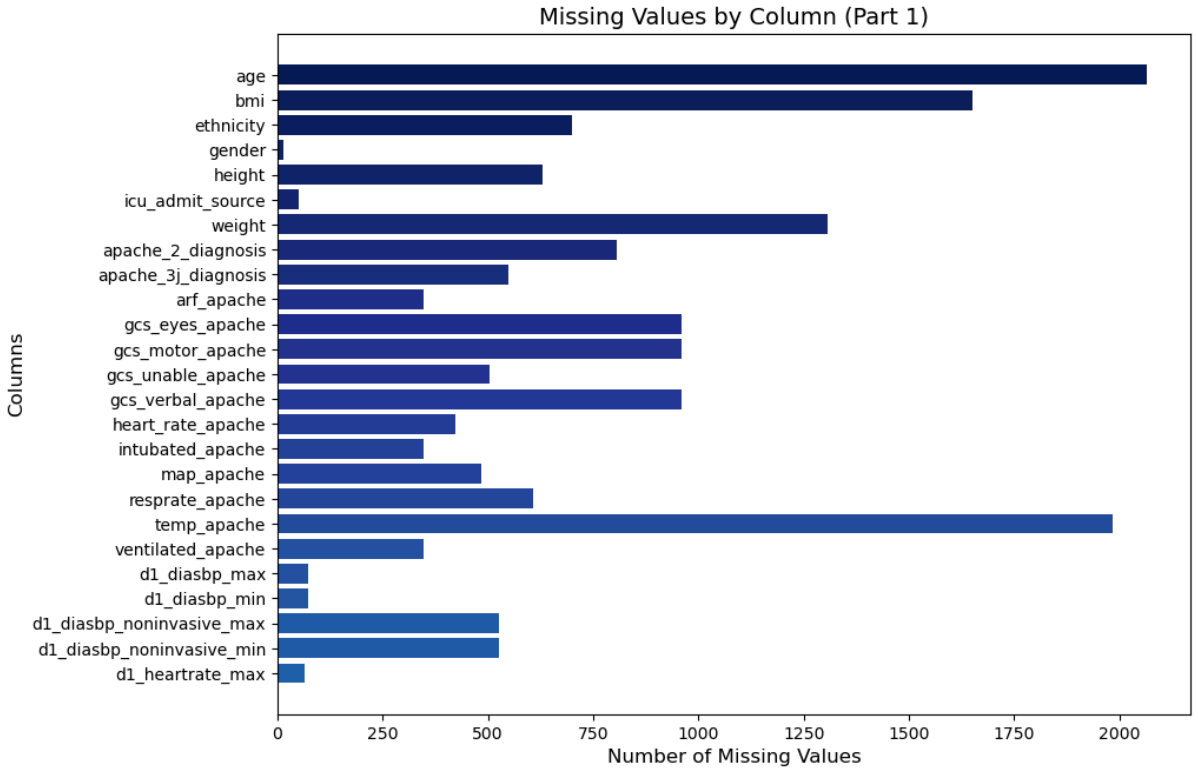
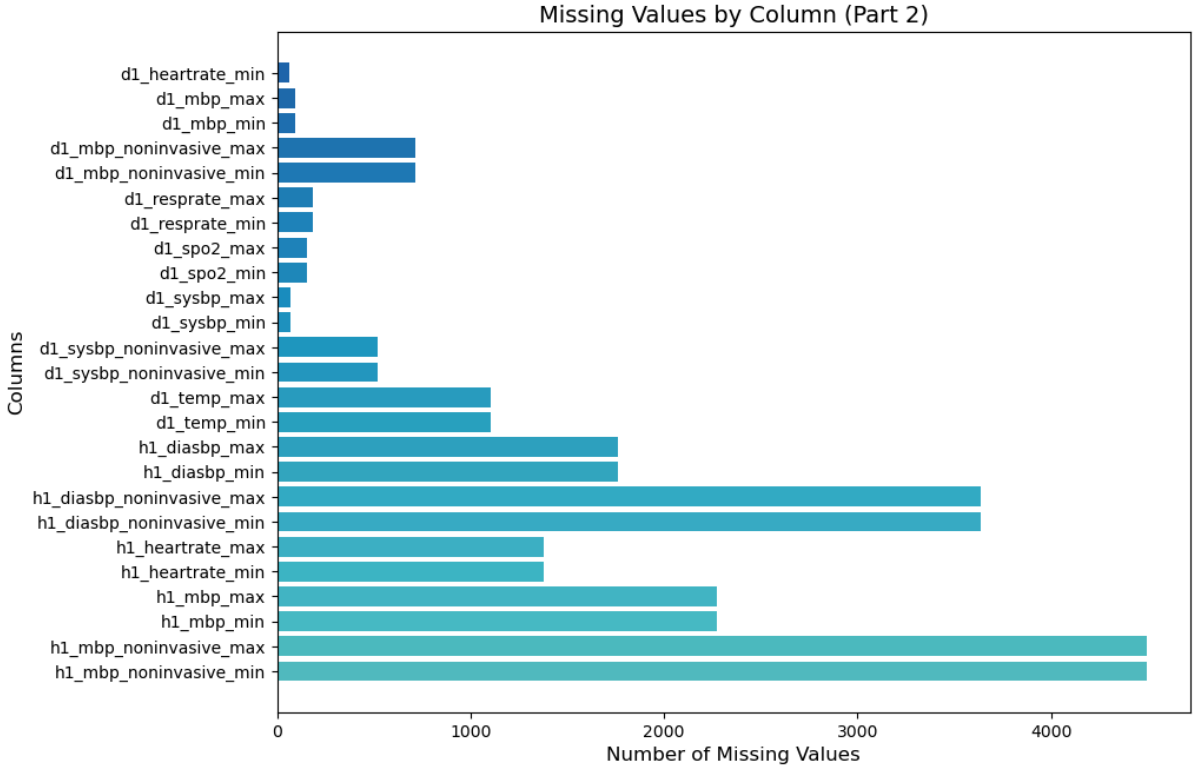
Data Mining HW2 Report 多工所碩一 313553024 蘇柏叡

1. Data-centric procedures

1.1 Introduction to our Datasets and columns

總共有train_X、test_X皆含83個欄位(dtypes: float64(70), int64(6), object(7))、train_y含目標欄位has_died(dtypes: int64(1))，其中引入資料集後發現資料集的欄位資料皆有大量缺失值，並且透過長條圖呈現個欄位缺失值情形如圖1-3





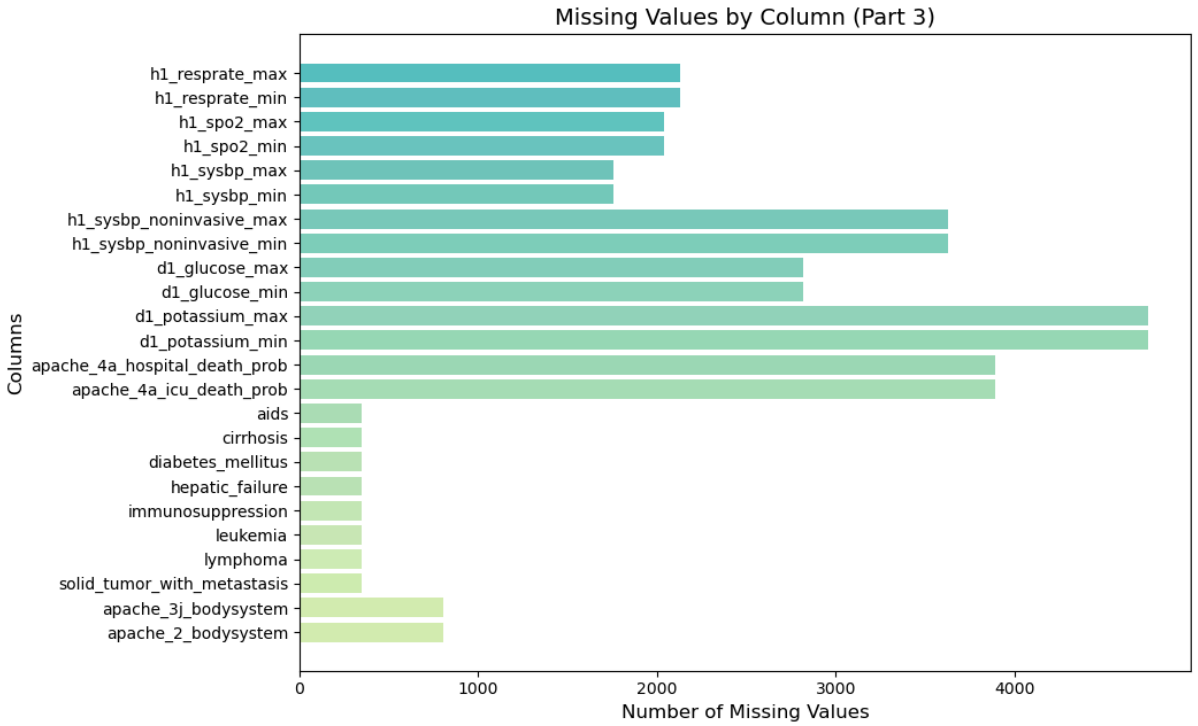


圖1-3 訓練集各個欄位缺失之情況(因篇幅故切分成三份輸出)

接著，我們將介紹各個欄位所代表之意義，以便我們後續針對資料進行較為合理之處理，此外我們特別將**id類別以及非侵入式量估標示為紅色**(因為大多數皆是對應侵入式的數值)，接著我們**將類別型的欄位(將進行Encoding)標示為綠色**，其中gender這個欄位為實施Label Encoding將Female、Male分別轉為0、1，其餘類別欄位則是採用One-Hot Encoding。至於藍色部分我們則是**參考醫學知識進行整合與分級、裝箱。**

欄位名稱	解釋	欄位名稱	解釋
encounter_id	就診編號	icu_admit_source	進入ICU的來源
patient_id	病人辨識碼	ethnicity	種族
hospital_id	醫院辨識碼	icu_stay_type	ICU住院類型
icu_id	ICU辨識碼	icu_type	ICU的類型(內外科等)
heart_rate_apache	心率(APACHE)	gender	性別
elective_surgery	是否為選擇性手術 (1 = 是，0 = 否)	apache_2_diagnosis / apache_3j_diagnosis	APACHE II / III-J 診斷碼
age	年齡	apache_post_operative	是否為術後狀況
pre_icu_los_days	在ICU之前的住院天數	arf_apache	是否有急性腎衰竭
gcs_eyes_apache	GCS 眼睛反應分數	height、weight	身高、體重
gcs_motor_apache	GCS 運動反應分數	bmi	身體質量指數

gcs_unable_apache	是否無法評估 GCS	intubated_apache	是否插管(APACHE)
gcs_verbal_apache	GCS 語言反應分數	ventilated_apache	使用呼吸器與否
d1_mbp_noninvasive_max / min	D1 非侵入性平均動脈壓**最高值/最低值**	d1_resprate_max/min	D1 呼吸速率 **最高值/最低值**
d1_sysbp_noninvasive_max / min	D1 非侵入性收縮壓**最高值/最低值**	d1_spo2_max / min	D1 血氧**最高值/最低值**
d1_diasbp_noninvasive_max/min	D1 非侵入性舒張壓**最高值/最低值**	d1_sysbp_max / min	D1 收縮壓**最高值/最低值**
h1_mbp_noninvasive_max / min	H1 非侵入性平均動脈壓**最高值/最低值**	h1_mbp_max / min	H1 平均動脈壓 **最高值/最低值**
h1_sysbp_noninvasive_max / min	H1 非侵入性收縮壓**最高值/最低值**	h1_sysbp_max / min	H1 收縮壓**最高值/最低值**
d1_heartrate_max / min	D1 心率**最高值/最低值**	d1_diasbp_max / min	D1 舒張壓**最高/最低值**
d1_mbp_max / min	D1 平均動脈壓 **最高值/最低值**	d1_temp_max / min	D1 體溫**最高值/最低值**
h1_resprate_max / min	H1 呼吸速率**最高值/最低值**	h1_diasbp_max / min	H1 非侵入性舒張壓**最高值/最低值**
h1_heartrate_max / min	H1 心率 **最高值/最低值**	h1_spo2_max / min	H1 血氧**最高值/最低值**
map_apache	平均動脈壓(APACHE)	d1_glucose_max / min	D1 血糖**最高值/最低值**
apache_4a_hospital_death_prob	APACHE IV 預測的住院死亡機率	apache_3j_bodysystem	APACHE III-J評分中的身體系統
apache_4a_icu_death_prob	APACHE IV 預測的ICU的死亡機率	apache_2_bodysystem	APACHE II評分中的身體系統
aids	是否罹患愛滋病	lymphoma	是否罹患有淋巴瘤
cirrhosis	是否罹患肝硬化	leukemia	是否罹患有白血病
solid_tumor_with_metastasis	是否有轉移性實體瘤	immunosuppression	是否罹患免疫抑制
hepatic_failure	是否罹患有肝功能衰竭	diabetes_mellitus	是否罹患有糖尿病
h1_diasbp_noninvasive_max/min	H1 非侵入性舒張壓**最高值/最低值**		

1.2 Data Preprocessing

1.2.1 Some Observations on Datasets and Some operations before Data Cleaning and Imputation (呼應1.1紅色欄位)

首先，在資料集中我們可以發現以圖4來說，欄位之間的配對為[1, 3]、[2, 4]、[7, 9]、[8, 10]是近乎一模一樣的，至於圖5而言則是[1, 3]、[2, 4]、[7, 9]、[8, 10]也是近乎一模一樣的。至於圖6則是[1, 3]、[2, 4]、[9, 11]、[10, 12]配對，總共有12個欄位是近乎重疊的，約僅有缺失值或者是少數筆有所不同，考量資料完整性以及之後在篩選欄位時不會篩選到過多自相關的特徵，因此我們**僅保留非non-invasive的欄位，換言之，我們會將non-invasive的欄位 drop掉。**

d1_diasbp_	d1_diasbp_	d1_diasbp_	d1_diasbp_	d1_heartra	d1_heartra	d1_mbp_m	d1_mbp_m	d1_mbp_n	d1_mbp_n
103	41	103	41	177	64	107	55	107	55
136	16			126	70	184	55		
84	65	84	65	175	91	103	85	103	85
85	28	85	28	108	69	103	39	103	39
65	44	65	44	122	80	79	58	79	58
58	58	58	58	110	110	69	69	69	69
69	46	69	46	90	56	96	63	96	63
94	62	94	62	131	59	115	71	115	71
76	44	76	44	72	56	97	58	97	58
153	77	153	77	146	77	157	102	157	102
69	39	69	39	80	48	80	50	80	50
85	44	85	44	127	76	93	59	93	59

d1_sysbp_	d1_sysbp_	d1_sysbp_	d1_sysbp_	d1_temp_m	d1_temp_m	h1_diasbp_	h1_diasbp_	h1_diasbp_	h1_diasbp_
155	76	155	76	37	36.6	58	48	58	48
232	75			39	35.9	50	30		
147	116	147	116	37.3	36.9	80	68	80	68
166	70	166	70	37.6	36.9	60	28	60	28
140	90	140	90	38.3	37.4	52	52	52	52
92	92	92	92	38.1	36.1	90	60		
141	76	141	76	38.2	36.1	53	46	53	46
166	41	166	41.03	37.7	36.7	87	87	87	87
145	78	145	78	37.4	36.9	63	52	63	52
187	127	187	127	37.7	36.6	81	81	81	81

h1_mbp_m	h1_mbp_m	h1_mbp_n	h1_mbp_n	h1_resprat	h1_resprat	h1_spo2_m	h1_spo2_m	h1_sysbp_	h1_sysbp_	h1_sysbp_	h1_sysbp_
74	55	74	55	25	18	100	99	121	76	121	76
87	66			16	16	100	100	134	105		
102	98	102	98	31	20	99	95	139	133	139	133
67	39	67	39	18	13	100	100	94	70	94	70
61	61	61	61	21	15	95	81	90	90	90	90
114	78			20	19	100	99	166	122		
89	67	89	67	13	12	100	95	97	76	97	76
				26	26	97	97	143	143	143	143
92	63	92	63	19	12	98	97	113	92	113	92
110	110	110	110	24	24	91	91	152	152	152	152

圖4-6 為大多重疊之欄位比較表格之節錄

此外，誠如1.1提到之id類別因為是用來辨識病患、ICU、醫院...等。具有唯一性的類別因此不必將其放入特徵作為後續分類依據，故在做資料集的插補時可以先將其drop如圖7，另將patient id保留起來以作為後續輸出csv時之欄位。

```
# 保留 patient_id 以便後續對應輸出
patient_ids = test_X['patient_id']

# Dropping some columns
'''
id class: 明顯對於預測目標無顯著助益
noninvasive: 由於大多數與侵入性做出來之數據近乎是一模一樣因此可不計
'''
columns_to_drop = [
    'encounter_id', 'patient_id', 'hospital_id', 'icu_id','d1_sysbp_noninvasive_max','d1_sysbp_noninvasive_min',
    'h1_diasbp_noninvasive_max','h1_diasbp_noninvasive_min','h1_mbp_noninvasive_max','h1_mbp_noninvasive_min',
    'h1_sysbp_noninvasive_max','h1_sysbp_noninvasive_min','d1_mbp_noninvasive_max','d1_mbp_noninvasive_min',
    'd1_diasbp_noninvasive_max','d1_diasbp_noninvasive_min',
]
train_X = train_X.drop(columns=columns_to_drop)
test_X = test_X.drop(columns=columns_to_drop, errors='ignore')
✓ 0.0s
```

