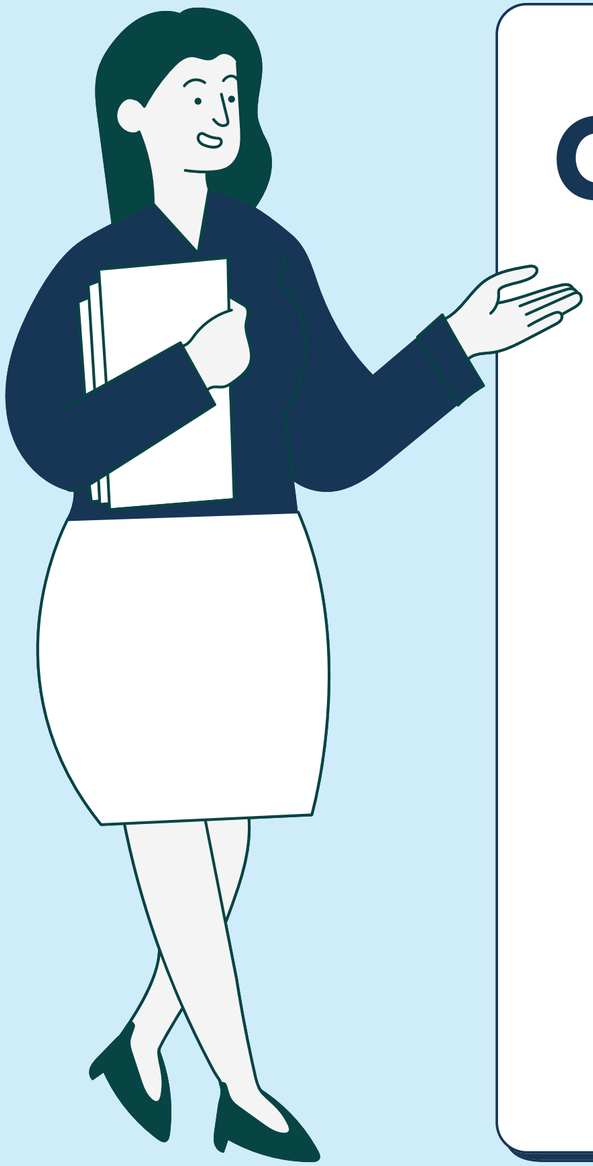
TEAM6

資料科學導論
競賽成果分享

H24064080 黃思媛
E44065020 何子安
H24041066 羅盼寧



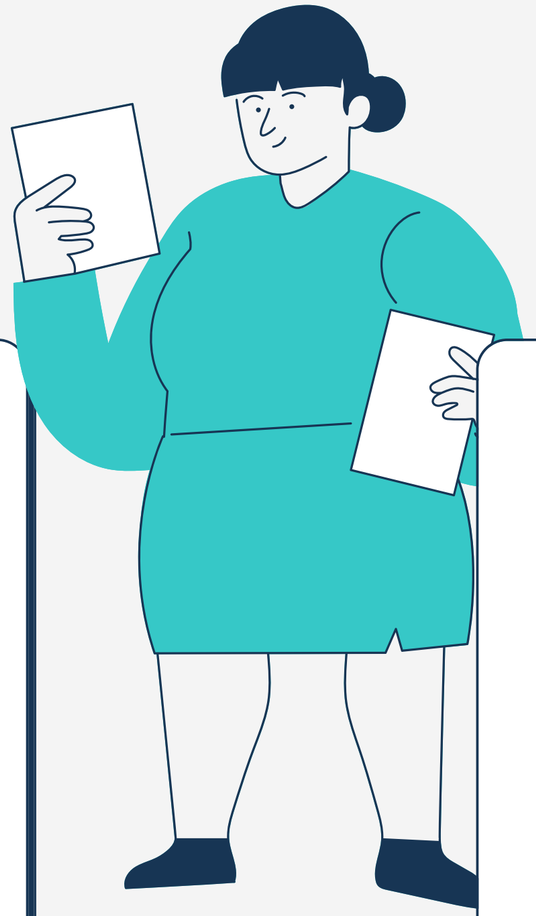
outline

資料前處理

嘗試過的模型方法

未來嘗試及改進
的方向

資料前處理



類別型資料

轉換為數字型態
one-hot encoding

切分資料

原訓練資料共 8000 筆
切分為 6000 筆訓練資料
和 2000 筆測試資料

特徵選擇

去除的特徵	採用的特徵	
Surname	CreditScore	Geography
RowNumber	Gender	Age
CustomerId	Tenure	Balance
	NumOfProducts	HasCrCard
	IsActiveMember	EstimatedSalary

嘗試過的模型方法

XGBoost

CatBoost

Random Forest

DNN

SVM ［最佳結果］

XGBoost & CatBoost

1

先選定一個參數後，透過嘗試此參數在不同數值下建立出的模型結果，
記錄下Accuracy Score, Precision Score, F1 score，
選擇最佳的final score作為調整此參數的依據。

