Data Mining Lab2 Report

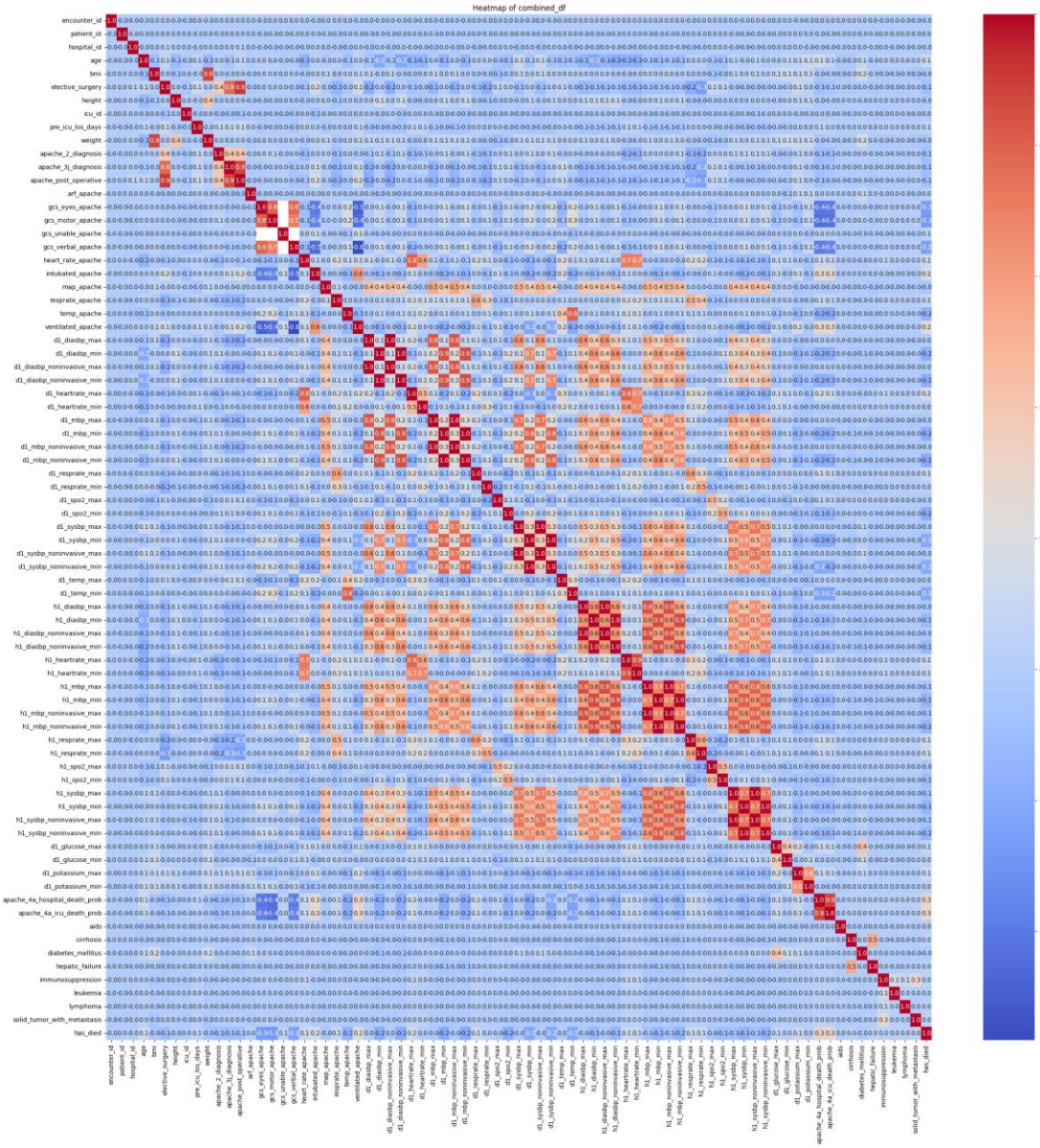
Name: 陳以瑄 Student ID: 109705001

1. Data pre-preprocessing

(1) Dataset analysis

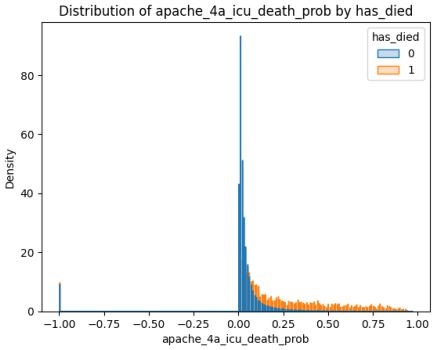
a. Correlation

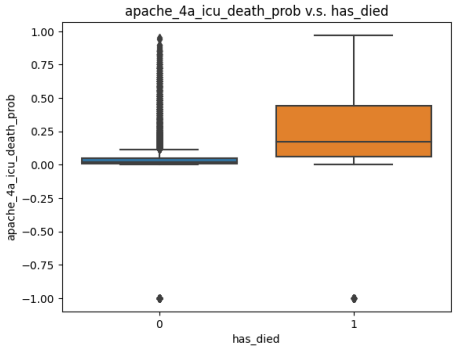
The following figure is the heatmap of all features and the target variable.



Observation 1:

As we can see, some features positively correlate with the target value. For example, the “apache_4a_hospital_death_prob” and the “apache_4a_icu_death_prob”.

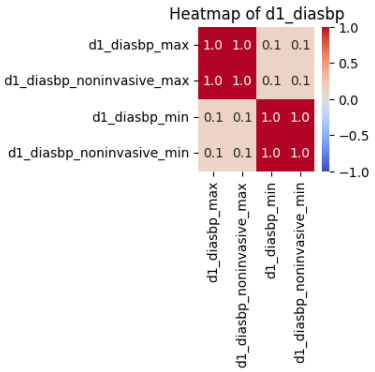




These are the histogram and the boxplot of “apache_4a_icu_death_prob”, we can observe that if the death probability is closer to 0, then the target value is more likely to be 0.

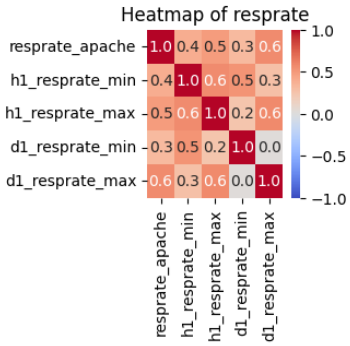
Observation 2:

As the heatmap shows, some features is highly correlated, especially the ‘{feature}_max/min’ and the ‘{feature}_noninvasive_max/min’. For example, 'd1_diasbp_max' and 'd1_diasbp_noninvasive_max'.



Observation3:

The columns with similar names will be positively correlated, but not as highly correlated as in the previous example. For example, the columns that contain ‘resprate’.



(2) Data cleaning

Type1: min greater than max

I've noticed that some rows have anomalies where the '{feature}_min' is

greater than '{feature}_max'. For example, the d1_diasbp_min/max:

```
1 train_X[train_X['d1_diasbp_min']>train_X['d1_diasbp_max']][['d1_diasbp_min','d1_diasbp_max']]
```

	d1_diasbp_min	d1_diasbp_max
574	82.0	46.0
6644	77.0	46.0
10644	50.0	46.0
38922	55.0	46.0
44131	52.0	46.0

We can observe that for this type of anomaly, the max value of the entity is the same, indicating that there is an issue with the max calculation. Therefore, my approach to clean this is to set the 'max' value to NaN for such rows.

Type2: IDs

Some columns pertain to 'id', such as 'encounter_id', 'hospital_id', and 'icu_id'. I have removed these columns from the dataset.

