通識教育與跨域研究第 20 期
民國一０八年十二月
General Education and
Trandisciplinary Research

# 運用機器學習自動批改中文作文之方法研究

**鄧秀梅**
環球科技大學副教授

## 摘要

當代這個世界，大數據分析、AI 人工智慧正大行其道，從製作、行銷、醫學、娛樂……等等，每一種行業、每一項事物，幾乎都有人嘗試用大數據分析將人工作業電腦化與自動化；那麼若將大數據分析應用在批改中文作文，不知是否行得通？假使學生的中文作文能夠透過演算模式自動分級給分，讓國文老師從改作文當中解脫出來，這對國文老師而言，將會是一項多麼大的福利！

為達此目的，本論文嘗試使用機器學習來從事這項研究，選擇的軟體程式為 R 語言。第一步是斷詞，把文章字串分割成詞，再標記詞性，從中提取與文章好壞相關的元素，設為特徵變數，把非結構化的文字轉換為結構化的資料。這裡所使用的套件程式為 jiebaR。第二步則是選擇特徵變數，在此本文揀選八個自變數，一個依變數，中間又添加分群變數，最後又補上「優雅詞彙」和「切題詞彙」兩個新變數，總共十二個變數來演算預測。

第三步選擇演算法，本文選擇六種演算法，比較其中的預測精準度。再一一搭配九個變數、十個變數與十二變數進行訓練、預測，看看何種演算法配合多少個變數預測效果最佳。

**關鍵字：機器學習，R 語言，中文作文，自動批改，非結構化資料**

## 壹、 前言

當今是資訊科技橫掃全球的時代，無論是機器學習、大數據分析或資料探勘，皆是社會的主流趨勢，幾乎各個產業，醫療、製造、金融等各行各業，只要與資訊科技掛勾連結，似乎就能起死回生，創造新的生機。然而傳統的文史哲領域，此類以文字論說為主的學科，是否也能與資訊科技沾上邊，分享電腦高性能所帶來的便利？固然從事文史哲領域的研究者，自可以專業性質差異、精神境界不同為由摒除資訊科技的入侵，然而一味封閉不與外界通氣，所造成的結果恐怕是日益被邊緣化，社會愈不重視這種缺乏有形產能的人文學。

有鑑於此，筆者嘗試人文與科技的結合能用在什麼地方。初步研究，若將機器學習、大數據分析應用在批改中文作文，或許是一種合理的結合，果真成功研究出讓學生的作文能夠透過演算模式自動分級給分的方法，這對國文老師而言，將會是一項多麼大的福利！是以本人擬欲從這方面著手，研究如何引進機器學習如何自動批改作文，當做筆者結合人文與科技研究的第一步。

## 貳、 機器學習的發展歷程

所謂「機器學習」（Machine Learing）是一門人工智慧的科學，它是實現人工智慧的一個途徑，其理論主要是設計和分析一些讓電腦可以自動「學習」的演算法。機器學習演算法是一類從資料中自動分析獲得規律，並利用規律對未知資料進行預測的演算法。所涉及的領域包括機率論、統計學、逼近論、凸分析（Convex analysis）、計算複雜性理論等多門學科。[1]

簡單而言，機器學習是統計學的延伸。傳統的統計學依靠的是母體與樣本的契合度，基於低效度的計算功能，與資訊取得不易和資訊量的短少，統計學家想要推算某項標的，只能從有限的資料蒐集群當中去測試，而此種蒐集來的資料樣本，也許僅佔所有母體的千分之一或萬分之一不到，因之為求推算精準，統計學家唯有在隨機抽樣方法下功夫。使用隨機抽樣，確能降低蒐集資料的成本，又能以推斷的方式，準確描繪母體，可是這畢竟是一種捷徑，是在無法全面蒐集完整資料下的不得已之辦法，而且隨機抽樣本身就帶著缺陷，它的成功取決於抽樣時究竟是否達到真正的隨機，蓋真正的隨機難度相當高，一旦抽樣的方式有偏差，則結果可能就差之千里。[2]

然而隨著網際網路的發達普及，大家隨時都在網路上公開分享各種資料日益增多，無論文字、影像或聲音的取得都不是難事，這便使得今

[1] 參見維基百科「機器學習」，網址：https://zh.wikipedia.org/wiki/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0

[2] 麥爾荀伯格、庫基耶，2013，**大數據**，台北：天下遠見，頁 36-37。

人欲索取幾近於母體數量的樣本變得不困難，既然取得母體不是一樁難事，自然可以捨棄隨機抽樣，直接以母體之數量推算統計。不過母體的數量想必十分龐大，動輒上億、兆的巨量資料，電腦的計算效能應付得來嗎？事實上自 1970 年起，Intel 公司便開發了微處理器，1980 年代處理器更縮小化，能夠處理的資料從 4bit（位元）→8bit→16bit→32bit 持續進化，這時的電腦稱之為第四代電腦，比起早期的電腦，其性能已經快上數百倍了。於處理器進步的同時，用來儲存資料的儲存裝置也與時俱進，1973 年 IBM 成功研發現代 HDD 的基礎，有了這種高性能又小型的儲存裝置，就能把它搭載在電腦上，而大量安裝這類的儲存裝置，電腦便可擁有數 G 的大容量，截至目前為止，電腦儲存裝置已高達數百 T 的容量了。[3]

既有高儲存裝置，又有巨量資料的取得來源，若缺乏演算這些巨量資料的數學程式，一切俱是枉然。其實在電腦性能升級的同時，程式演算法也是與日俱進，突破過往的限制。就以類神經網路為例，其原名是「人工神經網路」（Artificial Neural Network，ANN），簡稱神經網路（Neural Network，NN）或類神經網路，是一種模仿生物神經網路（動物的中樞神經系統，特別是大腦）的結構和功能的計算模型，用於對函式進行估計。[4]要重現大腦的神經網路，最重要的是如何呈現當資訊傳達給人腦神經突觸時的訊號變化。因此在設定輸入層的時候，便也須要給予加強或減弱訊號的功能，以便順利無誤的輸出。像這樣在特定的輸入層給予強化或減弱的動作稱為「加權計算」，依據要解決的問題調整適當的加權計算，即可解決各式各樣的問題。（三津村直貴，2018，頁 38）但是若要對全部的輸入層做加權計算，可是一件浩大的工程，所以必須加裝能夠自我學習「加權計算」的感知器[5]，讓人工神經細胞學習輸入層的加權計算，並且自我執行變更。然而早期的感知器只能解決「可被線性分割」的問題，若是多層次又混淆不單純的問題，感知器便失靈；早期的類神經網路即遇上這一類的困境，即使當時已有「多層感知器」的設想，但皆處於討論階段，電腦的性能也未能回應這樣的需求。（三津村直貴，2018，頁 52）直至「反向傳播法」（Backpropagation）的開發，才突破單層感知器的限制。再兼以「卷積神經網路」（Convolutional Neural

[3] 三津村直貴，2018，**圖解 AI 人工智慧大未來：關於人工智慧一定要懂的 96 件事**，台北：旗標科技，頁 62。

[4] 參見維基百科「人工神經網路」。網址：https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

[5] 感知器（Perceptron）是 Frank Rosenblatt 在 1957 年就職於康奈爾航空實驗室（Cornell Aeronautical Laboratory）時所發明的一種人工神經網路。它可以被視為一種最簡單形式的前饋神經網路，是一種二元線性分類器。也被稱為單層的人工神經網絡，以區別於較複雜的多層感知機。見維基百科「感知器」，網址：https://zh.wikipedia.org/wiki/%E6%84%9F%E7%9F%A5%E5%99%A8

Network）和「自動編碼器」（Autoencoder）技術的問世，催生了多層次類神經網路的機器學習系統。（三津村直貴，2018，頁 110）

以上僅就類神經網路演算法說明它的演化歷程，其他演算法的研發精進亦是如此。總之，諸多條件的配合，促使資訊科技高度發展，電腦不僅可以從事機械化作業，甚至亦具備人類智慧的程式，以精密複雜的演算法更快速、更高效率地解決人類的問題。若將此技術移至批改作文，效果會如何？以下即是筆者取機器學習方法用在批改中文作文的一次實驗內容。

## 參、 研究工具——選用 R 語言（R Language）

在眾多的新科技中，筆者選擇大數據語言與自己的文哲領域結合，希望藉由大數據的軟體研發一套由程式軟體自動批改中文作文。在眾多的大數據程式軟體中，本人選擇 R 語言作為本次實驗的工具，雖然近年有許多竄起的程式語言比諸 R 語言更熱門、更受歡迎，譬如 Java、Python、C、C++等，但是 R 畢竟有其不可取代的好處。對於一個非專業資訊工程背景的學者而言，R 能省略自己撰寫程式的困難步驟，可以直接使用專家寫好的套件程式，單是這一點便使得它受到許多非專業人士的青睞。

其次，R 是個完全免費的軟體，它的原始碼供人自由下載，可以在許多平台下執行，包括 Windows、UNIX（包涵 FreeBSD 與 Linux）以及 MacOS 等等。[6]而 R 可讀取的資料格式包含 excel、txt、csv、spss，甚至還能與 SQL 資料庫結合，這些標示了 R 運用範圍之廣幾乎是無遠弗屆。此外，它的超強的建模能力也是其他程式難以企及的[7]。

套一句 Hal Ronald Varian 的話：

> R 的最美之處在於，你能夠透過修改很多高手預先撰寫好的套件的程式，解決你想解決的各種問題。因此，事實上，使用 R，你已經站在了巨人的肩膀上。（*The great beauty of R is that you can modify it to do all sorts of things. And you have a lot of prepackaged stuff that's already available, so you're standing on the shoulders of giants.*）[8]

針對本文主題所進行的研究歷程大致可分成兩大步驟，一是資料前處理，二是資料分析。資料前處理的部份，筆者必須將每一篇作文擷取出優劣好壞的特徵，且這種特徵須得以數據的方式呈現。欲達到此目的，首先是完成中文分詞的工作。運用電腦程式讀取文章，進而擷取或評量

---

[6] 參見維基百科「R 語言」。參考網址：ttps://zh.wikipedia.org/wiki/R%E8%AF%AD%E8%A8%80

[7] R 語言大多數的統計建模功能是以套件的形式提供，至今 R 網站上已有超過 7000 個套裝程式，涵蓋基礎統計學、社會學、經濟學、生態學、空間分析、系統發展分析、生物資訊學等諸多方面，皆配有完整的 PDF 說明檔，且版本會隨 R 新版本的發布得到更新。如是完善的巨大功能，使得 R 永遠在大數據語言中佔有一席之地。參見游皓麟，2017，**實戰 R 語言預測分析**，松崗資產管理，臺北，頁 1-22。

