Big Data Analytics

Project Report

Financial Distress Prediction

for Credit Scoring

團隊成員：蔡旻容、3位成員

One-page Executive Summary

現在社會資金流動越來越方便，電子支付的盛行下，可以連結各家信用卡使用消費，過去在**2005**年發生雙卡風暴，持卡人信用過度擴張，在個人無法還款時，形成個人卡債，造成銀行呆帳的金融事件。

自**1997**年的亞洲金融風暴開始，台灣銀行逾放款增加，企業融資管道增加，企業金融業務開始緊縮。銀行開始轉向個人金融業務。銀行開始推動信用卡業務，但大眾當時尚不清楚信用卡分期付款的利息影響，造成信用擴張。**1999**年萬泰銀行推出「喬治瑪麗」現金卡，因景氣不佳，多人申辦，也有許多銀行引進現金卡業務，最後造成許多信用卡和現金卡的用戶未能如期繳款，因而造成雙卡風暴。

銀行在市場經濟中發揮著重要的作用，他們決定了誰可以獲得融資及在什麼條件下獲得融資，並可以做出或破壞投資決策。為了使市場和社會發揮作用，個人和公司需要獲得信貸。信用評分演算法可以猜測違約機率，是銀行用來決定是否應發放貸款的方法。

我們的專案是透過 **Kaggle** 的借款人相關資料去建立預測模型，藉由預測某人在未來兩年內經歷財務困境的機率**(**是否會發生逾期還款的機率**)**，來改善信用評分的狀況。**Kaggle** 提供的借款人相關資料，其包含四個資料：訓練資料集、測試資料集、提交樣本和資料字典。資料字典是訓練資料和測試資料中每個屬性的含義。資料集則包含如借款人年齡、月收入、負債比率、過去一段時間內逾期還款次數等欄位。

透過探索這些欄位屬性，利用多個特徵分析，使用大量已給出標籤的訓練資料來訓練，並建立決策樹與隨機森林的模型，再使用超參數調整來提高模型的準確性，達成預測借款人是否會逾期還款的目標，用以幫助銀行來決定是否發放貸款，並使用該模型來幫助銀行做出最佳財務決策。

Detailed Report

1. Problem Description

❖ Business Goal 商業目標

透過預測某人在未來兩年內經歷財務困境的機率來改善信用評分的最新水平，並找出什麼樣條件下的人容易逾期繳款(哪些變數會導致借款人逾期繳款)，幫助銀行來決定是否發放貸款，做出最佳財務決策。使用此模型在商業利益上，可以提升信用評分的準確率，減少人為判斷錯誤的風險，提高利息獲益，降低呆帳率，把資金放貸於合適的個人與公司。對於信用評分較高者，銀行可以給予更優質的貸款方案(較高貸款金額、較低利率或較長寬限期)，而信用評分較低者，銀行可以給予較為保守的貸款方案(較低貸款金額、較高利率或較短寬限期)。

❖ Machine Learning Goal 機器學習目標

利用多個特徵分析預測借款人是否會逾期還款，這是一個監督式學習的預測任務。

2. Data description

❖ kaggle 資料集

Give Me Some Credit | Kaggle

❖ 資料欄位屬性

Variable Name	Description	Type
SeriousDlqin2yrs	**Person experienced 90 days past due delinquency or worse**	**Y/N**
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits	percentage
age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years.	integer
DebtRatio	Monthly debt payments, alimony,living costs divided by monthy gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due.	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

中文

變數名稱	描述	型態
兩年內嚴重 Dlq	逾期逾期 90 天或更嚴重的人	percentage
年齡	借款人年齡（歲）	integer
逾期次數 30-59 天 嚴重程度	過去 2 年內，借款人逾期 30-59 天但沒有更嚴重的情況的次數。	integer
資產負債比率	每月債務償還、贍養費、生活費除以每月總收入	percentage
月收入	月收入	real
開放信貸額度和貸款數量	公開貸款數量（汽車貸款或抵押貸款等分期付款）和信用額度（例如信用卡）	integer
遲到 90 天的次數	借款人逾期 90 天或以上的次數。	integer
數量房地產貸款或額度	抵押貸款和房地產貸款數量，包括房屋淨值信貸額度	integer
逾期次數 60-89 天 嚴重程度	過去 2 年內，借款人逾期 60-89 天但沒有更嚴重的情況的次數。	integer
家屬關係	家庭中不包括其本人的受扶養人數（配偶、子女等）	integer

