

# Data Science Term-project Report

Team: 10

Stu IDs: r10922121 r10922104 r10922142 b07901169

Stu Names: 林詩哲 賴昱宏 林子甄 楊宗桓

## 題目

Email Spam Classification

## Problem Definition

To predict the 5172 emails of spam or not-spam according to the frequencies of words in the emails

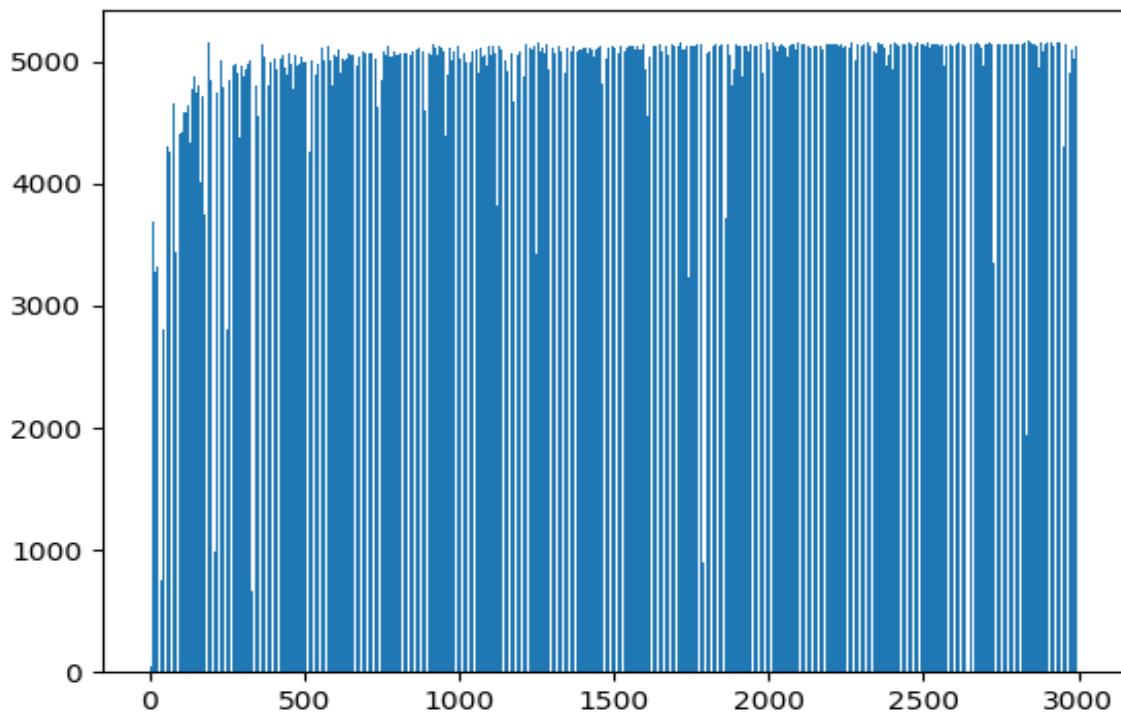
## Data Preprocessing

這個dataset只有一個csv, 裡面有5172個email對應到5172個row, 每個row記錄最常見的3000個字的數量, 而最後一個column記錄每個email是spam (1)還是not spam (0)。首先我們一開始去對整個dataset裡的值去做了一下觀察, 並將它們給output出來, 可以發現到因為是計算每個字出現的次數, 所以欄位裡並沒有任何的missing value的存在, 因此不需要去做有關於missing value方面的處理, 省去一些麻煩, 接著我們發現到了很多的字其實在大部分的信件中都是沒有出現的, 也就是說那個字所屬的那一個欄位中的值也絕大部分是0, 如圖

Email No.	the	to	ect	and	for	...	military	allowing	ff	dry	Prediction
Email 1	0	0	1	0	0	...	0	0	0	0	0
Email 2	8	13	24	6	6	...	0	0	1	0	0
Email 3	0	0	1	0	0	...	0	0	0	0	0
Email 4	0	5	22	0	5	...	0	0	0	0	0
Email 5	7	6	17	1	5	...	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...
Email 5168	2	2	2	3	0	...	0	0	0	0	0
Email 5169	35	27	11	2	6	...	0	0	1	0	0
Email 5170	0	0	1	1	0	...	0	0	0	0	1
Email 5171	2	7	1	0	2	...	0	0	1	0	1
Email 5172	22	24	5	1	6	...	0	0	0	0	0

