決策樹 (Decision Tree)

陸裕豪

December 9, 2020

文件矩陣

- 資料科學/機器學習的演算法有很多。
  - 決策樹是其中一種演算法。
  - 其他演算法例如: 類神經網路、kNN、羅吉斯迴歸、支援向量機等...
- 決策樹常用於分類問題，根據規則以樹狀結構把資料分類到可能相應的類別。
- 所謂的分類問題，是利用演算法，找出/分析特徵 (輸入資料) 與類別 (答案) 的內在關係。

決策樹的例子

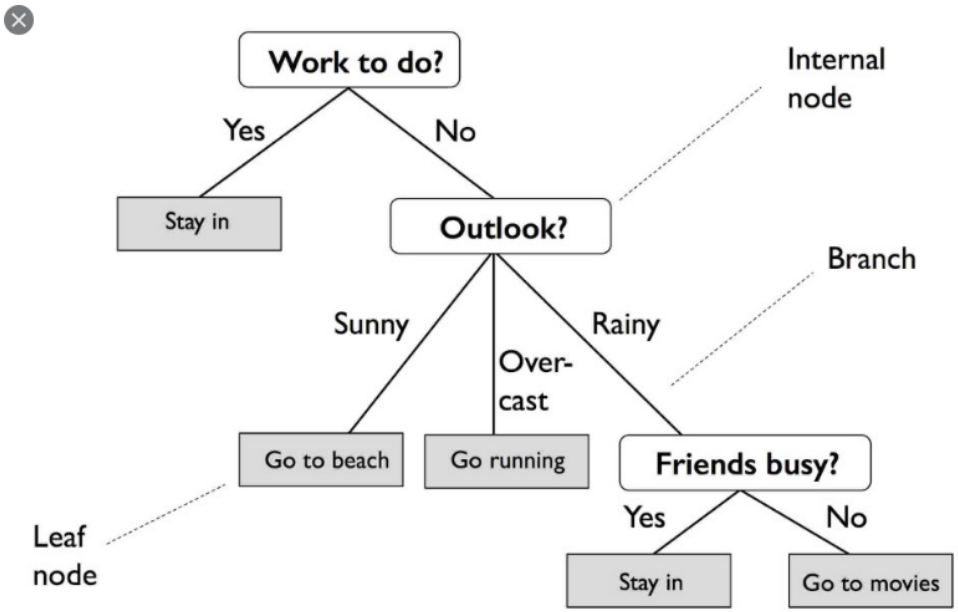


Figure: 決策樹的例子。

- 決策樹不只是給出預測結果 (留在家/出去)。
- 還會輸出一系列導致最終預測的中間決策。
- 我們可以對這些中間決策進行驗證與判斷。

[1]

