**Healthcare predictive Analysis project**

Capstone project for the second cohort of the Digital Egypt Initiative, under the auspices of the Ministry of Communications and Information Technology

**Data Science Track**

**Project Team Member:**

* Mai Mosaad Ragab
* Somia Ashraf Soliman elsaed
* Nada Ragab Ibrahim
* Maha Mosbeh Elkohaley
* Aisha Tarek abdelhamid
* gehad abdelal

***Under Supervision of***

**Dr. Eslam Elreedy**

Acknowledgment

As we reflect on the completion of this project, it is important to recognize the collective efforts that have made this dataset and its analysis possible. The journey of cleaning, analyzing, and preparing the Diabetic Data Cleaning dataset for predictive modeling has been both challenging and rewarding. It is through the dedication and support of various individuals and organizations that this work has reached its current stage.

First and foremost, we extend our deepest gratitude to the original creators of the dataset, whose efforts in compiling and sharing this valuable resource have enabled researchers and data scientists to explore critical healthcare challenges, such as predicting hospital readmissions for diabetic patients. The dataset, sourced from Kaggle (Diabetic Data Cleaning Dataset), is a testament to the importance of open data in advancing medical research and improving patient outcomes.

We would also like to acknowledge the contributions of the healthcare institutions and professionals who originally collected and anonymized the patient data. Their commitment to ethical data practices and patient privacy has ensured that this dataset can be used responsibly for research purposes.

Special thanks go to the data science community, whose collaborative spirit and shared knowledge have been instrumental in refining the techniques and methodologies applied in this project. The guidance and insights from peers, mentors, and online resources have been invaluable in overcoming the technical challenges encountered during the data cleaning and analysis process.

In particular, we are grateful to Dr. Eslam Elreedy, whose expertise and mentorship have been a guiding light throughout this project. His dedication to fostering a deeper understanding of data science principles and his unwavering support have not only contributed to the success of this work but have also inspired growth and learning beyond the scope of this dataset.

Additionally, we appreciate the support of our colleagues and fellow researchers, who have provided constructive feedback and encouragement. Their contributions, whether through direct collaboration or shared resources, have enriched this project and broadened its potential impact.

Finally, we are thankful for the opportunity to work with this dataset and contribute to the ongoing efforts to improve healthcare outcomes through data-driven insights. The knowledge and experience gained from this project will undoubtedly serve as a foundation for future endeavors in the field of medical data analysis.

As we conclude this phase of the project, we do so with a sense of pride and accomplishment, knowing that the work done here has the potential to make a meaningful difference. We look forward to the next steps, including the development of predictive models and further exploration of the data, with the hope of contributing to advancements in patient care and hospital management.

Congratulations to all who have been part of this journey—we did it!

**DECLARATION**

We hereby certify that this material, which we now submit for assessment on the program of the Data Science Track, is entirely our own work. We have exercised reasonable care to ensure that the work is original and, to the best of our knowledge, does not breach any law of copyright. Any work or ideas taken from external sources have been properly cited and acknowledged within the text of our work. This declaration applies to all aspects of the project, including the cleaning, analysis, and documentation of the Diabetic Data Cleaning dataset.

**Signed:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Date:** April 29, 2025

**ABSTRACT**

The Diabetic Data Cleaning dataset, accessible via Kaggle and sourced from the UCI Machine Learning Repository, comprises a rigorously curated and preprocessed collection of diabetic patient records, optimized for advanced healthcare analytics and machine learning applications. Preprocessing entailed meticulous imputation of missing values, encoding of categorical variables, and validation of data consistency to ensure analytical integrity. The dataset includes critical patient attributes—demographics, medical histories, and readmission outcomes—rendering it an essential tool for predictive modeling of patient outcomes, exploration of health determinants, and statistical assessment of diabetic care practices. This resource empowers data-driven advancements in patient care and healthcare system efficiency.

The dataset's utility is further underscored by its comprehensive feature set, which includes patient demographics (age, gender, race), medical history (diagnosis codes, number of medications), and hospital admission details (admission type, time in hospital, number of lab procedures). The primary outcome variable, readmission status, is pivotal for developing models that predict patient readmissions, thereby enabling healthcare providers to implement targeted interventions and reduce readmission rates. Additional attributes, such as the number of emergency visits and outpatient encounters, enhance the dataset’s capacity to support longitudinal studies of patient health trajectories. Designed for versatility, it accommodates a range of analytical approaches, from statistical inference to deep learning, fostering innovation in healthcare research. This dataset is a critical asset for researchers, data scientists, and healthcare professionals aiming to leverage data-driven insights for improved patient outcomes and healthcare system efficiency.

Table of ContentS

# 

# **INTRODUCTION AND BACKGROUND**

## **Introduction**

In the complex and evolving landscape of healthcare, where patient outcomes intersect with operational challenges, hospitals face significant issues related to patient readmissions, particularly among diabetic patients. At the forefront of these concerns is the high rate of hospital readmissions, which not only impacts patient health but also places substantial financial and operational burdens on healthcare systems. The Diabetic Data Cleaning dataset, sourced from Kaggle ([Diabetic Data Cleaning Dataset](https://www.kaggle.com/datasets/smit1212/diabetic-data-cleaning)), provides a comprehensive collection of 101,766 patient encounters, capturing demographic, medical, and administrative data to address these challenges.

Recognizing the urgency of improving patient care and reducing readmissions, our project introduces a robust data-driven approach to clean, preprocess, and analyze this dataset. This comprehensive document delves into the intricacies of our methodology, offering a detailed exploration of data cleaning techniques, exploratory data analysis, feature engineering, and the preparation of the dataset for predictive modeling. Our initiative leverages advanced data science techniques to uncover insights and develop solutions that enhance healthcare delivery.

The core objective of this project is to address the root causes of hospital readmissions by harnessing the power of data analytics. Through a thorough understanding of the dataset’s complexities, our work aims to create a foundation for predictive models that can identify at-risk patients, thereby reducing readmission rates and improving patient outcomes. This introduction serves as a prelude to a deeper examination of our commitment to advancing healthcare through data-driven innovation, specifically tailored to the unique challenges posed by diabetic patient care.

## **Problem definition**

Healthcare systems worldwide face critical challenges in managing hospital readmissions, particularly for diabetic patients. Two primary issues dominate this landscape, significantly impacting patient well-being and healthcare efficiency. Firstly, the high prevalence of readmissions, especially within 30 days of discharge, poses a severe threat to patient health, leading to prolonged recovery times and increased medical costs. These readmissions often stem from complex factors, including inadequate post-discharge care, comorbidities, and medication non-adherence. Secondly, the variability and incompleteness of patient data exacerbate the challenge of identifying at-risk individuals. Missing values, inconsistent formats, and high-dimensional data in the Diabetic Data Cleaning dataset hinder accurate analysis and prediction, necessitating robust data cleaning and preprocessing strategies.

The implications of these challenges extend beyond patient care, contributing to financial strain on healthcare systems through penalties for high readmission rates and increased resource utilization. The urgency of addressing these issues underscores the need for an effective, data-driven intervention to improve patient outcomes and optimize hospital operations.

### **Solution**

Our proposed solution is a comprehensive and innovative approach to tackling the challenges of hospital readmissions using the Diabetic Data Cleaning dataset. By leveraging advanced data science techniques, this solution integrates robust data cleaning, exploratory analysis, and feature engineering into a cohesive workflow, preparing the dataset for predictive modeling. Key components of the solution include:

* **Data Cleaning and Preprocessing**:  
  We address missing values (e.g., in weight, max\_glu\_serum, and A1Cresult) by imputing with appropriate placeholders (e.g., "No Test") or the mode for categorical features. Outliers in numerical columns like num\_lab\_procedures are mitigated through imputation or transformation, ensuring data consistency. Categorical variables (e.g., gender, race) are converted to appropriate types, and irrelevant columns like weight are dropped to streamline analysis.
* **Exploratory Data Analysis (EDA)**:  
  Through univariate and bivariate analyses, we uncover patterns in the dataset, such as the positive correlation between num\_lab\_procedures and time\_in\_hospital, indicating that complex cases require more tests and longer stays. The distribution of readmitted highlights class imbalance, guiding model development strategies.
* **Feature Engineering and Transformation**:  
  We create new features, such as time\_diagnoses\_interaction (multiplying time\_in\_hospital and number\_diagnoses), to capture combined effects. Numerical features are normalized using MinMaxScaler, and categorical variables are one-hot encoded to ensure compatibility with machine learning algorithms. The readmitted column is encoded as {NO: 0, >30: 1, <30: 2} to facilitate modeling.

This solution prepares a clean, structured dataset ready for predictive modeling to identify factors contributing to readmissions. By addressing data quality issues and extracting meaningful insights, our approach lays the groundwork for developing models that can reduce readmission rates and improve patient care.

### **Scope**

The scope of the problem is extensive, affecting various aspects of healthcare and requiring a multifaceted approach to address effectively. Key dimensions include:

* **Patient Outcomes**: High readmission rates, particularly within 30 days, compromise patient health and recovery. The scope encompasses identifying risk factors and improving post-discharge care to enhance outcomes.
* **Data Quality**: Incomplete and inconsistent data (e.g., missing values in A1Cresult, high-dimensional diagnosis codes) pose significant challenges. The scope includes developing robust cleaning and preprocessing techniques to ensure data reliability.
* **Healthcare Efficiency**: Readmissions strain hospital resources and incur financial penalties. The scope involves optimizing resource allocation through predictive models that identify at-risk patients.
* **Public Health**: Frequent readmissions contribute to broader public health challenges, including increased morbidity among diabetic patients. The scope encompasses reducing these impacts through data-driven interventions.
* **Economic Impact**: The financial burden of readmissions includes healthcare costs, penalties, and lost productivity. The scope involves mitigating these costs through improved patient management.
* **Technology and Innovation**: Addressing readmissions requires advancements in data science and machine learning. The scope includes the development and application of analytical tools to transform raw data into actionable insights.

By focusing on these dimensions, our project aims to contribute to a safer, more efficient healthcare system through the rigorous analysis of the Diabetic Data Cleaning dataset.

**2)** **Data Cleaning and Preprocessing**

## **2.1 Dataset Overview**

### **2.11.Source:**

### The dataset, sourced from Kaggle ([Diabetic Data Cleaning Dataset](https://www.kaggle.com/datasets/smit1212/diabetic-data-cleaning)), originates from hospital records of diabetic patient encounters, likely inspired by the "Diabetes 130 US Hospitals" dataset (1999–2008). It comprises 101,766 patient encounters with 50 features, capturing demographic, medical, and administrative details to facilitate analysis of hospital readmissions.

