greatlearning
Learning for Life

BACKORDER PREDICTION

Post Graduate Program in Data Science Engineering

Location: Bangalore Batch :

PGP DSE- OCT 22

Submitted by

RACHANA

PRAJESSH P

MD TOUQEER AR

SUDHAN MURUGAVEL

ANMOL GUPTA

Mentored by

PRACHI TARE

Table of Contents

INTRODUCTION:

When a customer orders a product, which is not available in the store or temporary out of stock, and the customer decides to wait until the product is available and promised to be shipped, then this scenario is called backorder of that specific product . If backorders are not handled promptly, they will have high impacts on the respective company's revenue, share market price, customers' trust, and may end up losing the customer or sale order. On the other hand, the prompt actions to satisfy backorders put enormous pressure on different stages of the supply chain which may exhaust the supply chain processes or may appear with extra labor and/or production costs, and associated shipment expenses. Moreover, the uncertainty in customers' demands causes difficulty in forecasting the demand which makes the traditional supply chain management systems less effective in many ways such as inaccurate demand forecasting or misclassifying of back-ordered products. Nowadays, some companies predict the backorders of products by applying machine learning prediction processes to overcome the associated tangible and intangible costs of backorders.

Machine learning models may misclassify many records if the dataset contains misleading or missing information. This issue is a challenge to analyze the dataset of this study. There are very high negative and positive values in several predicting features of this dataset. Our dataset contains the number of negative records related to the inventory. The negative inventory level suggests that the current stock level of the product is less than zero. The causes and effects of negative inventory are well understood in the supply chain industry. An inventory level dependent ordering model discusses the relationship between inventory level and demand which reflects how negative inventory level can affect the demand . Recent studies suggest that the correlation factor among the product variety, sales, and inventory level is biasing the inventory level . Backorder aging prediction can be feasible for the market with non-volatile demand where the lead time, price per unit, quantity of placed order, and product stock level are the main drivers. However, a sudden change in the demand may raise other risk fags associated with the supply chain and may lead to a loss . To cope with the challenges of stochastic demand, a few researchers developed multi-objective inventory models . It has been proven mathematically that the hybrid backorder (i.e., fixed and time-weighted backorder) inventory model is more efficient than the fixed backorder inventory model in the market with volatile demand.

BUSINESS PROBLEM STATEMENT:

Business Problem Understanding: The primary business problem that Back Order Prediction seeks to address is the issue of stockouts, where a business runs out of stock of a particular product. Classify the products whether they would go into Backorder (Yes or No) based on the historical data from inventory, supply chain and sales. This can result in lost sales, reduced customer satisfaction, and decreased revenue. Back Order Prediction aims to predict when a product may go out of stock and the estimated duration until it is restocked, allowing businesses to take proactive steps to avoid stockouts and optimize their inventory levels.

Business Objective: The primary business objective of Back Order Prediction is to improve inventory management and avoid stockouts. The following are the specific business objectives that Back Order Prediction seeks to achieve:

1. Increase sales revenue: By avoiding stockouts, businesses can ensure that they have sufficient stock to meet customer demand, which can lead to increased sales and revenue.
2. Improve customer satisfaction: By accurately predicting stock levels, businesses can ensure that they have the products customers need in stock. This can lead to improved customer satisfaction and loyalty.
3. Optimize inventory management: Back Order Prediction can help businesses optimize their inventory levels, reducing the risk of overstocking or understocking.
4. Improve supply chain management: By identifying suppliers with long lead times or other issues that impact product availability, Back Order Prediction can help businesses improve their supply chain management.
5. Reduce costs: By optimizing inventory levels, businesses can reduce storage and carrying costs associated with excess inventory. Additionally, avoiding stockouts can reduce the costs associated with lost sales and customer dissatisfaction.

Overall, the business objective of Back Order Prediction is to improve inventory management, increase sales revenue, and improve customer satisfaction by accurately predicting stock levels and avoiding stockouts.

LITERATURE REVIEW

Machine Learning (ML) techniques enable us to forecast accurately multiple aspects related to supply chain management such as demand, sale, revenue, production, and backorder. ML approaches have been used to predict manufacturers' garbled demands where some researchers applied a representative set of ML-based and traditional forecasting methods to the data to compare the precision of those used methods [1]. Those researchers found that the average performances of the ML method did not outperform the traditional methods, but when a Support Vector Machine (SVM) was trained on several demand-series, it produced the most precise predictions [2]. The same researchers extended their research works using Support Vector Machines (SVM) and Neural Networks (NN) [3]. They have found that the techniques of applying machine learning models provided noticeable improvements over the traditional models [4]. An analysis of the supply chain's demand prediction was carried out by applying the Support Vector Regression (SVR) method in the paper of Guanghui [5]. The outcome of that investigation indicated that the prediction performance of SVR is superior to Radial Basis Function (RBF) [6], as SVR produced smaller results of the relative mean square error along with higher forecast precision of the supply chain. However, several factors were not taken into account in that research (e.g., imbalance class problem, application of machine learning techniques like neural network and ensemble methods due to the limitations of the computational resources).

To minimize the supply chain and inventory control costs, a risk-based dynamic backorder replenishment planning framework was proposed by Shin et al. [7] applying the Bayesian Belief Network. A similar framework was prescribed by Acar and Gardner [8], using optimization and simulation techniques. Rodger [10] presented a risk triggering model using fuzzy feasibility Bayesian probabilistic evaluation of backorder. To deal with the imbalanced class problem efficiently, ML classifiers were examined in [11] to identify a suitable forecasting model. To carry out this task, they applied different measures along with the ensemble learning. The results of that investigation showed that the ensemble learning method provided feasible performance when precision recall curves were considered, and also minimized the computational costs. They also suggested applying different ML algorithms such as SVM and NN for the verification of potential performance improvements. Prak and Teunter [12] investigated the prediction uncertainty in an inventory model, and they proposed a framework to estimate the demand to obtain more accurate inventory decisions.

DATASET INFORMATION

The dataset of this research has been published in Kaggle. It is divided into the training and testing datasets. Each dataset contains 23 attributes with 1,687,862 and 242,077 observations for the training and testing sets, respectively. Both datasets contain a mix of features with floating-point, integer, and string values. For this study, we intend to use the most common data attributes that can be readily available for any business. Hence, we have chosen inventory, lead time, sales, and forecasted sale as our predicting variables and 'went on backorder' as our response variable. Our target variable is labeled with two classes. Hence, this scenario falls under the binary classification problem.

The inventory feature indicates an available stock of products, although it contains high numbers of negative records. The negative inventory may arise due to the machine or human error. It may also occur when a shipment is recorded as complete before it arrives. The 'lead time' feature indicates the elapsed time between the placement of products' orders and delivery of those products to the customers. The lead time in our dataset ranges from 0 to 52 weeks. The sales features are divided into four parts as one-month sale, three months sale, six months sale, and nine months sale.

The forecasted sale is divided into three columns showing the forecast of three months, six months, and nine months. Figure 1 shows the distribution of some samples in the dataset. It is observable in this figure that the data points of different features have many outliers with different ranges. In both training and testing datasets, a large number of missing values across the predicting variables are observed. Moreover, our response variable is highly imbalanced with 0.669% data from 'Yes' class, and 99.33% data from 'No' class. Figure 1 depicts how the data samples are distributed among two classes, where 0 indicates 'No' class or non-backorder items and 1 indicates 'Yes' class or backorder items.

VARIABLE IDENTIFICATION

Independent Variables: There are 23 independent variables are listed below :

Variable Name	Variable Description
Sku(Stock Keeping unit)	The product id – Unique for each row so can be ignored
National_inv	The present inventory level of the product
Lead_time	Transit time of the product
In_transit_qty	The amount of product in transit
Forecast_3_month , Forecast_6_month , Forecast_9_month	Forecast of the sales of the product for coming 3 , 6 and 9 months respectively
Sales_1_month , sales_3_month ,sales_6_month , sales_9_month	Actual sales of the product in last 1 , 3 ,6 and 9 months respectively
Min_bank	Minimum amount of stock recommended
Potential_issue	Any problem identified in the product/part
Pieces_past_due	Amount of parts of the product overdue if any
Perf_6_month_avg , perf_12_month_avg	Product performance over past 6 and 12 months respectively
Local_bo_qty	Amount of stock overdue
Deck_risk , oe_constraint, ppap_risk, stop_auto_buy, rev_stop	Different Flags (Yes or No) set for the product
Went_on_backorder	Product went on backorder(Target variable)

Fig: Variables

APPROACH

The approach to build a Back Order Prediction model typically involves the following steps:

1. Data collection: Gather historical data on product demand, inventory levels, supplier lead times, and other factors that can impact stock levels.

2. Data preprocessing: Clean and preprocess the data to remove outliers, missing values, and other errors. Aggregate the data into a format suitable for training machine learning models.

3. Feature engineering: Create new features or transform existing ones that can improve the accuracy of the model. For example, features such as product category, price, promotions, and seasonality may be included to help predict stock levels.

4. Model selection: Select an appropriate machine learning algorithm for the Back Order Prediction task. Commonly used algorithms include linear regression, decision trees, and neural networks.

5. Model training: Train the selected machine learning model on the preprocessed data, using techniques such as cross-validation to optimize model parameters.

6. Model evaluation: Evaluate the trained model's accuracy and performance using metrics such as mean squared error, root mean squared error, and R-squared.

7. Model deployment: Deploy the trained model in a production environment to make predictions on new data. Integrate the model with inventory management systems to provide real-time insights and recommendations.

8. Model monitoring and updating: Monitor the model's performance over time and retrain or update the model as needed to ensure its accuracy and effectiveness.

Overall, the approach to Back Order Prediction involves a combination of data processing, feature engineering, machine learning, and integration with inventory management systems to provide real-time insights and recommendations to businesses.

TARGET VARIABLE

The target variable of the above dataset is went_on_backorder. We have to predict whether the order will go into backorder or not. In the above dataset, 98.88% of orders are not going into backorders and 1.11% of are going for backorder. We observe that this data is very much imbalanced.

DATA PREPROCESSING:

Data pre-processing is a process of preparing the raw data and making it suitablefor a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre- processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Missing Value Treatment:

The next step of data pre-processing is to handle missing data in the datasets. Ifour dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

	Total values missing	Percentage of Missing Values
perf_6_month_avg	13432	7.240034
perf_12_month_avg	12338	6.650353
lead_time	9572	5.159440
national_inv	765	0.412346
rev_stop	1	0.000539
stop_auto_buy	1	0.000539
ppap_risk	1	0.000539
oe_constraint	1	0.000539
deck_risk	1	0.000539
local_bo_qty	1	0.000539
pieces_past_due	1	0.000539
potential_issue	1	0.000539
min_bank	1	0.000539
sales_9_month	1	0.000539
sales_6_month	1	0.000539
sales_3_month	1	0.000539
sales_1_month	1	0.000539
forecast_9_month	1	0.000539
forecast_6_month	1	0.000539
forecast_3_month	1	0.000539
in_transit_qty	1	0.000539
went_on_backorder	1	0.000539

Fig : Missing Value

In dataset we have missing values in all features.Lead_time features is integer datatype, replacing nan values with 0. Filling it with median imputation.For all other columns nan values are present which we can drop or duplicated rows are also present. Out of 23 features 3 feature variable has majority of the missing values.20 of them had less than 1% of missing values, hence the rows were deleted directly.Droping record with index=185523 since it is have null values all columns.