[1] 來源:https://zhuanlan.zhihu.com/p/128533291

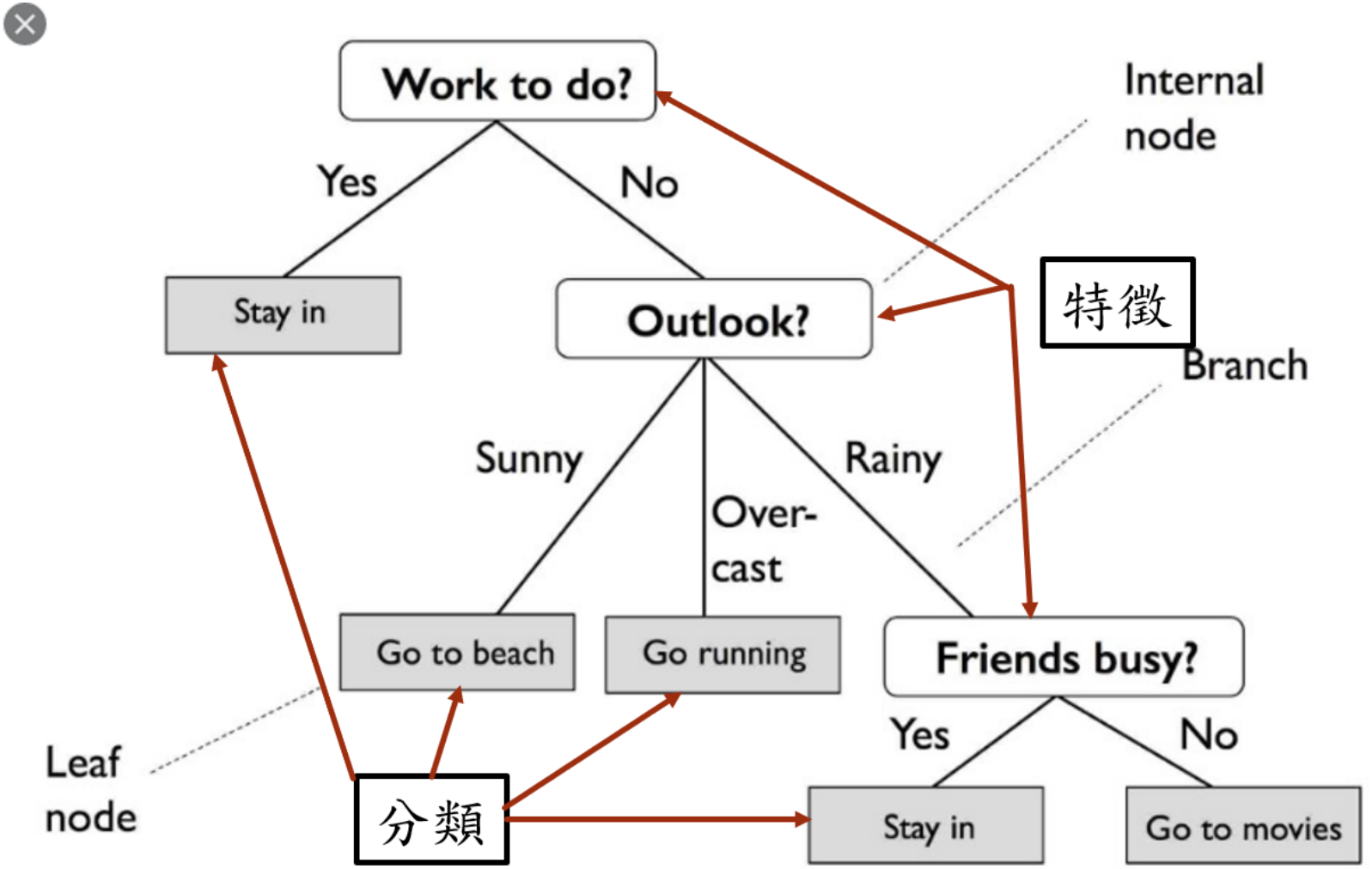
決策樹的優點

- 它最重要的優點在於其「可解釋性」
  - 隨著人工智慧應用在金融、會計、醫療、法律等領域，模型的可解釋性就顯得非常重要。
- 每個決策階段都相當的明確清楚。
- 前述的一些算法如類神經網路、支援向量機，就很難去演繹其結果背後的原因。

決策樹節點選擇原理

- 我們怎麼知道要分成這幾個節點?
- 哪一個節點適合作為啟始根部?
- 節點的判斷依據及數值的認定為何?
- 此時，就有各種決策樹演算法回答上述問題
  - e.g., ID3，C4.5，CART。
- 這些算法將輸入的特徵予以量化，自動的建構並決定決策樹的各個節點。

決策樹節點選擇原理


Work to do?
Internal node
Yes
No
Stay in
Outlook?
特徵
Branch
Sunny
Over-cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Leaf node
分類
Stay in
Go to movies

資訊增益 (Information Gain, IG)

- 使用 IG 將較高同質性的資料放置於相同的類別，以產生各個節點。此種演算法依賴所謂「Entropy（熵）」，其公式是：
- 哪一個節點適合作為啟始根部?
- 節點的判斷依據及數值的認定為何?
- 此時，就有各種決策樹演算法回答上述問題
- Entropy $= -p\log(p) - q\log(q)$
  - p：成功的機率（或 true 的機率）
  - q：失敗的機率（或 false 的機率）

資訊增益 (Information Gain, IG)

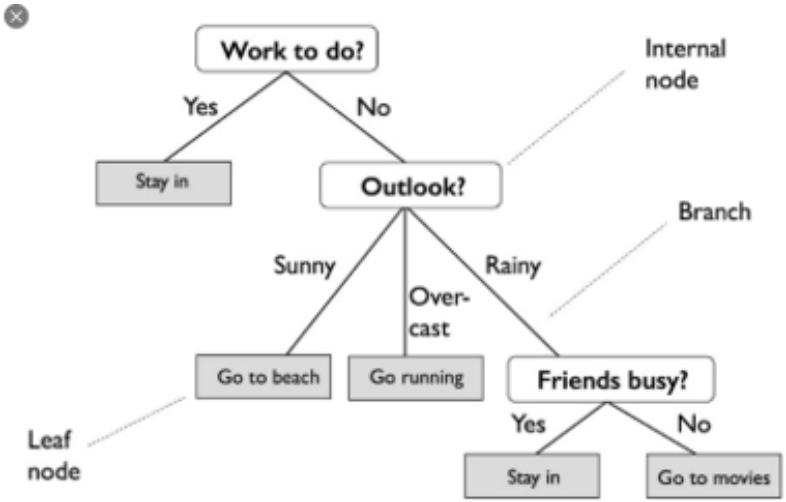
$$\text{資訊增益} IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

