INVENTORY MANAGEMENT AND DEMAND FORECASTING USING MACHINE LEARNING ARIMA MODEL

Minor project-1 report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Artificial Intelligence & Data Science

By

SETHU RAMAN Y	(22UEAD0067)	**(VTU 21546)**
INIYAVAN S	(22UEAD0013)	**(VTU 22510)**
NAVADEEP J	(22UEAD0040)	**(VTU 23403)**

Under the guidance of
Dr.V.Dhilipkumar,M.E,PhD,
Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

November, 2024

INVENTORY MANAGEMENT AND DEMAND FORECASTING USING MACHINE LEARNING ARIMA MODEL

Minor project-1 report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Artificial Intelligence & Data Science

By

SETHU RAMAN Y	(22UEAD0067)	**(VTU 21546)**
INIYAVAN S	(22UEAD0013)	**(VTU 22510)**
NAVADEEP J	(22UEAD0040)	**(VTU 23403)**

Under the guidance of
Dr.V.Dhilipkumar,M.E,PhD,,
Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

November, 2024

CERTIFICATE

This is to certify that the work contained in the project report titled ”INVENTORY MANAGEMENT AND DEMAND FORECASTING USING MACHINE LEARNING ARIMA MODEL” by SETHU RAMAN Y (22UEAD0067), INIYAVAN S (22UEAD0013), and NAVADEEP J (22UEAD0040) has been carried out under my supervision and has not been submitted elsewhere for a degree.

Signature of Supervisor
Dr. V. Dhilip Kumar
Professor
Computer Science Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
November, 2024

Signature of Head of the Department
Dr. P. Santhi
Professor & Head
Artificial Intelligence & Data Science
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
November, 2024

Signature of the Dean
Dr. S. P. Chokkalingam
Professor & Dean
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
November, 2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

SETHU RAMAN Y

Date: / /

(Signature)

INIYAVAN S

Date: / /

(Signature)

NAVADEEP J

Date: / /

APPROVAL SHEET

This project report entitled ”INVENTORY MANAGEMENT AND DEMAND FORECASTING USING MACHINE LEARNING ARIMA MODEL” by SETHU RAMAN Y (22UEAD0067), INIYAVAN (22UEAD0013), NAVADEEP J (22UEAD0040) is approved for the degree of B.Tech in Artificial Intelligence & Data Science.

Examiners **Supervisor**

Dr.V.Dhilipkumar, M.E, PhD,,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Honorable Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (Electrical), B.E. (Mechanical), M.S (Automobile),D.Sc., and Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Vel Tech Rangarajan Dr. Sagunthala R & D Institute of Science and Technology, for her blessings.

We express our sincere thanks to our respected Chairperson and Managing Trustee **Mrs. RANGARAJAN MAHALAKSHMI KISHORE,B.E.,** Vel tech Rangarajan Dr. Sagunthala R & D Institiute of Science and Technology for her blessings.

We are very much grateful to our beloved **Vice Chancellor Prof. RAJAT GUPTA,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. S. P. CHOKKALINGAM, M.Tech., Ph.D.,& Associate Dean, Dr. V. DHILIP KUMAR, M.E., Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Professor & Head, Department of Artificial Intelligence & Data Science, Dr. P. SANTHI, M.E., Ph.D.,** for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Dr. V. Dhilip Kumar, M.E, PhD.,** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinator Mr. R. DURAI VASANTH, M.E.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

SETHU RAMAN Y **(22UEAD0067)**

INIYAVAN S **(22UEAD0013)**

NAVADEEP **(22UEAD0040)**

ABSTRACT

Businesses must practice effective inventory management to satisfy client requests and keep prices down. Preventing instances where an oversupply or stock out largely depends on accurate demand forecasting. Inventory demand forecasting systems have advanced considerably by integrating machine learning techniques with large amounts of historical data. The objectives are to better match production schedules with purchasing trends,optimize inventory levels, make educated judgments through accurate demand estimates, and improve resource allocation for increased profitability and customer satisfaction. We use methods for preparing the data and addressing missing values as part of our data preparation. We then introduce ARIMA for time series forecasting and Random Forest Regression using supervised learning. According to our findings, both models successfully forecast inventory demand, which promotes optimal stock levels, lower costs, and increased operational effectiveness.

Keywords:LSTM (Long-Short-Term Memory), ARIMA (Autoregressive Integrated Moving Average), Random Forest, Supervised Learning, Optimization, Time Series.

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ARIMA	AutoRegressive Integrated Moving Average
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
SARIMA	Seasonal AutoRegressive Integrated Moving Average
SVM	Support Vector Machine

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

In today's competitive business environment, effective inventory management, and accurate demand forecasting continue more critical for organizations to achieve operational excellence and maximize returns. Inventory management is defined as the science of controlling stock levels, coordinating activities within the supply chain, and streamlining acquisition strategies so that the right products arrive at the right time. It is also important to underscore that effective inventory management is essentially important since surplus stock will be reduced largely .

On the other hand, the stock-outs can be seamlessly prevented which will translate into user satisfaction. Keeping the inventory levels in line with true demand, RMS can cut down the carrying costs, enhance cash now, and become responsive to changes in the market. The research indicates that improving the area of inventory will result in operation excellence and competitive advantages for companies.

1.2 Aim of the project

The aim of this project is to use machine learning and Python to improve the accuracy of inventory demand forecasts. The project's goal is to create reliable forecasting models utilizing ARIMA Time Series Forecasting and Random Forest Regression through analyzing past sales data and market trends. With the use of these models, businesses will be better able to forecast their future needs for inventory, which will minimize stock-outs, minimize overstock problems, and optimize inventory levels. In the end, this approach aims at enhancing client satisfaction, save expenditures, and improve operational efficiency through better informed inventory management decisions.

1.3 Project Domain

Our project focuses on developing a comprehensive inventory management and demand forecasting system using advanced machine learning techniques and time series forecasting. The system is designed to optimize inventory levels, minimize stockouts, and prevent overstock situations, all while enhancing overall operational efficiency. By analyzing datasets from customers, products, and inventory records, the project employs models like ARIMA for time series forecasting and Random Forest for sales prediction. These predictive insights enable businesses to align their inventory with customer demand more accurately, leading to cost savings and improved customer satisfaction.

In addition, the project integrates a user-friendly web interface where users can upload relevant datasets, such as customer information, product details, and inventory data, for automated analysis. The platform then generates sales predictions and inventory optimization reports, helping businesses make data-driven decisions. This system is particularly beneficial for industries like retail and e-commerce, where demand volatility and inventory management are critical to success.

1.4 Scope of the Project

The scope of our project involves building an intelligent inventory management and demand forecasting system that helps businesses maintain optimal stock levels while reducing operational costs. By accurately tracking inventory and efficiently managing stock levels, the system ensures that products are always available to meet customer demand. It integrates real time data analysis to handle orders effectively, ensuring seamless operations. With the system's ability to forecast future demand, businesses can align their inventory with expected sales trends, making informed decisions for better resource planning.

Additionally, the project focuses on optimizing the supply chain by preventing issues like overstocking or stockouts, which can impact profitability and customer satisfaction. By predicting future demand and automating inventory adjustments, the system minimizes waste, reduces excess costs, and enhances the overall efficiency of the supply chain. The system is designed to support businesses of various sizes, helping them maintain balanced stock levels and ensuring smooth, cost-effective operations.

Chapter 2

LITERATURE REVIEW

Literature Review on Demand Forecasting Techniques in Retail Demand forecasting is a critical aspect of retail management, influencing inventory control, supply chain efficiency, and overall business performance. The advent of machine learning (ML) has transformed traditional forecasting methods, enabling retailers to leverage vast amounts of data for improved accuracy. This literature review examines various demand forecasting techniques, highlighting the strengths and limitations of each approach while identifying current gaps and future research directions.

2.0.1 Traditional Techniques in Demand Forecasting

Historically, demand forecasting relied on statistical methods such as moving averages, exponential smoothing, and regression analysis. These techniques are straightforward and easy to implement but often fall short in capturing complex patterns in data. For instance, Carter et al. (2022) compared Seasonal Autoregressive Integrated Moving Average (SARIMA) models with Long Short-Term Memory (LSTM) networks, demonstrating that while SARIMA can effectively model seasonality, it struggles with non-linear relationships prevalent in retail demand data. This limitation underscores the need for more sophisticated methods that can adapt to varying demand patterns.[1]
Traditional methods like simple linear regression assume a linear relationship between historical sales and influencing factors such as price or promotions. However, as Smith and Johnson (2022) pointed out, consumer behavior is often influenced by a multitude of factors that interact in complex ways 3. As a result, reliance on these simpler models can lead to significant forecasting errors, particularly in dynamic retail environments.[2]

2.0.2 Machine Learning Approaches

The integration of machine learning into demand forecasting has gained significant traction in recent years. Various studies have explored the efficacy of different ML

algorithms in predicting retail sales. For example, Sharma and Kumar (2023) conducted a comparative analysis of demand forecasting techniques, finding that ML models outperformed traditional methods in terms of accuracy and adaptability to changing market conditions. Their research highlights the potential of machine learning to enhance decision-making processes within retail organizations.[3][4]

2.0.3 Deep Learning Models

Deep learning techniques, particularly LSTM networks, have shown great promise in time-series forecasting due to their ability to capture long-term dependencies in sequential data. Zhang et al. (2020) demonstrated that LSTM networks could outperform traditional models by effectively capturing these dependencies, leading to improved forecast accuracy .[5] Their findings suggest that deep learning models can significantly enhance forecast accuracy, especially in environments characterized by seasonal fluctuations. In addition to LSTMs, other deep learning architectures such as Convolutional Neural Networks (CNNs) have been explored for demand forecasting tasks. Wang et al. (2020) highlighted the effectiveness of CNNs in extracting features from time series data, which can then be used for accurate predictions.[6][7]

This combination of CNNs and LSTMs represents a powerful approach to modeling complex demand patterns. Moreover, Nguyen et al. (2020) explored a hybrid approach combining time series decomposition with machine learning techniques. Their results indicated that this combination could yield more accurate forecasts by addressing both seasonal trends and irregular demand patterns 5. This hybrid methodology presents an exciting avenue for future research, particularly in optimizing model selection based on specific retail contexts.[8][9]

2.0.4 Comparative Analyses of Forecasting Techniques

Several studies have focused on comparing various forecasting methodologies to identify the most effective approaches for different retail scenarios. For instance, Alavi et al. (2021) conducted a comparative study of several ML approaches for demand forecasting in retail supply chains. They found that ensemble methods—such as Random Forests and Gradient Boosting—often yield superior results compared to individual models due to their ability to aggregate predictions from multiple algorithms.[10][11] In another study, Thompson and Robson (2020) introduced hybrid

forecasting models that integrate both statistical and machine learning approaches for inventory optimization in retail settings. Their findings revealed that hybrid models not only improve forecast accuracy but also enhance decision-making processes related to inventory management . The exploration of hybrid models represents a significant trend in the literature, indicating a shift towards more integrated forecasting solutions.[12][13]

2.0.5 Challenges and Limitations

Despite the advancements in machine learning for demand forecasting, several challenges remain. One major issue is the requirement for large datasets to train complex models effectively. Mohd and Rahman (2019) noted that smaller retailers often lack sufficient historical data, limiting their ability to implement advanced ML techniques successfully . This challenge emphasizes the need for developing methodologies that can work effectively with limited data. Additionally, overfitting is a common concern with complex models like neural networks; Jain et al. (2020) emphasized the importance of proper tuning and validation to mitigate this risk.[14]

Overfitting occurs when a model learns noise rather than the underlying pattern in the training data, resulting in poor performance on unseen data. Another significant challenge is the interpretability of machine learning models. While deep learning methods can provide high accuracy, their ”black box” nature makes it difficult for practitioners to understand how predictions are made. This lack of transparency can hinder trust among stakeholders who rely on these forecasts for critical business decisions. Future research should focus on developing interpretable ML models that maintain high predictive performance while providing insights into their decision-making processes.[15][16]

2.0.6 Gaps in Current Research

While existing literature provides valuable insights into various forecasting techniques, several gaps remain unaddressed. First, there is a need for more comprehensive studies examining the impact of external factors such as economic conditions or consumer behavior changes on demand forecasts. For instance, during the COVID-19 pandemic, many retailers experienced unprecedented shifts in consumer demand that traditional models failed to predict accurately. Incorporating external variables

into ML models could enhance their robustness against sudden market changes. Furthermore, most studies focus on specific industries or product categories; thus, there is limited understanding of how these techniques perform across diverse retail sectors.[16][17]
Future research should aim to conduct cross-industry analyses to identify best practices and adaptable strategies for different retail environments. Another area that requires further exploration is the integration of real-time data into demand forecasting models. As retailers increasingly adopt IoT technologies and advanced analytics tools, there is an opportunity to leverage real-time sales data and external signals (such as social media trends or economic indicators) to refine forecasts dynamically.[18]

In conclusion, the evolution of demand forecasting techniques from traditional statistical methods to advanced machine learning approaches marks a significant advancement in retail management practices. While machine learning offers enhanced accuracy and adaptability, challenges such as data requirements and model interpretability persist. The literature indicates a growing trend towards hybrid models that combine various methodologies to optimize forecasting performance.