Imputation:

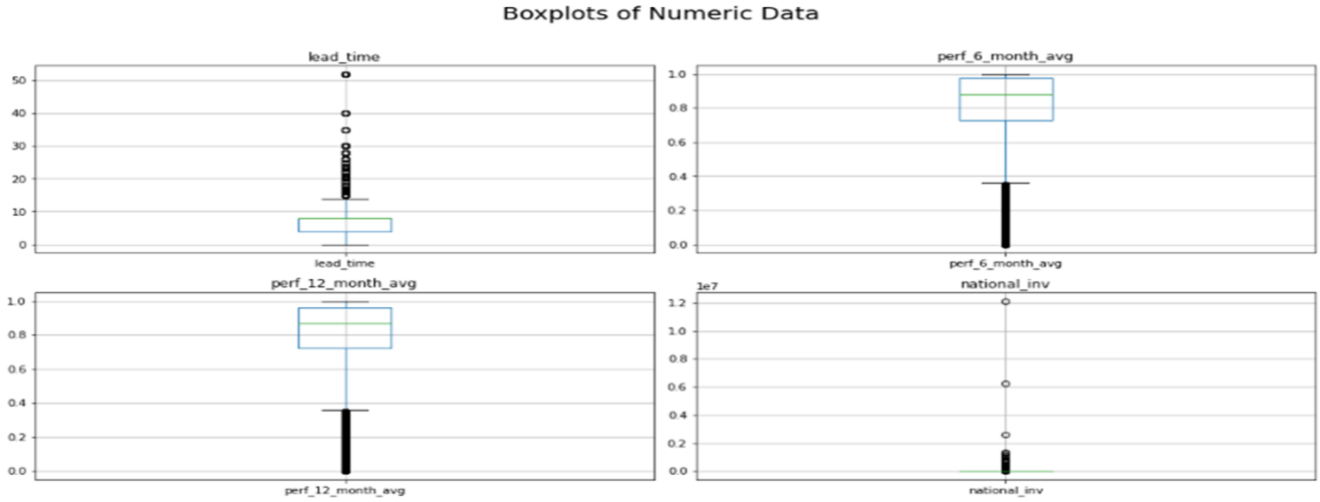


Fig: Boxplots of variable with null values

Imputing missing values is an important step in data preprocessing, and the choice of imputation method depends on the characteristics of the data and the nature of the missingness. Median imputation is one of the simplest and most commonly used methods for imputing missing values, particularly when the data contains outliers. The median is a robust measure of central tendency that is not affected by extreme values or outliers in the data. Therefore, median imputation can be a useful method when outliers are present in the data, as it provides a reasonable estimate of the missing value without being unduly influenced by extreme values.

However, it is important to note that median imputation has some limitations. For example, it assumes that the data are approximately normally distributed, which may not always be the case. Additionally, median imputation can introduce bias in the estimates of other parameters, such as variances and covariances.Therefore, while median imputation can be a useful method for imputing missing values in the presence of outliers, it is important to carefully consider the assumptions and limitations of the method and to explore alternative methods if necessary.

Checking for outliers:

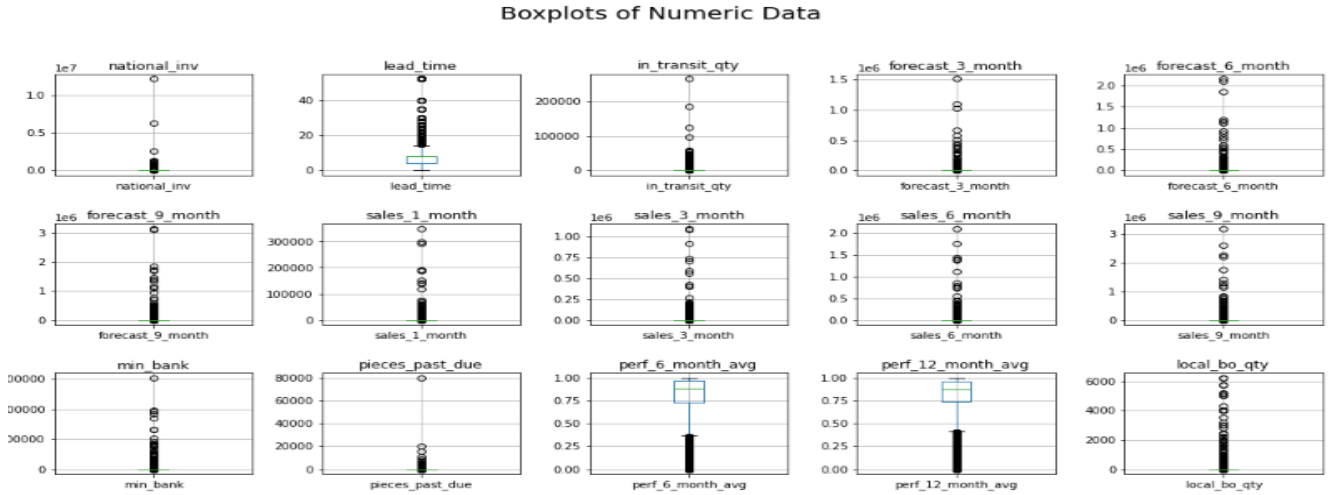


Fig: Boxplot of numerical variables

The box plots above are plotted for entire data. As there is a high variance in the scale(range) of each feature the Box or IQR(Inter Quartile Range) (25th — 75th Percentile) is not clearly visible. If there is a high variance in the scale or range of each feature, the Box or IQR may not be clearly visible in the box plots, and the box plots may not be the most effective way to visualize the data. In such cases, other visualization techniques may be more appropriate, such as normalized box plots, violin plots, or density plots.

Winsorization can be a good approach to handle a large number of outliers, particularly because the outliers are genuine extreme values and removing them would significantly alter the results of the analysis. Winsorization is a data preprocessing technique that involves replacing extreme values in a dataset with less extreme values, typically the highest or lowest non-extreme values. This approach can be useful for handling outliers because it preserves the original data distribution while reducing the influence of extreme values. Winsorization can also be used to reduce the impact of measurement errors, which can cause extreme values in the data. The variable pieces_past_due has one extreme record, therefore we are dropping the record.

STATISTICAL ANALYSIS

Chi-square test of independence:

Null hypothesis: There is no relationship between the two categorical variables.
Alternative hypothesis: There is a relationship between the two categorical variables.

Based on the results of the hypothesis tests, we can infer that the variables 'deck_risk' and 'stop_auto_buy' are significant in predicting whether an item will go on backorder or not.

On the other hand, the p-values for 'potential_issue', 'oe_constraint', 'ppap_risk', and 'rev_stop' are greater than the significance level of 0.05, indicating that we fail to reject the null hypothesis and cannot conclude that there is a significant association between these variables and the target variable 'went_on_backorder'.

	Variable	Target	p-value	Result
0	potential_issue	went_on_backorder	0.082022	Fail to reject null hypothesis
1	deck_risk	went_on_backorder	0.000000	Reject null hypothesis
2	oe_constraint	went_on_backorder	1.000000	Fail to reject null hypothesis
3	ppap_risk	went_on_backorder	0.323084	Fail to reject null hypothesis
4	stop_auto_buy	went_on_backorder	0.017481	Reject null hypothesis
5	rev_stop	went_on_backorder	1.000000	Fail to reject null hypothesis

Fig: Chi-Square Test

Mann-Whitney Test

Mann-Whitney test, also known as Wilcoxon rank-sum test, is used to compare the distribution of a continuous variable between two groups, where one group is binary. It does not assume any particular distribution of the data and is robust to outliers. The test calculates a rank-based statistic that measures the difference in the median ranks between the two groups.

The Mann-Whitney test is a non-parametric test that is used to determine whether two independent samples are drawn from populations with the same distribution. In this case, the test was used to compare the distribution of each numerical variable between the backorder and non-backorder groups.

The results of the Mann-Whitney test show that for all numerical variables, the null hypothesis (that the distributions of the variable are the same in both the backorder and non-backorder groups) is rejected with a p-value of 0. This indicates that there is strong evidence to suggest that the distribution of each numerical variable is different between the backorder and non-backorder groups. Therefore, we can conclude that there are significant differences in the numerical variables between the two groups, and these variables could be useful in predicting whether a product will go on backorder or not.

	variable	U_statistic	p_value	null_hypothesis
0	national_inv	94107397.500000	0.000000	Rejected
1	lead_time	221633312.000000	0.000000	Rejected
2	in_transit_qty	212318011.500000	0.000000	Rejected
3	forecast_3_month	341101238.500000	0.000000	Rejected
4	forecast_6_month	332131538.000000	0.000000	Rejected
5	forecast_9_month	325739182.500000	0.000000	Rejected
6	sales_1_month	282234360.000000	0.000000	Rejected
7	sales_3_month	279993823.000000	0.000000	Rejected
8	sales_6_month	270479856.000000	0.000000	Rejected
9	sales_9_month	265278705.500000	0.000000	Rejected
10	min_bank	215830188.500000	0.000000	Rejected
11	pieces_past_due	253927726.000000	0.000000	Rejected
12	perf_6_month_avg	202027255.500000	0.000000	Rejected
13	perf_12_month_avg	203350005.500000	0.000000	Rejected
14	local_bo_qty	263161458.500000	0.000000	Rejected

Fig: Mann-Whitney test

UNIVARIATE,BIVARIATE&MULTIVARIATE ANALYSIS

Univariate Analysis

Summary Statistics of Numerical Variable:

```
df_backorder_sample.describe().T
```

	count	mean	std	min	25%	50%	75%	max
national_inv	100000.000000	225.799821	577.626653	0.000000	8.000000	28.000000	128.000000	3036.950000
lead_time	100000.000000	7.108300	3.592086	2.000000	4.000000	8.000000	8.000000	17.000000
in_transit_qty	100000.000000	12.813920	43.584256	0.000000	0.000000	0.000000	0.000000	232.000000
forecast_3_month	100000.000000	234.911310	6342.653171	0.000000	0.000000	0.000000	13.000000	1510592.000000
forecast_6_month	100000.000000	444.335880	9945.571829	0.000000	0.000000	0.000000	35.000000	1858864.000000
forecast_9_month	100000.000000	642.787930	12809.297394	0.000000	0.000000	0.000000	55.000000	1858864.000000
sales_1_month	100000.000000	65.777030	1635.426861	0.000000	0.000000	1.000000	8.000000	349620.000000
sales_3_month	100000.000000	221.476450	5510.713264	0.000000	0.000000	4.000000	28.000000	1099852.000000
sales_6_month	100000.000000	429.151470	8683.939762	0.000000	1.000000	8.000000	60.000000	1433311.000000
sales_9_month	100000.000000	642.836780	12207.957444	0.000000	2.000000	12.000000	92.000000	2261761.000000
min_bank	100000.000000	26.380660	70.181285	0.000000	0.000000	1.000000	12.000000	354.000000
pieces_past_due	100000.000000	1.885070	58.627618	0.000000	0.000000	0.000000	0.000000	9216.000000
perf_6_month_avg	100000.000000	0.802413	0.233961	0.000000	0.730000	0.880000	0.970000	1.000000
perf_12_month_avg	100000.000000	0.796348	0.227394	0.000000	0.740000	0.870000	0.960000	1.000000
local_bo_qty	100000.000000	1.331090	60.859088	0.000000	0.000000	0.000000	0.000000	6232.000000

Fig:5 point summary

There is a large difference between mean and median values , indicating that the variables are highly skewed. There is a large difference between the maximum and mean values, it typically indicates that there are a few extremely large values in the dataset that are significantly driving up the maximum value.

