VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum - 590018

A Project report on

"Patient Readmission Prediction Based On Explainable AI "

submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

by

1CR19IS008 Adithyan R

1CR19IS023 C Tarun Teja

1CR19IS029 Chiragraju S

Under the Guidance of

Prof. Suguna Devi

Associate Professor

Department of ISE, CMRIT, Bengaluru

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

2022-23

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

Certificate

This is to Certified that the project work entitled **"Patient Readmission Prediction Based On Explainable AI"** carried out by **Adithyan R (1CR19IS008), C Tarun Teja (1CR19IS023), and Chiragraju S (1CR19IS029)** in partial fulfillment for the award of Bachelor of Engineering in **Information Science & Engineering** of the Visveswaraiah Technological University, Belgaum during the year **2022-23**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of Guide	Signature of HoD	Signature of Principal
Prof. Suguna Devi	**Dr. Farida Begam**	**Dr. Sanjay Jain**
Associate Professor	Professor & Head	Principal
Department of ISE	Department of ISE	CMRIT ,
CMRIT	CMRIT	Bengaluru-37

External Viva

	Name of the Examiners	Institution	Signature with Date
1.			
2.			

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

Declaration

We, **Adithyan R (1CR19IS008), C Tarun Teja (1CR19IS023), and Chiragraju S (1CR19IS029)** bonafide students of **CMR Institute of Technology**, Bangalore, hereby declare that the dissertation entitled, **"Patient Readmission Prediction Based On Explainable AI"** has been carried out by us under the guidance of **Prof. Suguna Devi**, Assoc. Professor, CMRIT, Bangalore, in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering, of the Visvesvaraya Technological University, Belgaum during the academic year 2018-2019. The work done in this dissertation report is original and it has not been submitted for any other degree in any university.

Adithyan R (1CR19IS008)
C Tarun Teja (1CR19IS023)
Chiragraju S(1CR19IS029)

Acknowledgement

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We would like to thank Dr. Sanjay Jain, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

We would like to express our gratitude towards Dr. Farida Begam, Professor and HOD, Department of Information Science & Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our internal guide Prof. Suguna Devi, Assoc. Professor, Department of Information Science & Engineering, CMRIT, Bangalore, for their valuable guidance throughout the tenure of this project work.

We would like to thank all the faculty members who have always been very cooperative and generous. Conclusively, we also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

Adithyan R
C Tarun Teja
Chiragraju S

Abstract

Addressing the issue of readmission to hospitals is a significant concern for patients, healthcare providers, and payers, as it can lead to increased healthcare costs and negative patient outcomes. To tackle this problem, our project focuses on exploring the application of deep learning and explainable artificial intelligence (AI) techniques in predicting readmission risk. We aim to review the current state of the field, examining the various approaches using deep learning and explainable AI for readmission prediction. Additionally, we will evaluate the performance of these methods by considering relevant accuracy measures. Recognizing the importance of explainability in AI models for healthcare applications, especially in readmission prediction, we will discuss its significance in terms of clinical interpretability, compliance with regulatory requirements, addressing fairness and bias issues, and establishing trust and transparency in the prediction process.

Table of Contents

List of Figures

Chapter 1

Preamble

1.1 Introduction

Patient readmission poses a significant challenge for hospitals and healthcare systems worldwide, as it not only adversely affects patient outcomes but also leads to a substantial increase in healthcare costs. To mitigate this problem, it is crucial to predict which patients are at risk of readmission. In recent years, machine learning and artificial intelligence (AI) have shown promise in this area. However, the opacity and inscrutability of these models create a sense of uncertainty and unreliability, as their decision-making process remains a black box.

In this project, we present a patient readmission prediction system that uses explainable AI to explain predicition, specifically designed to provide transparent and comprehensible predictions. By offering this level of transparency, our system aims to empower healthcare professionals in their decision-making process and enhance their understanding of patients' needs.

Our patient readmission prediction system offers valuable insights to healthcare professionals regarding the factors that contribute to readmission risks. By gaining a better understanding of these factors, healthcare professionals can make well-informed decisions related to discharge planning, post-discharge follow-up, and resource allocation, ultimately leading to improved patient outcomes.

To achieve explainability, our system incorporates techniques such as LIME and SHAP. These techniques provide clear justifications alongside predictions, enhancing trust and acceptance of AI-driven decision support tools in the healthcare industry. The transparency and interpretability of our system promote a collaborative approach between healthcare professionals and AI models, ensuring that decisions are based on a thorough comprehension of the underlying factors that impact patient readmission risks.

1.2 Existing System details updated

The existing patient readmission prediction system encounters challenges due to its lack of transparency and interpretability. Although it utilizes machine learning and artificial intelligence techniques, it falls short in providing comprehensible explanations for its predictions. This opaqueness undermines the system's reliability and diminishes the trust healthcare professionals can place in its outputs, limiting their ability to effectively utilize them.

The current system likely employs various machine learning algorithms, such as deep learning models or ensemble methods, to analyze patient data and generate predictions regarding readmission risks. However, without adequate mechanisms for explainability, it fails to offer insights into the reasoning behind these predictions. The absence of transparency hampers healthcare professionals' understanding of the factors influencing readmission risks and restricts their capacity to proactively address those risks.

Moreover, the limitations of the current system impede collaboration and shared decision-making between healthcare professionals and the AI model. In the absence of clear explanations, healthcare professionals may hesitate to solely rely on the system's predictions, resulting in suboptimal patient care and increased readmission rates.

1.2.1 Drawbacks

In healthcare, preventing patient readmissions is a significant challenge as it involves patients needing to be readmitted shortly after discharge due to worsening conditions or new health issues. Traditionally, healthcare professionals have relied on their clinical expertise and judgment to identify patients who may be at risk of readmission. However, this approach has limitations, and there is increasing recognition that machine learning and AI can offer valuable insights in predicting readmission risks. By analyzing extensive patient datasets, machine learning models can uncover patterns and trends that may not be readily apparent to humans, leading to more accurate and reliable predictions.

Despite the benefits, the black-box nature of these models presents a challenge in understanding and explaining the factors contributing to their predictions. The lack of transparency can hinder the adoption of machine learning and AI in healthcare, as it becomes difficult for healthcare professionals to trust and depend on these systems.

1.3 Proposed System

In this work, we present a patient readmission prediction system that uses explainable AI(XAI) to explain prediction to non-technical medical experts, which is designed to overcome these limitations. By using interpretable machine learning models and techniques, our system is designed to provide transparent and understandable predictions, allowing healthcare professionals to better understand and address the needs of their patients. Using explainable AI, our system aims to support healthcare professionals with their decision making and help them to better understand and address the needs of their patients.

1.4 Plan of Implementation

1. Obtain data from 130 US hospitals [6] in a zipped format.
2. Extract and load the data onto a Python kernel for analysis.
3. Perform data understanding, cleaning, and preprocessing.
4. Appropriately select relevant features, manipulate them, remove outliers, and normalize the data as per the model requirements.
5. Train and test several machine learning models such as Gradient Boost,Decision Tree, Decision Forest, and Logistic Regression.
6. Assess the models' performance by utilizing metrics such as precision, F1 score, recall, accuracy, and AUC.
7. Introduce and examine using an Explainable AI model.
8. Present the information in a format that is easily understandable for humans and show how a decision was made.

1.5 Problem Statement

This project addresses the lack of transparency and interpretability in existing machine learning models for patient readmission prediction, hindering effective decision-making by healthcare professionals. We are to develop an explainable AI system that provides transparent and understandable predictions, enabling informed interventions and improved patient outcomes.

1.6 Objective of the Project

- Improve readmission risk predictions for higher accuracy and reliability: By using advanced ML techniques and interpretable models, the system aims to provide more accurate and reliable predictions of readmission risk, which can help healthcare professionals to identify and intervene with high-risk patients.
- To increase transparency and explainability: In order to enhance transparency and explainability, the incorporation of explainable AI techniques enables the system to offer clear and comprehensible justifications for its predictions. This capability empowers healthcare professionals to gain a deeper understanding of the underlying factors influencing readmission risk, thereby facilitating informed decision-making.
- To support clinical decision-making: The system aims to provide real-time, actionable insights to healthcare professionals, helping them to identify and address the needs of patients at risk of readmission.

Chapter 2

Literature Survey

- **Hu, Y. and Sokolova, M., 2020. Explainable Multi-class Classification of Medical Data.**
  - This paper presents a multi-class classification method for a large medical dataset.
  - The authors discuss various techniques such as knowledge-based feature engineering, dataset balancing, parameter tuning, and model selection.
  - Six algorithms are evaluated: Logistic Regression, Gradient Boosting, Decision Trees, Naive Bayes, Random Forest, and Support Vector Machine.
  - Their findings indicate that Gradient Boosting and Random Forest algorithms exhibited superior performance in terms of three-class classification accuracy, surpassing other algorithms.
- **Tjoa, E. and Guan, C., 2020. A survey on explainable artificial intelligence (XAI): Toward medical XAI. IEEE Transactions on Neural Networks and Learning Systems, 32(11), pp.4793-4813.**
  - This study examines the application of Explainable AI (XAI) in the healthcare sector, aiming to enhance transparency and instill trust in AI/ML applications.
  - The research concentrates on various data sources, including images, omics data, and text, with the objective of promoting the integration of AI/ML in the medical field.

