
**International Journal of Multidisciplinary Research and Growth Evaluation**
**ISSN: 2582-7138**
**Received: 27-07-2020; Accepted: 22-08-2020**
**www.allmultidisciplinaryjournal.com**
**Volume 1; Issue 3; July-August 2020; Page No. 82-92**

# Demand-centric Inventory Forecasting Approach: Comparing Regression Methods

**Jwalin Thaker**
Software Engineer (AI/ML), Independent Researcher, Ahmedabad, India

Corresponding Author: **Jwalin Thaker**

**DOI: https://doi.org/10.54660/.IJMRGE.2020.1.3.82-92**

**Abstract**
With our ever-growing population and the increas- ing demand for goods (raw or finished), companies running manufacturing/production and retail stores managing products have had their plates full while trying to maintain the demand- supply equilibrium. Most of the established enterprises have a separate division focusing on keeping the supply and stocking afloat to meet the ongoing and future demand. To do a parallel study, we aim to get over 9 weeks of sales data from Grupo Bimbo (bakery industry) across Mexico and analyze it for purchase patterns to estimate the demand trend in the coming weeks. Our objective is to solve this problem statement by generating an inventory forecast based on the estimated demand by using machine learning techniques like Multiple Linear Regression, Stochastic Gradient Descent regression, Random forest regres- sion, and Gradient boosting (XGBoost).

**Keywords:** — Inventory forecasting, Machine learning, Multiple Linear Regression, Stochastic Gradient Descent, Random Forest, XGBoost, Demand prediction, Data analysis, Regression models

## 1. Introduction
The demand-based inventory forecasting is crucial for opti- mizing stock levels and minimizing costs. This paper explores various methodologies and introduces a machine learning approach to enhance forecasting accuracy. Under the large um- brella of inventory forecasting in various industries, we set out to understand the famous bakery industry in Mexico, ”Grupo Bimbo,” and how its various SKUs perform in different regions (towns and states) and across diverse clientele by the measure of Sales. By using the power of Python as a programming language and statistics and data modeling, we want to visualize the data or any standout trends and then estimate the demand that each product yields by applying regression techniques of machine learning. Please note that throughout the report, there has been a fair general assumption of neglecting classification techniques to solve this problem as it attributes properties analogous to a classic regression problem.

## 2. Related Work
There are various companies that tackle this problem with either manual computation and estimation by bookkeeping or by using statistical tools and methodologies to calculate the probable rate of demand of products against the current sales in the coming time frame. Only a few solutions exist that leverage Machine Learning and Artificial Intelligence to achieve a robust prediction system, minimizing production costs and optimizing inventory stocking for a company.
Recent research has shown that machine learning techniques can significantly improve supply chain demand forecasting compared to traditional statistical methods [1]. Lolli *et al* demonstrated how machine learning can be effectively applied to intermittent demand scenarios, which are particularly chal- lenging for inventory management [2]. In the retail sector, Chen and Lu combined clustering with machine learning techniques to enhance sales forecasting accuracy for computer products [3].
For intermittent demand patterns, which are common in inventory management, specialized forecasting methods have been developed. The seminal work by Croston [4] introduced a methodology specifically designed for intermittent demand patterns, while Babai *et al* [5] conducted empirical studies on the accuracy of intermittent demand forecasting and the associated risk of obsolescence.
More recently, deep reinforcement learning has been ex- plored as a promising approach to improve inventory man- agreement across various scenarios including dual sourcing, lost sales, and multi-echelon problems [6]. Additionally, Pavlyshenko [7] has shown how machine learning models can be effectively applied to sales time series forecasting.
Our work aims to combine data analysis studies that have been previously done and amalgamate them into learning the most relevant trends while forecasting inventory needs based on current demand. We wish to contribute to modeling the problem in a

way that eases inventory planning without compromising the efficiency of the ML prediction algorithm.

## 3. Our Solution

We propose a solution for calculating a measure of demand by analyzing a customer's purchase pattern based on the demographic region and store location town-wise, taking into account the sales, per unit and weight, every week and also the returns, per unit and weight, current and coming week. With the idea of getting demand data points of clients, products, and depot stores (across various locations), we plan to use machine learning techniques revolving around the concept of regression to train a system capable of predicting which products (in number of units) will need over or under stocking to meet the surge or dip in demand in the coming weeks.

The plan of action involves implementing well-known machine learning algorithms like Multiple Linear Regression, Stochastic Gradient Descent Regression, Random Forest Regression, and Gradient Boosting to fit the Grupo Bimbo Sales data and give insights into the inventory forecasting. Post application, we will compare results of all the algorithms, with the metrics being accuracy and error rate. Our approach is informed by the work of Bishop [8], which provides a comprehensive framework for pattern recognition and machine learning techniques.

### A. Description of data set

The data is provided as a set of CSV files with train and test data as separate files from the Grupo Bimbo Inventory Demand competition. The other files provided are Client master data tables, the product master data and Sales Depot master data. The training data has size of 74180460 rows and 10 columns/Features consisting of the following data fields:

- Semana: Week number (From Thursday to Wednesday) from Week 3-9
- Agencia ID: Sales Depot ID - unique ID for each sales Depot
- Canal ID: Sales Channel ID - unique ID for each sales Channel
- Ruta SAK: Route ID - many to one relationship with Sales Depot
- Cliente ID: Client ID - can be used to join with Client ID on Client table for Client name
- Producto ID: Product ID - A unique ID for each product which can be used as foreign key to join with Product Master data to retrieve the Product Name
- Venta uni hoy: Sales unit this week (integer) ID for Sales Unit
- Venta hoy: Sales this week (unit: pesos)
- Dev uni proxima: Unit for Returns next week (integer) ID for unit for Returns volume
- Dev proxima: Returns next week (unit: pesos)
- Demanda uni equil: Adjusted Demand (integer) is the target variable that will be predicted in the test data which is always $\leq 0$ based on the demand of previous weeks

The client master data tables consist of the below data: (Note: The Client ID however is not a unique identifier as there are multiple clients with the same name against multiple Client IDs, this will require further cleaning and standardization)

- Cliente ID: Client ID
- NombreCliente: Name of the Client

The product master data consists of the below data:

- Producto ID: Product ID
- NombreProducto: Name of Product

Sales Depot Master data consists of the below data:

- Sales Depot ID

| | Semana | Agencia_ID | Canal_ID | Ruta_SAK | Cliente_ID | Producto_ID | Venta_uni_hoy | Venta_hoy | Dev_uni_proxima | Dev_proxima | Demanda_uni_equil |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 1110 | 7 | 3301 | 15766 | 1212 | 3 | 25.14 | 0 | 0.0 | 3 |
| 1 | 3 | 1110 | 7 | 3301 | 15766 | 1216 | 4 | 33.52 | 0 | 0.0 | 4 |
| 2 | 3 | 1110 | 7 | 3301 | 15766 | 1238 | 4 | 39.32 | 0 | 0.0 | 4 |
| 3 | 3 | 1110 | 7 | 3301 | 15766 | 1240 | 4 | 33.52 | 0 | 0.0 | 4 |
| 4 | 3 | 1110 | 7 | 3301 | 15766 | 1242 | 3 | 22.92 | 0 | 0.0 | 3 |
| 5 | 3 | 1110 | 7 | 3301 | 15766 | 1250 | 5 | 38.20 | 0 | 0.0 | 5 |
| 6 | 3 | 1110 | 7 | 3301 | 15766 | 1309 | 3 | 20.28 | 0 | 0.0 | 3 |
| 7 | 3 | 1110 | 7 | 3301 | 15766 | 3894 | 6 | 56.10 | 0 | 0.0 | 6 |
| 8 | 3 | 1110 | 7 | 3301 | 15766 | 4085 | 4 | 24.60 | 0 | 0.0 | 4 |
| 9 | 3 | 1110 | 7 | 3301 | 15766 | 5310 | 6 | 31.68 | 0 | 0.0 | 6 |
| 10 | 3 | 1110 | 7 | 3301 | 15766 | 30531 | 8 | 62.24 | 0 | 0.0 | 8 |
| 11 | 3 | 1110 | 7 | 3301 | 15766 | 30548 | 4 | 21.52 | 0 | 0.0 | 4 |
| 12 | 3 | 1110 | 7 | 3301 | 15766 | 30571 | 12 | 75.00 | 0 | 0.0 | 12 |
| 13 | 3 | 1110 | 7 | 3301 | 15766 | 31309 | 7 | 43.75 | 0 | 0.0 | 7 |
| 14 | 3 | 1110 | 7 | 3301 | 15766 | 31506 | 10 | 62.50 | 0 | 0.0 | 10 |
| 15 | 3 | 1110 | 7 | 3301 | 15766 | 32393 | 5 | 15.10 | 0 | 0.0 | 5 |
| 16 | 3 | 1110 | 7 | 3301 | 15766 | 32933 | 3 | 21.12 | 0 | 0.0 | 3 |

**Fig 1:** Training Dataset Sample Showing Feature Structure and Value Distribution

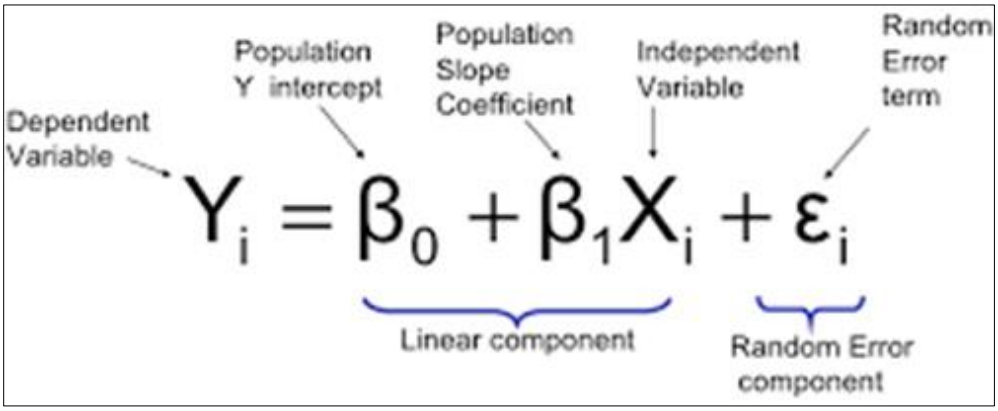


**Fig 2:** Multiple Linear Regression Equation with Feature Weights

- Town - Town of the Depot
- State - State of the Depot

As this dataset has a lot of features, we have therefore dropped unnecessary and redundant features. There were some null value entries within the dataset which we removed and cleaned from the main dataset.

**B. Machine learning algorithms**

Our primary objective is to predict Demand for future weeks based on the Returns/Sales of previous weeks therefore it makes sense to implement Regression algorithms as below.

- Multiple Linear Regression
- Stochastic Gradient Descent Regression (bonus)
- XGBoost Regression
- Random Forest Regression

**Multiple linear regression:** determines linear relationship that may exist between the features and the predictor variable. Mathematically it is represented as this expression:

In our case, if the columnar data like the product, client, and depot are all a match, we implement a linear regression model that includes a component for the specific case as well as for the more general case of product and route. We assume that each feature has some level of correlation to the predicted variable (adjusted demand in this case) and move ahead with that hypothesis. Due to the size of the data and its distribution, it is required to normalize each of the features for linear regression to make computationally shorter operations and be less performance intensive on the local machine.

**Stochastic gradient descent regression (SGD):** is a simple yet efficient machine learning technique to fit regressors and classifiers over convex loss functions. We fit this approach as a bonus because of the expected hardships to deal with large-scale learning from an enormous dataset. The advantages that it serves over conventional Ridge, Lasso or Elastic Net methodologies is that it picks up input batch data in a probabilistic random manner to train, thus being expedited and dynamic in terms of efficiency and areas of code tuning (has lots of parameters to tweak)

**Xtreme gradient boost (XGBoost)**: is an ensemble learning method and offers a straightforward and robust solution to combine the predictive power of multiple learning approaches. XGBoost has options to implement regularization to penalize complex models and handles missing data better due to the inherent Sparsity-aware Split algorithm, which is evident in our dataset. Due to the optimal hardware utilization properties owing to block structure, cache awareness, and out-of-core computing, we decided to start our implementation with the XGBoost algorithm.

**Random forest regression:** based on the fact that they perform better on large datasets and work well with missing data. The EDA has revealed that the sales/demand data for certain products/depots are non-linear hence Random forest seem very apt for such scenarios. We aim to explore all options available with the data sets and intend to cover as many hypotheses as we can to obtain optimum results and balance between performance and model robustness.

**C. Implementation Details**

**Exploratory data analysis**

Data pre-processing and preparation was started with treatment of null values, which were replaced with mean values as per the feature combinations. No rows were categorized as corrupt and thus all rows were retained. Normalization of data was performed wherever necessary for regressors to perform more efficiently.

Feature engineering was performed with the aim to select n best features from a bigger set N, based on metrics of correlation with fellow features and the dependent variable. As an outcome, feature reduction was carried out dropping feature columns like as Client/Product names and adding the state and town to identify relationship with depot and town locations of customer purchases. This approach aligns with the demand classification scheme proposed by Conceic ¸a˜o *et al* [9], which emphasizes the importance of proper feature selection for inventory models subject to stochastic demand.

About 3% of the data has no/zero returns and Sales. We decided to keep these data as part of the analysis because of the relationships and patterns observed in the EDA, for e.g. we could see customer with no sales for some weeks but had returns for the same weeks and vice versa.

