Demand Forecasting for Commercial Truck Assembly: A Data-Driven Approach

Chris Cooke, Brian Enns, Martin Guzman, Kuljinder Kaur, Sigao Li

Objective

Our mission is to devise a tailored demand forecasting model for a leading global commercial truck and bus manufacturer, taking into account the unique challenges their job shop assembly faces with inventory imbalances and efficiency. Drawing from historical purchase schedule demand data, forecast percentages from production build schedule, and item master data, we aspire to anticipate future demand patterns for key items. In parallel, our model will evaluate potential stockout risks. Based on these insights, we will propose a comprehensive strategy for proactive inventory management, ensuring optimized costs and heightened operational effectiveness.

Data Preparation

In accordance with the assignment 'real data' specifications, datasets were sourced from digital replicas of the company's ERP system tables, which are stored on the company's Hadoop data lake. This data was accessed via SQL queries and subsequently exported in .csv format. We have secured the necessary approvals from the company to utilize this data, under the stipulation that supplier identities are kept confidential and concealed.

At the outset of our analysis, three datasets were selected:

1. 'tbl_znmps3400': Primarily documenting forecast data, this dataset focuses on truck models, elucidating their production rates and associated timestamps.
2. 'ldos_archive': Serving as a comprehensive archive, this dataset reveals item master attributes. Noteworthy fields include volume calculations, part descriptors, and key inventory statistics.
3. 'tbl_nav830_hist': An articulation of the EDI830 Purchase Schedule as relayed to the supply base in weekly Electronic Data Interchange standard. It encompasses a wide array of information like specific part numbers, contractual details, and essential timestamps.

Prior to exporting from the proprietary environment, we undertook standard data cleaning measures. Within the Hadoop system, all data types were in string formats. To ensure consistency, we employed the trim() function to eliminate both leading and trailing spaces. However, we encountered challenges with date and timestamp formats. A mix of formats such as yyyy-MM-dd hh:mm:ss, yyyy-MM-ss, and international variations like dd-MM-yyyy, MM-dd-yyyy, and dd/MM/yyyy were observed. To maintain data integrity and streamline our analyses, all these different formats were normalized and cleaned to adhere to the ISO 8601 'yyyy-MM-dd' standard.

Scope Refinement:

Early in our analysis, a conundrum surfaced regarding the inclusion of tbl_znmps3400. The core challenge stemmed from the dataset's inability to precisely map items to individual Truck Models. This became pronounced when acknowledging that several items permeated multiple model variants. Owing to these complexities, we had to pragmatically exclude tbl_znmps3400 from our analytical scope.

Data Quality:

Date Standardization:

Given the imperative for temporal accuracy, timestamps across ldos_archive and tbl_nav830_hist underwent standardization. By transforming these dates to echo the format 'yyyy-MM-dd', corresponding to the proximate Sunday, we fostered a foundation of uniformity crucial for subsequent analyses.

Unique Data Verification:

Post-merging, validating the unique nature of our 'updated' column emerged as paramount. An exhaustive verification confirmed its uniqueness across prescribed groupings.

Key Insight: Our 'updated' column consistently retained its unique identity across all group combinations, underscoring the success of our data merge and cleaning operations.

Data Procurement and Processing:

Merging and Cleaning:

A crucial phase involved the merging of ldos_archive and tbl_nav830_hist, facilitated via shared columns, notably item, company, and updated. Despite the smooth merger, potential data redundancies surfaced, necessitating a meticulous cleaning process. The culmination of this effort was the genesis of a cleansed merged_df.

Key Insight: The evolved merged_df showcased a significant 913,952 rows spanning 60 columns, a testament to its exhaustive scope.

Cohort Construction:

To streamline analysis, we crafted a unique identifier, termed 'cohort_id', for every record housed within merged_df. This was realized through the concatenation of several pivotal columns.

Key Insight: This endeavor yielded a staggering 40,348 unique cohorts, spotlighting the dataset's inherent diversity. Furthermore, ‘update’ with ‘cohort_id’ suggests this data is 40,348 distinct time series with a mean of 22.65 distinct cohort_id – updated groupings.

