碩管一甲
M09218001 周彥廷、M09218010 葉庭佑、M09218014 蔡芳旻
數據分析與應用-作業 2：caret 分類器實作

請見網路大學中的”FattyLiver.csv”檔案，表中每列代表一位受檢者，”是否有脂肪肝”欄位代表該受檢者是否有患有脂肪肝，其他欄位則為體檢項目，試對該分類問題進行建模，建模過程至少須包含幾個元素：

(1)敘述欄位

(2)敘述建模流程圖

(3)將數據切成訓練與測試樣本，比例為 7:3、進行 5-fold CV

(4)盡可能地找出你的最佳分類器，並寫下該分類器的數學原理

(5)紀錄實驗結果，包含最佳超參數、測試樣本混淆矩陣、測試樣本預測正確率、測試樣本 precision、recall、重要變數列表等

目次：

I.安裝與載入所需套件

1.安裝所需套件。

```
install.packages(c("dplyr","caret","mlbench","glmnet"))
#載入套件
library(dplyr)
library(caret)
library(mlbench)
```

II. 資料前處理

2.讀取 FattyLiver.csv。

```
data<-read.csv("C:/Users/MCUT/Desktop/FattyLiver.csv",header=TRUE,sep =
",",na.strings=" ")
data1<-tibble::as_tibble(data)
class(data1) #確認 data1 格式
[1] "tbl_df"     "tbl"        "data.frame"
```

3.進行資料清洗去 NA 值，轉成 Tidy Data。

```
Tdata<-data1[complete.cases(data1),]
```

	是否有脂肪肝	年齡	性別	BMI	收縮壓	舒張壓	脈搏	抽菸喝酒檳榔	腰圍	白血球	紅血球	血色素	血中紅血球百分比	紅血球平均容積
1	NO	2	0	23.07	98.35	67.10	88.06	0	73.26	6.04	4.57	8.88	30.01	66.20
2	NO	2	0	20.72	103.23	72.77	94.56	0	67.94	4.69	5.53	12.06	39.88	71.77
3	NO	2	0	20.24	115.18	59.49	71.69	0	69.84	6.56	4.46	13.21	39.74	91.79
4	NO	3	0	21.25	98.20	62.03	68.74	0	72.33	3.82	4.16	12.64	38.59	92.36
5	YES	3	1	25.16	109.37	64.61	89.36	2	85.59	8.40	4.91	15.20	44.92	91.83

```
Tdata                    1877 obs. of 42 variables
```

4.使用 createDataPartition 將數據切成訓練與測試樣本，比例為 7:3。

```
set.seed(7) #設定隨機種子
trainIndex <- createDataPartition(Tdata$是否有脂肪肝, p = 0.7,
                                  list = FALSE,times=1)
```

FattyLiver_Train <- Tdata[trainIndex,] #0.7

	是否有脂肪肝	年齡	性別	BMI	收縮壓	舒張壓	脈搏	抽菸喝酒檳榔	腰圍	白血球	紅血球	血色素	血中紅血球百分比	紅血球平均容積
1	NO	2	0	23.07	98.35	67.10	88.06	0	73.26	6.04	4.57	8.88	30.01	66.2
2	NO	2	0	20.72	103.23	72.77	94.56	0	67.94	4.69	5.53	12.06	39.88	71.7
3	NO	2	0	20.24	115.18	59.49	71.69	0	69.84	6.56	4.46	13.21	39.74	91.7
4	NO	3	0	21.25	98.20	62.03	68.74	0	72.33	3.82	4.16	12.64	38.59	92.3
5	NO	2	0	20.48	93.38	61.89	74.11	0	62.34	6.44	4.36	13.34	40.39	92.9

```
FattyLiver_Train          1315 obs. of 42 variables
```

FattyLiver_Test <- Tdata[-trainIndex,] #0.3

	是否有脂肪肝	年齡	性別	BMI	收縮壓	舒張壓	脈搏	抽菸喝酒檳榔	腰圍	白血球	紅血球	血色素	血中紅血球百分比	紅血球平均容積
1	YES	3	1	25.16	109.37	64.61	89.36	2	85.59	8.40	4.91	15.20	44.92	91.83
2	NO	2	0	19.46	100.21	64.19	77.18	0	63.35	7.25	4.70	12.44	39.45	85.10
3	YES	2	0	29.73	102.03	60.18	61.60	0	84.39	8.53	4.59	13.36	40.03	89.71
4	YES	2	1	28.18	122.52	87.78	81.23	0	96.73	7.84	5.42	16.33	47.30	88.62
5	YES	2	1	28.68	119.30	66.59	68.65	1	99.99	8.40	4.78	15.05	44.34	93.36

```
FattyLiver_Test           562 obs. of 42 variables
```

#5.使用 preProcess 對 data 進行資料特徵轉換(Feature Transforms)。

```
preProValues<-preProcess(FattyLiver_Train, method = c('center', 'scale'))
traindata<-predict(preProValues,FattyLiver_Train) #轉換訓練集 0.7
testdata<-predict(preProValues,FattyLiver_Test)     #轉換測試集 0.3
```

