

SALES FORECASTING USING MACHINE LEARNING FOR GROCERY STORES

By

KIRAN SANTHOSH PARAKKAL

A DISSERTATION

Submitted to



The University of Liverpool

in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

22/09/2023

ABSTRACT

SALES FORECASTING USING MACHINE LEARNING FOR GROCERY STORES

By

KIRAN SANTHOSH PARAKKAL

This project focuses on the development of an accurate sales forecasting model tailored for grocery stores. It encompasses data preprocessing, exploratory data analysis, and the implementation of multiple forecasting models, including Extreme Gradient Boosting (XGBoost), Long Short-Term Memory (LSTM), and Random Forest. The project addresses the critical challenge faced by grocery stores in accurately predicting sales, which impacts inventory management, resource allocation, and customer satisfaction. The objective is to create a robust and data-driven forecasting system that leverages historical sales data and external factors. The solution aims to be user-friendly, scalable, and adaptable to various store sizes and product categories, aiding grocery store owners and managers in making informed decisions to optimize operations. The approach involves utilizing available datasets, cleaning and preprocessing the data, and performing feature engineering. Exploratory data analysis techniques will provide insights into variable relationships and their impact on sales. Multiple forecasting models will be constructed and assessed using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The best-performing model will be selected, further refined, and optimized. The project's outcomes will include data visualizations, the chosen optimized model, and the development of a user-friendly web application using Streamlit. This application will allow store employees to upload product data and receive sales predictions in CSV format, providing a practical tool for inventory management and forecasting accuracy enhancement. Overall, this project seeks to explore the application of machine learning models in sales forecasting and has the potential to extend its utility to various other domains.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I confirm that I have not copied material from another source nor committed plagiarism nor commissioned all or part of the work (including unacceptable proof-reading) nor fabricated, falsified or embellished data when completing the attached piece of work.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Kiran Santhosh Parakkal

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to all those individuals and institutions whose contributions and support have been instrumental in the successful completion of this project.

First and foremost, I extend my deepest appreciation to my project supervisor, Ramya Chavali. Your unwavering guidance, vast expertise, and continuous support have been invaluable throughout the entire journey of research and development. Your insights and mentorship have not only enriched the project but also enhanced my own growth and understanding. Your dedication to this endeavour has been a constant source of motivation and inspiration.

I would also like to extend my sincere thanks to the University of Liverpool for providing access to the essential resources and data that were crucial for the execution of this project. The resources and facilities made available by the institution have played a significant role in the project's success. To my friends and family, I owe a debt of gratitude for their unwavering understanding and support during this endeavour. Your encouragement, patience, and belief in my abilities have been a constant source of strength. Your faith in me has been a driving force that kept me focused and determined to achieve the project's goals.

Lastly, I want to express my thanks to the entire open-source community and the creators of the various tools and libraries that have been pivotal in bringing this project to fruition. The collective efforts of this community have enriched the fields of data science and machine learning and have made projects like mine possible.

This project stands as a testament to the collaborative efforts and support of all those mentioned above. I am deeply grateful for your contributions and for being an integral part of this rewarding journey.

TABLE OF CONTENT

	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
Chapter 1. Introduction	1
Chapter 2. Aims And Objectives	3
Chapter 3. Background and review of literature	4
Chapter 4. Ethical Use Of Data	6
Chapter 5. Design	6
Chapter 6. Implementation	16
Chapter 7. Evaluation	24
Chapter 8. Learning Points	25
Chapter 9. Professional Issues	26
Chapter 10. Conclusion	27
BIBLIOGRAPHY	29
Appendix A. Data Visualization / Outlier Experiment	32
Appendix B. Need for Data Preprocessing & Hyperparameter Tuning	35
Appendix C. Dataset Variable Description	35
Appendix D. Project Log	36

LIST OF TABLES

	Page
Table 1. Project Log.....	36

LIST OF FIGURES

	Page
Figure 1. Project Use case Diagram.....	6
Figure 2 Figure 2 XG Boost Model (Zou et al., 2022)	8
Figure 3 Figure 3 Random Forest Diagram (E R, 2021)	10
Figure 4. Figure 4. LSTM Architecture (d2l.ai)	11
Figure 5 Figure 5 Home and Contact Page Mockup	15
Figure 6 Figure 6 Predictor page before and after file upload mockup.....	15
Figure 7. Figure 7. Dataset (Before preprocessing)	17
Figure 8. Figure 8. Dataset (After Preprocessing)	17
Figure 9. Figure 9. Random Forest Evaluation Matrix	18
Figure 10. XGBoost Evaluation Matrix	19
Figure 11. Code realisation of LSTM model	20
Figure 12. LSTM Evaluation matrix.....	21
Figure 13. LSTM train/validation loss graph.....	21
Figure 14. Final Dataset with Predicted Sales & Inventory Recommendation	21
Figure 15. Web App Homepage	22
Figure 16. Web App Predictor Page	22
Figure 17. Web App Predictor Page (Uploaded and Predicted Data).....	23
Figure 18. Predicted Sales with Output download button	23
Figure 19. Web App Contact Us page	23
 Appendix Figure 1. Establishment_Year Countplot.....	 32
Appendix Figure 2. Item_Fat_Content Countplot	32
Appendix Figure 3. Dataset Correlation Matrix	32
Appendix Figure 4. Item_Type Countplot.....	33
Appendix Figure 5. Item_Weight Boxplot	33
Appendix Figure 6. Item_Visibility Boxplot	33
Appendix Figure 7. Item_MRP Boxplot.....	33
Appendix Figure 8. Item_Outlet_Sales Boxplot.....	33
Appendix Figure 9. Outlet_Years Boxplot	34
Appendix Figure 10. Item_Visibility Boxplot (After Box-Cox Transform)	34
Appendix Figure 11. Item_Outlet Sales Boxplot (After cube root transform)	34
Appendix Figure 12. R2 Scores without data preprocessing and parameter tuning ...	35

Chapter 1. INTRODUCTION

1.1 Scope

The project aims to develop a precise sales forecasting model tailored for grocery stores, utilizing historical sales data and influencing factors like store location, type, size, product details (type, weight, price), and in-store product visibility. The project encompasses data preprocessing, exploratory data analysis, and the implementation of multiple forecasting models (Extreme Gradient Boosting, LSTM, Random Forest) to predict sales trends. Model performance will be assessed using metrics like MAE, RMSE, R2 and MAPE, with the best-performing model selected for optimization. Additionally, a user-friendly web-based interface will be created using Streamlit, allowing store employees to easily upload product data and receive sales predictions in CSV format. Future expansion possibilities and project evaluation will also be considered to enhance inventory management and forecast accuracy.

1.2 Problem Statement

Sales forecasting is a critical aspect of managing a grocery store's operations efficiently. Accurate sales predictions enable inventory management, staff scheduling, and resource allocation, ensuring that customers find the products they need while minimizing waste and cost. However, many grocery stores still rely on traditional methods or lack sophisticated tools for forecasting sales, resulting in challenges such as overstocking or understocking of products, leading to customer dissatisfaction and financial losses.

In this context, the problem statement for our project on "Sales Forecasting for a Grocery Store" is as follows:

“The grocery store industry faces significant challenges in accurately forecasting sales, leading to suboptimal inventory management, resource allocation, and customer satisfaction. The objective of this project is to develop a robust and data-driven sales forecasting system that leverages historical sales data, external factors, and advanced machine learning techniques to provide accurate and actionable sales predictions for a grocery store. The solution should be user-friendly, scalable, and adaptable to varying store sizes and product categories, ultimately helping grocery store owners and managers make informed decisions to optimize their operations.”

This project aims to address the critical need for improving sales forecasting in grocery stores, ultimately enhancing their overall efficiency and profitability.

1.3 Approach

To accomplish this goal, the project will utilize available datasets obtained from online sources. These datasets consist of historical sales data and information about the factors that influence sales, mentioned earlier. By making use of these datasets, the project can train forecasting models that can make precise predictions. The project will explore various forecasting techniques, such as Extreme Gradient boosting(XGB), Long Short-Term Memory (LSTM), and Random Forest. These algorithms have proven to be effective in handling the complex and dynamic nature of grocery store sales data and are widely used in time series forecasting tasks.

The initial step of the project will involve preprocessing the data and performing feature engineering. This includes cleaning the datasets and extracting relevant features, which will enable meaningful analysis. Exploratory data analysis techniques will be employed to gain insights into the relationships between different variables and their impact on sales.

Following the data preprocessing stage, the project will focus on developing and training the forecasting models. Multiple models will be constructed, including Extreme Gradient boosting (XGB), LSTM, and Random Forest. The assessment of these models' performance will involve the use of suitable metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).The models will be further refined and optimized to ensure accurate and reliable predictions.

Finally the performance of all the 3 models with respect to the given dataset will be compared and the best model is selected. Overall, the project would explore, how we can use different machine learning models for the purpose of sales forecasting hence can extend this application into other fields of interests.

Develop a web-based interface utilizing Streamlit to , enabling store employees to upload their product data and receive predicted sales results in CSV format.

1.4 Outcomes

The project's objectives encompass three primary deliverables:

1. The creation of data visualizations that illustrate the relationships between different variables in the dataset and emphasize the importance of each variable.
2. The development of an optimized final model chosen from a pool of three pre-existing models, specifically LSTM, XG Boost, and Random Forest.
3. The prediction of sales values using the selected algorithm based on user-uploaded data within a web application. Additionally, users will have the capability to download the entire dataset in CSV format.

In summary, the project outcomes consist of data visualization, the selection of the best-performing model (based on the aforementioned evaluation criteria), and the provision of a web application for users to conduct sales predictions on their datasets, along with the option to download the results.

Chapter 2. AIMS AND OBJECTIVES

2.1. To develop a robust sales forecasting system for grocery stores. (Aim achieved)

Objectives:

- a) Identify the primary factors that impact sales.
- b) Collect and analyze pertinent datasets sourced from online platforms for utilization in forecasting models.
- c) Examine and contrast various forecasting methods like Extreme Gradient boosting(XGBoost), LSTM, and Random Forest to determine the model that offers the highest accuracy and dependability.

2.2.To predict future sales patterns based on the identified factors. (Aim achieved)

Objectives:

- a) Prepare and clean the gathered datasets, guaranteeing the quality and consistency of the data.
- b) Conduct feature engineering to extract meaningful features that capture the impact of store location, store type and product.
- c) Develop and train forecasting models using the chosen techniques, taking into account the identified factors as input variables.
- d) Evaluate the performance of the models using appropriate metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
- e) Refine and optimize the models to improve their accuracy and reliability in predicting future sales.

2.3. To assist grocery stores in enhancing their inventory management and stock levels. (Aim achieved)

Objectives:

- a) Apply the trained forecasting models to generate sales predictions for future time periods.
- b) Develop a simple front-end web application for users to upload product data and download predicted result.
- c) Offer valuable insights and suggestions to grocery stores based on the predicted sales patterns.
- d) Empower stores to make well-informed decisions concerning inventory management, supply chain optimization, and resource allocation, thereby ensuring consistent availability of adequate stock levels.

