房貸放款評估

- 評估客戶是否符合貸款標準
  - 性別
  - 婚姻狀況
  - 教育程度
  - 收入
  - 借貸金額
  - 信用紀錄

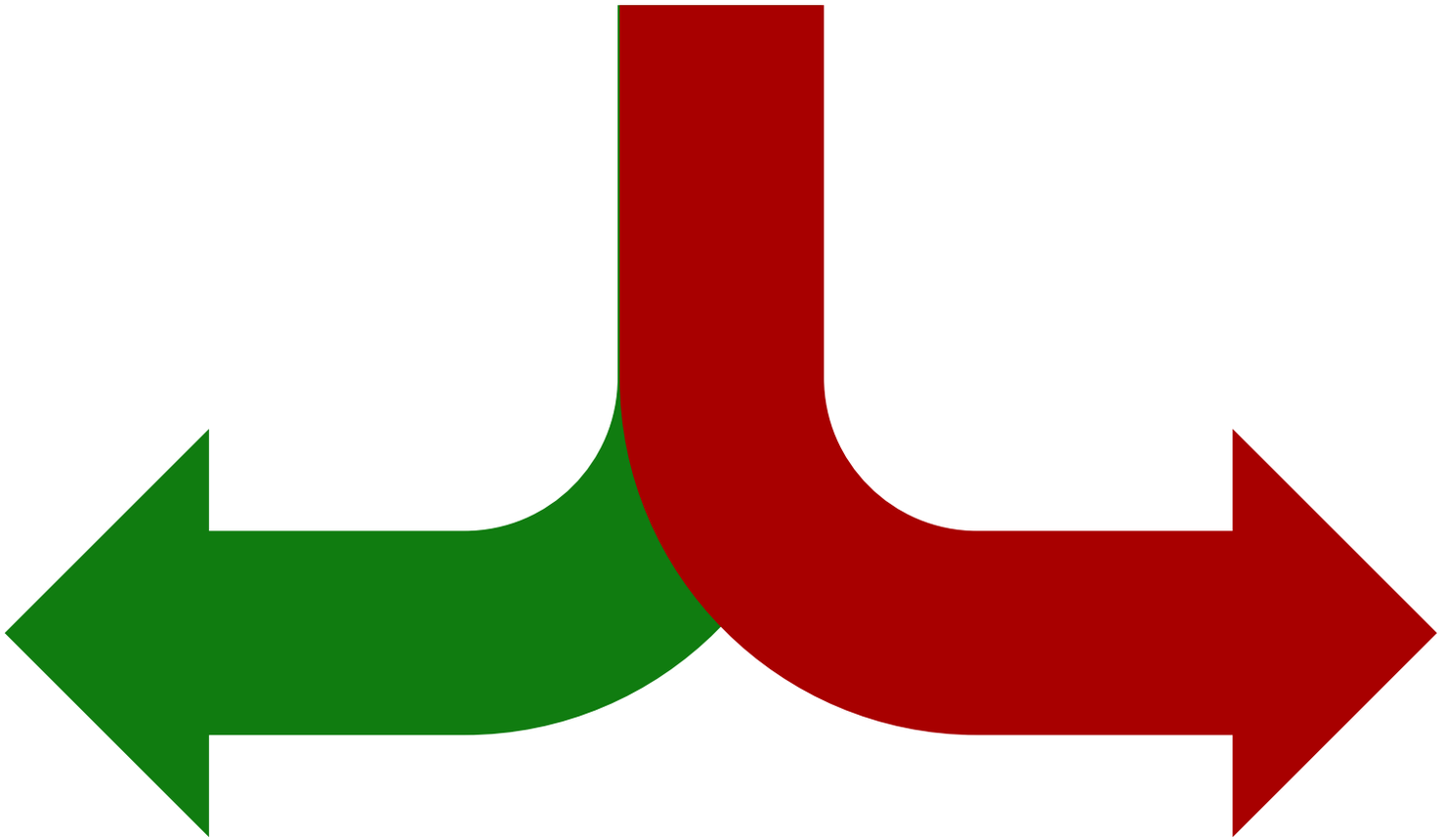
Ryan@MobileDev.TW

資料說明

變數名稱	說明	變數名稱	說明
Loan_ID	唯一識別ID	**CoapplicantIncome**	共同申請人收入
Gender	性別(Male/Female)	**LoanAmount**	借貸金額(美金千元)
Married	是否已婚 (Y/N)	**Loan_Amount_Term**	借貸時間(月)
Dependents	家屬人數	**Credit_History**	信用紀錄(1/0)
Education	教育程度 (Graduate/ Under Graduate)	**Property_Area**	房產位置 Urban/ Semi Urban/ Rural
Self_Employed	是否為自雇者 (Y/N)	**Loan_Status**	是否核准借貸 (Y/N)
ApplicantIncome	申請者本人收入		

目標

- 建立機器學習模型，決定是否要核准貸款
  - Yes
  - No



Ryan@MobileDev.TW

資料科學家的職業道德

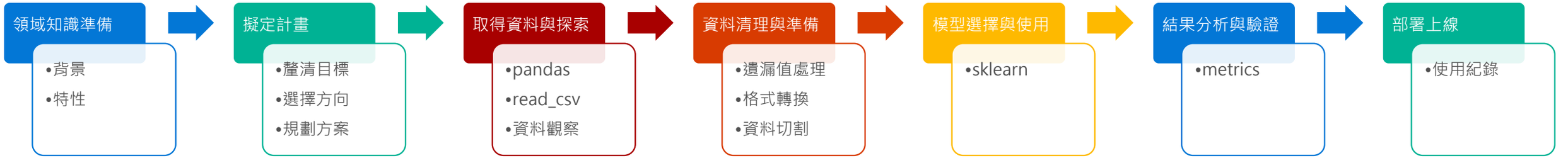
- 只蒐集必要的、分析需要的資料
- 界定與去除機敏性資料
- 判斷錯誤的備援方案準備

