Patient Readmission Prediction

Problem Setting

Predicting patient readmission can help healthcare providers identify high-risk patients and optimize discharge planning, follow-up care, and resource allocation. The challenge is to predict accurately whether a patient will be readmitted within 30 days after being discharged based on historical clinical data, demographic information, and hospitalization.

Problem Definition

This research aims at the prediction of readmission within a period of 30 days of discharge from hospital. The following research questions will guide this research:

- What factors have the greatest influence on whether a patient is hospitalized again?
- How do various comorbidities, previous medical history, and severity scores of disease affect readmission?
- Will machine learning algorithms prove to be helpful in predicting such readmissions, thus enabling timely interventions?

OKRs (Objectives and Key Results)

- Objective: Develop predictive models to identify patients at risk of being readmitted within 30 days of discharge
- Key Result 1: Achieve model accuracy of atleast 80% on the test dataset predctions and reducing the false negative rate (patients who are incorrectly predicted as not at risk) to below 10% to ensure timely intervention for patients
- Key Result 2: Improve model interpretability to identify most influential factors for readmission

KPIs (Key Performance Indicators)

- Model Accuracy: Measures the proportion of correct predicitions out of the total predictions, helps to guarantee that the model is reliable in identifying patients at risk of readmission
- Precision: Helps in minimizing number of false positives and also reduces the likelihood of avoidable readmissions
- Feature Importance Scores: Measures the contribution of each feature in the model to identify key factors that influence the result of readmission risk

Data Sources

Our dataset, MIMIC-IV contains de-identified health-related data from over 364,627 patients who have been admitted to critical care units at Beth Israel Deaconess Medical Center (BIDMC).
https://physionet.org/content/mimiciv/3.1/

Data Description

In the development of our readmission prediction model, we extract relevant variables from the following tables of MIMIC-IV:

Dataset	Columns/Features Used	Description
admissions	subject_id, hadm_id, admittime, dischtime, admission_type, discharge_location, insurance	Admission details, including timestamps and discharge status
patients	gender, anchor_age, anchor_year_group, ethnicity	Demographic information
icustays	first_careunit, last_careunit, intime, outtime	ICU stay details
diagnoses_icd	icd_code, icd_version	Patient diagnoses in ICD-9/ICD-10 format
procedures_icd	icd_code	Procedures performed on the patient
labevents	itemid, charttime, value	Laboratory test results (e.g., glucose, creatinine, hemoglobin)
prescriptions	drug, drug_type	Drug counts, prescribed durations
drgcodes	drg_type, drg_code, drg_severity, drg_mortality	DRG info, top DRG (drg does not mean drugs)
emar	medication	Administered medications
transfers	careunit	Unit transfers history

(The above table only represents the tables from which we extract various features from, not the actual features)

<u>Feature Extraction:</u>

Feature	Logic	Extraction Process
subject_id	Unique patient identifier	Directly selected from `admissions` table
hadm_id	Unique hospital admission identifier	Directly selected from `admissions` table
race	Race/ethnicity of the patient	From `admissions` table
admittime	Admission timestamp	From `admissions` table
dischtime	Discharge timestamp	From `admissions` table
hospital_expire_flag	Indicates if the patient expired during hospitalization	From `admissions` table
readmission_flag	1 if the patient was readmitted within 30 days, 0 otherwise	Joined from `readmission` table
num_procedures	Total number of procedures during admission	COUNT(icd_code) from `procedures_icd`
num_unique_procedures	Number of unique procedure codes	COUNT(DISTINCT icd_code) from `procedures_icd`
proc_icd_codes	Concatenated ICD codes sorted by chartdate	STRING_AGG(icd_code ORDER BY chartdate) from `procedures_icd`
num_unique_icd_codes	Number of unique ICD diagnoses	COUNT(DISTINCT icd_code) from `diagnoses_icd`

diag_icd_codes	Concatenated ICD diagnosis codes	STRING_AGG(icd_code ORDER BY seq_num) from `diagnoses_icd`
num_unique_drugs	Number of unique drugs prescribed	COUNT(DISTINCT drug) from `prescriptions` where stoptime >= starttime
num_prescribed_days	Total number of prescription days	"SUM(DATE_DIFF(stoptime, starttime)) from `prescriptions`"
num_prescription_records	Total number of prescription records	COUNT(*) from `prescriptions` where stoptime >= starttime
num_hospital_admissions	Sequential count of hospital admissions	ROW_NUMBER() OVER (PARTITION BY subject_id ORDER BY admittime) from `admissions`
num_prev_emergency	Count of previous emergency visits	"Join on `admissions` filter admission_location = 'EMERGENCY ROOM'"
num_prev_non_emergency	Count of previous non-emergency visits	"Join on `admissions` excluding 'EMERGENCY ROOM' and psych"
num_prev_general_practice	Count of previous general practice unit stays	"Join `transfers` before current admittime filter general practice units"
num_prev_general_surgery	Count of previous general surgery unit stays	"Join `transfers` before current admittime filter surgery units"
num_prev_internal_medicine	Count of previous internal medicine unit stays	"Join `transfers` before current admittime filter internal medicine units"
drg_concat	Concatenated DRG type and code	"STRING_AGG(CONCAT(drg_type drg_code)) from `drgcodes`"
top_drg_code	Top DRG code by severity and mortality	"ARRAY_AGG ORDER BY drg_severity drg_mortality LIMIT 1 from `drgcodes`"
top_drg_type	Top DRG type by severity and mortality	Same as above
top_drg_severity	Top DRG severity	Same as above
top_drg_mortality	Top DRG mortality	Same as above
num_emar_medications	Number of medications administered via EMAR	COUNT(medication) where event_txt = 'Administered' from `emar`
num_abnormal_labevents	Number of abnormal lab events	COUNT(labevent_id) where flag = 'abnormal' from `labevents`
age	Patient age at admission	Joined from processed `age` table
charlson_comorbidity_index	Charlson comorbidity index score	Joined from processed `comorbidity_index` table
procedures (vector embeddings)	All Procedures conducted represented as word embeddings	Used Word2Vec for creating features
diagnoses (vector embeddings)	All diagnoses given represented as word embeddings	Used Word2Vec for creating features