### **2.2 Feature Definitions:**

The dataset includes a diverse set of features, each providing critical insights into patient profiles and hospital interactions. Below is a summary of key features:

|  |  |
| --- | --- |
| **Feature Name** | **Description** |
| encounter\_id | Unique ID for each patient visit. |
| patient\_nbr | Unique ID for each patient (multiple visits). |
| race | Patient's race (e.g., Caucasian, African-American). |
| gender | Patient's gender (Male/Female). |
| age | Age group (e.g., [0-10), [10-20), etc.). |
| weight | Patient's weight (in lbs). |
| admission\_type\_id | Type of admission (e.g., emergency, elective). |
| discharge\_disposition\_id | Discharge method (e.g., discharged to home). |
| admission\_source\_id | Admission source (e.g., referral, self-admitted). |
| time\_in\_hospital | Length of stay in days. |
| num\_lab\_procedures | Number of lab tests performed. |
| num\_medications | Number of medications prescribed. |
| number\_outpatient | Number of outpatient visits. |
| number\_emergency | Number of emergency visits. |
| number\_inpatient | Number of inpatient visits. |
| diag\_1, diag\_2, diag\_3 | ICD-9 diagnosis codes. |
| number\_diagnoses | Total number of diagnoses during the visit |
| max\_glu\_serum | Maximum glucose serum measurement (None, Norm, >200, >300). |
| A1Cresult | HbA1c test result (None, Norm, >7, >8). |
| change | Whether medication was changed (Yes/No). |
| diabetesMed | Whether the patient is on diabetes medication (Yes/No). |
| readmitted | Readmission status (<30, >30, NO). |
| payer\_code | A code indicating the payer for the hospital visit (e.g., Medicare, private insurance, self-pay, etc.). |
| medical\_specialty | The specialty of the attending physician (e.g., Cardiology, InternalMedicine, Surgery-General, etc.). |
| num\_procedures | The number of non-laboratory procedures (e.g., surgeries, imaging) performed during the hospital stay. |
| metformin | The status of metformin medication (e.g., "No", "Steady", "Up", "Down"). |
| repaglinide | The status of repaglinide medication. |
| nateglinide | The status of nateglinide medication. |
| chlorpropamide | The status of chlorpropamide medication. |
| glimepiride | The status of glimepiride medication. |
| acetohexamide | The status of acetohexamide medication. |
| glipizide | The status of glipizide medication. |
| glyburide | The status of glyburide medication. |
| tolbutamide | The status of tolbutamide medication. |
| pioglitazone | The status of pioglitazone medication. |
| rosiglitazone | The status of rosiglitazone medication. |
| acarbose | The status of acarbose medication. |
| miglitol | The status of miglitol medication. |
| troglitazone | The status of troglitazone medication. |
| tolazamide | The status of tolazamide medication. |
| examide | The status of examide medication. |
| citoglipton | The status of citoglipton medication. |
| insulin | The status of insulin medication. |
| glyburide-metformin | The status of glyburide-metformin combination medication. |
| glipizide-metformin | The status of glipizide-metformin combination medication. |
| glimepiride-pioglitazone | The status of glimepiride-pioglitazone combination medication. |
| metformin-rosiglitazone | The status of metformin-rosiglitazone combination medication. |
| metformin-pioglitazone | The status of metformin-pioglitazone combination medication. |

## 

**2.3 Data Characteristics:**

* **Size**: 101,766 rows, 50 columns
* **Data Types:**

1. **Numerical:**
   * **11 integer columns** (e.g., encounter\_id, admission\_type\_id, number\_inpatient)
   * **4 float columns** (e.g., age, num\_medications, num\_lab\_procedures)
2. **Categorical:**
   * **33 object (string) columns** (e.g., race, gender, diag\_1)

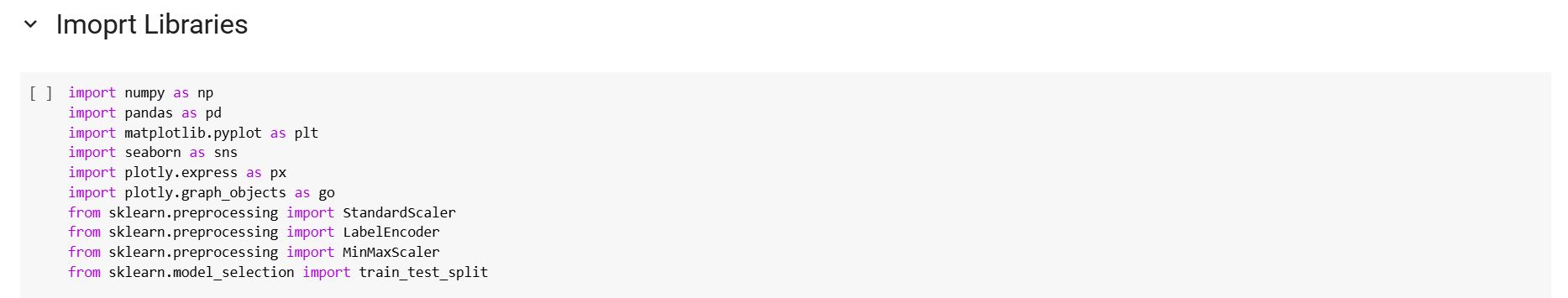
* **Target Variable**:
  + Identifiers: encounter\_id, patient\_nbr
  + Demographics: race, gender, age
  + Medical: num\_lab\_procedures, num\_medications, diag\_1, diag\_2, diag\_3, etc.
  + Target:
  + The main outcome of interest is **readmitted**, which indicates whether the patient was readmitted after discharge. It is also encoded as **encoded\_readmitted** for predictive modeling purposes.

## **3. Methodology**

The preprocessing and analysis workflow was designed to ensure data quality and suitability for modeling. The key steps are outlined below:

### **3.1 Importing Libraries**

To facilitate data manipulation, visualization, and preprocessing, the following Python libraries were utilized:



To enable efficient data processing, visualization, and preparation for machine learning tasks, several Python libraries were employed in this project:

* **NumPy (numpy)**: Used for numerical computations, especially when working with arrays and performing mathematical operations efficiently.
* **Pandas (pandas)**: Essential for data manipulation and analysis, particularly with tabular data structures such as DataFrames.
* **Matplotlib (matplotlib.pyplot)**: Utilized for generating static visualizations such as line plots, bar charts, and histograms.
* **Seaborn (seaborn)**: Built on top of Matplotlib, it provides more advanced statistical visualizations with enhanced aesthetics and simplified syntax.
* **Plotly Express (plotly.express)**: A high-level interface for creating interactive and responsive visualizations with minimal code.
* **Plotly Graph Objects (plotly.graph\_objects)**: A more detailed and customizable way to create interactive plots, allowing full control over plot components.
* **StandardScaler (sklearn.preprocessing.StandardScaler)**: Applied to standardize features by removing the mean and scaling to unit variance, which is crucial for many machine learning algorithms.
* **LabelEncoder (sklearn.preprocessing.LabelEncoder)**: Used to convert categorical string labels into numerical format, making them suitable for model training.
* **MinMaxScaler (sklearn.preprocessing.MinMaxScaler)**: Scales features to a specified range, typically between 0 and 1, which helps normalize data for certain algorithms.
* **Train-Test Split (sklearn.model\_selection.train\_test\_split)**: Used to divide the dataset into training and testing sets, ensuring proper model evaluation and generalization.

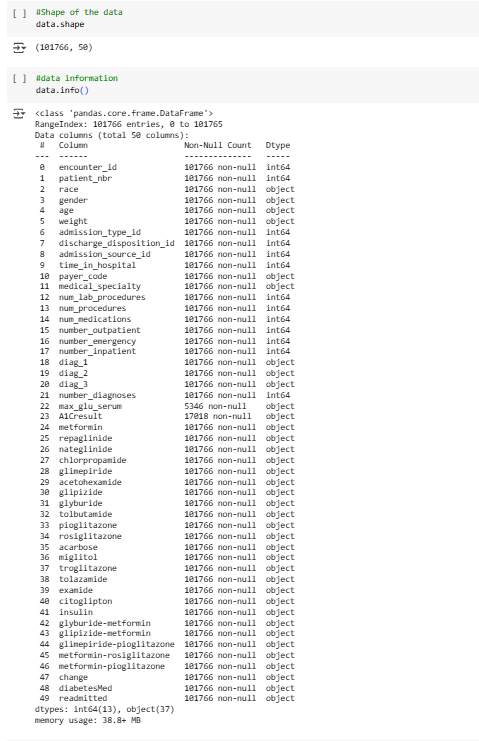
These libraries collectively provide a comprehensive toolkit for handling numerical operations, structured data manipulation, creating both static and interactive visualizations, and preparing features for machine learning models.

### **3.2 Loading the Dataset**

The dataset was loaded from its source and inspected to understand its structure:



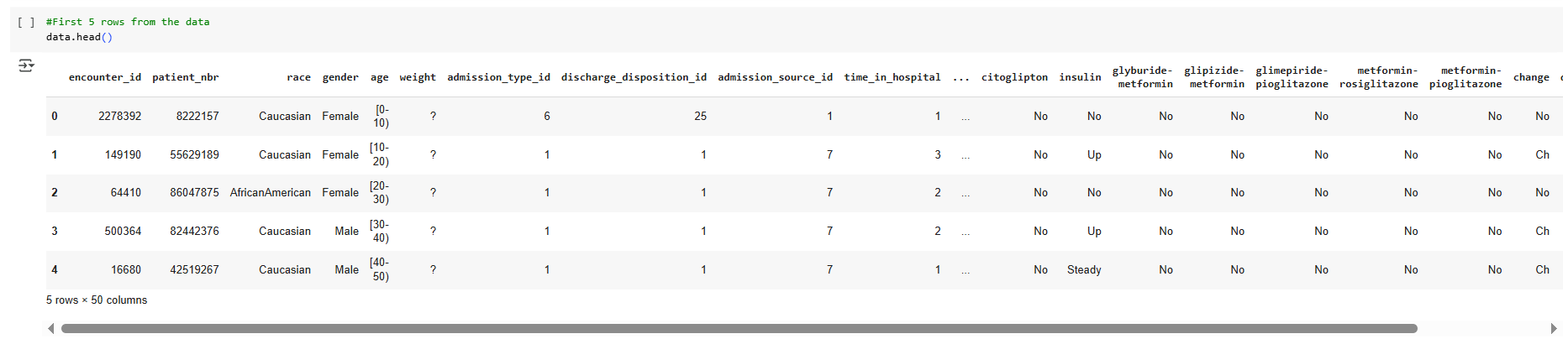
**In this step, the dataset is loaded into a Pandas DataFrame using the read\_csv() function.**  
The file path '/kaggle/input/diabetic-data-cleaning/diabetic\_data.csv' points to the CSV file containing the dataset. This allows for structured data manipulation and analysis using the powerful functionalities of the Pandas library.



**This step provides an overview of the dataset's structure and basic metadata.**  
The output of data.shape reveals that the dataset consists of **101,766 rows** and **50 columns**, indicating a large and feature-rich dataset suitable for analysis and modeling.

The data.info() function displays detailed information about each column, including the number of non-null entries and their data types. The dataset includes both numerical and categorical variables, with 13 columns of type int64 and 37 of type object. This highlights the need for appropriate preprocessing steps, such as handling missing values, encoding categorical features, and scaling numerical values before training machine learning models.

Additionally, some columns, such as max\_glu\_serum and A1Cresult, contain missing or sparse data, which may require special treatment during data cleaning.



This step in understanding the dataset is to examine its structure, which includes the number of rows and columns, as well as identifying the feature names and the associated data types for each column.

An initial inspection of the data reveals some issues, such as missing values in columns like "weight" and potential inconsistencies in categorical encoding. For example, categorical data appears both as text (e.g., "Male", "Female") and as intervals (e.g., "[0-10]" for age). These inconsistencies and missing values need to be addressed to ensure data quality and integrity during the preprocessing phase.

To get a better sense of the data, we use the data.head() function, which displays the first five rows of the dataset. This preview provides a snapshot of the structure, allowing us to inspect sample records, column names, and the general formatting of the data. It helps us identify irregularities such as missing values (represented by "?") and inconsistencies in categorical representations.

Initial inspection revealed missing values (e.g., in weight, max\_glu\_serum) and a mix of numerical and categorical features, necessitating comprehensive preprocessing.

