**A Minor Project Report (CS755PC)**

**on**

**E-Commerce Sales Analysis**

*Submitted*

*in fulfillment of the requirements for the award of the degree of*

**Bachelor of Technology**

in

**Computer Science and Engineering (AI & ML)**

by

**Ms. Shivkoti Usha Sree**

**21261A6656**

Under the guidance of

**Ms. B. Varalaxmi**

**Assistant Professor**

**Department of Emerging Technologies**

**Mahatma Gandhi Institute of Technology**

(Affiliated to Jawaharlal Nehru Technological University Hyderabad) Kokapet(V), Gandipet(M), Hyderabad.

Telangana - 500 075.

**2024 - 2025**

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY(A)**

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

GANDIPET, HYDERABAD – 500075, Telangana

**CERTIFICATE**



This is to certify that the project entitled **E-Commerce Sales Analysis** is being submitted by **SHIVKOTI USHA SREE** bearing **Roll No. 21261A6656** in fulfillment of the requirements for the **Minor Project (CS755PC)** in **Emerging Technologies** is a record of Bonafide work carried out by her. The results of the investigations enclosed in this report have been verified and found satisfactory.

**PROJECT GUIDE HEAD OF THE DEPARTMENT**

Ms. B. Varalaxmi Dr. M. Rama Bai

(Assistant Professor) (Professor, Dept. of ET)

**External Examiner**

**DECLARATION**

This is to certify that the work reported in this project **E-Commerce Sales Analysis** is a record of work done by me in the Department of Emerging Technologies, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

**Shivkoti Usha Sree**

**(21261A6656)**

**ACKNOWLEDGEMENTS**

I would like to express my sincere thanks to **Prof. G. Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

I wish to express my sincere thanks and gratitude to **Prof. M. Rama Bai, Professor and HoD, Department of ET**, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to **Ms. Md Nazini**, **Assistant Professor** and **Mr. Mallela Srikanth, Assistant Professor**, Department of ET, MGIT, minor project coordinators for their encouragement and support throughout the project.

I am extremely thankful and indebted to my internal guide **Ms. B. Varalaxmi, Assistant Professor**, Department of ET, for her constant guidance, encouragement, and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of ET Department who helped me directly or indirectly, for completing this project.

**Shivkoti Usha Sree**

**(21261A6656)**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** |  |
| Certificate | i |
| [Declaration](#_bookmark0) | [ii](#_bookmark0) |
| [Acknowledgement](#_bookmark1) | [iii](#_bookmark1) |
| [List of Figures](#_bookmark2) | [vi](#_bookmark2) |
| [List of Tables](#_bookmark3) | [vii](#_bookmark3) |
| [Abstract](#_bookmark4) | [1](#_bookmark4) |
| 1. [Introduction](#_bookmark5) | [2](#_bookmark5) |
| 1.1 Motivation | 3 |
| 1.2 [Problem Definition](#_bookmark6) | 3 |
| 1.3 Existing System | 3 |
| 1.4 [Proposed System](#_bookmark7) | 4 |
| 1.5 [Requirements Specification](#_bookmark8) | 5 |
| 2. [Literature Survey](#_bookmark11) | 6 |
| 3. [E-Commerce](#_bookmark12) Sales Analysis | 11 |
| 3.1 Amazon Sales Analysis | 12 |
| 3.2 Customer Segmentation and Behavioral Analysis | 15 |
| 3.3 Predictive Time Series Analysis | 17 |
| 3.4 [UML Diagrams](#_bookmark14) | [2](#_bookmark14)0 |
| 3.4.1 Sales Analysis UML Diagrams | 20 |
| 3.4.2 Customer Segmentation UML Diagrams | 24 |
| 3.4.3 Stock Price Prediction UML Diagrams | 28 |
| 4. Testing and Results | 32 |
| 5. Integrating with User Interface | 27 |
| 6. Conclusion and Future Work | 38 |
| 7. Bibliography | 41 |
| 8. Appendix | 42 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
|  |  |  |
| Figure 3.4.1.1 | Data Flow Diagram (SA) | 20 |
| Figure 3.4.1.2 | Sequence Diagram (SA) | 21 |
| Figure 3.4.1.3 | Activity Diagram (SA) | 22 |
| Figure 3.4.1.4 | Collaboration Diagram (SA) | 23 |
| Figure 3.4.2.1 | Data Flow Diagram (CBA) | 24 |
| Figure 3.4.2.2 | Sequence Diagram (CBA) | 25 |
| Figure 3.4.2.3 | Activity Diagram (CBA) | 26 |
| Figure 3.4.2.4 | Collaboration Diagram (CBA) | 27 |
| Figure 3.4.3.1 | Data Flow Diagram (PA) | 28 |
| Figure 3.4.3.2 | Sequence Diagram (PA) | 29 |
| Figure 3.4.3.3 | Activity Diagram (PA) | 30 |
| Figure 3.4.3.4 | Collaboration Diagram (PA) | 31 |
| Figure 4.1 | Accuracy metric of ARIMA Model | 32 |
| Figure 4.1.1 | Size Analysis | 33 |
| Figure 4.1.2 | Categorical Analysis | 33 |
| Figure 4.1.3 | B2B Analysis | 33 |
| Figure 4.1.4 | Fulfilment Analysis | 33 |
| Figure 4.1.5 | State Wise Analysis | 33 |
| Figure 4.2.1 | Customer Segmentation Analysis | 34 |
| Figure 4.3.1 | Drastic fall of stock price due to Covid-19 | 35 |
| Figure 4.3.2 | Decomposition of the Seasonality | 35 |
| Figure 4.3.3 | Forecast of Actual v/s Predicted stock price | 35 |
| Figure 5.1 | Sales Analysis Dashboard | 36 |
| Figure 5.2 | Customer Segmentation and Behavioral Dashboard | 36 |
| Figure 5.3 | Amazon Stock Price Prediction Web | 36 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table 3.1 | Comparison of Literature survey | 8 |
| Table 4.2.1 | Recency, Frequency and Monetary Calculation | 34 |

**ABSTRACT**

In today’s competitive digital marketplace, e-commerce businesses, whether hosted on platforms like Shopify or Square Online, need a sophisticated understanding of their financial and operational performance to stay ahead. The development of a comprehensive sales analysis tool would be transformative in helping businesses gain a clear, data-driven perspective on their sales, profitability, and customer behaviour. This tool would integrate seamlessly with the e-commerce ecosystem, providing real-time insights and predictive analytics that can be leveraged for strategic decision-making, customer engagement, and revenue optimization.

Key Features and Functionalities:

1.Sales Dashboard:

The sales dashboard serves as the tool’s core, offering a snapshot of essential metrics that reveal the store’s financial health. This includes total sales, revenue growth, profit margins, average order value (AOV), and sales trends over specific timeframes (e.g., daily, weekly, monthly). Interactive charts and graphs provide a visual summary of these metrics, allowing store owners to easily track performance fluctuations and seasonal trends. Additionally, profit margin breakdowns can be presented by product or category, giving visibility into what products drive the highest profitability. This dashboard ensures that users can understand and act on key sales indicators without needing extensive financial expertise.

2.Customer Behaviour Analysis:

Understanding customer behaviour is critical for developing loyalty and maximizing lifetime value. This feature would track purchase frequency, average spend per customer, most popular products, and customer lifetime value (CLV), helping businesses identify and segment their most valuable customers. It could further analyse data to identify repeat buyers versus one-time purchasers, seasonal shopping patterns, and purchase motivators, helping businesses tailor their marketing and retention strategies. Such insights into customer behaviour can guide inventory management, targeted promotions, and personalized recommendations, ultimately leading to a more satisfying customer experience.

3.Predictive Analysis and Forecasting:

Leveraging historical data, the tool would offer predictive insights into future sales performance, enabling businesses to prepare for peak seasons, recognize potential sales dips, and optimize stock levels. Machine learning algorithms could forecast demand trends, identify potential growth opportunities, and flag areas of concern, such as declining product popularity or customer churn risk. The predictive analysis component would be instrumental for financial planning, enabling e-commerce businesses to allocate resources effectively, set realistic goals, and make data-backed decisions that align with long-term growth.

**1. INTRODUCTION**

In the fast-paced world of online retail, where competition is high and consumer behaviour rapidly evolves, data-driven insights are essential for an e-commerce business to thrive. Platforms such as Shopify, Square Online, and others empower businesses with online presence and transaction capabilities. However, for these platforms to fully enable business growth, they must go beyond basic sales tracking and incorporate advanced analytics that provide a deeper understanding of financial health, customer behaviour, and future trends. This is where a dedicated sales data analysis tool can bridge the gap, equipping e-commerce operators with the means to not only monitor their financial performance but also drive strategic improvements based on comprehensive data insights.

An effective sales analysis tool should deliver actionable intelligence into three core areas: financial performance, customer behaviour, and predictive trends. First, a Sales Dashboard can offer a snapshot of key metrics such as total sales, profit margins, average order value (AOV), and overall sales trends over time. Unlike raw data, this dashboard should provide intuitive, real-time visualizations that transform metrics into comprehensible insights. For example, tracking daily, weekly, or monthly sales trends and comparing these against historical data enables businesses to identify fluctuations and patterns that may otherwise go unnoticed. With this information, businesses can make timely adjustments, whether they are preparing for a seasonal spike in demand or addressing a dip in sales for a particular product category.

Beyond financial performance, understanding Customer Behaviour is crucial for retaining valuable clientele and enhancing customer lifetime value (CLV). This tool would analyse customer actions, from purchase frequency and average spend to product preferences and behaviour patterns. Identifying high-value customer segments—such as frequent purchasers or customers with high AOV—can enable businesses to craft tailored loyalty programs and targeted marketing efforts. Additionally, by tracking the popularity of certain products and analysing customer segments, the tool can help businesses better manage inventory, optimize product listings, and make data-informed decisions about promotions.

Finally, Predictive Analysis is a powerful component that leverages historical sales data to forecast future performance. This enables businesses to not only predict overall demand but also recognize emerging opportunities and potential risks. Using advanced algorithms, the tool can highlight trends, project sales for specific time periods, and provide alerts for potential areas of concern, such as declining product demand or changes in customer preferences. With these insights, business owners and managers can make proactive adjustments to stock levels, marketing budgets, and operational strategies, ensuring readiness for future market conditions.

In summary, a robust sales data analysis tool will empower e-commerce businesses to navigate their financial landscape with clarity, optimize customer engagement, and prepare strategically for growth. By consolidating key metrics into an accessible platform, providing an in-depth view of customer preferences, and using predictive insights to guide future actions, this tool would help businesses make data-driven decisions that boost profitability and foster sustainable growth. Ultimately, such a tool would be invaluable for any online retailer seeking to stay competitive, responsive, and customer-focused in today’s dynamic digital marketplace

**1.1 Motivation**

The motivation for developing a comprehensive sales analysis tool lies in the immense impact it could have on e-commerce businesses. In today’s digital marketplace, online retailers face intense competition and constantly shifting customer preferences. To succeed, they need more than just a simple sales tracker; they require a sophisticated tool that can uncover hidden patterns, predict future sales, and reveal insights into customer behavior. This tool would empower business owners to make informed, data-backed decisions, allowing them to allocate resources wisely, maximize profitability, and identify growth opportunities. For smaller businesses, it provides a way to level the playing field against larger competitors by making advanced analytics accessible and actionable. For established retailers, it streamlines decision-making, improves operational efficiency, and helps focus on high-value customers. In essence, a well-designed sales analysis tool could be a game-changer for e-commerce, enabling businesses to thrive by using data intelligently. By automating insights and simplifying complex metrics, this tool would transform how businesses understand their financial performance, tailor marketing strategies, and prepare for future success.

**1.2 Problem Definition**

E-commerce businesses face several challenges in effectively analyzing and utilizing sales data to drive growth and improve financial performance. While platforms like Shopify and Square Online provide basic metrics and sales data, these offerings are often limited in scope and lack the depth required for strategic, data-driven decision-making. Without a dedicated analysis tool, business owners and managers struggle with understanding profitability, tracking customer behaviour, and accurately forecasting sales. As a result, they may miss opportunities for growth, fail to address areas of concern, and lack insight into their overall financial health. Currently, most small to medium e-commerce businesses lack the technical expertise or resources to conduct detailed financial analysis or interpret customer data effectively. Moreover, the absence of predictive analytics restricts their ability to anticipate sales trends or prepare for fluctuating demand, which can lead to stockouts, overstocking, and missed revenue opportunities. To address these issues, there is a need for a comprehensive, user-friendly sales analysis tool that provides an integrated view of financial performance, customer behaviour, and predictive insights, enabling e-commerce businesses to make informed, proactive decisions that support sustainable growth.

**1.3 Existing System**

The Existing System involves data mining techniques was conducted to extract knowledge in a data set with information about user’s history associated to an e-commerce website. These datasets are directly mined from Flipkart.com using an online software which converts html documents to data tables. The main purpose to web mine data is to apply a set of descriptive data mining techniques to induce rules that allow data analyst working at ecommerce companies make strategic decisions to boost their sales as well as provide effective customer service. Techniques used to discover patterns are web mining and decision tree algorithms.

**1.4 Proposed System**

The proposed system is an advanced sales data analysis tool designed specifically for e-commerce businesses on platforms like Shopify, Square Online, and similar online retail environments. This tool will integrate seamlessly with existing e-commerce platforms, pulling in real-time sales, customer, and inventory data to provide actionable insights that support strategic decision-making. The tool will feature a dashboard, customer behaviour analysis, and predictive analytics components, creating a holistic solution for understanding and enhancing business performance.

Key Components of the Proposed System:

Sales Dashboard:

The dashboard will present key financial metrics such as total sales, profit margins, average order value (AOV), and sales trends over customizable time periods (daily, weekly, monthly). Interactive data visualizations will make it easy for users to identify patterns, track growth, and spot potential issues. Additionally, users can drill down by product category, region, or time frame, giving a nuanced view of profitability and performance. This dashboard will serve as a comprehensive, accessible snapshot of the business’s financial health.

Customer Behaviour Analysis:

This module will analyse customer data to provide insights into purchasing patterns, average spending, purchase frequency, and customer lifetime value (CLV). The system will segment customers by behaviour, identifying high-value segments such as repeat buyers and high spenders. By understanding popular products and customer preferences, the business can optimize inventory, personalize promotions, and target customer retention strategies. These insights will help businesses build more engaging, loyalty-driven customer relationships.

