SPRING 2023
ISE 5984 Statistical Learning and Data Science
Project Report

Supply Chain Backorder Prediction

By
Manichandra Majji
Madhulika Sable
Maneesh Reddy

1. Introduction

In today's dynamic business environment, companies face the challenge of balancing customer demand with inventory management to ensure efficient supply chain operations. One such challenge is backorders, which occur when a product is out of stock or unavailable at the time of customer order placement. Backorders can result in dissatisfied customers, lost sales, and increased operational costs for businesses.

While increasing inventory levels may seem like a logical solution to preventing backorders, it is not always feasible for businesses. This is because increasing inventory levels can lead to additional expenses such as storage costs, which can significantly impact the bottom line. As such, businesses need an efficient solution to prevent backorders while maintaining optimal inventory levels.

Predictive models offer a solution to this problem by identifying products at a higher risk of going into backorder. By analyzing historical data and using advanced analytical techniques, businesses can accurately predict which products are likely to go out of stock or become unavailable in the future. This allows them to proactively plan for potential disruptions in their supply chain, reducing the risk of backorders and minimizing their negative impacts.

In this work, we aim to develop a binary classification model that can identify products at a higher risk of going into backorder ahead of time. By leveraging machine learning algorithms, we will analyze historical data on product demand, sales, and supply chain operations to identify patterns and trends that may lead to backorders. We will also identify various features that may impact the risk of backorders. This approach can improve customer satisfaction and reduce operational costs, making it a valuable investment for businesses.

2. Dataset

The dataset used in this data science project was the 'Predict Product Backorders' dataset, which is publicly available on Kaggle. The dataset comprises 23 features, of which 15 are numerical and 8 are categorical, including the target variable ‘went on backorder.’ The features

in the dataset provide information on product characteristics, inventory levels, and demand patterns. The target variable indicates whether a product went on backorder or not. The dataset contains approximately 1.9 million data points of which 99.27% of data points belong to the target class 'No' and 0.72% belong to the target class 'Yes'. This implies that the class ratio of minority class to majority class is 1:137 and we have a highly imbalanced dataset. Table 1 describes the features of the dataset.

Feature Name	**Description**	**Data type**
sku	Stock Keeping Unit	Numerical
national_inv	Current inventory level of component	Numerical
lead_time	Transit time	Numerical
in_transit_qty	In transit quantity	Numerical
forecast_3_month forecast_6_month forecast_9_month	Sales forecast for the net 3, 6, 9 months	Numerical
sales_1_month sales_3_month sales_6_month sales_9_month	Sales quantity for the prior 1, 3, 6, 9 months	Numerical
min_bank	Minimum recommended amount in stock	Numerical
potential_issue	Indicator variable noting potential issue with item	Categorical
pieces_past_due	Parts overdue from source	Numerical
perf_6_month_avg perf_12_month_avg	Source performance in the last 6 and 12 months	Numerical
local_bo_qty	Amount of stock overdue	Numerical
deck risk, oe constraint, ppap risk, stop auto buy, rev stop	Different Flags (Yes or No) set for the product	Categorical
went_on_backorder	Product went on backorder	Categorical

Table 1: Dataset Description

3. Exploratory Data Analysis

	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month	forecast_9_month	sales_1_month
count	1.929935e+06	1.814318e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06
mean	4.965683e+02	7.878627e+00	4.306440e+01	1.785399e+02	3.454659e+02	5.066067e+02	5.536816e+01
std	2.957343e+04	7.054212e+00	1.295420e+03	5.108770e+03	9.831562e+03	1.434543e+04	1.884377e+03
min	-2.725600e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	4.000000e+00	4.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	1.500000e+01	8.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
75%	8.000000e+01	9.000000e+00	0.000000e+00	4.000000e+00	1.200000e+01	2.000000e+01	4.000000e+00
max	1.233440e+07	5.200000e+01	4.894080e+05	1.510592e+06	2.461360e+06	3.777304e+06	7.417740e+05

sales_3_month	sales_6_month	sales_9_month	min_bank	pieces_past_due	perf_6_month_avg	perf_12_month_avg	local_bo_qty
1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06	1.929935e+06
1.746639e+02	3.415653e+02	5.235771e+02	5.277637e+01	2.016193e+00	-6.899870e+00	-6.462343e+00	6.537039e-01
5.188856e+03	9.585030e+03	1.473327e+04	1.257968e+03	2.296112e+02	2.659988e+01	2.588343e+01	3.543230e+01
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-9.900000e+01	-9.900000e+01	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.300000e-01	6.600000e-01	0.000000e+00
1.000000e+00	2.000000e+00	4.000000e+00	0.000000e+00	0.000000e+00	8.200000e-01	8.100000e-01	0.000000e+00
1.500000e+01	3.100000e+01	4.700000e+01	3.000000e+00	0.000000e+00	9.600000e-01	9.500000e-01	0.000000e+00
1.105478e+06	2.146625e+06	3.205172e+06	3.133190e+05	1.464960e+05	1.000000e+00	1.000000e+00	1.253000e+04

Figure 1: Descriptive Statistics

Upon calculating the descriptive statistics, it was found that the majority of the numerical features, with the exception of ‘perf_6_month_avg’ and ‘perf_12_month_avg’, might be skewed to the right. This claim can be proved by plotting probability density functions. Specifically, the mean values of these features exceed the 50th or 75th percentile, indicating a significant presence of high values. Moreover, the minimum and maximum values of ‘perf_6_month_avg’ and ‘perf_12_month_avg’ are -99 and 1, respectively, which suggests that missing values were substituted with -99. Another observation is that there is a considerable gap between the 75th percentile value and the maximum value for each feature, indicating there might be the presence of outliers which needs further investigation. These outliers may have a substantial impact on the overall analysis and should be addressed in subsequent modeling efforts. Furthermore, only one feature, ‘lead_time’, has null values which account for ~5% of the data, and the last two rows in the dataset contain all Nan values, which should be dropped.

