PREDICTING HOSPITAL READMISSION USING MACHINE LEARNING

BY

AUDU MOHAMMAD DANJUMA

PSC2008115

DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF PHYSICAL SCIENCES,
UNIVERSITY OF BENIN,
BENIN CITY,
EDO STATE, NIGERIA.

FEBURARY 2025

CERTIFICATION

This is to certify that this project work was carried out by **AUDU MOHAMMED DANJUMA** with Matriculation Number **PSC2008115** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.sc) Degree in Computer Science of the University of Benin

_____ _____

DR. (MRS) A.R. USIOBAIFO **DATE**
(Project Supervisor)

_____ _____

PROF.GODSPOWER O. EKUOBASE **DATE**
(Head of Department)

APPROVAL

This project work is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ _____

PROF. GODSPOWER O. EKUOBASE, PHD **DATE**
(Head of Department)

DEDICATION

This project is dedicated to God for granting me the strength and wisdom to see it through to completion and for His guidance throughout my time at the University of Benin (UNIBEN). I also dedicate it to my Mum, Mrs. Maryam Audu**,** and my sisters for their unwavering love, support, and guidance, as well as to everyone who has contributed significantly to the success of this project and my academic journey.

ACKNOWLEDGEMENT

I want to express my deepest gratitude to God Almighty for giving me the strength, wisdom, and direction to complete this project and for guiding me throughout my academic journey.

A special appreciation goes to my project supervisor, Dr. (Mrs.) A.R. Usiobaifo, whose patience, encouragement, and dedication made a huge difference in the success of this project. Her insightful feedback, constant support, and willingness to guide me every step of the way truly motivated me to do my best. I am incredibly grateful for her time, effort, and the knowledge she shared with me. Without her guidance, this project wouldn't have come together as it did.

I would also like to extend my sincere gratitude to Prof. G.O. Ekuobase, the Head of the Department of Computer Science, for his exceptional leadership, dedication, and contributions to the department. His commitment to academic excellence and student development has created an environment that fosters learning and growth. I truly appreciate his guidance, encouragement, and the impact he has had on my academic journey.

Finally, I want to thank everyone who supported me throughout this journey. A huge shoutout to my family and friends for always being there with encouragement, motivation, and support. Your belief in me kept me going.

Table of Contents

LIST OF FIGURES

ABSTRACT

Hospital readmissions create challenges for healthcare systems, increasing costs and putting pressure on resources. This project introduces a machine learning-based system designed to predict patient readmissions, helping medical personnel and hospitals take early action to improve patient care and manage resources more effectively. By analyzing electronic health record (EHR) data, the model assesses a patient's risk and provides explanations for key factors influencing the prediction.

The system was trained on a dataset containing patient details such as age, medical history, lab results, and past hospital visits. It was developed using Python for machine learning, Express.js for the backend, and TypeScript with React for the frontend, ensuring smooth data processing and an easy-to-use interface. Strong security features like authentication, encryption, and error handling were added to protect patient information.

The result shows that the model was able to achieve **63.87% accuracy**, with recall scores of **72%** and **55%** in different areas. These results highlight the model's ability to predict readmissions while also showing areas where improvements can be made through better data processing and tuning.

By using predictive analytics, this system helps healthcare professionals make informed decisions, reduce avoidable readmissions, and improve hospital efficiency. This project demonstrates how AI-powered solutions can transform healthcare by enabling proactive patient management and better decision-making.

CHAPTER ONE
INTRODUCTION

1.1 BACKGROUND STUDY

The term “hospital readmission” refers to a patient’s return to the hospital for treatment within a brief period after discharge, usually within 30 days. Effective transitions of care are essential for quality in healthcare, as the shift from hospital to post-acute care settings involves a complex set of activities that place patients at a heightened risk for adverse events. This process goes beyond simply transferring information from inpatient to outpatient clinicians; it also requires coordinating care between the inpatient team, the patient, and their family caregivers. The Centers for Medicare and Medicaid Services (CMS) developed the Hospital Readmission Reduction Program (HRRP) to focus on improving this transition, identifying high 30-day re- hospitalization rates as a key quality metric, and enforcing financial penalties of up to 3% on all Medicare admissions for excessive readmissions. Increasingly, healthcare systems are recognizing the importance of involving patients and their family caregivers in personalized discharge planning to help ensure a smooth and effective transition of care.

High readmission rates are often seen as an indicator that patients might not be receiving the complete care or support they need after leaving the hospital. For instance, a patient might be discharged but still have unresolved medical issues or need better guidance on how to manage their condition at home. When these needs are not fully met, the patient may become unwell.

Recent advances in technology have facilitated an increased flow of information, enhancing medical research and clinical care with a focus on supporting disease diagnosis, prediction, and personalized treatment options . Research indicates that artificial intelligence (AI) holds promise in predicting hospital readmissions, with machine learning capable of quickly and accurately evaluating complex clinical outcomes that might escape human perception. However, AI integration in healthcare remains limited, and many hospitals have yet to adopt these methods. Many healthcare professionals are unfamiliar with how to implement such tools in clinical practice. Additionally, this study target the prediction of preventable Hospital readmissions using machine learning, to ensure safe and effective healthcare decisions.

Applying machine learning (ML) to reduce hospital readmissions is transforming patient care and resource use in healthcare by analyzing electronic health record (EHR) data, identifying

which patients have a higher likelihood of readmission. This predictive capability allows for early interventions to prevent readmission. By leveraging data-driven insights, my model enables the creation of personalized discharge and follow-up care plans that address each patient's unique health profile and post-discharge needs, improving both care outcomes and resource efficiency within the healthcare setting.

1.2 STATEMENT OF PROBLEM

The problem of high hospital readmission rates is complex, with multiple contributing factors like "Fragmented care" where patients receive services from multiple healthcare providers, often leads to poor coordination and communication, increasing the likelihood of readmissions. The transition from hospital to community care settings is a particularly vulnerable period. Gaps in care, such as the absence of follow-up appointments and limited communication between healthcare professionals, can significantly contribute to patients being readmitted to the hospital. Addressing these coordination issues is essential to improving patient outcomes and reducing readmission rates.

Insufficient discharge planning and inadequate post-discharge support are also common drivers of hospital readmissions. Effective discharge planning involves ensuring patients receive appropriate follow-up care, medication management, and access to community support services. Companies like Autumna play a role in tackling these issues by providing valuable information about care homes and services for the elderly, helping bridge the gap between hospital and community care. Many patients within the NHS have complex medical and social needs, which further raise the chances of readmission. Factors such as advanced age, multiple chronic illnesses, mental health challenges, socioeconomic disadvantages, and limited support systems contribute to this complexity. To address these issues and reduce readmissions, healthcare providers must adopt personalized care plans and comprehensive approaches tailored to individual patient needs. High readmission rates not only impact patient health but also place a significant financial burden on healthcare systems. The Centers for Medicare and Medicaid Services (CMS) estimate that preventable readmissions result in billions of dollars in expenses annually. This led to the creation of the Hospital Readmission Reduction Program (HRRP), which imposes penalties on hospitals with high readmission rates. By incentivizing better care quality and promoting strategies to reduce readmissions, the HRRP aims to alleviate these financial and healthcare challenges.

1.3 AIMS AND OBJECTIVES

The aim of this study is to create a predictive model that achieves an accuracy of 70% or greater, focusing on identifying the key factors associated with hospital readmissions. This model will act as an essential resource for healthcare professionals, allowing them to

implement proactive strategies to minimize unnecessary readmissions and enhance patient care outcome.

To reach our goal, the project will focus on the following key objectives:

i. To use machine learning algorithms to create and test the predictive model, ensuring it meets the desired accuracy through rigorous validation techniques.
ii. To develop a simple interface for healthcare providers to access the model's predictions, aiding in informed decision-making regarding patient care and discharge planning.
iii. To establish a framework for regularly updating the model based on new data and changing patient needs to ensure its ongoing relevance and effectiveness.

1.4 MOTIVATION OF STUDY

Hospital readmission has become a serious challenge to the healthcare sector not only in Nigeria but worldwide, causing unnecessary increase in cost, resources and compromising patient outcomes. By leveraging machine learning (ML) to predict hospital readmissions presents a valuable opportunity for both research and practical implementation. Research indicates that predictive analytics can greatly enhance patient care by identifying individuals who are at risk and enabling timely interventions.

My project focuses on harnessing data from electronic health records (EHRs) to create a robust predictive model that aims to improve our understanding of the factors contributing to readmissions and ultimately decrease their frequency, this not only leads to cost of reductions but also significantly enhances patient experiences and outcomes. My model aims to equip healthcare providers with the insights needed to make well-informed decisions that cater to the unique needs of each patient, ultimately improving the overall standard of care. This personalized approach is essential for fostering better health results and patient satisfaction.

1.5 SCOPE OF STUDY

This project focuses on using machine learning to create a model that identify patients at high risk of hospital readmission by analyzing data from electronic health records (EHRs) which will help increase the reduction rates of hospitals readmissions in various healthcare centers, prevent unnecessary cost, enhancing patient outcomes and optimizing healthcare resource utilization.

1.6 RESEARCH METHODOLOGY

This project will adopt a systematic research methodology to develop a machine learning predictive model aimed at minimizing hospital readmissions. The approach will be divided into several crucial phases

Data Collection: Electronic health records (EHRs) will be collected from various participating healthcare facilities. This dataset will include diverse information such as patient demographics, clinical histories, treatment plans, medication changes, and post-discharge follow-up data, ensuring a broad representation of different patient populations and health conditions.

Data Preprocessing: The collected data will undergo analyzing and preprocessing to rectify missing values, outliers, and inconsistencies. This phase will focus on normalizing the data and encoding categorical variables to prepare it for analysis with machine learning algorithms.

Model Development: A variety of machine learning algorithms, including logistic regression, decision trees, random forests, and support vector machines, will be employed to construct the predictive model. The performance of the model will be thoroughly evaluated using cross- validation techniques to ensure reliability and reduce the likelihood of over-fitting.

Implementation of Insights: A user-friendly interface will be developed, enabling healthcare providers to easily access the model's predictions and insights. This tool will support clinicians in making informed decisions regarding patient care and discharge planning.

1.7 SIGNIFICANCE OF THIS PROJECT

This project is significant as it has the potential to improve hospital readmission management and enhance patient care. High readmission rates create financial burdens on healthcare systems, increasing costs and straining resources. By creating a machine learning predictive model, the project seeks to identify patients at risk of readmission, allowing for proactive interventions that can lead to customized discharge planning and better post-discharge support, ultimately minimizing unnecessary hospital visits.

Research indicates that effective discharge planning and patient education can greatly reduce readmission rates (Weiss et al., 2014). Additionally, predictive analytics have been shown to improve patient outcomes through timely interventions (Kharrazi et al., 2020; Geng et al., 2021). By prioritizing personalized care, this project not only aims to enhance patient experiences but also helps address the financial challenges faced by healthcare facilities, particularly regarding penalties from programs like the Hospital Readmission Reduction Program (HRRP) (CMS, 2021).

CHAPTER TWO
LITERATURE REVIEW

This chapter contains an overview of the predictive analysis model which is a machine learning model that will predict the outcomes of patient needed to be readmitted in a hospital. Machine learning is gradually becoming a huge impact into today's society by creating automated solutions to problems that would take a long time and resources to solve.

2.1 INTRODUCTION TO PREDICTIVE MODELING

Predictive modeling is a powerful approach within data analytics that uses advanced machine learning techniques to make informed predictions about future events. By examining patterns in historical data, predictive modeling not only forecasts likely outcomes but also reveals how future choices may impact current scenarios. This method is invaluable for organizations seeking to be proactive, as it enables them to make strategic decisions based on data-driven insights, anticipate risks or opportunities, and improve planning and performance across various applications, from finance and healthcare to logistics and customer service. Predictive modeling thus serves as a foundational tool that bridges the gap between data analysis and strategic foresight, empowering organizations to respond to challenges and optimize results in a fast-paced, data-rich world.

Research indicates that readmission rates are influenced by various factors, including patient age, existing comorbidities, and the duration of their hospital stay. Patients with a history of frequent hospitalizations are more likely to experience additional readmissions. Notably, readmission rates are particularly high among elderly individuals, especially those managing conditions like heart failure and chronic obstructive pulmonary disease (H ernandez, Greiner, and Fonarow 2010). The precise causes behind frequent readmissions for certain individuals remain unclear and require further investigation. Some research has utilized a scoring tool called LACE to assess readmission risk in clinical contexts (Walraven et al. 2010; Gruneir et al. 2011). LACE evaluates risk based on four factors: length of hospital stay (L); severity of the admission (A); patient comorbidities, measured using the Charlson comorbidity index (C); and emergency department visits (E) within the previous six months. Scores range from zero, indicating no risk, to a maximum of 19, reflecting a higher likelihood of readmission. Nonetheless, the LACE index has shown limited accuracy in predicting 30-day readmissions (Cotter et al. 2012).

Predictive models are increasingly enhancing the accuracy of readmission risk assessments, offering solutions beyond the limitations of tools like the LACE index. By integrating a wider array of clinical and demographic data, these models provide a more

precise evaluation of a patient's likelihood of readmission. Advanced analytics, often using machine learning, now identify subtle patterns in patient histories and medical profiles that contribute to readmissions. This enables healthcare providers to proactively target high-risk patients, implement timely interventions, and ultimately reduce preventable hospital returns, improving overall patient outcomes.

