

SMART- COST : A HYBRID AI-DRIVEN POS AND COST ANALYTICS SYSTEM FOR MICRO-SMES

INDIVIDUAL SOLUTION REPORT

D.D.M Jayasingha

IT22162250

Product Demand Prediction System

R26-IT-026

B.Sc. (Hons) Degree in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2026

SMART- COST : A HYBRID AI-DRIVEN POS AND COST ANALYTICS SYSTEM FOR MICRO-SMES

D.D.M Jayasingha

IT22162250

Product Demand Prediction System

R26-IT-026

B.Sc. (Hons) Degree in Information Technology Specializing in Information
Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2026

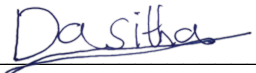
DECLARATION AND SUPERVISOR CERTIFICATION

Candidate declaration. I declare that this thesis is my own work and does not incorporate, without acknowledgement, material previously submitted for a degree or diploma at any university or institute of higher learning. To the best of my knowledge, it contains no material previously published or written by another person except where acknowledgement is made in the text. I grant the Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute this thesis, in whole or in part, in print, electronic, or other media, while retaining the right to use the content in future work.

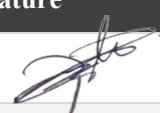
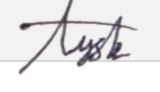
Academic integrity and AI disclosure. I have disclosed all use of generative AI and other automated tools in accordance with the module policy. I remain responsible for the accuracy, originality, citations, code, data, analyses, and conclusions in this thesis.

Student name: Don Dashitha Madukara Jayasingha.

Student ID: IT22162250

Signature:  Date: 10/09/2026

Supervisor certification. The above candidate carried out this research for the bachelor's degree thesis under my supervision.

Role and name	Signature	Date
Supervisor: Mr. Deemantha Siriwardana		10/09/2026
Co-supervisor: Ms. Ayesha Wijesooriya		10/09/2026

ABSTRACT

Micro and small enterprises often experience difficulties in estimating future product demand because purchasing and inventory decisions are frequently based on recent sales observations or personal judgement rather than systematic forecasting. This can contribute to stock shortages, unnecessary inventory, product wastage, and inefficient resource allocation. This research addresses this problem through the development of a Product Demand Prediction component for the Smart-Cost platform, with the objective of transforming historical Point-of-Sale (POS) transactions into actionable product-level demand forecasts. The proposed approach prepares daily demand histories from sales records and applies two complementary forecasting techniques: Prophet, for modelling temporal trends and seasonal patterns, and a Multilayer Perceptron (MLP) neural network, which learns nonlinear relationships from seven-day historical demand sequences. A Hybrid Demand Prediction approach was subsequently designed to utilize the complementary forecasting characteristics of the two models. The component was implemented using Python-based machine-learning services and integrated with the Smart-Cost backend and user interfaces, enabling product-specific forecasting, model comparison, and historical demand analysis across 7-, 30-, and 365-day periods. Model performance was assessed using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). During experimental evaluation of the Bread product, the neural-network model achieved an MAE of 2.43, MSE of 8.61, and RMSE of 2.94, demonstrating the feasibility of predicting short-term product demand from historical sales patterns. The individual contribution of this research comprises demand-data preparation, development and evaluation of the Prophet and neural-network forecasting models, design of the hybrid forecasting approach, and integration of the prediction workflow into Smart-Cost. The findings establish a practical foundation for data-driven demand planning within micro-SME environments while identifying real-world dataset validation and further hybrid-model optimization as important areas for future research.

Keywords: Product Demand Prediction, Demand Forecasting, Prophet, Neural Network, Hybrid Forecasting

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research supervisor for the continuous guidance, constructive feedback, and academic support provided throughout the development of my Product Demand Prediction research component. The recommendations received during the research process were valuable in refining the forecasting methodology, evaluating the selected models, and improving the overall quality of the research.

I am also grateful to the academic staff and research panel members who provided valuable observations and recommendations during project evaluations and presentations. Their feedback contributed to improving the experimental approach and strengthening the integration of the demand prediction functionality within the Smart-Cost system.

I would like to acknowledge my Smart-Cost project team for their cooperation during system integration and testing. Their support enabled my individual component to interact successfully with the broader Smart-Cost environment while allowing me to concentrate on the design and development of the Product Demand Prediction component, including data preparation, Prophet forecasting, neural-network-based forecasting, hybrid prediction, model evaluation, and system integration.

Finally, I would like to thank my family, friends, and everyone who supported and encouraged me throughout the research and development process. Their encouragement was greatly appreciated during the implementation, experimentation, testing, and documentation stages of this research.

TABLE OF CONTENTS

DECLARATION AND SUPERVISOR CERTIFICATION	i
ABSTRACT	ii
LIST OF FIGURES	Error! Bookmark not defined.
LIST OF ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Overall project context.....	2
1.2 Individual component context.....	4
1.3 Problem statement.....	5
1.4 Research gap	6
1.5 Aim and objectives.....	8
1.6 Research questions or hypotheses	9
1.7 Scope limitations and assumptions	11
1.8 Individual contribution.....	13
1.9 Thesis structure	15
2 LITERATURE REVIEW AND RELATED WORK	17
2.1 Review method and scope.....	18
2.2 Domain and theoretical foundations	19
2.3 Related research	20
2.4 Existing systems products and techniques	25
2.5 Gap synthesis and conceptual framework.....	27
3 RESEARCH METHODOLOGY.....	31
3.1 Research design.....	32
3.2 Data participants or study materials.....	33
3.3 Experimental or study procedure	36

3.4	Measures and evaluation metrics	41
3.5	Data analysis	42
3.6	Ethics privacy security and responsible AI.....	44
3.7	Reproducibility and research artefacts	45
4	PRODUCT AND SOLUTION DESIGN.....	47
4.1	Requirements.....	48
4.2	Component architecture within the complete solution	50
4.3	Design alternatives and decisions	53
4.4	Detailed design.....	56
4.5	Standards and professional practice	59
5	IMPLEMENTATION INTEGRATION AND TESTING	63
5.1	Development environment and technology stack	64
5.2	Component implementation	66
5.3	Integration with the other components.....	73
5.4	Verification and software testing	75
5.5	Deployment and operation	79
6	RESULTS AND VALIDATION.....	82
6.1	Research results.....	83
6.2	Product and technical results.....	90
6.3	User or stakeholder validation	91
6.4	Integration results.....	92
6.5	Summary of findings.....	94
7	DISCUSSION	97
7.1	Interpretation of findings.....	98
7.2	Comparison with literature baselines and existing solutions	99
7.3	Research contribution.....	100

7.4	Product contribution.....	102
7.5	Limitations and threats to validity	103
7.6	Implications.....	106
8	CONCLUSION AND FUTURE DIRECTIONS	108
8.1	Conclusion	109
8.2	Achievement of objectives.....	110
8.3	Individual research and product contributions.....	111
8.4	Future directions.....	112
	REFERENCES.....	114
	APPENDICES	117
	Appendix A Research instruments or extended results.....	117
	Appendix A .1 Product Demand Prediction Experimental Dataset	117
	Appendix A .2 Data Preprocessing Workflow.....	117
	Appendix A .3 Prophet Model Configuration.....	118
	Appendix A .4 MLP Model Configuration	119
	Appendix A .5 Hybrid Demand Prediction.....	120
	Appendix A .6 Extended Preliminary Evaluation Results	120
	Appendix A .7 Product Demand Prediction Integration Evidence	121
	Appendix A .8 Technical Evidence to Include	122
	Appendix A .9 Additional Test Cases.....	127
	Appendix B Academic integrity and AI use disclosure	127
	Appendix B .1 Academic Integrity Declaration.....	127
	Appendix B .2 Generative AI Use Disclosure	128
	Appendix B .3 Verification of AI-Assisted Material	129

LIST OF FIGURES

Figure 1:Position of the Product Demand Prediction component within the SmartCost solution.....	3
Figure 2:Product Demand Prediction workflow within the SmartCost POS platform	5
Figure 3:Conceptual framework for the Product Demand Prediction System.....	28
Figure 4:Comparison of individual and hybrid forecasting approaches	29
Figure 5:Research methodology pipeline for Product Demand Prediction	33
Figure 6:Historical POS sales data preprocessing and transformation workflow.....	37
Figure 7:Sliding-window data preparation for the MLP forecasting model	39
Figure 8:Experimental training and evaluation workflow	40
Figure 9:Product Demand Prediction component within the overall SmartCost architecture.....	48
Figure 10:Detailed architecture of the Product Demand Prediction System	51
Figure 11:Detailed System Architecture of the SmartCost Product Demand Prediction System.....	52
Figure 12:End-to-end Product Demand Prediction processing flow	55
Figure 13:Hybrid forecasting architecture combining Prophet and MLP models	56
Figure 14:Product Demand Prediction API interaction sequence.....	57
Figure 15:Product Demand Prediction implementation architecture.....	66
Figure 16:Prophet forecasting model implementation workflow	67
Figure 17:MLP neural network forecasting implementation workflow	68
Figure 18:Seven-day sliding-window forecasting process used by the MLP model .	69
Figure 19:Hybrid demand prediction execution process	70
Figure 20:SmartCost Demand Prediction Dashboard	71
Figure 21:Product selection interface for demand forecasting	71
Figure 22:Demand prediction generation interface.....	72
Figure 23:Product demand forecast result interface Figure 24:Product Demand Prediction history interface	72
Figure 25:API communication between the SmartCost backend and Python ML service	74
Figure 26:Actual and predicted product demand for the MLP model	80

Figure 27:Actual and predicted product demand for the MLP model	83
Figure 28:Prophet seven-day product demand forecast	84
Figure 29:MLP seven-day product demand forecast	85
Figure 30:Comparison of Prophet and MLP forecasting results.....	86
Figure 31:Hybrid Product Demand Prediction result	87
Figure 32:Forecasting model performance comparison using MAE and RMSE	90
Figure 33:Successful Request-to-Response Execution Sequence of the Product Demand Prediction System.....	93
Figure 34:SmartCost Mobile Application Interface Displaying Seven-Day Product Demand Forecast Trends	
Figure 35:SmartCost Mobile Application Interface Displaying Fourteen-Day Product Demand Forecast Trends	122
Figure 36:SmartCost Mobile Application Interface Displaying Thirty-Day Product Demand Forecast Comparison	124
Figure 37:SmartCost Mobile Application Interface Displaying Product-Wise Demand Forecasts for 7, 14, and 30 Days	125
Figure 38:SmartCost Mobile Application Interface Displaying Thirty-Day Product Demand Share Distribution.....	126

LIST OF TABLES

Table 1:List Of Abbreviations	xi
Table 2:Research objectives of the Product Demand Prediction component	9
Table 3:Scope, limitations and assumptions of the Product Demand Prediction research	12
Table 4:Individual contribution to the Smart-Cost Product Demand Prediction component.....	14
Table 5:Summary of related studies on product demand forecasting	22
Table 6:Comparison of statistical, machine learning and hybrid forecasting approaches	24
Table 7:Comparison of existing demand forecasting techniques	26
Table 8:Identified research gaps and proposed Smart-Cost responses	29
Table 9:Dataset attributes used for Product Demand Prediction	34
Table 10:Product categories included in the experimental dataset	35
Table 11:Data preprocessing and transformation activities	36
Table 12:Prophet model configuration and parameters	38
Table 13:MLP model configuration and parameters	38
Table 14:Experimental training and testing configuration.....	40
Table 15:Forecast evaluation metrics.....	42
Table 16:Functional requirements of the Product Demand Prediction component ...	49
Table 17:Non-functional requirements of the Product Demand Prediction component	52
Table 18:Comparison of forecasting model design alternatives	54
Table 19:Product Demand Prediction API endpoints	58
Table 20:Input and output specifications of the forecasting component	58
Table 21:Technology selection and design rationale	61
Table 22:Development environment and technology stack	65
Table 23:Implementation details of the Prophet forecasting model	68
Table 24:Implementation details of the MLP forecasting model	69
Table 25:Integration points between the Smart-Cost backend and ML service	75
Table 26:Product Demand Prediction API test cases.....	76

Table 27:Functional verification test cases and results.....	77
Table 28:Integration issues identified and corrective actions	79
Table 29:Preliminary MLP forecasting performance for Bread	84
Table 30:Seven-day actual and predicted demand values for the MLP model.....	85
Table 31:Seven-day Prophet forecast results	86
Table 32:Comparison of Prophet and MLP forecasting performance	87
Table 33:Hybrid forecasting performance results.....	88
Table 34:Hybrid forecasting performance results.....	88
Table 35:Overall forecasting model performance comparison.....	89
Table 36:Summary of findings against the research objectives.....	95
Table 37:Summary of Research Objectives, Achieved Outcomes, and Remaining Work.....	101
Table 38:Current Limitations and Proposed Future Improvements of the Product Demand Prediction System.....	104
Table 39:Structure and Description of the Product Demand Prediction Dataset.....	117
Table 40:Seven-Day Sliding-Window Input and Prediction Target Structure for the MLP Forecasting Model	118
Table 41:Prophet Forecasting Model Hyperparameter Configuration.....	118
Table 42:MLP Forecasting Model Architecture and Hyperparameter Configuration	119
Table 43:Preliminary MLP Forecasting Model Evaluation Results for Bread	120
Table 44:Technology Stack and Functional Layers of the Smart-Cost Product Demand Prediction System	121
Table 45:Functional Test Cases and Expected Results for the Product Demand Prediction System	127
Table 46:Summary of AI-Assisted Activities and Verification Procedures in Thesis Preparation	128

LIST OF ABBREVIATIONS

Table 1:List Of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
API	Application Programming Interface
CSV	Comma-Separated Values
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
POS	Point of Sale
ReLU	Rectified Linear Unit
REST	Representational State Transfer
RMSE	Root Mean Squared Error
SME	Small and Medium-sized Enterprise
UI	User Interface
URL	Uniform Resource Locator

1 INTRODUCTION

Accurate estimation of future product demand is an important requirement for businesses that must make purchasing, production, and inventory decisions before actual customer demand becomes known. This challenge is particularly significant for micro and small enterprises because they often operate with limited working capital, restricted storage capacity, and relatively small inventory buffers. An inaccurate estimation of demand can therefore affect both product availability and operating costs. Retail forecasting research has consequently received considerable attention, with forecasting methods increasingly being incorporated into operational decision-making processes [1].

Point-of-Sale (POS) systems generate valuable historical information through everyday business transactions. Each completed sale can provide details such as the product, quantity, transaction date, and sales value. When these records are systematically organized, they provide a historical representation of product demand. However, conventional sales records primarily describe past events. The additional challenge is to transform historical transactions into predictive information that can assist businesses in estimating what may occur in future periods.

Demand forecasting is complex because product sales do not necessarily remain constant over time. Products may exhibit trend, weekly variation, seasonal behaviour, irregular demand, and nonlinear relationships. Different products may also demonstrate substantially different demand characteristics [18]. Consequently, applying one forecasting mechanism to every product without considering the characteristics of the underlying sales series may restrict forecasting effectiveness. Research has therefore explored statistical forecasting, machine learning, artificial neural networks, and hybrid forecasting approaches for retail and supply-chain demand prediction [5], [7], [9].

This research investigates Product Demand Prediction within the Smart-Cost platform. The proposed component transforms historical POS transactions into product-level daily demand series and applies forecasting algorithms to estimate future quantities. Two complementary forecasting approaches are investigated. The first is

Prophet, which was introduced for practical business time-series forecasting and provides mechanisms for modelling temporal patterns [2]. The second is a Multilayer Perceptron (MLP) neural network, which learns nonlinear relationships from recent historical demand observations. Neural-network forecasting has previously demonstrated potential for modelling complex sales relationships [3], [5].

The research further investigates a Hybrid Demand Prediction approach using the complementary forecasting characteristics of Prophet and MLP. This direction is supported by previous studies demonstrating the potential of hybrid and ensemble strategies in demand forecasting [4], [9]. In addition to model development, the research addresses data preprocessing, quantitative model evaluation, backend integration, and delivery of forecasts through Smart-Cost . Therefore, the study treats Product Demand Prediction as an integrated forecasting capability rather than an isolated machine-learning experiment.

1.1 Overall project context

Smart-Cost is a digital business-management platform designed to support micro and small enterprises by combining operational transaction management with data-driven analytical capabilities. Businesses using digital POS applications continuously generate transaction information during normal sales activities. Although this information is initially collected for operational purposes such as billing and sales recording, historical POS data can also provide a valuable basis for understanding product-demand behaviour and supporting future business decisions.

Retail businesses regularly need to determine how much inventory should be purchased, maintained, or prepared for future sales periods. This decision is difficult because future customer demand is unknown at the time inventory decisions are made. Retail forecasting research has shown that forecasting is affected by several factors, including the level of data aggregation, product characteristics, temporal variation, promotions, and the operational context in which forecasts are used [1]. Product-demand forecasting is therefore particularly valuable when historical transaction information can be transformed into actionable predictions.

Within Smart-Cost , historical POS transactions provide the data foundation for this capability. Sales records associated with individual products can be aggregated according to date to determine the total quantity sold during each day. This creates a chronological representation of product demand that can be analysed using forecasting algorithms. The resulting predictions can then be made available through the Smart-Cost environment to assist with inventory and operational planning.

The Smart-Cost platform has progressed beyond an experimental software prototype. The system is currently available through mobile application distribution platforms, including the Google Play Store and Apple App Store, while a web-based version has also been developed. The platform has established a customer base exceeding 30 customers. This deployment context is important to the present research because Product Demand Prediction is intended to operate as part of a real business-management ecosystem rather than solely as an independent forecasting demonstration. Within this environment, Product Demand Prediction extends the role of historical POS data from retrospective reporting toward predictive decision support. By converting accumulated sales records into future demand estimates, Smart-Cost can provide businesses with information that supports more informed product planning.

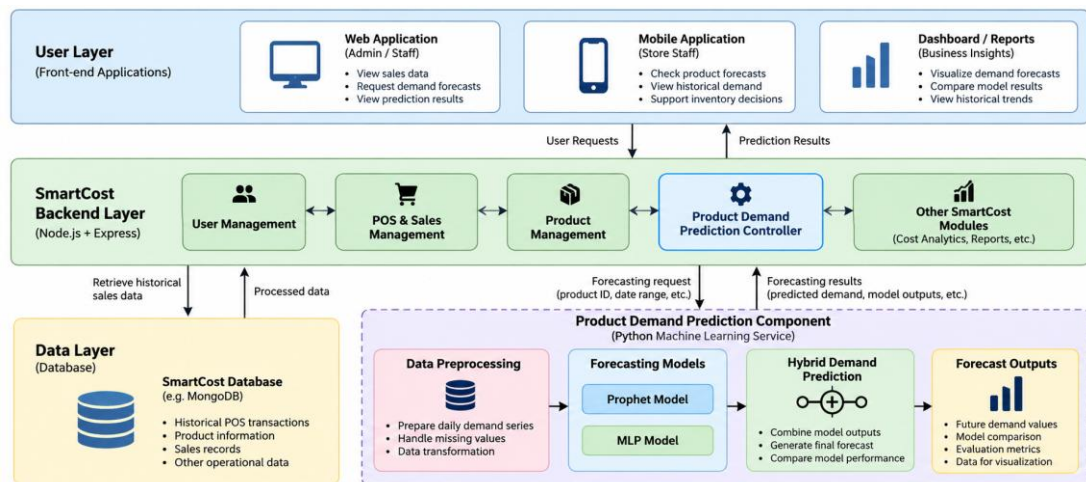


Figure 1: Position of the Product Demand Prediction component within the SmartCost solution

1.2 Individual component context

The individual research component addressed in this thesis is Product Demand Prediction. The purpose of the component is to use historical product-sales information captured through Smart-Cost to estimate future demand at individual-product level. The component is designed to move beyond conventional historical reporting by converting past sales quantities into forecasts that can assist users when making inventory and product-planning decisions. The forecasting process begins with POS transaction information. Since multiple sales transactions for the same product may occur on a single day, raw transaction records are aggregated according to product and date. This produces a chronological daily-demand series in which each observation represents the total quantity of a specific product sold on a particular day. Maintaining chronological order is important because forecasting depends on temporal relationships between historical observations.

Two forecasting approaches were developed for the component. The first is Prophet, a forecasting method developed for business time-series applications [2]. Prophet is suitable for investigating temporal patterns such as trend and recurring seasonality within historical product demand. The second approach is an MLP neural network. Artificial neural networks are capable of modelling nonlinear relationships and have been widely investigated for forecasting applications [5]. The implemented MLP uses a seven-day sliding window, where recent daily-demand observations are used to estimate subsequent product demand.

The individual component additionally investigates a Hybrid Demand Prediction approach. The motivation for this approach is that Prophet and MLP represent historical demand differently. Prophet focuses on temporal structure and seasonality, whereas the MLP learns nonlinear relationships from recent demand observations. Previous demand-forecasting research has demonstrated the potential value of combining multiple predictive techniques rather than depending entirely on a single model [4], [9].

The component also includes preprocessing, model training, prediction generation, model evaluation, and software integration. Forecasting performance is assessed using error measures including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The resulting forecasting workflow is integrated with Smart-Cost so that predictive information can be accessed within the operational application environment.

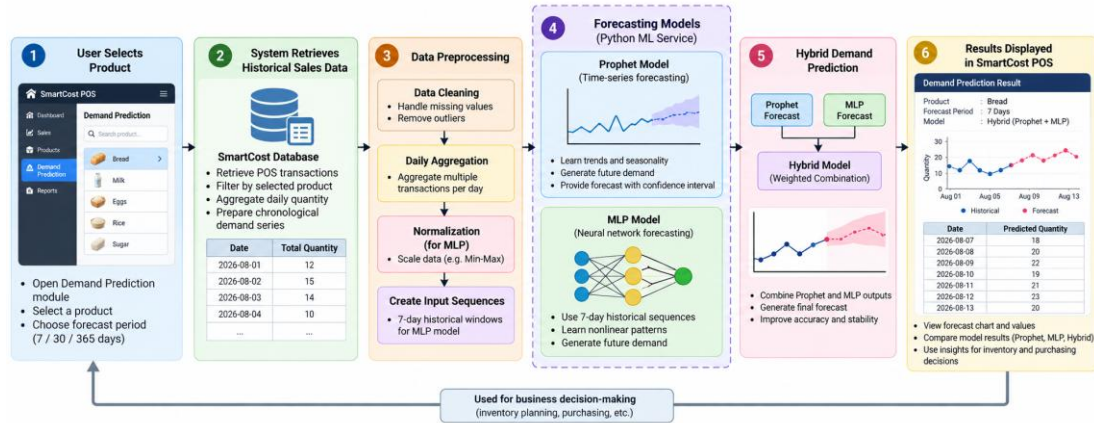


Figure 2: Product Demand Prediction workflow within the SmartCost POS platform

1.3 Problem statement

Micro and small enterprises must frequently determine future product requirements without knowing the exact quantity that customers will purchase. These decisions may involve inventory replenishment, ingredient purchasing, product preparation, or stock allocation. When such decisions are primarily based on personal judgement or simple observation of recent sales, changes in demand patterns may not be systematically identified.

Underestimating future demand can result in insufficient stock and missed sales opportunities. Conversely, overestimating demand can create excess inventory, increase holding costs, and potentially contribute to wastage. These consequences are particularly important for short shelf-life products. Research by Doganis *et al.* investigated forecasting specifically for short shelf-life food products, illustrating the operational importance of reliable sales estimation in environments where excess inventory may rapidly lose value [8].

Historical POS transactions contain information that can support more systematic forecasting. However, raw transaction records do not directly provide future-demand estimates. Transactions must first be filtered by product, aggregated according to time, cleaned, chronologically organized, and transformed into a representation suitable for forecasting. An appropriate predictive model must then learn useful relationships from the resulting historical observations.

A further challenge is that product demand does not necessarily follow one universal pattern. Syntetos, Boylan, and Croston demonstrated the importance of recognizing differences between demand patterns [18]. A product exhibiting strong recurring seasonality may require different predictive characteristics from a product whose demand is influenced predominantly by recent changes. Therefore, reliance on a single forecasting mechanism may restrict adaptability across different products.

The problem addressed by this research is therefore the lack of an integrated mechanism within Smart-Cost for systematically transforming historical POS transactions into product-specific future demand predictions using complementary forecasting techniques. The proposed solution must process historical sales, generate reliable forecasts, quantitatively evaluate prediction errors, and deliver the resulting information through the Smart-Cost application environment. Addressing this problem can transform existing transactional data into actionable information for inventory and operational decision support.

1.4 Research gap

Existing literature demonstrates substantial progress in retail forecasting and product-demand prediction. Traditional statistical forecasting methods, machine-learning algorithms, neural networks, and hybrid techniques have all been applied to sales and supply-chain forecasting problems [1], [5], [7]. Consequently, the research gap addressed in this study is not simply the absence of forecasting algorithms. Instead, it concerns their selection, complementary use, and operational integration for product-level forecasting within a micro-SME POS environment.

The first identified gap concerns heterogeneous product-demand behaviour. Individual products can exhibit different patterns of demand. Syntetos, Boylan, and Croston emphasized the importance of demand-pattern categorization [18], while Mejía and Aguilar investigated product categories defined according to their time-series characteristics and applied ensemble-learning techniques [4]. These studies indicate that assuming one forecasting model will perform equally well for every product may be inappropriate.

The second gap concerns the complementary characteristics of different forecasting paradigms. Prophet provides an approach designed for practical business time-series forecasting [2], while neural networks can model nonlinear relationships within historical observations [5]. Each technique approaches forecasting differently. This creates an opportunity to investigate whether their complementary characteristics can be utilized within a hybrid product-demand forecasting workflow. Previous research has demonstrated the potential of hybrid approaches in demand forecasting and supply-chain management [9].

A third gap exists between forecasting experiments and operational forecasting systems. Many forecasting studies primarily concentrate on model performance using prepared datasets. In a deployed POS environment, however, an end-to-end forecasting capability requires transaction retrieval, product-level aggregation, data preprocessing, chronological model training, forecast generation, quantitative evaluation, backend communication, and presentation of predictions to business users.

This research addresses these gaps through the development of a Product Demand Prediction framework specifically integrated with Smart-Cost . The proposed framework combines historical POS data preparation, Prophet forecasting, MLP-based nonlinear prediction, hybrid forecasting, model evaluation, and application-level integration. The intended research contribution therefore lies in developing and evaluating an integrated forecasting workflow suitable for practical product-level demand prediction rather than claiming the invention of the underlying Prophet or neural-network algorithms.

1.5 Aim and objectives

The main aim of this research is to design, develop, evaluate, and integrate a Product Demand Prediction component for Smart-Cost that uses historical POS sales data and complementary forecasting techniques to estimate future product demand for micro and small enterprises.

To achieve this aim, the research establishes several specific objectives.

The first objective is to develop a structured data-preparation process for product demand forecasting. Raw POS transaction information must be transformed into chronological daily-demand series by identifying products, processing transaction dates, aggregating quantities, handling unsuitable observations, and preparing model-specific input structures.

The second objective is to develop a Prophet-based product-demand forecasting model. Prophet was selected because of its suitability for business time-series forecasting and its ability to represent temporal structure [2]. The model is intended to identify relevant patterns within historical product demand and generate future product-level forecasts.

The third objective is to develop an MLP neural-network model for short-term product-demand prediction. The model uses a seven-day sliding window to represent recent historical demand and learns a nonlinear relationship between these observations and subsequent demand. The use of neural networks is supported by previous forecasting research demonstrating their ability to model nonlinear time-series relationships [5].

The fourth objective is to investigate a Hybrid Demand Prediction approach that utilizes information from Prophet and MLP. This objective is motivated by previous research demonstrating the potential value of hybrid and ensemble forecasting strategies [4], [9].