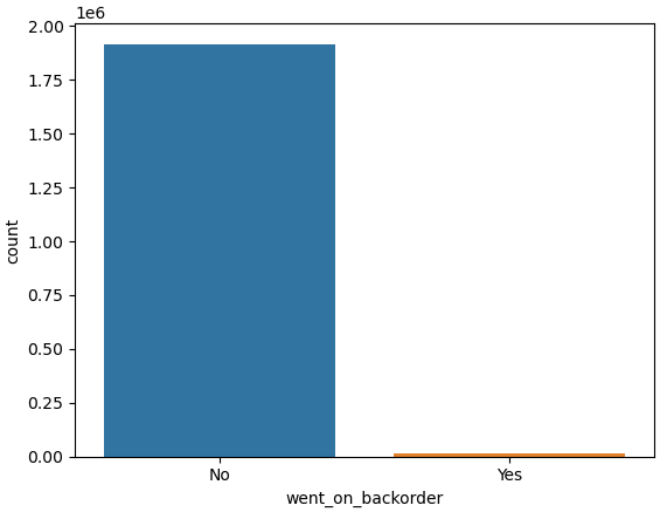
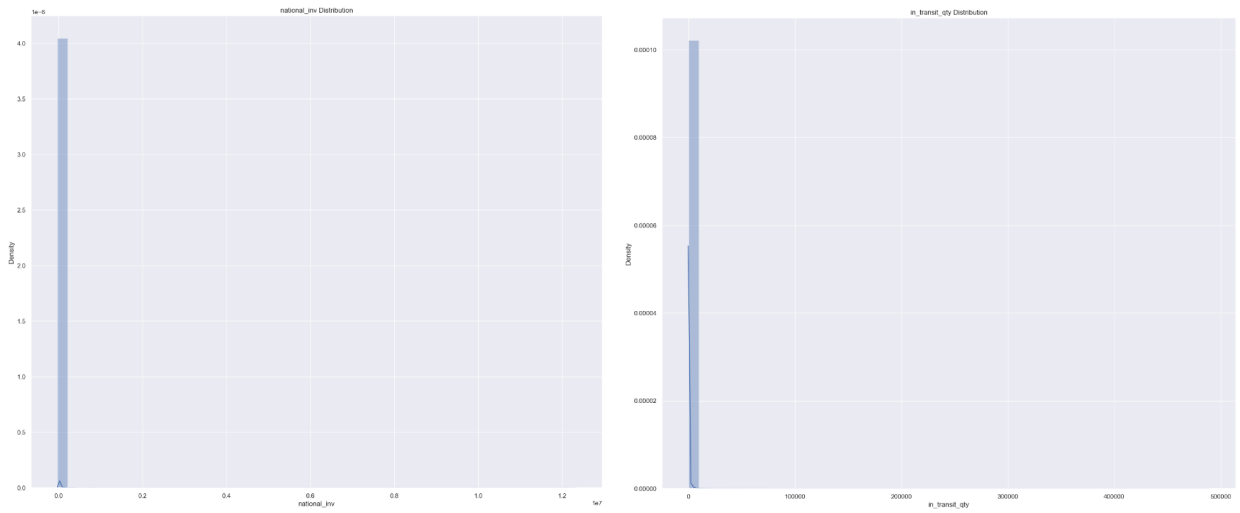


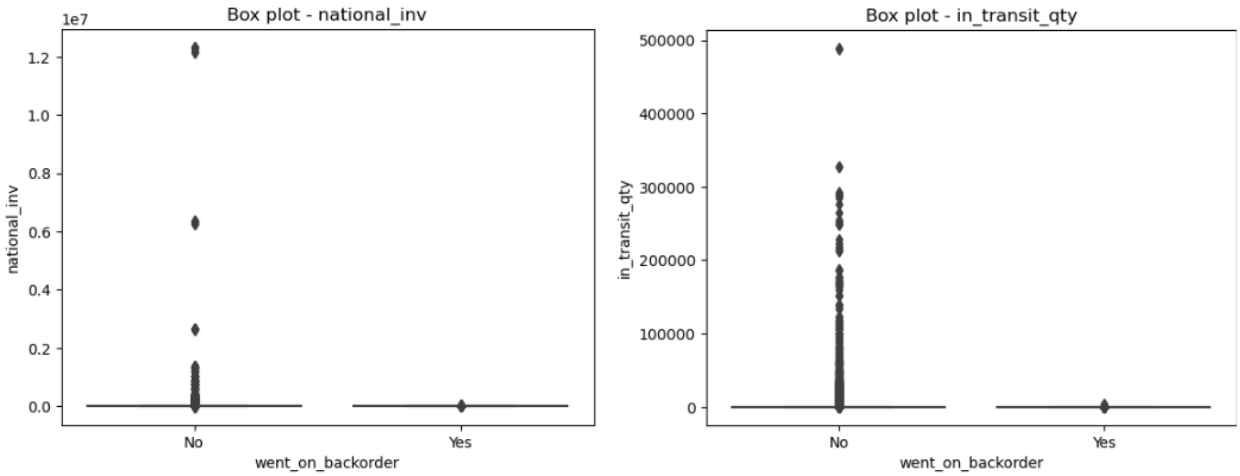
Figure 2: Class ratio of the target feature

As mentioned earlier, there is a very high-class imbalance (1:137) in the target feature of the dataset. Since, there are only 13981 data points for class ‘Yes’ and 1915954 data points for class ‘No’, it does not seem to be a good idea to undersample because that will result in very few data points available for training the model. We might have an issue where the model will not be able to learn enough from the train set to be able to predict on the test set. So, oversampling the data is a better idea in this case, which will be discussed in the later section.



