Modeling the Demand Trend for Automobile Parts using Machine Learning Methods

Oguzhan Akan

Department of Computer Science,
California State University,
Northridge, CA, USA
E-mail: oguzhan.akan.95@my.csun.edu

Abhishek Verma*

Department of Computer Science,
California State University,
Northridge, CA, USA
E-mail: abhishek.verma@csun.edu
* corresponding author

Sonika Sharma

Department of Commerce,
Shaheed Bhagat Singh College,
University of Delhi, Delhi, India
E-mail: sonika.sharma@sbs.du.ac.in

Abstract: This research utilizes the automotive spare parts sales dataset, containing 5,000 spare parts. Being able to accurately model demand trend for spare parts helps in reducing warehousing costs, production costs, and could result in higher sales via fulfilling sales orders in a timely manner. In order to model the demand for automobile spare parts we build machine learning models using K-Nearest Neighbor, Support Vector Regressor, Random Forest, Gradient Boosting, Back Propagation Neural Networks, and Deep Neural Networks. We conclude that Deep Neural Networks performs better than the rest of the machine learning models on the spare parts dataset for demand trend modeling with a RMSE validation score of 6.25. After a thorough literature review, it is observed that there are few previous researches done on Gradient Boosting Regression for demand trend modeling. Our research proves that Gradient Boosting's performance is better than other traditional machine learning models such as K nearest neighbor, Random Forest, and Support Vector Regressor.

Keywords: Demand Modeling; Gradient Boosting; K-Nearest Neighbors; Random Forest; Deep Neural Networks; Support Vector Machines

Reference to this paper should be made as follows: Akan, O., Verma, A., and Sharma, S. (202X) 'Modeling the Demand Trend for Automobile Parts using Machine Learning Methods,' *International Journal of Artificial Intelligence and Soft Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Oguzhan Akan received his Master of Science in Computer Science degree from California State University, Northridge. His research interests are in machine learning, deep learning, and big data analytics.

Abhishek Verma received his Ph.D. in Computer Science from New Jersey Institute of Technology, NJ, USA. His research interests are in data science, big data analytics, machine learning, and deep learning on big datasets.

Sonika Sharma received her Ph. D. in finance from University of Delhi, Delhi, India. She is Associate Professor, Department of Commerce, Shaheed Bhagat Singh College, University of Delhi, Delhi, India. Her research interests are in finance, insurance, and business analytics.

1 Introduction

Automobile companies have various spare part inventory in their warehouse to cater toward customer needs. The range of spare parts allows the automobile companies to have modularity in their vehicles and makes it easier to perform repair and maintenance. However, it is difficult to predict when, what parts, and in what quantity the customers would need. In order to address this issue, demand trend modeling using machine learning helps the automotive industry to recognize patterns and predict future sales. This allows their warehouses to be filled with parts customers will most likely need, it helps in reducing the cost of part storage and increase production in factories of parts that are highly requested. This results in lower production costs, higher sales, better management of the inventory, fulfillment of the part orders in a timely manner, higher profitability for the business, and overall improved customer satisfaction.

In order to prevent shortage of parts and to reduce wasted resources, existing works have used a wide range of methods to improve the performance of parts demand forecasting models. Also, common challenges for demand trend modeling of spare parts are the intermittent data and stochastic characteristics Amin-Naseri & Rostami Tabar (2008); RuiRui Xing, Xianliang Shi, Member (2019); Shenstone & Hyndman (2003); Xu et al. (2012). Due to the challenging nature of parts demand modeling and to improve modeling accuracy we make use of more modern supervised machine learning techniques that includes deep neural networks.

In this paper, we conduct a detailed comparative research on an extensive 3-years of spare part order history data from the automobile industry. The aim is to predict future trends in demand and compare results of multiple machine learning methods that includes K-Nearest Neighbor (KNN), Random Forest (RF), Gradient Boosting (GB), Support Vector Regressor (SVR), Back Propagation Neural Network (BPNNs), and Deep Neural Networks (DNNs). We do not compare our model results with statistical approaches like Croston's method or Markov bootstrapping as the scope of our work is in the area of Machine Learning, and it has been proven that Machine Learning approaches outperform statistical models in the previous studies Shenstone & Hyndman (2003).

We have chosen the Gradient Boosting and Random Forest methods for our research because they are ensemble models and historically seen to perform better than the base estimators. In our research, we found that hybrid models produced more effective results, which led us to conclude that ensemble models would produce more positive results than using a single model as in Hua & Zhang (2006); Qiu et al. (2019); Song et al. (2012). To further verify our expectation, we have selected the K-Nearest Neighbors method as a comparison method. SVR and Neural Networks are other mainstream machine learning methods used in this study. On the other hand, Deep Neural Networks have proven to perform better than traditional machine learning methods in demand trend modeling, we run various DNNs to compare the resulting error rates with other machine learning models.

This paper is organized as follows. Section 2 presents the related work. Next, we provide the descriptions dataset in Section 3. Sections 4 and 5 describe the proposed methodology. Next, the experiment setup and results are presented in Sections 6 and 7. Conclusion and future work are presented in Section 8.

2 Related Work

Hua & Zhang (2006) designed the Logistic Regression and Support Vector Machine Hybrid (LRSVM) to synthetically assess autocorrelation in time series and the relationship between explanatory variables and spare parts demand. It was observed that the LRSVM model outperformed Croston's method, the Markov bootstrapping method, the IFM method, and the SVM method. The hybrid method allowed for the SVM forecast results of nonzero occurrences to be replaced with explanatory variables to formulate a more accurate prediction.

Researchers used a hybrid model using the Grey's Forecasting model (GFM) and Back Propagation Neural Network (BPNN) Hybrid. The GFM was used to produce new data series from the original data with reduced noise to be used for trend modeling. The BPNN then uses the new data series to generate trend analysis results. By combining those two models, the authors were able to produce higher forecasting precision than the models executed separately Song et al. (2012).

In reference Qiu et al. (2019), Qiu et al. proposed a derivation of the Grey's model, the Fractional Order Discrete Grey Model (FOGM) addresses issues such as small sample size and limited information, enabling it to forecast data sequences even less data samples. The genetic algorithm is used to estimate fractional order and generate coefficient for the FOGM to minimize the MAE of the predicted value. With this modification, the researchers are able to produce a model that is closer to the actual value than a single gray model.

In Boukhtouta & Jentsch (2018) the Support Vector Machine (SVM) was used to make predictions of spare parts for the Canadian Armed Forces. In comparison to the ARIMA, Naive, Croston and SES, the researchers concluded that the SVM had a lower average A-MAPE. Additionally, they discovered that the Asymmetric Error, which aims to emphasize the costs associated with over-forecasting and the loss of operational availability resulting from under-forecasting, i.e., unfulfilled part orders, was lower for SVM when applied to intermittent demand series and compared to other machine learning models.

Authors in Wanchoo (2019) compared the Gradient Boosting Model (GB) and Deep Neural Networks (DNN) models in retail for forecasting univariate sales time series. Authors made the observation that either of the two models can be relied on when there is a lack of causal factors or historical data. GB performed better than the DNN and GB may have the tendency to overfit the data while the DNN was vulnerable to vanishing and exploding gradients.

