Retail Intelligence: Enhancing Demand Forecasting and Inventory Planning for Holidays

Namratha Nagathihalli Anantha
Master's in Data Science
Dept. Mathematical Science
Hoboken, USA
nnagathi@stevens.edu

Rohit Tiwari
Master's in Data Science
Dept. Mathematical Science
Hoboken, USA
rtiwari3@stevens.edu

Suvarsha Sai Punyala
Master's in Data Science
Dept. Mathematical Science
Hoboken, USA
spunyala@stevens.edu

***Abstract*—Demand forecasting plays a pivotal role in the domain of Supply Chain Management (SCM) within organizations, offering demand planners the ability to anticipate future trends. It is imperative to predict sales figures accurately, especially when sales patterns are influenced by various factors such as festival seasons like Thanksgiving, Diwali, Christmas, Ramzan, etc. characterized by unique seasonal promotions resulting in diverse sales trends. Similarly, products experiencing continuous growth or those with declining demand exhibit distinct sales patterns. To effectively forecast sales trends considering all of these variables, we will employ machine learning models, including Linear Regression, Decision Tree, Random Forest, XGBoost, and Feed-forward Neural Networks.**

Index- Machine learning, Supply Chain Management, Linear Regression, Decision Tree, Random Forest, XGBoost, and Multi-layer Perceptron Neural Networks.

I. Introduction

In the fast-paced realm of business, the ability to anticipate and adapt to changing consumer demands is a crucial factor that can define the success of enterprises. Demand forecasting stands as a keystone, providing organizations with the foresight needed to navigate the dynamic market. The ability of demand planners to anticipate future trends is a cornerstone in achieving operational efficiency and responsiveness. Accurate sales predictions are paramount, particularly when influenced by a myriad of factors, ranging from festive seasons such as Thanksgiving, Diwali, Christmas, etc. to special events like Labor Day, each characterized by unique promotional dynamics that shape diverse sales trends. Additionally, demand forecasting and planning might become complicated when we operate more than one selling point, there are different items in the stock, and some of them don't have sales data at all.

Machine learning gives forecasting models the capacity to analyze large data sets, spot trends, and gain knowledge from past performance. Machine learning models, as opposed to static forecasting techniques, adapt and get better with each new set of data, allowing companies to more precisely manage the challenges of demand prediction. Moreover, the sales trajectory of products is further complicated by continuous growth or declining demand, each exhibiting distinctive patterns. To address this complexity and enhance the precision of sales trend forecasts, we propose the utilization of advanced machine learning models. In this project, we will harness the predictive power of models such as Linear Regression, Decision Tree, Random Forest, XGBoost, and Feed-forward Neural Network, leveraging their capabilities to discern intricate patterns within the data landscape. Through the amalgamation of data-driven insights and cutting-edge machine learning techniques, our aim is to revolutionize demand forecasting, providing organizations with a strategic edge in the ever- evolving realm of Supply Chain Management. In the subsequent sections, we will explore specific machine learning techniques employed in demand forecasting, ranging to advanced predictive modelling. By embracing these cutting-edge technologies, organizations can not only meet current consumer demands more accurately but also position themselves to anticipate and adapt to future market trends with unprecedented precision.

II. Related Work

Different studies conducting on-demand forecasting models predicted sales calculated using Machine Learning (ML) regression models and methods with time series analysis for forecasting sales numbers using extended features. Azure Machine learning platform is used for that. Different regression techniques were applied to get a specific result. The fuzzy-neural network implemented for sales forecasting and demonstrated that this model's performance is better than traditional neural networks. An approach named Gray extreme machine learning with the Taguchi method which demonstrates better system performance than the performance of artificial neural networks. Forecasting for month electricity sales purposed in China uses clustering, regression, and time series analysis techniques. Time series analysis auto sales also carried out in China. The periodic effects were calculated by using the exponential weighted moving average. After that, considered effects and the calculated frequency are merged for the linear regression technique. ARMIMA and neural networks are combined for forecasting. Cloud computing-based forecasting system and applied time series analysis with the moving average method are also implemented for forecasting. Systems are installed on Azure cloud, used PHP programming language and MySQL database. Moving average methods are being implemented to forecast the sales for a long time in the literature. A forecasting engine based on genetic algorithm is also embedded in [1].

The key to success in today's business is controlling the retail supply chain. Predicting customer demand is very essential for supply chain management. The perfect prediction has an effective impact on earning a profit., storage., lost

profit., sales amount and consumer attraction. The article [2] will produce a new method- using machine learning that will help for accurate prediction. This method collects the previous data of a store and analyze those data. Gathering the important information process those data and get prepared for using in method. Applying related algorithms towards the process data. We know K-Nearest Neighbor, Support Vector Machine, Gaussian Nave Bayes, Random Forest, Decision Tree Classifier and regressions have recently used an algorithm for prediction. We collect real-life data from the market. This paper made with the combination of shop position, month and occasion on that month and other related data. Our country's geographical area has an impact on prediction, which we discuss in our research. Our model produces a tentative demand for a particular product. This estimation helps retails and their businesses. After making a data set and apply appropriate algorithms, we will find different results and accuracy of different used algorithms. Compare them with others, we find out Gaussian Nave Bayes has the best accuracy. This helps to estimate the accurate product demand for a shop.

Supply chain literature reveals that study of resilient supply chains and bullwhip effect (BE) have been receiving special attention during pandemic for supply chains with seasonal as well as nonseasonal demand components. The BE phenomenon has been detected in various industries and sectors, and causes multiple inefficiencies such as higher costs of producing more than needed, wastage and transportation costs. As a result, BE forecast is of great importance for academics and supply chain managers. Despite the multitude of studies that have emerged addressing this issue, the impact of the quality of dynamic forecasts on the BE has received limited coverage in the literature. Optimal dynamic forecasts of the demand could allow managers to mitigate the upstream amplification of orders (and thus the BE), as well as reduce unnecessary inventory costs. Order quantity and BE in a supply chain depend on the forecast of the future demand. Usually minimum mean square error (MMSE) forecasts of the future demand are obtained by fitting an appropriate seasonal auto-regressive moving average (ARMA) time series model. However, a major drawback of the MMSE forecasting method is that it does not provide the associated risk forecasts. In this paper, a simple yet effective machine learning demand forecasting approach without fitting any time series model is presented. Specifically, a novel data driven machine learning algorithm that bypasses traditional forecasting steps and allows forecast weights to be optimized by minimizing the one-step ahead forecast error sum of squares (FESS) is proposed. A novel stability metric of a supply chain is proposed as the risk adjusted forecast of the future demand. It is shown that the risk adjusted forecasts can be used to check whether a given supply chain is resilient. In order to be more resilient and competitive in the current market, business leaders around the world agree that it is necessary to modernize and make major changes to their supply chain strategies. Demand risk forecasts obtained by the proposed machine learning approach allow supply chain managers to enhance the forecasting power of the order quantity and construct more resilient supply chains. The performance of proposed approach is evaluated through numerical experiments using simulated data and weekly demand data of two products in article [3]. The results show that the performance of the proposed forecasts and risk adjusted forecasts of the future demand are better than the commonly used MMSE forecasts of the future demand.

Power demand forecasting with high accuracy is a guarantee to keep the balance between power supply and demand. Due to strong volatility of industrial power load, ultra-short-term power demand is difficult to forecast accurately and robustly. To solve this problem, this article proposes a Long Short-Term Memory (LSTM) network-based hybrid ensemble learning forecasting model. A hybrid ensemble strategy-which consists of Bagging, Random Subspace, and Boosting with ensemble pruning-is designed to extract the deep features from multivariate data, and a new loss function that integrates peak demand forecasting error is proposed according to bias-variance tradeoff. Experimental results on open data set and practical data set show that the proposed model in [4] outperforms several state-of-the-art time series forecasting models, and obtains higher accuracy and robustness to forecast peak demand.

Electricity is of great significance for national economic, social, and technological activities, such as material production, healthcare, and education. The nationwide electricity demand has grown rapidly over the past few decades. Therefore, efficient electricity demand estimation and management are required for better strategies planning, energy utilization, waste management, improving revenue, and maintenance of power systems. In this paper, we propose an empirical mode decomposition (EMD)- based deep learning approach which combines the EMD method with the long short-term memory network model to estimate electricity demand for the given season, day, and time interval of a day. For this purpose, the EMD algorithm decomposes a load time series signal into several intrinsic mode functions (IMFs) and residual. Then, a LSTM model is trained separately for each of the extracted IMFs and residual. Finally, the prediction results of all IMFs in [5] are combined by summation to determine an aggregated output for electricity demand. To demonstrate the applicability of the proposed approach, it is applied to electricity consumption data of city Chandigarh. Furthermore, the performance of the proposed approach is evaluated by comparing the prediction results with recurrent neural network (RNN), LSTM, and EMD-based RNN (EMD+RNN) models.

III. Our Solution

A. Description of data set

The sample data set considered for this study was from Kaggle, a subsidiary of Google, is an online community of data scientists and machine learning. It hosts an enormous collection of data sets that can be used to build AI models, publish data sets, and work with other data scientists and machine learning engineers.