III.選模與建模

#6. 透過 spot checking 進行初步篩選訓練 model

```
#spot checking
control <- trainControl(method='cv', number=5) #5-fold CV
# train Logistic Regression
set.seed(7)
fit.glm <- caret:: train(是否有脂肪肝~., data=Tdata, method='glm',
metric='Accuracy', preProc=c('center', 'scale'), trControl=control)
```

```
# train Linear Discriminant Analysis
set.seed(7)
fit.lda <- caret::train(是否有脂肪肝~., data=Tdata, method='lda', metric='Accuracy',
preProc=c('center', 'scale'), trControl=control)

# train Regularized Regression
set.seed(7)
fit.glmnet <- caret:: train(是否有脂肪肝~., data=Tdata, method='glmnet',
metric='Accuracy', preProc=c('center', 'scale'), trControl=control)

# train k-Nearest Neighbors
set.seed(7)
fit.knn <- caret::train(是否有脂肪肝~., data=Tdata, method='knn', metric='Accuracy',
preProc=c('center', 'scale'), trControl=control)

# train Naive Bayes
set.seed(7)
fit.nb <- caret::train(是否有脂肪肝~., data=Tdata, method='nb', metric='Accuracy',
trControl=control)

# train Support Vector Machine
set.seed(7)
fit.svmRadial <- caret::train(是否有脂肪肝~., data=Tdata, method='svmRadial',
metric='Accuracy', trControl=control)

# train Classification and Regression Trees
set.seed(7)
fit.rpart <- caret::train(是否有脂肪肝~., data=Tdata, method='rpart',
metric='Accuracy', trControl=control)

# collect resamples using resamples
results <- resamples(list(LR=fit.glm, LDA=fit.lda, RR=fit.glmnet, KNN=fit.knn,
NB=fit.nb, SVM=fit.svmRadial, CART=fit.rpart))

summary(results) #顯示各 model 的 accuracy 和 kappa
```

```
Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LR   0.7845745 0.7925532 0.8026667 0.8002255 0.8080000 0.8133333    0
LDA  0.7845745 0.7845745 0.7973333 0.7986298 0.8133333 0.8133333    0
RR   0.7872340 0.8031915 0.8106667 0.8092851 0.8186667 0.8266667    0
KNN  0.7313830 0.7360000 0.7653333 0.7575986 0.7712766 0.7840000    0
NB   0.7446809 0.7520000 0.7706667 0.7698496 0.7872340 0.7946667    0
SVM  0.7686170 0.8005319 0.8026667 0.7986298 0.8080000 0.8133333    0
CART 0.7632979 0.7765957 0.7813333 0.7858454 0.7920000 0.8160000    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LR   0.5694656 0.5849185 0.6052070 0.6004374 0.6159426 0.6266534    0
LDA  0.5693195 0.5694170 0.5947675 0.5973898 0.6266746 0.6267702    0
RR   0.5748049 0.6063384 0.6212606 0.6186218 0.6373101 0.6533950    0
KNN  0.4634036 0.4720788 0.5307167 0.5154272 0.5425532 0.5683837    0
NB   0.4904574 0.5042855 0.5422285 0.5403191 0.5749491 0.5896747    0
SVM  0.5376788 0.6010864 0.6053081 0.5973616 0.6159754 0.6267596    0
CART 0.5266225 0.5532421 0.5627133 0.5716615 0.5839260 0.6318036    0
```

diffs <- diff(results) # model 間的差異性

summary(diffs) #查看所有 model p-values for pair-wise comparisons

```
Call:
summary.diff.resamples(object = diffs)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Accuracy
     LR       LDA        RR         KNN        NB         SVM        CART
LR            0.001596 -0.009060  0.042627  0.030376  0.001596  0.014380
LDA  1.00000           -0.010655  0.041031  0.028780  0.000000  0.012784
RR   0.15629 1.00000              0.051687  0.039435  0.010655  0.023440
KNN  0.19337 0.61563    0.08210             -0.012251 -0.041031 -0.028247
NB   0.82448 1.00000    0.39272   1.00000             -0.028780 -0.015996
SVM  1.00000 1.00000    0.71536   0.08386   0.41635             0.012784
CART 1.00000 1.00000    0.77440   0.81824   1.00000   1.00000

Kappa
     LR       LDA         RR          KNN         NB          SVM         CART
LR            3.048e-03 -1.818e-02  8.501e-02  6.012e-02  3.076e-03  2.878e-02
LDA  1.00000            -2.123e-02  8.196e-02  5.707e-02  2.811e-05  2.573e-02
RR   0.15607 1.00000                1.032e-01  7.830e-02  2.126e-02  4.696e-02
KNN  0.19591 0.61773     0.08263               -2.489e-02 -8.193e-02 -5.623e-02
NB   0.85040 1.00000     0.40135    1.00000                -5.704e-02 -3.134e-02
SVM  1.00000 1.00000     0.72049    0.08452     0.42757               2.570e-02
CART 1.00000 1.00000     0.76729    0.82940     1.00000    1.00000
```

#7.進行 5-fold CV (k-folds Cross Validation)。

```
control_fit <- trainControl(method='cv', number=5,
                            classProbs=TRUE,
                            summaryFunction=multiClassSummary,
                            selectionFunction = 'best') #取最佳指標
```

