1/24/2024

Operations Management Project

Unveiling the Transformative Role of Artificial Intelligence in Inventory Management: An Integrative Research Analysis

Aditya Nikumbh
77120523529
NMIMS

ACKNOWLEDGEMENT

First of all, I would like to take this opportunity to thank the NMIMS Centre for Distance Learning and Online Education for having a project as a part of the PGDBM Curriculum.

I want to take this opportunity to express my sincere thanks to Director "Dr Shubhasheesh Bhattacharya" for providing me with a helpful environment in the complexity of this project and for giving me a chance to do this project in the esteemed organisation.

I want to acknowledge the experts and professionals who shared their knowledge and expertise, contributing significantly to the project's success.

An acknowledgement would only be complete with thanks to all our friends for their valuable suggestions.

Place: Mumbai STUDENT NAME: Aditya Vinodkumar Nikumbh

Date: 29/03/2024 SAP ID: 77120523529

Contents

List of Figures

Summary

This project delves into the transformative role of Artificial Intelligence (AI) in inventory management. The study encompasses a thorough literature review, impact analysis of AI on inventory management, identification of improvement areas, and formulation of pragmatic recommendations. The primary objective is to provide a comprehensive understanding of the historical context, contemporary developments, and the impact of AI on inventory management, ultimately guiding businesses in leveraging AI to effectively augment their inventory management practices.

The literature review synthesises insights from academic research papers, industry reports, and relevant publications, providing a holistic perspective on the evolution of inventory management techniques and recent advancements in AI technology. It lays the foundation for impact analysis, which seeks to elucidate how AI technologies have influenced inventory management strategies, operational efficiencies, and business performance metrics.

Furthermore, the study identifies specific areas within inventory management where AI implementation has been prominent, focusing on delineating avenues for reducing costs and lead times through AI-driven optimisation. Based on the literature review and data analysis insights, actionable recommendations are formulated to facilitate informed decision-making and drive tangible improvements in inventory management efficiency.

The project addresses the research questions by studying the evolution of different inventory management techniques and technology using literature, analysing the impact of Artificial Intelligence & Machine Learning algorithms in inventory management, and evaluating the effect of varying ML algorithms on back-order prediction through data analysis and literature review. The code for the same is in the appendix.

The study acknowledges certain limitations, such as the availability of limited datasets and the inability to perform hyperparameter tuning using the technology employed. Despite these limitations, the findings and recommendations presented in this project offer valuable insights for businesses seeking to harness the transformative potential of AI in inventory management, providing a roadmap for navigating the complexities of AI-driven inventory optimisation effectively.

1. Introduction

Inventory management is crucial in ensuring businesses' efficiency and profitability across various industries. With the rapid advancement of technology, particularly Artificial Intelligence (AI), there has been growing interest in exploring how AI can revolutionise traditional inventory management practices. This study aims to examine the literature on the evolution of inventory management and investigate the impact of AI on its optimisation.

1.1 Research Objectives:

Literature Review: The foremost objective of this study is to conduct an exhaustive literature review encompassing the evolution of inventory management techniques and the integration of AI within this domain. By synthesising insights from academic research papers, industry reports, and relevant publications, this review provides a comprehensive understanding of the historical context and contemporary developments in inventory management.

Impact Analysis: Building upon the foundation laid by the literature review, this study aims to delve deeper into the impact of AI on various facets of inventory management. Through rigorous analysis, we seek to elucidate how AI technologies have influenced inventory management strategies, operational efficiencies, and business performance metrics.

Identification of Improvement Areas: In tandem with the literature review and impact analysis, this study endeavours to identify specific areas within inventory management where AI implementation has been prominent. We will focus on delineating avenues for reducing costs and lead times through AI-driven optimisation.

Recommendation Formulation: Guided by the insights from the literature review and data analysis, this study will formulate actionable recommendations. These recommendations will serve as pragmatic guidelines for businesses leveraging AI to augment their inventory management practices effectively. Grounded in both theoretical insights and empirical evidence, our recommendations aim to facilitate informed decision-making and drive tangible improvements in inventory management efficiency.

1.2 Scope:

Literature Review: The scope of this study encompasses a thorough exploration of existing literature spanning academic research papers, industry reports, and pertinent publications. By tracing the historical evolution of inventory management techniques and elucidating recent advancements in AI technology, our review endeavours to provide a holistic perspective on the convergence of these domains.

Data Analysis: Through meticulous examination, we aim to uncover insights into existing inventory management systems and identify potential areas for optimisation.

The analysis will prioritise identifying strategies to mitigate costs and streamline lead times.

Recommendations: Drawing upon the literature review and data analysis findings, our study will proffer actionable recommendations tailored to businesses seeking to harness the transformative potential of AI in inventory management. By distilling insights from scholarly discourse and empirical observations, our recommendations aspire to empower organisations to navigate the complexities of AI-driven inventory optimisation effectively.

Research Question:

a) Study the evolution of different inventory management techniques and technology using literature.
b) Analyse the impact of Artificial Intelligence & Machine Learning algorithms in inventory management.
c) Analyse the impact of different ML algorithms on back-order prediction using data analysis and literature.

Research Gap:

There are several limitations to the study:

a) Limited datasets are available for the purpose.
b) The technology I used cannot perform hyperparameter tuning, which can improve the prediction model's performance.
c)

2. Literature Review

2.1 What is Inventory Management?

Inventory management is the art of balancing supply and demand to maintain optimal stock levels. It has served as a crucial function of an organisation's supply chain. It involves the management of raw materials, finished goods, and components, as well as the stocking and processing such items (Hayes, 2024).

Inventory management is a crucial aspect of the supply chain that involves monitoring inventory movement from manufacturers' facilities to warehouses and from there to the point of sale. Inventory management ensures that the right products are available at the right time and place (IBM, no date).

Traditionally, this process was carried out intuitively. However, with increasing levels of History, inventory management was predominantly a matter of intuition. Nevertheless, with the mounting fluctuations in demand and supply, relying on intuition or manual work has become increasingly intricate. Although the inception of inventory management remains ambiguous, it is generally accepted that merchants and shopkeepers were pioneers in this field. Inventory management before the Industrial Revolution was rudimentary. Shopkeepers and merchants depended on their handwritten notes and instincts to place orders. It was exceedingly challenging to account for stolen goods as shopkeepers had to dedicate hours, or even days, to count their inventory to identify discrepancies physically. It should be noted, however, that the magnitude of operations during that epoch was significantly smaller than those of today. Hence, there was no impetus to enhance efficiency, nor was there a comprehensive analysis of the existing operations within the healthcare supply chain (Peak Technologies, 2023).

In today's world, inventory management is a widely recognised challenge organisations face. Delays, stock-outs, and loss of production time are common issues researchers globally seek to address. Large organisations must have highly efficient delivery systems and supply chain management to ensure a smooth, efficient, and high-quality delivery of products and services. This is particularly essential in an environment where customer satisfaction is a prime factor that sets an organisation apart from its competitors. Effective inventory management is now seen more as a requirement than a trend. All organisations have inventory representing capital, which must be appropriately administered. They need to exercise greater inventory management control to reduce operational costs, which can motivate discerning organisations. Therefore, they employ various basic inventory management techniques or control methods to control their inventory costs. Inventory management is part of supply chain management and has become crucial in achieving organisational success (Aro-Gordon & Gupte, 2016).

2.2 Inventory Management Techniques

There are many inventory management techniques, each with its benefits and drawbacks. When choosing the best techniques, it is essential to consider the type of products sold, the size of the business, your overall budget and the level of accuracy needed to run an effective supply chain.

Here are the top 17 inventory management techniques and why they're essential:

1. *Demand Forecasting*

Accurately predicting future demand is the key to success in any business, and demand forecasting is the tool to achieve it. By forecasting demand, you can ensure that your business always has the optimal inventory levels to meet customer demand. Various demand forecasting techniques include moving averages, exponential smoothing, time series analysis, and judgmental forecasting. Considering its unique characteristics, it is crucial to choose the most suitable method for your business. However, it's important to remember that demand forecasting, while not entirely precise, can significantly reduce the chances of stockouts or overstocks (Tarver, 2023).

A paper by Aro-Gordon S. & Gupte J. (2016) has suggested, based on their experiment, maintaining an adequate amount of inventory in the warehouse to ensure efficient procurement of goods as and when required.

2. *ABC Analysis*

The ABC analysis technique is an invaluable tool for inventory management that enables businesses to prioritise their inventory items based on their significance. This approach provides better oversight of certain items, making it particularly useful when deciding which items to order and store. Conducting an ABC analysis requires dividing your inventory into three categories:

A Items: The most critical items account for 20% of the inventory but up to 80% of the inventory value.

B Items: Items with medium importance to the business account for 30% of the inventory items and roughly 15% of the inventory value.

C Items: The least essential items account for 50% of the inventory but only around 5% of the inventory value (Tarver, 2023).

Hence, a business must prioritise the order in which the goods and products must be replenished orderly (Aro-Gordon & Gupte, 2016).

3. *Safety Stock*

In inventory management, safety stock is critical to keeping businesses running smoothly. Essentially, it is a reserve of extra inventory that companies keep on hand in case of unexpected changes in demand or delivery issues. With this extra buffer, businesses can avoid stockouts that negatively impact their reputation and profits. Calculating the appropriate safety stock level depends on several factors, such as demand variability, delivery timeframes, stockout costs, and inventory holding costs. By finding the right balance, companies can maintain optimal levels of safety stock to ensure they always have enough inventory to meet customer needs, even during times of high demand or disruptions in the supply chain. Hence, Safety stock acts as a buffer amount that accounts for uncertainties such as:

- Excess demand.
- Supplier delays.
- Inaccurate demand or inventory forecasts.
- Failure to place timely reorders.
- Financial constraints (Jenkins, 2022).

cashflowinventory.com
Safety Stock
No Stockout
Data-driven
No Overstock
Optimize inventory and boost profitability
This process is fully automated

Fig 1: - Importance of Safety stock

Source: - https://cashflowinventory.com/blog/safety-stock/

In their 2004 paper, "The Effect of Lead Time Uncertainty on Safety Stocks," Chopra, Reinhardt, and Dada (2004) challenge the traditional understanding of safety stock management in inventory control. They argue that the commonly used normal distribution assumption for demand during lead time is more accurate. While previous research suggested that reducing lead time uncertainty leads to lower safety stock needs, Chopra et al. found that for cycle service levels exceeding 50%, decreasing lead time uncertainty can increase required safety stock. Their findings suggest that within this specific service level range, directly reducing lead times might be a more effective strategy for inventory reduction, mainly when demand exhibits high variability. This research highlights the importance of considering the specific characteristics of both demand and lead time distributions when making inventory management decisions.

4. *Reorder Points*

Inventory management is a crucial aspect of running a successful business. Paying close attention to reorder points is essential to ensure adequate inventory management. These points are the minimum inventory level at which new orders should be placed to avoid stockouts. It is important to note that reorder points are unique to each inventory item and are determined by considering average daily usage, order lead time, and safety stock levels (Kesavan, 2022).

Average daily usage refers to the number of products sold or used on an average day. This information is crucial when determining the reorder point as it helps ensure you have enough inventory to meet customer demand. On the other hand, order lead time is the time it takes to receive an order once it has been placed. Considering order lead time, you can ensure you have enough inventory to cover the time needed to receive a new order. Safety stock levels refer to the extra inventory kept on hand to ensure that unexpected spikes in demand can be met without causing stockouts (Kesavan, 2022).

Hence, the Reorder point calculation is done by using the formula:

ROP = (daily avg. usage X lead time) + safety stock

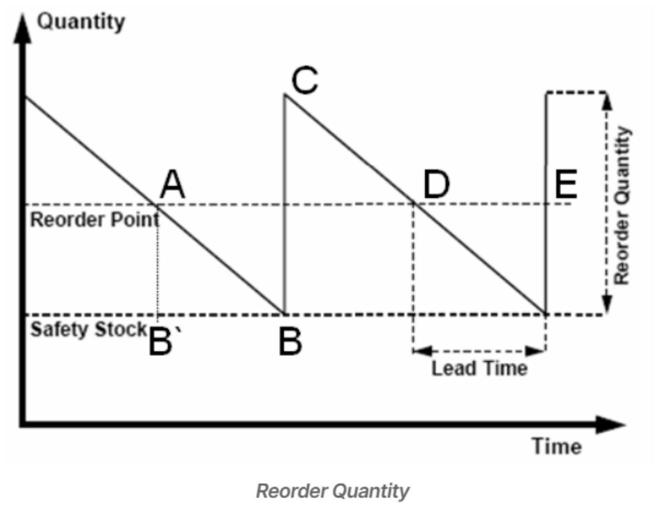


Fig 2: - Reorder Point

Source: - https://www.deskera.com/blog/inventory-reorder-point/

Establishing reorder points is critical to maintaining adequate inventory levels to meet customer demand consistently. It is a proactive approach to inventory management that helps avoid stockouts and ensures that customers are always satisfied with the products and services offered. Businesses can optimise inventory management practices and ensure long-term success by carefully analysing the factors contributing to establishing reorder points (Tarver, 2023).

5. *PAR Levels (Periodic Automatic Replenishment)*

PAR levels refer to the minimum and maximum inventory levels that should be maintained for an inventory item. A new order should be placed when the inventory quantity reaches the minimum level. However, the inventory level should stay within the maximum level. By implementing the PAR levels inventory management technique, businesses can avoid stockouts and overstock. This technique is beneficial for companies with perishable items, such as restaurants. The PAR levels are determined based on an item's average daily demand, lead time, and the amount of safety stock (Tarver, 2023).

Thus, the PAR levels allow the business to maintain the optimal safety stock levels.

There are several benefits of using this method: -

- Minimising inventory waste: - Optimally stored inventory helps reduce the amount of inventory going to waste due to being unsold or expired.
- Higher Profitability: - The PAR levels help reduce the costs associated with inventory waste and increase the business's profitability.

- Managing Seasonal Demands: - The PAR levels can be reset for every seasonal spike so that the business can respond to the changing needs of the hour (Hand, 2024).

6. *Just-in-Time (JIT) Inventory*

Just-in-time (JIT) inventory is a management approach aimed at minimising the amount of inventory you hold by ordering and delivering products only when you need them. To implement JIT, you need to meticulously monitor inventory levels and maintain a close relationship with suppliers. Although JIT can decrease inventory expenses, it can also lead to stockouts and may not be a feasible option for every business (Tarver, 2023).

In a 2004 European Journal of Operational Research publication, Pan, Lo, and Hsiao proposed a novel inventory management framework that surpasses traditional fixed lead times and examines methods for handling probabilistic demand. They question the Just-in-Time (JIT) ideal of eliminating lead time and accepting the inevitability of stockouts. To tackle this, they introduce the idea of negotiating lead times with suppliers and providing backorder discounts to customers willing to wait.

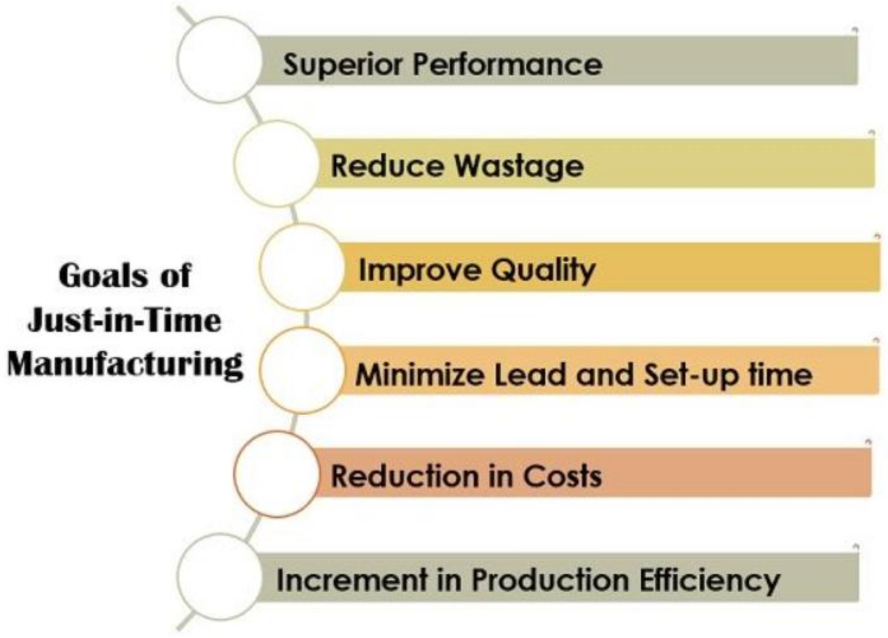

Goals of Just-in-Time Manufacturing
Superior Performance
Reduce Wastage
Improve Quality
Minimize Lead and Set-up time
Reduction in Costs
Increment in Production Efficiency

Fig3: - Just-in-time Manufacturing Goals

Source: - https://theinvestorsbook.com/just-in-time-manufacturing.html

Their models consider both normally and generally distributed demand and strive to optimise order quantity, lead time (considering lead time reduction costs), back ordering strategy (including discount levels), and reorder point while adapting to dynamic market conditions such as sales growth and demand volatility. This study presents a fresh perspective on inventory management in uncertain demand by integrating lead time negotiation, backorder discounts, and market adaptability. This dynamic approach can achieve significant cost savings and enhance customer satisfaction.

7. *Dropshipping*

Dropshipping is a revolutionary inventory management technique that has gained immense popularity among e-commerce businesses. This method allows sellers to offer a wide range of products without investing in stocking inventory. Instead, they partner with third-party suppliers who handle the products' storage, packaging, and shipping directly to the customers.

This approach significantly reduces the financial burden on sellers, as they no longer need to allocate funds for purchasing and storing inventory. It also minimises the risk of unsold stock and allows businesses to offer diverse products without the associated overhead costs.

However, it's essential to recognise that while dropshipping offers these compelling advantages, it also comes with inevitable trade-offs. Sellers need more control over the shipping and fulfilment process, which can impact the overall customer experience. Additionally, the dropshipping fees charged by third-party suppliers may affect the seller's profit margins, necessitating carefully considering pricing strategies to maintain profitability (Tarver, 2023).

8. *Cross-Docking*

Cross-docking is a highly advanced logistics strategy involving immediately transferring received inventory from direct inbound to outbound transportation, bypassing the traditional warehousing process. This method minimises or eliminates the need for storage, reducing handling and holding costs while significantly improving order fulfilment times. Its effectiveness is particularly pronounced in perishable products, where swift and efficient movement through the supply chain is crucial to maintaining product quality and reducing the risk of spoilage.

Successful implementation of cross-docking requires a high degree of coordination and collaboration with suppliers to ensure that products arrive and depart according to a carefully orchestrated schedule. Due to its reliance on precise timing and synchronisation, cross-docking may not be suitable for companies dealing with non-perishable products or items with low turnover rates, as the benefits of this approach are most evident in high-velocity supply chains characterised by rapid product movement and distribution (Tarver, 2023).

9. *Inventory Management Software*

Inventory management software is indispensable for businesses to oversee their supply chains effectively. This software lets companies closely track inventory levels, predict future demand, and make well-informed decisions when procuring goods. Inventory management software should be tailored to the unique requirements of each business.

Its implementation can substantially improve operational efficiency and cost savings across various industries (Tarver, 2023).

10. *FIFO and LIFO*

The first in, first out (FIFO) and last in, first out (LIFO) are two popular methods used in inventory management to determine the order in which inventory items are sold. With FIFO, the oldest inventory items are sold first, while with LIFO, the newest items are sold first.

FIFO is often employed in industries where perishable goods are involved, as it ensures that the oldest items are sold first, reducing the risk of spoilage and waste. On the other hand, LIFO is preferred in industries where the inventory cost tends to increase over time. This method helps to match current revenues with the most recent costs, and it can also have tax benefits by deferring taxes on the higher-cost inventory items (Tarver, 2023).

11. *Consignment Inventory*

Consignment inventory is a business arrangement in which a supplier, the consignor, provides goods to a retail business, known as the consignee, but maintains ownership of the goods until they are sold. The advantage of this arrangement is that the consignee can only pay for the goods once they are sold, and the consignor is responsible for shipping costs (Tarver, 2023). However, it is essential to note that the consignee is responsible for holding costs and selling the products, which means they could incur losses if they do not sell.

12. *Economic Order Quantity (EOQ)*

Economic order quantity (EOQ) is a widely used inventory management technique that helps businesses determine the optimal order quantity for a product to minimise total inventory costs. The EOQ calculation considers the specific product's annual demand, ordering cost, and holding cost to arrive at the optimal quantity that should be ordered each time. By using this technique, businesses can reduce costs associated with holding inventory while ensuring enough inventory is on hand to meet customer demand (Tarver, 2023).

13. *Perpetual Inventory Management*

Perpetual inventory management is a sophisticated inventory control system that utilises real-time updates to track inventory levels as goods are sold or received. This method offers an up-to-date understanding of inventory levels, leading to enhanced inventory turnover and reduced inventory stockouts. However, this approach may require more time, resources, and expertise than other inventory management techniques, such as periodic inventory management, where inventory levels are updated at predetermined intervals rather than continuously (Tarver, 2023).

14. *Minimum Order Quantity (MOQ)*

The minimum order quantity (MOQ) is the lowest number of items that must be ordered from a supplier simultaneously. Suppliers set MOQs to lower shipping costs, limiting a seller's ordering flexibility and leading to increased costs if more items are ordered than needed (Tarver, 2023).

15. *Six Sigma and Lean Six Sigma*

Six Sigma is a well-established methodology for managing inventory, with the primary goal of reducing variations and defects in the inventory management process. It utilises data-driven techniques, including statistical analysis, to identify and address potential issues. This methodology can be effectively used to improve inventory tracking and management processes.

Lean Six Sigma combines the principles of Six Sigma with lean manufacturing to optimise the efficiency of the inventory management process. Eliminating unnecessary steps can streamline the entire process, improving overall performance (Tarver, 2023).

16. *Bulk Shipping*

Bulk shipping refers to procuring and transporting a significant volume of inventory in one go. This method is advantageous as it can lead to substantial reductions in shipping expenses and may open up the possibility of securing discounts from suppliers. However, it's crucial to be mindful of the potential drawbacks, such as the risk of overstocking and the implications of selling perishable goods when employing this approach(Tarver, 2023).

17. *Batch Tracking*

Batch tracking is an inventory management technique that helps businesses monitor groups of similar items throughout the supply chain. It is commonly used for perishable inventory items and those that can be recalled.

Batch-tracking businesses typically employ barcodes or RFID tags to track items. This is especially helpful for maintaining quality control and compliance for items with an expiration date.

Effective inventory management is crucial for any business buying and selling goods. Inventory management techniques can help increase revenues, reduce costs, and improve customer satisfaction. Testing different methods to find the best combination for your business is essential. (Tarver, 2023).

2.3 Evolution of Inventory Management Systems

In 1889, Herman Hollerith created the first punch card that machines could read, revolutionising how data was recorded for various purposes such as census-taking and work timesheets—the punch card worked by creating tiny holes in sheets of paper. Building on Hollerith's idea, Harvard University developed the first modern check-out system in the 1930s. It used a punch card corresponding to catalogue items and was designed to manage inventory and generate billing. Customers would fill out the punch cards, and a computer would read them, sending the information to the storeroom, which would then bring the item to the customer. Although a similar system is still used today for expensive and controlled items such as medication, it did not become widely adopted due to its high costs and slow lead time (Peak Technologies, 2023).