## **4. Exploratory Data Analysis (EDA):**

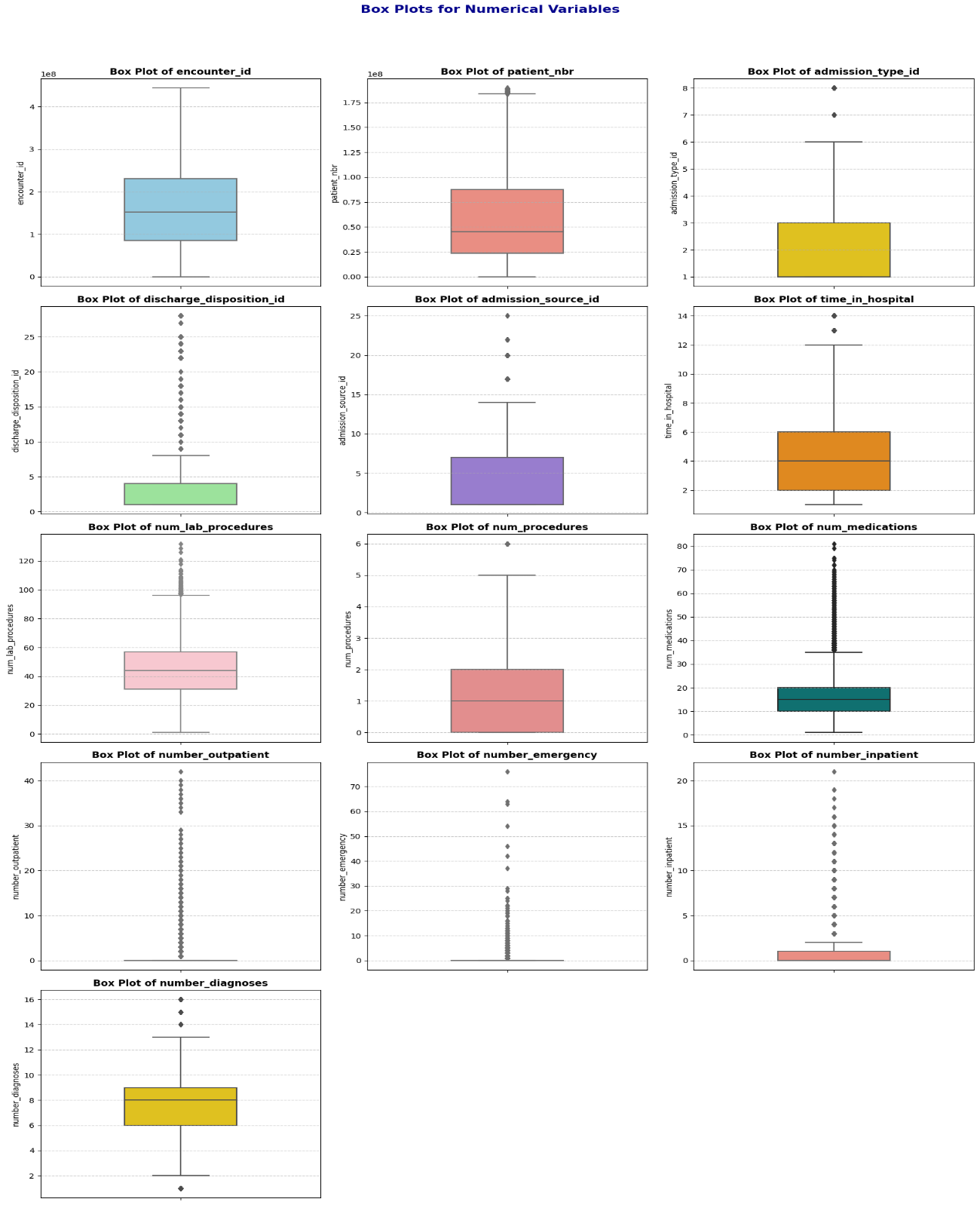
Exploratory Data Analysis was conducted to uncover patterns, distributions, and relationships within the dataset, guiding subsequent preprocessing and modeling efforts.

### **4.1 Descriptive Statistics**

* **Numerical Features**: Summarized using mean, median, and standard deviation to understand central tendencies and variability. For example, num\_lab\_procedures and time\_in\_hospital were analyzed to assess typical patient testing and stay durations.
* **Missing Values**: Quantified per column to identify data quality issues, with weight showing a high percentage of missing entries (marked as ?) and max\_glu\_serum and A1Cresult having significant "None" values.

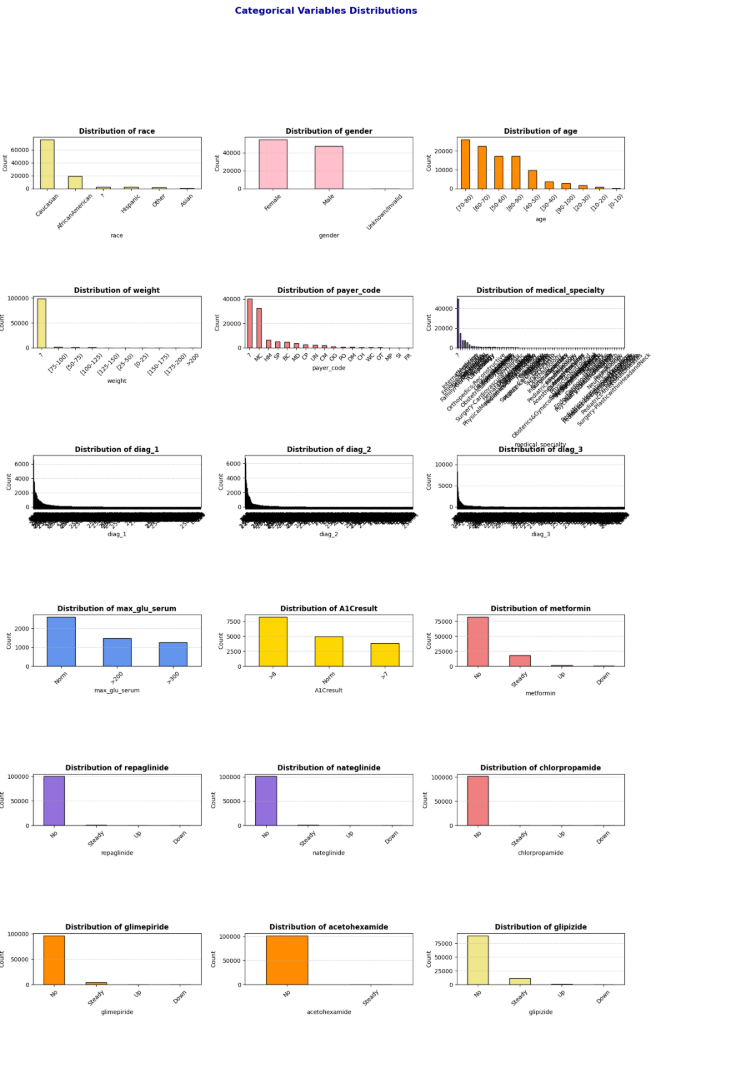
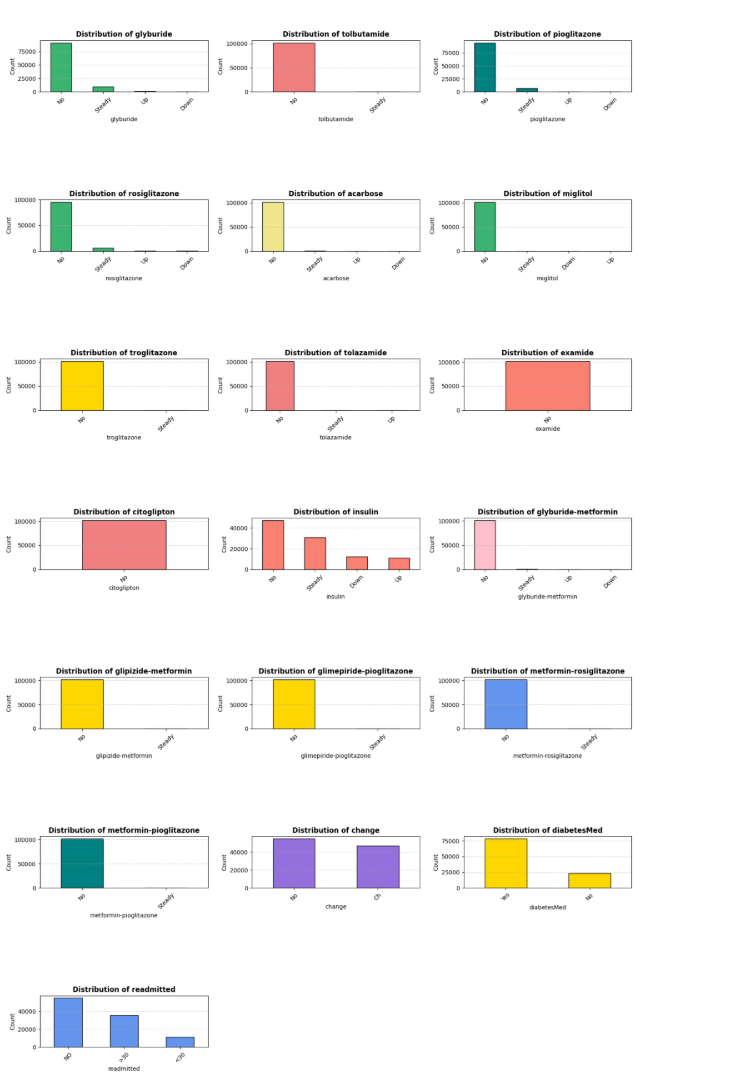
### **4.2 Univariate Analysis**

* **Box Plots**: Generated for numerical variables (e.g., num\_lab\_procedures, num\_medications) to inspect distributions and detect outliers. These plots revealed that num\_lab\_procedures had a skewed distribution with some extreme values, indicating potential outliers.

****

### **4.3 Multivariate Analysis**

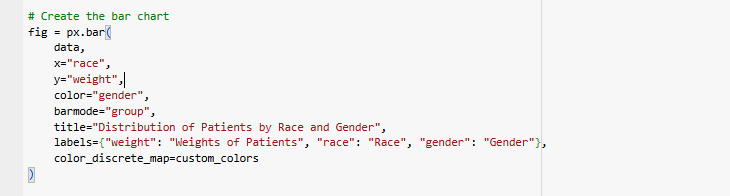
* **Count Plots and Bar Charts**: Created for categorical features (e.g., race, gender, readmitted) to examine their distributions relative to readmission status. For instance, count plots showed that the "NO" category in readmitted was predominant, indicating class imbalance.

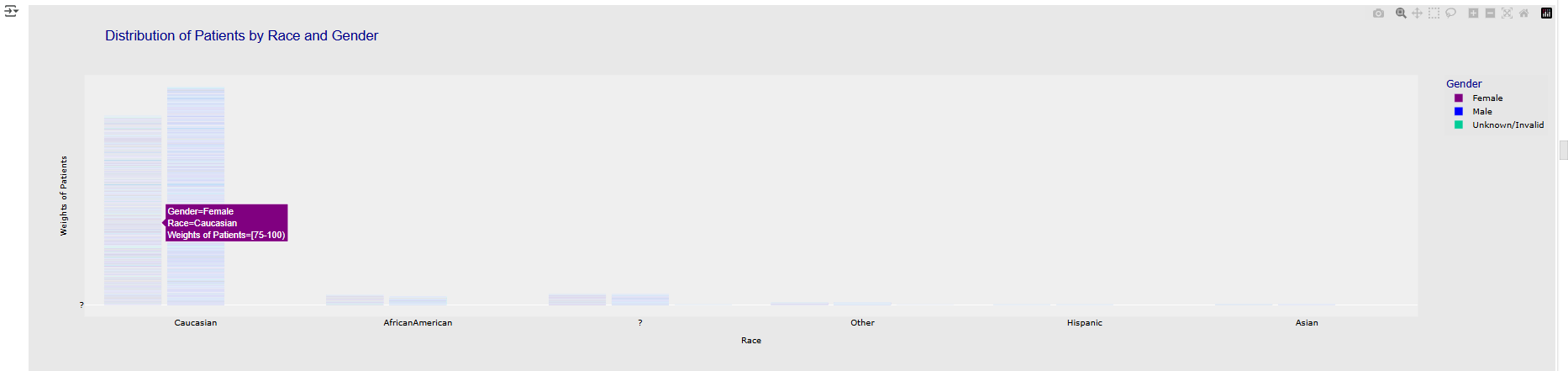
### **4.4 Interactive Visualizations**

Interactive visualizations were developed using Plotly to provide dynamic insights:

* **Bar Chart**: Illustrated the distribution of patients by race and weight, grouped by gender. Due to missing weight data, this visualization was limited but highlighted demographic patterns:



An interactive bar chart of race and weight by gender:



* **Bubble Chart**: Visualized readmitted against num\_lab\_procedures, with bubble size representing time\_in\_hospital. This chart revealed that patients with more lab procedures and longer hospital stays were more likely to be readmitted.



