BIG MART SALES PREDICTION AND ANALYSIS USING MACHINE LEARNING

B.Tech Final Year End Semester PPT

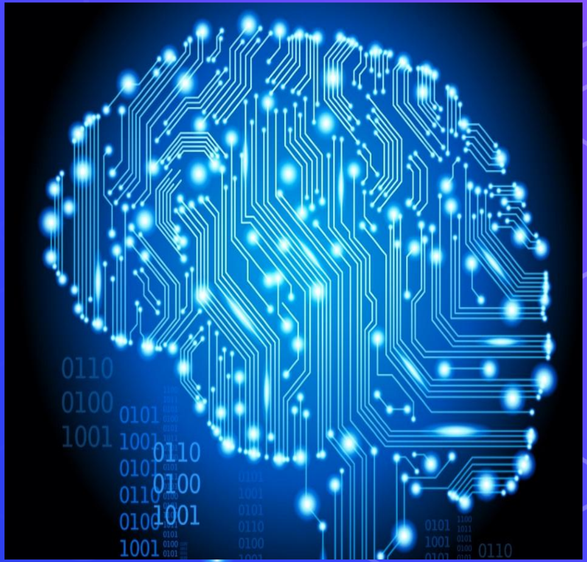
Dept of Computer Science & Engineering
IIIT Bhubaneswar

By:-
Chiranjibi Tajan(B516015)
Siona Panda(B518048)

Under the guidance of
Prof. Tushar Ranjan Sahoo

OUTLINE

- BACKGROUND
- INTRODUCTION
- MOTIVATION AND OBJECTIVES
- LITERATURE REVIEW
- PROBLEM STATEMENT
- METHODOLOGY
- PROPOSED ARCHITECTURE
- DESIGN AND DEVELOPMENT OF SOLUTION
- ALGORITHMS USED
- IMPLEMENTATION
- OBSERVATIONS
- RESULT ANALYSIS AND EVALUATION
- CONCLUSION
- SCOPE FOR FUTURE WORK
- BIBLIOGRAPHY/REFERENCES



BACKGROUND

- Sales forecasting is vital in marketing, retailing, wholesaling, and manufacturing.
- Many Big Marts do not have a superior yearly sales forecasting method nowadays.
- This is due to a lack of sales estimation skills, resources, and understanding.
- As a result, there is a pressing need in the contemporary marketplace for the development of a smart prediction system that is quick, adaptable, and accurate.
- This enables Big Marts to properly allocate cash, estimate realistic sales revenues, and build a better plan for potentially expanding the business, as well as better tactics that will lead to future growth.
- Shopping marts and superstores can keep track of individual items' sales data to predict future demand and adjust stock management.
- If items are not readily available or if supply exceeds demand, total profit may be jeopardized. As a result, product sales forecasting could be essential in reducing losses.

INTRODUCTION

- Good sales are the life of every organization,so the forecasting of sales plays an important role in any shopping complex.
- In today's modern world, large shopping centers such as malls and marts keep track of sales of items or products, as well as their many dependent and independent aspects, as a key step in forecasting future demand and inventory management.
- A standard sales prediction study can help in deeply analyzing the situations or the conditions previously occurred and then, the inference can be applied about customer acquisition, funds inadequacy and strengths before setting a budget and marketing plans for the upcoming year.
- Extensive research is going on in retailers domain for forecasting the future sales demand.
- The basic and foremost technique used in predicting sales is the traditional method, but these methods take much more time for predicting sales and to overcome these problems in traditional methods machine learning techniques are deployed.

MOTIVATION

- Let us consider the following scenario: we wish to open an inventory for selling everyday products such as ration items, and we have a central customer specialized location for service.
- Once all of the information about the types and quantities of foodstuffs, ration items, and everyday products used by the residents of a given area has been gathered,the information can then be given to specific shop owners or big-mart store salespeople who can make the products or services available.
- So that they may maintain their inventory in accordance with the data, minimizing the excess products in their inventory that remain there until the distributor returns them.
- As a result, new shop owners and existing shop owners who wish to maintain their inventory fresh with new products ready to sell must rely on the dependability of our processed data.
- The thought of establishing superior Business Strategies sparked motivation.

OBJECTIVES

- Provide information to anyone looking for average consumption data for a specific region.
- To determine crucial aspects that can boost sales and what modifications to the product or store's attributes could be made.
- The main goal of this project is to find out which machine learning model performs best when estimating sales for a specific collection of products and retailers.
- As a result, the purpose of this project is to improve forecasting models in order to reduce waste and boost product availability.
- The goal is to predict the pattern of sales and the quantities of products to be sold based on some key characteristics gleaned from the raw data.
- This will assist BigMart in increasing sales by learning how to better organise products within stores.
- To find out key factors that can increase their sales and what changes could be made to the product or store's characteristics.

LITERATURE REVIEW

- The two most fundamental notions of sellers and consumers are supply and demand.
- For firms to be able to develop future sales plans, it is critical to accurately predict demand.
- Sales Prediction is the process of anticipating sales for various Big Mart shops in order to adjust the company's future strategies depending on the expected sales.
- They suggest a method for Big Mart companies to forecast demand.
- Every business wants to be in the competition in today's market. An excellent concept for a corporation to examine product sales is to use sales forecast.