The precursor to the modern barcode was developed in the late 1940s and early 1950s. It consisted of ultraviolet light-sensitive ink and a reader. The first barcode, resembling a bullseye, was invented by two Drexel University students named Norman J Woodland and Bernard Silver in 1948. They wanted to solve the supermarket industry's inventory management and customer check-out problems. The duo received a patent for their invention in 1952. However, the barcode system could have been more practical due to technology limitations at that time despite working well in the lab. It was bulky and needed more computing power required for success. In the 1960s, retailers devised the modern barcode to monitor inventory. Cheaper and faster scanners were made possible by lasers. The Universal Product Code (UPC) was widely adopted in the late 1960s. On June 26, 1974, the first item, a 10-pack of Juicy Fruit gum, was scanned with a barcode at a Marsh supermarket in Troy, Ohio (Silverman, 2018).

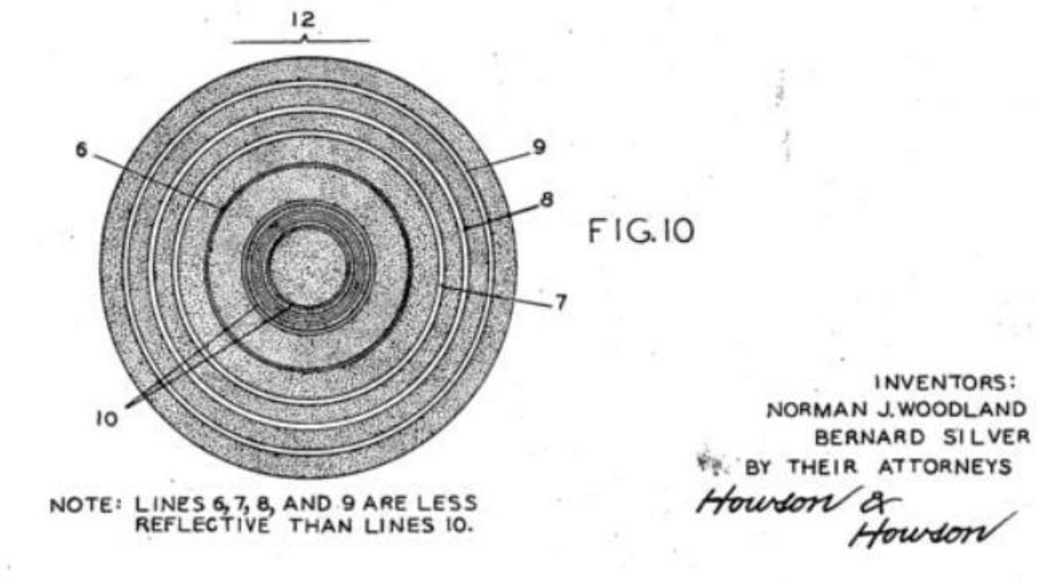


Fig 4: - First Barcode (Bulls eye design)

Source: - https://corp.trackabout.com/

The advent of advanced computer and software systems during the 1980s and 1990s resulted in more efficient inventory tracking. These systems followed a cycle encompassing purchasing, inventory tracking, monitoring, and backtracking. The widespread availability of personal computers led to a drastic reduction in the cost of barcodes and readers. However, barcode inventory management could have been faster to gain momentum, particularly among small and medium-sized businesses, due to the unavailability of proper storage facilities. The manual inventory tracking method was replaced with scanning products and entering data into computers by hand. The early 2000s saw the emergence of inventory management software that eliminated the need for manual data input. Barcode readers could instantly update databases, making inventory tracking even more efficient (Peak Technologies, 2023).

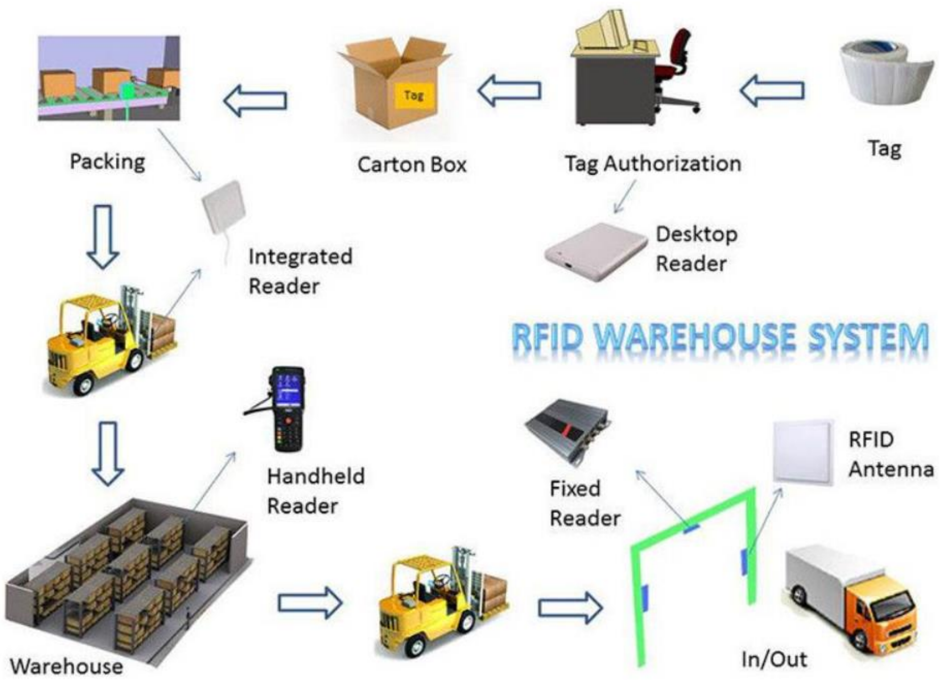


Fig5: - RFID Warehouse System

Source: - https://www.peerbits.com/blog/warehouse-smart-inventory-management-solution.html

The introduction of radio-frequency identification (RFID) technology in the 1970s marked a significant milestone in inventory management. While its widespread use in warehouses, factories, and retail stores began in the early 2000s, RFID technology has since emerged as an advanced alternative to traditional barcodes. Unlike barcodes, RFID tags can transmit vital information about a product, such as its serial number, type, or manufacturer, to a scanner without being in direct sight, making it ideal for warehouses with high shelves. Furthermore, RFID tags can store more information than typical barcodes, making them a valuable tool for businesses that require detailed product tracking.

In recent years, research has proposed various inventory management systems that leverage the combined power of the Internet of Things (IoT) and Machine Learning (ML) to improve efficiency further. A recent paper by Manaswini Mohapatro and Ayaskanta Mishra proposes a system that utilises real-time data collected through passive RFID tags to track items within a warehouse. This data is then fed into machine learning algorithms to **predict demand fluctuations, enabling proactive stock management and classification of Stock Keeping Units (SKUs)** based on anticipated demand variations. The proposed system has the potential to significantly improve inventory management efficiency and mitigate the disruptions caused by unexpected fluctuations, particularly during a pandemic.

"The Effect of RFID On Inventory Management and Control" is an insightful chapter contributed by Uttarayan Bagchi, Alfred Guiffrida, Liam O'Neill, Amy Zeng, and Jack Hayya to the book "Trends in Supply Chain Design and Management: Technologies and Methodologies." The authors provide a detailed analysis of how RFID technology can revolutionise inventory management and control. They argue that adopting RFID technology can improve the flow of information, reducing the variance of an inventory system and its associated costs. The authors offer a comprehensive overview of RFID technology, its potential applications in various industries, and its superiority over existing identification technologies.

Another research article by Preetam D'Souza, Max Guo, David Wang, and Insup Lee discusses the application of RFIDash. This middleware layer utilises RFID technology to offer real-time inventory management and checkout solutions. The RFIDash system has two modes - inventory mode and checkout mode. The former employs a rectangular setup with four antennae, whereas the latter uses a single antenna at the checkout aisle. The article highlights the future work required for RFIDash, including large-scale testing, full retail integration, automation, and transparent data collection. Additionally, the article sheds light on the ethical concerns associated with using RFID technology, particularly customer privacy concerns. **Collecting data on inventory and sales trends is crucial to maintaining a competitive advantage in the retail industry, and RFID technology can help businesses achieve this.**

Overall, the research findings demonstrate the potential of RFID technology in enhancing inventory management systems and highlight the importance of further research in this field. The authors also examine the ethical implications and societal trade-offs of adopting RFID technology, emphasising the need to balance the benefits of RFID technology with privacy concerns. They highlight the potential risks of excessive data sharing and the need for adequate safeguards to protect individual privacy.

In conclusion, RFID technology has the potential to revolutionise inventory management and control. Its adoption can lead to significant cost savings while enhancing supply chain performance. However, it is vital to balance the benefits of RFID technology with privacy concerns to ensure its adoption is ethical and sustainable. Next, we will investigate how AI has impacted Inventory Management since its advent.

2.4 Artificial Intelligence (AI) & Inventory Management

Inventory management has significantly evolved, with various technological advancements such as Artificial Intelligence (AI) emerging and transforming how businesses manage their inventory. The application of AI in inventory management has been a topic of extensive research, with several studies conducted to explore its benefits and potential future aspects.

According to the study by Albayrak Ünal et al. (2023), machine learning algorithms such as **neural networks, decision trees, and support vector machines** were the most used AI methods. These algorithms have been found to improve **demand forecasting accuracy, optimise inventory levels, reduce stockout costs, and increase customer**

satisfaction. For instance, historical sales data can be analysed to identify patterns and predict future demand, which can help companies avoid overstocking or stockouts. AI can also help companies optimise inventory replenishment policies by considering lead time, safety stock levels, and demand variability.

The study also identified several potential future aspects of inventory management that could benefit from the use of AI. Real-time inventory management systems that use sensors and IoT devices to track inventory levels and automatically trigger replenishment orders when inventory levels fall below a certain threshold can improve supply chain agility and responsiveness. Predictive maintenance systems that use AI algorithms to analyse sensor data and predict when maintenance is needed can reduce downtime and maintenance costs. Autonomous inventory replenishment systems that use AI algorithms to automatically place orders with suppliers when inventory levels fall below a certain threshold can improve supply chain efficiency and **reduce the risk of stockouts.**

Another study by Navdeep Singh and Daisy Adhikari encompasses a comprehensive analysis of the role of AI in managing inventory. The paper discusses various AI applications in inventory management, such as demand forecasting, stock optimisation, and automated reordering, highlighting AI's transformative impact on supply chain operations.

However, the authors also acknowledge the challenges associated with AI implementation, such as **data quality, interpretability, and model transparency**. They suggest that integrating AI with emerging technologies like the **Internet of Things** (IoT) could lead to innovative solutions that were previously unimaginable.

Moreover, the paper includes an in-depth review of inventory management's evolution and current state. The authors draw insights from studies that examine supply/demand mismatch and associated costs in supply chain processes. Through these insights, they provide a balanced view of the impact of AI on inventory management, emphasising both its current benefits and the need to overcome hurdles for successful integration.

Despite these challenges, the authors highlight AI's numerous opportunities for inventory management. They discuss the potential for AI to improve demand forecasting accuracy, optimise inventory levels, and automate reordering processes.

Healthcare is another critical area of research that requires innovative optimisation methods for drug replenishment in a hospital ward. Traditionally, most wards continue to keep an inventory of drugs necessary for short periods, between one and three days. A paper by Galli et al. (2023) proposes a model that uses machine learning coupled with stochastic optimisation to consider the historical usage patterns of drugs and the ward's current situation to minimise inventory levels and the necessity for emergency replenishments. The paper aims to show that machine learning methods can select a set of scenarios over which a stochastic optimisation approach can deliver good decisions regarding small order quantities and service quality (avoiding emergency replenishment orders).

The research used data from one year of drug administration in an Italian ward, and the results support the superiority of the proposed approach over traditional ones. The findings indicated that machine learning models still need to enter internal logistics in hospitals. However, the availability of vast amounts of data creates new opportunities for decision-making under uncertainty, and the renewed attention towards machine learning provides new tools that can be used to generate predictions of quantities that may be of interest for inventory management.

The paper's conclusions suggest that the proposed approach can improve inventory policies and generate considerable savings in a hospital's operating budget, improving the quality of patient care and services. The study recommends that hospitals prioritise inventory optimisation and internal supply chain management to ensure drug supplies are always accessible to clinical staff. In addition, the paper suggests that an ad hoc feature engineering procedure be designed to exploit ward features that are easily collected every day in the ward.

The study's methodology used machine learning algorithms such as **K-nearest neighbours, decision trees, random forests, and XGBoost** to predict drug consumption and optimise drug replenishment. The paper showed that machine learning algorithms can significantly improve inventory policies by reducing inventory levels and minimising emergency replenishment orders. The study's validation metric was tailored for selecting hyperparameters to balance optimality and robustness (Galli et al., 2021).

In conclusion, the proposed approach can improve inventory policies in healthcare facilities and generate significant cost savings. The study's methodology provides a framework that healthcare facilities can adopt to optimise drug replenishment and enhance the quality of patient care and services. Therefore, hospitals should prioritise supply chain management and inventory optimisation to ensure that drug supplies are always available and accessible to clinical staff. Additionally, future research can be conducted to evaluate the feasibility of the proposed approach in other healthcare facilities and to investigate the use of different machine learning algorithms to optimise drug replenishment in healthcare facilities.

A study by Kamble & Tewari (2022) aimed to explore the impact of predictive data analytics (PDA) using Big Data Analysis on the performance of inventory optimisation enablers in the Landmark Group, UAE. The research used a survey method with a structured questionnaire designed based on an initial theoretical construct derived from the literature review. The survey data were analysed using descriptive statistical analysis and multivariate analysis of variance with and without prominent data characteristics applied as moderators. The findings revealed that Support Vector Machine and Flow-based Events Analysis significantly positively impacted inventory optimisation enablers, even with limited usage. Other influential techniques were **Demands Pattern Recognition, Demand-Supply Predictions, and Regression Analysis**. These findings support previous studies showing that predictive analytics can improve supply chain performance by enhancing visibility, predictions, and forecasting, reducing communication gaps, reducing lead times, improving operational accuracy and effectiveness, and optimising costs.

However, the study also found that prominent data characteristics can significantly negatively impact the performance of PDA techniques. The risks of overloading PDA techniques with big data without appropriate processing include data quality issues, lack of context, and difficulties in categorisation, which supports previous research.

Hence, future research should explore the impact of PDA using extensive data analysis on other supply chain performance indicators, such as customer satisfaction, delivery performance, and inventory turnover.

A paper by Meller et al. (2018) investigates the performance drivers for **data-driven inventory management** in a Newsvendor setting with non-stationary demand. The authors analyse two novel approaches for inventory management based on machine learning techniques - **linear quantile regression and tree-based regression** - which use historical demand observations and auxiliary data to prescribe optimal inventory quantities. The authors identify three major performance drivers: non-linearity, heteroscedasticity, and usability. In evaluating both models through simulation experiments and real-world data sets, the authors recommend the **tree-based approach**, which they find to be more robust, particularly for feature-dependent noise and tiny datasets.

The authors find that traditional inventory models that rely solely on historical demand forecasts need to be revised for industries with non-stationary demand patterns. As a result, decision-makers frequently need more inventories and sizeable unmet demand. However, with the availability of large amounts of data that can potentially explain demand variations, recent work in operations management has utilised data such as Google searches, clickstreams, weather information, and sensor-equipped machinery to extract suitable features that potentially have predictive power and incorporate them into inventory models.

In conclusion, the authors' findings offer valuable insights into the applicability of machine learning techniques to inventory management in industries with nonstationary demand patterns. The authors recommend the tree-based approach, particularly for feature-dependent noise and tiny datasets, which provides decision-makers with a valuable tool for optimising inventory quantities. The authors' emphasis on usability highlights the importance of practical applicability in real-world settings, where decision-makers often face limited datasets and must make quick and informed decisions. Future research could build on the authors' findings by investigating the applicability of other machine learning techniques and the impact of different feature selection and data pre-processing techniques on the performance of machine learning-based inventory management models.

Green process innovation has been identified as a valuable approach to enhancing environmental performance in healthcare facilities. However, integrating green practices into operations and supply chain management (OSCM) requires practical data exploration technologies to improve visibility and decision-making. Big **Data Analytics Capability** (BDAC) is one such technology that has the potential to play a critical role in this integration. One such study by Benzidia S. et al. (2023) investigates the relationship between BDAC, environmental process integration, green process innovation in OSCM, and ecological performance in healthcare facilities. The study found that BDAC influences environmental process integration and green process innovation, enhancing environmental performance. Moreover, the study highlights the mediating role of green process innovation on environmental performance. The paper provides valuable insight for managers and stakeholders to support the application of BDAC in healthcare OSCM to create sustainable value.

BDAC positively impacts environmental process integration and green process innovation in healthcare OSCM. BDAC can be used to integrate environmental concerns into operations and supply chain management, further strengthening healthcare facilities' environmental performance. It also highlights the importance of green process innovation in achieving high-performance levels in green process integration. The paper provides valuable insight for managers and stakeholders to support the application of BDAC in healthcare OSCM to create sustainable value. The findings of this study can be helpful in healthcare facilities' implementation of green practices in their operations and supply chain management, which can lead to improved environmental performance.

Dhaliwal et al. (2023) studied the potential impacts of implementing Artificial Intelligence in inventory management practices and how it affects the profitability of an organisation. In their study, the authors evaluated techniques such as **Machine Learning, Predictive analytics**, and **computer vision** to **reduce overstocking and understocking** and accurately forecast the demand by analysing the quantitative and qualitative data from over 50 organisations. Implementing AI in the inventory management system has benefitted organisations by improving their efficiency, control over inventory, decision-making, visibility, and reduced costs. A regression analysis was performed in the study, which further cemented that AI implementation has positively impacted organisations' inventory management. Overall, the study highlights the potential positive effects of AI implementation in inventory systems while mentioning the future scope of the study.

3. Research Methodology

Literature Review on Inventory Management Techniques and Technology Evolution:

- Begin by conducting a comprehensive literature review to study the evolution of various inventory management techniques and technologies.
- Utilize academic databases such as Google Scholar, ScienceDirect, IEEE, and relevant industry journals to identify scholarly articles, research papers, and publications related to inventory management.
- Extract information on historical developments, emerging trends, and technological advancements in inventory management, including traditional methods, such as

ABC analysis and EOQ models, and modern technologies like RFID, IoT, and automation.
- Synthesize the findings to understand the evolution of inventory management practices over time and identify key drivers shaping the field.

Impact Analysis of Artificial Intelligence & Machine Learning in Inventory Management:

- Explore the role of Artificial Intelligence (AI) and Machine Learning (ML) algorithms in revolutionising inventory management practices.
- Review literature on applying AI and ML techniques in inventory management systems, including predictive analytics, demand forecasting, and optimisation algorithms.
- Analyze case studies, research papers, and industry reports to assess the benefits, challenges, and implications of AI and ML in inventory management, such as improved accuracy, efficiency, and decision-making capabilities.
- Examine real-world examples and success stories of organisations leveraging AI and ML for inventory optimisation and supply chain management.

Impact Analysis of ML Algorithms on Back-Order Prediction:

- Employ data analysis techniques to analyse the impact of different ML algorithms on back-order prediction.
- Acquire suitable datasets or access publicly available inventory management and back-order prediction datasets.
- Preprocess the data to handle missing values and outliers and ensure data quality.
- Utilize Python programming language and relevant libraries (e.g., pandas, scikit-learn) within the Jupyter Notebook environment to perform exploratory data analysis and feature engineering.
- Implement various ML algorithms for back-order prediction, such as Decision Trees, Random Forests, Support Vector Machines, and Neural Networks.
- Evaluate the performance of each ML algorithm using appropriate evaluation metrics, including accuracy, precision, recall, F1-score, and area under the ROC curve (AUC).
- Compare and contrast the effectiveness of different ML algorithms in predicting back-orders based on empirical results and findings from the literature.
- Interpret the implications of the analysis and provide insights into the suitability of ML techniques for improving back-order prediction accuracy and inventory management efficiency.

Ethical Considerations:

- Throughout the research process, ethical considerations were paramount, particularly regarding data usage, citation practices, and adherence to copyright regulations.

- Proper attribution and citation were maintained for all sources consulted during the literature review to ensure academic integrity and respect intellectual property rights.
- The publicly available datasets adhered to the terms of use and licensing agreements specified by the dataset providers.

Research Tools:

- In addition to Python and Jupyter Notebook, various libraries and frameworks within the Python ecosystem were leveraged for data analysis, including but not limited to pandas, NumPy, scikit-learn, and Matplotlib.
- These tools facilitated data preprocessing, exploratory data analysis, feature engineering, model development, evaluation, and visualisation.

This structured research methodology aims to delve into the evolution of inventory management techniques, evaluate the influence of Artificial Intelligence (AI) and Machine Learning (ML) on inventory management practices, and scrutinise the efficacy of ML algorithms in predicting back-orders. By doing so, the project endeavours to make meaningful contributions to supply chain management and operations research, thereby advancing our understanding of optimising inventory systems and enhancing decision-making processes in modern businesses.

4. Data Analysis

When a customer orders a product that is not currently available or temporarily out of stock and chooses to wait until the product is back in stock and promised to be shipped, this is known as a backorder. If not handled promptly, backorders can significantly impact a company's revenue, share market price, and trust, potentially resulting in lost sales or customers. However, promptly fulfilling backorders can add significant pressure to various supply chain stages, increasing labour, production, and shipping costs. Moreover, predicting demand can be challenging due to uncertainty about customer demand, making traditional supply chain management systems less effective. Some companies leverage machine learning prediction processes to forecast backorders and reduce tangible and intangible costs (Islam & Amin, 2020).

DATASET: The dataset was initially used for competition on Kaggle, after which it was made private. However, I managed to get the data from another source: https://data.world/amitkishore/can-you-predict-products-back-order.

Below are the features listed in the dataset as columns:

1. SKU – Random ID for the product.
2. national_inv – Current inventory level for the part.
3. lead_time – Transit time for product (if available).
4. in_transit_qty – Amount of product in transit from source.
5. forecast_3_month – Forecast sales for the next three months.
6. forecast_6_month – Forecast sales for the next six months.

7. forecast_9_month – Forecast sales for the next nine months.
8. sales_1_month – Sales quantity for the prior 1-month time period.
9. sales_3_month – Sales quantity for the prior 3-month time period.
10. sales_6_month – Sales quantity for the prior 6-month time period.
11. sales_9_month – Sales quantity for the prior 9-month time period.
12. min_bank – Minimum recommend amount to stock.
13. potential_issue – Source issue for part identified.
14. pieces_past_due – Parts overdue from source.
15. perf_6_month_avg – Source performance for the prior 6-month period.
16. perf_12_month_avg – Source performance for the prior 12-month period.
17. local_bo_qty – Amount of stock orders overdue.
18. deck_risk – Part risk flag.
19. oe_constraint – Part risk flag.
20. ppap_risk – Part risk flag
21. stop_auto_buy – Part risk flag.
22. rev_stop – Part risk flag.
23. went_on_backorder – The product went on backorder. This is the target value.

