HW2 Report

由於本次作業，主要需實踐 Classification，預測病人是否會過世。因此，本次共嘗試 3 種 classification 的版本：`RandomForest` \ `Xgboost` \ `Neural Network` ，最終決定使用 `Neural Network` 。

Implementation

Data Preprocessing

在本次資料處理中，共處理了以下步驟：

1. `ed_diagnosis` 欄位進行數字化處理：由於 `ed_diagnosis` 紀錄了病人的病徵，但是以 String 形式紀錄，並無法傳入 Model 進行 Prediction 。

   經過測試，得出以下結論：

   > Label Encoding 的方式，會讓 String 轉為 0, 1, 2 ... 的序數。此方法對於 Xgboost ＆Random Forest 兩種方法而言，較優秀的表現。
   >
   > One-Hot Encoder 的方式，將該欄位擴展至 n 個欄位 (n = total number of symptom) 。此方法直覺上，可以避免讓 Model 認為數字有關聯的錯覺。此方法會使 NN Training 較易找出最佳值。

2. `Columns Drop` ： `admission_datetime` \ `PATIENT ID` 三者皆與病人是否過世，理論上無太直接關聯。`sex` 經過 NN 測試、與 xgboost 的內建 optimizer 驗證後，也發現無太大關聯，本次實作將以上 columns Drop。

3. `Fill NAN` ：這次有相當多的數值，都產生 NaN 的數值。 `phmx` 開頭的資料中，大部分都是 `binary` 的數值，我使用 `mode` 進行資料補足。而對於非 `phmx` 開頭的資料，大多都是連續資料，就以 `mean` 進行資料補足。

```
headerList = df.columns.values.tolist()

# need to fill mode
filteredList = list(filter(lambda name: 'pmhx' in name, headerList))

# need to fill mean
otherList = list(filter(lambda name: 'pmhx' not in name, headerList))

for filteredCol in filteredList:
  df[filteredCol].fillna(df[filteredCol].mode()[0], inplace=True)
```

```
for otherCol in otherList:
  df[otherCol].fillna(df[otherCol].mean(), inplace=True)
```

4. 切分 Training Data 與 Testing Data: 使用 `train_test_split` 進行切分，比率設定為 `0.2` 。使用 `train_test_split` 的原因，在於其可以 randomly 切分資料，避免 testing data 都集中在某一區塊。
5. Data Augmentation: 由於 Dead 的資料遠少於 Alive 的資料，在我完成 training / testing 的 data split 後，便開始進行 training data 的 augmentation ，試圖讓 training data dead 與 alive 的數量儘量平均。此方法會讓 NN 表現大幅提升。

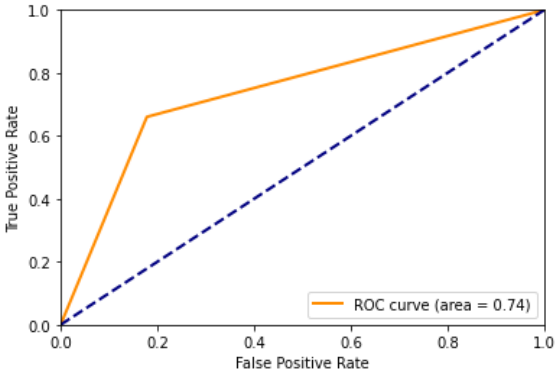
```
from imblearn.over_sampling import SMOTE
from collections import Counter
smote = SMOTE(ratio='minority')
training_data_list, training_ans_list = smote.fit_sample(training_data_list,
 training_ans_list)
```

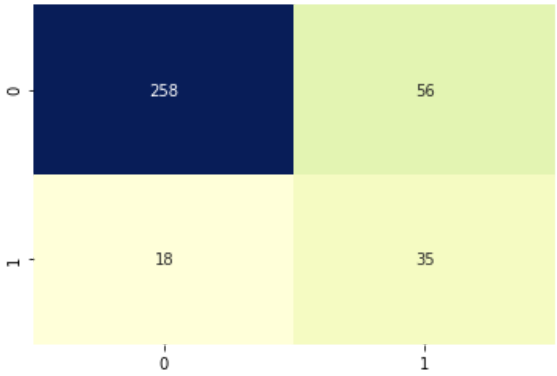
Model Training

Random Forest Classifier

Random Forest 的基本原理是，結合多顆CART樹，並加入隨機分配的訓練資料，以大幅增進最終的運算結果。基本上，這個方法基於下面的理論：Ensemble Method 。Ensemble Method（集成方法）的理論為，當個別分類器都表現尚可，則將多個分類器整合起來，其效能優過於單個分類器。

我們使用 Random Forest 進行測試，在大多數情況下， Precision & Recall Metrics 的加總數值，會介於 `0.9` - `1.0` 左右。當 `n_estimator` 調整至 `1000` 左右，其表現會越佳。



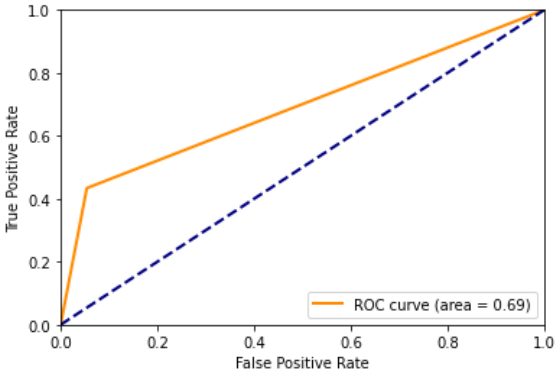


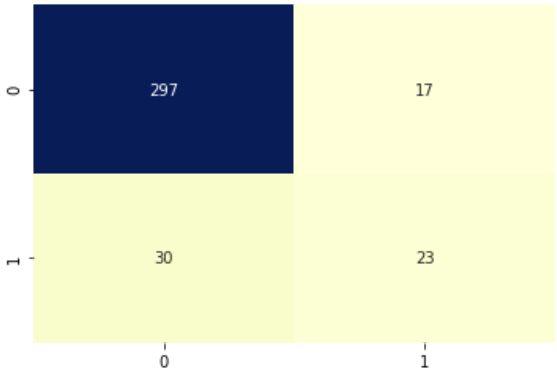
Xgboost

XGBoost (Extreme Gradient Boosting) 是基於 Gradient Boosted Decision Tree ，進行 Model 改良，被應用於解決監督式學習的問題。大致上，以 Greedy Method ，在樹的每層建構過程中，持續優化目標函示。有以下特性：