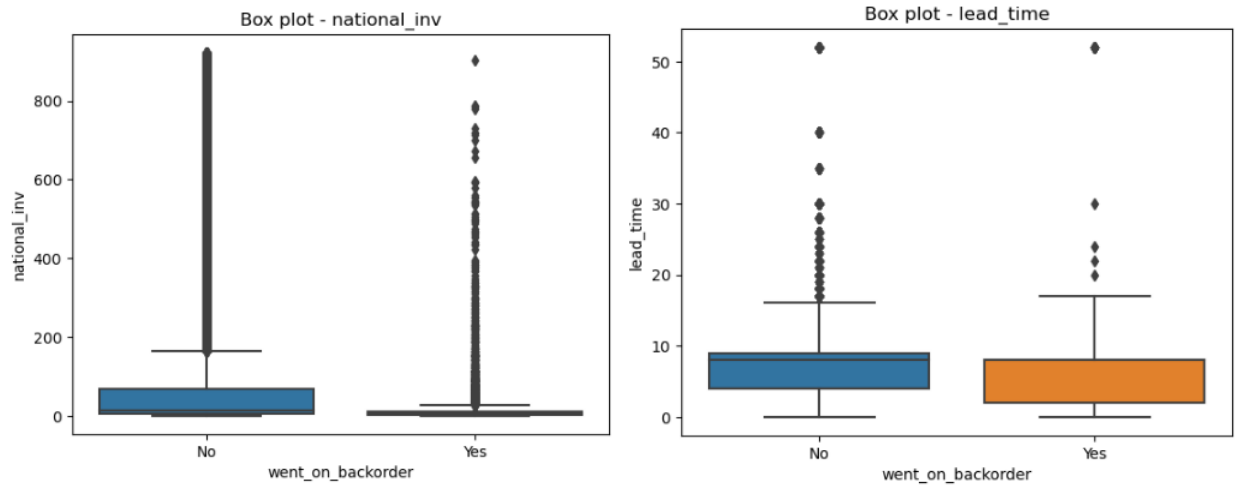
Figures 3,4: PDFs for ‘national_inv’ and ‘in_transit_qty’

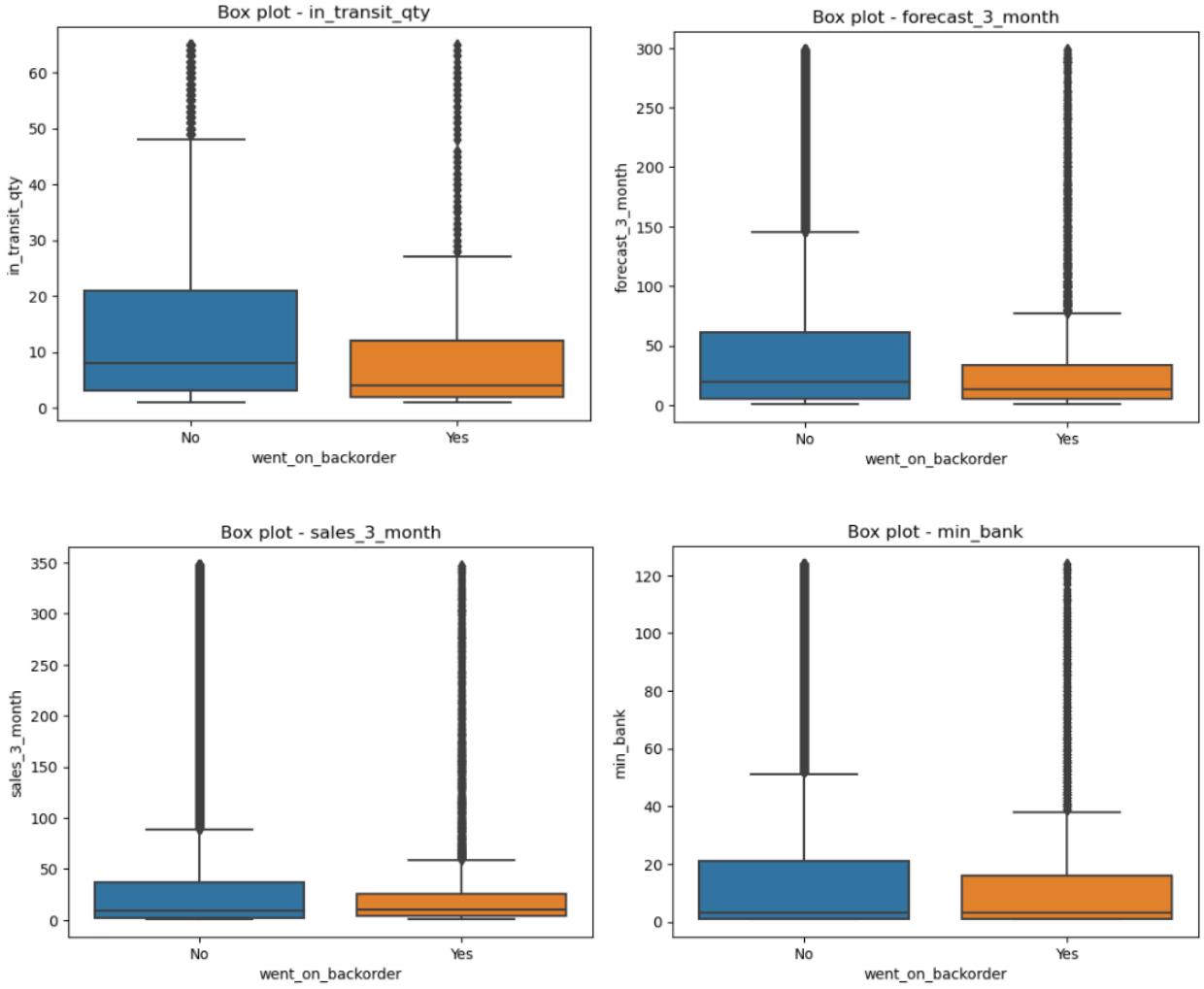
As discussed earlier, upon conducting a univariate analysis with respect to the class label, it is evident that a majority of the numerical features exhibit right skewness. As evidenced by the PDFs in Figures 3 and 4, both the ‘in_transit_quantity’ and ‘national_inv’ features display the right skewness in their distributions.



