

## 影響學生成績的因素

### 一、摘要

利用這份資料集，觀察各項指標對學生成績是否有影響。其中，我們猜測「家長或監護人的學歷」對學生表現影響最大，故以觀察「家長或監護人的學歷」是否會影響成績為主要目的。

可能影響學生成績因素包含以下五點：

1. 家長或監護人的學歷
2. 考前是否複習
3. 學生性別
4. 種族
5. 午餐狀況

### 二、資料集介紹

我們使用的資料集是 Student performance prediction，這個資料集展示了高中學生在數學方面的表現，包含他們的成績跟人口統計資訊（種族與父母或監護人教育程度）。資料來源於美國的三所高中。

資料集的欄位包含：

	欄位	說明
1	gender	學生的性別
2	race/ethnicity	學生的種族或民族背景
3	parental education	學生父母或監護人的最高學歷
4	lunch	學生是否獲得免費或較低價錢的午餐

5	test preparation course	學生是否有進行考前複習
6	math score	學生在標準化數學考試中成績
7	reading score	學生在標準化閱讀考試中成績
8	writing score	學生在標準化寫作考試中成績

### 三、選擇方式介紹

#### 1. 機器學習方法：Random Forest Classifier 隨機林分類器

在機器學習處理，我們使用的是分類當中的隨機林的方式進行。基於學生的成績、是否複習、父母教育程度，預測學生未來成績表現。

#### 2. 模型評估：

- scikit-learn的classification\_report：

可生成文字報告，顯示分類模型的性能指標。用於評估分類問題的預測結果。

可看到的指標為：precision（精確度）、recall（召回率）、f1-score、support、accuracy（準確度）、macro average（宏平均）、weighted average（加權平均）

- randomforest.score：

可用於計算模型在測試數據集上的分類準確率。

- randomforest.feature\_importances\_：

可獲得每個對於預測目標特徵的重要性分數，分數反映了每個特徵對於模型的預測能力的貢獻程度。分數越高代表其在模型中的重要性越高。

- Silhouette coefficient：

可用來評估分類結果的好壞，Silhouette 係數越接近 1，表示聚類效果越好；越接近 -1，表示聚類效果越差。

#### 四、程式說明

##### 1. 取得資料集

取得 exam.csv

```
url = 'exams.csv'
df = pd.read_csv(url)
```

取得資料集 "exam.csv"，並存入 df 中。

##### 2. 資料前處理

gender、test preparation course、lunch 重新編碼

```
# 製作一個有效編碼器
one_hot = LabelBinarizer()

# 以一位有效編碼法對特徵編碼
gender_encoding = one_hot.fit_transform(df['gender'])
pre_encoding = one_hot.fit_transform(df['test preparation course'])
lunch_encoding = one_hot.fit_transform(df['lunch'])

# 將 one-hot encoding 結果存到新的DataFrame中
gender_df = pd.DataFrame(gender_encoding, columns=['gender'])
pre_df = pd.DataFrame(pre_encoding, columns=['pre'])
lunch_df = pd.DataFrame(lunch_encoding, columns=['lunch'])
```

gender 資料內容分為「男性」及「女性」，test preparation course 為「有預習」及「沒有預習」，lunch 則是「標準午餐」與「免費/減免午餐」。將這三個欄位的資料值重新編碼，儲存

到新的 DataFrame 中。

- 轉換 gender -> 存入 gender\_df
- 轉換 test preparation course -> 存入 pre\_df
- 轉換 lunch -> 存入 lunch\_df

#### parental education 重新編碼

```
# 父母或監護人教育
high_school = ['some high school', 'high school']
tertiary_education = ["associate's degree", "bachelor's degree", 'college', 'some college']
postgraduate_education = ["master's degree"]

def categorize_education_level(level):
    if level in high_school:
        return 0 #高中
    elif level in tertiary_education:
        return 1 #大學教育程度
    elif level in postgraduate_education:
        return 2 #研究生教育程度
df['parental_education'] = df['parental_education'].apply(categorize_education_level)
```

父母或監護人教育階級，可以分成以下幾種：

1. some high school 高中肄業
2. high school 高中
3. associate's degree 副學士
4. bachelor's degree 學士學位
5. college 學院
6. some college 學院肄業
7. master's degree 研究所

為了更簡單分辨教育階級，我們將資料用以下方式進行分組，並重新編碼：

1. 高中教育程度：some high school、high school

2. 大學教育程度：associate's degree、bachelor's degree、college、some college
3. 研究所：master's degree

