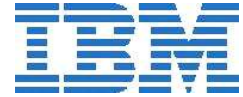


IBM U.S. Federal



26 February 2021

**Scientific and Technical Report
Backorder Prediction**

For

**DEFENSE LOGISTICS AGENCY
RESEARCH AND DEVELOPMENT**

CONTRACT NUMBER SP4701-20-C-0052

Submitted to: Defense Logistics Agency

Mr. Manuel Vengua
Contracting Officer's Representative
Manuel.M.Vengua@dla.mil

Submitted by: IBM Corporation

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188							
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.											
1. REPORT DATE (DD-MM-YYYY) 26022021		2. REPORT TYPE Scientific and Technical Report		3. DATES COVERED (From - To) Jul 20 - Feb 21							
4. TITLE AND SUBTITLE Scientific and Technical Report on Backorder Prediction				5a. CONTRACT NUMBER SP4701-20-C-0052							
				5b. GRANT NUMBER							
				5c. PROGRAM ELEMENT NUMBER							
				5d. PROJECT NUMBER							
6. AUTHOR(S) Tatko, Ryan W. White, Christopher R. Williams, Geoffrey M. Myers, Robin K.				5e. TASK NUMBER							
				5f. WORK UNIT NUMBER							
				8. PERFORMING ORGANIZATION REPORT NUMBER							
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) International Business Machines (IBM) GBS Federal Division 11107 Sunset Hills Rd Reston, VA 20190				10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)							
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Logistics Agency (DLA) Information Operations / Research and Development 725 John J Kingman Rd Ft. Belvoir, VA 22060											
12. DISTRIBUTION/AVAILABILITY STATEMENT											
13. SUPPLEMENTARY NOTES											
14. ABSTRACT DLA awarded IBM a contract to research the development of using machine learning algorithms to develop models to predict the likelihood of item backorders and to research the factors contributing to backorders. IBM researched two approaches, experimenting with classical machine learning models and with neural networks. IBM engineered numerous features to identify potential contributing factors. IBM demonstrated that machine learning models, using a cleaned, limited, and curated data set, can 1) provide predictions of the presence of a backorder (through the forward looking 12 months), 2) provide backorder counts that test well against actual inventory and backorder values, and 3) identify highly correlated causative features. This research provides DLA with the opportunity to further develop machine learning models, to integrate and scale additional capabilities / data, to better understand the factors contributing to backorders and allows DLA the opportunity to act to preclude predicted backorders, increasing product availability to its customers.											
15. SUBJECT TERMS											
16. SECURITY CLASSIFICATION OF: <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%; padding: 2px;">a. REPORT</td> <td style="width: 33%; padding: 2px;">b. ABSTRACT</td> <td style="width: 33%; padding: 2px;">c. THIS PAGE</td> </tr> <tr> <td style="text-align: center; padding: 2px;">U</td> <td style="text-align: center; padding: 2px;">U</td> <td style="text-align: center; padding: 2px;">U</td> </tr> </table>			a. REPORT	b. ABSTRACT	c. THIS PAGE	U	U	U	17. LIMITATION OF ABSTRACT UU		18. NUMBER OF PAGES 52
a. REPORT	b. ABSTRACT	c. THIS PAGE									
U	U	U									
			19a. NAME OF RESPONSIBLE PERSON Manuel M. Vengua								
			19b. TELEPHONE NUMBER (Include area code) 571-767-1390								

Abstract

The Defense Logistics Agency (DLA) awarded IBM a contract to research the development of predictive models using machine learning algorithms to predict item backorders and to research the factors contributing to the potential backorder. IBM researched two supervised machine learning approaches:

1. Classical learning models – widely used and easily interpretable algorithms such as decision trees and linear regression techniques that learn to map some input to an associated output through statistical methods.
2. Artificial Neural Network models – computational systems loosely based on biological neural networks that allow for more complex non-linear relationships between the response variable and its predictors.

IBM demonstrated that machine learning models—using a cleaned, limited, and curated dataset¹—can forecast monthly backordered versus in-stock items, predict the total number of backorders at the item level, and identify important correlated features for each month in the forecast, up to 12 forward looking months. This research provides an entry-point to enhance these initial machine learning models through integrating additional data sources, scaling to larger segments of data, and gaining deeper insight into contributing backorder factors. This research highlights the foresight necessary for DLA to act on anticipated backorders, increasing product availability to its customers.

¹ AAC D or Z NIINs with a complete 60 months of historical data, at least one backorder occurrence at any point in its history, and an annual demand frequency of at least one.

Table of Contents

Section	Page
Abstract.....	1
Table of Contents	2
List of Figures.....	3
List of Tables	3
Summary.....	4
Introduction.....	6
Scope	6
Methods, Assumptions, and Procedures	7
Modeling	20
Conclusions.....	35
Recommendations for Future Enhancements	36
Appendices.....	41
Appendix A. Comprehensive List of Model Features	41
Appendix B. Model Hyperparameters	43
Appendix C. Feature Importance Plots.....	45
Appendix D. Feature Importance Plots for Backorder Scenarios.....	50

List of Figures

Figure 1: Accuracy for Top Performing Backorder Prediction Models.....	5
Figure 2: CRISP-DM Process.....	8
Figure 3: Data Preprocessing Pipeline Overview	12
Figure 4: FSC One-Hot Encoding Example	14
Figure 5: Log Transformation Example.....	14
Figure 6: Feature Selection Pipeline	16
Figure 7: Example of a Decision Tree	18
Figure 8: Encoder-Decoder Architecture.....	19
Figure 9: Cross Validated Sliding Window Paradigm.....	20
Figure 10: Regression Metric Model Comparison.....	21
Figure 11: Classification (Backorder Class) Model Comparison	22
Figure 12: Classification (In-Stock Class) Model Comparison	23
Figure 13: Backorder Simulation.....	24
Figure 14: Sample Model Forecasts.....	24
Figure 15: Averaged 3-Fold Cross-Validated Paired T-Test Scores for Regression Metrics	27
Figure 16: Averaged 3-Fold Cross-Validated Paired T-Test Scores for Classification Metrics.....	28
Figure 17: Decision Tree Feature Importance	29
Figure 18: Gradient Boosted Trees Feature Importance	30
Figure 19: SHAP Force Plot	31
Figure 20: Neural Network Feature Importance	32
Figure 21: SHAP Dependence Plots	33
Figure 22: Historical and Future Backorder Count Sequence Pattern Pairing.....	34
Figure 23: Neural Network Feature Importance for Backorder Count Sequence Pairs.....	34
Figure 24: Representative Sample Decision Tree vs. Oversampled Decision Tree Models.....	37
Figure 25: NIIN Backorder History Grouping Pipeline.....	38
Figure 26: Undersampling and Oversampling Technique	38
Figure 27: Effect of High Value Signaling Features in Model Results	40

List of Tables

Table 1: Additional Software Installed on GFEs	10
Table 2: Data Provided by DLA	11
Table 3: Modeling Dataset Exclusion Criteria.....	13
Table 4: Feature Engineering Examples	15

Summary

The Defense Logistics Agency (DLA) continues to wrestle with the question of inventory management and how best to ensure appropriate stock is on-hand and available to its myriad customers. In order to ensure warfighter readiness, DLA tries to understand factors affecting material availability (MA). The process of maximizing material availability is directly dependent upon minimizing backorders. To help address this challenge, DLA awarded IBM a contract to research the development of a set of predictive models using machine learning (ML) algorithms to forecast product backorders and to research the factors driving potential backorders.

DLA provided the IBM team with datasets focused primarily on stock items' inventory features over a 60-month window. In order to target the issues that drive backorders in DLA's supply chain, IBM focused the models on a curated dataset representing the most at-risk population of items in DLA's inventory (DoD-stocked items with an existing history of backorders and an annual demand). Through this research, IBM investigated two different modeling approaches, experimenting with both classical machine learning (e.g., decision trees) and deep learning (e.g., artificial neural networks) frameworks, developing six different model architectures in total.

Using industry standard relational database management systems (PostgreSQL) and Python machine learning libraries (Scikit-Learn and TensorFlow), IBM developed a robust AI/ML data pipeline that stores, cleans, and preprocesses the data, builds and trains each supervised machine learning model through a custom cross-validation loop, and evaluates those models against a series of test datasets to measure general accuracy and precision.

As shown in Figure 1, the top performing models² developed in this research predict future in-stock items with a quarterly forecast accuracy between 95-93%, and future backordered items with a quarterly forecast accuracy between 71-25%. While accurate backorder forecasting becomes more challenging as the time horizon moves into the future, IBM believes this research has identified key insights into understanding future predictive backorder drivers.

² The top performing models as determined by this research are: the gradient boosting algorithm, the multilayer perceptron, and encoder-decoder neural networks.

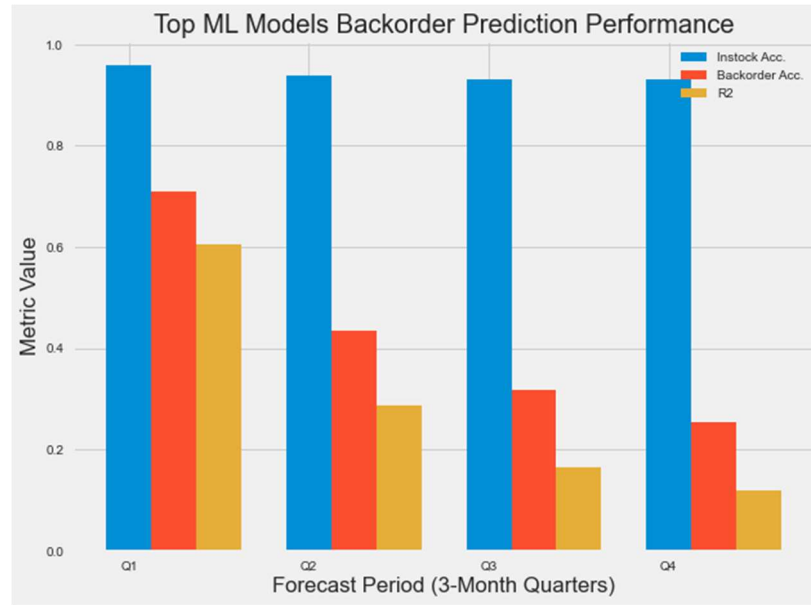


Figure 1: Accuracy for Top Performing Backorder Prediction Models

Throughout this research, IBM gained insight from DLA Subject Matter Experts (SMEs) to engineer and extract features from the raw data through a combination of domain knowledge and data mining techniques. Several of the engineered features demonstrated high predictive ability in identifying backorders, providing insight and explainability into the “why” – what factors drove the backorder. These insights help provide DLA the information necessary to act to correct or adjust inventory levels, and thus prevent a backorder situation.

As part of the technical proposal for this research, IBM proposed to DLA the comparison of IBM’s backorder prediction results with DLA’s backorder predictions. The IBM team learned during technical review meetings at the beginning of this research that DLA did not have backorder prediction results to share with IBM for the comparison. Thus, the IBM team was not able to perform this comparison; however, if baseline backorder prediction results could be shared, IBM would gladly perform this comparison.

The IBM research provides an entry-point to enhance these initial machine learning models through integrating additional data sources, scaling to larger segments of data, and gaining deeper insight into fundamental backorder factors. This research highlights the foresight necessary for DLA to act on anticipated backorders, increasing product availability to its customers.

Introduction

While DLA can currently view material availability and backorders covering the recent past, it lacks the ability to predict, both short and long-term potential backorders and inventory shortfalls, hampering its ability to ensure the right items are in stock at the right time for its customers. Such logistics challenges can pose significant risks to our military, its material readiness, and its current and future operations. Specifically, DLA faces challenges in the areas of demand forecasting and inventory optimization. Maintaining adequate supply levels in order to efficiently support the requirements of the armed services, especially when demand can be greatly variable, is an immense challenge. Such delays result in reduced responsiveness to the various services across the Department of Defense, increased carrying and storage costs when DLA preventatively stocks up on such items, and higher risk of being left with useless or obsolete parts.

Given the current state of expectations from both DLA leaders and customers on data access, data integration, predictive analytics, and real-time decision support tools, DLA desires to develop a backorder prediction capability as part of a larger analytics and decision support toolset. A new capability may be able to better serve both DLA's internal business and decision-making processes (to anticipate and address potential backorders) and to perhaps to provide external customers visibility into those processes.

Scope

In July 2020, DLA awarded IBM a contract under Emergent II Research and Development Broad Agency Announcement (BAA) 0001-17 to research the development of a set of models for backorder prediction using machine learning algorithms and to research the factors driving the potential backorder. Specifically, as described in IBM's accepted technical proposal,

“The IBM team will develop a robust set of predictive models using machine learning algorithms to predict the likelihood of a product backorder. This ensemble of predictive models will be underpinned by a prioritized list of model attributes that elucidate why the algorithms predicted a backorder, what the main drivers of the prediction are, and how proactive changes may impact the likelihood of a product ultimately being backordered.”

DLA agreed to provide the data to support this research. Data identification, access, capture, cleaning, normalizing, and storing are an oft-overlooked necessity to achieve valid, defensible machine learning models and analytics. Since it was likely that there would be a challenge in obtaining large, high quality datasets, with enough data upon which to develop AI/ML algorithms, IBM proposed cleaning the data (e.g., imputation of null values, data normalization, and other transformations to prepare the data for specific modeling approaches) and sampling the data for modeling. The sampling approach provided a consistent dataset over

time and accounted for imbalances in the minority (backordered) and majority (in-stock) classes within the data.

Further, IBM’s technical proposal stated that “IBM will use the current model developed by DLA as the baseline for performance. Each model developed during this research phase will be compared to the DLA model. . . .” However, during the early technical review meetings with the IBM team, DLA revealed that it would not be able to provide a dataset of backorder predictions, covering the same data provided to IBM, with which the IBM team could make these comparisons. While the IBM team would be able to do this work, this element of the research effort was not undertaken by IBM.

Methods, Assumptions, and Procedures

IBM approached this research and development challenge through the prism of stock inventory over time. As the IBM team received a collection of datasets detailing DLA’s inventory and supply data (available stock, backorders, reorder points, purchase requisitions, etc.) and not item-level demand requirements data (i.e., parts requisitions from the military Services), an item’s inventory values served as the entry point and independent variables for any modeling effort. Parallel to this inventory data, the flow characteristics of the inventory data are important as well. The flow data includes how many and when items are added to stock, when and for whom items are removed from stock, and any modifications to inventory availability (for expirations, or non-accessibility, for example).

Also, of particular interest is the metadata about the stock items in inventory as catalogued by DLA and the Services. Metadata, including supplier source, supply class, national stock number, applicable expiration dates, etc. provide the data necessary for more detailed analytical pivots.

IBM used these datasets to develop a robust set of predictive models using machine learning algorithms to forecast, at the item level, the total number of backorders against that item in each month up to 12 future months. The modeling approaches chosen for this research included classical machine learning algorithms as well as a more sophisticated model architecture using deep artificial neural networks. Through the evaluation of these models and their contributing factors, IBM identified the inherent strengths and weaknesses between approaches as well as applying machine learning to backorder prediction as a whole. Using SHAP, IBM provided a methodology to visualize causal features and gain local interpretability into the machine learning models.

The model forecasts and identification of the main drivers of backorders will allow DLA leadership to prioritize the items most at risk so that the appropriate resources can be allocated to sourcing a new supplier or stocking up in the short term. Similarly, should the models’ results identify a trade-off between predictive accuracy and the reasons for the prediction (a common occurrence in machine learning projects), then DLA leaders will be able to prioritize the relative

importance of these two factors.

Methodology

Successful model development and execution, especially with respect to data analysis and analytics, require strict adherence to best practices and processes and the use of experienced project teams who know how to avoid common pitfalls. To guide our work, IBM employed the Cross Industry Standard Process for Data Mining (CRISP-DM). CRISP-DM is an Agile-infused iterative methodology, the application of which results in higher quality forecasts and improved deployment of models. IBM has used CRISP-DM effectively and efficiently in engagements across a variety both public and private sector clients.

