BRAC UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence
Fall - 2024

Project Report

Project Title: BigMart Sales Prediction

Lab Section - 11

Group - 12	
ID	**Name**
21201613	MD. Shafiur Rahman
21201009	Md. Nafizur Rahman Bhuiya

Table of Contents

Project Link:

CSE422 Project1.ipynb

Dataset source -

Kaggle - **BigMart Sales Prediction**

Reference - *BigMart Sales Prediction Dataset*. (2018). Kaggle.

https://www.kaggle.com/datasets/shivan118/big-mart-sales-prediction-datasets

Introduction

BigMart is a well-known retail business that operates many stores and provides various kinds of products to consumers. Effective sales and inventory management is important for the success of such huge-scale companies. To accomplish this, BigMart must understand its sales patterns and the elements that drive them. The BigMart Sales Prediction project aims to use past sales data to forecast future sales of various products across several retailers. BigMart may use these projections to improve inventory management, optimize marketing efforts and make smarter business decisions.

The dataset used on this project includes crucial product information such as size, type and category along with shop details such as location, type and size. It also contains sales information for various goods. By studying this data, we want to find essential trends, identify variables that impact sales, and establish models of prediction that accurately predict future sales.

This report provides an overview over the steps involved in working with the dataset such as cleaning and preparing it, investigating the correlations between different variables and constructing predictive models. In addition, the study will highlight data insights and provide recommendations on how BigMart can use them to enhance operations and the customer experience.

Dataset Description and Visualization

There are 12 columns in this dataset and they are:

1. Item_Identifier
2. Item_Weight
3. Item_Fat_Content
4. Item_Visibility
5. Item_Type
6. Item_MRP
7. Outlet_Identifier
8. Outlet_Establishment_Year
9. Outlet_Size
10. Outlet_Location_Type
11. Outlet_Type
12. Item_Outlet_Sales

The first step in which we worked on the dataset was to insert it into our data frame(df). It allows us to specify the column name, position, and values. Then we decided what we are predicting or which column is the target column. As we are going to predict the sales of a specific item from an outlet, we choose the last feature **Item_Outlet_Sales** as our target feature. Then we tried to find some information regarding the columns such as non-null count, unique value count and data type. The BigMart sales prediction is a regression problem because the goal is to predict a continuous variable for each store-item combination based on features like product type store location and previous sales data. The target variable is numeric and can take a wide range of values which makes this task a classic example of regression. Here we found some issues and this data needs to be preprocessed. The issues are as follows:

Preprocessing

- Some Null values need to be Imputed from columns:
    - **Item_Weight**
    - **Outlet_Size**
- Some Categorical Columns need to be encoded into numerical type:
    - **Item_Identifier**
    - **Item_Fat_Content**
    - **Item_Type**
    - **Outlet_Identifier**
    - **Outlet_Size**
    - **Outlet_Location_Type**
    - **Outlet_Type**

- Standardizing **Item_Fat_Content** column
- Checking for unique columns and then dropping **the Item_Identifier** column as all its values are unique; thus, this is not needed in our dataframe.
- Scaling the numerical columns because it brings the values into the range from 0 to 1.

First we checked for any duplicate data. As there is no duplicate data so we move forward to checking null values. There were null values in **Item_Weight** and **Outlet_Size**. So we handled them by filling them with some relevant data. Then, we checked for the columns which contained all the unique data and found **Item_Fat_Content** as unique.so we removed that column as we dont need them. Then we encoded the categorical data as we need only numerical value to proceed. After using **one-hot encoding** and **label encoding**, we finally scaled the data using normalized scaling.

On the next step, we implemented **correlation heatmap**, which represents the **correlation matrix**. It's used to discover relationships between variables, select features for the model, detect multicollinearity, and understand key drivers.

Feature Scaling

Feature scaling is an essential step in preparing our dataset for machine learning models. It involves normalizing the range of features so that the models interpret them on the same scale. Our dataset has many varying ranges, units, etc. So to interpret those features on the same scale we did **Min-Max Scaling.** Because it brings the values from the range 0 to 1.

