National College of Ireland

Increasing Supply Chain Effectiveness: Forecasting Models for Order Quantity Prediction

MSc Research Project Answers
Artificial Intelligence for Business

Cem Akilli
Student ID: x23147016

School of Computing
National College of Ireland

Supervisor: Faithful Onwuegbuche

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Cem Akilli
Student ID:	x23147016
Programme:	Artificial intelligence for Business
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Faithful Onwuegbucher
Submission Due Date:	16/09/2024
Project Title:	Increasing Supply Chain Effectiveness:Forecasting Models for Order Quantity Prediction
Word Count:	1859
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Cem Akilli
Date:	14th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Cem AKilli
x23147016

1 Introduction

This study aims to give a thorough overview of the procedures needed to carry out the forecasting models project for supply chain management order quantity prediction. The main objective of this research was to evaluate the effectiveness of both traditional and modern time series forecasting models, specifically ARIMA and LSTM networks, in improving demand forecasting accuracy. By utilizing historical sales data, this study aims to assess the sensitivity, specificity, and accuracy of these models to provide actionable insights for optimizing inventory levels and operational efficiency. The following sections of this handbook detail the tools, methodologies, and strategies employed to achieve the defined objectives.

2 System Specification

To effectively implement the forecasting models for order quantity prediction, the following system specifications are recommended:

- **Operating System**: Windows 10 Home or later, macOS, or Linux
- **System Type**: 64-bit
- **Installed Memory (RAM)**: Minimum 8 GB (16 GB recommended)
- **Hard Drive**: Minimum 256 GB SSD
- **Processor**: Intel® Core™ i5 or equivalent (Intel® Core™ i7 or equivalent recommended)
- **GPU**: Integrated GPU is sufficient; however, a dedicated GPU.

For users without access to high-end hardware, utilizing cloud-based platforms like Google Colab is suggested.

3 Tools and Technologies

The Python programming language was utilized to complete this project, with Jupyter Notebook serving as the primary coding platform for developing and processing our code. Additionally, Google Colaboratory (Colab) was used as an alternative platform to leverage its free access to GPU resources for faster computations and model training.

- **Python Version**: Python 3.8 or later
- **Primary Development Environment**: Jupyter Notebook
- **Alternative Development Environment**: Google Colaboratory
- **Benefits**: Free cloud platform with access to GPU/TPU for enhanced performance

4 Environmental Setup

As previously said, Google Colab had been used to develop this research, and no local machine environmental setup was necessary. The coding files related to this study can be configured and executed by following these instructions. Options 1 and 2 are interchangeable, and either one ought to be taken. For faster processing, the runtime type should be changed to GPU after the session is initialized.[1]