Predictive Analytics:

Using machine learning algorithms, this module will analyse historical sales and customer data to forecast future sales trends, helping businesses prepare for high-demand periods, seasonal shifts, or changes in customer behaviour. Predictive analytics will highlight emerging growth opportunities, potential risks like declining product demand, and areas that need attention, such as customer churn. These forecasts enable proactive inventory management, budget allocation, and marketing decisions.

**1.5 Requirements Specification**

Software Requirements:

Language : Python 3.6

Operating system : Windows / Linux / macOS

Anaconda Navigator

Jupyter Notebooks

Libraries:

Pandas

Matplotlib

Plotly

Numpy

Stats Models

Streamlit

Pickle

Seaborn

Scikit-Learn

Hardware Requirements:

RAM : 16GB – 32GB

Processor : Intel Core i7

**2. LITERATURE SURVEY:**

1.Siddiqui, M., et al. (2020). *Sales Forecasting in E-commerce: A Machine Learning Approach* (Journal of Business Analytics)

Methodology: This paper applied machine learning algorithms, such as regression and decision trees, to predict sales trends based on historical data. The study used both traditional and machine learning approaches to assess which model best forecasts future sales in e-commerce contexts.

Year of Publishing: 2020

Author(s): Siddiqui et al.

2.Kumar, R., & Sharma, S. (2021). *Time-Series Models for E-commerce Sales Prediction* (International Journal of Marketing Analytics)

Methodology: The authors explored the application of time-series models, specifically ARIMA (Auto Regressive Integrated Moving Average), to forecast e-commerce sales. They compared the ARIMA model's performance with other machine learning algorithms and examined its accuracy in different sales cycles.

Year of Publishing: 2021

Author(s): Kumar and Sharma

3.Ghosh, S., et al. (2019). *Improving Forecast Accuracy with Customer Behaviour Data* (E-commerce Research Journal)

Methodology: This paper incorporated customer behaviour data, such as purchase frequency and product preferences, into predictive sales models to improve forecasting accuracy.

Year of Publishing: 2019

Author(s): Ghosh et al.

4.Smith, J., et al. (2021). *Customer Segmentation for Targeted Marketing* (Journal of Retail Marketing)

Methodology: This paper employed clustering techniques, particularly k-means clustering, to segment customers based on purchasing behaviour, which was used to tailor marketing campaigns and product offerings. The authors conducted case studies with major e-commerce brands to demonstrate the effectiveness of customer segmentation.

Year of Publishing: 2021

Author(s): Smith et al.

5.Chakraborty, S., & Gupta, R. (2021). *Optimizing E-commerce Inventory with Predictive Models* (Journal of Supply Chain Management)

Methodology: The authors discussed the application of predictive models, including machine learning techniques like Random Forest and Support Vector Machines (SVM), to optimize inventory management. Their models predict demand for products, reducing the risk of overstocking or understocking.

Year of Publishing: 2021

Author(s): Chakraborty and Gupta

6.Gupta, A., & Singh, S. (2020). *Real-Time Analytics for E-commerce Decision Making* (Journal of Digital Business)

Methodology: This study emphasized the importance of real-time data analytics in e-commerce, demonstrating how platforms could use real-time insights to adjust marketing campaigns, inventory management, and customer interactions. They proposed a framework for integrating real-time analytics into existing business operations.

Year of Publishing: 2020

Author(s): Gupta and Singh

7.Rani, P., & Yadav, A. (2020). *Clustering Approaches in Customer Behaviour Analysis* (Journal of E-commerce Studies)

Methodology: The authors used clustering methods to segment customers based on purchasing behaviour and identified patterns that can be used for targeted marketing. They applied algorithms like K-means and hierarchical clustering to identify different customer segments.

Year of Publishing: 2020

Author(s): Rani and Yadav

8.Rajput, A., et al. (2022). *Forecasting Demand in E-commerce: A Data-Driven Approach* (International Journal of Logistics and Supply Chain)

Methodology: This paper employed data-driven demand forecasting methods to predict sales based on both historical data and external factors, such as market trends and social media sentiment. They used ensemble models to combine the strengths of multiple forecasting techniques.

Year of Publishing: 2022

Author(s): Rajput et al.

9.Jain, P., & Kumar, S. (2020). *Data Quality in Predictive Models: Challenges and Solutions* (E-commerce Data Science Review)

Methodology: This study focused on the challenges related to data quality in predictive modelling for e-commerce. It provided insights into how businesses can ensure high-quality data by implementing better data cleaning processes, handling missing values, and mitigating noise.

Year of Publishing: 2020

Author(s): Jain and Kumar

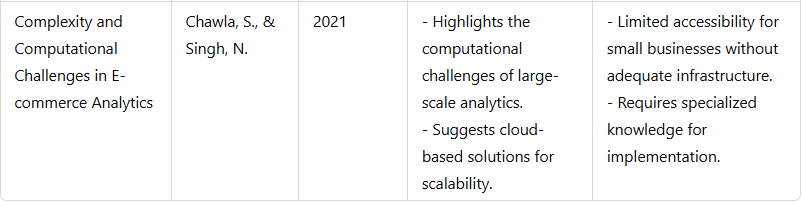
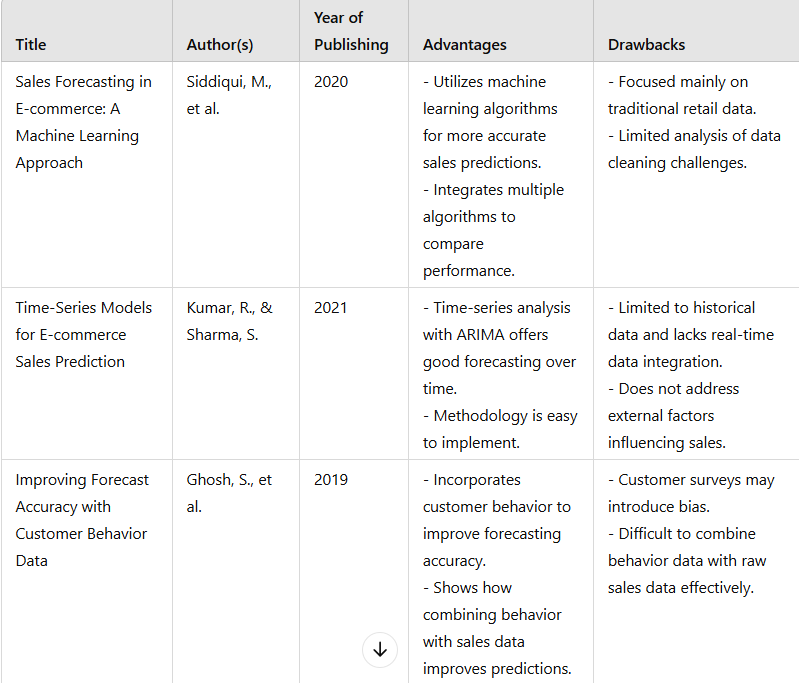
10.Chawla, S., & Singh, N. (2021). *Complexity and Computational Challenges in E-commerce Analytics* (Computational Intelligence Journal)

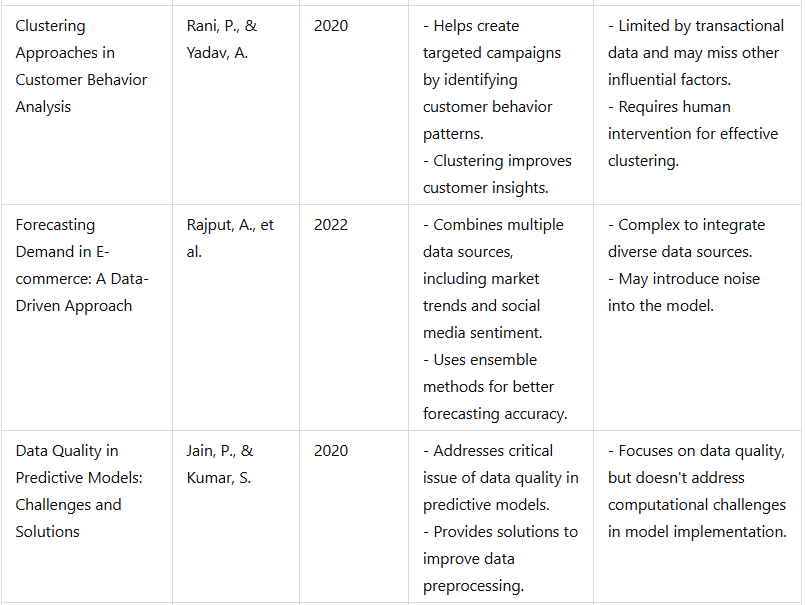
Methodology: The authors explored the computational challenges faced by e-commerce platforms when applying machine learning models to large datasets. They suggested using cloud-based solutions to manage complexity and computational demands.

Year of Publishing: 2021

Author(s): Chawla and Singh

Table 2.1: Comparison of Literature survey





**3. E-COMMERCE SALES ANALYSIS**

Development of a sophisticated tool which can perform sales analysis and provide actionable insights for the financial performance of the org in the digital market it can monitor sales data and customer feedback or reviews or their amount of usage of the organisation’s website based using business analytics.

Features:

1.Sales dashboard which will be built using key metrics like total sales, profit margin, trends

2.Customer Behaviour Analysis which show frequency of purchasing a particular item, amount spent, popular prod according to the trend or demand

3.Predictive Analysis: future sales, growth, risks, opportunities

Ecommerce sales analysis is used for performance metrics, customer behaviour, conversion rates for advertisement by using different attributes according to the requirement. we can conduct this by using various business analytical tools like google analytics, excel, power bi, Qlik etc., or python libraries

Difference from existing:

1.Advanced analytics

2.Usage of data from various e-commerce websites

3.Predictive analysis

4.Dashboards

5.Automation

**Data Collection:**

1.Kaggle

2.UCI Machine Learning Repository

3.[data.gov.in](http://data.gov.in): The official open data portal of the Indian government, providing access to various datasets from different departments and agencies.

4.National Informatics Centre (NIC): Offers datasets related to various sectors, including e-commerce and digital economy.

5.Department of Commerce: May provide data related to foreign trade, domestic trade, and retail sector, which can be relevant for e-commerce analysis.

**3.1 AMAZON SALES ANALYSIS:**

**Key metrics:**

1.Sales: Total revenue, average order value, conversion rate

2.Traffic: Website visitors, page views, bounce rate

3.Customer Behaviour: Customer lifetime value, purchase frequency, cart abandonment rate

4.Marketing Performance: Campaign effectiveness, ROI, customer acquisition cost

5.Product Performance: Best-selling products, product profitability, inventory levels

What can be the frameworks for building the analysis:

1.Data Analysis Tools: Python (Pandas, NumPy), R

2.Visualization Tools: Tableau, Power BI, Plotly, Matplotlib (Python)

3.Frameworks for Interactive Dashboards: Streamlit (Python), Dash (Python)

**Analysis and Metrics:**

1.Total Sales: Calculate using aggregation functions (SUM) in SQL or Pandas.

2.Profit Margin: Calculate by subtracting costs from revenue and dividing by revenue.

3.Trends: Time series analysis with moving averages, exponential smoothing, or ARIMA models.

**Understanding the Data:**

1. index (int64):

Represents a unique identifier for each row in the dataset. It serves as an indexing mechanism for data organization and is automatically generated unless specified otherwise.

2. Order ID (object):

A unique identifier assigned to each order. This helps track and differentiate between individual transactions.

3. Date (object):

Represents the date of the order. Typically formatted as a string but should be converted to a date/time format for analysis of time-series data like trends and seasonality.

4. Status (object):

Indicates the current state of the order, such as "Shipped," "Delivered to Buyer," "Cancelled," etc. This helps in filtering orders by fulfilment status.

5. Fulfilment (object):

Refers to the type or mode of fulfilment, "merchant," "Amazon,” This attribute can be used to analyse fulfilment preferences and efficiency.

6. Sales Channel (object):

The platform through which the sale was made,” Amazon.in.” Useful for understanding which channels drive more sales.

7. ship-service-level (object):

Specifies the shipping service level chosen for the order, such as "Expedited" or "Standard." Important for analysing shipping preferences and associated costs.

8. Category (object):

Denotes the product category, such as "T-shirt," "Shirt," etc. Helps segment sales data by product type.

9. Size (object):

Refers to the size attribute of the product, "S," "M," "L", etc. This can be useful for inventory and demand planning.

10. Courier Status (object):

Represents the delivery status of the order, such as "Shipped", "Cancelled". Critical for analysing delivery performance and delays.

11. Qty (int64):

The quantity of items purchased in the order. Useful for understanding order size and demand trends.

12. currency (object):

Specifies the currency used for the transaction, such as” INR.” Important for financial analysis in multi-currency environments.

13. Amount (float64):

The total amount for the order, expressed in the specified currency. Central to revenue and profitability analysis.

14. ship-city (object):

The city to which the order was shipped. Useful for geographic analysis of sales.

15. ship-state (object):

The state to which the order was shipped. Enables state-level analysis of sales trends and performance.

16. ship-postal-code (float64):

The postal code for the shipment's destination. Can help with granular geographic analysis and route optimization.

17. ship-country (object):

The country to which the order was shipped. Essential for understanding international sales and shipping patterns.

18. B2B (bool):

Indicates whether the transaction is a Business-to-Business (B2B) order (True) or not (False). Useful for segmenting sales into B2B and Business-to-Consumer (B2C).

19. fulfilled-by (object):

Specifies the party responsible for fulfilling the order, such as "Amazon," "Merchant," or "Easy Ship", Helps analyse the performance of fulfilment partners. Note the high number of null values, which may need cleaning or imputing.

**Algorithm Study:**

The Streamlit library is a popular Python framework designed to quickly build and share data-driven web applications. Its simplicity and efficiency make it an ideal choice for projects where you need an interactive UI for data visualization, analysis, or machine learning applications.

Key Uses of Streamlit

Interactive Dashboards: Streamlit makes it easy to create dynamic dashboards for visualizing data with components like sliders, charts, maps, and tables. For example, you can build dashboards for sales metrics, trends, and customer analytics.

Machine Learning Model Deployment: Streamlit simplifies deploying machine learning models by allowing users to interact with models in real-time, such as uploading input data and visualizing predictions or outputs.

Exploratory Data Analysis (EDA): It helps users explore datasets interactively by allowing them to adjust parameters, filter data, and generate insights through charts and tables.

