Lissandro Jose Alvarado
ANOP’s Consulting Group
201 7th Avenue
Lewisburg, PA 17837

Dec 11, 2023

207 Holmes Hall,
Fraternity Rd,
Lewisburg, PA 17837

Dear Operations Manager Alia Stanciu,
On behalf of ANOP’s Consulting Group, we express our sincere gratitude for this opportunity to collaborate and contribute to the success of your project for Bucky Company. In the last memo, we performed an Exploratory Data Analysis (EDA) where we discovered key predictive indicators, notably, the three-month sales forecast and sales from the previous month that emerged as having the strongest correlations with backorders. We also discovered the imbalance in the dataset for backorder, meaning is not common for this to happen. The next steps will involve a deeper dive using logistic regression, K-Nearest Neighbors Algorithm (KNN), Decision Tree, and Random Forest to refine our predictions to reduce occurrences of backorders.

Given the rarity of backorder, we will also assess the models using performance metrics such as accuracy, balanced accuracy, Operating Characteristic (ROC) Area Under the Curve (AUC) scores, and specialized gain metric that involves using the predicted probabilities from our models to identify the top 100 SKUs most likely to backorder and then determining the percentage of these that went into backorder. This specific analysis will help us understand the effectiveness of our models in real-world scenarios.

The business problem presented to us is the occurrence of backorder situations, when a customer places an order for a product that is not currently available in the company's inventory, potentially leading to lost sales, reduced revenue, and diminished customer loyalty. The goal of ANOP’s Consulting Group is to accurately predict the likelihood of products going on backorder through predictive analytics. By doing so, the company aims to proactively manage its inventory levels, optimize its supply chain, and maintain customer satisfaction.

2. Data Overview
The data was obtained directly from Bucky Company, comprising a comprehensive collection of 1,360,571 records across 23 columns. This dataset encapsulates 22 distinct features alongside a single target column designed to predict product backorders. The dataset presents historical data spanning several weeks leading up to the prediction week. It is worth noting that to prepare the data for analysis most of the data columns were standardized. Also, we imputed missing values in 'lead_time', ′perf_6_month_avg′, ‘perf_12_month_avg’ by replacing them with the median. The feature columns with their respective description can be found in the appendix section. To prepare the data for modeling we divided between 60% training and 40% for validation data for all models to have consistency across all models.

3. Modeling Analysis
For this analysis, we will use logistic regression, K-Nearest Neighbors Algorithm (KNN), Decision Trees, and Random Forest.

3.1 Logistic Regression

Our first step was to use logistic regression, recognizing its strong performance for binary outcomes—perfect for predicting the occurrence of backorders. In executing the model, we opted for the following parameters that were found after manually testing multiple combinations of parameters, this was the one with the best performance. an L2 penalty to curb overfitting by constraining coefficient inflation. We set C to an expansive 1e42, subtly steering the model away from excessive complexity. We chose the 'sag' solver for its efficient optimization capabilities, especially suited for our sizable dataset. To address the imbalance between backordered and non-backordered instances, we applied a 'balanced' class weight, ensuring that our model remains fair and considerate of both outcomes. With this model we discovered that at the baseline level, the log odds of a product being on backorder is 0.23, being the intercept of the model.

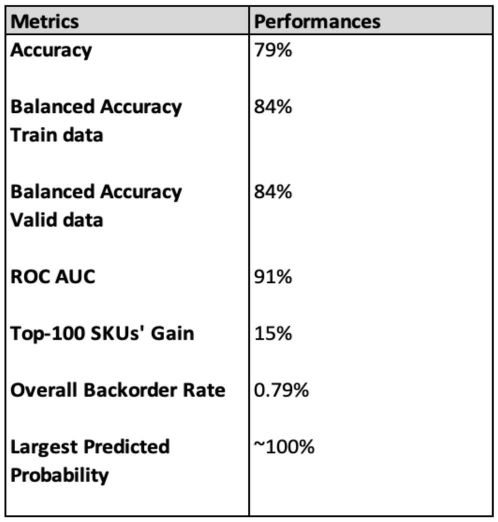
Predictor	Coefficient
national_inv	-4.98
lead_time	-1.05
in_transit_qty	-8.66
forecast_3_month	2.26
forecast_6_month	1.76
forecast_9_month	-0.55
sales_1_month	6.38
sales_3_month	4.17
sales_6_month	0.13
sales_9_month	-0.25
min_bank	0.00
pieces_past_due	1.66
perf_6_month_avg	11.75
perf_12_month_avg	-1.51
local_bo_qty	5.95
potential_issue_Yes	1.07
deck_risk_Yes	-0.06
oe_constraint_Yes	0.62
ppap_risk_Yes	0.26
stop_auto_buy_Yes	-0.47
rev_stop_Yes	-6.08