Authors in Zamani et al. (2025) predict lead time of orders from the Kanban system in the lean supply chain scenario. They compare the performance of multiple linear regression,

Table 1 Number of unique values for each descriptive feature in the spare part dataset

Feature	**# of Unique Values**
part_definition_id	1918
part_product_class_id	446
common_part_catalog_id	29
preferred_supplier_id	5
part_family_id	225

KNN, MLP, random forest and conclude that MLP is the best model for predicting orders and logistical regression is effective in lead time prediction. Authors in Lu et al. (2023) predict sales volume of auto parts using a bidirectional gated recurrent unit (Bi-GRU) model. Their results show that Bi-GRU model performs better than other compared models in predicting sales volume. Authors in Shan et al. (2023) use an improved grey model (1, 1) to perform functional transformation of the original time series parts data to increase the data smoothness. Thereafter they use genetic algorithm for predicting sales of auto parts from this transformed data. Authors in Ikizler et al. (2024) explore ARIMA and SARIMA models for predictive analytics of spare part inventory and service demand. They find that there is a strong seasonal pattern to spare part and service demand, the demand peaks in the second and fourth quarters of an year.

3 Description of Dataset and Research Hypothesis

The demand trend modeling dataset used in this research is obtained from Dogus Automotive, a private automobile distributor, from its spare parts demand trend modeling competition in 2020. The company distributes cars of several popular brands including Volkswagen, Seat, Audi, Bentley, and Lamborghini and their spare parts. The dataset consists of two tabular-format tables, which can be joined together using **part ID** column. Here are the details of the tables:

- **catalog**: The catalog table consists of information related to each automobile part. There are 5,000 parts and 5 descriptive features for each part: *part_definition_id, part_product_class_id, common_part_catalog_id, preferred_supplier_id, part_family_id*. Each descriptive feature refers to the ID number sourced in other tables. Therefore, we decided to use the ID numbers in the feature selection phase. In order to decide the approach to follow for each feature, we first look at how many unique values exist in each of them as shown in Table 1. If the number of unique values for a feature is high, then it is suitable for KNN clustering. If it is low (< 30), then an encoding algorithm can be implemented as the space constraint does not pose an issue. Table 2 shows sample data from the catalog table of the parts dataset, each part has : *part_id, part_definition_id, part_product_class_id, common_part_catalog_id, preferred_supplier_id, part_family_id, stock_card_create_date, part_season*.

- **orders**: The orders table consists of information about orders having 206,306 rows of data. An order consists of *part_id, order_quantity, date*, and *firm_id*. We can create features via Time Series Analysis based on the history of each part's orders. Table 3 shows sample data from the orders table of the parts dataset, each order has : *part_id, order_quantity, date, firm_id*. Figure 6 shows total order quantity across all order placed in a pariticular month during 2017-2019.

Table 2 Sample Data from the Catalog Table of the Parts Dataset

part_id	178	234	312	325	712
part_definition_id	4823	4823	3013	8726	10345
part_product_class_id	190	190	360	335	555
common_part_catalog_id	39	39	23	39	39
preferred_supplier_id	1	1	1	1	1
part_family_id	2	2	1	2	2
stock_card_create_date	1/10/2014	4/30/2010	6/20/2016	12/24/2004	12/24/2004
part_season	NULL	NULL	NULL	NULL	NULL

Table 3 Sample Data from the Orders Table of the Parts Dataset

part_id	**order_quantity**	**date**	**firm_id**
10407	3	5/23/2018	53
10945	3	11/28/2017	3
10232	3	2/22/2018	43
10021	3	5/16/2018	121
6124	3	4/18/2018	45
10578	3	1/30/2017	42
468	3	7/5/2017	88
10264	3	5/13/2019	102
10978	18	11/6/2017	84

The research hypothesis in this paper is to evaluate various machine learning methdologies along with data preprocessing and feature selection strategies to be able to correctly model the demand trends for automobile spare parts based on historical data so that the trained machine learning model could be used for prediction of future demand for spare parts based on the training and validation data from the preprocessed parts dataset. The model performance is compared using several different metrics for an in depth analysis of parts demand modeling in the experiment results section of this paper. Another goal of this research is to apply various state-of-the-art preprocessing and feature extraction techniques to process the parts dataset.

4 Preprocessing and Feature Generation Methodology

4.1 Handling Outliers

Before applying bins to the data, a clamp transformation has been applied to remove the outliers. We separate the data into bins while setting the maximum quantity to 72. This is based after observing the frequencies, values greater than 72 can be considered as outliers.

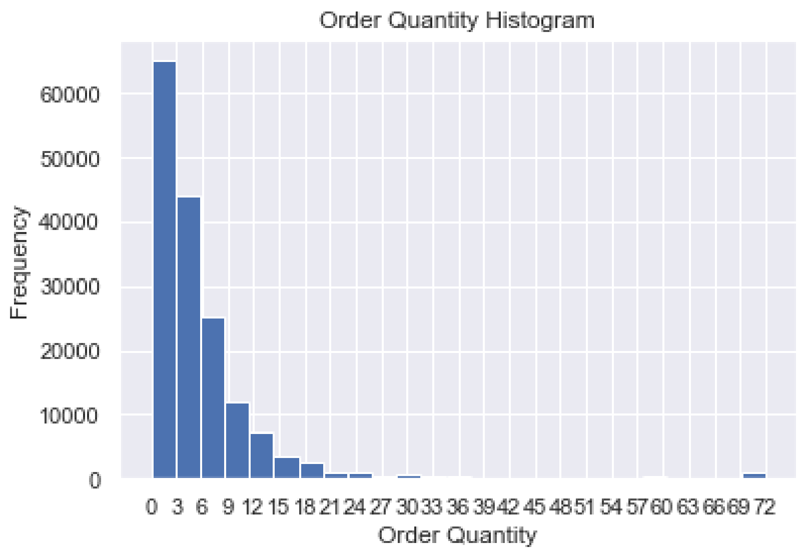

Figure 1: Order quantity bin histogram that shows the skewness in the data.

Since the lowest value of an order quantity can be zero, no transformation is applied to the lower bound. After applying clamp transformation to the target feature, *order_quantity*, it is seen that the target is much skewed to the left (positively skewed) with a skewness value of 4.51 as seen in Figure 1. It is expected that the frequency of smaller orders will be far higher than larger orders. We do not apply any normalization to this feature, it is the target feature we want to predict; thus we do not transform it to any other scale.

4.2 Data Imputation

The *orders* table consists of information about orders having 206,306 rows of data. An order consists of *part_id, order_quantity, date*, and *firm_id*. The *orders* table does not give any information about the days where there was no order placed. We inserted zero orders to the days where there is no order for a specific *part_id*, our conjecture is that it may be an insightful detail when building a prediction model. After this augmentation, the dataset's initial size of 206,306 rows increased to 237,805.

4.3 Handling Multiple Rows Corresponding to the Same Part

The original data has more than one order associated to several parts. The aim of the research is to predict the monthly orders for each *part_id*. Therefore, where there may be more than one order for a particular part in a specific month, we group it by the *part_id* and we compute sum of *order_quantity* for each part. Thus we reduce the 237,805 rows from the previous step to 167,059 rows, such that there is exactly one row corresponding to each part in the training data.

4.4 Generating Time Series Features

Time series analysis involves establishing a stochastic model for temporal data by examining its properties. This method utilizes the stochastic model to forecast the long-term trend Tingting Huang et al. (2010). In order to predict the order quantities for a given month we generate time series features in addition to the descriptive features of *part_id*s. We generate lag-N features where N represents the number of months. For example, *order_quantity_lag3* represents the order quantity with a lag of three months as a moving window of months. In addition we take the sum, mean, and standard deviations of lag-N features for $1 \leq N \leq 6$ to test which lag best explains the target feature.

