第一章 緒論

1.1研究動機

工廠機器的各項指標會反映出該機器當前的使用情形以及是否需要立即維護，對於不允許容錯的產線而言，能預測機器故障以及故障的類別便顯得很重要，這可以幫助公司進行品管的超前部屬，提前將需要維護的機器抓出來，降低最終有問題的產品數量，當產品出問題時，也可以排除是機器所造成的影響，這是身為工工系的學生能幫助到公司的事情。而我們要做的專題便是機器預測性維護，透過我們所找到的資料，利用演算法，分析預測在什麼樣的條件(變因)下，機器會故障。

1.2目的

我們將透過機器學習比較各種演算法在預測機器故障時的準確性，並且找出較佳的預測模型與成果，如此一來當機器呈現該類型的錯誤時，就能知道其Type I error和Type II error的大小，經由整理和歸納後可以幫助公司預測機器的故障情形，減少不必要的損失。

第二章 機器學習

2.1 機器學習

機器學習讓機器(計算機)具有學習能力並能從資料中自動學習規則，並利用規則對新的資料進行預測，其主要是設計和分析可以自動學習的演算法，讓計算機可以從過去的資料或經驗當中建立一個模型(Model)，而學習(Learning)就是指建模的過程。機器學習可以分成下面幾種類別:

1. 監督式學習(Supervised Learning) : 可以由訓練集資料(Training Data)及其對應的標籤(Label)中學到或建立一個模型，並依此模型預測新的案例。分類(Classification)是最常見到的監督式學習演算法。而決策樹(Decision Tree)就是監督式學習(Supervised Learning)中常見的方法，決策樹是以樹狀為基礎的演算法，透過歸納規則將資料從樹根開始分類，一節一節尋找最佳分割點來將資料分成為小單位的集合，不過當訓練資料集內的數目太少，而變數太多時，分類的效果會變差。另外，決策樹在分類上屬於固定的路徑，沒辦法像隨機森林或是XGBOOST在分類過程有容錯能力。
2. 非監督式學習(Unsupervised Learning) : 與監督式學習不同，訓練資料中並沒有對應的標籤(Label)。分群(Clustering)是最常見到的非監督式學習演算法。
3. 演化式學習(Evolutionary Learning) : 主要是基於模仿生物演化及行為所發展出來的學習演算法，例如基因演算法(Genetic Algorithm)、粒子群演算法(Particle Swarm Algorithm)、灰狼演算法(Grey Wolf Optimizer)、蜂群演算法(Bee Colony)等。
4. 混合式學習(Hybrid Learning) : 主要是結合多種演算法的優點，藉以提升學習的效能(Performance)或效率(Efficiency)，其結構通常是串連或是階層式的排列多個演算法。

在機器學習領域可以結合多個分類模型，以達到更佳之分類效能，而此種方法徵為整合式學習(Ensemble Learning)。除了以上幾類之外，還有半監督式學習(Semi-supervised Learning) 、單類別學習(One-class Learning)與增強式學習(Reinforcement Learning)等情境，可以應用在不同的實際問題中。

2.1.1隨機森林(Random Forest)：

隨機森林是一個包含多個決策樹的分類器，並且其輸出的類別是由個別樹輸出的類別的眾數而定。基本原理是每棵決策樹隨機選擇特徵，最後建構一個森林，可以比決策樹不容易過度擬合，並且讓預測能力提升。顧名思義就是由許多不同的決策樹所組成的一個學習器，這種方法又稱為Ensemble Method。其方法大致如下:

1.從訓練集(母體N)抽樣取出n筆資料(採抽樣可放回方式)來建立決策樹，並使用30%的資料當作test data來預測

2.每棵決策樹採用隨機挑選K個特徵建立

3.重複M次建立M棵決策樹

4.分類(預測)過程中，若為線性問題採平均數，非線性問題採眾數(或稱投票機制)

2.1.2 XGBOOST：

XGBOOST適用於數值資料分析，也非常適合應用於此維修保養預測的專題使用。基本原理是利用平行化運算執行 Gradient Boosting 演算法，且包含剪枝(決策樹修剪)，處理遺漏值和正規化處理，避免過度擬合資料。XGBOOST相比其他演算法的優點是訓練速度快、準確率也高。

2.2 SVM：

支援向量機是機器學習中的典型分類器，也是監督式學習(Supervised Learning)的一種，SVM使用最大化邊際的原理，想辦法將兩(多)個類別盡量地分開。當資料集在 k 維空間中並不是線性可分的時候，可以在目標函數部分加入懲罰值 (penalty)以求得軟性界線(soft bound)，或是透過核函數(kernel function)將資料點轉換到更高維度的空間，就有機會能夠透過超平面(hyperplane)將資料集分開。

2.3 KNN：

KNN分類法是監督式學習(Supervised Learning)的一種，其基本原理是以點間的距離作為分類的標準並且同時帶入訓練集和測試集，對於每一個未分類的樣本點，參考k個與自己最近的樣本點的Label，以多數決投票的方式進行類別的決定，k的值一般可以採樣本點數的平方根做參考，或是採用訓練集的分類準度最高的k值。