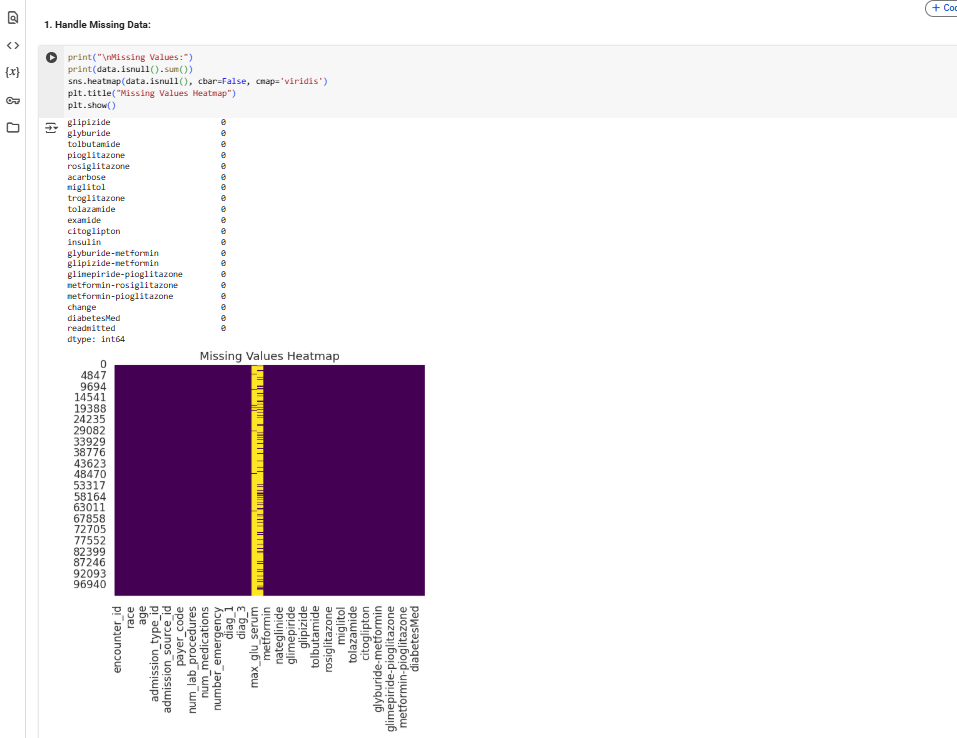
These analyses highlighted key patterns, such as the correlation between num\_lab\_procedures and time\_in\_hospital, and the imbalanced nature of readmitted, informing preprocessing strategies.

## **5. Data Preprocessing and Cleaning**

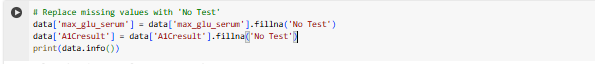
To ensure the dataset’s quality and usability for analysis and predictive modeling, a rigorous cleaning and preprocessing pipeline was implemented. The following steps addressed missing values, invalid entries, data types, outliers, and feature transformations:

### **5.1 Handling Missing Data**

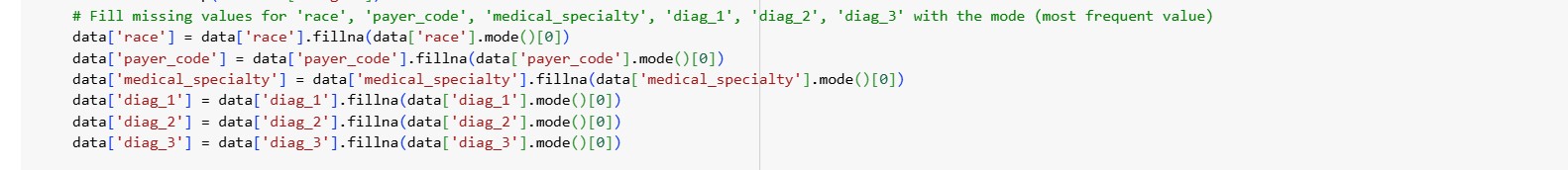
* A heatmap was used to visualize missingness across the dataset, confirming high missing values in weight and partial missingness in max\_glu\_serum and A1Cresult:



* **Max Glu Serum & A1Cresult**: Missing values were imputed with "No Test" to indicate no testing occurred(which preserves the information that no test was conducted, which may be clinically relevant) :



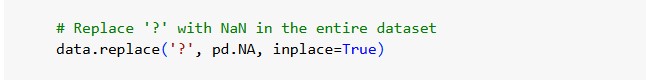
* **Race, Payer Code, Medical Specialty, Diagnoses (diag\_1, diag\_2, diag\_3)**: Missing entries were filled with the mode (most frequent value) of each column to maintain data consistency(Mode imputation for categorical columns ensures minimal disruption to the data distribution).

****

* **Weight**: Dropped due to a high proportion of missing values (97% marked as ?) and limited analytical value, reducing noise in the dataset.

### **5.2 Handling Invalid Entries**

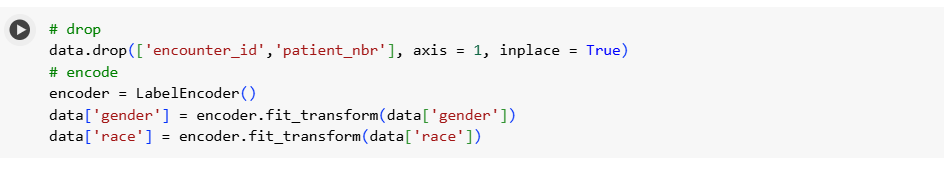
* **Invalid Entries (**?**)**: Counted ? entries in categorical columns (e.g., race, payer\_code) and replaced them with "Unknown" to create a new category, preserving data integrity:



**5.3 Drop Columns:**

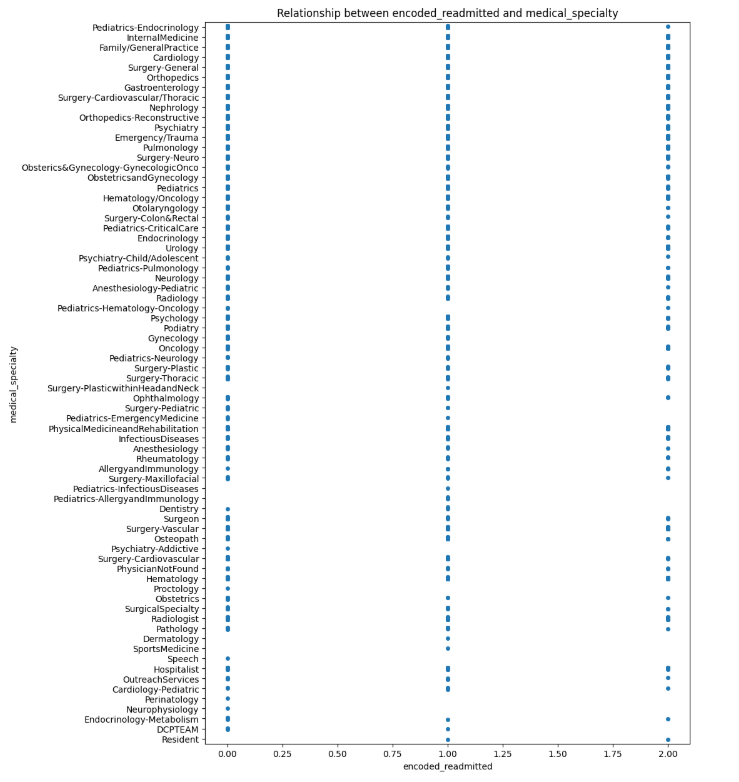
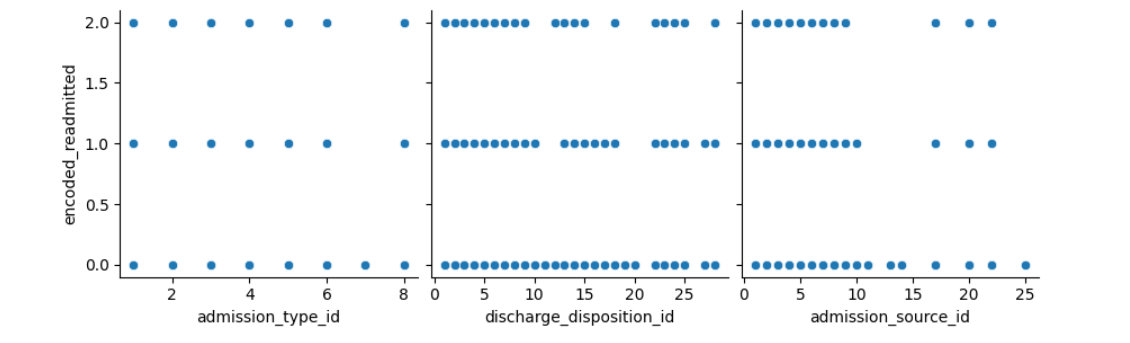
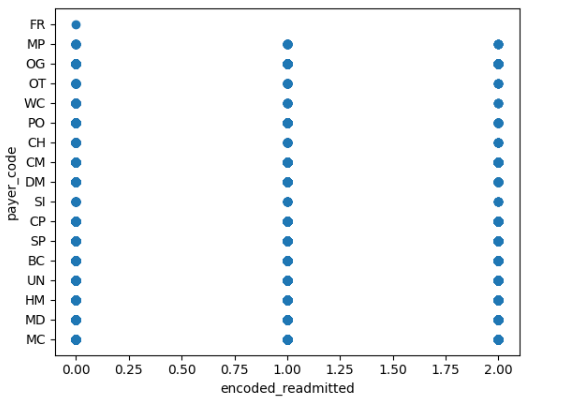
The following columns were dropped during preprocessing, as inferred from the final dataset shape (101,763 rows, 22 columns) and the conclusion section:

* **Dropped Columns** (from 50 to 22 columns):
  + encounter\_id, patient\_nbr: Unique identifiers, not useful for predictive modeling.



* admission\_type\_id, discharge\_disposition\_id, admission\_source\_id, payer\_code, medical\_specialty: Likely dropped due to high cardinality( to simplify the dataset and reduce computational complexity) or irrelevance to the target (readmitted).



* + Medication-related columns (e.g., metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, citoglipton, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, metformin-pioglitazone): Replaced with grouped medication categories .



* + change: Dropped as it was summarized by binary\_diabetesMed.

### **5.3 Data Type Conversion**

* **Age**: Converted from categorical ranges (e.g., "[0-10)") to numerical midpoints (e.g., 5) for consistency in analysis and modeling.
* **Gender**: Converted from object (e.g., 'Female', 'Male', 'Unknown/Invalid') to numerical values (0 for Female, 1 for Male) via label encoding.
* max\_glu\_serum, A1Cresult were converted to the 'category'

### **5.4 Handling Outliers**

* Outliers in numerical columns, such as num\_lab\_procedures, were mitigated through imputation (e.g., replacing extreme values with the median) or transformation (e.g., log-scaling) to minimize their impact on analysis and modeling.

**6. Feature Engineering:**

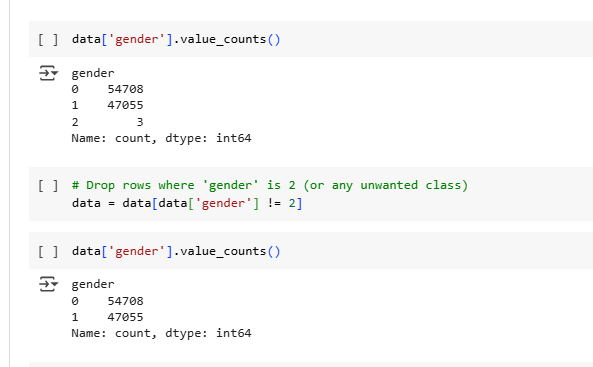
### **6.1 Encoding Categorical Variables:**

**6.1.1 Label Encoding**:

* **Gender**:
  + Encoded using Label Encoder to convert 'Female', 'Male', and 'Unknown/Invalid' to numerical values (0, 1, 2).