❖ 使用資料欄位、數量與樣本

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19382 entries, 120693 to 149979
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   RevolvingUtilizationOfUnsecuredLines  19382 non-null  float64
 1   age                                   19382 non-null  float64
 2   NumberOfTime30-59DaysPastDueNotWorse  19382 non-null  float64
 3   DebtRatio                             19382 non-null  float64
 4   MonthlyIncome                         19382 non-null  float64
 5   NumberOfOpenCreditLinesAndLoans       19382 non-null  float64
 6   NumberOfTimes90DaysLate               19382 non-null  float64
 7   NumberRealEstateLoansOrLines          19382 non-null  float64
 8   NumberOfTime60-89DaysPastDueNotWorse  19382 non-null  float64
 9   NumberOfDependents                    19382 non-null  float64
dtypes: float64(10)
memory usage: 1.6 MB
```

]:
```
input_cols = ['RevolvingUtilizationOfUnsecuredLines', 'age',
       'NumberOfTime30-59DaysPastDueNotWorse', 'DebtRatio', 'MonthlyIncome',
       'NumberOfOpenCreditLinesAndLoans', 'NumberOfTimes90DaysLate',
       'NumberRealEstateLoansOrLines', 'NumberOfTime60-89DaysPastDueNotWorse',
       'NumberOfDependents']
target_col = 'SeriousDlqin2yrs'
```

]:
```
inputs = train_df[input_cols].copy()
targets = train_df[target_col].copy()
```

In [37]:
```
inputs.head()
```

Out[37]:

	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpenCred
9791	0.001930	0.282907	0.000000	0.000283	0.200331	0.366747
116338	0.000353	0.565814	0.000000	0.000080	0.188076	0.419140
110840	0.000469	0.229862	0.053369	0.019545	0.159951	0.104785
105547	0.001637	0.618859	0.160107	0.000204	0.528520	0.995457
4162	0.000368	0.159135	0.053369	0.000040	0.186636	0.366747

In [38]:
```
test_inputs.head()
```

Out[38]:

	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpenCreditLines
0	0.001771	0.388997	0.000000	0.000048	0.182381	0.209570
1	0.000926	0.636540	0.000000	0.000141	0.292481	0.785887
2	0.000087	0.671904	0.000000	0.000184	0.162639	0.628710
3	0.000561	0.300589	0.053369	0.000248	0.102389	0.366747
4	0.002000	0.106090	0.000000	0.000005	0.123667	0.209570

3. Brief Data Preparation Details

❖ 資料前處理

(1) 查看訓練資料與測試資料的統計資訊

(2) 做undersampling

```
train_df['SeriousDlqin2yrs'].value_counts()
```

```
0    139974
1     10026
```

```
train0=train_df[train_df['SeriousDlqin2yrs']==0].sample(frac=0.06684)
train1=train_df[train_df['SeriousDlqin2yrs']==1].copy()
train_df=pd.concat([train0, train1], axis=0)
train_df['SeriousDlqin2yrs'].value_counts()
```

```
1    10026
0     9356
Name: SeriousDlqin2yrs, dtype: int64
```

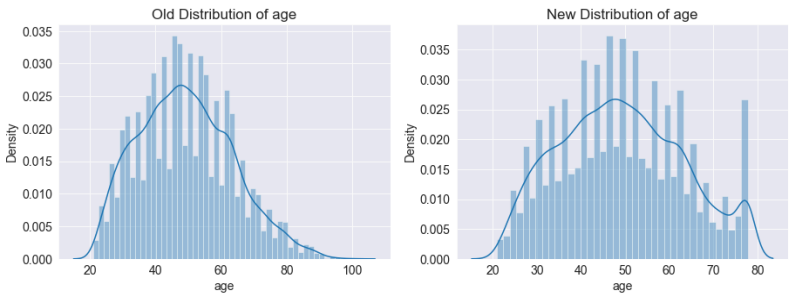
(3) Normalizer. 用兩倍標準差以下的值來取代異常值

選擇兩倍標準差，因為它涵蓋了分佈部分的 **95%**

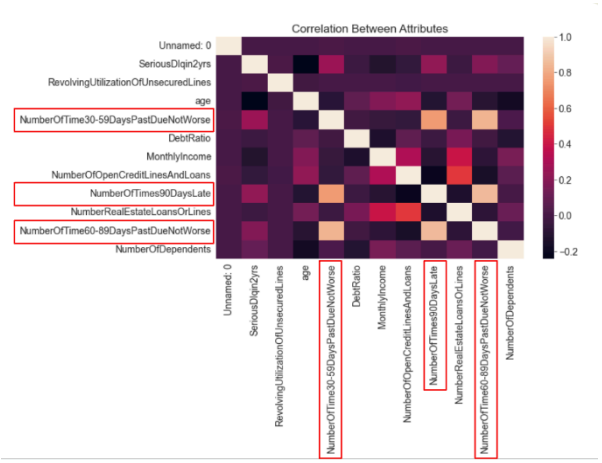
```
def normalizer(x, df):
        stats = df[x].describe()