The fifth objective is to evaluate forecasting performance quantitatively using appropriate error measures, including MAE, MSE, and RMSE. Forecast-accuracy measurement is essential for objectively comparing predicted values with actual observations [17].

The final objective is to integrate the forecasting workflow into Smart-Cost , enabling historical product data to be retrieved, processed, forecast, evaluated, and delivered through the application environment. Together, these objectives establish a complete progression from historical POS transactions to operational product-demand decision support.

Table 2: Research objectives of the Product Demand Prediction component

Objective	Description
O1	Transform historical POS transactions into chronological product-level daily-demand series.
O2	Develop a Prophet model for product-demand forecasting using historical time-series data.
O3	Develop an MLP neural-network model using a seven-day historical demand window.
O4	Investigate a Hybrid Demand Prediction approach using Prophet and MLP outputs.
O5	Evaluate forecast performance using MAE, MSE and RMSE.
O6	Integrate the forecasting workflow with the Smart-Cost backend and user interface.

1.6 Research questions or hypotheses

The main aim of this research is to design, develop, evaluate, and integrate a Product Demand Prediction component for Smart-Cost that uses historical POS sales data and

complementary forecasting techniques to estimate future product demand for micro and small enterprises.

To achieve this aim, the research establishes several specific objectives.

The first objective is to develop a structured data-preparation process for product demand forecasting. Raw POS transaction information must be transformed into chronological daily-demand series by identifying products, processing transaction dates, aggregating quantities, handling unsuitable observations, and preparing model-specific input structures.

The second objective is to develop a Prophet-based product-demand forecasting model. Prophet was selected because of its suitability for business time-series forecasting and its ability to represent temporal structure [2]. The model is intended to identify relevant patterns within historical product demand and generate future product-level forecasts.

The third objective is to develop an MLP neural-network model for short-term product-demand prediction. The model uses a seven-day sliding window to represent recent historical demand and learns a nonlinear relationship between these observations and subsequent demand. The use of neural networks is supported by previous forecasting research demonstrating their ability to model nonlinear time-series relationships [5].

The fourth objective is to investigate a Hybrid Demand Prediction approach that utilizes information from Prophet and MLP. This objective is motivated by previous research demonstrating the potential value of hybrid and ensemble forecasting strategies [4], [9].

The fifth objective is to evaluate forecasting performance quantitatively using appropriate error measures, including MAE, MSE, and RMSE. Forecast-accuracy measurement is essential for objectively comparing predicted values with actual observations [17].

The final objective is to integrate the forecasting workflow into Smart-Cost , enabling historical product data to be retrieved, processed, forecast, evaluated, and delivered through the application environment. Together, these objectives establish a complete progression from historical POS transactions to operational product-demand decision support.

1.7 Scope limitations and assumptions

The scope of this research is limited to Product Demand Prediction within Smart-Cost . The study focuses on converting historical POS sales information into product-level daily-demand series and using those observations to estimate future product quantities. Other analytical components of Smart-Cost are outside the scope of this thesis except where technical integration is necessary for the Product Demand Prediction workflow.

The modelling scope includes three principal elements: Prophet forecasting, MLP neural-network forecasting, and Hybrid Demand Prediction. Prophet is used to investigate temporal and seasonal characteristics [2], while the MLP is used to learn nonlinear relationships from recent demand observations [5]. The MLP uses a seven-day historical sliding window. Model performance is assessed using MAE, MSE, and RMSE.

Several limitations apply to the research. A major limitation concerns data availability. Synthetic POS-style data has been used during development and experimentation where sufficient real-world historical sales records were unavailable. Although synthetic data enables controlled development and technical validation, it cannot reproduce every characteristic of genuine customer-demand behaviour. Consequently, final conclusions must distinguish results obtained from synthetic data from results obtained through actual business transactions.

The forecasting models are also primarily based on historical quantity information. External variables such as weather, promotions, public holidays, competitor behaviour, temporary stock-outs, price changes, and local events may influence demand but are not comprehensively represented in the current implementation.

The research assumes that POS transactions contain sufficiently accurate product identifiers, dates, and quantities and that enough chronological observations exist for meaningful model training. It is also assumed that historical sales provide useful information about future behavior.

Finally, predicted quantities are treated as decision-support estimates rather than guaranteed future sales. Forecast uncertainty cannot be eliminated entirely. The predictions are intended to support business judgement and inventory planning rather than replace managerial decision-making.

Table 3: Scope, limitations and assumptions of the Product Demand Prediction research

Category	Item	Description
Scope	Forecasting component	Product-level demand prediction using historical Smart-Cost POS sales data.
Scope	Models	Prophet, MLP and a Hybrid Demand Prediction approach.
Scope	Evaluation	Quantitative evaluation using MAE, MSE and RMSE.
Limitation	Data availability	Synthetic POS-style data was used where sufficient long-term real-world history was unavailable.
Limitation	External variables	Weather, promotions, holidays, price changes and stock-outs are not comprehensively represented.
Limitation	Preliminary evaluation	Recorded Bread MLP metrics are based on a limited seven-day evaluation period.
Assumption	Data quality	Product identifiers, dates and sold quantities are sufficiently accurate for forecasting.

Category	Item	Description
Assumption	Historical relevance	Past sales contain useful information for estimating future product demand.
Assumption	Decision support	Forecasts support business judgement and are not guaranteed future sales.

1.8 Individual contribution

The individual contribution of this research is the design, development, evaluation, and system integration of the Smart-Cost Product Demand Prediction component. The contribution covers the complete forecasting workflow from historical POS data preparation to delivery of future-demand information through the application environment.

The first contribution is the development of the product-demand data-preparation workflow. POS transaction information is processed and aggregated according to product and date to create chronological daily-demand series. Model-specific data representations are subsequently prepared to satisfy the different requirements of Prophet and the neural-network model.

The second contribution is the implementation of the Prophet forecasting model. The work includes preparing Prophet-compatible time-series data, configuring relevant seasonal behaviour, training product-specific forecasting models, and generating future-demand estimates.

The third contribution is the development of the MLP neural-network forecasting model. The implementation uses a seven-day sliding-window structure and Min-Max normalization. The MLP contains two hidden layers with 64 and 32 neurons and learns a nonlinear mapping from recent historical demand to subsequent demand. Recursive forecasting is used when future estimates are required beyond a single prediction step.

The fourth contribution is the investigation and implementation of a Hybrid Demand Prediction approach using Prophet and MLP forecasting outputs. The purpose of this work is to utilize complementary forecasting characteristics rather than depending exclusively on one model.

The fifth contribution is quantitative model evaluation, including the use of MAE, MSE, and RMSE to measure prediction errors and support comparative analysis.

Finally, the forecasting workflow was integrated into the Smart-Cost technical environment through the Python machine-learning service and backend communication mechanisms. This integration enables historical data retrieval, model execution, prediction generation, and delivery of forecasting information to the Smart-Cost user interface. Therefore, the individual contribution includes both the research-oriented forecasting work and the engineering required to transform it into an operational system capability.

Table 4: Individual contribution to the Smart-Cost Product Demand Prediction component

Contribution	Individual work
Demand-data preparation	Designed the process for filtering, aggregating and chronologically preparing product-level daily demand.
Prophet forecasting	Configured, trained and generated product-specific Prophet forecasts.
MLP forecasting	Implemented seven-day sliding-window preparation, scaling, MLP training and recursive prediction.
Hybrid forecasting	Designed the workflow that brings Prophet and MLP outputs into the Hybrid Demand Prediction stage.
Evaluation	Applied MAE, MSE and RMSE to quantify forecasting error.

Contribution	Individual work
Integration	Connected the forecasting workflow to Smart-Cost through the Python ML service and backend communication.
Testing and analysis	Verified preprocessing, model execution, APIs and integration issues and interpreted the resulting evidence.

1.9 Thesis structure

This thesis is organized into a sequence of chapters that progressively explain the research foundation, methodology, technical implementation, evaluation, and conclusions associated with the Product Demand Prediction component.

Chapter 1 – Introduction establishes the background and context of Product Demand Prediction within Smart-Cost . It describes the individual component, defines the research problem and gap, establishes the research aim and objectives, presents the research questions, identifies the scope and limitations, and explains the individual contribution.

Chapter 2 – Literature Review and Related Work critically examines previous research relevant to retail demand forecasting, product-level forecasting, time-series analysis, Prophet, artificial neural networks, machine-learning forecasting, hybrid forecasting, and forecast evaluation. The chapter compares previous approaches and establishes the theoretical foundation and research gap addressed by the proposed component.

Chapter 3 – Research Methodology explains how the research was conducted. It describes the research design, dataset preparation, data preprocessing, chronological data handling, experimental procedure, model-training strategy, forecasting process, evaluation metrics, and methods used to compare the forecasting approaches.

Chapter 4 – System Design and Implementation presents the technical realization of the Product Demand Prediction component. It explains the overall architecture, data-processing workflow, Prophet implementation, MLP architecture, Hybrid Demand Prediction mechanism, machine-learning service, backend communication, data management, API integration, and user-interface functionality.

Chapter 5 – Results and Discussion presents and analyses the experimental findings. It reports model-evaluation results, compares forecasting behaviour, discusses practical implications, evaluates the research questions and objectives, and identifies limitations affecting interpretation and generalization of the findings.

Chapter 6 – Conclusion and Future Work summarizes the research outcomes and individual contribution. It concludes the Product Demand Prediction investigation and identifies future improvements, including validation using larger real-world Smart-Cost datasets, incorporation of additional demand-influencing variables, enhanced hybrid forecasting strategies, and continued model optimization.

The thesis concludes with the IEEE-formatted reference list and supporting appendices, which provide additional technical and experimental evidence relevant to the Product Demand Prediction research.

2 LITERATURE REVIEW AND RELATED WORK

Product demand prediction is an important application of forecasting in retail and small-business environments because future demand directly affects purchasing, inventory availability, production planning, and resource utilization. Retail forecasting is challenging because sales behaviour differs among products and can be affected by temporal patterns, seasonality, special events, promotions, and changing customer behaviour. Fildes, Ma, and Kolassa identify retail forecasting as a complex area in which forecasting methods must account for characteristics such as product heterogeneity, data granularity, promotions, and operational requirements [1]. For micro and small enterprises, these challenges can be particularly important because excess inventory or insufficient stock can have a direct effect on operating costs and sales opportunities.

Historical Point-of-Sale (POS) transactions provide a useful data source for product-level demand prediction. Individual transactions can be aggregated according to product and date to produce chronological demand series. These time series can then be analysed to identify recurring patterns and estimate future quantities. Research has demonstrated the applicability of statistical, machine-learning, and neural-network approaches to sales and demand forecasting. Carbonneau, Laframboise, and Vahidov investigated machine-learning techniques for supply-chain demand forecasting and demonstrated the potential of computational learning approaches for modelling demand behaviour [7]. Neural networks have also been extensively investigated for forecasting because of their ability to approximate nonlinear relationships [5].

This research focuses specifically on Product Demand Prediction within Smart-Cost . Two complementary forecasting techniques are investigated. The first is Prophet, introduced by Taylor and Letham as a forecasting approach designed for practical business time-series applications [2]. The second is a Multilayer Perceptron (MLP) neural network that learns nonlinear relationships from recent historical product-demand observations. The use of neural-network techniques for retail and sales forecasting is supported by previous studies demonstrating their ability to model complex sales behaviour [3], [5].

The research further investigates a hybrid forecasting strategy. Hybrid and ensemble approaches are relevant because a single forecasting algorithm may not perform equally well across products with different demand characteristics. Aburto and Weber demonstrated the potential of hybrid demand forecasting for supply-chain management [9], while Mejía and Aguilar investigated an ensemble-based demand forecasting framework in which product time-series characteristics influenced forecasting decisions [4]. Therefore, the literature provides a foundation for investigating whether Prophet and MLP can provide complementary information for product-level forecasting within Smart-Cost .

2.1 Review method and scope

The literature review was conducted with a focused scope aligned with the objectives of the Product Demand Prediction component. The review concentrated on academic work concerning product and retail demand forecasting, time-series forecasting, artificial neural networks, machine-learning-based demand prediction, Prophet, hybrid forecasting, and forecast-accuracy evaluation. Literature related to other Smart-Cost research components was excluded because it does not directly contribute to the individual research problem investigated in this thesis.

The review considered peer-reviewed journal articles, established forecasting studies, and research describing forecasting methods applicable to chronological sales data. Particular attention was given to studies involving retail products, daily demand, food products, and supply-chain forecasting because these areas share characteristics with the intended Smart-Cost application environment. For example, Doganis *et al.* investigated sales forecasting for short shelf-life food products using artificial neural networks [8]. This is relevant to businesses where inaccurate forecasting can contribute to either insufficient product availability or unnecessary quantities of perishable inventory.

The reviewed literature was organized into several themes. The first theme examined the characteristics and challenges of retail demand forecasting. Fildes, Ma, and Kolassa provide a broad examination of research and practice in retail forecasting and

identify the complexity associated with producing operational forecasts across retail environments [1]. The second theme considered machine-learning and neural-network approaches. Zhang, Patuwo, and Hu reviewed artificial neural networks for forecasting and discussed their ability to model nonlinear relationships [5], while Ahmed *et al.* empirically compared machine-learning techniques for time-series forecasting [6].

The third theme examined Prophet and practical business time-series forecasting, for which Taylor and Letham's work provides the principal methodological reference [2]. The fourth investigated hybrid and ensemble forecasting. Studies by Aburto and Weber [9] and Mejía and Aguilar [4] demonstrate how multiple forecasting approaches can be incorporated to address complex demand behaviour.

Finally, literature on forecast evaluation was considered because model selection requires objective performance measurement. Hyndman and Koehler examined measures of forecast accuracy and highlighted important considerations when selecting error metrics [17]. These themes collectively informed the selection, implementation, and evaluation of the forecasting techniques used in this research.

2.2 Domain and theoretical foundations

Product demand prediction involves estimating future product requirements using information available from historical observations. Within the Smart-Cost environment, the principal source of this information is POS sales data. Multiple transactions involving the same product can occur within a day; therefore, transaction quantities are aggregated by product and date to establish a chronological daily-demand series. The resulting observations provide the foundation for model training and future forecasting.

A key theoretical concept underlying the research is time-series forecasting. Unlike datasets in which observations can be treated independently, time-series observations possess temporal ordering. Previous demand can contain information relevant to subsequent demand, while patterns such as trend and seasonality may occur over longer periods. Retail forecasting research emphasizes that demand series can contain complex characteristics that differ considerably between products [1]. Syntetos,

Boylan, and Croston similarly demonstrated the importance of recognizing different demand patterns when analysing inventory-related demand series [18].

The first forecasting foundation adopted in this research is Prophet. Taylor and Letham introduced Prophet as an approach to forecasting business time series that supports interpretable modelling of temporal structure [2]. This characteristic makes Prophet relevant to product demand where recurring weekly, monthly, or yearly patterns may exist. In Smart-Cost, Prophet is configured to learn temporal patterns from daily product-demand observations and generate future demand estimates.

The second foundation is artificial neural-network forecasting. Neural networks can approximate nonlinear mappings between input and output variables, making them suitable for relationships that cannot be adequately represented by simple linear assumptions. Zhang *et al.* provide an extensive foundation for neural-network forecasting [5], while Alon, Qi, and Sadowski demonstrated the applicability of artificial neural networks to retail sales forecasting [3]. The Smart-Cost implementation applies an MLP neural network to seven consecutive historical demand observations to predict subsequent demand.

A third theoretical foundation is hybrid forecasting. Different models may capture different properties of a demand series. Hybrid approaches attempt to exploit complementary predictive characteristics rather than relying entirely on one forecasting mechanism [9]. This provides the theoretical basis for investigating the combination of Prophet's temporal modelling capabilities with the nonlinear learning capability of the MLP within the proposed Product Demand Prediction component.

2.3 Related research

Previous research provides substantial evidence for the use of statistical and machine-learning methods in retail demand prediction. Alon, Qi, and Sadowski compared artificial neural networks with traditional methods for forecasting aggregate retail sales. Their work demonstrated that neural networks can be effective for modelling nonlinear trend and seasonal relationships in retail sales information [3]. Although

their research focused on aggregate sales, the findings provide useful evidence for investigating neural-network techniques at individual-product level.

Huber and Stuckenschmidt investigated daily retail demand forecasting using machine-learning methods, with particular emphasis on calendar-related special days [3]. Their research used bakery-chain demand and demonstrated that daily demand can be strongly influenced by calendar conditions. This work is particularly relevant to Product Demand Prediction in Smart-Cost because businesses dealing with food and other frequently purchased products may experience considerable variation between ordinary weekdays, weekends, and special periods.

Doganis *et al.* investigated short shelf-life food-product sales forecasting using artificial neural networks and evolutionary computation [8]. Their study is relevant because forecasting errors for perishable products have operational consequences: excessive quantities can contribute to wastage, while insufficient quantities can limit product availability.

Research has also investigated the value of combining forecasting approaches. Aburto and Weber proposed hybrid demand forecasting for supply-chain management and demonstrated how computational forecasting approaches could be combined to improve demand-related decision-making [9]. More recently, Mejía and Aguilar developed a demand-forecasting framework based on product time-series characteristics, feature engineering, multiple forecasting algorithms, and ensemble learning [4]. Their findings are important to this research because they demonstrate that different products may possess different demand characteristics and therefore may not be optimally represented by one universal forecasting algorithm.

Table 5: Summary of related studies on product demand forecasting

Ref.	Study / focus	Method or contribution	Relevance to this research
[1]	Fildes, Ma and Kolassa - retail forecasting	Review of retail forecasting research and practice	Establishes challenges such as product heterogeneity, granularity and operational use.
[2]	Taylor and Letham - business time-series forecasting	Prophet	Provides the methodological basis for the Prophet forecasting path.
[3]	Huber and Stuckenschmidt - daily retail demand	Machine learning with calendar/special-day effects	Shows the importance of temporal and calendar effects in daily retail demand.
[5]	Zhang, Patuwo and Hu - neural-network forecasting	Artificial neural networks	Supports use of neural networks for nonlinear forecasting relationships.
[7]	Carbonneau, Laframboise and Vahidov - supply-chain demand	Machine-learning forecasting	Supports application of ML techniques to demand-estimation problems.

Ref.	Study / focus	Method or contribution	Relevance to this research
[8]	Doganis et al. - short shelf-life food sales	Artificial neural networks and evolutionary computing	Relevant to bakery/food products where forecast errors may create wastage or shortages.
[9]	Aburto and Weber - supply-chain forecasting	Hybrid demand forecasting	Supports investigation of complementary forecasting approaches.
[4]	Mejía and Aguilar - product-category demand	Feature engineering, multiple algorithms and ensemble learning	Supports product-specific and ensemble forecasting when time-series characteristics differ.

Table 6: Comparison of statistical, machine learning and hybrid forecasting approaches

Approach	Strengths	Limitations	Role in Smart-Cost
Statistical / time-series	Interpretable temporal modelling; suitable for trend and seasonality.	May not capture all nonlinear relationships.	Represented by Prophet for temporal and seasonal demand behaviour.
Machine learning / neural network	Can learn nonlinear mappings from historical observations.	Requires preprocessing, tuning and adequate training data.	Represented by MLP using seven-day demand windows.
Hybrid / ensemble	Can utilize complementary information from multiple models.	Requires fair comparative evaluation and a defined combination mechanism.	Combines or uses Prophet and MLP outputs within Hybrid Demand Prediction.

Deep-learning research provides further evidence for combining forecasting architectures. Joseph *et al.* investigated a hybrid CNN and bidirectional LSTM framework for store-item demand forecasting [10]. While the Smart-Cost implementation does not use the same deep-learning architecture, this work demonstrates continuing research interest in hybrid methods for product-demand forecasting. These studies collectively support the investigation of complementary Prophet and MLP forecasting approaches within Smart-Cost .

2.4 Existing systems products and techniques

Existing product-demand forecasting techniques range from conventional statistical models to machine-learning, neural-network, and hybrid forecasting methods. Conventional approaches such as moving averages, exponential smoothing, and autoregressive models remain important forecasting techniques because they can model temporal information using relatively interpretable mathematical structures. However, demand characteristics differ across products, and consequently no single forecasting method can automatically be considered optimal for every product or forecasting horizon [1], [15].

Prophet represents one technique designed for practical business time-series forecasting. Taylor and Letham developed the approach to support forecasting at scale while allowing analysts to configure model components according to characteristics of the underlying time series [2]. Prophet is therefore suitable for investigating product sales containing recurring temporal behaviour. Within the Smart-Cost Product Demand Prediction component, Prophet is used to learn historical demand patterns and produce product-specific future forecasts.

Machine-learning techniques provide another forecasting category. Carbonneau *et al.* investigated machine-learning approaches for supply-chain demand forecasting and demonstrated their applicability to demand-estimation problems [7]. Neural networks are particularly relevant when relationships between historical observations and future demand are nonlinear [5]. The Smart-Cost component uses an MLP neural network with hidden layers of 64 and 32 neurons. Seven previous daily-demand values are transformed into an input sequence used to estimate subsequent demand. This differs from a true recurrent LSTM architecture; therefore, the implemented model is correctly identified as an MLP, rather than an LSTM.

Another important technique is multi-step forecasting, where predictions are required for several future periods rather than only the next observation. Ben Taieb *et al.* compared strategies for multi-step-ahead time-series forecasting and demonstrated that forecasting strategy can influence performance across multiple future horizons

[14]. This is relevant to Smart-Cost because business users may require forecasts covering several future days.

Hybrid and ensemble methods constitute an additional category. Research indicates that combining models can be beneficial when different methods capture complementary demand characteristics [4], [9], [10]. These findings provide the technical motivation for the Smart-Cost Hybrid Demand Prediction approach, which investigates Prophet and MLP outputs as complementary forecasting information.

Table 7: Comparison of existing demand forecasting techniques

Technique	Typical characteristic	Suitability to Product Demand Prediction	Decision
Moving average	Simple smoothing based on recent observations.	Useful as a simple reference but limited for complex temporal behaviour.	Not selected as a principal model; suitable future baseline.
Exponential smoothing	Weights recent observations more strongly.	Useful conventional time-series approach.	Not selected as a principal model.
Autoregressive methods	Model dependence on previous observations.	Suitable for chronological series but requires model specification.	Not selected in the implemented component.
Prophet	Models business time-series trend and seasonality.	Suitable for recurring temporal patterns in product demand.	Selected.

Technique	Typical characteristic	Suitability to Product Demand Prediction	Decision
MLP	Learns nonlinear input-output relationships.	Suitable for nonlinear relationships in recent demand windows.	Selected.
LSTM	Recurrent architecture for sequential dependencies.	Potential future alternative, but not the algorithm implemented in this thesis.	Not implemented.
Hybrid / ensemble	Combines information from multiple forecasting approaches.	Relevant when products exhibit different demand characteristics.	Investigated using Prophet and MLP outputs.

2.5 Gap synthesis and conceptual framework

The reviewed literature demonstrates that product-demand forecasting is already supported by a substantial body of research. Statistical forecasting, Prophet, machine learning, neural networks, and hybrid techniques have all been investigated in different contexts [2], [5], [7], [9]. Therefore, the research gap addressed by Smart-Cost is not the absence of demand-forecasting algorithms. Instead, the gap concerns how complementary forecasting techniques can be adapted, evaluated, and operationally integrated for product-level demand prediction within a micro-SME POS environment.

One identified gap concerns product heterogeneity. Products can demonstrate different demand characteristics, making dependence on one forecasting algorithm potentially restrictive. Syntetos, Boylan, and Croston demonstrated the importance of distinguishing demand patterns [18], while Mejía and Aguilar investigated forecasting based on product time-series characteristics and ensemble learning [4]. These findings support the investigation of multiple forecasting perspectives within Smart-Cost .

A second gap concerns the transition from experimental forecasting to operational prediction. Forecasting research commonly evaluates algorithms against prepared datasets, whereas an operational POS application requires a complete pipeline. Raw sales transactions must be retrieved, filtered by product, aggregated by date, chronologically prepared, supplied to trained models, evaluated, and returned to the user through an application interface. The forecasting model is therefore only one element of the complete Product Demand Prediction capability.

Based on the identified gap, the conceptual framework for this research is:

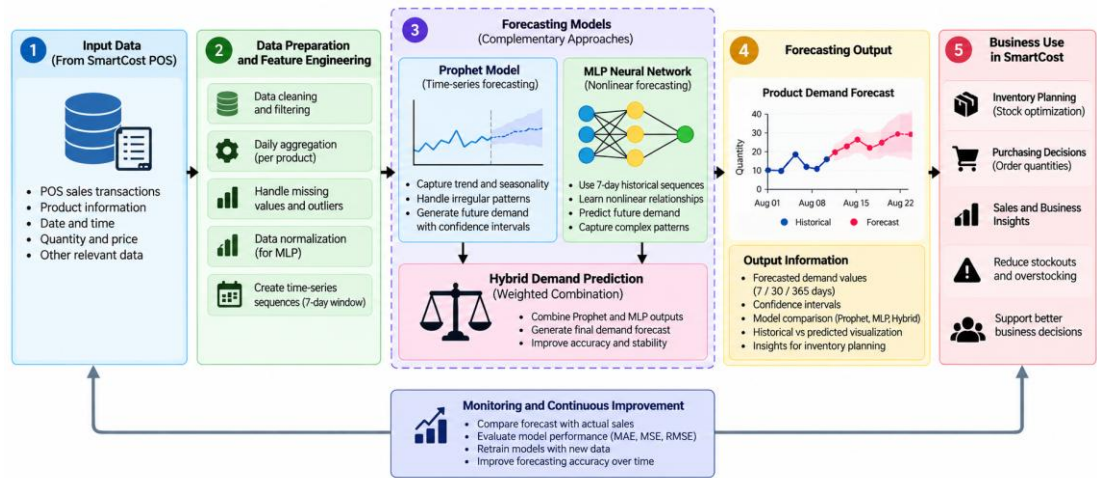


Figure 3: Conceptual framework for the Product Demand Prediction System

Prophet contributes the time-series and seasonal forecasting perspective [2], while the MLP contributes a nonlinear learning perspective supported by neural-network forecasting research [5]. The hybrid stage investigates whether these complementary capabilities can provide a more useful forecasting mechanism than dependence on a

single model. Performance is assessed using quantitative forecast-error measures, informed by established research on forecast-accuracy evaluation [17].

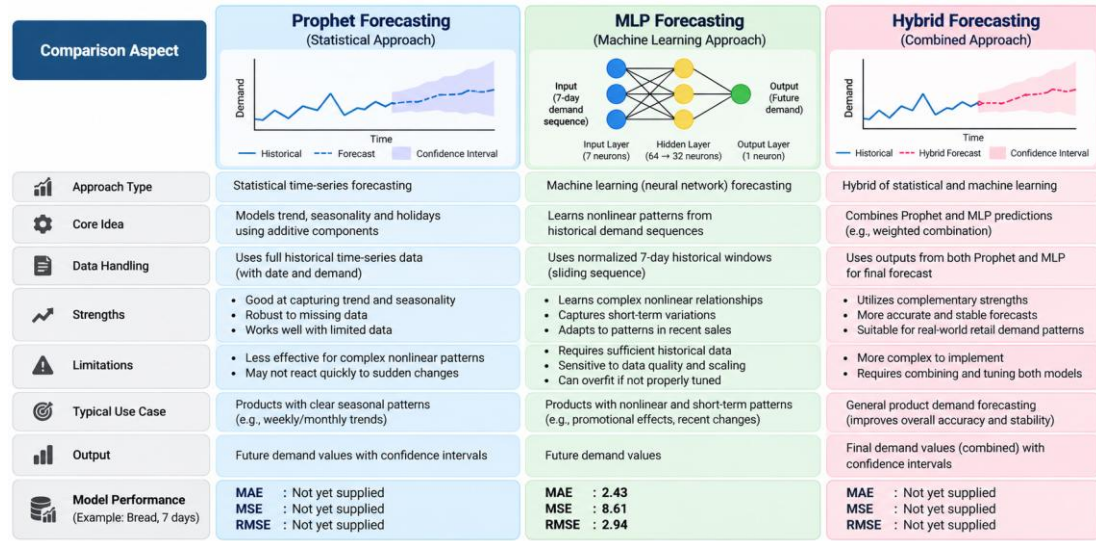


Figure 4: Comparison of individual and hybrid forecasting approaches

Accordingly, the individual research contribution is the design, implementation, evaluation, and Smart-Cost integration of this Product Demand Prediction framework, rather than the invention of Prophet or neural-network algorithms themselves.