第三章 研究方法與步驟

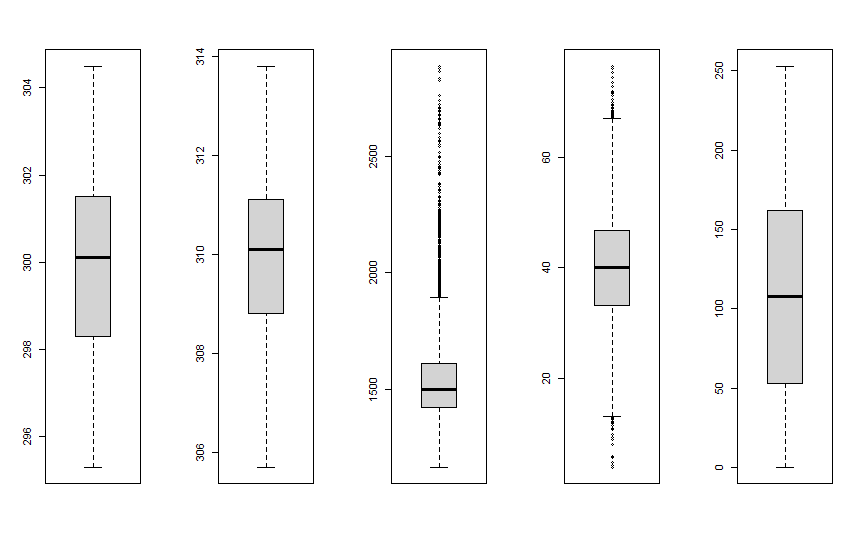
3.1資料說明

我們所獲得的資料是網路上找到的AI4I 2020預測性維護數據集(可參考第六章、資料來源)，此數據集是一個合成數據集，反映了行業中遇到的真實預測性維護數據。而透過此數據集，我們可以用來預測機器故障(二進位)和類型(多類)，此數據集由10000筆資料構成並用行的方式儲存成10個類別(自變數)進行分類，類別名稱如下表Y

|  |  |
| --- | --- |
| 類別名稱 | 說明 |
| UDI(Unique Device Identification) | 資料的唯一標籤 |
| Product ID | 產品編號 |
| Type (產品類型) | 取Product ID的首字母當作產品類型，共有L、M、H三種，表示低（佔所有產品的 50 %）、中 （30 %） 和高 （20 %） |
| Air temperature[K] (空氣溫度) | 使用RW法生成並將資料常態化，中心為300K，標準差為2K |
| Process temperature[K] (過程溫度) | 使用RW法生成並將資料常態化，中心為310K，標準差為1K |
| Rotational speed [rpm] (轉速) | 根據2860W的功率計算得出並且疊加常態分佈的雜訊 |
| Torque [Nm] (扭矩) | 常態分佈，中心為40Nm左右，Ïƒ = 10 Nm，沒有負值 |
| Tool wear [min] (刀具磨損) | 不同的產品類型( L / M / H ) 在加工過程中使用刀具的刀具磨損增加 2 / 3 / 5 分鐘。 |
| Target(目標) | 用0跟1表示Not Failure和Failure |
| Failure Type(錯誤類型) | 共有五種 : Tool Wear Failure、Random Failures、Power Failure、Overstrain Failure、Heat Dissipation Failure |

3.1.1 資料前處理

處理這筆資料時，確定無遺漏值後，我們繪製了此五種錯誤類型的盒鬚圖，發現Heat Dissipation Failure(中間那個圖)有很多離群值，因此我們試著將離群值拿掉後進行預測，但結果相當不理想，反而保留這些離群值會有較好的預測結果，顯示出剛好離群值裡面的這些資料預測出來的結果是我們要的，因此必須將這些離群值留下來。



3.2軟體介紹

3.2.1 R語言介紹

此專題我們是使用R語言來完成的，R語言是一種自由軟體程式語言與操作環境，主要用於統計分析、繪圖以及資料探勘。R由紐西蘭奧克蘭大學的統計學家羅斯·伊哈卡和羅伯特·傑特曼開發，現在由R核心小組負責開發。R主要用於資料分析。在R語言中，用於資訊儲存的資料結構包括向量、陣列、列表以及資料框。使用者可以用R來進行一些基本的統計檢驗，構建線性及非線性的模型，對時間序列加以分析，或對資料進行分類與聚類分析。R的另一強項是繪圖功能，畫出的圖表能夠達到專業出版物的要求，也可加入數學符號。模擬訓練中使用的套件如下表X

表X : R語言套件說明

|  |  |
| --- | --- |
| 套件名稱 | 說明 |
| rpart | 我們使用決策樹的套件為rpart，我們將資料其中的70%作為訓練集、30%作為測試集，並透過rpart此函式建立決策樹模型。 |
| rpart.plot | 用來繪出決策樹的套件。 |
| randomForest | 在R裡建立隨機森林時可以使用randomForest套件建立模型，建立的隨機森林模型和決策樹模型都可以對新資料進行分類、預測結果。 |
| performanceEstimation | 使用此套件中的smote語法來處理資料不平衡的問題。 |
| xgboost | 使用此套件來建立xgboost演算法的模型 |
| class | 使用一個函數 class()可讓R語言告訴我們輸入的變數是什麼型別。 |
| e1071 | 使用此套件來建立支援向量機(SVM)的模型。 |

3.3研究流程

此專題我們使用了以下四種演算法 : 隨機森林、XGBOOST、 SVM、 KNN以及使用SMOTE來處理資料不平衡的問題，以下為我們的研究流程圖。

資料取得

使用KNN來進行預測

使用SVM來進行預測

確定資料完整性(檢查是否有遺漏值和離群值)