2.4. To broaden the scope of applying sales forecasting models to additional fields or industries.

Objectives:

- a) Investigate the transferability of the developed forecasting models to other retail sectors or industries with similar sales dynamics.
- b) Evaluate the practicality and advantages of utilizing sales forecasting models in different areas of interest beyond the grocery store industry

Chapter 3. BACKGROUND AND REVIEW OF LITERATURE

3.1 Related Work

In a study conducted by (Mallik et al., 2022) ,the focus was on sales forecasting in marketing and its importance for product manufacturing. The study explored different techniques and machine learning algorithms used for sales forecasting, including Support Vector Machine, Decision Tree, Linear Regression, Random Forest, K-Nearest Neighbors, K-means Clustering, and Logistic Regression for review classification. The accuracy of each algorithm was evaluated using the Root Mean Square Error metric. The findings of the study revealed that Random Forest outperformed the other models, exhibiting the lowest Root Mean Square Error and thus providing the most accurate sales forecasts. For review classification, Logistic Regression demonstrated accurate results. The study emphasized the significance of these models and techniques in improving sales forecasting accuracy and aiding customers in making informed decisions based on the abundance of product reviews.

In their study, (Chen et al., 2021) ,focused on predicting the unit sales of Walmart retail goods using historical sales data and the dataset provided by the Kaggle competition. They utilized a Multi-layer feed-forward (MLF) neural network model to estimate sales for the next 28 days. Unlike previous approaches that heavily relied on statistical trend extrapolation and extensive customer and product analysis, the MLF model demonstrated superior performance. The evaluation of the model's predictions was done using the Root Mean Square Error (RMSE) metric. Python was the primary programming language used for data processing, modelling, and analysis throughout the research process. From this I got insight on the use of MLF neural network in sales forecasting.

(Chu and Zhang, 2003) , in their paper compares linear models and nonlinear neural networks for retail sales forecasting. Accurate predictions of retail sales are essential for improving operations and supply chains in the retail industry. The study focuses on the effects of different seasonal modelling strategies on forecasting accuracy. Time series approaches and regression approaches with seasonal variables are examined. The findings suggest that nonlinear methods, particularly neural networks, outperform linear models in capturing the movement of retail sales. The best-performing model for retail sales forecasting is the neural network model using deseasonalized time series data. This further justifies the preference of non-linear model in regression.

(Rao, Thangaraj and Kumari, 2023) in their project aimed to predict airfare for ticket bookings using the Gradient Boosting regression algorithm. The comparison was made with the new Linear Regression algorithm. A total of 20 examples from two different companies were used, with a sample size of 10 for each algorithm. The accuracy of the models was evaluated, and the Gradient Boosting regression achieved a higher accuracy of 82.5% compared to the new Linear Regression's accuracy of 62.5%.The better performance of the Gradient Boosting regressor could be attributed to its ability to handle nonlinear relationships, robustness to outliers, and its adaptability to complex data patterns.

In response to declining sales in retail stores caused by the rise of e-commerce, (Spuritha et al., 2021) proposes a solution involving sales prediction and dynamic pricing. The suggested model, based on XGBoost, is designed

to forecast sales for specific store-product combinations, optimizing parameters for enhanced prediction accuracy. The model successfully predicted sales for 10 stores with 50 products, achieving average MAPE, RMSE, and R2 values of 11.98%, 6.63, and 0.76, respectively. Additionally, dynamic pricing is applied to these forecasts, determining optimal product prices based on demand. Together, sales forecasting and dynamic pricing offer solutions for stock allocation and pricing challenges in the retail industry, effectively utilizing resources and boosting retail store sales.

3.2 Literature

The study done by (Anwer and Akyuz, 2022) compared the performance of five sales forecasting algorithms: MARS, LSTM, SARIMA, ARIMA, and RNN. The analysis used a real sales dataset from a mall spanning 2016 to 2021. R-squared (R2) and Root Mean Square Error (RMSE) were used as evaluation metrics. The results showed that SARIMA outperformed the other algorithms by effectively capturing both seasonality and trend in the data. However, ARIMA, LSTM, and RNN were not suitable for the dataset due to its seasonal and non-stationary nature. This highlights the limitations of certain machine learning algorithms when applied to specific dataset types.

The study performed by (dairu and Shilong, 2021) , aims to identify the most relevant features for sales prediction and utilize the XGBoost model for accurate forecasting. XGBoost, a highly efficient and flexible gradient boosting tree model, is found to be effective in both classification and regression tasks. Experimental results demonstrate the superior performance of the proposed XGBoost-based model in terms of prediction accuracy and running efficiency compared to other popular machine learning methods. Future work involves integrating other machine learning algorithms with XGBoost to enhance the model's capabilities, especially in handling hierarchical feature representations. Additionally, the study suggests exploring more efficient features through advanced feature engineering techniques to further enhance the accuracy of sales forecasting. Overall this paper also gave a brief knowledge on the working on XGBoost algorithm.

The article (KDnuggets, n.d.) addresses the importance of data engineering and feature engineering in machine learning. It emphasizes the significance of selecting, transforming, and enhancing source data to create meaningful predictive signals for the target variable in supervised learning. The article specifically delves into the preprocessing of data in a regression model, highlighting best practices using Python's pandas and sklearn libraries for data preparation, model training, and serving the model for prediction.

The article on "Detecting and Treating Outliers" by (Bonthu, 2021) provides a comprehensive guide to understanding, detecting, and handling outliers in data analysis. It covers various types of outliers, methods for detection, strategies for treatment, and their significance in machine learning models, offering practical examples and insights to help data professionals effectively manage outliers and improve data analysis and modeling accuracy.

The documentation for the Streamlit API (docs.streamlit.io, n.d.), provides a comprehensive overview of the process involved in constructing a web-based application using Streamlit. It offers guidance on key aspects, such as installing the necessary packages for Streamlit, introducing Streamlit's data model and development workflow, demonstrating the utilization of Streamlit's core functionalities for data retrieval and caching, chart creation, mapping data visualization, and the incorporation of interactive widgets like sliders for result filtering. In essence, this documentation furnishes a holistic understanding of how to seamlessly integrate and utilize Streamlit for web application development.

Chapter 4. ETHICAL USE OF DATA

As per the University's policy on research ethics, all research projects which involve the information that is freely available in the public domain does not require any prior ethical approval. Since the dataset (“Big Mart Sales Data”) we are using is taken from “Kaggle” which is a public domain, ethical approval is not required for our project.

Chapter 5. DESIGN

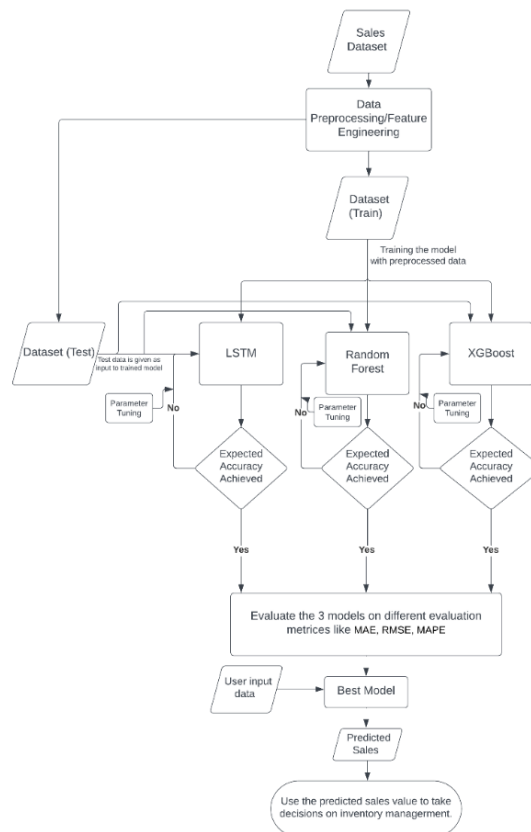


Figure 1. Project Use case Diagram

The provided use case diagram outlines the overall process of the project. Initially, the sales dataset is pre-processed and utilized as input for the three selected models. These models are then trained using the input dataset, followed by evaluating their performance using the test data. The pre-trained models generate results based on the test data input. To assess the accuracy of the models, a comparison is made between the generated results and the actual labels (sales values) of the test data. If the models fail to meet the desired accuracy, parameter tuning is performed. Subsequently, the best-performing versions of each model are evaluated using various metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) and Root Square Score (R2 Score). Ultimately, the most optimal model among the three is selected to predict sales based on user input. The predicted sales value is then utilized for inventory management decisions in grocery stores. A web-based application is created to enable users to input their datasets and subsequently download the predicted outcomes. Each steps of the project implementation are discussed in detail below.

5.1 Data Preprocessing

Data preprocessing is an essential step in data mining that aims to convert raw data into a more comprehensible and usable format. Sometime data can be incomplete, with missing attribute values or important attributes. It may also contain errors, outliers, or inconsistencies in coding or naming. Data preprocessing techniques are employed to address these challenges and rectify such issues.(KDnuggets, n.d.)

Some common data preprocessing techniques are discussed below.

1.Handling Missing Values in dataset: There are two approaches for dealing with missing values in a dataset. The first method involves removing rows with null values for a specific feature or columns with a high percentage of missing values (more than 75%). This approach is recommended when there are sufficient samples in the dataset. The second method involves replacing the missing values with the mean, median, or most frequent value. This approximation introduces some variance to the dataset but helps retain valuable information that would otherwise be lost by removing rows or columns. Replacing missing values with statistical approximations is considered a statistical technique and is also referred to as data leakage during training.(KDnuggets, n.d.)

2. Encode the Categorical data: Categorical data refers to variables that consist of labels instead of numeric values. To manage categorical data, we will utilize a method called label encoding. Label encoding involves transforming each value in a column into a numerical representation. To accomplish this, we will import the LabelEncoder module from the sklearn.preprocessing package. (KDnuggets, n.d.)

3.Feature Scaling: Feature scaling, also known as data normalization, is a technique utilized to standardize the range of independent variables or features in a dataset. The purpose is to ensure that the values of these variables are within a comparable range. Feature scaling becomes necessary when there is significant variation in the values of an independent variable. By applying feature scaling, we can bring all the values to a similar scale, facilitating fair comparison and analysis. There are different types of feature scaling methods like Standard Scaler,MinMax Scaler,Robust Scaler,Normalizer (KDnuggets, n.d.). In my project I will be using

MinMax Scaler, it transforms numerical features in a dataset so that they fall within a specific range, typically between 0 and 1.

4. Outlier Handling: An outlier refers to a specific data point in a dataset that stands out significantly from the majority of other data points, either by being notably larger or smaller in value compared to the rest of the observations. To address outliers, several methods can be employed, such as reducing the dataset size, applying various transformations, utilizing winsorization, or employing imputation techniques (Bonthu, 2021). An experiment will be carried out to detect outliers and determine the appropriate approach for managing these unusual data points.

5. Splitting the dataset in Train and Test Dataset: The dataset is divided into two subsets: the training dataset and the test dataset. The split ratio used is 80% for the training dataset and 20% for the test dataset. (KDnuggets, n.d.)

