# Predicting Emergency Visits using Machine Learning

**Homework Topic**: Self Learning

**ML Algorithms**: Logistic Regression, XGBoost Classifier

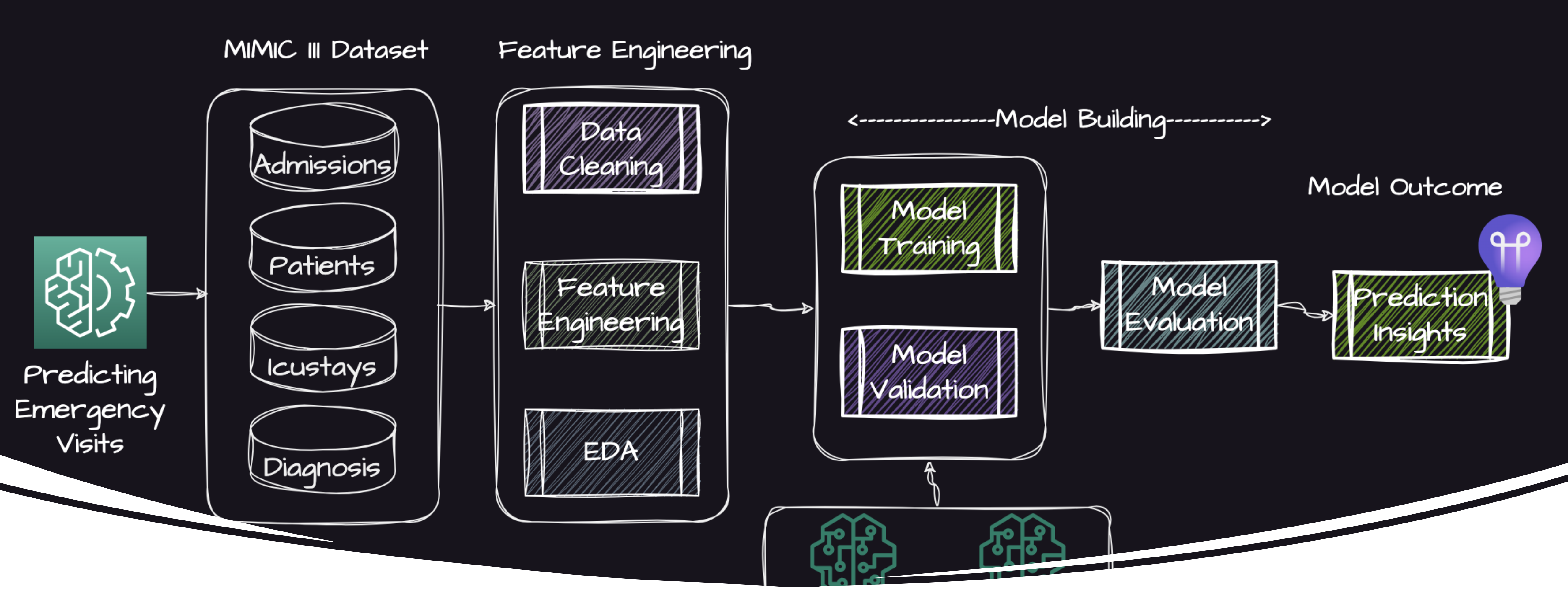
**Prepared By**: Suresh Venkatesan

**Course**: AI in Healthcare(AI 395T)

**Professor**: Dr. Ying Ding

## Overview

- This presentation explores the prediction of emergency visits using the MIMIC-III dataset, a rich source of clinical data.
- Analyzed patient demographics, diagnoses, and hospital stay (admission and ICU stays) information to build predictive models for emergency visits.
- Using both Logistic Regression and XGBoost models and discovered that factors like age, marital status, length of stay, mortality rates, and the presence of detailed medical records significantly contribute to predicting emergency visits.



## Process Flow Diagram for Predicting Emergency Visits

- The process flow diagram below outlines the steps taken to predict emergency visits using machine learning models. The workflow is divided into several key stages, each crucial for building and evaluating the predictive models.

- The machine learning process involves preparing data, building a model, evaluating its performance, and gaining insights from the results. The selected model (e.g., Logistic Regression or XGBoost) is used to predict future emergency visits based on patient data.

# Load the MIMIC-III Datasets

**Dataset Loading**

Loading the necessary datasets from the MIMIC-III database. The datasets include Admissions, Patients, Diagnosis, and Icustays. These datasets contain crucial information about patient admissions, demographics, diagnoses, and ICU stays, which will be used for our analysis.

The following steps are performed during the dataset loading process:

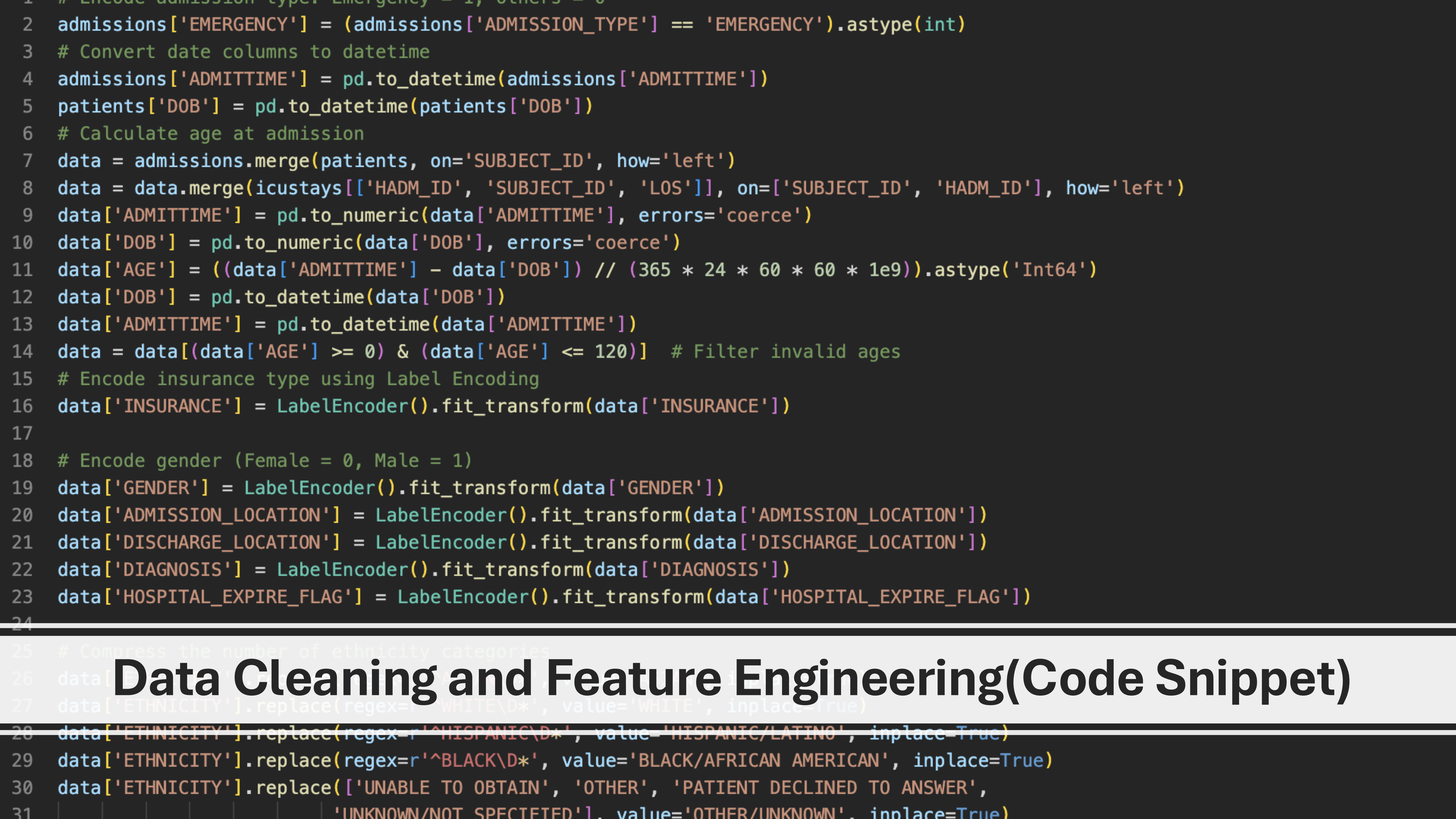
- **Loading Datasets**:
  - The datasets are loaded using the pandas library.
  - The paths to the datasets are specified, and the data is read into pandas DataFrames.
- **Initial Data Inspection**:
  - The first few rows of each dataset are displayed to understand the structure and contents.
  - This helps in identifying the key columns and any immediate data quality issues.
- **Merging Datasets**:
  - The datasets are merged based on common identifiers such as SUBJECT_ID and HADM_ID.
  - This creates a comprehensive dataset that combines information from multiple sources.
- **Handling Missing Values**:
  - Initial handling of missing values is performed to ensure the datasets are ready for further preprocessing and analysis.