Data Quantity:

- Patients: 364,627
- Admissions: 546,028
- ICU stays: 94,458

Data Understanding

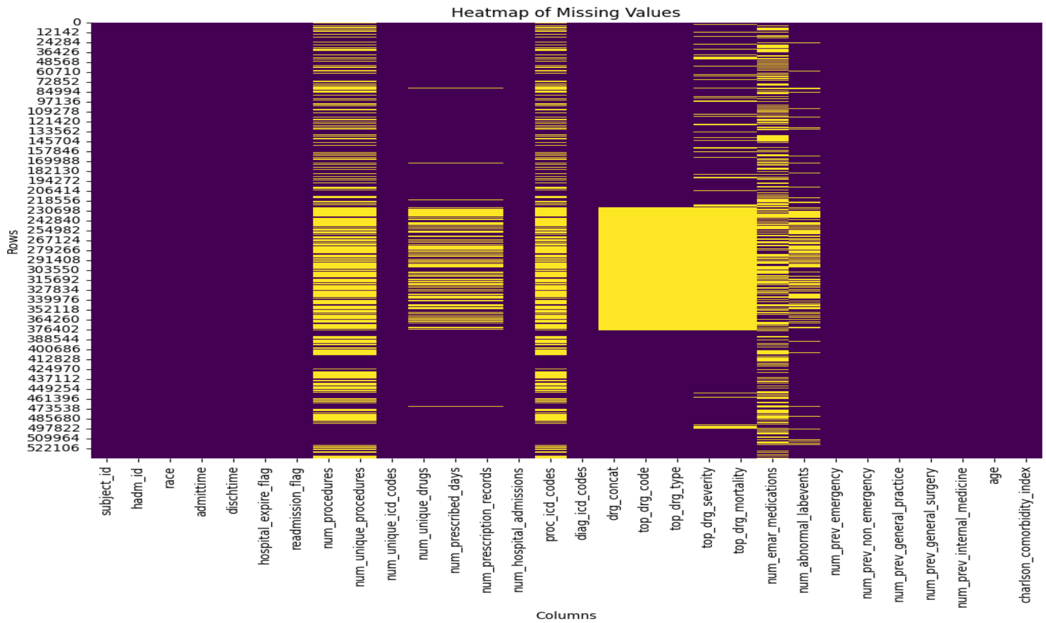
The dataset was collected from the MIMIC-IV v3.1 database, containing health records from over 364,627 patients and 546,028 admissions. Our task is to predict 30-day readmission using clinical and demographic information. The filtered dataset contains:

- 546,028 records
- 32 attributes, including numeric, categorical, embeddings and binary features
- The target variable is 'readmission_flag' (0 = No, 1 = Yes)

Data Pre-processing

Several steps were taken to clean and prepare the data for modeling:

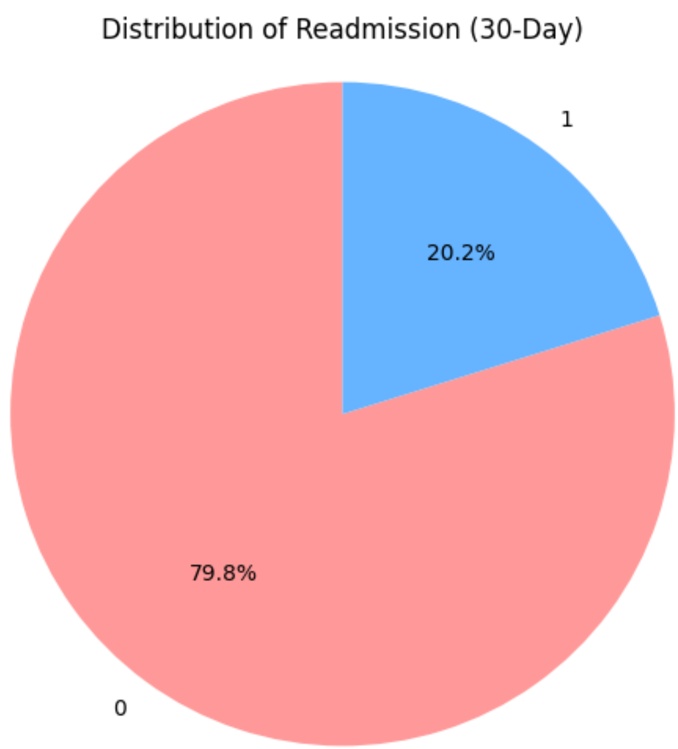
1. Column Selection and Removal:
   - Retained columns from key MIMIC-IV tables (admissions, patients, diagnoses_icd, procedures_icd, labevents, etc.).
   - Removed hadm_id and subject_id, admittime, dischtime from modeling as they are unique identifiers without predictive value.
2. Handling Missing Values:
   - Missing values were identified in the following columns: race, length_of_stay, num_prescribed_days, num_procedures, num_unique_procedures, num_unique_drugs, num_prescriptions, proc_icd_codes, top_drg_code, top_drg_type, top_drg_severity, top_drg_mortality, num_emar_medications, and num_abnormal_labevents.
   - Imputation Strategy:
     - Numerical Columns: According to column and domain knowledge, filled with either 0 or median
     - Categorical Columns (race): Filled with the mode ("UNKNOWN")
   - Columns that are counting number of prcodures, drugs, prescriptions are filled with 0.
   - No columns were dropped due to missing values, as imputation was sufficient



