國 立 清 華 大 學

工業工程與工程管理學系

智慧化企業整合

Intelligent Integration of Enterprise

Project 2

清大工工系職涯導航儀

第 5 組

109034036 柯妤萱

109034069 方弘德

110034539 廖淯舜

中 華 民 國 一 一 三 年 六 月

目錄

圖目錄

表目錄

第一章　背景介紹與問題分析

1.1 背景介紹

求職是畢業生的重大噩夢之一，往往我們這群剛踏入社會的新鮮人仍不清楚有哪些是適合自己的工作、也沒有特殊的夢想或興趣可以當作求職目標前進。當我們正迷惘時試著上網尋求建議，卻發現留言處有許多不真實的意見、或是「暗黑」他人的狀況發生，也因為公共平台上任意人士都可以留言，所以更難以獲得到真實有用的資訊。本 project 目標就是希望可以藉由使用機器學習模型、幫助使用者可以根據自己的能力、特質去找尋最適合自己的工作。且此出路推薦是透過學長姊的經歷所構成，也因而可以從中獲得最符合現實、獲得 offer 機率最高的工作選項。

1.2 問題分析(5W1H)

表一-1、5W1H 表

Who	☐ 清大工工系畢業學生
When	☐ 畢業、求職期間
Why	☐ 網路上眾說紛紜、且沒有一定的可信度 ☐ 許多人畢業之後會開始陷入迷惘、找不到努力的目標 ☐ 幫助使用者根據能力找到適合自己的工作
What	☐ 畢業出路推薦
Where	☐ 本網站的預測專區
How	☐ 網站結合 random forest 模型

第二章　模型訓練與績效

2.1 資料集介紹

本計畫目標是透過模型預測，推薦學生未來畢業後之職涯規劃。因為難以蒐集廣泛學生的成績資料，因此我們將客群主體設定為清大工工系的學生，透過與系辦的職員進行合作，再去除可以識別個人身分的特徵後獲得部分資料後，並與線上開源資料（Career-Prediction-System）結合。學生可以透過對於自己的各種能力自我評分，以及過去在校之核心課程成績（項目說明詳見表二-3），得到系統推薦之畢業職業推薦。

資料集一共蒐集 482 筆，根據工作類別可分為六種，分別為人因、智慧製造、產品、資料科學、最佳化、及品管相關，對應之資料數目如表二-2，可以看到本計畫所蒐集的資料算是相當不平衡的，多的數目有高達 204 筆，而少的卻只有 34 筆，因此本計畫會透過資料擴增的方式去進行平衡資料，如 2.2 節說明。

表二-1、資料集部分資料示意

Logical (	coding s	public s	self-lea	reading a	memory c	Smart Ab	Technica	Hard/Smart worker	Suggested Job Role	Operatio	Operatio	Engineer	Quality !	Producti	Statistic	Human Fa	Programmi
8	2	10	5	9	8	6	5	Hard Worker	Human Factors Engineer	61	68	65	86	97	69	94	67
7	5	10	2	8	9	6	5	Hard Worker	Human Factors Engineer	67	64	70	82	92	60	90	66
4	6	9	8	6	6	8	0	Smart Worker	Human Factors Engineer	63	60	75	70	80	89	99	74
3	2	9	7	6	6	2	3	Smart Worker	Human Factors Engineer	71	71	98	91	97	67	93	66
8	1	10	4	6	8	2	5	Smart Worker	Human Factors Engineer	78	72	79	75	99	87	97	74
4	4	7	4	7	9	4	3	Smart Worker	Human Factors Engineer	77	60	76	82	94	88	93	60
4	2	7	2	6	9	4	3	Hard Worker	Human Factors Engineer	64	63	77	87	95	93	97	75
8	6	10	2	8	7	6	2	Hard Worker	Human Factors Engineer	75	78	61	91	93	95	98	62
7	1	8	3	8	9	5	1	Smart Worker	Human Factors Engineer	77	60	71	74	85	71	94	63
4	1	10	3	6	7	7	0	Smart Worker	Human Factors Engineer	75	68	64	87	94	85	92	79
3	4	8	5	6	7	3	3	Hard Worker	Human Factors Engineer	64	67	81	72	100	94	95	76
5	1	9	3	9	9	2	1	Hard Worker	Human Factors Engineer	72	68	80	99	95	63	100	64
5	3	6	7	7	7	4	2	Hard Worker	Human Factors Engineer	64	78	100	78	82	89	99	60

表二-2、各類別之資料數目

Classes	Human factor	Smart manufacturing	Product manager	Data Scientist	Optimization	Quality control
Raw data	34	204	54	51	67	73

2.2 資料前處理

本節會說明本計畫所採用的三種資料前處理方式，分別為特徵選擇（feature selection）、特徵縮放（feature scaling）、資料擴增（data augmentation）。

- 特徵選擇

本計畫雖然將目標客群鎖定於清大工工系，但系上光是核心課程就高

達 69 學分（如圖二-1），由於本計畫為透過原型探討職業推薦系統之可行性，因此透過與組員間之腦力激盪，將輸入之科目減少為 8 個。

另外，在討論期間我們認為成績雖然可以代表能力的展現，例如計算機程式成績高代表在程式撰寫上有一定的能力，往程式相關性較高這方面的工作或許會比較適合。不過，成績或許會受到當學期其他修課之影響，或是不只是 “計算機程式”這項科目會與程式能力相關，因此在這一部分會提供學生自行輸入其認知的程式能力作為判斷，會比單純只有成績而言來的更加可信。