使用XGBOOST來進行預測

使用隨機森林來進行預測

分析預測結果並整理較好的預測方式

第四章 研究過程與成果

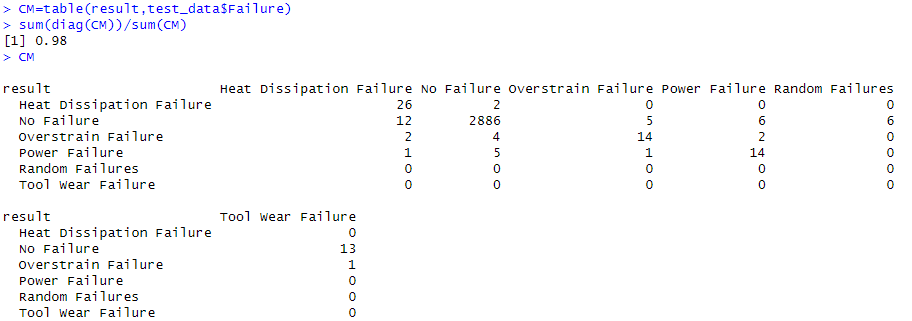
以下說明如何套用這些演算法來幫助我們進行機器故障的預測以及研究過程

4.1全分類預測

取全部資料後進行第一次分析

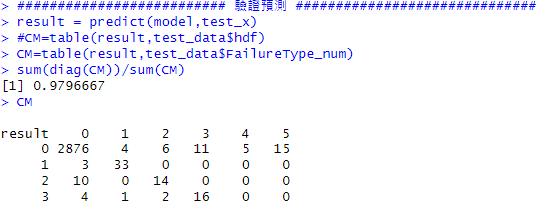
4.1.1 隨機森林

取全部資料進行第一次分析後，發現預測能力達98%，但是沒有成功採樣到Random Failures和 Tool wear Failure的數值，且Overstrain Failure的Type II error也高達39.13%，結果不甚理想，整體的Type II error為25%。



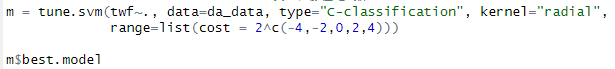
4.1.2 XGBOOST

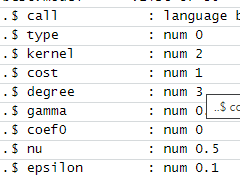
取全部資料進行第一次分析後，發現預測能力達97.97%，但是Overstrain Failure的Type II error Failure高達41.67%，整體的Type II error為24.10%且沒有成功採樣到Random Failures和 Tool wear Failure的數值。



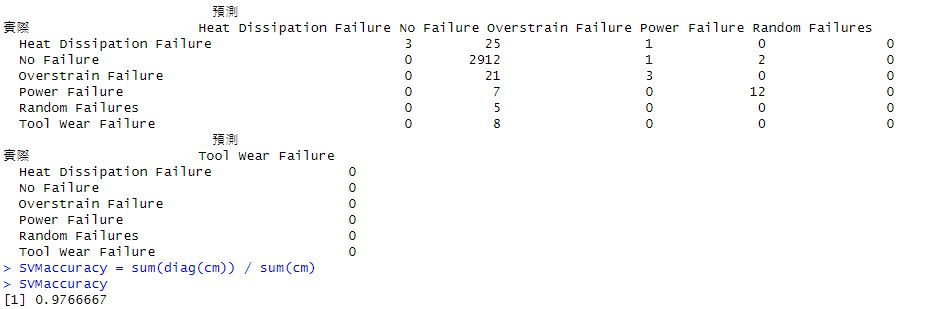
4.1.3 SVM

先找出最佳參數，再取全部資料進行第一次預測，發現預測能力達97.67%，但是除了NO Failure外，各項錯誤類型的Type II Error都很大，整體的Type II error為38.05%。



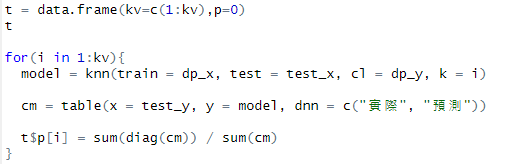


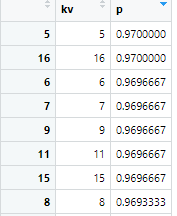
套模後得出cost=1



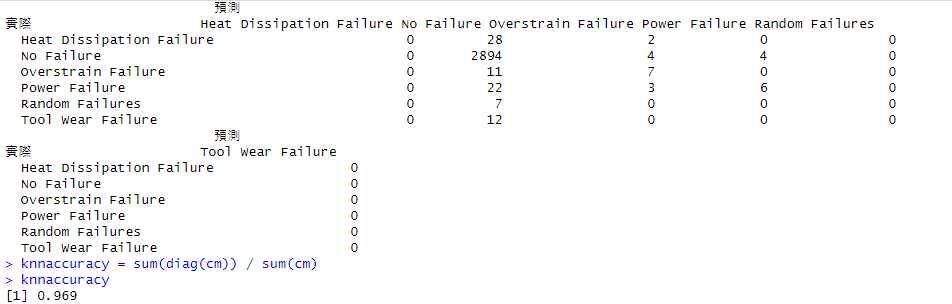
4.1.4 KNN

先找出最佳K值(方法如下)，發現K=5或16時預測率最佳





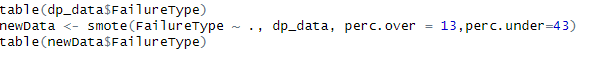
取全部資料進行第一次分析後，發現預測能力達96.9%，但是除了NO Failure外，各項錯誤類型的Type II Error都很高，整體的Type II error為68.37%。

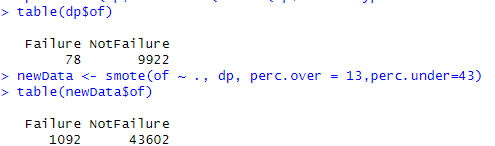


4.2 進行資料不平衡處理

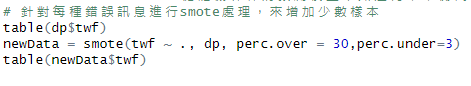
根據全分類預測的結果我們發現可能存在資料不平衡的問題，因此我們使用smote來解決此問題並再做一次預測。

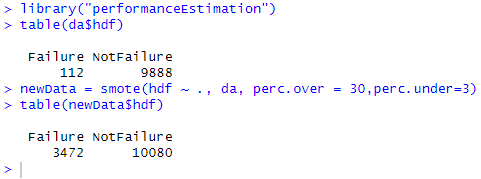
設定隨機森林的smote參數 : 經過我們多方嘗試後，依據失敗和成功比例1:40所得出來的結果