KDE PLOT

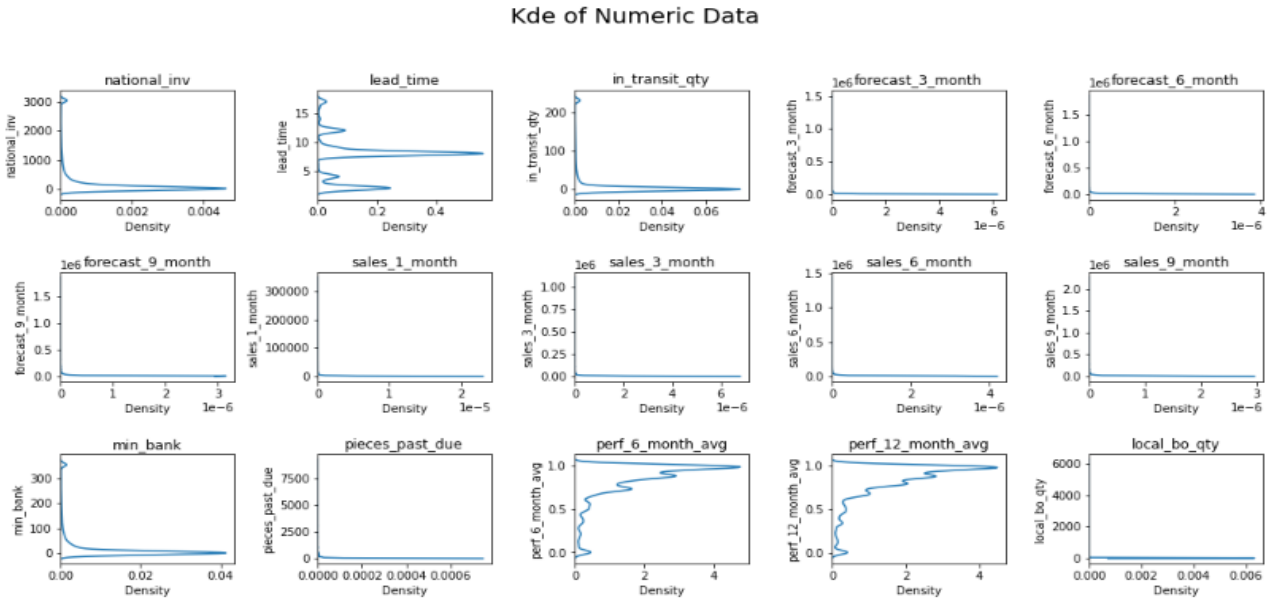


Fig: KDE Plot of numerical variables

The lead_time is moderately skewed. The variable perf_6_month_avg and perf_12_month_avg are negatively skewed. Rest of the variables are highly positively skewed.

VIOLIN PLOT

Majority of Data points are near to zero.

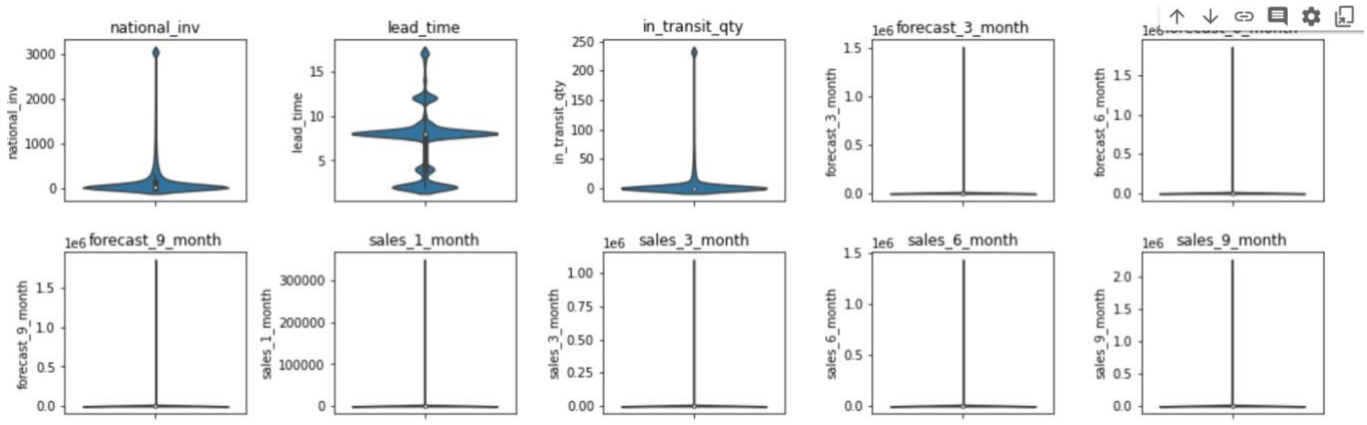


Fig: Violin Plot for Numerical Variables

Summary Statistics of Categorical Variable:

```
df_backorder_sample.describe(include=object).T
```

	count	unique	top	freq
potential_issue	100000	2	No	99972
deck_risk	100000	2	No	82393
oe_constraint	100000	2	No	99980
ppap_risk	100000	2	No	87496
stop_auto_buy	100000	2	Yes	95402
rev_stop	100000	2	No	99959
went_on_backorder	100000	2	No	98559

Fig: Summary Statistics of Categorical variables

There are six categorical variables: "potential_issue", "deck_risk", "oe_constraint", "ppap_risk", "stop_auto_buy", "rev_stop", and "went_on_backorder". Each variable has two possible values, indicated by the "unique" column: "Yes" or "No".The "potential_issue", "oe_constraint", "rev_stop", and "went_on_backorder" variables are highly imbalanced, with one value occurring much more frequently than the other. For example, the "potential_issue" variable has 99,972 "No" values and only 28 "Yes" values.

The "stop_auto_buy" variable is also imbalanced, but in the opposite direction, with 95,402 "Yes" values and only 4,598 "No" values.

This information can be useful for understanding the distribution of categorical variables in the dataset and identifying potential issues or biases in the data. For example, highly imbalanced variables may require special attention in the data analysis and modelling process, such as using appropriate sampling or weighting techniques to account for the imbalance.

BARPLOT AND PIECHART

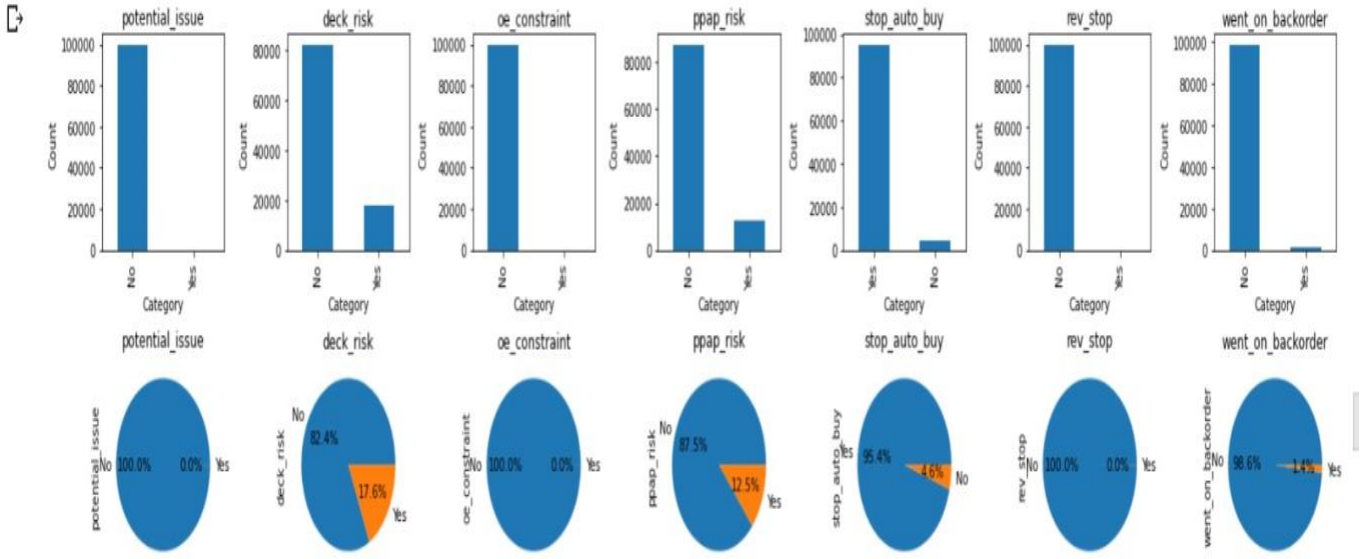


Fig: Categorical vs Categorical(went_on_backorder)

The Bar plots shows the percentage of product going/not going into backorder for every value of categorical Feature. For example , the Blue bar illustrates the percentage of products not going to Backorder when Potential_issue =’Yes’/’No’ and the Orange bar illustrates percentage of products going to backorder when Potential_issue=’Yes’/’No’.

BOX PLOT

]

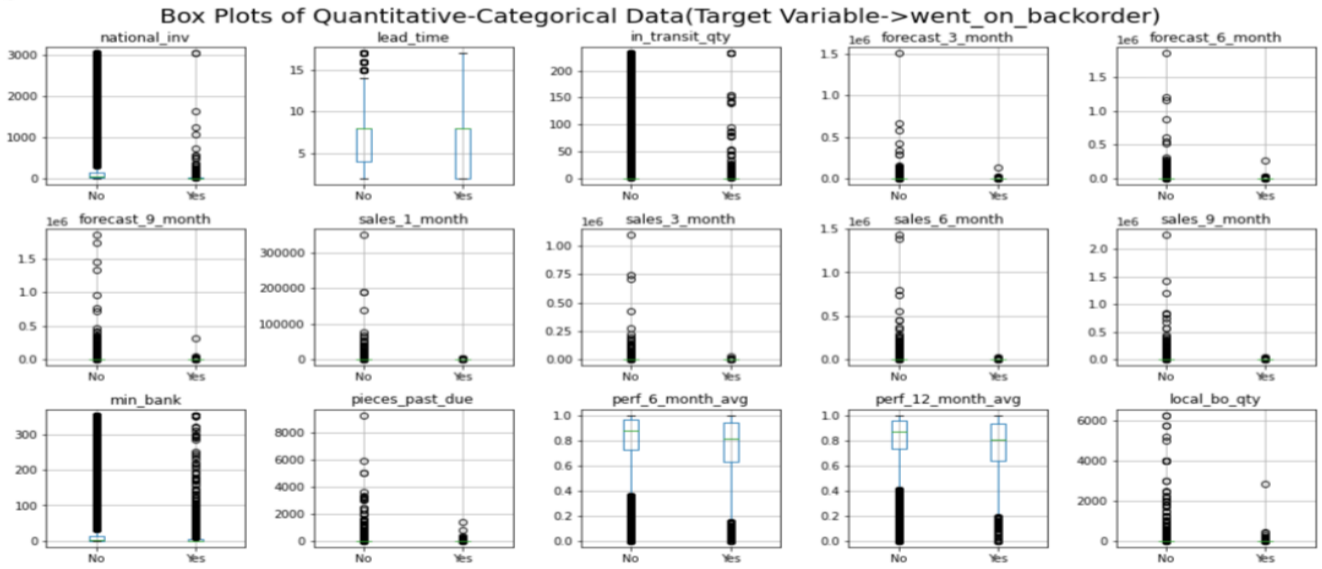


Fig: Boxplot of quantitative-categorical data(went_on_backorder)

The box plots above are plotted for entire data. As there is a high variance in the scale(range) of each feature the Box or IQR(Inter Quartile Range) (25th — 75th Percentile) is not clearly visible. If there is a high variance in the scale or range of each feature, the Box or IQR may not be clearly visible in the box plots, and the box plots may not be the most effective way to visualize the data. In such cases, other visualization techniques may be more appropriate, such as normalized box plots, violin plots, or density plots.