[8] 黃文、王正林編著，2016，**利用 R 語言打通大數據的經脈**，佳魁資訊，臺北，頁 0-2。

文章，這些多半應用在英文，鮮少用在中文，主因在於，英文的每個字詞（word）均以空白隔開，而中文字詞間則是緊密相連，沒有任何區隔標記，唯有句與句之間方以標點符號區隔。所以中文斷詞並不像英文可輕易的判斷出字詞間的不同。此外，中文的語意(Semantic)和句法(Syntactic)的基本單位是「詞」而非「字」，單一的字未必是語意分析的最小單位[9]，因此中文斷詞的技術相較於英文斷詞確實存在較高的困難度。完成斷詞步驟之後，才能進行詞性標記，此二步驟是自然語言處理中基礎且重要的一部份，舉凡資訊擷取、摘要製作及自動作文評分系統等，都需利用斷詞及詞性標記處理後的結果來進行下一步動作[10]。

R 語言中目前開發成熟的中文分詞套件有 rJava、Rwordseg、TM 以及 jiebaR，筆者採用的是 jiebaR 套件來進行文章的分詞，統計文章的詞彙數、各項詞性的數量和相同詞彙數、相異詞彙數等等，以作為將來資料分析的特徵變數。

當各種特徵變數統計出來之後，緊接著資料分析的套件，本文選用多種演算法，諸如類神經網絡、決策樹、線性迴歸、隨機森林等，均會一一嘗試，直到出現精準度最高的程式為止。

## 肆、 研究方法

### 一、界定評量作文的指標

批改作文是一件高度質化的工作，仰賴的是教師深厚的語文素養，而後去評量學生的作文是否切題、有無內涵深度、文句是否流暢通順、錯別字多寡……等，這些評量標準看似複雜多端，其實在一名有深厚語文素養的教師的腦海中，那也是一瞬間可完成的事。但是這些評量標準都難以量化，偏偏電腦批改作文依靠的是量化指標，研究第一步便遇上這等大難題！既然傳統評量作文的準則難以量化，就先擱置一旁，看看 R 語言能提供什麼評量項目。

前文提及本次實驗選用 R 當中的結巴（jiebaR）套件，結巴是近年異軍突起的一中文斷詞套件，它的強大與簡易的功能，遠勝過 Rwordseg、TM 等，此套件支持最大概率法（Maximum Probability），隱式馬爾科夫模型（Hidden Markov Model），索引模型（QuerySegment），混合模型（MixSegment）等四種分詞模式，同時還有詞性標識，關鍵詞抽取，文本 Simhash 相似度比較等功能；又支持使用者加載自定義用戶詞庫，設置詞頻、詞性；同時自動判斷編碼模式；最令人稱歎的是結巴不僅支持中文簡體，尚能應用在中文繁體斷詞，這對國內使用者而言，無疑是一項福音。

[9] 許菱祥，1986，**中文文法**，大中國圖書公司，臺北。

[10] 余思翰，2008，**中文作文寫作輔助系統**(碩士論文)，國立交通大學資訊科學與工程研究所，新竹市。

結巴分詞一篇文章之後，能產出許多數據，包括全文字數、詞彙數、相同詞彙數、相異詞彙數、名詞數量、形容詞數量……等諸多資料，在如許多的數據資料下，本人選擇全文字數、段落數、詞彙數、四字以上詞彙數比重、相異詞彙數比重、形容詞數比重、指稱詞數比重、語副詞數比重共八項數據。綜合過往改作文的經驗，作文字數過少，確實難以充分發揮題意；蓋作文講究起承轉合，字數過少實難展開這種過程，所以設定「全文字數」為變數之一。其次，作文若不分段，或者只有兩段，都難以呈現起承轉合的過程，三段是最基本的，若字數超過 600 字，分成四、五段更好。

另外，若一文中反覆出現相同的詞彙，如果這個詞彙不是專用術語，而是日常詞彙，例如「父母」一詞，可取「雙親」、「爸媽」或「父親母親」來變換，如果作者總是一個「父母」用到底，足以顯示該生腹笥甚窘，所熟悉的詞彙不多，故只好不停重複使用。是以相同詞彙愈多，就表示此人作文能力愈低下；反之，相異詞彙愈多，愈能彰顯此人寫作能力優異。此外亦可觀察該名學生使用四個字、五個字以上的長詞的能力有多強，一般而言，四字以上的長詞多半是遠離口語化、較有文氣的詞彙或成語，這一類的詞彙比重愈大，即顯示該名學生的寫作能力愈強。類似這種的指標就能間接分別出作文的高低好壞。其餘類推。

## 二、資料分析演算

機器學習就是讓機器（計算機）具備學習能力，從資料中自動學習規則，進而利用規則對新資料進行預測。其中過程乃是設計可以自動學習的演算法，讓機器從過去資料或經驗建立一個模型（Model），而學習（Learning）就是執行此模型[11]。目前常用的機器學習可分成監督式學習（Supervised Learning）和非監督式學習（Unsupervised Learning）兩種形態。前一種學習是從訓練資料中建立一個模型，並依此模型預測新資料，訓練資料是由輸入資料和預期輸出所組成。後一種學習則是預期資料中並無預期輸出，分群演算法就是常見的非監督式學習演算法。（李仁鐘、李秋緣，2017）本次研究合併採用監督式與非監督式的學習演算法，觀察哪一種演算法能達到最高精準度，作為日後電腦自動批閱作文的方法依據。

本論文借用 Jason Brownlee, Ph.D. 撰寫的” Your First Machine Learning Project in R Step-By-Step (tutorial and template for future projects)”[12]網頁中教導初學者如何利用各種簡單的演算法進入機器學習

[11] 李仁鐘、李秋緣，2017，**R 語言資料分析：從機器學習、資料探勘、文字探勘到巨集資料分析**，博碩文化，臺北，頁 56。

[12] ” Your First Machine Learning Project in R Step-By-Step (tutorial and template for future projects)” 網址為：https://machinelearningmastery.com/machine-learning-in-r-step-by-step/

的殿堂。首先將數據集分成十組，九組為訓練，一組當做測試，重複三次演算過程套用在每一種演算法上，以十倍交叉驗證來評估準確度。Jason Brownlee 列舉五種演算法，建置了五種模型，以評估何種演算法的準確度最高。五種演算法分別是：

1. Linear Discriminant Analysis (LDA)線性判別分析
2. Classification and Regression Trees (CART).分類與回歸樹
3. k-Nearest Neighbors (kNN).
4. Support Vector Machines (SVM) with a linear kernel.支持向量機器
5. Random Forest (RF)隨機森林

最後再加上一項常用的人工神經網路演算法（nn），共六種演算法從事計算分析。

## 三、實驗素材

本次研究素材是根據本校 102 學年度與 CWT 中文能力檢測中心合作舉辦大一學生中文能力檢定，CWT 全民中檢的測驗分兩部分，一是語文素養測驗，題型為選擇題；另一則是寫作測驗，當場要寫出一篇作文。兩種測驗皆有各自合格的標準，寫作測驗分成六級，四級分（含）以上才算通過寫作測驗。CWT 全民中檢的檢定共分為六個等級：基礎等、初等、中等、中高等、高等和優等，本校學生報考的是初等至高等四個等級，每個等級的寫作題目不一樣，筆者根據最大多數人報考的等級——初等——來進行本次研究。

102 學年度報考初等中文能力檢定測驗的學生共 411 人，通過四級分者有 270 人，未通過者有 141 人，同一道作文題目「我想說的話」，全部集中在一級、三級、四級、五級。每一篇作文經過結巴分詞以擷取前文所說的八項數據，此八項數據用以充當「自變數」，再加上閱卷老師所給予的評分等級充當「依變數」「quality」，總共九個變數作為演算預測的依據。以下即為結巴分詞與擷取過程。

# 伍、 結巴分詞與擷取變數數據

## 一、讀取內容，計算字數和段落數

茲取一篇作文為例，結巴的斷詞如下：

```
words <- readLines("D:/write/1503610003.txt", encoding = "UTF-8")
```

[1] "從小到大跟媽媽說過氣話也賭過氣，也曾經覺得媽媽不疼我，可是都會後悔自己講過想過的話和事情。"

[2] "從我有記憶以來，媽媽都對我很好，只是會有時候覺得媽媽對哥哥特別好，會因為這樣對媽媽生悶氣，現在想想，之前生的悶氣根本不值得。我常生病，每次生病都是媽媽在身邊照顧，還有我只要說想吃什麼，通常都會買給我，就算一開始不答應，也會因為我一直說想吃，到最後還是會買給我。現在我外面，離家很遠，媽媽都會打來關心我在這裡過得好不好、適不適應、有沒有交到朋友之類的。媽媽的工作辛苦，常常哪裡痠痛，我都會幫忙抓一抓，可是不曾說過，媽媽辛苦了。"

[3] "想說的是：媽媽辛苦了！謝謝妳！我身體不好也讓媽媽擔心，我會好好照顧身體不會讓妳擔心。也想再說一句:媽媽我愛妳！"

若想知道這篇文章的段落字數，可用 nchar()程式統計出來：

```
nchar(words)
[1]  50 216  59
sum(nchar(words))
[1] 325
```

總共三段，第一段 50 字，第二段 216 字，第三段 59 字，總和便是 325 字。此為最基本的讀取文本、統計全文字數和段落數的方式。

## （二）斷詞以統計詞彙數和各詞性的數量

依照結巴的程式設計，worker()是分詞引擎，segment()則是分詞的函數。試以上文做結巴斷詞的練習素材，檢查它的斷詞功能有多強大。

```
library(jiebaRD)
library(jiebaR)
paper<- segment(words, worker())
paper
```

結果如下：

[1] "從小到大" "跟" "媽媽" "說過" "氣話" "也" "賭"

[8] "過氣" "也" "曾經" "覺得" "媽媽" "不疼" "我"

[15] "可是" "都" "會" "後悔" "自己" "講過" "想過"

[22] "的話" "和" "事情" "從" "我" "有" "記憶"

[29] "以來" "媽媽" "都" "對" "我" "很" "好"

[36] "只是" "會" "有時候" "覺得" "媽媽" "對" "哥哥"

[43] "特別" "好" "會" "因為" "這樣" "對" "媽媽"

[50] "生悶氣" "現在" "想想" "之前" "生" "的" "悶氣"