The data set consists of sales value of a variety of products across 40 stores of Walmart chain within the timeframe of 02/05/2010 and 11/01/2012. It consists of the following fields:

- Store - the store number
- Date - the week of sales
- Weekly_Sales - sales for the given store

- Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non holiday week
- Temperature - Temperature on the day of sale
- Fuel_Price - Cost of fuel in the region
- CPI – Prevailing consumer price index
- Unemployment - Prevailing unemployment rate

Below is a snippet of the data headers and the sample data:

	store	date	weekly_sales	is_holiday	temperature	fuel_price	cpi	unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

Fig 1. Data Headers and sample data

During the initial pre-processing of the data set, no duplicates and null values were found. The date column contains the values in DD/MM/YYYY format. The columns of year, quarter, month, week and day of the week were created by extracting the information from the date column. Basis the values in the extracted quarter column, seasons of Winter, Summer, Spring and Fall were created and stored in 'seasons' column. The data set with added columns is as below:

store	date	weekly_sales	is_holiday	temperature	fuel_price	cpi	unemployment	year	quarter	season	month	month_name	week	day_of_week
1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	8.106	2010	1	Winter	2	February	5	Friday
1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	8.106	2010	1	Winter	2	February	6	Friday
1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	2010	1	Winter	2	February	7	Friday
1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	2010	1	Winter	2	February	8	Friday
1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	8.106	2010	1	Winter	3	March	9	Friday

Fig. 2. Data set with added Date, Day and Seasons column

Data Analysis: Basis their attributes, the data columns can be divided as containing numerical and categorical values.

- Numerical Columns: Temperature, Fuel Price, Unemployment and CPI
- Categorical Columns: is holiday, year, season, month_name and day_of_week

Plotting the Numerical Columns in a histogram, we get the visuals below:

Basis these graphs, we can conclude that weekly sales data distribution is right skewed, which indicates the rise in weekly sales over particular point of time such as holidays/weekends. The CPI (Consumer Price Index) and Fuel Price have a bimodal distribution while Temperature and Unemployment have normal distribution. While analyzing the individual fields, critical observations were made in the column values of holiday flag and day of the week as below:

The value 'Friday' predominates the other values in the column 'day of the week' as all the weekly readings of the

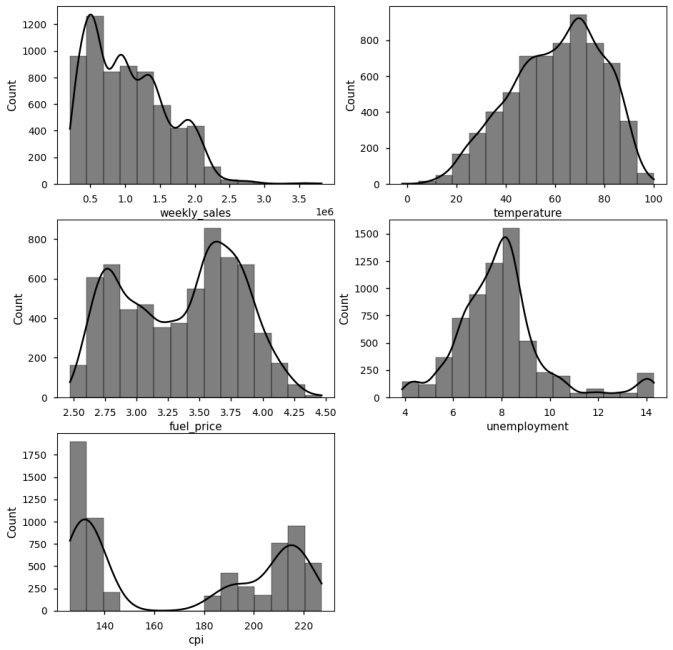

Fig. 3. Histogram of numerical data values

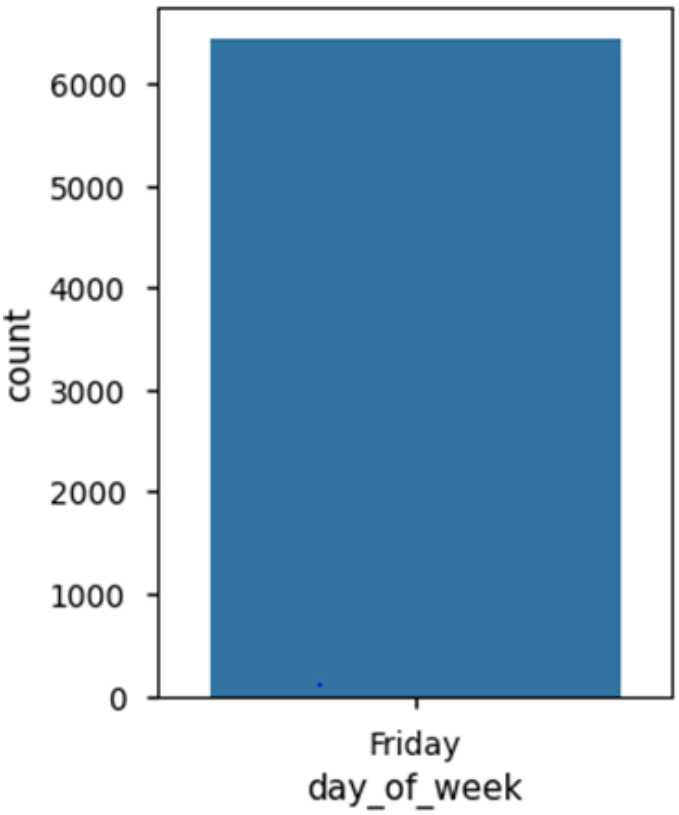

Fig 4. Days of the week

data set seems to be registered on the Friday. Hence the model would be biased to the class of 'Friday'. Thus, to avoid underfitting by our machine learning models, we need to drop the column 'day of the week'.

With respect to the Is Holiday flag, we can see that the average sales on a holiday increase significantly as compared

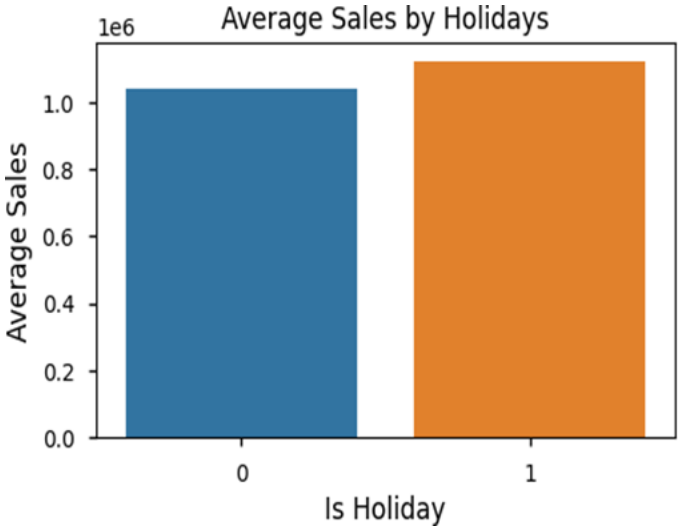

Fig. 5. Average Sales by Holidays

to the sales on a normal day increasing the importance of demand forecast. Comparisons could also be drawn between numerical values in the columns of fuel price, temperature, CPI and unemployment with the expected values of weekly sales.

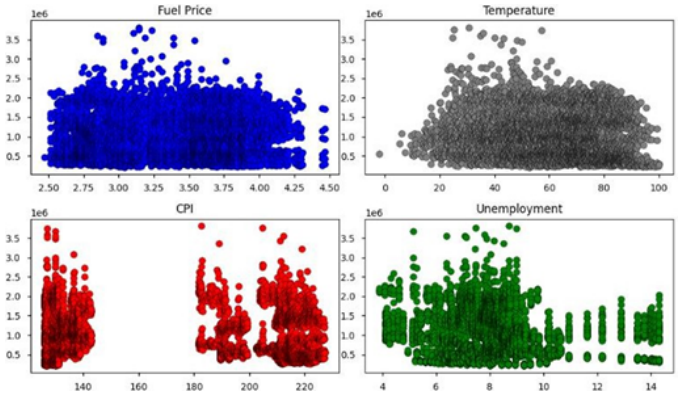

Fig. 6. Numerical Features vs Weekly Sales

We could observe an almost flat trend between fuel price and temperature with respect to weekly sales. Therefore, we can draw a conclusion that variations in the fuel prices and unemployment rate did not impact the weekly sales figures. However, the variation in the attributes of CPI and unemployment impacted the weekly sales figures as the sales were high when CPI was either low or significantly higher. This explains the demographics of the customers visiting the retails chain. Also, variations in the unemployment rate impacted the weekly sales as lower unemployment increased the purchase capacity of the shoppers.

Data Correlation: Based on the numerical and categorical data fields in the sample data set, we employed the two most common utilized methods to analyze data correlation:

- Pearson Correlation Coefficient: The Pearson correlation coefficient (r) is the most common way of measuring a linear correlation. It is a number between – 1 and 1 that measures the strength and direction of the relationship between two variables. A -1 means there is a strong negative correlation and +1 means that there is a strong positive correlation. A 0 means that there is no correlation. The formula for Pearson Correlation Coefficient (r) between two variables X and Y is given by:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

where n is the number of data points,
x and y are the variables,
$\sum$ denotes the sum over all data points.

Here, n is the number of data points, x_i and y_i are the individual data values for variables X and Y respectively.
By convention, when the p-value is less than 0.001: we say there is strong evidence that the correlation is significant. p-value is less than 0.05: there is moderate evidence that the correlation is significant. p-value is less than 0.1: there is weak evidence that the correlation is significant. p-value is greater than 0.1: there is no evidence that the correlation is significant. We used this to our advantage to analyse the numerical valued columns. The observations were as follows:

  - The Pearson Correlation Coefficient between temperature and weekly sales is -0.0638 with a P-value of 0.0000
  - The Pearson Correlation Coefficient between fuel price and weekly sales is 0.0095 with a P-value of 0.4478
  - The Pearson Correlation Coefficient between unemployment and weekly sales is -0.1062 with a P-value of 0.0000
  - The Pearson Correlation Coefficient between cpi and weekly sales is -0.0726 with a P-value of 0.0000

Hence, a strong correlation was found between temperature- weekly sales, unemployment-weekly sales and CPI-weekly sales as their p values were less than 0.001. On the contrary, the p- value of fuel price-weekly sales was found to be large and hence, the value of weekly sales is independent of fuel prices.

