目錄

環境設定

運行環境版本 ：Matlab R2017a
運作資料夾 ： method9

1. Matlab 安裝完成後, 在"我的文件"或是"文件"內會有一個資料夾 Matlab 請用壓縮檔 Matlab 中的 Matlab 資料夾取代它. 或是將內容物放置於其中

2. 所有的程式均放在 method9 內, 開啟 MATALB, 在 Maltab command 視窗操作進入資料夾 method9 之中, 即可開始執行程式

3. 建議電腦的記憶體在 8G 以上，因為訓練 SVM 分類器會耗費大量的記憶體，因此若是使用的記憶體小於 8G，則當使用較大的資料集時，訓練 SVM 分類器可能會錯誤

壓縮檔 Matlab 資料夾結構如下

Matlab
- 資料夾"資料夾紀錄" → 放置論文使用所有的資料集的特徵描述
- 資料夾"method9" → 放置論文使用的所有程式
  - 資料夾"data" → 放置論文使用的資料集(轉成 MATLAB DATA 格式)
  - 資料夾"資料集" → 放置論文使用的資料集(未轉成 MATLAB DATA 格式)
- 資料夾" 論文數據" → 放置論文的數據
  1. 文件檔:為各個 over-sampling 方法在 5 種分類器環境下的七種效能指標數字(使用的訓練集為文件檔名稱)
  2. Excel 檔-0622 比較表:
     甲、Wilcoxon signed rank 比較表
     乙、20 個資料集其分別跑三十次的平均效能表現
     丙、20 個資料集其跑三十次的 t-test 檢定結果
  3. Excel 檔-資料描述:對論文使用的資料集所做的前置資料處理動作

前言

Matlab 與 C 這種編譯式語言不同, 編譯式語言一次將所有原程式碼翻譯成另一種語言，而 Matlab 是一種直譯式語言, 不會一次把整個程式轉譯出來，而是每轉譯一行程式敘述就立刻執行，然後再轉譯下一行，再執行，如此不停地進行下去，此作法消除了編譯整個程式的負擔，程式可以拆分成多個部分來模組化，但這會讓執行時的效率打了折扣。
（不用像 C 那樣還要先編譯成.o 檔 電腦才能執行）

我自己為了執行上的方便，將程式放在同一層資料夾，不然指定路徑會是一個很討厭的問題。(若是你的程式呼叫兩個檔案名稱相同的檔案，Matlab 預設同一層資料夾的檔案的優先權最高，因此會優先呼叫同一層資料夾內的檔案)

預設的實驗流程是

1. over-sampling 方法替訓練集產生人造實例後得到一個新的訓練集，此新的訓練集會各自使用 KNN/DT/SVM/NB/Logitc 等五種分類演算法訓練一個分類器(共五個)，此五個分類器會以 Accuracy/TP/FP/Precision/AUC/G-mean/F-measure 七種效能指標衡量分類器的表現。
2. 上述動作每一個 over-sampling 方法都會執行一次
3. 預設要一起比較的 over-sampling 方法為
   {ISMOTE,B-SMOTE,ADASYN,MSMOTE,MWMOTE,SMOTE}共六種
   ISMOTE 方法在使用 Script_Experience 函數時可替換成別的方法。

當執行上述實驗流程後，提供兩種統計上的檢定可以作為驗證其實驗結果是否顯著有效。

Wilcoxon(使用 Performance_Wilcoxon) 和 T-test(使用 Performance_t_test)。

Script 程式

Script_Experience

輸入變數:

無

輸出內容:

使用者希望輸出的檢定的內容(文字檔)

函數功能:

1. **使用者可透過此函數設計出自己的實驗流程**，例如使用者自己設計的 over-sampling 方法稱為 QSMOTE2_1(亦作為檔案名稱)，若使用者想知道 QSMOTE2_1 與預設的五個方法在七種效能指標下用 Wilcoxon 檢定的結果，則可在 script_Experience 內使用 Performance_Wilcoxon 程式，若使用者想知道 T-test 檢定的結果，則在 Script_Experience 內使用 Performance_t_test 程式。使用者自己設計的方法可能不只 QSMOTE2_1 一個，也許另有一個方法稱為 QSMOTE2_2，則使用者並不需要改寫 Performance_Wilcoxon 以及 Performance_t_test 程式內呼叫的方法名稱(e.g.由 QSMOTE2_1 改寫成 QSMOTE2_2)。僅需要在 Script_Experience 程式內指定(輸入)此兩個方法名稱，並且分別選擇動作(哪種檢定)，在程式運作完成後即可得到使用者想知道的此兩個方法在檢定下的結果。若使用者自己設計的 over-sampling 方法多過於兩個方法則以此類推。其輸出結果依照指定的檢定不同會有不同的輸出結果。
2. **Script_Experience 的設計概念為實驗的腳本**，用來幫助使用者解決實驗排程的問題，使用者可**一次安排複數的實驗檢定**。Matlab 與 C 不同，一次只能執行一個程式，因此一個用來運行實驗程式的腳本程式有其方便性。當然若是使用者的實驗數目稀少或者使用者的時間很多(想要自己人工操作)則並非一定要透過 Script_Experience 程式才可操作 Performance_Wilcoxon 以及 Performance_t_test 程式，**亦可直接在 Matlab 視窗下呼叫 Performance_Wilcoxon 或者 Performance_t_test 程式並在執行完成後取得該程式執行的結果。**
3. 若是遇到錯誤則 err 訊息將會被印出在 command line 並等 60 秒後再度執行程式。Matlab 在學校有人數上的使用限制，若是碰到此限制，**可交由電腦每 60 秒測試一次，直到通過人數限制，即可自**

動開始執行程式，並不需等在電腦前以人工的方式測試現在是否可以使用 (請依照 err 訊息,自行判斷是學校使用限制所所造成的錯誤還是自己程式的錯誤.按下 Ctrl+C 可取消程式運作.

操作說明:

1. 在下圖”放入實驗的內容”任一區域之內，
   將你設計的over-sampling方法檔案名稱存(輸入)在變數yourfunction內 (over-sampling方法檔案名稱左側需要**加上@符號**)
2. 在上述變數 yourfunction 下方一行呼叫(填上)欲執行的檢定函數
   - 若欲執行 T-test 則呼叫 Performance_t_test 函數
   - 若欲執行 Wilcoxon 檢定則呼叫 Performance_Wilcoxon 函數
3. 2個檢定函數有共同的輸入參數yourfunction, yourFunctionName以及OutTextFileName。
   甲、變數Yourfunction表示使用者設計的over-sampling方法檔案名稱，必須要指定檔案名稱，檢定函數才有辦法找出此檔案，並執行此使用者設計的over-sampling方法(若變數Yourfunction指定的檔案在運作資料夾中找不到，則程式會出錯)
   乙、yourFunctionName表示使用者設計的over-sampling方法在檢定函數最終輸出的結果文字檔中，會以甚麼名字呈現(若不了解此文字描述，請看下面兩圖片的圖示)
   丙、OutTextFileName表示檢定函數最終輸出的結果文字檔的名稱(若不了解此文字描述，請看下面兩圖片的圖示)