As retailers continue to navigate an increasingly complex market landscape characterized by rapid changes in consumer behavior and external conditions, ongoing research will be vital in refining these techniques and addressing existing gaps. By leveraging advanced analytics and incorporating external factors into predictive models, retailers can improve their demand forecasting capabilities and ultimately enhance operational efficiency.[19]

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

In most existing inventory management systems, data collection, processing, and analysis are highly manual, leading to inefficiencies in tracking stock levels, demand forecasting, and supply chain optimization. These systems often rely on spreadsheets or standalone ERP (Enterprise Resource Planning) software, which might not be fully integrated with modern predictive analytics tools. As a result, the data processing capabilities are limited, and critical decisions such as reordering stock, managing seasonal demand, or addressing supply chain bottlenecks are based on guesswork rather than data-driven insights.

The inability to process and analyze large datasets in real-time further complicates the problem. Traditional systems do not leverage machine learning models or time-series analysis for forecasting, leading to situations where businesses either run out of stock or overstock, both of which result in financial losses. Moreover, these systems often lack automation for tasks like report generation, making it difficult for businesses to gain timely insights and act proactively. Another significant disadvantage is the high possibility of human error during data entry, resulting in inaccurate reporting and incorrect inventory levels.

3.2 Proposed System

The proposed system seeks to overcome the limitations of existing inventory management systems by integrating machine learning algorithms such as ARIMA (AutoRegressive Integrated Moving Average) and Random Forest. These models will help predict future demand more accurately by analyzing historical data, seasonal trends, and market fluctuations. By doing so, businesses can maintain optimal stock levels, avoiding both overstocking and stockouts. Additionally, the system will pro-

vide real-time analytics and automated report generation, allowing for timely and informed decision-making.

The system's ability to automate data collection, analysis, and reporting ensures that human error is minimized, increasing the accuracy of forecasts and inventory tracking. Another major advantage is its scalability; as businesses grow, the system can handle larger datasets and more complex inventory models. This results in improved operational efficiency, reduced costs, and better customer satisfaction, as businesses can fulfill orders without delay. The integration with cloud services ensures that the system is accessible from anywhere, further enhancing its flexibility and usability.

3.3 Feasibility Study

3.3.1 Economic Feasibility

The economic feasibility of the proposed inventory management system is highly favorable. Although there may be upfront costs associated with developing and deploying the system, such as software development and infrastructure setup, the long-term benefits far outweigh these initial expenses. By reducing the likelihood of overstocking and stockouts, businesses can significantly reduce the costs associated with excess inventory or lost sales. Additionally, the system's automation features will reduce labor costs, as fewer manual entries and reports are needed.

Moreover, businesses can adopt cloud-based solutions to run the system, further lowering infrastructure costs by reducing the need for expensive in-house servers. Cloud-based services offer subscription models, which allow businesses to pay for only the resources they need. This scalability means that small businesses can afford to implement the system without making large upfront investments, while larger businesses can scale up as their needs grow, making the system cost-effective for companies of all sizes.

3.3.2 Technical Feasibility

From a technical standpoint, the proposed system is highly feasible. It will be built using modern, proven technologies such as Python, Jupyter Notebooks, and machine

learning libraries like TensorFlow and scikit-learn. These tools are widely adopted in the data science community and are well-documented, ensuring that the system can be developed and maintained efficiently. Moreover, the use of cloud computing platforms such as AWS, Microsoft Azure, or Google Cloud ensures that the system can handle large datasets, provide real-time analytics, and be accessed remotely.

The technical infrastructure required to run the system is minimal, especially if it is cloud-based. With the availability of open-source libraries and tools, the cost and complexity of development are further reduced. The integration of machine learning models for demand forecasting also ensures that the system is capable of handling the complexities of inventory management in a modern business environment. Additionally, the system will be user-friendly, with a clear and intuitive interface, making it accessible to non-technical users as well.

3.3.3 Social Feasibility

The proposed system also stands out in terms of social feasibility. By automating repetitive tasks such as data entry and report generation, the system improves the work-life balance of employees, allowing them to focus on more meaningful tasks like strategic decision-making. This can lead to higher employee satisfaction and retention rates. Furthermore, by reducing errors and optimizing inventory levels, businesses can improve customer satisfaction by ensuring that products are always in stock and available for purchase.

In a broader context, the system contributes to corporate social responsibility by reducing waste. Overstocking often leads to unsold products, which may eventually be discarded, contributing to environmental harm. By optimizing inventory and reducing waste, the system helps businesses adopt more sustainable practices. Additionally, the system's data-driven approach enables businesses to be more agile and responsive to market trends, leading to better customer service and stronger relationships with suppliers and other stakeholders.

3.4 System Specification

3.4.1 Hardware Specification

- Processor: Intel Core i7 (10th Gen or later) or equivalent AMD Ryzen processor
- RAM: 16GB DDR4 or higher
- Storage: 512GB SSD or higher for faster data processing
- Graphics Processing Unit (GPU): NVIDIA GeForce RTX 2060 or higher (for faster ML computations)
- Network Connectivity: High-speed internet (1 Gbps or higher for cloud-based integration)
- Backup: External storage for backups (1TB HDD or SSD)
- Display: Full HD Monitor for better data visualization and analysis

3.4.2 Software Specification

- Operating System: Windows 10, macOS, or Linux (Ubuntu 20.04 or higher)
- Programming Language: Python 3.8 or higher (with libraries like TensorFlow, scikit-learn, Pandas, and NumPy)
- Development Environment: Jupyter Notebook, Visual Studio Code (VS Code)
- Database: MySQL, PostgreSQL, or MongoDB for data storage
- Cloud Service: AWS, Microsoft Azure, or Google Cloud for real-time processing and scalability
- Version Control: Git and GitHub/GitLab for version tracking
- ML Libraries: TensorFlow, scikit-learn, Matplotlib, Seaborn
- Security Protocols: SSL/TLS for secure data transfer, role-based access control for secure access

3.4.3 Standards and Policies

- **Anaconda Prompt**: Anaconda Prompt is a command-line interface that handles various machine learning and data science modules. It comes with an easy-to-navigate interface and supports multiple IDEs for coding, making it easier for data scientists to work across different platforms.
- **Standard Used**: ISO/IEC 27001 for ensuring the security and confidentiality of sensitive data, providing robust security management practices.
- **Jupyter**: Jupyter is an open-source web application that enables the creation and sharing of documents containing live code, equations, visualizations, and narrative text. It is widely used for data cleaning, transformation, statistical modeling, and machine learning tasks.
- **Standard Used**: ISO/IEC 27001, ensuring secure and efficient data handling during machine learning and data analysis tasks.

Chapter 4

METHODOLOGY

4.1 General Architecture

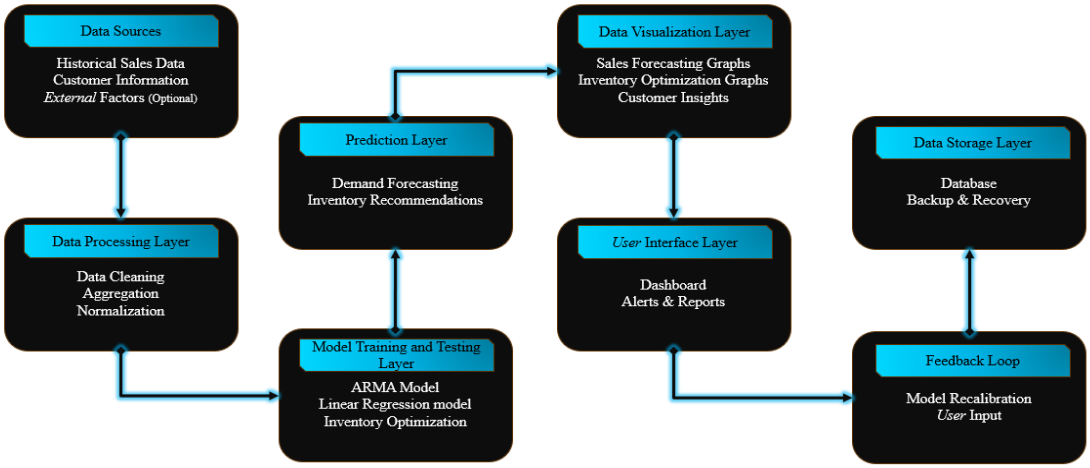

Figure 4.1: **Architecture diagram of Inventory model**

The Figure 4.1 likely showcases the overall system architecture, highlighting how data flows through various components. It should detail the interactions between data sources (such as sales data), machine learning models (ARIMA and Random Forest), and the system outputs (predictions and inventory optimization results). The architecture may include data storage, preprocessing units, and prediction modules that work together for inventory forecasting and demand optimization.

