HW2 Report

311554021 張祐閣

I. Data Preprocess

This dataset is about the information of the patients on admission to a hospital, and our task is to predict if a patient died or not, so we can expect that this dataset might be strongly imbalanced, and the statistical result shows in Figure 1. support this assumption. As a result, the most importanting thing to do in the data preprocessing stage is to solve this imbalanced data distribution. Before splitting the data into training and validation sets, I first remove the columns representing all kinds of IDs, which are usually considered as noise information.

```
data = data.drop(['encounter_id', 'icu_id', 'patient_id', 'hospital_id'], axis=1)
```

I upsample the data that represent a dead patient to one-third of the training set as shown in Figure 2. The upsampling is done by resampling the 'died' data with an amount of ⅓ 'survived' data. Other things that I do in data preprocessing include converting object type data to int and normalizing the data and the missing values are automatically handled by the classification model I use.

```
for c in data.columns:
        if data[c].dtype.name == "object":
            data[c] = pd.factorize(data[c], sort=True)[0].astype(int)

data = preprocessing.scale(data)
```

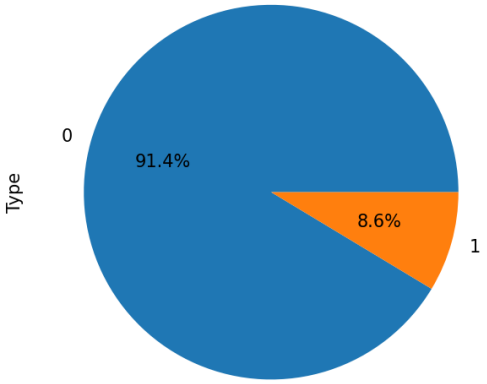

Figure 1.

The pie-chart shows that over 90% of the patients survived in this dataset.

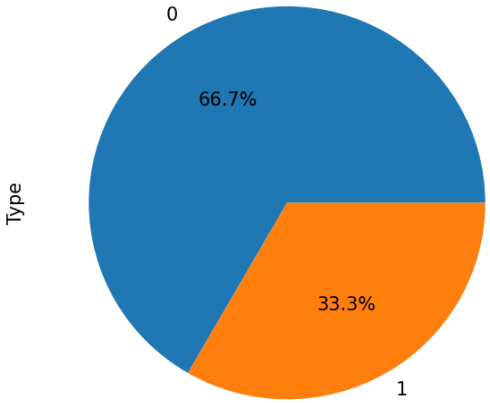

Figure 2.

After upsampling, now the training split contains one-third of dead patients data.

II. Classification Method

I use XGBoost as the classification model. XGBoost is a decision-tree based method that ensembles multiple decision trees with a gradient boost framework to build a strong classifier or regressor. A simple structure of XGBoost is shown in Figure 3. XGboost has been the most selected algorithm in Kaggle challenge, it is well developed and supports the Python API and can handle missing values by default.

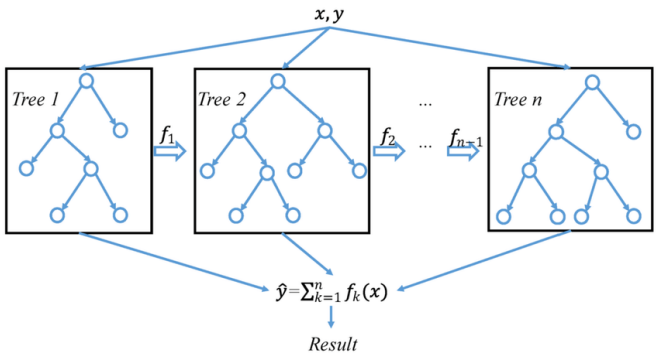

Figure 3.

A simple structure of XGBoost, it ensembles multiple decision trees with a gradient boosting framework.