圖7 為訓練、測試集drop之欄位

1.2.2 Data Cleaning and Imputation

首先，我們先查看剩下的類別型欄位，並且針對其欄位具有的值判斷要如何預處理，如圖8。接著，我們將一一分析這些類別型欄位的獨特值，並且再決定如何處理。（介紹於下一頁）

```
categorical_col = ['apache_3j_bodysystem', 'apache_2_bodysystem', 'icu_type',
                   'icu_admit_source', 'icu_stay_type', 'ethnicity','gender']

# Checking for unique values in each of the columns_to_encode in the train_X dataset.
unique_values = {col: train_X[col].unique() for col in categorical_col}

unique_values
✓ 0.0s
{'apache_3j_bodysystem': array(['Trauma', 'Cardiovascular', 'Neurological', 'Gastrointestinal',
        'Respiratory', 'Genitourinary', 'Metabolic', 'Sepsis',
        'Musculoskeletal/Skin', nan, 'Hematological', 'Gynecological'],
       dtype=object),
 'apache_2_bodysystem': array(['Trauma', 'Undefined diagnoses', 'Neurologic', 'Gastrointestinal',
        'Cardiovascular', 'Respiratory', 'Renal/Genitourinary',
        'Metabolic', 'Undefined Diagnoses', nan, 'Haematologic'],
       dtype=object),
 'icu_type': array(['Neuro ICU', 'CTICU', 'Med-Surg ICU', 'SICU', 'MICU', 'CSICU',
        'CCU-CTICU', 'Cardiac ICU'], dtype=object),
 'icu_admit_source': array(['Accident & Emergency', 'Operating Room / Recovery', 'Floor',
        'Other ICU', 'Other Hospital', nan], dtype=object),
 'icu_stay_type': array(['admit', 'transfer', 'readmit'], dtype=object),
 'ethnicity': array(['Asian', 'Caucasian', 'African American', 'Hispanic',
        'Other/Unknown', nan, 'Native American'], dtype=object),
 'gender': array(['M', 'F', nan], dtype=object)}
```

圖8 為類別型欄位中各種獨特值

1. apache_3j_bodysystem: 欄位中具有許多類別，不過也有包含空值需要填補。
2. apache_2_bodysystem: 欄位中依然具有許多類別，此外我們可發現有Undefined Diagnoses、Undefined diagnoses這兩個僅差在大小寫的欄位，此外依然有空值需要填補。
3. icu_type: 具有多種類別，不過並不具有空值，故不必填補。
4. icu_admit_source: 欄位中具有許多類別，不過也有包含空值需要填補。
5. icu_stay_type: 具有三種類別，不過因為為無序，因此仍是使用讀熱編碼。
6. ethnicity:除了具有空值外，也有含Other/Unknown
7. gender: 將缺失值使用最頻繁項做填補，並且使用label encoding。

綜合上述所述，我們決定僅針對gender做**Label encoding(F:0；M:1)**，至於缺失值部分**apache_3j_bodysystem、apache_2_bodysystem、icu_admit_source則是將其視為Unknown**，此外關於apache_2_bodysystem中Undefined Diagnoses、Undefined diagnoses的問題則是使用Unknown取代之。另外ethnicity則是將缺失值、Other/Unknown統一視為Unknown。程式碼與結果如**圖9。**

```
# Replace NaN and Undefined values with "Unknown" in the specified columns
train_X['icu_admit_source'].replace({np.nan: 'Unknown'}, inplace=True)
train_X['apache_3j_bodysystem'].replace({np.nan: 'Unknown'}, inplace=True)
train_X['apache_2_bodysystem'].replace({np.nan: 'Unknown', 'Undefined Diagnoses':
                                        'Unknown', 'Undefined diagnoses': 'Unknown'}, inplace=True)
train_X['ethnicity'].replace({np.nan: 'Unknown', 'Other/Unknown': 'Unknown'}, inplace=True)

test_X['icu_admit_source'].replace({np.nan: 'Unknown'}, inplace=True)
test_X['apache_3j_bodysystem'].replace({np.nan: 'Unknown'}, inplace=True)
test_X['apache_2_bodysystem'].replace({np.nan: 'Unknown', 'Undefined Diagnoses':
                                       'Unknown', 'Undefined diagnoses': 'Unknown'}, inplace=True)
test_X['ethnicity'].replace({np.nan: 'Unknown', 'Other/Unknown': 'Unknown'}, inplace=True)

unknown_counts_train = train_X[['icu_admit_source','apache_3j_bodysystem',
                                'apache_2_bodysystem','ethnicity']].apply(lambda col: col.value_counts().get('Unknown', 0))
unknown_counts_test = test_X[['icu_admit_source','apache_3j_bodysystem',
                              'apache_2_bodysystem','ethnicity']].apply(lambda col: col.value_counts().get('Unknown', 0))
# Display the results
print("Unknown counts in train set:")
print(unknown_counts_train)

print("\nUnknown counts in test set:")
print(unknown_counts_test)
```

✓ 0.0s

```
Unknown counts in train set:
icu_admit_source          51
apache_3j_bodysystem     807
apache_2_bodysystem     2812
ethnicity               2819
dtype: int64

Unknown counts in test set:
icu_admit_source          31
apache_3j_bodysystem     374
apache_2_bodysystem     1231
ethnicity               1201
dtype: int64
```

圖9 為處理類別型資料之程式碼

接著，我們針對數值型的資料分別採取**median**、**KNN插補(透過1.2.4章節更深入瞭解資料後再使用KNN因此獨立一小章節1.2.5進行介紹)並各自使用模型進行比較(後續1.1藍色部分將參考醫學知識進行整合與分級、裝箱，並且會於下一章節1.2.3 Data Transform中介紹)**，並且針對上述類別型欄位採取洽當之Encoding技術。在圖10中，我們先將資料分成數值、類別資料，數值型使用中位數進行填補，類別型除了將部分欄位值改為Unknown之外，也在gender部分採取最頻繁項集進行填補。此外，**為了遵守測試集不外洩之原則**，若是測試集具有缺失值，**一律採用訓練集該欄位的中位數、最頻繁項，以達到公平原則**。因此，在圖10、11可以看出我們的程式碼嚴謹遵守此原則，**使用median做填補後再transform到測試集**。接著，在圖11時首先先針對剩下仍保有缺失值的gender使用最頻繁項做填補，並且把類別型、數值型資料拼接起來，接著使用pd.get_dummies放入填補後的資料集、指定需要做One-Hot Encoding(符合該項給1，不符合該項就是0)的欄位，接著把經過獨熱編碼後的欄位放入imputed後的資料集(含訓練、測試)

```
import pandas as pd
from sklearn.impute import SimpleImputer
import numpy as np
'''
濾除掉一些欄位後，共剩下67欄位，其中numeric為60、categorical為7
numeric -> 缺失值使用 median做填補
categorical -> 缺失值使用最常出現項進行填補
另外後續再針對一些numeric可以用知識、專業方式進行分群、裝箱。
categorical則是視情況進行one hot(無序，例如:icu type)或者label encoding(例如:gender)。
'''
# Separate numeric and categorical columns
numeric_columns = train_X.select_dtypes(include=['number']).columns
categorical_columns = train_X.select_dtypes(include=['object']).columns

# Impute numeric features using median
numeric_imputer = SimpleImputer(strategy='median')
train_X_numeric_imputed = pd.DataFrame(numeric_imputer.fit_transform(train_X[numeric_columns]), columns=numeric_columns)
test_X_numeric_imputed = pd.DataFrame(numeric_imputer.transform(test_X[numeric_columns]), columns=numeric_columns)
```

圖10 使用中位數處理數值型欄位資料缺失值

```
# Impute categorical features using most frequent value->剩下gender
categorical_imputer = SimpleImputer(strategy='most_frequent')
train_X_categorical_imputed = pd.DataFrame(categorical_imputer.fit_transform(train_X[categorical_columns]),
                                           columns=categorical_columns)
test_X_categorical_imputed = pd.DataFrame(categorical_imputer.transform(test_X[categorical_columns]),
                                          columns=categorical_columns)

# Combine imputed numeric and categorical features
train_X_imputed = pd.concat([train_X_numeric_imputed, train_X_categorical_imputed], axis=1)
test_X_imputed = pd.concat([test_X_numeric_imputed, test_X_categorical_imputed], axis=1)

# Columns to apply one-hot encoding
columns_to_encode = ['apache_3j_bodysystem', 'apache_2_bodysystem', 'icu_type',
                     'icu_admit_source','icu_stay_type','ethnicity']

# Apply one-hot encoding on data with missing values handled
train_X_encoded = pd.get_dummies(train_X_imputed, columns=columns_to_encode, dtype=np.int64)
test_X_encoded = pd.get_dummies(test_X_imputed, columns=columns_to_encode, dtype=np.int64)

# Ensure training and test sets have the same columns
all_columns = pd.concat([train_X_encoded, test_X_encoded], axis=0).columns.unique()
train_X_imputed = train_X_encoded.reindex(columns=all_columns, fill_value=0)
test_X_imputed = test_X_encoded.reindex(columns=all_columns, fill_value=0)
```

圖11 使用最頻繁項填補性別欄位，並且針對其他類別型欄位做獨熱編碼

接著，我們查看插補後的資料型態發現gender仍尚未進行轉換，因此將訓練、測試集中的gender若是為Female則是apply 0，反之若是為M則是apply 1，最終如圖12可發現資料欄位擴增至105個其中全數轉換為數值型的資料。

```
# 將 gender 特徵轉換為數值類型（Female = 0, Male = 1）
train_X_imputed['gender'] = train_X_imputed['gender'].apply(lambda x: 1 if x == 'M' else 0)
test_X_imputed['gender'] = test_X_imputed['gender'].apply(lambda x: 1 if x == 'M' else 0)
```

✓ 0.0s

```
train_X_imputed.info()
test_X_imputed.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44939 entries, 0 to 44938
Columns: 105 entries, age to ethnicity_Unknown
dtypes: float64(60), int64(45)
memory usage: 36.0 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19260 entries, 0 to 19259
Columns: 105 entries, age to ethnicity_Unknown
dtypes: float64(60), int64(45)
memory usage: 15.4 MB
```

圖12 為轉換gender data為0、1之後的資料型態

1.2.3 Data Transformation and Visualization

(1) breathe_combine:

考量呼吸與插管於訓練集比例約各佔15、33%為較少數，考量呼吸與插管為加護病房中常見之維生器材，且插管治療的嚴重度普遍較高，因此增設此欄位欲綜合兩者之特性並且提供重要性進行區隔。此外，我們在輸出各值的比率結果中發現理論上**取值會是0、1、2、3皆存在，然而並未存在2。**換言之，**僅使用插管但不使用呼吸器的個案是不存在的。**不過，我們結合這兩種特性生成的欄位依然會保留下來，以供後續做**Feature Selection**時使用。

```
'''
建立新的欄位 'br_combine' 為 intubated_apache 和 ventilated_apache 綜合評估
插管、氧氣置有4種可能組合，原始各分兩個欄位，想嘗試結合兩者特性形成全新之欄位，並且配發新的權重。
考量需要插管在臨床上的嚴重程度較氧氣置嚴重些，因此權重則是設定為插管:呼吸器2:1
但當人的發現是竟然沒有人是有插管但沒有呼吸器
'''
train_X_imputed['breathe_combine'] = train_X_imputed['intubated_apache']*2 + train_X_imputed['ventilated_apache']
test_X_imputed['breathe_combine'] = test_X_imputed['intubated_apache']*2 + test_X_imputed['ventilated_apache']

# train_X_imputed['intubated_apache'].value_counts(normalize=True)
# train_X_imputed['ventilated_apache'].value_counts(normalize=True)
train_X_imputed['breathe_combine'].value_counts(normalize=True)
```

✓ 0.0s

```
breathe_combine
0.0    0.674715
1.0    0.174347
3.0    0.150938
Name: proportion, dtype: float64
```

圖13 breathe_combine欄位為結合intubated_apache 和 ventilated_apache 綜合評估

(2) gcs_overall、gcs_severity:

由於本資料集把**格拉斯哥昏迷量表之三項分開存取**，並且**gcs_unable_apache**用以判定是否可評估，換言之**gcs_unable_apache**為其他三項之補集，若是其他三項有值，則此項不存在。考量先前已經有把各欄位缺失值使用median進行填補了，因此可以將該欄位drop掉。另外，參考[1]將三項分數加總，若是**格拉斯哥昏迷指數為3-8分者為重度昏迷，9-12分則是中度昏迷，13-15分則是輕度昏迷。**此舉措為為避免在特徵篩選時因為拆分為三個欄位(搶佔其他欄位)導致犧牲掉從其他欄位特徵學習的機會。不過從定義來看我們可發現輕度昏迷者之比例大約為75%，剩下則是中度及重度的比例大約為25%，此外三項皆滿分者也是高達近6成。**不過仍可有效將多欄位資料濃縮。**

```python
import pandas as pd
'''
原始資料集中將gcs分為四個欄位: 'gcs_eyes_apache'、'gcs_motor_apache'、'gcs_verbal_apache'、'gcs_unable_apache'
然而最後一個欄位則是和前三者成為補集，因為其概念是無法評估gcs各項分數會在該格填入1，若可以評估則是0，因此查找相關資料佐證
則是使用了以下裝箱基準
'gcs_eyes_apache'、'gcs_motor_apache'、'gcs_verbal_apache'分別為E4V5M6(即是每項最低1分，最高為4 or 5 or 6)
[12, 15]: 1 ; [8, 12]: 2 ; [3, 8]: 3
'''
# 對訓練集計算 gcs_overall 和 gcs_severity
train_X_imputed['gcs_overall'] = train_X_imputed['gcs_eyes_apache'] + train_X_imputed['gcs_motor_apache'] + train_X_imputed['gcs_verbal_apache']
train_X_imputed['gcs_severity'] = pd.cut(
    train_X_imputed['gcs_overall'],
    bins=[2, 8, 12, 15],
    labels=[3, 2, 1],
    right=True
).astype(int)
train_X_imputed = train_X_imputed.drop(columns=['gcs_eyes_apache', 'gcs_motor_apache', 'gcs_verbal_apache'])

# 對測試集進行相同的處理
test_X_imputed['gcs_overall'] = test_X_imputed['gcs_eyes_apache'] + test_X_imputed['gcs_motor_apache'] + test_X_imputed['gcs_verbal_apache']
test_X_imputed['gcs_severity'] = pd.cut(
    test_X_imputed['gcs_overall'],
    bins=[2, 8, 12, 15],
    labels=[3, 2, 1],
    right=True
).astype(int)
test_X_imputed = test_X_imputed.drop(columns=['gcs_eyes_apache', 'gcs_motor_apache', 'gcs_verbal_apache'])

#由於先前就已經填補過資料了，因此'gcs_unable_apache'就沒有存在的必要性
train_X_imputed = train_X_imputed.drop(columns=['gcs_unable_apache'])
test_X_imputed = test_X_imputed.drop(columns=['gcs_unable_apache'])
train_X_imputed['gcs_overall'].value_counts(normalize=True)
```

圖14使用gcs_overall、gcs_severity針對三項加總並且依據輕度、中度、重度給予1-3分

gcs_overall	
15.0	0.587329
14.0	0.117915
3.0	0.056988
13.0	0.047865
10.0	0.033668
11.0	0.028861
12.0	0.028016
7.0	0.025523
8.0	0.023899
9.0	0.022720
6.0	0.018247
5.0	0.004517
4.0	0.004450

```
train_X_imputed['gcs_severity'].value_counts(normalize=True)
✓ 0.0s

gcs_severity
1    0.753110
3    0.133626
2    0.113265
Name: proportion, dtype: float64
```

圖15、16分別為GCS三項加總分數比例、輕度中度重度之分級比例

(3) chronic factors:

考量慢性病以及相關腫瘤疾病之樣本數量過少(例如愛滋病僅有32位)，因此想結合這幾項作為全新欄位(試圖綜合多項特徵以增加能夠讓模型學到較多元之特徵) 由於並無相關資料佐證何種疾病造成致死率較高，因此暫且將其權重設為相同，目的是希望能夠學習到有用之綜合特徵。

```
'''
考量慢性病以及相關腫瘤疾病之樣本數量過少，因此想結合這幾項作為全新欄位
由於並無相關資料佐證何種疾病造成致死率較高，因此暫且將其權重設為相同，目的是希望能夠學習到有用之綜合特徵
'''
chronic_columns = [
    'aids', 'cirrhosis', 'diabetes_mellitus', 'hepatic_failure',
    'immunosuppression', 'leukemia', 'lymphoma', 'solid_tumor_with_metastasis'
]

# 計算這些欄位的和，並存入新的欄位 'chronic_factor'
train_X_imputed['chronic_factor'] = train_X_imputed[chronic_columns].sum(axis=1)
test_X_imputed['chronic_factor'] = test_X_imputed[chronic_columns].sum(axis=1)

# 查看結果
train_X_imputed['chronic_factor'].value_counts(normalize=True)
```

✓ 0.0s

```
chronic_factor
0.0    0.726229
1.0    0.244442
2.0    0.024522
3.0    0.004539
4.0    0.000267
Name: proportion, dtype: float64
```

圖17 chronic factors為綜合8項慢性病之新欄位

(4) spo2_level:

根據相關研究指出血氧濃度最低的正常值約坐落在95+，若是低於90則是需要氧氣治療，低於80則會有器官損壞之問題。因此，設定以下標準[95, 100]: 1; [90, 95]: 2; [80, 90]: 3; [0, 80]: 4，不過其中我們可看出<80%的比例仍是極少。

```
'''
定義裝箱類別:
血氧最低的正常值約坐落在95+，若是低於90則是需要氧氣治療，低於80則會有器官損壞之問題。
因此，設定以下標準
[95, 100]: 1; [90, 95]: 2; [80, 90]: 3; [0, 80]: 4
train 、 test都做
'''
train_X_imputed['spo2_level'] = pd.cut(
    train_X_imputed['d1_spo2_min'],
    bins=[-float('inf'), 80, 90, 95, float('inf')],
    labels=[4, 3, 2, 1],  # 分別代表不同的 SPO2 等級
    right=False  # 左閉右開區間
).astype(int)

# 如果有測試集也需要進行相同操作
test_X_imputed['spo2_level'] = pd.cut(
    test_X_imputed['d1_spo2_min'],
    bins=[-float('inf'), 80, 90, 95, float('inf')],
    labels=[4, 3, 2, 1],
    right=False
).astype(int)

train_X_imputed['spo2_level'].value_counts(normalize=True)
```