系定必修（69學分）	工程導論	2	
	經濟學原理	3	經濟學原理一及經濟學原理二（擇 1 修之）
	微積分一、二	3 3	
	工程圖學	2	
	工場實習	1	
	普通物理一、二	3 3	
	普通物理實驗一、二	1 1	
	心理學	3	
	計算機程式語言	3	
	工作研究	3	
	機率論	3	
	工程經濟	3	
	工程統計	3	
	線性代數	3	
	離散數學	3	
	製造程序	3	
	人因工程一	3	
	品質管制	3	
	生產計劃與管制	3	
	會計學	3	
	作業研究一、二	3 3	
	工業工程專題	2	

圖二-1、清華大學工工系 112 學年大學部系定必修

- 特徵縮放

由於清華大學在成績單之登記上都是以等級制的方式（如 A+, A, A-）顯示，因此在輸入數據上會先將其轉為百分比制，如表二-3 之下半部分。

表二-3、模型輸入及輸出項目

名稱	說明	數值範圍
Logical quotient rating	邏輯能力	[1,10]
coding skills rating	程式能力	[1,10]
public speaking points	公開演講能力	[1,10]
self-learning capability?	自學能力	[1,10]

reading and writing skills	閱讀及寫作能力	[1,10]
memory capability score	記憶能力	[1,10]
Smart Ability score	聰明指數	[1,10]
Technical Skill Score	使用新技術的能力	[1,10]
Hard/Smart worker	努力型工作者或是 聰明型工作者	One-hot encoding Hard worker/ Smart worker
Operations Research I	作業研究一的學期成績	[0,100]
Operations Research II	作業研究二的學期成績	[0,100]
Engineering Economics	工程經濟的學期成績	[0,100]
Quality Management	品質管理的學期成績	[0,100]
Production Control	生產計劃與管制的學期成績	[0,100]
Statistics	工程統計的學期成績	[0,100]
Human Factors Engineering	人因工程一的學期成績	[0,100]
Programming	計算機程式語言的學期成績	[0,100]
Suggested Job Role	此模型推薦的工作類別	共有六種職業類別，如上所述

- 資料擴增

最後之資料前處理方式為資料擴增，資料擴增則是會用增加雜訊及 Resampling 兩種方法。由於本計畫所蒐集的資料數目不算多，且存在資料不平衡的現象，因此，會將各類別資料數目用至相同數目。

在增加雜訊這方面，由於成績在等級制轉換過程中，會有一段成績區間，例如百分比制中 90 分以上皆視為 A+。因此，將成績轉換過來後會在加上雜訊，使資料呈現增加隨機性，如圖二-2。

```python
def add_noise(value, range_min, range_max):
    if range_max > 10:
        noise = np.random.normal(0, 3)
        range_max = 100
        range_min = 0
    else:
        noise = np.random.normal(0, 1)
        range_max = 10
        range_min = 0

    noisy_value = value + noise
    return max(min(noisy_value, range_max), range_min)
```

圖二-2、增加雜訊程式碼

在 resampling 部分，透過對原先未平衡之資料進行重新抽樣並加上雜訊，使得各類別資料數目相同，避免模型學習未平衡資料之低準確度，圖二-3 為 resampling 程式碼。

```python
from sklearn.utils import resample

df = pd.read_csv('new_data.csv')

print(df['Suggested Job Role'].value_counts())

max_samples = df['Suggested Job Role'].value_counts().max()

df_human_factors = df[df['Suggested Job Role'] == 'Human Factors Engineer']
df_smart_manufacturing = df[df['Suggested Job Role'] == 'Smart Manufacturing Engineer']
df_product_manager = df[df['Suggested Job Role'] == 'Product Manager']
df_data_scientist = df[df['Suggested Job Role'] == 'Data Scientist']
df_optimization_engineer = df[df['Suggested Job Role'] == 'Optimization Engineer']
df_quality_control_engineer = df[df['Suggested Job Role'] == 'Quality Control Engineer']

df_human_factors_upsampled = resample(df_human_factors, replace=True, n_samples=max_samples, random_state=123)
df_smart_manufacturing_upsampled = resample(df_smart_manufacturing, replace=True, n_samples=max_samples, random_state=123)
df_product_manager_upsampled = resample(df_product_manager, replace=True, n_samples=max_samples, random_state=123)
df_data_scientist_upsampled = resample(df_data_scientist, replace=True, n_samples=max_samples, random_state=123)
df_optimization_engineer_upsampled = resample(df_optimization_engineer, replace=True, n_samples=max_samples, random_state=123)
df_quality_control_engineer_upsampled = resample(df_quality_control_engineer, replace=True, n_samples=max_samples, random_state=123)

df_balanced = pd.concat([df_human_factors_upsampled, df_smart_manufacturing_upsampled, df_product_manager_upsampled,
                         df_data_scientist_upsampled, df_optimization_engineer_upsampled, df_quality_control_engineer_upsampled])

balanced_file_path = 'balanced_data.csv'
df_balanced.to_csv(balanced_file_path, index=False)

print(f"Balanced data has been saved to {balanced_file_path}")
```