- **Ashfaq, A., Sant'Anna, A., Lingman, M., and Nowaczyk, S., 2019. Readmission prediction using deep learning on electronic health records. Journal of Biomedical Informatics, 97, p.103256.**
  - This paper offers a comprehensive examination of the interpretability and explainability aspects related to machine learning (ML) algorithms.
  - It categorizes different approaches to interpretability, including mathematical formalization, visual explanations, and task performance improvement with explanations.
  - The survey applies these categories to the medical field.
- **Moradi, M. and Samwald, M., 2021. Post-hoc explanation of black-box classifiers using confident itemsets. Expert Systems with Applications, 165, p.113941.**
  - The authors propose Counterfactual Explanations (CIE) for explaining predictions made by black-box classifiers.
  - These itemsets are used to create concise explanations for individual instances and class-wise explanations through optimization of fidelity, interpretability, and coverage objectives.
  - CIE is a post-hoc and model-agnostic approach that categorizes the input and extracts itemsets from features highly associated with a class label.

Chapter 3

Theoretical Background

3.1 Patient Readmission Prediction

Patient readmission refers to the situation where a patient is readmitted to a hospital again within a specific period(within 30 days according to CMS) after their initial discharge. It is a critical challenge faced by healthcare systems worldwide as it adversely affects patient outcomes and increases healthcare costs. Predicting which patients are at risk of readmission is crucial for healthcare professionals to provide timely interventions and improve patient care.

3.2 Machine Learning Models

ML models and AI offer significant potential in the prediction of patient readmission risk. By utilizing historical patient data encompassing demographics, medical history, and clinical variables, these models can identify patterns and generate accurate predictions. Diverse machine learning algorithms, such as Gradient Boosting, Random Forests, Decision Trees, and Logistic Regression, can be applied to analyze extensive datasets and extract meaningful patterns for predicting the probability of patient readmission.

3.3 Explainable AI

Explainable AI, also referred to as Interpretable AI, is dedicated to developing machine learning models that offer explanations for their predictions. In the specific context of patient readmission prediction, interpretability plays a crucial role in comprehending the factors that contribute to the risk of readmission. This interpretability enables healthcare professionals to have trust in and understand the predictions made by these models. Through the application of interpretable AI techniques like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations), it becomes feasible to generate clear and understandable explanations for individual predictions.

3.4 LIME

LIME (Local Interpretable Model-agnostic Explanations) is an artificial intelligence technique designed to explain predictions made by any black-box machine learning model. It achieves this by constructing an interpretable model around the specific prediction and approximating the behavior of the black-box model in the local vicinity of that prediction. LIME produces feature importance weights, indicating the contribution of each input feature to the model's prediction for a given instance. These explanations serve as valuable insights for healthcare professionals, enabling them to comprehend the reasons behind the prediction of readmission risk for individual patients.

3.5 SHAP

SHAP (SHapley Additive exPlanations) is another interpretable AI technique that assigns a value, called Shapley value, to each feature in the predicted value. It provides a unified framework based on game theory to explain the output of any machine learning model. SHAP values capture the mean contribution of each feature across all possible feature combinations. By calculating SHAP values for patient readmission predictions, healthcare professionals gain insights into the relative importance of different features in the model's decision-making process.

Chapter 4

System Requirements Specification

4.1 Functional Requirement

1. **Data Acquisition**: The system should have the capability to securely and efficiently retrieve patient data from electronic health records (EHR) or other relevant sources.

2. **Data Preprocessing**: The system should perform necessary data cleaning, transformation, and preprocessing techniques to ensure the quality and compatibility of the input data for the machine learning models.

3. **Feature Selection and Engineering**:The system should offer the capability to choose pertinent features from the input data and provide opportunities for applying feature engineering techniques to improve the predictive capability of the models.

4. **Model Training and Evaluation**: The system should conduct training on multiple machine learning models, including Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, and Gradient Boosting, using the preprocessed data. Additionally, it should assess the performance of these models using suitable evaluation metrics such as accuracy, precision, recall, F1 score, and area under the ROC curve.

5. **Interpretability Techniques Integration**: The system should integrate interpretability techniques like LIME or SHAP to generate explanations for the predictions made by the machine learning models. It should provide feature importance rankings, visualizations, or textual explanations to help understand the factors contributing to the risk of patient readmission.

6. **Real-time Prediction and Explanations**: The system should allow healthcare professionals to input patient information in real-time and generate predictions for the likelihood of readmission. It should also provide immediate explanations for these predictions, aiding clinicians in making informed decisions.

7. **User Interface**: The system should have an intuitive and user-friendly interface, preferably developed using Streamlit or other web application frameworks, to facilitate easy interaction and input of patient data. The interface should display prediction results, explanations, and relevant visualizations in a clear and understandable manner.

4.2 Non Functional Requirements

- **Reliability**: The system should be reliable, with a high uptime and a low rate of errors or failures.
- **Flexibility**: The system should be flexible, able to adapt to changing requirements or conditions without requiring major rework.
- **Handling Complex Problems**: The system should possess the capability to address intricate, multifaceted problems and offer explanations that consider the interplay and compromises among various factors.
- **Real-time Processing**: The system should be able to operate in real-time, providing explanations as decisions are made.
- **Cost Considerations**: A model that has been designed to consider cost factors and utilizes a combination of features generated both by human and machine processes.
- **Efficiency**: The system should be efficient, with a fast response time and low resource usage.

4.3 System Configuration

- **Processor**: A computer processor with at least 4 cores and a frequency of 2.0 GHz or higher.
- **Memory**: At least 8 GB of RAM to allow for efficient data processing and model training.
- **Storage**: A solid-state drive with at least 128 GB of storage to store data and machine learning models.
- **Operating System**: A 64-bit operating system such as Windows, macOS, or Linux to support machine learning frameworks and libraries.
- **Software**: A machine learning framework like scikit-learn can be utilized to construct and train machine learning models, while incorporating the SHAP (SHapley Additive exPlanations) framework for enhanced interpretability.
- **Development Environment**: A programming language like Python and an Integrated Development Environment (IDE) like Jupyter or PyCharm to write and run the code with.

Chapter 5

System Analysis

To solve a problem, one must first analyse the situation. By performing a system analysis, we can identify and characterise the challenges at hand, also define and assess potential solutions. It is a style of approaching the business and the issues it faces. The system utilizes various technologies to enhance problem-solving capabilities. In system analysis, which establishes the goals for design and development, feasibility studies play a crucial role.

5.1 Feasibility Study

5.1.1 Technical Feasibility

- All the libraries and technology used is readily available and is free. Setting up the application requires fair bit of technical knowledge
- After Evaluating the compatibility of the selected machine learning models, explainability techniques (e.g., SHAP, LIME), and streamlit framework with the project requirements. It seems SHAP and LIME work well with some models. Using a workaround they can be represented as a web application.
- The proposed web application's seamless integration with hospitals is highly feasible, leveraging its web-based nature and compatibility with existing healthcare systems.The system seamlessly integrates with electronic health records (EHRs), patient management systems, and decision support tools, ensuring effortless compatibility without requiring extensive modifications.

5.1.2 Economic Feasibility

The purpose of this study is to evaluate the financial implications of implementing the system within the organization. With limited resources allocated to research and development, it is crucial to justify the expenses involved. By leveraging freely available technologies and developing the readmission prediction system within the allocated budget, hospitals in a healthcare setting can effectively identify high-risk patients and implement timely interventions. This has the potential to mitigate healthcare costs associated with readmissions.

5.1.3 Social Feasibility

The project has the potential to positively impact society by improving patient outcomes, optimizing resource allocation, reducing healthcare costs, personalizing care, advancing interpretable AI in healthcare, and contributing to medical knowledge generation.While the project offers numerous benefits, there are potential risks to consider. These may include privacy and security concerns related to handling sensitive patient data, the need for robust data governance practices, potential biases or inaccuracies in the predictions, challenges in ensuring the reliability and trustworthiness of the AI models, and the need for proper training and understanding of the system by healthcare professionals. Additionally, there may be resistance or skepticism from some stakeholders in adopting AI technologies in healthcare. To ensure the mitigation of these risks, it is crucial to implement suitable safeguards, adhere to relevant regulations, and continuously monitor and evaluate the system. This proactive approach helps maintain the integrity and effectiveness of the system while safeguarding against potential pitfalls or adverse consequences.

5.2 Analysis

5.2.1 Performance Analysis

In the analysis process, various techniques are employed to evaluate the performance of the models, including metrics such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics serve as quantitative measures of the model's accuracy, its ability to correctly identify positive cases, its tendency to generate false positives, and the balance between precision and recall. The selection of models is based on whether they meet the specific requirements and objectives of the project, ensuring that they align with the intended goals and criteria.

5.2.2 Technical Analysis

1. **System Improvements:** The proposed system aims to bring positive changes, such as increased efficiency and improved customer service, compared to existing systems.

2. **Technical Skills:** The platforms and tools used, including machine learning frameworks, SHAP, LIME, and Streamlit, are widely used in many industries. Therefore, there is a readily available pool of skilled workforce with expertise in these technologies.