(3) Data transformation

Type1: calculate the difference between min and max

I believe that the difference between min and max values is more crucial than the individual min and max values themselves. Therefore, I have calculated the '{feature}_diff' for all '{feature}_min' and '{feature}_max' pairs. Taking 'd1_diasbp' as an example:

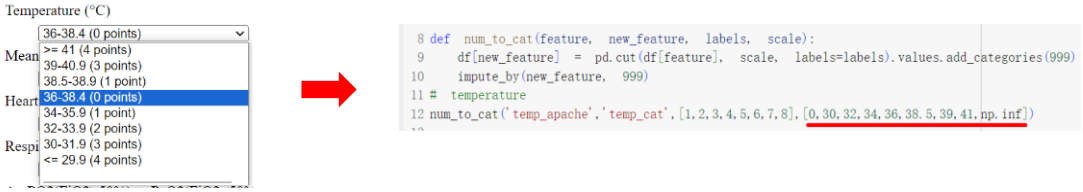
```
1 train_X.filter(like='d1_diasbp')
```

	d1_diasbp_max	d1_diasbp_min	d1_diasbp_diff
0	103.0	41.0	62.0
1	136.0	16.0	120.0
2	84.0	65.0	19.0
3	85.0	28.0	57.0

Type 2: numerical to category

Just as we have reference values during medical check-ups to assess whether our test results are normal, for features related to Apache, I have referred to the threshold values given by "Chiayi Chang Gung Memorial Hospital Apache II Score Site" to categorize continuous data into categorical data. (Reference link: https://www1.cgmh.org.tw/intr/intr5/c6210/APACHE%20II.html)

For example, it split temperature into 9 classes.

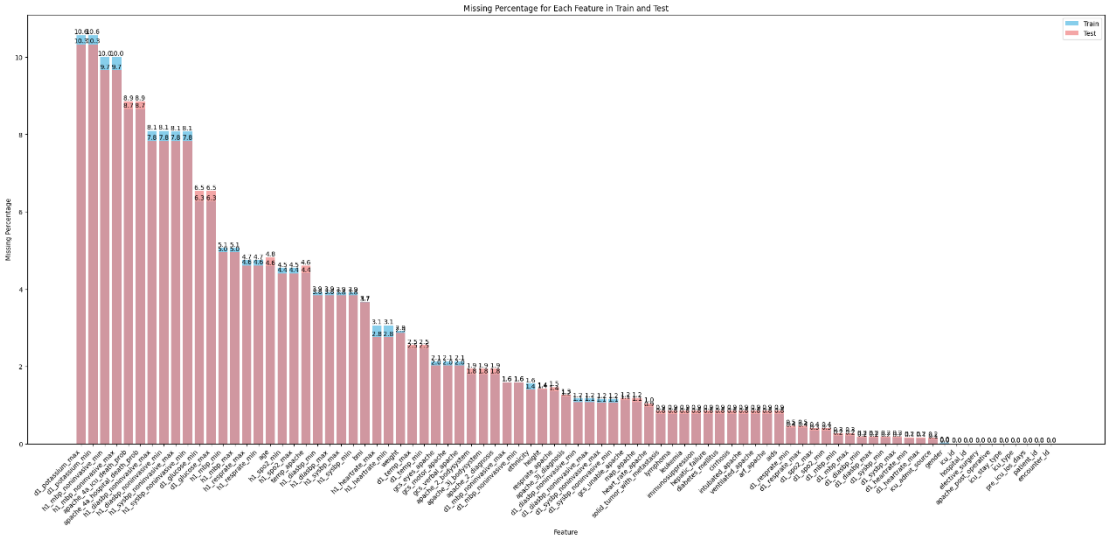


Type 3: category to one hot vector

To prevent potential misinterpretation stemming from the assignment of ordinal numbers to categorical variables and address non-numeric categorical data, I employ one-hot vector encoding on the categorical data.

(4) Data imputation

The following figure shows the missing rate of each column.



As we can see, the column has a higher missing rate in the training dataset and is also high in the testing dataset.

I have categorized the columns with missing values into several groups and applied different imputation methods to each of them.

Type1: Occurred or Not Occurred

Some columns, such as 'aids,' where 1 indicates the presence of HIV/AIDS and 0 indicates the absence. For missing values in this category, I have imputed them as 0, signifying the absence of the condition.