Table 1. Coefficient of logistic regression variables

The logistic regression model highlights that national inventory (coefficient: -4.98) greatly reduces backorder probability, while recent sales (coefficient: 6.38) significantly increase it. This contrast points to the critical balance between maintaining inventory levels to match sales activity to prevent backorders.

Logistic Regression: Model Performance

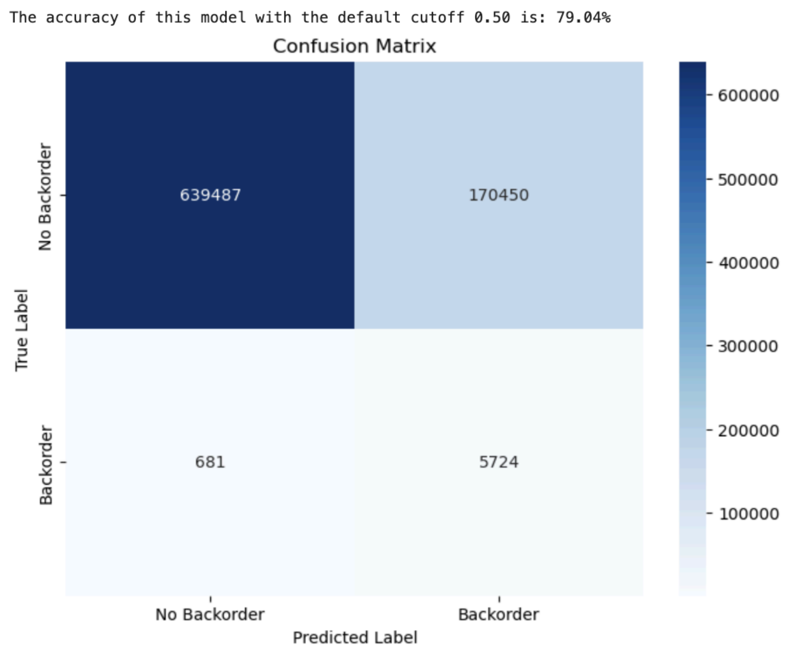
To fully understand the performance of the Logistic Regression we performed a performance analysis found in Table 2. Among the metrics, we have accuracy which gives us a baseline of correct predictions, but balanced accuracy is more informative as it accounts for the imbalance by treating both classes equally. The Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) is crucial for measuring the model's ability to discriminate between classes across all thresholds. The Top-100 SKUs' Gain is a metric reflecting the model's ability in identifying the most critical items at risk of a backorder. Together, these metrics provide a multi-faceted view of the model's effectiveness and reliability in a real-world operational setting.



Metrics	Performances
Accuracy	79%
Balanced Accuracy Train data	84%
Balanced Accuracy Valid data	84%
ROC AUC	91%
Top-100 SKUs' Gain	15%
Overall Backorder Rate	0.79%
Largest Predicted Probability	~100%

Table 2. Performance Metrics for Logistic Regression

The logistic regression model demonstrates a robust performance with an accuracy of 79%, indicating a fair degree of correctness in overall predictions. However, its balanced accuracy of 84% is particularly noteworthy in the context of our imbalanced dataset, reflecting a more reliable measure of effectiveness across both backorder and non-backorder classes. The ROC AUC score of 91% underscores the model's strong discriminative ability. Impressively, the model's Top-100 SKUs' Gain stands at 15%, significantly surpassing the dataset's overall backorder rate of 0.79%, highlighting the model's precision in flagging SKUs most at risk of back-ordering. The largest predicted probability nearing 100% signals high confidence in the model's top predictions. However, even with these strong metrics, these metrics don't tell the full story as seen in the confusion Matrix from Figure 1.



```
The accuracy of this model with the default cutoff 0.50 is: 79.04%
```

Figure 1. Confusion Matrix for Logistic Regression

Through the confusion matrix in Figure 1, we can see the model's tendency to over-predict backorders, which may lead to inefficiencies such as overstocking, increased holding costs, and potential wastage.