[5172 rows x 3001 columns]

於是為了去證實我們的想法, 我們去計算了整個dataset中每個字在所有信件中出現次數為0的總次數, 得出的結果如下圖



可以看出來這3000個字基本上各自分別至少在一半的信件上是沒有出現過的，因此我們便想到利用feature selection去將裡面一些對於我們分辨垃圾信件沒有幫助的字去將它給篩選掉。

## Domain Knowledge

### Feature Selection:

#### **Naive one-by-one selection**

第一種feature selection的方法是naive one-by-one selection, 利用chi-square statistic去決定每個feature (word) 的重要程度, 也就是一個score, 再將所有的feature根據這個score去做一個排序, 再去選出前m個feature。

#### **Subset-based feature selection with genetic algorithm**

第二種方法是subset-based feature selection, 我們的做法的framework屬於wrapper approach, metaheuristic是採用genetic algorithm, 透過genetic algorithm挑到一組subset之後, 再利用naive bayes classifier去做evaluation, 判斷選到的這個subset的表現, 最後達到了一定的generation之後就停止。

### Model Selection:

#### **Naïve Bayes:**

首先一開始我們利用了最基本的Naïve Bayes來當作我們的model, 在

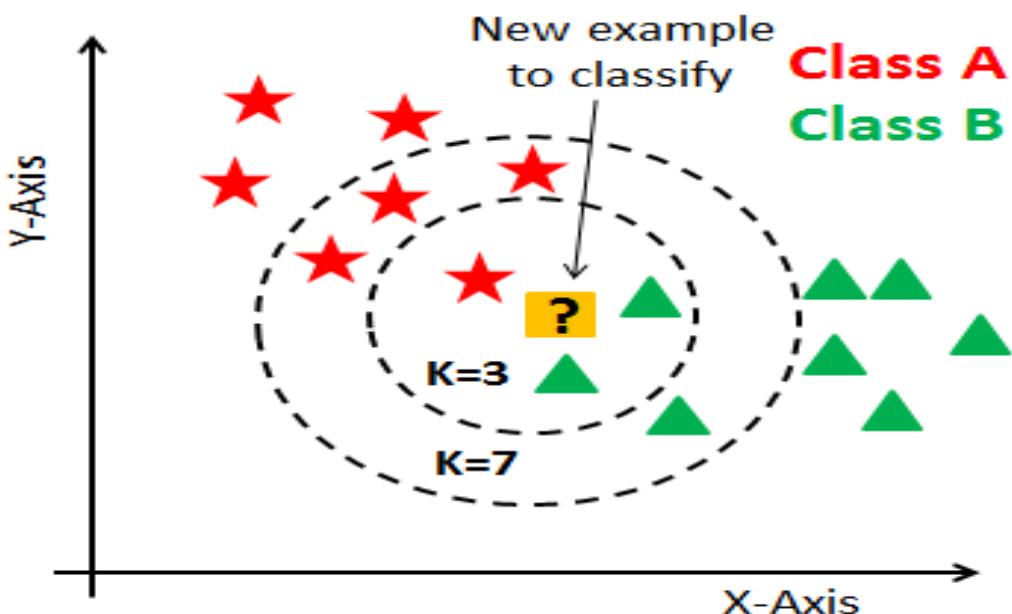
Naïve Bayes中是假設每一個字是conditionally independent的，也就是每個字之間沒有dependence的關係，而在我們整個dataset中的attribute是呈現離散的狀況下，我們使用到的是multinomialNB，適用於離散型的資料上面，基本上這些model都是以Bayes' theorem為原理去實現的，而這個Naïve Bayes本身也是具有一些優缺點，底下就大概列出其中幾項。

Advantages：易於實作，在大部分的狀況下可以得到不錯的結果

Disadvantages：由於model中有用到一些assumptions，因此可能會因此影響到出來的準確率，實際上，其實dataset中的attributes或多或少都存在著一些dependencies

