ABSTRACT

To minimizes costs and optimize process, it is essential for e-commerce companies to utilize demand forecasting. This project creates a demand forecast using the machine learning method to help businesses analyze and make choices based on the future sales. The dataset consists of 676 rows and 30 features: product attributes, price, temporal regularity, customer preferences, and influences from outside the system such as Google click-through rates and Facebook advert impressions. These features are very useful to dissect out the demand pattern and the seasonality of the demand. By applying the ARIMA algorithm for intrinsic time-series where it was possible to model trends and seasonality and other independency predictors by using an improved **Dynamic Regression Model**. This graph illustrated the forecasted sale using the actual sale for the next 10 periods with region of prediction uncertainty. The performance of the model was confirmed with low error indicating the capabilities of using the model in improving inventory management and decreasing operating expense. This project demonstrates the importance of using machine learning in improvement of demand forecasting in e-commerce aiding businesses to precisely predict and counteract the change of consumer trends in supply chain.

Keywords: Demand Forecasting, Electronic commerce, ARIMA, Machine Learning, Time series analysis, Inventory management.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

This research investigates the appropriate models for demand forecasting suitable for e-tail industries with real-time data on consumer buying behaviors, thus improving supply chain management and thus customer satisfaction. Power organisations are using data analytics to forecast future consumer demand, reduce on excess inventories and inventories shortages. Since the consumers' behaviour is constantly evolving, the basic strategies for analysing the demand forces do not suffice thus the need for businesses to employ state-of-the-art methodologies such as the machine learning algorithms to analyse the demand trends.

This project is centered on the need to create a strong and specific demand forecasting solution for an e-commerce firm. The major aim is the forecasting of product demand in the future given that product characteristics, price and product features together with external factors like marketing parameters. The model seeks to ensure that businesses have relevant sales forecasts, to help make required decisions such as stocking, purchasing and delivering of products. He noted that more so companies which anticipate the change in demand pattern are usually in a better position to optimise their activities and hence the cost effectively able to meet the needs of its clients.
The dataset applied in this project includes 676 records and 30 features and contains sufficient information about certain factors affecting demand in an e-commerce environment. Variables used include product related variables such as product_id_encoded, price related attributes including total_price, unit_price

temporal variables such as weekday and holiday, customers and qty, and competitive prices such as comp_1, and comp_2. These features are important because complex demand patterns have to be captured in order to learn from previous patterns, seasonal behaviour and impact of outside promotions.

To address the challenge of accurately predicting demand, two advanced machine learning models were employed: ARIMA (Auto-Regressive Integrated Moving Average) and Dynamic Regression. It is also highly useful for time series data, as has been used in analyzing historical sales data patterns including trends, seasonality and noise. At the same time, the Dynamic Regression Model refines the forecast with the help of outer variables, including Google click through rate, and Facebook impact on consumer behavior. The consolidation of these models would enable the project to facilitate the provision of accurate, dependable, and utilitarian demand estimates for e-commerce ventures.

To this end, different metrics and graphics of the forecasting model were applied and used in this project to ensure that the forecasts done were not only correct but also consistent. The results prove the efficiency of the model and its capability to define basic trends and patterns connected to demand and sales, as well as generate forecasts used by companies to predict future sales. By adopting such approach, e-commerce firms are able to maintain or even improve their stock in relation to costs while aligning their stocks with customer’s demand due to the growing competition.

In particular, this project reflects the increasing prominence within the e-commerce market of big data analysis and plans for managing demand based on machine learning. The application of such models will assist businesses in realizing better and faster predictions, greater efficiency, and increased competitiveness of the organization in the existing conditions of the market.

1.2 PROBLEM DEFINITION

The forecasting of future demand is a major problem in the cardboard manufacturing industry because of the instability of the market situations, climatic changes, and other circumstances. Some of the mismanagement risks caused by poor demand forecasting are; overstocking, outright stock out, high stocking expense, and missed sales. All these issues affect not only the logistics or performance but also the economic profitability of the firm. For this reason, the companies experiences a lot of difficulties in achieving the optimal volume of product stock, satisfying customers' needs when they need it, and excluding extra expenses on storage and immediate replenishment. In order to overcome these challenges, an integrated demand forecasting system must be established that need to involve the use of techniques such as analysis of big data. Combining historical sales data, economic variables, and other parameters the goal of creating a prognosis model for future demand to increase the effectiveness of inventories control and production .

