Predicting Hospital Readmissions in Diabetic Patients Using Stacking, Random Forest, and Gradient Boosting Classifiers

Diabetic patient readmission is a major problem faced by healthcare systems across the globe. Diabetes is a chronic condition that affects over 537 million adults worldwide (IDF Diabetes Atlas, 2021). This disease causes their body to lose the ability to normally control blood sugar levels. There are two main types of diabetes: Type 1, where the pancreas does not make insulin at all and Type 2, the pancreas makes less insulin than it is supposed to, and the rest of your body becomes resistant to insulin (Mayo Clinic, 2023). If either of these happen, they can lead to serious health complications. Managing diabetes correctly is a necessity to prevent further complications, especially with patient hospital discharge. A study published by the Journal of Diabetes Science and Technology in 2021 found that nearly 20% of the patients admitted with diabetes were readmitted to hospitals within 30 days post-discharge (Journal of Diabetes Science and Technology, 2021). These conditions only lead to the increasing costs associated with health care. Readmissions are caused by many complications from diabetes, failure to receive proper follow-up care as well as nonadherence with prescribed treatment plans (PubMed Central, 2023).

The financial impact is huge. In the U.S., the American Diabetes Association reports diabetes costs are around $327 billion each year in healthcare and lost productivity (American Diabetes Association, 2018). In Canada, diabetes costs the healthcare system around $13 billion each year (Diabetes Canada, 2021). Readmissions are a big part of this budget, and hospitals in primarily the United States can face penalties when unplanned readmissions happen, as they're seen as a sign of poor care quality (Centers for Medicare & Medicaid Services, 2022).

The challenge is predicting high-risk patients that are more likely to be readmitted. It's a complex task that involves clinical factors, personal history, and behavioral elements. Traditional predictive methods tend to have low precision scores, which in turn reduces the possibility of physicians making good decisions based on level of care (JAMA Network, 2020).

This project addresses this problem by using machine learning techniques to analyze data from patients, finding patterns and direct correlations that can show a higher chance for readmission. This analysis, if offered to health professionals across the United States and Canada can aid in making preventative strategies while reducing readmission rates and improving the quality of patient care. Moreover, a website helps make all these resources easily accessible to medical professionals, patients, and healthcare institutions, thus saving readmissions and improving outcomes for everyone.

Objectives

This project is aiming to create a machine-learning model that can accurately predict whether or not diabetic patients are readmitted in 30 days from a hospital based on historical data from 130 hospitals throughout the United States. The result of the program will be then published on a publicly available webpage and will ask the user several questions and will decide whether they are at risk of hospitalization. In addition, the created application tries to help healthcare professionals identify high-risk patients early so that better resource allocation, better care for the patients, and lower hospital costs can be achieved.

Variables

The variables used in this project are divided into independent and dependent categories. The independent variables, which are predictors and correlators that influence the likelihood of readmission, include numerical variables such as time in hospital, number of lab procedures, number of procedures, number of medications, number of outpatient visits, number of inpatient visits, and number of emergency visits. Additionally, some categorical variables include age, maximum glucose serum level, and A1C result, while binary variables include the presence of change and diabetes medication.

The dependent variable, or the result the model is predicting, is a binary variable showing whether a patient was readmitted to the hospital or not. Initially, the dataset categorized readmissions into three groups: greater than 30 days, less than 30 days, and no readmission. However, because of the inconsistency and limited representation of readmissions greater than 30 days, these categories were combined into a single "readmission" category.

Certain irrelevant and redundant variables were removed during data preprocessing, including patient id, encounter id, payer code, diagnosis 1, diagnosis 2, diagnosis 3, weight, race, metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, metformin-rosiglitazone, metformin-pioglitazone and medical specialty, as there was no direct correlation that could help find an accurate outcome.

Packages

These Python packages are used in this project:

- pandas: Used for data manipulation and analysis reads the CSV file containing the dataset, cleans data by replacing missing values and droping duplicates.
- numpy: For numerical computations and array operations, replaces missing values.
- scikit-learn: Machine learning library that handles most of the modelling and evaluations in the project.
    - Some functionalities include LabelEncoder, StandardScaler, RandomForest Classifier, GradientBoostingClassifier, Stacking. Classifier, acuracy_score, confusion_matrix and cross_val_score.
- imblearn: handles the imbalanced dataset, uses SMOTEENN, a combination of SMOTE (Synthetic Minority Over-sampling Technique) and ENN (Edited Nearest Neighbors), to balance the dataset by oversampling the minority class and undersampling majority class.
- joblib: Allows parallel processing across multiple machines or cores on one machine.
- matplotlib: For creating visualizations like flowcharts and performance graphs.
- shap: A way to explain the output of any machine learning model through graphs.
- knn-imputer: Part of scikit-learn, used to impute missing values in numerical columns based on the nearest neighbors algorithm.

Procedure

1. Data Acquisition:
    - Load the dataset from the provided CSV file using pandas.
    - Replace missing values with NaN.
2. Data Cleaning:
    - Remove irrelevant columns such as encounter_id, patient_nbr, payer_code, weight, and medical_specialty, as they do not provide meaningful information.
    - Drop duplicate entries to avoid bias in the model.
3. Feature Engineering:
    - Encoding: Convert categorical variables like max_glu_serum and A1Cresult into numerical indices using mapping. Use LabelEncoder for age groups.
    - Transformation: Apply logarithmic transformation to skewed numerical features like time_in_hospital and number_outpatient to reduce the effect of outliers.
    - Normalization: Use StandardScaler to ensure each category equally contributes to the accuracy of the model.
    - Missing Values: Use KNNImputer to fill in missing values for numerical features based on patterns from the nearest neighbors.
4. Class Balancing:

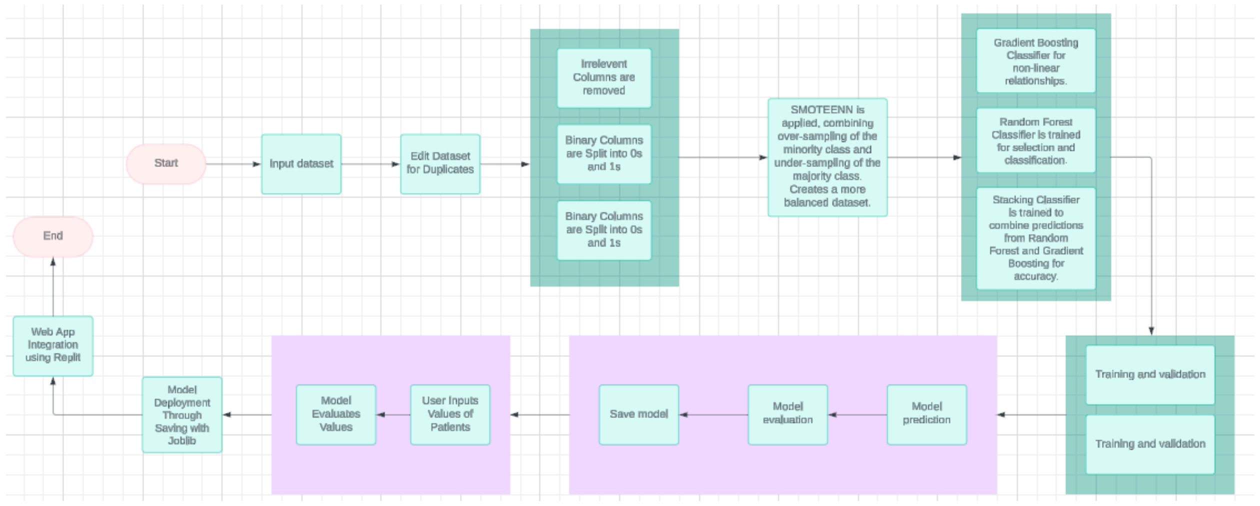
    - Apply SMOTEENN to address the class imbalance between patients readmitted (minority) and not readmitted (majority).
    - Oversample the minority class using SMOTE and clean the dataset by removing noisy examples using ENN.
5. Model Training:
    - Separate dataset into 15% testing and 85% training to avoid overfitting.
    - Use RandomForestClassifier and GradientBoostingClassifier as base models.
    - Combine them into an ensemble using StackingClassifier, where it refines the predictions of the base models.
    - Train the model with the training data and once finished, test it using the testing data.
6. Model Evaluation:
    - Evaluate the trained model using the following:
        - Accuracy: Measure the overall correctness of predictions.
        - Confusion Matrix: Find true positives, true negatives, false positives, and false negatives.
        - Cross-Validation: Perform cross-validation to assess model stability.