Figures 5,6: Box plots of ‘national_inv’ and ‘in_transit_qty’

The box plots presented above in Figures 5 and 6 were generated for the entire dataset. However, due to the high variance in the range of values for each feature, the plots are not visible, and it is very tough to infer from them. So, only the values between the 5th and 95th percentile are included to avoid this issue and are shown below.

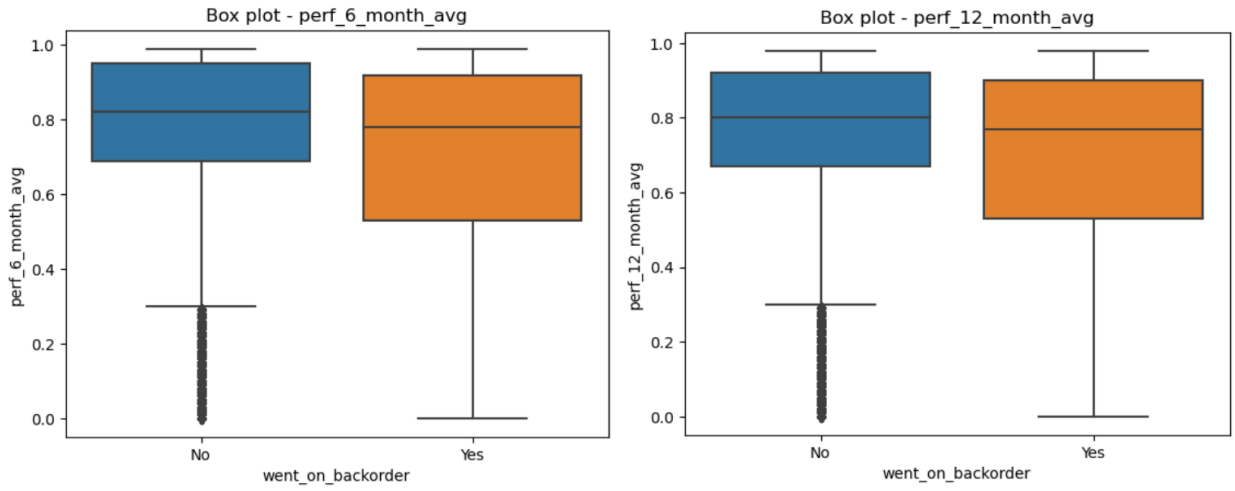




Figures 7-12: Box plots of various features with values between the 5th and 95th percentile

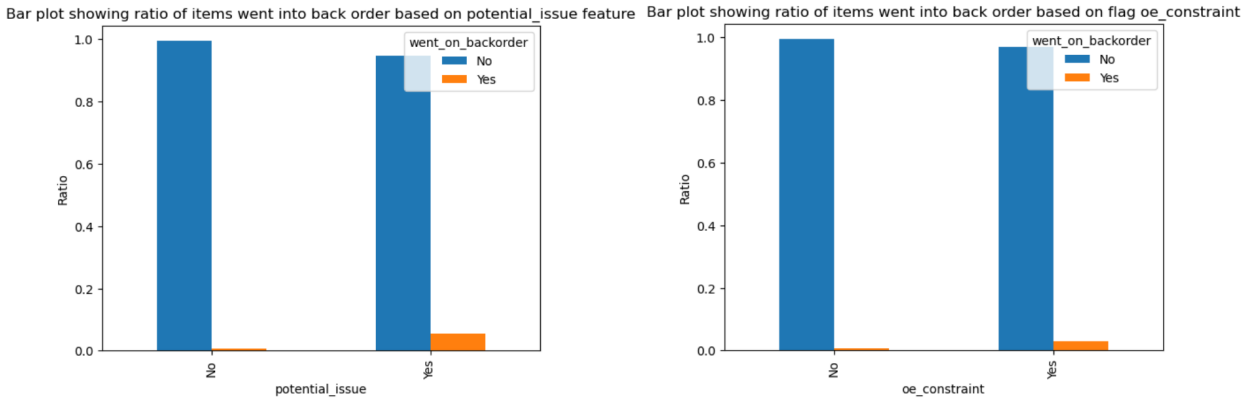
When considering the data between the 5th and 95th percentile values, the interquartile range (IQR) becomes evident as seen in Figures 7-12. This finding strengthens the earlier observation that the data is highly skewed toward the right. Specifically, the lower the value of 'national_inv', 'lead_time', and 'in_transit_qty', the higher the likelihood that a product will be back-ordered. This trend is evident from the box plots of these features. Reducing inventory levels increases the likelihood of products going out of stock during a sudden surge in demand, as there is limited stock available to meet the increased demand. Similarly, shorter lead times can also result in lower inventory levels, as products can be quickly received and replenished.