4.5 Clustering Categorical Features

The descriptive features of *part_id*s are in ID format, meaning that they cannot be used for modeling purposes as several of Sklearn's Pedregosa et al. (2011) algorithm implementation cannot handle categorical features. Thus, we need a way to convert these categorical features into continuous features. We use binary encoding, which encodes information to a bit, for the features that have a few number of categories Todd (2014). For the rest, we apply K-means clustering with $K = 5$. K-means clustering is an unsupervised learning algorithm that groups data points into K clusters by calculating their Euclidean distance Nuchprayoon (2014). The algorithm first randomly creates K center points, which are called centroids, then in each iteration, it assigns the data points to the cluster with the closest centroid.

The reason why we applied K-means clustering to some of our categorical features is to avoid the possibility of overfitting during modeling. As can be seen in Table 1, there are too many distinct values for *part_definition_id, part_product_class_id, and part_family_id* that can cause the model to overfit.

Therefore, we used normalized order quantity averages and standard deviations of each unique value for each feature to assign to clusters. Figure 2 shows scatterplot of mean versus standard deviation of order_quantity for three descriptive features, before and after K-means clustering. It can be observed that smaller size (quantity ordered) orders are more densely clustered with a lower mean and standard deviation compared to larger size orders that have a larger standard deviation. Similar clustering pattern can be seen across all three descriptive features namely *part_definition_id, part_product_class_id, part_family_id.*

4.6 Binary Encoding Categorical Features

We clustered three of five descriptive features of *part*s. We transform the remaining two features, *common_part_catalog_id* and *preferred_supplier_id*, into either a continuous variable or a binary variable in order to be able to use those for modeling. We chose to convert *common_part_catalog_id* with a default transformation as it has a lot of unique values. *part_family_id* can be transformed to a binary variable using binary encoding. Binary encoding is a categorical feature transformation method that creates N new binary features where N is the number of unique values for that categorical feature Todd (2014). Thus, in our case, we create 5 new features for *preferred_supplier_id* and specify those with a suffix *_binenc[N].*

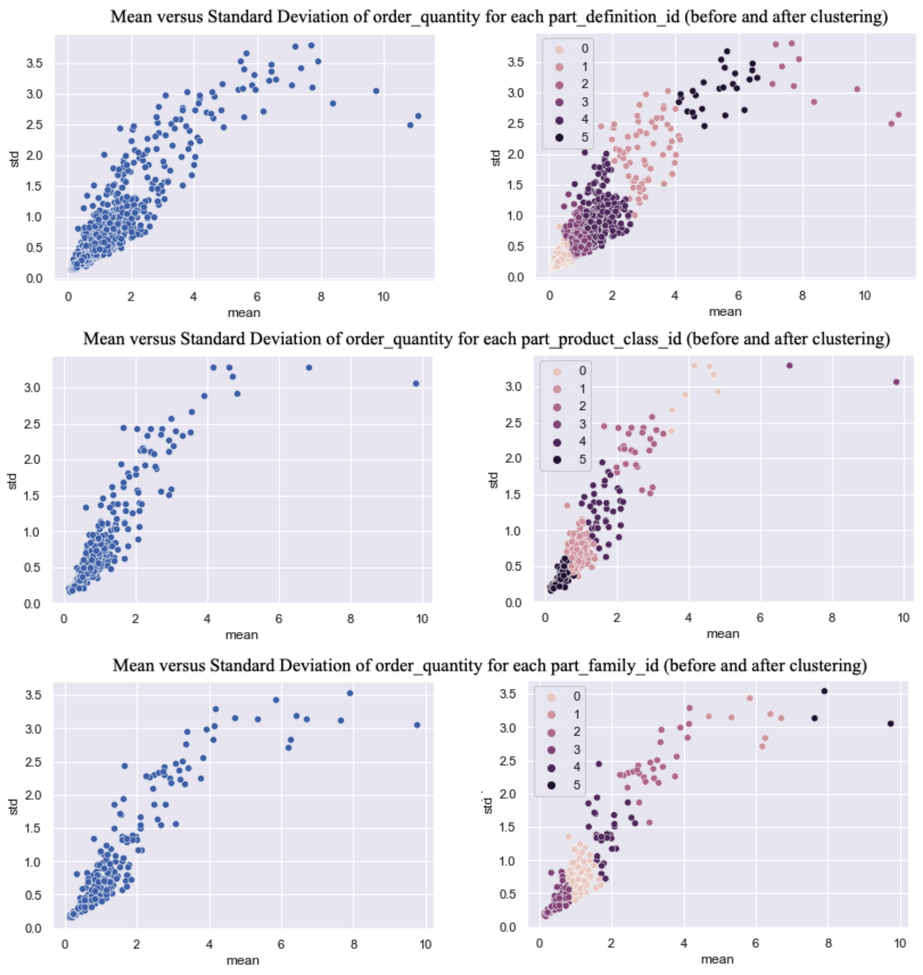

Figure 2: Scatterplot of mean versus standard deviation of order_quantity for three descriptive features, before and after K-means clustering.

4.7 Handling Null Values

When we create new features, particularly in Time Series Analysis, some values are null due to non-existent previous data. Therefore, in order to build the models, we need to fill those null values. As a common approach, we first tried to fill the nulls with their respective category mean value. After model trials, however, we decided to continue with filling with category maximums because we see a decreasing trend in order quantities. The intuition behind is that the lag-N features represent the order quantities N months ago, and they could be higher than the category maximum for some specific case.

4.8 Finalizing Training Data

See Figure 3 for the feature generation pipeline. Four types of features are generated on the raw data, namely, time series features, clustered features, binary encoded features and ordinal encoded features. Duplicates and outliers are removed and missing values are filled, the training data has 23 features and 167,059 data points. The training data consists of 167,059 rows with the target feature, *order_quantity_bin*, and descriptive features as below:

- time_diff_mean: Average time difference in days between the orders of this part_id.
- time_diff_std: Standard deviation of time differences in days between the orders of this part_id.
- order_quantity_lag[N] where $1 \leq N \leq 6$: The total order quantity for this part_id N months ago.
- order_quantity_lag[N]_mean where N in (3, 6): Average of last N months orders for this part_id.
- order_quantity_lag[N]_std where N in (3, 6): Standard deviation of last N months orders for this part_id.
- pdid_cluster: Clustered categorical feature from part_definition_id.
- pfid_cluster: Clustered categorical feature from part_family_id.
- pcid_cluster: Clustered categorical feature from part_product_class_id.
- psid_binenc[N] where $1 \leq N \leq 5$: Binary Encoded features from preferred_supplier_id.
- psid: Ordinal Encoded feature from preferred_supplier_id.
- cpcid: Ordinal Encoded feature from common_part_catalog_id.

4.9 Feature Selection

Before building the models, we want to eliminate some of the features that are highly correlated with each other. Decision has to be made to drop one of the two correlated features. In order to build better models, we need to know which feature is describing the outcome better than the other. Therefore, we first use a SVR algorithm from Sklearn's Linear Support Vector Regression module in order to get coefficients of each feature. After applying the SVR, we observe the most and least correlated features with the target feature as shown in Figure 4. The three most important features are *order_quantity_lag6_mean, order_quantity_lag3_mean, order_quantity_lag1*. The higher the importance of a feature the more it contributes toward characterizing the data.

