



國立高雄應用科技大學  
資訊工程系碩士在職專班  
碩士論文

生產力 4.0 機台異常分析

Productivity 4.0 Analysis of Machine  
abnormality

研究生：李欣

指導教授：林威成 教授

中華民國 106 年 6 月

# 生產力 4.0 機台異常分析

Productivity 4.0 Analysis of Machine abnormality

從實驗架設以及機台異常數據分析的研究  
-使用 R 語言作為分析的工具

Research on the Analysis of Abnormal Data from Experiment Erection and  
Machine  
- Use R language as a tool for analysis

研究生：李欣

指導教授：林威成 教授



A Thesis Submitted to  
college of electrical engineering and computer science  
National Kaohsiung University of Applied Sciences  
in Partial Fulfillment of the Requirements  
for the Degree  
of Master of Engineering in Department of  
Computer Science and Information Engineering

June 2017  
Kaohsiung, Taiwan, Republic of China

中華民國 106 年 6 月

國立高雄應用科技大學  
進修推廣處碩士在職專班學位論文考試審定書

本校

資訊工程系 碩士班

研究生

李欣

所提之論文

生產力 4.0 機台異常分析

合於 碩士 資格水準，業經本委員會評審認可。

學位考試委員會

召集人

林蕙佳

簽章

委員

林蕙佳

陳淑瑾

王麗玲

林海鈞

指導教授

林海鈞

簽章

系所主管

林海鈞

簽章

中華民國

106 年 6 月 22 日

# Productivity 4.0 Analysis of Machine abnormality

by  
HSIN LI

A Thesis Submitted to  
college of electrical engineering and computer science  
National Kaohsiung University of Applied Sciences  
in Partial Fulfillment of the Requirements  
for the Degree  
of Master of Engineering in Department of  
Computer Science and Information Engineering  
Jun 22, 2017

Approved by :

Wei-Ti Lin

Ju-Chin Chen

Dong-Chuan Wang

Wen-Hy Lin

Thesis Advisor : Wen-Hy Lin

Institute Director : Wen-Hy Lin

# 生產力 4.0 機台異常分析

學生:李欣

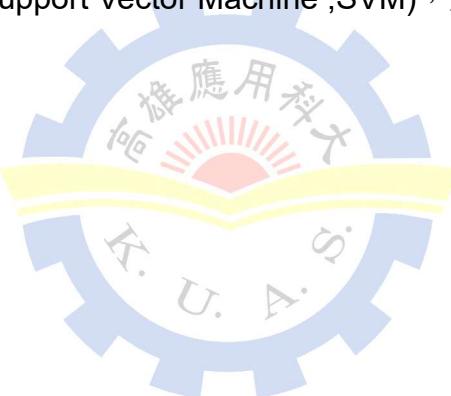
指導教授:林威成

國立高雄應用科技大學資訊工程系碩士在職專班

## 摘要

本研究擷取真實機台資料，長期蒐集並且儲存歷年資訊，平均產出約 10 萬筆/每天，因資料太龐大，若無整理對資訊系統來說是個巨大的負擔，且不切實際。本研究旨在探討如何以大數據分析方式，及資料的視覺化處理，來分析如資料與異常的關聯性，方便製程工程師或品保工程師做現場判斷。

內容包含如何運用 CentOS 7 建置 Hadoop，SPARK 環境，並安裝 Rstudio server 平台，並且使用 SparkR 裡的機器學習演算法來進行預測，使用的演算法為：支援向量機器(Support Vector Machine ,SVM)，將原始數據分類並塑造模型，預測未來資料。



關鍵字:大數據分析、異常分析、R 語言、SPARK、HADOOP、機器學習

# **Productivity 4.0 Analysis of Machine abnormality**

Student: HSIN LI

Advisors:Dr. Lin,Wei-Cheng

Institute Of Computer Science And Information Engineering

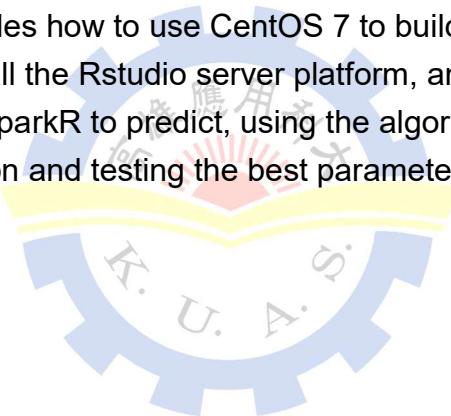
National Kaohsiung University of Applied Sciences

## **ABSTRACT**

This study captures real machine information, collects and stores information over a long period of time, with an average output of about 100,000 pieces per day. Because of the huge amount of information, it is unrealistic if there is no sort of information system.

The purpose of this study is to explore how to analyze the data and abnormalities in the way of large data analysis and visualization of data, so that the process engineers or quality assurance engineers can make the field judgment.

The content includes how to use CentOS 7 to build a Hadoop, SPARK environment, and install the Rstudio server platform, and use the machine learning algorithm in SparkR to predict, using the algorithm for the decision tree binary classification and testing the best parameter feedback At the end of the project.



Keywords: big data analysis, anomaly analysis, R language, SPARK, HADOOP, machine learning

## 誌謝

感謝指導教授林威成老師兩年來的指導與裁，在從事工作之餘，可以多多學習並拓展視野，此外還跟我們碩專班同學分享親身經驗及經歷，課程外還有業師來現身指導，對於我們的工作技能有正面的幫助。

此外也感謝公司能夠提供資料，讓我得以用這些數據資料完成論文實驗，另外也感謝碩專班同學，榮聰、俊麟，協助我使得此論文可以完成。



# 目錄

## 內容

摘要.....	I
誌謝.....	III
目錄.....	IV
表目錄 .....	VI
圖目錄 .....	VII
第一章 緒論 .....	1
前言 .....	1
1.1 研究背景:.....	1
1.2 研究動機:.....	1
1.3 研究架構:.....	2
第二章 文獻探討.....	3
2.1 Hadoop & Spark & Rstudio .....	3
2.1.1 Hadoop.....	4
2.1.2 Spark .....	5
2.1.3 Rstudio .....	6
2.2 異常診斷相關文獻 .....	6
2.3 大數據、機器學習 .....	7
2.3.1 大數據(BigData) .....	7
2.3.2 機器學習(Machine Learning) .....	7
2.3.3 機器學習類型 .....	8
2.3.4 監督式學習演算法介紹: .....	9
2.4 機器學習模型分類模型評估 .....	10
2.4.1 ROC 曲線 .....	11
2.5 資料檢視、資料視覺化、資料清理 .....	12
2.5.1 資料檢視 .....	12
2.5.2 資料視覺化 .....	14
2.5.3 資料清理(Data Cleaning) .....	14
2.6 公司簡介與產生此實驗數據的機台 .....	16
第三章 實驗環境架設 .....	18
3.1 安裝 CentOS 7 要點 .....	18
3.2 安裝 Hadoop .....	19
3.3 設定 Multi Node Cluster.....	20
3.4 安裝 Spark.....	21
3.5 安裝 Rstudio.....	21

第四章	實驗數據分析 .....	22
4.1	檢視原始數據 .....	22
4.2	原始資料切割 .....	23
4.3	切割後的資料視覺化.....	25
4.4	資料彙整.....	29
4.4.1	資料解釋.....	29
4.4.2	資料整理 .....	30
4.5	機器學習與 SVM 建模.....	34
4.5.1	匯入資料 .....	34
4.5.2	使用 SVM 建立預測模型 .....	34
4.5.3	計算預測準確度 .....	35
4.5.4	匯入測試資料 .....	36
4.5.5	計算測試資料的預測準確度 .....	36
4.5.6	刪除訓練資料的最後 3 欄位，並建模 .....	38
4.5.7	刪除測試資料的最後三欄，並計算模型準確度 .....	39
4.5.8	大數據模型建立，並刪除 A 槽所有資料 .....	40
4.5.9	匯入小筆數的測試資料，並計算模型準確度 .....	41
4.5.10	匯入建立模型(4/29)後，機台生產的實際資料 .....	42
4.5.11	SVM 的分佈圖 .....	44
4.5.12	小結 .....	45
第五章	結論 .....	46

## 表目錄

表 1 SPARK 特色 .....	5
表 2 R 語言的特點 .....	6
表 3 混亂矩陣 .....	10
表 4 AUC 判斷表.....	12



## 圖 目 錄

圖 1 研究架構圖 .....	2
圖 2 HADOOP & SPARK & SPARKR 架構圖 .....	3
圖 3 HADOOP 架構圖 .....	4
圖 4 機器學習架構圖 .....	8
圖 5 決策樹示意圖 .....	9
圖 6 驚尾花分布示意圖 .....	10
圖 7 ROC 曲線 .....	12
圖 8 資料檢視 .....	13
圖 9 蝕刻 .....	17
圖 10 MULTI NODE CLUSTER .....	18
圖 11 CENTOS 7 .....	18
圖 12 JAVA JDK .....	19
圖 13 HADOOP 狀態 .....	20
圖 14 RSTUDIO 開發工具 .....	22
圖 15 原始資料 .....	22
圖 16 原始資料切割程式 .....	23
圖 17 原始資料切割後 .....	23
圖 18 資料視覺化(直方圖) .....	25
圖 19 資料視覺化(直方圖) .....	25
圖 20 資料視覺化(直方圖) .....	26
圖 21 資料視覺化(折線圖) .....	28
圖 22 資料視覺化(折線圖) .....	28
圖 23 資料視覺化(折線圖) .....	28
圖 24 更改後的 LABEL 欄位 .....	29
圖 25 切割過後的原始資料 .....	29
圖 26 第一次蝕刻資料 .....	30
圖 27 非特徵資料 .....	30
圖 28 刪除非特徵欄位 .....	30
圖 29 刪除冗餘欄位 .....	31
圖 30 刪除 K 值後 .....	31
圖 31 去除 ADJ 欄位 .....	31
圖 32 JUDGE 欄位篩選 .....	32
圖 33 資料產生流程 .....	32
圖 34 更改後的 JUDGE 欄位 .....	33
圖 35 上傳檔案至 RSTUDIO-SERVER .....	34
圖 36 MOD1 的結果 .....	35

圖 37 MODEL_RESULT 的結果 .....	35
圖 38 混亂矩陣 .....	35
圖 39 計算後的數值 .....	36
圖 40 測試資料的統計狀況 .....	36
圖 41 測試資料的統計狀況 .....	37
圖 42 測試資料的統計狀況 .....	37
圖 43 訓練資料的統計狀況 .....	38
圖 44 測試資料的統計狀況 .....	39
圖 45 39 萬筆訓練資料的原始檔 .....	40
圖 46 39 萬筆訓練資料的統計 .....	40
圖 47 預測準確度 .....	41
圖 48 39 萬筆訓練資料的統計 .....	41
圖 49 69888 筆資料的預測情形 .....	42
圖 50 69888 筆資料的預測情形 .....	42
圖 51 92 個檔案共 70656 筆資料 .....	43
圖 52 真實資料良率 .....	43
圖 53 計算後的資料 .....	43
圖 54 資料散佈圖 .....	44
圖 55 資料散佈圖 .....	44



# 第一章 緒論

## 前言

生產力 4.0 為，台灣經濟部工業發展局提出的看法，由台灣行政院於 2015 年時推行，主要是欲藉由物聯網、資料分析運算、高速網路、IT 基礎建設，期望能將台灣的科技、傳統產業升級、農業發展能自動化、科技化，同時提高附加價值，並在高齡化社會中降低企業對於勞動人口的勞動需求，但因此專案涉及到的部分很多，因此在高應大資工系林威成教授的引導下，將此專案切割成為 3 個部份，分別為 1. 生產力 4.0-基礎建設，2. 生產力 4.0-資料收集器，3. 生產力 4.0-機台異常報告，而此篇論文為生產力 4.0-機台異常報告，會透過前述的論文為架構發展的資料分析系統，用來分析傳統產業機台產生的原始資料，並透過原始資料的建模與預測，能夠提前了解是否為機台異常，使其能夠先行排程及維護機台，節省企業的損失。

### 1.1 研究背景：

台灣的製造工業，在近幾年來的嚴格控管成本之下，機台的正常運作與不良品的生成原因，是企業永遠的戰爭。隨著資訊科技的不斷進步與發展、以及中國的紅色供應鏈崛起，越來越多的低價搶市，造成了越來越低的利潤，更使經理人更加嚴格控管成本，縮小研發實力，最後導致企業面臨生存危機。

因此在產品生產及製造過程中，若能快速的讓工程人員排除事故，找出事件的相互相關聯性，不但能提升整體良率，且使機台產出更有效率，更能降低維修人員的負擔，以及維修成本。

### 1.2 研究動機：

在這物聯網爆炸性成長的時代，傳統工廠的生產方式也面臨新的挑戰，越來越多的 PLC 裝置、機器視覺、微感應器…等，經由資料蒐集以及儲存技術的精進，大大的加速了資料堆積，而對於這些成堆的資料，越來越多企業選擇從中分析有價值的資訊或資料關聯性，進而提供決策者察覺商業趨勢、判斷研究品質，以提升企業競爭力。

而中小企業在處理這些雜亂的資訊時會受限於硬體或軟體的空間或效能不足導致無法有效的處理巨量資料，因此大數據分析開源技術平台 (Hadoop/Spark/Rstudio)，可以協助中小企業分析這些數據，並且不至於花費太高的成本。

機台在生產時會依製程工程師的條件設置下去製作，而在機台發生某些異常時，會出現某些特徵參數的變化，然而在近年來的機台能力越來越強，特徵參數數量也變多了，但真正具有關鍵影響的參數卻可能只存在少數幾個特徵參數上，且各種異常也會有不同的關鍵參數造成影響，因此將這些異常的關鍵因子找出，用來做判斷，用來減少不良品的產出，用以提升機台良率。

因此如何使用 Hadoop/Spark/Rstudio 找出產品異常的關鍵影響參數，以提升良率為本研究的動機。

### 1.3 研究架構:

本研究分主要分為五個章節。第一章為緒論，第二章節為文獻探討，探討參考各書籍的文獻及相關研究理論，並說明機器學習-監督式學習(Supervised Learning)-二元決策樹(Decision tree Binary)用途，讓決策者易於理解與解釋。第三章為實驗環境架設，實際的 CentOS + Hadoop + Spark + Rstudio-server 環境架設。第四章為研究方法與結果，放入真實數據並做模擬，並做出結果。第五章為結論。研究架構如圖 1 所示：

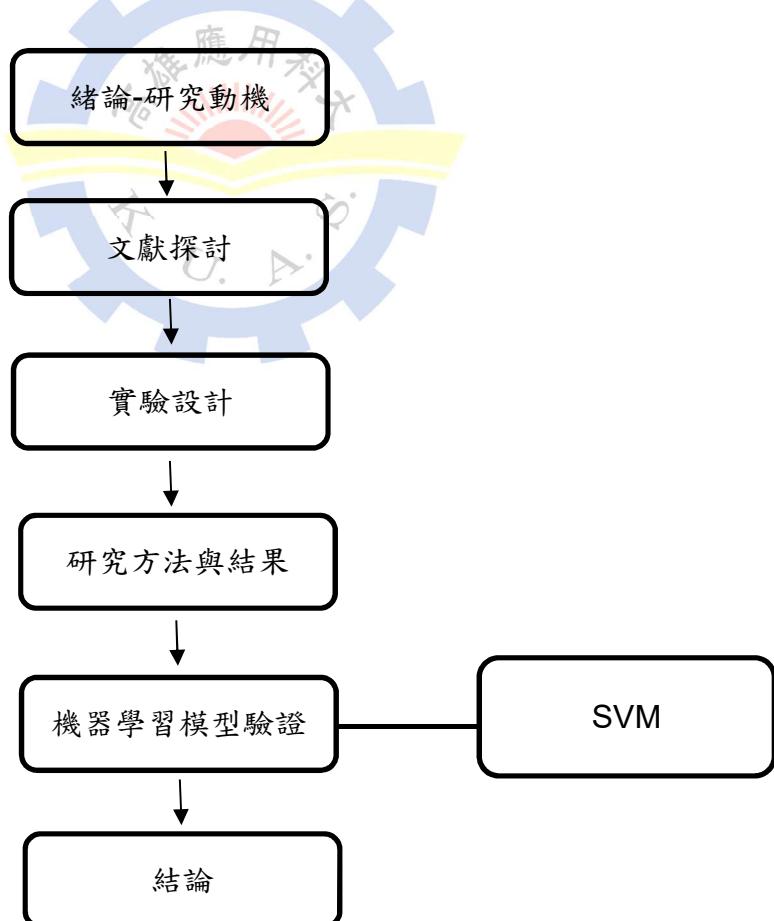


圖 1 研究架構圖

## 第二章 文獻探討

本章節介紹此論文中研究相關的軟體及機台功能與文獻探討

- 2.1 節介紹 Hadoop & Spark & Rstudio
- 2.2 節介紹異常診斷相關文獻
- 2.3 節介紹大數據、機器學習
- 2.4 節介紹決策樹模型分類模型評估
- 2.5 節介紹資料檢視、資料視覺化、資料清理
- 2.6 節公司簡介與產生此實驗數據的機台