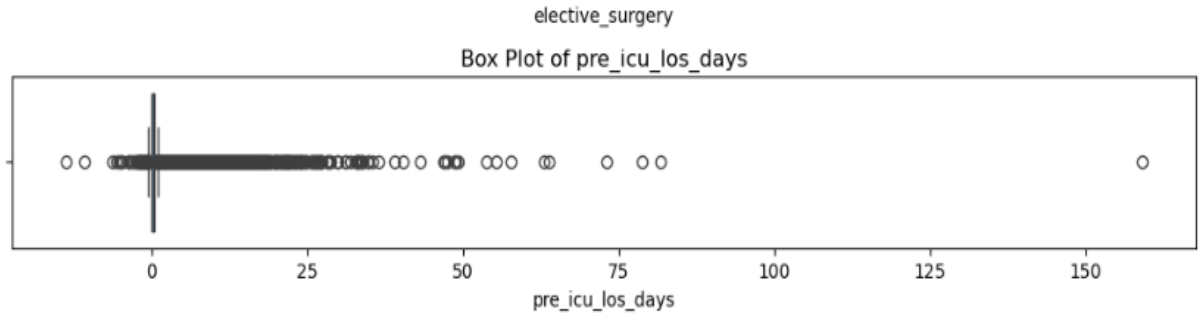
✓ 0.0s

```
spo2_level
2    0.440998
1    0.307706
3    0.187165
4    0.064131
Name: proportion, dtype: float64
```

圖18 spo2_level為將血氧濃度最低值進行分類裝箱

(5) icu_stay_duration:

由於從盒鬚圖中可發現大多數資料值皆是位於0-1之間，且有一些奇怪現象就是包含不少負值，因此決定採用裝箱法，將小於0的異常值分類在未提前住院(0)，若是住院未滿一天者則是(1)，另外住院未滿一周者則是(2)，最終住院大於一周者則是(3)。而我們將輸出後的bin列出發現，訓練集中大多數患者是住院未滿一天的(近8成)




```
'''
定義裝箱類別:
0以下為並未提前住院、0-1則表示未住滿一天(其體溫其實就是d1_temp)、1-7則是住院一周內、其他則是住院超過一星期
train 、 test都做
'''
train_X_imputed['icu_stay_duration'] = pd.cut(
    train_X_imputed['pre_icu_los_days'],
    bins=[-float('inf'),0, 1, 7, float('inf')],
    labels=[0, 1, 2, 3],
    right=True
).astype(int)


test_X_imputed['icu_stay_duration'] = pd.cut(
    test_X_imputed['pre_icu_los_days'],
    bins=[-float('inf'),0, 1, 7, float('inf')],
    labels=[0, 1, 2, 3],
    right=True
).astype(int)

train_X_imputed['icu_stay_duration'].value_counts(normalize=True)
```

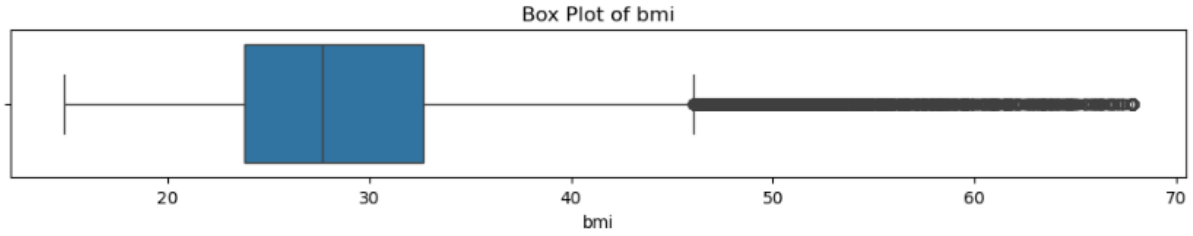
✓ 0.0s

```
icu_stay_duration
1    0.786043
2    0.141592
0    0.048288
3    0.024077
Name: proportion, dtype: float64
```

圖19、20分別為盒鬚圖以及針對pre_icu_los_days做裝箱後之結果

(6) bmi_category:

由盒鬚圖可發現，本資料的bmi具有大量遠超過50者(outliers皆是位於過胖)，因此參考國民健康署對於相關bmi值相關之定義，並且並非BMI值越高就越有健康隱憂，而是考量過瘦可能比過重帶有更大之健康隱憂，**因此將正常的18.5~24區間設為1，24-27設為2，18.5以下設為3(意即較危險)，而27-30則是設定為更危險的4，30-35則是5，至於重度肥胖患者則是將標準設為高於35(5)。**由分布可看出本資料集中，**過瘦的比例是較少的(僅約4%左右)**。其餘5個bin分布較為平均約坐落在18%附近。




```python
# 分箱並標記各 BMI 區間
train_X_imputed['bmi_category'] = pd.cut(
    train_X_imputed['bmi'],
    bins=[-float('inf'),18.5, 24, 27, 30, 35, float('inf')],
    labels=[3, 1, 2, 4, 5, 6],  # 設定標籤數字，其中較高數字代表風險較高
    right=False  # 左閉右開
)

# 測試集同樣處理
test_X_imputed['bmi_category'] = pd.cut(
    test_X_imputed['bmi'],
    bins=[-float('inf'),18.5, 24, 27, 30, 35, float('inf')],
    labels=[3, 1, 2, 4, 5, 6],
    right=False
)
# 查看結果
print("Train set bmi_category summary:")
print(train_X_imputed['bmi_category'].value_counts(normalize=True))
```

✓ 0.0s

```
Train set bmi_category summary:
bmi_category
1    0.221211
4    0.199025
5    0.181735
6    0.179109
2    0.178998
3    0.039921
Name: proportion, dtype: float64
```

圖21、22分別為盒鬚圖以及針對bmi做裝箱後之結果

(7) 新增之欄位與target之相關性:

左起為

'gcs_overall', 'gcs_severity','chronic_factor','d1_spo2_min',
'spo2_level','pre_icu_los_days', 'icu_stay_duration','bmi','bmi_category'
target: has_died

我們可以從圖23了解到經過我們處理裝箱後的相關性普遍是沒有明顯上升的甚至有微幅下滑，不過使用線性的方式把三項GCS評估項相加後，其相關係數其實是差不多的皆是-0.27，而慢性疾病因子項則是經過統合後算出來相關係數為0.042。然而，相關係數僅能代表該欄位和目標項的相關性，並不全然地代表abs(相關係數)越大，就代表其欄位對於prediction的貢獻就越大。

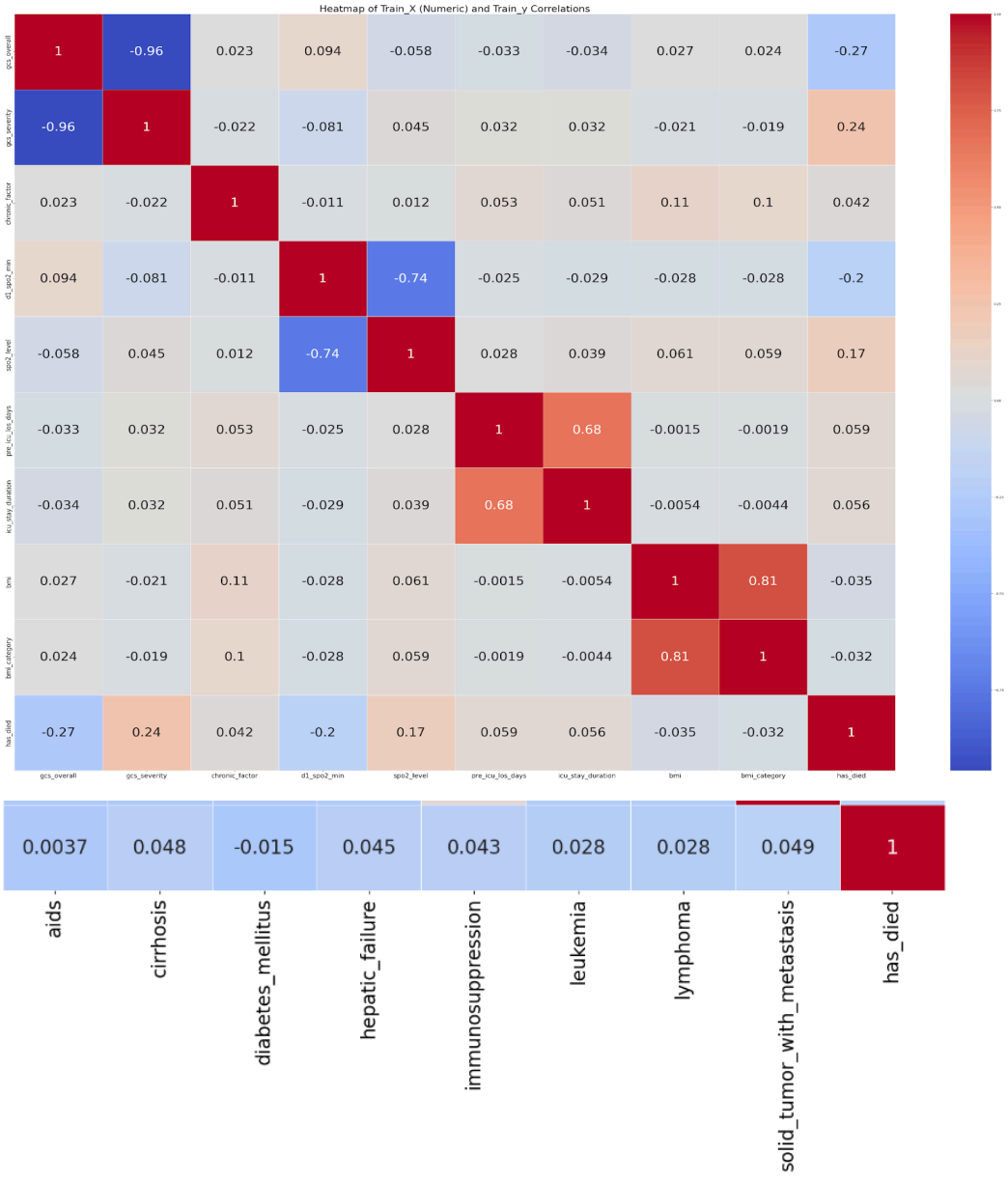


圖23 為新增之欄位及部分原先欄位與目標欄位之相關性矩陣

1.2.4 Other Columns Visualization and Outliers or Anomalies Analysis

(1) apache_3j_diagnosis

雖然由此盒鬚圖可能會發現超過1500的資料視為離群值，然而其實這個欄位是診斷碼，因此並不能視為離群值。

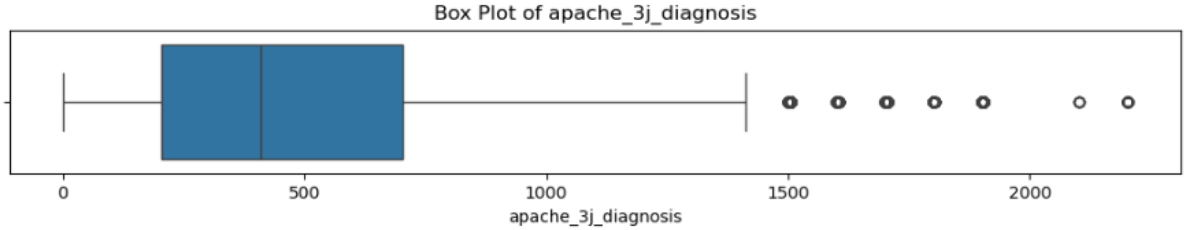


圖24 為apache_3j_diagnosis 盒鬚圖

(2) heart_rate_apache

由此盒鬚圖可發現大約在低於35或高於170時在此資料集中是被認定為離群值的，查看相關文獻會發現正常值大約坐落在60-100之間，因此可看出在本資料集中第25-75百分位的病患普遍是往較高的部分靠攏。

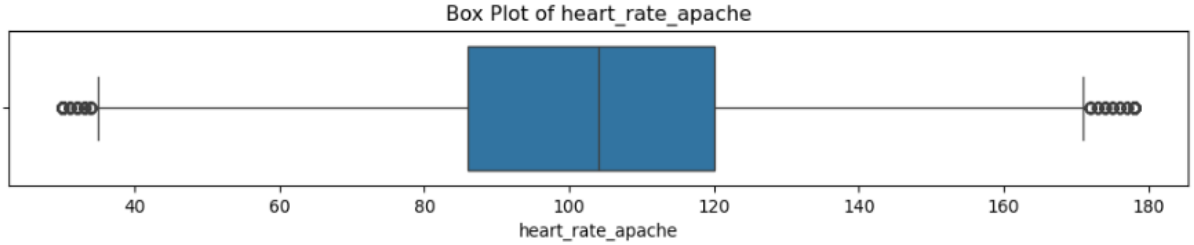


圖25 為heart_rate_apache 盒鬚圖

(3) temp_apache

正常體溫大約是36-37度之間，而由此盒鬚圖可以看出其實整體大都聚集在35.5-37.5之間，然離群值也是相當之多，不過由於資料全距坐落在約32-40之間是符合臨床數據，而並非包含一些誤植，因此不必針對離群值做截斷等動作。

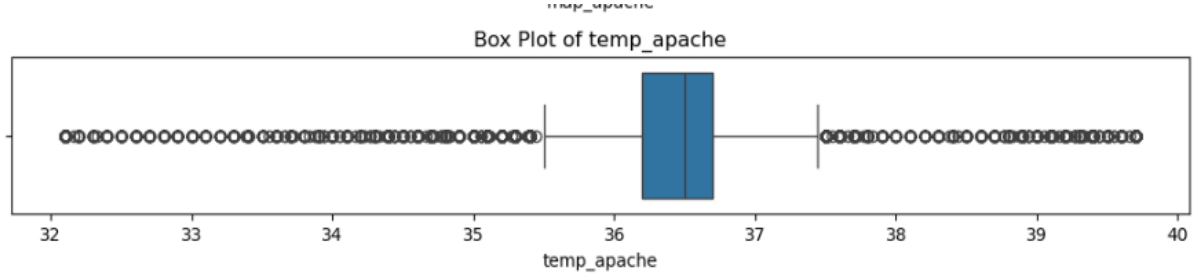


圖26 為temp_apache 盒鬚圖

(4) apache_4a_hospital_death_prob

由於大多數被判定死亡機率皆坐落在0-0.25之間，因此超過時即被判定成離群值，然而除了-1之外的離群值皆是合理範圍因此僅需要對-1做處理即可。

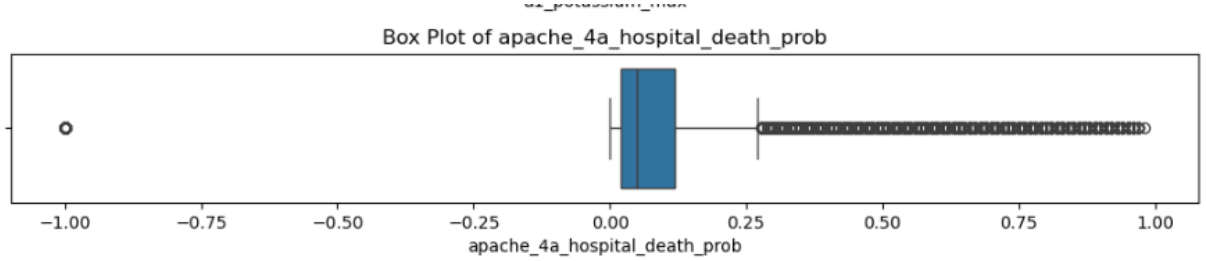


圖27 為 apache_4a_hospital_death_prob 盒鬚圖

(5) apache_4a_icu_death_prob

如同(4)由於大多數被判定死亡機率皆坐落在0-0.25之間，因此超過時即被判定成離群值，然而除了-1之外的離群值皆是合理範圍因此僅需要對-1做處理即可。

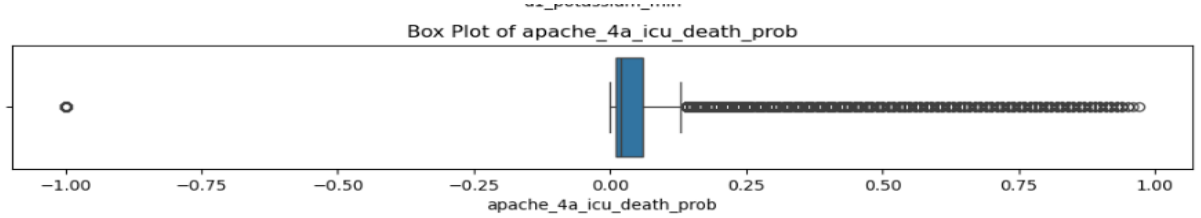


圖28 為 apache_4a_icu_death_prob 盒鬚圖

(6) d1_glucose_max、 d1_glucose_min

由於本欄為並未說明是飯前、飯後又或者是隨機血糖值，若是以正常飯前血糖值來說<(126mg/DL)為正常值；若是以飯後血糖而言則是以<140mg/DL作為標準。然而，若為隨機血糖值則是以<(200mg/DL)作為判斷基準。此時我們可以從**d1_glucose_max**中發現若是本欄是以飯前血糖或是飯後血糖作為標準時，大約有近5、60%的病患其實是患有糖尿病的，若是以隨機血糖值的話，可以看出大約有25%的病患有糖尿病之問題。此外，由於極度嚴重糖尿病患者的血糖值的確有可能突破600，因此，不必特別針對過高之離群值做其他處理。