#Model 選用 glmnet 和 svmRadial 訓練資料並比較

#glmnet

```
set.seed(7)
glmnet_fit<- caret::train(是否有脂肪肝~., data=traindata, method='glmnet',
                                                           metric='Accuracy',
                                                           trControl= control_fit,
                                                           verbose = FALSE)
```

```
glmnet

1315 samples
  41 predictor
   2 classes: 'NO', 'YES'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1053, 1052, 1052, 1052, 1051
Resampling results across tuning parameters:

  alpha  lambda       logLoss    AUC        prAUC      Accuracy
  0.10   0.00056706   0.4411100  0.8796871  0.8660281  0.7954447
  0.10   0.00567060   0.4350572  0.8809045  0.8682762  0.8007506
  0.10   0.05670600   0.4349972  0.8855631  0.8750880  0.8015082
  0.55   0.00056706   0.4406236  0.8797909  0.8660048  0.7962023
  0.55   0.00567060   0.4308854  0.8827871  0.8707200  0.8037924
  0.55   0.05670600   0.4515610  0.8830437  0.8706329  0.7931864
  1.00   0.00056706   0.4402713  0.8794794  0.8656693  0.7969598
  1.00   0.00567060   0.4281818  0.8845154  0.8731253  0.8060796
  1.00   0.05670600   0.4727083  0.8748475  0.8639422  0.7962426

Kappa      F1         Sensitivity  Specificity  Pos_Pred_Value
0.5908847  0.7942369  0.7988139    0.7921394    0.7908620
0.6015873  0.8017221  0.8141750    0.7876054    0.7905110
0.6031515  0.8038161  0.8218438    0.7815676    0.7881182
0.5924102  0.7951734  0.8003523    0.7921394    0.7912018
0.6076444  0.8041704  0.8141515    0.7936432    0.7954791
0.5864049  0.7930570  0.8019025    0.7846206    0.7861486
0.5939647  0.7967616  0.8049677    0.7891319    0.7897597
0.6122732  0.8077074  0.8233823    0.7891319    0.7940627
0.5925879  0.7979418  0.8142102    0.7786170    0.7835848

  Neg_Pred_Value  Precision  Recall     Detection_Rate  Balanced_Accuracy
  0.8018412       0.7908620  0.7988139  0.3954511       0.7954766
  0.8128817       0.7905110  0.8141750  0.4030528       0.8008902
  0.8185541       0.7881182  0.8218438  0.4068551       0.8017057
  0.8030521       0.7912018  0.8003523  0.3962115       0.7962459
  0.8140708       0.7954791  0.8141515  0.4030441       0.8038973
  0.8032050       0.7861486  0.8019025  0.3969778       0.7932616
  0.8060184       0.7897597  0.8049677  0.3984929       0.7970498
  0.8213445       0.7940627  0.8233823  0.4076098       0.8062571
  0.8119467       0.7835848  0.8142102  0.4030614       0.7964136

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were alpha = 1 and lambda = 0.0056706.
```

svmRadial

set.seed(7)

svmRadial _fit<- caret::train(是否有脂肪肝~., data=traindata,

method=' svmRadial ', metric='Accuracy', trControl=control_fit, verbose = FALSE)

```
Support Vector Machines with Radial Basis Function Kernel

1315 samples
  41 predictor
   2 classes: 'NO', 'YES'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1053, 1052, 1052, 1052, 1051
Resampling results across tuning parameters:

  C     logLoss    AUC        prAUC      Accuracy   Kappa
  0.25  0.4421278  0.8764886  0.8668310  0.7908618  0.5817304
  0.50  0.4372039  0.8791689  0.8687610  0.7885891  0.5771841
  1.00  0.4388807  0.8782619  0.8681256  0.7885861  0.5771965

 F1         Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value
 0.7897126  0.7957134    0.7861130    0.7859204       0.7991072
 0.7876087  0.7941750    0.7831055    0.7827868       0.7970175
 0.7877317  0.7941985    0.7831169    0.7823657       0.7964455

      Precision  Recall     Detection_Rate  Balanced_Accuracy
      0.7859204  0.7957134  0.3939244       0.7909132
      0.7827868  0.7941750  0.3931669       0.7886402
      0.7823657  0.7941985  0.3931610       0.7886577
```

IV.驗證模型與評估績效

#8.使用 predict 函數對 testdata 測試資料集 NO/YES 進行預測。

pred_prob<-predict(glmnet_fit, # svmRadial _fit

newdata=head(testdata), type='prob')

svmRadial

```
> pred_prob
          NO        YES
1 0.32096361 0.67903639
2 0.92660683 0.07339317
3 0.31815732 0.68184268
4 0.05167244 0.94832756
5 0.03836845 0.96163155
6 0.76107368 0.23892632
```

glmnet

```
> pred_prob
          NO        YES
1 0.26174250 0.73825750
2 0.95805619 0.04194381
3 0.16379753 0.83620247
4 0.10988229 0.89011771
5 0.04892981 0.95107019
6 0.87546901 0.12453099
```

pred_res<-predict(glmnet_fit, # svmRadial _fit

newdata=testdata)

#9.計算測試樣本混淆矩陣、測試樣本預測正確率、測試樣本 precision、recall

與其他相關驗證指標

```
confusionMatrix1<-confusionMatrix(pred_res,
                                  factor(testdata$是否有脂肪肝),
                                  dnn = c('Prediction', 'Reference'),
                                  mode = 'everything')
```

glmnet

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  231  56
       YES  47 228