### 2.1 Hadoop & Spark & Rstudio

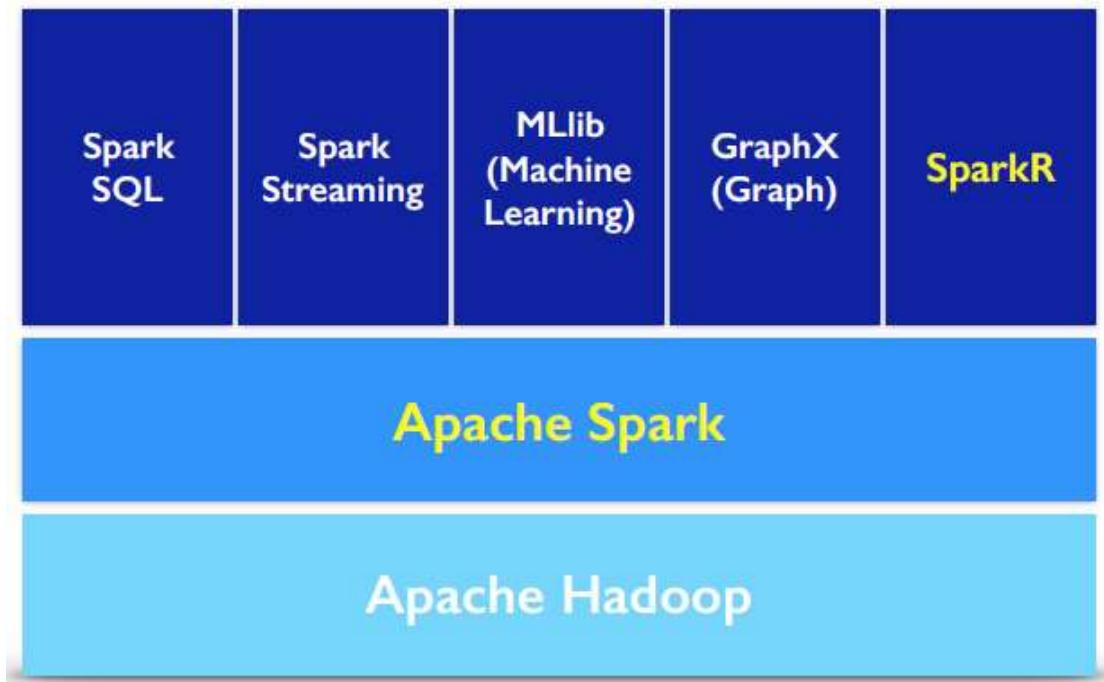


圖 2 Hadoop & Spark & SparkR 架構圖

## 2.1.1 Hadoop

Apache Hadoop 是用於巨量資料處理的開放原始碼平台，是 Google 公司發表了一篇關於 MapReduce 的論文去實作而成，Hadoop 1.0 最主要的核心架構為 HDFS(Hadoop Distributed File System)與 MapReduce。

HDFS 為分散式檔案系統，可以提供建置數台的機器作為叢集架構，用來儲存大量的檔案，且具有高度的容錯能力，但是主要設計是用來做批次處理，不是即時性的互動，優點是可以提高存取大量數據的能力，但卻犧牲了快速反應的時間。在 HDFS 檔案系統中，是由 NameNode 服務器管理 HDFS 目錄系統、並且控制檔案讀寫動作，而 DataNode 服務器儲存資料，在一個叢集架構中通常只會有一個 NameNode 來管理，但會有多組 DataNode 來儲存資料。

MapReduce 為分散式計算的技術，透過將大資料分割成數個小資料，分別交給各叢集節點運算，並回傳運算結果，使用這種平行運算方法來處理資料，可以大幅度的減少處理時間。但因 Hadoop MapReduce 在執行運算時，會將資料儲存於硬碟之中(即 HDFS)，而硬碟會有 I/O 效能的瓶頸。Hadoop 2.0 則增加了 YARN(Yet Another Resource Negotiator)作為 MapReduce 的新架構。

Hadoop 1.0 及 Hadoop 2.0 比較圖，如圖 3:

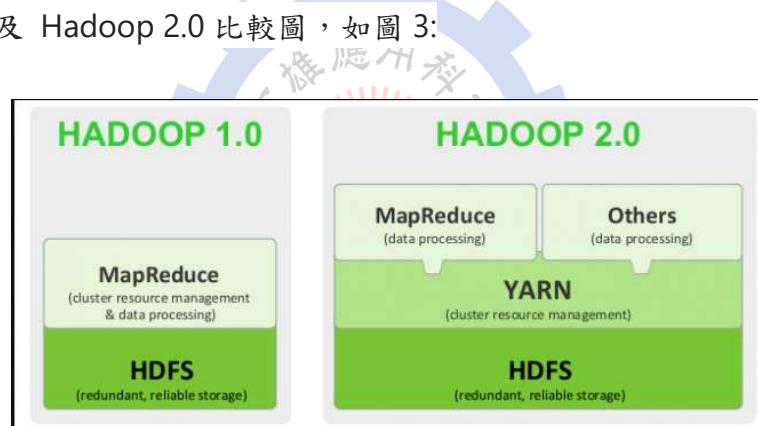


圖 3 Hadoop 架構圖

## 2.1.2 Spark

Apache Spark 是由 2009 年加州柏克萊分校實驗室(AMPLab)開發，Spark 的主要價值在於 RDD(Resilient Distributed Dataset)的運用，屬於一種分散式記憶體的概念，在其運算時會將產生的資料儲存在記憶體中，加快執行速度。Spark 與 Hadoop 最大的不同，在於前者使用記憶體做運算，後者使用硬碟做運算，因此在運算速度的比較上，Spark 速度增加約 100 倍，目前已發表至 Spark 2.1.0 版本(2016/12/28)。

Spark 特色如下表 1：

特色：	備註
速度快	Spark 是以記憶體計算的計算系統，因此在執行上速度及效益會大 Hadoop 很多。
用途廣泛	支援多語言 API，包含(Python、R、Java、Scala…等)，安裝也簡單容易。
通用	Spark 提供了強大的技術堆疊，可以在同一個程式中使用多種架構，使其在結構上一體化、功能上多元化。
整合 Hadoop	Spark 可以直接執行在 Hadoop 2.0 之上，並且完全支援 Hadoop YARN，因此無須安裝即可執行 Spark，且能讀取 HDFS、Hbase.... 等，以及任何 Hadoop 資料來源。
支援較多元	因相比於 Hadoop，有更簡潔的程式，安裝佈署也不需太複雜，因此造就非常多的工程師與實驗機構投入開發。

表 1 Spark 特色

### 2.1.3 Rstudio

RStudio 是一個用於 R 的集成開發環境 (IDE)，能在多數作業系統上執行 (如:windows、Mac、Linux)，Rstudio 易於使用且有完善的操作介面，以及出色的視覺化圖形、豐富的統計方法及高效率的更新，且擁有許多熱心的人致力開發多數套件，讓我們可以透過簡短的指令線上下載 API 並呼叫其功能，節省了許多重新編寫演算法的時間。RStudio 可以在開源和商業版本中使用，也可以支援架設於(Debian / Ubuntu，RedHat / CentOS 和 SUSE Linux)之中，而 RStudio Server(開源)或 RStudio Server Pro(商用版)，則是安裝於其上的服務，透過 Apache 伺服器的服務，可在合理的設置之下，可以透過任何的瀏覽器連接至此，並且使用其服務，在工廠環境裡可以易於與製程端討論實做目標。R 語言的特點如下表 2:

R 語言的特點
有效率的資料處理和儲存機制
R 語言是物件導向的統計程式語言
統計圖表可以直接簡單的畫出來
R 語言為自由軟體，不收取費用，但功能不比付費的統計軟體差
具有豐富的 Call out 功能，可簡單安裝在系統中使用
有非常多演算法及完整的資料分析工具

表 2 R 語言的特點

## 2.2 異常診斷相關文獻

吳佩馨(2011) 機台失效預警辨識模型建立-拋光機台為例，研究主要是利用支持向量機(Support Vector Machinces,SVM)、K-NN，利用屬性篩選，提升預測準確率，並在盡量不影響分類準確度的情況下，建立破片的失效預警辨識模型，用來使機台在異常發生前先做預防以及保養調整，以減少損失，進而提升機台良率。

謝惠如(2010) 拋光製程破片關鍵影響因子辨識模型，作者收集了會影響破片的 16 個關鍵特徵變數，並對其做邏輯回歸分析、MTS 方法，試圖找出影響破片的關鍵因子，最後篩選出 7 個真正影響破片的顯著因子，再利用樣本的測試資料對模型做評估比較。

吳旻穗(2013)退貨商品與製程關聯性分析，研究主要是蒐集 MES 系統與退貨資料，並針對其品質異常的部分運用 Apriori 演算法，建立預測模型，用來對後續退貨有異常的資料發生時能夠採取合適的對應方式。

## 2.3 大數據、機器學習

### 2.3.1 大數據(BigData)

巨量資料、海量資料。目前對於大數據定義可以歸類為下列 3 種(3V):

#### 1. 龐大的資料量(Volume)

機台生產時產出的資料、健康檢查時產生的相關數據、IOT 設備回傳的資料、或網管系統產生的網路封包資料....等。

#### 2. 變動的速度快(Velocity)

機台時刻的生產、IOT 設備蒐集資料、網管系統蒐集的資料...等。

#### 3. 資料多樣性(variety)

量測資訊、運轉速度、運轉秒數、電器特性、溫度、濕度...等。

滿足以上 3V 特性的數據，即可稱為大數據。在了解理解數據後，並作分析、篩選，找出資料與資料的相關連性，用來預測後續的趨勢、疾病...等。透過有條理判斷，而不是透過經驗法則及直覺來做決策。

### 2.3.2 機器學習(Machine Learning)

是透過已收集的歷史資料中分析訓練資料，並利用訓練完成後產生的模型，當未來有新的資料產生時，我們可以利用數據中的歷史關係和趨勢，用以預測未來，目前機器學習已廣泛的應用於 搜尋引擎(Google,yahoo)、醫學診斷、電腦視覺、語音辨識、推薦系統...等。

例如:Youtube 影音頻道就會透過使用者過去所觀看過的影片，找出使用者"應該會"喜歡的影片，放置於推薦影片之中。而在使用者觀看的影片類型、數量變多時，便會更加精準的預測出使用者的喜好，在 NetFlix 影音頻道更是大手筆的開出 100 萬美金的獎勵，讓能使使用者對影片喜好程度的預測準確率提升 10% 的人獲得。機器學習架構圖如圖 4 所示

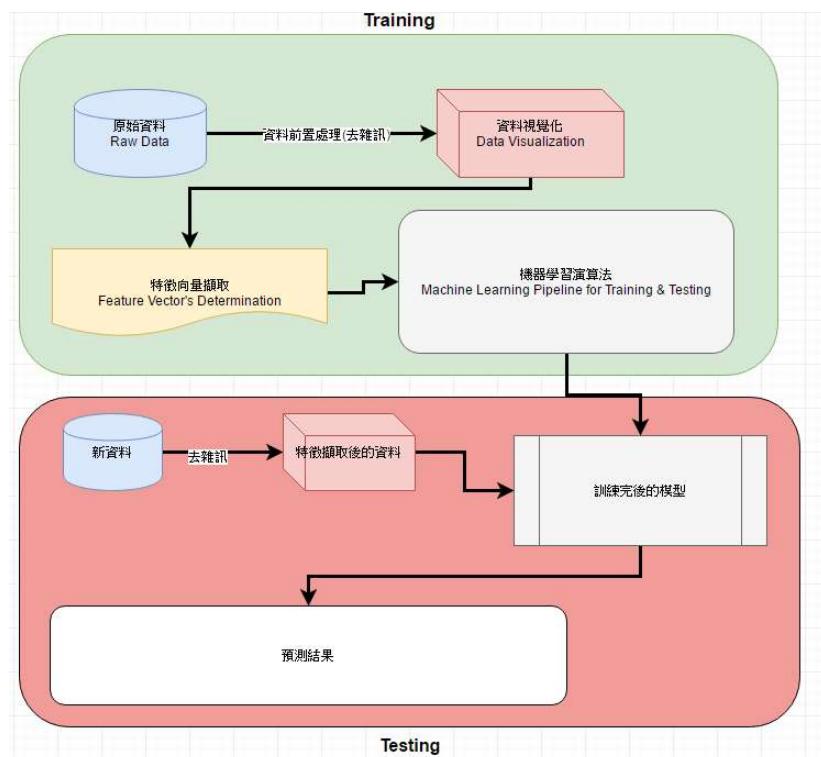


圖 4 機器學習架構圖

### 2.3.3 機器學習類型

機器學習又可約略分為兩種類別：

1. 監督式學習(Supervised Learning)  
(支援向量機(Support Vector Machine ,SVM)、決策樹(Decision tree)、迴歸分析(regression)...等)，數種不同的演算法，可依資料的結構或資料特性，選擇不同的演算法，最終的目的要透過學習到的模型，用來預測後續的趨勢或解答，並沒有特別一種演算法可以在於給定的資料中預測及判斷會最好，因此可以多方嘗試、調整參數並使用多種演算法組合成想要知道的資訊。
2. 非監督式學習(Unsupervised Learning)  
(聚類分析 (clustering analysis)、向度縮減 (dimensionality reduction))，為目前較為常用的演算法，因為在輸入資料時並無特別針對資料做標記，因此演算法會針對資料內部進行分類，嘗試將資料找出潛在的規則。兩者最大差別在於，監督式學習會從現有以標記的答案中，預測出答案，而非監督式學習則是在一堆複雜的數據中試圖分類，讓資料變的更簡潔。

### 2.3.4 監督式學習演算法介紹:

#### 1. 決策樹(Decision tree)

決策樹演算法屬於監督式學習法的一種，透過過去收集到的資料來建立樹狀式結構，從中找尋規則，並進行未知樣本預測，使用方法為：選出分類能力最好的屬性做為樹的內部節點，將內部節點的所有不同資料產生出對應的分支，遞迴重複上面的過程直到滿足終止條件。當我們使用決策樹分類演算法來訓練資料，訓練後會以特徵欄位(feature)與標籤欄位(label)建立決策樹。如圖 5：

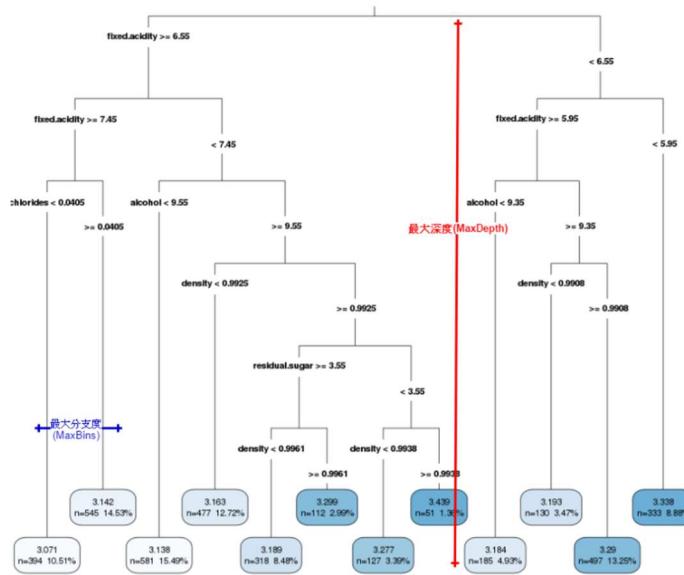


圖 5 決策樹示意圖

如圖 5，因決策樹不可能無限制成長，因此通常會限制最大分支度與深度，而在樹枝節點要分枝時，會使用吉尼指數(Gini)與熵(Entropy)作為評估分枝的方式。

#### 2. 支援向量機器(Support Vector Machine ,SVM)

classification and regression(分類及回歸)，主要功能在於將數據分類；SVM 可以使用非線性及線性的分類演算法，SVM 主要是利用其演算法在多維度的空間中(多欄位的訓練資料)，算出 Super Vector(支援向量)，並將用來尋找 Maximun Marginal Hyperplan(超平面空間)，藉此來分類資料類別，而我們則可以透過此 Maximun Marginal Hyperplan(超平面空間)，預測資料的類別。例如：鳶尾花(iris)問題，透過花萼的長寬，花瓣的長寬來分類出是哪種品種(setosa , versicolor , virginica)。如圖 6：

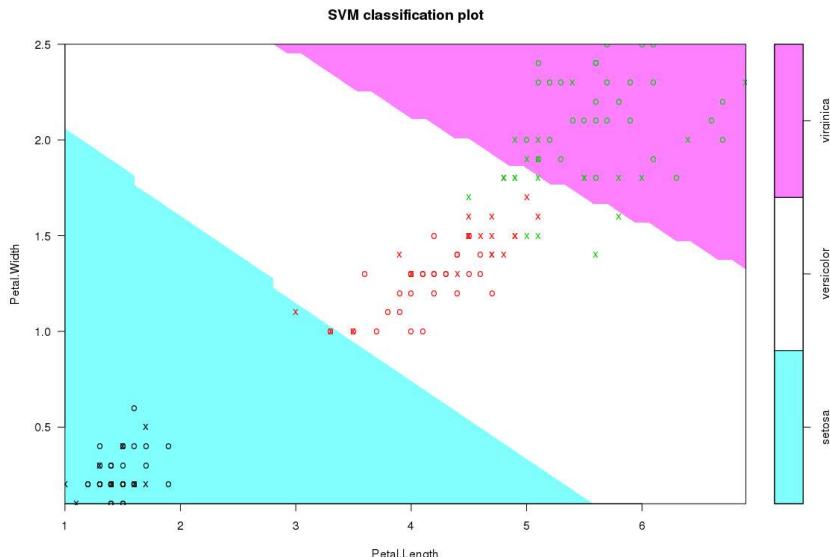


圖 6 驚尾花分布示意圖

## 2.4 機器學習模型分類模型評估

當機器學習計算完成後，會產生一個模型，我們希望知道這個模型預測的準確率，因此必須要有一個標準來評估模型準確率。

決策樹的分類模式隨著不同演算法而有不同的分類結果，由於分類規則的萃取隨著問題而異，會因環境而造成規則解的迥異，因此在客觀評估後，通常均需由該領域專家根據問題背景選出最適合的決策樹模型。