[57] "根本" "不" "值得" "我常" "生病" "每次" "生病"

[64] "都" "是" "媽媽" "在" "身邊" "照顧" "還有"

[71] "我" "只" "要說" "想" "吃" "什麼" "通常"

[78] "都" "會" "買給" "我" "就算" "一" "開始"

[85] "不" "答應" "也" "會" "因為" "我" "一"

[92] "直說" "想" "吃" "到" "最後" "還是" "會買給"

[99] "我" "現在" "我" "外面" "離家" "很遠" "媽媽"

[106] "都" "會" "打來" "關心" "我" "在" "這裡"

[113] "過得" "好不好" "適" "不" "適應" "有沒有" "交到"

[120] "朋友" "之類" "的" "媽媽" "的" "工作" "辛苦"

```
     [127] "常常"     "哪裡"     "痠痛"     "我"       "都"
"會"         "幫忙"
     [134] "抓"      "一"       "抓"       "可是"     "不曾"
"說過"       "媽媽"
     [141] "辛苦"     "了"       "想"       "說"       "的是"
"媽媽"       "辛苦"
     [148] "了"      "謝謝"     "妳"       "我"       "身體"
"不好"       "也"
     [155] "讓"      "媽媽"     "擔心"     "我會"     "好好"
"照顧"       "身體"
     [162] "不會"     "讓"       "妳"       "擔心"     "也"
"想"         "再說"
     [169] "一句"     "媽媽"     "我愛"     "妳"
```

綜觀這篇文章的斷詞結果，絕大部分的斷詞是正確的，譬如「從小到大」、「悶氣」、「擔心」、「之類」、「好好」、「好不好」等等，皆是日常用語，結巴均能準確分斷出來。其中有疑慮的是「生悶氣」，此文出現兩次「悶氣」，卻未能保持一貫的分詞，正確的斷法應是"生""悶氣"，「生」為獨立的動詞，無須與悶氣連結一起。又如"我愛" "妳"，嚴格來說應該是"我" "愛" "妳"。不過若是為了比對文章相似度，或提取關鍵詞，「我愛妳」或可改為一個完整的詞彙，成為"我愛妳"。類似這種新增詞彙的作法，結巴也有提供，一勞永逸的作法便是直接在 jiebaR 的字典裡(C:\Users\user\Documents\R\win-library\3.5\jiebaRD\dict)添加新詞即可。

斷詞已經有了結果，就可以根據這個結果進行詞彙數量的統計。

```
length(paper)
[1] 172
```

經由 segment()的斷詞得知該文有三段，325 個字元，172 個詞彙，下一步我們來檢視詞彙組成的實際情形。

```
nchar(paper)
[1] 4 1 2 2 2 1 1 2 1 2 2 2 2 1 2 1 1 2 2 2 2 2 1 2 1 1 1 2 2 2 1 1 1 1
2 1 3 2 2 1 2 2 1 1 2 2 1 2 3 2 2 2 1 1 2 2 1 2 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1
1 2 2 1
[79] 1 2 1 2 1 2 1 2 1 1 2 1 1 2 1 1 1 2 2 3 1 2 1 2 2 2 2 1 1 2 2 1 1 2
2 3 1 1 2 3 2 2 2 1 2 1 2 2 2 2 2 1 1 1 2 1 1 1 2 2 2 2 2 1 1 1 2 2 2 1 2 1 1
2 2 1 1 2
[157] 2 2 2 2 2 2 1 1 2 1 1 2 2 2 2 1
table(nchar(paper))
1   2   3   4
72  94  5   1
```

一個字組成的詞彙有 72 個，兩個字的詞彙有 94 個，三個字的則有 5 個，四個字的唯有 1 個。此即透露該名學生不擅長運用長詞，只會使用日常口語的字詞，這在評分上會是一項弱點。

此文不重複的詞彙有多少？重複的詞彙又是多少？以下就是結巴的統計。

```
colSums(table(freq(paper)))
1   2   3   4   5   6   7   11  12
79  15  6   1   1   1   1   1   1
```

這張表格的讀法是：有 1 個字詞重複 12 次，再有 1 個字詞重複 11 次，以下類推。至於重複 12 次、11 次、7 次的詞彙分別是什麼，可以使用 freq()函數完整呈現，在此不再重複。只有出現 1 次的詞彙有 79 個，這 79 個詞彙便是相異詞彙，所佔的比重為 45.93%，不到一半，顯見該名學生的作文能力並不出色。這是否成為評量作文的關鍵，在此難以判定；然若透過 R 的程式分析，或許能得出一個清楚的關連性法則。

接著是詞性的標記。結巴詞性標記的函數為 tagging()，做法如下：

```
x = worker("tag")
tagging(words, x)
```

```
      l       p       n       x       n
d     zg      n       d       d       v
n     x       r
  "從小到大"  "跟"  "媽媽"  "說過"  "氣話"
"也"  "賭"  "過氣"  "也"  "曾經"  "覺得"
"媽媽"  "不疼"  "我"
      c       d       v       v       r
x     x       u       c       n       zg
r     v       n
  "可是"  "都"  "會"  "後悔"  "自己"
"講過"  "想過"  "的話"  "和"  "事情"  "從"
"我"  "有"  "記憶"
      f       n       d       p       r
zg    a       c       v       l       v
n     p       ns
  "以來"  "媽媽"  "都"  "對"  "我"
"很"  "好"  "只是"  "會"  "有時候"  "覺得"
"媽媽"  "對"  "哥哥"
      d       a       v       x       r
p     n       i       t       v       f
vn    uj      n
  "特別"  "好"  "會"  "因為"  "這樣"
"對"  "媽媽"  "生悶氣"  "現在"  "想想"  "之前"
"生"  "的"  "悶氣"
      a       d       v       x       n
r     n       d       v       n       p
s     v       v
  "根本"  "不"  "值得"  "我常"  "生病"
"每次"  "生病"  "都"  "是"  "媽媽"  "在"
```

"身邊" "照顧" "還有"
r d c v v
r d d v v r
v m v
"我" "只" "要說" "想" "吃"
"什麼" "通常" "都" "會" "買給" "我"
"就算" "一" "開始"
d v d v x
r m v v v v
f c x
"不" "答應" "也" "會" "因為"
"我" "一" "直說" "想" "吃" "到"
"最後" "還是" "會買給"
r t r f n
d n d v v n
r p x
"我" "現在" "我" "外面" "離家"
"很遠" "媽媽" "都" "會" "打來" "關心"
"我" "在" "這裡"
x l v d v
v v n r uj n
uj vn a
"過得" "好不好" "適" "不" "適應" "
有沒有" "交到" "朋友" "之類" "的" "媽媽"
"的" "工作" "辛苦"
d x vn r d
v v v m v c
d x n
"常常" "哪裡" "痠痛" "我" "都"
"會" "幫忙" "抓" "一" "抓" "可是"
"不曾" "說過" "媽媽"
a ul v zg x

```
n          a          ul         nr         zg         r
n          d          d
        "辛苦"     "了"     "想"     "說"     "的是"
"媽媽"   "辛苦"     "了"   "謝謝"     "妳"     "我"
"身體"   "不好"     "也"
           v          n          v          n          d
v          n          v          v          zg         v
d          v          c
        "讓"     "媽媽"   "擔心"   "我會"   "好好"
"照顧"   "身體"   "不會"     "讓"     "妳"   "擔心"
"也"     "想"   "再說"
           m          n          x          zg
        "一句"   "媽媽"   "我愛"     "妳"
```

a、c、d、f……英文字母為詞性縮寫，「a」是形容詞，取英語形容詞 adjective 的第一個字母；「c」為連詞，取英語連詞 conjunction 的第一個字母；「d」則是副詞，取 adverb 的第二個字母……，這些縮寫各自對應下一列的詞彙。關於「的」、「了」等語副詞則為 uj、ul，單純陳列這些詞性的函數為 attributes()，再加上 table()便可在一瞬間算出全部的次數。

```
attributes(tagging(words, x))
 [1] "l"  "p"  "n"  "x"  "n"  "d"  "zg" "n"  "d"  "d"  "v"  "n"
"x"  "r"  "c"  "d"  "v"  "v"  "r"  "x"  "x"  "u"  "c"  "n"  "zg"
"r"  "v"  "n"  "f"  "n"  "d"
 [32] "p"  "r"  "zg" "a"  "c"  "v"  "l"  "v"  "n"  "p"  "ns" "d"
"a"  "v"  "x"  "r"  "p"  "n"  "i"  "t"  "v"  "f"  "vn" "uj" "n"  "a"
"d"  "v"  "x"  "n"  "r"
 [63] "n"  "d"  "v"  "n"  "p"  "s"  "v"  "v"  "r"  "d"  "c"
"v"  "v"  "r"  "d"  "d"  "v"  "v"  "r"  "v"  "m"  "v"  "d"  "v"
"d"  "v"  "x"  "r"  "m"  "v"  "v"
 [94] "v"  "v"  "f"  "c"  "x"  "r"  "t"  "r"  "f"  "n"  "d"
"n"  "d"  "v"  "v"  "n"  "r"  "p"  "x"  "x"  "l"  "v"  "d"  "v"
"v"  "v"  "n"  "r"  "uj" "n"  "uj"
[125] "vn" "a"  "d"  "x"  "vn" "r"  "d"  "v"  "v"  "v"  "m"
```

```
"v"  "c"  "d"  "x"  "n"  "a"  "ul" "v"  "zg" "x"  "n"  "a"  "ul"
"nr" "zg" "r"  "n"  "d"  "d"  "v"
    [156] "n"  "v"  "n"  "d"  "v"  "n"  "v"  "v"  "zg" "v"  "d"
"v"  "c"  "m"  "n"  "x"  "zg"
    table(attributes(tagging(words_test, x)))
    a  c  d  f  i  l  m  n nr ns  p  r  s  t  u uj ul  v vn
x  zg
    6  7 23  4  1  3  4 25  1  1  6 16  1  2  1  3  2 42
3 14  7
```

這裡可以看出形容詞 a 有 6 個，連詞 c 則有 7 個，語副詞 uj+ul 則有 5 個，指稱詞 r 則是 16 個。可是這樣的陳列方式實在令人眼花撩亂，一個不留神就會對應錯誤，想改善這種現象，可將 table 轉換成 data.frame 格式，如此無論檢視或計算皆能方便執行。

```
    y <- as.data.frame(table(attributes(tagging(words, x))))
    y
       Var1  Freq
    1     a     6
    2     c     7
    3     d    23
    4     f     4
    5     i     1
    6     l     3
    7     m     4
    8     n    25
    9    nr     1
    10   ns     1
    11    p     6
    12    r    16
    13    s     1
    14    t     2
    15    u     1
    16   uj     3
```

```
17    ul     2
18     v    42
19    vn     3
20     x    14
21    zg     7
```

