2020

智慧化企業整合期末報告

# 學生成績預測模型

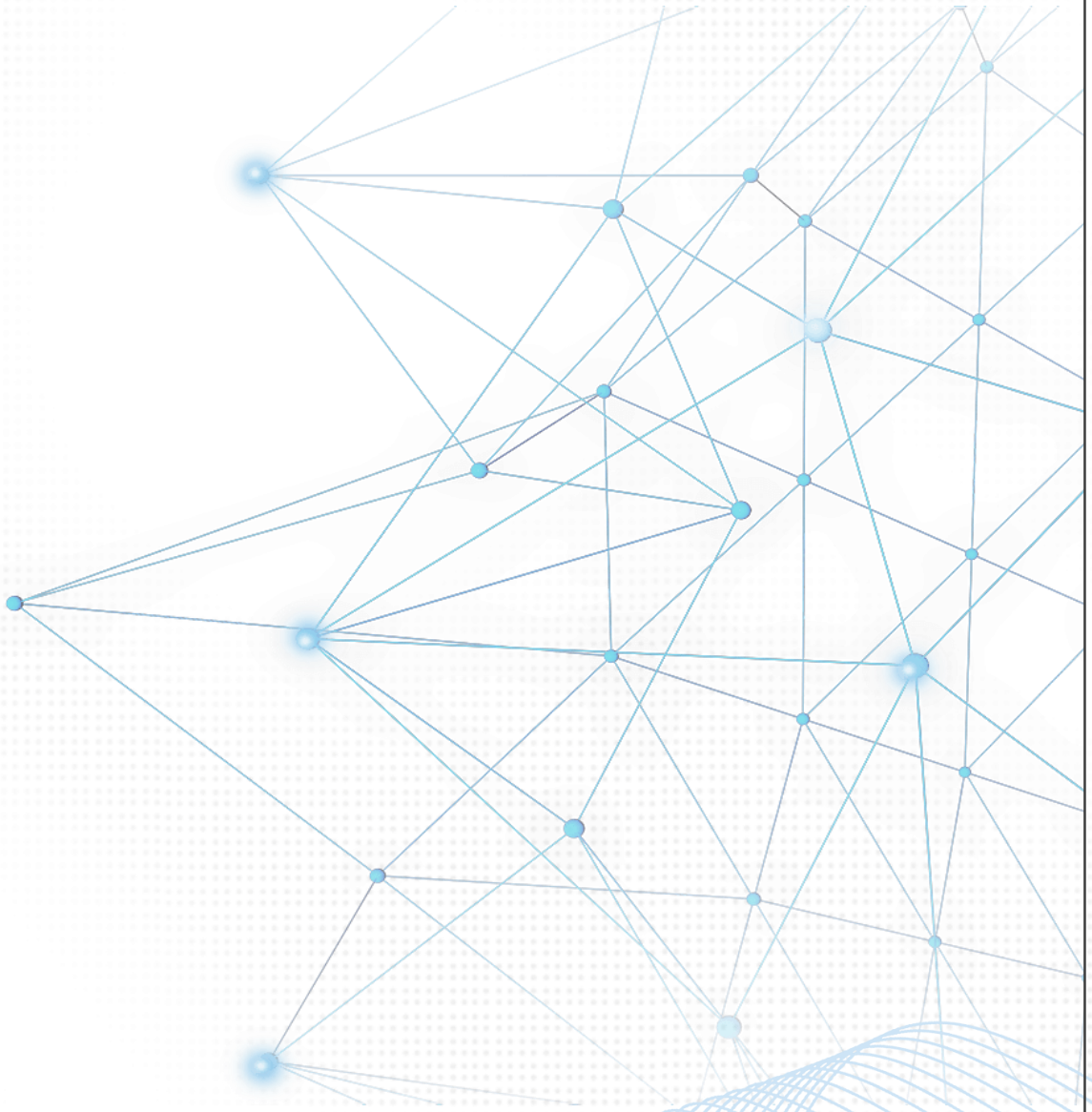
朱文伃

# 目錄

PART ONE

# Introduction

01



# 1 Introduction

## Where

台灣各地高級中學

## What

雖然教育程度已大幅提升
但學期成績不及格人數還
是佔有一定的比例



## Who

針對台灣高中生

## When

在期中、期末時進行預
測，提前給予學生協助

## How

藉由學生成績預測模型找出
影響學生表現的關鍵因素，
提前給予學生協助

# 2 Introduction

## 資料集

| 分數 | 16-20 | 14-15 | 12-13 | 10-11 | 0-9 |
|---|---|---|---|---|---|
| 等級 | A(1) | B(2) | C(3) | D(4) | F(5) |