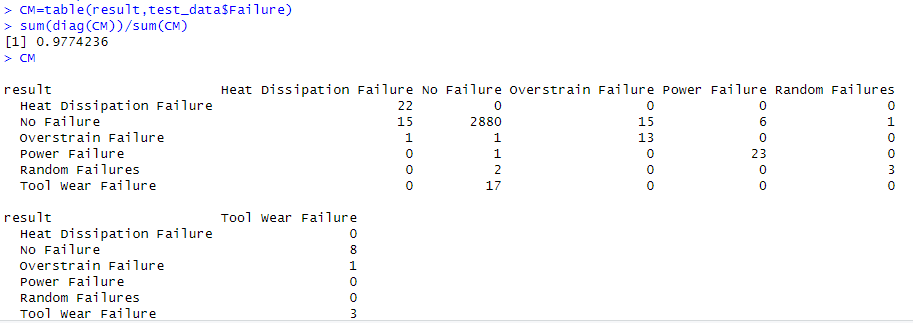
設定XGBOOST、SVM、KNN的smote參數 : 經過我們多方嘗試後，依據失敗和成功比例1:3所得出來的結果





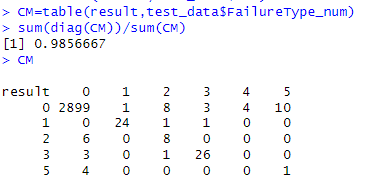
4.2.1 隨機森林

執行完smote後我們發現預測力微幅下降至97.77%，但是Random Failures的Type II Error高達100%，Tool Wear Failure還是採樣不到不出數值，Overstrain Failure的Type II Error高達30%，這樣的結果並不理想，整體的Type II error為27.94%。



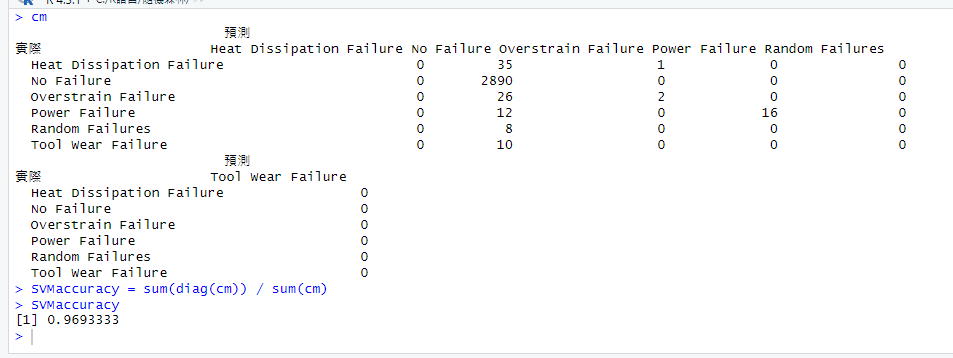
4.2.2 XGBOOST

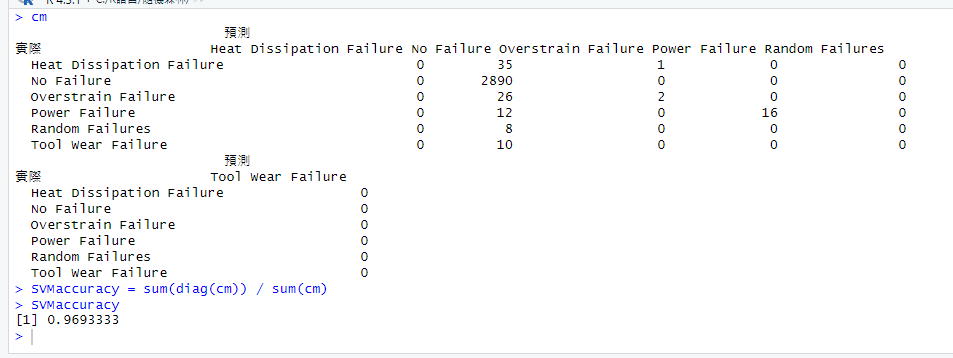
執行完smote後我們發現雖然預測能力達98.57%，但是Overstrain Failure的Type II error Failure高達42.86%，Tool Wear Failure的Type II Error高達80%，另外因為資料太過不平衡的關係，經過smote處理後，依舊採樣不到Random Failures的數值，整體的Type II error為26.67%。



4.2.3 SVM

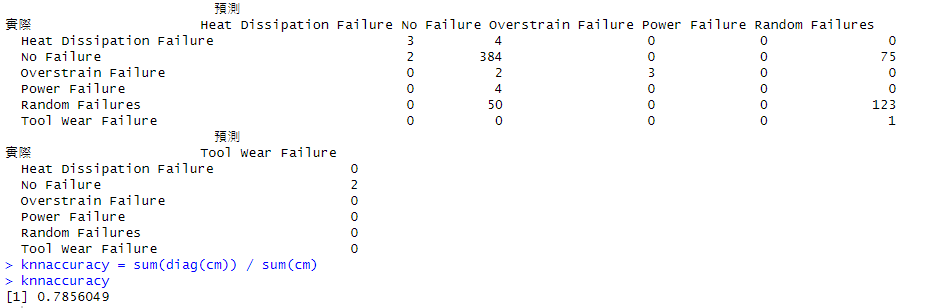
執行完smote後我們發現雖然預測能力達96.93%，但是除了NO Failure外，Type II Error都很大，這樣的結果不盡理想，整體的Type II error為30%。