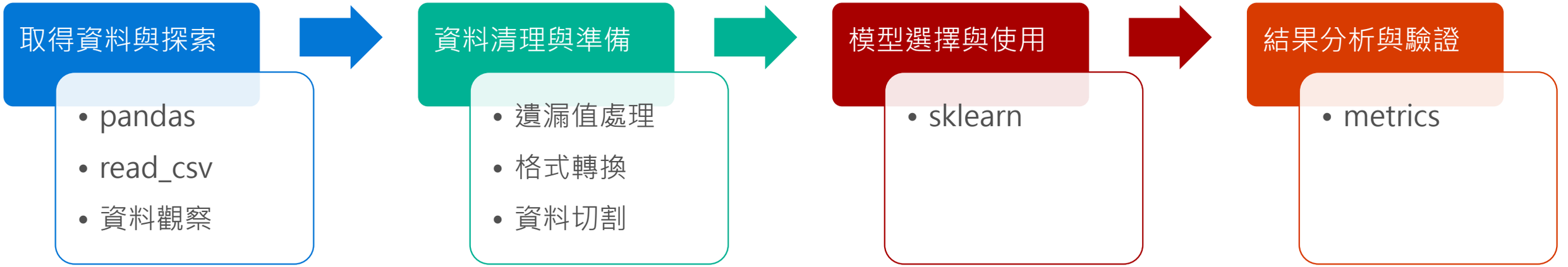
性別與婚姻狀態

- 屬於個人隱私資料，可考慮去除

Ryan@MobileDev.TW

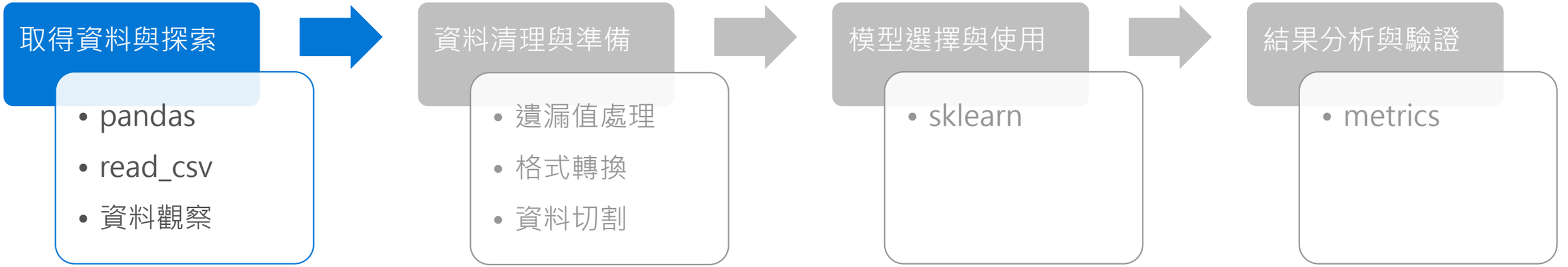
資料科學處理流程





Ryan@MobileDev.TW

資料科學處理流程




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib inline

df = pd.read_csv("loan_prediction_training_data.csv")
```


Ryan@MobileDev.TW

查閱方法說明

- 在VS Code中，將游標移置方法名稱上
- 例如：pd.read_csv

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("../../data/loan_prediction_training_data.csv")
```

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the online docs for `IO Tools <https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html>` _.

Parameters

filepath_or_buffer : str, path object or file-like object
Any valid string path is acceptable. The string could be a URL. Valid URL schemes include http, ftp, s3, and file. For file URLs, a host is expected. A local file could be: file://localhost/path/to/table.csv.

Ryan@MobileDev.TW

繼續探索資料 df.describe()

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
[7] df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Q.從這裡可得知哪幾項有缺失值?

Ryan@MobileDev.TW

繼續探索資料 df.describe()

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
[7] df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Q. Credit_History 為1的有幾筆?

Ryan@MobileDev.TW

繼續探索資料 df.info()

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

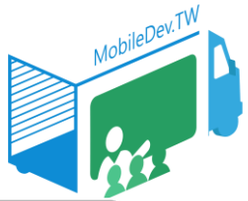
結果分析與驗證
- metrics

```
[8]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
Loan_ID                614 non-null object
Gender                 601 non-null object
Married                611 non-null object
Dependents             599 non-null object
Education              614 non-null object
Self_Employed          582 non-null object
ApplicantIncome        614 non-null int64
CoapplicantIncome      614 non-null float64
LoanAmount             592 non-null float64
Loan_Amount_Term       600 non-null float64
Credit_History         564 non-null float64
Property_Area          614 non-null object
Loan_Status            614 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.4+ KB
```

共有614筆資料

Ryan@MobileDev.TW

去除性別、婚姻資料



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
#剔除Gender, Married欄位與資料
df_no_G_M = df.drop(columns=['Gender','Married'])
#存成csv檔
df_no_G_M.to_csv('loan_prediction_training_data_no_G_M.csv')
```

```
[13] df.head()
```

	Loan_ID	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	0	Graduate	No	6000	0.0	141.0	360.0	1.0

Ryan@MobileDev.TW

調閱資料

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['Self_Employed'].value_counts()
df['Property_Area'].value_counts()
df['Education'].value_counts()
```

```
[22] df['Self_Employed'].value_counts()

No     500
Yes     82
Name: Self_Employed, dtype: int64
```

```
[23] df['Property_Area'].value_counts()

Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

```
[24] df['Education'].value_counts()

Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```


Ryan@MobileDev.TW

調閱資料 - 收入分布情形

取得資料與探索
- pandas
- read_csv
- 資料觀察

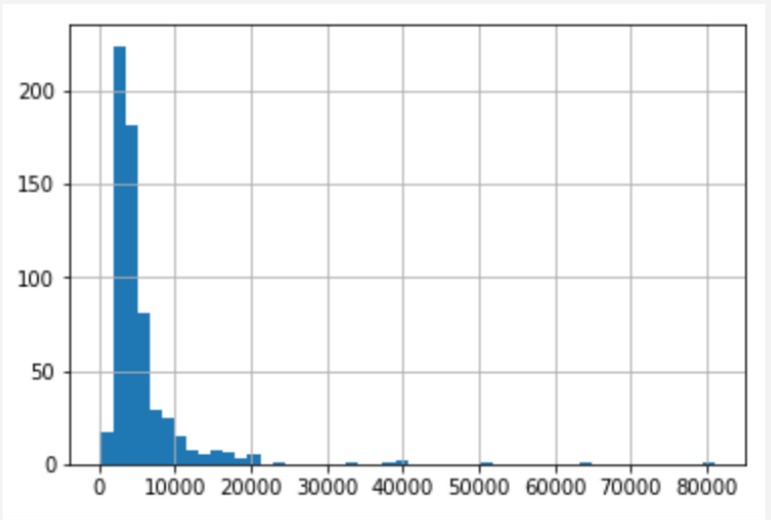
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['ApplicantIncome'].hist(bins=50)
```

```
[28] df['ApplicantIncome'].hist(bins=50)

<matplotlib.axes._subplots.AxesSubplot at 0x20c2bec9f60>
```



Ryan@MobileDev.TW

調閱資料 - 收入分布情形 – 換一種圖試試

取得資料與探索
- pandas
- read_csv
- 資料觀察

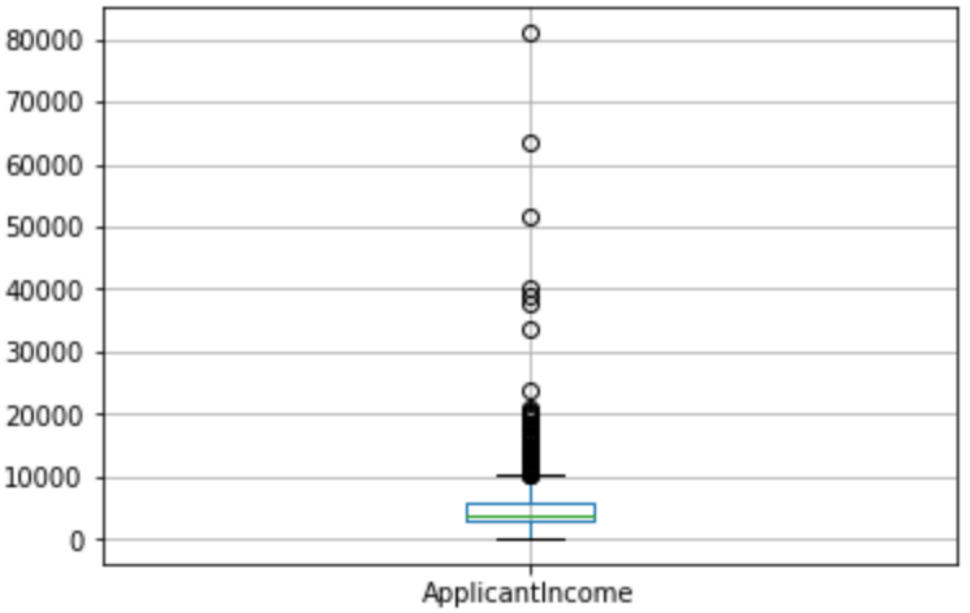
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df.boxplot(column='ApplicantIncome')
```