Post cleaning of data, we went onto exploration and gaining insights from the dataset, performing various kinds of EDA (exploratory data analysis) on it. We analyzed the average and

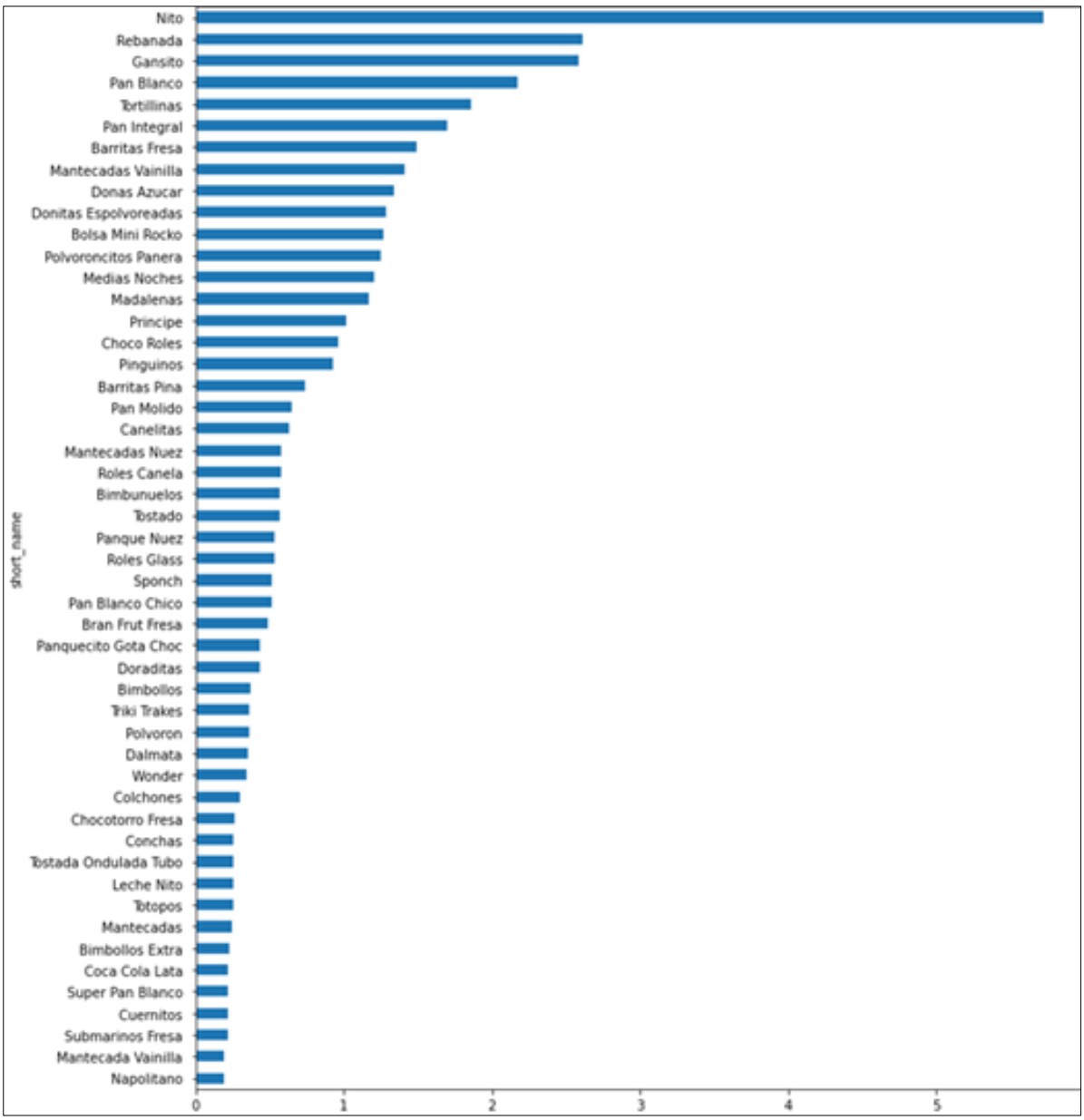


**Fig 3:** Demand Distribution for Top 20 Products Across All Depots

total demand for products over time across various attribute combinations like product/client or depot/client.

After model implementation, products should be intentionally overstocked if the product's demand is evaluated by data with no returns. We also noticed that returns were possibly a result of particular depots over estimating demand. This was derived from visual manual observation. We also tried to observe linear relationships between feature combinations (generating new features) and Adjusted Demands instead of individual features or all features. This was particularly challenging on the computing side, considering the size of the data so we had to resort to sampling techniques based on for

e.g. Weeks.

We also looked at the correlation matrix of all available features and predicted columns. The image below shows how each of the available columns inside the feature space fare out in terms of correlation against themselves, other features and the predicted variable (adjusted demand). General inference is that most of the independent variables are weakly correlated with others and thus ideally cannot be dropped during our model training. Also, we see that sales and sales (in weight) are moderately correlated implying that more the sales, more is the total weight of items sold that week, per depot. Also, sales seems to directly impact the dependent variable demand from the heatmap giving us an idea that it is an important factor in our regression analysis.

We also extend our EDA by looking at the total sales per week and the distribution of demand across states, and found that it is almost evenly distributed showing consistency against time. Some of the states have more customers buying

```
corr = train.corr()
corr.style.background_gradient(cmap='coolwarm')
```

| | Semana | Agencia_ID | Canal_ID | Ruta_SAK | Cliente_ID | Producto_ID | Venta_uni_hoy | Venta_hoy | Dev_uni_proxima | Dev_proxima | Demanda_uni_equil |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Semana | 1.000000 | 0.000453 | 0.007763 | -0.002924 | 0.002531 | 0.013829 | 0.003229 | 0.001296 | 0.000458 | 0.001432 | 0.003000 |
| Agencia_ID | 0.000453 | 1.000000 | -0.039682 | -0.008209 | 0.015518 | 0.006993 | -0.005505 | -0.006586 | -0.000117 | -0.000851 | -0.005479 |
| Canal_ID | 0.007763 | -0.039682 | 1.000000 | 0.471398 | 0.058214 | 0.029316 | 0.132532 | 0.106503 | 0.002742 | 0.017959 | 0.132098 |
| Ruta_SAK | -0.002924 | -0.008209 | 0.471398 | 1.000000 | 0.063129 | 0.284827 | 0.051319 | 0.022922 | 0.000369 | -0.001830 | 0.051844 |
| Cliente_ID | 0.002531 | 0.015518 | 0.058214 | 0.063129 | 1.000000 | -0.000501 | -0.011645 | -0.010770 | -0.000287 | -0.001983 | -0.011386 |
| Producto_ID | 0.013829 | 0.006993 | 0.029316 | 0.284827 | -0.000501 | 1.000000 | 0.046024 | 0.015644 | 0.000890 | 0.004669 | 0.045619 |
| Venta_uni_hoy | 0.003229 | -0.005505 | 0.132532 | 0.051319 | -0.011645 | 0.046024 | 1.000000 | 0.733678 | 0.008769 | 0.061938 | 0.997265 |
| Venta_hoy | 0.001296 | -0.006586 | 0.106503 | 0.022922 | -0.010770 | 0.015644 | 0.733678 | 1.000000 | 0.005383 | 0.057269 | 0.732973 |
| Dev_uni_proxima | 0.000458 | -0.000117 | 0.002742 | 0.000369 | -0.000287 | 0.000890 | 0.008769 | 0.005383 | 1.000000 | 0.129421 | 0.004135 |
| Dev_proxima | 0.001432 | -0.000851 | 0.017959 | -0.001830 | -0.001983 | 0.004669 | 0.061938 | 0.057269 | 0.129421 | 1.000000 | 0.035127 |
| Demanda_uni_equil | 0.003000 | -0.005479 | 0.132098 | 0.051844 | -0.011386 | 0.045619 | 0.997265 | 0.732973 | 0.004135 | 0.035127 | 1.000000 |