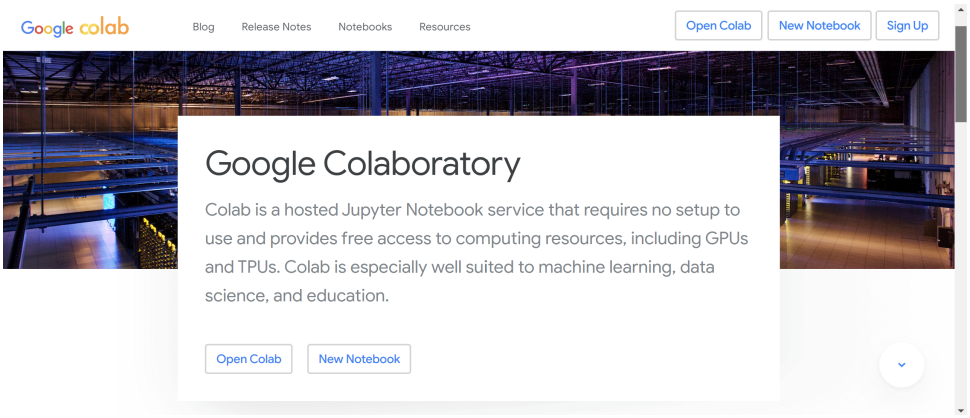


Figure 1: Option 1: Open Google Colab via Google

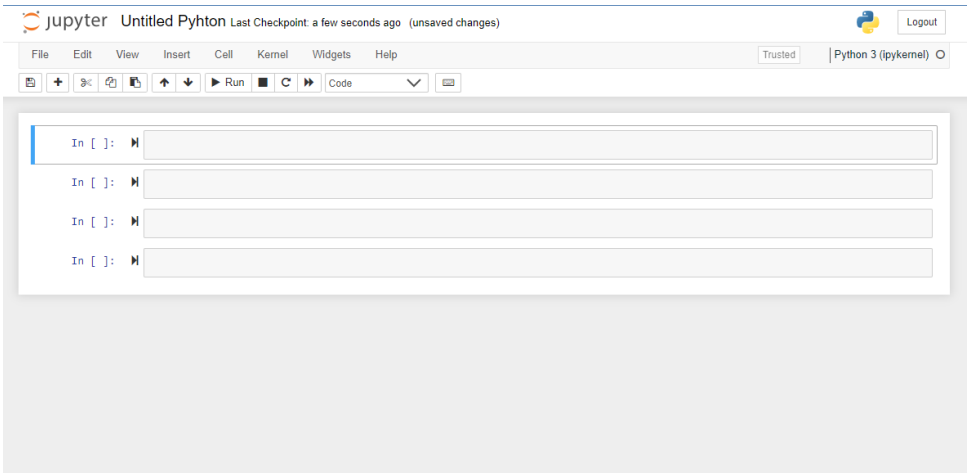


Figure 2: option 2: Open Jupyter Notebook via Hardware

[1] https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index#recent=true

5 Data Selection and Collection

The dataset used in this study was obtained from the publicly accessible, open-source Kaggle dataset platform. The data's source URL is provided below.[2]

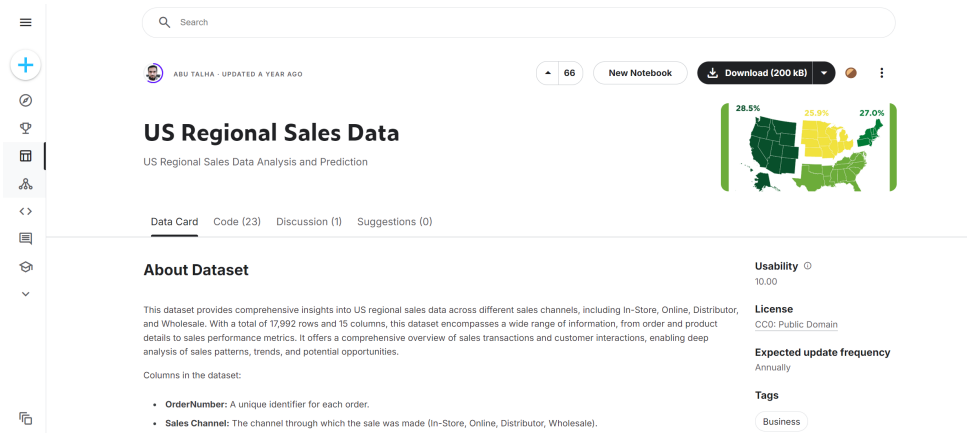
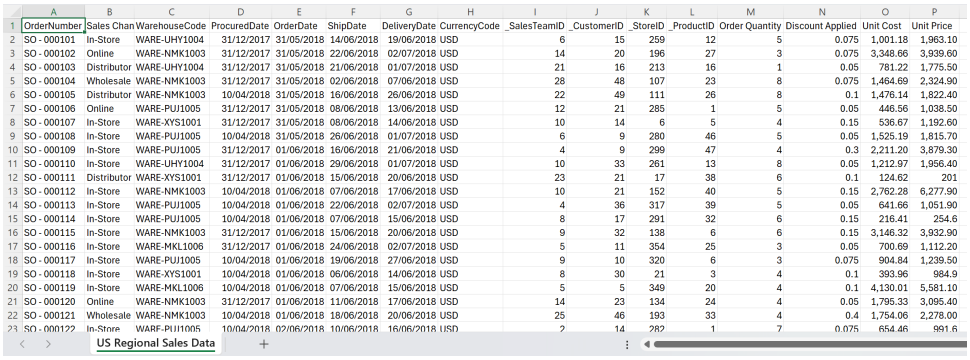


