KG COLLEGE OF ARTS AND SCIENCE
Autonomous Institution | Affiliated to Bharathiar University
Accredited with A++ Grade by NAAC
ISO 9001:2015 Certified Institution
KGiSL Campus, Saravanampatti, Coimbatore – 641 035

PROJECT REPORT

FUEL SALES PREDICTION BASED ON HISTORICAL DATA

Submitted to Bharathiar University in partial fulfillment of the requirements for the award of the of degree

Bachelor of Computer Applications

Submitted by

DHANVANTH. S

2222J1374

Under the Supervision and Guidance of

Dr. Ms.G.Priyadharshini M.Sc., M.Phil., Ph.D.,

Assistant Professor

Department of Computer Applications

MARCH 2025

DECLARATION

I declare that the work which is being presented in this project report entitled **“Fuel Sales Prediction Based on Historical data**” submitted to the Department of Computer Applications, KG College of Arts and Science, Saravanampatti, Coimbatore for the award of the degree of Bachelor of Computer Applications of the Bharathiar University, is an authentic record of my work carried out under the supervision and guidance of **Dr. G. Priyadharshini M.Sc., M.Phil., Ph.D.,** I have not plagiarized or submitted the same work for the award of any other degree.

March 2025

Coimbatore

DHANVANTH.S

CERTIFICATE

This is to certify that the project entitled **"Fuel Sales Prediction Based on Historical data"** submitted to Bharathiar University in partial fulfillment for the award of the degree of Bachelor of Computer Applications is a record of original work done by **DHANVANTH.S (22222JA09)** during the period of study in **KG College of Arts and Science** under the supervision of **Dr.P.Vijayakumar M.Sc., M.Phil., Ph.D., Associate Professor & Head** Department of Computer Applications.

Place: Coimbatore

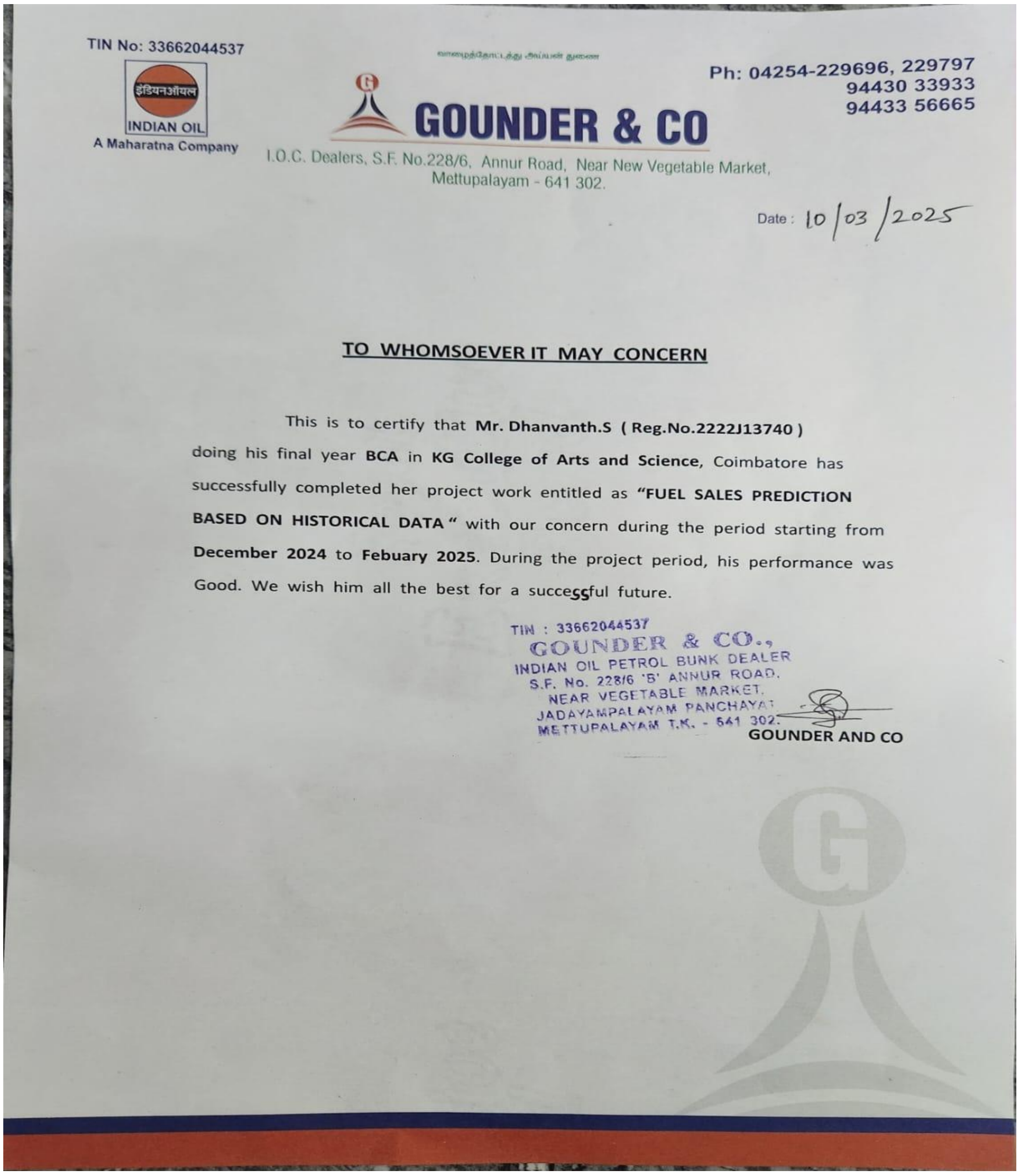
Date:

SIGNATURE OF THE GUIDE **SIGNATURE OF THE HOD**

(College Seal)

Submitted for the Viva-Voce Examination held on ____________________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**



TIN No: 33662044537

INDIAN OIL
A Maharatna Company

வாழைத்தோட்டத்து அய்யன் துணை

GOUNDER & CO

I.O.C. Dealers, S.F. No.228/6, Annur Road, Near New Vegetable Market,
Mettupalayam - 641 302.

Ph: 04254-229696, 229797
94430 33933
94433 56665

Date : 10/03/2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Dhanvanth.S (Reg.No.2222J13740)** doing his final year **BCA** in **KG College of Arts and Science**, Coimbatore has successfully completed her project work entitled as **"FUEL SALES PREDICTION BASED ON HISTORICAL DATA "** with our concern during the period starting from **December 2024** to **Febuary 2025**. During the project period, his performance was Good. We wish him all the best for a successful future.

TIN : 33662044537
GOUNDER & CO.,
INDIAN OIL PETROL BUNK DEALER
S.F. No. 228/6 'B' ANNUR ROAD,
NEAR VEGETABLE MARKET,
JADAYAMPALAYAM PANCHAYAT
METTUPALAYAM T.K. - 641 302.

[illegible]

GOUNDER AND CO

ACKNOWLEDGEMENT

I express my sincere thanks to **Dr.Ashok Bakthavathsalam, Managing Trustee** for allowing me to do this course of study and to undertake this project work.

It is my pleasure to express my sincere thanks to **Dr. B. Vanitha, Secretary**, KG College of Arts and Science for her valuable thoughts.

I take this opportunity to convey my sincere thanks to **Dr. J.Rathinamala, Principal,** KG College of Arts and Science for her ethical encouragement throughout the course.

I would like to express my heartfelt gratitude to **Dr.S.Vidhya, Vice Principal**, KG College of Arts and Science, for her unwavering moral support throughout the course.

I would like to express my gratefulness to **Dr.P.Ajitha, Dean-Curriculum Development Cell**, KG College of Arts and Science, for her untiring support throughout the course.

I take this opportunity to convey my sincere thanks to **Dr.M.Usha, Dean-School of Computational Science,** KG College of Arts and Science for her moral support throughout the course.

I have great pleasure in acknowledging my thanks to **Dr.P.Vijayakumar Head, Department of Computer Applications**, KG College of Arts and Science for his encouragement and help throughout the course.

I express my sincere thanks to **Dr.Ms.G.Priyadharshini M.Sc., M.Phil., Ph.D.,** Associate Professor and Head, Department of Computer Applications for his constant encouragement and motivation throughout the project. I thank for his endless support and encouragement towards the work.

DHANVANTH.S

CONTENTS

SYNOPSIS

In competitive fuel industry, accurately predicting fuel sales is crucial for optimizing inventory, reducing wastage, and ensuring smooth operations. This project aims to develop a Fuel Sales Prediction System using historical data and machine learning techniques. The system analyzes past sales records, identifies trends, and considers external factors such as fuel prices, weather conditions, and seasonal demand fluctuations to generate accurate forecasts.

The proposed system leverages time-series forecasting models such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet to enhance prediction accuracy. The data is preprocessed to remove inconsistencies, and advanced statistical techniques are applied to extract meaningful insights. The model's output helps fuel station managers make data-driven decisions, improving stock management and reducing operational inefficiencies.

