

Final Report

On

“Big Mart Sales Prediction”

Submitted To



School Of Data Science And Forecasting DAVV,
Indore(M.P)

Submitted By :

- Himanshu Jain
- Tarun Choudhary
- Tushar Sonp

Under The Guidance Of :

Prof. Vandit Hedau

Tech Event Organizer :

- Krati Vyas
- Hitesh Kumawat

ABSTRACT

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be applied in different areas and trained to match the expectations of management so that accurate steps can be taken to achieve the organization's target. In this paper, the case of Big Mart, a one-stop-shopping center, has been discussed to predict the sales of different types of items and for understanding the effects of different factors on the items' sales. Taking various aspects of a dataset collected for Big Mart, and the methodology followed for building a predictive model, results with high levels of accuracy are generated, and these observations can be employed to make decisions to improve sales.

INDEX

S.NO	TITLE	PAGE NO.
1	Introduction	4
2	General Description	4
3	Tool Used	5
4	Architecture	6
5	Data Description	7
6	Observation of data	7 - 9
7	Process Flow	9 - 14
8	Project Implementation	14 – 15
9	Conclusion	16
10	Future Scope	16
11	Reference	16

❖ INTRODUCTION: -

In today's modern world, huge shopping centers such as big malls and marts are recording data related to sales of items or products with their various dependent or independent factors as an important step to be helpful in the prediction of future demands and inventory management. The dataset built with various dependent and independent variables is a composite form of item attributes, data gathered using a customer, and also data related to inventory management in a data warehouse. The data is thereafter refined to get accurate predictions and gather new as well as interesting results that shed new light on our knowledge concerning the task's data. This can then further be used for forecasting future sales using employing machine learning algorithms such as random forests and simple or multiple linear regression models.

❖ GENERAL DESCRIPTION: -

➤ **Product Perspective: -**

The Store Sales Prediction is an ML-based Web Application that Can predict future product demand by analyzing records. It will give the number that will be the measure of product sales.

➤ **Problem Statement: -**

The data scientists at BigMart have collected sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

So, the idea is to find out the properties of a product, and store which impacts the sales of a product. Let's think about some of the analyses that can be done

➤ **Proposed Solution: -**

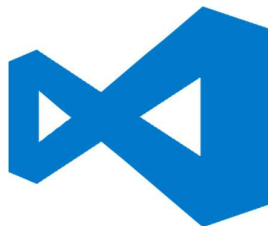
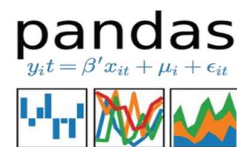
We will use performed EDA to find the important relation between different attributes and will use a machine-learning algorithm to predict the future sales demand. The client will be filled the required feature as input and will get results through the web application. The system will get features and it will be passed into the backend where the features will be validated and pre-processed and then it will be passed to a hyperparameter tuned machine learning model to predict the outcome.

➤ **Data Requirements: -**

The data is required for the building of the project is already available on the dashboard. The Store Sales Prediction data is recorded many product descriptions along with past sales quantity. For building the ml model we will use the dataset that is given. The data consists of 8523 rows and various information about products like product id, product category, store id, store location, etc.

❖ TOOL USED: -

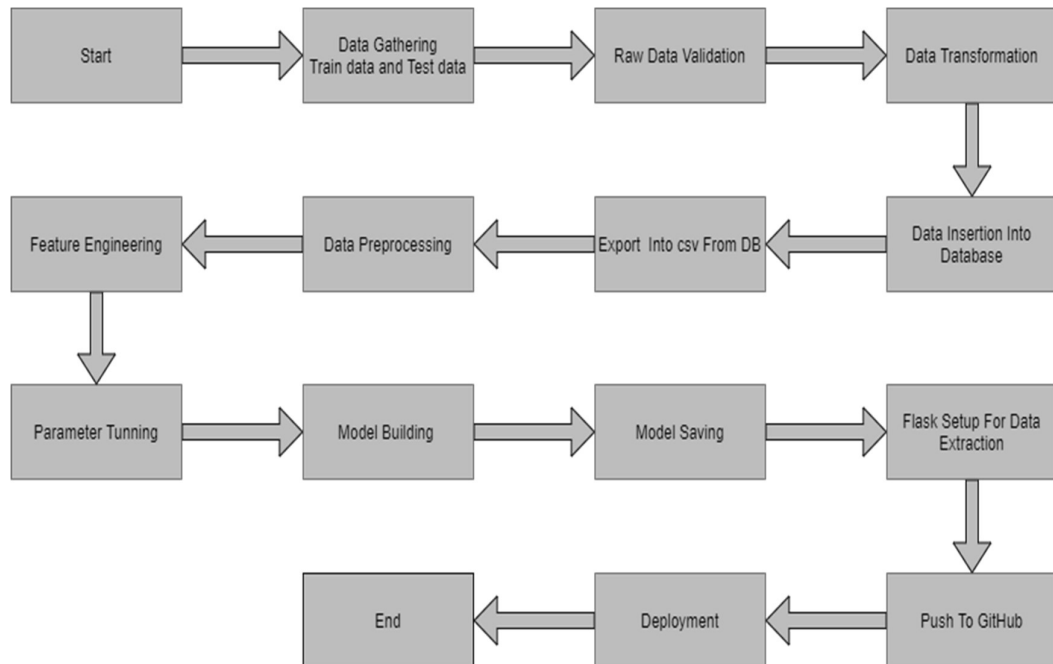
The programming language is Python is used here, also we will use some other python-based libraries like, for ml, we will use Scikit-Learn library, for data manipulation we will use pandas, for custom APIs creation Flask web frameworks. Visual Studio Code is used as python IDE for all modular coding and custom APIs creation. And storing all code files for publically available we will use GitHub. For the deployment process, we will be using Heroku cloud platforms for hosting our application.



