1. Introduction of Research Background

This notebook explores various predictive machine learning models and feature engineering methodologies in addressing research question 1 : predicting hospital mortality within the first 24 hours of ICU stay, utilizing the open-source MIMIC-III dataset. In the subsequent notebook, a multivariable analysis and cox proportional hazards survival model are developed using a filtered version of the MIMIC-III dataset to address research question 2: investigating whether admission to the ICU during weekends increases the risk of ICU mortality. This statistical model not only forecasts the risk factors associated with ICU mortality but also sheds light on the impact of weekend admissions.Python scripting is employed to implement four machine learning models: Logistic Regression, GradientBoostingClassifier, Random Forest and AdaBoostClassifier with SMOTE. Among these models, GradientBoostingClassifier and AdaBoostClassifier with SMOTE demonstrate better performance f1, also exhibiting the highest accuracy levels across both training and test sets. R language is scripted when evaluating research question 2. The findings indicate that ICU-admitted patients face a significantly elevated risk of hospital mortality, influenced by various clinical and non-clinical factors. The goal of this research is to furnish healthcare practitioners with a tool to predict the likelihood of a patient's hospital mortality within the initial 24 hours of ICU admission. Additionally, the statistical model aims to aid hospital management in determining the necessity of resource allocation, including nursing staff and physicians, especially during weekends, with the intention of reducing ICU mortality rates.

2. Data Munging

The data frame can be loaded with the following code: Exploratory Data Analysis: MIMIC-III database consists of 46,520 unique patients and 61,532 distinct ICU stays.

2.1 Datetime format conversion

Firstly, after loading all csv files, i replaced commas in the 'diagnosis' column in admission csv with empty spaces and saved modified dataframe to a new csv file. Then date columns “admittime”, “dischtime”, are converted to datetime format to enable efficient analysis of time-related data. Subsequently, “admittime”, “dischtime” are rounded to the nearest hour, facilitating aggregation and comparison of time intervals.

2.2 Merge Dataset

The datasets were then merged at different steps. Firstly, the new_admissions dataframe is merged with a subset of columns from the pt_icu_outcome dataframe (“subject_id”, “icustay_id”, “age_years”, “icu_expire_flag”, “ttd_days”) based on the common column subject_id. Only rows with matching subject_id values in both dataframes will be kept in merged_df. Then a ranking of “icustay_num” for each “icustay_id” within each unique subject_id group was calculated. “icustay_id” was displayed in order within each group, using tie breaking method “dense”. Later, the merged_df dataframe is further merged with a subset of columns from the icustays dataframe (“icustay_id”, “los”, “icu_intime”) based on the common column “icustay_id”. All rows in merged_df datadrame will be retained in the resulting

dataframe, and missing values will be filled with NaNs for rows where no matches were found. Finally, the resulting dataframe is merged with a subset of columns from the patients dataframe (“subject_id”, “gender”) based on the common column subject_id. Only rows with matching subject_id values in both dataframes are included in the resulting dataframe merged_df.

2.3 Create mp_cohort DataFrame

Additionally, a new dataframe, mp_cohort, was created using selected columns from merged_df to capture key attributes ('subject_id', 'hadm_id', 'admittime', 'deathtime', 'admission_type', 'admission_location', 'discharge_location', 'insurance', 'diagnosis', 'hospital_expire_flag', 'has_chartevents_data', 'admittime_hr', 'dischtime_hr', 'icustay_id', 'age_years', 'icustay_num', 'icu_intime', ‘gender, 'ethnicity', 'ttd_days', 'icu_expire_flag'), focusing on a defined timeframe. Missing ICU Length of Stay, ‘icu_los’ values were filled with 0. A new binary column icustay_24hr_or_less was introduced, indicating whether the ICU stay duration was 24 hours or less. Missing values in ‘hospital_expire_flag’, ‘ttd_days’, and ‘icu_expire_flag’ were encoded 0. A new column 'in_hospital_mortality_within24hours_icu' was created to indicate in-hospital mortality during 24 hours ICU stay where 'hospital_expire_flag’ is 1, and 'icustay_24hr_or_less' is 1. The establishment of exclusion criteria involved the creation of several columns: 'exclusion_adult' identified patients below 16 or above 89 years old; 'exclusion_death_before_icu' flagged cases where 'deathtime' occurred before 'icu_intime'; 'exclusion_valid_data' marked entries with missing 'has_chartevents_data', 'admittime_hr', or 'dischtime_hr'; 'exclusion_short_stay_1hr' identified patient stays equal to or less than 1 hour by calculating the difference between discharge and admission times. Furthermore, 'exclusion_organ_donor' indicated instances where the diagnosis included the term 'organ donor'. These criteria were meticulously established to eliminate irrelevant conditions for future analyses. Subsequently, the resulting dataset, named 'cohort', was saved to a CSV file, prepared for machine learning analyses and modeling.