圖二-3、resampling 程式碼

表二-4、各類別平衡後資料數目

Classes	Human factor	Smart manufacturing	Product manager	Data Scientist	Optimization	Quality control
Raw data	34	204	54	51	67	73
Resampling	204	204	204	204	204	204

2.3 模型選擇

本節將根據問題特性選擇模型，分別為隨機森林（Random forest）、多層感知器（Multilayer Perceptron, MLP）、支援向量機（Support Vector Machine, SVM）、K 近鄰演算法（K-nearest Neighbors, KNN），以下將會個別說明模型。

- **隨機森林（Random forest）**

隨機森林是一種基於多個決策樹的集成學習方法。其基本思想是通過構建多棵決策樹來改善模型的分類或回歸性能。每棵樹都是由隨機選擇的特徵子集和訓練數據集生成的，這樣能夠降低單一決策樹的過擬合問題。最終預測結果是通過多棵樹的預測結果進行平均（用於回歸問題）或多數表決（用於分類問題）。

隨機森林使用袋裝法（Bagging）對數據進行多次重複抽樣。這種方法從原始訓練數據集中有放回地隨機抽取若干樣本，生成多個訓練數據集，並用這些訓練集訓練多棵決策樹。每棵樹的生成過程是相互獨立的，這樣能夠提高模型的穩定性和泛化能力。適用本問題之原因：

1. **多特徵數據處理**：學生成績數據包含多個科目分數，隨機森林能有效處理這種多特徵數據，並能夠處理缺失值。
2. **特徵重要性**：隨機森林能夠提供每個特徵的重要性指標，幫助我們理解哪些科目對職涯規劃建議最重要。

- **多層感知器（MLP）**

多層感知器是一種前饋神經網絡，由輸入層、若干隱藏層和輸出層組成。隱藏層的數量和每層的神經元數可以根據具體問題進行調整。隱藏層中的每個神經元通過激活函數將輸入進行非線性變換，使得網絡能夠學習到數據中的複雜模式和非線性關係。

MLP 通過反向傳播算法（Backpropagation）訓練網絡，該算法基於梯度下降法，通過計算損失函數對每個權重的偏導數來更新網絡權重，從而最小化損失函數。適用本問題之原因：

1. **捕捉非線性關係**：學生成績和職涯規劃之間可能存在複雜的非線性關係，MLP 能有效捕捉這些關係。
2. **靈活性**：MLP 可以通過調整隱藏層和神經元數量來適應不同的數據特徵和問題需求。

- **支援向量機（SVM）**

SVM 通過在高維空間中找到最佳超平面來最大化不同類別之間的間距。這個超平面是分類邊界，能夠有效地將不同類別的數據點分開。SVM 使用核函數（如線性核、多項式核、RBF 核等）來處理非線性分類問題。核函數能夠將原始特徵空間映射到高維特徵空間，使得在高維空間中數據變得線性可分。適用本問題之原因：

1. **多特徵數據**：SVM 能夠處理多個特徵的學生成績數據。
2. **小樣本表現良好**：SVM 在小樣本數據下也能有良好的表現，適合於樣本量較少的情況。
3. **明確分類邊界**：SVM 可以清楚地解釋分類邊界，便於理解模型的決策過程。

- **K 近鄰演算法（KNN）**

KNN 是一種基於距離的實例學習算法，該算法通過計算新數據點與訓練數據集中所有數據點之間的距離，將新數據點分類到與其最近的 K 個鄰居的多數類別中。KNN 無需訓練過程，直接利用訓練數據進行分類。每次分類時，都需要計算新數據點與所有訓練數據點的距離，然後根據距離最近的 K 個鄰居進行分類。適用本問題之原因：

1. **易於理解和實現**：KNN 算法簡單易於理解和實現。
2. **無需訓練**：KNN 無需訓練過程，適合於不需要訓練模型的應用場景。
3. **特徵較少**：KNN 適合於特徵較少的數據，例如學生成績的核心課程數據相對簡單。

表 二-5、各模型優缺點比較

方法	優點	缺點	適用情況
Random forest	處理多特徵數據、提供特徵重要性、處理缺失值	訓練和預測時間較長、對於高維數據集性能下降	多特徵數據分析、需要解釋特徵重要性
MLP	捕捉複雜非線性關係、結構靈活	訓練時間長、需要大量數據、參數調整複雜	學生成績和職涯規劃之間存在複雜非線性關係
SVM	高維數據效果好、小樣本表現佳、分	計算複雜度高、不適合大規模數據集	多特徵數據分析、小樣本數據集

	類邊界明確		
KNN	易於理解和實現、無需訓練過程、適合特徵較少的數據	計算量大、對噪音敏感、需要大量存儲空間	簡單分類問題、特徵較少的數據

2.4 模型實驗結果

本節將分成三個小節，2.4.1 小節說明 2.3 節選用模型之超參數設定；2.4.2 小節則是針對超參數使用 Optuna 對超參數進行最佳化；2.4.3 小節則是各模型在不同情況下之績效比較。