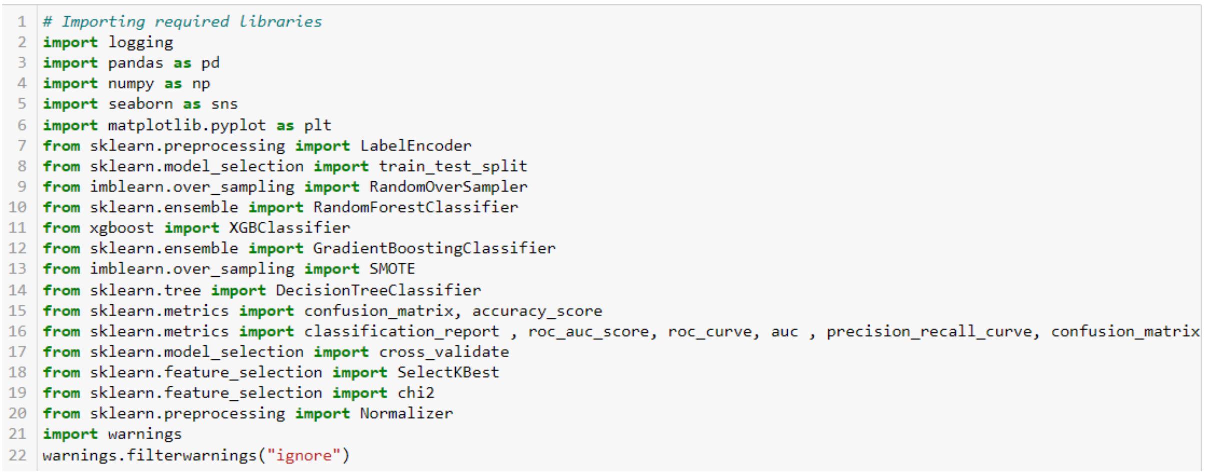
I have utilised Jupyter Notebook and Python to run backorder ML algorithms. First, the dataset was loaded in the Python kernel inside a Jupyter notebook.

As we have two datasets, training and testing, we combine them into a single data set to examine the dataset as one.

Below are the steps followed for the same:

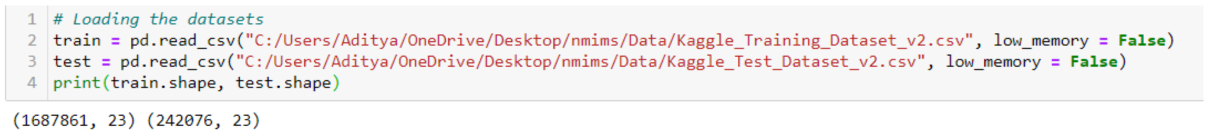
4.1Importing required libraries in the notebook.

The required libraries were imported to proceed with the backorder prediction problem.



```python
# Importing required libraries
import logging
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report , roc_auc_score, roc_curve, auc , precision_recall_curve, confusion_matrix
from sklearn.model_selection import cross_validate
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import Normalizer
import warnings
warnings.filterwarnings("ignore")
```

4.2Importing the datasets.



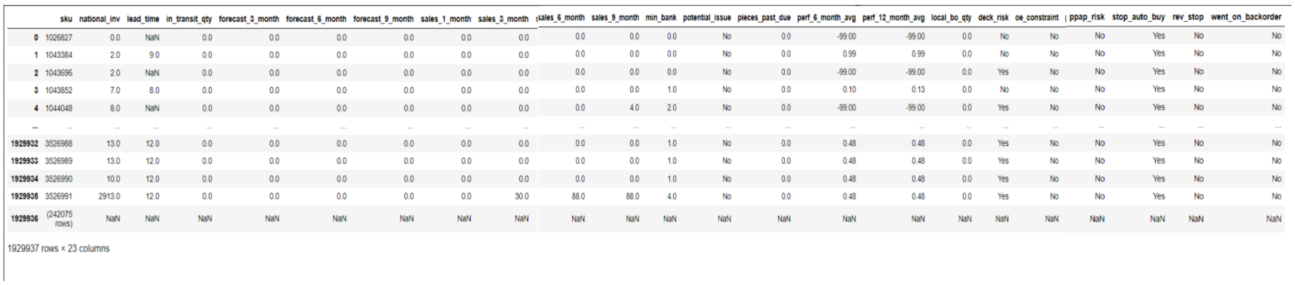
```
# Loading the datasets
train = pd.read_csv("C:/Users/Aditya/OneDrive/Desktop/nmims/Data/Kaggle_Training_Dataset_v2.csv", low_memory = False)
test = pd.read_csv("C:/Users/Aditya/OneDrive/Desktop/nmims/Data/Kaggle_Test_Dataset_v2.csv", low_memory = False)
print(train.shape, test.shape)
```

```
(1687861, 23) (242076, 23)
```

The above image shows the size of two datasets after importing. Both datasets have 23 columns. The training dataset has 1687861 rows, and the test data has 242076 rows.

4.3 Exploratory Data Analysis.

I combined the datasets into one and then loaded it again. So, the updated dataset has 1929937 rows and 23 columns. The serial numbers (indexing) of the rows were readjusted. The updated dataset is shown below.



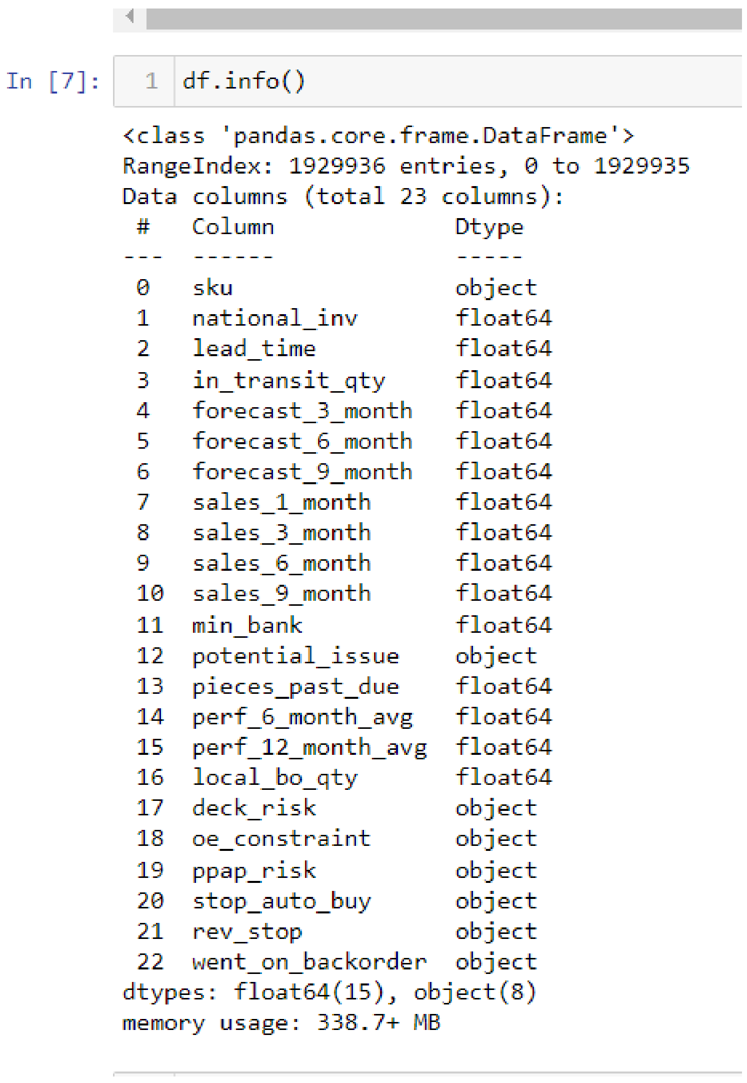
	sku	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9_month	sales_1_month	sales_3_month	sales_6_month	sales_9_month	min_bank	potential_issue	pieces_past_due	perf_6_month_avg	perf_12_month_avg	local_bo_qty	deck_risk	oe_constraint	ppap_risk	stop_auto_buy	rev_stop	went_on_backorder
0	1026827	0.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	No	0.0	-99.00	-99.00	0.0	No	No	No	Yes	No	No
1	1043384	2.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	No	0.0	0.99	0.99	0.0	No	No	No	Yes	No	No
2	1043696	2.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	No	0.0	-99.00	-99.00	0.0	Yes	No	No	Yes	No	No
3	1043852	7.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	No	0.0	0.10	0.13	0.0	No	No	No	Yes	No	No
4	1044048	8.0	NaN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	2.0	No	0.0	-99.00	-99.00	0.0	Yes	No	No	Yes	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1929932	3526988	13.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	No	0.0	0.48	0.48	0.0	Yes	No	No	Yes	No	No
1929933	3526989	13.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	No	0.0	0.48	0.48	0.0	Yes	No	No	Yes	No	No
1929934	3526990	10.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	No	0.0	0.48	0.48	0.0	Yes	No	No	Yes	No	No
1929935	3526991	2913.0	12.0	0.0	0.0	0.0	0.0	0.0	30.0	88.0	88.0	4.0	No	0.0	0.48	0.48	0.0	Yes	No	No	Yes	No	No
1929936	(242075 rows)	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1929937 rows × 23 columns

If we carefully observe the last row of the dataset, we will see that it has missing values throughout. Hence, we will drop the last row in the upcoming code.

Our target variable is went_on_backorder. This dataset feature mentions whether the product did or did not go on backorder.

Next, we will analyse the different types of data in our dataset.



```
In [7]:   df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1929936 entries, 0 to 1929935
Data columns (total 23 columns):
 #   Column             Dtype  
---  ------             -----  
 0   sku                object 
 1   national_inv       float64
 2   lead_time          float64
 3   in_transit_qty     float64
 4   forecast_3_month   float64
 5   forecast_6_month   float64
 6   forecast_9_month   float64
 7   sales_1_month      float64
 8   sales_3_month      float64
 9   sales_6_month      float64
 10  sales_9_month      float64
 11  min_bank           float64
 12  potential_issue    object 
 13  pieces_past_due    float64
 14  perf_6_month_avg   float64
 15  perf_12_month_avg  float64
 16  local_bo_qty       float64
 17  deck_risk          object 
 18  oe_constraint      object 
 19  ppap_risk          object 
 20  stop_auto_buy      object 
 21  rev_stop           object 
 22  went_on_backorder  object 
dtypes: float64(15), object(8)
memory usage: 338.7+ MB
```

The above image shows eight object features and fifteen numeric features. Float 64 refers to decimal values, whereas the object type data is generally text data (non-numeric) in a string format.

Next, I checked the distribution of our target variable.

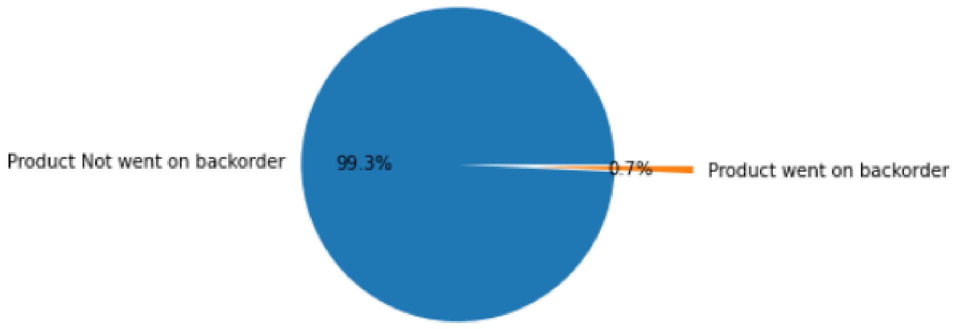
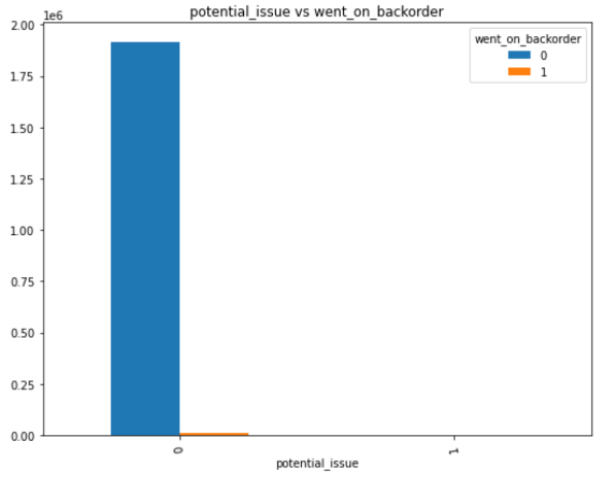


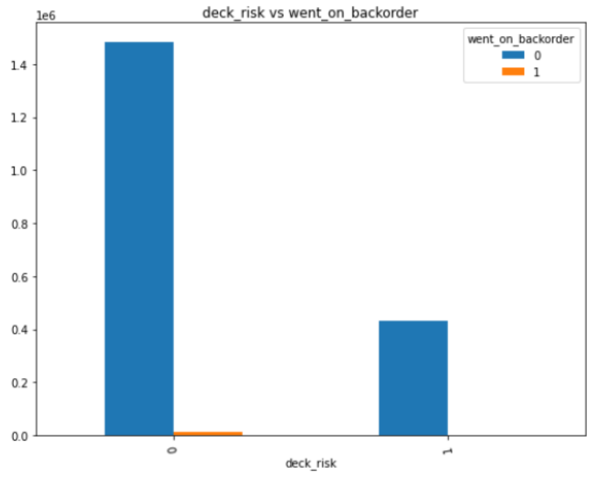
Fig 6: Class Balance of the dataset

As we can observe, the dataset is highly biased in favour of products that did not go on backorder. Only 0.7% of the dataset had products with backorder. Hence, to ensure that the model is trained correctly, we need to balance the data either by oversampling with the minority class or undersampling the majority class. This process is completed while building different models.

- Univariate Analysis
  Univariate analysis refers to the study of a single variable at a time. This helped me understand the distribution of data for different features. Initially, two data types were discovered in the original data: numeric (here only float 64) and categorical. Below are the univariate plots for the categorical data.

```
potential_issue state is 0:
   Product 0 0 1915020 times (99.28%)
   Product 1 0 13927 times (0.72%)
potential_issue state is 1:
   Product 0 1 935 times (94.54%)
   Product 1 1 54 times (5.46%)
deck_risk state is 0:
   Product 0 0 1482779 times (99.22%)
   Product 1 0 11704 times (0.78%)
deck_risk state is 1:
   Product 0 1 433176 times (99.48%)
   Product 1 1 2277 times (0.52%)
oe_constraint state is 0:
   Product 0 0 1915672 times (99.28%)
   Product 1 0 13972 times (0.72%)
oe_constraint state is 1:
   Product 0 1 283 times (96.92%)
   Product 1 1 9 times (3.08%)
ppap_risk state is 0:
   Product 0 0 1685534 times (99.30%)
   Product 1 0 11850 times (0.70%)
ppap_risk state is 1:
   Product 0 1 230421 times (99.08%)
   Product 1 1 2131 times (0.92%)
stop_auto_buy state is 0:
   Product 0 0 69966 times (99.18%)
   Product 1 0 578 times (0.82%)
stop_auto_buy state is 1:
   Product 0 1 1845989 times (99.28%)
   Product 1 1 13403 times (0.72%)
rev_stop state is 0:
   Product 0 0 1915120 times (99.28%)
   Product 1 0 13977 times (0.72%)
rev_stop state is 1:
   Product 0 1 835 times (99.52%)
   Product 1 1 4 times (0.48%)
```





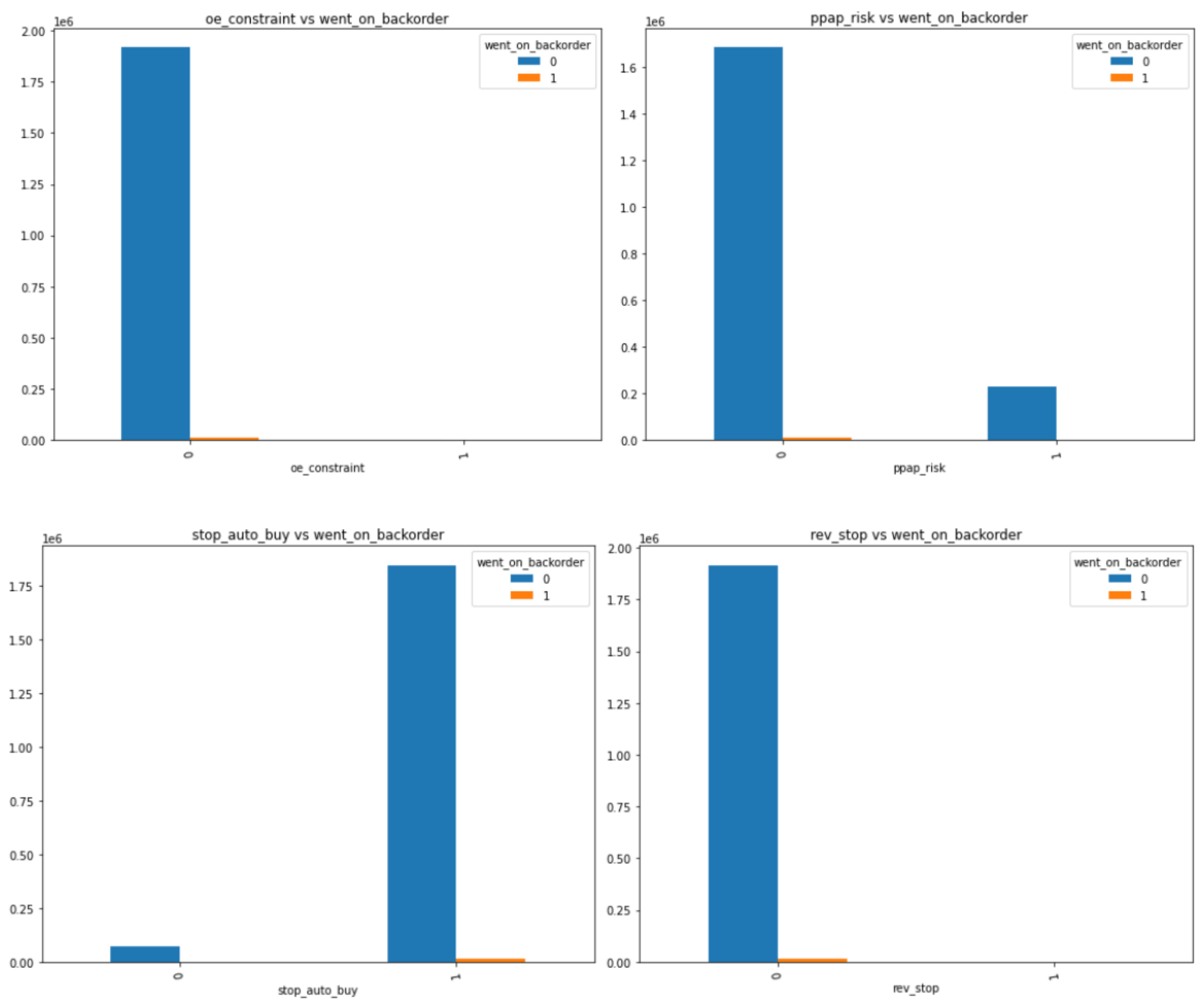


Fig 7: Univariate plots of categorical variables

As we can observe from the above plots, the one state or the 'Yes' state of the feature 'went_on_backorder' is very few instances.

1. **When the potential issue state is 0 (indicating no potential issues):**
   - For products that did not go on backorder:
     - Occurred 1,915,020 times (99.28% of occurrences).
   - For products that went on backorder:
     - Occurred 13,927 times (0.72% of occurrences).
2. **When the deck risk state is 0 (indicating low deck risk):**
   - For products that did not go on backorder:
     - Occurred 1,482,779 times (99.22% of occurrences).
   - For products that went on backorder:
     - Occurred 11,704 times (0.78% of occurrences).
3. **When the OE constraint state is 0 (indicating no OE constraints):**
   - For products that did not go on backorder:
     - Occurred 1,915,672 times (99.28% of occurrences).
   - For products that went on backorder:

        - Occurred 13,972 times (0.72% of occurrences).
4. **When the PPAP risk state is 0 (indicating low PPAP risk):**
    - For products that did not go on backorder:
        - Occurred 1,685,534 times (99.30% of occurrences).
    - For products that went on backorder:
        - Occurred 11,850 times (0.70% of occurrences).
5. **When the stop auto-buy state is 0 (indicating no stop auto-buy):**
    - For products that did not go on backorder:
        - Occurred 69,966 times (99.18% of occurrences).
    - For products that went on backorder:
        - Occurred 578 times (0.82% of occurrences).
6. **When the rev stop state is 0 (indicating no rev stop):**
    - For products that did not go on backorder:
        - Occurred 1,915,120 times (99.28% of occurrences).
    - For products that went on backorder:
        - Occurred 13,977 times (0.72% of occurrences).

Numerical Features

We tried to plot box plots of the numerical features of the datasets to understand their distribution.

In descriptive statistics, a box plot, also known as a box-and-whisker plot, is a graphical representation commonly used in exploratory data analysis. It provides a visual summary of the distribution and skewness of numerical data.

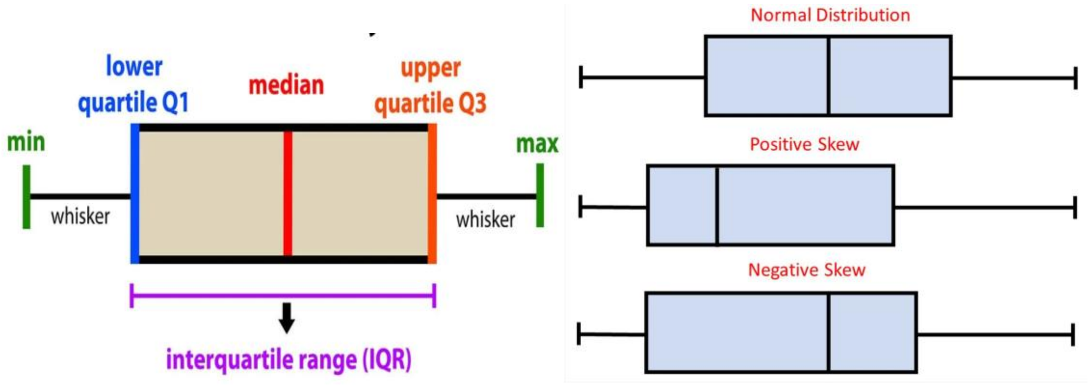


Fig 8: - Understanding Boxplot

Source: - https://www.simplypsychology.org/boxplots.html

These statistical measures are represented as follows:

- The box spans from the first quartile (Q1) to the third quartile (Q3), indicating the interquartile range (IQR).
- A line inside the box marks the median (Q2).
- The whiskers extend from the box to the minimum and maximum values within a specific range, typically 1.5 times the IQR.

- Outliers, if present, are shown as individual points beyond the whiskers.

When the whisker of a box plot is shorter towards the bottom of the plot, and the median is closer to the first quartile, the data is described as positively skewed. Conversely, the data is negatively skewed when the whisker is shorter towards the top of the plot, and the median is closer to the third quartile.