2.4 Filter cohort and create study_cohort DataFrame

The study_cohort was created by selecting only those rows where the predefined exclusion criteria was not met. Next, duplicated rows of 'icustay_id' were dropped, so each ICU stay was unique in study_cohort. Additionally, missing values were handled by forward filling to maintain data continuity. In preparation for subsequent machine learning analyses, a LabelEncoder was employed. This encoder was applied to columns 'admission_type', 'admission_location', 'discharge_location', 'insurance', 'gender', and 'ethnicity', effectively transforming categorical variables into numerical representations. The resulting label mappings were stored in a dictionary label_mappings. In study_cohort, there are 50,579 distinct ICU stays, and 36,496 distinct patients. In-hospital mortality ratio is 8.62%, 4358 patients died in hospital, 46,221 were not died using 'hospital_expire_flag' = 1.

2.5 New column 'weekend_admission' for question 2

A new column 'weekend_admission' was generated to indicate whether a patient's ICU admission, 'icu_intime' column, fell on a weekend. This binary column takes on values of 1 or 0, with 1 representing an ICU admission during the weekend, and 0 denoting admissions on weekdays. There are 10,893 patients admitted into ICU during the weekend, 39,686 patients otherwise.

2.6 Create dataset final_df for question 2

I combined the gcs_hourly and vitals_hourly DataFrames, as well as the labs_hourly DataFrame, by matching 'icustay_id' and 'hr' values with inner joins. Then, I filled missing values in the merged_hourly DataFrame by carrying forward the most recent non-missing value within each column. The resulting summary_df contains data for 42,355 unique 'icustay_id'. Since some values like FiO2 or respiratory rate may be artificial in ICU settings, I excluded these two and computed the max, min, and difference for each clinical measurement column per 'icustay_id'. Next, I merged study_cohort and summary_df based on 'icustay_id' with an outer join. I filled any missing values in the merged DataFrame using the forward fill method and dropped rows with any remaining missing values, as only a small percentage were missing.Outcome variables are 'icu_expire_flag': 0:survivor/1:died in icu. 'ddt_days':Length of survival (in days)