3. **Acceptability:** The system's structure and design are carefully crafted to ensure user-friendliness and ease of adoption. User feedback and requirements have been considered to ensure that the system is feasible from the user's perspective.

5.2.3 Economical Analysis

Implementing the developed readmission prediction system in a healthcare setting can have significant financial implications. By accurately identifying high-risk patients and enabling timely interventions, the system has the potential to reduce healthcare costs associated with readmissions. This leads to cost savings through optimized resource allocation, improved patient outcomes, and reduced hospitalizations. Conducting a thorough financial analysis helps determine the feasibility of investing resources into the system's research, development, and implementation, considering the potential financial benefits it offers to the organization.

Chapter 6

System Design

6.1 System Development Methodology

The iterative and incremental development methodology is highly suitable for developing a machine learning model in your project. This approach involves repetitive cycles of analysis, design, implementation, testing, and maintenance. It allows for flexibility and adaptation at each stage of the project lifecycle, enabling you to make iterative improvements and adjustments based on the requirements and insights.

- Requirements Analysis: This stage involves gathering and understanding the specific requirements of the machine learning model for my project. It includes identifying the key objectives, data sources, and desired outcomes.
- Design: A detailed design is created for the machine learning model, considering algorithm selection, data preprocessing techniques, feature engineering methods, and model architecture. This design ensures the model meets the project's requirements effectively.
- Implementation: In this stage, the design is translated into actual code. The machine learning model is developed using a programming language such as Python, and the necessary libraries and frameworks are utilized to implement the model.
- Testing: The developed machine learning model undergoes thorough testing to ensure its accuracy, reliability, and performance. Different test cases are executed, and the model's outputs are compared with the expected results.
- Evaluation and Refinement: The performance of the model is evaluated against predefined metrics and criteria. If necessary, the model is refined and adjusted based on the evaluation results, which may involve fine-tuning the hyperparameters, optimizing the model's performance, and addressing any identified issues.

6.2 Design using UML

Using both static and dynamic UML diagrams, the process of designing a UML diagram explains how the system's processes interact with its components and how the components communicate with one another. It has become increasingly challenging to create and maintain high-quality machine learning models in a timely manner in the ever-evolving field of AI development. The Unified Modeling Language (UML), which was created in response to this challenge and there was a need for a universal modeling language for object-oriented systems, serves as the blueprint for the information industry. It is a technique for comprehensively outlining the architecture of the system, enabling the construction and maintenance of a system that is easier to manage and more adaptable to changes in requirements.

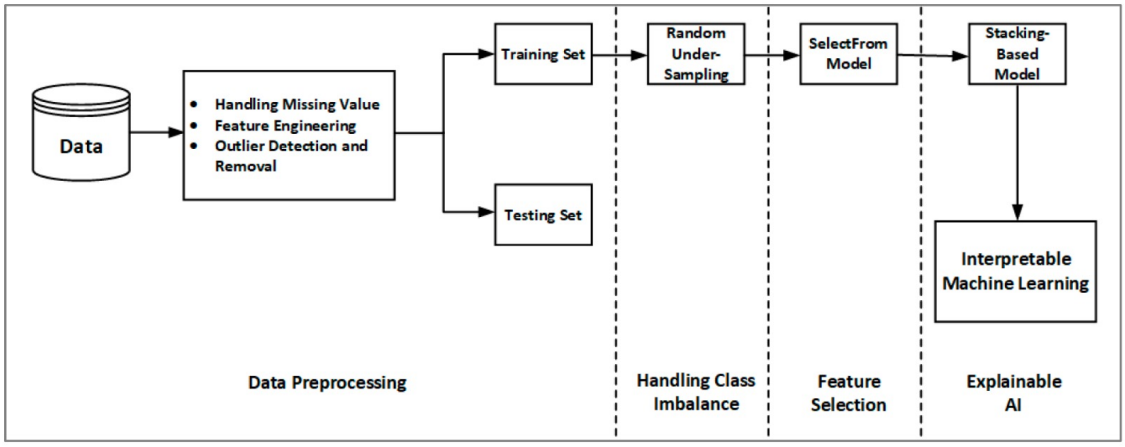

Figure 6.1: UML Diagram

6.3 Data Flow Diagram

A data flow diagram is a visual depiction of how data moves within your system, illustrating its flow from source to sink and highlighting any transformations that occur along the way. This diagram enables us to comprehend the functionalities of each system component and provides valuable insights to outsiders who seek to understand the existing system. By representing data movement and transformations visually, a data flow diagram serves as a crucial tool for comprehending and communicating the inner workings of the system.

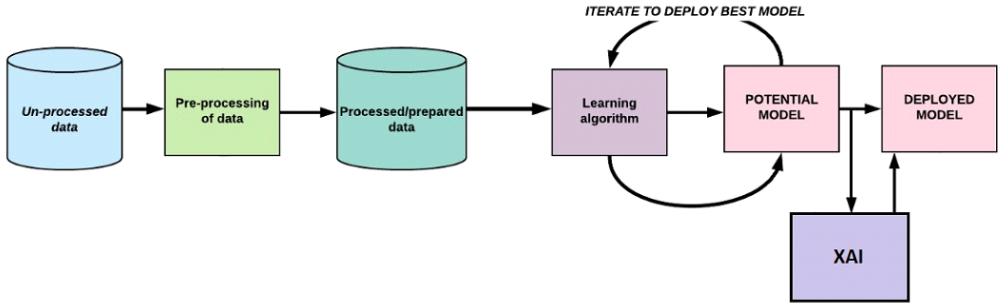

Figure 6.2: Data flow diagram

6.4 Component Diagram

The component diagram of a system outlines the interconnectedness of its components and the flow of data between them. It reflects the structure, functions, and relationships of the designed system and how it interacts with other systems and the external environment.

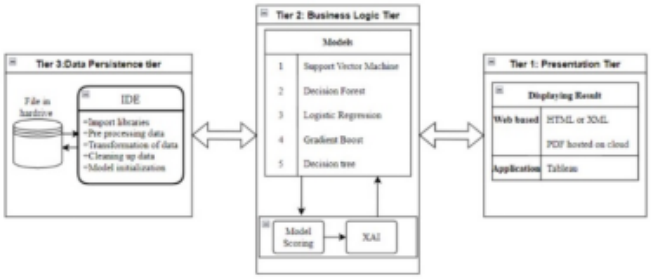

Figure 6.3: Component Diagram

6.5 Use Case Diagram

An interaction between external entities and the system under study that is goal-oriented is defined by a use case. The actors in the system are the external entities that communicate with it. The use case diagram can be used to graphically represent a group of use cases that define all system features in great detail.

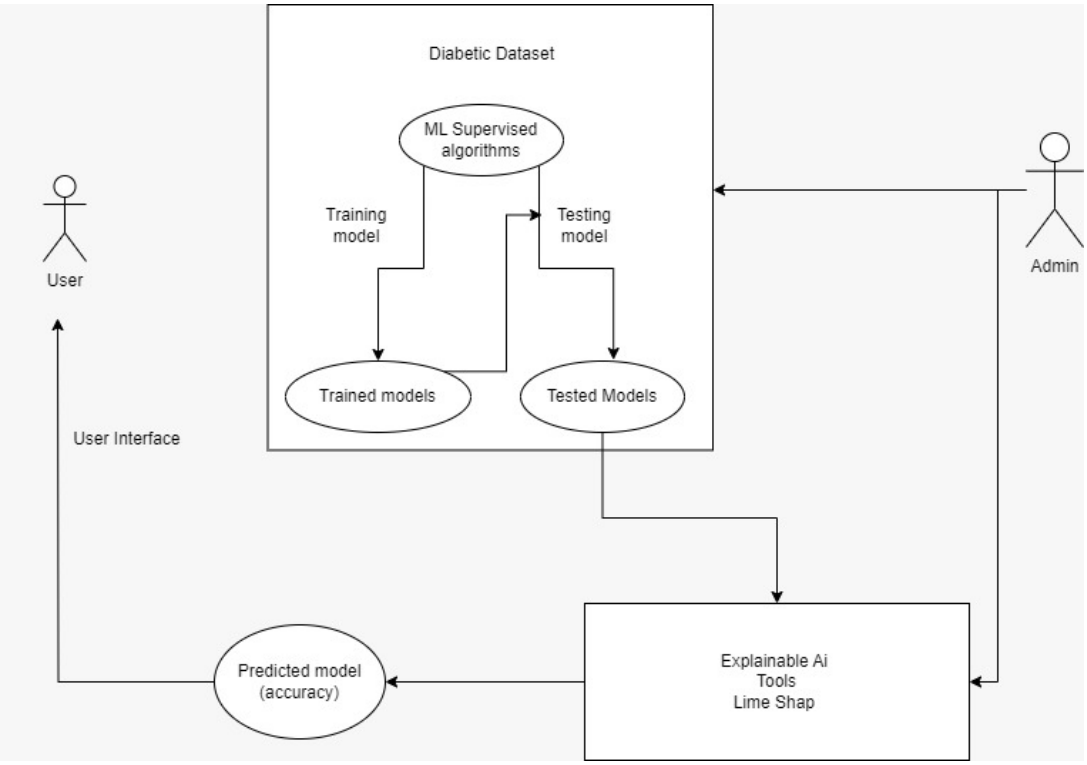

Figure 6.4: Use Case Diagram

Chapter 7

Implementation

7.1 Exploratory Data Analysis

7.1.1 Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import pandas as pd
import seaborn as sns
from interpret.blackbox import LimeTabular
from interpret import show
import shap
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import roc_auc_score, plot_confusion_matrix,
    ↪ f1_score, accuracy_score , precision_score, recall_score,
    ↪ confusion_matrix
from sklearn.linear_model import LogisticRegression
```

7.1.2 Data Analysis