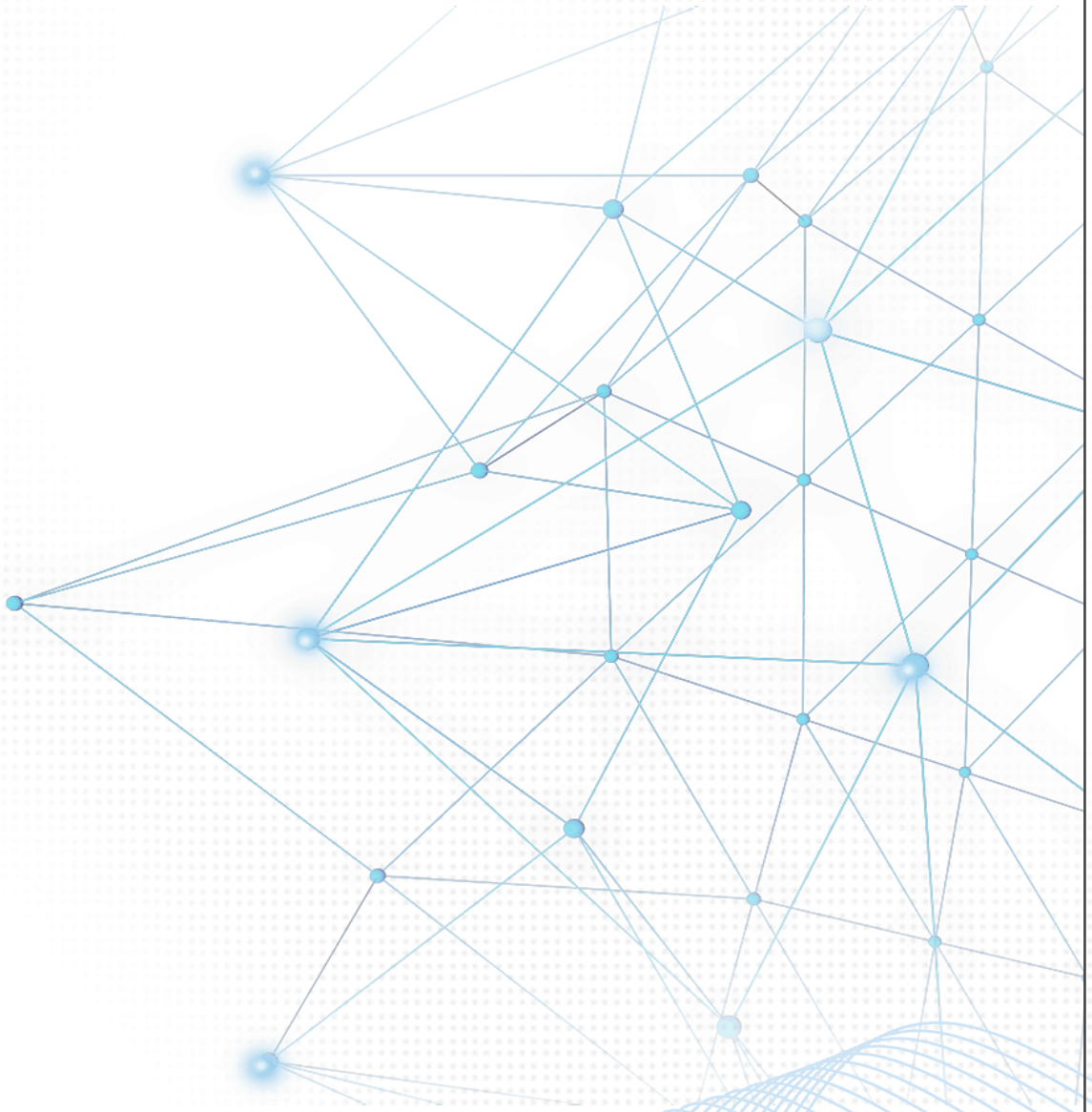
- 共1044筆資料
- 29個特徵預測數學成績(包含家中是否有網路、父母職業、課後娛樂時間等)
- 成績分布0-20分，將其分為A-F五個等級