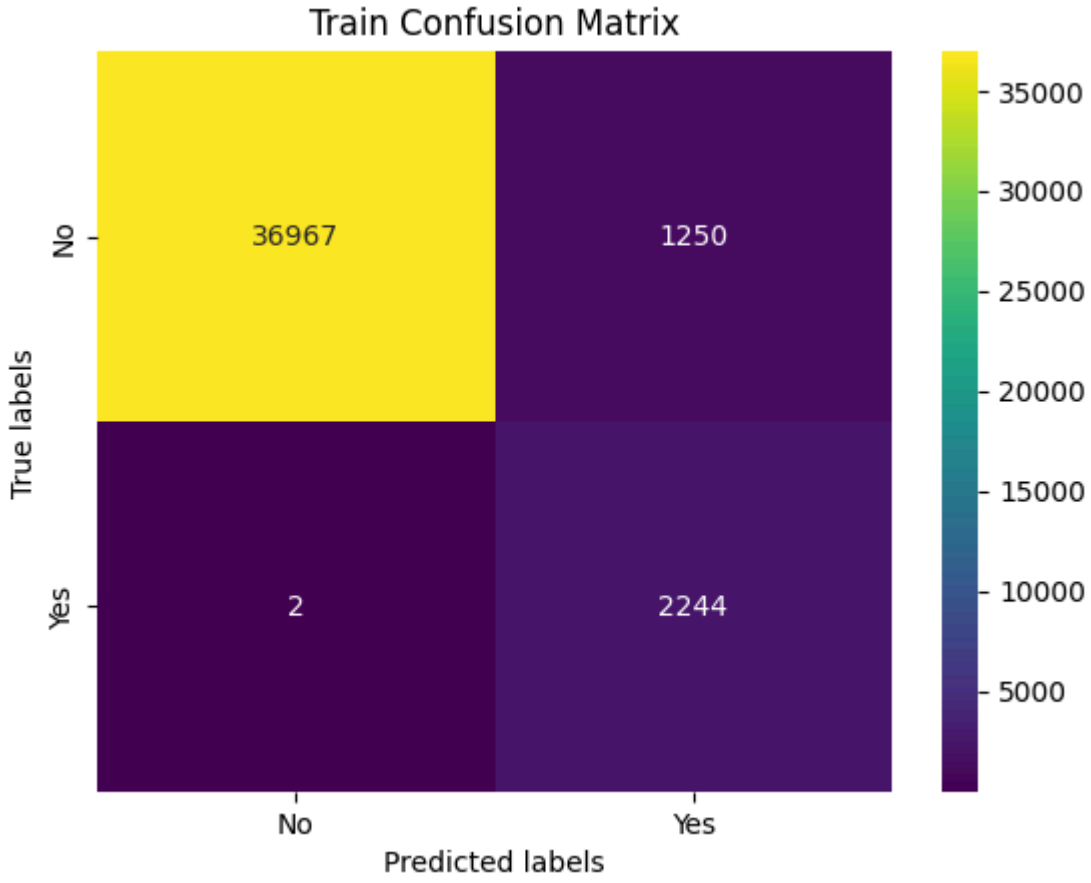
3. Analytics Conducted

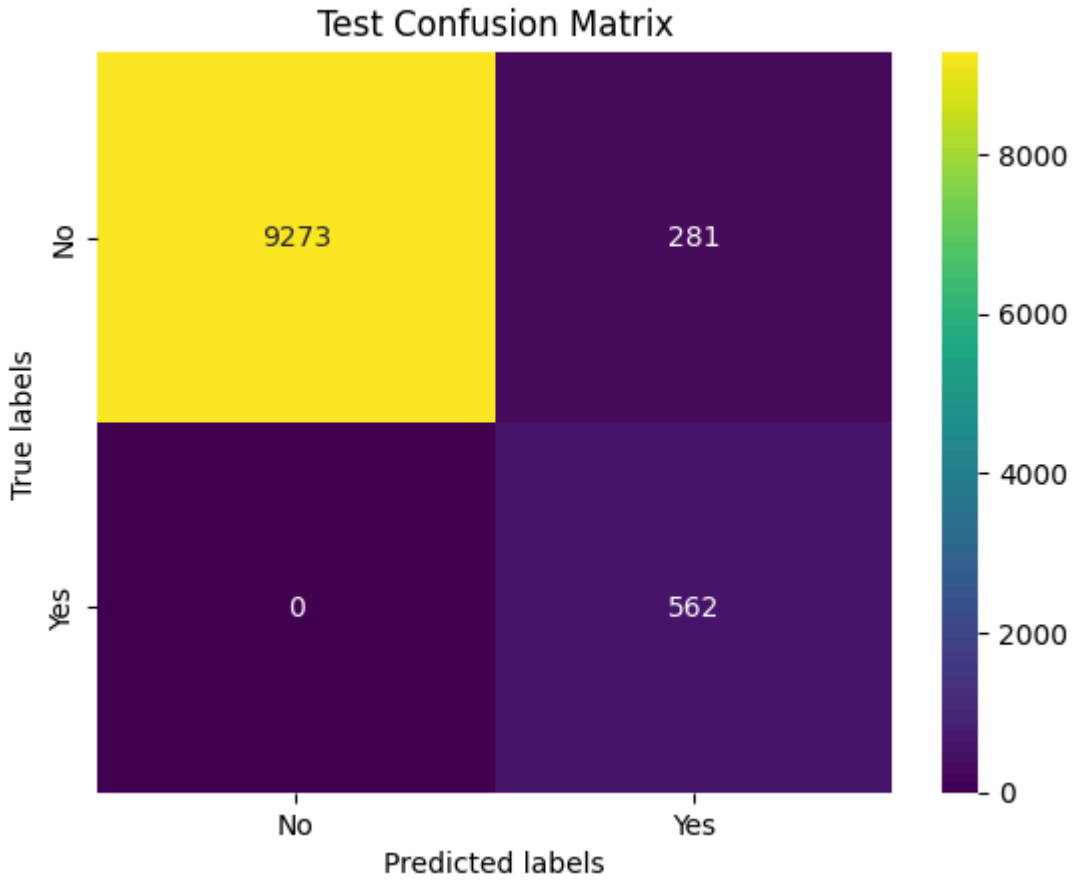
As in_hospital_mortality_within24hours_icu column from study_cohort is already in binary values, representing 0 for "no" and 1 for "yes". I proceeded to split the dataset into two DataFrames: the features, denoted as X (including all variables except for 'subject_id','hadm_id','icustay_id','admittime','diagnosis','hospital_expire_flag','dischtime_hr','in_hospital_mortality_within24hours_icu','icu_intime','admittime_hr','deathtime','icu_los','icustay_24hr_or_less','ttd_days','icu_expire_flag','exclusion_adult','exclusion_death_before_icu','exclusion_valid_data','exclusion_short_stay_1hr','exclusion_organ_donor', 'excluded' columns), and the target is denoted as y (containing only the in_hospital_mortality_within24hours_icu column). There are 47771 patients in class 0, 2808 in class 1, suggesting imbalance of dataset. Then I divided the data into 80% for training and 20% for testing, using stratify=y to preserve the class ratio on both sets, a random seed of 30 was employed to ensure the reproducibility of the data split. The training set was subsequently employed for model training, while the test set served to evaluate the model's generalization performance on unseen data.

