Final Project Report—臺灣海洋廢棄物預測

Team member

109061517 邱俊嘉 / 109061520 陳俊宇 / 109061807 吳亞澤

**-Title:**

臺灣海洋廢棄物預測

**-議題簡介:**

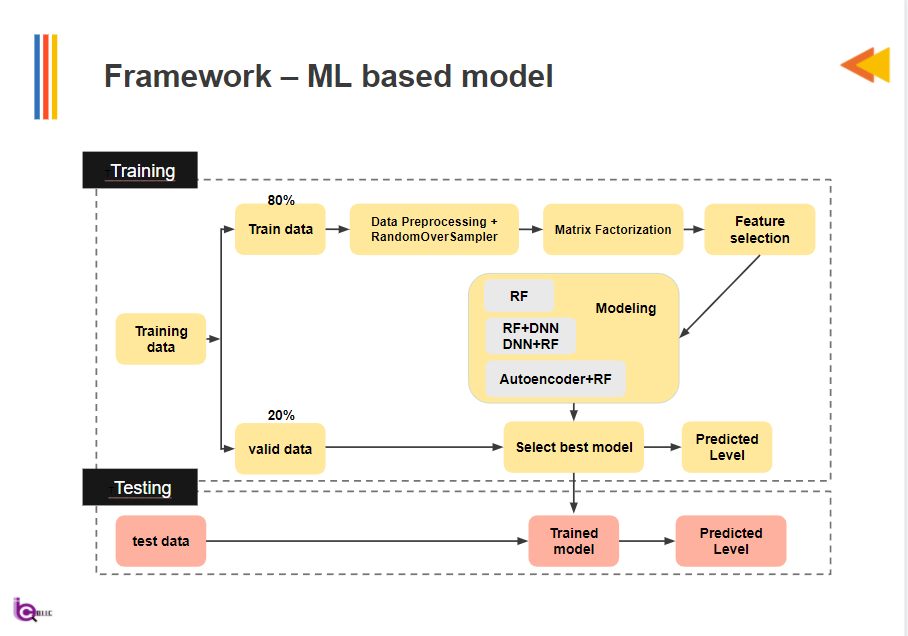
海岸廢棄物快篩調查可在短時間內做大範圍的抽樣調查，並量化廢棄物，可作為測量的方法之一，供淨灘選址參考。快篩的抽樣方式為於海岸線每隔 10 公里取一測站，以臺灣本島 1,210 公里海岸線為母體，即有 121 個測站。本議題希望藉由測站資訊預測相近測站的資訊，以達到減少測站和人力。

**-關鍵字:**

機器學習、深度學習、資料分析、神經網路模型、隨機森林模型、支援向量機。

**-Methods:**

**總流程**

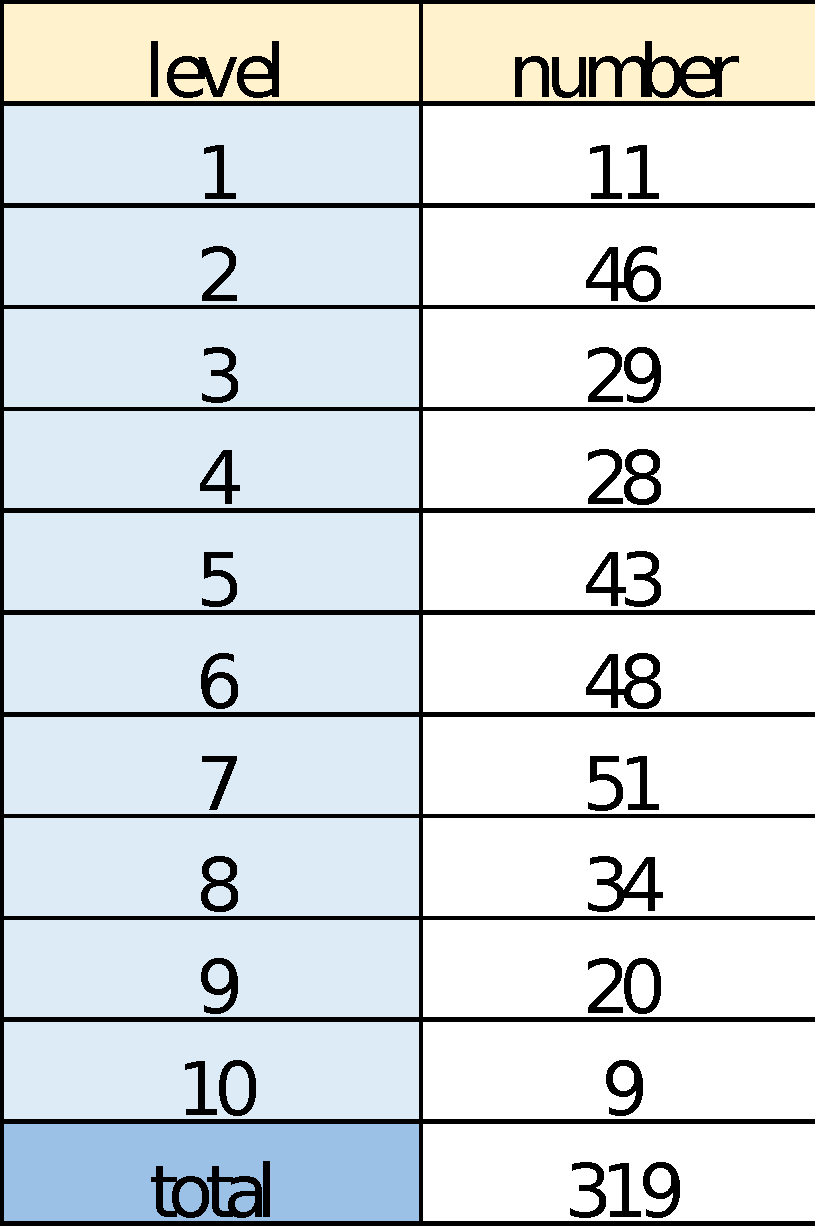
****

1. **資料處理與分析**

根據我們過去的經驗用AI方法來處理問題時，feature是非常重要的，同一個train data如果處理成不同feature，最後的performance甚至可能會差到十幾%，所以feature要如何改良是個關鍵

* 1. **train dataset**

下圖為AIea提供的train dataset之數量分布



總共有319筆sample，共10種level(海廢等級)，各level的數量分布不均勻，因此會做upsample讓每個level的數量都複製增加到51筆，所以最後總train data數會是510筆，這樣可以避免data imbalance的問題，不然model會傾向猜數量最多的那類