IG：獲得的資訊量,I：原本的資訊量,
$\sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$：經由分割後的資訊量

- 我們希望獲得的資訊量要最大。
- 因此經由分割後的資訊量要越小越好。

機器學習的輸入

- 我們需要輸入資料
  - 特徵 (工作是否完成、天氣、朋友是否有空)
  - 類別 (留在家、出去)
- 把資料分成訓練集 (e.g.,80%) 和測試集 (20%)
  - 訓練集用來訓練模型 (演算法) 參數
  - 測試集用來測試模型效能



混淆矩陣 (Confusion Matrix)

實際狀況:True/False
預測狀況:Positive/Negative

	實際 Yes	實際 No
預測 Yes	TP (True Positive)	FP (False Positive) Type I Error
預測 No	FN (False Negative) Type II Error	TN (True Negative)

混淆矩陣 (Confusion Matrix)

- True Positive 和 True Negative 都為正確判斷，愈多愈好。
- False Positive 和 True Negative 都為正確判斷，愈多愈好。

模型評估

- Accuracy 看答對的比例 (對 Imbalance Data, 並非好指標)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total Number of Samples}}$$

- Precision(準確率): 預測正向的情形下，實際的精準度是多少。eg. 門禁系統

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall(召回率): 在實際情形為正向的狀況下，預測能召回多少實際正向的答案。eg. 廣告投放

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

-

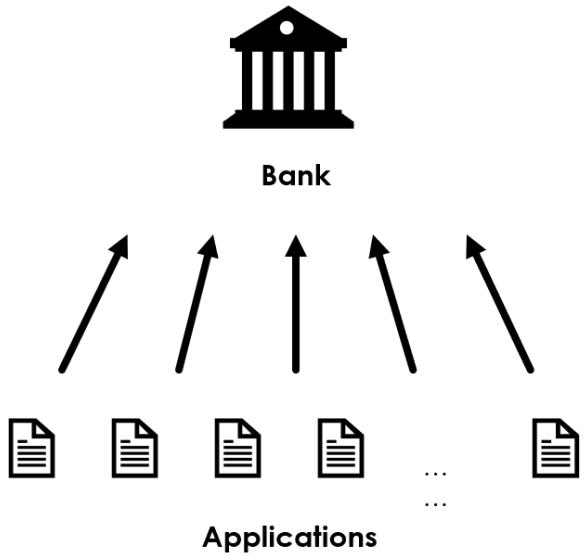
$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

大數據分析實例:

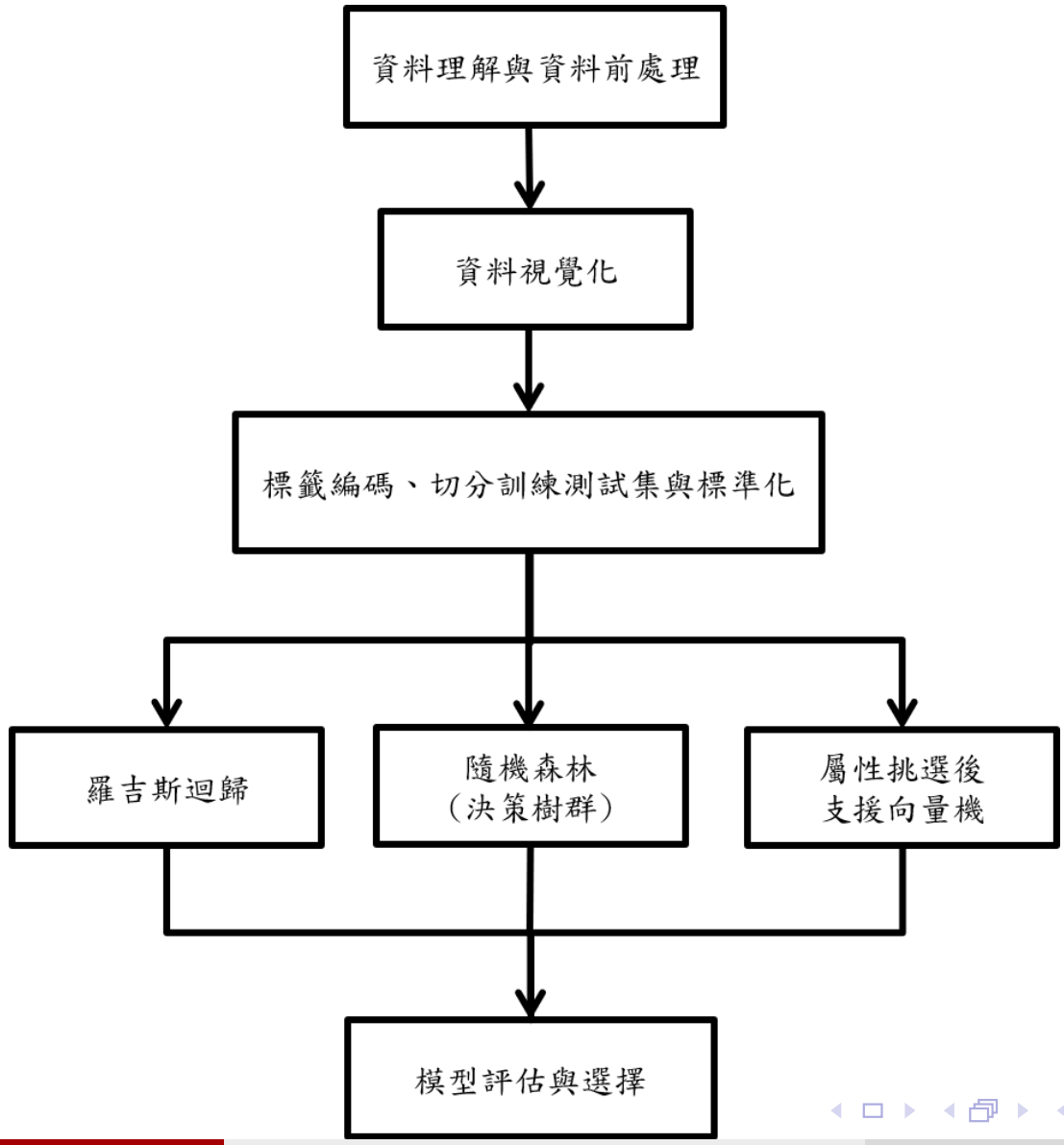
信用卡申請審查

目標

- 銀行業收到了許多信用卡申請。
- 手動處理每份申請書非常耗時，且需要許多人力，還容易出現人為錯誤。
- 因此，希望可以利用歷史數據來建立一個模型，幫忙篩選出待批准的候選申請者，以協助減少處理的人力成本及客戶等待的時間。



流程圖


資料理解與資料前處理
資料視覺化
標籤編碼、切分訓練測試集與標準化
羅吉斯迴歸
隨機森林
（決策樹群）
屬性挑選後
支援向量機
模型評估與選擇

資料初探

- 資料來源：
  http://archive.ics.uci.edu/ml/datasets/credit+approval
- 資料集包含過去 690 份信用卡申請 (690 筆觀測值)，每筆觀測值有 16 個變數，其中15 個變數表示申請者的各項資訊，例如：年齡、性別、婚姻狀況、工作年限等；最後一個變數表示申請結果是批准還是拒絕。

	Male	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	Approved
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

基本統計量

- 查看基本統計量，可以發現部分變數有一些遺缺值，在主要的目標變數 Approved 可以看出他似乎是個平衡的數據，目標變數”批准”與”拒絕”接近一半一半。

	Male	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	Approved
count	678	678.000000	690.000000	684	684	681	681	690.000000	690	690	690.00000	690	690	677	690.000000	690
unique	2	NaN	NaN	3	3	14	9	NaN	2	2	NaN	2	3	170	NaN	2
top	b	NaN	NaN	u	g	c	v	NaN	t	f	NaN	f	g	00000	NaN	-
freq	468	NaN	NaN	519	519	137	399	NaN	361	395	NaN	374	625	132	NaN	383
mean	NaN	31.568171	4.758725	NaN	NaN	NaN	NaN	2.223406	NaN	NaN	2.40000	NaN	NaN	NaN	1017.385507	NaN
std	NaN	11.957862	4.978163	NaN	NaN	NaN	NaN	3.346513	NaN	NaN	4.86294	NaN	NaN	NaN	5210.102598	NaN
min	NaN	13.750000	0.000000	NaN	NaN	NaN	NaN	0.000000	NaN	NaN	0.00000	NaN	NaN	NaN	0.000000	NaN
25%	NaN	22.602500	1.000000	NaN	NaN	NaN	NaN	0.165000	NaN	NaN	0.00000	NaN	NaN	NaN	0.000000	NaN
50%	NaN	28.460000	2.750000	NaN	NaN	NaN	NaN	1.000000	NaN	NaN	0.00000	NaN	NaN	NaN	5.000000	NaN
75%	NaN	38.230000	7.207500	NaN	NaN	NaN	NaN	2.625000	NaN	NaN	3.00000	NaN	NaN	NaN	395.500000	NaN
max	NaN	80.250000	28.000000	NaN	NaN	NaN	NaN	28.500000	NaN	NaN	67.00000	NaN	NaN	NaN	100000.000000	NaN

- 審批結果：共 690 筆：307(批准),383(拒絕) 資料不全：37(5%)