Feature Correlation Analysis and Selection

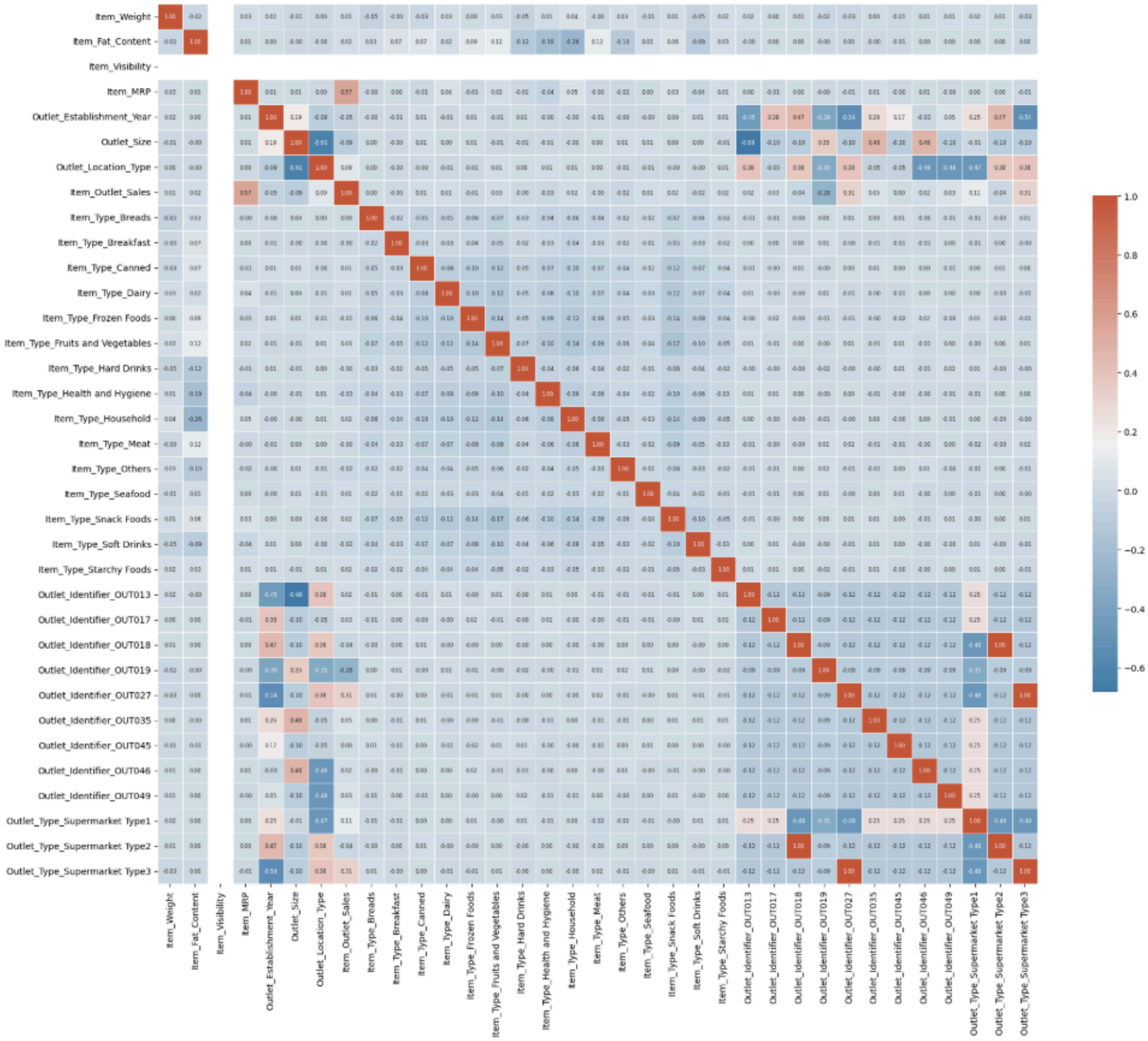

Fig1: Correlation matrix of dataset features after encoding

The feature correlation matrix shown in Fig after our feature scaling indicates that the feature **Item_Visibility** possesses only a single unique value. All the values of this particular feature become zero after encoding. It indicates that it will not contribute to resolving this problem. Based on our correlation analysis, we have decided that the feature **Item_Visibility** should be excluded prior to splitting our data and initiating model training.