Box plots offer insights into the dataset's central tendency, spread, and variability, making them valuable tools for understanding data distributions and identifying potential outliers (Mcleod, 2023).

```
<AxesSubplot:>
```

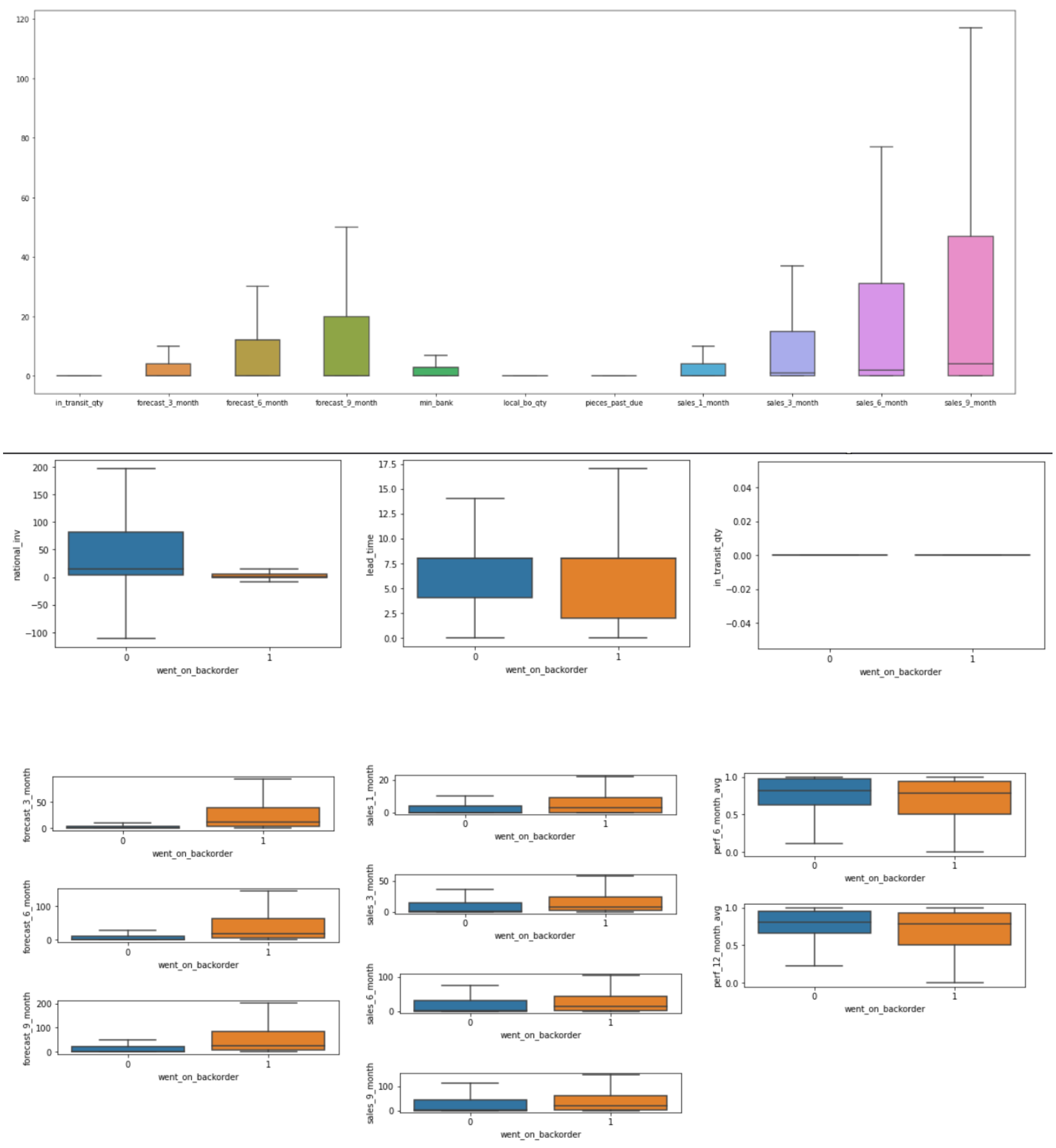


Fig 9: Box Plots of the numerical variables

As observed from all the plots above, the numerical features are highly positively skewed. This means that all the numeric variables have outliers, bringing the median closer to the first quartile than the mean. This helps us understand that outliers must be removed before proceeding further with the modelling. Furthermore, the additional subplots show how the individual numeric variables are distributed across our target variable.
After observing, we can say that all the features except the ‘perf_6_month_avg’ and ‘perf_12_month_avg’ are positively skewed.

- Bivariate Analysis
  This means that how two variables affect each other. In this case, both variables don't need to be dependent. But at least one of them will be dependent on the other. We have used a scatterplot to determine the relationship between two variables. Notably, the data is so imbalanced that we can hardly find any spots for the product with a backorder.

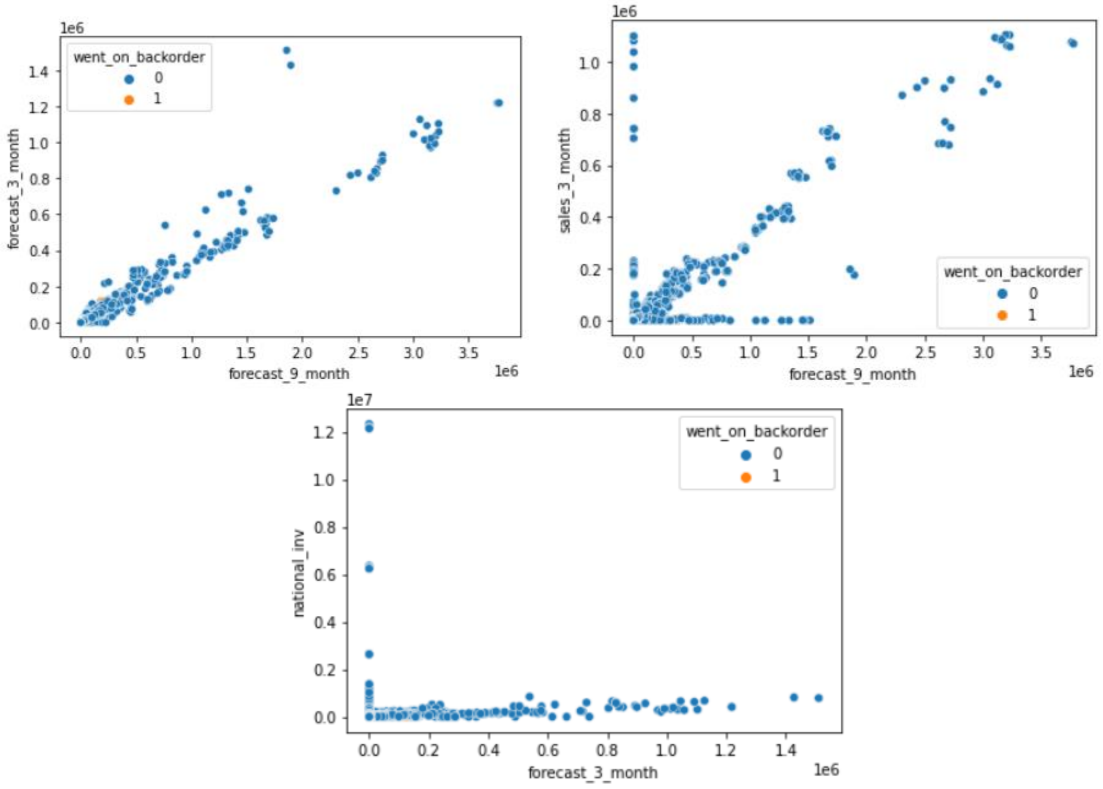
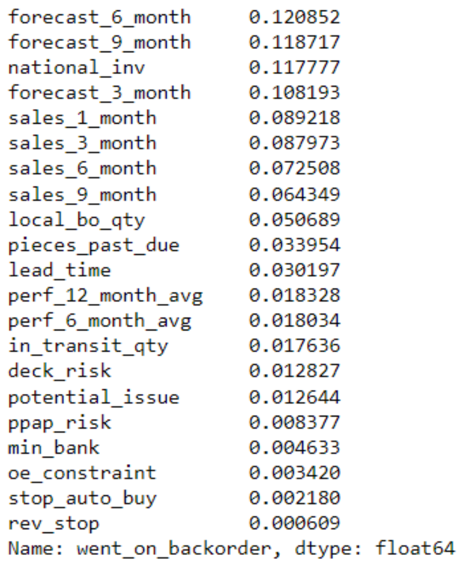


Fig 10: Scatter Plots of the numeric variables

4.4Data Preprocessing

- Null value removal
  This part of data preprocessing is performed before doing the exploratory data analysis to plot the box and scatter plots without the missing values.
- Outlier detection and Removal
  Outliers mainly affect the machine learning algorithms for predicting the appropriate values as they raise/lower the average value of any feature. Hence, it was removed before building a Machine Learning model.

- Normalising the numeric columns
  Normalising data involves scaling all the values in a dataset to fall within a specific range or distribution. This process ensures that all features contribute equally to the analysis and prevents features with larger scales from dominating the model. Normalisation typically involves scaling the data to have a mean of 0 and a standard deviation of 1 (z-score normalisation) or scaling the data to a range between 0 and 1 (min-max normalisation). This process can also be done before modelling rather than the exploratory data analysis process.
- Feature Selection
  I employed a correlation matrix with our target variable to select the relevant features that impacted whether the product went on backorder.



```
forecast_6_month     0.120852
forecast_9_month     0.118717
national_inv         0.117777
forecast_3_month     0.108193
sales_1_month        0.089218
sales_3_month        0.087973
sales_6_month        0.072508
sales_9_month        0.064349
local_bo_qty         0.050689
pieces_past_due      0.033954
lead_time            0.030197
perf_12_month_avg    0.018328
perf_6_month_avg     0.018034
in_transit_qty       0.017636
deck_risk            0.012827
potential_issue      0.012644
ppap_risk            0.008377
min_bank             0.004633
oe_constraint        0.003420
stop_auto_buy        0.002180
rev_stop             0.000609
Name: went_on_backorder, dtype: float64
```

Fig 11: Correlation values with the target variable

As we compare the correlation matrix values with our target variable, we select all the features for which this value is more significant than 0.1.

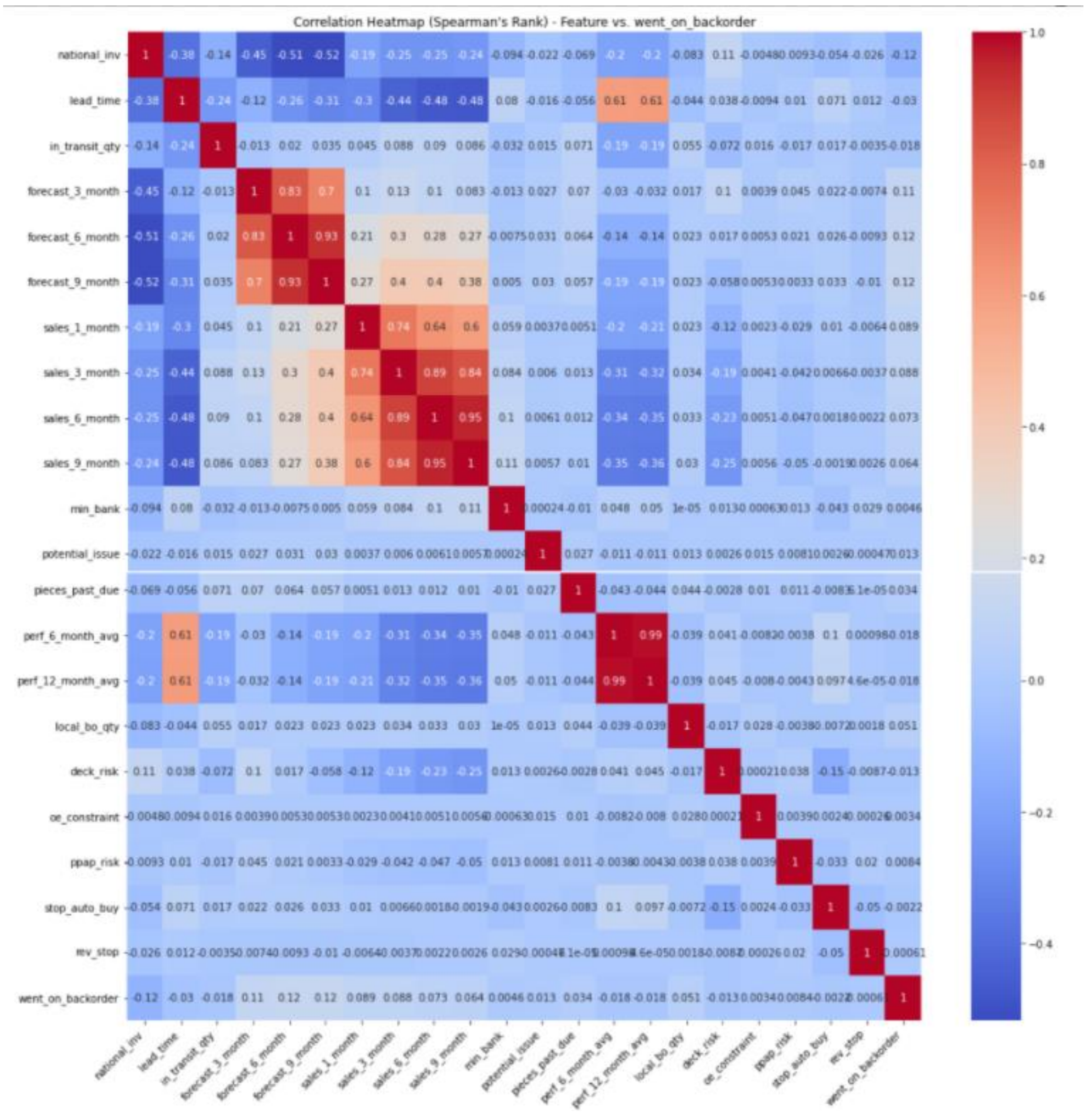


Fig 12: Correlation plot

4.3 Model Building

Nine models were utilised in this study. As we noticed earlier, our dataset is highly imbalanced against the favour of the 'Yes' state in the target variable. So before using the resampling techniques, I split the data into train and test data, which was 85% train and 15% test data. Below are the models used for this study:
Model 1: Random Over Sampler with Decision Tree Classifier
Model 2: Random Over Sampler with Random Forest Classifier
Model 3: XGBOOST Classifier with Random Oversampling
Model 4: Decision Tree Classifier with SMOTE Oversampling
Model 5 Smote Over sampler with Random Forest Classifier
Model 6: Smote Over sampler with XGBoost Classifier
Model 7: Smote Over sampler with Gradient Boosting Classifier
Model 8: Random Under Sampler with Decision Tree Classifier

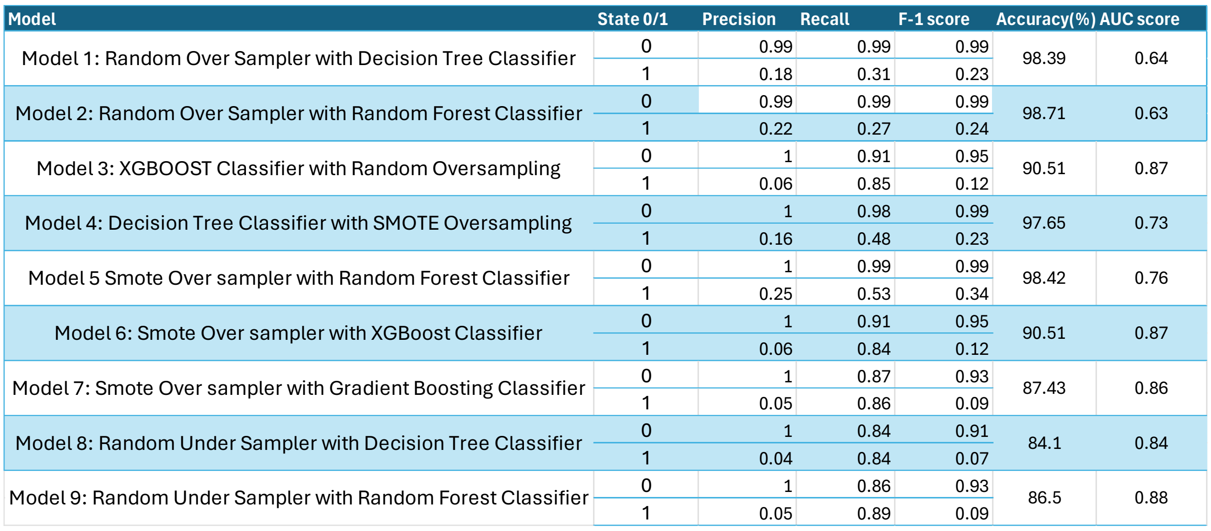
Model 9: Random Under Sampler with Random Forest Classifier

Here, two resampling techniques have been used.

Oversampling: The algorithm creates random data for the minority class to handle the class imbalance. In this case, the 'Yes' class was a minority, which was managed by the algorithm to create the balance.

Undersampling: This refers to the technique in which the majority class instances are removed to match the count with the minority class. In this study, we undersampled the dataset to balance the class.

5. Findings & Results



Model	State 0/1	Precision	Recall	F-1 score	Accuracy(%)	AUC score
Model 1: Random Over Sampler with Decision Tree Classifier	0	0.99	0.99	0.99	98.39	0.64
	1	0.18	0.31	0.23		
Model 2: Random Over Sampler with Random Forest Classifier	0	0.99	0.99	0.99	98.71	0.63
	1	0.22	0.27	0.24		
Model 3: XGBOOST Classifier with Random Oversampling	0	1	0.91	0.95	90.51	0.87
	1	0.06	0.85	0.12		
Model 4: Decision Tree Classifier with SMOTE Oversampling	0	1	0.98	0.99	97.65	0.73
	1	0.16	0.48	0.23		
Model 5 Smote Over sampler with Random Forest Classifier	0	1	0.99	0.99	98.42	0.76
	1	0.25	0.53	0.34		
Model 6: Smote Over sampler with XGBoost Classifier	0	1	0.91	0.95	90.51	0.87
	1	0.06	0.84	0.12		
Model 7: Smote Over sampler with Gradient Boosting Classifier	0	1	0.87	0.93	87.43	0.86
	1	0.05	0.86	0.09		
Model 8: Random Under Sampler with Decision Tree Classifier	0	1	0.84	0.91	84.1	0.84
	1	0.04	0.84	0.07		
Model 9: Random Under Sampler with Random Forest Classifier	0	1	0.86	0.93	86.5	0.88
	1	0.05	0.89	0.09		

Fig 13: Performance Metrics of different models

As we know, this is a classification problem where we must predict the class of the variable 'went_on_backorder'; its prediction has a concept called confusion matrix. This matrix has four components:

- True Positive: When the predicted data says that the product has a backorder and went on backorder in such a scenario, it is called a true positive.
- True Negative: When the prediction says that the product does not have backorder and comes true, it is called a true negative.
- False Positive: When the product does not have a backorder, but the prediction says otherwise, it is called a false positive.
- False Negative: When the product does have a backorder, but the prediction says it is not supposed to have it is known as a false negative (B, 2020).

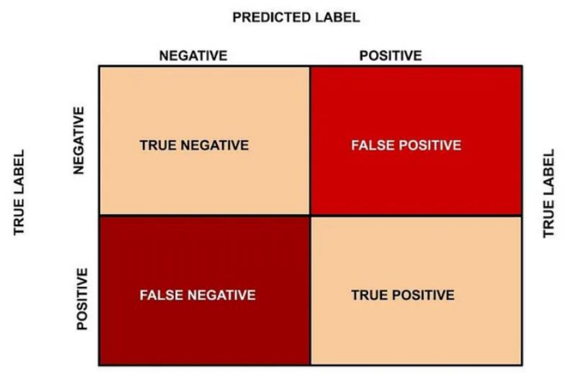


Fig 14: Confusion Matrix

Source: - https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

Generally, the false negative cases are considered more severe than the false positive. Let's understand with the below example.

- **False Positive Scenario:**
  Imagine a customer placing an order, and the ML algorithm falsely predicts a backorder due to low inventory levels. In response, the seller takes pre-emptive action to mitigate the perceived shortage, such as placing rush orders with suppliers or increasing production. However, since there was actually no shortage, these actions resulted in unnecessary overstocking. This surplus inventory incurs holding costs, increases the risk of obsolescence, reduces cash flow, and diminishes competitiveness in the market.
- **False Negative Scenario:**
  In contrast, consider a scenario where the ML algorithm incorrectly predicts that there won't be any inventory issues, leading the seller to assure the customer of swift delivery. However, in reality, there is a shortage of inventory. As a result, the seller fails to fulfil the promise, causing delays as they scramble to source the item from suppliers. This leads to increased wait times for the customer, decreased satisfaction, and potentially lower ratings and trust in the seller's service.
- **Assessment of Danger:**
  Both false positives and negatives pose significant risks to the seller's business. False positives result in overstocking, which ties up capital, increases holding costs, and reduces competitiveness. False negatives, on the other hand, lead to understocking, causing delays, customer dissatisfaction, and potential damage to the seller's reputation. However, false negatives may be considered more dangerous as they directly impact customer experience and satisfaction, leading to immediate consequences such as negative reviews and loss of trust. Additionally, false negatives may result in lost sales opportunities and long-term damage to customer relationships, making them potentially more detrimental to the seller's business in the long run.

As we observe the result from the table above, we notice that the accuracy for all the models is very high. However, as we discussed the class imbalance of the dataset, the accuracy is not the correct measure for comparing the model performance (B, 2020).

Hence, we check for other performance metrics such as precision score, recall score and f1 score to validate the model.

Let's analyse the models:

Model 1: Random Over Sampler with Decision Tree Classifier

While this model exhibits high precision and recall for state 0, its recall for state 1 is relatively low. The AUC score is also moderate at 0.64, indicating decent discrimination ability.

Model 2: Random Over Sampler with Random Forest Classifier

Like Model 1, this model shows high precision and recall for state 0 but lower values for state 1. The AUC score is also modest at 0.63.

Model 3: XGBOOST Classifier with Random Oversampling

This model demonstrates significantly higher recall for state one compared to the previous models, leading to a balanced F1 score. The AUC score is notably higher at 0.87, indicating better performance distinguishing between classes.

Model 4: Decision Tree Classifier with SMOTE Oversampling

While this model achieves high precision and recall for state 0, its recall for state 1 is notably higher than Models 1 and 2, leading to a more balanced F1 score. The AUC score is moderate at 0.73.

Model 5: SMOTE Over sampler with Random Forest Classifier

This model exhibits improved performance compared to Model 4, with higher precision, recall, and F1-score for both states. The AUC score is also higher at 0.76, indicating better discrimination ability.

Model 6: SMOTE Over sampler with XGBoost Classifier

Like Model 3, this model shows balanced performance across both states, with high recall for state 1. The AUC score matches Model 3 at 0.87, reflecting overall solid performance.

Model 7: SMOTE Over sampler with Gradient Boosting Classifier

This model demonstrates high recall for state 1, but its precision and AUC score are relatively lower than Models 3 and 6.

Model 8: Random Under Sampler with Decision Tree Classifier

While this model achieves high precision and recall for state 0, its recall for state 1 is relatively low. The AUC score is moderate at 0.84.

Model 9: Random Under Sampler with Random Forest Classifier

This model exhibits balanced performance across both states, with higher recall for state one compared to Models 1 and 2. The AUC score is notably higher at 0.88, indicating strong discrimination ability.

Considering the domain knowledge of backorder prediction in inventory management and the importance of correctly identifying instances of backorders (state 1), we prioritise models with higher recall for state one and higher AUC scores. Additionally, given the constraints on computing resources, we opt for models that do not require hyperparameter tuning.

Based on these criteria, Model 9: Random Under Sampler with Random Forest Classifier is the most suitable choice. It achieves a balanced performance across precision, recall, F1-score, and AUC scores.

6.Conclusion & Future Study

With adequate computing resources, this study can be further expanded by finely tuning the model's hyperparameters. Many similar studies have been done on the internet, including this process. One such study by Ntakolia C. et al. (2021) used this technique to improve the model further to balance the precision-recall score and develop a robust model for backorder prediction that helps efficient and smooth inventory management without facing any issues.