               Accuracy : 0.8167
                 95% CI : (0.7822, 0.8479)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6335

 Mcnemar's Test P-Value : 0.4305

            Sensitivity : 0.8309
            Specificity : 0.8028
         Pos Pred Value : 0.8049
         Neg Pred Value : 0.8291
              Precision : 0.8049
                 Recall : 0.8309
                     F1 : 0.8177
             Prevalence : 0.4947
         Detection Rate : 0.4110
   Detection Prevalence : 0.5107
      Balanced Accuracy : 0.8169

       'Positive' Class : NO
```

svmRadial

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  231  55
       YES  47 229

               Accuracy : 0.8185
                 95% CI : (0.7841, 0.8495)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6371

 Mcnemar's Test P-Value : 0.4882

            Sensitivity : 0.8309
            Specificity : 0.8063
         Pos Pred Value : 0.8077
         Neg Pred Value : 0.8297
              Precision : 0.8077
                 Recall : 0.8309
                     F1 : 0.8191
             Prevalence : 0.4947
         Detection Rate : 0.4110
   Detection Prevalence : 0.5089
      Balanced Accuracy : 0.8186

       'Positive' Class : NO
```

#10. importvariable 重要變數列表

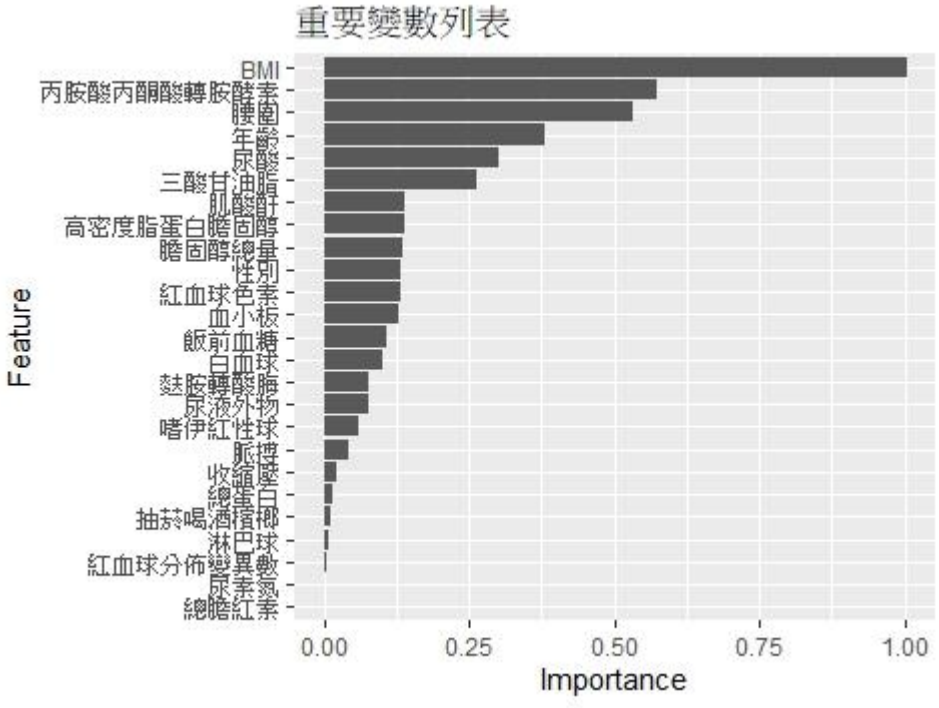
importance <- varImp(glmnet_fit, scale = FALSE)

```
glmnet variable importance

  only 20 most important variables shown (out of 41)

                                Overall
BMI                             1.00339
丙胺酸丙酮酸轉胺酵素            0.57466
腰圍                            0.53170
年齡                            0.38064
尿酸                            0.30028
三酸甘油脂                      0.26384
肌酸酐                          0.13965
高密度脂蛋白膽固醇              0.13807
膽固醇總量                      0.13642
性別                            0.13224
紅血球色素                      0.13163
血小板                          0.12758
飯前血糖                        0.10767
白血球                          0.10207
麩胺轉酸胺                      0.07887
尿液外物                        0.07603
嗜伊紅性球                      0.05842
脈搏                            0.04180
收縮壓                          0.02380
總蛋白                          0.01400
```

#繪製重要變數列表長條圖

#plot(importance)

ggplot(importance,scale = TRUE, top = 25) +ggtitle('重要變數列表')