Now that we know which feature performs better than the other, we can build a correlation matrix and determine a maximum threshold that can show correlation between two features. We select 0.90 as the threshold and starting from the most important feature, we scan the rest of the features and exclude them as necessary. Figure 5 shows the correlation matrix where darker color represents higher correlation.

After running the scan, we drop the following features due to high correlation with other features: *time_diff_mean and order_quantity_lag6_sum*.

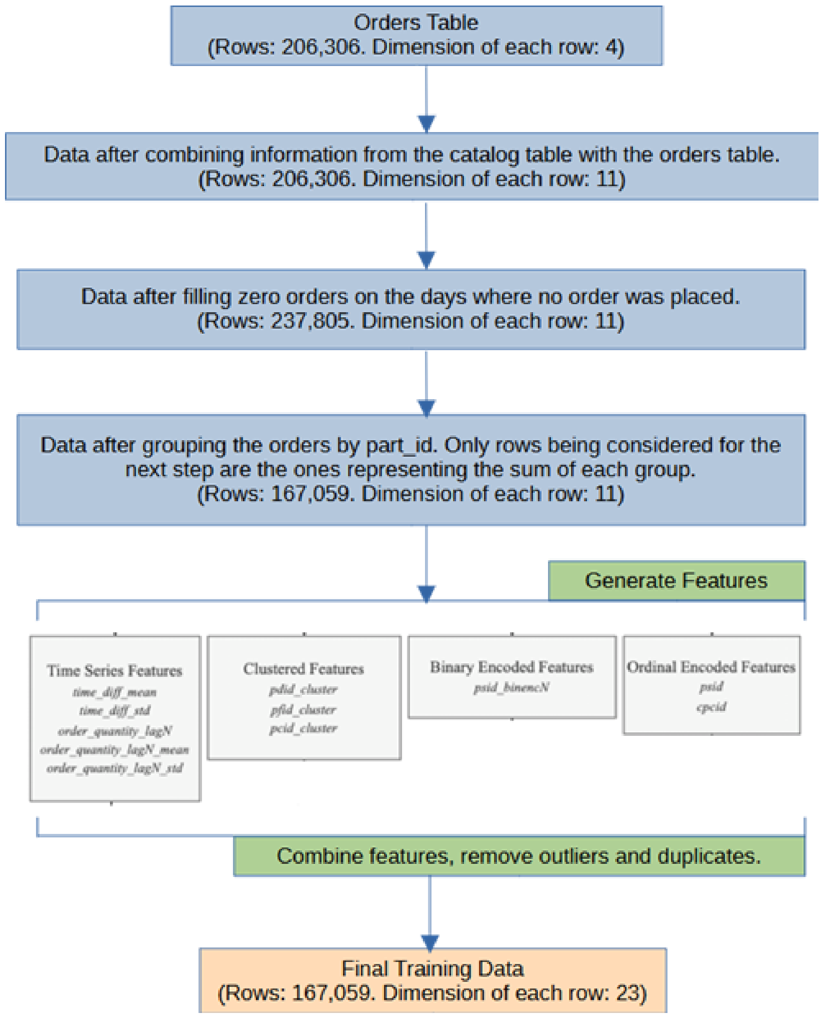
Orders Table
(Rows: 206,306. Dimension of each row: 4)
Data after combining information from the catalog table with the orders table.
(Rows: 206,306. Dimension of each row: 11)
Data after filling zero orders on the days where no order was placed.
(Rows: 237,805. Dimension of each row: 11)
Data after grouping the orders by part_id. Only rows being considered for the next step are the ones representing the sum of each group.
(Rows: 167,059. Dimension of each row: 11)
Generate Features
Time Series Features
time_diff_mean
time_diff_std
order_quantity_lagN
order_quantity_lagN_mean
order_quantity_lagN_std
Clustered Features
pdid_cluster
pfid_cluster
pcid_cluster
Binary Encoded Features
psid_binencN
Ordinal Encoded Features
psid
cpcid
Combine features, remove outliers and duplicates.
Final Training Data
(Rows: 167,059. Dimension of each row: 23)

Figure 3: Demand trend modeling dataset generation.

5 Supervised Machine Learning Methodology

For our research we use below machine learning methods. We found there were not many researches for automobile parts demand trend modeling that use ensemble models such as Random Forest and Gradient Boosting.

5.1 Support Vector Regressor (SVR)

Support Vector Regressor is a regression method that generates its predictions using a hyperplane created in the feature space. The hyperplane is created in such a way that it minimizes the total error (e.g., sum of the results from kernel function for each data point). We use the Radial Basis Function as the kernel function for demand trend modeling.

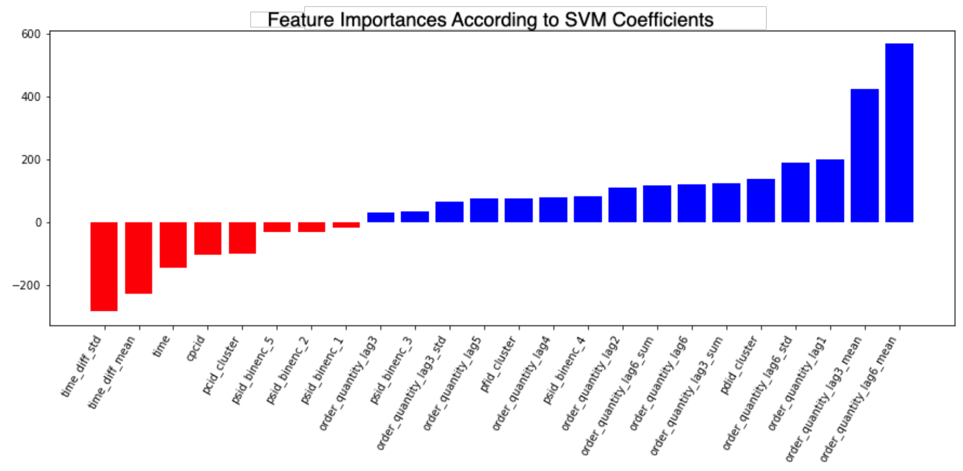

Figure 4: Bar chart of feature importance according to Support Vector Machine Regressor coefficients.

5.2 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a classification technique that uses a given set of neighbors to classify new data. The K represents the number of nodes the algorithm will use to classify new data by finding the distance of the new data from its neighbors Khan et al. (2019).

5.3 Random Forest (RF)

Random Forest is an ensemble learning method that utilizes decision trees to make predictions by taking either the mode of the decision trees or the mean prediction of each tree Lahouar & Ben Hadj Slama (2015).

5.4 Gradient Boosting (GB)

Gradient Boosting is a machine learning method for regression and classification problems. It produces predictions, usually by using decision trees, and minimizes error by modifying the weights of the trees that produce the best outcomes Gumus & Kiran (2017).