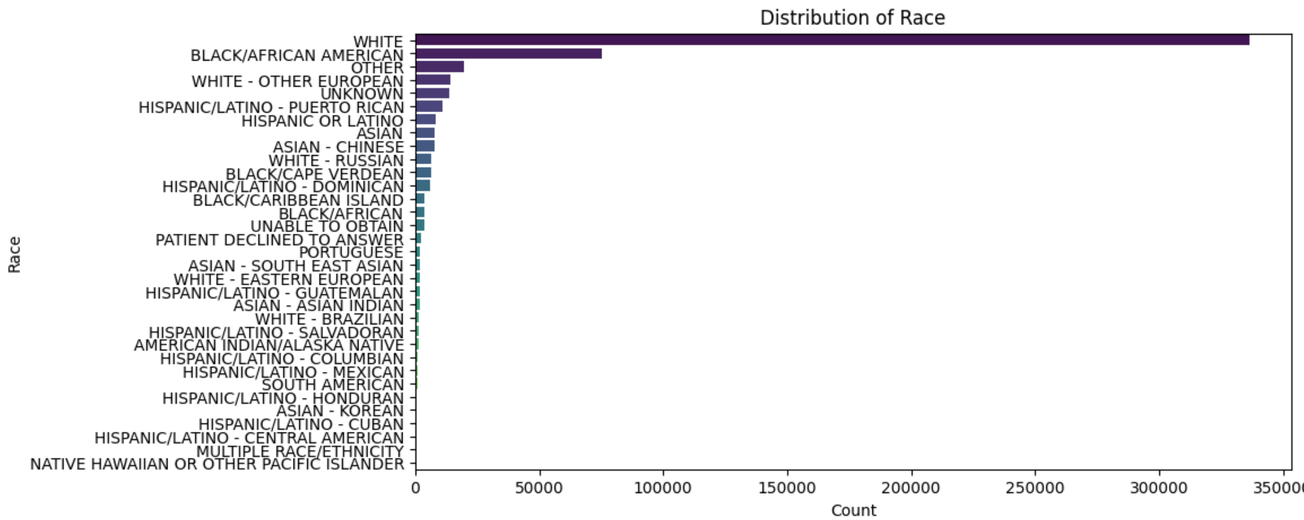
3. Duplicate Records:
    - Checked for duplicate patient-admission pairs and found none.

Data Exploration

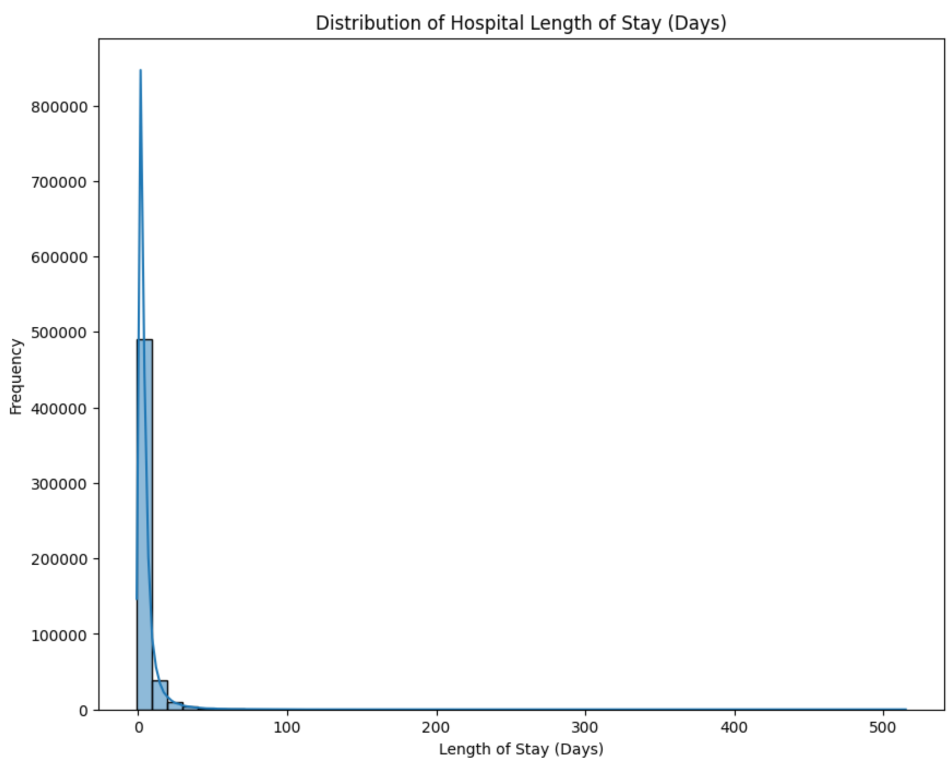
1. Checked for distribution/prevalence of class of interest and found that 20.2% of the patients are getting readmitted