Model Design

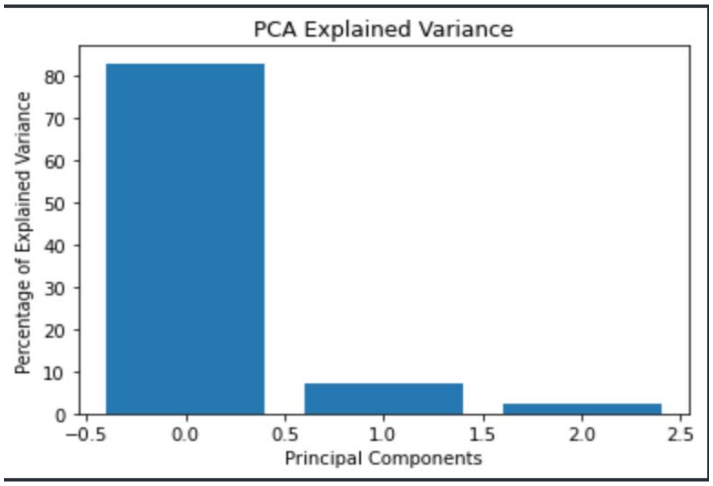
Unsupervised Machine Learning Treatment

Our project being a real-life live problem with real data, we started out by making an exhaustive attempt to understand as much about the problem as possible, in order to determine the right algorithm that would make sense in our case. While we did know that we wanted to forecast demand basis some historical data (hence making it a true supervised machine learning problem), given the kind of data and the complexities and intricacies amongst various variables, we started by exploring an unsupervised algorithm and attempted to do Clustering using k-means post doing principal components analysis (PCA) to see if there could be a smaller set of features that we can proceed with regression later.

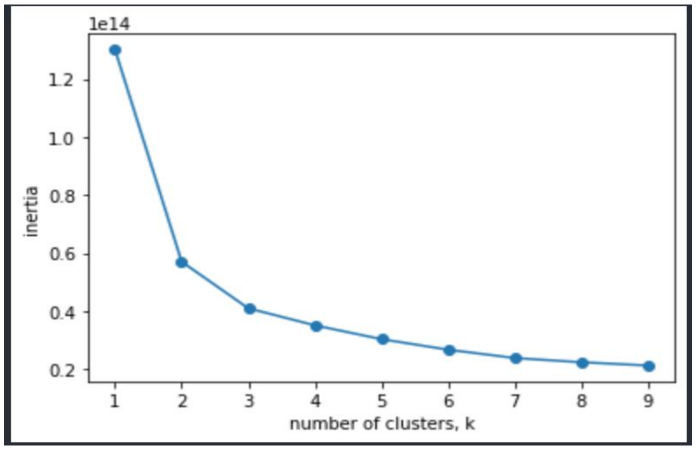
Dimensionality Reduction using PCA and Clustering using k-Means

Principal Component Analysis (PCA) is being utilized in this study as a part of the exploratory data analysis phase. The reason for incorporating PCA lies in its ability to reduce the dimensionality of the dataset while retaining most of the original variance. With a high-dimensional dataset, understanding the relationships between variables can be challenging due to the "curse of dimensionality." PCA helps us to overcome this problem by transforming the data into a new coordinate system where the first few principal components capture the majority of the information contained in the original dataset.

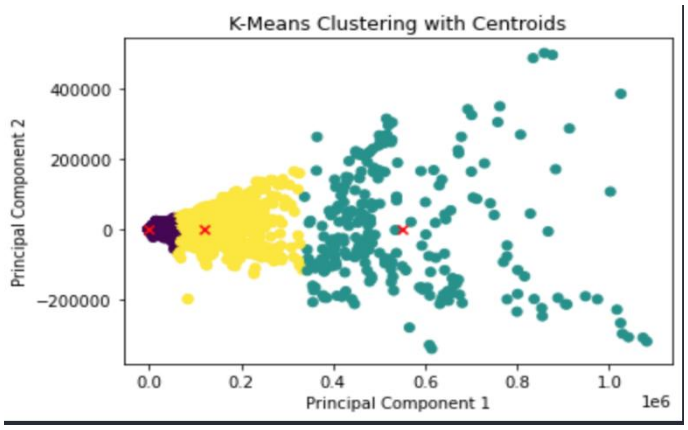
By focusing on these principal components, we are able to visualize and interpret the underlying structure of the data more effectively. This visualization helps in uncovering patterns, trends, or anomalies that might not be apparent when considering the original high-dimensional data. Additionally, by reducing the dimensionality, PCA assists in alleviating multicollinearity and overfitting issues, thereby paving the way for more robust modeling in subsequent analysis stages.



Following the dimensionality reduction achieved through Principal Component Analysis (PCA), the data was further analyzed using the KMeans clustering algorithm. The aim of this stage was to segment the dataset into coherent groups or clusters, where observations within the same cluster are similar to each other and dissimilar to those in other clusters.