這種對應型態即能一目了然，不會讓人眼花。此中筆者欲取形容詞（a）、指稱詞（r）、語副詞（uj+ul）三項作為評量的指標，取用指稱詞的用意和語副詞一樣，凡是動輒你、我、他，濫用指稱詞的文章，作者的文筆自然不佳，想不出其他代詞來變換。至於形容詞數量的多寡，在某種程度上也可以鑒別出文筆的好壞，擅長使用形容詞的人，他的想像力、描繪力總不會太差，文章除了陳述自己心意、想法之外，我們還希望他的陳述能夠生動活潑，引人入勝，此時便要看他的形容事物的能力是強是弱。結巴有標記詞性的功能，在這方面，筆者可以省略事先開發一套形容詞庫的辛苦工作，直接運用結巴"tagging"的公式即可抽提出形容詞的數量。

檢視形容詞（a）、指稱詞（r）、語副詞（uj+ul）三項指標的數量有多少，也無須一一察看，僅須透過 pipe 便能快速列出這些詞性的數目。

```
library(magrittr)
library(dplyr)
y %>% filter(Var1 %in% c("a","r","uj","ul"))
  Var1 Freq
1    a    6
2    r   16
3   uj    3
4   ul    2
```

綜合以上各種數據，筆者選出全文字數、段落數、詞彙數、四字以上詞彙數比重、相異詞彙數比重、形容詞數比重、指稱詞數比重、語副詞數比重共八項數據藉以成為評量文章好壞的特徵，當成未來數據分析的自變數，經過程式演算法決定文章的等級「quality」（亦即依變數），看看經過機器學習過後的等級是否與人工閱卷的等級差別有多大，而來決定它的精準度。

## 陸、 資料表結構檢視

筆者已將當年度初等考生所有的寫作測驗轉換成各種數據，製作

成.csv 檔，有 411 個觀測值，九個變數，最後一個是類別等級，為演算最後結果的比對依據。檔案如下：

```
> cwt <- read.csv(file.choose())
> head(cwt)
  段落 全文字數 詞彙數 相異詞比重 代詞數比重 語副詞比重 形容詞比重 四字以上比重 quality
1    5      414    232      0.517      0.086      0.121      0.052      0.013       4
2    3      307    183      0.464      0.191      0.120      0.000      0.000       4
3    2      188    111      0.477      0.171      0.063      0.036      0.027       3
4    2      264    171      0.439      0.170      0.064      0.012      0.018       3
5    4      448    281      0.331      0.167      0.075      0.028      0.014       4
6    5      509    337      0.315      0.098      0.062      0.027      0.009       3
> summary(cwt)
      段落          全文字數        詞彙數        相異詞比重      代詞數比重
 Min.   : 1.000   Min.   :  81.0   Min.   :  41.0   Min.   :0.1530   Min.   :0.0170
 1st Qu.: 3.000   1st Qu.:314.5    1st Qu.:191.5    1st Qu.:0.4105   1st Qu.:0.1050
 Median :4.000    Median :393.0    Median :238.0    Median :0.4950   Median :0.1290
 Mean   : 3.616   Mean   :393.0    Mean   :237.2    Mean   :0.4961   Mean   :0.1323
 3rd Qu.: 4.000   3rd Qu.:465.0    3rd Qu.:281.0    3rd Qu.:0.5820   3rd Qu.:0.1565
 Max.   : 11.000  Max.   :816.0    Max.   :438.0    Max.   :0.8080   Max.   :0.2800
   語副詞比重      形容詞比重      四字以上比重        quality
```

```
     Min.   :0.02000                  Min.   :0.0000                  Min.   :0.00000
Min. :1.000
     1st Qu.:0.06500       1st Qu.:0.0200          1st Qu.:0.00700              1st
Qu.:3.000
     Median :0.08200          Median :0.0280              Median :0.01300
Median :4.000
     Mean   :0.08298             Mean   :0.0294              Mean   :0.01512
Mean :3.582
     3rd Qu.:0.09800       3rd Qu.:0.0380          3rd Qu.:0.02000              3rd
Qu.:4.000
     Max.   :0.17600             Max.   :0.0810              Max.   :0.09800
Max. :5.000
```

列出前六筆資料，如上表所陳列，「quality」是人工閱卷老師決定的等級，屬於因素向量（factor），而非數值向量（vector），但資料結構顯示 quality 與其他變數一樣，同為數值型態，因此有必要將這一變數轉換成因素向量。

```
> cwt$quality <- as.factor(cwt$quality)
> class(cwt$quality)
[1] "factor"
```

經過 as.factor()函數的轉換，quality 的形態果然修正為"factor"。寫作測驗雖然有六個等級，實際上此次檢測的結果只有一級、三級、四級、五級，二級和六級付之闕如。因此 quality 裡的值為有 1，3，4，5 四個數字。411 個觀測值中沒有遺漏值，是十分齊備完整的資料。以下筆者即根據這張資料表進行演算分析。

## 一、 區分訓練資料與測試資料

### (一)、按照等級隨機抽樣

本研究的探勘過程，首先將 411 篇作文，按照 quality 的類別隨機抽樣，各類別均抽出 70%的樣本作為訓練資料。方法如下：

```
#各類品種各取 70%出來訓練
> train_cwt <- createDataPartition(cwt$quality, p=0.70, list=FALSE)
> traindata <- cwt[train_cwt,]
> testdata <- cwt[-train_cwt,]          #剩下 30%充當測試資料
```

檢視訓練和測試資料分別有多少觀察樣本。

```
> dim(traindata)
[1] 290   9
> dim(testdata)
[1] 121   9
```

訓練資料（traindata）有 290 個樣本，測試資料（testdata）則有 121 個樣本。而無論訓練資料或測試資料均含有"1" "3" "4" "5"四個等級的樣本，沒有特別偏重而導致遺失某一類樣本。

```
> levels(traindata$quality)
[1] "1" "3" "4" "5"
> levels(testdata$quality)
[1] "1" "3" "4" "5"
```

**2、計算 traindata 各類別的百分比和次數**

```
>percentage <- prop.table(table(traindata$quality)) *100      # 計算 quality 的百分比
> cbind(freq=table(traindata$quality), percentage=percentage)
     freq   percentage
1    10     3.448276
3    96    33.103448
4   179    61.724138
5     5     1.724138
```

## 二、　圖形繪圖

繪圖是 R 語言頗佔優勢的一強大功能，我們可以試著從各種圖像觀察訓練資料的各種變數的變化。茲以箱型圖繪出各個變數的變化情形：

```
> featurePlot(x=x, y=y, plot="box")
```

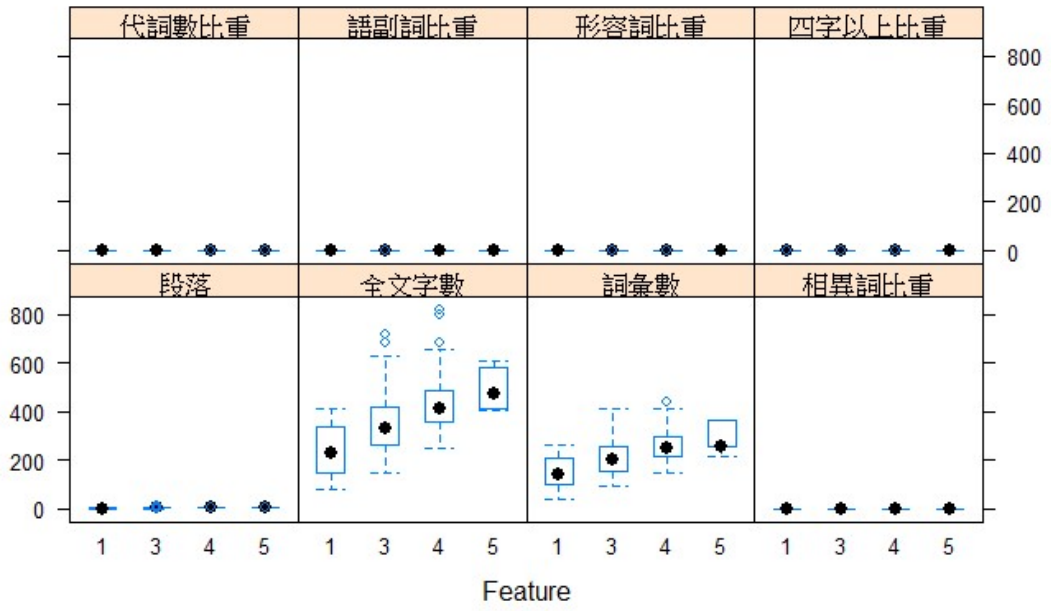


圖 1　featurePlot 箱型圖

另一種箱型圖：

```
> x <- traindata[,1:8]
> y <- traindata[,9]
> par(mfrow=c(1,5))
> for(i in 1:9) {
+     boxplot(x[,i], main=names(traindata)[i])
+ }
```

