iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** X **Month of publication:** October 2025

DOI: https://doi.org/10.22214/ijraset.2025.74691

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue X Oct 2025- Available at www.ijraset.com

A Smart Shopping Assistant with ML Based Price Prediction, Alerts and Deal Predictions

P. Keerthika[1], K. Padma[2], T.T.R. Ranganayagi[3], G. Rajeswari[4]
[1,2,3]*UG Scholar,* [4]*Assistant Professor/CSE, Department of Computer Science and Engineering, K.L.N. College of Engineering, Pottapalayam, Sivagangai*

Abstract: ***Prediction is an advanced and intelligent system created to help online customers make informed purchases and save money. The core purpose of this system is to predict future product prices, seasonal discounts, and potential deals using machine learning techniques to provide customers the timely information needed to buy at the right time. The system uses advanced model systems such as Long Short-Term Memory (LSTM) networks, that are very effective in the analysis of the previous price patterns and ability to predict or forecast future patterns more accurately.The system collects real-time data using the BeautifulSoup and Requests libraries for web scraping in order to gather data, including product name, price, discounts, and availability, from various e-commerce selling platforms. The collected real-time data is then stored and managed into the MongoDB system. MongoDB is a NoSQL database, which allows for flexible data and effective management of large amounts of ongoing real-time data updates. Automated personalized recommendations and smart alerts provided by the assistant is built on the user's preferences, browsing history, and purchasing history, to make the user aware of changing prices, current deals, or the best time to make a purchase.To further add to the shopping experience, the system includes predictive analytics forms of automation and recommendations. Using predictive analytics, the system learns from user interaction with the site and the market.***

Keywords: Predictive analytics, web scraping, machine learning, price prediction, smart alerts, recommendation systems, and e-commerce optimization.

I. INTRODUCTION

In the present digital era, the nature of consumer purchasing has shifted towards online shopping. Consumers are very frequently active on e- commerce platforms—such as Amazon, Flipkart, and many more—for the purchase of daily or regular items, including electronics and food products. Online shopping is now an integral part of day-to-day life due to ease of access, convenience, and availability of many choices. Nonetheless, one major hurdle for consumers is predicting the price of a product or an item as it varies significantly due to changing consumer demand, competitive pricing from retailers, promotional or seasonal sales, and other unpredictable pricing factors, which make it difficult to assess the optimal time to buy.

Existing online shopping services offer price listings or comparisons of products and items but do not include the capability of future pricing or product discounts notification. Ultimately, users miss out on potential savings or overspend on item pricing. Thus, there is a large opportunity to create a more intelligent product price comparison application that will analyze previous and current prices, predict prices in the future and alert consumers of a better time to buy.The proposed system, A Smart Shopping Assistant with Machine Learning-Based Price Prediction, Alerts, and Deal Prediction, addresses these issues. It integrates predictive analysis, automation, and personalization. Using machine learning algorithms, it analyzes large amounts of historical price data to predict discounts or price drops.

Within the framework of the system are time-series forecasting models, namely LSTM. These models can extract short-term fluctuations and long-term seasonal price trends. By leveraging the time-series forecasting models with the available historic product datasets, the system can more accurately anticipate future price trends.

Data is key to reliable predictions. The assistant is designed to automatically scrape real-time product data containing price, review and rating information using various web scraping libraries such as BeautifulSoup and Requests. Then, the various datasets are manipulated with various other libraries like Pandas and NumPy to remove duplicate variables, handle null values, and standardize variables. The final cleaned datasets are compiled into a structured format into a database, typically MongoDB, to enable a fast and efficient way to query and analyze the data for continual model training and predictive analysis.

In addition to predictive analytics, the system enhances user engagement through personalized recommendations and intelligent alerts. The system leverages content-based filtering to recommend products that are comparable to products the user has purchased or previously viewed on the platform.

©IJRASET: All Rights are Reserved

This alert module pushes real-time alerts through email, SMS, or in-app notifications when a predicted price drop or new deal is available. This way, users receive an alert when there is a current deal, instead of receiving an alert for a deal that has already expired.

The Smart Shopping Assistant is designed for scalability and efficiency with multiple components for data collection, preprocessing, applying machine learning model, alert generation, and data visualization. The dashboard interface uses visualization libraries such as Matplotlib and Plotly to display real-time price information, predicted trends, and information about deals in simple and understandable forms. Using this dashboard, users will be able to track price changes over time, compare platforms, and make informed purchase decisions.

