching right ...
## mtry = 6     OOB error = 46.4%
## -0.02943341 0.05
```



⇒ 在 OOB error 最小時決定參數 *mtry*

以此參數設定建構 randomforest model，然後對 test data 進行預測

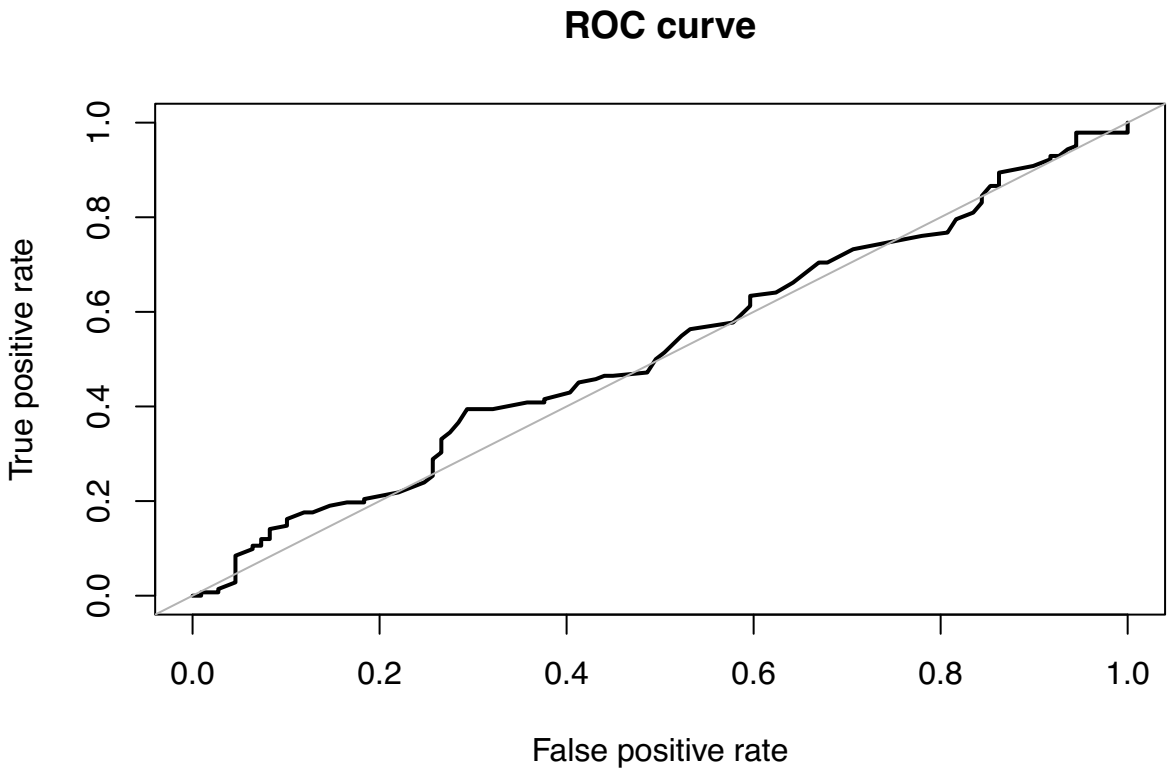
```
set.seed(01024)
m = tune_RF2[which.min(tune_RF2[,2]),1]
fit.rf_direct = randomForest(Direction ~ ., SPY.trend_train,
                             ntree = 300, mtry = m, importance = T)
pred_trend = predict(fit.rf_direct, SPY.trend_test)
```

```
confusionMatrix(pred_trend, SPY.trend_test$Direction, positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down  Up
##       Down   37  28
##       Up    105  81
##
##                Accuracy : 0.4701
##                  95% CI : (0.4071, 0.5339)
##     No Information Rate : 0.5657
##     P-Value [Acc > NIR] : 0.999
##
##                   Kappa : 0.0034
##
```

```
##  Mcnemar's Test P-Value : 4.397e-11
##
##             Sensitivity : 0.7431
##             Specificity : 0.2606
##          Pos Pred Value : 0.4355
##          Neg Pred Value : 0.5692
##              Prevalence : 0.4343
##          Detection Rate : 0.3227
##    Detection Prevalence : 0.7410
##       Balanced Accuracy : 0.5018
##
##        'Positive' Class : Up
##
```

```
pred_trend_prob = predict(fit.rf_direct, SPY.trend_test, type = "prob")[,1]
roc.curve(SPY.trend_test$Direction, pred_trend_prob, plotit = T)
```




```
## Area under the curve (AUC): 0.520
```

觀察 confusion matrix 和 ROC curve 以及計算出的 ACC, AUC，模型預測表現並不好

上述各種 regression 和 classification models 表現都不是很理想，由此可知想要僅僅藉由股市過去的走勢來做未來預測精準度是很低的，應該還要考慮該領域的專業知識作為輔助。