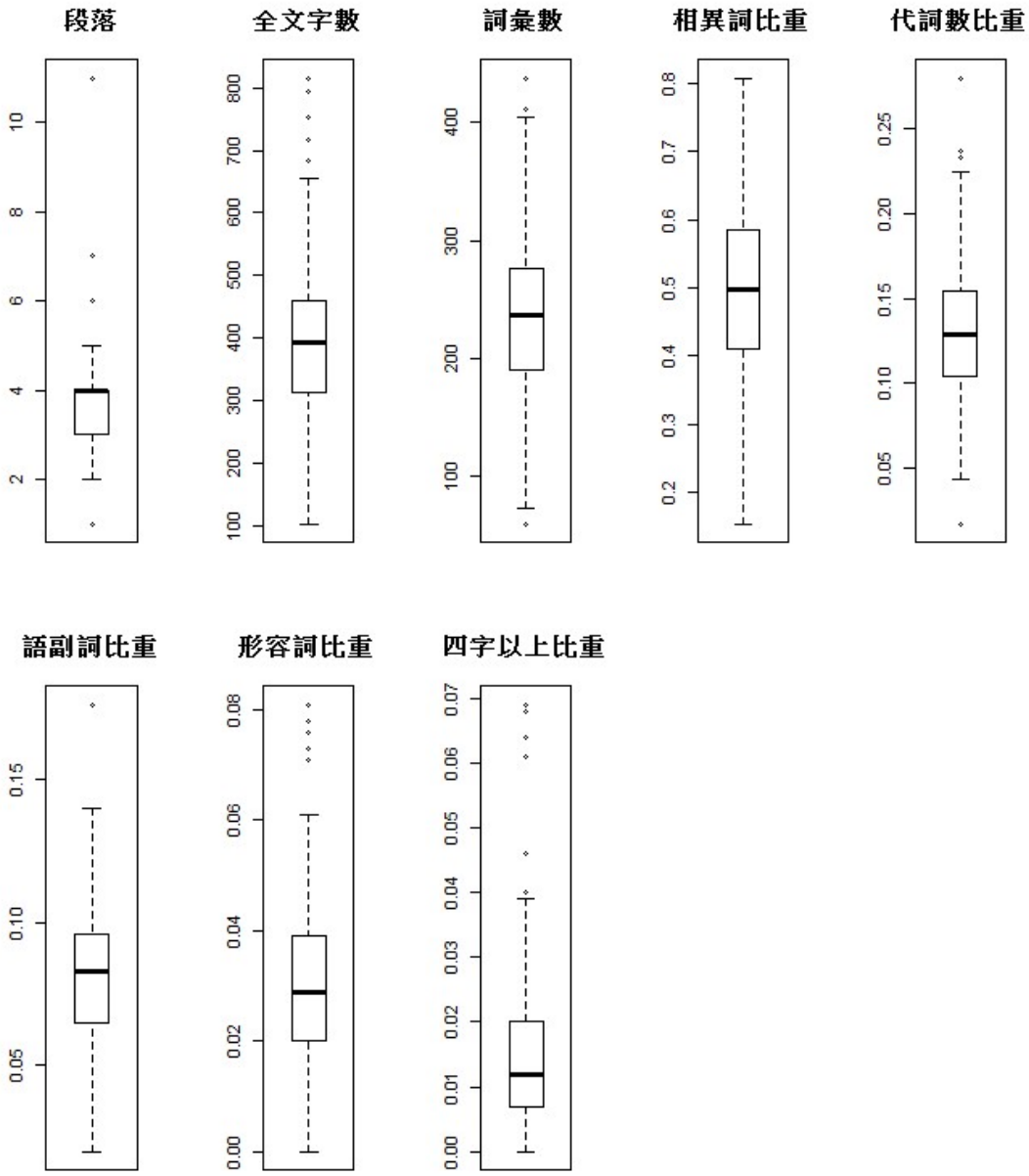


圖 2 個別變數之箱型圖

現使用 density()函數繪製 traindata 的機率密度圖形：

```
> scales <- list(x=list(relation="free"), y=list(relation="free"))
> featurePlot(x=x, y=y, plot="density", scales=scales)
```

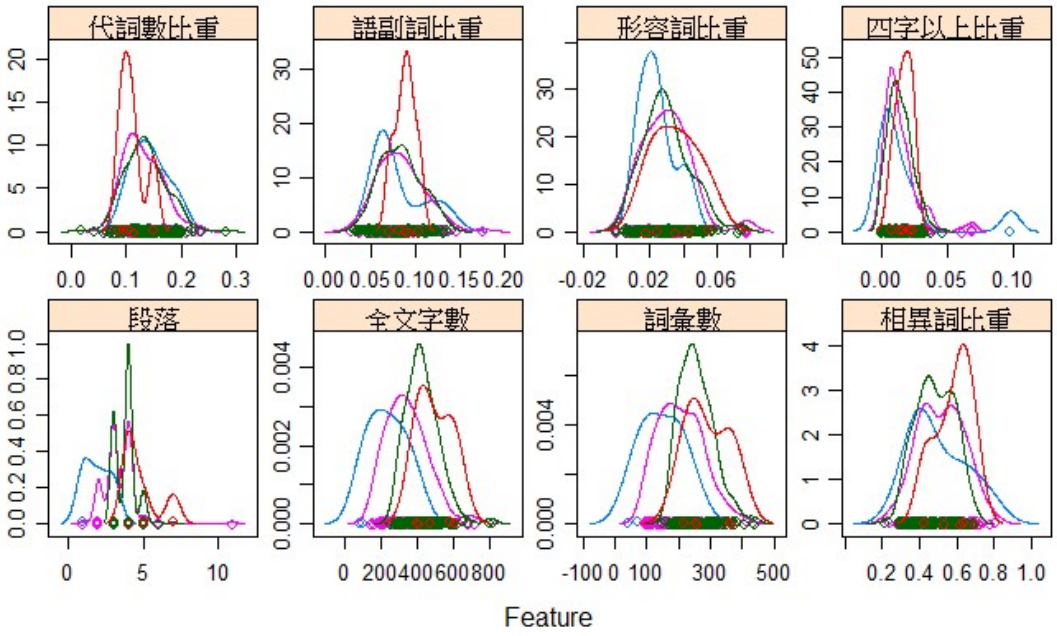


圖 3 機率密度圖

以上即是針對訓練資料（traindata）所繪製的各式圖形，自然單從圖形是無法斷言其中的關聯性，只是從圖形呈現可以得知各變數和 quality 等級的相關分布。

## 柒、 R 語言演算法

### 一、 各式演算法說明

LDA（Linear Discriminant Analysis）也叫做 Fisher 線性判別(Fisher Linear Discriminant, FLD)，是一種線性學習演算法，也是一種監督學習的降維技術。資料集的每個觀測樣本都有類別輸出。它的做法是先設定一組訓練樣本資料，設法把樣本資料投影到一條直線上，使同類樣本的投影點儘可能接近，不同類的樣本投影點盡量遠離。[13]在對新樣本進行分類時，同樣也將樣本投影到這條直線上，再根據投影點的位置來確定新樣本的類別。此謂之 LDA 線性判別分析。

分類與回歸樹(Classification and Regression Trees, CART)是決策樹的演算法之一，在機器學習的眾多方法中，決策樹可謂最具代表性的演算法之一，因為它具有不錯的預測精準度和解釋能力，廣為各個領域使用。常見的決策樹演算法包括分類與回歸樹(CART)、ID3(Inductive Dichotomiser)與 C5.0。CART 是 Breiman 等學者所開發出來的演算法，運算方式主要運用二元（Binary）分割原理，在運算過程中永遠分割出兩個子節點，這樣的過程會一再被重複。兩個子節點再分別切割出兩個子

[13] 參考「LDA 線性判別分析」，網址：https://zhuanlan.zhihu.com/p/32658341

節點，最後以所有的葉節點（類別）錯誤率的加權總數做為修剪樹狀的依據。安裝的套件是 rpart。（李仁鐘、李秋緣，2017）

KNN 演算法（又譯 K-近鄰演算法）是一種距離判別的演算法，意即根據待判斷樣本和已知類別樣本之間的距離遠近做出判別。[14]此種演算法採取向量空間模型來做分類，概念是相同的案例組成一類別，由於彼此具有高度的相似度，故而可以藉由演算與已知類別案例的相似度，來評估未知類別案例的可能分類。它的輸入包含特徵空間中的 $k$ 個最相近的訓練樣本。輸出則是一個分類族群。一個物件的分類是藉由它的鄰居「多數表決」來確定，$k$ 個最靠近鄰居（$k$ 為正整數，通常較小）中最常使用的分類決定了賦予該物件的類別。若 $k = 1$，則該物件的類別直接由最近的一個節點賦予。k-近鄰演算法是所有的機器學習演算法中最簡單的一種演算法。[15]

支持向量機器（Support Vector Machine，SVM）是資料探勘中的一項新技能，建立在統計學理論的風險最小原理和 VC 維理論結構的基礎上，根據樣本在模型的學習能力與複雜性之間尋求最佳折中，以期獲得最好的推廣能力。（黃文、王正林，2016，頁 12-2）蓋機器學習原本就是一種對於所研究問題的真實模型之逼近，此時會假設一個近似的模型，按照適當原理不斷逼近真實模型。然不管如何逼近，總會與真實模型有些許的差距，也就是誤差，此誤差稱之為風險，為極力縮小與真實模型之間的誤差，遂引進「結構風險最小原理」。SVM 即是建立在這個基礎上。（黃文、王正林，2016，頁 12-2）R 中支援向量機器建模和分析的程式套件為 e1071，其中的核心函數是 svm()。

隨機森林是近十幾年新竄起的一種演算法，它的前身為分類樹，20 世紀 80 年代 Breiman 等人發明分類樹演算法，透過反覆二分數據進行分類或回歸；2001 年 Breiman 把分類樹組合成隨機森林，在變數和資料的使用上進行隨機化，產生許多分類樹，再整理分類樹的結果。（黃文、王正林，2016，頁 11-2）與其有關的程式套件有 5 個函數，分別為 important()，MDSplot()，treesize()和 randomForest()，本文採用最後一種函數進行演算。

[14] 黃文、王正林編著，2016，**利用 R 語言打通大數據的經脈**，佳魁資訊，臺北，頁 8-4。

[15] 參見維基百科「最近鄰居法」一條。網址：https://zh.wikipedia.org/wiki/%E6%9C%80%E8%BF%91%E9%84%B0%E5%B1%85%E6%B3%95

## 二、　演算過程

本實驗共採用六種演算法，演算過程如下：

```
> control <- trainControl(method="cv", number=10)
> metric <- "Accuracy"
# a) linear algorithms     lda 線性演算法
> library(e1071)
> set.seed(123)     #設定隨機種子
> fit.lda <- train(quality~., data=traindata, method="lda", metric=metric, trControl=control)
# b) nonlinear algorithms
# CART   分類與回歸樹
> set.seed(123)
> fit.cart <- train(quality~., data=traindata, method="rpart", metric=metric, trControl=control)
# kNN   K-近鄰演算法
> set.seed(123)
> fit.knn<- train(quality~., data=traindata, method="knn", metric=metric, trControl=control)
#  支持向量機器 SVM 演算法
> set.seed(123)
> fit.svm <- train(quality~., data=traindata, method="svmRadial", metric=metric, trControl=control)
#  隨機森林演算法
> library(randomForest)
> set.seed(123)
> fit.rf <- train(quality~., data=traindata, method="rf", metric=metric, trControl=control)
```