CHAPTER 2

LITERATURE SURVEY

The coupling of incentive design and differential privacy is suggested to have a great potential in boosting the capability of demand forecasting in e-commerce. Given in the model incentives aspects can help to force data owners and interested parties to contribute key data that enhance sales forecast precision. In parallel, asserting differential privacy could protect the contribution of individual data and enhance the security of the model, therefore suitable for sensitive data settings. This approach suits the user requirements of e-commerce turnover applications since reliable and ethical sales forecasting necessitates high-quality and privacy preserving data[7].

An approach known as H-FL together with ConvLSTM to forecast the demand of the e-commerce products since such systems are vulnerable to the bullwhip effect and privacy issues. In turn, it promotes the sharing of data, which usually helps improve the accuracy of a forecast without compromising privacy. Compared to the basic models such as LSTM and BiLSTM, the proposed model minimizes the loss of efficiency in the supply chain while considering the development of sustainable e-commerce. This work presents a practical solution for improving demand forecasting based on federated learning in large scale and insecure environments, consequently establishing the theme for future studies of federated learning.[15]

The paper discusses the use of Federated Learning (FL) as a promising solution to address privacy and scalability issues associated with traditional centralized machine learning models. FL allows IoT devices to collaboratively train a global model without sharing raw data, thus enhancing data privacy and reducing computation costs. The authors propose a novel incentive mechanism that integrates Mechanism Design (MD) and Differential Privacy (DP) to encourage active participation from data owners while ensuring privacy. By combining these techniques, the paper aims to optimize the performance of FL systems in dynamic environments, striking a balance between privacy preservation and system efficiency. Experimental results indicate that the proposed method achieves superior device participation, system throughput, and learning performance compared to existing FL protocols[9].

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Presently, e-commerce firms usually carry out their demand forecasting in a basic way through the use of he sales history and simple mathematical formulae. Regular techniques like moving averages or a seasonal pattern are often employed in sales projection. These methods often do not consider things like advertisements, changes in the occasion, or competitor's action thereby making predictions less accurate. Further, most of the current systems do not support real time forecasting of market conditions and hence the business cannot easily change its strategies to meet changing market conditions. As useful as such conventional tools may be in establishing some of the simplest trends, they are inadequate in capturing more intricate trends in sales data as observed in e-commerce applications.

3.2 PROPOSED SYSTEM

The proposed system is intended for effective preliminary processing of data for the use of these products in machine learning and statistical analysis. This shall offer structured methods of dealing with raw data whereby it will be able to convert them to proper use formats. This system makes it possible to provide accuracy, scalability to data processing, as well as flexibility to cater for different requirements of data processing.

The highlighted features of the proposed system are as follows:

I. Data import and loading sub module

Supports the importation of data of different types such as CSV and Excel.

Allows smooth data flow with a lot of integration done automatically.

II. In the Data Cleaning and Preprocessing Module the following steps are included;

• Missing Value Management:

Imputes missing values using mean, median or modes or opt for the deletion method of missing values.

• Outlier Detection and Handling:

This is done by trying to identify any that is unusual using statistical methods like Z-score, or Interquartile range or a graphical method.

• Data Type Conversion:

Helps to maintain the data type disparity of numerical, categorical and time-series data constant.

Feature Engineering Module

III. Encoding Categorical Variables:

Some of the operations performed are related to modifying categorical variables using label encoding, or one-hot encoding.

• Scaling and Normalization:

Normalises data to make them more suitable for machine learning model algorithms.

• Feature Selection:

Automates identification and choice of the best features to use in modeling.

IV. Data Visualization Module

Describes the nature of data in terms of trends and distributions by generating histograms, scatter plots and HeatMap.

