Yogeshwaran L, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Time-Series Forecasting of Seasonal Items Sales Using Machine Learning.

Yogeshwaran L, Dinesh Karthick K, Siddarth KM, Vivekanantha Rajeshwaran S
Dept. of Artificial Intelligence and Data Science RVS Technical Campus, Coimbatore.

Abstract- Sales forecasting is an essential task in modern business environments, as it helps organizations make informed decisions related to inventory management, production planning, and resource allocation. Accurate prediction of sales becomes more challenging in the case of seasonal products, where demand varies significantly over time due to factors such as weather conditions, festivals, and consumer behaviour patterns. This paper presents a machine learning-based system for forecasting the sales of seasonal items using historical time-series data. The dataset is pre-processed by handling missing values, converting date formats, and organizing data in a structured manner. Feature engineering techniques extract meaningful temporal attributes such as month, day, and year, which play a crucial role in identifying seasonal patterns. A Linear Regression model analyses the relationship between the extracted features and sales values, achieving an R^2 score of 0.89 with low error metrics (MAE: 10.5, RMSE: 13.4). The system is deployed as a Flask-based web application enabling real-time sales predictions through a user-friendly interface. Results demonstrate that the proposed approach effectively captures seasonal trends and provides reliable predictions suitable for retail, e-commerce, and inventory management applications.

Keywords: Sales forecasting; time-series analysis; machine learning; linear regression; seasonal demand; Flask deployment; feature engineering; inventory management; real-time prediction.

I. INTRODUCTION

In the modern business environment, accurate prediction of sales is essential for effective planning and decision-making. Organizations depend on historical data to understand customer behaviour and market trends. Sales forecasting helps businesses in managing inventory, reducing losses, and improving profitability.

Seasonal products show variations in demand based on time-related factors such as month, weather, and festivals. For example, sales of winter clothing increase during colder months, while products like cold drinks and ice creams are in high demand during summer. These variations make it difficult to predict sales using traditional methods.

Machine learning provides an efficient solution by analysing historical data and identifying hidden patterns. By using machine learning techniques, it is possible to develop models that can accurately predict future sales. This project focuses on forecasting seasonal item sales using machine learning and deploying it through a web- based application.

The principal contributions of this work are as follows:

1. A complete end-to-end machine learning pipeline for seasonal sales forecasting using time-based feature engineering.
2. A feature extraction framework that derives month, day, year, and weekday attributes from raw date data to capture seasonal patterns.
3. A deployed Flask web application enabling real-time, user-interactive sales prediction without any specialist hardware.
4. Empirical evaluation demonstrating an R^2 score of 0.89 with MAE of 10.5 and RMSE of 13.4 on held-out test data.

The remainder of this paper is structured as follows. Section II surveys related work. Section III describes the proposed methodology. Section IV presents experimental results. Section V discusses findings and limitations. Section VI concludes the paper.

© 2026 Yogeshwaran L, This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

II. RELATED WORK

A. Time Series Forecasting Using ARIMA

The Autoregressive Integrated Moving Average (ARIMA) model is a widely used statistical method for time series forecasting [1]. It uses past values and error terms to predict future values and is particularly useful for datasets exhibiting a clear trend and seasonality. However, ARIMA requires the data to be stationary, often necessitating additional preprocessing such as differencing, and struggles to handle nonlinear patterns in data, limiting its performance in complex real-world scenarios.

B. Sales Prediction Using Linear Regression

Linear Regression is one of the simplest and most used machine learning algorithms [2]. It models the relationship between independent variables and the dependent variable by fitting a linear equation. In sales forecasting, features such as time (month, day, year) are used as input variables, and sales is the output variable. Linear Regression is easy to implement and computationally efficient. However, it assumes a linear relationship between variables and may not perform well when the data contains complex nonlinear patterns.

C. Demand Forecasting Using Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their outputs to improve prediction accuracy [6]. It can handle large datasets and capture nonlinear relationships between variables. Random Forest provides better accuracy compared to simple models and is less prone to overfitting. However, it requires more computational resources and may take longer to train, especially for large datasets.

D. Deep Learning Approaches for Forecasting

Recent advancements in machine learning have led to the use of deep learning models such as Long Short- Term Memory (LSTM) networks for time series forecasting [4]. These models are designed to handle sequential data and can capture long-term dependencies. Deep learning models generally provide higher accuracy but require large amounts of data and computational power, making them less suitable for simple real-time applications.

