商業智慧期末報告

HR Analytics: Job Change of Data Scientists

組員:
資管碩一 110423026 洪睿甫
資管碩一 110423057 張哲維

目錄

1. Business Understanding

1.1. 商業背景與目標

一間科技公司致力於從事於數據分析，為了擴大公司規模目前正在招募新的資料科學家，招聘方式為公司會先行開設多個跟公司業務相關的訓練課程，當參與課程的學員順利完成課程並通過考核後，公司會從中挑選適合的人員作為潛在的招聘對象。為了降低公司的招聘成本，公司希望能夠在訓練階段就能夠依照學員的背景如年齡，教育，經驗等相關資料來找出最有可能會想要離開目前的職位，並加入己方公司的學員，使公司能夠針對性提供特殊課程以吸引對方加入。

1.2. 預期目標

A. 能夠建立起一預測模型，透過輸入學員的資料即可判斷此人未來是否有興趣實際投遞履歷加入我方公司

B. 找出與有興趣投遞職缺最相關的資料欄位，以供人資部門作為未來接受課程申請時的判斷

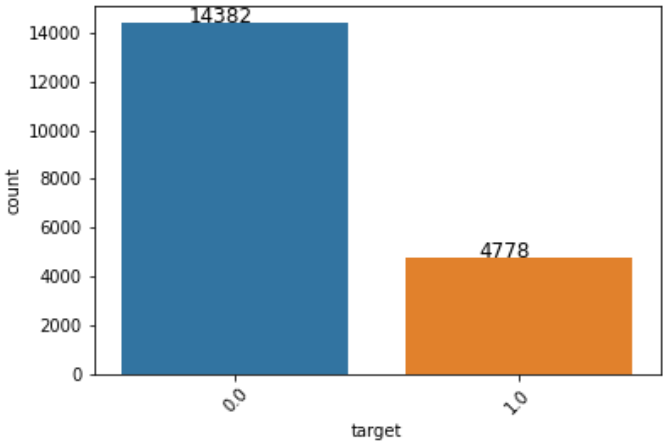
2. Data Understanding

2.1. 資料來源

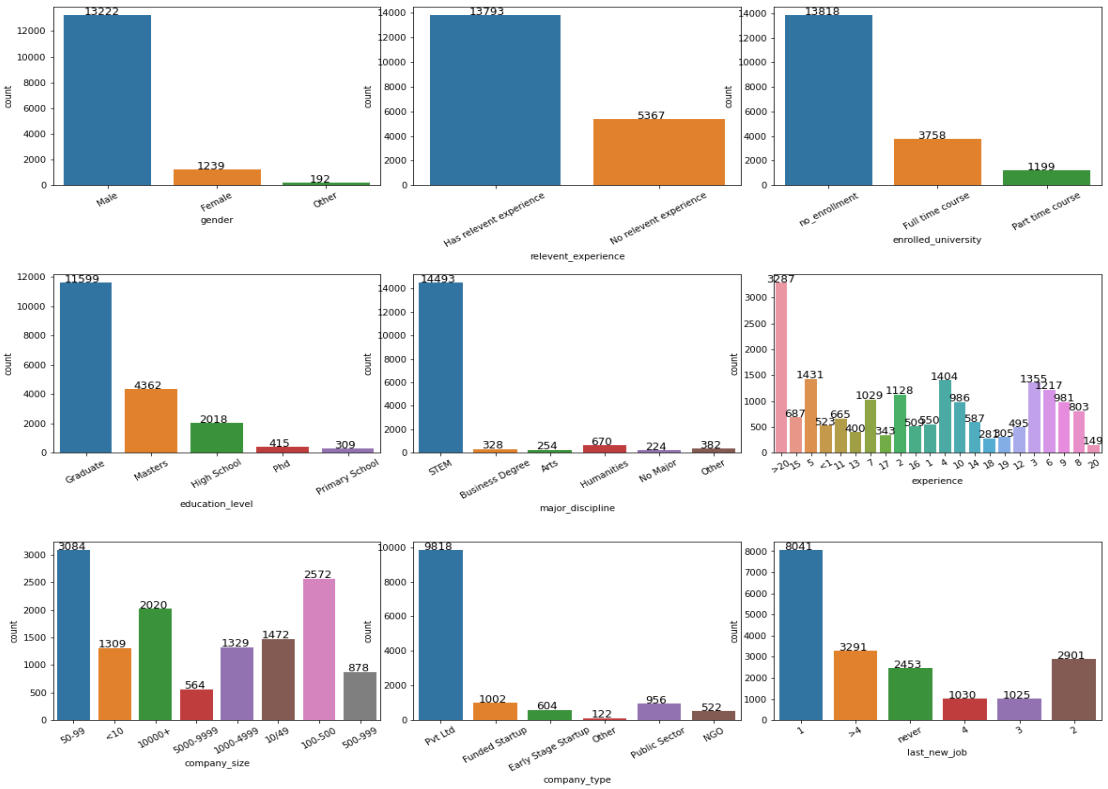
本資料集來源為 kaggle 上的資料集 [HR Analytics: Job Change of Data Scientists](), 為一授權為 CC0 的公開資料集。