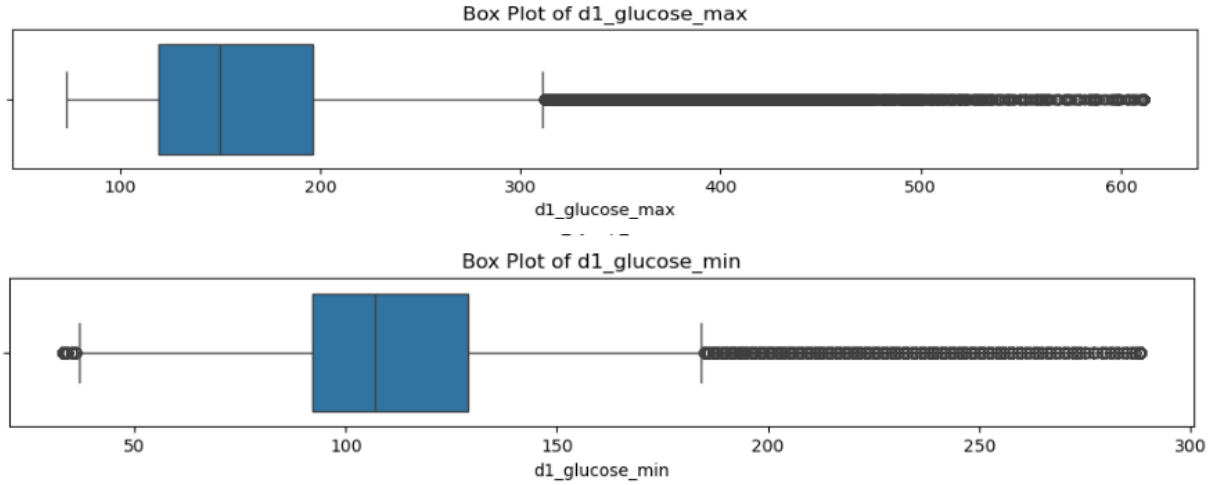


圖29、30 為 d1_glucose_max、 d1_glucose_min 盒鬚圖

(7) d1_gotassium_max、 d1_gotassium_min

經查找文獻後，發現正常範圍之鉀離子濃度為3.5-5.1 mEq/l，觀察本資料集病患之**d1_gotassium_max**數值可發現僅有少數比例以及些許離群值具有高血鉀、低血鉀之問題，不過從**d1_gotassium_min**中，我們可看出其實有約莫25%不到的比例是具有低血鉀問題的。

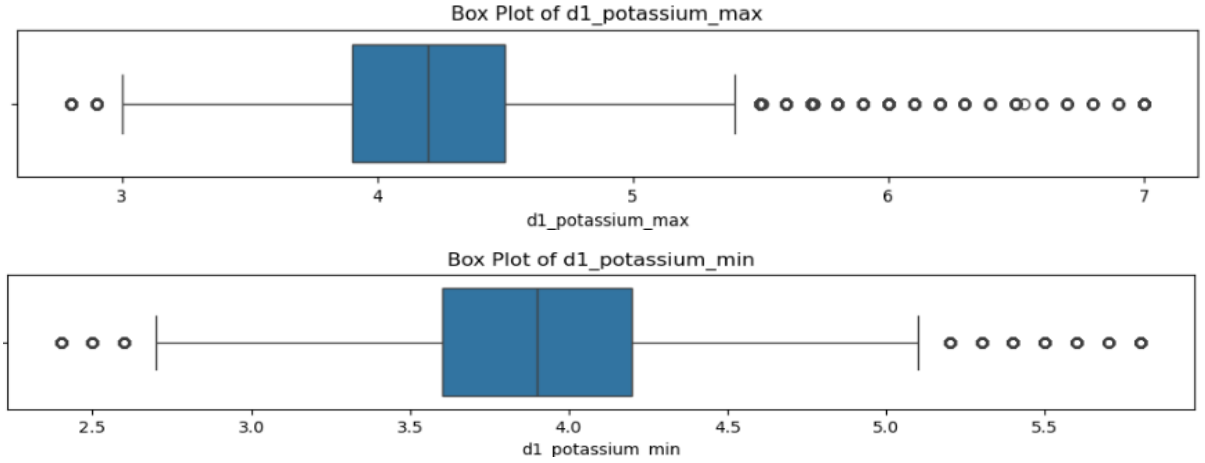


圖31、32 為 d1_gotassium_max、 d1_gotassium_min盒鬚圖

(8) h1_resprate_max、 h1_resprate_min

在此兩張盒鬚圖中我們可以看到有一些異常情況，首先在min的部分最高值有超過120次，然而max之最高值僅有60不到，因此我們決定將**h1_resprate_min超過55者一律剪裁至55，以維持資料合理性(盡可能避免當筆max < min)。**

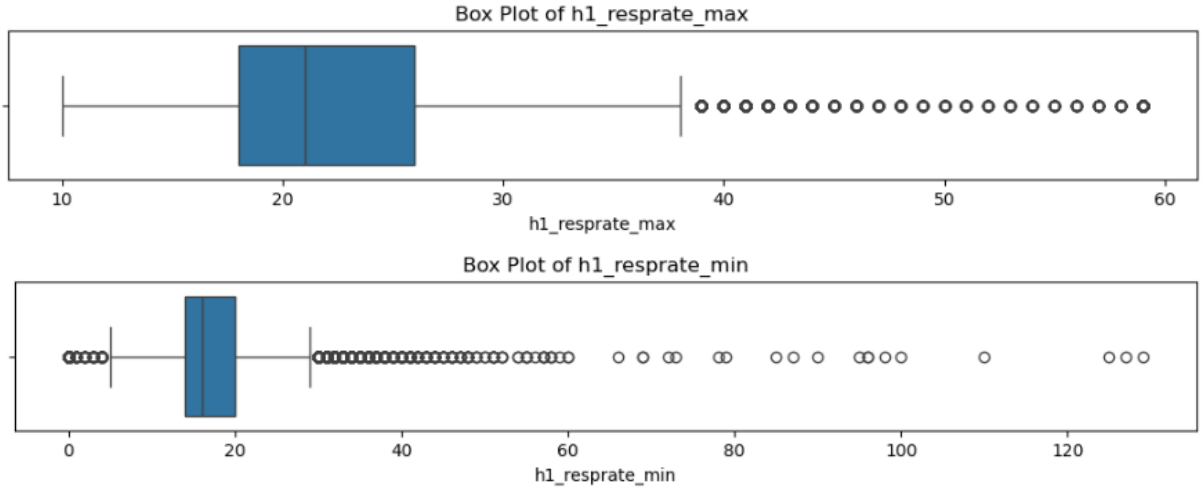


圖33、34 為h1_resprate_max、 h1_resprate_min盒鬚圖

(9) d1_resprate_max、 d1_resprate_min

如同(8)的問題，首先在min的部分最高值有超過90甚至100次，然而max之最高值僅超過90一些，因此我們決定將**d1_resprate_min超過90者一律剪裁至90，以維持資料合理性(盡可能避免當筆max < min)。**

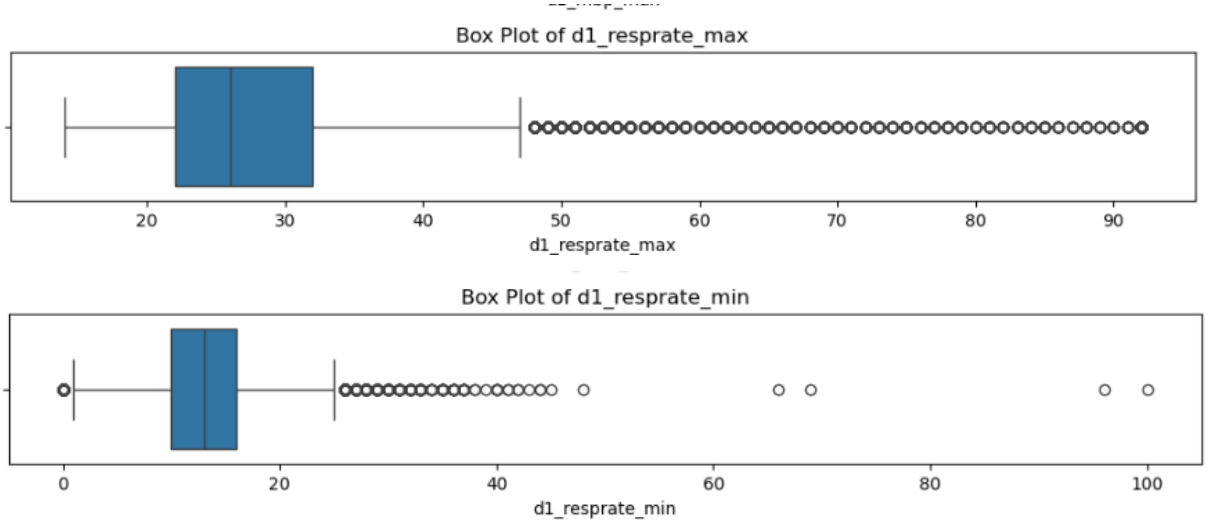


圖35、36 為d1_resprate_max、 d1_resprate_min盒鬚圖

(10) 針對上述提到之異常值進行處理

```
# 指定需要前處理的欄位
columns_to_adjust_positive = ['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob']
columns_to_adjust_absolute = ['pre_icu_los_days']

# 處理 apache_4a_hospital_death_prob 和 apache_4a_icu_death_prob：將小於 0 的值設為 0
for col in columns_to_adjust_positive:
    train_X[col] = train_X[col].apply(lambda x: max(x, 0) if pd.notnull(x) else x)
    test_X[col] = test_X[col].apply(lambda x: max(x, 0) if pd.notnull(x) else x)

# 處理 pre_icu_los_days：將小於 0 的值乘以 -1
for col in columns_to_adjust_absolute:
    train_X[col] = train_X[col].apply(lambda x: abs(x) if x < 0 and pd.notnull(x) else x)
    test_X[col] = test_X[col].apply(lambda x: abs(x) if x < 0 and pd.notnull(x) else x)
```

```
# 將 d1_resprate_min 超過 90 的值剪裁至 90
train_X_imputed['d1_resprate_min'] = train_X_imputed['d1_resprate_min'].apply(lambda x: min(x, 90) if pd.notnull(x) else x)
test_X_imputed['d1_resprate_min'] = test_X_imputed['d1_resprate_min'].apply(lambda x: min(x, 90) if pd.notnull(x) else x)

# 將 h1_resprate_min 超過 55 的值剪裁至 55
train_X_imputed['h1_resprate_min'] = train_X_imputed['h1_resprate_min'].apply(lambda x: min(x, 55) if pd.notnull(x) else x)
test_X_imputed['h1_resprate_min'] = test_X_imputed['h1_resprate_min'].apply(lambda x: min(x, 55) if pd.notnull(x) else x)

# 檢查處理結果
print("Train d1_resprate_min summary:")
print(train_X_imputed['d1_resprate_min'].describe())

print("\nTest d1_resprate_min summary:")
print(test_X_imputed['d1_resprate_min'].describe())

print("\nTrain h1_resprate_min summary:")
print(train_X_imputed['h1_resprate_min'].describe())

print("\nTest h1_resprate_min summary:")
print(test_X_imputed['h1_resprate_min'].describe())
```

圖37、38 針對上述提到之異常值進行處理

1.2.5 Data Imputing (KNN)

如同1.2.4章節中有些欄位之盒鬚圖明顯有異常值，如同pre_icu_los_days、apache_4a_hospital_death_prob、apache_4a_icu_death_prob會有低於合理下限值0的問題，若是貿然全部使用KNN則可能會造成缺失值往異常值靠攏。有鑑於此，我們先將機率欄位中為-1的資料全數轉為0，接著，再將**pre_icu_los_days**為負值的資料取絕對值，藉此避免缺失值填補時造成大量異常值。此外，由於數值型資料填補時可能原先會有些欄位是整數型態，填補後可能會有小數點，因此，在1.2.3章節中的部分欄位需要將輸出時轉成整數型態以正確型態填補。另外，本次實驗選擇的KNNImputer將n_neighbors設為5，並且由圖40可看出和使用中位數填補的值是有些微差異的。

```python
import pandas as pd
from sklearn.impute import KNNImputer

# 指定需要前處理的欄位
columns_to_adjust_positive = ['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob']
columns_to_adjust_absolute = ['pre_icu_los_days']

# 處理 apache_4a_hospital_death_prob 和 apache_4a_icu_death_prob：將小於 0 的值設為 0
for col in columns_to_adjust_positive:
    train_X[col] = train_X[col].apply(lambda x: max(x, 0) if pd.notnull(x) else x)
    test_X[col] = test_X[col].apply(lambda x: max(x, 0) if pd.notnull(x) else x)

# 處理 pre_icu_los_days：將小於 0 的值乘以 -1
for col in columns_to_adjust_absolute:
    train_X[col] = train_X[col].apply(lambda x: abs(x) if x < 0 and pd.notnull(x) else x)
    test_X[col] = test_X[col].apply(lambda x: abs(x) if x < 0 and pd.notnull(x) else x)

# 分離 numeric 和 categorical 欄位
numeric_columns = train_X.select_dtypes(include=['number']).columns
categorical_columns = train_X.select_dtypes(include=['object']).columns

# 初始化 KNNImputer
knn_imputer = KNNImputer(n_neighbors=5)

train_X_numeric_imputed = pd.DataFrame(
    knn_imputer.fit_transform(train_X[numeric_columns]),
    columns=numeric_columns
)

test_X_numeric_imputed = pd.DataFrame(
    knn_imputer.transform(test_X[numeric_columns]),
    columns=numeric_columns
)
```

```python
# 建立新欄位 'breathe_combine'
train_X_imputed['breathe_combine'] = train_X_imputed['intubated_apache'] * 2 + train_X_imputed['ventilated_apache']
test_X_imputed['breathe_combine'] = test_X_imputed['intubated_apache'] * 2 + test_X_imputed['ventilated_apache']

# 將 'breathe_combine' 四捨五入並轉為整數
train_X_imputed['breathe_combine'] = train_X_imputed['breathe_combine'].round().astype(int)
test_X_imputed['breathe_combine'] = test_X_imputed['breathe_combine'].round().astype(int)

# 檢查四捨五入後的分佈
print("Train 'breathe_combine' value counts (normalized):")
print(train_X_imputed['breathe_combine'].value_counts(normalize=True))

# print("\nTest 'breathe_combine' value counts (normalized):")
# print(test_X_imputed['breathe_combine'].value_counts(normalize=True))
```

```
124]

...  Train 'breathe_combine' value counts (normalized):
breathe_combine
0    0.673914
1    0.175149
3    0.150938
Name: proportion, dtype: float64
```

圖39、40 為使用KNN進行填補以及在需要裝箱之欄位處進行四捨五入(礙於篇幅僅以一個欄位做示範)

1.2.6 Data imbalance handling

在經過資料預處理後，我們將開始針對目標欄位進行了解。已知has_died的欄位值為0(存活)、1(死亡)，透過打印train_y.value_counts(normalize=True)我們發現，資料集是存在極度的不平衡，也就是存活比例約為0.914%，而死亡比例約為0.086%，換言之，兩者比例差至逾10.5倍。主流作法有使用SMOTE、DownSample兩種方式，前者為將少數類別的數量提升至多數類別指定倍數的量(意即若想擴增至多數類別的0.6倍。假設死亡筆數為5000筆，而存活筆數為50000筆時，可以將死亡筆數利用SMOTE擴增到30000筆)，至於DownSample則是會從多數類別中進行Sample(一樣舉死亡筆數為5000筆，而存活筆數為50000筆時，若是要採樣多數類數量至少數類別的2倍，則可以下採樣到剩下10000筆存活資料)。表1則是簡述並比較其優缺點。

	SMOTE	Down Sample
優點	1.可以保留多數類別樣本的數據分布 2.較不易過擬合	1.可以讓真實的正樣本(少數)能夠被model所學習 2. 減少資料量、運算成本
缺點	1.生成之少數類別樣本可能不符合實際情況(反而引入噪音) 2.擴大資料集、運算成本	1. 可能會有過擬合之風險 2.從多數類別樣本中抽取時，可能會丟失一些有用資訊。

表1 SMOTE、DownSample比較

為比較兩者何者較適合本資料集，因此本實驗將使用兩者進行比較，此外由於參數調試後發現若是採用多數類別:少數類別1:1的情況下普遍表現較不理想，因此經調試後比例採多數類別:少數類別為2:1時有較佳的模型表現。圖24、25分別為使用SMOTE、Downsample之程式碼截圖。

```
# Initialize SMOTE
smote = SMOTE(random_state=42, k_neighbors=3, sampling_strategy=0.5)
# Cross-validation loop
for fold, (train_index, val_index) in enumerate(kf.split(train_X_selected, train_y)):
    # Create a new model for each fold
    xgb_model = XGBClassifier(**xgb_model_params)

    # Split data into training and validation sets
    X_train, X_val = train_X_selected.iloc[train_index], train_X_selected.iloc[val_index]
    y_train, y_val = train_y.iloc[train_index], train_y.iloc[val_index]

    # Apply SMOTE only on the training set
    X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

    # Train the model
    xgb_model.fit(
        X_train_resampled,
        y_train_resampled,
        eval_set=[(X_val, y_val)],
        early_stopping_rounds=500,
        verbose=100,
    )
```