Enables interactive dashboards to be more effective with the decision-making process.

V. Data Splitting Module

Randomly partitions data into sets used to develop, fine-tune, and evaluate the model.

VI. Machine Learning Integration with Pipe Line

If I use and external SQL engine, it allows the users to pass easily cleaned and processed production data to the holding zone for analytics with machine learning or statistical models.

3.3 FEASIBILITY STUDY:

1. Technical Feasibility

Technology Stack:

The system leverages widely-used, robust technologies such as Python and its libraries (e.g., pandas, numpy, matplotlib, scikit-learn), which are highly capable of handling data preprocessing and analysis tasks.

System Requirements:

Hardware: Moderate processing power with sufficient memory (at least 8 GB RAM recommended).

Software: Python 3.x environment with required libraries installed.

Scalability: The system can handle large datasets with efficient memory management techniques.

Skills Required:

Development requires familiarity with Python, data analysis concepts, and machine learning basics. The skillset is common among data analysts and engineers.

Conclusion: The project is technically feasible with existing tools and infrastructure.

2. Operational Feasibility

End-User Impact:

The system is designed to streamline data preprocessing, reducing manual effort and errors. It will enhance productivity for data analysts, engineers, and machine learning practitioners.

Ease of Use:

With clear documentation and modular components, the system can be adopted quickly. Predefined workflows and templates will simplify tasks for non-technical users.

Implementation Time:

The development can be completed within a short time frame (4–6 weeks) for an initial prototype.

Conclusion: The system aligns well with operational needs and is user-friendly.

3. **Financial Feasibility**

Development Costs:

Minimal costs are involved as open-source tools and libraries will be used. Costs will primarily include developer salaries and compute resources.

Operational Costs:

No licensing fees since the system relies on free tools.

Maintenance and upgrades will require minimal ongoing investment.

Return on Investment (ROI):

By automating repetitive tasks and reducing errors, the system will save time and resources, justifying the investment.

Conclusion: The project is financially feasible with high ROI potential.

4. Legal and Ethical Feasibility

Data Privacy:

The system must adhere to relevant data protection regulations (e.g., GDPR, HIPAA). Sensitive data must be anonymized, and access controls should be implemented.

Ethical Concerns:

The system should avoid bias in data handling and ensure transparency in processing.

Conclusion: Compliance with legal and ethical standards is manageable with proper guidelines.

Overall Feasibility

The project is feasible across technical, operational, financial, and ethical dimensions. Its implementation will lead to significant efficiency gains in data preprocessing and analysis workflows.

CHAPTER 4

SYSTEM IMPLEMENTATION

GOOGLE COLAB:

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset=pd.read_csv('Data.csv')

missing_values = dataset.isnull()

print(missing_values)

missing_count=dataset.isnull().sum()

print(missing_count)

missing_count[missing_count > 0]

X=dataset.iloc[: , :-1].values

print(X)

Y=dataset.iloc[: , -1].values

print(Y)

unique_values = dataset['product_id'].unique()

print(unique_values)

value_counts = dataset['product_id'].value_counts()

print(value_counts)

from sklearn.preprocessing import LabelEncoder
```

```
  # Initialize the LabelEncoder

  label_encoder = LabelEncoder()

# Fit and transform the 'product_id' column

  dataset['product_id_encoded'] = label_encoder.fit_transform(dataset['product_id'])

# Display the first few rows to see the result

  dataset[ 'product_id_encoded'].head()

from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder

  label_encoder = LabelEncoder()

# Fit and transform the 'product_id' column

  dataset['product_category_name_encoded'] =
  label_encoder.fit_transform(dataset['product_category_name'])

# Display the first few rows to see the result

  dataset[ 'product_category_name_encoded'].head()

del dataset['product_id']  # Deletes column

  print(dataset)

  del dataset['product_category_name']  # Deletes column

  print(dataset)