Predictive modeling plays a crucial role in healthcare, enabling hospitals not only to improve patient outcomes but also to manage costs more effectively, enhance resource allocation, and ensure compliance with regulatory standards aimed at lowering readmission rates. By identifying patients at high risk of readmission, these models help hospitals make informed decisions that optimize care while meeting financial and regulatory objectives.

2.2 EVOLUTION OF PREDICTIVE MODELLING

The development of predictive analytics has transformed how we approach data, moving from basic statistical analysis to advanced machine learning- driven insights. Here's a breakdown of how predictive models have evolved over time:

1. Early Statistical Models

i. **Descriptive Analytics:** Predictive analytics began with simple descriptive methods, using historical data to analyze past trends. This stage involved basic statistical techniques like averages and frequencies, giving insights into "what happened."

ii. **Linear Regression :** Linear regression marked one of the first true predictive modeling techniques, enabling simple predictions by identifying relationships between variables. Later, logistic regression was introduced to handle binary outcomes (like yes/no scenarios).

2. Rule-Based Systems and Expert Systems

i. **IF-THEN Rules:** Early predictive models often relied on rule-based systems, where experts manually crafted IF-THEN logic to handle specific tasks. While useful in controlled environments, these systems struggled to adapt to complex, evolving data.

i. **Expert Systems:** These systems integrated specific domain rules, often used in specialized fields like medical diagnostics. Although powerful in their context, they weren't flexible enough to adapt to new patterns outside the rules provided.

3. Machine Learning Models (1980s–2000s)

i. **Decision Trees and Clustering:** Decision trees and clustering algorithms introduced a new level of adaptability, allowing data to be categorized or

grouped automatically. These models helped with tasks like classification and data segmentation.

ii. **Supervised Learning:** With computing power on the rise, more advanced models like Support Vector Machines (SVMs), Naive Bayes, and neural networks became common for tasks such as fraud detection and customer churn prediction.

iii. **Ensemble Methods:** Techniques like bagging (e.g., Random Forests) and boosting (e.g., AdaBoost) improved prediction accuracy by combining multiple models, minimizing errors and boosting overall performance.

4. Big Data and Advanced Machine Learning (2010s)

i. **Deep Learning:** The introduction of deep learning techniques, such as convolutional and recurrent neural networks, changed predictive analytics dramatically, particularly for analyzing complex data like images and sequences.

ii. **Natural Language Processing (NLP):** NLP algorithms enabled predictive analysis on text data, making things like sentiment analysis and chatbots possible by processing and understanding vast volumes of text.

iii. **Big Data Predictive Models:** As data volumes grew, new approaches emerged to handle massive, real-time datasets, with distributed computing platforms like Hadoop and Spark allowing predictive analytics to scale across various industries.

5. Real-Time and Automated Predictive Models (Late 2010s – Present)

i. **Automated Machine Learning (AutoML):** AutoML has made it easier for people without extensive machine learning knowledge to build models, with features like automated feature engineering, model selection, and hyperparameter tuning.

ii. **Real-Time Predictions:** Advances in cloud computing enabled real-time predictions, vital for areas like e-commerce personalization, fraud detection, and predictive maintenance in industrial applications.

iii. **Explainable AI:** As models grew more complex, tools for explaining machine learning predictions, such as SHAP and LIME, became essential, especially in high-stakes fields like healthcare and finance.

6. Future Trends: Integrative and Ethical Predictive Models:

i. **Integrative AI:** Future models will be more integrative, blending structured data, unstructured data, and real-time data from IoT devices, social media, and other

sources for even deeper insights.

ii. **Ethical, Bias-Checked Models:** There's a growing push for fair, bias-free models, ensuring accurate and unbiased predictions in sensitive fields like hiring and criminal justice.

iii. **Causal Inference and Prescriptive Analytics:** Predictive analytics is shifting towards causal inference and prescriptive analytics, aiming to not only predict future events but also to understand and potentially influence the factors behind those events.

2.3 TYPES OF PREDICTIVE MODELING TECHNIQUE

There are several types of predictive models, each designed for specific kinds of data patterns and forecasting needs

1. Classification Model

Classification models is one of the most popular predict analysis models been used, this model analyze historical data to categorize outcomes and are widely used across industries due to their ability to be retrained on updated data, yielding valuable, actionable insights. In healthcare, classification models are particularly useful for predicting hospital readmissions. By continuously training these models with recent patient data, hospitals can more accurately assess readmission risks and implement targeted interventions. Customizable and versatile, these models empower healthcare providers to proactively improve patient outcomes and reduce readmission rates, much like they help banks or retailers address industry- specific challenges.

Classification models also output discrete and continuous predictions, which can help healthcare providers make more precise decisions about patient care.

i. **Discrete Predictions:** These are the categorical labels that indicate whether a patient is likely to be readmitted within a certain timeframe (like 30 days) or not. For example, a classification model can categorize patients as either "High Risk" or "Low Risk" for readmission based on factors like medical history, age, discharge details, and follow-up care. Here, "High Risk" and "Low Risk" are the discrete predictions, providing healthcare teams with a straightforward way to identify patients who may need extra post-discharge support to prevent a return to the hospital.

ii. **Continuous Predictions:** In hospital readmission modeling, classification models can also assign continuous probabilities, known as “confidence scores”, to indicate the likelihood of a patient being readmitted. For example, a model might assign a probability of 0.78 to a patient being readmitted within

30 days, meaning there is a 78% likelihood of readmission and a 22% likelihood of no readmission. These confidence scores allow healthcare providers to assess the strength of the prediction, enabling them to prioritize care for patients with the highest readmission risk and tailor interventions accordingly.

2. Clustering Model

Clustering models are highly effective for grouping patients based on shared characteristics that may influence their likelihood of returning to the hospital. By analyzing data such as patient demographics, medical conditions, length of stay, and post-discharge support, clustering models divide patients into groups or "clusters" that reveal common patterns in readmission risk factors.

Types of Clustering in Readmission Prediction

i. **Hard Clustering**: In hard clustering, each patient is assigned to a single cluster, helping healthcare providers classify patients strictly based on specific risk factors. For instance, one cluster may contain elderly patients with chronic conditions and high readmission rates, while another may include younger patients with lower risk. This clear-cut classification helps hospitals design tailored intervention strategies based on specific patient profiles.

ii. **Soft Clustering**: Soft clustering assigns a probability of belonging to each cluster, offering a more flexible approach. For example, a patient might have a 60% probability of fitting into a "High Risk" cluster (frequent readmissions due to unmanaged chronic conditions) and a 40% probability of belonging to a "Moderate Risk" cluster (less severe cases). This probabilistic assignment can help clinicians assess patients with overlapping risk factors and implement customized follow-up plans accordingly.

3. Outliers Model

Unlike classification models, the outlier model focuses on identifying unusual or anomalous data points within a dataset, which may not fit into standard categories or expected patterns. In the context of hospital readmission, outlier models are particularly useful for detecting a typical cases that may indicate higher readmission risks or specific issues in patient care.

i. **Anomaly Detection in Hospital Readmission Prediction:** Outlier models can flag patients whose readmission risks or health conditions deviate significantly from the typical patterns observed within the hospital's patient population. For example, if

most patients with a certain condition have low readmission rates, but a few exhibit frequent or unexpected returns to the hospital, an outlier model would identify these cases as anomalies. This can prompt further investigation into specific factors, such as underlying health issues or gaps in discharge planning, that may be driving these atypical readmissions.

ii. **Value of Anomaly Detection in Healthcare:** Detecting outliers allows hospitals to address unique risks that might not be captured by standard models, making it easier to intervene with specialized care or follow-up support for high-risk patients. Moreover, outlier models help ensure that predictive analytics stay robust and accurate by identifying any data inconsistencies or anomalies in readmission patterns. Just as in finance or retail, where anomaly detection helps prevent fraud or losses, in healthcare, these models help hospitals improve patient outcomes by highlighting irregular cases that might benefit from customized, high-priority attention.

4. Forecast Model

The forecast model is a widely used predictive analytics tool that predicts future metric values by analyzing patterns and trends in historical data. It is particularly effective for predicting numerical values where historical data may be incomplete or missing. One of the key strengths of forecast models is their ability to manage multiple variables simultaneously, making them highly versatile and valuable in predictive modeling. In healthcare, forecast models can play a critical role in managing hospital readmissions. For example, hospitals can use forecast analytics to predict future readmission rates based on factors like seasonal trends, patient demographics, or specific health conditions. By forecasting readmission volumes, healthcare administrators can better allocate resources, such as staffing, bed availability, and follow-up care services, to manage anticipated patient demand. Just as a call center uses forecasting to predict support call volumes or a retailer forecasts inventory needs, hospitals can rely on forecast models to anticipate patient needs and optimize care delivery to reduce readmission rates and improve patient outcomes.

5 Time Series Model

Time series predictive models are designed to analyze datasets where time sequences serve as the primary input, allowing these models to capture and forecast trends over specific periods. By using historical data points, often spanning previous years, time series models generate numerical predictions that reveal underlying trends and seasonal patterns. Unlike traditional methods, time series models excel in their flexibility: they can simultaneously

produce forecasts across multiple regions or projects or concentrate on a single variable or area based on an organization's specific requirements. Time series predictive models can be relevant in managing hospital readmission rates, as they enable healthcare providers to track and analyze changes in readmission patterns over time. By using historical data such as past admissions, patient demographics, comorbidities, and seasonal factors these models can predict future readmission rates, offering valuable insights into when and why patients are likely to return. For instance, time series models allow hospitals to forecast readmission trends for specific departments, such as cardiology or pulmonology, where readmissions for conditions like heart failure and chronic obstructive pulmonary disease are common. These models can highlight periods of increased readmission risk, such as during flu season, and help identify external factors that may influence patient outcomes. This level of forecasting enables hospitals to prepare proactively, optimizing staff allocation, resources, and discharge planning efforts to reduce readmission rates effectively. By incorporating time-sensitive variables and fluctuations, time series models go beyond traditional risk assessments, providing a nuanced view of readmission risks over time. This allows hospitals to make data-driven decisions to improve patient care, minimize costs associated with preventable readmissions, and align with regulatory standards focused on reducing hospital readmission rates.

2.4 MACHINE LEARNING ALGORITHIMS

1. What is an algorithm?

An **algorithm** is a step-by-step procedure designed to perform a specific task or solve a particular problem. It is a well-defined sequence of instructions that takes an input, processes it, and produces an output in a finite number of steps. It provide a blueprint for solving problems systematically and efficiently. They are used in various fields, from computing and mathematics to everyday life (e.g., a recipe for cooking or a set of directions for getting somewhere).now this takes us into machine learning algorithms.

2. What are Machine Learning Algorithm

In machine learning, algorithms serve as the fundamental mathematical and computational structures that allow systems to learn from data, make predictions, and carry out tasks without requiring explicit programming for each situation. Unlike traditional programming, where every action and result must be manually written out, machine learning algorithms enable a system to identify patterns, adjust over time, and make decisions based on the input data. This capacity to "learn" from the data is what distinguishes machine learning from traditional programming approaches.

3. Types of Machine Learning Algorithm

These Machine learning algorithms can be broadly categorized into three main types: supervised, semi-supervised, unsupervised learning approach.

i. Supervised Learning Approach

In supervised learning, the model learns by example. Here, it's given a dataset with known inputs and outputs, and it works to discover how to link those inputs to the expected results. While the correct answers are known, the model identifies patterns, learns from the data, and starts making predictions. The model's predictions are then refined based on feedback, and this cycle repeats until the model reaches a high level of accuracy supervised learning can help models detect important risk patterns within past patient data, steadily enhancing its accuracy in forecasting the chances of readmission.

ii. Semi-Supervised Learning Approach

Semi-supervised learning builds on supervised learning by incorporating both labeled and unlabeled data. Labeled data provides meaningful tags that help the algorithm interpret information, while unlabeled data lacks these tags. In predicting hospital readmissions, this approach allows the model to learn from labeled patient records and use that knowledge to identify patterns and label new, unlabeled patient data, enhancing its predictive accuracy even with limited labeled information.

iii. Unsupervised Learning Approach

In unsupervised learning, the machine learning algorithm examines data to find patterns without any guidance or answer key. Instead of following instructions, it discovers correlations and relationships by analyzing the data on its own. In the context of hospital readmission prediction, an unsupervised learning model would explore large sets of patient data, identifying patterns that help describe the data's structure. It might, for example, group patients into clusters based on similar health risk factors or categorize them in a way that highlights meaningful trends. Over time, as it processes more data, the model's ability to draw insights and identify potential readmission risks becomes increasingly refined.

2.4 BENEFITS OF USING PREDICTIVE MODELLING IN HOSPITAL READMISSION

Machine learning models bring significant benefits to predicting hospital readmissions, advancing patient care quality and healthcare efficiency in multiple ways:

i. **Enhanced Accuracy**: These models can precisely identify high-risk patients, allowing hospitals to take early action to prevent avoidable readmissions, leading to better patient outcomes.

ii. **Customized Patient Care**: Machine learning considers diverse patient-specific

factors—such as medical history, existing health conditions, and social factors—enabling more personalized care and effective health management.

iii. **Resource Optimization**: By focusing on high-risk patients, hospitals can direct resources more efficiently, prioritizing post-discharge support and follow-ups for those who need it most, reducing resource strain.

iv. **Cost Efficiency**: Reducing readmissions results in lower healthcare costs and helps hospitals avoid financial penalties tied to high readmission rates.

v. **Improved Patient Satisfaction**: Targeted interventions help patients avoid unnecessary hospital stays, improving their experience and trust in the healthcare system.

vi. **Scalability and Flexibility**: Machine learning models are highly adaptable, allowing hospitals to integrate new data, such as real-time metrics from wearable devices, for more comprehensive and dynamic predictions.

vii. **Insightful Risk Identification**: Machine learning models analyze patterns in readmissions, uncovering previously unrecognized risk factors that guide proactive care strategies for patients with similar health profiles.

viii. **Continuous Learning and Adaptation**: These models are capable of learning from new data, allowing them to adapt to evolving healthcare needs and stay current with emerging trends

2.5 RELATED WORKS

Futoma et al. (2015) conducted a comprehensive study comparing predictive models for early hospital readmissions. They evaluated machine learning algorithms, including logistic regression, random forests, support vector machines, and neural networks, using data from 3.3 million hospital visits. Their findings showed that modern machine learning methods outperformed traditional approaches, with penalized logistic regression and random forests performing well. The study also highlighted the importance of condition-specific models for improving prediction accuracy.

