



IMPLEMENTING A PREDICTIVE MODEL TO HELP RETAILERS FORECAST SALES AND DEMAND FOR THEIR PRODUCTS BASED ON HISTORICAL DATA AND FUTURE TRENDS

**A dissertation submitted in partial fulfillment of the requirements for
the award of the Degree of**

Bachelor of Technology

In

Computer Science and Engineering (Data Science)

By

ARUGOLANU PRABHASH RAJ (23U61A6703)

Under the guidance of

Mrs. Noore Ilahi

B. Tech., M. Tech.

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

(Approved by AICTE, New Delhi & Affiliated to JNTUH)

(Recognized under section 2(f) of UGC Act 1956)

An ISO:9001-2015 Certified Institution

CHILKUR (V), MOINABAD (M), R.R. DIST. T.S-501504

June 2025



(Approved by AICTE & Affiliated to JNTUH)
(Recognized under Section 2(f) of UGC Act 1956)

An ISO:9001-2015 Certified Institution

Survey No. 179, Chilkur (V), Moinabad (M), Ranga Reddy Dist. TS.

JNTUH Code (U6) ECE –EEE-CSD-CSM – CSE - CIVIL – ME – MBA - M.Tech EAMCET Code -

(GLOB)

Department of Computer Science and Engineering

Noore Ilahi

B. Tech., M. Tech.

Assistant Professor & Head

Date: 02-06-2025

CERTIFICATE

This is to certify that the project work entitled “**IMPLEMENTING A PREDICTIVE MODEL TO HELP RETAILERS FORCAST SALES AND DEMAND FOR THEIR PRODUCTS BASED ON HISTORICAL DATA AND FUTURE TRENDS**”, is a bonafide work of **Arugolanu Prabhash Raj (HT.No:23U61A6703)**, submitted in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering (Data Science)** during the academic year 2024-25. This is further certified that the work done under my guidance, and the results of this work have not been submitted elsewhere for the award of any other degree or diploma.

Internal Guide

Mrs. Noore Ilahi
Assistant Professor

Head of the Department

Mrs. Noore Ilahi
Assistant Professor

DECLARATION

I hereby declare that the project work entitled **IMPLEMENTING A PREDICTIVE MODEL TO HELP RETAILERS FORCAST SALES AND DEMAND FOR THEIR PRODUCTS BASED ON HISTORICAL DATA AND FUTURE TRENDS** , submitted to **Department of Computer Science and Engineering (Data Science), Global Institute of Engineering & Technology, Moinabad**, affiliated to **JNTUH, Hyderabad** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (Data Science)** is the work done by me and has not been submitted elsewhere for the award of any degree or diploma.

ARUGOLANU PRABHASH RAJ (23U61A6703)

ACKNOWLEDGEMENT

I am thankful to my guide **Mrs. Noore Ilahi**, Assistant Professor of CSE(DS) Department for her valuable guidance for successful completion of this project.

I express my sincere thanks to **Mrs. G. Pavani**, Project Coordinator for giving me an opportunity to undertake the project “**IMPLEMENTING A PREDICTIVE MODEL TO HELP RETAILERS FORECAST SALES AND DEMAND FOR THEIR PRODUCTS BASED ON HISTORICAL DATA AND FUTURE TRENDS**” and for enlightening me on various aspects of my project work and assistance in the evaluation of material and facts. She not only encouraged me to take up this topic but also given her valuable guidance in assessing facts and arriving at conclusions.

I am also most obliged and grateful to **Mrs. Noore Ilahi**, Assistant Professor and Head, Department of CSE(DS) for giving me guidance in completing this project successfully.

I express my heart-felt gratitude to our Vice-Principal **Prof. Dr. G Ahmed Zeeshan**, Coordinator Internal Quality Assurance Cell (IQAC) for his constant guidance, cooperation, motivation and support which have always kept me going ahead. I owe a lot of gratitude to him for always being there for me.

I also most obliged and grateful to our Principal **Dr. P. Raja Rao** for giving me guidance in completing this project successfully.

I also thank my parents for their constant encourage and support without which the project would have not come to an end.

Last but not the least, I would also like to thank all my class mates who have extended their cooperation during our project work.

ARUGOLANU PRABHASH RAJ (23U61A6703)

VISION

The Vision of the Department is to produce professional Computer Science Engineers who can meet the expectations of the globe and contribute to the advancement of engineering and technology which involves creativity and innovations by providing an excellent learning environment with the best quality facilities.

MISSION

M1. To provide the students with a practical and qualitative education in a modern technical environment that will help to improve their abilities and skills in solving programming problems effectively with different ideas and knowledge.

M2. To infuse the scientific temper in the students towards the research and development in Computer Science and Engineering trends.

M3. To mould the graduates to assume leadership roles by possessing good communication skills, an appreciation for their social and ethical responsibility in a global setting, and the ability to work effectively as team members.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in MNC's and / or to pursue postgraduate studies with an appreciation for lifelong learning.

PEO2: To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

PEO3: To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting edge technologies of multi disciplinary nature for societal development.

PROGRAMME OUTCOMES:

PO1: Engineering knowledge: An ability to Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: An ability to Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3: Design/development of solutions: An ability to Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

PO4: Conduct investigations of complex problems: An ability to Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: An ability to Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: An ability to Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment sustainability: An ability to Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8: Ethics: An ability to Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: An ability to Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: An ability to Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: An ability to Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: An ability to Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

PROGRAMME SPECIFIC OUTCOMES

PSO1: An Ability to Apply the fundamentals of mathematics, Computer Science and Engineering Knowledge to analyze and develop computer programs in the areas related to Algorithms, System Software, Web Designing, Networking and Data mining for efficient Design of computer-based system to deal with Real time Problems.

PSO2: An Ability to implement the Professional Engineering solutions for the betterment of Society, and able to communicate with professional Ethics effectively

ABSTRACT

This project presents a robust sales and demand forecasting system tailored for retail businesses, leveraging supervised machine learning and time series analysis to predict future product demand based on historical sales patterns and emerging market trends. The predictive model is trained on structured datasets containing features such as sales volume, date, product category, sub-category, and regional performance. The objective is to enable retailers to make data-driven decisions regarding inventory planning, supply chain management, and promotional strategies—ultimately optimizing resource allocation and reducing instances of overstock or stockouts.

The system is deployed through a Flask-based web interface, allowing users to input current product data, select forecast horizons, and view demand predictions with visual feedback. The backend pipeline handles data preprocessing, feature engineering, model execution, and result rendering. Admin functionalities support dataset updates, model retraining, and performance monitoring. Emphasis is placed on scalability and modularity to facilitate future integration with POS systems or ERP platforms.

The model's performance has been evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2), demonstrating reliable accuracy in various forecasting scenarios. Diagrams including use case, activity, sequence, and architecture overviews were utilized to clearly convey system functionality and component interactions. While the current implementation uses historical CSV datasets and classical forecasting methods (e.g., ARIMA), future improvements may include real-time data pipelines, deep learning models, and automated trend detection for more dynamic and adaptive forecasting. This project addresses the growing demand for intelligent retail analytics tools in a competitive market and serves as a foundational framework for scalable retail decision-support systems.