Q. 看來有蠻多收入特別高的人
跟教育程度有沒有關聯性呢?



Ryan@MobileDev.TW

調閱資料 - 收入分布情形 – 換一種圖試試

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

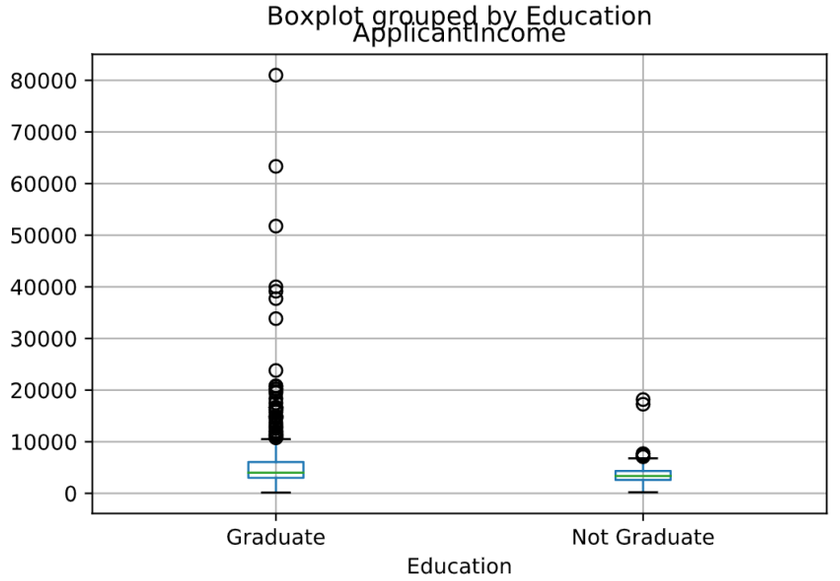
模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df.boxplot(column='ApplicantIncome', by = 'Education')
```

Q. 看來有蠻多收入特別高的人
跟教育程度有沒有關聯性呢?

幾乎都在有畢業的那一邊!



Ryan@MobileDev.TW

調閱資料 – 借貸金額分布情形

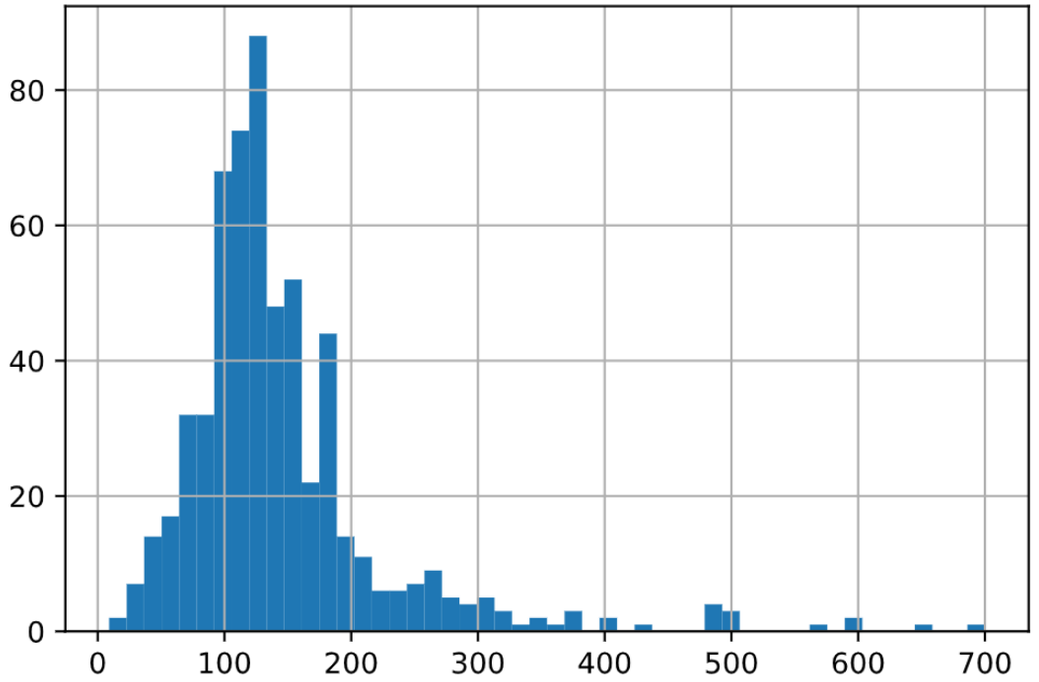
取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['LoanAmount'].hist(bins=50)
```



(美金千元)