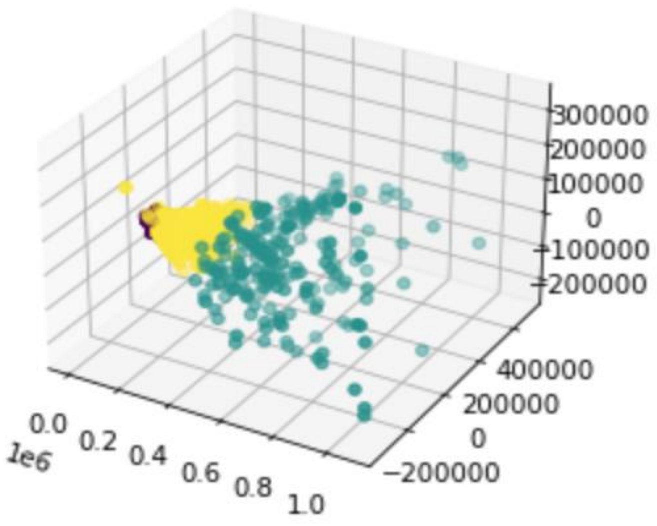


To determine the optimal number of clusters, an inertia test was conducted. Inertia, or the within-cluster sum of squares. A plot of inertia against different numbers of clusters typically exhibits an "elbow" where the reduction in inertia begins to slow down. In this specific analysis, the elbow method indicated that three clusters were the optimal choice, balancing the trade-off between minimizing inertia and avoiding over-segmentation.



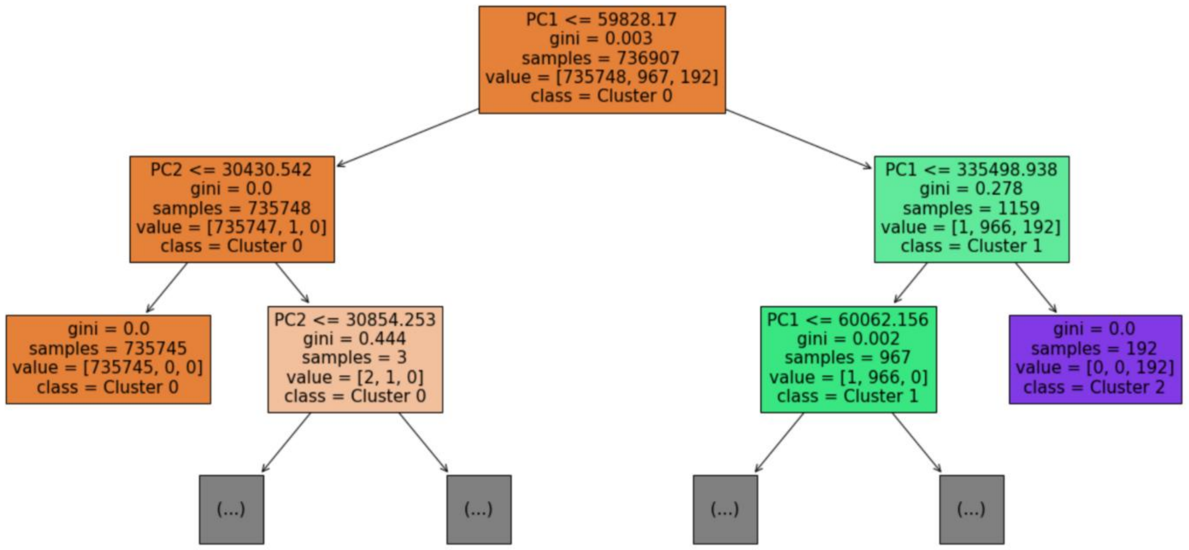
By segmenting the data into three clusters, we were able to uncover distinct patterns and relationships that were not apparent in the original high-dimensional space. These clusters represent different segments or subgroups within the data and may correspond to underlying phenomena or characteristics that are of interest to the study.

The combination of PCA and KMeans provided a two-step approach to understanding the dataset. First, PCA reduced the dimensionality and captured the essential structure of the data. Then, KMeans clustering grouped the data into meaningful clusters, providing further insights into the relationships and natural segmentation within the observations. This methodology facilitated a more nuanced and interpretable understanding of the complex dataset, forming a solid foundation for further analysis and interpretation. It also provided a clear illustration of potential scaling issues with the data strongly suggesting that different cohort_id groups be treated independently.



