Big Mart Sales Prediction

# Using Python and Machine Learning

## Table of Contents

- Dataset Overview  
- Feature Engineering  
- Exploratory Data Analysis (EDA)  
- Scope Of Work  
- Methodology  
- Tools and Technologies  
- Model Building  
- Model Evaluation and Results  
- Conclusion and Future Work

## 1. Introduction

In today's competitive retail market, accurately predicting sales is crucial for optimizing inventory, resource allocation, and marketing strategies. This "Big Mart Sales Prediction" project uses Python and machine learning techniques to forecast sales across various Big Mart outlets. By analyzing historical data, the goal is to build a predictive model to provide actionable insights for data-driven decision-making.

## 2. Objectives

The primary objectives of the project are:  
- Build a predictive model to estimate sales for various products at different Big Mart outlets.  
- Identify key factors influencing sales (e.g., product type, price, store type).  
- Handle missing data and perform feature engineering to improve model accuracy.  
- Optimize models like XGBoost for better performance.  
- Ensure the model is robust through data splitting techniques to avoid overfitting.  
- Deploy a scalable solution that can process real-time data for continuous sales predictions.

## 3. Dataset Overview

The dataset includes sales transactions at different Big Mart outlets. Key columns are:  
- Item\_Identifier: Unique item ID.  
- Item\_Weight: Weight of the item.  
- Item\_Fat\_Content: Indicates whether the item is low-fat or regular.  
- Outlet\_Identifier: Unique store ID.  
- Outlet\_Size: Size of the store (Small, Medium, High).  
- Item\_Outlet\_Sales: Sales value of the item (target variable).

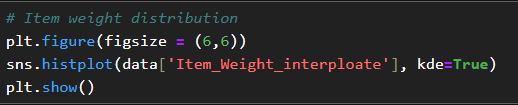
## 4. Feature Engineering

Step 1: Handling Missing Data  
- Imputed missing values for Item\_Weight using the mean and for Outlet\_Size using the mode.  
  
Step 2: Encoding Categorical Variables  
- Ordinal-encoded categorical columns like Item\_Fat\_Content, Outlet\_Size, and Outlet\_Location\_Type.  
  
Step 3: Feature Scaling  
- Standardized features like Item\_Weight and Item\_MRP using the StandardScaler method.

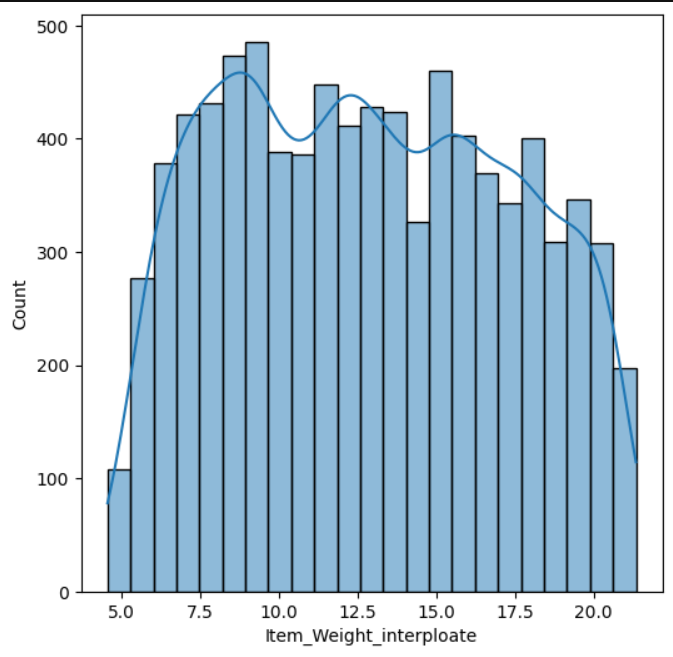
## 5. Exploratory Data Analysis (EDA)

- Item Weight Distribution: A histogram with a Kernel Density Estimate (KDE) was plotted to understand the spread and common values of Item\_Weight\_interpolate

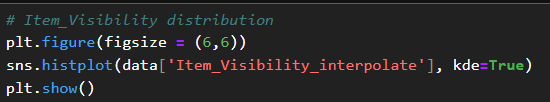
Code:-



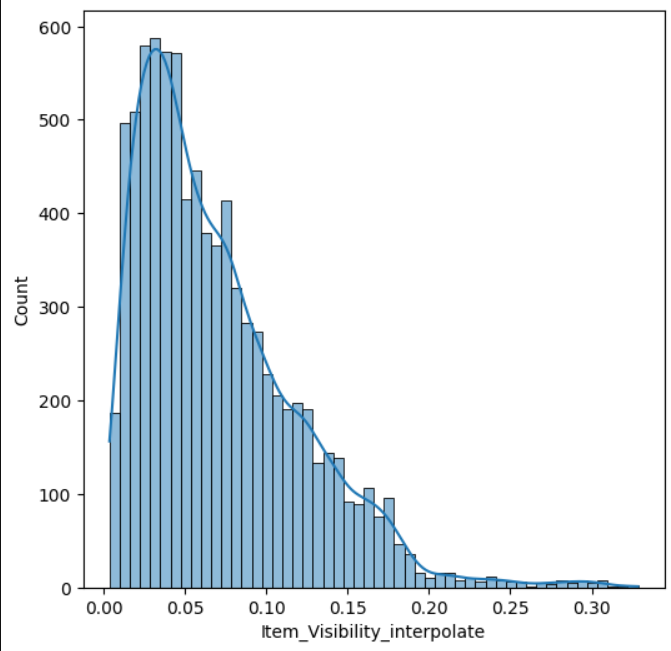
Output:-

  
- Item Visibility Distribution: A histogram with KDE was used to analyze how frequently items are visible to customers.