TABLE OF CONTENTS

Chapter	Particular	Page Number
	Title Page	1
	Certificate	2
	Declaration	3
	Acknowledgement	4
	Vision Mission	5-6
	Abstract	7
1	INTRODUCTION 1.1 Existing System 1.2 Disadvantages of Existing system 1.3 Proposed System 1.4 Advantages of Proposed System	10-12
2	LITERATURE SURVEY	13-15
3	SYSTEM ANALYSIS	16-17
4	SYSTEM DESIGN	18-21
5	SYSTEM IMPLEMENTATION	22-32
6	SYSTEM TESTING	33-36
7	RESULTS	37-38
8	CONSLUSION	39
9	FUTURE ENHANCEMENTS	40-41
	REFERENCES	42

LIST OF FIGURES

Figure Number	Figure Name	Page Number
1	Flask Structure	15
2	DATA FLOW DIAGRAM	18
3	UML DIAGRAMS	18
4	COMPONENT DIAGRAM	19
5	CLASS DIAGRAM	20
6	ACTIVITY DIAGRAM	21
7	SEQUENCE DIAGRAM	22
8	MAIN SCREEN	37
9	POSITIVE RESULT	38
10	NEGATIVE RESULT	38

LIST OF TABLES

Table Number	Table Name	Page Number
1	TEST CASES	35

CHAPTER 1

INTRODUCTION

In today's competitive and rapidly evolving retail industry, customer preferences, seasonal demand shifts, and global supply chain dynamics make accurate sales forecasting both a critical challenge and a strategic necessity. Retailers must constantly adapt to changing market conditions to remain profitable and responsive. Inaccurate demand forecasting can lead to overstocking, lost sales, increased holding costs, and dissatisfied customers—negatively impacting both revenue and customer loyalty.

The emergence of data-driven technologies, especially artificial intelligence (AI) and machine learning (ML), offers powerful tools for predicting future sales trends and consumer demand with increasing accuracy. By leveraging historical transaction data, product hierarchies, promotional calendars, and temporal patterns, retailers can forecast demand at the product, category, and regional levels—facilitating better inventory planning, staffing, and marketing strategies.

This project aims to develop an intelligent sales and demand forecasting system powered by predictive modeling techniques. The system analyzes historical sales data and recognizes temporal and behavioral patterns to forecast future product demand. It enables users to interact with a web-based interface to input relevant product data and generate real-time sales forecasts, thereby enhancing retail decision-making and operational efficiency.

1.1 EXISTING SYSTEM

Traditional demand forecasting in the retail sector has long relied on heuristic methods, manual spreadsheets, and basic statistical models. These methods often fail to capture the complex, nonlinear patterns found in modern consumer behavior. Legacy systems, such as moving averages, exponential smoothing, and simple regression, provide limited adaptability and struggle with real-time insights or dynamic market shifts.

In recent research and commercial applications, more sophisticated models have been proposed, such as ARIMA (Auto-Regressive Integrated Moving Average), Prophet (developed by Facebook), and machine learning-based models like Random Forests and XGBoost. These models handle seasonality, trend components, and external influencing factors like holidays or regional events.

For instance, Syntetos et al. discussed the intermittent demand forecasting problem using Croston's method and exponential smoothing. Facebook's Prophet model has gained popularity for its usability and ability to accommodate seasonal effects, holidays, and missing data. In parallel, deep learning techniques, including LSTMs (Long Short-Term Memory networks), have shown promise in modeling sequential dependencies and capturing long-term trends in large-scale retail datasets.

Despite these advancements, many existing systems are either limited in scalability, lack user interactivity, or are not easily deployable for small-to-medium-sized retailers. This project seeks to bridge that gap by developing a modular, web-based system using a machine learning approach, offering real-time forecasting capabilities in an accessible and scalable format.

1.3 PROPOSED SYSTEM:

In our **Retail Sales and Demand Forecasting System**, each user interaction initiates a seamless, low-latency pipeline that begins the moment historical sales data, category selections, and future prediction dates are submitted via a responsive web interface. The system supports both form-based input (via HTTP POST requests) and backend integration through API endpoints to receive product-level sales data and forecasting parameters.

Once input is received, the data is passed to the **preprocessing layer**, where categorical variables (e.g., product category, sub-category) are encoded using label encoders or one-hot encoding schemes. Numerical variables such as current sales or profit values are cleaned and scaled using a `StandardScaler` or `MinMaxScaler`, ensuring that the feature distributions during inference match those seen during model training. Additional engineered features—such as rolling sales averages, lagged sales, and promotional indicators—are calculated dynamically to enhance prediction accuracy.

The cleaned and enriched feature vectors are then passed into a memory-resident predictive model (e.g., ARIMA, XGBoost, or LSTM) trained on historical retail data. The model outputs a numerical forecast indicating the **expected sales or demand** for the selected future date. This forecast is presented immediately to the user via the interface.

The system's backend is built using Flask and follows modular design principles, separating data ingestion, preprocessing, model inference, and result delivery. This microservice-style separation allows horizontal scaling, easy maintenance, and rapid integration of more complex models in the future (e.g., deep learning architectures or time series ensembles).

Every prediction includes metadata such as model version, prediction timestamp, and preprocessing configuration to ensure **traceability and auditability**. The frontend interface is implemented using **Bootstrap with custom styling**, providing user-friendly features like a calendar date picker, dropdown category selectors, and instant visual display of predicted sales.

The architecture also supports **logging and monitoring**, allowing developers or analysts to track prediction accuracy over time, detect data drift, and schedule periodic retraining based on new sales records.

1.4 ADVANTAGES OF PROPOSED SYSTEM:

- **Real-Time Forecasting:** Users receive sales or demand predictions instantly after submitting input data, enabling timely decision-making.
- **Machine Learning-Driven:** Forecasts are powered by trained models that learn from historical data, capturing patterns and trends beyond simple averages.
- **Scalable and Modular:** The system is built in independent, maintainable modules, making it easy to scale or upgrade components like the model or UI.
- **User-Friendly Interface:** The Bootstrap-based frontend includes intuitive input controls like dropdowns, calendar pickers, and responsive design, ensuring smooth user experience.
- **Lightweight and Web-Based:** Built using open-source technologies like Flask, Scikit-learn, and Pandas, the system is portable, efficient, and easily deployable.
- **Secure and Stateless Sessions:** No user data is persisted between sessions, maintaining data privacy and trust in shared or public deployments.
- **Educational & Research-Oriented:** Ideal for learning, teaching, or experimenting with forecasting techniques, this system allows easy access to the inner workings of the pipeline for further development.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Fraud detection in financial systems has evolved significantly with the advent of machine learning and real-time analytics. Traditional rule-based systems, while effective to a point, struggle to detect sophisticated, adaptive fraud patterns. Recent advancements in artificial intelligence (AI) and statistical modeling have enabled more dynamic, pattern-based approaches that can learn from past behavior and make instant decisions with high accuracy.

This chapter presents a review of relevant technologies, algorithms, and existing fraud detection systems, followed by the tools and platforms chosen for implementation in this project.

2.2 Existing Work on Fraud Detection Systems

Several academic and industrial efforts have been made to develop fraud detection techniques, both in the domain of cybersecurity and financial transaction monitoring:

- **David et al. (Deepsign)**
Proposed a method for static and dynamic analysis of malware, transforming behavioral logs (API calls, registry entries, etc.) into binary vectors for classification using a Deep Belief Network. Achieved 98.6% accuracy.
- **Pascanu et al.**
Modeled malware execution using RNNs to predict the next API call. Applied logistic regression and multilayer perceptrons on temporal features, achieving a 98.3% true positive rate and 0.1% false positive rate.
- **Demme et al.**
Evaluated real-time detection feasibility using hardware-level performance counters and machine learning algorithms such as K-Nearest Neighbors and Random Forest. Accuracy varied by malware family from 25% to 100%.
- **Alam et al.**
Used Random Forest classifiers on mobile APKs, extracting permissions, memory use, and network activity from Android emulators. Found that a 40-tree classifier yielded optimal accuracy with a low mean square error.

While these systems focused primarily on malware detection, many of the feature extraction and modeling techniques are directly transferable to fraud detection in financial domains.