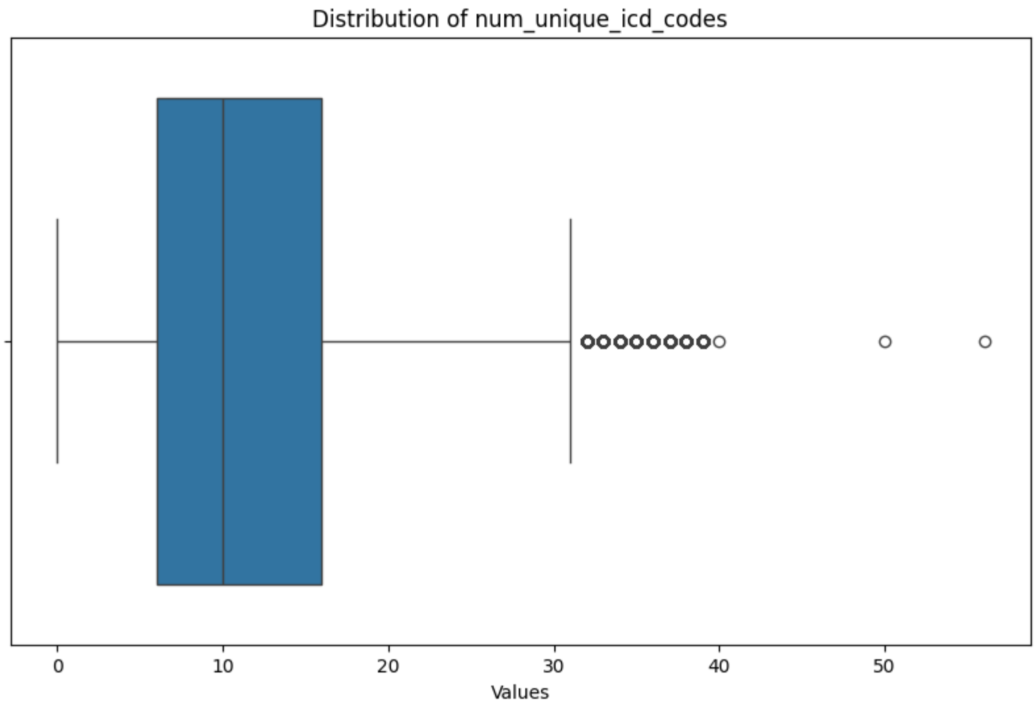
2. Checking for distribution of the race – we can clearly observe that race 'WHITE' is more dominant in the patients



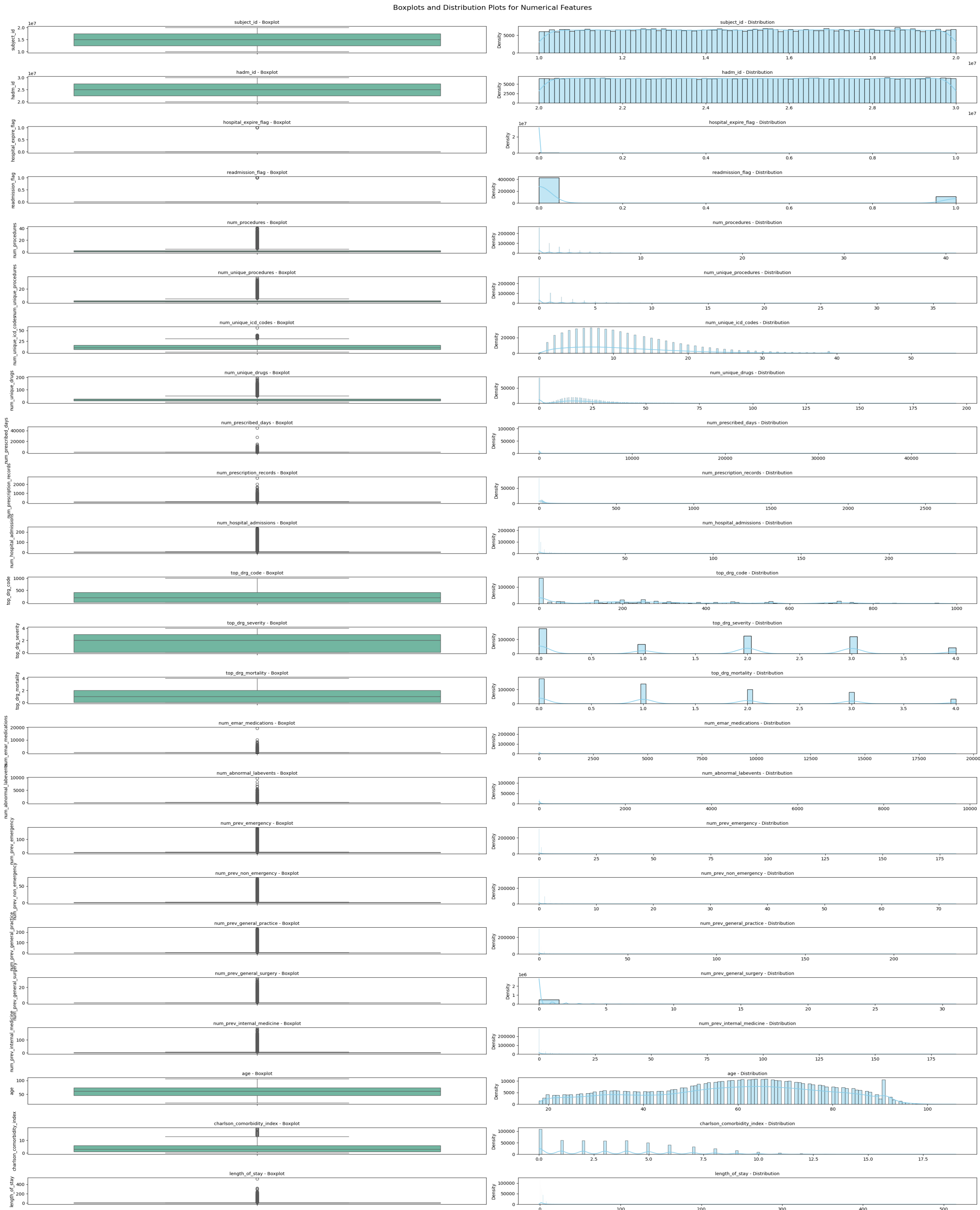
3. Checking the distribution of hospital length of stay in days – it is highly skewed due to patients that might have been in a coma