## Data Cleaning and Feature Engineering

In this section, we will perform data cleaning and feature engineering to prepare the dataset for modeling. The following steps are undertaken:

- **Encoding Admission Type**:
  - Encode the ADMISSION_TYPE column to create a binary EMERGENCY column where 'EMERGENCY' is 1 and others are 0.
- **Date Conversion and Age Calculation**:
  - Convert date columns to datetime format.
  - Calculate the age of patients at the time of admission.
  - Filter out invalid ages (e.g., negative ages or ages greater than 120).
- **Encoding Categorical Variables**:
  - Encode categorical variables such as INSURANCE, GENDER, ADMISSION_LOCATION, DISCHARGE_LOCATION, DIAGNOSIS, and HOSPITAL_EXPIRE_FLAG using label encoding.
- **Compressing Ethnicity Categories**:
  - Group similar ethnicity categories to reduce the number of unique values.
- **Handling Missing Values**:
  - Fill missing values in the MARITAL_STATUS column with 'UNKNOWN (DEFAULT)'.
  - Replace rare categories in the RELIGION column with 'RELIGIOUS'.
- **Dropping Unused Columns**:
  - Drop columns that are not needed for the analysis.
- **Creating Dummy Variables**:
  - Create dummy variables for categorical features to make them suitable for machine learning models.

## Data Cleaning and Feature Engineering(Code Snippet)



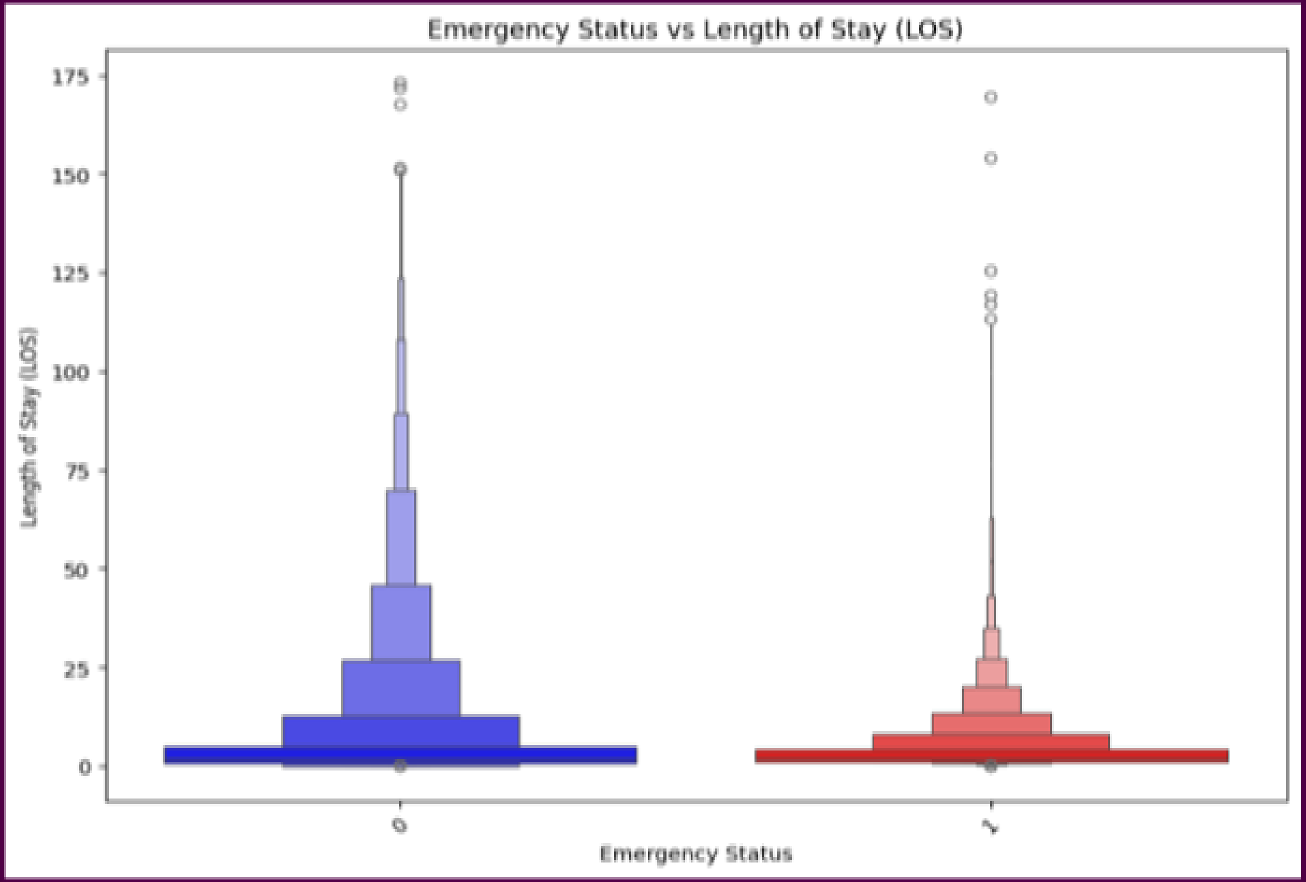
```python
# Encode admission type: Emergency = 1, Others = 0
admissions['EMERGENCY'] = (admissions['ADMISSION_TYPE'] == 'EMERGENCY').astype(int)
# Convert date columns to datetime
admissions['ADMITTIME'] = pd.to_datetime(admissions['ADMITTIME'])
patients['DOB'] = pd.to_datetime(patients['DOB'])
# Calculate age at admission
data = admissions.merge(patients, on='SUBJECT_ID', how='left')
data = data.merge(icustays[['HADM_ID', 'SUBJECT_ID', 'LOS']], on=['SUBJECT_ID', 'HADM_ID'], how='left')
data['ADMITTIME'] = pd.to_numeric(data['ADMITTIME'], errors='coerce')
data['DOB'] = pd.to_numeric(data['DOB'], errors='coerce')
data['AGE'] = ((data['ADMITTIME'] - data['DOB']) // (365 * 24 * 60 * 60 * 1e9)).astype('Int64')
data['DOB'] = pd.to_datetime(data['DOB'])
data['ADMITTIME'] = pd.to_datetime(data['ADMITTIME'])
data = data[(data['AGE'] >= 0) & (data['AGE'] <= 120)]  # Filter invalid ages
# Encode insurance type using Label Encoding
data['INSURANCE'] = LabelEncoder().fit_transform(data['INSURANCE'])

# Encode gender (Female = 0, Male = 1)
data['GENDER'] = LabelEncoder().fit_transform(data['GENDER'])
data['ADMISSION_LOCATION'] = LabelEncoder().fit_transform(data['ADMISSION_LOCATION'])
data['DISCHARGE_LOCATION'] = LabelEncoder().fit_transform(data['DISCHARGE_LOCATION'])
data['DIAGNOSIS'] = LabelEncoder().fit_transform(data['DIAGNOSIS'])
data['HOSPITAL_EXPIRE_FLAG'] = LabelEncoder().fit_transform(data['HOSPITAL_EXPIRE_FLAG'])

# Compress the number of ethnicity categories
data[[illegible]
data['ETHNICITY'].replace(regex=r'^WHITE\D*', value='WHITE', inplace=True)
data['ETHNICITY'].replace(regex=r'^HISPANIC\D*', value='HISPANIC/LATINO', inplace=True)
data['ETHNICITY'].replace(regex=r'^BLACK\D*', value='BLACK/AFRICAN AMERICAN', inplace=True)
data['ETHNICITY'].replace(['UNABLE TO OBTAIN', 'OTHER', 'PATIENT DECLINED TO ANSWER',
                          'UNKNOWN/NOT SPECIFIED'], value='OTHER/UNKNOWN', inplace=True)
```

## Exploratory Data Analysis (EDA)

In this section, we perform exploratory data analysis (EDA) to understand the relationships between various features and the target variable, EMERGENCY. The following visualizations will be used to perform EDA on MIMIC III datasets.

