



Hellenic Open University

Supply Chain Management

Postgraduate Dissertation

Artificial Intelligence Methodologies in Inventory Management

Efcharis Vourlioti

Supervisor: Petros Pallis

Patras, Greece, June 2025

Theses / Dissertations remain the intellectual property of students (“authors/creators”), but in the context of open access policy they grant to the HOU a non-exclusive license to use the right of reproduction, customization, public lending, presentation to an audience and digital dissemination thereof internationally, in electronic form and by any means for teaching and research purposes, for no fee and throughout the duration of intellectual property rights. Free access to the full text for studying and reading does not in any way mean that the author/creator shall allocate his/her intellectual property rights, nor shall he/she allow the reproduction, republication, copy, storage, sale, commercial use, transmission, distribution, publication, execution, downloading, uploading, translating, modifying in any way, of any part or summary of the dissertation, without the explicit prior written consent of the author/creator. Creators retain all their moral and property rights.



Artificial intelligence methodologies for inventory management

Efcharis Vourlioti

Supervising Committee

Supervisor:

Petros Pallis

Professor (Associate) at University
of the Aegean

Co-Supervisor:

Thomas Dasaklis

Assistant Professor at the
School of Social Sciences
of the Hellenic Open University

Patras, Greece, June 2025

Dedicated to my family

Abstract

Market complexity creates the need to adopt new management strategies which will keep the companies competitive. Especially, the inventory management sector is one of the most critical chapter of success. The customer's requirements have been increased. They always search for high quality and low-price products which they demand to be in the right place at the right time.

AI is a field of computer science which was introduced to simulate the human's intelligence. The adoption of AI in Inventory management is a revolutionary approach for the companies which can combine and analyze a huge volume of information, overcome complexity and learn from experience. Many famous companies have adopted AI methodologies, achieving high performance and cost reduction.

The purpose of this study it to analyze critical areas of inventory management such as demand prediction, stock levels optimization and reordering procedure and are presented AI techniques for achieving competitive advantage.

Specifically, the research is focusing deeply on the demand prediction of a furniture company. It is created a forecasting model with the help a time series dataset. This model, which will be created using Machine Learning methodologies, will forecast the Sales-Demand of the company for new orders that will be registered in its information system.

Python programming language, will help us in this target by exploring time series forecasting techniques, such as ARIMA , Prophet and correlation methodology. These methodologies will analyze the impact of other factors of the dataset on sales, and will build advanced forecasting models for the future demand of the furniture company in order to manage its inventory levels with the best way for increasing its performance.

Finally, it is going to be investigated the benefits of the AI implementation, and possible limitations of its use in Inventory management

Keywords

AI (Artificial Intelligence), Inventory Management, Machine Learning, Python

Μεθοδολογίες Τεχνητής Νοημοσύνης στην Διαχείριση Αποθεμάτων

Εύχαρις Βουρλιώτη

Περίληψη

Η πολυπλοκότητα της αγοράς έχει δημιουργήσει την ανάγκη υιοθέτησης νέων στρατηγικών διοίκησης από τις επιχειρήσεις που θα τις βοηθήσουν να παραμείνουν ανταγωνιστικές. Ειδικά, ο τομέας της διαχείρισης αποθεμάτων είναι ένα από τα πιο κρίσιμα κεφάλαια της επιτυχίας. Οι απαιτήσεις των πελατών έχουν αυξηθεί. Αναζητούν πάντα προϊόντα υψηλής ποιότητας και χαμηλής τιμής, τα οποία απαιτούν να βρίσκονται στο σωστό μέρος τη σωστή στιγμή.

Η Τεχνητή Νοημοσύνη είναι ένας τομέας της επιστήμης των υπολογιστών που εισήχθη για να προσομοιώσει την ανθρώπινη νοημοσύνη. Η υιοθέτηση της Τεχνητής Νοημοσύνης στη διαχείριση αποθεμάτων αποτελεί μια επαναστατική προσέγγιση για τις εταιρείες, η οποία μπορεί να συνδυάσει και να αναλύσει έναν τεράστιο όγκο πληροφοριών, να ξεπεράσει την πολυπλοκότητα και να μάθει από την εμπειρία. Πολλές διάσημες εταιρείες έχουν υιοθετήσει μεθοδολογίες Τεχνητής Νοημοσύνης, επιτυγχάνοντας υψηλή απόδοση και μείωση του κόστους.

Σκοπός της παρούσας μελέτης είναι να αναλύσει κρίσιμους τομείς της διαχείρισης αποθεμάτων, όπως η πρόβλεψη της ζήτησης, η βελτιστοποίηση των επιπέδων αποθεμάτων και η διαδικασία αναπλήρωσης αποθέματος, ενώ παρουσιάζονται τεχνικές Τεχνητής Νοημοσύνης για την επίτευξη ανταγωνιστικού πλεονεκτήματος.

Συγκεκριμένα, η έρευνα εστιάζει σε μεγάλο βαθμό στην πρόβλεψη της ζήτησης μιας εταιρείας επίπλων. Δημιουργείται ένα μοντέλο πρόβλεψης με τη βοήθεια ενός συνόλου δεδομένων χρονοσειρών. Αυτό το μοντέλο, το οποίο θα δημιουργηθεί χρησιμοποιώντας μεθοδολογίες Μηχανικής Μάθησης, θα προβλέπει τις Πωλήσεις-Ζήτηση της εταιρείας για νέες παραγγελίες που θα καταχωρηθούν στο πληροφοριακό της σύστημα. Η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί για την ανάλυση είναι η Python η οποία θα μας βοηθήσει σε αυτόν τον στόχο, εξερευνώντας τεχνικές πρόβλεψης χρονοσειρών, όπως ARIMA, Prophet και μεθοδολογία συσχέτισης. Αυτές οι μεθοδολογίες θα αναλύσουν την επίδραση άλλων παραγόντων του συνόλου δεδομένων στις πωλήσεις και θα δημιουργήσουν

προηγμένα μοντέλα πρόβλεψης για τη μελλοντική ζήτηση της εταιρείας επίπλων, προκειμένου να διαχειριστεί τα επίπεδα αποθεμάτων της με τον καλύτερο τρόπο για την αύξηση της απόδοσής της.

Τέλος, θα διερευνηθούν τα οφέλη της εφαρμογής της Τεχνητής Νοημοσύνης και οι πιθανοί περιορισμοί της χρήσης της στη διαχείριση αποθεμάτων.

Λέξεις – Κλειδιά : Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Διαχείριση Αποθεμάτων, Python

Table of Contents

Abstract.....	2
Περίληψη	3
Table of Contents.....	5
List of Figures	7
List of Tables	8
List of Abbreviations & Acronyms.....	9
1. Introduction.....	10
1.1 Problem Statement.....	10
1.2 Significance of the Research	11
1.3 Scope of the Research	11
1.4 Overview of the Thesis Structure	13
2. Literature Review	14
2.1 Theoretical background on Inventory Management	14
2.2 The role of forecasting in Inventory Management.....	17
2.3 The Power of AI in Inventory Management	18
2.3.1 Demand Forecasting	19
2.3.2 Logistics Activities	19
2.3.3 Procurement	21
2.4 Companies that have applied AI techniques in Inventory Management	22
2.5 Basic AI Concepts.....	23
2.5.1 Machine learning	24
2.5.2 Deep learning	24
2.5.3 Artificial Neural Network	24
3. Developing an Inventory management model	26
3.1 Software Selection	26
3.2 Set up the Program.....	27
3.3 Basic Model Creation.....	27
3.4 Checking the Accuracy of the model.....	29
3.4.1 Mean Absolute Error (MAE)	29
3.4.2 Root Mean Square Error (RMSE).....	29
3.4.3 Mean Square Error (MSE)	30
3.4.4 R- squared (R^2) Score.....	30
4. Research Analysis	31
4.1 Data description	31
4.2 Statistical analysis	33
4.3 Data preprocessing	33
4.3.1 Check for missing values and outliers	34
4.3.2 Decomposition of the Time Series Dataset.....	38
5. Modeling with Time Series Methodologies	41
5.1 ARIMA methodology	41
5.1.1 Tools for running ARIMA model.....	42
Autocorrelation and Partial Autocorrelation	42
Augmented Dickey-Fuller (ADF) test	44

KPSS Statistics.....	45
Difference Technique.....	45
5.1.2 Final running of ARIMA model for our data.....	48
5.2 SARIMA methodology.....	50
5.3 Prophet methodology.....	53
6. Modeling with multiple Linear Regression Methodology	57
6.1 Check for outliers and missing values	57
6.2 Variable Selection	57
6.3 VIF for multicollinearity	59
6.4 Error Metrics of Linear Regression	62
7. Discussion on Main Findings and Conclusion	66
7.1 Best Model Selection	66
7.1.1 Time Series Results.....	66
7.1.2 Multiple Regression model Results	67
7.2 Discussion	68
7.3 Conclusion	71
8. References.....	75
Appendix A: “Predicted Sales quantity for each group of price”	82
Appendix B: “Python Code”.....	83
B.1 Data Preprocessing	83
B.2 Time Series with Arima	87
B.3 Time Series with Prophet	95
B.4 Regression Analysis	96
B.5. Summary of Models and Best Model Selection.....	101

List of Figures

Figure 1: Relationship between AI,ML,DL (Bhatt et al., 2021).....	25
Figure 2: Why Python for AI	26
Figure 3: Average demand per month.....	31
Figure 4: Outliers of the independent numerical variables	35
Figure 5: Relationship between Z-scores and centiles, where the parameter is normally distributed.....	36
Figure 6: Monthly quantity sold Trend	37
Figure 7: Decomposition of multiplicative time series	39
Figure 8: Autocorrelation of furniture sales	42
Figure 9: Partially Autocorrelation	43
Figure 10: Average quantity difference between consecutive time points	46
Figure 11: Autocorrelation Chart with the average quantity difference between consecutive time points.....	46
Figure 12: Final Autocorrelation and Partial Autocorrelation Chart with the average quantity difference between consecutive time points	48
Figure 13: Arima model with parameters ($p=4, d=1, q=1$).....	49
Figure 14: Fitting of Arima model to the data	50
Figure 15: Results of 1st seasonal SARIMA model $(4,1,1)(1,1,0,12)$	51
Figure 16: Forecasts on Train and test data from SARIMA $(4,1,1)(1,1,0,12)$	52
Figure 17: Forecasting data for the next 3 years with SARIMA model $(4,1,1)(1,1,0,12)$...	52
Figure 18: Forecasted Values with Prophet Analysis	55
Figure 19: Graph of the Prophet model	55
Figure 20: Seasonality based on the Prophet model	56
Figure 21: Correlation between Price and other variables	58
Figure 22: Multicollinearity of the variables with VIF.....	59
Figure 23: Actual VS Predicted Quantity Sold.....	68

List of Tables

Table 1: Statistical description of the basic numerical values	33
Table 2: Average qty per Month	34
Table 3: Statistical data of numerical variables after removing outliers	36
Table 4: Result of ADF Test for average furniture quantity sold per month.....	44
Table 5: Result of KPSS statistic for average furniture quantity sold per month.....	45
Table 6: Result of KPSS statistic for average furniture quantity sold per month.....	47
Table 7: Result of ADF Test for average furniture quantity sold per month.....	47
Table 8: Overall Results of Time Series Analysis	53
Table 9: Prophet model error metrics	56
Table 10: Linear Regression model Error Metrics	63
Table 11: SVR model Error Metrics	63
Table 12: Decision Tree model Error Metrics	63
Table 13: KneighborsRegressor model Error Metrics	64
Table 14: XGB Regressor model Error Metrics	64
Table 15: Random Forest model Error Metrics	65
Table 16: Error Metrics of Time Series comparison	66
Table 17: Comparison between multivariable regression models	67

List of Abbreviations & Acronyms

ACF: Autocorrelation Function
AI: Artificial Intelligence
AIC: Akaike Information Criterion
ANN: Artificial Neural Networks
ADF: Augmented Dickey-Fuller
BIC: Bayesian Information Criterion
DL: Deep Learning
EDA: Exploratory Data Analysis
GPU: Graphics Processing Units
KPSS: Kwiatkowski–Phillips–Schmidt–Shin test
LM: Linear Regression Model
LSTM: Long Short-Term Memory
MAE: Mean Absolute Error
MAPE: Mean Absolute Percentage Error
ML: Machine Learning
PACF: Partial Autocorrelation Function
RMSE: Root Mean Squared Error
RNN: Recurrent Neural Network
SD: Standard Deviation
TPU: Tensor Processing Unit
VIF: Variance Inflation Factor

1. Introduction

This chapter is an introduction to the research approach by explaining the problem that was the motivation for this thesis. It presents the significance of the research and describes the basic scope that this thesis aims to explore. In addition, it provides a short description of the chapters, offering a framework for the reader to better understand the key components and results of this thesis.

1.1 Problem Statement

The basic challenge that faces the companies today is uncertainty. The market complexity, in combination with the rapid evolution of technology, brought the companies in front of a big challenge of making fast decisions and solving complex problems in order to always be competitive.

Many businesses are forced to maintain optimal inventory levels, leading to issues such as overstocking, stockouts, high holding costs, and inaccurate inventory data. Additionally, the rise of e-commerce has raised the demands of real-time inventory visibility and faster fulfillment, which traditional inventory systems often cannot support. The lack of integrated, technology-driven solutions results in inefficiencies, poor forecasting, and reduced customer satisfaction.

Nowadays, AI has come to cover this gap by making machines work by mimicking the human brain. This happens through providing high levels of criticism in decision-making rather than just automating the tasks (Moawad et al., 2024).

Artificial intelligence is a sector of computer technology that has as its purpose to simulate human intelligence. The term “AI” was first introduced in 1956, in an academic conference at Dartmouth College in Hanover (Robinson, n.d.). John McCarthy, who is considered to be the father of AI, said that “*AI is the science and engineering for making intelligent machines*” (Rajaraman, 2014). The application of AI in inventory forecasting is a big challenge for the companies because it gives a chance to the user to analyze a huge volume of information, such as demand and expected quantities.

The applications of AI can significantly impact inventory management practices. By applying the capabilities of AI, it is investigated how the companies can achieve maximum levels of optimizing inventory operations. Demand forecasting, optimizing warehouse

operations, predicting safety stock, and planning replenishment are some of the processes of inventory management that can be calculated fast with the help of AI.

1.2 Significance of the Research

Inventory management is a vital sector of companies and plays a great role in their success. This research offers significant value as it addresses the ongoing challenges companies face in balancing inventory levels, minimizing costs, and meeting customer demands in an increasingly dynamic and competitive market environment.

With rising customer expectations, product diversity, and global supply chain complexities, organizations must adopt smarter, more efficient inventory strategies. The research evaluates various forecasting techniques with the Python programming language, such as predictive analytics with machine learning models, and analyzes their role in improving demand forecasting accuracy. This will be done by investigating historical data of a furniture company, and it will be investigated how the analysis of them contributed to the furniture demand predictions for the future.

The significance is high because it is explored how AI offers powerful tools to transform how inventory is managed. It is enhanced forecasting accuracy by detecting anomalies in order to automate inventory procedures. Time series methodologies and linear regression models are going to be analyzed, and it will be investigated how these can help the user to make fast and best decisions on selecting the model that fits best to the data.

1.3 Scope of the Research

The purpose of this dissertation is to analyze various AI methodologies that have been adopted by the managers in the sector of inventory management. Especially the research will focus on machine learning forecasting models with the use of the Python programming language. Specifically, the machine learning models that will be analyzed will focus on demand prediction of a furniture company, and it will be investigated which model gives more accurate results. The methodologies that will be analyzed are very popular, and the demand prediction opens the way for simplifying operations performance, improving customer satisfaction, and achieving competitive advantage.

Except for the theoretical analysis of the methodologies, the research will focus on the analysis of a real dataset of a furniture company.

The main goal of the research is to answer a set of questions aimed at uncovering meaningful patterns and relationships within retail sales data of a furniture company that could help to improve the accuracy of demand predictions by investigating the demand patterns, focusing on trends, seasonal behaviors, and recurring sales dynamics. Secondly, it aims to analyze the impact of variables such as pricing, product category, customer etc. on sales performance, providing insights into how these variables shape consumer behavior and retail demand.

The dataset will be analyzed by quantitative forecasting methods such as time series and multiple regression models, and it will have as a purpose to answer the questions below:

- ✓ Which AI forecasting models (e.g., ARIMA, Random Forest, XGBoost) provide the most accurate predictions? What role do data quality and availability play in the performance of AI-driven forecasting models?
- ✓ How can AI models improve the accuracy of demand forecasting in inventory management compared to traditional methods? What types of AI algorithms (e.g., neural networks, decision trees, time series models) are most effective for forecasting inventory needs in dynamic markets?
- ✓ Which patterns are identified in the furniture sales of the research company and how is this pattern change in different years and months?
- ✓ How do pricing strategies impact sales performance and consumer behavior?

By using AI forecasting models, all these questions will be answered by following a structured research methodology. It will be determined the steps, tools, and techniques that will be used to collect and analyze data, ensuring that the research is reliable, replicable, and valid. It is going to be evaluated the significance of the forecasting models based on error metrics and it will be derived valuable outcomes.

The dissertation concludes with a discussion on the outcomes which reply to the questions of the research. Moreover, will be present possible opportunities that are created by the use of AI in inventory management and future implications of this research. It will also describe the limitations and the hindrances of the analysis and the use of AI.

1.4 Overview of the Thesis Structure

This section provides a summary of the chapters included in the thesis and the purpose of each. It helps readers understand the logical flow and how each part contributes to the overall research objectives.

The dissertation is structured as follows:

The 1st Chapter, gives an introductory view about the topic of the thesis, goal, and objective behind it.

Chapter 2 provides an overview of the major concepts of inventory management and models used in this field of knowledge. It provides a comprehensive review of existing studies and theoretical frameworks related to inventory management, demand forecasting, and the use of artificial intelligence (AI) and machine learning in supply chain optimization.

Chapter 3 analyzes the steps for creating machine learning models and describes variable methodologies that can help to reach this goal.

Chapter 4, it is presented the Research Analysis of the dataset. It will be focused on the main characteristics of the given variables and on the preprocessing of the data in order to execute the forecasting models.

Chapter 5 focuses on time series analysis. The evolution of the dependent variable is investigated in comparison with the independent variable.

Chapter 6 applies multiple regression model analysis, where the relationship between the variables is investigated, and multiple models compared.

Chapter 7 summarized and discussed the key findings by selecting the best model depending on the error metrics results. Except for this, a discussion on the results is presented with valuable results as well as offers recommendations for companies considering AI in inventory management and suggests areas for future research.

2. Literature Review

The chapter focuses on existing literature on the topic of Inventory management. It explains Inventory management practices and AI techniques and how this contribute to Inventory management.

2.1 Theoretical background on Inventory Management

Inventory is a basic sector of every company which has to focus on. Market competition and the increasing variety of products available worldwide have raised customer expectations, leading to continuously growing demand. The management of a company becomes very complicated if the stock levels are exceed their needs. If a company wants to be present in a competitive market, it needs to concentrate on the area of inventory management. According to Oluwaseyi, J. A., 2017, the “Inventory” as a term can be categorized in three groups: a. inventory of raw materials, b. inventory of work-in-progress and c. inventory of finished goods (Oluwaseyi et al., 2017).

The raw materials, are an investment for a company, which purchases products that will be consumed from production in order to produce the final products. The work-in progress products is the intermediate step where the products are partially completed but they are not finished. All these steps should be combined well for the smooth operation of a company. The ensuring of the smooth running of a company is done with the help of inventory management. In inventory management is important to focus on the factors that affect the inventory optimization. The general purpose of inventory management is to ensure good levels of raw materials, mill-finished products, and final products in order to cover its operational needs (Praveen K B & Bangalore Institute Of Technology, 2020).

Inventory is linked directly to the logistics activities and it is an essential control for manufacture, wholesale and retail business. Without this link of inventory and logistics it is not possible to achieve a company successful strategy for its business. Inventory management is an basic component of supply chain management which controls the flow of products of a company from the supply up to consumption in order to cover customer needs (D. Singh & Verma, 2018). According to (Chalotra, 2013), inventory management is a process where the company make actions in order to ensure the effective flow of goods services and products. It is one of the most important sectors of a company, which highly affects its performance (Sustrova, 2016). Moreover, (Oluwaseyi et al, 2017) mentions in his

article that inventory management is one of the most important process of a company as it relates purchase, sales and logistics procedures (Oluwaseyi et al., 2017). It is obvious that inventory management is vital for an organization in order to ensure the existence of best stock levels which combine coverage of demand with the minimum cost.

Inventory management, except for the smooth running of the materials flow, also faces some hindrances. According to (D. Singh & Verma, 2018) there are four hindrances that the company should pay attention in order to manage the inventory properly. The first problem that needs to be handled, is the *transportation cost*. The input and output of the products are interpreted as movements of items. These movements mean extra cost for the companies that have to transport the products fast and with low costs. When we speak about costs in transportation we mean time, distance, and energy consumption that are cause of unnecessary transportations (Jat & Muyldermans, 2013).

It is important for a company to ensure that the right quantity of goods will be available at the right time and in the right place. Effective inventory management is the system that ensures the requirements of the production process and the coverage of market needs by minimizing inventory and lowering costs. This will have as a result the maximization of customer satisfaction and the preservation of operational efficiency.

One second problem that inventory management is facing is the *holding cost*. The holding cost, is the total expenses that the company needs to take into account for storing and maintaining inventory for a specific period of time. These expenses can be considered the storage fees, possible insurance for storing, depreciation, etc. As it is obvious, it is very important for the company to manage the inventory properly in order to keep the holding costs levels low and to sell or produce just in time for achieving lower prices with better services.

The next problem is the *ordering cost*. These costs refer to the expenses that a company has to pay in order to replenish its stock. The ordering cost is not direct cost which is combined to the product, but is indirect cost which has to do with the administration cost (people who prepare the purchase orders), the communication cost (e-mails, calls, time from the personnel), receiving (inspection of the received goods).

Finally, the most important problem is the *maintenance cost*. Maintenance cost is the general guideline which an organization follows in order to ensure that the equipment, machinery, and systems are working in good condition and performing efficiently every time. It is very

crucial for a company to keep in mind maintenance costs in inventory management in order to ensure safety, maximize productivity and other industrial operations.

So it is obvious that accuracy of inventory management ensures the company's competitiveness, and its vital role is determined by the combination of sales, purchases, and logistics activities (Oluwaseyi et al., 2017).

All the hints that were referred to, were crucial to be controlled by the organizations. Up to the end of 1990, most of the publications were about traditional inventory management models. The inventory management was focusing on manual functions and simple techniques. Also, the traditional methods of inventory management relied on historical data, which could not forecast the demand of a dynamic market.

All these traditional methods were not enough for leading in such a complicated marketplace. The increase of consumer demands in combination with the globalization of the market, created the necessity to acquire new technologies from the companies, which could make them more adaptable to the complicated supplier networks. According to Ghiani et al, (2023), traditional demand analysis was not enough to predict all the factors that affect customer behavior (Ghiani et al., 2013). By the use of technological advancement, the companies would be able to implement innovative strategies for enhancing efficiency and customer satisfaction.

According to (Özer, 2011) there are four fundamentals for an effective inventory management. a: the know-how, b: measuring performance c: management of operations and d: effective tools. The need for measuring the performance of the inventory management process and the need to take actions for improvement, requires the managers' ability to use the existing tools and the ability to interpret the available information properly. They must also have the knowledge to quantify the data and interpret it, and to act fast as possible, and make decisions. The measuring of performance could enhance the efficiency the accuracy, and cost-effectiveness of a company.

The inventory management performance is affected also by the way the inventory was trucked, stored, and replenished. It is a complicated and multivariate process, where a system structure improvement was mandatory. By improving the structure of inventory systems, the companies could ensure smoother operations, reducing costs and enhancing accuracy across the supply chain. So, the optimizing of the tools, processes and communication channels could achieve stronger decision making through real time data and reporting.