2.2. 資料描述

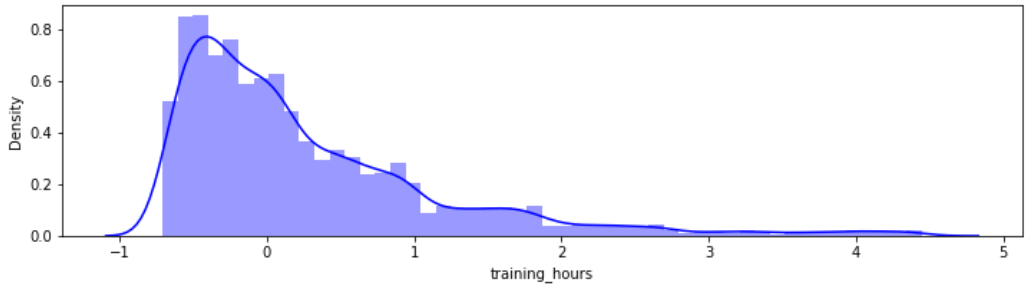
本資料集的總筆數共有 19160 筆，訓練的目標欄位為 1 及 0，分別代表有興趣更換任職公司與無興趣更換。而實際資料分布狀況如下圖

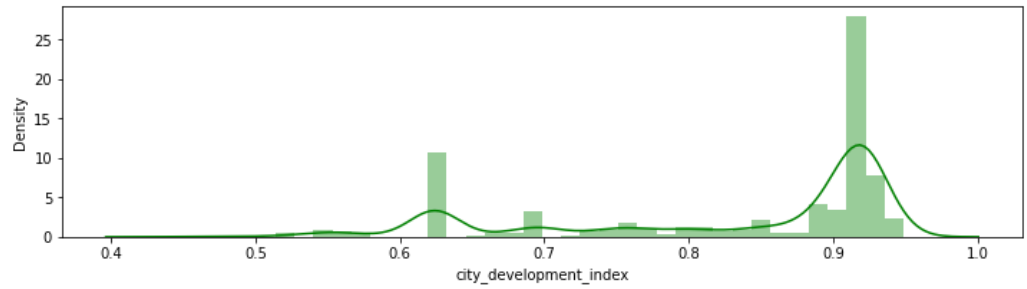


由此圖可見資料集有一定程度的不平衡。而除了目標欄位外其他總共還有 13 個欄位，部分欄位的分布狀況如下。



而 training hours 及城市發展程度則如下





並且我們也發現此資料集中存在大量出現空值的情況，因此我們資料前處理也會著重於處理這方面的問題

	enrollee_i	city	city_devel	gender	relevent_e	enrolled_u	education_	major_dis	experience	company_	company_	last_new_	training_h	target
2	8949	city_103	0.92	Male	Has releve	no_enrollr	Graduate	STEM	>20			1	36	1
3	29725	city_40	0.776	Male	No relever	no_enrollr	Graduate	STEM	15	50-99	Pvt Ltd	>4	47	0
4	11561	city_21	0.624		No relever	Full time (	Graduate	STEM	5			never	83	0
5	33241	city_115	0.789		No relevent	experien	Graduate	Business I	<1		Pvt Ltd	never	52	1
6	666	city_162	0.767	Male	Has releve	no_enrollr	Masters	STEM	>20	50-99	Funded St	4	8	0
7	21651	city_176	0.764		Has releve	Part time (	Graduate	STEM	11			1	24	1
8	28806	city_160	0.92	Male	Has releve	no_enrollr	High School		5	50-99	Funded St	1	24	0
9	402	city_46	0.762	Male	Has releve	no_enrollr	Graduate	STEM	13	<10	Pvt Ltd	>4	18	1
10	27107	city_103	0.92	Male	Has releve	no_enrollr	Graduate	STEM	7	50-99	Pvt Ltd	1	46	1
11	699	city_103	0.92		Has releve	no_enrollr	Graduate	STEM	17	10000+	Pvt Ltd	>4	123	0
12	29452	city_21	0.624		No relever	Full time (	High School		2			never	32	1
13	23853	city_103	0.92	Male	Has releve	no_enrollr	Graduate	STEM	5	5000-9999	Pvt Ltd	1	108	0
14	25619	city_61	0.913	Male	Has releve	no_enrollr	Graduate	STEM	>20	1000-4999	Pvt Ltd	3	23	0
15	5826	city_21	0.624	Male	No relevent	experience			2			never	24	0
16	8722	city_21	0.624		No relever	Full time (	High School		5			never	26	0
17	6588	city_114	0.926	Male	Has releve	no_enrollr	Graduate	STEM	16	10/49	Pvt Ltd	>4	18	0

3. Data Preparation

3.1. 特徵選取

因為本資料集的欄位較少，而且絕大多數的特徵依照常識判斷應該都有機會對目標有影響，因此在此階段我們僅把註冊時所提供的ＩＤ刪除，其他全部留著進入下一階段

```
df_train = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/datasets/aug_train.csv')
df_train = df_train.drop(columns = ['enrollee_id'])
cate_col = ['city', 'gender', 'relevent_experience', 'enrolled_university',
            'education_level', 'major_discipline', 'experience','company_size',
            'company_type', 'last_new_job']
```