調閱資料 -借貸金額分布情形 – 換一種圖

取得資料與探索
- pandas
- read_csv
- 資料觀察

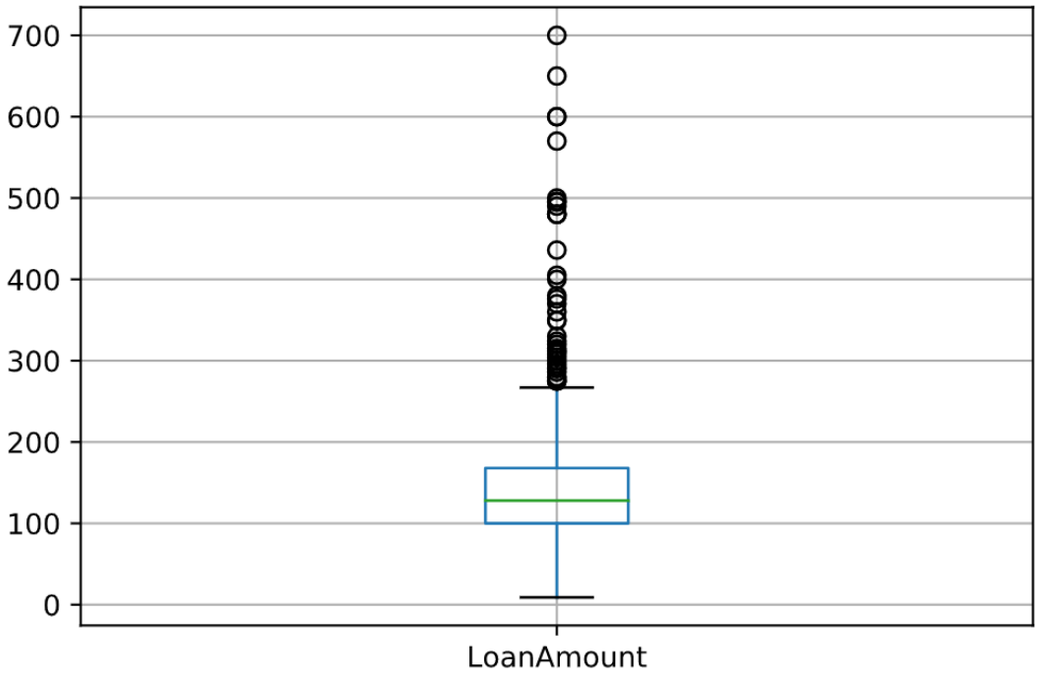
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df.boxplot(column='LoanAmount')
```

Q. 也有蠻多借貸金額特別高的人
跟教育程度有沒有關聯性呢?



Ryan@MobileDev.TW

調閱資料 -借貸金額分布情形 – 換一種圖

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

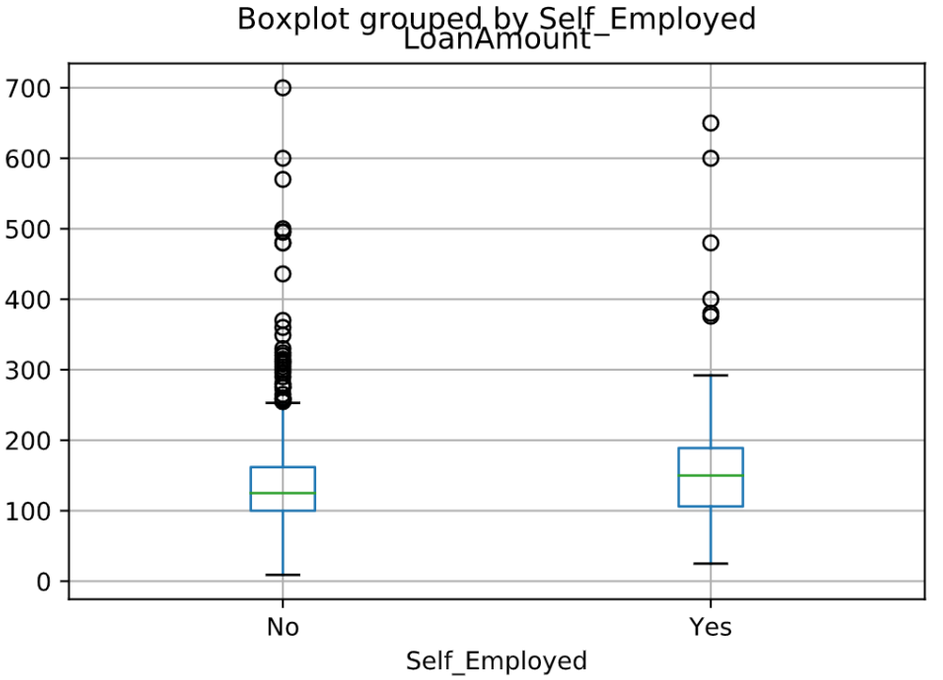
部署上線
- 使用紀錄

練習

借貸金額高低與是否為自營商的關聯

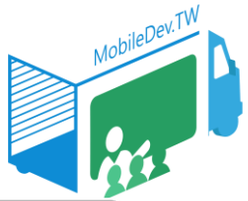
收入高低與是否為自營商的關聯

...



Ryan@MobileDev.TW

調閱資料 – 信用紀錄 VS. 借貸狀態



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
temp1 = df['Credit_History'].value_counts(ascending=True)
```

```
temp2 = df.pivot_table(values='Loan_Status', index=['Credit_History'],aggfunc=lambda
 x: x.map({'Y':1,'N':0}).mean())
```

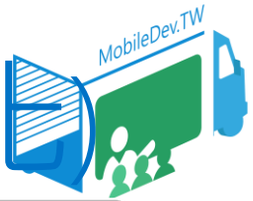
```
[45]  temp1

0.0     89
1.0    475
Name: Credit_History, dtype: int64
```

```
[47]  temp2

                Loan_Status
Credit_History
0.0                0.078652
1.0                0.795789
```

有信用紀錄的借貸成功比例高很多！

Ryan@MobileDev.TW

調閱資料 – 信用紀錄 VS. 借貸狀態(視覺化)



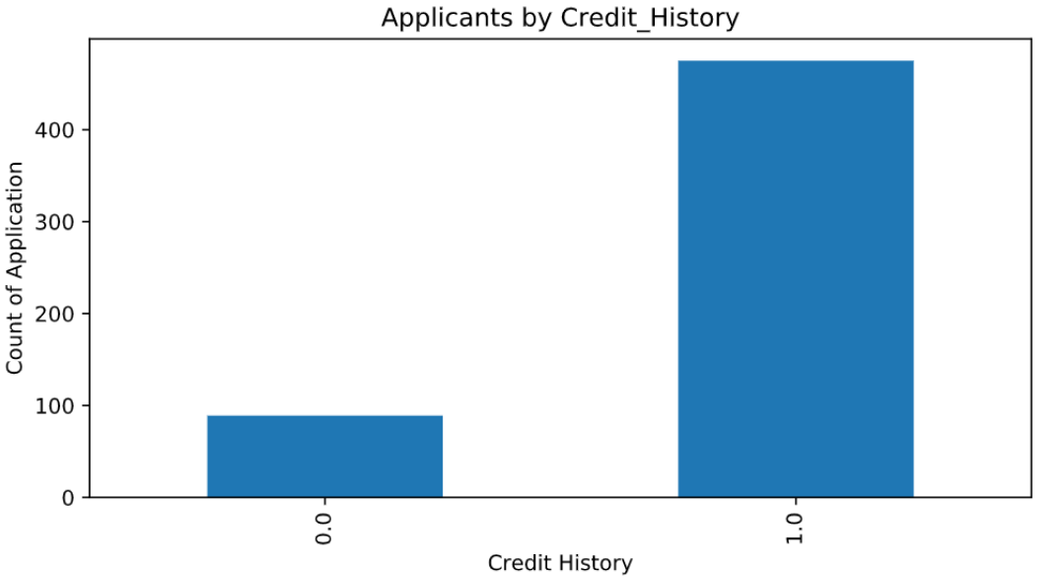
取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

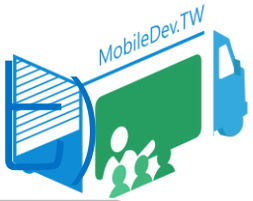
結果分析與驗證
- metrics

```
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(111)
ax1.set_xlabel('Credit History')
ax1.set_ylabel('Count of Application')
ax1.set_title('Applicants by Credit_History')
temp1.plot(kind = 'bar')
```



Ryan@MobileDev.TW

調閱資料 – 信用紀錄 VS. 借貸狀態(視覺化)



