資料探勘 作業二

資訊所 P76074509 辜玉雯

- ## 目的：

此次作業是設計具有 k 個 features 且大小為 M 的 dataset，並且設計幾個「absolutely right」的規則來分類 dataset，再利用不同的分類器作訓練並分類出每筆資料的類別。最後比較這些分類器的效果，以及他們採用的分類規則與自己設計的規則是否吻合。

- ## Dataset：

這裡設計的 dataset 是想從資料中預測某個人目前是否有購屋意願，共有 10 個 feature，共計 1000 筆資料。

Feature 包含年收入、性別、年齡、婚姻狀況、小孩數、居住區域、附近房價、每月償還貸款金額、貸款是否超過月薪四成以及購屋意願。

值得注意的一點是，dataset1 不具有貸款是否超過月薪四成這項 feature，而 dataset2 包含了該 feature，因為發現這項需要參考多個 feature 而產生的複合欄位，是否有特別標註會對分類效果有影響，所以特別分開兩種狀況討論。

- ## Absolutely Right Rules：

1. 假設年收入 100 萬以上 AND 已婚 AND 居住中部 AND 附近房價 20-30 萬
2. 假設已婚 AND 有小孩 AND 貸款每月償還不超過月薪四成
3. 假設女性 AND40-49 歲 AND 未婚 AND 貸款每月償還不超過月薪四成

共計 3 條規則，若符合任一規則，則判斷此人目前有購屋意願。

- 環境：

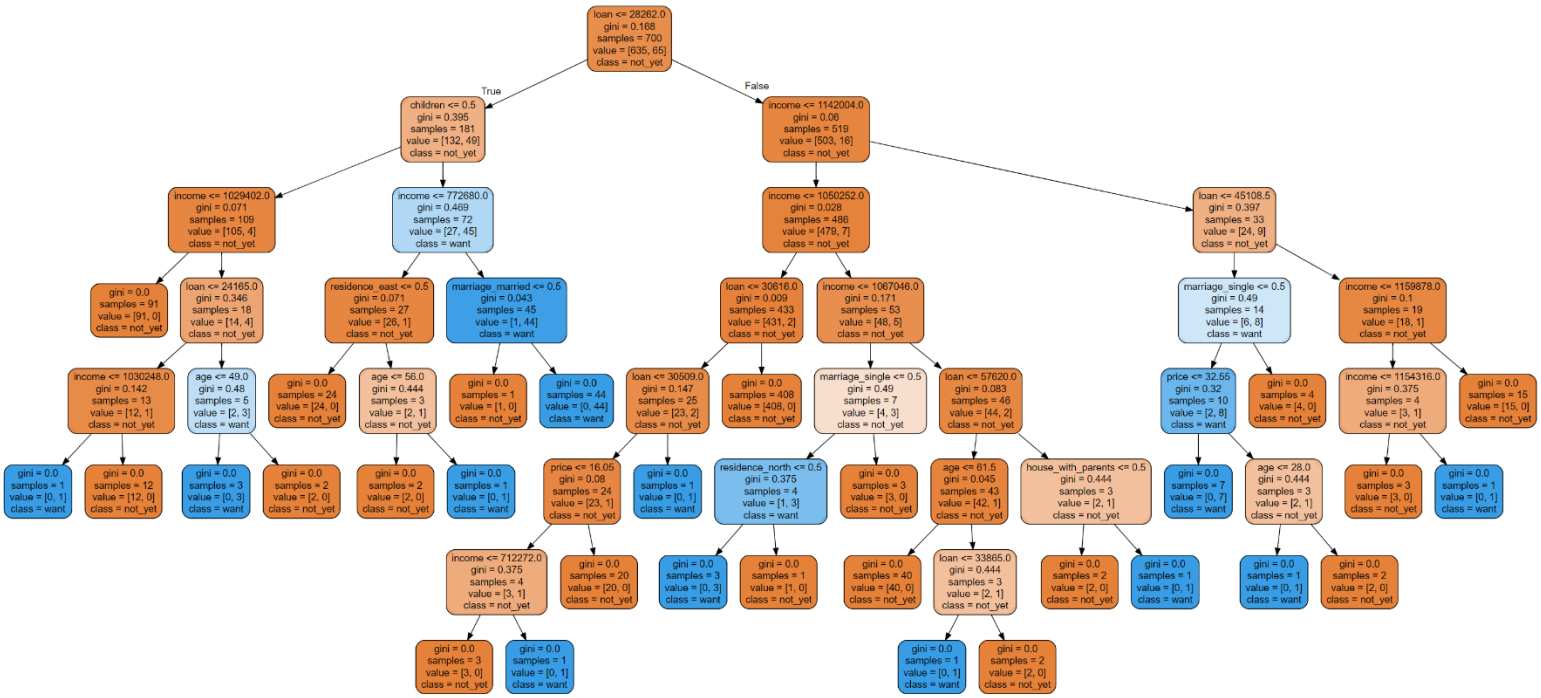
實作語言	Python
CPU	intel i5-6400 2.7GHz
Memory	16G
OS	Windows 10 x64