Using CRISP-DM, the IBM team focused on enrichment and refinement of the predictive analytics products (models, dataset development, etc.) through multiple small iterative sprints. This structured and deliberate approach mitigates inherent risks in model development, aids in smoother and more rapid prototyping, and allows the team to demonstrate incremental business value throughout the project. In the case of DLA, each internal sprint focused on a specific stage of the CRISP-DM process (see Figure 2 for the major steps and description of the macro-design).

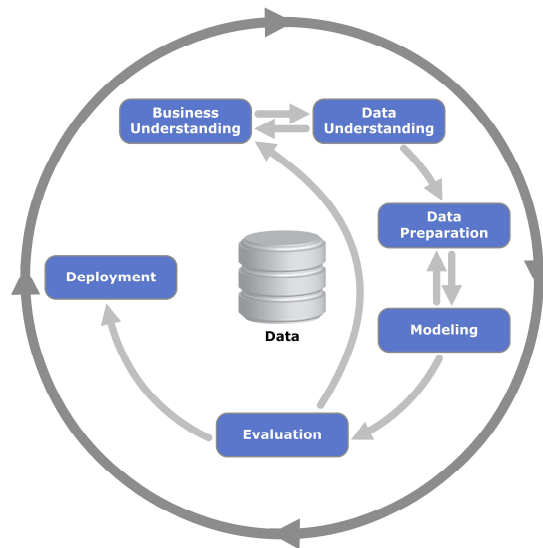


Figure 2: CRISP-DM Process

Problem Statement

As the nation’s combat logistics support agency, DLA carries out a variety of functions to support the “warfighter first” mission. One of the most common supply chain events which adversely impacts warfighter readiness is the occurrence of backorders. DLA strives to eliminate backorders by ensuring there is enough inventory available and ready for shipment when an order is placed so that right amount of an item is shipped and arrives at the right time. Being able

to effectively anticipate backorders and identify why that backorder may occur could allow DLA to take corrective actions to address and perhaps preclude the potential backorder.

Traditionally, statistical regressions techniques are the most commonly utilized method to analyze numerically based inventory values over time and to project the future. Comparison of the projected inventory values with actual stock inventory levels over time determined the accuracy of the regression algorithms against reality. These regression techniques are a widely accepted and easily interpretable modeling strategy in the time-series data domain. However, due to their inability to model complex non-linear relationships, these models may not be able to provide the most accurate future forecasts for more sophisticated problems.

Machine learning, a subdomain of artificial intelligence, represents a different approach from statistical models that focuses on developing algorithms that “learn” to solve a problem through experience and a process called “training.” More specifically, supervised machine learning models learn by training algorithms on known input and output data (inventory stock and backorders, for example), and through an iterative optimization process, learn to predict the outputs on its own.

The advantage of machine learning based predictive models is that the models generate feedback on the accuracy of their predictions—over time the model can be re-trained to incorporate the feedback in future model iterations. This feedback loop promotes continuous improvement in machine learning algorithms and accuracy of the models so that future models can build on the success of the initial generation. Over time, these models obtain feedback on the accuracy of their predictions and help mitigate the impact of performance degradation due to natural shifts in the input data.

IBM’s task with this research was to develop an initial set of models to address DLA’s near-term predictive needs. Based on the data and guidance inputs described, IBM employed predictive modeling techniques to identify critical latent relationships in the raw datasets and provide backorder prediction forecasts. IBM data scientists worked closely with DLA SMEs to investigate the data and determine the right data science protocols that are needed to identify the signals of interest. As discussed below, this effort involved preparing complex datasets, transforming the data, developing predictive models, evaluating, and cross-validating outcomes, and defining the precise order of operations for ongoing sustainment of the models. The output of the trained models (i.e., the forecasted predictions of backorders across a 12-month time horizon, evaluation and output metrics, and contributing factors) were validated against each item’s true backorder history in order to validate the models. These initial models, which leverage the inventory data currently available at DLA, serve as the output for this research effort.

Hardware and Software

Hardware. Team IBM was provided with 4 DELL Precision 15 7520 laptops each outfitted with an Intel i7-6820HQ CPU @ 2.70 GHz, 32GB of RAM, a 512GB SSD, and a NVIDIA Quadro M2200 graphical processing unit (GPU) with 4 GB of RAM. The internal Wi-

Fi adapters on each device were disabled, therefore the initial transfer of data onto the machines and sharing of code between machines was limited to file transfers via CD-ROMS that could be read by the laptops with external disk drives.

Software. Each laptop received by the team had clean installations of Windows 10 without additional software. Team IBM installed several software packages to configure the GFE environment to allow the team to perform data analysis, transformation, and modeling. Table 1 displays a comprehensive list of the software packages that were installed on the laptops as well as the justification for installation. The laptops did not have access to the internet; all software was installed using offline methods by transferring the installation files via CD-ROM.

GFE Software Installation	
Software	Description & Reason for Installation
PostgreSQL 12.4	Used for storage of the data extracts in order to query the data via filtering, merging, and aggregating
Anaconda (Individual Edition) for Python 3.7	Used for developing the models and maintaining/creating the structured databases
Psycopg 2.8.5	Python package that allows Python to connect with a PostgreSQL database
Notepad ++, Sublime Text Editor 3, Atom	text editors used for writing code and organizing software projects
Tslearn 0.4.1	Python package for machine learning on time series data
SHAP 0.37.0	Python package used for model output interpretability
Quadro-desktop-notebook-win10-64bit Driver	Tensorflow GPU dependency: driver for the NVIDIA Quadro 2200
Visual Studio 2019	Tensorflow GPU dependency: C++ development tools
CUDA 10.1	Tensorflow GPU dependency: CUDA
cuDNN 10.1	Tensorflow GPU dependency: cuDNN
Tensorflow-gpu 2.3.1	Python package for building neural network models using the laptop's GPU

Table 1: Additional Software Installed on GFEs

Data Storage and Management

Raw structured supply chain data in the format of comma-separated value (CSV) files was provided to Team IBM from DLA via several data drops, which consisted of distribution of the data among team members via CD-ROM. The delivered data on the disks was at the historical National Item Identification Numbers (NIIN) level and included the following supply chain areas: replenishment (purchase orders / purchase requisitions), historical inventory levels, item characteristics, warfighter orders, and historical backorder quantities for a data snapshot of five years between August 2015 to July 2020. Table 2 below displays the filenames and descriptions of the data that were used for this study.

Data Provided to IBM	
File	Data Description
IBM_ASSETS.txt	Snapshot of on hand inventory levels
IBM_BACKORDERS.txt	Snapshot of item backorder history and used to derive target variables of backorder count for a given month and whether an item went on backorder for a given month
IBM_ITEMS.txt	Reference list of all NIINs
IBM_LEVELS.txt	Item inventory levels
IBM_MMHIST.txt	Snapshot of historical item characteristics
IBM_ORDERS.txt	Historical item orders placed by warfighters
IBM_PRPO.txt	Historical item replenishment data consisting of purchase requisitions and purchase orders

Table 2: Data Provided by DLA

Due to the amount of raw data comprising each file—too large to fit into memory for simple queries—the team opted to ingest the raw text files into a local database using PostgreSQL. PostgreSQL is an open-source relational database management system (RMDBS) that can support hosting a variety of data types (raw text, Boolean, numeric, binary, JSON, etc.) and provide a fast, seamless connection to integration with Python via the Psycopg connector module. The PostgreSQL database was used to perform the vast majority of initial data processing and feature engineering, aggregating the data up to a NIIN-Month level snapshot table which was fed into downstream processes. The primary purpose of using the database was to create a centralized store for the data that could be easily maintained, accessed, queried, and transformed to develop a model-ready dataset as displayed in Figure 3.

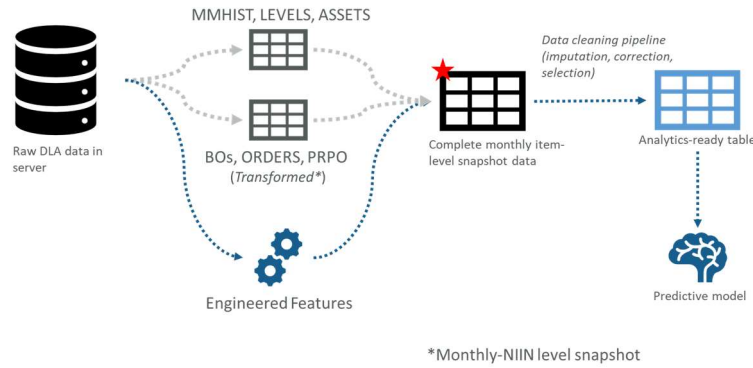


Figure 3: Data Preprocessing Pipeline Overview

Data Pre-Processing

The modeling dataset created via the aggregation of the raw tables consisted of an item-level “snapshot” table, with each row delineating that item’s corresponding monthly descriptive features (available inventory, protected stock, etc.) over a five-year history. Due to the inherent imbalance of backorders versus in-stock items (backorders are a rare event), with DLA’s support IBM decided to focus the effort on a subset NIINs that would provide enough backorder data to build models that can successfully forecast backorders for DLA’s most at-risk items.

Exclusionary criteria were determined between IBM and DLA during early technical working group meetings. The consensus of the technical working group was to focus on a subset of items that possessed a history of a backorders and annual demand. The complete list of criteria that were applied to the modeling dataset are provided in Table 3. After the application of these exclusions, the modeling dataset consisting of 33,318 NIINs. The resulting NIIN population consisting of items represent the most actionable set of items in the dataset on which to build predictive models. While the models in this study are trained on a reduced population of items, the modeling pipeline developed in this research can be used to train future models on any population of DLA data.

Exclusion Criteria	
Exclusion Rule	Reason for Exclusion
Each NIIN must possess 60 months of history.	The data was reduced to NIINs that possessed complete historical data for the entire 5 years (60 months) of provided data in order to ensure there would be a complete history for each NIIN for testing (cross-validated folds).
Each NIIN’s history must contain at least 1 occurrence of a backorder across the 60 months.	The data was limited to NIINs that have historical occurrences of backorders in order to focus the research effort on backorders and expose the models to NIINs which were more at risk of going on backorder. Most NIINs in the training data have 12-month periods where there are no backorders, i.e., target values of 0 for the entire forecast period.

Each NIIN must possess an Acquisition Advice Code (AAC) of D or Z.	Each NIIN must be DoD managed, stocked, and issued or an insurance stockage objective item as DLA may be able to take preventative measures to preclude backorder(s).
Each NIIN must possess an annual demand frequency of at least 1.	Each item must experience some level of demand across the 60 months, since orders are a key contributor in the occurrence of backorders and therefore the models must learn to detect the rare occurrence of backorders based on order history.

Table 3: Modeling Dataset Exclusion Criteria

Data Cleansing

A crucial step in data preparation for predictive models is the process of resolving corrupted, incomplete, and/or missing data to ensure high quality data is used for model training and evaluation. To standardize the data for modeling, any NIIN that lacked a complete 60-month history was not included in the modeling dataset per the exclusionary criteria. In the cases where a NIIN possessed 60 months of history, but some of the features possessed missing values, data imputation was applied by replacing null values with numeric zeroes. Although the IBM team experimented with handling potential outliers using common techniques, such as Winsorizing³ or clipping, the team decided to keep the representation of the original data intact and deal with any outlier issues through feature transformation and scaling.

Feature Transformation

Commonly in AI/ML, it is necessary to produce further transformations on raw data to get data in the correct format for machine learning algorithms. For example, text and dates are not easily interpretable by machine learning systems and must be converted or transformed into a new format to be useful. By transforming the data, the IBM team made it more informative so that the model can extract the maximum amount of information necessary. In order to transform the categorical, numerical, and temporal data into a format that machine learning models can interpret, several techniques were applied to the data including: one-hot encoding, log transformations, cyclical encoding, and scaling.

Several of the features within the modeling dataset consisted of categorical values, which consist of string values that must be converted into a numerical format. One common technique for converting strings to a numerical representation is One-Hot encoding, which converts categorical values into a binary representation. Each value within the feature receives a unique identifier, which allows the machine learning algorithm to interpret the relationships between codes and the other features. An example of how a categorical feature would be converted into a binary format for ML model algorithms to use as displayed in Figure 4.

³ “Winsorizing” or “winsorization” is the transformation of statistics by limiting extreme values in the statistical data to reduce the effect of possibly spurious outliers.

FSC Code		FSC_53	FSC_13	FSC_29
53	→	1	0	0
13		0	1	0
29		0	0	1

Figure 4: FSC One-Hot Encoding Example

After conducting an exploratory analysis of the data, the IBM team noted that nearly all of the continuous features (available inventory, order numbers, etc.) had severely skewed distributions—most elements are centered around a small value, but some values within an element are very large. It is often a best practice to address this issue and make the data more separable by *log transforming* those features. The log transformation takes the mathematical input value of x and replaces it with the value of $\log(x)$. This transformation is useful on continuous features with severely right-skewed data since it can de-emphasize outliers and normalize the distribution of features with large ranges (total order values, for example) as displayed in Figure 5.

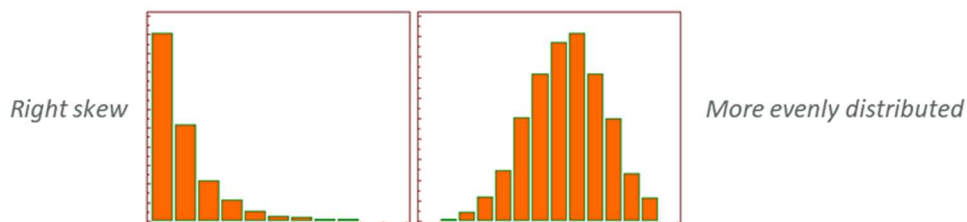


Figure 5: Log Transformation Example

Temporal features also prove to be challenging for machine learning algorithms to interpret, therefore can be encoded to better represent seasonality within the data. Sine/Cosine encoding is useful to preserve the nature of cyclical variables by encoding the value as separate sine/cosine waves, for example, the month of the year (December the 12th month, is close to January, the 1st month).

Since each feature can possess a numerical range that can vary across several orders of magnitude, it is crucial to standardize all the features. This can be accomplished by transforming the data by applying scaling, so that the model can learn from all of the features rather than focusing on the features with values that are several orders of magnitude larger than the others. Prior to modeling, the IBM team employed a robust scaler to scale each individual feature and to translate its values into the interquartile range, which reduces the impact of outliers. The scaler was fit on each training dataset for each fold of the walk forward cross-validation window as described in the Model Validation and Testing Strategy section. The separately fit scalers were then applied to both the training and testing datasets within each respective fold and transformed the datasets into the new interquartile range.

Target Features

The target feature for the machine learning models consisted of the sum total backorder count for each month over a forecast horizon of 12 months. The total backorder count is derived from summing the “new” backorders for an item in a given month with the “aged” backorders of that item. Aged backorders are defined as those that were established in any preceding month and are not yet fulfilled. This feature defines a time series regression problem, where the IBM team attempted to predict future backorder values using currently obtainable information on an item. Regression model results can be “flattened” to view the results of the classification problem, whether the item was predicted to be backordered or not at all, which will be discussed further in the Results section. In order to prepare the data for the supervised learning models, the data was transformed into a feature dataset and a target variable dataset. The feature dataset consisted of columns for each independent feature and a target dataset which consisted of the sequences of true total backorders over the successive 12 months.

Feature Engineering

At this stage, the IBM team isolated variables of importance and developed new features through business domain knowledge with input from DLA SMEs. The key component of feature engineering is understanding and identifying the areas of knowledge of the supply chain domain that contribute to backorders. The team then developed features from the existing data to capture backorder indicators in the form of additional variables. The goal of designing new features is to provide additional information from which the model can learn from and produce more informed predictions. The features developed by the team can be divided into the following overarching classes and sub-feature sets found in Table 4.

Feature Engineering		
Type of Feature	Feature Set	Examples
Continuous	Monthly snapshots	Material availability, production lead time, quantity above reorder point, etc.
	Lags	Available stock on hand last month, 3 months ago, 12 months ago, etc.
	Differences	Difference in order volume from current month to last month, 3 months, 12 months, etc.
	Rolling averages	Average of purchase order quantity over the last month, 3-month window, 12-month window, etc.
Categorical	Historical codes	Federal supply class (first 2 digits), acquisition method code, government fiscal quarter, etc.
	Binary indicators	Under reorder point indicator, out of stock indicator, purchase requisitions over administrative lead time indicator, etc.
Cyclical	Temporal data	Month of year, etc.