7. Deployment:
    - Save the trained model, feature list, and preprocessing tools using joblib.
8. Web Application:
    - Develop a user-friendly interface using Replit.
    - Collect user inputs for relevant features.
    - Preprocess the inputs in real time using the saved preprocessing tools.

Dataset

The dataset was found in UC Irvines' Machine Learning Repository and represents ten years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. Each of the accounts is hospital records of previous patients who had been diagnosed with diabetes. All patients underwent laboratory medications and stayed in the hospital for a maximum of 14 days. Although this dataset may be weak in Canada as hospitals in the United States differ from Canada's healthcare system, it can be related as both countries share similarities in terms of healthcare infrastructure, such as access to medical professionals, use of laboratory tests, and treatments for chronic conditions like diabetes.

Flowchart


Start
Input dataset
Edit Dataset for Duplicates
Irrelevent Columns are removed
Binary Columns are Split into 0s and 1s
Binary Columns are Split into 0s and 1s
SMOTEENN is applied, combining over-sampling of the minority class and under-sampling of the majority class. Creates a more balanced dataset.
Gradient Boosting Classifier for non-linear relationships.
Random Forest Classifier is trained for selection and classification.
Stacking Classifier is trained to combine predictions from Random Forest and Gradient Boosting for accuracy.
Training and validation
Training and validation
Model prediction
Model evaluation
Save model
User Inputs Values of Patients
Model Evaluates Values
Model Deployment Through Saving with Joblib
Web App Integration using Replit
End

Overview

This project shows the importance of predicting hospital readmissions in diabetic patients, which can play an important part in improving patient care and reducing healthcare costs. By analyzing feature importance and using data processing techniques like SMOTEENN, I was able to change the class imbalances to enhance the model performance. The visualizations created through either Matplotlib or Shap provided important insights into the dataset. Visualizing the importance of different features in the prediction process shows which factors most significantly influence the model's decisions. By looking at how the model considers certain variables like hospital visits, medications, and lab procedures, I could see the key predictors of readmission and could find other ways to refine the model, improve it, and create more accurate predictions. Combined, these visualizations provide a proper view of the data distribution and the factors influencing the model's predictions.

Importance of Accessibility

Making this predictive modelling accessible to healthcare providers is extremely important to my project as it can improve the efficiency of patient care. Through Replit, I created a custom-built website which asks the user a series of questions, processes them and then outputs a result of whether they are at risk for rehospitalization. By combining these tools into healthcare systems around the world, professionals can identify high-risk diabetic patients early, allowing for timely interventions and effective care plans. This not only helps reduce readmission rates but also allows resources to be allocated better. The ability to predict readmissions with high accuracy can also lead to better patient outcomes, fewer unnecessary admissions, and improvements in the quality of care provided to diabetic patients.

Sources of Error

Several sources of error could affect this project’s outcomes. Data quality issues, like missing or noisy data, could introduce biases. The class imbalance may still affect model performance. The possible overfitting in complex models like Random Forest and Gradient Boosting could lead to weaker predictions that are inflated, especially if important features were overlooked by my model. Additionally, the dataset might not be fully representative of patient populations as it is outdated and only dealing with 130 hospitals in the United States, possibly introducing additional bias. Furthermore, the evaluation metrics used may not fully capture the model’s performance in handling imbalances or other factors that could influence readmissions.

Accuracy

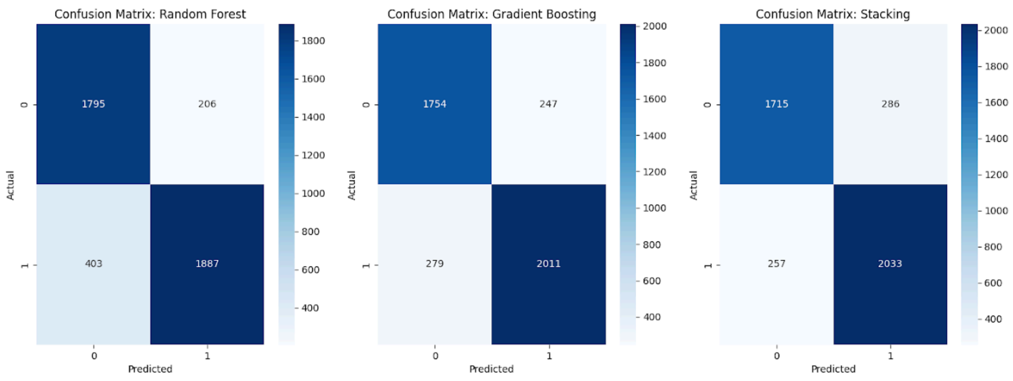
The model's accuracy, on average, is 86.7%, telling us that it performs well in predicting hospital readmissions for diabetic patients. Accuracy is shown through the equation (TP+TN)/(TP+TN+FP+FN), where T means true, P means positive, N means negative and F means false (Fawcett, Tom 2006). It measures the proportion of correctly classified instances. This high accuracy found suggests that the model can correctly classify the many instances, well distinguishing between readmitted and non-admitted patients. However, while accuracy can be used as a clear teller of reliability, it is important to look at other factors, such as class imbalance, which could affect the model’s ability to predict rare events accurately. In cases where the dataset contains uneven distributions of readmitted versus non-readmitted patients, the model’s accuracy might mask performance problems related to false positives or false negatives (Kuhn, Max, and Kjell Johnson 2013). Overfitting may also be possible. While these models are strong, they may not generalize well to new data. As a result, the model might perform well on the training data but have a reduced accuracy with real-world data. Throughout my tests, it has not shown weakness to unseen data but it is still possible. Techniques like cross-validation were also used to prevent overfitting.