資料前處理

- 將數值欄位的遺缺值用平均值填補。
- 將非連續型的資料用眾數填補。

```
# 各欄位NA的數量
data.isnull().sum()
```

```
Male               12
Age                12
Debt                0
Married             6
BankCustomer        6
EducationLevel      9
Ethnicity           9
YearsEmployed       0
PriorDefault        0
Employed            0
CreditScore         0
DriversLicense      0
Citizen             0
ZipCode            13
Income              0
Approved            0
dtype: int64
```

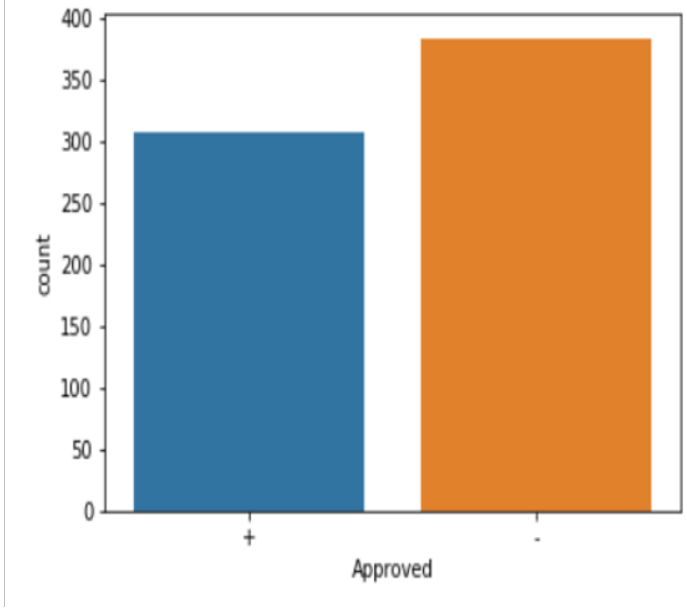
```
# 將數值欄位的遺缺值用平均值填補
data.fillna(data.mean(), inplace=True)
```

```
# 將非連續型的資料用眾數填補
def imputeWithMode(df):
    """
    Going through each columns and checking the type is object
    if it is object, impute it with most frequent value
    """
    for col in df:
        if df[col].dtypes == 'object':
            df[col] = df[col].fillna(df[col].mode().iloc[0])
imputeWithMode(data)
```