Table 4: Feature Engineering Examples

Feature Selection

After generating the new features, the modeling dataset consisted of many thousands of features—however, not all these features were useful in providing information about future backorder events. Although the IBM team experimented with using prior historical features (lags, rolling averages, etc.), these had little impact on model performance at the cost of longer model training times. For this reason, these features were not used by the majority of the models (aside from the LSTM-based encoder-decoder network, which necessarily takes past features into consideration). Team IBM implemented an automated approach for reducing the number of features by using a dual-stage filtering method. The first filter the IBM team applied acts as a redundancy filter that identifies and removes highly correlated features. The second filter the IBM team applied is an irrelevance filter, which relies on a variance selector threshold to remove features with low variance. For example, an FSC class feature that is only “on” for 3% of the data would be considered to have too low variance to include in the dataset. IBM also tested a strategy using a random forest as part of the feature selection process, although ultimately decided against using it in favor of the variance filter to reduce data preprocessing time. The resulting reduced feature space contains only the features with the expected highest relevance for backorder prediction. An overview of this process is displayed in Figure 6.

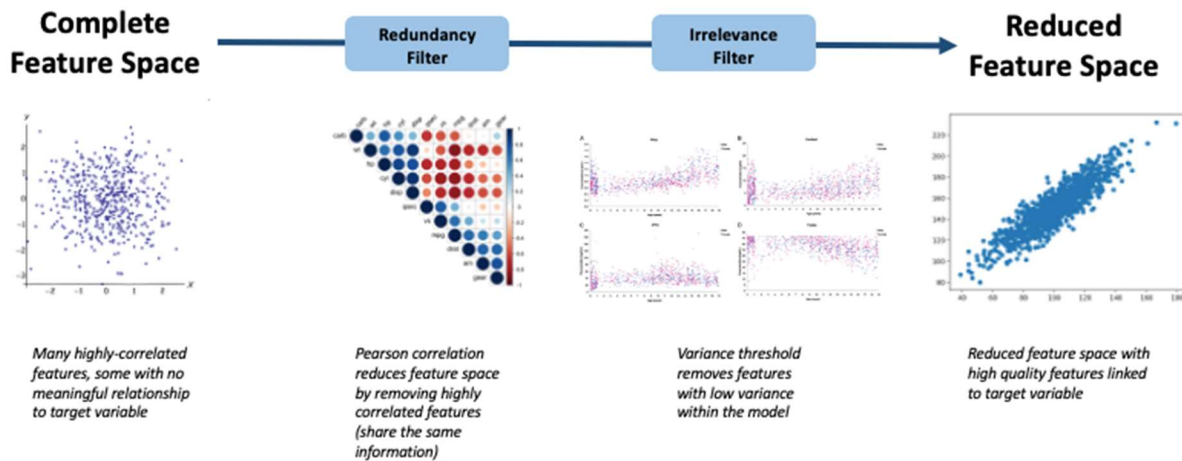


Figure 6: Feature Selection Pipeline

A complete list of the 73 features used in the modeling dataset can be found in Appendix A.

Supervised Learning Approach

Supervised learning is a subcategory of AI/ML that utilizes labeled data to train algorithms that can classify or predict some outcome. Supervised learning relies on a reinforcement learning process such that the algorithm learns to make better predictions over time, appropriately adjusting to reduce the error of the model’s loss function until minimized. The IBM team used supervised learning techniques to map the features in our dataset (available inventory, age of existing backorders, fiscal quarters, etc.) to a 12-month forecast of labeled

backorder data. This dataset, known as a target dataset, was then used to train the models to anticipate backorder events based on finding latent signals found within the feature space of the input dataset. In this section, the IBM team will discuss our modeling methodology and the types of models that were implemented in-depth.

More broadly, the IBM team used two different approaches—a classical approach, using traditional linear and tree-based machine learning techniques, and a “deep” learning approach, where the IBM team trained deep neural network structures to predict backorders. As the IBM team will discuss, either approach can be considered valid methodologies for time-series problems, with each possessing unique benefits and constraints. All six of the models were developed using common Python libraries, primarily Scikit-Learn and TensorFlow.

Models Overview

In this section the IBM team will discuss each of the six models used in the backorder prediction effort. Classical algorithms that the IBM team tested include the elastic net, decision tree, random forest, and gradient boosted tree models. Neural network architectures explored include a multilayer perceptron (MLP) and an encoder-decoder long short-term memory (LSTM) model.

Each of the models were chosen in order to provide a robust baseline of approaches to compare against. For example, if the MLP model does not outperform a simple decision tree model, one can assume a more complex model might not be worth further consideration for the backorder prediction effort. Similarly, these models were chosen to understand how fundamental linear (e.g., elastic net) and tree-based (e.g., decision tree) regression models differ or agree in their evaluation of the problem and contributing features. In order to create a more modular code pipeline, all of the Scikit-Learn models were trained using a meta-estimator that allows the models to perform 12-month forecast predictions within one function call instead of necessitating a custom training and testing loop to build separate models for each time-step in the forecast window. The team also implemented the option to use “chain” regression, where the model output of the first step is fed as an input into the second step, etc., however the team found no additional benefit to predictive power to using this option. Hyperparameters for all of the models were hand-tuned, and a list of final parameters for each model can be found in Appendix B.

Modeling Concepts

Linear Models

Linear models assume that the dependent variable and independent variables have a linear relationship. The linear model the IBM team used for this effort is the elastic net, a hybrid linear model that uses a combination of penalties from both ridge (absolute value) and lasso (squared) regression.

Tree-Based Models

Tree-based models are among the most commonly used and well-rounded supervised algorithms in machine learning, blending high performance with an easily interpretable structure. Unlike linear models, tree-based models are able to map non-linear relationships between variables. The IBM team used three different tree-based models in our experiments—a decision tree, random forest, and gradient boosted tree model. Random forest and gradient boosting algorithms are tree *ensemble* methods, which generally provide more precise, albeit less easily interpretable predictions. An example of a decision tree model for backorder prediction is given in Figure 7.

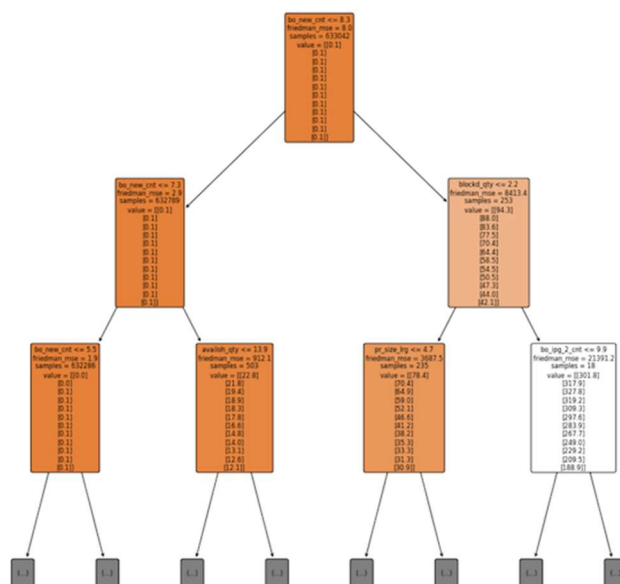


Figure 7: Example of a Decision Tree

Neural Network Models

Artificial neural networks (ANNs) are powerful machine learning structures capable of modeling highly complex relationships between input and output data. Neural networks are composed of a series of “neurons” or nodes, and “edges”, which represent the connections between the nodes. Neural network architectures such as MLPs and LSTM-based networks are well suited for multi-step time series regression problems, as they can be trained to predict the entire output sequence with one model. LSTMs contain internal state variables that capture the entire history of the time series, allowing the model to learn from events occurring prior to the prediction period input window. LSTM layers can be combined to build larger, deeper, and more complex model architectures that can generate forecasts from high-dimensional large-scale datasets.

The architecture of the encoder-decoder model consists of two components referred to as the *encoder*, which encodes the features, and *decoder*, which attempts to reconstruct features based on a compressed state of the input data. The encoder network learns a representation of the

input sequence that captures its characteristics or context and produces a context vector. The *decoder* network receives the context vector and learns to read and extract the output sequence from it. Both components contain an LSTM layer which captures and interprets the sequential input data. An overview of the architecture for the encoder-decoder neural network is displayed in Figure 8. Historical features refer to features that are only known in the past, but not the future. Future features are those known in the future, such as the month or fiscal quarter.

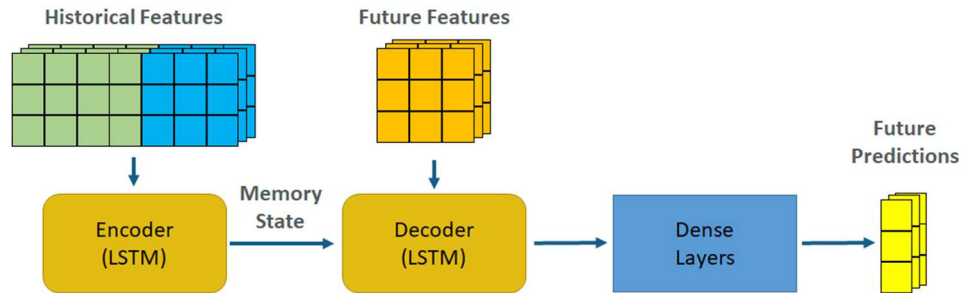


Figure 8: Encoder-Decoder Architecture

Model Validation and Testing Strategy

To perform a robust analysis on the performance of each of the six models, the team implemented a *walk-forward k-fold cross validation* methodology, with a total of three training and testing folds. This allowed the team to evaluate the performance of trained and tested models over several temporal segments within the data and use the average performance as shown in the Modeling Results section.

Although it is typical to take random samples of the training data as the test set in k-fold cross validation, the dependency of sample ordering in time series data necessitates a different approach. Here, the IBM team split the time series data into several segments based on a “windowing” function, where each split is then further partitioned into a training and testing window. In this paradigm, each training window comprises of the current and historical features (input) of the time series and their 12-month backorder counts (output) and the linked testing window represents a future set of time series and target data.

Each model is trained, tested, and evaluated separately on each fold, for a total of three different models. The results are then averaged together across each fold to obtain the produced performance across time series. Once the model has generated the predictions, the predictions are compared against the actual backorder events that occurred during the forecast horizon. An example of the walk-forward cross-validation data is displayed below in Figure 9.

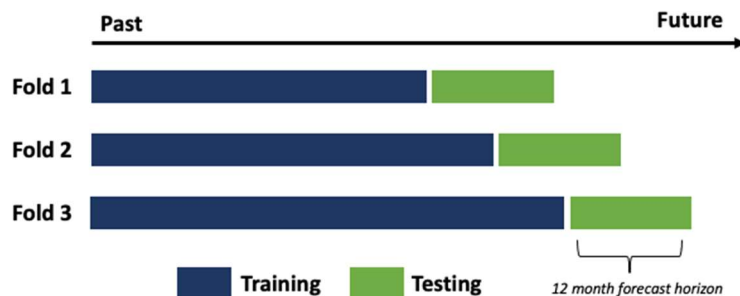


Figure 9: Cross Validated Sliding Window Paradigm

Modeling Results

Model Comparison Evaluation

Below is a walk-through of the results from the machine learning research. Each chart represents the average cross-validated result over each forecasted time horizon (1 to 12-month predictions) for the chosen evaluation metrics. The machine learning algorithms were evaluated on sets of test data in order to gauge their general performance on the backorder prediction problem using eight different metrics as evaluation criteria. The results of the model predictions were evaluated using the following regression metrics:

- Root Mean Squared Error
- Mean Absolute Error
- Explained Variance
- R-Squared

The following classification metrics were generated for both the backorder and in-stock classes:

- Receiver Operating Characteristic – Area Under Curve (ROC-AUC)
- F-1 Score
- Precision
- Recall

Regression Results

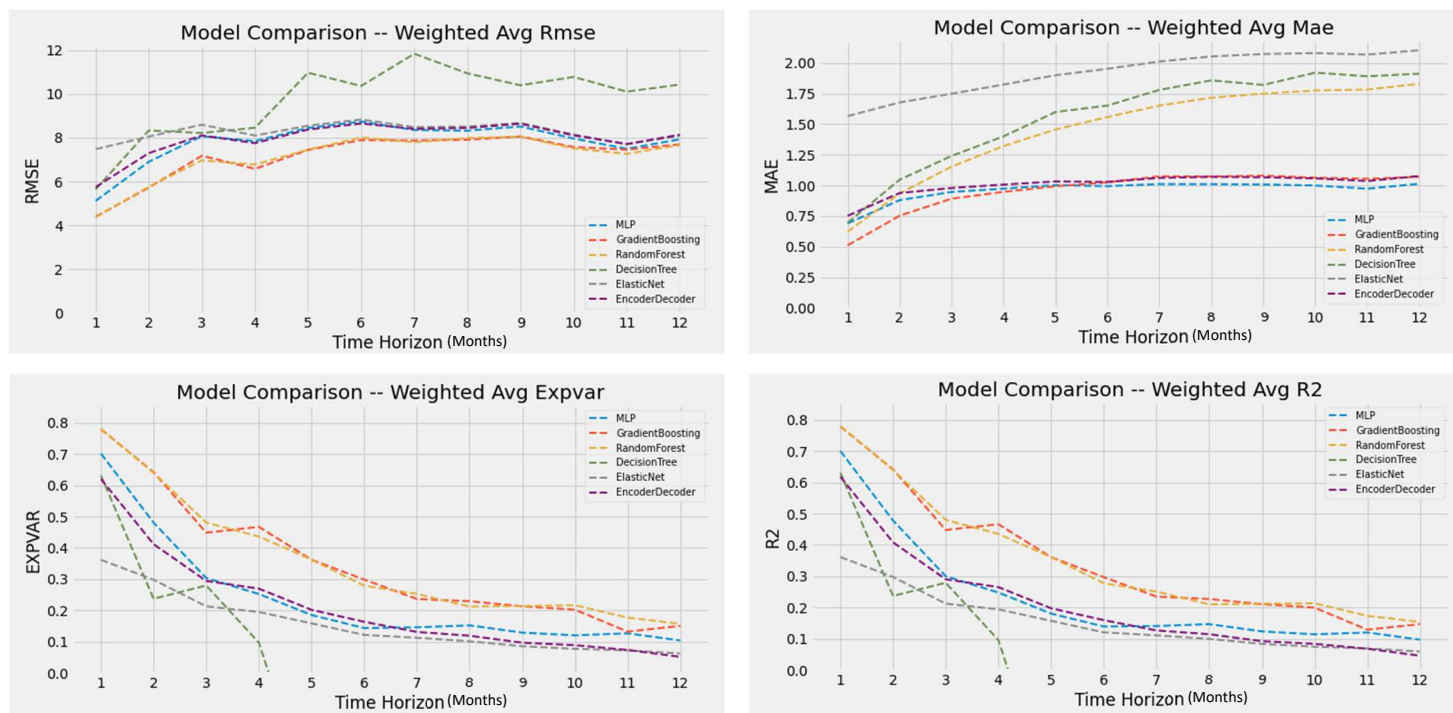


Figure 10: Regression Metric Model Comparison

In Figure 10, the IBM team present model results from four metrics used to analyze and interpret regression models—root mean square error (RMSE), mean absolute error (MAE), explained variance, and R-squared score, plotted across all time horizons.

Analyzing the MAE (top right) graph—which can be interpreted as how large of an error one can expect from a forecast, on average—one may see two distinct clusters of models. The IBM team interpreted this to mean that, on average, the difference between the forecast and the ground truth (labeled target data) is the closest for the MLP, encoder-decoder, and gradient boosting models. However, looking at RMSE (top left)—the standard deviation of the prediction errors—one sees that the gradient boosting and random forest models outperform the others. In fact, these two models outperform the others to a statistically significant degree in three of the four metrics (RMSE, R-squared, and explained variance).