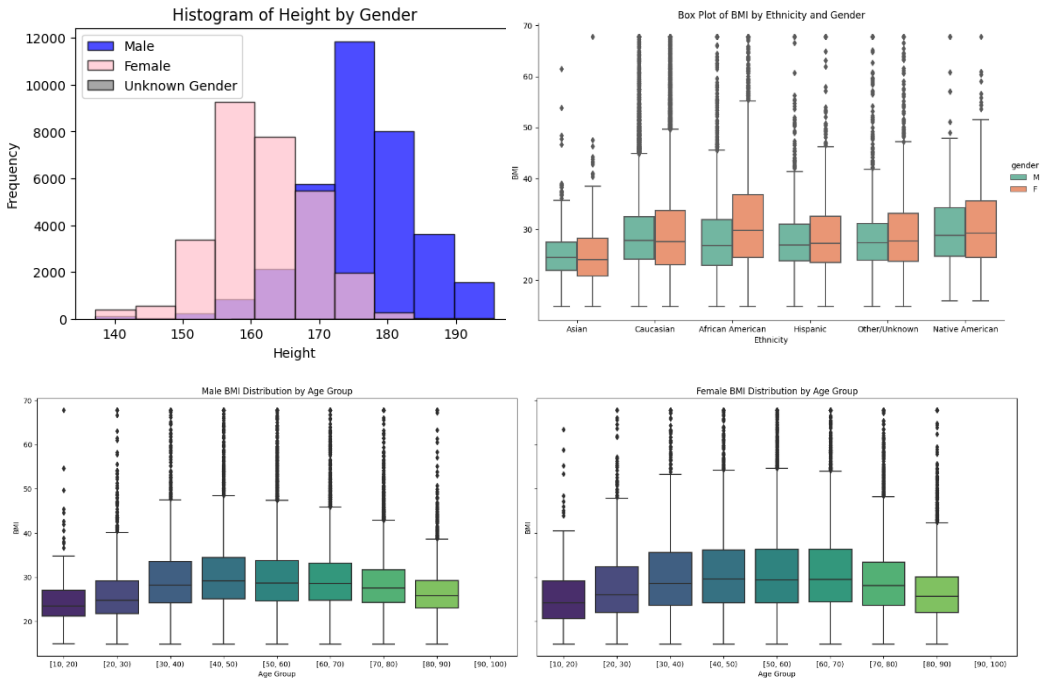
```
#type1  0/1  =>  null就當是沒發生  =>  補0
#  轉移性腫瘤  solid_tumor_with_metastasis
#  淋巴癌  lymphoma
#  血癌  leukemia
#  免疫抑制  immunosuppression
#  肝衰竭  hepatic_failure
#  糖尿病  diabetes_mellitus
#  肝硬化  cirrhosis
#  愛滋病  aids
#  apache系列  ventilated_apache,  intubated_apache,  gcs_unable_apache,  apache_post_operative,  arf_apache,  elective_surgery

TF_f  =  ['solid_tumor_with_metastasis','lymphoma','leukemia','immunosuppression',\
        'hepatic_failure','diabetes_mellitus','cirrhosis','aids','elective_surgery',\
        'ventilated_apache','intubated_apache','gcs_unable_apache','arf_apache','apache_post_operative',\
        'gcs_eyes_apache','gcs_verbal_apache','gcs_motor_apache']

for  feature  in  TF_f:
        impute_by(feature,  0.0)
```

Type2: 'nan' Indicates Unknown

For certain columns like 'gender,' where missing values signify unknown information and cannot be arbitrarily imputed, I have introduced a new category, 'None', for this type of column.

```
#type2  categorical  data  =>  不知道的填不知道
impute_by('ethnicity','None')
impute_by('gender','None')
impute_by('age',-1)
impute_by('icu_admit_source','None')
impute_by('apache_4a_icu_death_prob',-1)
impute_by('apache_4a_hospital_death_prob',-1)
```

Type3: height, weight, and bmi



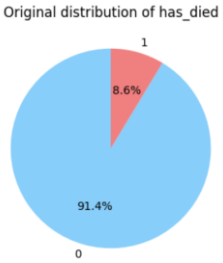
The top-left plot shows that different genders have different height distributions. From the top-right box plot, we can observe that people of different genders and ethnicities will have different BMI distributions. From the two bottom box plots, we can infer that different ages and genders will

result in different BMI distributions.