Jamei et al. (2017) proposed a study using artificial neural networks to predict 30-day hospital readmission risk, analyzing over 323,000 admissions. They compared neural networks with logistic regression, demonstrating superior performance in discrimination and calibration. By incorporating diverse patient data, their approach effectively captured complex patterns, improving readmission risk stratification.

Rajkomar et al. (2018) conducted a study exploring the use of deep learning models, such as neural networks, to predict 30-day hospital readmissions using EHR data. Their findings

indicated that deep learning models outperformed traditional machine learning algorithms in prediction accuracy, particularly for high-dimensional and complex healthcare datasets.

Mahmoudi et al. (2020) conducted a systematic review on electronic medical record-based hospital readmission prediction models. Analyzing 41 studies, they found significant variability in model development and validation. Their review highlighted gaps in external validation and data transparency, limiting cross-study comparisons. While EMR-based models showed promise, inconsistent variable selection hindered clinical implementation. They proposed standardized guidelines to improve reproducibility and utility.

Khalid et al. (2022) developed advanced readmission prediction models using LSTM networks on healthcare insurance data. Their study showed how recurrent neural networks capture temporal patterns in patient histories, improving prediction accuracy. Using large-scale claims data, they demonstrated that LSTM models outperformed traditional machine learning methods. Their work addressed challenges in processing sequential healthcare events, highlighting deep learning's potential for extracting meaningful insights from complex patient trajectories.

Kawakami et al. (2023) introduced a multi-omics approach to predicting patient recurrence time, offering insights relevant to hospital readmission prediction. While centered on cancer recurrence, their study demonstrated how integrating diverse data sources with advanced machine learning techniques improves predictive accuracy. Using Random Forest and XGBoost, they achieved high performance (AUC > 0.85) and identified key biomarkers linked to recurrence risk. Their use of ensemble learning and multi-modal data analysis presents a cutting-edge methodology adaptable for hospital readmission prediction.

Cai et al. (2024) conducted a comprehensive study comparing machine learning algorithms for predicting readmissions in heart failure patients. Evaluating ten models, including logistic regression, SVMs, random forests, and neural networks, they analyzed 38,888 admissions using cross-validation. Their results showed that ensemble methods, particularly XGBoost (AUC 0.857), outperformed other approaches. The study also identified key predictors of readmission, such as prior hospitalizations, comorbidities, and lab values, offering insights for both methodological improvements and clinical use.

A Table Summary of The Related Works

Authors	Year	Title/Framework	Summary	Research
Futoma et al.	2015	Machine Learning for Early Readmission Prediction	ML models outperformed traditional methods.	Predictive modeling for early hospital readmissions
Jamei et al.	2017	Neural Networks for 30-Day Readmission Risk	Neural networks outperformed logistic regression.	AI-based risk stratification
Rajkomar et al.	2018	Deep Learning for Readmission Prediction	Deep learning improved accuracy for complex EHR data.	EHR-driven deep learning models
Mahmoudi et al.	2020	EMR-Based Readmission Prediction Models	Identified inconsistencies and proposed guidelines.	Systematic review of EMR-based models
Khalid et al.	2022	LSTM Networks for Readmission Prediction	LSTM outperformed traditional ML using insurance data.	Temporal modeling using RNNs
Kawakami et al.	2023	Multi-Omics and ML for Recurrence Prediction	Multi-modal data integration improved prediction.	Multi-omics and ensemble learning approaches
Cai et al.	2024	ML Comparison for Heart Failure Readmissions	XGBoost (AUC 0.857) performed best.	Comparative analysis of ML techniques

CHAPTER THREE
SYSTEM ANALYSIS AND METHODOLOGY

This chapter presents a framework for the development process, along with an overview of the system analysis and methodology

3.1 System Analysis and Design

System analysis is a critical phase in the software development life cycle, involving a comprehensive examination of either an existing system or the requirements for a new one. The goal is to fully understand the system's components, features, processes, and interactions to develop efficient and effective solutions. This phase is essential for defining, documenting, and refining the system's requirements.

There are various methods of system analysis and design, with two primary approaches being:

i. Object-Oriented Analysis and Design (OOAD)
ii. Structured System Analysis and Design (SSAD)

For this project, the Object-Oriented Analysis and Design (OOAD) method was chosen. The primary objective of OOAD is to create a well-organized software solution that addresses practical issues. The methodology emphasizes understanding the problem domain, designing modular and reusable components, and ensuring the system is scalable, maintainable, and adaptable to evolving requirements.

3.2 Analysis of Existing System

The current system for predicting hospital readmissions is traditional and primarily relies on manual processes to track and analyze patient data. Healthcare providers use paper-based records and spreadsheets to gather patient information, such as medical history, previous hospitalizations, and demographic details, which are then used to assess the risk of readmission.

This method is time-consuming, prone to human error, and inefficient, particularly when managing large volumes of data. It also lacks integration with modern data processing tools, making it difficult to update and scale as the amount of patient data grows. There is a significant risk of missing or inaccurate data, which can lead to incorrect predictions and suboptimal care planning.

Additionally, healthcare providers often rely on subjective judgment when assessing a patient's risk of readmission, which can lead to inconsistencies and variability in the quality of care provided. This system is also limited in its ability to adapt to new medical knowledge or evolving patient needs, as updates and changes require manual intervention and are not dynamically integrated into the decision-making process.

Given these limitations, transitioning to a more modern, data-driven approach—using machine learning models to predict readmissions based on structured EHR data—offers a more accurate, efficient, and scalable solution. This approach will allow for more consistent predictions, improved patient outcomes, and better utilization of healthcare resources.

3.3 Problems of Existing models

The problems outlined in Chapter One regarding the existing system for hospital readmissions closely align with the challenges of fragmented care in healthcare. The lack of an integrated system results in disconnected and inefficient processes, making it difficult to maintain comprehensive and accurate patient records. This fragmentation mirrors the issues in traditional hospital readmission models, where gaps in care coordination can lead to incomplete patient histories, missed follow-ups, and inadequate post-discharge management.

Similarly, reliance on manual processes, such as physical documentation and paper-based tracking, introduces the risk of human error, further exacerbating fragmented care. Without a streamlined approach, hospitals may struggle to track patient progress effectively, leading to preventable readmissions and compromised patient outcomes. Addressing these challenges requires a more cohesive system that enhances care continuity, improves data accessibility, and supports better decision-making to reduce hospital readmission rates.

Moreover, the lack of real-time updates in the traditional system mirrors the limitations of outdated predictive models, where data is not continuously updated, reducing the accuracy of predictions. The traditional model is also limited in its predictive capabilities, relying on manual tracking and subjective judgment, which is akin to existing systems that use simple, rule-based methods that fail to account for the complex nature of healthcare data. Scalability issues in the paper-based attendance system reflect similar challenges in older predictive models that struggle to handle larger datasets and accommodate growing patient populations. The lack of personalization in the current system, where generic passes and attendance records are used, mirrors the issue in predictive models that do not account for individual patient needs. Additionally, the data privacy and security concerns in traditional systems, such as the potential for forgery and tampering, are similar to the risks faced by older models in healthcare, where sensitive patient data is not adequately protected. The inadequate integration of traditional methods with modern digital tools results in inefficiencies, similar to how outdated predictive models fail to integrate seamlessly with newer healthcare technologies. Finally, the limited adaptability of the traditional system to evolving medical needs and practices mirrors the slow adaptation of older models to new medical knowledge, requiring more flexible and adaptive approaches for improved accuracy and care. These problems highlight the need for modern, automated, and data-driven solutions that can

overcome the inefficiencies and risks of traditional systems.

3.4 Overview of the Model

The model aims to predict hospital readmissions using Electronic Health Records (EHR) data to identify high-risk patients. It analyzes factors like patient demographics, medical history, previous hospitalizations, and other health data to predict which patients are at risk of being readmitted. The goal is to help healthcare providers take proactive steps to intervene and provide targeted care for high-risk patients, improving patient outcomes and reducing readmissions. The model uses machine learning techniques, ensuring that it is scalable, accurate, and adaptable as patient data and medical practices evolve. This approach offers a more efficient, data-driven solution compared to traditional systems.

3.5 System Architecture of the Predictive Model

The architecture of the hospital readmission prediction model is structured to be both simple and effective, As shown in Figure 1.0 it begins with gathering patient information, including demographics, medical history, and prior hospitalizations, from Electronic Health Records (EHR). This data is then preprocessed through steps such as cleaning, normalization, and encoding to ensure uniformity and readiness for analysis. Key factors, like age and length of hospital stay, are identified as influential features for predicting readmissions. A machine learning algorithm, such as Logistic Regression is trained on the processed data to distinguish between patients at risk of readmission and those who are not. After training, the model is used to assess new patient data, generating either a risk score or a binary outcome. The predictions are presented to healthcare providers in an accessible format, enabling informed interventions to lower readmission rates and enhance patient care.

Figure 3.1 Three Phases for Building The Model

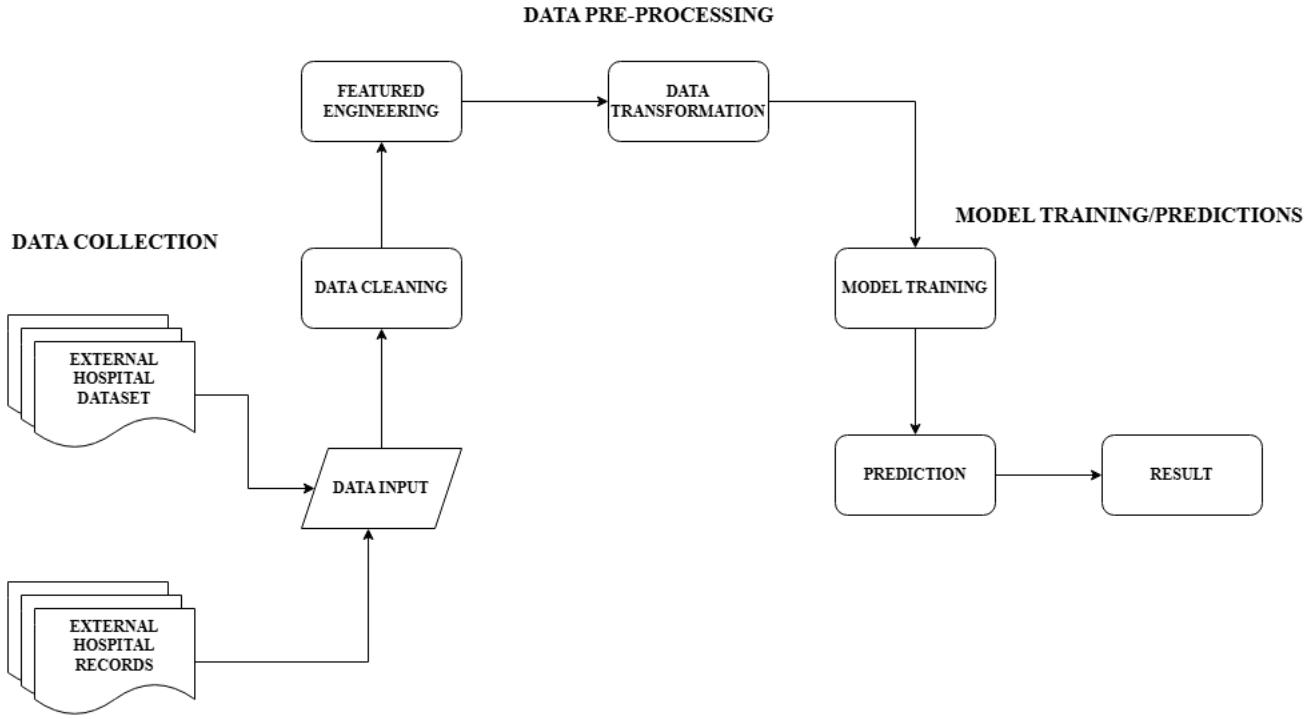

Figure 3.2 System architecture of the Predictive Model

The system for predicting hospital readmissions integrates multiple components to efficiently process data, develop predictive models, and deliver predictions. A key element is the data ingestion pipeline, which ensures the secure collection and processing of electronic health record (EHR) data while adhering to healthcare interoperability standards like Health Level Seven (HL7) and Fast Healthcare Interoperability Resources (FHIR). Once gathered, the data is stored in a secure, scalable database. Relational databases, such as PostgreSQL or MySQL, are used for structured data like demographics and vitals, while unstructured data is managed by NoSQL databases like MongoDB.

Feature engineering and model training represent the analytical backbone of the system, converting raw data into useful features and constructing predictive models. These models are deployed through RESTful APIs or gRPC protocols to enable smooth communication across components. Lastly, the monitoring and feedback mechanism continuously evaluates the model's performance using metrics like Area Under the Receiver Operating Characteristic Curve (AUC-ROC) and precision-recall. This system is equipped to retrain the model as new data becomes available, ensuring long-term accuracy and relevance.

