HW2 成功大學資工系大四 F74109016 葉惟欣

1 目的 Abstract

這次作業中我利用多種不同之分類器 1. 決策樹(Decision Tree)、2. KNN(K-近鄰演算法)、3.邏輯斯迴歸(Logistic regression）、 4.貝式分類器(Naive Bayes Classifier)、5. 隨機森林(Random Forest)、6. SVM(Support Vector Machines)，7. MLP(Neural Network)透過 ROC 分析在不同模型下對資料集的預測結果，而此次主要目標為「利用民眾之個人基本資料，來預測此人是否會離職。」。

2 特徵設定 Setting

2.1 Input 特徵設計

1. 性別（0：男生, 1：女生）
2. 身高（男生平均：175 cm，女生平均：160 cm，標準差 20 以高斯分佈生成）
3. 體重（男生平均：65 kg，女生平均：55 kg，標準差 10 以高斯分佈生成）
4. 教育（1：小學以下，2：國中，3：高中職，4：大專院校，5：研究所以上）
5. 年齡（設定平均年齡：45 歲，標準差 20，以高斯分佈生成）
6. 每天平均上班時間（0~24 小時，每天平均:9hr，標準差 3 以高斯分佈生成）
7. 工作滿意度（1：每天無精打采，2：開會打哈欠很多次，3：回 email 速度很快，4：每周固定回報工作進度，5：與同事合作討論工作， 1~5 均勻隨機生成）
8. 薪資水平（平均 ：40000，標準差 15000 以高斯分佈生成）
9. 個人生活狀況（1：有小孩，2：沒有小孩，3: 有結婚，4: 沒有結婚）
10. 參加公司社團頻率（1：幾乎不去，2：偶爾去，3：經常去，4：每天報到）
11. 距離上次加薪幾個月?（0 ~ 24 ，以均勻分配隨機生成)
12. 員工參與度(1: 會跟同事一起加班，2: 會跟同事一起吃飯，
    # 3: 會參加公司活動，4: 出國會帶點心，5: 會接下外務)
13. 上級主管關係（1 ~ 10，數字越高，代表關係越好）
14. 出社會後跳槽平均頻率（0：一年內, 1：兩年內 2: 五年內, 3: 沒換過公司）
15. 十五. 公司職務 （1:助理,2:職員,3:組長,4:經理,5:處長,6:總裁,7 董事長）
    分布比率為 (0.1,0.2,0.3,0.2,0.1,0.05,0.05)
16. 健康度 (1:健康,2:生病,3:住院,4:生病住院)分布比率為 (0.7,0.1,0.1,0.1)