Building on the insights gained through PCA and KMeans clustering, the identified clusters were subsequently used as labels for the dataset. This transformed the problem into a supervised learning task, allowing for the application of a Decision Tree Classifier. The objective of this phase was to elucidate the feature importance and to understand which variables are most influential in defining the cluster membership.

A Decision Tree Classifier was chosen for its interpretability and ability to reveal the hierarchical relationships between features. By constructing a tree-like model that splits the data based on feature values, the classifier was able to identify the key variables that were most predictive of the cluster labels. This provided a clear visual representation of how different features interact and contribute to the



segmentation discovered during the KMeans clustering.

The analysis of feature importance extracted from the decision tree further deepened the understanding of the dataset. By ranking the features based on their contribution to the classification, the model helped in uncovering the underlying mechanisms that differentiate the clusters. These findings not only validated the clustering process but also provided actionable insights that can guide future research, interventions, or business decisions related to the phenomena represented by the clusters.

In summary, the combination of PCA, KMeans clustering, and Decision Tree Classification formed a cohesive and comprehensive analytical pathway. Starting with dimensionality reduction, followed by unsupervised clustering, and culminating in a supervised classification to explain feature importance, this multi-stage analysis has illuminated complex relationships within the dataset. The resultant insights into cluster differentiation and feature significance present a nuanced understanding of the data and pave the way for targeted strategies and decision-making within the context of the study.

Supervised Machine Learning Treatment (Dependent variable = ‘wk02’)

Model: ‘Naive Model’ – Predict ‘wk02’ using ‘wk03’ from previous period

Seeking a step-wise approach to achieving our objective, we decided to start with a “naïve” model inspired by our initial discussions with the dataset. Without having the need to do feature scaling or build in any sophistication basis domain knowledge for this basic formulation, but with the belief that past patterns hold predictive power, we derived the 'wk03_s01' column by simply shifting the 'wk03' data by a single interval (or lag) to factor in the temporal component to be used as a single predictor of ‘wk02’. For comparison, we created two additional models (Simple Moving Average (SMA) and Exponential Moving Average (EMA)) leveraging the values 'wk03_s01', 'wk04_s02', 'wk05_s03' and 'wk06_s04'. Intuitively, this offers insights into how previous week's data might influence current week's outcomes. The below table shows our results, with exact matches for further insight into data behaviour.

Metrics for wk02_pred_last	**Metrics for wk02_pred_sma**	**Metrics for wk02_pred_ema**
RMSE: 811.1904719962276 R^2: 0.18441854388984913 Exact Matches for wk02_pred_last: 621565 (71.15%) Matches for wk02_pred_last: Within 5%: 718097 (82.20%) Within 10%: 746013 (85.39%) Within 20%: 779478 (89.23%)	RMSE: 912.1125721411872 R^2: -0.031142161158868698 Exact Matches for wk02_pred_sma: 394583 (45.17%) Matches for wk02_pred_sma: Within 5%: 601320 (68.83%) Within 10%: 637906 (73.02%) Within 20%: 691686 (79.18%)	RMSE: 830.4178481519167 R^2: 0.14529742854000127 Exact Matches for wk02_pred_ema: 387688 (44.38%) Matches for wk02_pred_ema: Within 5%: 611531 (70.00%) Within 10%: 656065 (75.10%) Within 20%: 718703 (82.27%)

Machine Learning Feature Engineering and Model Exploration

Given the nature of our data and the requirements of the problem, we explored various forecasting algorithms, with a consideration to the data challenges we uncovered in the data exploration and processing steps. We targeted predictive modelling for all ‘cohort_id’ considered together as opposed to by individual time series. Data scaling also proved to be a key challenge. In an attempt to address these challenges we implemented a custom Cross Validation object and a custom RowAbsMaxScaler object in Python. Lagged features were necessary as time series prediction is a required objective.

Training, Testing and Cross Validation

With data being a form of “grouped” timeseries, we looked to split our data by ‘updated’ date quantile, but not grouping by ‘cohort_id’ at this stage. Training was reserved as the first 80% of the data quantiles, and 20% reserved for testing. Additionally, the custom Cross Validation object split the data into n-folds (we used 5) based on quartiles. Note: due to the nature of the data, too tight of a date quantile split would cause Python errors (suggesting imbalanced ‘cohort_id’ – ‘updated’ groupings), so this would be a point of future scrutiny. A key challenge was working with the skLearn libraries assumptions about Pandas library indexing to ensure that independent and dependent variables match up throughout the cross validation process.