* 1. **缺失值處理: 以同一個地點的其他3個season的眾數來填補缺失值**

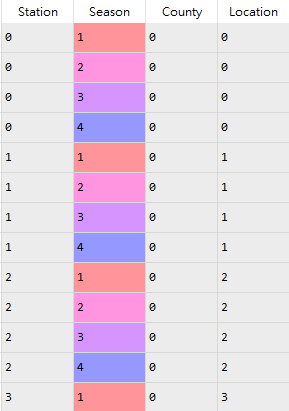
某些sample的部分特徵值可能會有缺失值，如下圖所示，一開始是用-1來取代缺失值，這個做法可能對於model的學習來說不太好，而且其實缺失值還蠻多的，佔了約25%，應該要更妥善處理，所以我們後來有想到一個的解決方案，那就是以同一個地點的其他3個season的眾數來填補缺失值，因為經過觀察，發現缺失值都是海岸地形種類，從直觀上來想，海岸地形因為會影響人類活動，理論上來說應該也會和海廢等級有關，因此海岸地形這個feature應該要用到，所以缺失值勢必要用某個值去填補，不能直接去掉

此外，我們還發現缺失值會發生在每一個地點的第4個season，而且同一個地點的其他3個season的海岸地形種類通常都很接近，代表第4個season很有可能也是同一個海岸地形，這其實蠻合理的，畢竟同一個地點的海岸地形通常不會因為season的不同而變化，不過同一個地點在不同season時，某些海岸地形會略有不同，所以我們最後決定以同一個地點的其他3個season的眾數(多數決的方式)來填補缺失值



* 1. **特徵值若為中文或英文就以正整數代號取代**

station、location和country這三個feature的特徵值為中文或英文，這會讓model無法學習，所以我們將其改成從1開始的正整數代號，如下圖所示

 🡺 

* 1. **將是否為同一個系列測站的資訊納入考慮**

如果一個station名稱就用一個數字代號的話，那就跟location feature一模一樣了，因此我們後來改成英文代號相同的station就視為同一個station並使用同一個數字代號，數字代號一樣從1開始，這樣就可以將是否為同一個系列測站的資訊納入考慮

舉例來說，E01和E02都視為E系列的station，所以他們的代號都是1

* 1. **將train data和test data還原成raw data**

我們發現train data和test data是由raw data拆分而成，所以其實train data和test data中的sample是有順序關係的，可以根據station將test data插入到train data的對應位置，即可還原成raw data

舉例來說，test data中的E04 station的4個sample可以插入到train data的E03和E05之間，這樣就可以考慮到鄰近關係，轉換出來的數字才會比較合理，如下圖所示

| **+  =**  **train data test data raw data** |
| --- |

* 1. **處理同一個地點，但是County或Location不一樣的情況**

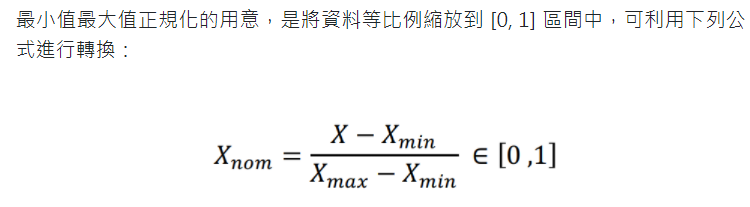
還原成raw data後會發現data中的County和Location很常出現誤植的情況，像是下圖的例子，明明是同一個地點，但是County或Location卻不一樣，這會造成後續轉換成代號時，model會將其視為不同的地點，就有可能會出問題，所以我將其都轉成同一個中文後才轉成數字代號(中文相同數字就相同)

|  |
| --- |
|  |

* 1. **將所有feature做個別的Min-Max Normalization**

因為不同feature的數值分布範圍差異甚大，某些feature只有0或1，某些feature從0到87，如果不做Normalization的話，某些feature值會dominate整個model的學習，導致model學不起來

原本是用z-score standardization，但是z-score standardization會將分布會強行轉成標準高斯分布，原本的分布可能會跑掉，所以當原本的分布很不對稱時會出問題，因此後來改用Min-Max Normalization，公式如下圖所示，每個feature都會按比例轉換成0~1之間的數



