AI-Based Parts Optimization Platform (POP)

Student Name: Sriyanshu Pokhrel

Student ID: 14556409

Institution: University of Technology Sydney

Course: 32513 31005 Advanced Data Analytics, Algorithms, Machine Learning

Module: Applied Machine Learning / Final Project

Submission Date: 5th October 2025

**Abstract**

Through this project, I was able to build a working web-based system called the Parts Optimization Platform (POP). It uses simple AI methods to handle data, make predictions and let users interact with it in real time. The system is designed for businesses in the auto parts industry, and the final result reflects that goal.

The developed system was based on the Flask web framework connecting machine learning models of demand forecasting, inventory, and sales management. The POP system will forecast demands intelligently using the forecasted time-series analysis (EMA/SMA), manage inventories with forecasted requirements, and record the purchase trends of the user for decision-making.

This platform will ensure secure access to users and administrators, OTP verification, and the ability to synchronize different data files upon use.

It's an AI integration method within the supply chain and retail operation that will create a solid basis for a smart inventory system. It has an objective to minimize overstocking and shortages while maximizing efficiency.

**Introduction**

Inadequate stock management of the present retail and manufacturing environment has been characterized by inefficiency, delays, and data inconsistencies. The project, AI-Based Parts Optimization Platform, hence, is an endeavor to resolve all the above-mentioned challenges by means of artificial intelligence and real-time data processing.

The system is a combination of data science principles, Flask-based web technologies, and machine-learning forecasting for creating a homogenized working interface for product management and customer interaction.

**Objectives**

-To develop an intelligent automated system that connects the products sold to future stock requirements.

-Implement real-time data replication with multiple modules, i.e., sales, stock, and information of the users.

-Secure login for users with access control.

-Integrating forecasting metrics using which one can evaluate model performance.

-Visualize all the data in a more friendly and interactive way by using animated dashboards.

This project exemplifies not only one's technical knowledge related to web frameworks and ML modeling but, more significantly, it demonstrates practical application and responsiveness in real time that is, in an AI-driven business environment

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**System design and Architecture**

The system has been built on a multi-tiered architecture. This ensures the system possesses modularity and dynamic synchronization.

1. Flask Application Layer:

This takes care of all activities that have to do with the application in question using the Flask framework. Jinja2 is used for rendering templates that have real-time updates coming directly from CSV files.Changes in data by the admin or user will instantly notify the rest of the system.

b. User Layer

Registration and Login: The user's username and password shall be 7 and 10 characters long, respectively; for uniformity of data in this project.

OTP Checkouts: A 5-digit OTP will be generated during checkout. The OTP stays valid for 30 seconds, and the cooldown period is 1 minute after 3 failed attempts.

Cart System and Billing: After OTP verification passes, a detailed invoice is generated and saved under a file called sales.csv.

Security Controls: Frontend controls are maintained for invalid OTPs or username length.

c. Admin Layer

Admin logs in using pre-set credentials (admin123 / sriyanshu1).

Features:

Product Management: Insert, modify, or delete product details in products.csv.

Demand Forecasting Dashboard: Shows SMA and EMA-based demand forecasts for various products.

User Management: Admin can see, reset, or modify user passwords (guarded by PIN 5624).

Sales Monitoring: Full records of all user transactions with time, quantity, and amount.

The admin is also given reorder recommendations and sales summaries dynamically generated from machine learning outputs.

d. Data Layer

All the important data is stored in CSV format for transparency and ease:

users.csv: Saves usernames, passwords, roles, and emails.

products.csv: Keeps stock, category, and price.

sales.csv: Saves all the purchases with order ID, user, timestamp, and total price.

reorder\_suggestions.csv and bundle\_suggestions.csv: Generated by the ML modules.

All files are constantly synchronized, so any update or addition is instantly available throughout the system.

e. Frontend Layer

Frontend utilizes HTML5, CSS, and Bootstrap, with animations for graph and visual transitions rendering.

Graphs in the insights and admin dashboard are built using Chart.js, which makes them interactive, responsive, and visually engaging.

**Machine Learning Part**

This project's predictive intelligence consists of its engine, an artificial intelligence that forecast demand and assists in inventory optimization in two principal models:

a. Simple Moving Average (SMA)

SMA removes short-time variations and outputs long-time trends from historical sales data.

It is appropriate for average product demand and provides a simple model for performance comparison.

b. Exponential Moving Average (EMA)

EMA places more stress on past sales, and therefore the system is more effective in anticipating abrupt shifts in demand quickly.