STRIP PLOT

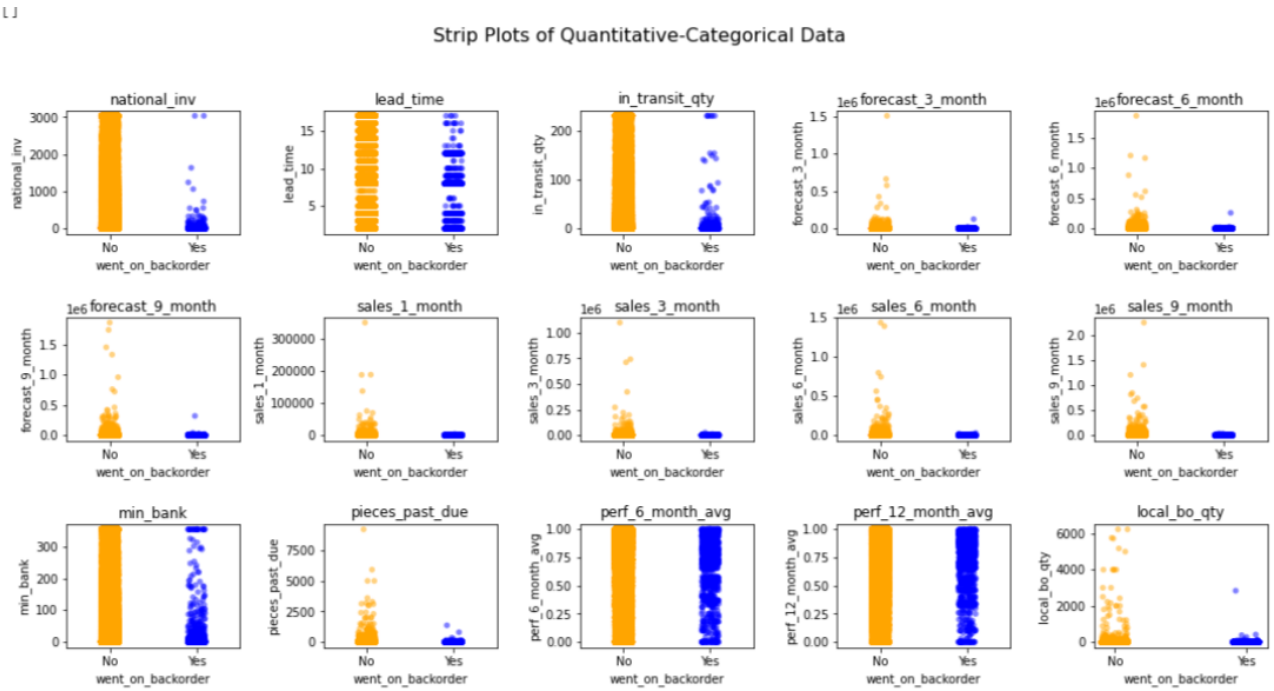


Fig: Strip plot of quantitative-categorical data(went_on_backorder)

A strip plot is a type of data visualization that shows the distribution of a variable or variables in a dataset. It is similar to a scatter plot, but instead of plotting points with varying x and y coordinates, a strip plot plots the values of a variable along one axis, typically the x-axis. In a strip plot, each data point is represented as a small dot or marker along the x-axis. If there are multiple groups or categories in the data, each group is represented by a different color or marker style. This allows for easy comparison of the distribution of the variable(s) across different groups or categories.

'Strip Plots of Quantitative-Categorical Data(Target Variable:-went_on_backorder)

Determine the range of the variable being plotted. This will give you an idea of the minimum and maximum values, as well as the spread of the data. Orange(No) category of the binary variable is densely distributed while the other is partially empty, it may suggest that the feature being visualized has a stronger relationship with one of the binary categories except for perf_6_month_avg and perf_12_month_avg.

Categorical - Categorical
Count Plot

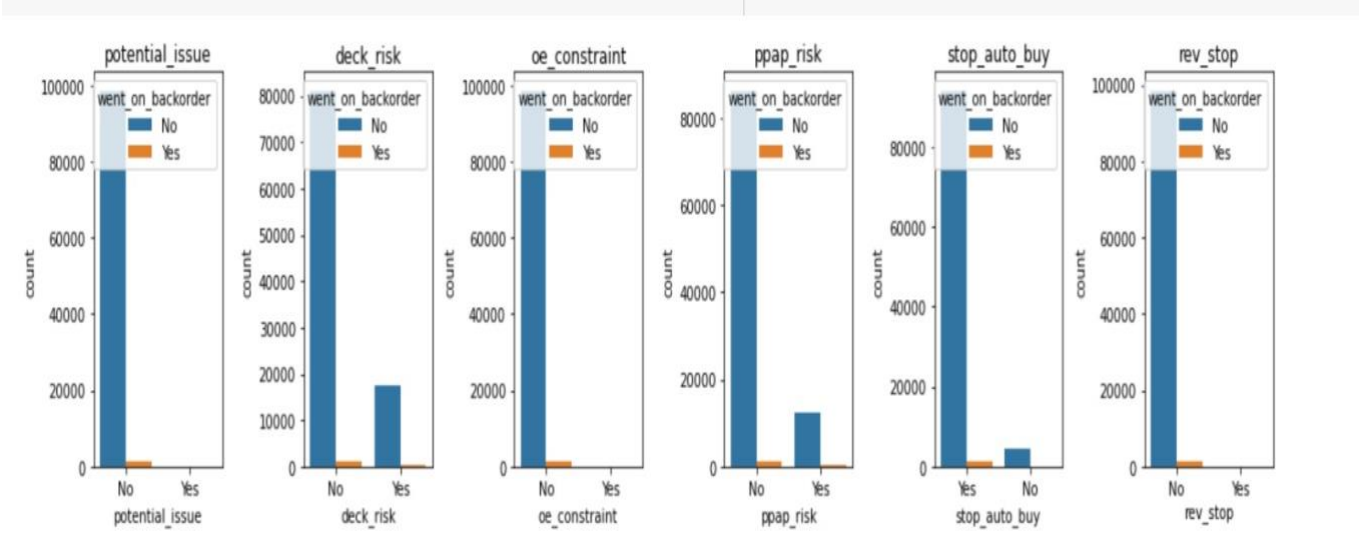


Fig: Count plot (Categorical vs categorical→went_on_backorder)

A count plot for categorical vs categorical variables is a type of data visualization that shows the distribution of one categorical variable relative to another categorical variable. It is essentially a bar chart where the height of each bar represents the frequency or count of observations in each category. To create a count plot for categorical vs categorical variables, one categorical variable is typically plotted on the x-axis and the other on the y-axis.

For the flag risks potential_issue,oe_constraint and rev_stop we can see that the product has not gone for backorder whereas for deck_risk there is 17.65% chance of going for backorderfor ppap_risk there is about 12.5% chance for backorder and for stop_auto_buy there is a minute chance of 4.6% of going into backorder.

Heat Map

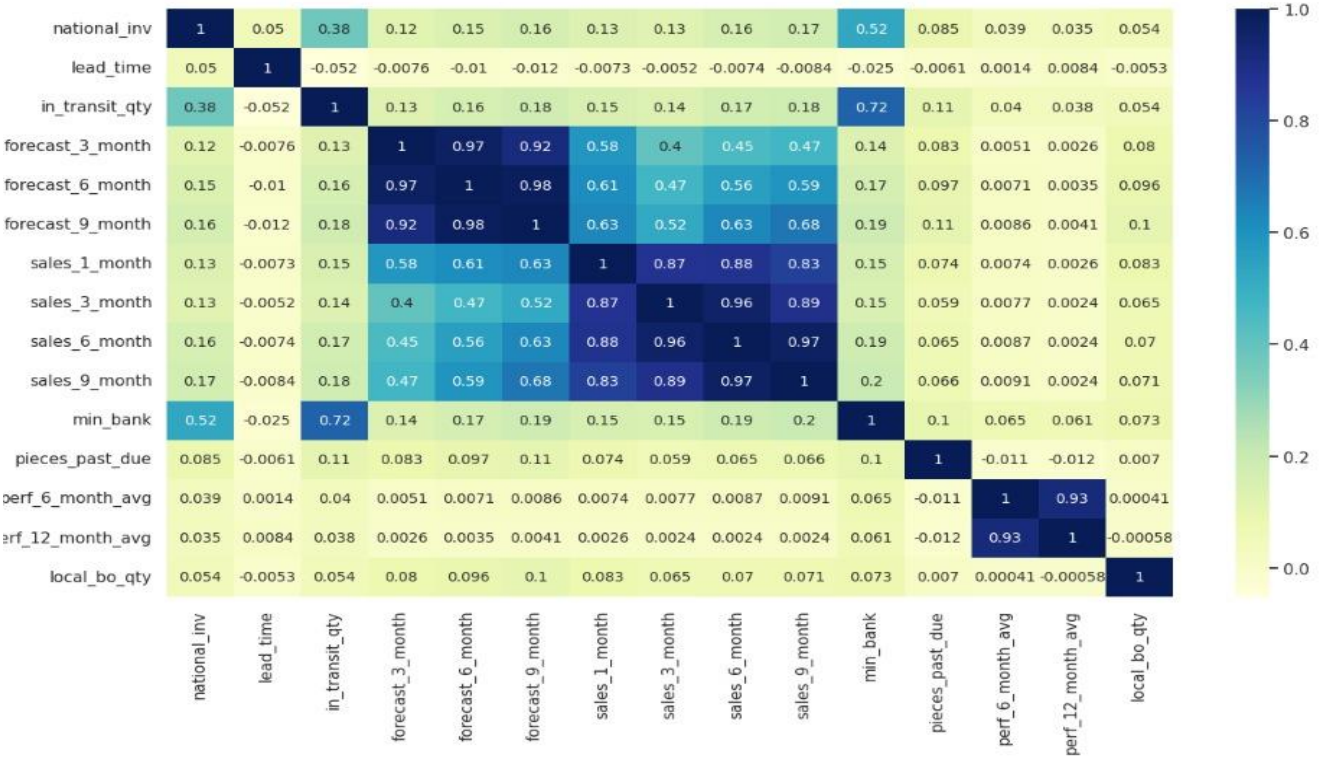


Fig: Heatmap (Numerical vs Numerical)

Fig: Heatmap(Correlation >0.7)

A heat map is a type of data visualization that uses color-coded squares or rectangles to represent the values of a two-dimensional dataset. It is particularly useful for visualizing large datasets and for identifying patterns or trends in the data.In a heat map, each row and column of the dataset is represented by a separate square or rectangle, and the color of the square represents the value of the corresponding cell in the dataset. The color scale typically ranges from a low value (e.g., blue or green) to a high value (e.g., red or yellow), with intermediate values represented by shades of the intermediate colors. They can also be used to visualize patterns or relationships in the data, such as correlations between variables or clusters of similar observations.