4. 完成上述設定(呼叫欲執行的檢定函數以及該檢定函數的輸入參數)後按下 F5 執行 Script_Experience 腳本程式。
   Script_Experience 程式將依序由上而下呼叫(執行)使用者在”放入實驗的內容”區域之內設定的檢定程式

```
again=true;
while(again)
    try
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  放入實驗的內容  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%         example

        yourfunction=@QSMOTE2_1;
        Performance_Wilcoxon( yourfunction,"yourFunctionName","outTextFileName")

%         yourfunction=@QSMOTE2_2;
%         Performance_Wilcoxon( yourfunction,"yourFunctionName2","outTextFileName2")

%         yourfunction=@QSMOTE2_1;
%         Performance_t_test( yourfunction,"yourFunctionName","outTextFileName")

%         yourfunction=@QSMOTE2_2;
%         Performance_t_test( yourfunction,"yourFunctionName2","outTextFileName2")



        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        again=false;
    catch err
        disp(['errorMessage : ' err.message]);
        disp('sleep 60 sec, if you dont want to wait , please click Ctrl+c  ');
        disp(" ");
        pause(60);
        again=true;
    end
end
```

圖示說明： 在矩形選取部分，設定使用者設計的 over-samping 方法檔案名稱為 QSMOTE2_1 並使用 Performance_Wilcoxon 函數做 Wilcoxon 檢定 QSMOTE2_1 這個 over-samping 方法與預設的五種 over-sampling 方法在七項效能指標的結果是否具有顯著差異，其檢定的實驗結果將會被輸出存入在文字檔內，此處的文字檔名稱被設定為” OutTextFileName”，在”OutTextFileName” 文字檔內，使用者設計的 over-sampling 方法其顯示的名稱被設定為為”yourFunctionName”。

1. 此處的 QSMOTE2_1 可代換成使用者自己設計的 over-sampling 方法
2. 此處的字串” yourFunctionName” ，可代換成使用者喜歡的名稱作為文字檔案內使用者設計的 over-samping 方法的名稱
3. 此處的字串” OutTextFileName” ，可代換成使用者喜歡的名稱作為文字檔的檔案名稱

(矩形選取部分下方的註解部分 ，依序是執行

1. QSMOTE2_2 這個 over-sampling 方法並得到其效能指標，並使

用 Wilcoxon 檢定檢查此效能結果是否顯著，其檢定結果將會被輸出在文字檔內

2. QSMOTE2_1 這個 over-sampling 方法並得到其效能指標，並使用 T 檢定檢查此效能結果是否顯著，其檢定結果將會被輸出在文字檔內
3. QSMOTE2_2 這個 over-sampling 方法並得到其效能指標，並使用 T 檢定檢查此效能結果是否顯著，其檢定結果將會被輸出在文字檔內

1. 上圖 yourfunction 處 輸入 你指定要呼叫的方法檔案名稱（上圖為呼叫 QSMOTE2_1）
2. 上圖 yourFunctionName 處 輸入 輸出檔案時,該檔案內容中所顯示的你的方法的名稱(e.g.下圖是在 yourFunctionName 處輸入 QSMOTE 的結果)
3. 上圖 OutTextFileName 處 輸入 輸出檔案時,你的檔案名稱為何

   (e.g.下圖文字檔名稱為在 OutTextFileName 處輸入的內容)

   注意：下圖文字的 average 是效能指標 accuracy 的意思,但程式打錯.

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	AUC	AUC	AUC	AUC	AUC
Btissue.mat	QSMOTE	0.592857	0.750000	0.578571	0.728571	0.728571
Btissue.mat	B-SMOTE	0.607143	0.685714	0.650000	0.735714	0.792857
Btissue.mat	ADAYSN	0.621429	0.707143	0.585714	0.728571	0.792857
Btissue.mat	MSMOTE	0.614286	0.700000	0.664286	0.707143	0.757143
Btissue.mat	MWMOTE	0.528571	0.714286	0.657143	0.685714	0.742857
Btissue.mat	SMOTE	0.614286	0.714286	0.614286	0.721429	0.707143

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	average	average	average	average	average
Btissue.mat	QSMOTE	0.542857	0.752381	0.561905	0.638095	0.714286
Btissue.mat	B-SMOTE	0.552381	0.695238	0.628571	0.657143	0.780952
Btissue.mat	ADAYSN	0.580952	0.752381	0.580952	0.647619	0.790476
Btissue.mat	MSMOTE	0.600000	0.685714	0.685714	0.657143	0.771429
Btissue.mat	MWMOTE	0.457143	0.723810	0.590476	0.619048	0.723810
Btissue.mat	SMOTE	0.600000	0.752381	0.609524	0.657143	0.714286

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	TP	TP	TP	TP	TP
Btissue.mat	QSMOTE	0.742857	0.742857	0.628571	1.000000	0.771429
Btissue.mat	B-SMOTE	0.771429	0.657143	0.714286	0.971429	0.828571
Btissue.mat	ADAYSN	0.742857	0.571429	0.600000	0.971429	0.800000
Btissue.mat	MSMOTE	0.657143	0.742857	0.600000	0.857143	0.714286
Btissue.mat	MWMOTE	0.742857	0.685714	0.857143	0.885714	0.800000
Btissue.mat	SMOTE	0.657143	0.600000	0.628571	0.914286	0.685714

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	FP	FP	FP	FP	FP
Btissue.mat	QSMOTE	0.557143	0.242857	0.471429	0.542857	0.314286
Btissue.mat	B-SMOTE	0.557143	0.285714	0.414286	0.500000	0.242857
Btissue.mat	ADAYSN	0.500000	0.157143	0.428571	0.514286	0.214286
Btissue.mat	MSMOTE	0.428571	0.342857	0.271429	0.442857	0.200000
Btissue.mat	MWMOTE	0.685714	0.257143	0.542857	0.514286	0.314286
Btissue.mat	SMOTE	0.428571	0.171429	0.400000	0.471429	0.271429

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	Precision	Precision	Precision	Precision	Precision
Btissue.mat	QSMOTE	0.415232	0.687500	0.351587	0.577778	0.581978
Btissue.mat	B-SMOTE	0.431242	0.570000	0.549206	0.626797	0.668803
Btissue.mat	ADAYSN	0.451956	0.803333	0.391111	0.601797	0.701632
Btissue.mat	MSMOTE	0.415556	0.554167	0.469524	0.635354	0.711156
Btissue.mat	MWMOTE	0.377045	0.659567	0.535556	0.577353	0.674359
Btissue.mat	SMOTE	0.425000	0.787273	0.439875	0.630159	0.549524

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	G_mean	G_mean	G_mean	G_mean	G_mean
Btissue.mat	QSMOTE	0.515180	0.721451	0.349725	0.570328	0.706786
Btissue.mat	B-SMOTE	0.484835	0.633113	0.545391	0.584651	0.780164
Btissue.mat	ADAYSN	0.542062	0.630543	0.350225	0.577375	0.779905
Btissue.mat	MSMOTE	0.576320	0.637365	0.531860	0.574552	0.723584
Btissue.mat	MWMOTE	0.398076	0.675382	0.515127	0.554779	0.703048
Btissue.mat	SMOTE	0.582518	0.638636	0.524782	0.582831	0.681603

		KNN	DT	SVM	NaiveB	LogitR
DataSet	Method	F_measure	F_measure	F_measure	F_measure	F_measure
Btissue.mat	QSMOTE	0.517037	0.671900	0.429143	0.699000	0.642071
Btissue.mat	B-SMOTE	0.537255	0.560314	0.562890	0.713282	0.720061
Btissue.mat	ADAYSN	0.539418	0.559495	0.432513	0.699949	0.722735
Btissue.mat	MSMOTE	0.497145	0.580030	0.483463	0.674393	0.650379
Btissue.mat	MWMOTE	0.488512	0.606638	0.612758	0.660676	0.685333