5.5 Back Propagation Neural Networks (BPNN) and Deep Neural Networks (DNN)

We use both multilayer perceptron model as well as DNNs with several hidden layers in our research. Deep Neural Networks (DNNs) are sophisticated Machine Learning models characterized by numerous hidden layers. The output of each layer serves as the training data for the next layer, culminating in the generation of classification probabilities or regression values at the final (prediction) layer. DNNs can feature multiple hidden layers, each potentially containing a different number of neurons. The activation function of each layer dictates how individual neurons respond to input instances. Lastly, the predictor layer combines probabilities produced in the next-to-last dense layer to come up with a single prediction for a data instance. A prevalent challenge encountered with DNNs is the convergence issue, often

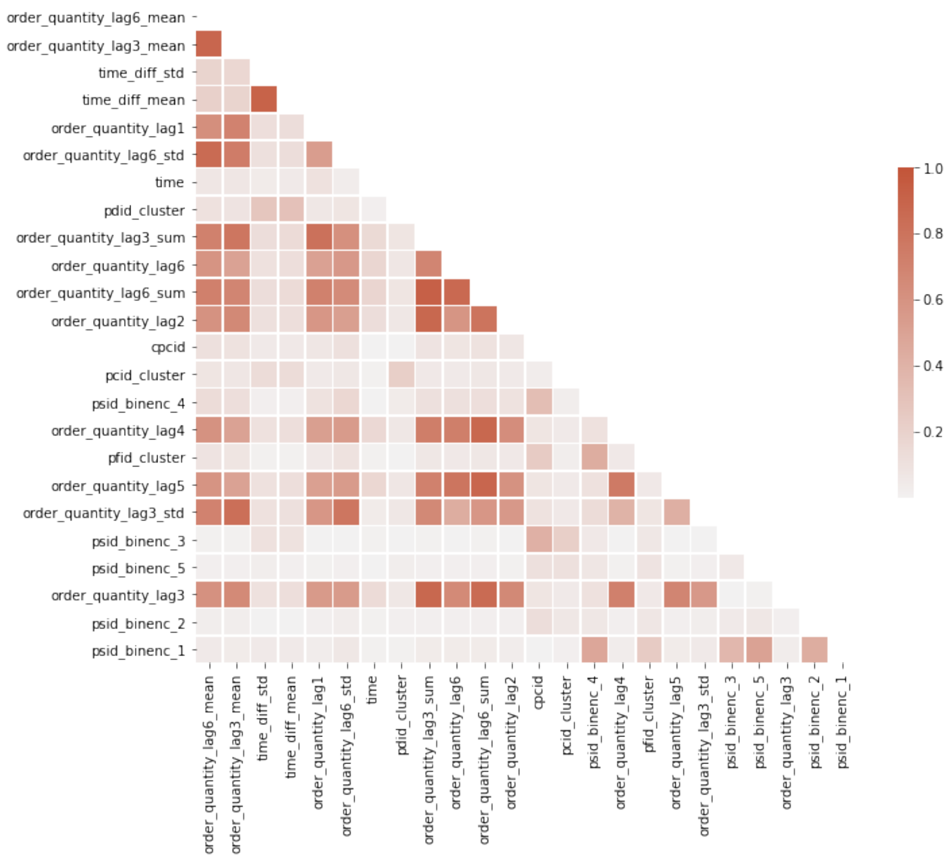

Figure 5: Correlation matrix of features ordered from most important to least important.

exacerbated by their susceptibility to overfitting. Mitigating overfitting can be accomplished by reducing model complexity, such as by introducing a dropout layer between dense layers.

5.6 *Validation Strategy*

We decided to move forward with Time Series Cross Validation (TSCV) method in this research. In TSCV, we select K folds just as we do in the K-fold cross validation, but we separate the folds according to a Time Series split instead of a random and/or shuffled split Medar et al. (2018). Figure 6 shows total order quantity for each month.

Next we define our validation strategy and select the time ranges for the models. We select five folds with each fold's validation period being five months as shown in Figure 7, this is based on empirical estimate that five months of order data consisting of 25,000 rows is an adequate number to objectively evaluate performance metrics.

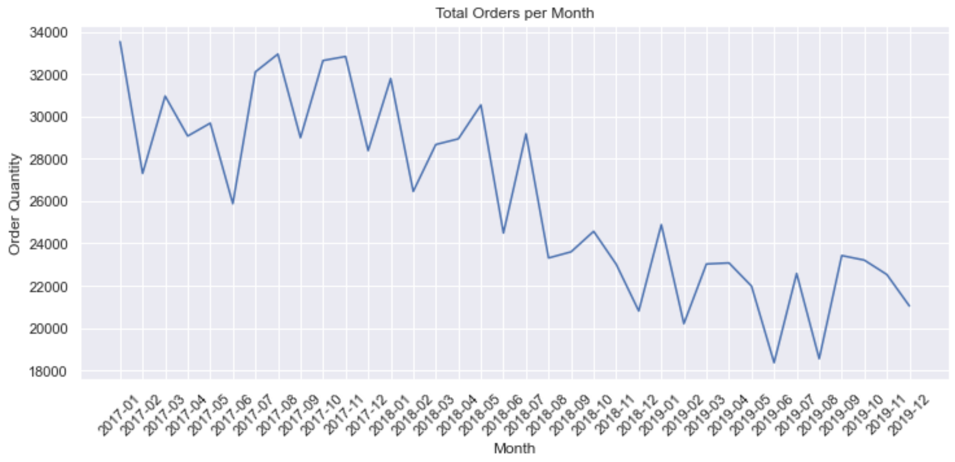

Figure 6: Total order quantity for each month.

Fold\Period	2017-01 2018-04	2018-04 2018-08	2018-08 2018-12	2018-12 2019-04	2019-04 2019-08	2019-08 2019-12
1	TRAINING	VALIDATION				
2	TRAINING		VALIDATION			
3	TRAINING			VALIDATION		
4	TRAINING				VALIDATION	
5	TRAINING					VALIDATION

Figure 7: Time Series Cross Validation (TSCV) with five folds used in this research.

6 Experiment Setup

To implement our research, we used Jupyter Notebook in Python 3 with the data manipulation and modeling tools such as Sklearn, Pandas, and Numpy. For data visualization we made use of Seaborn and Matplotlib. The main programming language is Python and the tools used in the study are Pandas, NumPy, Plotly, and Sci-kit learn Ghaddar & Naoum-Sawaya (2018); McKinney & Team (2015); Pedregosa et al. (2011); Walt et al. (2011).

After inserting missing rows, correcting the data, clustering categorical features, filling null values, dropping correlated features, deciding on the modeling methods and validation strategy, we finally build the KNN, RF, GB, SVR, BPNN, and DNN models. There are several parameters used in each model, in this research we will focus our discussion on some of the more important parameters. We use Sklearn's Pedregosa et al. (2011) machine learning library to build our models.

To assess the performance of each model type, we opted for Mean Absolute Error (MAE) and Mean Squared Error (MSE) as they are widely utilized metrics in machine learning. MAE range is in the target's scale, thus making it easier to interpret the results. Table 4 provides a detailed description of parameter set used to train various machine learning models for parts demand modeling.