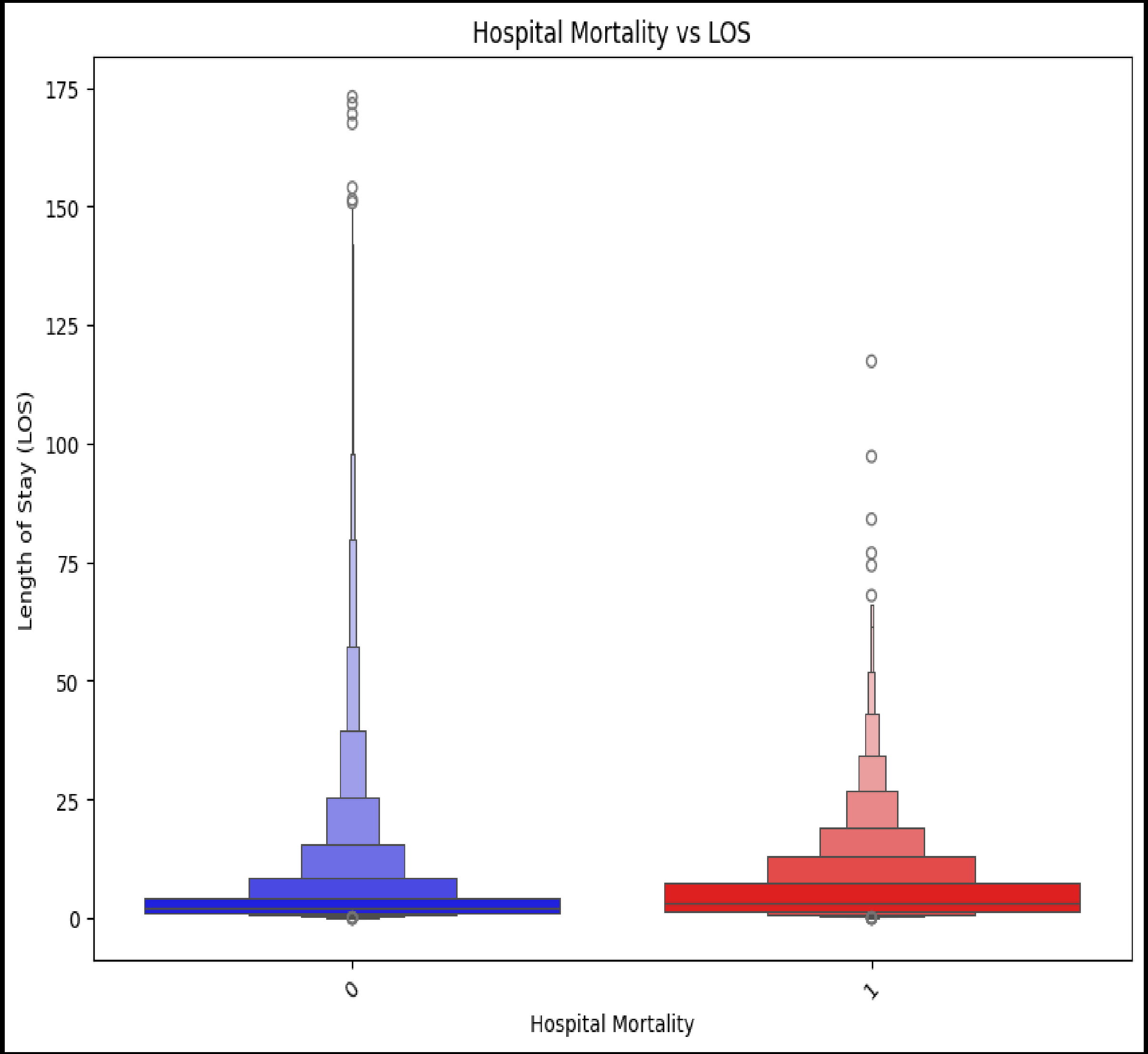
## Emergency Visits vs Length of Stay (LOS)

This boxplot compares the number of emergency visits, non-emergency visits vs LOS. It helps to visualize the distribution of the target variable, EMERGENCY.



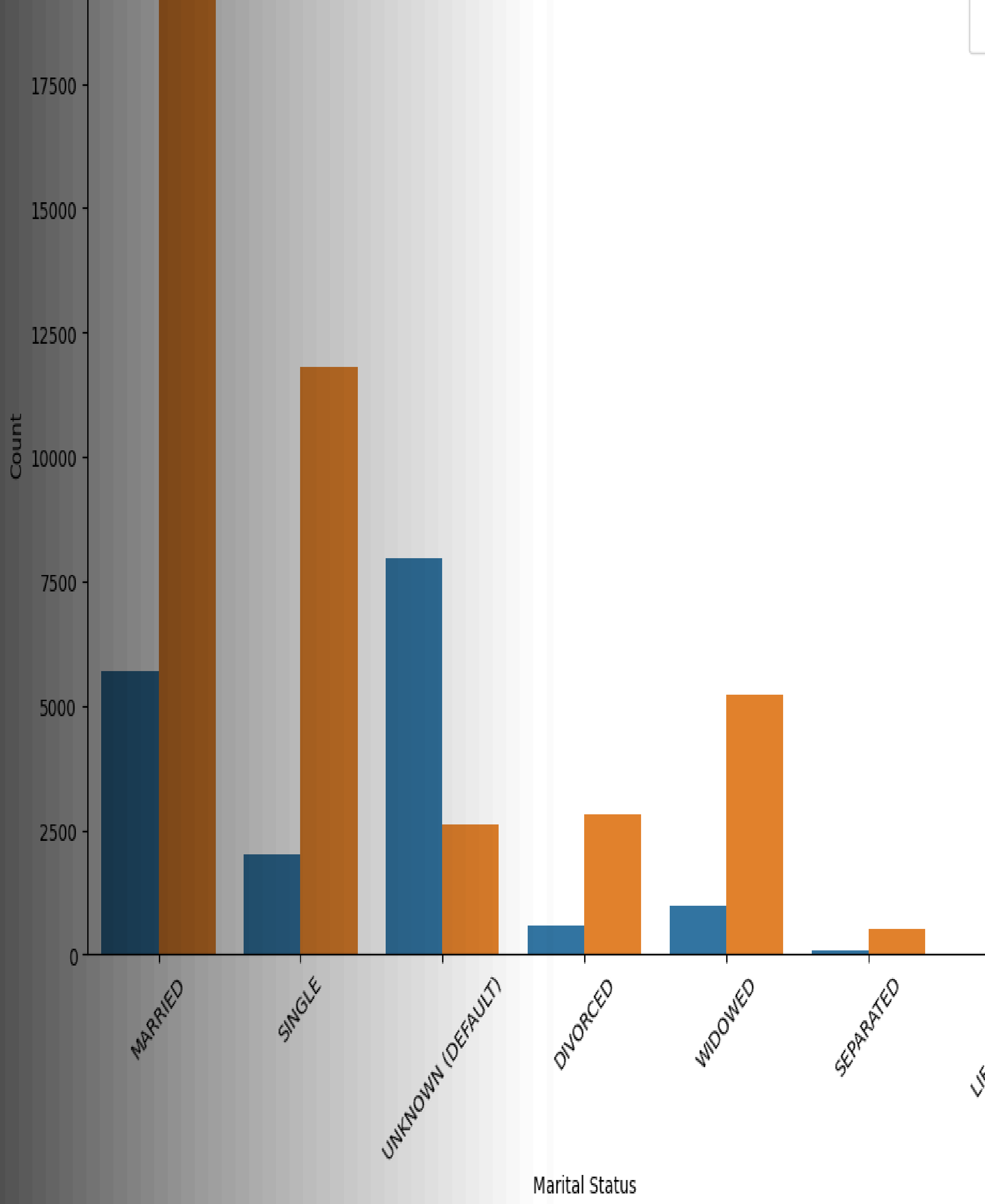
## Hospital Mortality vs Length of Stay (LOS)

Understanding the distribution of hospital mortality versus LOS is crucial for developing robust predictive models and ensuring accurate evaluation metrics.