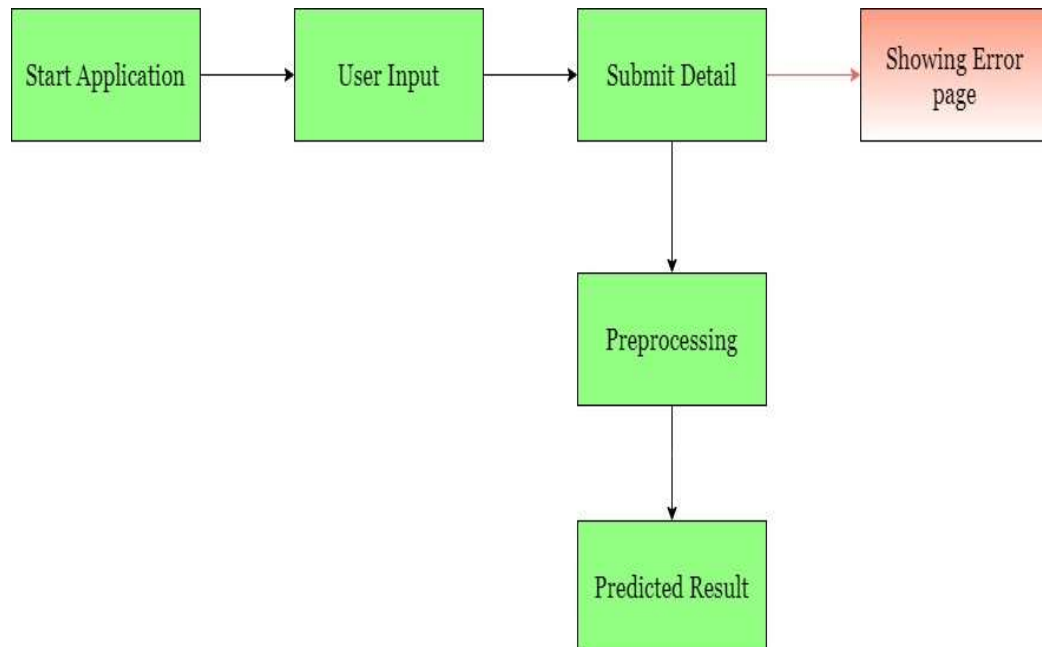
❖ ARCHITECTURE: -

We will be using the following architecture for this project

➤ Model Training and Evaluation: -



➤ Deployment architecture: -



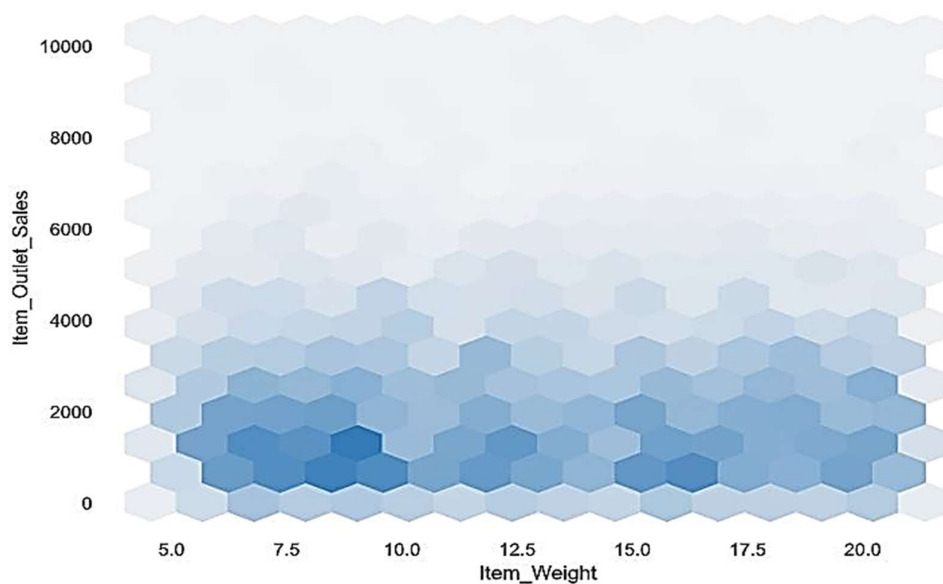
❖ DATA DESCRIPTION: -

Name	Data Type	Description
Item_Identifier	String	Unique product ID
Item_Weight	Float	Weight of product
Item_Fat_Content	String	Whether the product is low fat or not
Item_Visibility	Float	The % of a total display area of all products in a store allocated to the particular product
Item_Type	String	The category to which the product belongs
Item_MRP	Float	Maximum Retail Price (list price) of the product
Outlet_Identifier	String	Unique store ID