依據預測結果與資料的實際類別，共有四種組合，二元類別的分類結果可產生一個混亂矩陣(confusion matrix)，如下表 3：

		Acture 真實值	
		Positive	negatives
Predict 預測值	Positive	True Positive (TP)	False Positive (FP)
	negatives	False Negative (FN)	True Negative (TN)

表 3 混亂矩陣

假設有 2 個類別，分別為良品與不良品。

True Positive (TP): 預測為 ” 良品 ” 且實際為 ” 良品 ”

False Positive (FP): 預測為 ” 良品 ” 但實際為 ” 不良品 ”

True Negative (TN): 預測為 ” 不良品 ” 且實際為 ” 不良品 ”

False Negative (FN): 預測為 ” 不良品 ” 但實際為 ” 良品 ”

依據上述分類結果，可計算出正確率(Accuracy)或分類錯誤率(Misclassification error rate)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Error rate} = 1 - \text{Accuracy} = \frac{FP + FN}{TP + TN + FP + FN}$$

評估一個分類模型對兩個類別的分類或預測能力包含兩個指標：

### 1. 敏感度(Sensitivity)

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

亦即良品被正確預測為良品的比例。

### 2. 準確度(Specificity)

$$\text{Specificity} = \frac{TN}{TN + FP}$$

亦即不良品被正確預測為不良品的比例。

實際上，當分類結果判斷會根據不同的門檻值而有所差異，也就是改變門檻值將增加或減少判斷為良品的結果，進而影響敏感度與準確度時，可使用ROC曲線(Receiver Operating Characteristic Curve)，來判斷分類準確度。

#### 2.4.1 ROC 曲線

如圖 7 為例，其中令 TPR 為 Y 軸，且在所有實際為良品的樣本中被正確預測為良品的比率，令 FPR 為 X 軸，且在所有實際為不良品的樣本中被誤判為良品的比率，公式如下：

$$\text{TPR} = \frac{TP}{TP + FN} = \text{Sensitivity}$$

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{Specificity}$$

一般而言 TPR 越大越好，而 FPR 越小越好。因為當 TPR 增加時準確度也會減少，而  $\text{FPR}=1-\text{Specificity}$ ，所以 FPR 反而會增加，因此 ROC 曲線可以作為衡量不同 FPR 下 TPR 的變化。ROC 曲線如圖 7 所示：

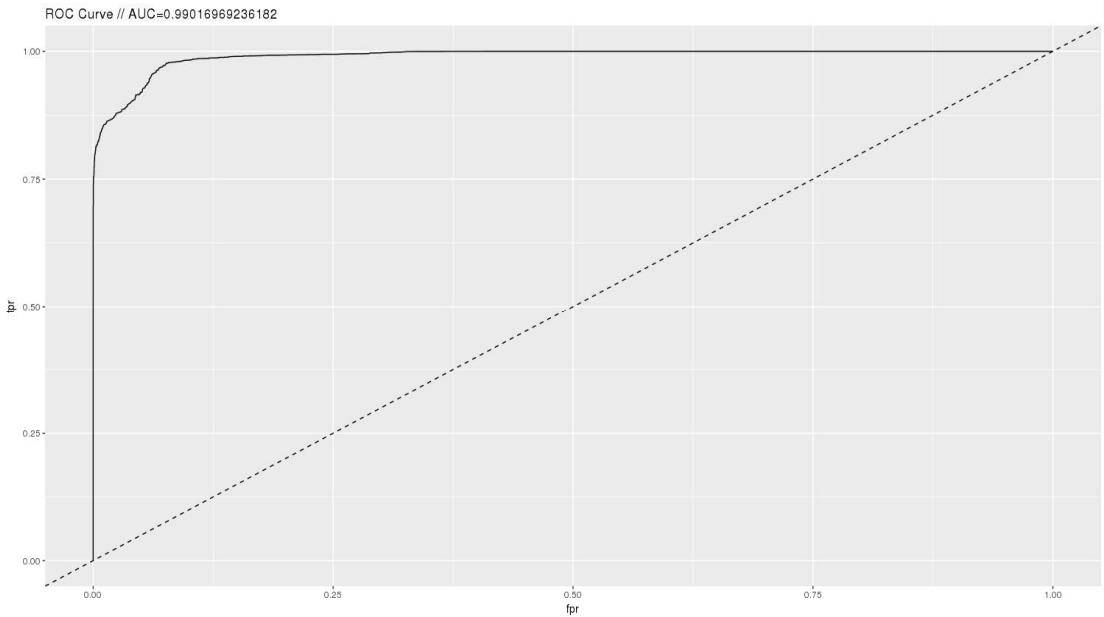


圖 7 ROC 曲線

一般而言，FPR 的結果會根據分類門檻值的不同而有所變化，此時也可以根據 ROC 曲線下的面積大小作為最佳分類結果模式也就是 AUC(Area under the Curve of ROC )來判斷分類優劣。若 ROC 曲線下的面積愈大，表示分類效果越好，反之若該模型的分類能力不佳，其面積會接近 0.5。

AUC 判斷分類優劣如表 4:

條件	說明
AUC=1	完美預測，預測準確率 100%，但不可能存在。
$0.5 < \text{AUC} < 1$	優於隨機猜測，具有預測價值。
AUC=0.5	與隨機猜測一樣。
$\text{AUC} < 0.5$	比隨機猜測還差。

表 4 AUC 判斷表

## 2.5 資料檢視、資料視覺化、資料清理

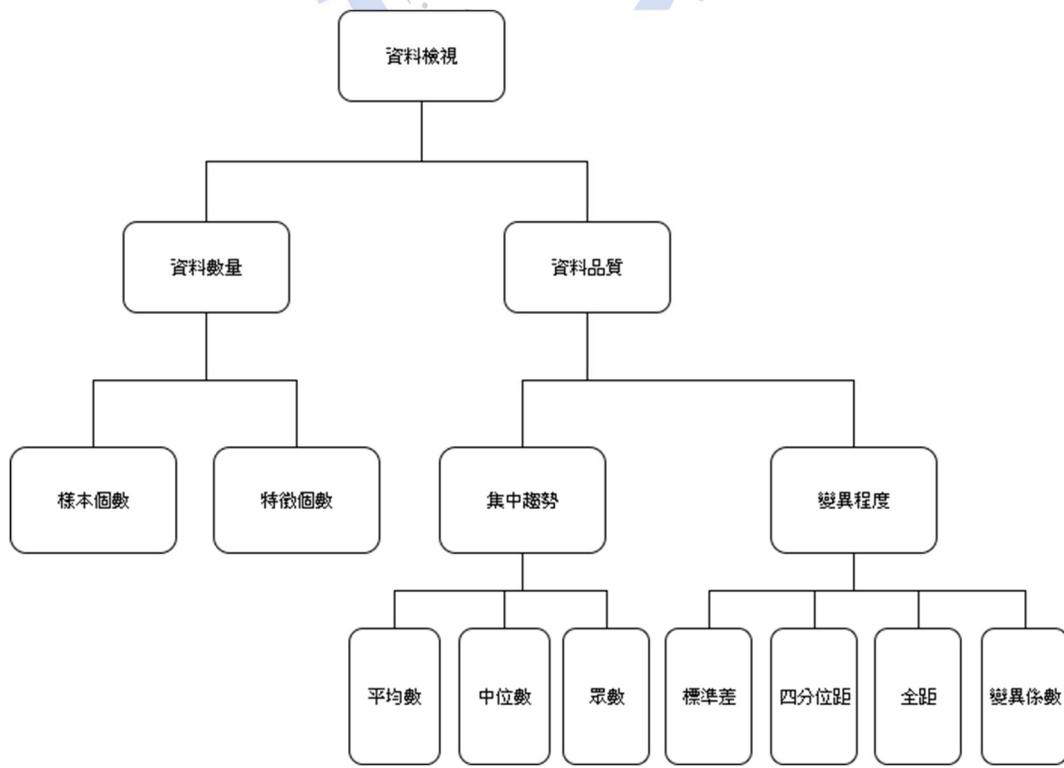
### 2.5.1 資料檢視

我們在工廠機台生產的資料往往不見得可以立即適用於資料分析，因此對資料前置處理將會較易於在資料分析時發現有意義的結果與模型。而資料檢視(data inspection)是資料預先處理的第一個步驟，檢視所獲得的資料，找出有問題的地方，並與製程工程師討論資料的目的相係數，並決定是否刪減其中的資料，否則只是從一堆錯誤的資料學習了錯誤資訊，導致一錯再錯。

資料檢視可從資料的品質及資料的數量來做歸納，資料的數量應該要檢視，樣本數量(數值筆數)、特徵資料數量(要計算的欄位個數)。樣本數量(數值筆數)太少會很容易造成誤判影響最後的分類結果，當樣本數量(數值筆數)太多時，計算時會相當耗時，且容易造成過度訓練(overfitting)，並且使得統計分析不見得有顯著意義。若特徵資料數量(要計算的欄位)太多時則會造成特徵向量(Eigenvectors)過多，增加計算的時間。

資料品質的檢視可以利用資料的集中趨勢以及變異程度(dispersion degree)，集中趨勢衡量方法包括了平均數、中位數…等；當取得一份原始時，我們可以針對原始資料中的每一個特徵欄位，使用一些基礎統計方法來評判資料，如：資料分離的程度與範圍(離散趨勢)；資料的常態分布是否為雙邊規格或單邊規格，資料的平均值是否偏離規格中心；綜合樣本資料基礎統計分析的訊息所整理出來的數值，以了解資料的變異程度及集中趨勢。

而變異程度可則可利用平均值、標準差、四分位距(interquartile range, IQR)、全距或變異係數等進行衡量，並應考慮資料的完整性，如資料分佈的一致性、或資料定義上的偏差…等；另外針對不同的資料品質，可以利用統計圖表的方式來檢視，會較易理解。如：針對資料遺漏與資料趨勢可以使用折線圖(line plot)或者是散佈圖(scatter plot)來檢視；檢視離群值的多寡可以使用盒鬚圖(box plot)來檢視；資料雜訊則可以使用管制圖(control)來檢視。資料檢視流程如圖 8



## 2.5.2 資料視覺化

在原始資料按照一定的規格，以及系統化的整理後，會的到一組資料則稱為，統計表(statistical table)，以便於後續的計算與分析。統計圖(statistical chart)為依照不同需求或特定規則，以不同圖表來表示統計資料，讓資料分析人員更容易瞭解資料的分布情形及隱含訊息。統計資料不盡相同，所適合使用的統計圖表要因地制宜的使用，才能在用較短的時間內判斷出想知道的資訊；例如：資料分佈較為離散適合使用長條圖、散佈圖，若為要表示連續性及有序性的資料，則可以直方圖與圓餅圖來表示...等。讓使用者對資料有基本的瞭解，以便進行後續的推論與分析。

## 2.5.3 資料清理(Data Cleaning)

由於人為疏忽、設備異常或抽樣方法等因素，有時會發生資料誤植、資料遺失或資料不一致、重複、矛盾等不同類型的資料問題。若直接分析這些有問題的資料將會產生錯誤或無意義的結果，因此必須藉由資料整合與清理的過程，在建立資料機器學習模式前予以修正。

資料清理的目的主要是填充且刪除遺漏值、降低雜訊與處理離群資料。

資料問題可歸類為下列幾點：

1. 不正確的資料
2. 資料的計量
3. 冗餘的資料(redundancy Data)
4. 雜訊(noise)
5. 離群值

針對各問題的處理方式如下：

### 1. 不正確的資料

要確認資料來源是否正確，是否將蝕刻機台的資料放到出貨檢查機的資料，造成資料合併到了錯誤的資料，例如：治具上的產品數量為 256 個，量測資料應為 256 筆。接下來要驗證資料的合理性，例如：一批生產的產品在某一特性下應有的值為 0~100ppm，但此筆資料卻是 999ppm，即可歸類此資料為不正確資料，通常會出現不正確資料會是作業人員手動輸入時發生的錯誤機率較大。

## 2. 資料的計量

資料的計量問題主要是由於製造業機輸出的資料，並不一定會依照機台的特性做資料分類，通常會將長度單位、重量單位、頻率單位、電壓、電流單位或轉速…等，放置於同份資料中，不同機台的來源的資料大致上不可能相同，在資料表示、界定或單位上也會有所不同，因此產生的數值欄位會有不一致的狀況。針對數值欄位的不一致情形，例如：點膠大小為 0.2mm，Base 大小為 0.2 公分，可以使用簡單的單位換算方式做處理即可。

## 3. 冗餘的資料(redundancy Data)

冗餘資料的處理方式主要是對具有相同性質或相互存在著已知數學關係的欄位，也就是此欄位的數值或代表的意義可由另一個欄位的數值推導而得。舉例來說：若資料中已經紀錄加速時間(ppm/sec)，而資料欄位中卻多紀錄了 ppm 及 sec 的欄位，則可以將上述欄位做刪除。

## 4. 雜訊(noise)

雜訊(noise)是因為資料中產生的誤差資料或干擾，產生的原因有非常多種，有可能因為機台的感應不良，探針氧化導致量測值誤差，或測不到資料等，或在工單結尾時的尾數產品，會有大部分的治具無乘載產品，因此在資料分析過程中，必須要刪除這些資料，或透過雜訊辨識技術將雜訊辨識出來後分出的資料再行刪除，以免之後的預測失真。

## 5. 離群值

在蒐集的資料中，若某一些資料的表現明顯與其他資料不一樣時，這些資料稱為離群值；例如某些特性欄位的資料大多介於 50~60ppm 之間，但有某些筆資料會落在 90ppm，合理範圍為(0~100ppm)，則稱這些資料為離群值。離群值太多會導致預測模式失焦導致預測不正確，因此，在建立挖礦模式前必須先行處理離群值，主要有下列三種處理方式：

### I. 直接刪除

當發現資料完全不合理的時候，因為有可能是空測或是探針損壞，即可考慮直接刪除該筆資料。

### II. 將資料替代

若是資料為空白或不是數值資料，且資料對於預測模式是有意義的，則必須使用容易辨識的數值來替換；例如：將多元的分類特徵欄位，使用 one hotencoder 的方式轉換為分類字典，方便後續程式提取。

### III. 集群分析

離群值可以利用集群分析偵測而得，藉由將相似的資料點結合為一個群體或集合，而散落在群集集合之外的資料值即可視為離群資料值。

## 2.6 公司簡介與產生此實驗數據的機台

公司位於高雄大寮區公司主要生產的產品為(Crystal)石英晶體振盪器，及(Oscillator)晶體振盪器，藉由與日本石英晶體振盪器公司技術提供，OEM 生產；Crystal、Oscillator 皆為目前科技產業不可或缺的一塊，主要用途為透過壓電效應，產生電子震盪電路，目的為調整電路頻率或是用來濾波，去除雜訊，端看使用者端如何設計電路，特別適合針對需要使用皮爾斯振盪器(Pierce oscillator)的各項電子產品，且產品所需零件較單純，狀態也穩定，且成本較低，因此目前被廣泛的應用在各種電子產品上。

在公司產品製程中因來料的不一致性，會透過二次加工手段作為穩定產品頻率的技術，其中 Milling 則為其中之一的製程，主要原理為濺射(Sputter)技術，在高真空的環境之下利用高電壓透過氩氣(Ar)，分離出氳離子，透過電位差的使其轟擊/ion bombardment)產品[類似撞球原理]將產品上多餘的金屬撞擊下來，最後透過邊施加電力清洗，邊將 N2 吹進產品的 Cavity，通過吸引後，可排出異物。完成頻率調整工程，整個工程中會依客戶需求，針對產品特性會設定數個階段調整速率，最終期望將產品調整至目標的頻率值之內。如圖 9 所示