Therefore, I imputed missing values for each (ethnicity, gender, age) group using the mean of that specific group. For entities lacking ethnicity or age information, I used the gender-specific average for imputation. If gender information was also missing, I utilized the overall average for imputation. Subsequently, I filled in missing BMI values by calculating weight divided by the square of height.

Others:
For the rest columns, I just impute by mean.

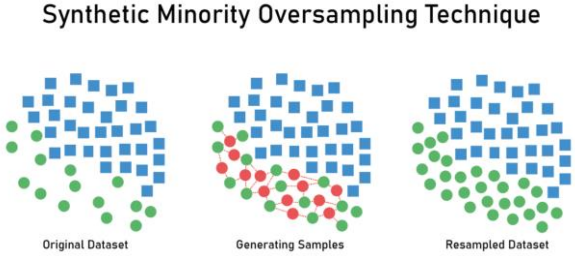
(5) Data imbalance handling



As the pie plot shows, the dataset is highly imbalanced. Therefore, I combine two methods, SMOTE and Tomek to handle the data imbalance problem.

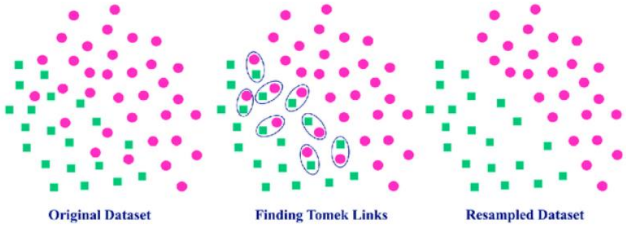
```
smote = SMOTE(random_state=42, sampling_strategy=0.4)
tomek = TomekLinks()
smote_tomek = SMOTETomek()
X_resampled, y_resampled = smote_tomek.fit_resample(X, y)
```

SMOTE is a technique that generates synthetic samples for the minority class by interpolating between existing minority class instances. This helps in balancing the class distribution and improves the performance of machine learning models by providing them with more representative training data for the minority class.



Source: https://medium.com/@parthdholakiya180/smote-synthetic-minority-over-sampling-technique-4d5a5d69d720

Tomek links are pairs of instances, one from the minority class and one from the majority class, that are close to each other but belong to different classes. Removing these instances can help in creating a clearer separation between the classes and improve the decision boundary of the model.



Source: https://www.researchgate.net/figure/llustration-of-Tomek-Links-technique_fig3_374101430

The reason why I combine them is to take advantage of the strengths of both methods. SMOTE helps in oversampling the minority class, addressing the scarcity of instances in that class, while Tomek links provide a means to clean the data by removing potentially noisy or ambiguous samples near the decision boundary. This combined approach aims to create a more balanced and refined dataset.