The implementation is done in Python with several packages that are popular for data analysis and machine learning such as Pandas, Matplotlib, Scikit-learn and of course the classification model I use, XGBoost. The detailed environment setup is described in *environment.yml* that is attached in the hand-in zip file. You can create a same environment with conda command: `conda env create -f /path/to/environment.yml`

III. Results

The validation result is done on the validation set consisting of 30% of the original training data. Figure 4. and Figure 5. show the top 20 features with highest importance and the ROC curve. From Figure 4., we can observe that some basic health indexes such as age, BMI, and weight are indeed important features to predict if a patient can survive at the end. Other measurements that can reflect the patient's immediate vital signs like, heart rate, body temperature are also closely related to the death rate. Figure 6. is the final F1 score and AUROC of validation.

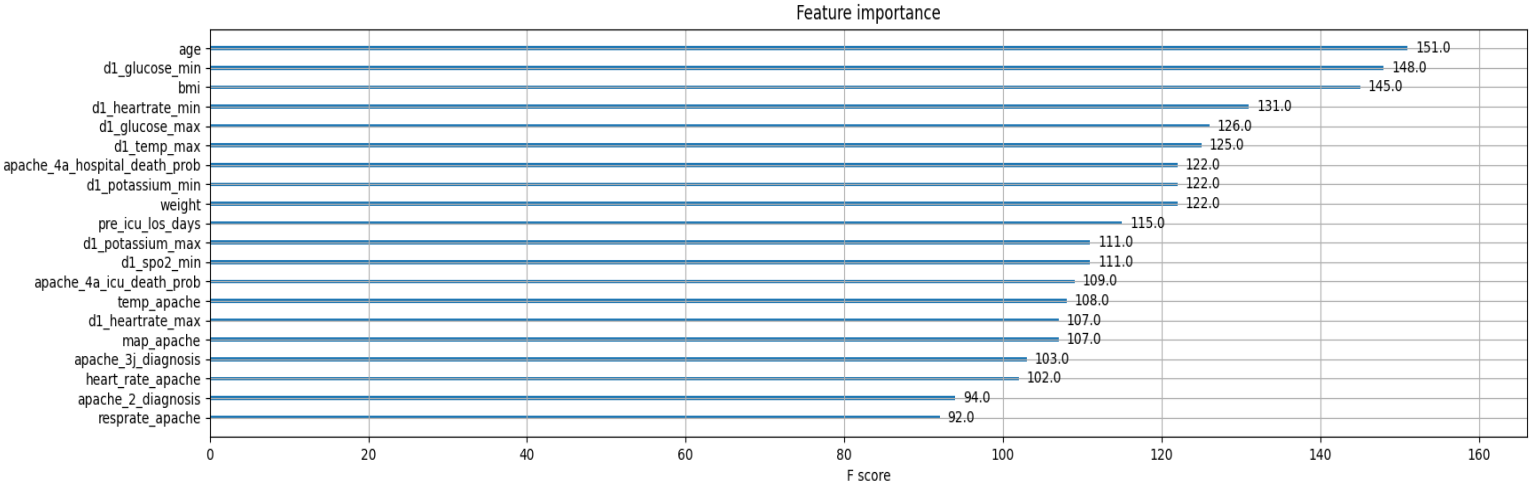

Figure 4.

The top 20 features with highest importance

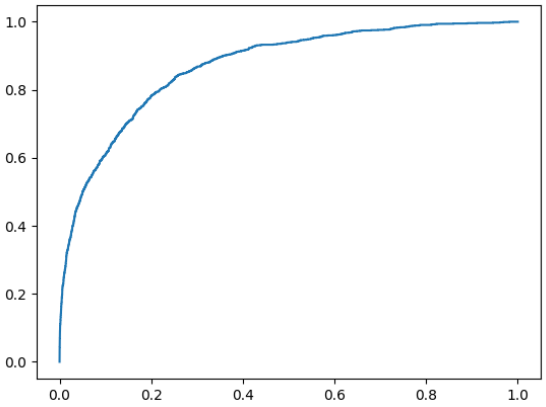

Figure 5.

The ROC curve of validation result

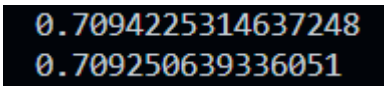

Figure 6.

The resulting F1 score and AUROC of validation