From the correlation matrix , we can easily interpret that features from in_transit_qty to min_bank are more correlated with each other. Also the perf_6_months_avg and perf_12_months_avg is highly correlated with each other. Apart from the above mentioned features, other features are very weakly correlated.

ENCODING:

Encoding refers to the process of converting data from one representation to another. In the context of machine learning, encoding is often used to convert categorical variables (e.g., colors, categories, labels) into a numerical representation that can be used by algorithms. This is important because many machine learning algorithms require numerical inputs.

There are several methods of encoding categorical variables, including one-hot encoding, label encoding, and ordinal encoding. One-hot encoding involves creating a binary column for each category, where the column is 1 if the category is present and 0 otherwise. Label encoding involves assigning each category a numerical label (e.g., 0, 1, 2, etc.), which can be problematic if the labels have an order or hierarchy to them. Ordinal encoding is similar to label encoding, but it preserves the order of the categories by assigning each category a numerical value based on its position in a predetermined order.

SCALING & TRANSFORMATION

Scaling and transformation are two common data preprocessing techniques used to prepare data for analysis or modelling . Scaling involves transforming the values of the features in a dataset to a common scale, typically by subtracting the mean and dividing by the standard deviation. This is done to ensure that all features have the same scale and range, which can be important for certain algorithms, such as those that are distance-based. Transformation involves applying a mathematical function to the values of the features in a dataset to transform them in a specific way.

Robust scaling is a data normalization technique used in machine learning to scale numerical data so that it has a mean of 0 and a standard deviation of 1, while being robust to the presence of outliers. It is similar to standard scaling, but instead of using the mean and standard deviation of the data, it uses the median and interquartile range (IQR).

Yeo-Johnson Power Transformation is a data transformation technique used to normalize numerical data that may not follow a normal distribution. It is an extension of the Box-Cox transformation that can handle both positive and negative values. The Yeo-Johnson transformation applies a power transformation to the data using a lambda parameter that is chosen to maximize the normality of the transformed data.

CHECKING FOR CLASS IMBALANCE

In this case, the '0' class represents about 98.56% of the samples, while the '1' class represents only 1.44% of the samples. This level of imbalance can cause problems for a binary classification model, as it may be biased towards predicting the majority class and may not perform well on the minority class. To address this imbalance, we can use algorithms specifically designed for imbalanced data, such as SMOTE (Synthetic Minority Over-sampling Technique).

SMOTE

```
from imblearn.over_sampling import SMOTE
X = transformed_df.drop('went_on_backorder', axis=1)
y = transformed_df['went_on_backorder']
smote = SMOTE(sampling_strategy=0.20, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

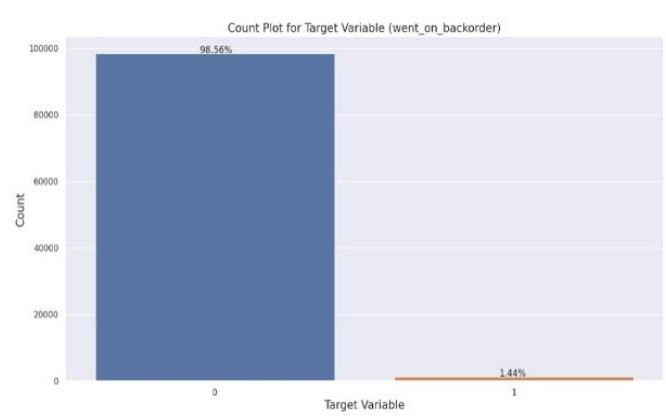


Fig: Imbalance before SMOTE

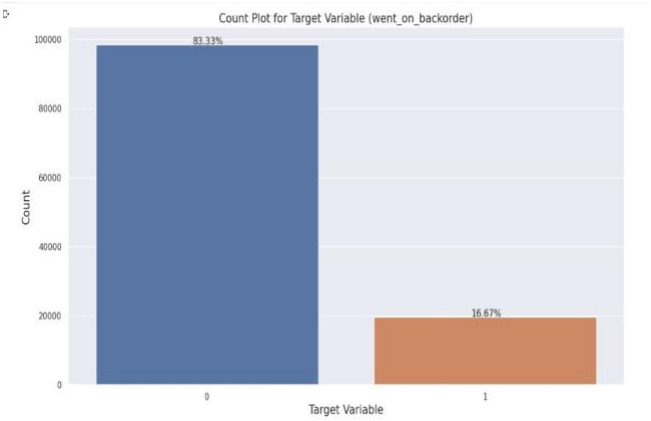


Fig: Imbalance after SMOTE

Minority class will be oversampled to have 20% of the number of samples of the majority class. If there are 1 lakh records with 98.56% majority class, it means that there are 98,560 samples in the majority class and only 1,440 samples in the minority class. To address this class imbalance, the minority class needs to be oversampled. Oversampling the minority class to have 20% of the number of samples of the majority class means that the minority class needs to have 20% of 98,560, which is 19,712 samples. Since there are only 1,440 samples in the minority class, this means that the minority class needs to be oversampled by a factor of approximately 13.7 to reach the desired number of samples. This oversampling can be done by replicating the minority class samples multiple times or by using more advanced techniques such as SMOTE to generate synthetic samples

MODEL BUILDING

Step by step approach for model building: -

1.Data Preprocessing: Clean and preprocess the data by handling missing values, encoding categorical features, scaling numerical features, and removing outliers.
2.Data Splitting: Split the data into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune the hyperparameters of the model, and the testing set is used to evaluate the final performance of the model.
3.Feature Engineering: Create new features from existing ones that can improve the model's predictive power. This can involve transforming or combining features, extracting new features from text or images, or engineering domain-specific features.
4.Model Selection: Select an appropriate machine learning model for the problem at hand, based on the type of data, the size of the dataset, and the performance metrics. Consider using simple models first, and gradually increasing complexity if necessary.
5.Hyperparameter Tuning: Tune the hyperparameters of the chosen model to optimize its performance on the validation set. This can involve selecting the best learning rate, regularization parameter, activation function, number of hidden layers, and other parameters.
6.Model Training: Train the final model on the entire training set using the optimized hyperparameters.
7.Model Evaluation: Evaluate the final model on the testing set using appropriate performance metrics such as accuracy, precision, recall, F1-score, or AUC-ROC curve. Compare the performance of the model with the baseline and previous state-of-the-art methods.
8.Model Deployment: Deploy the final model in a production environment, such as a web application, mobile app, or API, and monitor its performance over time. Update the model as necessary to improve its accuracy or adapt to changing data distributions.

Overall, this step-by-step approach provides a structured and systematic way to build, evaluate, and deploy machine learning models, and can help ensure the quality and robustness of the models.

BASE MODEL (Logistic Regression Model)

LOGISTIC REGRESSION SUMMARY:

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.282113
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:      went_on_backorder   No. Observations:                82789
Model:                          Logit   Df Residuals:                    82768
Method:                           MLE   Df Model:                           20
Date:                Mon, 17 Apr 2023   Pseudo R-squ.:                 0.3759
Time:                        17:20:09   Log-Likelihood:                -23356.
converged:                      False   LL-Null:                       -37422.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
national_inv        -7.4655      0.104    -71.501      0.000      -7.670      -7.261
lead_time           -0.1545      0.014    -11.185      0.000      -0.182      -0.127
in_transit_qty      -2.1009      0.062    -33.656      0.000      -2.223      -1.979
forecast_3_month     3.2032      0.161     19.955      0.000       2.889       3.518
forecast_6_month     0.9672      0.440      2.197      0.028       0.104       1.830
forecast_9_month    -3.5822      0.394     -9.090      0.000      -4.355      -2.810
sales_1_month        2.7809      0.118     23.573      0.000       2.550       3.012
sales_3_month        4.1842      0.273     15.299      0.000       3.648       4.720
sales_6_month        1.3671      0.515      2.655      0.008       0.358       2.376
sales_9_month       -4.5291      0.423    -10.697      0.000      -5.359      -3.699
min_bank            -0.3424      0.082     -4.176      0.000      -0.503      -0.182
pieces_past_due     24.1848      1.665     14.523      0.000      20.921      27.449
perf_6_month_avg    -0.2993      0.063     -4.714      0.000      -0.424      -0.175
perf_12_month_avg    0.0544      0.057      0.948      0.343      -0.058       0.167
local_bo_qty        69.4625      2.459     28.252      0.000      64.644      74.281
potential_issue     -0.1809      0.837     -0.216      0.829      -1.821       1.459
deck_risk           -1.1866      0.038    -30.869      0.000      -1.262      -1.111
oe_constraint      -17.1365   1882.352     -0.009      0.993   -3706.478    3672.206
ppap_risk           -0.1898      0.037     -5.138      0.000      -0.262      -0.117
stop_auto_buy       -2.9440      0.027   -108.964      0.000      -2.997      -2.891
rev_stop           -25.9042   4.01e+04     -0.001      0.999   -7.87e+04    7.86e+04
==============================================================================
```

Based on the p-values, several of the predictor variables appear to be significantly associated with the probability of an item going on backorder, including "national_inv", "lead_time", "in_transit_qty", "forecast_3_month", "forecast_9_month", "sales_1_month", "sales_3_month", "sales_9_month", "min_bank", "pieces_past_due", "perf_6_month_avg", "local_bo_qty", "deck_risk", "ppap_risk", and "stop_auto_buy".The coefficients for "potential_issue", "oe_constraint", and "rev_stop" do not appear to be significant as their p-values are greater than 0.05. However, it's important to note that the model failed to converge and further analysis may be needed to determine the robustness of the results.

TEST REPORT FOR BASE MODEL:

Test Report:

	precision	recall	f1-score	support
0	0.91	0.96	0.93	29643
1	0.71	0.49	0.58	5838
accuracy			0.88	35481
macro avg	0.81	0.73	0.76	35481
weighted avg	0.87	0.88	0.87	35481

Fig: Test Report (Base Model)

The precision of the model for class 0 (negative instances) is 0.91, which means that out of all instances predicted as negative, 91% were actually negative. The recall for class 0 is 0.96, which means that out of all actual negative instances, the model correctly identified 96%. The F1-score for class 0 is 0.93, which is the harmonic mean of precision and recall for class 0.

Similarly, for class 1 (positive instances), the precision of the model is 0.71, which means that out of all instances predicted as positive, 71% were actually positive. The recall for class 1 is 0.49, which means that out of all actual positive instances, the model correctly identified 49%. The F1-score for class 1 is 0.58.