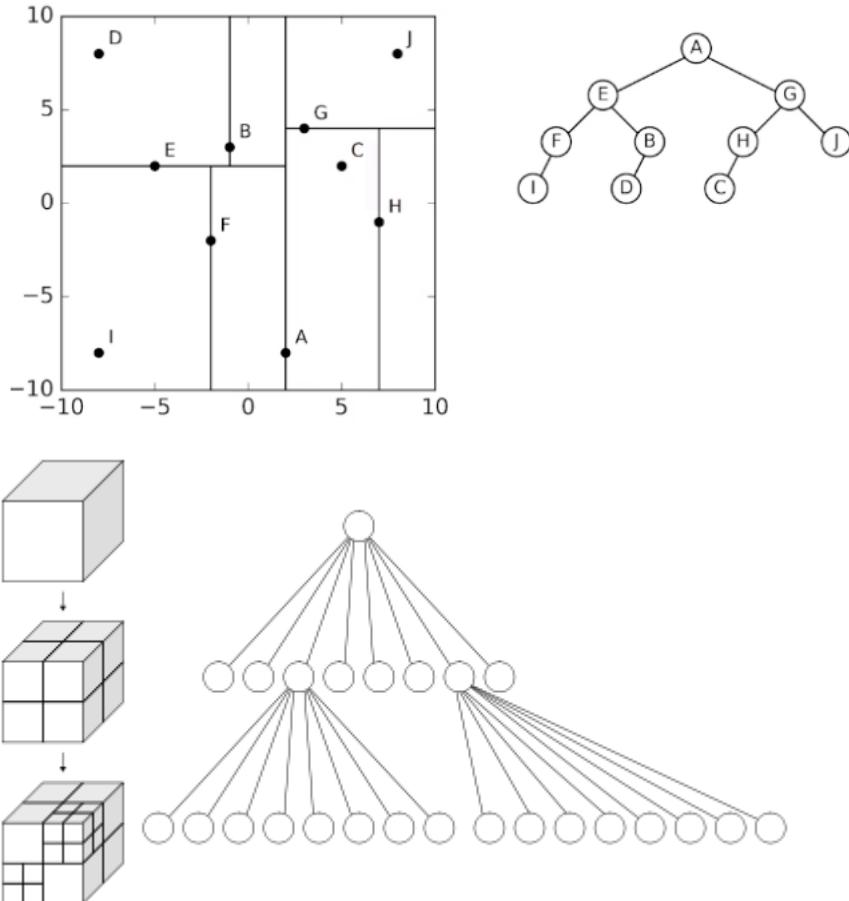
### K Nearest Neighbor

接著用到的model是K Nearest Neighbor也就是所謂的KNN，KNN是一個常被用來做分類的演算法，基本上KNN簡單來說就是"近朱者赤，近墨者黑"，透過分析鄰近的data中那一類class較多，便將我們的data assign給那個class，大致上的過程就如下圖



當K=3時，我們的data周圍是屬於三角形的数据最多，因此我們便將我們的数据分類到三角形這一類，若我們的K=7時，可以看到周圍的数据屬於星星的反而變成多數了，因此我們的数据便被分類到星星這一類，可以看出以上的過程就是周圍是哪一種類的数据，我們的数据就有很大的機率會是那一個種類的数据，可以看到K值的不同對於我們最後data所屬的類別會有所影響，因此在整個KNN的演算法過程中，K值的選取是十分重要的，如果我們將K值設定太小的話，雖說在data size很大的時候，這樣子的得到的結果是滿不錯的，但是若在data size沒那麼大的情況下卻容易造成overfitting的情況發生，然而，若我們將K直設的很大，那麼基本上我們的数据類別就取決於原先的dataset中哪一個class多了，也會增加我們的

computation cost, 因此K值得選擇是非常重要的，除了K值的選擇，如何去找到所謂的"最近"也是十分重要的一個問題，一般來說KNN中的"最近"的判斷都是利用Euclidean Distance，然而還有許多其他的距離計算方式，像是Manhattan Distance 和 Cosine Distance等等都是，因此我們在找我們想要的"最近"的data時，也要去思考那一種距離的計算方式所得到的結果才會是我們想要的結果，接著就是一些NN常用的data structure