After exploring the Logistic Regression model's robust predictive abilities, we now transition to analyzing the KNN approach. KNN's instance-based learning could provide a contrasting perspective to the equation-driven nature of Logistic Regression, offering a valuable comparison of methodologies.

3.2 K-Nearest Neighbors
In refining our K-Nearest Neighbors model through grid search, we set `n_neighbors` to 23 to smooth predictions and reduce noise sensitivity. Additionally, we implemented `weights' to 'distance'` to ensure closer neighbors have more influence in the voting process. This approach not only enhances the model's generalization but also addresses class imbalance by giving more weight to minority class backorder.

K-Nearest Neighbors: Model Performance
We proceed to use our performance metrics in our KNN model.

Metrics	Performances
Accuracy	~100%
Balanced Accuracy Train data	~100%
Balanced Accuracy Valid data	56%
ROC AUC	88%
Top-100 SKUs' Gain	98%
Overall Backorder Rate	0.79%
Largest Predicted Probability	100%

Table 3. Performance Metrics for KNN

The KNN model's performance metrics exhibit near-perfect accuracy and balanced accuracy on the training data, suggesting an excellent fit to the training set. However, the sharp drop in balanced accuracy to 56% on the validation data indicates overfitting, as the model doesn't generalize well to unseen data. Despite this, the model achieves a respectable ROC AUC of 88%. A Top-100 SKUs' Gain of 98% is particularly impressive, significantly higher than the overall backorder rate of 0.79%, indicating the model's strength in identifying the most likely cases of backorders. The largest predicted probability at 100% suggests the model is very confident in its predictions, although the potential for overconfidence needs to be considered, given the overfitting indicated by the validation data accuracy, so this model is not the best in predicting new data.

Having analyzed the KNN model, we now pivot our attention to decision trees. The decision tree will offer us a structured, rule-based approach, which may be a better generalization to new data compared to KNN, especially given the overfitting we've observed.

3.3 Decision Tree

We conducted a comprehensive search for the best hyperparameters using grid search. We adjusted the class weight to 'balanced', acknowledging the rarity of backorders and correcting for the skew in the dataset. The tree's depth was capped at 11 levels to prevent overfitting, maintaining a balance between model complexity and generalization. Furthermore, we set the minimum impurity decrease and a minimum sample split of 1000, fine-tuning the model's sensitivity to the data split criteria, all to ensure that our decision tree delineates the nuances of backorder likelihood with precision and reliability. The full Decision Tree can be found in the appendix code. Below you can find a feature importance plot that helps us understand the most influential features in this model.