The system features an interactive web-based interface where users can upload data, view visualized trends, and download reports. It also integrates APIs to fetch real-time fuel prices and weather data, further improving prediction reliability. By implementing this system, fuel retailers can optimize sales strategies, prevent stock shortages, and enhance customer satisfaction. Overall, this AI-powered forecasting tool provides a scalable, accurate, and automated solution for fuel sales prediction, ensuring efficient fuel distribution and business growth. The system is designed to be scalable and adaptable, allowing integration with various fuel station management systems. By leveraging machine learning and cloud computing, the model continuously learns from new data, improving its forecasting accuracy over time. The use of a NoSQL database ensures efficient data storage and retrieval, handling large volumes of historical and real-time data seamlessly. The user-friendly interface enables fuel station managers to access sales trends, demand patterns, and predictive analytics in an intuitive dashboard. With these capabilities, the Fuel Sales Prediction System empowers businesses to make informed decisions, minimize financial risks, and optimize fuel distribution strategies for maximum efficiency.

1. INTRODUCTION

Fuel sales prediction plays a vital role in the fuel industry by helping station owners, suppliers, and distributors optimize their inventory and ensure a steady supply. Inaccurate forecasting can lead to fuel shortages, excess stock, or financial losses. By leveraging historical data and advanced predictive models, businesses can make data-driven decisions to manage demand fluctuations effectively. This project focuses on developing an accurate fuel sales prediction system to assist stakeholders in planning their operations efficiently.

The primary objective of this project is to analyze historical fuel sales data and predict future demand using machine learning and time-series forecasting techniques. Various factors, such as seasonal trends, economic conditions, fuel prices, weather patterns, and holidays, influence sales. Traditional statistical models like ARIMA and SARIMA are effective in capturing time-dependent trends, while machine learning models like XGBoost and deep learning methods such as LSTM can handle complex relationships between multiple variables. Additionally, Facebook Prophet is explored for its flexibility in handling business-related forecasting challenges.

To achieve accurate predictions, the project involves data collection, preprocessing, feature engineering, and model selection. Historical sales data is analyzed to identify trends, seasonality, and anomalies. The selected models are trained and evaluated based on performance metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The best-performing model is then used to generate future sales predictions, enabling better stock management and operational planning.

This project's implementation benefits fuel station owners by minimizing losses, reducing operational inefficiencies, and improving profitability. By integrating a robust predictive system, businesses can anticipate demand surges, adjust procurement strategies, and enhance customer satisfaction. The insights gained from this study can also be extended to other retail industries where demand forecasting plays a crucial role in inventory management and business decision-making.

1.1 ORGANIZATION PROFILE

At Gounder and Co, Mettupalayam, Coimbatore, we are committed to powering industries, businesses, and communities with high-quality non-renewable fuels. Specializing in the distribution and supply of petrol and diesel, we ensure reliable energy solutions that drive economic growth and everyday mobility.

With a strong foundation built on efficiency, safety, and customer satisfaction, we have established ourselves as a trusted name in the fuel industry. Our extensive distribution network, backed by state-of-the-art infrastructure and advanced logistics, ensures uninterrupted fuel supply to our customers, ranging from retail fuel stations to commercial enterprises.

Quality assurance is at the core of our operations. We adhere to stringent safety and environmental standards, ensuring that every drop of fuel meets the highest industry benchmarks. As we continue to expand, our focus remains on delivering cost-effective and sustainable energy solutions while minimizing environmental impact.

At Gounder and Co, we believe in innovation and excellence, striving to meet the ever-evolving demands of the fuel sector. Whether it’s powering vehicles, industries, or businesses, we are dedicated to fueling progress and reliability every step of the way.

1.2 SYSTEM SPECIFICATION

1.2.1 HARDWARE SPECIFICATION

● Hard Disk	: 512 GB SSD (Minimum) / 36Gb SSD (R
● Processor	: 12th Gen Intel(R) Core (TM) i5-12500H 2.50 GHz
● RAM Capacity	: 4 GB (Minimum) / 8 GB (Recommended)
● Network Speed	: 93 Mbps Speed
● System bus	: 64-bit

1.2.2 SOFTWARE SPECIFICATION

● Operating System	: Windows 11
● Front End	: Streamlit
● Back End	: Python
● IDE	: VS CODE
● ML Framework	: XGBoost , Prophet
● Data Base	: CSV File
● Libraries	: Numpy , Seaborn, Pandas, Matplotlib
● Styling	: CSS

1.3 SOFTWARE FEATURES

- **User-Friendly Interface** : Provides an interactive and responsive UI for easy navigation and visualization.
- **Time-Series Analysis & Prediction** : Supports multiple forecasting models to identify trends and seasonal patterns.
- **Database Management** : Utilizes MySQL for efficient storage and management of fuel sales data.
- **Machine Learning & Deep Learning Integration** : Implements advanced ML and deep learning models for accurate fuel sales forecasting.
- **Real-Time Data Processing** : Dynamically updates and processes sales data for real-time insights.
- **Browser Compatibility** : Fully optimized for smooth performance on Google Chrome and Microsoft Edge.
- **Notification System** : Uses React Hot Toast for real-time alerts on demand fluctuations and fuel shortages.
- **Unique ID Generation** : Implements UUID for generating unique transaction and record identifiers.
- **Model Deployment & API Integration** : Uses Flask or FastAPI to deploy models and provide API endpoints.
- **Data Visualization** : Displays sales trends and forecasts using Matplotlib and Seaborn.
- **Scalability & Performance** : Optimized for handling large datasets and supporting multiple fuel stations.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

The current fuel sales prediction system relies on manual estimation, where past sales records and human judgment determine future demand, often leading to inaccuracies. Traditional methods have limited consideration of external factors, such as fuel price fluctuations, weather conditions, and seasonal variations, making predictions unreliable. Additionally, there is a lack of real-time data processing, meaning sales trends are not dynamically updated, resulting in delayed decision-making. This inefficiency leads to poor inventory management, causing either fuel shortages or excess stock, ultimately impacting profitability and operational efficiency.

Drawbacks

- **Inaccurate Forecasting** : Manual estimation and basic statistical methods fail to provide precise sales predictions.
- **Ignores External Factors** : Fuel price changes, weather conditions, and seasonal trends are not considered in traditional methods.
- **No Real-Time Updates** : Lack of dynamic data processing results in delayed and ineffective decision-making.
- **Inventory Management Issues** : Poor demand prediction leads to fuel shortages or excess stock, causing financial losses.
- **High Operational Costs** ; Inefficient forecasting increases transportation and storage costs for fuel suppliers.
- **Limited Automation** : Most systems require manual intervention, reducing efficiency and increasing human errors.

2.2 PROPOSED SYSTEM

The proposed Fuel Sales Prediction System leverages machine learning and time-series forecasting techniques to provide accurate and automated fuel demand predictions. Unlike the existing system, this model considers historical sales data, external factors (fuel prices, weather conditions, holidays), and seasonal trends to enhance forecasting accuracy.

By implementing advanced models such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet, the system can analyze complex patterns and generate reliable predictions. The system also incorporates real-time data processing, ensuring dynamic updates for better decision-making.

With automated inventory management, fuel stations can optimize stock levels, reducing shortages and excess storage costs. Additionally, a user-friendly dashboard provides visual insights into fuel demand trends, allowing stakeholders to make informed business decisions.

FEATURES

- **Accurate Fuel Sales Prediction** : Utilizes advanced machine learning models (ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet) for precise demand forecasting.
- **Real-Time Data Processing** : Continuously updates and processes sales data dynamically for better decision-making.
- **Consideration of External Factors** : Incorporates fuel prices, weather conditions, holidays, and seasonal trends to improve forecasting accuracy.
- **Automated Inventory Management** : Helps optimize fuel stock levels, preventing shortages and reducing excess inventory costs.
- **User-Friendly Dashboard** : Provides interactive visualizations and reports for better business insights.
- **Scalable & High Performance** : Designed to handle large datasets efficiently, making it suitable for multiple fuel stations.
- **Automated Alerts & Notifications** : Sends real-time alerts for demand surges, low stock, or unusual sales patterns.

- **Seamless API Integration** : Offers RESTful APIs for easy integration with fuel station management systems.
- **Historical Data Analysis** : Evaluates past sales trends to identify patterns and improve long-term planning.
- **Cross-Platform Compatibility** – Works on Windows, macOS, and Linux with browser support for Google Chrome and Microsoft Edge.

3. SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

The Fuel Sales Prediction System follows a structured file design to ensure efficient data management, processing, and system functionality. The system consists of multiple files, categorized based on their purpose, including frontend, backend, database, and machine learning models**.**

Input Files:

- **Sales Data File (CSV/Excel)** : Contains historical fuel sales data, including date, quantity sold, fuel price, weather conditions, and other influencing factors.
- **External Factors Data (API/CSV)** : Stores fuel prices, weather reports, and holiday data for accurate forecasting.