V.嘗試調整其他參數與改變 model 來提高正確率並與 SVM 正確率比較

Boosted Logistic Regression (LogitBoost)在測試樣本預測正確率可以達到 82.55%，因此使用 LogitBoost 模型訓練。

```
fitControl <- trainControl(method = 'repeatedcv', #cv 改成 repeatedcv
                           number = 5,
                           repeats = 5,  #設定重複 5 次
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary,
                           search = 'random') #設置 tuneLength
set.seed(7)
LogitBoost_fit <- caret::train(是否有脂肪肝~., data = traindata,
                               method = ' LogitBoost ', # svmRadial
                               metric = 'ROC',
                               tuneLength = 10, #設置 10
                               trControl = fitControl)
```

```
Boosted Logistic Regression

1315 samples
  41 predictor
   2 classes: 'NO', 'YES'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 1053, 1052, 1052, 1052, 1051, 1052, ...
Resampling results across tuning parameters:

  nIter  ROC        Sens       Spec
   8     0.8293531  0.8300095  0.8150304
  15     0.8241995  0.7736136  0.7464183
  22     0.8243493  0.7991382  0.8171702
  31     0.8238959  0.7640540  0.7478788
  40     0.8226578  0.7926689  0.7994493
  66     0.8240079  0.7947418  0.8030391
  67     0.8229608  0.7536301  0.7656756
  88     0.8183253  0.7835834  0.7840512
  90     0.8182364  0.7808116  0.7963030
  92     0.8199756  0.7829920  0.7918120

ROC was used to select the optimal model using the largest value.
The final value used for the model was nIter = 8.
```

```
pred_prob<-predict(LogitBoost_fit, newdata=head(testdata), type='prob')
```

```
           NO        YES
1 0.11920292 0.88079708
2 0.98201379 0.01798621
3 0.11920292 0.88079708
4 0.11920292 0.88079708
5 0.01798621 0.98201379
6 0.88079708 0.11920292
```

```
pred_res<-predict(LogitBoost_fit, newdata=testdata,)
confusionMatrix1<-confusionMatrix(pred_res,
                                  factor(testdata$是否有脂肪肝),
                                  dnn = c('Prediction', 'Reference'),
                                  mode = 'everything')
```

LogitBoost 與 SVM 混淆矩陣比較結果如下：

LogitBoost

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  168  38
       YES  36 182

               Accuracy : 0.8255
                 95% CI : (0.7859, 0.8604)
    No Information Rate : 0.5189
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6506

 Mcnemar's Test P-Value : 0.9075

            Sensitivity : 0.8235
            Specificity : 0.8273
         Pos Pred Value : 0.8155
         Neg Pred Value : 0.8349
              Precision : 0.8155
                 Recall : 0.8235
                     F1 : 0.8195
             Prevalence : 0.4811
         Detection Rate : 0.3962
   Detection Prevalence : 0.4858
      Balanced Accuracy : 0.8254

       'Positive' Class : NO
```

svmRadial(repeatedcv=5)

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  230  54
       YES  48 230

               Accuracy : 0.8185
                 95% CI : (0.7841, 0.8495)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6371

 Mcnemar's Test P-Value : 0.6205

            Sensitivity : 0.8273
            Specificity : 0.8099
         Pos Pred Value : 0.8099
         Neg Pred Value : 0.8273
              Precision : 0.8099
                 Recall : 0.8273
                     F1 : 0.8185
             Prevalence : 0.4947
         Detection Rate : 0.4093
   Detection Prevalence : 0.5053
      Balanced Accuracy : 0.8186

       'Positive' Class : NO
```

結語：經過 repeatcv 的方式，超參數設定一致，發現 LogitBoost 在測試樣本預測正確率上相較於 svmRadial 有所提升。

VI.最佳分類器數學原理

Spot checking 得出的最佳分類器：SVM

SVM(support vector machine)簡介與樹數學原理：是一個二元分類器，可在低維度空間將線性不可分的樣本反射到高維度空間中，目的找出一個超平面(Hyperplane）將樣本做有效的切割，且超平面兩邊樣本要盡可能地遠離這個超平面。