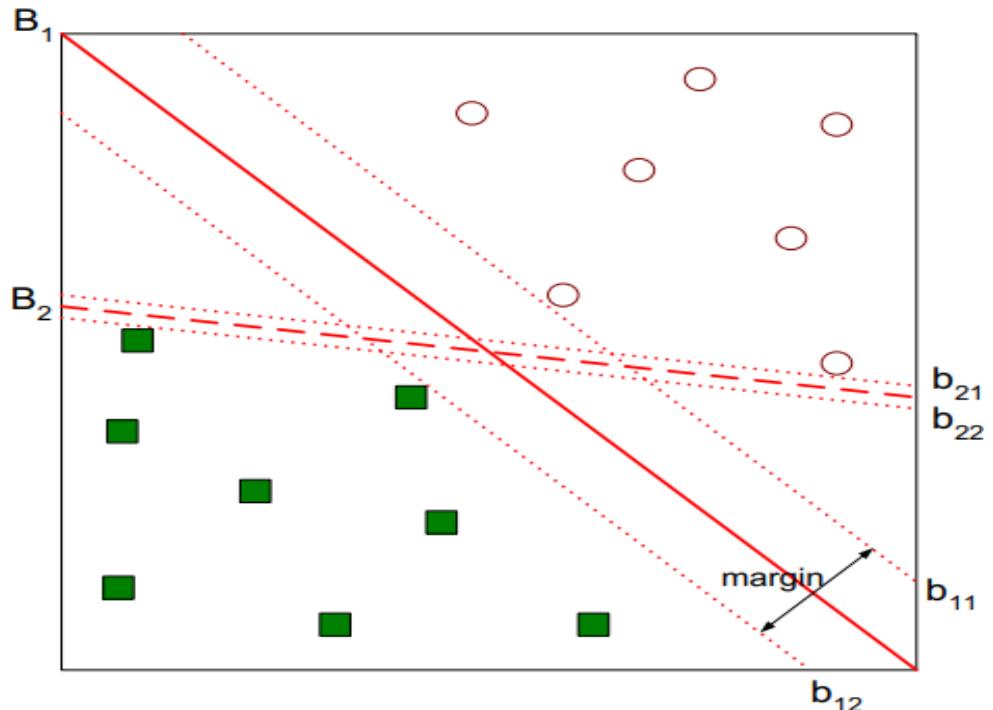
分別是KD Tree和Octree，基本上都是利用空間分割的原理，去把空間切成幾個比較小的區域，藉此去減少需要計算比較的data數量。而KNN的優缺點大致如下

Advantages : 易於實作，不用花時間training

Disadvantages : 在dataset太大或者是dimensions太高時效果不是很好

## SVM

第三個用到的model是svm，它的特點就是在所有可以正確分類的hyperplane中找出一個margin最大的hyperplane，由於svm的數學方面比較複雜這邊就不詳細解釋，其概念如下圖



就是要去maximize圖中的margin, 接著是關於SVM的優缺點

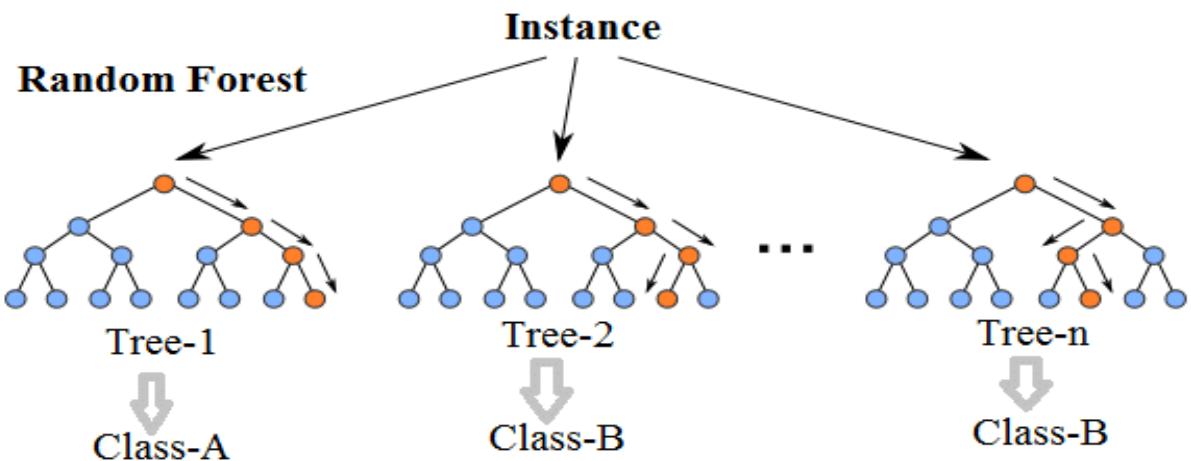
**Advantages :** 在high dimension時有不錯的效果, 在dimension比data samples數量還多的時候十分有效