Processing Files:

- **Data Preprocessing Module (Python – Pandas, NumPy)** : Cleans and processes input data, handling missing values and formatting for model training.
- **Feature Engineering Module** : Extracts meaningful features such as seasonal trends, demand spikes, and correlations with external factors.
- **Machine Learning Model Files (Python – ARIMA, SARIMA, XGBoost, LSTM, Prophet)** : Implements predictive models for fuel sales forecasting.

Database Files:

- **SQL Database (MySQL/PostgreSQL)** : Stores processed data, prediction results, and historical records for efficient retrieval.
- **User Authentication File** : Manages login credentials, user roles, and permissions for system security.

Output & Visualization Files:

- **Prediction Output File (JSON/CSV)** : Stores the forecasted sales data for future analysis.
- **Dashboard & Reports (React.js, Chart.js, Matplotlib, Seaborn)** : Displays interactive graphs, reports, and insights for better decision-making.

Deployment & Integration Files:

- **API Endpoints (Flask/FastAPI – Python)** : Provides RESTful API services for integrating predictions with external applications.
- **Frontend Files (HTML, CSS, JavaScript, Tailwind CSS, React.js)** : Manages user interactions and data visualization.

3.2 INPUT DESIGN

The Fuel Sales Prediction System requires structured and well-formatted input data to ensure accurate forecasting. The input design focuses on capturing essential data points, validating input values, and ensuring seamless data flow for processing.

Input Sources:

- **Historical Sales Data (CSV/Database)** : Includes past fuel sales records such as date, quantity sold, and fuel type.
- **External Factors (API/CSV)** : Fetches real-time data such as fuel prices, weather conditions, and holidays.
- **User Inputs (Web Interface)** : Allows users to upload custom sales data, adjust parameters, and configure forecasting settings.

Input Fields:

Table name : Fuel_Sales

Primary key : Sale_id

Field Name	Description	Data Type	Example
Date	Timestamp of fuel sales	Date	2024-03-07
Fuel Type	Type of fuel (Petrol/Diesel)	String	Petrol
Quantity Sold	Total liters sold	Float	1500.50
Fuel Price	Price per liter at the time	Float	105.75
Weather	Weather conditions (API-based)	String	Rainy
Holiday Flag	Indicates if it's a holiday	Boolean	Yes (1) / No (0)

Input Validation & Constraints:

- **Date Validation** : Ensures the correct date format (YYYY-MM-DD) and prevents duplicate entries.
- **Numeric Range Checks** : Validates that sales quantity and fuel price are within a logical range.
- **Data Completeness** : Requires all essential fields to be filled to prevent missing values in predictions.
- **Real-Time API Validation** : Fetches accurate external data (fuel prices, weather, holidays) to improve forecast accuracy.

Data Input Methods:

- **Manual Entry via Web Dashboard** : Users can input or modify data directly through a user-friendly interface.

- **Bulk Upload (CSV/Excel)** : Allows uploading large datasets for efficient processing.
- **Automated API Integration** : Fetches real-time external data to enhance prediction accuracy.

3.3 OUTPUT DESIGN

The Fuel Sales Prediction System provides well-structured and user-friendly outputs to help stakeholders make informed decisions. The output is designed for clarity, accuracy, and real-time accessibility, ensuring that sales trends, predictions, and insights are presented effectively.

Output Types:

- **Fuel Sales Forecast Reports** : Displays predicted sales for upcoming days/weeks/months.
- **Visual Data Insights** : Graphs and charts for trend analysis.
- **Real-Time Alerts & Notifications** : Warnings about demand fluctuations, low stock, or unusual trends.
- **Exportable Reports** : Downloadable reports in CSV, Excel, or PDF format.

Output Formats:

- **Web Dashboard (React.js, Chart.js, Matplotlib, Seaborn)** : Displays real-time sales predictions and trends.
- **Graphical Representation** : Line charts, bar graphs, and pie charts for sales trends and seasonal variations.
- **Tabular Reports** : Detailed numerical breakdowns of past and predicted fuel sales.
- **Downloadable Reports** : Users can export data in **CSV, Excel, and PDF** formats for record-keeping and analysis.

Output Delivery Methods:

- **On-Screen Display** : Interactive web dashboard for quick insights.

- **Email & Notifications** : Alerts sent to fuel station managers about expected demand fluctuations.
- **API Integration** : Allows integration with third-party applications for automated decision-making**.**

3.4 SYSTEM DEVELOPMENT

The Fuel Sales Prediction System is developed using the Agile methodology, ensuring iterative improvements and seamless functionality. The system leverages React.js for the frontend, Node.js with Express.js for the backend, and MySQL for efficient data storage. Machine learning models such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet are integrated for accurate fuel demand forecasting, while data processing is handled using Python libraries (Pandas, NumPy, Scikit-learn). The system comprises essential modules, including data collection, a prediction engine, a user-friendly dashboard, and real-time alerts & notifications. Data security is ensured through encryption, and database queries are optimized for high performance. The system is deployed on Vercel (frontend), Render (backend), and cloud databases for scalability and seamless integration.

MODULES :

- Data Collection Module
- Data Processing & Feature Engineering Module
- Prediction Engine Module
- User Interface & Visualization Module
- Alerts & Notifications Module
- Database Management Module
- API & Integration Module
- Security & Authentication Module

3.4.1 DESCRIPTION OF MODULES

Data Collection Module: Data Collection module is responsible for gathering historical fuel sales data from various sources such as databases, CSV files, and APIs. It also collects external factors like fuel prices, weather conditions, and holidays that may impact sales trends. The system ensures that the data is fetched in real-time or uploaded manually by the user. Proper validation checks are applied to prevent incorrect or missing data entries. The collected data serves as the foundation for accurate sales forecasting.

Data Processing & Feature Engineering Module: Data Processing & Feature Engineering module processes raw input data by cleaning, filtering, and structuring it for analysis. It removes missing values, normalizes data, and handles outliers to ensure accurate predictions. Feature engineering techniques extract useful insights, such as identifying seasonal trends, fluctuations, and peak demand periods. It also converts categorical data into numerical formats to enhance machine learning model performance. This step ensures the prediction engine receives high-quality input.

Prediction Engine Module: The core of the system, Prediction Engine module applies machine learning models such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet to generate fuel sales forecasts. It analyzes past sales patterns, incorporates external factors, and detects demand trends. The module continuously refines the model by adjusting hyperparameters and improving prediction accuracy over time. Users can view forecasts for different time frames, such as daily, weekly, or monthly sales. This helps fuel station managers make data-driven decisions.

User Interface & Visualization Module: User Interface & Visualization Module provides an interactive and user-friendly dashboard designed with React.js and Chart.js for graphical data representation. Users can view historical sales trends, predicted sales, and real-time analytics through charts, tables, and reports. The interface allows users to filter data by date range, fuel type, and external factors. It also includes options to upload data, configure settings, and generate downloadable reports in formats like CSV and PDF.

Alerts & Notifications Module: Alerts & Notifications module ensures that users receive timely alerts about significant changes in fuel sales trends. It generates notifications for low stock levels, unexpected demand surges, or declining sales. Alerts are sent via email, dashboard pop-ups, or SMS (if integrated). This feature helps fuel station managers take proactive measures such as adjusting inventory levels, optimizing fuel purchases, and planning promotional activities.

Database Management Module: The system stores and manages all collected data, processed information, and prediction results in a MySQL database. This module ensures secure storage and efficient retrieval of data for analysis. It maintains a structured database schema, indexing data for faster access. It also supports backup and recovery options to prevent data loss. The module allows integration with cloud databases to ensure scalability and reliability.

API & Integration Module: API & Integration module enables seamless connectivity between the fuel sales prediction system and external applications using RESTful APIs developed with Node.js & Express.js. It allows the system to fetch real-time data from external sources like fuel price APIs, weather APIs, and sales records from existing management software. The module ensures smooth data exchange between the backend and frontend, enhancing system efficiency and interoperability.

Security & Authentication Module : Security is crucial in handling sensitive fuel sales data, and Security & Authentication module ensures data protection through encryption and secure access control mechanisms. User authentication, role-based access control, and data encryption are implemented to prevent unauthorized access. It enforces password hashing, secure login methods, and protection against cyber threats such as SQL injection and cross-site scripting (XSS). This module ensures data confidentiality, integrity, and availability.

4. IMPLEMENTATION AND TESTING

4.1 SYSTEM IMPLEMENTATION

The Fuel Sales Prediction System is implemented through a structured and phased approach to ensure accuracy, reliability, and efficiency in forecasting fuel sales. The implementation process involves integrating various components, including the frontend, backend, machine learning models, and database, to create a seamless and fully functional system.

Frontend Implementation:

The user interface is developed using React.js with Tailwind CSS for styling, ensuring a responsive and intuitive dashboard. Users can upload historical sales data, view interactive charts, and analyze predictions. The UI components interact with the backend through RESTful API calls, providing real-time updates and smooth data visualization.

Backend Implementation:

The backend, built using Node.js and Express.js, handles business logic, data processing, and API management. It connects the frontend with the database and prediction engine, ensuring smooth communication. API endpoints are created for data retrieval, storage, and prediction generation. Security measures, such as authentication and data encryption, are implemented to protect user information.