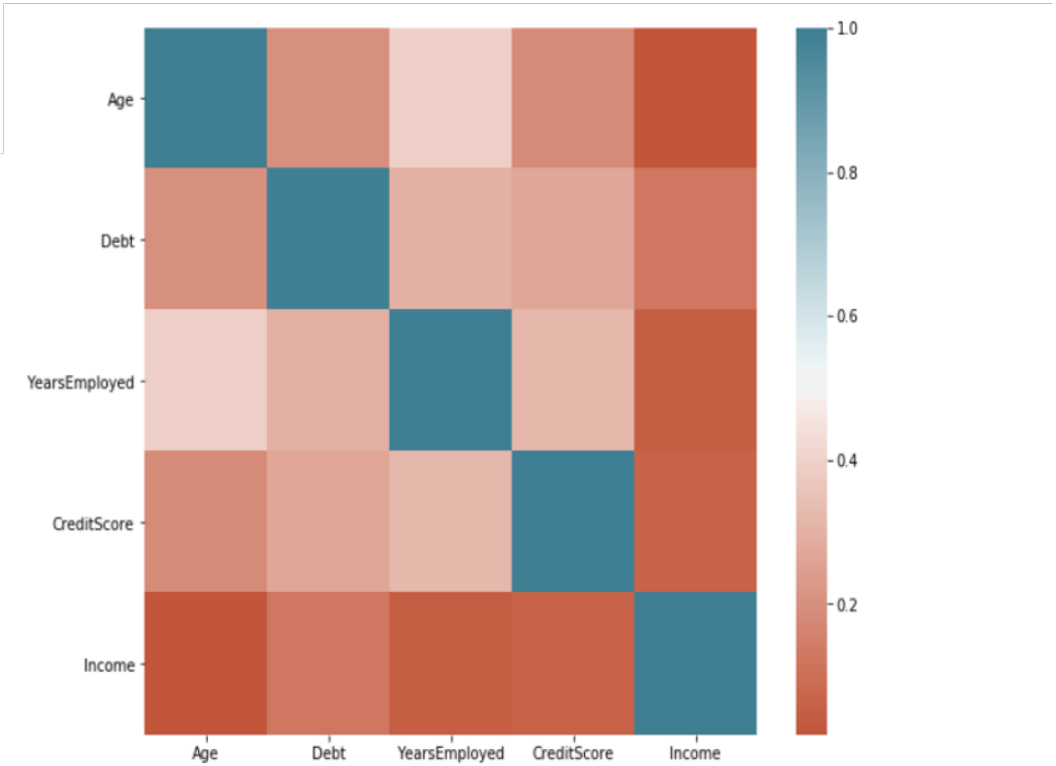
資料視覺化

- 將數值欄位的遺缺值用平均值填補。
- 將非連續型的資料用眾數填補。

批准與拒絕接近一半一半。

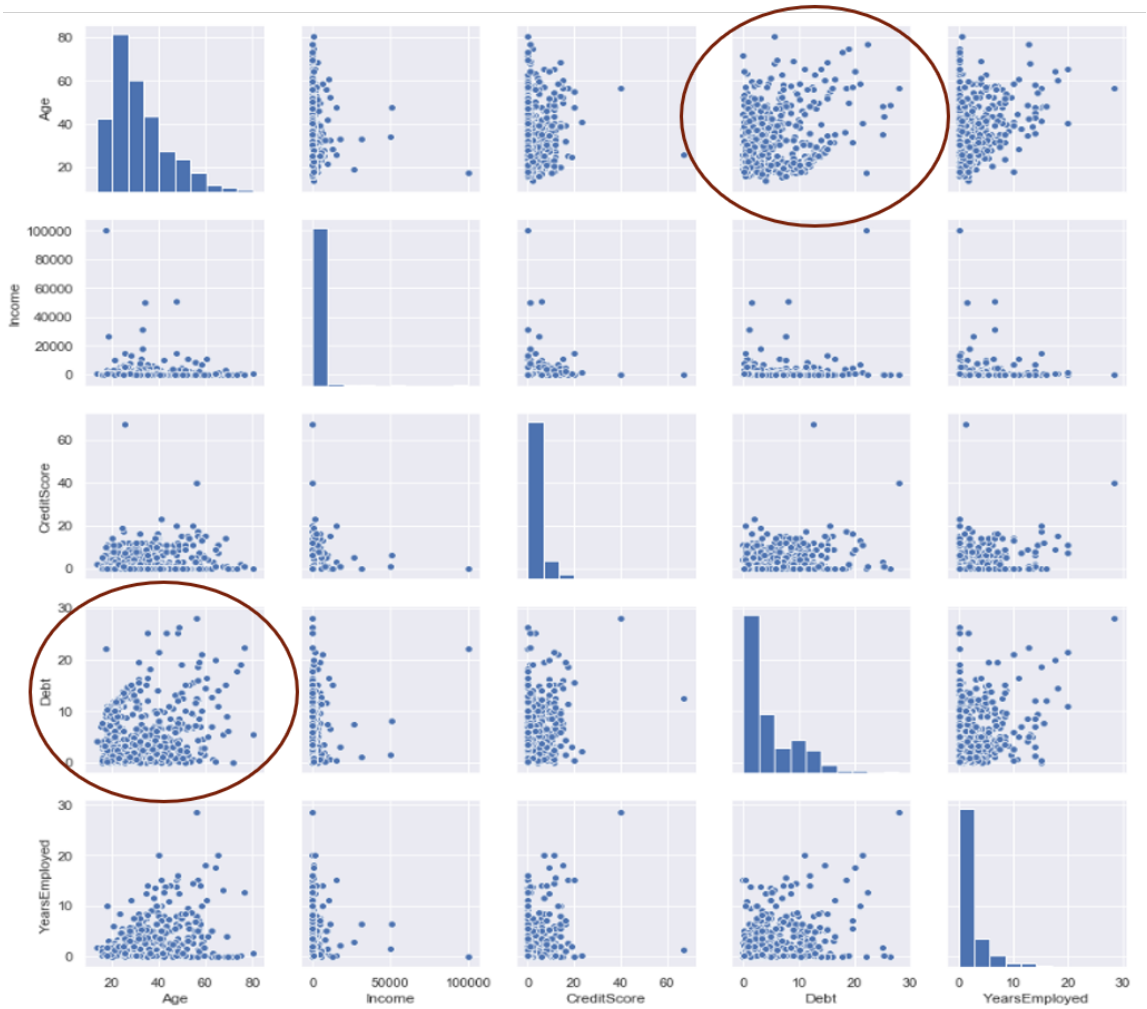


- 年齡與工作年資比較明顯正相關



資料視覺化

- 年齡、工作年資與債務有比較明顯正相關



標籤編碼

- 針對不連續的資料作標籤編碼。

```
# 針對不連續的資料作標籤編碼
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
# # Looping for each object type column
#Using label encoder to convert into numeric types
for col in data:
    if data[col].dtypes=='object':
        data[col]=le.fit_transform(data[col])
```

```
data.tail()
```