4.2.4 KNN

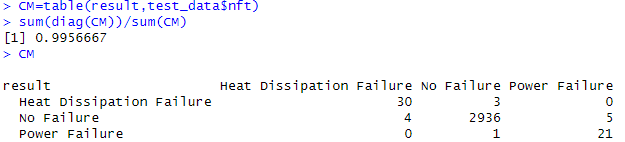
執行完smote後我們發現雖然預測能力下降至78.56%，並且除了NO Failure外，Type II Error都還是很大，這樣的結果依舊不理想，整體的Type II error為32.11%。



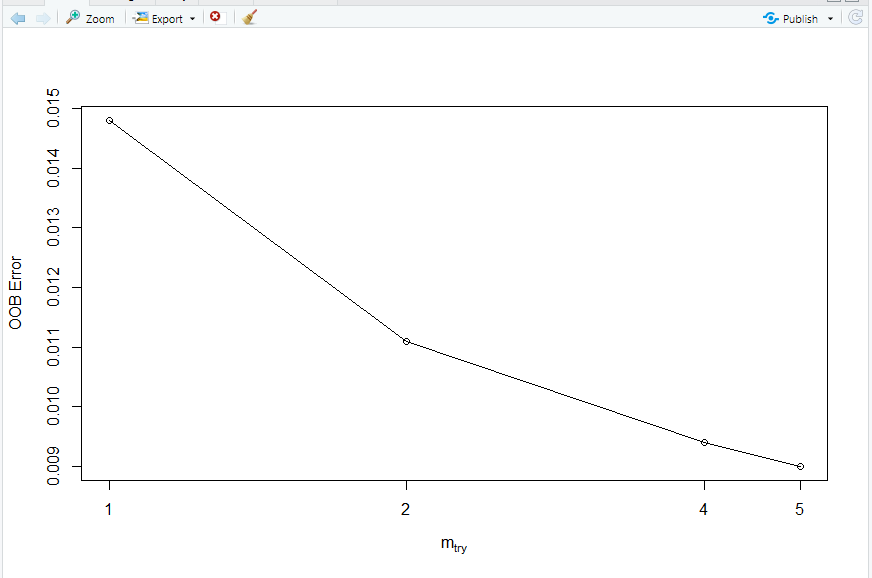
4.3將特定欄位單獨進行預測

執行完smote後我們發現針對不同的演算法，有些欄位的預測錯誤率(Type II error)很高，因此我們決定針對特定錯誤分類進行學習及預測。

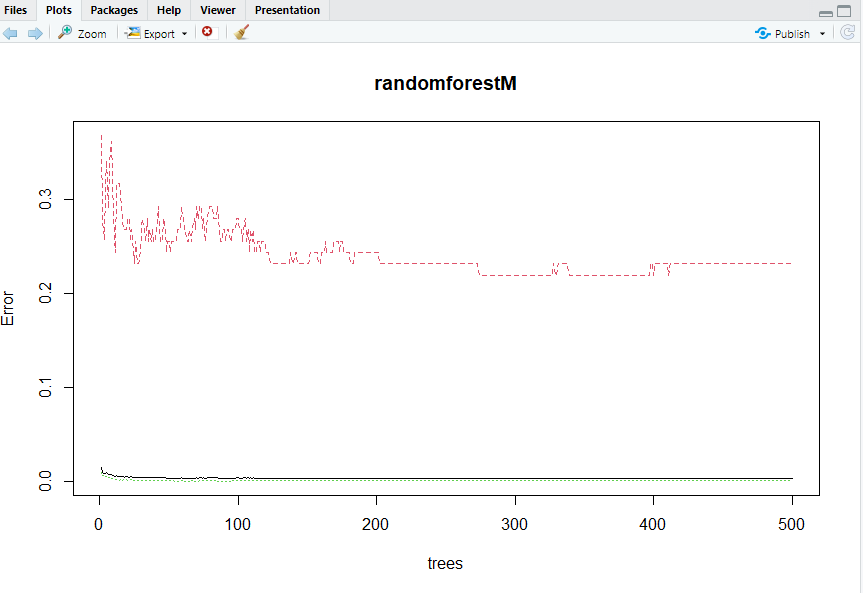
4.3.1 隨機森林—針對Power Failure和Heat Dissipation Failure預測

預測時只針對Power Failure和Heat Dissipation Failure這兩個錯誤類別預測並挑選適當的mtry個數(5)以及樹的數量(200棵)進行預測，發現預測率上升至99.57%，整體的Type II error僅7.27%，結果相當優異。 

當OOB Error最低時，mtry即為最佳



當樹的數量超過200時Error趨於平緩，因此我們挑選200棵樹來進行訓練

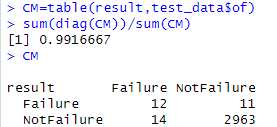


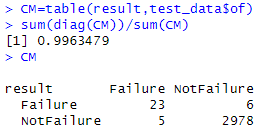
4.3.1.2 隨機森林—單獨針對Overstrain Failure、Tool Wear Failure、Random Failures進行預測

根據先前的預測結果，我們決定單獨針對Overstrain Failure、Tool Wear Failure、Random Failures這三個預測較差的錯誤類型進行單獨預測。

4.3.1.3單獨針對Overstrain Failure進行預測

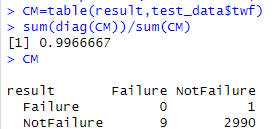
有99.17%的預測率，且整體的Type II error為20.69%，顯示單獨針對Overstrain Failure進行預測的效果還可以。