3.2. label encoding

因我們的特徵中有需多都屬於名目型資料，為方便後續模型訓練，因此先將名目型資料的所有類別都先列舉出來。

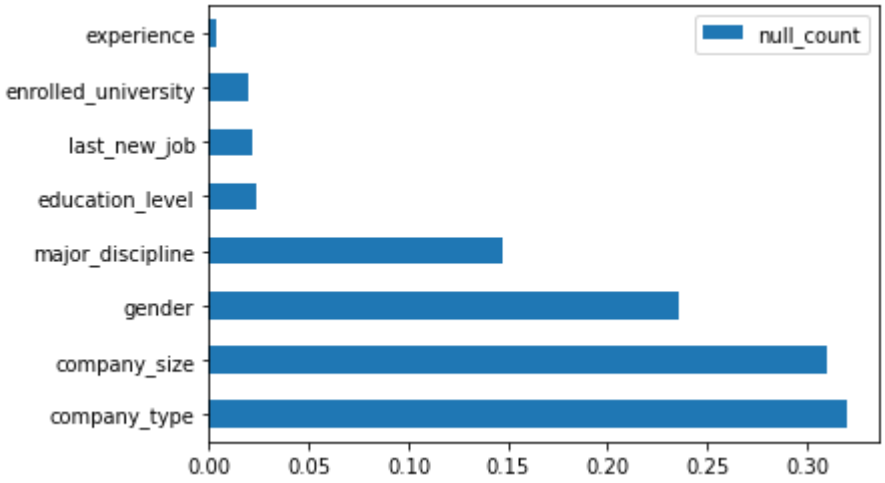
```
dict_education_level = {value : key for key,value in enumerate(df_train.education_level.value_counts().index)}
dict_major_discipline = {value : key for key,value in enumerate(df_train.major_discipline.value_counts().index)
dict_experience = {value : key for key,value in enumerate(df_train.experience.value_counts().index)}
dict_company_size = {value : key for key,value in enumerate(df_train.company_size.value_counts().index)}
dict_company_type = {value : key for key,value in enumerate(df_train.company_type.value_counts().index)}
dict_company_last_new_job = {value : key for key,value in enumerate(df_train.last_new_job.value_counts().index)
```

之後再將這些資料轉換為非連續數值型的標籤

```
#categoricalconvert into numeric label
df_train['gender'] = df_train['gender'].map({'Male':0,'Female':1,'Other':2})
df_train['enrolled_university'] = df_train['enrolled_university'].map({'no_enrollment':0,'Full time course':1,'Part time course':2})
df_train['education_level'] = df_train['education_level'].map(dict_education_level)
df_train['major_discipline'] = df_train['major_discipline'].map(dict_major_discipline)
df_train['experience'] = df_train['experience'].map(dict_experience)
df_train['company_size'] = df_train['company_size'].map(dict_company_size)
df_train['company_type'] = df_train['company_type'].map(dict_company_type)
df_train['last_new_job'] = df_train['last_new_job'].map(dict_company_last_new_job)
```

3.3. 補空值

從下圖的空值百分比值條圖可以看到，在大多數的類別特徵中都存在空值，就以 company_size 和 company_type 為例，這兩個特徵的空值就佔了整份資料集的三成多左右，因此要把含有空值的資料刪除的話，可能會需要刪除掉整份資料及三成以上的資料，而考量到資料量的大小可能會影響到模型的效能好壞，我們主要採用了三種的補空值策略來避免減少資料量的問題。在之後我們將建立模型對三種補值策略來做評估，檢查哪種補值方式對模型可以達到最佳的表現。



3.3.1. 以眾數補值:

找出每項特徵的出現次數最頻繁的類別，並將空值補上此類別

```
#strategy1  空值補眾數
def  fill_median(df):
    imputer  =  SimpleImputer(missing_values=np.nan,  strategy='most_frequent')
    imp  =  imputer.fit(df)
    dfimpute  =  imp.transform(df)
    dfimpute  =  pd.DataFrame(dfimpute,columns  =  df.columns)
    dfimpute[catecol]  =  dfimpute[catecol].astype(int)
    dfimpute['target']  =  dfimpute.target.astype(int)
    return  dfimpute
```

3.3.2. 分群補值

將資料以 KNN 分成三群，並將空值以其他同群中的類別補上。

```
#strategy2  KNNimputer
def  knnimputer(df):
    impute  =  KNNImputer(n_neighbors  =  3)  #KNN  imputation
    dfimpute  =  np.round(impute.fit_transform(df))
    dfimpute  =  pd.DataFrame(dfimpute,  columns  =  df.columns).round(1)
    dfimpute[catecol]  =  dfimpute[catecol].astype(int)
    dfimpute['target']  =  dfimpute.target.astype(int)
    return  dfimpute
```