The technology integration was the solution on the previous hindrances. The improving of the digital tools, the automations and streamline in inventory related processes transformed from a manual, ordinary process into an automated, intelligent and proactive system. As an example of the future in inventory management we can refer the robots in supply chain (AlRushood et al., 2023). The integration of robotics and the advancement of technology generally had as a result lower costs, accuracy and customer satisfaction. Looking ahead, the future of inventory management will depend on the harmonization of technological advancement with sustainability and resilience. This integration is essential not only to meet the expectations of consumers but also to drive sustainable growth and maintain a competitive in the global market.

In the next chapters, will be analyzed how the technology integration improved the inventory management processes and how crucial was the role of artificial intelligence in this sector.

2.2 The role of forecasting in Inventory Management

Lambert and Stock positioned forecasting as “the driving force behind all forward planning activities within the firm” (Stock & Lambert, 2009). There are many requirements that should be covered for an accurate inventory forecast (Schary, 1984).

Firstly, it is important to take into account the time of ordering. There is always a gap between the order registration and the delivery of this order. So that means that the company should have taken this gap into account in order to maintain the stock at the correct levels by calculating the safety stock.

Also, the inventory forecasting requires good cooperation with production planning in order to ensure that the future demands will be covered. So the facilities and equipment requirements can be adjusted in order to increase or decrease the capacity. Except for this, forecasting methods may reveal patterns of demand change during the year, which means adaptations in the production and inventory requirements.

For all the above, it can be said that a well-designed and adaptable supply chain network is a prerequisite for effective inventory forecasting.

The Market offers plenty of software for inventory forecasting. It is very important for a company to be able to select an inventory forecasting software that fits its needs. The right knowledge and approach to the software that will be used by a company is the most important factor of a secure solution for optimizing its inventory.

A big variety of inventory forecasting methods have been developed, and it is a very complex job to select the best model. Magee et al. (1985) distinguish the forecasting methods in two categories: the long term and the short term (Magee et al., 1985). Also, Makridakis and Wheelwright (1977), identified six factors that affect the evaluation of the forecasting method. The factors are: a. accuracy, b. forecast time horizon, c. value of forecasting, d. availability of data, e. type of data pattern, and f. the user's experience (Makridakis & Wheelwright, 1977). Nowadays, except for the basic factors that are taken into account and were referred to previously, there are more key features that affect good inventory management software, such as the user-friendly interface, the support and updates, and the customization.

2.3 The Power of AI in Inventory Management

AI has been applied in recent years, and its role in inventory management is vital. Dhaliwal et al. (2023) mentions that the AI involvement in inventory management increases the profitability of the company by improving stock levels and demand forecasting accuracy (Dhaliwal et al., 2023). Also, Singh N. gives emphasis to the many advantages of AI in inventory management (N. Singh, 2023). He also believes that AI techniques have affected a lot of inventory management applications, such as demand forecasting, stock optimization, automatic reordering, supplier selection, and relationship management. The same view is also held by other researchers who believe in the role of AI in inventory management and especially in the identification of dead stock (Cherukuri & Ghosh, 2016).

AI contributes to many sectors today, like healthcare, engineering, medicine, etc. The most significant part of it is that the machines have the ability to understand and to make decisions. Many companies, such as Zara, Amazon, and Walmart have implemented AI processes in their supply chain operations, achieving high performance and cost reduction (Stanisławski & Szymonik, 2021).

Inventory management is affected by the internal and external sources (raw materials and products in the supply chain), which should be combined correctly by good planning, controlling, and tracking of the materials that are moved into, through, and out of the firm. So inventory planning, inventory tracking and inventory control, the procurement process, and inventory optimization are the basic elements for an effective network to fill the customer requests and maximize their satisfaction.

Below is going to be analyzed the basic inventory management elements and how the AI implications have contributed to the performance of the processes and the firm's profitability generally.

2.3.1 Demand Forecasting

Demand forecasting is the process of predicting the customer needs for goods and services in the future (Acar et al., 2014). An accurate prediction of demand is crucial for the profitability of a company. It can avoid overstock and stockout and maximize customer satisfaction by permanent availability with the lowest cost and investment.

Initially the demand forecasting was based on traditional methods especially by monitoring and interpret historical data and making calculations according to them. Early statistical forecasting models were the exponential smoothing models, time series analysis, and regression analysis (Hyndman & Athanasopoulos, 2018). The traditional techniques were not able to capture the unexpected factors that can affect the company's performance, so they were not reliable for the non-linear data (Gutierrez et al., 2008).

AI demand forecasting techniques were introduced in order to combine historical data, market trends, competitors' behavior, and possible customer needs.

The demand forecasting can be designed by artificial intelligence models using artificial neural networks. Algorithms used for training such models include batch gradient descent and variable learning rate (Plinere et al., 2015).

2.3.2 Logistics Activities

✓ Automatic stock allocation

Stock allocation can be automated with the help of machine learning, enhancing the order fulfillment process. It takes into account historical data, seasonality, and customer behavior, algorithms predict future demand patterns, allocating the stock of items in the orders. With this process, it is reducing the delays and risk of stockout or overstock and helps with a faster and smoother order execution process, which improves inventory management and customer satisfaction. (Vanessa Munoz Macas et al., 2021)

✓ Automatic reordering

The application of automatic reordering with the help of AI gave great precedence in inventory management. AI systems gave the possibility to a company to place orders automatically based on the demand forecasting and the stock levels. This process can give the company a competitive advantage because it saves time and can prevent human errors. Specifically, it combines data and, with a predictive analysis, manages to keep optimal inventory levels and orders products automatically as needed (Eckert, 2007). With this process and by taking into account market changes and seasonality, it updates the reorder point when it is needed, preventing overstocking and stock-outs, ensuring the availability of the products, and enhancing customer satisfaction.

✓ Real-time tracing and controlling

An important chapter of inventory management is the effectiveness of information use. When a firm knows how to use information effectively, it gives to the manager the chance to act fast and make decisions about which materials to buy and store. So the accuracy of real-time product tracking is vital for important decisions that are taken by the managers (Oluwaseyi et al., 2017).

AI has also been involved in this area by the Internet of Things (IoT). According to Nyathani, 2023, IoT devices are placed on materials or assets and help the company to have information on the location, status, and condition of them. So it is possible to have real-time insight into the movement of the items (Nyathani, 2023). AI algorithms in these devices can give solutions to potential delays by identifying patterns, and they have the possibility to prevent obstacles or transportation delays.

✓ Data quality

The data quality can be achieved with the help of Robotic Process Automation (RPA) (Yarlagadda, 2018). The same author has claimed that the introduction of robotic models in inventory systems can achieve accuracy by processing routine tasks such as data entry and inventory tracking.

✓ Stock Optimization

AI systems gave to the companies the possibility to monitor the stock levels, the stock movements, and the stock location real-time in the warehouse (Lebhar et al., 2023). AI-driven robotics was a revolution for picking activities. Moreover, Autonomous Robotics

Mobile Robots (AMRs) have been designed with AI algorithms in order to navigate in the warehouse and move products of an order in the picking location by selecting the best route without obstacles and with the lowest traffic. The most important thing is that these robots have the possibility to prioritize orders, and to eliminate the execution time of an order minimizing the human effort. These flexible systems can offer maximum performance to the company by achieving accuracy and customer satisfaction (Grover & Ashraf, 2024).

Safety stock is the stock that help the company to feel comfortable that it will be always ready to cover the customer needs. In everyday activities many unexpected incidents can happened by either fluctuation of demand or delays in supply. So the safety stock works as a buffer against all the unexpected incidents. The implications of AI technologies enable dynamic stock optimization by ensuring the optimal safety stock based on forecasting the demand and supply fluctuations.

✓ Route Optimization

The route optimization is part of the transportation planning is a procedure that is executed help of AI algorithms. It can get historical data, and with the help of a machine learning platform, it analyzes traffic data and road conditions and project requirements to determine the most efficient routes for the materials transportation (Oluwatoyin Ajoke Farayola et al., 2023). Except for these factors, the AI-driven programs can take into account fuel consumption and weather conditions and adjust the routing options according to the case. The AI-driven logic adaptation has helped in overcoming the static transportation models and contributes to the efficiency of the transportation process and generally to the customer satisfaction.

✓ Storage and Warehousing

The evolution of warehouse automation with the help of AI technology assisted in the enhancement of storage and warehousing tasks. Robots designed by AI algorithms gave the possibility for automating everyday tasks such as packaging and storing.

2.3.3 Procurement

Procurement is a critical part of inventory management and highly affects the business decisions. Traditionally, the procurement process was done by static calculations, which were time-consuming, didn't ensure accuracy, and were sensitive to human errors. AI

created new opportunities in this sector and enhanced the value of the company. According to Loo and Santhiram (2018), AI is applied in the procurement sector and especially in the B2B market to automate tasks (Loo & Santhiram, 2018). There are many authors that have studied the role of AI in the procurement process, such as (Brintrup et al., 2020) and (Dubey et al., 2018).

The AI algorithms can make a demand analysis and forecast the material requirements accurately (Anamu et al., 2023). The AI procurement software programs can automate various tasks, such as supplier selection, and recommend optimal procurement strategies, taking into account the competition. The automative processes can help in overcoming manual routine approvals by analyzing historical approval patterns and by taking into account risk factors (Sanni et al., 2024). This means acceleration of routine tasks and taking fast decisions with accuracy.

Also, AI technology can promote the negotiation strategies by giving the procurement team a combination of information such as payment terms, prices, and market conditions, which will affect the negotiation strategy that the company will follow with the potential suppliers.

2.4 Companies that have applied AI techniques in Inventory Management

Case studies help to understand the impact of AI in inventory management and offer a deep understanding of how AI has affected the companies that have adopted AI methodologies for inventory prediction. Specifically, it can strengthen the argumentation of AI implementation.

Many famous companies have adopted AI techniques in order to improve their inventory management and to increase their profitability and customer satisfaction. Walmart, Zara, Alibaba, and Pfizer are some of them that, by the use of AI, achieved greater accuracy, efficiency, and responsiveness in their inventory management processes, leading to significant operational improvements and competitive advantages (Manaviriyaphap, 2024).

Zara, the fashion retailer, uses AI to enhance its inventory management and supply chain processes. According to Zhong & Chen (2022), the use of AI helped Zara maintain its competitive advantage in the fast-paced fashion industry (Zhong & Chen, 2022). According to Manaviriyaphap, (2024) the same logic was followed by the retail company Walmart, which integrated AI into its inventory management system. Both of the companies gather

the data from sales, market trends, and social media and make analyses in order to accurately forecast demand for their products. The continuous analysis helps the company to make accurate predictions and to be flexible. It has the possibility to adjust its inventory levels dynamically, ensuring that popular items are always in stock. At the same time, it minimizes overstock and under max and min levels.

Next, the famous company, Amazon, was among the first companies that applied AI models for its inventory management operations. AI helped Amazon to forecast customer demand, giving the company the possibility to maintain optimal stock and reduce delivery days. The AI predictive analytic models ensure that Amazon has the right amount of stock to meet customer demands. So they could avoid stockout and sales losses. Except for this, AI helped Amazon to personalize the customers' preferences. The algorithms of AI are analyzing the browsing and purchasing of each customer and recommending offers that are close to his preferences or close to other customers' searches with similar preferences.

Another important example is the pharmaceutical company Pfizer. The industry adopted AI for inventory management to ensure the timely availability of critical medications. Reliability is a very critical factor for the company, which applied AI in order to make accurate predictions of the demand for critical medication in order to be available to the customer at a particular time because this will have a serious impact on the patient's health.

2.5 Basic AI Concepts

The biggest evolution of AI was in the 20th century. The development of AI created new concepts, which was revolutionary for the technology. The AI goal was to make machines able to solve complex problems and make decisions by mimicking the human brain. This happens through providing high levels of criticism in decision-making rather than just automating the tasks (Moawad et al., 2024).

Some examples of AI technology revolutions were AI applications in medical projects and self-driving cars. Below are presented and described the basic dimensions of AI: a) deep learning, b) machine learning, and c) artificial neural networks. These concepts are combined together and are demonstrated in Figure 1, which shows how AI, ML, and DL relate to each other.

2.5.1 Machine learning

Machine learning, is a subcategory of AI. It is a number of algorithms that help the machines to learn from their own actions. It focuses on learning and improving independently of experience. The machine gets inputs, and by analyzing the data, it creates patterns of these data. The data can be numbers, texts, photos, etc. All these data are gathered and prepared to be entered in the algorithm. The more data, the better the algorithm. It is used mainly in sectors such as speech recognition, robotics, and predictive analytics (Blasch et al., 2021).

The process that programmers follow is to select a machine learning model, then upload the data, and after that, let the model train itself. By this process they find patterns in order to make predictions.

There are three categories of machine learning. Supervised, unsupervised, and reinforcement (Muhammad & Yan, 2015). When we speak about supervised machine learning model we mean models that use existing data for forecasting. In this category the best result is achieved only if the programmer has defined a specific dataset. Unsupervised is the machine learning model in which the programmers make predictions without previous experience. In this category, machine learning can give patterns that people were not looking for. The algorithm finds patterns without the need for human involvement. Finally, the reinforcement learning model trains machines to get the best decisions by finding errors and helping them to learn over time and indicating what action it should take.

2.5.2 Deep learning

Deep learning, is a development step of machine learning. According to Addo et al., 2020, “deep learning is part of machine learning, and it is based on useful data representations or features from the raw data” (Addo et al., 2020). The performance of machine learning is affected by the accuracy of the features that are identified. Deep learning, on the other hand, is trying to understand the high-level features from the data (Graves et al., 2013). It can imitate human logic and execute human tasks, such as describing images.

2.5.3 Artificial Neural Network

Artificial Neural Network (ANN) is part of AI and was introduced to overcome the traditional practices and to overcome their limitations. By acquiring ANN algorithms, the company can interpret the relationships between stock levels, customer needs, production planning, and customer satisfaction.

ANN simulates the activity of the brain and the way that it processes information. So it can accept many different inputs, and they have the possibility to analyze it and learn from patterns (Cerulli, 2023). The inputs are information with specific nodes from many sources, such as stock levels, customer needs, production planning, weather conditions, and customer satisfaction. Then these inputs are combined with nodes processed, and an outcome is exported. This outcome is evaluated for a separate set of data inputs and compared with a standard model. If the outcome meets the standard model and reaches the desired criteria (Abiodun et al., 2018). The figure below shows how the AI concepts are related to each other.

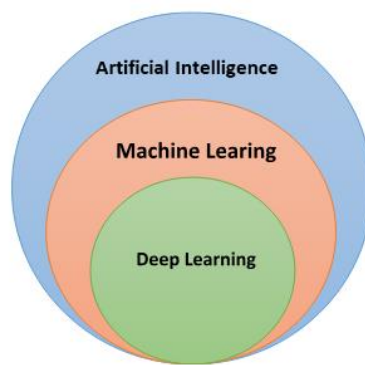


Figure 1: Relationship between AI,ML,DL (Bhatt et al., 2021)

3. Developing an Inventory management model

This chapter describes the research methodology on the way of developing a model for research, preparing for the analysis in the following sections. It gives a detailed overview of the scientific processes and techniques used for the model creation and gives plenty of information for further discussion and analysis.

3.1 Software Selection

Artificial intelligence means making the machines think and behave like a human. Machines are controlled by software, which is the platform where the capabilities of the machines are designed (Artasanchez & Joshi, 2020). The development of AI gave the possibility for improving the capabilities of computer hardware and software, giving new opportunities to inventory management (Albayrak Ünal et al., 2023).

There are many programming languages that can be used for developing artificial intelligence models, such as Java, Python, C++, etc. All these programming languages are very popular, but especially Python is the most widely selected. The biggest advantage of the Python language is that it has very simple syntax that can be learned easily. Also, Python for AI applications has libraries for all AI projects. NumPy, Matplotlib, and Pandas are some of the libraries that help with programming AI projects.

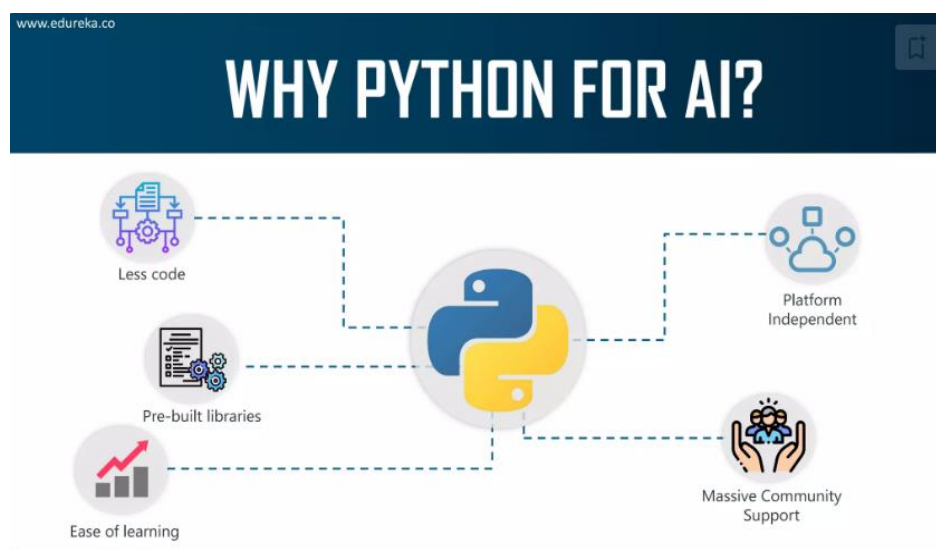


Figure 2: Why Python for AI

Python can contribute to the optimization of many inventory management processes, such as reorder point calculations, safety stock calculations, multi-echelon inventory

optimization, device location tracking, maintenance prediction, demand forecasting, and inventory cost reduction.

Python programs will be used in this research to create forecasting models in Inventory management. The program is project-oriented, and by creating an efficient code, we can have very useful information.

Below it will be presented the steps that need to be followed in order to prepare a model framework. The steps of this model framework are presented below:

3.2 Set up the Program

Set up the Python analysis program and import libraries (Pandas, Numpy, Matplotlib, etc.) The libraries help to analyze and demonstrate the data.

For our analysis was used the Google Colab. Google Colab is a cloud-based Jupyter notebook environment from Google (<https://colab.research.google.com/>). Google's Colab is a free web platform that enables users to run Python and TensorFlow calculations for free, using Google's infrastructure. The service requires no setup to use, providing free access to computer resources without the necessity for the new user to download, install, and configure Python and their libraries to their personal computer.

It is very user-friendly and the biggest advantage is that it requires zero configuration. Except for this, the access to GPUs is free of charge. It provides free access to powerful hardware like GPUs and TPUs, which significantly speeds up tasks like machine learning and deep learning training, and it is very easy to share the code because Colab also integrates with other Google services, such as Google Drive and GitHub, making it easy to share your work.

3.3 Basic Model Creation

The data are inserted in the system as a csv file. The data are formed and prepared in order to be able to be analyzed with the reference machine learning.

When we say to prepare the data in appropriate form, we mean to preprocess it in order to be recognized by the software. That means that the data are inserted into the system through a csv file and initially cleaning by missing values, eliminating any outliers, and preparing data for subsequent analysis.

An outlier is an extreme value within the data set. Specifically, there is an element in the dataset that declines significantly from the rest elements (higher or lower). It is very important in statistics to identify outliers, because they affect very much the statistical analysis and finally the results of the research analysis

It is created the initial-reference model with specific parameters by the exploratory data analysis (EDA). EDA is required for analyzing data using visual techniques. It is used to design trends, patterns, or explore relationships with different variables, which will help with the statistical results.

The set of data will be used as historical data in order to create the initial model. When we create the initial inventory model, this will be the beginning of transfer learning. The model is trained on the prepared dataset (prophet_train) to learn patterns and trends.

Given the data, the next step is to forecast future inventory needs based on past usage patterns. It can be applied many predictive models to estimate future stock requirements.

These reference models incorporate classic and machine learning forecasting approaches:

- ARIMA model: The ARIMA model is considered a traditional time series approach. It was developed in the 1970s by George Box and Gwilym Jenkins. It is a very popular model because it can be attributed to their ability to predict linear relationships in time series data, effectively capturing seasonality and trends (Hyndman & Athanasopoulos, 2018). While this model was limited in its use cases, the variation of the basic model overcame the limitations, like Seasonal ARIMA, Vector ARIMAX, etc.
- Prophet model: Prophet is a machine learning technique in time series forecasting, and it does not require complex approaches in modeling the data.
- Random Forest model: This model makes a combination of predictions for multiple machine learning algorithms. According to Zhang et al., 2022, it more efficient to use a combination of numerous decision trees except for an individual decision tree for determining representative results and more accurate predictions (Zhang et al., 2022).
- Linear Regression Model: This model is used for investigating the regression, and it has been used for forecasting a dependent variable based on one or more independent variables (Uyanık & Güler, 2013). It is the simplest form of forecasting model and the logic of its usage is that it assumes a linear relationship between input (predictors) and output (response).

- XBoost : XGBoost belongs to the group of widely used tree learning algorithms (He et al., 2014). A decision tree allows making predictions on an output variable based on a series of rules arranged in a tree-like structure. It offers scalability, efficiency, and portability (Chen & Guestrin, 2016).
- Decision Tree: This model is very popular for ML methods and provides impressive accuracy in sales forecasting and other applications (Bojer & Meldgaard, 2020) (Makridakis et al., 2022). The rules are determined by the algorithm that is used for growing the tree, taking into account the objectives and limitations that are defined by the user and, as with all data-driven methods, the data set available for training the model.

3.4 Checking the Accuracy of the model

After creating our model, the next step is to check its accuracy. The accuracy measurement can be separated into scale-dependent measures, measures based on percentage errors, scaled errors (Athanasopoulos et al., 2017), and measures based on relative errors (Hyndman & Koehler, 2006).

These metrics can help us to understand how well the applied models are aligned with the real observations. In our research we used the error metrics below:

3.4.1 Mean Absolute Error (MAE)

MAE calculates the average absolute difference between the predicted observations \hat{y}_i and the actual observations y_i . The MAE provides a clear representation of the magnitude of errors without considering their direction.

The closer to zero the MAE is, the better is the forecasting.

The MAE is defined as follows:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (3.1)$$

3.4.2 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is an error metric that was used in our study to forecast accuracy. This metric helps us to see the performance of our models. Big value errors are affecting our models negatively, because represent high errors. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (A_i - F_i)^2} \quad (3.2)$$

Where A_i = actual value

F_i = forecasted value

n = number of observations

It shows the square root of the average of the squared differences between the predicted and actual values.

3.4.3 Mean Square Error (MSE)

One other popular metric in statistics and machine learning is the Mean Squared Error (MSE). It measures the square root of the average discrepancies between the actual values of the dataset and the project value. MSE is used broadly and gives a high view if the predictive models work well.

The MSE calculation formula is:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.3)$$

where:

n = dataset elements

x_i = the actual or observed value for the i -th data point.

y_i = the predicted value for the i -th data point.

3.4.4 R-squared (R^2) Score

R-Squared, or Coefficient of Determination is the most popular regression metric and it is defined as the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It is calculated as:

$$R^2 = 1 - \left(\frac{SSR}{SST} \right) \quad (3.4)$$

Where:

R^2 is the R-Squared.

SSR is the sum of squared residuals between the predicted values and actual values.

SST is the total sum of squares, which measures the total variance in the dependent variable.

The value of R^2 can get values between 0 and 1. The closer to 1 the variable is, the more perfectly the model forecasts the dependent variable using the independent variable(s). A 0 value means the model cannot predict the dependent variable at all using the independent variable(s).

In the models that will be created, cross-validation with metrics MSE, RMSE, MAE, MAPE will be used to measure accuracy.

4. Research Analysis

This chapter will provide the analysis of the furniture company data. Firstly, the dataset variables described, and after that, the data preparation will follow, ensuring its quality and consistency by checking for missing values or outliers and other issues and transforming them as required. Then, follow some descriptive statistics on the basic variables, and explore patterns that can affect the forecasting results.

4.1 Data description

This research utilizes a dataset obtained from Kaggle's e-commerce platform. The creation of a personal account was mandatory for accessing real-time and historical data that reflect actual market conditions and operational realities.