Classification Algorithms

The model uses classification algorithms to predict whether a patient will be readmitted to the hospital, a binary outcome encoded as 0 (not readmitted) or 1 (readmitted). More specifically, it uses a combination of RandomForestClassifier, GradientBoostingClassifier, and StackingClassifier. The RandomForestClassifier is an ensemble model that combines the predictions from multiple decision trees, which are each trained on random subsets of the data. This reduces overfitting and improves generalization by averaging the outputs of the trees. As a result, this makes it easier for the model to find nonlinear relationships within the data. The GradientBoostingClassifier, on the other hand, creates decision trees. Each tree, however, focuses on correcting the errors made by the previous ones. By using GradientBoostingClassifier, it improves prediction accuracy. The StackingClassifier combines the strengths of these models by using their predictions as inputs for a final meta-classifier, which further improves the overall performance. To further improve these classifiers, the code has steps like feature scaling with StandardScaler and handling imbalanced data using SMOTEENN.

Data Trends & Model Trends

Figure 1. Heatmap of missing data.

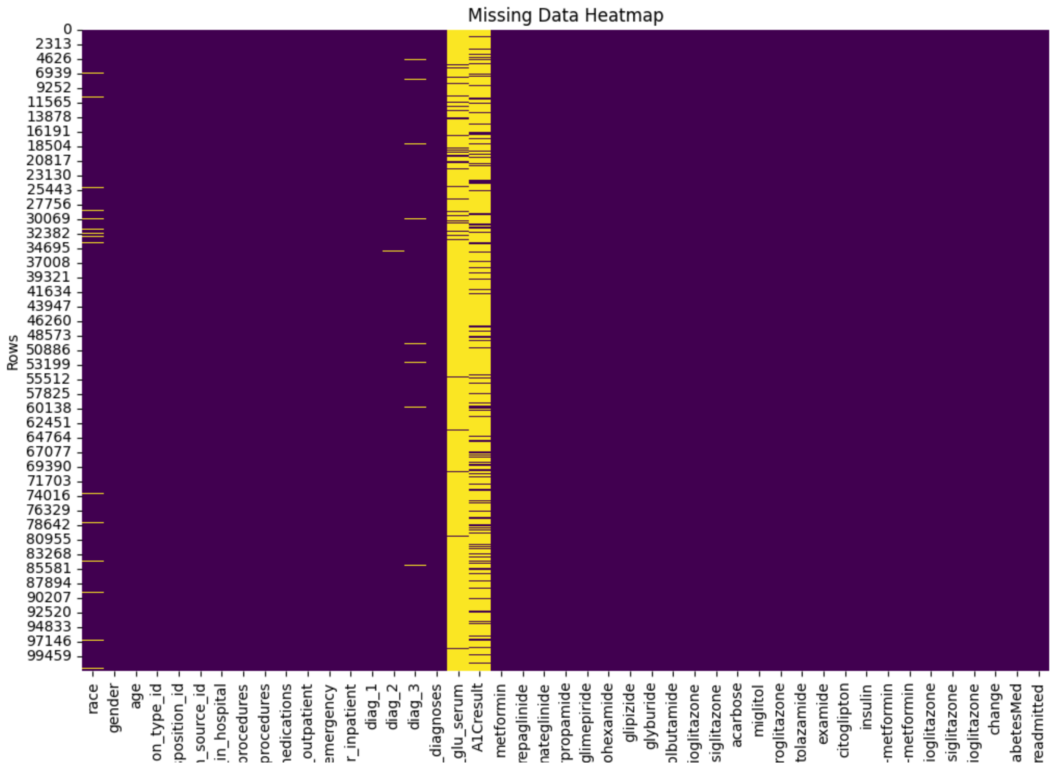


In this graph, the confusion matrices compare the performance of my three machine learning models: Random Forest, Gradient Boosting, and Stacking. Each matrix tests the models' ability to classify data by displaying the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

The Random Forest model got 1,795 true negatives, 1,887 true positives, 206 false positives, and 403 false negatives, having balanced performance but leaving room for improvement in reducing false negatives. Gradient Boosting achieved 1,754 true negatives and 2,011 true positives, with 247 false positives and 279 false negatives, having a better balance between sensitivity and precision. Meanwhile, the Stacking model has the highest number of true positives (2,033) and the lowest false negatives (257) but comes with 286 false positives. This indicates that Stacking prioritizes minimizing false negatives, which is beneficial in applications where missing a positive case has significant consequences.

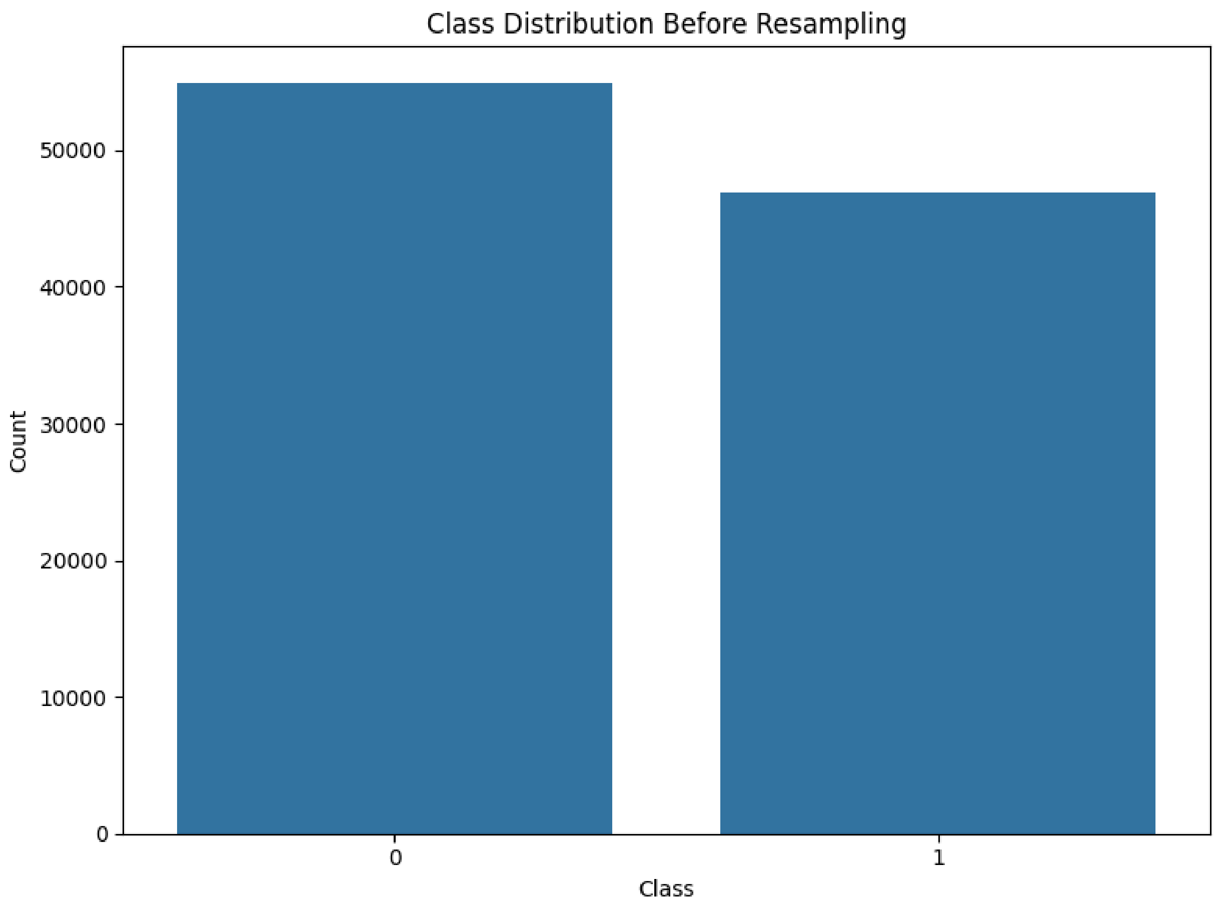
To achieve the highest accuracy and optimize classification performance, I combined these models using an ensemble approach. This strategy takes advantage of the strengths of each model, improving overall accuracy and reducing the weaknesses of individual models. By combining the predictions, the model effectively balances precision, recall, and overall classification accuracy. This ensemble approach makes sure there are always reliable predictions, which are required for applications such as healthcare, where both false negatives and false positives can have significant implications. These confusion matrices helped me find weaknesses within each model and specifically train them to learn from that part of the dataset more significantly than other parts of the dataset. Overall, this bettered the ensemble model and improved accuracies across the board.