2

將已經選定的參數固定，繼續嘗試調整下一個參數。

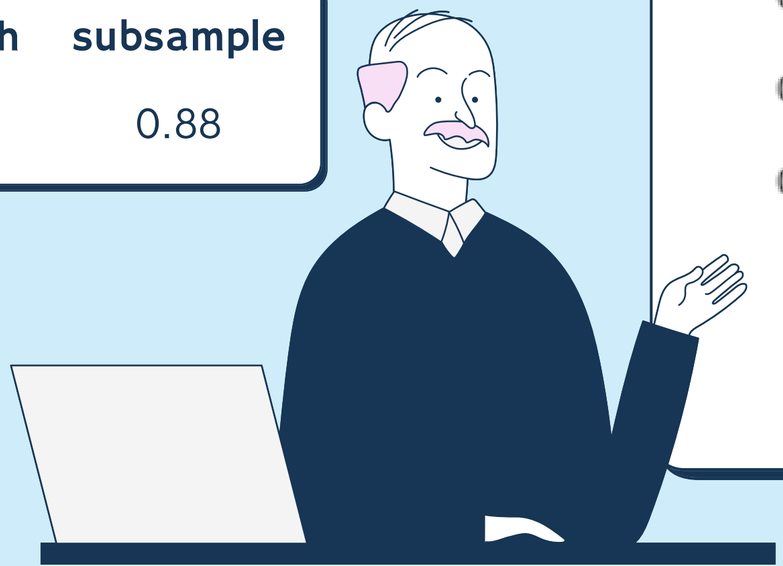
XGBoost & CatBoost

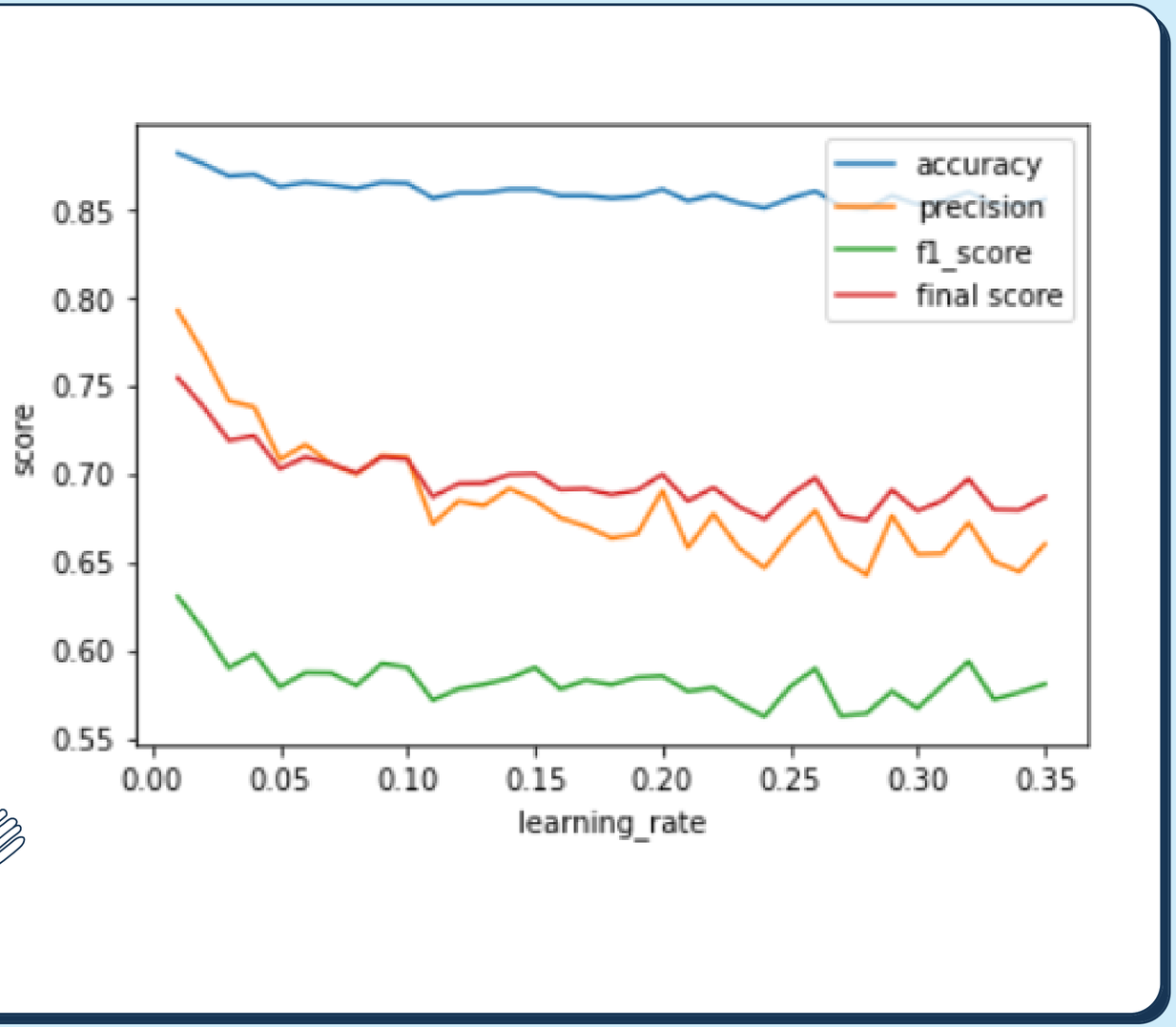
XGBoost

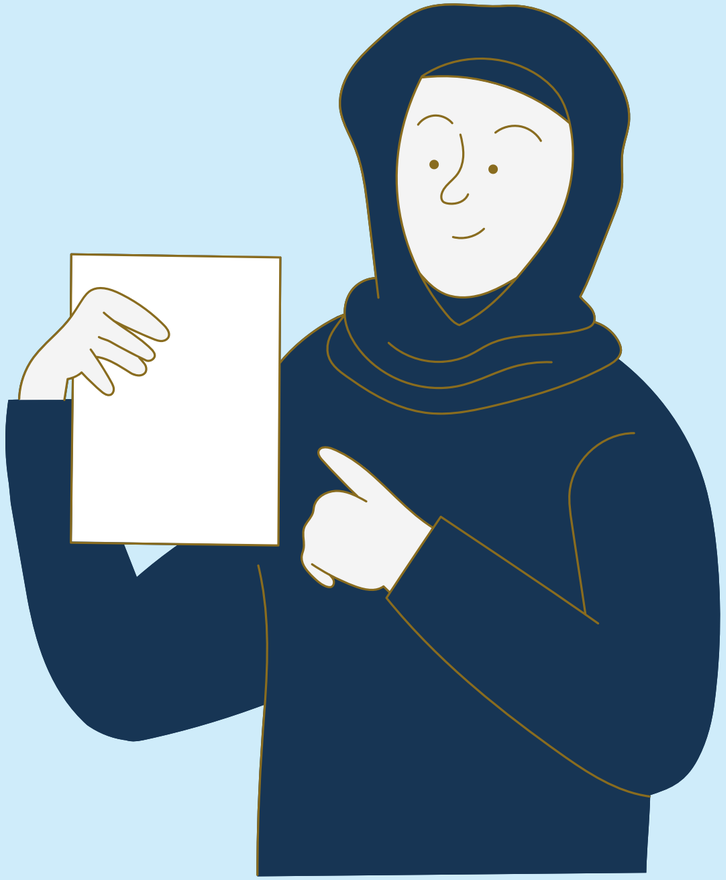
learning_rate	Max_depth	subsample	colsample_bytree
0.01	5	0.88	0.85

CatBoost

learning_rate	Max_depth	subsample
0.01	5	0.88







XGBoost

split train data	Accuracy	Precision	F1 score
Label Encoding	85.65%	68.40%	56.18%
未調整參數	85.75%	67.47%	57.75%
learning_rate調整後	87.30%	77.50%	59.42%
max_depth調整後	87.40%	78.15%	59.62%
subsample調整後	87.55%	78.19%	60.41%
colsample_bytree調整後	87.55%	79.65%	59.64%