Dataset Splitting

We divided the column features into **X** and **y** variables where **y** contains the target feature "**Item_Outlet_Sales**" which will be predicted and x contains the remaining other features. Following the completion of our feature selection process, and prior to commencing model training on the dataset, we split it into a 70% training and 30% testing ratio. We used 30% instances for our training because the dataset size is not very big and smaller training size may compromise model evaluation accuracy.

Model Training and Testing

We have used the following regression models to train and test the dataset:

A. **Linear Regression:** Linear Regression is used for predicting a continuous target variable based on one or more independent variables. It assumes a linear relationship between the features and the target variable.

It fits a straight line or hyperplane to the data by minimizing the sum of squared differences between the predicted and actual values.

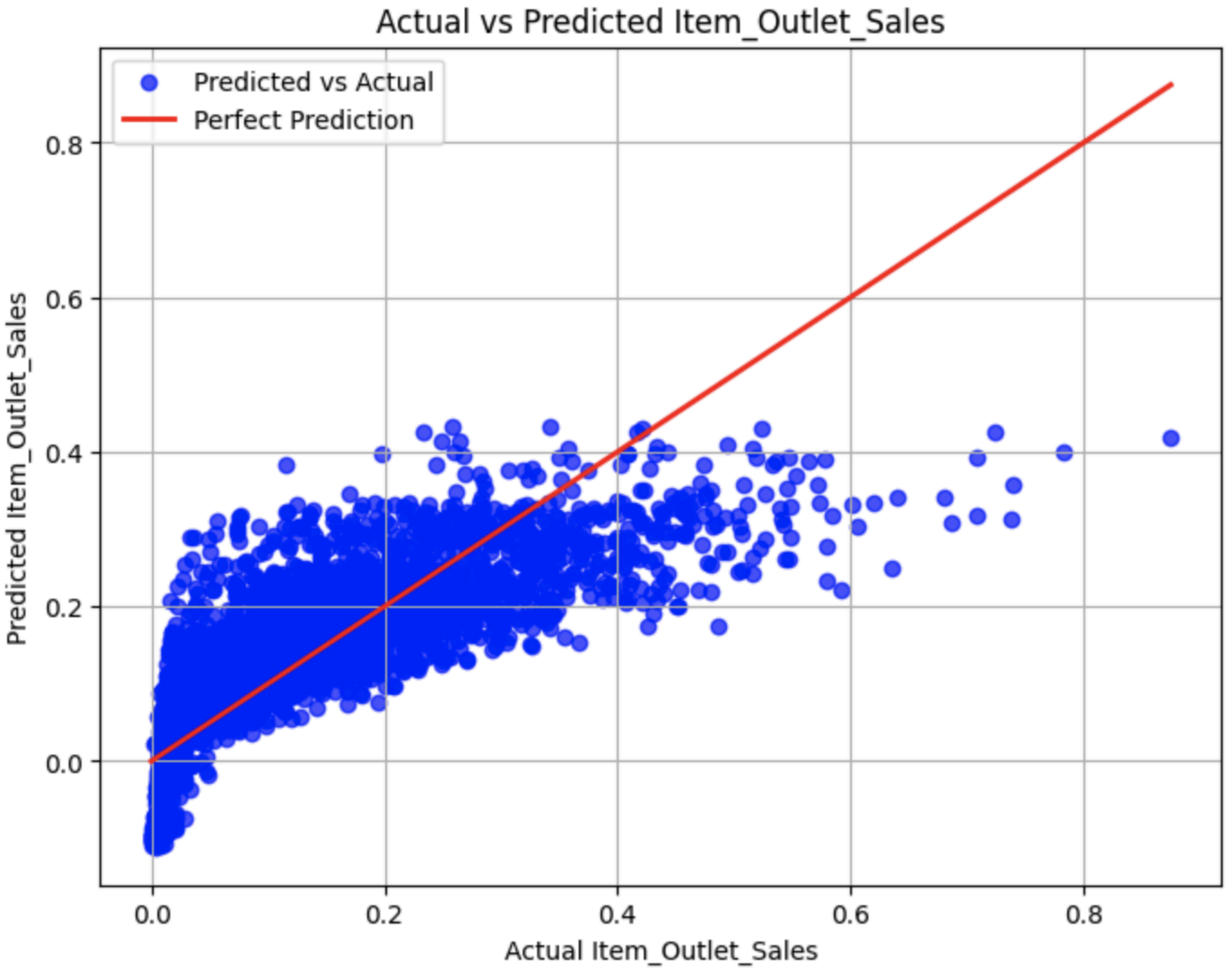

Fig2.1: Correlation matrix of dataset features after encoding

From this graph, we can see that The scatter points are closely distributed around the red line (perfect prediction line), indicating a reasonably good fit.

However, some deviations from the line show that the model struggles with certain predictions.

B. **Random Forest:** Random Forest is an ensemble learning method used for classification and regression. It builds multiple decision trees and merges them to improve the overall prediction.

It creates many decision trees using different random subsets of the data (bootstrapping) and averages their predictions (for regression) or uses majority voting (for classification).