```
    df = pd.read_csv("./dataset/diabetic_data.csv")
    df.head(10).T
    df.shape
```

```
    df.dtypes
    df.describe().T
    count=0
    for i in df:
        count+=1
            if(count<25):
                print(i)
    Visualising_list={'race','gender','age','admission_type_id','
        ↪ discharge_disposition_id','admission_source_id','
        ↪ time_in_hospital','num_lab_procedures','num_procedures','
        ↪ num_medications','number_outpatient','number_emergency','
        ↪ number_inpatient'}
    for i in Visualising_list:
        a=df[i]
        print(f"\n
            ↪ ----------------------------------------------------------------
            ↪ n{i}")
        print(a.value_counts())
        fig = plt.figure(figsize=(9,5))
        sns.countplot(y= df[i], hue = df.readmitted).set_title(f'{i} VS
            ↪ . Readmission')
        plt.show()
```

7.2 Preprocessing

```
sns.countplot(x=df.readmitted, data=df, palette="pastel", edgecolor="
    ↪ .3")
plt.show()
```

7.2.1 Missing Value Removal

```
#checking target attribute
df['readmitted'] = df['readmitted'].replace('NO', 0)
df['readmitted'] = df['readmitted'].replace('<30', 0)
df['readmitted'] = df['readmitted'].replace('>30', 1)
df['readmitted'].value_counts()
labels=['0','1']
df.readmitted.value_counts().plot.pie(autopct="%1.2f%%",labels=labels)
plt.show()
for col in df.columns:
    if df[col].dtype == object:
        print(col,df[col][df[col] == '?'].count())

#Replacing missing value
df.replace('?', np.nan , inplace=True)
df["race"].fillna(df["race"].mode()[0], inplace = True)

#Dropping values with Unkown gender
df.gender.replace('Unknown/Invalid', np.nan , inplace=True)
df.dropna(subset=['gender'], how='all', inplace = True)
df["race"].isnull().sum()


# Get count of unique values for all columns
unique_counts = df.apply(pd.Series.nunique)

# Sort the unique value counts in descending order
unique_counts_sorted = unique_counts.sort_values(ascending=True)

# Print the sorted unique value counts
print(unique_counts_sorted)


#dropping some attributes due to missing value
drop_list = ['examide' , 'citoglipton', 'weight','encounter_id','
    ↪ patient_nbr','payer_code','medical_specialty']
df.drop(drop_list,axis=1, inplace=True)
```

```
# Determining of Numerical and Categorical Columns

# Find numerical columns
num_cols = df.select_dtypes(include=['int64']).columns.tolist()

#Removing them because even though they have numbers. The numbers
    ↪ represent a category.Refer ID mapping.
num_cols.remove('admission_type_id')
num_cols.remove('discharge_disposition_id')
num_cols.remove('admission_source_id')
num_cols.remove('readmitted')

len(num_cols), num_cols

# Find categorical columns
cat_cols = df.select_dtypes(include=['object']).columns.tolist()
cat_cols.append('admission_type_id')
cat_cols.append('discharge_disposition_id')
cat_cols.append('admission_source_id')
cat_cols.append('readmitted')
len(cat_cols), cat_cols
```

7.3 Feature engineering

```
# re-encoding admission type, discharge type and admission source
    ↪ into fewer categories
df = df.loc[~df.discharge_disposition_id.isin([11,13,14,19,20,21])]
```

7.3.1 Outlier removal

```
count = 0
fig, ax =plt.subplots(nrows=2,ncols=4, figsize=(16,8))
for i in range(2):
    for j in range(4):
        sns.boxplot(x = df[num_cols[count]], palette=["#7FFFD4"],ax=ax[
            ↪ i][j]) # palette = rocket, Wistia
        count = count+1
```

```
#Fun fact:IQR value of 1.5x determines what are outliers
for col in num_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 3 * IQR
        upper_bound = Q3 + 3 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

df.shape
f,ax = plt.subplots(figsize=(8, 6))
sns.heatmap(df[num_cols].corr(), annot=True, linewidths=0.5,linecolor=
    ↪ "black", fmt= '.2f',ax=ax,cmap="coolwarm")
plt.show()

from sklearn.ensemble import IsolationForest

clf = IsolationForest(n_estimators=100, max_samples='auto',
    ↪ contamination=0.03)
clf.fit(df[num_cols])
outliers = clf.predict(df[num_cols])
df = df[outliers != -1]
df.shape
```

7.3.2 Encoding

7.3.2.1 For attributes that are numerical but will be classified

```
df.replace('?', np.nan , inplace=True)
df["race"].isnull().sum()
print(df['diag_1'].isnull().sum())
print(df['diag_2'].isnull().sum())
print(df['diag_3'].isnull().sum())
df['diag_1'].fillna('NaN', inplace=True)
df['diag_2'].fillna('NaN', inplace=True)
df['diag_3'].fillna('NaN', inplace=True)
print(df['diag_1'].nunique())
print(df['diag_2'].nunique())
print(df['diag_3'].nunique())
```

```
df['diag_1'].value_counts()
diag_cols = ['diag_1','diag_2','diag_3']
for col in diag_cols:
    df[col] = df[col].str.replace('E','0')
    df[col] = df[col].str.replace('V','0')
    df[col] = df[col].str.replace('NaN','-1')
    #because it has 250.0X as value which makes it hard to encode
    condition = df[col].str.contains('250')
    df.loc[condition,col] = '250'
for i in df['diag_1']:
    if(i=="?" or i=="NaN" or i=="E" or i == "V57" or i=="250.03"):
        print(i)
type(df['diag_1'][1])
df[diag_cols] = df[diag_cols].astype(float)
type(df['diag_1'][1])
df['A1Cresult'] = df['A1Cresult'].map({'Norm': 0, '>7': 1, '>8': 1, '
    ↪ None': -99})
df['max_glu_serum'] = df['max_glu_serum'].map({'Norm': 0, '>200': 1, '
    ↪ >300': 1, 'None': -99})
def assign_category(value):
    if value >= 390 and value <= 459:
        group = 'Diseases of the Circulatory System'
    elif value >= 460 and value <= 519:
        group = 'Diseases of the Respiratory System'
    elif value >= 520 and value <= 579:
        group = 'Diseases of the Digestive System'
    elif value >= 580 and value <= 629:
        group = 'Diseases of the Genitourinary System'
    elif value >= 630 and value <= 679:
        group = 'Complications of Pregnancy, Childbirth, and the
            ↪ Puerperium'
    elif value >= 680 and value <= 709:
        group = 'Diseases of the Skin and Subcutaneous Tissue'
    elif value >= 710 and value <= 739:
        group = 'Diseases of the Musculoskeletal System and Connective
            ↪ Tissue'
    elif value >= 740 and value <= 759:
        group = 'Congenital Anomalies'
    elif value >= 760 and value <= 779:
```

```
        group = 'Certain Conditions Originating in the Perinatal Period
            ↪ '
    elif value >= 780 and value <= 799:
        group = 'Symptoms, Signs, and Ill-Defined Conditions'
    elif value >= 800 and value <= 999:
        group = 'Injury and Poisoning'
    else:
        group = 'Other'
    return group
for col in diag_cols:
     df[col] = df[col].apply(assign_category)
print(df.diag_1.value_counts())
sns.countplot(y="diag_1", hue="readmitted", data=df).set_title('
    ↪ Primary Diagnosis VS. Readmission')
plt.show()
print(df.diag_2.value_counts())
sns.countplot(y="diag_2", hue="readmitted", data=df).set_title('
    ↪ Secondary Diagnosis VS. Readmission')
plt.show()
print(df.diag_3.value_counts())
sns.countplot(y="diag_3", hue="readmitted", data=df).set_title('
    ↪ Additional Secondary Diagnosis VS. Readmission')
plt.show()
```

7.3.2.2 For normal categorical data

```
drugs = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
    ↪ 'glimepiride', 'glipizide', 'glyburide', 'pioglitazone',
    'rosiglitazone', 'acarbose', 'miglitol', 'insulin', 'glyburide-
        ↪ metformin', 'tolazamide', 'metformin-pioglitazone',
    'metformin-rosiglitazone', 'glimepiride-pioglitazone', 'glipizide-
        ↪ metformin', 'troglitazone', 'tolbutamide', 'acetohexamide']
for col in drugs:
df[col] = df[col].replace('No', 0)
df[col] = df[col].replace('Steady', 1)
df[col] = df[col].replace('Up', 1)
df[col] = df[col].replace('Down', 1)
df[col] = df[col].astype(int)
```