- Cramer's V correlation: Cramér's V is a measure of association between two categorical variables. It is an extension of the chi-squared test for independence and provides a value between 0 and 1, where 0 indicates no association and 1 indicates a strong association.
Cramer's V formula for a contingency table is given by:

$$V = \sqrt{\frac{\chi^2}{n \cdot \min(k-1, r-1)}}$$

where:

χ^2 is the chi-squared test statistic,
n is the total number of observations,
k is the number of columns in the table,
r is the number of rows in the table.

Our observations were as follows:

- Cramér's V for is holiday vs. weekly sales: 1.0000
- Cramér's V for year vs. weekly sales: 1.0000
- Cramér's V for season vs. weekly sales: 1.0000
- Cramér's V for month name vs. weekly sales: 1.0000
- Cramér's V for day of week vs. weekly sales: nan

Hence, all the attributes, except for 'day of the week' had a strong correlation with the weekly sales value. Therefore, the column day of the week was dropped from consideration in the sample data set basis the obtained results.

Outlier Elimination: Outliers in the numerical valued columns of temperature, fuel price, unemployment and consumer price index were eliminated using visual inspection technique. Graphs of the individual fields were plotted as below:

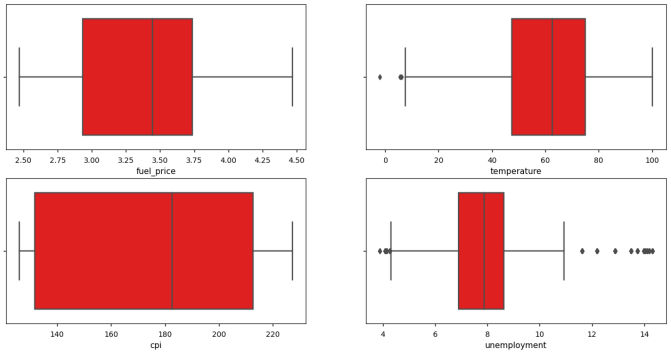

Fig. 7. Outlier Identification

Basis the scale and density of the plotted values, outliers were identified and eliminated from the data set. Post outlier elimination, the data values in the fields are as below:

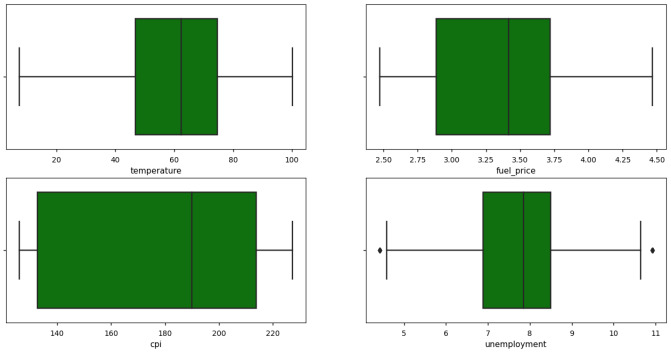

Fig. 8. Post-Outlier Elimination

Data Transformation: Since the data contains both the numerical and categorical data fields, two seperate encoding methods were used.

Standard Scalar: A Standard Scaler is a preprocessing technique used to standardize the features of a dataset. Standardization involves transforming the data in a way that the mean of each feature becomes 0 and the standard deviation becomes 1. This is achieved by subtracting the mean and dividing by the standard deviation for each data point. The numerical data fields of 'temperature', 'fuel price', 'unemployment' and 'CPI' were encoded using Standard Scalar

Binary Encoder: In binary encoding, each category is assigned a unique binary code. The length of the binary code depends on the number of categories. For example, if you have n categories, you would need $\lceil \log_2(n) \rceil$ bits to represent each category uniquely. The categorical data fields of 'is holiday' and 'season' were encoded using a binary encoder.

The data is now ready for machine learning models to be implemented.

B. Machine Learning Algorithms

1) **Linear Regression:**
Linear Regression is a fundamental and widely used technique in machine learning, specifically in the domain of supervised learning. It's a type of regression algorithm that models the relationship between a dependent variable (also called the target or response) and one or more independent variables (features or predictors) by fitting a linear equation to the observed data. The basic idea is to find the best-fitting linear relationship that minimizes the difference between the predicted and actual values of the dependent variable. This relationship is represented by the equation:

$$Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_nX_n$$

where:

- Y is the dependent variable,
- X1, X2,. . . ,Xn are the independent variables,
- b0 is the y-intercept,
- b1, b2,. . . ,bn

are the coefficients representing the impact of each independent variable on the dependent variable Linear Regression is interpretable, easy to implement, and computationally efficient. However, it assumes a linear relationship between the input features and the output, which may not always hold in real-world scenarios. In cases where the relationship is non-linear or more complex, other regression techniques or advanced models may be considered.

2) **Decision Tree:** A Decision Tree is a popular machine learning algorithm used for both classification and regression tasks. It is a tree-like structure where each node represents a decision based on the value of a particular feature, and each leaf node represents the predicted output label or value. Decision Trees are easy to understand and interpret, making them valuable for both beginners and experts in machine learning.

Decision Trees are a fundamental building block for ensemble methods like Random Forests and Gradient Boosted Trees, which aim to address some of the limitations of individual Decision Trees. Decision Trees can handle both categorical and numerical features. The decision tree algorithm selects the best feature to split the data at each decision node.

3) **Random Forest:** Random Forest is an ensemble learning algorithm that operates by constructing a multitude of decision trees during training and outputs the mode of the classes (for classification problems) or the mean prediction (for regression problems) of the individual trees. It is a powerful and versatile algorithm that tends to perform well across various types of data and problem domains. Random Forest builds multiple decision trees during training. Random Forest tends to be more robust to over-fitting compared to individual decision trees. Random Forest can provide a measure of feature importance based on how frequently a feature is used to split the data across all trees. Training individual trees can be done in parallel, making Random Forest scalable and efficient for large data sets.

4) **XGBoost:** XGBoost (eXtreme Gradient Boosting) is a powerful and popular machine learning algorithm that belongs to the family of gradient boosting methods. It is known for its efficiency, speed, and high performance in various machine learning tasks, particularly in structured/tabular data and competitive data science challenges. XGBoost supports cross-validation to evaluate model performance and tune hyper-parameters effectively. XGBoost is an implementation of the gradient boosting framework, which builds a predictive model by combining the predictions of multiple weak models, typically decision trees. XGBoost incorporates L1 (Lasso) and L2 (Ridge) regularization terms in the objective function to control over-fitting. XGBoost is designed to be highly scalable and can take advantage of parallel and distributed computing. XGBoost can be used for both classification and regression problems.

5) **Feed Forward Neural Network:** A feed-forward neural network, also known as a multi-layer perceptron (MLP), is a fundamental architecture in machine learning and deep learning. Comprising an input layer, one or more hidden layers, and an output layer, this network processes information in a unidirectional flow. The input layer receives features of the data, which then traverse through the hidden layers where weighted sums are computed and activation functions introduce non-linearity. The final output layer produces predictions or classifications. During training, the network adjusts weights and biases through the back-propagation algorithm, aiming to minimize the discrepancy between predicted and actual outputs, as quantified by a chosen loss function. Feed-forward neural networks are widely used for tasks such as regression and classification, making them a cornerstone in the field of artificial intelligence.

C. Implementation Details

We have implemented the below 5 machine learning models. Beginning with the simpler model of Linear Regression, we moved towards complex models of Decision Tree, Random Forest, XGBoost and Feed-Forward Neural Network.

We employed hyperparameter tuning in each of these algorithms to optimize the model's paramters, thus improving its performance. Hyperparameter tuning is the process of systematically searching for the best set of hyperparameters for a machine learning model. To achieve this, we employed the technique of Grid Search from Scikit-learn.

GridSearch is an approach to hyperparameter tuning that involves specifying a grid of hyperparameter values and evaluating the model performance for each combination of values within the grid. GridSearchCV performs cross-validation along with grid search. Cross-validation is a technique where the dataset is split into multiple folds, and the model is trained and evaluated on different subsets of the data. This helps in obtaining a more robust estimate of the model's performance.

We used two performance parameters to calculate the accuracy of our models:

1) Root Mean Square Error: Root Mean Square Error (RMSE) measures the average magnitude of the errors between predicted and actual values. The RMSE is particularly useful because it provides a measure of how well a model's predictions align with the true values in the same unit as the target variable. The Root Mean Square Error (RMSE) is calculated using the following formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

n is the number of observations,

y_i is the actual value of the target variable for the i-th observa

$\hat{y}_i$ is the predicted value of the target variable for the i-th obse

2) R2 Squared Error: R2 score, also known as the coefficient of determination, represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 0 indicates that the model does not explain the variability of the response data, and 1 indicates a perfect fit. The R-squared (coefficient of determination) is calculated using the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where:

n is the number of observations,

y_i is the actual value of the target variable for the i-th observa

$\hat{y}_i$ is the predicted value of the target variable for the i-th obse

$\bar{y}$ is the mean of the actual values.

The implementation steps for the models is as below.