Performance_t_test

輸入參數:

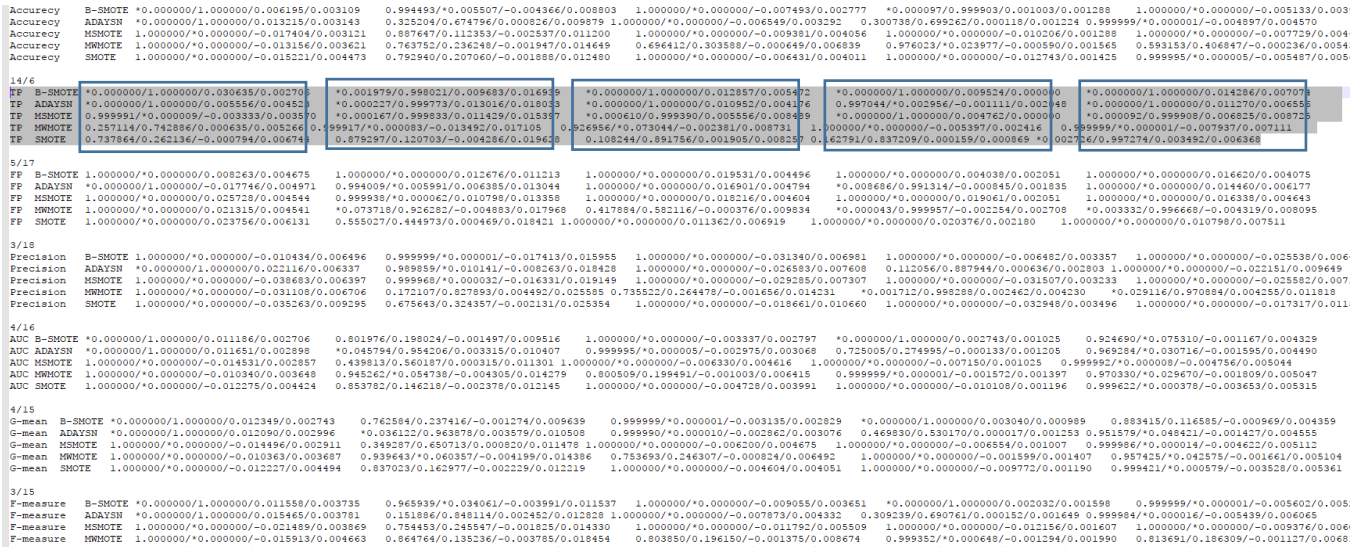
1. yourFunction : 你欲呼叫的方法檔案名稱－作為日後程式自動幫你呼叫方法的呼叫依據
2. yourFunctionName : 你的方法名稱 －在輸出檔案中的你的方法的名字
3. outTextFileName : 產生的文字檔名稱 － 用來避免產生相同文字檔名稱(舊的會被覆蓋)

輸出內容:

1. 變數yourFunction指定的over-sampling方法以及預設的五種over-sampling方法在五種分類器環境下的七個效能指標(實驗使用的每一個資料集會輸出一個文字檔)
2. 變數yourFunction指定的over-sampling方法對預設的五種over-sampling方法在五種分類器環境下的七個效能指標使用t檢定的檢定結果(實驗使用的每一個資料集會輸出一個文字檔) e.g.以下圖所選取部分對於TP-rate效能指標的描述依序為**KNN/決策樹/SVM/NB/Logistic演算法**的檢定結果(框框)