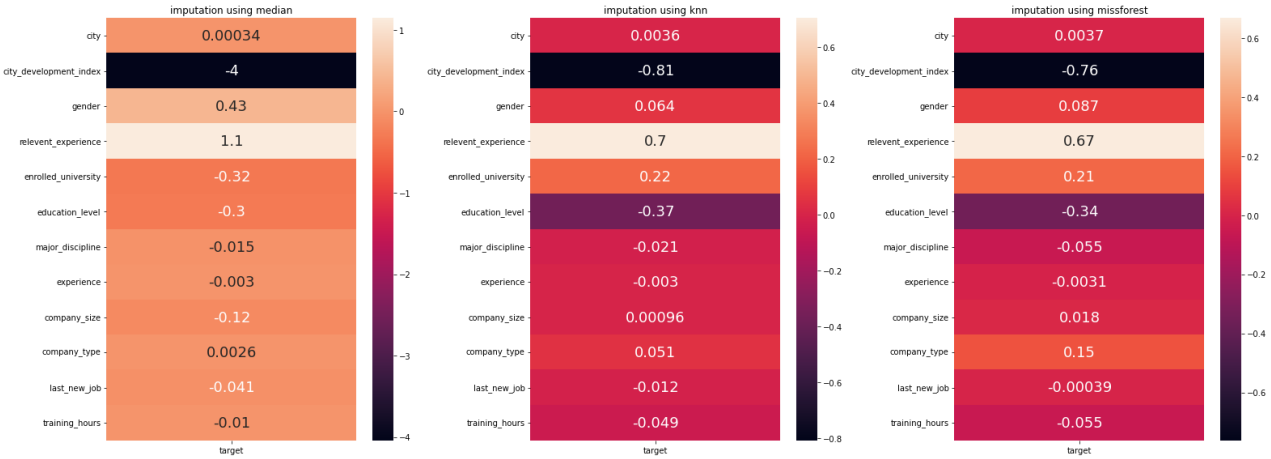
3.3.3. 以 MissForest 補值

將空值作為一個預測的目標，並以其他的存在的特徵來建立 RandomForest，並預測這個空的特徵值為何。並進行 5 個循環來提高準確度。

```
def missforest(df):
    imputer = MissForest() #miss forest
    dfimpute = np.round(imputer.fit_transform(df))
    dfimpute = pd.DataFrame(dfimpute, columns = df.columns).round(1)
    dfimpute[catecol] = dfimpute[catecol].astype(int)
    dfimpute['target'] = dfimpute.target.astype(int)
    return dfimpute
```

3.4. 進行資料前處理完的新發現

將以上步驟皆處理完後，我們將三種不同的補值方式都以 Logistic Regression 建立一模型來去分析各項不同的特徵跟目標之間的關係重要程度。並將數值繪製成下圖 heat map



可以發現過去具有相關工作經驗對於目標有最大的正向效果，也就是有相關經驗者最有可能會想要變換工作到本公司。而居住地的城市發展程度則有最明顯的負向關係。

4. Modeling

4.1. 模型設計概述

首先訓練資料會如前段所描述的，會以三種不同的補值方式產生三種不同的訓練資料。並且我們也選擇 Random Forest, Naive Bayes, XGBoost 作為三種測試模型，每種模型都會分別以三個不同的訓練資料集進行訓練，因此最後會有 3X3 種共 9 個模型產生。

完成初步的模型訓練後，接著觀察何種模型組合效果最好，並再進行參數微調，最後將三種模型結合以 Voting Classfier 與 Stacking 做 ensable leanring 產生新模型，最後再與原始模型做比較。

4.2. 切分訓練與測試資料

為了後續的模型評估，首先我們將資料集以 9:1 的比例切分為訓練集跟測試集，並且為了維持訓練集跟測試集的代表性，我們以目標欄位做分層抽樣。讓測試集跟訓練集中的兩種類別的比例相同。

```
#進行train test split
from sklearn.model_selection import train_test_split
X_train_median, X_test_median,Y_train_median, Y_test_median= train_test_split(df_median_encoding.iloc[:,0:12],df_median_encoding['Class'],test_size=0.1, stratify=df_median_encoding['Class'])
X_train_knn, X_test_knn,Y_train_knn, Y_test_knn= train_test_split(df_knn_encoding.iloc[:,0:12],df_knn_encoding['Class'],test_size=0.1, stratify=df_knn_encoding['Class'])
X_train_missforest, X_test_missforest, Y_train_missforest, Y_test_missforest= train_test_split(df_missforest_encoding.iloc[:,0:12],df_missforest_encoding['Class'],test_size=0.1, stratify=df_missforest_encoding['Class'])
```

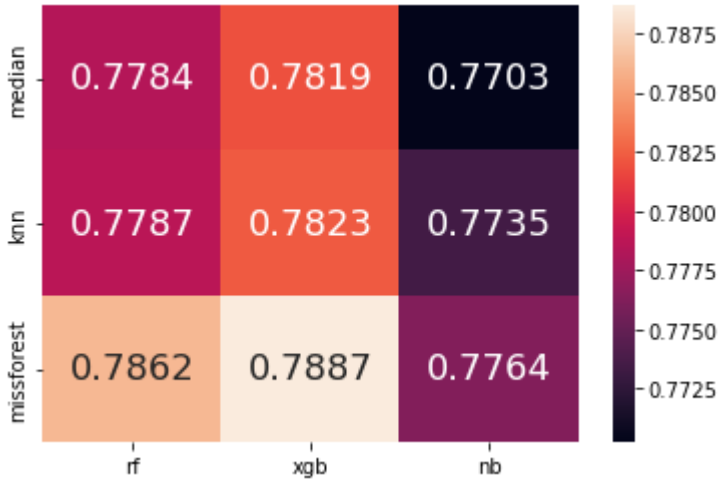
4.3. 產生初步模型

首先將三種模型:Random Forest, Naive Bayes, XGBoost 分別給予三種不同補值方式的資料集進行 training set 的 crossvalidation，因此我們會得到九種不同的模型分數。

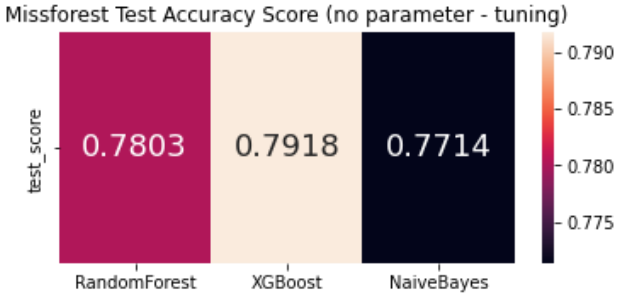
```
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
def get_models():
    models = dict()
    models['rf'] = RandomForestClassifier()
    models['xgb'] = XGBClassifier()
    models['nb'] = GaussianNB()
    return models

def evaluate_model(model,X,Y):
    cv = KFold(n_splits=5, random_state=42, shuffle=True)
    scores = cross_val_score(model, X, Y, scoring='accuracy', cv = cv)
    return scores

def evaluate_testset(model,X_train,Y_train,X_test,Y_test):
    model = model.fit(X_train)
    y_pred = model.predict(X_test)
    acc_score = acc(Y_test,y_pred)
    return acc_score

def show_result(X,Y):
    models = get_models()
    results, names = list(), list()
    for name, model in models.items():
        scores = evaluate_model(model, X, Y)
        results.append(mean(scores))
        names.append(name)
    return results, names
```