3.6 Components diagram

The system's components can be visualized as a flow of interconnected modules. The frontend consists of a user-friendly dashboard designed for clinicians to interact with and view patient readmission predictions. On the backend, an ETL (Extract, Transform, Load) pipeline processes raw data into usable formats. This data feeds into the machine learning (ML) model serving layer, which handles model inference. An API gateway connects the backend services to the frontend while managing access and ensuring security. A robust database stores both raw patient data and prediction results, and a model registry is maintained to manage multiple versions of predictive models, facilitating updates and rollback as needed.

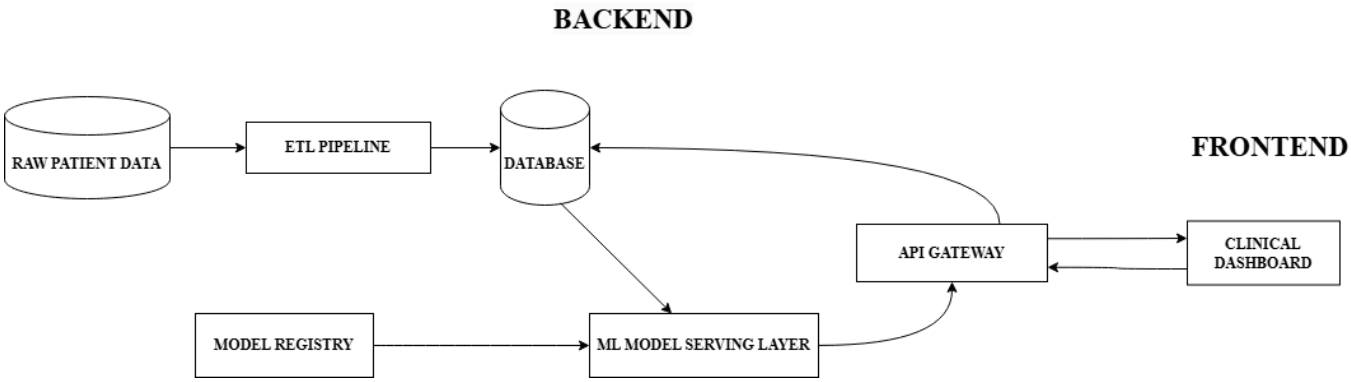

Figure 3.3 Component Diagram of the Model.

3.7 Methodology

This encompasses of several key processes to ensure a robust and effective system. It begins with data collection and preprocessing, where electronic health record (EHR) data, including demographics, vitals, and diagnoses, is gathered and cleaned to address issues like missing values and inconsistencies. Exploratory Data Analysis (EDA) follows, identifying trends and insights that guide feature engineering and model selection. The process continues with model development and validation, using advanced machine learning techniques and evaluation metrics to ensure reliability. Finally, the methodology includes deployment and post-deployment monitoring to maintain performance, adapt to new data, and ensure long-term effectiveness in real-world healthcare applications.

3.8 Data Collection and Pre-processing

The success of a predictive system relies heavily on the quality of its input data. For this project, EHR data, including demographics, vitals, medications, and diagnoses, is collected from healthcare providers. Data cleaning processes address common issues such as missing values, duplicates, and inconsistent formats. Missing data can be imputed using statistical

techniques or machine learning models, while normalization ensures that numerical values are scaled appropriately for model input. To protect patient privacy, all data is anonymized through techniques like hashing or pseudonymization, ensuring compliance with regulations such as HIPAA and GDPR.

3.9 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) plays a critical role in understanding and interpreting the dataset. This step involves identifying patterns and relationships through visualizations and statistical summaries. By examining trends such as readmission rates across different medical conditions or demographic groups, the process helps pinpoint valuable insights that guide feature selection and model development. Graphs and charts, such as histograms, scatter plots, and box plots, make it easier to identify correlations, anomalies, and outliers. Additionally, EDA allows researchers to refine their hypotheses and establish realistic expectations for model performance. This foundational step ensures that the data is well-understood before proceeding to complex modeling techniques.

3.10 Model Development

The model development begins with establishing a baseline using simple algorithms such as logistic regression or decision trees. These models provide a benchmark for evaluating more advanced techniques. For this project, ensemble models like Random Forest and XGBoost are explored for their ability to handle complex, non-linear relationships in data. In cases where temporal data is involved, deep learning models such as LSTMs (Long Short-Term Memory) or Transformers can capture sequential dependencies effectively. The models are evaluated using metrics such as AUC-ROC and precision-recall, which measure their ability to correctly predict readmissions without excessive false positives or negatives.

3.11 Model Validation

Robust validation techniques are essential for assessing the reliability of predictive models. Cross-validation, where the dataset is split into multiple subsets for training and testing, ensures that the model generalizes well to unseen data. Stratified sampling maintains the proportional distribution of classes, such as readmitted and non-readmitted patients, across training and testing sets. This prevents skewed results and ensures a fair evaluation.

3.12 Deployment

Once validated, the model is prepared for deployment. Model artifacts are saved in formats such as .pkl or .h5, which can be easily loaded for inference. Deployment platforms like Flask or FastAPI are used to serve the model via APIs, allowing external systems to query predictions. To ensure scalability, the system is containerized using Docker and orchestrated with Kubernetes. This setup enables the system to handle increased workloads and facilitates

seamless updates.

3.13 Post-Deployment Monitoring

Post-deployment monitoring tracks the system's performance in real-world settings. Metrics such as prediction accuracy and latency are logged and analyzed to detect issues like model drift, where the data distribution changes over time, leading to reduced model effectiveness. Tools such as Prometheus and Grafana provide real-time dashboards for monitoring system health, while MLFlow helps manage and version models. A feedback loop allows new data to be incorporated into the training pipeline, ensuring that the model evolves with changing healthcare trends

3.14 TOOLS AND TECHNOLOGIES

I. Data Handling

Effective data handling forms the backbone of any data-driven project. For processing and analyzing structured datasets, Pandas and NumPy are essential libraries. Pandas simplifies tasks like data cleaning, manipulation, and aggregation, while NumPy provides powerful tools for numerical computations, especially with multi-dimensional arrays and matrices. When working with databases, PostgreSQL or SQLite are ideal for managing structured, relational data, offering robust querying capabilities and scalability. For projects requiring flexibility or dealing with unstructured data, MongoDB**,** a NoSQL database, provides a schema-less architecture that supports dynamic data formats, making it ideal for scenarios where data structure evolves over time.

II. Visualization

Data visualization is key to understanding trends and insights during exploratory data analysis (EDA). Jupyter Notebook is a versatile environment for initial prototyping and visual exploration, offering an interactive and user-friendly interface. Libraries like Matplotlib and Seaborn complement this by enabling the creation of detailed visualizations, from simple plots to complex statistical charts. Matplotlib provides granular control over plots, while Seaborn simplifies creating aesthetically pleasing and informative visualizations with minimal code.

III. Model Development

Building and evaluating predictive models requires specialized tools and libraries**.** Scikit-learn is the go-to library for traditional machine learning, offering pre-implemented algorithms, metrics, and preprocessing utilities. For deep learning tasks, TensorFlow and PyTorch provide powerful frameworks for creating and training complex neural networks. TensorFlow is

known for its scalability and production-ready capabilities, while PyTorch is favored for its flexibility and ease of experimentation, making it suitable for research and rapid prototyping.

IV. Web Interface

Interactivity in presenting machine learning models or analytics results is crucial for user engagement. Tools like Streamlit and Gradio allow for the quick development of interactive interfaces, enabling users to interact with models without extensive front-end expertise. For more polished analytical dashboards, Dash combines the power of Python with responsive web design. When customizability and scalability are required, React.js can be paired with a backend (e.g., FastAPI) to create sophisticated web applications tailored to specific use cases.

V. API Development

APIs facilitate seamless integration between different systems and enable data exchange. FastAPI is a high-performance, modern framework for developing APIs. Its asynchronous capabilities make it ideal for handling large-scale hospital data efficiently, supporting real-time interactions and faster response times. With built-in support for data validation and OpenAPI documentation, FastAPI simplifies development and ensures robust, production-ready APIs.

VI. Open-Source Capability

Open-sourcing a project fosters collaboration and wider adoption. Hosting code repositories on GitHub allows contributors to review, suggest improvements, and extend functionalities. GitHub also provides tools for managing version control, tracking issues, and automating workflows, ensuring that the project remains organized and accessible to the community.

VII. Deployment

Deploying models and interfaces to a live environment ensures accessibility for end-users. Vercel is a popular platform for deploying web applications, offering a streamlined process for hosting and scaling projects. Its integration with GitHub allows for automatic deployment on code updates, while its serverless architecture ensures high performance and reliability for modern web applications.

CHAPTER FOUR
IMPLEMETATION AND TESTING

This chapter discusses the process I went through in building and implementing the predictive model. It covers the exploratory data analysis (EDA) I conducted, the technologies used, and the steps taken to develop and integrate the model into a functional API, along with its interface for user interaction.

4.0 SOFTWARE IMPLEMENTATION TOOLS

1. **Implementation Languages**

   The project is developed using TypeScript, a superset of JavaScript that introduces static typing, improving code quality, maintainability, and error detection during development. HTML (HyperText Markup Language) serves as the foundation for structuring content on the web, ensuring a well-organized and semantically meaningful layout. CSS (Cascading Style Sheets) is responsible for the visual design and aesthetics of the web application, controlling layout, colors, fonts, and responsiveness. Python is also used for backend development, particularly in data processing, machine learning, and automation tasks, providing a powerful and versatile programming environment.

2. **Frameworks**

   This project utilizes React.js, a powerful JavaScript library for building interactive and dynamic user interfaces through a component-based architecture that promotes reusability and efficiency. On the backend, Express.js is employed as a fast and lightweight web framework for Node.js, simplifying backend development by handling HTTP requests, middleware, and routing with ease.

3. **Development Tools**

   To enhance productivity, the development process is carried out using Visual Studio Code (VS Code), a feature-rich code editor with built-in support for TypeScript, debugging, extensions, and seamless integration with Git for version control. Additionally, Postman is used as an API development and testing tool, enabling efficient debugging, request simulation, and performance validation of backend services.

4.1. Dataset Overview

For this project, I obtained a healthcare dataset from an open-source platform. This dataset serves as the foundation for building and testing the predictive model. It contains key patient information, including demographics, medical history, and hospital visit details. Some of the

critical features include age brackets, length of hospital stay, number of procedures and medications administered, previous inpatient and outpatient visits, emergency room visits, and the medical specialty of the admitting physician. Additionally, it includes diagnostic codes for primary and secondary conditions, glucose and A1C test results, medication changes, and whether a patient was readmitted. By analyzing these variables, I was able to extract meaningful insights and develop a robust model to predict hospital readmissions, Below is a summary of the features in the dataset:

- **age** – Age bracket of the patient
- **time_in_hospital** – Length of hospital stay (1 to 14 days)
- **n_procedures** – Number of procedures performed during the hospital stay
- **n_lab_procedures** – Number of laboratory tests conducted
- **n_medications** – Number of medications administered
- **n_outpatient** – Number of outpatient visits in the past year
- **n_inpatient** – Number of inpatient visits in the past year
- **n_emergency** – Number of emergency room visits in the past year
- **medical_specialty** – Specialty of the admitting physician
- **diag_1** – Primary diagnosis (e.g., circulatory, respiratory, digestive, etc.)
- **diag_2** – Secondary diagnosis
- **diag_3** – Additional secondary diagnosis
- **glucose_test** – Glucose serum test result (high, normal, or not performed)
- **A1Ctest** – A1C level test result (high, normal, or not performed)
- **change** – Whether there was a change in diabetes medication (yes or no)
- **diabetes_med** – Whether a diabetes medication was prescribed (yes or no)
- **readmitted** – Whether the patient was readmitted (yes or no)

4.2. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) helps in understanding the dataset before developing a model. It includes summarizing the data, identifying missing values, visualizing distributions, and analyzing relationships between variables, I carried out the EDA using Jupyter Notebook, which provided an interactive environment for data exploration, visualization, and analysis. Through this process, I was able to gain insights into the dataset, detect any inconsistencies, and make necessary improvements to enhance the model's performance.

To begin the analysis, I first loaded the dataset using the Pandas library with df = pd.read_csv('hospital_readmissions.csv'), which stores the data in a DataFrame for easier manipulation. To get a quick overview, I used df.shape to check the number of rows and

columns and df.head() to display the first five rows. These steps helped me understand the dataset's structure and contents before proceeding with further analysis.

```
#  To Load the dataset
df = pd.read_csv('hospital_readmissions.csv')

# Overview of the dataset
print("Dataset Shape:", df.shape)
print("First 5 Rows:")
display(df.head())  # for better visualization

# Check for missing values
print("Missing Values:")
print(df.isnull().sum())

# Data types and general info
print("\nDataset Info:")
print(df.info())
```

Figure 4.1 Loading the Dataset.