   每一個框框內(某一個分類器環境),由左而右依序為

   1. 變數yourFunction指定的方法**勝過**該列over-sampling方法的P-Value
   2. 變數yourFunction指定的方法**輸給**該列over-sampling方法的P-Value
   3. 變數yourFunction指定的方法與該列over-sampling方法的效能指標的差的平均值
   4. 變數yourFunction指定的方法與該列over-sampling方法的效能指標的差的標準差



```
Accurecy    B-SMOTE *0.000000/1.000000/0.006195/0.003109    0.994493/*0.005507/-0.004366/0.008803    1.000000/*0.000000/-0.007493/0.002777    *0.000097/0.999903/0.001003/0.001288    1.000000/*0.000000/-0.005133/0.003
Accurecy    ADAYSN  *0.000000/1.000000/0.013215/0.003143    0.325204/0.674796/0.000826/0.009879 1.000000/*0.000000/-0.006549/0.003292    0.300738/0.699262/0.000118/0.001224 0.999999/*0.000001/-0.004897/0.004570
Accurecy    MSMOTE  1.000000/*0.000000/-0.017404/0.003121    0.887647/0.112353/-0.002537/0.011200    1.000000/*0.000000/-0.009381/0.004056    1.000000/*0.000000/-0.010206/0.001288    1.000000/*0.000000/-0.007729/0.004
Accurecy    MWMOTE  1.000000/*0.000000/-0.013156/0.003621    0.763752/0.236248/-0.001947/0.014649    0.696412/0.303588/-0.000649/0.006839    0.976023/*0.023977/-0.000590/0.001565    0.593153/0.406847/-0.000236/0.0054
Accurecy    SMOTE   1.000000/*0.000000/-0.015221/0.004473    0.792940/0.207060/-0.001888/0.012480    1.000000/*0.000000/-0.006431/0.004011    1.000000/*0.000000/-0.012743/0.001425    0.999995/*0.000005/-0.005487/0.005

14/6
TP  B-SMOTE *0.000000/1.000000/0.030635/0.002709    *0.001979/0.998021/0.009683/0.016939    *0.000000/1.000000/0.012857/0.005672    *0.000000/1.000000/0.009524/0.000000    *0.000000/1.000000/0.014286/0.007078
TP  ADAYSN  *0.000000/1.000000/0.005556/0.004523    *0.000227/0.999773/0.013016/0.018033    *0.000000/1.000000/0.010952/0.004176    0.997044/*0.002956/-0.001111/0.002948    *0.000000/1.000000/0.011270/0.006556
TP  MSMOTE  0.999991/*0.000009/-0.003333/0.003570    *0.000167/0.999833/0.011429/0.015367    *0.000610/0.999390/0.005556/0.008489    *0.000000/1.000000/0.004762/0.000000    *0.000092/0.999908/0.006825/0.008728
TP  MWMOTE  0.257114/0.742886/0.000635/0.005266 0.999917/*0.000083/-0.013492/0.017105    0.926956/*0.073044/-0.002381/0.008731    1.000000/*0.000000/-0.005397/0.002416    0.999999/*0.000001/-0.007937/0.007111
TP  SMOTE   0.737864/0.262136/-0.000794/0.006743    0.879297/0.120703/-0.004286/0.019638    0.108244/0.891756/0.001905/0.008257 0.162791/0.837209/0.000159/0.000869 *0.002726/0.997274/0.003492/0.006368

5/17
FP  B-SMOTE 1.000000/*0.000000/0.008263/0.004675    1.000000/*0.000000/0.012676/0.011213    1.000000/*0.000000/0.019531/0.004496    1.000000/*0.000000/0.004038/0.002051    1.000000/*0.000000/0.016620/0.004075
FP  ADAYSN  *0.000000/1.000000/-0.017746/0.004971    0.994009/*0.005991/0.006385/0.013044    1.000000/*0.000000/0.016901/0.004794    *0.008686/0.991314/-0.000845/0.001835    1.000000/*0.000000/0.014460/0.006177
FP  MSMOTE  1.000000/*0.000000/0.025728/0.004544    0.999938/*0.000062/0.010798/0.013358    1.000000/*0.000000/0.018216/0.004604    1.000000/*0.000000/0.019061/0.002051    1.000000/*0.000000/0.016338/0.004643
FP  MWMOTE  1.000000/*0.000000/0.021315/0.004541    *0.073718/0.926282/-0.004883/0.017968    0.417884/0.582116/-0.000376/0.009834    *0.000043/0.999957/-0.002254/0.002708    *0.003332/0.996668/-0.004319/0.008095
FP  SMOTE   1.000000/*0.000000/0.023756/0.006131    0.555027/0.444973/0.000469/0.018421 1.000000/*0.000000/0.011362/0.006919    1.000000/*0.000000/0.020376/0.002180    1.000000/*0.000000/0.010798/0.007511

3/18
Precision   B-SMOTE 1.000000/*0.000000/-0.010434/0.006496    0.999999/*0.000001/-0.017413/0.015955    1.000000/*0.000000/-0.031340/0.006981    1.000000/*0.000000/-0.006482/0.003357    1.000000/*0.000000/-0.025538/0.006
Precision   ADAYSN  *0.000000/1.000000/0.022116/0.006337    0.989859/*0.010141/-0.008263/0.018428    1.000000/*0.000000/-0.026583/0.007608    0.112056/0.887944/0.000636/0.002803 1.000000/*0.000000/-0.022151/0.009649
Precision   MSMOTE  1.000000/*0.000000/-0.038683/0.006397    0.999968/*0.000032/-0.016331/0.019149    1.000000/*0.000000/-0.029285/0.007307    1.000000/*0.000000/-0.031507/0.003233    1.000000/*0.000000/-0.025582/0.007
Precision   MWMOTE  1.000000/*0.000000/-0.031108/0.006706    0.172107/0.827893/0.004492/0.025585 0.735522/0.264478/-0.001656/0.014231    *0.001712/0.998288/0.002462/0.004230    *0.029116/0.970884/0.004255/0.011818
Precision   SMOTE   1.000000/*0.000000/-0.035263/0.009295    0.675643/0.324357/-0.002131/0.025354    1.000000/*0.000000/-0.018661/0.010660    1.000000/*0.000000/-0.032948/0.003496    1.000000/*0.000000/-0.017317/0.011

4/16
AUC B-SMOTE *0.000000/1.000000/0.011186/0.002706    0.801976/0.198024/-0.001497/0.009516    1.000000/*0.000000/-0.003337/0.002797    *0.000000/1.000000/0.002743/0.001025    0.924690/*0.075310/-0.001167/0.004329
AUC ADAYSN  *0.000000/1.000000/0.011651/0.002898    *0.045794/0.954206/0.003315/0.010407    0.999995/*0.000005/-0.002975/0.003068    0.725005/0.274995/-0.000133/0.001205    0.969284/*0.030716/-0.001595/0.004490
AUC MSMOTE  1.000000/*0.000000/-0.014531/0.002857    0.439813/0.560187/0.000315/0.011301 1.000000/*0.000000/-0.006330/0.004616    1.000000/*0.000000/-0.007150/0.001025    0.999992/*0.000008/-0.004756/0.005044
AUC MWMOTE  1.000000/*0.000000/-0.010340/0.003648    0.945262/*0.054738/-0.004305/0.014279    0.800509/0.199491/-0.001003/0.006415    0.999999/*0.000001/-0.001572/0.001397    0.970330/*0.029670/-0.001809/0.005047
AUC SMOTE   1.000000/*0.000000/-0.012275/0.004424    0.853782/0.146218/-0.002378/0.012145    1.000000/*0.000000/-0.004728/0.003991    1.000000/*0.000000/-0.010108/0.001196    0.999622/*0.000378/-0.003653/0.005315

4/15
G-mean  B-SMOTE *0.000000/1.000000/0.012349/0.002743    0.762584/0.237416/-0.001274/0.009639    0.999999/*0.000001/-0.003135/0.002829    *0.000000/1.000000/0.003040/0.000989    0.883415/0.116585/-0.000969/0.004359
G-mean  ADAYSN  *0.000000/1.000000/0.012090/0.002996    *0.036122/0.963878/0.003579/0.010508    0.999990/*0.000010/-0.002862/0.003076    0.469830/0.530170/0.000017/0.001253 0.951579/*0.048421/-0.001427/0.004555
G-mean  MSMOTE  1.000000/*0.000000/-0.014496/0.002911    0.349287/0.650713/0.000820/0.011478 1.000000/*0.000000/-0.006200/0.004675    1.000000/*0.000000/-0.006554/0.001007    0.999986/*0.000014/-0.004622/0.005112
G-mean  MWMOTE  1.000000/*0.000000/-0.010363/0.003687    0.939643/*0.060357/-0.004199/0.014386    0.753693/0.246307/-0.000824/0.006492    1.000000/*0.000000/-0.001599/0.001407    0.957425/*0.042575/-0.001661/0.005104
G-mean  SMOTE   1.000000/*0.000000/-0.012227/0.004494    0.837023/0.162977/-0.002229/0.012219    1.000000/*0.000000/-0.004604/0.004051    1.000000/*0.000000/-0.009772/0.001190    0.999421/*0.000579/-0.003528/0.005361

3/15
F-measure   B-SMOTE *0.000000/1.000000/0.011558/0.003735    0.965939/*0.034061/-0.003991/0.011537    1.000000/*0.000000/-0.009055/0.003651    *0.000000/1.000000/0.002032/0.001598    0.999999/*0.000001/-0.005602/0.005
F-measure   ADAYSN  *0.000000/1.000000/0.015465/0.003781    0.151886/0.848114/0.002452/0.012828 1.000000/*0.000000/-0.007873/0.004332    0.309239/0.690761/0.000152/0.001649 0.999984/*0.000016/-0.005439/0.006065
F-measure   MSMOTE  1.000000/*0.000000/-0.021489/0.003869    0.754453/0.245547/-0.001825/0.014330    1.000000/*0.000000/-0.011792/0.005509    1.000000/*0.000000/-0.012156/0.001607    1.000000/*0.000000/-0.009376/0.006
F-measure   MWMOTE  1.000000/*0.000000/-0.015913/0.004663    0.864764/0.135236/-0.003785/0.018454    0.803850/0.196150/-0.001375/0.008674    0.999352/*0.000648/-0.001294/0.001990    0.813691/0.186309/-0.001127/0.0068
F-measure   SMOTE   1.000000/*0.000000/-0.018537/0.005881    0.829084/0.170916/-0.002756/0.015621    1.000000/*0.000000/-0.008188/0.005218    1.000000/*0.000000/-0.015767/0.001830    0.999990/*0.000010/-0.006771/0.007
```

圖示內容若看不清楚,請自行放大

函數說明:

以T檢定的方式檢驗變數yourFunction指定的over-sampling方法與預設的五種over-sampling方法在五個分類器環境下的七個效能指標是否有顯著的差異. 並將結果輸出為文字檔, 文字檔名稱為變數outTextFileName所儲存的內容, 文字檔案內關於使用者設計的方法名稱將以變數yourFunctionName所儲存的內容呈現