圖41 為使用SMOTE之程式碼片段

```
X_train, X_val = train_X_selected.iloc[train_index], train_X_selected.iloc[val_index]
y_train, y_val = train_y.iloc[train_index], train_y.iloc[val_index]

# Downsample majority class
train_data = pd.concat([X_train, y_train], axis=1)
majority_class = train_data[train_data['has_died'] == 0]
minority_class = train_data[train_data['has_died'] == 1]
majority_downsampled = resample(majority_class, replace=False, n_samples=int(2.0 * len(minority_class)), random_state=42)
downsampled_train_data = pd.concat([majority_downsampled, minority_class])
X_train_downsampled = downsampled_train_data.drop('has_died', axis=1)
y_train_downsampled = downsampled_train_data['has_died']

# Set decayed learning rate after reaching decay point
if fold + 1 >= decay_point:
    xgb_model.set_params(learning_rate=initial_learning_rate * decay_rate)

# Train the model
xgb_model.fit(
    X_train_downsampled, y_train_downsampled,
    eval_set=[(X_val, y_val)],
    verbose=False
)
```

圖42 為使用Down Sample之程式碼片段

2. Classification Methods (程式碼部分詳見4.Supplement)

2.1 Experiments Abstract

本次實驗在模型方面將實作XGBoost、CatBoost兩種Boosting算法並且透過SHAP分別針對各自模型選取其對於模型表現最有助益之20個欄位。此外，本次模型訓練與驗證方式為採用**K-Fold Cross Validation，其中K=5**，並且會去計算與打印每個Fold中表現最佳的threshold，並在最終生成csv檔案時將5個Fold的threshold取平均作為閾值並輸出。如同先前章節所述資料集的填補方式為各自分別使用KNN、Median填補，此外考量資料欄位不平衡之問題，我們也使用SMOTE、DownSample兩種學習策略進行比較。

2.2 Algorithm Selection & Introduction of the algorithms which we used

(1) Algorithm Selection:

本次實驗所使用之資料集具有多欄位(特徵)且輸出目標欄位為**分類**之特性，因此，當時在選用模型、演算法時有考慮a.**可解釋性較高的Tree-Based Algorithm中的Boosting**、**b.Linear Model中較適合二元分類的Logistic Regression**、**c.引入神經網路的ANN等**。

首先，**Boosting為集成學習之概念**，主要是**使用多個weak classifier**(eg: Decision Tree)提升分類表現，具體做法是**每個weak classifier會針對前一個classifier做不好的地方進行改進**逐步提升表現，此外此種方式對於處理非線性的問題有較佳的表現。**Logistic Regression則是使用Sigmoid**將線性的輸出結果**mapping到**[0,1]區間作為機率是相當適合二元分類的，然而因其模型較於簡單故實作上對於較複雜之非線性問題表現是較差的。ANN可以引入一些**激活函數**，例如**ReLU、Leaky ReLU等以處理XOR、非線性問題**，然而由於本資料集其實規格並不算太大易過擬合，因此調整隱藏層數以及調參上往往需要有較大的成本且相較於前兩者較不具可解釋性，此外經實作後，效果並不算理想。

因此，在本次實驗中，為了實現高準確性以及兼顧模型的解釋性，我**最終選擇了Tree-Based Algorithm的Boosting系列作為本次實驗使用技術**。下一小節我將介紹Boosting中我選擇的演算法以及實作時使用之套件。

(2) XGBoost[1]:

XGBoost是基於梯度提升樹（GBDT）改進的演算法，相較於傳統的梯度提升樹加入了以下幾點進行優化與改進(**僅列出主要差異**)。其技術的概念為不斷透過新增Tree，去fitting上個弱分類器預測的殘差，當新增的樹逐步增加至指定數量時，模型也就建構完畢。至於預測部分，則是每當模型要對於新的數據進行預測時，模型會把資料放到每一棵樹裡，並查看會坐落在哪一個子節點中，並獲得該子節點的分數。最終，把所有樹的節點分數相加即獲得，該樣本的最終預測值。

(1) **加入正則化項**(包含控制葉節點數量、權重懲罰)以避免模型過擬合

(2) 在迭代過程中同時使用**損失函數的一階和二階導數**，提升精確度的同時也加速收斂。

(3) 使用**Histogram-based Splitting對特徵值做裝箱**並計算**Gain**以快速找到分裂點

(3) CatBoost[2]:

CatBoost可以不必先針對類別型的資料進行Encoding特別是部分欄位原先使用One-Hot Encoding容易造成欄位(特徵)擴張的問題，而之所以可以直接使用類別欄位進行訓練是因為**CatBoost**採用**Target Encoding**的方式可以將類別型特徵轉換為與目標變量的條件機率，藉此保留類別與目標值之間的關聯性(後面會單獨拿僅處理缺失值的資料送入model進行training和做完ch1所有預處理動作的結果作比較)。此外，**CatBoost**和**GDBT**最大差異在於**CatBoost**每棵樹的結構是對稱的，即所有節點的分裂方式沿著相同的路徑，因此**CatBoost**通常可以被使用於特徵值重要程度差不多的資料集。

(4) 使用套件指令:

a. **Feature Selection: import shap**
b. **Call Model:**
   i. **XGBoost: from xgboost import XGBClassifier**
   ii. **CatBoost: from catboost import CatBoostClassifier**
c. **K-Fold(K=5): from sklearn.model_selection import StratifiedKFold**
d. **Evaluation Metrics: from sklearn.metrics import roc_auc_score, f1_score, classification_report, precision_recall_curve**
e. **Handling Imbalance:**
   i. **DownSample: from sklearn.utils import resample**
   ii. **SMOTE: from imblearn.over_sampling import SMOTE**
f. **Others:**
   i. **numpy**
   ii. **pandas**

(5) How could the results be reproduced?

Ans: 設定隨機種子，本次實驗將種子皆設定為42，包含K-Fold切分資料集、SMOTE、DownSample、模型內的random_state也是設定42。

(6) Reference:

[1] https://github.com/dmlc/xgboost

[2] https://github.com/catboost/catboost

3. Results & Analysis

本實驗將使用CatBoost、XGBoost針對以下case進行實驗，並分別比較其表現。

1. 使用中位數、最頻繁項進行插補，且把類別型資料使用encoding
2. 使用KNN針對數值型資料進行插補，類別型依然使用最頻繁項進行插補，且把類別型資料使用encoding。
3. 先選擇中位數、KNN表現較佳者做為填補，且由於CatBoost、XGBoost支援使用類別型資料，因此保留原始類別型欄位。

以上3種case皆會使用DownSample(將多數類別抽樣到少數類別之2倍數量)、SMOTE兩種方式進行比較(將少數類別擴增到多數類別之0.5倍數量)。

3.1 Case 1 包含DownSample、SMOTE以及使用CatBoost、XGBoost進行比較

(1) XGBoost於Case 1 (DownSample)

- 超參數設定:

n_estimators = 8000
learning_rate= 0.001 (有設置Decay_point於iter = 5000處)
Decay_rate = 0.1 (也就是說當觸發decay_point時學習率*0.5)
max_depth=7
colsample_bytree=0.8
subsample=0.8

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8868874286634486
Fold 1 Best Threshold: 0.6747241616249084
Fold 1 Validation F1 Score: 0.7311742065499783
Fold 1 Training Loss: 0.06938423564545658, Validation
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8213
           1       0.49      0.53      0.51       775

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.889709583912906
Fold 2 Best Threshold: 0.6870341897010803
Fold 2 Validation F1 Score: 0.7374448215289202
Fold 2 Training Loss: 0.07110281226456346, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.53      0.50      0.52       776

    accuracy                           0.92      8988
   macro avg       0.74      0.73      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8745857991322732
Fold 3 Best Threshold: 0.6585239171981812
Fold 3 Validation F1 Score: 0.7139685157194615
Fold 3 Training Loss: 0.06820014053654008, Validation L
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.47      0.49      0.48       776

    accuracy                           0.91      8988
   macro avg       0.71      0.72      0.71      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8911496753556524
Fold 4 Best Threshold: 0.6417117118835449
Fold 4 Validation F1 Score: 0.7324428740608437
Fold 4 Training Loss: 0.069545778466811, Validation Lo
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.57      0.52       776

    accuracy                           0.91      8988
   macro avg       0.72      0.75      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 5 Validation AUROC: 0.8863123988498343
Fold 5 Best Threshold: 0.6178461313247681
Fold 5 Validation F1 Score: 0.7295028432659094
Fold 5 Training Loss: 0.07012031957310985, Validation
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95      8212
           1       0.45      0.60      0.51       775

    accuracy                           0.90      8987
   macro avg       0.71      0.76      0.73      8987
weighted avg       0.92      0.90      0.91      8987
```

圖43-47為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8857 ; Average macro F1 Score: 0.7287

```
Average AUROC: 0.8857082161391403
Average F1 Score: 0.7287159726391546
Best Thresholds per Fold: [0.67472416, 0.6870342, 0.6596819, 0.6417117, 0.61784613]
```

圖48 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(2) XGBoost於Case 1 (SMOTE)

n_estimators = 8000

learning_rate= 0.003 (有設置Decay_point於iter = 5000處)

Decay_rate = 0.1 (也就是說當觸發decay_point時學習率*0.5)

max_depth=7

colsample_bytree=0.8

subsample=0.8

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8803412371417462
Fold 1 Best Threshold: 0.300904780626297
Fold 1 Validation F1 Score: 0.7264780576718959
Fold 1 Training Loss: 0.08265300222633444, Validation L
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.48      0.52      0.50       775

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8920587360212111
Fold 2 Best Threshold: 0.2485225945711136
Fold 2 Validation F1 Score: 0.7328338909780665
Fold 2 Training Loss: 0.08242169237783419, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.58      0.52       776

    accuracy                           0.91      8988
   macro avg       0.71      0.76      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 3 Validation AUROC: 0.8688812198392094
Fold 3 Best Threshold: 0.23462679982185364
Fold 3 Validation F1 Score: 0.7081738119623946
Fold 3 Training Loss: 0.08289559642676304, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      8212
           1       0.42      0.54      0.47       776

    accuracy                           0.90      8988
   macro avg       0.69      0.73      0.71      8988
weighted avg       0.91      0.90      0.90      8988
```

```
Fold 4 Validation AUROC: 0.8861926819439493
Fold 4 Best Threshold: 0.2975967228412628
Fold 4 Validation F1 Score: 0.7322142750244438
Fold 4 Training Loss: 0.0822488648169132, Validation L
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.50      0.53      0.51       776

    accuracy                           0.91      8988
   macro avg       0.73      0.74      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 5 Validation AUROC: 0.8850289898339172
Fold 5 Best Threshold: 0.2715124785900116
Fold 5 Validation F1 Score: 0.7353000986226694
Fold 5 Training Loss: 0.08266450546167217, Validation
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.48      0.57      0.52       775

    accuracy                           0.91      8987
   macro avg       0.72      0.76      0.74      8987
weighted avg       0.92      0.91      0.91      8987
```

圖49-53為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8825 ; Average macro F1 Score: 0.7270

```
Average AUROC: 0.8825005729560067
Average F1 Score: 0.7270000268518941
Best Thresholds per Fold: [0.30090478, 0.2485226, 0.2346268, 0.29759672, 0.27151248]
```

圖54 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(3) CatBoost於Case 1 (DownSample)

- 超參數設定:

iterations = 8000

learning_rate= 0.001

eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8868409248909085
Fold 1 Best Threshold: 0.6701468897538518
Fold 1 Validation F1 Score: 0.73186271793932
Fold 1 Training Loss: 0.10093969965384844, Validation
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.51      0.51      0.51       775

    accuracy                           0.92      8988
   macro avg       0.73      0.73      0.73      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 2 Validation AUROC: 0.8901822389663604
Fold 2 Best Threshold: 0.6448228134878807
Fold 2 Validation F1 Score: 0.7386092452369322
Fold 2 Training Loss: 0.10217524800958533, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.51      0.54      0.52       776

    accuracy                           0.92      8988
   macro avg       0.73      0.75      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8718825480438483
Fold 3 Best Threshold: 0.57524262891082
Fold 3 Validation F1 Score: 0.7049289590106844
Fold 3 Training Loss: 0.10027295878862666, Validation L
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      8212
           1       0.41      0.54      0.47       776

    accuracy                           0.89      8988
   macro avg       0.68      0.73      0.70      8988
weighted avg       0.91      0.89      0.90      8988
```

```
Fold 4 Validation AUROC: 0.8913290394745432
Fold 4 Best Threshold: 0.708351888908274
Fold 4 Validation F1 Score: 0.7386133291972639
Fold 4 Training Loss: 0.10096644120490 25, Validation L
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.56      0.48      0.52       776

    accuracy                           0.92      8988
   macro avg       0.76      0.72      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 5 Validation AUROC: 0.8851785742344013
Fold 5 Best Threshold: 0.6410075929088672
Fold 5 Validation F1 Score: 0.7336658732745955
Fold 5 Training Loss: 0.10190099605968687, Validation
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.48      0.55      0.52       775

    accuracy                           0.91      8987
   macro avg       0.72      0.75      0.73      8987
weighted avg       0.92      0.91      0.91      8987
```

圖56-60為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8851 ; Average macro F1 Score: 0.7295

```
Average AUROC: 0.8850826651220123
Average F1 Score: 0.7295360249317592
Best Thresholds per Fold: [0.6701468897538518, 0.6448228134878807, 0.57524262891082, 0.708351888908274, 0.6410075929088672]
```

圖61 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(4) **CatBoost於Case 1 (SMOTE)**

- 超參數設定:

iterations = 8000

learning_rate= 0.003

eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8811805045502212
Fold 1 Best Threshold: 0.36494698617413845
Fold 1 Validation F1 Score: 0.7257080078125
Fold 1 Training Loss: 85.40363126522165/1000, Validati
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.49      0.51      0.50       775

    accuracy                           0.91      8988
   macro avg       0.72      0.73      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8883515637161609
Fold 2 Best Threshold: 0.33371512775538426
Fold 2 Validation F1 Score: 0.7347738482097254
Fold 2 Training Loss: 84.43503874674123/1000, Validati
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.50      0.53      0.52       776

    accuracy                           0.91      8988
   macro avg       0.73      0.74      0.73      8988
weighted avg       0.92      0.91      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8701876120437277
Fold 3 Best Threshold: 0.25517067940851634
Fold 3 Validation F1 Score: 0.707471377918411
Fold 3 Training Loss: 85.21761350587741/1000, Validati
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      8212
           1       0.41      0.57      0.48       776

    accuracy                           0.89      8988
   macro avg       0.68      0.75      0.71      8988
weighted avg       0.91      0.89      0.90      8988
```

```
Fold 4 Validation AUROC: 0.8896593682365762
Fold 4 Best Threshold: 0.2725206317334506
Fold 4 Validation F1 Score: 0.7257725348515054
Fold 4 Training Loss: 84.74017337179765/1000, Validati
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      8212
           1       0.44      0.60      0.51       776

    accuracy                           0.90      8988
   macro avg       0.70      0.76      0.73      8988
weighted avg       0.92      0.90      0.91      8988
```

```
Fold 5 Validation AUROC: 0.8822513080778718
Fold 5 Best Threshold: 0.319496897593262
Fold 5 Validation F1 Score: 0.7366978271242981
Fold 5 Training Loss: 85.04168313464058/1000, Validati
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.48      0.57      0.52       775

    accuracy                           0.91      8987
   macro avg       0.72      0.76      0.74      8987
weighted avg       0.92      0.91      0.91      8987
```

圖62-66為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8823 ; Average macro F1 Score: 0.7261

```
Average AUROC: 0.8823260713249116
Average F1 Score: 0.726084719183288
Best Thresholds per Fold: [0.36494698617413845, 0.33371512775538426, 0.25517067940851634, 0.2725206317334506, 0.319496897593262]
```

圖67 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(5) Case 1挑選的20個欄位及可視化(SHAP):

Top features across all folds:

['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob', 'age', 'd1_heartrate_min', 'ventilated_apache', 'd1_spo2_min', 'd1_sysbp_min', 'd1_heartrate_max', 'apache_3j_diagnosis', 'd1_resprate_min', 'd1_temp_min', 'd1_temp_max', 'd1_resprate_max', 'gcs_overall', 'd1_mbp_min', 'pre_icu_los_days', 'resprate_apache', 'd1_sysbp_max', 'bmi', 'd1_glucose_min']

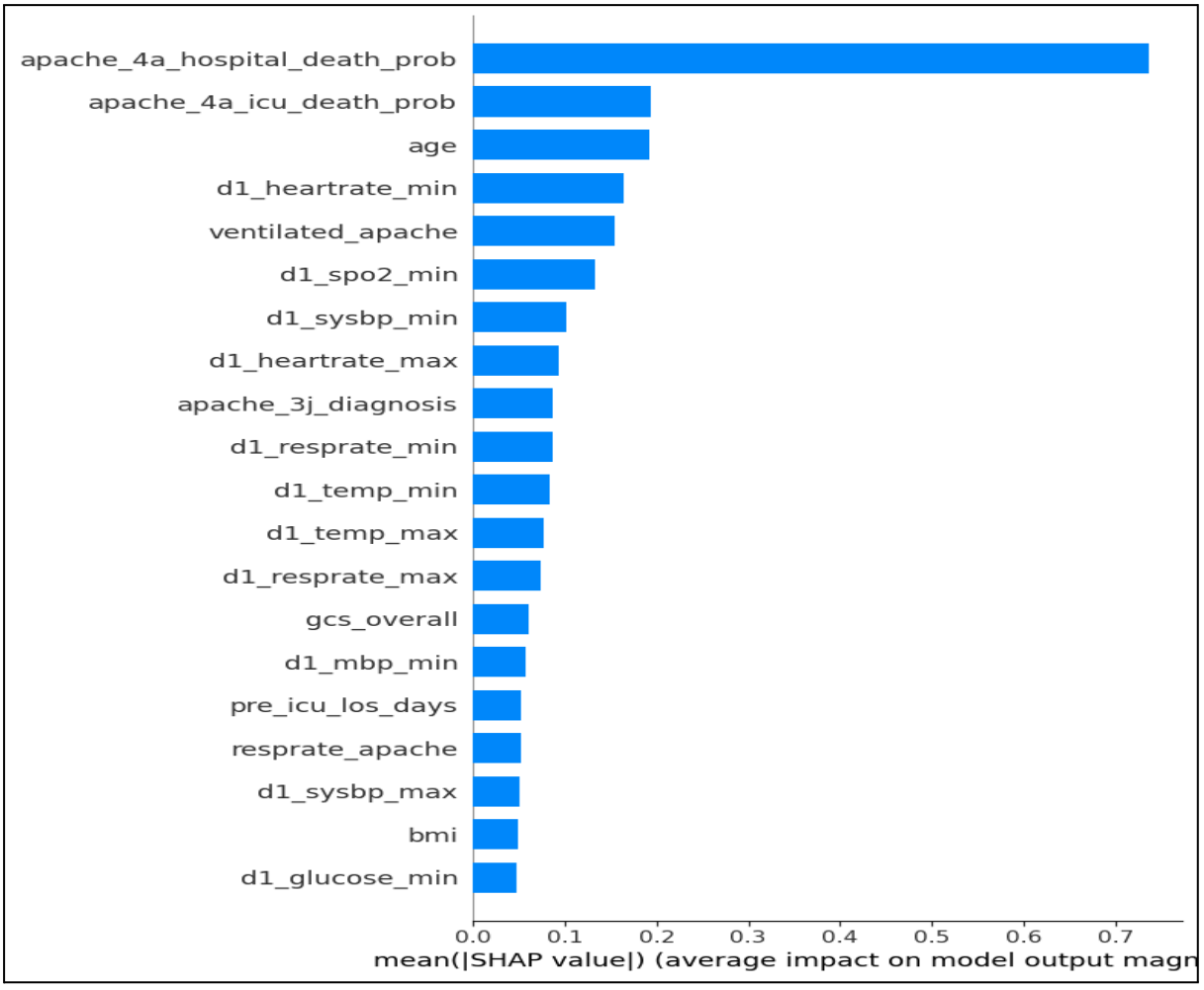


圖55 可視化挑選之20個欄位重要度(SHAP)