Table 4: Description of Parameters used in Various Models for Parts Demand Modeling

Model Name and Parameter(s)
Model Name: SVR *max_iter=10,000*. Maximum number of iterations for running update function. *kernel='rbf'*. The radial basis function is used in the model. *tol=0.00001*. Tolerance value for early stopping.
Model Name: RF *n_estimators=100*. Number of decision trees to be used in the forest. *max_depth=7*. Maximum depth of a decision tree. *min_samples_split=100*. Minimum number of data instances required to make a split from a node. *min_samples_leaf=40*. Minimum number of data instances required in a leaf node. *max_leaf_nodes=45*. Maximum number of leaf nodes. *bootstrap=True*. Use bootstrapping when generating subset for each estimator.
Model Name: GB *n_estimators: [100, 200, ..., 1000]*. Number of decision trees to be used in the forest. *learning_rate: 0.01*. The value is used to determine how fast the estimator will learn the target features. *max_depth: 7*. Maximum depth of a decision tree. *max_leaf_nods: 45*. Maximum number of leaf nodes.
Model Name: DNN **Base Model:** Normalization Layer Dense Layer with 128 nodes and ReLU activation Prediction Layer *Loss = MAE, Optimizer = Adam, Learning Rate = 0.01, Epochs = 20*
2-Layer Model with SGD Normalization Layer Dense Layer with 128 nodes and ReLU activation

Continued on next page

Table 4: Description of Parameters used in Various Models for Parts Demand Modeling (Continued)

Dense Layer with 64 nodes and ReLU activation Prediction Layer *Loss = MAE, Optimizer = SGD, Learning Rate = 0.001, Epochs = 30*
3-Layer Model with Adam: Normalization Layer Dense Layer with 128 nodes and ReLU activation 30% Dropout Layer Dense Layer with 64 nodes and ReLU activation 30% Dropout Layer Dense Layer with 32 nodes and ReLU activation Prediction Layer *Loss = MAE, Optimizer = Adam, Learning Rate = 0.001, Epochs = 40*
3-Layer Model with SGD: Normalization Layer Dense Layer with 128 nodes and ReLU activation 30% Dropout Layer Dense Layer with 64 nodes and ReLU activation 30% Dropout Layer Dense Layer with 32 nodes and ReLU activation Prediction Layer *Loss = MAE, Optimizer = SGD, Learning Rate = 0.001, Epochs = 40*
Best Model: Normalization Layer Dense Layer with 128 nodes and ReLU activation Dense Layer with 64 nodes and ReLU activation Dense Layer with 32 nodes and ReLU activation Prediction Layer *Loss = MAE, Optimizer = Adam, Learning Rate = 0.001, Epochs = 50*

Table 5 Training and validation results of KNN model

	Five-Fold Results						**Standard**
Metric	**1**	**2**	**3**	**4**	**5**	**Average**	**Deviation**
RMSE Train	6.14	6.25	6.37	6.52	6.71	6.40	0.18
RMSE Validation	7.02	6.90	7.04	7.37	7.66	7.20	0.25
MAE Train	3.51	3.59	3.70	3.82	3.96	3.72	0.15
MAE Validation	3.92	3.88	3.90	4.24	4.58	4.10	0.25

7 Experiment Results and Discussion

7.1 K-Nearest Neighbor Model (KNN) Results

In the KNN model we initially set *n_neighbors* parameter to *5*. The remaining parameters are set to their default value. For example, the *weights* parameter is set to ‘uniform’, meaning that all the points in each neighborhood are weighted equally.

After running the KNN model, we obtained an average RMSE of *7.20* and MAE of *4.10* on our validation data. Table 5 shows the metrics details for each fold. The standard deviation across five-folds on the validation set for the RMSE and MAE is *0.25*. Since training and validation performance is close to each other, we do not see any issue of overfitting.

7.1.1 KNN with increasing number of neighbors

After running the KNN model with 2, 3, 4, ..., 10, 20 neighbors, we end up with an optimal RMSE of *6.77* on validation set when the neighbors $n = 20$, which is significantly better than it was when neighbors $n = 5$ (7.20). Figure 8 depicts how training and validation scores differ at varying number of neighbors.

7.2 Random Forest Model (RF) Results

RF model can have many parameters for creating a tree or when ensembling the trees. Below are our settings for the parameters:

- $n\_estimators = 100$
- $criterion = mse$
- $max\_depth = 7$
- $min\_samples\_split = 100$
- $min\_samples\_leaf = 40$
- $max\_leaf\_nodes = 45$

After running the RF model, we obtained a RMSE of *6.6147* and MAE of *3.82* on our validation data. Table 6 represents the metrics details for each fold. The standard deviation across five-folds on the validation set for the RMSE and MAE is *0.19* and *0.46* respectively. Since training and validation performance is close to each other, we do not see any issue of overfitting.

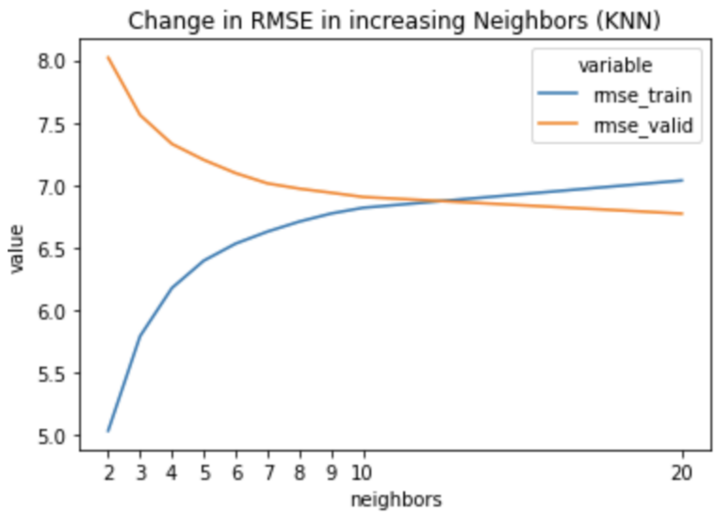

Figure 8: KNN model results at varying number of neighbors.

Table 6 Training and validation results of RF model

	Five-Fold Results						**Standard**
Metric	**1**	**2**	**3**	**4**	**5**	**Average**	**Deviation**
RMSE Train	6.79	6.87	7.00	7.13	7.27	7.02	0.16
RMSE Validation	6.46	6.43	6.50	6.71	6.97	6.61	0.19
MAE Train	3.98	4.06	4.17	4.29	4.44	4.19	0.15
MAE Validation	3.62	3.63	3.72	4.93	4.20	3.82	0.46

7.2.1 RF with increasing number of estimators

The RF model's performance changes only slightly at increasing number of estimators. Figure 9 shows RMSE at increasing number of estimators. RF model has an average RMSE of 6.61 on the validation set, which is better than the KNN model.

7.3 Gradient Boosting Model (GB) Results

In GB model we set the parameters same as they are being set for RF model. Since the two methods are both based on decision trees, they have similar parameters. Methods differ in the way they ensemble.

After running the GB model, we obtained an average RMSE of *6.8383* and MAE of *4.09* on our validation data. Table 7 shows the metrics details for each fold. The standard deviation across five-folds on the validation set for the RMSE and MAE is *0.23* and *0.26* respectively. Since training and validation performance is close to each other, we do not see any issue of overfitting.

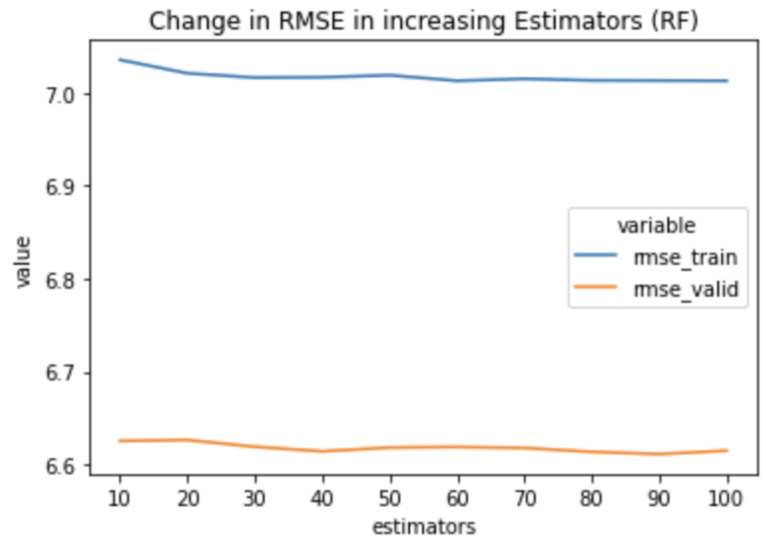

Figure 9: RF model results at increasing number of estimators.