## Marital Status and Emergency

The bar plot comparing emergency visits to marital status provides valuable insights into the distribution of the target variable, EMERGENCY. It appears that married individuals tend to visit the emergency room more frequently than others.

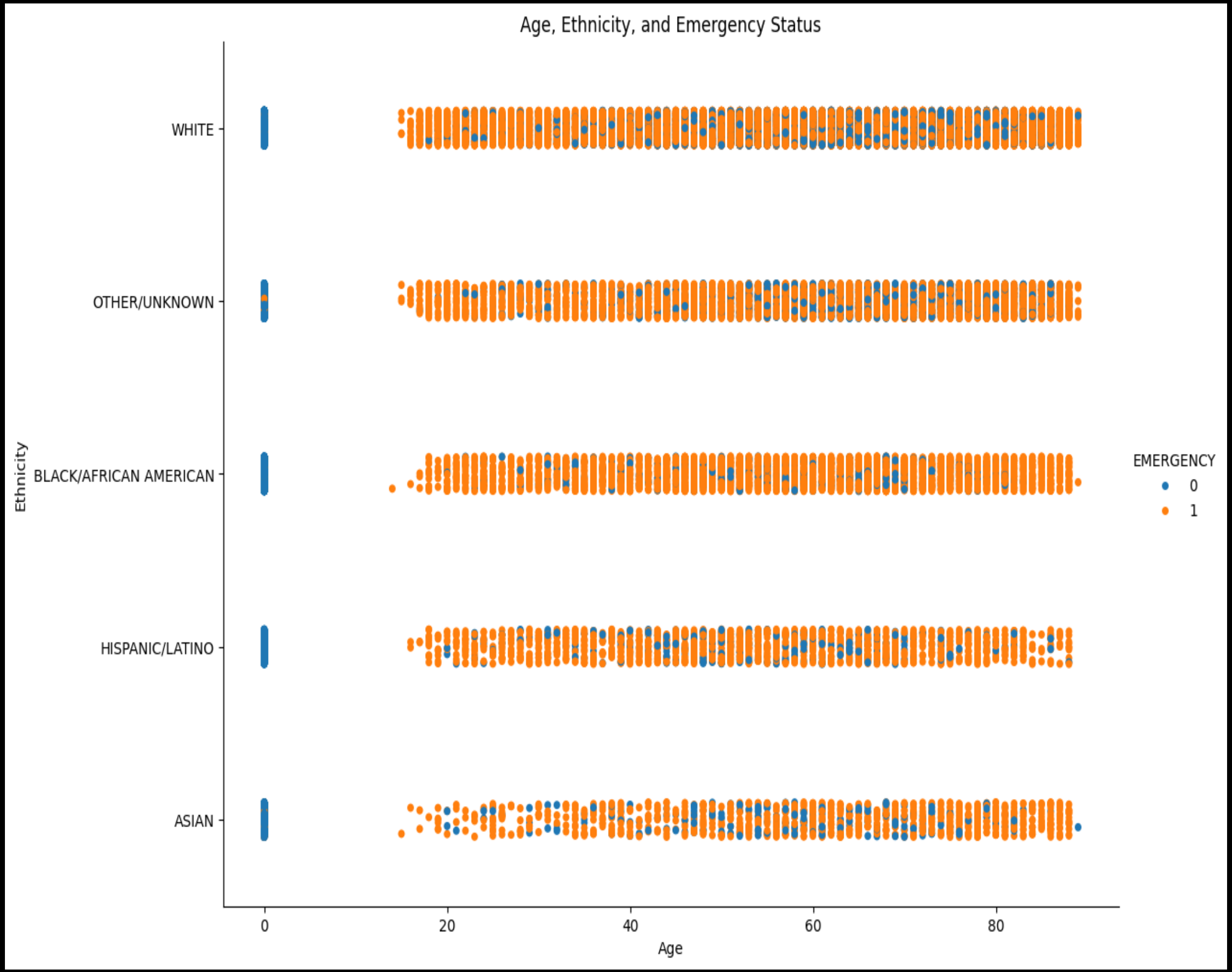


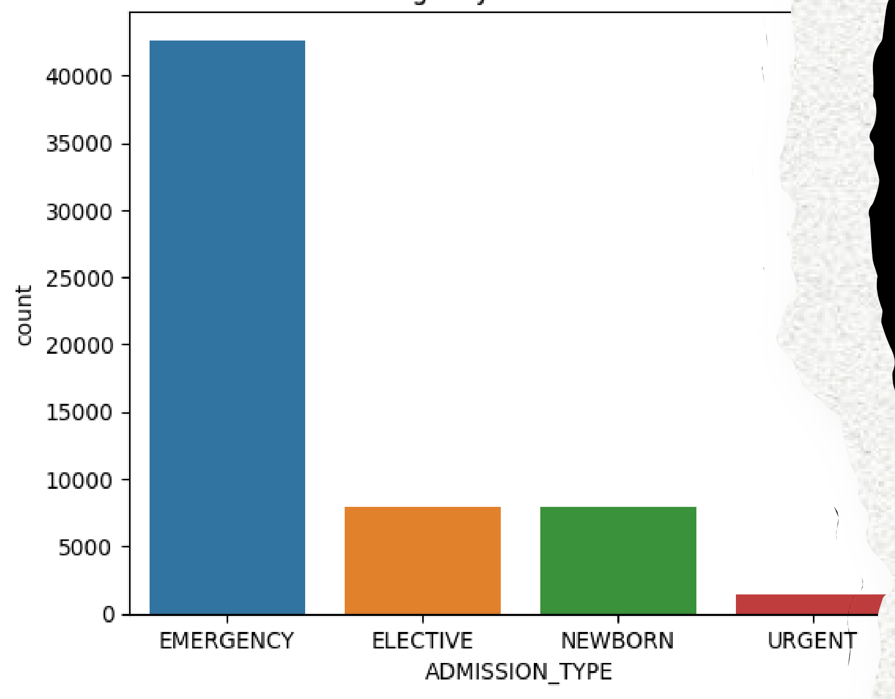
# Age, Ethnicity, and Emergency Status

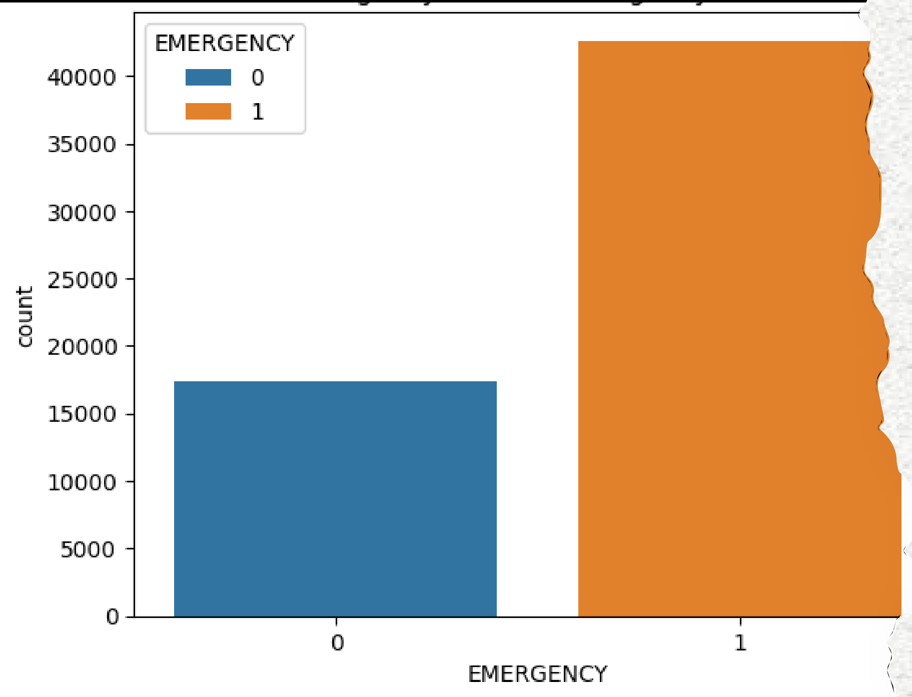
This scatter plot compares the distribution of emergency visits and non-emergency visits with age and ethnicity to provide more insights. The plot highlights the frequency of emergency visits in comparison to non-emergency visits, which is crucial for understanding the overall distribution of the target variable, EMERGENCY.