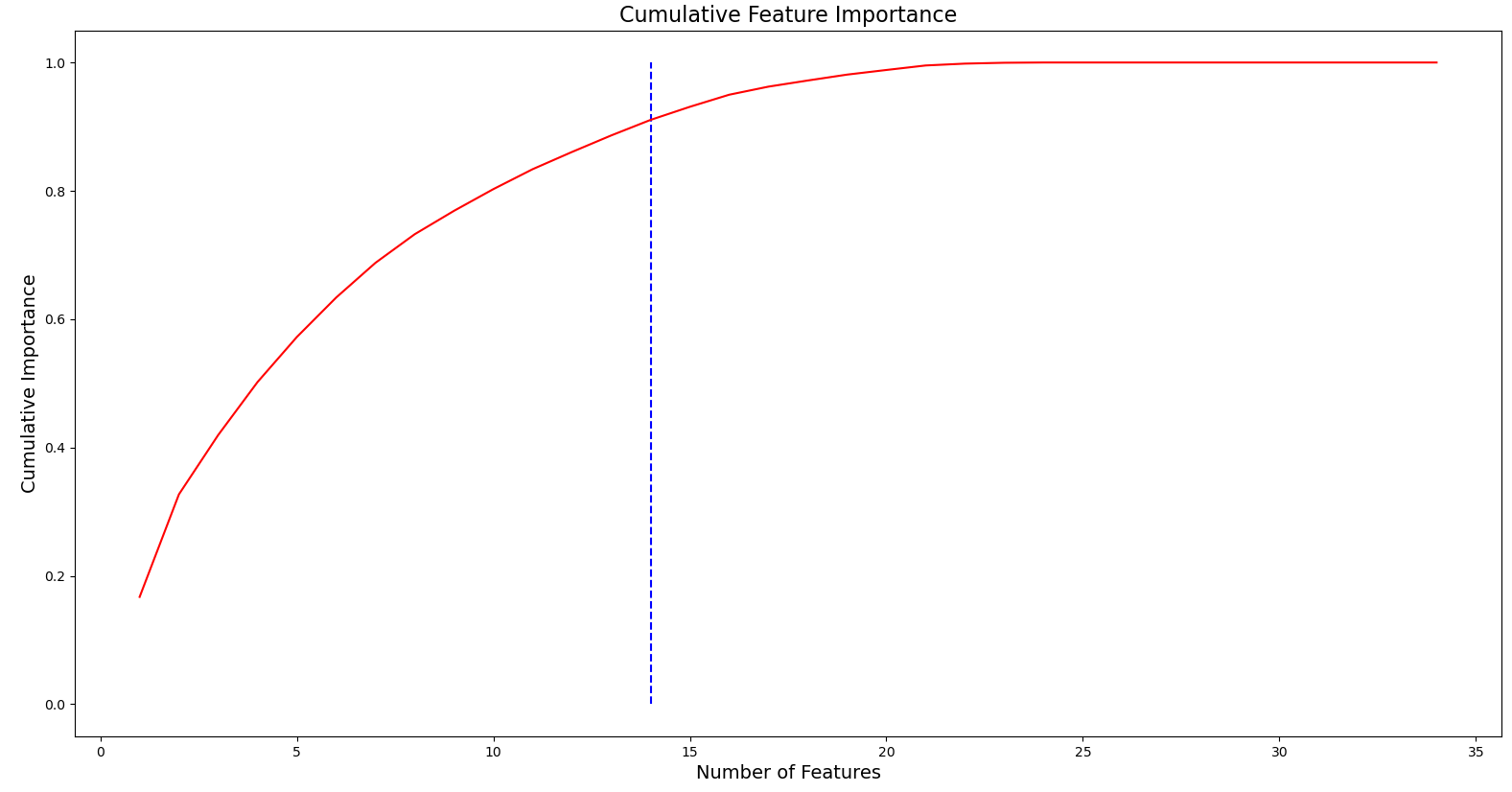
* 1. **做feature selection找出關鍵feature**
* 海廢等級代表海岸的汙染程度，海廢等級越高就代表海岸越髒，因此海廢等級和垃圾量有關，而垃圾量和人類活動有關，而人類活動可能和地點、季節、地形(ex: 沙灘)、垃圾類型這些feature有關，所以我們可以去求這些feature和海廢等級的關係，像是feature importance、correlation等，藉此找出關鍵feature
* 我們使用一個名叫feature-selector的package當作特徵選擇工具，feature-selector是由Feature Labs的一名數據科學家williamkoehrsen在2017年寫的，在github有1900多個star
  + feature-selector會用train data訓練出一個Gradient Boosting machine(GBM)，然後由GBM得到每一個feature的重要性分數，再對所有特徵的重要性分數做normalize
  + 為了使計算得到的feature重要性分數具有很小的方差，GBM會訓練10次，取多次訓練的平均值來當作最終的feature importance
  + 同時為了防止過擬合，會從train data中選一部份data當作validation data，在訓練GBM的時候，計算GBM在validation data上的accuracy，當accuracy達一定epoch都沒有上升後，就停止訓練GBM
* 經過計算後，各feature的feature importance如下圖所示

|  |
| --- |
|  |

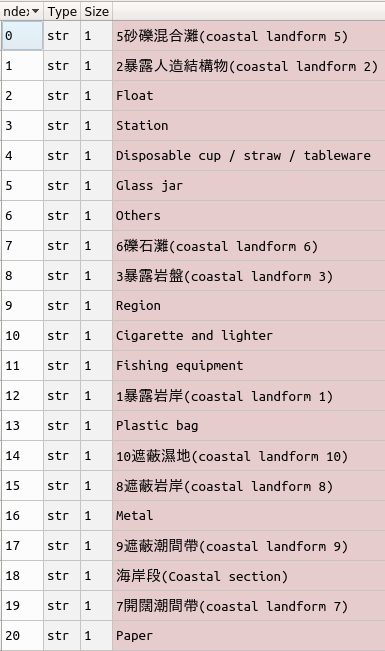
* 因為train data只有319筆，但是feature卻有34種，這樣很容易會產生過擬合的情況，必須適度地將一些多餘的feature去除，因此我們會根據得到的feature importance和correlation將以下三種feature去掉
  + zero importance feature
    - 對模型預測結果毫無貢獻的feature
    - 下圖為zero importance feature



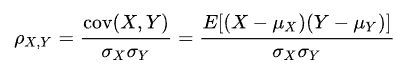
* + low importance feature
    - 先將feature importance由高排到低，依次將feature importance做疊加，當累積和達到0.9後剩下的feature就視為low importance feature去掉，所以zero importance feature 也必定包含於low importance feature中，這有點類似PCA中留下主要分量去除不重要分量的概念
    - 下圖中的藍色虛線和紅色實線的交點所對應的cumulative feature importance為0.9，所以只會留下藍色虛線以左的feature



* + - 下圖為low importance feature



* + high correlation feature
    - 若兩個feature的相關係數超過0.95，就將feature importance比較低的feature去掉，因為相關性超過0.95就代表這兩個feature非常接近，其實只需要一個當代表即可，不然可能會導致過擬合
    - 這邊的相關係數為Pearson correlation coefficient，公式如下圖所示，其值代表兩個feature之間的線性相關(相依)程度，其值介於-1與1之間，越接近1或-1就越相關

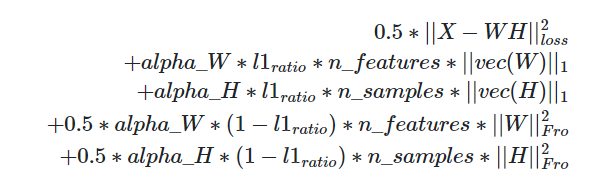


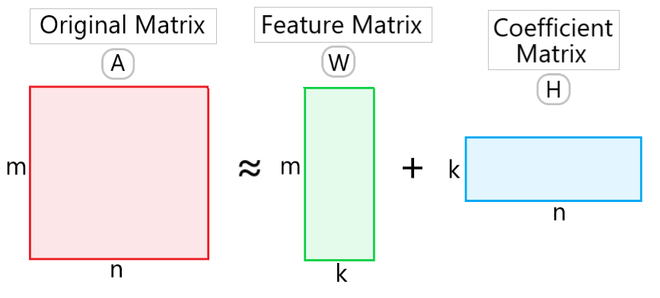
* + - 下圖為high correlation feature和其對應的correlation coefficient

|  |
| --- |
|  |
|  |

* 1. **Non-Negative Matrix Factorization**

由於此資料庫含有大量binary的特徵(0/1)，從"曝露岩岸"到"Others"一共22維的特徵皆屬此類。這些特徵形成的矩陣又多為稀疏矩陣，也就是說裡面只有少部份為1而大部分為0，如果拿此特徵進去模型學習會造成模型學習困難，因此我們將採用在機器學習中常拿被量化稀疏矩陣的方法學Non-Negative Matrix Factorization對此類特徵進行量化。以下為此演算法的公式與示意圖：