3.2 Case 2 包含DownSample、SMOTE以及使用CatBoost、XGBoost進行比較

(1) XGBoost於Case 2 (DownSample)

- 超參數設定:

n_estimators = 8000

learning_rate= 0.001 (有設置Decay_point於iter = 5000處)

Decay_rate = 0.1 (也就是說當觸發decay_point時學習率*0.5)

max_depth=7

colsample_bytree=0.8

subsample=0.8

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8860518532774555
Fold 1 Best Threshold: 0.7051290273666382
Fold 1 Validation F1 Score: 0.7250088267438006
Fold 1 Training Loss: 0.06759873033051236, Validation L
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      8213
           1       0.51      0.49      0.50       775

    accuracy                           0.92      8988
   macro avg       0.73      0.72      0.73      8988
weighted avg       0.91      0.92      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8921726628368845
Fold 2 Best Threshold: 0.6395183801651001
Fold 2 Validation F1 Score: 0.7340664513064181
Fold 2 Training Loss: 0.06965306526508838, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.48      0.56      0.52       776

    accuracy                           0.91      8988
   macro avg       0.72      0.75      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 3 Validation AUROC: 0.8726256615915355
Fold 3 Best Threshold: 0.6745278239250183
Fold 3 Validation F1 Score: 0.7131168263381995
Fold 3 Training Loss: 0.06577327327161904, Validation L
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.48      0.47      0.48       776

    accuracy                           0.91      8988
   macro avg       0.72      0.71      0.71      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8904521482266335
Fold 4 Best Threshold: 0.6410039067268372
Fold 4 Validation F1 Score: 0.731724078641337
Fold 4 Training Loss: 0.06812727871240805, Validation
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.57      0.51       776

    accuracy                           0.91      8988
   macro avg       0.71      0.75      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 5 Validation AUROC: 0.886372421161793
Fold 5 Best Threshold: 0.6154341697692871
Fold 5 Validation F1 Score: 0.7300399252128938
Fold 5 Training Loss: 0.06844418329837125, Validation
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95      8212
           1       0.45      0.60      0.51       775

    accuracy                           0.90      8987
   macro avg       0.71      0.77      0.73      8987
weighted avg       0.92      0.90      0.91      8987
```

圖69-73為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8855 ; Average macro F1 Score: 0.7268

```
Average AUROC: 0.8855349494188604
Average F1 Score: 0.7267912216485299
Best Thresholds per Fold: [0.705129, 0.6395184, 0.6745278, 0.6410039, 0.61543417]
```

圖74 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(2) XGBoost於Case 2 (SMOTE)

- 超參數設定:

n_estimators = 8000

learning_rate= 0.003（有設置Decay_point於iter = 5000處）

Decay_rate = 0.1（也就是說當觸發decay_point時學習率*0.5）

max_depth=7

colsample_bytree=0.8

subsample=0.8

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8818790037823592
Fold 1 Best Threshold: 0.3392833173274994
Fold 1 Validation F1 Score: 0.725280116379844
Fold 1 Training Loss: 0.08387679733459084, Validation
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.49      0.51      0.50       775

    accuracy                           0.91      8988
   macro avg       0.72      0.73      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8890178629714626
Fold 2 Best Threshold: 0.3126925826072693
Fold 2 Validation F1 Score: 0.7282873644245764
Fold 2 Training Loss: 0.08328426416630448, Validation L
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.49      0.52      0.51       776

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 3 Validation AUROC: 0.8713968682993457
Fold 3 Best Threshold: 0.2939603924751282
Fold 3 Validation F1 Score: 0.7143308153565685
Fold 3 Training Loss: 0.08403304413194299, Validation L
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.46      0.50      0.48       776

    accuracy                           0.91      8988
   macro avg       0.71      0.72      0.71      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8847319549967109
Fold 4 Best Threshold: 0.31765735149383545
Fold 4 Validation F1 Score: 0.7309037656293313
Fold 4 Training Loss: 0.0834832247190746, Validation L
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.49      0.53      0.51       776

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 5 Validation AUROC: 0.8858578319689517
Fold 5 Best Threshold: 0.34936121106147766
Fold 5 Validation F1 Score: 0.7336231885403879
Fold 5 Training Loss: 0.08368077076155982, Validation L
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.51      0.52      0.51       775

    accuracy                           0.91      8987
   macro avg       0.73      0.74      0.73      8987
weighted avg       0.92      0.91      0.92      8987
```

圖75-79為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8826 ; Average macro F1 Score: 0.7265

```
Average AUROC: 0.882576704403766
Average F1 Score: 0.7264850500661416
Best Thresholds per Fold: [0.33928332, 0.31269258, 0.2939604, 0.31765735, 0.3493612]
```

圖80 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(3) CatBoost 於Case 2 (DownSample)

- 超參數設定:

iterations = 8000

learning_rate= 0.0015

eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8862811514396923
Fold 1 Best Threshold: 0.6857989554278577
Fold 1 Validation F1 Score: 0.731941799354504
Fold 1 Training Loss: 0.08566829364259432, Validation L
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.51      0.51      0.51       775

    accuracy                           0.92      8988
   macro avg       0.73      0.73      0.73      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 2 Validation AUROC: 0.8914942804344661
Fold 2 Best Threshold: 0.6536541955200167
Fold 2 Validation F1 Score: 0.7353727101603733
Fold 2 Training Loss: 0.08729665541813315, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.50      0.53      0.52       776

    accuracy                           0.91      8988
   macro avg       0.73      0.74      0.74      8988
weighted avg       0.92      0.91      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8712171903324779
Fold 3 Best Threshold: 0.6094425570761607
Fold 3 Validation F1 Score: 0.7108131421638377
Fold 3 Training Loss: 0.08494780800089097, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      8212
           1       0.44      0.52      0.48       776

    accuracy                           0.90      8988
   macro avg       0.70      0.73      0.71      8988
weighted avg       0.91      0.90      0.90      8988
```

```
Fold 4 Validation AUROC: 0.8911104443585199
Fold 4 Best Threshold: 0.6835808589839051
Fold 4 Validation F1 Score: 0.7391662561367149
Fold 4 Training Loss: 0.08702679686883535, Validation
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.52      0.52      0.52       776

    accuracy                           0.92      8988
   macro avg       0.74      0.74      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 5 Validation AUROC: 0.8859557217604449
Fold 5 Best Threshold: 0.6230096453942506
Fold 5 Validation F1 Score: 0.7351970094985041
Fold 5 Training Loss: 0.08712353780552388, Validation L
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.59      0.52       775

    accuracy                           0.91      8987
   macro avg       0.71      0.76      0.74      8987
weighted avg       0.92      0.91      0.91      8987
```

圖81-85為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8852 ; Average macro F1 Score: 0.7305

```
Average AUROC: 0.8852117576651203
Average F1 Score: 0.7304981834627868
Best Thresholds per Fold: [0.6857989554278577, 0.6536541955200167, 0.6094425570761607, 0.6835808589839051, 0.6230096453942506]
```

圖86 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(4) CatBoost 於Case 2 (SMOTE)

- 超參數設定:

iterations = 8000

learning_rate= 0.003

eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.881136514495116
Fold 1 Best Threshold: 0.3524318226403233
Fold 1 Validation F1 Score: 0.7267244901651511
Fold 1 Training Loss: 0.08411245519137343
Fold 1 Validation Loss: 0.11079234937784123
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.49      0.51      0.50       775

    accuracy                           0.91      8988
   macro avg       0.72      0.73      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8877651387710216
Fold 2 Best Threshold: 0.3163965204248709
Fold 2 Validation F1 Score: 0.7294435551771214
Fold 2 Training Loss: 0.08360040017689866
Fold 2 Validation Loss: 0.1084389927291169
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.48      0.54      0.51       776

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 3 Validation AUROC: 0.8693309639903386
Fold 3 Best Threshold: 0.2884817827662627
Fold 3 Validation F1 Score: 0.7111077410935055
Fold 3 Training Loss: 0.08402502996458734
Fold 3 Validation Loss: 0.10742230310266437
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      8212
           1       0.44      0.52      0.48       776

    accuracy                           0.90      8988
   macro avg       0.70      0.73      0.71      8988
weighted avg       0.91      0.90      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8875743192009682
Fold 4 Best Threshold: 0.34185493210125917
Fold 4 Validation F1 Score: 0.7390960625187648
Fold 4 Training Loss: 0.08394283457795326
Fold 4 Validation Loss: 0.10928314213610751
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.51      0.54      0.52       776

    accuracy                           0.92      8988
   macro avg       0.73      0.74      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 5 Validation AUROC: 0.8850176767279982
Fold 5 Best Threshold: 0.3620141073795765
Fold 5 Validation F1 Score: 0.7366779348254682
Fold 5 Training Loss: 0.08367954553444909
Fold 5 Validation Loss: 0.11031449565026531
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.51      0.52      0.52       775

    accuracy                           0.92      8987
   macro avg       0.73      0.74      0.74      8987
weighted avg       0.92      0.92      0.92      8987
```

圖87-91為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8822 ; Average macro F1 Score: 0.7286

```
Average AUROC: 0.8821649226370886
Average F1 Score: 0.7286099567560023
Best Thresholds per Fold: [0.3524318226403233, 0.3163965204248709, 0.2884817827662627, 0.34185493210125917, 0.3620141073795765]
```

圖92 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(5) XGBoost於Case 2挑選的20個欄位及可視化(SHAP):

Top features across all folds:
['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob', 'age', 'ventilated_apache', 'd1_heartrate_min', 'd1_spo2_min', 'gcs_overall', 'd1_sysbp_min', 'apache_3j_diagnosis', 'd1_resprate_min', 'd1_heartrate_max', 'd1_resprate_max', 'd1_temp_min', 'd1_temp_max', 'resprate_apache', 'pre_icu_los_days', 'd1_glucose_min', 'd1_potassium_max', 'd1_sysbp_max', 'd1_mbp_min']

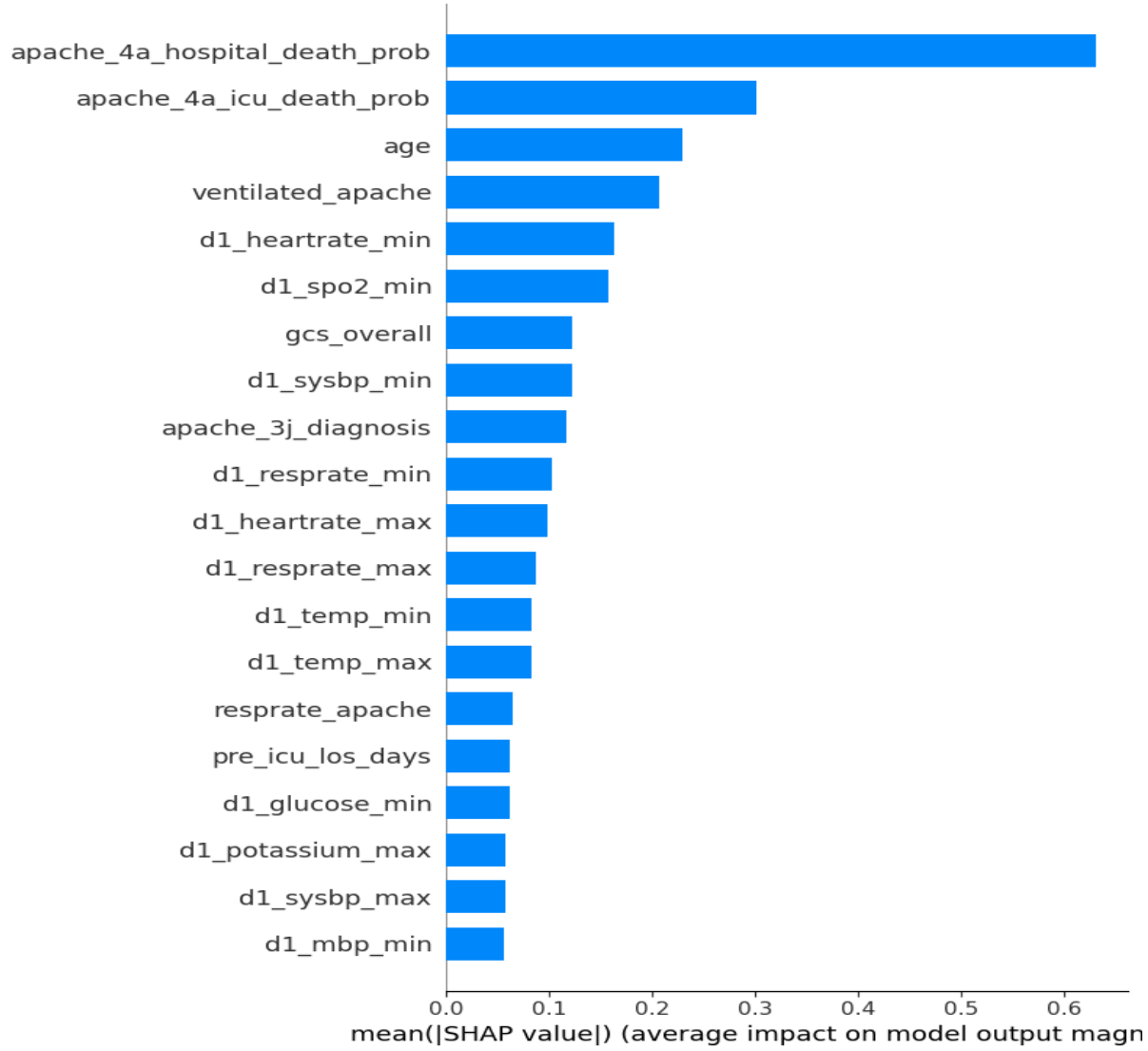


圖93 可視化挑選之20個欄位重要度(SHAP)

3.3 Case 3 僅使用包含DownSample 以及使用CatBoost、XGBoost進行比較

由於SMOTE需要透過插植以實現生成少數類別人工資料，然而因為Case3包含了類別行欄位，若是先對其做Encoding則實質上等同於退化成Case2，故本案例並不列入SMOTE。

(1) XGBoost於Case 3 (DownSample)

- 超參數設定:

n_estimators = 8000

learning_rate= 0.001（有設置Decay_point於iter = 5000處）

Decay_rate = 0.1（也就是說當觸發decay_point時學習率*0.5）

max_depth=7

colsample_bytree=0.8

subsample=0.8

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8866999996072317
Fold 1 Best Threshold: 0.675308346748352
Fold 1 Validation F1 Score: 0.7280557082646142
Fold 1 Training Loss: 0.06669520738066108, Validation Lo
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.49      0.52      0.50       775

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.8923741532381578
Fold 2 Best Threshold: 0.6314111948013306
Fold 2 Validation F1 Score: 0.7303846876671534
Fold 2 Training Loss: 0.06951296845325308, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.56      0.51       776

    accuracy                           0.91      8988
   macro avg       0.71      0.75      0.73      8988
weighted avg       0.92      0.91      0.91      8988
```

```
Fold 3 Validation AUROC: 0.8746033746189885
Fold 3 Best Threshold: 0.6640057563781738
Fold 3 Validation F1 Score: 0.7082794783017865
Fold 3 Training Loss: 0.06771553146044385, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.46      0.47      0.47       776

    accuracy                           0.91      8988
   macro avg       0.71      0.71      0.71      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8932948262788678
Fold 4 Best Threshold: 0.71253901720047
Fold 4 Validation F1 Score: 0.7382427573853901
Fold 4 Training Loss: 0.06807370570321805, Validation
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.54      0.50      0.52       776

    accuracy                           0.92      8988
   macro avg       0.75      0.73      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 5 Validation AUROC: 0.8859143974985465
Fold 5 Best Threshold: 0.6420779228210449
Fold 5 Validation F1 Score: 0.7303188784673381
Fold 5 Training Loss: 0.06865055909125503, Validation
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.57      0.51       775

    accuracy                           0.91      8987
   macro avg       0.71      0.75      0.73      8987
weighted avg       0.92      0.91      0.91      8987
```

圖94-98為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8866 ; Average macro F1 Score: 0.7271