Machine Learning Model Integration:

The prediction engine is powered by machine learning models such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet. The system processes historical data, extracts meaningful patterns, and generates future sales forecasts. The models are trained using Python libraries (Pandas, NumPy, Scikit-learn, TensorFlow, and Statsmodels) to ensure high accuracy. Predictions are generated dynamically and stored for further analysis.

Database Integration:

A MySQL database is used for storing historical sales data, processed features, prediction results, and user configurations. Proper indexing and query optimization techniques are applied to ensure fast and efficient data retrieval. The database also supports backup and recovery mechanisms to prevent data loss.

Deployment & Hosting:

The system is deployed using Vercel for the frontend and Render for the backend, ensuring scalability and performance. The database is hosted on a cloud-based MySQL service to allow seamless access. APIs are tested for reliability and security before deployment.

Testing & Performance Optimization:

The entire system undergoes rigorous testing, including unit testing, integration testing, and performance testing, to identify and fix bugs. The system is optimized for fast response times, high accuracy, and minimal computational cost. Load testing is conducted to ensure the system can handle large datasets efficiently.

4.2 TESTING

The Fuel Sales Prediction System undergoes rigorous testing to ensure its accuracy, efficiency, and reliability. Various testing methodologies are implemented to validate both functional and non-functional requirements. The primary goal is to detect and resolve errors, optimize performance, and enhance user experience before deployment.

Unit Testing:

- Each module, such as data collection, preprocessing, prediction engine, and user interface, is tested individually.
- Functions like data input validation, API requests, and model predictions are checked for correctness.
- Automated test cases are written using Jest (for React.js frontend) and Mocha/Chai (for Node.js backend).

Integration Testing:

- The interaction between the frontend, backend, database, and machine learning models is tested.
- Ensures that API endpoints correctly retrieve and send data between the user interface and prediction engine.

- Confirms that predictions are generated and stored in the database without errors.

Performance Testing:

- Tests system speed and efficiency in handling large datasets with thousands of fuel sales records.
- Ensures fast response times for API calls, database queries, and ML model execution.

- Load testing is conducted using JMeter to simulate multiple users accessing the system simultaneously.

Functional Testing:

- Verifies whether all system features work as expected, including file uploads, prediction generation, data visualization, and notifications.
- Ensures the system correctly handles user authentication, role-based access, and input validation.

- Each function is tested using real-world data to ensure practical usability.

User Acceptance Testing (UAT):

- Conducted with fuel station managers and business stakeholders to ensure the system meets business requirements.
- Collects user feedback on ease of use, prediction accuracy, and dashboard functionality.

- Final refinements are made based on user suggestions before deployment.

Security Testing:

- Ensures data encryption, user authentication, and access control mechanisms are properly implemented.
- Checks for vulnerabilities like SQL injection, cross-site scripting (XSS), and unauthorized API access.
- Tools like OWASP ZAP and Postman are used to simulate potential security threats.

Test Case

Test ID	**Test Scenario**	**Description**	**Input**	**Expected Output**	**Actual Output**	**Status**
TC_001	Upload historical fuel sales data	Validate uploading a valid CSV/Excel file and processing the data correctly	1. Open the sidebar and use the file uploader. 2. Select a valid file. 3. Observe that data loads and a success message appears.	File is successfully loaded, sample data is replaced (if applicable), and column names are displayed	As expected	Pass

Test ID	Test Scenario	Description	Input	Expected Output	Actual Output	Status
TC_002	Upload invalid file format	Validate error handling when an invalid or corrupted file is uploaded	1. Open the sidebar and use the file uploader. 2. Select an invalid file format. 3. Check for an error message.	Appropriate error message is shown and file processing stops.	As expected	Pass
TC_003	Date Column Conversion	Verify that the selected date column is properly converted to datetime	1. Select the date column from the dropdown. 2. Confirm the date conversion by checking the displayed date range.	Date range is displayed with correct formatting; data is indexed or converted as datetime.	As expected	Pass
TC_004	Line Chart Visualization	Validate that a line chart is generated with the correct x-axis and y-axis data	1. Choose "Line Chart" from the chart type dropdown. 2. Select x-axis and one or more y-axis columns. 3. Generate the	A line chart appears displaying the trends of the selected data over time.	As expected	Pass

Test ID	Test Scenario	Description	Input	Expected Output	Actual Output	Status
			chart.			
TC_ 005	Heatmap Visualization	Validate that a correlation heatmap is generated for the selected numerical columns	1. Select "Heatmap" as the chart type. 2. Choose the numerical columns for analysis. 3. Generate the heatmap visualization.	A heatmap is displayed showing correlation values among the selected columns.	As expected	Pass
TC_ 006	Dashboard Metrics Display	Verify that key dashboard metrics (e.g., Total Diesel, Total Petrol, Diesel:Petrol Ratio, Best Day) are calculated and displayed	1. Load a valid file. 2. Navigate to the Dashboard tab. 3. Check the displayed metric cards for accuracy.	All metric cards show accurate and meaningful computed values based on the data.	As expected	Pass
TC_ 007	Raw Data Pagination	Verify that raw data is paginated correctly based on user input	1. Set the upload dataset per “Rows page" slider to a desired value. 2. Change the	Data table displays only the rows or the current page as per the selected	As expected	Pass

Test ID	**Test Scenario**	**Description**	**Input**	**Expected Output**	**Actual Output**	**Status**
		from the slider	page number using the input. 3. Observe the data table update.	page size.		
TC_ 008	Statistical Summary	Validate that a statistical summary of the dataset is generated	1.Navigate to the Detailed Analysis tab.	A complete statistical summary of the dataset is displayed.	As expected	Pass
TC_ 009	ARIMA Prediction Generation	Validate that predictions are generated using the ARIMA model for the target column	File loaded with time-series data; target column selected ; ARIMA chosen	1. Select ARIMA as the prediction model. 2. Set the number of prediction days. 3. Click "Generate Prediction."	As expected	Pass
TC_ 010	XGBoost Prediction Generation	Validate that predictions are generated using the XGBoost model with feature engineering	File loaded with time-series data; required features available; XGBoost chosen	1. Select XGBoost as the prediction model. 2. Configure parameters (e.g., n_estimators, max_depth). 3. Click "Generate	As expected	Pass

Test ID	Test Scenario	Description	Input	Expected Output	Actual Output	Status
				Prediction.		
TC_011	Advanced Model Parameters	Validate that manually set advanced parameters (for ARIMA/ SARIMA, etc.) affect the prediction output	File loaded; prediction model selected; auto-parameter option unchecked	1. Uncheck auto-select for model parameters. 2. Adjust parameters manually using sliders. 3. Generate predictions.	As expected	Pass
TC_012	Data Preprocessing Feature Engineering	Validate that preprocessing step correctly imputes missing values and creates new time-based features	1. Load a file containing missing values. 2. Run the preprocessing function. 3. Verify that missing values are imputed and new columns (day_of_week, month, quarter, etc.) are added.	Data is cleaned (missing values filled) and enhanced with new features for further analysis.	As expected	Pass

Test ID	Test Scenario	Description	Input	Expected Output	Actual Output	Status
TC_ 013	Browser Compatibility Testing		1. Validate that the dashboard displays correctly across various web browsers and screen resolutions	The dashboard is deployed and accessible via URL; multiple browsers (Chrome, Firefox, Edge, Safari) are available	As expected	Pass

5. CONCLUSION

The Fuel Sales Prediction System is designed to provide an accurate and efficient method for forecasting fuel sales based on historical data and external influencing factors. By leveraging advanced machine learning techniques such as ARIMA, SARIMA, XGBoost, LSTM, and Facebook Prophet**,** the system helps fuel station owners and managers make data-driven decisions, optimize inventory levels, and minimize fuel shortages or overstocking.

The system features an intuitive React.js-based user interface that allows users to visualize sales trends, generate forecasts, and interact with predictive insights. The backend, powered by Node.js and Express.js, ensures seamless communication between the database and prediction models, while MySQL is used for secure and scalable data storage. Additionally, integration with external data sources such as fuel prices, weather conditions, and holiday calendars enhances the accuracy of predictions.

To ensure reliability, the system undergoes rigorous testing procedures, including unit testing, integration testing, performance testing, and security testing. These measures help in identifying and mitigating potential system vulnerabilities, improving response times, and optimizing predictive accuracy. Furthermore, security mechanisms such as user authentication, data encryption, and access control safeguard sensitive business information.

In conclusion, the Fuel Sales Prediction System is a robust, scalable, and business-friendly tool that enhances decision-making for fuel station operators. By providing accurate sales forecasts, automated alerts, and interactive data visualization, the system empowers businesses to improve operational efficiency, reduce losses, and maximize profitability. With its seamless integration, user-friendly design, and AI-driven forecasting, this system is a valuable asset for fuel management and business intelligence.

BIBLIOGRAPHY

Books & Research Papers

Box, G. E. P., & Jenkins, G. M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice. OTexts.