問題概述：

簡單平面問題且在同一平面上即可用一條線來分類如圖 1 所示。

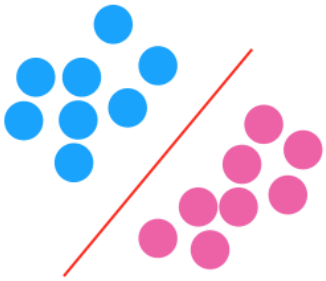


圖 1

如果藍紅球無法在平面上看出如何有效分類，透過特徵空間找出一個能夠有效分割藍紅球之超平面，如圖 2 所示。

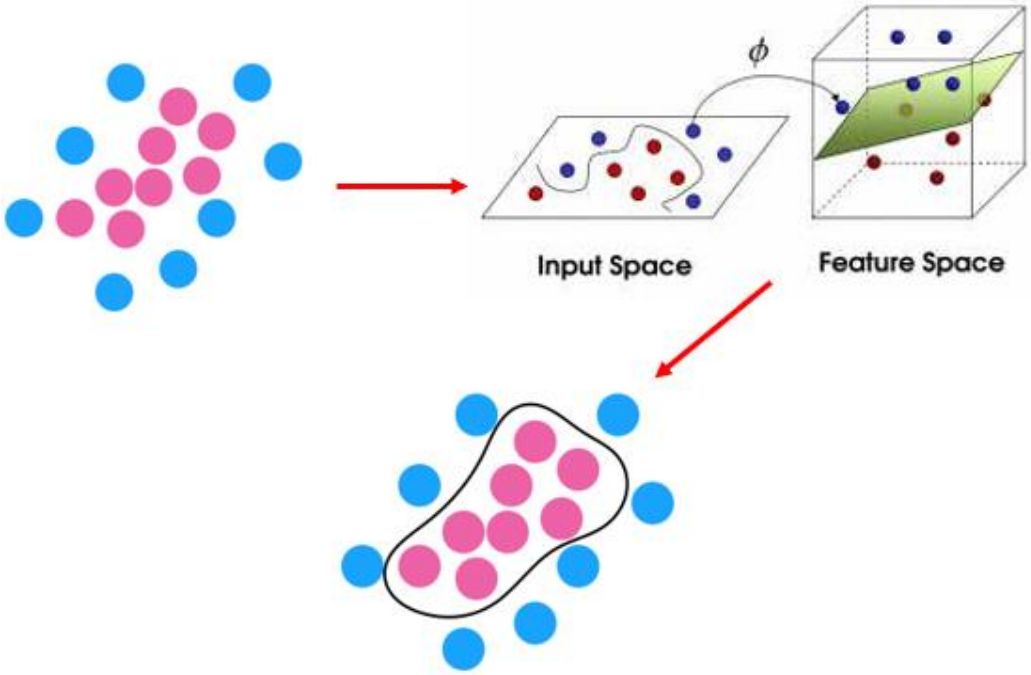


圖 2

Margin 最佳化：

紅線到黑線的距離稱為 Margin，SVM 就是透過去找 Margin 最大的那個紅線，來找最好的線，如何使 Margin 最大呢?假設紅線是 w*x = 0 在紅線上方的區域就是 w*x >0 紅線下方的區域就是 w*x <0，同理類推來看在左邊虛線上方的區域是 w*x<-k 在右邊虛線下方的區域是 w*x>k，虛線中間不會有資料點，如圖 3 所示。

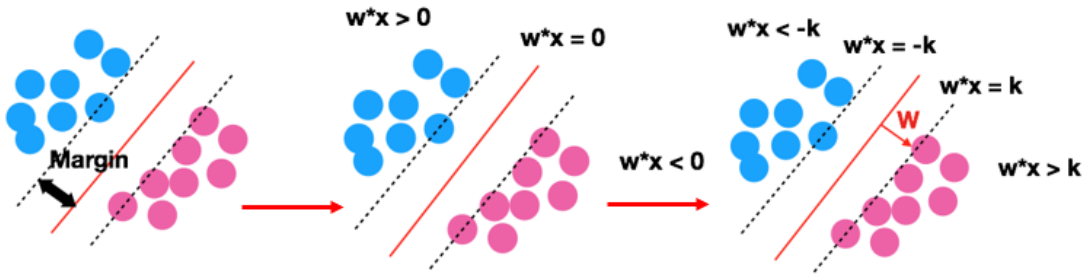


圖 3

數學原理：

虛線上的點 X1,X2 其實就是所謂的支援向量(Support vector)，我們主要是利用支援向量來算出 Margin，並最大化 Margin。那要怎麼計算 Margin 呢？ 利用 X1 向量-X2 向量得到的向量投影到 W 就可以了！接下來就是在 Y*(W*X) ≥k 的條件下(虛線中間沒有點)來最大化 Margin ，如圖 4 所示。

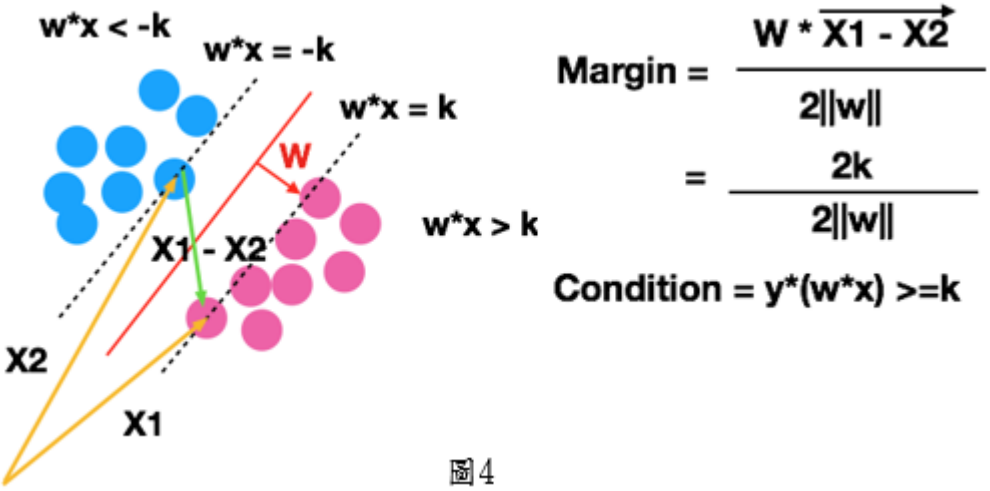


圖 4

SVM 資料參考來源：

https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC3-4%E8%AC%9B-%E6%94%AF%E6%8F%B4%E5%90%91%E9%87%8F%E6%A9%9F-support-vector-machine-%E4%BB%8B%E7%B4%B9-9c6c6925856b

Boosted Logistic Regression (LogitBoost)簡介與數學原理

一、Boosting 算法概念：將很多個弱的分類器(weak classifier)進行合成變成一個強分類器(Strong classifier)，和 Bagging 不同的是分類器之間是有關聯性的，透過舊分類器將錯誤資料權重提高，然後再訓練新的分類器，新分類器就會學習到錯誤分類資料(misclassified data)特性，進而提升分類結果。