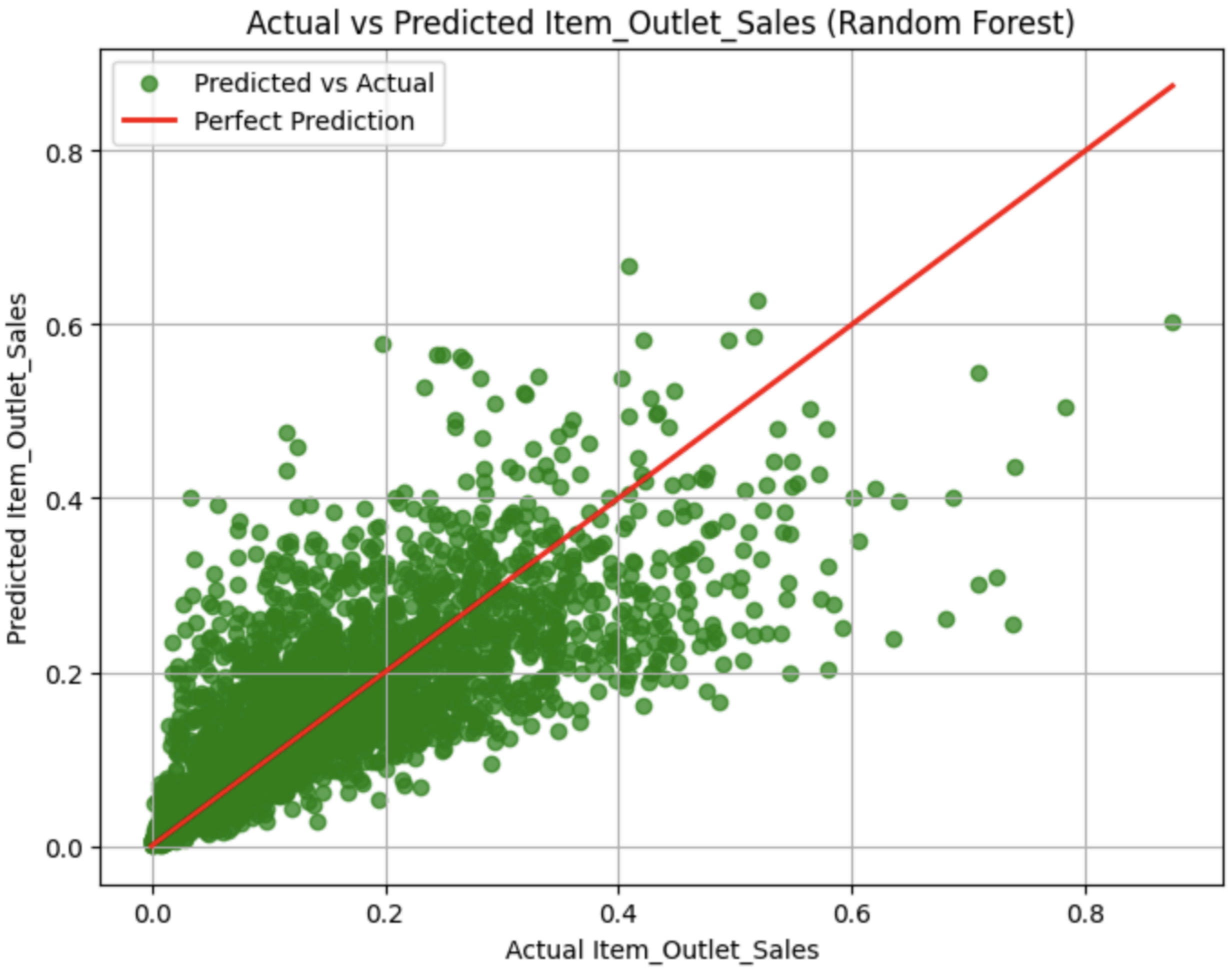

Fig2.2: Correlation matrix of dataset features after encoding

The scatter points are better distributed along the red line compared to the Decision Tree but less tightly clustered than Linear Regression.

It captures the trends well but has some spread, especially at higher sales values.

C. **Decision Tree:** A Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the data into smaller subsets based on feature values to make predictions.

The model recursively splits the data based on the feature that provides the best separation.