W為我們最後想要求的矩陣，而我們可以決定此W的維度有多少，藉以達到降維和從稀疏矩陣中提取重要特性的目的。在實作上，我們採用了sklearn的工具包幫助我們，並依序用2,4,8三個不同的W維度來進行稀疏矩陣的量化。我們會先把34維中後22維的矩陣分別變成2,4,8維(最終變成14,16,20維)，再套用1.8的方法來進行重要特徵的篩選，而我們最後給模型的維度=8維。

1. **模型架構**

我們的model主要分成2大類，一類是純機器學習方法，像是Random Forest，另一類是加入neural network概念的深度學習方法，像是encoder + Random Forest、DNN + Random Forest、Random Forest + DNN，下面會做細部說明

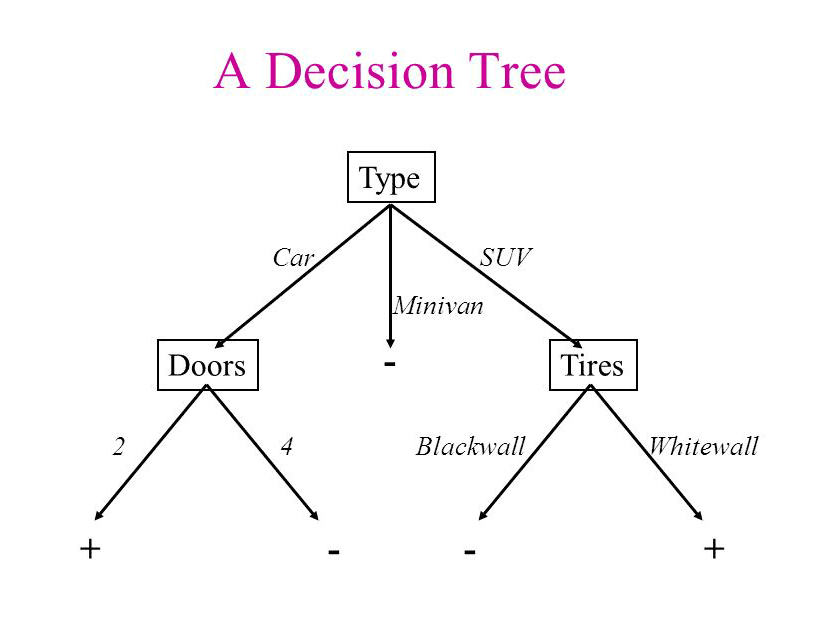
* 1. **隨機森林模型(Random Forest)(RF)**
     1. **目的**

random forest classifier的時間複雜度較低，為O(n\*log(n)\*d\*k)，也更適合處理高維度的資料。在分析上能顯示各個特徵的重要性，還能利用oob error快速評估模型的表現，非常適合用於本次的實驗