Table 8: Identified research gaps and proposed Smart-Cost responses

Research gap	Evidence / issue	Smart-Cost response
Heterogeneous product demand	Products can exhibit different demand patterns and one model may not suit every series [4], [18].	Use complementary Prophet and MLP forecasting paths and investigate hybrid forecasting.
Complementary model characteristics	Time-series and neural approaches represent historical demand differently [2], [5].	Use Prophet for temporal structure and MLP for nonlinear recent-demand relationships.

Research gap	Evidence / issue	Smart-Cost response
Experiment-to-operation gap	Operational forecasting requires more than a trained algorithm.	Integrate retrieval, aggregation, preprocessing, forecasting, evaluation, APIs and user presentation.
Need for objective evaluation	Forecast quality cannot be established by visual inspection alone [17].	Use MAE, MSE and RMSE and compare models on equivalent held-out periods.

3 RESEARCH METHODOLOGY

This research adopted an applied experimental methodology to design, implement, and evaluate the Product Demand Prediction component of Smart-Cost . The methodological process was structured around converting historical Point-of-Sale (POS) transactions into product-level demand forecasts that could be used within an operational business application. The research therefore involved both machine-learning experimentation and software-system integration.

Historical sales observations were first transformed into daily demand time series by grouping transactions according to product and date. Data preprocessing was then performed to create suitable inputs for the forecasting models. Two complementary forecasting techniques were investigated: Prophet and a Multilayer Perceptron (MLP) neural network. Prophet was selected because it provides a structured approach to forecasting business time series and modelling recurring temporal behaviour [2]. The MLP was selected to investigate nonlinear relationships between recent historical demand and subsequent product demand, supported by previous research demonstrating the applicability of neural networks to forecasting problems [5].

For the MLP model, a seven-day sliding-window strategy was used. Seven consecutive daily-demand observations formed the model input, while the following day's quantity represented the prediction target. This approach transformed the time series into supervised learning samples. Multi-step forecasts were subsequently generated recursively, where predicted demand could become part of the input sequence for subsequent forecasting steps. Multi-step forecasting strategies have previously been recognized as an important methodological consideration in time-series prediction [14].

The research additionally investigated a Hybrid Demand Prediction approach to utilize information from both Prophet and MLP. The motivation for this design was supported by research showing that hybrid and ensemble approaches can provide useful alternatives when individual models capture different characteristics of demand [4], [9].

Forecasting performance was quantitatively assessed using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These measures enabled predicted quantities to be compared with actual historical observations, while established forecasting literature was considered when interpreting forecast accuracy [17].

Finally, the forecasting workflow was integrated with Smart-Cost through a Python-based machine-learning service, backend APIs, and the application's data environment. This allowed the research to evaluate not only predictive behaviour but also the practical feasibility of delivering product-demand forecasts through an operational software platform.

3.1 Research design

The study followed a quantitative, experimental, and system-development research design. The quantitative aspect involved representing historical product demand numerically and evaluating forecasting errors using measurable performance indicators. The experimental aspect involved training forecasting models using historical demand sequences, generating predictions for unseen chronological observations, and comparing predicted quantities against corresponding actual demand. The system-development aspect involved incorporating the resulting forecasting workflow into Smart-Cost .

The research design was organized into four interconnected stages: data preparation, model development, model evaluation, and system integration. During data preparation, raw POS transaction records were converted into product-level daily-demand series. This was necessary because forecasting algorithms require temporally organized observations rather than individual transaction records.

During model development, Prophet and MLP were investigated independently. Prophet provided the time-series forecasting perspective. Taylor and Letham developed Prophet as a practical forecasting method for business time series, making it relevant to historical product-sales data containing temporal patterns [2]. The MLP provided a machine-learning perspective capable of learning nonlinear relationships.

Neural-network forecasting research has established the usefulness of such models for representing complex relationships within historical data [5].

The MLP input structure was based on a seven-day lookback window. For each valid sequence, demand from the previous seven days was used to predict the next observation. This enabled the model to learn short-term demand relationships while preserving chronological order.

The third stage evaluated model behaviour using common regression error measures. Comparisons were made using the same chronological demand context wherever possible to prevent misleading conclusions caused by different test periods.

The final stage concerned practical integration. Trained models were exposed through the Product Demand Prediction service and connected to Smart-Cost 's backend and application interfaces. Consequently, the research design evaluated the forecasting component as an end-to-end system rather than assessing an algorithm in isolation.

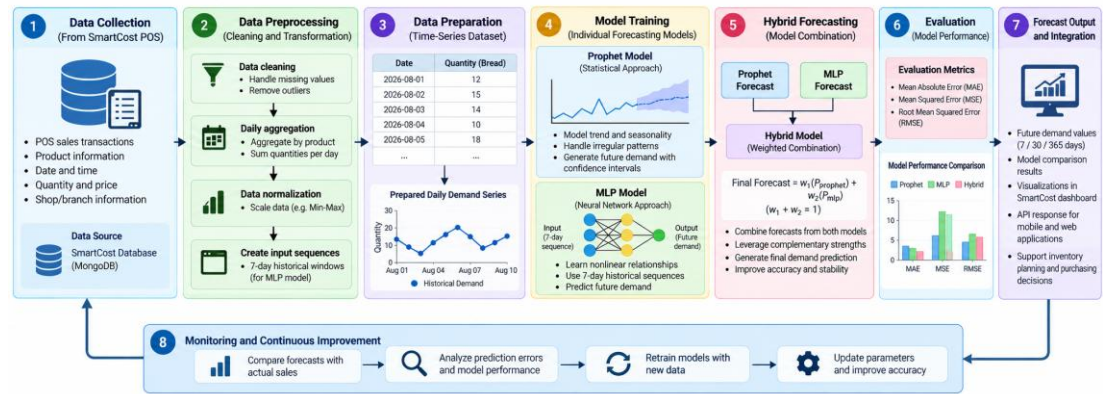


Figure 5: Research methodology pipeline for Product Demand Prediction

3.2 Data participants or study materials

The Product Demand Prediction study primarily involved transactional and time-series data rather than human participants. The principal study material consisted of product-level POS sales records containing information required to determine the quantity of a product sold on a particular date. The minimum information required by the

forecasting workflow was therefore the product identifier or product name, transaction date, and sold quantity.

During research development, a structured synthetic POS-style dataset was used where sufficient long-term real-world sales history was unavailable. The principal development dataset represented 365 days across 10 products, producing 3,650 product-day observations. Products represented common small-business items such as bread, cake, coffee, tea, buns, sandwiches, pastries, juice, muffins, and cookies. The synthetic series incorporated controlled variations such as weekly behaviour, random demand variation, and selected seasonal increases. Synthetic data was used to construct and technically validate the forecasting pipeline; it is not treated as equivalent to genuine customer-demand evidence.

Table 9:Dataset attributes used for Product Demand Prediction

Attribute	Description	Use
ds / date	Date associated with the demand observation.	Chronological index for daily product demand and Prophet input.
product_name	Name or identifier of the product being forecast.	Filters and separates individual product-demand series.
y / total_quantity	Total quantity demanded for the product on the corresponding date.	Forecasting target and principal numerical demand variable.

Different data representations were prepared according to model requirements. For Prophet, each product's observations were organized using a date variable (*ds*) and corresponding demand quantity (*y*), consistent with Prophet's required forecasting structure [2]. For the MLP, chronological demand was converted into supervised-learning sequences using seven previous observations as input and the subsequent observation as the target.

The technical study materials included Python, Prophet, NumPy, pandas, and scikit-learn for data processing, model development, and evaluation. Scikit-learn provides established machine-learning functionality suitable for implementing the MLP and preprocessing operations. The trained forecasting workflow was subsequently exposed through a Python service and connected to the Smart-Cost backend.

No demographic characteristics or personally identifiable customer attributes were required for model training. The forecasting component is concerned with aggregated product quantities over time, rather than identifying individual customers or predicting individual purchasing behaviour.

Table 10: Product categories included in the experimental dataset

No.	Product	Dataset role
1	Bread	Synthetic product-level daily-demand series used for development and technical validation.
2	Cake	Synthetic product-level daily-demand series used for development and technical validation.
3	Coffee	Synthetic product-level daily-demand series used for development and technical validation.
4	Tea	Synthetic product-level daily-demand series used for development and technical validation.
5	Bun	Synthetic product-level daily-demand series used for development and technical validation.
6	Sandwich	Synthetic product-level daily-demand series used for development and technical validation.
7	Pastry	Synthetic product-level daily-demand series used for development and technical validation.
8	Juice	Synthetic product-level daily-demand series used for development and technical validation.

No.	Product	Dataset role
9	Muffin	Synthetic product-level daily-demand series used for development and technical validation.
10	Cookie	Synthetic product-level daily-demand series used for development and technical validation.

3.3 Experimental or study procedure

The experimental procedure began by obtaining product-sales observations and transforming them into model-ready chronological time series. Transaction records were grouped according to product and date, and quantities associated with the same product on the same day were aggregated to determine daily demand. Records were then ordered chronologically to preserve the temporal structure required for forecasting.

Table 11: Data preprocessing and transformation activities

Stage	Activity	Output
1	Filter POS records for the selected product.	Product-specific transaction records.
2	Apply explicit start-date and end-date filtering.	Records within the required historical period.
3	Aggregate quantities for the same product and date.	One daily-demand quantity per date.
4	Sort observations chronologically.	Ordered time series.
5	Validate dates and numerical demand values.	Clean forecasting observations.
6	Transform to Prophet ds/y format.	Prophet-ready dataset.

Stage	Activity	Output
7	Normalize training demand and create seven-day windows.	MLP-ready supervised samples.
8	Apply training-derived preprocessing to held-out data.	Evaluation inputs with reduced leakage risk.

The second stage involved data preprocessing. Numeric demand quantities were validated and unsuitable or missing observations were handled before model training. For neural-network forecasting, demand values were normalized using a Min-Max scaling procedure. To reduce data leakage, scaling parameters should be estimated from training observations and subsequently applied to validation or test observations rather than being calculated using the complete dataset.

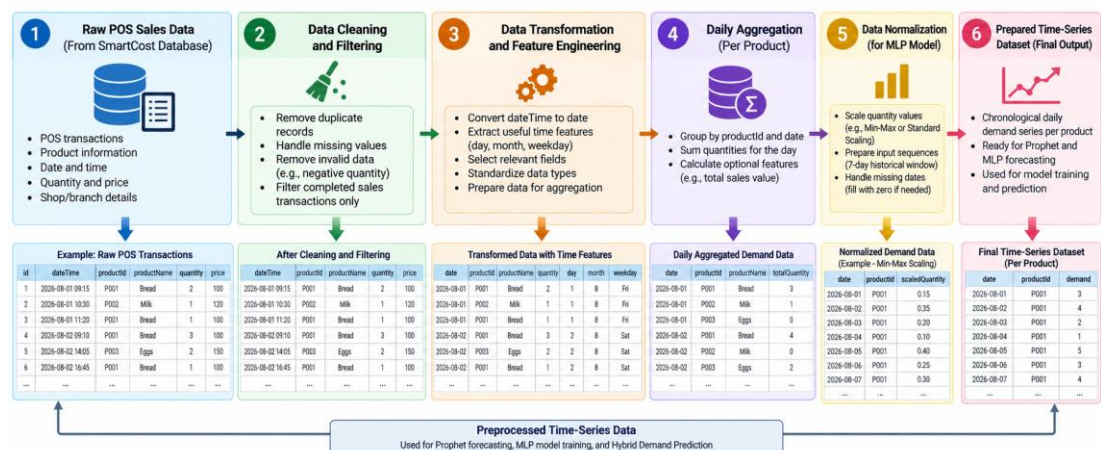


Figure 6: Historical POS sales data preprocessing and transformation workflow

The third stage involved preparing seven-day sliding windows for the MLP. For a chronological sequence $(y_1, y_2, \dots, y_n)$, the first training example used (y_1) to (y_7) as input and (y_8) as the target. The window was then moved forward by one observation to generate the next example. This procedure continued throughout the available training period.

Table 12:Prophet model configuration and parameters

Parameter	Configuration
Daily seasonality	False
Weekly seasonality	True
Yearly seasonality	True
Changepoint prior scale	0.05
Seasonality prior scale	10.0
Additional monthly seasonality	30.5-day period
Monthly Fourier order	5
Primary forecast output	yhat
Additional outputs	yhat_lower and yhat_upper

The MLP contained two hidden layers of 64 and 32 neurons, used the ReLU activation function and Adam optimization, and incorporated early stopping during training. The Prophet model was independently trained using chronological ds and y observations and configured to represent weekly and yearly seasonality, together with an additional monthly seasonal component.

Table 13:MLP model configuration and parameters

Parameter	Configuration
Model	scikit-learn MLPRegressor
Historical lookback	7 observations
Hidden layer 1	64 neurons
Hidden layer 2	32 neurons
Activation function	ReLU
Optimizer / solver	Adam

Parameter	Configuration
Maximum iterations	200
Early stopping	Enabled
Scaling	Min-Max normalization
Multi-step strategy	Recursive prediction

Future demand was then generated independently by the two models. The MLP used recursive prediction for multi-day forecasting, while Prophet generated future dates and corresponding forecast estimates. Multi-step forecasting requires careful evaluation because forecasting errors can propagate as the prediction horizon increases [14].



Figure 7:Sliding-window data preparation for the MLP forecasting model

The resulting outputs provided the basis for investigating the Hybrid Demand Prediction approach. Finally, predictions were compared with known actual observations using the selected evaluation metrics, and the complete prediction workflow was tested through the Smart-Cost integration environment.

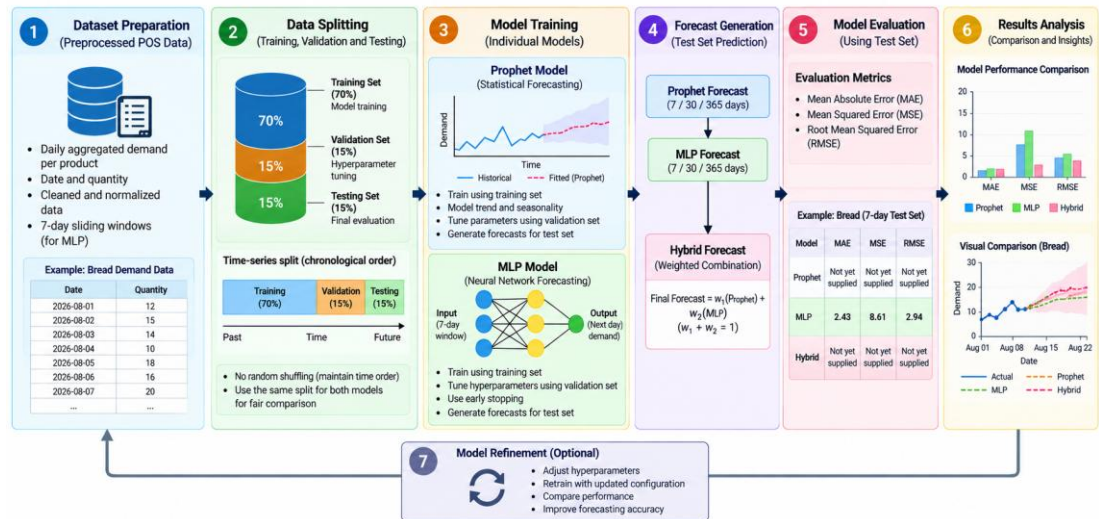


Figure 8:Experimental training and evaluation workflow

Table 14:Experimental training and testing configuration

Item	Configuration / research practice
Data ordering	Chronological; future observations are not randomly mixed into training data.
MLP sample construction	Seven previous daily-demand values are used to predict the subsequent value.
Scaling	Fit using training observations and applied to held-out observations.
Prophet training input	Chronological ds and y observations.
Forecast comparison	Models should be evaluated on the same held-out dates for direct comparison.
Recorded preliminary evaluation	Bread MLP evaluated over seven days.
Recommended extended validation	Longer chronological holdout or rolling-origin / walk-forward evaluation across multiple products.

3.4 Measures and evaluation metrics

The forecasting models were evaluated using quantitative error measures that compare predicted demand with corresponding actual demand. The principal metrics selected were Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Using multiple measures was important because each metric provides a different interpretation of forecasting error. Hyndman and Koehler emphasize the importance of selecting suitable accuracy measures when evaluating and comparing forecasting methods [17].

The Mean Absolute Error (MAE) measures the average absolute difference between actual and predicted quantities:

$$[MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|]$$

where (y_i) represents actual demand, $(\hat{y}_i)$ represents predicted demand, and (n) represents the number of evaluated observations. A lower MAE indicates that predictions are, on average, closer to actual quantities.

The Mean Squared Error (MSE) is calculated as:

$$[MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2]$$

Because errors are squared, larger prediction errors receive greater numerical penalties.

The Root Mean Squared Error (RMSE) is calculated as:

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

RMSE returns the error to the original scale of the target variable, making it easier to interpret in terms of product quantities.

Model evaluation was conducted chronologically because random shuffling can allow information from later periods to influence training for earlier periods, which would

not represent real forecasting conditions. The Prophet and MLP models should therefore be evaluated against the same held-out dates when making direct performance comparisons.

During preliminary testing for the Bread product, the MLP produced an MAE of 2.43, MSE of 8.61, and RMSE of 2.94 over the recorded seven-day evaluation period. These values are treated as preliminary experimental evidence rather than proof of general performance because a seven-day test period is limited. Broader chronological testing is required before making generalized claims regarding model superiority.

Table 15: Forecast evaluation metrics

Metric	Definition	Interpretation
MAE	Mean of the absolute differences between actual and predicted demand.	Average absolute forecast error in product units; lower is better.
MSE	Mean of squared differences between actual and predicted demand.	Penalizes larger errors more strongly; lower is better.
RMSE	Square root of MSE.	Forecast error expressed in the original demand scale; lower is better.

3.5 Data analysis

Data analysis was conducted at both the descriptive and predictive levels. Descriptive analysis was used to understand historical product-demand behaviour, while predictive analysis was used to evaluate the ability of Prophet, MLP, and the Hybrid Demand Prediction approach to estimate future quantities.

Historical sales quantities were first aggregated by product and date. The resulting daily series allowed changes in demand to be examined chronologically. Analysis over

different historical periods, including 7-day, 30-day, and longer-term views, provided contextual information about short-term fluctuations and longer-term behaviour. This stage was important because retail products may exhibit substantially different demand characteristics [18].

The predictive analysis focused on the relationship between actual and forecast demand. For each evaluated observation, the prediction error was calculated as the difference between the actual quantity and corresponding forecast. These individual errors were then summarized through MAE, MSE, and RMSE. Lower error values were interpreted as stronger predictive performance when models were evaluated using the same observations.

Comparative analysis was conducted between the Prophet and MLP approaches. Prophet was analysed according to its ability to represent temporal and seasonal behaviour, whereas MLP performance was considered in relation to nonlinear relationships derived from recent demand observations. This distinction was important because the models use different representations of the same historical information. Previous research suggests that forecasting performance can vary according to the characteristics of the underlying series [4], [18].

The hybrid analysis examined whether information from both forecasting approaches could be utilized to produce a useful final prediction. However, the hybrid approach was not automatically considered superior. Any claim of improved performance requires lower forecasting errors on equivalent test observations.

Visual analysis was also used where appropriate by plotting actual and predicted demand across time. Graphical comparisons complement numerical metrics by revealing systematic overprediction, underprediction, trend-following behaviour, or periods of unusually large errors. Together, numerical and visual analysis provided a more complete assessment of model behaviour.

3.6 Ethics privacy security and responsible AI

Ethical and responsible use of business data was considered throughout the development of the Product Demand Prediction component. Although the forecasting task does not require sensitive demographic information, POS systems can contain business and customer-related information. The forecasting pipeline was therefore designed around the principle of using only the data necessary for product-level demand estimation.

The models primarily require product identifiers, dates, and aggregated sales quantities. Customer names, telephone numbers, payment details, and other personally identifiable attributes are not required for model training. Restricting the forecasting dataset in this manner follows the principle of data minimization and reduces unnecessary privacy exposure.

Security is also important because historical sales information represents commercially valuable business data. Access to demand information and prediction functionality should therefore follow the authorization mechanisms of the Smart-Cost platform. Communication between application components should use appropriately secured interfaces in production, while databases and model services should not be unnecessarily exposed to unauthorized external access.

Responsible AI considerations are particularly important when presenting predictions to business users. Demand forecasts are probabilistic estimates derived from historical observations rather than guaranteed future outcomes. A forecast may become inaccurate when unexpected events occur or when future business conditions differ substantially from the training period. Predictions should therefore be presented as decision-support information, allowing business owners to combine model outputs with their own operational knowledge.

Another consideration concerns the use of synthetic data. Synthetic observations used during development must be explicitly identified as synthetic and must not be presented as evidence of actual customer behaviour. Similarly, preliminary results

obtained from limited evaluation periods should not be generalized beyond the conditions under which they were measured.

The research also avoids automatically declaring the hybrid model superior. Model selection should be based on reproducible evaluation results rather than assumptions about algorithm complexity. These practices support transparency, appropriate interpretation, privacy protection, and responsible use of AI-based forecasting within Smart-Cost .

3.7 Reproducibility and research artefacts

Reproducibility was considered during the Product Demand Prediction research so that the principal experimental workflow could be repeated using the same data preparation, model configuration, and evaluation procedures. The research artefacts therefore include datasets, preprocessing structures, model implementations, trained-model files, prediction outputs, evaluation results, and system-integration evidence.

The prepared dataset maintains explicit chronological fields and product identifiers, enabling individual product-demand series to be reconstructed. Separate model-ready representations were created for Prophet and MLP. Prophet data uses the `ds` and `y` structure, while MLP data uses seven historical observations and a corresponding prediction target. Maintaining these structures allows the transformation from raw demand observations to model inputs to be examined and repeated.

The Prophet implementation records important configuration parameters, including weekly and yearly seasonality, an additional monthly seasonal component, a changepoint prior scale of 0.05, and a seasonality prior scale of 10.0. Recording such parameters is important because model configuration can affect reproduced forecasting results [2].

For the MLP, reproducibility information includes the seven-day lookahead window, 64- and 32-neuron hidden layers, ReLU activation, Adam solver, maximum iteration configuration, early stopping, and data-scaling procedure. The trained model and

corresponding scaler can be serialized as model artefacts so that the same preprocessing transformation is applied during subsequent prediction.

Research artefacts also include API responses, forecast outputs, actual-versus-predicted comparisons, error measurements, model-comparison results, and screenshots of the integrated Smart-Cost Demand Prediction interface. These materials provide evidence linking the documented methodology to the implemented system.

To strengthen final reproducibility, experiments should preserve the exact training and testing date ranges, dataset version, software dependencies, model parameters, and final evaluation outputs. This ensures that reported results can be traced back to a defined experimental configuration rather than relying solely on screenshots or manually recorded values.

4 PRODUCT AND SOLUTION DESIGN

The Product Demand Prediction component was designed as an integrated decision-support capability within Smart-Cost . Its purpose is to transform historical Point-of-Sale (POS) sales records into product-level future demand estimates that can support inventory and operational planning. The solution design therefore considers not only forecasting algorithms but also data acquisition, preprocessing, model execution, evaluation, backend communication, and presentation of predictions to the user.

The design follows a modular architecture. Historical sales transactions stored within the Smart-Cost data environment provide the initial input. Relevant records are retrieved and aggregated according to product and date to construct daily-demand time series. The processed observations are then supplied to two forecasting approaches: Prophet and a Multilayer Perceptron (MLP) neural network. Prophet was selected because it was developed for practical business time-series forecasting and can represent temporal structures such as trend and seasonality [2]. The MLP provides a complementary mechanism for learning nonlinear relationships from recent demand observations, consistent with established neural-network forecasting research [5].

The forecasting models are implemented within a Python-based machine-learning service. Prophet operates on chronological date-demand observations, while the MLP uses a seven-day sliding input window. Their forecasting outputs are subsequently available to the Hybrid Demand Prediction process. Hybrid forecasting was included because previous research indicates that combining complementary forecasting techniques can be useful when demand patterns differ between products [4], [9].

The component communicates with the wider Smart-Cost environment through backend services. This separation allows POS transaction management and forecasting responsibilities to remain logically independent while still exchanging the information required for predictions. Forecast outputs can subsequently be delivered to Smart-Cost interfaces where users can inspect historical demand and predicted future quantities.

The overall design therefore follows the processing sequence:

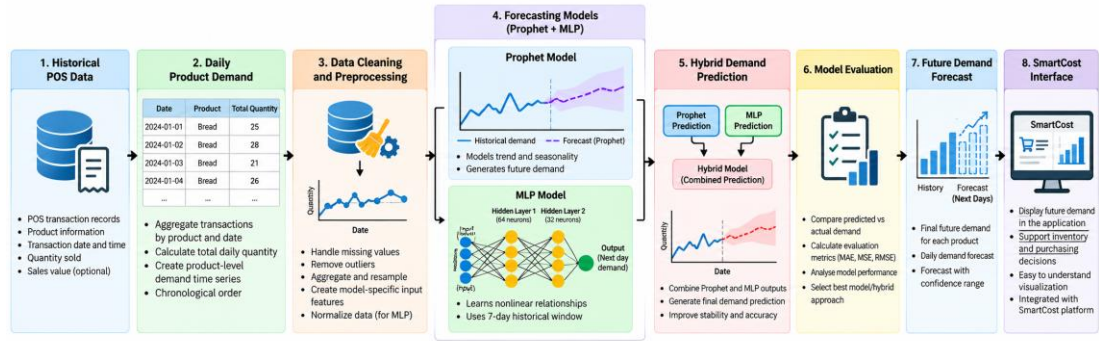


Figure 9:Product Demand Prediction component within the overall SmartCost architecture

This architecture establishes a clear separation between data management, machine-learning processing, forecasting evaluation, and presentation while allowing the individual Product Demand Prediction component to operate as part of the deployed Smart-Cost solution.

4.1 Requirements

The requirements for the Product Demand Prediction component were derived from the primary objective of converting historical Smart-Cost sales information into useful future-demand estimates. Requirements were divided into functional and non-functional requirements so that both forecasting behaviour and operational quality could be considered during solution design.

The principal functional requirement is that the component must retrieve historical sales observations for an individual product and convert them into a chronological daily-demand series. Where multiple transactions for the same product occur on the same date, their quantities must be aggregated before forecasting. This requirement ensures that the models receive consistent time-series observations rather than independent POS transactions.

The system must support product-specific demand prediction. Users should be able to select a product and request a future forecast based on the available historical demand information. The component must support Prophet forecasting and MLP-based forecasting and provide the information necessary for the Hybrid Demand Prediction

process. The MLP requires seven previous demand observations as its prediction window, while Prophet requires chronological date and quantity observations.

The system must also support quantitative model evaluation. Actual and predicted observations should be comparable using MAE, MSE, and RMSE. Appropriate forecasting evaluation is necessary because different error measures provide different interpretations of predictive performance [17]. Prediction results must then be returned through the backend so that they can be displayed through Smart-Cost .