PROBLEM STATEMENT

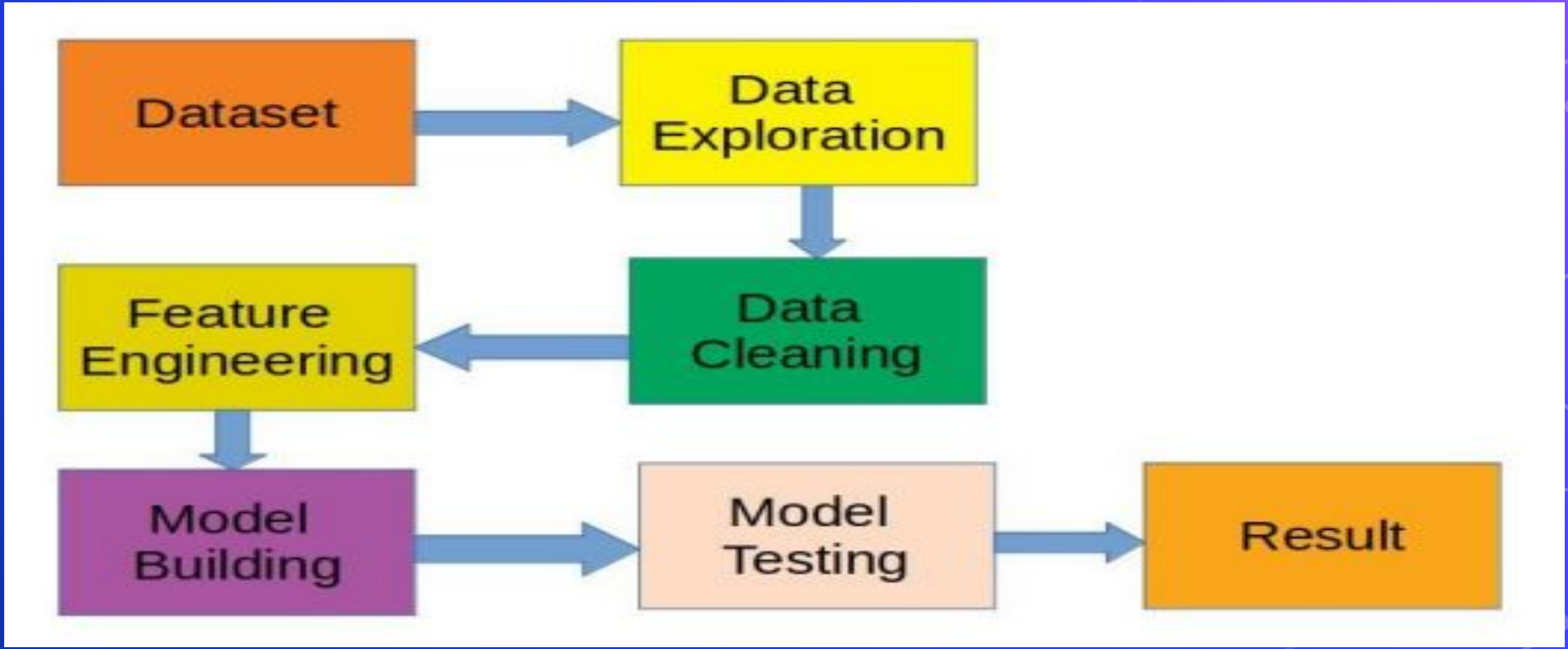
- By studying Big Mart sales, we can figure out what role certain qualities of an item play and how they affect its sales.
- To assist BigMart in achieving this goal, a predictive model can be constructed to determine the important elements that can enhance sales in each store and what adjustments to the product or store's attributes could be made.
- The dataset was created from 2013 sales data for 1559 products across 10 retailers in various cities.
- The goal is to create a prediction model and determine the sales of each product at a specific store.
- Using This Model, Big Mart Will Try To Understand The Properties Of Products And Stores Which Play A Key Role In Increasing Sales.

METHODOLOGY

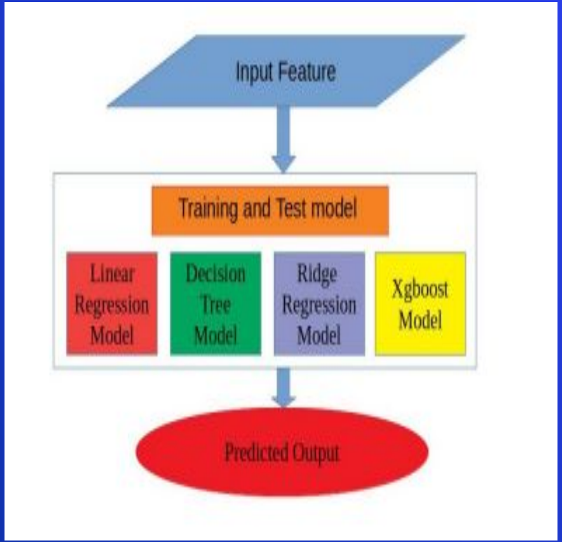
DATASET DESCRIPTION

ATTRIBUTE	DESCRIPTION
Item Identifier	Product Unique Id
Item Weight	Product Weight
Item Fat Content	Fat Content of the Product
Item Visibility	Total percentage of allocation to this store
Item Type Category	Categorization of the product
Item MRP	Product Price
Outlet Identifier	Unique ID of the outlet
Outlet Establishment Year	Store Establishment Year
Outlet Size	Store size
Outlet Location Type	Type of located cities
Outlet Type	Supermarket sort
Item Outlet Sales	Product sales in particular area

PROPOSED ARCHITECTURE



PROPOSED ARCHITECTURE



- The name of the dataset is "BigMart" Dataset and it consists of 12 attributes.
- Out of these 12 attributes, the response variable is the Item Outlet Sales and remaining are mostly used as the predictor variables.
- This data-set consists of 8523 products across the different cities.
- A dataset is formed and finally the dataset is divided into two sets, training dataset and testing dataset in the ratio 80:20.
- After pre-processing and filling the missing values, an ensemble classifier using Random Forest, Decision trees, Ridge regression, Xgboost and Linear regression is implemented.
- We propose a model using Xgboost technique.
- Both MAE and RMSE are used as accuracy metrics for predicting the sales in Big Mart.
- From the accuracy metrics it was found that the model will predict best using minimum MAE and RMSE.

DESIGN AND DEVELOPMENT OF SOLUTION

Our project involves the following broad steps:-

1. Dataset Collection
2. Data Exploration
3. Data Cleaning
4. Feature Engineering
5. Model Building
6. Model Testing
7. Result Analysis

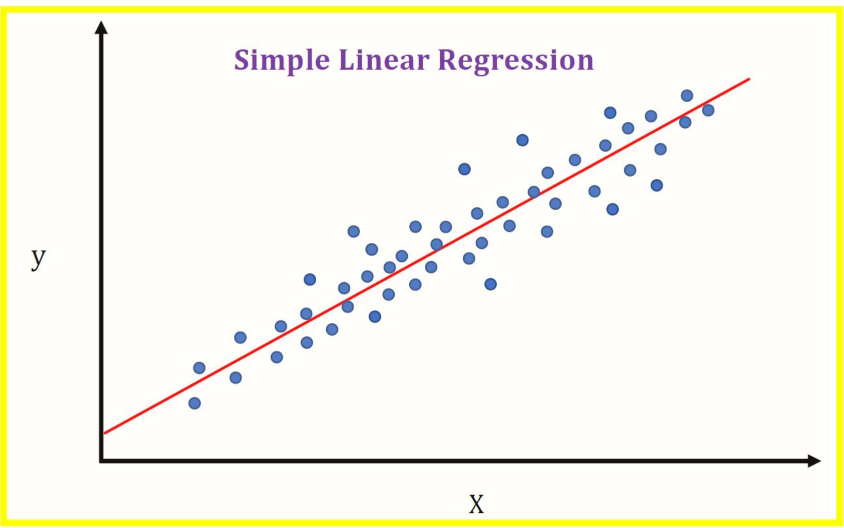
DESIGN AND DEVELOPMENT OF SOLUTION

1. **Hypothesis Generation** – understanding the problem better by brainstorming possible factors that can impact the outcome.
2. **Data Exploration** – looking at categorical and continuous feature summaries and making inferences about the data.
3. **Data Cleaning** – imputing missing values in the data and checking for outliers.
4. **Feature Engineering** – modifying existing variables and creating new ones for analysis.
5. **Model Building** – making predictive models on the data.