Hence, it is accurate to state that lower inventory levels and lead times can result in products going out of stock.



Figures 13, 14: Box plots of various features with values between the 5th and 95th percentile

The boxplot for features 'perf_6_month_avg' and 'perf_12_month_avg' in Figures 13 and 14 illustrate that the values are confined within the range of 0 to 1 when only considering the values between the 5th and 95th percentile. Since the maximum allowable value for these features is 1 and there are observations of -99, it can be deduced that these values represent missing data, as the next highest value after -99 is 0. This observation holds true for both features.



Figures 15, 16: Bar plots of 'potential_issue' and 'oe_constraint'

The bar plots in Figures 15 and 16 reveal that when there is a potential issue with a product, it does not go to backorder 94.53% of the time, while when there is no potential issue, the product does not go to backorder 99.25% of the time. Thus, the probability of a product going to backorder is 7.5 times higher when there is a potential issue as compared to when there is none. Similarly, when 'oe_constraint' is set to 'No' the product did not go to backorder 99.27% of the time, whereas when it is set to 'Yes' the product did not go to backorder 96.91% of the time. Thus, the product going to backorder is 4 times higher when 'oe_constraint' is set to 'Yes' than when it is set to 'No.' The other flags that are categorical do not provide any significant information in predicting whether a product goes to backorder, as the distribution of classes for these flags is almost equal between both classes. Therefore, these flags do not help in distinguishing between the two classes of the target feature.

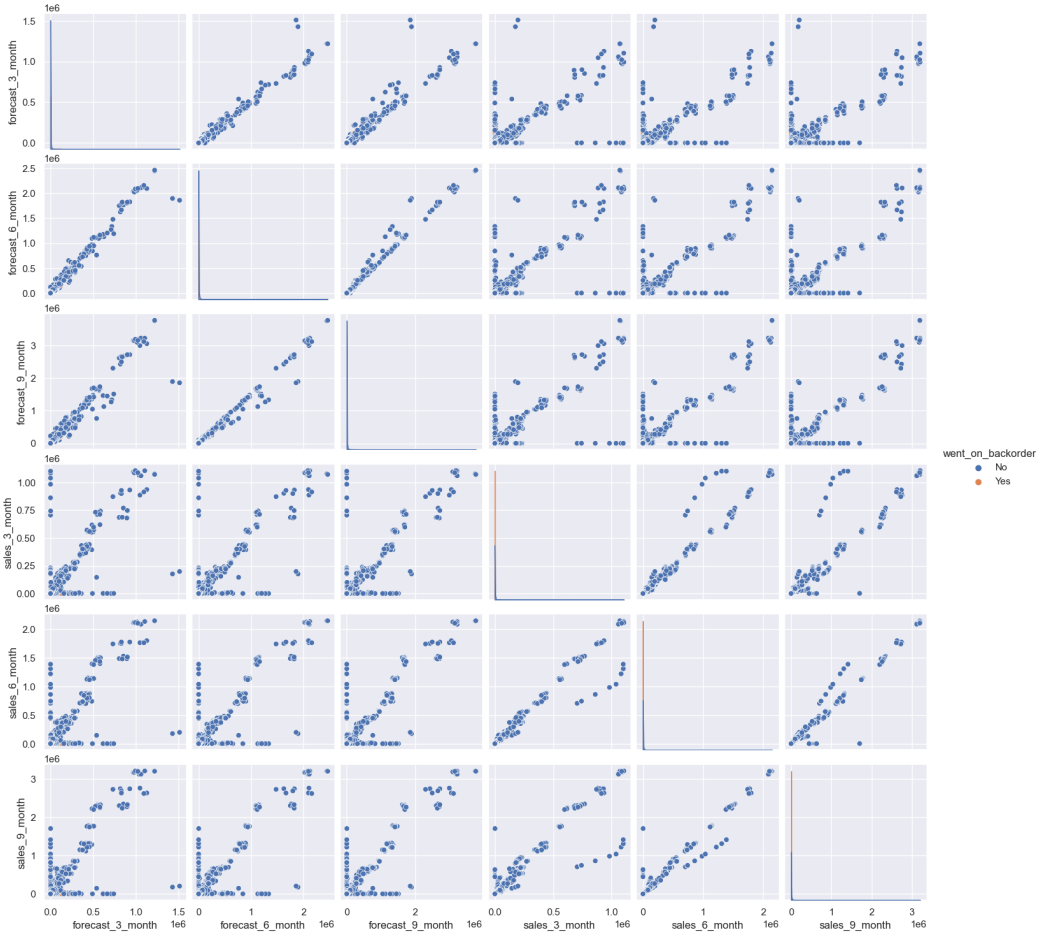


Figure 17: Pairplots of sales and forecast features

The pairplots shown in Figure 17 indicate a strong association between the likelihood of a product going into backorder and the product's forecast and sales values. This is in line with the expectation that products with low inventory levels are more likely to go into backorder. However, a few data points appear to deviate from this linear trend, which may be due to outliers. It is important to note that a product with a high sales value cannot have a significantly low forecast value and vice versa, so large discrepancies between these values may result in outliers. Strategies for addressing outliers are discussed in a later section on data cleaning.