The overall accuracy of the model is 0.88, which means that the model correctly predicted 88% of the instances in the test set.

The macro average of precision, recall and F1-score for both classes is 0.81, 0.73 and 0.76 respectively. The weighted average of precision, recall and F1-score is 0.87, 0.88 and 0.87 respectively, taking into account the class imbalance present in the data.

Based on the test report, the model appears to perform better on negative instances than positive instances. There may be further room for improvement in the model's performance, especially for positive instances.

CONFUSION MATRIX

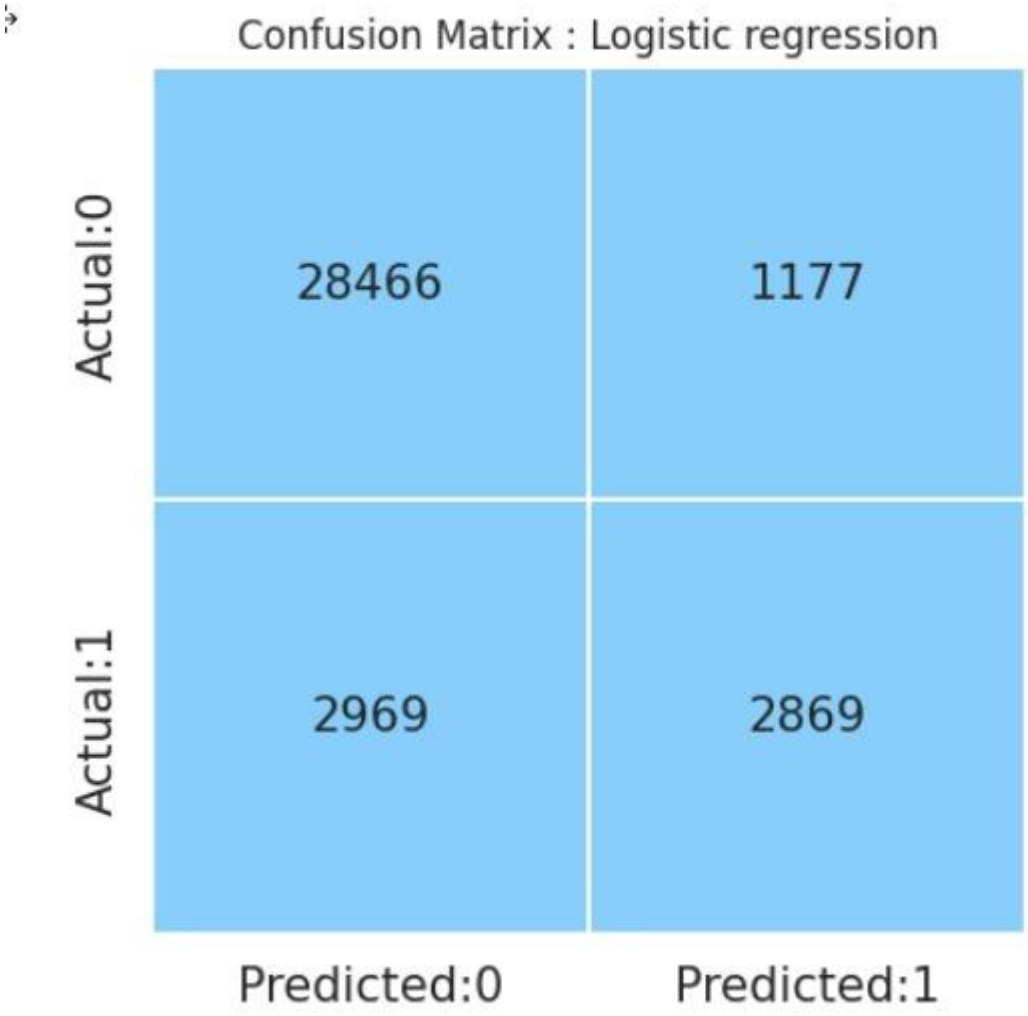


Fig: Confusion Matrix

In this case, your logistic regression model produced the following results:

There were 28,466 true negatives (i.e., the model correctly predicted that 28,466 instances were negative). There were 1,177 false positives (i.e., the model incorrectly predicted that 1,177 instances were positive). There were 2,969 false negatives (i.e., the model incorrectly predicted that 2,969 instances were negative). There were 2,869 true positives (i.e., the model correctly predicted that 2,869 instances were positive). The confusion matrix can be used to calculate various performance metrics for your logistic regression model, such as accuracy, precision, recall, and F1 score.

PERFORMANCE METRICS:

```
Accuracy: 0.8831487274879513
Precision: 0.7090954028670292
Recall: 0.49143542309009935
F1-score: 0.5805341966815054
AUC-ROC: 0.7258647951735624
kappa value: 0.5152324292555936
Cross Entropy 0.2624479553914746
```

Fig: Performance Metrics(Base Model)

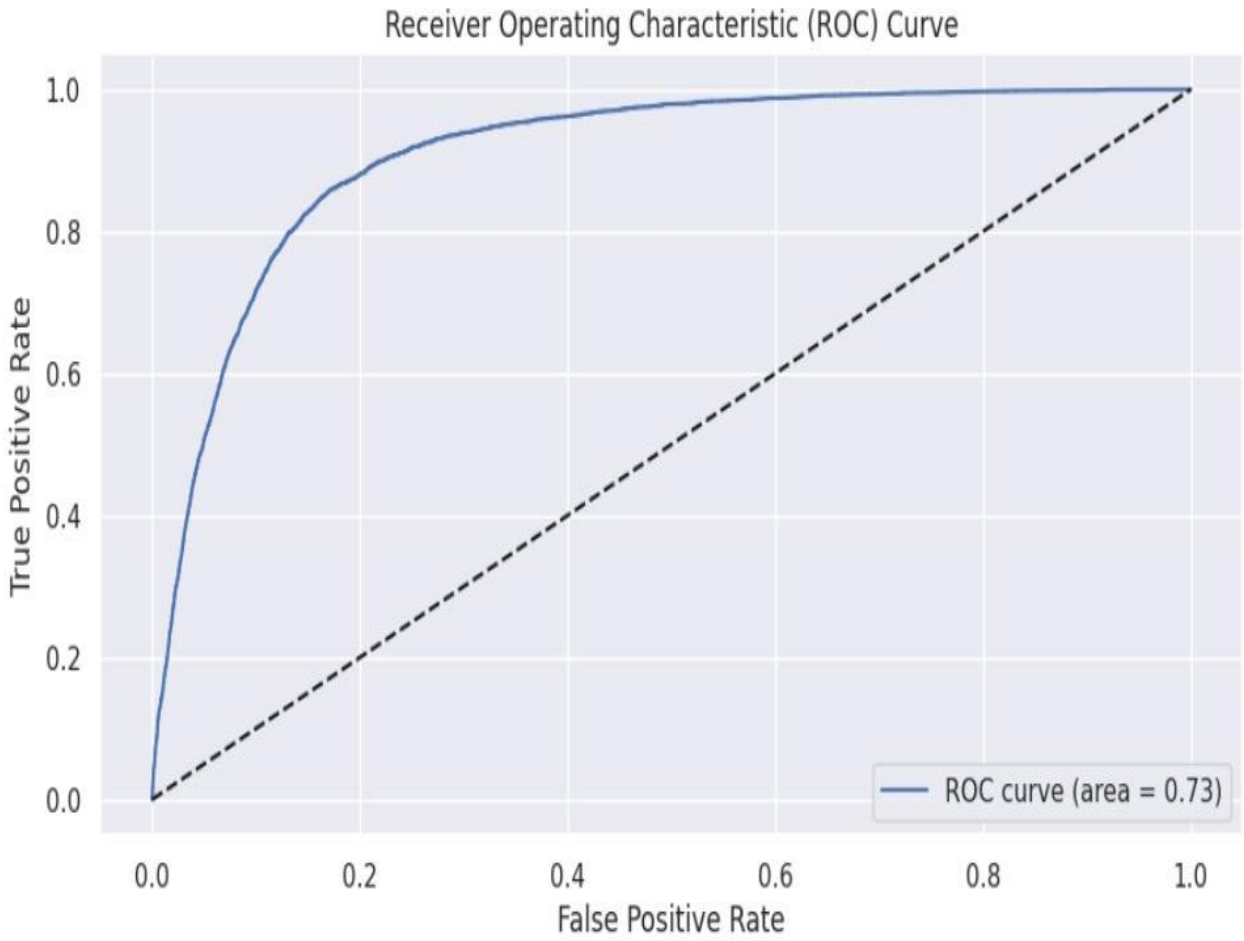
The model's accuracy is 0.883, which indicates that the model correctly classified 88.3% of the instances in the dataset. The precision of the model is 0.709, which means that out of all instances predicted as positive, 70.9% were actually positive. The recall of the model is 0.491, which means that out of all actual positive instances, the model correctly identified 49.1%. The F1-score of the model is 0.581, which is the harmonic mean of precision and recall, and provides a balance between these two metrics. The AUC-ROC score of 0.726 indicates that the model has a moderate ability to distinguish between positive and negative instances.

The kappa value of 0.515 suggests that there is moderate agreement between the predictions made by the model and the actual values. This metric is useful in assessing inter-rater agreement, and values close to 1 indicate strong agreement, while values close to 0 indicate chance agreement. AUC-ROC is a metric that measures the area under the receiver operating characteristic (ROC) curve. The ROC curve is a plot of true positive rate (recall) versus false positive rate (1 - specificity) at various thresholds. In this case, the AUC-ROC is 0.725, indicating that the model performs moderately well at distinguishing between positive and negative instances.

The computed cross-entropy loss on the test data is 0.2624, which is a measure of how well the logistic regression model fits the data and predicts the probability of the target variable (went_on_backorder). The lower the cross-entropy loss, the better the model. In this case, the obtained value of 0.2624 is relatively low, indicating that the logistic regression model has a good fit to the data and can predict the probability of a product going on backorder with reasonable accuracy.

Overall, the model appears to have moderate performance in terms of accuracy, precision, recall, F1-score, AUC-ROC, kappa value and cross-entropy. There may be room for improvement in the model's performance, especially in identifying positive instances.

ROC CURVE




```
print('AUC:', roc_auc)
```

```
AUC: 0.7258647951735624
```

Fig: ROC CURVE

The AUC of your logistic regression model on the test data is approximately 0.73. AUC-ROC is a metric that measures the area under the receiver operating characteristic (ROC) curve. The ROC curve is a plot of true positive rate (recall) versus false positive rate (1 - specificity) at various thresholds. In this case, the AUC-ROC is 0.7258, indicating that the model performs moderately well at distinguishing between positive and negative instances.

Cross-Validation

```
scores = cross_val_score(LR, X_train, y_train, cv=5)
scores
```

```
array([0.88247373, 0.88199058, 0.88035995, 0.8798164 , 0.88331219])
```

Fig: Cross-Validation

These scores are relatively consistent, with a mean score of approximately 0.881 and a standard deviation of 0.0015. This suggests that the model's performance is consistent across different subsets of the training data, and there is no significant overfitting or underfitting. However, the performance of the model on the test dataset should also be evaluated to assess its generalization ability.