Figure 2. Heatmap of missing data.



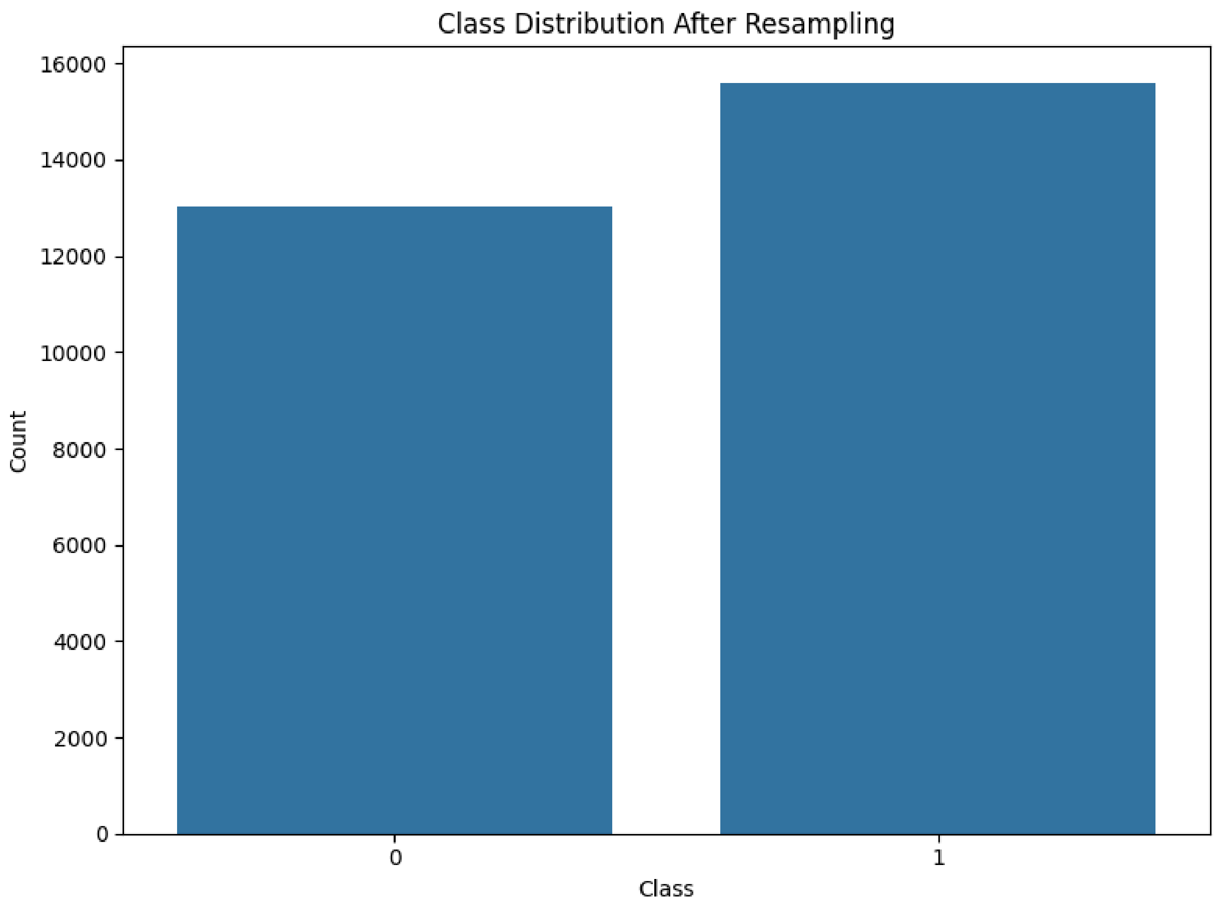
The missing data heatmap shows gaps and missing data within the dataset, with yellow indicating missing values and purple representing complete data. For this dataset, the heatmap identifies that features like 'max_glu_serum' and 'A1Cresult' contain many missing values, while others (e.g., 'weight' and 'medical_specialty') are excluded from the overall project. Despite its missing values, 'max_glu_serum' is still used in this project because it contains significant clinical information which directly correlates with diabetes and potential readmission risk. Similarly, 'A1Cresult' is a key indicator of long-term blood sugar control. Even with incomplete data, these features provide predictive power when imputed. Features like 'weight' and 'payer_code' are dropped due to lack of relevance to the clinical questions. The visualization shows the importance of addressing missing data. KNN imputation used in this analysis ensures that these critical features are not being simply thrown away but used as valuable predictive information. This missing data heatmap helped me analyse and decide which columns need to be prioritized when filling in missing values and coming up with an accurate prediction.

Figure 3. Dependent Variable Distribution Before Resampling.



This plot shows the imbalance in the responding variable, 'readmitted'. Non-readmitted cases significantly outnumber readmitted ones. This imbalance is a common challenge in healthcare datasets and can lead to biased models favouring the majority class. This distribution justifies the necessity of SMOTEENN, a technique combining oversampling and undersampling to address the imbalance. Balancing the classes ensures that the model learns patterns associated with both outcomes (readmitted and not readmitted), essential for accurate predictions in healthcare decision-making. This model was extremely important when deciding whether SMOTEENN was a valuable resource or rather an additional step. Since the difference in data shows nearly a 10000 case difference, I used SMOTEEN to balance out the dataset, increasing the accuracy by removing bias.

Figure 4. Dependent Variable Distribution After Resampling.