  # Get the list of all columns

  columns = list(dataset.columns)
  # Specify the encoded columns to be moved to the front
  columns_to_move = ['product_id_encoded', 'product_category_name_encoded']
# Reorder the columns: put the encoded columns first, then the rest of the columns
  new_column_order = columns_to_move + [col for col in columns if col not in columns_to_move]
  # Apply the new column order to the DataFrame
  dataset1 = dataset[new_column_order]
  #dataset1 = dataset.drop(['product_id', 'product_category_name'], axis=1)
  # Display the DataFrame with the new column order
  dataset1.head()
  # Convert 'month_year' to datetime, adjusting the format to include day, month, and year
  dataset1['month_year'] = pd.to_datetime(dataset1['month_year'], format='%d-%m-%Y')
  # Extract year and month as separate features (if needed)
  dataset1['year'] = dataset1['month_year'].dt.year
  dataset1['month'] = dataset1['month_year'].dt.month
  #del dataset1['month_year']
  # Display the result to confirm
  dataset1.head()
```

```
dataset1['weekday'] = dataset1['month_year'].dt.weekday  # 0=Monday,
6=Sunday

# Check the updated DataFrame

dataset1.columns

del dataset1['month_year']

# Display the result to confirm

dataset1.head()

# Apply first-order differencing

dataset1['total_price_diff'] = dataset1['total_price'].diff().dropna()

# Export the dataset to CSV

dataset1.to_csv('dataset_full_view.csv', index=False)

# Download the CSV file to view locally

from google.colab import files

files.download('dataset_full_view.csv')

from statsmodels.tsa.stattools import adfuller

result = adfuller(dataset1['total_price_diff'].dropna())

# Extracting the test statistic and p-value

adf_statistic = result[0]

p_value = result[1]

print(f'ADF Statistic: {adf_statistic}')

print(f'p-value: {p_value}')
```

```
if p_value < 0.05:
    print("Reject H0: The series is stationary.")
else:
    print("Fail to reject H0: The series is non-stationary.")
from statsmodels.tsa.stattools import kpss
kpss_statistic, p_value, lags, critical_values =
kpss(dataset1['total_price_diff'].dropna(), regression='c')
print(f'KPSS Statistic: {kpss_statistic}')
print(f'p-value: {p_value}')
print('Critical Values:')
for key, value in critical_values.items():
    print(f'  {key}: {value}')
if p_value < 0.05: print("Reject H0: The series is non-stationary.")
else: print("Fail to reject H0: The series is stationary.")
import pandas as pd
correlation = dataset1.corr()  # Compute the correlation matrix
target_correlation = correlation['qty'].sort_values(ascending=False)
print(target_correlation)
import seaborn as sns
plt.figure(figsize=(15, 12))
sns.heatmap(dataset1.corr(), annot=True, cmap='coolwarm', center=0, fmt=".2f",
  annot_kws={"size": 8})

  plt.title('Correlation Matrix', fontsize=18)

  plt.xticks(fontsize=10)

  plt.yticks(fontsize=10)

  plt.show()

  # Select only numerical columns

  numerical_data = dataset1.select_dtypes(include=['float64', 'int64'])

  # Calculate skewness for numerical columns

  skewness = numerical_data.skew()

  # Display skewness values

  print(skewness)

  # Set a threshold for skewness

  threshold = 1

  # Identify highly skewed columns

  highly_skewed_cols = skewness[abs(skewness) > threshold]

  print("Highly skewed columns:")

  print(highly_skewed_cols)

 import matplotlib.pyplot as plt

  import seaborn as sns

# Plot histograms for highly skewed columns

  for col in highly_skewed_cols.index:
    plt.figure(figsize=(10, 4))

    sns.histplot(numerical_data[col], bins=30, kde=True)

    plt.title(f'Distribution of {col}')

    plt.show()

skewness=dataset1.skew()

print(skewness)

import matplotlib.pyplot as plt

import seaborn as sns

import scipy.stats as stats

# Plot histogram and KDE for qty

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

plt.subplot(1, 2, 1)

sns.histplot(dataset1['qty'], kde=True)

plt.title('Distribution of qty')