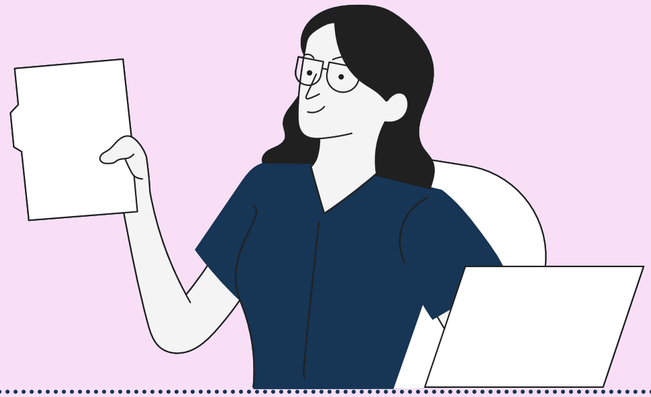
upload	Accuracy	Precision	F1 score
Label Encoding	0.86	0.6615	0.6056
未調整參數	0.8725	0.7407	0.6107
調整參數後	0.88	0.7959	0.6195

CatBoost

split train data	Accuracy	Precision	F1 score
Label Encoding	87.35%	75.29%	61.02%
未調整參數	87.45%	76.06%	61.29%
learning_rate調整後	88.15%	79.22%	63.03%
max_depth調整後	88.15%	79.22%	63.03%
subsample調整後	88.15%	79.22%	63.03%

upload	Accuracy	Precision	F1 score
Label Encoding	0.87	0.6984	0.6286
調整參數後	0.87	0.7119	0.6176

Random Forest



1

一次只設定一種參數，並根據設定之的參數值畫出對應的評估指標值(包含accuracy、precision、f-score)，最後使用與網站排行相同計算方式之加權分數找出最佳的參數設定值。