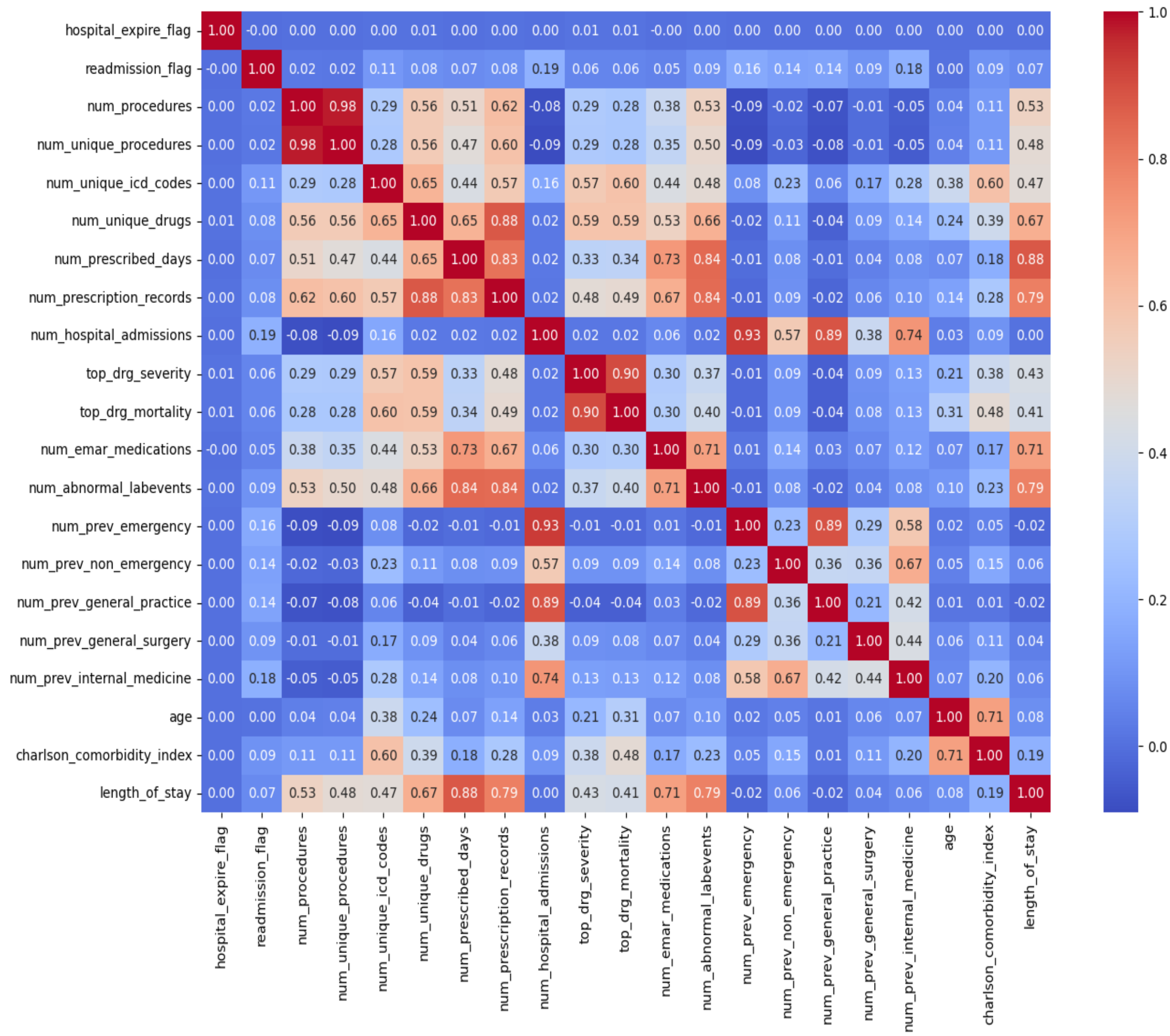
4. Distribution of number of unique diagnose codes per patient



5. Checking overall distribution of numerical columns:



6. Checking the correlation matrix for the numerical columns in the dataset



7. Performing initial PCA to see the variance captured by the data:

```
pca.explained_variance_ratio_
✓ 0.0s

array([0.86915905, 0.07795239])
```

Inference – 86.91% of Variance is captured from all the numerical features in PC1 and 7.79% in PC2!

Data Mining Tasks

In this large dataset, we have a lot of numerical columns and few categorical columns – but the number of categories in these columns are high. For example, the column ‘race’ has more than 10 categories in this dataset. For these categorical columns, we created the dummy variables and that increased the number of categories to 54.