- 實驗：

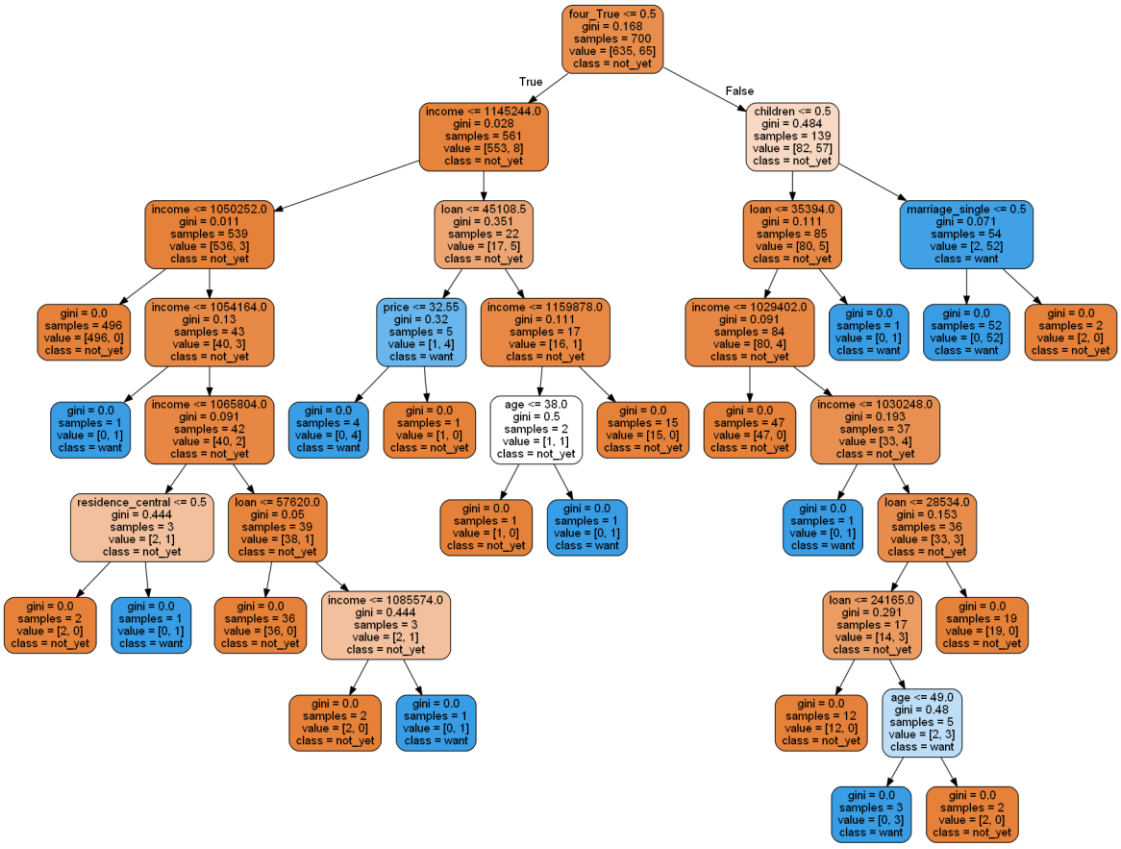
實驗一、

比較利用決策樹模型對 dataset1 與 dataset2 的分類效果，以及透過視覺化 decision tree 來觀察分類的規則。

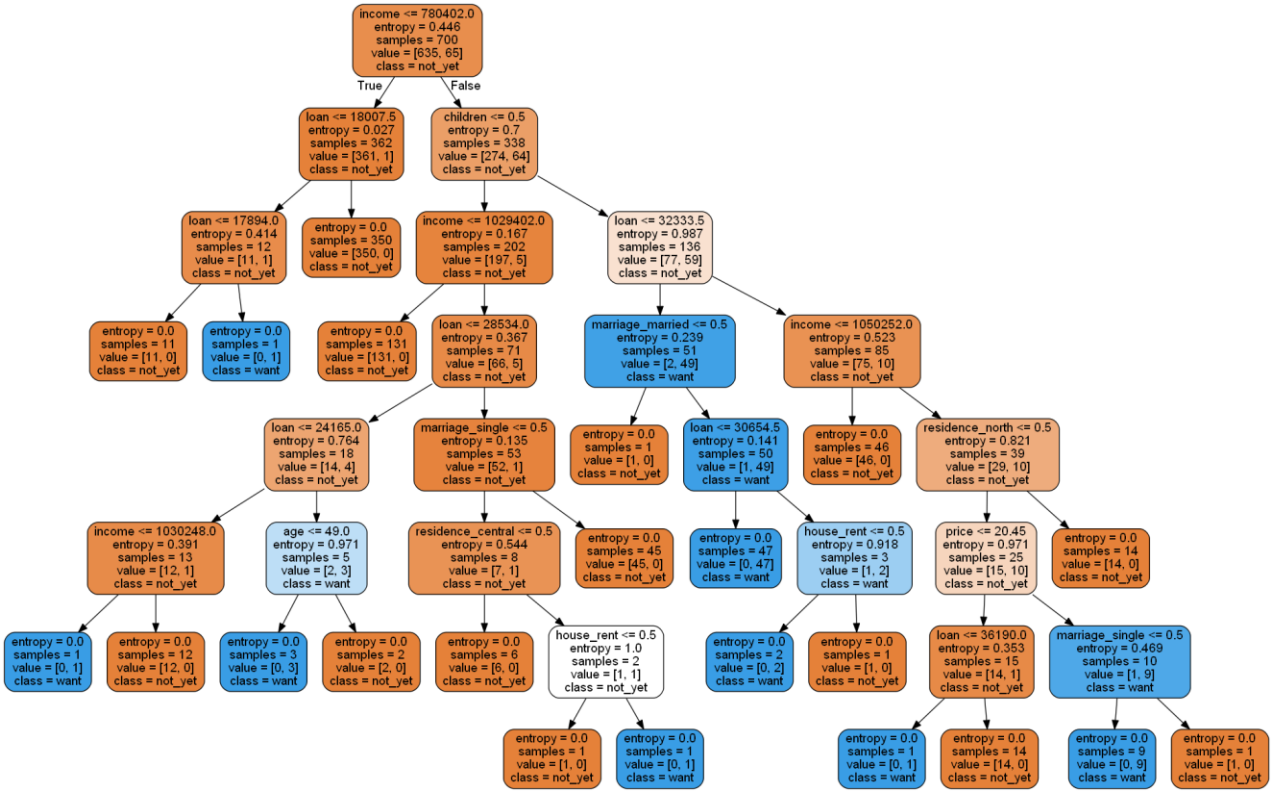
Criterion = gini	
Dataset1	precision recall f1-score support F 0.95 0.98 0.96 265 T 0.83 0.57 0.68 35
Dataset2	precision recall f1-score support F 0.97 0.99 0.98 265 T 0.93 0.80 0.86 35
Criterion = entropy	
Dataset1	precision recall f1-score support F 0.95 0.99 0.97 265 T 0.88 0.60 0.71 35
Dataset2	precision recall f1-score support F 0.98 1.00 0.99 265 T 0.97 0.86 0.91 35