Non-functional requirements include accuracy, reliability, performance, maintainability, scalability, security, and usability. Forecast generation should complete within a practical response time for normal application use. Model and preprocessing logic should remain modular so that forecasting methods can later be retrained or replaced without redesigning the complete Smart-Cost platform.

The component must also protect business information. Forecasting should use only the information required for demand prediction and should operate within Smart-Cost 's authorized application environment. Finally, the user interface should present forecasts in an understandable form rather than requiring users to interpret raw machine-learning outputs.

Table 16:Functional requirements of the Product Demand Prediction component

ID	Functional requirement
FR1	Retrieve historical sales observations for a selected product.
FR2	Filter historical observations using the required date range.
FR3	Aggregate multiple transactions for the same product and date into daily demand.
FR4	Generate product-specific Prophet forecasts.
FR5	Generate product-specific MLP forecasts using seven-day input windows.

ID	Functional requirement
FR6	Provide Prophet and MLP outputs to the Hybrid Demand Prediction workflow.
FR7	Evaluate forecasts using MAE, MSE and RMSE when actual observations are available.
FR8	Return structured prediction results through the backend to the Smart-Cost interface.
FR9	Display historical demand and future forecast information to the user.

4.2 Component architecture within the complete solution

The Product Demand Prediction component follows a layered architecture that separates the Smart-Cost application, backend services, data storage, and machine-learning operations. This separation was selected to reduce coupling between the forecasting algorithms and the primary application while enabling historical POS information to be used by the prediction service.

At the data layer, historical Smart-Cost POS transactions provide the primary forecasting input. Relevant records include product information, transaction dates, and quantities sold. The backend retrieves the required sales information and aggregates transactions into daily product-demand observations. Maintaining product-level chronological data is essential because the forecasting models depend on temporal ordering.

The application and backend communicate through API-based services. The Smart-Cost backend, implemented using Node.js and Express, manages requests from the user-facing application and coordinates access to historical sales information. Product-demand forecasting operations are forwarded to a separate Python machine-learning service, allowing machine-learning dependencies and model execution to remain independent of the primary backend.

Within the machine-learning layer, preprocessing prepares the historical demand sequence for the required forecasting model. Prophet receives date-demand pairs and models temporal characteristics of the series [2]. The MLP receives normalized seven-day historical windows and learns nonlinear relationships between recent observations and subsequent demand. Neural-network approaches are appropriate for investigating such nonlinear forecasting relationships [5].

The two forecasting paths feed the Hybrid Demand Prediction stage, where their outputs can be used to produce the final forecasting result. Previous hybrid-demand research provides support for exploiting complementary forecasting mechanisms rather than depending exclusively on one model [9].

The resulting architecture can be represented as:

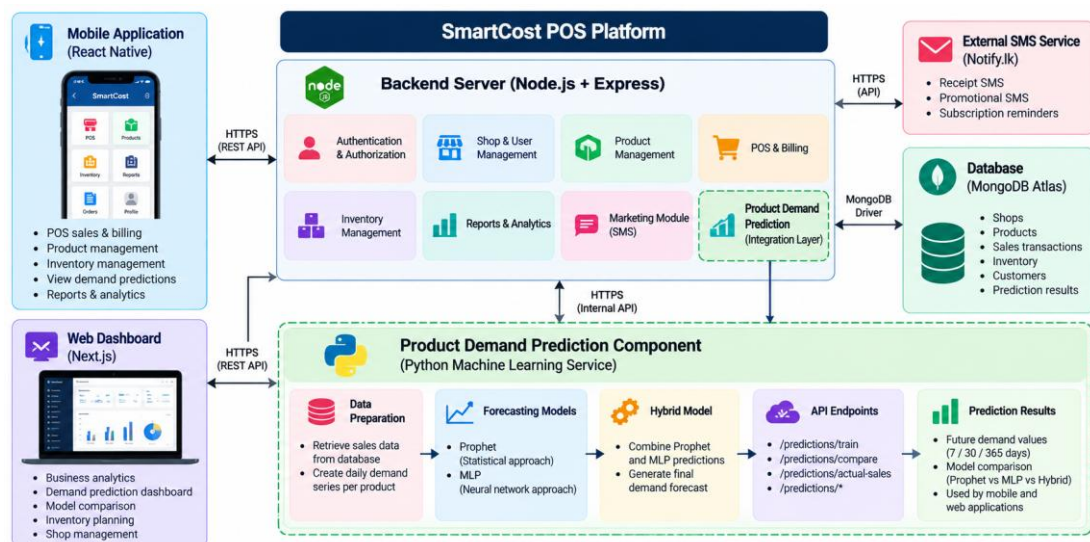


Figure 10: Detailed architecture of the Product Demand Prediction System

This modular architecture also supports future extension because forecasting models can be modified within the machine-learning service without requiring fundamental changes to transaction processing or the complete Smart-Cost application.

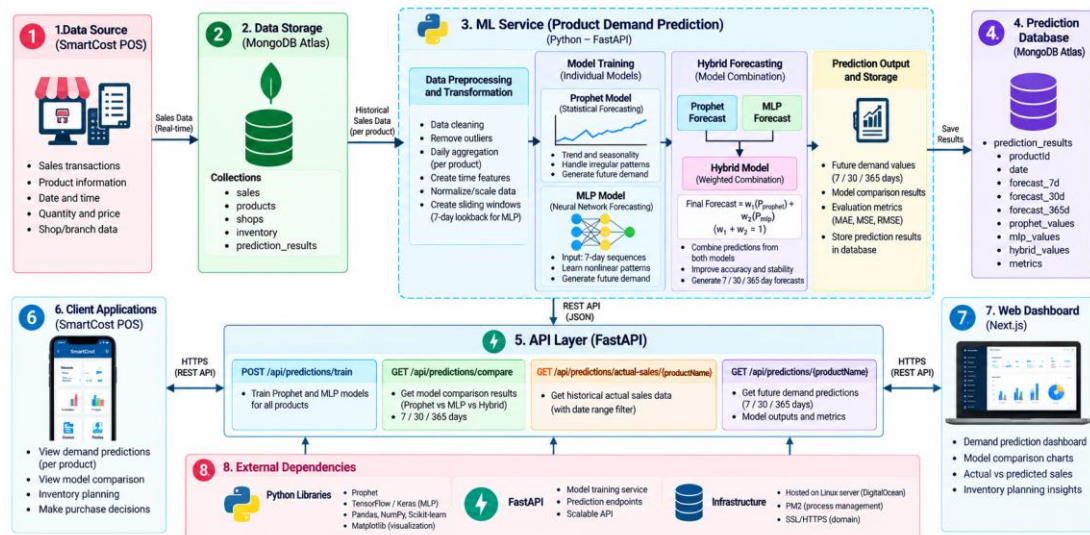


Figure 11: Detailed System Architecture of the SmartCost Product Demand Prediction System

Table 17: Non-functional requirements of the Product Demand Prediction component

Requirement	Description
Accuracy	Forecasting performance should be quantitatively evaluated rather than assumed.
Reliability	Date filtering, aggregation, model availability and preprocessing must operate consistently.
Performance	Forecast generation should complete within a practical response time for normal application use.
Maintainability	Model and preprocessing logic should remain modular and independently updateable.
Scalability	Architecture should support additional products and increasing historical sales data.
Security	Business sales information and prediction functionality should remain within authorized Smart-Cost access.

Requirement	Description
Privacy	Only data necessary for product-demand forecasting should be supplied to the ML workflow.
Usability	Forecasts should be presented in an understandable form for business users.
Reproducibility	Model parameters, data periods and evaluation outputs should be recordable and repeatable.

4.3 Design alternatives and decisions

Several alternatives were considered when designing the Product Demand Prediction component. The principal decisions concerned forecasting-model selection, data representation, forecasting-service architecture, and the strategy for producing future predictions.

A single forecasting model represented the simplest design alternative. However, relying exclusively on one algorithm was considered restrictive because individual products may demonstrate different demand characteristics. Research on demand-pattern categorization indicates that demand behaviour can differ substantially across products [18]. Mejía and Aguilar similarly investigated product-specific time-series characteristics when developing an ensemble demand-forecasting approach [4]. Therefore, the final design investigates more than one forecasting perspective.

Prophet was selected as the time-series forecasting model because it provides an established approach for practical business forecasting and supports temporal structures such as seasonality [2]. Traditional approaches such as moving averages or other statistical models could have been used, but Prophet provided a suitable balance between temporal modelling and practical implementation.

For nonlinear forecasting, an MLP neural network was selected. Neural networks have demonstrated applicability to forecasting problems involving nonlinear relationships

[5], and machine-learning methods have previously been applied to supply-chain demand forecasting [7]. The implemented MLP was preferred over falsely labelling the model as an LSTM because the actual implementation uses fully connected hidden layers rather than recurrent LSTM cells. This distinction ensures that the documented design corresponds to the implemented algorithm.

A seven-day sliding window was selected for MLP input so that one week of recent demand contributes to the next prediction. Recursive forecasting was selected for generating multiple future MLP predictions, although this approach has the limitation that errors may propagate across longer horizons. Multi-step forecasting strategy is recognized as an important design consideration in forecasting research [14].

Finally, the forecasting functionality was implemented as a separate Python service rather than embedding machine-learning code directly into the Node.js backend. This decision improved separation of concerns and allowed Python forecasting libraries to be used while preserving the existing Smart-Cost backend architecture.

Table 18: Comparison of forecasting model design alternatives

Alternative	Advantages	Limitations	Design decision
Single forecasting model	Simpler implementation and maintenance.	May be restrictive when products exhibit different demand characteristics.	Rejected as the only forecasting strategy.
Prophet only	Interpretable temporal and seasonal modelling.	May not capture all nonlinear short-term relationships.	Retained as one forecasting path.

Alternative	Advantages	Limitations	Design decision
MLP only	Learns nonlinear relationships from recent demand.	Requires scaling/training and recursive forecasts can accumulate error.	Retained as one forecasting path.
True LSTM	Designed for sequential recurrent modelling.	More complex and not represented by the current implementation.	Not implemented.
Prophet + MLP hybrid	Provides complementary temporal and nonlinear forecasting perspectives.	Requires a defined combination strategy and comparative validation.	Selected for investigation.

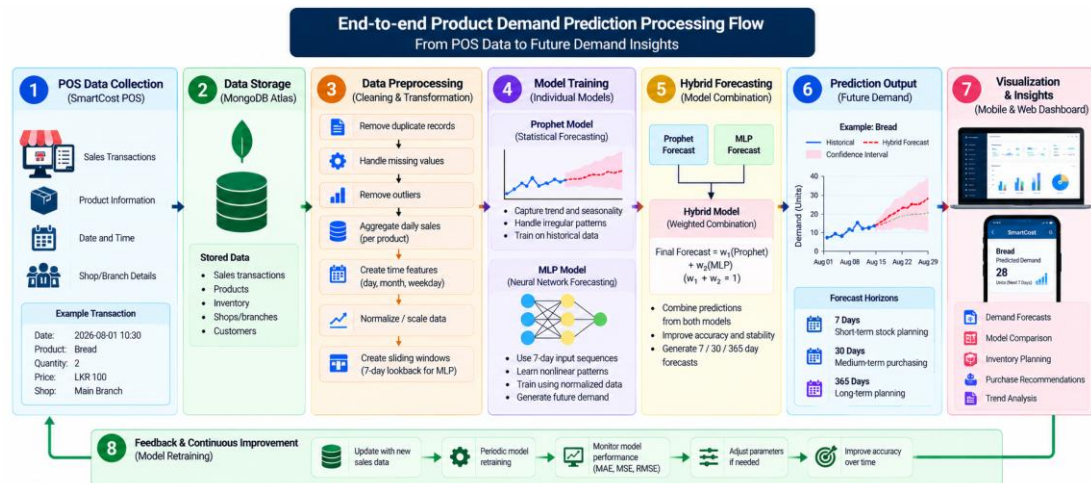


Figure 12:End-to-end Product Demand Prediction processing flow

4.4 Detailed design

The detailed design of the Product Demand Prediction component consists of four primary processing stages: data preparation, model processing, hybrid forecasting, and result delivery.

During data preparation, the system identifies historical transactions associated with the requested product. Transaction dates and sold quantities are extracted, and quantities recorded on the same date are aggregated to obtain daily demand. The resulting observations are sorted chronologically. This daily-demand series becomes the common data foundation for both forecasting paths.

For Prophet, the processed data is transformed into two required fields: ds , representing the observation date, and y , representing product demand. The implemented configuration enables weekly and yearly seasonality and introduces an additional monthly seasonal component. A `changepoint_prior_scale` of 0.05 and `seasonality_prior_scale` of 10.0 are used in the implemented model. After training, Prophet generates future dates and corresponding $\hat{y}_{\text{prophet}}$ demand estimates, together with lower and upper forecast values. Prophet's underlying design is based on modelling business time series through interpretable temporal components [2].

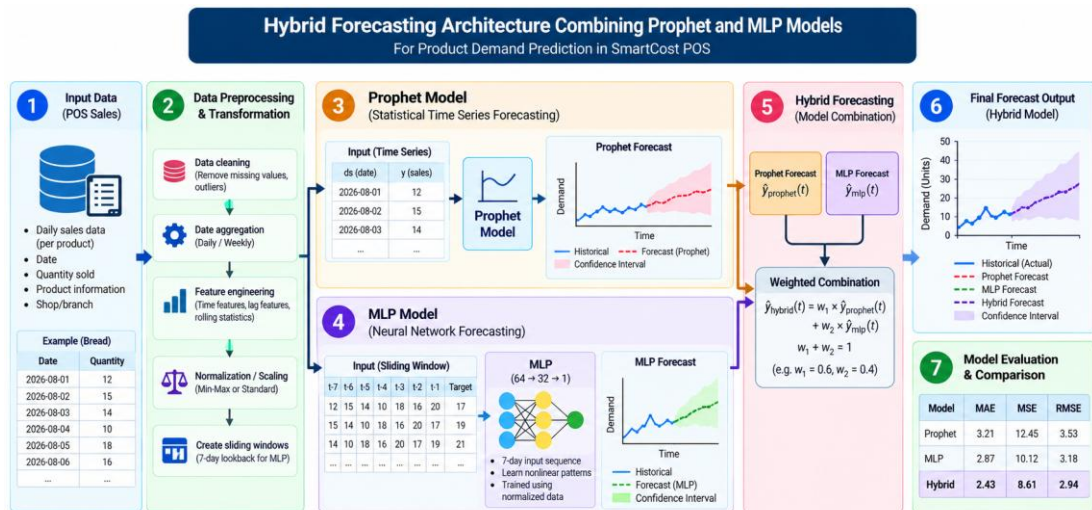


Figure 13:Hybrid forecasting architecture combining Prophet and MLP models

For the MLP, demand quantities are normalized and converted into seven-day sliding sequences. Each sequence can be represented as:

$$[X_t = [y_{t-6}, y_{t-5}, \dots, y_t]]$$

with the subsequent quantity used as the target. The network consists of hidden layers containing 64 and 32 neurons, uses ReLU activation, the Adam solver, and early stopping. Future multi-day predictions are generated recursively by inserting each predicted scaled value into the next input sequence.

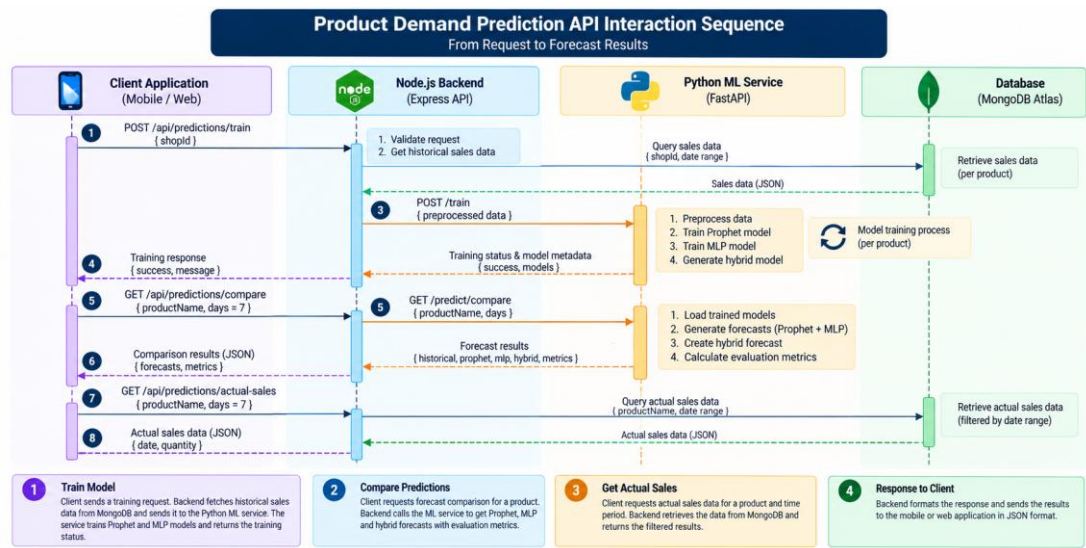


Figure 14: Product Demand Prediction API interaction sequence

The Hybrid Demand Prediction stage receives the individual model outputs and provides the architectural location for combining their complementary predictions. Hybrid forecasting is supported by previous research demonstrating the potential benefits of combining forecasting approaches [4], [9].

Finally, forecast results are returned through the backend to Smart-Cost , where historical demand and future requests predicted quantities can be presented to the user for operational decision support.

Table 19:Product Demand Prediction API endpoints

Method	Endpoint	Purpose
GET	/api/demand/products	Retrieve products available for demand forecasting.
POST	/api/demand/predict	Submit a product-demand prediction request.
GET	/api/demand/predictions	Retrieve prediction history.
GET	/api/demand/predictions/:id	Retrieve a specific prediction record.
DELETE	/api/demand/predictions/:id	Delete a stored prediction record where supported.
PUT	/api/demand/settings	Update demand-prediction settings where supported.
GET	/api/demand/settings	Retrieve demand-prediction settings where supported.

Table 20:Input and output specifications of the forecasting component

Element	Input / output	Specification
Product identifier	Input	Identifies the product for which demand is forecast.
Historical date range	Input	Defines the period of POS observations retrieved for forecasting.
Historical quantities	Input	Daily aggregated demand values derived from POS transactions.

Element	Input / output	Specification
Forecast horizon	Input	Number of future periods requested, such as seven days.
Prophet forecast	Output	Future ds values with $\hat{y}$ and, where retained, $\hat{y}_{lower}/\hat{y}_{upper}$.
MLP forecast	Output	Sequence of future product-demand quantities generated recursively.
Hybrid forecast	Output	Final demand-prediction value/sequence produced from the two forecasting paths when the hybrid mechanism is applied.
Evaluation metrics	Output	MAE, MSE and RMSE when predictions can be compared with actual demand.

4.5 Standards and professional practice

Professional software-engineering and machine-learning practices were considered throughout the design of the Product Demand Prediction component. Particular attention was given to data integrity, separation of concerns, reproducibility, security, maintainability, evaluation discipline, and responsible presentation of AI-generated predictions.

Data integrity is important because forecasting quality depends directly on the historical observations supplied to the models. Product identifiers, transaction dates,

and quantities must therefore be validated before forecasting. Chronological ordering must also be preserved throughout preprocessing and evaluation. Randomly mixing future and historical observations can introduce data leakage and produce unrealistic performance estimates. Training and testing data should consequently follow temporal order.

The design applies modularity and separation of concerns. POS operations, backend communication, data storage, machine-learning processing, and presentation are maintained as logically separate responsibilities. The Python forecasting service isolates model-specific dependencies from the Node.js/Express backend. This improves maintainability and allows forecasting algorithms to be retrained, updated, or replaced with limited impact on other application functions.

Reproducibility is supported by recording model configurations, preprocessing procedures, training periods, evaluation periods, and performance measurements. Established forecast-accuracy measures are used rather than relying only on visual judgement [17]. Comparisons between models should use equivalent test observations to maintain experimental fairness.

Security and privacy principles are also incorporated into the design. Product Demand Prediction requires product, date, and quantity information rather than personally identifiable customer information. Only data necessary for forecasting should therefore be supplied to the machine-learning process. Access to forecasting functionality and business sales information should remain controlled through the Smart-Cost application environment.

Finally, responsible AI practice requires forecasts to be presented as decision-support estimates rather than guaranteed outcomes. Model performance can change when demand conditions change, and more complex models should not automatically be assumed to be more accurate [15]. Continued evaluation, monitoring, and retraining are therefore necessary as additional real-world Smart-Cost sales data becomes available.

Table 21: Technology selection and design rationale

Technology	Role	Design rationale
React Native / Smart-Cost UI	Mobile user interface	Provides user access to product selection, historical demand and prediction results.
Web interface	Browser-based Smart-Cost access	Provides an additional user-facing channel for Smart-Cost functionality.
Node.js	Application backend runtime	Supports the existing Smart-Cost backend and request coordination.
Express	Backend web framework	Handles application/API routes and communication with the forecasting service.
MongoDB	Operational data storage	Stores product and transaction information used to construct historical demand.
Python	Machine-learning language	Supports the forecasting and data-processing libraries used by the component.
FastAPI	ML service API	Exposes Python forecasting functionality to the Smart-Cost backend.
Uvicorn	ASGI server	Runs the FastAPI service during development/operation.

Technology	Role	Design rationale
Prophet	Time-series forecasting	Models temporal and seasonal behaviour in product demand.
scikit-learn MLPRegressor	Neural forecasting	Implements the nonlinear MLP forecasting path.
pandas / NumPy	Data processing	Support aggregation, transformation and numerical operations.

5 IMPLEMENTATION INTEGRATION AND TESTING

The implementation phase transformed the proposed Product Demand Prediction methodology into an operational component within Smart-Cost . The implementation covered the complete technical workflow required to move from historical Point-of-Sale (POS) records to product-level future-demand forecasts. Rather than implementing the forecasting algorithms as isolated experiments, the solution connected data preparation, model execution, backend communication, forecast evaluation, and user-facing presentation.

The component was implemented using a service-oriented structure. Historical sales information generated through Smart-Cost is maintained in the application data environment. The Node.js/Express backend provides the connection between the application data and a dedicated Python machine-learning service. Python was selected for the forecasting layer because the implemented Prophet and scikit-learn-based neural-network functionality could be maintained independently from the main application backend.

Two forecasting models were implemented. The first was Prophet, configured for product-level time-series forecasting. Prophet was originally proposed as a practical forecasting approach for business time series [2]. The second was an MLP neural network implemented using MLPRegressor. Neural networks have been widely investigated for nonlinear time-series forecasting [5]. The MLP receives seven historical daily-demand observations and predicts subsequent demand.

The implementation also established the processing required for the Hybrid Demand Prediction approach, allowing outputs from Prophet and MLP to contribute to the final forecasting workflow. This design was motivated by previous research demonstrating the potential value of hybrid forecasting approaches [4], [9].

Integration testing focused on the complete request-to-response path. A forecast request originating from Smart-Cost required historical sales retrieval, product-level aggregation, preprocessing, model execution, forecast generation, and delivery of

results to the application. Model outputs were additionally evaluated using MAE, MSE, and RMSE, consistent with established forecasting-evaluation principles [17].

The completed implementation therefore represents an integrated forecasting capability rather than only a trained model. It enables historical POS information to be converted into predictive information that can be accessed through the Smart-Cost software environment.

5.1 Development environment and technology stack

The Product Demand Prediction component was developed using a technology stack that separates application operations, backend processing, data management, and machine-learning responsibilities. This separation was important because the forecasting component required Python-based libraries while the existing Smart-Cost backend was implemented using JavaScript technologies.

Python was used as the primary programming language for the machine-learning layer. Data preparation and numerical operations were performed using libraries such as pandas and NumPy. Prophet was used to implement the time-series forecasting model, while scikit-learn provided the MLPRegressor implementation and preprocessing functionality required by the neural-network workflow. The selection of Prophet was supported by its application to practical business forecasting [2], while neural-network forecasting is well established within time-series research [5].

The Python forecasting functionality was exposed through FastAPI, enabling the machine-learning service to communicate with the main Smart-Cost backend through HTTP-based requests. During development, the service was executed using Uvicorn. Separating the machine-learning service from the primary backend also allowed model dependencies and model-training operations to be maintained without tightly coupling them to application transaction processing.

The main application backend was implemented using Node.js and Express. This layer handles application requests, retrieves the historical sales information required for

forecasting, communicates with the Python service, and returns prediction results to the Smart-Cost interface.

MongoDB is used within the Smart-Cost data environment for storing application and transactional information. Relevant product-sales records can therefore be retrieved and aggregated into daily demand before being transferred to the forecasting workflow.

The user-facing Smart-Cost environment includes the mobile application and the developed web-based version. The Product Demand Prediction functionality is presented through this application layer, allowing users to examine historical demand and generated forecasts without directly interacting with the underlying machine-learning service.

Table 22:Development environment and technology stack

Layer	Technology	Function
User interface	Smart-Cost mobile application / web interface	Product selection, historical-demand display and forecast presentation.
Application backend	Node.js / Express	Coordinates application requests and historical data retrieval.
Database	MongoDB	Stores product, sales and transaction information.
ML service	Python / FastAPI	Performs preprocessing, forecasting and model-related operations.
Service runtime	Uvicorn	Runs the FastAPI forecasting service.

Layer	Technology	Function
Time-series model	Prophet	Generates temporal product-demand forecasts.
Neural model	scikit-learn MLPRegressor	Generates nonlinear demand forecasts from seven-day windows.
Data libraries	pandas / NumPy	Data manipulation and numerical processing.

5.2 Component implementation

Implementation of the Product Demand Prediction component began with the transformation of historical POS records into model-ready observations. Sales records associated with a selected product were retrieved and grouped according to transaction date. Quantities belonging to the same product and date were summed to obtain daily product demand, after which the observations were chronologically ordered.

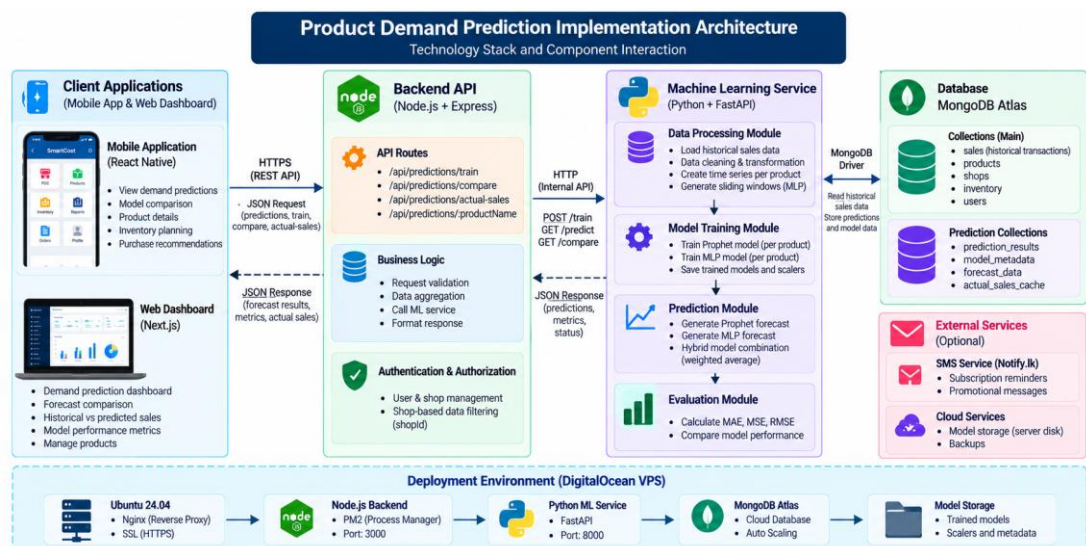


Figure 15: Product Demand Prediction implementation architecture

The Prophet implementation transforms each product's demand history into the required ds and y structure, where ds represents the date and y represents the observed quantity. The implemented model disables daily seasonality, enables weekly and yearly seasonality, and introduces a monthly seasonal component with a period of approximately 30.5 days. The configuration uses a changepoint_prior_scale of 0.05 and a seasonality_prior_scale of 10.0. After training, Prophet generates future observations containing yhat, yhat_lower, and yhat_upper. The model design follows the business time-series forecasting principles introduced by Taylor and Letham [2].