Word2Vec:

In addition to the manual features, we also took freatures from longitudinal health records of the patients. Information like diagnosis codes, medications and procedures are alpha-numeric values, so, we decided to extract feature vectors of these columns. We specifically used Word2Vec (CBOW) algorithm to extract word-embeddings of these features. We began by constructing “patient sentences,” which are sequences of medical codes formed by aggregating all medical data entries - specifically, Diagnosis ICD codes and Procedure ICD codes - associated with a patient. These codes were arranged chronologically to reflect the timeline of events during a hospital admission. For instance, if a patient had a diagnosis code “K65” followed by a procedure code “J01DH,” their sentence would be “K65 J01DH.” In practice, patients often have multiple diagnosis and procedure codes recorded within a single admission.

Using these sentences, we applied a Word2Vec model to learn fixed-length numerical representations (embeddings)(size 100) for each individual medical code. The model captures the contextual relationships between codes: those that frequently appear together in proximity have embeddings that are numerically similar. Once embeddings for all medical codes are learned, each patient’s sentence can be transformed into a sequence of corresponding numeric vectors.

To create a feature vector representing a patient’s medical history, we used a simple method: we summed the embeddings of the most recent 25 medical codes in their sequence. This approach ensures that if two patients share the exact same ordered set of 25 most recent medical codes, they will have identical feature vectors.

The final dataset is of the size:

```
df.shape
✓ 0.0s

(534238, 255)
```

Data Partitioning

Our dataset is highly imbalanced which is evident in the image below:

```
readmission_flag
0    0.793671
1    0.206329
Name: proportion
```

To balance the dataset, we have used the Oversampling Strategy where data is split in such a way that, in training data, there is an equal proportion of samples belonging to both the classes (50% class 0 and 50% class 1) and the testing split retains the same proportion as the original dataset before splitting(79.4% class 0 and 20.6% class 1). Based on this strategy we have trained all the models mentioned in the next section.

Data Mining Models/Methods

For our dataset, we have used 6 different models:

- Logistic Regression
- Random Forest
- CatBoost
- LightGBM
- XGBoost

1. Logistic Regression

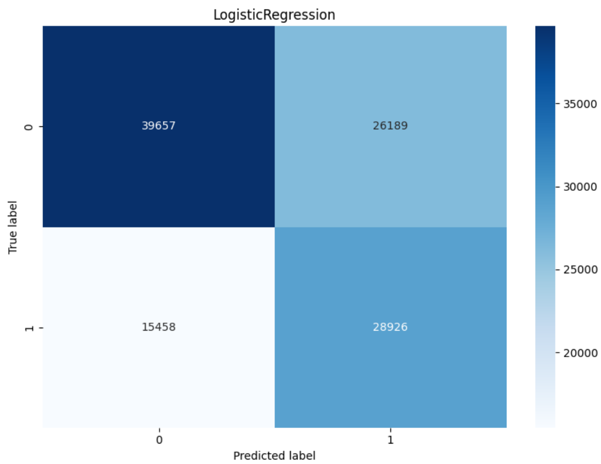
A simple, yet effective linear model which we have used for our binary classification task. It models the probability that given an input belongs to a certain class using the sigmoid function. It works best when the relationship between the features and the log-odds of the outcome is linear in nature. Despite being a basic algorithm, it is widely used in medical diagnosis applications, credit scoring, etc.

Advantages:

- Simple and easy to implement
- Fast training and predictions, even with large datasets
- Provides clear interpretation of feature influence through coefficients

Disadvantages:

- Assumes a linear relationship, which may not hold for complex data
- Struggles with non-linear relationships and interactions between features
- Sensitive to outliers and multicollinearity



2. Random Forests

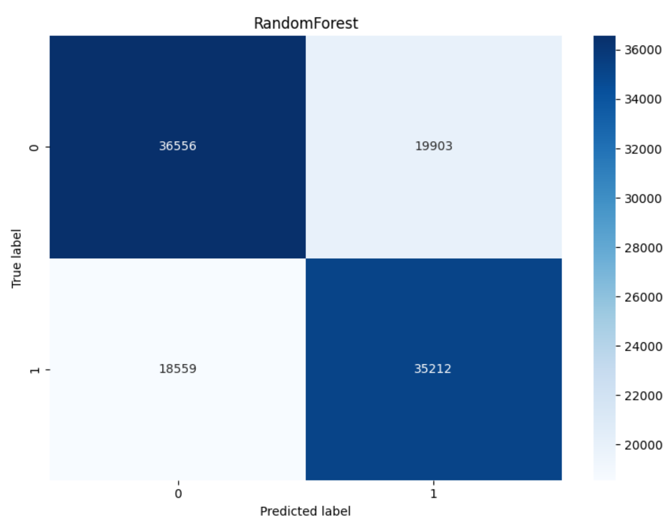
Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of their predictions (classification) or mean prediction (regression). It reduces overfitting by averaging multiple trees and introduces randomness through bootstrapped datasets and feature selection.

Advantages:

- Robust to overfitting by averaging multiple decision trees.
- Handles both numerical and categorical data well.
- Provides feature importance for better interpretability.

Disadvantages:

- Slower predictions due to ensemble of trees.
- High memory and computational cost, especially with large datasets.
- Less interpretable than simpler models like logistic regression.