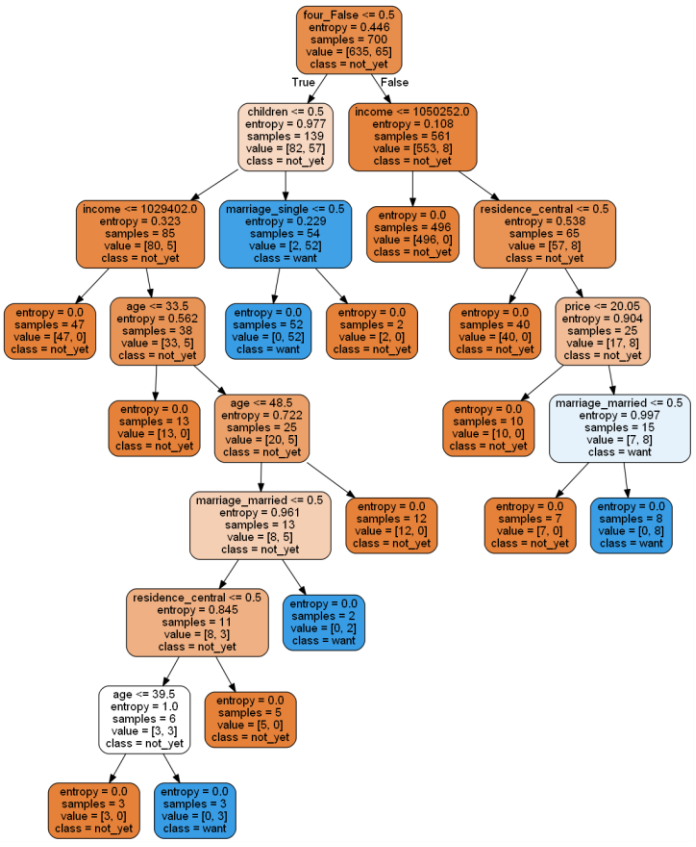
〈圖一〉Decision tree 視覺化結果（criterion=gini，dataset1）。



〈圖二〉Decision tree 視覺化結果（criterion=gini，dataset2）。



〈圖三〉Decision tree 視覺化結果（criterion=entropy，dataset1）。



〈圖四〉Decision tree 視覺化結果（criterion=entropy，dataset2）。

實驗結果：

在 dataset1 沒有標註貸款是否超過月薪四成這項，decision tree 的做法是統整出有購屋意願的人，直接針對他們貸款的範圍，然後去判斷是否在這個範圍內進而做其他 feature 的分枝，這樣的做法效果自然比較差，因為他無法掌握到複合資料的值與意義，僅僅利用訓練資料當中，有購屋意願者的貸款金額來做判斷，並沒有真正考慮到”四成”這個條件。

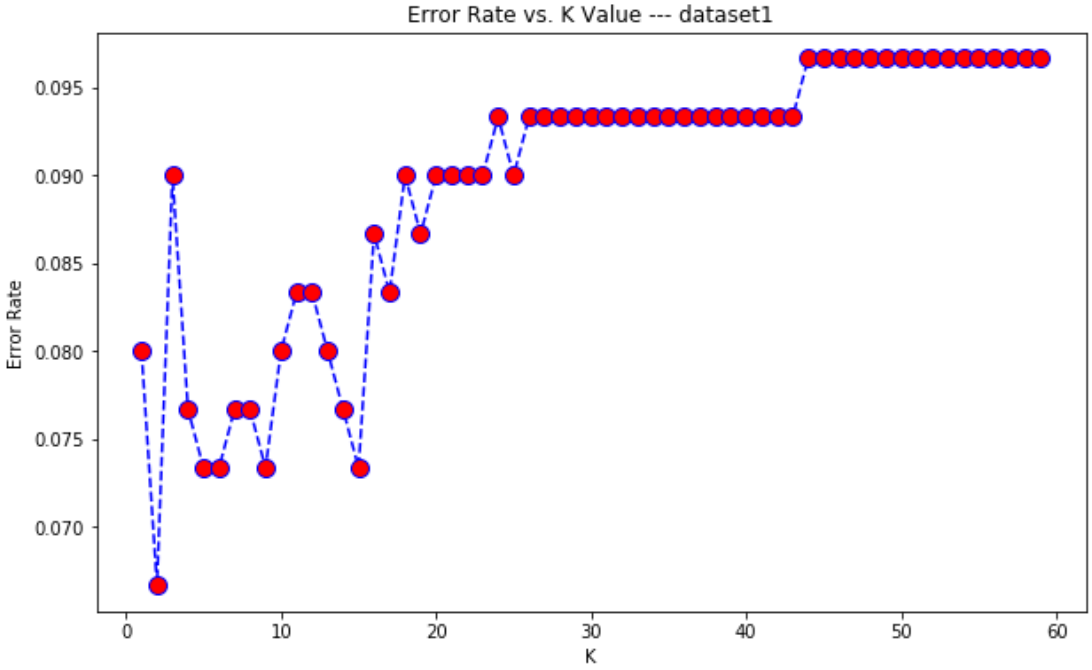
而 dataset2 因為有貸款是否超過月薪四成這一欄位，因此在做 decision tree 時，分枝的效果增進很多，因為他不是利用無真實意義的貸款金額去做處理，而是直接採用是否大於”四成”來做判斷。