Customizable User Interfaces: With widgets like checkboxes, dropdowns, and sliders, you can create tailored interfaces for various use cases, such as parameter tuning or filtering large datasets.

Integration with Visualization Libraries: Streamlit supports popular visualization libraries such as Matplotlib, Seaborn, Plotly, and Altair, making it easy to integrate complex visualizations into apps.

Real-time Data Applications: The st.cache and st.session\_state features allow for efficient real-time updates, enabling apps to respond dynamically to user input or external data streams.

Advantages of Streamlit

Quick Development: You can go from a Python script to a fully functional app in a matter of minutes.

No Frontend Knowledge Required: Streamlit abstracts away HTML, CSS, and JavaScript complexities.

Lightweight and Open-Source: The library is lightweight and free to use, making it accessible for small projects and startups.

Integration with Python Ecosystem: Directly supports Pandas, NumPy, Scikit-learn, TensorFlow, and other Python libraries.

**3.2 CUSTOMER SEGMENTATION AND BEHAVIORL ANALYSIS:**

**Key Metrics:**

1.Data Analysis Tools: Python (Pandas, Scikit-learn), R

2.Visualization Tools: Tableau, Power BI, Plotly

**Analysis and Metrics:**

1.Frequency of Purchasing: Count purchases per customer.

2.Amount Spent: Calculate total spending per customer.

3.Popular Products: Identify most frequently purchased products or analyse with RFM (Recency, Frequency, Monetary) segmentation.

**Understand the Data:**

1. Invoice (object):

A unique identifier for each transaction or invoice.

Represents the purchase order issued during a transaction, which may involve one or multiple items.

Important for tracking sales and linking transactions to customers.

2. Stock Code (object):

A unique code assigned to each product in the inventory.

Used for identifying specific items and linking them to their descriptions, prices, and other attributes.

Helps with inventory management and sales analysis.

3. Description (object):

The textual description of the product associated with the Stock Code.

Provides insight into the nature of the product sold.

4. Quantity (int64):

The number of units purchased for each product in the invoice.

Positive values represent sales, while negative values (if present) could indicate returns or cancellations.

Useful for understanding demand trends and inventory planning.

5. Invoice Date (object):

The date and time when the invoice is generated.

Crucial for time-series analysis, such as identifying sales trends, seasonality, or peak shopping periods.

6. Price (float64):

The price of a single unit of the product.

Used to calculate total revenue and analyse pricing strategies.

7. Customer ID (float64):

A unique identifier for each customer.

Helps in customer segmentation, behaviour analysis, and calculating metrics such as Customer Lifetime Value (CLV).

8. Country (object):

The country where the customer is located or where the invoice was generated.

Important for geographic sales analysis and understanding market distribution.

Useful in international market performance evaluations and logistics planning.

**Algorithm Study:**

RFM is a combination of the initials of the terms Recency, Frequency and Monetary, each of which deals with a different characteristic of the customer. These characteristics are as follows;

Recency is calculated by subtracting the date of the last interaction (shopping) of the customers from the date of the analysis (if it is a historical data set, the date of the analysis is determined close to the date of data collection), giving the date difference. The more recently a customer has shopped, the more likely they are to keep the brand (company) in mind for future purchases. It can also be used to identify customers who have not shopped for a long time and encourage them to visit the store again.

Frequency indicates the number of times a customer makes a purchase. If it can be recognized that the customer's purchases are in a cycle, actions can be taken to predict when the customer will come back to the store or to remind them of their needs.

The monetary value is the money left by the customer because of these purchases, indicating the total expenditure. While the monetary value can identify the customers who spend the most, it also carries the risk of alienating customers who spend relatively small amounts or new customers.

**3.3 PREDICTIVE TIME SERIES ANALYSIS:**

**Key Metrics:**

1.Data Analysis Tools: Python (Scikit-learn, stats models), R (forecast package)

2.Machine Learning Tools: Linear regression, time series forecasting (ARIMA, Prophet)

**Analysis and Metrics:**

1.Predictive Sales: Use historical data to forecast future sales.

2.Growth: Analyse sales trends to identify potential growth patterns.

3.Risks: Identify potential factors that could negatively impact sales (e.g., seasonality, competition).

4.Opportunities: Identify opportunities for sales growth (e.g., new product launches, marketing campaigns).

**Understand the Data:**

1. Date (object):

Represents the specific day the financial data corresponds to.

It is crucial for time-series analysis, as it allows for the examination of trends, seasonality, and historical performance.

Should be converted to a datetime format for accurate plotting and analysis.

2. Open (float64):

The opening price of the stock or asset at the start of the trading session for the specified date.

Useful for understanding the initial market sentiment and comparing with other metrics like Close to identify daily trends.

3. High (float64):

The highest price of the stock or asset during the trading session.

Indicates the peak value reached in a day and helps identify volatility and intraday price spikes.

4. Low (float64):

The lowest price of the stock or asset during the trading session.

Complements the High value in understanding the range of price movement within the trading session.

5. Close (float64):

The final price of the stock or asset at the end of the trading session.

Frequently used for trend analysis, technical indicators, and financial modelling.

6. Adj Close (float64):

The adjusted closing price accounts for corporate actions like dividends, stock splits, or other adjustments.

Provides a more accurate representation of the stock's value and is often preferred for long-term analysis and comparisons.

7. Volume (int64):

The total number of shares or contracts traded during the trading session.

Serves as a measure of market activity and liquidity. High volume often indicates significant market interest or reaction to news/events.

**Algorithm Study:**

1.Rolling Mean (Moving Average)

Definition: The rolling mean is the average of a specified number of data points within a defined "window" that moves over the data set. For example, a rolling mean of 7 days for daily data would average the values of each 7-day window as it moves forward day by day.

Purpose: It smooths out short-term fluctuations, making it easier to observe trends and patterns over time. This is particularly useful in identifying the underlying trend in noisy time series data.

adata['Rolling\_Mean'] = adata['Value']. rolling(window=7). mean ()

2.Rolling Standard Deviation

Definition: The rolling standard deviation measures the amount of variation or dispersion of a set of values within a rolling window. It moves along the data set, just like the rolling mean.

Purpose: It is helpful to understand the volatility of a time series. High rolling standard deviation indicates more variability or volatility within that window, while a low rolling standard deviation indicates stability.

adata['Rolling\_Std'] = adata['Value']. rolling(window=7). std ()

In stock price analysis rolling mean gives the general trend in price and rolling deviation indicates changes in volatility

3.Augmented Dickey-Fuller (ADF) Test:

Dickey-Fuller Test, specifically the Augmented Dickey-Fuller (ADF) Test, is a statistical test used to determine if a time series is stationary. Stationarity means the statistical properties (mean, variance, autocorrelation) of a time series do not change over time, which is an essential assumption for many time series models, including ARIMA.

Key Concepts

* Null Hypothesis (H₀): The time series has a unit root (i.e., it is non-stationary).
* Alternative Hypothesis (H₁): The time series does not have a unit root (i.e., it is stationary).

If the test rejects the null hypothesis, it indicates that the time series is stationary, making it suitable for time series modelling without needing additional transformations.

Interpreting Results

* p-value: If the p-value is less than a chosen significance level (e.g., 0.05), you can reject the null hypothesis and conclude that the time series is likely stationary.
* Test Statistic: This value is compared with critical values for various confidence levels (1%, 5%, and 10%). If the test statistic is less than the critical value, the null hypothesis is rejected, suggesting stationarity.

4.Log transformation

A log transformation is a data transformation technique commonly used in time series and statistical analysis to stabilize the variance, reduce skewness, and make the data more normally distributed. It is particularly helpful when dealing with non-stationary time series data with exponential growth or with data that has outliers.

Why Use a Log Transformation?

1. Reduce Variance: Log transformation can reduce the scale of large values, which is useful if your data has high variance.

2. Make Data Stationary: It can help in achieving stationarity by transforming an exponentially growing series into a more linear form.

3. Handle Skewness: Log transformation can make right-skewed distributions more symmetric, making the data more suitable for certain analyses.

4. Manage Outliers: By compressing larger values more than smaller values, log transformation can lessen the impact of outliers.

5.SARIMAX (Seasonal Auto Regressive Integrated Moving Average with exogenous factors) is an advanced time series forecasting model that extends ARIMA by incorporating seasonal components and the ability to handle external (exogenous) variables. It is particularly useful for datasets with patterns of seasonality and trends influenced by additional explanatory factors.

Key Components of SARIMAX

SARIMAX builds upon the ARIMA model, so it shares these components:

AR (Auto Regressive): Past values are used to predict the current value.

I (Integrated): Differencing is applied to make the data stationary.

MA (Moving Average): Past forecast errors are used to model the series.

In addition, SARIMAX includes:

4. Seasonality: Captures repeating patterns or cycles in the data, specified by seasonal ARIMA terms (P, D, Q, m):

P: Seasonal autoregressive order.

D: Seasonal differencing order.

Q: Seasonal moving average order.

m: Number of observations per seasonal cycle (e.g., 12 for monthly data with yearly seasonality).

Exogenous Variables (X): External predictors or covariates that influence the time series.

SARIMAX Model Notation

SARIMAX(p,d,q)×(P,D,Q,m)SARIMAX(p, d, q) \times (P, D, Q, m)SARIMAX(p,d,q)×(P,D,Q,m)

Where:

(p, d, q) are the ARIMA terms.

(P, D, Q, m) are the seasonal terms.

**3.4 UML DIAGRAMS:**

**For Sales Analysis:**

**3.4.1.1 Data Flow Diagram:**

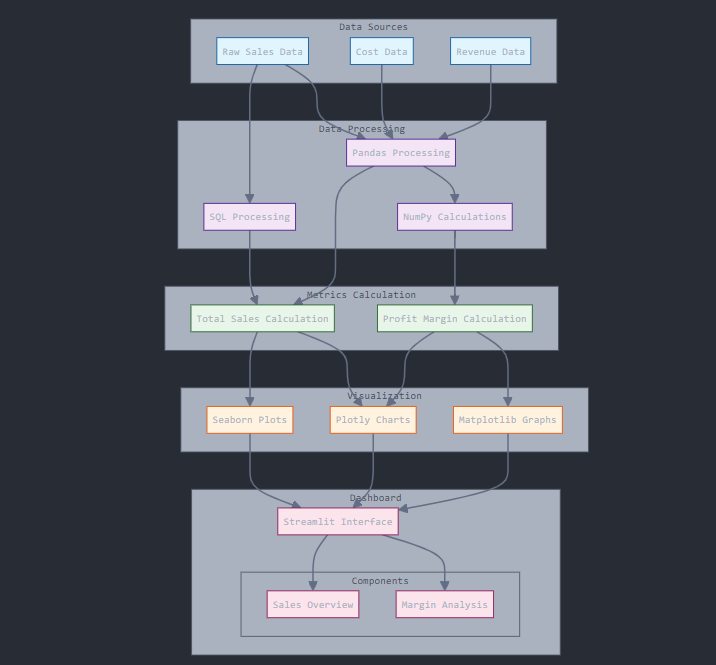


Figure 3.4.1.1

The data flow begins with sources such as raw sales data, cost data, and inventory data feeding into the processing layer, were Python scripts and SQL queries handle data cleaning and transformation. Analysis tools like Pandas and NumPy perform data manipulation and numerical computations, complemented by time series models for trend analysis. Metrics such as total sales, profit margins, and trend insights are generated for actionable insights. The visualization layer employs tools Plotly, and Matplotlib to create compelling visuals, while interactive dashboards are developed using Streamlit or Dash to enable user-friendly exploration of the insights.

**3.4.1.2. Sequence Diagram:**

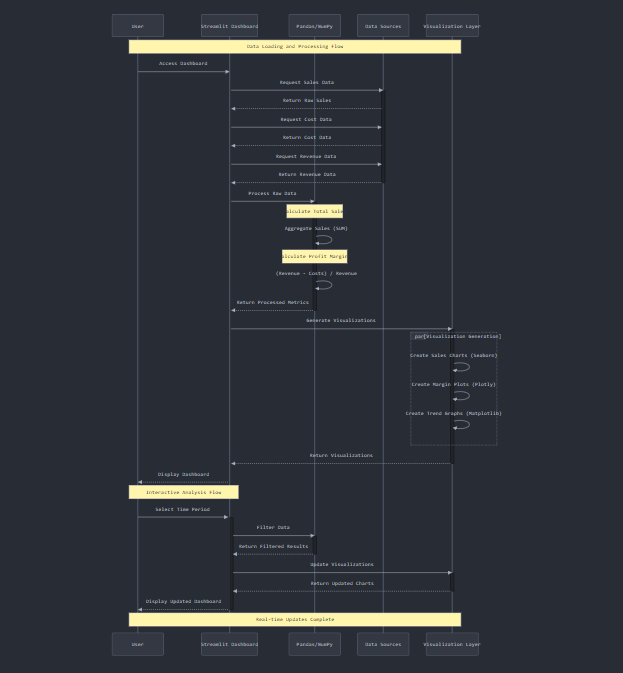


Figure 3.4.1.2

The sequence begins with the user accessing the dashboard, prompting the system to retrieve sales, cost, and revenue data for processing using Pandas and NumPy. Metrics such as total sales and profit margins are calculated, followed by the parallel generation of various visualizations using Seaborn, Plotly, and Matplotlib. These charts are then displayed on the dashboard. Interactive analysis enables users to select specific time periods, triggering real-time data filtering and updates to the visualizations, ensuring a dynamic and responsive analytical experience.