2.4.1 超參數說明及設定

本計畫所使用四種模型之超參數說明如表二-6；表二-7 為各超參數預設設定（Default setting）。

表二-6、各模型超參數說明

模型	參數名稱	描述
Random forest	n_estimators	森林中樹的數量
	max_depth	每棵樹的最大深度
	min_samples_split	每個節點至少需要的樣本數，以便進行分裂
	min_samples_leaf	每個葉子節點最少的樣本數
MLP	hidden_layer_sizes	每個隱藏層中的神經元數
	learning_rate_init	初始學習率
SVM	C	正則化（regularization）參數
	kernel	核函數類型
	degree	多項式核函數的度數
	gamma	核函數的係數
KNN	n_neighbors	K 值，即參與分類的最近鄰居數量

表二-7、各模型超參數設定

模型	參數名稱	參數設定
Random forest	n_estimators	100
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
MLP	hidden_layer_sizes	(100,)
	learning_rate_init	0.001
SVM	C	1.0
	kernel	Linear
	degree	3
	gamma	scale
KNN	n_neighbors	5

2.4.2 超參數優化

本計畫使用 Optuna 進行超參數最佳化。Optuna 為基於貝氏最佳化（Bayesian optimization）的套件，通常應用在機器學習中，用來找出最佳的超參數組合，比起使用網格搜索、隨機搜索的方式找，貝氏最佳化能更聰明的選擇超參數，可以從較少的試驗次數下找到更好的結果。

本計畫共跑 50 次 Trials 以找到最佳之參數配置，各參數最佳化範圍設定如表二-8。圖二-4 為 MLP 之 Optuna 超參數最佳化程式，要以 Optuna 進行最佳化須包含幾個要素，定義最佳化函式、最佳化範圍、超參數最佳化時之形式，如 categorical, integer, loguniform，需要根據超參數範圍定義，如 n_estimators、max_depth 這些數值為須為整數，因此在最佳化時會是整數的方式進行搜索。

表二-8、各模型超參數最佳化範圍

模型	參數名稱	參數設定
Random forest	n_estimators	[10, 200]
	max_depth	[2, 32]
	min_samples_split	[2, 20]
	min_samples_leaf	[1, 20]
MLP	hidden_layer_sizes	(50,), (100,), (50,50), (100,50), (100,100)
	learning_rate_init	[1e-5, 1e-1]
SVM	C	[1e-5, 1e2]
	kernel	'linear'、'poly'、'rbf'、'sigmoid'
	degree	[2, 5]若 kernel=‘poly’，則為 3
	gamma	‘scale’、'auto'
KNN	n_neighbors	[1, 20]

```
def mlp_objective(trial):
    hidden_layer_sizes = trial.suggest_categorical('hidden_layer_sizes', [(50,), (100,), (50,50), (100,50), (100,100)])
    learning_rate_init = trial.suggest_loguniform('learning_rate_init', 1e-5, 1e-1)
    mlp_classifier = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes, max_iter=500, learning_rate_init=learning_rate_init, random_state=42)
    mlp_cv_scores = cross_val_score(mlp_classifier, X_train, y_train, cv=5)
    return mlp_cv_scores.mean()

# 執行 MLP 的超參數調整
mlp_study = optuna.create_study(direction='maximize')
mlp_study.optimize(mlp_objective, n_trials=50)

# 輸出最佳參數
print("Best parameters for MLP:", mlp_study.best_params)

# 使用最佳參數進行訓練和預測
best_mlp_classifier = MLPClassifier(hidden_layer_sizes=mlp_study.best_params['hidden_layer_sizes'],
                                    max_iter=500,
                                    learning_rate_init=mlp_study.best_params['learning_rate_init'],
                                    random_state=42)
best_mlp_classifier.fit(X_train, y_train)
y_pred_best_mlp = best_mlp_classifier.predict(X_test)
best_mlp_accuracy = accuracy_score(y_test, y_pred_best_mlp)
best_mlp_classification_report = classification_report(y_test, y_pred_best_mlp)
```

圖二-4、Optuna 超參數程式-以 MLP 為例

2.4.3 模型比較

本計畫將原始資料(未平衡資料前)，並使用基礎模型參數(未用 Optuna 最佳化）與平衡資料及使用最佳參數組合比較。表二-9~表二-12 為四種模

型之結果評估，Random Forest 在所有條件下都表現得相對穩定且優越，尤其是在不平衡數據的情況下；MLP 在最佳化調參後的績效有所波動，但在平衡數據下表現出色；SVM 的最佳化調參效果不如預期，但在平衡數據下也有不錯的表現；KNN 在最佳化調參後，平衡數據下的績效大幅提升，但在不平衡數據下表現相對較差。

表二-9、Random forest 評估結果表

Data form & parameter setting	precision	recall	f1-score
Unbalanced data with default parameter	0.75	0.77	0.75
Unbalanced data with optimizing parameter	0.7	0.76	0.73
Balanced data with default parameter	0.98	0.98	0.98
Balanced data with optimizing parameter	0.98	0.98	0.98