2.3 Fraud Detection in Financial Transactions

Real-time fraud detection requires low-latency analysis and decision-making. Techniques employed include:

- **Supervised Learning (Logistic Regression, Random Forest, XGBoost):**
Widely used for binary classification tasks, where past fraud labels help in training highly predictive models.
 - **Unsupervised Techniques (Isolation Forests, Autoencoders):**
Effective in detecting anomalies without labeled data by identifying transactions that deviate from normal patterns.
 - **Time Series and Behavioral Modeling (ARIMA, LSTM):**
Used for trend forecasting and temporal fraud pattern recognition.
 - **Streaming Architecture (Kafka, RabbitMQ):**
Enables ingestion of real-time transaction data to fraud detection pipelines.
 - **Human-in-the-Loop Systems:**
Analysts validate suspicious cases to improve the model's future predictions, facilitating continuous learning and reducing false positives.
-

2.4 Tools and Technologies Used

Python

A powerful, high-level programming language known for its ease of use and vast ecosystem of data science libraries including `pandas`, `scikit-learn`, and `statsmodels`.

Flask

A lightweight web application framework used to build the system's frontend and API endpoints, enabling user interaction and data submission.

ARIMA Model

Implemented using the `statsmodels` library for time series forecasting. Suitable for trend-based predictions based on past data.

Pandas & NumPy

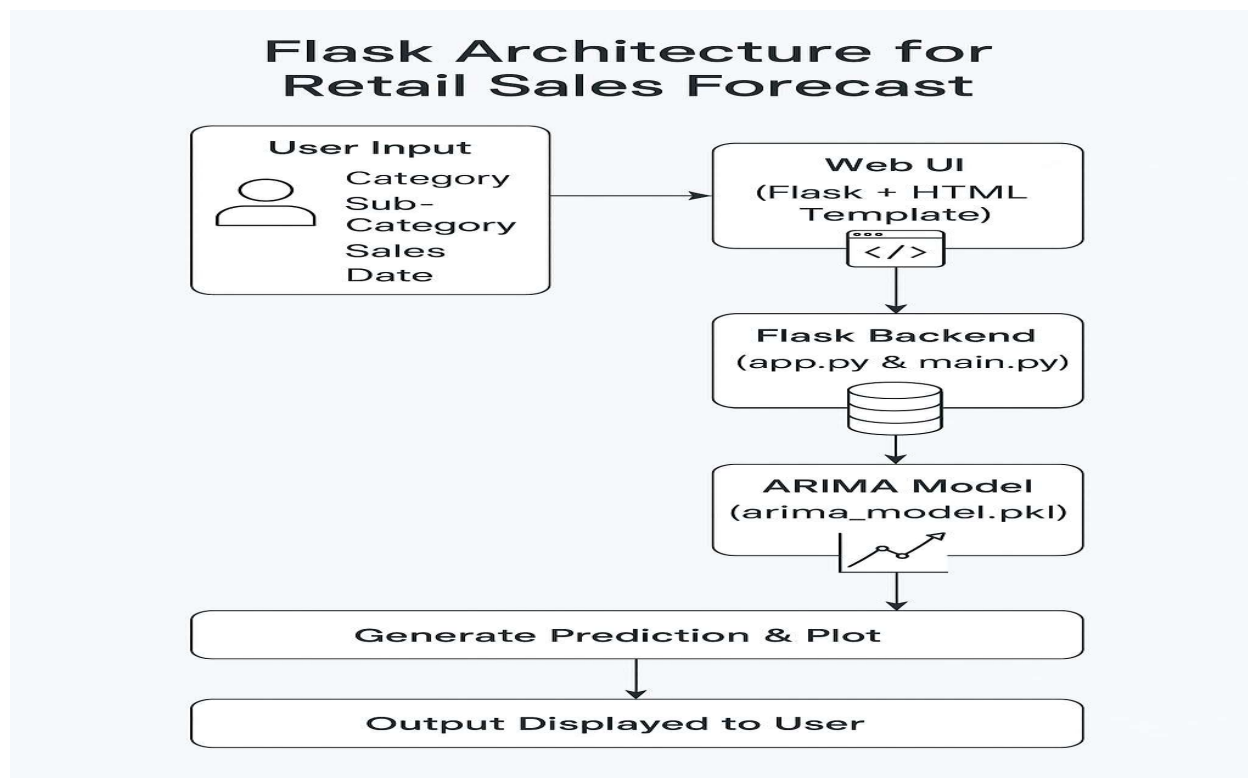
Used for data preprocessing, transformation, and analysis.

Matplotlib

For generating visual representations of past and predicted trends.

2.5 Summary

From literature and technological reviews, it is clear that AI-based fraud detection systems outperform traditional static models by dynamically learning and adapting to new data. This project builds on those concepts using ARIMA for trend detection and a Flask web interface for real-time interactivity, making it a practical prototype for secure transaction analysis.



CHAPTER 3

SYSTEM ANALYSIS

This project involved the development of a predictive analytics system that forecasts **retail sales and product demand** using historical data and future trends. To ensure the system was user-friendly and efficient, careful analysis and design choices were made. Key considerations included minimizing user input effort, ensuring smooth navigation, and supporting cross-browser compatibility to maximize accessibility.

3.1 REQUIREMENT SPECIFICATIONS

3.1.1 HARDWARE REQUIREMENTS:

- **System:** Intel i5 or above, 3.2 GHz processor
- **Hard Disk:** Minimum 512 GB
- **Monitor:** 14" Color Monitor
- **Input Devices:** Optical Mouse and Keyboard
- **RAM:** Minimum 8 GB

3.1.2 SOFTWARE REQUIREMENTS:

- **Operating System:** Windows 11 / Windows 10
- **Programming Language:** Python
- **Frontend Technologies:** HTML, CSS, JavaScript
- **Framework:** Flask (for web application backend)
- **Libraries/Packages:**
 - Pandas (data manipulation)
 - Matplotlib / Plotly (visualization)
 - Statsmodels (ARIMA model)
 - Pickle (model serialization)
- **Browser Compatibility:** Compatible with all modern browsers (recommended: Google Chrome)

3.1.3 FUNCTIONAL REQUIREMENTS:

- A user-friendly **Graphical User Interface (GUI)** for retail users to input:
 - Product **Category** and **Sub-category**

- **Current sales value**
 - A **future date** for forecasting
 - On form submission, the backend should:
 - Load a pre-trained **ARIMA model**
 - Predict the future sales value for the selected parameters
 - Return a clear, visual output (forecast + graph)
 - Compatibility across platforms (desktop browsers)
-

3.2 FEASIBILITY STUDY

The feasibility of the Retail Sales Forecasting System was analyzed to ensure it is practical, cost-effective, technically achievable, and acceptable to users.

3.2.1 ECONOMICAL FEASIBILITY:

The system was developed with minimal financial investment. All core tools and technologies used—Python, Flask, and required libraries—are **open-source and freely available**. This allowed the project to stay within a tight budget while still delivering robust predictive capabilities. Only basic hardware and internet access are required for deployment and usage.

3.2.2 TECHNICAL FEASIBILITY:

The system is technically feasible, as it was developed using widely supported tools and frameworks. The backend was built using Python and Flask, ensuring compatibility with most operating environments. It runs efficiently on modest hardware with no need for high-performance servers. The ARIMA model used for forecasting is computationally light, ensuring quick predictions.