Figure 3: Dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	OrderNumber	Sales Chan	WarehouseCode	ProcuredDate	OrderDate	ShipDate	DeliveryDate	CurrencyCode	_SalesTeamID	_CustomerID	_StoreID	_ProductID	Order Quantity	Discount Applied	Unit Cost	Unit Price
2	SO - 000101	In-Store	WARE-UHY1004	31/12/2017	31/05/2018	14/06/2018	19/06/2018	USD	6	15	259	12	5	0.075	1,001.18	1,963.10
3	SO - 000102	Online	WARE-NMK1003	31/12/2017	31/05/2018	22/06/2018	02/07/2018	USD	14	20	196	27	3	0.075	3,348.66	3,939.60
4	SO - 000103	Distributor	WARE-UHY1004	31/12/2017	31/05/2018	21/06/2018	01/07/2018	USD	21	16	213	16	1	0.05	781.22	1,775.50
5	SO - 000104	Wholesale	WARE-NMK1003	31/12/2017	31/05/2018	02/06/2018	07/06/2018	USD	28	48	107	23	8	0.075	1,464.69	2,324.90
6	SO - 000105	Distributor	WARE-NMK1003	10/04/2018	31/05/2018	16/06/2018	26/06/2018	USD	22	49	111	26	8	0.1	1,476.14	1,822.40
7	SO - 000106	Online	WARE-PUJ1005	31/12/2017	31/05/2018	08/06/2018	13/06/2018	USD	12	21	285	1	5	0.05	446.56	1,038.50
8	SO - 000107	In-Store	WARE-XYS1001	31/12/2017	31/05/2018	08/06/2018	14/06/2018	USD	10	14	6	5	4	0.15	536.67	1,192.60
9	SO - 000108	In-Store	WARE-PUJ1005	10/04/2018	31/05/2018	26/06/2018	01/07/2018	USD	6	9	280	46	5	0.05	1,525.19	1,815.70
10	SO - 000109	In-Store	WARE-PUJ1005	31/12/2017	01/06/2018	16/06/2018	21/06/2018	USD	4	9	299	47	4	0.3	2,211.20	3,879.30
11	SO - 000110	In-Store	WARE-UHY1004	31/12/2017	01/06/2018	29/06/2018	01/07/2018	USD	10	33	261	13	8	0.05	1,212.97	1,956.40
12	SO - 000111	Distributor	WARE-XYS1001	31/12/2017	01/06/2018	15/06/2018	20/06/2018	USD	23	21	17	38	6	0.1	124.62	201
13	SO - 000112	In-Store	WARE-NMK1003	10/04/2018	01/06/2018	07/06/2018	17/06/2018	USD	10	21	152	40	5	0.15	2,762.28	6,277.90
14	SO - 000113	In-Store	WARE-PUJ1005	10/04/2018	01/06/2018	22/06/2018	02/07/2018	USD	4	36	317	39	5	0.05	641.66	1,051.90
15	SO - 000114	In-Store	WARE-PUJ1005	10/04/2018	01/06/2018	07/06/2018	15/06/2018	USD	8	17	291	32	6	0.15	216.41	254.6
16	SO - 000115	In-Store	WARE-NMK1003	31/12/2017	01/06/2018	15/06/2018	20/06/2018	USD	9	32	138	6	6	0.15	3,146.32	3,932.90
17	SO - 000116	In-Store	WARE-MKL1006	31/12/2017	01/06/2018	24/06/2018	02/07/2018	USD	5	11	354	25	3	0.05	700.69	1,112.20
18	SO - 000117	In-Store	WARE-PUJ1005	10/04/2018	01/06/2018	19/06/2018	27/06/2018	USD	9	10	320	6	3	0.075	904.84	1,239.50
19	SO - 000118	In-Store	WARE-XYS1001	10/04/2018	01/06/2018	06/06/2018	14/06/2018	USD	8	30	21	3	4	0.1	393.96	984.9
20	SO - 000119	In-Store	WARE-MKL1006	10/04/2018	01/06/2018	07/06/2018	15/06/2018	USD	5	5	349	20	4	0.1	4,130.01	5,581.10
21	SO - 000120	Online	WARE-NMK1003	31/12/2017	01/06/2018	11/06/2018	17/06/2018	USD	14	23	134	24	4	0.05	1,795.33	3,095.40
22	SO - 000121	Wholesale	WARE-NMK1003	10/04/2018	01/06/2018	18/06/2018	20/06/2018	USD	25	46	193	33	4	0.4	1,754.06	2,278.00
23	SO - 000122	In-Store	WARE-PUJ1005	10/04/2018	02/06/2018	10/06/2018	16/06/2018	USD	2	14	282	1	7	0.075	654.46	991.6

US Regional Sales Data

Figure 4: Overview of the Dataset

This dataset, which has 17,992 rows and 15 columns overall, contains a variety of data, including order and product details as well as sales performance measures.