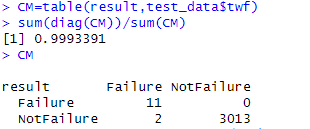


4.3.1.4單獨針對Tool Wear Failure進行預測

第一次預測時發現出現資料不平衡的問題，因此做了一次smote後再預測一次。

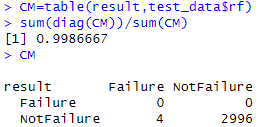


第二次預測結果發現有99.93%的預測率，且Type II error為0%，顯示單獨針對Tool Wear Failure進行預測的效果相當好。

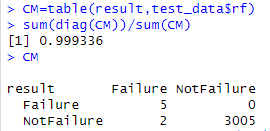


4.3.1.5單獨針對Random Failures進行預測

第一次預測時發現出現資料不平衡的問題，因此做了一次smote後再預測一次。



第二次預測結果發現有99.93%的預測率，且Type II error為0%，顯示單獨針對Random Failures進行預測的效果相當好。

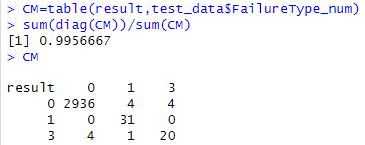


4.3.2 XGBOOST

執行完smote後的結果不慎理想，因此我們打算將各個錯誤類型分開預測看看，並分析哪個自變數影響最甚。

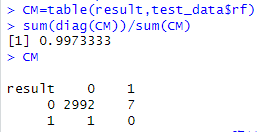
4.3.2.1 針對Power Failure和Heat Dissipation Failure預測

預測時只針對Power Failure和Heat Dissipation Failure這兩個錯誤類別預測，結果發現預測率高達99.57%且不論是哪一個類型的Failure，Type II error都很小，整體的Type II error大幅下降至8.93%。

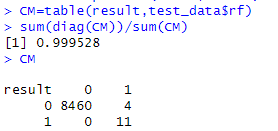


4.3.2.2 單獨針對Random Failures進行預測並做自變數調整

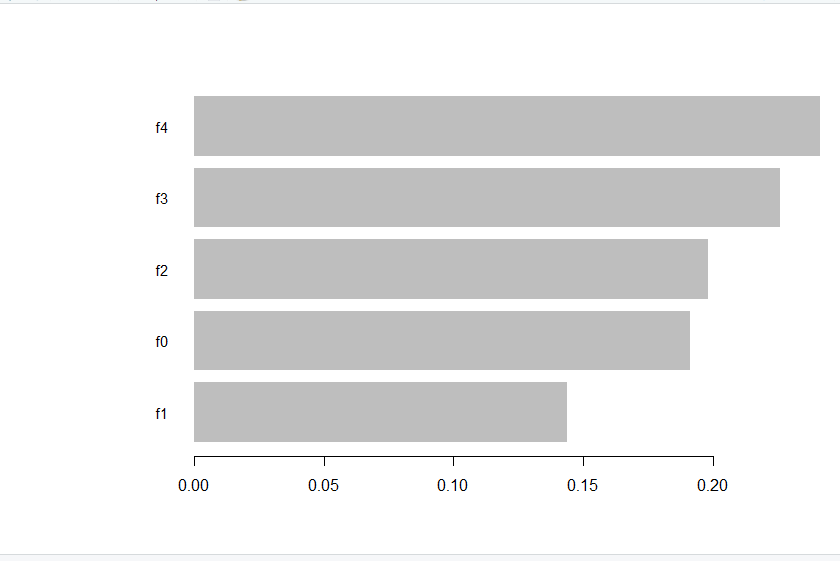
第一次預測時發現存在資料不平衡的問題，因此做了一次smote後再預測一次。

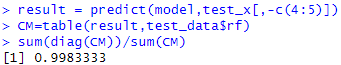


第二次預測結果發現雖然有99.95%的預測率，但是Type II error卻高達100%，顯示單獨針對Random Failures進行預測的效果不好。



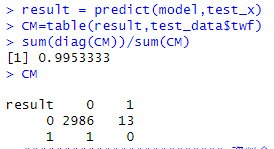
接著我們使用XGB內建的功能進行自變數調整，觀察哪些變數比較有用，並再做一次預測，結果發現f4(ToolWear\_min)影響結果最甚，但5個自變數的占比相當接近，而預測率為99.83%。



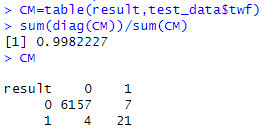


4.3.2.3 單獨針對Tool Wear Failure進行預測並做自變數調整

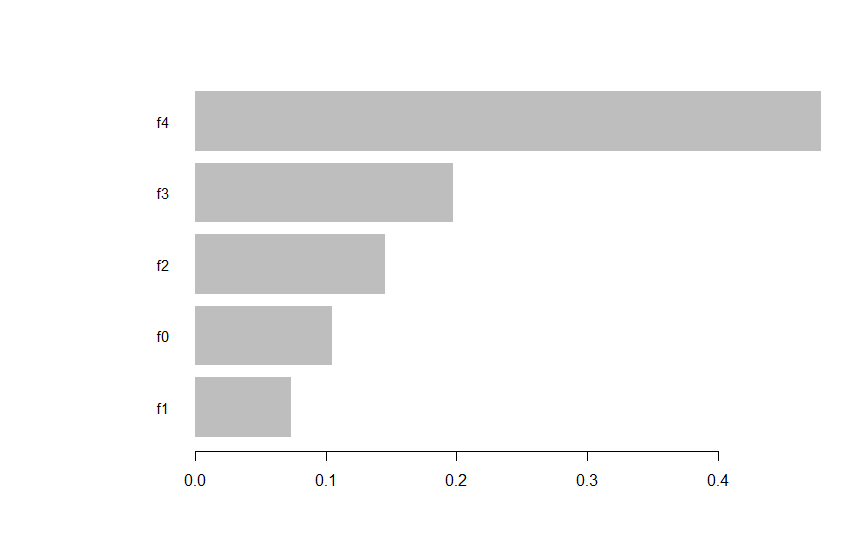
第一次預測時發現存在資料不平衡的問題，因此做了一次smote後再預測一次。



第二次預測結果發現有99.82%的預測率，且Type II error僅16%，顯示單獨針對Random Failures進行預測的效果還不錯。



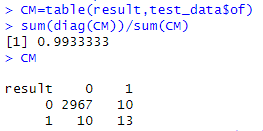
接著我們使用XGB內建的功能進行自變數調整，觀察哪些變數比較有用，並再做一次預測，結果發現f4(ToolWear\_min)影響結果最甚，預測率上升至99.57%。