3.1 Logistic Regression

The features in training set was standardised using StandardScaler. For logistic regression model, solver='saga' and l1_ratio=0.5 were used to balance L1 and L2 regularisations. The solver=saga supported these fast processing of large datasets. Class weights were applied to handle the imbalanced data (38217 patients in class 0 in y_train, and 2246 patients in class 1) by assigning higher weights to the minority class for accurate predictions. Grid search was used to find the best hyperparameters. The printed results reveal the best hyperparameters discovered by the search and the corresponding F1 score 0.78. Confusion matrices were computated for each set to further assess the predictions. Accuracy score of test and train dataset are 0.972 and 0.969 respectively.

Show code






```
Accurary score of test dataset: 0.9722222222222222
Accurary score of train dataset: 0.9690581518918518
```

3.2 GradientBoostingClassifier Model

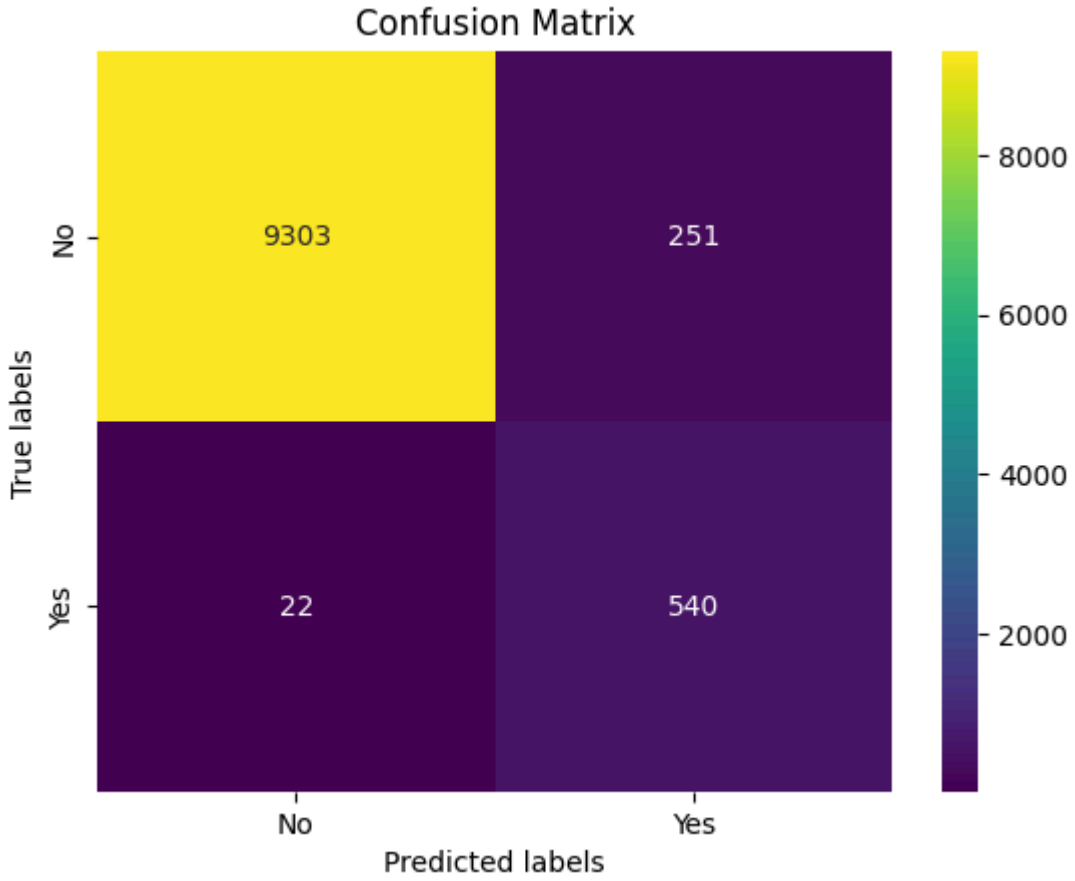
I instantiated GradientBoostingClassifier Model with parameters specifying 40 boosting stages and a fixed random seed 42 for reproducibility. This ensemble technique can handle complex dataset well, and is trained on X_train and y_train, followed by making predictions on test set using gbm_clf.predict(X_test). The accuracy of the model on the test set is 0.973, and overall weighted average F1-score is 0.98. Confusion matrice for each set was computated to further assess the predictions using plot_confusion_matrix function. The classification report was extracted from the confusion matrix for each training and test set.