**3.4.1.3 Activity Diagram:**

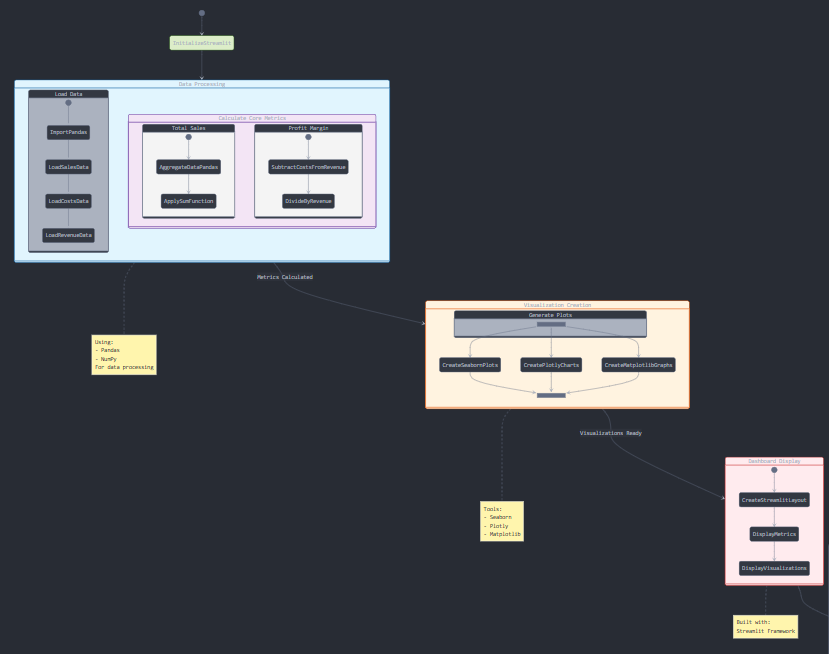


Figure 3.4.1.3

The activity begins with the initial setup, where the dashboard is initialized, and the data collection process is triggered. Sales, cost, and revenue data are loaded, followed by data validation checks and outlier handling to ensure accuracy. In the data processing phase, sales are aggregated, margins calculated, and key metrics computed. Visualization generation involves the parallel creation of various chart types, their assembly into the dashboard, and the integration of visuals. Interactive features enable users to provide inputs, update filters, trigger real-time data refreshes, and dynamically update visualizations, delivering a seamless and engaging analytical experience.

**3.4.1.4 Collaboration Diagram:**

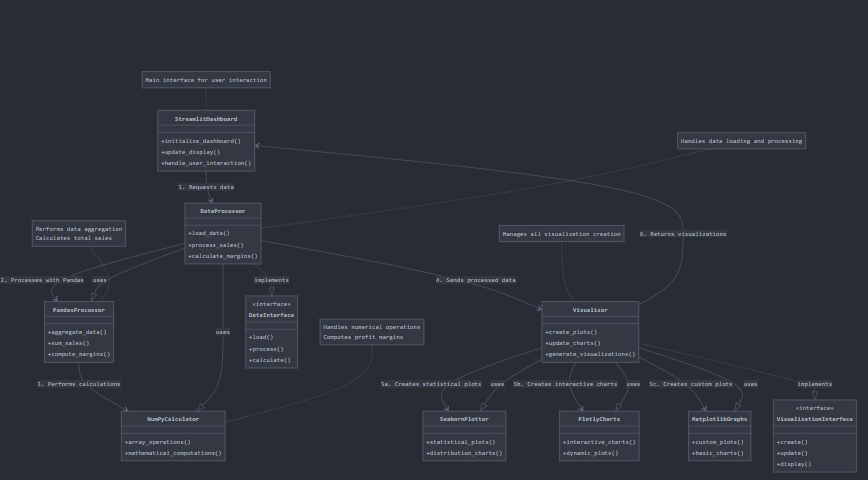


Figure 3.4.1.4

The collaboration diagram highlights the core components and their interactions, starting with the Streamlit Dashboard as the main interface facilitating user interaction and managing display updates. The Data Processor handles data loading, validation, and management by leveraging Pandas for data aggregation and sales calculations and NumPy for numerical operations such as profit margin computations. The Visualizer, which integrates tools like Seaborn, Plotly, and Matplotlib, generates various types of charts. The Dashboard requests data from the Data Processor, which performs calculations and passes results to the Visualizer for chart creation, ultimately returning the visual insights to the Dashboard for user display. Each component focuses on a specific responsibility: the Streamlit Dashboard manages user interface and interactivity, the Data Processor ensures data integrity and processing, and the Visualizer manages chart generation and presentation.

**For Customer Segmentation and Behavioral Analysis:**

**3.4.2.1. Data Flow Diagram:**

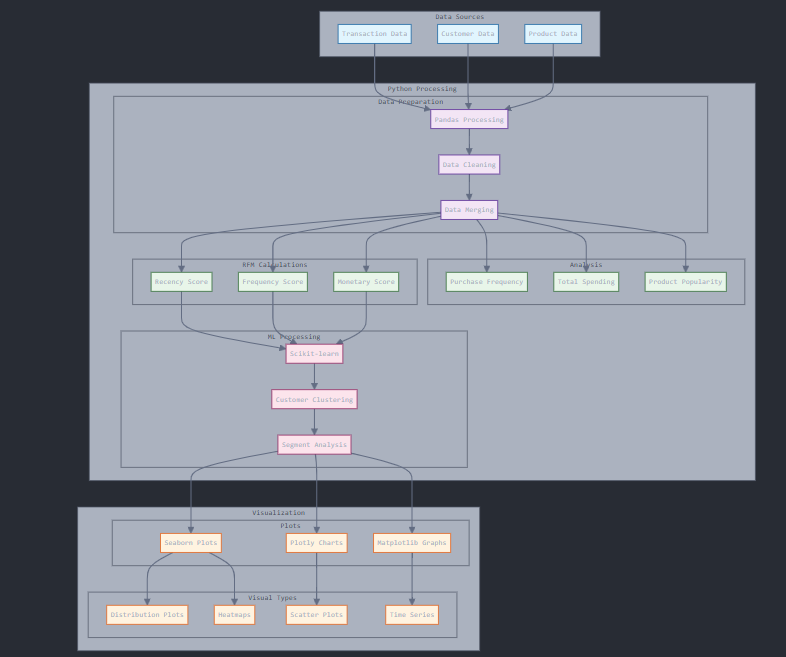


Figure 3.4.2.1

The data flow begins with sources like transaction, customer, and product data feeding into the processing layer, where Python with Pandas and Scikit-learn performs advanced analysis, including RFM (Recency, Frequency, Monetary) calculations. Metrics such as purchase frequency, total spending, and product popularity are generated. In the machine learning layer, customer segmentation, clustering analysis, and pattern recognition provide deeper insights into behaviour. The visualization layer employs tools like Tableau, Power BI, and Plotly to create interactive dashboards and charts. The analysis outputs deliver actionable insights, highlighting customer behaviour patterns, segmentation results, popular products, and product combination trends.

**3.4.2.2 Sequence Diagram:**

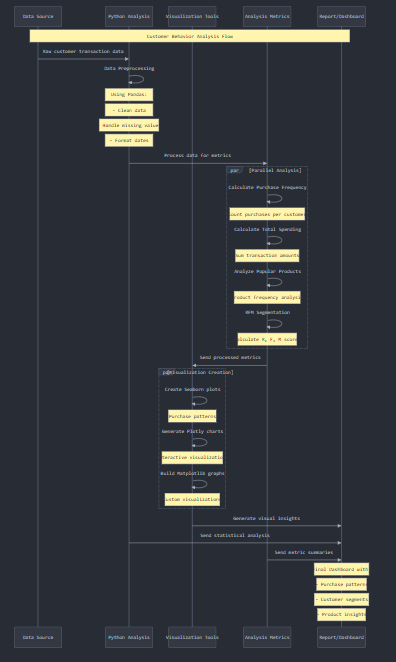


Figure 3.4.2.2

The sequence diagram outlines the flow starting with the retrieval of transaction, customer, and product data during the initial data collection phase. The RFM analysis process calculates recency, frequency, and monetary values, forming the foundation for advanced analysis, including clustering with Scikit-learn, customer segmentation, and product popularity insights. Visualization generation occurs in parallel, Plotly interactive charts. The interactive analysis flow supports segment-specific insights with real-time visualization updates. Key features include parallel processing of visualizations, a clear separation of processing steps, and bidirectional communication between components to ensure seamless data flow. The diagram highlights activation periods for each component, timing of operations, and annotations explaining critical interactions across the workflow.

**3.4.2.3 Activity Diagram:**

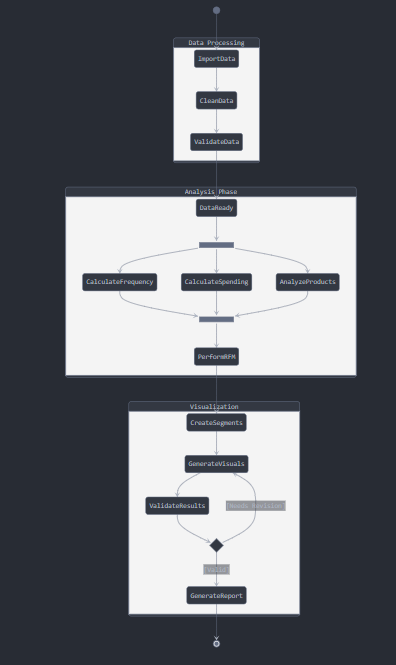


Figure 3.4.2.3

The UML activity diagram is structured into three main partitions: Data Processing, Analysis Phase, and Visualization, with clear state transitions and control flows. It begins at the initial node, where data collection and validation occur in the Data Processing partition. Fork and join nodes manage parallel activities, such as loading transaction, customer, and product data while performing validation checks. The Analysis Phase includes parallel activities like RFM analysis, clustering, and customer segmentation, leveraging nested states to show the interdependencies of these computations. The workflow transitions to the Visualization partition, where chart generation is performed using Plotly. A decision point evaluates validation success, determining whether to finalize the visualizations or return to previous phases for corrections. The diagram concludes at the final node, with all activities organized within swim lanes to reflect distinct responsibilities for data processing, analysis, and visualization.

**3.4.2.4 Collaboration diagram:**

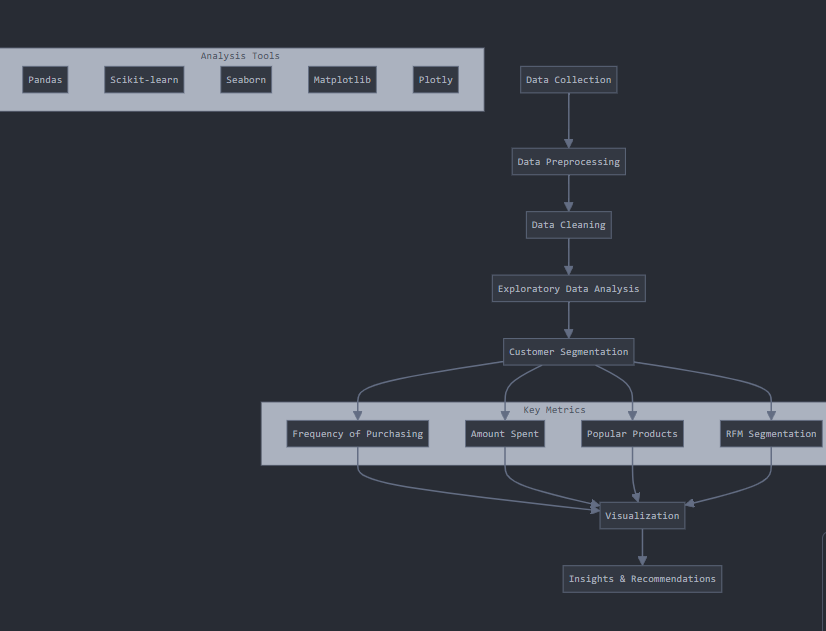


Figure 3.4.2.4

The collaboration diagram features objects representing key components of the data analysis pipeline, including AnalysisController, DataProcessor, CustomerMetrics, RFMCalculator, ProductAnalytics, and Visualizer. The process begins with the AnalysisController initiating the analysis (1), triggering DataProcessor to handle raw data using Pandas (2). Processed data is passed to CustomerMetrics for calculating purchase frequency and spending (3), which is then forwarded to RFMCalculator for segmentation (4). Simultaneously, ProductAnalytics uses customer data to analyze product popularity (5). The results are sent to the Visualizer, which generates visualizations for RFM segmentation and product metrics using Plotly (6). The diagram illustrates one-to-one relationships between components, follows the sequence of the data pipeline, and clearly indicates the direction of communication through numbered message flows.

**For Predictive Time Series Analysis:**

**3.4.3.1 Data Flow Diagram**

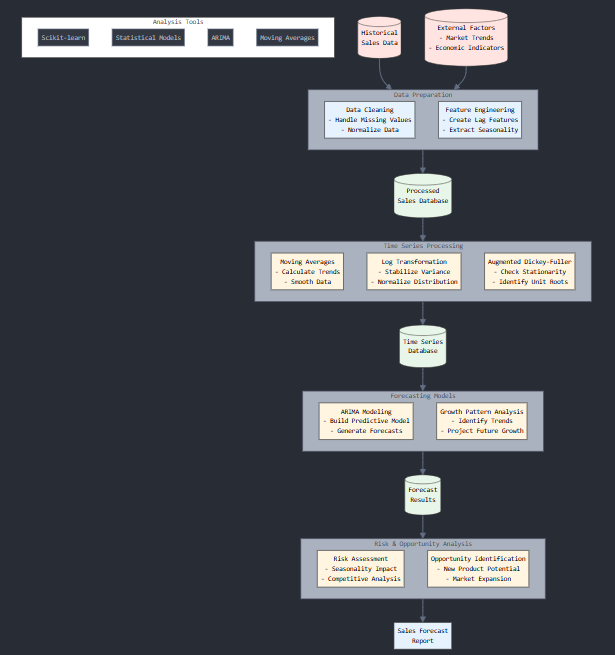


Figure 3.4.3.1

The comprehensive data flow diagram for the Predictive Sales Analysis system illustrates the seamless interaction between its components. It begins with Data Sources, including Historical Sales Data and External Factors such as market trends and economic indicators. These inputs are processed through Key Processing Stages, comprising Data Preparation, Time Series Processing, Forecasting Models, and Risk & Opportunity Analysis, ensuring refined data for analysis. The processed data is stored in dedicated Data Storage Points, such as the Processed Sales Database, Time Series Database, and Forecast Results Database, to facilitate efficient retrieval and future analyses. The Analysis Components include sophisticated techniques like Moving Averages, Log Transformation, the Augmented Dickey-Fuller Test, ARIMA Modelling, Growth Pattern Analysis, Risk Assessment, and Opportunity Identification, which together deliver actionable insights. To enhance clarity, the diagram employs Visualization Features, including color-coded components to distinguish stages, clear data flow arrows to illustrate the process, and descriptive process labels for better understanding.