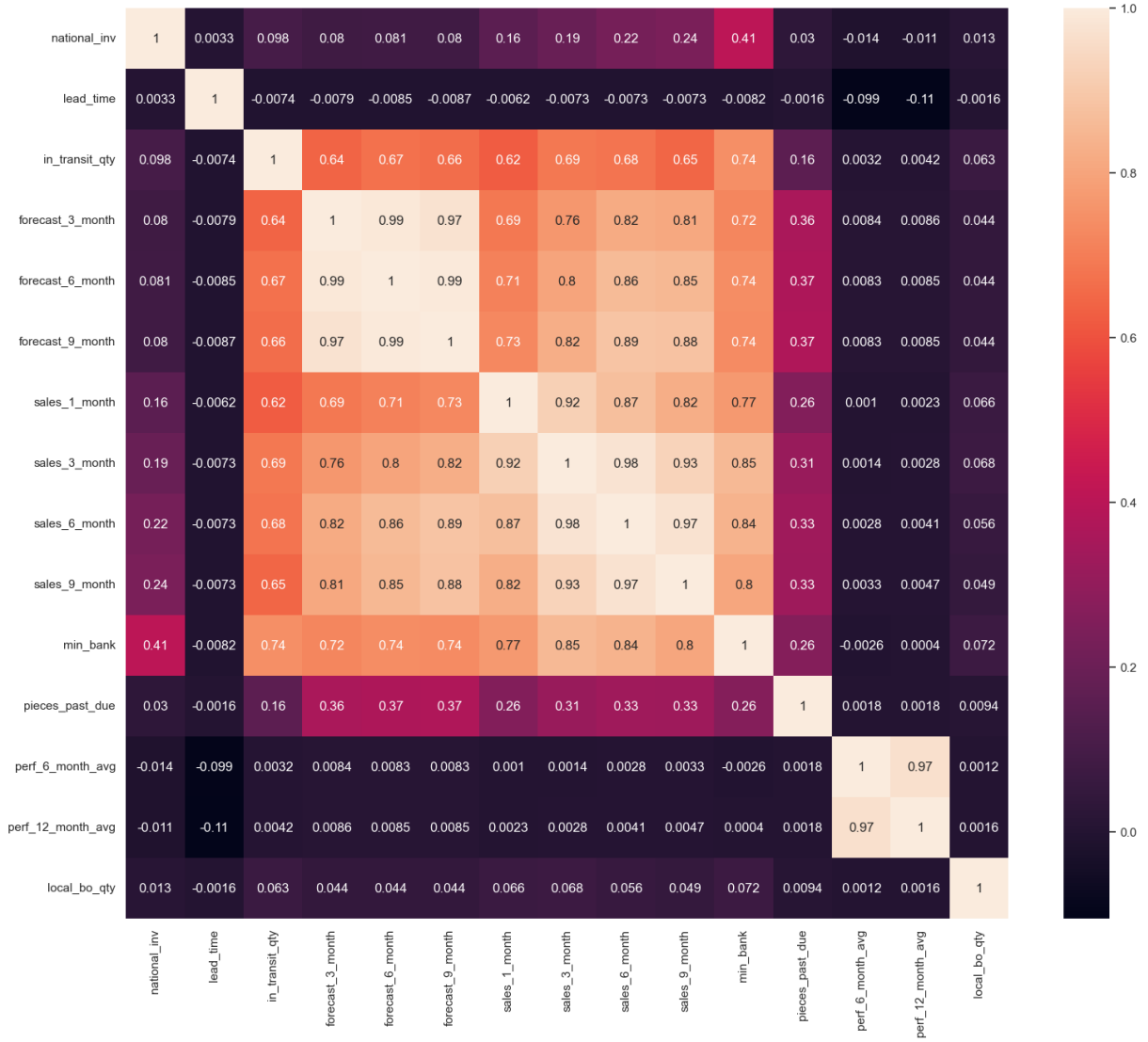


Figure 18: Correlation Matrix

As depicted in Figure 18, all of the significant correlations identified among the features are positive in nature. The features 'forecast_3_month', 'forecast_6_month', and 'forecast_9_month' are found to be extremely strongly correlated with each other, with a correlation coefficient of 0.99. Similarly, the features 'sales_1_month', 'sales_3_month', 'sales_6_month', and 'sales_9_month' are also found to be strongly correlated with each other, with correlation coefficients ranging from 0.82 to 0.98. The forecast and sales features are also correlated with each other, with a minimum correlation coefficient of 0.62, and a maximum correlation coefficient of 0.88. This correlation is evident, as high sales figures in the past often lead to higher forecasted values for the same product in the coming months, and vice versa. Finally, the feature 'min_bank', which represents the minimum recommended stock level, is found to be highly correlated with the sales and forecast features, as the stock levels in inventory are directly proportional to sales numbers.

4. Data Cleaning

During the data cleaning process, several steps were taken to clean the dataset. As discussed in the previous section, we have come to the conclusion that all the missing values in the features 'perf_6_month_avg' and 'perf_12_month_avg' have been replaced with -99. So, we replaced all instances of -99 in the features 'perf_6_month_avg' and 'perf_12_month_avg' with NaN to ensure uniformity in the dataset. Now, we have missing values in three features namely 'lead_time', 'perf_6_month_avg' and 'perf_12_month_avg' and they only account for 5.9%, 7.6% and 7.2% of the entire dataset respectively. Since they do not account for a significant percentage of the entire dataset, they can be eliminated, but we have considered imputing them using mean to avoid the loss of information. As discussed in the previous section, though there are many data points outside the third quartile, not all of them are outliers. But, from the univariate analysis, it has been identified that there is a very large difference between the values of the 99th and 100th percentile for a majority of the numerical features, and all the data points above the 99th percentile have been dropped for these features. In addition, we have identified that there are a few potential outliers in the sales and forecast features from the

pairplots and have leveraged Mahalanobis distance to eliminate them. Finally, it has been identified that the features ‘pieces_past_due’ and ‘local_bo_qty’ have the same value for more than 95% of the dataset. These features might not add value to the modeling and as a result, have been dropped. These data-cleaning steps were crucial in ensuring the dataset was clean and ready for analysis.