Outlet_Establishment_Year	Integer	The year in which the store was established
Outlet_Size	String	The size of the store in terms of ground area covered
Outlet_Location_Type	String	The type of city in which the store is located
Outlet_Type	String	Whether the outlet is just a grocery store or some sort of supermarket
Item_Outlet_Sales	Float	Sales of the product in the particular store. This is the outcome variable to be predicted.

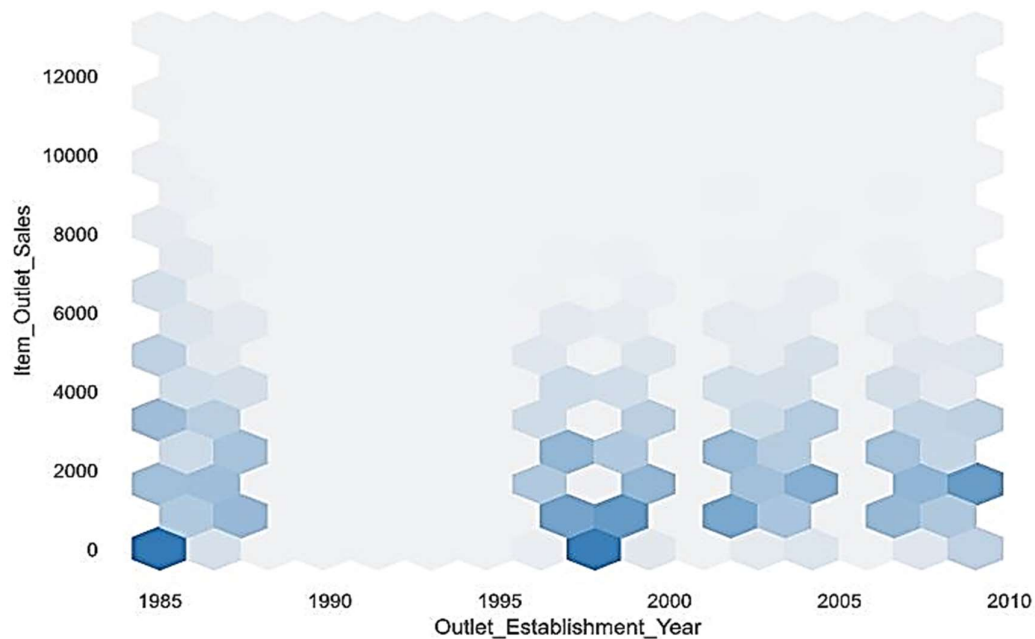
❖ OBSERVATION OF DATA: -

Correlation is used to understand the relation between a target variable and predictors. In this work, Item-Sales is the target variable, and its correlation with other variables is observed.