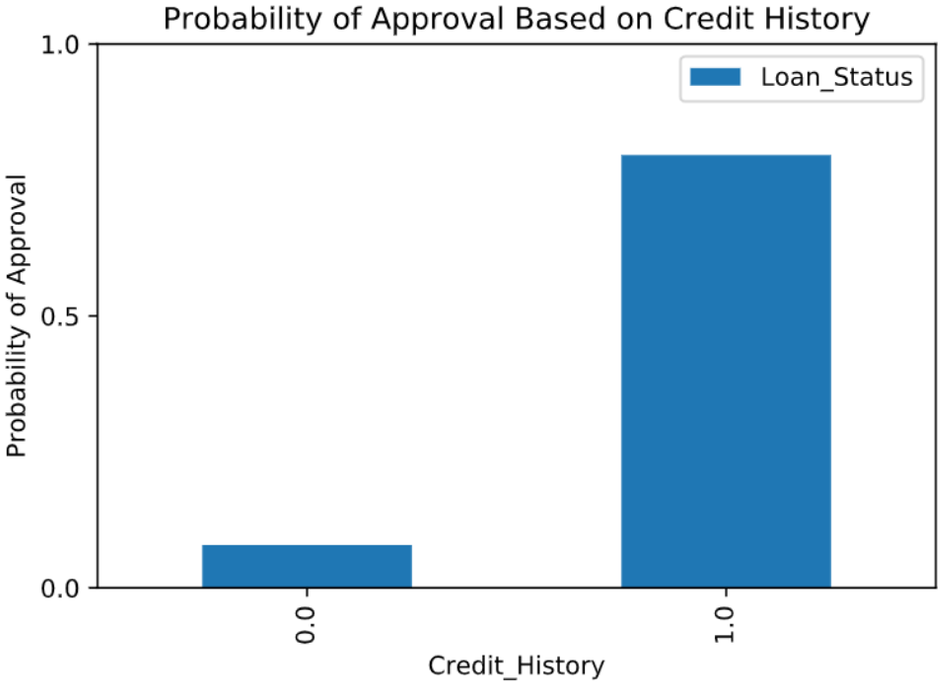
取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
temp2.plot(kind = 'bar',yticks=[0,0.5,1]
,ylabel='Probability of Approval',title=
'Probability of Approval Based on Credit
 History')
```



Ryan@MobileDev.TW

調閱資料 – 房產位置 VS. 借貸狀態

取得資料與探索

- pandas
- read_csv
- 資料觀察

資料清理與準備

- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用

- sklearn

結果分析與驗證

- metrics

練習

請用相同方式

觀察房產位置與借貸狀態是否有關連性

Ryan@MobileDev.TW

調閱資料 – 自雇者 VS. 借貸狀態

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

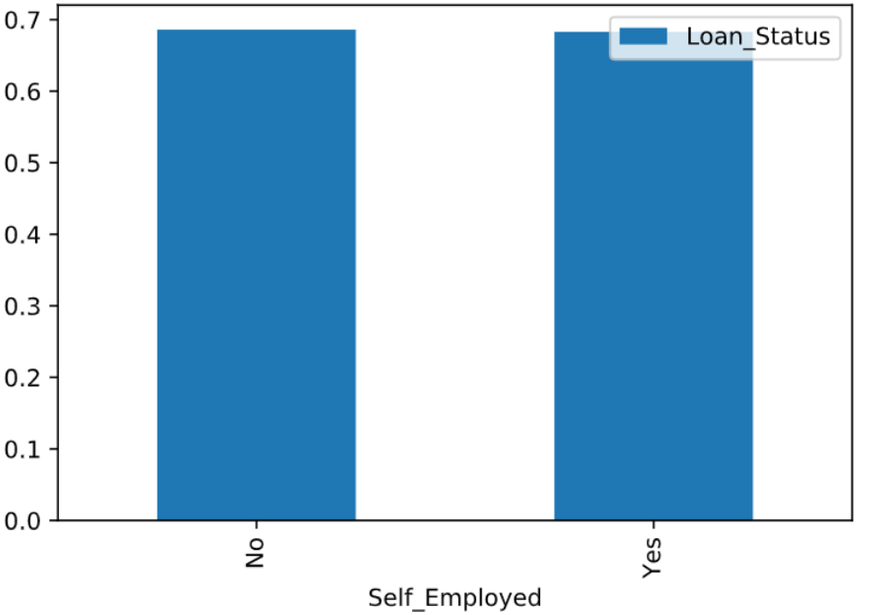
模型選擇與使用
- sklearn

結果分析與驗證
- metrics

練習

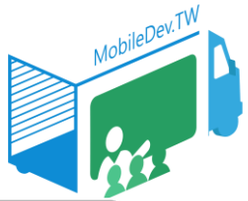
請用相同方式

觀察自雇者與借貸狀態是否有關連性



Ryan@MobileDev.TW

調閱資料 – 信用紀錄 VS. 借貸狀態



取得資料與探索
- pandas
- read_csv
- 資料觀察

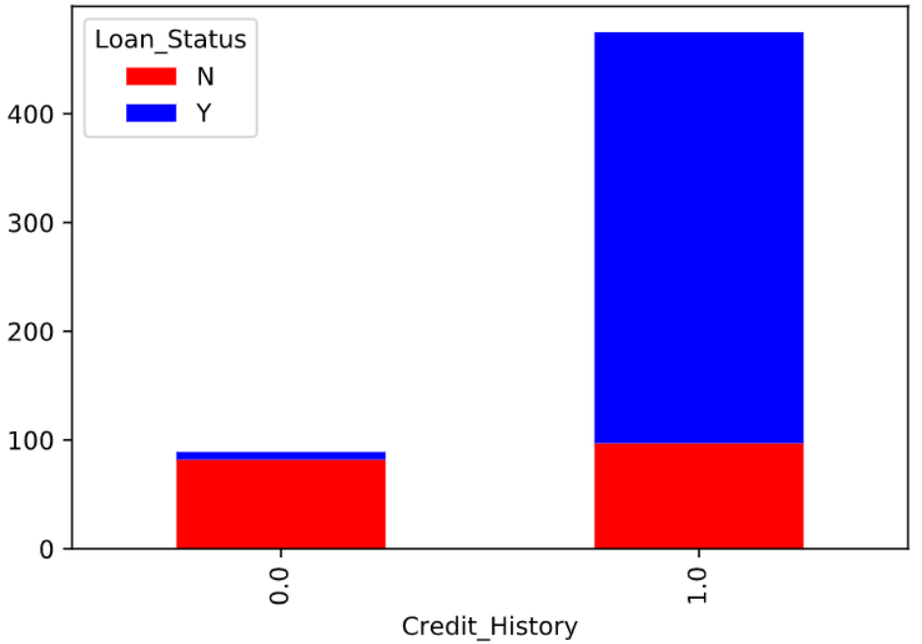
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

換一種呈現方式！