二、Boost 算法簡介：1. Discrete Adaboost 2. Real Adaboost 3. Gentle AdaBoost 4. LogitBoost，上述 4 個 boost 算法，大體結構都是相似，關於損失函數，通常較多採用均方誤差和似然函數，算法中，Discrete AdaBoost、Real AdaBoost 與 Gentle AdaBoost 算法都是採用對數損失函數，而 Logit Boost 算法則採用最大化對數似然函數來推導。

三、Logit Boost 數學原理：

LogitBoost (*J classes*)

1. Start with weights $w_{ij} = 1/N$, $i = 1, \ldots, N$, $j = 1, \ldots, J$, $F_j(x) = 0$ and $p_j(x) = 1/J \; \forall \, j$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Repeat for $j = 1, \ldots, J$:
      (i) Compute working responses and weights in the jth class,

$$z_{ij} = \frac{y^*_{ij} - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))},$$

$$w_{ij} = p_j(x_i)(1 - p_j(x_i)).$$

      (ii) Fit the function $f_{mj}(x)$ by a weighted least-squares regression of z_{ij} to x_i with weights w_{ij}.

   (b) Set $f_{mj}(x) \leftarrow \frac{J-1}{J}\left(f_{mj}(x) - \frac{1}{J}\sum_{k=1}^{J} f_{mk}(x)\right)$, and $F_j(x) \leftarrow F_j(x) + f_{mj}(x)$.
   (c) Update $p_j(x)$ via (40).
3. Output the classifier $\arg\max_j F_j(x)$.

四、具體優化方法：Discrete AdaBoost 與 Real AdaBoost 主要通過類似梯度下降方法來優化，而 Gentle AdaBoost 與 Logit Boost 都是採用類似牛頓迭代的方式優化。

參考資料來源

https://www.twblogs.net/a/5bbceb182b71776bd30bb5dd

(Friedman et al., 2000)

https://web.stanford.edu/~hastie/Papers/AdditiveLogisticRegression/alr.pdf

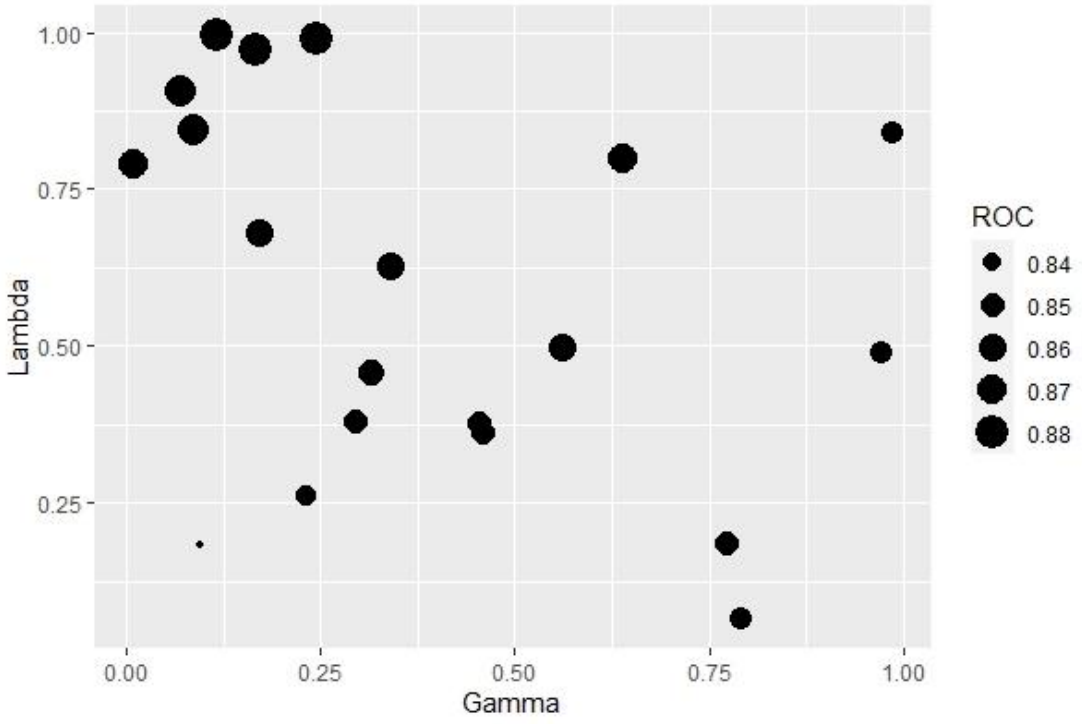
VII.正確率整理總表

Model	Accuracy	Precision	Recall
LogitBoost	0.8255	0.8155	0.8235
rda	0.8221	0.8007	0.8525
svmRadial	0.8185	0.8077	0.8309
glmnet	0.8167	0.8049	0.8309
gbm	0.8167	0.8028	0.8345
ada	0.8132	0.8057	0.8201