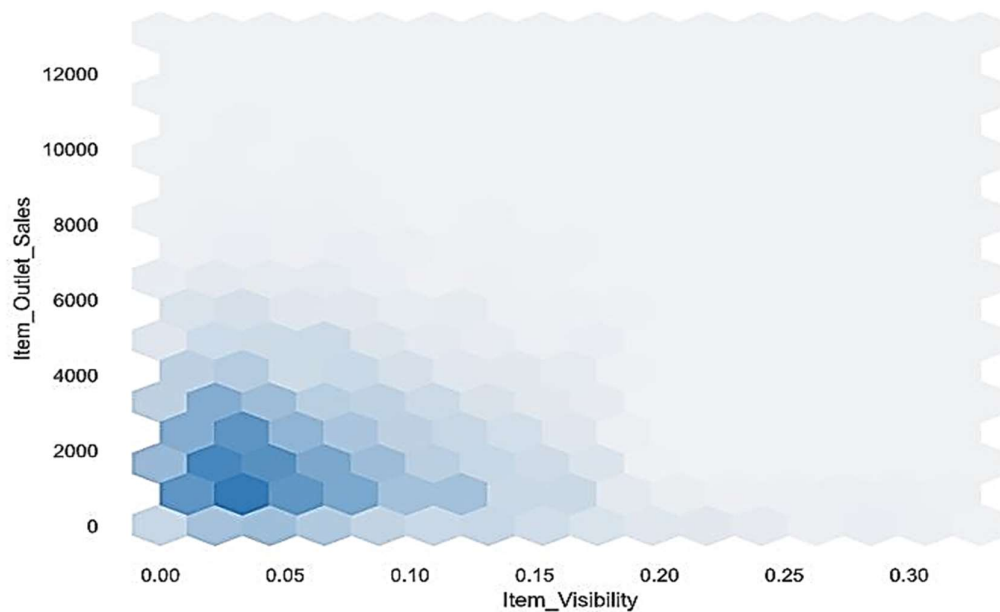
Considering case 1 of Item-Weight, the feature item weight is shown to have a low correlation with the target variable Item-Outlet-Sales in below Fig.



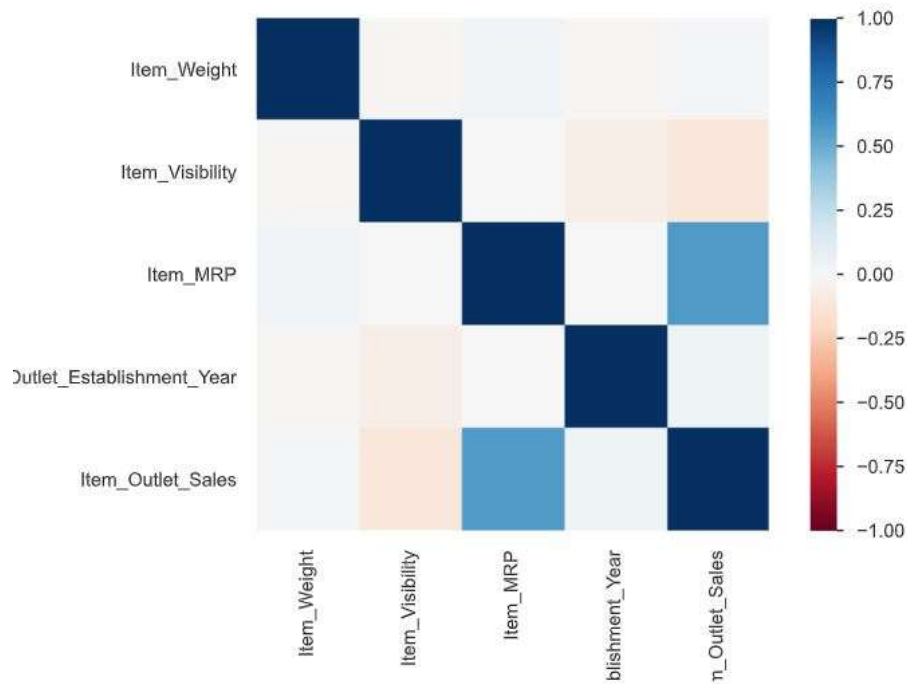
Considering case 2 There is no significant relationship found between the year of store establishment and the sales for the items. Values can also be combined into variables that classify them into periods and give meaningful results.



Considering case 3 The place where an item is placed in a store, referred to as Item visibility affects the sales. However, the plot chart shows that the flow is on the opposite side. One of the reasons might be that daily used products don't need high visibility. However, there is an issue that some products have zero visibility, which is quite impossible.



Correlation: -



- Item visibility is having nearly zero correlation with our dependent variable Item_Outlet_Sales and grocery store outlet type. This means that the sales are not affected by the visibility of items which is a contradiction to the general assumption of “more visibility thus, more sales”.
- Item_MRP (maximum retail price) is positively correlated with sales at an outlet, which indicates that the price quoted by an outlet plays an important factor in sales.
- Variation in MRP quoted by various outlets depends on their sales

❖ PROCESS FLOW: -

➤ Data Gathering: -

Data source: <https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data>

Train and Test data are stored in .csv format.