This research uses data from a furniture company. It gives information on daily demand and sales from a data set. It used data from 1 January 2014 up to 31 of August 2017 Data are available at [www.kaggle.com](https://www.kaggle.com/code/zahraaalaatageldein/forecast-furniture-sales/notebook) and specifically in the path:

<https://www.kaggle.com/code/zahraaalaatageldein/forecast-furniture-sales/notebook>.

The CSV file's title is "Sales-for-furniture-store". The CSV file was uploaded, and the sales column was selected as the forecasting target. Sales data consisted of 2121 rows and 20 columns containing detailed information about sales transactions.

Historical demand of all the furniture codes is graphically shown in Figure below:

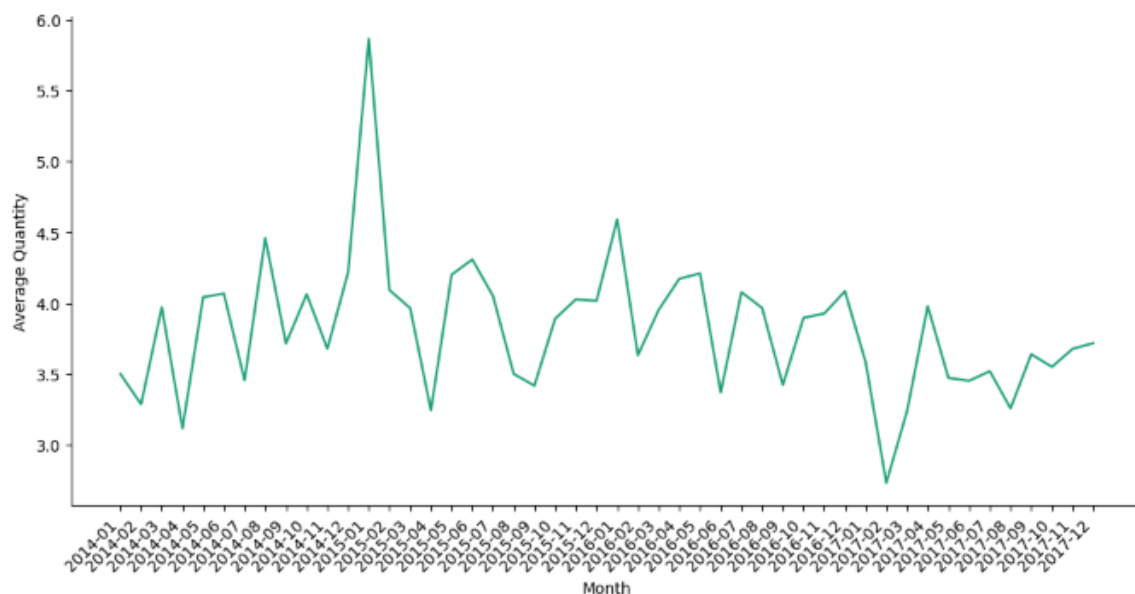


Figure 3: Average demand per month

It is important to mention that the dataset that are examined contain both categorical and numerical variables. Categorical are the variables that consist of text instead of numerical values. These variables, are used to forecast or predict the target variable. The Table below shows the data attributes and type of our dataset, which was used for our research. These include:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2121 entries, 0 to 2120
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Row ID                2121 non-null   int64
1   Order ID              2121 non-null   object
2   Order Date            2121 non-null   object
3   Ship Date             2121 non-null   object
4   Ship Mode             2121 non-null   object
5   Customer ID           2121 non-null   object
6   Customer Name         2121 non-null   object
7   Segment              2121 non-null   object
8   Country               2121 non-null   object
9   City                 2121 non-null   object
10  State                2121 non-null   object
11  Postal Code          2121 non-null   int64
12  Region              2121 non-null   object
13  Product ID           2121 non-null   object
14  Category             2121 non-null   object
15  Sub-Category         2121 non-null   object
16  Product Name         2121 non-null   object
17  Sales                2121 non-null   float64
18  Quantity             2121 non-null   int64
19  Discount             2121 non-null   float64
20  Profit              2121 non-null   float64
```

‘Int64’ type variable is a numerical value

‘Object’ variable is a text value

‘Float64’ is a numerical value with less than the int64 type

‘Raw ID’ is a serial number. ‘Order ID’ is a unique number per order. The ‘Order Date’ column is the date that the order was registered. The ‘Ship Date’ column is the date that the Order was delivered to the customer. The ‘Ship Mode’ column is the method that was used for dispatching the order to the customer. The ‘Customer ID’ column is unique per customer name. The ‘Segment’ column refers to the customer category. The ‘Country’, ‘City’, ‘State’, ‘Postal Code’ and ‘Region’ columns have to do with the address of the customer of the customer and are fixed data for him. Then, the ‘Product ID’ column is a unique per Product Name variable. The ‘Category’ and ‘Sub-Category’ columns refers to the product Category and Sub-category. The ‘Sales’ column is the value of the sales quantity. The ‘Quantity’

column is the quantity of the product sold in each order. The 'Discount' column is the variable that has been taken into account in the product Sales variable and the 'Profit' column is the variable that shows if the products that are sold per order have added value in the company or not.

4.2 Statistical analysis

In this part the research data are going to be explored with various methods and aims to detect possible relationships, associations, or patterns between variables in the dataset. The statistics were executed with the help of the Python language program for both non-graphical (summary statistics) and graphical for the variables that are in the dataset.

We calculate basic statistics for the basic numerical variables in the dataset – *Sales*, *Quantity*, *Discount* and *Profit* and the results are presented in Table 1. We calculate a few major summary statistics to understand all the data variables.

Statistical description of the basic numerical dataset

	Sales	Quantity	Discount	Profit
count	2121.000000	2121.000000	2121.000000	2121.000000
mean	349.834887	3785.007	0.173923	8699.327
std	503.179145	2251.620	0.181547	136.049246
min	1892.000	1000.000	0.000000	-1.862.312400
25%	47.040000	2000.000	0.000000	-12.849000
50%	182.220000	3000.000	0.200000	7.774800
75%	435.168000	5000.000	0.300000	33.726600
max	4416.174000	14000.000	0.700000	1.013.127000

Table 1: Statistical description of the basic numerical values

4.3 Data preprocessing

Data preprocessing is mandatory in order to be sure about the accuracy of the results. The raw data in this dataset will be the training input, which significantly influences the accuracy of the model. We loaded the data from CSV file and organized it in appropriate data columns.

We are going to get rid of redundant columns, we do not need both text and numeric cells with the same information. So we get rid of the columns 'Row ID', 'Category', 'Country', 'Customer ID', 'Product ID'.

'Customer ID' and 'Product ID' give the same information as the 'Customer Name' and 'Product Name'. Also, the 'Category' and 'Country' are variables with unique information for all the observations, and the Row ID is a serial number.

‘Sales’ and ‘Quantity’ are combined in order to create a new variable of 'Price'.

```
df['Price'] = df['Sales']/df['Quantity']*(1-df['Discount'])
```

Next, a new column “**Month**” is created, which will help in monitoring the data per month. The “Month” column is important for creating graphs that will be easy to read and for giving us vital information. Except for the column “Month”, the column “**average Quantity**” is created also, which will be the average quantity sold per Month.

The table that is created is as follows:

	<u>Month</u>	<u>Average Qty</u>
0	2014-01	3.210526
1	2014-02	3.285714
2	2014-03	3.969697
3	2014-04	3.115385
4	2014-05	4.041667

Table 2: Average qty per Month

We rename columns with the use of *rename()* in order to select names that are more suitable to our needs.

Machine learning algorithms are not able to operate on categorical data directly. So, it is required to convert all the input variables to numeric. We reshape the data frame, and we convert the text to unique values so the object variables are converted to ‘int’ variables. This will help us in regression analysis.

Initially the raw data was cleaned by checking for duplicates, processing missing values, renaming columns, and removing outliers.

4.3.1 Check for missing values and outliers

Before starting the research, we need to “clean” our data, which means to make an analysis of missing values and outliers. Python can help us in this direction by providing commands such as *null()* for checking for missing values and *dropna()*, which drops columns of the dataset.

The basic numerical variables, as can be seen from Table 1, are *Quantity*, *Sales*, *Profit*, *Discount*.

“Profit” is affected by the “Sales” of the product. So ‘Profit’ is the dependent variable.

The new column “**Price**” that was created will be considered as the dependent variable, and the independent numerical variables are *Quantity*, *Sales*, and *Discount*.

The first part for identifying outliers is by visualization of the data, which will show the presence of outliers. Then, a statistical method will help us to calculate it, and after that, the outliers will be dropped.

Outliers in the independent variables are identified using boxplots, as shown below:

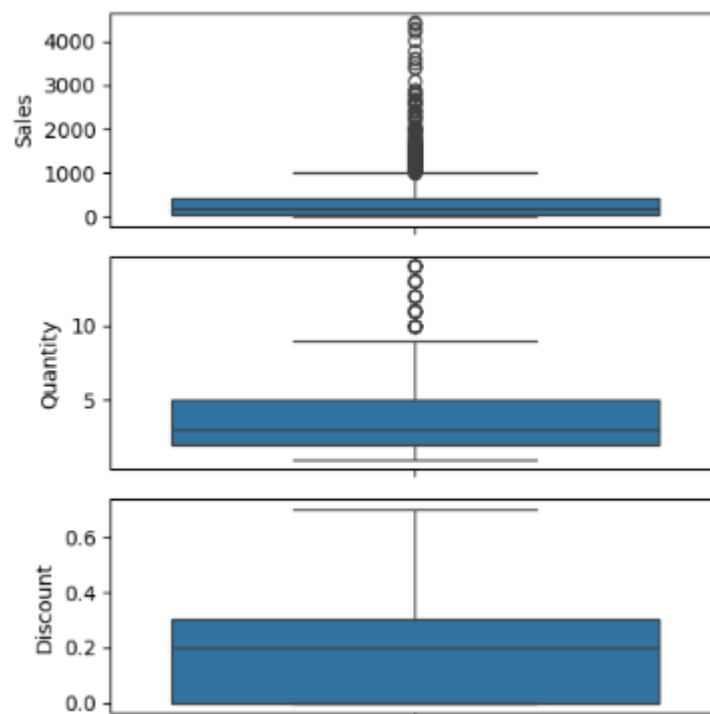


Figure 4: Outliers of the independent numerical variables

It is evident that the ‘*Sales*’ and ‘*Quantity*’ columns contain outliers, whereas ‘*Discount*’ does not. So the next step is to remove these outliers in order to prepare the dataset to be investigated without “noise”.

The outliers that were detected are going to be removed with the Z-scored method (Curtis et al., 2016). Z-score analysis, also known as the standard score, helps measure how far a data point is from the mean.

$$Z\text{-score} = (\text{data_point} - \text{mean}) / \text{std. deviation} \quad (4.1)$$

We determine a limit value for the Z-score, and elements from the dataset with Z-scores beyond this limit are considered potential outliers. We will decide on limit of -3 and +3. Any element with a z-score below -3 or above +3 will be considered an outlier. The reason for this is that 99.7% of the values in a standard normal distribution fall between -3 and +3.

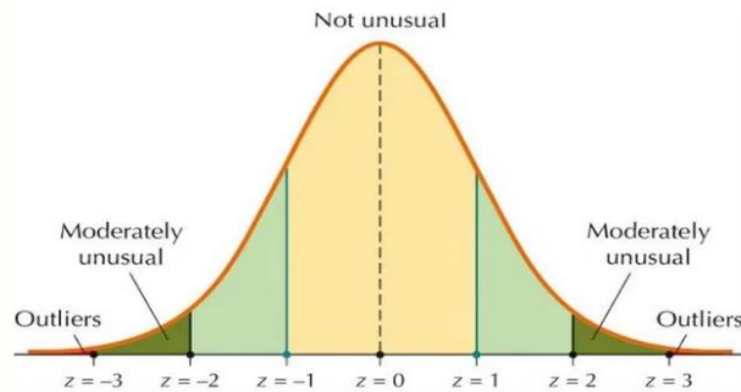


Figure 5: Relationship between Z-scores and centiles, where the parameter is normally distributed

Finally, by the use of the command `stats.zscore()` the outliers are identified, and with `drop()` they are dropped (Appendix B.1 Data Preprocessing). It has created a new data set table with the name **df_clean_zscore** instead of **df**, which is was named up to now.

After dropping the outliers the rows that remain are 2054 as it can be seen below:

	Price	Profit	Discount	Quantity	Sales
count	2054.000000	2054.000000	2054.000000	2054.000000	2054.000000
mean	69.244999	8.819472	0.173558	3.577132	284.897620
std	75.261461	86.935084	0.181122	1.968086	333.062384
min	0.465600	-384.716400	0.000000	1.000000	1.892000
25%	12.640000	-12.196000	0.000000	2.000000	44.128000
50%	45.555200	7.437600	0.200000	3.000000	170.072000
75%	98.643200	30.781800	0.300000	5.000000	386.680000
max	636.508050	412.539400	0.700000	10.000000	1779.900000

Table 3: Statistical data of numerical variables after removing outliers

```
<class 'pandas.core.frame.DataFrame'>
Index: 2054 entries, 0 to 2120
Data columns (total 16 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Price           2054 non-null   float64
1   Profit          2054 non-null   float64
2   Discount        2054 non-null   float64
3   Quantity        2054 non-null   int64
4   Sales           2054 non-null   float64
5   Sub-Category    2054 non-null   int64
```

```

6   Address      2054 non-null   int64
7   Ship_Mode    2054 non-null   int64
8   Order        2054 non-null   int64
9   Customer     2054 non-null   int64
10  Product      2054 non-null   int64
11  Segment      2054 non-null   int64
12  City         2054 non-null   int64
13  Postal_Code  2054 non-null   int64
14  Month        2054 non-null   int64
15  Region       2054 non-null   int64
dtypes: float64(4), int64(12)
memory usage: 272.8 KB

```

The Time Series analysis of the furniture dataset will be focused on two basic variables. As a dependent variable, we will consider the “*average quantity*”, and as an independent variable, we will consider the “Month”. So, we will try to predict the average quantity sold for the period after 12/2017. In Figure 6, the sales quantity trend presented per year separately in order to see if it is a pattern in yearly sales.

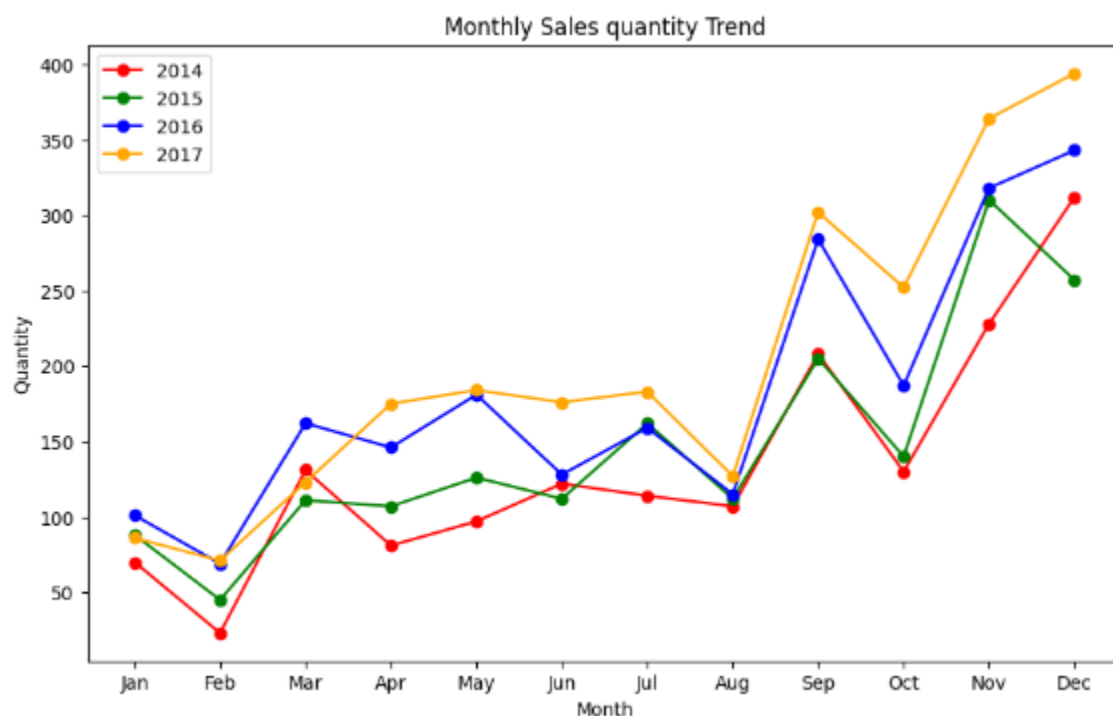


Figure 6: Monthly quantity sold Trend

As we can see from the graph, the years follow a similar trend from 2014 up to 2017, except in some cases, such as the high decrease of quantity sales in December 2015 and the increase of quantity sales in April 2017 compared to the other years.

4.3.2 Decomposition of the Time Series Dataset

One of the most important parts of time series is to monitor the changes that happen during the year. In this part is going to be analyzed the patterns within the data. Through this analysis, the elements of the dataset are captured over time, and this will be helpful for identifying trends, seasonal cycles, and other time and other relationships. The seasonal patterns that will be found will be critical for training our model and in order to make forecasts.

For the Time Series models, the analysis will be based on one variable, “*Average Quantity*”, which was created for analyzing the average quantities of furniture sold per month and monitoring its behavior over the course of time. The average quantity variable will be used as train data which will help to fine-tune the model.

When we speak about decomposition in forecasting, we mean the process of breaking a time series into pieces. Each piece is forecasted, and then the pieces are reassembled (Tessier & Armstrong, 2015). Decomposition allows a researcher to make predictions by using different methods and data for each component. These pieces are data points, and they monitor the chronological sequence of them into its fundamental components: trend, seasonality, and residual (noise). The graphs below, show data of the furniture data set time series and the three components: the trend, the seasonal, and the residual component. The time series of the data ranges from Jan 2014 to Dec 2017, which is all the available data from our source. There is a six-month periodicity.

There are two types of decomposition models: additive and multiplicative. (Prema & Rao, 2015). In the additive model, trend and seasonal variations are constant over time (i.e., they don't increase or decrease as the data grows).

The function of the **additive model** is as follows:

$$\text{Observed} = \text{Trend} + \text{Seasonality} + \text{Residual}$$

In the multiplicative model, the seasonality changes with the trend.

The function of the **multiplicative model** is as follows:

$$\text{Observed} = \text{Trend} \times \text{Seasonality} \times \text{Residual}$$

We select the *multiplicative model* because, as can be seen from Figure 6, there are seasonal fluctuations in changes in size relative to the level of the data. The seasonal effects seem to grow alongside the trend, indicating that the series is likely multiplicative.

The increase of seasonal patterns and their fluctuations is affected by the economy and market. The analysis was done with python program and for the decomposition of the data, we used the *decompose()* command.

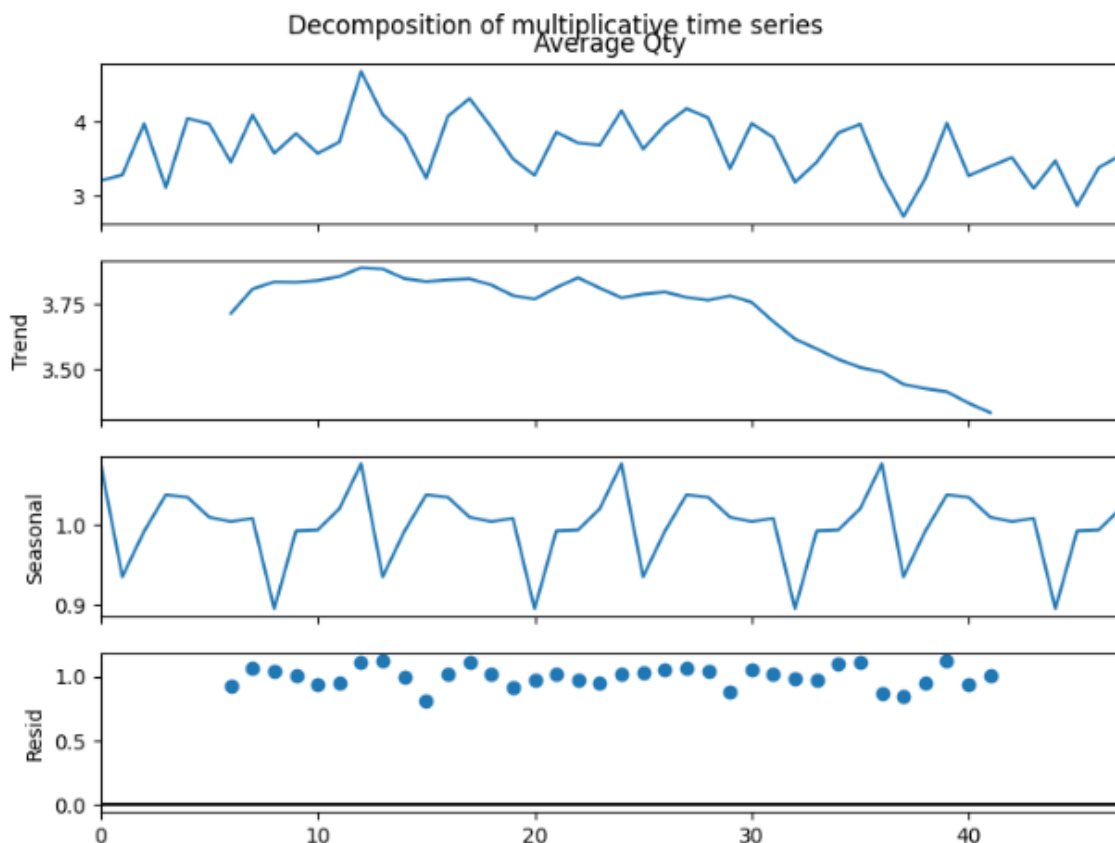


Figure 7: Decomposition of multiplicative time series

This method is very important because it creates patterns that help the analysis. The elements of a time series are messy, and many patterns are hidden in them. The decomposition method helps in creating forecasting models because the researcher can find the trend, the seasonality, and the residual of the time series.

Time series data can be investigated by several periods, such as daily, weekly, monthly, or yearly. For our data set, we insert a new column, 'Month', which will show the average quantity of sales of each month. All the components are according to the average sales quantity per "Month".

The first plot shows the observations of average sales per month for all the years of the monitoring research. The second plot shows the 'Trend' component, which is the long-term movement or direction in the time series data. It reveals the changes of the preferences over time, such as overall growth or decline. The trend of our time series increased over the period July 14-Feb 15, and then consistently decreased. That means that the furniture sales are slowly decreasing from February 2015, which may be a cause of the economic crisis. The third plot shows the seasonality of the dataset, and this component is periodic fluctuations. There are many factors that can affect the seasonality set, such as holidays and weather. The seasonality is a critical factor and if we ignore it, we may conclude in wrong results because we will overestimate or underestimate the demand and the decisions will be weak. From the graph we observe that the seasonal components for the furniture sales are negative during the periods of Feb-Mar and Aug-Sept, with the lowest value occurring in the months of March and October. In July the sales are stable. We observe also that the seasonal components for the quantity sales are positive during the periods of Oct-Feb and March-April and June-July. The pattern is the same every year. The fourth plot is the residual component (in other words, error or noise), which includes any remaining variations in the data that cannot be attributed to the trend or seasonality. It refers to the random or irregular variations that remain unexplained after accounting for the trend and seasonal components. In our graph it can be seen that there are 5 observed fluctuations in Sept-14, Feb-15, Jul-15, Aug-16, and Feb-17.

After the initial preprocessing, the data is split into two separate categories: training data and testing data. The first step of the model training and forecasting is the separation of training and test data. The training data is utilized to train the model, whereas the test data is employed to assess its performance. This separation is crucial for ensuring accurate evaluation of the model. In this part a future dataset is created as a test dataset (with the same length as the initial dataset), and forecasts is generated using the trained model.

After separating the training and the test data, we can continue with the optimization: all the parameters are combined in order to maximize the model performance. The combination is being done with optimization techniques. The accuracy of the model is evaluated by comparing the actual values with forecasted values for the test set.

Finally, we create an optimized model: after training is complete, the optimized model parameters are saved for future use.

5. Modeling with Time Series Methodologies

This chapter presents different time series methodologies for creating forecasting models and the way of selecting the best model that fits best to the research dataset. The data will be explored with ARIMA, SARIMA, and Prophet models, and the error metrics will help us to select the model that fits best to our data. The exploration was executed with the help of the Python programming language.

5.1 ARIMA methodology

Accurate forecasting is important in many aspects of a company, such as money savings, inventory planning, and production (Kumar Dubey et al., 2021). Specifically, the ARIMA model helps in forecasting the customer demand by optimizing production and purchase planning levels to control inventory. For that reason, it will be a very helpful tool for predicting the demand of furniture sales.

The ARIMA model, or else the Autoregressive Integrated Moving Average, was introduced by Box and Jenkins in 1960, and it is used to forecast a single variable (Box & Jenkins, 1976). This statistical model predicts future values based on past periods. ARIMA is one of the most common methods of time series forecasting, which involves predicting future values based on previously observed values.

The ARIMA model is specified by three order parameters: (p, d, q). (Siarni-Namini et al., 2018)

AR(p) Autoregression – a regression model that monitors the dependent relationship between an observation in the present and observations over one previous period.

I(d) Integration – involves differencing of the observations—subtracting each observation from the previous one to transform the time series into a stationary series. This process is repeated *d* times, where differencing consists of subtracting the current value from the value at the previous time step.

MA(q) Moving Average – This variable checks the dependency between an observation and a residual error from a moving average model. The order *q* represents the number of terms that should be included in the model.

These parameters are used in ARIMA models to capture trends, seasonality, and noise in the data, leading to enhanced forecasting accuracy (Peixeiro, 2022).

5.1.1 Tools for running ARIMA model

Before running the ARIMA model, it is important, firstly, to time series data to be stationary. This can be achieved by running the ACF(Autocorrelation Function) and PACF(Partial Autocorrelation Function). These tools will also help us to determine the values of p, d, and q.

Autocorrelation and Partial Autocorrelation

Auto-correlation calculates the level to which a time series is correlated with its past values. Autocorrelation determines the relationship of a series with its lags. If the past values of the series (lags) are significantly autocorrelated, then these past values can improve the forecast of the modern value. Partial autocorrelation gives the real correlation of a series with its lags, but it also gives the connection contribution from the intermediate lags (“Time-Series Models,” 2006).

Autocorrelation plots (ACF) and partial autocorrelation plots (PACF) help identify significant lag values and potential autoregressive or moving average components.

An autocorrelation value near 1 or -1 signifies a strong positive or negative autocorrelation, respectively. On the other hand, an autocorrelation value close to 0, means a weak or no autocorrelation.

By running the autocorrelation and partial autocorrelation code in Python, we conclude with the figures below, which give valuable information for our research analysis. As we referred to above, “*Average Quantity*” will be the variable that will be examined and will be used as training data, and its behavior will be monitored over the course of time.

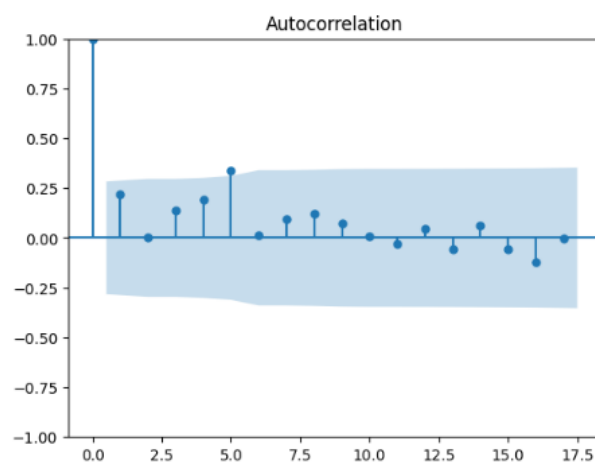


Figure 8: Autocorrelation of furniture sales

The x-axis shows the number of lags, and the y-axis shows the autocorrelation at that number of lags. By default, the plot begins at lag = 0. In lag = 0 the autocorrelation is always equal to 1.

Autocorrelation: 0.22081556788978057

Based on the result above, since the autocorrelation value is close to 0 (0.22), it suggests that there is a weak autocorrelation. The initial observation reveals the presence of positive autocorrelation. A positive autocorrelation indicates that there is a relationship between the current sales quantity, and the previous sales quantities but it is not so strong.

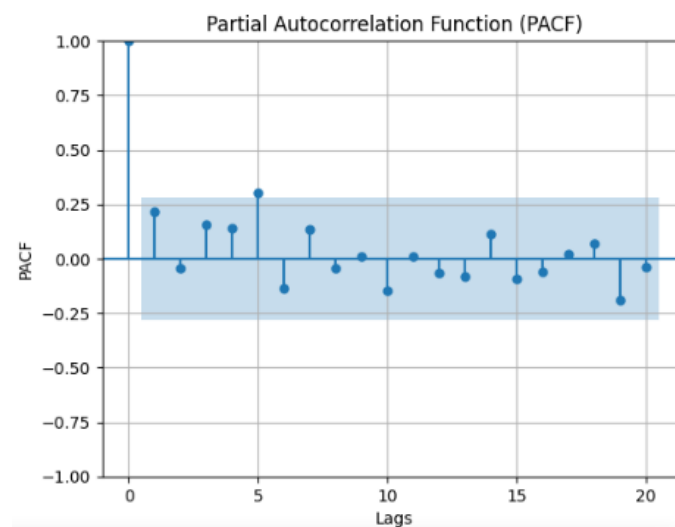


Figure 9: Partially Autocorrelation

Partial Autocorrelation Function (PACF) values:

```
Lag 0: 1.0
Lag 1: 0.22243469616549483
Lag 2: -0.04790808478773663
Lag 3: 0.16731277297608466
Lag 4: 0.15143688419321402
Lag 5: 0.3440897091179477
Lag 6: -0.15230046008683135
Lag 7: 0.16879632947280349
Lag 8: -0.05532999642457382
Lag 9: 0.018651917080794826
Lag 10: -0.19499124644123286
Lag 11: 0.024680846567728
Lag 12: -0.1015682706754399
Lag 13: -0.11503145206403842
Lag 14: 0.16281599733231447
Lag 15: -0.16011214278104366
Lag 16: -0.08277761154009498
Lag 17: 0.03196477422422942
Lag 18: 0.12282650702372983
Lag 19: -0.36754764336755735
Lag 20: -0.05507045398108932
<Figure size 1000x500 with 0 Axes>
```

Every calculation is every lag of the plot and represent the Partial Autocorrelation Function (PACF) values for each lag. The PACF values of each line represent a positive or negative correlation, while values close to zero suggest weaker correlations at that lag.

One step forward in order to produce a more accurate decision is necessary to do statistical testing for stationarity.

There are various statistical tests to check stationarity. The most common are the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

Augmented Dickey-Fuller (ADF) test

The Augmented Dickey-Fuller (ADF) test is a statistical test that is used to determine whether a time series is stationary or non-stationary. The ADF test evaluates the null hypothesis that the time series has a unit root, indicating non-stationarity (Dickey & Fuller, 1979). If the time series has more than one unit root, then it will be stationary.

The ADF test gives us two factors: one is the ADF statistic, and the second is the p-value. The ADF statistic is always a negative number, and the more negative it is, the stronger the evidence against the null hypothesis. The p-value indicates the probability of observing the ADF statistic or a more extreme value, assuming the null hypothesis is true. A low p-value represents strong evidence against the null hypothesis and suggest that the time series is stationary.

The outcomes of the ADF test are summarized in table below:

ADF Statistic:	-0.8624547353173821
p-value:	0.800085115348
Critical Values:	'1%': np.float64(-3.5925042342183704)
	'5%': np.float64(-2.931549768951162)
	'10%': np.float64(-2.60406594375338)

Table 4: Result of ADF Test for average furniture quantity sold per month

The ADF statistic is -0.8624547353173821. This statistic is a negative value and is not more negative than the critical values at significance levels of 1% , 5% and 10%. That means that we do not reject the null hypothesis of a unit root, indicating that the time series **is non-stationary**.

The p-value is 0.800085115348. Typically, if the p-value is below a chosen significance level (e.g., 0.05), it indicates strong evidence to reject the null hypothesis. In this case, the p-value is not below the significance level of 0,1 and that means that there is not strong evidence against the presence of a unit root, indicating that the time series **is non-stationary**.

KPSS Statistics

KPSS Statistic:	0.5974922458816495
p-value:	0.022864341283486404
Critical Values:	'10%': 0.347
	'5%': 0.463
	'2.5%': 0.574
	'1%': 0.739

Table 5: Result of KPSS statistic for average furniture quantity sold per month

The KPSS statistic is 0.597. This statistic quantifies the difference between the observed series and the underlying trend of the series. A smaller KPSS statistic suggests a closer fit to stationarity.

The p-value is 0.0228. This value is in the significance level between 1% and 2,5%, but the KPSS statistics are higher than the limit values in these significant levels. Since there's no strong evidence to rule out a unit root, the time series appears to be **non-stationary**.

At this point we notice that the two tests give conflicting results, so we need to investigate it more. If the series is non-stationary, we need to difference the series to make it stationary. Differencing is the method of calculating the difference between a current observation and the one preceding it. The command that was used for this reason is the *diff()* and its use is the reduction of trends and seasonality and to make the time series more stationary.

Difference Technique

The differencing technique is used to change the non-stationarity of the time series data and to convert it into stationary. Stationary time series are not influenced by the specific time period in which they are observed (Hyndman & Athanasopoulos, 2018). According to the same author, it is important to examine if the data contain trends and seasonality because these affect the value of time series at different times. So they are not stationary. Also from the previous investigation, it was noticed that the time series data were non-stationary. With

the help of the difference method, we will extract the seasonality and trend of the data. Differencing helps stabilize the mean of a time series by removing shifts in its level, thereby reducing or eliminating trends and seasonal patterns. (Kwiatkowski et al., 1992).

This methodology will also help us later in order to run the ARIMA model, which only works with stationary time series data.

The differenced series is the change between consecutive observations in the original series (Hyndman & Athanasopoulos, 2018), and can be written as:

$$y'_t = y_t - y_{t-1} \quad (5.1)$$

y_t = the first observation

The `diff()` function computes the difference between consecutive elements along a specified axis. The `diff()` command will be used, for the calculation of the average quantity difference between successive time points, and the result will be added in a new column called `diff_average_quantity`.

	Month	Average Qty	diff_Average Qty
1	2014-02	3.285714	0.075188
2	2014-03	3.969697	0.683983
3	2014-04	3.115385	-0.854312
4	2014-05	4.041667	0.926282
5	2014-06	3.965517	-0.076149

Figure 10: Average quantity difference between consecutive time points

We are going to run again the autocorrelation Chart which is presented below.

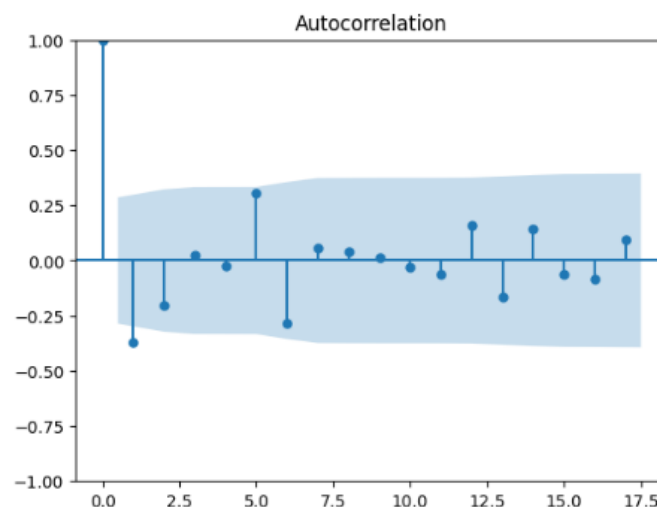


Figure 11: Autocorrelation Chart with the average quantity difference between consecutive time points

Autocorrelation: -0.3689967182042513

As is obvious by using the difference method, we can see that the autocorrelation is improved (value = -0,369).

The same information we get also by running the tests ADF and KPSS.

KPSS Statistic:	0.2249509531135273
p-value:	0.1
Critical Values:	'10%': 0.347
	'5%': 0.463
	'2.5%': 0.574
	'1%': 0.739

Table 6: Result of KPSS statistic for average furniture quantity sold per month

As it can be observed now, the series is stationary, so the trend has been removed. The p-value is 0.1, and the KPSS value is lower than the critical value at the significance level of 10% which means that there is strong evidence against the presence of a unit root, indicating that the time series is **stationary**.

ADF Statistic:	-7.547092897780843
p-value:	3.261778518574035e-11
Critical Values:	'1%': np.float64 (-3.5925042342183704)
	'5%': np.float64 (-2.931549768951162)
	'10%': np.float64 (-2.60406594375338)

Table 7: Result of ADF Test for average furniture quantity sold per month

ADF = -7,547, which is more negative than the critical values of 1%, 5%, and 10%. The same is observed also in p-value, where the number is close to zero and determines **the stationarity** of the time series.

5.1.2 Final running of ARIMA model for our data

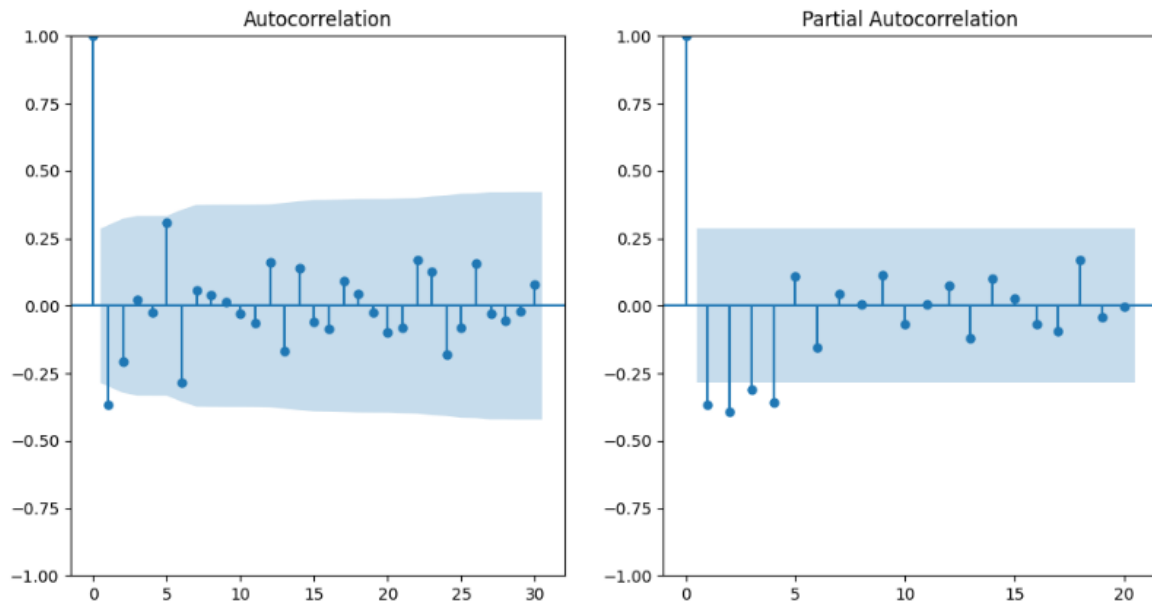


Figure 12: Final Autocorrelation and Partial Autocorrelation Chart with the average quantity difference between consecutive time points

As it has already been referred to, p is the number of AR (Auto-Regressive) terms (lags). In order to select the p number, we notice in the PACF plot to identify the last lag where the PACF value is out of the significance band (displayed by the confidence interval). This happens in lag 4, so we select $p=4$.

As far as the d parameter, it is the number of differences required to make the series stationary. Differencing involves the subtraction of the current values of a series from its previous values d number of times. The trend was removed after running the differencing technique once, so in the ARIMA model the $d=1$.

Finally, for the q parameter, we observe the ACF plot to identify the last lag where the ACF value is out of the significance band (displayed by the confidence interval). So we select $q=1$.

Firstly, a model will be created using ARIMA methodology, which is a classical approach to time series datasets. The ARIMA model, or else Autoregressive Integrated Moving Average, was introduced by Box and Jenkins in 1960, and it is used to forecast a single variable (Box & Jenkins, 1976). This statistical model predicts future values based on previous periods, and it does not take into account seasonality.

These components allow the ARIMA model to account for trends, seasonality, and noise within the data, resulting in an improved forecasting performance (Peixeiro, 2022).

```
p = 4
d = 1
q = 1
```

We run the model and the results are as follows:

```

=====
SARIMAX Results
=====
Dep. Variable:          y      No. Observations:          47
Model:                ARIMA(4, 1, 1)  Log Likelihood        -16.936
Date:                 Wed, 28 May 2025  AIC                  45.872
Time:                 21:33:43      BIC                  56.844
Sample:              0      HQIC                  49.983
                   - 47
Covariance Type:      opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
ar.L1        -1.2676     0.192     -6.617     0.000     -1.643    -0.892
ar.L2        -1.1336     0.231     -4.915     0.000     -1.586    -0.681
ar.L3        -0.8652     0.213     -4.059     0.000     -1.283    -0.447
ar.L4        -0.5978     0.140     -4.273     0.000     -0.872    -0.324
ma.L1         0.5835     0.219      2.663     0.008      0.154     1.013
sigma2        0.1161     0.034      3.380     0.001      0.049     0.183
=====
Ljung-Box (L1) (Q):                0.02  Jarque-Bera (JB):                1.29
Prob(Q):                          0.90  Prob(JB):                  0.53
Heteroskedasticity (H):            0.84  Skew:                   -0.07
Prob(H) (two-sided):              0.75  Kurtosis:                2.19
=====

```

Figure 13: Arima model with parameters (p =4,d=1,q=1)

The AR.L(i) values are as many as the p-values are. The p value=1 so is presented one AR value (AR.L(1))

- AR(L1 up to L4): The coefficients of the time series have a negative relationship with the current value, and this effect is statistically significant (p-value = 0.0000).
- MA(1): The coefficient of 0.58 suggests that past forecast errors have a positive influence on the current value, and this effect is statistically significant (p-value = 0.008).
- σ^2 : This means that the difference between the actual and predicted values has a variance of 0.1161. A small σ^2 represent a better model performance, so the value of 0,1161 means a good model performance.

The error metrics are as below:

Mean Absolute Error (MAE): 0.2337
Mean Squared Error (MSE): 0.0888
Root Mean Squared Error (RMSE): 0.2979

In the ARIMA model, the R^2 is not important because it is used for regression models. The model that we examine in this case is time series, so for that reason, it is meaningless to evaluate it.

As far as the other error metrics, the lower RMSE is, the better the ARIMA model is indicated, which means smaller differences between actual and predicted values. That is, the RMSE of 0.2979 signifies that, on average, our model's predictions deviate from the actual close sales by 0,516.

The fitting of the model to the data can be seen by the graph below:

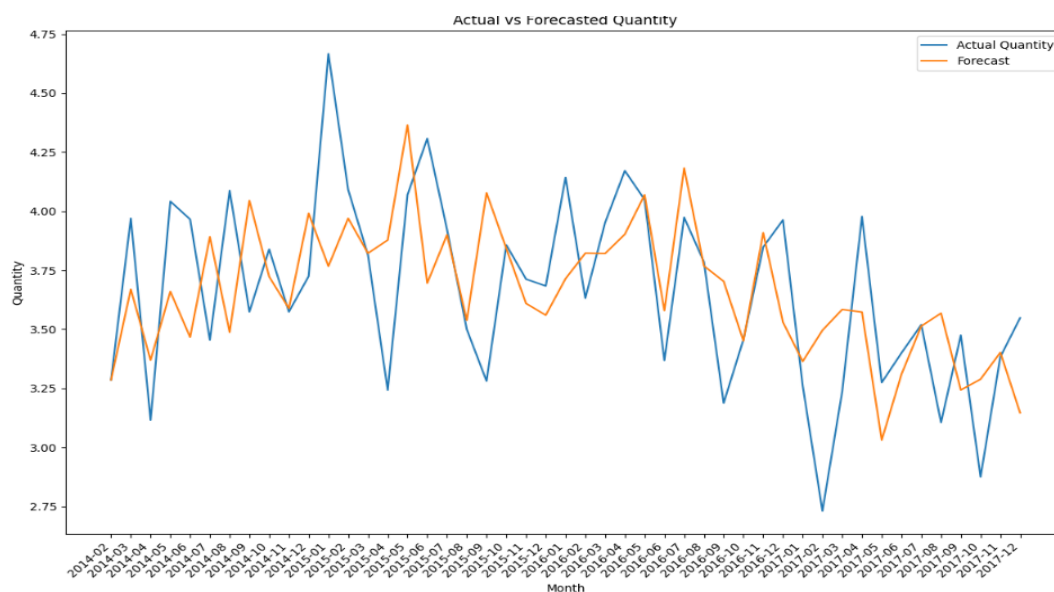


Figure 14: Fitting of Arima model to the data

By generating the prediction in contrast to the dataset, it can be confirmed that the prediction model partially fits on the data.

5.2 SARIMA methodology

The Seasonal Autoregressive Integrated Moving Average (SARIMA), also known as Seasonal ARIMA, is an advanced version of the ARIMA model designed to handle time series data that exhibits seasonal patterns. The SARIMA model includes both ARIMA

parameters (p, d, q) and seasonal terms (P, D, Q, s), where P is the seasonal AR term, D is the seasonal differencing term, and Q is the seasonal moving average term.

The SARIMA model will help capture seasonal effects in the sales data of the research time series, and to improve the overall performance of our model.

In order to select the P value, we consider the seasonal autoregressive order. It models how the series depends on past values from the same season. We will make some tests by selecting different P, D and Q values in order to see with which values our model fits better.

Firstly, by looking at the decomposition model, there is a seasonality, so we select D=1. But we have

Also, the data are at the monthly level, so s=12.

Moreover, in order to select P and Q:

If ACF is positive at lag 12 then $P \geq 1$.

If ACF is negative at lag 12 then $Q \geq 1$.

From Figure 15, it is visible that lag 12 is positive, so it is selected $P=1$ and $Q=0$.

So, by running the SARIMA model with $P,D,Q = 1,1,0$ the results are the below:

```
=====
Dep. Variable:          Average Qty      No. Observations:          36
Model:                SARIMAX(4, 1, 1)x(1, 1, [], 12)  Log Likelihood            3.653
Date:                  Thu, 29 May 2025              AIC                    6.695
Time:                  19:34:54                      BIC                    6.316
Sample:                01-31-2014                    HQIC                   2.015
                   - 12-31-2016
Covariance Type:      opg
=====
              coef  std err      z    P>|z|    [0.025    0.975]
-----
ar.L1          0.1490   1090.018    0.000    1.000   -2136.247   2136.545
ar.L2         -1.1948    550.283   -0.002    0.998   -1079.730   1077.340
ar.L3          0.1808    949.427    0.000    1.000   -1860.662   1861.024
ar.L4         -0.2709    551.388   -0.000    1.000   -1080.971   1080.429
ma.L1         -0.5172   1295.816   -0.000    1.000   -2540.269   2539.235
ar.S.L12       0.5631    117.069    0.005    0.996   -228.887    230.013
sigma2         0.0208     0.755    0.028    0.978    -1.459     1.500
=====
Ljung-Box (L1) (Q):          0.67  Jarque-Bera (JB):          0.30
Prob(Q):                    0.41  Prob(JB):              0.86
Heteroskedasticity (H):      1.74  Skew:                  0.09
Prob(H) (two-sided):         0.73  Kurtosis:              2.00
=====
```

Figure 15: Results of 1st seasonal SARIMA model (4,1,1)(1,1,0,12)

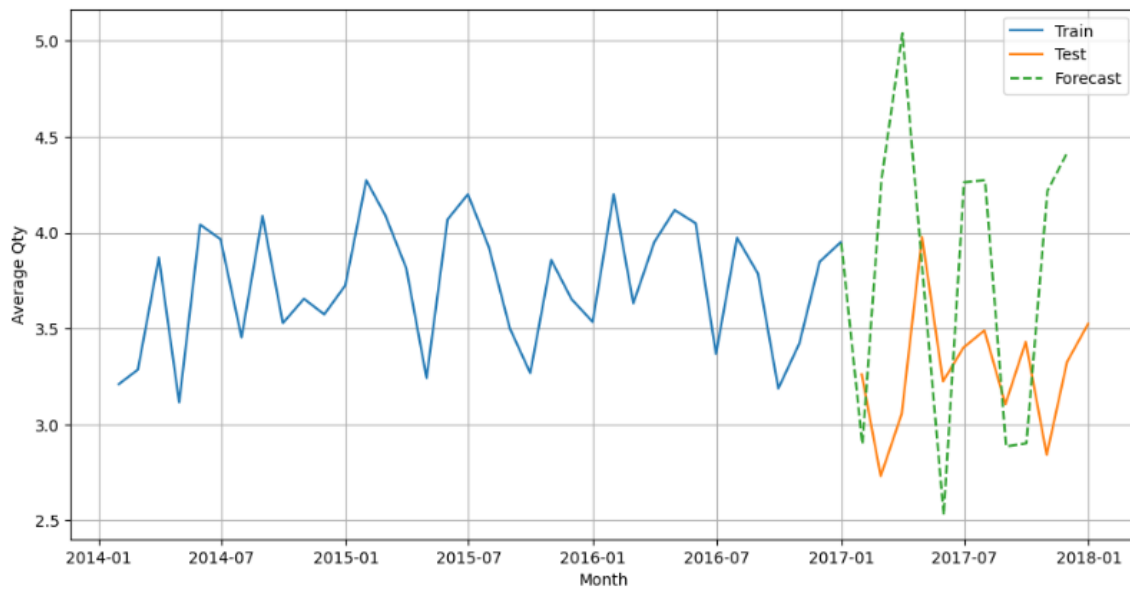


Figure 16: Forecasts on Train and test data from SARIMA(4,1,1)(1,1,0,12)



Figure 17: Forecasting data for the next 3 years with SARIMA model(4,1,1)(1,1,0,12)

We will continue by making a comparison by selecting different P,D,Q parameters in order to see in which the model fits better. We will add in the comparisons also the AIC (Akaike Information Criterion) indicator. This indicator helps us to compare the parameters of the dataset to find the best **fit with the least complexity**.

The results are presented below:

MODEL	ORDER	SEASONAL ORDER	AIC	RMSE	MAE	MSE
ARIMA	4,1,1	-	45.872	0.30	0.23	0.09
SARIMA	4,1,1	0,1,0	21.335	0.55	0.30	0.44
SARIMA	4,1,1	1,1,0	6.695	0.53	0.43	0.28
SARIMA	4,1,1	1,1,1	8.45	0.49	0.42	0.24
SARIMA	4,1,1	1,0,1	17.076	0.35	0.30	0.12

Table 8: Overall Results of Time Series Analysis

From the results, the **ARIMA (4,1,1)** model demonstrated the best overall performance in terms of forecasting accuracy, with the lowest RMSE (0.30), MAE (0.23), and MSE (0.09). While some SARIMA models yielded lower AIC values—suggesting better model fit from a statistical complexity perspective—their forecasting errors were consistently higher.

This suggests that seasonal components did not significantly improve model performance for this particular dataset, likely due to the absence of strong seasonality patterns in the historical data. As a result, the non-seasonal ARIMA model was found to be the most suitable time series forecasting approach for the research data.

Moving deeper into machine learning, we will continue with deeper analysis of AI methodologies in time series analysis using methods such as Prophet, and it will be investigated how this methodology can provide benefits in forecasting and decision-making.

5.3 Prophet methodology

The Prophet forecasting model was developed by Taylor & Letham, (2018) from Facebook. It is currently the latest published forecasting model. Prophet is a flexible and scalable algorithm that can make predictions using simple parameters, can detect daily, weekly, and yearly seasonality, and takes into account seasonality and holidays.

Prophet works with three main components: trend, seasonality, and holidays:

$$y(t) = g(t) + s(t) + h(t) + \epsilon t \quad (5.2)$$

where

$g(t)$ is the trend function, $s(t)$ is periodic change, $h(t)$ is the impact of holidays, and ϵt is error for changes that are not usual and not accommodated by the model.

The library that will be used in Python for the Prophet forecasting model is *scikits-learn* (*SKlearn*) which includes many tools for machine learning models. Except from this it is going to be used the library Matplotlib for displaying Graphs. It is also very important that we can give information about public holidays or other important events that may affect the demand. The graphs will help us to interpret the predictions. The accuracy of the model again can be found with the metrics of R^2 Score (R^2), mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) that we have already described in Chapter 3.

firstly inserted the library Prophet is inserted, and we set the uncertainty interval to 95%. The time lags of our dataset are per month. So, as it is defined in the code, the desired frequency of the time points is Month (`freq='M'`). So we ask to create 36 future time points for our time series, which means that we are going to have predictions for the next 3 years.

In this step, we generate forecasts for our time series by supplying Prophet with a new DataFrame that includes a '*ds*' column, which contains the dates we want to predict.

In the code below is given the rule to Python to generate 36 date points in the future.

```
future_Dates = my_model.make_future_dataframe(periods=36, freq='M')
future_Dates.head()
```

	ds
0	2014-01-01
1	2014-02-01
2	2014-03-01
3	2014-04-01
4	2014-05-01

The new time points will then be used as input to the predict method of our fitted model.

ds: The date stamp of the predicted sales quantity (specific month as time point)

yhat: The predicted value of our metric

yhat_lower: The lower point of our prediction

yhat_upper: The upper point of our prediction

	ds	yhat	yhat_lower	yhat_upper
0	2014-01-01	3.677371	3.228575	4.103007
1	2014-02-01	3.409670	2.976520	3.826736
2	2014-03-01	3.748532	3.347647	4.166363
3	2014-04-01	3.512682	3.066429	3.926780
4	2014-05-01	3.979809	3.544750	4.413184

Figure 18: Forecasted Values with Prophet Analysis

The plot below was generated by running Prophet and present the observed values of our time series (the black dots), the forecasted values (the blue line), and the uncertainty levels of our forecasts (the blue shaded regions).

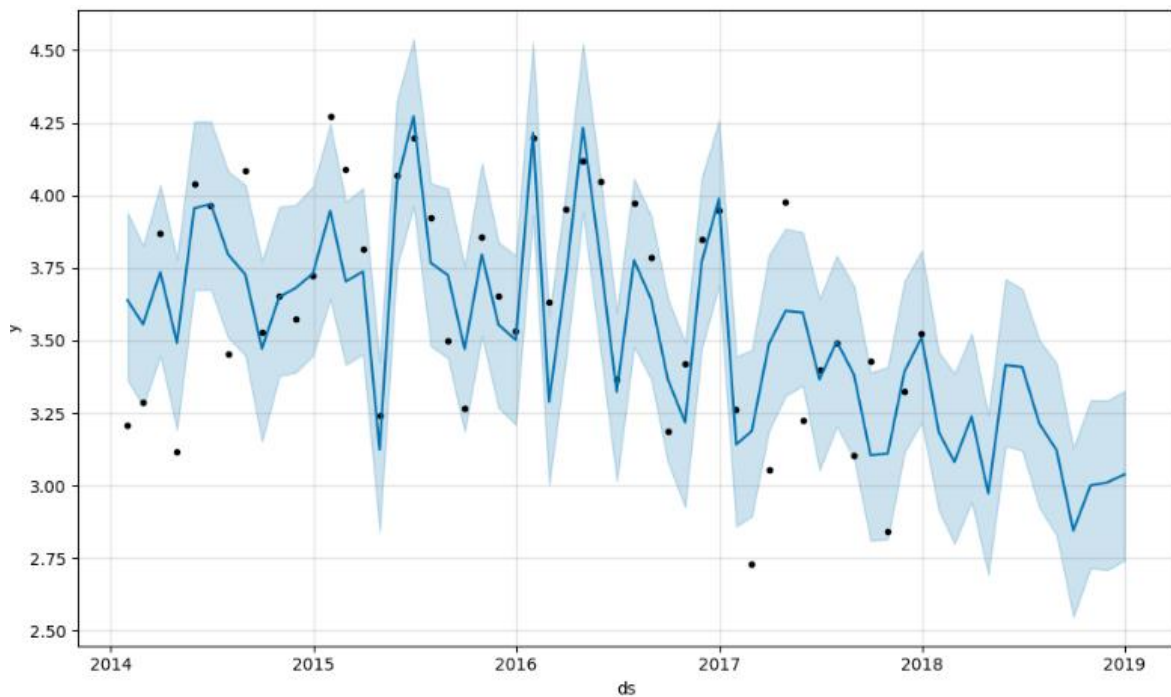


Figure 19: Graph of the Prophet model

As can be seen from the plot, the forecasted data appear from 1/1/2018 up to 31/12/2021. It is observed that the predictions are affected very much by the upper and lowest observations. These observations create a seasonality in the forecasted furniture sales quantities, where the peak is at the end of the year, which means that most people buy furniture at the end of the year and the biggest drop is in January. These upper and lower observations negatively affect our forecasting, which has big fluctuations.

Moreover, the Prophet model has the ability to return the components of our forecasts. This can help reveal patterns of the time series. In Figure 21, the trend and the monthly seasonality of our time series data can be seen.

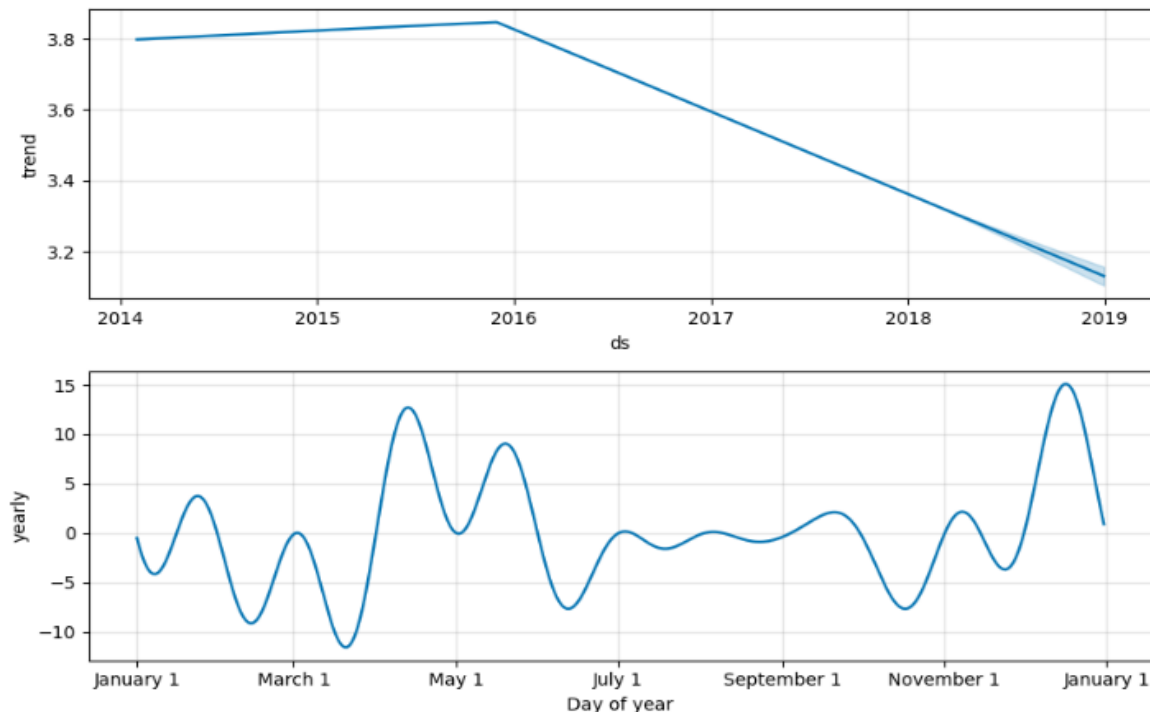


Figure 20: Seasonality based on the Prophet model

The above plot provides interesting insights.

The first plot presents the long-term evolution of the time series. From the plot, it is visible that from 2015 and then, the general sale quantities are reduced, which may be a cause of the economic crisis that affected the purchasing power of the customers. Except for this, it can be seen that from 2018 and then on, when are presented the predictions, the uncertainty is becoming bigger as the time passes.

As far as the seasonality, it can be observed from the outcome that was mentioned above that the upper and lower limits of sales create a seasonal pattern, which shows that customers prefer making purchases close to the end of the year (Christmas period), and after that the sales fall sharply, and in April they increase again until June. Then the sales become stable up to November, where they start making furniture purchases again.

The error metrics of the Prophet model are as follows:

	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Mean Absolute Error (MAE)
PROPHET	3.09	1.76	1.17

Table 9: Prophet model error metrics

6. Modeling with multiple Linear Regression Methodology

This chapter presents different linear regression methodologies for creating forecasting models and the way of selecting the best model that fits best to the research dataset. The data will be explored with machine learning models such as XGBoost, Random Forest etc. and the error metrics will help us to select the model that fits best to our data. The exploration was executed with the help of the Python programming language.

Multiple Linear Regression is a very important statistical technique that is used to investigate the relationship between one independent variable and multiple independent variables (Tabachnick & Fidell, 2019). The basic purpose of its use is to identify the relationship between the variables and to predict dependent variable value, based on independent variable values.

Libraries such as *scikit-learn* and *statsmodels* help in regression analysis. In the analysis below it is going to be described the way of implementing, interpreting, and evaluating our model using Python.

6.1 Check for outliers and missing values

As it has already been referred to in chapter 4.3 (Data preprocessing), before starting the research, we need to “clean” the research data, which means to make an analysis of missing values and outliers. Python can help us in this direction by providing commands such as *null()* for checking for missing values and *dropna()*, which drops columns of the dataset.

For the regression analysis, all the variables are going to be checked for outliers, because all the variables will be taken into account for the model selection. Then, a statistical method will help us to calculate it, and after that, the outliers will be dropped. It is created a new data set table with the name **df_clean** instead of **df**, which was its name up to now. The code for outliers is presented in Appendix B.4 – Regression Analysis.

6.2 Variable Selection

Variable selection is a valuable part of machine learning. It is the process of identifying and selecting the variables that will be more representative in model construction. The target is to improve the model's performance by reducing overfitting, improving accuracy, and reducing training time.

In order to select the best features for our model, it is needed firstly to reduce overfitting, and reduce the number of variances.

By reducing the number of variables, the training time of the model is reduced. Except for this, the trained data will have less ‘noise’, which will affect our predictions.

In the following steps, the relevance of our variables will be evaluated, by using the statistical technique correlation coefficient.

Firstly, the Python library Scikit-Learn, will help with the variable selection. Scikit-Learn provides a complete set of regression metrics for evaluating the performance of regression models.

We select only numerical features for our correlation analysis. Our data have all been transformed to numerical, so all our variables will take part in this correlation. The variable ‘Price’ will be dependent, and it will be correlated with the other independent variables.

The matrix is created, and the correlations are as follows:

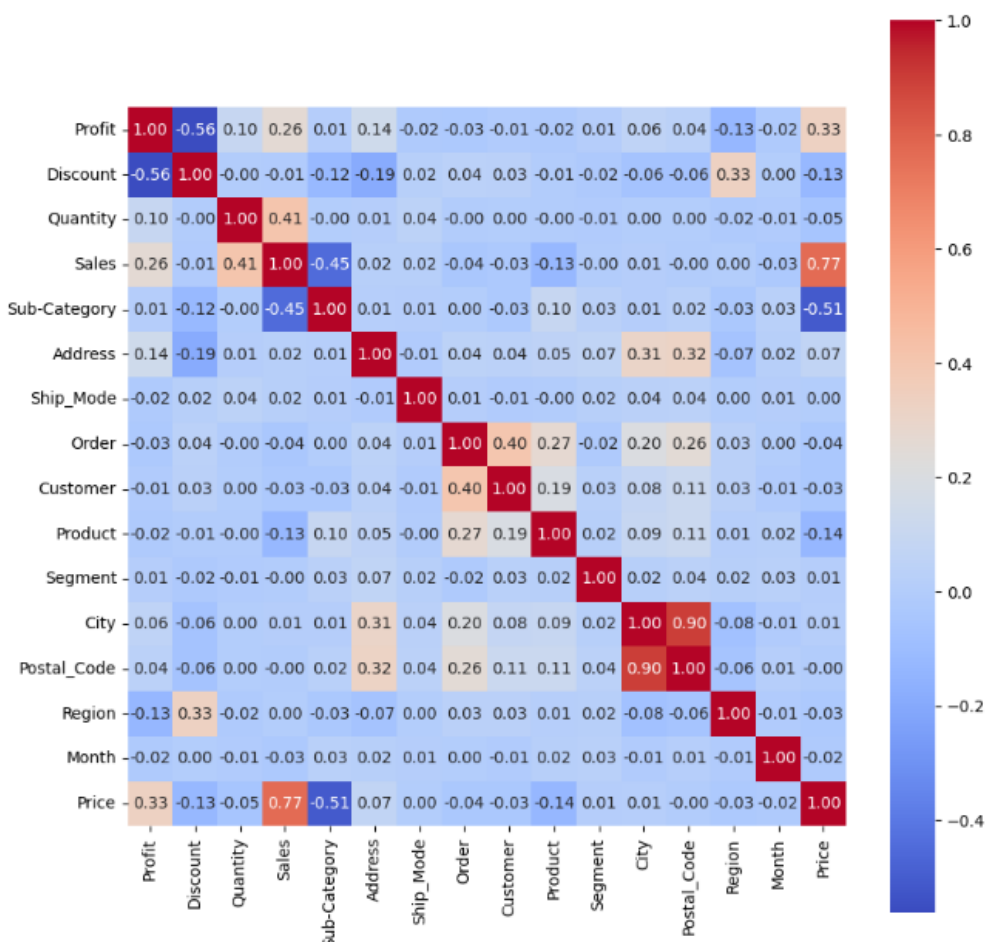


Figure 21: Correlation between Price and other variables

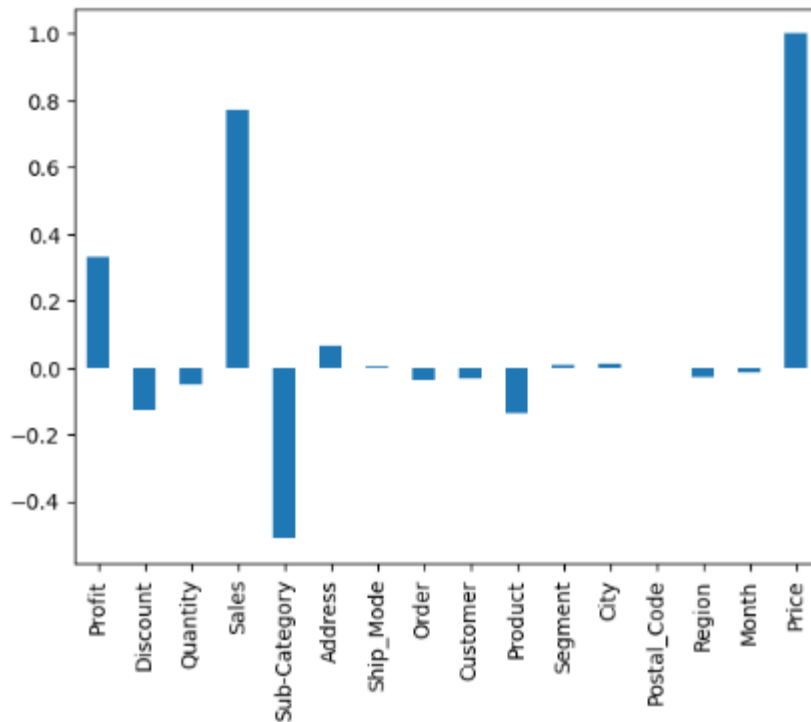


Figure 22: Bar plot of the variances correlation

In this part, it will be removed firstly the highly correlated variables. By removing variances that are not relevant, we improve the model accuracy. We will start by checking the multicollinearity with the VIF factor.

6.3 VIF for multicollinearity

The variance inflation factor (VIF) is a diagnostic tool that measures the amount of multicollinearity in regression analysis (Kim, 2019). So it is calculated the VIF measure and it will be excluded the high VIF values.

	feature	VIF
0	Profit	1.641223
1	Discount	3.211028
2	Quantity	5.257461
3	Sales	2.826810
4	Sub-Category	4.574226
5	Address	2.433083
6	Ship_Mode	2.784611
7	Order	5.217453
8	Customer	4.107258
9	Product	3.841851
10	Segment	1.720243
11	City	9.390183
12	Postal_Code	12.392826
13	Region	3.726498
14	Month	3.013069

Figure 22: Multicollinearity of the variables with VIF

The variables '*Postal_Code*' and '*City*' have a high amount of multicollinearity, so it is excluded from the research. Except for this, '*Quantity*', '*Order*', '*Customer*' and '*Sub-Category*' and '*Product*' have moderate multicollinearity, and it will be investigated further if they will be dropped or not.

By investigating the relationship between the variables, we notice that the variable '*Sales*' has a strong relationship with the variable '*Price*'. The variable '*Order*', has zero correlation with the variables '*Quantity*', '*Sub-Category*' and '*Month*' so it is dropped off. Also, the variable '*Ship_Mode*' has zero correlation with the variables '*Region*', '*Month*' and '*Price*' so it is dropped off too. Except for this, '*Customer*' has zero correlation with '*Quantity*', and the variable '*Quantity*' has zero correlation with '*Sub-Category*' so both '*Quantity*' and '*Customer*' are dropped off. Finally, '*Month*' has zero correlation with '*Discount*', so it is dropped off. Then, the correlation analysis is running again. The updated table is presented below:

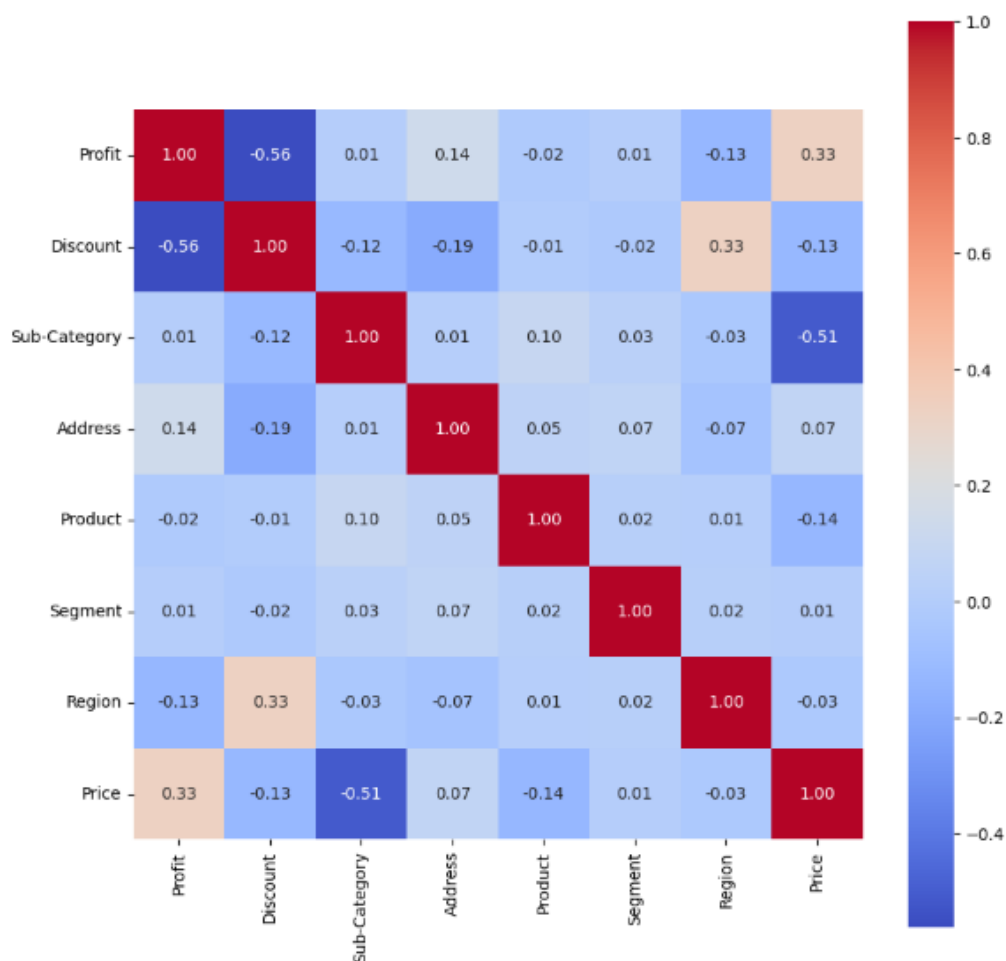


Figure 24: Correlation between Price and other variables after removing variables

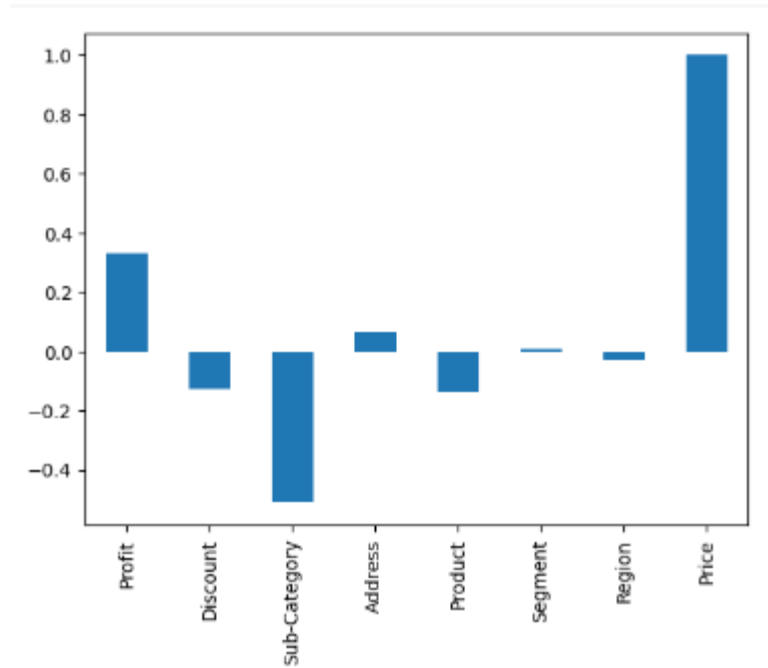


Figure 25: Plot chart of the variables correlation after removing variables

After concluding with the correlations between the variables, the importance of them is checked. With the help of the random forest classifier, the feature importance is determined (Figure 27).

	Feature	Importance
2	Sub-Category	0.383470
0	Profit	0.369166
4	Product	0.121610
1	Discount	0.049422
3	Address	0.040783
5	Segment	0.018599
6	Region	0.016950

Figure 26: Importance of the correlation matrix

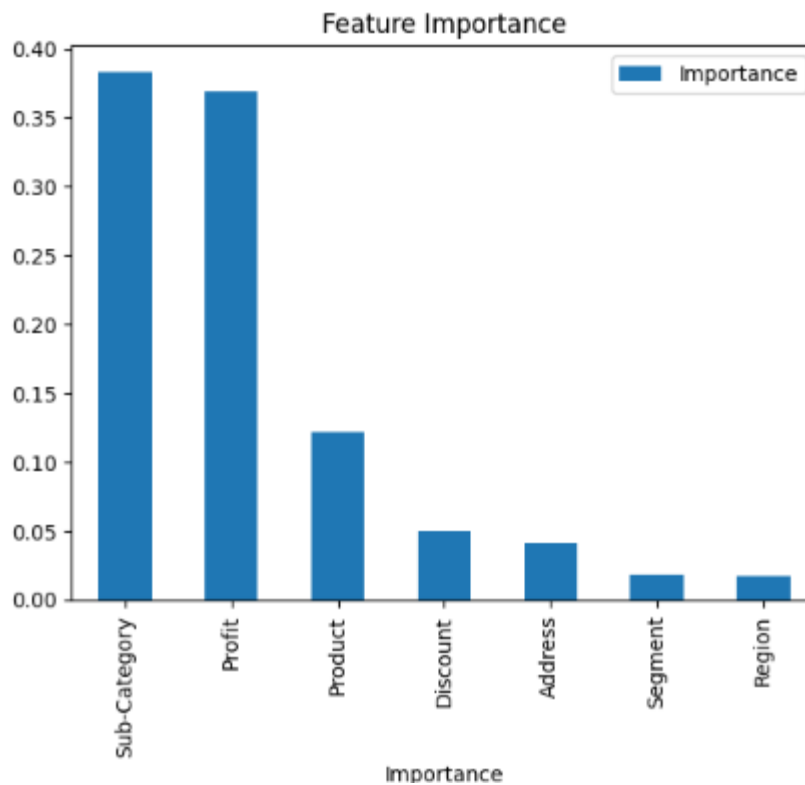


Figure 27: Graph of the Importance of correlation matrix

In this step, is it going to divide the dataset into independent variables (x) and the target dependent variable (y) by removing the 'Price' column from the DataFrame and allocating it to X while assigning the 'Price' column to y.

Following the initial pre-processing step, the dataset was divided into two subsets: a training set used to develop the forecasting models and a test set used to evaluate their predictive performance. The training data will be used to educate the model, while the test data will be used to evaluate its performance. This division is important for the research for the accurate evaluation of the model.

We are also going to calculate the model performance, separating the R^2 in testing and training, so our model selection will be much more representative.

6.4 Error Metrics of Linear Regression

In this step, it going to be evaluated the error metrics with the Linear Regression models (lr = LinearRegression()).

svr = SVR(), DT = DecisionTreeRegressor(), KNR = KNeighborsRegressor(), XGB = XGBRegressor(), RFR = RandomForestRegressor()