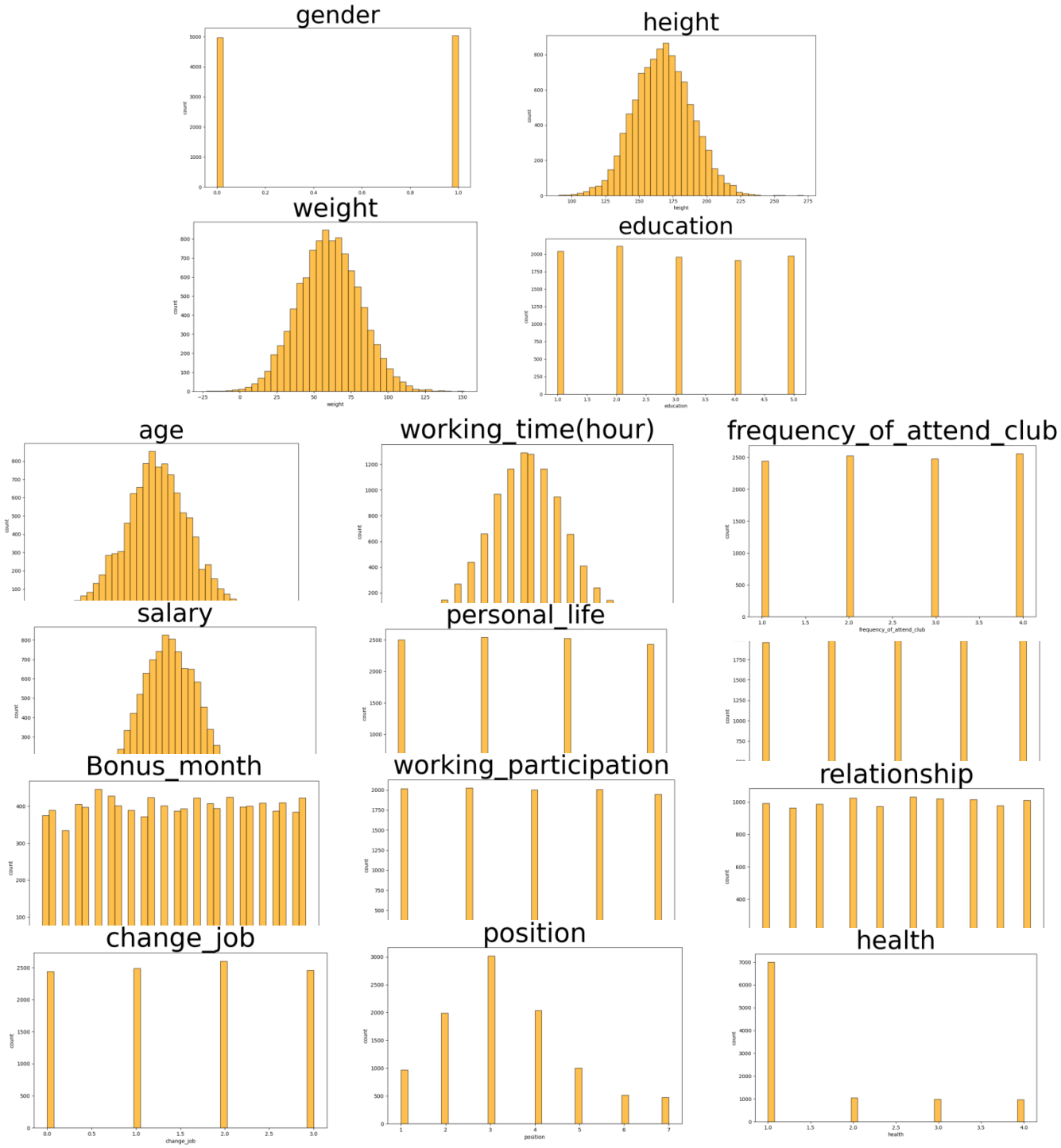
2.2 Output 特徵設計

是否會離職(離職 ：True, 不離職 ：False)，原先離職率為 0.1%

2.3 Absolute right rule

1. 性別（0： 男生, 1： 女生）
2. 身高（男生平均身高：175 cm , 女生平均身高：160 cm，以高斯分佈生成）
3. 體重（男生平均體重：65 kg , 女生平均體重：55 kg，以高斯分佈生成）
4. 教育（1：小學以下, 2：國中, 3：高中職, 4：大專院校, 5：研究所以上）
5. 年齡 ：>=65 離職率上升 20%， <=25 離職率上升 10%
6. 每天平均上班時間 ：>= 14 離職率上升 20%
7. 工作滿意度 ：<= 2 離職機率上升 20%
8. 薪資水平 ：<= 25000 離職機率上升 5%
9. 個人生活狀況 ：1 離職機率上升 20 %
10. 參加公司社團的頻率 ：3 與 4 離職機率降低 5 %
11. 距離上次加薪幾個月 ：>= 18 離職率上升 5% <=6 離職率下降 20%
12. 員工參與度(5: 會接下外務, 4: 出國會帶點心, 3: 會參加公司活動, 2: 會跟同事一起吃飯,1: 會跟同事一起加班) :離職率分別降低(20%, 10%, 5%, 5%, 5%)
13. 上級主管關係 ：<= 3 離職率上升 10%
14. 出社會後跳槽平均頻率（0： 一年內, 1： 兩年內 2: 五年內, 3: 沒換過公司）
    0： 一年內 離職率上升 10%
15. 公司職務
    （1:助理,2:職員,3:組長,4:經理,5:處長,6:總裁,7 董事長）:5 6 7 離職率為 0
16. 健康度 ：>=2 離職率上升 5 %