| traveltime | studytime | failures | schoolsup | famsup | paid | activities | nursery | higher | internet | romantic | famrel | freetime | goout | Dalc | Walc | health | absences | score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 0 | yes | no | no | no | yes | yes | no | no | 4 | 3 | 4 | 1 | 1 | 3 | 6 | 5 |
| 1 | 2 | 0 | no | yes | no | no | no | yes | yes | no | 5 | 3 | 3 | 1 | 1 | 3 | 4 | 5 |
| 1 | 2 | 3 | yes | no | yes | no | yes | yes | yes | no | 4 | 3 | 2 | 2 | 3 | 3 | 10 | 7 |
| 1 | 3 | 0 | no | yes | yes | yes | yes | yes | yes | yes | 3 | 2 | 2 | 1 | 1 | 5 | 2 | 15 |
| 1 | 2 | 0 | no | yes | yes | no | yes | yes | no | no | 4 | 3 | 2 | 1 | 2 | 5 | 4 | 6 |
| 1 | 2 | 0 | no | yes | yes | yes | yes | yes | yes | no | 5 | 4 | 2 | 1 | 2 | 5 | 10 | 15 |
| 1 | 2 | 0 | no | no | no | no | yes | yes | yes | no | 4 | 4 | 4 | 1 | 1 | 3 | 0 | 12 |
| 2 | 2 | 0 | yes | yes | no | no | yes | yes | no | no | 4 | 1 | 4 | 1 | 1 | 1 | 6 | 6 |
| 1 | 2 | 0 | no | yes | yes | no | yes | yes | yes | no | 4 | 2 | 2 | 1 | 1 | 1 | 0 | 16 |

PART TWO

# 資料前處理

02



# 1 特徵值編碼

使用標籤編碼器(LabelEncoder)將資料集統一轉為數值

使預測模型能更好地理解數據，以便模型訓練和預測進行

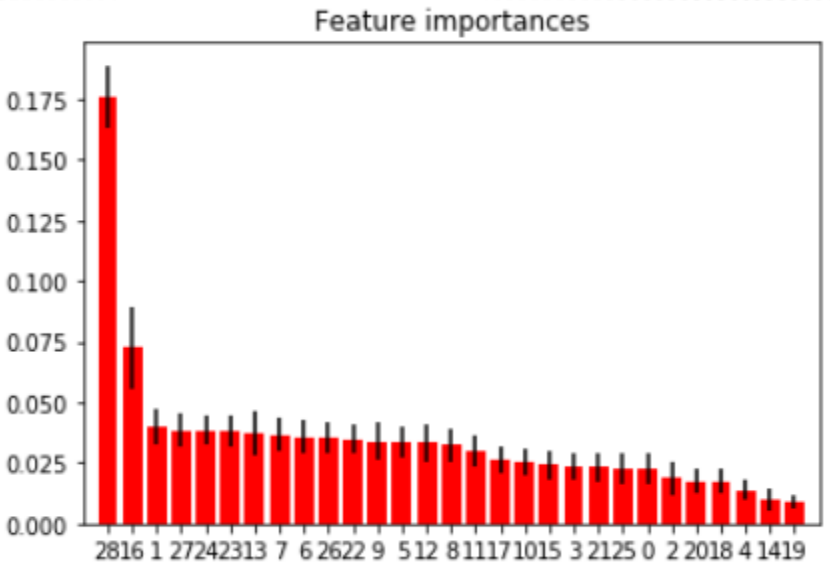
```
df = pd.read_csv("student_initial.csv")
df.columns = ['sex','age','address','famsize',  'Pstatus','Medu','Fedu','Mjob','Fjob','rea
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['sex']= labelencoder.fit_transform(df['sex'])
df['address']= labelencoder.fit_transform(df['address'])
df['famsize']= labelencoder.fit_transform(df['famsize'])
```

| | sex | age | address | famsize | Pstatus | ... | Dalc | Walc | health | absences |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 18 | 1 | 0 | 0 | ... | 1 | 1 | 3 | 6 |
| 1 | 0 | 17 | 1 | 0 | 1 | ... | 1 | 1 | 3 | 4 |
| 2 | 0 | 15 | 1 | 1 | 1 | ... | 2 | 3 | 3 | 10 |
| 3 | 0 | 15 | 1 | 0 | 1 | ... | 1 | 1 | 5 | 2 |
| 4 | 0 | 16 | 1 | 0 | 1 | ... | 1 | 2 | 5 | 4 |