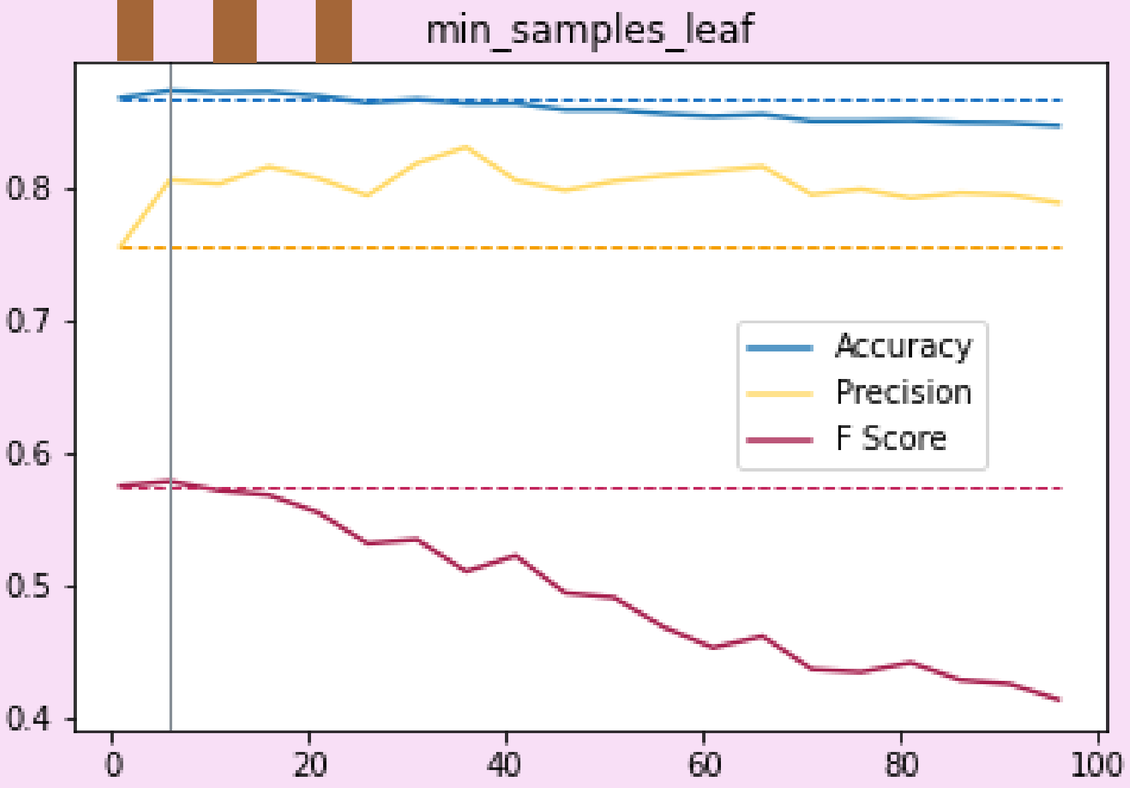
若該模型之加權分數與基本模型之加權分數相比，分數提升較少，代表該參數對於提升模型效果無明顯影響，則後續的探討不列入使用。

2

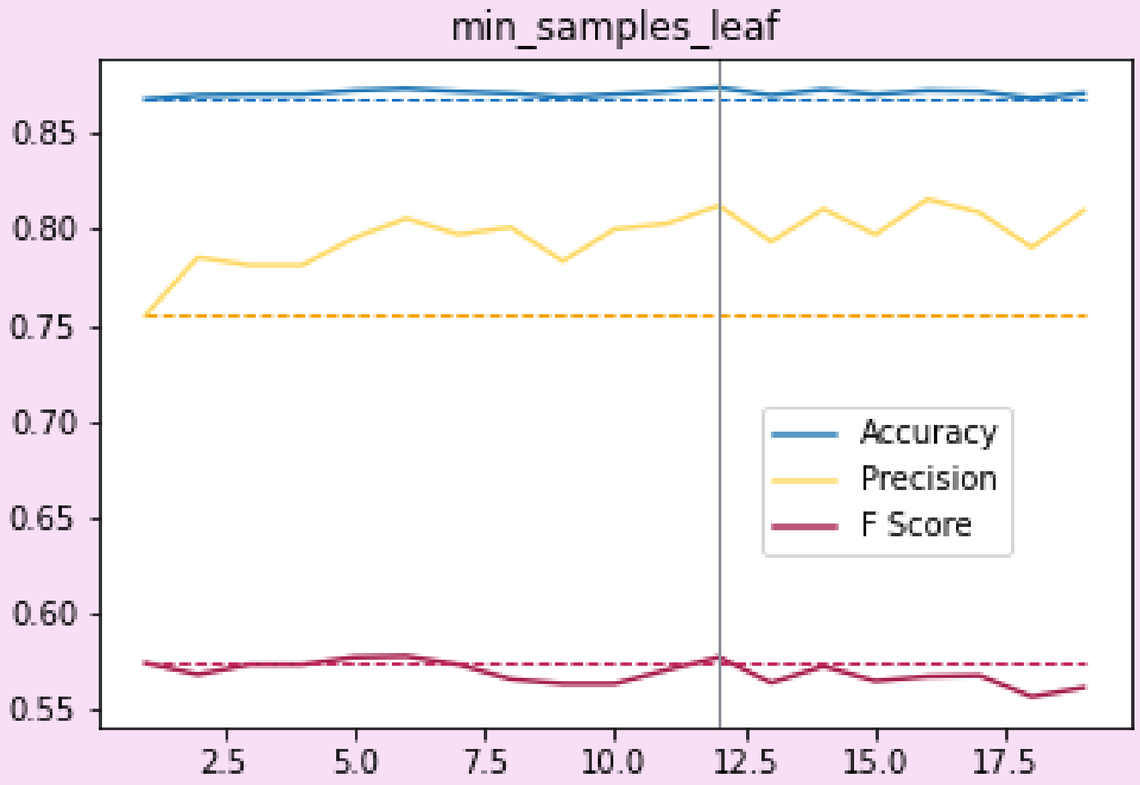
比較使用多項參數的模型之間的表現結果，找出最佳模型。

step 1

Random Forest



range(1, 100, 5)



range(1, 20)