1. 基於 Tree Ensemble 模型：透過**增量訓練 (additive training)** 的方式，每一次保留原來的模型，再加入一個新的函數至模型中。也就是說，每新的一次 iteration ，皆會在前一步的基礎上增加一顆樹，以利修復上一顆樹的不足。
2. Based on C++：執行速度很快。

我們使用 xgboost 進行測試，在大多數情況下， Precision & Recall Metrics 的加總數值，會介於 `0.9` - `1.0` 左右。當 `n_estimator` 調整至 `1000` 左右，其表現會越佳。



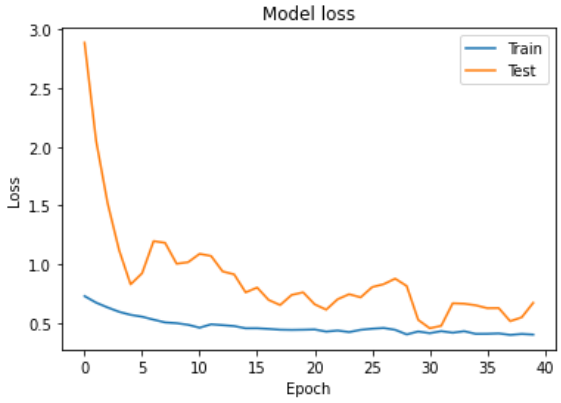


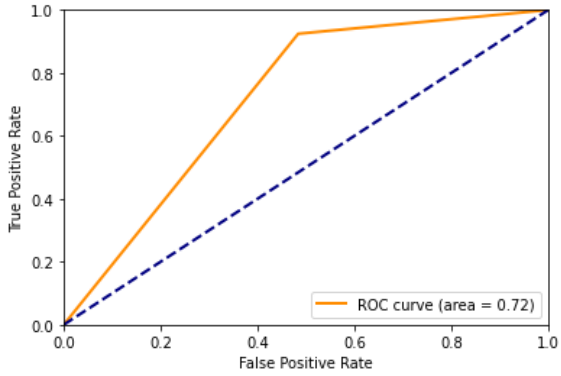
Neural Network

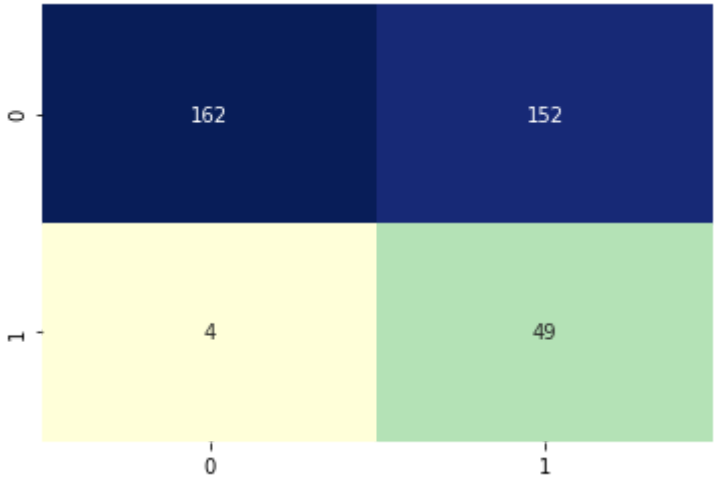
本次使用 Keras ，建造一個基礎的 Neural Network 。由於本次 Model Data 量不大，設計上以 `簡單` （避免過多層為核心目標）。在試過將近百種版本後，得出以下設計：

1. 鑑於資料，本次僅設計三層 Network ，每一層的 Neural 數目不要太高，有較佳效果。
2. Activation Function 根據上課講解，以 `relu` 作為核心使用。試過 `sigmoid` 效果非常差。
3. 由於原先的資料並非介於 0 → 1 的數值，因此在 NN 中加入了 `BatchNormalization` 。
4. 因為 Testing Loss 一直降不下來，所以有使用 `Dropout` ，試圖拉低 Testing Loss 。
5. 因為最終結果是以 Binary Output ，故 Model 以 `binary_crossentropy` ，作為 `loss` 計算標準。
6. 用 `Adam` 作為 `optimizers` ，對比於 `SGD` ，效果相當好。
7. 因為 Deep Learning Workload ，越後面的 Neural 的 Output 會影響越大，有使用 Learning Rate ExponentialDecay ，試圖降低此效應。
8. Data Preprocessing 的好壞（如 label encoding 或是 one-hot 的設計、Data Augmentation），大幅影響了後期 Neural Network 的效能。
9. Batch Size 設定在 300 上下，Epoch 設定在 30 - 40 ，效果較好。

本次花費相當大的心力，在 fine-tuning Neural Network 的參數，目前最好的結果為 Precision & Re call 加總為 `1.19`。







How to use the model

打開 `106062322_HW2_Model.ipynb` 後，直接執行 `Predict Final Result Block` ，即可完成 Predict ，得到 final csv。