```
# Evaluate the model
```

```
evaluate_model(lr, X_train, y_train, X_holdout, y_holdout)
```

LinearRegression

LinearRegression()

Metric	Value
Training R^2 Score	0.3733
Testing R^2 Score	0.4102
Mean Squared Error (MSE)	2435.03
Root Mean Squared Error (RMSE)	49.35
Mean Absolute Error (MAE)	36.69

Table 10: Linear Regression model Error Metrics

SVR

SVR()

Metric	Value
Training R^2 Score	0.1676
Testing R^2 Score	0.1864
Mean Squared Error (MSE)	3358.67
Root Mean Squared Error (RMSE)	57.95
Mean Absolute Error (MAE)	42.47

Table 11: SVR model Error Metrics

DecisionTreeRegressor

DecisionTreeRegressor()

Metric	Value
Training R^2 Score	1
Testing R^2 Score	0.5563
Mean Squared Error (MSE)	1831.7
Root Mean Squared Error (RMSE)	42.8
Mean Absolute Error (MAE)	25.45

Table 12: Decision Tree model Error Metrics

▼ KNeighborsRegressor 1 2
KNeighborsRegressor()

Metric	Value
Training R ² Score	0.5839
Testing R ² Score	0.3522
Mean Squared Error (MSE)	2674.19
Root Mean Squared Error (RMSE)	51.71
Mean Absolute Error (MAE)	36.98

Table 13: KneighborsRegressor model Error Metrics

▼ XGBRegressor 1
XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)

Metric	Value
Training R ² Score	0.9938
Testing R ² Score	0.7999
Mean Squared Error (MSE)	826.09
Root Mean Squared Error (RMSE)	28.74
Mean Absolute Error (MAE)	18.63

Table 14: XGB Regressor model Error Metrics