4.4. 初步評估

根據上一步驟，我們將三種不同的補值方式與三種模型進行交叉比對，分別以準確度(accuracy)來評估各自的好壞，並將結果繪製成熱度圖。圖中的 X 軸為模型種類，Y 軸則為補值方式。



在此圖中可以發現，以模型的角度來看，RandomForest 與 XGBoost 有相當類似的表現，而 Naive Bayes 的表現則相對來說就比較差。而已補值方式的角度來比較，MissForest 的表現則明顯好於其他兩種，因此在下階段的模型微調與訓練，就統一採用 MissForest 作為後續的訓練資料集。此外我們也可以看到使用 missforest 當作補值方法來對 testset 來做預測的分數如下圖所示



4.5. 模型參數微調

由於三種模型都是使用預設的模型參數， 可能沒辦法得到最佳的分數，因此我們透過 parameter tuning 中的 gridsearchCV 方法來找出每個模型的最佳參數，希望能藉此提高最終的預測分數

* 三種模型在做 gridsearch 時都使用 cv=5, scoring=’ accuracy’

- RandomForest

  調整的參數分別是

  'n_estimators': [200, 500, 700],

  'max_features': ['auto', 'sqrt', 'log2'],

  'max_depth' : [4,5,6,7,8],

  'criterion' :['gini', 'entropy']

  我們在 randomforest 的調參結果如下 :criterion=gini, max_depth=8, max_feature='log2', n_estimators=700

```
Random Forest GridSearch Best Parameter : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2', 'n_estimators': 700}
```

- Naive Bayes

  'var_smoothing': np.logspace(0,-9, num=100)

```
Naive Bayes GridSearch Best Parameter : {'var_smoothing': 0.0015199110829529332}
```

- XGBoost

  'learning_rate': [0.05, 0.1, 0.20],

  'gamma': [0.5, 1, 1.5, 2, 5],

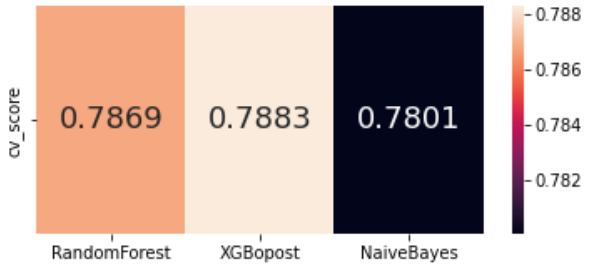
  'subsample': [0.6, 0.8, 1.0],

  'colsample_bytree': [0.6, 0.8, 1.0],

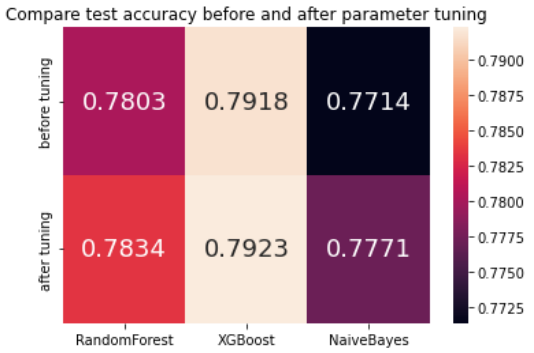
  'max_depth': [3, 4, 5, 6]

```
XGBoost GridSearch Best Parameter : {'colsample_bytree': 1.0, 'gamma': 1, 'learning_rate': 0.1, 'max_depth': 4, 'subsample': 1.0
```

- 三種模型使用最佳參數後再對 trainset 做 Kfold validation 的平均分數如下，基本上除了 XGboost 之外 accuracy 都有些許的提升。



- 接著再使用最佳參數的模型來對 testset 做預測，可以看到在調過參數之後每個分類器的預測分數都有所提昇(圖中第一列是調參數之前，第二列是調參數後)



4.6. Ensemble Method

除了使用單一個模型來預測之外我們嘗試了 Ensemble 的方法包括了 Voting 和 Stacking，主要比較使用 ensemble 的方法是否能夠再進一步的提升預測能力

*在此階段我們的模型都是使用 gridsearch 後的最佳參數去 ensemble learning

- Voting

使用 sklearn 的 VotingClassifier 套件，voting 主要會對每個模型的預測出來的機率來以多數決的方式決定說這一筆資料是屬於哪一類

```
rf = RandomForestClassifier(**rf_girdsearch.best_params_)
nb = GaussianNB(**nb_girdsearch.best_params_)
xgb = XGBClassifier(**xgb_girdsearch.best_params_)
voting_ensemble = VotingClassifier(estimators=[('RandomForest', rf),
                                               ('NaiveBayes', nb),
                                               ('XGBoost', xgb)], voting='hard')
voting_ensemble = voting_ensemble.fit(X_train_missforest,Y_train_missforest)
```

