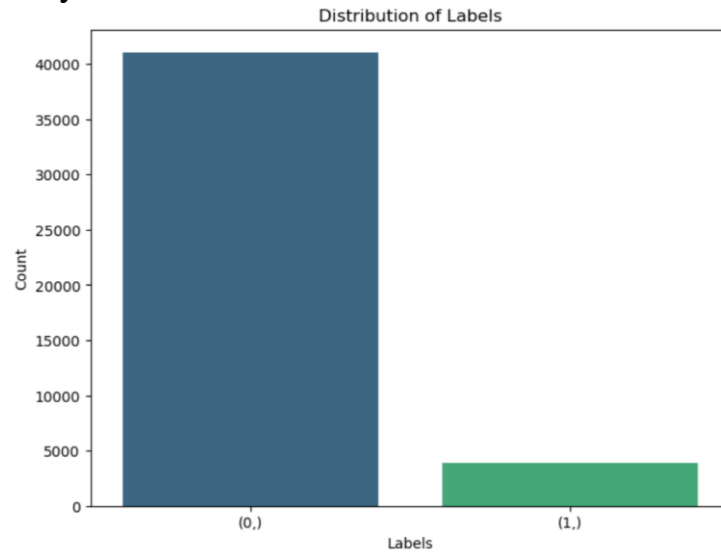


# HW2\_Report

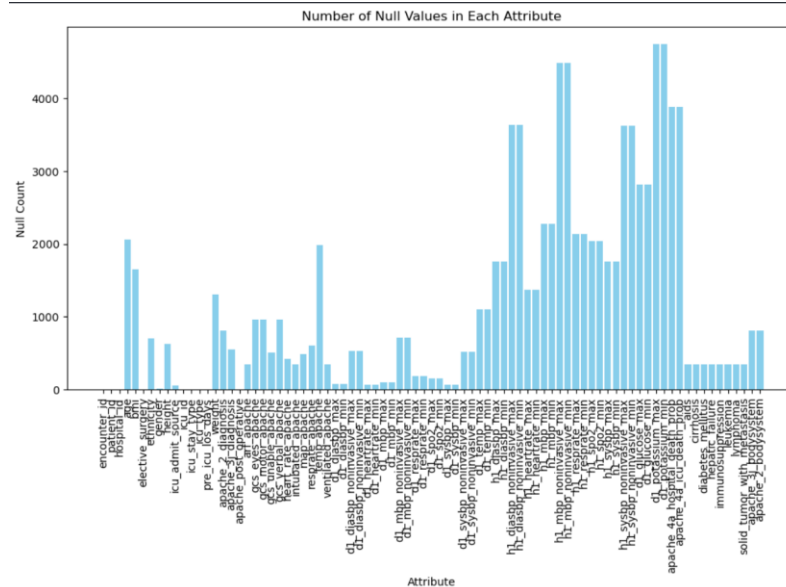
系級:智能系統 姓名:張宸瑋 學號:312581006

(1) Data pre-preprocessing and any other data-centric procedures you conducted

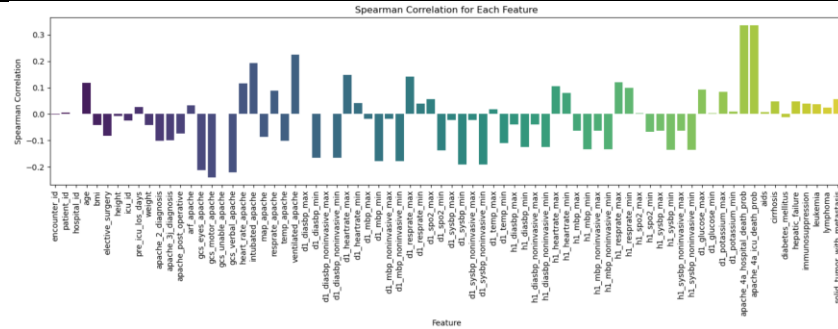
### (1) Dataset analysis



### 訓練資料標籤分佈狀況直方圖



資料缺失值統計



每個特徵對於死亡的 Spearman 相關系數直方圖



透過 Pandas Profiling 做探索性數據分析(因為頁面空間，只顯示部分結果)