6 Implementation

6.1 Loading Dataset

To begin with the implementation, we first need to load the dataset into our environment. As shown in the Figure 5, Google Colab has been used for this purpose.

First, the pandas library is imported to handle data manipulation and analysis. Then, the dataset, named "US Regional Sales Data.csv," is uploaded to the Colab environment.

[2] `https://www.kaggle.com/datasets/talhabu/us-regional-sales-data`

The file path is specified, and the dataset is read into a pandas DataFrame using the appropriate pandas function.

Figure 5: Overview of the Dataset

6.2 EDA

The initial step in Exploratory Data Analysis (EDA) involves cleaning the dataset. Unnecessary columns such as 'WarehouseCode', 'ProcuredDate', 'OrderDate', 'ShipDate', 'Unit Cost', and 'CurrencyCode' are dropped as they are not relevant for our analysis. The 'Unit Price' column is corrected by removing commas and converting it to a float data type. A new column 'TotalSales' is created by multiplying 'Order Quantity' and 'Unit Price'. Finally, the first few rows of the cleaned dataset are displayed to confirm the changes. This step ensures that the dataset is properly formatted and ready for further analysis.

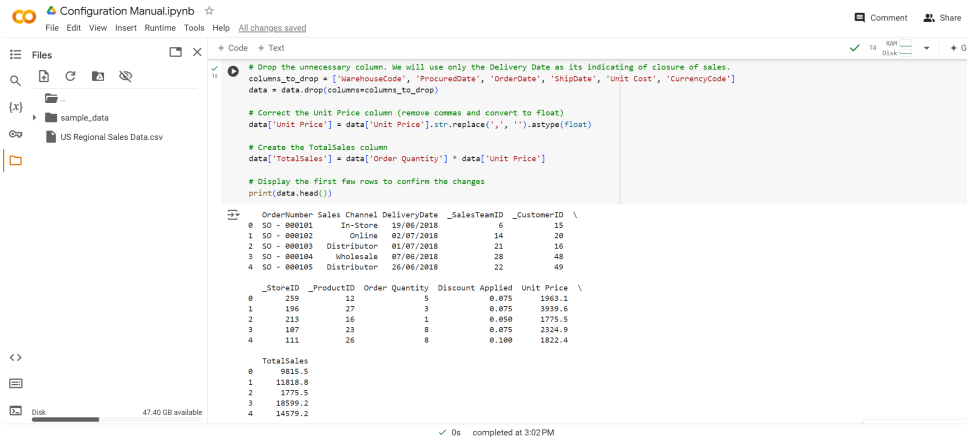


Figure 6: Initial Data Cleaning and Preparation

We examine the dataset for any missing values in order to guarantee data quality. The dataset is full and prepared for additional analysis, as the findings show that there are no missing values in any of the columns. This stage is critical because it assists in locating and addressing any data gaps that can compromise the forecasting models' accuracy.

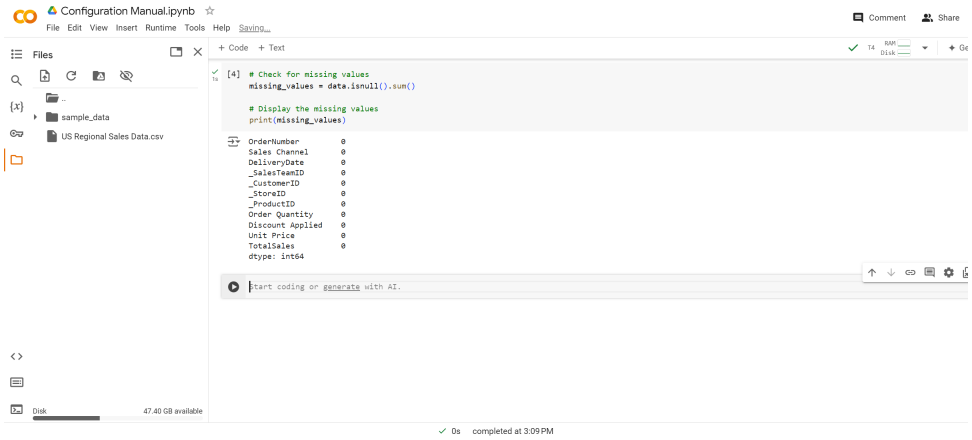


Figure 7: Checking for Missing Values

6.3 ARIMA

We still need to make some adjustments before the model. The total sales for each product are computed once the data is sorted by "Product ID". Next, the products are arranged in ascending order based on their total sales to determine which five have the lowest total sales. To facilitate appropriate time series analysis, the 'DeliveryDate' column in the dataset is changed to datetime format once it has been filtered to only include these five products.