```
Average AUROC: 0.8865773502483585
Average F1 Score: 0.7270563020172565
Best Thresholds per Fold: [0.67530835, 0.6314112, 0.66400576, 0.712539, 0.6420779]
```

圖99 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

(2) XGBoost於Case 3 (DownSample)挑選的20個欄位及可視化(SHAP):

Top features across all folds:
['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob', 'age', 'ventilated_apache', 'd1_heartrate_min', 'd1_spo2_min', 'gcs_overall', 'd1_sysbp_min', 'd1_resprate_min', 'apache_3j_diagnosis', 'd1_heartrate_max', 'd1_resprate_max', 'icu_admit_source', 'd1_temp_max', 'd1_temp_min', 'apache_3j_bodysystem', 'resprate_apache', 'd1_glucose_min', 'd1_potassium_max', 'd1_sysbp_max']

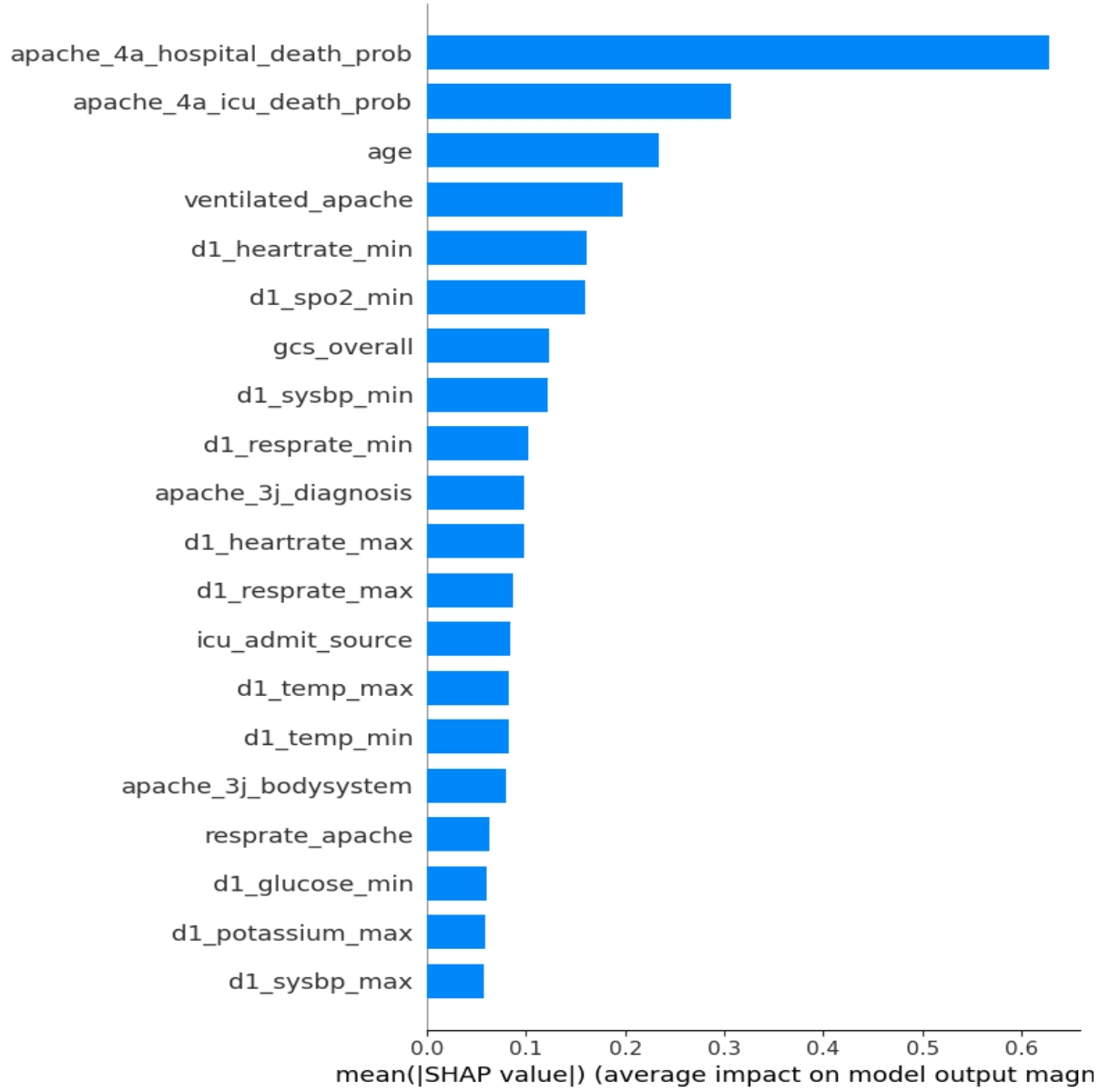


圖100 可視化挑選之20個欄位重要度(SHAP)

(3) CatBoost 於Case 3 (DownSample)

- 超參數設定:

iterations = 8000

learning_rate= 0.003

eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8842392273461036
Fold 1 Best Threshold: 0.6551146738583848
Fold 1 Validation F1 Score: 0.7296140908417936
Fold 1 Training Loss: 0.10517457079348223, Validation
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.49      0.53      0.51       775

    accuracy                           0.91      8988
   macro avg       0.72      0.74      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 2 Validation AUROC: 0.889183574703351
Fold 2 Best Threshold: 0.6588456023320777
Fold 2 Validation F1 Score: 0.7341275486139751
Fold 2 Training Loss: 0.1065877436089313, Validation L
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      8212
           1       0.52      0.51      0.51       776

    accuracy                           0.92      8988
   macro avg       0.74      0.73      0.73      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8719812532326342
Fold 3 Best Threshold: 0.5796353723217744
Fold 3 Validation F1 Score: 0.705853133546164
Fold 3 Training Loss: 0.10370681439172387, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.94      8212
           1       0.42      0.54      0.47       776

    accuracy                           0.90      8988
   macro avg       0.69      0.73      0.71      8988
weighted avg       0.91      0.90      0.90      8988
```

```
Fold 4 Validation AUROC: 0.8919016551086918
Fold 4 Best Threshold: 0.6984046777969496
Fold 4 Validation F1 Score: 0.7379936701726665
Fold 4 Training Loss: 0.1044879644880235, Validation L
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.54      0.50      0.52       776

    accuracy                           0.92      8988
   macro avg       0.75      0.73      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 5 Validation AUROC: 0.8828414751033107
Fold 5 Best Threshold: 0.6009624388187321
Fold 5 Validation F1 Score: 0.72860727160849
Fold 5 Training Loss: 0.10616919392982452, Validation L
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95      8212
           1       0.45      0.59      0.51       775

    accuracy                           0.90      8987
   macro avg       0.71      0.76      0.73      8987
weighted avg       0.92      0.90      0.91      8987
```

圖101-105為 5-Fold之驗證集表現、分類報告

b. 5-Fold平均AUROC、F1-Score

average AUROC: 0.8840 ; Average macro F1 Score: 0.7272

```
Average AUROC: 0.8840294370988182
Average F1 Score: 0.7272391429566178
Best Thresholds per Fold: [0.6551146738583848, 0.6588456023320777, 0.5796353723217744, 0.6984046777969496, 0.6009624388187321]
```

圖106 為 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值

（4）CatBoost於Case 3 (DownSample)挑選的20個欄位及可視化(SHAP):

Top features across all folds:

['apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob', 'age', 'ventilated_apache', 'd1_sysbp_min', 'gcs_overall', 'icu_admit_source', 'breathe_combine', 'd1_spo2_min', 'd1_heartrate_min', 'd1_resprate_min', 'spo2_level', 'd1_heartrate_max', 'd1_temp_min', 'apache_3j_diagnosis', 'd1_resprate_max', 'd1_mbp_min', 'apache_3j_bodysystem', 'd1_temp_max', 'resprate_apache']

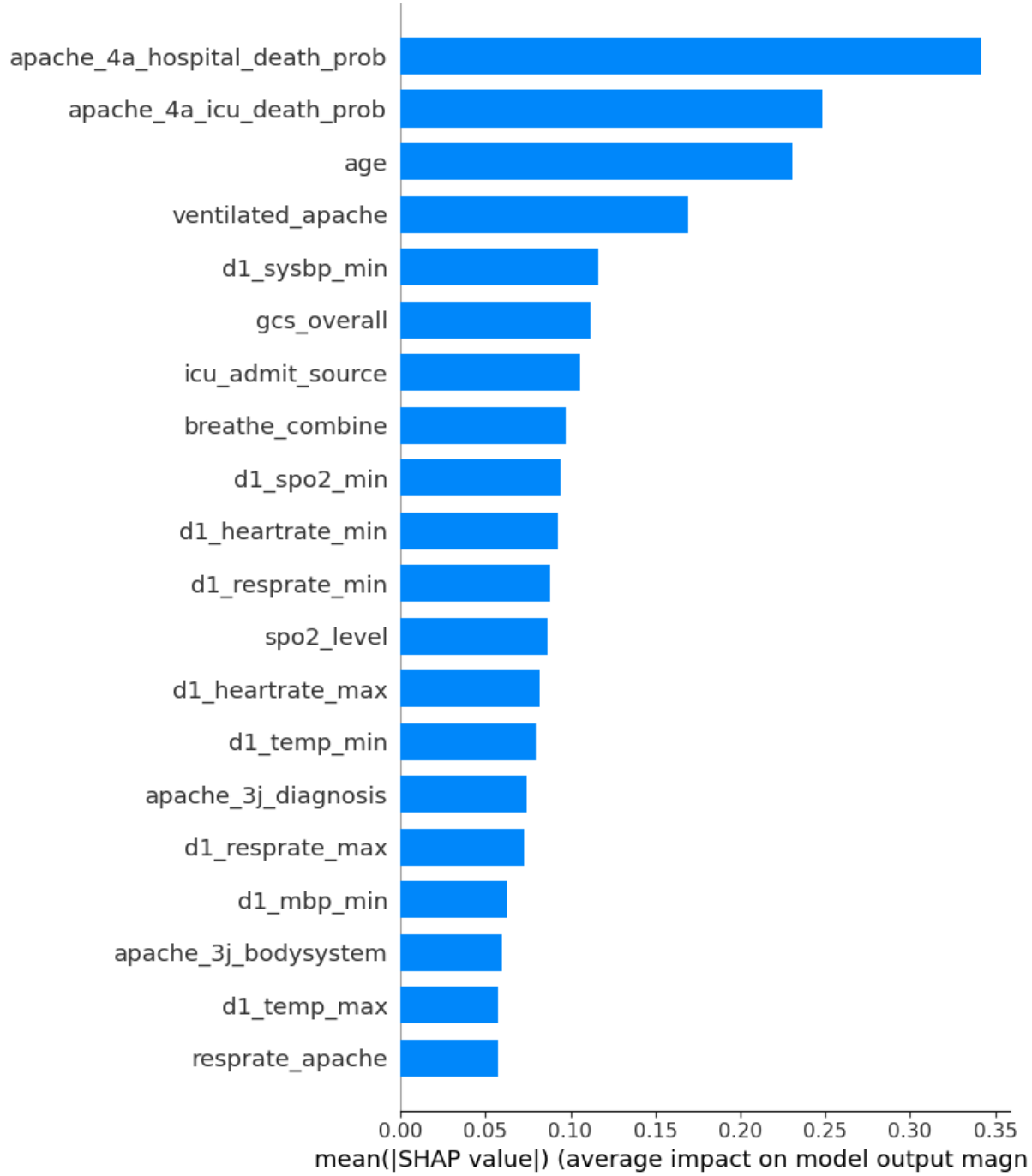


圖107 可視化挑選之20個欄位重要度(SHAP)

3.4 結合特徵取聯集測試效果

```
union = [
    'apache_3j_bodysystem', 'd1_heartrate_min', 'd1_glucose_min',
    'apache_4a_hospital_death_prob', 'bmi', 'age', 'd1_resprate_max',
    'apache_3j_diagnosis', 'd1_heartrate_max', 'spo2_level',
    'd1_temp_max', 'd1_potassium_max', 'ventilated_apache',
    'resprate_apache', 'apache_4a_icu_death_prob', 'gcs_overall',
    'd1_resprate_min', 'd1_temp_min', 'icu_admit_source',
    'd1_sysbp_min', 'breathe_combine', 'd1_spo2_min',
    'd1_mbp_min', 'd1_sysbp_max', 'pre_icu_los_days'
]
```

圖108 由先前提供的重要特徵進行聯集的union共有25個欄位

(1) CatBoost (採用KNN進行插補、使用DownSample)於測試集獲得
macro F1 score:0.727
- 超參數設定:
iterations = 8000
learning_rate= 0.003
eval_metric='AUC'

a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8868770595790311
Fold 1 Best Threshold: 0.6827696388950544
Fold 1 Validation F1 Score: 0.7338943038122618
Fold 1 Training Loss: 0.1008166504544045, Validation Lo
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      8213
           1       0.52      0.50      0.51       775

    accuracy                           0.92      8988
   macro avg       0.74      0.73      0.73      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 2 Validation AUROC: 0.8925183663836177
Fold 2 Best Threshold: 0.6590675217351903
Fold 2 Validation F1 Score: 0.737090718518806
Fold 2 Training Loss: 0.10197053348676985, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.51      0.53      0.52       776

    accuracy                           0.92      8988
   macro avg       0.73      0.74      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

```
Fold 3 Validation AUROC: 0.8733320549259066
Fold 3 Best Threshold: 0.648263462959405
Fold 3 Validation F1 Score: 0.7108307318493655
Fold 3 Training Loss: 0.09948143158756786, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.47      0.47      0.47       776

    accuracy                           0.91      8988
   macro avg       0.71      0.71      0.71      8988
weighted avg       0.91      0.91      0.91      8988
```

```
Fold 4 Validation AUROC: 0.8929376672809717
Fold 4 Best Threshold: 0.691648665120135
Fold 4 Validation F1 Score: 0.7413666370457195
Fold 4 Training Loss: 0.10074739376015517, Validation
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      8212
           1       0.55      0.51      0.53       776

    accuracy                           0.92      8988
   macro avg       0.75      0.73      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```

圖109-112為 5-Fold前4 Fold之驗證集表現、分類報告

```
Fold 5 Validation AUROC: 0.8854288767028581
Fold 5 Best Threshold: 0.6039719419271156
Fold 5 Validation F1 Score: 0.7337306132577541
Fold 5 Training Loss: 0.10162313513846659, Validation Loss: 0.15940628226140957
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95      8212
           1       0.46      0.61      0.52       775

    accuracy                           0.90      8987
   macro avg       0.71      0.77      0.73      8987
weighted avg       0.92      0.90      0.91      8987

Average AUROC: 0.886218804974477
Average F1 Score: 0.7313826008967814
Best Thresholds per Fold: [0.6827696388950544, 0.6590675217351903, 0.648263462959405, 0.691648665120135, 0.6039719419271156]
```

圖113 為第5Fold驗證集表現、分類報告 以及 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值(avg AUROC = 0.8862; avg F1 Score = 0.731)

(2) XGBoost於Case 3 (DownSample)

- 超參數設定:

n_estimators = 8000

learning_rate= 0.001 (有設置Decay_point於iter = 5000處)

Decay_rate = 0.1 (也就是說當觸發decay_point時學習率*0.5)

max_depth=7

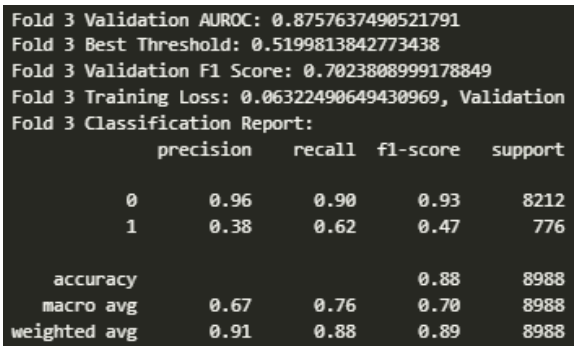
colsample_bytree=0.8

subsample=0.8

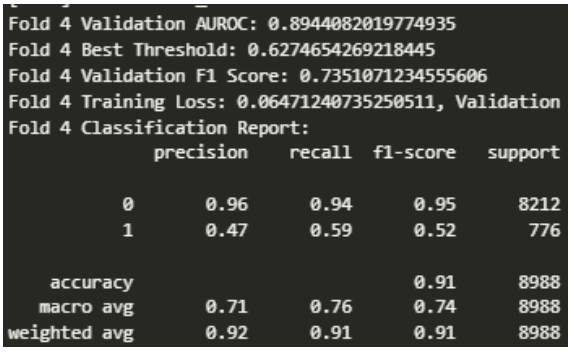
a. 各Fold之分類報告