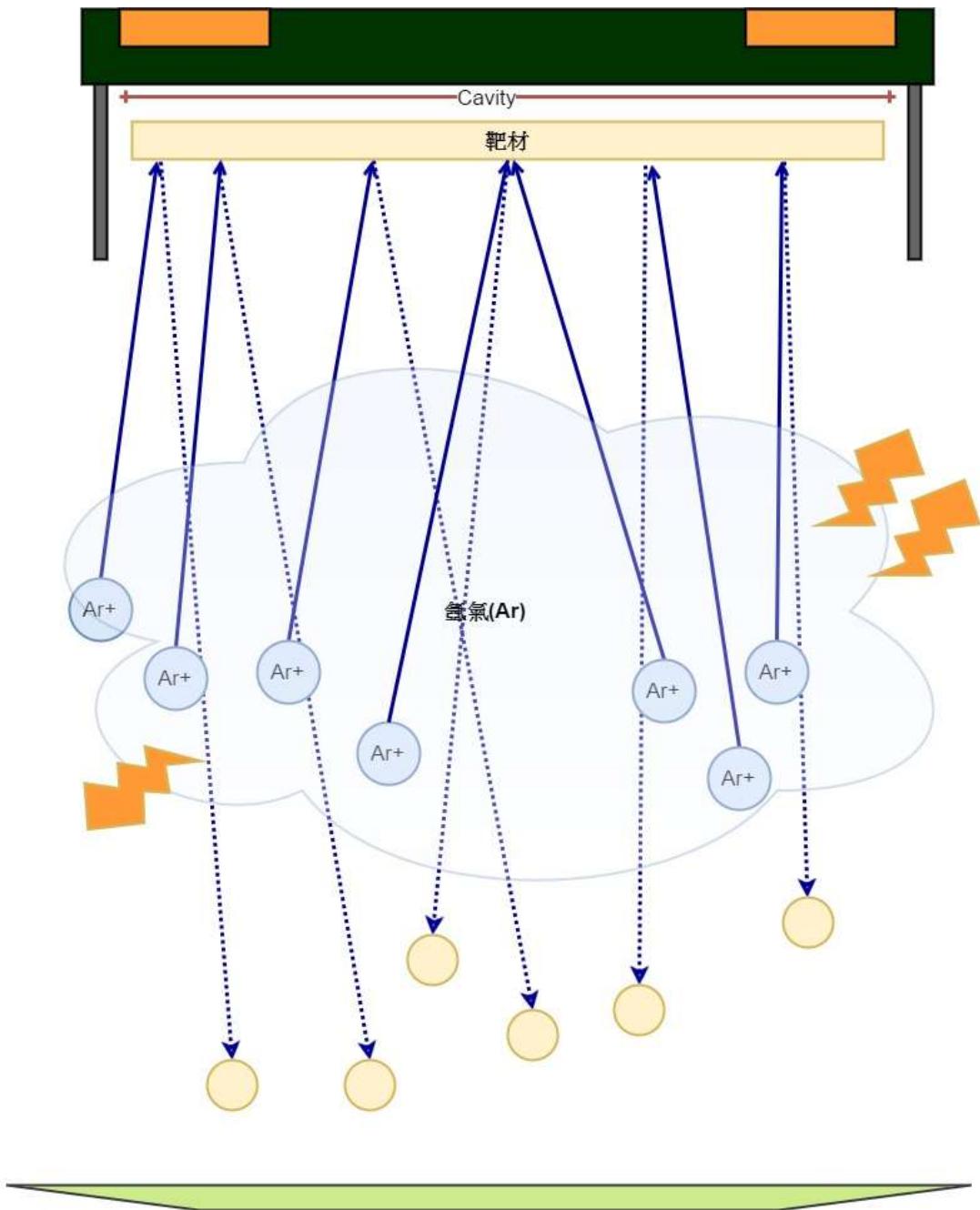


圖 9 蝕刻

## 第三章 實驗環境架設

本實驗因要模擬實際工作環境，因此安裝於 3 部虛擬機器上，而 3 部主機的名稱分別為 master、slave1、slave2，處理多部主機叢集運算功能，加快數據運算速度，實現平行化處理的目標。架構方式如圖 10 所示：

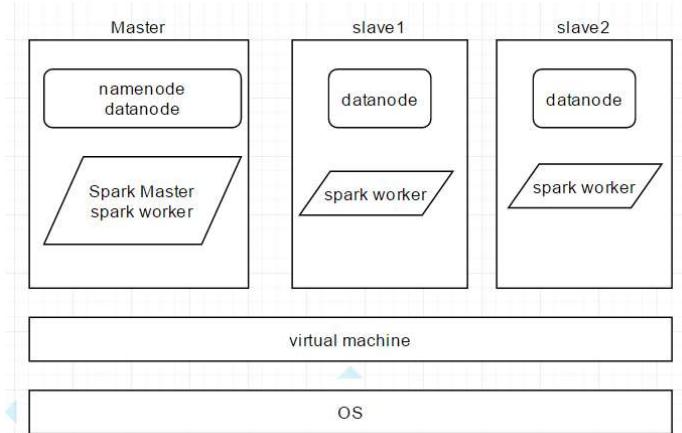


圖 10 Multi Node cluster

### 3.1 安裝 CentOS 7 要點

因為目前工作上接觸到的 Linux 系統以 CentOS 為大宗，且”鳥哥的 Linux”，在查詢及使用上幫助不少，因此此實驗底層架構為 CentOS 7，相關的 CentOS 安裝步驟已在鳥哥的網站上有詳細的說明了，在此不再贅述。

若有需要可以前往以下網址前往觀看，一步一步照著鳥哥的說明步驟去實作即可。安裝 CentOS 7 完成後的畫面如圖 11：



圖 11 CentOS 7

## 3.2 安裝 Hadoop

### 1. 安裝 Java JDK

因 CentOS 7 所預載的 JDK 為 openjdk，雖然說也可以使用，但後續會造成變數的混亂，因此要先移除，下載安裝 Oracle java JDK 並且替換掉原本的 OpenJDK，如下圖 12



圖 12 Java JDK

2. 下載 Hadoop 並安裝可至 Apache Hadoop 官網下載 source，下載完將其解壓縮放在/usr/local/Hadoop 資料夾中即可。
3. 設定 Hadoop 環境變數、組態檔及格式化 HDFS，其中因 OpenJDK 修改為 Oracle java JDK，因此要將 JAVA\_HOME 改為 Oracle java JDK 才行。
4. 啟動及查看 Hadoop 是否為執行中。

最後在終端機輸入 start-all.sh 即可以啟用 Hadoop，可以透過 Web 介面：

<http://localhost:8888> 查看是否啟動，或是於終端機輸入 JPS 即可查看目前狀態，最後還可以特別使用 hadoop dfsadmin -report 查詢 Hadoop 狀態。

如圖 13 所示：

```

[root@master ~]# jps
3297 NodeManager
3717 Worker
2886 DataNode
3191 ResourceManager
3863 Worker
3640 Master
3800 Worker
10682 Jps
3037 SecondaryNameNode
2782 NameNode
[root@master ~]# hadoop dfsadmin -report
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Configured Capacity: 88993210368 (82.88 GB)
Present Capacity: 68944945152 (64.21 GB)
DFS Remaining: 68944920576 (64.21 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

-----
Live datanodes (3):

Name: 172.16.0.218:50010 (master)
Hostname: master
Decommission Status : Normal
Configured Capacity: 29664403456 (27.63 GB)
DFS Used: 8192 (8 KB)
Non DFS Used: 8149688320 (7.59 GB)
DFS Remaining: 21514706944 (20.04 GB)
DFS Used%: 0.00%
DFS Remaining%: 72.53%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Thu May 04 06:22:37 CST 2017

Name: 172.16.0.219:50010 (slave1)
Hostname: slave1
Decommission Status : Normal
Configured Capacity: 29664403456 (27.63 GB)
DFS Used: 8192 (8 KB)
Non DFS Used: 5904003072 (5.50 GB)
DFS Remaining: 23760392192 (22.13 GB)
DFS Used%: 0.00%
DFS Remaining%: 80.10%

```

圖 13 Hadoop 狀態

### 3.3 設定 Multi Node Cluster

1. 設定好 master 與 slave 的 IP 及 hosts name 的相互對應關係
2. 在 master 與 slave 之間設定好 SSH 的無密碼登入憑證，方便 NameNode 於 DataNode 之間得相互溝通與資料傳輸。

### 3.4 安裝 Spark

1. 下載 scala 解壓縮並安裝於 /usr/local/scala，並且設置路徑。
2. 下載可支援 Hadoop 版本的 spark，解壓縮並安裝於 /usr/local/scala，並且設置相關參數。
3. 更改 /usr/local/spark/conf/slave，增加 master 與 slave。
4. 啟動 spark，/usr/local/spark/sbin/start-all.sh

### 3.5 安裝 Rstudio

1. 請將 CentOS 7 更新到最新版本。
2. 因為要使用 Rstudio 所以需要下載 R 的相關套件如 yum-cron、epel-release、R，使用 YUM 下載即可。
3. 步驟 2 做完後可直接至 Rstudio 官網下載 Rstudio-Sever，依循 Linux 的版本，參照官網指示安裝，即可。
4. 打開瀏覽器輸入 <http://master:8787> 輸入帳號及密碼後即可在瀏覽器上使用 Rstudio 了。執行畫面如圖 14 所示

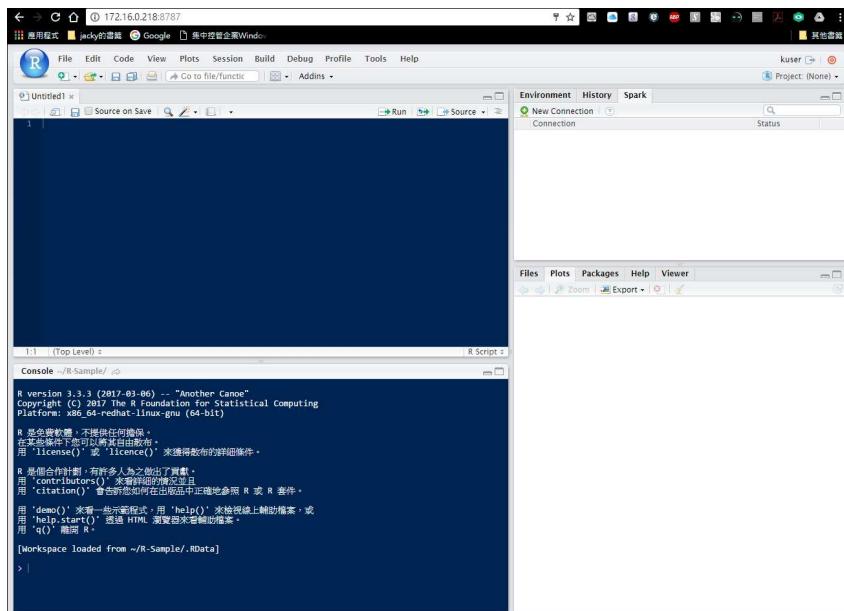


圖 14 Rstudio 開發工具

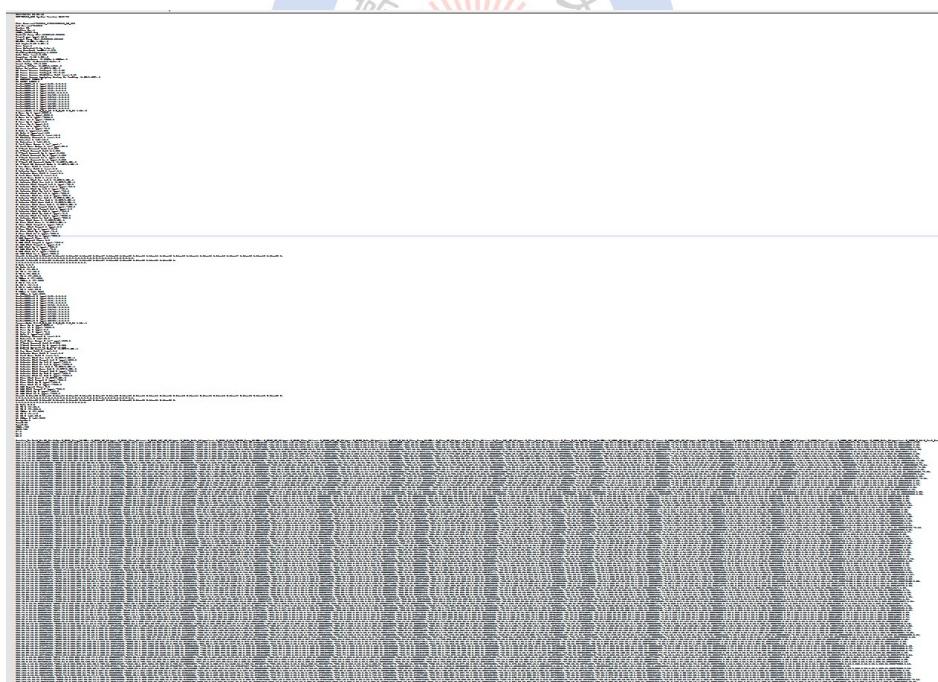
## 第四章 實驗數據分析

本章節介紹此論文中研究相關的數據，並依文獻探討中的論述做剖析

- 4.1 節-檢視原始數據
- 4.2 節-原始資料切割
- 4.3 節-切割後的資料視覺化
- 4.4 節-資料彙整
- 4.5 節-機器學習與 SVM 建模

### 4.1 檢視原始數據

在實務上而言，機台數據的生成取決於工廠製程端的管理，及程式設計者依機台的特性做的，而 `RowData` 一直以來並無特定格式或規範(也許有的公司有)，因此在拿到資料要做分析時可以先刪除一些不需要的資料，比如說這次要研究的機台資料。原始資料如圖 15:



The image shows a screenshot of a computer screen displaying a large dataset in RStudio. The data is presented as a grid of numerous small, dark grey numbers, filling most of the visible area. Above the grid, there is a watermark or logo that reads "台灣應用科大" (Taiwan University of Technology) in a circular arrangement. The overall appearance is that of a raw, unprocessed data file.

圖 15 原始資料

從圖中可看到前面一大段的資料，對於放入大數據研究時會不好分析，就這台機台，資料在輸出時會限制資料筆數。因此若我們需要的是數百萬筆的分析資料，就得將此資料做合併及刪除的動作才能進行下一步的匯入數據。

## 4.2 原始資料切割

在此使用 Visual Studio 撰寫 windows form 做分析前期資料切割及合併多筆資料，如圖 16 所示：

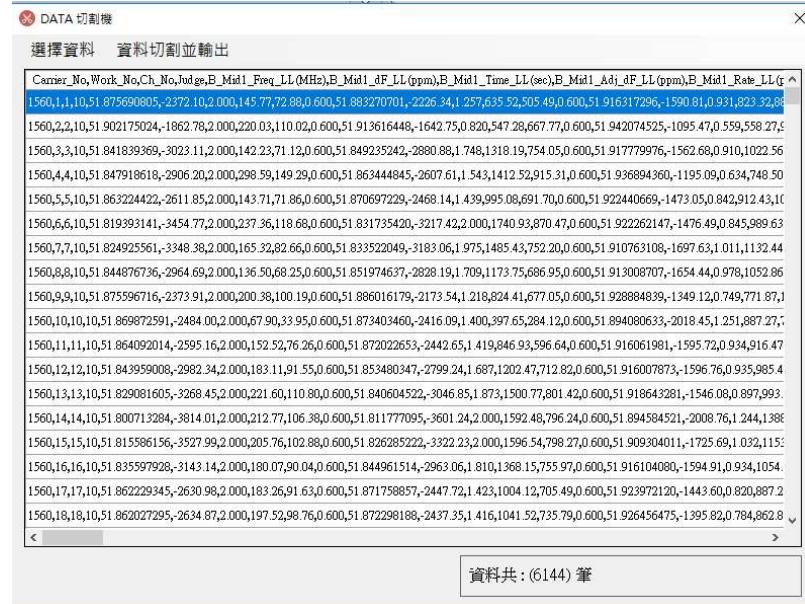


圖 16 原始資料切割程式

切割合併完的資料會較為整齊，也易於後續分析如圖 17 所示

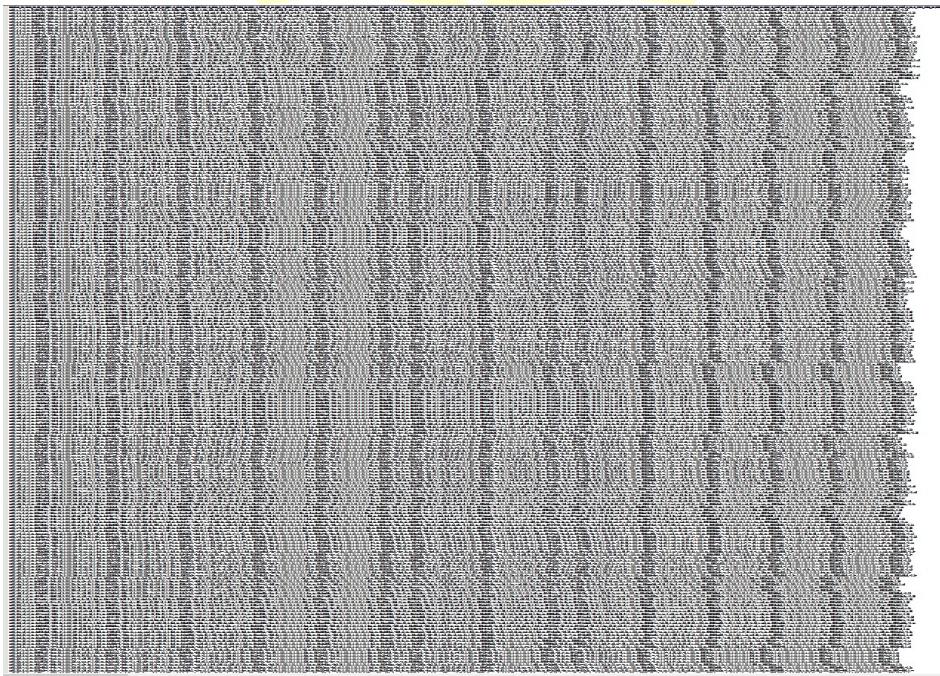


圖 17 原始資料切割後

因資料的欄位數量取決於製程的設定，因此可以透過與製程工程師的討論找尋在意的特徵欄位，若一時間無法決定可透過下個步驟，資料視覺化找尋特徵欄位。

Code:{

```

Try
    ReadData = Split(FileReader, vbCrLf)
    //找原始資料的標題欄
    For j = 0 To ReadData.Count() - 1
        If InStr(ReadData(j), "Carrier_No,Work_No,Ch_No,Judge,", CompareMethod.Text) >= 1
            Then
                HeaderTitle = j
                Exit For
            End If
        Next
        //讀取標題欄之後的檔案，因為廠商在出資料時最後一筆資料後，會有多一筆空資料因此將 ReadData.Length - 2
        For k = HeaderTitle + 1 To ReadData.Length - 2
            If ReadData(k).Length > 0 Then
                Data = ReadData(k)
                ZAJTable.Rows.Add(Data)
            End If
        Next
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    Try
        TitleStr = ZAJTable.Columns(0).ColumnName
        //因為原始檔案在每列的最後有多出一個空格，所以將它讀出來再塞回去。
        For i = 0 To ZAJTable.Rows.Count() - 1
            DataStr = ZAJTable.Rows(i).Item(0)
            Str = DataStr.Substring(0, Len(DataStr) - 1)
            ZAJTable.Rows(i).Item(0) = Str
        Next
        //一格一格輸出太慢，當資料量大時很耗時，所以一列一列的 I/O 輸出較快
        FileWriteLine.WriteLine(ZAJTable.Columns(0).ColumnName)
        For i = 0 To ZAJTable.Rows.Count - 1
            FileWriteLine.WriteLine(ZAJTable.Rows(i).Item(0))
        Next
        Catch ex As Exception
            MsgBox(ex.Message)