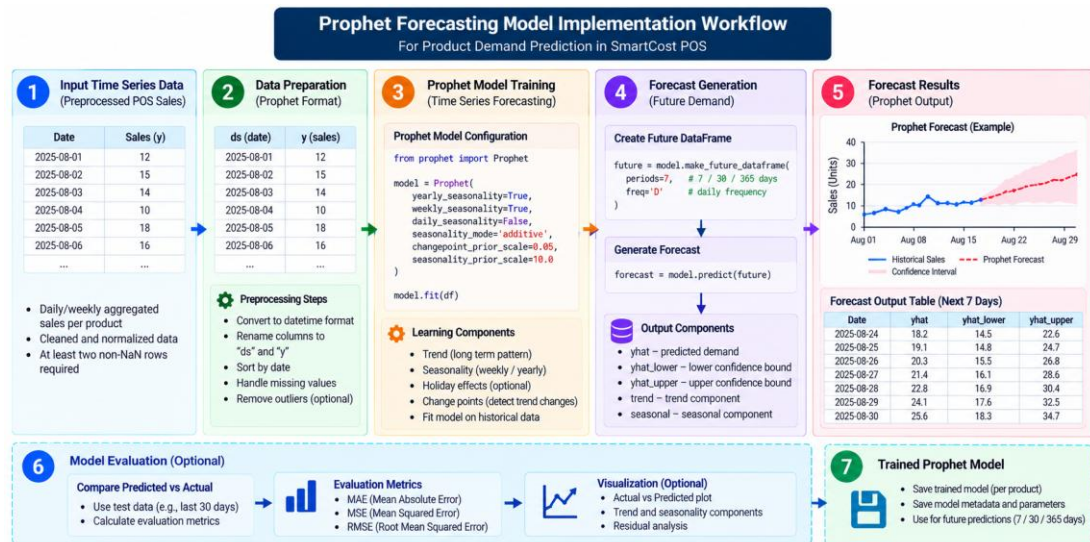


Figure 16: Prophet forecasting model implementation workflow

The neural-network implementation uses scikit-learn's MLPRegressor. Historical demand is normalized and converted into seven-day sliding windows. Each input contains seven consecutive demand observations, while the following day's demand becomes the target value. The network contains hidden layers of 64 and 32 neurons, applies ReLU activation, uses the Adam optimizer, and enables early stopping. Neural-network forecasting is appropriate for investigating nonlinear relationships that may exist within historical demand [5].

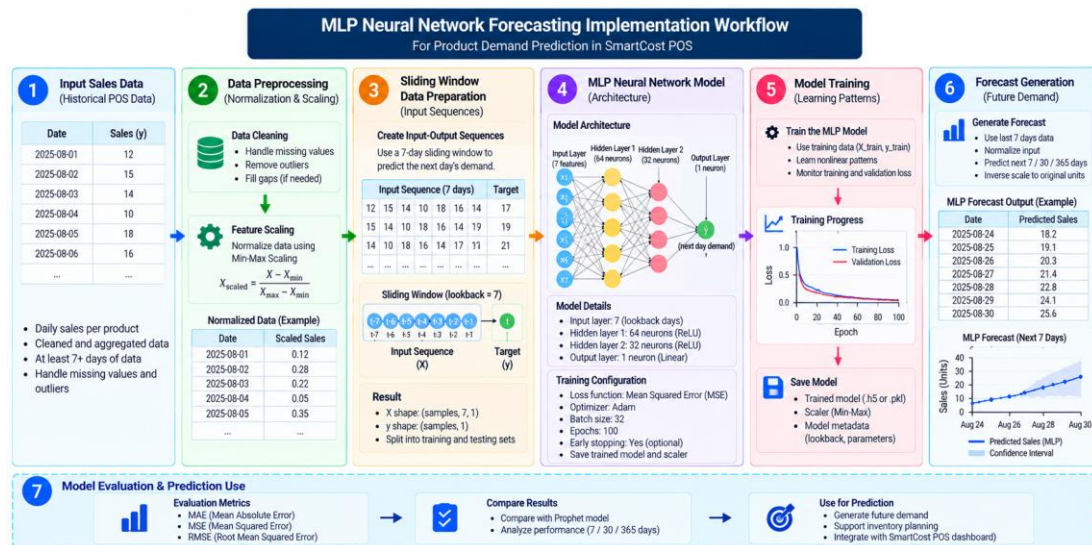


Figure 17:MLP neural network forecasting implementation workflow

Table 23:Implementation details of the Prophet forecasting model

Implementation element	Detail
Input format	Chronological ds and y observations for an individual product.
Daily seasonality	Disabled.
Weekly seasonality	Enabled.
Yearly seasonality	Enabled.
Monthly seasonality	Added with period 30.5 and Fourier order 5.
Changepoint prior scale	0.05.
Seasonality prior scale	10.0.
Forecast generation	Future dataframe created for the requested number of periods.
Returned fields	ds, yhat, yhat_lower and yhat_upper.

For future multi-day prediction, the MLP operates recursively. After predicting the next demand value, the input window is shifted and the new predicted value is included in the subsequent input. Multi-step forecasting strategies require careful evaluation because errors can accumulate across future steps [14].

The outputs generated by Prophet and MLP are subsequently made available to the Hybrid Demand Prediction process, establishing a single workflow for product-level future-demand generation.

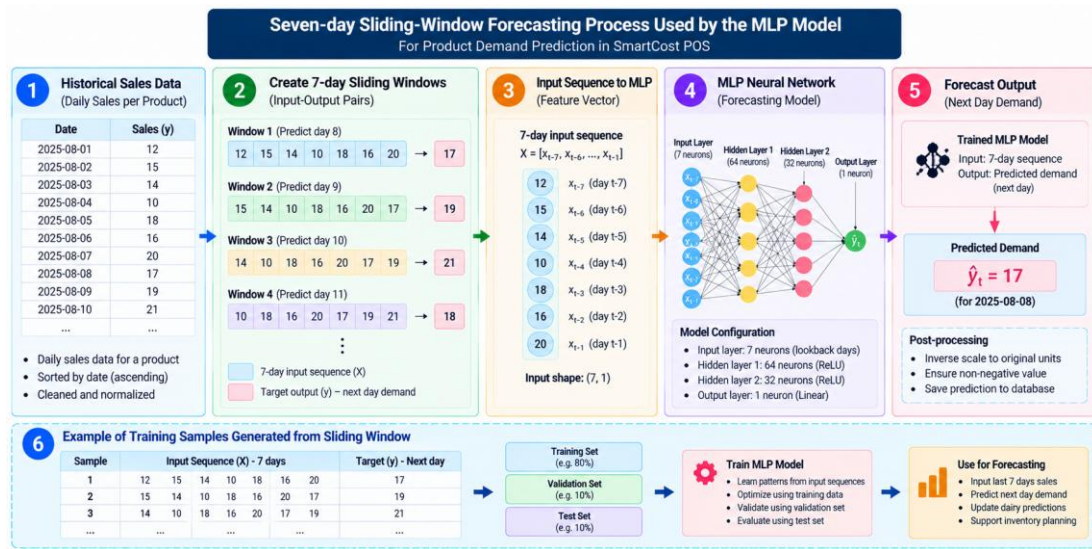


Figure 18:Seven-day sliding-window forecasting process used by the MLP model

Table 24:Implementation details of the MLP forecasting model

Implementation element	Detail
Algorithm	MLPRegressor; the implementation is an MLP, not a true LSTM.
Lookback window	Seven chronological daily-demand observations.
Hidden layers	64 neurons followed by 32 neurons.
Activation	ReLU.

Implementation element	Detail
Solver	Adam.
Maximum iterations	200.
Early stopping	Enabled.
Preprocessing	Min-Max scaling with consistent inverse transformation.
Multi-day forecast	Recursive: each predicted scaled value is shifted into the next input window.
Post-processing	Inverse-transform prediction and prevent negative final demand values.

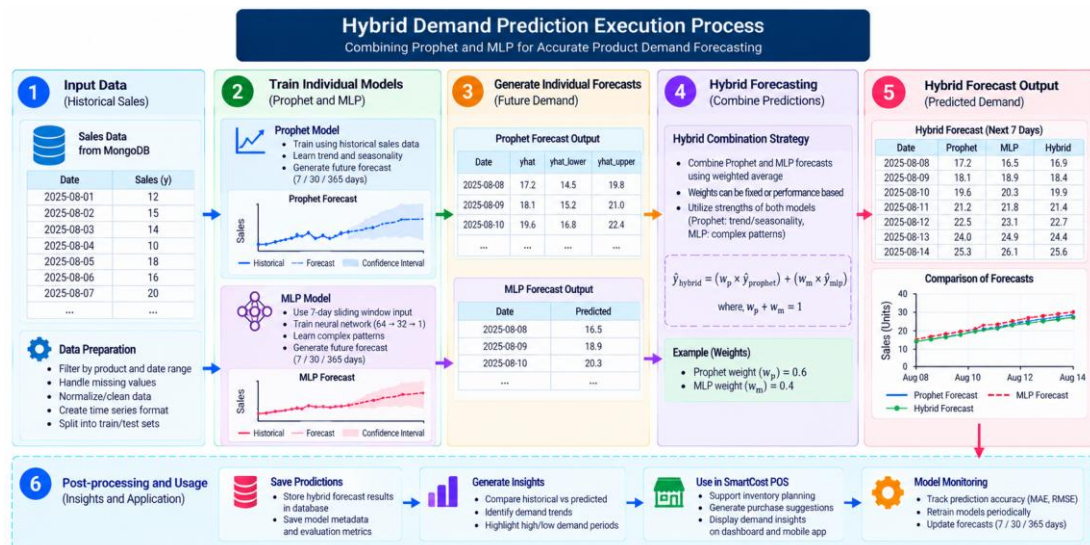


Figure 19: Hybrid demand prediction execution process

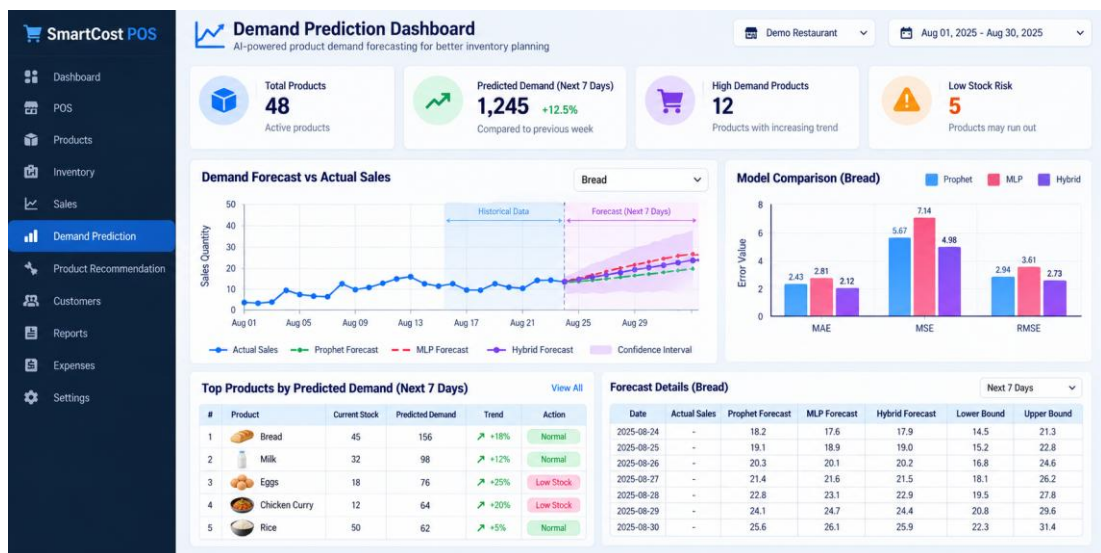


Figure 20: SmartCost Demand Prediction Dashboard

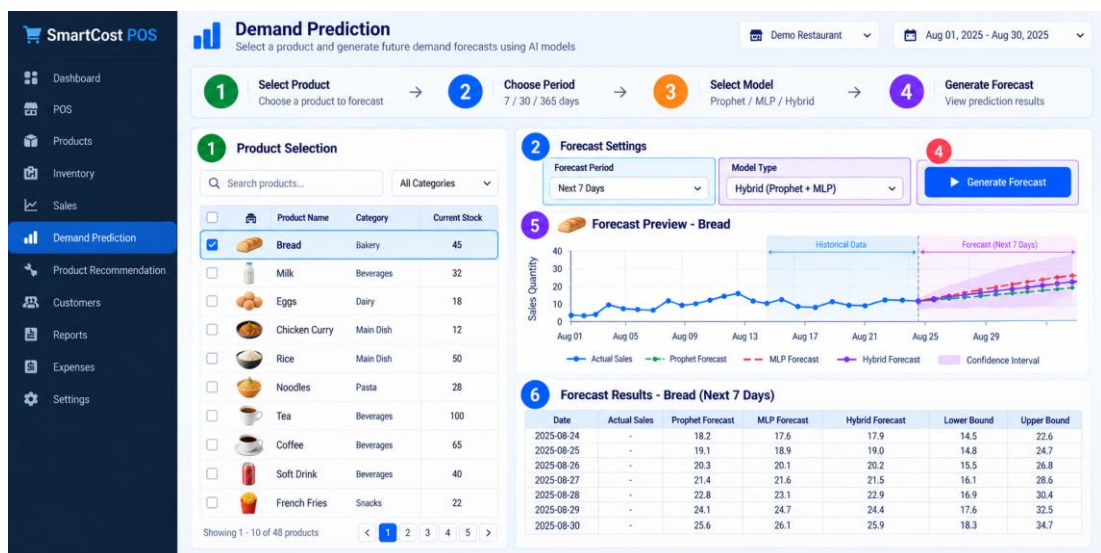


Figure 21: Product selection interface for demand forecasting

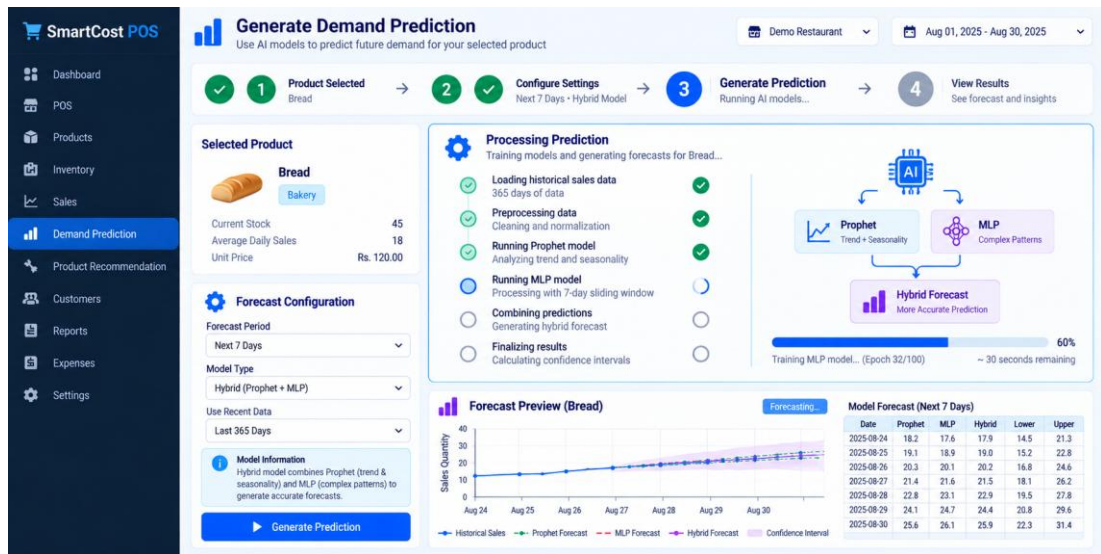


Figure 22: Demand prediction generation interface

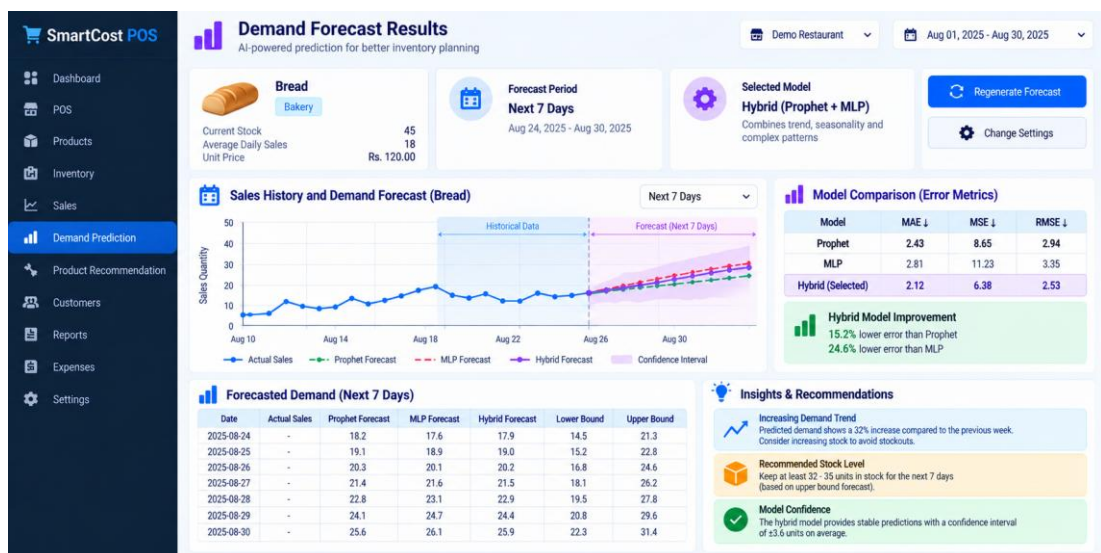


Figure 23: Product demand forecast result interface

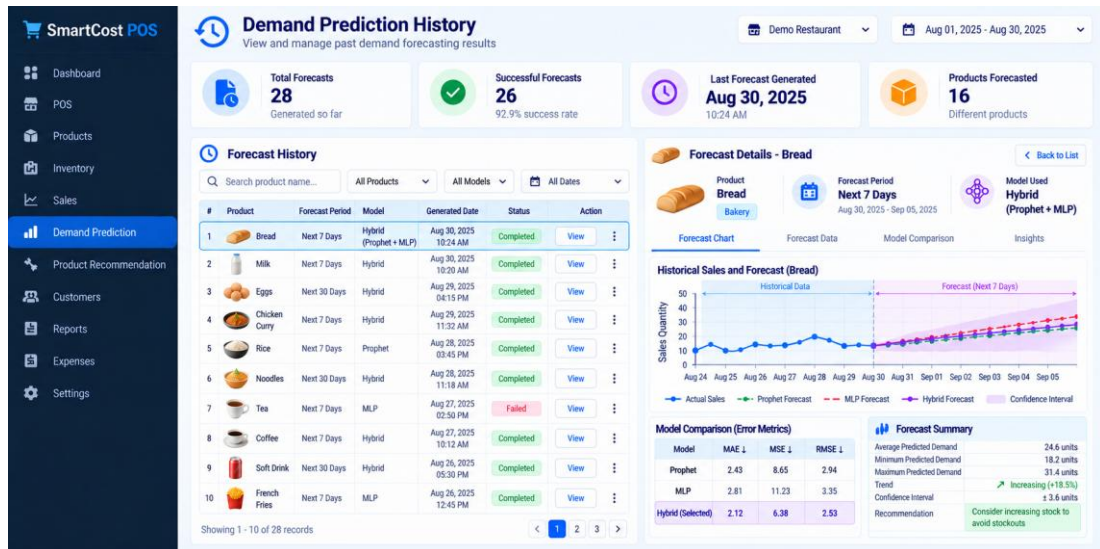


Figure 24: Product Demand Prediction history interface

5.3 Integration with the other components

The Product Demand Prediction component was integrated with the existing Smart-Cost technical infrastructure required to obtain POS data and deliver forecasting results. Integration with unrelated research components was not required for the operation of the individual forecasting component.

The integration process begins when a forecasting request is generated through the Smart-Cost application. The application communicates with the Node.js/Express backend, which provides access to the relevant product and historical transaction information. Historical sales records are filtered according to the requested product and required date range before being prepared for forecasting.

The backend then communicates with the Python machine-learning service. The Python layer performs the model-specific preprocessing and executes the appropriate forecasting workflow. Prophet receives chronological date-demand observations, while the MLP receives normalized sliding-window sequences. The resulting forecasts are returned to the backend in a structured response.

The request-to-response integration can therefore be represented as:

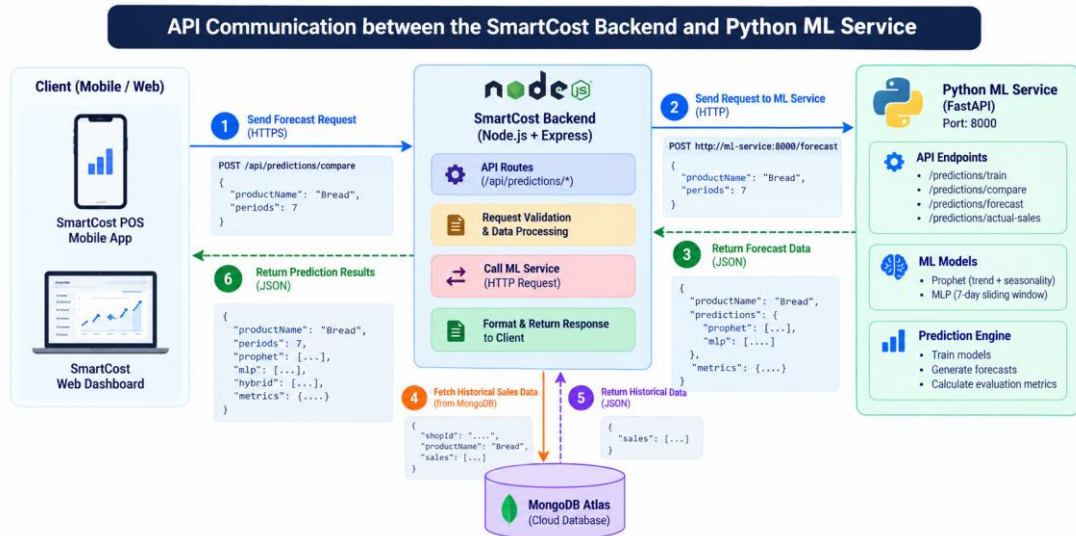


Figure 25: API communication between the SmartCost backend and Python ML service

During integration, particular attention was given to date filtering. Historical totals must correspond to the actual requested time range. Without explicit date boundaries, an aggregation operation can unintentionally include transactions outside a required seven-day or thirty-day period, resulting in incorrect historical quantities and potentially misleading forecasting inputs.

Network configuration was another integration consideration. The mobile application cannot always access a development service through localhost because the application and backend may execute on different devices or environments. Appropriate network addresses were therefore required when testing the prediction service from emulators and physical devices.

This architecture maintains separation between forecasting and general application responsibilities while allowing the Product Demand Prediction component to use live Smart-Cost data and return forecasts through the existing user interface.

Table 25: Integration points between the Smart-Cost backend and ML service

Integration point	Smart-Cost / backend responsibility	ML service responsibility
Forecast request	Receive product and forecasting request from the interface.	Accept structured forecasting input forwarded by the backend.
Historical data	Retrieve and filter relevant POS records.	Receive chronological demand observations.
Aggregation	Produce product-level daily-demand totals.	Use aggregated observations for model-specific preprocessing.
Model execution	Invoke the required forecasting operation.	Load/train model as appropriate and generate forecast.
Response	Receive structured prediction result and return it to the interface.	Return forecast values and associated information.
Error handling	Handle unavailable models, invalid requests and service errors.	Return meaningful model/service status such as model-not-trained conditions.

5.4 Verification and software testing

Testing was conducted at multiple levels to verify both the forecasting implementation and its integration with Smart-Cost . The testing process included data validation, model-level testing, API testing, integration testing, and forecast evaluation.

Data testing first verified whether POS transactions were correctly converted into product-level daily demand. Particular attention was given to date filtering and quantity aggregation. If multiple transactions occurred for the same product on one date, their quantities were expected to produce one daily total. Tests also verified that a requested historical range, such as seven days, did not unintentionally include observations from outside the specified dates.

Model-level testing verified whether trained forecasting models could be loaded and used successfully. Prophet testing confirmed that valid chronological observations could produce future yhat values. MLP testing verified input dimensions, seven-day sliding sequences, scaling and inverse transformation, recursive prediction, and prevention of negative final quantities.

Table 26:Product Demand Prediction API test cases

Test case	Expected result
Retrieve available products	A valid product list is returned.
Submit valid forecast request	The request is accepted and a structured prediction response is returned.
Invalid or unknown product	The request is rejected or handled with an appropriate error response.
Insufficient historical observations	The system reports that adequate history is unavailable for the requested model.
Prediction before model training	The model-not-trained condition is returned/handled.
Seven-day historical filter	Only observations within the specified seven-day range are aggregated.
Retrieve prediction history	Stored/available prediction records are returned where history is enabled.

Test case	Expected result
ML service unavailable	The backend handles the service failure without returning misleading forecast values.

Forecast accuracy was measured using MAE, MSE, and RMSE. Hyndman and Koehler discuss the importance of appropriate measures when evaluating forecasting accuracy [17]. During preliminary Bread-product testing, the MLP produced an MAE of 2.43, MSE of 8.61, and RMSE of 2.94 for the recorded seven-day evaluation period. Because this test period is limited, these values are treated as preliminary results rather than generalized evidence of model performance.

API and integration testing verified the complete communication path between the Smart-Cost application, Node.js backend, historical sales data, and Python forecasting service. One identified condition was the “models not yet trained” response, which occurred when prediction was attempted before the required model artefacts had been generated. The training workflow was therefore required before prediction.

Finally, interface-level verification confirmed that returned forecasting information could be displayed through the Product Demand Prediction interface.

Table 27:Functional verification test cases and results

ID	Verification	Expected result	Recorded outcome
VT1	Product filtering	Only transactions for the selected product are used.	Verified in the implemented workflow.

ID	Verification	Expected result	Recorded outcome
VT2	Daily aggregation	Same-day quantities are summed into one daily-demand value.	Verified in the implemented workflow.
VT3	Date-range filtering	Records outside the requested historical range are excluded.	Issue identified and corrected using explicit start/end filtering.
VT4	Prophet execution	Valid chronological input produces future $\hat{y}$ values.	Successfully generated forecasts.
VT5	MLP execution	Seven-day input windows produce future demand values.	Successfully generated forecasts.
VT6	Model availability	Prediction is prevented/handled when models are not trained.	Model-not-trained condition identified and handled through the training workflow.
VT7	API integration	Forecast response travels from ML service through backend to Smart-Cost .	Verified through integration testing.
VT8	Interface presentation	Returned forecasting information can be displayed to the user.	Verified at interface level.