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. The American Statistician, 72(1), 37-45.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Kaggle Datasets: Fuel Sales Trends and Forecasting. Retrieved from https://www.kaggle.com

Official Documentation of TensorFlow, Scikit-learn, and Prophet. Retrieved from https://www.tensorflow.org, https://scikit-learn.org, and https://facebook.github.io/prophet/

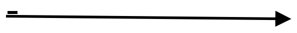
APPENDICES

A. DATA FLOW DIAGRAM

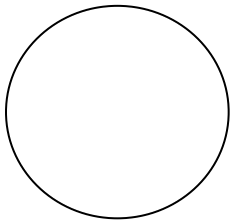
A Data Flow Diagram (DFD) is a diagram that describes the flow of data and the processes that change data throughout a system. It's a structured analysis and design tool that can be used for flowcharting in place of or in association with information. Oriented and process-oriented system flowcharts. Four basic symbols are used to construct data flow diagrams. They are symbols that represent data source, data flows, and data transformations and data storage. The points at which data are transformed are represented by enclosed figures, usually circles, which are called nodes.

Data Flow Diagram Symbols:

- Source or Destination of data



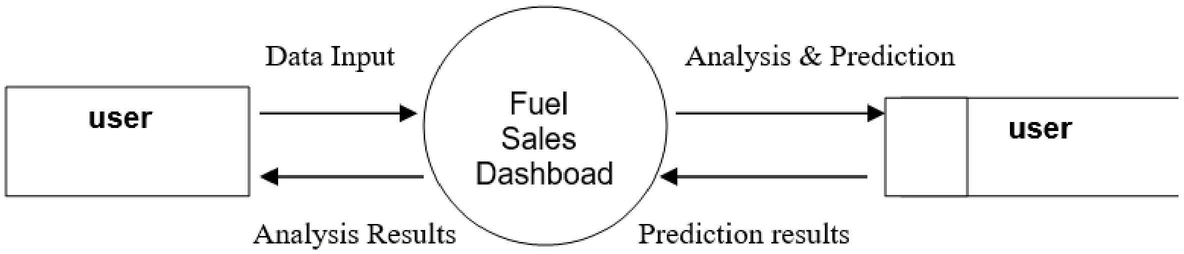
- Data Flow



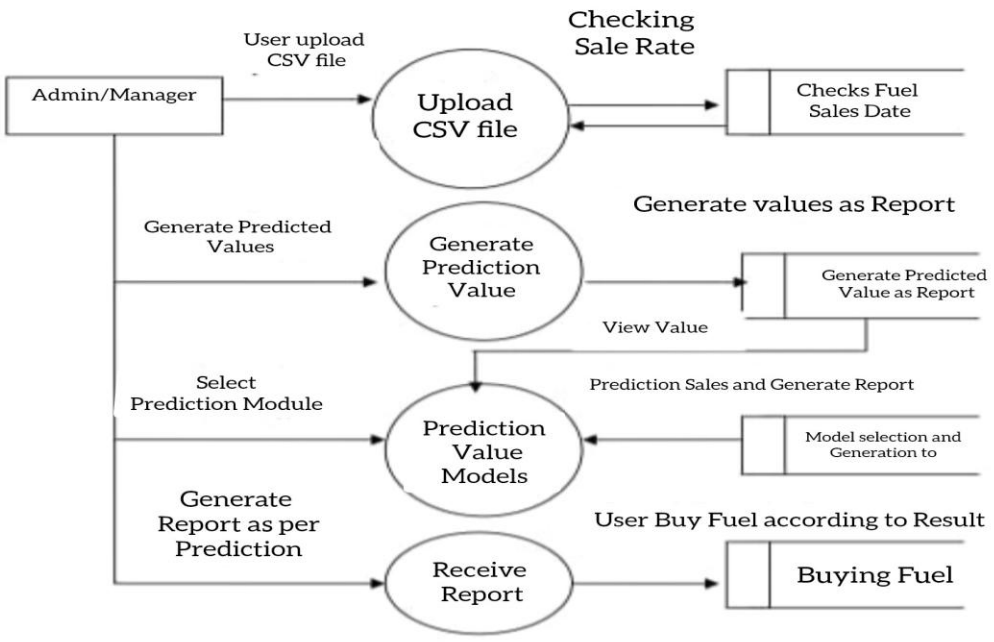
-Process

- Data store

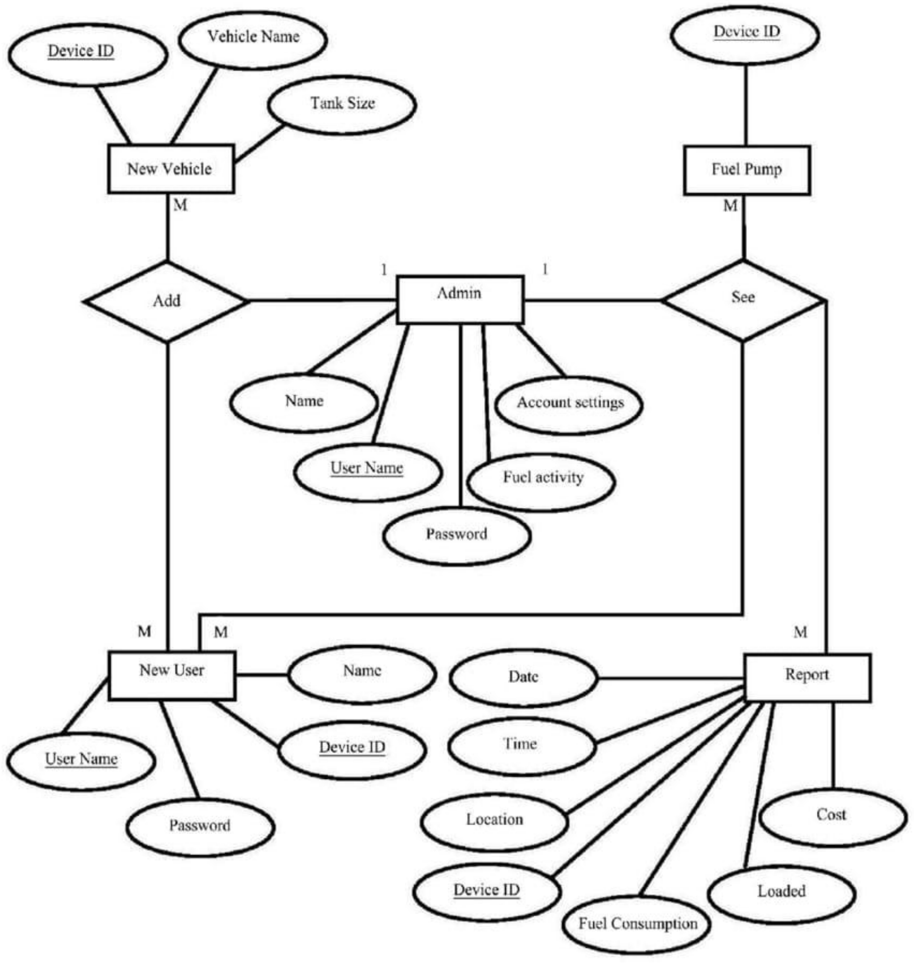
Level-0



Level-1


User upload CSV file
Admin/Manager
Upload CSV file
Checking Sale Rate
Checks Fuel Sales Date
Generate Predicted Values
Generate Prediction Value
Generate values as Report
Generate Predicted Value as Report
View Value
Select Prediction Module
Prediction Value Models
Prediction Sales and Generate Report
Model selection and Generation to
Generate Report as per Prediction
Receive Report
User Buy Fuel according to Result
Buying Fuel

ER-diagram


Device ID
Vehicle Name
Tank Size
New Vehicle
M
Device ID
Fuel Pump
M
1
1
Add
Admin
See
Name
User Name
Password
Fuel activity
Account settings
M
M
M
New User
Name
Device ID
User Name
Password
Date
Report
Time
Location
Device ID
Fuel Consumption
Loaded
Cost

B.TABLE STRUCTURE

Field name	**Data type**	**Size**	**Description**
sale_id	Int	10	Unique identifier for each record
Date	Date	0	Date of the fuel sale (indexed in the dashboard)
Day	Varchar	15	Day of the week (e.g., Monday, Tuesday, etc.)
Total_Die	Decimal	(10,2)	Total diesel sales in liters
Total_Petrol	Decimal	(10,2)	Total petrol sales in liters
New_Liters	Decimal	(10,2)	Additional/new fuel product sales in liters (optional field)

C. SAMPLE CODING

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
import plotly.express as px
import plotly.graph_objects as go
from datetime import datetime, timedelta
from prophet import Prophet
import warnings
warnings.filterwarnings('ignore')

# Set page configuration
st.set_page_config(page_title="Fuel Sales Analyzer", layout="wide",
initial_sidebar_state="expanded")

# Custom CSS for better styling with dark theme
st.markdown("""
<style>
.main-header {
font-size: 2.5rem;
color: #1E88E5;
text-align: center;
margin-bottom: 1rem;
}
.sub-header {
font-size: 1.5rem;
color: #E0E0E0;
margin-bottom: 1rem;
}
.card {
background-color: #1E1E1E;
border-radius: 5px;
padding: 1rem;
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
}
.metric-card {
background-color: #1E88E5;
color: white;
border-radius: 5px;
padding: 1rem;
text-align: center;
}
.model-info {
background-color: #1E1E1E;
border-left: 4px solid #1E88E5;
padding: 1rem;
margin-bottom: 1rem;
color: #E0E0E0;
}
.stButton>button {
background-color: #1E88E5;
color: white;
font-weight: bold;
padding: 0.5rem 1rem;
width: 100%;
}
</style>
""", unsafe_allow_html=True)

# Title and intro
st.markdown("<h1 class='main-header'>Fuel Sales Analysis & Prediction Dashboard</h1>", unsafe_allow_html=True)
st.markdown("Upload your sales data to analyze trends and forecast future sales using five advanced algorithms.")