```

```

End Try
}

```

### 4.3 切割後的資料視覺化

在實務上使用者較常利用常態分佈的直條圖，來做集中趨勢的衡量、變異程度，及資料的離散趨勢，或者利用折線圖觀看數據的直接變化程度。

在此使用 Vistual Studio 撰寫 windows form，做資料的趨勢圖以利工程師做專業的判斷依據。

直條圖：

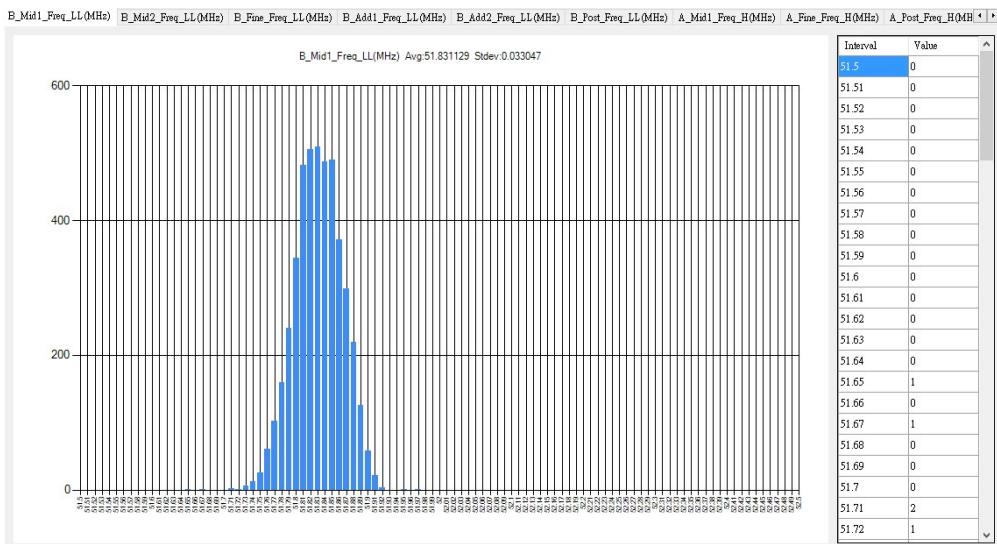


圖 18 資料視覺化(直方圖)

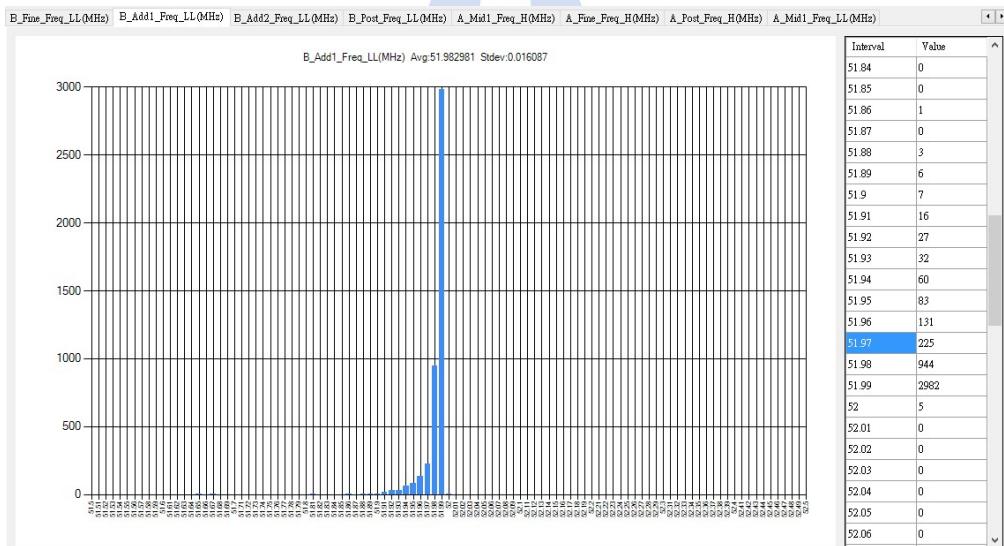


圖 19 資料視覺化(直方圖)



圖 20 資料視覺化(直方圖)

Code:{

Try

```

k = 0
RowCount=Data_Table.Rows.Count
P1(RowCount - 1)
XMax, Xmin, Xinterval, XGaps
XMax =52.5
Xmin = 51.5
XGaps =100
Xinterval = (XMax - Xmin) / XGaps
For i = 0 To RowCount - 1
    Try
        a1 = Data_Table.Rows(i).Item(ColumePosition))
    Catch ex As Exception
        a1 = 0
    End Try
    //使用數學方法做上下界的區別，好處是計算較快
    If XMax >= a1 And a1 >= Xmin Then
        'Dim a2 = Math.Abs(a1 - Maxmu)
        Dim a2 = Math.Abs(Xmin - a1)
        Dim a3 = a2 / Xinterval
        Dim a4 = Math.Ceiling(a3) '抓上界
        Dim a5 = Math.Floor(a3) '抓下界
        'Dim a6 = Maxmu - (a5) * InterVal
        'Dim a7 = Maxmu - (a4) * InterVal
        Dim a6 = Xmin + (a5) * Xinterval
    End If
End Sub

```

```

Dim a7 = Xmin + (a4) * Xinterval
If a1 > a7 And a1 <= a6 Then
    P1(k) = a6
Else
    P1(k) = a7
End If
k = k + 1
End If

Next
Array.Resize(P1, k)
Array.Sort(P1)

//計算平均值、標準差、
PAvg = P1.Average
For j = 0 To k - 1
    PStdev = PStdev + (P1(j) - PAvg) ^ 2
Next
PStdev = Math.Sqrt(PStdev / (k - 1))
//將剛剛整理過後的值塞進 Dictionary 之中，方便日後查詢使用
DValue = String
Dcount = Integer
FeqD = Dictionary(Of String, Integer)

For j = 0 To k - 1
    DValue = P1(j)
    Try
        Dcount = FeqD(DValue)
        FeqD(DValue) = Dcount + 1
    Catch ex As Exception
        FeqD(DValue) = 1
    End Try
Next

Catch ex As Exception
    MsgBox(ex.Message)
End Try
}

```

折線圖：可以明顯看出離群值，以利後續資料清理。

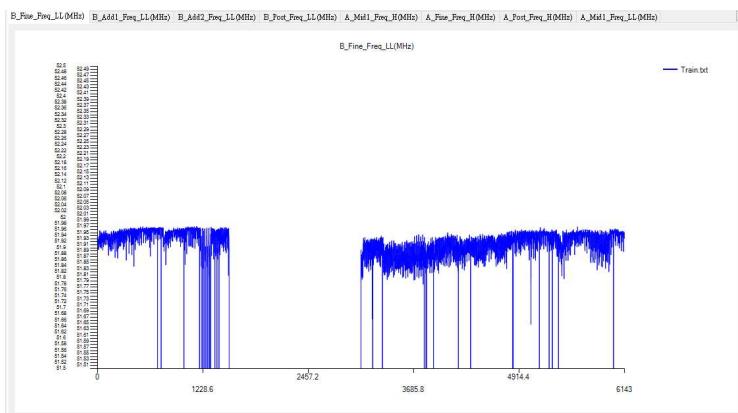


圖 21 資料視覺化(折線圖)

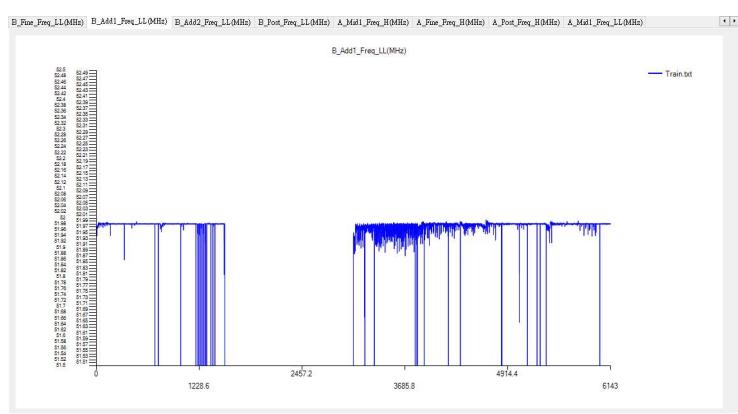


圖 22 資料視覺化(折線圖)

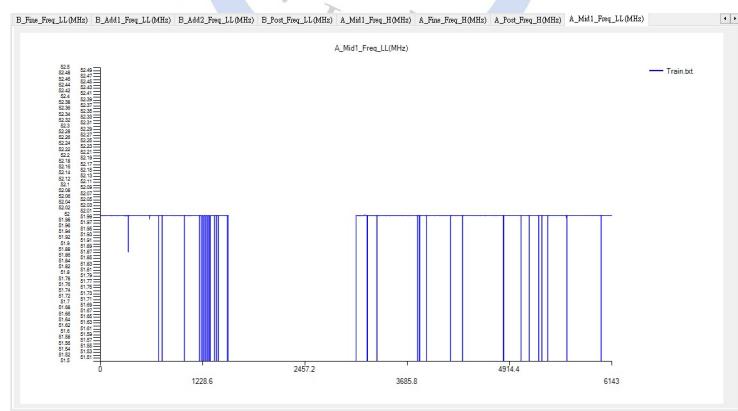


圖 23 資料視覺化(折線圖)

Line Code:

```
For i = 0 To LoadingData_Table.Rows.Count - 1  
    L1(i) = LoadingData_Table.Rows(i).Item(ColumePosition)  
Next
```

## 4.4 資料彙整

### 4.4.1 資料解釋

機台生產資料的產生流程，如圖 24：

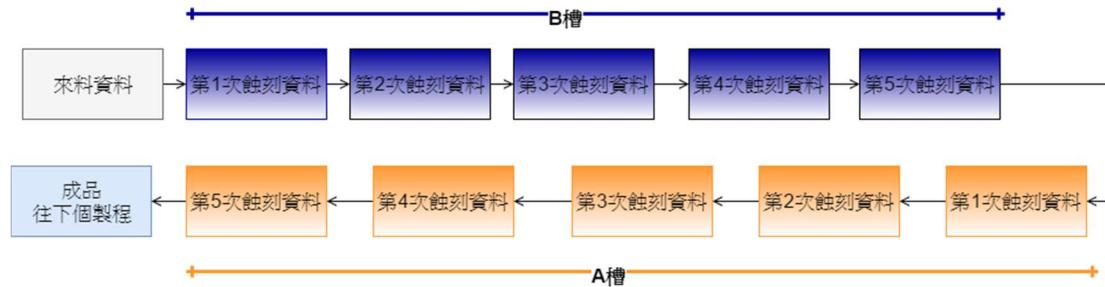


圖 24 更改後的 Label 欄位

透過 4.1 小節，使用程式合併出來的原始資料如圖 25。

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
1	Carrier_No	Work_No	Ch_No	Judge	B_Mid1_F	B_Mid1_d	B_Mid1_l	T_B_Mid1_A	B_Mid1_R	B_Mid1_K	B_Mid2_F	B_Mid2_d	B_Mid2_l	T_B_Mid2_A	B_Mid2_R	B_Mid2_K	E_Fine_Freq	E_Fine_dTime	E_Fine_Time	E_Fine_Adjust
2	489	1	1	10	51.79982	-383.91	2	257.14	128.57	0.6	51.81319	-3574.77	2	2221.92	1110.96	0.6	51.92873	-1352.85	0.802	1019.33
3	489	2	2	10	51.76201	-4559.1	2	283	141.5	0.6	51.79763	-4276.11	2	2347.3	1173.65	0.6	51.89878	-1928.81	1.263	1664.93
4	489	3	3	10	51.69341	-5878.47	2	270.81	135.4	0.6	51.70749	-5607.67	2	2328.54	1164.27	0.6	51.82857	-3279.13	2	2645.8
5	489	4	4	10	51.77192	-4368.5	2	211.88	105.94	0.6	51.78294	-4156.62	2	2246.09	1123.05	0.6	51.89973	-1910.53	1.248	1649.04
6	489	5	5	10	51.78301	-4155.32	2	188.61	94.3	0.6	51.79281	-3966.71	2	2089.22	1044.61	0.6	51.90145	-1877.49	1.222	1535.63
7	489	6	6	10	51.83474	-3160.42	2	109.6	54.8	0.6	51.80404	-3050.82	2	1485.35	742.68	0.6	51.91768	-1565.47	0.972	1066.81
8	489	7	7	10	51.81502	-3539.68	2	242.3	121.15	0.6	51.82762	-3297.38	2	2217.96	1108.98	0.6	51.94295	-1079.42	0.584	758.5
9	489	8	8	10	51.80802	-3674.21	2	192.52	96.26	0.6	51.81803	-3481.69	2	1869.2	934.6	0.6	51.91523	-1612.49	1.01	1189.98
10	489	9	9	10	51.75244	-4743.09	2	455.68	227.84	0.6	51.77614	-4287.41	2	2691.98	1345.99	0.6	51.91612	-1595.43	0.996	1440.7
11	489	10	10	10	51.72749	-5222.96	2	202.27	101.14	0.6	51.73801	-5020.69	2	1959.91	979.96	0.6	51.83992	-3060.78	2	2403.98
12	489	11	11	10	51.71266	-5508.18	2	138.29	69.14	0.6	51.71985	-5369.9	2	1782.34	891.17	0.6	51.81253	-3587.56	2	2405.29
13	489	12	12	10	51.77878	-4236.52	2	136.83	68.42	0.6	51.78759	-4099.69	2	1611.22	805.61	0.6	51.86968	-2488.47	1.711	1892.14
14	489	13	13	10	51.77402	-4328.09	2	184.2	92.1	0.6	51.78336	-4143.89	2	2026.82	1013.41	0.6	51.88899	-2117.07	1.414	1764.19
15	489	14	14	10	51.78811	-4057.23	2	284.69	142.34	0.6	51.80291	-3772.54	2	2316.58	1158.29	0.6	51.92337	-1455.96	0.885	1154.14
16	489	15	15	10	51.83098	-3232.71	2	215.06	107.53	0.6	51.84216	-3017.65	1.974	2038.03	1032.37	0.6	51.94814	-979.62	0.504	617.91
17	489	16	16	10	51.84365	-2989.04	2	126.33	63.17	0.6	51.85022	-2862.7	1.85	1495.1	808.09	0.6	51.92797	-1367.6	0.814	924.75
18	489	17	17	10	51.781	-4193.92	2	322.37	161.18	0.6	51.79776	-3871.55	2	2411.43	1205.71	0.6	51.92315	-1460.12	0.888	1198.03
19	489	18	18	10	51.76313	-4537.55	2	353.58	176.79	0.6	51.78152	-4183.97	2	2463.7	1231.85	0.6	51.90963	-1720.27	1.096	1469.9
20	489	19	19	10	51.73467	-5084.86	2	239.18	119.59	0.6	51.74711	-4845.71	2	2236.55	1118.28	0.6	51.86341	-2609.15	1.807	2376.52
21	489	20	20	10	51.72044	-5538.58	2	222.74	111.37	0.6	51.73202	-5135.84	2	2159.81	1079.9	0.6	51.84433	-2976.03	2	2561.47
22	489	21	21	10	51.78196	-4175.42	2	168.43	84.21	0.6	51.79072	-4006.99	2	1900.82	950.41	0.6	51.88956	-2106.17	1.405	1674.96
23	489	22	22	10	51.80309	-3769.15	2	225.21	112.61	0.6	51.81448	-3543.94	2	2099.07	1049.53	0.6	51.92395	-1444.87	0.876	1077.72
24	489	23	23	10	51.81598	-3521.22	2	383.28	191.64	0.6	51.83591	-3137.94	2	2529	1264.5	0.6	51.96742	-608.94	0.207	278.98
25	489	24	24	10	51.821	-3424.59	2	102.35	51.17	0.6	51.82633	-3322.24	2	1352.26	676.13	0.6	51.89664	-1969.98	1.296	1371.12
26	489	25	1	10	51.78353	-4145.2	2	325.71	162.86	0.59	51.80047	-3819.49	2	2330.94	1165.47	0.59	51.92168	-1488.54	0.896	1151.93
27	489	26	2	10	51.74853	-4818.27	2	334.87	167.44	0.569	51.76595	-4483.4	2	2410.52	1205.26	0.569	51.89129	-2072.88	1.307	1718.95
28	489	27	3	10	51.73003	-5174.12	2	361.57	180.79	0.567	51.74883	-4812.54	2	2457.97	1228.99	0.567	51.87664	-2354.57	1.515	2007.42
29	489	28	4	10	51.74232	-4937.69	2	456.65	228.33	0.568	51.76607	-4481.04	2	2549.04	1274.52	0.568	51.88862	-1932	1.198	1590.74
30	489	29	5	10	51.73481	-5082.24	2	221.43	110.72	0.597	51.74632	-4860.81	2	2168.76	1084.38	0.597	51.8591	-2692.04	1.864	2351.69
31	489	30	6	10	51.77839	-4244.03	2	129.52	64.76	0.684	51.78513	-4114.52	2	1632.95	816.47	0.684	51.87004	-2481.57	1.943	2173.26
32	489	31	7	10	51.81609	-3519.06	2	311.52	155.76	0.577	51.83229	-3207.54	2	2380.5	1190.25	0.577	51.95607	-827.05	0.367	480.5
33	489	32	8	10	51.83572	-3141.49	2	183.63	91.82	0.637	51.84527	-2957.86	2	1911.11	955.56	0.637	51.94465	-1046.75	0.591	699.35
34	489	33	9	10	51.77115	-4383.4	2	469.5	234.75	0.519	51.79556	-3913.9	2	2730.7	1365.35	0.519	51.93755	-1183.2	0.576	843.73
35	489	34	10	10	51.75918	-4613.56	2	207.09	103.55	0.624	51.76995	-4406.46	2	1991.41	995.71	0.624	51.8735	-2415.05	1.718	2074.38