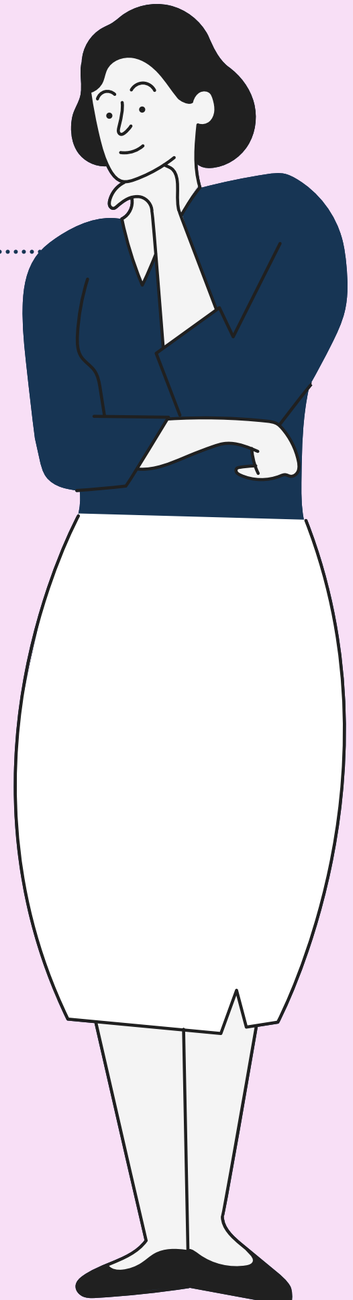
Random Forest

step 1
result

drop parameter： n_jobs、criterion

	parameter	value	model score	induced
0	n_estimators	165	0.7328	0.0161
1	max_depth	12	0.7383	0.0216
2	min_samples_split	17	0.7420	0.0253
3	min_samples_leaf	12	0.7368	0.0201
4	max_features	2	0.7271	0.0104
5	class_weight	{0: 6, 1: 1}	0.7215	0.0048
6	max_samples	628	0.7413	0.0246