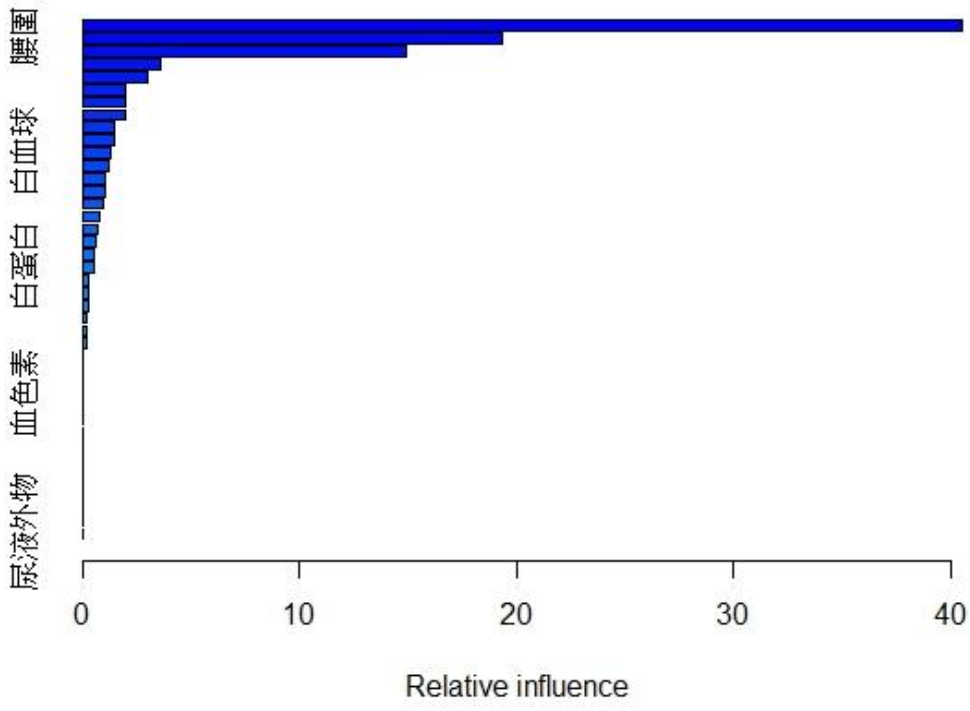
VIII.附錄：使用其他 model 預測樣本正確率

程式碼：

```
fitControl_rda <- trainControl(method = 'repeatedcv',
                               number = 5,
                               repeats = 5,
                               classProbs = TRUE,
                               summaryFunction = twoClassSummary,
                               search = 'random')
set.seed(7)
rda_fit <- caret::train(是否有脂肪肝~., data = traindata,
                        method = 'rda',
                        metric = 'ROC',
                        tuneLength = 20,
                        trControl = fitControl_rda)
ggplot(rda_fit)
```



```
gbmGrid <- expand.grid(interaction.depth = c(1, 5, 9),
                       n.trees = (1:30)*50,
                       shrinkage = 0.1,
                       n.minobsinnode = 20)
fitControl_gbm <- trainControl(method = "repeatedcv",
                               number = 5,
                               repeats = 10,
                               classProbs = TRUE,
                               summaryFunction = twoClassSummary,
                               search = "grid")
set.seed(7)
gbm_Fit2<- caret::train(是否有脂肪肝~., data = traindata, method = 'gbm',
                        trControl = fitControl_gbm,
                        verbose = FALSE,
                        tuneGrid = gbmGrid,
                        metric = 'ROC')
tibble::as_tibble(summary(gbmFit_2))
```



rda

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  237  59
       YES  41 225

               Accuracy : 0.8221
                 95% CI : (0.7879, 0.8528)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.6443

 Mcnemar's Test P-Value : 0.08913

            Sensitivity : 0.8525
            Specificity : 0.7923
         Pos Pred Value : 0.8007
         Neg Pred Value : 0.8459
              Precision : 0.8007
                 Recall : 0.8525
                     F1 : 0.8258
             Prevalence : 0.4947
         Detection Rate : 0.4217
   Detection Prevalence : 0.5267
      Balanced Accuracy : 0.8224

       'Positive' Class : NO
```

gbm

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  232  57
       YES  46 227

               Accuracy : 0.8167
                 95% CI : (0.7822, 0.8479)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6336

 Mcnemar's Test P-Value : 0.3245

            Sensitivity : 0.8345
            Specificity : 0.7993
         Pos Pred Value : 0.8028
         Neg Pred Value : 0.8315
              Precision : 0.8028
                 Recall : 0.8345
                     F1 : 0.8183
             Prevalence : 0.4947
         Detection Rate : 0.4128
   Detection Prevalence : 0.5142
      Balanced Accuracy : 0.8169

       'Positive' Class : NO
```

```
control_test <- trainControl(method='cv', number=5, classProbs=TRUE,
                             summaryFunction=multiClassSummary,
                             selectionFunction = 'best')
set.seed(7)
tree_fit <- caret::train(是否有脂肪肝~., data = traindata, method = 'ada',
                         metric = "ROC", trControl = control_test)
```

ada

```
Confusion Matrix and Statistics

          Reference
Prediction  NO YES
       NO  228  55
       YES  50 229

               Accuracy : 0.8132
                 95% CI : (0.7784, 0.8446)
    No Information Rate : 0.5053
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6264

 Mcnemar's Test P-Value : 0.6963

            Sensitivity : 0.8201
            Specificity : 0.8063
         Pos Pred Value : 0.8057
         Neg Pred Value : 0.8208
              Precision : 0.8057
                 Recall : 0.8201
                     F1 : 0.8128
             Prevalence : 0.4947
         Detection Rate : 0.4057
   Detection Prevalence : 0.5036
      Balanced Accuracy : 0.8132

       'Positive' Class : NO
```