Moreover, the system utilizes ongoing learning, in which prediction models are retrained frequently with updated data. This allows for improved accuracy in the forecasts and maintains the effectiveness of the model as the market evolves. The automated systems used in combination with machine learning reduce the need for manual monitoring while providing intelligent assistance for the online shopper.

II. METHODOLOGY

The proposed Smart Shopping Assistant with Machine Learning-Based Price Prediction, Alerts, and Deal Prediction takes advantage of multiple steps, such as data collection, data preprocessing, predictive modeling, recommendations, and alert generation. The original workflow is building an automated working environment for real-time price detection, prediction, and notification to consumers, ensuring that it is accurate and functioning as designed.

The Smart Shopping Assistant will initially crawl product data from e-commerce websites, such as Amazon and Flipkart, consisting of relevant information (e.g. name, category, price, ratings, availability). The web scraping packages, BeautifulSoup and Requests, will be utilized to automatically crawl product data over a predetermined time, following scheduling routine to allow the dataset to be as current as possible to capture any changes to pricing. After crawling the data set, the data set will require pre-processing to account for duplicity and missing values to contain either complete missing data or removed to normalise the standard format of numerical data and text. Following prep processing state, transformation and normalisation will take place to determine if the training set is useful. Once cleaned, the data will be properly stored in the MongoDB database to efficiently run the Smart Shopping Assistant and easily access and analysis of data.

Upon data preprocessing steps, machine learning models are trained to forecast price fluctuations based on recurring price patterns from historical data. The models utilize statistical and deep learning approaches such as Long Short Term Memory (LSTM) to analyze time-series data. LSTM has the advantage of capturing complex, non-linear price fluctuations of a product. To improve the forecasting performance and stability of the machine learning models, a hybrid and ensemble model is used to combine the ARIMA and Prophet models. The models are used to make predictions, which inform the users regarding the best time to purchase products, helping them prepare for upcoming discounts and deals before they happen.

After the predictions are generated, the system applies a recommendation and alerting mechanism to better facilitate user engagement. The recommendation module utilizes content-based filtering using TF-IDF and cosine similarity to recommend products that are similar to products previously browsed by users and their preferences. Meanwhile, the alert system continuously monitors the predicted data and automatically alerts the user through an email, sms or app, when there is a deal or price decline price drops over the user's threshold for significant savings. This allows users to benefit from offers proactively instead of reactively.

Finally, an interactive dashboard displays all essential outputs, including live prices, forecasted trends, and personalized product recommendations. Visualization tools such as Matplotlib and Plotly are used to present this information clearly and intuitively. The system also incorporates a continuous learning mechanism that retrains the models with newly collected data, enabling it to adapt to evolving market behaviors and maintain long-term accuracy. Through this combination of automation, predictive Finally, an interactive interface displays all critical outputs, including live prices, projected trends, and tailored product suggestions. To communicate this information clearly and intuitively, there are several visualization tools, including Matplotlib and Plotly. Additionally, the system has a continual learning model that retrains the models with the newly collected data, which allows it to become accustomed to the changing market behaviors and provide reliable predictions long-term. All together, when combining automation, predictive analytics, and user personalization, the Smart Shopping Assistant provides a reliable, intelligent, and time-efficient experience for today's online shopper.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

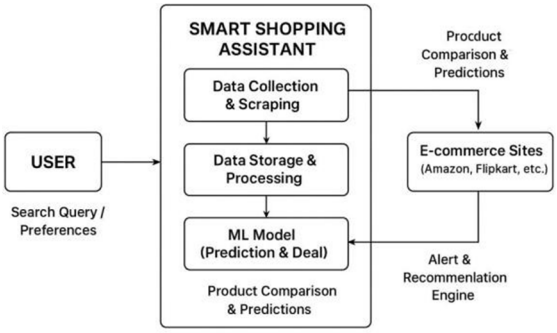


Fig.1 Flow Diagram

III. SYSTEM DESIGN

The Smart Shopping Assistant is developed as a minimal, service-oriented platform that deploys functionality as separate but interdependent components. This composable design establishes a data-centric workflow in which data move from ingestion to analysis and predictions to user-facing display. At the input stage, the system employs lightweight web scrapers and API connectors that continue to ingest product specifications, like prices, ratings, and stocktag status, from multiple e-commerce sites. The data ingestion methods pipeline the data ingestion, separating data collection from data processing allowing for consistent usage patterns even under heavy loads. A modular design enables efficient data collection and handling, scalability across multiple sources, and promotes maintainability when adding new sites or APIs.