4.2 Design Phase

4.2.1 Data Flow Diagram

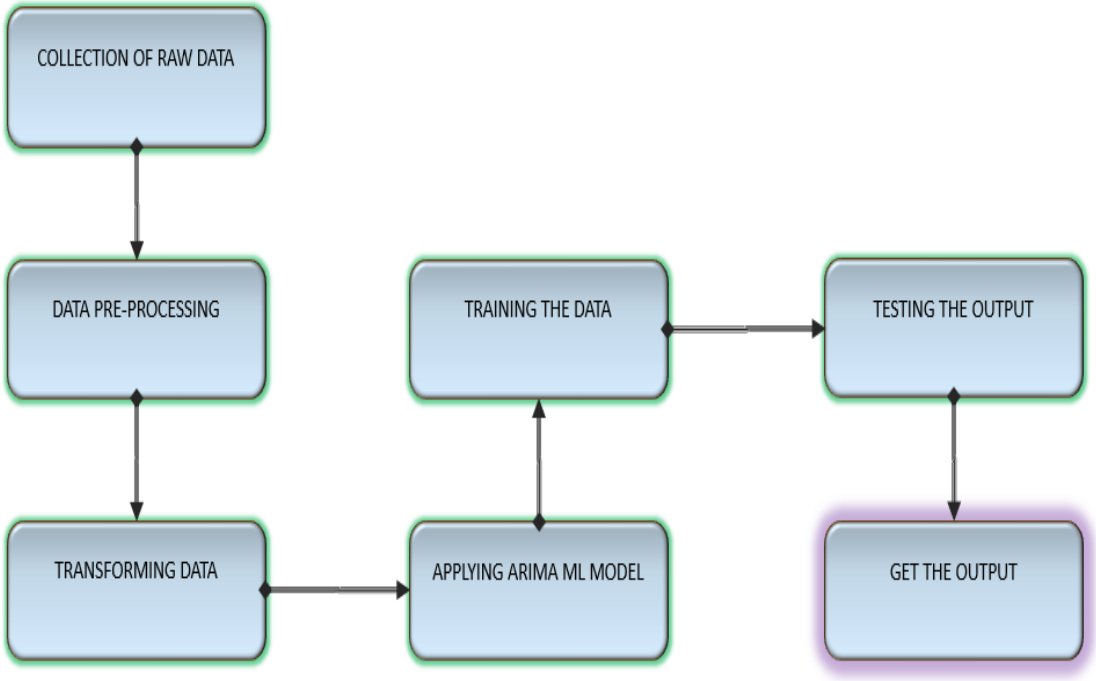

Figure 4.2: **Data flow of model**

The Figure 4.2 (DFD) illustrates the flow of data within the system, showing how data is inputted, processed, and outputted. It likely starts from historical sales data input, followed by data preprocessing, applying machine learning models (ARIMA, Random Forest), and ends with generating forecast outputs and inventory suggestions.

4.2.2 Use Case Diagram

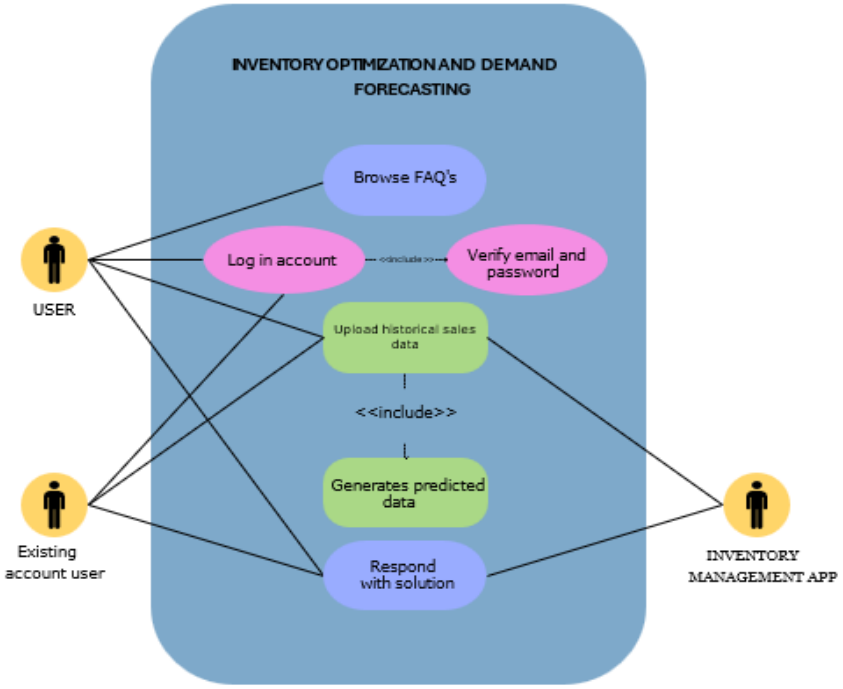

Figure 4.3: **use case**

The Figure 4.3 describes the different user interactions with the system. It might represent users (like business owners) logging in, uploading historical sales data, and generating demand forecasts. Other possible interactions could include managing inventory levels based on forecasts, and viewing the predicted demand through a dashboard.

4.2.3 Class Diagram

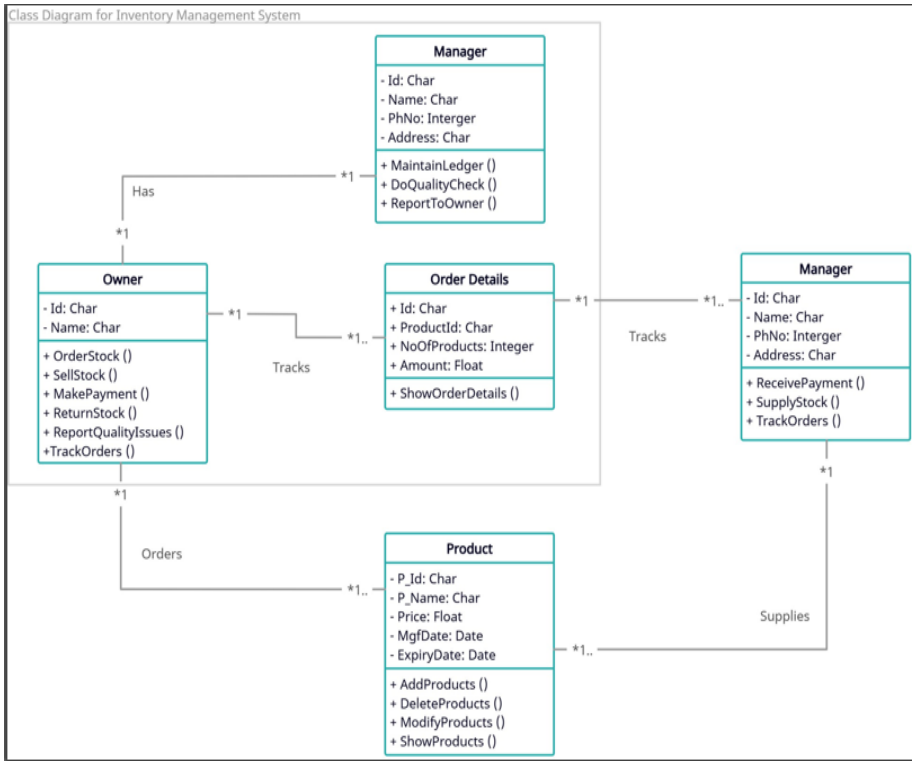

Figure 4.4: **class diagram of model**

The Class Diagram provides an object-oriented view of the system, representing the structure of the system in terms of classes and their relationships. Classes like ”User,” ”SalesData,” ”Inventory,” and ”PredictionModel” may be included, defining attributes (e.g., sales quantity, stock levels) and methods (e.g., forecast demand, optimize inventory).

4.2.4 Sequence Diagram

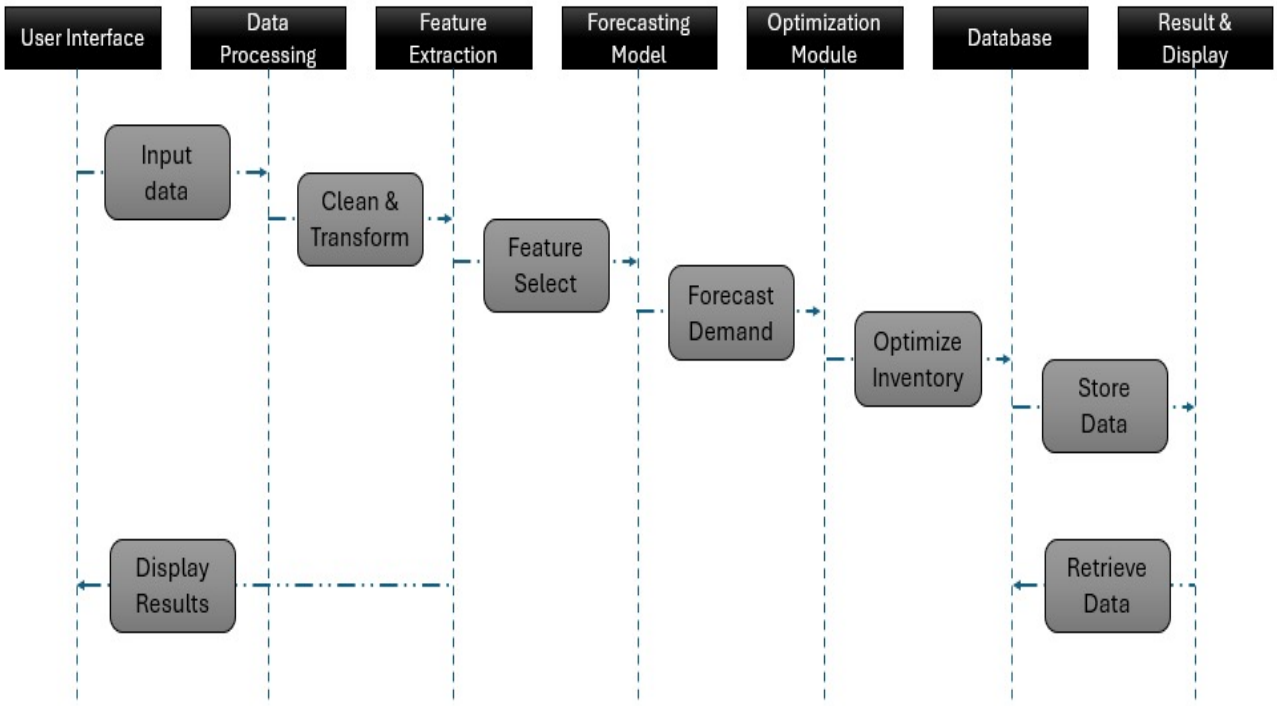

Figure 4.5: **sequence diagram of model**

The Figure 4.5 shows the order in which processes are executed over time. It captures the interaction between different system components, such as user actions triggering data input, model application, and response in the form of optimized inventory and demand forecasts.