圖 25 切割過後的原始資料

每一行的資料值皆是每顆產品的製程數據，如圖 4.4.2 : Work\_No=4，這一行，在底色為黃色的欄位中，會是第一次蝕刻時的量測資料，底色為綠色的欄位中，會是第二次蝕刻時的量測資料，量測資料的產生為每次蝕刻後探針去量測的資料，另外每次的蝕刻均會產生 6 個欄位，分別為 Freq(MHZ)-頻率、df(ppm)-蝕刻濃度，Time(sec)-時間，Adj(ppm)-此次蝕刻刪減量，Rate(ppm/sec)-蝕刻速率，K-初始化時間。圖 26 為每次蝕刻時的資料：

C	D	E	F	G	H	I	J	K
Ch_No	Judge	B_Mid1_Freq_LL(MHz)	B_Mid1_dF_LL(ppm)	B_Mid1_Time_LL(sec)	B_Mid1_Adj_dF_LL(ppm)	B_Mid1_Rate_LL(ppm/sec)	B_Mid1_K_LL	B_Mid1_K_H
2	1	10	51.7998236	-383.91	2	257.14	128.57	0.6
3	2	10	51.76201033	4559.1	2	283	141.5	0.6
4	3	10	51.69340475	-5878.47	2	270.8	135.4	0.6
5	4	10	51.77192145	4368.5	2	211.88	105.94	0.6
6	5	10	51.78300691	4155.32	2	188.61	94.3	0.6
7	6	10	51.83474044	-3160.42	2	109.6	54.8	0.6

圖 26 第一次蝕刻資料

#### 4.4.2 資料整理

針對原始資料整理，透過經驗法則了解資料的形態及組態後，我們可以再進一步的了解資料的特性，並且透過特徵欄位的平均值、標準差、常態分佈圖、折線圖，可以了解此資料的狀態，並且刪除不正確的資料及冗餘的資料、或是非特徵值的資料。

##### 4.4.2.1 刪除非特徵資料

圖 27 欄位底色為淺藍色的資料為非特徵資料值，可以直接做刪除。

圖 27 非特徵資料

刪除後的資料如圖 28:

A	B	C	D	E	F	G	H	I	J	K	L	M
Judge	B_Mid1_Freq_LL	B_Mid1_dF_LL	B_Mid1_K_LL	B_Mid1_A	B_Mid1_R	B_Mid1_KB_F	B_Mid2_d	B_Mid2_T	B_Mid2_A	B_Mid2_R	B_Mid2_KB_F	
11	10	51.72749	-5222.96	2	202.27	101.14	0.6	51.73801	-5020.69	2	1959.91	979.96
12	10	51.71266	-5508.18	2	138.29	69.14	0.6	51.71985	-5369.9	2	1782.34	891.17
13	10	51.77878	-4236.52	2	136.83	68.42	0.6	51.7859	-4099.69	2	1611.22	805.61
14	10	51.77402	-4328.09	2	184.2	92.1	0.6	51.7836	-4143.89	2	2026.82	1013.41
15	10	51.78811	-4057.23	2	284.69	142.34	0.6	51.80291	-3772.54	2	2316.58	1158.29
16	10	51.83098	-3232.71	2	215.06	107.53	0.6	51.84216	-3017.65	1.974	2038.03	1032.37
17	10	51.84365	-2989.04	2	126.33	63.17	0.6	51.85022	-2862.7	1.85	1495.1	808.09
18	10	51.781	-4193.92	2	322.37	161.18	0.6	51.79776	-3871.55	2	2411.43	1205.71
19	10	51.76313	-4537.55	2	353.58	176.79	0.6	51.78152	-4183.97	2	2463.7	1231.85
20	10	51.73467	-5084.88	2	239.18	119.59	0.6	51.74711	-4845.71	2	2236.55	1118.28
21	10	51.72044	-5358.58	2	222.74	111.37	0.6	51.73202	-5135.84	2	2159.81	1079.9
22	10	51.78196	-4175.42	2	168.43	84.21	0.6	51.79072	-4006.99	2	1900.82	950.41
23	10	51.80309	-3769.15	2	225.21	112.61	0.6	51.8148	-3543.94	2	2099.07	1049.53
24	10	51.81598	-3521.22	2	383.28	191.64	0.6	51.83591	-3137.94	2	2529	1264.5
25	10	51.821	-3214.50	2	102.35	51.17	0.6	51.88633	-3322.24	2	1352.26	676.13

圖 28 刪除非特徵欄位



Judge 欄位分析，因為要實驗的 SVM 為機器學習中的監督式學習法，因 Judge 為識別欄位分為 10(良品)、11(不良品-不良原因 1)、12(不良品-不良原因 2)、13(不良品-不良原因 3)、14(不良品-不良原因 4)，可透過 execl 的篩選判斷出來，如圖 32。

圖 32 Judge 欄位篩選

其中 Judge=11 時有時候會有值，有時候沒有值，因此在判斷上無法將其判斷為離峰資料作為刪除依據，因此將其保留。

圖 33 資料產生流程

另外因為 10、11、12、13、14 在識別因為是數字不好識別，因此將此 5 個數字直接使用 EXECL 的取代功能做修改，故將 10 改成 OK，11、12、13、14 分別改為 error1、error2、error3、error4。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Judge	[r]	B_Mid1	[r]	B_Mid1	[r]	B_Mid	[r]	B_Mid2	[r]	B_Mid2	[r]	B_Fine_	[r]	B_Fine_	[r]
191	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
196	error4	51.8025	-3780.51	2	51.82022	-3439.76	2	51.82022	-3439.76	0	51.82022	-3439.76	0	51.82022	-3439.76	0
325	error4	51.76996	4406.31	2	51.776	-4290.1	2	51.86156	-2644.74	2	51.97717	-421.28	0.064	51.97717	-421.28	0
441	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
921	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1060	error4	51.85815	-2710.3	2	51.87848	-2319.21	1.343	51.87848	-2319.21	0	51.87848	-2319.21	0	51.87848	-2319.21	0
1425	error4	51.76006	-4596.61	2	51.77101	-4385.96	2	51.87813	-2325.98	1.731	51.9814	-340.03	0	51.9814	-340.03	0
1582	error4	51.796	-3905.52	2	51.80019	-3824.84	2	51.80019	-3824.84	0	51.80019	-3824.84	0	51.80019	-3824.84	0
1610	error4	51.85568	-2757.77	2	51.85568	-2757.77	0	51.85568	-2757.77	0	51.85568	-2757.77	0	51.85568	-2757.77	0
1654	error4	51.74705	-4846.88	2	51.75201	-4751.52	2	51.8489	-2888.17	2	51.97114	-537.28	0.159	51.98035	-360.12	0
1839	error4	51.79107	-4000.32	2	51.80221	-3786.03	2	51.92411	-1441.78	0.832	51.98133	-341.29	0	51.98133	-341.29	0
2130	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2197	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2438	error4	51.81133	-3610.65	2	51.81133	-3610.65	0	51.81133	-3610.65	0	51.81133	-3610.65	0	51.81133	-3610.65	0
2688	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2850	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2881	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2911	error1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2943	error1	51.77953	-4222.1	2	51.79935	-3841.01	2	51.93019	-1324.85	0.747	51.98124	-343.03	0	51.98124	-343.03	0
3821	error4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3957	error4	51.83287	-3196.43	2	51.84643	-2935.54	1.828	51.84643	-2935.54	0	51.84643	-2935.54	0	51.84643	-2935.54	0
3992	error1	51.67435	-6244.86	0	51.67435	-6244.86	0	51.67435	-6244.86	0	51.67435	-6244.86	0	51.67435	-6244.86	0
4016	error1	51.67002	-6328.17	0	51.67002	-6328.17	0	51.67002	-6328.17	0	51.67002	-6328.17	0	51.67002	-6328.17	0
4040	error1	51.66656	-6394.64	0	51.66656	-6394.64	0	51.66656	-6394.64	0	51.66656	-6394.64	0	51.66656	-6394.64	0
4048	error1	51.68231	-6091.9	0	51.68231	-6091.9	0	51.68231	-6091.9	0	51.68231	-6091.9	0	51.68231	-6091.9	0
4064	error1	51.66833	-6360.66	0	51.66833	-6360.66	0	51.66833	-6360.66	0	51.66833	-6360.66	0	51.66833	-6360.66	0

圖 34 更改後的 Judge 欄位

以上，彙整出最後的資料，可以使用 Excel 工具匯出成 CSV 檔，可以用來幫助放入 SVM 時問題的釐清。



## 4.5 機器學習與 SVM 建模

### 4.5.1 匯入資料

連線至第三章所建置的 Rstudio-server 將 4.4 節整理過後的 TrainingData.csv(包含 Label 共 37 個 variables，資料筆數為 69888 筆) 匯入檔案夾中，方便日後取用 Data，如圖 35。

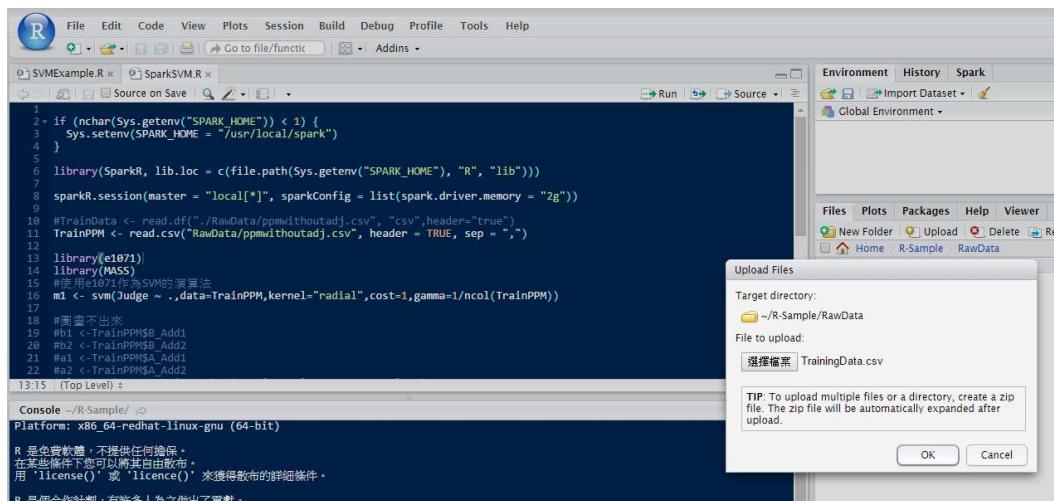


圖 35 上傳檔案至 Rstudio-server

### 4.5.2 使用 SVM 建立預測模型

1. 使用 R 語言的套件 e1071 作為 SVM 的演算法

```
library(e1071)
```

2. 使用 MASS 建立混亂矩陣

```
library(MASS)
```

3. 計算混亂矩陣

```
library(caret)
```

4. 載入訓練資料

```
TrainData <- read.csv("RawData/TrainingData.csv", header = TRUE, sep = ",")
```

5. 將資料 table 切割成 x:不含 Label 的資料，與 Label 資料

```
x <- subset(TrainData, select=-Judge)
```

```
y <- TrainData$Judge
```

6. 開始建立模型

```
mod1 <- svm(x,y)
```

7. 查看模型狀態，如用的是哪種分類法，預設使用的是 c-classification，kernel 為:radial，及可以看到 Number of Support Vectors 為 263
- ```
summary(mod1)
```

```
> summary(mod1)

Call:
svm.default(x = x, y = y)

Parameters:
  SVM-Type: C-classification
  SVM-Kernel: radial
    cost: 1
   gamma: 0.02777778

Number of Support Vectors: 263
( 146 22 66 29 )

Number of Classes: 4

Levels:
 error1 error3 error4 OK
```

圖 36 mod1 的結果

#### 4.5.3 計算預測準確度

1. 將模型擲回目的:確認模型準確度

```
Model_result <- predict(mod1,x)
```

2. 查看計算結果

```
summary(Model_result)
```

```
> summary(Model_result)
error1 error3 error4      OK
 1101     15     87 68685
```

圖 37 Model\_result 的結果

3. 將結果與 Label 做成混亂矩陣

```
table(Model_result,y)
```

```
> table(Model_result,y)
      y
Model_result error1 error3 error4      OK
  error1     1101      0      0      0
  error3       0     15      0      0
  error4       0       0     87      0
  OK          0      15      5 68665
```

圖 38 混亂矩陣

4. 使用 caret 的函數 confusionMatrix 計算混亂矩陣，並查看結果。

```
cm <- confusionMatrix(table(Model_result,y))
```

```
cm
```

```
> cm
Confusion Matrix and Statistics

      labelPPM
pred_result Error1 Error2 Error3    OK
Error1       1093      0      0      0
Error2        0     10      0      0
Error3        0      0     86      0
OK           8     20      6 68665

Overall Statistics

    Accuracy : 0.9995
    95% CI : (0.9993, 0.9997)
    No Information Rate : 0.9825
    P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.9857
    Mcnemar's Test P-Value : NA

Statistics by Class:

          Class: Error1 Class: Error2 Class: Error3 Class: OK
Sensitivity      0.99273   0.3333333   0.934783   1.0000
Specificity      1.00000   1.0000000   1.000000   0.9722
Pos Pred Value   1.00000   1.0000000   1.000000   0.9995
Neg Pred Value   0.99988   0.9997138   0.999914   1.0000
Prevalence       0.01575   0.0004293   0.001316   0.9825
Detection Rate   0.01564   0.0001431   0.001231   0.9825
Detection Prevalence 0.01564   0.0001431   0.001231   0.9830
Balanced Accuracy 0.99637   0.6666667   0.967391   0.9861
```

圖 39 計算後的數值

#### 4.5.4 匯入測試資料

1. 放入別的資料測試 mod1

```
TestData <-read.csv("RawData/TestData.csv", header = TRUE, sep = ",")
```

2. 測試資料

```
z1 <- subset(TestData,select=-Judge)
```

```
z2 <- TestData$Judge
```

#### 4.5.5 計算測試資料的預測準確度

1. 將模型擲回目的:確認模型準確度

```
M_result <- predict(mod1,z1)
```

2. 預測測試資料的結果，此時可以用此資料來確認預測準確度

```
summary(M_result)
```

```
> summary(M_result)
error1 error3 error4    OK
  139      6     128 36591
```

圖 40 測試資料的統計狀況

3. 將結果與 Label 做成混亂矩陣

```
table(M_result,z2)
```

| M_result | error1 | error3 | error4 | OK    |
|----------|--------|--------|--------|-------|
| error1   | 139    | 0      | 0      | 0     |
| error3   | 0      | 6      | 0      | 0     |
| error4   | 0      | 2      | 81     | 45    |
| OK       | 0      | 37     | 7      | 36547 |