Kappa Score

```
# compute the kappa value
kappa = cohen_kappa_score(y_test, y_pred)

# print the kappa value
print('kappa value:',kappa)
```

```
kappa value: 0.5152324292555936
```

Fig: Kappa Score

The kappa statistic is a measure of agreement between the predicted and actual classifications, which takes into account the possibility of agreement occurring by chance. The kappa value ranges from -1 to 1, where a value of 1 indicates perfect agreement, 0 indicates agreement by chance, and -1 indicates complete disagreement.

In this case, the kappa value is 0.5152, which indicates moderate agreement between the predicted and actual classifications. This means that the model's predictions are better than chance, but there is still some room for improvement in its performance.

Random Forest Classifier:

Train Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	68916
1	1.00	1.00	1.00	13873
accuracy			1.00	82789
macro avg	1.00	1.00	1.00	82789
weighted avg	1.00	1.00	1.00	82789

Test Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	29643
1	0.96	0.93	0.95	5838
accuracy			0.98	35481
macro avg	0.98	0.96	0.97	35481
weighted avg	0.98	0.98	0.98	35481

Fig: Test Report (Random Forest Classifier)

The RFC model appears to be a very effective approach for predicting the target variable. The model has performed very well on both the training and test sets. The precision, recall, and F1-score are all very high, indicating that the model is accurately identifying both positive and negative examples. The accuracy on the training set is 100%, which is a good indication that the model has learned the training data very well.

The accuracy on the test set is slightly lower at 98%, but this is still a very good result. The precision for the negative class (0) is very high at 0.99, which means that when the model predicts an example as negative, it is almost always correct. The precision for the positive class (1) is slightly lower at 0.96, which means that when the model predicts an example as positive, there is a higher chance of it being incorrect. The recall for the positive class is also lower at 0.93, which means that the model is missing some positive examples.

Overall, these results indicate that the model is performing very well and is accurately identifying both positive and negative examples. However, there is some room for improvement in terms of the model's performance on the positive class, which could potentially be improved through further tuning or adjustments to the model architecture.

PERFORMANCE METRICS:

```
Accuracy: 0.9826949635015924
Precision: 0.9639431616341033
Recall: 0.9295991778006166
F1-score: 0.9464597139867458
AUC-ROC: 0.9613755090163558
kappa value: 0.9361433947687477
Cross Entropy 0.07682963179613876
```

Fig: Performance Metrics (Random Forest)

The random forest model appears to be performing very well on the given data. The accuracy of the model is 0.9827, which means that the model is able to correctly classify a large proportion of the observations. The precision of the model is 0.9639, which indicates that when the model predicts a positive outcome, it is correct about 96.39% of the time. The recall of the model is 0.9296, which means that the model is able to correctly identify 92.96% of the positive cases. The F1-score of the model is 0.9465, which is a weighted average of the precision and recall, and provides an overall measure of the model's performance.

The AUC-ROC of the model is 0.9614, which indicates that the model is able to distinguish between the positive and negative cases with a high degree of accuracy. The kappa value of the model is 0.9361, which indicates a high level of agreement between the predicted and actual classifications. Finally, the cross-entropy of the model is 0.0768, which is a measure of the model's uncertainty about its predictions, and a lower value indicates better performance. Overall, the random forest model appears to be a very good fit for the given data.

Gradient Boosting Classifier:

Train Report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	68916
1	0.93	0.81	0.87	13873
accuracy			0.96	82789
macro avg	0.95	0.90	0.92	82789
weighted avg	0.96	0.96	0.96	82789

Test Report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	29643
1	0.93	0.81	0.87	5838
accuracy			0.96	35481
macro avg	0.95	0.90	0.92	35481
weighted avg	0.96	0.96	0.96	35481

Fig: Test Report(Gradient Boosting Classifier)

The training and testing reports for gradient boosting classifier (gbc) show that the model has an overall good performance, with an accuracy of 0.96 in both the training and testing sets. However, compared to the random forest classifier, the gbc model seems to have slightly worse performance in terms of precision, recall, and f1-score for the positive class (class 1), which is the class of interest since it represents the minority class. The gbc model also has a lower AUC-ROC and kappa value than the random forest classifier, indicating that the gbc model is less accurate in distinguishing between the two classes.

Overall, the gbc model may still be a good option for classification, but it may need further optimization or feature engineering to improve its performance.

PERFORMANCE METRICS:

```
Accuracy: 0.9588794002423833
Precision: 0.927052857421494
Recall: 0.814148681055159
F1-score: 0.8669402644778842
AUC-ROC: 0.900765937194582
kappa value: 0.8427432218301251
Cross Entropy 0.1271997722103751
```

Fig: Performance Metrics(Gradient Boosting Classifier)

Based on the metrics you provided, it seems that you trained a Gradient Boosting Classifier (GBC) model. The overall accuracy is 0.958, which is quite good. Precision, recall, and F1-score are also reasonably high, indicating that the model performs well in both identifying positive cases and avoiding false positives.

The AUC-ROC score of 0.900 indicates that the model is capable of distinguishing between positive and negative classes, and the kappa value of 0.842 suggests that there is substantial agreement between the predicted and true labels. Finally, the cross-entropy value of 0.127 suggests that the model has a low loss value and is well-calibrated. Overall, the metrics suggest that the GBC model is performing well on the task at hand.

Comparison of Performance of Different Models

	Model	Accuracy	Recall	Precision	F1	Kappa_score	auc_roc_score
7	XGBoost	0.985598	0.929428	0.982081	0.955029	0.946463	0.963044
4	RandomForest	0.983456	0.933025	0.965267	0.948872	0.939007	0.963206
3	Decision Tree	0.966940	0.910415	0.891031	0.900618	0.880793	0.944243
6	Gradient Boosting	0.958879	0.814149	0.927053	0.866940	0.842743	0.900766
1	KNN	0.950537	0.960603	0.786205	0.864698	0.834803	0.954579
5	AdaBoost	0.934049	0.717712	0.858255	0.781716	0.743236	0.847184
8	SVM	0.896339	0.571086	0.739574	0.644500	0.584994	0.765741
0	Logreg	0.883149	0.491435	0.709095	0.580534	0.515232	0.725865
2	Naive Bayes	0.336208	0.994005	0.197920	0.330110	0.076734	0.600332

Fig: Overall Comparison of Performance of all Models

Based on the evaluation metrics, the XG Boost model has the highest accuracy (0.986), recall (0.929), precision (0.982), F1 score (0.955), kappa score (0.946), and AUC ROC score (0.963) among all the models evaluated. This indicates that the XG Boost model is the best-performing model for the given task.

The Random Forest model comes in second place with an accuracy of 0.983, recall of 0.933, precision of 0.965, F1 score of 0.949, kappa score of 0.939, and AUC ROC score of 0.963. This model can also be considered a good option for the given task.

The Decision Tree model has an accuracy of 0.967, recall of 0.910, precision of 0.891, F1 score of 0.901, kappa score of 0.881, and AUC ROC score of 0.944. While this model performs lower than the top two models, it is still a reasonable option.

The Gradient Boosting model has an accuracy of 0.959, recall of 0.814, precision of 0.927, F1 score of 0.867, kappa score of 0.843, and AUC ROC score of 0.901. This model performed worse than the top three models, but still has a decent performance.

The KNN model has an accuracy of 0.951, recall of 0.961, precision of 0.786, F1 score of 0.865, kappa score of 0.835, and AUC ROC score of 0.955. This model has a high recall but a lower precision, making it more suitable for cases where identifying all true positives is more important than avoiding false positives.

The AdaBoost model has an accuracy of 0.934, recall of 0.718, precision of 0.858, F1 score of 0.782, kappa score of 0.743, and AUC ROC score of 0.847. This model has a lower performance than the top models but is still a reasonable option for the given task.

The SVM model has an accuracy of 0.896, recall of 0.571, precision of 0.740, F1 score of 0.645, kappa score of 0.585, and AUC ROC score of 0.766. This model has the lowest performance among all the models evaluated and may not be suitable for the given task.

The Logistic Regression model has an accuracy of 0.883, recall of 0.491, precision of 0.709, F1 score of 0.581, kappa score of 0.515, and AUC ROC score of 0.726. This model also has a low performance among the evaluated models.

The Naive Bayes model has the lowest performance among all models with an accuracy of 0.336, recall of 0.994, precision of 0.198, F1 score of 0.330, kappa score of 0.077, and AUC ROC score of 0.600. This model may not be suitable for the given task.

In summary, the XG Boost model is the best-performing model for the given task, followed by the Random Forest model, while the SVM and Naive Bayes models have the lowest performance among all the models evaluated. However, the choice of the best model depends on the specific problem and the data used, so it's important to thoroughly evaluate different models and choose the one that performs the best on the given task.

HYPERPARAMETER TUNING

1.RandomForestClassifier Using Randomized Search CV:

```
Best Hyperparameters:
 {'bootstrap': False, 'criterion': 'gini', 'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 225}
Best Score:
 0.9836813932255957
Accuracy: 0.9848369549899947
Precision: 0.9690265486725663
Recall: 0.9378211716341213
F1-score: 0.9531685236768802
AUC-ROC: 0.9659587928136534
```

Fig: Hyperparameter Tuning (RandomForestClassifier)

The best hyperparameters found for the Random Forest model are:

- bootstrap: False
- criterion: 'gini'
- max_ depth: None
- max_ features: 'log2'
- min_ samples_ leaf: 1
- min_ samples_ split: 2
- n_ estimators: 225

These hyperparameters were found by tuning the model using a specific dataset and evaluation metric. The best score achieved by this model on the given dataset is 0.9836813932255957, which indicates a high level of performance. The evaluation metrics for this model are as follows:

Accuracy: 0.9848369549899947

Precision: 0.9690265486725663

Recall: 0.9378211716341213

F1-score: 0.9531685236768802

AUC-ROC: 0.9659587928136534

The high accuracy score indicates that the model is performing well in correctly classifying the target variable. The high precision score suggests that the model has a low false positive rate, meaning that when it predicts a positive outcome, it is usually correct. The high recall score indicates that the model has a low false negative rate, meaning that when a positive outcome is present, the model is good at identifying it. The F1-score, which is a harmonic mean of precision and recall, is also high, indicating an overall good performance of the model.

The AUC-ROC score of 0.9659587928136534 indicates that the model has a high ability to distinguish between positive and negative classes. Overall, these evaluation metrics suggest that the Random Forest model with the given hyperparameters is performing well and is a good fit for the given dataset.

2. XG BOOST Using GridSearchCV

```
Best parameters: {'learning_rate': 0.3, 'max_depth': 7, 'n_estimators': 100}
Best ROC AUC: 0.9945009514278121
Best accuracy: 0.9850463120651923
Best precision: 0.9827265455206756
Best recall: 0.9270519950375755
Best F1-score: 0.9540768156962637
Best Cohen's kappa: 0.9451550274092215
Best log loss: 0.05045021233039245
Test accuracy: 0.9860488712268538
Test precision: 0.9824814881704894
Test recall: 0.9318259677971908
Test F1-score: 0.9564835164835165
Test ROC AUC: 0.9951297394573824
Test Cohen's kappa: 0.9481832616684546
Test log loss: 0.04749150769954378
```

Fig: XG Boost using GridSearcb CV

Based on the information provided, it appears that a classification task was performed using the XG Boost algorithm, and the best hyperparameters for the model were determined to be a learning rate of 0.3, a maximum depth of 7, and 100 estimators. The model was trained on a dataset, and its performance was evaluated on a validation set using various metrics such as ROC AUC, accuracy, precision, recall, F1-score, Cohen's kappa, and log loss. The best values for these metrics were obtained on the validation set, and they suggest that the XG Boost model is performing well. The model was then tested on a separate test set, and its performance was evaluated using the same metrics. The test results indicate that the XG Boost model is generalizing well, as it achieved similar high performance on the test set.