#### race/ethnicity 重新編碼

```
#種族
def race_level(level):
    if level in "group A":
        return 1
    elif level in "group B":
        return 2
    elif level in "group C":
        return 3
    elif level in "group D":
        return 4
    elif level in "group E":
        return 5

df['race/ethnicity'] = df['race/ethnicity'].apply(race_level)
```

將原始種族資料轉換為數字，重新編碼。

#### math score 重新編碼

```
#數學成績標
lower_q=np.quantile(df['math score'],0.25,interpolation='lower')#下四分位數
higher_q=np.quantile(df['math score'],0.75,interpolation='higher')#上四分位數
def math_level(lev):
    if (lev <= lower_q):
        return 0 #後標
    elif (lower_q < lev and lev < higher_q):
        return 1 #均標
    elif (lev >= higher_q):
        return 2 #前標

math = df['math score'].apply(math_level)
math_df = pd.DataFrame({'math_level': math})
```

找出原始數學成績的上四分位數與下四分位數，判斷學生成績分組，加入 math\_df：

1. 分數大於上四分位數 -> 「前標」
2. 分數小於下四分位數 -> 「後標」

### 3. 其餘屬於「均標」

#### reading score 與 writing score 重新編碼

```
#讀成績標
lower_r=np.quantile(df['reading score'],0.25,interpolation='lower')#下四分位數
higher_r=np.quantile(df['reading score'],0.75,interpolation='higher')#上四分位數
def read_level(lev):
    if (lev <= lower_r):
        return 0 #後標
    elif (lower_r < lev and lev < higher_r):
        return 1 #均標
    elif (lev >= higher_r):
        return 2 #前標

df['reading score'] = df['reading score'].apply(read_level)

#寫成績標
lower_w=np.quantile(df['writing score'],0.25,interpolation='lower')#下四分位數
higher_w=np.quantile(df['writing score'],0.75,interpolation='higher')#上四分位數
def writing_level(lev):
    if (lev <= lower_w):
        return 0 #後標
    elif (lower_w < lev and lev < higher_w):
        return 1 #均標
    elif (lev >= higher_w):
        return 2 #前標

df['writing score'] = df['writing score'].apply(writing_level)
```

找出原始讀寫成績的上四分位數與下四分位數，判斷學生成績分組，更新至df：

1. 分數大於上四分位數 -> 「前標」
2. 分數小於下四分位數 -> 「後標」
3. 其餘屬於「均標」

#### 合併資料轉換結果

```
# 合併成一個DataFrame
df.drop(['gender', 'test preparation course', 'lunch'], axis=1, inplace=True)
df = pd.concat([gender_df, pre_df, lunch_df, df, math_df], axis=1)
```

將上述所有資料轉換結果，與原始 DataFrame(df) 合併。

### 3. 預測 - 隨機森林分類器(RandomForestClassifier)

建立訓練及測試集、建立列表儲存預測結果

```
#按照parental education分組
features = df[['gender', 'prepare', 'lunch', 'race/ethnicity', 'reading score', 'writing score']]
target = df['math_level']
```

取得資料集中的性別、是否預習、午餐狀況、種族、閱讀成績、寫作成績加入features 作為特徵，再以數學成績分組(math\_level)當作 target，代表目標。

```
results = {
    'edu_level': [],
    'low_achievements': [],
    'high_achievements': []
}
```

建立 results 用於儲存預測結果

## 取出特徵與目標進行隨機森林

```
for level, group in df.groupby('parental education'):
    group_features = features.loc[group.index][group['parental education'] == level]
    group_target = target[group.index][group['parental education'] == level]

    #分訓練集和測試集
    group_features_train, group_features_test, \
    group_target_train, group_target_test = \
    train_test_split(group_features, group_target, \
    test_size=0.3, random_state=42)
```

將特徵及目標依父母教育程度進行分組，形成新特徵(group\_features)及目標(group\_target)。將新目標及新特徵分為訓練集和測試集，切分比例為7:3。

```
# 產生隨機林迴歸器物件，訓練模型
randomforest = RandomForestClassifier(
    n_estimators=100,
    min_samples_split=5,
    min_samples_leaf=4,
    max_features='sqrt',
    max_depth=10,
    random_state=42
)

randomforest.fit(group_features_train, group_target_train)

# 使用訓練資料預測
pred_df = randomforest.predict(group_features_test)
```