5.2 Machine Learning Models

5.2.1 Extreme Gradient Boosting (XGBoost)

XGBoost is a powerful library that utilizes **gradient boosting** to train machine learning models effectively and efficiently. It is designed to handle large datasets and can distribute the training process across multiple computing resources. By combining the predictions of multiple weaker models, XGBoost creates a more accurate and robust final prediction. (GeeksforGeeks, 2021)

Boosting: It is an ensemble modelling technique that aims to create a powerful classifier by combining multiple weak classifiers. It involves building a series of models, where each subsequent model focuses on correcting the errors made by the previous models. This iterative process continues until either the entire training dataset is accurately predicted or a predetermined maximum number of models is reached. (GeeksforGeeks, 2021)

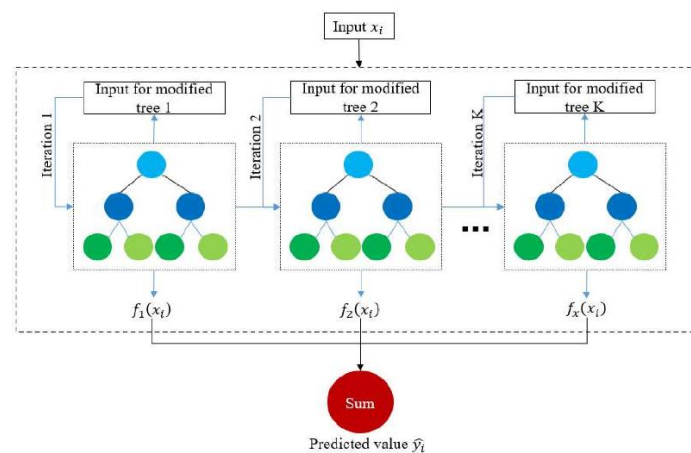


Figure 2 XG Boost Model (Zou et al., 2022)

Pseudocode:

1. Initialize the model parameters, such as the learning rate, number of trees, and maximum depth.
2. Split the dataset into the training set and the validation set.
3. Initialize the ensemble model, which is typically an empty list of decision trees.
4. For each iteration (tree) in the specified number of trees:
 - a. Calculate the gradients and Hessians (second derivative) for each instance in the training set.
 - b. Construct a decision tree using the training set, using the gradients and Hessians as the targets.
 - c. Add the constructed tree to the ensemble model.
 - d. Make predictions on the training set using the ensemble model.
 - e. Calculate the loss or error between the predicted values and the actual values.
 - f. Update the model parameters using gradient descent to minimize the loss.
1. Repeat steps 4 until the specified number of trees is reached.
2. Make predictions on the validation set using the ensemble model. (GeeksforGeeks, 2021)

Step by step explanation of the model

1. **Initialize the model:** XGBoost starts by initializing the model with an initial prediction. This could be a simple prediction based on the mean or any other suitable value.
2. **Boosting rounds:** XGBoost constructs a collection of weak learners, specifically decision trees, in an iterative and sequential manner. Each iteration, referred to as a "boosting round," adds a new weak learner to the ensemble.
3. **Calculate gradients and Hessians:** XGBoost computes the gradient(first derivative) and Hessian(second derivative) of the loss function for every training instance. These derivatives indicate the error's direction and magnitude in relation to the current model's predictions.
4. **Fit a weak learner:** A weak learner, which is a decision tree, is trained using the negative gradient (and Hessian) calculated from the training data. The decision tree is constructed to minimize the loss function, and the depth of the tree is controlled through hyperparameters like max_depth and min_child_weight.
5. **Update the model:** The newly trained decision tree is incorporated into the ensemble, and the model's predictions are adjusted by combining the predictions from all the trees in the ensemble. The influence of each tree on the final predictions is regulated by a learning rate, which scales the predictions to control their individual impact.
6. **Regularization:** XGBoost utilizes regularization techniques to mitigate the risk of overfitting.
7. **Termination condition:** XGBoost keeps adding new trees to the ensemble until a stopping condition is satisfied. This condition can be defined by the maximum number of boosting rounds, lack of substantial improvement in model performance, or any other use specified stopping criterion. (GeeksforGeeks, 2021)

Given the operational principles of the XGBoost model, it is anticipated to yield highly accurate results for our sales value prediction task. Therefore, I have chosen this model as the ideal solution for performing the task at hand.

5.2.2 Random Forest

Random Forest is a widely utilized and highly favoured algorithm in the field of data science. It is commonly employed for both classification and regression tasks. The algorithm constructs decision trees on various subsets of the data and combines their predictions through majority voting for classification problems or averaging for regression problems. A notable characteristic of the Random Forest Algorithm is its ability to handle datasets that include both continuous variables, such as in regression problems, and categorical variables, as seen in classification problems. (E R, 2021)

Random Forest algorithm is based on bagging ensemble technique.

Bagging: It is a technique that involves randomly selecting subsets of data from the original dataset. Each subset is used to train an independent model, generating individual results. The process of randomly selecting subsets with replacement from the original data is called bootstrap sampling. The models trained on these subsets are then combined by aggregating their results through majority voting. This aggregation step determines the final output of the ensemble, leveraging the collective predictions of the individual models. (E R, 2021)

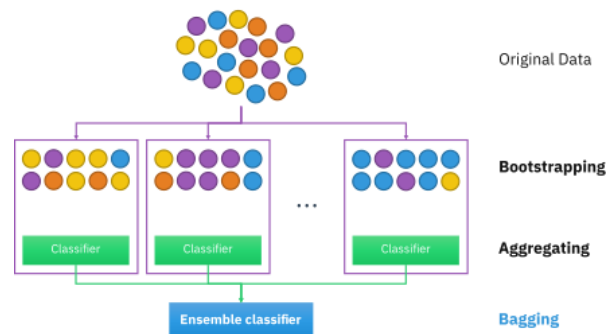


Figure 3 Random Forest Diagram (E R, 2021)

Pseudocode:

Random Forest Train: m -total features in dataset

1. Randomly choose " k " features from a pool of " m " total features.
 - Where $k \ll m$.
2. Using the best split point, determine the node " d " based on the " k " randomly selected features.
3. Divide the node into child nodes by utilizing the optimal splitting technique.
4. Continue executing steps 1 to 3 iteratively until a total of " l " nodes have been generated.
5. Construct a forest by repeating steps 1 to 4 " n " times, resulting in the creation of " n " trees. (Polamuri, 2017)

Random Forest Test:

1. Utilize the test features and apply the rules of each randomly generated decision tree to make predictions and store the predicted outcomes (targets).
2. Compute the vote count(classification) or value(regression) for each predicted target.
3. Take the predicted target with the highest number of votes(classification) or average of predicted values (regression) as the final prediction generated by the random forest algorithm. (Polamuri, 2017)

Working in brief:

In the Random Forest model, we select a subset of data points and features to build each decision tree. Put simply, from a dataset containing k records, we randomly choose n records and m features.

A separate decision tree is built for each sample in the dataset. Every decision tree will produce an output or prediction. The final output is determined by applying either Majority Voting for classification or Averaging for regression. (E R, 2021)

5.2.3 Long Short Term Memory (LSTM)

Long Short-Term Memory Networks (LSTMs) are a class of deep learning models specifically designed for sequential data, aiming to overcome the problem of vanishing gradients often encountered in traditional Recurrent Neural Networks (RNNs). LSTMs excel at capturing and leveraging information across extensive time intervals, enabling improved learning and representation of sequential patterns. In Python, one can implement LSTMs using the Keras library. (saxena, 2021)

LSTM Architecture explained:

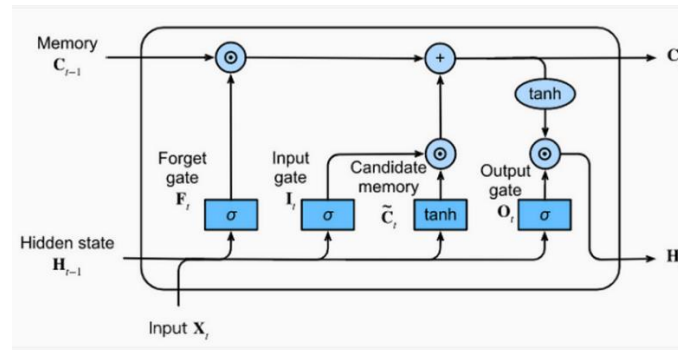


Figure 4. LSTM Architecture (d2l.ai)

Forget Gate: Within an LSTM cell, the initial step involves determining whether to retain or disregard the information from the previous time step. This decision is made using the forget gate equation.

$$f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$$

X_t : input to the current timestamp, U_f : weight associated with the input, H_{t-1} : The hidden state of the previous timestamp, W_f : It is the weight matrix associated with the hidden state

A sigmoid function is applied to the result, which restricts the value of "ft" between 0 and 1. This "ft" value is then multiplied with the cell state from the previous timestamp " C_{t-1} ". If value of "ft" is zero the model forgets everything else the data from previous layer is retained. (saxena, 2021)

Input Gate: The input gate is employed to evaluate the importance of the new information present in the input. The equation associated with the input gate quantifies this significance.

$$i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$$

Once again, a sigmoid function is applied to the input gate, resulting in a value between 0 and 1 for "I" at timestamp "t".(saxena, 2021)

New Information:

$$N_t = \tanh(x_t * U_c + H_{t-1} * W_c) \text{ (new information)}$$

To incorporate the new information into the cell state, a combination of the previous hidden state at timestamp "t-1" and the input "x" at timestamp "t" is modified using the activation function tanh. This transformation ensures that the new information, denoted as "Nt," has a value ranging from -1 to 1. The incorporation of "Nt" into the cell state is not a direct addition; instead, the following equation is utilized to update the cell state, considering whether "Nt" is negative or positive:

$$C_t = f_t * C_{t-1} + i_t * N_t \text{ (updating cell state)}$$

If the value of this new information "Nt" is negative, it is subtracted from the cell state, and if it is positive, it is added to the cell state at the current timestamp. Here, Ct-1 is the cell state at the current timestamp. (saxena, 2021)

Output Gate:

$$o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$$

The value of "Ot" is obtained by applying a sigmoid function, ensuring it falls within the range of 0 and 1. To calculate the current hidden state, we combine "Ot" with the hyperbolic tangent (tanh) of the updated cell state, as depicted in the equation below.

$$H_t = o_t * \tanh(C_t)$$

The hidden state (Ht) is influenced by both the long-term memory (Ct) and the current output. When there is a need to obtain the output at the current timestamp, the hidden state (Ht) can be processed through the SoftMax activation function. (saxena, 2021)

$$\text{Output} = \text{Softmax}(H_t)$$

Due to its ability to store and utilize crucial past information and effectively handle intricate datasets, the LSTM model is anticipated to demonstrate strong performance in tasks such as sales forecast prediction.

5.3 Development Environment and Implementation Language

5.3.1. Environment: The development environment to be used is Visual Studio Code. Visual Studio Code (VS Code) is a lightweight and powerful text editor that provides developers with numerous customizable features through plugins. It allows developers to create a tailored development environment and can be effortlessly installed on various platforms. (www.turing.com, n.d.)