ALGORITHMS USED

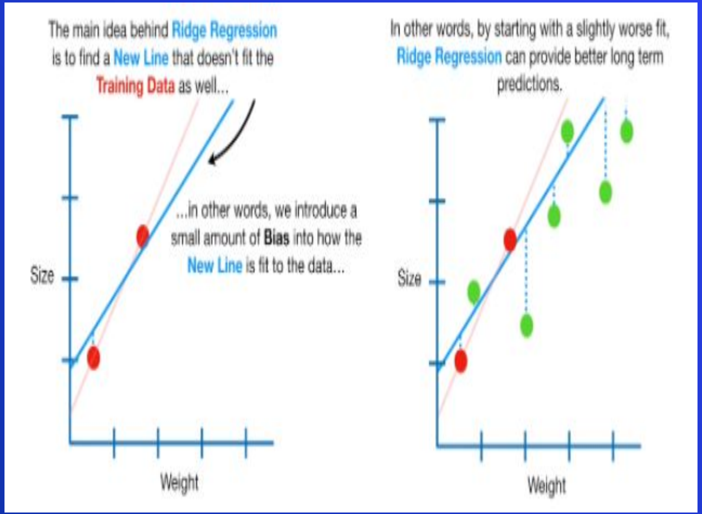
LINEAR REGRESSION

- It finds the relationship between the dependent variable (Y) and one or more independent variables (X) using one straight line which is the best fit line also termed as the regression line. The equation representing this line is:Y=a+b*X+e
- In the above equation: a is intercept, b is the slope of the line,e is the error term.
- The accuracy can be found out using this method.
- Although this model is very famous for analysis its disadvantage is that it gives less accurate results.



ALGORITHMS USED

RIDGE REGRESSION

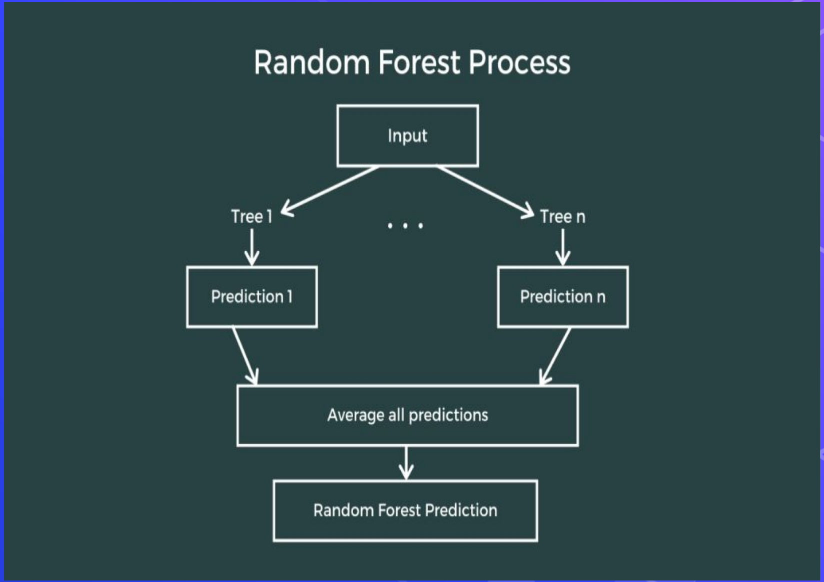


- Ridge Regression is a method used where multicollinearity (independent variables are highly correlated) affects outcomes.
- While the least square estimates are objective in multicollinearity, their variances are broad and deviate from the true value.
- By applying a degree of bias to regression calculations, ridge regression eliminates standard errors.

ALGORITHMS USED

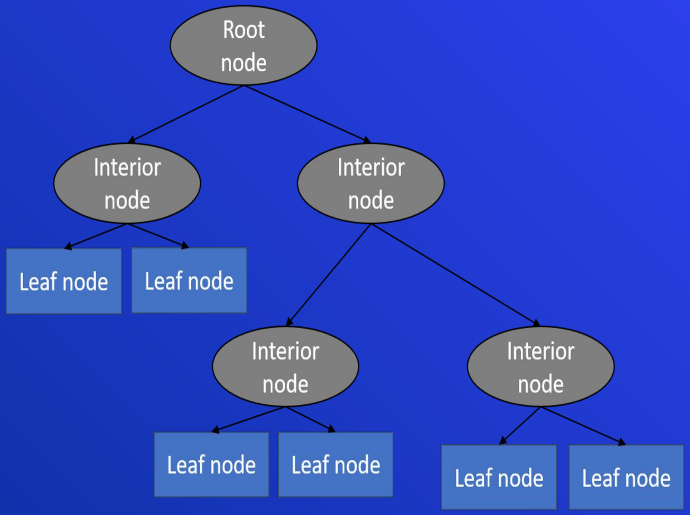
RANDOM FOREST REGRESSION

- Random forest classifiers are employed in sales prediction because they have decision tree-like hyperparameters.
- The tree model is similar to a decision making tool.
- A random forest model is created for each individual learner using a random set of rows and a few randomly selected factors.
- The final forecast may be based on all of the individual learners' guesses.
- In the case of a regression problem, the final forecast may be the average of all previous predictions.



ALGORITHMS USED

DECISION TREE REGRESSION



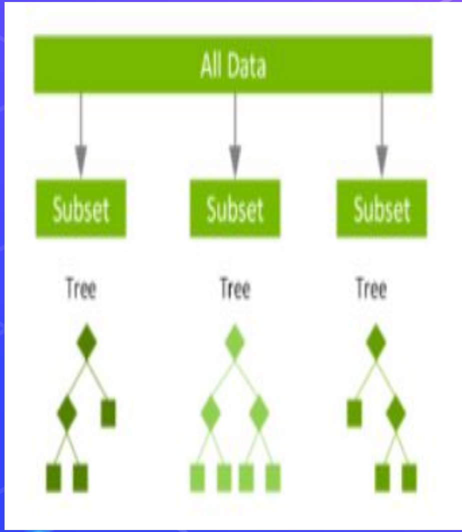
- Decision Trees are supervised machine learning algorithms.
- In which decision trees work is they smash down the entire data set represented in the root node of the tree into smaller and extra homogeneous units at the same time as on the equal time building a decision tree in an incremental manner.
- Each inner node of a tree is a donation for a check on one feature, every branch represents a final result of a check, and every leaf in the tree, or may be called as a terminal node, holds a decision that is in the regression case.