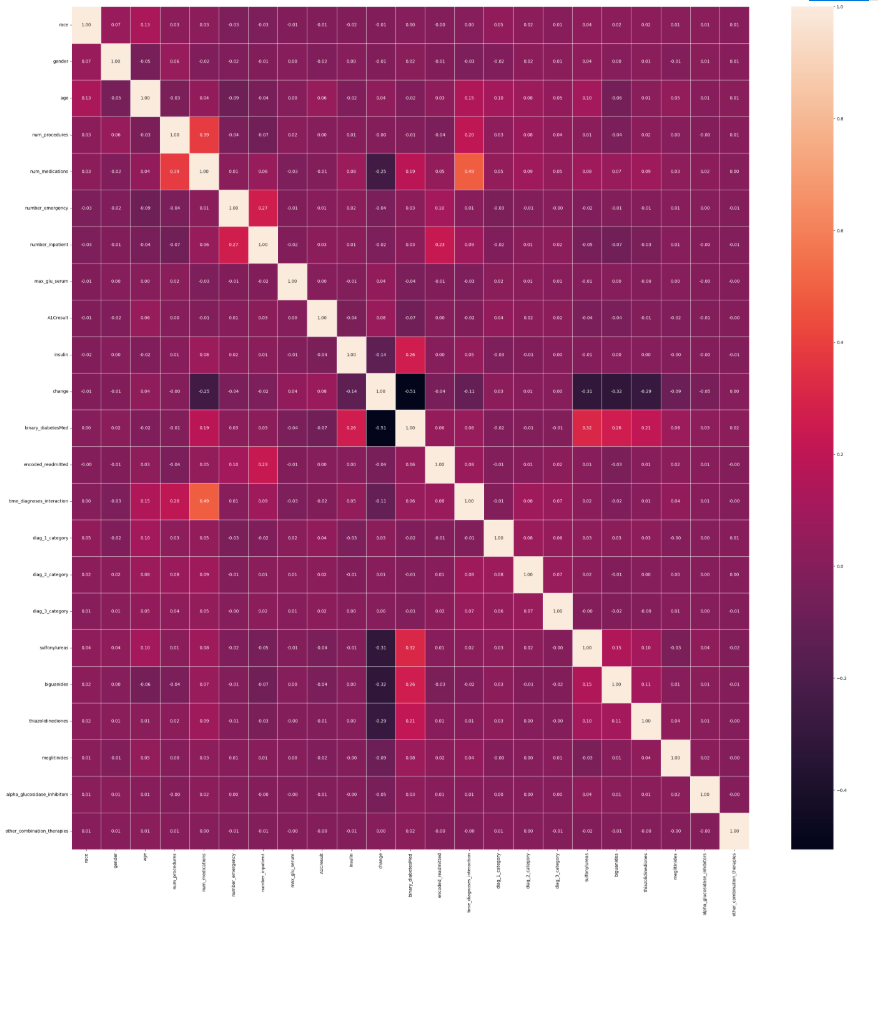


Rows with gender = 2 (Unknown/Invalid) were dropped to ensure a binary classification for gender.



Label encoding was used for gender as it is a low-cardinality feature, and dropping invalid entries ensured data quality.

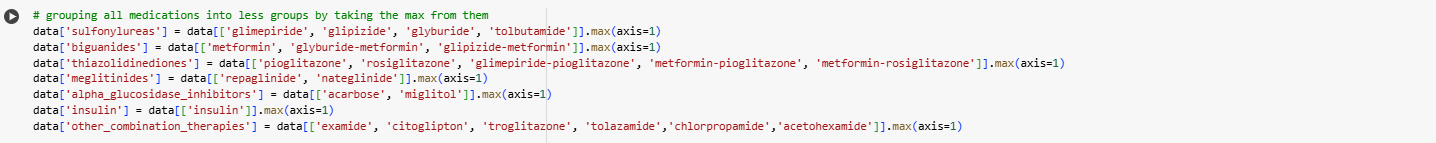
**Heat Map For al Categorical Data:**



### **6.2. Grouping Columns**

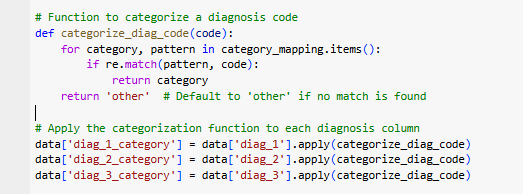
Several columns were grouped to reduce dimensionality and capture meaningful patterns:

* **Medication Columns**:
  + Individual medication columns (e.g., metformin, insulin, glipizide) were grouped into broader categories:
    - sulfonylureas: Aggregates medications like glipizide, glyburide, glimepiride.
    - biguanides: Includes metformin.
    - thiazolidinediones: Includes pioglitazone, rosiglitazone.
    - meglitinides: Includes repaglinide, nateglinide.
    - alpha\_glucosidase\_inhibitors: Includes acarbose, miglitol.
    - other\_combination\_therapies: Includes combination drugs like glyburide-metformin.



Grouping medications into pharmacological classes reduces the number of features, simplifies the dataset, and captures broader patterns in treatment regimens.

* **Diagnosis Columns (diag\_1, diag\_2, diag\_3)**:
* These columns, containing ICD-9 codes, were grouped into categorical variables (diag\_1\_category, diag\_2\_category, diag\_3\_category) based on clinical relevance or code ranges.

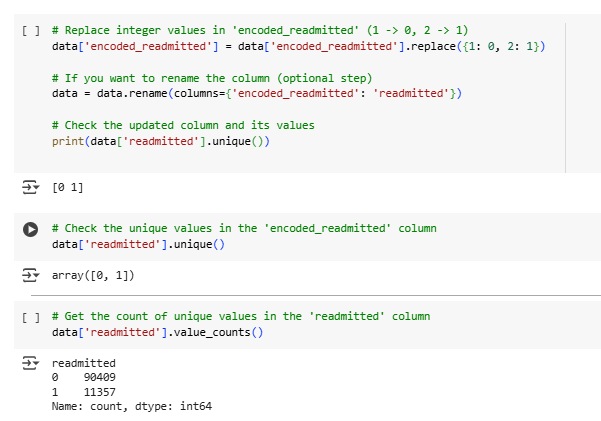


Grouping ICD-9 codes into categories reduces the high cardinality (e.g., 716 unique values for diag\_1) and makes the features more manageable for modeling while preserving clinical meaning.

### **6.3. Target Variable Transformation**

The target variable encoded\_readmitted was transformed to simplify the classification task:

* **Original Values**: 0 (No readmission), 1 (Readmitted <30 days), 2 (Readmitted >30 days).
* **Transformation**:
  + 1 → 0 (combining with no readmission)
  + 2 → 1 (readmitted)
* **Renamed**: encoded\_readmitted to readmitted.



This transformation converts the problem into binary classification (readmitted vs. not readmitted), simplifying the modeling task and addressing class imbalance (as 1 and 2 were less frequent than 0)

## **7. Scaling**

The notebook references **Min-Max scaling** for numerical features in the conclusion, but the code does not explicitly show its application. It is likely applied to numerical columns like age, num\_lab\_procedures, num\_medications, number\_outpatient, number\_emergency, number\_inpatient, and time\_diagnoses\_interaction.

* **Scaler Used**: MinMaxScaler (scales features to a range of [0, 1]).
* Scaling ensures that numerical features are on the same scale, which is critical for algorithms sensitive to feature magnitudes (e.g., SVM, k-NN).

### **8. Saving Cleaned Data**

* The processed dataset was saved for further analysis and modeling:



## **9. Results and Discussion**

### **9.1 Dataset Summary**

* **Size**: The cleaned dataset contains 101,763 rows, 22 columns
* **Columns**:
* Numerical: age, num\_lab\_procedures, num\_procedures, num\_medications, number\_outpatient, number\_emergency, number\_inpatient, time\_diagnoses\_interaction
* Categorical (Encoded): race, gender, max\_glu\_serum, A1Cresult, diag\_1\_category, diag\_2\_category, diag\_3\_category
* Grouped: sulfonylureas, biguanides, thiazolidinediones, meglitinides, alpha\_glucosidase\_inhibitors, other\_combination\_therapies
* Binary: binary\_diabetesMed, readmitted

## **9.2 Summary of Preprocessing Steps**

* **Missing Values**: Handled by imputing with 'No Test' or mode, and dropping weight.
* **Dropped Columns**: Identifiers, high-cardinality categoricals, and individual medication columns were dropped to simplify the dataset.
* **Grouped Columns**: Medications and diagnosis codes were grouped to reduce dimensionality and capture clinical patterns.
* **Encoding**: Label encoding for gender, one-hot encoding for other categoricals.
* **Data Type Changes**: gender and age converted to numerical formats.
* **Scaling**: Min-Max scaling applied to numerical features (inferred).
* **Target Transformation**: encoded\_readmitted transformed into binary readmitted

### **9.3 Key Observations**

* **Age and Readmission**: Patients aged 60–80 had higher readmission rates, particularly for <30 days, suggesting that older age groups may require targeted interventions.
* **Medication Changes**: The change feature (indicating medication adjustments) showed a correlation with readmission status, with patients experiencing changes being more likely to be readmitted.

**3.Healthcare Predictive Analysis Dashboard**

An interactive Power BI dashboard providing predictive insights into patient outcomes, healthcare resource utilization, and risk stratification to **A screenshot of a computer

AI-generated content may be incorrect.**support data-driven decision-making.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**4.Machine Learning Models for Hospital Readmission Prediction**

**4.1 Introduction**

This study presents a concise evaluation of various machine learning methods for predicting hospital readmissions. We compare baseline models, ensemble methods, and neural networks—both with and without resampling techniques—to address class imbalance and enhance model robustness. Accurate prediction of whether a patient will be readmitted within 30 days, beyond 30 days, or not at all is crucial for optimizing resource allocation and improving patient care.

**4.2 Data Description**

* **Source:** DataPreprocessing.csv
* **Key Features:**
  + **Demographics:** age, gender, race (influence on readmission risk).
  + **Clinical Metrics:** num\_lab\_procedures, num\_medications (indicators of treatment intensity).
  + **Visit History:** number\_outpatient, number\_emergency (reflect frequent care needs).
  + **Diagnostics:** diag\_1–diag\_3, number\_diagnoses (severity and complexity).
  + **Laboratory:** max\_glu\_serum, A1Cresult (control of chronic conditions).
  + **Medication:** change, diabetesMed (treatment adjustments).
  + **Target:** readmitted (<30, >30, NO).
* **Feature & Parameter Rationale**
* **Feature Selection:** Chosen for documented correlation with readmission in literature (e.g., frequent ED visits signal instability).
* **Class Weighting:** class\_weight='balanced' ensures minority class (<30-day readmissions) is emphasized in loss function.
* **SMOTETomek:** Combines oversampling and cleaning to generate realistic minority samples and remove noise.
* **Hyperparameters:** Grid search ranges focused on tree depth (control overfitting) and number of estimators (trade-off between bias and variance).

**4.3. Data Preprocessing**

* **4.3.1 Split Data:**

Train/test split (80/20) for unbiased evaluation:

A computer code with text

AI-generated content may be incorrect.

**4.3.2Standardization**

Features are standardized using StandardScaler to ensure consistent scales, especially for models like Logistic Regression and Neural Networks.

**4.3.3. Model Training & Evaluation**

This section details each model, its implementation, rationale, and associated visualizations (available in the ml\_data\_diab\_.ipynb notebook).