**Disadvantages :** 不太適用於dataset size太大的情況之下, 在noise太多的狀況表現也不佳, 最後是所需要的computation cost滿高的, 因此十分耗時

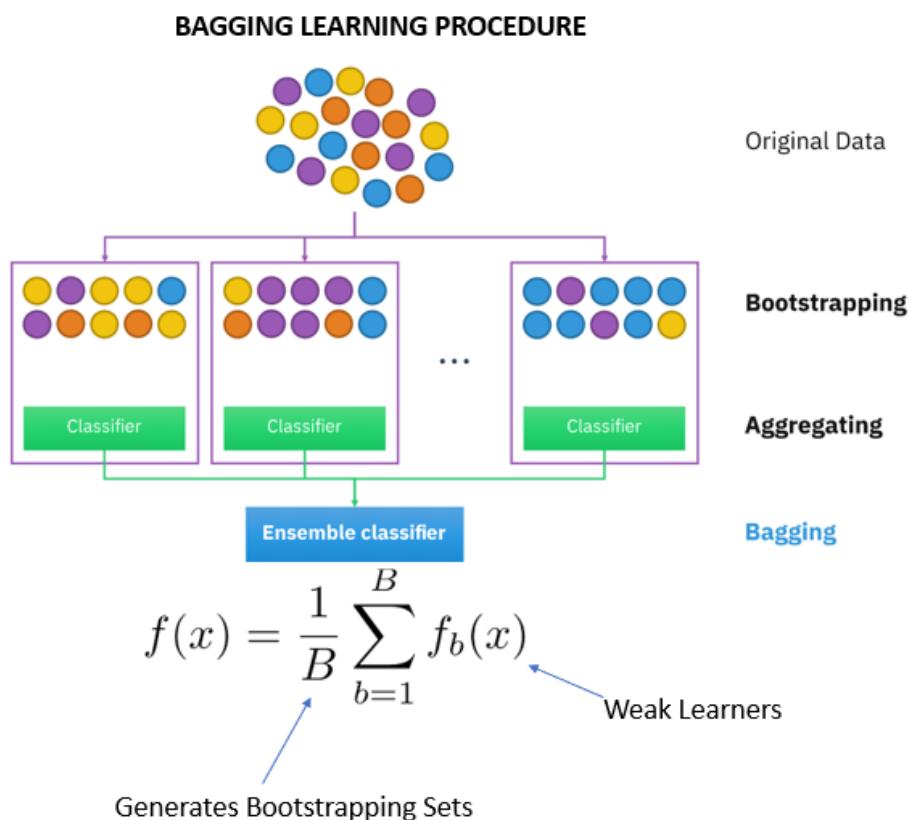
### Random Forest

最後是Random Forest, 它是一個常見用來做分類的model, 基本上random forest, 顧名思義, forest就是由許多的trees所組成的, 而random forest就是由許多的decision trees所組合而成的, 透過建出許多decision trees, 然後將我們的input data分別丟進這些decision trees中去進行分類, 接著將它們得出的結果去進行分析, 看是哪一個class的結果比較有可能, 再依這個結果去對我們的data進行分類。

## Random Forest Simplified



上圖就是一個簡單的random forest的例子，接著是bagging，一個在random forest中常用到的一個手法，簡單來說bagging就是bootstrap加上aggregation的結合，首先利用bootstrap去sample出幾個不同的dataset，接著透過這些dataset去訓練出不同的classifiers，接著將input丟到這些classifier中去做分類，得到結果之後再去將這些結果aggregate起來，去決定哪個class的機會較高，以此進行最後的assign。