1) **Linear Regression:** Linear Regression is chosen here for its simplicity, interpretability, and efficiency in capturing linear relationships between input features and the target variable. It will serve as a baseline model for regression tasks and is particularly suitable as the relationship between variables can be adequately represented by a linear equation. Additionally, its computational efficiency makes it a pragmatic choice for large data sets.
The initial parameter used to train the model is 'poly_feat_degree'. The parameter 'poly_feat_degree' likely refers to the degree of the polynomial features used in conjunction with linear regression. Polynomial features enable the model to capture nonlinear relationships by transforming the input features. For example, if 'poly_feat_degree' is set to 2, the model will include quadratic terms.
In the context of hyperparameter of a linear regression with polynomial features, common hyperparameters include the degree of the polynomial ('poly_feat_degree'). After hyperparameter tuning, the best parameter estimated for the training data set was 3.

We also utilized Cross-validation to help prevent overfitting or underfitting. Grid search was used to search through different combinations of hyperparameters. A k-fold cross-validation is used where the data set is divided into k equally-sized folds. Here, we used the default value of k = 5 for cross-validation. Cross-validation scores provided an average performance metric across the folds.

The implementation steps of the model involves:

a) Constructing a Linear Regression model.
b) Model evaluation through training data, yielding Root Mean Squared Error and r2 score.
   - Normalized Root Mean Squared Error: 0.137
   - R-Square score Training: 24.26%
c) Generating a Distribution Plot of predicted values versus actual values in the training data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Training Data."

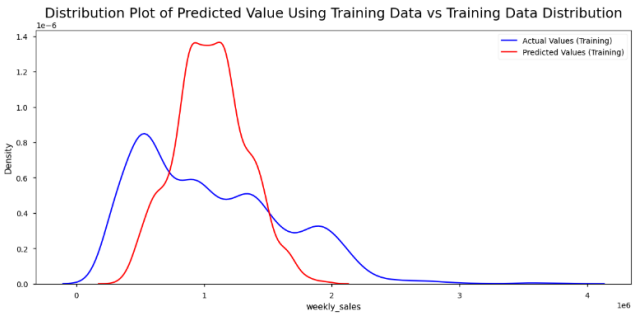

Fig. 9. Training Values vs predicted Training Values

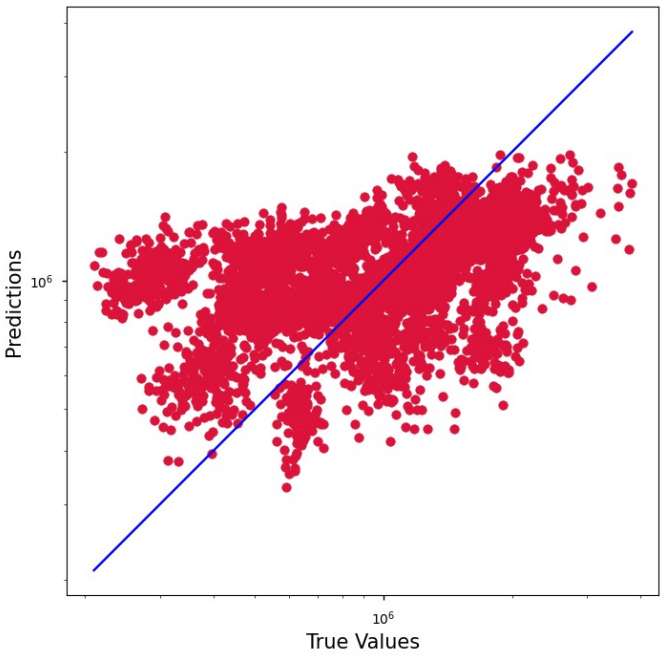

Fig. 10. Scatter Plot: Training Values vs Predicted Training Values (before hyperparameter tuning)

d) To address the lack of correlation between variables and the target, we enhance complexity by incorporating polynomial features before modeling. Model tuning to identify optimal parameters and achieve the best score.
e) Re-evaluation post-tuning to assess training accuracy, resulting in:
   - Normalized Root Mean Squared Error:0.018
   - R-Square score Training: 98.66%
f) Plotting reveals an improved model that better captures the training data set.

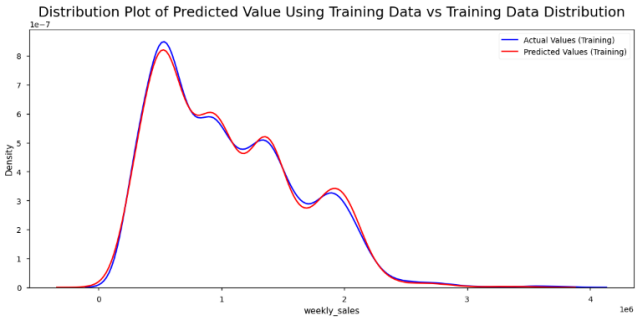

Fig. 10. Training Values vs Predicted Training Values (after hyperparameter tuning)

g) Employing cross-validation to determine mean and standard deviation, yielding:
   - Cross Validation Scores: [0.95955817 0.96588319 0.96974753 0.96201577 0.96514597]
   - Mean of Scores: 96.45%
   - Standard Deviation of Scores: 0.00347

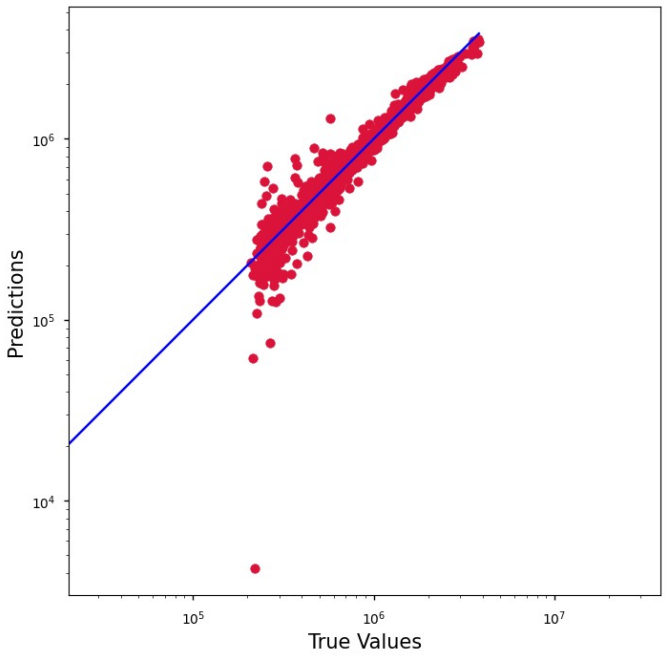

Fig. 11. Scatter Plot: Training Values vs Predicted Training Values (after hyperparameter tuning)

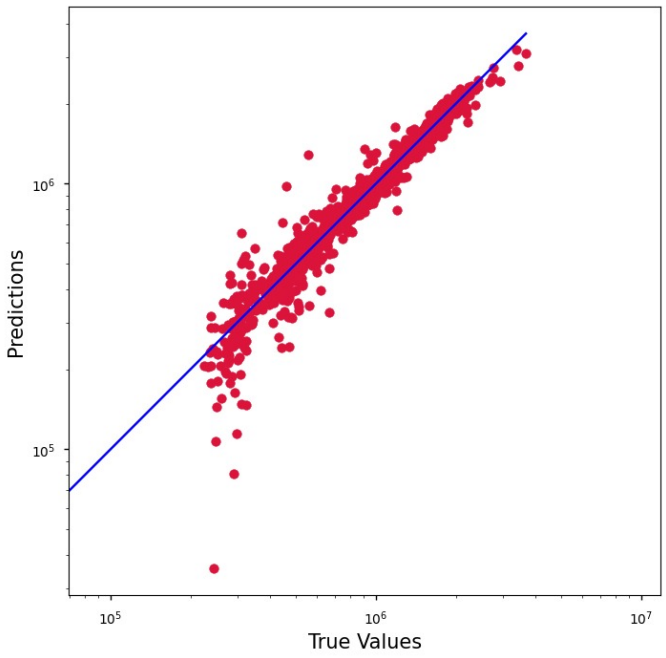

Fig. 1. Fig. 13. Scatter Plot: Testing Values vs Predicted Testing Values

h) Testing the model to assess accuracy, resulting in:
- Normalized Root Mean Squared Error: 0.0279
- R-Squere score Training: 97.13%

i) Plotting predicted values versus actual values in the testing data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Testing Data."

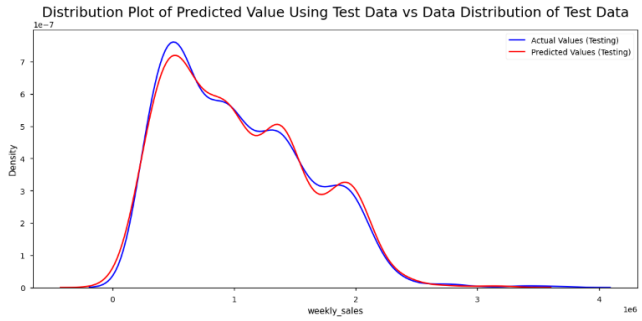

Fig. 12. Testing Values vs Predicted Testing Values

The achieved accuracy using the Linear Regression model is 97.13%.
The Linear Regression models depict a good accuracy with the degree of polynomial as 3. However, Linear regression models assume a linear relationship between variables, and they may not capture complex non-linear relationships present in the data. Also, Multi-collinearity occurs when independent variables are highly correlated with each other. This can lead to unstable coefficient estimates, making it challenging to interpret the individual contributions of each variable. So, we move on to implement complex models for better robustness and accuracy.