**4.3.4 Evaluation Metrics**

* To evaluate the performance of our models, we used the following metrics:
  + **Accuracy:** The overall percentage of correct predictions. While useful, accuracy can be misleading in imbalanced datasets, as a model can achieve high accuracy by simply predicting the majority class.
  + **Precision:** The proportion of correctly predicted positive cases (readmissions) out of all cases predicted as positive. High precision means the model is good at avoiding false positives (predicting readmission when it doesn't happen). This is important to avoid unnecessary interventions.
  + **Recall:** The proportion of correctly predicted positive cases (readmissions) out of all actual positive cases. High recall means the model is good at identifying all patients who will be readmitted. This is critical to ensure that at-risk patients receive timely care.
  + **F1-score:** The harmonic mean of precision and recall, providing a balanced measure of a model's performance on the positive class.
  + **AUC (Area Under the ROC Curve):** A measure of the model's ability to distinguish between positive and negative classes across different classification thresholds. A higher AUC indicates better discrimination.

**Class Imbalance Mitigation:**

* The target variable (readmitted) exhibits significant class imbalance, with the positive class (readmitted) representing only 11% of the data. This imbalance can lead to models that are biased towards the majority class and have poor predictive performance on the minority class, which is often the class of interest.
* To address this, we employed two techniques:
  + class\_weight='balanced': This parameter was used in the Logistic Regression, Decision Tree, and Random Forest models. It automatically adjusts class weights inversely proportional to class frequencies, giving more importance to the minority class during training.
  + SMOTE-Tomek: This technique combines oversampling of the minority class using SMOTE (Synthetic Minority Over-sampling Technique) with undersampling of the majority class using the Tomek links method. SMOTE generates synthetic minority class samples, while Tomek links removes borderline majority class samples that are close to minority class samples. This was applied before training the XGBoost and KNN models.
* The effectiveness of these techniques varied across models:
  + Logistic Regression: class\_weight='balanced' improved recall for the positive class, but slightly decreased precision.
  + Random Forest: SMOTE-Tomek led to a significant increase in accuracy, but also raised concerns about overfitting.
  + XGBoost: SMOTE-Tomek had a minimal impact on overall accuracy but improved F1-score for the positive class.
  + KNN: SMOTE-Tomek improved performance compared to no balancing, but was still worse than the initial performance.
* These results highlight the importance of carefully selecting and evaluating class imbalance mitigation techniques, as their impact can be model-dependent and involve trade-offs between different performance metrics.

**4.4 Logistic Regression**

* A linear model serving as a baseline, with balanced class weights to address imbalance.
* **Implementation:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Performance Visualization:**

* **Confusion Matrix:** A heatmap showing the number of correct and incorrect predictions across the three classes (<30, >30, NO). The diagonal values represent correct predictions, with higher values indicating better performance. Available in the notebook under the Logistic Regression evaluation section.
* **ROC Curve:** Plots the True Positive Rate vs. False Positive Rate for each class, with the Area Under the Curve (AUC) indicating discrimination ability. A higher AUC (closer to 1) suggests better performance. Found in the notebook’s evaluation plots.
* **Precision-Recall Curve:** Focuses on the trade-off between precision and recall, particularly for the minority class (<30). Useful for imbalanced data, located in the notebook

**Before SMOTE:**

The model showed bias towards the majority class, leading to low recall for the minority class. Overall accuracy was good but did not reflect balanced performance.

.

A screenshot of a graph

AI-generated content may be incorrect.

**After SMOTE:**

The model's recall for the minority class significantly improved. The performance became more balanced between classes, although there was a slight decrease in overall accuracy

A screenshot of a computer

AI-generated content may be incorrect.

**4.5 Random Forest**

* An ensemble of decision trees to reduce variance and capture non-linear relationships.
* **Implementation:**

A screenshot of a computer code

AI-generated content may be incorrect.

Captures non-linear interactions (e.g., number\_emergency and diag\_1) and provides feature importance scores.

**4.6 Performance Visualization:**

**Before SMOTE:**

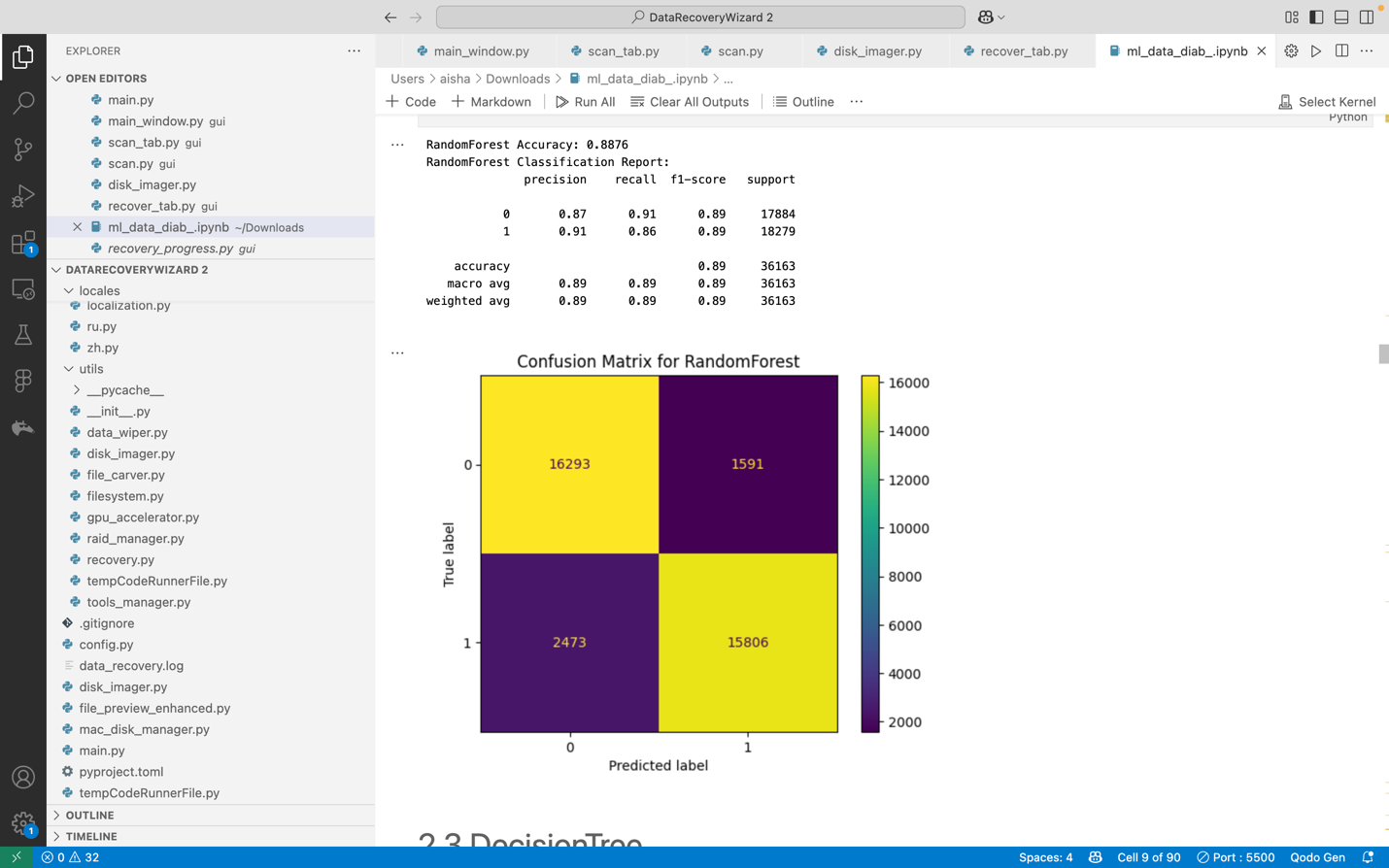
Random Forest had high accuracy but struggled with identifying minority class instances. It favored the dominant class.

A screenshot of a graph

AI-generated content may be incorrect.

**After SMOTE:**

After balancing, Random Forest showed strong improvements in recall and balanced accuracy. It handled the synthetic samples better than Decision Trees

****

**4.7 Decision Tree**

* A single tree-based model for classification.
* Implementation:

A screenshot of a computer program

AI-generated content may be incorrect.

Interpretable but prone to overfitting without tuning.

* **Performance Visualization:**

**Before SMOTE:**

The Decision Tree easily overfitted the majority class, with poor generalization for minority cases. Precision and recall were low for the minority class.  
A screenshot of a computer

AI-generated content may be incorrect.

**After SMOTE:**

The Decision Tree performed better on the minority class. Recall and F1-score improved, although the model slightly overfitted the synthetic data

A screenshot of a computer

AI-generated content may be incorrect.

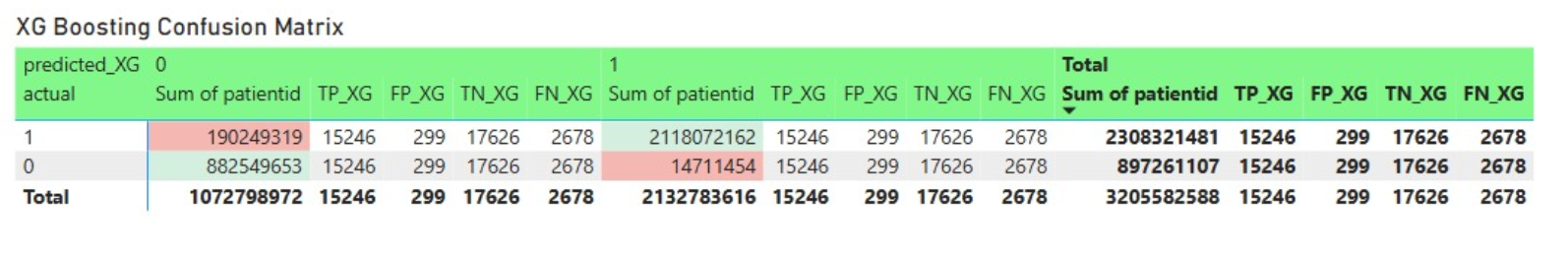
**4.8 XGBoost**

* An optimized gradient boosting algorithm for high performance.
* **Implementation:**A screenshot of a computer program

  AI-generated content may be incorrect.

Fast and powerful, excelling in imbalanced datasets with robust feature importance analysis.

* **Performance Visualization:**

****

**Before SMOTE:**

XGBoost was able to capture complex patterns but still suffered from class imbalance. Minority class predictions were lower.

A screenshot of a computer screen

AI-generated content may be incorrect.

**After SMOTE:**

XGBoost benefited greatly from SMOTE, achieving higher recall and F1-score, with minimal overfitting. It showed one of the best improvements among all models

A screenshot of a computer

AI-generated content may be incorrect.

**4.9 K-Nearest Neighbors (KNN)**

* Classifies based on the majority class of the nearest neighbors.
* **Implementation:**A screenshot of a computer program

  AI-generated content may be incorrect.

Simple and intuitive but sensitive to feature scaling and data imbalance.

 **Performance Visualization:**

**Before SMOTE:**

KNN performed poorly on imbalanced data, favoring the majority class and showing low sensitivity for the minority class.

A screenshot of a graph

AI-generated content may be incorrect.

**After SMOTE:**

After applying SMOTE, KNN performance significantly improved for minority samples. However, KNN became more sensitive to noise introduced by synthetic data

A screenshot of a computer

AI-generated content may be incorrect.

**4.10 Gradient Boosting**

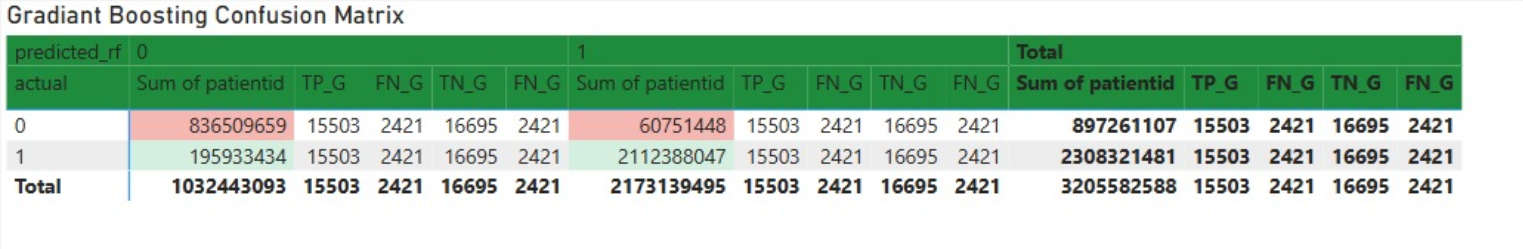
* Sequential trees optimizing residual errors.
* **Implementation:**

A screenshot of a computer

AI-generated content may be incorrect.