# 2 特徵選取

目的：避免過擬合和維度災難的問題

以Extra Trees選出重要分數>0.35者
→包含出席率、和同儕相處狀況、父親教育程度等十項



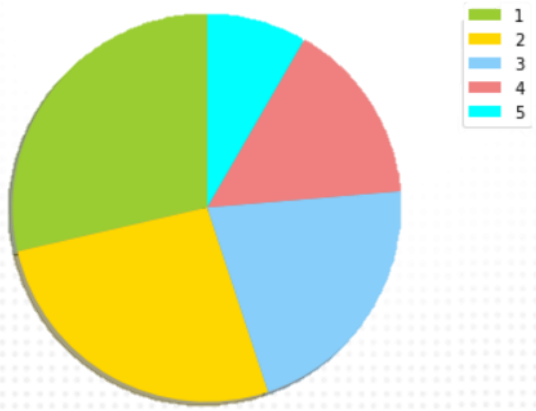

```
Feature ranking:
1. feature 28 (0.176227)
2. feature 16 (0.072523)
3. feature 1 (0.039866)
4. feature 27 (0.038395)
5. feature 24 (0.038381)
6. feature 23 (0.037975)
7. feature 13 (0.036982)
8. feature 7 (0.036505)
9. feature 6 (0.035764)
10. feature 26 (0.035726)
11. feature 22 (0.034753)
12. feature 9 (0.033956)
13. feature 5 (0.033498)
14. feature 12 (0.033401)
15. feature 8 (0.032396)
16. feature 11 (0.029522)
17. feature 17 (0.026006)
```

# 3 過取樣

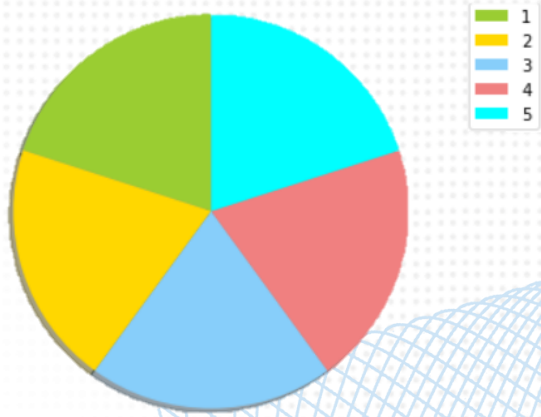
原資料集中五個類別樣本數不平衡

將20%資料當作測試集，80%當作訓練集並以SMOTE法進行過取樣

```
Before OverSampling, counts of label '1': [71]
Before OverSampling, counts of label '2': [134]
Before OverSampling, counts of label '3': [183]
Before OverSampling, counts of label '4': [213]
Before OverSampling, counts of label '5': [234]
```