4.2.5 ER diagram

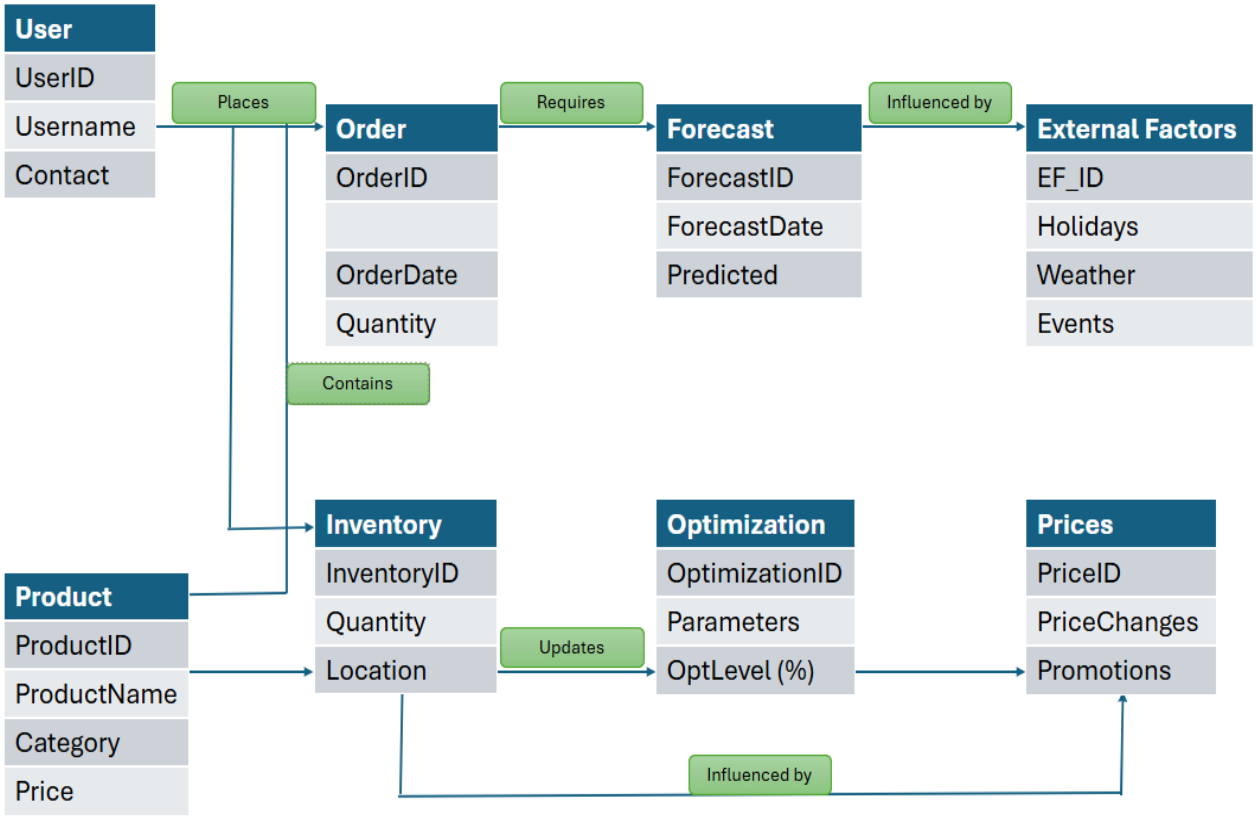

Figure 4.6: **ER diagram of model**

The Figure 4.6 represents the relationships between entities in the database, such as ”Customer,” ”Product,” ”Sales,” ”Inventory,” and ”Purchase History.” It showcases how these entities are interconnected, like customers purchasing products, products being part of inventory, and inventory being influenced by sales data.

4.2.6 Activity Diagram

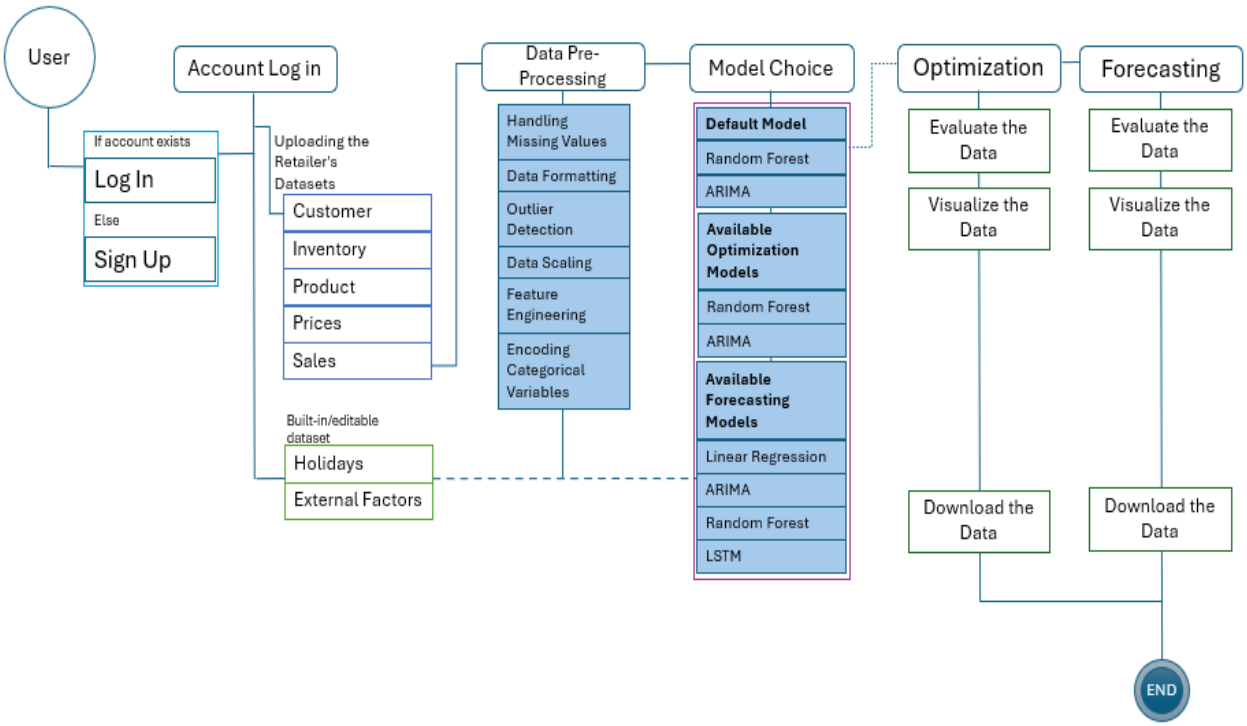

Figure 4.7: **activity diagram of model**

The Figure 4.7 visualizes the workflow for the system's key functions, such as requesting forecasts, processing sales data, and optimizing inventory. It follows a step-by-step flow of activities from when a user uploads data, through preprocessing, model training (Random Forest, ARIMA), and outputting predictions and optimizations.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

The algorithm for inventory optimization and demand forecasting involves several key steps to ensure accurate predictions and effective inventory management. First, the system loads historical sales data, which serves as the basis for predicting future demand. This data is preprocessed by handling missing values, converting data types, and managing categorical variables. The data is then split into training and testing sets to ensure the models can be trained and evaluated properly. The next step is to apply the Random Forest Regression model, which is trained on the pro-

cessed data. Random Forest builds multiple decision trees and combines their results to make predictions, reducing overfitting and handling complex datasets effectively. Simultaneously, the ARIMA (AutoRegressive Integrated Moving Average) model is used for time series forecasting, particularly useful for non-stationary data. ARIMA combines autoregression, integration, and moving averages to forecast future values. Finally, the predictions generated by both models are used to optimize inventory levels, ensuring the right amount of stock is available, preventing stockouts or overstock situations. This optimization is crucial for maintaining business efficiency, reducing costs, and satisfying customer demand.

4.3.2 Pseudo Code

```
BEGIN
    Load historical_sales_data
    Preprocess_data:
        Handle_missing_values
        Convert_data_types
        Handle_categorical_variables
        Split_data_into_train_and_test

    // Random Forest Regression Model
    Train_random_forest_model(train_data)
    predicted_demand_rf = Predict_demand_using_rf(test_data)

    // ARIMA Model
    Check_stationarity(historical_sales_data)
    IF data_not_stationary THEN
        Apply_differencing(historical_sales_data)
    Identify_ARIMA_parameters()
```

```
    Fit_ARIMA_model(train_data)
    predicted_demand_arima = Forecast_ARIMA(test_data)

    // Inventory Optimization
    Optimize_inventory(predicted_demand_arima)

    OUTPUT predicted_demand_rf, predicted_demand_arima,
    optimized_inventory
END
```

4.4 Module Description

4.4.1 Module 1: Random Forest Regression for Demand Prediction

The first module in the system is dedicated to using the Random Forest Regression algorithm to predict future demand based on historical sales data. Random Forest is an ensemble learning method that constructs multiple decision trees and merges their results to provide more accurate predictions. By training the model on historical data, it can analyze complex patterns and trends that might be hidden in the data. The advantage of Random Forest is its ability to handle large datasets with a variety of features without overfitting, making it robust and reliable for predicting future demand. This module plays a critical role in forecasting, as it helps businesses anticipate future sales, thereby allowing them to plan their inventory needs more effectively. The model's ability to aggregate predictions from different decision trees ensures that the forecast is less prone to errors, providing a strong foundation for the next stage of the project—inventory optimization.

4.4.2 Module 2: ARIMA Model for Time Series Demand Forecasting

The second module utilizes the ARIMA (AutoRegressive Integrated Moving Average) model, which is specifically designed for time series forecasting. ARIMA is highly effective for analyzing non-stationary data, which often reflects real-world

business environments where trends and patterns change over time. The ARIMA model first checks the stationarity of the dataset and applies differencing if necessary to make the series stationary. Once the data is stationary, the model identifies the autoregressive (AR), integration (I), and moving average (MA) components that best represent the dataset. The ARIMA model is then fitted to the training data and used to forecast future sales demand. This module is particularly useful for capturing and predicting seasonal trends, making it a valuable tool in industries where demand fluctuates over time. By combining this module with the Random Forest model, businesses can leverage both machine learning and statistical methods to generate more accurate and reliable demand forecasts.

4.4.3 Module 3: Inventory Optimization Based on Forecasted Demand

The third module focuses on optimizing inventory levels based on the forecasts generated by the Random Forest and ARIMA models. After predicting future demand, the system uses these predictions to adjust inventory levels, ensuring that businesses maintain sufficient stock to meet customer demand without overstocking. Inventory optimization involves analyzing the forecasted sales and comparing them with current stock levels to recommend changes. The goal is to maintain a balance where businesses can fulfill orders promptly, avoid stockouts that disrupt customer satisfaction, and prevent overstocking, which can lead to excess costs. This module also includes a visualization component that presents sales trends, allowing businesses to make data-driven decisions. The insights provided by this module enable companies to improve operational efficiency, reduce storage costs, and enhance customer satisfaction by ensuring that the right products are available at the right time.

4.5 Steps to Execute/Run/Implement the Project

4.5.1 Step 1: Data Collection and Preprocessing

The first step in executing the project is collecting and preprocessing historical sales data, which forms the backbone of the demand forecasting process. Data collection involves obtaining relevant sales data from databases or other sources. Once the data is collected, it undergoes preprocessing, where missing values are handled, and data types are adjusted for consistency. Categorical variables are also managed, often through encoding, to make them suitable for machine learning algorithms. The pre-

processed data is then split into training and test sets, allowing the models to learn from one subset while being evaluated on another. This ensures that the models are not overfitted and can generalize well to new data. Preprocessing is a crucial step that ensures the accuracy and reliability of the forecasting models.

4.5.2 Step 2: Model Training and Prediction

The second step involves training the machine learning models—Random Forest and ARIMA—on the preprocessed data. First, the Random Forest Regression model is trained on the training dataset, learning from the historical patterns and relationships in the data. The trained model is then used to predict future demand on the test data, providing the first set of demand forecasts. Simultaneously, the ARIMA model is applied to the time series data. The stationarity of the data is checked, and if necessary, differencing is applied to make the data stationary. The ARIMA model is then fitted to the training data, and its parameters (AR, I, MA) are adjusted to provide the best forecast. The model is used to predict future demand based on historical trends. The output from both models gives a comprehensive view of the forecasted demand for the next period.

4.5.3 Step 3: Inventory Optimization and Visualization

The final step in the process involves inventory optimization, which is critical for translating the demand forecasts generated by the machine learning models into actionable strategies for maintaining optimal stock levels. In this step, the predictions produced by both the Random Forest and ARIMA models are carefully analyzed to determine how much inventory should be held at any given time in order to meet future customer demand effectively. This analysis helps in calculating the precise stock levels required to minimize the risk of stockouts, which occur when a business runs out of stock and is unable to meet customer demand, leading to potential revenue loss and customer dissatisfaction. At the same time, the system also works to avoid overstocking, where excess inventory ties up resources and increases storage costs, which can reduce profitability and operational efficiency.

To achieve this balance, the system incorporates sophisticated algorithms that take into account both the demand forecasts and other relevant business factors such as lead times, order quantities, seasonality, and market trends. The optimization process

is designed to ensure that the inventory levels are flexible and adaptive, responding dynamically to changes in predicted demand, helping businesses to reduce costs associated with excess inventory and improve cash flow by maintaining just the right amount of stock. By striking the right balance between having too little and too much inventory, businesses can streamline their supply chain operations and enhance overall efficiency.