2) **Decision Tree:** Decision trees are chosen for their interpretability, ease of implementation, and ability to handle both classification and regression tasks. They mimic human decision-making processes, breaking down complex decisions into a series of simpler ones. Additionally, decision trees can handle non-linear relationships and interactions between features, making them versatile for various types of data sets. The initial parameters used to train the model are 'max_depth' and 'min_samples_split'.
- 'max_depth': This parameter defines the maximum depth of the tree. It controls the complexity of the model, preventing it from overfitting to the training data.
- 'min_samples_split': This parameter sets the minimum number of samples required to split an internal node. It helps control the growth of the tree and prevents the creation of nodes that only capture noise.

Graphing the training data with varying values of these parameters can illustrate their impact.
Hyperparameter tuning involves finding the best combination of hyperparameters for optimal model performance. Common techniques include grid search or random search.After hyperparameter tuning, the best parameters, such as the optimal max_depth and min_samples_split, are determined.They are 'max_depth': 13, 'min_samples_split': 30. These values enhance the model's predictive accuracy on unseen data.
Cross-validation is used to assess the model's gen-

eralization performance. Cross-validation scores are obtained by splitting the training data into subsets for training and validation.

The implementation steps of the model involves:

a) Constructing a Decision Tree Regressor model.

b) Model evaluation through training data, yielding Root Mean Squared Error and r2 score.
- Normalized Root Mean Squared Error: 0.0
- R-Square score Training: 100.0%

c) Generating a Distribution Plot of predicted values versus actual values in the training data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Training Data."

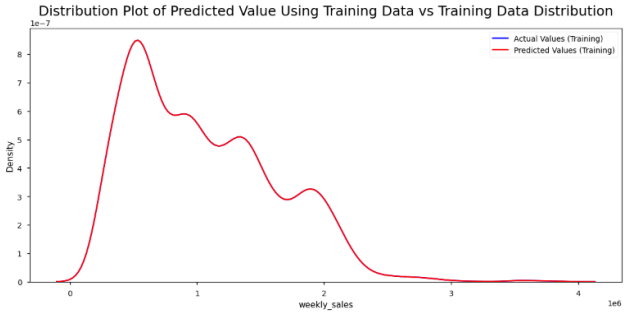

Fig. 14. Training Values vs Predicted Training Values

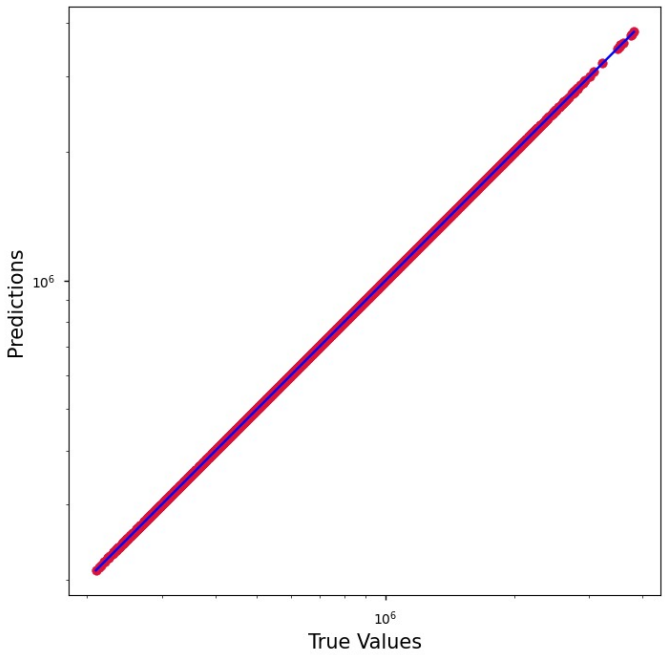

Fig. 15. Scatter Plot : Training Values vs Predicted Training Values(without tuning)

d) To address the lack of correlation between variables and the target, we enhance complexity by incorporating polynomial features before modeling. Model tuning to identify optimal parameters and achieve the best score.

e) Re-evaluation post-tuning to assess training accuracy, resulting in:
- Normalized Root Mean Squared Error: 0.0334
- R-Square score Training: 95.52%

f) Plotting reveals an improved model that better captures the training data set.

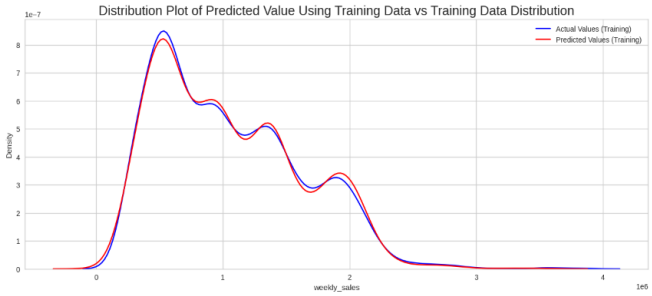

Fig. 16. Scatter Plot : Training Values vs Predicted Training Values(after tuning)

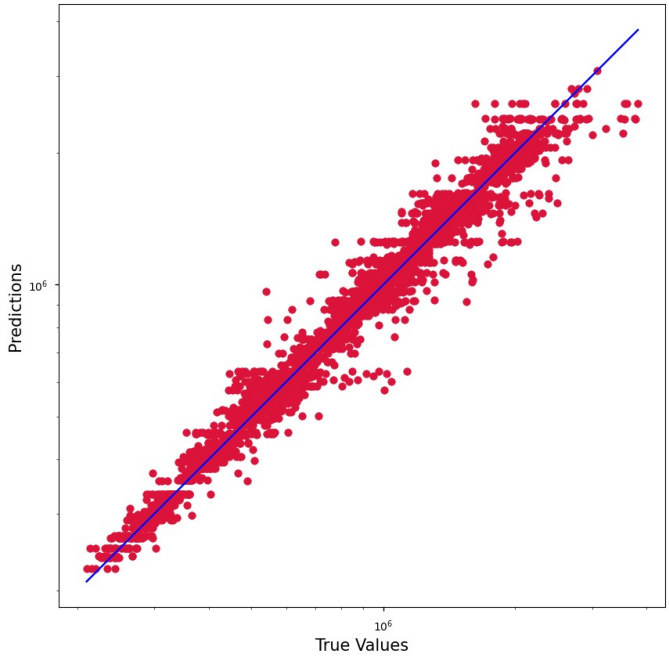

Fig. 17. Scatter Plot: Training Values vs Predicted Training Values (after hyperparameter tuning)

g) Employing cross-validation to determine mean and standard deviation, yielding:
- Cross Validation Scores: [0.91626155 0.93674064 0.9278914 0.92195092 0.88405018 0.91331679 0.92466785 0.90900891 0.89630786 0.94763668]
- Mean of Scores: 91.78%
- Standard Deviation of Scores: 0.0176

h) Testing the model to assess accuracy, resulting in:

- Normalized Root Mean Squared Error: 0.0489
- R-Square score Training: 91.22%

i) Plotting predicted values versus actual values in the testing data, with ”weekly_sales” on the x-axis and ”density” on the y-axis, titled ”Distribution Plot of Predicted Value vs Actual Value using Testing Data.”

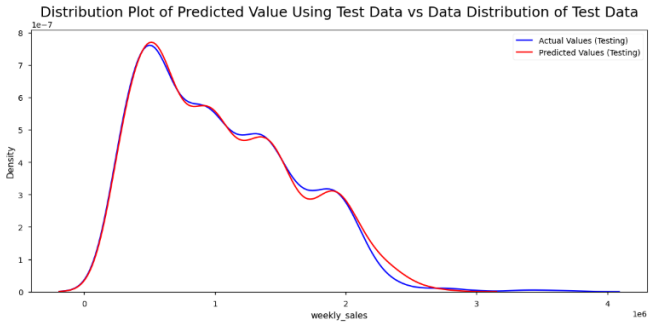

Fig. 18. Testing Values vs Predicted Testing Values

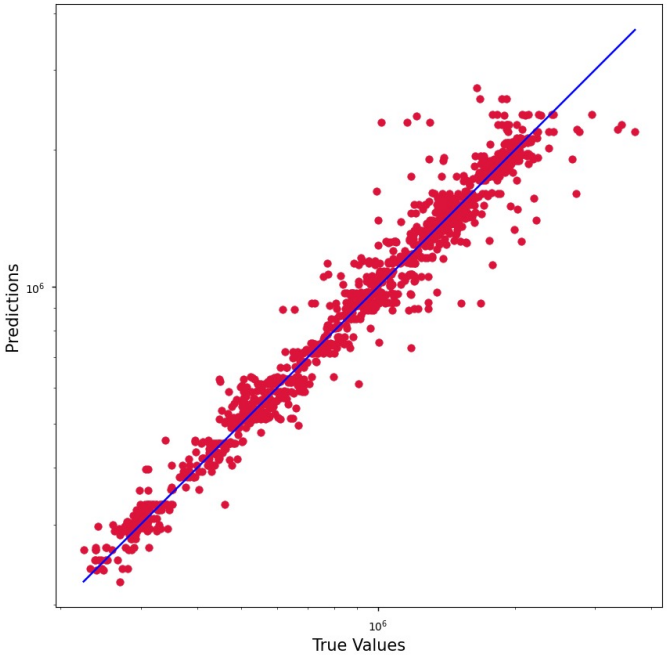

Fig. 19. Scatter Plot: Test Values vs Predicted Test Values

The achieved accuracy using the decision tree model is 91.22%.
As per the scatter plot, the model is overfitting. So, the Random Forest model is chosen as the next model to obtain better accuracy.

3) **Random Forest:** Random Forest is often chosen for its robustness, versatility, and high predictive accuracy. It is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. Random Forest is less prone to overfitting compared to individual decision trees, and it can handle large data sets with high dimensionality. It is also effective in capturing complex relationships within the data and handling both numerical and categorical features. The initial parameters used to train the model are ’n_estimators’, ’max_features’, ’max_depth’, ’min_samples_split’, ’min_samples_leaf’.