## (2) Feature Selection

### 方法一：過濾式特徵選取

```
#透過pearson來計算強關聯的特徵
numeric_correlation_list = []
#print correlation length
print("原始特徵數量")
print(len(correlations))
for feature, corr_info in correlations.items():
    pearson_corr = corr_info['Pearson']
    pearson_p = corr_info['Pearson_p']

    if abs(pearson_corr) >= 0.023 and pearson_p <= 0.05:
        numeric_correlation_list.append(feature)
    else :
        print(f"{feature} is not correlated")
print("強關聯的特徵")
print(numeric_correlation_list)
print("刪減後的特徵數量")
print(len(numeric_correlation_list))
```

對於數值型的資料，透過 pearson 相關系數獲取與目標特徵為強關聯性的特徵。

```
from scipy.stats import chi2_contingency
non_numeric_columns = original_train_data.select_dtypes(exclude='number').columns
non_numeric_correlation = []
for columns in non_numeric_columns:
    contingency_table = pd.crosstab(original_train_data[columns].dropna(), original_train_label['has_died'])
    chi2, p, _, _ = chi2_contingency(contingency_table)
    print(f"Feature : {columns}")
    print(f"Chi2 : {chi2}")
    print(f"P-value : {p}")
    if(p < 0.05 and chi2 > 100):
        non_numeric_correlation.append(columns)
    else:
        print(f"{columns} Not correlated")
```

對於類別型的資料，透過卡方檢定，獲取與目標特徵之間的關聯性，判斷是否為相互獨立，抑或是有相依的關係存在。

### 方法二：嵌入式特徵選取

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=300, random_state=42)

model.fit(original_train_data, train_label)

feature_importances = model.feature_importances_

selected_features = original_train_data.columns[feature_importances > 0.003]

print(selected_features)
```

透過隨機森林能夠為每個特徵提供分數，這些分數衡量了每個特徵對於模型的性能有多大的貢獻，透過這些特徵重要性來進行特徵選取。

### 方法三：降維方法

```
#use PCA
import pandas as pd
from collections import Counter
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
pca.fit(train_data)
train_data = pca.transform(train_data)
```

透過主成分分析，將原本的特徵空間轉換為新的特徵空間，保留最重要的訊息。

實驗結果：

在我最後的實驗結果中，透過隨機森林的方式選擇特徵，獲得了最好的表現。

(3)

### (3) Data cleaning

```
#outlier detection
from sklearn.ensemble import IsolationForest

clf = IsolationForest(contamination=0.0003)
print("Original data shape: ", train_data.shape)
outliers = clf.fit_predict(train_data)
train_data = train_data[outliers == 1]
train_label = train_label[outliers == 1]
print("Clean data shape: ", train_data.shape)
```

透過孤立森林處理異常值

```
#drop the columns which is encounter_id and patient_id and hospital_id
original_train_data = original_train_data.drop(['encounter_id', 'patient_id'], axis=1)

print("Work done!")
```

在資料分析階段，我觀察到 encounter\_id 以及 patient\_id 的每個值都是唯一的(圖 1)，因此選擇把這些特徵從原始資料刪除。

```
encounter_id has unique values
patient_id has unique values
```

圖 1

### (4) Data transformation

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
#use one-hot encoding for categorical features
original_train_data = pd.get_dummies(original_train_data)
#use standardization for numeric features

numeric_columns = original_train_data.select_dtypes(include='number').columns
scaler = StandardScaler()
original_train_data[numeric_columns] = scaler.fit_transform(original_train_data[numeric_columns])

print("Work done!")
```

對於數據集中的分類特徵做 One-Hot Encoding，並且數值特徵做標準化。

## (5) Data imputation

```
import pandas as pd
from scipy.interpolate import interp1d
# find the columns which are numeric
numeric_columns = original_train_data.select_dtypes(include='number').columns

# interpolate the missing values
for column in numeric_columns:
    original_train_data[column] = original_train_data[column].interpolate(method='linear', limit_direction='both')

for column in original_train_data.columns:
    if original_train_data[column].isnull().any():
        value_counts = original_train_data[column].value_counts()
        most_frequent_value = value_counts.idxmax()
        original_train_data[column] = original_train_data[column].fillna(most_frequent_value)

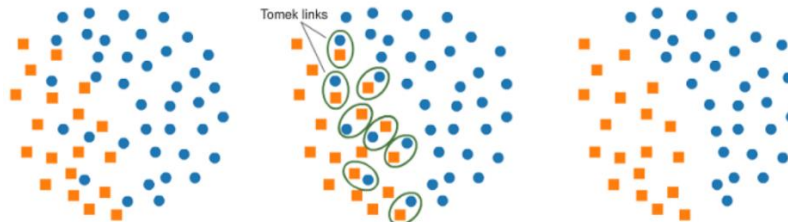
print(original_train_data.head())
```