As a general trend, each of the model’s predictive power degrades as they generate predictions for months further out from the present within the forecast horizon. One possible intuitive explanation for this is that, at the time of prediction, there is not yet a clear signal of an oncoming backorder.

Classification Results – Backordered Class

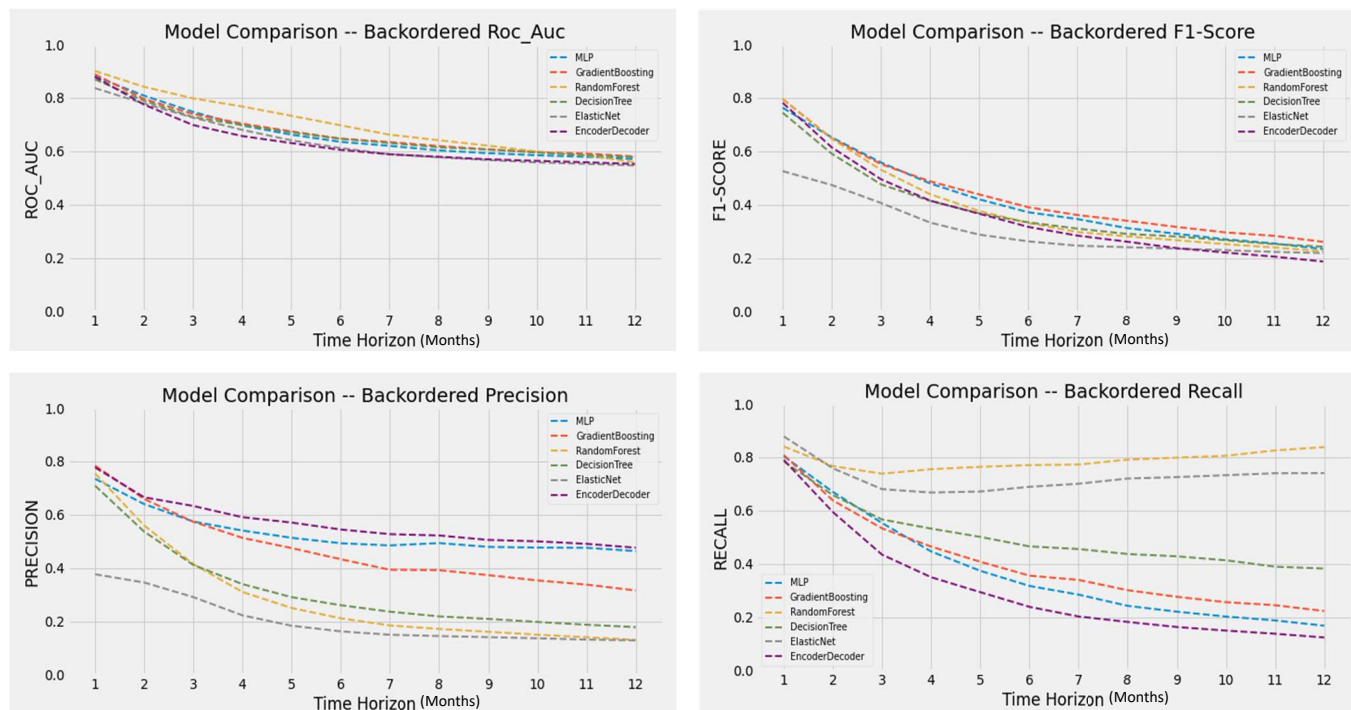


Figure 11: Classification (Backorder Class) Model Comparison

Figure 11 displays the results for each model for the four metrics used to analyze and interpret classification results for the backorder class—ROC-AUC, F-1 score, precision, and recall. The primary differences between the models can be seen when comparing precision and recall measures. Precision (also known as *positive predicted value*, or PPV) can be interpreted as the proportion of correctly classified items to the total items labeled with that classification. Recall (also known as sensitivity) refers to the proportion of total data points that are classified correctly by the model. Typically, there is a trade-off between precision and recall for a given classification model. For example, a model may be less precise (i.e., less certain a random item picked from a class is classified correctly), but able to recall more backorders. There is a distinct trade-off between the models that are able to maintain a higher precision across the forecast period—MLP, gradient boosting, and encoder-decoder architectures—and the other models, where precision in classifying backorders degrades by 40 to 50% over the time horizon, on average. Conversely, the three models with the highest average backorder precision also have the lowest average recall of backorders over the forecast horizon. This can be interpreted as the models that maintain high precision are more *conservative* with their forecasts, only predicting backorders when there is a clear indication of a future backorder.

Classification Results – In-Stock Class

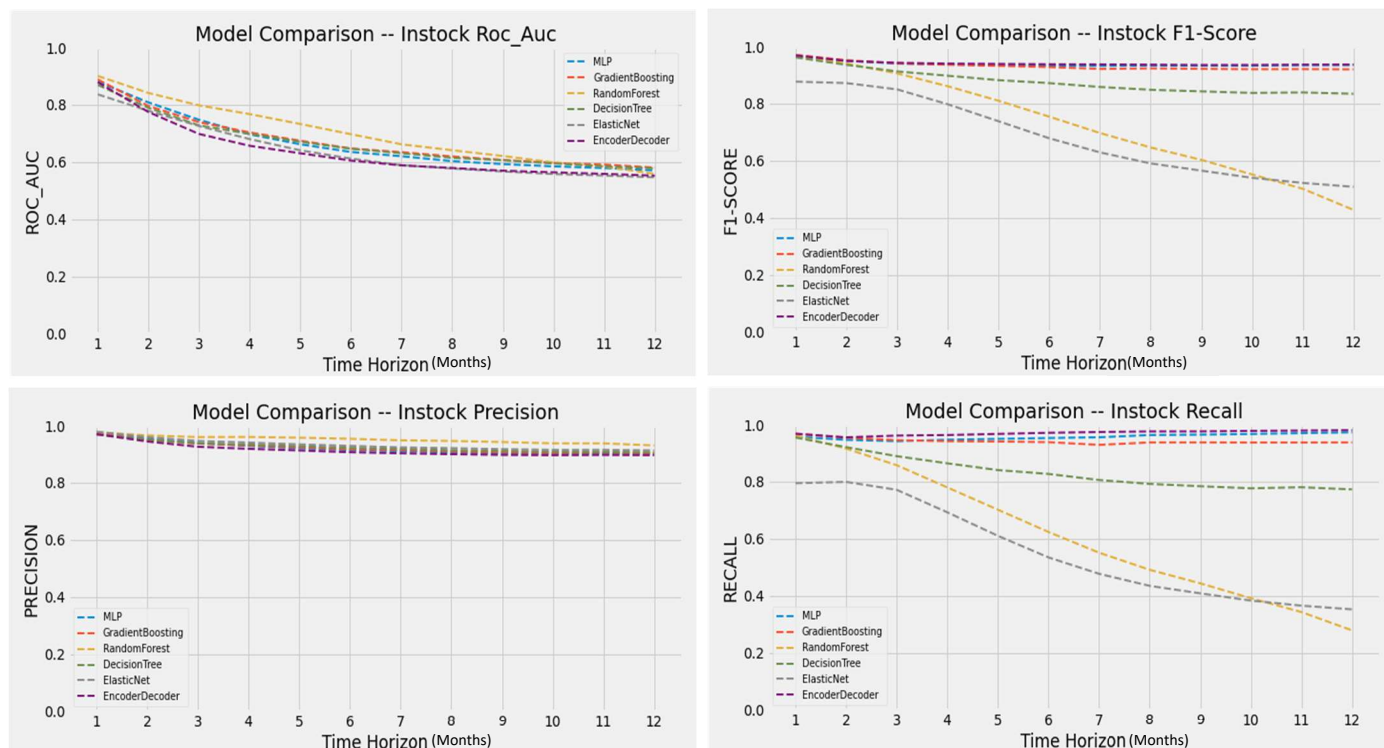


Figure 12: Classification (In-Stock Class) Model Comparison

In Figure 12, the model results for the four metrics used to analyze and interpret classification models are displayed for the in-stock class—ROC-AUC, F-1 score, precision, and recall. Similar to the backorder class, there is a wide discrepancy between the performance of the top three models (MLP, gradient boosting, encoder-decoder) and the others as one expands further out in the forecast horizon. The top performing models can maintain a recall of in-stock items between 97-94% in our cross-validated average, while there is a statistically significant drop-off in recall for the decision tree, elastic net, and random forest models. This can be visualized by simulating backorders over time using one of the test set folds in the figure below:

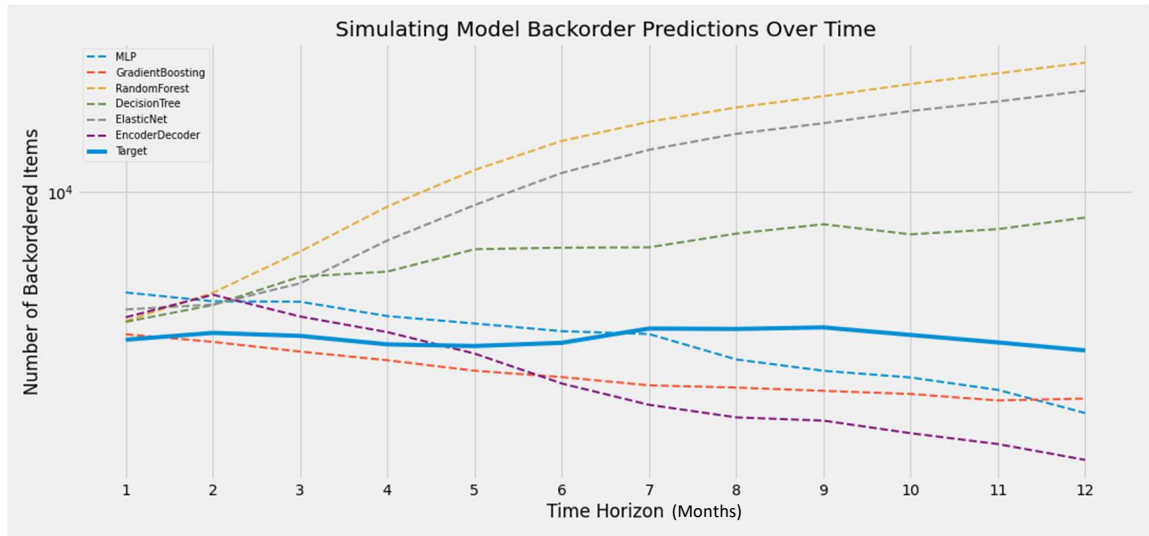


Figure 13: Backorder Simulation

The models with higher precision in predicting backorders tend to forecast an underestimate of backorders the further the prediction is from the present. These models provide more conservative forecast estimates, only predicting a backorder when there is a strong signal indicating a backorder will occur. The models with low precision in predicting backorders tend to overestimate the number of backorders over the same forecast horizon, and although they have higher backorder recall, lead to more false positives in backorder identification.

Sample Backorder Forecasts

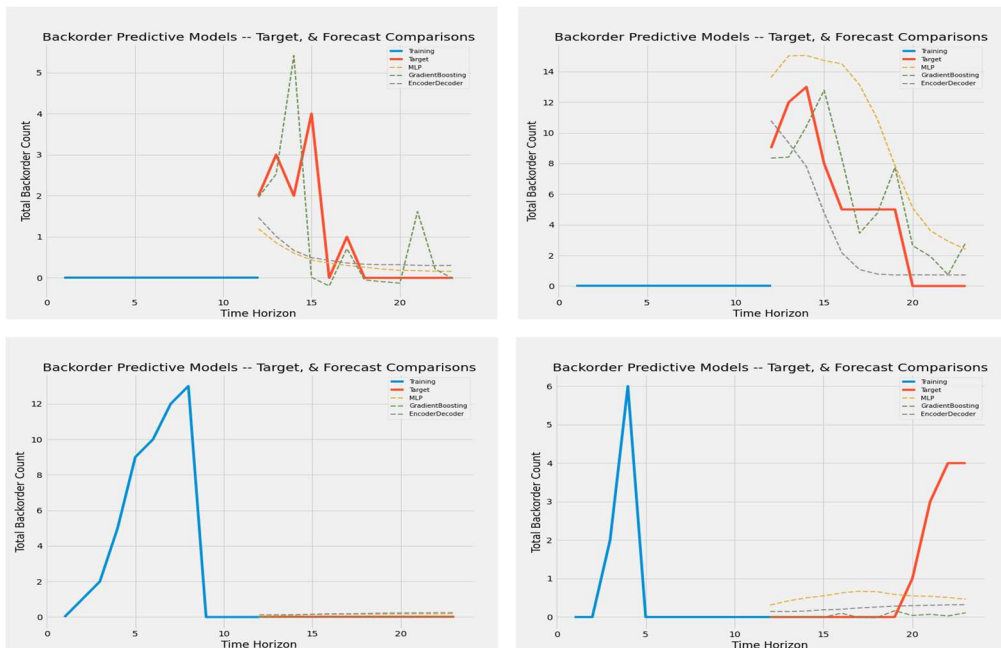


Figure 14: Sample Model Forecasts

In Figure 14, four sample forecasts are provided, with the backorder sequence the model was trained on, the target (ground truth) sequence the model is tested against, as well as the forecasted backorder count as predicted by the three models with the greatest precision.

The graphs in the top row, provide forecasts for two different items with future backorders. Note that in both cases, the models pick up on the initial backorder occurring even without the presence of a backorder in the item's training data.

The bottom left plot displays the models' ability to generate stable, precise forecasts for currently healthy items in the high-risk item population—even those with a history of backorders. Since the vast majority of items will not have backorders in any single 12-month future window, this is a necessary test to validate that the models do not exhibit erratic predictive regression behavior for NIINs with healthy characteristics moving forward into the future.

In the bottom right plot, it can be seen that the models essentially miss the backorder that occurs later in the forecast horizon. From the degradation in R-squared from the regression metrics, occurrences of this happening might seem obvious to occur—none of the models find a signal in the current data that suggests a backorder will occur later in the product's history. This is a shortcoming that the IBM team will address later in the Recommendations for Future Enhancements section.

Statistical Significance Testing for Model Comparison

While various metrics can be used to estimate the performance differences between algorithms, it can be challenging to determine whether the differences between skill scores are true or the result of a statistical fluke. Therefore, a statistical significance test can be applied to determine whether two models, A and B, perform differently.

The core assumption of statistical significance test is the null hypothesis, which is a hypothesis that states that the parameters or mathematical characteristics of two or more populations are identical. To determine the significance between models, a statistical measure is computed and compared to the null hypothesis in order to decide whether to accept or reject the null hypothesis. In this study, the IBM team implemented the k-fold cross-validated paired t-test to evaluate the significance of model performances for pairs of models.

The k-fold cross-validated paired t-test is performed by training and evaluating two models on each of the k folds of data and computing the t statistic based on the difference between the model performances for a given metric. The paired Student's *t* statistic is calculated using the following equation:

$$t = \frac{\bar{p} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n-1} (p^{(i)} - \bar{p})^2}{n-1}}}$$

where $\bar{p}$ is the mean difference of scores between models A and B over the folds and $p^{(i)}$ is the score difference for the i^{th} fold where $p^{(i)} = p_A^{(i)} - p_B^{(i)}$. Under the null hypothesis, t has a t distribution with $k-1$ degrees of freedom, where k is the number of folds. The null hypothesis can be rejected if $|t| > t_{k-1}$ for a given confidence interval. When the t score is below the threshold, then the null hypothesis is accepted and both models are statistically equal.

In this study a t statistic threshold of $t_{2,0.95} = 4.303$ was selected from the two-tailed t -distribution values table for 2 degrees of freedom at a 95% confidence interval. The t scores for each of the 12 months in the forecast horizon within each of the test folds were calculated for each pairing of models for each of the classification and regression performance metrics. To facilitate model comparison, an aggregated view of the t scores was generated by averaging the paired t scores across the 12-month horizon to facilitate the comparison between models. Figures 15 and 16 display the averaged k -fold cross-validated paired t -test scores for the model pairing between the gradient boosting algorithm and each of the other algorithms for both the classification and regression metrics for the backordered class. The gradient boosting algorithm was chosen for the examples since was a top performer for both the classification and regression performance metrics.