        return {
                'Attribute': x,
                'upper_boundary': stats['mean'] + 2 * stats['std'],
                'lower_boundary': stats['mean'] - 2 * stats['std'],
                'max_att': stats['max'],
                'min_att': stats['min']
        }
```

Training data 的 Normalizer前(左) & Normalizer後(右)



(4) 使用heatmap查看欄位的Correlation (Training data)



(5) Imputing missing values，用該列的中位數填補缺失值

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(strategy = 'median').fit(train_df[input_cols])
```

```
inputs[input_cols] = imputer.transform(inputs[input_cols])
test_inputs[input_cols] = imputer.transform(test_inputs[input_cols])
```

```
inputs[input_cols].isna().sum()
```

```
RevolvingUtilizationOfUnsecuredLines    0
age                                     0
NumberOfTime30-59DaysPastDueNotWorse    0
DebtRatio                               0
MonthlyIncome                           0
NumberOfOpenCreditLinesAndLoans         0
NumberOfTimes90DaysLate                 0
NumberRealEstateLoansOrLines            0
NumberOfTime60-89DaysPastDueNotWorse    0
NumberOfDependents                      0
dtype: int64
```

(6) Scaling Numeric Features 將資料縮放成0到1

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler().fit(train_df[input_cols])
```

```
inputs[input_cols] = scaler.transform(inputs[input_cols])
test_inputs[input_cols] = scaler.transform(test_inputs[input_cols])
```

```
inputs.describe().loc[['min', 'max']]
```

```
inputs.head()
```

	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome
75466	0.002598	0.511027	0.0	0.000000	0.144045
32784	0.000243	0.405297	0.0	0.000198	0.324183
80257	0.001149	0.669622	0.0	1.000000	0.270152
78260	0.000514	0.158595	0.0	0.000012	0.372054
48423	0.000458	0.422919	0.0	0.000156	1.000000

```
test_inputs.head()
```

	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	Numb
0	0.002300	0.387676	0.000000	0.000066	0.307974	
1	0.001203	0.634378	0.000000	0.000197	0.493892	
2	0.000112	0.669622	0.000000	0.000257	0.274637	
3	0.000728	0.299568	0.053548	0.000346	0.172897	
4	0.002598	0.105730	0.000000	0.000007	0.208828	

4. Machine Learning Solution

❖ Decision Tree 決策樹模型

(1) Splitting 拆分資料集

將資料集**(inputs**和**targets)**分成訓練集和驗證集。
利用訓練集的資料來訓練模型，然後使用驗證集的資料來評估模型的性能和泛化能力。**test_size**：驗證集的比例，**0.25**代表將資料集的**25%**分配給驗證集。
random_state：確保每次運行程式時，資料分割的結果都是相同的。

```
from sklearn.model_selection import train_test_split
X_train, X_val, train_targets, val_targets = train_test_split(inputs, targets, test_size
=0.25,random_state=42)
```

(2) Training

使用**sklearn.tree**的**DecisionTreeClassifier** 訓練決策樹。根據資料集中的特徵值和目標值來構建決策樹模型，以便模型能夠學習資料的模式和特徵。

(3) Evaluation

評估訓練集上決策樹模型的性能。預測結果、預測概率、以及準確率等評估指標可以幫助了解模型在訓練數據上的效果，並且有助於判斷模型是否存在過度擬合或者其他問題。