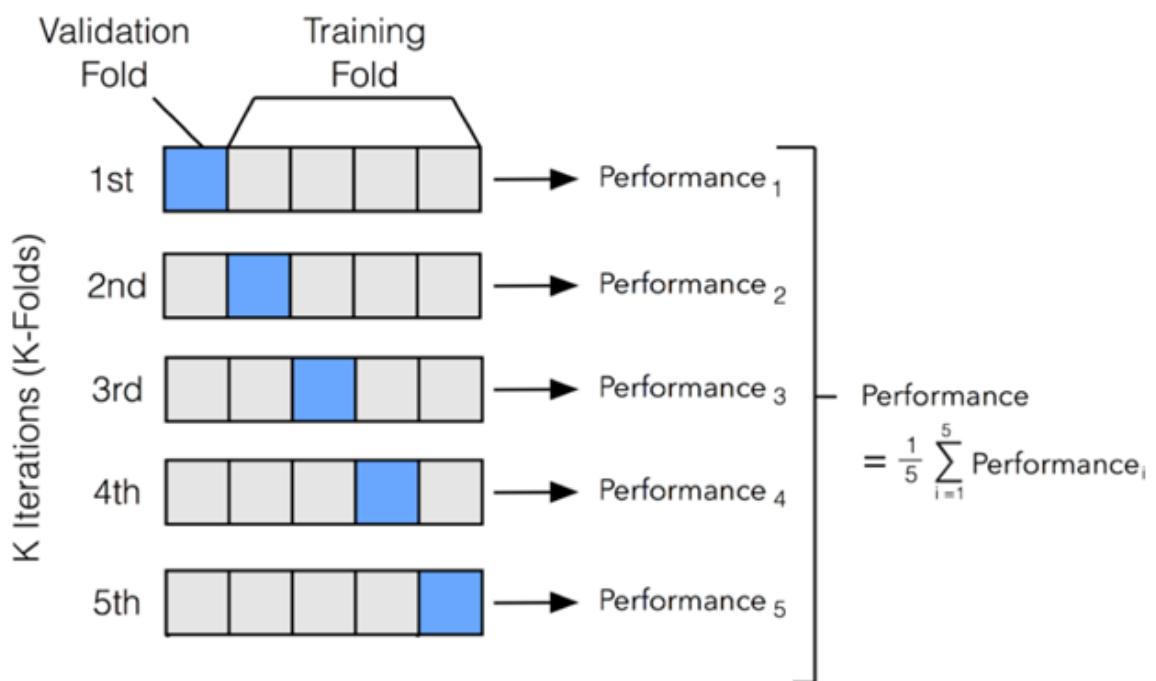


最後也是介紹一下random forest的優缺點

Advantages : 多樣性，可以用在classification和regression，在dataset size大的時候效果不錯

Disadvantages : tree的數量太多的話可能會使得計算時間太長，沒辦法及時地做出分類預測，容易有overfitting的狀況發生。

大致介紹了一下我們所用到的4個不同的Model，接著就是去對這些不同的model去進行evaluation，然而我們這邊並沒有test data，因此這邊使用的方法是K-fold Cross Validation，主要是透過將原本的training data切成幾個小部分，然後留下其中一部分當作test data，剩下的來去做training，最後訓練出的model再用前面的那一部分進行testing，這樣子拿原本training data中的資料去當作test data的過程就是Validation，而切成不同部分輪流test和train就是所謂的k-fold，我們便是利用此方式得到的validation error大小來判斷我們的model的performance。



上圖就是5-fold的示意圖，透過這樣的方式就可以讓我們在沒有test data的狀況之下對我們的model去進行評估。

## Hyper-parameter tuning

### Naive one-by-one selection

這裡可以調整的就是前面提到的m的值，也就是我們要選多少feature，我們有嘗試m的值從100到500，目的是要觀察當m到多少的時候會有一定的表現(例如accuracy超過90%)，因為當m比較小的時候training的時間比較短但是accuracy不太好，隨著m增加training的時間變長但accuracy會變好，所以這裡就要對training時間跟accuracy做一個tradeoff。

## Genetic Algorithm

這裡可以調整的就是generation跟population的數量，還有max feature number, generation設為50, population size設為100, 而max feature number有all、100、500和1000這4個，會去比較看不同的model哪一個比較好。