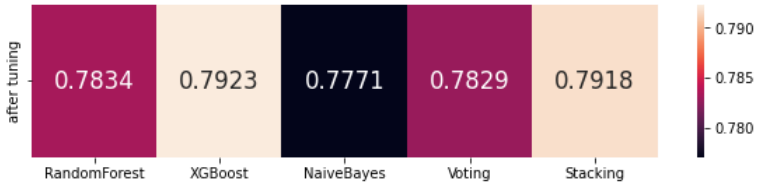
- Stacking

使用 sklearn 的 StackingClassifier 套件，Stacking 當中需要 based model 和 metamodel，因此我們使用了前面的三個模型(randomforerst, naivebayes, xgboost)來當作是 based model，而在 meta model 的部份，根據網路上查到的說法，使用較簡單的分類器可以有效的避免 overfitting 的問題並取得較好的結果，因此我們選用了 logistic regression 來當作第二層的分類器。

```
level0 = list()
level0.append(('rf', RandomForestClassifier(**rf_girdsearch.best_params_)))
level0.append(('xgb', XGBClassifier(**xgb_girdsearch.best_params_)))
level0.append(('nb', GaussianNB(**nb_girdsearch.best_params_)))
level1 = LogisticRegression()
stack_model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)
```

● 預測結果

我們最後也使用了 voting 和 stacking 來對 testset 做預測，下圖是三種分類器和兩種 ensemble method 在 test set 上的預測分數，可以看到兩種的 ensemble method 並沒有達到最佳的結果。比起原本的三種模型，XGBoost 還是好於兩種 ensemble method。對於這種結果我們有一些猜測，可能原因有下列幾個:1. 我們 ensemble 的分類器數量不夠多，像是常見的 KNN 和 SVM 等，在我們實驗當中都沒有使用到; 2. 在 stacking 中的 meta model 部分我們使用的 LogisticRgression 是使用預設的參數，若調過參數之後可能會有較佳的結果。

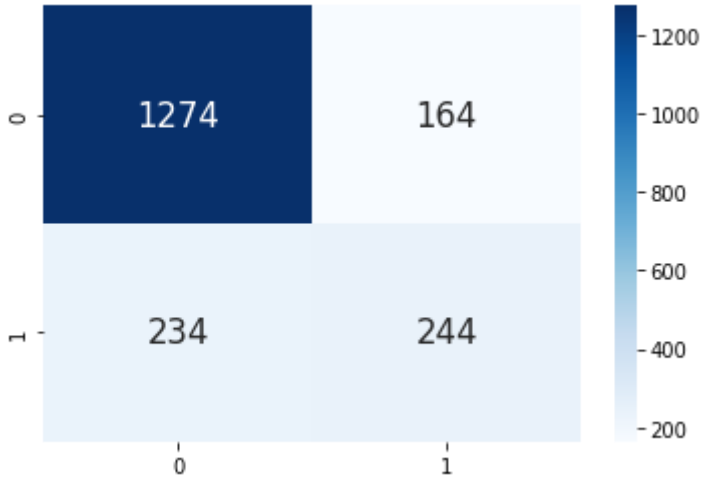


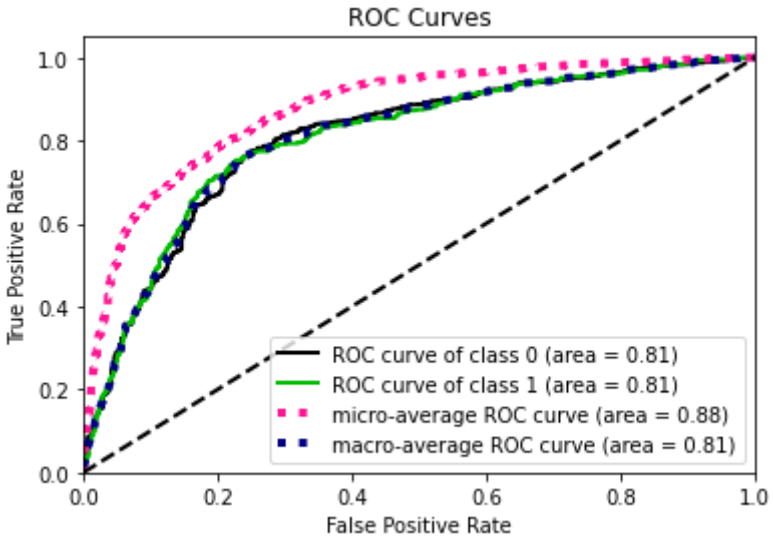
4.7. 原始三種模型表現

因採用 ensemble method 後的模型表現效果並沒有較好，相對來說只是徒增成本，因此這個階段我們僅會針對原始的三種 Random Forest, Naive bayes, XGboost 三種進行分析。以下呈現各自的 Confusion maxtrix，及各項指標。0 代表沒有意願轉職，1 代表有意願轉職。

4.7.1. XGBoost

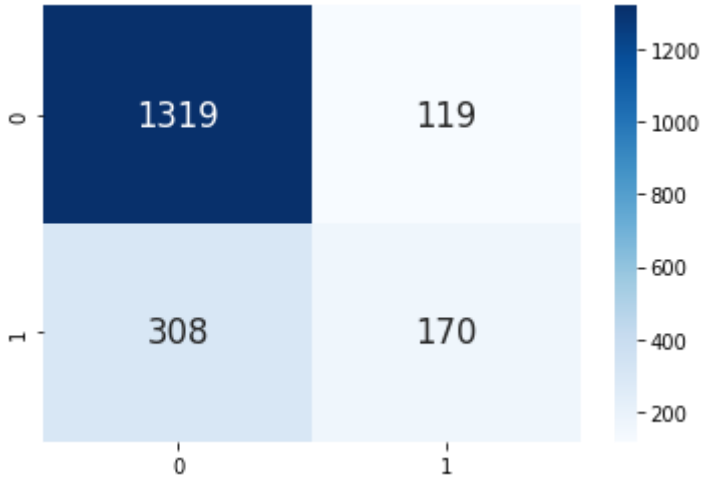
	precision	recall	f1-score	support
0	0.84	0.89	0.86	1438
1	0.60	0.51	0.55	478
accuracy			0.79	1916
macro avg	0.72	0.70	0.71	1916
weighted avg	0.78	0.79	0.79	1916