```

RandomForestRegressor
RandomForestRegressor()

```

Metric	Value
Training R ² Score	0.9599
Testing R ² Score	0.7383
Mean Squared Error (MSE)	1080.22
Root Mean Squared Error (RMSE)	32.87
Mean Absolute Error (MAE)	21.19

Table 15: Random Forest model Error Metrics

7. Discussion on Main Findings and Conclusion

This chapter presents the outcome that arises after the application of the forecasting models to the historical data of the furniture company. In this chapter, we are also going to answer the main research questions raised in the beginning, and we are going to provide a summary of key conclusions.

7.1 Best Model Selection

The Best Model Selection gives an answer to the question “Which model fits best to the data?” that was presented in the introduction. It was used two methodologies for forecasting a. Time Series Forecasting and Multiple Regression Forecasting. The accuracy of all the models was examined with the performance metrics MAE, RMSE, MSE and R^2 for the regression models. The performance metrics provide valuable insights of model performance and the overall forecasting accuracy.

7.1.1 Time Series Results

The time series analysis was based on the variable “Quantity” sold from the furniture company between the years 2014 and 2017. By the investigation of these data, future sales forecasted and the forecasting accuracy was evaluated with the evaluation of performance metrics. Specifically, three Time Series models, was used for this analysis, which was evaluated by three key metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Squared Error (MSE).

MODEL	RMSE	MAE	MSE
ARIMA	0.30	0.23	0.09
SARIMA	0.35	0.30	0.12
PROPHET	1.76	1.17	3.09

Table 16: Error Metrics of Time Series comparison

7.1.2 Multiple Regression model Results

The Multiple Regression Analysis was applied to study the statistical significance of the data set variables. Initial modeling was performed using variables. Firstly, the correlation analysis of the variables was issued, and only the statistical significant factors were kept. Then the statistical performance of each model is checked through statistical measures such as adjusted R^2 and RMSA, MAE, MSE.

From the analysis we concluded at the table below:

Regression Models	Training R^2 Score	Testing R^2 Score	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Mean Absolute Error (MAE)
Linear Regression	0.3733	0.4102	2435.03	49.35	36.69
Decission Tree Regressor	1	0.5563	1831.7	42.8	25.45
RFR= RandomForestRegressor	0.9599	0.7383	1080.22	32.87	21.19
XGB = XGBRegressor	0.9938	0.7999	826.09	28.74	18.63
SVR = Support Vector Regression	0.1676	0.1864	3358.67	57.95	42.47
KNR= KNeighborsRegressor	0.5839	0.3522	2674.19	51.71	36.98

Table 17: Comparison between multivariable regression models

In summary, the *XGB regressor* model performed well in the evaluation. XGB is the best model for predicting demand in the furniture store. It has the highest testing R^2 score (0.9931). Also, the consistent performance between training and testing indicates no overfitting or underfitting. Except for this, the XGB model gives the lowest error rates (MSE = 826.09, RMSE = 28.74, MAE = 18.63). These results fully confirm the effectiveness of the XGB regressor model for forecasting and proactively managing inventory levels.

The predicted sales for the new orders were calculated based on the model that fits better to the data and it is the **XGBRegressor** model. The graph below shows the predicted quantities in comparison with the actual quantities. From this graph it can be seen that for low quantities the predictions are more accurate than for big quantities.

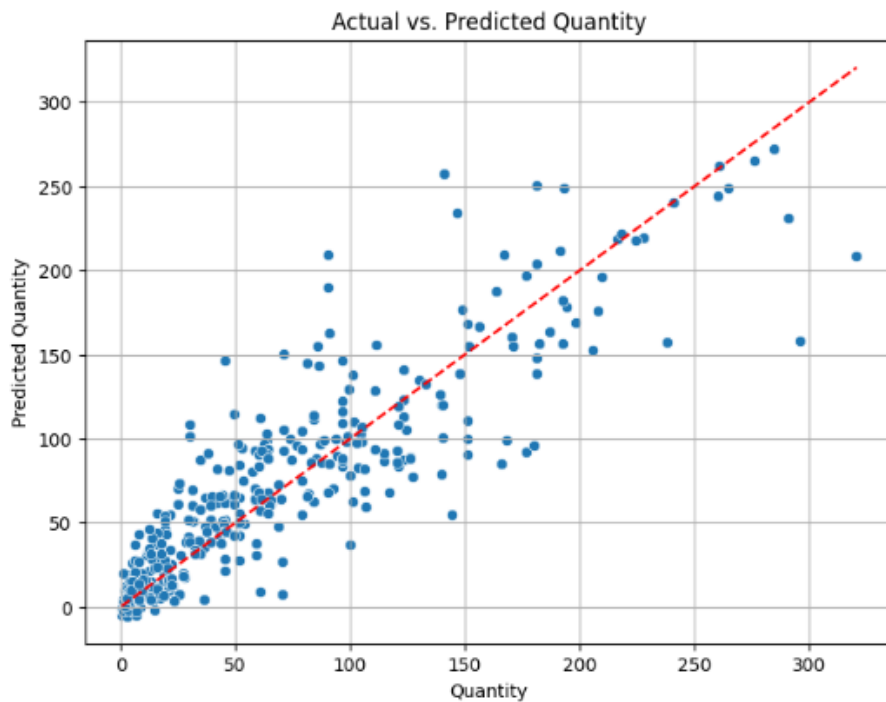


Figure 23: Actual VS Predicted Quantity Sold

7.2 Discussion

The purpose of this research was to examine how AI has affected inventory management, especially in the areas of demand forecasting and inventory optimization. The methodologies that were used, reinforced the value of AI in this field.

A variety of forecasting models were employed in this study, encompassing both classical time series techniques—such as ARIMA, SARIMA, and Prophet—and advanced machine learning algorithms, including Random Forests and XGBoost. These methodologies were applied to forecast product demand for a furniture company, with the objective of enabling faster and more accurate inventory replenishment.

As far as the time series methodologies are concerned, one dependent variable, “Average Quantity” and one independent variable “Month” were used. As far as the machine learning models, multiple variables were used, which were combined in order to find the connection between them and finally to select the best model that fits best to the data. The dependent variable was the “Price” and the correlation with the other independent variables were examined. An important finding from this study is the comparison of the performance of univariate time series models versus multiple regression-based machine learning models in forecasting demand. While machine learning models such as XGBoost and Random Forest

are often preferred for their ability to capture complex, non-linear relationships and interactions among multiple variables, they did not outperform the classical time series models in this specific context. This finding replies on the first question of the study: *“Which AI forecasting models (e.g., ARIMA, Random Forest, XGBoost) provide the most accurate predictions for univariate or multivariate dataset? What role do data quality and availability play in the performance of AI-driven forecasting models?”*

The univariate ARIMA (4,1,1) model had the lowest error values across all key performance metrics (RMSE, MAE, MSE), demonstrating high predictive accuracy. In contrast, the multiple regression models, though capable of leveraging additional explanatory variables, exhibited higher forecasting errors. This discrepancy can be derived from several factors, such as a: the *data characteristics*: The dataset lacked seasonal or multi-variable dependencies, favoring simpler models b: *Model complexity*: Machine learning models may be unable to generalize well due to limited or noisy feature data, c: *Signal strength*: The historical demand patterns in the data were effectively captured by time series structures alone, making additional variables less impactful.

These results suggest that for the dataset used in this research—comprising primarily historical sales data for a furniture company—univariate time series models are more suitable for demand forecasting than more complex regression-based approaches.

The next very important finding, replies on the question: *“What types of AI machine learning algorithms (e.g., decision trees, Random Forests and XGBoost) are most effective for forecasting inventory needs in dynamic markets such as in the environment of the furniture sector?”*.

From the results it was found that the XGBoost model is more accurate and fits better to the data.

One other very important finding replies to the question: *“How do pricing strategies impact sales performance and consumer behavior?”*.

It was found a negative correlation between “discount” and “profit” variable, which means that when the discount increases, the profit decreases. On the other hand, the “price” variable had a positive correlation with the “sales” variable. It is very important to mention that for low prices, the forecasted quantities that are going to be sold are more accurate than the high prices, where the forecasts are not so accurate. So pricing strategies play a great role in the

sales performance, but pricing strategies that have to do with discounts should be applied carefully because they may have a negative impact on the company's profit.

One last critical question of the research was: “*Which patterns are identified in the furniture sales of the research company, and how does this pattern change in different years and months?*”

According to this question, it is very important to mention that by running decomposition analysis, seasonal patterns were revealed, but from the SARIMA and Prophet analysis, it was revealed that there is no seasonal affection on the predicted data. This is a very important analytical insight. The decomposition analysis is purely descriptive (it shows historical patterns in the data). SARIMA and Prophet attempt to model seasonality only if it's strong and consistent enough to improve forecasting accuracy. While decomposition analysis of the time series data revealed observable seasonal patterns across months, both SARIMA and Prophet models did not identify statistically significant seasonality when trained on the dataset. This suggests that although seasonal effects exist historically, they may not be strong, consistent, or regular enough to be effectively captured by these forecasting models. Therefore, incorporating seasonality into the model did not enhance predictive accuracy, indicating a potential mismatch between historical seasonality and future forecasting utility.

Nonetheless, it is important to recognize that integrating AI in the field of inventory management presents several challenges. The greatest challenge is the data quality, which plays a critical role in the effectiveness of AI systems. Since AI algorithms rely heavily on the data they are trained with, inaccurate, incomplete, or inconsistent data can result in misleading forecasts and poor decision-making. Furthermore, the complexity of many AI models often makes them difficult to interpret, posing a barrier to trust and transparency in their outputs.

Equally important are the ethical considerations surrounding the implementation of AI, which must be addressed to ensure responsible and fair use of these technologies. The environment that gained the training data can result in discriminatory outcomes in different environments, particularly in sensitive areas such as procurement and resource allocation. It is important to follow a very sensitive approach that includes strong data validation, the advancement of explainable AI models, and the adoption of ethical frameworks to guide the responsible development and deployment of AI in inventory management.

Despite these challenges, the future of AI in inventory management and, by extension, in supply chain management is highly promising. Ongoing advancements in AI research are expected to enhance existing methodologies and will offer new applications. Furthermore, the evolution of edge computing technologies holds the potential to support real-time decision-making at the operational level, allowing supply chains to respond more rapidly and precisely to disruptions or emerging market opportunities.

As AI technologies continue to evolve and mature, new methodologies are implemented, and their integration in inventory management is expected to become increasingly sophisticated. This advancement enables organizations to more effectively manage complexity and uncertainty within rapidly changing market conditions.

7.3 Conclusion

This research had the purpose of demonstrating different AI methodologies that can be used in companies to improve inventory management. In this thesis, various demand forecasting methodologies were developed and applied for inventory control purposes, utilizing historical sales data and evaluating their performance through forecasting error metrics. The research involved training the models on the initial data of the dataset and testing it with test data, which made the predictions. After the model training and testing process, the results were analyzed and errors were recorded to evaluate the performance of the model.

Many forecasting models were applied, including classical time series models such as ARIMA, SARIMA, and Prophet, as well as advanced machine learning algorithms like Prophet, Random Forests, and XGBoost.

The forecasting methodologies were analyzed in order to predict the quantities that are going to be sold in a furniture company, which will help in replenishing its products quickly and accurately.

Through deep model validation, the study identified the most suitable models for the historical data that was used. The forecasting accuracy of the models was further evaluated and compared using performance metrics such as RMSE, MAPE, and MAE, providing important information about the suitability of the models for demand forecasting of our research data.

As far as the time series models, for the data set that was used, the ARIMA model fits better on the data because as it was revealed from the results, finally there were no seasonality patterns in the forecasted data. As far as the regression models, is observed from the results of the calculated errors that the XGBoost model gives better results compared to conventional forecasting methods. In other words, the XGBoost model is the optimal choice for demand forecasting in the research company.

One of the key findings is that univariate time series models (particularly ARIMA) outperformed more complex regression-based machine learning models in this specific case. This suggests that, contrary to common assumptions, simpler models may still be more effective when the data lacks strong multivariate relationships or when seasonality is weak. Additionally, the decomposition analysis revealed historical seasonal patterns, yet these were not statistically significant enough by SARIMA or Prophet models, highlighting the importance of model-data alignment.

The results also confirmed a moderate negative correlation between discounting and profitability, emphasizing the balance businesses must maintain between pricing strategies and bottom-line performance.

From a theoretical perspective, the research contributes to the growing literature that examines the comparative performance of statistical and AI-based forecasting models in real-world business contexts. Specifically, it is highlighted the need for selecting the most appropriate model, and except for this, the value of AI model interpretation is highlighted, especially in critical areas such as inventory planning and procurement. Finally, the research supports the argument that AI implementation is not limited to automation but also extends to augmenting human decision-making through data-driven insights.

The research findings can also have a practical impact on the inventory managers who enhanced the forecasting accuracy by using AI tools that are supported by qualitative data. The AI benefit is that it has the ability to learn from the raw data, to predict future trends, and to automate processes. By implementing AI in inventory management, companies can act fast with better decisions based on accurate predictions, while they achieve a reduction in the investment costs and maximize customer satisfaction. It is important to mention that companies should not automatically assume that complex machine learning models are superior but rather evaluate models based on performance metrics and interpretability.

Except for this, the identification of trend and seasonal behavior can help the strategic decisions that have to do with product offering and stock levels.

The data that was used for our analysis was from a database source from an internet site, as a result, historical data patterns may not provide an accurate representation of the current or future state of affairs, causing forecasting models to struggle in making reliable predictions.

Accurate sales forecasting helps companies in many sectors. Firstly, it can optimize inventory. Moreover, the forecasting of sales can help avoid overstock or stockouts by predicting demand more accurately. Except for this, the improvement in replenishment rate can improve customer satisfaction. The alignment of production and supply chain with demand gives a boost to the companies and their profitability. The manager who makes decisions based on analysis of existing data and making forecasts achieves the highest performance in all the supply chain.

There are a few key limitations to consider. AI-based models can handle large volumes of datasets and complex patterns, making them superior to traditional forecasting methods. Moreover, it enables real-time forecasting, which offers high benefits in rapidly evolving market environments, where fast decision-making is critical. It also enables real-time forecasting, which is particularly advantageous in fast-paced and dynamically changing markets. The accuracy of the dataset is the most important feature for effective demand forecasting. Deep learning requires a large dataset for accurate predictions. This implies that acquiring such a system requires a significant investment of both time and financial resources. Also, AI success lies on the human, as both rely on each other. So the data quality is affected very much by the people who register data, but also by the people who interpret the results. It is essential to ensure effective alignment between employees and AI systems to maximize their potential. Additionally, missing or incomplete data can significantly compromise the accuracy of the predictions. The data, as it has been referred to, has been retrieved from an open-source site, and we cannot be sure that it is real.

Despite the challenges identified in this study, the future of AI in inventory management remains highly promising. AI offers a proactive, data-driven, and adaptive approach that has the potential to significantly enhance inventory management practices. Through the implementation of AI-powered solutions for predictive maintenance, demand forecasting, and inventory optimization, companies gain efficiency, responsiveness, and customer satisfaction. As AI technologies continue to mature, their integration into inventory

management is expected to become increasingly sophisticated, enabling the creation of more agile, responsive, and sustainable global supply networks.

By acknowledging the limitations of the current research and promoting further exploration in the areas identified, such as model transparency, ethical AI use, and real-time analytics, this study contributes to a deeper understanding of AI's multifaceted impact. It supports the ongoing development of intelligent inventory management that not only drives global business success but also contributes to a more sustainable and efficient global economy.

8. References

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), e00938. <https://doi.org/10.1016/j.heliyon.2018.e00938>
- Acar, A. Z., Yilmaz, B., & Kocaoglu, B. (2014). DEMAND FORECAST, UP-TO-DATE MODELS, AND SUGGESTIONS FOR IMPROVEMENT AN EXAMPLE OF A BUSINESS. *Journal of Global Strategic Management*, 1(8), 26–26. <https://doi.org/10.20460/JGSM.2014815650>
- Addo, A., Centhala, S., & Shanmugam, M. (2020). *Artificial intelligence for risk management* (First edition). Business Expert Press.
- Albayrak Ünal, Ö., Erkeyman, B., & Usanmaz, B. (2023). Applications of Artificial Intelligence in Inventory Management: A Systematic Review of the Literature. *Archives of Computational Methods in Engineering*. <https://doi.org/10.1007/s11831-022-09879-5>
- AlRushood, M. A., Rahbar, F., Selim, S. Z., & Dweiri, F. (2023). Accelerating Use of Drones and Robotics in Post-Pandemic Project Supply Chain. *Drones*, 7(5), 313. <https://doi.org/10.3390/drones7050313>
- Anamu, U. S., Ayodele, O. O., Olorundaisi, E., Babalola, B. J., Odetola, P. I., Ogunmefun, A., Ukoba, K., Jen, T.-C., & Olubambi, P. A. (2023). Fundamental design strategies for advancing the development of high entropy alloys for thermo-mechanical application: A critical review. *Journal of Materials Research and Technology*, 27, 4833–4860. <https://doi.org/10.1016/j.jmrt.2023.11.008>
- Artasanchez, A., & Joshi, P. (2020). *Artificial intelligence with python: Your complete guide to building intelligent apps using python 3.x, second edition* (2nd ed). Packt Publishing.
- Athanasopoulos, G., Hyndman, R. J., Kourentzes, N., & Petropoulos, F. (2017). Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1), 60–74. <https://doi.org/10.1016/j.ejor.2017.02.046>
- Bhatt, C., Kumar, I., Vijayakumar, V., Singh, K. U., & Kumar, A. (2021). The state of the art of deep learning models in medical science and their challenges. *Multimedia Systems*, 27(4), 599–613. <https://doi.org/10.1007/s00530-020-00694-1>

- Blasch, E., Pham, T., Chong, C.-Y., Koch, W., Leung, H., Braines, D., & Abdelzaher, T. (2021). Machine Learning/Artificial Intelligence for Sensor Data Fusion—Opportunities and Challenges. *IEEE Aerospace and Electronic Systems Magazine*, 36(7), 80–93. <https://doi.org/10.1109/MAES.2020.3049030>
- Bojer, C. S., & Meldgaard, J. P. (2020). *Learnings from Kaggle’s Forecasting Competitions*. <https://doi.org/10.13140/RG.2.2.21579.75046>
- Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control* (Rev. ed). Holden-Day.
- Brintrup, A., Pak, J., Ratiney, D., Pearce, T., Wichmann, P., Woodall, P., & McFarlane, D. (2020). Supply chain data analytics for predicting supplier disruptions: A case study in complex asset manufacturing. *International Journal of Production Research*, 58(11), 3330–3341. <https://doi.org/10.1080/00207543.2019.1685705>
- Cerulli, G. (2023). The Basics of Machine Learning. In G. Cerulli, *Fundamentals of Supervised Machine Learning* (pp. 1–17). Springer International Publishing. https://doi.org/10.1007/978-3-031-41337-7_1
- Chalotra, V. (2013). Inventory Management and Small Firms Growth: An Analytical Study in Supply Chain. *Vision: The Journal of Business Perspective*, 17(3), 213–222. <https://doi.org/10.1177/0972262913496726>
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Cherukuri, M. B., & Ghosh, T. (2016). Control Spare Parts Inventory Obsolescence by Predictive Modelling. *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 865–869. <https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.178>
- Curtis, A., Smith, T., Ziganshin, B., & Eleftheriades, J. (2016). The Mystery of the Z-Score. *AORTA*, 04(04), 124–130. <https://doi.org/10.12945/j.aorta.2016.16.014>
- Dhaliwal, N., Tomar, P. K., Joshi, A., Reddy, G. S., Hussein, A., & Alazzam, M. B. (2023). A detailed Analysis of Use of AI in Inventory Management for technically better management. *2023 3rd International Conference on Advance Computing and*

- Innovative Technologies in Engineering (ICACITE)*, 197–201.
<https://doi.org/10.1109/ICACITE57410.2023.10183082>
- Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366), 427. <https://doi.org/10.2307/2286348>
- Dubey, R., Gunasekaran, A., Childe, S. J., Luo, Z., Wamba, S. F., Roubaud, D., & Foropon, C. (2018). Examining the role of big data and predictive analytics on collaborative performance in context to sustainable consumption and production behaviour. *Journal of Cleaner Production*, 196, 1508–1521.
<https://doi.org/10.1016/j.jclepro.2018.06.097>
- Eckert, S. G. (2007). Inventory Management and Its Effects on Customer Satisfaction. *Journal of Business and Public Policy*, 1(3).
- Ghiani, G., Laporte, G., & Musmanno, R. (2013). *Introduction to Logistics Systems Management* (1st ed.). Wiley. <https://doi.org/10.1002/9781118492185>
- Graves, A., Mohamed, A., & Hinton, G. (2013). *Speech Recognition with Deep Recurrent Neural Networks* (No. arXiv:1303.5778). arXiv.
<https://doi.org/10.48550/arXiv.1303.5778>
- Grover, A. K., & Ashraf, M. H. (2024). Leveraging autonomous mobile robots for Industry 4.0 warehouses: A multiple case study analysis. *The International Journal of Logistics Management*, 35(4), 1168–1199. <https://doi.org/10.1108/IJLM-09-2022-0362>
- Gutierrez, R. S., Solis, A. O., & Mukhopadhyay, S. (2008). Lumpy demand forecasting using neural networks. *International Journal of Production Economics*, 111(2), 409–420.
<https://doi.org/10.1016/j.ijpe.2007.01.007>
- He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., & Candela, J. Q. (2014). Practical Lessons from Predicting Clicks on Ads at Facebook. *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 1–9. <https://doi.org/10.1145/2648584.2648589>
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688.
<https://doi.org/10.1016/j.ijforecast.2006.03.001>

- Jat, M. N., & Muyldermans, L. (2013). Time-differentiated service parts distribution: Costs under hierarchical and non-hierarchical setups. *2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)*, 17–24. <https://doi.org/10.1109/CIPLS.2013.6595195>
- Kim, J. H. (2019). Multicollinearity and misleading statistical results. *Korean Journal of Anesthesiology*, 72(6), 558–569. <https://doi.org/10.4097/kja.19087>
- Kumar Dubey, A., Kumar, A., García-Díaz, V., Kumar Sharma, A., & Kanhaiya, K. (2021). Study and analysis of SARIMA and LSTM in forecasting time series data. *Sustainable Energy Technologies and Assessments*, 47, 101474. <https://doi.org/10.1016/j.seta.2021.101474>
- Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54(1–3), 159–178. [https://doi.org/10.1016/0304-4076\(92\)90104-Y](https://doi.org/10.1016/0304-4076(92)90104-Y)
- Lebhar, I., Dadda, A., & Ezzine, L. (2023). Artificial Intelligence Applications in the Global Supply Chain: Benefits and Challenges. In J. Kacprzyk, M. Ezziyani, & V. E. Balas (Eds.), *International Conference on Advanced Intelligent Systems for Sustainable Development* (Vol. 712, pp. 282–295). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-35251-5_27
- Loo, S. K., & Santhiram, R. R. (2018). *Emerging Technologies for Supply Chain Management* (Open Access under the Attribution-ShareAlike 4.0 International (CC- BY-SA 4.0) license; p. 22). WOU Press, Wawasan Open University.
- Magee, J. F., Copacino, W. C., & Rosenfield, D. B. (1985). *Modern logistics management: Integrating marketing, manufacturing, and physical distribution*. Wiley.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364. <https://doi.org/10.1016/j.ijforecast.2021.11.013>
- Makridakis, S., & Wheelwright, S. C. (1977). Forecasting: Issues & Challenges for Marketing Management. *Journal of Marketing*, 41(4), 24–38. <https://doi.org/10.1177/002224297704100403>
- Manaviriyaphap, W. (2024). AI-Driven Optimization Techniques in Warehouse Operations: Inventory, Space, and Workflow Management. . . *AI*.

- Moawad, A., Ebada, A. I., El-Harby, A. A., & Al-Zoghby, A. M. (2024). An Automatic Artificial Intelligence System for Malware Detection. In A. K. Tyagi (Ed.), *Automated Secure Computing for Next-Generation Systems* (1st ed., pp. 115–138). Wiley. <https://doi.org/10.1002/9781394213948.ch6>
- Muhammad, I., & Yan, Z. (2015). Supervised Machine Learning Approaches. *ICTACT Journal on Soft Computing*, 05(03), 946–952. Southwest Jiaotong University, China. <https://doi.org/10.21917/ijsc.2015.0133>
- Nyathani, R. (2023). AI-Driven HR Analytics: Unleashing the Power of HR Data Management. *Journal of Technology and Systems*, 5(2), 15–26. <https://doi.org/10.47941/jts.1513>
- Oluwaseyi, J. A., Onifade, M. K., & Odeyinka, O. F. (2017). Evaluation of the Role of Inventory Management in Logistics Chain of an Organisation. *LOGI – Scientific Journal on Transport and Logistics*, 8(2), 1–11. <https://doi.org/10.1515/logi-2017-0011>
- Oluwatoyin Ajoke Farayola, Adekunle Abiola Abdul, Blessing Otohan Irabor, & Evelyn Chinedu Okeleke. (2023). INNOVATIVE BUSINESS MODELS DRIVEN BY AI TECHNOLOGIES: A REVIEW. *Computer Science & IT Research Journal*, 4(2), 85–110. <https://doi.org/10.51594/csitrj.v4i2.608>
- Özer, Ö. (2011). Inventory Management: Information, Coordination, and Rationality. In K. G. Kempf, P. Keskinocak, & R. Uzsoy (Eds.), *Planning Production and Inventories in the Extended Enterprise* (Vol. 151, pp. 321–365). Springer US. https://doi.org/10.1007/978-1-4419-6485-4_13
- Peixeiro, M. (2022). *Time series forecasting in Python*. Manning Publications Co.
- Plinere, D. S., Borisov, A. N., & Aleksejeva, L. Ya. (2015). Interaction of software agents in the problem of coordinating orders. *Automatic Control and Computer Sciences*, 49(5), 268–276. <https://doi.org/10.3103/S0146411615050089>
- Praveen K B & Bangalore Institute Of Technology. (2020). Inventory Management using Machine Learning. *International Journal of Engineering Research And*, V9(06), IJERTV9IS060661. <https://doi.org/10.17577/IJERTV9IS060661>
- Prema, V., & Rao, K. U. (2015). Time series decomposition model for accurate wind speed forecast. *Renewables: Wind, Water, and Solar*, 2(1), 18. <https://doi.org/10.1186/s40807-015-0018-9>

- Rajaraman, V. (2014). JohnMcCarthy—Father of artificial intelligence. *Resonance*, 19(3), 198–207. <https://doi.org/10.1007/s12045-014-0027-9>
- Robinson, R. N. (n.d.). *Artificial Intelligence: Its Importance, Challenges and Applications in Nigeria*.
- Sanni, O., Adeleke, O., Ukoba, K., Ren, J., & Jen, T.-C. (2024). Prediction of inhibition performance of agro-waste extract in simulated acidizing media via machine learning. *Fuel*, 356, 129527. <https://doi.org/10.1016/j.fuel.2023.129527>
- Schary, P. B. (1984). *Logistics decisions: Text and cases*. Dryden.
- Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–1401. <https://doi.org/10.1109/ICMLA.2018.00227>
- Singh, D., & Verma, A. (2018). Inventory Management in Supply Chain. *Materials Today: Proceedings*, 5(2), 3867–3872. <https://doi.org/10.1016/j.matpr.2017.11.641>
- Singh, N. (2023). AI in Inventory Management: Applications, Challenges, and Opportunities. *International Journal for Research in Applied Science and Engineering Technology*, 11(11), 2049–2053. <https://doi.org/10.22214/ijraset.2023.57010>
- Stanisławski, R., & Szymonik, A. (2021). Impact of Selected Intelligent Systems in Logistics on the Creation of a Sustainable Market Position of Manufacturing Companies in Poland in the Context of Industry 4.0. *Sustainability*, 13(7), 3996. <https://doi.org/10.3390/su13073996>
- Stock, J. R., & Lambert, D. M. (2009). *Strategic logistics management* (4. ed., internat. ed., [Repr.]). McGraw-Hill Irwin.
- Sustrova, T. (2016). A Suitable Artificial Intelligence Model for Inventory Level Optimization. *Trends Economics and Management*, 10(25), 48. <https://doi.org/10.13164/trends.2016.25.48>
- Tabachnick, B. G., & Fidell, L. S. (2019). *Using multivariate statistics* (Seventh edition). Pearson.
- Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), 37–45. <https://doi.org/10.1080/00031305.2017.1380080>

- Tessier, T. H., & Armstrong, J. S. (2015). Decomposition of time-series by level and change. *Journal of Business Research*, 68(8), 1755–1758.
<https://doi.org/10.1016/j.jbusres.2015.03.035>
- Uyanık, G. K., & Güler, N. (2013). A Study on Multiple Linear Regression Analysis. *Procedia - Social and Behavioral Sciences*, 106, 234–240.
<https://doi.org/10.1016/j.sbspro.2013.12.027>
- Vanessa Munoz Macas, C., Andres Espinoza Aguirre, J., Arcentales-Carrion, R., & Pena, M. (2021). Inventory management for retail companies: A literature review and current trends. *2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)*, 71–78.
<https://doi.org/10.1109/ICI2ST51859.2021.00018>
- Yarlagadda, R. T. (2018). *The RPA and AI Automation*. 6(3).
- Zhang, H., Wang, S., Liu, K., Li, X., Li, Z., Zhang, X., & Liu, B. (2022). Downscaling of AMSR-E Soil Moisture over North China Using Random Forest Regression. *ISPRS International Journal of Geo-Information*, 11(2), 101.
<https://doi.org/10.3390/ijgi11020101>
- Zhong, X., & Chen, Z. (2022). Design of Logistics Warehousing System Based on Artificial Intelligence Technology. *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, 711–717.
<https://doi.org/10.1109/ICETCI55101.2022.9832379>

Appendix A: “Predicted Sales quantity for each group of price”

The mean (average) quantity for each group of identical prices for the new Orders are [175.178 3.516 34.165 5.572 171.885 192.524 39.521 36.664 178.954

12.223	71.373	94.566	52.965	48.363	66.142	176.88	19.506	54.052
11.86	75.457	9.947	324.36	31.179	176.854	28.874	152.92	87.88
25.225	86.968	36.789	108.562	81.492	17.89	5.959	102.906	3.339
323.381	43.537	86.6	44.61	174.583	0.17	44.858	183.253	3.884
52.734	94.566	30.549	2.818	7.028	46.13	16.144	37.742	24.613
4.725	108.357	44.002	2.189	108.516	54.658	15.447	219.093	57.671
165.134	65.01	122.199	101.954	309.219	41.979	121.904	5.038	7.729
118.231	78.791	213.579	164.699	1.462	107.584	54.668	102.39	1.481
145.273	20.419	41.364	2.331	57.567	140.076	3.396	85.397	82.237
65.893	79.83	128.175	90.118	53.573	226.937	228.384	63.204	207.123
128.162	25.634	43.332	71.343	7.239	27.933	57.108	86.543	114.303
30.317	42.151	3.175	5.285	205.326	34.137	145.985	60.191	42.911
4.284	5.586	8.733	40.242	62.451	116.047	7.033	-1.29	57.611
44.619	7.172	20.072	1.058	71.87	17.4	30.553	12.904	16.288
105.397	7.59	7.487	57.671	9.262	42.641	163.771	184.863	19.846
9.743	62.579	35.017	7.133	71.414	49.945	135.003	47.986	106.702
60.533	10.072	17.228	3.194	84.409	131.978	115.835	35.093	34.722
68.638	14.679	96.547	5.016	120.845	73.426	102.746	60.533	35.596
60.483	89.877	37.911	36.286	79.062	5.271	2.009	14.427	10.046
36.098	97.983	87.023	169.981	60.947	14.264	6.013	5.813	95.345
2.682	78.928	31.499	57.649	31.846	35.812	15.282	274.817	74.98
312.661	220.372	73.628	5.692	31.845	5.201	139.956	24.16	12.403
22.934	70.471	34.352	93.911	2.309	74.134	151.453	6.013	7.004
318.273	17.376	7.678	5.246	90.161	53.312	50.037	83.131	41.009
129.699	72.866	22.708	2.332	169.246	59.399	23.542	113.983	132.897
15.771	49.695	154.227	27.264	1.024	256.727	83.341	39.418	29.591
136.588	1.397	32.791	12.069	20.031	7.747	4.99	14.637	62.51
1.556	162.768	188.816	43.653	179.719	4.162	8.427	2.892	126.367
2.295	8.516	178.073	320.615	0.32	84.596	4.875	12.515	14.414
4.882	93.396	29.379	34.829	170.582	19.994	3.763	41.953	83.68
1.505	9.849	60.425	127.096	125.695	3.533	7.093	191.809	26.829
20.039	48.725	32.447	68.634	25.794	17.049	4.078	103.911	3.118
3.163	9.563	5.974	263.791	19.87	81.265	43.689	205.756	79.532
5.237	58.504	97.036	103.868	15.57	136.687	116.228	9.943	2.168
3.454	73.538	18.252	189.062	3.31	1.338	46.563	12.236	222.958
79.398	4.871	55.309	128.923	12.378	13.749	199.274	5.85	9.449
6.291	4.893	33.402	102.39	109.243	63.975	61.112	60.219	130.818
142.382	97.418	48.671	9.063	182.753	186.992	64.175	19.228	107.452
16.356	104.291	125.682	37.072	130.466	106.436	10.309	2.725	46.329
20.721	24.44	20.054	81.283	7.457	58.529	6.158	4.974	94.165
88.274	128.104	240.974	132.375	1.231	60.686	35.89	16.142	54.909
5.554	46.445	142.639	284.432	1.777	108.044	19.631	8.427	13.59
158.087	2.655	62.543	7.108	49.246	278.743	7.772	7.17	240.381
64.199	86.477	94.926	2.693	14.003	46.259	102.391	44.765	1.478
6.384	4.986	41.219	59.437	44.717	143.3]		

Appendix B: “Python Code”

B.1 Data Preprocessing

```
# Import Dataset
raw_csv_data =
pd.read_csv('furniture_sales_forecasting.csv',encoding='ISO-8859-1')

# Drop Columns
df.drop(columns=['Row ID', 'Category', 'Country', 'Customer ID',
'Product ID'])

# Check for Duplicates
df.duplicated().sum()

# Create 'Month' column
df['Order_Month'] = pd.to_datetime(df['Order Date'],
format='%m/%d/%Y').dt.to_period('M')
df.info()
df['Ship_Month'] = pd.to_datetime(df['Ship Date'],
format='%m/%d/%Y').dt.to_period('M')
df.info()
if 'Price' not in df.columns:
    df['Price'] = df['Sales'] / df['Quantity'] * (1 - df['Discount'])

#Rename columns
df.rename(columns={"Order ID": "Order", "Customer Name": "Customer",
"Product Name": "Product", "Ship Mode": "Ship_Mode", "Postal Code":
"Postal_Code"}, inplace=True)

df['Order'] = df['Order'].map({k:i for i,k in
enumerate(df['Order'].unique())})
df['Ship_Mode'] = df['Ship_Mode'].map({k:i for i,k in
enumerate(df['Ship_Mode'].unique())})
df['Sub-Category'] = df['Sub-Category'].map({k:i for i,k in
enumerate(df['Sub-Category'].unique())})
df["Address"] = df["Region"] + df["State"]
df['Address'] = df['Address'].map({k:i for i,k in
enumerate(df['Address'].unique())})
df['Customer'] = df['Customer'].map({k:i for i,k in
enumerate(df['Customer'].unique())})
df['Product'] = df['Product'].map({k:i for i,k in
enumerate(df['Product'].unique())})
df['Order_Month'] = df['Order_Month'].map({k:i for i,k in
enumerate(df['Order_Month'].unique())})
df['Segment'] = df['Segment'].map({k:i for i,k in
enumerate(df['Segment'].unique())})
```

```

df['City'] = df['City'].map({k:i for i,k in
enumerate(df['City'].unique())})
df['Postal_Code'] = df['Postal_Code'].map({k:i for i,k in
enumerate(df['Postal_Code'].unique())})
df['Region'] = df['Region'].map({k:i for i,k in
enumerate(df['Region'].unique())})

# Get the existing column names
existing_columns = df.columns.tolist()

# Replace the first column name with 'Sales' and keep the rest
unchanged
new_columns = ['Quantity'] + existing_columns[1:]

# Assign the new column names
df.columns = new_columns

# Now you can select the columns
data_numerical = df[['Price', 'Profit', 'Discount', 'Quantity',
'Sales']]
data_categorical = df[['Sub-Category', 'Address', 'Ship_Mode',
'Order', 'Customer', 'Product', 'Segment', 'City',
'Postal_Code', 'Month', 'Region']]

df = pd.concat([data_numerical, data_categorical], axis=1)

print('Numerical:')
print(data_numerical.head())
print('Categorical:')
print(data_categorical.head())
#First convert the 'date' from object to date time
df = df.copy()
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])

# create new columns 'month' 'year'
df['month'] = df['Order Date'].dt.month
df['year'] = df['Order Date'].dt.year

df = raw_csv_data.copy()

df['Month'] = pd.to_datetime(df['Order Date'],
format='%m/%d/%Y').dt.to_period('M')

average_qty =
df.groupby(['Month'])['Quantity'].mean().reset_index(name='Average
Quantity')

#Check for outliers
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(df['Sales'], ax = axs[0])
plt2 = sns.boxplot(df['Quantity'], ax = axs[1])
plt3 = sns.boxplot(df['Discount'], ax = axs[2])

```

```
plt.tight_layout()
```

```
#Drop outliers
```

```
import pandas as pd
from scipy.stats import zscore
```

```
# Recreate df from raw_csv_data to ensure original column names are present
```

```
df = raw_csv_data.copy()
```

```
# Compute Z-scores for the relevant columns using original column names
```

```
df[['Sales_z_score', 'Quantity_z_score']] = zscore(df[['Sales', 'Quantity']])
```

```
# Keep rows where both Z-scores are within  $\pm 3$ 
```

```
df_clean_zscore = df[
    (df['Sales_z_score'].abs() <= 3) &
    (df['Quantity_z_score'].abs() <= 3)
]
```

```
#drop Z-score columns
```

```
df_clean_zscore = df_clean_zscore.drop(columns=['Sales_z_score', 'Quantity_z_score'])
```

```
print("Data after removing outliers (Z-Score):\n", df_clean_zscore)
```

```
#Check for missing values
```

```
if 'Price' not in df_clean.columns:
    df_clean['Price'] = df_clean['Sales'] / df_clean['Quantity'] * (1 - df_clean['Discount'])
```

```
#Get the categorical columns from the original df, not the df_clean
```

```
df_categorical_clean = df_clean[['Sub-Category', 'Address', 'Ship_Mode', 'Order', 'Customer', 'Product', 'Segment', 'City', 'Postal_Code', 'Region']].reset_index(drop=True)
```

```
num_rows = df_numerical_clean.shape[0] # Change this to df_numerical_clean
```

```
print("Number of rows num: ", num_rows)
```

```
any_missing = df_numerical_clean.isnull().any().any() # Change this to df_numerical_clean
```

```
print("Are there any missing values? ", any_missing)
```



```

cat_rows = df_categorical_clean.shape[0]
print("Number of rows num: ", cat_rows)

any_missing = df_categorical_clean.isnull().any().any()

print("Are there any missing values? ", any_missing)

print('Numerical:')
print(df_numerical_clean.head())
print('Categorical:')

```

```

# @title Month vs Average Sales

from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd # Import pandas

def _plot_series(series, series_name, series_index=0):
    palette = list(sns.palettes.mpl_palette('Dark2'))
    xs = series['Month'].astype(str) # Convert Period objects to
strings
    ys = series['Average Qty']

    plt.plot(xs, ys, label=series_name, color=palette[series_index %
len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = average_quantity.sort_values('Month', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('Month')
_ = plt.ylabel('Average Quantity')
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better
readability

plt.show() # Add plt.show() to display the plot

```

```

# Plotting the decomposed components
plt.figure(figsize=(12, 8))

plt.subplot(4, 1, 1)
# Use the average_sales DataFrame instead of df
plt.plot(average_qty['Average Quantity'], label='Original Data')
plt.legend(loc='upper left')
plt.title('Original Time Series')

#Multiplicative decomposition of the serie
qty_ts = average_qty['Average Quantity']
decomposition = sm.tsa.seasonal_decompose(qty_ts, model='additive',
period=12)

fig = decomposition.plot()
fig.set_figwidth(8)
fig.set_figheight(6)
fig.suptitle('Decomposition of multiplicative time series')
decomposition = sm.tsa.seasonal_decompose(qty_ts,
model='multiplicative', period=12)
plt.show()

#Decomposition of multiplicative time series
fig = decomposition.plot()
fig.set_figwidth(8)
fig.set_figheight(6)
fig.suptitle('Decomposition of multiplicative time series')
plt.show()

```

B.2 Time Series with Arima