Table 7 Training and validation results of GB model

	Five-Fold Results						Standard
Metric	**1**	**2**	**3**	**4**	**5**	**Average**	**Deviation**
RMSE Train	7.14	7.25	7.42	7.59	7.74	7.43	0.20
RMSE Validation	6.66	6.64	6.66	6.94	7.30	6.84	0.23
MAE Train	4.19	4.31	4.69	4.63	4.78	4.47	0.21
MAE Validation	3.79	3.87	3.99	4.29	4.56	4.09	0.26

7.3.1 GB with increasing number of estimators

The GB model has the average RMSE of 6.56 when the number of estimators is $n = 1000$, the result is better than KNN and RF. Since the ensembling method of GB is different from RF, that is, GB calculates the error at every other estimator while RF is a voting-based method, we expected a decrease in the error when we increased the number of estimators. Due to our limited computing power, we did not investigate further the effects of increasing the estimators beyond $n = 1000$. Figure 10 shows how the change in estimators changes the model performance.

7.4 Model Comparison between KNN, RF, and GB

After running each of the three models, i.e., KNN, RF, and GB with different parameter sets, we observe that the best RMSE of 6.56 on validation set is obtained from GB. We present the actual versus predicted order quantities graph in Figure 11, we observe that the prediction trend fits the actual outcome, which shows the importance of parameter tuning of machine learning models. We observe that GB is close to the actual trend. Although KNN

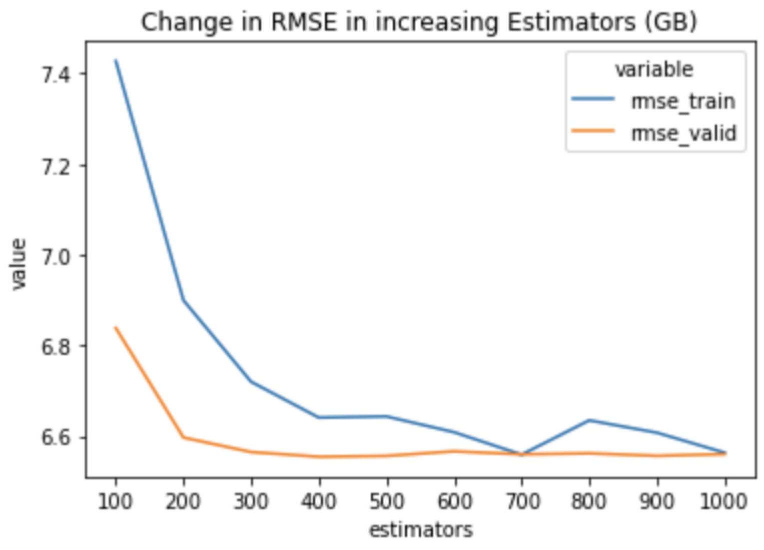

Figure 10: GB model results at increasing number of estimators.

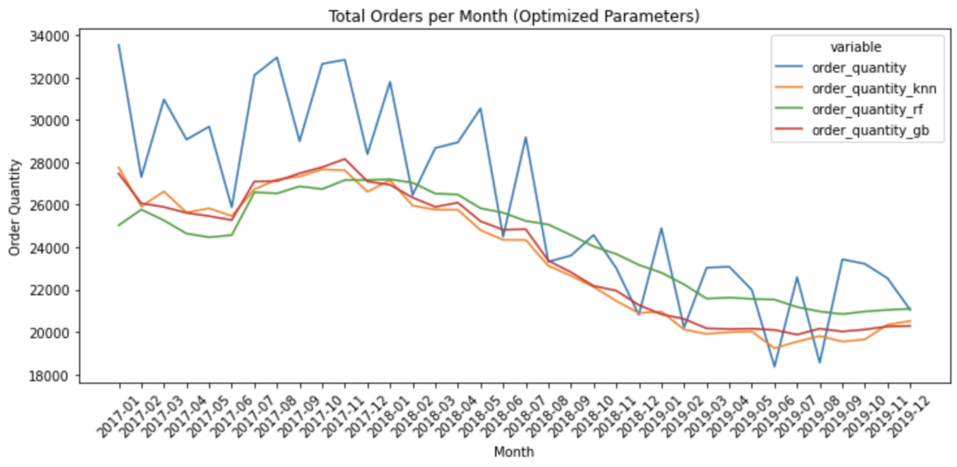

Figure 11: Actual versus predicted order quantities with KNN, RF, and GB models.

predicted close to GB, GB was slightly more successful in modeling the ups and downs of the parts demand trend.

7.5 Time Efficiency of KNN, RF, and GB Models

Due to the inherent complexity, the ensemble models such as RF and GB are expected to take longer than KNN. KNN model fitted in *5.053* seconds, while RF fitted in *58.620* seconds, and GB fitted in *103.104* seconds. Fitting time of RF could be reduced by using hardware that supports greater parallelism.

Table 8 Training and validation results of SVR model

	Five-Fold Results						**Standard**
Metric	**1**	**2**	**3**	**4**	**5**	**Average**	**Deviation**
RMSE Train	7.42	7.55	7.74	7.94	8.20	7.77	0.25
RMSE Validation	6.77	6.80	6.79	7.19	7.49	7.01	0.26
MAE Train	4.21	4.32	4.99	4.66	4.87	4.51	0.28
MAE Validation	3.61	3.64	3.69	3.95	4.20	3.82	0.21

Table 9 Training and validation results for BPNN model

	Five-Fold Results						**Standard**
Metric	**1**	**2**	**3**	**4**	**5**	**Average**	**Deviation**
RMSE Train	6.74	6.93	7.04	7.30	7.41	7.08	0.22
RMSE Validation	6.59	6.69	6.60	6.94	7.22	6.81	0.22
MAE Train	3.85	4.03	4.19	4.34	4.36	4.15	0.18
MAE Validation	3.57	3.64	3.63	3.93	4.18	3.79	0.21

7.6 Support Vector Regressor (SVR) Model Results

After running the SVR model, we obtained the results presented in Table 8. With 7.01 average RMSE and MAE of *3.82* on validation set, the SVR model performs lower than the RF and GB models. The standard deviation across five-folds on the validation set for the RMSE and MAE is *0.26* and *0.21* respectively. Since training and validation performance is close to each other, we do not see any issue of overfitting. Thus, we conclude that the ensemble models perform better on the spare parts dataset.

7.7 Back Propagation Neural Networks (BPNN) Model Results

The results obtained from BPNN model are shown in Table 9. We built BPNN model in Python using Scikit Learn's MLPRegressor module with a maximum number of iterations from 200, 300, 400, ..., 1000. Ultimately obtained an average RMSE score of *6.81* and MAE of *3.79* on the validation set, which is slightly higher than GB model's result *6.56*. The standard deviation across five-folds on the validation set for the RMSE and MAE is *0.22* and *0.21* respectively. Since training and validation performance is close to each other, we do not see any issue of overfitting.