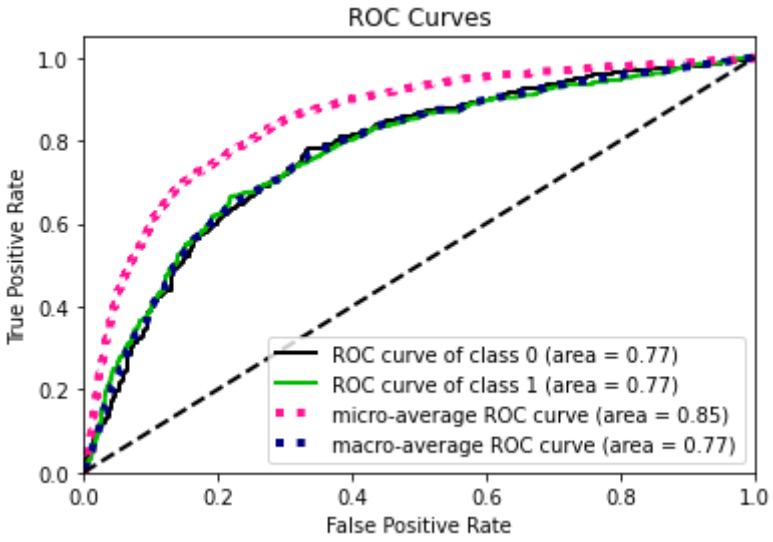




4.7.2. Naive Bayes

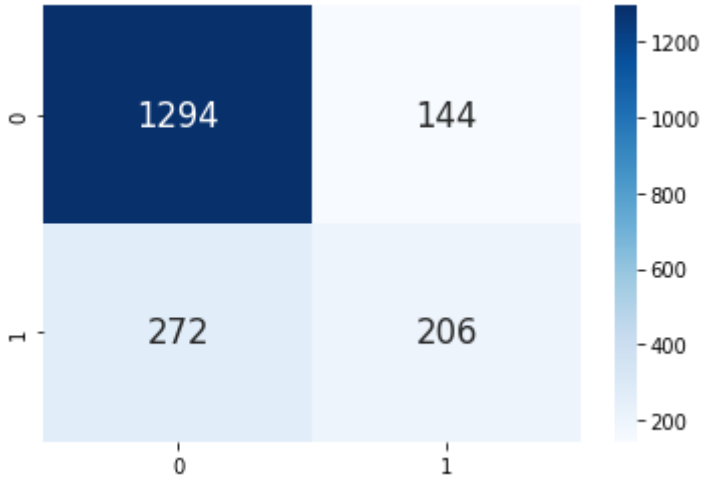
	precision	recall	f1-score	support
0	0.81	0.92	0.86	1438
1	0.59	0.36	0.44	478
accuracy			0.78	1916
macro avg	0.70	0.64	0.65	1916
weighted avg	0.76	0.78	0.76	1916

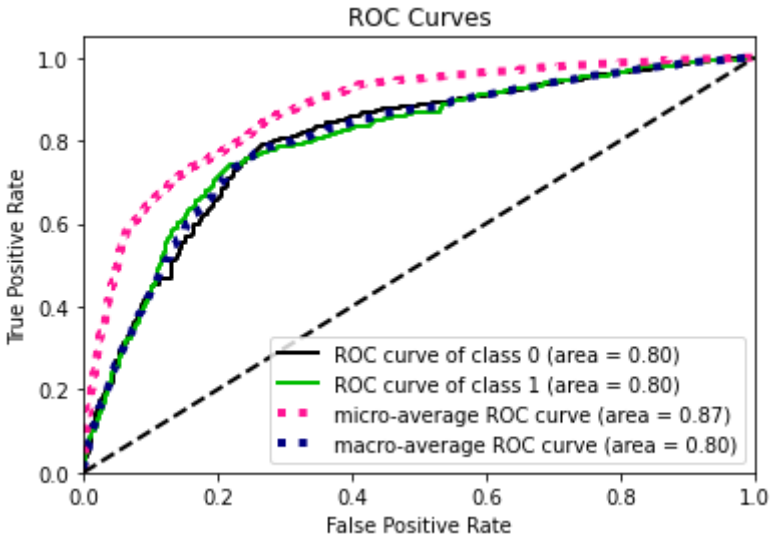




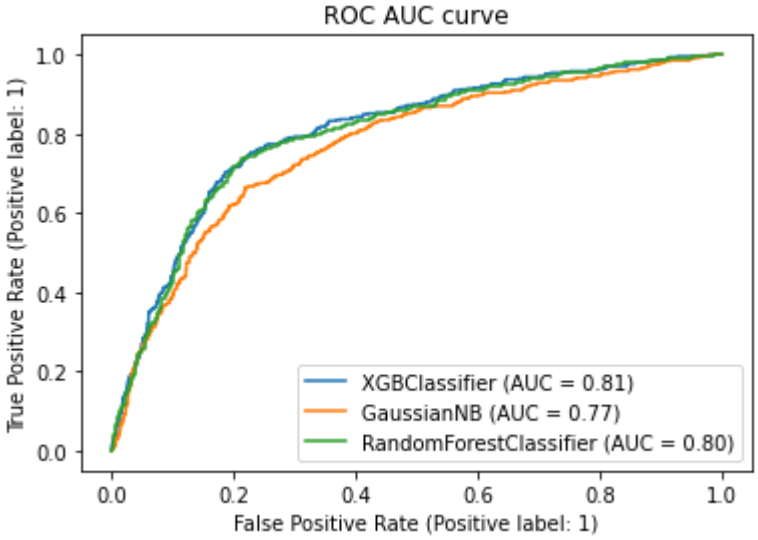
4.7.3. Random Forest

	precision	recall	f1-score	support
0	0.83	0.90	0.86	1438
1	0.59	0.43	0.50	478
accuracy			0.78	1916
macro avg	0.71	0.67	0.68	1916
weighted avg	0.77	0.78	0.77	1916