程式內可修改設定的參數:

1. trainFile : 實驗使用的資料集名稱
2. path : 資料集所在位置的路徑(包含欲使用之資料集的資料夾位址)
3. KFold : 執行K-fold次數
4. KNighbor : K位鄰居的參數設定
5. t_testCount : 執行t-test前各方法的執行次數
6. method : 所有實驗方法的名稱 -輸出文字檔內方法顯示的名稱

   下圖一為在下圖二method處輸入

   {QSTMOE,B-SMOTE,ADAYSN,MSMOTE,MWMOTE,SMOTE}的結果
7. alpha : 檢定的阿法值
8. methodCount : 程式預設為method的成員個數

   加入實驗的方法數量 e.g.設定為3.則共僅有前3個over-sampling方法會透過檢定比較效能.(參考下圖三)

DataSet	KNN DT Method	AUC	NaiveB AUC	LogitR AUC	AUC	AUC
wdbc.mat	QSMOTE	0.921496	0.922468	0.968813	0.922803	0.962643
wdbc.mat	B-SMOTE	0.910463	0.918679	0.968712	0.922267	0.966868
wdbc.mat	ADAYSN	0.904058	0.912408	0.968712	0.924212	0.948357
wdbc.mat	MSMOTE	0.934071	0.925285	0.970121	0.931690	0.957344
wdbc.mat	MWMOTE	0.914453	0.909893	0.952783	0.925721	0.951911
wdbc.mat	SMOTE	0.926492	0.920523	0.959725	0.930282	0.964923

```
function [  ] = Performance_t_test( yourFunction,yourFunctionName,outTextFileName )

%  dbstop if error %如果有問題matlab會停在出錯的那行，並且保存所有相關變數-除錯用

tic  %碼表開始倒數

%使用的訓練集
trainFile={'abalone9-18.mat','Btissue.mat','ecoli1.mat'...
    ,'pima.mat','vehicle1.mat'...
    ,'yeast.mat','segment0.mat'...
    ,'wisconsin.mat','appendicitis.mat','bupa.mat','heart.mat'...
    ,'haberman.mat','glass0.mat','new-thyroid1.mat','ecoli3.mat','vehicle3.mat','wdbc.mat'...
    ,'cleveland-0_vs_4.mat','page-blocks0.mat','Robot.mat'};

%%設定資料集所在根目錄
path='./data/';
%%設定K-FOLD次數
KFold=5;
%%設定鄰居數
KNighbor=5;
%做test n次
t_testCount=5;
%實驗方法名稱-輸出用
method={yourFunctionName,'B-SMOTE','ADAYSN','MSMOTE','MWMOTE','SMOTE'};
%檢定的阿法值
alpha=0.1;
%方法數
methodCount=6;
```

```
90          %執行match個方法以及在五個環境
91 -        for match=1 : methodCount
92 -            switch match
93 -                case {1}
94 -                    yourDataSet=yourFunction(trainData,majorClassNo,minorClassNo,KNighbor);
95 -                    [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(yourDataSet,testDa
96 -                    [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(yourDataS
97 -                    [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(yourDataSet,testDat
98 -                    [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(yourDataSet,testData
99 -                    [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(yourData
00
01 -                case {2}
02 -                    Borderline_SMOTEDataSet=Borderline_SMOTE(trainData,minorClassNo,KNighbor);
03 -                    [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(Borderline_SMOTEDa
04 -                    [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(Borderlin
05 -                    [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(Borderline_SMOTEDat
06 -                    [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(Borderline_SMOTEData
07 -                    [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(Borderli
08
09 -                case {3}
10 -                    ADAYSNDataSet=ADAYSN(trainData,minorClassNo,KNighbor);
11 -                    [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(ADAYSNDataSet,test
12 -                    [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(ADAYSNDat
13 -                    [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(ADAYSNDataSet,testD
14 -                    [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(ADAYSNDataSet,testDa
15 -                    [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(ADAYSNDa
16
17 -                case {4}
18 -                    MSNOTEDataSet=MSNOTE(trainData,minorClassNo,radio,KNighbor);
```

備註 ： 1. 請將 Case 1 處固定為呼叫你設計的 over-sampling 方法的呼叫處

上圖 94-99 行執行內容為

“呼叫使用者設計的 over-sampling 方法產生新的訓練集 ， 並以此訓練集 ，依序訓練 KNN/DT/SVM/NB/Logistic 五個分類器.並回傳此五個分類器在七項指標的效能表現 “

若欲修改比較對象方法 ， 請修改

For match 迴圈內 Case 2 到 6 的敘述

Case2-6 敘述執行的 over-sampling 方法依序為

B-SMOTE, ADAYSN, MSMOTE, MWMOTE, SMOTE

運作邏輯與 Case1 相同

2. 將dbstop if error的註解拿掉 ， 若是程式有錯誤則matlab會停在錯誤發生的那一行 ， 除錯用

Performance_Wilcoxon

輸入參數:

1. yourFunction : 你欲呼叫的方法檔案名稱－作為日後程式自動幫你呼叫方法的呼叫依據
2. yourFunctionName : 你的方法名稱 － 顯示在輸出檔案中的你的方法名稱
3. outTextFileName : 產生的文字檔名稱 － 用來避免產生相同文字檔名稱(舊的會被覆蓋)

輸出內容:

1. 變數yourFunction指定的over-sampling方法對預設的五種over-sampling方法在五種分類器環境下的七個效能指標使用Wilcoxon檢定的檢定結果
   (文字檔一個)
2. 變數yourFunction指定的over-sampling方法對預設的五種over-sampling方法在五種分類器環境下的七個效能指標
   (實驗使用的每一個資料集會輸出一個文字檔)

函數說明:

以Wilcoxon sign rank檢定的方式檢驗變數yourFunction指定的方法與預設的五種over-sampling方法在五個分類器環境下的七個效能指標是否有顯著的差異. 並將結果輸出為文字檔, 文字檔名稱為變數outTextFileName所儲存的內容, 文字檔案內關於使用者設計的方法名稱將以變數yourFunctionName所儲存的內容呈現

程式內可修改設定的參數:

1. trainFile : 使用的訓練集名稱
2. path : 訓練集所在位置的路徑(包含欲使用之資料集的資料夾位址)
3. KFold : 執行K-fold次數
4. KNighbor : K位鄰居的參數設定
5. method : 所有實驗方法的名稱 －輸出文字檔內方法顯示的名稱(下圖一)
6. alpha : 檢定的阿法值
7. methodCount : 程式預設為method的成員個數
   加入實驗的方法數量 e.g. 設定為3. 則僅有3個方法參與檢定.

