ML-CYO-PROJECT

Demica Smit 577875
Bianca Long 600476
Trent Rhett Evans 600383
Jade Adam Riley 578125

Platform: https://mlg382-project2-souk.onrender.com/analysis

Repo: [https://github.com/DemicaSmit/Project2]

Problem Statement

Inventory management is a critical component for any retail or supply chain-based operation. Stockouts lead to missed revenue opportunities and reduced customer satisfaction, while overstocking increases carrying costs and ties up capital. Businesses often lack predictive tools to estimate future demand and reorder points accurately. This project aims to develop a data-driven inventory management system capable of optimizing stock levels, minimizing stockouts, and reducing carrying costs. The system utilizes machine learning techniques to predict purchasing quantities based on historical transactional data, enabling smarter restocking decisions and streamlined inventory flow.

Inventory Management

Develop an inventory management system to optimize stock levels, minimize stockouts, and reduce carrying costs by predicting demand and reorder quantities.

Understanding the Data

The dataset contains over 500,000 records of online retail transactions. Each transaction includes:

- **InvoiceDate:** timestamp of the transaction
- **StockCode:** product identifier
- **Description:** product name
- **Quantity:** number of units purchased
- **UnitPrice:** price per unit
- **CustomerID** and **Country**

The target variable is **Quantity**, and features selected include time-based attributes (Hour, Day of Week), product price, customer region, and product code.

Data Exploration and Pre-processing

The data was loaded and inspected to understand the structure, data types and presence of missing or duplicate values. The rows containing missing values were removed as well as invalid entries and returns. We removed missing values by looking at the rows with missing customer ID or description. Invalid entries and returns were removed by looking at quantity and unit price to see if the values are smaller or equal to 0. We then converted the Invoice Data column to a date-to-time format so that we can extract it into month, day and hour columns. We also encoded Country and Stock code into numerical values so that it can be used in the machine learning models. To ensure consistency we scaled the unit price and quantity so that they are using the same scale. Lastly we analysed the variables to see what could influence purchasing quantity then separated them into input and target variables .

Feature Engineering

Final feature set:

- **CountryCode** (encoded from Country)
- **ProductCode** (encoded from StockCode)
- **Hour**, **DayOfWeek**, **Month** (from InvoiceDate)

Scaling: StandardScaler applied to UnitPrice to ensure numeric stability across features.

Target variable: Quantity

Develop Machine Learning Models

In this part of the inventory management system, machine learning models were developed to predict purchase quantities, enabling optimized stock levels and minimized stockouts. Tree regression models were developed and trained on the processed dataset to model relationships between input features (e.g. ProductCode, UnitPrice, Hour, DayOfWeek, CountryCode) and the target variable- Linear Regression, Random Forest Regressor, and Support Vector Regressor. Hyperparameter optimization was performed for Random Forest and Support Vector Regressor using RandomizedSearchCV, with performance evaluated through R^2 score, Mean Squared Error, and Mean Absolute Error on standardized and original scales. Robust assessment was ensured via 10-Fold cross-validation for Linear Regression and

Random Forest. The Random Forest Regressor displayed excellent performance, achieving a test $R^2$of 0.2459 and a cross-validation mean R^2 of 0.1508, compared to 0.0051 and 0.0078 for Linear Regression, and approximately 0.003 for Support Vector Regressor. Consequently, the Random Forest Regressor was selected as the final mode, with comprehensive visualizations, performance metrics, and training statistics.

Dash App

The **Overview** page provides a high-level summary of the model's performance and its practical implications. This includes:

- **Predicted vs. Actual Quantity Visualization**: A comparative graph showing how closely the model's predictions align with actual values, helping users understand the real-world applicability of the model.
- **Model Distribution by MAE and RSSE**: Performance comparison of each trained model using Mean Absolute Error (MAE) and Root Sum of Squared Errors (RSSE) metrics. This view offers insights into the error distribution across models, enabling users to assess which models are more reliable under various conditions.

The **Predict** page allows users to interact with the trained model by entering custom input values. Features include:

- **User Input Form**: Users can input specific features .
- **Instant Prediction**: Once input is submitted, the app returns the predicted academic outcome offering immediate feedback on potential quantity orders
- **Practical Use Case**: This functionality is especially useful for logistic operations or stakeholders aiming to predict future outcomes and tailor interventions accordingly.

The **Dataset** page provides transparency into the data that powers the predictive models. Key features include:

- **Full Dataset Table View**: Users can browse the raw data that was used for training, validation, and testing of the model.
- **Column Descriptions**: Each feature is documented to help users understand the role of individual variables in prediction

The **Analysis** page provides a deep dive into the performance metrics and evaluation of each model. This includes:

- **Model Evaluation Metrics**: Display of Accuracy, MAE, RSSE, F1 Score, and other relevant metrics for each algorithm
- **Visual Comparison**: Graphs comparing **Precision** and **Recall** scores for each model to visualize trade-offs in classification performance.
- **Confusion Matrix (Random Forest)**: A matrix showing true vs. predicted class labels for the Random Forest model. This allows users to identify where the model performs well and where misclassifications occur, critical for understanding model reliability and bias.

Continuous Improvement

As we progressed through the project, we gave feedback to each other on how we can improve our parts. This means while we were developing our project there were continuous improvements. Some improvements we made was adding more exploratory data analysis graphs to understand the data better. We also improved the dash app design and added extra models to help with the predictions.

Reflection and Future Work

In the future to improve the dash app we will be looking at integrating time-series forecasting models for seasonal trend prediction. We will also be looking at adding anomaly detection for return fraud or unusual buying behaviours. By adding these improvements in our future work it will make the predictions even more accurate and make the application more functional to use.