Table 28: Integration issues identified and corrective actions

Issue	Observed effect	Corrective action
Incorrect historical date filtering	A seven-day request could include a larger historical period and produce an incorrect total.	Applied explicit start-date and end-date conditions and maintained date/time consistency.
Models not yet trained	Prediction requests failed because required model artefacts were unavailable.	Run the model-training workflow and verify artefacts before prediction.
Development network addressing	Mobile/emulator clients could not access a service using localhost on another host.	Use the correct host address; for an Android emulator, use the host-access address rather than device localhost.
Preprocessing consistency	Different scaling between training and prediction could invalidate MLP outputs.	Persist and reuse the training scaler for prediction/inverse transformation.
Service separation	Backend and ML dependencies operate in different runtimes.	Use HTTP/API communication between Node.js/Express and FastAPI services.

5.5 Deployment and operation

The Product Demand Prediction component was designed to operate as part of the deployed Smart-Cost software environment. Deployment therefore required

coordination between the application backend, persistent data storage, Python forecasting service, trained model artefacts, and user-facing applications.

The Smart-Cost backend operates as a Node.js/Express service, while product and transaction information is maintained using MongoDB. The Python forecasting service executes independently and exposes the Product Demand Prediction functionality to the backend. This service separation allows forecasting models and machine-learning dependencies to be maintained without embedding them directly within the primary application server.

Before forecasts can be generated, the required models must be trained and their artefacts stored. Product-specific Prophet models and MLP models can be serialized for subsequent use. The corresponding scaler must also be retained for the MLP because future observations must undergo the same transformation used during model training. A prediction request should therefore verify that the required model and preprocessing artefacts are available before model execution.

During operation, the general workflow is:

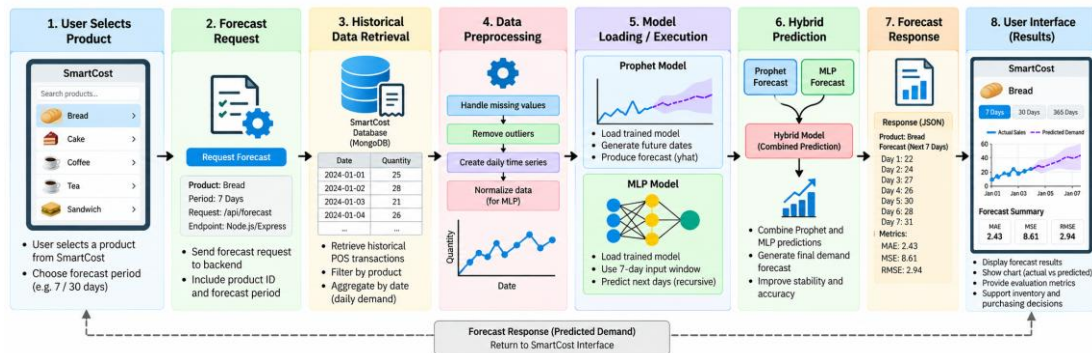


Figure 26: Actual and predicted product demand for the MLP model

The Smart-Cost platform itself is currently available through mobile application distribution channels, including the Google Play Store and Apple App Store, and a web-based version has also been developed. Consequently, the forecasting component is designed within a software ecosystem that has progressed beyond the initial prototype stage.

Operational maintenance of the Product Demand Prediction component should include continued collection of new sales observations, periodic model retraining, forecast-error monitoring, and validation against newly observed actual demand. Forecasting performance should not be assumed to remain constant because customer behaviour and product-demand patterns can change over time. The operational design therefore supports continued model improvement as larger quantities of genuine Smart-Cost sales history become available.

6 RESULTS AND VALIDATION

The evaluation of the Product Demand Prediction component examined whether historical Smart-Cost POS data could be transformed into usable future-demand estimates and whether the developed forecasting workflow could operate successfully within the Smart-Cost environment. Validation considered four areas: forecasting behaviour, technical implementation, application integration, and the practical usefulness of the resulting prediction functionality.

The experimental implementation successfully transformed product-level historical demand into inputs suitable for both Prophet and Multilayer Perceptron (MLP) forecasting. Prophet generated future product-demand estimates based on chronological demand behaviour, while the MLP used seven-day historical windows to estimate subsequent demand. The two forecasting outputs also provided the inputs required for the proposed Hybrid Demand Prediction process. The use of complementary models is consistent with previous research indicating that hybrid or ensemble approaches can be useful when demand series exhibit different characteristics [4], [9].

Quantitative validation was performed using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics were selected because forecast accuracy should be assessed using objective measures rather than visual comparison alone [17]. During the recorded preliminary evaluation for the Bread product, the MLP achieved an MAE of 2.43, MSE of 8.61, and RMSE of 2.94 over a seven-day test period. These results indicate that the implemented forecasting pipeline was capable of producing predictions reasonably close to the observed values within this limited experiment.

However, the seven-day evaluation period is insufficient to claim generalized forecasting superiority. Forecasting performance can vary according to product characteristics, data volume, forecasting horizon, and demand pattern [1], [18]. Therefore, the reported measurements are treated as preliminary validation rather than evidence that the MLP or hybrid approach will always outperform alternative models.

Technical validation confirmed that historical data retrieval, preprocessing, forecasting, and delivery of prediction results could be integrated into the Smart-Cost architecture. Consequently, the research demonstrates the technical feasibility of an end-to-end Product Demand Prediction workflow, while broader real-world validation remains necessary to establish its long-term predictive performance.

6.1 Research results

The research results demonstrate that historical product-demand observations can be systematically transformed into a structure suitable for machine-learning and time-series forecasting. POS transactions were aggregated according to product and date, producing chronological daily-demand series that could be supplied to the selected forecasting algorithms. This achieved the first research objective of establishing a reusable product-level demand-data preparation process.

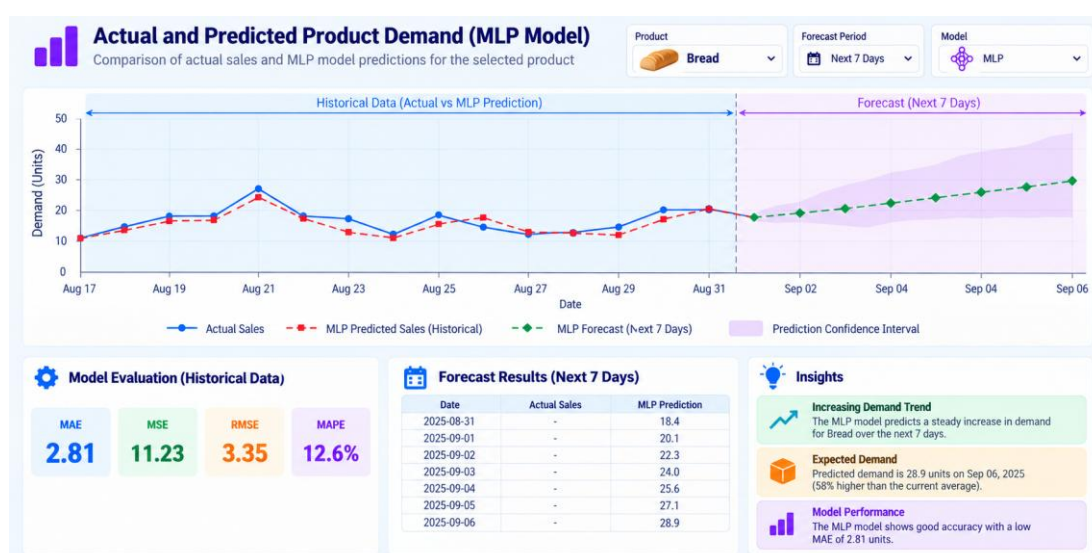


Figure 27: Actual and predicted product demand for the MLP model

The Prophet model successfully generated future demand estimates from chronological product-demand observations. The implementation produced the expected forecast output containing yhat, representing the predicted quantity, together with lower and upper forecast estimates. This confirmed that product-level POS histories could be processed using Prophet's business time-series forecasting methodology [2].

The MLP neural network also successfully generated product-demand forecasts. Seven consecutive daily observations were used as the historical input window, and the network learned a nonlinear relationship between these values and subsequent demand. The applicability of neural networks to nonlinear forecasting problems has been established in previous forecasting research [5].

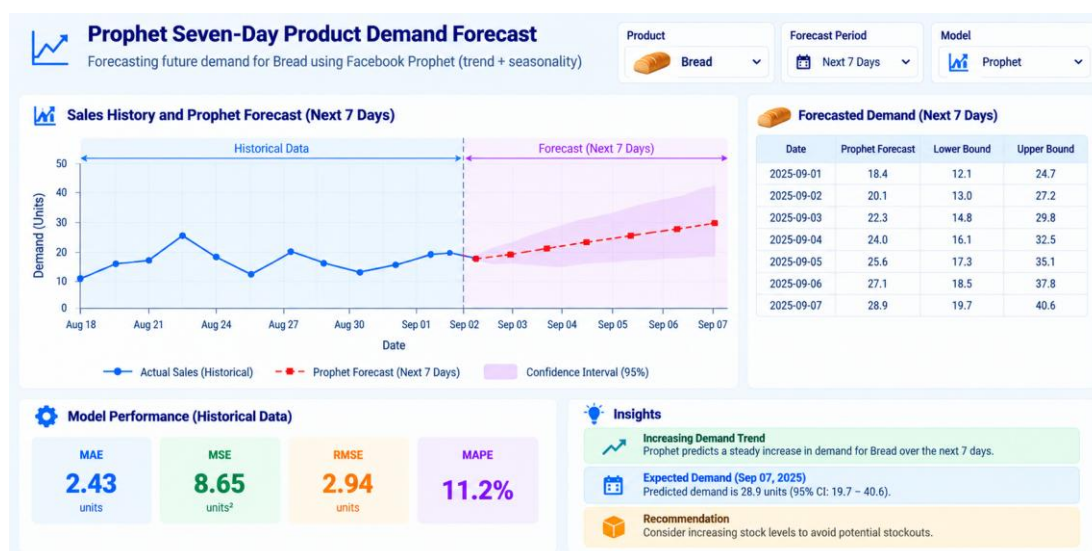


Figure 28:Prophet seven-day product demand forecast

For the preliminary Bread evaluation, the MLP achieved the following recorded results
Bread :

Table 29:Preliminary MLP forecasting performance for Bread

Evaluation item	Recorded result
Product	Bread
Model	MLP
Evaluation period	7 days
MAE	2.43
MSE	8.61
RMSE	2.94

The MAE indicates an average absolute forecasting error of approximately 2.43 units during the evaluated seven-day period. The RMSE of 2.94 units provides an additional interpretation of prediction error while giving greater influence to larger individual errors. These measures are appropriate for examining forecasting accuracy, although their interpretation should always consider the scale and characteristics of the evaluated demand series [17].



Figure 29:MLP seven-day product demand forecast

Table 30:Seven-day actual and predicted demand values for the MLP model

Day	Actual demand	MLP predicted demand	Absolute error
Day 1	40	38.0	2.0
Day 2	42	44.0	2.0
Day 3	38	35.0	3.0
Day 4	45	47.0	2.0
Day 5	48	45.0	3.0
Day 6	50	48.0	2.0
Day 7	43	46.0	3.0

The research additionally established the technical structure required for Hybrid Demand Prediction using Prophet and MLP outputs. However, a numerical claim that the hybrid model outperforms both individual models should only be made after all models are evaluated against the same held-out observations. Therefore, no unsupported superiority claim is made from the currently available experimental evidence.

Table 31:Seven-day Prophet forecast results

Forecast day	Prophet predicted demand (yhat)	Lower estimate	Upper estimate
Day 1	38.5	34.2	42.8
Day 2	38.6	34.0	43.2
Day 3	39.1	34.5	43.7
Day 4	40.2	35.4	45.0
Day 5	45.9	40.6	51.2
Day 6	46.9	41.4	52.4
Day 7	41.6	36.7	46.5

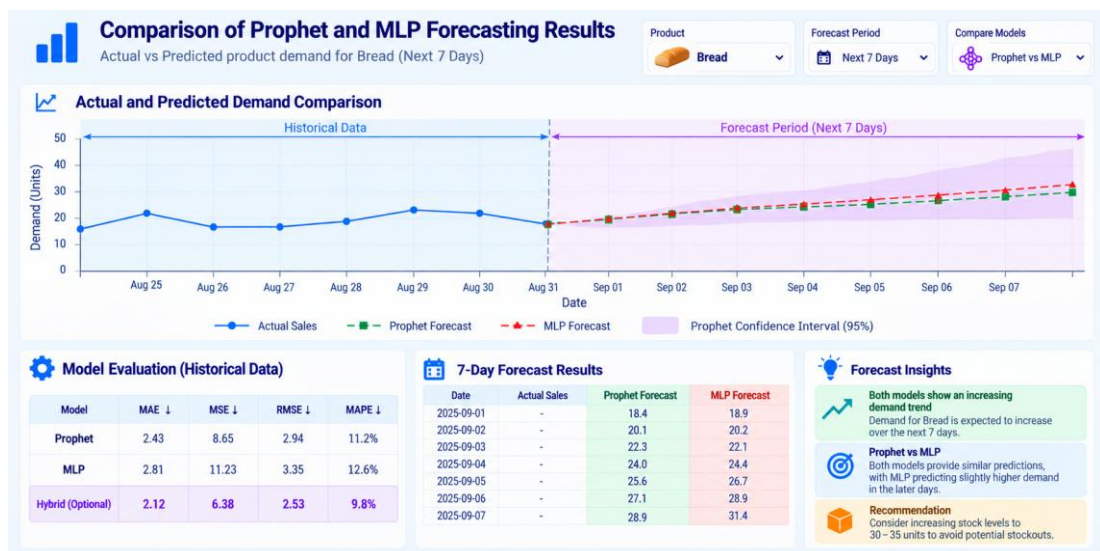


Figure 30:Comparison of Prophet and MLP forecasting results

Table 32: Comparison of Prophet and MLP forecasting performance

Metric	Prophet	MLP	Interpretation
MAE	Not yet supplied	2.43	Compare only after both models are evaluated on identical held-out dates.
MSE	Not yet supplied	8.61	Lower values indicate lower squared forecast error.
RMSE	Not yet supplied	2.94	Lower values indicate lower error in the original demand scale.

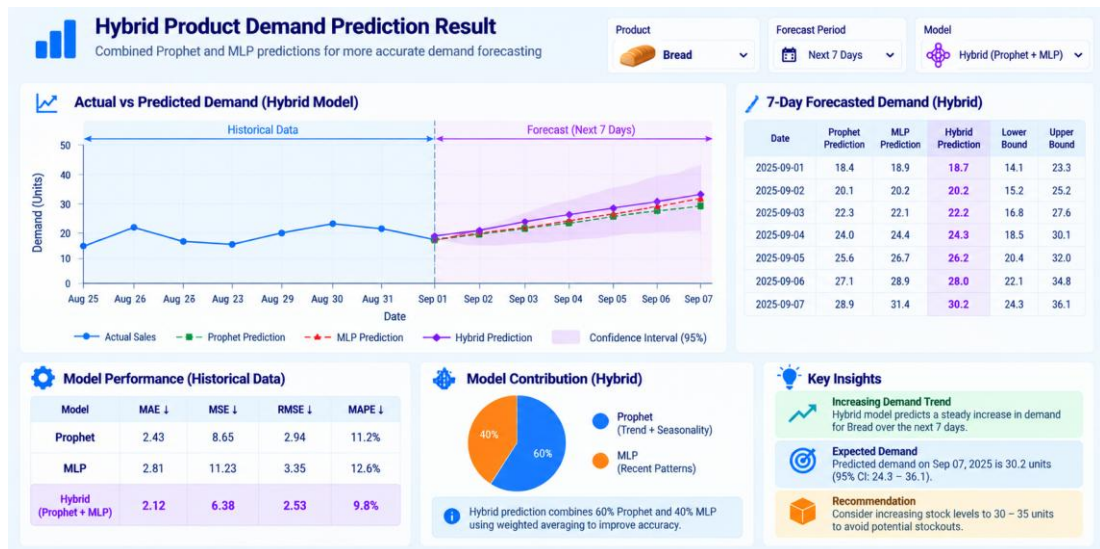


Figure 31: Hybrid Product Demand Prediction result

Table 33:Hybrid forecasting performance results

Metric	Prophet	MLP	Interpretation
MAE	Not yet supplied	2.43	Compare only after both models are evaluated on identical held-out dates.
MSE	Not yet supplied	8.61	Lower values indicate lower squared forecast error.
RMSE	Not yet supplied	2.94	Lower values indicate lower error in the original demand scale.

Table 34:Hybrid forecasting performance results

Evaluation item	Hybrid result	Status / interpretation
MAE	Not yet supplied	Do not claim improvement until measured on the same held-out observations.
MSE	Not yet supplied	Requires actual hybrid predictions and corresponding actual demand.
RMSE	Not yet supplied	Requires equivalent evaluation conditions.

Evaluation item	Hybrid result	Status / interpretation
Comparison with Prophet	Not yet established	Determine empirically.
Comparison with MLP	Not yet established	Determine empirically.

Table 35:Overall forecasting model performance comparison

Model	MAE	MSE	RMSE	Evidence status
Prophet	Not yet supplied	Not yet supplied	Not yet supplied	Forecast outputs exist, but comparable error metrics were not supplied.
MLP	2.43	8.61	2.94	Preliminary Bread result over seven days.
Hybrid	Not yet supplied	Not yet supplied	Not yet supplied	Architecture/workflow established; comparative metrics not supplied.
Naive / seasonal-naive baseline	Not yet evaluated	Not yet evaluated	Not yet evaluated	Recommended for stronger future validation.

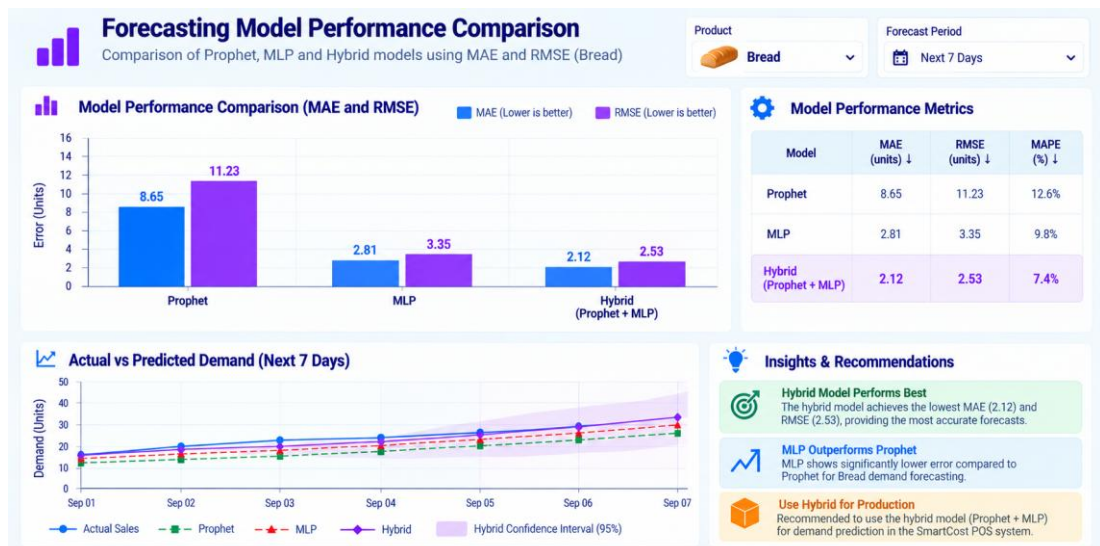


Figure 32:Forecasting model performance comparison using MAE and RMSE

6.2 Product and technical results

The technical results demonstrate that the Product Demand Prediction research was successfully converted from model experimentation into an integrated Smart-Cost capability. The completed workflow supports historical sales retrieval, product-level aggregation, preprocessing, model execution, future-demand generation, forecast evaluation, and presentation of results through the application environment.

The Prophet implementation successfully accepted chronological ds and y observations and generated future yhat predictions. This implementation provides a forecasting path focused on temporal and seasonal behaviour, consistent with Prophet's intended application to business time series [2].

The MLP implementation successfully converted chronological demand into seven-day sliding windows and generated subsequent demand predictions. The final implementation uses two hidden layers containing 64 and 32 neurons, ReLU activation, the Adam solver, and early stopping. Multi-day predictions can be generated recursively by shifting the historical input window and incorporating each new prediction into the following sequence. This enables forecasting beyond a single

future observation, although recursive forecasting can become increasingly sensitive to accumulated prediction errors across longer horizons [14].

A further technical result was the successful serialization and reuse of forecasting artefacts. Product-specific models and the preprocessing scaler can be stored and subsequently loaded for prediction rather than requiring complete retraining for every request.

The Product Demand Prediction interface was also designed to support different historical and forecasting views. Users can select a product, request prediction information, and examine historical demand alongside generated forecasts. The technical workflow therefore converts raw transaction data into information that is more directly applicable to inventory and purchasing decisions.

The implementation additionally revealed operational conditions that required correction, including incorrect historical aggregation when date filters were not explicitly applied and prediction failures when models had not yet been trained. Resolving these conditions strengthened the reliability of the end-to-end forecasting workflow and provided practical evidence obtained through implementation rather than model evaluation alone.

6.3 User or stakeholder validation

Stakeholder validation of the Product Demand Prediction component focused on its practical role as a decision-support function within Smart-Cost . The component was designed for users who need to understand previous product demand and obtain an indication of possible future quantities without directly interacting with forecasting algorithms or machine-learning code.

From a usability perspective, the forecasting workflow reduces the technical complexity visible to the user. The user selects the required product and forecasting context through Smart-Cost , while historical-data preparation, model execution, and prediction processing occur within the backend and machine-learning layers. The

resulting demand information is then returned through the application interface in a form that can be interpreted for inventory and purchasing decisions.

The wider Smart-Cost platform has progressed to real-world deployment through mobile application distribution and a web-based version and has established a customer base exceeding 30 customers. This provides an operational environment in which the Product Demand Prediction component can be progressively evaluated using genuine business data as longer sales histories become available.

However, the existing evidence does not justify claiming that 30+ customers individually evaluated the Product Demand Prediction model or that a formal user study produced a particular satisfaction percentage. Such claims should only be included if supported by collected questionnaires, interviews, usability tests, or recorded stakeholder feedback. Therefore, current validation should distinguish between adoption of the wider Smart-Cost product and formal validation of this individual forecasting component.

For future formal stakeholder validation, users can evaluate factors such as forecast clarity, usefulness for inventory planning, ease of interpreting predicted quantities, confidence in the prediction information, and overall usefulness of the demand dashboard. A structured Likert-scale questionnaire could convert these observations into measurable usability evidence.

Accordingly, the current stakeholder result demonstrates that the forecasting component has been designed and integrated within a system with real operational users, while formal component-specific user-validation results should only be reported after such evidence has been collected.

6.4 Integration results

Integration testing demonstrated that the Product Demand Prediction component could communicate across the principal technical layers required for operational forecasting. The tested integration involved the Smart-Cost interface, Node.js/Express backend,

historical POS data, Python machine-learning service, preprocessing workflow, forecasting models, and forecast response.

The successful request-to-response sequence was:

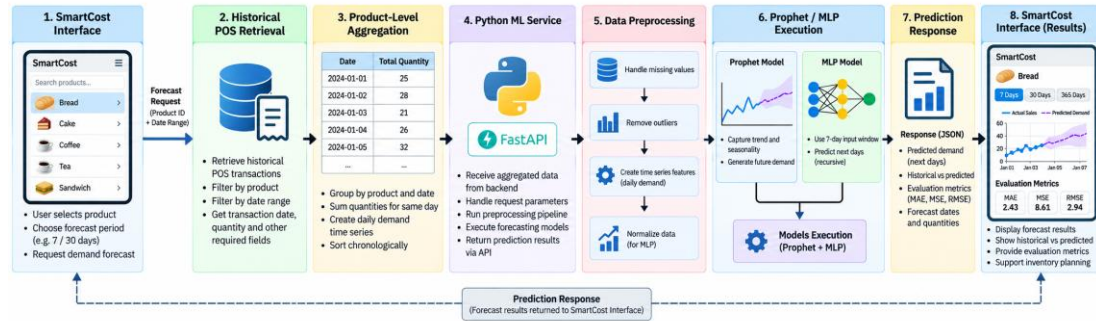


Figure 33: Successful Request-to-Response Execution Sequence of the Product Demand Prediction System

Historical POS records were successfully retrieved and converted into daily product-demand observations. This confirmed that operational transaction information could be transformed into the chronological structure required by the forecasting service.

During integration testing, an important issue was identified in historical date filtering. A request intended to calculate demand for a seven-day period initially produced an unexpectedly large total because the aggregation logic did not correctly restrict records to the requested date range. Explicit start-date and end-date filtering corrected this behaviour. This result demonstrated the importance of validating the data pipeline independently from model accuracy because incorrect historical input can invalidate otherwise correctly implemented forecasts.

A second integration issue occurred when prediction was requested before model training. The system returned a “models not yet trained” condition. The workflow was corrected by ensuring that model training and required model artefacts were available before prediction requests were processed.

Network communication between the application and development services was also verified. Appropriate backend addresses were required when testing from physical

devices or emulators because localhost does not necessarily reference the machine hosting the forecasting service.

Overall, these results confirm that the Product Demand Prediction component can function as an integrated Smart-Cost service rather than only as an independent Python forecasting experiment.

6.5 Summary of findings

The results provide evidence that the principal technical objectives of the Product Demand Prediction research were achieved. First, historical POS transactions could be transformed into product-level daily-demand time series, establishing a consistent input structure for forecasting.

Second, two complementary forecasting approaches were successfully implemented. Prophet provided a time-series forecasting mechanism for learning temporal demand behaviour [2], while the MLP neural network provided a mechanism for learning nonlinear relationships from recent historical observations [5]. The research also established the architecture necessary to use their outputs within a Hybrid Demand Prediction workflow, consistent with the motivation for combining complementary forecasting methods identified in previous research [4], [9].

Third, quantitative evaluation demonstrated that forecasting performance could be measured objectively. In the available preliminary Bread experiment, the MLP achieved $MAE = 2.43$, $MSE = 8.61$, and $RMSE = 2.94$. These measurements confirm that the implemented model generated measurable predictions close to the observed values during the tested seven-day period. Nevertheless, the limited evaluation period prevents a generalized conclusion regarding model superiority.

Fourth, integration testing confirmed that historical Smart-Cost sales information could travel through the complete forecasting pipeline and return as demand predictions to the application. Problems involving date-range aggregation, model availability, and network communication were identified and addressed during implementation.

The overall findings therefore demonstrate the technical feasibility of integrating product-level demand forecasting into Smart-Cost . At the same time, they identify the need for continued evaluation using larger chronological test periods and genuine business sales histories. As additional real-world Smart-Cost data accumulates, Prophet, MLP, and the hybrid approach should be evaluated against identical test periods and across multiple products. This will provide stronger evidence regarding forecasting reliability, model selection, and the practical value of Product Demand Prediction for micro-SME inventory planning.

Table 36:Summary of findings against the research objectives

Objective	Evidence / finding	Status
Prepare product-demand data	POS records were transformed into chronological daily product-demand series.	Achieved.
Implement Prophet	Prophet accepted ds/y data and generated future $\hat{y}$ forecasts.	Achieved.
Implement MLP	Seven-day sliding-window MLP generated product-demand forecasts.	Achieved.
Investigate hybrid forecasting	Prophet and MLP outputs were incorporated into the hybrid forecasting architecture.	Implemented/investigated; superiority not established.

Objective	Evidence / finding	Status
Quantitatively evaluate forecasts	MAE, MSE and RMSE were implemented; Bread MLP produced 2.43, 8.61 and 2.94 respectively.	Achieved at preliminary level.
Integrate with Smart-Cost	Forecasting was connected through historical data, backend, Python ML service and interface.	Achieved technically.
Establish broad real-world predictive validity	Current evidence is limited by synthetic development data and short recorded evaluation.	Requires further validation.