In addition to calculating optimal stock levels, the system also includes a powerful visualization component. This visualization allows business managers and decision-makers to see the predicted demand trends over time in a clear and intuitive format, often through graphs, charts, or dashboards. The visual representation of data plays an essential role in making complex information easy to understand, enabling managers to grasp the forecasted sales volumes and their implications for inventory management quickly. The visualization may include key metrics such as sales forecasts for the next period, stock levels required to meet demand, and potential inventory shortfalls or surpluses

.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

The **Input Design** in this project plays a crucial role in ensuring the accuracy of the demand forecasting and inventory optimization system. The primary input for this system is *historical sales data*, which includes detailed information such as product categories, sales volumes, prices, customer demographics, and purchase history. These inputs are preprocessed to remove any noise, handle missing values, and convert data into suitable formats. The input design also involves categorizing variables into numerical and categorical types to ensure they are appropriately handled by machine learning models. Additionally, for time series forecasting using ARIMA, the input includes time-based data that is converted into a format suitable for detecting trends, seasonality, and other temporal patterns. Accurate input design ensures that both the Random Forest and ARIMA models are provided with well-structured data that can be used to train and validate forecasting models efficiently.

5.1.2 Output Design

The **Output Design** focuses on the results produced by the system after the data is processed and models are applied. The primary output includes the *predicted demand* for future periods, generated by both Random Forest Regression and ARIMA models. These forecasts are displayed in a clear and understandable format, such as visual graphs and tables, showing expected sales trends for the next period (e.g., 30 days). Additionally, the system provides *optimized inventory levels*, which recommend how much stock should be maintained to meet predicted demand while minimizing overstocking or stockouts. These outputs are crucial for decision-makers to monitor inventory requirements and The output also includes performance metrics of the models, such as accuracy, error rates, and visualization of trends, enabling businesses to evaluate the reliability of the forecast.

5.2 Testing

Testing is a vital phase in ensuring that the system functions as expected and produces reliable outputs. Throughout the development of the demand forecasting and inventory optimization project, several testing strategies are employed to evaluate individual components and the overall system. Testing ensures that the forecasting models work accurately, data is processed correctly, and inventory recommendations are valid based on the forecasts. Additionally, testing the system helps identify any issues or bugs that may arise, such as incorrect predictions, data handling errors, or interface problems. Multiple types of testing are carried out to verify the integrity of the project.

5.3 Types of Testing

5.3.1 Unit Testing

Unit testing involves verifying the smallest testable parts of the project, primarily focusing on the core logic of the models used for demand forecasting. Specifically, the unit tests evaluate the functionality of the *Random Forest Regression* and *ARIMA* models by examining their ability to process data inputs and generate accurate predictions.

- **Input**: The inputs for unit testing include small datasets with known historical sales records. The test focuses on ensuring that when the models receive clean input data, they output expected forecast values.
- **Test Result**: The test result checks whether the output prediction values match the expected demand forecasts. If successful, the test confirms that the model is functioning correctly and producing valid forecasts based on historical data.

5.3.2 Integration Testing

Integration testing evaluates how different components of the project work together. It checks the interaction between modules such as data preprocessing, model execution (Random Forest, ARIMA), and inventory optimization. The purpose of integration testing is to ensure that data flows smoothly between these components without errors or data mismatches.

- **Input**: The integration testing input consists of complete datasets passed through the entire pipeline, from preprocessing to model prediction and inventory calculation.
- **Test Result**: The result confirms whether the system generates seamless outputs—accurate forecasts and optimized inventory—without any failures in the data-handling process across different modules.

5.3.3 System Testing

System testing is the final phase of testing and examines the project as a whole. It ensures that the full demand forecasting and inventory optimization system is working correctly and is capable of handling large datasets, producing accurate forecasts, and optimizing inventory. The entire system is tested end-to-end, from input data entry to the final visualization and inventory suggestion output.

- **Input**: The input for system testing includes a large-scale historical sales dataset with various product categories, sales volumes, and time-based data for forecasting.
- **Test Result**: The system testing result checks whether the forecasting models are accurately predicting future demand, and if the inventory optimization module is recommending appropriate stock levels. The outputs are also compared to real-world scenarios to validate the system's effectiveness in inventory management and demand forecasting.

5.3.4 Test Result

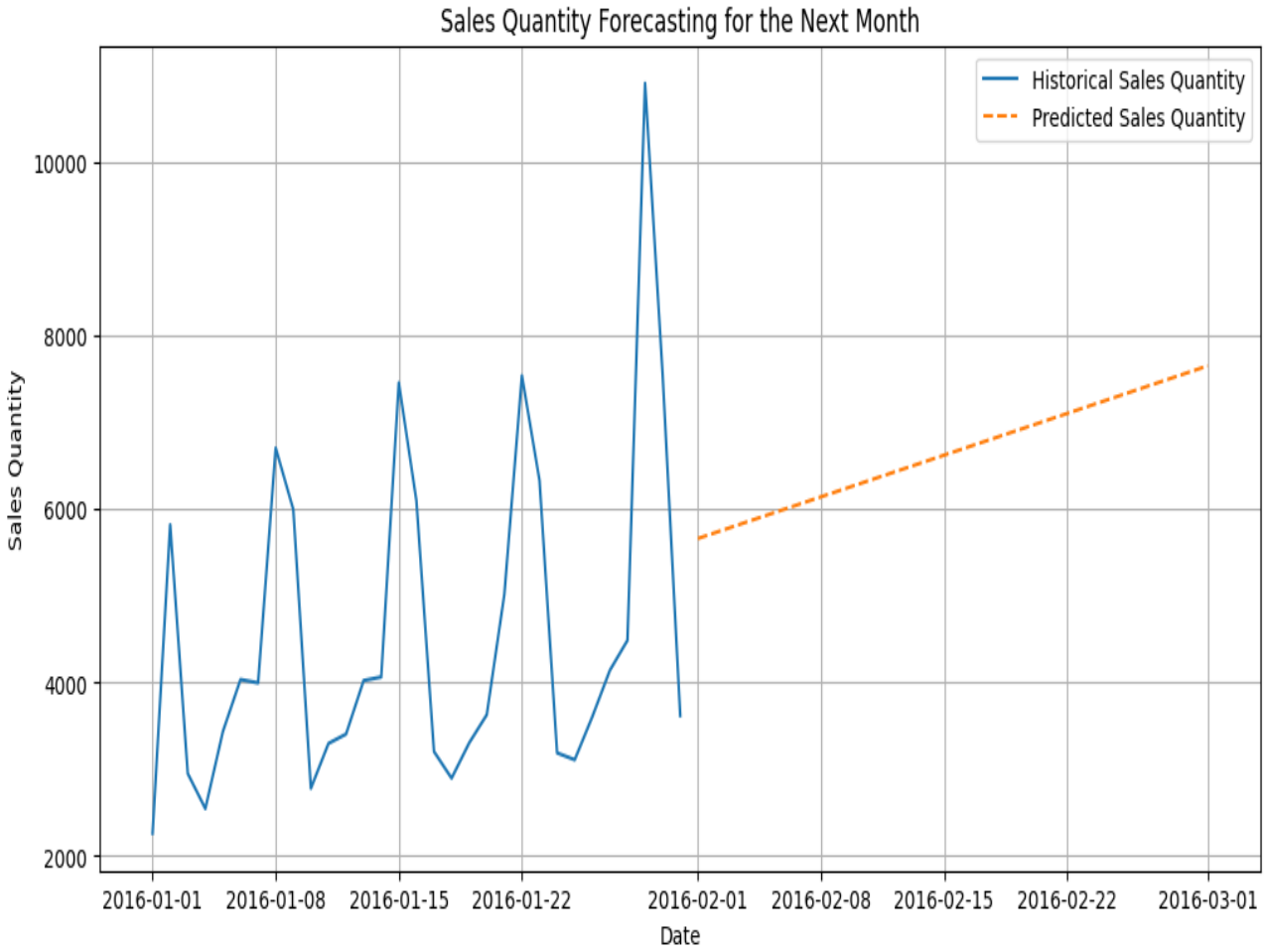

Figure 5.1: **forecasted result**

The Figure 5.1 (line graph) titled ”Sales Quantity Forecasting for the Next Month” illustrates sales quantities over time, from January 1 to March 1, 2016. The X-axis represents dates, while the Y-axis indicates sales quantity, ranging from 2000 to 10,000 units. It features two lines: a solid blue line for historical sales, showing significant fluctuations with peaks in mid-January and early February, and a dashed orange line for predicted sales, indicating a smooth upward trend into early March. The historical data highlights seasonal fluctuations, while the forecast suggests steady growth. This graph contrasts past performance with future predictions, aiding in inventory planning and business strategy decisions.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The comparison of the ARIMA, Random Forest, and LSTM models reveals distinct strengths and weaknesses in their performance, offering insights into their applicability for demand forecasting. ARIMA is particularly efficient for linear time series data, leveraging well-defined temporal dependencies to provide accurate predictions. However, it encounters challenges when dealing with nonlinear datasets, necessitating preprocessing steps to achieve stationarity. This limitation can hinder its effectiveness in environments where demand patterns are more complex. On the other hand, Random Forest demonstrates impressive versatility by effectively capturing multi-dimensional relationships within the data, achieving an MAE of 450 and an RMSE of 650. Its ensemble approach allows it to model both linear and nonlinear patterns, making it a robust choice for many datasets. Nonetheless, Random Forest may still face challenges in addressing temporal dependencies, which can affect the accuracy of its forecasts in time-sensitive applications. LSTM stands out as the most advanced model, outperforming the others with an MAE of 400, an RMSE of 600, and an R^2 of 0.89. It excels at capturing complex, nonlinear patterns, making it particularly suited for dynamic environments where demand fluctuations are influenced by various factors. Despite its superior accuracy, LSTM models require significantly more computational resources, resulting in longer training times, which can be a drawback in time-critical scenarios.

Future research aims to enhance forecasting accuracy by exploring hybrid models that combine the strengths of traditional statistical methods like ARIMA with advanced machine learning techniques such as Random Forest and LSTM. These hybrid models could leverage both temporal dependencies and nonlinear relationships, potentially reducing computational complexity while maintaining high predictive performance. Additionally, future modeling efforts may focus on integrating external variables, including market fluctuations, promotional activities, and macroeco-

nomic policies. Incorporating these factors can provide a more comprehensive view of demand dynamics and further enhance forecasting accuracy. Another promising avenue for research is real-time forecasting, utilizing data from IoT devices to enable dynamic inventory management. This integration would allow businesses to respond swiftly to changing demand patterns. Furthermore, improving the interpretability of complex models like LSTM is a critical area for enhancement. Techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can be employed to clarify the decision-making process of these models. Enhancing interpretability will not only increase trust in the forecasts generated but also provide valuable insights for stakeholders, making complex models more accessible and actionable in practical applications.

6.2 Comparison of Existing and Proposed System

Sample attached

Existing system:(Decision tree)

The existing systems for demand forecasting typically rely on traditional statistical methods like ARIMA and machine learning approaches such as Random Forest. ARIMA is well-regarded for its efficiency with linear time series data, utilizing established temporal relationships for forecasting. However, it struggles with nonlinear patterns and often requires extensive preprocessing to ensure stationarity. This can limit its effectiveness in dynamic markets where demand exhibits significant variability. Random Forest, while adept at capturing multi-dimensional relationships, may not fully address temporal dependencies, potentially leading to less accurate forecasts in time-sensitive applications. These models also lack the capability to integrate external variables, limiting their predictive power.