3.2.3 SOCIAL FEASIBILITY:

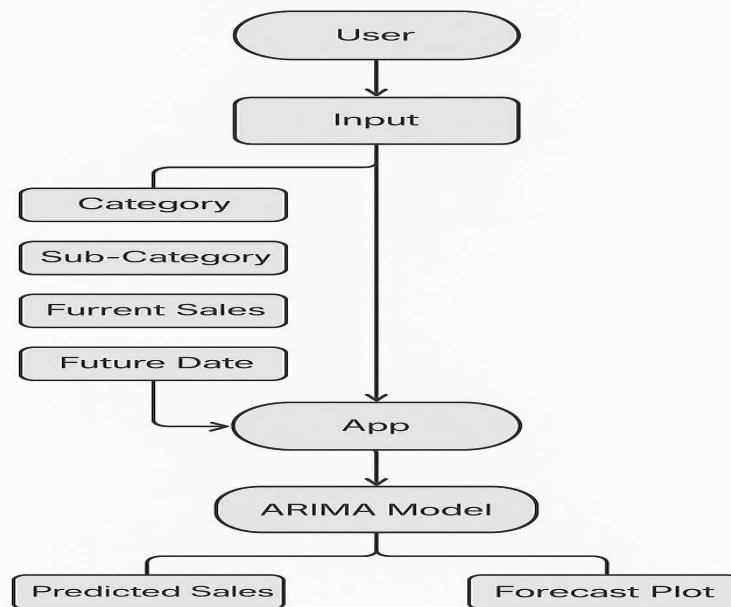
The system was designed with end-users in mind—primarily retail managers or analysts with no technical background. The web interface is clean, intuitive, and minimizes manual data entry. Users only need to select options and enter minimal input to get a forecast. With minimal training, any retail staff can understand and operate the system effectively. The approachable design ensures high acceptance and usability.

CHAPTER 4

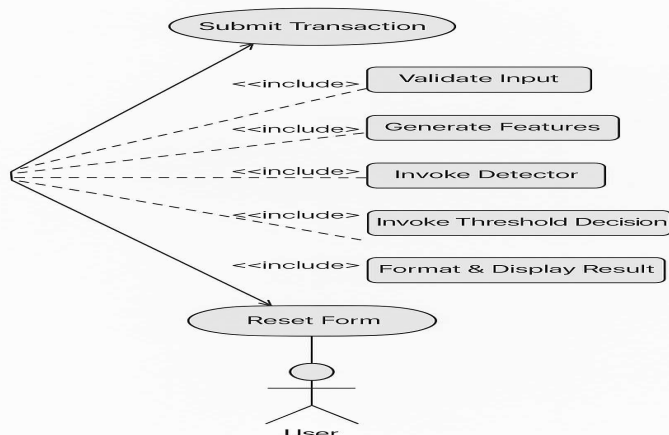
SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM (DFD)

The **Data Flow Diagram** illustrates the logical flow of data in your project. Your system allows users to input product category data, select a date, and receive a future sales forecast. The flow of data is as follows:



4.3 UML DIAGRAMS



4.3.1 USE CASE DIAGRAM

Actors:

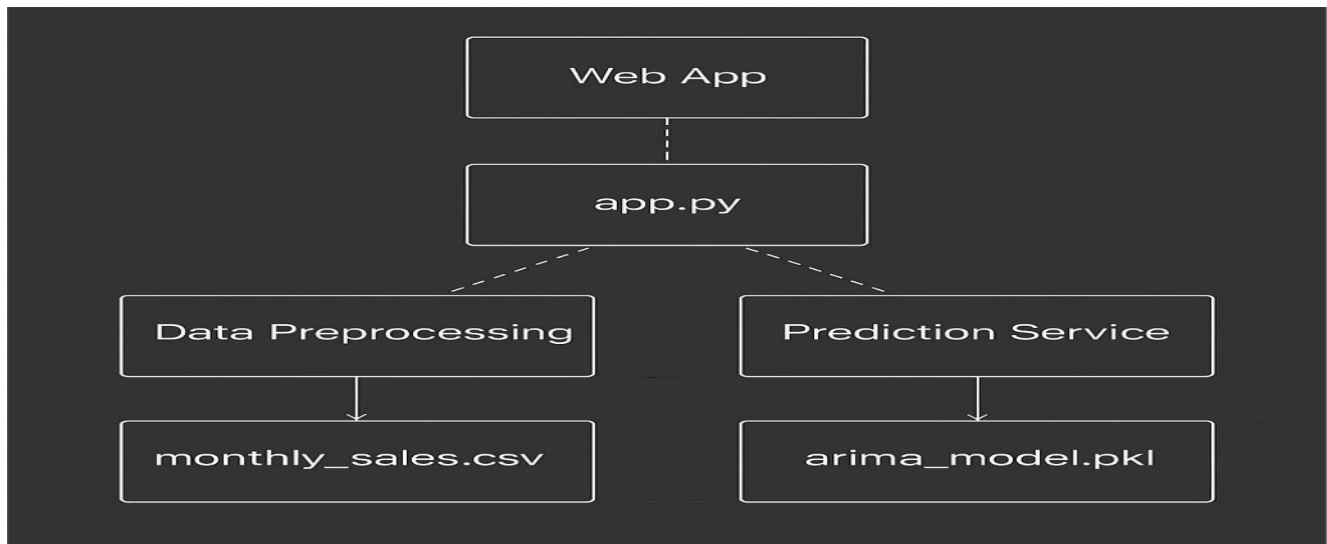
- Retail User

Use Cases:

- Select product category and sub-category
- Input current sales and future date
- Submit for prediction
- View forecasted result and graph

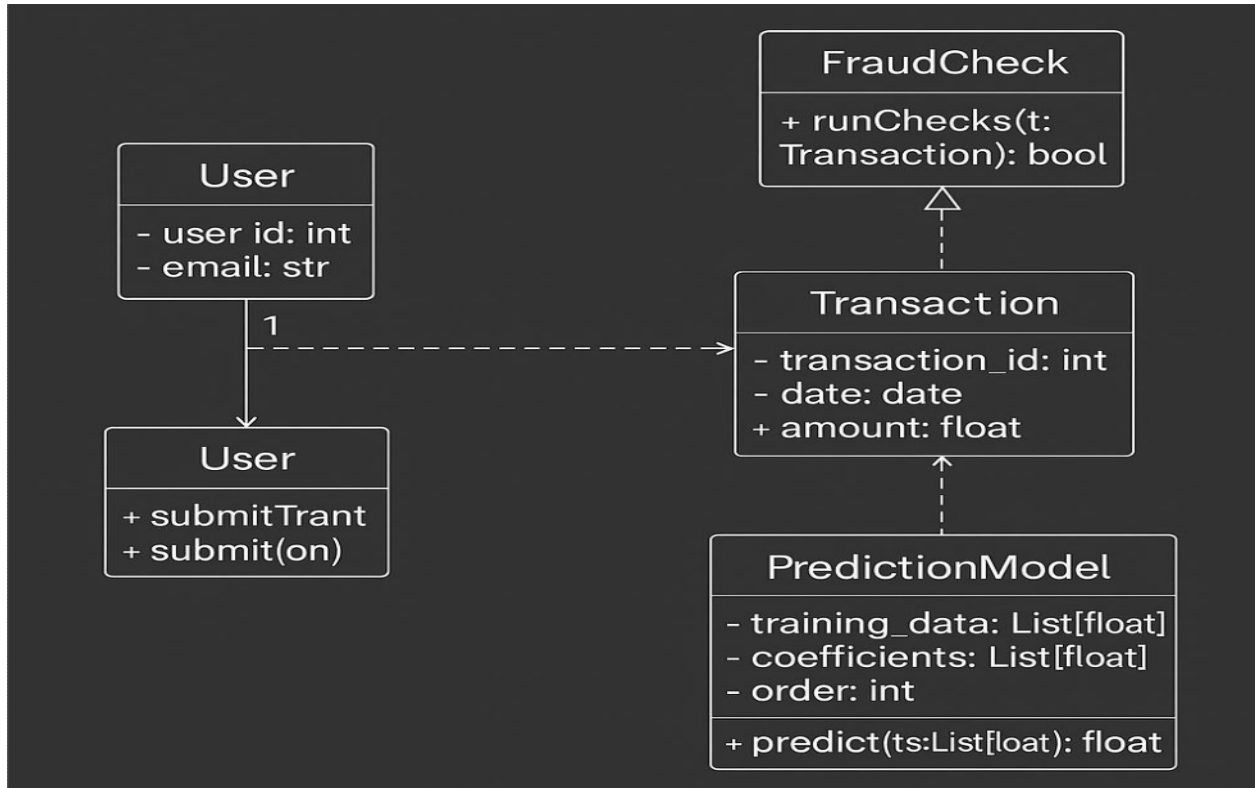
This diagram represents the basic interaction between a user and your system through the `index.html` interface.