3. CatBoost

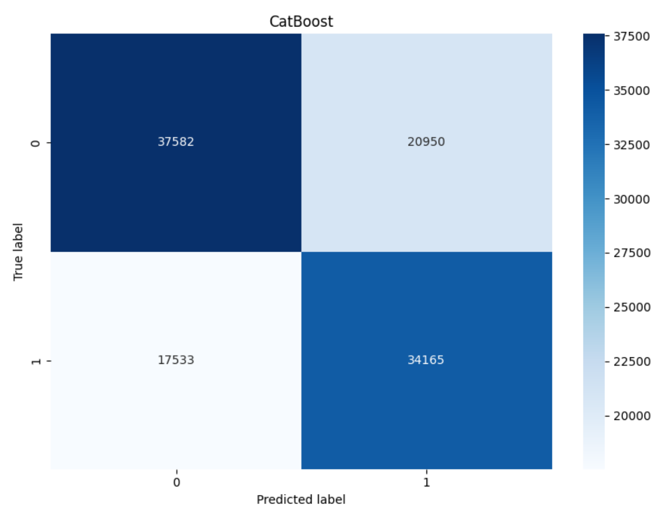
CatBoost (Categorical Boosting) is a gradient boosting algorithm that is designed especially for fast handling of categorical features. Unlike other boosters, it does not handle categorical variables through heavy preprocessing like one-hot encoding. It employs ordered boosting to reduce target leakage and overfitting, thereby making the model stable for any dataset. CPU and GPU are both optimized for the algorithm so that training and inference occur rapidly.

Advantages:

- It handles categorical data without extensive preprocessing
- It avoids overfitting with ordered boosting and efficient regularization
- Optimized for both CPU and GPU, enabling fast training and prediction

Disadvantages:

- Slower than LightGBM on very large datasets
- Requires careful parameter tuning to achieve the best performance
- Consumes more memory compared to simpler tree-based models



4. LightGBM

LightGBM is an efficient and fast gradient boosting library. It uses a different leaf-wise growth policy instead of the level-wise policy used by other algorithms, so that it can converge faster than other algorithms. The algorithm is fast even with large datasets, and it natively supports categorical features. The histogram-based approach allows for low memory consumption without sacrificing accuracy.

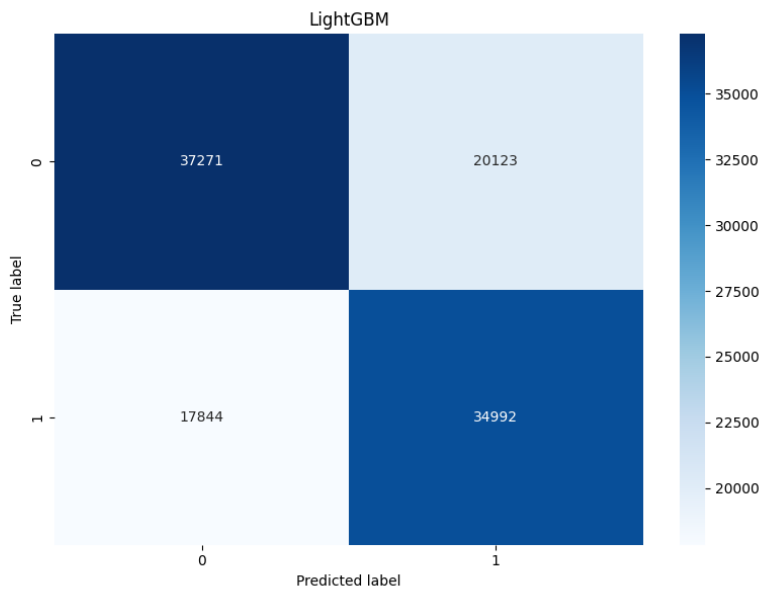
Advantages:

- Faster training and lower memory usage compared to XGBoost.

- Fast processing of large datasets efficiently with support for categorical features.
- Achieves high accuracy while being computationally efficient.

Disadvantages:

- Can overfit on small datasets due to its aggressive leaf-wise splitting.
- Less interpretable than other simpler tree-based models.
- It Struggles with very sparse datasets without proper preprocessing.



5. XGBoost

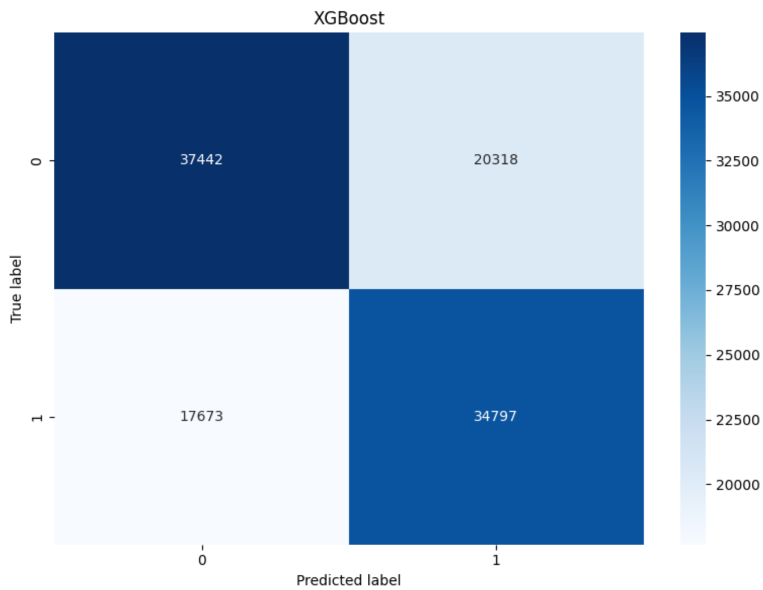
XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm based on decision trees optimized for speed and performance. XGBoost makes use of gradient boosting to refine weak models recursively by minimizing mistakes. Overfitting is averted using features like L1 and L2 regularization. Missing values and large datasets are manageable using parallel processing.