**Fig 4:** Feature Correlation Matrix Showing Relationships Between Key Variables

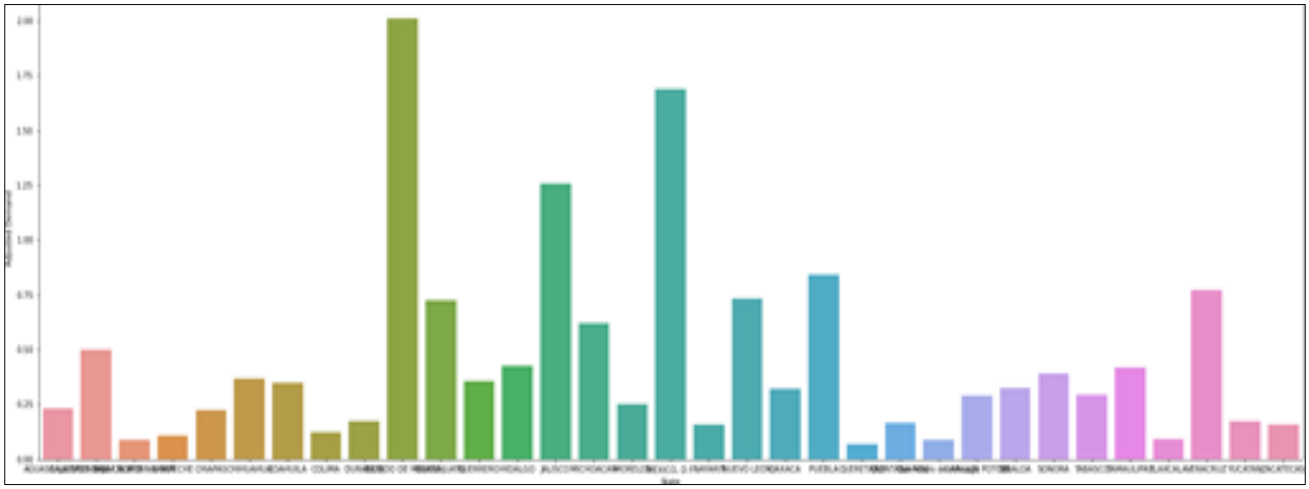

**Fig 5:** Geographic Demand Distribution Across Mexican States

frequently (maybe every week) but most of the demand per state is evenly distributed across weeks except few like Estado Mexico and taking an average yield an almost similar trend.

**Multiple linear regression**

Multivariate/Multiple Linear Regression (MLR) is a shared study between Statistics and Machine Learning which makes use of several predictor or independent variables as features and try to find a linear relationship of them to the predicted or dependent variable.

MLR is generally adopted against datasets where we have two or more predictor variables and we want to see if they factor into predicting or estimating the final predicted variable. In our case, the sales data not follow a time-series pattern thus assuming that the data is stationary across a period is not possible. For this very reason, we cannot resort to mini-batch training as upon applying that the training and testing accuracy take a huge dip due to less contextual knowledge during that iteration/epoch.

We skip data pre-processing, as most of the data has apt quantitative data-types helpful for a linear regression approach. We take up the cleaned data (from EDA) with all the relevant features (from correlation matrix observation) and normalize it using the linear scaling technique utilizing standard deviation and variance.

Then the OLS (ordinary least squares) baseline linear regression model is implemented as the earliest version. To see consistent results against regression, we played around with the shuffle and random seeding state to get a proper train-test splitting on every fresh run.