3 生成資料



透過等下決策樹 Decision Tree

Generate Fake Data:　# of leave :　1697　# of not leave :　8303

Generate Real Data :　# of leave :　1605　# of not leave :　8395

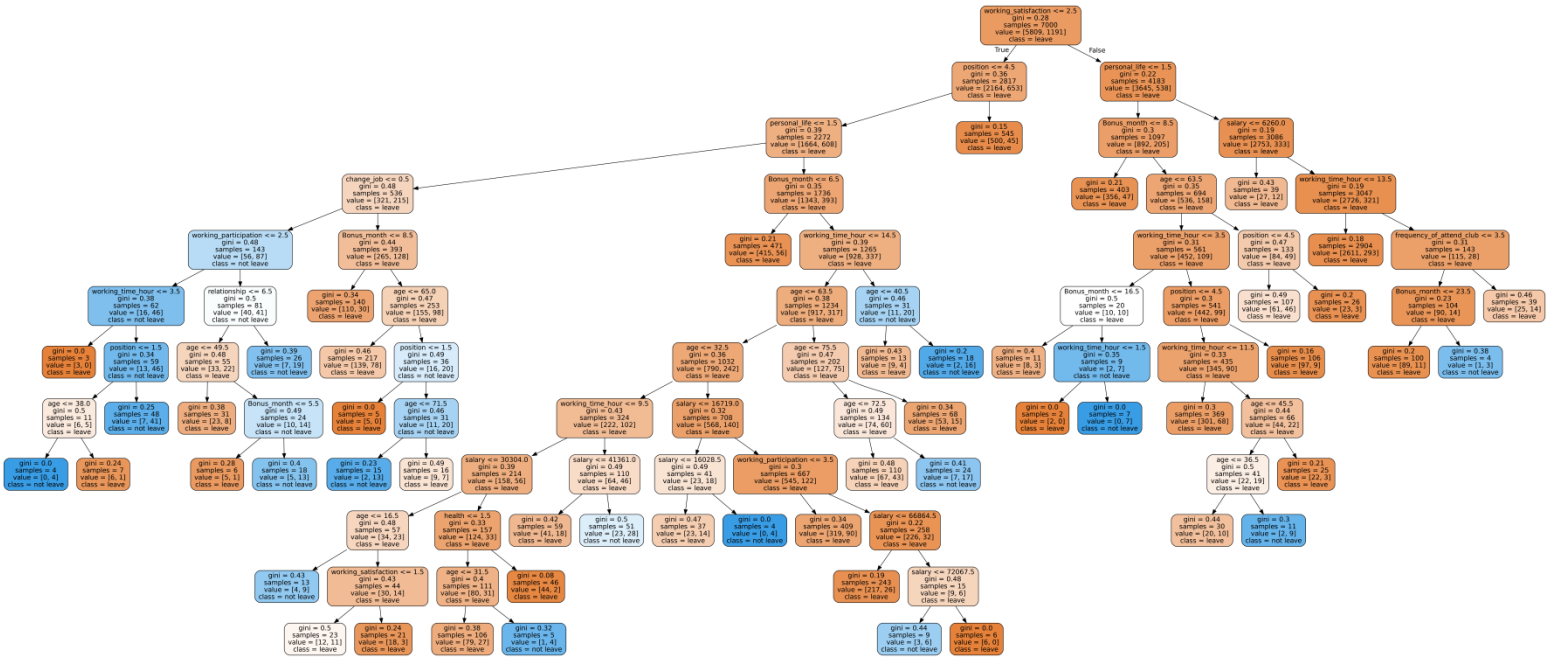
4 模型建立與評估 Model and Evaluation

Use the data generated in Step 1 to construct your classification model

本次作業選取了下列幾種 sklearn 套件中的模型進行建模與比較，依序為：4.1- 決策樹 Decision Tree、4.2- KNN(K-近鄰演算法)、4.3- 邏輯斯迴歸(Logistic regression、4.4- 貝式分類器(Naive Bayes Classifier)、4.5- 隨機森林(Random Forest)、4.6- SVM(Support Vector Machines)，4-7. MLP (Neural Network)透過 4-8. ROC 羅吉斯迴歸比較 7 個模型，先生成 10000 筆的資料，再透過 sklearn 套件中之 utils.shuffle 打亂排序，用 train_test_split 函數將資料分割成 7:3 的比例（7000 筆訓練資料，3000 筆測試資料），再將資料帶入 7 個不同的模型。此外也比較 fake dataset 的表現與 real dataset 的表現，並在最終比較其模型分類之差別以找出其中關鍵因素。

4.1 決策樹 Decision Tree

決策樹為一個一堆判斷式的問題將資料進行切割，為 if else。也就是當在做 predict 資料的屬性時，會用 binary search 的演算法爬過二元素 binary tree。而在訓練決策樹的方式則是透過 information gain 來作為判斷，當 training dataset 被一個屬性分割為兩個 child training dataset。會去判斷這樣的切分是否兩個 child training dataset 會更有明顯的區分。以這個題目，如果”年紀大於 65 歲”這個特徵能夠直接判斷此人會不會離職，那”年紀大於 65 歲”這個特徵對決策樹分類器帶來的 information gain 就是非常大的。而通常來計算 information gain 會透過計算分出去 child node 的 impurity，常見有 Gini，entropy，classification error。而因為透過 scikit learn 的時候如果將填充顏色設為 filled=True,就會有以下的顏色。深橘與深藍的 boxe 為 Gini index=0 (也就是 class 被分類的非常純全都為 class 0 或 class 1).而為橘色還是藍色取決於式屬於 class 0 還是 class 1。而顏色的深度代表純度 ，如果為白色則代表 class 0 與 class 1 有同樣的數量。



分析 decision dree 的屬性後最左邊的分支也就是一路 leave 為 true 的屬性分別如下 ： 會離職

working_satisfaction <= 2.5 ➔ will leave

position <= 4.5 ➔ will leave

personal_life <= 1.5 ➔ will leave

Change_job <= 0.5 ➔ will leave

Working_participation <= 2.5 ➔ will leave

分析 decision dree 的屬性後最左邊的分支也就是一路 leave 為 false 的屬性不離職

working_satisfaction > 2.5 ➔ not leave

Personal_life > 1.5 ➔ not leave

Salary > 62600 ➔ not leave

Working_time_hour > 13.5 ➔ not leave

Frequency_of_attend_club >= 3.4 ➔ not leave

透過這些屬性產生 real dataset

```
# working_satisfaction <= 2.5 -> will leave
if(attribute['working_satisfaction'] <= 2.5):
  probOfLeave += 0.2
# position <= 4.5 -> will leave
if(attribute['position'] <= 4.5):
  probOfLeave += 0.2
# personal_life <= 1.5 -> will leave
if(attribute['personal_life'] <= 1.5):
  probOfLeave += 0.2
#Change_job <= 0.5 -> will leave
if(attribute['change_job'] <= 0.5):
  probOfLeave += 0.2
# Working_participation <= 2.5 -> will leave
if(attribute['working_participation'] <= 2.5):
  probOfLeave += 0.2

# working_satisfaction > 2.5 -> not leave
if(attribute['working_satisfaction'] > 2.5):
  probOfLeave -= 0.2
# Personal_life > 1.5 -> not leave
if(attribute['personal_life'] > 1.5):
  probOfLeave -= 0.2
#Salary > 62600 -> not leave
if(attribute['salary'] > 62600):
  probOfLeave -= 0.2
#Working_time_hour > 13.5 -> not leave
if(attribute['working_time(hour)'] > 13.5):
  probOfLeave -= 0.2
#Frequency_of_attend_club >= 3.4 -> not leave
if(attribute['frequency_of_attend_club'] >= 3.4):
  probOfLeave -= 0.2
```

4.1.A – fake data

Train accuracy : 0.8482857142857143

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5745	64
	TRUE	998	193

	precision	recall	f1-score	support
FALSE	0.85	0.99	0.92	5809
TRUE	0.75	0.16	0.27	1191
accuracy			0.85	7000
macro avg	0.8	0.58	0.59	7000
weighted avg	0.83	0.85	0.81	7000