References

1. Albayrak Ünal, Ö., Erkayman, B. and Usanmaz, B. (2023) ‘Applications of artificial intelligence in inventory management: A systematic review of the literature’, Archives of Computational Methods in Engineering [Preprint]. doi:10.1007/s11831-022-09879-5.
2. Aro-Gordon, S. and Gupte, J.A. (2016) ‘Review of Modern Inventory Management Techniques’, The Global Journal of Business and Management, 1(2).
3. Bagchi, U. et al. (2007) ‘The Effect of RFID On Inventory Management and Control’, in Trends in Supply Chain Design and Management: Technologies and Methodologies. London, Greater London: Springer, pp. 71–92.
4. Benzidia, S. et al. (2023) ‘Big Data Analytics Capability in Healthcare Operations and Supply Chain Management: The Role of Green Process Innovation’, Annals of Operations Research, 333(2–3), pp. 1077–1101. doi:10.1007/s10479-022-05157-6.
5. B, H.N. (2020) Confusion matrix, accuracy, precision, recall, F1 score, Medium. Available at: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd (Accessed: 29 March 2024).
6. Chopra, S., Reinhardt, G. and Dada, M. (2004) ‘The effect of Lead time uncertainty on safety stocks’, Decision Sciences, 35(1), pp. 1–24. doi:10.1111/j.1540-5414.2004.02332.x.
7. Dhaliwal, N. et al. (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) [Preprint]. doi:10.1109/icacite57410.2023.10183082.
8. D'Souza, P., Guo, M., Wang, D., & Lee, I. (2013). Real-time Inventory Management with RFIDash. Dept. of CIS - Senior Design 2012-2013, Univ. of Pennsylvania, Philadelphia, PA.
9. Galli, L. et al. (2020) ‘Prescriptive analytics for inventory management in Health Care’, Journal of the Operational Research Society, 72(10), pp. 2211–2224. doi:10.1080/01605682.2020.1776167.
10. Hand, R. (2024) Learn the power of periodic automatic replenishment (PAR): Inventory management with ShipBob, ShipBob. Available at: https://www.shipbob.com/blog/periodic-automatic-replenishment/ (Accessed: 18 March 2024).
11. Hayes, A. (2024) Inventory management defined, plus methods and Techniques, Investopedia. Available at: https://www.investopedia.com/terms/i/inventory-management.asp (Accessed: 11 March 2024).
12. IBM (no date) What is inventory management?, IBM. Available at: https://www.ibm.com/topics/inventory-management (Accessed: 11 March 2024).

13. Islam, S. and Amin, S.H. (2020) ‘Prediction of probable backorder scenarios in the supply chain using distributed random forest and Gradient Boosting Machine Learning Techniques’, Journal of Big Data, 7(1). doi:10.1186/s40537-020-00345-2.
14. Jenkins, A. (2022) How to calculate safety stock, Oracle NetSuite. Available at: https://www.netsuite.com/portal/resource/articles/inventory-management/safety-stock.shtml (Accessed: 18 March 2024).
15. Kamble, N.M. and Tewari, A. (2022) ‘Effects of Predictive Data Analytics Using Big Data Analytics on Inventory Optimization in Landmark Group’, *WESTFORD RESEARCH JOURNAL*, 2(1), pp. 59–77.
16. Kesavan, S. (2022) What is reorder point?: Reorder point definition & formula, Essential Business Guides. Available at: https://www.zoho.com/inventory/guides/what-is-a-reorder-point.html#:~:text=What%20is%20a%20reorder%20point,t%20run%20out%20of%20stock. (Accessed: 18 March 2024).
17. Mcleod, S. (2023) Box plot explained: Interpretation, examples, & comparison, Simply Psychology. Available at: https://www.simplypsychology.org/boxplots.html (Accessed: 29 March 2024).
18. Meller, J., Taigel, F. and Pibernik, R. (2018) ‘Prescriptive analytics for inventory management: A comparison of new approaches’, SSRN Electronic Journal [Preprint]—doi:10.2139/ssrn.3229105.
19. Mishra, A. and Mohapatro, M. (2020) ‘Real-time RFID-based item tracking using IOT & Efficient Inventory Management Using Machine Learning’, 2020 IEEE 4th Conference on Information &amp; Communication Technology (CICT) [Preprint]. doi:10.1109/cict51604.2020.9312074.
20. Ntakolia, C. et al. (2021) ‘An explainable machine learning model for material backorder prediction in inventory management’, Sensors, 21(23), p. 7926. doi:10.3390/s21237926.
21. Peak Technologies (2023) The evolution of inventory management, Peak Technologies. Available at: https://www.peaktech.com/gb/blog/the-evolution-of-inventory-management/ (Accessed: 11 March 2024).
22. Silverman, L. (2018) Barcodes: A brief history, Tracking Software for Returnable Containers. Available at: https://corp.trackabout.com/blog/barcodes-brief-history (Accessed: 11 March 2024).
23. Singh, N. and Adhikari, D. (2023) ‘AI in inventory management: Applications, Challenges, and opportunities’, International Journal for Research in Applied Science and Engineering Technology, 11(11), pp. 2049–2053. doi:10.22214/ijraset.2023.57010.
24. Tarver, E. (2023) 17 Essential Inventory Management Techniques, Forbes. Available at: https://www.forbes.com/advisor/business/inventory-management-techniques/ (Accessed: 18 March 2024).

Appendix

BackOrder_Prediction

March 28, 2024

[1]:
```
# Importing required libraries
import logging
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import classification_report , roc_auc_score, roc_curve,␣
↪auc , precision_recall_curve, confusion_matrix, accuracy_score, f1_score,␣
↪recall_score
from sklearn.model_selection import cross_validate
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import Normalizer
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
#Setting up a log
logging.basicConfig(filename = 'logfile.log', level=logging.INFO, format␣
↪='%(asctime)s-%(levelname)s-%(message)s')
```

[3]:
```
pd.set_option('display.max_columns', None)
```

[4]:
```
# Loading the datasets
train = pd.read_csv("C:/Users/Aditya/OneDrive/Desktop/nmims/Data/
↪Kaggle_Training_Dataset_v2.csv", low_memory = False)
test = pd.read_csv("C:/Users/Aditya/OneDrive/Desktop/nmims/Data/
↪Kaggle_Test_Dataset_v2.csv", low_memory = False)
print(train.shape, test.shape)
```

```
(1687861, 23) (242076, 23)
```

0.1 Exploratory Data Analysis

[5]:
```
# let's make it into a single dataset
df = train.append(test)
df.reset_index(drop=True, inplace=True)
df
```

[5]:
```
                       sku  national_inv  lead_time  in_transit_qty  \
0                  1026827           0.0        NaN             0.0
1                  1043384           2.0        9.0             0.0
2                  1043696           2.0        NaN             0.0
3                  1043852           7.0        8.0             0.0
4                  1044048           8.0        NaN             0.0
...                    ...           ...        ...             ...
1929932            3526988          13.0       12.0             0.0
1929933            3526989          13.0       12.0             0.0
1929934            3526990          10.0       12.0             0.0
1929935            3526991        2913.0       12.0             0.0
1929936  (242075 rows)               NaN        NaN             NaN

         forecast_3_month  forecast_6_month  forecast_9_month  sales_1_month  \
0                     0.0               0.0               0.0            0.0
1                     0.0               0.0               0.0            0.0
2                     0.0               0.0               0.0            0.0
3                     0.0               0.0               0.0            0.0
4                     0.0               0.0               0.0            0.0
...                   ...               ...               ...            ...
1929932               0.0               0.0               0.0            0.0
1929933               0.0               0.0               0.0            0.0
1929934               0.0               0.0               0.0            0.0
1929935               0.0               0.0               0.0            0.0
1929936               NaN               NaN               NaN            NaN

         sales_3_month  sales_6_month  sales_9_month  min_bank  \
0                  0.0            0.0            0.0       0.0
1                  0.0            0.0            0.0       0.0
2                  0.0            0.0            0.0       0.0
3                  0.0            0.0            0.0       1.0
4                  0.0            0.0            4.0       2.0
...                ...            ...            ...       ...
1929932            0.0            0.0            0.0       1.0
1929933            0.0            0.0            0.0       1.0
1929934            0.0            0.0            0.0       1.0
1929935           30.0           88.0           88.0       4.0
1929936            NaN            NaN            NaN       NaN
```

```
        potential_issue  pieces_past_due  perf_6_month_avg  perf_12_month_avg  \
0                    No              0.0            -99.00             -99.00
1                    No              0.0              0.99               0.99
2                    No              0.0            -99.00             -99.00
3                    No              0.0              0.10               0.13
4                    No              0.0            -99.00             -99.00
...                 ...              ...               ...                ...
1929932              No              0.0              0.48               0.48
1929933              No              0.0              0.48               0.48
1929934              No              0.0              0.48               0.48
1929935              No              0.0              0.48               0.48
1929936             NaN              NaN               NaN                NaN

         local_bo_qty deck_risk oe_constraint ppap_risk stop_auto_buy  \
0                 0.0        No            No        No           Yes
1                 0.0        No            No        No           Yes
2                 0.0       Yes            No        No           Yes
3                 0.0        No            No        No           Yes
4                 0.0       Yes            No        No           Yes
...               ...       ...           ...       ...           ...
1929932           0.0       Yes            No        No           Yes
1929933           0.0       Yes            No        No           Yes
1929934           0.0       Yes            No        No           Yes
1929935           0.0       Yes            No        No           Yes
1929936           NaN       NaN           NaN       NaN           NaN

        rev_stop went_on_backorder
0             No                No
1             No                No
2             No                No
3             No                No
4             No                No
...          ...               ...
1929932       No                No
1929933       No                No
1929934       No                No
1929935       No                No
1929936      NaN               NaN

[1929937 rows x 23 columns]
```

[6]:
```
# as we observe the last row has all the values as NaN, we will remove the last␣
 ↪row
df = df.iloc[:-1]
df
```

[6]:

```
             sku  national_inv  lead_time  in_transit_qty  forecast_3_month  \
0        1026827           0.0        NaN             0.0               0.0
1        1043384           2.0        9.0             0.0               0.0
2        1043696           2.0        NaN             0.0               0.0
3        1043852           7.0        8.0             0.0               0.0
4        1044048           8.0        NaN             0.0               0.0
...          ...           ...        ...             ...               ...
1929931  3526987          12.0       12.0             0.0               0.0
1929932  3526988          13.0       12.0             0.0               0.0
1929933  3526989          13.0       12.0             0.0               0.0
1929934  3526990          10.0       12.0             0.0               0.0
1929935  3526991        2913.0       12.0             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0            0.0
1                     0.0               0.0            0.0            0.0
2                     0.0               0.0            0.0            0.0
3                     0.0               0.0            0.0            0.0
4                     0.0               0.0            0.0            0.0
...                   ...               ...            ...            ...
1929931               0.0               0.0            0.0            0.0
1929932               0.0               0.0            0.0            0.0
1929933               0.0               0.0            0.0            0.0
1929934               0.0               0.0            0.0            0.0
1929935               0.0               0.0            0.0           30.0

         sales_6_month  sales_9_month  min_bank potential_issue  \
0                  0.0            0.0       0.0              No
1                  0.0            0.0       0.0              No
2                  0.0            0.0       0.0              No
3                  0.0            0.0       1.0              No
4                  0.0            4.0       2.0              No
...                ...            ...       ...             ...
1929931            0.0            0.0       1.0              No
1929932            0.0            0.0       1.0              No
1929933            0.0            0.0       1.0              No
1929934            0.0            0.0       1.0              No
1929935           88.0           88.0       4.0              No

         pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
0                    0.0            -99.00             -99.00           0.0
1                    0.0              0.99               0.99           0.0
2                    0.0            -99.00             -99.00           0.0
3                    0.0              0.10               0.13           0.0
4                    0.0            -99.00             -99.00           0.0
...                  ...               ...                ...           ...
1929931              0.0              0.48               0.48           0.0
```

```
1929932             0.0             0.48              0.48           0.0
1929933             0.0             0.48              0.48           0.0
1929934             0.0             0.48              0.48           0.0
1929935             0.0             0.48              0.48           0.0

        deck_risk oe_constraint ppap_risk stop_auto_buy rev_stop  \
0              No            No        No           Yes       No
1              No            No        No           Yes       No
2             Yes            No        No           Yes       No
3              No            No        No           Yes       No
4             Yes            No        No           Yes       No
...           ...           ...       ...           ...      ...
1929931       Yes            No        No           Yes       No
1929932       Yes            No        No           Yes       No
1929933       Yes            No        No           Yes       No
1929934       Yes            No        No           Yes       No
1929935       Yes            No        No           Yes       No

        went_on_backorder
0                      No
1                      No
2                      No
3                      No
4                      No
...                   ...
1929931                No
1929932                No
1929933                No
1929934                No
1929935                No

[1929936 rows x 23 columns]
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1929936 entries, 0 to 1929935
Data columns (total 23 columns):
 #   Column             Dtype
---  ------             -----
 0   sku                object
 1   national_inv       float64
 2   lead_time          float64
 3   in_transit_qty     float64
 4   forecast_3_month   float64
 5   forecast_6_month   float64
 6   forecast_9_month   float64
 7   sales_1_month      float64
 8   sales_3_month      float64
 9   sales_6_month      float64
 10  sales_9_month      float64
 11  min_bank           float64
 12  potential_issue    object
 13  pieces_past_due    float64
 14  perf_6_month_avg   float64
 15  perf_12_month_avg  float64
 16  local_bo_qty       float64
 17  deck_risk          object
 18  oe_constraint      object
 19  ppap_risk          object
 20  stop_auto_buy      object
 21  rev_stop           object
 22  went_on_backorder  object
dtypes: float64(15), object(8)
memory usage: 338.7+ MB
```

[8]:
```
df.describe()
```

[8]:
```
        national_inv     lead_time  in_transit_qty  forecast_3_month  \
count   1.929935e+06  1.814318e+06    1.929935e+06      1.929935e+06
mean    4.965683e+02  7.878627e+00    4.306440e+01      1.785399e+02
std     2.957343e+04  7.054212e+00    1.295420e+03      5.108770e+03
min    -2.725600e+04  0.000000e+00    0.000000e+00      0.000000e+00
25%     4.000000e+00  4.000000e+00    0.000000e+00      0.000000e+00
50%     1.500000e+01  8.000000e+00    0.000000e+00      0.000000e+00
75%     8.000000e+01  9.000000e+00    0.000000e+00      4.000000e+00
max     1.233440e+07  5.200000e+01    4.894080e+05      1.510592e+06

        forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
count       1.929935e+06      1.929935e+06   1.929935e+06   1.929935e+06
mean        3.454659e+02      5.066067e+02   5.536816e+01   1.746639e+02
std         9.831562e+03      1.434543e+04   1.884377e+03   5.188856e+03
min         0.000000e+00      0.000000e+00   0.000000e+00   0.000000e+00
25%         0.000000e+00      0.000000e+00   0.000000e+00   0.000000e+00
50%         0.000000e+00      0.000000e+00   0.000000e+00   1.000000e+00
75%         1.200000e+01      2.000000e+01   4.000000e+00   1.500000e+01
max         2.461360e+06      3.777304e+06   7.417740e+05   1.105478e+06

        sales_6_month  sales_9_month      min_bank  pieces_past_due  \
count    1.929935e+06   1.929935e+06  1.929935e+06     1.929935e+06
mean     3.415653e+02   5.235771e+02  5.277637e+01     2.016193e+00
std      9.585030e+03   1.473327e+04  1.257968e+03     2.296112e+02
min      0.000000e+00   0.000000e+00  0.000000e+00     0.000000e+00
25%      0.000000e+00   0.000000e+00  0.000000e+00     0.000000e+00
50%      2.000000e+00   4.000000e+00  0.000000e+00     0.000000e+00
```

```
75%     3.100000e+01   4.700000e+01  3.000000e+00     0.000000e+00
max     2.146625e+06   3.205172e+06  3.133190e+05     1.464960e+05

       perf_6_month_avg  perf_12_month_avg  local_bo_qty
count      1.929935e+06       1.929935e+06  1.929935e+06
mean      -6.899870e+00      -6.462343e+00  6.537039e-01
std        2.659988e+01       2.588343e+01  3.543230e+01
min       -9.900000e+01      -9.900000e+01  0.000000e+00
25%        6.300000e-01       6.600000e-01  0.000000e+00
50%        8.200000e-01       8.100000e-01  0.000000e+00
75%        9.600000e-01       9.500000e-01  0.000000e+00
max        1.000000e+00       1.000000e+00  1.253000e+04
```

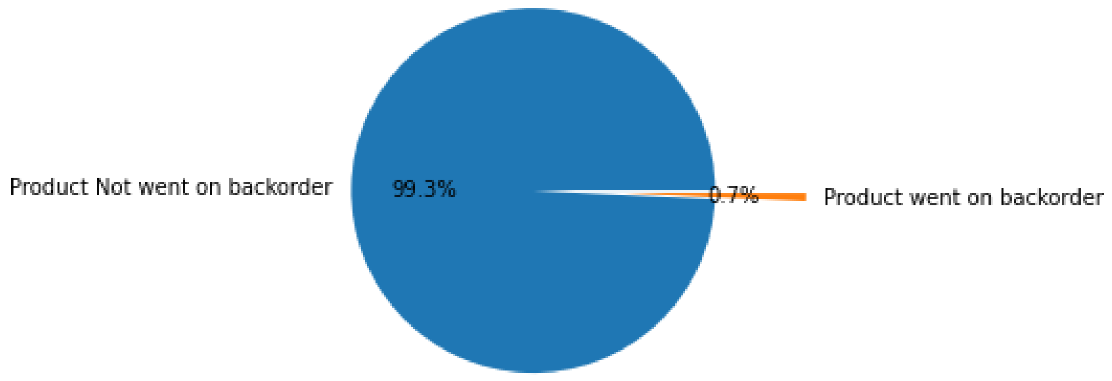
0.2 Distribution

[9]:
```
# As we want to predict if the product went on backorder we will check the data␣
↪distribution
p = df['went_on_backorder'].value_counts()
p
```

[9]:
```
No     1915954
Yes      13981
Name: went_on_backorder, dtype: int64
```

[10]:
```
# Let's make a pie chart to understand the distribution
fig, ax = plt.subplots()

labels = ['Product Not went on backorder',
         'Product went on backorder']
explode = (0, 0.5)
percentages = [p[0]*100/(p[1]+p[0]), p[1]*100/(p[1]+p[0])]

#Draw pie chart
ax.pie(percentages, labels=labels, shadow=False,explode=explode,autopct='%1.
↪1f%%')
```

[10]:
```
([<matplotlib.patches.Wedge at 0x274aeecea00>,
  <matplotlib.patches.Wedge at 0x274b0a65190>],
 [Text(-1.0997151375435508, 0.02503230428005669, 'Product Not went on
backorder'),
  Text(1.599585654182675, -0.036410643127857384, 'Product went on backorder')],
 [Text(-0.5998446204783003, 0.013653984152758194, '99.3%'),
  Text(1.099715137250589, -0.025032317150401946, '0.7%')])
```



As we can observe only **0.7%** of the data points represent the "Yes" observation. So, the data is highly imbalanced and needs to be balanced before it can be used to train a model.

0.3 Data Preprocessing

[11]:
```
logging.info("PREPROCESSING DATA....")
```

[12]:
```
#Removing null values
def remove_null(data):
    # Check if there are any missing values left
    if data.isnull().sum().sum() == 0:
        print('The dataset does not have missing values')
        null_col = []  # Initialize list to store columns with missing values
    else:
        # Initialize list to store columns with missing values
        null_col = []

        # Identify columns with missing values
        for col in data.columns:
            if data[col].isnull().sum() != 0:
                null_col.append(col)

                # Fill missing values based on column type
                if data[col].dtype in ['int64', 'float64']:  # Numeric columns
                    if data[col].mean() == data[col].median():
                        data[col].fillna(data[col].mean(), inplace=True)
                    else:
                        data[col].fillna(data[col].median(), inplace=True)
                else:  # Categorical columns
                    mode_val = data[col].mode()[0]
                    data[col].fillna(mode_val, inplace=True)
    # Return updated data
    return data
```

[13]:
```
remove_null(df)
```

[13]:
```
             sku  national_inv  lead_time  in_transit_qty  forecast_3_month  \
0        1026827           0.0        8.0             0.0               0.0
1        1043384           2.0        9.0             0.0               0.0
2        1043696           2.0        8.0             0.0               0.0
3        1043852           7.0        8.0             0.0               0.0
4        1044048           8.0        8.0             0.0               0.0
...          ...           ...        ...             ...               ...
1929931  3526987          12.0       12.0             0.0               0.0
1929932  3526988          13.0       12.0             0.0               0.0
1929933  3526989          13.0       12.0             0.0               0.0
1929934  3526990          10.0       12.0             0.0               0.0
1929935  3526991        2913.0       12.0             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0            0.0
1                     0.0               0.0            0.0            0.0
2                     0.0               0.0            0.0            0.0
3                     0.0               0.0            0.0            0.0
4                     0.0               0.0            0.0            0.0
...                   ...               ...            ...            ...
1929931               0.0               0.0            0.0            0.0
1929932               0.0               0.0            0.0            0.0
1929933               0.0               0.0            0.0            0.0
1929934               0.0               0.0            0.0            0.0
1929935               0.0               0.0            0.0           30.0

         sales_6_month  sales_9_month  min_bank potential_issue  \
0                  0.0            0.0       0.0              No
1                  0.0            0.0       0.0              No
2                  0.0            0.0       0.0              No
3                  0.0            0.0       1.0              No
4                  0.0            4.0       2.0              No
...                ...            ...       ...             ...
1929931            0.0            0.0       1.0              No
1929932            0.0            0.0       1.0              No
1929933            0.0            0.0       1.0              No
1929934            0.0            0.0       1.0              No
1929935           88.0           88.0       4.0              No

         pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
0                    0.0            -99.00             -99.00           0.0
```

```
1                   0.0           0.99           0.99          0.0
2                   0.0         -99.00         -99.00          0.0
3                   0.0           0.10           0.13          0.0
4                   0.0         -99.00         -99.00          0.0
...                 ...            ...            ...          ...
1929931             0.0           0.48           0.48          0.0
1929932             0.0           0.48           0.48          0.0
1929933             0.0           0.48           0.48          0.0
1929934             0.0           0.48           0.48          0.0
1929935             0.0           0.48           0.48          0.0

        deck_risk oe_constraint ppap_risk stop_auto_buy rev_stop  \
0              No            No        No           Yes       No
1              No            No        No           Yes       No
2             Yes            No        No           Yes       No
3              No            No        No           Yes       No
4             Yes            No        No           Yes       No
...           ...           ...       ...           ...      ...
1929931       Yes            No        No           Yes       No
1929932       Yes            No        No           Yes       No
1929933       Yes            No        No           Yes       No
1929934       Yes            No        No           Yes       No
1929935       Yes            No        No           Yes       No

        went_on_backorder
0                      No
1                      No
2                      No
3                      No
4                      No
...                   ...
1929931                No
1929932                No
1929933                No
1929934                No
1929935                No

[1929936 rows x 23 columns]
```

[14]:
```
# As we know sku or Stock Keeping Units is a unique identifier, its useless to␣
↪include this column for further analysis
df.drop('sku', axis=1, inplace=True)
```

[15]:
```
# For machine learning techniques only integer or float data types can be used.␣
↪Hence, we convert all the object type data to numeric format.
# First let's identify categorical columns
def identify_cat_cols(df):
  categorical_cols = []
  for col in df.columns:
    if df[col].dtype == 'object':
      categorical_cols.append(col)
  return categorical_cols
```

[16]:
```
cat_col=identify_cat_cols(df)
```

[17]:
```
#Let's convert the data type
def conv_col(cat_cols,df):
    for i in cat_cols:
        df[i].replace(['No','Yes'], [0,1], inplace=True)
    return df
```