2. Classification Methods

(1) Model selection

The machine learning algorithm I selected is LightGBM. This choice was based on comparing the performance of various algorithms using PyCaret, where LightGBM demonstrated the best F1 score in a 5-fold cross-validation.

```
compare_models(fold = 5)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9552	0.9871	0.9209	0.9777	0.9484	0.9089	0.9102	8.4200
et	Extra Trees Classifier	0.9549	0.9877	0.9147	0.9833	0.9478	0.9082	0.9101	16.7580
xgboost	Extreme Gradient Boosting	0.9540	0.9864	0.9223	0.9734	0.9472	0.9065	0.9075	5.3680
rf	Random Forest Classifier	0.9502	0.9878	0.9221	0.9649	0.9430	0.8988	0.8996	26.5720
gbc	Gradient Boosting Classifier	0.9497	0.9846	0.9166	0.9692	0.9422	0.8977	0.8988	67.8520
ridge	Ridge Classifier	0.9476	0.0000	0.8856	0.9969	0.9380	0.8929	0.8976	0.5600
lda	Linear Discriminant Analysis	0.9476	0.9827	0.8856	0.9969	0.9380	0.8929	0.8976	2.5000
lr	Logistic Regression	0.9422	0.9774	0.9085	0.9601	0.9336	0.8825	0.8836	17.5280
ada	Ada Boost Classifier	0.9366	0.9800	0.9164	0.9402	0.9282	0.8714	0.8717	15.0000
dt	Decision Tree Classifier	0.9093	0.9088	0.9041	0.8942	0.8991	0.8167	0.8167	3.2320
nb	Naive Bayes	0.8771	0.9289	0.9109	0.8306	0.8689	0.7537	0.7566	0.3500
knn	K Neighbors Classifier	0.8530	0.9312	0.9186	0.7879	0.8482	0.7074	0.7152	15.5080
svm	SVM - Linear Kernel	0.7842	0.0000	0.8334	0.8235	0.7927	0.5828	0.6210	6.4360
qda	Quadratic Discriminant Analysis	0.5823	0.6199	0.9754	0.5186	0.6768	0.2216	0.3249	2.4200
dummy	Dummy Classifier	0.5529	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2200

(2) LGBM

Algorithm description:

Light GBM is a gradient-boosting framework that uses tree-based learning algorithm. Unlike other tree-base algorithms, where the nodes are split level by level, creating a symmetric tree; LightGBM follows a leaf-wise strategy, selecting the leaf with the maximum loss reduction to split. This approach tends to result in lower loss compared to the level-wise strategy.

Other algorithm LightGBM algorithm

Reference:

1. Paper: Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
2. LightGBM site: https://lightgbm.readthedocs.io/en/stable/index.html

(3) K-fold cross validation

I use 6-fold cross-validation in this assignment. The following grid shows how it separate into training and validation dataset:

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	score
Iteration1	validate	train	train	train	train	Train	➔ $F1_1$
Iteration2	train	validate	train	train	train	Train	➔ $F1_2$
Iteration3	train	train	validate	train	train	Train	➔ $F1_3$
Iteration4	train	train	train	validate	train	Train	➔ $F1_4$
Iteration5	train	train	train	train	validate	Train	➔ $F1_5$
Iteration6	train	train	train	train	train	Validate	➔ $F1_6$

➔ Average performance

(4) Implement:

Since I could not resolve the randomization issue in PyCaret, I switched to the LightGBM package, which was provided by Microsoft. And using the scikit-

learn package to do the 6-fold cross-validation and count the AUROC and macro F1 score.

The following part is the source code

The first part is the parameter setting. These are the parameter settings I used to obtain the scores on the leaderboard (public score: 0.696).

```
params = {
        'objective': 'binary',
        'metric': 'binary_logloss',
        'boosting_type': 'gbdt',
        'learning_rate': 0.10,
        'num_leaves': 33,
        'max_depth': 11,
        'feature_fraction': 0.7,
        'random_state':0
}

n_splits = 6
skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)
```

This is the training and validating part. I first fit the LGBM model with the training dataset, then calculate its AUROC and macro f1 score on the validation dataset. Since we need to find out the feature importance, so I save the best model with the highest f1 score.

```
for fold, (train_idx, valid_idx) in enumerate(skf.split(train_X_no_id, train_y)):
        X_train, X_valid = train_X_no_id.iloc[train_idx], train_X_no_id.iloc[valid_idx]
        y_train, y_valid = train_y.iloc[train_idx], train_y.iloc[valid_idx]

        model = lgb.LGBMClassifier(** params)
        model.fit(X_train, y_train)

        y_pred_prob = model.predict_proba(X_valid)[:, 1]

        auroc = roc_auc_score(y_valid, y_pred_prob)
        auroc_scores.append(auroc)

        y_pred_binary = (y_pred_prob > 0.5).astype(int)

        f1_macro = f1_score(y_valid, y_pred_binary, average='macro')
        f1_scores.append(f1_macro)

        if f1_macro > best_f1:
                best_f1 = f1_macro
                best_model = model
```

And the next part is to show the feature importance plot. I retrieve the most

importance features by using ‘best_model.feature_importances_’.

```
feature_importance = best_model.feature_importances_
feature_names = X_train.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
top_features = feature_importance_df.head(20)
plt.figure(figsize=(30, 6))
plt.bar(top_features['Feature'], top_features['Importance'])
plt.ylabel('Importance')
plt.title('Top 20 Feature Importance')
plt.xticks(rotation=45)
plt.show()
```

Lastly, this is how I save the result file.