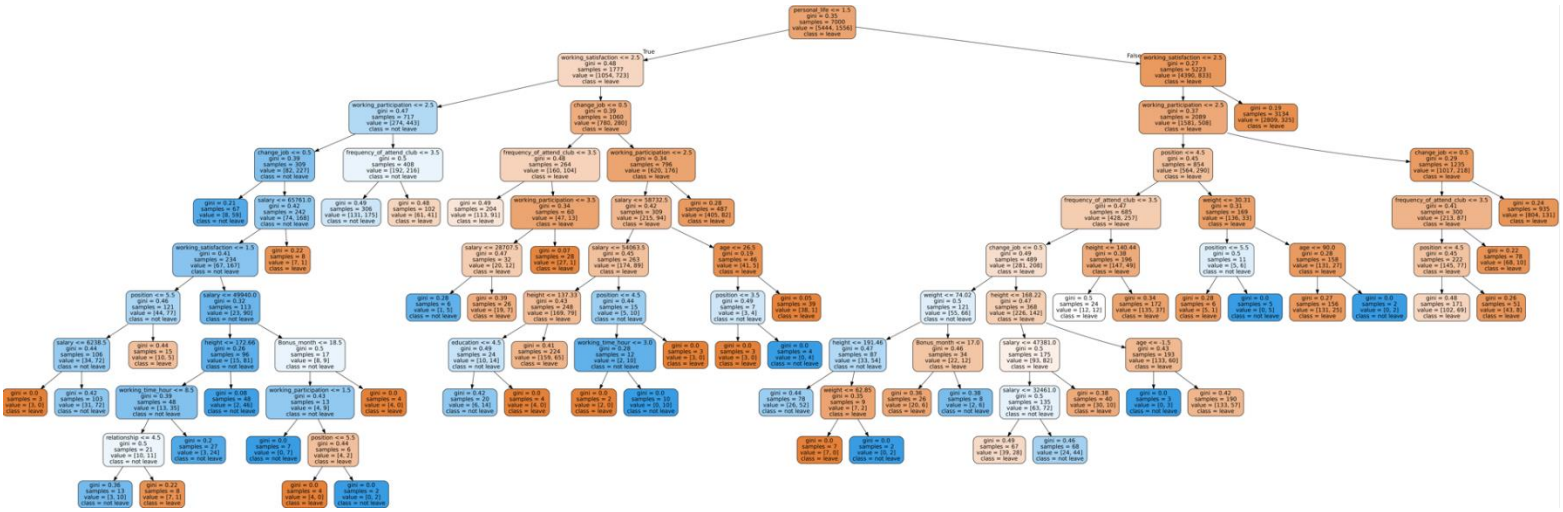
Test Accuracy : 0.822

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2430	64
	TRUE	470	36

	precision	recall	f1-score	support
FALSE	0.84	0.97	0.9	2494
TRUE	0.36	0.07	0.12	506
accuracy			0.82	3000
macro avg	0.6	0.52	0.51	3000
weighted avg	0.76	0.82	0.77	3000

4.1.B – real data

(可以看到 Decision Tree 的藍色跟橘色純度比較明顯區分) 可見附錄 P17



Train Accuracy : 0.8577142857142858

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5741	125
	TRUE	871	263

	precision	recall	f1-score	support
FALSE	0.87	0.98	0.92	5866
TRUE	0.68	0.23	0.35	1134
accuracy			0.86	7000
macro avg	0.77	0.61	0.63	7000
weighted avg	0.84	0.86	0.83	7000

Test Accuracy : 0.8366666666666666

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2437	92
	TRUE	398	73

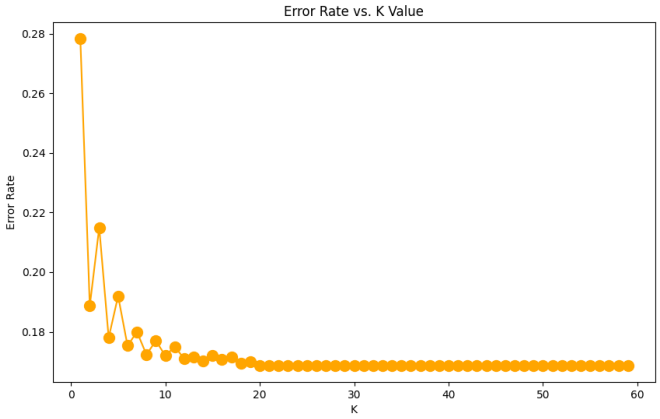
	precision	recall	f1-score	support
FALSE	0.86	0.96	0.91	2529
TRUE	0.44	0.15	0.23	471
accuracy			0.84	3000
macro avg	0.65	0.56	0.57	3000
weighted avg	0.79	0.84	0.8	3000

4.2 K-近鄰演算法 KNN

KNN 的實作方式，可以看到會計算各個樣本間與**目標點(target)**的距離為多少，此為歐幾里得距離。在來決定 K 值，選擇用 K 個距離最接近**目標點(target)**的 K 個樣本去決定目標。決定的方式就是 K 個樣本中最多或是平均後的類別就會成為**目標點(target)**的類別。也因此選 K 的值是非常重要的，如果 K 太大，就會加入許多沒有意義的資訊。而 K 值太小就會受到 noise 的影響。在實做 KNN 的時候我先將找出 error 最小的 K，再用該 K 值去訓練模型已得到最後的準確率。我覺得對於 10000 筆資料在 K<=60 的情況都不太會有 K 太大加入太多沒有意義的資訊，但可以看到 K 很小的時候錯誤率很大，因為決策的結果可能因為鄰近的幾個點個別分到的權重很大，造成誤判。

smallest error rate : 0.1686

K = 20



4.2.A – fake data

Train accuracy : 0.829857142857142

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5809	0
	TRUE	1191	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0	0	0	1191
accuracy			0.83	7000
macro avg	0.41	0.5	0.45	7000
weighted avg	0.69	0.83	0.75	7000