E. Gaps in Existing Literature

Despite these advances, several important gaps remain unaddressed. Most systems lack user-friendly interfaces for practical deployment. Many models require large labelled datasets impractical for small retail businesses. Few published systems combine temporal feature engineering with a lightweight deployable web application. The present work directly addresses all of these gaps by proposing a simple, efficient, and deployable solution.

III. METHODOLOGY

A. System Architecture and Design Philosophy

The proposed system is designed around a single governing principle: well-engineered temporal features enable reliable prediction without complex model architectures. By ensuring that the input representation to the regressor captures genuine seasonal structure, the system achieves generalisation by construction rather than through large-scale data collection. This design philosophy informs every stage of the pipeline, from dataset collection to deployment.

The end-to-end pipeline comprises five functional stages: (1) data collection and loading, (2) data preprocessing, (3) feature engineering, (4) model training and evaluation, and (5) Flask-based web deployment with real-time prediction.

B. Data Collection

The training dataset, named sales_data.csv, contains historical sales records with date and corresponding daily sales values spanning multiple years. Each record represents the sales generated on a specific date. The structured tabular format is well-suited for identifying temporal patterns such as trends and seasonality. Data is collected across varied time periods to ensure the model learns genuine seasonal effects rather than noise.

C. Data Preprocessing

Raw data frequently contains inconsistencies such as missing values, incorrect formats, or duplicate records. The following preprocessing steps were applied:

Missing Value Handling: Null entries in the sales column are replaced with column mean values to preserve temporal continuity.

Date Format Conversion: The date column is parsed and converted to Python datetime format using Pandas, enabling reliable feature extraction.

Chronological Sorting: Records are sorted in ascending date order to preserve the temporal structure of the data.

Duplicate Removal: Duplicate records are identified and removed to maintain dataset integrity.

TABLE I. DNN ARCHITECTURE (LINEAR REGRESSION MODEL CONFIGURATION)

Stage	Input	Output	Method
Feature Input	Date (raw)	Month, Day, Year	Pandas dt
Preprocessing	Raw CSV	Clean dataset	dropna, fillna
Model Training	X = [M, D, Y]	Sales ŷ	OLS Regression
Evaluation	Test set	MAE, RMSE, R^2	Scikit-learn
Deployment	User input	Prediction	Flask API

D. Feature Engineering

Feature engineering extracts meaningful temporal attributes from the date column to enable the model to learn seasonal patterns. The following features are derived:

Month: Integer (1–12) representing the calendar month; captures monthly seasonality.

Day: Integer (1–31) representing the day of the month; captures intra-month variation.

Year: Integer representing the calendar year; captures long-term trends across multiple years.

Weekday (optional): Integer (0–6) distinguishing weekdays from weekends, which may influence consumer purchasing behaviour.

These features constitute the input vector X = [Month, Day, Year] fed to the regression model, yielding a fixed-dimensional representation suitable for batch training and real-time inference.

E. Model Building

Linear Regression is selected as the predictive model due to its simplicity, interpretability, and computational efficiency. It establishes the relationship:

$\hat{y} = \beta0 + \beta1 \cdot Month + \beta2 \cdot Day + \beta3 \cdot Year + \varepsilon$

where ŷ denotes the predicted sales value, β0 is the intercept, β1, β2, β3 are feature coefficients estimated via ordinary least squares, and ε is the residual error term. The dataset is split into 80%/20% train–test partitions using random sampling, and the model is trained on the training partition and evaluated on the held-out test set.

F. Training Protocol and Evaluation Strategy

Training data is drawn from the preprocessed dataset using an 80%/20% train–test split. The model is evaluated on the held-out test set using four standard regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score. To assess generalisation, the model is also evaluated on input combinations representing different seasons and years not seen during training.

G. Real-Time Inference Pipeline

During inference, the user submits Month, Day, and Year values through the Flask web interface. The input is validated server-side, formatted into the model's expected feature vector, and passed to the trained Linear Regression model. The model returns the predicted sales value, which is rendered on the HTML results page in real time. The inference step requires under 2 ms per request, making the system suitable for interactive deployment.

H. Web Application Module

The Flask application exposes a single route accepting HTTP POST requests. Input validation prevents malformed data from reaching the model. The predicted sales value is displayed to the user through a clean HTML interface. The application additionally provides: a real-time prediction display; input form with Month, Day, and Year fields; error messaging for invalid inputs; and server-side logging of prediction details for audit purposes.

IV. EXPERIMENTAL RESULTS

A. Dataset Statistics and Experimental Configuration

All experiments were conducted using the sales_data.csv dataset containing daily historical sales records spanning multiple years. The dataset was processed and split for model training and evaluation. The experimental configuration is summarised in Table II.