類神經網路演算法（nn）演算如下：

```
> library(nnet)
> set.seed(123)
> fit.nn <- train(quality~., data=traindata, method="nnet", metric=metric, trControl=control)
```

```
# weights:  17
initial  value 309.164468
iter  10 value 220.380836
final  value 220.379519
converged
# weights:  43
initial  value 344.673137
iter  10 value 220.467262
final  value 220.379519
converged
……    #中間過程省略
# weights:  69
initial  value 391.125431
iter  10 value 222.983616
iter  20 value 222.857507
final  value 222.857357
converged
# weights:  69
initial  value 492.204181
iter  10 value 248.511746
iter  20 value 244.883017
iter  30 value 239.042413
iter  40 value 208.208263
iter  50 value 199.035959
iter  60 value 197.093697
iter  70 value 193.638596
iter  80 value 192.426825
iter  90 value 190.176130
iter 100 value 189.443260
final  value 189.443260
stopped after 100 iterations
```

類神經網路是經典的機器學習模型，流行已有半個世紀之久，雖然預測精準，但計算量非常龐大，因此本文將其中間計算過程省略。正因其計算量過於龐大，方有隨機森林演算法出現。

## 三、　比較演算法之精確度

六種演算法演算過後，現在檢視演算的結果，並繪圖比較六種演算法的精確度。

```
> results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf, nn=fit.nn))
> summary(results)
Call:
summary.resamples(object = results)
Models: lda, cart, knn, svm, rf, nn
Number of resamples: 10
Accuracy
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lda  0.6206897 0.6696429 0.7068966 0.7034319 0.7435345 0.7931034    0
cart 0.6206897 0.6896552 0.7071429 0.7173563 0.7560345 0.7931034    0
knn  0.5862069 0.6785714 0.6896552 0.6824959 0.7181034 0.7333333    0
svm  0.6333333 0.6896552 0.7370690 0.7278161 0.7823276 0.8000000    0
rf   0.6896552 0.7606322 0.7894089 0.7792036 0.7931034 0.8620690    0
nn   0.6333333 0.6785714 0.6896552 0.7033005 0.7500000 0.7666667    0
Kappa
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
      lda       0.12121212   0.2557451      0.3047399      0.3368082      0.3906286
0.5639098     0
      cart    0.05341246       0.2634370      0.3449964      0.3399148      0.4523077
0.5258856     0
      knn       0.08900524   0.2752373      0.3384899      0.3017492      0.3505029
0.4230769     0
      svm       0.15816327       0.2673688      0.4208725      0.3834586      0.5026049
0.5918367    0
      rf        0.27500000       0.4870956      0.5290718      0.5143008      0.5508558
0.6979167     0
      nn    0.15816327    0.2629861    0.3054358    0.3292776    0.4125250 0.5227273
0
      > dotplot(results)    # 繪圖
```

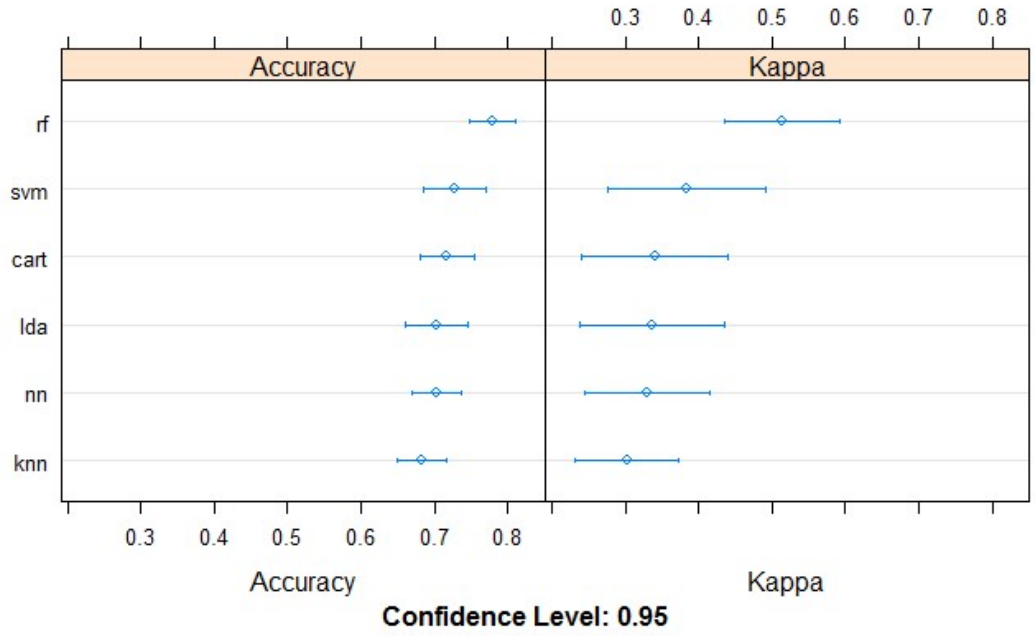


圖 4　Accuracy 比較圖

就圖形來看，精確度（Accuracy）為 rf（隨機森林）表現最佳，其次是 svm（支援向量機）；若嫌圖形尚未能精準呈現每個演算法的精確度，我們還可以直接列出每一種演算法的數據結果。

```
      > print(fit.lda)
      Linear Discriminant Analysis
      290 samples
```

```
  8 predictor
  4 classes: '1', '3', '4', '5'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...
Resampling results:
  Accuracy   Kappa
  0.7034319  0.3368082


> print(fit.cart)
CART
290 samples
  8 predictor
  4 classes: '1', '3', '4', '5'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...
Resampling results across tuning parameters:
  cp           Accuracy   Kappa
  0.02402402   0.6998522  0.3491451
  0.07207207   0.7173563  0.3399148
  0.27027027   0.6481034  0.1185714
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.07207207.


> print(fit.knn)
k-Nearest Neighbors
290 samples
  8 predictor
  4 classes: '1', '3', '4', '5'
No pre-processing
```

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...

Resampling results across tuning parameters:

| k | Accuracy | Kappa |
|---|---|---|
| 5 | 0.6513300 | 0.2574637 |
| 7 | 0.6824959 | 0.3017492 |
| 9 | 0.6654844 | 0.2447019 |

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 7.

> print(fit.rf)

Random Forest

290 samples

8 predictor

4 classes: '1', '3', '4', '5'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...

Resampling results across tuning parameters:

| mtry | Accuracy | Kappa |
|---|---|---|
| 2 | 0.7792036 | 0.5143008 |
| 5 | 0.7586289 | 0.4720969 |
| 8 | 0.7618391 | 0.4780019 |

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2.

> print(fit.nn)

Neural Network

290 samples

8 predictor

```
  4 classes: '1', '3', '4', '5'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...
Resampling results across tuning parameters:
  size  decay  Accuracy   Kappa
  1     0e+00  0.6240805   0.020212766
  1     1e-04  0.6174138   0.000000000
  1     1e-01  0.6788013   0.224456415
  3     0e+00  0.6558374   0.122466591
  3     1e-04  0.6307471   0.044444444
  3     1e-01  0.6997373   0.309707612
  5     0e+00  0.6376437   0.075520903
  5     1e-04  0.6139655  -0.003571429
  5     1e-01  0.7033005   0.329277621
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 5 and decay = 0.1.

> print(fit.svm)
Support Vector Machines with Radial Basis Function Kernel
290 samples
  8 predictor
  4 classes: '1', '3', '4', '5'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...
Resampling results across tuning parameters:
  C     Accuracy   Kappa
  0.25  0.6830870  0.2198053
  0.50  0.7247209  0.3624057
  1.00  0.7278161  0.3834586
```

```
Tuning parameter 'sigma' was held constant at a value of 0.1462679
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1462679 and C = 1.
```

綜合以上演算法的結果，每一種演算的預測精確度表現如下：

lda 精確度： 0.7034319

cart 精確度： 0.7173563，cp = 0.072

knn 精確度： 0.6824959，k = 7

nn 精確度： 0.7033005，size = 5，decay = 0.1

svm 精確度： 0.7278161，sigma = 0.1462679，C = 1

rf 精確度： 0.7792036，mtry = 2

排列比較的結果，隨機森林演算法（rf）高達 0.779 的精確度最高，以上是演算 70%的 traindata 的結果；若要預測 testdata 的新值，自然是以隨機森林為首選的演算法。以下是以 rf 預測 testdata 的結果：

```
> predictions <- predict(fit.rf, testdata)
> confusionMatrix(testdata$quality,predictions )
Confusion Matrix and Statistics
          Reference
Prediction  1  3  4  5
         1  1  3  0  0
         3  1 19 20  0
         4  0  1 75  0
         5  0  0  0  1
Overall Statistics
               Accuracy : 0.7934
```

```
                    95% CI : (0.7103, 0.8616)
       No Information Rate : 0.7851
       P-Value [Acc > NIR] : 0.4643

                     Kappa : 0.534
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: 1 Class: 3 Class: 4 Class: 5
Sensitivity          0.500000   0.8261   0.7895 1.000000
Specificity          0.974790   0.7857   0.9615 1.000000
Pos Pred Value       0.250000   0.4750   0.9868 1.000000
Neg Pred Value       0.991453   0.9506   0.5556 1.000000
Prevalence           0.016529   0.1901   0.7851 0.008264
Detection Rate       0.008264   0.1570   0.6198 0.008264
Detection Prevalence 0.033058   0.3306   0.6281 0.008264
Balanced Accuracy    0.737395   0.8059   0.8755 1.000000
> table(testdata$quality, predictions)
   predictions
     1  3  4  5
  1  1  3  0  0
  3  1 19 20  0
  4  0  1 75  0
  5  0  0  0  1
```

Accuracy 高達 0.7934，比諸訓練集 traindata 的精確度又更提升了 0.02。其餘演算法的預測皆不及 rf，故本文便不再展示其他演算法預測 testdata 的歷程。觀 rf 運算的精準度相當接近 80%，這樣的結果是出乎筆者意料之外；但若能調整或加入新的變數，是否能突破 0.80 的關卡？若能突破，則運用機器學習自動批改作文的能力又將備受肯定。

## 捌、 新增分群變數

分群分析是一種運用廣泛、原理簡單的探勘技術，做法是把許多事物按照某種標準分門別類，較為相近的便歸為一類，如此按照使用者指定的分類數目分成多類。此種方法能夠在客戶分類、文字分類、基因類別取得卓越功效，因此，此種分析方法目前正處於蓬勃發展的階段，成位資料探勘研究中的焦點。

分群演算法絕大多數皆能在 R 中演算，其中最實用、最具代表性的五種演算法如下：

1. K—平均值分群（K-Means）
2. k—中心點分群（K-Medoids）
3. 密度分群（Densit-based Spatial Clustering of Application with Noise，DBSCAN）
4. 系譜分群（Hierarchical Clustering，HC ）
5. 期望最大分群（Expectation Maximization，EM）