Test accuracy : 0.831333333333333

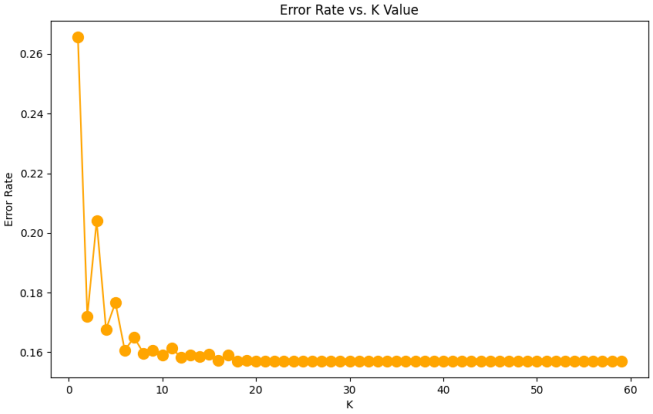
Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	506	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	0	0	0	506
accuracy			0.83	3000
macro avg	0.42	0.5	0.45	3000
weighted avg	0.69	0.83	0.75	3000

4.2.B – real data

smallest error rate : 0.157

K : 18



Train accuracy : 0.837857142857142

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5865	1
	TRUE	1134	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	0	0	0	1134
accuracy			0.84	7000
macro avg	0.42	0.5	0.46	7000
weighted avg	0.7	0.84	0.76	7000

Test accuracy : 0.843

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2529	0
	TRUE	471	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0	0	0	471
accuracy			0.84	3000
macro avg	0.42	0.5	0.46	3000
weighted avg	0.71	0.84	0.77	3000

4.3 Logistic Regression

4.3.A - fake data

Train accuracy : 0.829857142857142

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5809	0
	TRUE	1190	1

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0	0	0	1191
accuracy			0.83	7000
macro avg	0.91	0.5	0.45	7000
weighted avg	0.86	0.83	0.75	7000

Test accuracy : 0.831333333333333

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	506	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	0	0	0	506
accuracy			0.83	3000
macro avg	0.42	0.5	0.45	3000
weighted avg	0.69	0.83	0.75	3000

4.3.B – real data

Train accuracy : 0.838

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5866	0
	TRUE	1134	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	0	0	0	1134
accuracy			0.84	7000
macro avg	0.42	0.5	0.46	7000
weighted avg	0.7	0.84	0.76	7000

Test accuracy: 0.843

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2529	0
	TRUE	471	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0	0	0	471
accuracy			0.84	3000
macro avg	0.42	0.5	0.46	3000
weighted avg	0.71	0.84	0.77	3000

4.4 Naive Bayes

我查詢網路過後覺得用一句話概括貝式分類器很好”透過貝氏定理計算樣本在不同類別條件下的條件機率，並取得條件機率最大者作為預測類別。”

4.4.A fake data

Train accuracy : 0.830714285714285

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5804	5
	TRUE	1180	11

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0.69	0.01	0.02	1191
accuracy			0.83	7000
macro avg	0.76	0.5	0.46	7000
weighted avg	0.81	0.83	0.76	7000

Test accuracy : 0.832

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	504	2

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	1	0	0.01	506
accuracy			0.83	3000
macro avg	0.92	0.5	0.46	3000
weighted avg	0.86	0.83	0.76	3000

4.4.B real data

Train accuracy : 0.838428571428571

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5861	5
	TRUE	1126	8

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	0.62	0.01	0.01	1134
accuracy			0.84	7000
macro avg	0.73	0.5	0.46	7000
weighted avg	0.8	0.84	0.77	7000

Test accuracy : 0.842666666666666

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2526	3
	TRUE	469	2

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0.4	0	0.01	471
accuracy			0.84	3000
macro avg	0.62	0.5	0.46	3000
weighted avg	0.77	0.84	0.77	3000

4.5 Random Forest

random forest 是結合許多 decision tree，透過 ensemble method 實做。因為只有一個 training dataset，為了要產生許多不同的 decision tree，就需要產生不同的子 dataset 才能就夠不一樣的 decision tree，如此 ensemble method 才會具有效過。[Ensemble Methods]

Ensemble Method 分為 Step1-Bagging, Step2: Boosting, Step3: Stacking

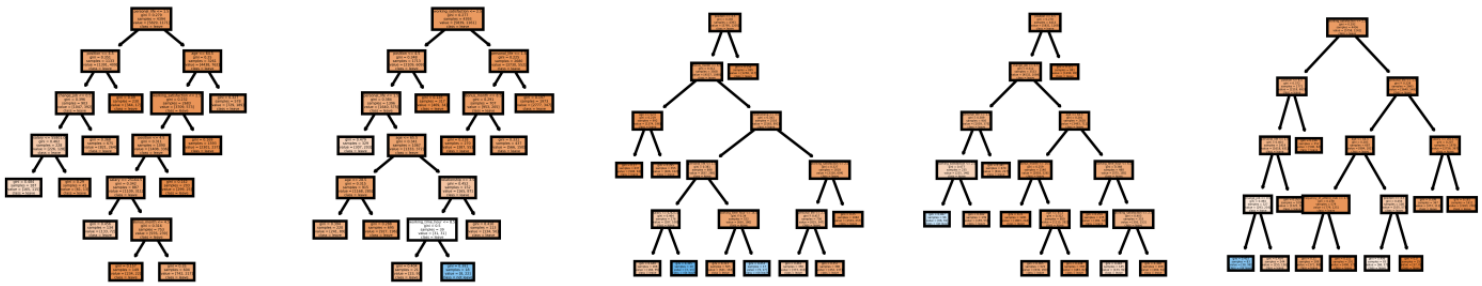
Step1-Bagging： 使用替換過程將 dataset 分成 n 個 dataset 個別訓練成單一的 decision tree。再聚合，如此會有 decision tree 使用到同樣的數據，不過仍保留每個 decision tree 獨一無二的特性，而最後再用投票的方式決定最後 predict