ALGORITHMS USED

XGBOOST (EXTREME GRADIENT BOOSTING) REGRESSION

- The XGBoost algorithm is developed using Decision trees and Gradient boosting.
- This algorithm stands on the principle of boosting other weaker algorithms placed in a gradient descent boosting framework.
- This approach works very accurately beating almost all other algorithms in providing accurate prediction.
- It can be defined as an extension to Gradient Boosting algorithm.
- Features of XG Boost are:

1. Parallelized tree building.
2. Efficient handling of missing data.



IMPLEMENTATION

IMPORTING REQUIRED LIBRARIES AND UPLOADING DATA FILE

Importing required libraries

```
[1] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
from google.colab import files
uploaded = files.upload()
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=len(uploaded[fn])))
```

Choose Files Big_mart.csv

- **Big_mart.csv**(text/csv) - 869537 bytes, last modified: 5/18/2022 - 100% done

```
Saving Big_mart.csv to Big_mart (7).csv
User uploaded file "Big_mart.csv" with length 869537 bytes
```

IMPLEMENTATION

IMPORTING DATASET AND CHECKING ITS PROPERTIES

Importing Dataset

```
[3] df=pd.read_csv('Big_mart.csv')
```

```
[4] df.shape
```

```
(8523, 12)
```

```
[5] df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locatior
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

IMPLEMENTATION

IMPORTING DATASET AND CHECKING ITS PROPERTIES

```
[5] df.head()
```

_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

```
df.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

IMPLEMENTATION

IMPORTING DATASET AND CHECKING ITS PROPERTIES

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
```

```
[8] df.columns
```

```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type', 'Item_Outlet_Sales'],
      dtype='object')
```

IMPLEMENTATION

DATA CLEANING AND REMOVING NULL VALUES

Cleaning Data

```
[9]  df.isnull().sum()
```

```
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
```

Removing NULL values from all the columns

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

```
[11]  df['Outlet_Size']=df['Outlet_Size'].fillna('Medium')
```

IMPLEMENTATION

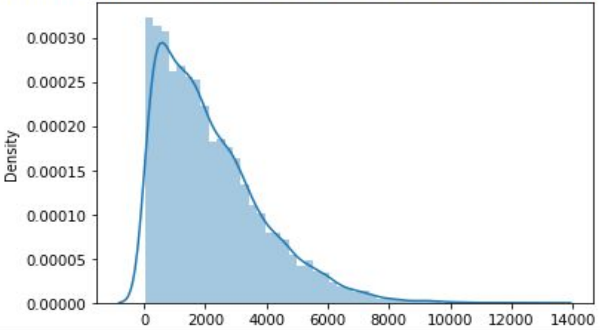
EXPLORATORY DATA ANALYSIS (UNIVARIATE ANALYSIS)

Exploratory Data Analysis

Univariate Analysis

```
sns.distplot(df['Item_Outlet_Sales'])
```

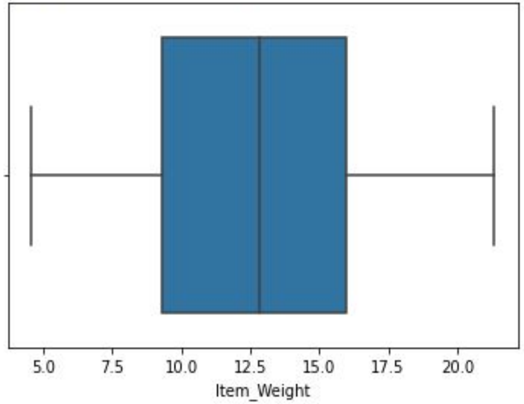
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:261
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee5568510>
```



We can see that the Outlet_sales is normally distributed with right skewness.

```
sns.boxplot(df['Item_Weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee54a1590>
```



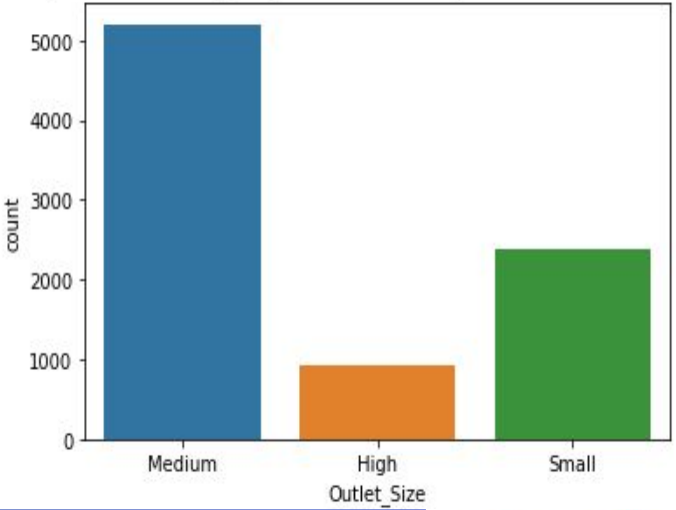
IMPLEMENTATION

EXPLORATORY DATA ANALYSIS (UNIVARIATE ANALYSIS)

Most of the items have weight in range of 8-16. The mode of weight is near to 13.

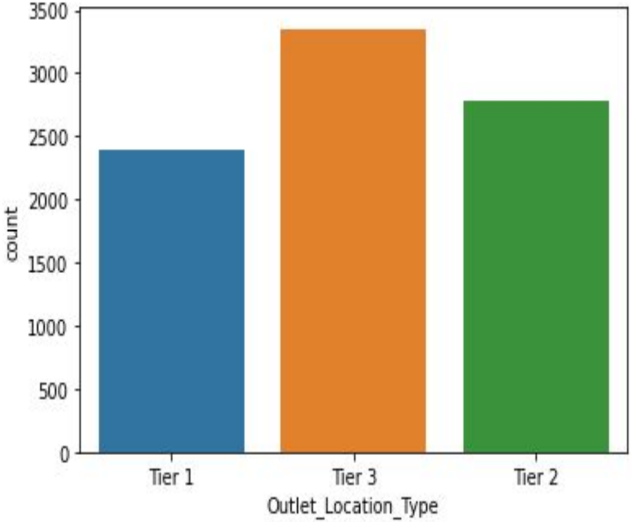
```
sns.countplot(x='Outlet_Size',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee2f95dd0>
```




```
sns.countplot(df['Outlet_Location_Type'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee2f01c90>
```