# Sidebar for data upload and configuration
with st.sidebar:
st.header("Data & Configuration")

uploaded_file = st.file_uploader("Upload your Excel file:", type=['xlsx', 'xls', 'csv'])

if uploaded_file is None:
st.info("No file uploaded. Using sample data for demonstration.")

# Create sample data similar to the image
dates = pd.date_range(start='2024-03-01', periods=30)
# Generate more realistic sample data with weekly patterns
weekday_factor = np.array([0.8, 0.9, 1.0, 0.95, 1.1, 1.2, 0.85])  # Mon-Sun factors
base_diesel = 700 + np.random.normal(0, 50, 30)
base_petrol = 500 + np.random.normal(0, 70, 30)
base_new = 200 + np.random.normal(0, 30, 30)

sample_data = pd.DataFrame({
'Date': dates,
'Day': [d.strftime('%A') for d in dates],
'Total Die': base_diesel * [weekday_factor[d.weekday()] for d in dates],
'Total Petrol': base_petrol * [weekday_factor[d.weekday()] for d in dates],
'NEW(Liters)': base_new * [weekday_factor[d.weekday()] for d in dates]
})

df = sample_data
else:
try:
# Try to determine file type and read accordingly
if uploaded_file.name.endswith('.csv'):
df = pd.read_csv(uploaded_file)
else:
df = pd.read_excel(uploaded_file)
```

```
st.success(f"File loaded successfully! {len(df)} records found.")
except Exception as e:
st.error(f"Error loading file: {e}")
st.stop()

st.subheader("Available Columns")
all_columns = df.columns.tolist()
st.write(all_columns)

# Ensure date column is properly formatted
date_col = st.selectbox("Select Date Column:", all_columns,
index=all_columns.index('Date') if 'Date' in all_columns else 0)

# Try to convert the date column to datetime
if date_col in df.columns:
try:
df[date_col] = pd.to_datetime(df[date_col])
min_date = df[date_col].min()
max_date = df[date_col].max()
st.write(f"Date range: {min_date.date()} to {max_date.date()}")
except:
st.warning("Could not convert the selected column to date. Please ensure it
contains date values.")

# Analytics configuration
st.subheader("Analysis Configuration")
chart_type = st.selectbox("Chart Type:", ["Line Chart", "Bar Chart",
"Area Chart", "Scatter Plot", "Heatmap"])

numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()

# Added option for X-axis selection
if chart_type != "Heatmap":
x_column = st.selectbox("Select X-axis column:", all_columns,
index=all_columns.index(date_col) if date_col in all_columns else 0)
y_columns = st.multiselect("Select Y-axis column(s):", numerical_cols,
default=numerical_cols[:2] if len(numerical_cols) >= 2 else
```

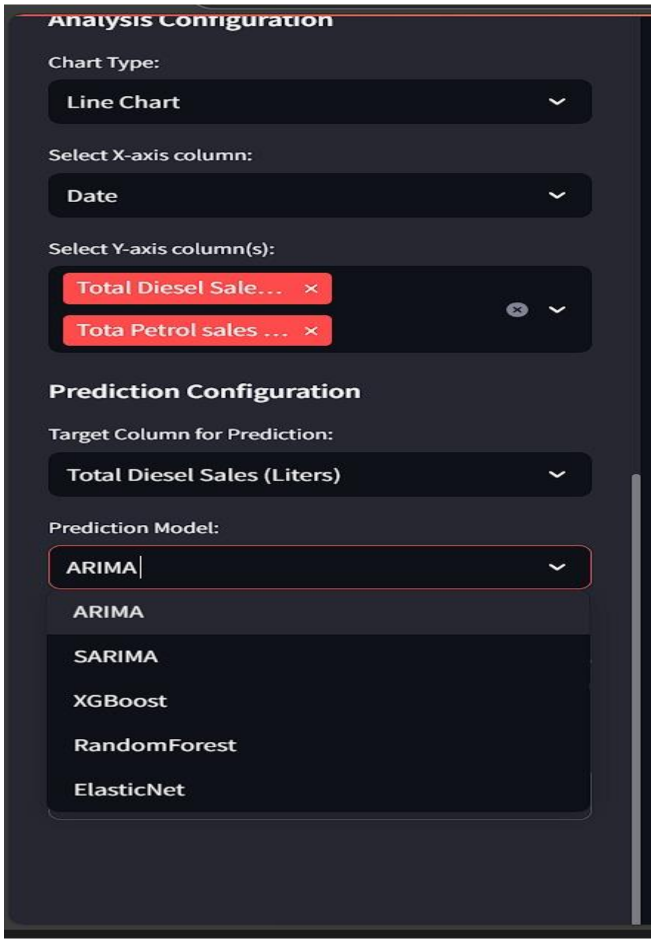
```
numerical_cols[:1])
else:
y_columns = st.multiselect("Select columns for correlation:", numerical_cols,
default=numerical_cols)
# Prediction configuration
st.subheader("Prediction Configuration")
target_column = st.selectbox("Target Column for Prediction:",
numerical_cols)
prediction_model = st.selectbox("Prediction Model:",
["ARIMA", "SARIMA", "XGBoost", "RandomForest", "ElasticNet"])
prediction_days = st.slider("Number of days to predict:", 1, 30, 7)
# Advanced model parameters
st.subheader("Advanced Model Parameters")
with st.expander("Model Parameters"):
if prediction_model == "ARIMA" or prediction_model == "SARIMA":
auto_param = st.checkbox("Auto-select parameters", value=True)
if not auto_param:
p_value = st.slider("p (AR order):", 0, 5, 1)
d_value = st.slider("d (Differencing):", 0, 2, 1)
q_value = st.slider("q (MA order):", 0, 5, 1)
if prediction_model == "SARIMA":
m_value = st.slider("m (Seasonal periods):", 2, 12, 7)

elif prediction_model == "XGBoost":
n_estimators = st.slider("Number of estimators:", 50, 500, 100)
max_depth = st.slider("Max depth:", 3, 10, 6)
learning_rate = st.slider("Learning rate:", 0.01, 0.3, 0.1)

elif prediction_model == "RandomForest":
rf_n_estimators = st.slider("Number of trees:", 50, 500, 100)
rf_max_depth = st.slider("Max depth:", 3, 20, 10)
rf_min_samples_split = st.slider("Min samples split:", 2, 10, 2)
elif prediction_model == "ElasticNet":
alpha = st.slider("Alpha:", 0.01, 1.0, 0
```

D.SAMPLE INPUT

Analysis Configuration



Analysis Configuration

Chart Type:

Line Chart

Select X-axis column:

Date

Select Y-axis column(s):

Total Diesel Sale...

Tota Petrol sales ...

Prediction Configuration

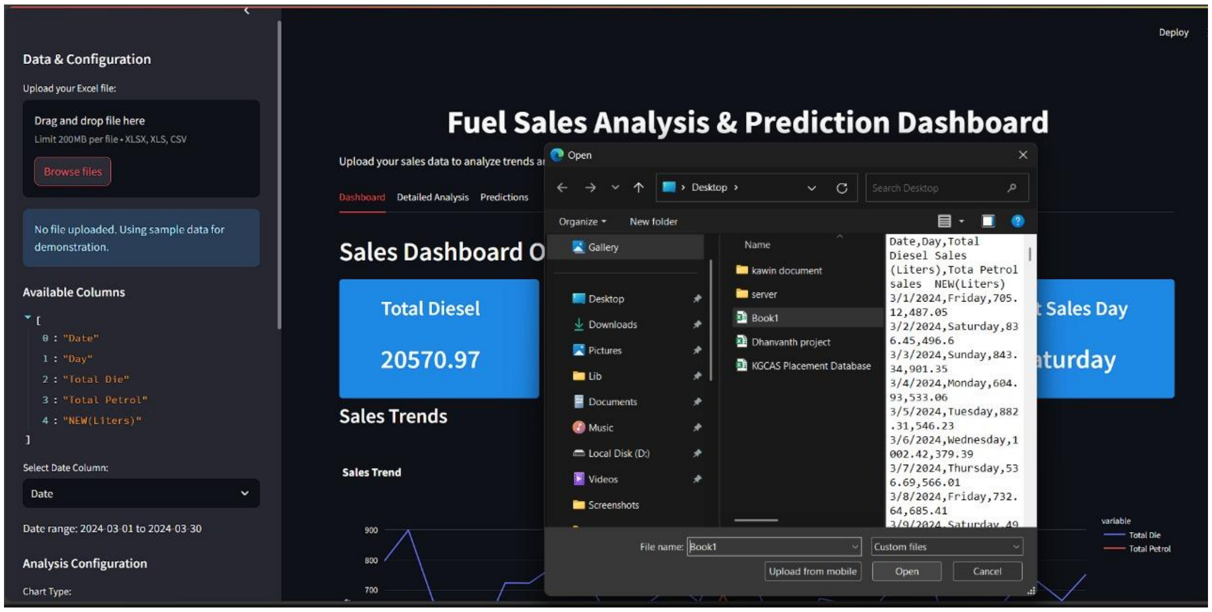
Target Column for Prediction:

Total Diesel Sales (Liters)

Prediction Model:

ARIMA

CSV File Upload:



Upload your Excel file:

Drag and drop file here
Limit 200MB per file • XLSX, XLS, CSV

No file uploaded. Using sample data for demonstration.

Available Columns