Custom Data Scaling

We considered various options for feature scaling. Initial exploratory model runs were not providing satisfactory results, so we explored “by-row normalization after lags using maximum value found in quantity-related fields”. The anticipation would be that this would normalize data for like-to-like comparison across cohort looking for predictive model within this context. Note: models including this scaling would not be easily interpretable, but could be analysed to assess predictive features. Furthermore, since ‘wk02’ is a quantity value, scaling was applied to the dependent variable to match with normalization applied to various other qty values.

Lagged Features for Time Series

The dataset consists of a projected scheduled production schedule consisting of fields from wk01-wk12 (and other monthly features) likely to hold predictive value (see Naïve Model above). Grouping by ‘cohort_id’ was essential for proper shifting. To predict wk02 from a past period, not current period information, we shifted wk02 from a future period back to the current period. Note: shifting introduces NaN values when there is no corresponding. These were dropped reducing total rows from 913952 to 767139.

Key Features

Potential Predictive Indicators:
As discussed above, quantity-related fields may hold predictive value.

Pause Indicator - 'wk02_s01':
By constructing 'wk02_s01', we capture potential predictive signals from pauses or halts in the process. This shift-based approach is envisaged to highlight any discernable patterns during such halts, especially when juxtaposed with other features.

Supplier Planning - 'wk05_08_avg':
Anticipating that suppliers often resort to forecasting for monthly planning, we generated an average field 'wk05_08_avg' covering the span of four weeks. It's pivotal to note that while we currently operate on the assumption that suppliers' decision-making is weekly yet stretched across an entire month, any alterations in this premise might necessitate realignment.

Supplier Forecast Alterations:
Deviations in supplier forecasts can be revelatory. To that end, we devised 'wk05_08_diff' to illuminate changes in the supplier forecast. Furthermore, to provide a time-shifted context, 'wk05_08_diff_s03' and 'wk05_08_diff_s02' were constructed, which essentially are 3 and 2-week shifted versions of our difference data, respectively.

Pre-Processing and Pipelines

We aligned our workflow for pipelines, but opted to preprocess the data instead to optimize timing for multiple model exploration (ie pre-processing not run multiple times). A number of categorical fields were included, on which we OneHotEncoded, as well as the RowAbsMaxScaler() applied to quantity values. Note: for this exercise, we chose to limit categorical variables to those with 15 or fewer categories to avoid a combinatorial explosion of features.

Model Exploration and Evaluation

Cross Validated Root Mean Squared Error (RMSE) and R2 Score used for model exploration, with RMSE and R2 Score for validation and comparison with the Naïve Model. We opted to explore a sample of models using a number with default hyperparameters as a starting point. These models included: BayesianRidge, RandomForest, Ridge, Decision Tree, Extra Trees, GradientBoosting, MLP, AdaBoost, kNN, Dummy(mean value), Lasso and ElasticNet. SVR was excluded as with our dataset size, model fit speed was an observed concern. Ensemble models were included in the review. In an attempt to improve training performance, parallel functions were leveraged from the joblib Python library for those models investigated. A custom scorer to return both RSME and R2 Score was used, as well as the SKLearn cross_validate() function with our custom cross validation object. Top models for Cross Validated scores were considered for further hyperparameter tuning and evaluation.

Model: Predict non-shifted 'wk02' (Current, Row Normalized Data)

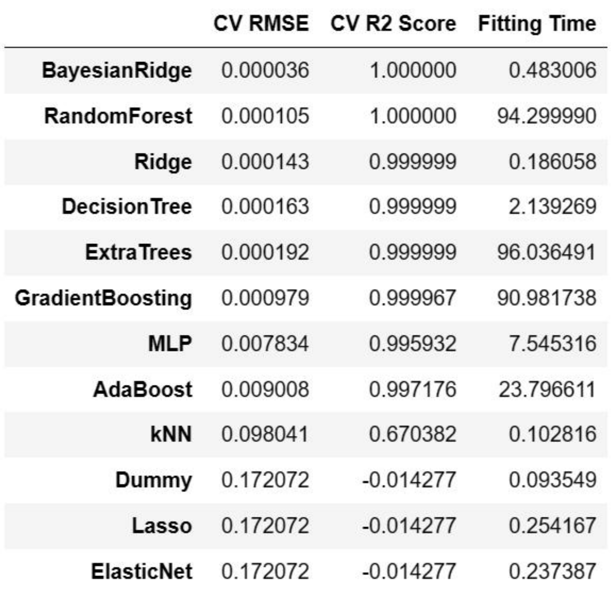
Quantity values normalized with our RowAbsMaxScaler(), attempting to predict current "wk02" using current independent features, plus lagged features (see above).