但也因為在 decision tree 當中沒有特別設定剪枝的參數，所以視覺化 decision tree 可以發現，他們分枝都分的很細，如果套用到其他 testing data 可能會有 overfitting 的狀況，這是未來可以再修改的地方。

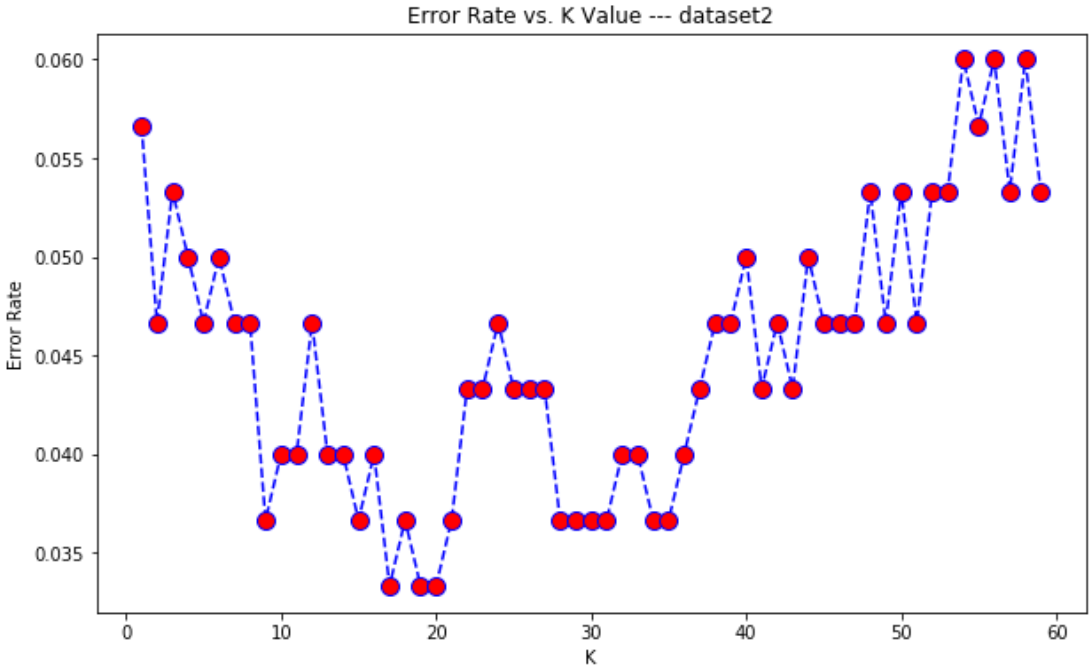
實驗二、

比較利用 KNN 模型對 dataset1 與 dataset2 的分類效果，在此採取各自最佳的 k 值找出最佳狀況來進行比較，dataset1 採用 k 值為 2，dataset2 採用 k 值為 20。以及比較在不同 k 值下的 error rate。

採取各自最佳的 k 值來進行比較				
Dataset1		precision	recall	f1-score	support
	F	0.94	0.99	0.96	271
	T	0.85	0.38	0.52	29
Dataset2		precision	recall	f1-score	support
	F	0.97	0.99	0.98	271
	T	0.88	0.76	0.81	29