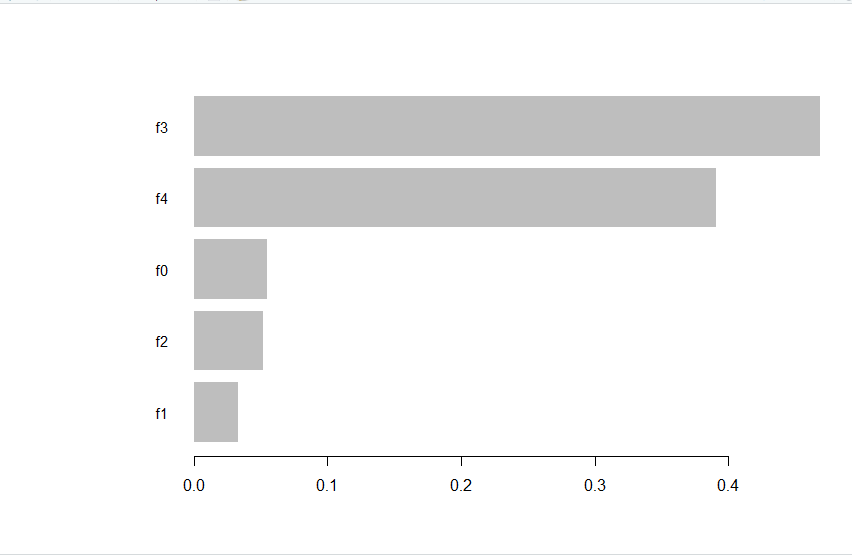


4.3.2.3 單獨針對Overstrain Failure進行預測並做自變數調整

預測結果發現有99.33%的預測率，但是Type II error為43.48%，顯示單獨針對Overstrain Failure進行預測的效果不夠理想。



接著我們使用XGB內建的功能進行自變數調整，觀察哪些變數比較有用，並再做一次預測，結果發現f3(Torque\_Nm)和 f4(ToolWear\_min)影響結果最甚，而預測率微幅下降到99.23%。



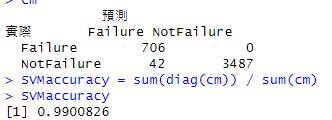


4.3.3 SVM

處理完資料不平衡的問題後，SVM的預測結果依舊不理想，因此我們認為SVM一次處理類別較多時的效果不佳，我們決定要就單一個類別的錯誤一個一個分項去做預測。

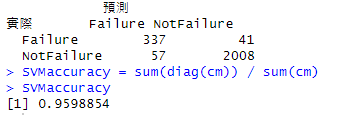
4.3.3.1單獨針對Overstrain Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達99%的預測率，顯示單獨針對Overstrain Failure進行預測是好的。



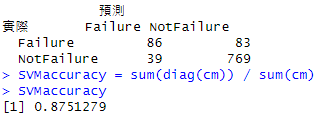
4.3.3.2單獨針對Tool Wear Failure進行預測

不論是Type I error或是Type II error都很低，並且有96%的預測率，顯示單獨針對Tool Wear Failure進行預測是好的。



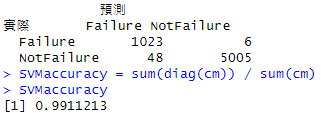
4.3.3.3單獨針對Random Failures進行預測

雖然Type I error很低，但Type II error將近50%，顯示單獨針對Random Failure進行預測不夠理想。



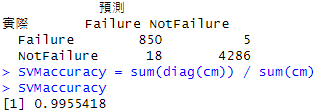
4.3.3.4單獨針對Heat Dissipation Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達99.1%的預測率，顯示單獨針對Heat Dissipation Failure進行預測是好的。



4.3.3.5單獨針對Power Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達99.55%的預測率，顯示單獨針對Power Failure進行預測是好的。

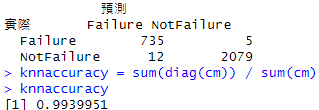


4.3.4 KNN

處理完資料不平衡的問題後，KNN的預測結果依舊不理想，因此我們認為KNN一次處理類別較多時的效果不佳，我們決定要就單一個類別的錯誤一個一個分項去做預測。

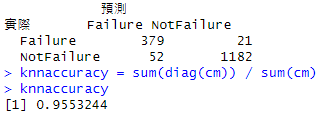
4.3.4.1單獨針對Overstrain Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達99.4%的預測率，顯示單獨針對Overstrain Failure進行預測是好的。



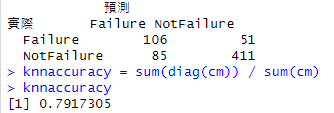
4.3.4.2單獨針對Tool Wear Failure進行預測

不論是Type I error或是Type II error都很低，並且有95.53%的預測率，顯示單獨針對Tool Wear Failure進行預測是好的。



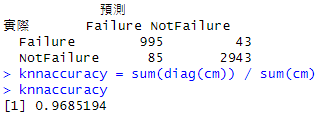
4.3.4.3單獨針對Random Failures進行預測

雖然Type I error不高，但Type II error 達32.4%，且整體預測率只有79.17%，顯示單獨針對Random Failure進行預測不夠理想。