## K Nearest Neighbor

在KNN這邊我們調整的參數只有K值，我們將K值設成1,3,5,7,9然後利用這5個值去跑，然後將跑出來validation error最低的K值當作我們最好的K

## Support Vector Machine

SVM中我們的參數就只有kernel function的不一樣，分別是linear,sigmoid和rbf, 不過跑出來的結果顯示出linear的kernel function遠比其他2個還要來的好，因此SVM最好的kernel function是linear

## Random Forest

在random forest中能夠調整的參數就比較多了，而我們這邊調的分別是tree的數量以及tree的depth大小，以及我們不是用常用的gini index來當作split的判斷方式，我們是改用entropy來判斷，一般來說效果是差不多的，但entropy會稍微好一點點，不過計算會比gini index還要來的長

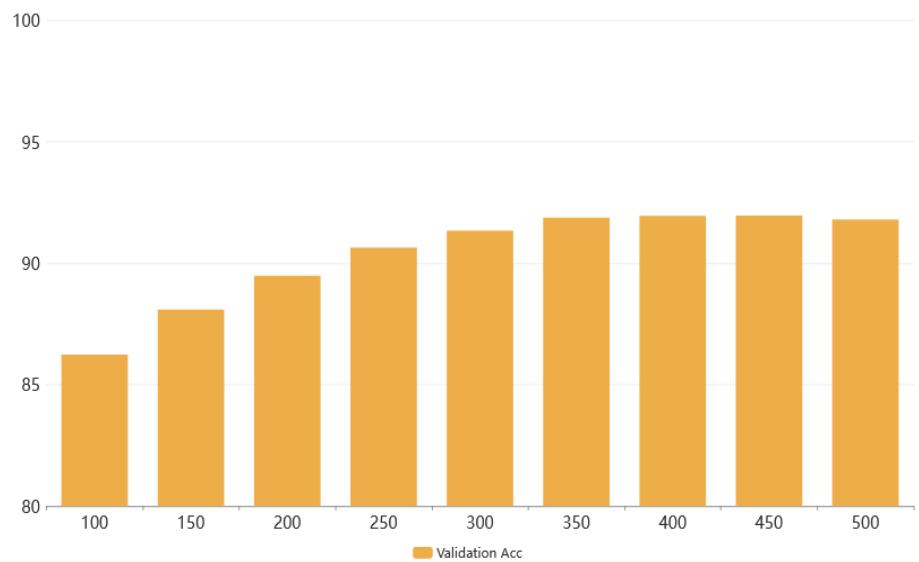
```
estimator_param = [50, 100, 150]
depth_param = [5, 10, 15, 20, 25, 30, 40]
```