可以看出調整
”min_samples_split”
時有最好的表現



step 2
result

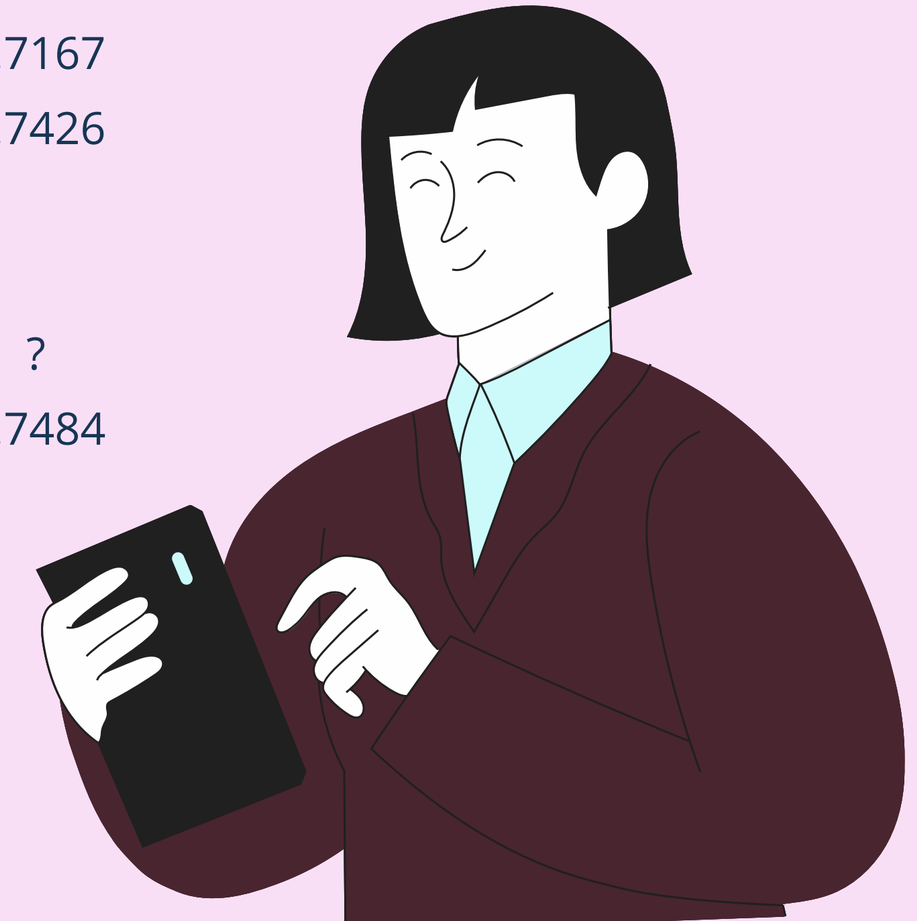
Random Forest

5 : 使用所有第二階段被調整過的參數

	model	acc	precision	f_score	final_score
0	base	0.8675	0.755274	0.5746	0.716672
1	RandomForestClassifier(max_depth=12, n_estimators=165, random_state=6)	0.8735	0.806500	0.5804	0.736200
2	RandomForestClassifier(max_depth=12, min_samples_split=17, n_estimators=165,\n random_state=6)	0.8745	0.802700	0.5878	0.738300
3	RandomForestClassifier(max_depth=12, min_samples_leaf=12, min_samples_split=17,\n n_estimators=165, random_state=6)	0.8705	0.792600	0.5705	0.727100
5	RandomForestClassifier(max_depth=12, max_features=2, max_samples=628,\n min_samples_leaf=12, min_samples_split=17,\n n_estimators=165, random_state=6)	0.8395	0.842100	0.3326	0.637500
4	RandomForestClassifier(max_depth=12, max_features=2, min_samples_leaf=12,\n min_samples_split=17, n_estimators=165, random_state=6)	0.8680	0.814400	0.5448	0.722600
6	RandomForestClassifier(max_depth=12, min_samples_split=17, random_state=6)	0.8710	0.793600	0.5728	0.728500
7	RandomForestClassifier(min_samples_split=17, n_estimators=165, random_state=6)	0.8745	0.805400	0.5865	0.738600
10	RandomForestClassifier(max_samples=628, min_samples_split=17, random_state=6)	0.8680	0.811200	0.5464	0.722300
11	RandomForestClassifier(max_features=2, min_samples_leaf=12, random_state=6)	0.8695	0.827200	0.5477	0.728100
12	RandomForestClassifier(max_features=2, max_samples=628, min_samples_leaf=12,\n random_state=6)	0.8440	0.836400	0.3710	0.652500
13	RandomForestClassifier(max_samples=628, n_estimators=165, random_state=6)	0.8730	0.814300	0.5738	0.735700
14	RandomForestClassifier(max_features=2, max_samples=628, random_state=6)	0.8640	0.813200	0.5211	0.711600
15	RandomForestClassifier(max_features=2, max_samples=628, n_estimators=165,\n random_state=6)	0.8665	0.821600	0.5324	0.719400
16	RandomForestClassifier(min_samples_split=17, random_state=6)	0.8760	0.808000	0.5934	0.742600

Random Forest

	Accuracy	Precision	F1 score	Final score
split train data				
base model	0.8675	0.7553	0.5746	0.7167
final model	0.8760	0.808	0.5934	0.7426
upload				
base model	0.8525	0.6607	?	?
final model	0.88	0.7736	0.6308	0.7484



DNN資料前處理

訓練模型的特徵對其做標準化（standardization）

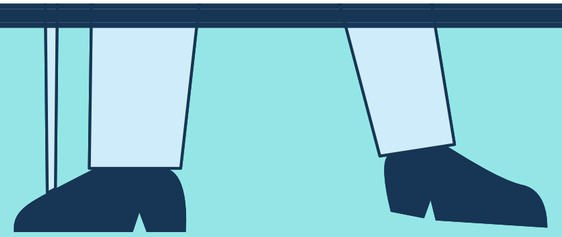
對 HasCrCard 與 IsActiveMember 的值從(0,1)換為(-1,1)

對 label(0,1)做 one-hot encoding 為讓模型輸出兩個值

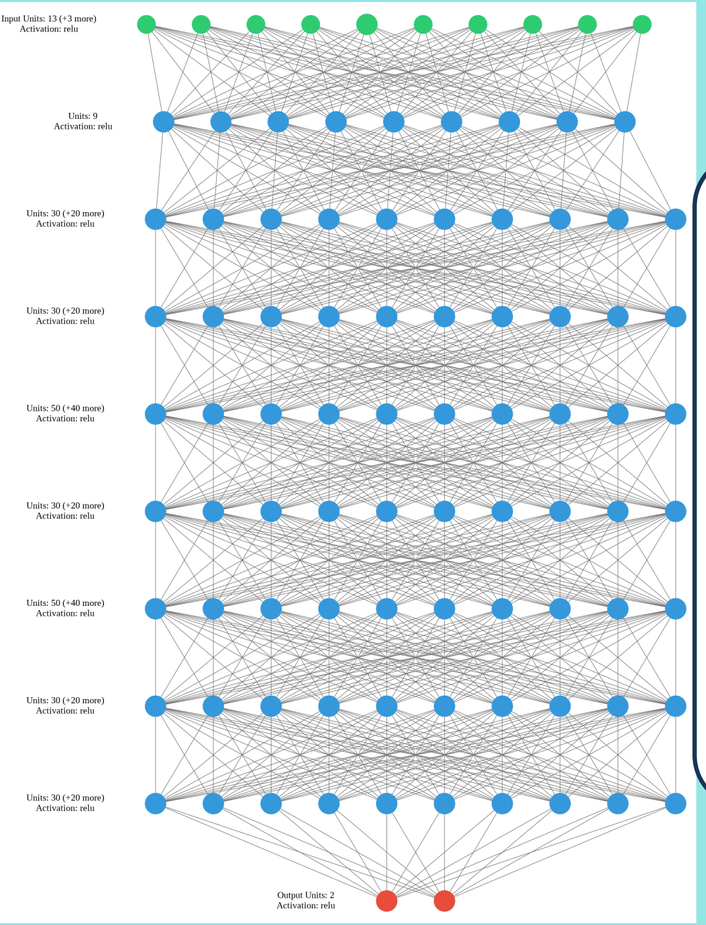
流程中使用 Synthetic Minority Oversampling Technique (SMOTE) 來處理資料不平衡的方式