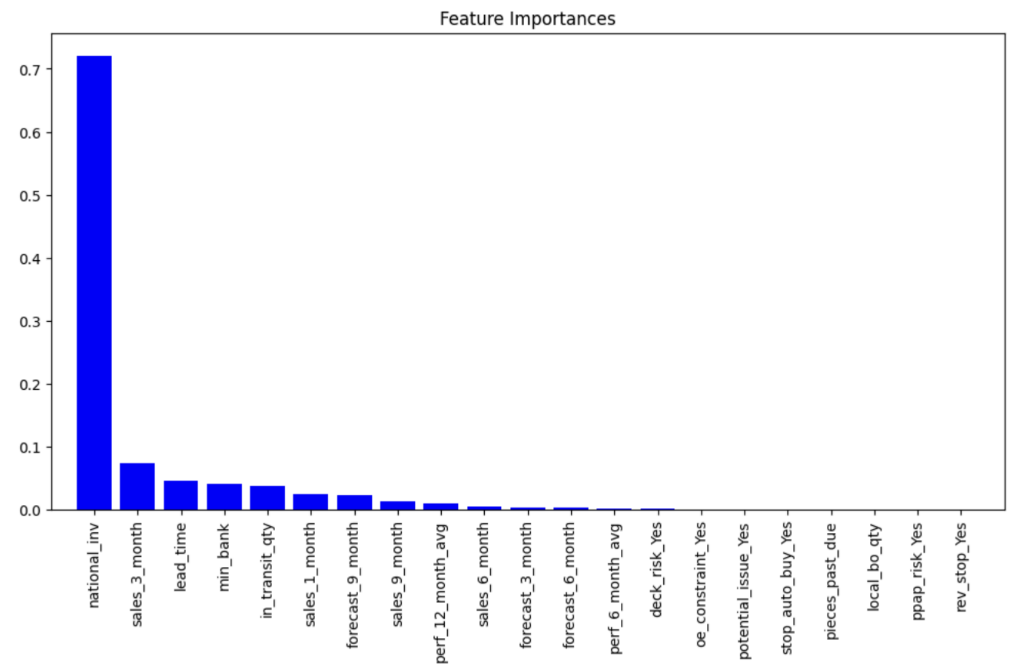


Figure 2. Feature Importance Plot for Decision Tree

The feature importance plot emphasizes national_inv as the most critical predictor in determining the target outcome, reflecting its prominent role in the initial splits of the decision tree, where it heavily influences the model's path. The secondary features like sales_3_month, lead_time, min_bank, and in_transit_qty carry less weight but are integral to refining the model's decisions, often affecting class separation in the tree's deeper, more detailed branches. In the decision tree, the root node's split on `lead_time` (≤ 0.949) with a Gini impurity of 0.276, affects a substantial part of the dataset, covering 135,666 samples, hinting at its importance for class separation. Conversely, the left child node split on `national_inv` (≤ 0.119) has a maximum Gini impurity of 0.5, indicating an even class distribution among its 81,6342 samples, suggesting it's not as discriminative. Deeper nodes like `perf_12_month_avg` (≤ 0.058) and `perf_6_month_avg` (≤ 0.2) show varying Gini impurities (0.453 and 0.069, respectively) and fewer samples, implying more specific rules but different levels of class purity. Moreover, the `sales_3_month` (≤ 0.229) node guides a significant split with a Gini of 0.296 across 680,676 samples, mainly predicting non-backorders. Further down, `forecast_9_month` (≤ 0.051) with a Gini of 0.218 across 636,955 samples, and a deeper node on `national_inv` (≤ 0.346) with a Gini of 0.235 and 102,274 samples, both indicate meaningful but complex class separations.

Decision Tree: Model Performance
We proceed to use our performance metrics in our decision tree model.

Metrics	Performances
Accuracy	86%
Balanced Accuracy Train data	88%
Balanced Accuracy Valid data	87%
ROC AUC	93%
Top-100 SKUs' Gain	21%
Overall Backorder Rate	0.79%
Largest Predicted Probability	96%

Table 4. Performance Metrics for Decision Tree

An accuracy of 86% along with a balanced accuracy of 88% for training data and 87% for validation data suggests the model is robust, providing reliable predictions for both backorder and no-backorder instances despite the unbalance in the dataset. The ROC AUC score of 93% is noteworthy, indicating a high level of discriminative ability, as this metric summarizes the model's performance across all classification thresholds. The Top-100 SKUs' Gain at 21% demonstrates the model's practical value, significantly outperforming the overall backorder rate of 0.79% by identifying the SKUs most likely to go on backorder. Also, the largest predicted probability at 96% reflects a high confidence level in the model's predictions. These metrics together indicate a decision tree model that is both accurate and practical.

Knowing this we can confidently try one last model, Random Forest. Random forest tends to handle unbalanced data better and make more robust and accurate predictions than Decision Tree could.

3.4 Random Forest
We optimized the Random Forest model's parameters to use 'balanced_subsample' for class weight to address the data's imbalance. A maximum depth of 20 and a minimum sample split of 15 ensure the model is complex enough to capture relevant features without overfitting. With 120 estimators, the model gains robustness and accuracy, making it a solid tool for predicting backorders efficiently.

Random Forest: Model Performance
We proceed to use our performance metrics in our decision tree model.

Metrics	Performances
Accuracy	97%
Balanced Accuracy Train data	98%
Balanced Accuracy Valid data	83%
ROC AUC	96%
Top-100 SKUs' Gain	85%
Overall Backorder Rate	0.79%
Largest Predicted Probability	99%

Table 5. Performance Metrics for Random Forest

The performance metrics for the Random Forest model suggest exceptional predictive power. An accuracy of 97% indicates that the model correctly predicts backorders and non-backorders in most cases. The balanced accuracy of 98% for training data and 83% for validation data reflects the model's effectiveness across both classes, which is particularly notable given the likely class imbalance. The ROC AUC score of 96% demonstrates the model's strong discriminative ability between the backorder and no-backorder conditions. Impressively, the Top-100 SKUs' Gain of 85% reveals that the model is highly adept at identifying the products most at risk of a backorder, far surpassing the overall backorder rate of 0.79%. Furthermore, the largest predicted probability score of 99% indicates that the model predicts backorders with a high degree of confidence, reinforcing its reliability in practical applications.