圖 41 測試資料的統計狀況

4. 計算混亂矩陣

```
zz <- confusionMatrix(table(M_result,z2))
```

5. 察看計算混亂矩陣後的結果，顯示預測資料有 99.75% 預測準確度。

```
zz
```

| M_result                        | error1        | error3        | error4        | OK        |
|---------------------------------|---------------|---------------|---------------|-----------|
| error1                          | 139           | 0             | 0             | 0         |
| error3                          | 0             | 6             | 0             | 0         |
| error4                          | 0             | 2             | 81            | 45        |
| OK                              | 0             | 37            | 7             | 36547     |
| Overall Statistics              |               |               |               |           |
| Accuracy : 0.9975               |               |               |               |           |
| 95% CI : (0.997, 0.998)         |               |               |               |           |
| No Information Rate : 0.9926    |               |               |               |           |
| P-Value [Acc > NIR] : < 2.2e-16 |               |               |               |           |
| Kappa : 0.8321                  |               |               |               |           |
| McNemar's Test P-Value : NA     |               |               |               |           |
| Statistics by Class:            |               |               |               |           |
|                                 | Class: error1 | Class: error3 | Class: error4 | Class: OK |
| Sensitivity                     | 1.000000      | 0.1333333     | 0.920455      | 0.9988    |
| Specificity                     | 1.000000      | 1.0000000     | 0.998722      | 0.8382    |
| Pos Pred Value                  | 1.000000      | 1.0000000     | 0.632812      | 0.9988    |
| Neg Pred Value                  | 1.000000      | 0.9989419     | 0.999809      | 0.8352    |
| Prevalence                      | 0.003771      | 0.0012207     | 0.002387      | 0.9926    |
| Detection Rate                  | 0.003771      | 0.0001628     | 0.002197      | 0.9914    |
| Detection Prevalence            | 0.003771      | 0.0001628     | 0.003472      | 0.9926    |
| Balanced Accuracy               | 1.000000      | 0.5666667     | 0.959588      | 0.9185    |

圖 42 測試資料的統計狀況

6. 將結果輸出至 server 上的 output 資料夾，方便日後查詢

```
write.table(M_result,file="Output/Svm_result.csv",sep=",")
```

#### 4.5.6 刪除訓練資料的最後 3 欄位，並建模

1. 接下來刪除最後面 3 欄，刪除第 34,35,36 欄

```
A1 <- x[,-c(34,35,36)]
```

2. 重新建立模型

```
M1_Model <- svm(A1,y)
```

3. 查看模型，並與訓練資料比對

```
summary(M1_Model)
```

```
M1_result=predict(M1_Model,A1)
```

```
summary(M1_result)
```

4. 建立混亂矩陣

```
table(M1_result,y)
```

```
confusionMatrix(table(M1_result,y))
```

```
> M1_result=predict(M1_Model,A1)
> summary(M1_result)
error1 error3 error4      OK
 1101     14     87  68686
> table(M1_result,y)
      y
M1_result error1 error3 error4      OK
error1    1101      0      0      0
error3      0     14      0      0
error4      0      0     87      0
OK        0     16      5  68665
> confusionMatrix(table(M1_result,y))
Confusion Matrix and Statistics

      y
M1_result error1 error3 error4      OK
error1    1101      0      0      0
error3      0     14      0      0
error4      0      0     87      0
OK        0     16      5  68665

Overall Statistics

  Accuracy : 0.9997
  95% CI  : (0.9995, 0.9998)
  No Information Rate : 0.9825
  P-Value [Acc > NIR] : < 2.2e-16

  Kappa : 0.9912
  Mcnemar's Test P-Value : NA

Statistics by Class:

   Class: error1 Class: error3 Class: error4 Class: OK
Sensitivity                           1.00000   0.4666667   0.945652   1.0000
Specificity                            1.00000   1.0000000   1.000000   0.9828
Pos Pred Value                         1.00000   1.0000000   1.000000   0.9997
Neg Pred Value                         1.00000   0.9997710   0.999928   1.0000
Prevalence                             0.01575   0.0004293   0.001316   0.9825
Detection Rate                          0.01575   0.0002003   0.001245   0.9825
Detection Prevalence                  0.01575   0.0002003   0.001245   0.9828
Balanced Accuracy                      1.00000   0.7333333   0.972826   0.9914
```

圖 43 訓練資料的統計狀況

#### 4.5.7 刪除測試資料的最後三欄，並計算模型準確度

- 匯入測試資料，刪除 34,35,36 欄

```
z5 <- z1[,-c(34,35,36)]
```

- 比對測試資料

```
M2_result=predict(M1_Model,z5)
```

- 比對後的結果

```
summary(M2_result)
```

- 建立混亂矩陣並計算模型準確度

```
table(M2_result,z2)
```

```
confusionMatrix(table(M2_result,z2))
```

```
> z5 <- z1[,-c(34,35,36)]
> M2_result=predict(M1_Model,z5)
> summary(M2_result)
error1 error3 error4    OK
 139     8    148  36569
> table(M2_result,z2)
      z2
M2_result error1 error3 error4    OK
error1    139     0     0     0
error3     0     8     0     0
error4     0     3    81    64
OK        0    34     7 36528
> confusionMatrix(table(M2_result,z2))
Confusion Matrix and Statistics

      z2
M2_result error1 error3 error4    OK
error1    139     0     0     0
error3     0     8     0     0
error4     0     3    81    64
OK        0    34     7 36528

Overall Statistics:

    Accuracy : 0.9971
    95% CI : (0.9965, 0.9976)
    No Information Rate : 0.9926
    P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.8085
    Mcnemar's Test P-Value : NA

Statistics by Class:

   Class: error1 Class: error3 Class: error4 Class: OK
Sensitivity                         1.000000   0.177778   0.920455   0.9983
Specificity                          1.000000   1.000000   0.998178   0.8493
Pos Pred Value                      1.000000   1.000000   0.547297   0.9989
Neg Pred Value                      1.000000   0.998996   0.999869   0.7831
Prevalence                           0.003771   0.001221   0.002387   0.9926
Detection Rate                      0.003771   0.000217   0.002197   0.9909
Detection Prevalence                0.003771   0.000217   0.004015   0.9920
Balanced Accuracy                   1.000000   0.588889   0.959316   0.9238
```

圖 44 測試資料的統計狀況

#### 4.5.8 大數據模型建立，並刪除 A 槽所有資料

- 依照上面資料清理的方法，接下來放入產品生產日期為 2017/04/29~2017/04/30，221 個原始資料合併的 39.9 萬筆資料，作為 Model。如圖 45

| 名稱                       | 修改日期               | 類型   | 大小     |
|--------------------------|--------------------|------|--------|
| _170429214634_AB_189.txt | 2017/4/29 下午 09:52 | 文字文件 | 464 KB |
| _170429215308_AB_190.txt | 2017/4/29 下午 09:59 | 文字文件 | 464 KB |
| _170429220018_AB_191.txt | 2017/4/29 下午 10:07 | 文字文件 | 468 KB |
| _170429220758_AB_192.txt | 2017/4/29 下午 10:14 | 文字文件 | 465 KB |
| _170429221440_AB_193.txt | 2017/4/29 下午 10:21 | 文字文件 | 465 KB |
| _170429222127_AB_194.txt | 2017/4/29 下午 10:27 | 文字文件 | 465 KB |
| _170429222818_AB_195.txt | 2017/4/29 下午 10:34 | 文字文件 | 465 KB |
| _170429223459_AB_196.txt | 2017/4/29 下午 10:41 | 文字文件 | 465 KB |
| _170429224157_AB_197.txt | 2017/4/29 下午 10:48 | 文字文件 | 465 KB |
| _170429224913_AB_198.txt | 2017/4/29 下午 10:56 | 文字文件 | 468 KB |
| _170429225642_AB_199.txt | 2017/4/29 下午 11:02 | 文字文件 | 466 KB |
| _170429230321_AB_200.txt | 2017/4/29 下午 11:09 | 文字文件 | 465 KB |
| _170429231008_AB_201.txt | 2017/4/29 下午 11:16 | 文字文件 | 465 KB |
| _170429231700_AB_202.txt | 2017/4/29 下午 11:23 | 文字文件 | 465 KB |
| _170429232344_AB_203.txt | 2017/4/29 下午 11:30 | 文字文件 | 465 KB |
| _170429233027_AB_204.txt | 2017/4/29 下午 11:37 | 文字文件 | 465 KB |
| _170429233739_AB_205.txt | 2017/4/29 下午 11:44 | 文字文件 | 467 KB |
| _170429234516_AB_206.txt | 2017/4/29 下午 11:51 | 文字文件 | 465 KB |
| _170429235159_AB_207.txt | 2017/4/29 下午 11:58 | 文字文件 | 466 KB |
| _170429235836_AB_208.txt | 2017/4/30 上午 12:04 | 文字文件 | 465 KB |
| _170430000509_AB_209.txt | 2017/4/30 上午 12:11 | 文字文件 | 465 KB |
| _170430001147_AB_210.txt | 2017/4/30 上午 12:18 | 文字文件 | 465 KB |
| 170430001826_AB_211.txt  | 2017/4/30 上午 12:24 | 文字文件 | 465 KB |

圖 45 39 萬筆訓練資料的原始檔

```
> summary(TestData$Judge)
error1 error3 error4      OK
 2139     283     617 392481
```

圖 46 39 萬筆訓練資料的統計

- 匯入訓練資料，並刪除 A 槽的所有資料

```
TestData <- read.csv("RawData/Train1.csv", header = TRUE, sep = ",")  
A2 <- TestData[,-c(17:37)]
```

- 建模，因為放入的資料的多寡，會需要時間的處理

```
M2_Model <- svm(Judge ~., data=A2)
```

- 與訓練資料作比對，並查看模型預測準確度

```
M2_result <- predict(M2_Model,A2)
confusionMatrix(table(M2_result,A2$Judge))
```

```
> confusionMatrix(table(M2_result,A2$Judge))
Confusion Matrix and Statistics

M2_result error1 error3 error4      OK
error1     1818      0     94      3
error3      0    111      2      0
error4      57      0   328     13
OK        264    172   193 392465

Overall Statistics

Accuracy : 0.998
95% CI : (0.9978, 0.9981)
No Information Rate : 0.9923
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8532
McNemar's Test P-Value : NA

Statistics by Class:

          Class: error1 Class: error3 Class: error4 Class: OK
Sensitivity       0.849930    0.3922261    0.5316045  1.0000
Specificity       0.999753    0.9999949    0.9998227    0.7930
Pos Pred Value    0.949347    0.9823009    0.8241206    0.9984
Neg Pred Value    0.999184    0.9995650    0.9992686    0.9934
Prevalence        0.005408    0.0007155    0.0015600    0.9923
Detection Rate    0.004596    0.0002806    0.0008293    0.9923
Detection Prevalence 0.004842    0.0002857    0.0010063    0.9939
Balanced Accuracy 0.924842    0.6961105    0.7657136    0.8965
```

圖 47 預測準確度

#### 4.5.9 匯入小筆數的測試資料，並計算模型準確度

- 匯入前，查看測試資料統計

```
summary(TestData$Judge)
summary(M2_result)
```

```
> summary(M2_result)
error1 error3 error4      OK
  1915    113    398 393094
> summary(TestData$Judge)
error1 error3 error4      OK
  2139    283    617 392481
```

圖 48 39 萬筆訓練資料的統計

- 將測試資料中 A 欄資料刪除

```
A3 <- TrainData[,-c(17:37)]
```

- 將測試資料用的 69888 筆資料放入剛剛建立的模型

```
TResult <- predict(M2_Model,A3)
```

#### 4. 預測準確度

```
confusionMatrix(table(TResult,A3$Judge))
```

```
> confusionMatrix(table(TResult,A3$Judge))
Confusion Matrix and Statistics

TResult  error1 error3 error4    OK
error1     859      0     17      0
error3      0      9      0      0
error4      9      0     54      0
OK        233     21     21  68665

Overall Statistics

    Accuracy : 0.9957
    95% CI  : (0.9952, 0.9962)
    No Information Rate : 0.9825
    P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.8594
    Mcnemar's Test P-Value : NA

Statistics by Class:

          Class: error1 Class: error3 Class: error4 Class: OK
Sensitivity           0.78020   0.3000000   0.5869565   1.0000
Specificity            0.99975   1.0000000   0.9998711   0.7751
Pos Pred Value         0.98059   1.0000000   0.8571429   0.9960
Neg Pred Value         0.99649   0.9996995   0.9994558   1.0000
Prevalence              0.01575   0.0004293   0.0013164   0.9825
Detection Rate          0.01229   0.0001288   0.0007727   0.9825
Detection Prevalence    0.01253   0.0001288   0.0009014   0.9864
Balanced Accuracy       0.88998   0.6500000   0.7934138   0.8876
```

圖 49 69888 筆資料的預測情形

#### 5. 查看預測的資料詳細統計狀況

```
summary(TResult)
```

```
summary(TrainData$Judge)
```

```
> summary(TResult)
error1 error3 error4    OK
  876     9    63  68940
> summary(TrainData$Judge)
error1 error3 error4    OK
 1101    30    92  68665
> summary(M3,na.rm=TRUE)
```

圖 50 69888 筆資料的預測情形

#### 4.5.10 匯入建立模型(4/29)後，機台生產的實際資料

1. 因前面的實驗透過 2017/04/29~2017/04/30 的 39 萬筆資料建立了模型，因此我們可以利用建構好的模型，並載入 2017/05/27 的資料共 92 個檔案 70656 筆資料是否也能正常地預測。如圖 51

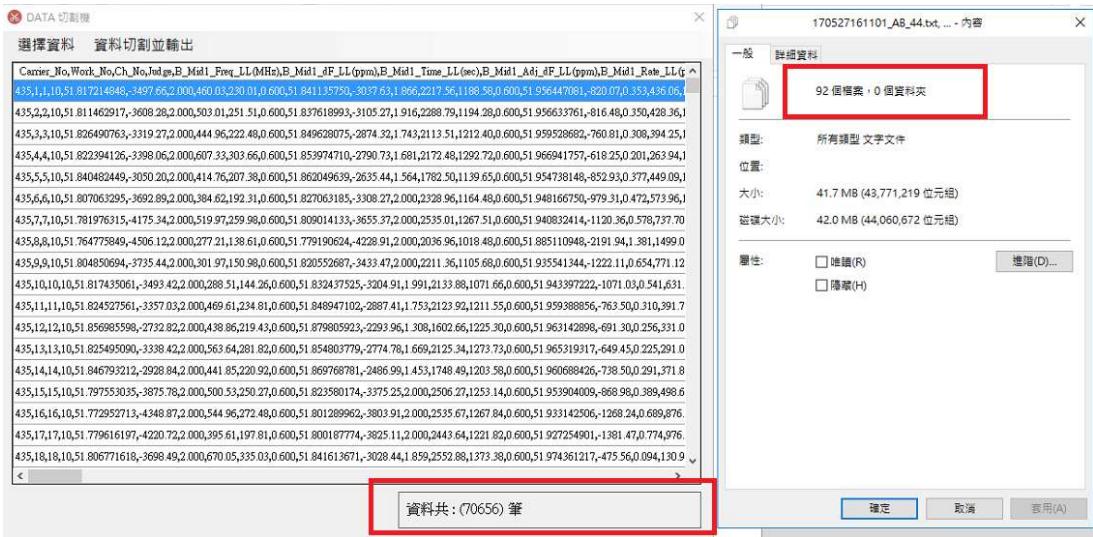


圖 51 92 個檔案共 70656 筆資料

2. 匯入後先算出真實資料的良率約為 99.24%，如圖 52

```
> summary(Data20170527$Judge)
error1 error3 error4      OK
  419     46    70 70121
```

圖 52 實際資料良率

3. 與 4/29 的模型作比對，並算出模型準確度。

```
> A0527 <- Data20170527[,-c(17:37)]
> A0527_result <- predict(M2_Model,A0527)
> confusionMatrix(table(A0527_result,Data20170527$Judge))
Confusion Matrix and Statistics

A0527_result error1 error3 error4      OK
  error1    374      0     16      0
  error3      0      7      0      0
  error4      9      0     23      1
  OK        36     39     31 70120

Overall Statistics

    Accuracy : 0.9981
    95% CI : (0.9978, 0.9984)
    No Information Rate : 0.9924
    P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.8624
    Mcnemar's Test P-Value : NA

Statistics by Class:

          Class: error1 Class: error3 Class: error4 Class: OK
Sensitivity           0.892601   1.522e-01   0.3285714   1.0000
Specificity            0.999772   1.000e+00   0.9998583   0.8019
Pos Pred Value         0.958974   1.000e+00   0.6969697   0.9985
Neg Pred Value         0.999360   9.994e-01   0.9993345   0.9977
Prevalence              0.005930   6.510e-04   0.0009907   0.9924
Detection Rate          0.005293   9.907e-05   0.0003255   0.9924
Detection Prevalence    0.005520   9.907e-05   0.0004671   0.9939
Balanced Accuracy       0.946187   5.761e-01   0.6642149   0.9009
> summary(A0527_result)
error1 error3 error4      OK
  390      7     33 70226
```

圖 53 計算後的資料

#### 4.5.11 SVM 的分佈圖

1. 透過簡單的指令畫出此 SVM 模型的分布狀態，如圖 54、55。

```
plot(M2_Model,Data20170527,B_Mid1_Time_LL.sec.~  
B_Fine_Time_LL.sec.,svSymbol = 1, dataSymbol = 2, symbolPalette =  
rainbow(4),color.palette = terrain.colors)
```