表二-10、MLP 評估結果表

Data form & parameter setting	precision	recall	f1-score
Unbalanced data with default parameter	0.76	0.76	0.75
Unbalanced data with optimizing parameter	0.85	0.82	0.8
Balanced data with default parameter	0.99	0.99	0.99
Balanced data with optimizing parameter	0.98	0.98	0.98

表二-11、SVM 評估結果表

Data form & parameter setting	precision	recall	f1-score
Unbalanced data with default parameter	0.74	0.76	0.74

Unbalanced data with optimizing parameter	0.71	0.78	0.74
Balanced data with default parameter	0.83	0.82	0.82
Balanced data with optimizing parameter	0.97	0.97	0.97

表二-12、KNN 評估結果表

Data form & parameter setting	**precision**	**recall**	**f1-score**
Unbalanced data with default parameter	0.75	0.74	0.74
Unbalanced data with optimizing parameter	0.84	0.8	0.77
Balanced data with default parameter	0.85	0.85	0.85
Balanced data with optimizing parameter	0.97	0.97	0.97

根據表二-13，在數據平衡過後，所有模型之 Cross-validation accuracy 皆有上升；但 Random forest 之模型在任何情況下，皆優於其餘模型，其中在數據平衡後以及使用預設參數，為所有組合下準確率最高的，因此後續預測模型將會使用此模型。

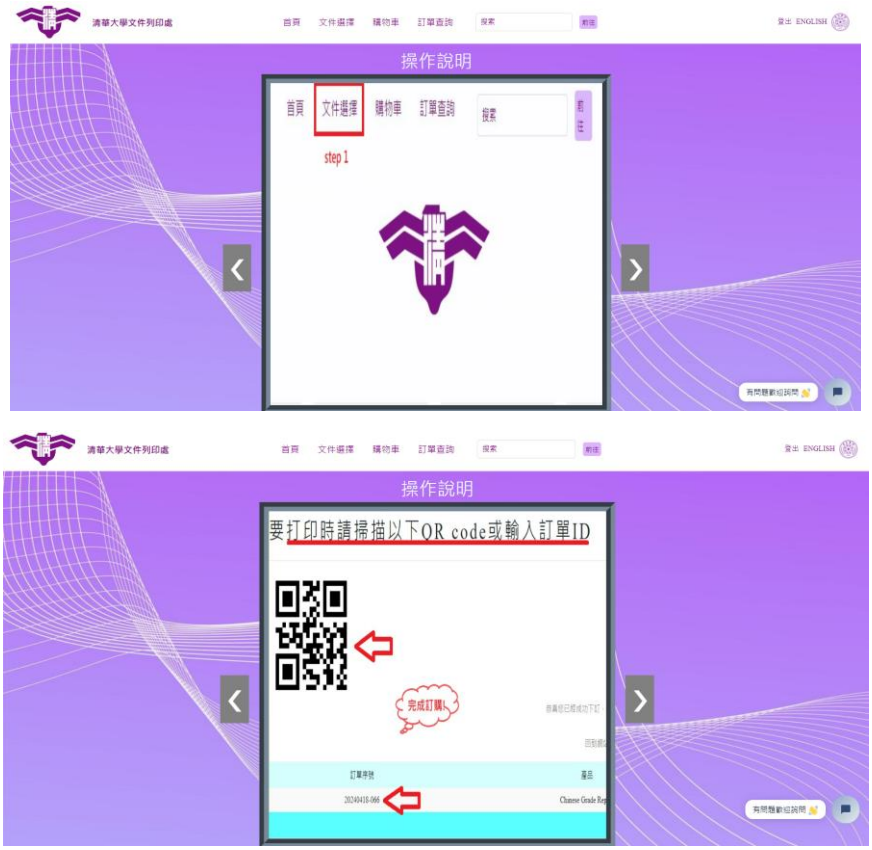
表二-13、四種模型之 Cross-validation accuracy

	Random forest	MLP	SVM	KNN
Unbalanced data with default parameter	**0.7330**	0.7185	0.708	0.4783
Unbalanced data with optimizing parameter	**0.7370**	0.6087	0.4224	0.5424
Balanced data with default parameter	**0.9804**	0.7908	0.8554	0.7255

Balanced data with optimizing parameter	**0.9779**	0.7917	0.848	0.9436

第三章　Web

3.1 Web-前端功能介紹



圖三-1、Web 前端-首頁

首頁的正中心處即有完整的操作說明，讓使用者馬上就能簡單易瞭此系統如何使用。

圖三-2、Web 前端-選擇文件

文件選擇中，我們比照現有機台的文件來做設置，讓使用者可以順利選取想要的文件。

圖三-3、Web 前端-購物車

在此處，可以確認所選購的文件項目。除此之外，當文件數量欲更改時，直接在數量的欄位修改、並按下確認修改鍵，即可順利完成修改。在右下角有設置總金額幫助使用者確認金額總數。