- ’n_estimators’: The number of trees in the forest.
- ’max_features’: The maximum number of features considered for splitting a node.
- ’max_depth’: The maximum depth of each tree in the forest.
- ’min_samples_split’: The minimum number of samples required to split an internal node.
- ’min_samples_leaf’: The minimum number of samples required to be at a leaf node.

The best parameters are the ones that result in the highest cross-validation score. These values are determined through the hyperparameter tuning process. For example, after performing grid search or randomized search, the combination of ’n_estimators’: [25, 50, 70, 100], ’max_features’: [None, 1,3,5,8], ’max_depth’: np.arange(2,15), ’min_samples_split’: [2,5,10], and ’min_samples_leaf’: [1, 2, 4] that maximizes the cross-validation score would be considered the best set of hyperparameters for the Random Forest model.
The implementation steps of the model involves:

a) Constructing a Random Forest Regressor model.

b) Model evaluation through training data, yielding Root Mean Squared Error and r2 score.

- Normalized Root Mean Squared Error: 0.0143
- R-Square score Training: 99.18%

c) Generating a Distribution Plot of predicted values versus actual values in the training data, with ”weekly_sales” on the x-axis and ”density” on the y-axis, titled ”Distribution Plot of Predicted Value vs Actual Value using Training Data.”

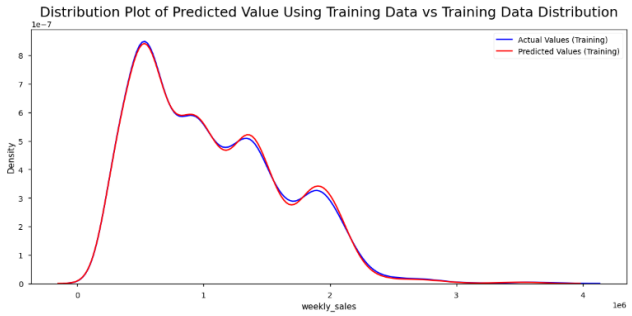

Fig. 20. Training Values vs Predicted Training Values (before parameter tuning)

d) To address the lack of correlation between variables and the target, we enhance complexity by incorporating polynomial features

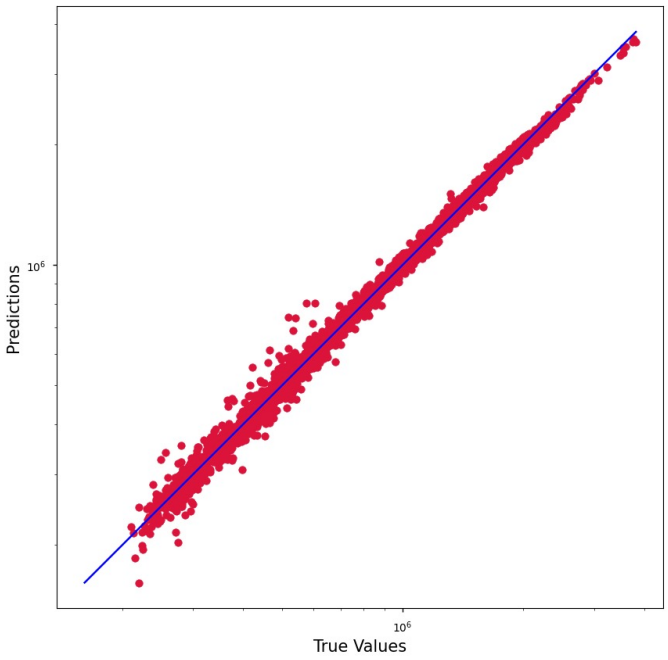

Fig. 21. Scatter Plot : Training Values vs Predicted Training Values (before hyperparameter tuning)

before modeling. Model tuning to identify optimal parameters and achieve the best score.

e) Re-evaluation post-tuning to assess training accuracy, resulting in:
- Normalized Root Mean Squared Error: 0.0152
- R-Square score Training: 99.06%

f) Plotting reveals an improved model that better captures the training data set.

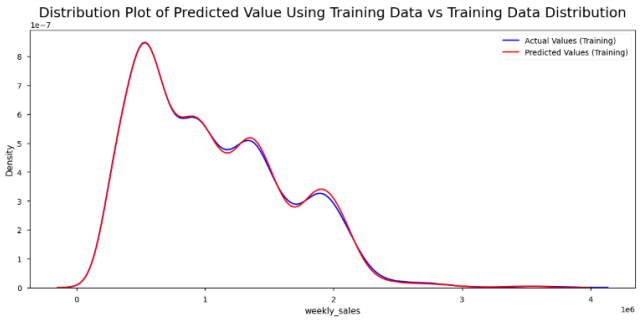

Fig. 22. Training Values vs Predicted Training values (after hyperparameter tuning)

g) Employing cross-validation to determine mean and standard deviation, yielding:
- Cross Validation Scores: [0.94372224 0.95224094 0.94774706 0.94307089 0.92997839 0.95047328 0.95215728 0.93792037 0.93657662 0.96320551]
- Mean of Scores: 94.57%
- Standard Deviation of Scores: 0.00905

h) Testing the model to assess accuracy, resulting in:

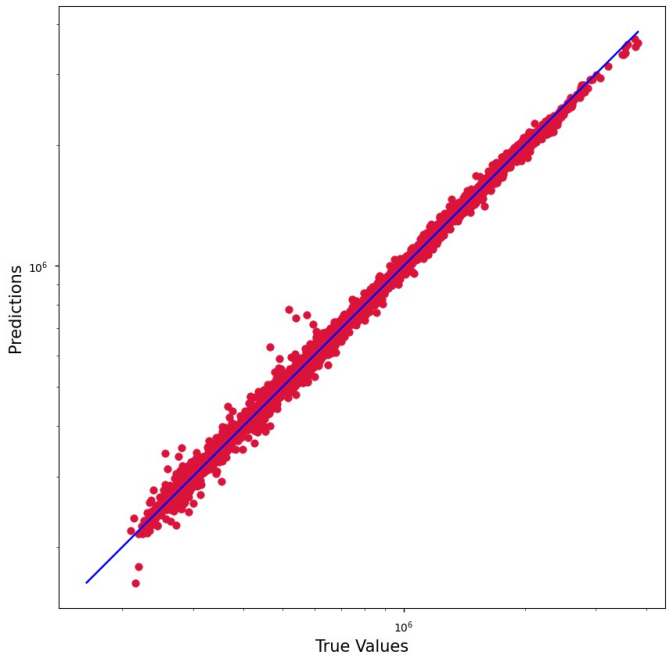

Fig. 23. Scatter Plot: Training Values vs Predicted Training Values (after hyperparameter tuning)

- Normalized Root Mean Squared Error: 0.03767
- R-Square score Training: 94.8%

i) Plotting predicted values versus actual values in the testing data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Testing Data."

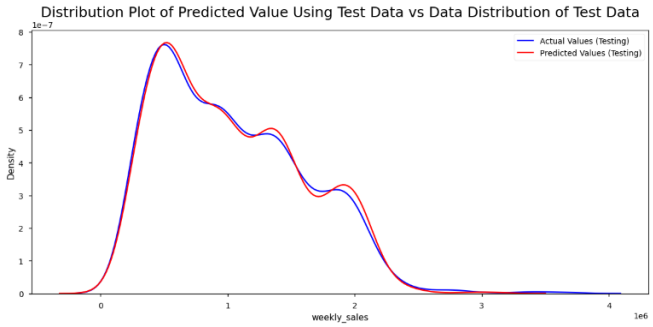

Fig. 24. Testing Values vs Predicted Testing values

The achieved accuracy using the Random Forest model is 94.8%. Training a large number of decision trees in Random Forest can be computationally expensive, especially for large datasets like this case. The algorithm requires more time and resources to compute as compared to simpler models, especially after multiple iterations for hyperparameter tuning. Also, basis the scatter plot, we can observe that the model captures noise or specific patterns in the training data that do not generalize well to test data set. Hence, we move forward to the XGBoost model to obtain better accuracy.

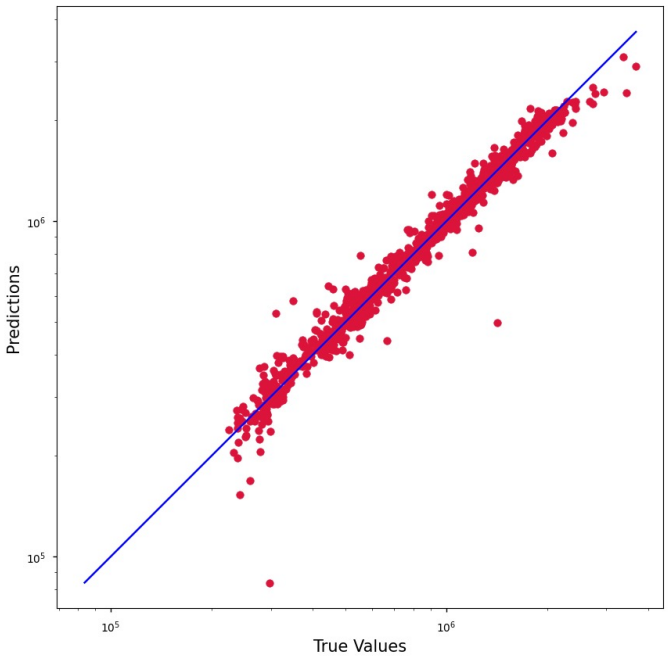

Fig. 25. Scatter Plot : Testing Values vs Predicted Testing values

4) **XGBoost:** XGBoost is chosen due to its exceptional performance in machine learning tasks, especially in the context of structured/tabular data. XGBoost stands out for its efficiency, scalability, and ability to handle complex relationships within the data. Its gradient boosting framework and regularization techniques contribute to robust predictive modeling, making it a popular choice in various competitions and real-world applications.
The initial parameters used to train the model are 'n_estimators' and 'max_depth'.
   - 'n_estimators': This parameter defines the number of boosting rounds or decision trees to be built during the training process. A higher number of estimators may increase model complexity but could lead to overfitting.
   - 'max_depth': Specifies the maximum depth of each decision tree. Controlling the tree depth helps prevent overfitting. A shallow tree (lower 'max_depth') captures simpler patterns, while a deeper tree can capture more complex relationships.