如許眾多的分群方法本文不可能一一採用，本文選用的是第一種 K-Means 演算法。K-Means 是最早出現的分群分析演算法之一，它是以隨機選取的 k（預設類別數）個樣本充當起始的中心點，把其餘樣本歸納於相似度最高中心點所在的簇（cluster），在確立目前簇中樣本座標的平均值為新的中心點，依次循環反覆地運算下去，直至所有樣本所屬類別不再變動。（黃文，王正林，2016，頁 7-2）

K 的預設類別預計分為四群、六群、八群來做演算，再將演算結果做為一個新變數（group），與原先的九個變數一起重新演算，看看是否能進一步提高精確度。

### 一、 分為四群

```
> cwt <- read.csv(file.choose())
> data <- cwt[, -9]
> head(data)
```

| 段落 | 全文字數 | 詞彙數 | 相異詞比重 | 代詞數比重 | 語副詞比重 | 形容詞比重 | 四字以上比重 |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 414 | 232 | 0.517 | 0.086 | 0.121 | 0.052 | 0.013 |
| 2 | 3 | 307 | 183 | 0.464 | 0.191 | 0.120 | 0.000 | 0.000 |
| 3 | 2 | 188 | 111 | 0.477 | 0.171 | 0.063 | 0.036 | 0.027 |

```
    4      2        264      171        0.439        0.170        0.064        0.012
0.018
    5      4        448      281        0.331        0.167        0.075        0.028
0.014
    6      5        509      337        0.315        0.098        0.062        0.027
0.009
    > kmeans.cluster <- kmeans(data, centers=4)
    > cwt$group <- print(kmeans.cluster$cluster)
          [1] 4 1 2 1 4 3 1 3 1 1 2 4 4 3 1 2 4 3 4 4 2 4 4 2 3 1 1 4 1 3 3 4
3 1 1 1 2 3 3 3 3 1 4 2 1 1 4 4 1 4 1 4 3 4 4
      [56] 1 3 4 4 1 3 1 1 3 4 2 1 4 4 3 1 4 1 4 1 2 1 1 4 3 1 1 1 4 4 1 1 4
1 2 1 1 1 2 4 1 1 2 1 2 1 4 1 1 1 1 1 3 4 1
    [111] 4 2 1 4 1 2 4 4 1 3 1 1 4 1 4 3 4 1 2 2 1 4 1 1 3 4 2 4 4 1 4 4 1
2 1 3 1 2 3 4 3 2 1 1 2 4 3 4 4 1 4 2 1 1 4
    [166] 4 4 1 1 4 1 4 4 3 4 1 1 1 1 1 4 3 4 1 3 3 3 4 3 1 2 4 4 3 1 1 4 3
3 4 1 4 4 4 4 4 4 1 2 3 2 4 3 4 3 3 1 1 4 2
    [221] 3 4 3 3 4 3 4 3 2 3 3 1 1 4 1 1 4 4 4 3 1 1 2 3 1 1 1 4 1 4 1 4 4
1 3 1 4 3 4 1 4 1 4 4 1 4 1 4 4 3 4 4 4 1 1
    [276] 4 1 1 4 2 3 1 1 1 4 4 1 2 3 1 2 2 4 4 4 4 1 3 1 4 3 1 3 4 1 1 3 1
4 1 1 3 2 4 4 1 2 4 4 1 4 3 3 3 4 3 4 4 1 3
    [331] 4 4 4 4 4 2 4 4 1 1 4 4 2 1 1 4 1 1 2 1 4 4 4 4 4 1 2 1 3 4 3 3 4 1 3
4 2 4 2 1 1 4 3 1 4 4 2 4 2 4 1 1 2 4 1 4 1
    [386] 4 3 4 3 3 3 4 3 1 2 4 4 2 1 4 4 2 2 4 1 4 4 4 4 3 3
    > kmeans.cluster$withinss
    [1] 194434.8 137317.0 460023.6 236856.6
    > table(kmeans.cluster$cluster, cwt$quality)
          1   3   4   5
      1   5  53  80   0
      2   8  36   4   0
      3   0  18  54   3
      4   1  29 117   3
```

將此結果應用在原先的 traindata 資料集中，結果如下：

```
> cwt <- cwt[ ,c( "段落","全文字數","詞彙數","相異詞比重","代詞數比重","語副詞比重","形容詞比重", "四字以上比重","group","quality")]
> cwt$quality <- as.factor(cwt$quality)
> head(cwt)
> library(caret)
> train_cwt <- createDataPartition(cwt$quality, p=0.70, list=FALSE)
> traindata <- cwt[train_cwt,]
> testdata <- cwt[-train_cwt,]
> levels(traindata$quality)
[1] "1" "3" "4" "5"
> levels(testdata$quality)
[1] "1" "3" "4" "5"
> percentage <- prop.table(table(traindata$quality))
> cbind(freq=table(traindata$quality), percentage=percentage)
   freq   percentage
1    10    3.448276
3    96   33.103448
4   179   61.724138
5     5    1.724138
> x <- traindata[,1:9]
> y <- traindata[,10]
> control <- trainControl(method="cv", number=10)
> metric <- "Accuracy"

> set.seed(123)
> fit.rf <- train(quality~., data=traindata, method="rf", metric=metric, trControl=control)
> results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf, nn=fit.nn))
```

```
> summary(results)
Call:
summary.resamples(object = results)
Models: lda, cart, knn, svm, rf, nn
Number of resamples: 10
Accuracy
       Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lda  0.5517241 0.6206897 0.6785714 0.6553695 0.6896552 0.7241379    0
cart 0.7000000 0.7264368 0.7543103 0.7520772 0.7586207 0.8275862    0
knn  0.5862069 0.6105296 0.6775862 0.6785550 0.7310345 0.7931034    0
svm  0.6551724 0.6896552 0.7000000 0.7105665 0.7410714 0.7586207    0
rf   0.6896552 0.7264368 0.7583333 0.7621921 0.7912562 0.8275862    0
nn   0.5862069 0.6922414 0.7120690 0.7244828 0.7500000 0.8620690    0
Kappa
       Min.    1st Qu.     Median      Mean   3rd Qu.      Max. NA's
lda  0.11084906 0.2059528 0.2502242 0.2773956 0.3587693 0.4313725    0
cart 0.33823529 0.3708223 0.4108262 0.4332472 0.4620167 0.6291560    0
knn  0.07507508 0.2151376 0.2988247 0.3120514 0.4457000 0.5335121    0
svm  0.24281984 0.2807354 0.3274946 0.3482462 0.4225327 0.4808184    0
rf   0.30400000 0.4421548 0.4773875 0.4866070 0.5288869 0.6596244
```

```
0
      nn     0.16346154 0.3184692 0.3423571 0.3903874 0.4725123 0.7025641
0
      > print(fit.rf)
      Random Forest
      290 samples
        9 predictor
        4 classes: '1', '3', '4', '5'
      No pre-processing
      Resampling: Cross-Validated (10 fold)
      Summary of sample sizes: 261, 261, 262, 262, 261, 261, ...
      Resampling results across tuning parameters:
        mtry  Accuracy   Kappa
        2     0.7621921  0.4866070
        5     0.7451724  0.4710493
        9     0.7347044  0.4555316
      Accuracy was used to select the optimal model using the largest
value.
      The final value used for the model was mtry = 2
```

此時 rf 的精確度為 0.7621921，比諸未加入分群變數時降了 0.03%。不過這是用在練習集（traindata）當中，若應用在測試集（testdata）不知效果又會如何？

將 rf 演算法移用到 testdata，結果是：

```
      > predictions <- predict(fit.rf, testdata)
      > confusionMatrix(testdata$quality,predictions )
      Confusion Matrix and Statistics
                  Reference
      Prediction  1  3  4  5
                  1  2  2  0  0
                  3  1 18 21  0
                  4  0  3 73  0
                  5  0  0  1  0
```

```
Overall Statistics
                    Accuracy : 0.7686
                      95% CI : (0.6832, 0.8404)
    No Information Rate : 0.7851
    P-Value [Acc > NIR] : 0.7149
                     Kappa : 0.4779
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: 1 Class: 3 Class: 4 Class: 5
Sensitivity           0.66667   0.7826   0.7684       NA
Specificity           0.98305   0.7755   0.8846 0.991736
Pos Pred Value        0.50000   0.4500   0.9605       NA
Neg Pred Value        0.99145   0.9383   0.5111       NA
Prevalence            0.02479   0.1901   0.7851 0.000000
Detection Rate        0.01653   0.1488   0.6033 0.000000
Detection Prevalence  0.03306   0.3306   0.6281 0.008264
Balanced Accuracy     0.82486   0.7791   0.8265       NA
> table(testdata$quality, predictions)
   predictions
     1  3  4  5
  1  2  2  0  0
  3  1 18 21  0
  4  0  3 73  0
  5  0  0  1  0
```

精確度依然不理想，僅有 0.7686。以上是加入分成四群的新變數的分析結果，看來成效不佳，若分成六群，不知精確度會如何？

## 二、　分為六群

```
> kmeans.cluster <- kmeans(data, centers=6)
> cwt$group <- print(kmeans.cluster$cluster)
> kmeans.cluster$withinss
[1]     80050.49     57011.78     73667.18   116944.94     52091.71
```

```
151203.04
    > table(kmeans.cluster$cluster, cwt$quality)
         1  3   4   5
    1    1  24  84  3
    2    3  37  28  0
    3    3  30  59  0
    4    0  16  67  0
    5    7  23  0   0
    6    0  6   17  3
```

**(一)、** traindata 訓練

由於演算公式一模一樣，因此不再詳列計算過程。在此直接取 rf 的程式運算結果。其精確度為 0.7654105。

**(二)、** testdata 預測

預測結果 rf 的精確度為 0.7355 較諸分為四群，精確度又略微下降。不知分作八群，成效又是如何？

## 三、 分為八群

```
    > cwt <- read.csv(file.choose())
    > data <- cwt[, -9]
    > kmeans.cluster <- kmeans(data, centers=8)
    > cwt$group <- print(kmeans.cluster$cluster)
    > kmeans.cluster$withinss
    [1] 35776.71 31580.98 86797.69 43630.97 34365.73 24633.12
30418.16 73262.78
    > table(kmeans.cluster$cluster, cwt$quality)
         1  3  4  5
    1    1 18 66  2
    2    0 10 43  1
    3    0  5 10  1
    4    7 20  0  0
    5    0 15 40  0
    6    3 29  8  0
```

```
        7   3 27 46   0
        8   0 12 42   2
```

traindata 訓練結果，精確度為 0.7388834。

testdata 預測結果，精確度為 0.8099。

表 1：9 個變數與 10 個變數比較

| 演算法 | 9 個變數 | 10 個變數（加入 group4） | 10 個變數（加入 group6） | 10 個變數（加入 group8） |
|---|---|---|---|---|
| rf 演算法 | 0.7934 | 0.7686 | 0.7355 | 0.8099 |