DNN

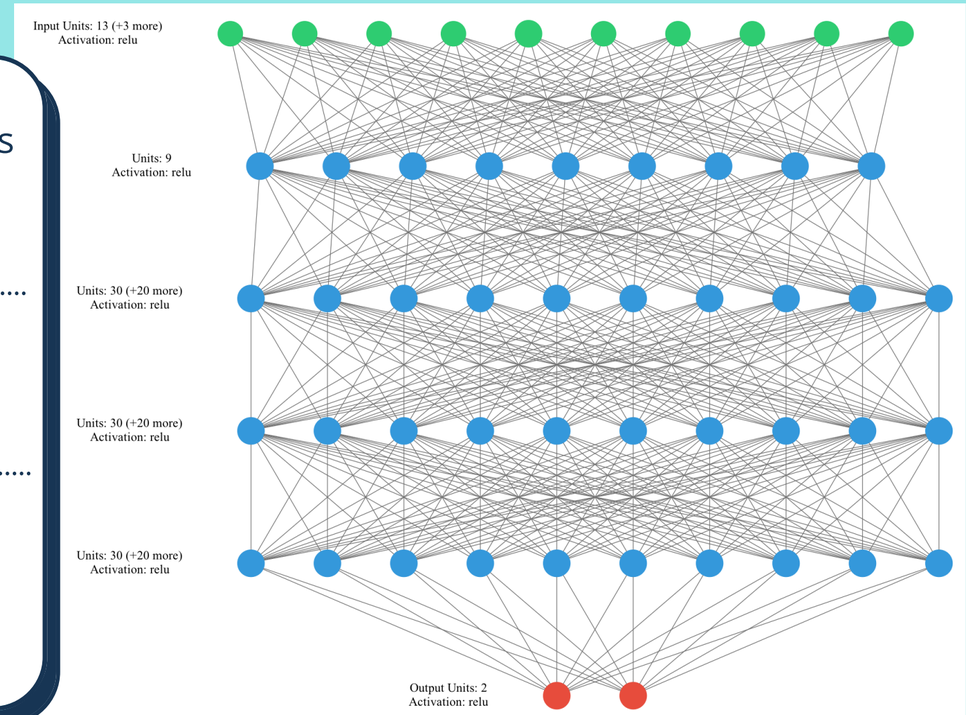
early stopping	settings	metrics	settings	compile	settings
monitor	val_accuracy	**accuracy**	none	**loss**	binary_crossentropy
min delta	0.01	**precision**	thresholds=0.7	**optimizer**	rmsprop
patience	300	**f1-score**	num_classes=2	**metrics**	accuracy, precision, f1-score
verbose	1				
model	max				

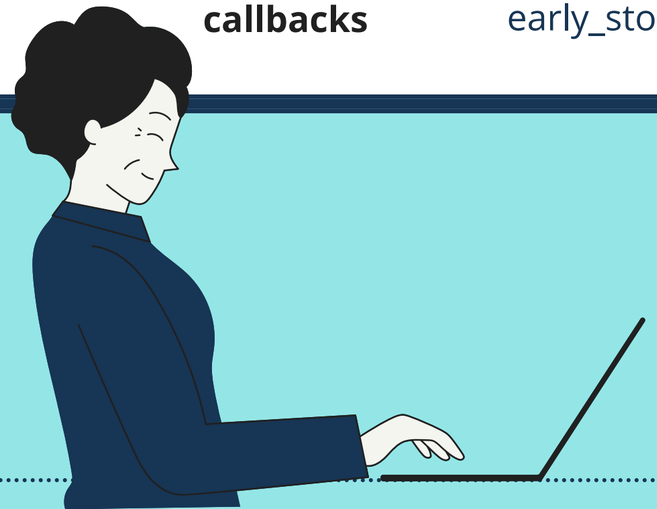


DNN



neurons	**input layers**	dimension of features
	hidden layers	4 layers (30 each)
	output layers	2
activations	**input layers**	relu
	hidden layers	relu
	output layers	sigmoid
fit	**epochs**	1000
	batch_size	50
	callbacks	early_stopping





DNN

- **調整隱藏層數量比調整神經元數量會更能影響DNN的表現**
  - https://case.ntu.edu.tw/blog/?p=26340
  - https://zhuanlan.zhihu.com/p/100419971
- **小batch size對DNN的表現會更勝於大batch size**
  - https://youtu.be/zzbr1h9sF54
  - https://arxiv.org/abs/1609.04836

SVM

before one-hot encoding	**after one-hot encoding**
對特徵項做標準化	**對特徵項做標準化**
kernel = {rbf, linear, poly, sigmoid, precomputed}	**kernel = rbf**
training set = {6000, 8000}	**training set = 6000**