To check for missing values, I used df.isnull().sum(), which counts the number of missing entries in each column. Identifying these columns was important for deciding how to handle missing data later. Next, I used df.info() to get details about the dataset, including column names, data types, and the number of non-empty values in each column. This step helped me determine whether columns were numerical, categorical and if they required reformatting.

```
Dataset Shape: (25000, 17)
First 5 Rows:
```

	age	time_in_hospital	n_lab_procedures	n_procedures	n_medications	n_outpatient	n_inpatient	n_emergency	medical_specialty	diag_1	diag_2	diag_3
0	[70-80)	8	72	1	18	2	0	0	Missing	Circulatory	Respiratory	Other
1	[70-80)	3	34	2	13	0	0	0	Other	Other	Other	Other
2	[50-60)	5	45	0	18	0	0	0	Missing	Circulatory	Circulatory	Circulatory
3	[70-80)	2	36	0	12	1	0	0	Missing	Circulatory	Other	Diabetes
4	[60-70)	1	42	0	7	0	0	0	InternalMedicine	Other	Circulatory	Respiratory

Figure 4.2 Collecting of information

From the above information, it shows that the dataset contains 2,500 rows and 17 columns and as observed from the table, numerical columns like "age," "time_in_hospital," and "n_lab_procedures" capture essential patient statistics, while categorical columns such as "medical_specialty" and "diag_1" to "diag_3" provide descriptive information about medical categories and diagnoses. Missing values, particularly in columns like "medical_specialty," may need careful handling to ensure analysis accuracy. The next phase will focus on addressing these missing values, identifying trends or anomalies, and preparing the data for further analysis.

```
Missing Values:
age                  0
time_in_hospital     0
n_lab_procedures     0
n_procedures         0
n_medications        0
n_outpatient         0
n_inpatient          0
n_emergency          0
medical_specialty    0
diag_1               0
diag_2               0
diag_3               0
glucose_test         0
A1Ctest              0
change               0
diabetes_med         0
readmitted           0
dtype: int64
```

Figure 4.3 using df.isnull().sum to get missing values

The analysis confirms that the dataset is complete, with no missing values in any of the columns. Each column has been checked for null entries, ensuring that all data points are accounted for, eliminating the need for imputation. This makes the dataset ready for further analysis and modeling. However, it is still essential to check for potential issues such as inconsistencies or outliers. Outliers are values that are very different from the rest of the data. They can happen because of mistakes, measurement errors, or just natural differences. For example, if most patients stay in the hospital for 3 to 7 days**,** but a few stay for 50 days**,** those unusual values are outliers. Finding these outliers is important because they can affect the accuracy of the analysis. As I go through the data, I will check for outliers and decide whether to keep or remove them.

4.1. Distribution of Hospital Rates

To visualize the distribution of hospital readmission rates, I used Seaborn and Matplotlib. The following code generates a count plot, setting the figure size to 6x4 and using a 'muted' color palette to display the frequency of each readmission category.

```
# Distribution of the target variable (readmitted)
plt.figure(figsize=(6, 4))
sns.countplot(x='readmitted', data=df, palette='muted')
plt.title("Readmission Counts")
plt.xlabel("Readmitted")
plt.ylabel("Count")
plt.show()
```

Figure 4.4 code to identify imbalances

This visualization helps identify any imbalances in the dataset, such as whether one category is overrepresented or underrepresented. It also provides insight into how readmission rates are distributed, which can guide further analysis and improve the design of a predictive model.

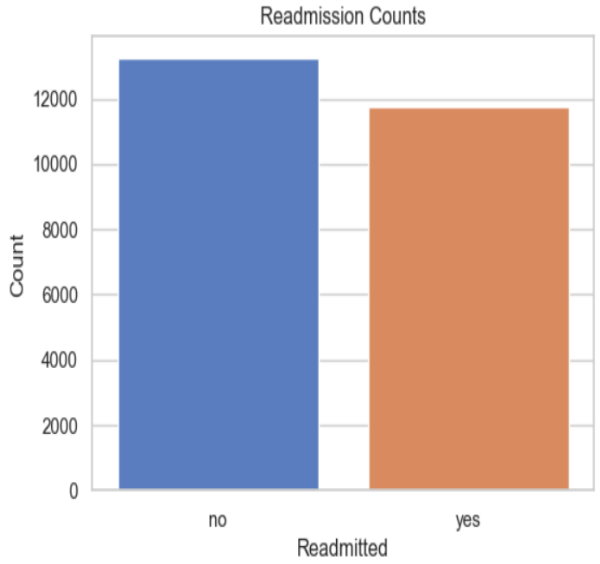

Figure 4.5 distribution chart of the hospital readmission

The chart above shows the distribution of hospital readmissions. Around 13,000 patients were not readmitted (represented with by blue bar), while approximately 12,000 patients were readmitted (represented with by orange bar). This indicates a fairly balanced distribution, with slightly more patients not being readmitted.

Once I was done with the Exploratory Data analysis, then I moved unto the next step in developing this model.

4.2. Data pre-processing

The Data pre-processing stage plays a very important role in preparing raw patient data for my hospital readmission prediction model by converting it into a machine-learning-compatible format. The preprocess_data(df) function takes a pandas DataFrame as input and first addresses missing values by replacing explicit "Missing" entries with NumPy's `nan` values and removing incomplete rows. The target variable 'readmitted' is then transformed from categorical ('yes'/'no') to numerical values, mapping 'yes' to 1 and 'no' to 0 for model compatibility. Age data is standardized using a custom convert_age function, ensuring

consistency. To handle categorical features, I identify object-type columns, apply LabelEncoder to convert them into numerical representations, and store the encoders for potential inverse transformation. This streamlined pre-processing pipeline enhances data quality, ensures uniformity, and optimally prepares the dataset for effective machine learning model training while maintaining result interpretability.

The following image illustrates the implementation of our pre-processing pipeline, demonstrating each step described above:

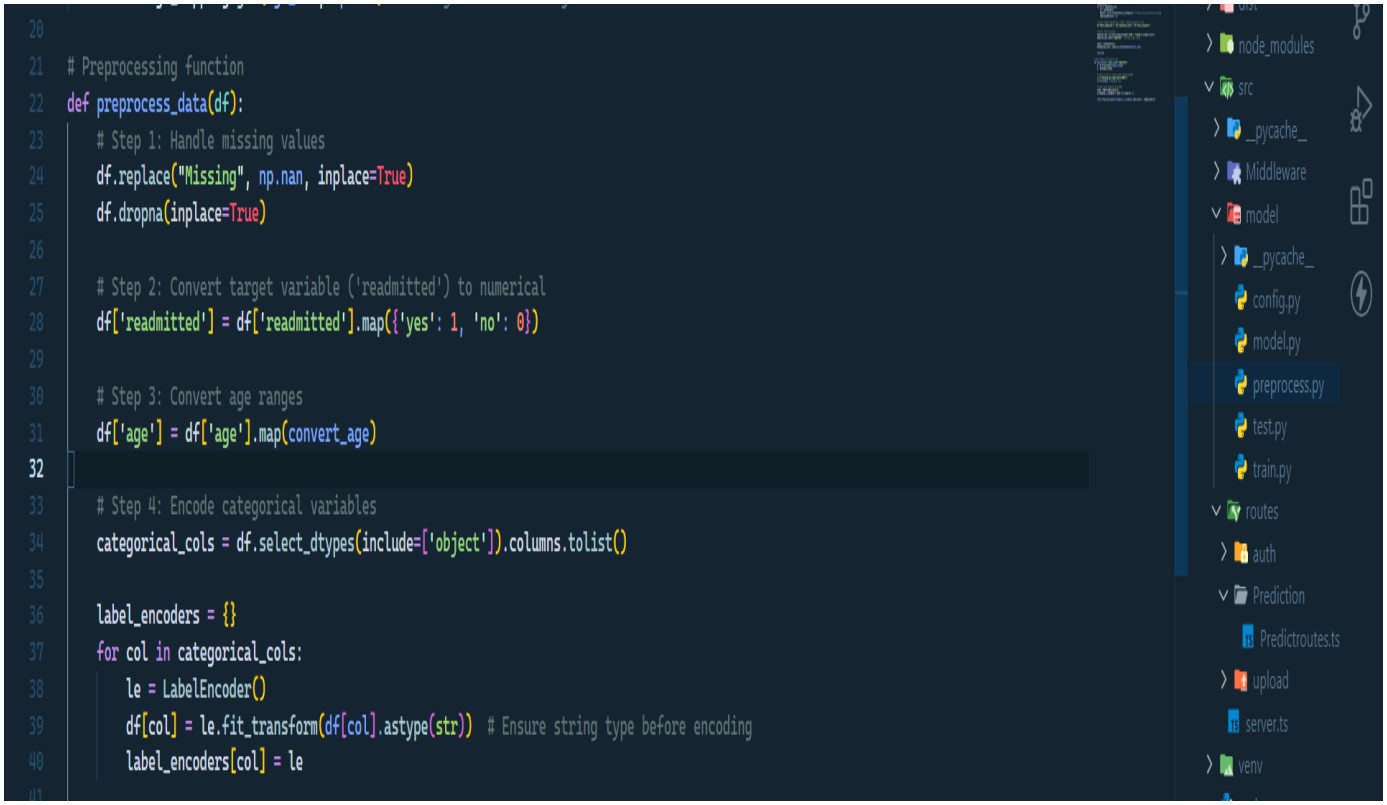

Figure 4.6 Code used for Data preprocessing

The pre-processing step improves data quality by creating a new feature called "health_complexity," which is calculated by multiplying comorbidity count with time spent in the hospital to better represent a patient's health condition. Next, all numerical columns in the dataset are identified, except for the target variable (readmitted), which is excluded to avoid errors. These numerical features are then scaled using StandardScaler(), which adjusts the values so they have a consistent range, making it easier for the model to learn patterns effectively.

4.3. Data Splitting and Handling Class Imbalance with SMOTE

The function split_data prepares the dataset for training and testing. First, it separates the target variable (readmitted) from the other features. Before applying SMOTE, it checks if all data types are numeric. SMOTE (Synthetic Minority Over-sampling Technique) is used to handle class imbalance by generating synthetic samples for the minority class instead of just

duplicating existing ones. This ensures the model doesn't become biased toward the majority class. Finally, the resampled dataset is split into training and testing sets.

```python
# Split dataset into train & test
def split_data(df, target_column='readmitted'):
    X = df.drop(columns=[target_column])
    y = df[target_column]

    # Ensure all data is numeric before applying SMOTE
    print("Checking data types before SMOTE:")
    print(X.dtypes)  # Debugging step

    # Handle class imbalance with SMOTE
    smote = SMOTE(random_state=42)
    X_resampled, y_resampled = smote.fit_resample(X, y)

    return train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

Figure 4.7 Data Splitting and Handling

4.4. Model Training and Performance Analysis

After training the hospital readmission prediction model, i evaluated its performance using various metrics such as accuracy, precision, recall, and F1-score. The image below presents the model's evaluation results, showcasing how well it distinguishes between patients who will be readmitted and those who will not. These insights help in assessing the model's effectiveness and identifying areas for improvement.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Predictions completed!
Evaluating model...
Model Accuracy: 0.6387665198237885
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.72      0.67      1399
           1       0.65      0.55      0.60      1325

    accuracy                           0.64      2724
   macro avg       0.64      0.64      0.63      2724
weighted avg       0.64      0.64      0.64      2724


Sample Readmission Predictions:
Patient 1: No Readmission
Patient 2: Needs Readmission
Patient 3: Needs Readmission
Patient 4: No Readmission
Patient 5: Needs Readmission
Patient 6: Needs Readmission
Patient 7: Needs Readmission
Patient 8: No Readmission
Patient 9: Needs Readmission
Patient 10: No Readmission
Model saved successfully!
PS C:\Users\user\Desktop\HOSPITAL READMISSION\backend>
```

Figure 4.8 Results after Training the Model

The results shows that the model achieve an accuracy of 63.87% and indicate that the model is better at identifying patients who won't be readmitted **(**72% recall**)** than those who will **(**55% recall**),** meaning it sometimes misses actual readmissions. The F1-score of 0.64 reflects a balance between precision and recall. A test run on 10 patients showed mixed predictions,

with some flagged for readmission and others not. The trained model has been successfully saved for future use and was then integrated into the UI, allowing users to easily upload patient data and receive predictions in a user-friendly format.

4.5 User Interface

The Hospital Readmission Prediction System is designed to help healthcare professionals identify patients at risk of readmission by providing a user-friendly interface for data upload and real-time predictions. The system consists three main components for seamless user interaction. It includes User Authentication, which handles secure registration and login; the Model Dashboard, where users can upload patient data and receive readmission predictions; and the User Profile**,** which allows users to manage their details. This structure ensures a smooth and efficient experience while navigating the system.

4.6 Screenshots of working system

i. User Authentication

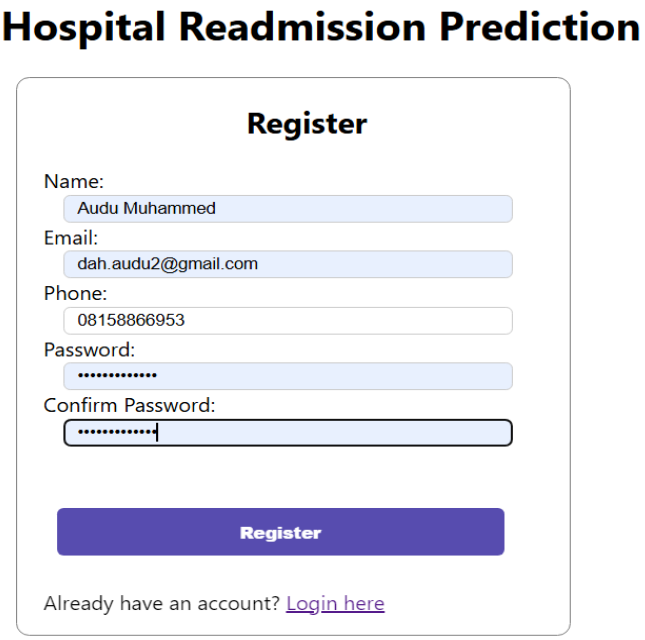

Figure 4.9 Registration form

This is the registration page for the Hospital Readmission Prediction system, where new users create an account by entering their name, email, phone number, and password. The form is simple and straight to the point, making it easy to fill out. Once the user submits their details, the data is sent to the backend for validation. If everything checks out, an account is created, and they can log in.