產生隨機林迴歸器物件，訓練模型。將預測結果儲存至 pred\_df。

```
feature_importances = np.round(randomforest.feature_importances_, 2)

print('訓練集: ', randomforest.score(group_features_train, group_target_train))
print('測試集: ', randomforest.score(group_features_test, group_target_test))
print('特徵: [gender, prepare, lunch, race/ethnicity, reading score, writing score]')
print('特徵重要程度: ', feature_importances)
```

印出隨機森林模型在訓練集和測試集上的準確度，以及特徵與特徵重要程度。



## 產生預測結果

```
# 印出結果
class_names = ['high', 'normal', 'low']
achievement_low = np.mean(pred_df == 0)
achievement_high = np.mean(pred_df == 2)

if(level == 0):
    edu_level = '高中教育程度'
    edu_level_title = 'high school'
elif(level == 1):
    edu_level = '大學教育程度'
    edu_level_title = 'tertiary'
else:
    edu_level = '研究所程度'
    edu_level_title = 'postgraduate'

results['edu_level'].append(edu_level_title)
results['low_achievements'].append(achievement_low)
results['high_achievements'].append(achievement_high)

print(f"父母或監護人教育背景: {edu_level}")
print("學生的下一個數學成績預測結果: 前標 ->", round(achievement_high, 2), "/ 後標 ->", round(achievement_low, 2))
```

以 class\_names 定義成績表現。計算前標(achievement\_high)比例與後標(achievement\_low)比例，將結果儲存至 results。最後列印出各個父母教育階層下，學生的成績表現。

## 4. 模型 - 邏輯迴歸(LogisticRegression)

## 產生分類報告

```
# 產生邏輯迴歸
classifier = LogisticRegression()

# 訓練模型並作預測
model = classifier.fit(group_features_train, group_target_train)
target_predicted = model.predict(group_features_test)

# 產生分類報告
print("----- 分類報告 -----")
print(classification_report(group_target_test,
                             target_predicted,
                             target_names=class_names))
print("-----")
print()
print(target_predicted)
```

\*class\_names = ['high', 'normal', 'low'] (前標、均標、後標)

產生邏輯迴歸，對模型進行訓練並預測結果，最後印出分類報告(結果報告呈現於模型評估說明)

## 5. 模型 - Kmeans模型

## 產生Silhouette Score

```
#Silhouette Score
features, _ = make_blobs(n_samples=1000,
                        n_features=10,
                        centers=2,
                        cluster_std=0.5,
                        shuffle=True,
                        random_state=1)

model = KMeans(n_clusters=2, random_state=1).fit(features)

target_predicted = model.labels_

score = silhouette_score(features, target_predicted)

print("Silhouette Score:", score)
```

產生特徵矩陣，使用 KMeans 演算法預測分群資料的分類，將預測分類存在target\_predicted變數中，使用 silhouette\_score 函式，印出 Silhouette Score。

## 6. 預測結果圖表

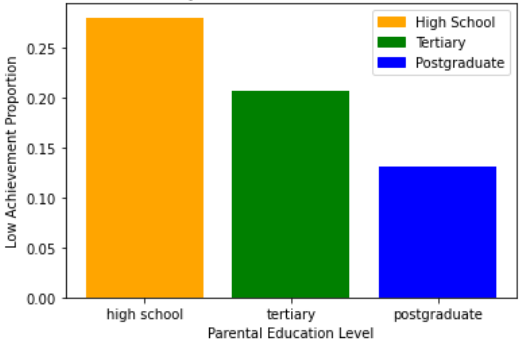
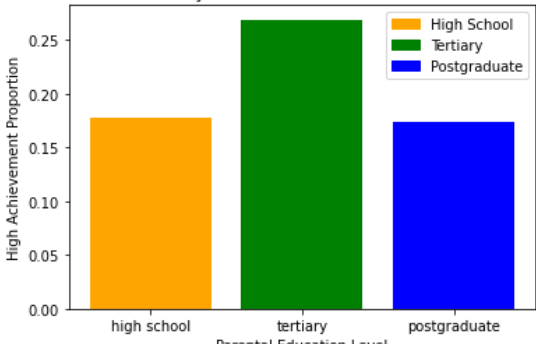
我們猜測，父母教育程度愈高，可能使學生成績表現愈好，反之，愈低則可能愈差。