```
test_pred = best_model.predict(test_X_no_id)
ans_df = pd.DataFrame({'patient_id': test_X.iloc[:, 0], 'pred': test_pred})
ans_df.to_csv("LGBM.csv",index=False)
```

Related package:

1. PyCaret
   Site: https://pycaret.org/
2. LightGBM
   Site: https://lightgbm.readthedocs.io/en/stable/
3. Scikit-learn
   Site: https://scikit-learn.org/stable/modules/model_evaluation.html

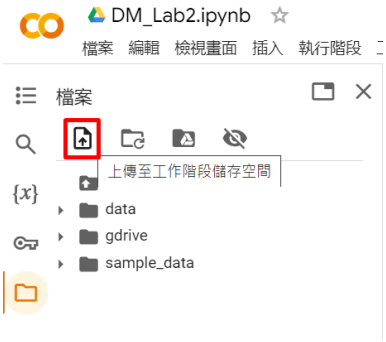
(5) Reproduce the results

I run my code on google colab.

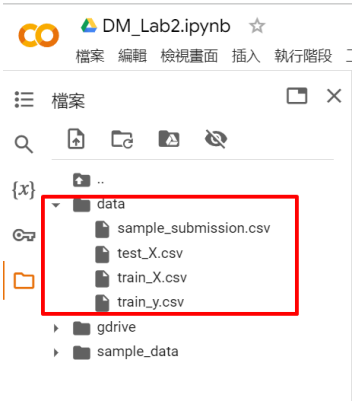
Step1: read or up load the .csv files to ‘data’ folder

You can change the relative file path in cell 2 and run the following code

Or you can just simply upload the data by this button

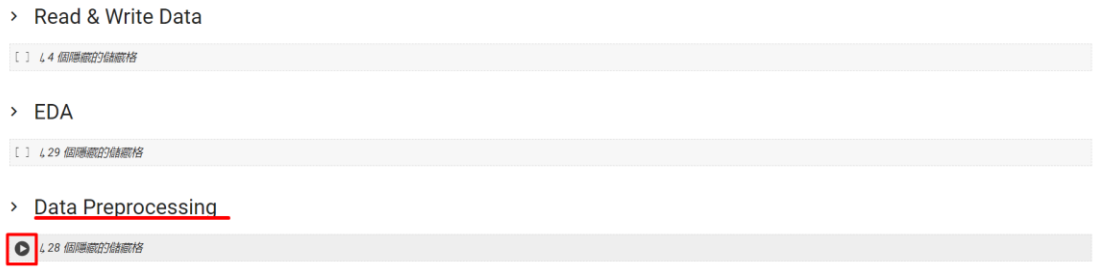


Make sure your files are under the ‘data’ folder like this.



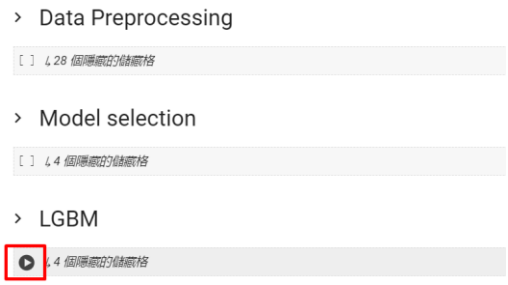
Step2: run the data preprocessing part

Click this button to run all the data preprocessing part.

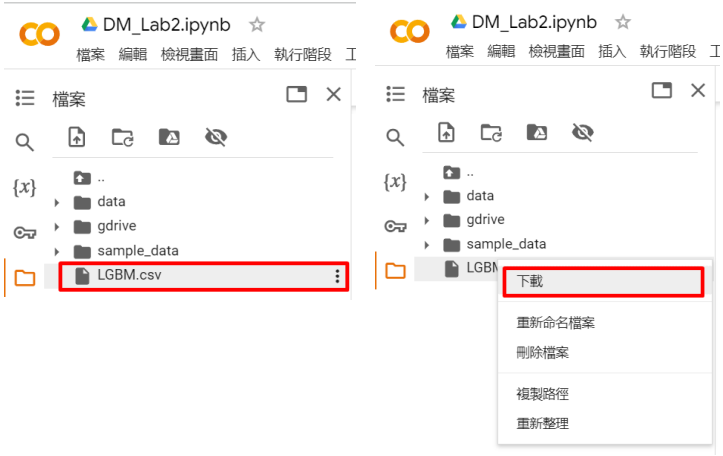


Step3: run the LightGBM

Run this cell to get the result



And you can find the result file here.



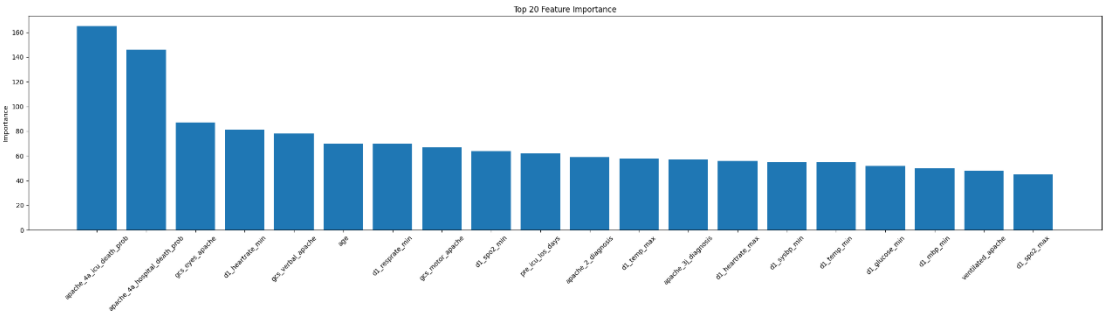
3. Results & Analysis

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

The average AUC is 0.9871 and the average F1 score is 0.95513 after the 6-fold cross validation.