Results

The results appeared highly predictive, with the majority of models performing very well. Given that the dependent variable was in the same time period as non-shifted features, this may be suggestive of data leakage, or just a high correlation between current period values. We pivoted to predicting a future 'wk02' from current and lagged values, although analysing the key predictive features could provide further insight into various important correlations. Not evaluated on test data.

Evaluation

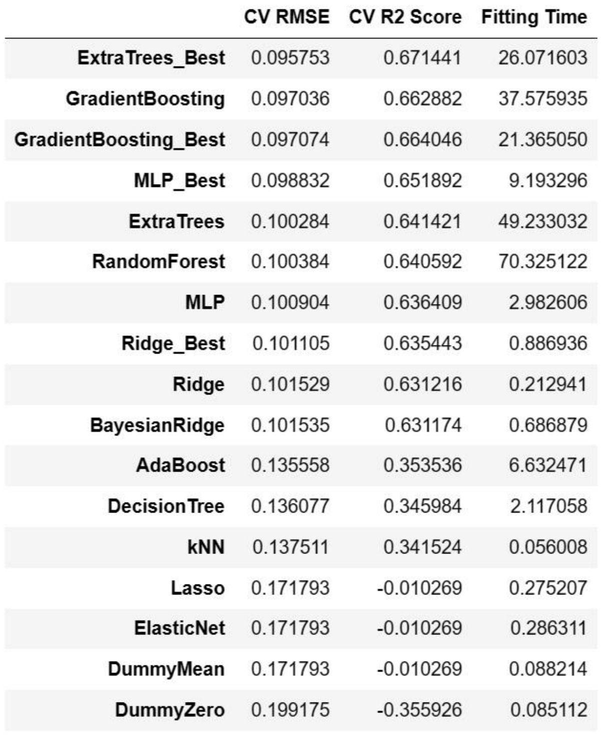
None



	CV RMSE	CV R2 Score	Fitting Time
BayesianRidge	0.000036	1.000000	0.483006
RandomForest	0.000105	1.000000	94.299990
Ridge	0.000143	0.999999	0.186058
DecisionTree	0.000163	0.999999	2.139269
ExtraTrees	0.000192	0.999999	96.036491
GradientBoosting	0.000979	0.999967	90.981738
MLP	0.007834	0.995932	7.545316
AdaBoost	0.009008	0.997176	23.796611
kNN	0.098041	0.670382	0.102816
Dummy	0.172072	-0.014277	0.093549
Lasso	0.172072	-0.014277	0.254167
ElasticNet	0.172072	-0.014277	0.237387

Model: Predict shifted ‘wk02’ (Future, Row Normalized Data)

Similar models to above, but switched to predict a future “wk02”. We executed cross_validation() on default parameters to assess initial performance, then selected the top four models (sorted by “CV RSME”) for hyperparameter tuning using RandomizedSearchCV(), implementing our custom cross validation object.