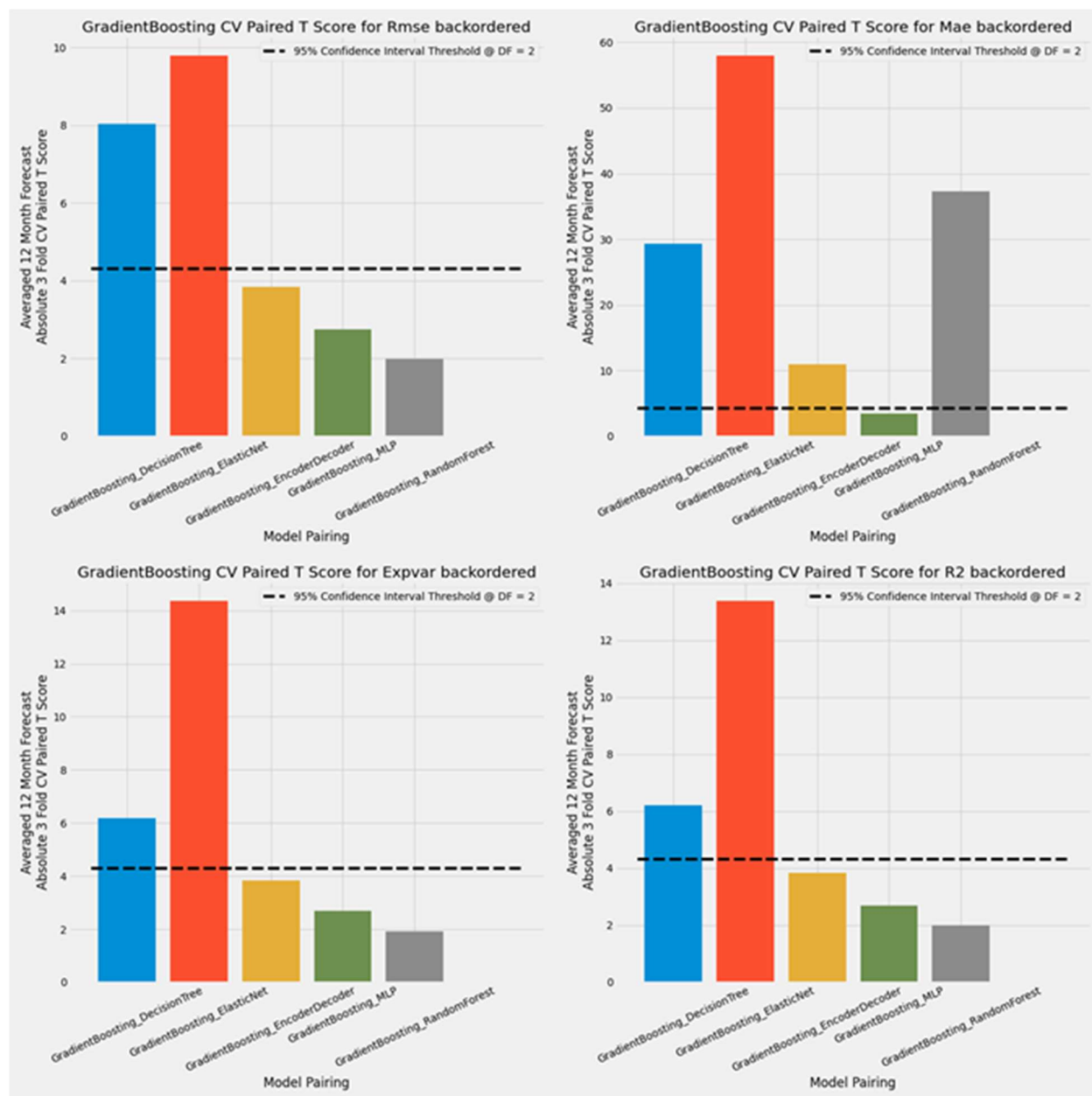


Figure 15: Averaged 3-Fold Cross-Validated Paired T-Test Scores for Regression Metrics

In terms of the regression metrics, the null hypothesis can be accepted across all four classification metrics for the gradient boosting and MLP pairing since the MLP t scores are below the t threshold $t_{2,0.95} = 4.303$ line. The encoder-decoder network is the next most similar algorithm to the gradient boosting algorithm since the null hypothesis is accepted for all the metrics except for the MAE score, where the paired t score displays a slightly elevated t score above the threshold. The remaining model pairs reject the null hypothesis, and therefore perform significantly differently compared to the gradient boosting algorithm.

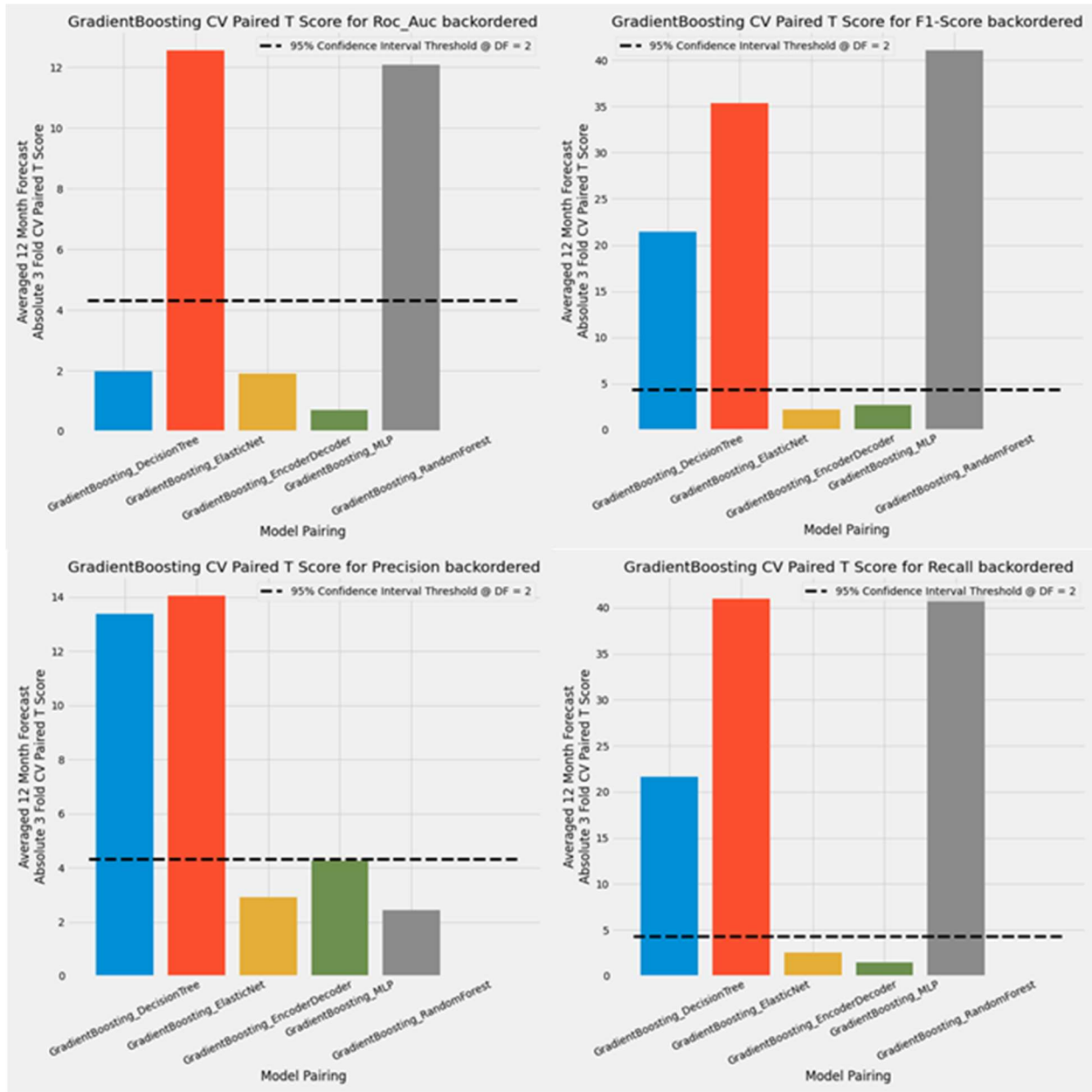


Figure 16: Averaged 3-Fold Cross-Validated Paired T-Test Scores for Classification Metrics

When comparing the averaged t score pairs classification metrics for the gradient boosting algorithm pairs, the most similar algorithms to the gradient boosting algorithm are the encoder-decoder and the MLP models, since the null hypothesis can be accepted for all of the classification metrics. Whereas the null hypothesis is rejected for the vast majority of the metrics for the decision tree, elastic net and random forest pairs, indicating that these models perform significantly differently than the gradient boosting algorithm.

Model Interpretability

One common concern with complex machine learning architectures such as neural networks or sophisticated ensemble techniques is the lack of transparency in the inner logic the model uses to make these predictions (i.e., “a black box” model). While our modeling strategy goal is to make the most accurate model possible, the IBM team also understands that the ability to interpret the results and reasoning behind the predictions is equally as important. In this section the IBM team will discuss the feature importance plots for a subset of the models (the remaining model importance diagrams can be found the Appendix C.) and how we used a widely used model interpretability algorithm (SHAP) to further explore our model predictions.

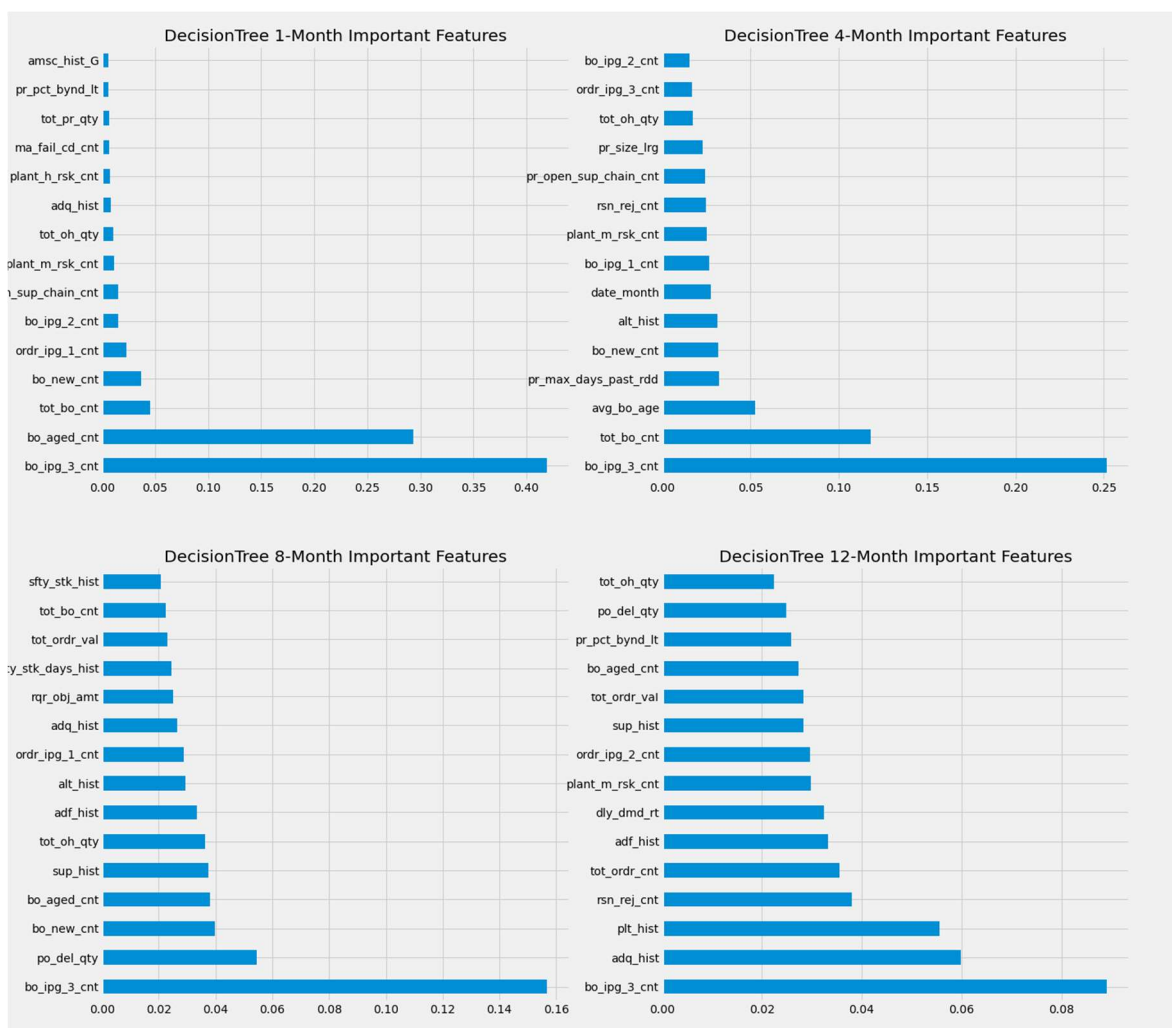


Figure 17: Decision Tree Feature Importance

In the group of plots above, the IBM team have the most important features as determined by the decision tree model. Decision trees use a rule-based logic to learn how best to split the data to predict the target feature—important features are determined by how well a node is able to reduce the mean squared error criterion (also known as Gini importance). In the near-

term, the decision tree model assigns the highest importance to backorder features such as the count of issue priority group 3 backorders, aged backorders, and total backorders. In the long-term, the decision tree model assigns historical data markers more weight as the important features such as lead times, historical annual demand frequency, and historical annual demand quantity.

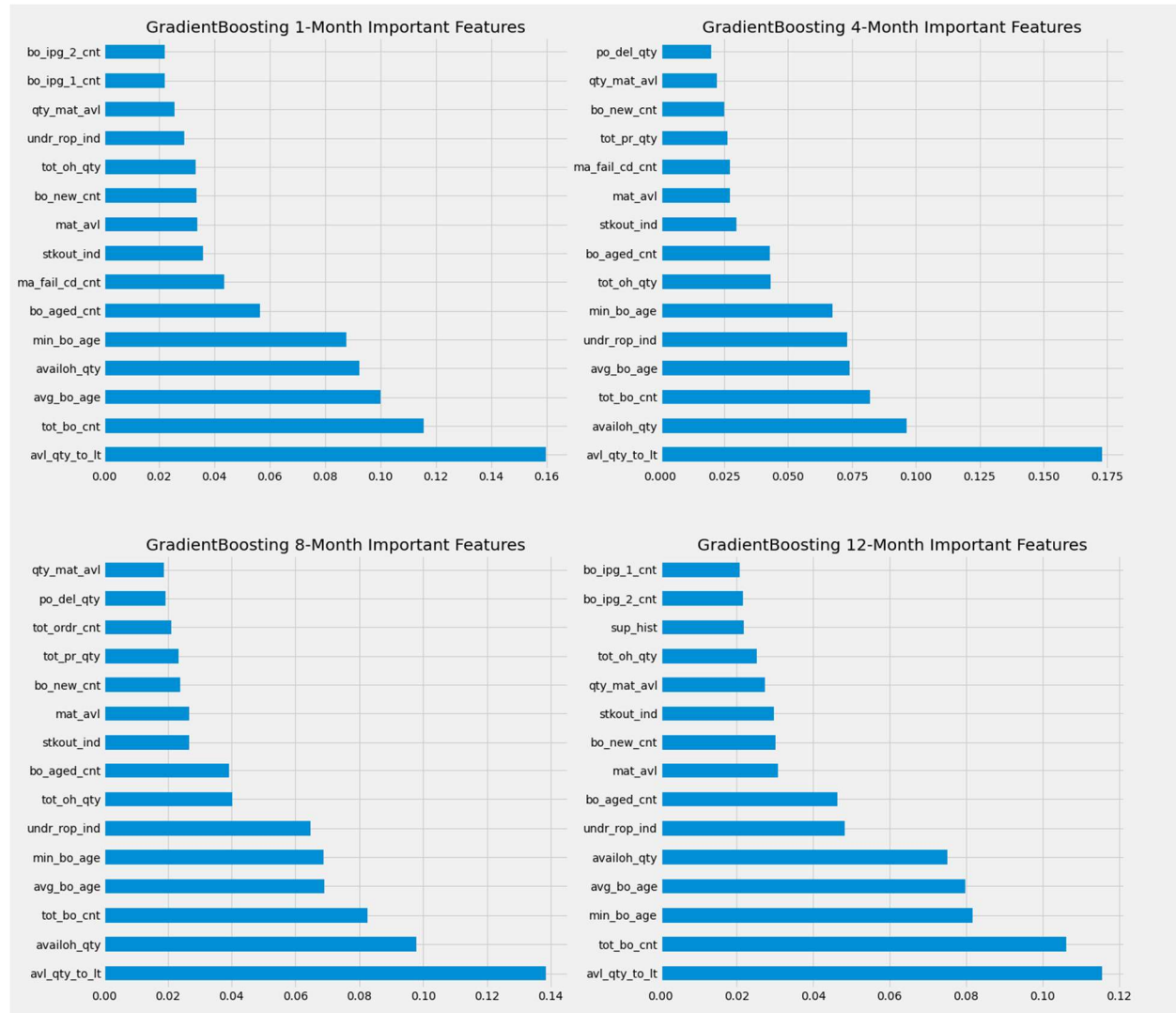


Figure 18: Gradient Boosted Trees Feature Importance

Although the gradient boosting model uses an ensemble of decision trees to make predictions, one can clearly see in Figure 18 that the two models attribute importance a different set of features. The most important features in the gradient boosting model are not only the common backorder features, but other engineered features like the available quantity to lead time ratio, material availability, and under reorder-point indicator. In fact, twelve of the top fifteen features in the first month prediction are features engineered by the IBM team that were not a part of the original dataset. The relative importance of the available quantity to lead time ratio and backorder age features in all time horizons may be a potential signal that the model identifies

a relationship between the lead times of an item and how long an item is expected to stay on backorder.