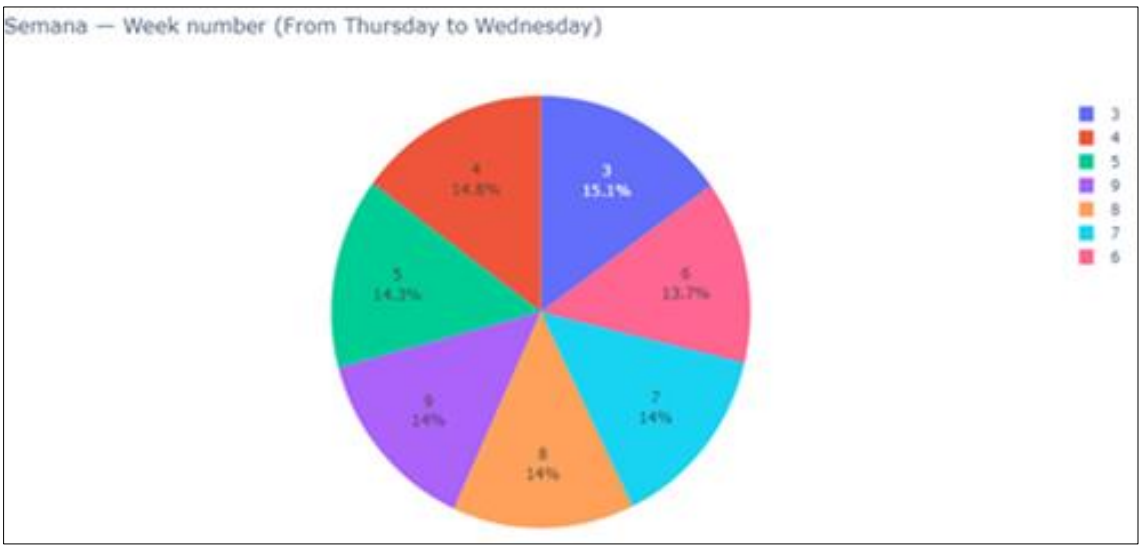


**Fig 6:** Weekly Sales Distribution Pattern Analysis

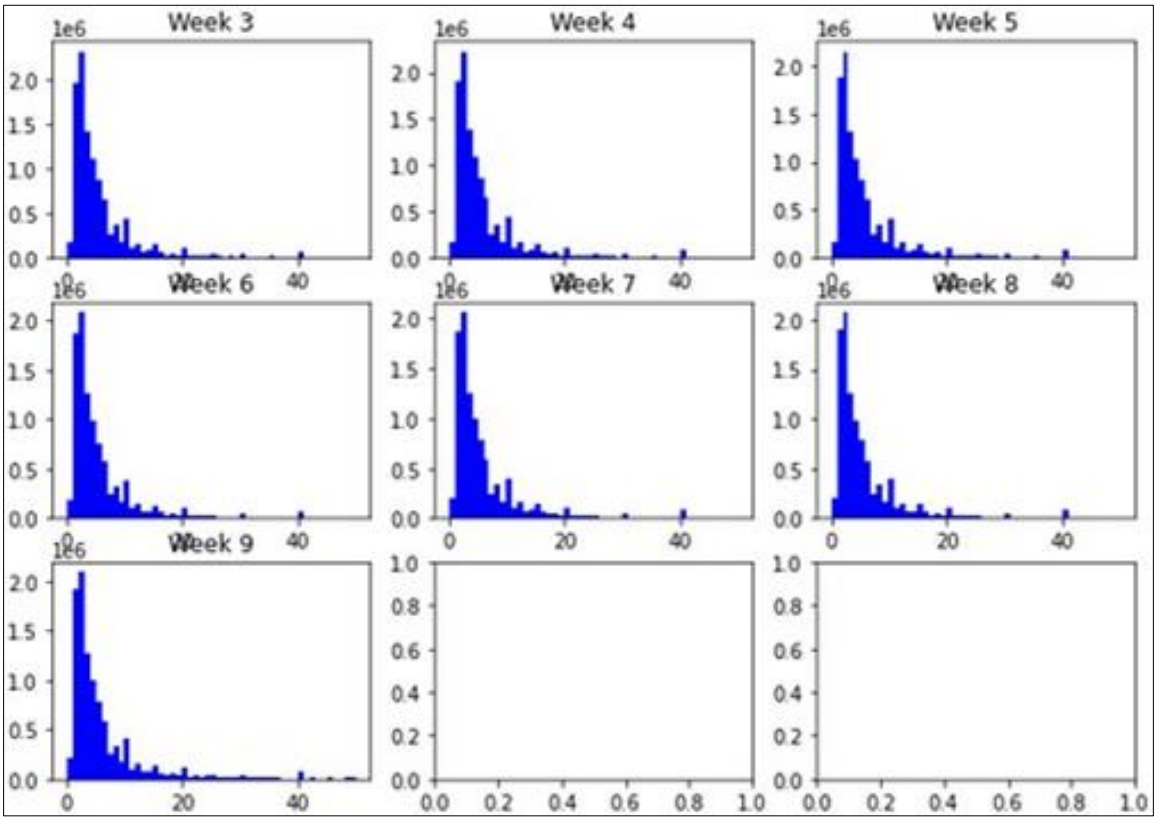


**Fig 7:** Demand Fluctuation Analysis Across 9-Week Period

To test out models we use the following metrics:

1) **Loss function** - RMSE (Root mean square error), which is the mean of the sum of square errors between the predicted and actual values in the dataset
2) **Accuracy/Improvement** - R2 score (measure of variance between the predicted and the actual value in the validation dataset)

The model results of the OLS MLR algorithm look like:

We can thus infer that our initial hypothesis that the columns partially factor into the adjusted demand (dependent variable) stands.

**Stochastic gradient descent regression**

Stochastic Gradient Descent Regression (SGD) has been widely used in day-to-day machine learning use cases due to its ability to work on large-scale sparse data. It is an iterative approach with picks up the input sample item randomly from the bigger sample space and adjust the gradient at each step, in a decreasing sequence governed by the learning rate parameter. It also introduces a lot of hyperparameters to control how the model can be run in different scenario as per the developer's taste.

In our scenario, because of the average performance from

| RMSE | R2 Score |
|---|---|
| 13.063 | 0.639 |

**Fig 8:** Multiple Linear Regression results

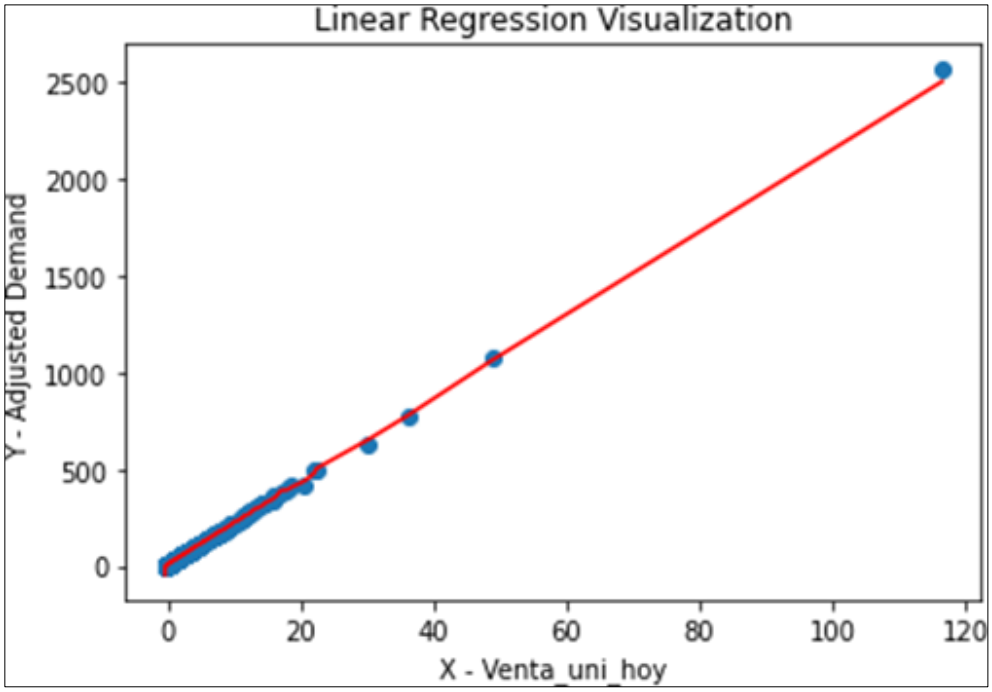


**Fig 9:** Sales (scatter) vs MLR prediction (line)

the MLR after longer computation cycles, we shifted to SGD in order to speed up the training time and also play around with different parameters to try and get a good accuracy against our training data.

The model results for the SGD regression algorithm look like:

General inference from this model is that it performs really well with **99%** accuracy but is very unreliable as it relies on probabilistic random picking of samples to train. But, once trained well, we can use it for accurate prediction against any new incoming data.