圖三-4、Web 前端-訂購資料填寫

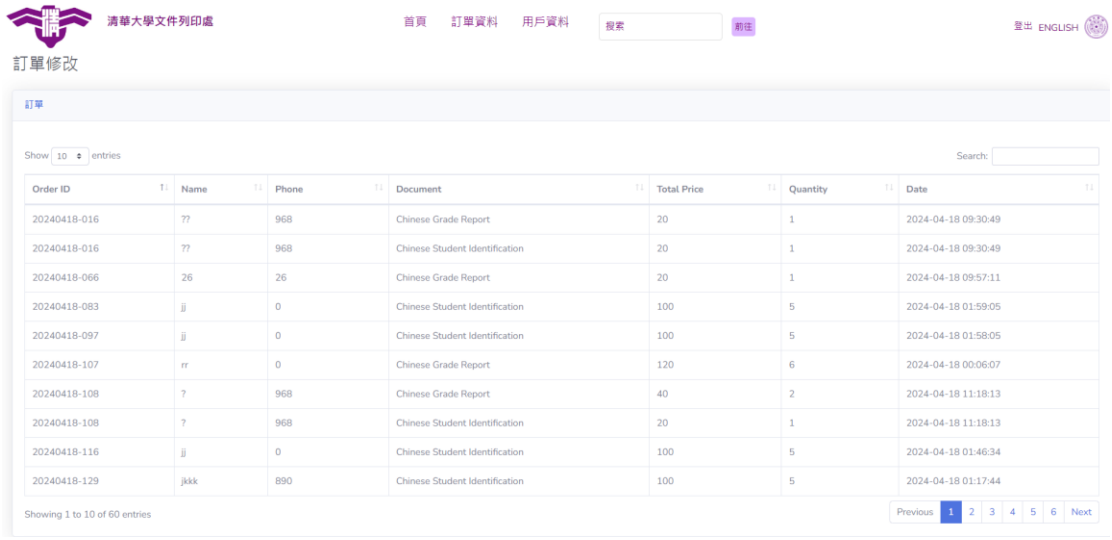
此處是確認訂購人的資料以便確認。

圖三-5、Web 前端-訂購完成畫面

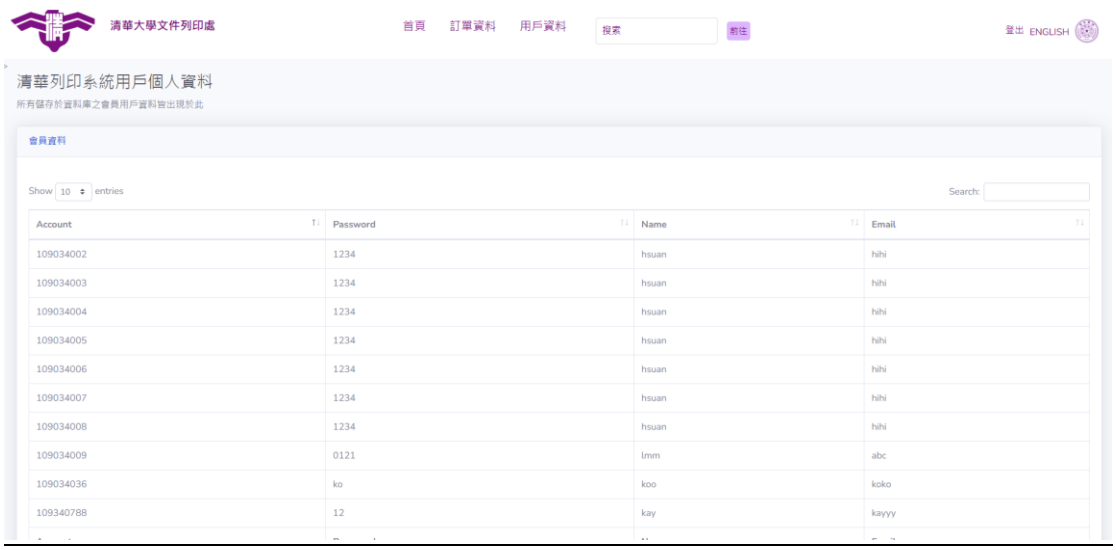
當訂單送出後，只要到機台掃描此處的 QR code 或是手動輸入序號即可順利獲取文件。

圖三-6、Web 前端-訂單查詢

3.2 Web-後端功能介紹



圖三-7、Web 後端-訂單資料

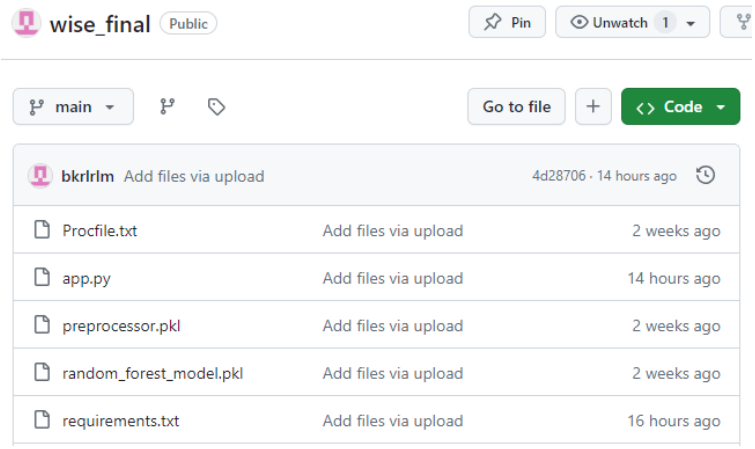


圖三-8、Web 後端-用戶資料

後端可以幫助確認使用者的訂單，也可以修改用戶的個人資料。

3.3 AI—學生工作出路推薦

為了讓建立的網站可以與模型串接，我們使用 Flask 來處理預測的請求，再使用 Github 來進行版本控制、並將其應用部署到 render 上，此部分的主要架構有: prepocessor.pkl、random_forest_model.pkl、Procfile.txt、requirements.txt、app.py 等部分。