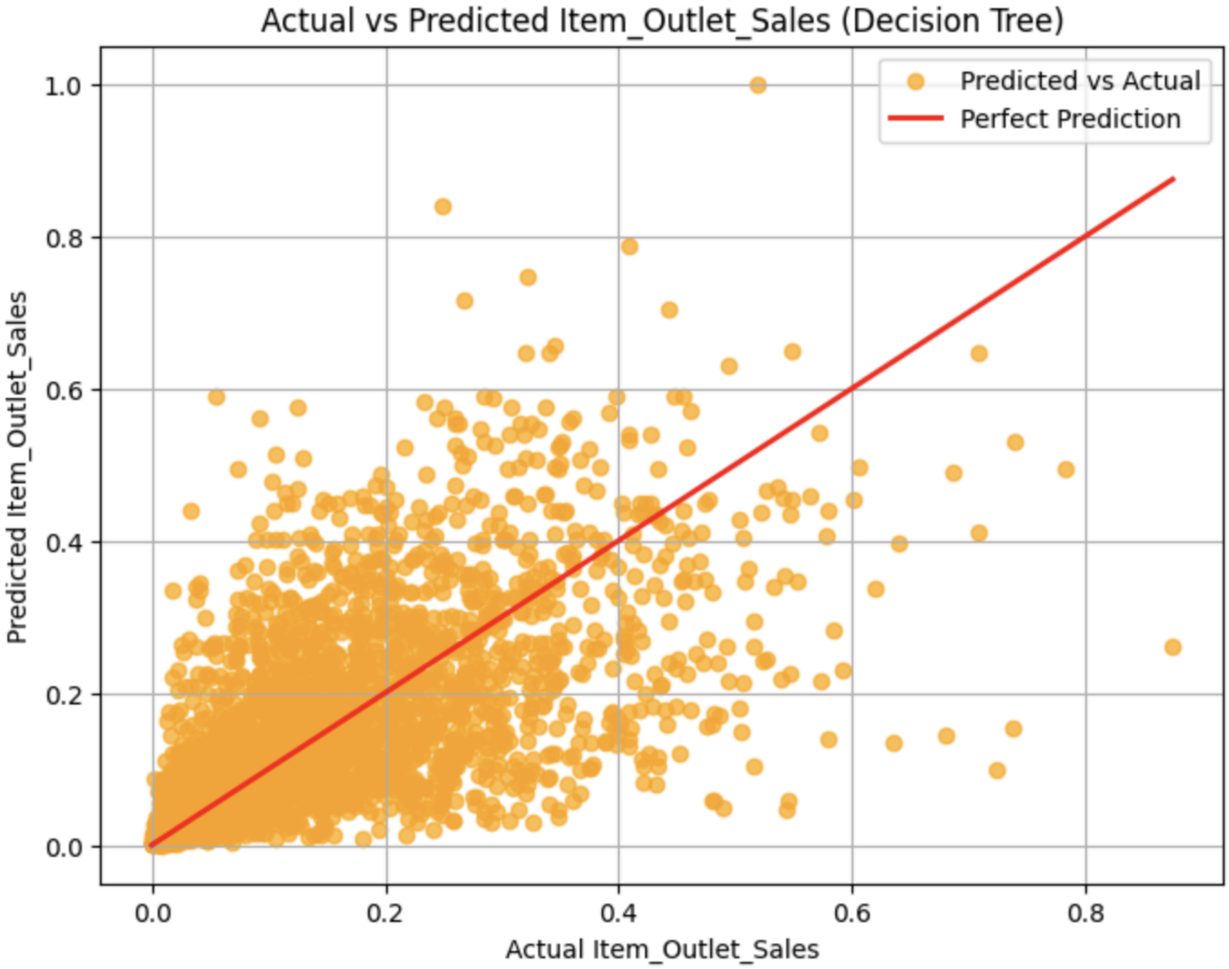

Fig2.3: Correlation matrix of dataset features after encoding

The scatter points show significant dispersion from the red line.

The model overfits, struggling to generalize well to new data, leading to less accurate predictions.

D. **XGBoost:** XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting, widely used for classification and regression tasks. XGBoost builds decision trees sequentially, with each tree trying to correct the errors of the previous one. It uses gradient descent to minimize errors.