7.8 Deep Neural Networks (DNN) Model Results

We used Keras' DNN model with various configurations that could outperform the Gradient Boosting model. These variations included 2-3 hidden layers with a default normalization layer, and dropout layers. We also used learning decay to achieve the local minima, reaching the RMSE validation score of 6.25. This score is significant to conclude that Deep Neural Networks are better in predicting the intermittent demand in comparison to the rest of the models.

Table 10 Training and validation results of various DNN models

Model Name	**Layers**	**Epochs**	**Dropout**	**Optimizer**	**Learning Rate**	**RMSE Train**	**RMSE Validation**
Base	128	20	-	Adam	0.01	**7.71**	**7.10**
2-Layer SGD	128-64	30	-	SGD	0.001	**6.76**	**6.78**
3-Layer Adam	128-96-64	40	30%	Adam	0.01	**7.36**	**7.08**
3-Layer SGD	128-64-32	40	10%	SGD	0.001	**7.33**	**7.04**
Best Model	128-64-32	50	10%	Adam	0.001	**5.21**	**6.25**

Table 10 shows the results of DNN models. More than 20 configurations of DNN with different parameters have been run and 5 significant models are presented. The best result is presented in the last row - the three layers, 128-64-32 node DNN model at 100 epochs of training. Since training and validation RMSE performance is close to each other, we do not see any issue of overfitting. The 6.25 RMSE score on the validation set obtained from the best model is significantly better than the results obtained from rest of the machine learning models in this research

7.9 Challenges in Demand Trend Modeling of Spare Parts

Challenges of this research include newly added products, missing parameters, outlier detection, and high computational requirements. For the purposes of demand trend modeling, newly added products may not have enough available historical data to make an accurate prediction. One of the missing parameters is price, if available it may have strong correlations to sales orders and could improve the prediction. The price of an item may be correlated to sales orders because if the product is expensive, there may be fewer sales orders placed and opposite if the price is low. The computational requirements are expensive as hyperparameter grid is increased.

8 Conclusion and Future Work

Based on the research conducted, we conclude that DNN performs better than rest of the machine learning models on the spare parts dataset. Performance of BPNN and GB is close to each other. After a thorough literature review, it is observed that there are few previous researches done on Gradient Boosting Regression for demand trend modeling. Our research proves that Gradient Boosting's performance is strong across traditional machine learning models. Furthermore, SVR has been previously used for demand trend modeling of spare parts but other models that we tested perform better.

To get a rich feature set our feature generation strategy includes generating time series features, we applied clustering the categorical features, inserting zero orders for the days that there were no orders, and Binary Encoding of the rest of categorical features.

As future work we plan to look into specific DNNs that might perform well in demand trend modeling, such as Recursive DNNs. Other methods that we plan to explore are Bayesian hyperparameter optimization for the GB model and combining DNNs with traditional machine learning methods.

References

Amin-Naseri, M. R. & Rostami Tabar, B. (2008), 'Neural network approach to lumpy demand forecasting for spare parts in process industries', *Proceedings of the International Conference on Computer and Communication Engineering 2008, ICCCE08: Global Links for Human Development* pp. 1378–1382.

Boukhtouta, A. & Jentsch, P. (2018), Support vector machine for demand forecasting of canadian armed forces spare parts, *in* '2018 6th International Symposium on Computational and Business Intelligence (ISCBI)', pp. 59–64.

Ghaddar, B. & Naoum-Sawaya, J. (2018), 'High dimensional data classification and feature selection using support vector machines', *European Journal of Operational Research*.

Gumus, M. & Kiran, M. S. (2017), 'Crude oil price forecasting using XGBoost', *2nd International Conference on Computer Science and Engineering, UBMK 2017* pp. 1100–1103.

Hua, Z. & Zhang, B. (2006), 'A hybrid support vector machines and logistic regression approach for forecasting intermittent demand of spare parts', *Applied Mathematics and Computation* **181**(2), 1035–1048.

Ikizler, T., Ozcelik, A. E. & Uslu, B. C. (2024), 'Predictive analysis of the number of spare parts and techincians in service employment in the automobile industry', *Selcuk University Journal of Engineering Sciences* **23**(3), 92–98.

Khan, S., Khan, Z. A., Noshad, Z., Javaid, S. & Javaid, N. (2019), 'Short Term Load and Price Forecasting using Tuned Parameters for K-Nearest Neighbors', *ITT 2019 - Information Technology Trends: Emerging Technologies Blockchain and IoT* pp. 89–93.

Lahouar, A. & Ben Hadj Slama, J. (2015), 'Random forests model for one day ahead load forecasting', *2015 6th International Renewable Energy Congress, IREC 2015*.

Lu, C., Shang, G., Xu, L., Shao, H. & Zhang, B. (2023), 'Sales volume forecast of typical auto parts based on bigru: A case study', *E3S Web Conf.* **409**, 04008.

McKinney, W. & Team, P. D. (2015), 'Pandas - powerful python data analysis toolkit', *Pandas - Powerful Python Data Analysis Toolkit*.

Medar, R., Rajpurohit, V. S. & Rashmi, B. (2018), 'Impact of Training and Testing Data Splits on Accuracy of Time Series Forecasting in Machine Learning', *2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA 2017* (April 2020), 1–6.

Nuchprayoon, S. (2014), 'Electricity load classification using K-means clustering algorithm', *IET Conference Publications*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Qiu, Q., Qin, C., Shi, J. & Zhou, H. (2019), Research on demand forecast of aircraft spare parts based on fractional order discrete grey model, *in* ‘2019 IEEE 5th International Conference on Computer and Communications (ICCC)’, pp. 2212–2216.

RuiRui Xing, Xianliang Shi, Member, I. (2019), ‘A BP-SVM combined model for intermittent spare parts demand prediction’, pp. 1085–1090.

Shan, H. S., Qin, M., Zhang, L., Meng, Z. & Peng, P. (2023), ‘The annual sales forecast for a chinese auto parts manufacturer based on igm (1,1)’, *Journal of Grey System* **35**(1), 113.

Shenstone, L. & Hyndman, R. J. (2003), ‘Stochastic models underlying croston ’ s method for intermittent demand forecasting stochastic models underlying croston ’ s method for intermittent demand forecasting’, *Business*.

Song, H., Zhang, C., Liu, G. & Zhao, W. (2012), Equipment spare parts demand forecasting model based on grey neural network, *in* ‘2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering’, pp. 1274–1277.

Tingting Huang, Wang, L. & Jiang, T. (2010), Prognostics of products using time series analysis based on degradation data, *in* ‘2010 Prognostics and System Health Management Conference’, pp. 1–5.

Todd, M. D. (2014), *Sensor data acquisition systems and architectures*, Vol. 1, Woodhead Publishing Limited.

Walt, S. V. D., Colbert, S. C. & Varoquaux, G. (2011), ‘The numpy array: A structure for efficient numerical computation’, *Computing in Science and Engineering*.

Wanchoo, K. (2019), ‘Retail Demand Forecasting: a Comparison between Deep Neural Network and Gradient Boosting Method for Univariate Time Series’, pp. 5–9.

Xu, Q., Wang, N. & Shi, H. (2012), ‘A Review of Croston’s method for intermittent demand forecasting’, *Proceedings - 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2012* (61100009), 1456–1460.

Zamani, F., Ebrahimi, A., Soltani, R. & Farhang Moghaddam, B. (2025), ‘Predicting the lead time of auto parts orders in the supply chain using machine learning’, *Business Intelligence Management Studies* **13**(52), –.