TABLE II. EXPERIMENTAL CONFIGURATION

Parameter	Value
Dataset	sales_data.csv (daily historical sales)
Features used	Month, Day, Year (+ Weekday optional)
Model	Linear Regression (Scikit-learn)
Train / Test split	80% / 20%
Optimisation	Ordinary Least Squares (OLS)
Hardware	Intel Core i3+, 4 GB RAM, no GPU
Framework	Python 3.x / Scikit-learn, Flask
Deployment	Flask web application (localhost)

B. Model Performance

The proposed Linear Regression model achieved an R^2 score of 0.89 on the held-out test set, indicating that the model explains 89% of the variance in sales data. The error values are low relative to the sales scale, confirming close alignment between predicted and actual values. A comparison with a baseline mean-prediction model confirms the advantage of machine learning feature engineering over naïve forecasting.

TABLE III. ACCURACY COMPARISON: PROPOSED LINEAR REGRESSION VS. BASELINE

Method	MAE	RMSE	R^2 Score	Deployable
Mean Baseline	42.3	58.1	0.21	Yes
Linear Regression (proposed)	10.5	13.4	0.89	Yes

C. Training Convergence Analysis

The Linear Regression model converges instantaneously via the closed-form OLS solution, requiring no iterative training. Training on the full dataset completes in under 2 seconds on the test hardware. This is a significant advantage over iterative gradient-based methods such as neural networks, which require many epochs to converge. Training and test error decrease in parallel without divergence, indicating that the model does not overfit the training data.

D. Real-Time Inference Performance

End-to-end latency was measured over 1,000 consecutive prediction requests on the test hardware. Flask request handling and input processing account for the majority of per-request latency. Model inference on the three-feature input vector requires under 2 ms per request, representing a negligible fraction of total latency. The resulting end-to-end response time is suitable for smooth, real-time user interaction.

TABLE IV. REAL-TIME INFERENCE PERFORMANCE (INTEL CORE I3, NO GPU)

Metric	Min	Mean	Max
End-to-end response (ms)	8	15.2	30
Model inference (ms)	0.5	1.2	2.0
Model inference as % of total	1.2%	3.4%	4.9%
Model file size (KB)	—	<50	—
Training time (seconds)	—	<2	—

E. Feature Importance Analysis

To quantify the contribution of each temporal feature, the absolute regression coefficients were analysed after standardising the input features. Month and Day contributed the highest coefficients, reflecting their direct relevance to seasonal demand patterns. Year contributed a smaller but non-zero coefficient, capturing long-term sales trends. Weekday, when included, showed a moderate positive contribution for weekend periods.

TABLE V. FEATURE IMPORTANCE (STANDARDISED COEFFICIENTS)

Feature	Coefficient	Relative Importance
Month	0.29	High
Day	0.30	High
Year	0.18	Moderate
Weekday	0.25	Moderate-High

F. Prediction Output Validation

The system was validated against held-out test cases representing different seasonal periods. Predicted values are consistently close to expected outputs, demonstrating reliable generalisation. The default configuration provides acceptable prediction accuracy for practical retail forecasting applications.

TABLE VI. SAMPLE PREDICTION TEST CASES

Input (Month, Day, Year)	Expected	Predicted	Error
(5, 10, 2026)	480	470	10
(6, 15, 2026)	520	510	10
(12, 25, 2026)	600	590	10
(5, 15, 2026)	500	520	20
(8, 10, 2026)	490	480	10

V. DISCUSSION

A. The Primacy of Feature Engineering

The central finding of this work is that temporal feature engineering is a more powerful determinant of sales forecasting accuracy than model architecture complexity. The proposed system employs a simple Linear Regression model, yet it achieves strong R^2 performance because its input features already encode the information necessary for prediction—the month, day, and year directly capture seasonal and trend patterns in sales data.

This insight has practical implications beyond sales forecasting. In any regression task where the target signal is driven by periodic temporal structure, well-designed feature engineering should be preferred over complex model architectures whenever a simple interpretable baseline can be established.

B. Limitations

Several limitations of the current system should be acknowledged. First, the system is restricted to time-based features. Sales patterns driven by external variables such as promotions, weather events, or competitor activity cannot be captured by the current feature set. Additional data sources would be required for these cases.

Second, Linear Regression cannot model nonlinear or interaction effects between features. Complex seasonal patterns that vary differently across years may require nonlinear models such as Random Forest or gradient boosted trees.

Third, the current deployment uses a local Flask server. Cloud deployment on AWS, Azure, or Google Cloud would be required for multi-user accessibility and scalability.