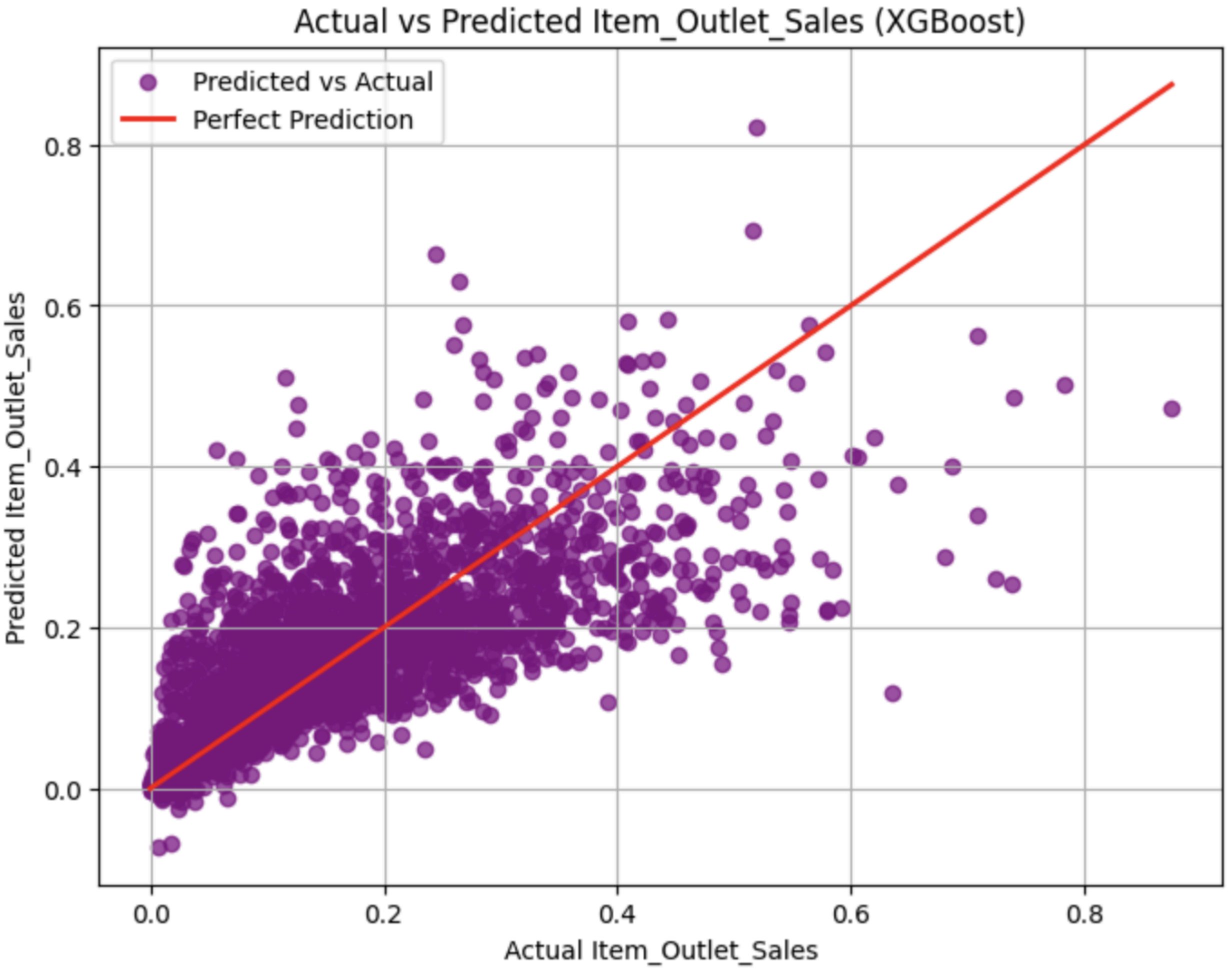

Fig2.4: Correlation matrix of dataset features after encoding

The scatter points are better distributed along the red line compared to the Decision Tree but less tightly clustered than Linear Regression.

It captures the trends well but has some spread, especially at higher sales values.

Model Selection / Comparison Analysis

R^2, RMSE, MAE and MSE score comparison:

A. **Linear Regression:**

R^2 Score: 0.5681260906679102

Root Mean Squared Error (RMSE): 0.08426056758866686

Mean Squared Error (MSE): 0.007099843250364296

Mean Absolute Error (MAE): 0.062030351285813766

B. **Random Forest:**

R^2 Score: 0.5189003350709548

Root Mean Squared Error (RMSE): 0.08893309845041511

Mean Squared Error (MSE): 0.007909095999991227

Mean absolute error (MAE): 0.062030351285813766

C. **Decision Tree:**

R^2 Score: 0.14032866539411515

Root Mean Squared Error (RMSE): 0.11888090952338914

Mean Squared Error (MSE): 0.014132670649108236

Mean Absolute Error (MAE): 0.08269784918565957

D. XGBoost:

R² Score: 0.5039298160363641

Root Mean Squared Error (RMSE): 0.09030617727642055

Mean Squared Error (MSE): 0.008155205654280295

Mean Absolute Error (MAE): 0.06258724943295198

Model	R² Score (Higher = Better)	RMSE (Lower = Better)	MSE (Lower = Better)	MAE (Lower = Better)
Linear Regression	0.5681	0.0843	0.0071	0.0620
Random Forest	0.5189	0.0889	0.0079	0.0620
Decision Tree	0.1403	0.1189	0.0141	0.0827
XGBoost	05039	0.0082	0.0082	0.0626