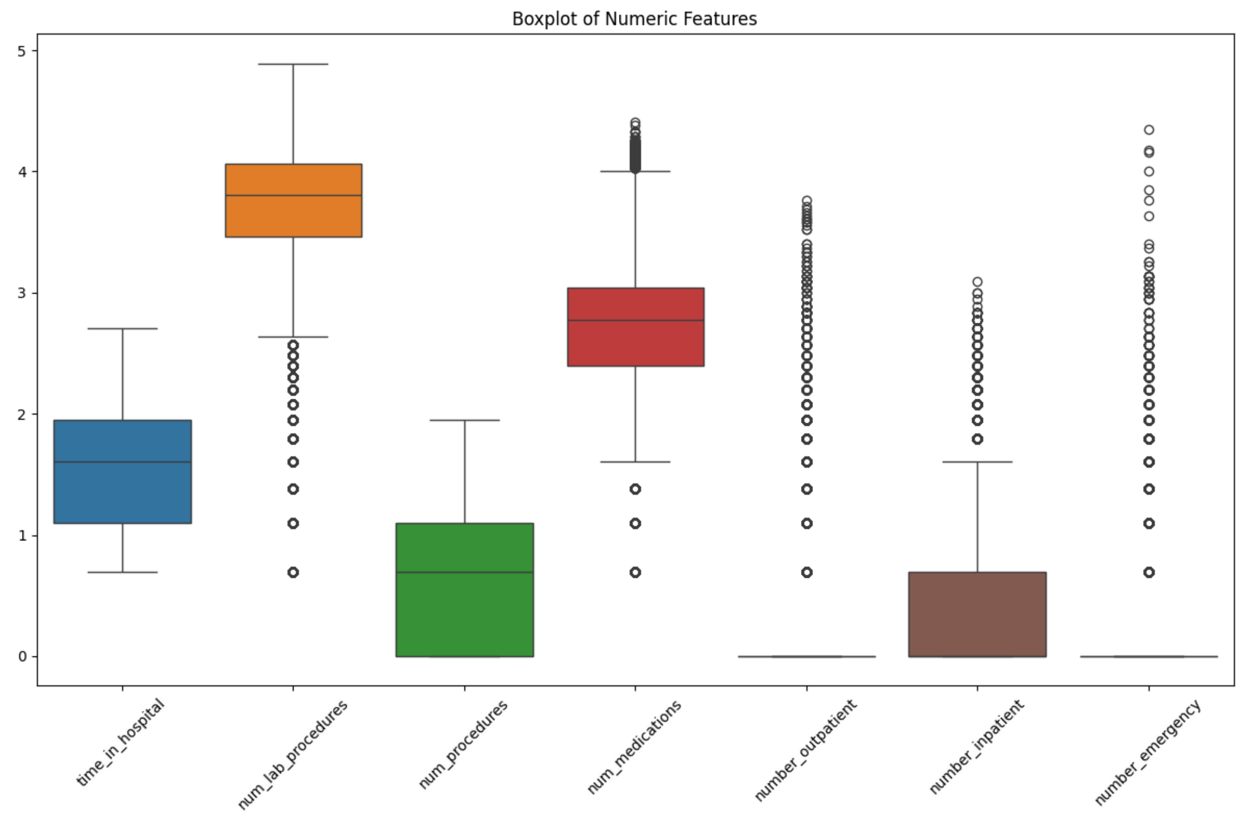
This graph displays the distribution of the target classes after applying SMOTEENN and duplicate deletion. Its importance is in confirming that resampling has successfully nearly balanced the dataset, providing a less biased dataset for training the machine learning model. A balanced dataset allows the model to learn from both classes, which enhances prediction accuracy for the minority class. This step improves the project by ensuring the model has a fair chance to detect patterns in both classes, resulting in a more reliable and accurate prediction model. Although the difference is still nearly 4000 cases, the resampling ensures that the model is not too biased toward the majority class. In the future, better balancing and data preprocessing could advance this project.

SMOTE works by generating synthetic data points for the minority class using the Euclidean distance formula to find k-nearest neighbors, then creating new samples between the current instance and its neighbors using the formula: $new = x + \lambda(x_{neighbor} - x)$, where λ is a random factor between 0 and 1, x is an instance from the minority class that you're trying to generate synthetic data for and $x_{neighbor}$ represents one of the nearest neighbors of x from the minority class (Chawla, Nitesh V., et al. 2002). SMOTEENN extends SMOTE by applying Edited Nearest

Neighbors to clean up the data, removing noisy or ambiguous points that do not align with the majority class of their neighbors.

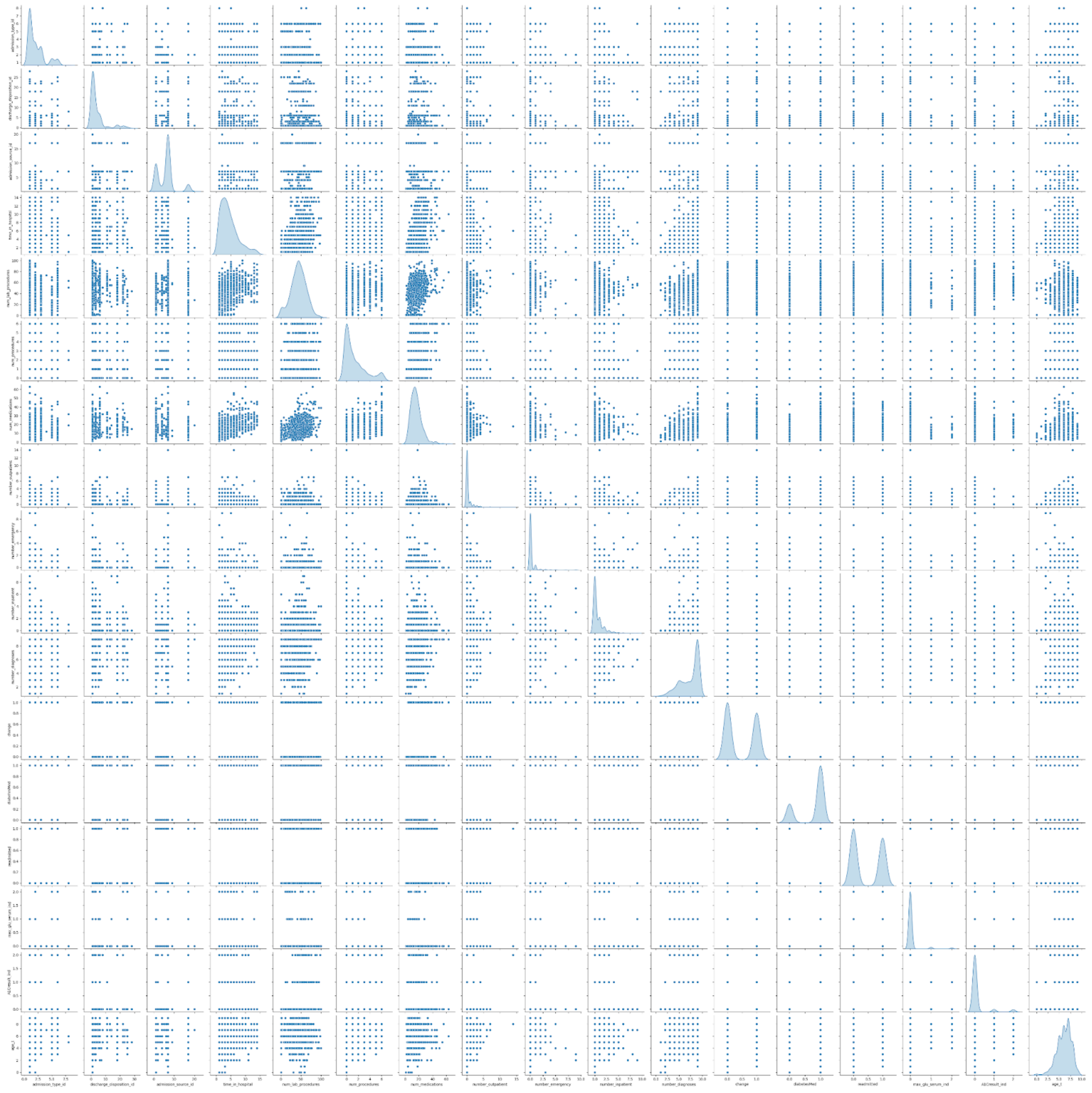
While SMOTEENN has some drawbacks, such as removing too many samples in my case, its benefits outweigh these issues (Batista, Gustavo E. A. P. A., et al. 2004). By balancing and cleaning the data, SMOTEENN reduces bias toward the majority class and helps the model recognize patterns in both classes more effectively. This ensures the model is trained on a clean and representative dataset, improving prediction reliability. As a result, SMOTEENN is a crucial tool for building a fair and robust machine learning model for this project.