〈圖五〉Dataset1 在不同 k 值時，各自的 error rate。判斷最佳的 k 值為 2，因為它的 error rate 最低。



〈圖六〉Dataset2 在不同 k 值時，各自的 error rate。判斷最佳的 k 值為 20，因為它的 error rate 最低。

實驗結果：

可以觀察到 dataset2 的表現仍然優於 dataset1，但因為一開始隨機設

定 k 值的狀態下，跑出來的結果不是很理想，所以後來多跑了幾個 k 值去找出最佳解，由此可以發現 KNN 比較麻煩的地方就是需要自行設定 K 值，此外，在效果上也沒有 Decision tree 來的好。

實驗三、

比較利用 SVM 模型對 dataset1 與 dataset2 的分類效果。以及在線性與非線性下的準確率。以及在非線性的狀況下，調整 C 與 gamma 值觀察它們對準確率的影響，在 svm 函式中可以設定 C (懲罰係數)，決定對誤差的寬容度，C 越高代表越不能容許誤差，但就有可能 overfitting；而 gamma 則是決定 data 映射到新的特徵空間的分布，gamma 越大代表支持向量越少，反之。然而這兩個參數都會影響訓練與預測的速度。

<table>
<tr><td colspan="2">Kernel function=linear</td></tr>
<tr><td>Dataset1</td><td>

	precision	recall	f1-score	support
F	0.96	0.98	0.97	271
T	0.79	0.66	0.72	29

</td></tr>
<tr><td>Dataset2</td><td>

	precision	recall	f1-score	support
F	0.97	0.99	0.98	271
T	0.87	0.69	0.77	29

</td></tr>
<tr><td colspan="2">Kernel function=RBF</td></tr>
<tr><td>Dataset1</td><td>

	precision	recall	f1-score	support
F	0.96	0.98	0.97	271
T	0.79	0.66	0.72	29

</td></tr>
<tr><td>Dataset2</td><td>

	precision	recall	f1-score	support
F	0.98	0.99	0.99	271
T	0.89	0.83	0.86	29

</td></tr>
</table>

```
Best parameters set found on development set:

{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
```

```
0.945 (+/-0.075) for {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
0.874 (+/-0.115) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
0.449 (+/-0.003) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.449 (+/-0.003) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.920 (+/-0.100) for {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
0.970 (+/-0.044) for {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
0.865 (+/-0.080) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.449 (+/-0.003) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.903 (+/-0.069) for {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
0.925 (+/-0.051) for {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
0.908 (+/-0.126) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.865 (+/-0.080) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.903 (+/-0.069) for {'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
0.900 (+/-0.058) for {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
0.942 (+/-0.059) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.908 (+/-0.126) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.924 (+/-0.101) for {'C': 1, 'kernel': 'linear'}
0.922 (+/-0.097) for {'C': 10, 'kernel': 'linear'}
0.923 (+/-0.099) for {'C': 100, 'kernel': 'linear'}
```

〈圖七〉Dataset2 在不同 C 值與 gamma 值組合時，各自的分數。判斷最佳的組合為 C=10、gamma=0.01，因為此時分數最高。

實驗結果：

由表格可以看出在 svm 中利用 RBF 這個 kernel function 比線性的做法效果更好，此外，經過測試，我們從 [0.1,1,10,100,1000]這些選項中挑出最佳的 C；從[1,0.1,0.01,0.001,0.0001]當中挑出最佳的 gamma 值，找出兩者組合最佳的狀況。經過測試採用了 C=10、gamma=0.01 這個組合來進行訓練以及預測。

實驗四、

比較利用 Random Forest 模型對 dataset1 與 dataset2 的分類效果。位觀察兩個 dataset 各自效果，因此將 tree 個數都設為 100。

		precision	recall	f1-score	support
Dataset1	F	0.95	1.00	0.97	271
	T	1.00	0.52	0.68	29
Dataset2	F	0.98	1.00	0.99	271
	T	1.00	0.79	0.88	29