```
# One hot Encoding Race to convert categorical values to numerical
    ↪ ones
one_hot_data = pd.get_dummies(df, columns=['race'], prefix=["enc"])

# One hot Encoding Admission, Discharge, and Admission Source IDs
columns_ids = ['admission_type_id', 'discharge_disposition_id', '
    ↪ admission_source_id']
#one_hot_data = pd.get_dummies(one_hot_data, columns=columns_ids,
    ↪ dtype=int, prefix=columns_ids)
one_hot_data[columns_ids] = one_hot_data[columns_ids].astype('str')
one_hot_data = pd.get_dummies(one_hot_data, columns=columns_ids)

df.info()

# code age intervals [0-10) - [90-100) from 1-10
for i in range(0,10):
    df['age'] = df['age'].replace('['+str(10*i)+'-'+str(10*(i+1))+')',
        ↪ i+1)
df['age'] = df['age'].astype(int)
df['age'].value_counts()

df.to_csv('./dataset/PreprocessedData.csv')
```

7.3.2.3 Label Encoding binary data

```
df['gender'].value_counts()
df['race'].value_counts()
df['change'].value_counts()
df['diabetesMed'].value_counts()
df['diag_1'].value_counts()

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])
df['race'] = le.fit_transform(df['race'])
df['change'] = le.fit_transform(df['change'])
df['diabetesMed']=le.fit_transform(df['diabetesMed'])
df['diag_1']=le.fit_transform(df['diag_1'])
df['diag_2']=le.fit_transform(df['diag_2'])
```

```
df['diag_3']=le.fit_transform(df['diag_3'])
df['diag_1'].value_counts()
df['diag_2'].value_counts()
df['diag_3'].value_counts()
df['gender'].value_counts()
df['race'].value_counts()
df['change'].value_counts()
df['diabetesMed'].value_counts()
for i in df:
    print(df[i
print(df['diag_1'])
```

7.4 Model Building

7.4.1 Resampling

```
from sklearn.utils import resample



not_readmitted = df[df.readmitted==0]
readmitted = df[df.readmitted==1]


not_readmitted_sampled = resample(not_readmitted,
                                  replace = False,
                                  n_samples = len(readmitted),
                                  random_state = 42)

downsampled = pd.concat([not_readmitted_sampled, readmitted])
downsampled.readmitted.value_counts()
df = pd.DataFrame(downsampled)
df.shape
df['readmitted'].value_counts()
labels=['0','1']
df.readmitted.value_counts().plot.pie(autopct="%1.2f%%",labels=labels)
plt.show()
```

7.4.2 Getting data ready

```
#df=pd.read_csv('./dataset/Encoded.csv')
df = pd.read_csv('C:/Users/adith\Desktop/Doing it all over again\
    ↪ dataset/Encoded_binary.csv')
df.shape
df['readmitted'].value_counts()
X=df.iloc[:,1:-1].values
y=df.iloc[:,-1].values
print(X)
print(y)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,
    ↪ random_state=0)
```

7.4.3 Test case function

```
    def test_model(model):
    li_score={}
    # Predict probabilities for test set
    y_prob = model.predict_proba(X_test)[:,1]

    # Calculate ROC AUC score
    roc_auc = roc_auc_score(y_test, y_prob)
    print('ROC AUC Score:', roc_auc)



    # Make predictions on test set
    y_pred = model.predict(X_test)

    # Calculate F1 score
    f1 = f1_score(y_test, y_pred)
    print('F1 Score:', f1)




    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    print('Accuracy Score:', accuracy)
```

```
    # calculate precision and recall
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    print('Precision:', precision)
    print('Recall:', recall)

    li_score.update({'Accuracy':accuracy,'Precision':precision,'Recall
        ↪ ':recall,'f1_score':f1,'ROC score':roc_auc})


    cm = confusion_matrix(y_test, y_pred)
    # plot confusion matrix
    sns.heatmap(cm, annot=True, cmap="Blues")
    plt.xlabel("Predicted as readmitted")
    plt.ylabel("Actually readmitted")
    plt.show()

    return(li_score)
```

7.4.4 Logistic Regression

```
#Changing the model a bit
from sklearn.linear_model import LogisticRegression
lg_new=LogisticRegression(solver = "liblinear",class_weight="balanced"
    ↪ ,random_state = 42)
lg_new.fit(X_train,y_train)
Log_R=test_model(lg_new)
```

7.4.5 Random Forest

```
rm = RandomForestClassifier(n_estimators = 10, max_depth=25, criterion
    ↪  = "gini", min_samples_split=10)
rm.fit(X_train, y_train)
Rand_forest=test_model(rm)
print(rm.feature_importances_)
```

7.4.6 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=28, criterion = "entropy",
    ↪ min_samples_split=10)
dtree.fit(X_train, y_train)
d_tree=test_model(dtree)
```

7.4.7 Gradient Boost

```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(X_train, y_train)
grad_boost=test_model(xgb)
```

7.5 Explainable AI

7.5.1 Intializing data

```
df_rd = pd.read_csv('./dataset/Encoded_binary.csv')
df=df_rd.iloc[:,1:]
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
train_test_split(X, y, test_size=0.20, random_state=2021)
# Split the data for evaluation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    ↪ =0.20, random_state=2021)
print(X_train.shape)
print(X_test.shape)
train_test_split(X, y, test_size=0.20, random_state=2021)
```

7.5.2 Interpretable models

Some models are interpretable by default and hence adding few changes to it makes it visually interpretable.

7.5.2.1 Logistic regression

```
# Fit logistic regression model
lr = LogisticRegression(
    random_state=2021, feature_names=X_train.columns, penalty='l1',
        ↪ solver='liblinear')
lr.fit(X_train, y_train)
print("Training finished.")
print("Training finished.")
y_pred = lr.predict(X_test)
print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
print(f"Accuracy {accuracy_score(y_test, y_pred)}")
lr_local = lr.explain_local(
    X_test[:100], y_test[:100], name='Logistic Regression')
show(lr_local)
# Explain global logistic regression model
lr_global = lr.explain_global(name='Logistic Regression')
show(lr_global)
```

7.5.2.2 Decision Tree

```
# Fit decision tree model
tree = ClassificationTree()
tree.fit(X_train, y_train)
print("Training finished.")
y_pred = tree.predict(X_test)
print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
print(f"Accuracy {accuracy_score(y_test, y_pred)}")
# Explain local prediction
tree_local = tree.explain_local(X_test[:100], y_test[:100], name='Tree
    ↪ ')
show(tree_local)
```

7.5.2.3 Explainable Boosting Classification

```
# Fit Explainable Boosting Machine
ebm = ExplainableBoostingClassifier(random_state=2021)
ebm.fit(X_train, y_train)
print("Training finished.")
y_pred = ebm.predict(X_test)
```

```
print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
print(f"Accuracy {accuracy_score(y_test, y_pred)}")
# Explain locally
ebm_local = ebm.explain_local(X_test[:100], y_test[:100], name='EBM')
show(ebm_local)
# Explain globally
ebm_global = ebm.explain_global(name='EBM')
show(ebm_global)
```

7.5.3 Lime

```
# Fit blackbox model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
print(f"Accuracy {accuracy_score(y_test, y_pred)}")
# %% Apply lime
# Initilize Lime for Tabular data
lime = LimeTabular(model=rf,
                   data=X_train,
                   random_state=1)


# Get local explanations
lime_local = lime.explain_local(X_test[-20:],
                                y_test[-20:],
                                name='LIME')

show(lime_local)
```

7.5.4 SHAP

```
# Create SHAP explainer
explainer = shap.TreeExplainer(rf)
# Calculate shapley values for test data
start_index = 1
end_index = 100
```

```
shap_values = explainer.shap_values(X_test[start_index:end_index])
X_test[start_index:end_index]
print(shap_values[0].shape)
shap_values
shap.initjs()
prediction = rf.predict(X_test[start_index:end_index])[0]
print(f"The RF predicted: {prediction}")
shap.force_plot(explainer.expected_value[1],
                shap_values[1],
                X_test[start_index:end_index]) # for values
shap.summary_plot(shap_values, X_test)
```

7.5.4.1 Choosing best attributes and using pickle to store new model

```
import xgboost
import shap

# train an XGBoost model
model = xgboost.XGBRegressor().fit(X_train, y_train)

# explain the model's predictions using SHAP
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
top_attr = ['admission_type_id', 'admission_source_id', '
    ↪ discharge_disposition_id',
           'time_in_hospital', 'number_diagnoses', 'diag_1', 'diag_2',
               ↪ 'num_medications',
           'age', 'gender', 'number_inpatient', 'number_outpatient']
X = df[top_attr]
Y = pd.get_dummies(df['readmitted'])
from sklearn.model_selection import train_test_split
# X=df.drop('readmitted',axis=1)
# Y=df['readmitted']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size
    ↪ =0.2, random_state=0)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
```

```
print(f"Accuracy {accuracy_score(y_test, y_pred)}")
import pickle
filename= 'trained_model.sav'
pickle.dump(rf,open(filename,'wb'))
```