Once the data is ingested into the processing pipeline, a preprocessing module accomplishes a series of critical discussion tasks to get the data ready for analysis. These tasks include removing duplicates, converting currency and units to be standard across all selected sites, filling in missing values, and creating time-based features of interest such as day of week and percent change. Text data like product titles, descriptions, and review comments are tokenized, and vectorized, using TF-IDF or embeddings, to prepare them for computation tasks of interest such as similarity, clustering, and recommendation. The cleaned and transformed data is stored in a relational database some of which will allow for filtered, time series queries while metadata is stored in a separate database.

The analytical and decision-making layer provides intelligence to the system. For example, forecasting models such as LSTM leverage historical price data to predict future price trajectories and an ensemble mechanism can aggregate these outputs to improve precision and reliability. A recommendation module utilizes vector similarity to recommend alternative products on the market, uses forecast data to adjustprioritization of items most likely to decrease in price. The alerting system continuously tracks all forecasting and will notify when user- defined thresholds have been reached or general business rules have been met. Alert is sent out via email, in-application or SMS to quickly inform the user of an opportunity.

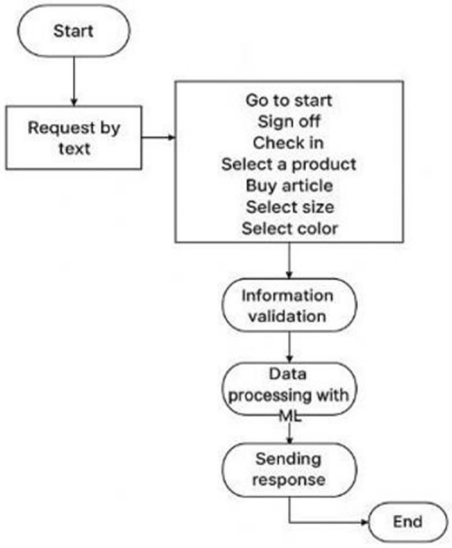


Fig.2 System Design

IV. SYSTEM ARCHITECTURE

The architecture is based on a layered microservices model, promoting scalability, modularity, and resilience. At the bottom is the data layer, which includes relational databases for structured records, a document store or columnar store for vector data, and a message queue to asynchronously complete background tasks. All these components combine to decouple processes involved in scraping, preprocessing the data, and creating alerts, which increases fault tolerance and throughput; though they are coupled with asynchronous messaging, each service can run independently of each other in failures or varying loads. In addition, we maintain a distributed cache, generally Redis, to improve responsiveness, which holds frequently accessed queries and prediction results fostering an improved user experience by increasing performance and decreasing core database load as queries can originate in the service caching and not the database.

Above the data layer is an application layer that runs multiple specialized microservices that can be packaged independently to predict, recommend, and alert. The prediction service contains the trained models and exposes REST APIs to send requests for inference on new data. Model training jobs are run in allocated resources at scheduled intervals, and that way they complete outside of the service's execution. In the service we subscribe to new model parameters which push appropriate changes into the inference microservice. The recommendation and alerting modules communicate through event-driven messaging as well allowing for adaptive scaling of the cluster based on load. The alert orchestrator monitors price forecasts and applies decision rules to determine how and if to notify the user. The services communicate securely through the API gateway utilizing the decoupled boundaries, allowing independent deploys and simple service upgrades and updates.

The interface and operational layers are at the top of the architecture, which manages the presentation, security, and deployment of the system. The front-end dashboard presents data visualization insights to the user by connecting to backend services using a REST API and WebSockets, displaying historical price charts, trade prediction signals, and personalized recommendations to the user. Tools for containerization and orchestration (e.g. Docker and Kubernetes) are being used to manage the deployment, scaling, and failover of the system. Continuous integration and monitoring pipelines support the oversight of scraping performance, latency of models for notifications sent to the user, and alert delivery metrics to facilitate timely diagnosis. Finally, security considerations are integrated throughout the architecture, including encrypted communication, OAuth authentication, and secure credential management. Overall, the architecture is designed to operate with high performance, real-time, and security to facilitate future.