5.3.2. Implementation language: Python programming language is to be used in this project. The reason for selecting python are: Presence of third-party modules, Extensive support libraries, Versatile, Easy to read, learn and write, User-friendly data structures, Portable across Operating systems etc. (GeeksforGeeks, 2017)

5.4 Data Sources

5.4.1 About Dataset: The dataset used in this project is derived from the sales data of BigMart in 2013. The dataset includes information for 1559 different products sold across 10 stores located in various cities. We have two datasets: the main dataset, consisting of **8523 records**, and the test dataset(user input) which contains similar data-points as main dataset except for the sales to be predicted, consisting of **5681 records(this dataset is given as input to the final model to predict sales values)**. The main dataset contains both the input variables and the corresponding output variable(s).

5.4.2. Dataset Variables/Features:

Main Dataset : ProductID, Weight, FatContent, Visibility(in-store), ProductType, MRP, OutletID, EstablishmentYear, OutletSize, LocationType, OutletType, OutletSales.

User Input: ProductID, Weight, FatContent, Visibility(in-store), ProductType, MRP, OutletID, EstablishmentYear, OutletSize, LocationType, OutletType, Current_Inventory. (*refer appendix C*)

5.4.3 Data Source: The dataset has been taken from “Kaggle” which is an open source platform. (www.kaggle.com, n.d.)

5.5 Testing and Evaluation

Testing: To assess the performance of the trained models, the dataset has been divided into a training set and a testing set. The training set contains 80% of the data, while the remaining 20% is allocated for testing. The predicted output from the models is then compared to the actual output from the testing dataset using different evaluation metrics to evaluate their performance.

Evaluation: To evaluate the performance of models various evaluation matrices are used like:

1. **Mean Absolute Error (MAE):** The Mean Absolute Error (MAE) score is determined by taking the average of the absolute differences between the predicted and expected values. By using the absolute function, the MAE ensures that the differences are always positive, regardless of whether the predicted value is greater or smaller than the expected value. To calculate the MAE, the absolute differences are summed up and divided by the total number of observations. (www.sciencedirect.com/MAE,n.d)

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_1 - \hat{y}_i|^2$$

2. **Root Mean Squared Error (RMSE):** The Root Mean Squared Error (RMSE) is derived by first squaring each error, then calculating the average of these squared errors, and finally taking the square root of that average. In other words, the RMSE involves squaring the errors, finding their mean, and then taking the square root to obtain the RMSE value.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n S_i - O_i^2}$$

where O_i are the observations, S_i predicted values of a variable, and n the number of observations available for analysis. (www.sciencedirect.com/RMSE,nd)

3. **Mean Absolute Percentage Error (MAPE):** It is calculated by finding the average of the absolute percentage difference between the predicted and actual values for each time period, where the absolute

percentage difference is obtained by taking the absolute value of the difference and dividing it by the actual value.

$$M = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$$

where n is the number of fitted points, A_t is the actual value, F_t is the forecast value. (Glen, 2017)

4. **R-Squared (R2 Score):** R-squared (R2) is a statistical metric used to quantify the portion of the variability observed in a dependent variable that can be attributed to an independent variable within a regression model. R-squared (R2) elucidates how much of the variation in one variable can be accounted for by the variation in another variable. In simpler terms, if a model has an R2 value of 0.50, it means that roughly 50% of the differences or fluctuations seen in the data can be attributed to the factors or inputs used in the model.

The formula for R-squared is:

$$R^2 = 1 - (SSR / SST)$$

SSR (Sum of Squared Residuals) : Sum of the squared differences between the predicted values (from the regression model) and the actual observed values of the dependent variable.

SST (Total Sum of Squares) : Sum of the squared differences between the actual observed values of the dependent variable and their mean. (Fernando, 2023)

The final model for making predictions on user data is chosen based on the model that achieves the highest score across all four evaluation metrics.

5.6 Web Application Design

The web application is created using the Streamlit (python library), which comes equipped with built-in styling features and interactive user input elements. This empowers users to effortlessly upload their datasets in the .csv format directly to the web application. Once the data is provided, the application utilizes it to make predictions using the most optimal model stored in the backend. The resulting predictions are showcased on the web page, and users have the option to download these predictions in .csv format for administrative use.

5.6.1 Streamlit

Streamlit is an open-source Python library designed to simplify and accelerate the development of web applications for data science, machine learning, and data analytics. It allows users, particularly data scientists and engineers, to create interactive and data-driven web applications with minimal effort and coding. (mhadhbi, 2021)

Key features and benefits of Streamlit include:

1. **Ease of Use:** Streamlit offers a straightforward and intuitive API that doesn't require extensive web development knowledge. Users can build web apps using familiar Python scripting.
2. **Rapid Prototyping:** With just a few lines of Python code, you can turn data scripts, data visualizations, and machine learning models into interactive web applications, making it ideal for rapid prototyping and experimentation.
3. **Built-in Widgets:** Streamlit provides a variety of built-in widgets (e.g., sliders, input boxes, buttons) to create interactive user interfaces and collect user input easily.
4. **Integration:** It seamlessly integrates with popular Python libraries and frameworks like Pandas, Matplotlib, Plotly, and scikit-learn, making it suitable for data analysis and machine learning tasks. (mhadhbi, 2021)

It can be installed by running the following command in your terminal: "pip install streamlit."

5.6.2 Web application Mockup

Below give are the initial mockup design of different pages in the web application.

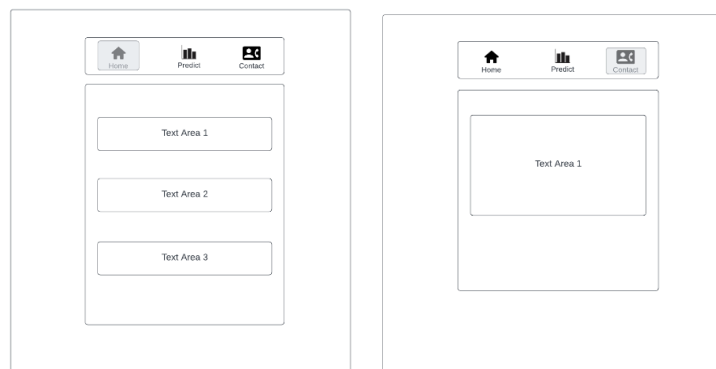


Figure 5 Home and Contact Page Mockup

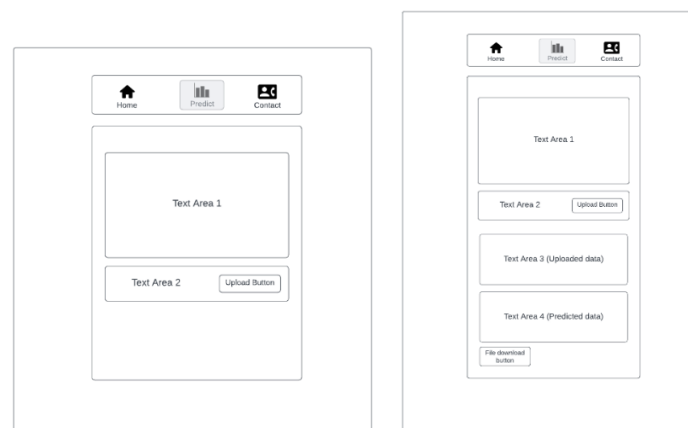


Figure 6 Predictor page before and after file upload mockup

Chapter 6. IMPLEMENTATION

6.1 Data Preprocessing (Implementation)

Various data preprocessing methods have been employed to modify the data in order to achieve the most optimized outcome. Below are these techniques and their implementation within the project.

1. Null values detection / handling : Null values (cells with no data) are detected by using a pandas inbuilt function “isnull()”, which detects null values in pandas dataframe and return boolean value. Inorder to find the total number of null values in each column we use “sum()” along with “isnull()”, hence the code is like “data.isnull().sum()”. Subsequently, missing values were identified in two columns of the dataset: "Item_Weight (float)" and "Outlet_Size (string)." To handle these missing values, null values in "Item_Weight" were replaced by substituting them with the column's mean value, while for "Outlet_Size," missing values were handled by determining and using the mode, which represents the most commonly occurring data in that particular column.

2. Zero values detection / handling : In the dataset, zero values were identified in the "Item_Visibility" field for specific items, even though these items had recorded sales (which is logically inconsistent because for an item to be sold, it must be visible to customers). To rectify this issue, the zero values were substituted with the mean values from the same column.

3. Removing insignificant columns / adding new columns : The columns “Item_Identifier” and “Outlet_Identifier” were found to be insignificant to our model training, as these columns consisted of unique combinations of characters and numbers used solely for identifying items or outlets, and they had no impact on the item sales. Consequently, both of these columns were removed from the dataset. Additionally, the 'Outlet_Establishment_Year' column was replaced with a new column named 'Outlet_Years,' which represents the total number of years each outlet has been in operation. This was computed by taking the difference between the current year's value and the year of outlet establishment (e.g., 2023 - 1997).

4. Label encoding : The columns "Item_Fat_Content," "Item_Type," "Outlet_Size," "Outlet_Location_Type," and "Outlet_Type" contain non-numeric data, so it's necessary to apply label encoding to these columns. To carry out label encoding, the "LabelEncoder()" function from the "sklearn.preprocessing" package is employed. Label encoding converts each unique value within these columns into a corresponding numerical representation.

5. Outlier Detection / Handling : Box plots were utilized to identify outliers in every column of the dataset. Substantial numbers of outliers were detected in the "Item_Visibility" and "Item_Outlet_Sales" columns. To address these outliers, the "Box-Cox transform" was applied to one column, and the "cube root transform" was applied to the other. (*Outlier detection experiment explained in detail in appendix section*)

6. Data Scaling : Min-Max scaling technique was applied on the dataset to normalize the values and reduce the standard deviation among them.

7. Train / Test split of data : The dataset is split into train and test data in the ratio of 80% for training and 20% for testing purpose. This has been implemented using “train_test_split” from “sklearn.model_selection” package.

The below screenshots shows data before and after data preprocessing.

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	15.515420
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	7.623152
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	12.799691
3	FDX07	19.20	Regular	0.070482	Fruits and Vegetables	182.0950	OUT010	1998	Medium	Tier 3	Grocery Store	9.012329
4	NCD19	8.93	Low Fat	0.070482	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	9.983305

Figure 7. Dataset (Before preprocessing)

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outlet_Years
0	0.282525	0.0	0.014752	0.266667	0.927507	0.5	0.0	0.333333	15.515420	0.416667
1	0.081274	1.0	0.017432	0.933333	0.072068	0.5	1.0	0.666667	7.623152	0.000000
2	0.770765	0.0	0.015351	0.666667	0.468288	0.5	0.0	0.333333	12.799691	0.416667
3	0.871986	1.0	0.049855	0.400000	0.640093	0.5	1.0	0.000000	9.012329	0.458333
4	0.260494	0.0	0.049855	0.600000	0.095805	0.0	1.0	0.333333	9.983305	0.916667

Figure 8. Dataset (After Preprocessing)

6.2 Machine Learning Model Implementation

6.2.1 Random Forest Model

In the project, a Random Forest Regressor model was employed because sales forecasting involves a regression task. To achieve this, the "**RandomForestRegressor()**" class from the "**sklearn.ensemble**" package was utilized for the specific purpose of building the model.