Figure 5. Boxplot of Numeric Features.



The boxplot visualizes the spread and outliers of selected numeric features in the dataset, such as "time_in_hospital" or "num_medications." This graph is crucial for identifying potential outliers and understanding the variance of features, as outliers can significantly affect the performance of machine learning models. Shown by the circles on the plot, some data columns contain many more outliers than others. This boxplot guide helped me create a program which was only for reducing the impact of these outliers. They used data transformation techniques, like logarithmic scaling or imputation strategies, which help normalize the features to enhance the accuracy of the model. Addressing these outliers and variations is a requirement for improving model stability and performance.

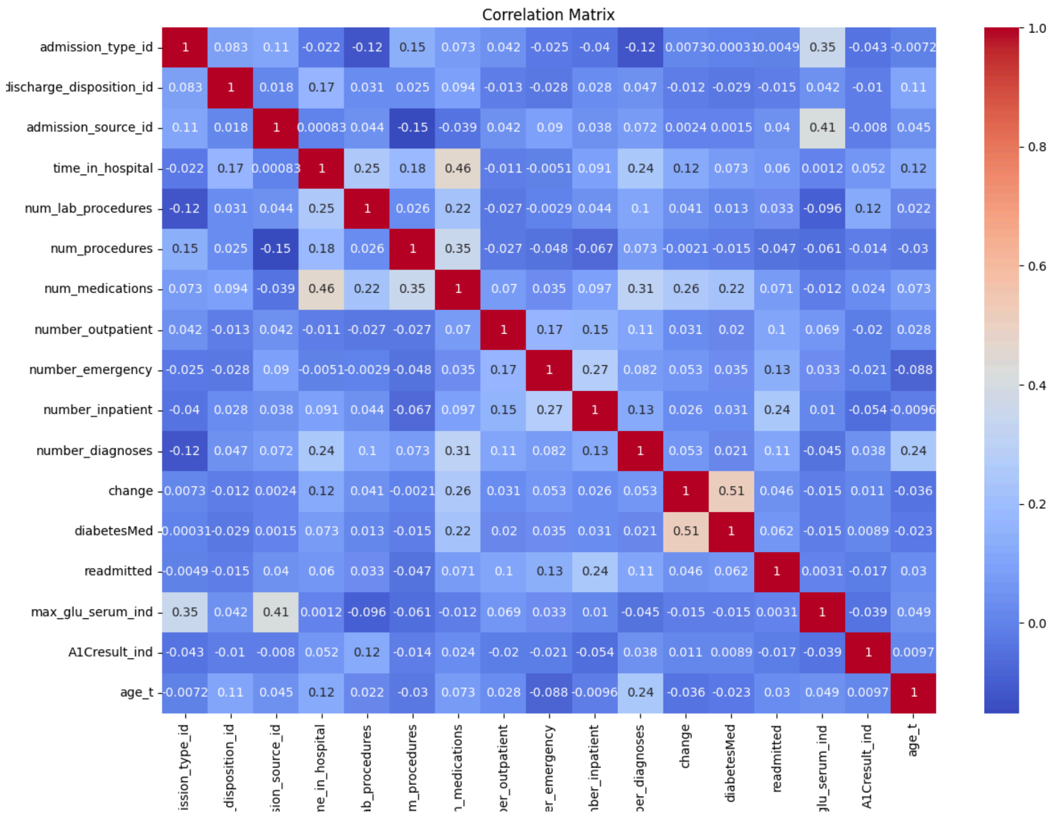
Figure 6. Pairplot Selected Features.



This pair plot shows the pairwise relationships between certain features, such as "time_in_hospital," "num_medications," and “number_inpatient.” The graph is important because it helps identify correlations and patterns between features that may be predictive of hospital readmissions. By finding relationships within the data, the pair plot helps inform the feature engineering and selection process. This step improves the project by refining the model’s input features, leading to better accuracy and efficiency.

Compared to a correlation matrix, the pairplot offers a more visual representation of the relationships between features, which can help identify not only linear correlations but also non-linear relationships. A correlation matrix only shows the strength and direction of linear associations, while a pairplot shows a more comprehensive view by displaying the distribution of each feature and how they interact with each other, including outliers and clustering.

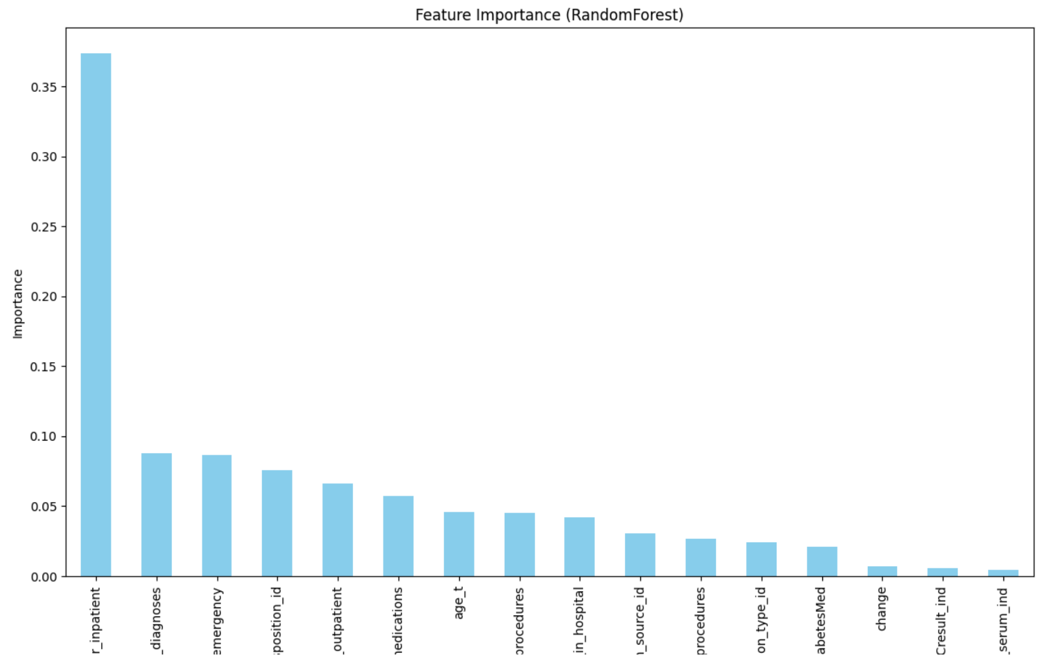
Figure 7. Correlation Matrix.