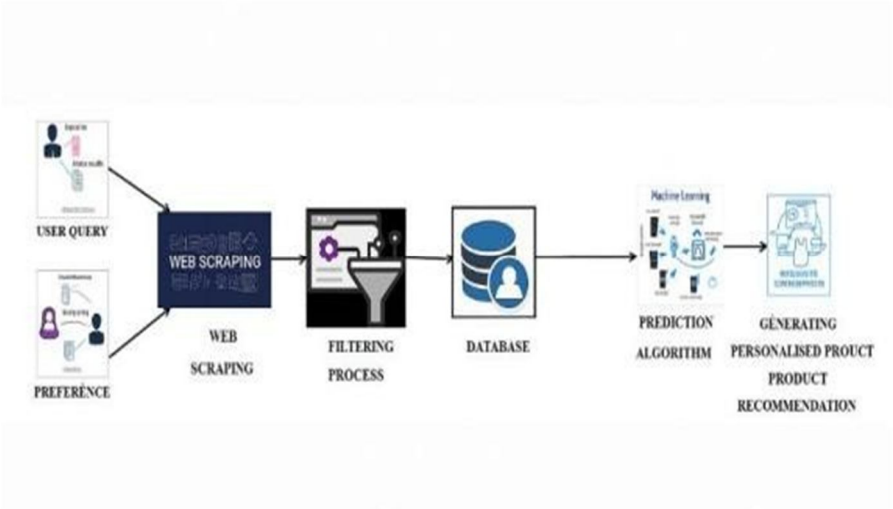


Fig 3. System Archietecture

A. Data Collection Using Web Scraping

The Smart Shopping Assistant relies on the process of data collection as the foundation for machine learning predictions and monitoring deals in real-time. Product-level data points such as product names, prices, ratings, reviews, and stock status, are extracted from the largest e- commerce platforms as Amazon and Flipkart via a process called automated web scraping, which depends on bit of code running on a web server to extract the data. Libraries such as Selenium, and Requests can be utilized efficiently to parse and retrieve data from client web pages. The scrapers are designed to run on a periodic basis and are structured to obtain both current prices and historical price data to build a complete time-series data set. The idea is to continuously scrape product data, so the current product database is always compiling and providing the best information for prediction and analytics.

The raw data is "cleaned" before reuse to reduce inconsistencies. Duplicates, missing values, and non-standard formats are eliminated. All currency and unit conversions are made to form a singular "currency" across data points before standardization and normalization occur. Product data points and price can then be structured and stored in a relational database such as MongoDB or other standard data source to obtain more structured data for retrieval and efficiency. Relevant data points such as time stamps to denote scraping time, product category tags, and signs of discount will also be appended to a structured or unstructured product object, which would allow the prediction models to account for seasonal or trend-based product data and pricing.

The web scraping process has been specifically developed both to be effective and to operate ethically. The scrapers have implemented rate limiting, random delays, and polite request headers to prevent overloading the e-commerce platform websites and violating e-commerce platform policies. If an official API is available, scrapers will use it to collect authenticated data. The data is then securely stored and updated automatically with the help of schedulers in order to work indefinitely without human involvement. This structured and automated data collection pipeline lays a strong automated foundation for accurate price predictions, alerts, and recommendations in the Smart Shopping Assistant system.

B. Discount Prediction Using LSTM Algorithm

The LSTM (Long Short-Term Memory) algorithm is a deep learning model that fits within the broader family of recurrent neural networks (RNNs) specifically designed to learn temporal dependencies of data in sequences. In the case of the Smart Shopping Assistant system, the LSTM model is utilized to predict the likelihood of discount in the future by analyzing price patterns, seasonality and promotional cycles using price data from previous time periods collected through web scraping. In contrast to traditional statistical modeling methods, such as ARIMA or Prophet, that assume linear dependencies, LSTM networks can capture complex, nonlinear dependencies that occur over long spans of time - making them an advantageous technique for forecasting e-commerce prices. The price history of each individual product of interest is gathered from online retailers, such as Amazon or Flipkart, and is passed to the LSTM model as a time-series sequence.

Inside the LSTM structure, three gates—forget, input, and output gates—decide how information moves through time steps. The forget gate says what past information to forget, the input gate determines how much information to save, and the candidate cell state provides updated information to learn. These functions update the cell's memory in every time step by combining past knowledge with today's new input. The output gate produces a new hidden state that applies the learned time-sensitive knowledge. By alternating through time to remember long- term dependencies while disregarding irrelevant variability, the model learns to identify when a price trend or discount is occurring in the e- commerce data. The result of the output provides a prediction of price for information for a future period, which is compared to today's price to establish a potential discount percentage.

Forget Gate:

$$ft = (Wf[ht-1, xt] + bf)$$

Input Gate:

$$it = (Wi\,[ht-1, xt] + bi)$$

Output Gate:

$$ot = (Wo[ht-1, xt] + bo)$$

Prior to training, all collected price data is subjected to preprocessing and normalization to ensure stable convergence. The historical prices are segmented into time windows, representing fixed-length sequences, allowing the model to learn from past movements and to predict the next expected value. During training, the objective is to minimize the difference between predicted prices and actual prices using standard loss functions like mean squared error. Once adequately trained, the model can then predict future prices for each product, and if a predicted price drops significantly compared to the previous price, the system flags it as a potential discount event and provides the user an alert. Prediction accuracy is validated based on forecast performance metrics like mean absolute error and root mean square error, comparing how closely the predicted prices come to what was observed in real market data.