4.7.4. 三者疊合的 ROC curve



5. Evaluation

在不同的情境當中我們對於模型的評估指標會不同，像是在醫療領域中藥模型去預測病患是否得病的話我們就需要這個模型具有較高的 recall 值，這樣才能將所有可能發病的病患識別出來，相反的如果今天是在一個銀行信用評分的模型當中，我們會希望它具有較高的 precision，因為若是模型的 recall 值太高的話可能會導致許多可能原本是有心用的客戶被誤判為沒有信用的。如此一來，銀行就會損失潛在的客戶。

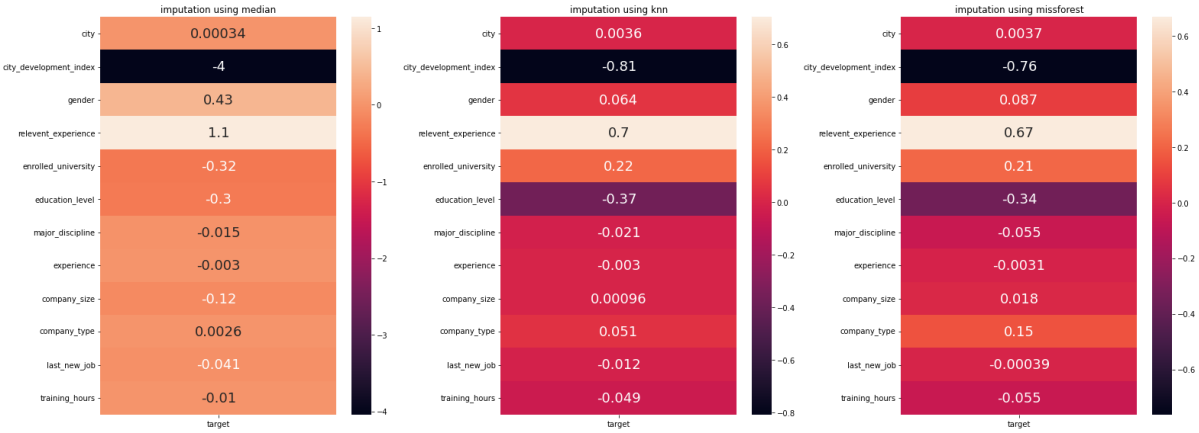
而在我們的轉職預測當中，若我們是站在雇主的角度當中，我們會希望留住優秀的人才，因此在這個情境底下我們會更傾向於一個 recall 值較高的模型。而就我們這次實驗的三種模型當中，其中表現最好的 XGBoost 它對於會轉職的人(target = 1)recall 有達到 0.5，表示可以召回近一半可能會轉職的人，此外 precision 也有 0.6。因此在這些被預測為可能會轉職的人當中，站在雇主角度的我們可以進一步的去分析這些被預測為可能轉職的人的詳細資料，像是他的教育程度，相關的經驗，訓練過的時數等等，來提前保留住優秀的人才。

而我們認為 target 0 和 target 1 的表現明顯不同的原因是原始的訓練集就屬於不平衡的，target 1 的原始資料就較少，因此判斷上的敏感度也較低。

6. Deployment

6.1. 學員資料與未來轉職關係

在最前段商業目標中有提到，除了希望能夠產生一模型來預測學員未來是否有機會轉職並進入本公司外。另外一個目標是希望能夠找出與轉職機會最相關的欄位，用以提供人資單位在一開始接受學員課程申請，就能預先篩選出較有可能真正加入本公司的人，藉此降低公司的授課成本。而使用 Logostic Regression 進行分析並繪製成 heat map 後。



可以發現具有相關工作經驗的人最有可能會實際轉職，因此人資單位在接受申請時可以考慮優先同意具有相關經驗的人。而最有可能使學員不轉職的欄位則是城市發展指數，不過若因為申請人的居住地而對其在申請時有差別待遇，可能會造成一些法律或是道德上的歧視問題，因此項資料我們選擇不納入考慮範圍中。

6.2. 最終模型部屬

最後我們所訓練出的最佳模型在測試集上的準確度可以達到 79%，這個數值對於人資單位判斷一個學員未來有沒有機會轉職應該已經可以發揮一定的參考作用。且模型在判斷上的速度也相當快，不會有太高的成本。

不過這組訓練資料其實還是存在一些問題，例如原始資料欄位其實偏少，應該還有更多資料可以蒐集來加強模型的準確度。另外一個問題則是這些資料都是在學員開始上課前預先蒐集的，如果可以加入一些課程中的資料，例如期中考核成績或是對課程的感興趣程度等，這些資料比起事前蒐集，應該能夠發揮出更好的效果。

因此除了模型之外，制度上面應該也還要再調整才有機會讓這樣的預測模型能夠表現得更好。

附錄:

參考程式碼:[商業智慧.ipynb - Colaboratory (google.com)]