## 根據預測資料製作圖表

```
colors = ['orange', 'green', 'blue']
labels = ['High School', 'Tertiary', 'Postgraduate']

# 直條圖
plt.bar(results['edu_level'], results['low_achievements'], color=colors)
plt.title('Math Score: Low Achievements \nby Parental Education Level')
plt.xlabel('Parental Education Level')
plt.ylabel('Low Achievement Proportion')
legend_patches = [mpatches.Patch(color=color, label=label) for color, label in zip(colors, labels)]
plt.legend(handles=legend_patches)
plt.show()

# 直條圖
plt.bar(results['edu_level'], results['high_achievements'], color=colors)
plt.title('Math Score: High Achievements \nby Parental Education Level')
plt.xlabel('Parental Education Level')
plt.ylabel('High Achievement Proportion')
legend_patches = [mpatches.Patch(color=color, label=label) for color, label in zip(colors, labels)]
plt.legend(handles=legend_patches)
plt.show()
```

「後標」成績比例	「前標」成績比例																
<p>Math Score: Low Achievements by Parental Education Level</p>  <table border="1"> <thead> <tr> <th>Parental Education Level</th> <th>Low Achievement Proportion</th> </tr> </thead> <tbody> <tr> <td>high school</td> <td>0.28</td> </tr> <tr> <td>tertiary</td> <td>0.21</td> </tr> <tr> <td>postgraduate</td> <td>0.13</td> </tr> </tbody> </table>	Parental Education Level	Low Achievement Proportion	high school	0.28	tertiary	0.21	postgraduate	0.13	<p>Math Score: High Achievements by Parental Education Level</p>  <table border="1"> <thead> <tr> <th>Parental Education Level</th> <th>High Achievement Proportion</th> </tr> </thead> <tbody> <tr> <td>high school</td> <td>0.18</td> </tr> <tr> <td>tertiary</td> <td>0.28</td> </tr> <tr> <td>postgraduate</td> <td>0.17</td> </tr> </tbody> </table>	Parental Education Level	High Achievement Proportion	high school	0.18	tertiary	0.28	postgraduate	0.17
Parental Education Level	Low Achievement Proportion																
high school	0.28																
tertiary	0.21																
postgraduate	0.13																
Parental Education Level	High Achievement Proportion																
high school	0.18																
tertiary	0.28																
postgraduate	0.17																
<p>利用 plt.bar 繪製直條圖，訂定標題、x軸、y軸。將數據依照 "parental education" 分組，計算每個分組中的後標與前標比例，最後繪製成直方圖。</p> <p>呈現結果：</p> <ol style="list-style-type: none"> <li>依據左圖，可以觀察到，在各自的分組下，父母或監護人教育程度愈高，學生後標比例愈小於前標，與我們的假設一致。</li> <li>根據右圖，父母或監護人教育程度為大學的族群內，獲得前標比例最高，父母或監護人教育程度高中與研究生，前標比例相仿。</li> </ol>																	

## 五、模型評估說明

- 圖中的 high, normal, low 分別表示數學成績的前標、均標、後標
- precision(精確率)、recall(召回率)：衡量機器學習模型分類效能的指標，數字愈大代表表現愈好
- f1-score：值愈接近 1，代表模型在 precision 和 recall 之間取得了一個平衡，模型的整體表現愈好。

家長或監護人學歷為「高中教育程度」的結果

```

訓練集: 0.7408759124087592
測試集: 0.7033898305084746
特徵: [gender, prepare, lunch, race/ethnicity, reading score, writing score]
特徵重要程度: [0.12 0.05 0.1 0.17 0.35 0.22]
父母或監護人教育背景: 高中教育程度
學生的下一個數學成績預測結果: 前標 -> 0.18 / 後標 -> 0.28
----- 分類報告 -----

```

	precision	recall	f1-score	support
high	0.74	0.70	0.72	37
normal	0.71	0.74	0.72	57
low	0.75	0.75	0.75	24
accuracy			0.73	118
macro avg	0.73	0.73	0.73	118
weighted avg	0.73	0.73	0.73	118

### 1. 分類準確率(randomforest.score)：

測試集的準確度為70.3%，略低於訓練集的74%，但二者相差不大，表示這個模型可以對未見過的數據進行有效的預測。

### 2. 特徵重要程度(randomforest.feature\_importances\_)：

可知預習、午餐減免(0.05 / 0.1)的重要性分數較低，對模型的預測貢獻度較低。而讀寫成績貢獻較高(0.35 / 0.22)。

## 3. 分類報告(classification\_report)：

precision 和 recall 的平衡表現不錯，且 high、normal、low 間 f1-score 差距不大。

## 家長或監護人學歷為「大學教育程度」的結果

訓練集： 0.8042895442359249

測試集： 0.71875

特徵：[gender, prepare, lunch, race/ethnicity, reading score, writing score]