# Q-Q plot

plt.subplot(1, 2, 2)

stats.probplot(dataset1['qty'], dist="norm", plot=plt)

plt.title('Q-Q Plot for qty')

plt.tight_layout()

plt.show()

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

from statsmodels.tsa.stattools import adfuller

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# ADF test

result = adfuller(dataset1['qty'])

print('ADF Statistic:', result[0])

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

# Interpretation: If p-value > 0.05, the data is non-stationary.

# Plot ACF and PACF

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

plt.subplot(1,2,1)

plot_acf(dataset1['total_price_diff'].dropna(), lags=20, ax=plt.gca())

plt.subplot(1,2,2)

plot_pacf(dataset1['total_price_diff'].dropna(), lags=20, ax=plt.gca())

plt.show()

# Forecasting the next 10 periods

forecast_results = arima_result.get_forecast(steps=10)


# Extract forecasted values
```

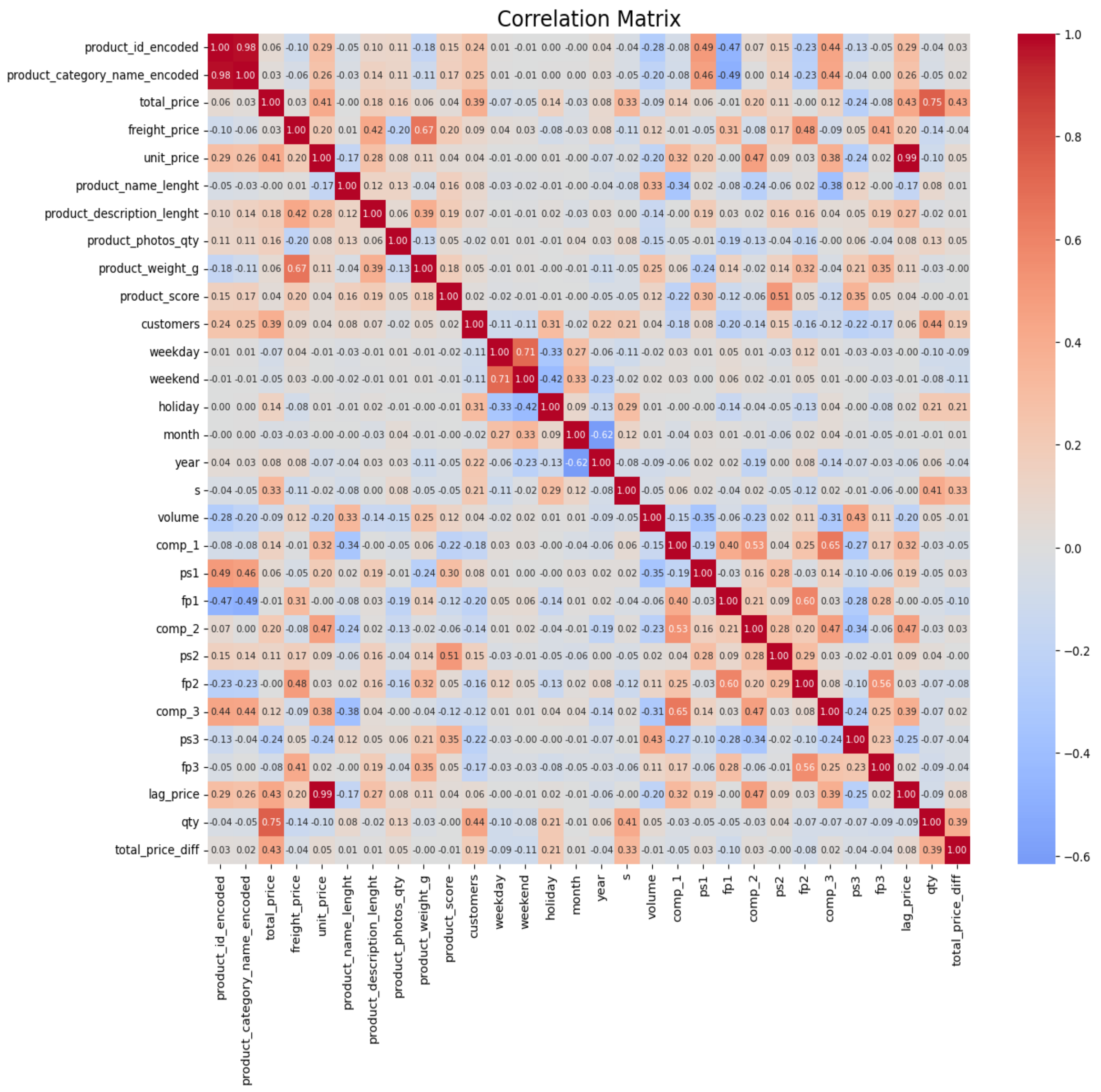
```
forecast = forecast_results.predicted_mean

# Extract confidence intervals

conf_int = forecast_results.conf_int()

# Display forecasted values with confidence intervals

print("Forecasted values:", forecast)

print("Confidence intervals:", conf_int)

# Plot the results

import matplotlib.pyplot as plt

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

plt.plot(dataset1['qty'], label='Observed')

plt.plot(arima_result.predict(start=0, end=len(dataset1)-1), label='Fitted',
color='orange')

plt.plot(range(len(dataset1), len(dataset1)+10), forecast, label='Forecast',
color='green')

plt.fill_between(range(len(dataset1), len(dataset1)+10), conf_int.iloc[:, 0],
conf_int.iloc[:, 1], color='green', alpha=0.3)

plt.xlabel('Time')

plt.ylabel('Quantity')

plt.legend()