The correlation matrix uses a heatmap to show relationships between numerical features. Features with strong correlations (e.g., 'number_inpatient' and 'number_emergency') might introduce complicating model interpretation. Managing these correlations through dimensionality reduction or feature selection enhances model performance. The matrix also highlights weak correlations between some features and the target variable ('readmitted'), guiding decisions to prioritize or exclude features based on their predictive power. For example, weakly correlated features may contribute less value and could be deprioritizing the model. These could be used to find patterns with specific data points and values.

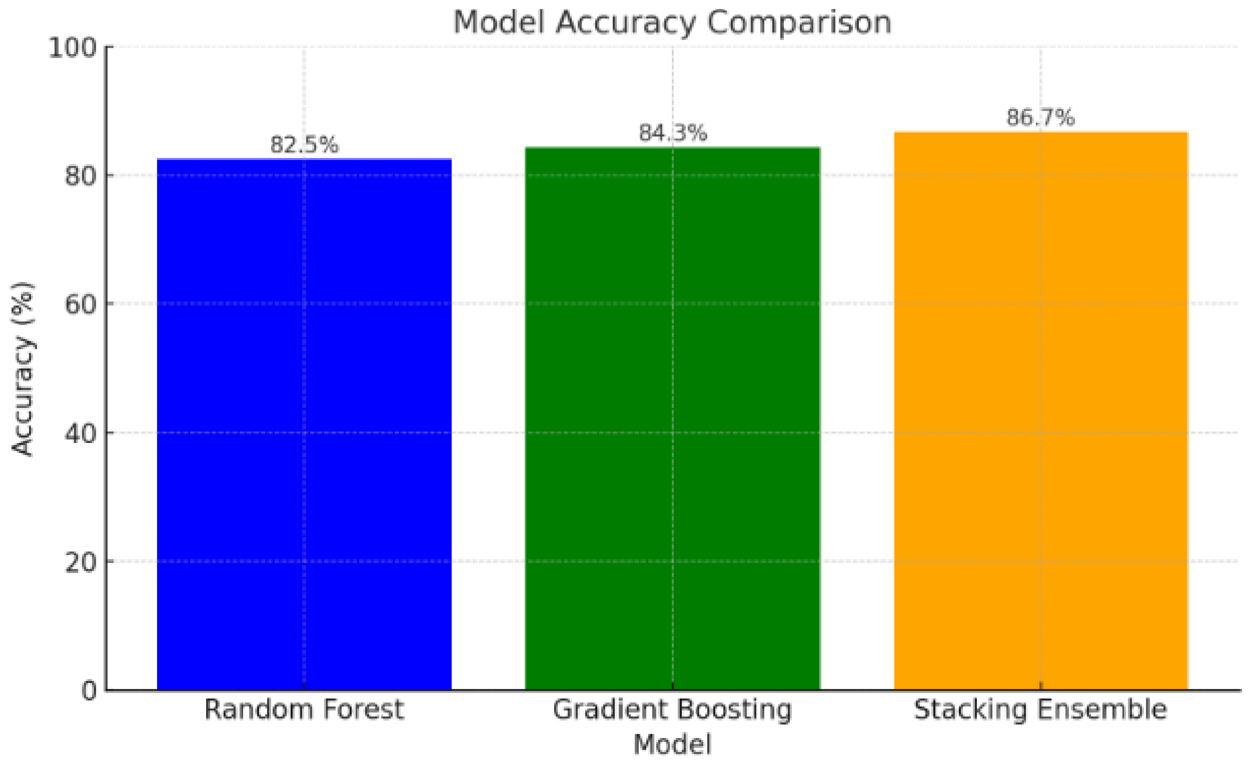
A correlation matrix can be better than a pairplot in certain situations because it provides a clear, compact numerical summary of the linear relationships between features. It displays the strength and direction of correlations in a concise table, making it easier to quickly identify strong or weak relationships. It allows for a more precise and efficient assessment of relationships, finding redundant features and ensuring the model is not biased by strong correlations.

Figure 8. Feature Importance.



The feature importance plot ranks variables based on their influence on predictions. Features such as 'number_inpatient', 'number_emergency', and 'number_medications' have high importance scores. For example, frequent hospitalizations and patient diagnoses are directly linked to patient outcomes and readmission risk. This ranking helps clinicians, and researchers focus on high-impact variables, allowing for highly targeted interventions for at-risk patients. Understanding the importance of features also helps refine the model to improve interpretability and effectiveness. My model utilized this as a way to understand how important certain variables were in deciding if the patient had a likelihood of being readmitted.

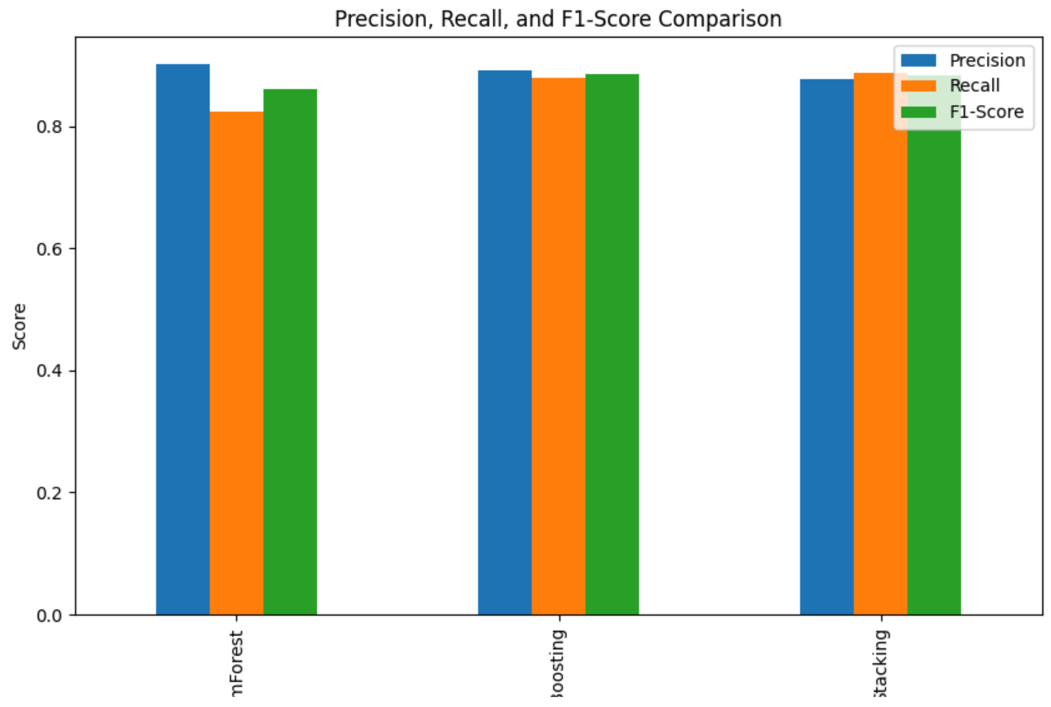
Figure 9. Model Accuracies.