This model was chosen because it is highly reactive to existing data which is ideal for fast-moving inventory like automotive parts.

c. Implementations of Forecasting Models

Both SMA and EMA were implemented using Python, mainly with the pandas library to process historical sales data. For SMA, the system calculates a rolling average over a selected number of past data points for each product. EMA was implemented with a smoothing factor so that recent sales have more influence on the forecast than older data. This approach was embedded directly into the code so that predictions update automatically whenever new sales records are added.

During deployment, the system reads the updated CSV files and refreshes the forecasts. Historical data is used to generate the baseline, while new entries trigger real-time prediction outputs.

d. Forecasting Process

The daily sales is aggregated from sales.csv.

The program calculates SMA and EMA for all products from past sales.

Forecasted quantities of demand for the future period are computed.

These performance indicators such as MAE, RMSE, and MAPE are produced.

The program saves its outputs to reorder\_suggestions.csv and displays trends for admin verification.

e. Model Inputs and Outputs

The forecasting features in this system rely on time-series data stored in CSV files such as sales.csv, products.csv, and related stock records. These contain product IDs, past sales, stock quantities, and timestamps. This data acts as the input for the SMA and EMA models. The output is a predicted demand value for each product, which is then used for automated restocking suggestions, inventory planning, and dashboard display.

f. Evaluation Metrics

- Different Metric Description MAE (Mean Absolute Error) estimates average magnitude of the errors in predictions.

- RMSE (Root Mean Square Error)Strongly punishes larger errors more than MAE.

- MAPE (Mean Absolute Percentage Error) Reports accuracy in actual values in terms of percentages. These measurements appear in the admin panel, and they're used to measure model reliability and guide improvements.

**Discussion and Observations**

The project successfully demonstrates how a traditional inventory management system can be enhanced with machine learning and web automation. But certain things became clear:

-Flatline predictions take place when too little data points exist; thus, the model is more dependent on EMA for adaptive forecast.

-CSV-oriented synchronization is lean, but performance-critical at scale — and thus may migrate to a database in future releases.

-The use of OTP and PIN ensures efficient real-time security without losing usability.

-Overall, the model achieves real-time adaptability, transparent reporting, and intelligent forecasting within a single cohesive platform.

**Challenges and Solutions**

Challenge Implemented Solution

* Delayed stock synchronization Implemented immediate CSV rewrite functions.
* Flatlined forecasts Improved with EMA weighting and rolling window adjusted.
* Admin editing issues Security concerns with admin editing Entered PIN (5624) for password change.
* Limited dataset size Enabled continuous update learning from new sales.
* UI Responsiveness Used Chart.js animations and Bootstrap grid system.

**Future Enhancements**

To improve the system’s scalability and predictive power:

- Replace CSVs with PostgreSQL or Firebase database for real-time concurrency support.

- Use LSTM (Long Short-Term Memory) neural networks for more reliable demand forecastings.

- Deploy on Docker for cloud scalability or AWS Elastic Beanstalk.

- Add the notification system that will notify admin of products in low-stocks.

- Include email-based OTPs for additional user security.

**Conclusion**

The Parts Optimization Platform for AI Successfully Merges AI-Fueled Forecasting and Real-Time Web Technologies.

By combining Flask's versatility with judicious data modeling, the system streamlines the entire product management lifecycle — from recording sales to forecasting stocks. From this project, the application illustrates that intelligent automation, precision, and dynamic decision-making is achievable in practical, real-world applications even through lightweight architecture.

It forms the foundation for eventual growth to an entirely automated, cloud-deployed retail optimization platform.

**Use of AI tools.**

I used AI as support throughout the project, mainly when I needed help understanding certain concepts like checking parts of my code or finding better ways to structure and explain my ideas. It also helped me improve my writing by rewording or cleaning up some sections of the journal. However, the core work like the logic behind the forecasting, the coding, the system design and the decisions taken, was done by me. Whenever I used AI, I made sure to edit the suggestions so they matched my own thinking and the actual goals of the project.

**Project Repository Link**

The full code and files for the Parts Optimization Platform (POP), including the forecasting logic, are available at the link below:

<https://github.com/pokhrelsriyanshu791-lang/14556409-Sriyanshu-Pokhrel.git>

**References**

Chart.js Contributors. (n.d.). Chart.js documentation. https://www.chartjs.org/docs/

Chatfield, C. (2000). Time-series forecasting. Chapman & Hall/CRC.

Grus, J. (2019). Data science from scratch: First principles with Python (2nd ed.). O’Reilly Media.

Pallets Projects. (n.d.). Flask documentation. https://flask.palletsprojects.com/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.