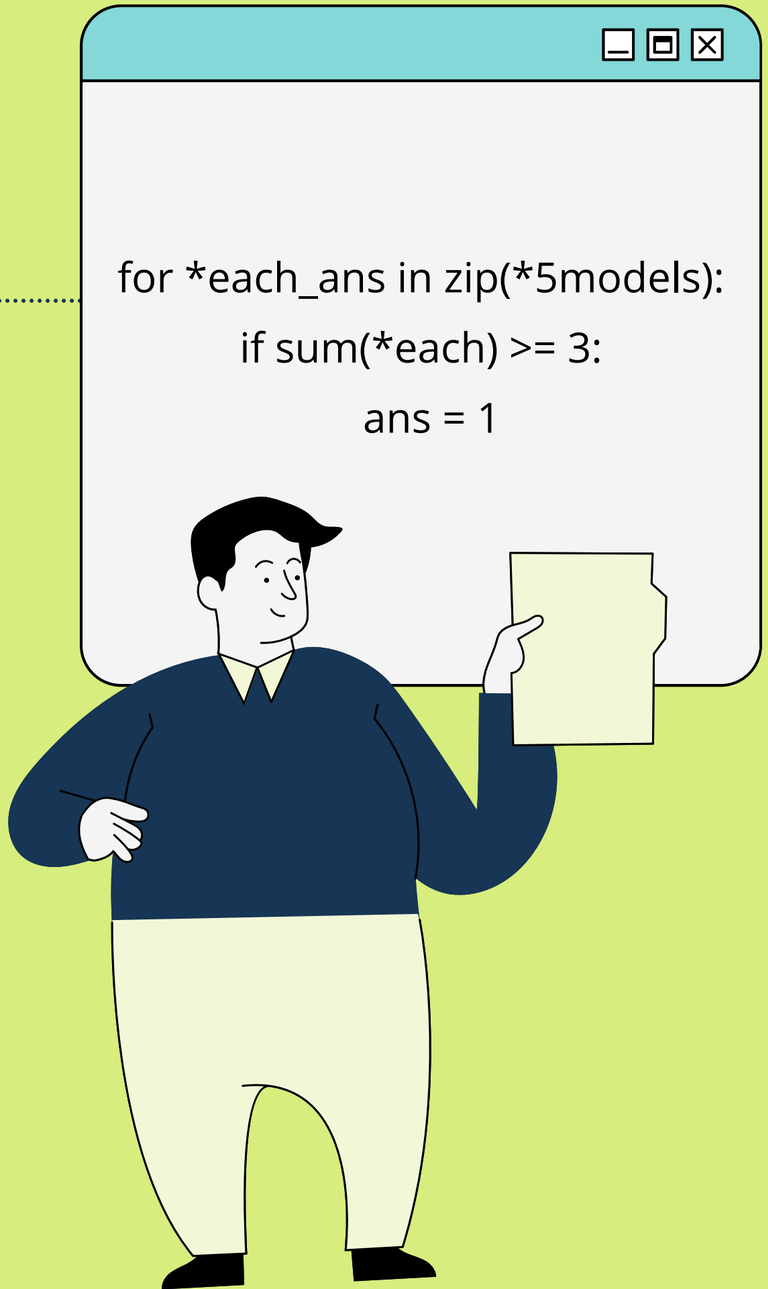
推測為在切分訓練與測試資料集時所使用的 random_state 參數剛好能將資料切分為對未知資料所能得到最好表現的分佈

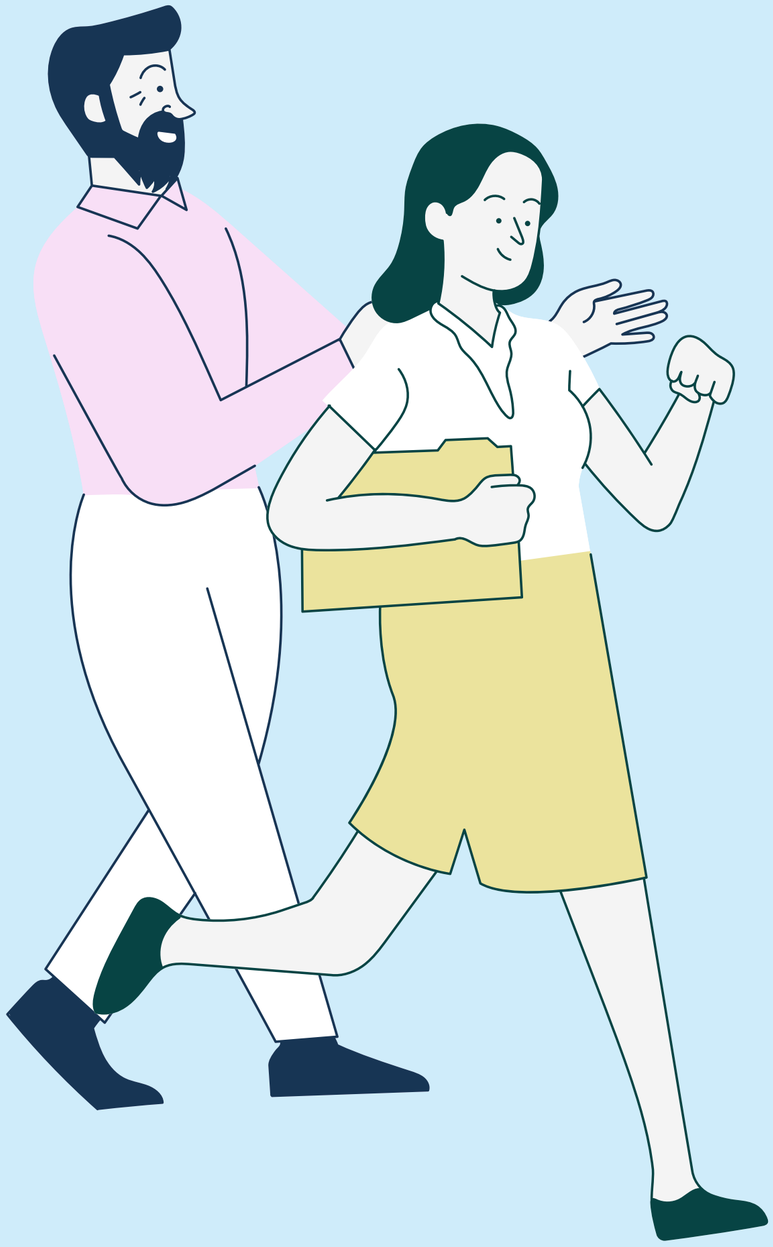
Vote

1 取所有模型所預測出來的值

2 以模型所預測出來的結果之比例
來決定新一份預測檔案的值

但這樣的作法並沒有「預測」出更好的結果





未來嘗試及改進的方向

感謝聆聽！

大家都辛苦ㄌ，祝大家有一個健康的期末 ><