的結果。

Step2-Boosing：針對錯誤的部分加強學習來提升 accuracy。如果有 decision tree 分類錯誤，那就把他的投票權種提高，來迫使該 decision tree 能夠學習，有點像 train neural network drop out 的概念：如果有神經元隊最終正確答案不貢獻，那就把貢獻多的神經元 drop out 掉，避免搭便車效應。

4.5.A – fake data(可以看到每棵樹的藍色跟橘色較不明顯區分)

Estimator: 0 Estimator: 1 Estimator: 2 Estimator: 3 Estimator: 4



Train accuracy : 0.830714285714285

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5809	0
	TRUE	1191	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0	0	0	1191
accuracy			0.83	7000
macro avg	0.41	0.5	0.45	7000
weighted avg	0.69	0.83	0.75	7000

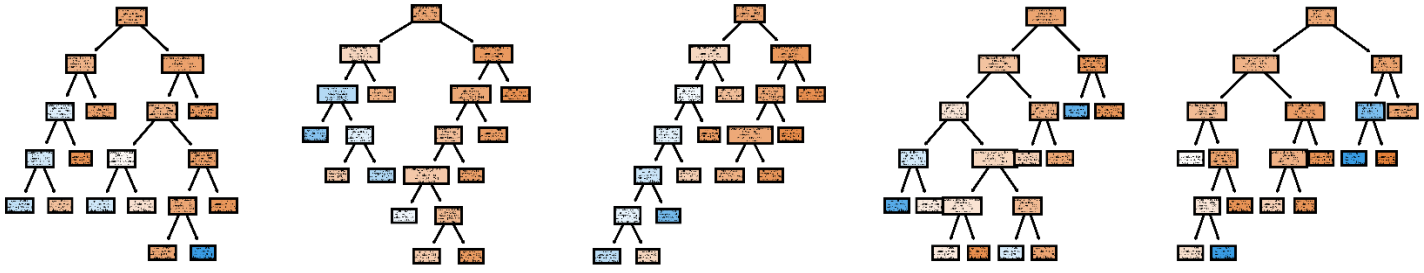
Test accuracy : 0.831333333333333

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	506	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	0	0	0	506
accuracy			0.83	3000
macro avg	0.42	0.5	0.45	3000
weighted avg	0.69	0.83	0.75	3000

4.5.B – real data (可以看到每棵樹的藍色跟橘色比較明顯區分)

Estimator: 0 Estimator: 1 Estimator: 2 Estimator: 3 Estimator: 4



Train accuracy : 0.838

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5866	0
	TRUE	1134	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	0	0	0	1134
accuracy			0.84	7000
macro avg	0.42	0.5	0.46	7000
weighted avg	0.7	0.84	0.76	7000