Hyperparameters Used:

1. **n_estimators** : The number of decision trees (estimators) to include in the forest. Increasing the number of trees can lead to better model performance, but it also increases computation time.
2. **max_depth** : The maximum depth of each decision tree in the forest. Controlling the depth helps prevent overfitting. Smaller values restrict the depth, making the trees shallower.
3. **min_sample_split** : The minimum number of samples required to split an internal node. A higher value prevents the model from splitting nodes that have too few samples, which can help avoid overfitting.
4. **min_sample_leaf** : The minimum number of samples required to be in a leaf node. This parameter controls the minimum size of leaf nodes, helping to prevent overfitting. Smaller values can lead to more complex trees. (Koehrsen, 2018)

Hyperparameters Tuning: 'GridSearchCV()' is used to perform hyperparameter tuning for random forest. GridSearchCV, abbreviated as Grid Search Cross-Validation, is a method applied in machine learning to optimize hyperparameters. It is a feature of the Python library scikit-learn (sklearn) and represents a structured method for identifying the most effective combination of hyperparameters for a given machine learning model. (Mujtaba, 2020) .

After performing GridSearch Cross Validation the best set of hyperparameters obtained for random forest model are {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 500}.

Random Forest model training / testing

The Random Forest model, configured with its optimal hyperparameters, is trained using the training dataset. Subsequently, the model's performance is assessed by making predictions on the test data (X_test) and comparing these predictions (y_pred) to the known test outcomes (y_test). Various evaluation metrics, such as R-squared (R2) score, MAE , RMSE and MAPE are utilized to gauge how well the model performs. The screenshot below displays the values obtained for each of these evaluation metrics.

```
Random Forest r2Score : 0.704838590799787
Random Forest train r2Score : 0.767777431750653
Random Forest mean absolute error : 1.5077628223163224
Random Forest root mean square error : 1.936183205551795
Random Forest mean absolute percentage error : 14.205802616810859
```

Figure 9. Random Forest Evaluation Matrix

(Note: The values of mae and rmse are in one digit decimal values as we have applied a cube transform on sales values (target variable) as part of outlier handling, original sales values are in range of thousands (4 digits).)

6.2.2 Extreme Gradient Boost (XGBoost) Model

The model was constructed using the "XGBRegressor()" class found within the "xgboost" package.

Hyperparameters Used:

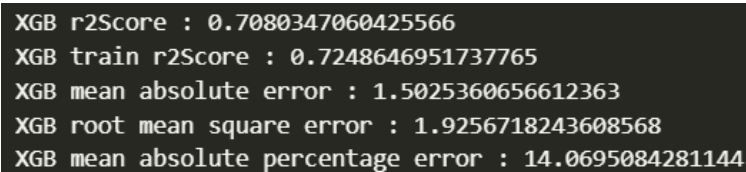
1. max_depth : The maximum depth of each decision tree. Controls the complexity of individual trees. Deeper trees can capture more complex relationships but may lead to overfitting.
2. learning rate : The step size or shrinkage applied during each boosting iteration. A lower learning rate makes the model more robust but requires more trees (higher n_estimators) for similar performance.
3. n_estimators : The number of boosting rounds or decision trees to build. Increasing this value typically improves model performance until it starts overfitting.
4. min_child_weight : The minimum sum of instance weight (hessian) needed in a child. It helps prevent partitioning into regions with few samples.

5. `gamma` : A regularization term that specifies the minimum loss reduction required for a split. Higher values encourage more conservative tree growth.
6. `alpha` : Regularization terms applied to control overfitting. Add regularization penalties to the objective function. (Jain, 2016)

Hyperparameters Tuning: 'GridSearchCV()' is employed to conduct hyperparameter optimization for the XGBoost model as well. Following the execution of GridSearch Cross Validation, the optimal hyperparameter configuration is obtained as follows: {'`alpha`': 0.1, '`gamma`': 0.1, '`learning_rate`': 0.01, '`max_depth`': 5, '`min_child_weight`': 3, '`n_estimators`': 500}.

XGBoost model training / testing

The XGBoost model, fine-tuned with its best hyperparameters, is trained using the training dataset. Following the training process, the model's performance is evaluated by making predictions on the test data (`X_test`) and then comparing these predictions (`y_pred`) to the known test results (`y_test`). The values obtained for various evaluation metrics are presented in the screenshot below.



```
XGB r2Score : 0.7080347060425566
XGB train r2Score : 0.7248646951737765
XGB mean absolute error : 1.5025360656612363
XGB root mean square error : 1.9256718243608568
XGB mean absolute percentage error : 14.0695084281144
```

Figure 10. XGBoost Evaluation Matrix

6.2.3 Long Short Term Memory (LSTM) Model

LSTM model has been built with the help of tensorflow and keras packages. The model used in this project is a basic feedforward LSTM model with one LSTM layer followed by one or more dense layers. It's a relatively simple architecture and can be further customized based on the specific requirements of your regression task.

The **Sequential** class from the Keras library is employed to construct a linear arrangement of layers for building our LSTM model. To incorporate each layer into the neural network, we utilize the **add()** method provided by Keras. To introduce an LSTM layer, we make use of the **LSTM()** function from **keras.layers** in conjunction with the **add()** method. In our model, we incorporate one or more dense layers, which are inserted into the neural network using the **Dense()** function from **keras.layers**. Furthermore, we include a dropout layer within the network, and this is included using the **Dropout()** function from **keras.layers**.

```
def create_lstm_model(units=32, dense_units=[64, 32], activation='relu', Learning_rate=0.001):
    model = Sequential()
    model.add(LSTM(units, input_shape=(9, 1)))
    for units in dense_units:
        model.add(Dense(units, activation=activation))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='linear'))
    optimizer = tf.keras.optimizers.Adam(Learning_rate=Learning_rate)
    model.compile(loss='mse', optimizer=optimizer, metrics=['mse', 'mae'])
    return model
```

Figure 11. Code realisation of LSTM model

Hyperparameters Used:

1. **units:** This hyperparameter specifies the number of LSTM units (neurons) in the LSTM layer. Increasing the number of units allows the model to capture more complex patterns in the data and may increase the risk of overfitting if not carefully regularized.
2. **dense_units:** This is a list of integers that specifies the number of units in each dense (fully connected) layer added to the model after the LSTM layer. The number of units in dense layers determines the capacity of the model to learn complex relationships in the data.
3. **activation:** It defines the activation function used in the dense layers. Common choices include 'relu' (Rectified Linear Unit), 'sigmoid', or 'tanh'. The choice of activation function affects the non-linearity of the model.
4. **Dropout :** Dropout is a regularization technique that randomly sets a fraction of input units to 0 during training, which helps prevent overfitting. This reduces the reliance on any single neuron and can improve the model's generalization.
5. **Optimizer :** The optimizer influences how quickly the model converges during training and can affect the final model performance.
6. **learning_rate :** This hyperparameter sets the step size for weight . ([Eckhardt, 2018](#))

Hyperparameters Tuning: Hyperparameter tuning for the LSTM model involved experimenting with various values for parameters such as "units," "dense_units," "activation," and "learning_rate.", whereas for other hyperparameters a suitable values has been assigned. Each combination of these hyperparameters was evaluated by training and testing the model, and the selection of the best set of hyperparameters was based on the highest R2 score, which measures the model's goodness of fit. After this tuning process, the optimal hyperparameters chosen for the LSTM model were determined to be as follows: {'units': 32, 'dense_units': [64, 32], 'activation': 'relu', 'learning_rate': 0.01}.

LSTM model training / testing

The LSTM model, which underwent fine-tuning with its most effective hyperparameters, was trained and evaluated using both the training and testing datasets. The results, including various evaluation metrics, are displayed in the provided screenshot.

```
LSTM r2Score : 0.7020475353045104
LSTM train r2Score : 0.6975749050593627
LSTM mean absolute error : 1.5156441197276445
LSTM root mean square error : 1.9453159708253827
LSTM mean absolute percentage error : 14.831712421285017
```

Figure 12. LSTM Evaluation matrix

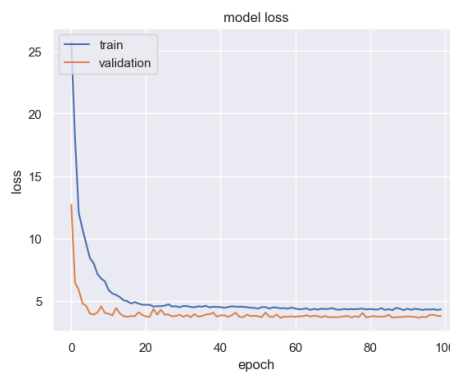


Figure 13. LSTM train/validation loss graph

6.2.4 Model Comparison & Prediction using best model

Among the 3 model it has been found that XGBoost model has the best performance, with and r2 score of 0.708 , mae of 1.5 , rmse of 1.9 and mape of 14.06.Hence we will be using XGBoost model for our final prediction task.This final model is being saved as “model.pkl” file .The user dataset serves as input for the XGBoost model that has been trained. The resulting outcome of the trained model is the projected sales figure for each individual product. The generated dataset consists of 2 new columns namely “Predicted_Sales” and “Inventory_recommendation”, which showcase the predicted sales values and recommendation to the user based on inventory management.

utlet Identifier	Outlet Establishment Year	Outlet Size	Outlet Location Type	Outlet Type	Current Inventory	Predicted Sales	Difference	Inventory recommendation
OUT049	1999	Medium	Tier 1	Supermarket Type1	1500	1570.0	-70.0	Order 70 items
OUT017	2007	NaN	Tier 2	Supermarket Type1	1500	1442.0	58.0	Sufficient stock present
OUT010	1998	NaN	Tier 3	Grocery Store	1500	635.0	865.0	Sufficient stock present
OUT017	2007	NaN	Tier 2	Supermarket Type1	1500	2318.0	-818.0	Order 818 items
OUT027	1985	Medium	Tier 3	Supermarket Type3	1500	5652.0	-4152.0	Order 4152 items

Figure 14. Final Dataset with Predicted Sales & Inventory Recommendation

6.3 Web Application Implementation

The web application was developed using the Streamlit package. User inputs are handled through the **"file_uploader()"** class provided by Streamlit, which creates a file upload area in the web front-end. The uploaded data is stored as a pandas dataframe, and all necessary preprocessing steps are applied to this data. Subsequently, the processed data is fed into the pre-trained XGBoost model, and the results are generated. These generated results are then presented on the webpage using the **"write()"** class from Streamlit. Additionally, download buttons for accessing the results are implemented through the **"download_button()"** class. The web application's appearance and style, including elements like a navigation bar and text area, are customized using the **"markdown()"** and **"option_menu()"** classes provided by Streamlit.

Below given are screenshots of web application that is developed as part of this project.