備註 : 將 dbstop if error 的註解拿掉 , 若是程式有錯誤則 matlab 會停在錯誤發生的那一行 , 除錯用

		KNN	DT	SVM	NaiveB	LogitR
	Method	P_Value	P_Value	P_Value	P_Value	P_Value
TP	B-SMOTE	*0.001531/0.998672	*0.010894/0.990210	*0.000543/0.999524	*0.072267/0.933537	*0.001235/0.998932
TP	ADAYSN	*0.002569/0.997734	*0.001795/0.998459	*0.005233/0.995496	*0.073242/0.936401	*0.002251/0.998062
TP	MSMOTE	*0.023175/0.978945	*0.088396/0.918382	*0.002415/0.997947	*0.016577/0.985280	*0.000994/0.999143
TP	MWMOTE	0.516052/0.500000	0.904721/0.102282	0.961706/*0.041763	0.952698/*0.053497	0.996515/*0.004013
TP	SMOTE	0.147963/0.861870	0.960002/*0.043590	0.308204/0.706954	0.652522/0.363707	0.180155/0.831053

```
%執行match個方法以及在五個環境
for match=1 : methodCount
    switch match
        case {1}
            yourDataSet=yourFunction(trainData,majorClassNo,minorClassNo,KNighbor);
            [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(yourDataSet,testDa
            [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(yourDataS
            [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(yourDataSet,testDat
            [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(yourDataSet,testData
            [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(yourData

        case {2}
            Borderline_SMOTEDataSet=Borderline_SMOTE(trainData,minorClassNo,KNighbor);
            [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(Borderline_SMOTEDa
            [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(Borderlin
            [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(Borderline_SMOTEDat
            [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(Borderline_SMOTEData
            [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(Borderli

        case {3}
            ADAYSNDataSet=ADAYSN(trainData,minorClassNo,KNighbor);
            [tempknnAccurecy,tempknnTPK,tempknnFP,tempknnPrecision,tempknnAUC,tempknnG_mean,tempknnF_measure]=KnnA(ADAYSNDataSet,test
            [tempTreeAccurecy,tempTreeTP,tempTreeFP,tempTreePrecision,tempTreeAUC,tempTreeG_mean,tempTreeF_measure]=decTree(ADAYSNDat
            [tempSVMAccurecy,tempSVMTPK,tempSVMFP,tempSVMPrecision,tempSVMAUC,tempSVMG_mean,tempSVMF_measure]=Svm(ADAYSNDataSet,testD
            [tempNBAccurecy,tempNBTP,tempNBFP,tempNBPrecision,tempNBAUC,tempNBG_mean,tempNBF_measure]=naiveBayes(ADAYSNDataSet,testDa
            [tempLogRAccurecy,tempLogRTP,tempLogRFP,tempLogRPrecision,tempLogRAUC,tempLogRG_mean,tempLogRF_measure]=logistic(ADAYSNDa

        case {4}
            MSNOTEDataSet=MSNOTE(trainData,minorClassNo,radio,KNighbor);
```

備註： 1. 請將 Case 1 固定為呼叫你的 over-sampling 方法的使用處

上圖 94-99 行執行

“呼叫你的 over-sampling 方法產生新的訓練集 ， 並以此訓練集 ， 依序訓練 KNN/DT/SVM/NB/Logis 五個分類器. 並回傳此五個分類器在各項指標的效能表現 “

若欲修改比較對象方法 ， 請修改.

For match 迴圈內 Case 2 到 6 的敘述

Case2-6 敘述執行的 over-sampling 方法依序為

B-SMOTE, ADAYSN, MSMOTE, MWMOTE, SMOTE

運作邏輯與 Case1 相同

```
%% 各項指標在所有資料集中贏的總共count數
% method={'QSMOTE','B-SMOTE','ADAYSN','MSMOTE','MWMOTE','SMOTE'};
% indexName={'Accurecy','TP','FP','Precision','AUC','G-mean','F-measure'};
% fileID=fopen('count.txt','w');
% for index_choice=1:7
%     for Competitor_choice=2:methodCount
%         switch index_choice
%             case {1}
%                 %                 Accurecy
%
%                 c1=length(find(KNN_Accurecy(1,:)-KNN_Accurecy(Competitor_choice,:)>0));
%                 c2=length(find(DT_Accurecy(1,:)-DT_Accurecy(Competitor_choice,:)>0));
%                 c3=length(find(SVM_Accurecy(1,:)-SVM_Accurecy(Competitor_choice,:)>0));
%                 c4=length(find(NaiveB_Accurecy(1,:)-NaiveB_Accurecy(Competitor_choice,:)>0));
%                 c5=length(find(logisticR_Accurecy(1,:)-logisticR_Accurecy(Competitor_choice,:)>0));
%
%             case {2}
%                 %                 TP
%                 c1=length(find(KNN_TP(1,:)-KNN_TP(Competitor_choice,:)>0));
%                 c2=length(find(DT_TP(1,:)-DT_TP(Competitor_choice,:)>0));
%                 c3=length(find(SVM_TP(1,:)-SVM_TP(Competitor_choice,:)>0));
%                 c4=length(find(NaiveB_TP(1,:)-NaiveB_TP(Competitor_choice,:)>0));
%                 c5=length(find(logisticR_TP(1,:)-logisticR_TP(Competitor_choice,:)>0));
%             case {3}
%                 %                 FP
```

上圖程式碼 : 輸出 在各個分類器環境(KNN/DT/SVM/NB/Logistic)的七項效能指標下，變數 yourFunction 指定的 over-sampling 方法勝過其他 over-sampling 的贏的數目(總計所有的資料集,下圖使用 20 個資料集,因此最大數字為 20)(贏不需要具備顯著差異)。以下圖矩形選取區域為例，yourFunction 指定的 over-sampling 方法在 TP_Rate 這個效能指標勝過 B-SMOTE 的個數，每一個資料集會比一次，因此二十個資料集會比二十次，每贏一次，則 count 數加一。因此矩形選取區域的 16 代表說 yourFunction 指定的 over-sampling 方法在 20 個資料集合中的表現有 16 個資料的表現是勝過 B-Smote。(功能目前關閉-若欲打開,請解除註解)

下圖為其輸出文字檔之結果

```
Accurecy    B-SMOTE 6   4   3   5   4
Accurecy    ADAYSN  3   8   2   2   3
Accurecy    MSMOTE  3   8   4   6   6
Accurecy    MWMOTE  12  17  13  12  13
Accurecy    SMOTE   7   7   11  5   13

TP  B-SMOTE 16  15  17  12  17
TP  ADAYSN  17  14  14  11  17
TP  MSMOTE  14  14  16  13  15
TP  MWMOTE  11  10  6   5   9
TP  SMOTE   14  6   10  9   13

FP  B-SMOTE 3   2   2   3   2
FP  ADAYSN  1   3   2   2   3
FP  MSMOTE  3   7   3   6   4
FP  MWMOTE  11  13  12  12  15
FP  SMOTE   5   8   8   7   9

Precision   B-SMOTE 5   4   3   4   3
Precision   ADAYSN  5   5   4   1   4
Precision   MSMOTE  4   8   6   5   6
Precision   MWMOTE  11  16  14  10  16
Precision   SMOTE   6   5   10  5   11

AUC B-SMOTE 15  12  15  11  18
AUC ADAYSN  12  10  12  6   12
AUC MSMOTE  10  12  11  12  13
AUC MWMOTE  12  14  10  11  13
AUC SMOTE   10  6   14  7   14

G-mean  B-SMOTE 15  13  15  11  17
G-mean  ADAYSN  12  12  11  7   13
G-mean  MSMOTE  9   12  13  10  14
G-mean  MWMOTE  12  15  12  10  13
G-mean  SMOTE   9   6   13  7   15

F-measure   B-SMOTE 8   11  10  6   12
F-measure   ADAYSN  10  9   10  6   10
F-measure   MSMOTE  8   10  12  10  12
F-measure   MWMOTE  10  16  12  11  14
F-measure   SMOTE   7   8   13  6   13
```