   Hyperparameter tuning involves optimizing the model's performance by searching for the best combination of hyperparameters. Common hyperparameters include learning_rate, subsample, and colsample_bytree. Cross-validation is used to assess model performance on different subsets of the training data. After hyperparameter tuning, the best parameters for the XGBoost model are determined by the grid search which are given by, max_depth': 7, 'n_estimators': 60.
   The implementation steps of the model involves:
   a) Constructing an XGBoost Regressor model.
   b) Model evaluation through training data, yielding Root Mean Squared Error and r2 score.
      - Normalized Root Mean Squared Error: 0.00868
      - R-Squere score Training: 99.7%
   c) Generating a Distribution Plot of predicted values versus actual values in the training data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Training Data."

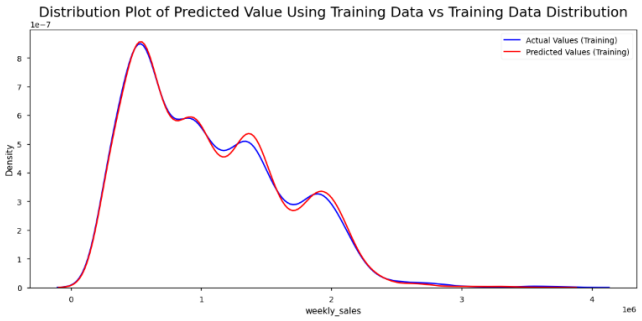

Fig. 26. Training value vs Predicted Training value (before hyperparamter tuning)

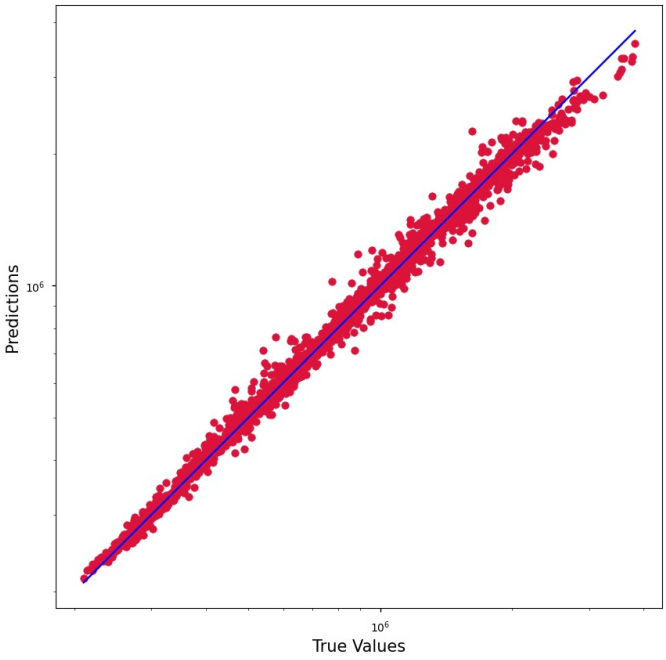

Fig. 2. Fig. 27. Scatter Plot : Training Values vs Predicted Training Values (before hyperparameter tuning)

   d) To address the lack of correlation between variables and the target, we enhance complexity by incorporating polynomial features before modeling. Model tuning to identify optimal parameters and achieve the best score.
   e) Re-evaluation post-tuning to assess training accuracy, resulting in:

- Normalized Root Mean Squared Error: 0.00837
- R-Square score Training: 99.72

f) Plotting reveals an improved model that better captures the training data set.

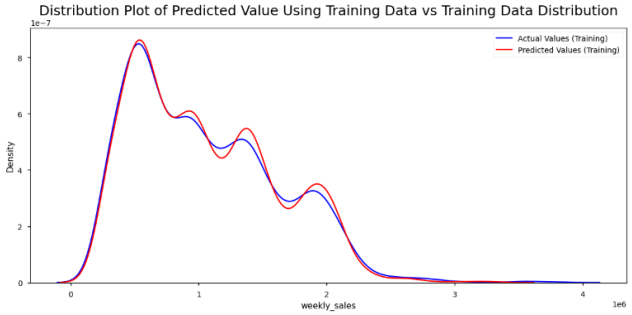

Fig. 28. Training value vs Predicted Training value (after hyperparamter tuning)

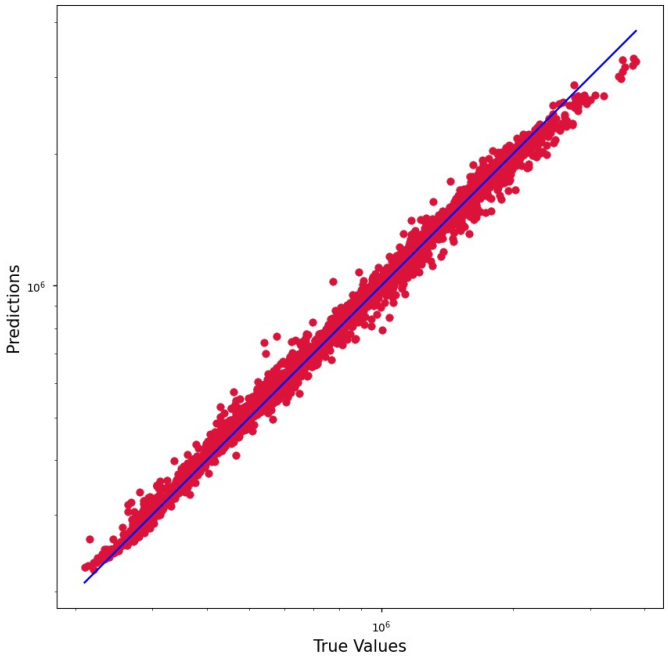

Fig. 3. Fig 29. Scatter Plot: Training Values vs Predicted Training Values (after hyperparameter tuning)

g) Employing cross-validation to determine mean and standard deviation, yielding:
- Cross Validation Scores: [0.96349866 0.97354279 0.96550554 0.96408454 0.96671613 0.97188208 0.96104139 0.96091998 0.96999453 0.9694661]
- Mean of Scores: 96.67%
- Standard Deviation of Scores: 0.00419

h) Testing the model to assess accuracy, resulting in:
- Normalized Root Mean Squared Error: 0.0258
- R-Square score Training: 97.55%

i) Plotting predicted values versus actual values in the testing data, with "weekly_sales" on the x-axis and "density" on the y-axis, titled "Distribution Plot of Predicted Value vs Actual Value using Testing Data."

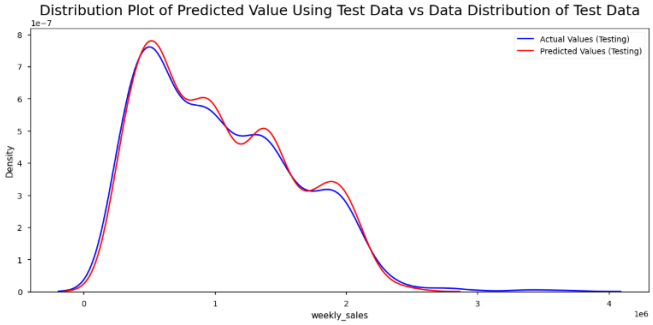

Fig. 30. Testing Values vs Predicted Testing values

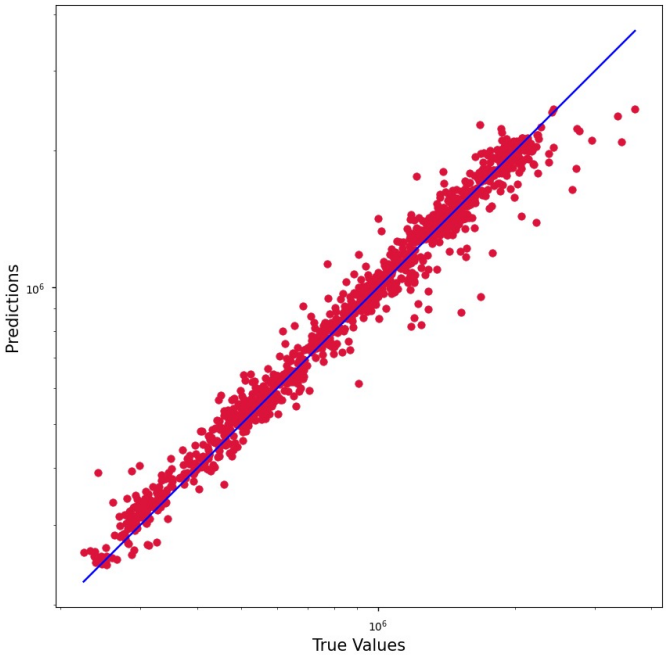

Fig. 4. Fig. 31. Scatter Plot: Testing Values vs Predicted Testing values

The achieved accuracy using the XGBoost model is 97.55%.