Code:-

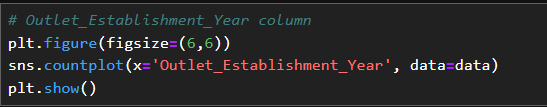


Output:-

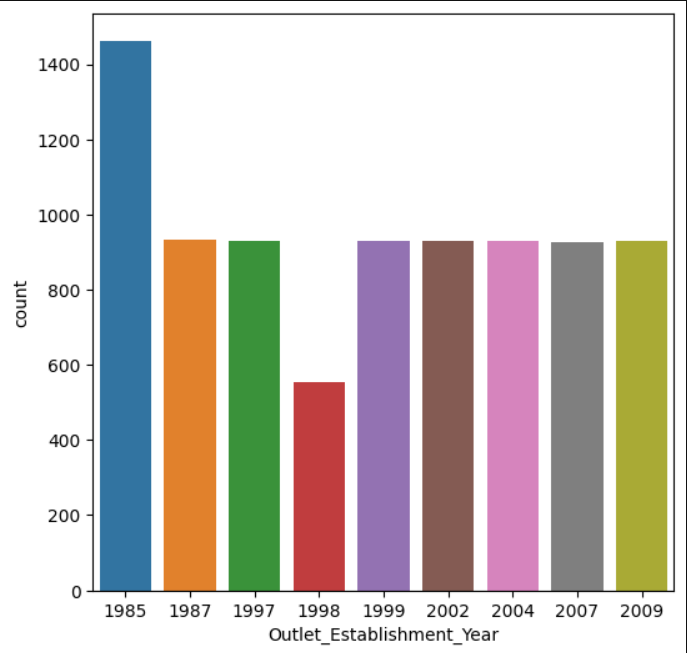


- Outlet Establishment Year: A bar chart highlighted outlet growth, with the highest number of stores established in 1985.

Code:-

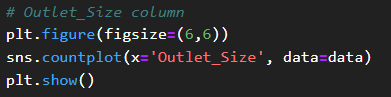


Output:-

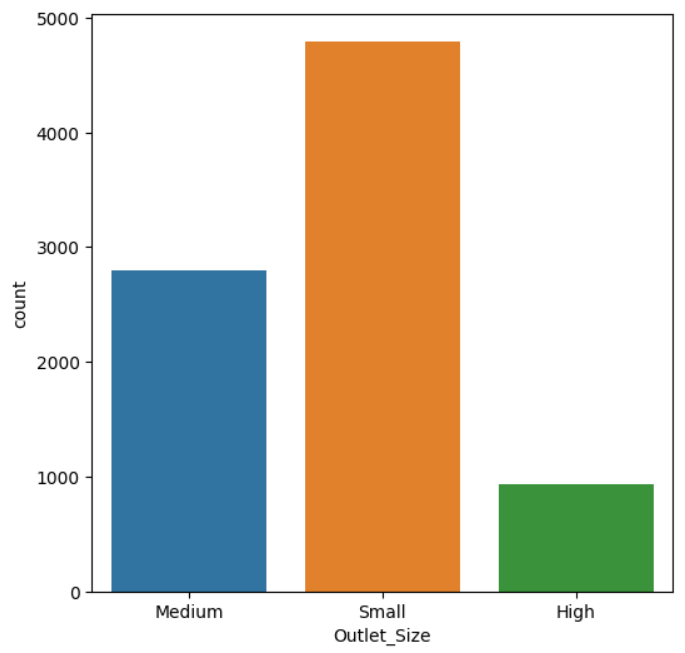


- Outlet Size: Bar charts indicated that small outlets are the most prevalent, followed by medium and high outlets.