圖三-9、使用 Github 版本控制

(1)prepocessor.pkl 與 random_forest_model.pkl

這兩個檔案均為模型參數檔，prepocessor.pkl 是用來處理使用者輸入的數值、將其特徵進行標準化，使 input 給模型時可以保證是相同的 transform 方式。而 random_forest_model.pkl 是主要模型部分，用來預測最佳推薦工作出路。

```
import joblib

joblib.dump(rf_classifier, 'random_forest_model.pkl')
joblib.dump(preprocessor, 'preprocessor.pkl')
```

圖 三-10、模型保存

(2) Procfile.txt 與 requirements.txt

Procfile.txt 是用來指定應用入口點，requirements.txt 是為了列出執行此程式時，所需要的所有依賴 package，這些 package 會在部署時自動安裝。

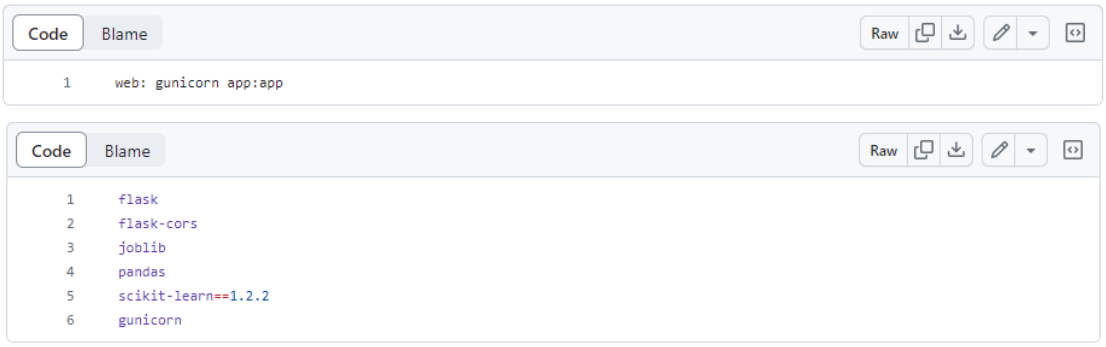


圖 三-11、部署相關文件

(3)app.py

此部分是最主要創建 flask 來處理網頁端的 POST 請求、以及利用 json 檔來做 response。除此之外，我們也加入"CORS"來處理跨域的問題，讓此 API 可以被其他域訪問。程式碼如下:

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import joblib
import pandas as pd
import logging

logging.basicConfig(level=logging.INFO)

try:
    rf_classifier = joblib.load('random_forest_model.pkl')
    preprocessor = joblib.load('preprocessor.pkl')
except Exception as e:
    logging.error(f"Error loading model or preprocessor: {e}")
    raise e

app = Flask(__name__)
CORS(app)

@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()
        logging.info(f"Received data: {data}")


        required_fields = [
            'Logical quotient rating', 'coding skills rating', 'public speaking points',
            'self-learning capability?', 'reading and writing skills', 'memory capability score',
            'Smart Ability score', 'Technical Skill Score', 'Hard/Smart worker',
            'Operations Research I', 'Operations Research II', 'Engineering Economics',
            'Quality Management', 'Production Control', 'Statistics',
            'Human Factors Engineering', 'Programming'
        ]

        for field in required_fields:
            if field not in data:
                return jsonify({'error': f"'{field}' field is missing"}), 400

        new_data_df = pd.DataFrame([data])
        new_data_df = pd.get_dummies(new_data_df, columns=['Hard/Smart worker'], drop_first=True)

        for col in preprocessor.feature_names_in_:
            if col not in new_data_df.columns:
                new_data_df[col] = 0

        new_data_df = new_data_df[preprocessor.feature_names_in_]
        new_data_scaled = preprocessor.transform(new_data_df)

        prediction = rf_classifier.predict(new_data_scaled)
        logging.info(f"Prediction: {prediction[0]}")

        return jsonify({'predicted_job_role': prediction[0]})
    except Exception as e:
        logging.error(f"Error during prediction: {e}")
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

圖 三-12、app.py 具體程式碼

(4) 成績輸入與能力評估前端頁面

此網頁讓用戶對自身能力（如公眾發言、程式能力、自我提升能力等等）以及主科的成績進行評估與輸入。輸入的成績將被傳送到已訓練的機器學習模型進行工作出路的預測與推薦，並將結果返回頁面供學生參考。