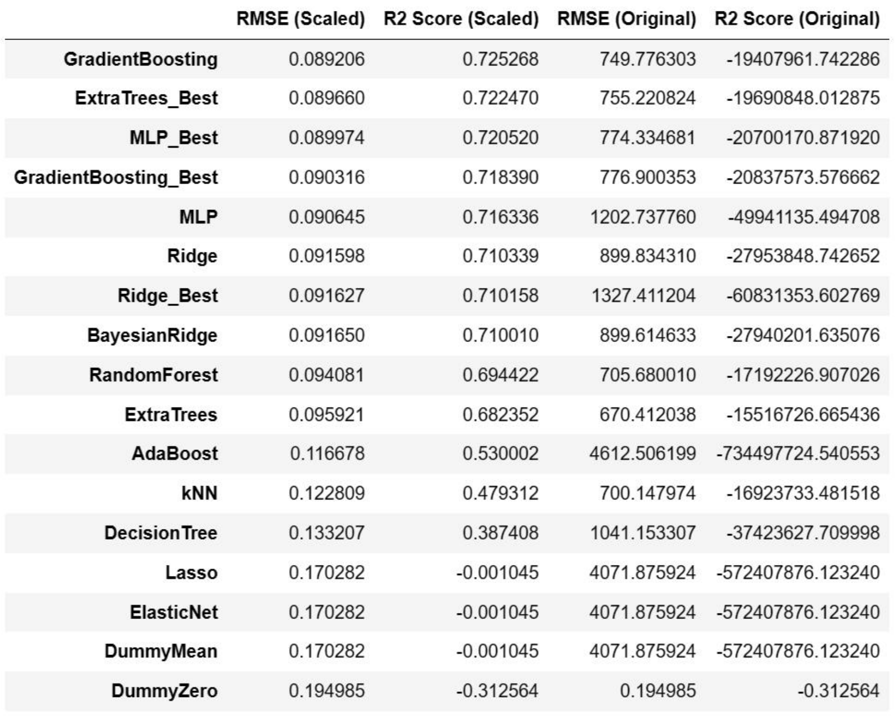
	CV RMSE	CV R2 Score	Fitting Time
ExtraTrees_Best	0.095753	0.671441	26.071603
GradientBoosting	0.097036	0.662882	37.575935
GradientBoosting_Best	0.097074	0.664046	21.365050
MLP_Best	0.098832	0.651892	9.193296
ExtraTrees	0.100284	0.641421	49.233032
RandomForest	0.100384	0.640592	70.325122
MLP	0.100904	0.636409	2.982606
Ridge_Best	0.101105	0.635443	0.886936
Ridge	0.101529	0.631216	0.212941
BayesianRidge	0.101535	0.631174	0.686879
AdaBoost	0.135558	0.353536	6.632471
DecisionTree	0.136077	0.345984	2.117058
kNN	0.137511	0.341524	0.056008
Lasso	0.171793	-0.010269	0.275207
ElasticNet	0.171793	-0.010269	0.286311
DummyMean	0.171793	-0.010269	0.088214
DummyZero	0.199175	-0.355926	0.085112

Results

The results were much more down to Earth (see right). The “_Best” models were top four from our initial default parameter review displayed with results after hyperparameter tuning. The tree-based, GradientBoosting, Multi-Layer Perceptron Regressor (MLP) performed the best, suggesting there are non-linearities and complex data relationships within the dataset. Unfortunately, the data in this format is not directly comparable to our Naïve Model. Computation limitations inhibited further hyperparameter tuning.

Evaluation

Test data was held back for review (see above). Fitting on test data seems to have improved scores, but that is likely attributed to the models in the list on the right being trained on the full 0.8 quantile training data (as opposed to 5-fold cross-validation). Note that inverse scaling was applied to the predicted / actual values in an attempt to get a score comparable to the Naïve model. Note: RMSE does appear comparable to the Naïve model, but the R2 Score may be negatively affected by the inverse scaling attempt on the predicted values.



	RMSE (Scaled)	R2 Score (Scaled)	RMSE (Original)	R2 Score (Original)
GradientBoosting	0.089206	0.725268	749.776303	-19407961.742286
ExtraTrees_Best	0.089660	0.722470	755.220824	-19690848.012875
MLP_Best	0.089974	0.720520	774.334681	-20700170.871920
GradientBoosting_Best	0.090316	0.718390	776.900353	-20837573.576662
MLP	0.090645	0.716336	1202.737760	-49941135.494708
Ridge	0.091598	0.710339	899.834310	-27953848.742652
Ridge_Best	0.091627	0.710158	1327.411204	-60831353.602769
BayesianRidge	0.091650	0.710010	899.614633	-27940201.635076
RandomForest	0.094081	0.694422	705.680010	-17192226.907026
ExtraTrees	0.095921	0.682352	670.412038	-15516726.665436
AdaBoost	0.116678	0.530002	4612.506199	-734497724.540553
kNN	0.122809	0.479312	700.147974	-16923733.481518
DecisionTree	0.133207	0.387408	1041.153307	-37423627.709998
Lasso	0.170282	-0.001045	4071.875924	-572407876.123240
ElasticNet	0.170282	-0.001045	4071.875924	-572407876.123240
DummyMean	0.170282	-0.001045	4071.875924	-572407876.123240
DummyZero	0.194985	-0.312564	0.194985	-0.312564