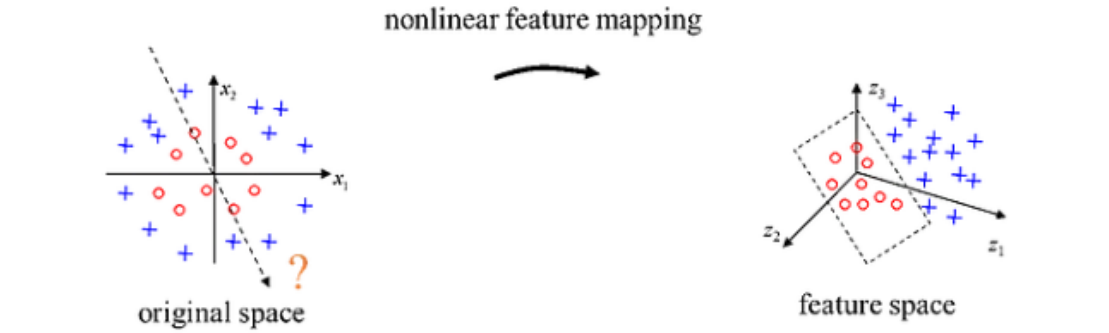
Test accuracy : 0.843

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2529	0
	TRUE	471	0

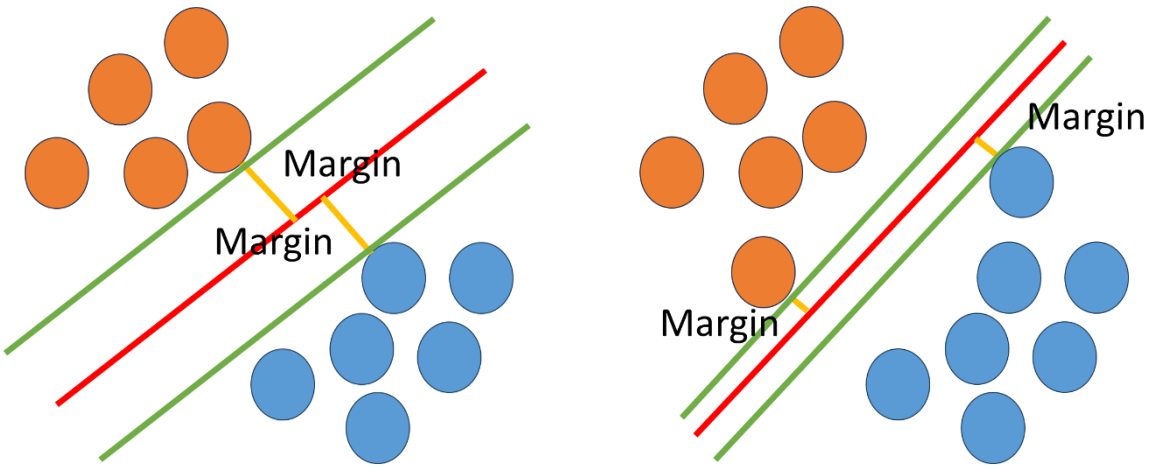
	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0	0	0	471
accuracy			0.84	3000
macro avg	0.42	0.5	0.46	3000
weighted avg	0.71	0.84	0.77	3000

4.6 SVM

SVM 的實做與 Kernel function 核函數的設計有密切關係。當兩個類別的資料在原始的空間無法透過 SVM 一條線直接切分隔出來，那就要透過非線性的投影將資料在更高維度的空間去區分開來。(如圖，取自 Link)而這個更跟維度的則稱作 Hilbert space(H)，但因為很難去設計好的非線性投影公式，因此需要 kernel 函數來幫忙。我查詢網路上的說法 kernel 函數的挑選方式與參數都用 training data 全部的組合都跑一次(grid search)，來暴力尋找最好的一組解。



而當投影到更高維度後要如何找到一條線或是平面去區分兩個類別，此時要透過 maximize margin 的方式，找到一個決策分界他能夠與兩類型的邊界最大即可，當資料新增則不斷的優化。



4.6.A – fake data

Train Accuracy : 0.829857142857142

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5809	0
	TRUE	1191	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0	0	0	1191
accuracy			0.83	7000
macro avg	0.41	0.5	0.45	7000
weighted avg	0.69	0.83	0.75	7000

Test accuracy : 0.831333333333333

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	506	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	0	0	0	506
accuracy			0.83	3000
macro avg	0.42	0.5	0.45	3000
weighted avg	0.69	0.83	0.75	3000

4.6.B – real data

Train accuracy : 0.838

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5866	0
	TRUE	1134	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	0	0	0	1134
accuracy			0.84	7000
macro avg	0.42	0.5	0.46	7000
weighted avg	0.7	0.84	0.76	7000

Test accuracy : 0.843

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2529	0
	TRUE	471	0

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0	0	0	471
accuracy			0.84	3000
macro avg	0.42	0.5	0.46	3000
weighted avg	0.71	0.84	0.77	3000

4.7 MLP

為 neural network 的方法。之所以會被叫做 Multi-Layer Perceptron 多層感知器，是因為神經網路很多層。而每一層由多個神經元組成。神經元會接受上一層許多 input 權重加總然後傳給 activate function。當值超過 activate function 的 threshold 則 output 為 1，反之 output 為 0。而這種複雜的網路讓複雜的屬性都能夠被學起來。

4.7.A Fake dataset

Train Accuracy : 0.829857142857142

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5809	0
	TRUE	1191	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	5809
TRUE	0	0	0	1191
accuracy			0.83	7000
macro avg	0.41	0.5	0.45	7000
weighted avg	0.69	0.83	0.75	7000

Test Accuracy : 0.831333333333333

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2494	0
	TRUE	506	0

	precision	recall	f1-score	support
FALSE	0.83	1	0.91	2494
TRUE	0	0	0	506
accuracy			0.83	3000
macro avg	0.42	0.5	0.45	3000
weighted avg	0.69	0.83	0.75	3000

4.7. Real Dataset

Train Accuracy : 0.838285714285714

Train Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	5866	0
	TRUE	1132	2

	precision	recall	f1-score	support
FALSE	0.84	1	0.91	5866
TRUE	1	0	0	1134
accuracy			0.84	7000
macro avg	0.92	0.5	0.46	7000
weighted avg	0.86	0.84	0.76	7000

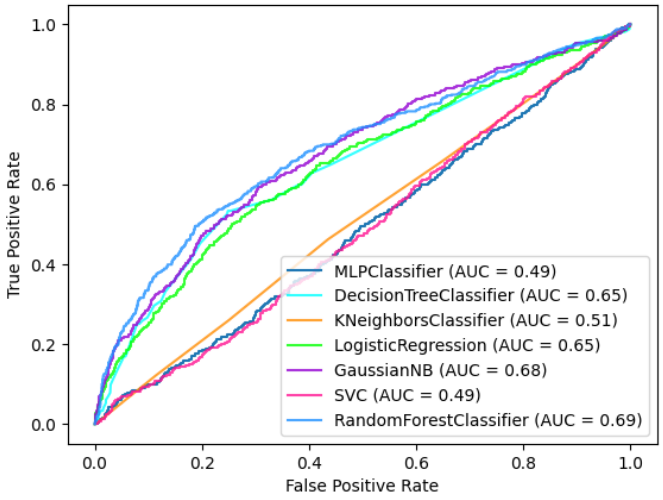
Test Accuracy : 0.842333333333333

Test Confusion Matrix:			
	Predict Label		
True Label		FALSE	TRUE
	FALSE	2527	2
	TRUE	471	0