Having covered Logistic Regression, KNN, Decision Tree, and Random Forest, we have enough information to confidently recommend the best model for predicting backorders.

4. Best Model

Based on our thorough analysis of all models presented we can confidently conclude that the Random Forest model is the best option for backorder prediction due to its high accuracy rate of 97%, indicating reliable predictions in most instances. It achieves a balanced accuracy of 98% for training data and 83% for validation, showcasing its robustness against class imbalance. The model's ROC AUC of 96% signifies its capability to distinguish between classes. An impressive Top-100 SKUs' Gain of 85% far exceeds the general backorder rate. Finally, the highest score for the complete dataset in Kaggle ended up being 9.22 which is a stronger predictor model compared to other previous attempts with other models.

Even though there are models that may surpass Random Forest in certain aspects such as Top-100 SKUs' Gain or accuracy, they each have limitations that make them less suited for predicting backorders with this data set. Logistic Regression, despite its simplicity and high accuracy, is not very practical for real-life use due to its low accuracy. KNN can struggle with large datasets and is sensitive to irrelevant features, leading to overfitting as observed previously when testing with validation data. Finally, Decision Tree was a strong model for this problem, due to the unbalanced nature of the dataset Random Forest was a strong model to address these concerns.

5. Conclusion

Once again, on behalf of ANOP's Consulting Group, we thank you for trusting us with this project.
Our collaboration has demonstrated the effectiveness of data-driven decision-making through meticulous analysis and model selection. The Random Forest model's excellent performance in prediction accuracy and its nuanced understanding of backorder risks underscores our commitment to delivering actionable insights. Looking ahead, we recommend exploring other influencing factors for backorders beyond historical sales and inventory levels, such as market trends and supplier reliability, to continuously refine the predictive analytics framework established here.

We hope that you will take us into consideration for our continued partnership with Bucky Company, aiming for sustained progress and success.

6. Appendix

6.1 Feature Columns for Dataset.

- **sku:** Internal Stock Keeping Unit, Unique for each product (int as string)
- **national_inv:** Current inventory level in units for the part (int)
- **in_transit_qty:** Amount of product in transit from source (int)
- **lead_time:** Transit time for product from order to delivery in days (int)
- **Min_bank:** Minimum recommended amount to stock (int)
- **pieces_past_due:** Number of parts overdue from source (int)
- **forecast_3_month:** Forecast sales for the next three months (int)
- **forecast_6_month:** Forecast sales for the next six months (int)
- **forecast_9_month:** Forecast sales for the next nine months (int)
- **sales_1_month:** Sales quantity for the prior one month time period (int)
- **sales_3_month:** Sales quantity for the prior three month time period (int)
- **sales_6_month:** Sales quantity for the prior six month time period (int)
- **sales_9_month:** Sales quantity for the prior nine month time period (int)
- **perf_6_month_avg:** Sourcing performance for prior six month period. Proportion of orders placed that are not backordered.
- **perf_12_month_avg:** Sourcing performance for prior twelve month period. Proportion of orders placed that are not backordered.
- **local_bo_qty:** Amount of stock orders overdue (int)
- **potential_issue:** Sourcing issue for part identified (yes/no)
- **deck_risk:** Internally used part risk flag (yes/no)
- **oe_constraint:** Internally used part risk flag (yes/no)
- **ppap_risk:** Internally used part risk flag (yes/no)
- **stop_auto_buy**: Internally used part risk flag (yes/no)
- **rev_stop:** Internally used part risk flag (yes/no)
- **went_on_backorder:** Product actually went on backorder, target value (yes/no)

6.2 Jupyter Notebook Analysis

For this report, the Jupyter Notebook was used as the main tool of analysis. The analysis code and the extended version of the analysis can be found in a different document in Moodle. Also the Decision Tree and Random Forest Model can be found in the code under their respective category of analysis in the Jupyter Notebook.