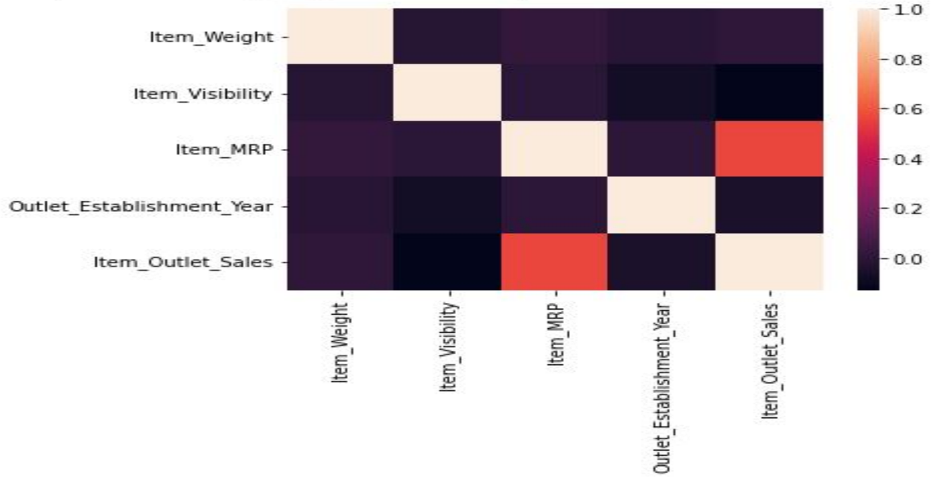
1. Most of the Outlet are located in Tier 3.
2. Tier 2 location is second most after Tier 3.

IMPLEMENTATION

CORRELATION AND HEATMAP(BIVARIATE ANALYSIS)

```
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee034fbd0>
```




```
df.corr()
```

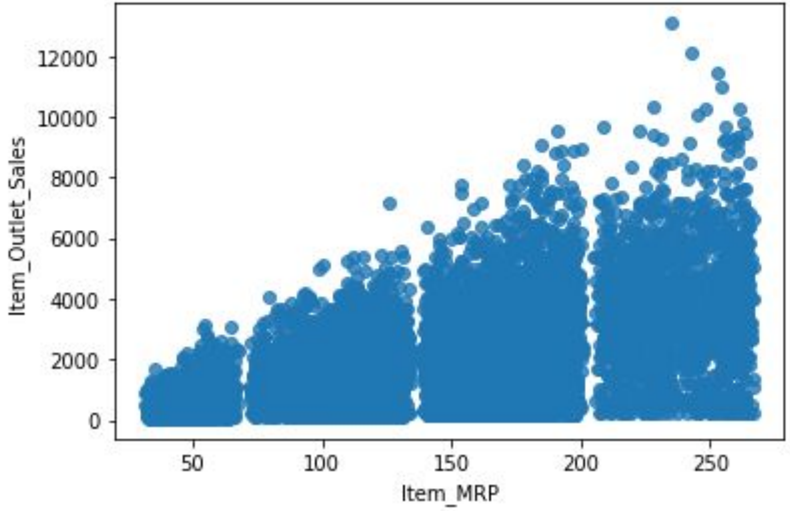
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
Item_Weight	1.000000	-0.012049	0.024756	-0.008301	0.011550
Item_Visibility	-0.012049	1.000000	-0.001315	-0.074834	-0.128625
Item_MRP	0.024756	-0.001315	1.000000	0.005020	0.567574
Outlet_Establishment_Year	-0.008301	-0.074834	0.005020	1.000000	-0.049135
Item_Outlet_Sales	0.011550	-0.128625	0.567574	-0.049135	1.000000

IMPLEMENTATION

SCATTER PLOT(BIVARIATE ANALYSIS)

```
sns.regplot(x='Item_MRP',y='Item_Outlet_Sales',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ee0295790>
```



1. The plot shows that more the price of item, the better is its outlet sale.
2. The relation is almost linear.

IMPLEMENTATION

FEATURE ENGINEERING

Feature Engineeering

Converting categorical data into numerical

```
[26] df['Item_Fat_Content'].unique()

     array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

```
[27] def fun(x):
       if x=='Low Fat' or x=='LF' or x=='low fat':
         return(0)
       else:
         return(1)
```

```
df['Item_Fat_Content']=df['Item_Fat_Content'].apply(fun)
```

IMPLEMENTATION

CONVERTING CATEGORICAL DATA INTO NUMERICAL DATA

```
[29] df['Item_Fat_Content'].head()
```

```
0    0
1    1
2    0
3    1
4    0
Name: Item_Fat_Content, dtype: int64
```

```
[30] df['Item_Type'].unique()
```

```
array(['Dairy', 'Soft Drinks', 'Meat', 'Fruits and Vegetables',
       'Household', 'Baking Goods', 'Snack Foods', 'Frozen Foods',
       'Breakfast', 'Health and Hygiene', 'Hard Drinks', 'Canned',
       'Breads', 'Starchy Foods', 'Others', 'Seafood'], dtype=object)
```

```
df['Outlet_Size'].unique()
```

```
array(['Medium', 'High', 'Small'], dtype=object)
```

```
[32] def fun1(x):
       if x=='Medium':
         return(0)
       elif x=='High':
         return(1)
       else:
         return(2)
```

```
[33] df['Outlet_Size']=df['Outlet_Size'].apply(fun1)
```

```
df['Outlet_Size'].head()
```

```
0    0
1    0
2    0
3    0
4    1
Name: Outlet_Size, dtype: int64
```

IMPLEMENTATION

CONVERTING CATEGORICAL DATA INTO NUMERICAL DATA

```
[35] df['Outlet_Location_Type'].unique()

     array(['Tier 1', 'Tier 3', 'Tier 2'], dtype=object)

[36] def fun2(x):
       if x=='Tier 1':
         return(0)
       elif x=='Tier 2':
         return(1)
       else:
         return(2)

[37] df['Outlet_Location_Type']=df['Outlet_Location_Type'].apply(fun2)

[38] df['Outlet_Location_Type'].head()

     0    0
     1    2
     2    0
     3    2
     4    2
     Name: Outlet_Location_Type, dtype: int64

[39] df['Outlet_Type'].unique()

     array(['Supermarket Type1', 'Supermarket Type2', 'Grocery Store',
            'Supermarket Type3'], dtype=object)
```

IMPLEMENTATION

CONVERTING CATEGORICAL DATA INTO NUMERICAL DATA