Interpretability Methodology – SHAP

To provide insight into the output of our model, the IBM team used the SHAP (SHapley Additive exPlanations) algorithm—a game theoretic, model agnostic method that underpins the interpretability of the model. The goal of SHAP is to assign importance *values* to model features, closely linked to Shapley values from cooperative game theory, where the feature set is a coalition and the values tell one how to distribute the payout (i.e., prediction) among the feature space.

SHAP values quantify the marginal contribution that each feature provides to the prediction made by the model, providing a way to visualize and interpret the prediction on two levels:

- Local—each individual prediction has a set of SHAP values that can be used to explain feature importance on a single forecast.
- Global—SHAP values can also be aggregated on a dataset-level to provide insight into feature importance on the entire population of data.



Figure 19: SHAP Force Plot

Figure 19 displays an example force plot—which displays the features that “push” the model output from the base value towards the model output higher or lower (higher to ~8 predicted backorders in this case). In this specific example, the features with the biggest contribution are the total backorder count (tot_bo_cnt) and aged backorder count (bo_aged_cnt), which makes intuitive sense as one expects current and recent existing backorders to have a significant impact on whether the model predicts future backorders to occur for an item. One can also see the features that work to “pull” the output down, such as the number of IPG 3 orders and large size purchase requisitions. One may create this force plot to study the predictive explanation behind any forecastable item to analyze interesting features and their perceived impact on the models.

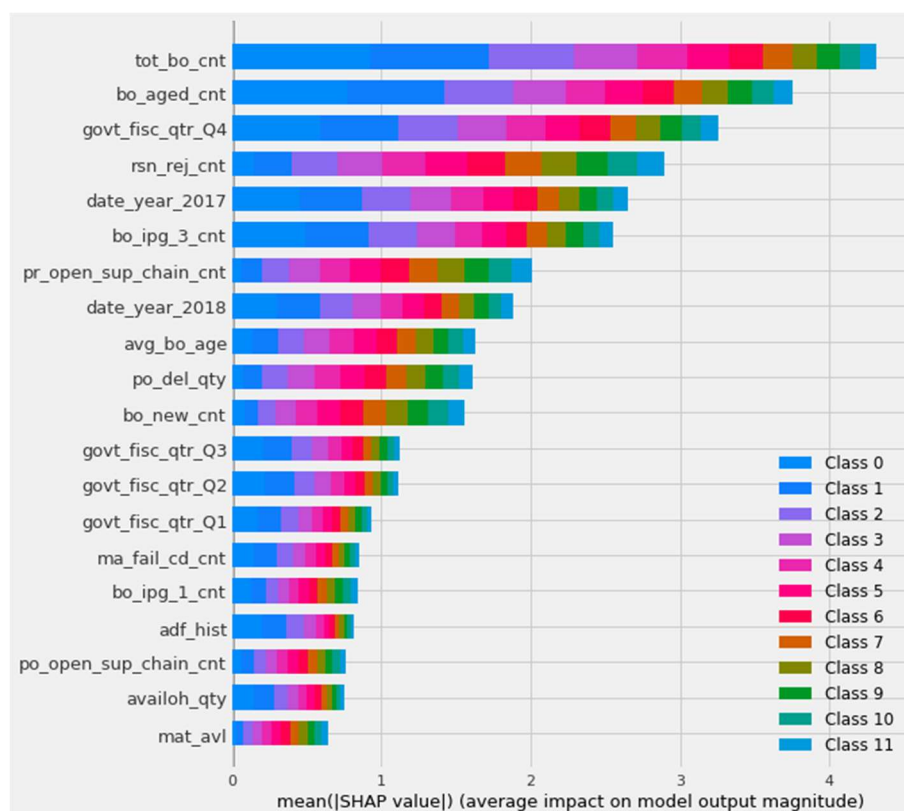


Figure 20: Neural Network Feature Importance

In Figure 20 the IBM team have displayed the top-20 features based on SHAP values for one of the cross-validated folds of the multilayer perceptron neural network. The IBM team sees that backorder-dependent features such as the count of aged backorders, total backorders, and average backorder age have a significant impact on the model predicting future backorder count values, with their relative importance decreasing as one pushes further out in the forecast horizon.

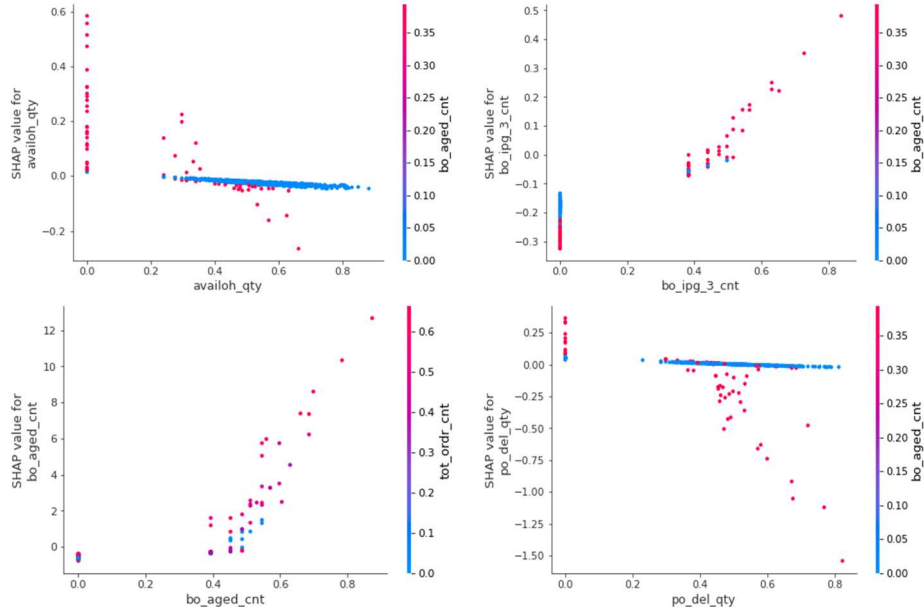


Figure 21: SHAP Dependence Plots

To understand how a single feature contributes to the model output, the IBM team created dependence plots that highlight the interactions between each feature and its SHAP value. For example, in the top left plot of Figure 21, one may see that only low values of available on hand quantity of stock contribute to positive SHAP values—in other words, the higher the available on hand stock value the less impact it has on the model’s prediction. The IBM team observed a similar but opposite trend with the purchase order deleted quantity, with higher quantities contributing negatively (lower predicted backorders), but only in the presence of aged backorders (note the coloring). These types of plots can reveal hidden relationships between features and provide DLA with even deeper insights into model predictions.

Combining both global and local interpretability allows one to cast both a wide and deep view of the decision boundary of the model’s forecast predictions.

SHAP Feature Importance for Backorder Scenarios

To gain further insight into features that contribute to backorder predictions, the IBM team explored the MLP model’s ability to generate predictions for several different backorder scenarios. The team grouped NIINs into paired 2-class categories by identifying backorder patterns within last 24 months of 60-month time series for each NIIN. The 24-month period was split into historical and future sequences consisting of 12-months each and both sequences were tagged according to the following patterns: flat, spike, and plateau. Sequences that were classified as flat were characterized by the absence of backorders during the entire 12 months. The spike classification was applied to any sequence that contained the presence of 1-2 backorder events occurring in consecutive lags. Any sequences that were characterized with backorder events across three or more consecutive lags received the plateau classification. Each

of the three pattern classes for the historical and future sequences are displayed in Figure 22 and includes the classification criteria for each pattern. All three classes were rearranged in different combinations of historical-future sequences for a total of nine unique pairs.

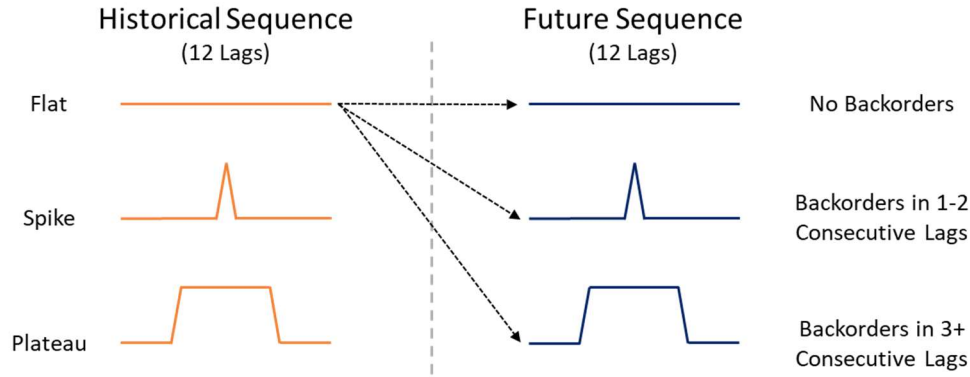


Figure 22: Historical and Future Backorder Count Sequence Pattern Pairing

Using the nine historical-future sequence class pairs, a total of nine subsets of NIIN populations were isolated to create training and testing datasets for each scenario using the last fold of the cross-validation fold set. Next, nine MLP models were fit and evaluated using the respective training and testing datasets for each class pair. Each was model analyzed using the SHAP model explainer to generate feature importance graphs for the top 20 features based on SHAP values.

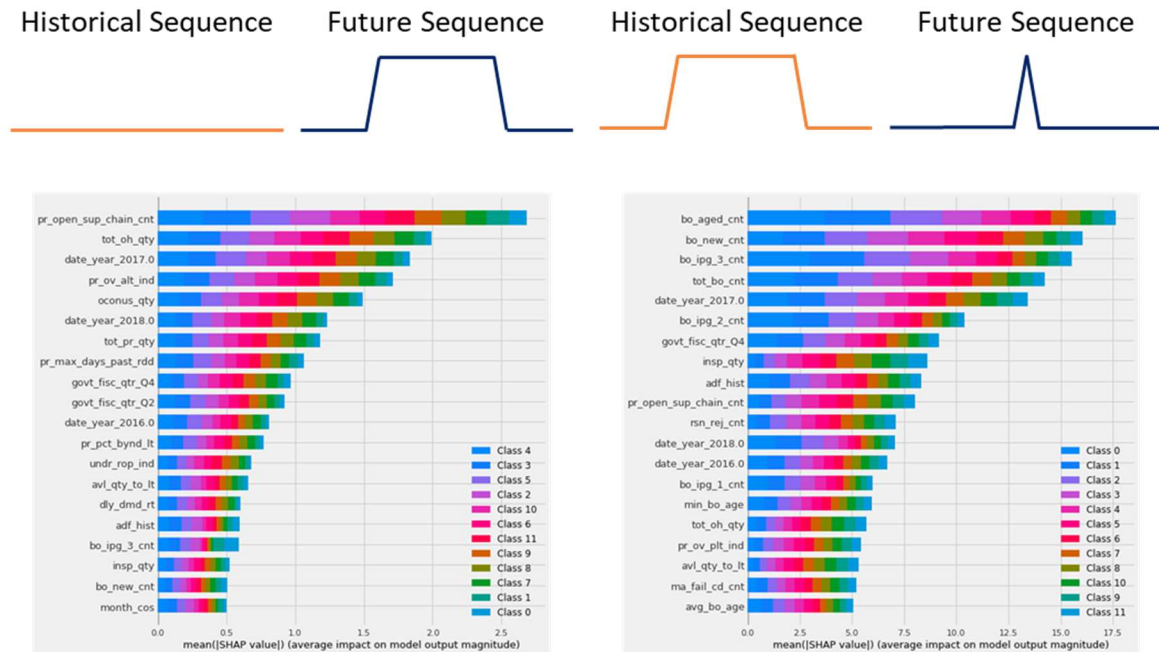


Figure 23: Neural Network Feature Importance for Backorder Count Sequence Pairs

In Figure 23, IBM considers two scenarios that consist of backorders occurring in the future sequence. These cases provide further insight into the predictions made by the neural network and highlight the decision boundary used to generate forecasts for different backorder scenarios.

The scenario in the left graph of Figure 23 presents a situation in which backorders are not present in the historical sequence (i.e., item has been in-stock for the last year), while there is a sustained plateau of backorders in the future sequence. The key features that contribute to the model's predictions are the number of open purchase requisitions in the supply chain, the total on hand quantity of an item, and the presence of open purchase requisition orders exceeding the administrative lead time. In fact, purchase requisition features make up a significant portion of the top features in this scenario.

For the scenario in the right graph, the historical sequence consists of large, sustained quantities of backorders whereas the future sequence consists of a short spike in the backorder count at some point in the sequence. In this scenario, the primary features that contribute to the model's ability to predict future backorder spikes are the count of aged backorders, count of newly established backorders, and items that possess an issue priority group of three.

This analysis demonstrated the ability to isolate populations of NIINs to create scenarios in which models can be trained and analyzed to understand the features that contribute to predictions for various backorder patterns. A complete set of paired sequence feature importance results for each of the MLP models is provided for the nine scenarios in Appendix D.

Conclusions

Through this research effort, IBM has developed supervised machine learning models using curated DLA-provided inventory data that forecast item backorders for each month over a 12-month forecast period. The top performing models developed in this research (gradient boosting algorithm, the multilayer perceptron, and encoder-decoder neural networks) predict future in-stock items with a quarterly forecast accuracy between 95-93% over the forecast year, and future backordered items with a quarterly forecast accuracy from 71% in the first three month period to 25% in the last three month period of the forecast year.

Further, IBM has identified several typical scenarios of backorder occurrences (backorder spikes and plateaus) as well as the key driving factors that contribute to these scenarios such as the "age" of existing backorders, the ratio of available inventory to the item's lead time, and the number of open purchase requisitions in the supply chain. Integrating explainability methodologies like SHAP provide DLA with both global and local interpretability into the driving factors of the model predictions.

From this research, DLA may further enhance and fine-tune the IBM developed machine learning data pipeline through integrating additional data sources, scaling to more item groups,

and developing visualizations on top of the provided interpretability framework to better understand backorder contributing factors. The foundation provided by this research will provide DLA with actionable insights in anticipating backorders, increasing product availability to its customers.

Recommendations for Future Enhancements

The sections below discuss some additional future enhancements that may provide value to improving results and insights of the backorder prediction models.