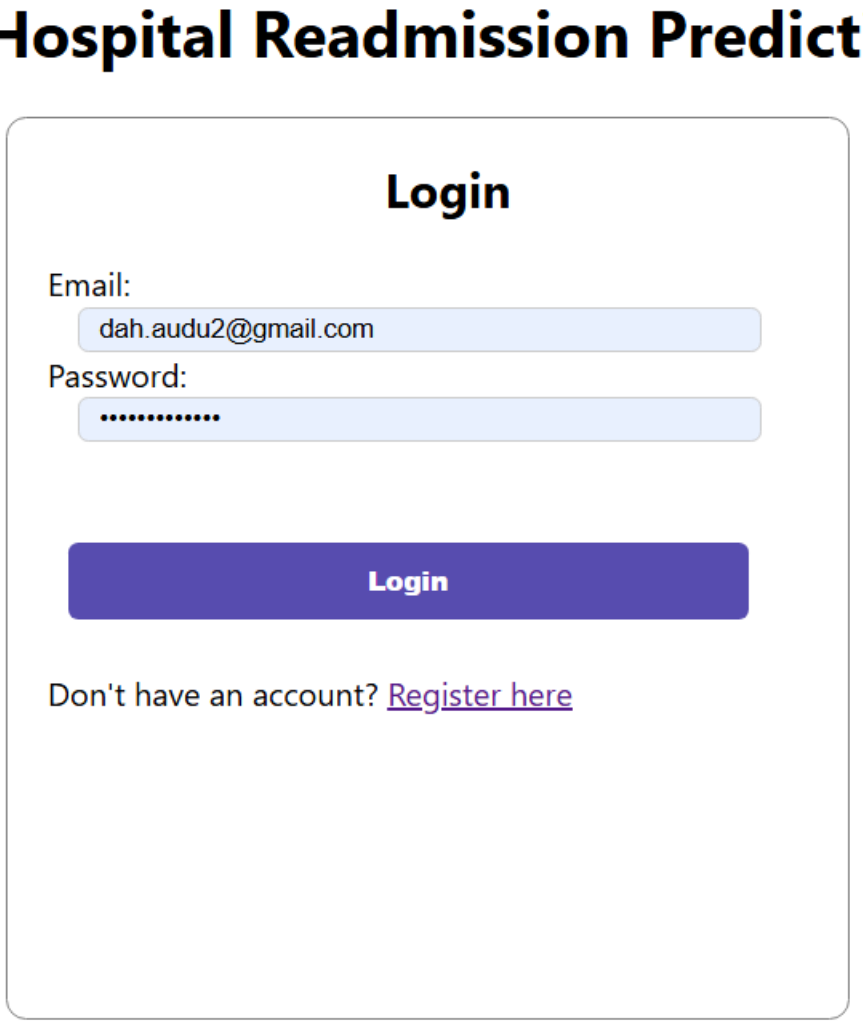

Figure 4.10 Login form

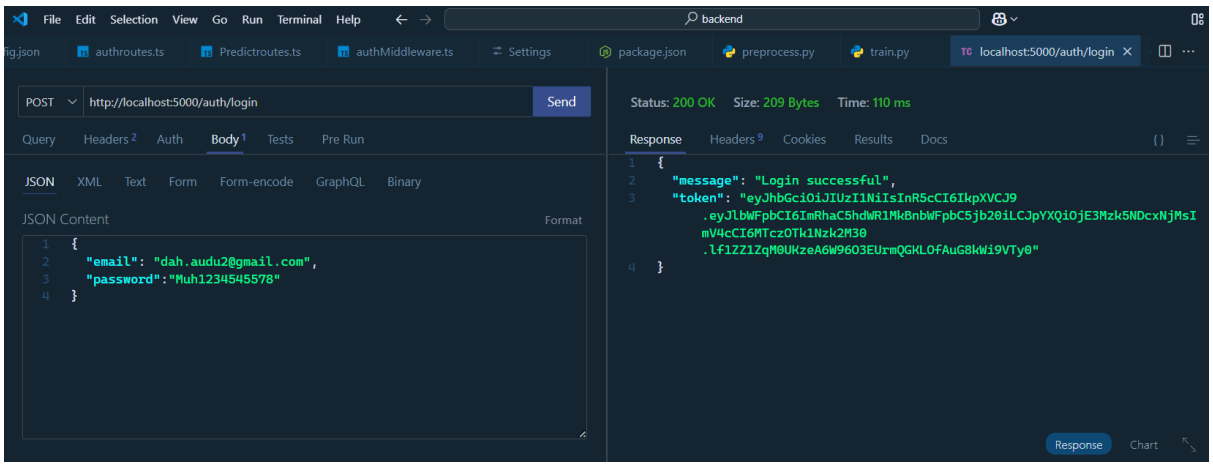

Figure 4.11 Api-endpoint for the login

The authentication flow starts here—once registered, users can log in, get a JWT token, and access restricted parts of the app like the dashboard and profile. If they already have an account, they can just click the Login here link to switch to the login page instead.

ii. Model Dashboard

This Dashboard is where patient data gets uploaded and analyzed to determine who's at high risk and needs extra attention versus those at low risk who require standard care. At the top, you can see real-time stats for the current session, including the total number of patients analyzed.

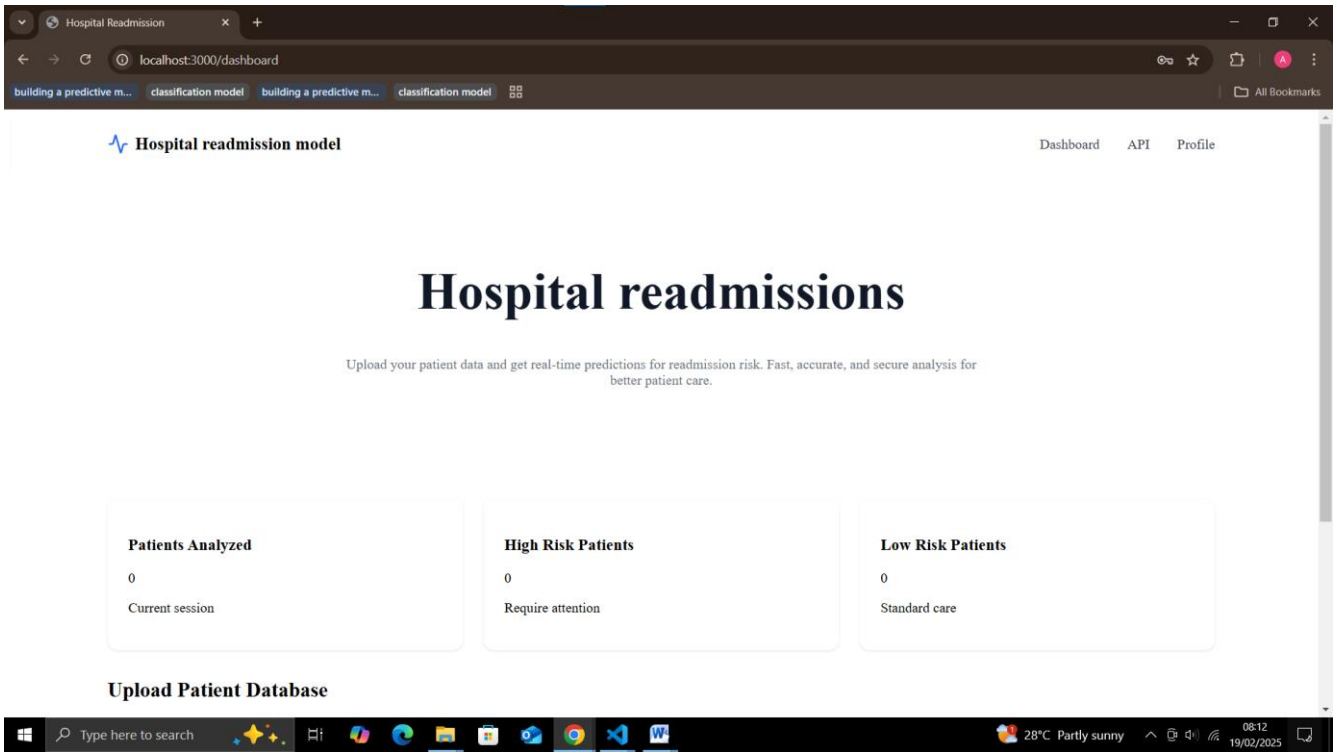

Figure 4.12 Model homepage

Below that, there's an upload section that supports CSV, JSON, and TXT formats. Once a file is uploaded, hitting the "Analyze Patients" button processes the data, classifies patients based on risk, and updates the stats.

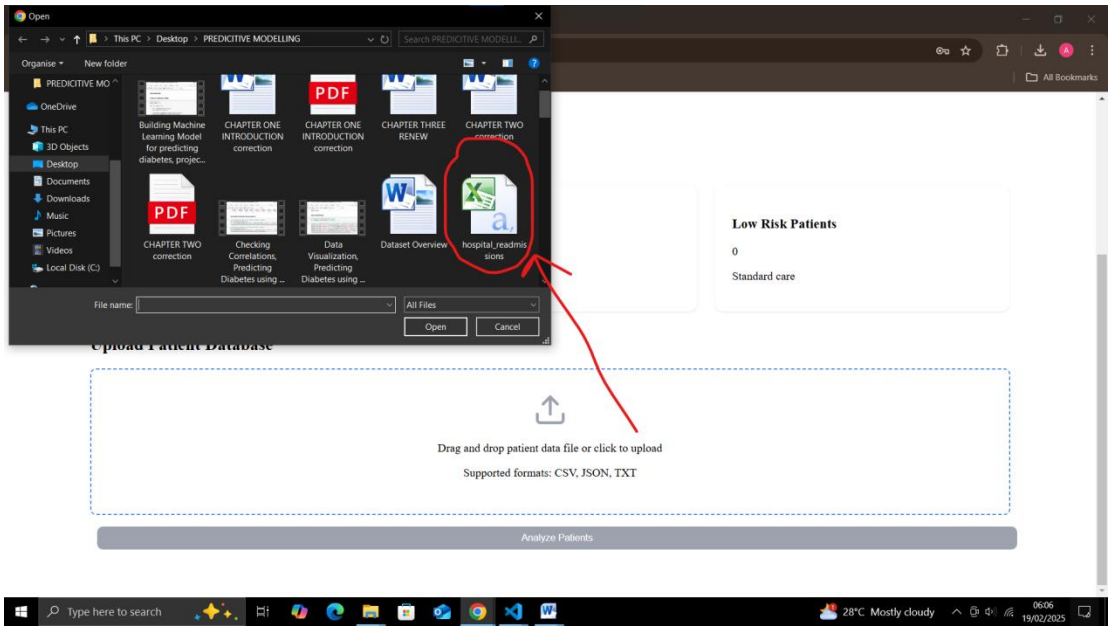

Figure 4.13 Uploading a dataset

Here's a preview of the file upload process in my Hospital Readmission Prediction dashboard. The system allows users to select a dataset containing patient information for analysis. In this case, I'm about to upload the hospital_readmissions dataset, which will be processed to classify patients into high-risk or low-risk groups.

Once uploaded, clicking "Analyze Patients" will trigger the backend to process the data and update the dashboard with predictions.

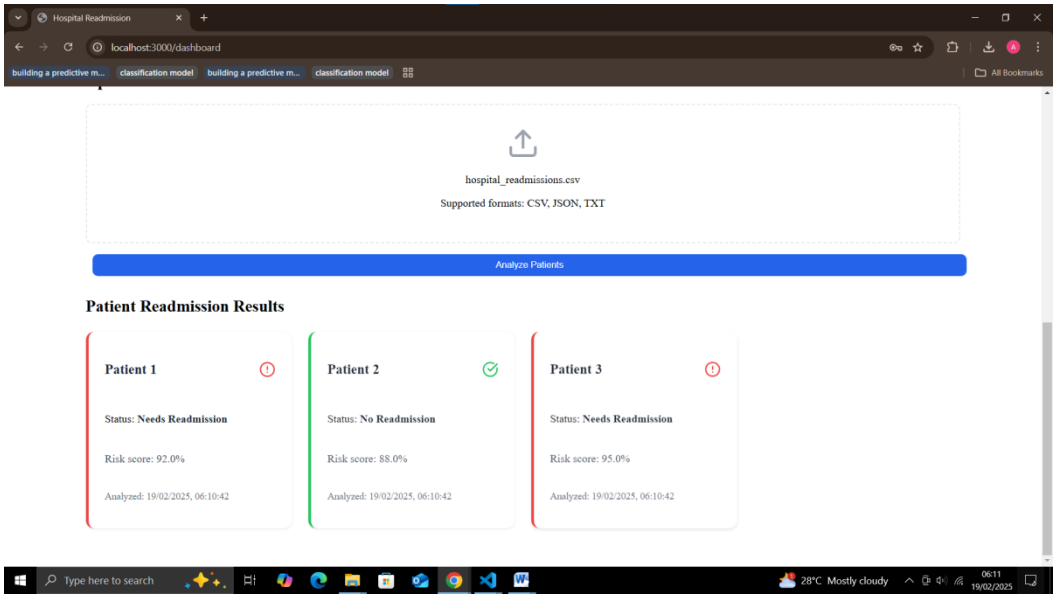

Figure 4.14 Display of prediction results

After uploading the **hospital_readmissions.csv** file, the system analyzes patient data and displays readmission predictions with risk scores. Patients with a high risk score are marked "Needs Readmission" in red, while those with a lower risk score are labeled "No Readmission" in green. In this case, Patient 1 (92%) and Patient 3 (95%) require readmission, while Patient 2 (88%) does not. This visualization allows for quick decision-making regarding patient care.