➤ Data Exploration: -

Let's start by loading the required libraries and data.

Import Libraries

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import pickle
from pandas_profiling import ProfileReport
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
import xgboost as xgb
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
```

Loading Data

```
In [2]: df = pd.read_csv('Train.csv')
```

then we check the missing value of every column

```
In [6]: df.isnull().sum()
```

```
Out[6]: Item_Identifier      0
Item_Weight      1463
Item_Fat_Content      0
Item_Visibility      0
Item_Type      0
Item_MRP      0
Outlet_Identifier      0
Outlet_Establishment_Year      0
Outlet_Size      2410
Outlet_Location_Type      0
Outlet_Type      0
Item_Outlet_Sales      0
dtype: int64
```

Item_weight and Outlet_size have missing values so we will impute the missing values in Item_Weight and Outlet_Size in the data cleaning section.

When we did some other observations Some of the 'Low Fat' values miscoded as 'low fat' and 'LF'. Also, some of 'Regular' are mentioned as 'regular'.

```
In [15]: df.Item_Fat_Content.value_counts()
```

```
Out[15]: Low Fat    5089
Regular    2889
LF         316
reg        117
low fat    112
Name: Item_Fat_Content, dtype: int64
```

We replace this miscoded value in data cleaning

➤ **Data Cleaning: -**

This step typically involves imputing missing values and treating outliers. Though outlier removal is very important in regression techniques, advanced tree-based algorithms are impervious to outliers.

Imputing Missing Values

We found two variables with missing values – Item_Weight and Outlet_Size. Let's impute the former by the mean weight of the particular item. And Outlet_Size with the value "Medium" of the Outlet_Size

```
In [10]: df['Item_Weight'] = df['Item_Weight'].fillna(df['Item_Weight'].mean())
```

```
In [11]: df['Outlet_Size'].unique()
```

```
Out[11]: array(['Medium', nan, 'High', 'Small'], dtype=object)
```

```
In [12]: df = df.fillna({'Outlet_Size': 'Medium'})
```

➤ **Feature Engineering: -**

Label Encoding:

Scikit-learn accepts only numerical variables, I converted all categories of nominal variables into numeric types

Let's start with coding all categorical variables as numeric using 'LabelEncoder' from sklearn's preprocessing module.

```
In [14]: lb = LabelEncoder()
```

```
In [2]: df['Outlet_Size'] = lb.fit_transform(df['Outlet_Size'])
df['Item_Fat_Content'] = lb.fit_transform(df['Item_Fat_Content'])
df['Outlet_Location_Type'] = lb.fit_transform(df['Outlet_Location_Type'])
df['Outlet_Type'] = lb.fit_transform(df['Outlet_Type'])
df['Item_Type'] = lb.fit_transform(df['Item_Type'])
df['Outlet_Identifier'] = lb.fit_transform(df['Outlet_Identifier'])
```

Standard Scaling: -

The data obtained contains features of various dimensions and scales altogether. Different scales of the data features affect the modeling of a dataset adversely. So, we did standard scaling of data by this the entire data set scales with zero mean and unit variance

Standard scaling

```
In [26]: scaler = StandardScaler()

In [27]: x_scalar = scaler.fit_transform(x)

In [28]: x_scalar
Out[28]: array([[ -0.84187169, -0.73814723, -0.97073217, ..., -0.28458121,
-1.36933384, -0.25265831],
[ -1.64170589,  1.35474328, -0.90811123, ..., -0.28458121,
 1.09156913,  1.00297245],
[  1.09855449, -0.73814723, -0.95691733, ..., -0.28458121,
-1.36933384, -0.25265831],
...,
[ -0.53424315, -0.73814723, -0.59978449, ...,  1.38127431,
-0.13888236, -0.25265831],
[ -1.33644372,  1.35474328,  1.53287976, ..., -0.28458121,
 1.09156913,  1.00297245],
[  0.45963367, -0.73814723, -0.41193591, ...,  1.38127431,
-1.36933384, -0.25265831]])
```

➤ Model Building: -

Now that we have the data ready, it's time to start making predictive models. I will take you through 3 models XgBoost, Gradient Boost, Random Forest

Before applying the algorithm, we split the data set into a two-part train and test data set

```
x_train , x_test ,y_train , y_test = train_test_split(x_scalar , y , test_size = .10,random_state = 250)
```

Then we apply algorithms to build models

First, we apply XgBoost

XgBoost

```
In [91]: xgb_model_1 = xgb.XGBRegressor(learning_rate = 0.03 ,base_score = 5)