Model Segmentation

As a supplemental enhancement to the core set of supervised models, IBM recommends leveraging the robust modeling pipeline developed by IBM to train additional segments of supervised models in order to gain forecast coverage on item populations outside the scope of this effort. For example, DLA can use the provided code to model on top of NIINs with different AAC codes, less item history than the 60 months observed in this report, or items with lower annual demand. This kind of segmented approach would provide a straightforward framework for DLA to achieve backorder forecast coverage on *all* of their inventory in a reliable and scalable manner, while training models with predictive capabilities over each of the item populations.

To demonstrate this concept, IBM took a subsample of NIINs more representative of DLA's entire inventory (removing exclusion criteria related to annual demand and a history of backorders) and ran it through the modeling pipeline to generate a decision tree model. Below we compare high level metrics with the model trained on the representative sample and the decision tree model trained on the more selective criteria on a test set of the new sample and evaluated results on the last test fold.

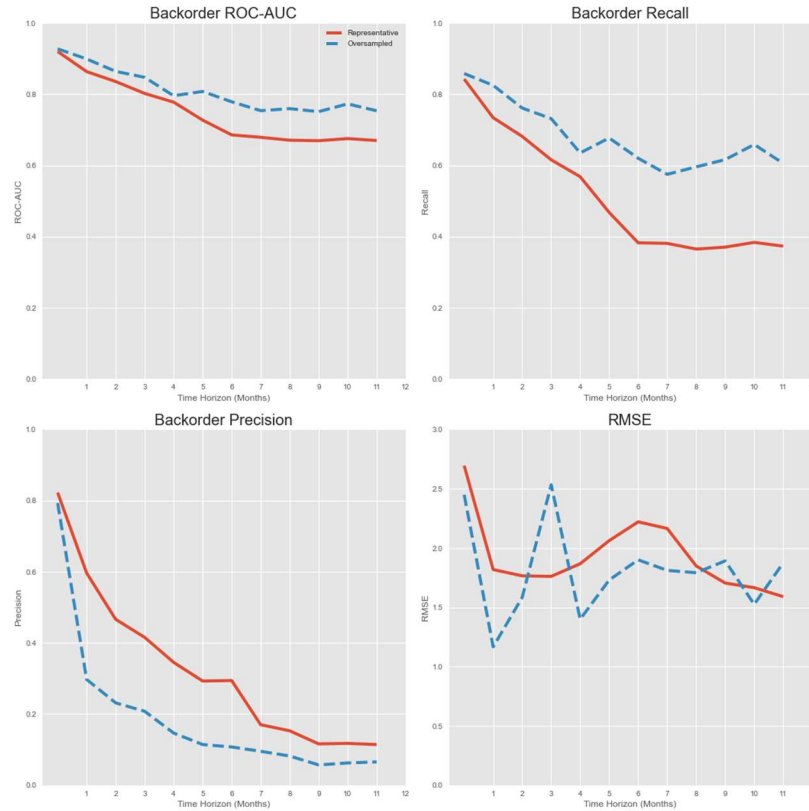


Figure 24: Representative Sample Decision Tree vs. Oversampled Decision Tree Models

This experiment shows that the decision tree trained on the entire list of exclusion criteria has a higher recall of backorders on the test set of the representative sample, but it is also less precise at predicting backorders than the model trained on the representative sample overall. This can be attributed to the exclusionary criteria oversampling the backorder class, so it is able to “see” more backorders during training and thus identify them in the test set more often. In this scenario, which of the models provides more insight is dependent on DLA’s priorities—identify more backorder events in total, or eliminate more false positives by higher backorder classification precision. This segmentation approach to modeling, combined with the flexibility of IBM’s developed modeling pipeline, provides DLA the flexibility in choosing a modeling strategy aligned with its short and long-term goals.

Data Augmentation

Data augmentation is the process of adding additional data by incorporating slightly modified copies or generating synthetic data from the existing training data. This technique can be useful in instances of small datasets, or significant class imbalances in the dataset (the backorder class compared to in-stock class, for example) exist. One method the IBM team proposes that might warrant additional consideration is experimenting with different resampling techniques to drive higher backorder prediction recall.

The original training dataset is imbalanced based on NIIN backorder count histories, resulting in groups of NIINs that form a majority class with several minority classes. This bias in the training dataset can lead to machine learning algorithms that ignore the minority classes entirely. To overcome this issue, the team explored oversampling and under sampling techniques. An overview of the NIIN grouping is displayed in Figure 22.

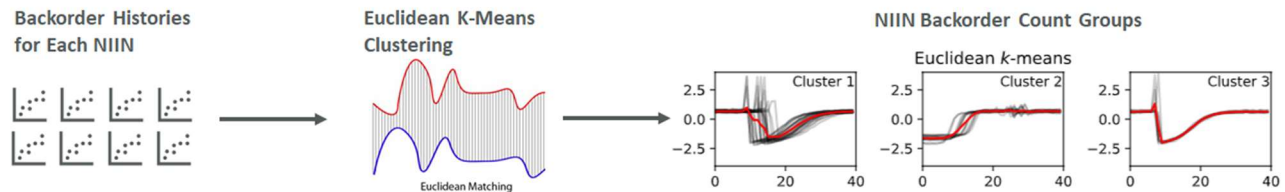


Figure 25: NIIN Backorder History Grouping Pipeline

By clustering all the NIINs, one may be able to identify the groups of NIINs that formed the majority and minority classes, from which one could apply oversampling and undersampling techniques. Undersampling is the process of randomly deleting observations from the majority class to match the number of observations in the minority class. Oversampling is the process of randomly duplicating observations within the minority class in order match the number of records in the majority class. An example of undersampling and oversampling is displayed in Figure 23.

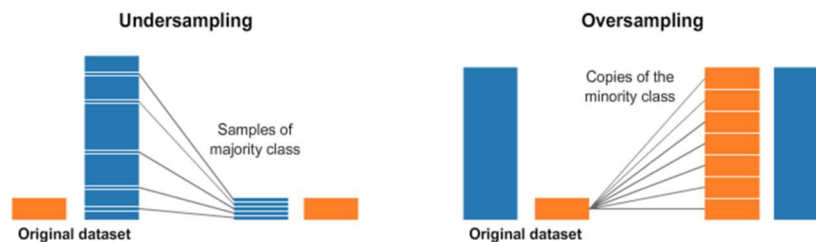


Figure 26: Undersampling and Oversampling Technique

Although the IBM team experimented with adding oversampling techniques to the data pipeline, it was decided that the additional benefit did not overcome the additional added code complexity. The IBM team recommends further experimenting with a more streamlined version of this methodology as a potential future enhancement to improve overall backorder recall.

Hyper-parameter Tuning

One of the challenges in training a machine learning algorithm is selecting the optimal hyper-parameters, which can be time intensive. Hyper-parameters are parameters set outside of the model training boundary that can significantly impact model results. Parameters like loss

functions, training epochs, batch sizes, or tree depths can be thought of as hyper-parameters that must be “tuned” to extract the best possible model. For this research, hyper-parameters were tuned manually through multiple experiments, which requires a human in the loop to manually run training experiments and compare results against one another. Through the course of this research project, the IBM team have tested a method to automate this activity for neural network models using TensorBoard, an interactive visualization toolkit from TensorFlow. TensorBoard would allow the user to visualize how each parameter contributed to the training and validation loss, as well as how the model architecture itself (number of dense layers, dropout layers, etc.) performs on the problem. Additional hyper-parameter tuning search algorithms such as random search or grid search may be explored to provide an efficient way of generating the best possible predictive model for each training iteration.

Incorporating Additional Data Sources and Signals

Another area of potential enhancements for the models is the incorporation of additional sources of data that might contain high value features. High value features are those that provide a clear link to the target variable. For example, leveraging additional demand-side data, if available, may help provide strong signals of future demand that could in turn help the models understand when and of what magnitude future backorder events might occur.

To test this hypothesis of incorporating high-value features—those that contain a strong future signals of item inventory—and how those features might impact our models, the IBM team ran a test using the true value of available on hand quantity of stock in each future month as additional model features. As seen in Figure 24 below, this gave the subset of models the IBM team tested this approach against a greater ability to predict backorder events—even those occurring farther in the future—with significantly improved accuracy. This simulation shows a clear path to the potential for further significant improvements in the models if any accurate forecasts of item inventory data can be incorporated or established.

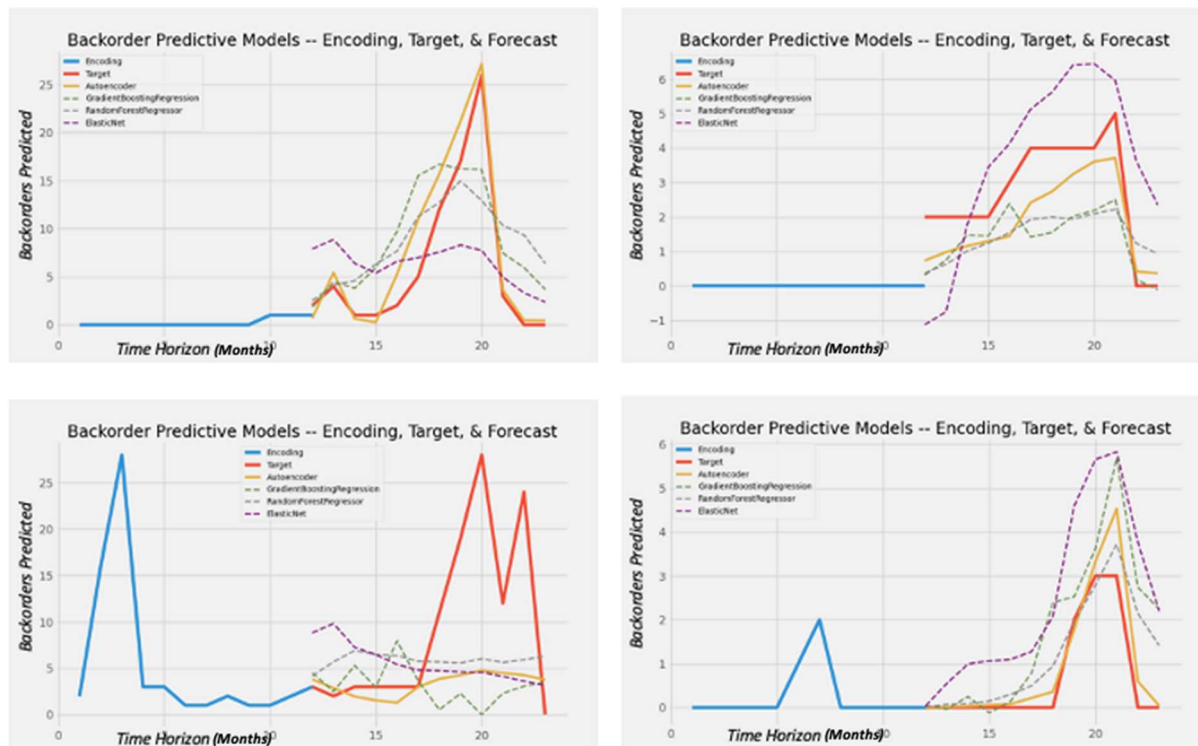


Figure 27: Effect of High Value Signaling Features in Model Results

Appendices

Appendix A. Comprehensive List of Model Features

The following table provides a list of all features used as model inputs for each of the models, as well as a short description detailing the meaning of each feature. The features used in the modeling are a combination of those taken straight from the raw data (e.g., availoh_qty, insp_qty, prot_qty) and those engineered by IBM with help from DLA SMEs (e.g., min_bo_age, mat_avl) that provide additional predictive power to the models.

Feature	Description
month_sin	month of year (sine transformation)
month_cos	month of year (cosine transformation)
date_year_2016	year of 2016 indicator
date_year_2017	year of 2017 indicator
date_year_2018	year of 2018 indicator
date_year_2019	year of 2019 indicator
govt_fisc_qtr_Q1	first fiscal quarter indicator
govt_fisc_qtr_Q2	second fiscal quarter indicator
govt_fisc_qtr_Q3	third fiscal quarter indicator
govt_fisc_qtr_Q4	fourth fiscal quarter indicator
amsc_hist_C	acquisition method suffix code (AMSC) C indicator
amsc_hist_G	acquisition method suffix code (AMSC) G indicator
fsc_hist_53	federal supply class (FSC) 53 indicator
undr_rop_ind	under reorder point indicator
stkout_ind	out of available stock indicator
tot_stkout_ind	out of total stock indicator
pr_cncl_ind	PR cancel indicator
pr_ov_alt_ind	PR over administrative lead time indicator
pr_ov_plt_ind	PR over production lead time indicator
availoh_qty	issuable on hand quantity
prot_qty	protected quantity
blockd_qty	blocked quantity
insp_qty	inspection quantity
tot_oh_qty	total on hand quantity
oconus_qty	outside contiguous United States quantity
dly_dmd_rt	daily demand rate
rqr_obj_amt	required objective amount
sfty_stk_hist	historical safety stock
sfty_stk_days_hist	historical days of safety stock (safety stock/daily demand rate)
cvrgr_drtn_hist	historical coverage duration
adf_hist	historical annual demand frequency (ADF)
adq_hist	historical annual demand quantity (ADQ)
alt_hist	historical administrative lead time (ALT)
plt_hist	historical production lead time (PLT)
amc_hist	historical acquisition method code (AMC)
sucf_hist	historical sales unit conversion factor (SUCF)

sup hist	historical standard unit price (SUP)
avl qty to lt	available quantity to lead time ratio
bo new cnt	count of new backorders
bo aged cnt	count of aged backorders
tot bo cnt	count of total backorders
mat avl	material availability
qty mat avl	quantity material availability
nmcs cnt	count of not mission capable supply (NMCS)
bo ipg 1 cnt	count of issue priority group (IPG) 1 backorders
bo ipg 2 cnt	count of issue priority group (IPG) 2 backorders
bo ipg 3 cnt	count of issue priority group (IPG) 3 backorders
avg bo age	average age of backorders (in days)
min bo age	minimum age of backorders (in days)
tot odr cnt	count of total orders
tot odr qty	quantity of total orders
tot odr val	value of total orders
odr ipg 1 cnt	count of issue priority group (IPG) 1 orders
odr ipg 2 cnt	count of issue priority group (IPG) 2 orders
odr ipg 3 cnt	count of issue priority group (IPG) 3 orders
plant h rsk cnt	count of orders from high risk plants
plant m rsk cnt	count of orders from medium risk plants
plant l rsk cnt	count of orders from low risk plants
plant n rsk cnt	count of orders from no risk plants
ma fail cd cnt	count of material availability failure codes
rsn rej cnt	count of reason for rejections
odr dmd spk	order demand spike
pr pct bynd lt	purchase requisition (PR) percent beyond lead time
pr max days past rdd	purchase requisition (PR) max days past required delivery date (RDD)
tot pr qty	total purchase requisition (PR) quantity
pr open sup chain cnt	count of purchase requisitions (PR) yet to be delivered
pr prob late	probability of late purchase requisition (PR)
pr size sml	indicates small purchase requisition (PR)
pr size lrg	indicates large purchase requisition (PR)
po open cnt	count of open purchase orders (PO)
po del qty	quantity of deleted purchase orders (PO)
po open sup chain cnt	count of purchase orders (PO) yet to be delivered
po prob late	probability of late purchase order (PO)

Appendix B. Model Hyperparameters

The following tables detail a complete list of models and the associated hyperparameters that were used to tune the final model. Hyperparameters are values that are controlled by the user before the training process that can have significant impact on the evaluation outcomes for that model. Typically, the more hyperparameters required to tune a model the more sensitive the model results are to the values of those parameters (i.e., a neural network will train and evaluate much differently for different hyperparameter values, for example).