[18]:
```
conv_col(cat_col,df)
```

[18]:
```
         national_inv  lead_time  in_transit_qty  forecast_3_month  \
0                 0.0        8.0             0.0               0.0
1                 2.0        9.0             0.0               0.0
2                 2.0        8.0             0.0               0.0
3                 7.0        8.0             0.0               0.0
4                 8.0        8.0             0.0               0.0
...               ...        ...             ...               ...
1929931          12.0       12.0             0.0               0.0
1929932          13.0       12.0             0.0               0.0
1929933          13.0       12.0             0.0               0.0
1929934          10.0       12.0             0.0               0.0
1929935        2913.0       12.0             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0            0.0
1                     0.0               0.0            0.0            0.0
2                     0.0               0.0            0.0            0.0
3                     0.0               0.0            0.0            0.0
4                     0.0               0.0            0.0            0.0
...                   ...               ...            ...            ...
1929931               0.0               0.0            0.0            0.0
1929932               0.0               0.0            0.0            0.0
1929933               0.0               0.0            0.0            0.0
1929934               0.0               0.0            0.0            0.0
1929935               0.0               0.0            0.0           30.0

         sales_6_month  sales_9_month  min_bank  potential_issue  \
0                  0.0            0.0       0.0                0
1                  0.0            0.0       0.0                0
2                  0.0            0.0       0.0                0
3                  0.0            0.0       1.0                0
```

```
4                   0.0            4.0       2.0                0
...                 ...            ...       ...              ...
1929931             0.0            0.0       1.0                0
1929932             0.0            0.0       1.0                0
1929933             0.0            0.0       1.0                0
1929934             0.0            0.0       1.0                0
1929935            88.0           88.0       4.0                0

         pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
0                    0.0            -99.00             -99.00           0.0
1                    0.0              0.99               0.99           0.0
2                    0.0            -99.00             -99.00           0.0
3                    0.0              0.10               0.13           0.0
4                    0.0            -99.00             -99.00           0.0
...                  ...               ...                ...           ...
1929931              0.0              0.48               0.48           0.0
1929932              0.0              0.48               0.48           0.0
1929933              0.0              0.48               0.48           0.0
1929934              0.0              0.48               0.48           0.0
1929935              0.0              0.48               0.48           0.0

         deck_risk  oe_constraint  ppap_risk  stop_auto_buy  rev_stop  \
0                0              0          0              1         0
1                0              0          0              1         0
2                1              0          0              1         0
3                0              0          0              1         0
4                1              0          0              1         0
...            ...            ...        ...            ...       ...
1929931          1              0          0              1         0
1929932          1              0          0              1         0
1929933          1              0          0              1         0
1929934          1              0          0              1         0
1929935          1              0          0              1         0

         went_on_backorder
0                        0
1                        0
2                        0
3                        0
4                        0
...                    ...
1929931                  0
1929932                  0
1929933                  0
1929934                  0
1929935                  0
```

```
[1929936 rows x 22 columns]
```

[19]:
```
df
```

[19]:
```
         national_inv  lead_time  in_transit_qty  forecast_3_month  \
0                 0.0        8.0             0.0               0.0
1                 2.0        9.0             0.0               0.0
2                 2.0        8.0             0.0               0.0
3                 7.0        8.0             0.0               0.0
4                 8.0        8.0             0.0               0.0
...               ...        ...             ...               ...
1929931          12.0       12.0             0.0               0.0
1929932          13.0       12.0             0.0               0.0
1929933          13.0       12.0             0.0               0.0
1929934          10.0       12.0             0.0               0.0
1929935        2913.0       12.0             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0            0.0
1                     0.0               0.0            0.0            0.0
2                     0.0               0.0            0.0            0.0
3                     0.0               0.0            0.0            0.0
4                     0.0               0.0            0.0            0.0
...                   ...               ...            ...            ...
1929931               0.0               0.0            0.0            0.0
1929932               0.0               0.0            0.0            0.0
1929933               0.0               0.0            0.0            0.0
1929934               0.0               0.0            0.0            0.0
1929935               0.0               0.0            0.0           30.0

         sales_6_month  sales_9_month  min_bank  potential_issue  \
0                  0.0            0.0       0.0                0
1                  0.0            0.0       0.0                0
2                  0.0            0.0       0.0                0
3                  0.0            0.0       1.0                0
4                  0.0            4.0       2.0                0
...                ...            ...       ...              ...
1929931            0.0            0.0       1.0                0
1929932            0.0            0.0       1.0                0
1929933            0.0            0.0       1.0                0
1929934            0.0            0.0       1.0                0
1929935           88.0           88.0       4.0                0

         pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
0                    0.0            -99.00             -99.00           0.0
1                    0.0              0.99               0.99           0.0
2                    0.0            -99.00             -99.00           0.0
```

```
3                   0.0          0.10          0.13          0.0
4                   0.0        -99.00        -99.00          0.0
...                 ...           ...           ...          ...
1929931             0.0          0.48          0.48          0.0
1929932             0.0          0.48          0.48          0.0
1929933             0.0          0.48          0.48          0.0
1929934             0.0          0.48          0.48          0.0
1929935             0.0          0.48          0.48          0.0

         deck_risk  oe_constraint  ppap_risk  stop_auto_buy  rev_stop  \
0                0              0          0              1         0
1                0              0          0              1         0
2                1              0          0              1         0
3                0              0          0              1         0
4                1              0          0              1         0
...            ...            ...        ...            ...       ...
1929931          1              0          0              1         0
1929932          1              0          0              1         0
1929933          1              0          0              1         0
1929934          1              0          0              1         0
1929935          1              0          0              1         0

         went_on_backorder
0                        0
1                        0
2                        0
3                        0
4                        0
...                    ...
1929931                  0
1929932                  0
1929933                  0
1929934                  0
1929935                  0

[1929936 rows x 22 columns]
```

0.4 Univariate Analysis

[20]:
```
df.dtypes
```

[20]:
```
national_inv          float64
lead_time             float64
in_transit_qty        float64
forecast_3_month      float64
forecast_6_month      float64
forecast_9_month      float64
```

```
sales_1_month        float64
sales_3_month        float64
sales_6_month        float64
sales_9_month        float64
min_bank             float64
potential_issue        int64
pieces_past_due      float64
perf_6_month_avg     float64
perf_12_month_avg    float64
local_bo_qty         float64
deck_risk              int64
oe_constraint          int64
ppap_risk              int64
stop_auto_buy          int64
rev_stop               int64
went_on_backorder      int64
dtype: object
```

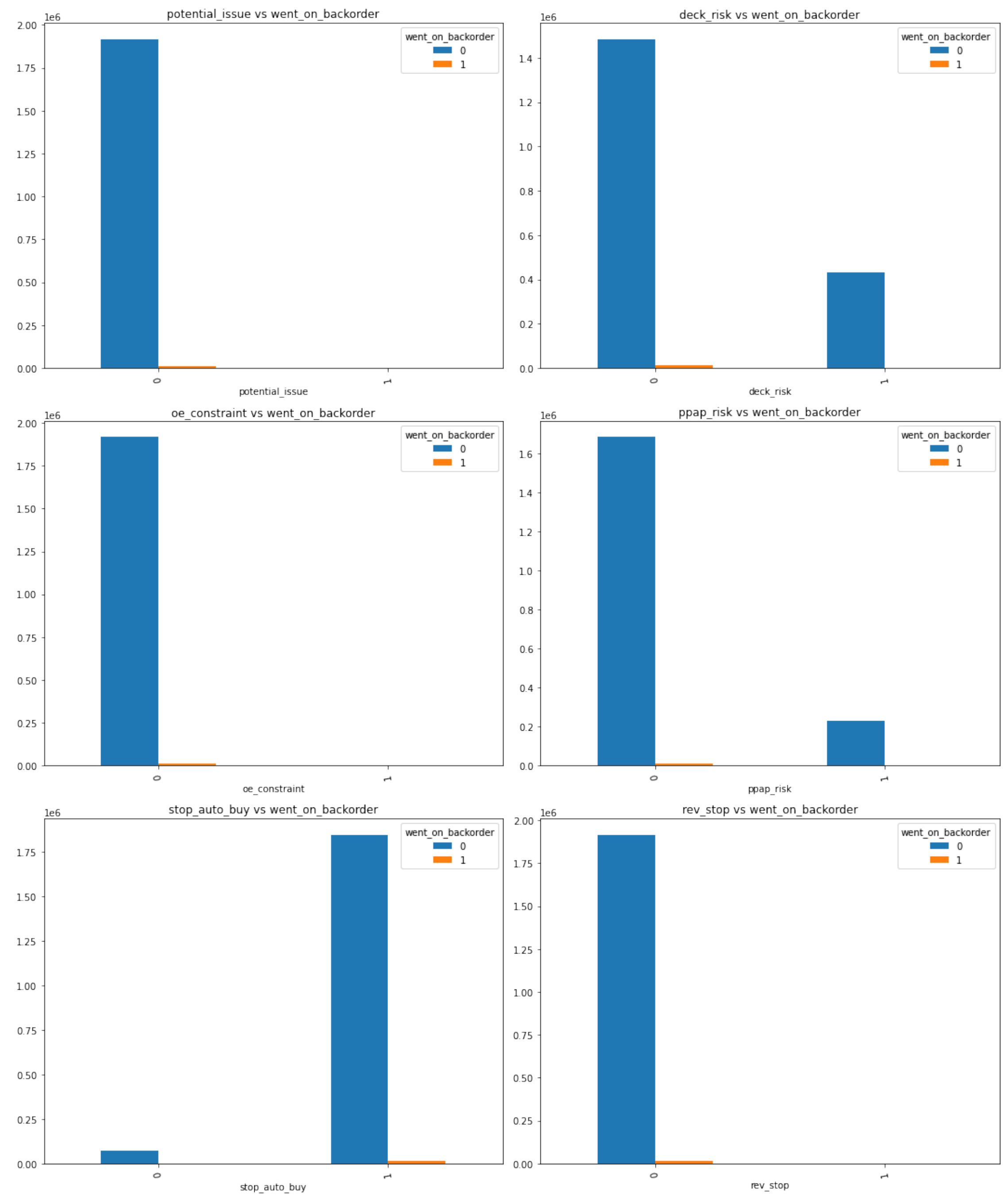
###Categorical_features

[21]:
```
def univariate_plot(data_df):
    # Select categorical columns except 'went_on_backorder'
    categorical_features = [col for col in data_df.
 ↪select_dtypes(include=['int64']).columns if col != 'went_on_backorder']

    num_features = len(categorical_features)
    fig, axes = plt.subplots(nrows=num_features//2 + num_features%2, ncols=2,␣
 ↪figsize=(15, 6*num_features//2))

    for i, feature in enumerate(categorical_features):
        row = i // 2
        col = i % 2
        # Group by feature and 'went_on_backorder', then count occurrences
        counts = data_df.groupby([feature, 'went_on_backorder']).size().
 ↪unstack().fillna(0)

        # Plot bar chart
        ax = counts.plot(kind='bar', rot=100, title=f'{feature} vs␣
 ↪went_on_backorder', ax=axes[row, col])

        # Print counts and percentages
        for index, row in counts.iterrows():
            print(f"{feature} state is {index}:")
            for column, value in row.iteritems():
                percentage = value * 100 / row.sum()
                print(f"   Product {column} {index} {value} times ({percentage:.
 ↪2f}%)")
```

```
    plt.tight_layout()
    plt.show()
```

[22]:
```
univariate_plot(df)
```

```
potential_issue state is 0:
   Product 0 0 1915020 times (99.28%)
   Product 1 0 13927 times (0.72%)
potential_issue state is 1:
   Product 0 1 935 times (94.54%)
   Product 1 1 54 times (5.46%)
deck_risk state is 0:
   Product 0 0 1482779 times (99.22%)
   Product 1 0 11704 times (0.78%)
deck_risk state is 1:
   Product 0 1 433176 times (99.48%)
   Product 1 1 2277 times (0.52%)
oe_constraint state is 0:
   Product 0 0 1915672 times (99.28%)
   Product 1 0 13972 times (0.72%)
oe_constraint state is 1:
   Product 0 1 283 times (96.92%)
   Product 1 1 9 times (3.08%)
ppap_risk state is 0:
   Product 0 0 1685534 times (99.30%)
   Product 1 0 11850 times (0.70%)
ppap_risk state is 1:
   Product 0 1 230421 times (99.08%)
   Product 1 1 2131 times (0.92%)
stop_auto_buy state is 0:
   Product 0 0 69966 times (99.18%)
   Product 1 0 578 times (0.82%)
stop_auto_buy state is 1:
   Product 0 1 1845989 times (99.28%)
   Product 1 1 13403 times (0.72%)
rev_stop state is 0:
   Product 0 0 1915120 times (99.28%)
   Product 1 0 13977 times (0.72%)
rev_stop state is 1:
   Product 0 1 835 times (99.52%)
   Product 1 1 4 times (0.48%)
```

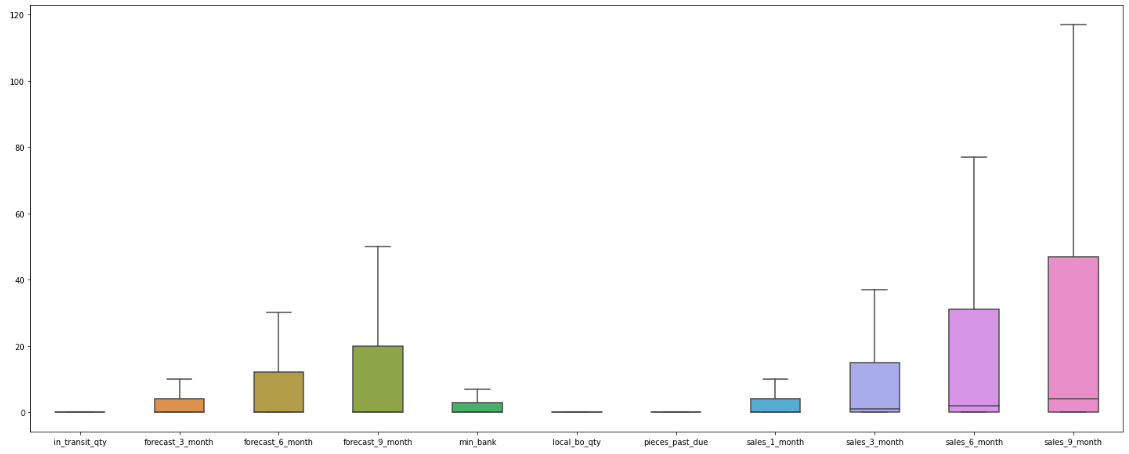


0.5 Numerical_Features

[23]:
```
num_data = [ 'in_transit_qty', 'forecast_3_month',
             'forecast_6_month', 'forecast_9_month', 'min_bank',
             'local_bo_qty', 'pieces_past_due', 'sales_1_month',
             'sales_3_month', 'sales_6_month', 'sales_9_month',]
```

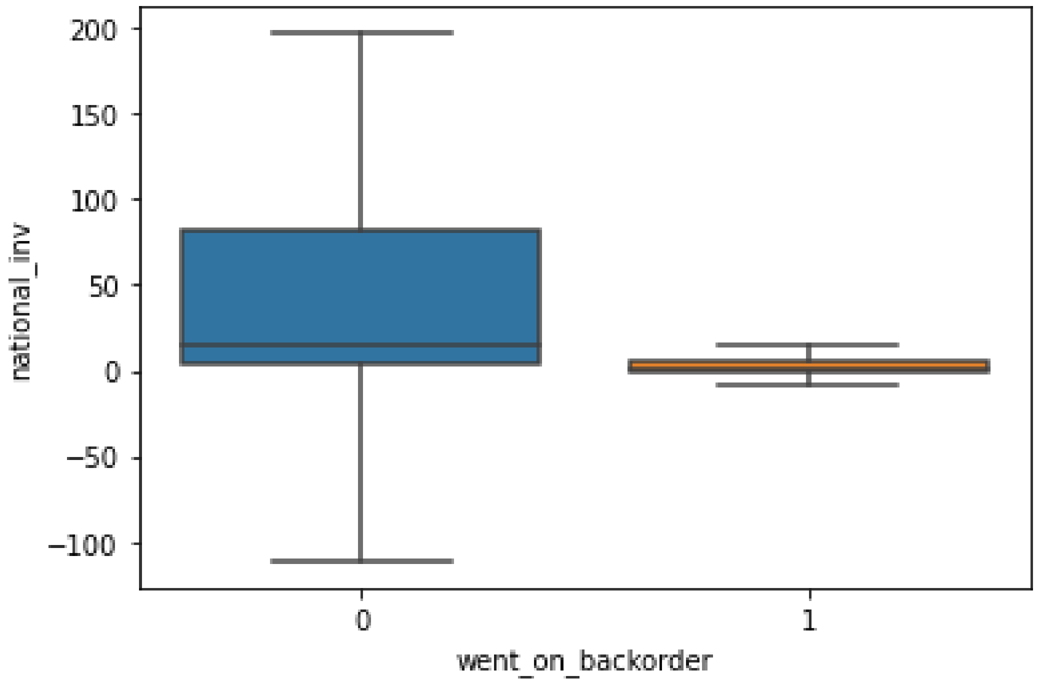
```
fig,ax = plt.subplots(figsize=(25,10),facecolor='white')
sns.boxplot(data=df[num_data],ax=ax,width=0.5,fliersize=4,showfliers=False)    #␣
↪will not Show the outliers beyond the caps.
```

```
[23]: <AxesSubplot:>
```



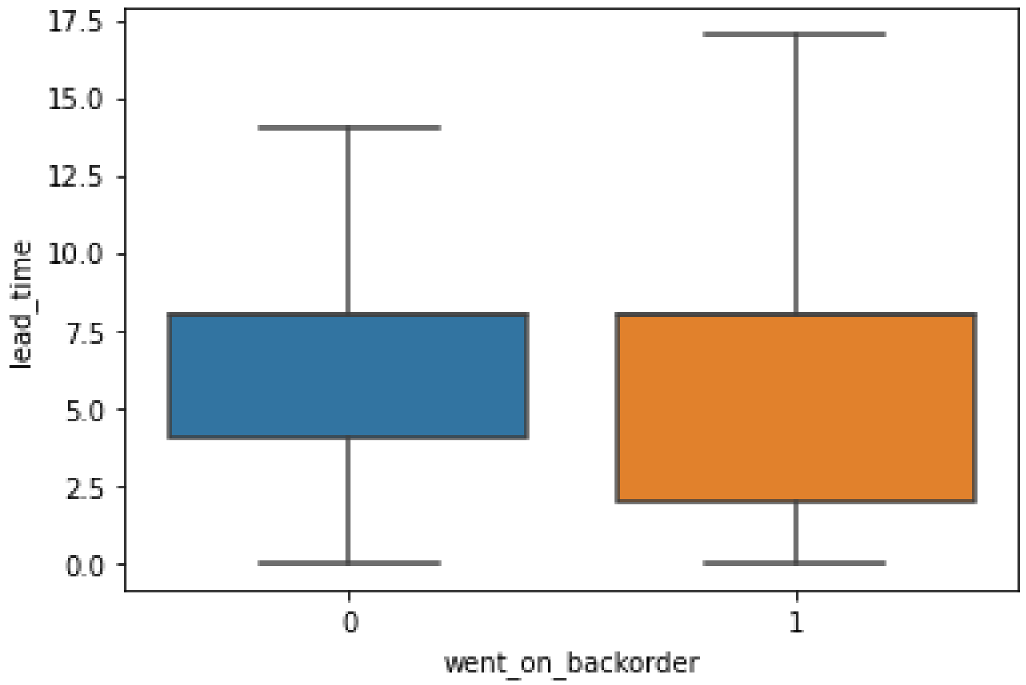
0.5.1 National_Inv

```
[24]: # Plotting Boxplot
sns.boxplot(  x=df["went_on_backorder"] ,y=df["national_inv"], showfliers=False)
plt.show()
```



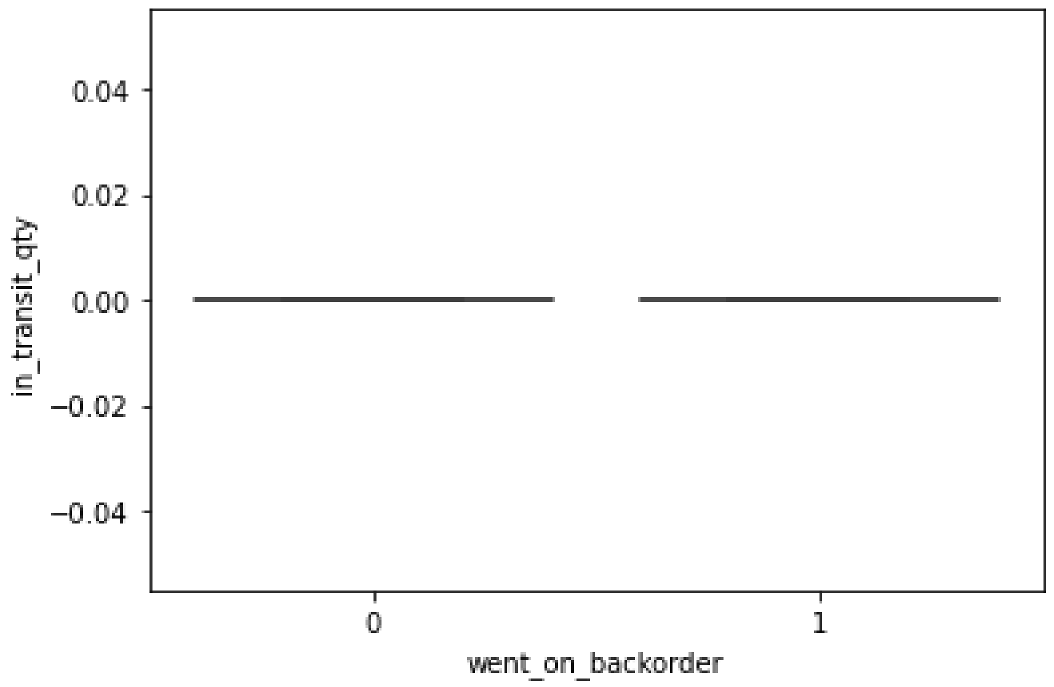
0.5.2 Lead_time

```
[25]: # Plotting Boxplot
      sns.boxplot(  x=df["went_on_backorder"] ,y=df["lead_time"], showfliers = False)
      plt.show()
```



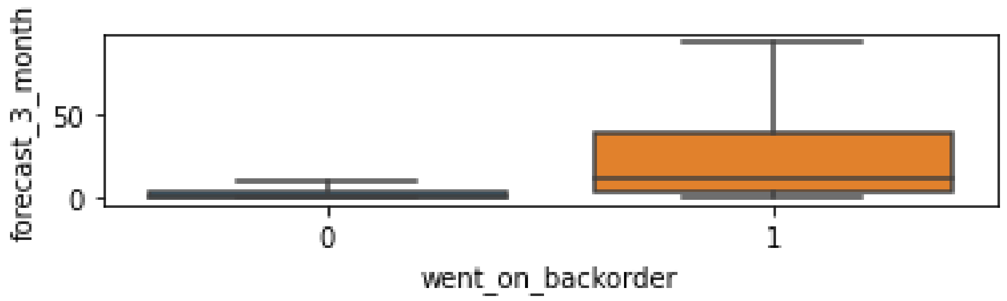
0.5.3 In_trsansit_qty

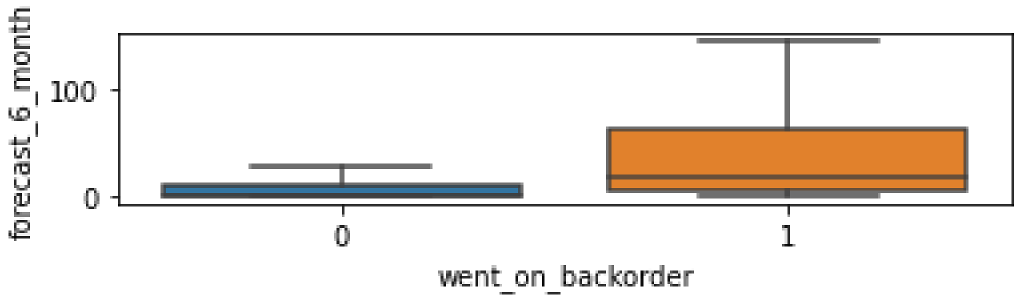
```
[26]:  # Plotting Boxplot
      sns.boxplot(  x=df["went_on_backorder"] ,y=df["in_transit_qty"], showfliers =␣
       ↪False)
      plt.show()
```