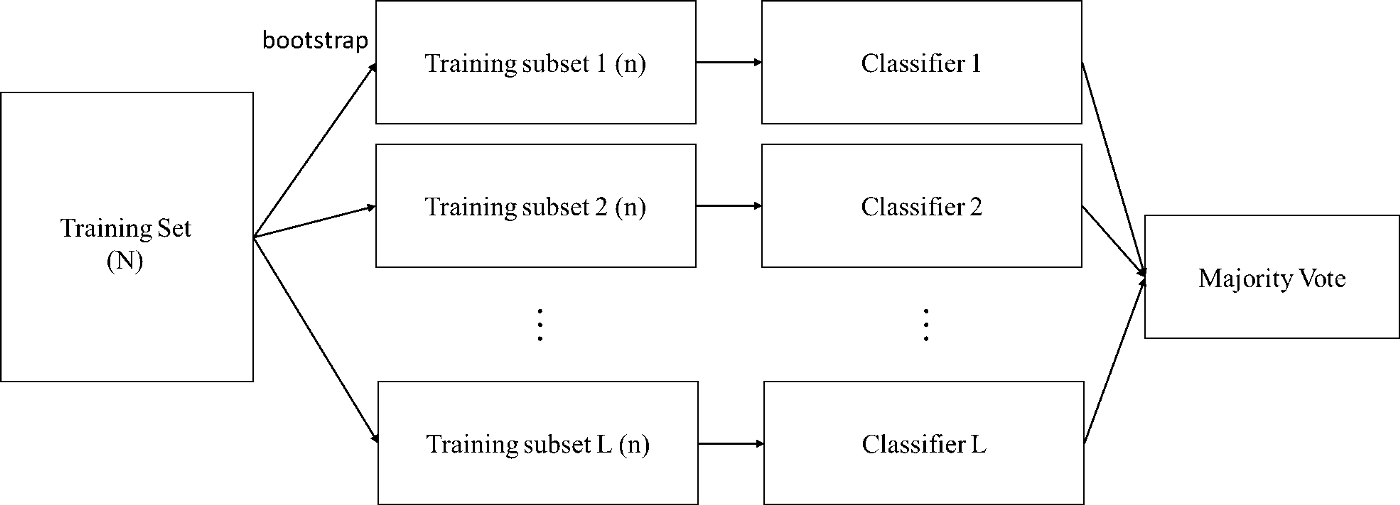
* + 1. **原理**

在講Random forest之前，必須要先介紹Decision tree和Bagging這兩個概念

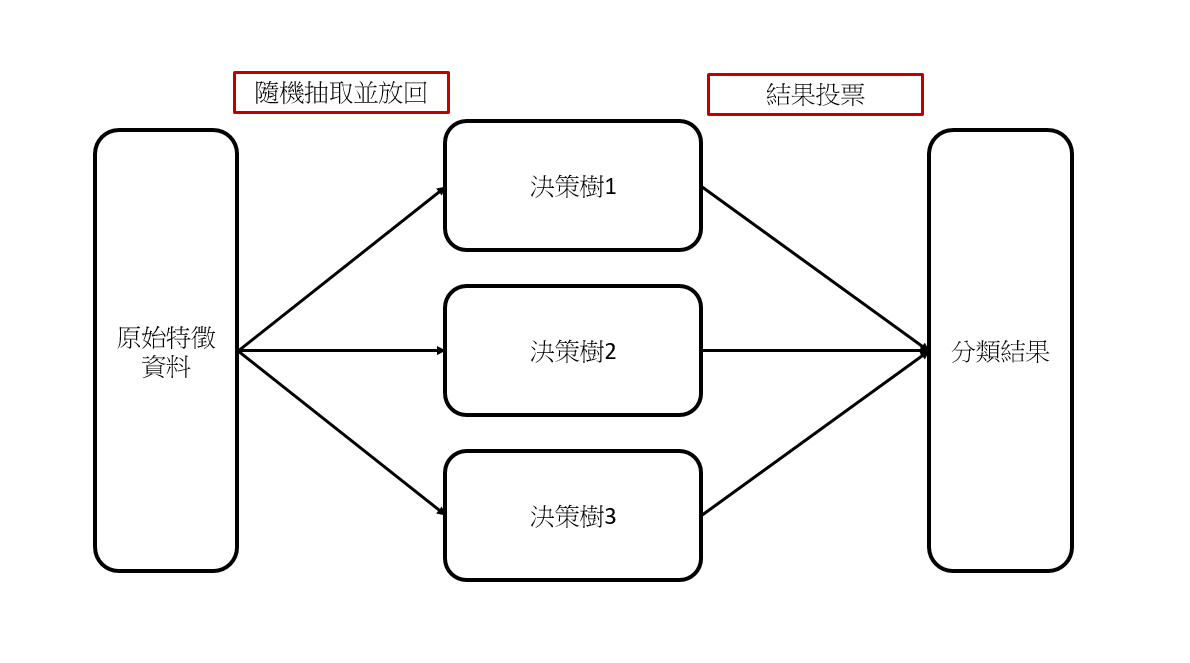
Decision tree：透過一層一層的決策，逐步篩選出符合的結果，如下圖所示



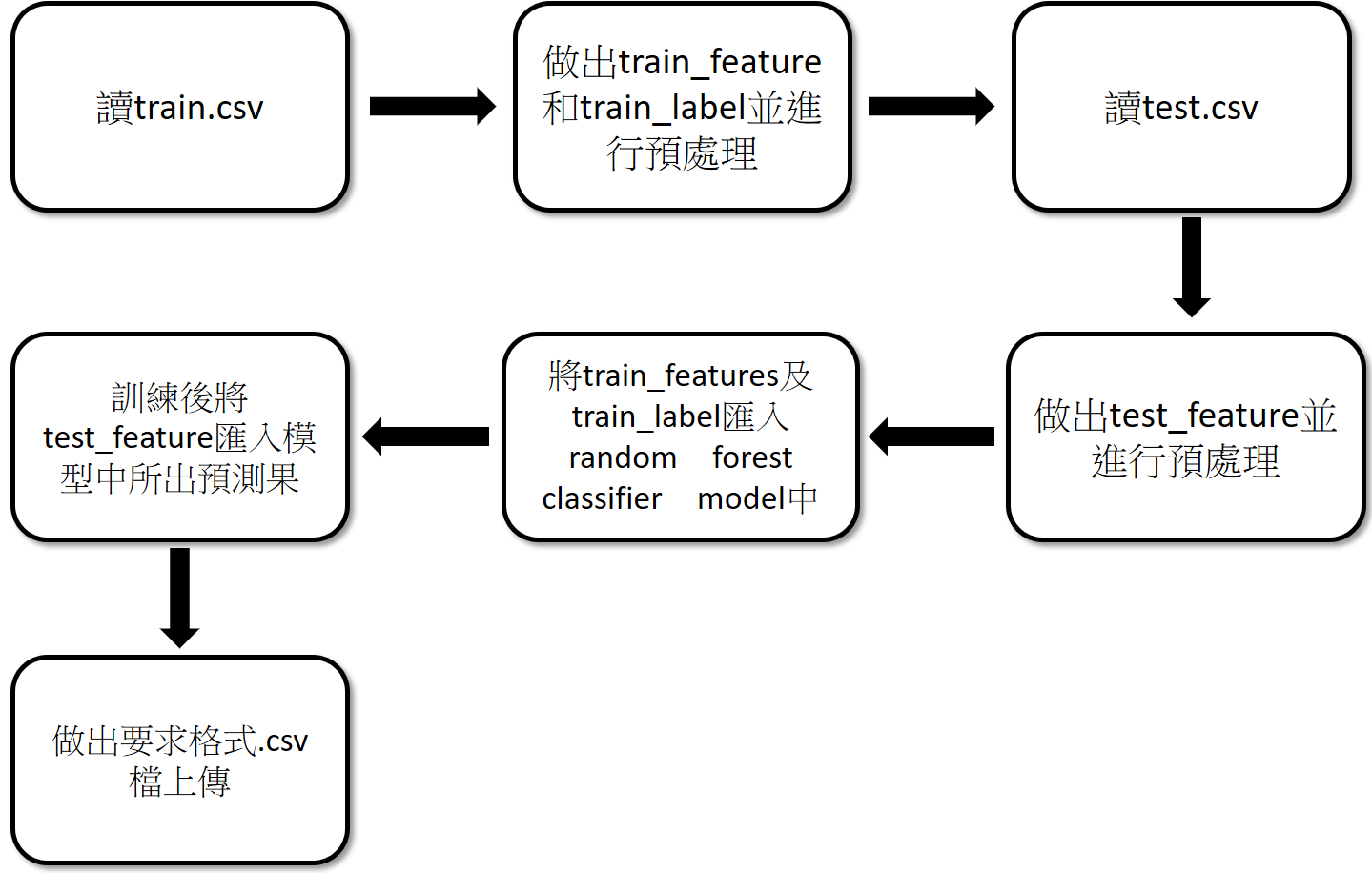
Bagging：從訓練資料中隨機抽取，取出後放回(bootstrap)，利用抽出之樣本訓練多個分類器，每個分類器的權重一致，最後用投票方式(Majority vote)得到最終結果，如下圖所示



Random forest：是一個結合bagging及decision trees的演算法，如其名字所述，由多個decision tree所組成，每顆決策樹獨立運算出結果，並透過投票得到最後的分類結果，如下圖所示