```
temp5 = pd.crosstab(df['Credit_History'],df
['Loan_Status'])
temp5.plot(kind='bar', stacked=True, color=
['red','blue'],grid=False)
```



Ryan@MobileDev.TW

調閱資料 – 信用紀錄/性別 VS. 借貸狀態

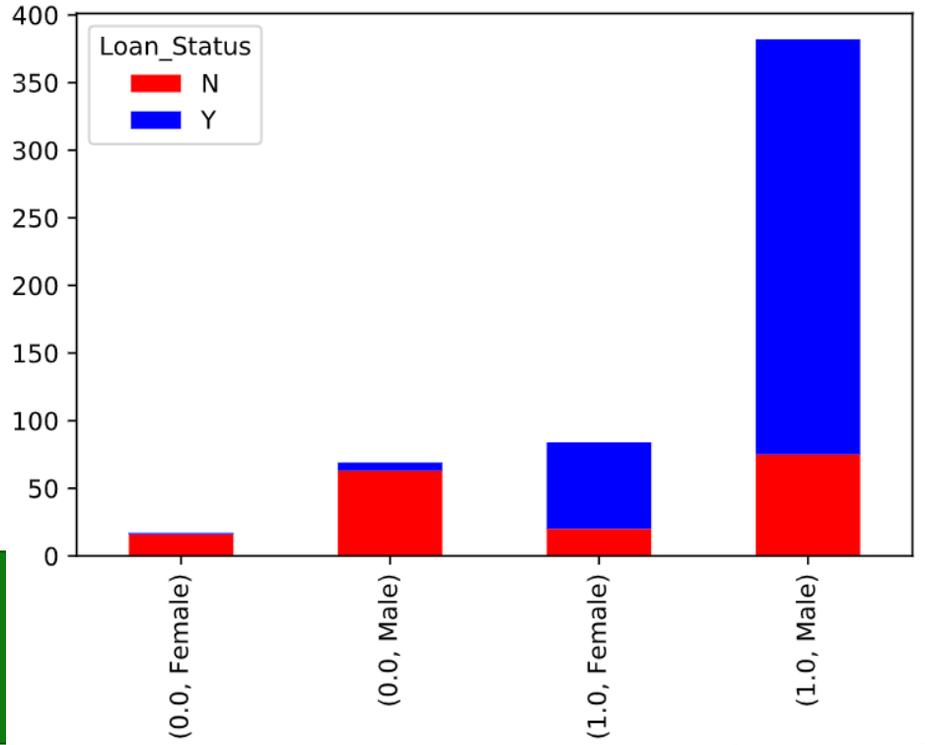
取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

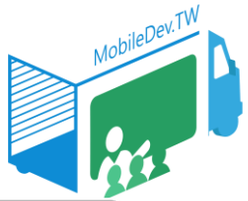
結果分析與驗證
- metrics

```
temp6 = pd.crosstab([df['Credit_History']
,df['Gender']],df['Loan_Status'])
temp6.plot(kind='bar',stacked=True, color
=['red','blue'])
```



男性&有信用紀錄的，借貸核准機會最大！
注意：有使用到性別欄位，前面若有刪除需重新載入數據

調閱資料 – 遺漏值綜覽



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

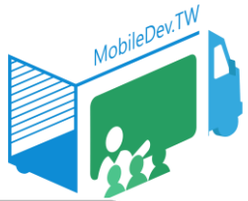
結果分析與驗證
- metrics

```
df.apply(lambda x:sum(x.isnull()),axis=0).sort_values(ascending=False)
```

遺漏值最多的是：

Ryan@MobileDev.TW

處理是否為自雇者的遺漏值 – 使用多數



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['Self_Employed'].value_counts()
print(500/(500+82))
```

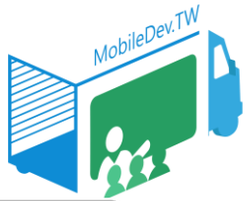
```
df['Self_Employed'].value_counts()

No      500
Yes      82
Name: Self_Employed, dtype: int64
```

```
print(500/(500+82))

0.859106529209621 9
```

非自雇者的比例為85.9 %

Ryan@MobileDev.TW

處理是否為自雇者的遺漏值 – 使用多數



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['Self_Employed'].value_counts()
print(500/(500+82))

df['Self_Employed'].fillna("No", inplace=True)
```

所以使用No來填補是否為自雇者的遺漏值

Ryan@MobileDev.TW

處理[借貸金額]的遺漏值常見方法 - 使用平均值

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

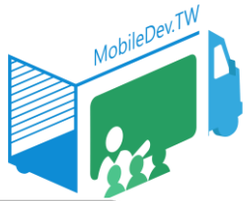
```
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
df.apply(lambda x: sum(x.isnull()),axis=0)

df['LoanAmount'].value_counts()
```

```
146.412162    22
120.000000    20
110.000000    17
100.000000    15
160.000000    12
              ..
570.000000     1
300.000000     1
376.000000     1
117.000000     1
311.000000     1
Name: LoanAmount, Length: 204, dtype: int64
```

是常見的做法，等等，還有沒有其他方式呢?

Ryan@MobileDev.TW

處理[借貸金額]的遺漏值 – 取得個別情況的中位數



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
table = df.pivot_table(values='LoanAmount',index='Self_Employed',columns='Education',aggfunc=np.median)
def fage(x):
    return table.loc[x['Self_Employed'],x['Education']]

df['LoanAmount'].fillna(df.apply(fage, axis=1),inplace=True)
```

依是否畢業、是否為自雇者分成四類，算出個別中位數
使用該中位數來填補
注意：需先確認要用到的Self_Employed、Education已無遺漏值
注意二：若有實作前一頁的方法(用平均值填補)記得先還原