```
[40] def fun3(x):
       if x=='Supermarket Type1':
         return(0)
       elif x=='Supermarket Type2':
         return(1)
       elif x=='Supermarket Type3':
         return(2)
       else:
         return(3)
```

```
[41] df['Outlet_Type']=df['Outlet_Type'].apply(fun3)
```

```
df['Outlet_Type'].head()
```

```
0    0
1    1
2    0
3    3
4    0
Name: Outlet_Type, dtype: int64
```

```
[43] df['Outlet_Identifier'].unique()
```

```
array(['OUT049', 'OUT018', 'OUT010', 'OUT013', 'OUT027', 'OUT045',
       'OUT017', 'OUT046', 'OUT035', 'OUT019'], dtype=object)
```

```
[44] df1=pd.get_dummies(df['Outlet_Identifier'])
```

```
[45] df=pd.concat([df,df1],axis=1)
```

```
[46] df=df.drop(['Outlet_Identifier'],axis=1)
```

```
df.columns
```

```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Establishment_Year', 'Outlet_Size',
       'Outlet_Location_Type', 'Outlet_Type', 'Item_Outlet_Sales', 'OUT010',
       'OUT013', 'OUT017', 'OUT018', 'OUT019', 'OUT027', 'OUT035', 'OUT045',
       'OUT046', 'OUT049'],
      dtype='object')
```

IMPLEMENTATION

VIEWING THE CATEGORICAL COLUMNS AFTER CONVERTING THEM INTO NUMERICAL COLUMNS

```
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
0	FDA15	9.30	0	0.016047	Dairy	249.8092	1999	0	0
1	DRC01	5.92	1	0.019278	Soft Drinks	48.2692	2009	0	2
2	FDN15	17.50	0	0.016760	Meat	141.6180	1999	0	0
3	FDX07	19.20	1	0.000000	Fruits and Vegetables	182.0950	1998	0	2
4	NCD19	8.93	0	0.000000	Household	53.8614	1987	1	2

5 rows × 21 columns

```
df.head()
```

_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	...	OUT010	OUT013	OUT017	OUT018	OUT019	OUT027	OUT035	OUT045	OUT046	OUT049
8092	1999	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2692	2009	0	2	1	...	0	0	0	1	0	0	0	0	0	0
6180	1999	0	0	0	...	0	0	0	0	0	0	0	0	0	1
0950	1998	0	2	3	...	1	0	0	0	0	0	0	0	0	0
8614	1987	1	2	0	...	0	1	0	0	0	0	0	0	0	0

IMPLEMENTATION

FEATURE SELECTION

Feature selection

From the co-relation map we plotted before, we can see that there almost no co-relation between 'Item_Identifier', 'Item_type' and 'Outlet_Establishment_year' with our target i.e 'Item_Outlet_sales'

So we will drop these columns and will not use in training out model.

Assigning data to X and Y variable

```
[49] x=df.drop(['Item_Identifier','Item_Type','Outlet_Establishment_Year','Item_Outlet_Sales'],axis=1)
     y=df['Item_Outlet_Sales']
```

```
[50] x.head()
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	OUT010	OUT013	OUT017	OUT018	OUT019	OUT027	OUT035
0	9.30	0	0.016047	249.8092	0	0	0	0	0	0	0	0	0	0
1	5.92	1	0.019278	48.2692	0	2	1	0	0	0	1	0	0	0
2	17.50	0	0.016760	141.6180	0	0	0	0	0	0	0	0	0	0
3	19.20	1	0.000000	182.0950	0	2	3	1	0	0	0	0	0	0
4	8.93	0	0.000000	53.8614	1	2	0	0	1	0	0	0	0	0

IMPLEMENTATION

TRAINING,TESTING & SPLITTING

```
[50] x.head()
```

t_Content	Item_Visibility	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	OUT010	OUT013	OUT017	OUT018	OUT019	OUT027	OUT035	OUT045	OUT046	OUT049
0	0.016047	249.8092	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0.019278	48.2692	0	2	1	0	0	0	1	0	0	0	0	0	0
0	0.016760	141.6180	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0.000000	182.0950	0	2	3	1	0	0	0	0	0	0	0	0	0
0	0.000000	53.8614	1	2	0	0	1	0	0	0	0	0	0	0	0

```
[51] y.head()

0    3735.1380
1     443.4228
2    2097.2700
3     732.3800
4     994.7052
Name: Item_Outlet_Sales, dtype: float64
```

Training Testing and splitting

```
[52] from sklearn.model_selection import train_test_split
     from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score,accuracy_score,classification_report,confusion_matrix

[53] x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

IMPLEMENTATION

MODEL BUILDING (LINEAR REGRESSION MODEL)

Linear Regression Model

```
[54] from sklearn.linear_model import LinearRegression
```

```
[55] lrm=LinearRegression()
```

```
[56] lrm.fit(x_train,y_train)

     LinearRegression()
```

```
[57] lrm_predict=lrm.predict(x_test)
```

Evaluation of Linear Regression model

```
[58] print("MEAN SQUARED ERROR(MSE)",mean_squared_error(y_test,lrm_predict))
     print("MEAN ABSOLUTE ERROR(MAE)",mean_absolute_error(y_test,lrm_predict))
     print("ROOT MEAN SQUARED ERROR(RMSE)",np.sqrt(mean_squared_error(y_test,lrm_predict)))
     lrm_score=r2_score(y_test,lrm_predict)
     print("R2 SCORE",r2_score(y_test,lrm_predict))

     MEAN SQUARED ERROR(MSE) 1283022.2303069932
     MEAN ABSOLUTE ERROR(MAE) 839.1924227990719
     ROOT MEAN SQUARED ERROR(RMSE) 1132.705712136649
     R2 SCORE 0.5724696849282869
```

IMPLEMENTATION

MODEL BUILDING (RIDGE REGRESSION MODEL)

Ridge Regression Model

```
[59] from sklearn.linear_model import Ridge
```

```
[60] rid=Ridge(alpha=0.009)
```

```
[61] rid.fit(x_train,y_train)

     Ridge(alpha=0.009)
```

```
[62] rid_predict=rid.predict(x_test)
```

Evaluation of Ridge Regression Model

```
[63] print("MEAN SQUARED ERROR(MSE)",mean_squared_error(y_test,rid_predict))
     print("MEAN ABSOLUTE ERROR(MAE)",mean_absolute_error(y_test,rid_predict))
     print("ROOT MEAN SQUARED ERROR(RMSE)",np.sqrt(mean_squared_error(y_test,rid_predict)))
     rid_score=r2_score(y_test,rid_predict)
     print("R2 SCORE",r2_score(y_test,rid_predict))

     MEAN SQUARED ERROR(MSE) 1283022.244564803
     MEAN ABSOLUTE ERROR(MAE) 839.191913653632
     ROOT MEAN SQUARED ERROR(RMSE) 1132.7057184303446
     R2 SCORE 0.5724696801772812
```