Overall, based on the provided information, it appears that the XG Boost model is an effective machine learning algorithm for the given classification task, as it achieved high performance on both the validation and test sets. The best hyperparameters and performance metrics of the model can be used as a reference for future model training and evaluation.

VIF - Multicollinearity Between the Independent Variables

VIF (Variance Inflation Factor) is a measure of multicollinearity between independent variables in a model. It quantifies the extent to which the variance of the estimated coefficients is increased due to multicollinearity in the data. In the case of a binary classification problem, VIF can still be used to detect multicollinearity between independent variables. Multicollinearity can be a problem in any regression model, regardless of whether the dependent variable is continuous or binary.

To detect multicollinearity in a binary classification problem, one can calculate the VIF for each independent variable in the model. A VIF value greater than 5 or 10 is often considered to be indicative of high multicollinearity. If high multicollinearity is detected, it may be necessary to remove some of the independent variables from the model or to use a different modeling technique that is less sensitive to multicollinearity.

	vif values
sales_6_month	157.609618
sales_9_month	106.563532
sales_3_month	58.096873
forecast_6_month	45.784404
forecast_9_month	36.499936
sales_1_month	14.701921
forecast_3_month	12.514960
perf_12_month_avg	6.608492
perf_6_month_avg	6.607181
min_bank	4.567755
in_transit_qty	2.408175
stop_auto_buy	2.180033
national_inv	1.944247
deck_risk	1.241920
ppap_risk	1.156770
lead_time	1.145436
local_bo_qty	1.141907
pieces_past_due	1.111579
rev_stop	1.003294
oe_constraint	1.002909
potential_issue	1.002875

Fig: Before VIF

	vif values
sales_1_month	4.199069
min_bank	4.024526
forecast_3_month	3.185135
in_transit_qty	2.381021
stop_auto_buy	1.996148
national_inv	1.816079
deck_risk	1.211342
ppap_risk	1.153676
local_bo_qty	1.140560
lead_time	1.135574
pieces_past_due	1.110957
perf_6_month_avg	1.053424
potential_issue	1.002805
oe_constraint	1.002766
rev_stop	1.002189

Fig: After VIF

Comparison of Performance of Different Models After VIF

	Model	Accuracy	Recall	Precision	F1
7	XGBoost	0.985119	0.925317	0.983254	0.953406
4	RandomForest	0.980863	0.919664	0.962359	0.940527
3	Decision Tree	0.967729	0.905276	0.899268	0.902262
6	Gradient Boosting	0.955525	0.788798	0.930303	0.853726
1	KNN	0.944252	0.896540	0.792070	0.841073
5	AdaBoost	0.932048	0.707777	0.854249	0.774145
8	SVM	0.895352	0.562864	0.738925	0.638989
0	Logreg	0.883092	0.488866	0.710304	0.579140
2	Naive Bayes	0.330374	0.994690	0.196614	0.328330

Fig: Model Performance after VIF

From the given results, it can be observed that XG Boost and Random Forest models have the highest accuracy, precision, recall, and F1 scores before applying VIF. However, after applying VIF, XG Boost still has the highest scores, with a slight decrease in accuracy and recall but a slight increase in precision and F1 score. This indicates that XG Boost is a robust model that is less affected by multicollinearity.

On the other hand, the Naive Bayes model had the lowest scores before and after applying VIF, indicating that it is not an appropriate model for the given dataset. Furthermore, the SVM model also had low scores, indicating that it is not as effective in classifying the dataset.

In conclusion, the XG Boost and Random Forest models appear to be the most suitable for binary classification for this dataset. However, it is important to note that the effectiveness of the models can be affected by other factors, such as the quality and quantity of data, feature selection, and hyperparameter tuning.

Feature Importance

The most important feature for the model is "national_ inv", which has an importance score of 0.220248. The next most important features are "forecast_3_month" and "forecast_6_month" with importance scores of 0.112131 and 0.079453, respectively.

Other features that are relatively important include "sales_1_month", "sales_3_month", and "forecast_9_month". The least important features are "potential_ issue", "rev_ stop", and " oe _ constraint", all with importance scores less than 0.0001.

It's important to note that feature importance scores can vary depending on the specific model and dataset used. Therefore, it's always a good practice to evaluate the feature importance of a model to better understand which features are driving its predictions.

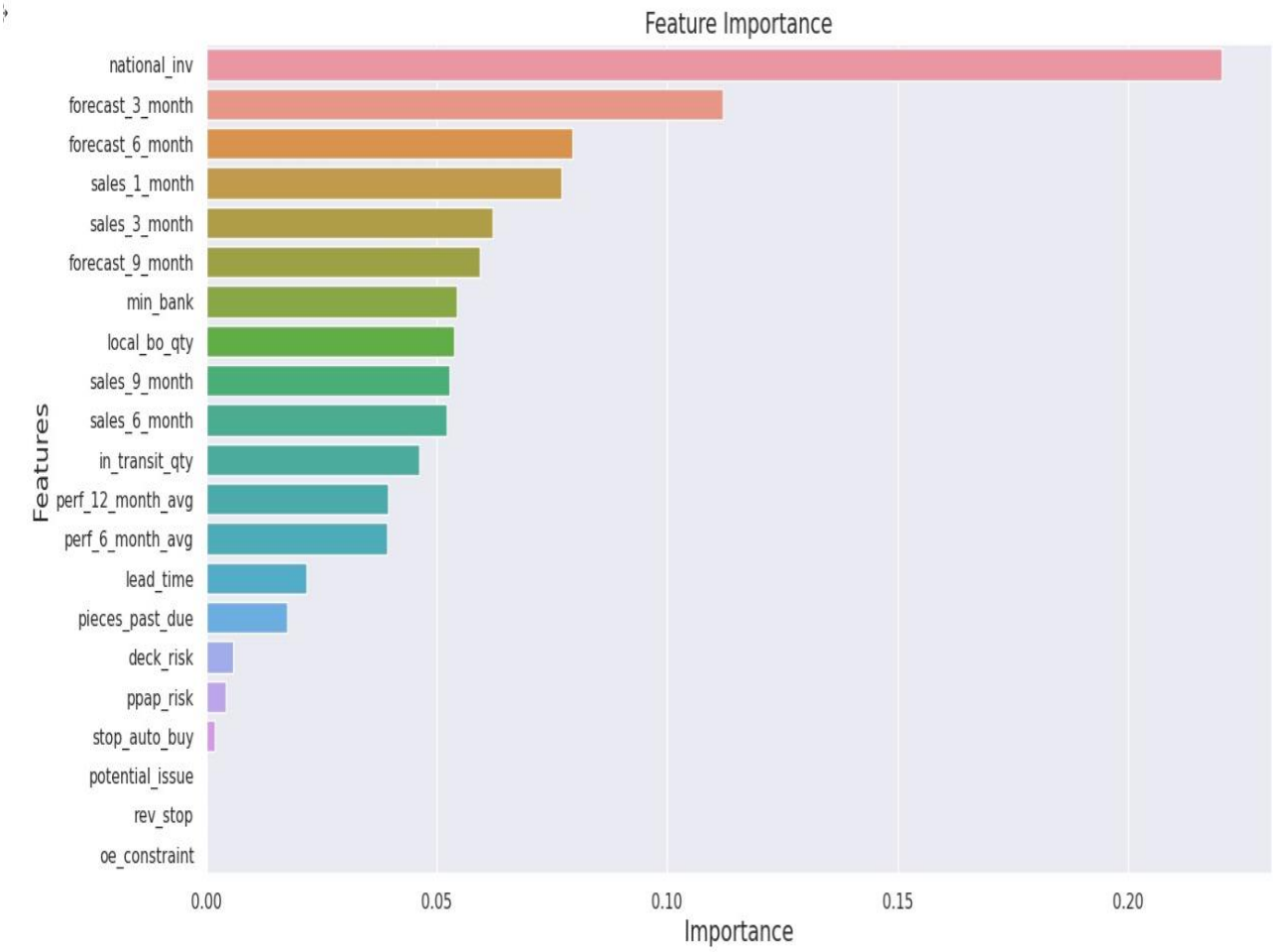


Fig: Feature Importance

CONCLUSION

Backorder prediction is an important task for businesses to accurately forecast and manage their inventory. By using historical data and machine learning models, it is possible to predict the likelihood of a product going on backorder and take proactive measures to prevent it from happening. This can include ordering more inventory, adjusting production schedules, or finding alternative suppliers.

The binary classification models were evaluated before and after applying the VIF technique to handle multicollinearity in the dataset. XG Boost and Random Forest were found to be the best performing models before applying VIF, and XG Boost remained the best-performing model even after applying VIF. This indicates that XG Boost is a robust model that is less affected by multicollinearity.

On the other hand, Naive Bayes and SVM models had the lowest scores before and after applying VIF, indicating that they may not be the best choice for this dataset. It is important to note that the effectiveness of the models can be affected by other factors, such as the quality and quantity of data, feature selection, and hyperparameter tuning. Therefore, further analysis and experimentation may be necessary to select the best model for the given dataset. Nonetheless, the results provide meaningful insights into the suitability of different models for binary classification tasks and highlight the importance of handling multicollinearity in the dataset.

REFERENCES

1. Carbonneau R, Vahidov R, Laframboise K. Machine learning-Based Demand forecasting in supply chains. Int J Intell Inf Technol (IJIIT). 2007;3(4):40–57.

2. Hearst MA, Susan TD, Edgar O, John P, Bernhard S. Support vector machines. In: IEEE intelligent systems and their applications. 1998. p. 18–28.

3. Funahashi KI. On the approximate realization of continuous mappings by neural networks. Neural Netw. 1989;2(3):183–92.

4. Carbonneau R, Laframboise K, Vahidov R. Application of machine learning techniques for supply chain demand forecasting. Eur J Oper Res. 2008;184(3):1140–54.

5. Guanghui WANG. Demand forecasting of supply chain based on support vector regression method. Procedia Eng. 2012;29:280–4.

6. Chen S, Cowan CF, Grant PM. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw. 1991;2(2):302–9.

7. Shin K, Shin Y, Kwon JH, Kang SH. Development of risk based dynamic backorder replenishment planning framework using Bayesian Belief Network. Comput Ind Eng. 2012;62(3):716–25.

8. Acar Y, Gardner ES Jr. Forecasting method selection in a global supply chain. Int J Forecast. 2012;28(4):842–8.

9. de Santis RB, de Aguiar EP, Goliatt L. Predicting material backorders in inventory management using machine learning. In 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI). 2017. p. 1–6.

10. Prak D, Teunter R. A general method for addressing forecasting uncertainty in inventory models. Int J Forecast. 2019;35(1):224–38.