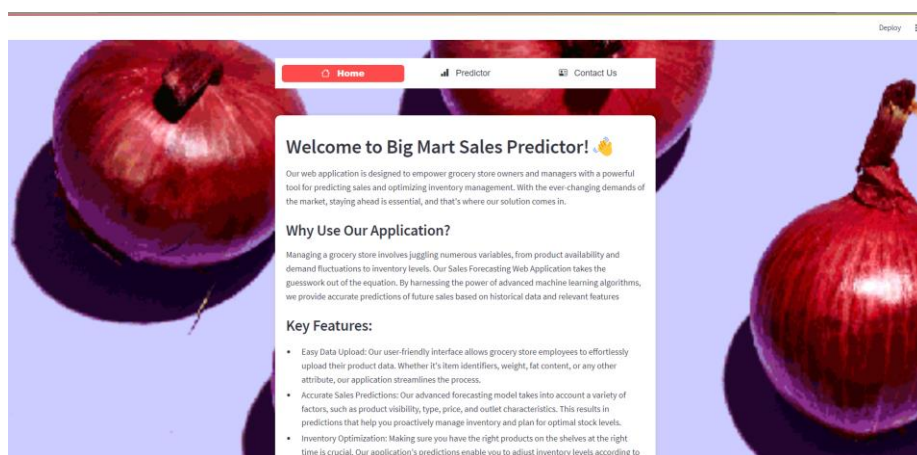


Figure 15. Web App Homepage

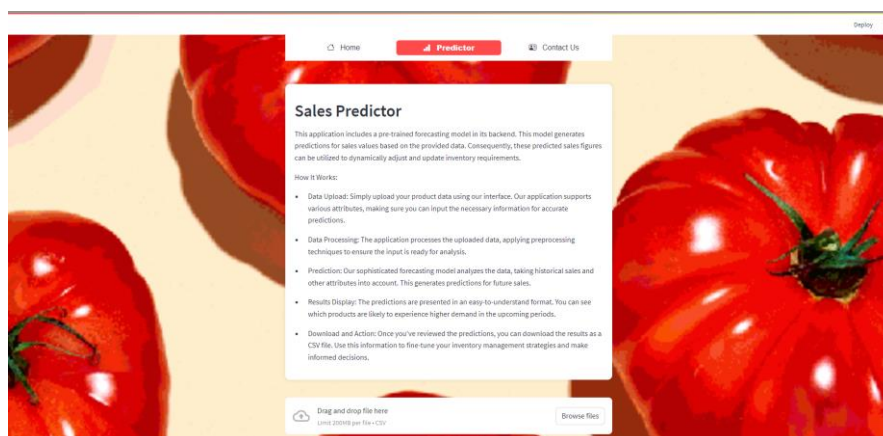


Figure 16. Web App Predictor Page

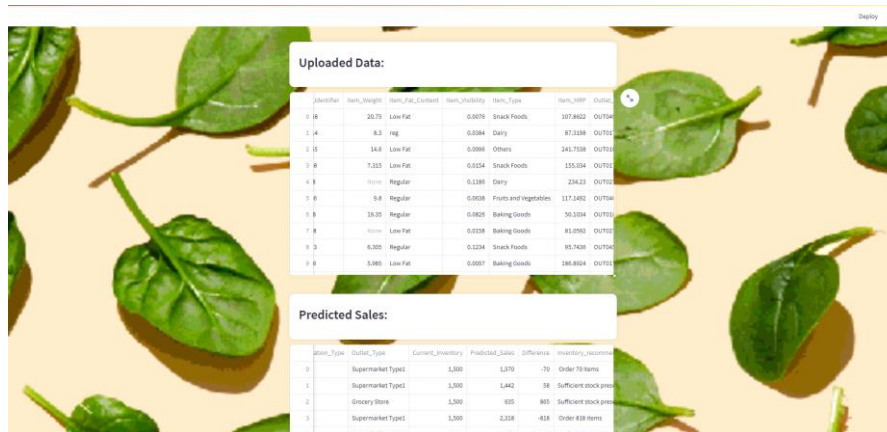


Figure 17. Web App Predictor Page (Uploaded and Predicted Data)

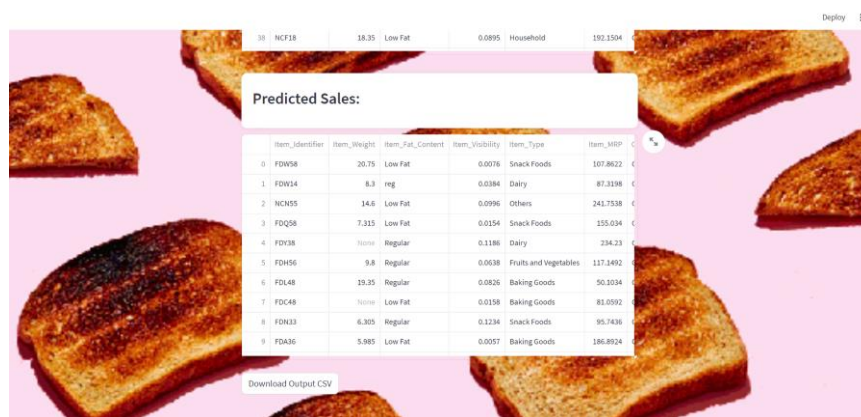


Figure 18. Predicted Sales with Output download button

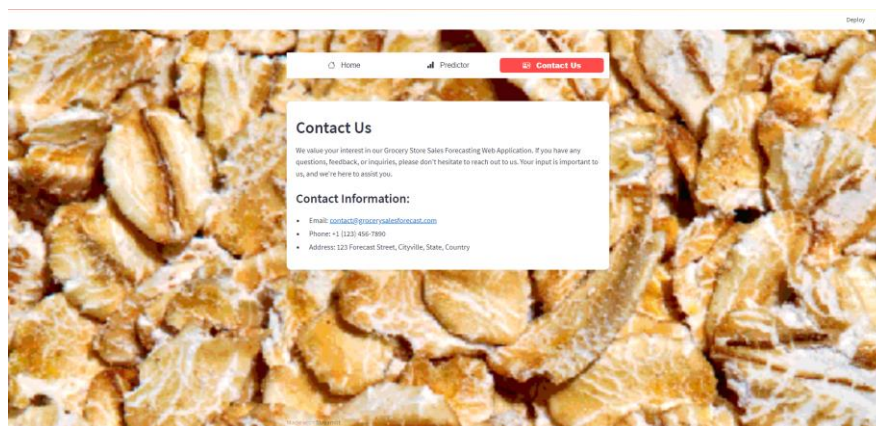


Figure 19. Web App Contact Us page

6.4 Difficulties encountered and their solutions

1. Very low r^2 Score (accuracy) was noticed for all the 3 models, which was later on found to be as a result of outliers in the dataset. Hence performed Outlier handling to improve the accuracy to around 71%,

2. There were issues encountered during the deployment of the web page's front-end, primarily stemming from version mismatches. These problems were successfully resolved by performing the required updates to ensure compatibility.
3. The CSS formatting styles added to the web application were not taking effect on the website as expected. To address this issue, various alternative CSS codes for achieving the same design were tested and experimented with. Ultimately, the version that worked correctly was applied to the website.

Chapter 7. EVALUATION

7.1.1 Achievements / Strengths

1. Investigated key elements influencing the sales of grocery stores and opted for a pertinent dataset from an accessible online resource, such as Kaggle.
2. Examined the dataset entries and conducted data visualization techniques to unveil correlations within the data.
3. Utilized data preprocessing methods to create an enhanced version of the dataset suitable for training purposes.
4. Conducted training for three models (Random Forest, XGBoost, and LSTM) and executed hyperparameter tuning to identify the most optimal configuration for each model.
5. Evaluated the trained models using the test dataset, achieving an accuracy level of approximately 71%, equivalent to an r2Score of about 0.71.
6. Utilized the highest-performing model (XGBoost) to make sales predictions and generate inventory suggestions based on user-provided input data.
7. Developed a User Interface for the project using “Streamlit”, where users can upload the input dataset and generate the output.

7.2 Weaknesses

All the milestone proposed in design phase were successfully met. However, while aiming for significantly improved accuracy through hyperparameter tuning during the design phase, the achieved accuracy reached a maximum of 71%. This outcome can be attributed to a range of factors, such as the presence of outliers in the dataset (despite applying outlier handling techniques), as well as the existence of non-linear relationships within the data.

Chapter 8. LEARNING POINTS

This project lead to the acquisition of various skills and knowledge. Here are some of the skills and knowledge that I gained from this project:

1. **Data Preprocessing and Cleaning:** Gained insights on how to clean and preprocess datasets which includes handling missing data, dealing with outliers, and ensuring data quality.
2. **Exploratory Data Analysis (EDA):** EDA is an essential step in understanding the data and uncovering insights. Gained proficiency in using various data visualization techniques to explore relationships between variables and identify patterns.
3. **Machine Learning Algorithms:** The project involves implementing and comparing multiple machine learning algorithms, including Extreme Gradient Boosting (XGBoost), Long Short-Term Memory (LSTM), and Random Forest. This has deepened my understanding of these algorithms and how to apply them to time series forecasting tasks.
4. **Model Evaluation:** Acquired the skill of evaluating machine learning model performance through the utilization of metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
5. **Hyperparameter Tuning:** Acquired knowledge regarding the importance of various hyperparameters in influencing the model's behavior, emphasizing the necessity of conducting hyperparameter tuning, and understanding the methods employed to achieve it.
6. **Web Application Development:** The project includes building a user-friendly web-based interface using Streamlit. Gained knowledge on how to create interactive web applications to present data and predictions.
7. **Project Management:** Managing a complex project like this involves planning, organizing, and executing various tasks in a structured manner. This improved my project management skills..

Overall, this project appears to provide a well-rounded learning experience, combining technical skills with domain-specific knowledge and project management abilities. The skills and knowledge acquired can be applied in a wide range of data science and machine learning projects across different domains.

Things to do differently in the future to avoid current pitfalls and issues

1. The moderate accuracy of 71% that has been recorded for the models is due to the presence of outlier in dataset. Hence while selecting dataset for a project, I would try to select the best dataset that has least outliers.

2. At present, only a subset of important hyperparameters has been taken into account for tuning due to the prolonged execution time associated with computational complexity. In the future, it is possible to include additional hyperparameters to further improve the results.
3. Incorporate a broader range of regression models to facilitate the process of identifying the top-performing model.

Chapter 9. PROFESSIONAL ISSUES

The project described aligns with several aspects of the BCS (The Chartered Institute for IT) Code of Conduct, which outlines the ethical and professional standards expected of IT professionals. Here's how the project aligns with some key principles of the BCS Code of Conduct:

1. **Public Interest:** The project addresses the public interest by improving the efficiency and effectiveness of grocery store operations through accurate sales forecasting. This benefits not only store owners but also customers by reducing waste and ensuring product availability.
2. **Professional Competence and Integrity:** The project demonstrates professional competence by utilizing advanced machine learning techniques and data analysis to create a robust sales forecasting model. The selection and optimization of forecasting models are based on sound data-driven practices, ensuring integrity in the decision-making process.
3. **Duty to the Profession:** The project contributes to the advancement of the IT profession by applying state-of-the-art machine learning techniques to solve real-world business challenges. It promotes best practices in data analysis, modelling, and software development, which are central to the IT profession.
4. **Public Relations:** The web-based interface created as part of the project serves as a means to communicate the results and predictions to store employees in a user-friendly manner. This aligns with fostering positive public relations.
5. **Privacy and Confidentiality:** The project handles only data that is available from an open source platform. Hence it does not violate privacy or confidentiality of any person or organisation.
6. **Professional Development:** The project involves the application of various machine learning techniques, which aligns with the principle of ongoing professional development. The project enhances skills and knowledge in this area.