**Xtreme gradient boosting regression**

We proceed to implement the XGBoost Regressor on the input sample space to train and forecast the prediction. We use this approach due to two main reasons:

- Execution speed
- Model performance

For the data prepossessing steps, we performed a missing values analysis and removed irrelevant data columns and rows. For the rest, we filled them with zero or the mean of data based on general trend followed by the feature. Missing values were causing incorrect visualization of some features and thus this exercise was carried out. Then we have plotted a correlation matrix to check the feature correlation.

In terms of feature engineering, we created new columns by clubbing individual features. The features:

nAgencia, nRuta_SAK, nCliente_ID, nProducto_ID were made by grouping the features Agencia, Client_ID, Ruta_SAK, Producto_ID with respect to Semana (week num- ber) individually against adjust demand. Furthermore a feature named" lags" was created by grouping Semana, Cliente_ID, Producto_ID and taking mean of Demanda_uni_equil(Adjust Demand).

| RMSE | R2 Score |
|---|---|
| 1.750 | 0.993 |

**Fig 10:** Stochastic Gradient Descent results

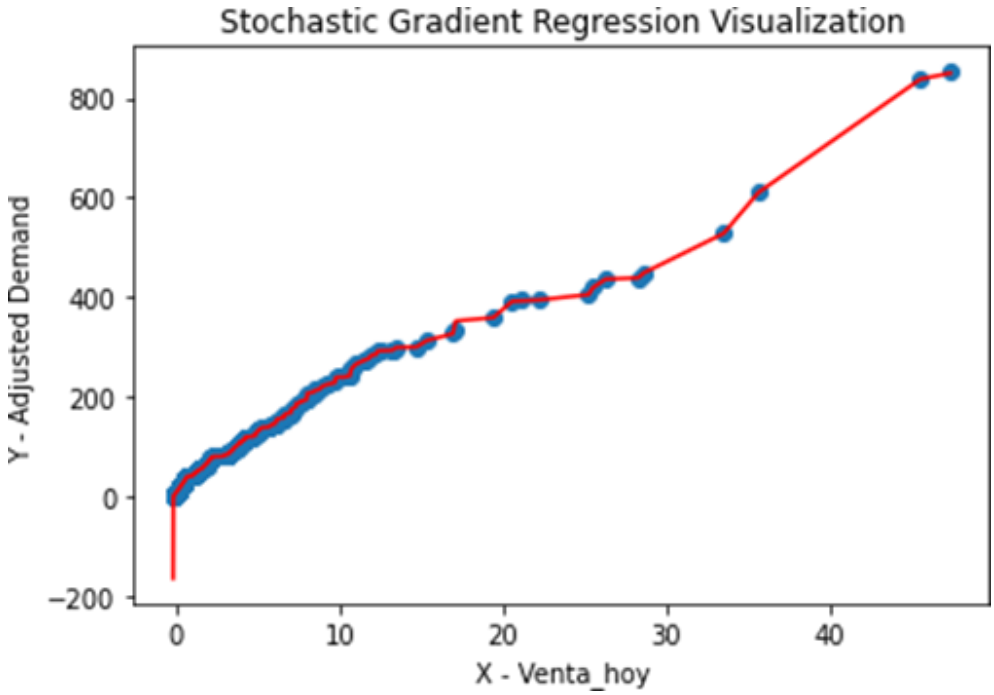


**Fig 11:** Sales in weight (scatter) vs SGD prediction (line)

We then applied boosting algorithm XGBoostRegressor for training and forecasting prediction with early stopping parameter set to 100 to avoid over-fitting. XGBoostRegressor gave out an R2 score of 0.70 and root mean squared error (RMSE) of
0.45. As per our stance, XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. This aligns with findings from Pavlyshenko [7], who demonstrated the effectiveness of machine learning models, including gradient boosting approaches, for sales time series forecasting.

**Random forest regression**

Random Forest (RF) is a Supervised learning algorithm that is based on the ensemble learning method from many Decision Trees. Random Forest is a bagging technique, so all calculations are run in parallel and there is no interaction between the Decision Trees when building them. RF can be used to solve both Classification and Regression tasks. The name" Random Forest" comes from the Bagging idea of data randomization (Random) and building multiple Decision Trees (Forest). The Random Forest algorithm searches for best features among a random subset of features which results in greater tree diversity and trades a higher bias for lower variance.

In the Regression case, one should implement Random Forest if:

- It is not a time series problem
- The data has a non-linear trend and extrapolation is not crucial

Since the Forecasting data model does not involve seasonality as inferred from the EDA, Random Forest Regression seems like an apt approach to predicting the forecasts. We started the implementation by using all features available in the train data set and executing the Random Forest Regression. To

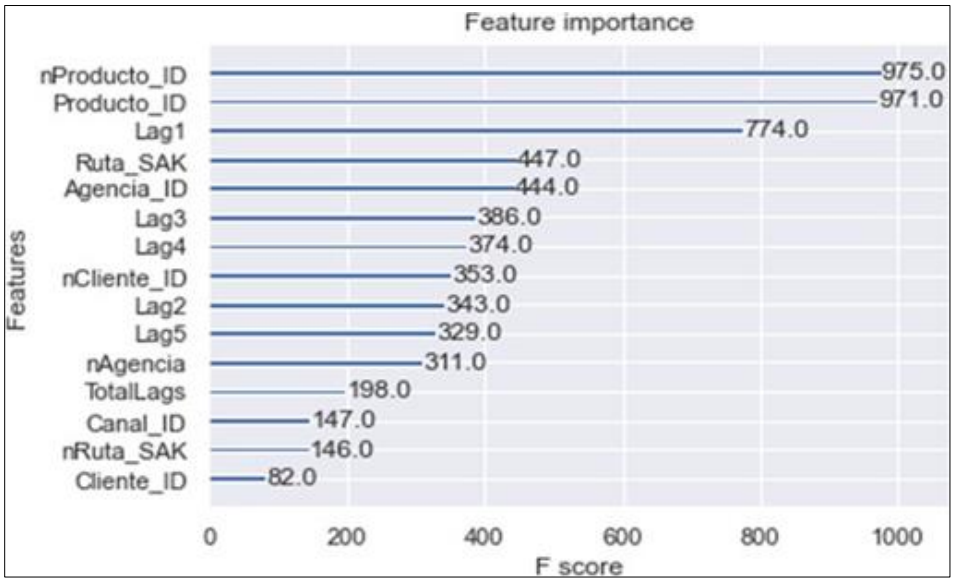


**Fig 12:** Feature Importance Coefficients

| RMSE | R2 Score |
|---|---|
| 0.45 | 0.70 |

**Fig 13:** XGBoost Regression results

Identify the initial hyperparameters we ran the GridSearchCV to identify initial set of parameters.