**3.4.3.2 Sequence Diagram:**

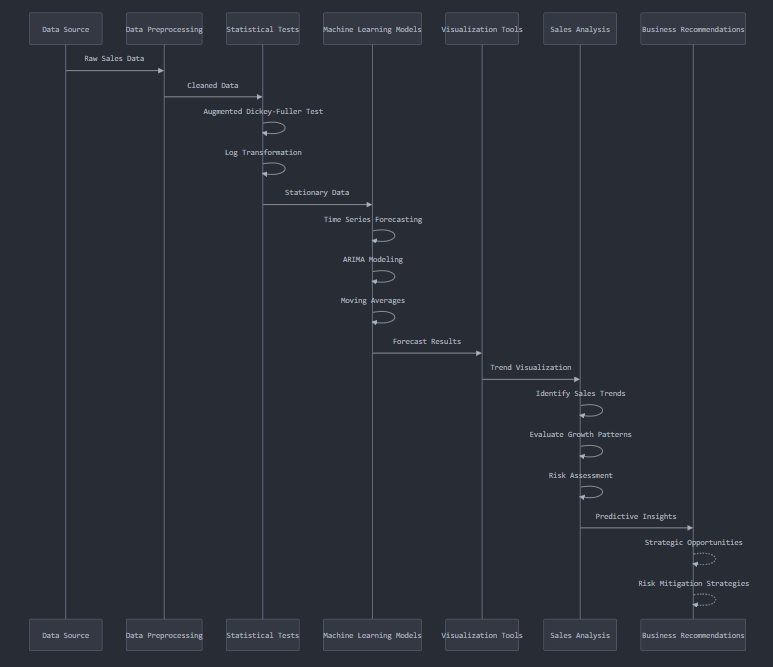


Figure 3.4.3.2

The sequence diagram for Predictive Sales Analysis begins with the ingestion of raw sales data from the data source, which undergoes preprocessing to clean and structure it for analysis. Statistical tests, including the Augmented Dickey-Fuller test, are conducted to assess stationarity, followed by log transformation to normalize the data, preparing it for machine learning modeling. In the modeling phase, time series forecasting techniques, such as ARIMA modeling and moving average calculations, are applied to predict sales trends. The results are visualized to reveal forecast insights, growth patterns, and risks, providing a foundation for actionable business recommendations. These include identifying strategic opportunities and developing risk mitigation strategies to support informed decision-making.

**3.4.3.3 Activity Diagram:**

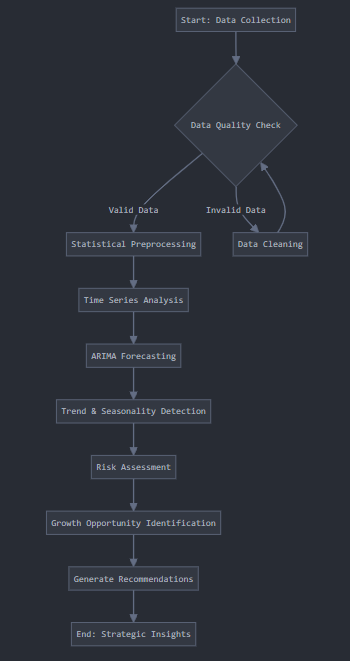


Figure 3.4.3.3

The activity diagram for Predictive Sales Analysis begins with data collection, followed by a data quality check to ensure accuracy and reliability. The process transitions to statistical preprocessing, including steps like normalization and stationarity checks, to prepare the data for analysis. Time series techniques are then applied, with ARIMA forecasting used to model and predict sales trends. The analysis detects underlying trends and seasonality, assesses potential risks, and identifies growth opportunities. The final step involves generating strategic recommendations, enabling businesses to make informed decisions based on the predictive insights.

**3.4.3.4 Collaboration Diagram:**

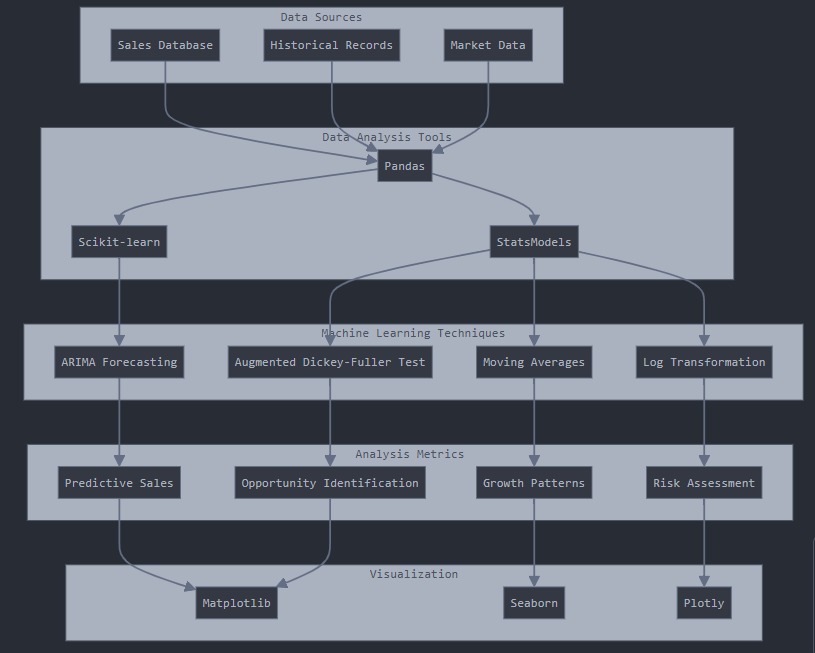


Figure 3.4.3.4

The Predictive Sales Analysis system integrates various components to deliver accurate sales forecasts and insights. It begins with Data Sources such as the Sales Database, Historical Records, and Market Data, which serve as the foundation for analysis. These data inputs are processed using Data Analysis Tools like Pandas for data manipulation, Scikit-learn for machine learning, and StatsModels for statistical evaluations. The system employs sophisticated Machine Learning Techniques, including ARIMA Forecasting for time series predictions, Moving Averages for trend analysis, Log Transformation for data normalization, and the Augmented Dickey-Fuller Test to check for stationarity in time-series data.

The analysis focuses on key Metrics such as Predictive Sales, Growth Patterns, Risk Assessment, and Opportunity Identification, enabling businesses to make data-driven decisions. Finally, the results are presented through intuitive and interactive visualizations created with tools like Matplotlib, Seaborn, and Plotly, ensuring the insights are accessible and actionable for stakeholders. This cohesive framework ensures a comprehensive approach to understanding and predicting sales dynamics.

**4. TESTING AND RESULTS**

For the Prediction of stock prices, we have built and trained the model using ARIMA which is used to understand and analysis the trends with seasonal and economical changes which affect the stock market of the organization (Amazon)

The accuracy of the model has been calculated using the metrics:

1.Mean squared error

2.Mean absolute error

3.Root mean squared error

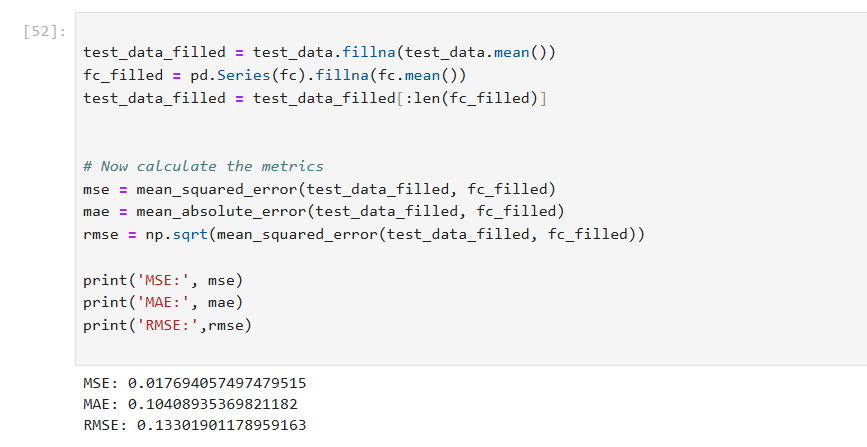


Figure 4.1 Accuracy metrics of ARIMA model

The code in the image shows a process for evaluating the performance of a forecasting model by calculating three error metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Here is an explanation of the key steps:

**Code Explanation:**

1. Filling Missing Values in Test Data:

test\_data\_filled = test\_data.fillna(test\_data.mean())

Any missing values in the test\_data series are replaced with the mean of the series to ensure there are no NaN values during computation.

1. Filling Missing Values in Forecasted Data:

fc\_filled = pd.Series(fc).fillna(fc.mean())

Similarly, the forecasted data (fc) is converted into a pandas Series, and missing values are replaced with the mean of the forecasted series.

1. Adjusting the Length of test\_data\_filled:

test\_data\_filled = test\_data\_filled[:len(fc\_filled)]

The test\_data\_filled series is truncated to match the length of the forecasted data (fc\_filled) to ensure they align during error calculation.

1. Calculating Error Metrics:

Mean Squared Error (MSE):

\_squared\_error(test\_data\_filled, fc\_filled)

This measures the average squared difference between actual values and predicted values. A lower value indicates better accuracy.

Mean Absolute Error (MAE):

mae = mean\_absolute\_error(test\_data\_filled, fc\_filled)

This calculates the average absolute difference between the actual and predicted values. It is more interpretable than MSE because it represents the error in the original units of the data.

Root Mean Squared Error (RMSE):

rmse = np.sqrt(mean\_squared\_error(test\_data\_filled, fc\_filled))

RMSE is the square root of the MSE, which puts the error metric back into the same units as the original data and is sensitive to larger errors.

1. Printing the Metrics:

print('MSE:', mse)

print('MAE:', mae)

print('RMSE:', rmse)

The computed metrics are printed. In this case, the values are:

MSE: 0.017694057497479515 (low indicates a good model)

MAE: 0.10408935369821182 (represents the average absolute error in data units)

RMSE: 0.13301901178959163 (interpretable as the average magnitude of error).

**Purpose:**

This code evaluates the accuracy of the forecasting model by comparing its predictions (fc) against actual test data (test\_data). The metrics help quantify how well the model performs, with smaller values indicating better predictions.

4.1 Results of Sales Analysis:

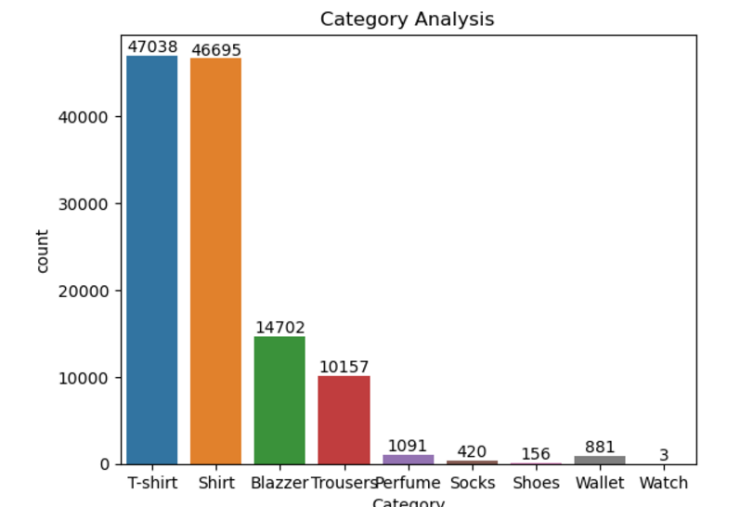
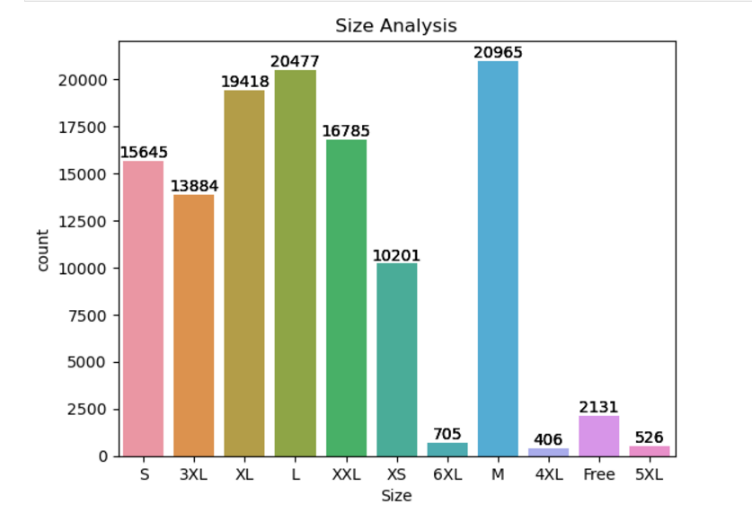


Figure 4.1.1 Size Analysis Figure 4.1.2 Categorical Analysis

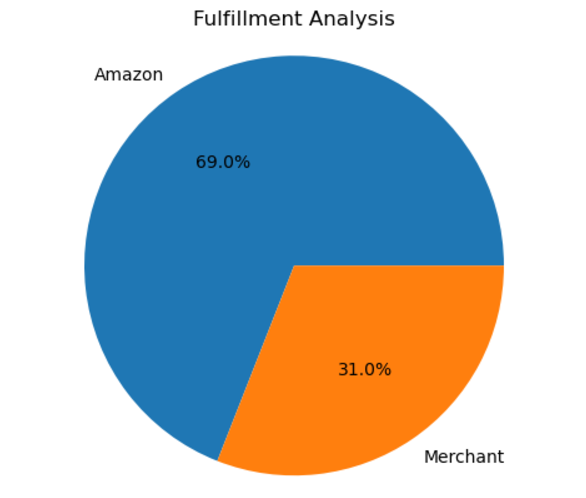
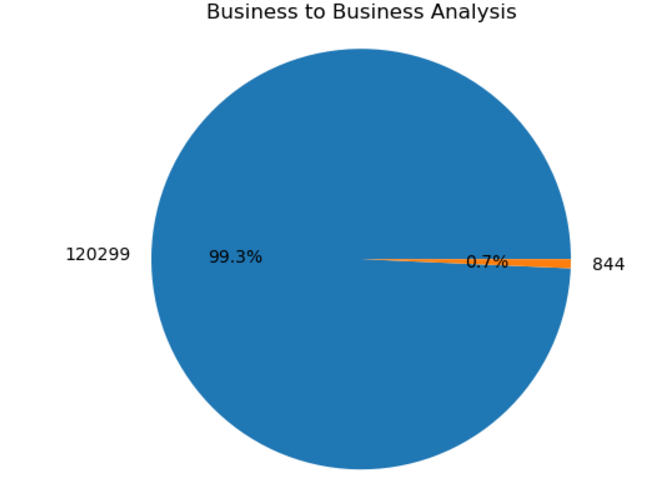