Ryan@MobileDev.TW

借貸金額的觀察 – 取對數

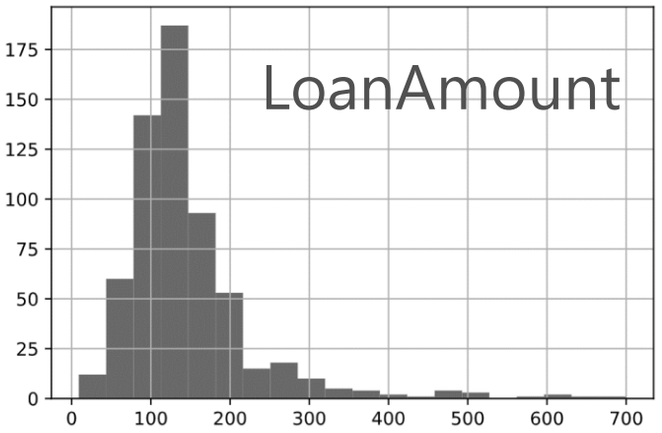
取得資料與探索
- pandas
- read_csv
- 資料觀察

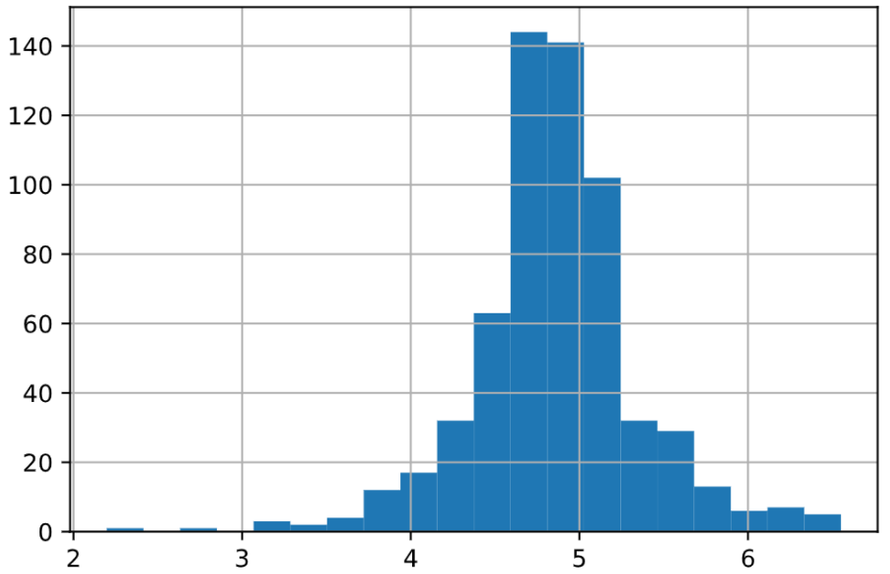
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['LoanAmount_log'] = np.log(df['LoanAmount'])
df['LoanAmount_log'].hist(bins=20)
```





透過對數的轉換來處理異常值，而非刪除

Ryan@MobileDev.TW

申請者本人收入 + 共同申請者收入

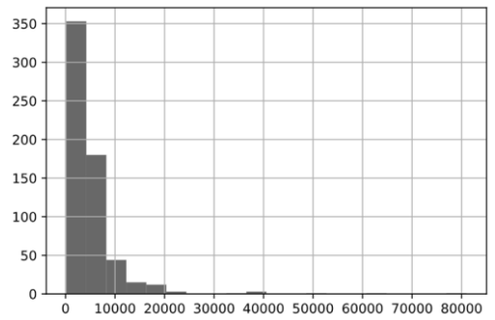
取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

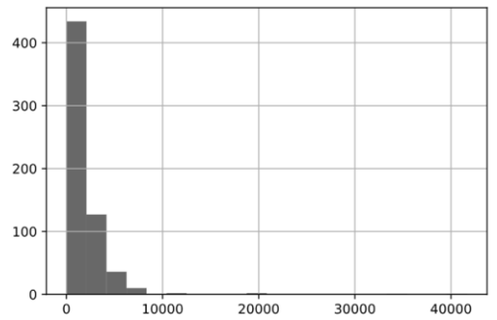
模型選擇與使用
- sklearn

結果分析與驗證
- metrics

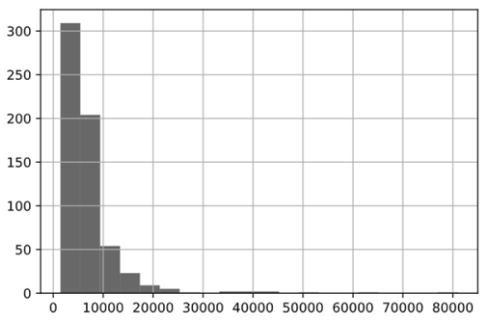
```
df['TotalIncome'] = df['ApplicantIncome']+df['CoapplicantIncome']
df['TotalIncome_log'] = np.log(df['TotalIncome'])
df['TotalIncome_log'].hist(bins=20)
```



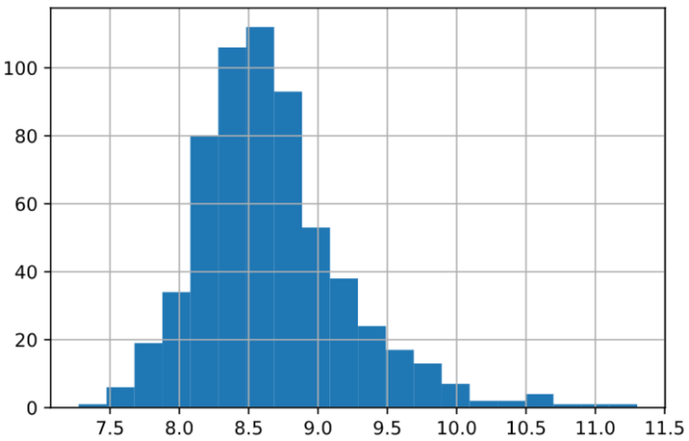
ApplicantIncome



CoapplicantIncome



TotalIncome

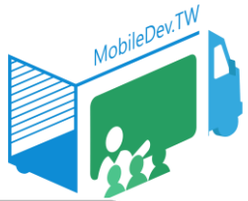


TotalIncome_log

透過對數的轉換來處理異常值，而非刪除

Ryan@MobileDev.TW

遺漏值填補：剩下的都用最高頻率值填補



取得資料與探索
- pandas
- read_csv
- 資料觀察

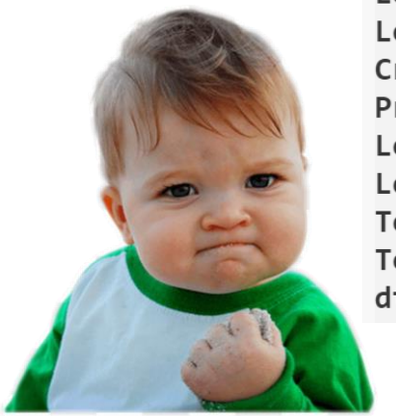
資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0],inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0],inplace=True)

df.apply(lambda x: sum(x.isnull()),axis=0)
```

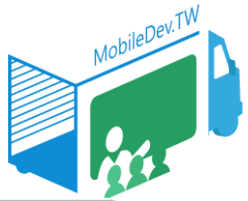
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
LoanAmount_log       0
TotalIncome          0
TotalIncome_log      0
dtype: int64
```



一個集合的mode就是最常出現的值，可能回傳多個所以取第0個

Ryan@MobileDev.TW

將非數值轉換為數值



取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
df.dtypes
from sklearn.preprocessing import LabelEncoder

var_mod = ['Gender','Married','Dependents','Education',
'Self_Employed','Property_Area','Loan_Status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

```
Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
LoanAmount_log      float64
TotalIncome         float64
TotalIncome_log     float64
dtype: object
```

```
Loan_ID              object
Gender                int32
Married               int32
Dependents            int32
Education             int32
Self_Employed         int32
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area         int32
Loan_Status           int32
LoanAmount_log      float64
TotalIncome         float64
TotalIncome_log     float64
dtype: object
```

fit_transform() : 回傳處理好的數值

Ryan@MobileDev.TW

匯入相關模組、建立套用模型的函數

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score

def loan_model(model, data, predictors, outcome, t_size, rs_number):
    X = data[predictors]
    y = data[outcome]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=t_size, random_state=rs_number)
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    recall = recall_score(y_test, predictions)
    precision = precision_score(y_test, predictions)
    print(f"Accuracy:{accuracy}")
    print(f"Recall:{recall}")
    print(f"Precision:{precision}")
```


Ryan@MobileDev.TW

使用LogisticRegression

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

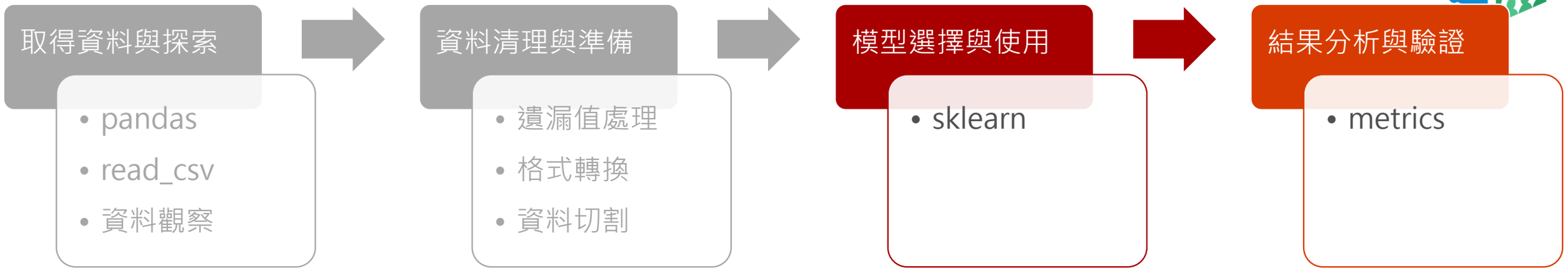
先只用信用紀錄來進行訓練

```
#1
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History']
loan_model(model, df, predictor_var, outcome_var, 0.3, 6)
```

```
Accuracy:0.8162162162162162
Recall:0.9921875
Precision:0.79375
```