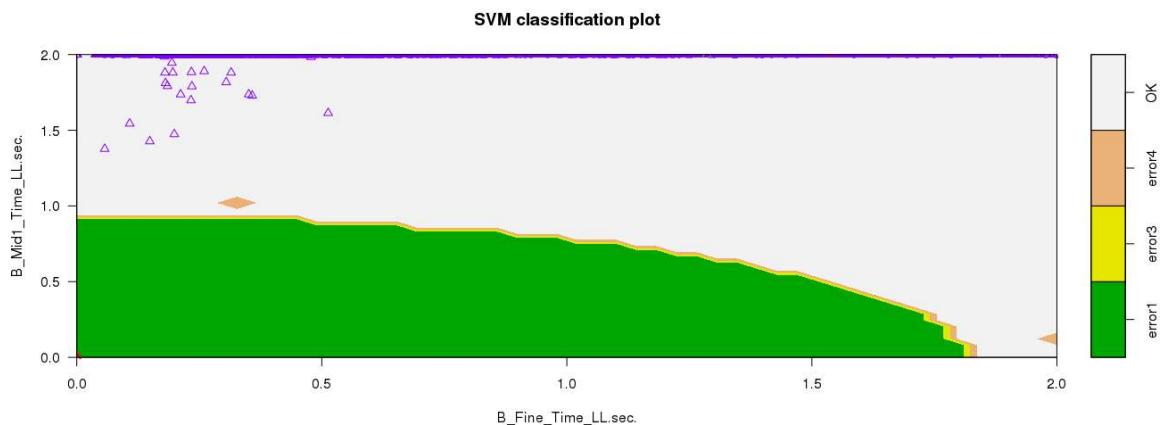


圖 54 資料散佈圖

```
plot(M2_Model,Data20170527,B_Mid1_dF_LL.ppm.~  
B_Mid1_Time_LL.sec.,svSymbol = 1, dataSymbol = 2, symbolPalette =  
rainbow(4),color.palette = terrain.colors)
```

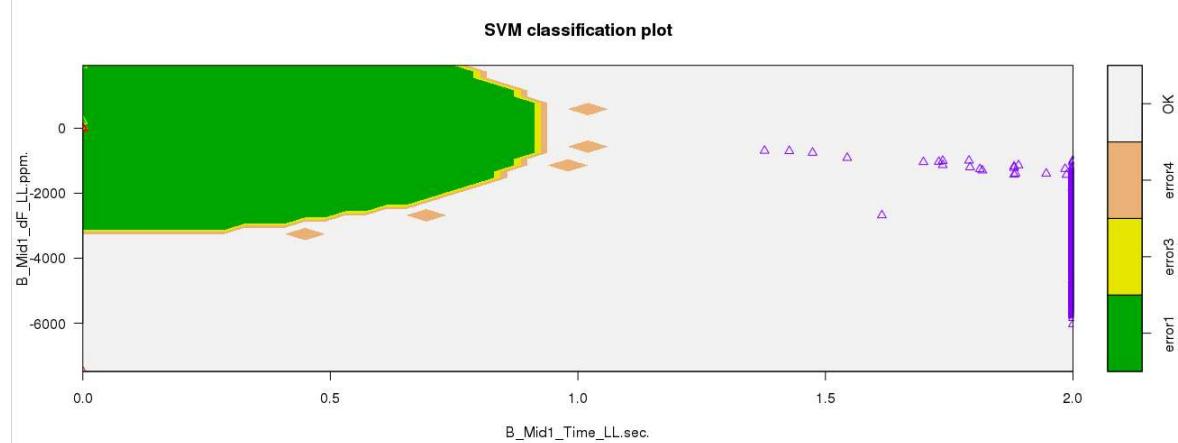


圖 55 資料散佈圖

#### 4.5.12 小結

原本訓練資料的良率為 98.25%，使用測試資料預測資料良率為 98.64%，因此可得知若建立模型的數據越多，則預測準確率越高，反之則越低。

另外因此實驗刪除了所有 A 槽的資料來做預測，且預測準確度 ( $AUC=0.9957 < 1$ )，故此實驗建立的模型可以在機台做完 B 槽的蝕刻之後，預測出將會遇到的不良情況，若 error1 的預測過多時可預先調整機台的探針，以免測試不到或空測，若 error3 的預測較多時可以先行調整機台內的參數讓產品的頻率不至於過高，若 error4 的預測較多時可以先行調整機台內的參數讓產品的頻率不至於過低。以上皆是得到預測資料時可以先行調整的項目。

且將訓練模型套用至 1 個月後的機台生產資料，得到的 AUC 為 0.9981 < 1，因此次模型的預測為準確的，且預測的的良率約為 99.39% 較高於正確資料的預測，但因資料良率無相差太多，故無過度訓練(overfitting)的問題，因此所建構出來的模型針對此機台來做預測會有相當高的可靠度。



## 第五章 結論

本實驗建置在生產力 4.0-基礎建設上，並經由這些網路服務及虛擬化，且利用生產力 4.0-資料收集系統，收集機台資料，並放置於基礎建設的檔案伺服器上，讓管理人員方便彙整資料，分析清理資料，並傳入生產力 4.0-異常分析伺服器中，且透過 Hadoop+Spark+Rstudio server 工具的幫助，減輕人員對於統計分析的不熟悉與排斥感，透過簡單的 R 語言指令可以讓分析人員在分析資料時不用再 Coding 的那麼辛苦，只要將該模型所需的資料格式及參數填入即可，且還能使用多種圖形建置給上層主管查看，非常的好用。

本研究以水晶震盪器產業的 Milling 機台為研究對象，運用大數據的分類工具，SVM 作為分類工具，在經過一連串的數據，透過資料檢視、資料清理、整理後篩選剔除較無關的資料後，放入機器學習的模型中，並且找尋出規則，建立有效的模型，在此期間因模型準確度(AUC)約為 99.97%，因此可以考慮刪減特徵向量，來做到預測效果，最後的實驗為只抓取 B 槽的資料，在產品尚未置入 A 槽時可以做分類預測，且預測準確度(AUC)約為 99%表示，此模型的預測非常的有效果，足以讓機台在未置入 A 槽時，先行判斷良率，若良率不高時則可以查看機台 B 槽是否有異常，讓維修人員先行評估是否該預先作機台保養，更換耗材，以便讓機台能不間斷的順利運作，達到節省時間成本的效果。

參考文獻：

(一)書籍

1. 林大貴,2015,hadoop+spark 大數據巨量分析與機器學習整合開發實戰,博碩
2. 簡禎富、許嘉裕,2014,資料挖礦與大數據分析,前程
3. 鳥哥,2016,鳥哥的 Linux 私房菜:基礎學習篇,基峰
4. 王家林,2016,大數據分析處理：Spark 技術、應用與性能優質化,上奇資訊
5. 陳景祥,2010,R 軟體：應用統計方法 (修訂版),東華
6. 方匡南/朱建平/姜葉飛,2015,R 語言資料分析活用範例詳解 ,基峰
7. 丘祐瑋,2016,機器學習與 R 語言實戰,機械工業出版社

(二)學位論文

1. 吳佩馨,2010,機台失效預警辨識模型-拋光機台為例,碩士論文,國立清華大學  
工業工程學系
2. 謝惠如,2009,拋光製程破片關鍵影響因子辨識模型,碩士論文,國立清華大學  
工業工程學系
3. 吳曼穗,2013,退貨商品與製程關聯性分析,碩士論文,明志科技大學工業工程  
與管理研究所
4. 宋堯正,2013, 資料採礦分析法於解析客訴不良品之應用 - 以 TFT-LCD 製  
造廠某 C 公司為例,碩士論文,國立中央大學工業管理研究所在職專班

(三)網頁

1. Apache Spark Ecosystem ,Available at <https://databricks.com/spark/about>
2. Hadoop,Available at <http://hadoop.apache.org/>
3. spark,Available at <https://spark.apache.org/>
4. Rstudio,Available at <https://www.rstudio.com/>
5. 圖解機器學習,取自於 <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
6. Rstudio 介紹,取自於  
[https://joe11051105.gitbooks.io/r\\_basic/content/environment\\_settings/RStudio\\_introduction.html](https://joe11051105.gitbooks.io/r_basic/content/environment_settings/RStudio_introduction.html)
7. SparR Doc,Available at <http://spark.apache.org/docs/latest/sparkr.html>
8. R SVM library(e1071),Available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

## 附件-安裝指令檔

---

### 安裝 CentOS

---

01. 設定英文語系(為了 home 的資料夾名稱)
  02. 設定時間
  03. 設定輸入法，設定 ctrl+shift 切換輸入法
  04. 設定軟體，含有 GUI 的伺服器+相容性函式庫+開發工具
  05. 設定硬碟，讓他自動安裝
  06. 設定網路為 eth0
  07. 設定 root 密碼 kuas，建立使用者 kuser 密碼 kuas
- 

### 安裝及設定 Hadoop Mult-Node Cluster

---

01. 為了省事，避免意外的情況，關閉 SELinux (Security Linux ) 和 iptables  
`#setenforce 0`
02. 設定 reboot 後自動關閉 SELinux  
`#vi /etc/selinux/config`  
`SELINUX=disabled`
03. 關閉防火牆  
`#systemctl stop firewalld`  
`#systemctl disable firewalld`
04. 安裝 oracle jdk,至以下網站下載 java rpm 檔  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>
05. 安裝  
`#sudo yum install jdk-8u121-linux-x64.rpm`
06. 設定 oracle java 當作預設值  
`#sudo alternatives --config java`  
輸入完後,選剛剛安裝的版本
07. 建議把 openjdk 移除，因為 hadoop 只要用 oracle jdk 就好，避免後續變數的錯亂。  
移除後上面的 `alternatives --config java` 就會只剩下一個選項。

找套件名稱

```
#rpm -qa | grep openjdk  
#sudo yum remove [填入要移除的套件名稱] -y  
找看看是否還有先前的套件  
#java -version
```

#### 08. 修改~/.bashrc

```
#vim ~/.bashrc  
  
#Hadoop Variables  
export JAVA_HOME=/usr/java/jdk1.8.0_121  
export HADOOP_HOME=/usr/local/hadoop  
export PATH=$PATH:$HADOOP_HOME/bin  
export PATH=$PATH:$HADOOP_HOME/sbin  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export  
  
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"  
export  
  
JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native:$JAVA_LIBRARY_PATH  
#Hadoop Variables  
  
#spark variable  
export SPARK_HOME=/usr/local/spark  
export PATH=$PATH:$SPARK_HOME/bin  
#spark variable  
  
#source ~/.bashrc
```

#### 09. 修改 NameNode 設定

```
#vim /usr/local/hadoop/etc/hadoop/master  
master  
#vim /usr/local/hadoop/etc/hadoop/core-site.xml
```

```

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
#vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>
      file:/usr/local/hadoop/hadoop_data/hdfs/namenode</value>
    </property>
    <property>
      <name>dfs.datanode.data.dir</name>
      <value>
        file:/usr/local/hadoop/hadoop_data/hdfs/datanode</value>
      </property>
      <property>
        <name>dfs.permissions</name>
        <value>false</value>
      </property>
    </configuration>

```

#### 10. JobTracker 設定 ( on master ) — 設定 JobTracker 的名稱與埠號

```

#cd /usr/local/hadoop/etc/hadoop
#cp mapred-site.xml.template mapred-site.xml
#vim /usr/local/hadoop/etc/hadoop/mapred-site.xml
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>master:54311</value>
  </property>
</configuration>

```

```
#vim gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.resource-
tracker.address</name>
    <value>master:8025</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>master:8030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>master:8050</value>
  </property>
</configuration>
```

## 11. DataNode 和 TaskTracker 設定

```
#vim /usr/local/hadoop/etc/hadoop/slaves
master
slave1
slave2
```

## 12. 更改"各別"虛擬機器的 hostname，及加入 hosts

```
#hostnamectl set-hostname master
#hostnamectl set-hostname slave1
#hostnamectl set-hostname slave2
#vi /etc/hosts
192.168.50.21      master
```

```
192.168.50.22    slave1  
192.168.50.23    slave2
```

若網卡無法在開機時啟用可以

```
#cd /etc/sysconfig/network-scripts/  
#vi ifcfg-enp0s8
```

將 noboot 設為 yes

### 13. ssh 無密碼連線設定 (在 master 上)

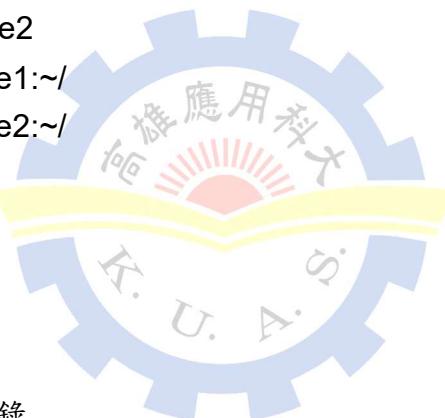
```
#cd ~/.ssh  
#su - kuser  
#ssh-keygen -t rsa  
#cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
#chmod 400 ~/.ssh/authorized_keys
```

複製至 slave1,slave2

```
#scp -r ~/.ssh slave1:~/  
#scp -r ~/.ssh slave2:~/
```

測試連線至 slave1

```
#ssh slave1  
#exit
```



### 14. 分別設定 HDFS 目錄

```
on slave1,slave2  
#sudo rm -rf /usr/local/hadoop/hadoop_data/hdfs  
#sudo chown -R kuser:kuser /usr/local/hadoop  
#mkdir -p /usr/local/hadoop/hadoop_data/hdfs/datanode
```

on master

```
#sudo rm -rf /usr/local/hadoop/hadoop_data/hdfs  
#sudo chown -R kuser:kuser /usr/local/hadoop  
#mkdir -p /usr/local/hadoop/hadoop_data/hdfs/namenode  
#mkdir -p /usr/local/hadoop/hadoop_data/hdfs/datanode
```

### 15. 格式化 datanode,namenode

```
on slave1,slave2  
#cd /usr/local/hadoop
```

```
#hadoop datanode -format  
  
on master  
#cd /usr/local/hadoop  
#hadoop namenode -format  
#hadoop datanode -format
```

16. 啟用 Hadoop 完全分散模式

```
#start-all.sh
```

17. 查看是否有成功

```
#jps  
#hadoop dfsadmin -report
```

or FireFox browser: <http://master:8088>

or FireFox browser: <http://master:50070>



=====

CentOS 安裝 Spark 2.1.0 Standalone Cluster

=====

01. 查 hadoop 是否正在執行中

```
#jps
```

02. 請依 hadoop 版本做查找，書上使用版本為 2.6，spark 版本為 1.4.0  
至 <https://spark.apache.org/downloads.html>

03. 解壓縮

```
#tar zxf spark-1.4.0-bin-hadoop2.6.tgzta
```

04. 移動目錄

```
#sudo mv spark-1.4.0-bin-hadoop2.6 /usr/local/spark
```

05. 編輯 bashrc 檔案

```
#vim/.bashrc
```

```
#spark variable
```

```
export SPARK_HOME=/usr/local/spark
```

```
export PATH=$PATH:$SPARK_HOME/bin
```

```
#spark variable
```

06. 建立 Spark 執行環境

```
#cd /usr/local/spark/spark-2.1.0-bin-hadoop2.6/conf
```

```
#cp spark-env.sh.template spark-env.sh
```

```
#sudo vim spark-env.sh
```

```
export SPARK_MASTER_IP=master # Setting master's IP
```

```
export SPARK_WORKER_CORES=1 # Setting each worker's CPU  
number
```

```
export SPARK_WORKER_MEMORY=400m # Setting each worker's  
memory size : 400 MB
```

```
export SPARK_WORKER_INSTANCES=2 # Setting worker's  
instances : 2 workers on this node
```

07. 在 master 上，編輯 slaves 檔案：

```
#cd /usr/local/spark/conf/
```

```
#cp slaves.template slaves
```

```
#sudo vim slaves
```

```
# A Spark Worker will be started on each of the machines listed below.
```

```
master
```

```
slave1
```

```
slave2
```

08. 在 master 上，複製 /usr/local/spark 中的檔案至 slave1,slave2

```
#cd ~/  
#ssh slave1  
#sudo mkdir /usr/local/spark  
#sudo chown kuser:kuser /usr/local/spark # 變更擁有權為 kuser  
#exit  
#sudo scp -r /usr/local/spark kuser@slave1:/usr/local #複製檔案至 slave1
```

```
#cd ~/  
#ssh slave2  
#sudo mkdir /usr/local/spark  
#sudo chown kuser:kuser /usr/local/spark # 變更擁有權為 kuser  
#exit  
#sudo scp -r /usr/local/spark kuser@slave2:/usr/local #複製檔案至 slave2
```

09. 使 bashrc 生效

```
#source ~/.bashrc
```

10. 測試是否安裝成功

```
#spark-shell
```



11. 設定 spark-shell 的顯示訊息

```
#cd /usr/local/spark/conf  
#cp log4j.properties.template log4j.properties
```

12. 編輯 log4j.properties

```
#vim log4j.properties
```

將第二行中 log4j.rootCategory=INFO,console INFO 改為 WARN,即可使互動介面的訊息少很多

13. 啟動 Spark standalone cluster mode

```
#/usr/local/spark/spark-2.1.0-bin-hadoop2.6/sbin/start-all.sh
```

14. 查看是否有成功

```
#jps
```

or FireFox 瀏覽器上，輸入 <http://master:8080>

```
=====
```

CentOS 安裝 R

```
=====
```

```
#sudo yum -y update
#sudo yum -y install yum-cron
#sudo yum -y install epel-release
#sudo yum -y install R
#cd /home/kuser/Download
#wget https://download2.rstudio.org/rstudio-server-rhel-1.0.136-x86_64.rpm
#sudo yum install --nogpgcheck rstudio-server-rhel-1.0.136-x86_64.rpm
在 FireFox，輸入：http://master:8787/
```