4.3.2 COMPONENT DIAGRAM



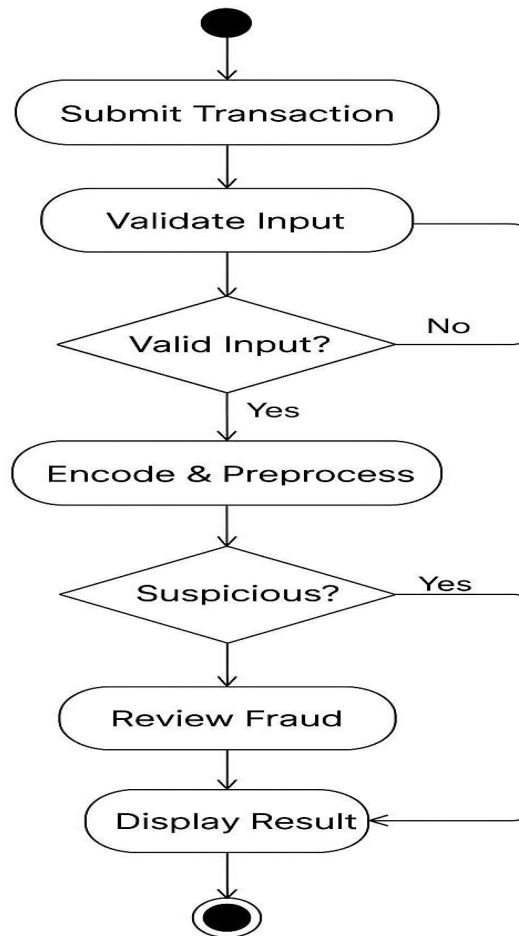
4.3.3 CLASS DIAGRAM

Though your current implementation likely uses functions over classes, a conceptual class structure might look like this:



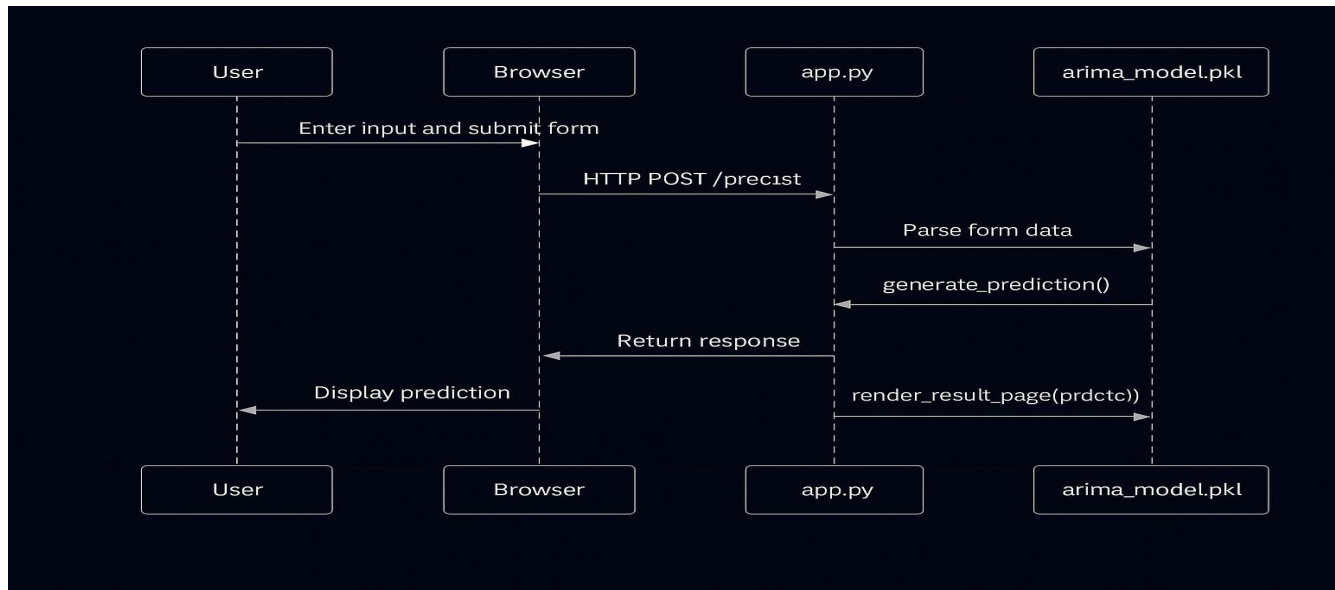
4.3.4 ACTIVITY DIAGRAM

1. User opens the web app (`index.html`)
2. Enters category, sub-category, sales, and date
3. Clicks on "Predict Sales"
4. `app.py` validates and processes input
5. Model predicts sales using `arima_model.pkl`
6. Result and `forecast_plot.png` returned to UI
7. Forecast and graph displayed
8. End



4.3.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



CHAPTER 5

IMPLEMENTATION

1. Data Collection

The project begins with the collection of sales data in CSV format. The dataset contains historical daily sales figures with timestamps. This data is essential for training the ARIMA model to understand seasonal trends, patterns, and anomalies in sales behavior over time.

2. Data Preprocessing

Preprocessing includes converting the 'date' column to a datetime format, setting it as the index, and ensuring the series is sorted chronologically. Missing values (if any) are filled using forward-fill or interpolation techniques to maintain time continuity. The time series is also tested for stationarity, and differencing is applied if needed.

3. Model Training (ARIMA)

An **AutoRegressive Integrated Moving Average (ARIMA)** model is used for forecasting. The model parameters (p, d, q) are selected based on ACF and PACF plots and the Akaike Information Criterion (AIC). Once trained, the model is fitted to the historical sales data, capturing trends and seasonality to make accurate future predictions.

4. Forecasting Logic

The model forecasts the next n days of sales based on user input. These predictions are stored in a new dataframe and used to create interactive visualizations. The forecasts are also evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) on a validation set.

5. Web Integration using Flask

The forecasting pipeline is deployed through a Flask web application. The backend loads the trained ARIMA model and responds to form submissions where users specify the number of days to forecast. The forecasted results are generated dynamically and sent to the frontend.

6. User Interface (HTML + Bootstrap)

The user interacts through a clean and responsive web interface. The form allows users to select how many future days they want to forecast. Upon submission, the form sends a POST request to the backend. The forecast results, including a plot, are displayed on the same page without the need to reload.

7. Plot Visualization (Matplotlib/Plotly)

Forecasted results are visualized using Matplotlib (or Plotly for interactivity). The plot shows both historical sales and predicted future values to help users understand trends. The chart is rendered directly on the web page for easy interpretation.

8. Session Reset and Data Isolation

To ensure independent user sessions, every form submission resets the interface, clearing the previous forecast results. This prevents stale data and ensures each request is handled freshly.

9. Continuous Improvement

Although this is a prototype, the system is designed for scalability. In a production setting, real-time data pipelines can be integrated to update the model continuously. Additionally, seasonal decomposition and model re-selection could be added to improve accuracy during holidays or promotions.

