Big Mart Sales Prediction – Full Documentation

This document provides a complete explanation of the Big Mart Sales Prediction machine learning project. It includes an overview of the workflow, models used, preprocessing steps, feature engineering, evaluation, and the final code extracted from the notebook.

■ Machine Learning Models Used

The notebook uses the following ML model:
- **XGBRegressor** – an optimized gradient boosting model widely used for regression tasks. It handles non-linear relationships, missing values, and performs well with tabular data.

■ Steps Followed in This ML Project

Below are the standard ML pipeline steps applied in the project:

1. Problem Definition

Predict the sales of Big Mart outlets based on historical data and store/item-related attributes.

2. Data Loading

The dataset is loaded using Pandas, allowing exploration, cleaning, and manipulation.

3. Exploratory Data Analysis (EDA)

- Understanding feature distributions - Missing value detection - Checking categorical vs numerical features - Identifying anomalies or outliers

4. Data Cleaning

- Handling missing values - Fixing inconsistent labels - Replacing zero or null values - Standardizing categorical labels

5. Feature Engineering

- Encoding categorical variables - Creating new meaningful variables - Scaling or transforming numerical features (if needed)

6. Model Selection

XGBoost (XGBRegressor) was chosen due to: - High accuracy - Ability to model complex relationships - Robustness to missing data - Feature importance insights

7. Model Training

The model is trained on the processed dataset using train-test split to avoid overfitting.

8. Model Evaluation

Common regression metrics include:
- RMSE (Root Mean Squared Error) - MAE (Mean Absolute Error) - R^2 Score

9. Prediction
After training, predictions are generated for test samples and compared with actual values.

10. Saving the Model (Optional)
Models are often saved using joblib or pickle for reuse.

--

■ Complete Code (Extracted from Notebook)
Below is the entire code from the Jupyter Notebook.

Full Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder #used to convert categorical (text) values into
 numeric values.

from sklearn.model_selection import train_test_split

from xgboost import XGBRegressor

from sklearn import metrics

big_mart_data = pd.read_csv('/content/drive/MyDrive/Data Science/Projects/12 Big Mart Sales Pr
ediction/Train.csv')

big_mart_data.head()

big_mart_data.shape

big_mart_data.info()

#Categorical Features:

 #Item_Identifier

 #Item_Fat_Content

 #Item_Type

 #Outlet_Identifier

 #Outlet_Size

 #Outlet_Location_Type

 #Outlet_Type
```

```
#Checking missing values

big_mart_data.isnull().sum()

big_mart_data['Item_Weight'].mean()

big_mart_data['Item_Weight'].fillna(big_mart_data['Item_Weight'].mean(), inplace=True)

big_mart_data.isnull().sum()

mode_of_outlet_size = big_mart_data.pivot_table(values='Outlet_Size', columns='Outlet_Type', a
ggfunc=(lambda x: x.mode()[0]))

mode_of_outlet_size

missing_values = big_mart_data['Outlet_Size'].isnull()

missing_values

big_mart_data.loc[missing_values, 'Outlet_Size'] = big_mart_data.loc[missing_values, 'Outlet_T
ype'].apply(lambda x: mode_of_outlet_size.loc['Outlet_Size', x])

big_mart_data.isnull().sum()

big_mart_data.describe()

sns.set()

#Item_Weight distribution

plt.figure(figsize=(6,6))

sns.distplot(big_mart_data['Item_Weight'])

plt.show()

#Item_Visibility distribution

plt.figure(figsize=(6,6))

sns.distplot(big_mart_data['Item_Visibility'])

plt.show()

#Item_MRP distribution

plt.figure(figsize=(6,6))

sns.distplot(big_mart_data['Item_MRP'])

plt.show()

#Item_Outlet_Sales distribution

plt.figure(figsize=(6,6))

sns.distplot(big_mart_data['Item_Outlet_Sales'])

plt.show()

#Outlet_Establishment_Year distribution

plt.figure(figsize=(6,6))

sns.countplot(x='Outlet_Establishment_Year', data=big_mart_data)

plt.show()
```

```
#Item_Fat_Content distribution
plt.figure(figsize=(6,6))
sns.countplot(x='Item_Fat_Content', data=big_mart_data)
plt.show()

#Item_Type distribution
plt.figure(figsize=(25,6))
sns.countplot(x='Item_Type', data=big_mart_data)
plt.show()

#Outlet_Size distribution
plt.figure(figsize=(6,6))
sns.countplot(x='Outlet_Size', data=big_mart_data)
plt.show()

big_mart_data.head()

big_mart_data['Item_Fat_Content'].value_counts()

big_mart_data.replace({
    'Item_Fat_Content': {'low fat':'Low Fat', 'LF':'Low Fat', 'reg':'Regular'}
}, inplace=True)

big_mart_data['Item_Fat_Content'].value_counts()

encoder = LabelEncoder()

 #Item_Identifier
 #Item_Fat_Content
 #Item_Type
 #Outlet_Identifier
 #Outlet_Size
 #Outlet_Location_Type
 #Outlet_Type
big_mart_data['Item_Identifier'] = encoder.fit_transform(big_mart_data['Item_Identifier'])
big_mart_data['Item_Fat_Content'] = encoder.fit_transform(big_mart_data['Item_Fat_Content'])
big_mart_data['Item_Type'] = encoder.fit_transform(big_mart_data['Item_Type'])
big_mart_data['Outlet_Identifier'] = encoder.fit_transform(big_mart_data['Outlet_Identifier'])
big_mart_data['Outlet_Size'] = encoder.fit_transform(big_mart_data['Outlet_Size'])
big_mart_data['Outlet_Location_Type'] = encoder.fit_transform(big_mart_data['Outlet_Location_T
ype'])
big_mart_data['Outlet_Type'] = encoder.fit_transform(big_mart_data['Outlet_Type'])

big_mart_data.head()

x = big_mart_data.drop(['Item_Outlet_Sales'], axis=1)
```

```
y = big_mart_data['Item_Outlet_Sales']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
x.shape, x_train.shape, x_test.shape, y_train.shape, y_test.shape

regressor = XGBRegressor()
regressor.fit(x_train, y_train)

#prediction on training data
training_data_prediction = regressor.predict(x_train)

#R squared value
r2_train = metrics.r2_score(y_train, training_data_prediction)
r2_train # if it is close to 1 it is good

#prediction for testing data
test_data_prediction = regressor.predict(x_test)

#R squared value
r2_test = metrics.r2_score(y_test, test_data_prediction)
r2_test

input_data = x_test.iloc[1]

predictions = regressor.predict([input_data])
predictions

y_test
```

Author: Satyam Gajjar