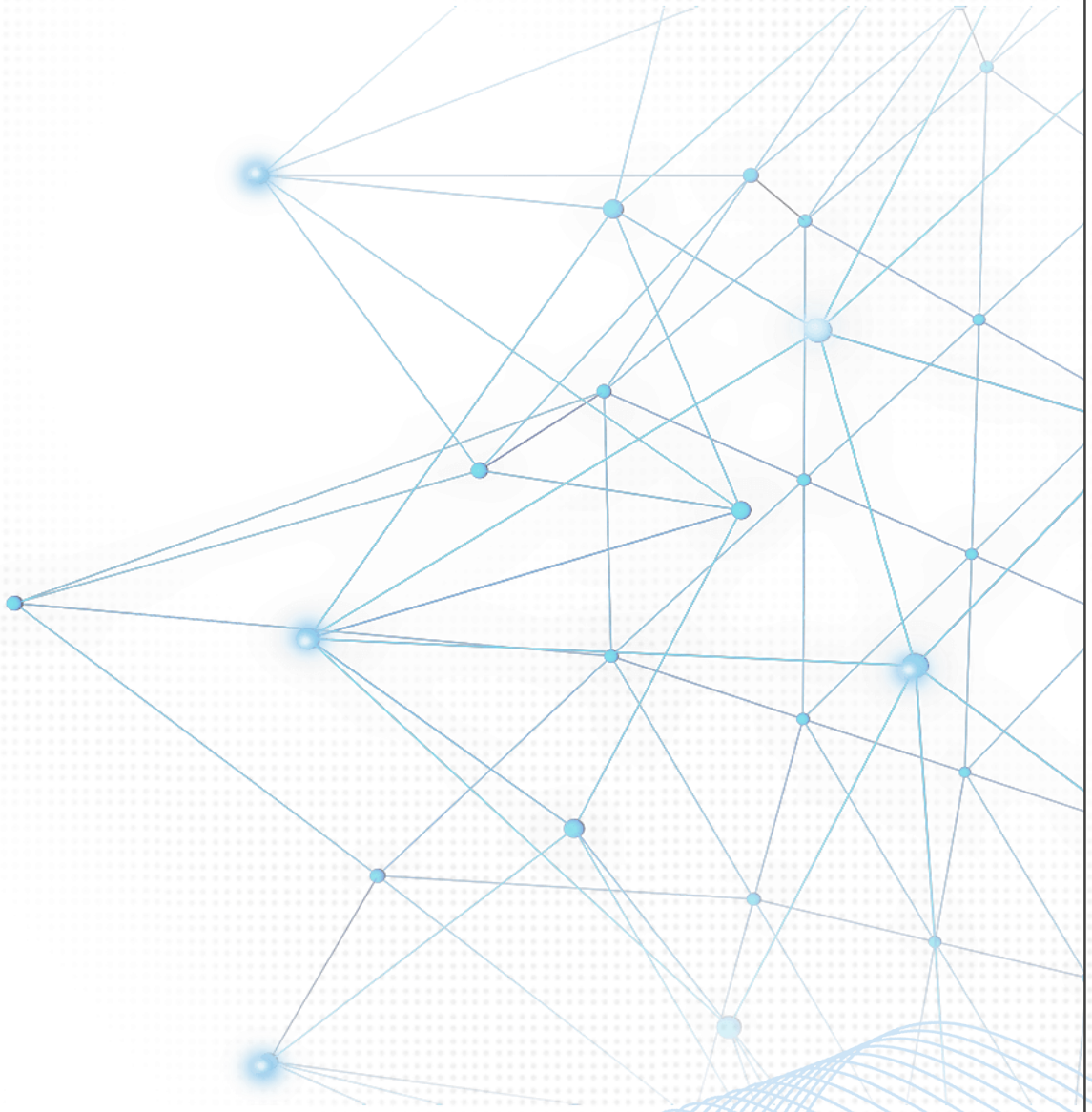

```
After OverSampling, counts of label '1': 234
After OverSampling, counts of label '2': 234
After OverSampling, counts of label '3': 234
After OverSampling, counts of label '4': 234
After OverSampling, counts of label '5': 234
```



PART THREE

# 預測模型

03



# 應用步驟

01

## 特徵選取

Extra Trees

02

## 過取樣

以SMOTE合成類別較少的資料

03

## 建立模型

Naïve Bayes、SVM、MLP、DNN

04

## 評估模型表現

比較各模型準確率、正規化與未正規化之結果

# 1 Naïve Bayes模型

→假設每個特徵互相獨立，以貝氏定理維基礎依據特徵計算出機率最大的分類

將資料集拆成訓練集與測試集

引入GuassianNB開始訓練

```
from sklearn.model_selection import train_test_split
X = np.array(data.ix[:, data.columns != 'score'])  #不是score的欄位
y = np.array(data.ix[:, data.columns == 'score'])  #score
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=0) #

#建立模型並且訓練
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train_res, y_train_res)

#預測結果
y_pred = model.predict(X_test)

#顯示模型準確率
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

| | 準確率 |
|---|---|
| 未正規化 | 0.258 |
| 正規化 | 0.254 |

# 2 SVM模型

→找出一個超平面(hyperplane)，使之將兩個不同的集合分開

設定多組參數尋找最佳參數

Kernel以RBF、linear做比較

引入SVC開始訓練