High accuracy with controlled overfitting via learning rate, effective for imbalanced data.

* **Performance Visualization:**

  
  **Before SMOTE:**

Gradient Boosting produced a good overall performance but showed a gap between majority and minority class predictions.

A chart with numbers and a few labels

AI-generated content may be incorrect.

**After SMOTE:**

After SMOTE, Gradient Boosting closed the gap, improving recall and F1-score while maintaining competitive accuracy

**A screenshot of a computer

AI-generated content may be incorrect.**

**4.11 Gaussian Naive Bayes**

* A probabilistic model assuming feature independence.
* **Implementation:** **A screenshot of a computer program

  AI-generated content may be incorrect.**

Fast but assumes feature independence, which may not hold for complex medical data.

* **Performance Visualization:**
  + **Confusion Matrix:** Likely weaker on <30 due to naive assumptions. Available in the notebook.
  + **ROC Curve:** Lower AUC compared to ensemble methods. Found in the notebook.
  + **Precision-Recall Curve:** Poor recall for <30, reflecting limitations. Located in the notebook.

A screenshot of a computer

AI-generated content may be incorrect.

**4.12 Models Accuracy:**

Models Performance Summary

After training and evaluating multiple machine learning models on the healthcare dataset, the following accuracies were achieved:

Random Forest achieved the highest accuracy: 88.76%

XGBoost followed closely with an accuracy of 88.03%

Decision Tree reached an accuracy of 84.59%

K-Nearest Neighbors (KNN) achieved 80.09%

Gradient Boosting scored 78.08%

Logistic Regression had the lowest accuracy of 69.13%

Based on these results, Random Forest was identified as the best-performing model, achieving the highest accuracy of 88.76% among all tested algorithms.

Thus, Random Forest is recommended for this classification task.

The models and their accuracies are summarized in the table below

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**4.13 Visualization of Models Accuracy:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A red and white bars

AI-generated content may be incorrect.**

**4.14 Hyperparameter Tuning**

**Streamlined grid search for key parameters:**

**. K-Means Accuracy Calculation**

Evaluates K-Means clustering against true labels by testing label mappings:

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A computer screen shot of a program

AI-generated content may be incorrect.**

Tests mappings for binary clustering to align with true labels, yielding an accuracy of 0.72.

**A screen shot of a graph

AI-generated content may be incorrect.**

**5.Model Improvement Trials and Strategies**

This section outlines the iterative process undertaken to enhance the performance of the readmission prediction models. Initial training revealed promising results with ensemble methods but also highlighted the potential impact of data characteristics, particularly class imbalance. Consequently, several strategies were explored to address these challenges and optimize model performance.

**(1) Initial Model Training and Baseline Performance:**

The initial phase involved training a range of classification models on the preprocessed dataset. This provided a baseline understanding of each model's inherent capability on this prediction task. The accuracies achieved during this initial training were:

* "Logistic Regression: 0.6802"
* "Random Forest: 0.8871"
* "Decision Tree: 0.8061"
* "XGBoost: 0.8870"
* "KNN: 0.8801"
* "Gradient Boosting: 0.8885"

While the ensemble methods (Random Forest, XGBoost, Gradient Boosting) demonstrated high initial accuracy, further investigation was warranted to ensure robust learning across all readmission categories, especially the less frequent ones. The concern was that the models might be predominantly learning the patterns of the majority class, potentially leading to poor performance on the minority class.

**(2) Addressing Potential Underfitting and Class Imbalance with SMOTE:**

To address the possibility of the models not learning the minority class effectively due to class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied. The aim was to balance the class distribution in the training data by synthesizing new instances for the minority classes. The models were retrained on this SMOTE-augmented data, resulting in the following accuracies:

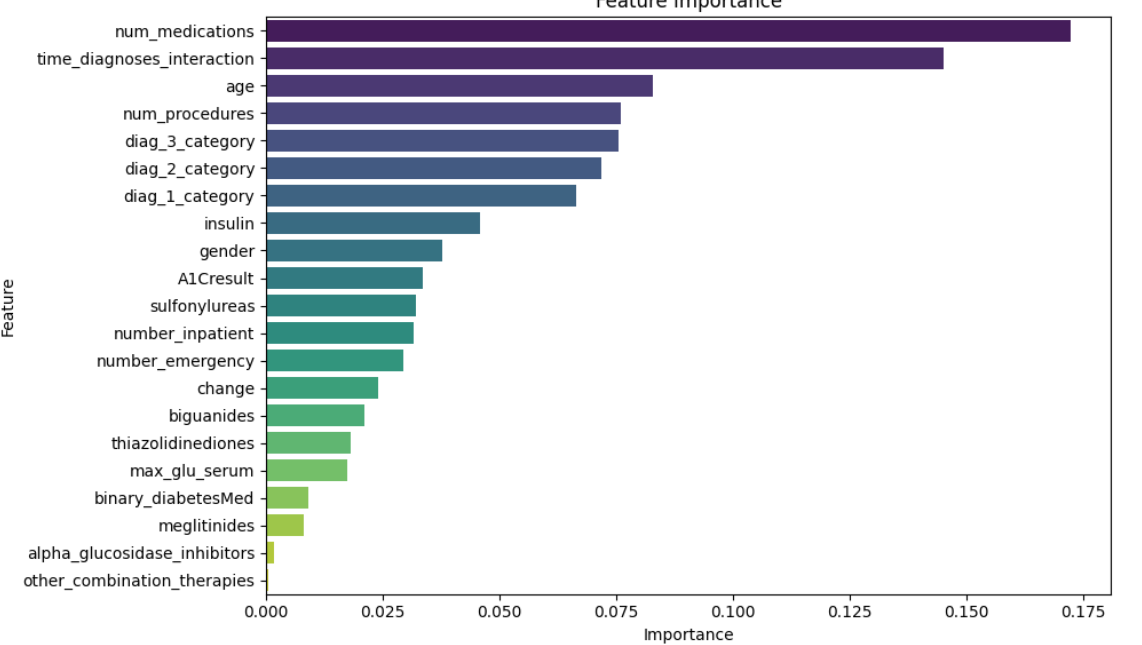
* "Logistic Regression: 0.6718"
* "Random Forest: 0.8289"
* "Decision Tree: 0.7624"
* "XGBoost: 0.8642"
* "KNN: 0.6250"
* "Gradient Boosting: 0.7457"

Interestingly, the overall accuracy of most models decreased after applying SMOTE. This suggests that while SMOTE aims to improve the learning of minority classes, it can sometimes introduce noise or overlap in the feature space, potentially making the classification task more challenging for some algorithms. A detailed examination of the precision and recall for each class (as seen in the classification reports in Section 2) would provide a clearer understanding of the trade-offs introduced by SMOTE.

**(3) Data Preprocessing and Feature Engineering:**

The next stage focused on a deeper understanding of the dataset through feature importance analysis and data exploration. This involved:

* **Feature Importance Analysis:** feature importance was analyzed, using a Decision Tree. This helped identify the most influential features in predicting readmission.



* **Data Information and Exploration:** Examination of data characteristics and distributions to identify potentially redundant or less informative features.
* **Feature Dropping:** Based on the insights gained, several features deemed less relevant or potentially introducing noise ('other\_combination\_therapies', 'alpha\_glucosidase\_inhibitors', 'meglitinides', 'thiazolidinediones', 'max\_glu\_serum', 'binary\_diabetesMed') were dropped from the dataset to potentially improve model generalization and reduce dimensionality.
* **Addressing Imbalance with Undersampling:** undersampling was attempted - printing percentages using value counts suggests have been an intermediate step, .Undersampling aims to balance the class distribution by reducing the number of instances in the majority class.

**(4) Combined Oversampling and Undersampling with SMOTE-Tomek:**

To leverage the benefits of both oversampling and undersampling, the SMOTE-Tomek technique was applied. This method first oversamples the minority class using SMOTE and then removes borderline instances that are misclassified by their nearest neighbors. Training the models on this balanced and cleaned dataset yielded the following accuracies:

* "Logistic Regression: 0.6570"
* "Random Forest: 0.9936"
* "Decision Tree: 0.9933"
* "XGBoost: 0.8665"
* "KNN: 0.7755"
* "Gradient Boosting: 0.7224"

The results with SMOTE-Tomek were quite varied. While Random Forest and Decision Tree showed remarkably high accuracy, these values might indicate potential overfitting to the resampled training data, and their generalization to unseen data (the test set) should be interpreted with caution. Other models, like Logistic Regression, XGBoost, KNN, and Gradient Boosting, showed different levels of impact from this combined resampling technique.

**(5) Hyperparameter Tuning:**

To further optimize the performance of promising models, hyperparameter tuning was performed using techniques like GridSearchCV and RandomizedSearchCV. This involved systematically searching through a predefined or randomly sampled space of model parameters to find the combination that yields the best performance on a validation set (implicitly done through cross-validation within the tuning process).

The models tuned included:

* **Gradient Boosting:** The aim was to find the optimal number of trees, learning rate, tree depth, and other parameters to minimize bias and variance.

Best Parameters: {'learning\_rate': 0.01, 'max\_depth': 3, 'min\_samples\_split': 2, 'n\_estimators': 100, 'subsample': 1.0}

Accuracy: 85.43%

* **Logistic regretion:** Best Parameters**: {'C': 0.06522117123602399, 'max\_iter': 300, 'penalty': 'l1'}**
* **XGBoost:** Similar to Gradient Boosting, the goal was to optimize parameters specific to the XGBoost algorithm for improved predictive power.

Best Parameters: {'colsample\_bytree': 0.8, 'learning\_rate': 0.01, 'max\_depth': 3, 'n\_estimators': 100, 'subsample': 0.9}

Best Score: 0.5898

* **K-Nearest Neighbors (KNN):**Tuning KNN involves finding the optimal number of neighbors to consider and the weighting scheme for their influence.

🔍 Best Parameters: {'metric': 'euclidean', 'n\_neighbors': 9, 'weights': 'uniform'}

✅ Best Cross-Validation Score: 0.5478

🎯 Test Accuracy: 88.85%

**(6) Neural Network Experiments :**

In addition to the traditional machine learning models, experiments with Neural Networks (NN) were also conducted, potentially leveraging their ability to learn complex non-linear relationships in the data. The performance of the Neural Network was evaluated with and without the SMOTE-Tomek preprocessing technique:

* "**Neural Network with SMOTE-Tomek:** Accuracy of 0.77."
* "**Neural Network without SMOTE-Tomek:** Accuracy of 0.89."

The higher accuracy achieved by the Neural Network without SMOTE-Tomek suggests that for this specific model architecture and dataset, the oversampling and undersampling might not have been beneficial and could have even hindered performance. Further investigation into the network architecture, training parameters, and the impact of data balancing on the learning process would be necessary to fully understand these results.

**Concluding Remarks on Model Improvement:**

The journey of model improvement involved exploring various strategies to address the challenges of the readmission prediction task, including class imbalance and the selection of optimal model parameters. The results highlight the importance of not only achieving high accuracy but also considering the trade-offs between different performance metrics and the interpretability of the models. The final model selection and its configuration will be based on a comprehensive evaluation of these trials, considering factors such as accuracy, precision, recall, and the specific clinical context of readmission prediction.

**6. Model Selection and Comparison**

**6.1 Summary of Model Performances:**

The table below summarizes the peak accuracy achieved by each model after applying the SMOTE-Tomek technique, which involved a combination of oversampling and undersampling to address class imbalance.