In summary, the project aligns with several principles of the BCS Code of Conduct by addressing the public interest, demonstrating professional competence and integrity, and contributing to the advancement of the IT profession. It also underscores the importance of ethical considerations such as data privacy and confidentiality in IT projects.

Chapter 10. CONCLUSION

10.1 Summary

1. **Developed a robust sales forecasting system for grocery stores and predicted future sales patterns based on the identified factors :** After a comprehensive performance evaluation of three different models, namely Random Forest, LSTM, and XGBoost, successfully developed and chosen a robust sales forecasting model, specifically the XGBoost model. This best-performing model with 71 % accuracy has been employed to make predictions regarding future sales values, utilizing the provided input dataset.. *(Refer section 6.2 of Chapter 6)*
2. **Assist grocery stores in enhancing their inventory management and stock :** Created a web application using Streamlit that empowers store staff to upload monthly product data and instantly receive sales forecasts. Furthermore, the generated output now incorporates an "Inventory Recommendation" column, providing valuable guidance to employees for making informed inventory management decisions. *(Refer section 6.3 of Chapter 6)*
3. **Broaden the scope of applying sales forecasting models to additional fields :** Sales forecasting tasks similar to this one can be encountered in a wide range of sectors, including the automobile, textile, electrical appliances, and electronic gadgets industries. To adapt this project for compatibility with other sectors that involve sales forecasting, the primary adjustments needed are changing the dataset and applying preprocessing techniques that are relevant to the specific task. However, the overall project framework and workflow remain consistent. Consequently, this project can be readily applied to address sales forecasting challenges in various sectors with minimal modifications.

10.2 Main Findings

1. Data preprocessing and hyperparameter tuning play a crucial role in enhancing model performance. As a part of the project, we conducted a minor experiment to validate this assertion. The results demonstrated a significant improvement in accuracy, with an increase of approximately 10-15% when using preprocessed data and hyperparameter tuning for the models. *(Refer Appendix B)*
2. Outliers were found to exert a substantial impact on model performance. Upon addressing the issue of outliers, there was a noticeable enhancement in model accuracy, with an increase of approximately 10%.
3. The variance between train and test r^2 scores (accuracy) for all the 3 models are very low, hence we can confirm that the models are not underfitting or overfitting.
4. XGBoost model was found to be the best performing model among the 3 models. The possible reasons could be:

- a) It is an ensemble learning method that utilizes gradient boosting, which is a powerful technique for regression.
- b) XGBoost is designed for efficiency and can leverage parallel and distributed computing resources, making it suitable for large datasets.
- c) XGBoost can handle imbalanced datasets well.
- d) XGBoost can effectively capture non-linear relationships in the data. It combines the predictions of multiple trees, each addressing different aspects of the data, to produce a more accurate overall prediction.
- e) XGBoost is less sensitive to noisy data compared to other models. ([GeeksforGeeks, 2021](#))

10.3 Direction for further work

Below mentioned are the additional enhancements that can be made to the project in the future.

1. Conduct a comparative analysis of additional regression models such as ARIMA and SARIMA, which exhibit greater resilience to outliers when compared to the existing models.
2. Substitute Streamlit with Flask, a choice that provides users with significantly more flexibility in customizing the web front end.
3. Incorporate some data visualizations on the web application, so that store employees get a graphical understanding of sales.
4. Expand the project scope to encompass logistics and inventory management preparations that precede the acquisition of materials. This would involve factors such as determining the space required for storing the materials and establishing the necessary logistical support for the efficient transportation of purchased materials.

BIBLIOGRAPHY

- Mallik, R.S., Abhiram, R., Reddy, S.R. and Jagadish, R.M. (2022). *A Comprehensive Survey on Sales Forecasting Models Using Machine Learning Algorithms*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICERECT56837.2022.10060168>.
- Chen, J., Koju, W., Xu, S. and Liu, Z. (2021). Sales Forecasting Using Deep Neural Network And SHAP techniques. *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. doi:<https://doi.org/10.1109/icbaie52039.2021.9389930>.
- Chu, C.-W. and Zhang, G.P. (2003). A comparative study of linear and nonlinear models for aggregate retail sales forecasting. *International Journal of Production Economics*, [online] 86(3), pp.217–231. doi:[https://doi.org/10.1016/s0925-5273\(03\)00068-9](https://doi.org/10.1016/s0925-5273(03)00068-9).
- Anwer, M.O. and Akyuz, S. (2022). Sales Forecasting of a Hypermarket: Case Study in Baghdad Using Machine Learning. *2022 30th Signal Processing and Communications Applications Conference (SIU)*. doi:<https://doi.org/10.1109/siu55565.2022.9864955>.
- Rao, N.Sri.S.V.S., Thangaraj, S.John.J. and Kumari, V.S. (2023). *Flight Ticket Prediction Using Gradient Boosting Regressor Compared With Linear Regression*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICONSTEM56934.2023.10142428>.
- dairu, X. and Shilong, Z. (2021). *Machine Learning Model for Sales Forecasting by Using XGBoost*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICCECE51280.2021.9342304>.
- KDnuggets. (n.d.). *From Data Pre-processing to Optimizing a Regression Model Performance*. [online] Available at: <https://www.kdnuggets.com/2019/07/data-pre-processing-optimizing-regression-model-performance.html>.
- docs.streamlit.io. (n.d.). *Get started - Streamlit Docs*. [online] Available at: <https://docs.streamlit.io/library/get-started>.
- Spuritha, M., Kashyap, C.S., Nambiar, T.R., Kiran, D.R., Rao, N.S. and Reddy, G.P. (2021). *Quotidian Sales Forecasting using Machine Learning*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICSES52305.2021.9633975>.

Bonthu, H. (2021). *Detecting and Treating Outliers | How to Handle Outliers*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/>.

GeeksforGeeks (2021). *XGBoost*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/xgboost/>.

Zou, M., Jiang, W.-G., Qin, Q.-H., Liu, Y.-C. and Li, M.-L. (2022). Optimized XGBoost Model with Small Dataset for Predicting Relative Density of Ti-6Al-4V Parts Manufactured by Selective Laser Melting. *Materials*, 15(15), p.5298. doi:<https://doi.org/10.3390/ma15155298>.

E R, S. (2021). Random Forest | Introduction to Random Forest Algorithm. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>.

Polamuri, S. (2017). How the random forest algorithm works in machine learning. [online] Dataaspirant. Available at: <https://dataaspirant.com/random-forest-algorithm-machine-learning/>.

saxena, S. (2021). LSTM | Introduction to LSTM | Long Short Term Memor. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/>.

d2l.ai. (n.d.). 9.2. Long Short Term Memory (LSTM) — Dive into Deep Learning 0.14.4 documentation. [online] Available at: https://d2l.ai/chapter_recurrent-modern/lstm.html.

www.turing.com. (n.d.). Visual Studio vs Visual Studio Code - What's Best In 2022? [online] Available at: <https://www.turing.com/kb/ultimate-guide-visual-studio-vs-visual-studio-code>

GeeksforGeeks (2017). Python Language Advantages and Applications - GeeksforGeeks. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/python-language-advantages-applications/>.

www.kaggle.com. (n.d.). Big Mart Sales Prediction Datasets. [online] Available at: <https://www.kaggle.com/datasets/shivan118/big-mart-sales-prediction-datasets> [Accessed 3 Jul. 2023].

www.sciencedirect.com. (n.d.). Mean Absolute Error - an overview | ScienceDirect Topics. [online] Available at: <https://www.sciencedirect.com/topics/engineering/mean-absolute-error>.

www.sciencedirect.com. (n.d.). Root-Mean-Squared Error - an overview | ScienceDirect Topics. [online] Available at: <https://www.sciencedirect.com/topics/engineering/root-mean-squared-error>.

Glen, S. (2017). Mean absolute percentage error (MAPE). [online] Statistics How To. Available at: <https://www.statisticshowto.com/mean-absolute-percentage-error-mape/>.

Fernando, J. (2023). *R-Squared Definition*. [online] Investopedia. Available at: <https://www.investopedia.com/terms/r/r-squared.asp>.

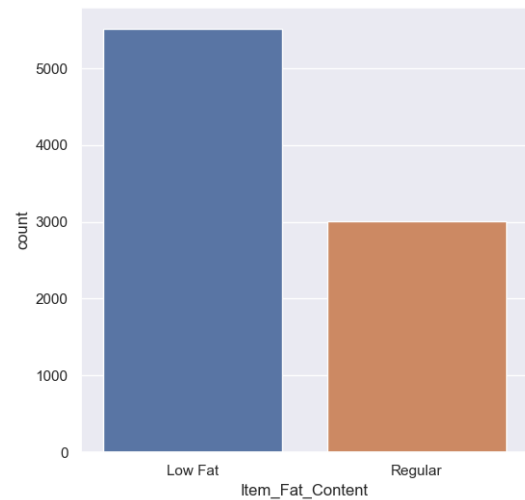
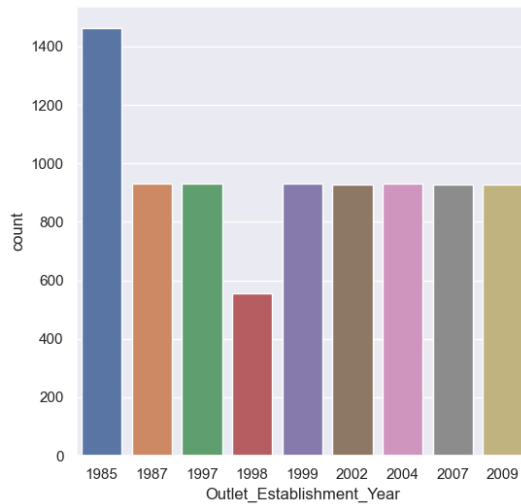
mhadhbi, N. (2021). *Python Tutorial: Streamlit*. [online] [www.datacamp.com](https://www.datacamp.com/tutorial/streamlit). Available at: <https://www.datacamp.com/tutorial/streamlit>.

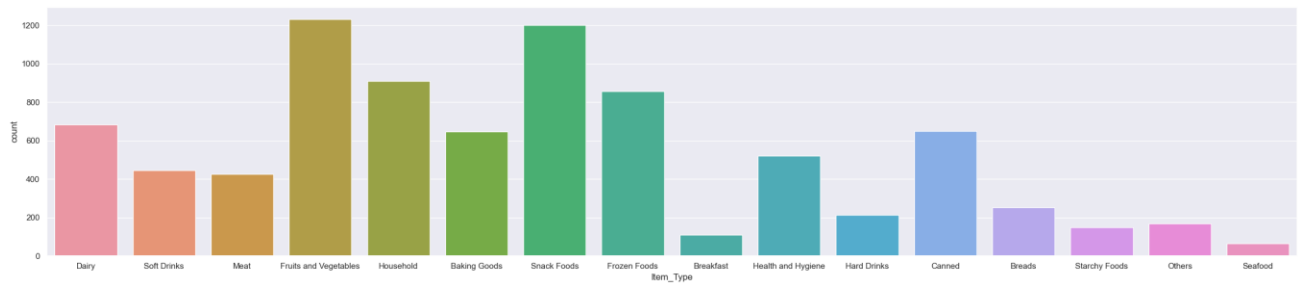
Koehrsen, W. (2018). *Hyperparameter Tuning the Random Forest in Python*. [online] Medium. Available at: <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>.