```
# 設定多種參數
params_grid = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                    'C': [1, 10, 100, 1000]},
                   {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
from sklearn.svm import SVC
svm_model = GridSearchCV(SVC(), params_grid, cv=5)
svm_model.fit(X_train,y_train)
# 顯示最佳的參數為何
print('Best C:',svm_model.best_estimator_.C,"\n")
print('Best Kernel:',svm_model.best_estimator_.kernel,"\n")
print('Best Gamma:',svm_model.best_estimator_.gamma,"\n")

final_model = svm_model.best_estimator_
y_pred = final_model.predict(X_test)

print(classification_report(y_test,y_pred))
```

| | 準確率 |
|---|---|
| 未正規化 | 0.374 |
| 正規化 | 0.387 |

# 3 MLP模型

→是一種前向傳遞的類神經網路

設定2層隱藏層，並在使準確率提升、損失函數下降的情況下，找出適當神經元個數

激活函數為ReLU，優化方法為Adam

```
clf = MLPClassifier(hidden_layer_sizes=(21,50), max_iter=1000,activation = 'relu',solver='adam',random_state=1)
clf.fit(X_train, y_train)
print (clf.n_layers_)
print("Accuracy of MLPClassifier : '", clf.score(X_test,y_test))
```

| | 準確率 |
|---|---|
| 未正規化 | 0.369 |
| 正規化 | 0.392 |

# 4 DNN模型

→和MLP相似，可以理解為有更多隱藏層的神經網路

設定四層隱藏層

分別使用70、30、10、10個神經元

激活函數為ReLU、sigmoid
優化方法為Adam