Key observations from the plot include:

- The number of emergency visits is significantly higher than non-emergency visits.
- This distribution indicates a higher prevalence of emergency cases in the dataset, which may impact the performance and evaluation of the classification models.





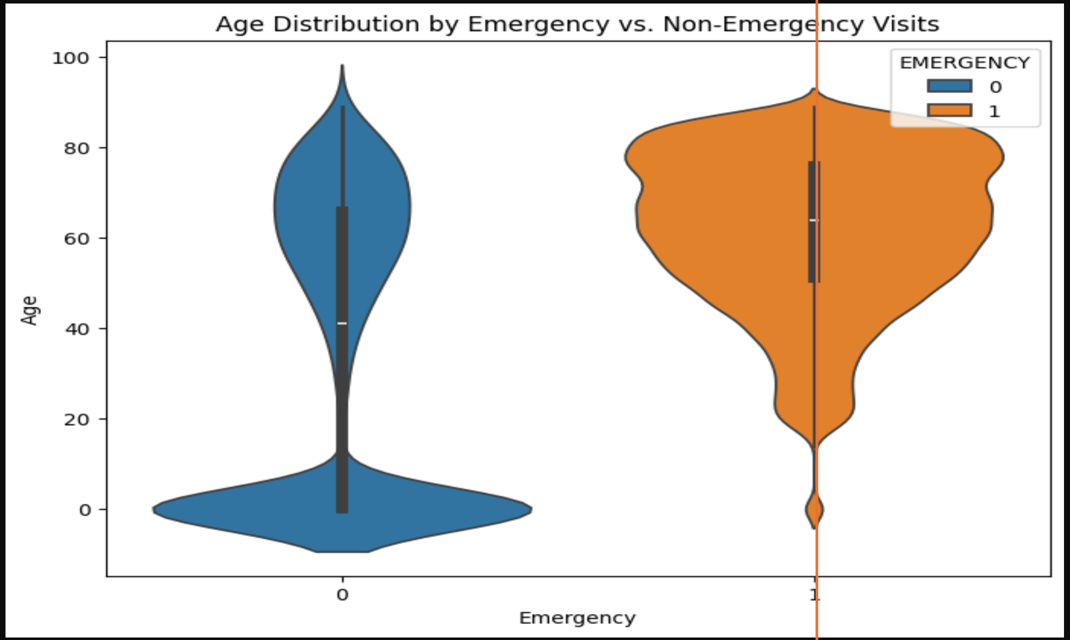


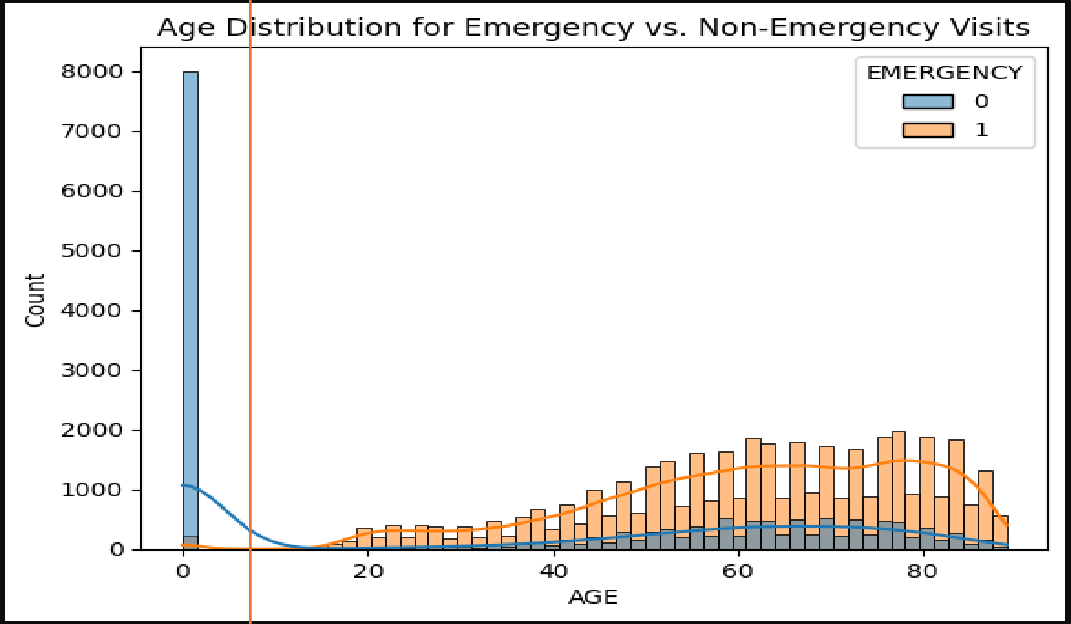
# Emergency vs. Other Visits

- This plot on the top compares the distribution of emergency visits versus other types of visits.
- This plot on the bottom compares the distribution of emergency visits versus non-emergency visits to provide more insights into the dataset.

## Age Distribution for Emergency vs. Non-Emergency Visits

These plots are used to visualize the age distribution for emergency and non-emergency visits. This plot provides an easy understanding of the relationship between age and emergency visits, showing that older individuals tend to visit the emergency department more frequently.