Mujtaba, H. (2020). *An Introduction to Grid Search CV / What is Grid Search*. [online] GreatLearning. Available at: <https://www.mygreatlearning.com/blog/gridsearchcv/>.

Jain, A. (2016). *Mastering XGBoost Parameter Tuning: A Complete Guide with Python Codes*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/#XGBoost_Parameters [Accessed 17 Sep. 2023].

Eckhardt, K. (2018). *Choosing the right Hyperparameters for a simple LSTM using Keras*. [online] Medium. Available at: <https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046>.

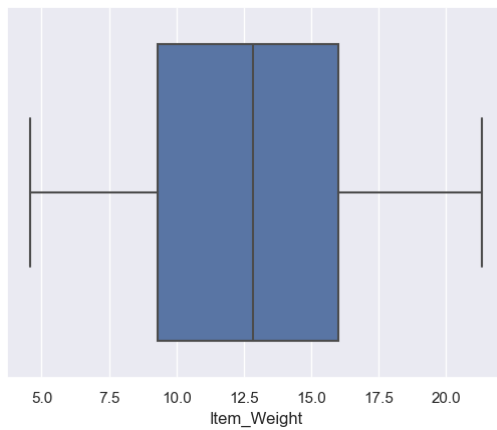




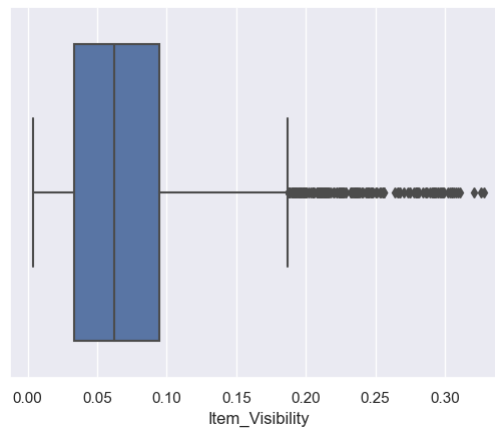
Appendix Figure 4. Item_Type (Count Plot)

A.2 Outlier Detection / Handling Experiment

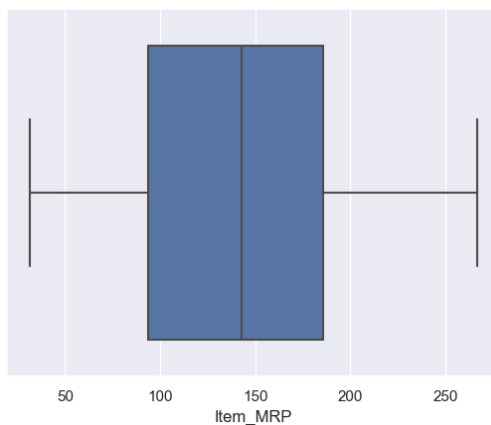
Outliers in the dataset are detected using “Box-plot” techniques. “Box-plot” are plotted for each field of the dataset. Below shown are the box-plot for each data category.



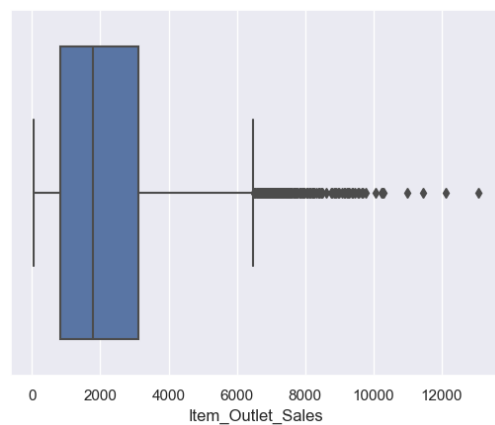
Appendix Figure 5, Item_Weight Boxplot



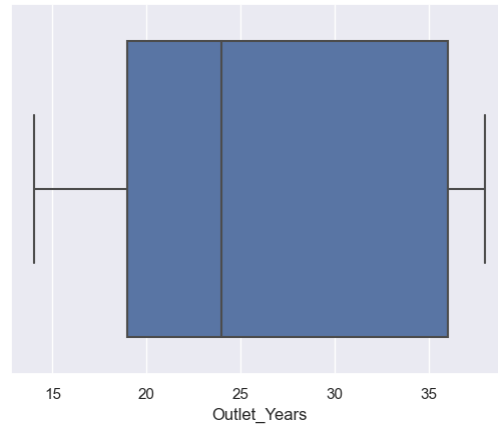
Appendix Figure 6, Item_Visibility Boxplot



Appendix Figure 7, Item_MRP Boxplot

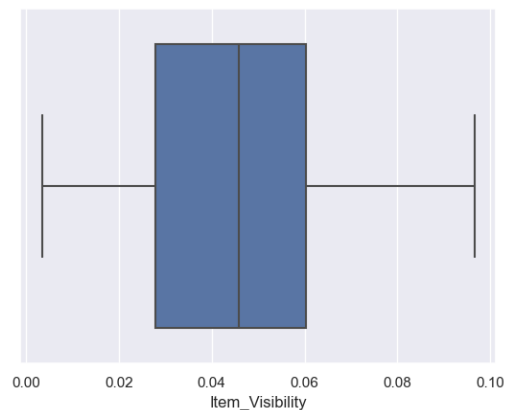


Appendix Figure 8, Item_Outlet_Sales Boxplot

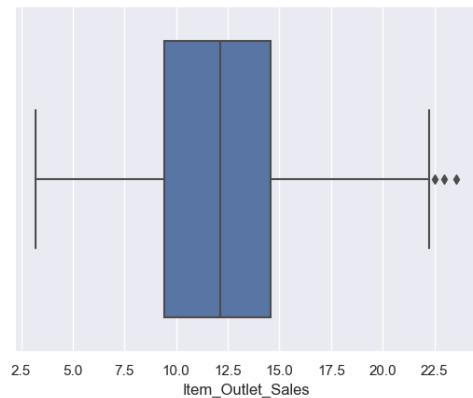


Appendix Figure 9, Outlet_Years Boxplot

Based on the previous box plots, it's evident that a notable number of outliers exist in both the "Item_Visibility" and "Outlet_Sales" columns. These outliers significantly hinder the training of our machine learning models, as it was observed that without outlier removal, the R2 score for each model was consistently within the 0.6 range. To address this outlier problem, we applied a "Box-Cox Transform" to the "Item_Visibility" column and a "Cube Root Transform" to the "Outlet Sales" column. Below, you can see the box plots after the outliers have been appropriately handled.



Appendix Figure 10, Item_Visibility Boxplot (After Box-Cox Transform)



Appendix Figure 11, Item_Sales Boxplot (cube root Transform)

Model r2 Score increased to 0.7 after outlier handling.

Appendix B. Need for Data Preprocessing & Hyperparameter Tuning

As a component of the project, I carried out a brief experiment to assess the impact of data preprocessing techniques, including data scaling, outlier handling, the removal of irrelevant columns, and hyperparameter tuning. The experiment involved a straightforward comparison of R2 scores between models trained using preprocessed and unprocessed datasets, as well as models with and without hyperparameter tuning. The below screenshots show the r2 scores of the models obtained without any data preprocessing or hyperparameter tuning.

```
R2 scores before data scaling,outlier detection or hyperparameter tuning
=====
R2 score of random forest model: 0.5730567939227657
R2 score of XGB model: 0.5539640075888196
R2 score of lstm model: 0.6109981911373693
```

Appendix Figure 12. R2 Scores without preprocessing and parameter tuning

The R2 scores achieved for all three models after undergoing preprocessing and hyperparameter tuning fall within the range of approximately 0.7, which translates to an accuracy of around 70% (as detailed in the implementation section). Consequently, there is a notable decline in accuracy by approximately 10 to 15 percentage points for models that do not undergo any parameter tuning or dataset preprocessing. This underscores the essential role of hyperparameter tuning and data preprocessing in improving the project's outcomes and highlights their significance.

Appendix C. Dataset Variable Description

Main Dataset Variable Description

1. ProductID: A distinct identifier for each product.
2. Weight: The mass of the products.
3. FatContent: Indicates whether the product has a low fat content or not.
4. Visibility: The proportion of the total display area in a store allocated to a specific product.
5. ProductType: The category that the product belongs to.
6. MRP: The highest retail price (the listed price) of the products.
7. OutletID: A unique identifier for each store.
8. EstablishmentYear: The year when the store was established.
9. OutletSize: The physical size of the store in terms of the ground area it covers.
10. LocationType: The classification of the city or area where the store is situated.
11. OutletType: Specifies whether the store is a simple grocery store or a more complex supermarket.
12. OutletSales: (Target variable) The amount of product sales in a particular store.

Input Dataset Variable Description

1. ProductID: A distinct identifier for each product.
2. Weight: The mass of the products.
3. FatContent: Indicates whether the product has a low fat content or not.
4. Visibility: The proportion of the total display area in a store allocated to a specific product.

5. ProductType: The category that the product belongs to.
6. MRP: The highest retail price (the listed price) of the products.
7. OutletID: A unique identifier for each store.
8. EstablishmentYear: The year when the store was established.
9. OutletSize: The physical size of the store in terms of the ground area it covers.
10. LocationType: The classification of the city or area where the store is situated.
11. OutletType: Specifies whether the store is a simple grocery store or a more complex supermarket.
12. Current_Inventory : Current stock of product in the store. (www.kaggle.com, n.d.)

Appendix D. Project Log

The table below provides a project log for this undertaking, where significant dates in the project's timeline are documented. These dates encompass milestones such as the completion of key project stages , review meetings, and various quality assurance activities.

Table 1. Project Log

Event	Dates	Status / Comments
Background Reading/Literature Review	June 05 – June 23	Completed
Dataset Selection	June 05 – June 16	Completed
Development of project specification and proposed design	June 19 – July 7	Completed
Review Meeting - 1	July 4	Review of design document by guide.
Data Preprocessing	July 10 – July 21	Completed
Model-1 (Random Forest) Implementation	July 17 – July 28	Completed
Model-2 (XGBoost) Implementation	July 24 -August 4	Completed
Model-3 (LSTM) Implementation	July 31 – August 11	Completed
Fine Tuning Models to improve Accuracy	August 7 – August 11	Completed
Models Performance Evaluation	August 7 – August 11	Completed
Final Model Selection and using it to forecast grocery store sales	August 7 – August 11	Completed

Review Meeting - 2	August 8	Discussed issues in model accuracy and obtained guidance to resolve it.
Development of Web Application	August 14 – August 18	Completed
Review Meeting - 3	August 21	Reviewed the completed source code with guide.
Adding comments to code and final small improvements	August 21 -August 25	Completed
Final Presentation ppt and report preparation	August 26 – September 1	Completed
Final Dissertation Report	September 3 – September 22	Completed