分為八群的結果，精確度一下子提升不少，竟然超過 80%，果然加入分群新變數是有益於提高預測的精準度。雖然這樣的結果已超出本人預期之外，但研究就是要秉持著精益求精的原則，若有改善的方法，自然不容錯過。目前的自變數項目有九個，若再添上「優雅詞彙」和「切題詞彙」兩個新變數，精準度是否又會更往上提升？

## 玖、 加入優雅詞彙、切題詞彙新變數

縱然筆者無法像資工背景的學者一樣，擁有獨立撰寫程式的能力，但筆者身為國文教師，對於文章優劣的評量能力還是有的。作文除了從段落、全文字數、代名詞、語副詞的多寡綜合評量之外，一篇文章是否切題，或遣詞造字是否得體優雅，仍然可以使之量化而成為新的變數。文章是否切題，主要在於他是否寫出與題目相關的詞語。以本次 CWT 全民中檢的初等寫作測驗而言，它的統一題目是「我想說的話」，由這個題目所開展的內容，首先必與「想說的話」有關，因此「想說的話」即是最切題要的關鍵詞。但「想說的話」也可以用其他方式表現，不一定要直白寫出「想說的話」，例如「有些話藏在心裡」，「說不出口的話」，或者「難言之隱」、「苦衷」……等等亦可間接暗喻想說的話。

其次，「我想說的話」也必然隱含一個說話的對象，此對象可以是一人或多人，或社會大眾、國家民族都行，即使對象是全宇宙也不算離題，甚至反身自躬，以自己為對象，說話給自己聽也行，只要有個明確對象即可。若缺乏明確對象，僅是空談浮論心裡有話應當說、必須說，以免空留遺憾等等理論，則非題目之宗旨，這即是離題、不切題。是以這個題目的典型寫法即是有具體的想說的內容，而且有個明確之說話對象，能同時兼具二者，至少分數有三級分。根據題目旨意延伸出來的對象有可能是父母、朋友、手足、情人，或者是冤家敵人也行；而從中可能會

說的具體內容，對父母可能會說感恩的話、道歉的話，對著情人、朋友可能訴說想念、思念之情，因此「感謝」、「感恩」、「對不起」、「想你」、「我愛你」等等之類的話語便可列入切題詞彙。筆者整理了可用為本題目的切題詞彙有：「有些話」,「想說的話」,「說不出」,「說出口」,「感謝」,「懷念」,「對你說」,「對他說」,「感激」,「感恩」,「謝謝」,「心裡話」,「我愛你」,「想念」,「一句話」,「我愛你們」,「難言之隱」,「苦衷」。

綜合前面九個變數，如今再增添兩個新變數，依照分為四群、六群、八群的演算結果，rf 的精準度變化如下：

表 2：四群、六群、八群比較

| | 分為四群 | | 分為六群 | | 分為八群 | |
|---|---|---|---|---|---|---|
| | traindata | testdata | traindata | testdata | traindata | testdata |
| **rf 演算** | 0.73841 | 0.8264 | 0.73103 | 0.8099 | 0.75826 | 0.7521 |

測試結果，以 K-means 分為四群的成效最好。

總結以上所有類型的運算結果，成效最佳的莫過於隨機森林演算法，茲以隨機森林為主，從原本的九個變數，其次增添分群變數，再其次增添兩個新變數，全部的預測演算結果綜合如下：

表 3：全體比較

| | 變數數量 | 資料集 | | 演算結果 | 精確度排名 |
|---|---|---|---|---|---|
| 9 個變數 | 9 個變數 | traindata | | 0.77920 | |
| | | testdata | | 0.7934 | 3 ★ |
| | 10 個變數（增加 group） | 四群 | traindata | 0.76219 | |
| | | | testdata | 0.7686 | 4 |
| | | 六群 | traindata | 0.76541 | |
| | | | testdata | 0.7355 | 5 |
| | | 八群 | traindata | 0.7388 | |
| | | | testdata | 0.8099 | 2 ★ |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | ★ |
| 11 個變數 | 11 個變數 | traindata | | 0.73506 | |
| | | testdata | | 0.8099 | 2 ★★ |
| | 12 個變數（增加 group） | 四群 | traindata | 0.73841 | |
| | | | testdata | 0.8264 | 1 ★★★ |
| | | 六群 | traindata | 0.73103 | |
| | | | testdata | 0.8099 | 2 ★★ |
| | | 八群 | traindata | 0.75826 | |
| | | | testdata | 0.7521 | 6 |

## 壹拾、　結論

研究如何使用軟體程式自動評量作文，此為筆者從事機器學習研究的首次實驗結果，雖然預測度未能達到 90%，但有高於 80%的準確度也著實令人欣慰，這暗示著使用電腦程式評量中文作文並非不可行，筆者所擷取的特徵變數是有效的。只不過若要達到更精確的準確度，有必要再增減一些特徵變數，例如把連詞變數納入進來，去掉某些關連性低的變數，改用另一種分群分析法來自動分類，試著運算字詞之間的關聯性等等作法，說不定能將精確度衝高至 90%。這些改進的運算都是筆者日後所要精進的工作。

至於筆者所選擇的變數項目乃是基於多年批閱作文的經驗，並無確實的統計量化根據，畢竟使用 R 語言自動評量作文為國內首創之舉，之前並沒有可參考的資料；而且這項工作也帶有實驗性質，筆者只能不斷測試何種變數具高效性，何種變數是低效性，持續增減變數項目，以提升精準度。

此研究結果對筆者的教學工作也十分有幫助，在教導學生如何寫作文自然有助益，若校方能使用筆者所建構出來的演算模式進行自動批改，豈非節省巨大人力！即使筆者所建構的模式未能達至百分之百，至少可以先過濾一些粗劣、不合格的作文，然後再做精細的批閱，同樣也能減少人力成本。

本論文的研究成果之延續、擴展尚不止於以上所說。此次研究所蒐集的都是非結構化資料的文字，非結構化資料較諸結構化資料更難以處理，而且在資料探勘（Data Mining）中，非結構化的資料往往比結構化的資料多出百倍、千倍不止，因此資料探勘的趨勢必然是朝向非結構化

資料的蒐集，然後轉化為結構資料，再予以演算分析。若缺乏這項技能，就不用奢談資料探勘了。而這項技能筆者於本次研究中充分學習到了，這項技能定然對筆者日後的探勘研究有莫大益處。

其次，筆者所研究的與文字有關，雖然稱不上「文字探勘」，但文字探勘的首要工作——中文斷詞，筆者已經能夠輕鬆駕馭，循此繼續深入文字探勘的學習，應該不是太困難的事。文字探勘（Text Mining）在當今網路普及、網站林立、電子郵件廣泛使用下，再加上社群網站如雨後春筍一個一個冒出來的時代，倍顯重要。試想想，全球的網頁、Facebook、Twitter、E-mail、Line 每天所增加的文字資訊量會有多少？肯定相當驚人，而這些都是屬於半結構或非結構化的資訊，卻潛藏大量有用的資訊，如何把這些潛藏的資訊發掘出來，便有賴於文字探勘了。

文字探勘目前被廣泛應用在文件摘要、概念擷取、資訊過濾、意見分析、情緒分析、關係探索、文本分類、文本分群等議題[16]，由於有如此強大功能，現在文字探勘也被運用在行銷當中，特別是客戶管理，可用文字探勘分析客戶的流動率。此外，文字探勘當然也能運用在學術研究，尤其是中文學術研究，無論是文學、哲學、史學，運用文字探勘技術重新發掘以往不曾探討過議題，或者經由探勘掘發出來的潛在訊息，都可以讓學術研究的方向更加多元化，更能展現傳統學術研究的新風貌。本人在楊教授的指導下，完成這一主題的論文研究，在新知識、新技能的學習上，在個人的學術研究上，收穫甚為豐富。

[16] 謝邦昌，2017，**Text Mining 文本探勘**，元華文創，臺北，頁 1-4。

## 參考文獻

三津村直貴，2018，圖解 AI 人工智慧大未來：關於人工智慧一定要懂的 96 件事，台北：旗標科技。

李仁鐘、李秋緣，2017，R 語言資料分析：從機器學習、資料探勘、文字探勘到巨集資料分析，博碩文化，臺北。

許菱祥，1986，中文文法，大中國圖書公司，臺北。

黃文、王正林編著，2016，利用 R 語言打通大數據的經脈，佳魁資訊，臺北。

游皓麟，2017，實戰 R 語言預測分析，松崗資產管理，臺北。

謝邦昌，2017，Text Mining 文本探勘，元華文創，臺北，頁 1-4。

麥爾荀伯格、庫基耶，2013，大數據，台北：天下遠見。

余思翰，2008，中文作文寫作輔助系統(碩士論文)，國立交通大學資訊科學與工程研究所，新竹市。

# A Study of Automatically Marking Chinese Composition by using Machine learning

**Teng, Hsiu-Mei**

Associate Professor/ TransWorld University

## Abstract

Big data analysis and AI artificial intelligence is popular at this time, almost every field has their trail, everything in the world, almost everyone tries to computerize and automate manual work with big data analysis calculus. So if the application of big data analysis in the correction of Chinese composition, I do not know whether it works? If student's Chinese composition can be automatically graded through the calculation mode, this is a great benefit exactly for Chinese teachers!

To achieve this goal, this paper attempts to use the R language to deal with this research. The first work is to break words, and marking part of speech, then extract the elements related to the good or bad of the articles, set as feature variables, convert unstructured text into structured data. The kit program I am using is jiebaR. Second work is to select feature variables, from the first nine variables, gradually increase until twelve variables, there are a total of twelve variables to calculate the prediction.

The third step is to choose the algorithm. This paper selects six algorithms and compares the prediction accuracy. Training and forecasting with nine variables, ten variables and twelve variables, to see which algorithm works best with how many variables are predicted.

**Keywords：Machine learning, R language, Chinese Composition, automatically marking, unstructured data**