Figure 4.1.3 B2B Analysis Figure 4.1.4 Fulfilment Analysis

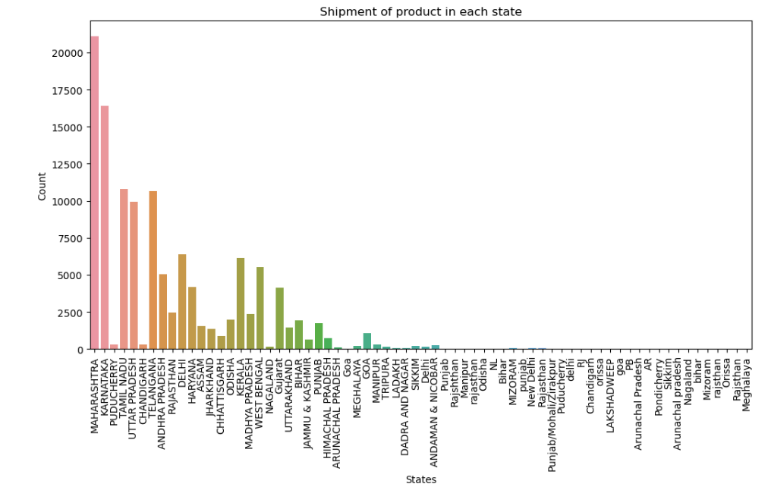


Figure 4.1.5 State Wise Analysis

4.2 Results of Customer Segmentation and Behavioral Analysis:

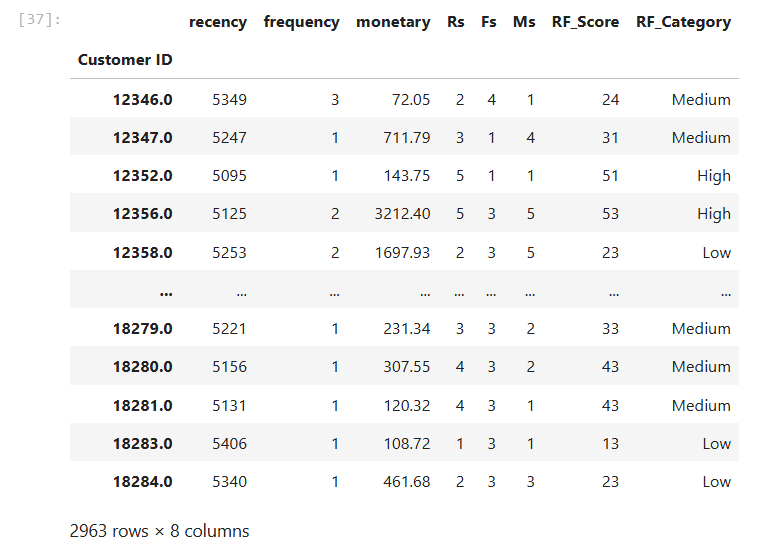


Table 4.2.1 Recency,Frequency and Monetary Calculated Table

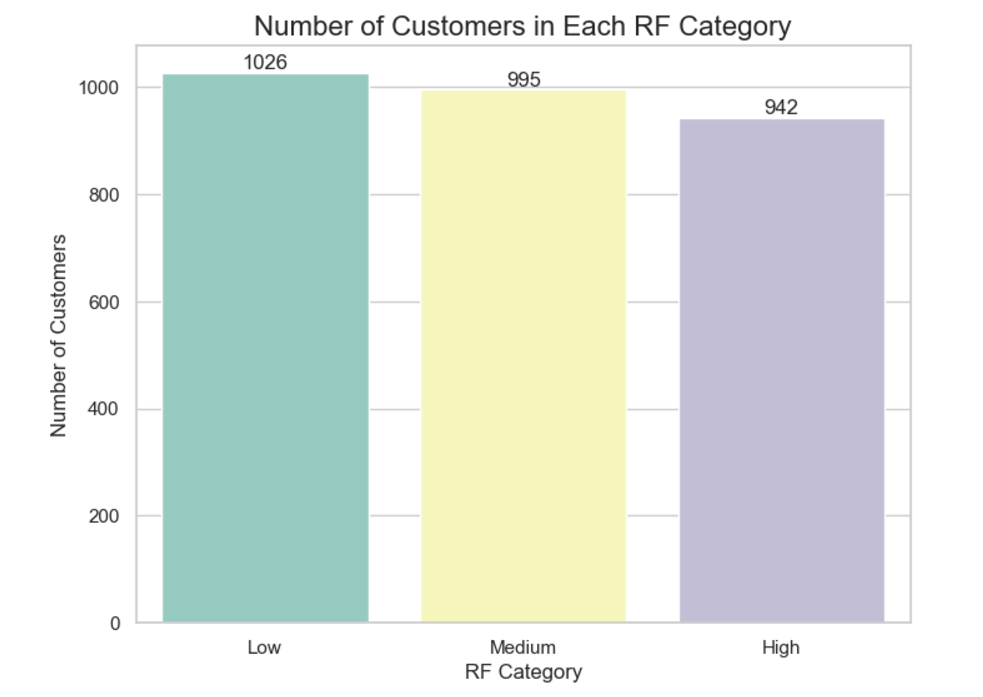


Figure 4.2.1 Customer Segmentation Analysis

4.3 Result of Predictive Time Series Analysis:

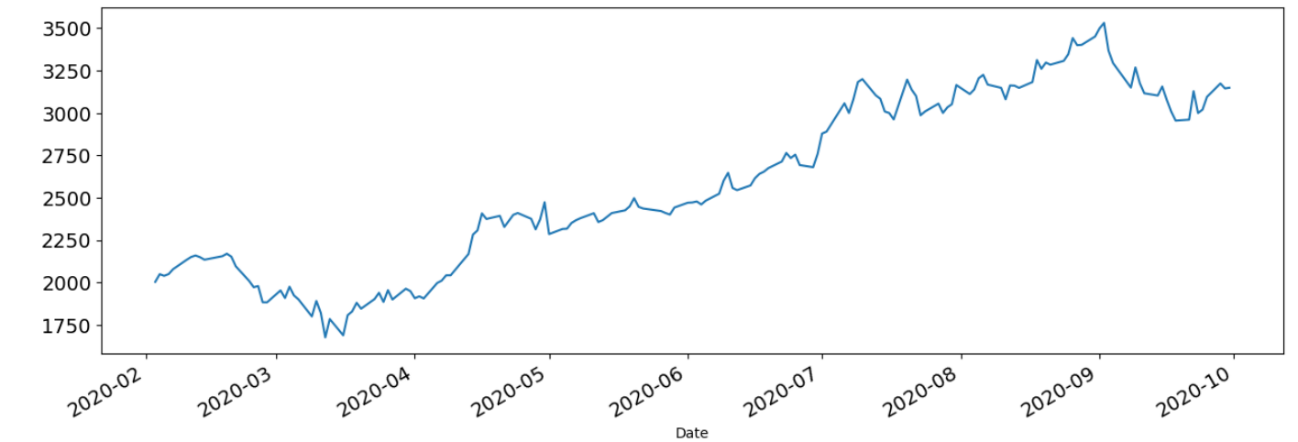


Figure 4.3.1 Drastic fall of stock price due to Covid-19

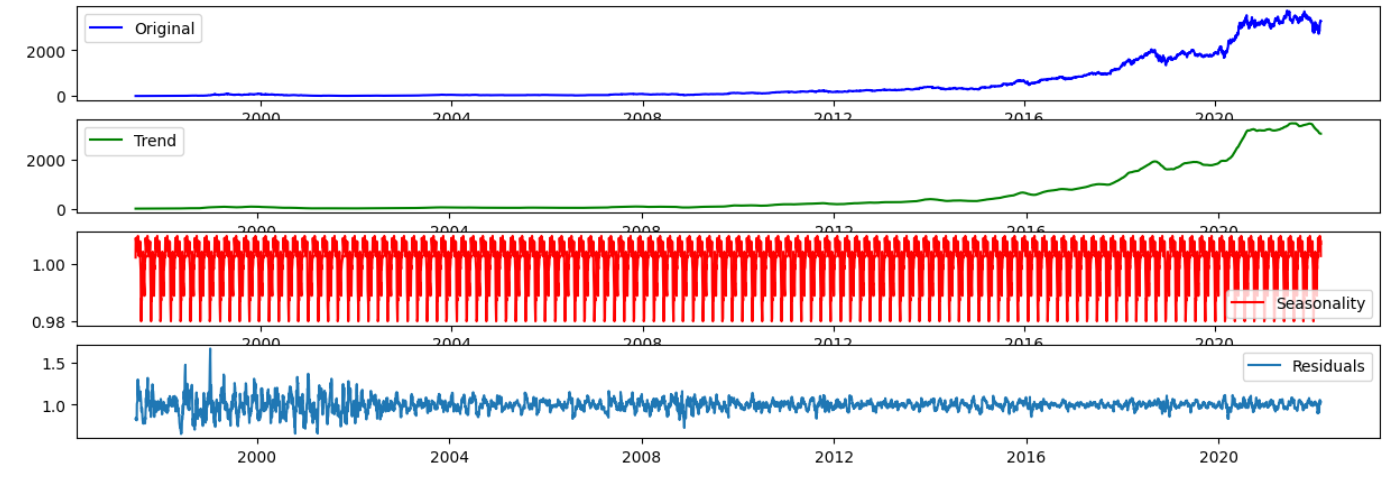


Figure 4.3.2 Decomposition of the Seasonality

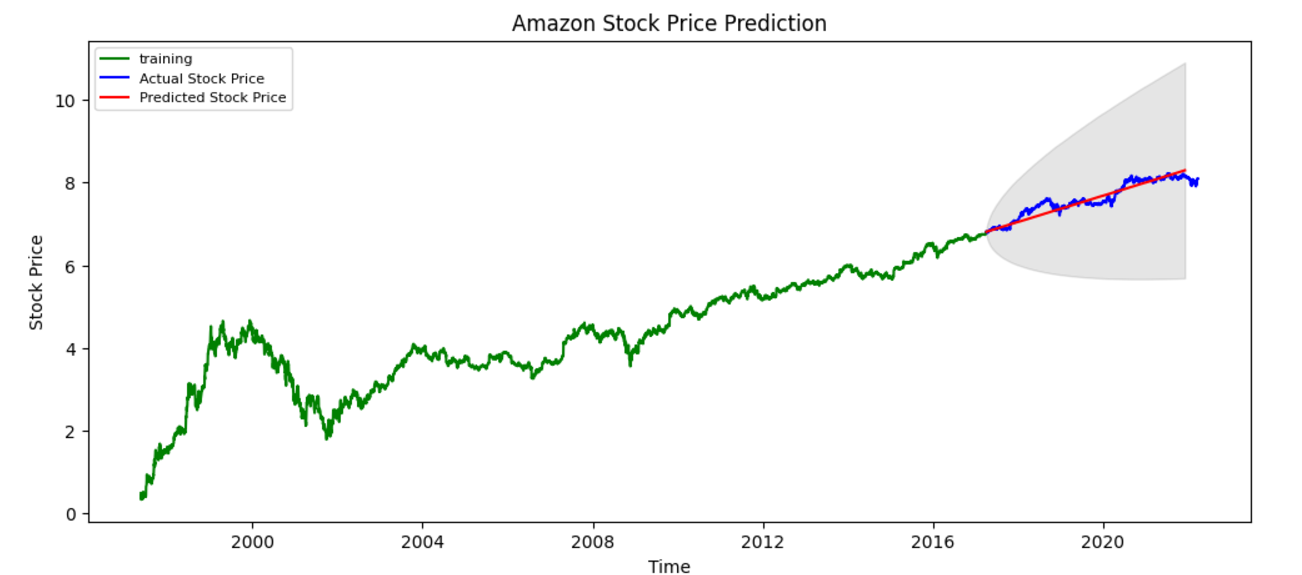


Figure 4.3.3 Forecast of Actual stock price v/s Predicted stock price

**5. Integration with User Interface using Streamlit:**

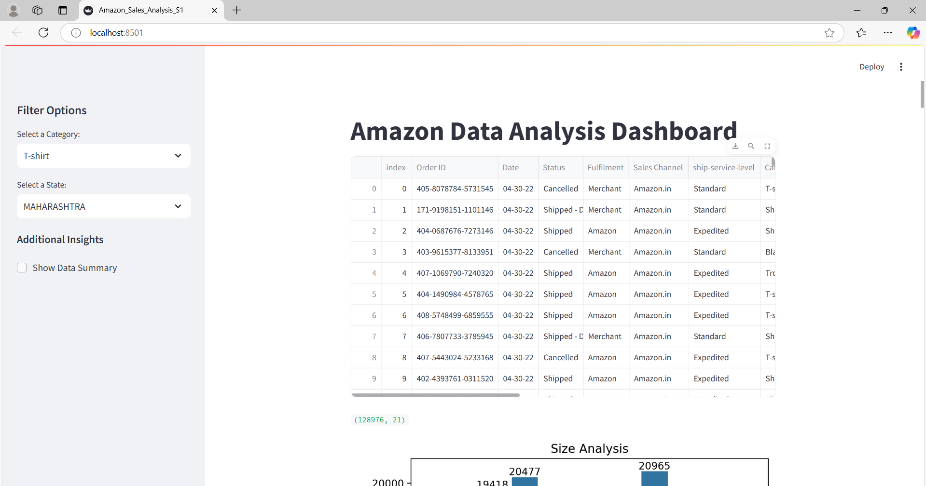


Figure 5.1 Sales Analysis Dashboard

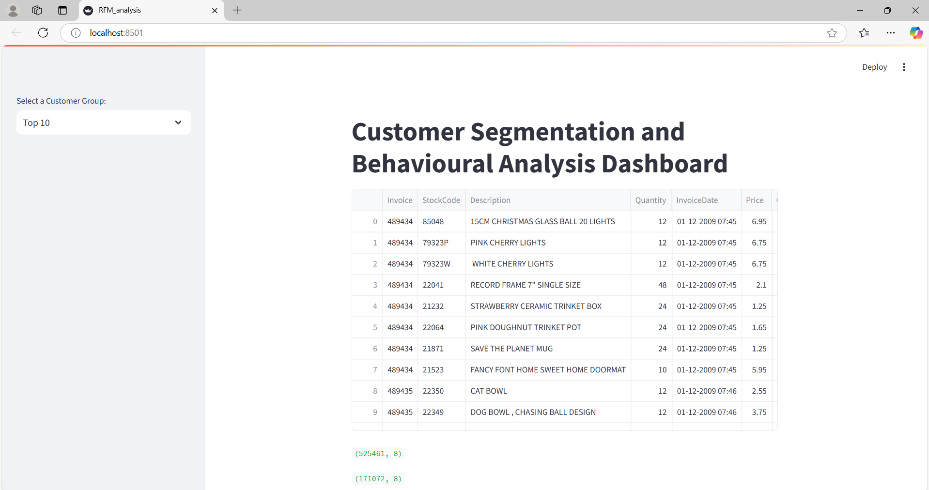


Figure 5.2 Customer Segmentation and Behavioral Dashboard

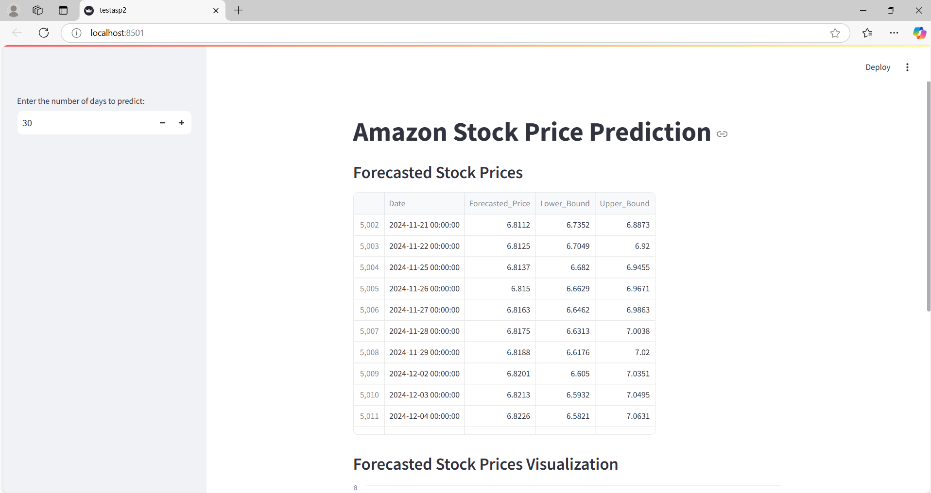


Figure 5.3 Amazon Stock Price Prediction

**6. CONCLUSION AND FUTURE WORK**

The E-Commerce Sales Analysis Project provides a comprehensive framework for analysing sales performance, customer behaviour, and predictive trends, tailored to the needs of businesses seeking to harness data for informed decision-making. By leveraging Python’s robust ecosystem, including libraries like Pandas, NumPy, and Scikit-learn for data processing and analysis, and visualization tools such as Seaborn, Plotly, and Matplotlib, the project offers insightful metrics and actionable visualizations. These tools transform raw sales data into meaningful narratives, empowering stakeholders with a clear understanding of operational performance and market trends.

**Key Features and Functionalities**

The project focuses on delivering key metrics that are pivotal for business success. Metrics such as total sales, profit margins, purchasing frequency, and customer spending patterns are computed and visualized to provide a granular understanding of sales dynamics. For instance, tracking profit margins enables businesses to identify underperforming products or regions, while purchasing frequency insights shed light on customer loyalty and engagement levels. Customer spending patterns help in segmentation, enabling personalized marketing strategies and better resource allocation.

**Predictive Analytics for Strategic Planning**

A cornerstone of this project is its predictive analytics capability, powered by ARIMA models and time series forecasting techniques. These methods analyse historical sales data to forecast future trends, empowering businesses to anticipate demand fluctuations, optimize inventory, and plan marketing campaigns effectively. The inclusion of predictive analytics goes beyond traditional reporting by identifying growth opportunities and potential risks, such as seasonal variations or declining product interest, thereby enabling proactive decision-making.

**Interactive Dashboards for Real-Time Insights**

The integration of interactive dashboards through Streamlit ensures that data insights are presented in an intuitive and user-friendly manner. These dashboards provide real-time access to metrics and visualizations, allowing stakeholders to explore data dynamically. Users can filter data by time periods, regions, or product categories, gaining tailored insights without requiring extensive technical expertise. This accessibility fosters collaboration among teams and accelerates decision-making processes.

**Data-Driven Insights and Benefits**

The project’s holistic approach supports businesses in multiple dimensions:

1. **Identifying Popular Products:** By analysing sales data, businesses can pinpoint best-selling products and focus efforts on expanding their reach, either through increased marketing or optimized inventory management.
2. **Customer Segmentation:** Detailed analysis of customer behaviour enables segmentation based on purchasing habits, preferences, and lifetime value. This segmentation allows for personalized marketing campaigns, improving customer retention and acquisition rates.
3. **Risk Management:** Insights into declining trends or high return rates help identify potential risks. By addressing these issues promptly, businesses can mitigate financial losses and improve customer satisfaction.
4. **Operational Optimization:** By understanding operational inefficiencies and regional sales discrepancies, businesses can streamline their processes and allocate resources more effectively.
5. **Scalability and Customization:**

The project is highly scalable and adaptable to varying business sizes and industries. It accommodates large datasets and diverse sales scenarios, ensuring relevance for both small enterprises and large corporations. The modular architecture allows for easy integration of additional features, such as advanced machine learning models or external APIs for real-time data updates.

In conclusion, the Sales Dashboard and Analytical Tools project provides an indispensable solution for businesses aiming to thrive in a competitive market. By combining powerful analytics, interactive visualizations, and predictive capabilities, it delivers a comprehensive platform that transforms data into actionable strategies. This empowers businesses to optimize operations, maximize revenue, and build a foundation for sustained growth.

**Future Work**

The E-Commerce Sales Analysis can be significantly enhanced with the incorporation of advanced features and methodologies, making it more robust, adaptive, and insightful for businesses seeking to optimize their operations and stay ahead in competitive markets. Below are some proposed enhancements, elaborated to reflect their impact on the system's overall functionality:

**1. Advanced Predictive Modelling**

To improve the accuracy and reliability of sales forecasts, integrating advanced machine learning models such as Prophet and Long Short-Term Memory (LSTM) networks is recommended. Prophet, designed for time-series data with strong seasonal components, handles missing data and outliers effectively, making it ideal for businesses with fluctuating sales trends. LSTMs, on the other hand, excel at capturing complex patterns in sequential data, providing precise predictions for dynamic sales environments. Additionally, ensemble methods such as stacking or boosting can be employed to combine the strengths of different models, further enhancing predictive performance and addressing seasonality with greater precision.

**2. Customer Behaviour Analysis (CBA)**

A deeper understanding of customer behaviour is essential for personalized marketing and customer retention. Clustering techniques like K-means or hierarchical clustering can segment customers into meaningful groups based on attributes like purchase frequency, average spending, and product preferences. Insights derived from these segments can guide targeted marketing campaigns. Furthermore, incorporating advanced models for Customer Lifetime Value (CLV) estimation and churn prediction helps businesses identify high-value customers and those at risk of attrition, enabling proactive engagement strategies. These insights empower businesses to tailor offerings, improve customer satisfaction, and boost long-term profitability.