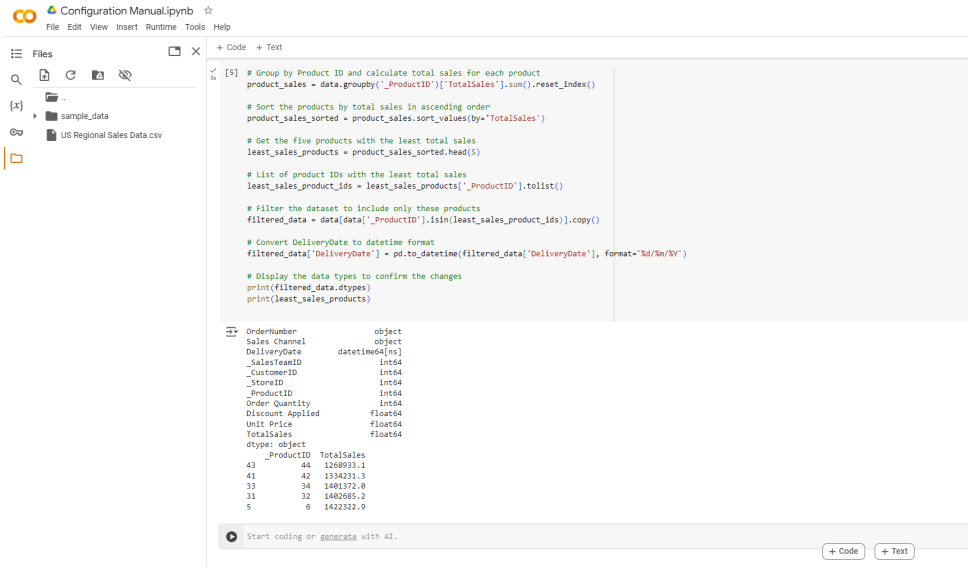


Figure 8: Preparing Dataset for ARIMA

Afterwards, the data undergoes resampling at a daily interval to guarantee temporal series consistency. Order quantity zero is used to fill in missing days in order to keep the timeline continuous. Re-sampling and filling are essential steps in time series modeling accuracy.

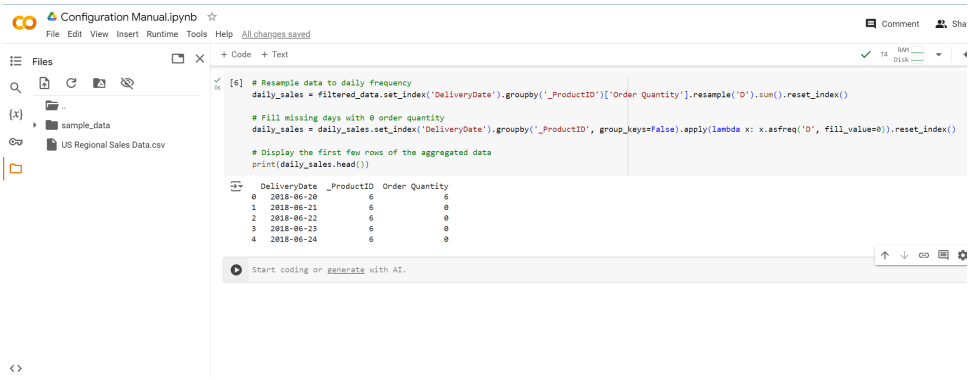


Figure 9: Resampling Data to Daily Frequency

6.3.1 Run the ARIMA

To apply the ARIMA model, the following steps are executed:

Filter Data: The dataset is filtered for one of the least sold products, such as Product 44. The data is split into training and test sets, with the last 30 days excluded from the training set.

Train the Model: The ARIMA model is trained using the training data. A rolling forecast is performed where the model is repeatedly updated with new observations to predict the next value.

Make Predictions: Predictions for the last 30 days are generated and compared against the actual order quantities. The results are stored in a comparison table, which includes the real and predicted order quantities, as well as the difference between them.