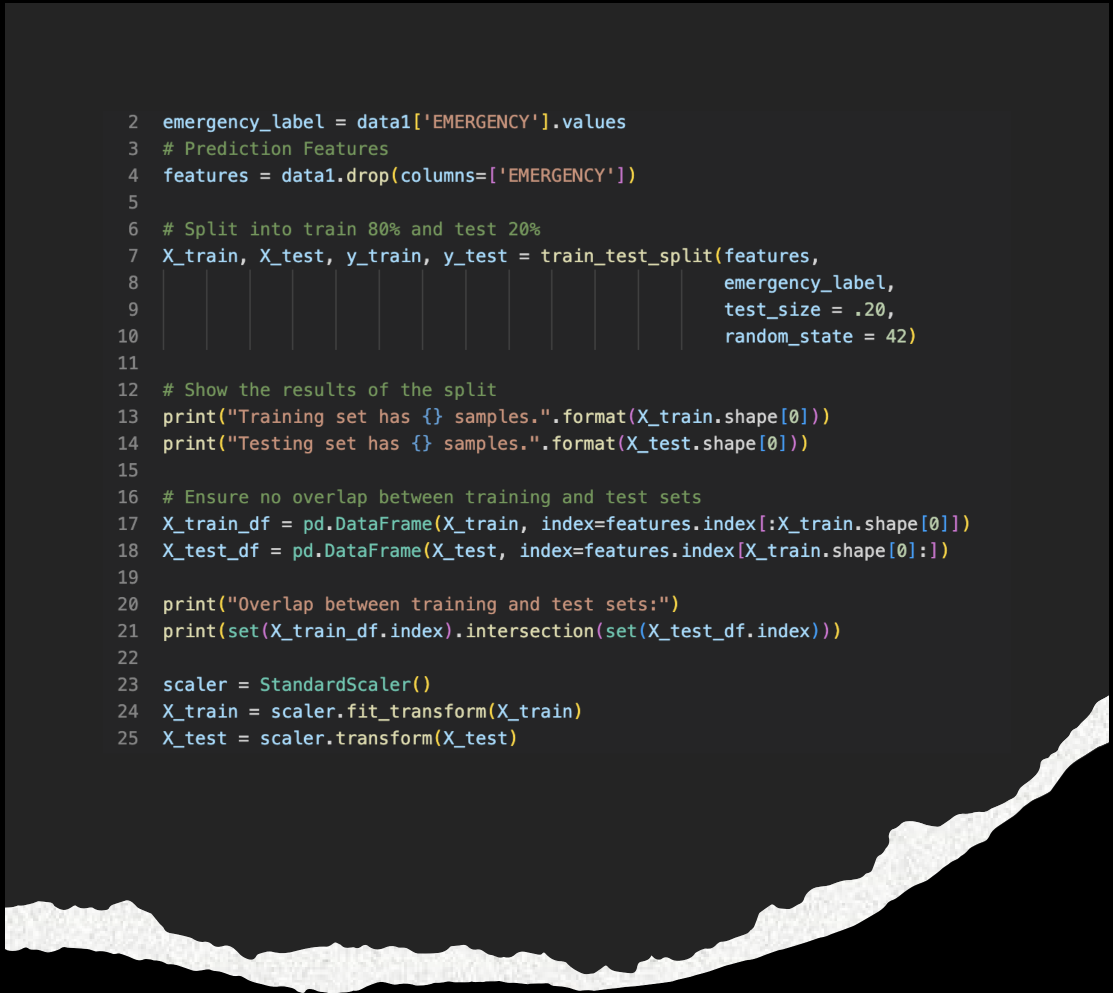




# Data Splitting for Model Training and Validation

Split the dataset into training and testing sets to build and validate our machine learning models. The following steps will be performed:

- **Target Variable**: The target variable for prediction is `EMERGENCY`, which indicates whether a visit is an emergency or not.
- **Feature Selection**: The features used for prediction are all columns except the target variable `EMERGENCY`.
- **Train-Test Split**: The dataset is split into training (80%) and testing (20%) sets to evaluate the model's performance on unseen data.
- **Standardization**: The features are standardized to ensure they have a mean of 0 and a standard deviation of 1, which helps in improving the performance of certain machine learning algorithms.



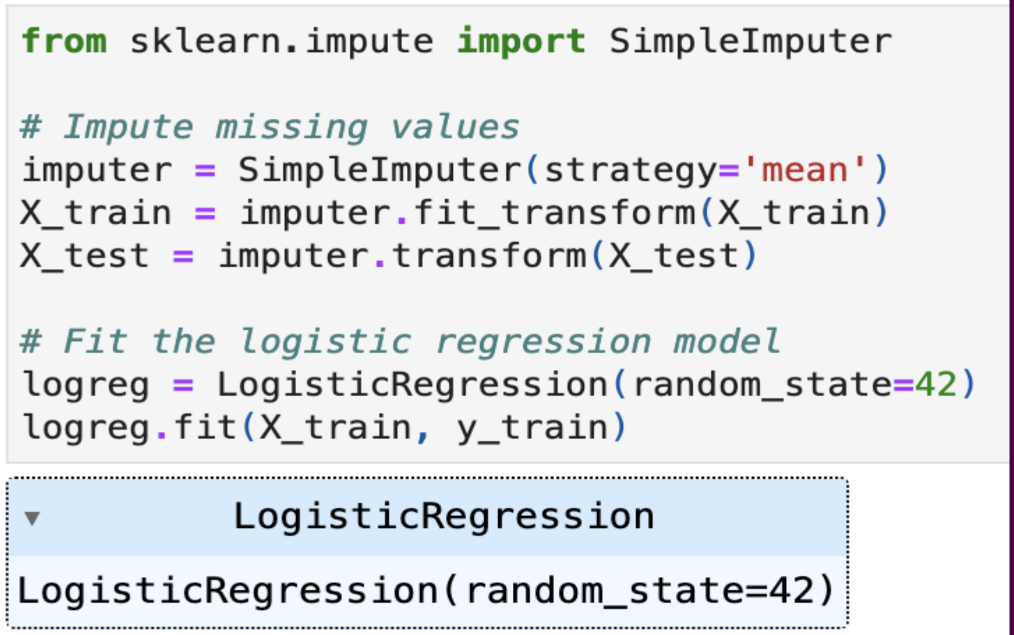
```python
emergency_label = data1['EMERGENCY'].values
# Prediction Features
features = data1.drop(columns=['EMERGENCY'])

# Split into train 80% and test 20%
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    emergency_label,
                                                    test_size = .20,
                                                    random_state = 42)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))

# Ensure no overlap between training and test sets
X_train_df = pd.DataFrame(X_train, index=features.index[:X_train.shape[0]])
X_test_df = pd.DataFrame(X_test, index=features.index[X_train.shape[0]:])

print("Overlap between training and test sets:")
print(set(X_train_df.index).intersection(set(X_test_df.index)))

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

## Building and Training the Models

In this section, we will build and train the Logistic Regression and XGBoost models using the preprocessed dataset. The steps include:

- Imputing Missing Values: Handle any missing values in the dataset using the `SimpleImputer`.
- Training Logistic Regression Model: Fit the Logistic Regression model on the training data.
- Training XGBoost Model: Fit the XGBoost model on the training data.

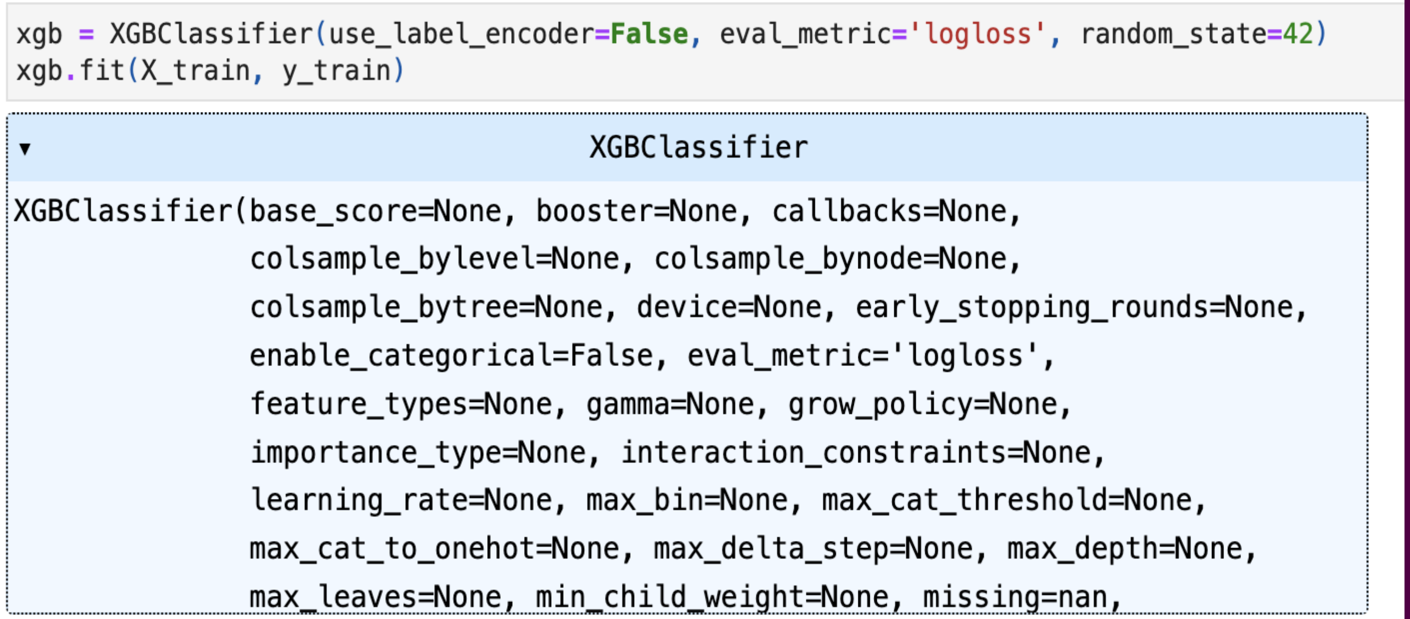
### Logistic Regression