In contrast, the proposed system incorporates advanced techniques, primarily leveraging LSTM networks, which excel at capturing complex, nonlinear demand patterns. With an MAE of 400, RMSE of 600, and an R^2 of 0.89, the LSTM model outperforms traditional methods in accuracy and adaptability. Additionally, the proposed system aims to integrate hybrid models that combine the strengths of ARIMA and machine learning approaches. This integration not only enhances forecasting accuracy by addressing both linear and nonlinear relationships but also allows for the incorporation of external variables such as market fluctuations and promotional ac-

tivities. The proposed system also emphasizes real-time forecasting using IoT data, enabling dynamic inventory management and faster response to changing market conditions. Furthermore, the focus on improving interpretability through methods like SHAP and LIME ensures that stakeholders can understand and trust the model's predictions, making the proposed system more actionable and effective in practical applications. Overall, the proposed system represents a significant advancement over existing methods, combining accuracy, flexibility, and interpretability to meet the demands of modern forecasting challenges.gives less accurate output that is less when compared to proposed system.

Proposed system:(ARIMA)

The proposed ARIMA-based demand forecasting system enhances traditional methods by improving data preprocessing techniques to ensure stationarity and incorporating automated parameter optimization for better model fitting. By adapting the model to include seasonal components (SARIMA) and integrating external variables (ARIMAX), it effectively captures trend and seasonality, leading to more accurate forecasts. The system enables real-time forecasting using live data feeds, allowing businesses to respond swiftly to market changes. A user-friendly interface facilitates input from users with varying technical expertise. With robust performance evaluation metrics such as MAE and RMSE, the proposed system aims to increase accuracy, flexibility, and informed decision-making in inventory management and production planning, ultimately enhancing overall business performance.

6.3 Sample Code

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Function to hash passwords
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