In [92]: xgb_model_1.fit(x_train,y_train)

Out[92]: XGBRegressor(base_score=5, booster='gbtree', colsample_bylevel=1,
      colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
      gamma=0, gpu_id=-1, importance_type=None,
      interaction_constraints='', learning_rate=0.03, max_delta_step=0,
      max_depth=6, min_child_weight=1, missing=nan,
      monotone_constraints=()), n_estimators=100, n_jobs=4,
      num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
      reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
      validate_parameters=1, verbosity=None)

In [89]: y_prediction = xgb_model_1.predict(x_test)
rmse = np.sqrt(mean_squared_error(y_prediction,y_test))
rmse

Out[89]: 1031.1457485841584

In [90]: mean_absolute_error(y_prediction,y_test)

Out[90]: 746.3393163981643
```

Then we apply Gradient Boost

Gradient Boost

```
In [99]: gb_model = GradientBoostingRegressor(subsample = best_param['subsample'],
                                             n_estimators = best_param['n_estimators'],
                                             min_samples_split = best_param['min_samples_split'],
                                             min_samples_leaf = best_param['min_samples_leaf'],
                                             max_features = best_param['max_features'],
                                             max_depth = best_param['max_depth'],
                                             learning_rate = best_param['learning_rate'])
```

```
In [100]: gb_model.fit(x_train,y_train)
```

```
Out[100]: GradientBoostingRegressor(learning_rate=0.05, max_depth=4, max_features=3,
                                     min_samples_split=40, subsample=0.85)
```

```
In [101]: y_prediction = gb_model.predict(x_test)
          rmse = np.sqrt(mean_squared_error(y_prediction,y_test))
          rmse
```

```
Out[101]: 1036.2813477855773
```

```
In [102]: mean_absolute_error(y_prediction,y_test)
```

```
Out[102]: 750.224953128082
```

Then we apply Random Forest

Random Forest

```
In [56]: rf_model = RandomForestRegressor(n_estimators = best_param['n_estimators'],
                                         min_samples_split = best_param['min_samples_split'],
                                         min_samples_leaf = best_param['min_samples_leaf'],
                                         max_features = best_param['max_features'],
                                         max_depth = best_param['max_depth'],
                                         bootstrap = best_param['bootstrap'])
```

```
In [57]: rf_model.fit(x_train,y_train)
```

```
Out[57]: RandomForestRegressor(max_depth=10, max_features='sqrt', min_samples_leaf=2,
                               n_estimators=400)
```

```
In [61]: rmse = np.sqrt(mean_squared_error(y_prediction,y_test))
          rmse
```