```
model = Sequential()
model.add(Dense(70, input_dim=29, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
print(model.summary())
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
train_history=model.fit(X_train_res, y_train_res,batch_size=100,
                        epochs=30,verbose=2,
                        validation_split=0.1)
scores = model.evaluate(X_test, y_test, verbose=1)
print("Accuracy:",scores[1])
```

| | 準確率 |
|---|---|
| 未正規化 | 0.421 |
| 正規化 | 0.416 |

# 改善模型表現

## 原分類：五類

| 分數 | 16-20 | 14-15 | 12-13 | 10-11 | 0-9 |
|---|---|---|---|---|---|
| 類別 | A(1) | B(2) | C(3) | D(4) | F(5) |

## 改善後分類：兩類

| 分數 | 16-20 | 14-15 | 12-13 | 10-11 | 0-9 |
|---|---|---|---|---|---|
| 類別 | 及格(1) | 及格(1) | 及格(1) | 及格(1) | 不及格(0) |

| activities | nursery | higher | internet | romantic | famrel | freetime | goout | Dalc | Walc | health | absences | score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 4 | 3 | 4 | 1 | 1 | 3 | 6 | 0 |
| 0 | 0 | 1 | 1 | 0 | 5 | 3 | 3 | 1 | 1 | 3 | 4 | 0 |
| 0 | 1 | 1 | 1 | 0 | 4 | 3 | 2 | 2 | 3 | 3 | 10 | 0 |
| 1 | 1 | 1 | 1 | 1 | 3 | 2 | 2 | 1 | 1 | 5 | 2 | 1 |
| 0 | 1 | 1 | 0 | 0 | 4 | 3 | 2 | 1 | 2 | 5 | 4 | 0 |
| 1 | 1 | 1 | 1 | 0 | 5 | 4 | 2 | 1 | 2 | 5 | 10 | 1 |
| 0 | 1 | 1 | 1 | 0 | 4 | 4 | 4 | 1 | 1 | 3 | 0 | 1 |

# 3 改善後模型表現

## 五個類別

| | Naïve Bayes | SVM | MLP | DNN |
|---|---|---|---|---|
| 未正規化 | 0.258 | 0.374 | 0.369 | 0.421 |
| 正規化 | 0.254 | 0.387 | 0.392 | 0.416 |

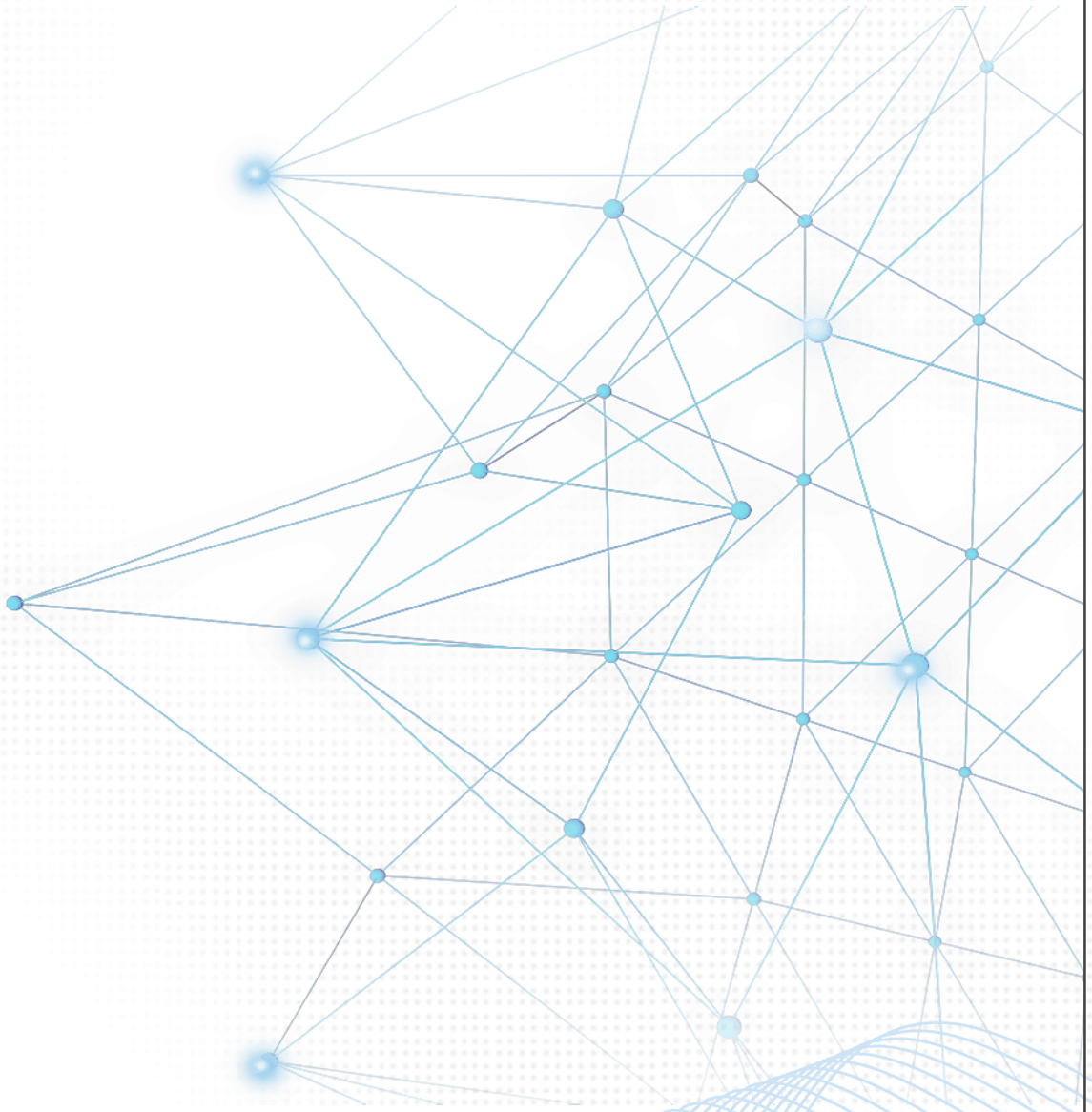
## 兩個類別

| | Naïve Bayes | SVM | MLP | DNN |
|---|---|---|---|---|
| 未正規化 | 0.766 | 0.765 | 0.770 | 0.751 |
| 正規化 | 0.765 | 0.770 | 0.728 | 0.732 |

PART FOUR

# 結論與未來展望

04

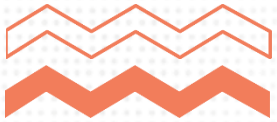


# 1 結論與未來展望

01 透過naïve Bayes、SVM、MLP、DNN四個模型預測之準確率皆在75%左右

02 特徵項與預測項關聯度不高導致準確率難以提升

03 未來應慎選特徵值並增加資料量以提升模型表現



# Thanks for listening