Calculate MSE: The Mean Squared Error (MSE) for the predictions over the last 30 days is calculated to assess the model's performance.

Visualize Results: The training data, actual order quantities, and predictions for the last 6 months are plotted to visually evaluate the model's performance.

These steps demonstrate the application of the ARIMA model to forecast order quantities, focusing on accurate predictions and performance evaluation as shown in Figure 10.

Product ID	MSE
32	4.378866337300955
44	3.301619571778478
42	6.709885578182906
34	6.541816452367126
6	0.2991129174585303

Table 1: MSE Results for ARIMA

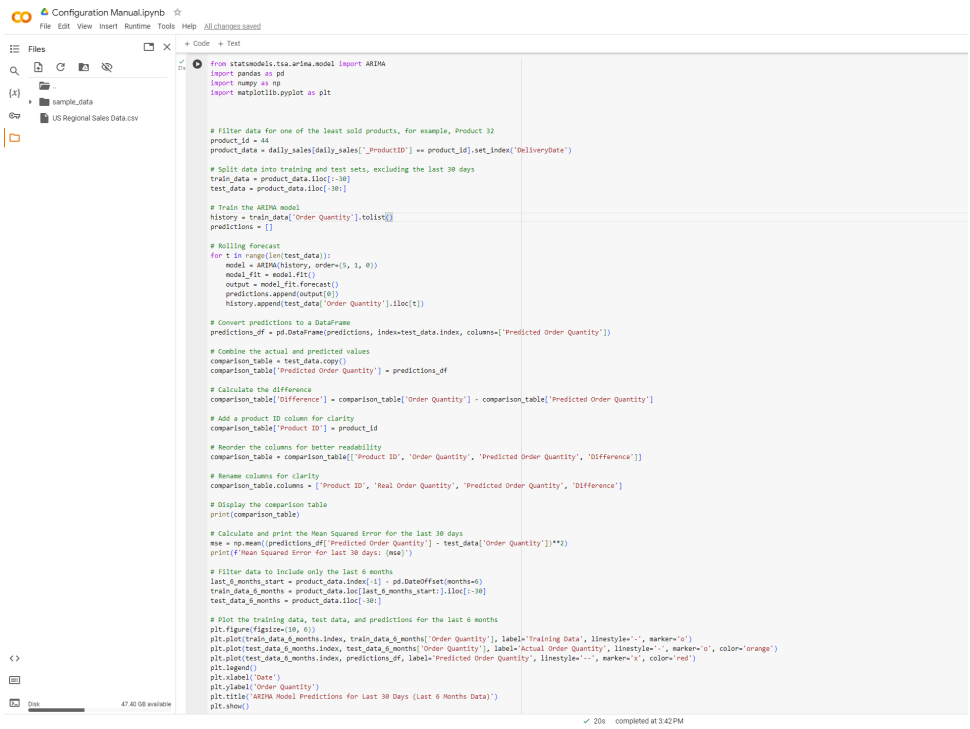


Figure 10: Running the ARIMA Model

6.4 LSTM Networks

6.4.1 Data Normalization and Preparation for LSTM

In order to get the dataset ready for LSTM networks, the following actions are taken:

- **Data Preprocessing**: Relevant columns are converted to the proper data types, and unnecessary columns are eliminated, much like with ARIMA preparation.
- **Product Filtering**: The dataset is filtered to include the five products with the least total sales.
- **Resampling**: The data is resampled to a daily frequency, and missing days are filled with an order quantity of zero.
- **Data Transformation**: The data is scaled and made into a supervised learning problem in order to turn the dataset into a format that can be used to train LSTM models.

To ensure that the dataset is clean and in a suitable format for training LSTM models—which are especially well-suited for capturing temporal relationships in time series data—a similar preparation to that done in ARIMA is being carried out here.