對於數據集中的缺失值，數值特徵透過線性插值填充，並且針對分類特徵，計算該分類特徵的眾數，也就是以出現頻率最高的值進行填充。

## (6) Data imbalance handling

```
X_resampled, y_resampled = tomelink.fit_resample(X_train, y_train)
```

在資料 Imbalance 方面我透過 Tomek Links 來緩解資料不平衡的問題，其原理為如果有兩個不同類別的樣本，他們分別是彼此之間的最近鄰，也就是 A 的最近鄰是 B，B 的最近鄰是 A，那麼 A、B 就是 Tomek Links，而如果其中一個組成 Tomek Links 的點屬於多數類的樣本，那就把他刪除，示意圖如下。



Tomek Links 示意圖

討論：

### (1) discuss problems encountered

在資料分析時，我發現資料有非常嚴重的 imbalance 問題，並且有不少的特徵都有缺失值的情況。

### (2) explain how you deal with them

在資料 imbalance 方面，我嘗試了不同的演算法，像是 Tomek Links、Smote+ENN 以及 Smote+Tomek Links，來緩解資料不平衡的問題，最後的實驗結果顯示 Tomek Links 的效果最好。

而在資料缺失值方面，對於分類特徵我只採用了透過計算該分類特徵出現頻率最多的值，來填補缺失值，而在數值特徵方面，我嘗試使用眾數、線性插值、以及平均值來填補缺失值，最後實驗下來的結果是線性插值最好。

## (2) Classification Methods

- (1) Describe clearly the machine learning algorithms (give References to the used algorithms) and related packages you used.

Describe clearly the machine learning algorithms

```
# 初始化模型
model = XGBClassifier(n_estimators=2200, random_state=42, eta=0.01, max_depth=17, subsample=0.8, colsample_bytree=0.8, gamma=0.1, scale_pos_weight=10)
```

這次我使用的是XGboost演算法，全名為 eXtreme Gradient Boosting，這個演算法是目前最常見的演算法之一，它是以Gradient Boosting為基礎下去實作，並增加一些新的技巧，並且結合了Bagging跟Boosting的優點。

在XGboost中，每一棵樹是互相關聯的，目的是希望可以藉由後面生成的樹能夠修正前面一顆樹犯錯的地方，且採取了隨機採樣的技巧，和隨機森林一樣，在生成每一棵樹的時候，隨機提取特徵，因此每棵樹的生成並不是由所有的特徵參與決策，並且XGboost在目標函數中添加標準化，這樣有助於幫助模型控制複雜度，並且避免過擬合，也提高了模型的泛化能力。

並且XGboost能夠計算每個特徵的重要性分數，幫助我們能夠理解不同特徵對於分類目標特徵的貢獻程度。






```
from xgboost import XGBClassifier
```

這次的實作，我使用到了xgboost庫，並且使用了它所提供的XGBClassifier類別來完成這次的分類任務。

- (2) Describe how to reproduce the results with your source code files.

```
(datamining) C:\Users\Steven\Desktop\課程資料\交大\2023-1\Data Mining\hw\hw2\code>python main.py
Start testing...
Finish testing...
Save prediction to test_pred.csv
```

透過執行 main.py 生成預測結果

 main.py	2023/12/21 上午 12:18	Python 來源檔案	5 KB
 model.json	2023/12/20 下午 11:58	JSON 來源檔案	102,800 KB
 sample_submission.csv	2023/11/28 下午 02:31	Microsoft Excel 逗點...	173 KB
 test_pred.csv	2023/12/21 上午 12:30	Microsoft Excel 逗點...	173 KB
 test_X.csv	2023/11/28 下午 02:31	Microsoft Excel 逗點...	8,538 KB