* + 1. **流程**



* + 1. **優缺點**

**優點**

1. 時間複雜度低，適合處理大量高維度的資料
2. 附有feature importance以及oob error等有利分析的功能
3. 訓練速度快
4. 能夠平衡失衡資料集的誤差
5. 對於缺失值以及離群值的敏感度低
6. 能夠避免overfitting
7. 能解決回歸與分類兩種問題

**缺點**

1. 對於資料數量少，或是低維度的資料，分類結果較差
2. 在某些noise較大的分類或迴歸問題上會過擬合
3. 相對於Decision tree，需要更長的時間以及更多的儲存空間作運算
   1. **深度神經網路(Deep Neural Network)(DNN)**
      1. **目的**

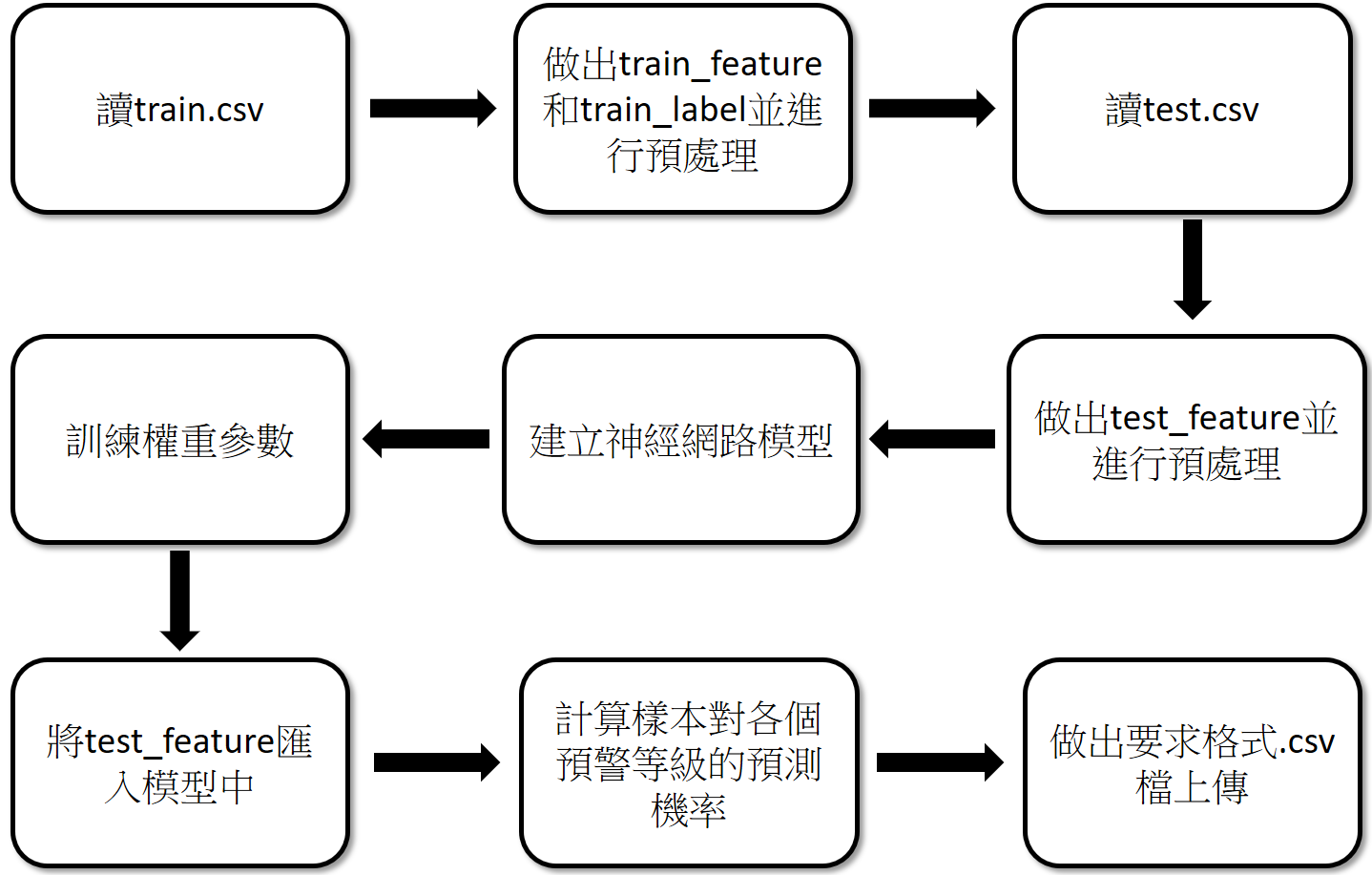
在機器學習的世界中，神經網路就像是人類的大腦神經結構，而神經元就像是大腦的神經細胞，是神經網路中最基礎的結構，在它們互相結合下，可以建構龐大的運作網路，能大大的提高預測準確率，也非常適合處理分類問題，故也選用此方法

* + 1. **原理**

1. 建立輸入層、多層隱藏層、輸出層和每個層的神經元，每個神經元都有一個輸出，其輸出稱為激勵值(介於0~1)，每一層神經元中激勵值的操作結果會影響下一層的激勵值，一層一層之間激勵值的傳遞最後即可得到輸出判斷結果
2. 前一層的每個神經元的輸出分別乘其對應的「權重值」最後相加 🡺 減下一層神經元的「偏置值」🡺 帶入激勵函式將其轉為介於0~1的「激勵值」
3. 數學形式:

0 ≤ 激勵值 = f(a1w1+a2w2+……+anwn - b) ≤ 1

* + 1. **流程**



* + 1. **優缺點**

**優點**

1. 可以建構非線性的模型
2. 有良好的推廣性，對於未知的輸入亦可得到正確的輸出
3. 可以接受不同種類的變數作為輸入，適應性強
4. 可應用的領域相當廣泛

**缺點**

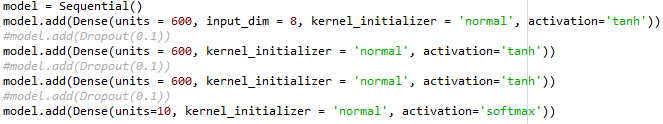
1. 以迭代方式更新鍵結值與閥值，計算量大，相當耗費電腦資源

訓練的過程中無法得知需要多少神經元個數，太多或太少的神經元均會影響系統的準確性，因此往往需以試誤的方式得到適當的神經元個數

* 1. **Encoder + Random Forest**

我們訓練了一個autoencoder來去對原始的feature做representation。我們用了簡單用了一層FC當encoder，和一層FC當decoder，並採用gelu當作activation function。loss的部分則是使用mean square error當作Loss function，最佳化的方法則是採用了adam。實作上，我們使用了三組不同的模型，將維度從原本的8維分別升到64、128、256維，當作最終分類器的輸入。