7.6 Frontend GUI

```
    import pandas as pd
import streamlit as st
import numpy as np
import pickle
from sklearn.metrics import f1_score, accuracy_score
import streamlit.components.v1 as components
import shap
st.title("Diabetes Prediction App")


load_model = pickle.load(open('trained_model.sav', 'rb'))



#input
input = []

#1 attribute
admission_types = {
    'Emergency': 1,
    'Urgent': 2,
    'Elective': 3,
    'Newborn': 4,
    'Not Available': 5,
    'NULL': 6,
    'Trauma Center': 7,
    'Not Mapped': 8
}
selected_id = st.selectbox(
    'Select an admission type ID', list(admission_types.keys()))
selected_desc = admission_types[selected_id]
```

```
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)



#second attribute
admission_source_dict = {
    "Physician Referral": 1,
    "Clinic Referral": 2,
    "HMO Referral": 3,
    "Transfer from a hospital": 4,
    "Transfer from a Skilled Nursing Facility (SNF)": 5,
    "Transfer from another health care facility": 6,
    "Emergency Room": 7,
    "Court/Law Enforcement": 8,
    "Not Available": 9,
    "Transfer from critical access hospital": 10,
    "Normal Delivery": 11,
    "Premature Delivery": 12,
    "Sick Baby": 13,
    "Extramural Birth": 14,
    "Transfer From Another Home Health Agency": 18,
    "Readmission to Same Home Health Agency": 19,
    "Not Mapped": 20,
    "Unknown/Invalid": 21,
    "Transfer from hospital inpatient/same facility resulting in a
       ↪ separate claim": 22,
    "Born inside this hospital": 23,
    "Born outside this hospital": 24,
    "Transfer from Ambulatory Surgery Center": 25,
    "Transfer from Hospice": 26,
    "NULL": 17
}

selected_id = st.selectbox('Select an admission source ID', list(
   ↪ admission_source_dict.keys()))
selected_desc = admission_source_dict[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)
```

```
#3
discharge_dispositions = {
'Discharged to home': 1,
'Discharged/transferred to another short term hospital': 2,
'Discharged/transferred to SNF': 3,
'Discharged/transferred to ICF': 4,
'Discharged/transferred to another type of inpatient care institution'
    ↪ : 5,
'Discharged/transferred to home with home health service': 6,
'Left AMA': 7,
'Discharged/transferred to home under care of Home IV provider': 8,
'Admitted as an inpatient to this hospital': 9,
'Neonate discharged to another hospital for neonatal aftercare': 10,
'Still patient or expected to return for outpatient services': 12,
'Discharged/transferred within this institution to Medicare approved
    ↪ swing bed': 15,
'Discharged/transferred/referred another institution for outpatient
    ↪ services': 16,
'Discharged/transferred/referred to this institution for outpatient
    ↪ services': 17,
'NULL': 18,
'Discharged/transferred to another rehab fac including rehab units of
    ↪ a hospital.': 22,
'Discharged/transferred to a long term care hospital.': 23,
'Discharged/transferred to a nursing facility certified under Medicaid
    ↪  but not certified under Medicare.': 24,
'Not Mapped': 25,
'Unknown/Invalid': 26,
'Discharged/transferred to another Type of Health Care Institution not
    ↪  Defined Elsewhere': 30,
'Discharged/transferred to a federal health care facility.': 27,
'Discharged/transferred/referred to a psychiatric hospital of
    ↪ psychiatric distinct part unit of a hospital': 28,
'Discharged/transferred to a Critical Access Hospital (CAH).': 29
}

selected_id = st.selectbox('Select an discharge disposition ID', list(
    ↪ discharge_dispositions.keys()))
```

```
selected_desc = discharge_dispositions[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)

#4
time_in_hospital = st.number_input("Enter the time spent in hopsital(
    ↪ in days):")
st.write(f"Entered {time_in_hospital}")
time_in_hospital = round(time_in_hospital)
input.append(time_in_hospital)

#5
number_diagnosis = st.number_input("Enter the number of diagnosis done
    ↪ :")
st.write(f"Entered {number_diagnosis}")
number_diagnosis = round(number_diagnosis)
input.append(number_diagnosis)

#6 and 7
icd={
    'Complications of Pregnancy, Childbirth, and the Puerperium': 0,
    'Congenital Anomalies': 1,
    'Diseases of the Circulatory System': 2,
    'Diseases of the Digestive System': 3,
    'Diseases of the Genitourinary System': 4,
    'Diseases of the Musculoskeletal System and Connective Tissue': 5,
    'Diseases of the Respiratory System': 6,
    'Diseases of the Skin and Subcutaneous Tissue': 7,
    'Injury and Poisoning': 8,
    'Other': 9,
    'Symptoms, Signs, and Ill-Defined Conditions': 10
}

selected_id = st.selectbox('Select the primary health problem:', list(
    ↪ icd.keys()))
selected_desc = icd[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)
```

```
selected_id = st.selectbox('Select the seco=ondary health problem:',
    ↪ list(icd.keys()))
selected_desc = icd[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)

#8
number_medication = st.number_input("Number of distinct generic names
    ↪ administered:")
st.write(f"Entered {number_medication}")
number_medication=round(number_medication)
input.append(number_medication)

#9

age = {'0-10': 1,
       '11-20': 2,
       '21-30': 3,
       '31-40': 4,
       '41-50': 5,
       '51-60': 6,
       '61-70': 7,
       '71-80': 8,
       '81-90': 9,
       '91+': 10
    }

selected_id = st.selectbox('Select the primary health problem:', list(
    ↪ age.keys()))
selected_desc = age[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)


#10
gender={
    'Female':0,
    'Male':1
```

```
}
selected_id = st.selectbox('Select the primary health problem:', list(
    ↪ gender.keys()))
selected_desc = gender[selected_id]
st.write('You selected:', selected_id, '(', selected_desc, ')')
input.append(selected_desc)

#11
number_inpatient=st.number_input("Number of emergency visits of the
    ↪ inpatient in the year preceding the encounter:")
st.write(f"Entered {number_inpatient}")
number_inpatient=round(number_inpatient)
input.append(number_inpatient)

#12
number_outpatient=st.number_input("Number of emergency visits of the
    ↪ outpatient in the year preceding the encounter:")
st.write(f"Entered {number_outpatient}")
number_outpatient=round(number_outpatient)
input.append(number_outpatient)



# MODEL
input_df = pd.DataFrame([input], columns=['admission_type_id', '
    ↪ admission_source_id' ,
                                        'discharge_disposition_id', '
                                            ↪ time_in_hospital',
                                        'number_diagnoses', 'diag_1', '
                                            ↪ diag_2' ,
                                        'num_medications', 'age', '
                                            ↪ gender',
                                        'number_inpatient', '
                                            ↪ number_outpatient'])

pred = load_model.predict(input_df)
print(pred)
st.write(' ')
st.write(' ')
```

```
if (pred[0][0] == 1):
    st.write('### **Readmission chance is low**')
else:
    st.write('### **Chance of readmission is high**')


def st_shap(plot, height=None):
    shap_html = f"<head>{shap.getjs()}</head><body>{plot.html()}</body
        ↪ >"
    components.html(shap_html, height=height)
st.write(' ')
st.write(' ')
st.write(' ')
import shap
explainer = shap.TreeExplainer(load_model)

st.write
# Calculate Shap values
shap_values = explainer.shap_values(input_df)
shap.initjs()
pred = load_model.predict(input_df)
print(pred)
st.write(' ')
st.write(' ')
if (pred[0][0] == 1):
    st.write('### **Readmission chance is low**')
elif(pred[0][1]==1):
    st.write('### **Chance of readmission is high**')
else:
    st.write("Something went wrong")

st.write(input)
def st_shap(plot, height=None):
    shap_html = f"<head>{shap.getjs()}</head><body>{plot.html()}</body
        ↪ >"
    components.html(shap_html, height=height)
st.write(' ')
st.write(' ')
st.write(' ')
```

```
import shap
explainer = shap.TreeExplainer(load_model)
# Calculate Shap values
shap_values = explainer.shap_values(input_df)
shap.initjs()
st.write('### Contributions of the values are as such(Here scale is
    ↪ 0-1):')
st_shap(shap.force_plot(explainer.expected_value[1], shap_values[1],
    ↪ input_df),400)
st_shap(shap.force_plot(explainer.expected_value[1], shap_values[1],
    ↪ input_df),400)
```

Chapter 8

Testing

8.1 Testing Methodologies

- Data Preprocessing Testing: Create test cases to validate the preprocessing steps applied to the input data. Ensure that the data cleaning, missing value handling, feature scaling, and encoding of categorical variables are performed correctly. Test how the model handles edge cases and outliers in the data.
- Model Evaluation Testing: Develop test cases to evaluate the model's performance and metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. Verify that the model's predictions align with the expected outcomes, especially for critical scenarios such as identifying patients at high risk of readmission.
- Explainability Testing: Validate the interpretability and explainability of the model by testing the effectiveness of the techniques used. Verify if the generated feature importance scores, visualizations, or other interpretability methods accurately reflect the model's decision-making process and provide meaningful insights.
- Robustness Testing: Test the model's performance under different conditions, such as variations in the patient population, changes in data distributions over time, and different hospital settings. Assess how the model handles unforeseen situations and edge cases to ensure its reliability and generalizability.
- Regression Testing: After any modifications or updates to the code or model, run automated tests to ensure that the changes have not introduced new issues or regressions. Test the modified code against a set of predefined test cases to verify that the expected outputs are still generated correctly.