**3. Real-Time Data Integration**

The integration of real-time data sources through APIs ensures that analytics remain current and relevant. By fetching live sales and customer data, businesses can make informed decisions without delay. Automating data pipelines with tools like Apache Airflow or AWS Lambda streamlines the process of data ingestion, transformation, and loading (ETL), reducing manual effort and enhancing efficiency. This real-time capability is crucial for responding promptly to market dynamics and maintaining an edge in fast-paced environments.

**4. Enhanced Visualization**

Visualization is key to delivering insights in an accessible and engaging format. Incorporating 3D visualizations and animated dashboards can make complex data more comprehensible and engaging for stakeholders. For example, 3D bar graphs or rotating heatmaps can represent multi-dimensional data effectively. Adding drill-down capabilities to interactive dashboards allows users to explore data at a granular level, such as sales by region, product category, or customer demographics. These advanced visualization techniques foster deeper data engagement and better decision-making.

**5.Sales Optimization**

Optimizing sales strategies is a critical aspect of this project. By implementing recommendation systems, businesses can identify upselling and cross-selling opportunities, enhancing revenue streams. For instance, suggesting complementary products to customers based on their purchase history can significantly boost average transaction values. Additionally, integrating A/B testing analysis enables businesses to evaluate the effectiveness of marketing campaigns, pricing strategies, and promotional offers. Insights from these tests help refine strategies, ensuring better conversion rates and customer satisfaction.

Overall Impact

These enhancements will elevate the project’s capabilities, making it a cutting-edge solution for businesses striving to maximize efficiency and profitability. Advanced predictive models and customer behavior insights ensure data-driven foresight, while real-time data integration and enhanced visualizations foster timely and actionable decision-making. By including features like sales optimization through recommendation systems and A/B testing, the project ensures alignment with evolving business needs and technological advancements, cementing its relevance in the dynamic market landscape.

**7. BIBLIOGRAPHY**

1.Python Libraries for Data Analysis and Visualization

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.

2.Sales and Customer Behaviour Analytics

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

3.Predictive Sales Analysis and Forecasting

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (1998). *Forecasting Methods and Applications*. Wiley.

4.Data Visualization Techniques

Waskom, M. (2021). *Seaborn Documentation*. Retrieved from https://seaborn.pydata.org.

Plotly Technologies Inc. (2015). *Interactive Graphing and Cross filtering*. Retrieved from <https://plotly.com>.

5.Interactive Dashboard Development

Streamlit Inc. (2021). *Streamlit Documentation*. Retrieved from <https://streamlit.io>.

6.Machine Learning and Statistical Modelling

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

7.Data Set Links

<https://github.com/SwatiR20/Python_Amazon_Sales_Analysis/blob/main/Amazon%20Sale%20Report.csv>

<https://github.com/deshpandenu/Time-Series-Forecasting-of-Amazon-Stock-Prices-using-Neural-Networks-LSTM-and-GAN-/blob/master/AMZN.csv>

<https://archive.ics.uci.edu/dataset/502/online+retail+ii>

8.https://www.researchgate.net/publication/290786852\_Sales\_Analysis\_of\_E-Commerce\_Websites\_using\_Data\_Mining\_Techniques

9.https://www.researchgate.net/publication/360397827\_Predicting\_E-commerce\_Sales\_Inventory\_Management\_using\_Machine\_Learning

10. https://ieeexplore.ieee.org/document/10206072

**8. APPENDIX**

**1.Sales Analysis**

#Importing libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import ipywidgets as widgets

from IPython.display import display

import plotly.express as px

import streamlit as st

data = pd.read\_csv("Amazon Sale Report (1).csv",low\_memory = False)

st.title("Amazon Data Analysis Dashboard")

#Data overview

data.shape

data.isnull().sum()

data.describe()

data.info()

#Data Manipulation

data = data.dropna(subset=['ship-postal-code'])

data.drop(['New','PendingS'], axis=1, inplace=True)

data = data.dropna(subset=['currency'])

data['Date'] = pd.to\_datetime(data['Date'],dayfirst = True)

data = data.set\_index('Date')

data['year'] = data.index.year

data['month'] = data.index.month

data['day'] = data.index.day

data['ship-postal-code']=data['ship-postal-code'].astype('int')

#Data Visualisation

ax=sns.countplot(x='Size' ,data=data)

ax.set\_title('Size Analysis')

for bars in ax.containers:

ax.bar\_label(bars)

plt.show()

fig, ax = plt.subplots() # `fig` for the figure, `ax` for the axes

sns.countplot(x=data['Size'], data=data, ax=ax)

ax.set\_title('Size Analysis')

for bars in ax.containers:

ax.bar\_label(bars)

# Display the plot in Streamlit

st.pyplot(fig)

selected\_category = st.sidebar.selectbox(

"Select a Category:",

options=data['Category'].unique(),

key="unique\_category\_select"

)

# Function to display sizes by category

def display\_sizes\_by\_category(category):

category\_data = data[data['Category'] == category]

category\_sizes = category\_data.groupby('Size')['Qty'].sum()

# Create the plot

plt.figure(figsize=(12, 6))

plt.plot(category\_sizes.index.astype(str), category\_sizes.values, marker='o', color='purple')

plt.title(f'Sizes for each Category: {category}')

plt.xlabel('Size')

plt.ylabel('Total Quantity')

plt.xticks(rotation=45)

plt.grid(True)

# Display the plot in Streamlit

st.pyplot(plt.gcf())

# Call the function with the selected category

display\_sizes\_by\_category(selected\_category)

ax=sns.countplot(x='Category' ,data=data)

ax.set\_title('Category Analysis')

for bars in ax.containers:

ax.bar\_label(bars)

plt.show()

fig, ax = plt.subplots() # `fig` for the figure, `ax` for the axes

sns.countplot(x=data['Category'], data=data, ax=ax)

ax.set\_title('Category Analysis')

for bars in ax.containers:

ax.bar\_label(bars)

# Display the plot in Streamlit

st.pyplot(fig)

# Function to display sizes by category

def display\_sales\_by\_category(category):

category\_data = data[data['Category'] == category]

category\_sales = category\_data.groupby('month')['Amount'].sum()

# Create the plot

plt.figure(figsize=(12, 6))

plt.plot(category\_sales.index.astype(str), category\_sales.values, marker='o', color='purple')

plt.title(f'Sales Trend for Category: {category}')

plt.xlabel('Month')

plt.ylabel('Total Amount')

plt.xticks(rotation=45)

plt.grid(True)

# Display the plot in Streamlit

st.pyplot(plt.gcf())

# Call the function with the selected category

display\_sales\_by\_category(selected\_category)

data['Courier Status'].unique()

plt.figure(figsize=(12,7))

ax = sns.countplot(data = data, x='Courier Status',hue='Status')

plt.show()

st.pyplot(plt.gcf())

data['Category'].value\_counts().plot(kind='bar')

data['B2B'].value\_counts().plot(kind='pie')

B2B\_Check = data['B2B'].value\_counts()

plt.title('Business to Business Analysis')

plt.pie(B2B\_Check, labels=B2B\_Check, autopct='%1.1f%%')

plt.axis('equal')

plt.show()

fb = data['Fulfilment'].value\_counts()

plt.title('Fulfillment Analysis')

plt.pie(fb, labels=fb.index, autopct='%1.1f%%')

plt.axis('equal')

plt.show()

fb = data['Fulfilment'].value\_counts()