Table1: Correlation matrix of dataset features after encoding

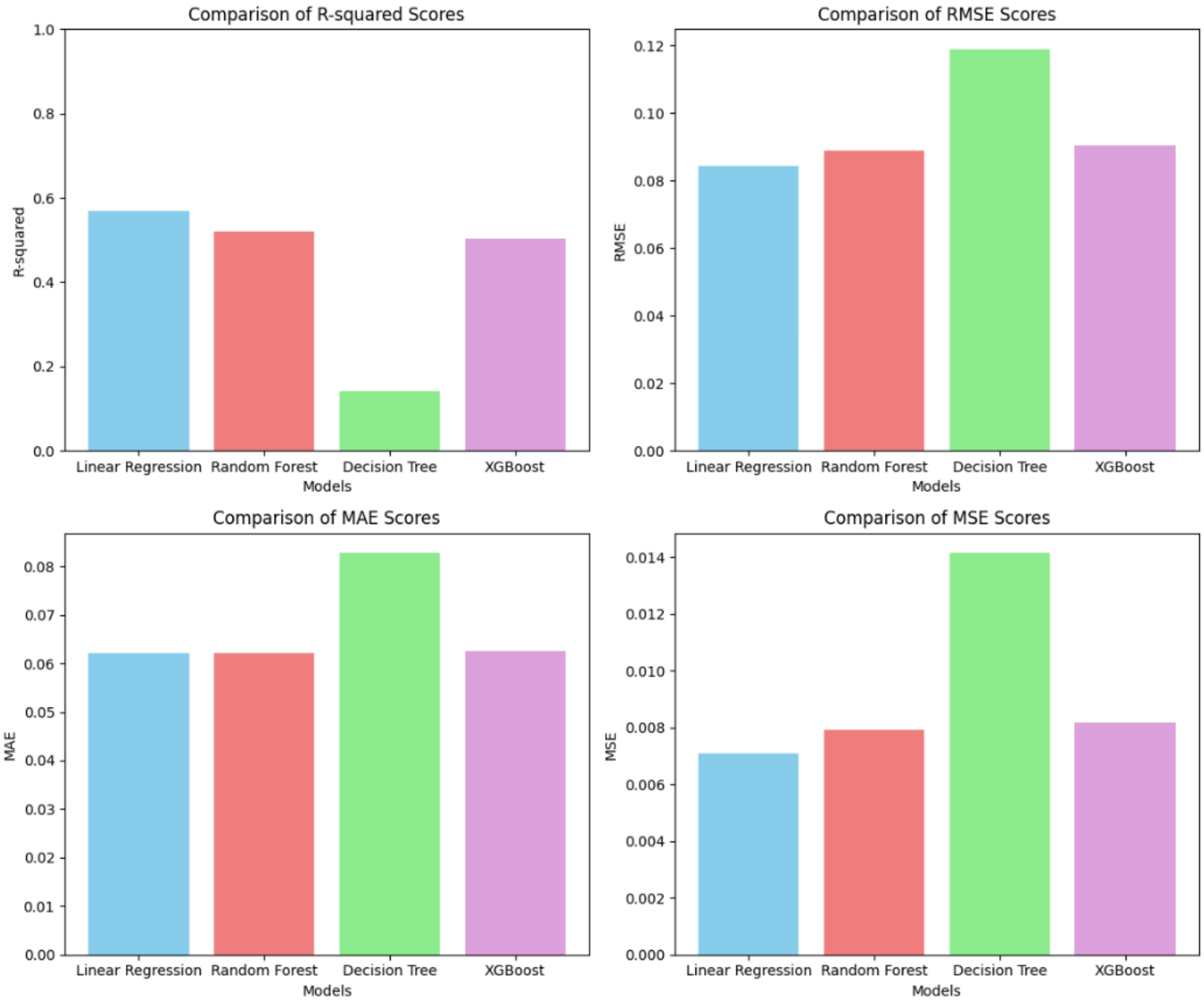

Fig3: Correlation matrix of dataset features after encoding

Observations:

From the table, we can see that **Linear Regression** has the highest R² score (0.5681). It indicates the most variance in the target variable. Also, **Linear Regression** has the lowest RMSE

(0.0843), MSE (0.0071), and MAE (0.0620). Which makes it the most accurate model among these four models based on error metrics.

However, consider the simplicity of the dataset and whether a more complex model like Random Forest or XGBoost might improve performance with hyperparameter tuning or on a larger, more complex dataset.

Conclusion

This project's goal was to evaluate how well a few machine learning models such as XGBoost, Random Forest, Decision Tree and Linear Regression performed on the given dataset. It utilizes evaluation measures like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R2 Score. The model that performed most efficiently was linear regression which had the lowest RMSE (0.0843), MSE (0.0071) and MAE (0.0620) and the greatest R2 score (0.5681). As a result, Linear Regression was able to produce predictions that were most precise with the least amount of err**or.** Random Forest and XGBoost did quite well. However in this case, Linear Regression outperformed them marginally in terms of R2 scores and error metrics. The Decision Tree model performed the worst with the greatest error metrics and the lowest R2 score (0.1403). It suggests that it was less useful for this particular dataset. The best model for this dataset was determined to be linear regression because of its ease of use, readability and excellent results. However, On bigger and more complicated datasets or with further feature engineering, Random Forest and XGBoost could perform better. This project highlights the importance of evaluating multiple models and using appropriate performance metrics to select the best model for a specific problem.