Predicted Discount Percentage:

$$Discount_{predicted} = \frac{P_{current} - P_{forecast}}{P_{current}} \times 100$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - y_i)^2$$

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - y_i|$$

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Overall, the LSTM discount prediction method improves the system's capacity to accurately predict drops in price compared to conventional approaches. By leveraging the ability to discern long-term temporal dependencies, and more accurately learn the relationships within product pricing data, the system is able to provide advance notice of predicted discounts Informed shopping decisions can be made by consumers, as they can proceed on a planned basis with their purchases before prices go back up - while also obtaining useful savings in their purchasing costs. And through continual retraining against new data the system can continuously train.

C. Machine Learning Models with Database

In this project, we employ the Long Short-Term Memory (LSTM) model to forecast discount trends from historical sales data. LSTM is a type of recurrent neural network meaning it has particularly good performance for time-series prediction as it can find long-term relationships in sequential data. The architecture relies on memory cells and gates, which control whether information is ignored or passed on, compounding the ability of the model to capture patterns in discount variability over time. The LSTM model is trained using historical transaction data and learns from sequences of sales, information about seasonal effects, and previously provided discounts to predict pricing strategies.

For data management, we use MongoDB, a NoSQL database, because it effortlessly works with semi-structured data like JSON documents. MongoDB optimally stores the sales records, product data, and historical discounts providing scalable capabilities for both read and write operations. This capability is particularly useful as the sales field evolves. MongoDB's dynamic schema ensures that new data can be added to the existing datasets without restructuring the data. We are able to extract pre-processed data without complication. MongoDB coupled with LSTM provides a robust pipeline to forecast discounts based on data.

D. Streamlit Dashboard Visualization

To create an interactive and friendly user experience for the discount prediction system, a Streamlit dashboard is introduced. Streamlit is used to create real-time visualization of the LSTM model's predictions, allowing users to access historical discount history, predicted values, and essential sales metrics easily.

The dashboard completes retrieves data from MongoDB, ensuring the status of the data they are viewing is contemporary and not days or months old. The various visualization design elements, including line charts, bar charts for visualizing actual versus predicted discounts, and tables with various relevant metrics, allow different stakeholders to view trends, as well as a variety of metrics to analyze the model's performance here too. Students and professors, and marketing executives can use Streamlit's visualization features, including the sliders, dropdowns, and date selection tools, to dynamically filter products, time periods, or categories to create a unique lens for viewing data. Not only does this integration enhance the interpretability of the LSTM's predictions, but it also provides ease of transforming overwhelming sequential series data into meaningful, actionable data to inform pricing or marketing initiatives.

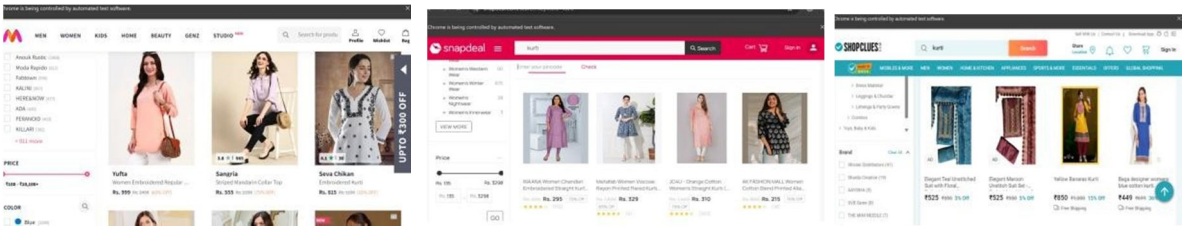


Fig.4. Web Scrabing

This project uses a Long Short Term Memory (LSTM) model to forecast trends in discounts using historical sales data. LSTM is a unique brand of recurrent neural networks that is designed to work with sequentially formed data over time, which includes capturing long-range dependencies. Its memory cells, gating units, and other mechanics help it retain relevant information and filter out useless data patterns. This makes LSTM an excellent modeling option for forecasting discount behavior over elongated periods of time. By using sequences of prior sales, trends by seasons, and prior discounts in forming the learned accuracy, the predictions provided to the user will create the ability to have a more efficient basis for pricing planning.