```

#Calculate of Autocorrelation
from statsmodels.graphics.tsaplots import plot_acf

qty_series = average_quantity['Average Qty']
autocorr_values = qty_series.autocorr()
print("Autocorrelation:", autocorr_values)

plot_acf(average_quantity['Average Qty'])

#Calculate Partial Autocorrelation
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import pacf # Import the pacf function

qty_series = (average_quantity['Average Qty'])
autocorr_values = qty_series.autocorr()
print("Autocorrelation:", autocorr_values)

```

```

# Assuming 'data' is your time series data (e.g.,
average_qty['Average Quantity'])
data = (average_quantity['Average Qty'])
pacf_values = pacf(data, nlags=20)

# Print PACF values
print("Partial Autocorrelation Function (PACF) values:")
for lag, pacf_val in enumerate(pacf_values):
    print(f"Lag {lag}: {pacf_val}")

# Plot PACF
plt.figure(figsize=(10, 5))
plot_pacf(data, lags=20) # Change lags according to your data
plt.title('Partial Autocorrelation Function (PACF)')
plt.xlabel('Lags')
plt.ylabel('PACF')
plt.grid(True)
plt.show()

```

```

#KPSS calculation stationarity checking
from statsmodels.tsa.stattools import kpss # Import the kpss function

ts = (average_quantity['Average Qty'])
result = kpss(ts)

# Extract and print the test statistic and p-value
kpss_statistic = result[0]
p_value = result[1]
print("KPSS Statistic:", kpss_statistic)
print("p-value:", p_value)

critical_values = result[3]
print(f"Critical Values: {critical_values}")

```

```

#ADF Sattistics - Check for stationarity
def adf_test(series):

    # Perform the ADF test
    result = adfuller(ts)

    # Extract and print the test statistics and p-value
    adf_statistic = result[0]
    p_value = result[1]
    # - Critical values at 1%, 5%, and 10% (result[4])
    print("ADF Statistic:", adf_statistic)
    print("p-value:", p_value)
    print(f"Critical Values: {result[4]}")

adf_test(ts)

```

```

# difference methodology
average_quantity['diff_Average Qty'] = average_quantity['Average
Qty'].diff()

```

```

average_quantity = average_quantity.dropna()

# Display the differenced data
print(average_quantity[['Average Qty', 'diff_Average Qty']])

average_quantity.head()

import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import pandas as pd

# Assuming average_qty is your actual data
data = average_quantity['diff_Average Qty'] # Replace data with
average_qty['diff_Average Quantity']

# Plot ACF and PACF
plt.figure(figsize=(12, 6))

# ACF plot (useful for selecting q)
plt.subplot(121)
plot_acf(data, lags=30, ax=plt.gca()) # Use data and appropriate
lags

# PACF plot (useful for selecting p)
plt.subplot(122)
# Reduce the number of lags to be less than 50% of the sample size
plot_pacf(data, lags=20, ax=plt.gca()) # Use data and appropriate
lags

plt.show()

# Compute again the KPSS- Check for stationarity
from statsmodels.tsa.stattools import kpss # Import the kpss function

ts = average_quantity['diff_Average Qty']
result = kpss(ts)

# Extract and print the test statistic and p-value
kpss_statistic = result[0]
p_value = result[1]
print("KPSS Statistic:", kpss_statistic)
print("p-value:", p_value)

critical_values = result[3]
print(f"Critical Values: {critical_values}")

# ADF Statistics -Check for stationarity
def adf_test(series):

    # Perform the ADF test
    result = adfuller(ts)

    # Extract and print the test statistics and p-value
    adf_statistic = result[0]

```

```

    p_value = result[1]
    # - Critical values at 1%, 5%, and 10% (result[4])
    print("ADF Statistic:", adf_statistic)
    print("p-value:", p_value)
    print(f"Critical Values: {result[4]}")

adf_test(ts)

```

```

# ARIMA Model

```

```

p = 4

```

```

d = 1

```

```

q = 1

```

```

av_qty = average_quantity['Average Qty'].values.astype('float64')

```

```

model = sm.tsa.ARIMA(av_qty, order=(p, d, q))

```

```

result = model.fit()

```

```

# Split data into train / test sets

```

```

train = average_quantity.iloc[:len(average_quantity)-12]

```

```

test = average_quantity.iloc[len(average_quantity)-12:]

```

```

# Make predictions

```

```

start_idx = len(av_qty)

```

```

end_idx = len(av_qty) + len(test) - 1 # Changed 'test_df' to 'test'

```

```

predictions = result.predict(start=start_idx, end=end_idx)

```

```

# Print the predictions

```

```

print(predictions)

```

```

import numpy as np

```

```

from sklearn.metrics import mean_squared_error, mean_absolute_error

```

```

# Split data into train / test sets AFTER differencing

```

```

train = average_quantity.iloc[:len(average_quantity)-12]

```

```

test = average_quantity.iloc[len(average_quantity)-12:]

```

```

# Make predictions (Ensure 'result' and 'av_qty' are still accessible
from previous cells)

```

```

start_idx = len(av_qty) # Or len(train) if av_qty was created from
the full differenced series

```

```

end_idx = len(av_qty) + len(test) - 1

```

```

predictions = result.predict(start=start_idx, end=end_idx)

```

```

# Use the 'diff_Average Quantity' column from the test DataFrame for
actual values

```

```

actual_values = test['Average Qty']

# Ensure predictions and actual values are aligned in length
# This step might not be strictly necessary if start_idx and end_idx
align with test length,
# but it's good for robustness.
actual_values = actual_values[-len(predictions):]
predictions = predictions[:len(actual_values)]

# Evaluation metrics
mae = mean_absolute_error(actual_values, predictions)
mse = mean_squared_error(actual_values, predictions)
rmse = np.sqrt(mse)

# Print results
print("Mean Absolute Error (MAE):", round(mae, 4))
print("Mean Squared Error (MSE):", round(mse, 4))
print("Root Mean Squared Error (RMSE):", round(rmse, 4))

```

```

# Generate forecast and store in 'forecast' column

df_clean_zscore['Month'] = pd.to_datetime(df_clean_zscore['Order
Date'], format='%m/%d/%Y').dt.to_period('M')

start = average_quantity.index[0]
end = average_quantity.index[-1]
average_quantity['forecast'] = result.predict(start=start, end=end,
dynamic=False, typ='levels')

# Plotting
plt.figure(figsize=(12, 8))

# Plot actual quantities
plt.plot(average_quantity['Month'].astype(str),
average_quantity['Average Qty'], label='Actual Quantity')

# Plot forecast (aligned by the same Month column)
plt.plot(average_quantity['Month'].astype(str),
average_quantity['forecast'], label='Forecast')

plt.legend()
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for
better readability
plt.title('Actual vs Forecasted Quantity')
plt.xlabel('Month')
plt.ylabel('Quantity')
plt.tight_layout()
plt.show()

```

```

# SARIMA Model
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX

```

```

# Recreate average_qty with a datetime index from the start
# Assuming your original DataFrame (raw_csv_data) still contains
'Order Date'
# Copy the original dataframe
df_clean_zscore_temp = raw_csv_data.copy()

# Calculate Z-scores for numeric columns
z_scores =
df_clean_zscore_temp.select_dtypes(include=np.number).apply(zscore)

# Define threshold for outliers (e.g., 3 standard deviations)
threshold = 3

# Keep rows where all z-scores are within the threshold (i.e., no
outliers)
df_clean_zscore_temp = df_clean_zscore_temp[(z_scores.abs() <
threshold).all(axis=1)]
df_clean_zscore_temp['Order Date'] =
pd.to_datetime(df_clean_zscore_temp['Order Date'], format='%m/%d/%Y')
average_quantity = df_clean_zscore_temp.groupby(pd.Grouper(key='Order
Date', freq='M'))['Quantity'].mean().reset_index(name='Average Qty')

# Ensure the index is datetime
average_quantity.set_index('Order Date', inplace=True)

# Drop NaN after differencing

# This now operates on a DataFrame with a DatetimeIndex
ts = average_quantity['Average Qty'].dropna()

# Split train and test
# Slicing with index will retain the datetime index
train = ts[:-12]
test = ts[-12:]

# Define SARIMA model parameters
p, d, q = 4, 1, 1
P, D, Q, s = 1, 0, 1, 12 # seasonal order

# Fit SARIMA model
model = sm.tsa.statespace.SARIMAX(
    train,
    order=(p, d, q),
    seasonal_order=(P, D, Q, s),
    enforce_stationarity=False,
    enforce_invertibility=False
)
result = model.fit()

# Print summary
print(result.summary())

# Forecast next 12 months
forecast = result.forecast(steps=12)

# Define the start of the forecast period

```

```

# Now train.index[-1] is a Timestamp, so adding MonthBegin is valid
forecast_start = train.index[-1] + pd.offsets.MonthBegin(1)
forecast_index = pd.date_range(start=forecast_start, periods=12,
freq='MS') # Use 'MS' for Month Start frequency

# Plot
plt.figure(figsize=(12, 6))
plt.plot(train.index, train, label='Train')
plt.plot(test.index, test, label='Test')
plt.plot(forecast_index, forecast, label='Forecast', linestyle='--')
# Assuming 'subcat' is defined somewhere else, otherwise remove or
define it.
# plt.title(f'SARIMA Forecast for {subcat}')
plt.xlabel('Month')
plt.ylabel('Average Qty')
plt.legend()
plt.grid(True)
plt.show()

from statsmodels.tsa.statespace.sarimax import SARIMAX

model = SARIMAX(train,
                 order = (4, 1, 1),
                 seasonal_order =(1, 0, 1, 12))

# Train the model on the full dataset (average_qty['diff_Average
Quantity'] is the full differenced series)
model = SARIMAX(average_quantity['Average Qty'],
                 order=(4, 1, 1),
                 seasonal_order=(1, 0, 1, 12))
result = model.fit()

start_forecast_index = len(average_quantity['Average Qty'])
end_forecast_index = start_forecast_index + (3 * 12) - 1 # Forecast
for 3 years (36 months)

forecast = result.predict(start=start_forecast_index,
                          end=end_forecast_index).rename('Forecast')

# Plot the forecast values
average_quantity['Average Qty'].plot(figsize=(12, 5), legend=True)
forecast.plot(legend=True)

start = len(train)
end = len(train) + len(test) - 1

# Predictions for one-year against the test set
predictions = result.predict(start, end, typ='levels')

# Convert the predictions to a pandas Series with the correct index
predictions = pd.Series(predictions, index=test.index,
name="Predictions")

# plot predictions and actual values
predictions.plot(legend=True)

```



```

average_quantity['Average Qty'].iloc[len(average_quantity)-
12:].plot(legend=True)
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pandas as pd # Ensure pandas is imported

# Drop the first row which will have NaN after differencing
average_quantity_differenced = average_quantity.dropna().copy() # Use
a new variable and .copy()

train =
average_quantity_differenced.iloc[:len(average_quantity_differenced)-
12]
test =
average_quantity_differenced.iloc[len(average_quantity_differenced)-
12:]

start_idx = len(train)
end_idx = len(average_quantity_differenced) -1 # Predict up to the
end of the test set
predictions = result.predict(start=start_idx, end=end_idx)

# Keep rows where all z-scores are within the threshold (i.e., no
outliers)
df_clean_zscore_temp = df_clean_zscore_temp[(z_scores.abs() <
threshold).all(axis=1)].copy() # Added .copy()

y_hat_sarima = pd.DataFrame({'sarima_forecast': predictions},
index=test.index)

# Drop rows with NaN values in either column before calculating RMSE
# This check is still useful if there were NaNs introduced elsewhere
valid_indices = test['Average Qty'].notna() &
y_hat_sarima['sarima_forecast'].notna()
test_valid = test[valid_indices].copy()
y_hat_sarima_valid = y_hat_sarima[valid_indices].copy()

from sklearn.metrics import mean_squared_error, r2_score,
mean_absolute_error
import numpy as np

# Calculate evaluation metrics
mse = format(float(mean_squared_error(test_valid['Average Qty'],
y_hat_sarima_valid['sarima_forecast'])), '.2f')
mae = format(float(mean_absolute_error(test_valid['Average Qty'],
y_hat_sarima_valid['sarima_forecast'])), '.2f')
rmse = np.sqrt(mean_squared_error(test_valid['Average Qty'],
y_hat_sarima_valid['sarima_forecast'])).round(2)

# Ensure test_valid['diff_Average Quantity'] does not contain zero
before calculating MAPE
# Using a small epsilon to avoid division by zero if
test_valid['diff_Average Quantity'] is exactly 0
epsilon = 1e-8

```

```

mape = np.round(np.mean(np.abs(test_valid['Average Qty'] -
y_hat_sarima_valid['sarima_forecast']) / (test_valid['Average
Qty'].abs() + epsilon)) * 100, 2) # Use .abs() here

results = pd.DataFrame({'Method': ['Seasonal autoregressive integrated
moving average (SARIMA) method'], 'RMSE': [rmse], 'MAE': [mae],
'MSE': [mse] })

results = results[['Method', 'RMSE', 'MAE', 'MSE']]
results

```

```

result.plot_diagnostics(figsize=(14,7))
plt.show()

```

B.3 Time Series with Prophet

```

!pip install prophet
# Import the Prophet class from the fbprophet library
from prophet import Prophet

# set the uncertainty interval to 95% (the Prophet default is 80%)
my_model = Prophet(interval_width=0.95)

# Assuming your original DataFrame (raw_csv_data) still contains
'Order Date'
df = raw_csv_data.copy() # Create a copy of the original DataFrame

# Now proceed with your calculations:
df['Month'] = pd.to_datetime(df['Order Date'],
format='%m/%d/%Y').dt.to_period('M')
average_qty =
df.groupby('Month')['Quantity'].mean().reset_index(name='diff_Average
Quantity')
df_prophet = average_qty[['Month', 'diff_Average Quantity']].copy() #
Use average_sales instead of df
df_prophet.rename(columns={'Month': 'ds', 'diff_Average Quantity':
'y'}, inplace=True) # Rename 'Month' to 'ds' and 'Average Quantity'
to 'y'

# Convert 'ds' column to timestamp
df_prophet['ds'] = df_prophet['ds'].dt.to_timestamp() # Convert
Period objects to timestamps
my_model = Prophet(interval_width=0.95) # Create a new Prophet object

# Fit the Prophet model

future_Dates = my_model.make_future_dataframe(periods=36, freq='M')
future_Dates.head()

future_Dates = my_model.make_future_dataframe(periods=36, freq='M')

```

```
future_Dates.head()
```

```
future_Dates = my_model.make_future_dataframe(periods=36, freq='M')
future_Dates.head()
```

```
forecast = my_model.predict(future_Dates)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head()
```

```
my_model.plot(forecast, uncertainty=True)
```

```
my_model.plot_components(forecast)
```

```
# Calculate error metrics

import numpy as np
from prophet.diagnostics import cross_validation, performance_metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Assuming 'pro_change' is your Prophet model object
# and 'df_prophet' is the DataFrame used to train it

# Perform cross-validation using the Prophet model (pro_change)
df_cv = cross_validation(pro_change, initial='730 days', period='180
days', horizon='365 days')

# Calculate performance metrics
df_metrics = performance_metrics(df_cv)

# Calculate MAE, MSE, and RMSE, R2
mae = mean_absolute_error(df_cv['y'], df_cv['yhat'])
mse = mean_squared_error(df_cv['y'], df_cv['yhat'])
rmse = np.sqrt(mse)

print(f'Mean Absolute Error: {mae:.2f}')
print(f'Mean Squared Error: {mse:.2f}')
print(f'Root Mean Squared Error: {rmse:.2f}')
```

B.4 Regression Analysis

```
# Check for outliers
from scipy import stats
import numpy as np

# Select only numerical columns from df for z-score calculation
numerical_cols = df.select_dtypes(include=np.number).columns

data_numerical = df[numerical_cols]
```

```

# Calculate z-scores for the numerical columns
z_scores = stats.zscore(data_numerical)

# Assume that data in each column is normally distributed, outliers
are values with a z-score greater than 3 or less than -3
# Filter outliers based on numerical columns
outliersZ = df[(np.abs(z_scores) > 3).any(axis=1)]

print('Outliers:', outliersZ)

# Remove outliers and keep df_clean as a DataFrame
df_clean = df[(np.abs(z_scores) <= 3).all(axis=1)].copy()

# Calculate z-scores for the cleaned data and store in a new variable
df_scaled_numerical = stats.zscore(df_clean[numerical_cols])

num_rows = df_clean.shape[0]
print("Number of rows num: ", num_rows)

any_missing = np.isnan(df_scaled_numerical).any().any()

print("Are there any missing values? ", any_missing)

cat_rows = df_clean.shape[0]
print("Number of rows num: ", cat_rows)

any_missing = np.isnan(df_scaled_numerical).any().any()

print("Are there any missing values? ", any_missing)

print('clean scaled numerical data (first 5 rows):')
print(df_scaled_numerical[:5])

# Correlation analysis
import seaborn as sns
import matplotlib.pyplot as plt

# Select only numerical features for correlation analysis
numerical_features = df_clean.select_dtypes(include=['number'])

# Calculate the correlation matrix
corr_matrix = numerical_features.corr()

# Get the correlation of 'Price' with other features
# Note: 'Sales' was changed to 'net_profit_margin'
#       as 'Sales' is not in the DataFrame
Sales_corr = corr_matrix['Price']
# Create bar plot
Sales_corr.plot(kind='bar')
plt.show()

# Create a heatmap
plt.figure(figsize=(10,10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True)

plt.show()

```

```

# Find and remove highly correlated features
from statsmodels.stats.outliers_influence import
variance_inflation_factor

df_predictors = df_clean_zscore.drop('Price', axis=1)

# Select only numerical features for VIF calculation
df_predictors_num = df_predictors.select_dtypes(include=['number'])

# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = df_predictors_num.columns

# Calculate VIF for numerical features only
vif_data["VIF"] =
[variance_inflation_factor(df_predictors_num.values, i)
    for i in range(len(df_predictors_num.columns))]

print(vif_data)

```

```

#Postal Code and City have high multicollinearity
df_clean.drop('City', axis=1, inplace=True)
df_clean.drop('Postal_Code', axis=1, inplace=True)

```

```

#Sales have high correlation with Price
df_clean.drop('Sales', axis=1, inplace=True)
#Order has no correlation with Quantity, Sub-Category, Region and
price
df_clean.drop('Order', axis=1, inplace=True)
#Ship Mode has no correlation with Product and Region
df_clean.drop('Ship_Mode', axis=1, inplace=True)
#Customer has no correlation with Quantity
df_clean.drop('Customer', axis=1, inplace=True)

```

```

# Correlation analysis again
# Calculate the correlation matrix

# Select only numerical features for correlation analysis
numerical_features = df_clean.select_dtypes(include=['number'])

# Calculate the correlation matrix using only numerical features
corr_matrix = numerical_features.corr()

# Get the correlation of 'Sales' with other features
Sales_corr = corr_matrix['Price']

# Create bar plot
Sales_corr.plot(kind='bar')
plt.show()

# Create a heatmap
plt.figure(figsize=(10,10))

```

```
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True)

plt.show()
```

```
#Quantity has no correlation with Product
df_clean.drop('Quantity', axis=1, inplace=True)
#Month has no correlation with Discount
df_clean.drop('Month', axis=1, inplace=True)
```

```
df_clean.head()
```

```
# Feature importance
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(random_state=42)

# Use the features DataFrame (df_clean_zscore without 'Price') and
the target variable (df_clean_zscore['Price'])
X_features = df_clean.drop('Price', axis=1)
y_target = df_clean['Price']

# Fit the model using the features and target
rf.fit(X_features, y_target)

importances = rf.feature_importances_

# The feature names should come from the columns of X_features
feature_importances = pd.DataFrame({"Feature": X_features.columns,
"Importance": importances})
feature_importances = feature_importances.sort_values("Importance",
ascending=False)

print(feature_importances)
```

```
from tabulate import tabulate
import numpy as np # Import numpy for square root calculation

def evaluate_model(model, x_train, y_train, x_test, y_test):
    # Calculate metrics
    train_r2 = model.score(x_train, y_train)
    y_pred = model.predict(x_test)
    test_r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    # Calculate RMSE manually
    rmse = np.sqrt(mse)

    mae = mean_absolute_error(y_test, y_pred)

    # Prepare the table data
    metrics = [
```

```

        ["Training R2 Score", f"{train_r2:.4f}"],
        ["Testing R2 Score", f"{test_r2:.4f}"],
        ["Mean Squared Error (MSE)", f"{mse:.2f}"],
        ["Root Mean Squared Error (RMSE)", f"{rmse:.2f}"],
        ["Mean Absolute Error (MAE)", f"{mae:.2f}"],
    ]

    # Print the table
    print(tabulate(metrics, headers=["Metric", "Value"],
tablefmt="grid"))

# Assuming X_train and y_train were created using train_test_split
earlier
lr = LinearRegression()

# Fit the model using X_train (capital X) instead of x_train
lr.fit(X_train, y_train)

# Evaluate the model
#Pass the correct variable names to the evaluate_model function.
evaluate_model(lr, X_train, y_train, X_holdout, y_holdout)

svr = SVR()

# Replace x_train with X_train
svr.fit(X_train, y_train)

evaluate_model(svr, X_train, y_train, X_holdout, y_holdout)

DT = DecisionTreeRegressor()

DT.fit(X_train, y_train)

evaluate_model(DT, X_train, y_train, X_holdout, y_holdout)

KNR = KNeighborsRegressor()

KNR.fit(X_train,y_train)

evaluate_model(KNR, X_train, y_train, X_holdout, y_holdout)

XGB = XGBRegressor()
XGB.fit(X_train,y_train)

evaluate_model(XGB, X_train, y_train, X_holdout, y_holdout)

RFR = RandomForestRegressor()
RFR.fit(X_train,y_train)

evaluate_model(RFR, X_train, y_train, X_holdout, y_holdout)

```

B.5. Summary of Models and Best Model Selection

```
# Use XGBoost since it seems to work the best from the k-fold cross-
validation
xb = XGBRegressor(random_state=42)
xb.fit(X_train, y_train)

# Make prediction on test set
predictions = xb.predict(X_holdout)

print(f'The predicted Prices for the new Orders are {predictions}')
print(f'The actual Prices for the Orders are {y_holdout}')

# Evaluate model
mse = mean_squared_error(y_holdout, predictions)
mae = mean_absolute_error(y_holdout, predictions)
r2 = r2_score(y_holdout, predictions)

print(f'Mean Squared Error: {mse}')
print(f'Mean Absolute Error: {mae}')
print(f'R-squared: {r2}')

# Calculate feature importances
importances = xb.feature_importances_
feature_importances = pd.DataFrame({"Feature": X_train.columns,
"Importance": importances})
feature_importances = feature_importances.sort_values("Importance",
ascending=False)

print(feature_importances)

# Create a bar plot of feature importances
plt.figure(figsize=(10, 8))
feature_importances.sort_values("Importance",
ascending=False).plot(kind='bar', x='Feature', y='Importance')
plt.xlabel('Importance')
plt.title('Feature Importance')
plt.show()

# Select only numerical features for correlation analysis
numerical_features = df.select_dtypes(include=['number'])

# Calculate the correlation matrix
corr_matrix = numerical_features.corr()

# Get the correlation of 'Quantity' with other features
net_profit_corr = corr_matrix['Price']

# Calculate the correlation between each feature and the target
variable
correlations = numerical_features.corr()['Price'].sort_values()
print(correlations)

Price_grouped = df.groupby("Price") ['Quantity'].mean()
print(Price_grouped)
```



```

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_holdout, y=predictions)
plt.plot([y_holdout.min(), y_holdout.max()], [y_holdout.min(),
y_holdout.max()], '--r') # identity line
plt.xlabel('Quantity')
plt.ylabel('Predicted Quantity')
plt.title('Actual vs. Predicted Quantity')
plt.grid(True)
plt.show()

```

Author's Statement:

I hereby expressly declare that, according to the article 8 of Law 1559/1986, this dissertation is solely the product of my personal work, does not infringe any intellectual property, personality and personal data rights of third parties, does not contain works/contributions from third parties for which the permission of the authors/beneficiaries is required, is not the product of partial or total plagiarism, and that the sources used are limited to the literature references alone and meet the rules of scientific citations.