上面是我們所設定的參數，透過迴圈去兩兩組合建出random forest，然後再根據validation error來判斷最好的tree數量和深度大小。

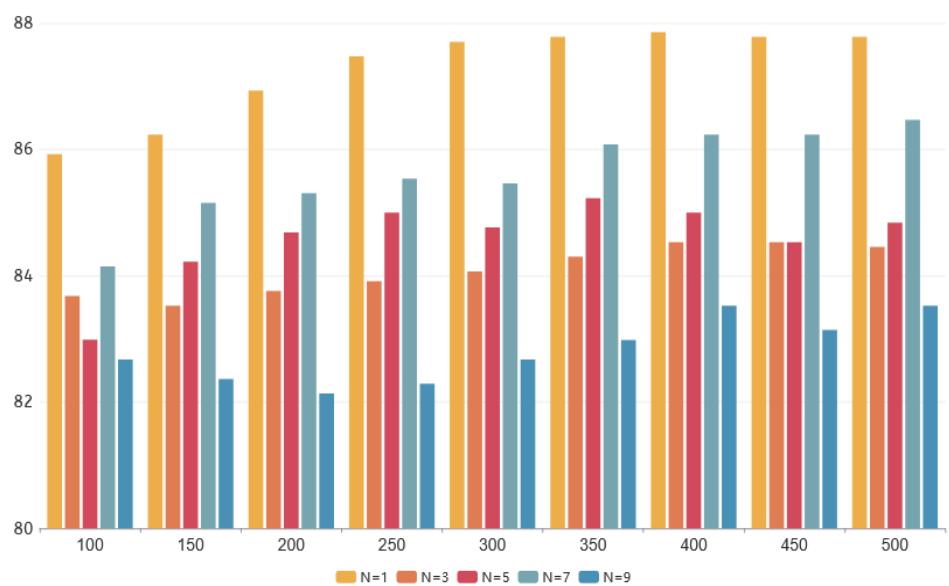
## **Evaluation**

### Naive one-by-one selection

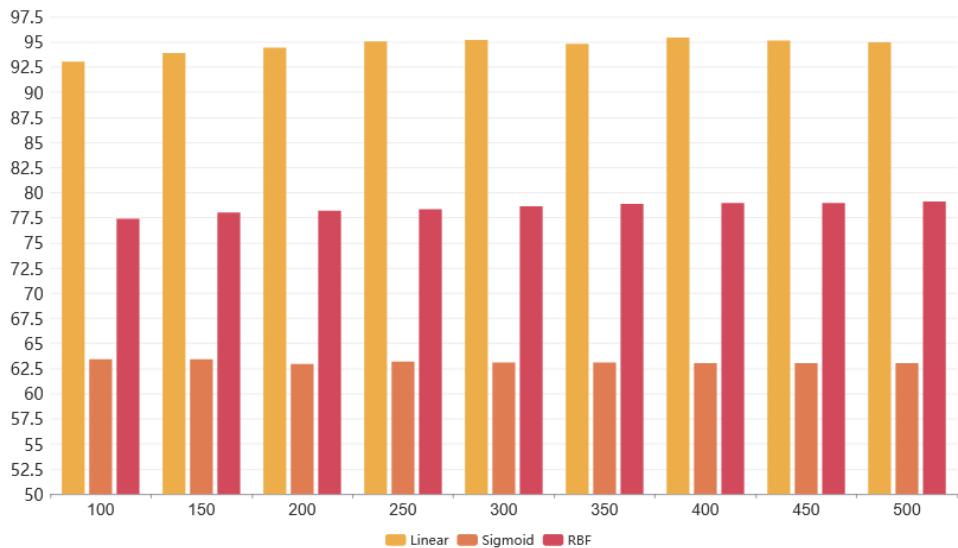
## NaiveBayes One-by-One



## KNN One-by-One



## SVM One-by-One



以上是Naive Bayes, SVM和KNN利用One-by-One Selection所得出的數據圖，可以大概看到最好的feature數量落在250~400之間，幾乎都呈現一個先增後減的情形，在KNN中我們去調整K值的大小，發現到正如前面所說到的K值的確會影響到我們最後的結果好壞

### Subset-based feature selection

這邊evaluation我們都有用genetic algorithm先做feature selection，參數如下圖所示

由圖中的結果可以看出，利用random forest可以達到最好的validation error, KNN的效果是最差的，KNN表現不好的原因可能是因為data size的關係，以及我們的feature只是簡單地去計算出單字出現的次數，並沒有進一步的去分析出更深的關係，因此沒辦法正確地找出我們所想要的"最近"的data，而Random Forest的表現其實也滿符合預期的，因為它對於簡單的分類問題是大部分能夠發揮出不錯的效果的，這也是它滿受到歡迎的原

因之一。

---

## Subset-based Feature Selection (Genetic Algorithm)

(generations: 50, n\_population=100, GA+NB)

max_features	all	100(93)	500 (485)	1000 (984)
naive bayes	93.9675	87.1616	94.0449	<b>94.4316</b>
svm(linear/sigmoid/rbf)	<b>96.2104</b> /61.794 2/81.8252	86.6976/66.279 9/83.9907	93.8902/62.258 3/82.2119	95.7463/60.324 8/75.7154
knn(1,3,5,7,9)	87.2390/86.156 2/86.7749/86.6 976/ <b>87.3937</b>	75.7154/82.753 3/82.5213/82.2 892/82.7533	84.6094/85.846 9/85.6922/84.6 094/84.8415	82.3666/83.062 6/83.2947/83.5 267/82.2892
rf(best)	<b>97.5251</b> (d=40,d=50)	89.7912(d=20,n=50)	94.8956(d=30,n=150)	95.9010(d=30,n=150)

n: estimators d: depth