5. Feature Engineering

Having learned that the majority of the numerical features are right-skewed, there is a need to transform them, especially for linear models. This is because the linear model assumes that the residuals (the difference between the predicted values and the actual values) are normally distributed. If the features are not normally distributed, the residuals may also not be normally distributed, and this can affect the accuracy of the model. Log transformation has been applied to these features to make them normally distributed. On the other side, this is not a requirement for models like decision trees and random forests, as these models are not sensitive to the distribution of the features. In addition, the categorical features were one-hot encoded to transform them to numerical form so that the machine learning algorithms can read them.

Finally, the numerical features have been standardized to have a mean of zero and a standard deviation of one. There are several reasons why standardizing the data is important:

Equal scale: Different features can have different scales or units, and this can affect the performance of some machine learning algorithms. For example, features that have larger values may have a larger impact on the outcome of the model, even if they are not necessarily more important.

Improving convergence: Some algorithms, such as gradient descent, converge faster when the data is standardized. This is because standardizing the data results in a better-conditioned optimization problem, meaning that the optimization algorithm can more easily find the optimal solution.

Reducing the impact of outliers: Outliers can have a significant impact on the model's performance, especially when the data is not standardized. Standardizing the data can reduce the impact of outliers.

6. Modeling

Before moving ahead to discuss the algorithms that have been used, we had to address the issue of class imbalance. It is important to make sure that both classes are balanced because class imbalance can introduce bias in the model. If a model is trained on imbalanced data, it may learn to predict the majority class more accurately and ignore the minority class, resulting in poor performance on the minority class. And if the model is biased towards the majority class, it may have a high overall accuracy but a low accuracy on the minority class, which can be critical. So, an oversampling technique called SMOTE (Synthetic Minority Over-sampling Technique) has been leveraged to address the issue of class imbalance. It works by generating synthetic samples for the minority class by interpolating between existing minority class samples. After oversampling, the number of data points in the training dataset is approximately 2.6 million with close to 1.3 million data points in each target class. On the flip side, given the large size of the training dataset, it takes forever to train the models with the computational resources available to us. To address this problem, reducing the data size seemed to be the best option. A sample of 200,000 data points was randomly selected, with an equal distribution of 100,000 data points for each target class to make it easier for the machine learning models to learn.

Five machine learning models were built and hyper-parameter tuning was performed using a 3-fold cross-validation approach. The models are discussed below.

6.1 Logistic Regression

Logistic regression works by modeling the probability of the binary classes as a function of the input features using a sigmoid function. The logistic function maps any input value to a value between 0 and 1, which can be interpreted as a probability. In order to tune the

hyper-parameters of a logistic regression model for a binary classification problem, several hyper-parameters have been considered, namely regularization strength (C) and solver. The regularization hyper-parameter controls the strength of the regularization applied to the model. A higher value of C leads to weaker regularization and may result in overfitting, while a lower value of C leads to stronger regularization and may result in underfitting. The solver hyper-parameter determines the optimization algorithm used to fit the logistic regression model. The regularization strength has been tuned for 5 values: 100, 10, 1.0, 0.1, 0.01 and the solver parameter has been tuned for 'newton-cg', 'lbfgs', and 'liblinear' solvers.

6.2 Decision Tree

Decision tree is a popular machine-learning model used for classification and regression tasks. In a binary classification problem, decision trees can be used to learn a set of rules that classify a given input into one of two classes. In order to tune the hyper-parameters of a decision tree model, maximum depth has been considered. The maximum depth of a decision tree refers to the maximum number of nodes that can exist between the root node and the leaf nodes. Increasing the maximum depth can increase the complexity of the model, but may also lead to overfitting. The maximum depth of a decision tree has been tuned over a range of values from 1 to 20 with a step size of 1.

6.3 Random Forest

Random Forest is a popular ensemble machine-learning algorithm that is widely used for classification and regression problems. Random forest builds a set of decision trees on randomly sampled subsets of the training data and aggregates the predictions of each tree to make a final prediction. The hyper-parameters that were tuned for random forest are the maximum depth and number of estimators. Maximum depth refers to the maximum depth of each decision tree and has been tuned over a range of values from 1 to 20 with a step size of 1. The number of estimators refers to the number of decision trees in the random forest and has been tuned for 3 values: 50, 100, and 150.

6.4 XGBoost

XGBoost (Extreme Gradient Boosting) is a popular machine-learning algorithm for both regression and classification problems. It is an ensemble learning method that combines multiple weak learners to form a strong learner. It is known for its execution speed and model performance.

6.5 Custom Ensemble

A custom ensemble model was built by utilizing the four aforementioned models. The final prediction was generated by taking the average of the predictions made by each model.

7. Results and Discussion

The objective is to minimize both false positives and false negatives since misclassifying a product as 'back ordered' when it is not, or vice versa, can result in significant production or shortage costs, respectively, and may also lead to loss of customers. Consequently, every misclassification carries a cost. Hence, the F-1 score is selected as the primary performance metric apart from accuracy to evaluate the model's performance.