```
from sklearn.impute import SimpleImputer

# Impute missing values
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Fit the logistic regression model
logreg = LogisticRegression(random_state=42)
logreg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(random_state=42)
```

### XGBoost Classifier



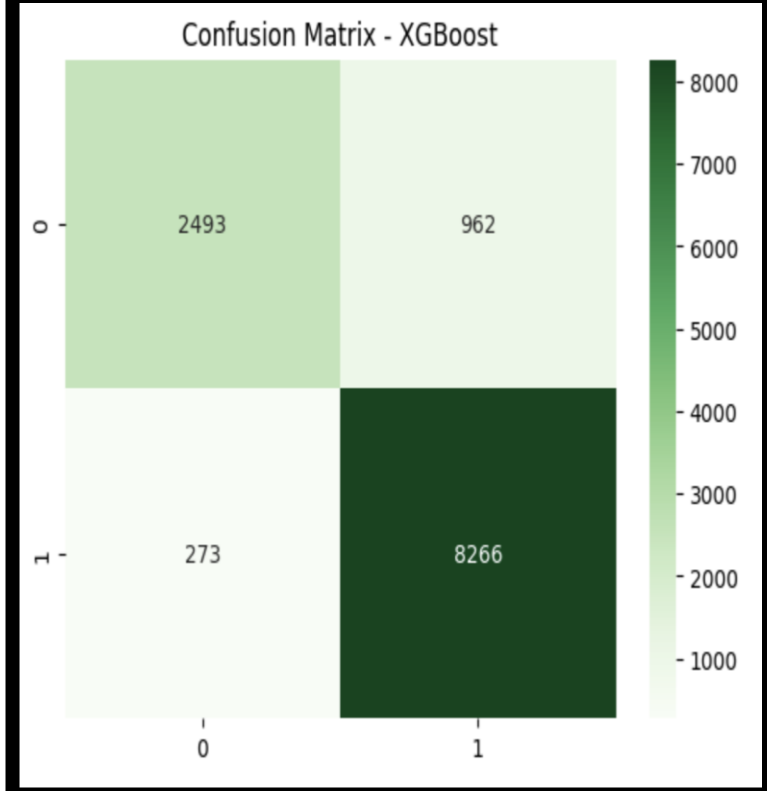
```
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb.fit(X_train, y_train)
```

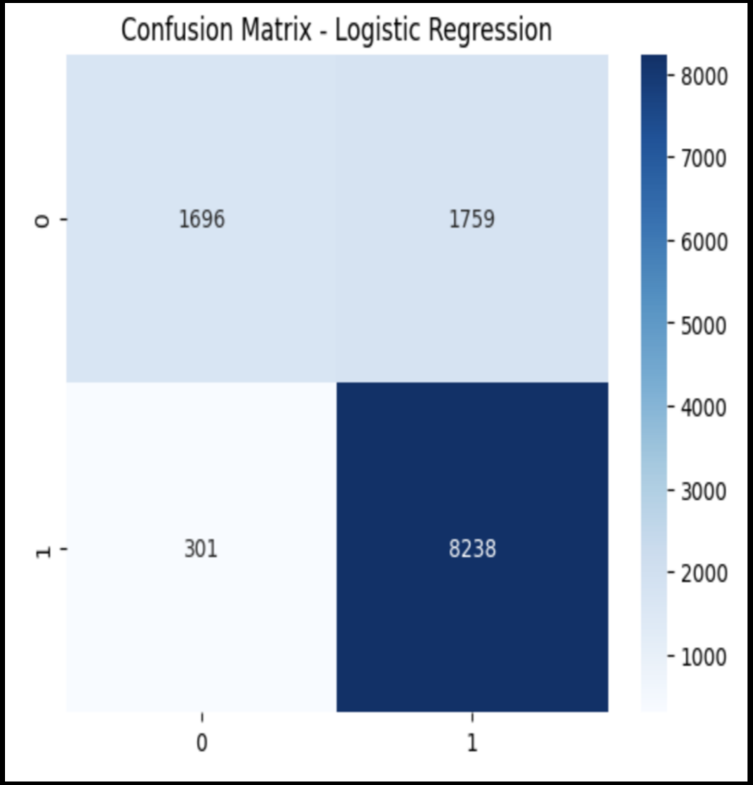
```
▾ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
```

# Model Validation and Evaluation

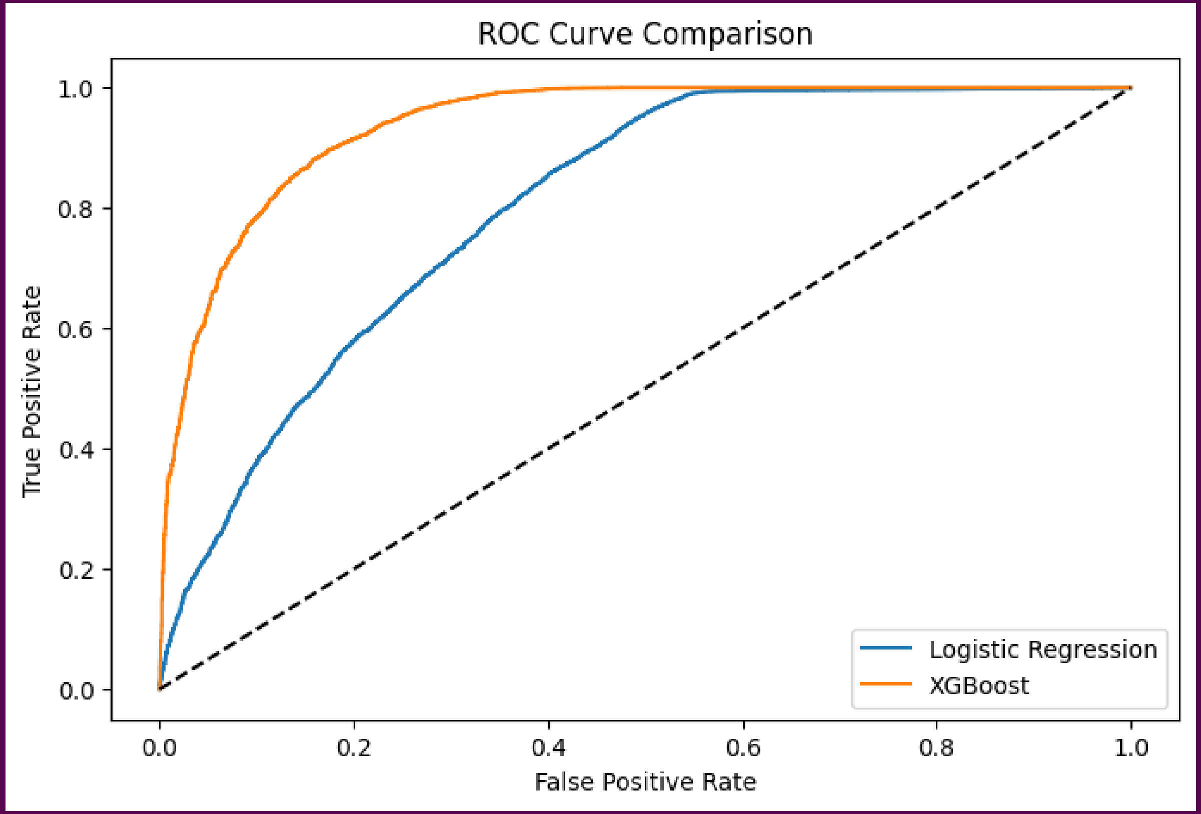
In this section, we will validate and evaluate the performance of the Logistic Regression and XGBoost models. The following steps will be performed:

- ✓ **Evaluate Logistic Regression(left)**
- ✓ **Evaluate XGBoost Classifier(Right)**





## ROC-AUC Comparison for Logistic Regression and XGBoost Classifier Models



In this section, we compare the ROC-AUC scores for the Logistic Regression and XGBoost models. The ROC-AUC score is a performance measurement for classification problems at various threshold settings. It tells how much the model is capable of distinguishing between classes.

- **Logistic Regression ROC-AUC**: 0.80
- **XGBoost ROC-AUC**: 0.94

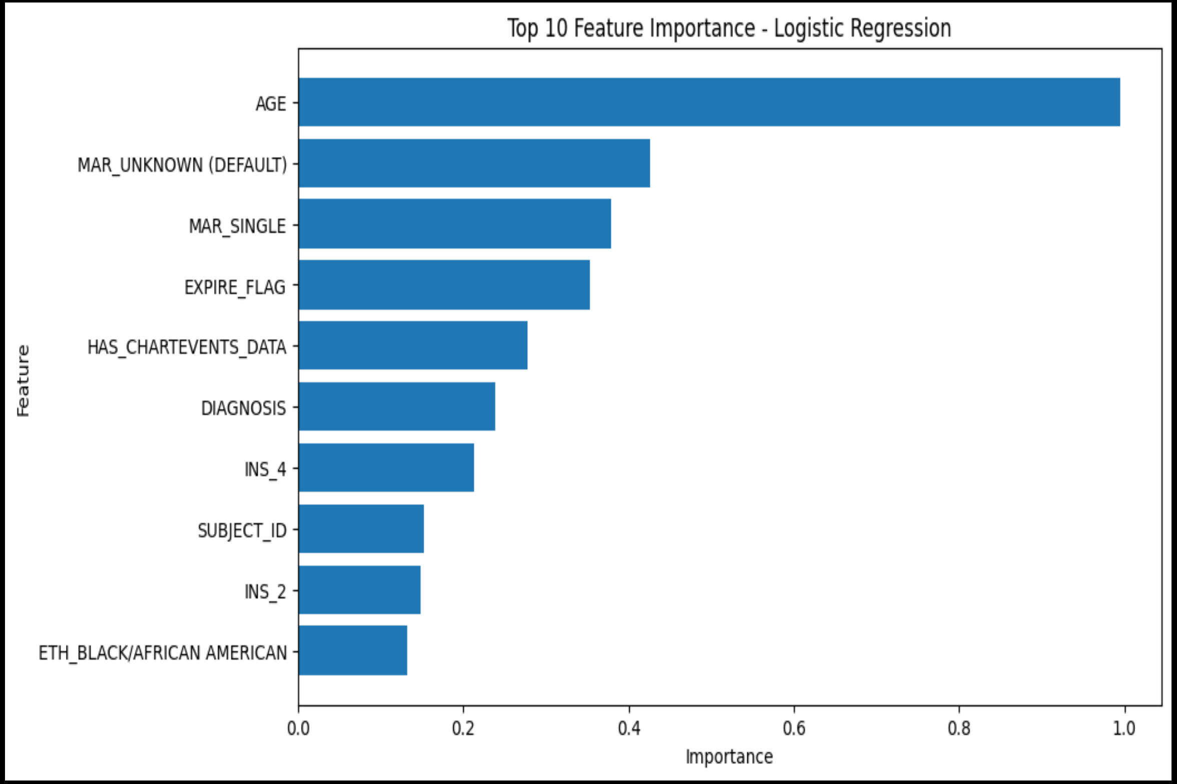
The XGBoost model outperforms the Logistic Regression model, indicating its superior ability to distinguish between emergency and non-emergency visits.

# Feature Importance

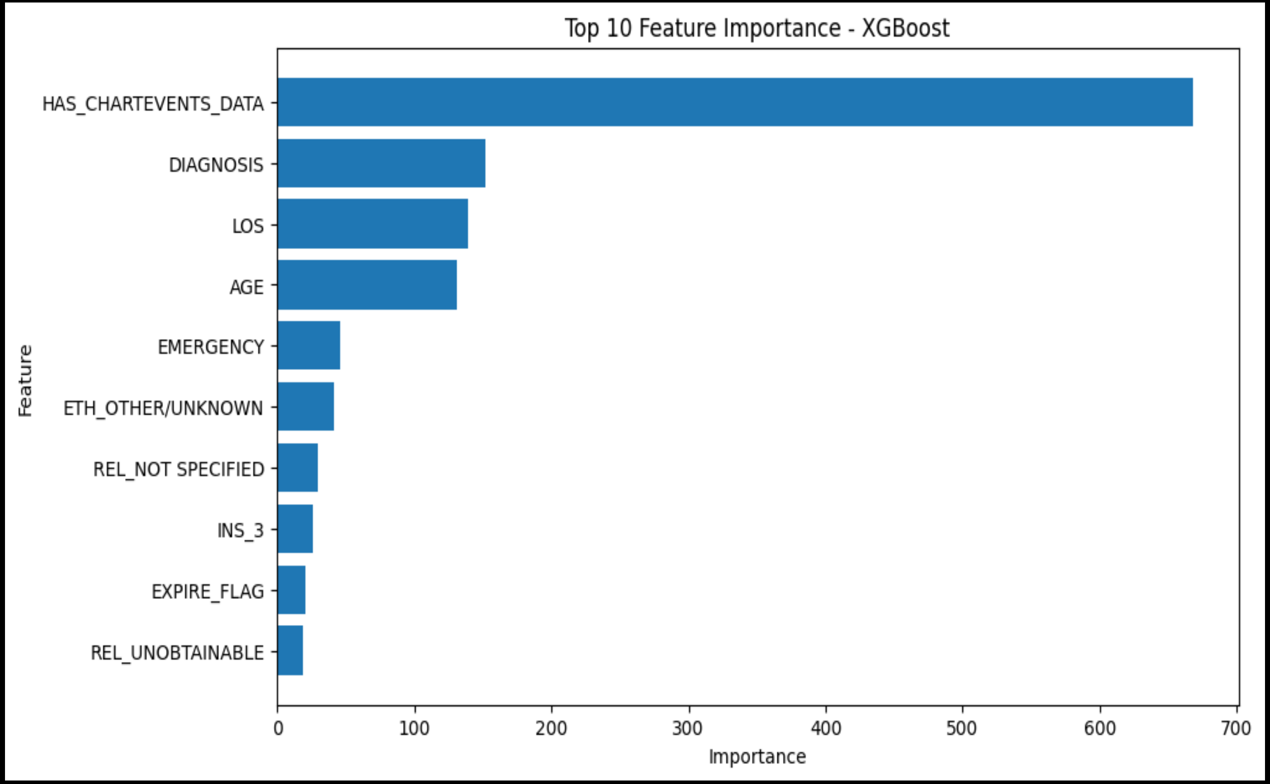
In this section, we will analyze the feature importance for both the Logistic Regression and XGBoost models. Understanding which features contribute the most to the predictions can provide valuable insights into the factors influencing emergency visits.

By comparing the feature importance from both models, we can gain a comprehensive understanding of the key factors driving emergency visits.

## Feature Importance for Logistic Regression



## Feature Importance for XGBoost Classifier



## Conclusion and Insights

- **Age and Emergency Visits**: Age is a significant predictor of emergency visits, with older patients more likely to have emergency admissions.
- **Length of Stay (LOS)**: Emergency visits are associated with longer hospital stays, indicating the severity and complexity of cases.
- **Insurance and Marital Status**: Certain insurance types and marital statuses are more prevalent among emergency visits, suggesting socio-economic factors play a role.
- **Model Selection**: The XGBoost model is more effective for predicting emergency visits, highlighting the importance of using advanced machine learning techniques for complex classification tasks.
- **Additional Features**: Exploring additional features and external data sources could enhance the model's accuracy.
- **Real-time Prediction**: Implementing the model in a real-time clinical setting could provide timely insights and improve patient care.

The analysis successfully identified key features and built predictive models for emergency visits, with the XGBoost model showing superior performance. These insights can help healthcare providers better understand and manage emergency admissions.

# Thank You