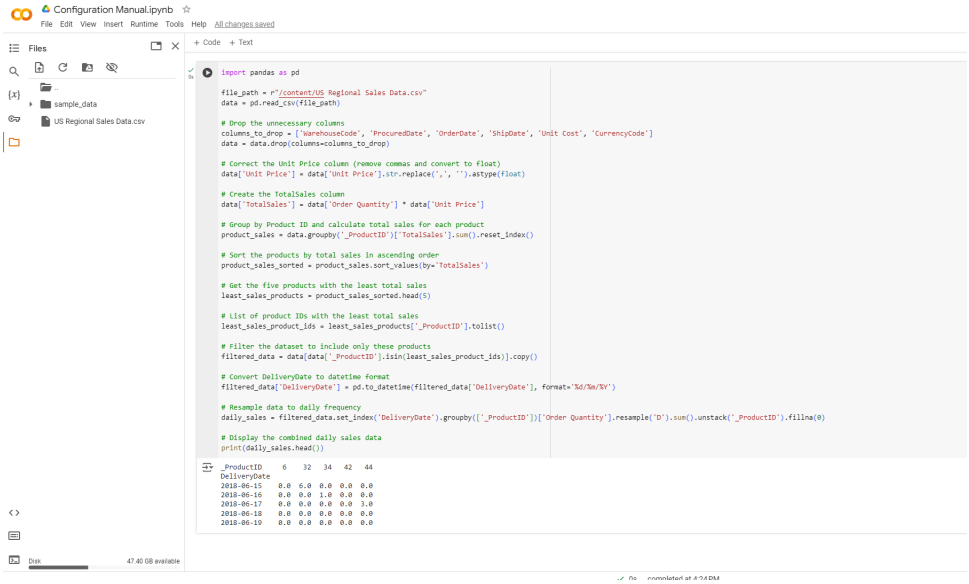


Figure 11: Data Preparation for LSTM Networks

The next steps are to define the architecture of the LSTM model, train the model on the prepared dataset, and assess the model's performance on a test set. The objective of this procedure is to increase the accuracy of order quantity projections by utilizing LSTM networks' capacity to represent sequential data.

Figure 12: Data Normalization and Preparation for LSTM

- **Normalize the data**: The MinMaxScaler function from the sklearn library is used to standardize the data. In doing so, the data is scaled to a range of 0 to 1.
- **Prepare the data for LSTM**: The dataset is transformed into sequences suitable for LSTM input. A sequence length of 30 is used, meaning each input sample will contain 30 time steps.
- **Split data into training and test sets**: Eighty percent of the data is used for testing and the remaining twenty percent is used for training. This split facilitates the assessment of the model's performance with unknown data.

- **Display the shapes of the training and test sets**: The shapes of the training and test sets are displayed to verify that the data has been correctly split and prepared.

In order to prepare the data for training the LSTM model, this procedure makes sure it is normalized, converted into sequences, and divided into training and test sets.

6.4.2 Building and Training the LSTM Model

- **Build the LSTM model**: The Keras library is used to generate a sequential model. The model consists of two 50-unit LSTM layers, and then a 5-unit Dense layer. The Adam optimizer and mean squared error loss are used in the compilation of the model.
- **Train the model**: The model is trained on the training data for 20 epochs with a batch size of 1. The training process is verbose, providing detailed output for each epoch.
- **Display the model summary**: An overview of the layers and parameters is given by displaying the model architecture summary.

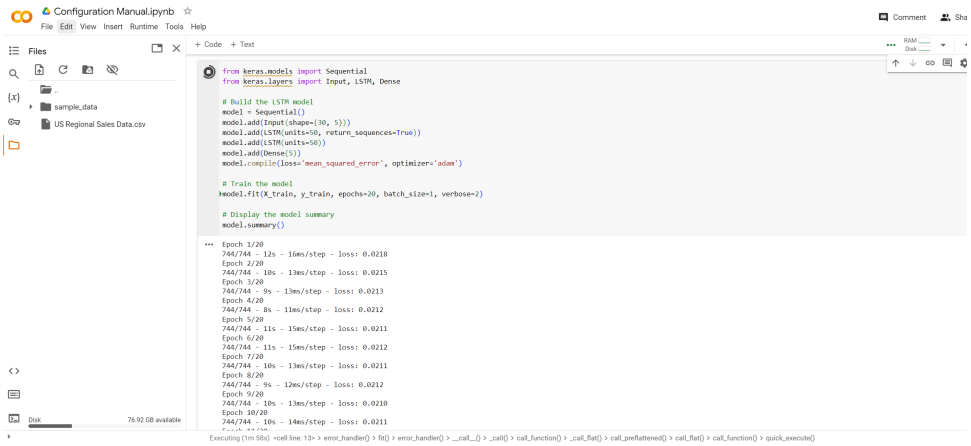


Figure 13: Building and Training the LSTM Model

This method describes how to create, train, and summarize the long short-term memory (LSTM) model, which is intended to forecast future values using the prepared dataset.