實驗結果：

由表格可以看出在 Random Forest 中同樣是 dataset2 的效果較好，而且也不會像 Decision Tree 有 overfitting 的問題。

實驗五、

比較各種分類器對 dataset2 的分類效果。

分類器名稱	執行結果
DT_{gini}	precision recall f1-score support F 0.99 0.99 0.99 271 T 0.86 0.86 0.86 29
$DT_{entropy}$	precision recall f1-score support F 0.99 1.00 0.99 271 T 1.00 0.90 0.95 29
KNN	precision recall f1-score support F 0.94 0.99 0.96 271 T 0.85 0.38 0.52 29
SVM_{linear}	precision recall f1-score support F 0.97 0.99 0.98 271 T 0.87 0.69 0.77 29
SVM_{RBF}	precision recall f1-score support F 0.98 0.99 0.99 271 T 0.89 0.83 0.86 29
Random Forest	precision recall f1-score support F 0.98 1.00 0.99 271 T 1.00 0.79 0.88 29

實驗結果：

從 precision 來看所以分類器不相上下，但從 f1-score 就可以看出他

們的差異。

$$DT_{entropy} > \text{RF} > SVM_{RBF} > DT_{gini} > SVM_{linear} > \text{KNN}$$

Decision tree 的效果在這個例子看起來不錯，但容易有 overfitting 的問題。

SVM 的效果比 Decision tree 略遜，但相對之下 overfitting 的狀況較少，所以整體來說 SVM 是更加適合的分類器。但是缺點是執行時間比 decision tree 長，因為它的時間複雜度較高，所以一旦資料量變大，可能就需要找尋其他更適合的分類器。

Random Forest 以多個決策樹為基礎去修正，也可以看出它的效果甚至會比 SVM 來得好，而且他也改善了 Decision Tree overfitting 的缺點，所以也是一個合適的分類器。

而 KNN 則是一個最基礎的 baseline 方法，僅透過鄰近點來分類，在只有一條規則的狀況下，可能會有不錯的效果，但因為我設定了三條規則，所以透過鄰近點區分並不是一個最完善的做法。

- 討論：

  在實作這份作業的過程，我遇到幾個問題值得提出來討論。

  - 是否有複合欄位的差別

    一開始在設計 absolutely right rules 時，只有包含年齡是否在某個區段、收入是否在某個區段，所以在使用分類器分類時，效果其實都非常好，但是這樣一來就無法比較他們的效果，而且從現實面來看，條件通常也不會如此單純。因此，我又加入貸款每月償還不超過月薪四成這項，這個條件需要比較貸款金額與年收入/12，需要有較多的四則運算，這樣就可以明顯看出分類器的執行效果。

    但又會產生另一個問題，就是有沒有將四則運算結果特地標出、特地設一個欄位，對分類器也很有影響，有了該欄位就可以直接利用那個 feature 進行分類，如果沒有就要整理有購屋意願的人他們在各個 feature 的內容大概位於什麼範圍，進而去分類，但效果自然會較差，

而且若用其他 testing data 來分類，錯誤率容易較高。

因此，在做資料探勘時，適當的前處理，是必要也確實可以增進分類效果的步驟。

◆ 是否有標準化的差別

在設計好 dataset 與 rules 之後，一開始只利用 decision tree 分類時並沒有發現異樣，直到利用 SVM 時，發現效果非常差，經過一番搜尋發現問題，因為 feature 包含了年收入，數值約從幾十萬到一百多萬，而類別資料經過轉換，數值都是個位數，因此在進行分類時，數值差異太大，對分類器來說非常影響效果。

而在將各欄位進行標準化過後，SVM 的效果便正確顯現出來。因此，我也瞭解除了需要將類別資料轉成數值資料，將各個 feature 進行標準化也是重要的前處理之一。

- 結論：

在這次的作業中不僅嘗試各種分類器，比較他們的效果並了解分類的運作方式，更進一步了解作資料探勘時，進行前處理的必要性，必須先了解資料的特性，找出與預測目標有關的 feature，並處理類別資料，因為大部分分類器不接受類別資料，爾後還要進行標準化，使得預測結果不會受到某一個數值較大的 feature 影響而有不準確的狀況。