```
Out[61]: 1033.015564841547
```

```
In [62]: mean_absolute_error(y_prediction,y_test)
```

```
Out[62]: 747.1366758007465
```

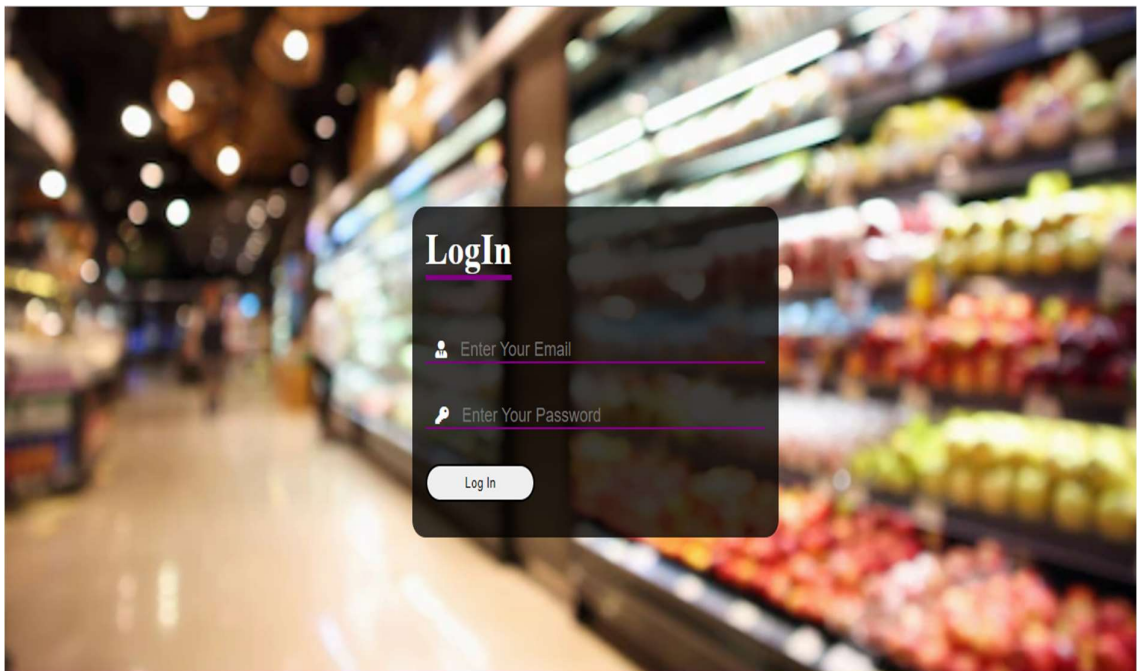
After applying the algorithm and making a model all three has a small difference in Root mean square value and mean absolute error so we select XgBoost

- **Model Saving: -**
Model is then saved using pickle library in .pkl format.
- **Flask setup for data extraction: -**
After saving the model, the API building process started using Flask. Web application creation was created here. Whatever the data user will enter and then that data will be extracted by the model to predict the prediction of sales, this is performed in this stage
- **Git Hub: -**
The whole project directory will be pushed into the GitHub repository.
- **Deployment: -**
The cloud environment was set up and the project was deployed from GitHub into the Heroku cloud platform.

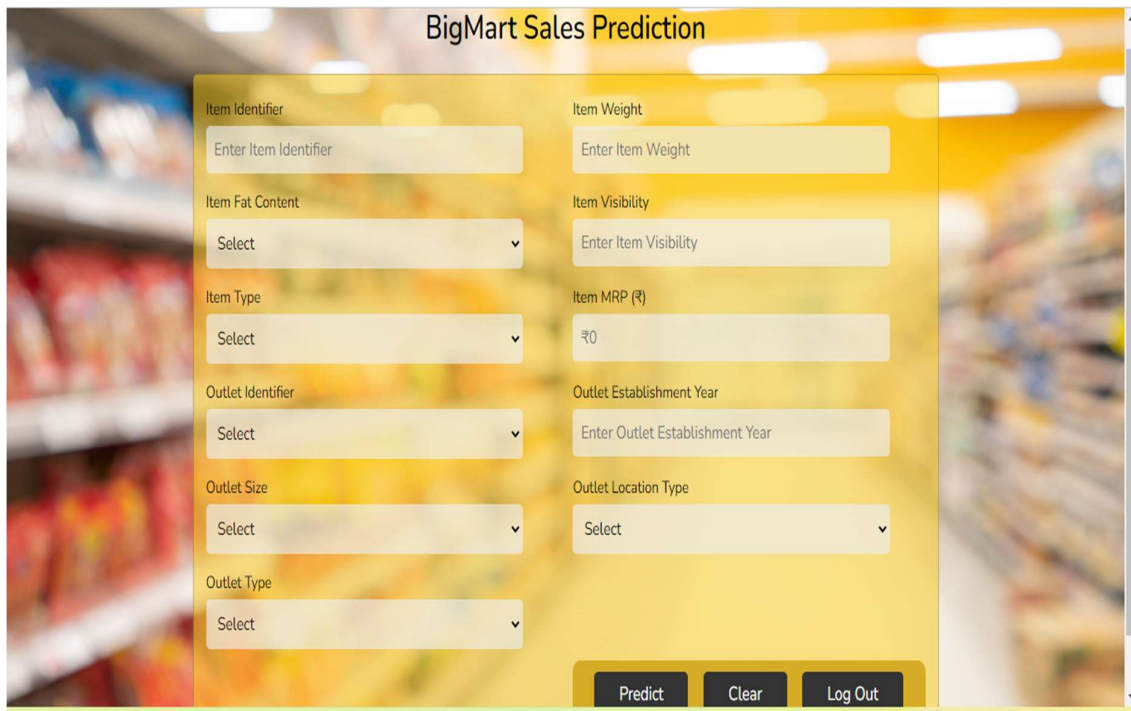
App link: <https://bigmartprediction147.herokuapp.com/>

❖ PROJECT IMPLEMENTATION: -

Login Page: -

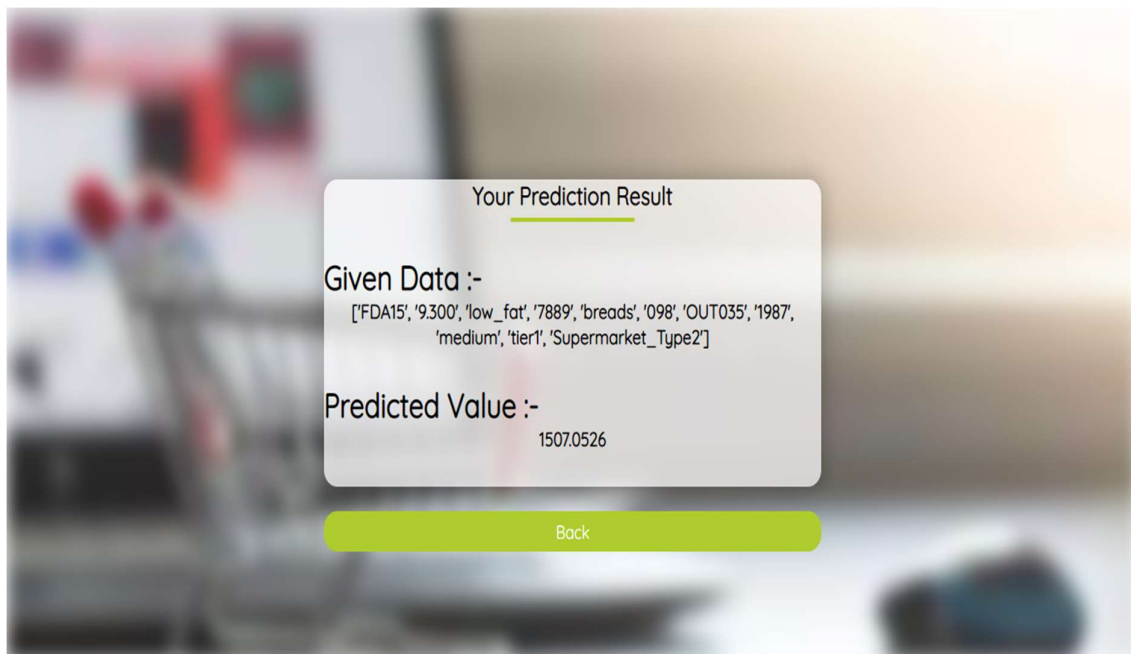


Prediction Page: -



The image shows a web application interface titled "BigMart Sales Prediction". It features a yellow background with a blurred image of a supermarket aisle. The form is divided into two columns of input fields. The left column includes: "Item Identifier" (text input), "Item Fat Content" (dropdown menu), "Item Type" (dropdown menu), "Outlet Identifier" (dropdown menu), "Outlet Size" (dropdown menu), and "Outlet Type" (dropdown menu). The right column includes: "Item Weight" (text input), "Item Visibility" (text input), "Item MRP (₹)" (text input with a rupee symbol), "Outlet Establishment Year" (text input), and "Outlet Location Type" (dropdown menu). At the bottom right, there are three buttons: "Predict", "Clear", and "Log Out".

Result Page: -



The image shows a web application interface titled "Your Prediction Result". It features a blurred background of a supermarket aisle. A white box with a green border contains the following text: "Given Data :-" followed by a list of values in single quotes: ["FDA15", "9.300", "low_fat", "7889", "breads", "098", "OUT035", "1987", "medium", "tier1", "Supermarket_Type2"]. Below this, it says "Predicted Value :-" followed by the value "1507.0526". At the bottom, there is a green button labeled "Back".