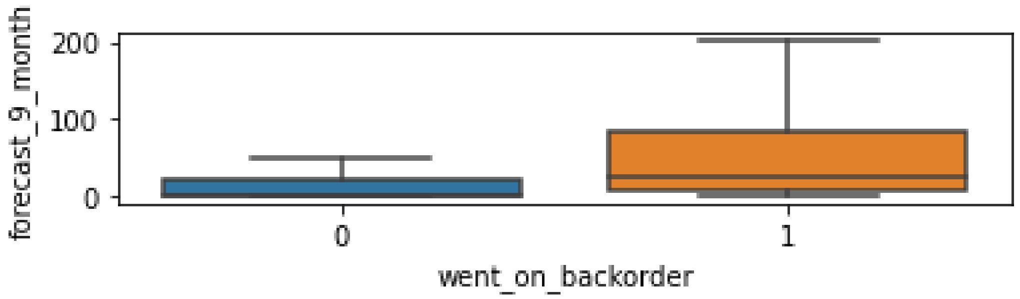


0.5.4 forecast for 3,6,9 months

[27]:
```
# Plotting Boxplot
plt.subplot(3,1,1)
sns.boxplot(  x=df["went_on_backorder"]␣
↪,y=df["forecast_3_month"],showfliers=False)
plt.show()

plt.subplot(3,1,2)
sns.boxplot(  x=df["went_on_backorder"]␣
↪,y=df["forecast_6_month"],showfliers=False)
plt.show()

plt.subplot(3,1,3)
sns.boxplot(  x=df["went_on_backorder"]␣
↪,y=df["forecast_9_month"],showfliers=False)
plt.show()
```

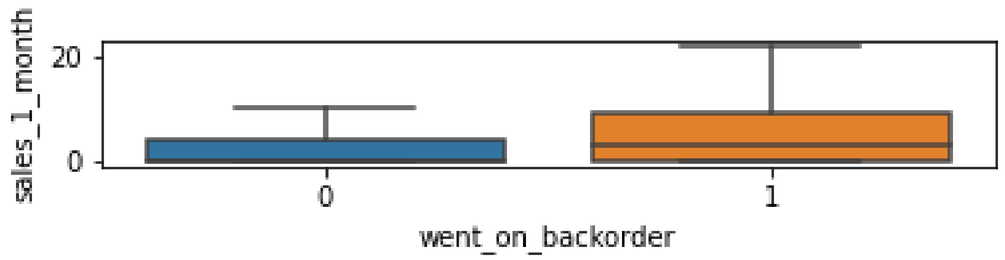


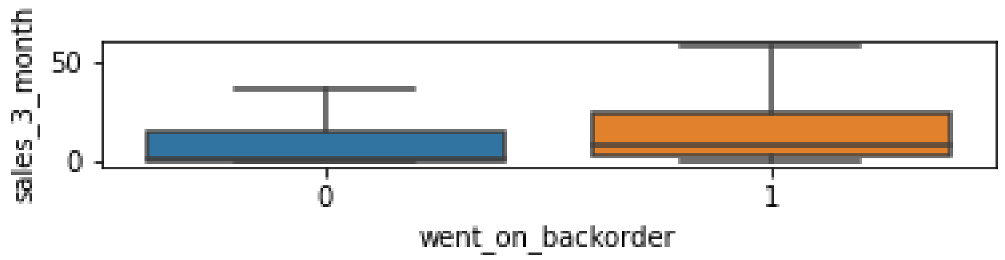


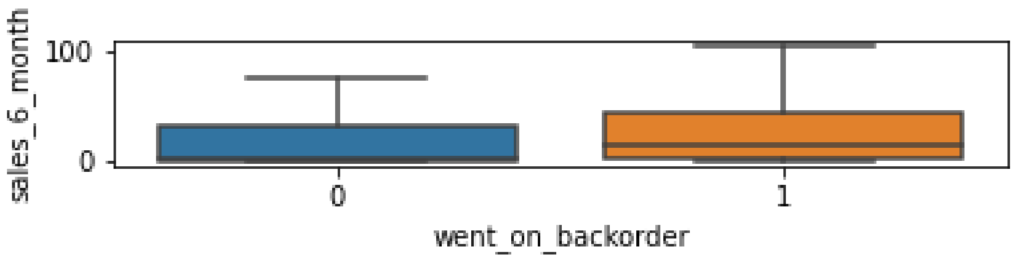


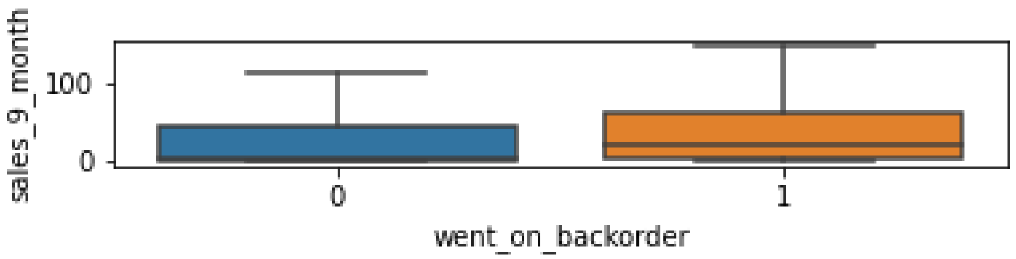
0.5.5 sales_1_month, sales_3_month, sales_6_month, sales_9_month

[28]:
```
plt.subplot(4,1,1)
sns.boxplot(  x=df["went_on_backorder"] ,y=df["sales_1_month"],showfliers=False)
plt.show()

plt.subplot(4,1,2)
sns.boxplot(  x=df["went_on_backorder"] ,y=df["sales_3_month"],showfliers=False)
plt.show()
plt.subplot(4,1,3)
sns.boxplot(  x=df["went_on_backorder"] ,y=df["sales_6_month"],showfliers=False)
plt.show()

plt.subplot(4,1,4)
sns.boxplot(  x=df["went_on_backorder"] ,y=df["sales_9_month"],showfliers=False)
plt.show()
```



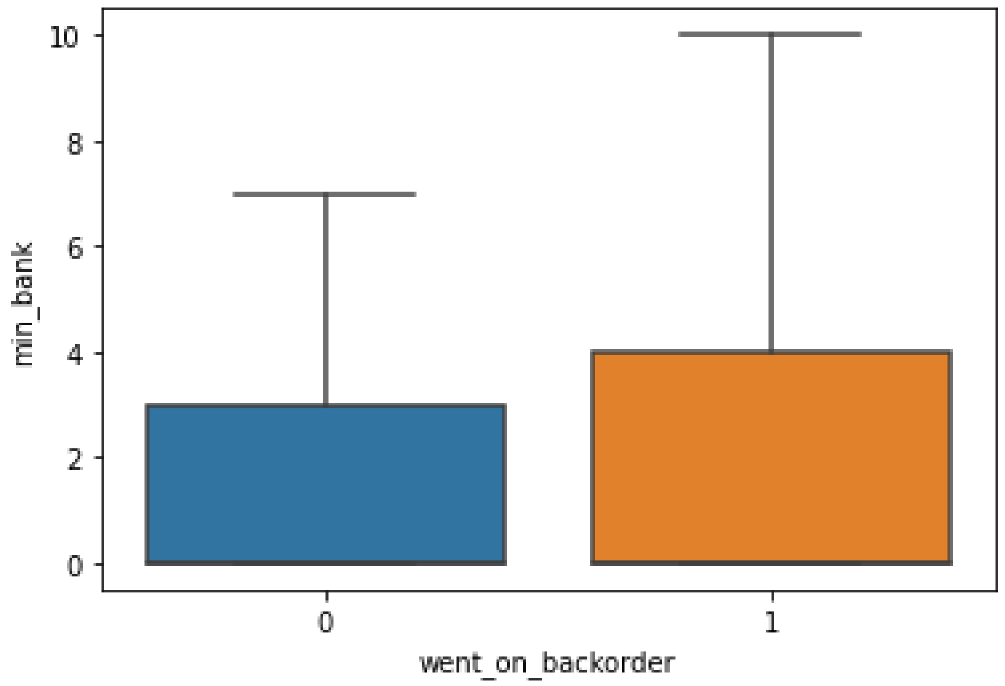






0.5.6 min_bank

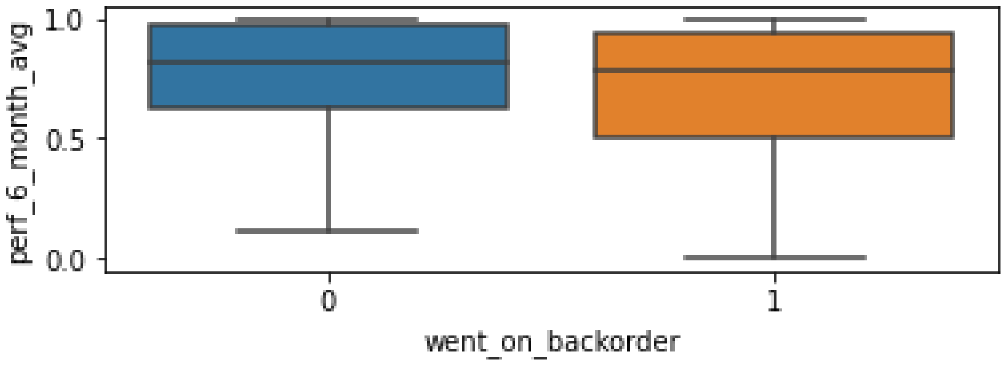
[29]:
```
sns.boxplot(  x=df["went_on_backorder"] ,y=df["min_bank"],showfliers=False)
plt.show()
```

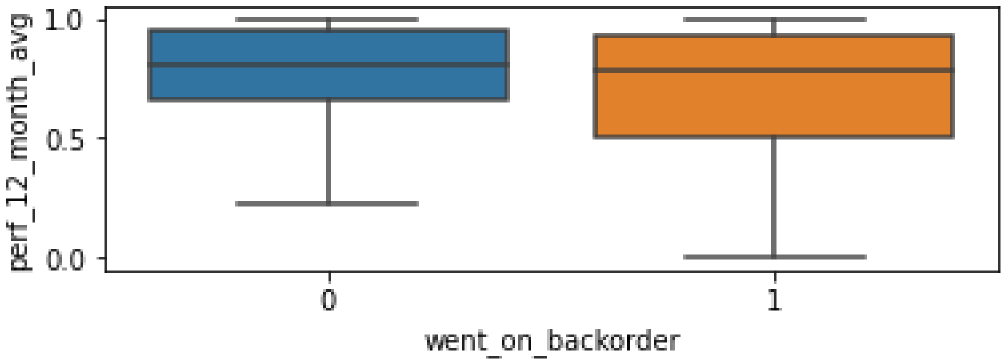


0.5.7 Performance_AVG 6 months, 9months

[30]:
```
plt.subplot(2,1,1)
sns.boxplot(  x=df["went_on_backorder"]␣
↪,y=df["perf_6_month_avg"],showfliers=False)
plt.show()

plt.subplot(2,1,2)
```

```
sns.boxplot(  x=df["went_on_backorder"]␣
↪,y=df["perf_12_month_avg"],showfliers=False)
plt.show()
```



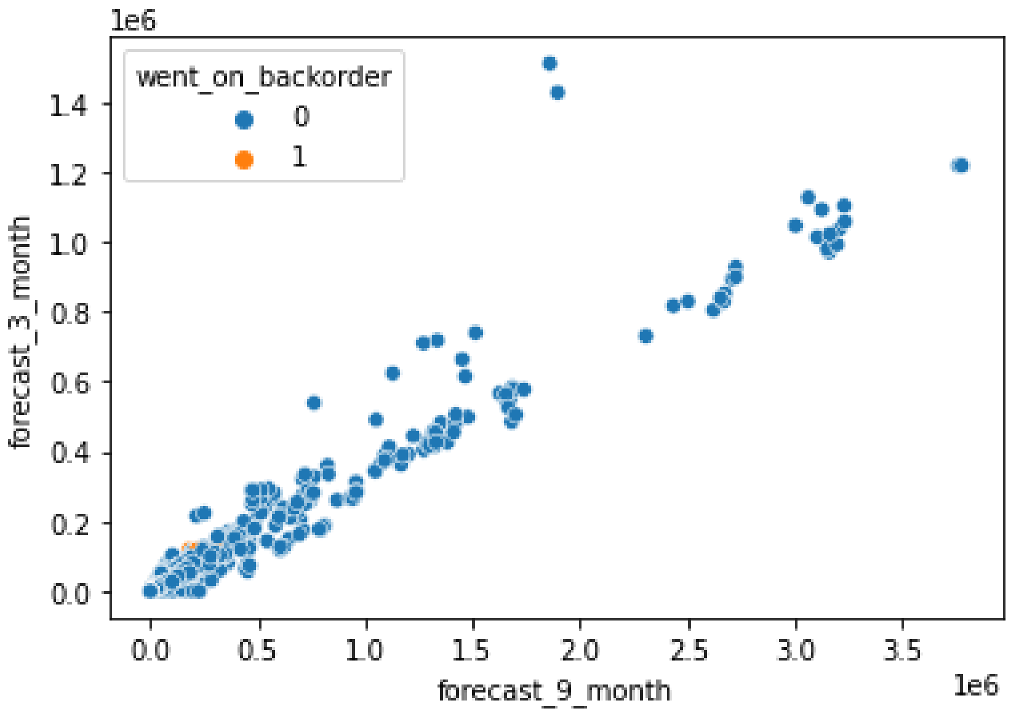


0.6 Bivariate_Analysis

We will plot scatter plots using any two variables as x & y and hue as went_on_backorder

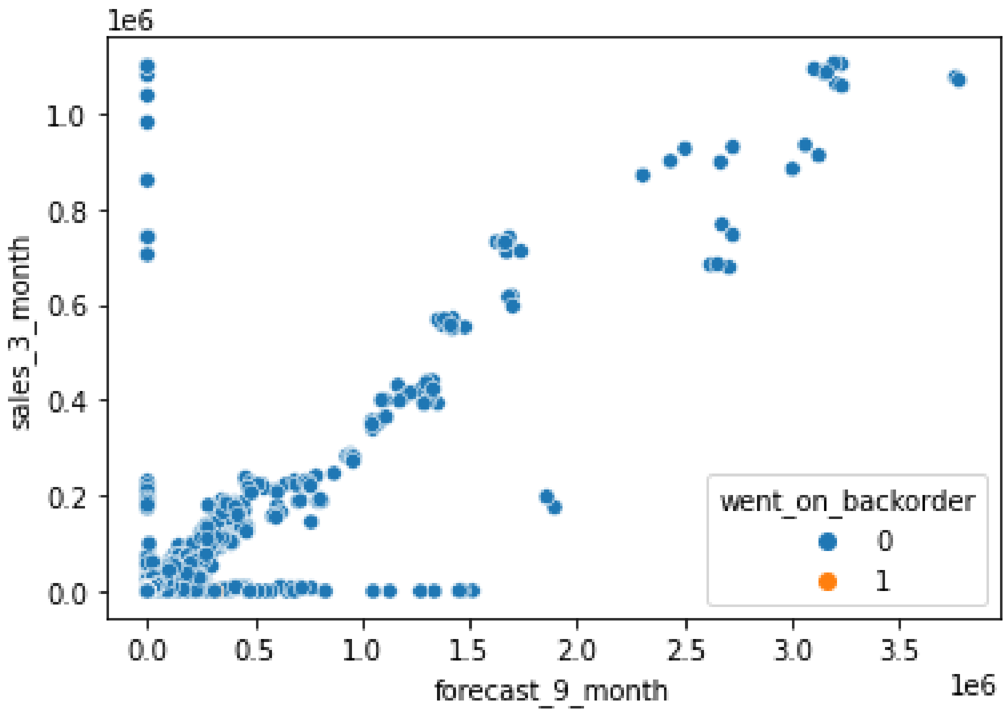
```
[31]: sns.scatterplot(x = df.forecast_9_month,y = df.forecast_3_month,hue=df.
       ↪went_on_backorder)
```

```
[31]: <AxesSubplot:xlabel='forecast_9_month', ylabel='forecast_3_month'>
```

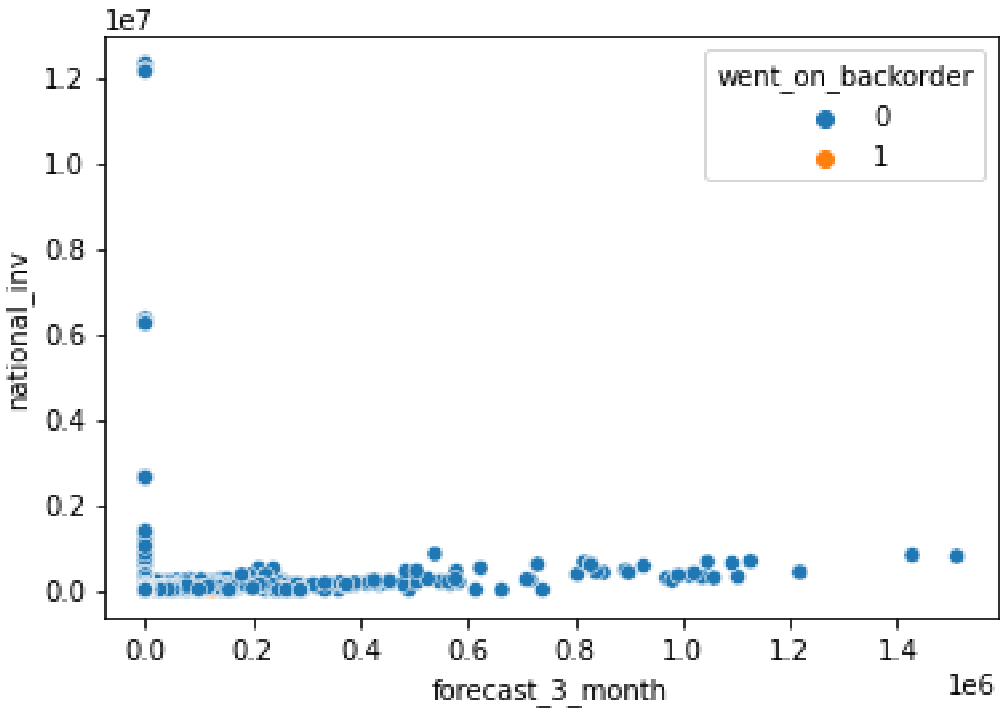


[32]:
```
sns.scatterplot(x = df.forecast_9_month,y = df.sales_3_month,hue=df.
 ↪went_on_backorder)
```

[32]:
```
<AxesSubplot:xlabel='forecast_9_month', ylabel='sales_3_month'>
```

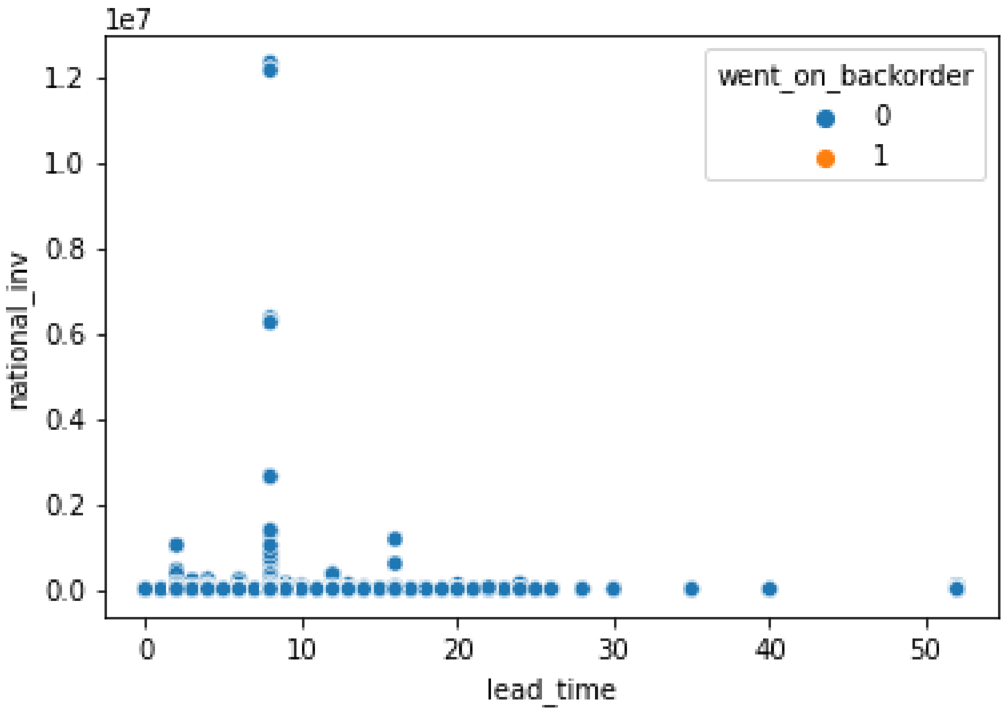


```
[33]: sns.scatterplot(x = df.forecast_3_month,y = df.national_inv,hue=df.
       ↪went_on_backorder)

[33]: <AxesSubplot:xlabel='forecast_3_month', ylabel='national_inv'>
```



```
[34]: sns.scatterplot(x = df.lead_time,y = df.national_inv,hue=df.went_on_backorder)
```

```
[34]: <AxesSubplot:xlabel='lead_time', ylabel='national_inv'>
```



0.7 Outlier detection & Removal

[35]:
```
import pandas as pd

def impute_outliers(data_series):
    # Calculate quartiles and interquartile range
    Q1 = data_series.quantile(0.25)
    Q3 = data_series.quantile(0.75)
    IQR = Q3 - Q1
    # Calculate lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Create a new series to store the imputed values
    imputed_series = pd.Series(index=data_series.index)

    # Iterate over the data and impute outliers
    for i, value in data_series.items():

        if lower_bound == upper_bound:
          imputed_series[i]=value
        elif value < lower_bound:
            imputed_series[i] = lower_bound
        elif value > upper_bound:
            imputed_series[i] = upper_bound
        else:
            imputed_series[i] = value

    print("The new series is without outliers")
    return imputed_series
```