IMPLEMENTATION

MODEL BUILDING (RANDOM FOREST REGRESSION MODEL)

Random Forest Regressor

```
[64]  from sklearn.ensemble import RandomForestRegressor
```

```
[65]  rfg=RandomForestRegressor()
```

```
[66]  rfg.fit(x_train,y_train)

      RandomForestRegressor()
```

```
[67]  rfg_predict=rfg.predict(x_test)
```

Evaluation of Random Forest Regression model

```
[68]  print("MEAN SQUARED ERROR(MSE)",mean_squared_error(y_test,rfg_predict))
      print("MEAN ABSOLUTE ERROR(MAE)",mean_absolute_error(y_test,rfg_predict))
      print("ROOT MEAN SQUARED ERROR(RMSE)",np.sqrt(mean_squared_error(y_test,rfg_predict)))
      print("R2 SCORE",r2_score(y_test,rfg_predict))

      MEAN SQUARED ERROR(MSE) 1335343.740588017
      MEAN ABSOLUTE ERROR(MAE) 804.9766139683284
      ROOT MEAN SQUARED ERROR(RMSE) 1155.5707423554895
      R2 SCORE 0.5550350440880252
```

IMPLEMENTATION

HYPERPARAMETER TUNING IN RANDOM FOREST REGRESSION MODEL FOR OBTAINING BETTER RESULTS

Use GridSearchCV to find the best parameter for RandomforestRegressor

```
[69] from sklearn.model_selection import GridSearchCV
```

```
[70] grid_para={'n_estimators':[50,100,150,200,500],'max_depth':[2,4,5,6,7,8]}
```

```
[71] gsc=GridSearchCV(RandomForestRegressor(),grid_para)
     gsc.fit(x_train,y_train)

     GridSearchCV(estimator=RandomForestRegressor(),
                  param_grid={'max_depth': [2, 4, 5, 6, 7, 8],
                              'n_estimators': [50, 100, 150, 200, 500]})
```

Best Parameter

```
[72] gsc.best_params_

     {'max_depth': 6, 'n_estimators': 200}
```

```
[73] grid_rbgr=RandomForestRegressor(max_depth= 5, n_estimators= 150)
```

```
[74] grid_rbgr.fit(x_train,y_train)

     RandomForestRegressor(max_depth=5, n_estimators=150)
```

IMPLEMENTATION

MODEL BUILDING (DECISION TREE REGRESSION MODEL)

Accuracy by best parameter

```
[75] grid_rbgr_score=grid_rbgr.score(x_test,y_test)
     print(grid_rbgr_score)

     0.60227323591256
```

Decision Tree Regressor

```
[76] from sklearn.tree import DecisionTreeRegressor
```

```
[77] dt = DecisionTreeRegressor()
```

```
[78] dt.fit(x_train,y_train)

     DecisionTreeRegressor()
```

```
[79] dt_predict=dt.predict(x_test)
```

IMPLEMENTATION

HYPERPARAMETER TUNING IN DECISION TREE REGRESSION MODEL FOR OBTAINING BETTER RESULTS

Evaluation of Decision Tree Regression Model

```
[80] print("MEAN SQUARED ERROR(MSE)",mean_squared_error(y_test,dt_predict))
     print("MEAN ABSOLUTE ERROR(MAE)",mean_absolute_error(y_test,dt_predict))
     print("ROOT MEAN SQUARED ERROR(RMSE)",np.sqrt(mean_squared_error(y_test,dt_predict)))
     print("R2 SCORE",r2_score(y_test,dt_predict))
```

```
MEAN SQUARED ERROR(MSE) 2392124.9911261336
MEAN ABSOLUTE ERROR(MAE) 1081.8039454545453
ROOT MEAN SQUARED ERROR(RMSE) 1546.6496019222109
R2 SCORE 0.20289303880388088
```

Use GridSearchCV to find the best parameter for DecisionTreeRegressor

```
[81] from sklearn.model_selection import GridSearchCV
```

```
[82] grid_para={'min_samples_leaf':[50,100,150,200,500],'max_depth':[2,18,10,15,20]}
     gsc=GridSearchCV(DecisionTreeRegressor(),grid_para)
     gsc.fit(x_train,y_train)
```

```
GridSearchCV(estimator=DecisionTreeRegressor(),
             param_grid={'max_depth': [2, 18, 10, 15, 20],
                         'min_samples_leaf': [50, 100, 150, 200, 500]})
```

IMPLEMENTATION

HYPERPARAMETER TUNING IN DECISION TREE REGRESSION MODEL FOR OBTAINING BETTER RESULTS

001

Best Parameter

```
[83] gsc.best_params_

{'max_depth': 18, 'min_samples_leaf': 100}
```

```
[84] dec_grid=DecisionTreeRegressor( max_depth=18, min_samples_leaf=50)
     dec_grid.fit(x_train,y_train)

DecisionTreeRegressor(max_depth=18, min_samples_leaf=50)
```

Accuracy by best parameter

```
[85] grid_dec_score=dec_grid.score(x_test,y_test)
     print(grid_dec_score)

0.5902335202106415
```

IMPLEMENTATION

MODEL BUILDING (XGBOOST REGRESSION MODEL)

XG Boost Regressor

```
[86] from xgboost import XGBRegressor
     xgb = XGBRegressor(n_estimators=1000, learning_rate=0.05)
     xgb.fit(x_train,y_train, early_stopping_rounds=5,
                  eval_set=[(x_test,y_test)], verbose=False)
```

```
[08:00:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor(learning_rate=0.05, n_estimators=1000)
```

```
[87] xgb_predict = xgb.predict(x_test)
```

Evaluation of XG Boost Regression Model

```
[88] print("MEAN SQUARED ERROR(MSE)",mean_squared_error(y_test,xgb_predict))
     print("MEAN ABSOLUTE ERROR(MAE)",mean_absolute_error(y_test,xgb_predict))
     print("ROOT MEAN SQUARED ERROR(RMSE)",np.sqrt(mean_squared_error(y_test,xgb_predict)))
     xgb_score=r2_score(y_test,xgb_predict)
     print("R2 SCORE",r2_score(y_test,xgb_predict))
```

```
MEAN SQUARED ERROR(MSE) 1169207.4056684782
MEAN ABSOLUTE ERROR(MAE) 759.989053429918
ROOT MEAN SQUARED ERROR(RMSE) 1081.298943710054
R2 SCORE 0.6103952069403983
```

OBSERVATIONS

- The implementation, shows 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.
- In the below graph, we can observe that the feature with the lowest correlation with our target variable is the Item Visibility.
- So, the less available the commodity is the higher the price would be in the shop.
- The most positive finding is from Item MRP.