```
[
0 : "Date"
1 : "Day"
2 : "Total Die"
3 : "Total Petrol"
4 : "NEW(Liters)"
]
```

Select Date Column: Date

Date range: 2024-03-01 to 2024-03-30

Analysis Configuration

Chart Type:

Fuel Sales Analysis & Prediction Dashboard

Upload your sales data to analyze trends a

Sales Dashboard O

Total Diesel 20570.97

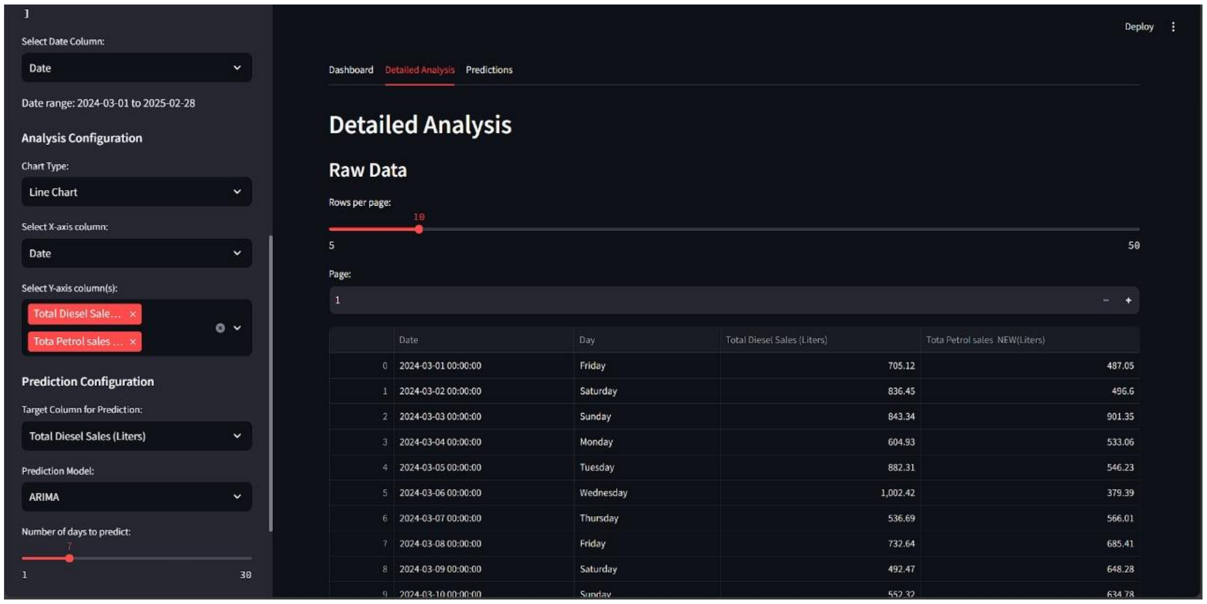
t Sales Day aturday

Sales Trends

File name: Book1