4.7 Challenges faced during Development

During the development of this hospital readmission prediction system, I encountered several challenges, including handling missing data, improving model accuracy, and ensuring smooth API integration with the frontend. Debugging file upload issues and optimizing real-time data processing also required significant effort. Overcoming these obstacles not only enhanced the system's reliability but also strengthened my problem-solving skills.

Despite these challenges, the project successfully delivers a functional tool that helps healthcare professionals assess patient readmission risks and enables good decision-making through an intuitive interface and risk score analysis.

CHAPTER FIVE
SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.0. Summary

This project started with the goal of predicting hospital readmissions using machine learning. It began by collecting and pre-processing the dataset, ensuring it was properly cleaned and structured for training. After conducting exploratory data analysis (EDA) in Jupyter Notebook, we moved to VS Code for further development. The model was trained and evaluated for accuracy, with plans to integrate explainability techniques like SHAP for better interpretability.

On the backend, we developed a RESTful API using Express.js, implementing JWT authentication for secure access control. The API was designed to handle user authentication, model predictions, and profile management while maintaining security and efficiency.

For the frontend, a dashboard was built using TypeScript and React, ensuring a smooth user experience. It includes key features like authentication (registration and login), a model dashboard for predictions, and a profile page for user data management. Styling was done with CSS to maintain a clean and structured interface. Along the way, challenges such as debugging API integrations, managing authentication flows, and refining the UI for better usability were addressed.

5.1. Conclusion

In conclusion, This project has been a rewarding journey, from setting up the backend and training the model to integrating authentication and building a functional dashboard. There were challenges along the way, like debugging authentication issues, optimizing API responses, and refining the UI, but each hurdle was a learning opportunity. While there's always room for improvement, the system is now in a stable and functional state, achieving its core objectives.

5.2 Recommendation

While the Hospital Readmission Prediction system is effective, further improvements can be made by enhancing model accuracy and integrating real-time risk monitoring for better decision-making. Refining the frontend dashboard to provide clearer insights and user-friendly visualizations will improve usability. Additionally, ensuring scalability and robust logging can help track prediction trends and detect anomalies. Lastly, conducting regular model evaluations and updating the dataset will ensure the system remains reliable and accurate over time

REFERENCES

Cai, A., Zheng, D., Qiu, R., & Yu, H. (2024). Predicting 1-year readmission for heart failure: A comparative study of machine learning algorithms. *ESC Heart Failure, 11(1)*, 11–20. https://doi.org/10.1002/ehf2.14589

Futoma, J., Morris, J., & Lucas, J. (2015). A comparison of models for predicting early hospital readmissions. *Journal of Biomedical Informatics, 56*, 229–238. PubMed

Jamei, M., Nisnevich, A., Wetchler, E., Sudat, S., Liu, E., & Pressman, A. (2017). Predicting all-cause risk of 30-day hospital readmission using artificial neural networks. *PLOS ONE, 12(7)*, e0181173. https://doi.org/10.1371/journal.pone.0181173

Kawakami, E., Tabata, J., Yanaihara, N., Ishikawa, T., Koseki, K., Iida, Y., ... & Okamoto, A. (2023). Application of artificial intelligence for preoperative diagnostic and prognostic prediction in epithelial ovarian cancer based on blood biomarkers. *Clinical Cancer Research, 29(5)*, 1184–1194. https://doi.org/10.1158/1078-0432.CCR-22-2319

Khalid, S., Khalil, T., & Nasreen, S. (2022). A survey of feature selection and feature extraction techniques in machine learning. *Proceedings of the 2014 Science and Information Conference*, 372–378. https://doi.org/10.1109/SAI.2014.6918213

Mahmoudi, E., & Kamdar, N. (2020). A longitudinal analysis of data quality in a large clinical data research network. *BMC Medical Informatics and Decision Making, 20*, 71. https://doi.org/10.1186/s12911-020-1051-8

Mortazavi, B. J., Downing, N. S., Bucholz, E. M., Dharmarajan, K., Manhapra, A., Li, S. X., Negahban, S. N., & Krumholz, H. M. (2016). Analysis of machine learning techniques for heart failure readmissions. *Circulation: Cardiovascular Quality and Outcomes, 9(6)*, 629–640. https://doi.org/10.1161/CIRCOUTCOMES.116.003039

Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., ... & Dean, J. (2018). Scalable and accurate deep learning with electronic health records. *npj Digital Medicine, 1(1)*, 18. https://doi.org/10.1038/s41746-018-0029-1

Shams, I., Ajorlou, S., & Yang, K. (2015). A predictive analytics approach to reducing 30-day avoidable readmissions among patients with heart failure, acute myocardial infarction, pneumonia, or COPD. *Health Care Management Science, 18(1)*, 19–34. https://doi.org/10.1007/s10729-014-9278-y

APPENDIX

SOURCE CODE

APP.TSX

```
import { Route, Routes, useLocation } from "react-router-dom";
import Header from "./components/header";
import Login from "./pages/Login";
import Register from "./pages/Register";
import Dashboard from "./pages/Dashboard";
import Profile from "./pages/profile"

function App() {
  const location = useLocation();
  const isDashboard = location.pathname === "/dashboard";
  const isProfile = location.pathname === "/profile"; // Ensure Profile is separate

  return (
    <>
      {isDashboard || isProfile ? (
        // Render Dashboard and Profile without "new/App" layout
        <Routes>
          <Route path="/dashboard" element={<Dashboard />} />
          <Route path="/profile" element={<Profile />} />
        </Routes>
      ) : (
        // Layout for Login and Register
        <div className="new">
          <Header />
          <div className="App">
            <Routes>
              <Route path="/" element={<Login />} />
              <Route path="/register" element={<Register />} />
            </Routes>
          </div>
        </div>
      )}
    </>
  );
}

export default App
```

INDEX.CSS

```
form {
    display: flex;
    flex-direction: column;
    max-width: 400px;

  }

  input {
    margin: 1px 0 0px 15px;
```

```
    padding: 10px 70px 10px 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    display: flex;
    flex-direction: column;

    width : 260px;
    height: 1px;
  }

  button {
    margin: 50px 20px 10px 10px;
    padding: 10px 10px;
    border-radius: 5px;
    width : 340px;
    border: none;
    background-color: #574caf;
    color: white;
    cursor: pointer;
    align-self: justify;
    font-weight: 800;

  }

  button:hover {
    background-color: #455da0;
  }
  .App{
    border : 1px solid gray;
    padding-left : 20px;
    margin-left: 20px;
    width : 400px;
    height : 450px;
    border-radius: 10px;
    text-align: justify;
  }
  .register{
    text-align: center;
  }
  .new{
    margin : 100px 250px 100px 500px;
    align-content: center;
    text-align: center;
    padding-right: 250px;
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

.nav-container {
  background-color: white;
```

```
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.05);
}

.nav-content {
  max-width: 1280px;
  margin: 0 auto;
  padding: 0 1rem;
  display: flex;
  justify-content: space-between;
  height: 4rem;
  align-items: center;
}

.logo-section {
  display: flex;
  align-items: center;
}

.logo-text {
  margin-left: 0.5rem;
  font-size: 1.25rem;
  font-weight: bold;
}

.nav-links {
  display: none;
}

.nav-links a {
  color: #374151;
  text-decoration: none;
  transition: color 0.2s;
}

.nav-links a:hover {
  color: #2563eb;
}

.hero-section {
  background-color: white;
  padding: 4rem 1rem;
  text-align: center;
}

.hero-title {
  font-size: 2.25rem;
  font-weight: 800;
  color: #111827;
}
```

```
.hero-description {
  margin-top: 1rem;
  max-width: 48rem;
  margin-left: auto;
  margin-right: auto;
  color: #6b7280;
}

.dashboard-container {
  max-width: 1280px;
  margin: 0 auto;
  padding: 3rem 1rem;
}

.stats-grid {
  display: grid;
  gap: 1.5rem;
  margin-bottom: 1.5rem;
}

.upload-zone {
  border: 2px dashed #e5e7eb;
  border-radius: 0.5rem;
  padding: 2rem;
  text-align: center;
  cursor: pointer;
  transition: border-color 0.2s;
}

.upload-zone:hover {
  border-color: #3b82f6;
}

.submit-button {
  width: 100%;
  padding: 0.5rem 1rem;
  border-radius: 0.5rem;
  font-weight: 500;
  margin-top: 1rem;
  transition: background-color 0.2s;
}

.submit-button-active {
  background-color: #2563eb;
  color: white;
}

.submit-button-active:hover {
  background-color: #1d4ed8;
}
```

```
.submit-button-disabled {
  background-color: #9ca3af;
  cursor: not-allowed;
  color: white;
}

.results-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 1rem;
  margin-top: 1rem;
}

.footer {
  background-color: white;
  border-top: 1px solid #e5e7eb;
  margin-top: 3rem;
  padding: 3rem 1rem;
  text-align: center;
  color: #6b7280;
}

@media (min-width: 768px) {
  .nav-links {
    display: flex;
    gap: 2rem;
  }

  .mobile-menu {
    display: none;
  }

  .hero-title {
    font-size: 3.75rem;
  }

  .stats-grid {
    grid-template-columns: repeat(3, 1fr);
  }
}
.patient-cards {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 1.5rem;
  margin-top: 1.5rem;
}

.patient-card {
  background-color: white;
```

```
  border-radius: 0.75rem;
  padding: 1.5rem;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
  transition: transform 0.2s, box-shadow 0.2s;
}

.patient-card:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.patient-card.high-risk {
  border-left: 4px solid #ef4444;
}

.patient-card.low-risk {
  border-left: 4px solid #22c55e;
}

.patient-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 1rem;
}

.patient-header h3 {
  font-size: 1.25rem;
  font-weight: 600;
  color: #1f2937;
}

.risk-icon {
  width: 24px;
  height: 24px;
}

.risk-icon.high {
  color: #ef4444;
}

.risk-icon.low {
  color: #22c55e;
}

.patient-details {
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
}
```

DASHBOARD.TSX

```
import React, { useState } from 'react';
import { Upload, Menu, Activity, Github, AlertCircle, CheckCircle } from 'lucide-react';
import { Link } from "react-router-dom";

interface PatientPrediction {
  id: number;
  label: 'Needs Readmission' | 'No Readmission';
  confidence: number;
  timestamp: string;
}

const Dashboard: React.FC = () => {
  const [file, setFile] = useState<File | null>(null);
  const [predictions, setPredictions] = useState<PatientPrediction[]>([]);
  const [isLoading, setIsLoading] = useState(false);

  const handleFileDrop = (e: React.DragEvent<HTMLDivElement> |
      React.ChangeEvent<HTMLInputElement>) => {
    e.preventDefault();
    const droppedFile = 'dataTransfer' in e
      ? e.dataTransfer?.files[0]
      : e.target.files?.[0];
    setFile(droppedFile || null);
  };

  const handleSubmit = () => {
    setIsLoading(true);
    // Simulate predictions for multiple patients
    setTimeout(() => {
      const samplePredictions: PatientPrediction[] = [
        {
          id: 1,
          label: 'Needs Readmission',
          confidence: 0.92,
          timestamp: new Date().toLocaleString()
        },
        {
          id: 2,
          label: 'No Readmission',
          confidence: 0.88,
          timestamp: new Date().toLocaleString()
        },
        {
          id: 3,
          label: 'Needs Readmission',
          confidence: 0.95,
          timestamp: new Date().toLocaleString()
        }
      ];
```

```
    setPredictions(samplePredictions);
    setIsLoading(false);
  }, 1500);
};

return (
  <div>
    {/* Navigation */}
    <nav className="nav-container">
      <div className="nav-content">
        <div className="logo-section">
          <Activity className="h-8 w-8" style={{ color: '#2563eb' }} />
          <span className="logo-text">Hospital readmission model</span>
        </div>
        <div className="nav-links">
          <a href="#">Dashboard</a>
          <a href="#">API</a>
          <Link to="/profile">Profile</Link>
        </div>
        <div className="mobile-menu">
          <Menu className="h-6 w-6" />
        </div>
      </div>
    </nav>

    {/* Hero Section */}
    <div className="hero-section">
      <h1 className="hero-title">Hospital readmissions</h1>
      <p className="hero-description">
        Upload your patient data and get real-time predictions for readmission risk.
        Fast, accurate, and secure analysis for better patient care.
      </p>
    </div>

    {/* Dashboard Content */}
    <div className="dashboard-container">
      <div className="stats-grid">
        <div className="stat-card">
          <h3>Patients Analyzed</h3>
          <p className="stat-value">{predictions.length || 0}</p>
          <p className="stat-subtitle">Current session</p>
        </div>

        <div className="stat-card">
          <h3>High Risk Patients</h3>
          <p className="stat-value">
            {predictions.filter(p => p.label === 'Needs Readmission').length || 0}
          </p>
          <p className="stat-subtitle">Require attention</p>
        </div>
```

```
  <div className="stat-card">
    <h3>Low Risk Patients</h3>
    <p className="stat-value">
      {predictions.filter(p => p.label === 'No Readmission').length || 0}
    </p>
    <p className="stat-subtitle">Standard care</p>
  </div>
</div>

{/* Upload Section */}
<div className="upload-section">
  <h2>Upload Patient Database</h2>
  <div
    className="upload-zone"
    onDragOver={(e) => e.preventDefault()}
    onDrop={handleFileDrop}
    onClick={() => document.getElementById('fileInput')?.click()}
  >
    <input
      id="fileInput"
      type="file"
      style={{ display: 'none' }}
      onChange={handleFileDrop}
    />
    <Upload className="mx-auto mb-4" size={48} style={{ color: '#9ca3af' }} />
    <p className="upload-text">
      {file ? file.name : 'Drag and drop patient data file or click to upload'}
    </p>
    <p className="upload-subtitle">
      Supported formats: CSV, JSON, TXT
    </p>
  </div>

  <button
    className={`submit-button ${file && !isLoading ? 'submit-button-active' : 'submit-
button-disabled'}`}
    onClick={handleSubmit}
    disabled={!file || isLoading}
  >
    {isLoading ? 'Analyzing...' : 'Analyze Patients'}
  </button>
</div>

{/* Results Section */}
{predictions.length > 0 && (
  <div className="results-section">
    <h2>Patient Readmission Results</h2>
    <div className="patient-cards">
      {predictions.map((prediction) => (
```

```
            <div
              key={prediction.id}
              className={`patient-card ${
                prediction.label === 'Needs Readmission' ? 'high-risk' : 'low-risk'
              }`}
            >
              <div className="patient-header">
                <h3>Patient {prediction.id}</h3>
                {prediction.label === 'Needs Readmission' ? (
                  <AlertCircle className="risk-icon high" />
                ) : (
                  <CheckCircle className="risk-icon low" />
                )}
              </div>
              <div className="patient-details">
                <p className="prediction-label">
                  Status: <span>{prediction.label}</span>
                </p>
                <p className="confidence">
                  Risk score: {(prediction.confidence * 100).toFixed(1)}%
                </p>
                <p className="timestamp">
                  Analyzed: {prediction.timestamp}
                </p>
              </div>
            </div>
          ))}
        </div>
      </div>
    )}
  </div>

 </div>
 );
};

export default Dashboard;
```