執行完後會生成出 test\_pred.csv

```
usage: main.py [-h] [--input_file INPUT_FILE] [--output_file OUTPUT_FILE] [--submission_file SUBMISSION_FILE]
Data mining hw2

options:
  -h, --help            show this help message and exit
  --input_file INPUT_FILE, -i INPUT_FILE
                        Input file path
  --output_file OUTPUT_FILE, -o OUTPUT_FILE
                        Output file path
  --submission_file SUBMISSION_FILE, -s SUBMISSION_FILE
                        Sample submission file path
```

程式所對應的參數分別是 test\_X.csv 的位置、輸出 test\_pred.csv 的位置以及 sample\_submission.csv 的位置

	main.py	2023/12/21 上午 12:18	Python 來源檔案	5 KB
	model.json	2023/12/20 下午 11:58	JSON 來源檔案	102,800 KB
	sample_submission.csv	2023/11/28 下午 02:31	Microsoft Excel 逗點...	173 KB
	test_pred.csv	2023/12/21 上午 12:28	Microsoft Excel 逗點...	173 KB
	test_X.csv	2023/11/28 下午 02:31	Microsoft Excel 逗點...	8,538 KB

注意:要將 model.json 放到跟 main.py 同一層資料夾下

### (3) Results & Analysis

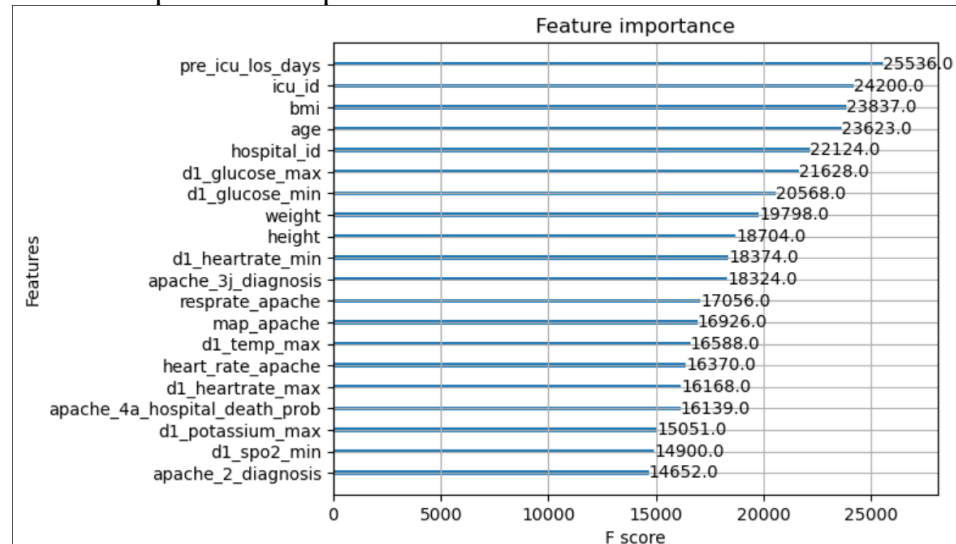
- (1) Count Average AUROC and macro F1-Score with cross-validation method.

The best result is with Average AUROC 0.8848518245330528 and macro F1-Score: 0.7290485148960764

結果截圖

- (2) Explainable experiment:

- Show the Top 20 most important features.



- Give some analysis and insights about the high-importance features.

透過上方特徵重要性的圖表中我們可以看到最重要的兩個特徵為 pre\_icu\_los\_days 以及 icd\_id，上網查詢這兩個特徵代表的意義好像是在病患進入 icu 前在醫院內的住院天數以及不同的重症監護單元標識符，我認為以這個來作為判斷依據算是有道理的，並且在前幾個重要的特徵中透過 bmi 以及 age 這種較為基本的病患資料，也能夠符合我們的想法，因為如果年紀越大，病患死亡的機率確實有可能更高，但我比較意外的是 bmi 以及 age 的特徵重要性居然能排那麼前面，因為在它們後面的特徵看起來像是病患的一些更詳細的身體特徵，透過這次的實驗我認為透過像 XGBOOST 這種具有可解釋的模型，能夠讓我們挖掘到一些我們可能沒辦法觀察到的特徵重要性，並且也可以讓我們了解到模型是如何預測，並且追蹤模型的決策過程。

Reference:

1. <https://zhuanlan.zhihu.com/p/142413825>
2. <https://medium.com/@emilykmarsh/xgboost-feature-importance-233ee27c33a4>
3. <https://zh.wikipedia.org/zh-tw/XGBoost>