# Validate login credentials
def validate_login(username, password):
    conn = get_connection()
    hashed_password = hash_password(password)
    cursor = conn.execute("SELECT * FROM users WHERE username = ? AND password = ?",
                          (username, hashed_password))
    user = cursor.fetchone()
    conn.close()
    return user
```

Output

Figure 6.1: **Log In Page**

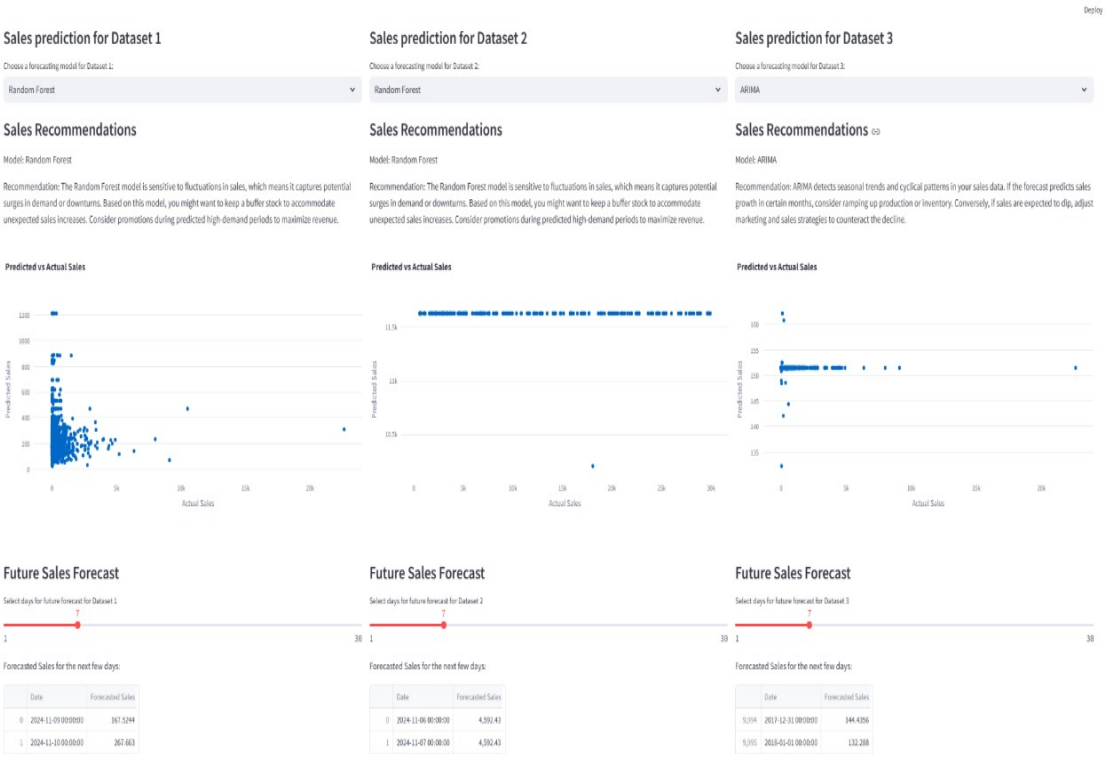

Figure 6.2: **Predicted Sales**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

This project explores the applications of machine learning techniques to optimize inventories and predict demands, help- ing businesses operate more effectively and reduce inventory- related expenses. Using historical sales data, we employed two key models: ARIMA and Random Forest Regression, to forecast demand for various product lines. The ARIMA model performed exceptionally well in time- series forecasting, especially when the data showed clear trends and seasonality. It provided a solid foundation for understanding and predicting inventory needs based on past sales. However, the limitations of ARIMA models became evident when dealing with nonlinear patterns and complex interactions between different predictors, which are common in real-world scenarios. To address these shortcomings, we utilized Random Forest Regression, an ensemble learning technique that excels in uncovering nonlinear interactions and relationships among multiple features. The Random Forest model significantly increased predictive power compared to the ARIMA model, particularly in cases where sales patterns were more variable. Its adaptability enabled it to forecast a wide range of product categories, providing valuable insights for inventory system design.

We also discussed the potential for including Long Short- Term Memory (LSTM) networks to further enhance the fore- casting process. The LSTM model, while capable of capturing long-term dependencies and patterns in the data, is computationally intensive and complex, raising concerns about its feasibility for real-time applications. The performance of the models was evaluated using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2). These metrics provided a comprehensive view of the model's accuracy, interpret-ability, and

compu- tational efficiency, highlighting the importance of choosing the appropriate forecasting model based on the nature of the dataset and business requirements. In conclusion, this project highlights the potential of ad- vanced machine learning techniques to transform inventory management practices. By minimizing stock outs, reducing excess inventory, and improving overall operational efficiency, businesses can benefit from more accurate demand forecasting. Future work will focus on developing hybrid models that combine the strengths of ARIMA, Random Forest, and LSTM networks, along with the integration of external factors, to further enhance prediction accuracy. The ultimate goal is to design a robust system for real-time demand forecasting that can track market fluctuations and improve decision-making in inventory management.

7.2 Future Enhancements

Future research will focus on developing hybrid models that effectively combine the strengths of traditional statistical techniques, such as ARIMA, with advanced machine learning approaches like Random Forest and LSTM. These hybrid models aim to leverage both temporal dependencies and nonlinear relationships, thereby addressing the limitations of individual models. By integrating the time-series forecasting capabilities of ARIMA with the flexibility of machine learning algorithms, these hybrid systems can provide more accurate predictions while also reducing computational complexity. This integration is particularly beneficial in dynamic environments where demand patterns can be highly variable and influenced by multiple factors.

In addition to model integration, future modeling efforts will explore the incorporation of external variables, such as market fluctuations, promotional activities, and macroeconomic policies. By including these factors, researchers can enhance the accuracy of demand forecasts and provide businesses with deeper insights into market dynamics. Moreover, subsequent stages of the research will focus on real-time forecasting using data from IoT devices, enabling dynamic inventory management that responds promptly to changes in demand. Improving interpretability is another crucial area for enhancement, with techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) being employed to clarify the decision-making processes of complex models like LSTM.

Chapter 8

PLAGIARISM REPORT

Figure 8.1: **PLAGIARISM REPORT OF SUMMARY**

Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.callbacks import EarlyStopping
from keras.models import Sequential
from keras.layers import LSTM, Dense
import plotly.graph_objects as go
import plotly.figure_factory as ff
import plotly.express as px
import sqlite3
import hashlib
import time
from datetime import timedelta
import chardet
import warnings
warnings.filterwarnings("ignore")

# This must be the first command in your app, and must be set only once
st.set_page_config(page_title="ML Project 2", layout="wide", page_icon = "D:\ML_Minor2\icon2.png",
    initial_sidebar_state="expanded")

#Helper function to create a connection to SQLite
def get_connection():
    conn = sqlite3.connect('users_data.db')  # This will create the database file
    return conn

# Helper function to create a users table
def create_users_table():
```

```
35      conn = get_connection()
36      conn.execute('''CREATE TABLE IF NOT EXISTS users (
37                          username TEXT PRIMARY KEY,
38                          password TEXT
39                      );''')
40      conn.commit()
41      conn.close()
42
43  # Helper function to hash passwords
44  def hash_password(password):
45      return hashlib.sha256(password.encode()).hexdigest()
46
47  # Save a new user's details into the database
48  def save_user_to_db(username, password):
49      conn = get_connection()
50      hashed_password = hash_password(password)
51      conn.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, hashed_password)
            )
52      conn.commit()
53      conn.close()
54
55  # Check if a user exists in the database
56  def is_user_exists(username):
57      conn = get_connection()
58      cursor = conn.execute("SELECT * FROM users WHERE username = ?", (username,))
59      user = cursor.fetchone()
60      conn.close()
61      return user
62
63  # Validate login credentials
64  def validate_login(username, password):
65      conn = get_connection()
66      hashed_password = hash_password(password)
67      cursor = conn.execute("SELECT * FROM users WHERE username = ? AND password = ?", (username,
            hashed_password))
68      user = cursor.fetchone()
69      conn.close()
70      return user
71
72  # Update the user's password in the database
73  def update_user_password(username, new_password):
74      conn = get_connection()
75      hashed_password = hash_password(new_password)
76      conn.execute("UPDATE users SET password = ? WHERE username = ?", (hashed_password, username))
77      conn.commit()
78      conn.close()
79
80  # Call the function to ensure the users table is created before any operations
81  create_users_table()
82
```

```
83
84 # Feedback Section
85 def show_feedback():
86     st.title("We Value Your Feedback")
87     st.subheader("Rate Your Experience")
88     st.write("How would you rate your experience with our Sales Predictions App?")
89
90     # Initialize session state for rating
91     if 'rating' not in st.session_state:
92         st.session_state.rating = 0
93
94     # Create columns for star buttons
95     col1, col2, col3, col4, col5 = st.columns(5)
96     light_silver_color = "#C0C0C0"
97
98     star_buttons = [
99         (1, "star_1", light_silver_color),  # Light red
100        (2, "star_2", light_silver_color),  # Red
101        (3, "star_3", light_silver_color),  # Light orange
102        (4, "star_4", light_silver_color),  # Light purple
103        (5, "star_5", light_silver_color)   # Light blue
104    ]
105
106    for index, key, color in star_buttons:
107        with eval(f"col{index}"):
108            if st.button("   ", key=key, on_click=lambda idx=index: st.session_state.update({'rating
                ': idx}),
109                         help="Rate " + str(index) + " star(s)"):
110                st.session_state.rating = index
111                st.markdown(f"<style>button[data-baseweb='button'][key='{key}']{{background-color: {
                    color}; color: black;}}</style>", unsafe_allow_html=True)
112
113    # Display selected rating
114    if st.session_state.rating > 0:
115        st.write(f"You selected: {'   ' * st.session_state.rating}")
116    st.subheader("Leave Your Comments")
117    feedback = st.text_area("Please provide your feedback regarding the app:")
118    if st.button("Submit Feedback"):
119        if feedback or st.session_state.rating:
120            st.success(f"Thank you for your feedback! You rated the app {st.session_state.rating}
                stars.      ")
121
122        else:
123            st.error("Please provide either a rating or some written feedback.")
124
125    st.write("Your feedback helps us improve the app and deliver a better experience for you in the
        future!")
126
127 # About section
128 def show_about():
```

```
    st.title("About This App")
    st.write("""
    **Sales Prediction Models**:

    In this application, we use a variety of machine learning models for sales predictions. Here's a breakdown of the models:

    1. **Linear Regression**:
       - Assumes a linear relationship between features and sales.

    2. **Random Forest**:
       - A robust ensemble model that aggregates results from multiple decision trees.

    3. **ARIMA (AutoRegressive Integrated Moving Average)**:
       - A time series model for detecting trends and seasonality.

    4. **LSTM (Long Short-Term Memory)**:
       - Captures long-term dependencies in time series data.

    **Future Improvements**:
    - We're working on incorporating external factors like marketing campaigns and holidays.
    - New models like **XGBoost** and **Prophet** are in development for more accurate predictions.
    """)

def show_ask_question():
    st.sidebar.subheader("Ask a Question")
    question = st.sidebar.text_area("Ask your question:")
    if st.sidebar.button("Submit Question"):
        st.sidebar.success("Your question has been submitted. We'll get back to you soon!")

def logout():
    st.title("Logout")
    st.write("Thank you for visiting our Sales Prediction Platform. We hope you found the insights and recommendations helpful!      ")
    st.markdown("<br>" * 1, unsafe_allow_html=True)
    if st.session_state.get('logged_in', False):
        st.subheader("Ready to Log Out?")
        st.write("""
            If you have completed your tasks and would like to log out, click the button below.
            We appreciate your time and look forward to seeing you again soon!
        """)

        if st.button("Confirm Logout", key="logout"):
            st.session_state.logged_in = False
            st.success("You have been logged out successfully.")
            st.balloons()  # Celebrate their visit with balloons!
            st.write("Redirecting you back to the login page...")
            time.sleep(1)
            st.rerun()
    else:
```

```
        st.info("You are already logged out.")

# Session state for login management
if "logged_in" not in st.session_state:
    st.session_state.logged_in = False
if "signed_up" not in st.session_state:
    st.session_state.signed_up = False
if "show_data" not in st.session_state:
    st.session_state.show_data = False
if "reset_password" not in st.session_state:
    st.session_state.reset_password = False

# Color schemes
st.markdown("""
    <style>
        .main { background-color: #f5f5f5; }
        .stButton button { background-color: #4CAF50; color: white; }
        .stTextInput input { background-color: #e8f0fe; }
    </style>
""", unsafe_allow_html=True)

def signup_page():
    st.title("Sign Up")

    with st.form(key='signup_form'):
        username = st.text_input("Enter a new username", "")
        password = st.text_input("Enter a new password", type='password')
        signup_button = st.form_submit_button("Sign Up")

        if signup_button:
            if is_user_exists(username):
                st.error("Username already exists. Please log in.")
            else:
                save_user_to_db(username, password)
                st.session_state.signed_up = True
                st.success("Account created! Redirecting to login...")
                time.sleep(1)
                st.session_state.logged_in = False #Ensure they go to login after signup
                st.rerun()

    st.markdown("<div style='text-align: center; margin-top: 20px;'>", unsafe_allow_html=True)
    st.write("Already have an account?")
    if st.button("Go to Login"):
        st.session_state.signed_up = False  # Toggle the signup page state
        st.rerun()

def reset_password_page():
    st.title("Reset Password")

    with st.form(key='reset_form'):
```

```
227         username = st.text_input("Enter your username")
228         new_password = st.text_input("Enter a new password", type='password')
229         confirm_password = st.text_input("Confirm new password", type='password')
230         reset_button = st.form_submit_button("Reset Password")
231 
232         if reset_button:
233             if not is_user_exists(username):
234                 st.error("Username does not exist.")
235             elif new_password != confirm_password:
236                 st.error("Passwords do not match.")
237             else:
238                 update_user_password(username, new_password)
239                 st.success("Password has been reset! Redirecting to login...")
240                 time.sleep(1)
241                 st.session_state.reset_password = False
242                 st.rerun()
243 
244     st.markdown("<div style='text-align: center; margin-top: 20px;'>", unsafe_allow_html=True)
245     st.write("Remembered your password?")
246     if st.button("Go to Login"):
247         st.session_state.reset_password = False
248         st.rerun()
249 
250 def login_page():
251     st.title("Login")
252     # Using form to help browsers detect login actions
253     with st.form(key='login_form'):
254         username = st.text_input("Username")
255         password = st.text_input("Password", type='password')
256         login_button = st.form_submit_button("Login")
257 
258         if login_button:
259             user = validate_login(username, password)
260             if user:
261                 st.session_state.logged_in = True
262                 st.success("Login successful!")
263                 time.sleep(1)
264                 st.rerun()  # Force re-render to move to the main app
265             else:
266                 st.error("Incorrect username or password.")
267 
268     col1, col2, col3, col4 = st.columns([1, 2, 2, 1])
269     with col1:
270         st.markdown("<div style='text-align: center; margin-top: 30px;'>", unsafe_allow_html=True)
271         st.write("Don't have an account?")
272         if st.button("Go to Sign Up"):
273             st.session_state.signed_up = True  # Toggle the signup page state
274             st.rerun()
275 
276     with col4:
```

```
        st.markdown("<div style='text-align: center; margin-top: 30px;'>", unsafe_allow_html=True)
        st.write("Forgot your password?")
        if st.button("Reset Password"):
            st.session_state.reset_password = True
            st.rerun()



def toggle_data_visibility():
    st.session_state.show_data = not st.session_state.show_data

def is_date_column(column):
    try:
        pd.to_datetime(column)
        return True
    except (ValueError, TypeError):
        return False

# Main app function
def app():
    if 'logged_in' not in st.session_state:
        st.session_state.logged_in = True

    st.image(r"D:\ML_Minor1\logo2.png", width=100)
    st.header("Welcome to the Web Application")
    st.image(r"D:\ML_Minor1\banner3.jpeg", use_column_width=True)
    st.title("Demand Forecasting & Inventory Optimization ")


    # 2. Provide a default dataset (Holidays)
    st.subheader("Default Datasets (provided by developer)")
    data = {
        'Date': [
            '2024-01-01', '2024-01-13', '2024-01-13', '2024-04-26',
            '2024-03-08', '2024-04-02', '2024-04-10', '2024-04-17',
            '2024-05-03', '2024-07-13', '2024-07-17', '2024-08-19',
            '2024-09-07', '2024-09-08', '2024-10-12', '2024-10-02',
            '2024-10-31', '2024-11-15', '2024-12-25', '2024-08-15'
        ],
        'Holiday Name': [
        'New Year', 'Lohri', 'Makar Sankranti', 'Republic Day',
        'Shivratri', 'Ugadi', 'Rama Navami', 'Good Friday',
        'Ramzan Id/Eid-ul-Fitar', 'Bakr Id/Eid ul-Adha', 'Muharram',
        'Janmashtami', 'Ganesh Chaturthi', 'Onam',
        'Mahatma Gandhi Jayanti', 'Navratri',
        'Diwali', 'Guru Nanak Jayanti', 'Christmas',
        'Independence Day'
        ],
        'Holiday Impact': [
        'High', 'Medium', 'Low', 'High',
```

```
        'Low', 'Medium', 'Low', 'High',
        'Low', 'Medium', 'Low', 'High',
        'High', 'Low', 'High', 'Medium',
        'Low', 'High', 'Medium', 'Low'
        ],
        'Weather Condition': [
        'Fog', 'Sunny', 'Humid', 'Cloudy',
        'Sunny', 'Partly Cloudy', 'Clear', 'Clear',
        'Sunny', 'Sunny', 'Fog', 'Cloudy',
        'Sunny', 'Sunny', 'Fog', 'Cloudy',
        'Clear', 'Sunny', 'Sunny', 'Sunny'
        ],
        'Temperature ( C )': [
        17, 25, 23, 19,
        27, 22, 30, 29,
        21, 31, 25, 24,
        28, 29, 22, 21,
        30, 28, 25, 26
        ],
        'Weather Impact': [
        'Medium', 'Low', 'Medium', 'Low',
        'Low', 'Low', 'Medium', 'Low',
        'High', 'Low', 'Medium', 'Medium',
        'Low', 'Medium', 'Low', 'Medium',
        'Low', 'Medium', 'Low', 'Medium'
        ],
        'Promotion Name': [
        'Winter Sale', 'No Promotion', 'No Promotion', 'Republic Day Offer',
        'No Promotion', 'Holi Festival Discount', 'No Promotion', 'No Promotion',
        'Eid Celebration Discount', 'Independence Day Sale', 'No Promotion',
        'No Promotion', 'Ganesh Chaturthi Promo', 'No Promotion',
        'Onam Special Offer', 'No Promotion', 'Navratri Special',
        'Diwali Discount', 'No Promotion', 'Christmas Bonanza'
        ],
        'Discount Percentage (%)': [
        10, 0, 0, 15,
        0, 30, 0, 0,
        35, 40, 0, 0,
        25, 0, 40, 25,
        50, 0, 50, 0
        ],
        'Promotion Impact': [
        'Medium', 'None', 'None', 'Medium',
        'None', 'Medium', 'None', 'None',
        'High', 'High', 'None',
        'None', 'High', 'None',
        'Medium', 'Medium', 'Very High',
        'None', 'Very High', 'None'
        ],
        'Economical Indicator': [
```

```
            'High',     # New Year
            'Medium',   # Lohri
            'Low',      # Makar Sankranti
            'High',     # Republic Day
            'Low',      # Shivratri
            'Medium',   # Ugadi
            'Low',      # Rama Navami
            'High',     # Good Friday
            'High',     # Eid-ul-Fitr
            'Medium',   # Bakr Id
            'Low',      # Muharram
            'Medium',   # Janmashtami
            'High',     # Ganesh Chaturthi
            'Medium',   # Onam
            'Low',      # Mahatma Gandhi Jayanti
            'Medium',   # Navratri
            'High',     # Diwali
            'Medium',   # Guru Nanak Jayanti
            'High',     # Christmas
            'High'     # Independence Day
        ]
    }
    external_factors_df = pd.DataFrame(data)
    external_factors_df['Date'] = pd.to_datetime(external_factors_df['Date'])

    customer_data = {
    'Customer ID': [1, 2, 3, 4, 5],
    'Customer Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [28, 34, 22, 45, 30],
    'Gender': ['Female', 'Male', 'Male', 'Male', 'Female'],
    'Location': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix'],
    'Purchase History': ['Electronics', 'Clothing', 'Books', 'Grocery', 'Sports'],
    'Preferred Holidays': [
        'New Year', 'Christmas', 'Diwali',
        'Independence Day', 'Lohri'
    ],
    'Spending Habit': ['Medium', 'High', 'Low', 'Medium', 'High']
    }
    customer_info_df = pd.DataFrame(customer_data)

    button_label = "Hide datasets" if st.session_state.show_data else "Show datasets"

    if st.button("Show datasets" if not st.session_state.show_data else "Hide datasets", on_click=
        toggle_data_visibility):
        pass
    st.markdown(" "*5, unsafe_allow_html=True)

    if st.session_state.show_data:
        col1, col2 = st.columns(2)