特徵重要程度： [0.09 0.03 0.08 0.12 0.41 0.27]

父母或監護人教育背景：大學教育程度

學生的下一個數學成績預測結果：前標 -> 0.27 / 後標 -> 0.21

```
----- 分類報告 -----
              precision    recall  f1-score   support

   high         0.66         0.64         0.65         39
  normal         0.69         0.70         0.70         77
    low         0.77         0.77         0.77         44

 accuracy                   0.71         160
 macro avg              0.71         0.71         0.71         160
weighted avg              0.71         0.71         0.71         160
-----
```

## 1. 分類準確率(randomforest.score)：

測試集的準確度為71.8%，略低於訓練集的80.4%，但仍是一個較好的模型。

## 2. 特徵重要程度(randomforest.feature\_importances\_)：

可知預習、午餐減免、性別(0.03 / 0.8 / 0.09)的重要性分數較低，對模型的預測貢獻度較低。而讀寫成績貢獻較高(0.41 / 0.27)。

## 3. 分類報告(classification\_report)：

precision 和 recall 的平衡表現不錯，且 high、normal、low 間 f1-score 差距不大。

家長或監護人學歷為「研究所程度」的結果

```

訓練集: 0.8461538461538461
測試集: 0.782608695652174
特徵: [gender, prepare, lunch, race/ethnicity, reading score, writing score]
特徵重要程度: [0.27 0.06 0.07 0.06 0.29 0.25]
父母或監護人教育背景: 研究所程度
學生的下一個數學成績預測結果: 前標 -> 0.17 / 後標 -> 0.13
----- 分類報告 -----
              precision    recall  f1-score   support

   high         0.75         0.75         0.75         4
  normal         0.86         0.80         0.83        15
    low         0.60         0.75         0.67         4

 accuracy              0.78         23
 macro avg         0.74         0.77         0.75         23
weighted avg         0.79         0.78         0.79         23
-----

```

1. 分類準確率(randomforest.score):

測試集的準確度為78.3%，略低於訓練集的84.6%，但仍是一個較好的模型。

2. 特徵重要程度(randomforest.feature\_importances\_):

可知預習、午餐減免、種族(0.06 / 0.07 / 0.06)的重要性分數較低，對模型的預測貢獻度較低。而讀寫成績與性別貢獻較高(0.29 / 0.25 / 0.27)。

3. 分類報告(classification\_report):

由 f1-score 可見，precision 和 recall 的平衡在數學成績前標(high)與均標(normal)表現良好，對於數學成績低標(low)的預測表現普通，且略有差距。但前標與均標的 support(在測試集中實際出現的樣本數)數值較低，可能導致模型在該類別的預測表現不佳，因此可能需要對模型進行調整或改進以提高其整體表現。

由此可知，模型不存在過於簡單或過度擬合的問題，且預習、午餐減免對整體預測的影響最低，其次為種族與性別。最後，圖3 的 low 類別表現較差，可能原因為父母或監護人教育階級為研究生實際出現樣本數相較於其他兩個類別較少，因此導致模型在該類別的預測表現不佳。

三張圖 weighted avg 的值在均在0.7以上(有70%以上的預測結果是正確的)，表示模型在多分類問題上表現不錯

Silhouette coefficient
<div>Silhouette Score: 0.8916265564072142</div>
評估結果顯示為0.892。由此可顯示我們的資料組內(性別、預習、午餐減免、種族、閱讀成績、手寫成績)相似度頗高，且組間(父母或監護人教育階級：高中、大學、研究所)相似度頗低。

## 六、結論

由此可知，模型不存在過於簡單或過度擬合的問題，且預習、午餐減免對整體預測的影響最低，其次為種族與性別。最後，圖3 的 low 類別表現較差，可能原因為父母或監護人教育階級為研究生實際出現樣本數相較於其他兩個類別較少，因此導致模型在該類別的預測表現不佳。

根據觀察，在父母或監護人學歷不同的情況下的學生，閱讀成績及手寫成績明顯重要於其他特徵，可知閱讀能力與手寫能力會影響其數學成績。反之，預習、午餐減免較不影響。

根據我們的假設，父母或監護人的教育程度會影響學生成績。因此根據圖表預測結果，父母或監護人教育程度愈高，將會降低學生獲得「低標」的比例，但不會影響學生獲得「高標」比例。

- 可以改進的地方：調整模型參數或增加訓練資料量，讓測試集的數值大於或等於訓練集，避免模型過度擬合的問題