範例文字檔

```
% % Wilcoxon rank sum test
% alpha=0.1;
% method={'QSMOTE','B-SMOTE','ADAYSN','MSMOTE','MWMOTE','SMOTE'};
% indexName={'Accurecy','TP','FP','Precision','AUC','G-mean','F-measure'};
% fileID=fopen('Wilcoxon rank sum test.txt','w');
% for index_choice=1:7
%     fprintf(fileID,'\t\t%s\t%s\t%s\t%s\t%s\n','KNN','DT','SVM','NaiveB','LogitR');
%     fprintf(fileID,'\t%s\t%s\t%s\t%s\t%s\t%s\n','Method','P_Value','P_Value','P_Value','P_Value','P_Value');
%     for Competitor_choice=2:6
%         switch index_choice
%             case {1}
%                 %                 Accurecy
%                 [p1,h1]=ranksum(KNN_Accurecy(1,:),KNN_Accurecy(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p2,h2]=ranksum(DT_Accurecy(1,:),DT_Accurecy(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p3,h3]=ranksum(SVM_Accurecy(1,:),SVM_Accurecy(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p4,h4]=ranksum(NaiveB_Accurecy(1,:),NaiveB_Accurecy(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p5,h5]=ranksum(logisticR_Accurecy(1,:),logisticR_Accurecy(Competitor_choice,:),'tail','right','alpha', alpha);
%
%             case {2}
%                 %                 TP
%                 [p1,h1]=ranksum(KNN_TP(1,:),KNN_TP(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p2,h2]=ranksum(DT_TP(1,:),DT_TP(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p3,h3]=ranksum(SVM_TP(1,:),SVM_TP(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p4,h4]=ranksum(NaiveB_TP(1,:),NaiveB_TP(Competitor_choice,:),'tail','right','alpha', alpha);
%                 [p5,h5]=ranksum(logisticR_TP(1,:),logisticR_TP(Competitor_choice,:),'tail','right','alpha', alpha);
%             case {3}
%                 %                 FP
```

此段程式碼：(功能已關閉-若遇打開,請解除註解)

使用 Wilcoxon rank sum test 檢定 變數 yourFunction 指定的 over-sampling 方法在各項指標在各個分類器環境(KNN/DT/SVM/NB/Logistic)下與其他 over-sampling 方法的比較結果是否具有顯著差異。

(輸出的文字格式與 Wilcoxon sign rank 相同，僅是使用的統計檢定方法不同。)

實作方法程式 ：

Smote ： SMOTE Synthetic Minority Over-sampling Technique-2002

輸入參數

1. trainSet : 原始資料集
2. minorClassNo : 少數類別的代號
3. N : N 是少數類別增加的倍數
4. K : K 是 KNN 演算法中的最近 K 個鄰居數

輸出變數

經過 SMOTE 這個 over-sampling 方法處理後的資料集

函數說明

實作 SMOTE 方法替少數類別實例產生人造實例

ADAYSN ： ADASYN Adaptive Synthetic Sampling Approach for Imbalanced Learning -2008

輸入參數

1. trainSet : 原始資料集
2. minorClassNo : 少數類別的代號
3. K : K 是 KNN 演算法中的最近 K 個鄰居數

輸出變數

經過 ADAYSN 這個 over-sampling 方法處理後的資料集

函數說明

實作 ADASYN 方法替少數類別實例產生人造實例

Borderline_SMOTE: Borderline-SMOTE A New Over-Sampling Method in Imbalanced Data Sets Learning -2005

輸入參數

1. trainSet :資料集
2. minorClassNo :少數類別的代號
3. K : K 是 KNN 演算法中的最近 K 個鄰居數

輸出變數

經過 Borderline-SMOTE 這個 over-sampling 方法處理後的資料集

函數說明

實作 Borderline-SMOTE 方法替少數類別實例產生人造實例

MSNOTE: MSMOTE Improving classification performance when training data is imbalanced-2009

輸入參數

1. trainSet :資料集
2. minorClassNo : 少數類別的代號
3. N : N 是少數類別增加的倍數
4. K : K 是 KNN 演算法中的最近 K 個鄰居數

輸出變數

經過 MSMOTE 這個 over-sampling 方法處理後的資料集

函數說明

實作 MSMOTE 方法替少數類別實例產生人造實例

MWMOTE: MWMOTE—Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning-2014

輸入參數

1. trainSet :資料集
2. minorClassNo : 少數類別的代號
3. N : N 是少數類別增加的倍數
4. K1 : 找出雜訊實例時的 KNN 的 K 參數
5. K2 : 找出邊界多數類別時的 KNN 的 K 參數
6. K3 : 找出邊界少數類別時的 KNN 的 K 參數

輸出變數

經過 MWMOTE 這個 over-sampling 方法處理後的資料集

函數說明

實作 MWMOTE 方法替少數類別實例產生人造實例

QSMOTE2 :

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

每一個少數類別(去除 noise 後)找他最近的多數類別,以此距離各掃一圈算出此兩點附近的其各自的同類別實例有多少. 每一個少數類別與距離他最近的多數類別內插產生人造實例

距離限制 d = $\frac{\text{多數類別同伴數}+1}{\text{多同伴數}+\text{少同伴數}+1+1}$

每一個**少數類別產生個數 N 皆相同 : 設定 N 為使資料集到達類別平衡的數字.** 使用”距離限制 d”對人造實例的產生位置做出限制

QSMOTE2_1 ：(論文的方法)

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

每一個少數類別(去除 noise 後)找他最近的多數類別,以此距離各掃一圈算出此兩點附近的其各自的同類別實例有多少. 每一個少數類別與距離他最近的多數類別內插產生人造實例

距離限制 d = $\frac{\text{多數類別同伴數+1}}{\text{多同伴數+少同伴數+1+1}}$

權重 w = $\frac{\text{多數類別同伴數+1}}{\text{少數類別同伴數+1}}$

每一個少數類別產生個數不一定相同 : 依照該少數類別實例的權重 w 決定分配人造實例的多寡 , 使用”距離限制 d”對人造實例的位置做出限制

QSMOTE2_1_1 :

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

每一個少數類別(去除 noise 後)找他最近的多數類別,以此距離各掃一圈算出此兩點附近的其各自的同類別實例有多少. 每一個少數類別與距離他最近的多數類別內插產生人造實例

距離限制 d = 無

權重 w = $\frac{\text{多數類別同伴數}+1}{\text{少數類別同伴數}+1}$

每一個少數類別產生個數不一定相同 : 依照該少數類別實例的權重決定分配人造實例數的多寡 , **並沒有對人造實例的產生位置作限制**

QSMOTE2_1_2

輸入參數

1. trainSet : 資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

每一個少數類別(去除 noise 後)找他最近的多數類別,以此距離各掃一圈算出此兩點附近的其各自的同類別實例有多少. 每一個少數類別與距離他最近的多

數類別內插產生人造實例

距離限制 d = $\frac{1}{\text{少數類別同伴數}+1}$

權重 w = $\frac{\text{多數類別同伴數}+1}{\text{少數類別同伴數}+1}$

每一個少數類別產生個數不一定相同 : 依照該少數類別實例的權重決定分配人造實例的多寡 , **修改人造實例產生位置(“距離限制”)的公式**

QSMOTE2_2 :