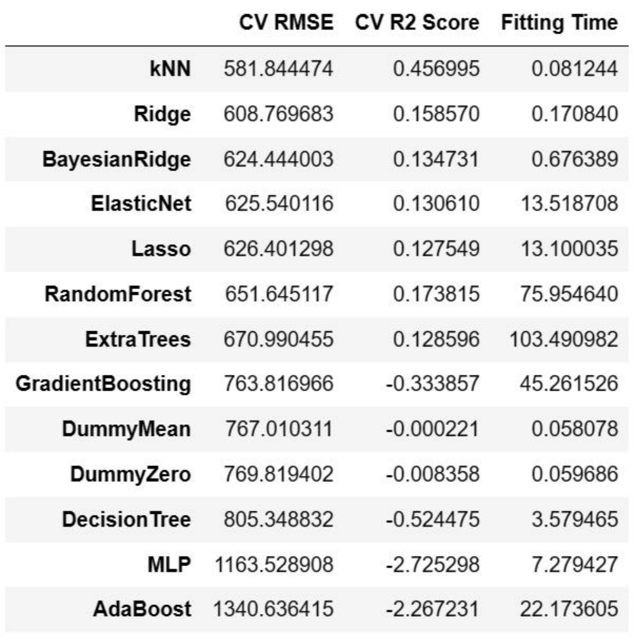
Model: Predict shifted ‘wk02’ (Future, Non-Normalized Data)

Similar models to the above, same protocols, but normalizing quantity fields *not* performed.

Results

With the non-normalize quantity data, the top-performing default hyperparameter models shifted to kNN and Ridge and Lasso-based models (linear assumptions with regularization) from previous normalized explorations. This suggests that regularization of the non-normalized data is a key for model performance, and hint at clustering cohort_id into “like” quantity levels for better performance. Note that in the majority of the cases, RMSE score is comparable or better than the Naïve Model with kNN beating it by a large margin.

Hyperparameter-tuned results not available for publication



	CV RMSE	CV R2 Score	Fitting Time
kNN	581.844474	0.456995	0.081244
Ridge	608.769683	0.158570	0.170840
BayesianRidge	624.444003	0.134731	0.676389
ElasticNet	625.540116	0.130610	13.518708
Lasso	626.401298	0.127549	13.100035
RandomForest	651.645117	0.173815	75.954640
ExtraTrees	670.990455	0.128596	103.490982
GradientBoosting	763.816966	-0.333857	45.261526
DummyMean	767.010311	-0.000221	0.058078
DummyZero	769.819402	-0.008358	0.059686
DecisionTree	805.348832	-0.524475	3.579465
MLP	1163.528908	-2.725298	7.279427
AdaBoost	1340.636415	-2.267231	22.173605

Evaluation

Evaluation results not available for publication

Model Evaluation

Based on the results for future-predicted ‘wk02’ values, we have two approaches related to normalized data vs non-normalized data. The GradientBoostRegressor for “**Model: Predict shifted ‘wk02’ (Future, Row Normalized Data)**” may hold information that can be useful for further analysis of predictive factors for ‘wk02’, but would not be useful for direct prediction (although it did have an RSME marginally better than the **Naïve Model**). Preliminary results from “**Model: Predict shifted ‘wk02’ (Future, Non-Normalized Data)**” suggest that kNN with default values may be best for direct prediction, with score results trending better than the **Naïve Model** (final results pending and are not available for publication).

With the current structured model evaluation framework in place, additional model review and more thorough hyper-parameter tuning can be investigated and reviewed as compute resource dictate.

Conclusions

We have taken a critical first step in anticipating future demand patterns for key items and evaluating potential stockout risks. Our hypothesis that historical requirements data, combined with BOM configurations and external factors, can predict demand has been supported. Insights gathered also highlighted potential areas of stockouts, suggesting areas for proactive inventory management. The kNN model on non-normalized data presents a better model for predicting the future value of ‘wk02’ than a Naïve Model (and other options explored), and which can only be improved through an in-depth and focused exploration of individual item cohorts (possibly ARIMA and autocorrelation timeseries analysis), better quantity normalization, and dimensionality reduction. Additionally, the analysis and modeling pipeline can be quickly adapted for additional review and prediction of further elements in the schedule such as ‘wk04’, ‘wk05’ and so on.

Going forward, we recommend continuous validation of the model with newer data, exploring other external factors that might influence demand, and considering the inclusion of real-time data streams to enhance forecast accuracy.

References

Scikit-learn: Machine Learning in Python. (n.d.). Retrieved from https://scikit-learn.org/stable/

pandas: powerful Python data analysis toolkit. (n.d.). Retrieved from https://pandas.pydata.org/

Joblib: A set of tools to provide lightweight pipelining in Python. (n.d.). Retrieved from https://joblib.readthedocs.io/