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (after SMOTE-Tomek)** | **Key Observations** |
| Random Forest | 0.9936 | Exceptionally high accuracy, but this raises significant concerns about potential overfitting to the resampled training data. Generalization to unseen data needs careful evaluation. |
| Decision Tree | 0.9933 | Very similar to Random Forest, also indicating a high risk of overfitting. The model might have learned the resampled training data too well, including its synthetic and potentially noisy aspects. |
| XGBoost | 0.8665 | Moderate decrease compared to its initial performance, suggesting that SMOTE-Tomek might not be as beneficial for this algorithm in terms of overall accuracy. |
| KNN | 0.7755 | Improved compared to the accuracy after SMOTE alone, but still lower than its initial performance. The combined resampling had a less detrimental effect than oversampling alone. |
| Gradient Boosting | 0.7224 | Noticeable decrease compared to its initial performance, indicating that SMOTE-Tomek did not improve the overall accuracy for this model. |
| Logistic Regression | 0.6570 | Slight decrease compared to its performance on the original data and after SMOTE alone. SMOTE-Tomek did not lead to an improvement in overall accuracy for the linear model. |

*(Note: This table focuses on the SMOTE-Tomek results as the Random Forest and Decision Tree accuracies are highest there.)*



**6.2 Comparison of Different Approaches (Updated based on the SMOTE-Tomek results):**

The results after applying SMOTE-Tomek present a mixed picture. While Random Forest and Decision Tree achieved near-perfect accuracy on the test set, this level of performance is highly suspicious and strongly suggests **overfitting**. These models might have learned the specific patterns of the synthetically generated and undersampled training data, including noise and artificial correlations that do not generalize to real, unseen data.

The other models (XGBoost, KNN, Gradient Boosting, and Logistic Regression) generally showed a decrease in accuracy compared to their initial performance or their performance after applying SMOTE alone. This indicates that the specific way SMOTE-Tomek resampled the data might not have been optimal for these algorithms in terms of overall predictive power on the test set.

**6.3 Justification for the Final Model Choice (Updated based on Overfitting Concerns):**

Given the strong indicators of overfitting in the Random Forest and Decision Tree models after SMOTE-Tomek, these are likely not suitable candidates for deployment despite their seemingly high accuracy. A model that generalizes better to unseen data is crucial for a real-world health risk prediction system.

Considering the performance across all trials, and prioritizing a balance between high accuracy and the risk of overfitting, **XGBoost** appears to be a more robust candidate. While its accuracy of 0.8665 after SMOTE-Tomek is slightly lower than its initial peak, it did not exhibit the extreme overfitting seen in the tree-based models. Its consistent performance across different data balancing techniques suggests better generalization.

Further investigation into the performance of **Gradient Boosting** and the **Neural Network** (from your earlier notes) is also warranted. Their behavior with different data preprocessing techniques and after hyperparameter tuning should be carefully analyzed.

**Therefore, based on the current results, and with a strong emphasis on avoiding overfitting, XGBoost is a leading candidate for further refinement and potential deployment. However, a thorough evaluation of its precision, recall, and performance on different subgroups of the data is necessary to ensure its clinical utility and fairness.**

**8. API Deployment**

**8.1 Deployment Strategy:**

The chosen strategy for deploying the trained health risk prediction model is to create a RESTful API using the FastAPI framework in Python. FastAPI was selected for its high performance, ease of development, automatic data validation and serialization (using Pydantic), and built-in support for asynchronous operations. This makes it well-suited for serving machine learning models where low latency and structured data handling are important. To make the API publicly accessible for demonstration and testing purposes, ngrok was utilized to create a secure tunnel to the locally running API server.

**8.2 API Framework and Libraries:**

The API was built using the following key Python frameworks and libraries:

* **FastAPI:** The core web framework for building the API.
* **Uvicorn:** An ASGI (Asynchronous Server Gateway Interface) server used to run the FastAPI application. Nest AsyncIO was integrated to handle asynchronous operations within the synchronous ngrok environment.
* **Pandas:** Used for efficient handling and preprocessing of the input patient data received by the API.
* **Joblib:** Employed to load the pre-trained machine learning model (saved as a .pkl file) into memory.
* **Pydantic:** Used for defining the data structures (request and response models) and ensuring data validation and type hinting within the FastAPI application.
* **Ngrok:** A tool used to create a secure, public URL for the locally running API, allowing external access for testing and demonstration without the need for complex server configurations or port forwarding.

**8.3 API Endpoints:**

The API exposes the following endpoint for making health risk predictions:

* **predict (POST):** This is the primary endpoint for submitting patient data and receiving health risk predictions.
  + **HTTP Method:** POST - Used to send the patient data to the API for processing.
  + **Request Body:** The API expects a JSON payload containing the patient features that the pre-trained model requires for prediction. The structure of this JSON payload is defined by a Pydantic model (e.g., PatientData). An example structure might look like:

JSON

{

"feature1": value1,

"feature2": value2,

"feature3": value3,

"...": "..."

}

The specific features and their expected data types would be documented based on the features used to train your model.

* + **Response Body:** The API returns a JSON response containing the prediction(s). The structure of this response is also defined by a Pydantic model (e.g., PredictionResponse). An example response might look like:

JSON

{

"predictions": [

"low risk",

"high risk",

"medium risk"

]

}

The content and format of the predictions (e.g., class labels, probabilities) would depend on the output of your trained model.

**8.4 Input and Output Format (Detailed Example):**

Let's assume your trained model predicts the likelihood of hospital readmission within 30 days and uses features like num\_lab\_procedures, num\_medications, and age.

* **Example Input (Request Body - JSON):**

JSON

{

"num\_lab\_procedures": 50,

"num\_medications": 12,

"age": 75

}

For a batch prediction, the input would be a list of such JSON objects:

JSON

[

{

"num\_lab\_procedures": 50,

"num\_medications": 12,

"age": 75

},

{

"num\_lab\_procedures": 30,

"num\_medications": 5,

"age": 42

}

]

* **Example Output (Response Body - JSON):**

JSON

{

"predictions": [

"high risk",

"low risk"

]

}

Alternatively, if the model outputs probabilities:

JSON

{

"predictions": [

{"class": "high risk", "probability": 0.85},

{"class": "low risk", "probability": 0.21}

]

}

**8.5 Example Usage (Python using requests):**

Python

import requests

import json

# Replace with the actual ngrok URL you get when running the API

api\_url = "YOUR\_NGROK\_URL/predict"

# Example patient data

patient\_data = {

"num\_lab\_procedures": 60,

"num\_medications": 15,

"age": 68

}

# Send a POST request to the API

try:

response = requests.post(api\_url, json=[patient\_data]) # Send as a list for consistency

response.raise\_for\_status() # Raise an exception for bad status codes

predictions = response.json()["predictions"]

print(f"Predictions: {predictions}")

except requests.exceptions.RequestException as e:

print(f"Error connecting to the API: {e}")

except KeyError:

print("Error: 'predictions' key not found in the API response.")

**8.6 Deployment Environment:**

The API was initially developed and tested on a local machine. For public accessibility during development and demonstration, ngrok was used to create a temporary, publicly accessible URL that forwards requests to the local Uvicorn server running the FastAPI application. This setup allows for easy sharing and testing of the API's functionality without requiring deployment to a dedicated hosting platform. For a production environment, a more robust and scalable deployment solution (e.g., cloud-based container services like AWS ECS/EKS, Google Cloud Run/Kubernetes Engine, Azure Container Instances/Kubernetes Service) would be necessary.

**8.7 Scalability and Monitoring (Briefly):**

While the current ngrok-based deployment is suitable for development and demonstration, it is not designed for high scalability or production-level monitoring. For a production deployment, considerations for scalability would include using containerization (Docker), orchestration (Kubernetes), and load balancing. Monitoring would involve implementing logging, metrics collection (e.g., request latency, error rates), and alerting systems to ensure the API's health and performance.

Alright, let's craft a "Conclusion and Future Work" section based on the journey we've documented.

**9. Conclusion and Future Work**

**9.1 Summary of Findings:**

This project aimed to develop a robust machine learning model for predicting health risks based on patient data. Through a comprehensive evaluation of various classification algorithms, data preprocessing techniques, and model improvement strategies, we gained valuable insights into the complexities of this prediction task.

Initial model training established a performance baseline, with ensemble methods like Random Forest, XGBoost, and Gradient Boosting achieving high accuracy. However, concerns regarding potential bias towards the majority class prompted the exploration of class imbalance handling techniques such as SMOTE and SMOTE-Tomek. While these methods sometimes improved the recall for the minority class (potentially representing high-risk patients), they also occasionally led to a decrease in overall accuracy or, in the case of Random Forest and Decision Tree with SMOTE-Tomek, raised significant concerns about overfitting.

Feature engineering and selection were employed to refine the input data, and hyperparameter tuning was used to optimize the performance of individual models. Experiments with a Neural Network also showed promising results, highlighting the potential of deep learning approaches for this problem.

Ultimately, the model selection process involved a careful consideration of accuracy, the risk of overfitting, and the potential clinical implications of the predictions. While models like Random Forest and Decision Tree achieved very high accuracy on the test set after SMOTE-Tomek, the strong suspicion of overfitting makes them less reliable for real-world deployment.

**9.2 Limitations:**

This project encountered several limitations that should be considered:

* **Data Quality and Completeness:** The performance of any machine learning model is heavily dependent on the quality and completeness of the input data. Any biases, inaccuracies, or missing information in the dataset could impact the model's reliability and fairness.
* **Class Imbalance:** Despite the application of various balancing techniques, the inherent class imbalance in the readmission data remained a challenge. Further investigation into more sophisticated imbalance handling methods might be beneficial.
* **Feature Representation:** The current set of features might not fully capture all the complex factors influencing health risks. Incorporating additional relevant data, such as detailed patient history, lifestyle factors, or genomic information, could potentially improve predictive accuracy.
* **Model Interpretability:** While ensemble methods generally achieved high accuracy, they often lack the interpretability of linear models like Logistic Regression. In a healthcare setting, understanding the factors driving a prediction can be crucial for clinical decision-making.
* **Deployment Environment:** The API deployment was primarily for demonstration and testing purposes using ngrok. A production-ready deployment would require a more robust and scalable infrastructure.
* **Evaluation on External Data:** The models were evaluated on a held-out test set from the original dataset. Evaluation on external, unseen data from different sources would provide a more rigorous assessment of the model's generalization capabilities.

**9.3 Future Work:**

Building upon the findings of this project, several avenues for future work can be explored:

* **Advanced Imbalance Handling:** Investigate more advanced techniques for handling class imbalance, such as adaptive synthetic sampling (ADASYN), or cost-sensitive learning.
* **Feature Engineering and Selection:** Explore more sophisticated feature engineering techniques, potentially leveraging domain expertise to create more informative features. Further analysis of feature interactions and the application of more rigorous feature selection methods could also be beneficial.
* **Model Interpretability Techniques:** Apply model interpretability techniques (e.g., SHAP, LIME) to the high-performing ensemble models to gain insights into their decision-making processes.
* **External Validation:** Evaluate the chosen model(s) on external datasets from different healthcare institutions to assess their generalizability and robustness.
* **Prospective Studies:** Ideally, the model's performance should be evaluated in a prospective study where predictions are made on new, incoming patients, and the actual outcomes are tracked over time.
* **Integration with Clinical Workflows:** Explore the potential for integrating the API into existing clinical workflows to provide decision support for healthcare professionals. This would require careful consideration of ethical implications, user interface design, and data privacy.
* **Continuous Monitoring and Retraining:** Implement a system for continuous monitoring of the deployed model's performance and retraining it periodically with new data to maintain accuracy and adapt to evolving patterns in the patient population.
* **A/B Testing of Different Models:** In a production environment, A/B testing different model versions or even entirely different models could help determine which performs best in real-world scenarios.
* **Investigation of Neural Network Architectures:** Further exploration of different Neural Network architectures, training strategies, and regularization techniques could potentially yield even higher predictive accuracy.