This graph is an extremely important component of my analysis on predicting hospital readmissions in diabetic patients, as it compares the prediction accuracy of the three models I implemented—Random Forest, Gradient Boosting, and Stacking Ensemble. The Stacking Ensemble achieves the highest accuracy (86.7%), outperforming Gradient Boosting (84.3%) and Random Forest (82.5%), showing the strong accuracy of ensemble methods. This result shows the effectiveness of stacking in combining the strengths of multiple models to improve the accuracy of a machine learning model.

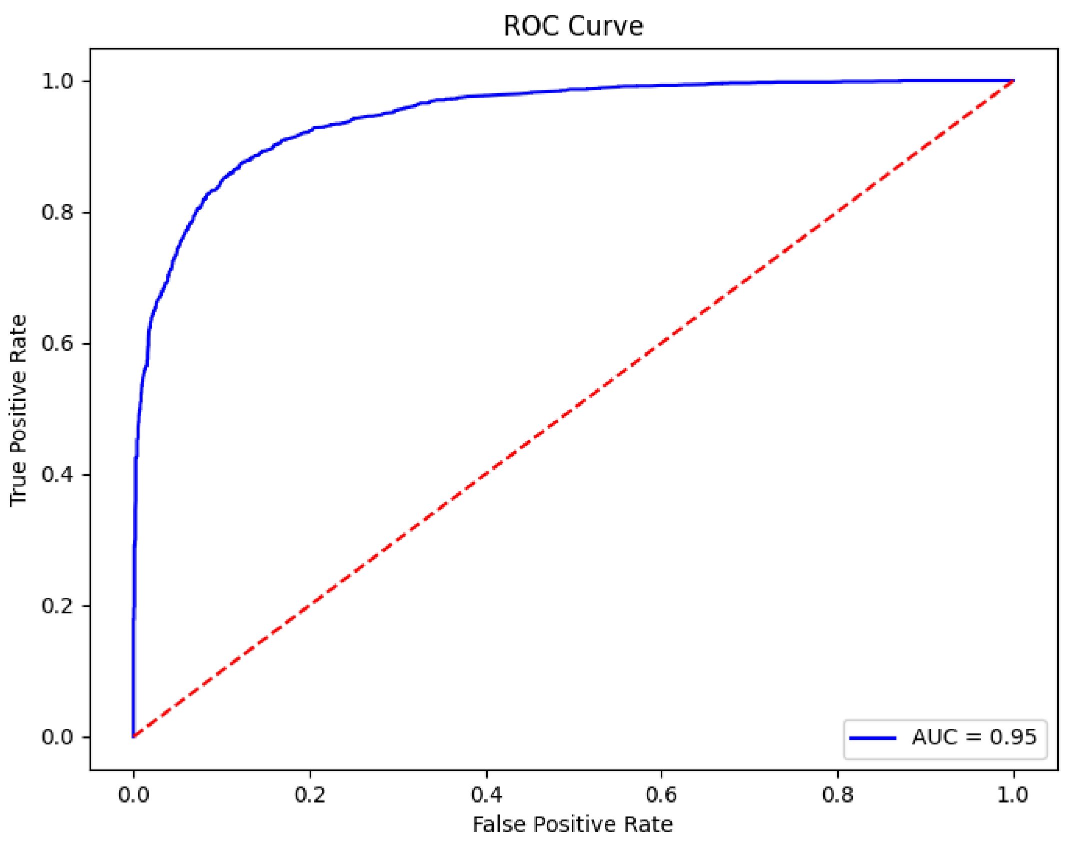
This graph is more important than a simple performance comparison; it shows the amazing role of advanced ensemble techniques in healthcare applications. Accurate predictions of hospital readmissions can have a huge impact on patient care. By using the ensemble models like the Stacking Ensemble I used in this project, the complexities of readmission prediction become reduced.

Figure 10. Precision, Recall and F1-Scores.



The precision, recall, and F1-score triple bar graph shows the performance of three classification models—Random Forest, Gradient Boosting, and Stacking—using key metrics from the confusion matrix (Powers, David M. W., 2011). Precision, calculated as *Precision = TP/(TP+FP)*, measures the proportion of positive predictions that are actually correct. A higher precision indicates the model effectively minimizes false positives, which is crucial when incorrect positive predictions have significant consequences. Recall or sensitivity, calculated as *Recall* = *TP/(TP+FN)*, assesses the model's ability to identify all actual positive cases. Doing this focuses on minimizing false negatives, which is extremely important in healthcare fields, where missing a positive case—such as a patient readmission—can lead to dangerous outcomes. The F1-score, calculated as *F1 = 2(Precision*Recal)/(Precision+Recall)*, provides an average of precision and recall, balancing the two and evaluating the model's performance, especially when dealing with imbalanced datasets (Saito, Takafumi, and Makoto Rehmsmeier, 2015).

Figure 11. ROC Curve.



This ROC (Receiver Operating Characteristic) graph shows how well my machine learning model can identify positive cases correctly. In my project, the ROC is correctly identifying patient readmission within the time span of 30 days. It graphs the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x axis (Hastie, Trevor, et al 2009). The TPR shows the ratio of correct positive cases found by the model, found through the equation for sensitivity, *sensitivity = a/(a+b)*, where a is the count of true positives and b is the count of false negatives. FPR is found by *1-specificity*, *specificity = d/(b+d)*, where d is the number of true negatives (correctly identified negative cases) and b is the number of false positives (negative cases incorrectly identified as positive). The FPR shows the ratio of negative cases incorrectly thought to be positives, calculated as *False Positive Rate = c/(d+c)*, where c is the count of false positives and d is the count of true negatives (Fawcett, Tom, 2006).

The Area Under the ROC Curve is also created in the graph above. A higher AUC means better model performance, showing that it can distinguish between positive and negative classes.

Specifically, an AUC closer to 1 means a model has excellent distinguishing ability. Throughout my time working on this project, the AUC often fluctuated below 0.8 because the TPR wasn't strong enough. This led to many errors when predicting whether my patient was at risk of rehospitalization. To fix this, I constantly resampled the data to find a point where false positives were minimized.

Conclusion

In conclusion, this project shows the importance of predicting hospital readmissions in diabetic patients to improve patient care and reduce healthcare costs. By using different Python packages related to machine learning, handling class imbalances and graphing, we can create a strong model that can detect patterns a human can't simply see. The development of a user-friendly tool on Replit demonstrates the application of this predictive model, making it accessible to healthcare providers for early intervention and personalized care planning. This project can advance the healthcare industry and save countless lives across healthcare systems across the world.

Next Steps

In the future, this model could include additional data sources, such as socioeconomic factors, to enhance accuracy. Testing the model across diverse healthcare settings and patient populations will ensure its generalizability. Additionally, integrating the tool into real-world healthcare systems and conducting user feedback studies will help optimize its functionality, contributing to better outcomes for diabetic patients and more efficient resource utilization in healthcare.