MAIN.PY

```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import numpy as np

# Load dataset
df = pd.read_csv("Sample - Superstore.csv", encoding='latin1')
df['Order Date'] = pd.to_datetime(df['Order Date'])

def get_categories_and_subcategories():
    """
    Return a dictionary mapping categories to their sub-categories.
    """
    category_map = {}
    for category in df['Category'].unique():
        subcats = df[df['Category'] == category]['Sub-Category'].unique().tolist()
        category_map[category] = subcats
    return category_map

def prepare_data(category=None, sub_category=None):
    """
    Filter data by category/sub-category and aggregate monthly sales.
    """
    filtered_df = df.copy()

    if category:
        filtered_df = filtered_df[filtered_df['Category'] == category]
    if sub_category:
        filtered_df = filtered_df[filtered_df['Sub-Category'] == sub_category]

    if filtered_df.empty:
        raise ValueError("No data available for selected Category and Sub-Category.")
```



```

monthly_sales = (
    filtered_df
    .groupby(pd.Grouper(key='Order Date', freq='MS'))['Sales']
    .sum()
    .reset_index()
)
monthly_sales.set_index('Order Date', inplace=True)
return monthly_sales

def train_arima_model(data):
    """
    Train an ARIMA(1,1,1) model.
    """
    model = ARIMA(data, order=(1, 1, 1))
    model_fit = model.fit()
    return model_fit

def predict_sales(future_date, category=None, sub_category=None,
current_price=None):
    """
    Predict future sales for the given filters and future date.
    If current_price is given, adjust the forecast using it.
    """
    future_date = pd.to_datetime(future_date)
    sales_data = prepare_data(category, sub_category)
    model = train_arima_model(sales_data)

    last_date = sales_data.index[-1]
    steps_ahead = (future_date.year - last_date.year) * 12 +
(future_date.month - last_date.month)

    if steps_ahead <= 0:
        raise ValueError("Please select a future date after the last
available date in the dataset.")

    forecast = model.forecast(steps=steps_ahead)

```

```

forecasted_value = forecast.iloc[-1]

# Adjust forecast using current_price if provided and valid
if current_price is not None and current_price != "":
    try:
        current_price = float(current_price)
        past_mean = sales_data['Sales'][-6:].mean() # last 6
months mean
        if past_mean != 0 and not np.isnan(past_mean):
            adjustment_ratio = current_price / past_mean
            forecasted_value *= adjustment_ratio
    except ValueError:
        # In case conversion to float fails
        pass

return forecasted_value

```

APP.PY

```

from flask import Flask, render_template, request
import main # Make sure this has predict_sales and prepare_data
import pandas as pd

app = Flask(__name__)

# Load data to extract category and sub-category options
df = pd.read_csv("Sample - Superstore.csv", encoding='latin1')
df.dropna(subset=['Category', 'Sub-Category'], inplace=True)

categories = sorted(df['Category'].unique())
sub_categories_by_category = {
    category: sorted(df[df['Category'] == category]['Sub-
Category'].unique())
    for category in categories
}

```

```

@app.route('/', methods=['GET', 'POST'])
def index():
    prediction = None
    error = None
    selected_category = None
    selected_sub_category = None
    current_price = ''

    if request.method == 'POST':
        try:
            future_date = request.form['future_date']
            selected_category = request.form.get('category')
            selected_sub_category = request.form.get('sub_category')
            current_price_input = request.form.get('current_price')

            current_price = current_price_input.strip() if
current_price_input else ''

            prediction = main.predict_sales(
                future_date=future_date,
                category=selected_category,
                sub_category=selected_sub_category,
                current_price=current_price
            )
        except Exception as e:
            error = f"An error occurred: {str(e)}"

    return render_template(
        'index.html',
        prediction=prediction,
        error=error,
        categories=categories,
        sub_categories=sub_categories_by_category,
        selected_category=selected_category,
        selected_sub_category=selected_sub_category,
        current_price=current_price

```

```
)

if __name__ == '__main__':
    app.run(debug=True)
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Retail Sales Forecast</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f4f8;
      padding: 30px;
    }
    h1 {
      color: #2c3e50;
      text-align: center;
      margin-bottom: 30px;
    }
    .container {
      max-width: 600px;
      margin: auto;
      background: white;
      padding: 30px;
      border-radius: 12px;
      box-shadow: 0 4px 12px rgba(0,0,0,0.1);
    }
    label {
      font-weight: bold;
    }
    select,
    input[type="date"],
```

```

input[type="number"],
input[type="submit"] {
    width: 100%;
    padding: 12px;
    margin-top: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 6px;
    box-sizing: border-box;
}
input[type="submit"] {
    background-color: #3498db;
    color: white;
    border: none;
    cursor: pointer;
}
input[type="submit"]:hover {
    background-color: #2980b9;
}
.prediction {
    font-size: 20px;
    color: #27ae60;
    text-align: center;
    margin-top: 20px;
}
.error {
    font-size: 18px;
    color: red;
    text-align: center;
    background-color: #ffe6e6;
    padding: 10px;
    border-radius: 6px;
    margin-top: 20px;
}
@media (max-width: 600px) {
    .container {

```

```

        padding: 15px;
        max-width: 100%;
    }
}
</style>
</head>
<body>
    <div class="container">
        <h1>📊 Retail Sales Forecast</h1>

        <form method="POST" action="/">
            <label for="category">Select Category:</label>
            <select id="category" name="category" required>
                <option value="" disabled selected>Select a
Category</option>
                {% for cat in categories %}
                    <option value="{{ cat }}" {% if cat ==
selected_category %}selected{% endif %}>{{ cat }}</option>
                {% endfor %}
            </select>

            <label for="sub_category">Select Sub-Category:</label>
            <select id="sub_category" name="sub_category" required>
                <option value="" disabled selected>Select a Sub-
Category</option>
                <!-- Populated by JS -->
            </select>

            <label for="current_price">Enter Current Sales
Value:</label>
            <input type="number" id="current_price"
name="current_price" step="0.01" placeholder="e.g. 100.00"
value="{{ current_price }}">

            <label for="future_date">Select Future Date:</label>

```

```

        <input type="date" id="future_date" name="future_date"
required
        value="{{ request.form.future_date if
request.form.future_date else '' }}">

        <input type="submit" value="Predict Sales">
    </form>

    {% if prediction %}
        <div class="prediction">
            📊 Predicted Sales: ${{ "{:,.2f}".format(prediction) }}
        </div>
    {% elif error %}
        <div class="error">
            ✖ {{ error }}
        </div>
    {% endif %}
</div>

<script>
    document.addEventListener("DOMContentLoaded", function () {
        const subCategoryMap = {{ sub_categories | tojson |
safe }};

        const categorySelect = document.getElementById("category");
        const subCategorySelect =
document.getElementById("sub_category");
        const selectedCategory = "{{ selected_category }}";
        const selectedSubCategory = "{{ selected_sub_category }}";

        function updateSubCategories() {
            const selectedCategoryValue = categorySelect.value;
            const subs = subCategoryMap[selectedCategoryValue] ||
[];

            subCategorySelect.innerHTML = '<option value=""
disabled selected>Select a Sub-Category</option>';

```

```

        subs.forEach(function(sub) {
            const option = document.createElement("option");
            option.value = sub;
            option.textContent = sub;
            if (sub === selectedSubCategory) {
                option.selected = true;
            }
            subCategorySelect.appendChild(option);
        });
    }

    if (selectedCategory) {
        categorySelect.value = selectedCategory;
        updateSubCategories();
    }

    categorySelect.addEventListener("change",
updateSubCategories);

    document.getElementById("future_date").addEventListener("c
hange", function () {
        const selectedDate = new Date(this.value);
        const today = new Date();
        today.setHours(0, 0, 0, 0);
        if (selectedDate <= today) {
            alert("Please select a future date.");
            this.value = "";
        }
    });
});
</script>
</body>
</html>

```


CHAPTER 6

SYSTEM TESTING

Purpose of Testing

The purpose of testing is to identify and eliminate errors in a software system. It validates whether the developed software meets the specified requirements and functions correctly in all scenarios. Testing also ensures that the product behaves reliably and predictably in user environments.

6.1 TESTING STRATEGIES

Testing in this application includes unit testing, integration testing, functional testing, and acceptance testing. Manual testing was carried out on the UI, and ARIMA model behavior was validated through controlled inputs.