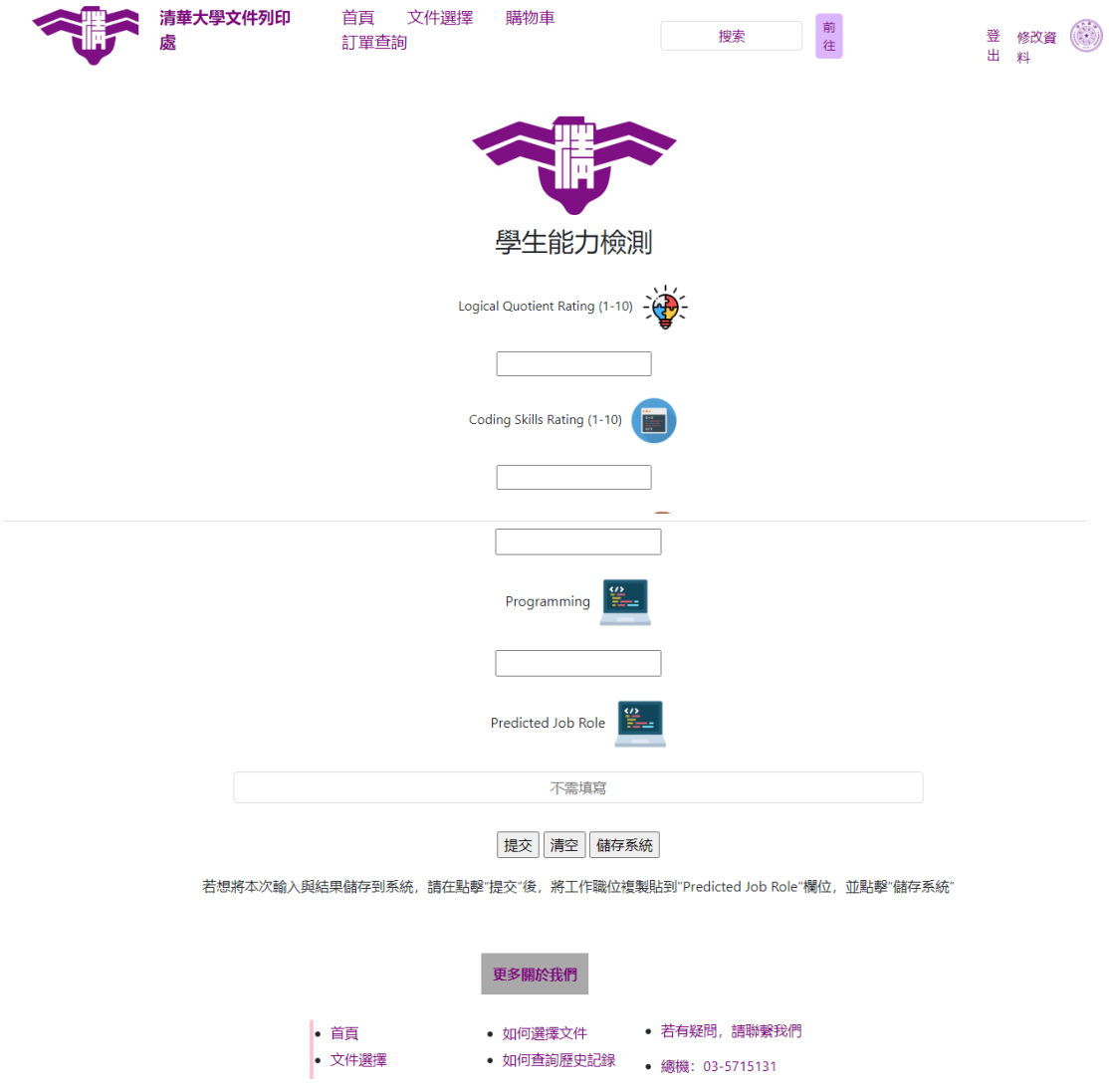


圖 三-13 、AI 前端網站圖

圖 三-14、預測結果示例

(5) 選擇是否將成績與預測結果儲存到後臺系統

學生在獲取預測結果後，可選擇是否要保存該次的成績輸入與預測結果到後台系統（與先前介紹之 Web 後端相同）。若是，先將預測結果複製到”Predicted Job Role”欄位，再點擊“儲存系統”按鈕；若否，則離開該網頁即可。

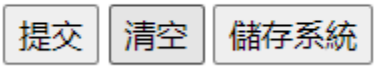


圖 三-15、按鍵功能圖

第四章　結論

我們之前透過所建置的網站、後台、chatbot 以及 APP，讓使用者可以方便地在我們的系統中獲得完整的清大成績或相關文件列印服務，同時管理者也可以藉由在後台管理、讓資訊可以更順利流通。這一次我們提供了清大工業工程與工程管理學系學生工作推薦的服務。通過結合機器學習模型，讓清大工工系學生在網站輸入主科成績與對自身軟技能的評分，再通過模型根據以上數據資料進行工作職位的推薦。接著，學生也可選擇是否將該次的輸入與預測結果保存在系統後台。我們的機器學習模型不止針對主科成績，也對軟實力或自我發展技能納入考量，因此相信可以提供學生高精確度與參考價值的工作推薦。我們希望未來可以提升模型的泛用性，讓全台灣的工業工程與工程管理學系學生都能透過我們的模型進行工作職位的推薦，為所有學生提供更多的便利。