Ryan@MobileDev.TW

使用DecisionTree試試



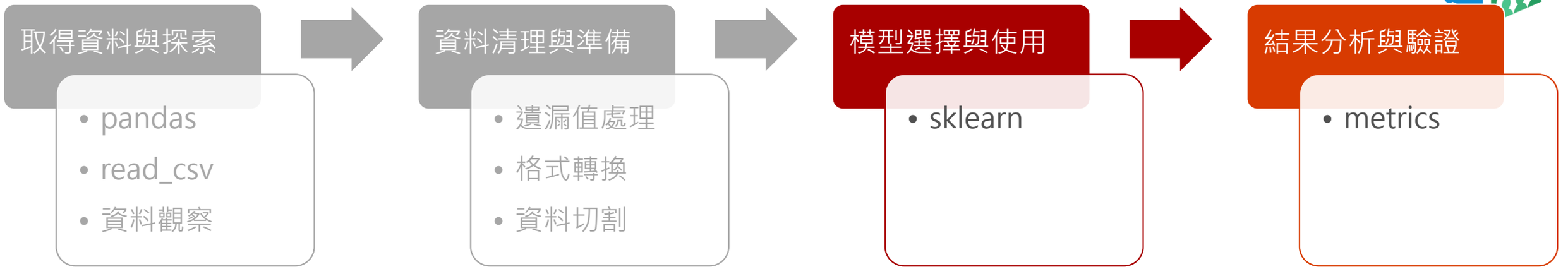

```
#2
outcome_var = 'Loan_Status'
model2 = DecisionTreeClassifier()
predictor_var = ['Credit_History']
loan_model(model2, df, predictor_var, outcome_var, 0.3, 6)
```

結果相同

```
Accuracy:0.8162162162162162
Recall:0.9921875
Precision:0.79375
```


Ryan@MobileDev.TW

多加幾個預測參數試試



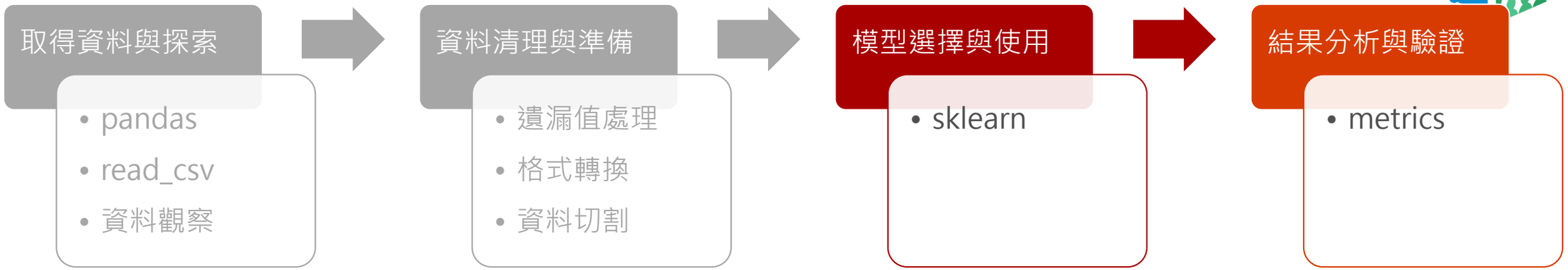
結果相同

```
#3
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History','Gender','Married','Education']
loan_model(model, df, predictor_var, outcome_var, 0.3, 6)
```

```
Accuracy:0.8162162162162162
Recall:0.9921875
Precision:0.79375
```


Ryan@MobileDev.TW

再換一個模型




```
#4
outcome_var = 'Loan_Status'
model3 = RandomForestClassifier(n_estimators=10)
predictor_var = ['Credit_History','Gender','Married','Education']
loan_model(model3, df, predictor_var, outcome_var, 0.3, 6)
```

結果相同

```
Accuracy:0.8162162162162162
Recall:0.9921875
Precision:0.79375
```


Ryan@MobileDev.TW

再多加幾個參數，包含調整過的參數

取得資料與探索
- pandas
- read_csv
- 資料觀察

資料清理與準備
- 遺漏值處理
- 格式轉換
- 資料切割

模型選擇與使用
- sklearn

結果分析與驗證
- metrics

```
#5
outcome_var = 'Loan_Status'
model3 = RandomForestClassifier(n_estimators=10)
predictor_var = ['Credit_History','Gender','Married','Education','Dependents','Self_Employed',
'Property_Area','LoanAmount_log','TotalIncome_log']
loan_model(model3, df, predictor_var, outcome_var, 0.3, 6)
```

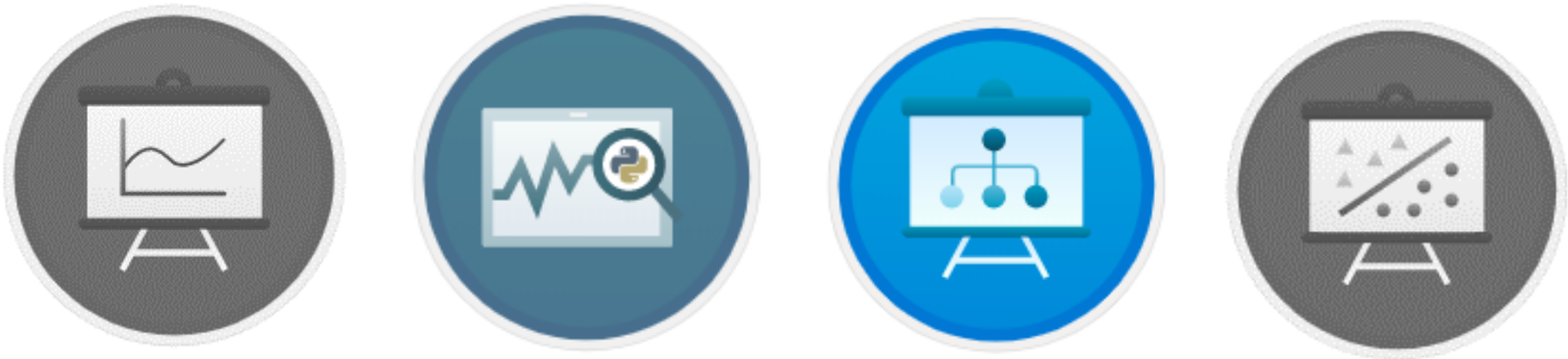
```
Accuracy:0.7567567567567568
Recall:0.8671875
Precision:0.7985611510791367
```

多不一定比較好!

Ryan@MobileDev.TW

小結

- 成熟的模型未必一定能帶來最佳成效，數據的篩選與轉換有時才是勝出的關鍵!
- 多了解各種模型的特性與使用時機，多多實驗，累積經驗
- 特徵工程(Feature Engineering)影響力高，讓數據更適合當前的模型!



Ryan@MobileDev.TW