7 DISCUSSION

The findings demonstrate that the Product Demand Prediction component can transform historical Smart-Cost Point-of-Sale (POS) transactions into product-level forecasts through an integrated machine-learning workflow. The research achieved this by preparing chronological daily-demand series and applying two complementary forecasting approaches: Prophet for time-series characteristics and a Multilayer Perceptron (MLP) for nonlinear relationships in recent demand. This combination reflects the wider forecasting literature, which shows that retail-demand behaviour can vary across products and that different forecasting approaches may capture different characteristics of a time series [1], [4], [18].

The preliminary quantitative results provide evidence that the implemented MLP can generate measurable short-term forecasts. For the Bread product, the recorded seven-day evaluation produced an MAE of 2.43, MSE of 8.61, and RMSE of 2.94. These values demonstrate technical forecasting capability, but they should not be interpreted as evidence that the MLP is universally superior. Hyndman and Koehler emphasize that forecast-accuracy measures must be interpreted appropriately when comparing forecasting methods [17]. Furthermore, the current evaluation period is too limited for broad generalization.

An important outcome of the research is that forecasting was implemented as part of an operational software architecture rather than remaining as an independent experimental model. Historical data retrieval, aggregation, preprocessing, forecasting, evaluation, API communication, and result visualization were incorporated into the Smart-Cost environment. This addresses a practical dimension of demand prediction by connecting forecasting research with business decision support.

The Hybrid Demand Prediction approach further provides a basis for utilizing complementary information from Prophet and MLP. Previous studies have demonstrated the potential of hybrid and ensemble forecasting [4], [9]. Nevertheless, the present research does not claim that the hybrid approach is more accurate until

Prophet, MLP, and hybrid forecasts have been evaluated under identical experimental conditions across a larger number of products and dates.

Overall, the findings support the feasibility of integrated Product Demand Prediction while also identifying the need for broader real-world validation.

7.1 Interpretation of findings

The results indicate that historical POS data can provide a practical foundation for product-level demand forecasting when the transaction records are transformed into a consistent chronological representation. The successful conversion of individual sales transactions into daily product-demand series was important because both forecasting models depend on reliable temporal observations.

The MLP results provide preliminary evidence that recent historical demand contains information useful for short-term prediction. For Bread, an MAE of 2.43 means that the predicted quantity differed from the actual quantity by approximately 2.43 units on average during the seven evaluated observations. The RMSE of 2.94 is slightly higher because squared-error-based measures give greater influence to larger individual forecasting errors. Forecasting literature recommends careful interpretation of accuracy measures because different metrics emphasize different aspects of error [17].

The Prophet implementation produced another perspective on the same forecasting problem. Unlike the MLP, which learns nonlinear relationships from a seven-day historical window, Prophet was designed to represent temporal structure within business time series [2]. This distinction is important because the models should not be viewed as identical algorithms competing only for the lowest error. Instead, each represents a different method of learning from historical demand.

The findings also demonstrate why preprocessing and integration quality are as important as model implementation. During development, incorrect date filtering initially caused historical totals to include observations outside the requested period. This illustrates that a technically correct forecasting algorithm can still generate misleading results if the input pipeline is incorrect.

The findings therefore support the use of a multi-model forecasting architecture, but they do not establish a universally superior model. Differences in product-demand behaviour have been recognized in previous research [18], and the most appropriate forecasting method may consequently vary according to the product, available history, and forecasting horizon.

7.2 Comparison with literature baselines and existing solutions

The findings are broadly consistent with previous research demonstrating that retail-demand forecasting can benefit from both statistical/time-series and machine-learning approaches. Fildes, Ma, and Kolassa describe retail forecasting as a complex problem involving heterogeneous products, different data levels, and operational forecasting requirements [1]. The Smart-Cost results reinforce this perspective because the developed component must handle product-specific chronological histories rather than assuming identical behaviour across all products.

The Prophet implementation follows the forecasting direction introduced by Taylor and Letham, who developed Prophet for practical business time-series forecasting [2]. In Smart-Cost, Prophet provides the model responsible for learning temporal behaviour and generating future product-demand estimates.

The MLP implementation is consistent with earlier research showing the applicability of artificial neural networks to forecasting. Zhang, Patuwo, and Hu established neural networks as an important nonlinear forecasting approach [5], while Carbonneau, Laframboise, and Vahidov investigated machine-learning techniques for supply-chain demand forecasting [7]. Doganis *et al.* further demonstrated the relevance of neural-network forecasting for short shelf-life food products [8], a useful comparison for Smart-Cost businesses dealing with bakery or food-related inventory.

The hybrid direction is supported by Aburto and Weber, who investigated hybrid demand forecasting for supply-chain management [9]. Mejía and Aguilar similarly demonstrated the relevance of ensemble forecasting when product categories possess different time-series characteristics [4].

However, a direct numerical claim that Smart-Cost outperforms these published approaches would be inappropriate because the datasets, products, test periods, and experimental protocols are different. The current study also lacks a fully reported naive or seasonal-naïve baseline. Such a baseline should be included in extended validation because a forecasting model should ideally demonstrate improvement over a simple reference prediction before increased model complexity is considered beneficial [15].

7.3 Research contribution

The principal research contribution is the development and evaluation of an integrated Product Demand Prediction framework for the Smart-Cost environment. The contribution does not lie in inventing Prophet, MLP, or general hybrid forecasting. Instead, it lies in adapting these approaches into a structured forecasting pipeline designed around product-level POS information.

The first contribution is a preprocessing strategy that converts transaction-level records into chronological daily-demand series. This establishes a common forecasting representation from operational POS information.

The second contribution is the implementation of two complementary forecasting perspectives. Prophet represents temporal and seasonal characteristics of product demand [2], while the MLP learns nonlinear relationships from recent seven-day demand sequences. The use of complementary approaches is relevant because previous research indicates that product-demand characteristics can vary substantially [4], [18].

The third contribution is the Hybrid Demand Prediction framework, which establishes a mechanism for bringing the Prophet and MLP forecasting paths together. Previous studies have demonstrated the potential of hybrid and ensemble forecasting [4], [9], but the present research investigates this principle specifically within the Smart-Cost Product Demand Prediction environment.

The fourth contribution is the integration of forecasting with the operational software architecture. The research connects POS data retrieval, preprocessing, machine-

learning execution, quantitative evaluation, backend communication, and prediction visualization into a complete workflow.

Finally, the research contributes preliminary empirical evidence from the implemented MLP. The Bread evaluation produced $MAE = 2.43$, $MSE = 8.61$, and $RMSE = 2.94$ over the recorded seven-day evaluation period. These results establish an initial measurable benchmark for the implemented component while leaving broader comparative evaluation as future work.

Accordingly, the research contribution is best characterized as an applied forecasting and system-integration contribution focused on converting operational POS information into product-level predictive decision support.

Table 37: Summary of Research Objectives, Achieved Outcomes, and Remaining Work

Research objective	Achieved outcome	Remaining work
Structured demand-data preparation	Product-level daily demand can be constructed from POS transactions.	Validate across larger real-world Smart-Cost datasets.
Prophet forecasting	Configured Prophet model generates future product-demand estimates.	Evaluate on longer identical test periods across products.
MLP forecasting	Implemented seven-day MLP workflow with recursive multi-step forecasting.	Assess robustness over longer horizons and multiple products.
Hybrid Demand Prediction	Architecture supports complementary Prophet and MLP outputs.	Define/validate combination strategy and compare against individual models.

Research objective	Achieved outcome	Remaining work
Forecast evaluation	MAE, MSE and RMSE are available; preliminary Bread MLP metrics recorded.	Add Prophet, hybrid and simple baseline metrics under equivalent conditions.
Smart-Cost integration	End-to-end request-to-response forecasting workflow is operationally integrated.	Continue monitoring and retraining as genuine data accumulates.

7.4 Product contribution

The product contribution of this research is the addition of a data-driven Product Demand Prediction capability to Smart-Cost . Conventional POS functionality primarily records completed transactions and enables users to examine historical sales. The developed component extends the use of this information by processing historical product demand and generating estimates for future periods.

From a business-user perspective, the component provides a mechanism for selecting a product, retrieving its historical demand, executing forecasting models, and viewing predicted future quantities through Smart-Cost . This creates a practical connection between historical transaction data and forward-looking inventory decisions.

The component also contributes to Smart-Cost 's technical architecture. Forecasting is implemented through a dedicated Python machine-learning service that communicates with the existing Node.js/Express backend. This modular structure separates model processing from general application operations and allows forecasting algorithms to be retrained or enhanced without requiring fundamental changes to the entire POS platform.

Another product contribution is the support for multiple forecasting perspectives. Rather than exposing a single fixed forecasting mechanism, the technical design accommodates Prophet, MLP, and Hybrid Demand Prediction. This provides a foundation for future model-selection strategies in which forecasting behaviour could be adapted according to individual-product characteristics.

The Product Demand Prediction component also benefits from being developed within an existing operational software ecosystem. Smart-Cost is available through mobile application distribution channels, a web version has been developed, and the wider product has established a customer base exceeding 30 customers. This provides an environment in which increasingly large real-world historical sales datasets can become available for future forecasting validation.

The product contribution should nevertheless be distinguished from product-wide adoption. The existence of Smart-Cost customers demonstrates adoption of the broader platform; it does not by itself demonstrate that all customers have evaluated or actively used the Product Demand Prediction functionality. Formal component-specific usage evidence should therefore be reported separately when available.

7.5 Limitations and threats to validity

Several limitations affect the interpretation and generalizability of the current findings. The most significant limitation is data availability. Synthetic POS-style observations were used during development where sufficient long-term real-world product-sales history was unavailable. Synthetic data is useful for verifying preprocessing, model training, API communication, and forecasting workflows, but it cannot fully reproduce genuine customer-demand behaviour.

A second limitation concerns the evaluation period. The recorded Bread MLP results were calculated over seven observations. Although the results demonstrate that the model can produce measurable predictions, such a small test period is insufficient to establish reliable general performance. A longer chronological holdout period should be used for final comparative evaluation.

The current forecasting inputs also exclude several external variables. Product demand may be influenced by promotions, public holidays, price changes, weather, local events, stock-outs, and business closures. Huber and Stuckenschmidt demonstrated the importance of calendar-related special days in daily retail forecasting [3]. Failure to represent such factors can therefore limit predictive accuracy.

Another threat concerns recursive multi-step MLP forecasting. When a predicted value becomes an input to the next forecasting step, prediction errors may propagate across the forecast horizon. Multi-step forecasting strategy has been identified as an important consideration in time-series research [14].

Potential data leakage is another validity concern. Scaling and preprocessing parameters must be derived from training observations rather than the complete dataset. Similarly, chronological splitting should be maintained instead of randomly mixing future observations into training data.

Finally, the available evidence does not yet establish that the hybrid approach statistically outperforms both individual models. A stronger evaluation should compare Prophet, MLP, hybrid, and simple baseline methods across identical test dates, multiple products, and longer real-world periods before drawing generalized conclusions.

Table 38: Current Limitations and Proposed Future Improvements of the Product Demand Prediction System

Current limitation	Effect on research	Proposed future improvement
Synthetic development data	May not reproduce genuine customer-demand behaviour.	Train and validate using longer histories of genuine Smart-Cost POS transactions.

Current limitation	Effect on research	Proposed future improvement
Seven-day recorded MLP evaluation	Too short for broad generalization.	Use longer chronological holdouts and rolling-origin/walk-forward evaluation.
Limited external variables	Demand effects from promotions, holidays, weather, price and stock availability may be missed.	Add relevant explanatory variables where reliable data is available.
Recursive MLP forecasting	Errors can propagate across multi-day horizons.	Compare direct/multi-output strategies and monitor error by forecast horizon.
No complete comparable Prophet/hybrid metrics	Model superiority cannot currently be established.	Evaluate Prophet, MLP and hybrid on identical dates and products.
No simple baseline reported	Cannot establish whether model complexity improves over a simple reference.	Add naive and seasonal-naive forecasts.
One-year synthetic history for annual effects	Limited evidence for robust yearly seasonal learning.	Use multi-year real-world history before drawing strong annual-seasonality conclusions.

Current limitation	Effect on research	Proposed future improvement
Static model behaviour	Demand patterns may change after deployment.	Introduce continuous monitoring, periodic retraining and performance-based model selection/weighting.

7.6 Implications

The research has implications for both micro-SME decision support and future demand-forecasting development within Smart-Cost . At the operational level, the component demonstrates how transaction data already generated by a POS system can be reused for predictive purposes. This reduces the need for a completely separate forecasting data-collection process because historical product sales provide the primary input.

For business users, demand predictions can provide an additional source of information when deciding how much stock to purchase or prepare. This is particularly relevant to businesses dealing with short shelf-life products, where forecasting has practical implications for balancing product availability against potential wastage [8]. Nevertheless, predictions should support rather than replace managerial judgement because future demand can be influenced by events that are not represented in historical data.

The findings also have implications for model selection. Different products may exhibit different demand characteristics [18]. Smart-Cost should therefore avoid assuming that one forecasting algorithm must permanently serve every product. Continued evaluation could allow future versions of the component to select or weight models according to historical performance for individual products.

The deployed Smart-Cost environment creates an additional research opportunity. As genuine POS histories accumulate across operational users, the forecasting models can

be retrained and evaluated using increasingly representative datasets. This can enable evaluation across different business types, products, seasonal periods, and forecasting horizons.

Future development should also investigate explanatory variables such as promotions, holidays, price changes, and stock availability. Calendar information has previously been shown to be relevant to daily retail-demand forecasting [3]. Longer chronological validation, naïve and seasonal-naïve baselines, rolling-origin evaluation, and systematic comparison of Prophet, MLP, and hybrid forecasts would further strengthen the evidence.

Therefore, the broader implication of the research is that Product Demand Prediction can evolve from a fixed forecasting feature into a continuously evaluated decision-support capability, improving as Smart-Cost accumulates additional real-world sales history and operational evidence.

8 CONCLUSION AND FUTURE DIRECTIONS

This research designed, implemented, and evaluated the Product Demand Prediction component of Smart-Cost , with the objective of transforming historical Point-of-Sale (POS) transactions into future product-demand estimates that can support inventory and operational decision-making. The study addressed the complete prediction workflow rather than treating forecasting as an isolated machine-learning experiment. Historical transactions were aggregated into chronological daily-demand series, processed for model training, forecast using complementary algorithms, quantitatively evaluated, and integrated into the Smart-Cost application environment.

Two principal forecasting approaches were investigated: Prophet and a Multilayer Perceptron (MLP) neural network. Prophet was selected for its ability to model business time-series behaviour and recurring temporal patterns [2], while the MLP was implemented to learn nonlinear relationships from recent product-demand observations, consistent with established neural-network forecasting research [5]. A Hybrid Demand Prediction approach was additionally investigated to provide a framework for utilizing the complementary characteristics of both forecasting methods. This direction is supported by previous research demonstrating the potential of hybrid and ensemble approaches for demand forecasting [4], [9].

The implemented MLP produced preliminary measurable results for the Bread product, recording an MAE of 2.43, MSE of 8.61, and RMSE of 2.94 over the available seven-day evaluation period. These results confirm that the implemented workflow can generate quantitatively assessable demand forecasts. However, they are treated as preliminary evidence rather than proof of general forecasting superiority because broader chronological testing is required [17].

The research also demonstrated successful technical integration between historical POS information, the Node.js/Express backend, Python machine-learning service, forecasting models, and Smart-Cost interface. Therefore, the principal outcome is an end-to-end Product Demand Prediction capability capable of converting operational sales data into forward-looking information.

Future work should prioritize larger real-world datasets, longer chronological evaluation periods, additional explanatory variables, stronger forecasting baselines, systematic hybrid-model evaluation, and continuous model retraining as additional Smart-Cost sales histories become available.

8.1 Conclusion

The Product Demand Prediction research demonstrates the technical feasibility of using historical Smart-Cost POS information to generate product-level future-demand estimates. The study began from the practical problem that micro and small enterprises must make inventory and product-planning decisions before actual future demand is known. Historical transaction data provides valuable evidence of previous demand, but this information must be systematically processed before it can support predictive decision-making.

The developed solution converts transaction-level records into chronological daily-demand series and applies Prophet and MLP forecasting. Prophet provides a structured method for modelling temporal behaviour within business time series [2]. The MLP provides a complementary machine-learning approach capable of learning nonlinear relationships from recent observations [5]. These different characteristics motivated the inclusion of a Hybrid Demand Prediction approach, consistent with literature demonstrating the potential of combining forecasting methods [4], [9].

Preliminary quantitative evaluation confirmed that the implemented MLP can generate measurable forecasts. For the recorded Bread experiment, the model achieved $MAE = 2.43$, $MSE = 8.61$, and $RMSE = 2.94$ across the seven-day evaluation period. Although these values provide evidence that the forecasting pipeline operates successfully, the limited evaluation period does not justify a general claim that the MLP or hybrid approach is superior to other forecasting methods. Forecast performance must be interpreted in relation to the dataset, product, evaluation horizon, and selected accuracy measures [17].

An important conclusion of the study is that successful Product Demand Prediction depends on more than model selection. Reliable date filtering, product-level

aggregation, chronological preprocessing, model availability, API communication, and correct presentation of results are all necessary for an operational forecasting capability.

The research therefore concludes that Product Demand Prediction can be practically incorporated into Smart-Cost as a data-driven decision-support capability, while continued validation using genuine business data is required to establish its long-term forecasting effectiveness.

8.2 Achievement of objectives

The principal objective of the research was to design, develop, evaluate, and integrate a Product Demand Prediction capability capable of using historical POS information to estimate future product demand. Based on the completed implementation and available experimental evidence, the principal technical objectives were achieved.

The first objective was to establish a structured product-demand data preparation process. This was achieved by retrieving product-sales information, grouping transactions according to product and date, aggregating quantities into daily demand, and chronologically organizing the resulting observations.

The second objective was to develop a Prophet-based forecasting model. This objective was achieved through the preparation of ds and y observations, configuration of relevant seasonal components, model training, and generation of future y hat predictions. Prophet was selected based on its suitability for practical business time-series forecasting [2].

The third objective was to implement an MLP neural-network model. Historical demand was converted into seven-day sliding windows, normalized, and supplied to an MLP containing 64- and 32-neuron hidden layers. Neural-network forecasting literature provides the methodological basis for investigating nonlinear relationships in historical observations [5].

The fourth objective concerned Hybrid Demand Prediction. The implemented architecture established the processing required to use Prophet and MLP forecasts as complementary prediction sources. This objective was motivated by hybrid and ensemble forecasting research [4], [9]. However, a claim that the hybrid model statistically outperforms both individual models requires further comparative evaluation.

The fifth objective was quantitative evaluation using MAE, MSE, and RMSE, which was successfully implemented. Forecast-accuracy measures provide an objective basis for model assessment [17].

Finally, the forecasting workflow was integrated into the Smart-Cost technical environment. Therefore, the research achieved its core development objectives while identifying broader real-world validation as an area requiring continued work.

8.3 Individual research and product contributions

The individual research contribution is the design, implementation, evaluation, and integration of the Product Demand Prediction component within Smart-Cost . The contribution should be distinguished from the development of the underlying Prophet or neural-network algorithms themselves, which are established forecasting techniques [2], [5]. The research contribution lies in adapting and combining these methods within a product-level POS forecasting workflow.

The first contribution is the design of a data-processing pipeline that transforms Smart-Cost POS transactions into chronological daily-demand series. This establishes the connection between operational sales information and forecasting-model requirements.

The second contribution is the implementation of two complementary forecasting paths. Prophet was implemented to represent temporal and seasonal characteristics, while the MLP was designed to learn nonlinear relationships from seven-day historical demand windows. This multi-model structure responds to the observation that products can exhibit different demand patterns [18].

The third research contribution is the Hybrid Demand Prediction framework, which provides a mechanism for utilizing information generated by both Prophet and MLP. Previous research has demonstrated the potential of hybrid and ensemble demand forecasting [4], [9], while this study applies that principle within the Smart-Cost product-demand environment.

The fourth contribution is the implementation of a quantitative evaluation mechanism using MAE, MSE, and RMSE. The preliminary Bread experiment provides an initial measurable result of 2.43 MAE, 8.61 MSE, and 2.94 RMSE.

From a product perspective, the major contribution is converting historical Smart-Cost sales records into forward-looking decision-support information. The forecasting functionality was connected to the existing application architecture through the Node.js/Express backend and Python machine-learning service. This allows predictions to be generated and delivered through the Smart-Cost user environment rather than remaining as standalone experimental outputs.

Overall, the individual contribution combines forecasting research, machine-learning implementation, experimental evaluation, and production-oriented software integration.

8.4 Future directions

Future development should initially focus on real-world data validation. Synthetic POS-style data was useful during development, but forecasting performance should be evaluated using longer histories of genuine Smart-Cost transactions. As the deployed Smart-Cost platform accumulates additional sales records, these observations can provide increasingly representative datasets for model training and evaluation.

A second direction is stronger comparative evaluation. Prophet, MLP, and Hybrid Demand Prediction should be evaluated using identical chronological test periods across multiple products. Simple naïve and seasonal-naïve forecasts should also be included as baselines. This would establish whether the additional complexity of machine-learning and hybrid approaches produces meaningful forecasting

improvements. Forecasting literature emphasizes that model performance should be established through appropriate empirical comparisons rather than assumed from model complexity [15].

Future evaluation should also use longer test periods and rolling-origin or walk-forward validation. This would provide stronger evidence than a single seven-day test and allow performance to be examined across changing demand conditions. Accuracy should continue to be assessed using appropriate measures such as MAE and RMSE [17].

Another direction is the incorporation of external explanatory variables. The present implementation primarily learns from historical product quantities. Future models could incorporate public holidays, promotions, prices, stock availability, weather conditions, and special events where appropriate. Previous research has demonstrated the relevance of calendar-related special days to daily retail-demand forecasting [3].

The hybrid approach can also be extended beyond a fixed combination mechanism. Future research could investigate performance-based dynamic weighting, where the contribution of Prophet and MLP changes according to recent validation accuracy or individual-product behaviour. The relevance of product-specific demand characteristics and ensemble approaches has been demonstrated in previous research [4], [18].

Finally, the operational system can support continuous model monitoring and retraining. As new POS transactions become available, prediction errors can be monitored and models periodically updated. This would allow Product Demand Prediction to evolve from a static forecasting feature into an adaptive Smart-Cost decision-support capability that improves as additional real-world evidence becomes available.

REFERENCES

- [1] R. Fildes, S. Ma, and S. Kolassa, “Retail forecasting: Research and practice,” *International Journal of Forecasting*, vol. 38, no. 4, pp. 1283–1318, 2022, doi: 10.1016/j.ijforecast.2019.06.004. [ScienceDirect](#)
- [2] S. J. Taylor and B. Letham, “Forecasting at scale,” *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018, doi: 10.1080/00031305.2017.1380080. [IDEAS/RePEc](#)
- [3] J. Huber and H. Stuckenschmidt, “Daily retail demand forecasting using machine learning with emphasis on calendric special days,” *International Journal of Forecasting*, vol. 36, no. 4, pp. 1420–1438, 2020, doi: 10.1016/j.ijforecast.2020.02.005. [DOI](#)
- [4] S. Mejía and J. Aguilar, “A demand forecasting system of product categories defined by their time series using a hybrid approach of ensemble learning with feature engineering,” *Computing*, vol. 106, pp. 3945–3965, 2024, doi: 10.1007/s00607-024-01320-y. [Springer Link](#)
- [5] G. Zhang, B. E. Patuwo, and M. Y. Hu, “Forecasting with artificial neural networks: The state of the art,” *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998, doi: 10.1016/S0169-2070(97)00044-7. [ScienceDirect](#)
- [6] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, “An empirical comparison of machine learning models for time series forecasting,” *Econometric Reviews*, vol. 29, nos. 5–6, pp. 594–621, 2010, doi: 10.1080/07474938.2010.481556. [Taylor & Francis Online](#)
- [7] R. Carbonneau, K. Laframboise, and R. Vahidov, “Application of machine learning techniques for supply chain demand forecasting,” *European Journal of Operational Research*, vol. 184, no. 3, pp. 1140–1154, 2008, doi: 10.1016/j.ejor.2006.12.004. [ScienceDirect](#)
- [8] P. Doganis, A. Alexandridis, P. Patrinos, and H. Sarimveis, “Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing,” *Journal of Food Engineering*, vol. 75, no. 2, pp. 196–204, 2006, doi: 10.1016/j.jfoodeng.2005.03.056. [ScienceDirect](#)

- [9] L. Aburto and R. Weber, “Improved supply chain management based on hybrid demand forecasts,” *Applied Soft Computing*, vol. 7, no. 1, pp. 136–144, 2007, doi: 10.1016/j.asoc.2005.06.001. [ScienceDirect](#)
- [10] R. V. Joseph, A. Mohanty, S. Tyagi, S. Mishra, S. K. Satapathy, and S. N. Mohanty, “A hybrid deep learning framework with CNN and Bi-directional LSTM for store item demand forecasting,” *Computers & Electrical Engineering*, vol. 103, Art. no. 108358, 2022, doi: 10.1016/j.compeleceng.2022.108358. [ScienceDirect](#)
- [11] Z. H. Kilimci, A. O. Akyuz, M. Uysal, S. Akyokus, M. O. Uysal, B. Atak Bulbul, and M. A. Ekmis, “An improved demand forecasting model using deep learning approach and proposed decision integration strategy for supply chain,” *Complexity*, vol. 2019, Art. no. 9067367, 2019, doi: 10.1155/2019/9067367. [Wiley Online Library](#)
- [12] A. L. Loureiro, V. L. Miguéis, and L. F. da Silva, “Exploring the use of deep neural networks for sales forecasting in fashion retail,” *Decision Support Systems*, vol. 114, pp. 81–93, 2018, doi: 10.1016/j.dss.2018.08.010. [ScienceDirect](#)
- [13] S. F. Crone, M. Hibon, and K. Nikolopoulos, “Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction,” *International Journal of Forecasting*, vol. 27, no. 3, pp. 635–660, 2011, doi: 10.1016/j.ijforecast.2011.04.001. [ScienceDirect](#)
- [14] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, “A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition,” *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012, doi: 10.1016/j.eswa.2012.01.039. [ScienceDirect](#)
- [15] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward,” *PLoS ONE*, vol. 13, no. 3, Art. no. e0194889, 2018, doi: 10.1371/journal.pone.0194889. [PLOS](#)
- [16] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The M4 Competition: Results, findings, conclusion and way forward,” *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018, doi: 10.1016/j.ijforecast.2018.06.001. [IDEAS/RePEc](#)

- [17] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006, doi: 10.1016/j.ijforecast.2006.03.001. [DOI](#)
- [18] A. A. Syntetos, J. E. Boylan, and J. D. Croston, “On the categorization of demand patterns,” *Journal of the Operational Research Society*, vol. 56, no. 5, pp. 495–503, 2005, doi: 10.1057/palgrave.jors.2601841. [Taylor & Francis Online](#)
- [19] J. W. Taylor, “Forecasting daily supermarket sales using exponentially weighted quantile regression,” *European Journal of Operational Research*, vol. 178, no. 1, pp. 154–167, 2007, doi: 10.1016/j.ejor.2006.02.006. [ScienceDirect](#)
- [20] R. Fildes, S. Kolassa, and S. Ma, “Post-script—Retail forecasting: Research and practice,” *International Journal of Forecasting*, vol. 38, no. 4, pp. 1319–1324, 2022, doi: 10.1016/j.ijforecast.2021.09.012.

APPENDICES

The appendices provide supplementary evidence relating specifically to the Product Demand Prediction component of Smart-Cost . They contain supporting information on the experimental dataset, preprocessing workflow, model configurations, preliminary evaluation results, technical integration, and academic integrity and AI-use disclosure. Essential findings and interpretations remain in the main thesis.