plt.show()
```

CHAPTER 5

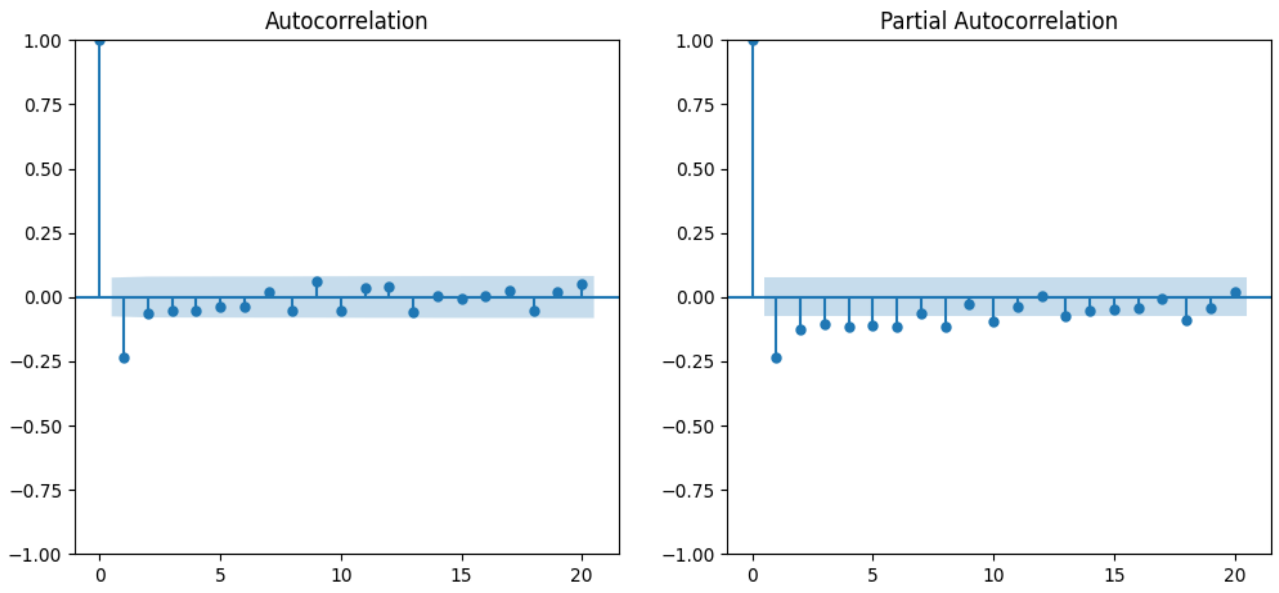
OUTPUT

Here the main visuals and outputs are displayed