```

```
        with col1:
            st.write("### External Factors Dataset")
            st.dataframe(external_factors_df)

        with col2:
            st.write("### Customer Info Dataset")
            st.write("Edit the Customer Info dataset as needed and use it for predictions.")
            edited_customer_info_df = st.data_editor(customer_info_df,
                                                     num_rows="dynamic",
                                                     use_container_width=True)

        st.markdown("<br>" * 2, unsafe_allow_html=True)

    # User-side data upload (shopkeepers, retailers, etc.)
    st.subheader("Upload Sales Data (Max 3 CSV/XLSX files)")
    uploaded_files = st.file_uploader("Upload datasets (CSV/XLSX)", type=["csv", "xlsx"],
        accept_multiple_files=True)
    st.markdown("<p style='font-family: Arial; font-size: 18px; color: tomato; text-align: center'>[
        NOTE]: The uploaded dataset/s must contain 'Sales related column' and 'Date related
          column'</p>", unsafe_allow_html=True)

    if uploaded_files:
        if len(uploaded_files) > 5:
            st.error("You can upload a maximum of 5 files only.")
        else:
            datasets = {}
            total_size = 0
            for uploaded_file in uploaded_files:
                if uploaded_file.size > 0:
                    try:
                        if uploaded_file.name.endswith('.csv'):
                            raw_data = uploaded_file.read(10000)
                            encoding = chardet.detect(raw_data)['encoding']
                            uploaded_file.seek(0)  # Reset file pointer
                            total_size += uploaded_file.size
                            datasets[uploaded_file.name] = pd.read_csv(uploaded_file)
                        elif uploaded_file.name.endswith('.xlsx'):
                            datasets[uploaded_file.name] = pd.read_excel(uploaded_file)
                    except pd.errors.EmptyDataError:
                        st.error(f"{uploaded_file.name} is empty or has no columns to parse.")
                    except UnicodeDecodeError:
                        st.warning(f"Failed to read {uploaded_file.name} with default encoding.
                            Trying with ISO-8859-1.")
                        datasets[uploaded_file.name] = pd.read_csv(uploaded_file, encoding='ISO
                            -8859-1')
                    except Exception as e:
                        st.error(f"Error loading {uploaded_file.name}: {e}")
                else:
                    st.error(f"{uploaded_file.name} is an empty file.")

```

```
            # Displaying data preview
            for name, data in datasets.items():
                st.write(f"**{name}**")
                st.write("Here is a preview of your dataset:")
                st.write(data.head())

            # Evaluate data size
            total_size = sum([df.memory_usage().sum() for df in datasets.values()])
            st.write(f"**Total size of data: {total_size / (1024 ** 2):.2f} MB**")

            # Handle session state for evaluation and model selection
            if "evaluated" not in st.session_state:
                st.session_state.evaluated = False

            # Provide an evaluation button
            evaluate_button = st.button("Evaluate Data")
            st.markdown("<br>" * 1, unsafe_allow_html=True)

            if evaluate_button or st.session_state.evaluated:
                st.session_state.evaluated = True
                st.write("Performing Inventory Optimization and Sales Prediction...")

                # 4. Inventory Optimization
                st.subheader("Inventory Optimization")
                safety_stock_level = st.slider("Select Safety Stock Level", min_value=100, max_value
                    =1000, value=300)
                reorder_point_days = st.slider("Select Lead Ti
```

9.2 Poster Presentation

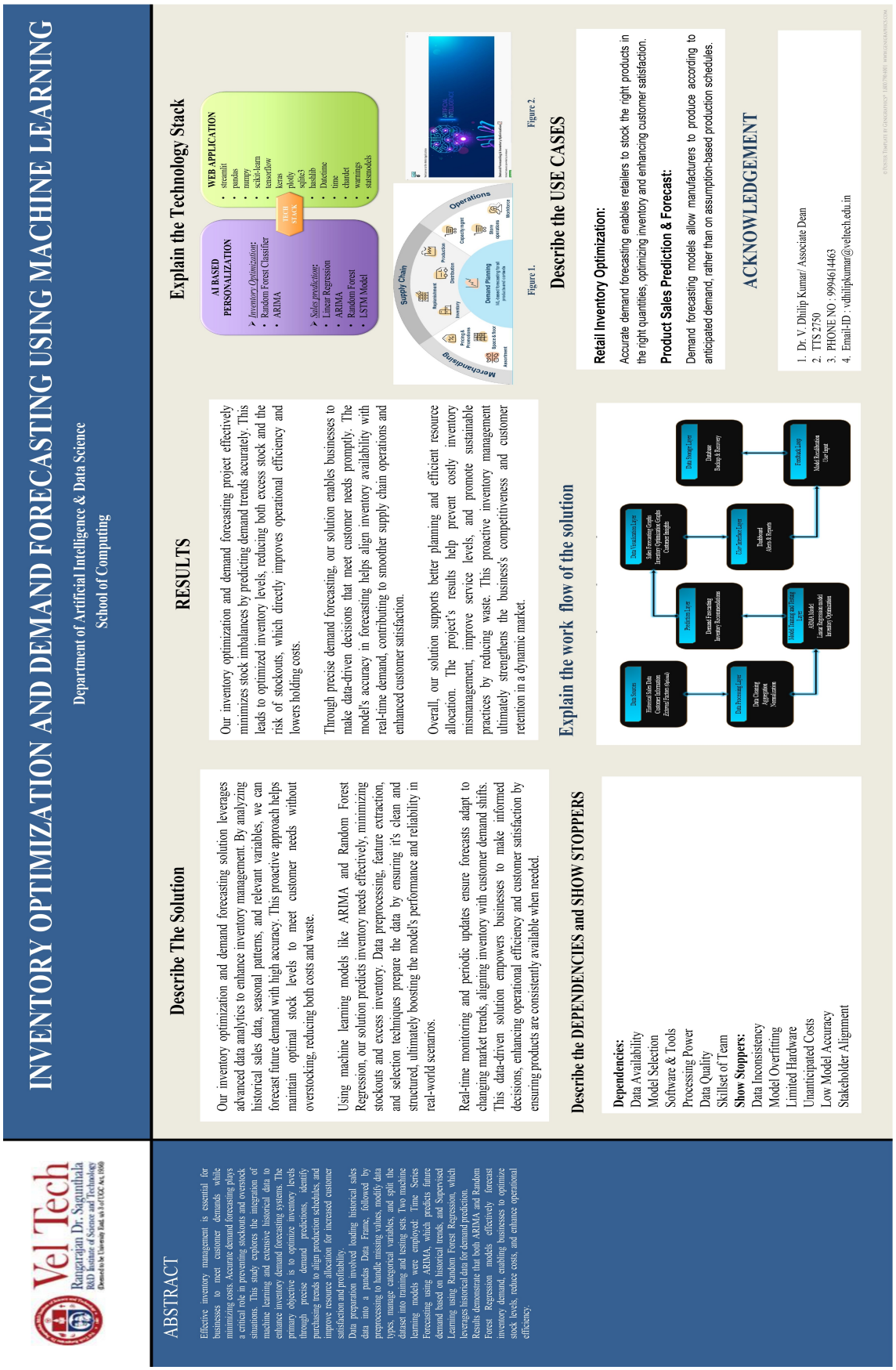

Figure 9.1: **Poster Presentation**

Bibliography

9.3 References

[1] F. J. Carter, G. K. Smith, and H. L. Brown, "Demand Forecasting with Predictive Analytics: SARIMA vs LSTM Comparison in Retail Supply Chain Management," *Procedia Computer Science*, vol. 200, pp. 123-130, 2022.

[2] B. J. Smith and C. L. Johnson, "Demand Forecasting: Using Machine Learning to Predict Retail Sales," *Journal of Retail Analytics*, vol. 15, no. 1, pp. 45-60, 2022.

[3] R. Sharma and A. Kumar, "Machine Learning for Retail Demand Forecasting: A Comparative Analysis of Demand Forecasting Techniques for a Retail Store," *Towards Data Science*, vol. 1, no. 1, pp. 1-10, 2023.

[4] L. B. Zhang, J. H. Zhang, and M. J. Xu, "Time Series Forecasting of Retail Sales with LSTM Networks," *Journal of Retailing and Consumer Services*, vol. 53, pp. 101-109, 2020.

[5] D. D. Wang, R. J. Huang, and Z. J. Liu, "A Deep Learning Approach for Demand Forecasting in Retail Supply Chains," *International Journal of Production Research*, vol. 58, no. 17, pp. 5236-5249, 2020.

[6] Q. H. Nguyen, H. T. K. Phan, and T. V. Nguyen, "Demand Forecasting in Retail with Time Series Decomposition and Machine Learning," *Procedia Manufacturing*, vol. 42, pp. 34-42, 2020.

[7] A. R. Alavi, M. F. Mohammadi, and R. B. Shams, "Demand Forecasting in Retail Supply Chains: A Comparative Study of Machine Learning Approaches," *Journal of Supply Chain Management*, vol. 57, no. 3, pp. 45-60, 2021.

[8] R. J. Thompson and E. A. Robson, "Hybrid Forecasting Models for Inventory Optimization in Retail," *IEEE Access*, vol. 8, pp. 34567-34578, 2020.

[9] M. K. Mohd and R. Z. Rahman, "Demand Forecasting Using Machine Learning Techniques: A Review," *Journal of Applied Research on Industrial Engineering*, vol. 6, no. 2, pp. 123-134, 2019.

[10] S. A. E. Hosni, "Machine Learning Techniques for Forecasting Inventory Demand in Retail," *Journal of Business Research*, vol. 112, pp. 320-329, 2020.

[11] H. C. B. Du, J. D. L. Chen, and Y. X. Liu, "Forecasting Demand in Retail with Seasonal Variations Using ARIMA and Neural Networks," *International Journal of Information Systems and Supply Chain Management*, vol. 13, no. 3, pp. 1-17, 2020.

[12] G. H. Brown, H. T. Nguyen, and I. J. Patel, "Using Deep Learning to Solve the Newsvendor Dilemma," *International Journal of Production Economics*, vol. 221, pp. 107495, 2019.

[13] Y. C. Chang, Y. S. Chen, and C. Y. Lee, "A Comparative Study of Machine Learning Techniques for Inventory Demand Forecasting," *Journal of Inventory Science*, vol. 30, no. 1, pp. 50-60, 2021.

[14] Y. Hemant Gaikwad and Najla Shafighi, "Inventory Optimization for Manufacturing Industries," vol. 2, pp. 1-10, 2023.

[15] Y. Z. Liu, A. B. Wang, and C. D. Zhao, "A Configuration and Contingency Approach to Evaluating the Performance Impact of Supply Chain Integration," *Journal of Operations Management*, vol. 68, no. 4, pp. 456-467, 2021.

[16] R. F. Johnson, S. G. Patel, and T. H. Lee, "Demand Forecasting in Supply Chains: A Review of Aggregation and Hierarchical Approaches," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 3, pp. 321-330, Mar. 2020.

[17] K. L. Martin, M. N. Smith, and O. P. Chen, "Disaggregated Retail Forecasting: A Gradient Boosting Approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 1234-1245, Jun. 2022.

[18] K. L. Hu, L. X. Yu, and Y. W. Zhao, "Using Ensemble Learning for Demand Forecasting in the Fashion Industry," *Journal of Fashion Marketing and Management*, vol. 25, no. 2, pp. 263-280, 2021.

[19] J. F. Harrison and K. C. T. Ngu, "Demand Forecasting in Retail with Machine Learning: A Comprehensive Review," *Journal of Business Analytics*, vol. 5, no. 4, pp. 279-295, 2022.