Advantages:

- It is highly efficient and scalable due to parallel processing and optimized memory usage.
- It prevents overfitting with built-in regularization techniques like L1 and L2.
- Handles missing values automatically and supports custom objective functions.

Disadvantages:

- Computationally expensive for very large datasets with deep trees.
- It requires careful hyperparameter tuning to achieve optimal performance.
- It is less interpretable than simpler models like logistic regression or decision trees.



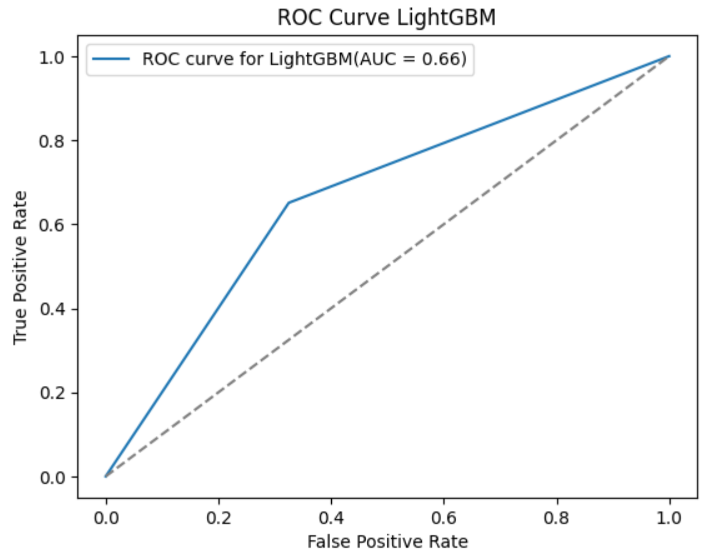
Performance Evaluation

Model	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
XGBoost	**62.97%**	0.626	0.643	0.634	0.633	0.617	0.625
LightGBM	**63.16%**	0.628	0.645	0.637	0.635	0.618	0.626
CatBoost	**62.93%**	0.626	0.643	0.634	0.633	0.616	0.624
Logistic Regression	**62.83%**	0.626	0.639	0.632	0.631	0.617	0.624
RandomForests	**62.20%**	0.614	0.656	0.635	0.631	0.588	0.609

The best model over here is the **LightGBM** model as per the accuracy, f1-score, precision and recall of majority class (class 1).

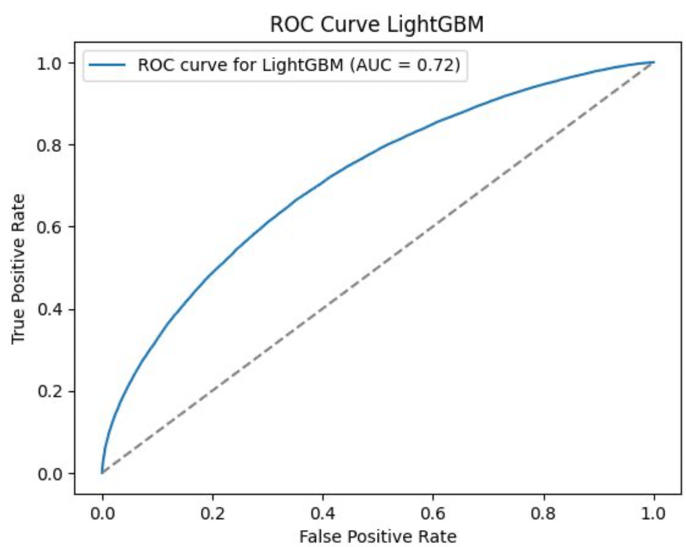
Before Hyperparameter Tuning

For further performance analysis, we have plotted the ROC Curve for the model and shown below:



After Hyperparameter Tuning

After performing further hyperparmeter tuning, the improved ROC-AUC curve is shown below:



The performance of the LightGBM model after hyperparameter tuning is as below:

Model:	precision:	recall:	f1-score:	support:	accuracy:
LightGBM	0.834393	0.670133	0.722684	424010.0	0.670133

<u>Project Results</u>

After seeing the above results, the best classification model for the dataset and the task would be the LightGBM model. The reason why we have chosen this model is because it has high F1-

score, better AUC scores and good precision. Although our model has not achieved higher accuracy and better scores, this approach would still be reasonable given the scale of the dataset, the number of features and the high imbalance in the dataset.

Impact of the project Outcomes

The impact of this patient readmission prediction project can be significant in improving healthcare outcomes. By accurately identifying patients at risk of being readmitted within 30 days, healthcare providers can intervene early to prevent unnecessary readmissions, ultimately improving patient outcomes and reducing hospital costs. This model can also help optimize hospital resource allocation, ensuring that care teams prioritize high-risk patients. Additionally, understanding the key factors contributing to readmission can guide healthcare policy decisions and enhance patient care strategies. The application of machine learning in this domain demonstrates its potential to drive data-driven decisions, transforming healthcare delivery with predictive analytics. Furthermore, the ability to personalize discharge planning and follow-up care based on risk factors can improve overall hospital efficiency and patient satisfaction.

References:

Davis, S., Zhang, J., Lee, I., Rezaei, M., Greiner, R., McAlister, F. A., & Padwal, R. (2022, November 24). *Effective hospital readmission prediction models using machine-learned features - BMC Health Services Research*. BioMed Central. https://doi.org/10.1186/s12913-022-08748-y