Model: Elastic Net

Hyperparameter	Value
Alpha	.15
L1 ratio	.95

Model: Decision Tree

Hyperparameter	Value
Criterion	Friedman MSE
Max depth	8
Min samples split	10
Min samples leaf	5

Model: Random Forest

Hyperparameter	Value
Criterion	MSE
Max depth	5
Min samples split	3
Min samples leaf	3
Max features	log2

Model: Gradient Boosted Trees

Hyperparameter	Value
Loss	LAD
Criterion	Friedman MSE
Max depth	10
Max features	log2

Model: MLP

Hyperparameter	Value
Epochs	10
Batch size	64
Dense activation	ReLU
Output activation	linear
Dense layers	2
Dense nodes (l1)	128
Dense nodes (l2)	64
Dropout	0.11
Optimizer	Nadam
Learning rate	0.001
Loss function	huber

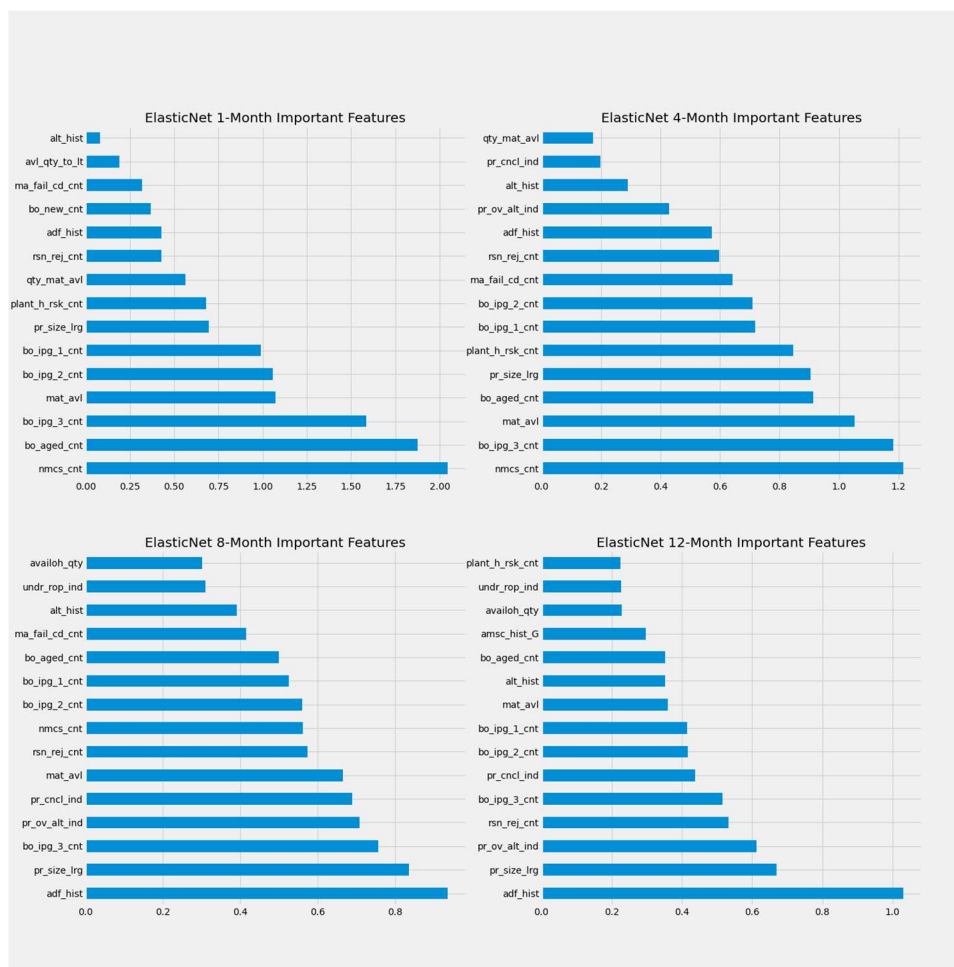
Model: Encoder-Decoder

Hyperparameter	Value
Epochs	15
Batch size	256
Dense activation	ReLU
Output activation	linear
Latent dimensions	512
Dense layers	3
Dense nodes (l1)	128
Dense nodes (l2)	64
Dense nodes (l3)	32
Dropout	.22
Optimizer	Nadam
Learning rate	0.001
Loss function	huber

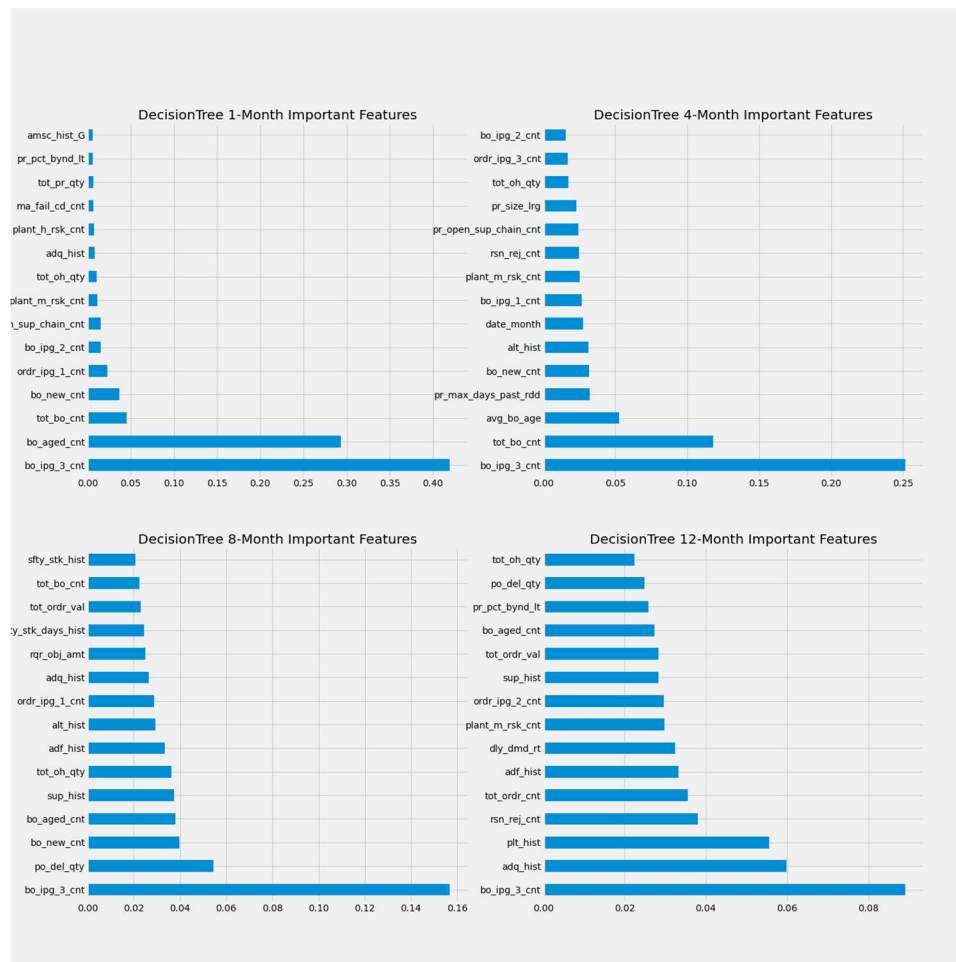
Appendix C. Feature Importance Plots

The following plots provide the top feature importances for each of the model architectures trained and evaluated in this effort. The scikit-learn models are displayed at four month intervals in separate plots using scikit's built-in feature importances/coefficient attribute for each model. The TensorFlow model importances are derived from the mean SHAP values for each of the 12 month forecast periods. Currently, multi-input recurrent neural networks (encoder-decoder model) built using TensorFlow 2.0 are not supported in SHAP so only importances from the multilayer perceptron model are able to be provided.

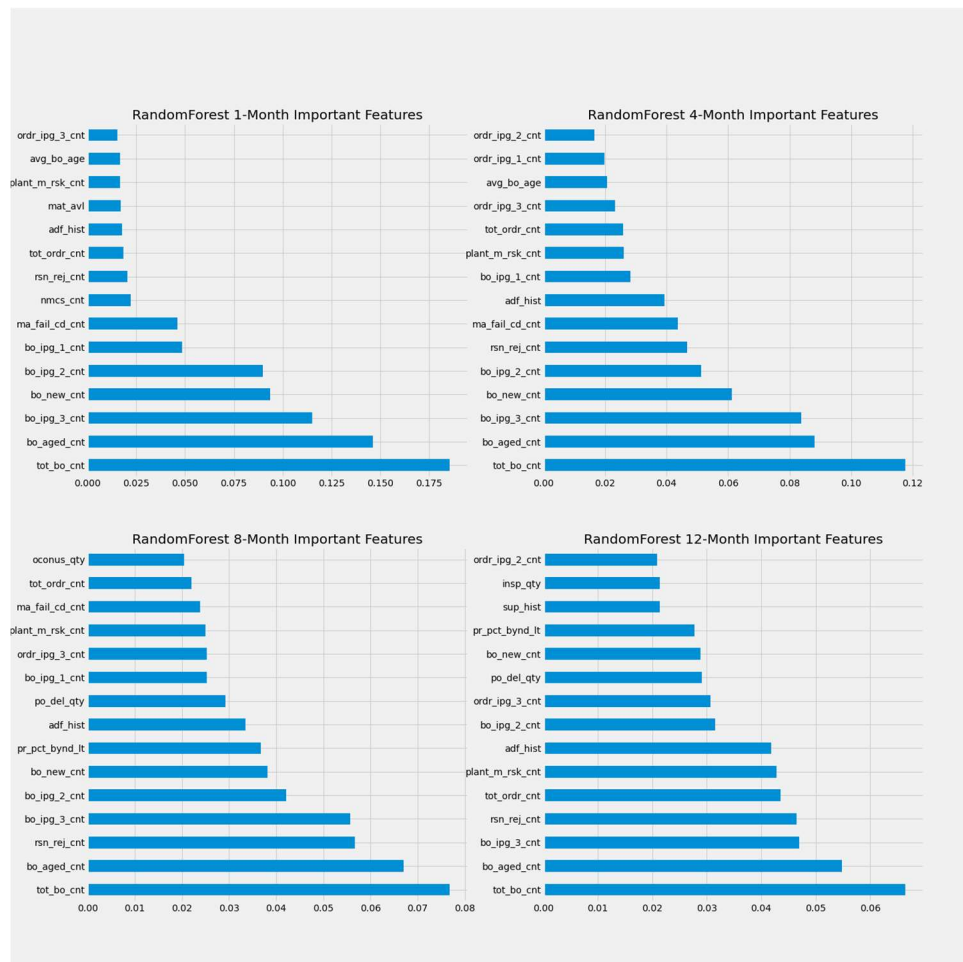
Model: Elastic Net



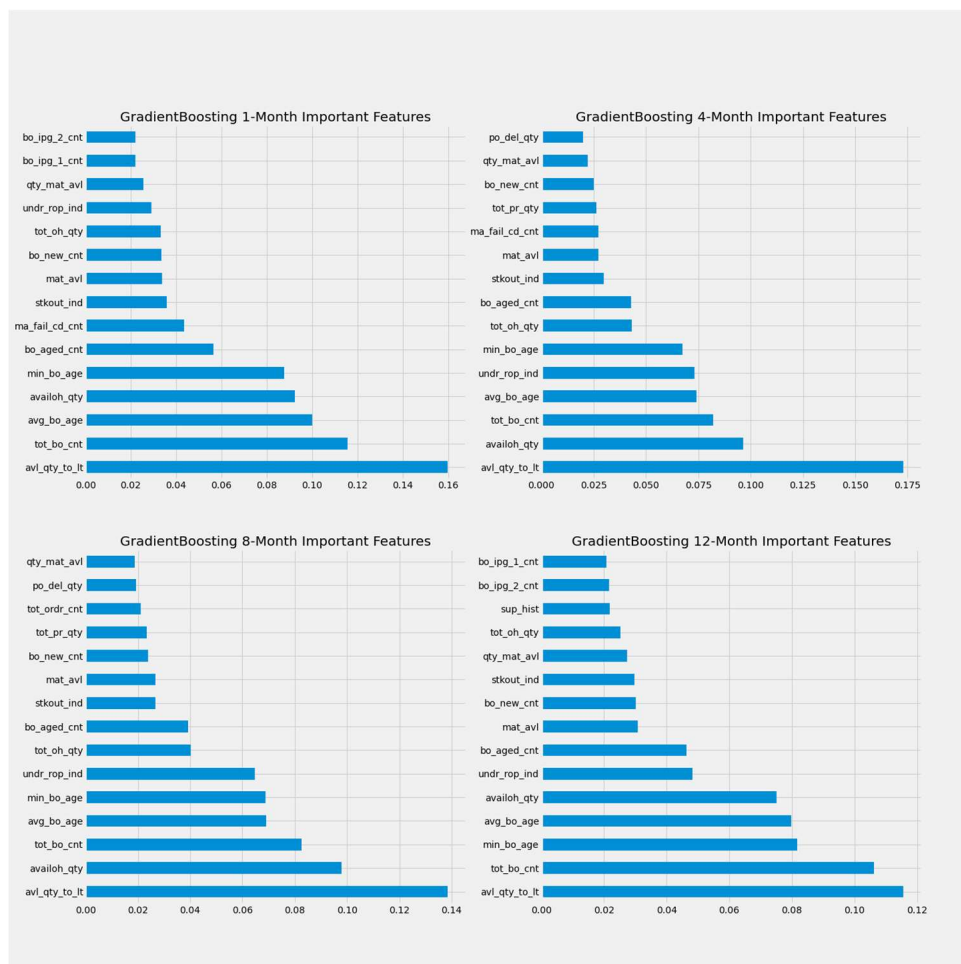
Model: Decision Tree



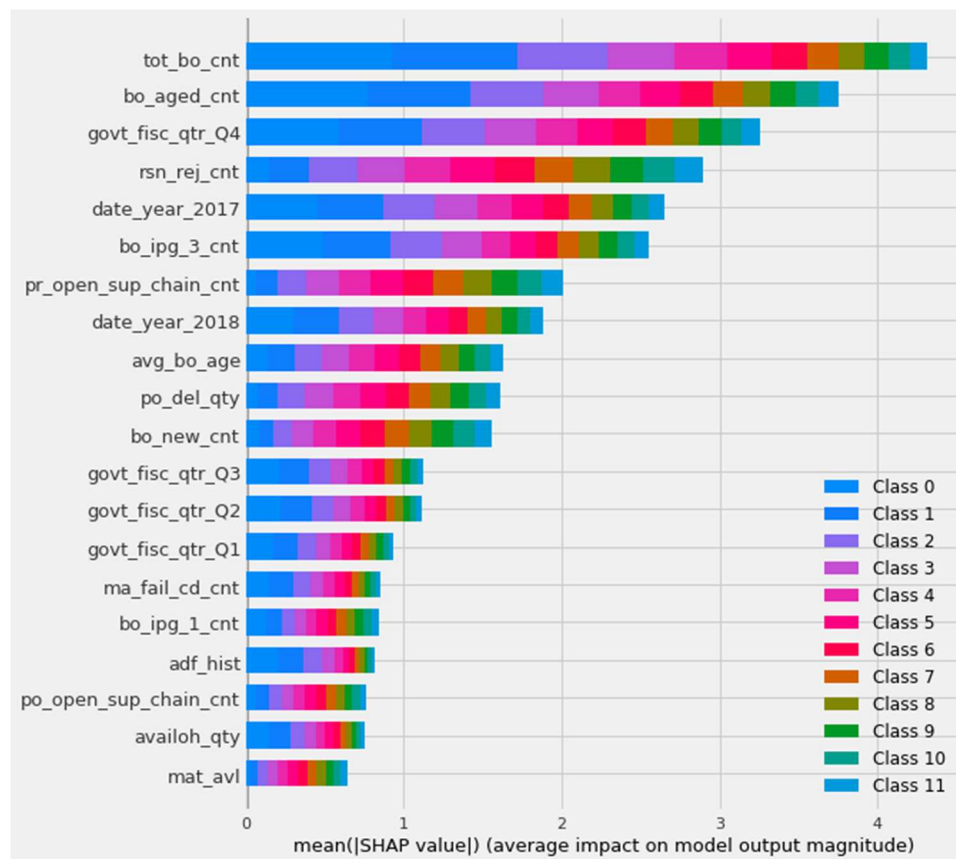
Model: Random Forest



Model: Gradient Boosted Trees



Model: Multilayer Perceptron



Appendix D. Feature Importance Plots for Backorder Scenarios

The following plots provide the top feature importances for each of the 9 multilayer perceptron models trained and evaluated on the paired backorder scenarios consisting of NIIN populations that were subset based on the patterns within the historical and future 12-month sequences that occurred in the last 24 months of the data.