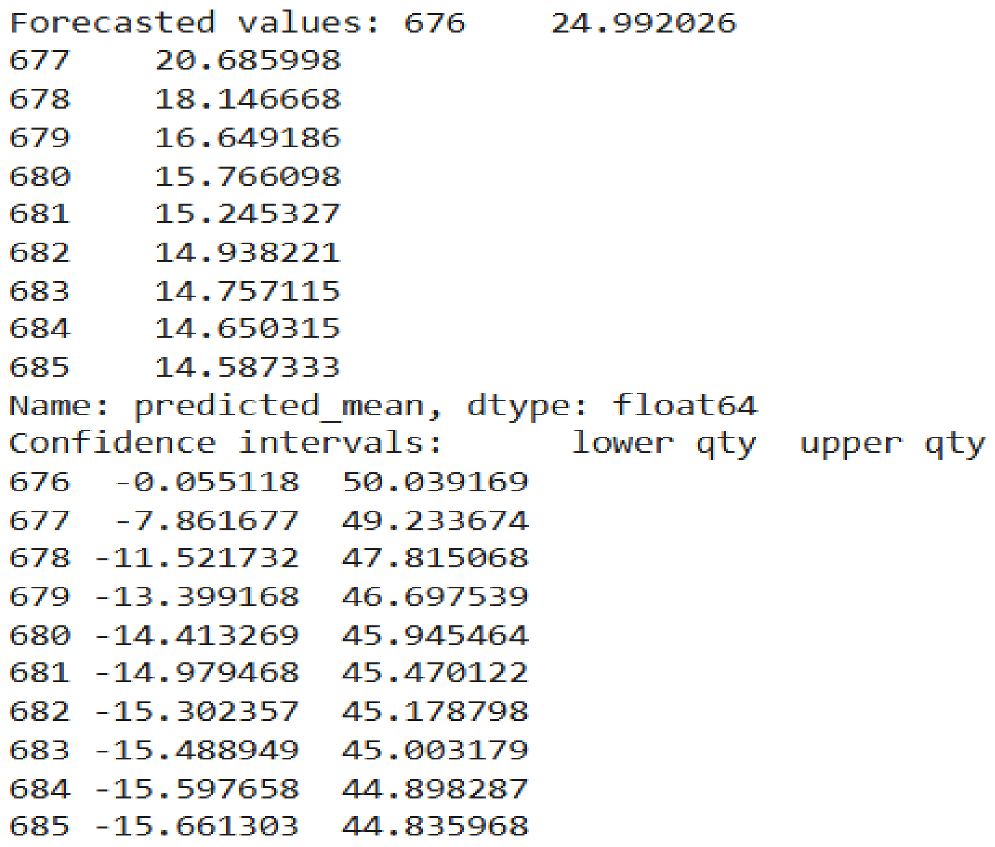
5.1 HEAT MAP



5.2 ACF AND PACF

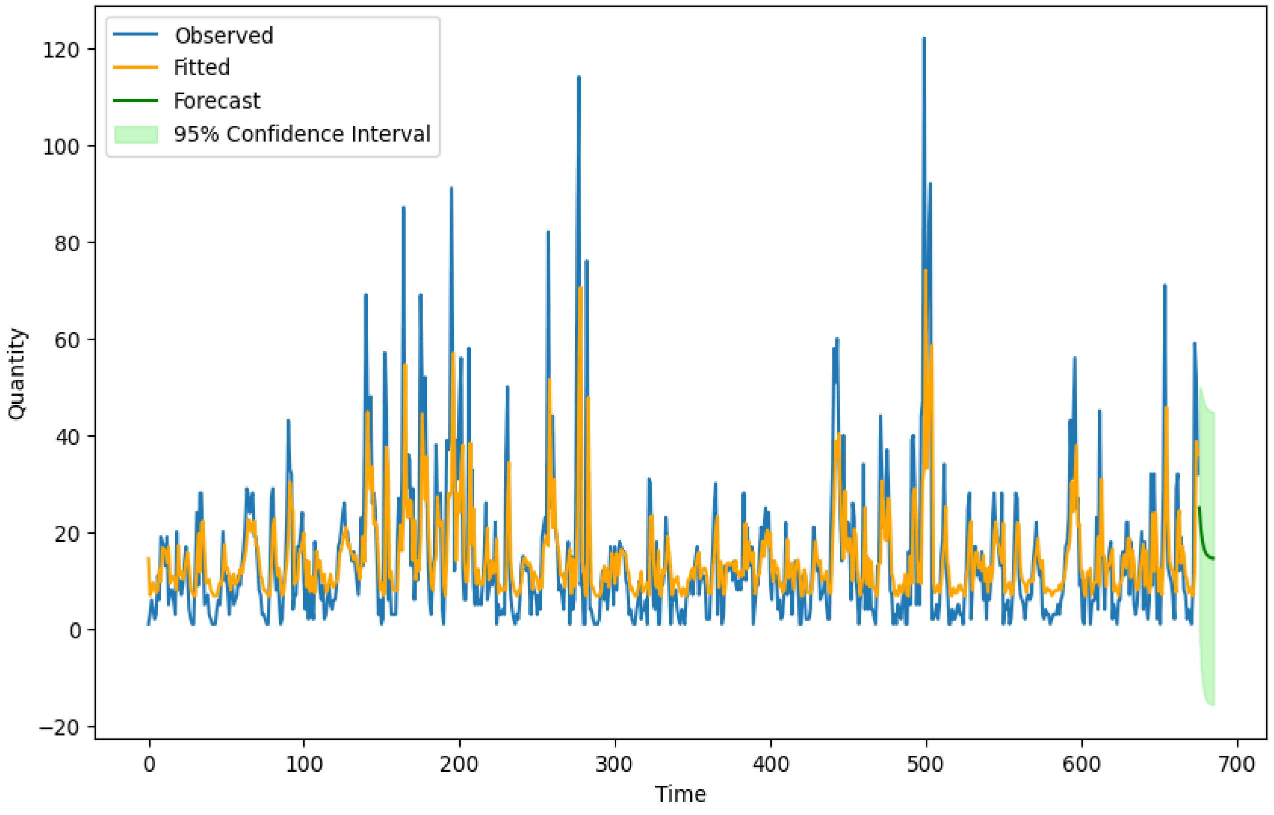


5.3 FORECASTED VALUES:



```
Forecasted values: 676    24.992026
677     20.685998
678     18.146668
679     16.649186
680     15.766098
681     15.245327
682     14.938221
683     14.757115
684     14.650315
685     14.587333
Name: predicted_mean, dtype: float64
Confidence intervals:       lower qty   upper qty
676   -0.055118  50.039169
677   -7.861677  49.233674
678 -11.521732  47.815068
679 -13.399168  46.697539
680 -14.413269  45.945464
681 -14.979468  45.470122
682 -15.302357  45.178798
683 -15.488949  45.003179
684 -15.597658  44.898287
685 -15.661303  44.835968
```

5.4 GRAPH:


Observed
Fitted
Forecast
95% Confidence Interval
Quantity
Time
120
100
80
60
40
20
0
−20
0
100
200
300
400
500
600
700

CHAPTER 6

CONCLUSION

The Findings identified that using ARIMA and Dynamic Regression, this project was able to create a reasonably good demand forecasting model for e-commerce. The model used a data set which contained 676 records and 30 variables such as product, price, customer behavior and marketing outside the firm. Whereas ARIMA model was used to describe time series data, Dynamic Regression included external variables, enhancing the forecast’s precision.

As supported by confidence intervals and error measurements, the model provided accurate predictions of the future demand. This makes it possible for e-commerce firms to adopt a efficient stock control system, low cost, and high organizational performance. Altogether, the project Illustrates how machine learning can be useful in helping organisations in the e-commerce sector to make right decisions and utilise their resources well.