8.2 Unit Testing

Developers often carry out unit testing, which entails creating and running test cases for each unit of code. The test cases are created to ensure that the unit of code carries out the required function, generates the anticipated output, and appropriately handles edge cases and fault situations. Since unit tests are frequently automated, testing during the development process can be done quickly and effectively. Every time the code is modified, automated unit tests can be run automatically to ensure that no new issues or regression.Unit testing has a number of advantages like helping to find defects early, improving code quality, etc.

8.3 System Testing

System testing is a kind of software testing that involves thoroughly examining an entire system or software application. It is done to make sure the program complies with the requirements and runs properly in the environment it was designed for The integrated system is subjected to system testing to ensure that all modules or components function as intended and that the system satisfies the criteria outlined in the software requirements specification (SRS).

8.4 Quality Assurance

Software quality standards and criteria are ensured through the quality assurance (QA) process, which is a step in the software development process. By setting and upholding stringent quality standards throughout the development process, QA's major objective is to prevent errors and problems from appearing in the product. The following steps are often included in the QA process:

- Gathering and analyzing the software application's requirements is the first step in the quality assurance process.
- Following the identification of the requirements, a test plan is created to make sure that each requirement is adequately tested.
- Any flaws or errors found during testing are noted, followed up on, and sent to the development team to be fixed.
- The reports give a general assessment of the software's quality and point out any shortcomings.

Chapter 9

Results and Performance Analysis

9.1 Evaluation of models

We have implemented a function that takes a machine learning model as input and calculates several evaluation metrics, including the ROC AUC score, F1 score, accuracy, precision, and recall. This function serves as a valuable tool for assessing the performance .

```
def test_model(model):
    li_score={}
    # Predicting the output for the ROC score
    y_prob = model.predict_proba(X_test)[:,1]

    # Calculate ROC AUC score
    roc_auc = roc_auc_score(y_test, y_prob)
    print('ROC AUC Score:', roc_auc)



    # Make predictions on test set
    y_pred = model.predict(X_test)

    # Calculate F1 score
    f1_per = f1_score(y_test, y_pred)
    print('F1 Score:', f1_per)




    # Calculation of the accuracy score
    accu_score = accuracy_score(y_test, y_pred)
```

```
    print('Accuracy Score:', accu_score)



    # calculation of the precision and recall
    preci_per = precision_score(y_test, y_pred)
    rec_per = recall_score(y_test, y_pred)
    print('Precision:', preci_per)
    print('Recall:', rec_per)

    li_score.update({'Accuracy':accu_score,'Precision':preci_per,'
        ↪ Recall':rec_per,'f1_score':f1,'ROC score':roc_auc})



    cm = confusion_matrix(y_test, y_pred)
    # plot confusion matrix
    sns.heatmap(cm, annot=True, cmap="Blues")
    plt.xlabel("Predicted as readmitted")
    plt.ylabel("Actually readmitted")
    plt.show()

    return(li_score)

#Names of trained models
print(Log_R,Rand_forest,d_tree,grad_boost)
model={
    'LogisticRegression':Log_R,
    'Random forest':Rand_forest,
    'Decision Tree Classifier':d_tree,
    'Gradient Boosting':grad_boost
}

def create_metrics_table(metrics_dict):
    df = pd.DataFrame(metrics_dict)
    df = df.transpose()
    df.columns = ['Accuracy', 'Precision', 'Recall', 'F1 score','Roc
        ↪ score']
    df.index.name = 'Model'
    return df
```

```
table=create_metrics_table(model)
print(table)
```

Thus,the performance of each model is as follow:

9.1.1 Logistic Regression

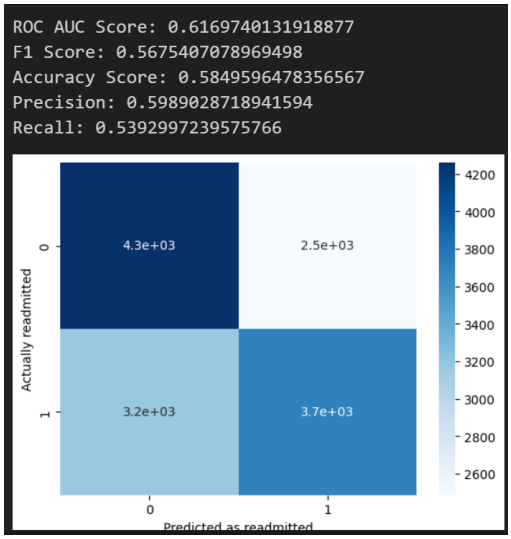

Figure 9.1: Logistic Regression performance

9.1.2 Decision Tree

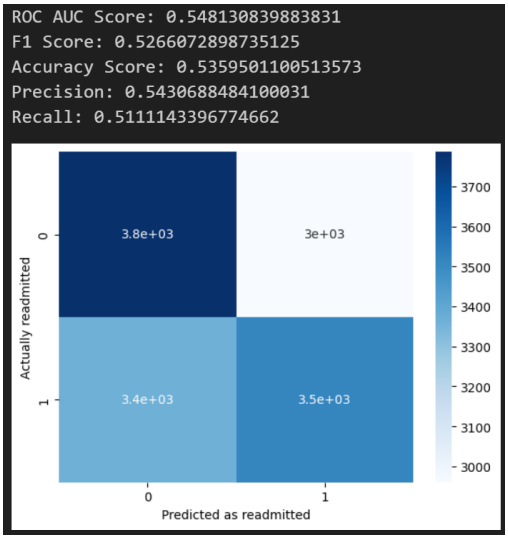

Figure 9.2: Decision Tree performance

9.1.3 Gradient Boost

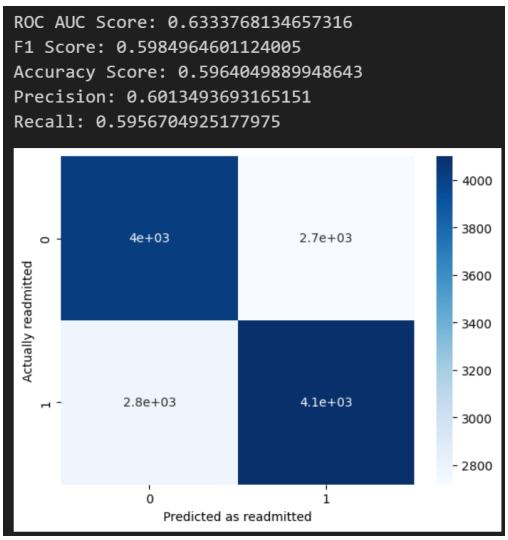

Figure 9.3: Gradient Boos performance

9.1.4 Random Forest

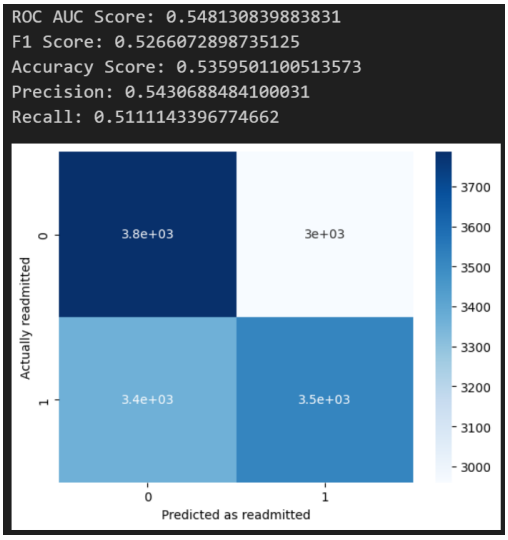

Figure 9.4: Random Forest performance

Overall view of the performance:

9.1.5 Overall performance

Model	Accuracy	Precision	Recall	F1 score	Roc score
LogisticRegression	0.584960	0.598903	0.539300	0.567541	0.616974
Random forest	0.577916	0.582410	0.580125	0.581265	0.609550
Decision Tree Classifier	0.535950	0.543069	0.511114	0.526607	0.548131
Gradient Boosting	0.596405	0.601349	0.595670	0.598496	0.633377