Model Tuning: We implemented the RF Regression with all features in the data barring the measures like Sales, Returns and their units. This was done as we intend to explore an alternative approach for the relationship of Adjusted Demand with the nominal predictors of the dataset. The initial run yielded results which were evaluated using Regression Metrics like RMSE and R2 score.

We then proceeded to implement Random Forest with varying max depth to test the performance requirements over the whole dataset. The then available features were then engineering into feature space 2 below based on the Feature importance obtained from the trained models. We then further experimented with additional features to develop better scores. The Depot ID was not contributing to the model accuracy and the wasn't part of the feature space 2. We therefore then decided to introduce the Town and state features as this would add another set of dimensions to diversify our analysis.

Our final feature spaces looked like:

Feature space 1(initial) = ['Sales Depot ID', 'Sales Channel ID', 'Route ID', 'Client ID', 'Product ID']

Feature space 2 = ['Sales Channel', Route ID, 'Product ID']

Feature space 3 = ['Town',' State', 'Sales Channel ID', 'Route ID,' Client ID, 'Product ID']

The results can be summarized as below:

## 4. Comparison

After implementing all the proposed models and the ones we changed during the roadmap, we found out that the order of best performing models was in this form:

a) Stochastic Gradient Descent Regression
b) XGBoost Regression
c) Multiple Linear Regression

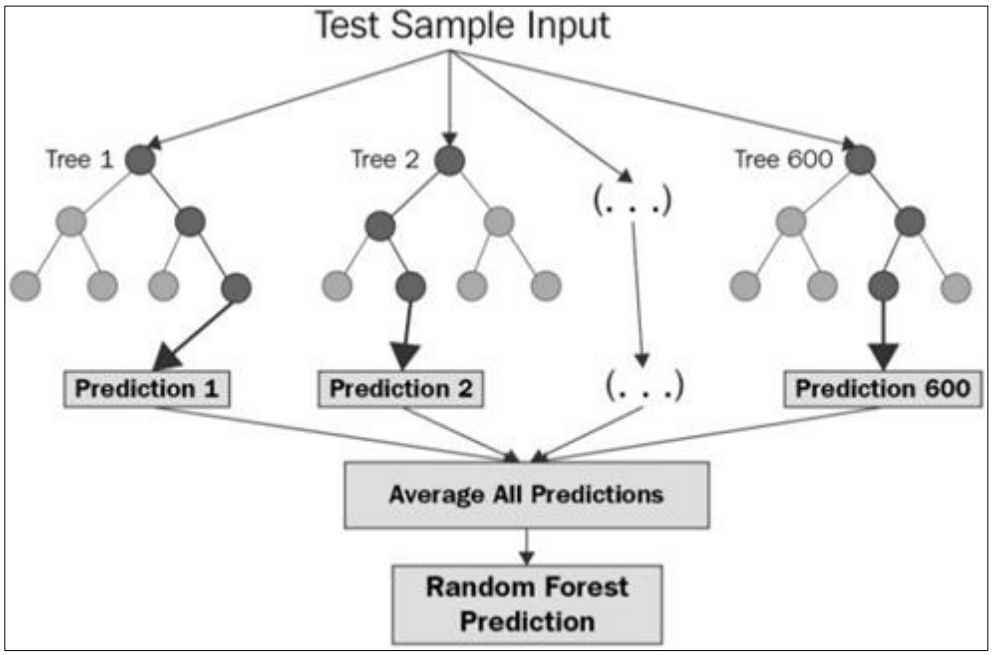


**Fig 14:** Random Forest Regression (Working)

| Models | RMSE | R2 Score |
|---|---|---|
| Feature Space 1 | 19.25 | 0.246 |
| Feature Space 2 | 47.95 | 0.498 |
| Feature Space 3 | 13.19 | 0.337 |

**Fig 15:** Random Forest Regression Scores

d) Random Forest Regression

The best performing model turned out to be the SGD with around 99% accuracy against the validation data and a model fitting score of 99% as well. The comparison of its predicted average demand across the weeks to the actual test data demand mean was almost the same. The only downside was the variability of this model outcomes because of its probabilistic nature in terms of the approach towards training. With every iteration of model run we got highly random and sometimes erroneous RMSE and R2 scores but on some hyperparameters where it worked well, the trained model could definitely be used as a great regression approach.

The second model in terms of accuracy and the best in terms of model training speed stood XGBoost Regression with R2 score coming out to be 0.70, thus **70%**. The model gave consistent output with various iterations and runs, while model parameter tweaking didn't yield any significant output changes.

Multiple linear regression (OLS) model gave the 3rd best accuracy with 64% while the least scoring algorithm was Random Forest regression with accuracy around **50%**. We suspected the data size and model computation limitations to be the culprits for giving such results for the RF multiple decision tree training approach. Considering the size and variance of the data, the RF models required higher estimators and more depth which in turn required higher computing capacities. Below is a visual representation of all the algorithms and their RMSE values and R2 scores (ranked high to low).

## 5. Future Directions

As per the future scope of this project we would like to address three major areas:

a) Data size constraints

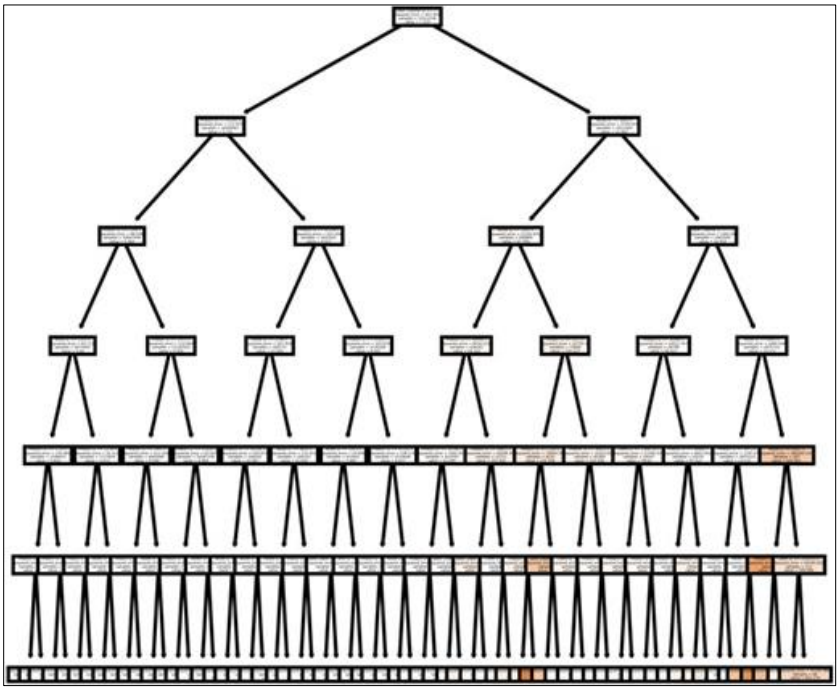

**Fig 16:** Random Forest Regression Tree (Feature Space 1)

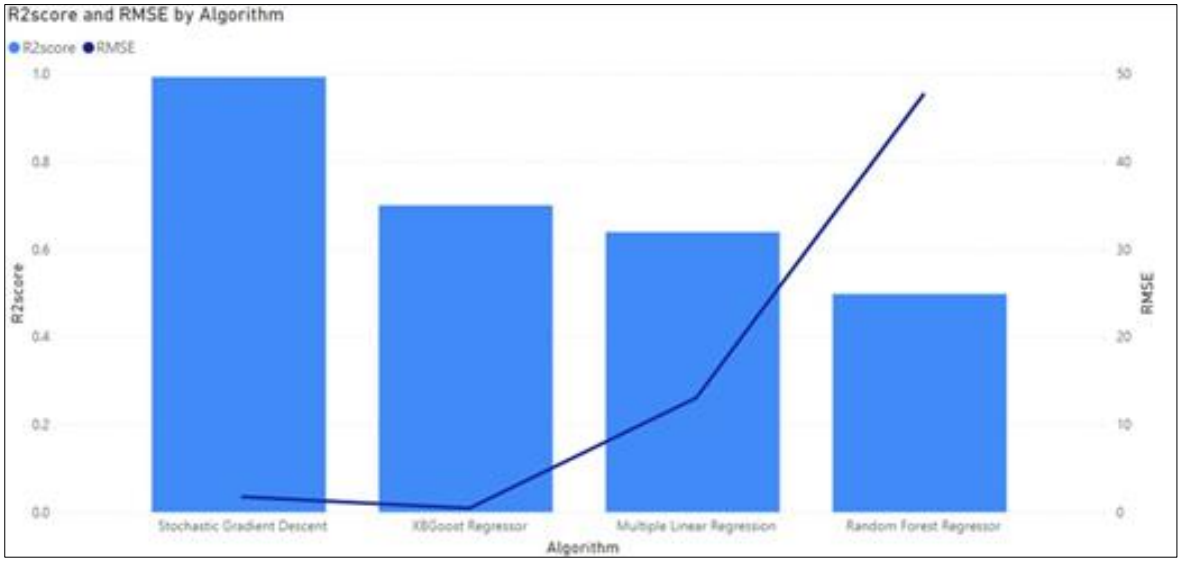


**Fig 17:** Comparison of algorithms

b) Model training constraints
c) Model Robustness

In terms of the data constraints, we plan to use better standardized big data frameworks like Spark/Hadoop that are built specifically to solve handling of large datasets. Leveraging cloud AI infrastructures like Vertex AI of Google and SageMaker of AWS can also prove useful to offload computation, saving time and memory for data related transformations and training and testing various models. On the level of code, we want to implement logic to read data in chunks and also explore mini-batch training to make the execution less computing intensive. Another piece of optimization that can be done is to prune the model or compress it thus reducing the training time (with a slight trade-off against the accuracy). To boost our model robustness we want to test our selected and fitted model against other (similarly cleaned) FMCG companies' data to identify further common traits that different SKUs have to offer and also look at recommendation models that also take in similar data and then perform relative hyper- parameter tuning to better output each product's demand and stock rate. We would want to implement a simpler version for online training and tune it to be resilient to scenarios of incoming outlier samples. With an iterative approach, we would also like to keep doors open to find more machine learning algorithms with similar prediction accuracy.
In further stages of implementation, we would recommend adding more data and features to the selected model and state-of-the-art ensemble techniques to achieve robustness and set clear identification of scalability that a firm might have to undergo to meet the demand-supply steady growth in terms of inventory planning. Future work could also explore the analysis of order-up-to-level inventory systems with compound Poisson demand as described by Babai *et al* [10], which could provide additional insights for inventory optimization.

## 6. Conclusion

In conclusion, the integration of machine learning techniques in demand-based inventory forecasting can significantly enhance the accuracy and efficiency of stock management systems. After performing adequate exploratory data analysis, going through visualization, running rigorous feature engineering and implementing several algorithms to tackle the inventory forecasting problem, we found that Stochastic gradient descent yielded the highest accuracy and a moderate training time to deal with a dataset containing millions of data points. Because it is not that reliable in terms of training and accommodating new incoming training samples, the fastest and most consistent algorithm turned out to be XGBoost with the accuracy against validation data to be around **70%**. We propose a deploy-able solution using this algorithm to be installed by firms looking for inventory forecasting against their SKUs.
Our findings align with previous research by Carbonneau *et al* [1], who demonstrated the application of machine learning techniques for supply chain demand forecasting, and Babai *et al* [5], who studied the accuracy of intermittent demand forecasting and its implications for inventory management.

## Data Availability

The Grupo Bimbo inventory demand dataset used in this study is publicly available through the Kaggle platform (https://www.kaggle.com/competitions/grupo-bimbo inventory-demand/data).

## Conflict of Interest

The author declares no conflict of interest in the preparation and publication of this research.

## 7. References

1. Carbonneau R, Laframboise K, Vahidov R. Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research. 2008;184(3):1140–1154.
2. Lolli F, Balugani E, Ishizaka A, Gamberini R, Rimini B, Regattieri A. Machine learning for multi-criteria inventory classification applied to intermittent demand. Production Planning & Control. 2019;30(1):76–89.
3. Chen I-F, Lu C-J. Sales forecasting by combining clustering and machine-learning techniques for computer retailing. Neural Computing and Applications. 2017;28:2633–2647.
4. Croston JD. Forecasting and stock control for intermittent demands. Journal of the Operational Research Society. 1972;23(3):289–303.
5. Babai MZ, Syntetos A, Teunter R. Intermittent demand forecasting: An empirical study on accuracy and the risk of obsolescence. International Journal of Production Economics. 2014;157:212–219.
6. Gijsbrechts J, Boute RN, Van Mieghem JA, Zhang D. Can deep reinforcement learning improve inventory management: Performance on dual sourcing, lost sales, and multi-echelon problems. 2019.
7. Pavlyshenko BM. Machine-learning models for sales time series forecasting. Data. 2019;4(1):15.
8. Bishop CM, Nasrabadi NM. Pattern recognition and machine learning. Vol. 4. Springer; 2006.
9. Conceição SV, Da Silva GLC, Lu D, Nunes NTR, Pedrosa GC. A demand classification scheme for spare part inventory model subject to stochastic demand and lead time. Production Planning & Control. 2015;26(16):1318–1331.
10. Babai MZ, Jemai Z, Dallery Y. Analysis of order-up-to-level inventory systems with compound Poisson demand. European Journal of Operational Research. 2011;210(3):552–558.