Model	Train Accuracy	Test Accuracy	F-1 Score
Logistic Regression	0.830	0.839	0.662
Decision Tree	0.983	0.924	0.753
Random Forest	0.997	0.953	0.826
XGBoost	0.973	0.944	0.787
Custom Ensemble	0.986	0.941	0.787

Table 2: Performance of all the models

Table 2 displays the performance of the models following hyper-parameter tuning to maximize the F-1 score. The results indicate that logistic regression exhibited the weakest performance among all the models. This outcome can be attributed to the presence of a few correlated features, which typically pose a challenge for linear models. In contrast, the remaining four models exhibited similar train and test accuracies but varied in their F-1 scores. Random forest

achieved the highest performance, which is likely due to the ensemble nature of the algorithm and its use of bagging. By fitting individual decision trees to each bootstrap sample and averaging the results, bagging significantly reduces the variance of the random forest compared to the variance of individual decision trees. This results in superior performance when compared to the decision tree. XGBoost followed closely behind with the second-best performance, performing similarly to the custom ensemble model. Since the predictions of all four models were averaged to generate the custom ensemble's prediction, its performance exceeded that of logistic regression and decision tree.

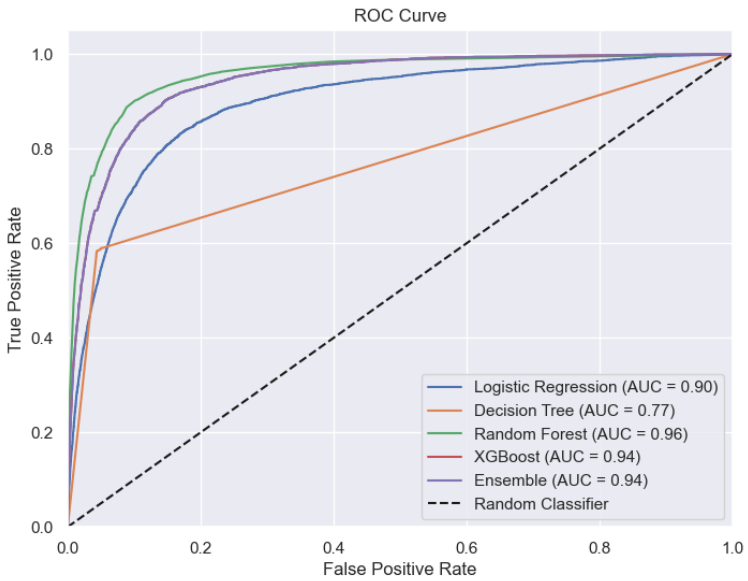


Figure 19: ROC curves for all the models

Furthermore, the models were evaluated using the AUC-ROC score, which measures the trade-off between true positive rate and false positive rate across various thresholds. The results in Figure 19 show that logistic regression achieved an AUC-ROC score of 0.90, while decision tree achieved a score of 0.77. In contrast, the random forest and XGBoost models achieved scores of 0.96 and 0.94, respectively, which were significantly higher than those of logistic regression and decision tree. The custom ensemble model also achieved an AUC-ROC score of 0.94.

Overall, the results suggest that the random forest and XGBoost models performed the best among the five models, achieving high F1 and AUC-ROC scores. This outcome is likely due to the ensemble nature of these algorithms, which involves fitting multiple decision trees and averaging the results. These models are known for their ability to reduce variance and overfitting, resulting in superior performance. Moreover, the custom ensemble model, which leverages the predictions of all four models, performed similarly to XGBoost and achieved a competitive F1 score and AUC-ROC score.

```
Feature ranking:
1. national_inv (0.194863)
2. forecast_3_month (0.123013)
3. forecast_6_month (0.110555)
4. forecast_9_month (0.109572)
5. sales_1_month (0.061791)
6. perf_6_month_avg (0.054928)
7. perf_12_month_avg (0.053152)
8. in_transit_qty (0.052963)
9. sales_3_month (0.048352)
10. sales_6_month (0.047767)
11. sales_9_month (0.046447)
12. min_bank (0.035616)
13. lead_time (0.033082)
14. ppap_risk (0.011174)
15. deck_risk (0.010938)
16. stop_auto_buy (0.005380)
17. potential_issue (0.000218)
18. rev_stop (0.000145)
19. oe_constraint (0.000044)
```

Figure 20: Feature importance

In Figure 20, the influence of features on the output of the best model, random forest, is illustrated. The features that were used to make the forecast, along with the feature 'national_inv', were found to have a greater impact on the predictions. This is reasonable since both inventory levels and forecasted values are crucial factors when attempting to predict whether a product will go into backorder in the future.

8. Future scope

Several future experiments are being planned to further understand these models. One of the primary experiments involves utilizing the original imbalanced dataset and evaluating the model performance using the balanced accuracy metric. The goal is to compare the performance of

this approach with the model that was built using an oversampling technique to handle the class imbalance. The results of this comparison will help determine how oversampling the dataset affects the model performance.

In addition, it is suggested that a regression model be fit to the predictions of the models used to build the custom ensemble. This will enable the identification of coefficients to be used as weights while building the custom ensemble model. Currently, the custom ensemble model gives equal weight to each model, including both the worst and best-performing models, which negatively affects the ensemble's performance. By assigning higher weights to the best-performing models and lower weights to the worst-performing models, the performance of the custom ensemble model can be improved.

Furthermore, it would be beneficial to consider using more samples of the data from the dataset, rather than relying on a single sample of 200,000 data points that have currently been used to build the model. This can aid in improving the model's ability to generalize and perform well on unseen data. By incorporating more samples from the dataset, the model can learn more about the underlying patterns and relationships in the data, leading to more accurate predictions.