To manage the underlying data structure, MongoDB was used as the database. MongoDB is based on a flexible, scalable database structure. MongoDB performances compared to traditional relational databases do not require a static schema. This is important because sales transactions, product-level details, and historical discount records can be stored as JSON-like documents. This enables the mocking of new data fields without the need to modify the existing structure of the previously loaded file type. Additionally, MongoDB can perform quick read and write operations, making it easier to extract and pre-process the data for training and testing the LSTM. Using MongoDB allows for clean, structured, and updated datasets as the foundation of the predictive process. A Streamlit dashboard is implemented to visualize the predictions and allow action upon the predictions

V. BEST DEAL AND ALERT MODULES

The Best Deal feature is set up to determine the best discounts for customers based on what is predicted with prices going forward over the longer term, as well as historical pricing data. It uses predicted discount data in the LSTM model to highlight products that are most discounted or are predicted to be the best price discount over a given time. This feature compares predicted prices to what the price point historically has been, as well as current market prices, to ensure that users are getting the best deals. In addition to positively impacting user engagement, the Best Deal feature also allows the retailer or brand to make pricing decisions guided toward pricing actions based on which items would have the most reason to sell based on discount quality.

The Alert feature extends the Best Deal feature by giving users a real-time alert of when the brand or retailer has the opportunity to take advantage of significant or good discounts. Being integrated with MongoDB, the system sits and monitors product prices, predicted discounts, and user preferences. Once the product meets a threshold that the brand or retailer would deem excess, for example the discounted price had gone down a certain percentage, it would alert in either the Streamlit dashboard or by automation. Using this method of alert or notification, the system increases overall user satisfaction and brand loyalty or retailer loyalty by offering users a reason to avoid or not miss a deal.

Both modules were effortlessly incorporated into the Streamlit dashboard so that users can interactively explore deals and alerts. Visual elements such as tables, highlight boxes, and charts that respond to changes in data allow users to compare deals, analyze alert histories, and filter notifications by category, price, and percentage off in an easy way. This cohesion ensures that complicated predictive data from the LSTM model is easily converted into action, allowing customers and decision-makers to rapidly and practically identify opportunities while also maintaining a user-friendly experience.

VI. RESULT AND DISCUSSION

The suggested discount prediction system yields positive forecasts for future discounts using past sales data. Overall, the LSTM model creates meaningful representations from the time-based structure of sequential sales data and learns trends that traditional models do not capture. Forecasting is motivated by two factors: if the model has been trained on past sequences of discounts, and if the model understands the seasonality of discounts. The model predicts similar accuracy to historical value discounts. Lastly, forecasting was tested in a number of product categories consistently demonstrated aspects of sensible price behavior confirming suitability of LSTM models for forecasting problems in time series.

The integration with the MongoDB ensures the model has and continuous access to clean, structured, and scalable data for model training and prescriptive analysis. Historical transactions (in addition to product characteristics) are quickly queried from a document-based storage and retrieval paradigm. This process is efficient for maintaining speed and transformation amount at times of quick data pre-processing into the annotation data pipeline all while providing the ability to continually train the model as new sales occur. The database supports a dynamic schema, allowing immutable data as defined by rapid or account changes, without restructuring any data, which is important for ensuring reliable forecasts as some products or categories could be expected to obtain discounted pricing on sale.



Fig.5 View of Database

Visualizing with the Streamlit dashboard makes the results more interpretable. The user can see the predicted discounts together with actual historic trends in clear line charts, bar graphs, and tables. Using sliders, dropdowns, and date filters to interactively filter stakeholders to examine specific products, times (stats for timeframes), or product categories for a narrower view of the data. This means the stakeholders visually isolate cases and will likely assist them in identifying trends, tracking potential drifts, and justifying the forecasted views, which adds to the level of transparency and trust in the system as a whole.

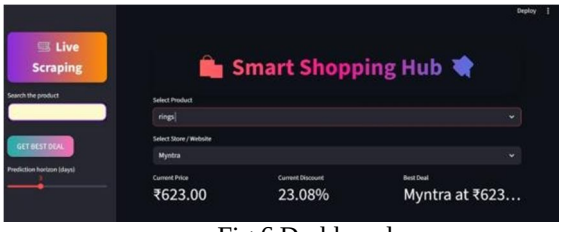


Fig.6 Dashboard

In addition, the Best Deal module uses the model's predictions to identify the best products with the biggest discount. By comparing forecasted prices to historic trends, the system identifies the best users might want to buy. This is a valuable tool for both customers interested in the best discount and businesses trying to push products for sales. In testing, the module performed very well and was able to successfully showcase value deals in line with previous trends in the marketplace, thus validating its use.



	date	price	predicted_discount	estimated_price
0	2025-10-11	623	23.08	479.21
1	2025-10-12	623	23.08	479.21
2	2025-10-13	623	23.08	479.21
3	2025-10-14	623	23.08	479.21
4	2025-10-15	623	23.08	479.21
5	2025-10-16	623	23.08	479.21

Fig.7 Discount Prediction

The Alert module, likewise, delivers notifications in real-time whenever the discounted predictions surpass a set limit. This allows the user to stay informed about a better deal faster, which will help an overall positive user experience. The alert module accurately activated based on the testing's predetermined conditions, confirming the alert module's functionality. The combination of predictive accuracy, real-time alerts, and easy-to-use visualizations allows the user to quickly access smarter solutions. Thus, the predictive analytics and data become actionable insights.