# Create the pie chart

plt.figure(figsize=(8, 6)) # Set the figure size

plt.title('Fulfillment Analysis')

plt.pie(fb, labels=fb.index, autopct='%1.1f%%')

plt.axis('equal') # Ensures the pie chart is a circle

# Render the plot in Streamlit

st.pyplot(plt.gcf())

plt.figure(figsize=(12,6))

sns.countplot(data = data, x='ship-state')

plt.xlabel('States')

plt.ylabel('Count')

plt.title('Shipment of product in each state')

plt.xticks(rotation=90)

plt.show()

st.pyplot(plt.gcf())

st.sidebar.header("Filter Options")

selected\_state = st.sidebar.selectbox("Select a State:", options=data['ship-state'].unique())

#selected\_category = st.sidebar.selectbox("Select a Category:", options=data['Category'].unique())

filtered\_data = data[(data['ship-state'] == selected\_state) & (data['Category'] == selected\_category)]

st.write(f"### Shipment Quantities in {selected\_state} for {selected\_category}")

if not filtered\_data.empty:

fig = px.bar(

filtered\_data,

x=data['Size'],

y=data['Qty'],

color=data['Size'],

title=f"Shipment Quantities in {selected\_state} for {selected\_category}",

labels={'Quantity': 'Number of Shipments', 'Size': 'Product Size'}

)

st.plotly\_chart(fig, use\_container\_width=True)

else:

st.write("No data available for the selected filters.")

st.sidebar.markdown("### Additional Insights")

if st.sidebar.checkbox("Show Data Summary"):

st.write("#### Data Summary")

st.write(filtered\_data.describe())

**2.Customer Segmentation and Behavioral Analysis:**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import streamlit as st

data = pd.read\_csv('online\_retail\_II.csv')

st.title("Customer Segmentation and Behavioural Analysis Dashboard")

data.info()

data.describe()

data.isnull().sum()

data.shape

data['InvoiceDate']=pd.to\_datetime(data['InvoiceDate'],errors='coerce')

data.dropna(inplace = True)

data['Invoice'].nunique()

data['Customer ID'].nunique()

data['T\_Price']=data['Price']\*data['Quantity']

# Group data by date and count the number of invoices per date

daily\_invoices = data.groupby(data['InvoiceDate'].dt.date)['Invoice'].count()

# Plot the number of invoices over time

plt.figure(figsize=(12, 6))

plt.plot(daily\_invoices.index, daily\_invoices.values, marker='o', color='b')

plt.title("Number of Invoices Over Time")

plt.xlabel("Date")

plt.ylabel("Number of Invoices")

plt.xticks(rotation=45)

plt.show()

st.pyplot(plt.gcf())

plt.figure(figsize=(12, 6))

sns.histplot(data['InvoiceDate'], bins=40, kde=True) # Adjust bins as needed

plt.title("Distribution of Invoices by Date")

plt.xlabel("Invoice Date")

plt.ylabel("Frequency")

plt.xticks(rotation=45)

plt.show()

st.pyplot(plt.gcf())

purc\_count = data.groupby('Customer ID')['Invoice'].nunique()

# Sidebar options

options = ["Top 10", "Top 30", "Top 50", "Bottom 10"]

selection = st.sidebar.selectbox("Select a Customer Group:", options)

# Determine the number of customers to display based on selection

num\_customers = int(selection.split(" ")[1])

order = selection.split(" ")[0].lower()

# Select top or bottom customers

if order == "top":

selected\_customers = purc\_count.sort\_values(ascending=False).head(num\_customers)

else:

selected\_customers = purc\_count.sort\_values(ascending=True).head(num\_customers)

# Plot the selected customers' purchase counts

plt.figure(figsize=(14, 8))

selected\_customers.plot(kind='bar', color='skyblue')

plt.title(f"{selection} Customers with Purchase Counts", fontsize=16)

plt.xlabel("Customer ID", fontsize=12)

plt.ylabel("Number of Purchases", fontsize=12)

plt.xticks(rotation=90)

# Display the plot in Streamlit

st.pyplot(plt.gcf())

data.groupby('Description').agg({'Quantity':'sum'}).sort\_values('Quantity',ascending=False)

data.groupby('Invoice').agg({'T\_Price':'sum'})

data = data[~data['Invoice'].str.contains('C',na=False)]

print('Farthest Time: ',data['InvoiceDate'].max())

print('Nearest Time: ',data['InvoiceDate'].min())

from datetime import date

# Get today's date

today\_date = pd.to\_datetime(date.today())

# Perform RFM aggregation

rfm = data.groupby('Customer ID').agg({

'InvoiceDate': lambda InvoiceDate: (today\_date - InvoiceDate.max()).days, # Recency

'Invoice': lambda Invoice: Invoice.nunique(), # Frequency

'T\_Price': lambda T\_Price: T\_Price.sum() # Monetary

})

# Rename the columns to recency, frequency, and monetary

rfm.columns = ["recency", "frequency", "monetary"]

rfm.describe()

rfm['Rs'] = pd.qcut(rfm['recency'],5,labels=[5,4,3,2,1])

rfm['Fs'] = pd.qcut(rfm['frequency'].rank(method='first'),5,labels=[1,2,3,4,5])

rfm['Ms'] = pd.qcut(rfm['monetary'].rank(method='first'),5,labels=[1,2,3,4,5])

rfm['RF\_Score'] = (rfm['Rs'].astype(str) + rfm['Fs'].astype(str))

Max\_rf\_score = rfm['RF\_Score'].sort\_values(ascending=False).head(50)

rfm['RF\_Score'] = pd.to\_numeric(rfm['RF\_Score'], errors='coerce')

rfm['RF\_Category'] = pd.qcut(rfm['RF\_Score'], 3, labels=["Low", "Medium", "High"])

rfm\_category\_counts = rfm['RF\_Category'].value\_counts()

# Plotting the bar plot using seaborn

sns.set(style="whitegrid")

plt.figure(figsize=(8, 6))

# Create a bar plot

ax=sns.barplot(x=rfm\_category\_counts.index, y=rfm\_category\_counts.values, palette='Set3')

for bars in ax.containers:

ax.bar\_label(bars)

# Adding titles and labels

plt.title('Number of Customers in Each RF Category', fontsize=16)

plt.xlabel('RF Category', fontsize=12)

plt.ylabel('Number of Customers', fontsize=12)

# Show the plot

plt.show()

st.title("RF Category Analysis")

st.write("### Number of Customers in Each RF Category")

# Display the counts in a table

st.write(rfm\_category\_counts)

# Plotting the bar plot

sns.set(style="whitegrid")

plt.figure(figsize=(8, 6))

ax = sns.barplot(x=rfm\_category\_counts.index, y=rfm\_category\_counts.values)

for bars in ax.containers:

ax.bar\_label(bars)

plt.title('Number of Customers in Each RF Category', fontsize=16)

plt.xlabel('RF Category', fontsize=12)

plt.ylabel('Number of Customers', fontsize=12)

st.pyplot(plt)

top\_rows = rfm.nlargest(20, 'RF\_Score')

st.title("RF Score Analysis")

st.write("Top 20 rows with the maximum RF\_Score:")

st.dataframe(top\_rows)

**3.Predictive Time Series Analysis:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.tsa.arima\_model import ARIMA

from pmdarima.arima import auto\_arima

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

plt.rcParams['figure.figsize'] = [15, 5]

adata = pd.read\_csv('AMZN.csv')

adata

adata.info()

adata.describe()

adata.head()

adata['Date'] = pd.to\_datetime(adata['Date'])

adata = adata.set\_index('Date')

adata['year'] = adata.index.year

adata['month'] = adata.index.month

adata['day'] = adata.index.day

close\_mean=adata.groupby(['Date'])['Close'].mean()

close\_mean.plot(figsize=(15,5),title=('Closing Prices(Month Wise)'),fontsize=14)

close\_mean.rolling(window=100).mean().plot(figsize=(15,5), fontsize=14)

#There is a sudden rise in the stock price in 2018 according to the two plots shown here

adata['Close'].loc['2019':'2020'].plot(figsize=(15,5), fontsize=14)

adata['Close'].loc['2019':'2020'].rolling(window=100).mean().plot(figsize=(15,5), fontsize=14)

adata['Close'].loc['2020-02':'2020-09'].plot(figsize=(15,5), fontsize=14)

adata.loc['1998':'2019'].groupby('month')['Close'].mean().plot.bar()

adata=adata[['Close']]

def stationary(ts):

rm=ts.rolling(12).mean()

rd=ts.rolling(12).std()

plt.plot(ts,color='blue',label='Original')

plt.plot(rm,color='yellow',label='Rolling Mean')

plt.plot(rd,color='green',label='Rolling Std')

plt.legend(loc='best')

plt.title('Rolling Mean and Standard Deviation')

plt.show(block=False)

print('Results of dickey fuller test')

result = adfuller(ts, autolag='AIC')

print('Test statistic: ' , result[0])

print('p-value: ' ,result[1])

print('Critical Values:' ,result[4])

stationary(adata)

decomposition = seasonal\_decompose(adata, model='multiplicative', period=52)

trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

plt.subplot(411)

plt.plot(adata, 'blue', label = 'Original')

plt.legend(loc='best')

plt.subplot(412)

plt.plot(trend, 'green', label = 'Trend')

plt.legend(loc='best')

plt.subplot(413)

plt.plot(seasonal, 'red', label='Seasonality')

plt.legend(loc='best')

plt.subplot(414)

plt.plot(residual, label='Residuals')

plt.legend(loc='best')

plt.tight\_layout

adatalog = np.log(adata)

moving\_avg = adatalog.rolling(12).mean()

stationary(adatalog)

diffadata\_pre\_log = adatalog - moving\_avg

diffadata\_pre\_log.dropna(inplace = True)

stationary(diffadata\_pre\_log)

train\_data, test\_data = adatalog[3:int(len(adatalog)\*0.8)], adatalog[int(len(adatalog)\*0.8):]

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Dates')

plt.ylabel('Closing Prices')

plt.plot(adatalog, 'green', label='Train data')

plt.plot(test\_data, 'blue', label='Test data')

plt.legend()

model\_autoARIMA = auto\_arima(train\_data, start\_p=1, start\_q=1, max\_p=5, max\_q=5, m=12,

start\_P=0, seasonal=False, d=1, D=1, trace=True,

error\_action='ignore',

suppress\_warnings=True,

stepwise=True)

print(model\_autoARIMA.summary())

model\_autoARIMA.plot\_diagnostics(figsize=(15,8))

plt.show()

from statsmodels.tsa.statespace.sarimax import SARIMAX

# SARIMAX model with intercept

model = SARIMAX(train\_data, order=(0, 1, 0), trend='c')

fitted = model.fit()

print(fitted.summary())

forecast = fitted.get\_forecast(steps=1176)

forecast\_mean = forecast.predicted\_mean

confidence\_interval = forecast.conf\_int()

print(forecast\_mean)

print(confidence\_interval)

start = len(train\_data)

end = len(train\_data) + len(test\_data)

predictions = fitted.predict(start, end,

type = 'levels').rename("Predictions")

# plot predictions and actual values

predictions.plot(color='red', legend = 'True')

test\_data.plot(color = 'blue', legend='True')

fc = forecast.predicted\_mean

conf = forecast.conf\_int(alpha=0.05) # 95% confidence intervals

fc\_series = pd.Series(fc)

lower\_series = pd.Series(conf.iloc[:, 0])

upper\_series = pd.Series(conf.iloc[:, 1])

fc\_series.index = test\_data.index[:len(fc)]

lower\_series.index = test\_data.index[:len(conf)]

upper\_series.index = test\_data.index[:len(conf)]

plt.figure(figsize=(12,5), dpi=100)

plt.plot(train\_data, color = 'green', label='training')

plt.plot(test\_data, color = 'blue', label='Actual Stock Price')

plt.plot(fc\_series, color = 'red',label='Predicted Stock Price')

plt.fill\_between(lower\_series.index, lower\_series, upper\_series,

color='k', alpha=.10)

plt.title('Amazon Stock Price Prediction')

plt.xlabel('Time')

plt.ylabel('Stock Price')

plt.legend(loc='upper left', fontsize=8)

plt.xlim()

plt.show()

test\_data\_filled = test\_data.fillna(test\_data.mean())

fc\_filled = pd.Series(fc).fillna(fc.mean())

test\_data\_filled = test\_data\_filled[:len(fc\_filled)]

mse = mean\_squared\_error(test\_data\_filled, fc\_filled)

mae = mean\_absolute\_error(test\_data\_filled, fc\_filled)

print('MSE:', mse)

print('MAE:', mae)

import pickle

with open('sarimax\_model.pkl', 'wb') as file:

pickle.dump(fitted, file)

print("Model saved successfully!")

**4.Streamlit App for Prediction of Stock Prices:**

import pickle

import pandas as pd

import streamlit as st

# Function to load the trained SARIMAX model

@st.cache\_data

def load\_sarimax\_model():

with open('sarimax\_model.pkl', 'rb') as file:

model = pickle.load(file)

return model

# Main app

st.title("Amazon Stock Price Prediction")

# Load the trained model

model = load\_sarimax\_model()

# Sidebar for user input

steps = st.sidebar.number\_input(

"Enter the number of days to predict:",

min\_value=1, max\_value=365, value=30, step=1

)

# Forecasting function

def forecast\_stock\_price(steps):

# Forecasting using the SARIMAX model

forecast = model.get\_forecast(steps=steps)

forecast\_values = forecast.predicted\_mean

forecast\_conf\_int = forecast.conf\_int()

# Create a DataFrame to display the forecast and its confidence intervals

forecast\_df = pd.DataFrame({

'Date': pd.date\_range(start='2024-11-21', periods=steps, freq='B'), # Replace with actual start date

'Forecasted\_Price': forecast\_values,

'Lower\_Bound': forecast\_conf\_int.iloc[:, 0],

'Upper\_Bound': forecast\_conf\_int.iloc[:, 1]

})

return forecast\_df

# Get the forecast for the selected number of days

forecast\_df = forecast\_stock\_price(steps)

# Display forecasted values and confidence intervals

st.write("### Forecasted Stock Prices")

st.write(forecast\_df)

# Plot forecasted values

st.write("### Forecasted Stock Prices Visualization")

st.line\_chart(forecast\_df.set\_index('Date'))