```
Fold 1 Validation AUROC: 0.8879318782575225
Fold 1 Best Threshold: 0.6981326341629028
Fold 1 Validation F1 Score: 0.7277864678795427
Fold 1 Training Loss: 0.06339735244329965, Validation
Fold 1 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8213
           1       0.50      0.50      0.50       775

    accuracy                           0.91      8988
   macro avg       0.73      0.73      0.73      8988
weighted avg       0.91      0.91      0.91      8988
```

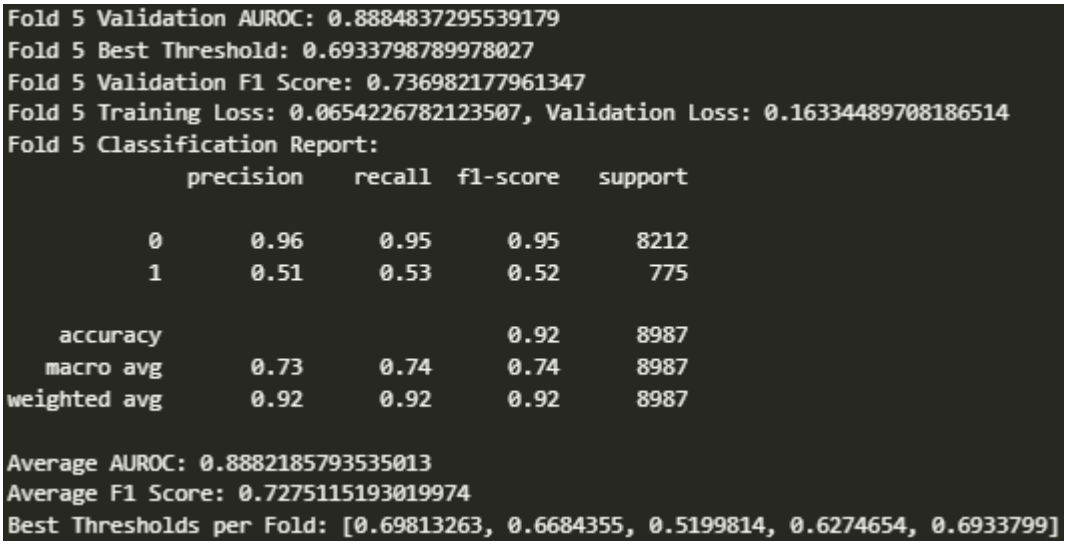
```
Fold 2 Validation AUROC: 0.8945053379263939
Fold 2 Best Threshold: 0.6684355139732361
Fold 2 Validation F1 Score: 0.7353009272956519
Fold 2 Training Loss: 0.06621340879118857, Validation
Fold 2 Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      8212
           1       0.51      0.52      0.52       776

    accuracy                           0.92      8988
   macro avg       0.73      0.74      0.74      8988
weighted avg       0.92      0.92      0.92      8988
```



```
Fold 3 Validation AUROC: 0.8757637490521791
Fold 3 Best Threshold: 0.5199813842773438
Fold 3 Validation F1 Score: 0.7023808999178849
Fold 3 Training Loss: 0.06322490649430969, Validation
Fold 3 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.90      0.93      8212
           1       0.38      0.62      0.47       776

    accuracy                           0.88      8988
   macro avg       0.67      0.76      0.70      8988
weighted avg       0.91      0.88      0.89      8988
```



```
Fold 4 Validation AUROC: 0.8944082019774935
Fold 4 Best Threshold: 0.6274654269218445
Fold 4 Validation F1 Score: 0.7351071234555606
Fold 4 Training Loss: 0.06471240735250511, Validation
Fold 4 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      8212
           1       0.47      0.59      0.52       776

    accuracy                           0.91      8988
   macro avg       0.71      0.76      0.74      8988
weighted avg       0.92      0.91      0.91      8988
```



```
Fold 5 Validation AUROC: 0.8884837295539179
Fold 5 Best Threshold: 0.6933798789978027
Fold 5 Validation F1 Score: 0.736982177961347
Fold 5 Training Loss: 0.0654226782123507, Validation Loss: 0.16334489708186514
Fold 5 Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      8212
           1       0.51      0.53      0.52       775

    accuracy                           0.92      8987
   macro avg       0.73      0.74      0.74      8987
weighted avg       0.92      0.92      0.92      8987

Average AUROC: 0.8882185793535013
Average F1 Score: 0.7275115193019974
Best Thresholds per Fold: [0.69813263, 0.6684355, 0.5199814, 0.6274654, 0.6933799]
```

圖115-119 為5Fold驗證集表現、分類報告 以及 5-Fold後之平均AUROC、F1 Score以及打印出各Fold最佳之閾值(avg AUROC = 0.8882; avg F1 Score = 0.7275)

3.5 實驗綜合分析

(1)實驗選取欄位方法與分析:

在3.3章節中，我們採用的是SHAP進行欄位分析，以我們的角度而言，我們希望在模型訓練前可以先選取有用且對於分類是最有貢獻的欄位。而SHAP則是用來展現模型對於該資料集最為依賴的是哪些欄位。此外有別於一些基於統計的方法選取重要度較高欄位時面臨到類別型欄位是需要事前先Encoding的，但如同前面章節可以發現，當我們把一些類別型欄位做編碼，再做Info Gain後，往往那些欄位依然是選不進設定的欄位數量(意即貢獻度較差)。因此我們採用SHAP一方面可以選擇可支援類別型欄位的Model進行訓練，一方面也可以利用SHAP的特型找出我們實作模型對於本資料集最依賴的欄位是哪些。

(2) 選取的欄位分析:

由於如同3.3章已經有於case1、2分別各挑選XGBoost只是因為有別於缺失值插補方式的SHAP後的Top 20 features以及在3.4 case3部分把皆使用KNN插補的CatBoost、XGBoost分別各選定的Top 20 features，為避免疏漏因此我在這邊將採聯集如3.4使用之欄位共25個欄位進行介紹與分析並且依據依賴程度排序。

紅字部分為1.2章節依據醫學知識針對原先欄位做裝箱、合併...等。而透過以下25個欄位介紹，我們可發現挑選出來的feature間也有一些具有不小的相關性，如平均動脈壓、第一日測量的最高收縮壓，以及當初根據血氧濃度新增的spo2_level其實就是拿d1_spo2_min做裝箱，不過，我們仍可發現在本資料集中，和病人死亡有關的一些欄位可能包含了血壓、呼吸頻率、心肺功能以及一些如血糖、血鉀等因素，因此在未來若是有機會開發一套健康檢測系統，這些項目會是必須考量的項目。

1. apache_4a_hospital_death_prob:

根據 Apache 4A 模型預測的病人在住院期間死亡的機率。

2. apache_4a_icu_death_prob:

根據 Apache 4A 模型預測的病人在 ICU 期間死亡的機率。

3. age:

病人的年齡。

4. d1_heartrate_min:

第一日內測量的最低心率。

5. ventilated_apache:

是否使用呼吸器。

6. d1_resprate_min:

第一日內測量的最低呼吸率。

7. d1_spo2_min:

第一日內測量的最低血氧飽和度。

8. gcs_overall:

格拉斯哥昏迷指數（GCS）總分，衡量病人的意識水平。

9. d1_mbp_min:

第一日內測量的最低平均動脈壓。

10. apache_3j_bodysystem:

病人的主要器官系統分類

11. apache_3j_diagnosis:

病人的主要診斷分類代碼

12. d1_temp_min:

第一日內測量的最低體溫。

13. d1_sysbp_max:

第一日內測量的最高收縮壓。

14. d1_sysbp_min:

第一日內測量的最低收縮壓。

15. d1_resprate_max:

第一日內測量的最高呼吸率。

16. icu_admit_source:

病人被收治入 ICU 的來源（如急診室、手術後等）。

17. d1_temp_max:

第一日內測量的最高體溫。

18. bmi:

病人的身體質量指數（Body Mass Index）。

19. d1_glucose_min:

第一日內測量的最低血糖值。

20. d1_potassium_max:

第一日內測量的最高血鉀值。

21. d1_heartrate_max:

第一日內測量的最高心率。

22. pre_icu_los_days:

病人進入 ICU 前在醫院住院的天數。

23. resprate_apache:

基於 Apache 評分的呼吸率數據。

24. spo2_level:

血氧飽和度水平

25. breathe_combine:

綜合插管、呼吸器的欄位

(3)實驗結果與討論:

Case 1 實驗結果如下其中，我們可看出在使用相同模型的情況下，使用DownSample會較使用SMOTE優。此外我們也可看出若是以macro F1-Score而言，使用CatBoost會較XGBoost略優一些。

Model	Sampling Method	Average AUROC	Average F1-Score
XGBoost	DownSample	0.8857	0.7287
XGBoost	SMOTE	0.8825	0.7270
CatBoost	**DownSample**	**0.8851**	**0.7295**
CatBoost	SMOTE	0.8833	0.7261

Case 2 實驗結果如下和Case1相似，我們可看出在使用相同模型的情況下，使用DownSample會較使用SMOTE優。此外我們也可看出若是以macro F1-Score而言，使用CatBoost會較XGBoost略優一些。至於和Case1比較時，我們皆拿CatBoost+DownSample之組合進行繳交，不過兩者F1-Score則都是保持0.724不變(圖120)，但考量驗證集與測試集存在0.006有點大的落差，故開始考慮使用類別型之資料(Case3)。且後來考慮inference之穩定度後，我們發現在Case2中其precision、Recall的落差較使用median的case1小，因此，在選擇實作Case3時我們將一樣選擇使用KNN作為Imputer。

Model	Sampling Method	Average AUROC	Average F1-Score
XGBoost	DownSample	0.8855	0.7268
XGBoost	SMOTE	0.8826	0.7265
CatBoost	**DownSample**	**0.8852**	**0.7305**
CatBoost	SMOTE	0.8822	0.7286

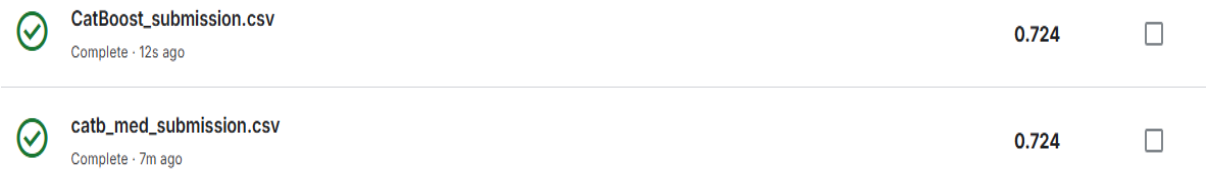


圖120 為Case1、2中表現最佳的組合進行繳交之結果。

Case 3較出乎意料的是，當我們使用KNN針對數值型資料進行插補後，使用支援類別型的XGBoost、CatBoost，兩者在F1-Score部分只有XGBoost比Case1、Case2好，然而出乎意料地在繳交後，發現圖121、122中XGBoost僅有0.720之macro F1 score，**而CatBoost反而是有0.725較佳的macro F1 Score。**

Model	Sampling Method	Average AUROC	Average F1-Score
XGBoost	DownSample	0.8866	0.7271
CatBoost	**DownSample**	**0.8840**	**0.7272**

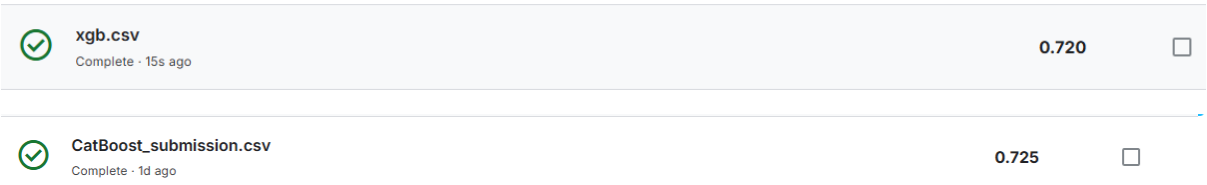


圖121、122 為XGBoost、CatBoost於Case3之表現

(4) 結合特徵取聯集測試效果

最終結合先前提供四種top 20 features的SHAP Bar Chart取聯集後，共有25個欄位，其詳細內容已於本章節前說明。其實驗結果如下，然而XGBoost在測試時效果依然不佳，而使用CatBoost並採用DownSample的組合時，不僅是在驗證集的Average AUROC、Average F1-Score有明顯提升，並且在測試時有提升至0.727的macro F1-score如圖123、124，因此我們將採用CatBoost並採用DownSample的組合，作為本次實驗最終版本。

Model	Sampling Method	Average AUROC	Average F1-Score
XGBoost	DownSample	0.8882	0.7275
CatBoost	**DownSample**	**0.8862**	**0.7314**

4. Supplement

(1) SHAP (以XGBoost為例)

首先，先模擬真實狀況設定5-Fold，並且先處理DownSample使多數類別變成少數類別之2倍數量，並確保不會重複抽樣。接著設立XGBoost模型，並且若為Case3則將允許類別型欄位設True，最終依據SHAP會檢視模型對於各欄位與結果之依賴程度進行排名，並且輸出特徵、長條圖。

```python
import xgboost as xgb
import shap
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.utils import resample

# 初始化 Stratified K-Fold
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# 用於存儲每折 SHAP 值的列表
shap_values_list = []
feature_importance_all_folds = []

for fold, (train_index, val_index) in enumerate(kf.split(train_X_imputed, train_y)):
    print(f"Processing Fold {fold + 1}...")

    # 分割數據
    X_train, X_val = train_X_imputed.iloc[train_index], train_X_imputed.iloc[val_index]
    y_train, y_val = train_y.iloc[train_index], train_y.iloc[val_index]

    # Downsample 多數類別
    train_data = pd.concat([X_train, y_train], axis=1)
    majority_class = train_data[train_data['has_died'] == 0]
    minority_class = train_data[train_data['has_died'] == 1]

    majority_downsampled = resample(
        majority_class,
        replace=False,  # 不進行重複抽樣
        n_samples=int(2 * len(minority_class)),  # 多數類樣本為少數類樣本的兩倍
        random_state=42
    )
    downsampled_train_data = pd.concat([majority_downsampled, minority_class])

    X_train_downsampled = downsampled_train_data.drop('has_died', axis=1)
    y_train_downsampled = downsampled_train_data['has_died']
```

```python
# 初始化 XGBoost 模型
xgb_model = xgb.XGBClassifier(
    tree_method="hist",
    device="cuda",
    n_estimators=8000,
    learning_rate=0.001,
    max_depth=7,
    colsample_bytree=0.8,
    subsample=0.8,
    min_child_weight=3,
    eval_metric='auc',
    random_state=42,
    enable_categorical=True  # 啟用類別特徵支持
)

# 訓練模型
xgb_model.fit(X_train_downsampled, y_train_downsampled)

# 使用 SHAP 解釋模型
explainer = shap.TreeExplainer(xgb_model)
shap_values = explainer.shap_values(X_val)
shap_values_list.append(shap_values)
```

```
    # 計算當前折的特徵重要性
    feature_importance = pd.DataFrame({
        'feature': X_val.columns,
        'importance': np.abs(shap_values).mean(axis=0)
    })
    feature_importance = feature_importance.sort_values(by="importance", ascending=False)
    feature_importance_all_folds.append(feature_importance)

    # 打印前幾個重要特徵
    print("Top features for this fold:")
    print(feature_importance.head(20))

# 彙總所有折的特徵重要性
combined_importance = pd.concat(feature_importance_all_folds)
mean_importance = combined_importance.groupby('feature').mean().sort_values(by="importance", ascending=False)

# 繪製 SHAP 總結圖
shap.summary_plot(np.concatenate(shap_values_list), train_X_imputed.iloc[val_index], plot_type="bar")

# 選擇總體重要性最高的前20個特徵
top_features = mean_importance.head(20).index.tolist()
print("Top features across all folds:", top_features)
```

圖123-125 為SHAP程式碼

(2) DownSample、SMOTE實作

DownSample部分則是將訓練資料的has_died欄位分成0(未死亡是多數)、1(死亡是少數)並且resample時確保不可以重複抽取，並且為下降到為少數類別之2倍，最後再拼接回去即可。

```
X_train, X_val = train_X_selected.iloc[train_index], train_X_selected.iloc[val_index]
y_train, y_val = train_y.iloc[train_index], train_y.iloc[val_index]

# Downsample majority class
train_data = pd.concat([X_train, y_train], axis=1)
majority_class = train_data[train_data['has_died'] == 0]
minority_class = train_data[train_data['has_died'] == 1]
majority_downsampled = resample(majority_class, replace=False, n_samples=int(2 * len(minority_class)), random_state=42)
downsampled_train_data = pd.concat([majority_downsampled, minority_class])
X_train_downsampled = downsampled_train_data.drop('has_died', axis=1)
y_train_downsampled = downsampled_train_data['has_died']
```

圖126 為DownSample程式碼部分

SMOTE部分則是可以設定k_neighbors、sampling_strategy，這裡我將k_neighbors設為5，並且一樣是把少數類別的資料變成多數類別的0.5倍，不過和DownSample不一樣的是，SMOTE是把少數類別擴增上去，而DownSample是把多數類別抽樣降低數量成少數類別的數量。

```
# 初始化 SMOTE
smote = SMOTE(sampling_strategy=0.5, random_state=42, k_neighbors=5)
```

```
# 使用 SMOTE 增強少數類樣本
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

圖127、128 為SMOTE程式碼部分

(3) XGBoost、CatBoost、輸出測試csv程式碼

```
xgb_model = XGBClassifier(
    tree_method="hist",
    #device="cuda",
    n_estimators=n_estimators_total,
    learning_rate=initial_learning_rate,
    max_depth=7,
    colsample_bytree=0.8,
    subsample=0.8,
    min_child_weight=3,
    enable_categorical=True,
    eval_metric='auc',
    random_state=42
)
# Train the model
xgb_model.fit(
    X_train_smote, y_train_smote,
    eval_set=[(X_val, y_val)],
    verbose=100
)
```

```
catboost_model = CatBoostClassifier(
iterations=8000,
learning_rate=0.0015,
depth=7,
verbose=100,

eval_metric='AUC',
random_seed=42,
)

catboost_model.fit(X_train_downsampled, y_train_downsampled, eval_set=(X_val, y_val), verbose=100)
```

```
    # 獲取測試集的預測概率
    test_pred_proba = xgb_model.predict_proba(test_X_selected)[:, 1]

    # 使用平均閾值進行分類
    k = [0.69813263, 0.6684355, 0.5199814, 0.6274654, 0.6933799]

    test_predictions = (test_pred_proba >= np.mean(k)).astype(int)

    # 構建提交格式
    output = pd.DataFrame({
        'patient_id': patient_ids,
        'pred': test_predictions
    })

    # 儲存提交檔案
    output.to_csv('../testing_result.csv', index=False)

    print("Submission file 'testing_result.csv' has been saved with threshold:", np.mean(k))
```

圖129-131為XGBoost、CatBoost、輸出測試csv程式碼程式碼部分