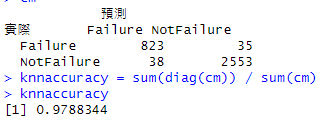
4.3.4.4單獨針對Heat Dissipation Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達96.85%的預測率，顯示單獨針對Heat Dissipation Failure進行預測是好的。



4.3.4.5單獨針對Power Failure進行預測

不論是Type I error或是Type II error都很低，並且有高達97.88%的預測率，顯示單獨針對Power Failure進行預測是好的。



第五章 結果與討論

5.1結果呈現

以下數據為預測三次後取最佳結果的數值

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 最佳預測率(%) | | | | | | | |
| 全分類 | 全分類+ smote | 不含Random  Failure 和 Tool Wear  Failure和Overstrain Failure | Tool Wear  Failure | Power  Failure | Overstrain Failure | Random  Failures | Heat Dissipation Failure |
| 隨機森林 | 98.00 | 97.97 | 99.57 | 99.93 |  | 99.17 | 99.93 |  |
| XGBOOST | 97.97 | 98.57 | 99.57 | 99.82 |  | 99.33 | 99.95 |  |
| SVM | 97.67 | 96.93 |  | 96.00 | 99.55 | 99.00 | 87.51 | 99.11 |
| KNN | 96.90 | 78.56 |  | 95.53 | 97.88 | 99.40 | 79.17 | 96.85 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Type I error(%) | | | | | | | |
| 全分類(含smote後結果) | 全分類+ smote | 不含Random  Failure 和 Tool Wear  Failure和Overstrain Failure | Tool Wear  Failure | Power  Failure | Overstrain Failure | Random  Failures | Heat Dissipation Failure |
| 隨機森林 | 1.43 | 1.54 | 0.31 | 0.07 |  | 0.47 | 0.07 |  |
| XGBOOST | 1.41 | 0.89 | 0.27 | 0.11 |  | 0.34 | 0.05 |  |
| SVM | 0.10 | 0.00 |  | 2.76 | 0.42 | 1.19 | 4.83 | 0.95 |
| KNN | 0.28 | 1.71 |  | 4.21 | 1.47 | 0.57 | 17.14 | 2.81 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Type II error(%) | | | | | | | |
| 全分類(含smote後結果) | 全分類+ smote | 不含Random  Failure 和 Tool Wear  Failure和Overstrain Failure | Tool Wear  Failure | Power  Failure | Overstrain Failure | Random  Failures | Heat Dissipation Failure |
| 隨機森林 | 25 | 27.94 | 7.27 | 0 |  | 20.69 | 0 |  |
| XGBOOST | 24.1 | 26.67 | 8.93 | 16 |  | 43.48 | 0 |  |
| SVM | 78.82 | 30 |  | 10.85 | 0.58 | 0 | 49.11 | 0.58 |
| KNN | 68.37 | 32.11 |  | 5.25 | 4.08 | 0.68 | 32.48 | 4.14 |

5.2結果分析

透過以上三份表格可以得出以下幾個結論

1. 不論使用哪種演算法，全分類預測此筆資料集的效果都不好，Type II error都很大。

2. 使用隨機森林和XGBOOST演算法並將Random Failure 和 Tool Wear Failure和 Overstrain Failure這三個預測能力不佳的分類剔除後進行預測的效果都不錯，而隨機森林的Type II error又再更低一些。

3. 使用隨機森林演算法單獨針對Tool Wear Failure來進行預測是效果最好的，有最好的預測率和最低的Type I error、 Type II error，其餘三種演算法都有Type I error或 Type II error較高的情形發生。

4. 使用SVM演算法單獨針對Power Failure來進行預測是效果比KNN更好，有最好的預測率和最低的Type I error、 Type II error，但兩者整體而言都不差。

5. 使用SVM和KNN演算法單獨針對Overstrain Failure來進行預測是效果比隨機森林和XGBOOST更好，在預測率相當接近的前提下，後者的Type II error高上許多。

6. 使用隨機森林和XGBOOST演算法單獨針對Random Failures來進行預測的效果比SVM和KNN更好，不論是預測率還是Type I error、Type II error都好上許多。

7. 使用SVM演算法單獨針對Heat Dissipation Failure

來進行預測是效果比KNN更好，有較好的預測率和較低的Type I error、 Type II error，但兩者整體而言都不差。

8. 沒有最好的演算法，只有相對較好的演算法，該採用哪種演算法去預測還是要針對公司需求以及多次的分析預測才能確定。

9. 造成以上的預測結果我們認為可能的原因為:

9.1在預測不同的錯誤類別時有適合該預測類別的演算法，沒有固定。

9.2 原始資料有明顯資料不平衡的問題，可能對於結果有些許影響。

9.3 有些預測較差的項目可能是因為該演算法不適合此錯誤類別，更可能是因為問題是出在自變數而非預測模型本身。

5.3 未來展望

這次的專題我們運用了機器學習裡的四種演算法來完成，並且比較了不同錯誤類型的預測率，礙於時間的關係這次只有探討應變數本身的預測能力以及使用XGBOOST演算法來探討自變數與應變數的相關性，在未來希望可以用其他演算法並且從自變數下手，探討其和最終預測結果的關聯性。

第六章 資料來源

R語言介紹

<https://zh.wikipedia.org/zh-tw/R%E8%AF%AD%E8%A8%80>

XGBOOST運作方式<https://docs.aws.amazon.com/zh_tw/sagemaker/latest/dg/xgboost-HowItWorks.html>

XGBOOST介紹

<https://ithelp.ithome.com.tw/articles/10276808>

隨機森林介紹

<https://ithelp.ithome.com.tw/articles/10303882>

XGBOOST套件

<https://ithelp.ithome.com.tw/m/articles/10300662>

SVM解說

<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>

KNN解說

<https://ithelp.ithome.com.tw/m/articles/10269826>