Figure 9.5: Performance analysis

9.2 Interpreting Models Result

9.2.1 Logistic Regression

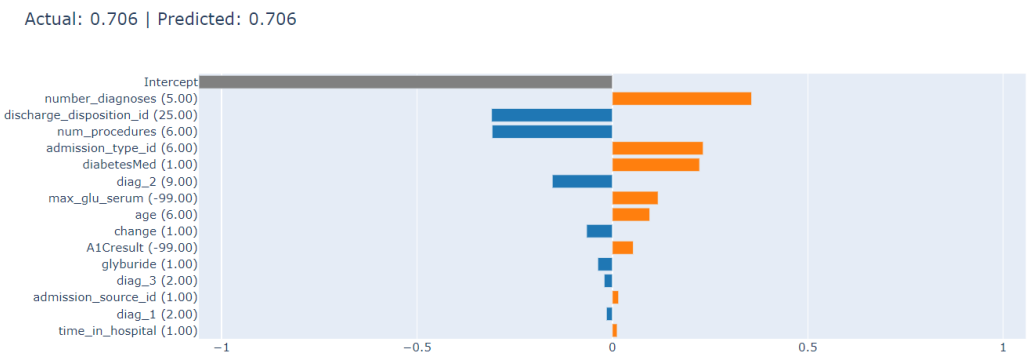

Figure 9.6: Attributes effects on Logistic Regression

9.2.2 Decision Tree

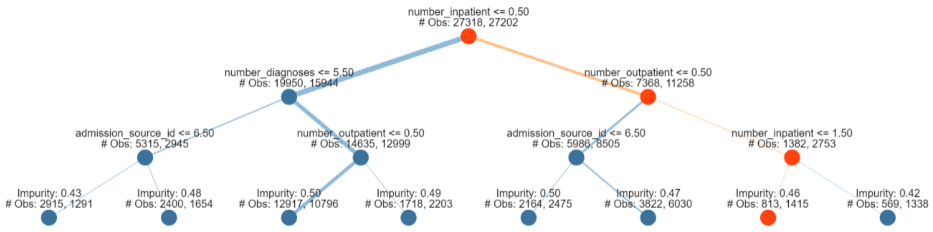

Figure 9.7: Attributes effects on Decision Tree

9.2.3 Gradient Boost

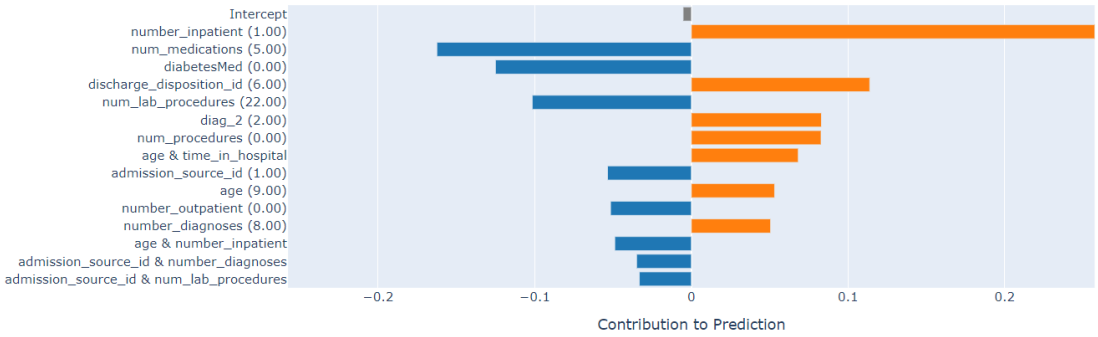

Figure 9.8: Attributes effects on Gradient Boost

9.2.4 Random Forest

9.2.4.1 LIME

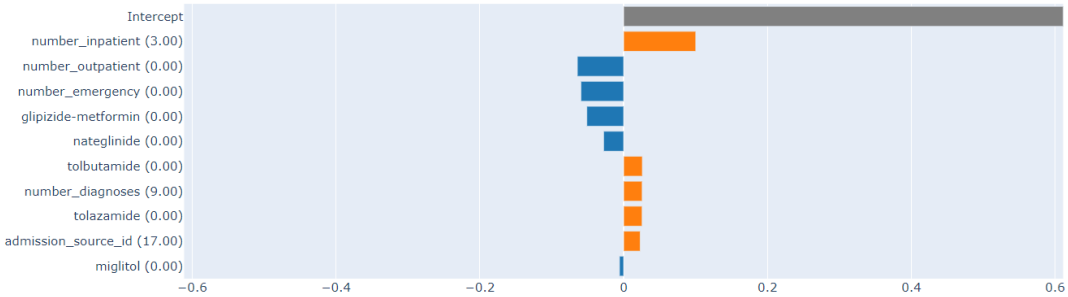

Figure 9.9: Attributes effects on Random Forest explained using LIME

9.2.4.2 SHAP

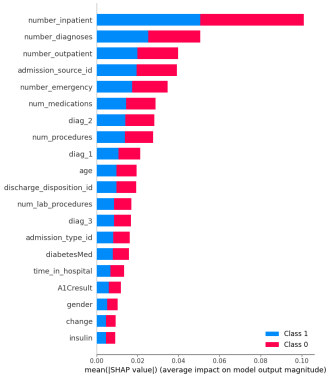

Figure 9.10: Attributes effects on Random Forest explained using SHAP

9.3 GUI interface

9.3.1 GUI page

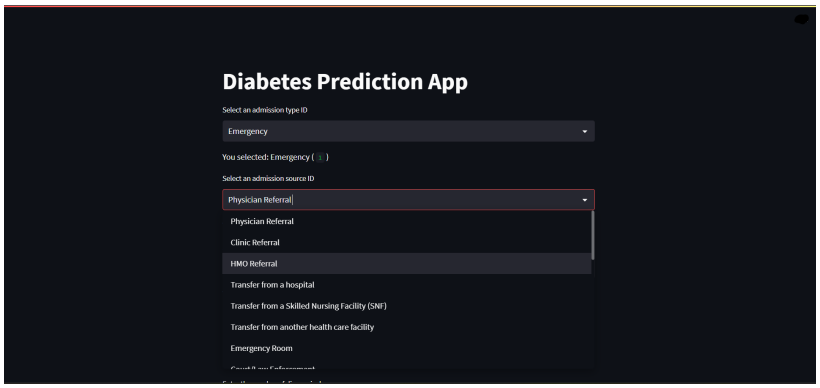

Figure 9.11: Interact-able GUI

9.3.2 Prediction Result

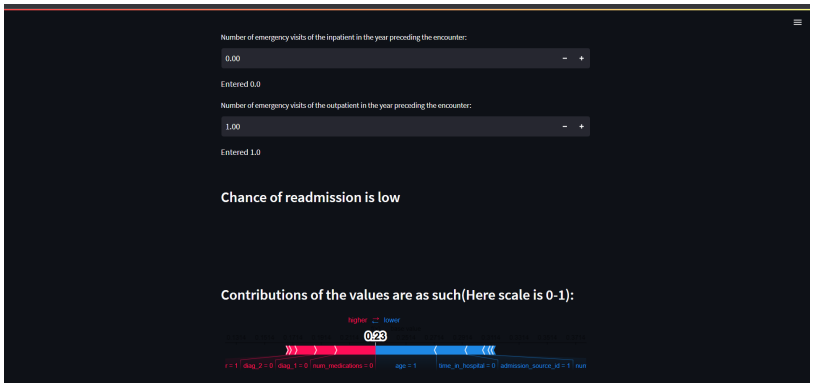

Figure 9.12: Result and its explanation on what attribute contributed to the prediction

Chapter 10

Conclusion

The use of machine learning and Explainable AI (XAI) techniques has shown promise in recent years for predicting patient readmission. Machine learning models, such as neural networks, are well-suited for handling complex healthcare datasets, making them valuable tools in the industry. By employing XAI techniques, medical professionals can gain a better understanding of the predictions generated by these machine learning models, leading to increased trust in the results and facilitating early intervention and quality patient care. However, it is important to acknowledge that these methods are still relatively new and require further research to fully grasp their potential and limitations. Ethical considerations are also paramount when utilizing machine learning and XAI in healthcare, as biases and potential negative consequences must be minimized. In conclusion, the integration of machine learning and XAI in predicting patient readmission holds promise for improving patient outcomes, but ongoing research and ethical awareness are necessary to fully harness the benefits of these technologies in healthcare.

References

[1] Molnar, C. (2020). *Interpretable Machine Learning*. Retrieved from https://christophm.github.io/interpretable-ml-book/

[2] Singh, S., Sharma, D. (2020). *Explainable AI: A Primer*. Retrieved from https://arxiv.org/abs/2002.04838

[3] Lohn, A., Murray, R., Rutherford, A. W. (2018). *Explainable AI: From Black Box to Glass Box*. Retrieved from https://ieeexplore.ieee.org/abstract/document/8623907

[4] Kaggle. (2019). *Interpretable Machine Learning with LIME and SHAP*. Retrieved from https://www.youtube.com/watch?v=6tDkSroC6_U

[5] IBM Developer. (2020). *Explainable AI and Machine Learning Models*. Retrieved from https://www.youtube.com/watch?v=tjbSv_XK6x4

[6] Hu, Y., & Sokolova, M. (2020). Explainable Multi-class Classification of Medical Data. arXiv preprint arXiv:2012.13796.

[7] Tjoa, E., & Guan, C. (2020). A survey on explainable artificial intelligence (XAI): Toward medical XAI. IEEE transactions on neural networks and learning systems, 32(11), pp.4793-4813.

[8] Ashfaq, A., Sant'Anna, A., Lingman, M., & Nowaczyk, S. (2019). Readmission prediction using deep learning on electronic health records. Journal of biomedical informatics, 97, p.103256.

[9] Moradi, M., & Samwald, M. (2021). Post-hoc explanation of black-box classifiers using confident itemsets. Expert Systems with Applications, 165, p.113941.

[10] Documentation on Interpretable/Explainable AIhttps://github.com/jphall663/awesome-machine-learning-interpretability