[36]:
```
df['national_inv']=impute_outliers(df['national_inv'])
df['lead_time']=impute_outliers(df['lead_time'])
df['in_transit_qty']=impute_outliers(df['in_transit_qty'])
df['forecast_3_month']=impute_outliers(df['forecast_3_month'])
df['forecast_6_month']=impute_outliers(df['forecast_6_month'])
df['forecast_9_month']=impute_outliers(df['forecast_9_month'])
df['sales_1_month']=impute_outliers(df['sales_1_month'])
df['sales_3_month']=impute_outliers(df['sales_3_month'])
df['sales_6_month']=impute_outliers(df['sales_6_month'])
df['sales_9_month']=impute_outliers(df['sales_9_month'])
df['min_bank']=impute_outliers(df['min_bank'])
df['pieces_past_due']=impute_outliers(df['pieces_past_due'])
df['perf_6_month_avg']=impute_outliers(df['perf_6_month_avg'])
```

```
df['perf_12_month_avg']=impute_outliers(df['perf_12_month_avg'])
df['local_bo_qty']=impute_outliers(df['local_bo_qty'])
df
```

```
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
The new series is without outliers
```

```
[36]:          national_inv  lead_time  in_transit_qty  forecast_3_month  \
      0                 0.0        8.0             0.0               0.0
      1                 2.0        9.0             0.0               0.0
      2                 2.0        8.0             0.0               0.0
      3                 7.0        8.0             0.0               0.0
      4                 8.0        8.0             0.0               0.0
      ...               ...        ...             ...               ...
      1929931          12.0       12.0             0.0               0.0
      1929932          13.0       12.0             0.0               0.0
      1929933          13.0       12.0             0.0               0.0
      1929934          10.0       12.0             0.0               0.0
      1929935         194.0       12.0             0.0               0.0

               forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
      0                     0.0               0.0            0.0            0.0
      1                     0.0               0.0            0.0            0.0
      2                     0.0               0.0            0.0            0.0
      3                     0.0               0.0            0.0            0.0
      4                     0.0               0.0            0.0            0.0
      ...                   ...               ...            ...            ...
      1929931               0.0               0.0            0.0            0.0
      1929932               0.0               0.0            0.0            0.0
      1929933               0.0               0.0            0.0            0.0
      1929934               0.0               0.0            0.0            0.0
      1929935               0.0               0.0            0.0           30.0

               sales_6_month  sales_9_month  min_bank  potential_issue  \
      0                 0.0           0.0      0.0              0
      1                 0.0           0.0      0.0              0
      2                 0.0           0.0      0.0              0
      3                 0.0           0.0      1.0              0
      4                 0.0           4.0      2.0              0
      ...               ...           ...      ...            ...
      1929931           0.0           0.0      1.0              0
      1929932           0.0           0.0      1.0              0
      1929933           0.0           0.0      1.0              0
      1929934           0.0           0.0      1.0              0
      1929935          77.5          88.0      4.0              0

               pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
      0                    0.0             0.135              0.225           0.0
      1                    0.0             0.990              0.990           0.0
      2                    0.0             0.135              0.225           0.0
      3                    0.0             0.135              0.225           0.0
      4                    0.0             0.135              0.225           0.0
      ...                  ...               ...                ...           ...
      1929931              0.0             0.480              0.480           0.0
      1929932              0.0             0.480              0.480           0.0
      1929933              0.0             0.480              0.480           0.0
      1929934              0.0             0.480              0.480           0.0
      1929935              0.0             0.480              0.480           0.0

               deck_risk  oe_constraint  ppap_risk  stop_auto_buy  rev_stop  \
      0                0              0          0              1         0
      1                0              0          0              1         0
      2                1              0          0              1         0
      3                0              0          0              1         0
      4                1              0          0              1         0
      ...            ...            ...        ...            ...       ...
      1929931          1              0          0              1         0
      1929932          1              0          0              1         0
      1929933          1              0          0              1         0
      1929934          1              0          0              1         0
      1929935          1              0          0              1         0

               went_on_backorder
      0                        0
      1                        0
      2                        0
      3                        0
      4                        0
      ...                    ...
      1929931                  0
      1929932                  0
```

```
1929933                  0
1929934                  0
1929935                  0

[1929936 rows x 22 columns]
```

0.8 Normalisation

[40]:
```
# Select numerical columns from the DataFrame
numerical_cols = df.select_dtypes(include=['float64'])

# Initialize the Normalizer
normalizer = Normalizer()

# Fit and transform the data
df_normalized = pd.DataFrame(normalizer.fit_transform(numerical_cols),␣
 ↪columns=numerical_cols.columns)

# Combine normalized numerical columns with non-numerical columns
for col in df.columns:
    if col not in numerical_cols.columns:
        df_normalized[col] = df[col]

# Reorder columns to match the original DataFrame
df = df_normalized[df.columns]
df
```

[40]:
```
         national_inv  lead_time  in_transit_qty  forecast_3_month  \
0            0.000000   0.999463             0.0               0.0
1            0.214472   0.965122             0.0               0.0
2            0.242413   0.969652             0.0               0.0
3            0.655412   0.749042             0.0               0.0
4            0.657443   0.657443             0.0               0.0
...               ...        ...             ...               ...
1929931      0.705320   0.705320             0.0               0.0
1929932      0.733095   0.676703             0.0               0.0
1929933      0.733095   0.676703             0.0               0.0
1929934      0.638277   0.765932             0.0               0.0
1929935      0.847115   0.052399             0.0               0.0

         forecast_6_month  forecast_9_month  sales_1_month  sales_3_month  \
0                     0.0               0.0            0.0       0.000000
1                     0.0               0.0            0.0       0.000000
2                     0.0               0.0            0.0       0.000000
3                     0.0               0.0            0.0       0.000000
4                     0.0               0.0            0.0       0.000000
...                   ...               ...            ...            ...
```

```
1929931              0.0              0.0             0.0       0.000000
1929932              0.0              0.0             0.0       0.000000
1929933              0.0              0.0             0.0       0.000000
1929934              0.0              0.0             0.0       0.000000
1929935              0.0              0.0             0.0       0.130997

         sales_6_month  sales_9_month  min_bank  potential_issue  \
0              0.00000       0.000000  0.000000                0
1              0.00000       0.000000  0.000000                0
2              0.00000       0.000000  0.000000                0
3              0.00000       0.000000  0.093630                0
4              0.00000       0.328722  0.164361                0
...                ...            ...       ...              ...
1929931        0.00000       0.000000  0.058777                0
1929932        0.00000       0.000000  0.056392                0
1929933        0.00000       0.000000  0.056392                0
1929934        0.00000       0.000000  0.063828                0
1929935        0.33841       0.384259  0.017466                0

         pieces_past_due  perf_6_month_avg  perf_12_month_avg  local_bo_qty  \
0                    0.0          0.016866           0.028110           0.0
1                    0.0          0.106163           0.106163           0.0
2                    0.0          0.016363           0.027271           0.0
3                    0.0          0.012640           0.021067           0.0
4                    0.0          0.011094           0.018491           0.0
...                  ...               ...                ...           ...
1929931              0.0          0.028213           0.028213           0.0
1929932              0.0          0.027068           0.027068           0.0
1929933              0.0          0.027068           0.027068           0.0
1929934              0.0          0.030637           0.030637           0.0
1929935              0.0          0.002096           0.002096           0.0

         deck_risk  oe_constraint  ppap_risk  stop_auto_buy  rev_stop  \
0                0              0          0              1         0
1                0              0          0              1         0
2                1              0          0              1         0
3                0              0          0              1         0
4                1              0          0              1         0
...            ...            ...        ...            ...       ...
1929931          1              0          0              1         0
1929932          1              0          0              1         0
1929933          1              0          0              1         0
1929934          1              0          0              1         0
1929935          1              0          0              1         0

         went_on_backorder
0                        0
```

```
1                           0
2                           0
3                           0
4                           0
...                       ...
1929931                     0
1929932                     0
1929933                     0
1929934                     0
1929935                     0

[1929936 rows x 22 columns]
```

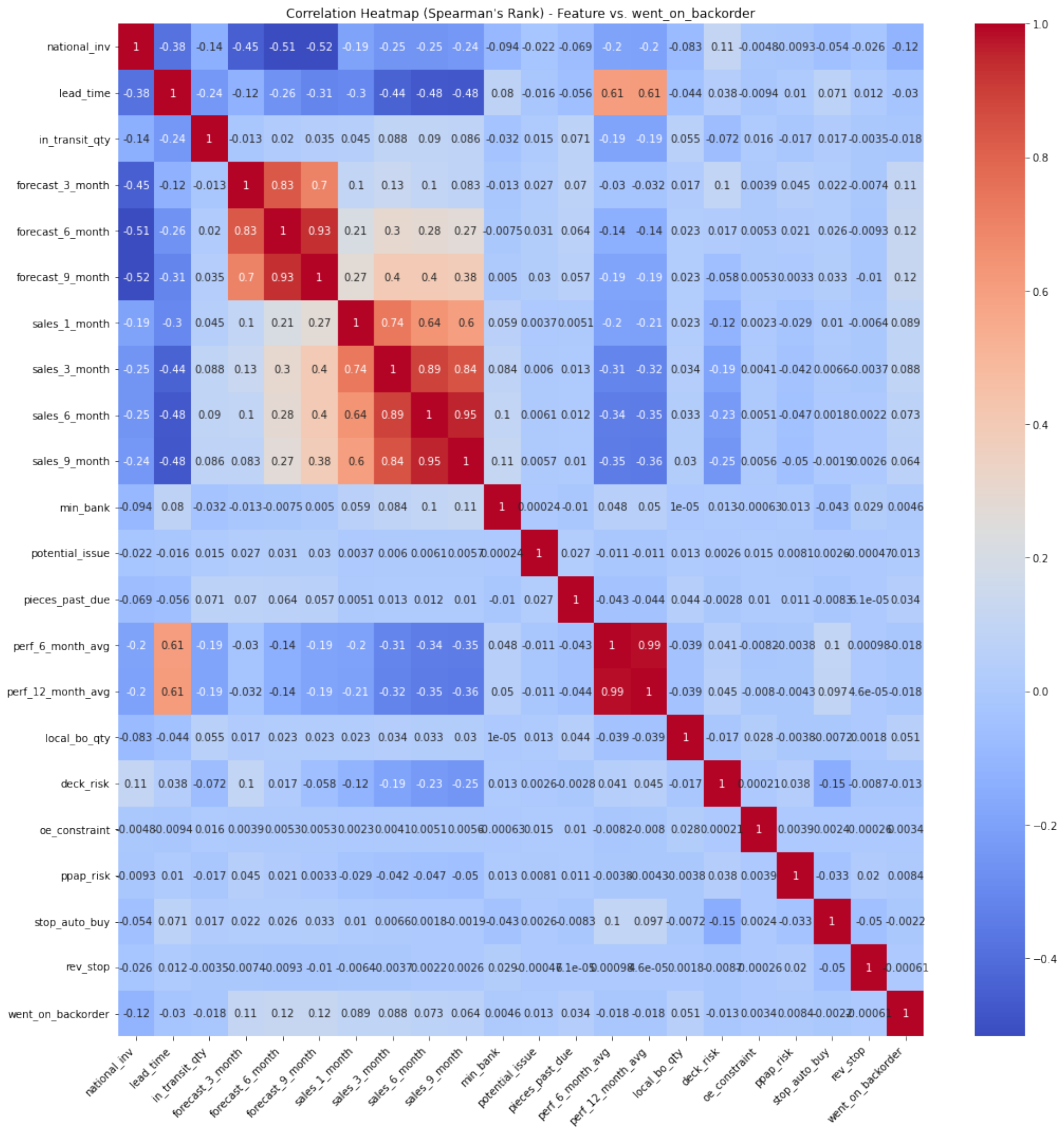
0.9 Feature Seclection

[41]:
```
corr_matrix = df.corr()
corr_matrix_y = corr_matrix['went_on_backorder'].drop('went_on_backorder')
corr_matrix_y=corr_matrix_y.abs().sort_values(ascending=False)
corr_matrix_y
```

[41]:
```
forecast_6_month     0.120852
forecast_9_month     0.118717
national_inv         0.117777
forecast_3_month     0.108193
sales_1_month        0.089218
sales_3_month        0.087973
sales_6_month        0.072508
sales_9_month        0.064349
local_bo_qty         0.050689
pieces_past_due      0.033954
lead_time            0.030197
perf_12_month_avg    0.018328
perf_6_month_avg     0.018034
in_transit_qty       0.017636
deck_risk            0.012827
potential_issue      0.012644
ppap_risk            0.008377
min_bank             0.004633
oe_constraint        0.003420
stop_auto_buy        0.002180
rev_stop             0.000609
Name: went_on_backorder, dtype: float64
```

[42]:
```
logging.info('DATA VISUALISATION...')
# Plotting Heatmap
plt.figure(figsize=(15,15))
sns.heatmap(corr_matrix, annot=True, cmap = 'coolwarm')
```

```
plt.title(f"Correlation Heatmap (Spearman's Rank) - Feature vs.␣
↪went_on_backorder")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.tight_layout()
plt.show()
```



[51]:
```
# for selecting the features we will select the ones which has value for␣
↪corr_matrix_y > 0.01
features = ['national_inv', 'lead_time', 'forecast_3_month', 'forecast_6_month',␣
↪'forecast_9_month', 'sales_1_month',
            'sales_3_month', 'sales_6_month', 'sales_9_month',␣
 ↪'perf_6_month_avg', 'perf_12_month_avg',
            'pieces_past_due','in_transit_qty' , 'deck_risk','local_bo_qty']
x = df[features]
y = df['went_on_backorder']
print(x.shape, y.shape)
```

```
(1929936, 15) (1929936,)
```

0.10 Model Building

[52]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.85,␣
 ↪random_state=42)
print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
```

```
(1640445, 15) (289491, 15)
(1640445,) (289491,)
```

[53]:
```
y_train.value_counts()
```

[53]:
```
0    1628649
1      11796
Name: went_on_backorder, dtype: int64
```

0.10.1 Model 1: RandomOversampler with DecisionTreeClassifier

[54]:
```
#Using RandomOverSampler
oversample = RandomOverSampler(sampling_strategy='auto', random_state=42)
x_resampled, y_resampled = oversample.fit_resample(x_train, y_train)
```

[55]:
```
y_resampled.value_counts()
```

[55]:
```
0    1628649
1    1628649
Name: went_on_backorder, dtype: int64
```

[56]:
```
# Import necessary libraries
from sklearn.metrics import accuracy_score, confusion_matrix

# Decision tree classifier
model1 = DecisionTreeClassifier()
model1.fit(x_resampled, y_resampled)
y_pred1 = model1.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score1 = accuracy_score(y_test, y_pred1)
con_mat1 = confusion_matrix(y_test, y_pred1)
```

```
print('Accuracy of the Decision Tree Model in validation set is:', accu_score1)
print('Confusion Matrix:\n', con_mat1)
print(classification_report(y_test,y_pred1))

from sklearn.metrics import roc_auc_score, average_precision_score, make_scorer

roc1 =  roc_auc_score(y_test, y_pred1)
print("AUC Score is :", roc1)
```

```
Accuracy of the Decision Tree Model in validation set is: 0.9839649591869868
Confusion Matrix:
 [[284170   3136]
 [  1506    679]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    287306
           1       0.18      0.31      0.23      2185

    accuracy                           0.98    289491
   macro avg       0.59      0.65      0.61    289491
weighted avg       0.99      0.98      0.99    289491

AUC Score is : 0.6499199786365443
```

0.10.2 Model 2: RandomOversampler with RandomForestClassifier

[57]:
```
from sklearn.ensemble import RandomForestClassifier

#RandomForest Classifier
model2 = RandomForestClassifier()
model2.fit(x_resampled, y_resampled)
y_pred2 = model2.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score2 = accuracy_score(y_test, y_pred2)
con_mat2 = confusion_matrix(y_test, y_pred2)

print('Accuracy of the Random Forest Model in test set is:', accu_score2)
print('Confusion Matrix:\n', con_mat2)
#AUC score
print(classification_report(y_test,y_pred2))
roc2 =  roc_auc_score(y_test, y_pred2)
print("AUC Score is :", roc2)
```

```
Accuracy of the Random Forest Model in test set is: 0.9871222248705487
Confusion Matrix:
 [[285172   2134]
 [  1594    591]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    287306
           1       0.22      0.27      0.24      2185

    accuracy                           0.99    289491
   macro avg       0.61      0.63      0.62    289491
weighted avg       0.99      0.99      0.99    289491

AUC Score is : 0.6315264642370717
```

0.10.3 Model 3: XGBOOST Classifier with Random Oversampling

[58]:
```python
from xgboost import XGBClassifier

# XGBoost Classifier
model3 = XGBClassifier()
model3.fit(x_resampled, y_resampled)
y_pred3 = model3.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score3 = accuracy_score(y_test, y_pred3)
con_mat3 = confusion_matrix(y_test, y_pred3)

print('Accuracy of the XGBoost Model in test set is:', accu_score3)
print('Confusion Matrix:\n', con_mat3)

# Print classification report
print(classification_report(y_test, y_pred3))

# Calculate AUC score
roc3 = roc_auc_score(y_test, y_pred3)
print("AUC Score is :", roc3)
```

```
Accuracy of the XGBoost Model in test set is: 0.9051680363120097
Confusion Matrix:
 [[260178  27128]
 [   325   1860]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95    287306
           1       0.06      0.85      0.12      2185

    accuracy                           0.91    289491
   macro avg       0.53      0.88      0.53    289491
weighted avg       0.99      0.91      0.94    289491
```

```
AUC Score is : 0.8784183030297025
```

As we have treid Random Oversampling but for all the 4 models the recall score and f1-score are not upto the mark so we will try Synthetic Oversampling Technique to balance the class in the train data set

0.10.4 SMOTE OverSampling

[59]:
```
from imblearn.over_sampling import SMOTE

# Create SMOTE object
smote = SMOTE(sampling_strategy='auto', random_state=42)

# Resample the training data using SMOTE
x_resampled, y_resampled = smote.fit_resample(x_train, y_train)
print(x_resampled.shape,y_resampled.shape)
```

```
(3257298, 15) (3257298,)
```

0.10.5 Model 4: DecisionTree Classifier with SmoteOversampling

[60]:
```
# Decision tree classifier
model4 = DecisionTreeClassifier()
model4.fit(x_resampled, y_resampled)
y_pred4 = model4.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score4 = accuracy_score(y_test, y_pred4)
con_mat4 = confusion_matrix(y_test, y_pred4)

print('Accuracy of the Decision Tree Model in validation set is:', accu_score4)
print('Confusion Matrix:\n', con_mat4)
print(classification_report(y_test,y_pred4))

# Calculate AUC score
roc4 = roc_auc_score(y_test, y_pred4)
print("AUC Score is :", roc4)
```

```
Accuracy of the Decision Tree Model in validation set is: 0.9765277677026228
Confusion Matrix:
 [[281655   5651]
 [  1144   1041]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    287306
           1       0.16      0.48      0.23      2185

    accuracy                           0.98    289491
   macro avg       0.58      0.73      0.61    289491
```

```
weighted avg       0.99      0.98      0.98    289491

AUC Score is : 0.728380640763806
```

0.10.6 Model 5 SmoteOversampler with RandomForestClassifier

[61]:
```
model5 = RandomForestClassifier()
model5.fit(x_resampled, y_resampled)
y_pred5 = model5.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score5 = accuracy_score(y_test, y_pred5)
con_mat5 = confusion_matrix(y_test, y_pred5)

print('Accuracy of the Random Forest Model in validation set is:', accu_score5)
print('Confusion Matrix:\n', con_mat5)
print(classification_report(y_test, y_pred5))
# Calculate AUC score
roc5 = roc_auc_score(y_test, y_pred5)
print("AUC Score is :", roc5)
```

```
Accuracy of the Random Forest Model in validation set is: 0.9841963998880794
Confusion Matrix:
 [[283756   3550]
 [  1025   1160]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    287306
           1       0.25      0.53      0.34      2185

    accuracy                           0.98    289491
   macro avg       0.62      0.76      0.66    289491
weighted avg       0.99      0.98      0.99    289491

AUC Score is : 0.7592681423505896
```

0.10.7 Model 6: SmoteOversampler with XGBoostClassifier

[62]:
```
model6 = XGBClassifier()
model6.fit(x_resampled, y_resampled)
y_pred6 = model6.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score6 = accuracy_score(y_test, y_pred6)
con_mat6 = confusion_matrix(y_test, y_pred6)

print('Accuracy of the XGBoost Model in validation set is:', accu_score6)
print('Confusion Matrix:\n', con_mat6)
print(classification_report(y_test, y_pred6))
# Calculate AUC score
roc6 = roc_auc_score(y_test, y_pred6)
print("AUC Score is :", roc6)
```

```
Accuracy of the XGBoost Model in validation set is: 0.9051265842461423
Confusion Matrix:
 [[260183  27123]
 [   342   1843]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95    287306
           1       0.06      0.84      0.12      2185

    accuracy                           0.91    289491
   macro avg       0.53      0.87      0.53    289491
weighted avg       0.99      0.91      0.94    289491

AUC Score is : 0.874536844370447
```

0.10.8 Model 7: SmoteOversampler with GradientBoostingClassifier

```
[63]: model7 = GradientBoostingClassifier()
model7.fit(x_resampled, y_resampled)
y_pred7 = model7.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score7 = accuracy_score(y_test, y_pred7)
con_mat7 = confusion_matrix(y_test, y_pred7)

print('Accuracy of the Gradient Boosting Model in validation set is:',␣
 ↪accu_score7)
print('Confusion Matrix:\n', con_mat7)
print(classification_report(y_test, y_pred7))

# Calculate AUC score
roc7 = roc_auc_score(y_test, y_pred7)
print("AUC Score is :", roc7)
```

```
Accuracy of the Gradient Boosting Model in validation set is: 0.8743898774055152
Confusion Matrix:
 [[251245  36061]
 [   302   1883]]
              precision    recall  f1-score   support

           0       1.00      0.87      0.93    287306
           1       0.05      0.86      0.09      2185
```

```
    accuracy                           0.87    289491
   macro avg       0.52      0.87      0.51    289491
weighted avg       0.99      0.87      0.93    289491

AUC Score is : 0.8681353184839752
```

0.10.9 Model 8: RandomUnderSampler with DecisionTreeClassifier

[64]:
```
from imblearn.under_sampling import RandomUnderSampler

# Undersampling
undersampler = RandomUnderSampler(random_state=42)
x_resampled, y_resampled = undersampler.fit_resample(x_train, y_train)
print(x_resampled.shape,y_resampled.shape)
```

```
(23592, 15) (23592,)
```

[66]:
```
# Decision Tree Classifier
model8 = DecisionTreeClassifier()
model8.fit(x_resampled, y_resampled)
y_pred8 = model8.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score8 = accuracy_score(y_test, y_pred8)
con_mat8 = confusion_matrix(y_test, y_pred8)

print('Accuracy of the Decision Tree Model in validation set is:', accu_score8)
print('Confusion Matrix:\n', con_mat8)
print(classification_report(y_test, y_pred8))

# Calculate AUC score
roc8 = roc_auc_score(y_test, y_pred8)
print("AUC Score is :", roc8)
```

```
Accuracy of the Decision Tree Model in validation set is: 0.8409967840105564
Confusion Matrix:
 [[241627  45679]
 [   351   1834]]
              precision    recall  f1-score   support

           0       1.00      0.84      0.91    287306
           1       0.04      0.84      0.07      2185

    accuracy                           0.84    289491
   macro avg       0.52      0.84      0.49    289491
weighted avg       0.99      0.84      0.91    289491

AUC Score is : 0.8401842526361157
```

[68]:
```
model9 = RandomForestClassifier(n_estimators=100, max_depth=10,␣
 ↪class_weight='balanced', random_state=72)
model9.fit(x_resampled, y_resampled)

# Predict on the test set
y_pred9 = model9.predict(x_test)

# Calculate accuracy and confusion matrix
accu_score9 = accuracy_score(y_test, y_pred9)
con_mat9 = confusion_matrix(y_test, y_pred9)

print('Accuracy of the Random Forest Model in validation set is:', accu_score9)
print('Confusion Matrix:\n', con_mat9)
print(classification_report(y_test, y_pred9))

# Calculate AUC score
roc9 = roc_auc_score(y_test, y_pred9)
print("AUC Score is :", roc9)
```

```
Accuracy of the Random Forest Model in validation set is: 0.8648628109336732
Confusion Matrix:
 [[248429  38877]
 [   244   1941]]
              precision    recall  f1-score   support

           0       1.00      0.86      0.93    287306
           1       0.05      0.89      0.09      2185

    accuracy                           0.86    289491
   macro avg       0.52      0.88      0.51    289491
weighted avg       0.99      0.86      0.92    289491

AUC Score is : 0.8765069314865193
```