**Title:** Big Sales Prediction using Random Forest Regressor | Machine Learning Project

---

**Objective:**
The objective of this project is to develop a predictive model using a Random Forest Regressor to accurately forecast sales based on various product and outlet attributes. This will enable businesses to make data-driven decisions, optimize inventory management, and improve sales strategies.

**Dataset:**
https://github.com/YBIFoundation/Dataset/raw/main/Big%20Sales%20Data.csv

**Project outline**

**1. Understanding the Dataset**

You are working with a dataset that contains 12 variables, with the following columns:

1. **Item_Identifier:** Unique ID for each item
2. **Item_Weight:** Weight of each item
3. **Item_Fat_Content:** The fat content of the item (Low Fat, Regular, etc.)
4. **Item_Visibility:** How much of the product is visible to consumers
5. **Item_Type:** Category of the item (e.g., Baking Goods, Snack Foods, etc.)
6. **Item_MRP:** Maximum Retail Price of the item
7. **Outlet_Identifier:** Unique ID for each retail outlet
8. **Outlet_Establishment_Year:** Year the outlet was established
9. **Outlet_Size:** Size of the outlet (Small, Medium, High)

10. **Outlet_Location_Type: Type of location (Tier 1, Tier 2, Tier 3)**
11. **Outlet_Type: Type of outlet (e.g., Supermarket Type 1, Grocery store**
12. **Item_Outlet_Sales: Target variable, representing sales for each item in each outlet**

**2. Libraries Required**

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

**3. Loading the Dataset**

**You can use the dataset from the given URL or a local path.**

```
df = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Big%20Sales%20Data.csv')
```

**4. Exploring the Data**

**Get the first five rows:**

```
df.head()
```

**General information about the dataset:**

**df.info()**

**Check for missing values:**

**df.isnull().sum()**

**Get the summary statistics:**

**df.describe()**

**View column names:**

**df.columns**

**5. Handling Missing Values**

**For Item_Weight, you can fill missing values using the mean weight for the corresponding Item_Type:**

```
df['Item_Weight'] =
df['Item_Weight'].fillna(df.groupby('Item_Type')['Item_Weight'].transform('mean'))
```

**6. Handling Categorical Variables**

**Convert categorical variables to numerical values for the model:**

```
df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({'LF': 'Low Fat', 'reg': 'Regular', 'low fat': 'Low Fat'})
```

```
df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({'Low Fat': 0, 'Regular': 1})
```

**# Similarly, encode other categorical variables**

**df['Outlet_Size'] = df['Outlet_Size'].replace({'Small': 0, 'Medium': 1, 'High': 2})**

**df['Outlet_Location_Type'] = df['Outlet_Location_Type'].replace({'Tier 1': 0, 'Tier 2': 1, 'Tier 3': 2})**

**df['Outlet_Type'] = df['Outlet_Type'].replace({'Grocery Store': 0, 'Supermarket Type1': 1, 'Supermarket Type2': 2, 'Supermarket Type3': 3})**

**7. Defining X (Features) and y (Target)**

**Target variable y is Item_Outlet_Sales, and the features X will exclude this column.**

**X = df.drop(['Item_Outlet_Sales', 'Item_Identifier', 'Outlet_Identifier'], axis=1)**

**y = df['Item_Outlet_Sales']**

**8. Train-Test Split**

**Split the dataset into training and testing sets.**

**X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)**

**9. Building the Random Forest Regressor Model**

**rf = RandomForestRegressor(n_estimators=100, random_state=42)**

**rf.fit(X_train, y_train)**

**10. Model Evaluation**

**Evaluate the performance of the model using Mean Squared Error.**

**y_pred = rf.predict(X_test)**

**mse = mean_squared_error(y_test, y_pred)**

**print(f"Mean Squared Error: {mse}")**

**This should give you an overall structure for predicting sales using a Random Forest model. You can expand on this with feature engineering, hyperparameter tuning, and further analysis of the results.**