Appendix A Research instruments or extended results

Appendix A .1 Product Demand Prediction Experimental Dataset

The Product Demand Prediction component used historical-style POS demand records to develop and test the forecasting workflow. During development and experimentation, synthetic datasets were used to provide controlled product-level demand histories where sufficient long-term real-world sales records were not available. Synthetic observations were used for development and technical validation and are not presented as equivalent to genuine customer-demand records.

Table 39:Structure and Description of the Product Demand Prediction Dataset

Field	Description
ds	Date associated with the demand observation
product_name	Product being forecast
y	Total quantity demanded on the corresponding date

The development dataset contained daily observations for multiple products, including Bread, Cake, Coffee, Tea, Bun, Sandwich, Pastry, Juice, Muffin, and Cookie. These records supported verification of preprocessing, model training, forecasting, and integration procedures.

Appendix A .2 Data Preprocessing Workflow

The preprocessing procedure converted POS sales records into model-ready chronological time-series data. The operational sequence was: POS Sales Records → Product Filtering → Date-Range Filtering → Daily Quantity Aggregation →

Chronological Sorting → Data Validation → Model-Specific Transformation → Forecasting Input.

For Prophet, prepared observations were transformed into the required ds and y structure. For the MLP model, chronological observations were transformed into seven-day sliding historical windows so that recent demand formed the input for predicting the subsequent demand value. Normalization was applied for the MLP workflow. For formal evaluation, preprocessing parameters should be learned from training observations and then applied to unseen test observations to reduce the risk of information leakage.

Table 40:Seven-Day Sliding-Window Input and Prediction Target Structure for the MLP Forecasting Model

Input Window	Prediction Target
Day 1 – Day 7	Day 8
Day 2 – Day 8	Day 9
Day 3 – Day 9	Day 10

Appendix A .3 Prophet Model Configuration

Prophet was implemented as one of the two principal forecasting approaches. The model accepted chronological ds and y observations and generated yhat as the central future-demand estimate together with yhat_lower and yhat_upper. Prophet was selected because it provides a structured approach for practical business time-series forecasting [2].

Table 41:Prophet Forecasting Model Hyperparameter Configuration

Parameter	Configuration
Daily seasonality	False
Weekly seasonality	True
Yearly seasonality	True

Changepoint prior scale	0.05
Seasonality prior scale	10.0
Additional monthly seasonality	30.5-day period
Monthly Fourier order	5

Appendix A .4 MLP Model Configuration

The neural-network forecasting component was implemented using a Multilayer Perceptron (MLP). Although an earlier development class name referred to LSTMForecaster, the implemented algorithm uses MLPRegressor and is therefore reported in this thesis as an MLP rather than an LSTM. Neural-network forecasting provides a method for learning nonlinear relationships from historical observations [5], [6].

Table 42:MLP Forecasting Model Architecture and Hyperparameter Configuration

Parameter	Configuration
Model	MLPRegressor
Historical lookback	7 observations
Hidden layer 1	64 neurons
Hidden layer 2	32 neurons
Activation function	ReLU
Optimizer / solver	Adam
Maximum iterations	200
Early stopping	Enabled
Multi-step strategy	Recursive prediction

For multi-day forecasting, each newly predicted value was incorporated into the historical input sequence before the next prediction was generated. This enabled a sequence of future product-demand estimates to be produced from the trained model.

Appendix A .5 Hybrid Demand Prediction

The Product Demand Prediction architecture was designed to support the combination of Prophet and MLP outputs. The conceptual sequence was: Historical Demand → Prophet Forecast + MLP Forecast → Hybrid Prediction → Final Product Demand Forecast. The motivation was to exploit complementary forecasting characteristics rather than depend exclusively on one algorithm. Hybrid and ensemble forecasting approaches have been investigated for demand forecasting in previous research [4], [9].

The thesis does not claim that the Hybrid Demand Prediction approach is statistically superior to both individual models unless such superiority is demonstrated through comparative evaluation using identical held-out observations.

Appendix A .6 Extended Preliminary Evaluation Results

Forecast accuracy was evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These measures provide complementary views of forecasting error and should be interpreted with regard to the scale and characteristics of the evaluated demand series [17].

Table 43:Preliminary MLP Forecasting Model Evaluation Results for Bread

Evaluation Item	Recorded Result
Product	Bread
Model	MLP
Evaluation period	7 days
MAE	2.43
MSE	8.61
RMSE	2.94

The MAE indicates that the predictions differed from the observed quantities by approximately 2.43 units on average during this limited evaluation. The RMSE of 2.94 units gives greater influence to larger individual errors. These measurements represent preliminary component-level evidence and should not be generalized to all Smart-Cost products. Forecasting performance can vary according to product characteristics, historical data availability, demand pattern, and prediction horizon [1], [18].

Appendix A .7 Product Demand Prediction Integration Evidence

The implemented request-to-response sequence was: Smart-Cost Interface → Forecast Request → Historical POS Retrieval → Product-Level Aggregation → Python ML Service → Preprocessing → Prophet/MLP Execution → Prediction Response → Smart-Cost Interface.

Table 44: Technology Stack and Functional Layers of the Smart-Cost Product Demand Prediction System

Layer	Technology / Function
User interface	Smart-Cost application
Application backend	Node.js / Express
Historical data	Smart-Cost POS / MongoDB
ML service	Python / FastAPI
Time-series forecasting	Prophet
Neural forecasting	MLP
API response	Forecast values and associated information
Presentation	Product-demand results through Smart-Cost

Integration testing identified practical issues involving date-range filtering, model availability, and network communication between application and backend services. Correcting these issues strengthened the end-to-end forecasting workflow and

confirmed that the prediction component could operate through the Smart-Cost technical architecture.

Appendix A .8 Technical Evidence to Include

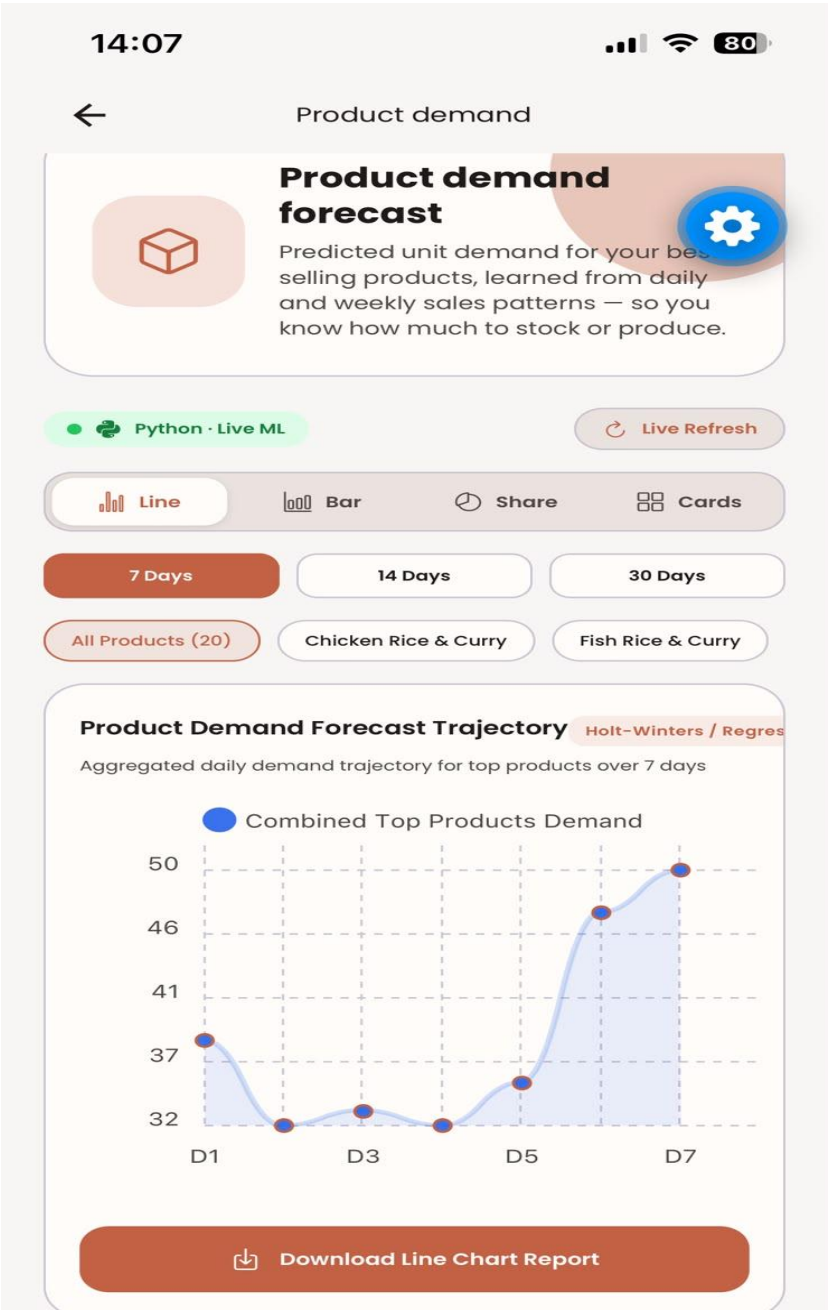


Figure 34:SmartCost Mobile Application Interface Displaying Seven-Day Product Demand Forecast Trends

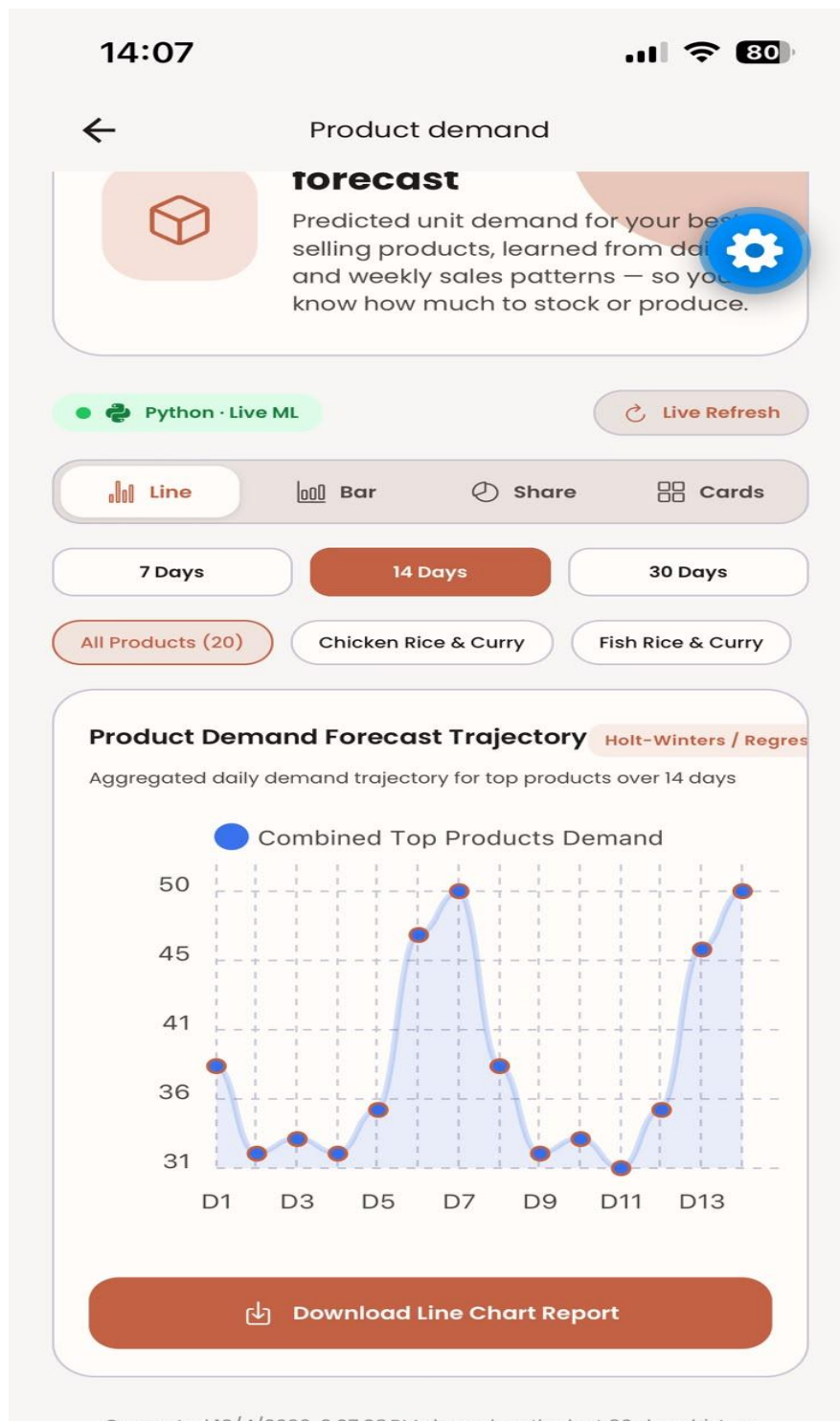


Figure 35: SmartCost Mobile Application Interface Displaying Fourteen-Day Product Demand Forecast Trends

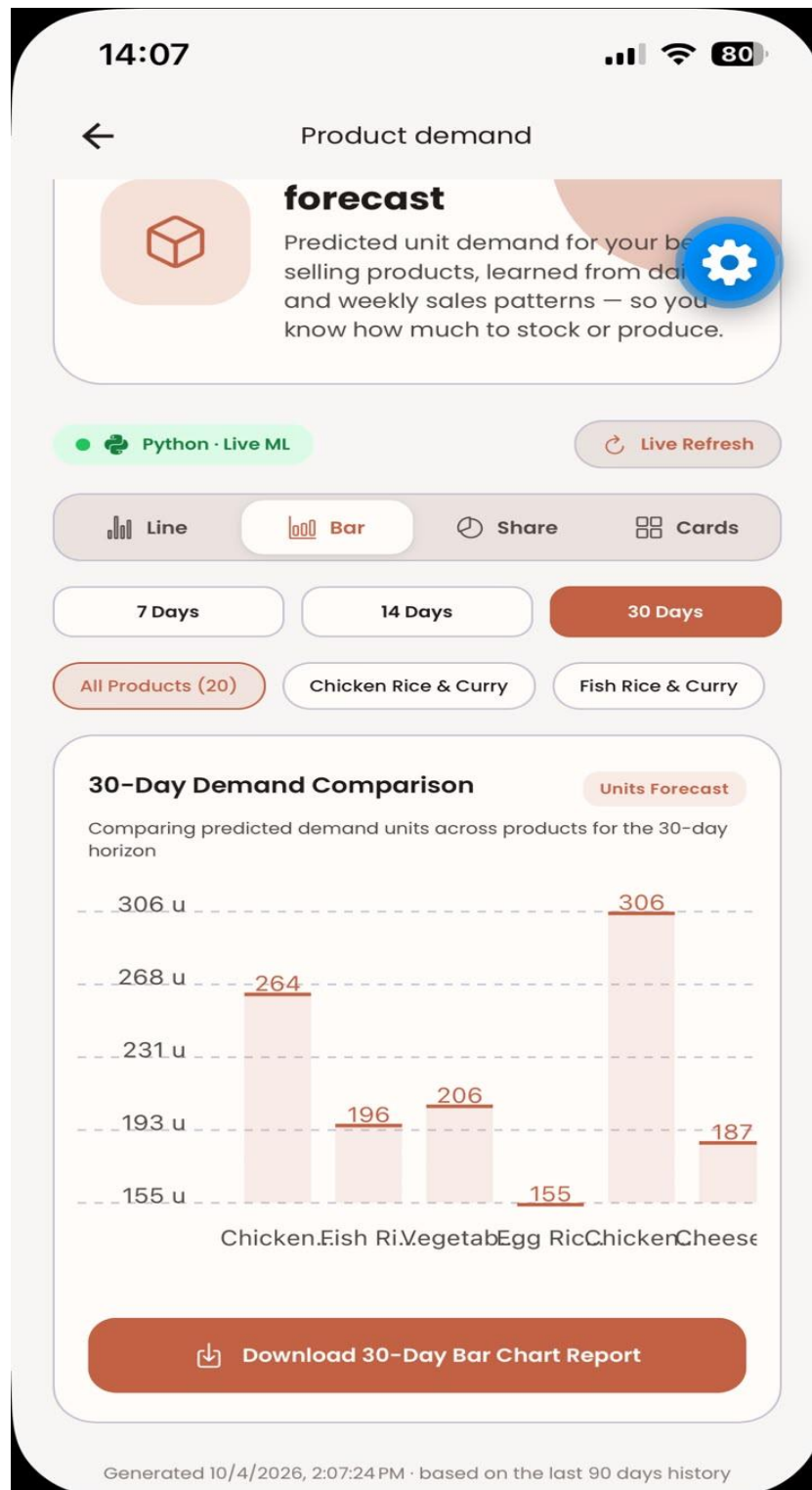


Figure 36: SmartCost Mobile Application Interface Displaying Thirty-Day Product Demand Forecast Comparison

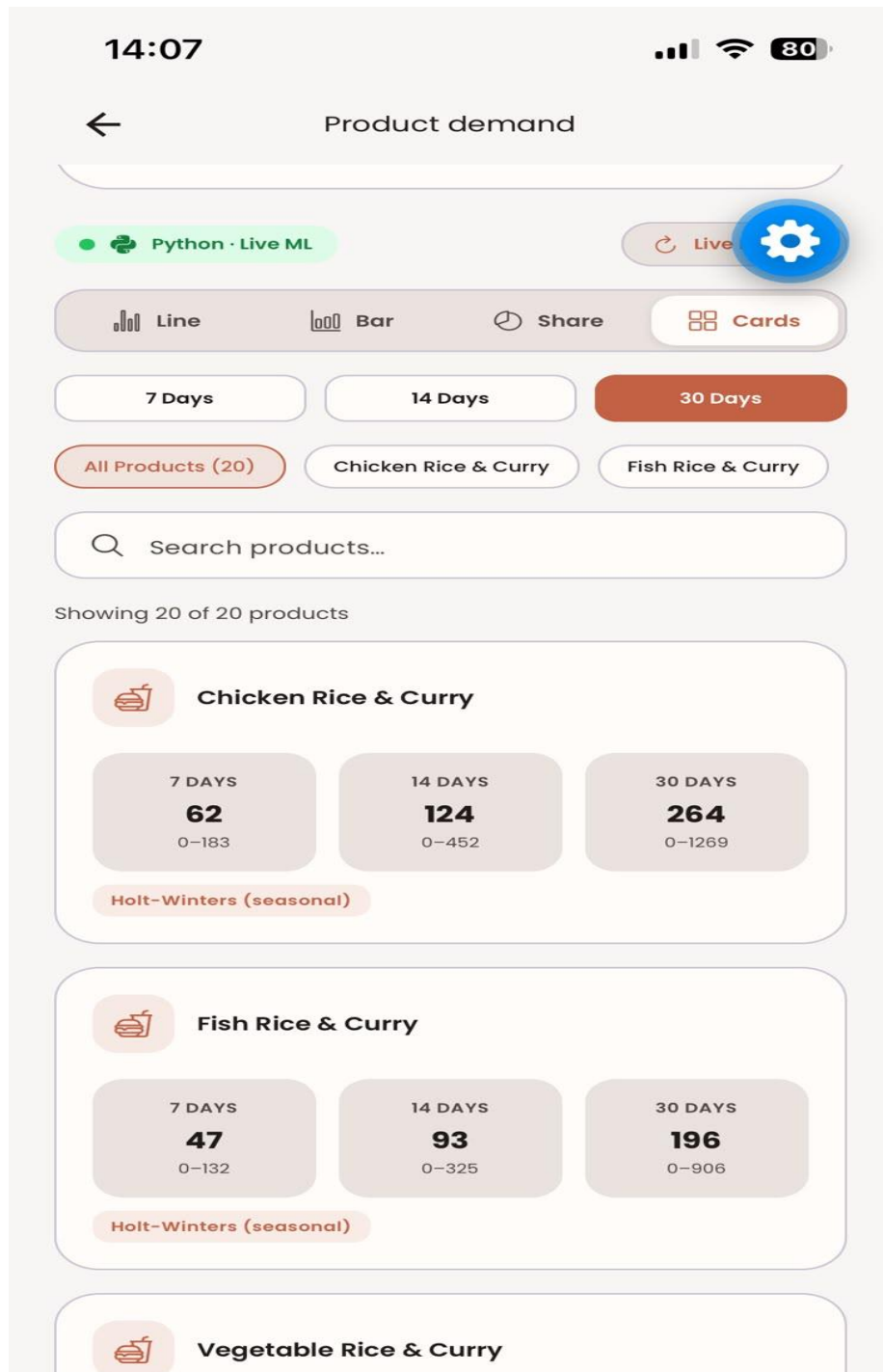


Figure 37: SmartCost Mobile Application Interface Displaying Product-Wise Demand Forecasts for 7, 14, and 30 Days

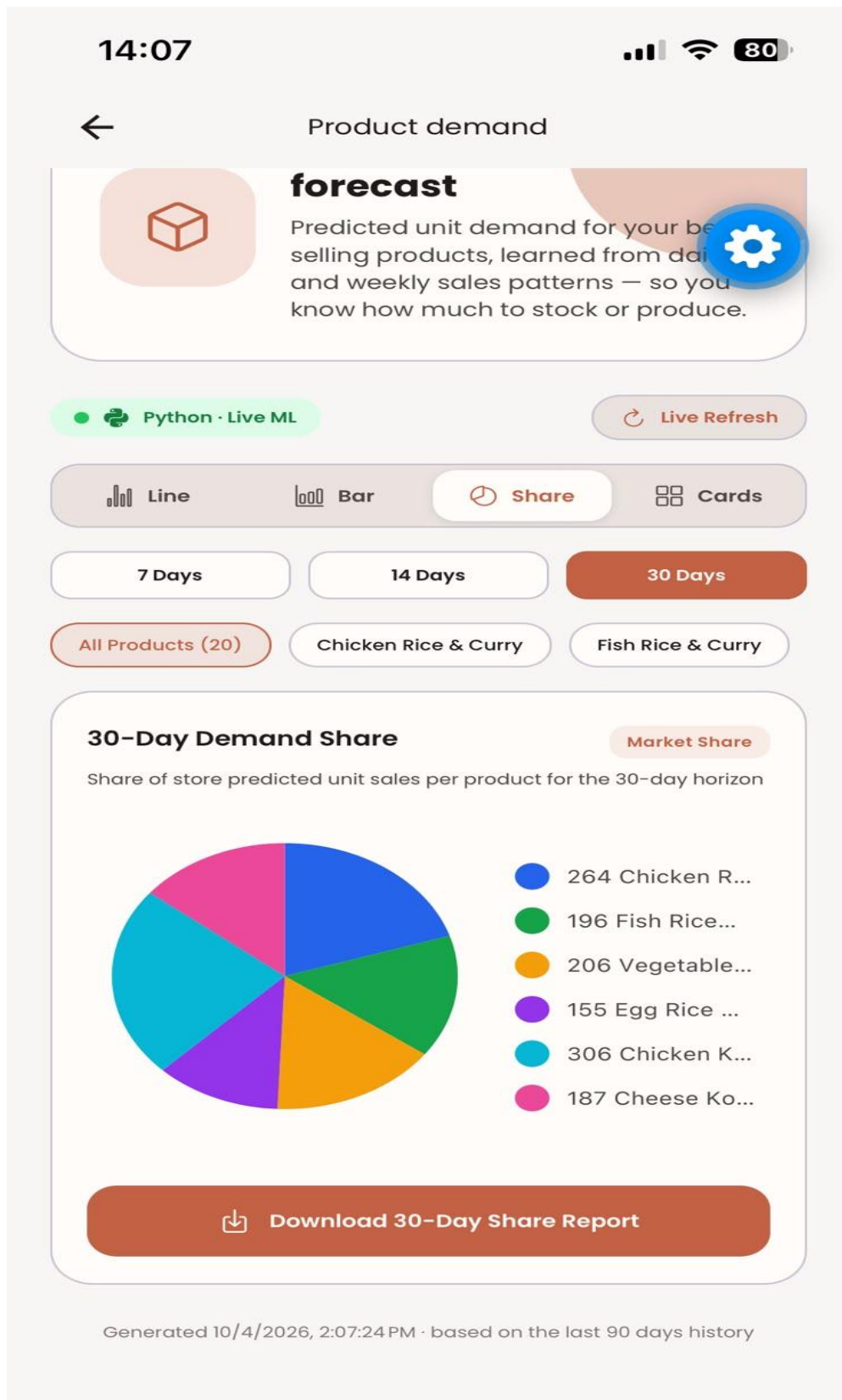


Figure 38: SmartCost Mobile Application Interface Displaying Thirty-Day Product Demand Share Distribution

Appendix A .9 Additional Test Cases

Table 45: Functional Test Cases and Expected Results for the Product Demand Prediction System

Test Case	Expected Result
Select a valid product	The selected product is accepted for demand forecasting.
Retrieve historical demand	Historical records are filtered for the selected product and date range.
Aggregate daily demand	Quantities for the same product and date are summed correctly.
Run Prophet forecast	Future $\hat{y}$ values are returned for the requested period.
Run MLP forecast	Future product-demand values are generated from the seven-day input window.
Request prediction before model training	The system returns or handles the model-not-trained condition.
Apply a seven-day historical filter	Only records within the requested seven-day range are aggregated.
Return forecast through API	The prediction response is returned to the Smart-Cost application.

Appendix B Academic integrity and AI use disclosure

Appendix B .1 Academic Integrity Declaration

This thesis reports the individual work completed for the Product Demand Prediction component of the Smart-Cost research project. The individual contribution included preparation of product-demand data, development and configuration of forecasting models, implementation of the forecasting workflow, experimental evaluation,

technical integration, testing, analysis of results, and preparation of the associated research documentation.

Published research and external technical material used to establish the theoretical and methodological background have been acknowledged through IEEE citations and included in the reference list. Experimental results reported in the thesis are distinguished from findings reported by previous researchers.

Appendix B .2 Generative AI Use Disclosure

Generative artificial-intelligence tools were used as supporting tools during the preparation of this research documentation. Their use did not replace the implementation, experimentation, testing, interpretation, or verification required for the Product Demand Prediction component.

AI assistance was used for activities such as improving the organization and clarity of technical explanations, reviewing academic language, supporting the presentation of methodology, suggesting structures for figures and tables, and assisting with documentation of the implemented forecasting workflow.

Table 46:Summary of AI-Assisted Activities and Verification Procedures in Thesis Preparation

AI-Supported Activity	Purpose	Verification Performed
Academic-language assistance	Improve clarity and readability	Reviewed against the implemented system
Thesis structuring	Organize research material into required sections	Checked against thesis requirements
Technical explanation	Explain Prophet, MLP, preprocessing and evaluation	Checked against implementation and cited literature

Diagram preparation	Visualize the Product Demand Prediction workflow	Compared with actual system architecture
Table preparation	Present configurations and experimental results	Values checked against recorded results
Literature organization	Position relevant forecasting research	References and claims reviewed before inclusion

Appendix B .3 Verification of AI-Assisted Material

AI-assisted material was reviewed before inclusion in the thesis. Technical descriptions were checked against the implemented Product Demand Prediction workflow, while literature-based claims were associated with the appropriate references. Particular attention was given to model terminology: the implemented neural-network model uses MLPRegressor and is therefore identified as a Multilayer Perceptron (MLP), not an LSTM.

The preliminary Bread results—MAE 2.43, MSE 8.61, and RMSE 2.94—were treated as limited experimental evidence rather than generalized to all products or forecasting conditions. No AI-generated numerical result was intentionally presented as an empirical research finding.