Code:-



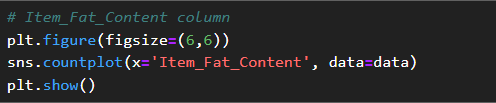
Output:-



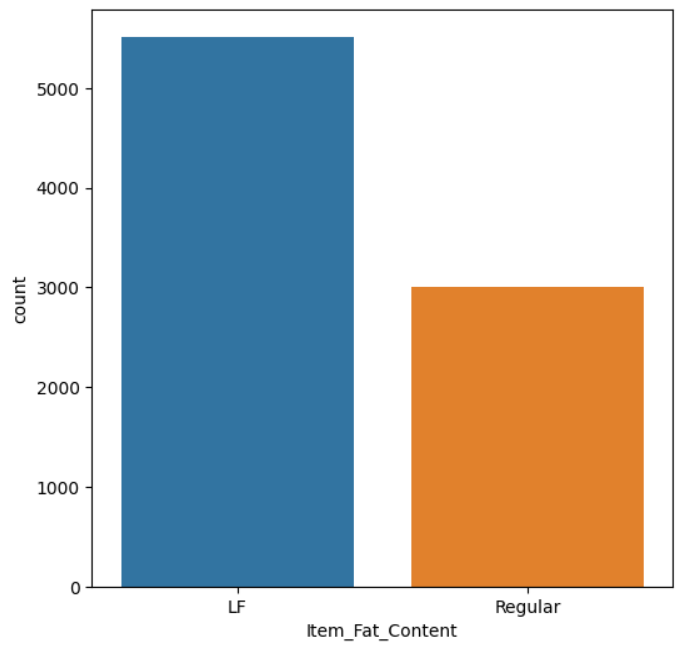
- Item\_Fat\_Contentent :- The bar chart shows the count of items for each category in the Item\_Fat\_Content column. There are two categories:

**LF (Low Fat)**: , indicating that there are more low-fat items in the dataset.

**Regular**:, this category has a lower count compared to the LF category.  
Code:-



Output:-



## 6. Scope Of Work

The steps undertaken in this project include:  
1. Data Collection & Preparation: Gather and clean historical sales data.  
2. Feature Engineering: Develop features relevant to sales predictions.  
3. Model Development: Build predictive models using algorithms like XGBoost and Random Forest.  
4. Model Optimization: Fine-tune models to improve accuracy.  
5. Evaluation & Validation: Use cross-validation to test model robustness.  
6. Deployment: Create a deployable model for real-time predictions.

## 7. Methodology

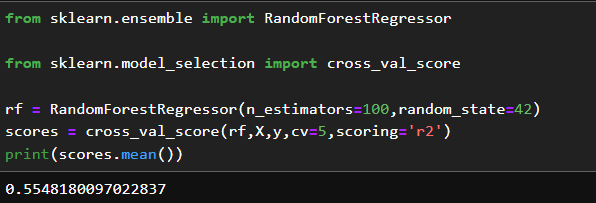
1. Data Collection: The dataset was sourced from Kaggle’s Big Mart Sales dataset.  
2. Data Preprocessing: Handled missing values, standardized data, and removed irrelevant features.  
3. Exploratory Data Analysis (EDA): Relationships between variables were explored using visual trends.  
4. Feature Engineering: Categorical features were encoded, and new features were created.  
5. Model Selection: XGBoost was chosen for the final model.  
6. Model Tuning: Hyperparameter optimization was conducted using Grid Search for XGBoost.  
7. Evaluation: Model performance was measured using MAE and RMSE.

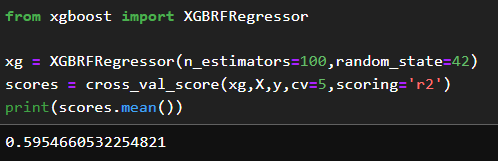
## 8. Tools and Technologies

- Python Libraries:  
 - Pandas for data manipulation.  
 - NumPy for numerical computations.  
 - Matplotlib/Seaborn for data visualization.  
 - Scikit-learn for machine learning.  
 - XGBoost for optimized gradient boosting.  
- Environment: Jupyter Notebook.  
- Version Control: Git/GitHub.

## 9. Model Building

1. Random Forest Regressor:  
 - Built 100 decision trees and evaluated using 5-fold cross-validation.  
 - Average R² score achieved: 0.5548.  
Code:-

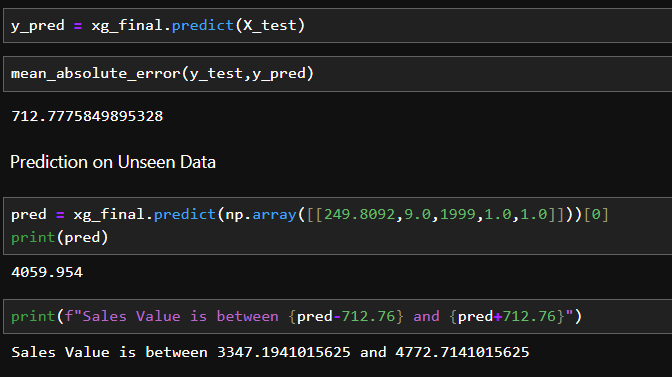
  
2. XGBoost RF Regressor:  
 - XGBoost achieved a better R² score of 0.5954, outperforming Random Forest.

  
  
XGBoost was chosen over Random Forest due to better accuracy and more effective handling of complex relationships.

## 10. Model Evaluation and Results

The XGBoost model produced a Mean Absolute Error (MAE) of 712.77 on the test data. Predictions on unseen data showed that the sales value for a sample item could range from 3347 to 4772.

Code:-



## 11. Conclusion and Future Work

The XGBoost model proved to be the most effective for predicting Big Mart sales. Future work can focus on:  
- Hyperparameter tuning for further improving model performance.  
- Incorporating additional features like seasonal trends and customer demographics.  
- Exploring advanced techniques such as deep learning models for more sophisticated predictions.