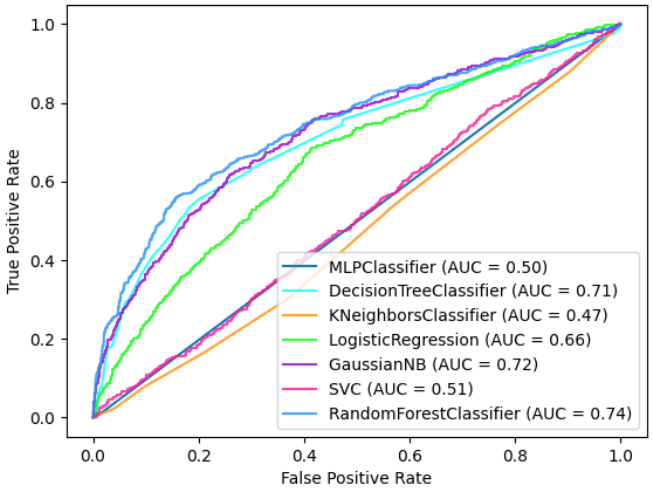
	precision	recall	f1-score	support
FALSE	0.84	1	0.91	2529
TRUE	0	0	0	471
accuracy			0.84	3000
macro avg	0.42	0.5	0.46	3000
weighted avg	0.71	0.84	0.77	3000

4.8 ROC curve

4.8.A Fake dataset



4.8.B Real Dataset



5. 結論 Conclusion

Compare the rules in the decision tree from Step 2 and the rules you used to generate your ‘right’ data :

由於我的 real data 產生的規則，也就是 decision tree 的最左邊分支會離職與最右邊分支不會離職的屬性，而這些屬性跟原先產生 fake data 其實沒有差很多，唯一的差別只在於最左邊分支與最右邊分支剛好各五個屬性，而這 5 個屬性我以相同的權重(也就是對於最後是否離職的機率都以增加與下降 0.2 的機率去計算)。那這樣生出來的 fake data 與 real data 其實很接近。但從 ROC curve 觀察可以發現 Fake Dataset 比 Real Dataset 來的更好，我分析原因後發現我生成 Real Dataset 的方式騎士也沒有那麼 Real，反倒是生成 Fake Dataset，會自己依照一些嘗試去調整每個屬性的權重(也就是對於最後是否離職的機率都以增加與下降不同的機率去計算)。而在生成 Real Dataset 時卻沒有做到這一點。這是這次作業還可以再改進的地方。但也由於 Real Dataset 的規則比較簡單所以產生出來的 AUC 都比較高，同時 Decision Tree 與 Random Forest 的樹藍橘色也都比較明顯可以被區分。

KNN 的分析 ：

其實觀察每個分類器他們在準確率上面都沒有特別明顯的差異。但畫成 ROC curve 卻可以有明顯的差異。比較不同的模型，像是 KNN 在資料不平衡的情況下表現的非常差，常會有預測錯誤的情況，因為當會離職的相對於不會離職的人少的多時，當判斷每個點會不會離職，都會被旁邊佔大多數不會離職的點所左右判斷。因此 KNN 在 ROC 的表現上不好。

Logistic Regression 與 SVM 的分析 ：

對於兩個比較像的 Logistic Regression 與 SVM 的比較，Logistic Regression 因為有一個 sigmoid function 使其決策的邊界為一個機率分布，不唯一，同時在訓練的過程中，樣本一旦經由決策的邊界被分成兩類後，Logistic Regression 則停止繼續優畫，而 SVM 由於決策的邊界唯一，因此會不斷的優化決策的邊界。而對於 SVM 為什麼會是所有的表現上倒數差的，由於 SVM 存在一些重要的參數，如核函数参数等，这些参数的选择对模型的性能影響大。因此選擇參數和核函數是 SVM 的關鍵問題題，這通常需要不同參數組合進行 cross validation，增加調參的複雜度。

Naïve Bayes 的分析 ：

由於我設計 dataset 的時候並沒有讓特定屬性去影響其他屬性的分布，每個屬性之間相互獨立。也因為我的資料集的設計都源自於機率分布，而 Naïve Bayes 的算法會對整個資料量進行區樣計算為每個類別在某個屬性下的條件機率為多少，而計算出最終多個屬性下為該類別的機率，也因此在 ROC curve 上會表現得比其他好。

Random Forest 與 Decision Tree 的分析 ：

隨機森林中每棵樹會用到哪些訓練資料及特徵都是由隨機決定，訓練或是預測的階段每一棵樹都能平行化的運行，採用多個決策樹的投票機制來改善決策樹，這種 ensemble 的方式讓集結 decision tree 的力量，避免過度擬合與偏差的問題。同時因為任何資料型態都有屬性，而 Decision Tree 是一種非參數化的模型，因此甚麼數據的分布以及特性他都很容易的整合。

MLP 的分析 ：

由於現在的機器學習最常用的方法為 MLP，MLP 的 ROC curve 卻表現不理想，試著找尋原因，我認為一方面是因為不會調整參數，另一方面是因為 MLP 比起其他分類器在結構化資料的表現上沒有那麼出色，MLP 可能更擅長聲音與影像的非結構化資料型態。

6. 參考 ：

1. 資料視覺化之 Decision tree (決策樹)範例與 Machine Learning (機器學習)概念簡單教學(入門)
2. Python 學習筆記#14：機器學習之 KNN 實作篇
3. Graphviz download
4. A Step-by-Step Guide to Building Accurate Predictive Decision Tree Model
5. 決策樹
6. 隨機森林
7. KNN
8. 羅吉斯回歸
9. SVM
10. Kernel 函數 11. Kernel 函數直覺理解
12. SVM 優缺點