```
Fold 1: AUROC = 0.9872674532816279, F1 Score = 0.9571926180967947
Fold 2: AUROC = 0.9880356325102237, F1 Score = 0.9573801864061307
Fold 3: AUROC = 0.9867445890982325, F1 Score = 0.9560548370475577
Fold 4: AUROC = 0.9872789711709364, F1 Score = 0.9530770312272272
Fold 5: AUROC = 0.9850088155444078, F1 Score = 0.9511328951362136
Fold 6: AUROC = 0.9883767167307985, F1 Score = 0.9559893792791658

Average AUROC across 6 folds: 0.9871186963893712
Average Macro F1 Score across 6 folds: 0.9551378245321817
```

(2) Explainable experiment



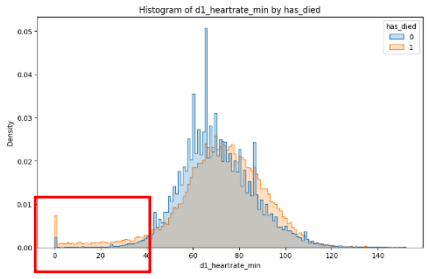
These are the Top 20 most important features.

The first 2 is the apache_icu_death_prob and the apache_hospital_death_prob, this is not that surprise since we have notice that these two features are positively correlated to the target in the data analysis part.

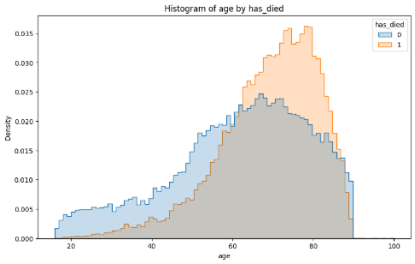
The 3, 5 and 8 place is gcs_eyes_apache, gcs_verbal_apache, and gcs_motor_apache, they are the three aspects of the Glasgow Coma Scale.

(reference link: https://biic.ee.nthu.edu.tw/blog-detail.php?id=10) This shows that my model is highly influenced by the Glasgow Coma Scale (GCS) score.

The fourth place is the d1_heartrate_min. This surprised me a bit because the relationship between the two was 0 on the heatmap. However, when I plotted the histogram, I found that at very low heart rates (<20), the target value is more likely to be 1.



The sixth place is age, this makes sense because generally, the higher the age, the higher the probability of mortality.



Most of the remaining features are related to Apache, which is a Severity Scoring System. This confirms that the indicators used by this scoring system can indeed assist in predicting mortality rates.