As per the scatter plot, this model performs the best amongst the other regression models. However, XGBoost is computationally expensive and time-consuming, especially for large datasets such as retail chain data as it creates large number of trees and deep trees. Training a complex ensemble of trees requires substantial computational resources. Hence, we move on to implementing a Feed Forward neural network.

5) **Feed-Forward Neural Network:** A Feedforward Neural Network (FNN) is a fundamental type of artificial neural network where information moves in only one direction—forward—from the input layer through the hidden layers (if any) to the output layer. There are no cycles or loops in the network, which

makes it a straightforward and simple architecture. It's called "feedforward" because the information flows through the network in a forward direction without any feedback loops.
Here, the data contains both numerical and categorical features. As an input to the Neural network, we need to encode them using appropriate encoders for each of the data field type. For columns with numerical features, we use Standard Scalar for encoding whereas for columns with categorical features, we use One Hot Encoder to encode all the values. The neural network has one input layer for numerical features and multiple input layers for each categorical feature. For the given data set, we applied ReLU as activation function for the input and hidden layers and Linear activation function for the output layer. To compile the model, we used Stochastic Gradient Descent as an optimizer and mean square error as a metric to improve performance. The performance parameters achieved are as below:

- Normalized Root Mean Squared Error: 0.0187
- R-Square score Testing: 96.47%

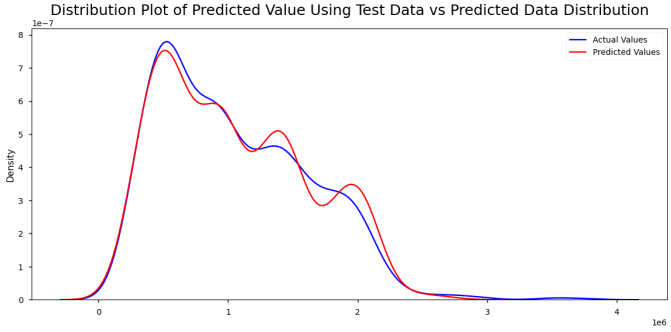

Fig. 32. Testing Values vs Predicted Testing values

As visible in the scatter plot, the Feed Forward neural network performs optimally with robustness as compared to many other regression models. It also handles noise and variations in the parameters better than the regression models without any additional coding required. The inclusion of non-linear activation functions in hidden layers allows the feedforward network to model complex relationships in the given data set. This enables it to capture non-linear patterns that linear models were struggling to represent. Feedforward network automatically learn relevant features from the input data during the training process. This can be especially beneficial when dealing with high-dimensional data, as the network learns to extract meaningful representations.

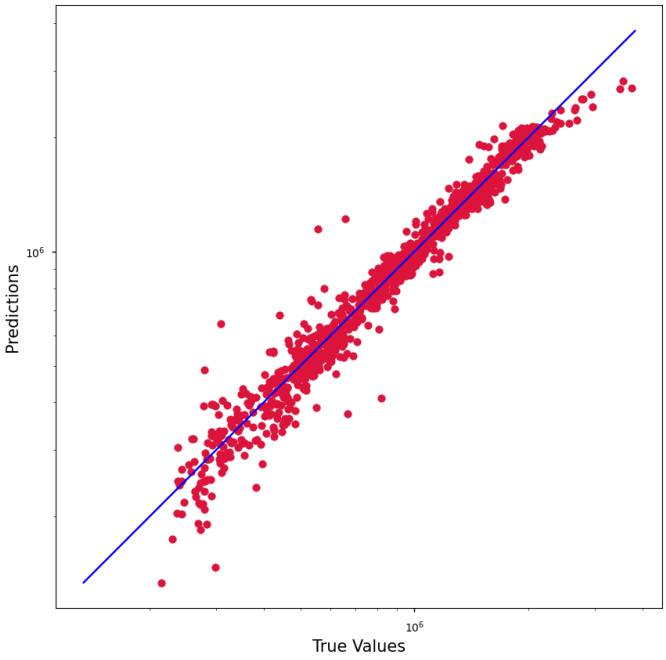

Fig. 33. Scatter Plot: Testing Values vs Predicted Testing values

IV. Comparison

In the study, we have implemented teh simpler model of Linear regression as the first model. The accuracy of the trained model on the test data was 97.13%. Apart from implementing this simple algorithm, we also implementedd multiple models with increasing complexity such as Decision Tree, Random Forest, XG Boost, and Feed Forward Neural Network. seasons.

Metrics	Linear Regression	Decision Tree	Random Forest	XGBoost	Feedforward Neural Network
Normalized RMS	0.0279	0.0489	0.03767	0.0258	0.0187
R-square score	97.13%	91.22%	94.8%	97.55%	96.47%
Mean of Scores	96.45%	91.78%	94.57%	96.67%	-
Standard Deviation of Scores	0.00347	0.0176	0.00905	0.00419	-

Fig.34. Performance Comparison of Models

Comparing the implemented models, We observed that the Root Mean Square Error for Feedforward network is the lowest, indicating its robustness, whereas complex models of XGBoost and Linear Regression display exceptional performance on the test data set with R-square accuracy of 97.55% and 97.13% respectively. However, XGBoost can better handle variations in the features as the data set increases in size and complexity as compared to Linear Regression. The Decision Tree model overfitted on both training and test data set hence, displaying a mediocre R-Square accuracy of 91.22%, hence it is not a recommended model for retail demand forecasting. The random forest algorithm overfitted before hyperparameter tuning and hence, for future scope of increasing the data set size for prediction, the intricate hyperparameter tuning would be required for every iteration, Thus, basis our observation, it is recommended to use XGBoost and Feed-Forward Neural

Network as a preferred algorithms for Retail Demand Forecasting handling the sudden surge in demand during the holiday season.

V. Future Directions

This demand forecasting may include possible future work investigating forecasting at different levels of aggregation, ex: at a store or market unit level.

- Real-time Forecasting: Exploring and implementing real-time demand forecasting capabilities, allowing organizations to make instantaneous adjustments based on the most recent data and market trends.
- Hybrid Models: Explore the integration of traditional forecasting methods with machine learning models to harness the strengths of both approaches for more accurate predictions.
- Incremental Learning: Implement techniques for incremental learning, allowing models to adapt to new data without retraining the entire model, thereby improving efficiency.
- Multi-Modal Data Fusion: Integrate information from various modalities, such as text, images, and time-series data, to capture a more comprehensive understanding of demand-influencing factors.

VI. Conclusion

Correspondingly with the technological development, the importance of the demand forecast has been increasing day by day for the enterprises. This study has shown how machine learning algorithms can transform businesses perceive, anticipate, and react to market demand. It became clear from exploring a variety of machine learning approaches from complex ensemble models that these techniques perform better than conventional ones. We have achieved significant advancements in accurately forecasting demand, which is crucial for effective inventory management and resource allocation. Our best performing machine learning model of XGBoost demonstrated a commendable accuracy of 97.55% on the test data set in predicting demand, as evidenced by its low mean absolute error and R-squared values. Similarly, our robust nueral network model of Feed-forward displayed the test data accuracy of 96.47%. This indicates that our models are robust and reliable in capturing the underlying patterns in demand data.

Implementing these machine learning models in the retail chain ensures accurate demand forecast based on the historical data and thus, allowing for better resource planning and allocation. Looking ahead, the success of the demand forecasting depends on its ongoing dedication to innovation. Future research would examine developing user-friendly interfaces, integrating external data sources, and forecasting in real-time.

Demand forecasting with machine learning integrated becomes a strategic need as industries navigate a more tightly connected and complicated global market, in addition to being a technological one. It provides companies with the tools they need to manage uncertainty, keep ahead of market dynamics, and eventually optimize their operations for long-term success in a rapidly changing business environment.

References

1) M. A. Khan et al., "Effective Demand Forecasting Model Using Business Intelligence Empowered with Machine Learning," in IEEE Access, vol. 8, pp. 116013-116023, 2020, doi: 10.1109/ACCESS.2020.3003790.
2) M. A. I. Arif, S. I. Sany, F. I. Nahin and A. S. A. Rabby, "Comparison Study: Product Demand Forecasting with Machine Learning for Shop," 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 2019, pp. 171-176, doi: 10.1109/SMART46866.2019.9117395.
3) M. E. Hoque, A. Thavaneswaran, S. S. Appadoo, R. K. Thulasiram and B. Banitalebi, "A Novel Dynamic Demand Forecasting Model for Resilient Supply Chains using Machine Learning," 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 2021, pp. 218-227, doi: 10.1109/COMPSAC51774.2021.00040.
4) M. Tan, S. Yuan, S. Li, Y. Su, H. Li and F. He, "Ultra-Short- Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning," in IEEE Transactions on Power Systems, vol. 35, no. 4, pp. 2937-2948, July 2020, doi: 10.1109/TPWRS.2019.2963109.
5) J. Bedi and D. Toshniwal, "Empirical Mode Decomposition Based Deep Learning for Electricity Demand Forecasting," in IEEE Access, vol. 6, pp. 49144-49156, 2018, doi: 10.1109/ACCESS.2018.2867681.
6) P. K. Bala, "Decision tree based demand forecasts for improving inventory performance," 2010 IEEE International Conference on Industrial Engineering and Engineering Management, Macao, China, 2010, pp. 1926-1930, doi: 10.1109/IEEM.2010.5674628.
7) A. Leenatham and P. Khemavuk, "Demand Forecasting Using Artificial Neural Network Based on Quantitative and Qualitative Data," 2020 1st International Conference on Big Data Analytics and Practices (IBDAP), Bangkok, Thailand, 2020, pp. 1-6, doi: 10.1109/IBDAP50342.2020.9245614.