	Male	Age	Debt	Married	BankCustomer	EducationLevel	Ethnicity	YearsEmployed	PriorDefault	Employed	CreditScore	DriversLicense	Citizen	ZipCode	Income	Approved
685	1	21.08	10.085	2	2	4	3	1.25	0	0	0	0	0	90	0	1
686	0	22.67	0.750	1	0	1	7	2.00	0	1	2	1	0	67	394	1
687	0	25.25	13.500	2	2	5	2	2.00	0	1	1	1	0	67	1	1
688	1	17.92	0.205	1	0	0	7	0.04	0	0	0	0	0	96	750	1
689	1	35.00	3.375	1	0	1	3	8.29	0	0	0	1	0	0	0	1

資料切割成訓練集與測試集、並作標準化

- 將資料切分成訓練集與測試集：
  - 80% 作為訓練集
  - 20% 作為測試集
- 並做標準化：
  - 把資料縮放到介於 0–1 之間
- 資料準備完畢，就可輸入進決策樹群模型

演算法：隨機森林 (決策樹群)

```python
# 建立 random forest 模型
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,roc_auc_score
forest = RandomForestClassifier(n_estimators=800)
forest.fit(rescaledX_train, y_train)

# 預測並印出準確度
rfc_pred = forest.predict(rescaledX_test)
print("Random Forest classifier has accuracy of: ", forest.score(rescaledX_test, y_test))

# 評估混淆矩陣
print(confusion_matrix(y_test, rfc_pred))

rfc_probs = forest.predict_proba(rescaledX_test)
rfc_probs = np.delete(rfc_probs, 0, 1)
from sklearn import metrics
rfc_fpr, rfc_tpr, thresholds = metrics.roc_curve(y_test, rfc_probs)
rfc_auc = metrics.auc(rfc_fpr, rfc_tpr)
print("auc value: ", rfc_auc)
```

```
Random Forest classifier has accuracy of: 0.8695652173913043
[[48 14]
 [ 4 72]]
auc value:  0.9210526315789473
```

```python
pd.crosstab(y_test, rfc_pred,
            rownames=['actual'],
            colnames=['preds'])
```