* 1. **DNN + Random Forest**
* 接4層的DNN(最後一層是output layer)來train 🡺 test在train data和test data上 🡺 取倒數第二層dense layer的feature embedding當作新的train data和test data 🡺 用random forest來train和test
* 除output layer以外，神經元數都是600
* 新的train data和test data的feature dimension為600
* 架構如下圖所示



* 1. **Random Forest + DNN**
* 與2.4相反，將Random Forest產出的predicted probability(10維)當作DNN的features做訓練，而DNN的output也是10維的結果，此作法用意在於透過DNN強化random forest的結果，看能否在testing data上有更好的表現。
* DNN架構為4層(包含input layer與output layer)，各層神經元個數分別為(10, 64, 32, 16, 10)，層跟層之間皆有做batch normalization，activate function皆為GELU，Dropout rate設定為0.2。

**-結果與分析**

* 下面的test kappa皆為AIdea public leaderboard的評估結果

| RandomForest | | | | |
| --- | --- | --- | --- | --- |
| Feature Name | Train Kappa | Train UAR | Train ACC | Test Kappa |
| no\_remove | 0.876 | 91.5 % | 89.03 % | 0.6523 |
| remove\_low\_importance\_and\_high\_correlation (RLIHC) | 0.8618 | 90.4 % | 87.77 % | 0.6753 |
| remove\_low\_importance | 0.7235 | 78.94 % | 75.55 % | 0.7059 |
| remove\_zero\_importance | 0.6122 | 70.55 % | 65.52 % | 0.754 |
| Non-Negative Matrix Factorization\_RandomForest | | | | |
| FA2, no\_remove | 0.4459 | 57.14 % | 50.47 % | 0.6612 |
| FA2, RLIHC | **0.5823** | **69.22 %** | **62.7 %** | **0.7789\*** |
| FA4, RLIHC | 0.6626 | 74.96 % | 69.91 % | 0.7678 |
| FA8, RLIHC | 0.609 | 71.34 % | 65.2 % | 0.6505 |
| RLIHC+upsampling | | | | |
| RandomForest, n\_iter=45 | 0.6707 | 75.55 % | 70.85 % | 0.6749 |
| DecisionTreeClassifier, n\_iter=40 | 0.9109 | 93.65 % | 92.16 % | 0.5305 |
| Gradient Boosting Classifier, n\_iter=100 | 0.603 | 70.38 % | 64.89 % | 0.5817 |
| Non-Negative Matrix FA\_RandomForest+upsample | | | | |
| FA2, RLIHC | 0.6673 | 75.59 % | 70.53 % | 0.6878 |
| FA4, RLIHC | 0.7089 | 77.62 % | 74.29 % | 0.6651 |
| DNN(RELU) -> RandomForest | | | | |
| FA2, no\_remove | 0.8727 | 91.95% | 88.71% | 0.6864 |
| FA2, RLIHC | 0.8063 | 87.82% | 82.76% | 0.7511 |
| FA4, RLIHC | 0.5923 | 69.79% | 63.64% | 0.6972 |
| RandomForest -> DNN(GELU) | | | | |
| FA2, RLIHC | 0.5726 | 67.48 % | 61.96 % | 0.7643 |
| FA2, RLIHC, weighted loss | 0.5361 | 66.57 % | 58.43 % | 0.6896 |
| Autoencoder (GELU) -> RandomForest | | | | |
| FA2, RLIHC, Hidden dim=64 | 0.6246 | 73.46% | 66.46% | 0.5093 |
| FA2, RLIHC, Hidden dim=128 | 0.4703 | 60.15% | 52.66% | 0.5527 |
| FA2, RLIHC, Hidden dim=256 | 0.5746 | 69.05% | 62.07% | 0.5458 |
| FA2, RLIHC, Hidden dim=512 | 0.5706 | 68.58% | 61.76% | 0.5861 |

| no\_remove\_train/test\_feature+upsampling | | | | |
| --- | --- | --- | --- | --- |
|  | Train Kappa | Train UAR | Train ACC | Test Kappa |
| RandomForest, n\_iter=45 | 0.693 | 77.07 % | 72.73 % | 0.6885 |
| DecisionTreeClassifier, n\_iter=40 | 0.9216 | 94.24 % | 93.1 % | 0.6241 |
| Gradient Boosting Classifier, n\_iter=100 | 0.6744 | 75.87 % | 71.16 % | 0.4532 |

* **non upsampling和upsampling比較**

觀察training data我們發現有嚴重的data imbalance問題，我們有嘗試透過RandomOverSampler將training data的各種類別做隨機上採樣至同樣數量，再當成模型輸入做訓練，而這樣的做法train kappa可以逼近1，但是test kappa會很差，表示過擬和於training data，所以我們後來並沒有採用upsampling來開發模型。

* **Feature selection (FS)效益討論：**

由Table可得知，有採用feature　selection的結果會比較好，其中又以remove\_zero\_importance的表現最為突出，相比完全沒用的**test kappa上升0.1017**。