Fig8 Alert Message

In summary, the findings suggest that the combined system: LSTM predictions, MongoDB storage, and Streamlit visualizations, along with Best Deal and Alert modular add-ons, provide an efficient discount forecasting system. The combination of: accurate predictability, real-time alerts, and user-interactive visualization will result in a better decision-making framework and an improved user experience. This indicates that the use of this type of pipeline can be generalized to other retail or e-commerce use cases to aid in strategic discount pricing decisions and improve shopper satisfaction based on better visibility into real-time data.

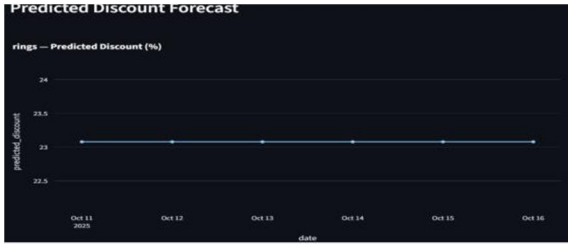


Fig.7 Discount History

VII. CONCLUSION

The recommended discount forecasting solution successfully combines LSTM-based forecasting, MongoDB data management, and Streamlit visualization to provide actionable insights and reliable recommendations for pricing strategies. The solution takes advantage of LSTM's sequential learning capabilities to forecast discounted prices based on historical sale sequences, utilizing trends that traditional models tend to overlook. MongoDB allows for a scalable and flexible data architecture for querying, storing and preprocessing large datasets on sales history. The Streamlit application brings the complexities of the predictive analytics into a friendly, interactive dashboard with Best Deal and Alert modules that can also notify users when to take action and assist in identifying the discounted price of an item.

As a whole, the system is functional, scalable and can be practically applied to live retail and e-commerce environments. The results show that predictive modeling combined with dynamic visualization and alerting mechanisms assists in making decisions for businesses and customers. Unlike theoretical implementations or improvements in technology- all which were demonstrated in the project- taking advantage of both machine learning and modern database and dashboard technology can lead to intelligent data-driven systems. Future enhancements could include additional features such as customer analysis, competitor pricing data, and streaming live updates or data to enhance prediction accuracy