❖ CONCLUSION: -

In this project, the basics of machine learning and the associated data processing and modeling algorithms have been described, followed by their application for the task of sales prediction in Big Mart shopping centers at different locations. On implementation, the prediction results show the correlation among different attributes considered and how a particular location of medium size recorded the highest sales, suggesting that other shopping locations should follow similar patterns for improved sales.

Also, it can be concluded that more locations should be switched or shifted to Tier-3 in outlet type “Supermarket Type3” to increase the sales of products at Big Mart

By using this method we make any other store model like that to increase the sale of the store

❖ FUTURE SCOPE: -

The project can be further collaborated in any device supported with an in-built intelligence by the Internet of Things (IoT), to be more feasible for use. Various stakeholders concerned with sales information can also provide more inputs to help in hypothesis generation and more instances can be taken into consideration such that more precise results that are closer to real-world situations are generated. When combined with effective data mining methods and properties, the traditional means could be seen to make a higher and positive effect on the overall development of a simpler algorithms corporation’s tasks on the whole. One of the main highlights is more expressive regression outputs, which are more understandable bounded with some accuracy. Moreover, the flexibility of the proposed approach can be increased with variants at a very appropriate stage of regression model building. There is a

❖ REFERENCE: -

- <https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data>
- <https://ineuron.ai/>
- https://devdocs.io/scikit_learn/
- <https://xgboost.readthedocs.io/en/stable/>
- <https://www.adamsmith.haus/python/docs/pandas>
- <https://www.adamsmith.haus/python/docs/seaborn>