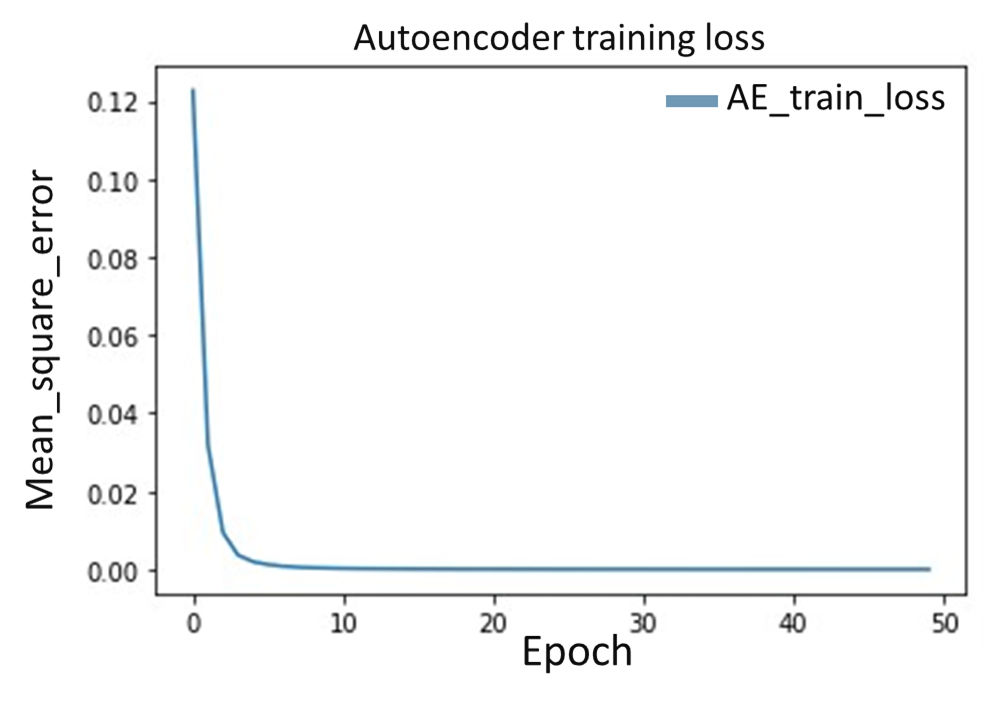
* **Matrix Factorization (FA)效益討論：**

Factorization的結果則是remove low importance and high correlation (RLIHC)的平均表現較優秀，其中又以**降到兩維之後再取FS**的結果最為突出，比只做**FS再上升0.290**，**test kappa達0.7789**。這也是我們最好的結果。

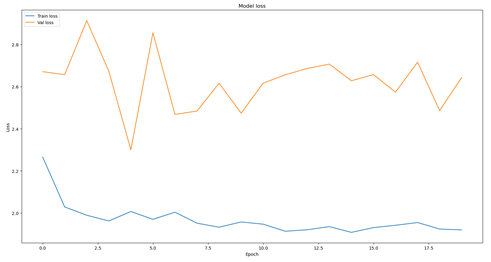
* **傳統機器學習模表現比較：**

RandomForest (RF)、DecisionTreeClassifier、Gradient Boosting Classifier三者中，還是以**RandomForest的結果最好**，RandomForest相對DecisionTreeClassifier與Gradient Boosting Classifier的(Train Kappa-Test Kappa)較小，表示較沒有overfitting的問題，所以我們後來選用RandomForest來測試不同Feature的表現

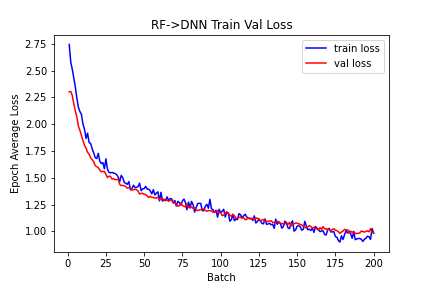
* **類神經網路表現討論：**
  + 我們進行了三種NN架構對該任務進行不同方式的訓練，並把training loss於下方呈現。雖然參數皆有收斂，但表現都沒有優於原始的FA+FS+RF，由此可以推測此**資料集的數量可能過少**，且**類別過多**，若使用NN造成**模型過於複雜**，導致test kappa降低。
  + RF + DNN的train kappa值與單純RF接近，但是test kappa會較差
  + 用DL方法後會變得比只有Random Forest差，看來DNN會把feature破壞掉，反而會幫倒忙，可能是因為有效train data數只有319筆的關係，導致DL方法無法發揮
  + Encoder + Random Forest



* + DNN + Random Forest



* + Random Forest + DNN



**-References**

1. PATTERN RECOGNITION AND MACHINE LEARNING
2. PYTHON機器學習與深度學習特訓班
3. Github
4. [資料分析&機器學習] 第4.1講 : Kaggle競賽-鐵達尼號生存預測-(前16%排名)

<https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC4-1%E8%AC%9B-kaggle%E7%AB%B6%E8%B3%BD-%E9%90%B5%E9%81%94%E5%B0%BC%E8%99%9F%E7%94%9F%E5%AD%98%E9%A0%90%E6%B8%AC-%E5%89%8D16-%E6%8E%92%E5%90%8D-a8842fea7077>

1. 我如何分析客戶流失預測？Kaggle比賽思路分享

<https://medium.com/finformation%E7%95%B6%E7%A8%8B%E5%BC%8F%E9%81%87%E4%B8%8A%E8%B2%A1%E5%8B%99%E9%87%91%E8%9E%8D/%E6%88%91%E5%A6%82%E4%BD%95%E5%88%86%E6%9E%90%E5%AE%A2%E6%88%B6%E6%B5%81%E5%A4%B1%E9%A0%90%E6%B8%AC-kaggle%E6%AF%94%E8%B3%BD%E6%80%9D%E8%B7%AF%E5%88%86%E4%BA%AB-daecd888a91>

1. [資料分析&機器學習] 第3.4講：支援向量機(Support Vector Machine)介紹

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

1. [機器學習 ML NOTES]Kaggle比賽心得(2%經歷)

<https://medium.com/@super135799/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ml-notes-kaggle%E6%AF%94%E8%B3%BD%E5%BF%83%E5%BE%97-2-%E7%B6%93%E6%AD%B7-7e8667cf1dc6>

1. Feature Engineering 特徵工程中常見的方法

<https://vinta.ws/code/feature-engineering.html>

1. Basic feature analysis (Date+Categorical+Revenue)

<https://www.kaggle.com/super13579/basic-feature-analysis-date-categorical-revenue>

1. R筆記–Ensemble Learning(集成學習)

<https://rpubs.com/skydome20/R-Note16-Ensemble_Learning>

1. 機器學習模型的時間複雜度

<https://kknews.cc/zh-tw/code/zyv254a.html>

1. 機器學習: Ensemble learning之Bagging、Boosting和AdaBoost

<https://medium.com/@chih.sheng.huang821/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ensemble-learning%E4%B9%8Bbagging-boosting%E5%92%8Cadaboost-af031229ebc3>

1. 隨機森林 RF 算法的原理（一）

<https://www.twblogs.net/a/5c8a02b2bd9eee35cd6a97fc>

1. 隨機森林(RANDOM FOREST)的底層概念、操作細節，與推薦相關資源

<http://notebookpage1005.blogspot.com/2018/03/random-forest.html>

1. 隨機森林（Random forest,RF）的生成方法以及優缺點

<https://www.itread01.com/content/1547100921.html>

1. feature selector官方code

<https://github.com/WillKoehrsen/feature-selector>