```
train_preds = model.predict(X_train)
```

```
train_probs = model.predict_proba(X_train)
```

```
base_acc = accuracy_score(train_targets, train_preds), model.score(X_val, val_targets)
base_acc
```

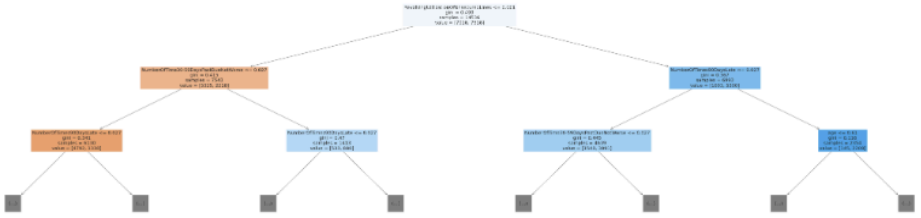
```
(0.999724821133737, 0.7028477094510936)
```

在這個模型中，訓練集的準確率接近**100%**，接著評估模型在驗證集上的表現，並比較訓練集和驗證集上的準確率。我們發現這個模型有**overfitting**的問題，模型看起來已經完美地學習了訓練的範例，並且不能很好地泛化到以前未見過的範例。

(4) Visualization

使用**plot_tree**函數來視覺化決策樹。

```
plot_tree(model, feature_names=X_train.columns, max_depth=2, filled=True);
```



❖ RandomForest

(1) Training and Evaluation

使用**sklearn.tree**的**RandomForestClassifier**訓練隨機森林。根據資料集中的特徵值和目標值來構建隨機森林模型，以便模型能夠學習資料的模式和特徵。

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_jobs=-1, random_state=42)

model.fit(X_train, train_targets)
```

```
model.score(X_train, train_targets)
```

```
0.9996560264171711
```

```
model.score(X_val, val_targets)
```

```
0.7816756087494842
```

(2) Feature Importance：判斷每個欄位對於結果的影響程度。

	feature	importance
0	RevolvingUtilizationOfUnsecuredLines	0.271561
3	DebtRatio	0.133339
1	age	0.120223
4	MonthlyIncome	0.114178
2	NumberOfTime30-59DaysPastDueNotWorse	0.095900
6	NumberOfTimes90DaysLate	0.078722
5	NumberOfOpenCreditLinesAndLoans	0.078617
8	NumberOfTime60-89DaysPastDueNotWorse	0.040939
9	NumberOfDependents	0.033597
7	NumberRealEstateLoansOrLines	0.032925

```
importance_df = pd.DataFrame({
    'feature': X_train.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
importance_df.head(10)
```

❖ Hyperparameter Tuning

利用迴圈測試髓設定的超參數數值，並圖示化準確率在不同超參數設定上的表現，找出適合的超參數。

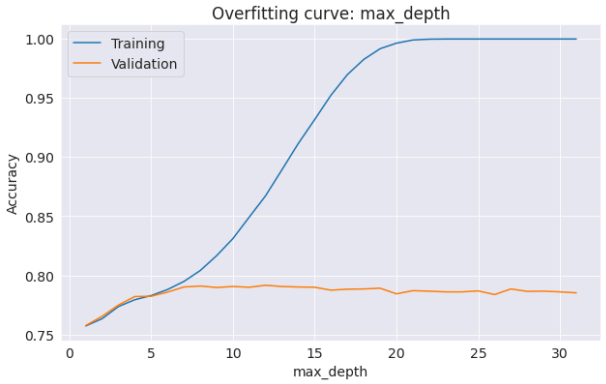
```
def test_params(**params):
    model = RandomForestClassifier(random_state=42, n_jobs=-1, **params)
    model.fit(X_train, train_targets)
    train_score = accuracy_score(model.predict(X_train), train_targets)
    val_score = accuracy_score(model.predict(X_val), val_targets)
    return train_score, val_score
n_estimators=[100, 200, 300, 400, 500,600, 700, 800, 900, 1000]
test_param_and_plot('n_estimators',n_estimators)
```

```
def test_params(**params):
    model = RandomForestClassifier(random_state=42, n_jobs=-1,n_estimators=800, **params)
    model.fit(X_train, train_targets)
    train_score = accuracy_score(model.predict(X_train), train_targets)
    val_score = accuracy_score(model.predict(X_val), val_targets)
    return train_score, val_score
max_depth=[i for i in range(1,32)]
test_param_and_plot('max_depth',max_depth)
```

```
model = RandomForestClassifier(random_state=42, n_jobs=-1,
                               n_estimators=800, max_depth=12,
                               max_leaf_nodes=89, max_features=5,
                               min_samples_split=11,
                               min_samples_leaf=1,
                               min_impurity_decrease=0.0)
model.fit(X_train, train_targets)
print(base_accs)
model.score(X_train, train_targets), model.score(X_val, val_targets)
```

```
(0.9996560264171711, 0.7816756087494842)

(0.8097826086956522, 0.7942633099463475)
```