```
Date,Day,Total
Diesel Sales
(Liters),Tota Petrol
sales  NEW(Liters)
3/1/2024,Friday,705.
12,487.05
3/2/2024,Saturday,83
6.45,496.6
3/3/2024,Sunday,843.
34,901.35
3/4/2024,Monday,604.
93,533.06
3/5/2024,Tuesday,882
.31,546.23
3/6/2024,Wednesday,1
002.42,379.39
3/7/2024,Thursday,53
6.69,566.01
3/8/2024,Friday,732.
64,685.41
```

Inserted Sales Data:



Select Date Column: Date

Date range: 2024-03-01 to 2025-02-28

Analysis Configuration

Chart Type: Line Chart

Select X-axis column: Date

Select Y-axis column(s): Total Diesel Sale... Tota Petrol sales ...

Prediction Configuration

Target Column for Prediction: Total Diesel Sales (Liters)

Prediction Model: ARIMA

Number of days to predict: 7 (1–30)

Detailed Analysis

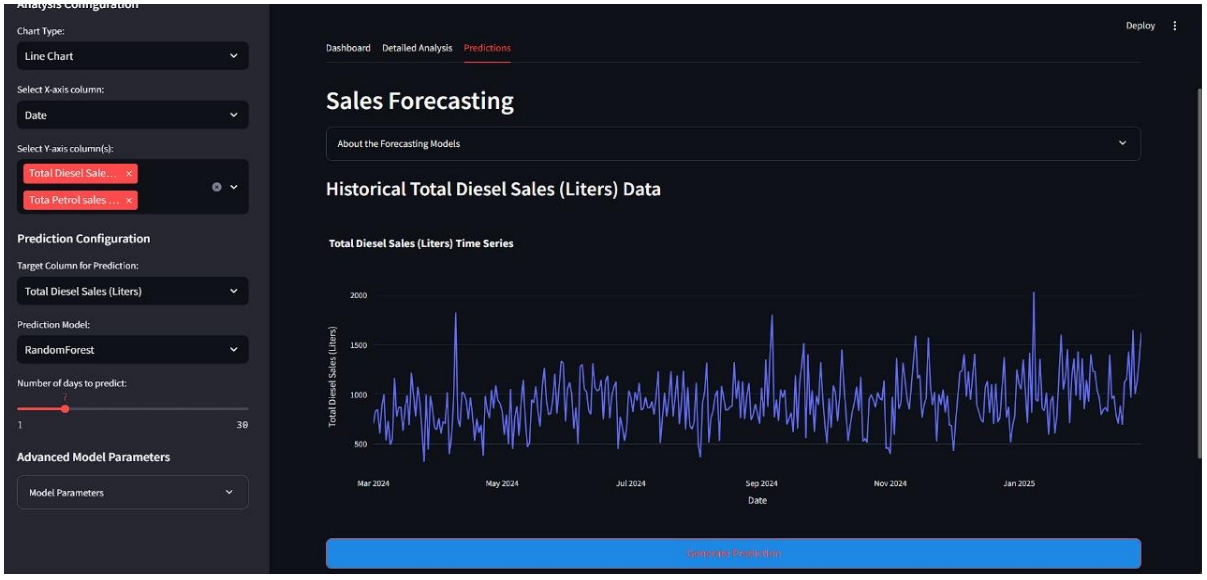
Raw Data

Rows per page: 10 (5–50)

Page: 1

	Date	Day	Total Diesel Sales (Liters)	Tota Petrol sales NEW(Liters)
0	2024-03-01 00:00:00	Friday	705.12	487.05
1	2024-03-02 00:00:00	Saturday	836.45	496.6
2	2024-03-03 00:00:00	Sunday	843.34	901.35
3	2024-03-04 00:00:00	Monday	604.93	533.06
4	2024-03-05 00:00:00	Tuesday	882.31	546.23
5	2024-03-06 00:00:00	Wednesday	1,002.42	379.39
6	2024-03-07 00:00:00	Thursday	536.69	566.01
7	2024-03-08 00:00:00	Friday	732.64	685.41
8	2024-03-09 00:00:00	Saturday	492.47	648.28
9	2024-03-10 00:00:00	Sunday	552.32	634.78

Predicted Sales:



Analysis Configuration

Chart Type: Line Chart

Select X-axis column: Date

Select Y-axis column(s): Total Diesel Sale... Tota Petrol sales ...

Prediction Configuration

Target Column for Prediction: Total Diesel Sales (Liters)

Prediction Model: RandomForest

Number of days to predict: 7

Advanced Model Parameters

Model Parameters

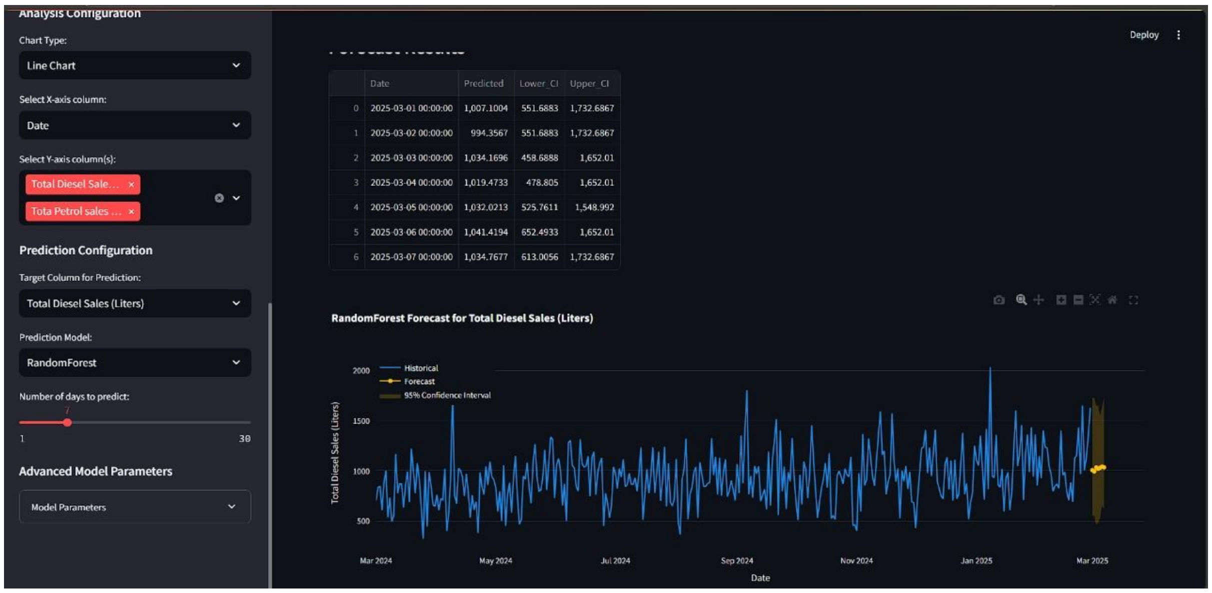
Sales Forecasting

About the Forecasting Models

Historical Total Diesel Sales (Liters) Data

E. SAMPLE OUTPUT

Random Forest Model :



Analysis Configuration

Chart Type: Line Chart

Select X-axis column: Date

Select Y-axis column(s): Total Diesel Sale… Tota Petrol sales …

Prediction Configuration

Target Column for Prediction: Total Diesel Sales (Liters)

Prediction Model: RandomForest

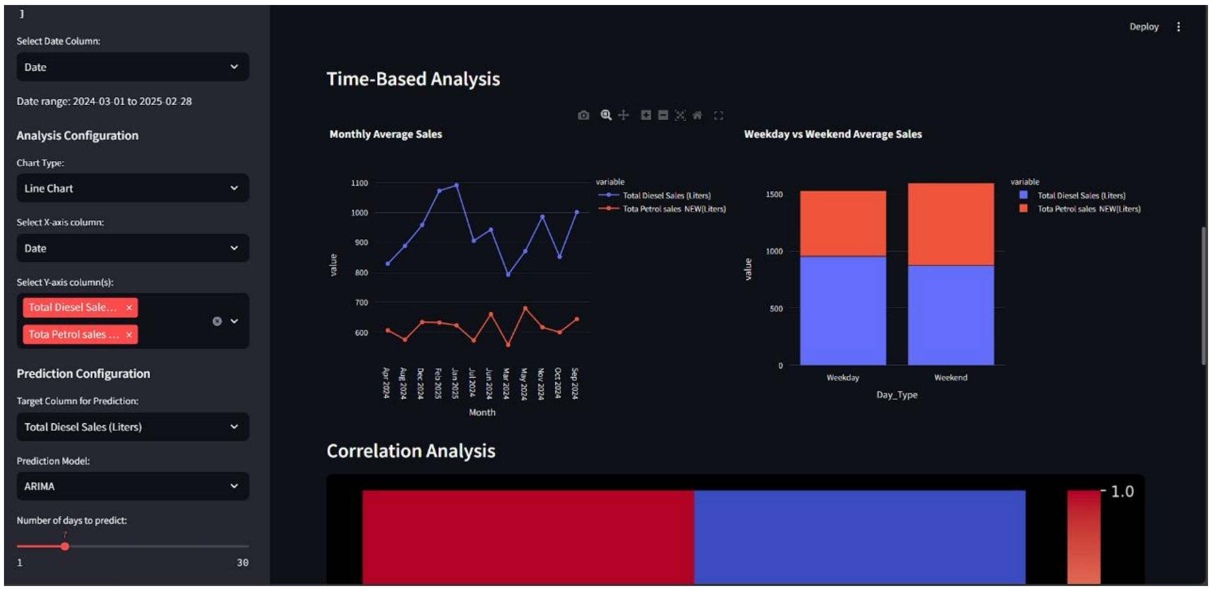
Number of days to predict: 7

Advanced Model Parameters

Model Parameters

	Date	Predicted	Lower_CI	Upper_CI
0	2025-03-01 00:00:00	1,007.1004	551.6883	1,732.6867
1	2025-03-02 00:00:00	994.3567	551.6883	1,732.6867
2	2025-03-03 00:00:00	1,034.1696	458.6888	1,652.01
3	2025-03-04 00:00:00	1,019.4733	478.805	1,652.01
4	2025-03-05 00:00:00	1,032.0213	525.7611	1,548.992
5	2025-03-06 00:00:00	1,041.4194	652.4933	1,652.01
6	2025-03-07 00:00:00	1,034.7677	613.0056	1,732.6867

ARIMA Model:



Select Date Column: Date

Date range: 2024-03-01 to 2025-02-28

Analysis Configuration

Chart Type: Line Chart

Select X-axis column: Date

Select Y-axis column(s): Total Diesel Sale… Tota Petrol sales …

Prediction Configuration

Target Column for Prediction: Total Diesel Sales (Liters)

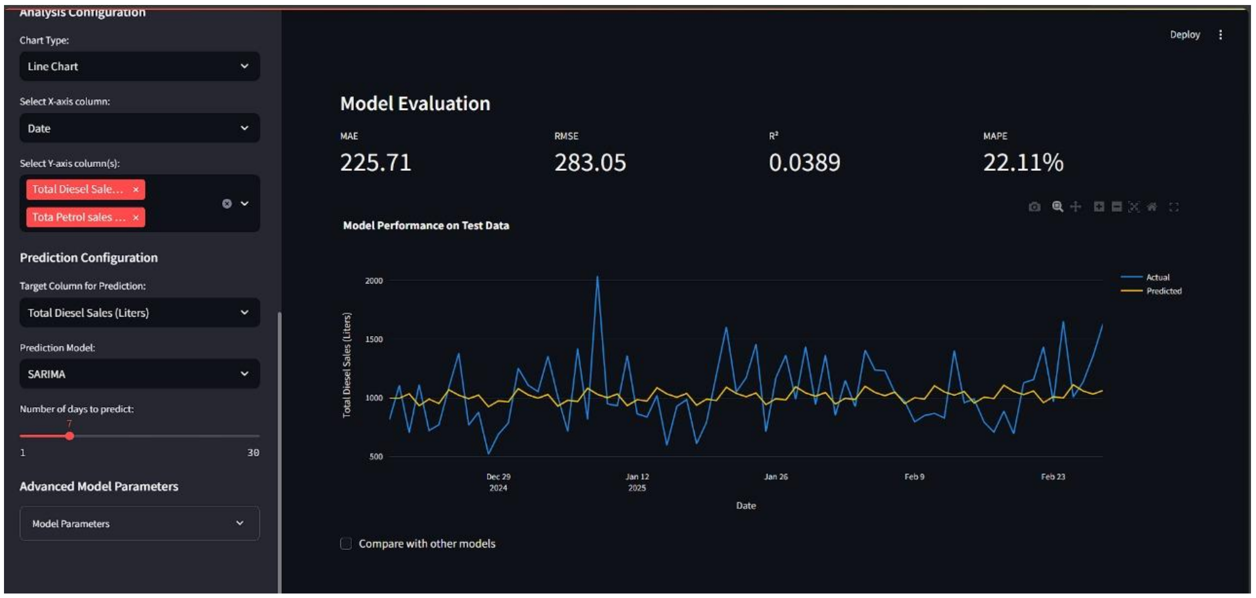
Prediction Model: ARIMA

Number of days to predict: 7

Time-Based Analysis

Correlation Analysis

SARIMA Model :



Analysis Configuration

Chart Type: Line Chart

Select X-axis column: Date

Select Y-axis column(s): Total Diesel Sale... Tota Petrol sales ...

Prediction Configuration

Target Column for Prediction: Total Diesel Sales (Liters)

Prediction Model: SARIMA

Number of days to predict: 7

Advanced Model Parameters

Model Parameters

Model Evaluation

MAE	RMSE	R²	MAPE
225.71	283.05	0.0389	22.11%

Deploy
Model Performance on Test Data
Actual
Predicted
2000
1500
1000
500
Total Diesel Sales (Liters)
Dec 29 2024
Jan 12 2025
Jan 26
Feb 9
Feb 23
Date
1
30
Compare with other models