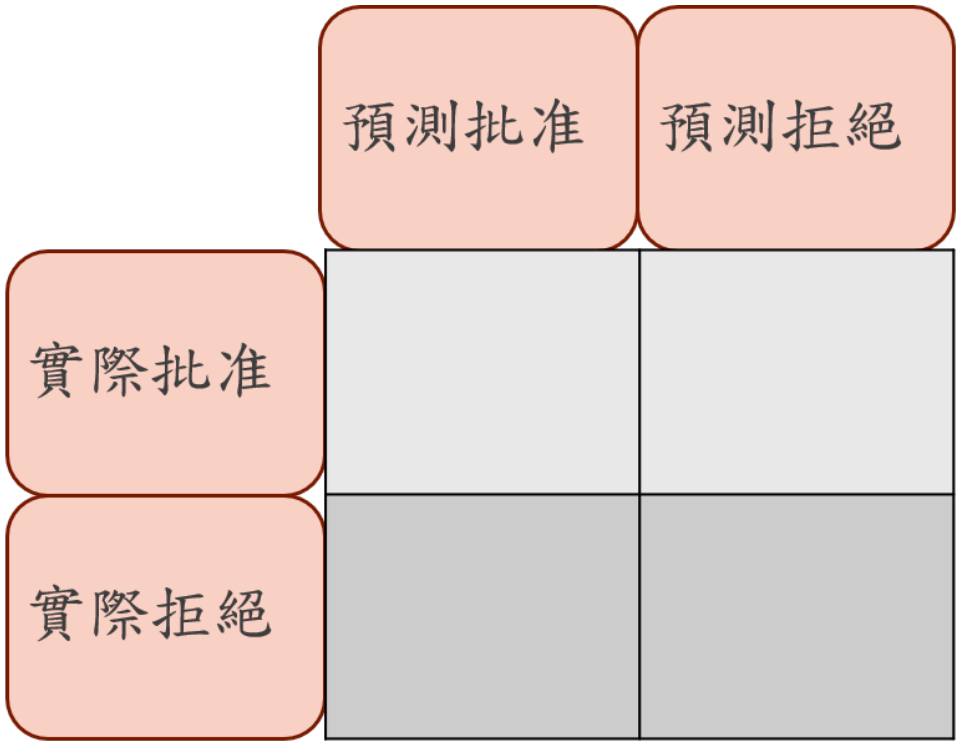
preds	0	1
actual		
0	48	14
1	4	72

0：批准
1：拒絕

138筆(20%)的資料作為測試集

評估準則：混淆矩陣

- 測試資料 138 筆



模型評估

- 47: 預測批准、實際也是批准
- 15: 預測拒絕、實際上是批准
- 4: 預測批准、實際是拒絕
- 72: 預測拒絕、實際也是拒絕

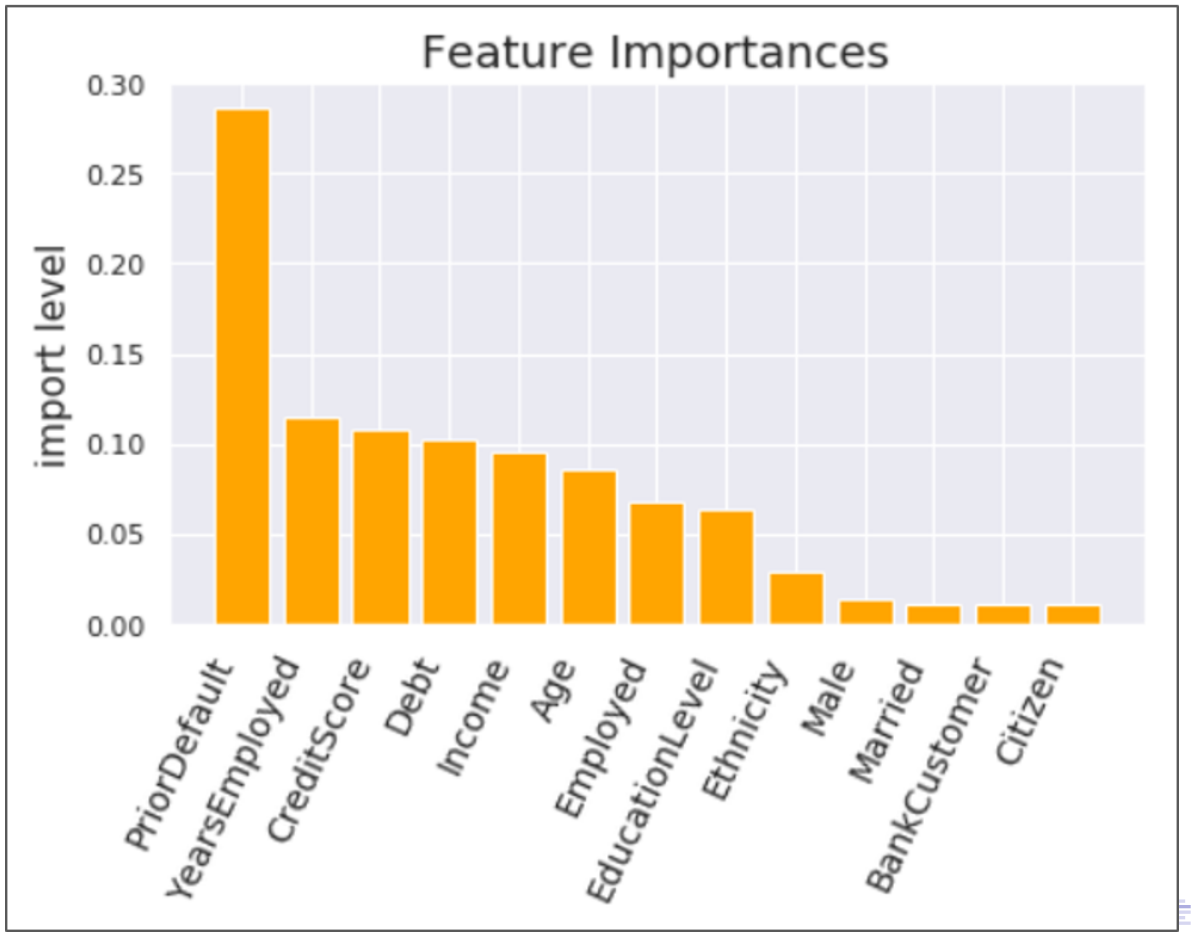
```
==================================
* Random Forest classifier
   accuracy =  0.8623188405797102
   auc value =  0.9211587436332767
   [47 15]
   [ 4 72]
==================================
* Logistic regression classifier
   accuracy =  0.8695652173913043
   auc value =  0.9093803056027163
   [54  8]
   [10 66]
```

```
* Support Vector classifier(feature selection)
   accuracy =  0.855072463768116
   auc value =  0.8565365025466892
   [54  8]
   [12 64]
```

模型評估

- 根據隨機森林分類器，事前違約是最重要的屬性，其次是工作年限，信用評分，債務，收入和年齡。

基於資訊增益(IG)而排序出來



小結

- 從混淆矩陣可觀察到，隨機森林 (決策樹群) 在預測信用卡拒絕 (不批准) 事件相較於其他模型來的好。
- 若以銀行的角度來思考，他們會比較在乎應該拒絕但反而沒被篩選到的申請，因為這類錯誤造成的後果相對嚴重。
- 因此在這案例中，相對其他模型，隨機森林 (決策樹群) 會是相對好的選擇。