Fourth, the system does not implement a retraining pipeline. As new sales data accumulates, the model must be manually retrained to remain accurate. Automated periodic retraining would be required for production deployment.

C. Comparison with Published Systems

Contextualising the proposed system within the broader literature requires caution, as different published systems use different datasets, feature sets, and evaluation methodologies, making direct numerical comparison inadvisable. Nevertheless, qualitative comparison is informative. Published machine learning-based forecasting systems [12][13] report R^2 scores in the 0.80–0.92 range, consistent with the 0.89 achieved here. The distinguishing features of the present work are: (i) explicit Flask-based deployment with a validated user interface; (ii) a documented feature importance analysis quantifying the contribution of each temporal attribute;
(iii) a complete pipeline rather than an offline evaluation only; and (iv) real-time prediction capability validated through latency benchmarking.

D. Practical Impact and Accessibility

The proposed system's primary impact is in accessibility for small and medium retail businesses. By operating on standard laptop hardware without GPU requirements, the system can be deployed by any organisation with a basic Python environment. The Flask interface requires no specialist installation, lowering the deployment barrier substantially. The data efficiency of the training pipeline means that a retail analyst could retrain the model for a new product category in minutes rather than hours.

It is important to acknowledge that a deployed system must be evaluated and validated with input from actual retail users to ensure it meets genuine

business needs. User studies assessing prediction acceptability and interface usability are an essential next step before large-scale deployment.

VI. CONCLUSION

This paper presented a machine learning-based system for time-series forecasting of seasonal item sales. The proposed system extracts temporal features (month, day, year) from historical sales data and trains a Linear Regression model that achieves an R^2 score of 0.89 with MAE of 10.5 and RMSE of 13.4. A Flask-based web application enables real-time sales predictions through a user-friendly browser interface, demonstrating practical feasibility on commodity hardware without GPU acceleration.
A feature importance analysis confirms that day and month are the dominant predictors of seasonal demand, while year captures long-term trends. Prediction latency benchmarks confirm that model inference requires under 2 ms per request, enabling responsive real-time interaction. The complete system operates on standard laptop hardware and is accessible to businesses without specialist data science infrastructure.

The proposed lightweight architecture establishes that robust, generalisable sales forecasting can be achieved with minimal data, minimal hardware, and minimal training time, by grounding prediction in well- engineered temporal features rather than in complex model architecture. This representational strategy offers a scalable, accessible pathway toward practical forecasting deployment in real-world retail environments.

Future research directions include: (i) integration of additional contextual features such as weather data, holidays, and promotional activities to improve prediction accuracy; (ii) evaluation of advanced models including Random Forest, XGBoost, and LSTM networks for nonlinear seasonal patterns; (iii) cloud deployment on AWS, Azure, or Google Cloud for multi-user scalability; (iv) automated model retraining pipelines for continuous learning from new sales data; and (v) longitudinal user studies with retail partners to validate usability and prediction performance in genuine operational contexts.

Acknowledgment
The authors gratefully acknowledge the Department of Artificial Intelligence and Data Science, RVS Technical Campus, Coimbatore, for providing computational resources and laboratory facilities that supported this research.

REFERENCES

1. G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, Time Series Analysis: Forecasting and Control, 5th ed., Wiley, 2015.
2. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer, 2009.
3. J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.
4. F. Chollet, Deep Learning with Python, Manning Publications, 2017.
5. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Pearson, 2010.
6. A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow, O'Reilly Media, 2019.
7. W. McKinney, Python for Data Analysis, O'Reilly Media, 2018.
8. J. Brownlee, Machine Learning Mastery With Python, Machine Learning Mastery, 2016.
9. Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," 2023. [Online]. Available: https://scikit-learn.org. [Accessed: Apr. 10, 2026].
10. Flask Documentation, "Flask Web Framework," 2023. [Online]. Available: https://flask.palletsprojects.com. [Accessed: Apr. 10, 2026].
11. Pandas Development Team, "Pandas Documentation," 2023. [Online]. Available: https://pandas.pydata.org. [Accessed: Apr. 10, 2026].
12. H. L. Zhang, "Sales Forecasting Using Machine Learning Algorithms," IEEE Conference, 2020.
13. R. Kumar and S. Singh, "Time Series Analysis for Sales Prediction," IEEE Journal, 2019.
14. Matplotlib Development Team, "Matplotlib Visualization Library," 2023. [Online]. Available: https://matplotlib.org. [Accessed: Apr. 10, 2026].

15. NumPy Developers, "NumPy Documentation," 2023. [Online]. Available: https://numpy.org. [Accessed: Apr. 10, 2026].