5. Conclusions

- 用網格或隨機搜尋的方式尋找最佳參數，取代原始找超參數迴圈會更有效率。
- 在蒐集資料可以再增加 一些欄位資料，像是顧客職業資訊等，透過更多詳細資訊有助於預測。

6. 附錄

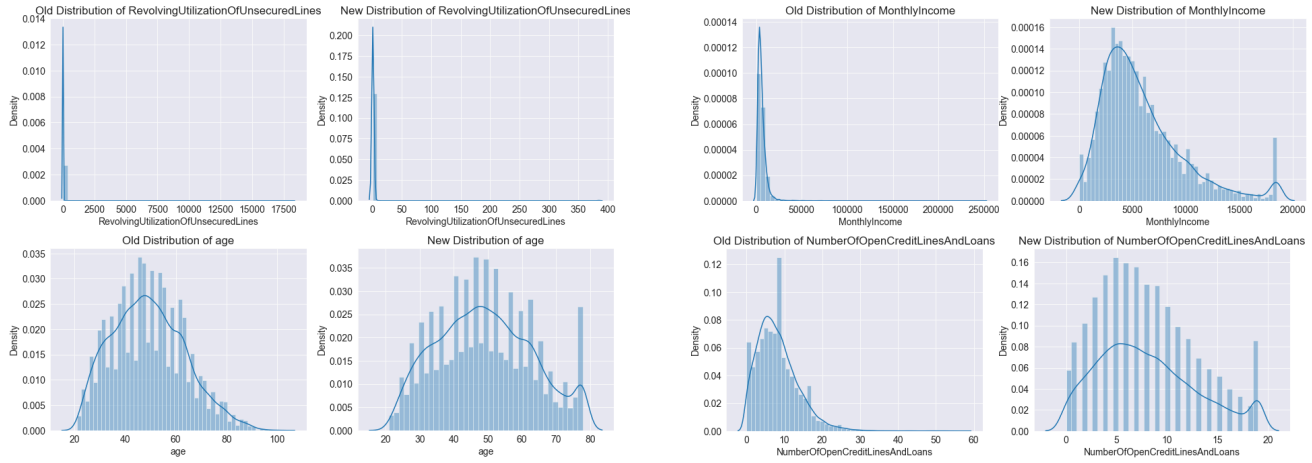
- 訓練資料與測試資料的統計資訊

```
In [9]: train_df.describe()
```

Out[9]:

	Unnamed: 0	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRat
count	150000.000000	150000.000000	150000.000000	150000.000000	150000.000000	150000
mean	75000.500000	0.066840	6.048438	52.295207	0.421033	353.005
std	43301.414527	0.249746	249.755371	14.771866	4.192781	2037.81
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.0000(
25%	37500.750000	0.000000	0.029867	41.000000	0.000000	0.1750
50%	75000.500000	0.000000	0.154181	52.000000	0.000000	0.3665(
75%	112500.250000	0.000000	0.559046	63.000000	0.000000	0.8682
max	150000.000000	1.000000	50708.000000	109.000000	98.000000	329664

- **Training data 每個欄位的 Normalizer前(左) & Normalizer後(右)**



- 訓練決策樹 **sklearn.tree的DecisionTreeClassifier**

```
In [40]: # 構建決策樹模型
         from sklearn.tree import DecisionTreeClassifier

In [41]: model = DecisionTreeClassifier(random_state=42)

In [42]: #fitting the model # 使用模型進行訓練
         # 根據資料集中的特徵值和目標值來構建決策樹模型，以便模型能夠學習資料的模式和特徵。
         model.fit(X_train, train_targets)

Out[42]: DecisionTreeClassifier(random_state=42)
```