LOGIN.TSX & REGISTER.TSX

```
// src/pages/Login.tsx
import React, { useState } from "react";
import { Link,  useNavigate } from "react-router-dom";
import axios from "axios";

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState('');
  const navigate = useNavigate(); // Hook to navigate programmatically
```

```
const handleLogin = async (e: React.FormEvent) => {
  e.preventDefault();
  console.log("Login form submitted");
  // Handle login logic here (e.g., send to backend)
  try {
    const response = await axios.post('http://localhost:5000/auth/login', {
      email,
      password
    });
    console.log(response.data);  // This will show the response from the backend (e.g., token)
    // You can store the token in localStorage or context here
    // Redirect to dashboard on successful login
     navigate("/dashboard");
  } catch (err: any) {
    console.error("Error during login request:", err);
    setError(err.response?.data?.error || 'Something went wrong!');
  }
};
return (
  <div>
    <h2 className="register">Login</h2>
    <form onSubmit={handleLogin}>
      <div>
        <label htmlFor="email">Email:</label>
        <input
          type="email"
          id="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
      </div>
      <div>
        <label htmlFor="password">Password:</label>
        <input
          type="password"
          id="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
      </div>
      <button type="submit">Login</button>
    </form>
    <p>
      Don't have an account? <Link to="/register">Register here</Link>
    </p>
  </div>
);
```

```
};

export default Login;

import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";

const Register = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleRegister = async (e: React.FormEvent) => {
    e.preventDefault();
    setError("");

    if (password !== confirmPassword) {
      setError("Passwords do not match!");
      return;
    }

    try {
      const response = await axios.post("http://localhost:5000/auth/register", {
        email,
        password,
      });
      console.log("Registration successful:", response.data);
      alert("Registration successful! Please login.");
      navigate("/"); // Redirect to login page
    } catch (err: any) {
      console.error("Registration failed:", err.response?.data || err.message);
      setError(err.response?.data?.message || "Registration failed. Try again.");
    }
  };

  return (
    <div>
      <h2 className="register">Register</h2>
      {error && <p style={{ color: "red" }}>{error}</p>}
      <form onSubmit={handleRegister}>
      <div>
          <label htmlFor="email">Name:</label>
          <input
            type="string"
            id="name"
          />
        </div>
```

```
        <div>
          <label htmlFor="email">Email:</label>
          <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
        </div>
        <div>
          <label htmlFor="email">Phone:</label>
          <input
            type="number"
            id="email"
          />
        </div>
        <div>
          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <div>
          <label htmlFor="confirmPassword">Confirm Password:</label>
          <input
            type="password"
            id="confirmPassword"
            value={confirmPassword}
            onChange={(e) => setConfirmPassword(e.target.value)}
            required
          />
        </div>
        <button type="submit">Register</button>
      </form>
      <p>
        Already have an account? <Link to="/">Login here</Link>
      </p>
    </div>
  );
};export default Register;
```

BACKEND

```
import pandas as pd
import numpy as np
import pickle
import joblib
```

```
import os

# Get the current script directory
BASE_DIR = os.path.dirname(os.path.abspath(__file__))

# Load trained model
MODEL_PATH = r"C:\Users\user\Desktop\HOSPITAL READMISSION\backend\random_forest_model.pkl"
SCALER_PATH = r"C:\Users\user\Desktop\HOSPITAL READMISSION\backend\scaler.pkl"

if not os.path.exists(MODEL_PATH) or not os.path.exists(SCALER_PATH):
    raise FileNotFoundError("Model or scaler file not found. Train the model first.")

# Load the saved model and scaler
model = joblib.load(MODEL_PATH)
scaler = joblib.load(SCALER_PATH)

# List of expected features (must match training data)
feature_names = [
    "age", "time_in_hospital", "n_lab_procedures", "n_procedures",
    "n_medications", "n_outpatient", "n_inpatient", "n_emergency",
    "medical_specialty", "diag_1", "diag_2", "diag_3", "glucose_test",
    "A1Ctest", "change", "diabetes_med", "comorbidity_count",
    "comorbidity", "health_complexity"
]

def predict_readmission(new_data):
    """
    Predicts hospital readmission based on patient data.

    Args:
        new_data (list of lists): Each sublist represents a patient's feature values.

    Returns:
        list: Predictions as "Needs Readmission" or "No Readmission".
    """

    # Convert input data to a DataFrame
    feature_names = ['age', 'blood_pressure', 'cholesterol']  # Example column names
    df = pd.DataFrame([patient.features], columns=feature_names)

    # Handle missing features by filling with 0
    for col in feature_names:
        if col not in df.columns:
            df[col] = 0  # Default value
    #print("Features expected by scaler:", scaler.feature_names_in_)
    #print("Features provided by script:", df.columns.tolist())

    # Reorder columns to match training data
```

```
    df = df[feature_names]

    # Scale features
    df_scaled = scaler.transform(df)
    print("First row of scaled test data:", df_scaled[0])
    # Predict readmission
    predictions = model.predict(df_scaled)

    # Convert predictions to readable format
    results = ["Needs Readmission" if pred == 1 else "No Readmission" for pred in
       predictions]

    return results




    print(f"Patient {i+1}: {result}")
```

TRAINED MODEL

```
from preprocess import load_data, preprocess_data, split_data
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
import joblib

# Load dataset
print("Loading data...")
df = load_data('data/cleaned_hospital_readmissions.csv')
print("Data loaded successfully!")

# Preprocess
print("Preprocessing data...")
df = preprocess_data(df)
print("Data preprocessed successfully!")

# Split data
print("Splitting data...")
X_train, X_test, y_train, y_test = split_data(df, target_column="readmitted")
print("Data split successfully!")

feature_names = X_train.columns.tolist()
# Scale features
print("Scaling data...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Data scaling completed!")

# Save the scaler
```

```
joblib.dump(scaler, "scaler.pkl")
print("Scaler saved successfully!")
print("First row of scaled training data:", X_train_scaled[0])

# Train model
print("Training model...")
model = RandomForestClassifier(n_estimators=200, max_depth=10, random_state=42)
model.fit(X_train_scaled, y_train)
print("Model training completed!")

# Make predictions
print("Making predictions...")
y_pred = model.predict(X_test_scaled)
print("Predictions completed!")

# Evaluate model
print("Evaluating model...")
print("Model Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Sample Readmission Predictions
print("\nSample Readmission Predictions:")
readmission_results = ["Needs Readmission" if pred == 1 else "No Readmission" for pred in
      y_pred]

# Display first 10 results
for i in range(10):
   print(f"Patient {i+1}: {readmission_results[i]}")

# Save trained model
joblib.dump(model, "random_forest_model.pkl")
print("Model saved successfully!")
```

API ENDPOINTS

```
import { Router, Request, Response } from "express";
import { authenticateToken,AuthRequest} from "../../Middleware/authMiddleware";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";

const router = Router();
const SECRET_KEY = "your_secret_key"; // Change this later

// Temporary storage (Replace with a database later)
const users: { email: string; password: string }[] = [];

router.post("/register", async (req: Request, res: Response): Promise<void>=> {
  try {
    const { email, password } = req.body;

    if (!email || !password) {
```

```
     res.status(400).json({ error: "Email and password are required" });
     return;
    }

    const hashedPassword = await bcrypt.hash(password, 10);
    users.push({ email, password: hashedPassword });

     res.status(201).json({ message: "User registered successfully" });
     return;
  } catch (error) {
    res.status(500).json({ error: "Internal server error" });
    return;
  }
});

router.post("/login", async (req:Request, res:Response): Promise<void> => {
  try {
    const { email, password } = req.body;

    // Validate input
    if (!email || !password) {
      res.status(400).json({ error: "Email and password are required" });

    }

    // Find user
    const user = users.find((user) => user.email === email);
    if (!user) {
      res.status(401).json({ error: "Invalid credentials" });
      return;
    }

    // Compare password
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      res.status(401).json({ error: "Invalid credentials" });
      return;
    }

    // Generate JWT token
    const token = jwt.sign({ email: user.email }, SECRET_KEY, { expiresIn: "3h" });

    res.status(200).json({ message: "Login successful", token });
  } catch (error) {
    res.status(500).json({ error: "Internal server error" });
  }
});

router.get("/history", authenticateToken, (req: AuthRequest, res: Response) => {
  res.json({ message: "This is protected data", user: req.user });
```

```
});
export default router;

import { Router, Request, Response } from "express";
import { authenticateToken } from "../../Middleware/authMiddleware";
import axios from "axios";

const Predictionrouter = Router();

Predictionrouter.post("/predict", async (req: Request, res: Response): Promise<void> => {
  try {
    const patientData = req.body;

    // Validate input (ensure required fields exist)
    if (!patientData || Object.keys(patientData).length === 0) {
      res.status(400).json({ error: "Patient data is required" });
      return;
    }

    // Make a POST request to FastAPI
    const apiUrl = "http://0.0.0.0:8000/predict";  // Your FastAPI server URL
    const response = await axios.post(apiUrl, patientData);

    if (response.status === 200) {
      const prediction = response.data;  // Data from FastAPI
      res.status(200).json(prediction);
    } else {
      res.status(response.status).json({ error: "Error from FastAPI" });
    }
  } catch (error) {
    console.error("Error:", error);
    res.status(500).json({ error: "Internal server error" });
  }
});

const predictionHistory: { user: string; prediction: any }[] = [];

Predictionrouter.get("/prediction-history", authenticateToken, async (req: Request, res:
      Response) : Promise<void> => {
  try {
    // Assuming user authentication is required (JWT middleware needed)
    const userEmail = (req as any).user.email; // Replace with actual user from token

    const userPredictions = predictionHistory.filter(
      (entry) => entry.user === userEmail
    );

     res.status(200).json(userPredictions);
     return;
  } catch (error) {
```

```
    res.status(500).json({ error: "Internal server error" });
    return;
  }
});

export default Predictionrouter;

import { Router, Request, Response } from "express";
import multer from 'multer';
import path from 'path';

const Uploadroutes = Router();

// Configure multer for file storage
const storage = multer.diskStorage({
   destination: (req, file, cb) => {
      cb(null, './uploads'); // Ensure this folder exists
   },
   filename: (req, file, cb) => {
      cb(null, Date.now() + path.extname(file.originalname)); // Unique filename
   }
});

const upload = multer({ storage });

// Upload route
Uploadroutes.post('/upload', upload.single('dataset'), (req: Request, res: Response) : void =>
      {
   if (!req.file) {
      res.status(400).json({ error: 'No file uploaded.' });
      return;
   }
   res.json({
      fileName: req.file.filename,
      filePath: `/uploads/${req.file.filename}`
   });
});

export default Uploadroutes;
```

SERVER USING EXPRESS.JS

```
import express from "express";
import authRoutes from "./routes/auth/authroutes"; // Ensure the path is correct
import Predictionrouter from "./routes/Prediction/Predictroutes";
import Uploadroutes from "./routes/upload/upload";
import cors from 'cors'; // Import cors

const app = express();

app.use(express.json()); // Middleware to parse JSON
```

```
app.use(
  cors({
    origin: "http://localhost:3000", // Allow frontend
    methods: "GET,POST,PUT,DELETE",
    allowedHeaders: "Content-Type,Authorization",
    credentials: true, // If using cookies/auth tokens
  })
);


app.use("/auth", authRoutes)
app.use("/Prediction",Predictionrouter )
app.use("/Upload",Uploadroutes)

app.options('*', cors())
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```