6.4.3 Evaluating and Forecasting with the LSTM Model

- **Make predictions on the test set**: On the test set, predictions are made using the trained LSTM model.
- **Calculate MSE for the test set**: The Mean Squared Error (MSE) for the test set predictions is calculated to evaluate the model's performance.
- **Create a comparison DataFrame**: A DataFrame is created to compare the actual values and the predicted values for the test set.

- **Make predictions for the next 30 days**: The model is used to forecast the order quantities for the next 30 days.
- **Visualize the results**: The historical data and the forecasted data for the next 30 days are plotted to visualize the model's performance and predictions.
- **Display the forecast table**: The forecasted values for the next 30 days are displayed in a table.
- **Display the comparison table**: The comparison of actual and predicted values on the test set is displayed in a table.

```
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Ensure the LSTM model variable is correctly defined
ar_lstm_model_final = model  # or ar_lstm_model if that's the name you used

# Make predictions on the test set
test_predictions = ar_lstm_model_final.predict(X_test)

# Inverse transform the predictions and the actual values
test_predictions = scaler.inverse_transform(test_predictions)
y_test_inverse = scaler.inverse_transform(y_test)

# Calculate MSE for the test set
mse = mean_squared_error(y_test_inverse, test_predictions)
print(f'Mean Squared Error on Test Set: {mse}')

# Create a DataFrame to compare actual and predicted values
comparison_df = pd.DataFrame(y_test_inverse, columns=daily_sales.columns, index=daily_sales.index[-len(y_test_inverse):])
comparison_df.columns = [f'Actual {col}' for col in comparison_df.columns]
for i, col in enumerate(daily_sales.columns):
    comparison_df[f'Predicted {col}'] = test_predictions[:, i]

# Make predictions for the next 30 days
forecast = []
current_input = scaled_data[-30:].reshape(1, 30, 5)  # Use scaled_data instead of scaled_residuals
for _ in range(30):
    prediction = ar_lstm_model_final.predict(current_input)
    forecast.append(prediction[0])
    current_input = np.append(current_input[:, 1:, :], prediction.reshape(1, 1, 5), axis=1)

# Inverse transform the predictions to the original scale
forecast = scaler.inverse_transform(np.array(forecast).reshape(-1, 5))
forecast_index = pd.date_range(start=daily_sales.index[-1] + pd.Timedelta(days=1), periods=30, freq='D')
forecast_df = pd.DataFrame(forecast, index=forecast_index, columns=daily_sales.columns)

# Visualize the results
plt.figure(figsize=(12, 6))
for column in daily_sales.columns:
    plt.plot(daily_sales.index[-180:], daily_sales[column][-180:], label=f'Historical Data {column}', linestyle='-', marker='o')
    plt.plot(forecast_df.index, forecast_df[column], label=f'Forecasted Data {column}', linestyle='--', marker='x')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Order Quantity')
plt.title('LSTM Model Forecast for Upcoming 30 Days (Combined Data)')
plt.show()

# Display the forecast table
print("Forecast for the Next 30 Days:")
print(forecast_df)

# Display the comparison table of actual and predicted values for the test set
print("Comparison of Actual and Predicted Values on Test Set:")
print(comparison_df)
```

Figure 14: Evaluating and Forecasting with the LSTM Model

These steps, offers a thorough assessment of the LSTM model's predicting skills by analyzing its performance on the test set and utilizing it to project future order volumes.

6.4.4 Additional Metrics and Final Evaluation

- **Calculate additional metrics for the test set**: The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for the test set predictions are calculated to provide a more comprehensive evaluation of the model's performance.
- **Ensure the LSTM model variable is correctly defined**: The LSTM model variable is correctly assigned to ensure consistency in the code.
- **Predict using the LSTM model on the test set**: On the test set, predictions are generated using the LSTM model, and these predictions are subsequently inversely converted to the original scale.
- **Calculate MSE for the LSTM model**: The Mean Squared Error (MSE) for the LSTM model predictions is calculated and displayed to assess the model's accuracy.

Figure 15: Final Evaluation

In order to ensure a comprehensive evaluation of the LSTM model's performance and accuracy in predicting order quantities, this final section employs additional measures.

7 Conclusion

Using the data from the previous sections, the complete project's implementation process has been described in a clear, comprehensive, and orderly manner. Everywhere they were utilized, the required packages have been noted. To improve readability and comprehension, every code segment has a comment and is separated into sections.