Show code

```
Accuracy Score on Test Set: 0.9730130486358244
The classification report for test set:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      9554
           1       0.68      0.96      0.80       562

    accuracy                           0.97     10116
   macro avg       0.84      0.97      0.89     10116
weighted avg       0.98      0.97      0.98     10116
```



3.3 Random Forest

A random state of 30 is used to ensure reproducibility during data splitting. A dictionary grid was then created to facilitate grid search, allowing us to explore various hyperparameter combinations. To address the imbalanced nature of the data, class weights were defined, assigning higher weights to the minority class and mitigating biases toward

the majority class. The grid search algorithm iteratively searched for the best hyperparameters, optimizing the model's performance. Upon fitting the model to the training data, we applied the ravel() function to y_train. This conversion from a 2D to a 1D array helps prevent potential errors or inconsistencies during model training. The best hyperparameters were displayed, and the resulting F1 score is 0.75, reflecting the model's performance.

3.4 AdaBoostClassifier with SMOTE

SMOTE was utilized to oversample the minority class in the training data. This approach improves the performance on imbalanced data by generating synthetic samples of the minority class 1 and training a boosting ensemble model on the balanced dataset. The trained AdaBoostClassifier is used to make predictions on the unseen test set (X_test). The accuracy is 0.97, with the weighted average f1-score from classification report is also 0.97. The feature importances is derived from the AdaBoostClassifier, indicating that first 3 most important variables of affecting hospital mortality in first 24 hours icu stays are icustay_num, ethnicity, age_years, admission_location and insurance. Admission type and weekend admission doesn't have high influence on the outcome.

4. Conclusion for Research Question 1

Both GradientBoostingClassifier and AdaBoostClassifier with SMOTE demonstrate good performance across various metrics such as accuracy, precision, recall, and F1-score when predicting hospital mortality within the initial 24 hours of ICU admission. However, the key focus of the research is to correctly classify patients at high risk of mortality. Thus we need to focus on precision and recall for the minority class as well for the reliability and integrity of the predictive model.Based on the classification report findings, the GradientBoostingClassifier model outperforms other three model in predicting class 1, representing patients with hospital mortality within the first 24 hours of ICU stay. This model exhibits a precision of 0.7, indicating its ability to correctly identify class 1 instances 70% of the time, along with the highest recall of 0.96, signifying its capacity to accurately identify 96% of actual instances within class 1. Consequently, it achieves the highest F1-score of 0.8. The AdaBoostClassifier with SMOTE also demonstrates favorable recall (0.99) and precision (0.67), resulting in an F1-score of 0.8 for predicting mortality in the test set. Misidentifying patients at high mortality risk within the initial 24 hours of ICU admission can have severe implications, including adverse health outcomes, inefficient resource allocation, compromised patient safety, and negative repercussions on the hospital's reputation. These consequences may lead to financial penalties or reduced reimbursements from government agencies.

Note: below is the classfication report and confusion matrix of AdaBoostClassifier with SMOTE

Show code

```
Accuracy Score on Test Set: 0.9717279557137208
The classification report for test set:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      9554
           1       0.67      0.99      0.80       562

    accuracy                           0.97     10116
   macro avg       0.83      0.98      0.89     10116
weighted avg       0.98      0.97      0.97     10116
```