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

得到邊界少數類別實例後(去除 noise 後),對每一個邊界少數類別實例,找他最近的多數類別,以此距離各掃一圈算出此兩點附近的其各自的同類別實例有多少. 每一個邊界少數類別與距離他最近的多數類別內插產生人造實例

$$\text{距離限制 d} = \frac{\text{多數類別同伴數}+1}{\text{多同伴數}+\text{少同伴數}+1+1}$$

$$\text{權重 w} = \frac{\text{多數類別同伴數}+1}{\text{少數類別同伴數}+1}$$

只針對邊界少數類別實例產生人造實例, 對每一個邊界少數類別實例產生人造實例個數不一定相同 : 依照該少數類別實例的權重 w 決定分配人造實例的多寡 , 使用”距離限制 d”對人造實例位置做出限制

QSMOTE3 ：（以下 3 開頭的程式重點在如何分群-目前無使用）

輸入參數

1. trainSet : 資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

1. 每一個少數類別實例(去除 noise 後)以它到他最近的多數類別的距離,掃一圈,若有掃到同類別的實例,則將其加入成為一個集合,若集合之間有共同的元素,則將其合併成一個集合,
2. N 為達成資料集類別比相同的倍率。
   替每一個邊界少數類別實例產生相同個數 N 的人造實例 - 選擇邊界少數類別實例後,依據其所屬的群, 由其群之中隨機選擇一個少數類別實例. 由兩者內插產生人造實例（替每一個邊界實例產生相同數目的人造實例）

QSMOTE3_1

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

1. 每一個少數類別實例(去除 noise 後)以它到他最近的多數類別的距離, 掃一圈,若有掃到同類別的實例,則將其加入成為一個集合, 若集合之間有共同的元素,則將其合併成一個集合.
2. 替每一個邊界少數類別實例產生人造實例 - 選擇邊界少數類別實例後, 依據其所屬的群, 由其群之中隨機選擇一個少數類別實例. 由兩者內插產生人造實例.(替每一個邊界實例產生相同數目的人造實例 N，N 為達成資料集類別比相同的倍率。)
3. 若某群只有一個實例, 則不隨機選取該群中任一點做內插(因為只有一點),而是選擇距離此少數類別實例最近的實例,兩點做內插

QSMOTE3_1_1

輸入參數

1. trainSet : 資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

1. 每一個少數類別實例(去除 noise 後)以它到他最近的多數類別的距離, 掃一圈,若有掃到同類別的實例,則將其加入成為一個集合, 若集合之間有共同的元素,則將其合併成一個集合,
2. 替每一個邊界少數類別實例產生人造實例 - 隨機選擇距離其最近的 K 個實例中的一個 , 兩點做內插
3. 實例權重大小與實例所屬群的個數大小成反比 , 權重越大則實例分配越多的人造實例. 權重公式:$\frac{1}{群的成員個數}$

QSMOTE3_1_2

輸入參數

1. trainSet :資料集
2. majorClassNo : 多數類別的編號
3. minorClassNo : 少數類別的編號
4. K : 找出雜訊實例時的 KNN 的 K 參數

輸出變數

經過此程式這個 over-sampling 方法處理後的資料集

函數說明

1. 每一個少數類別實例(去除 noise 後)以它到他最近的多數類別的距離,掃一圈,若有掃到同類別的實例,則將其加入成為一個集合, 集合和集合並不合併,
2. 替每一個邊界少數類別實例產生人造實例 –找出包含此實例的所有集合,選擇其中個數最大的,作為該實例隸屬的群,由該群之中隨機選擇一點,兩點做內插
3. 每一點邊界實例的權重不同, 依據其所屬群的個大小以及密度

權重= $\frac{\textit{多數類別個數}}{\textit{少數類別個數}+1} * \frac{\textit{所屬群的個數大小}}{\textit{所有群的個數加總}}$ ，權重越大則該實例分配的人造實例數越多

分類器演算法

KnnA

輸入參數

1. Trainset : 訓練集
2. testSet : 測試集
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號
5. featureNum : 訓練集的feature個數

輸出變數

所建立的 KNN 分類器的七項效能指標
accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

依據訓練集建立KNN分類器模型, 並以測試集得到此KNN分類器的七項效能指標(accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

備註:

```
Mdl = fitcknn(trainSet(:,1:featureNum-1),trainSet(:,featureNum),'NumNeighbors',10);
```

可以用 NumNeighbors 設定 KNN 分類器的 K 參數設定
上圖為使用 10 作為參數 K
上圖由左至右的輸入為

1. 資料集的 feature 資料
2. 資料集的 label 資料
3. K 參數設定

decTree

輸入參數

1. Trainset : 訓練集
2. testSet : 測試集
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號
5. featureNum : label feature在訓練集的行數

輸出變數

所建立的決策樹分類器的七項效能指標

accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

依據訓練集建立決策樹分類器模型,並以測試集得到此決策樹分類器的七項效能指標(accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

備註:

```
CMdl = fitctree(trainSet(:,1:featureNum-1),trainSet(:,featureNum),'MergeLeaves','off');
```

可以用 MergeLeaves 設定決策樹不會合併樹葉.

上圖為關閉合併樹葉功能

上圖由左至右的輸入為

1. 資料集的 feature 資料
2. 資料集的 label 資料
3. 關閉合併樹葉功能

Svm

輸入參數

1. Trainset : 訓練集
2. testSet : 測試集
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號
5. featureNum : label feature在訓練集的行數

輸出變數

所建立的 SVM 分類器的七項效能指標
accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

依據訓練集建立SVM分類器模型,並以測試集得到此SVM分類器的七項效能指標(accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

備註:

```
CMdl = fitcsvm(trainSet(:,1:featureNum-1),trainSet(:,featureNum),'Standardize',true );
```

對使用之訓練集執行標準化動作.以建立 SVM model
上圖為打開標準化功能
上圖由左至右的輸入為

1. 資料集的 feature 資料
2. 資料集的 label 資料
3. 打開標準化功能

naiveBayes

輸入參數

1. rainset : 訓練集
2. testSet : 測試集
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號
5. featureNum : label feature在訓練集的行數

輸出變數

所建立的 naiveBayes 分類器的七項效能指標

accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

依據訓練集建立naiveBayes分類器模型,並以測試集得到此naiveBayes分類器的七項效能指標

(accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

Logistic

輸入參數

1. Trainset : 訓練集
2. testSet : 測試集
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號
5. featureNum : label feature在訓練集的行數

輸出變數

所建立的羅吉斯分類器的七項效能指標

accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

依據訓練集建立羅吉斯分類器模型,並以測試集得到此羅吉斯分類器的七項效能指標(accuracy,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

額外功用函數

findRank

輸入參數

1. dist : 原本的資料
2. data_sorted : 排序後的資料
3. K : K個鄰居數

輸出變數

最近K位鄰居名次在原資料集的編號

函數說明

用來找出K位鄰居的距離名次順序

returnPredition

輸入參數

1. test : 測試集
2. finalDecison : 分類器在測試集做出的預測
3. majorClassNo : 多數類別的代號
4. minorClassNo : 少數類別的代號

輸出變數

回傳分類器的效能指標

(average,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

函數說明

計算指定分類器的效能指標

(average,TP_rate,FP_rate,Precision,AUC,G_mean,F_measure)

recognizeMajorClassAndOtherClass

輸入參數

1. Data ： 全部的資料

輸出變數

多數類別以及少數類別的在資料集合的 label 編號

函數說明

用來辨認兩個類別之中 哪一個代號是多數類別,哪一個代號是少數類別.

並分別回傳其代號