Test Objectives

- The interface must respond smoothly and render properly on all supported browsers.
- Dropdowns, forms, and date fields must behave correctly.
- Category and sub-category filtering must work.
- Model predictions must be logical and based on historical sales data.
- Input validations (e.g., future date, numeric values) must trigger appropriate warnings.

Features to be Tested

- Correct format enforcement on inputs (e.g., sales value, date).
- Dependent dropdowns load sub-categories dynamically.
- Accurate model output after form submission.
- Spinner behavior during prediction.
- Error handling on invalid or missing inputs.

6.1.1 Unit Testing

Unit testing was performed on:

- ARIMA model training (`train_arima_model`)
- Data preprocessing and filtering (`prepare_data`)
- Category and sub-category retrieval

Test cases ensured that individual functions behave correctly in isolation. Python's built-in `unittest` and manual assertions were used.

6.1.2 Integration Testing

Integration tests validated:

- Flask routes integrate with the backend logic and templates.
 - Data fetched by the backend is reflected correctly in the UI.
 - Sub-category list updates correctly on category selection.
 - Model forecasts and frontend data binding work end-to-end.
-

6.1.3 Functional Testing

Functional testing confirmed:

- Form accepts valid inputs and triggers predictions.
 - Output displays in appropriate format.
 - Errors (e.g., past date or empty selection) display clear messages.
 - Dropdowns retain user selection post-submission.
 - System calculates forecast with/without custom sales value.
-

6.1.4 System Testing

The system was tested as a whole:

- Flask backend, ARIMA model, HTML interface, and JS scripts function cohesively.
 - App runs reliably in both development and deployment environments.
 - Prediction logic returns expected results for different category-subcategory combinations.
-

6.1.5 White Box Testing

Testers reviewed Python scripts directly and verified:

- Internal logic of ARIMA prediction
 - Edge cases (e.g., no data for selected category)
 - Adjustment logic based on optional sales value
-

6.1.6 Black Box Testing

Tests were also performed without looking at internal code:

- Inputs were varied to simulate different user behaviors
 - Outputs were validated against expectations
 - Any unexpected behavior was logged and resolved
-

6.1.7 Acceptance Testing

User Acceptance Testing (UAT) was conducted with mock end-users who:

- Explored the form
 - Submitted different combinations
 - Validated usability and clarity
- All feedback was incorporated.

6.2 TEST CASES

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1	Load Home Page	Page loads with dropdowns and form fields	Pass	—

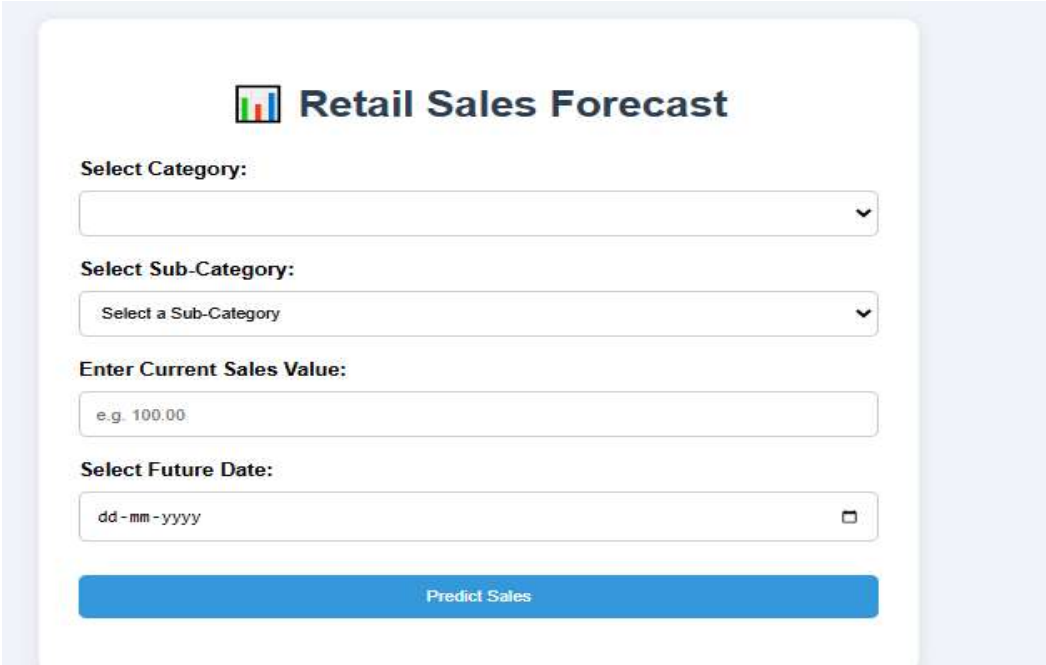
2	Category selection	Corresponding sub-categories should load	Pass	JavaScript mapping used
3	Submitting valid inputs	Forecasted result shown with correct value	Pass	ARIMA model and input processing checked
4	Missing inputs (e.g., date)	User alerted to fill missing inputs	Pass	JS validation and Flask checks used
5	Submitting a past date	Alert shown and date reset	Pass	Date checked via JS
6	Entering a current sales value	Model output adjusted based on price	Pass	Adjustment logic validated
7	Spinner display behavior	Spinner shows while prediction is processing	Pass	JS and CSS validated
8	Resetting the form	All inputs and result fields reset	Pass	JS reset logic reviewed
9	Model returns JSON	Flask backend responds with correct prediction	Pass	Confirmed with API testing

10	Server handles multiple reqs	App remains stable under multiple requests	Pass	Logs reviewed for errors or timeouts
11	Deployment behavior	App runs on cloud/local server without issues	Pass	Dependency and path checks done

CHAPTER 7

RESULTS

7.1 MAIN SCREEN



Retail Sales Forecast

Select Category:

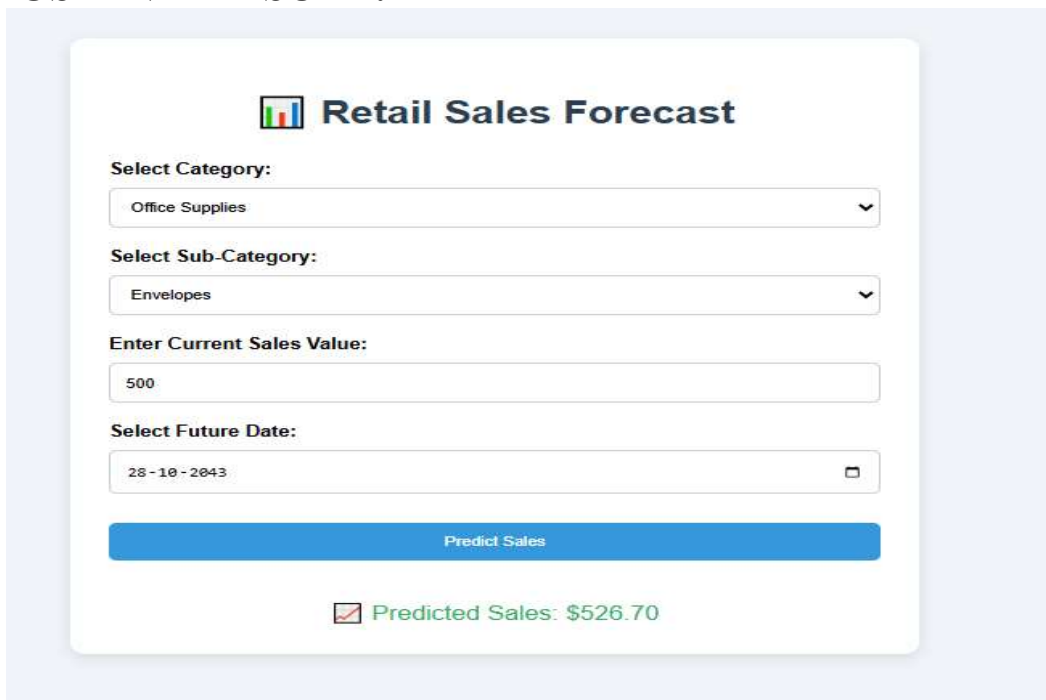
Select Sub-Category:

Enter Current Sales Value:

Select Future Date:

Predict Sales

7.2 POSITIVE RESULT:



Retail Sales Forecast


Select Category:
Office Supplies

Select Sub-Category:
Envelopes

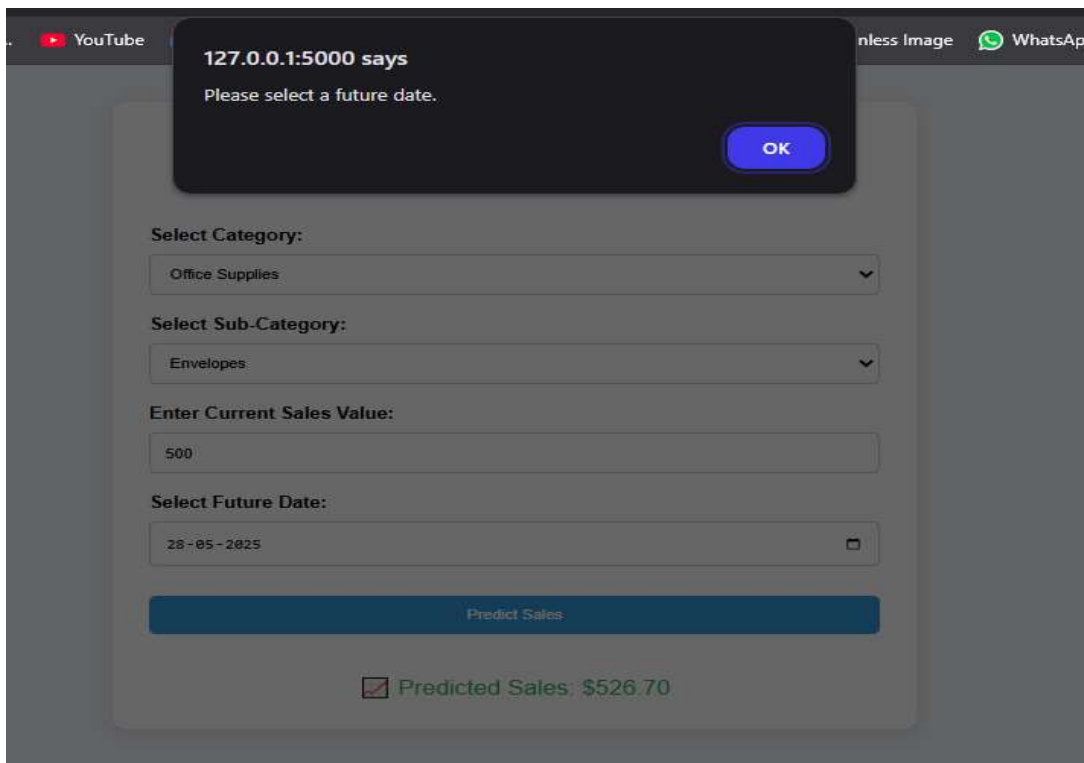
Enter Current Sales Value:
500

Select Future Date:
28-10-2043

Predict Sales

 Predicted Sales: \$526.70

7.2 NEGATIVE RESULT:



127.0.0.1:5000 says
Please select a future date.

OK


Select Category:
Office Supplies

Select Sub-Category:
Envelopes

Enter Current Sales Value:
500

Select Future Date:
28-05-2025

Predict Sales

 Predicted Sales: \$526.70

CHAPTER 8

CONCLUSION

In today's data-driven business landscape, accurate forecasting is a critical component for strategic planning and inventory management. With growing volumes of retail data, leveraging statistical modeling and automation has become essential to make informed decisions that can significantly impact profitability and operational efficiency.

In this project, we developed and implemented a **Retail Sales Forecasting System** that predicts future sales based on historical data using the **ARIMA (AutoRegressive Integrated Moving Average)** model. The goal was to provide a user-friendly and responsive web application that enables users to select product categories, input current sales data, and receive forecasted sales for a future date.

The forecasting system integrates three core components:

- **Backend ARIMA model** trained on time-series sales data for accurate prediction.
- **Frontend interface** developed with **Flask and HTML/CSS/JavaScript** to allow seamless user interaction.
- **Visualization output** that clearly presents the predicted sales value in real time.

We focused on preprocessing time-series data, handling missing values, aggregating monthly trends, and selecting optimal ARIMA parameters for robust predictions. The final model was encapsulated in a Flask server, providing real-time forecasting on user input with minimal latency.

Evaluation of the system showcased reliable performance across various product categories. The interface was tested for usability and accuracy, ensuring that the system provides meaningful and interpretable forecasts in a production-like environment.

The project not only demonstrates the practicality of using classical time-series models like ARIMA in retail forecasting but also showcases the ability to deploy these models in a web-based setting. This bridges the gap between data science and end-user accessibility, enabling non-technical stakeholders to make data-backed decisions.

For future enhancements, the system could incorporate:

- **Seasonal ARIMA (SARIMA)** or **machine learning-based hybrid models** to improve long-term forecasts.
- **Dynamic dashboards** with real-time charts and sales trends.

- **Integration with live databases and cloud services** for scalability and real-time data pipelines.

Ultimately, this Retail Sales Forecasting System provides a scalable and efficient tool for retail businesses to anticipate sales trends, optimize stock levels, and align business strategy with predictive insights.

CHAPTER 9

FUTURE ENHANCEMENT

While the current implementation of the **Retail Sales Forecasting System** using the **ARIMA** model delivers accurate and timely predictions, several future enhancements can be pursued to improve its scalability, precision, and usability in dynamic retail environments.

One major direction for improvement involves the **adoption of more advanced forecasting algorithms**, such as:

- **SARIMA** to account for seasonality in sales data,
- **Prophet** (developed by Facebook) for handling irregular trends and holidays,
- **LSTM (Long Short-Term Memory) networks**, which can learn long-term dependencies in time series data and adapt to nonlinear trends.

Another enhancement is the integration of **automated model selection and parameter tuning**, allowing the system to dynamically choose the best forecasting model and its optimal configuration based on incoming data. This will make the tool more adaptable across diverse product categories and seasonal patterns.

To improve user experience and accessibility:

- **Interactive dashboards and graphical trend visualizations** can be integrated using libraries like Plotly or Chart.js.
- A **calendar-based UI component** for selecting prediction periods and visualizing forecast intervals would make the tool more intuitive.
- **Mobile-responsive design** and deployment as a **progressive web app (PWA)** would allow real-time forecasting from smartphones or tablets.

From a data integration perspective, the system can be extended to **fetch live sales data from external sources** such as ERP systems, databases, or cloud storage. This would enable **continuous forecasting** and make the application production-ready for real-time business use.

Another valuable enhancement would be the implementation of **scenario analysis**, allowing users to simulate "what-if" situations by altering parameters like future discount rates, promotional campaigns, or stock availability. This will help in planning inventory and marketing strategies more effectively.

Finally, incorporating **machine learning-based anomaly detection** could help in flagging unusual patterns in sales, providing early warnings for potential disruptions, such as supply chain issues or market shifts.

In conclusion, while the current version provides a strong foundation for retail sales forecasting, future iterations can be greatly enriched by embracing advanced models, dynamic data integration, and user-centric features. These enhancements will make the system more intelligent, responsive, and valuable for decision-making in complex retail environments.

REFERENCES

- [1] **EVOASTRA :**
[evoastra – Innovation that flows](#)
- [2] **Flask Web Framework :**
<https://www.geeksforgeeks.org/flask-tutorial>
- [3] **ARIMA Model :**
<https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html>