VIII. FUTURE ENHANCEMENT

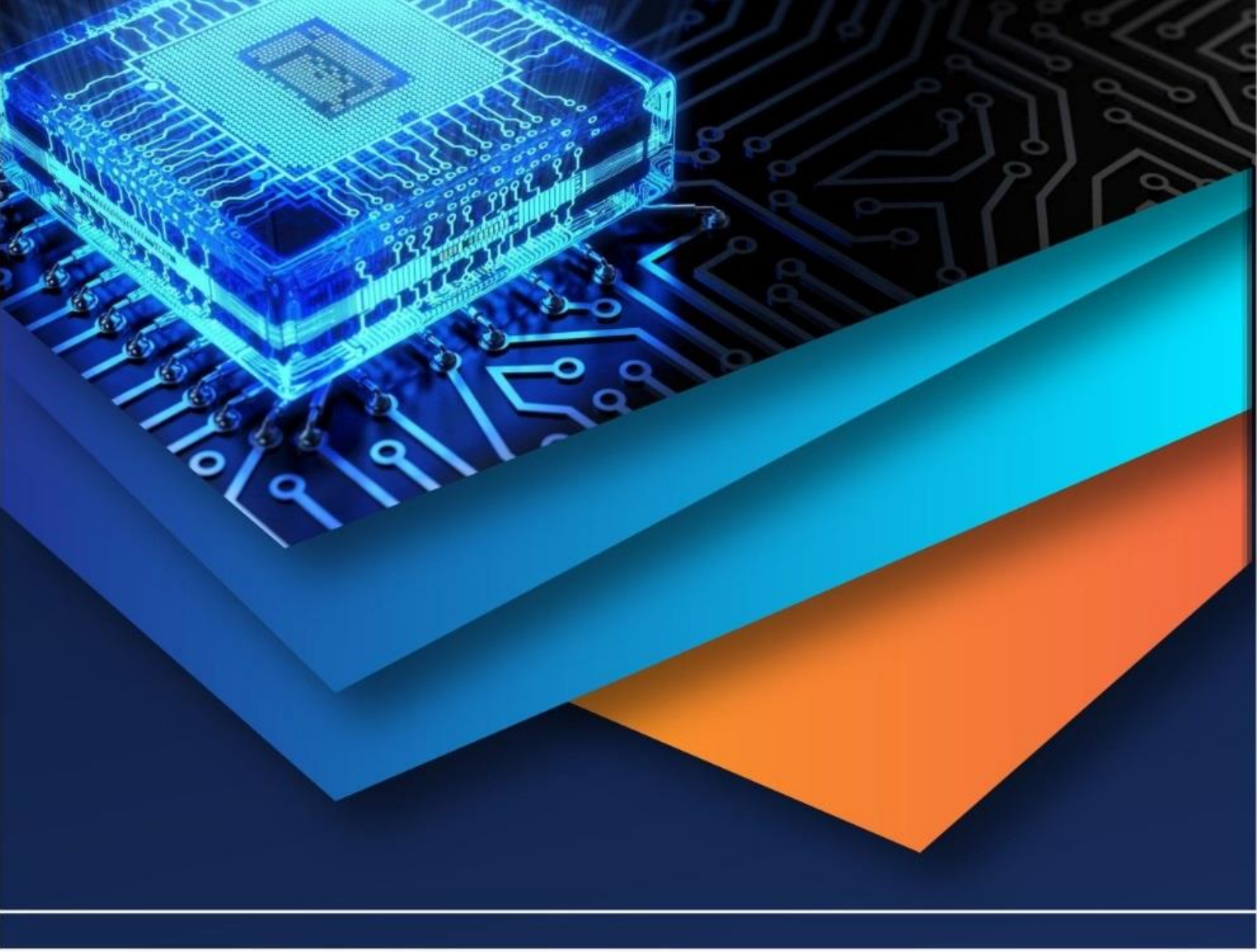
To extend its global market reach and enable better deal discovery, the system could integrate multiple eCommerce platforms through APIs so the user has a broader choice of items to compare prices on. Sophisticated deep learning models (such as Transformers) can be leveraged to increase the accuracy of price trend forecasting and personalized product recommendation, which could provide the user compelling data informed solutions around price and budget spending.

Additionally, the platform could even add voice assistant capabilities so the user can receive price predictions, offer alerts, and deals updates via smart speakers or voice apps on mobile for convenience. Bridging product searches based on images using computer vision technology would be helpful by allowing the user to search for prices of similar products simply by uploading their product. Furthermore, the platform could include budgeting/savings goals, which in return enables the app to recommend deals based on each user's financial behavior patterns enabling even more customized and enhanced intelligent shopping experience.

©IJRASET: All Rights are Reserved

REFERENCES

[1] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
[2] Brownlee, J. (2018). Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.
[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
[4] Chodorow, K. (2013). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media.
[5] Streamlit Inc. (2023). Streamlit Documentation - Create Interactive Web Apps for Machine Learning and Data Science. https://docs.streamlit.io
[6] Brownlee, J. (2020). Time Series Forecasting With LSTMs in Python. Machine Learning Mastery. https://machinelearningmastery.com
[7] Raschka, S., & Mirjalili, V. (2019). Python Machine Learning (3rd ed.). Packt Publishing.
[8] Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks. Springer.
[9] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In ICLR 2015.
[10] Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14(1), 35-62.
[11] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Forecasting methods for statistical and machine learning: Issues and recommendations. PLoS ONE, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889
[12] Zhang, C., & Lu, Y. (2021). Artificial Intelligence in E-commerce: A survey and manager implications for pricing prediction and recommendation systems. Electronic Commerce Research and Applications, 48, 101073. https://doi.org/10.1016/j.elerap.2021.101073
[13] Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and practice (3rd ed.). OTexts. https://otexts.com/fpp3/
[14] Sun, P., Ma, X., & Zhao, Y. (2020). Short-term E-commerce sales forecasting using a hybrid deep-learning model. IEEE Access, 8, 155850– 155860. https://doi.org/10.1109/ACCESS.2020.3019355
[15] Liu, Y., Singh, P., & Sidorov, K. (2022). Intelligent price prediction and dynamic deal alert generation through hybrid machine learning approaches. Expert Systems with Applications, 198, 116870. https://doi.org/10.1016/j.eswa.2022.116870



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)