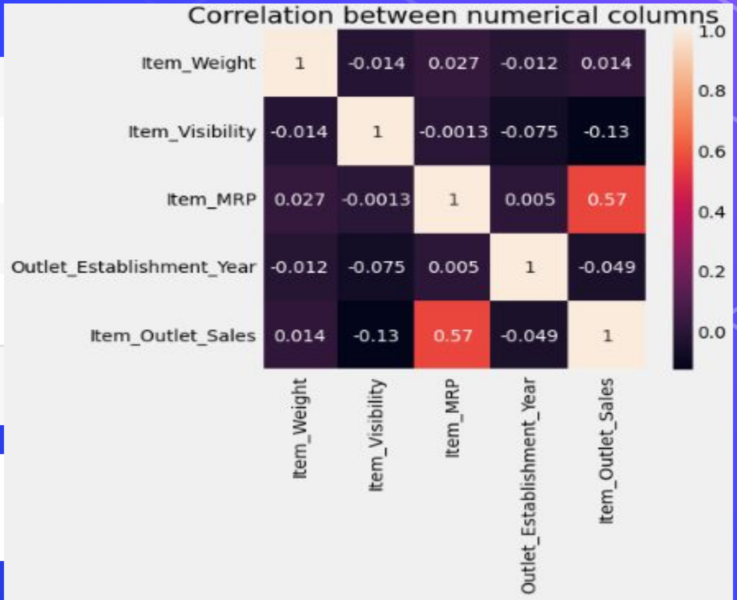
```
df['Item_Outlet_Sales'].max()
```

13086.9648

```
df.loc[df['Item_Outlet_Sales']==13086.9648]
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
7188	NCE42	12.857645	Low Fat	0.010551	Household	234.9958	OUT027	1985	Medium

_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Low Fat	0.010551	Household	234.9958	OUT027	1985	Medium	Tier 3	Supermarket Type3	13086.9648



OBSERVATIONS

- The largest location did not provide the most revenue.
- The location that generated the most sales was Supermarket Type3, which had a medium size in our dataset.
- This outlet performed significantly better than any other outlet location in the sample, regardless of size.
- To improve the sales of products at Big Mart, more sites should be switched or changed to Supermarket Type3.

RESULT ANALYSIS AND EVALUATION

COMPARISON OF ACCURACY OF DIFFERENT MODELS

Comparison of Different Models

Comparing Accuracy of Different Models

```
[89] model=['linear Regressor','Ridge Regressor','RandomForestRegressor','DecisionTreeRegressor','XGBoost Regressor']
     score=[lrm_score,rid_score,grid_rbgr_score,grid_dec_score,xgb_score]
     predict=[lrm_predict,rid_predict,rfg_predict,dt_predict,xgb_predict]
     compare=pd.DataFrame({'Model':model,'Score':score},index=[i for i in range(1,6)])
```

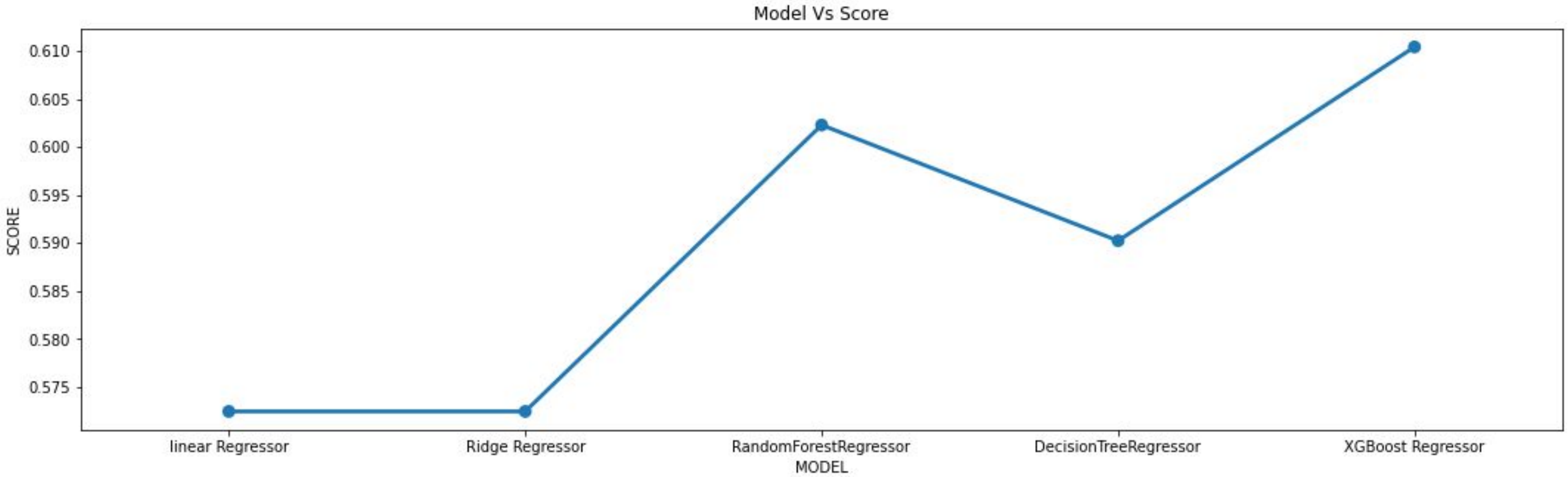
```
compare.T
```

	1	2	3	4	5
Model	linear Regressor	Ridge Regressor	RandomForestRegressor	DecisionTreeRegressor	XGBoost Regressor
Score	0.57247	0.57247	0.602273	0.590234	0.610395

RESULT ANALYSIS AND EVALUATION

POINT PLOT SHOWING MODEL Vs ACCURACY SCORE

```
plt.figure(figsize=(18,5))
sns.pointplot(x='Model',y='Score',data=compare)
plt.title('Model Vs Score')
plt.xlabel('MODEL')
plt.ylabel('SCORE')
plt.show()
```



RESULT ANALYSIS AND EVALUATION

COMPARISON OF ACCURACY SCORES OF DIFFERENT MODELS TO FIND THE BEST ONE AMONGST THEM

Comparing all Models and finding the best one.

1. Linear Regression score: 0.57247
2. Ridge Regression Score: 0.57247
3. Random Forest Regression Score: 0.602773
4. Decision Tree Regression Score: 0.590234
5. XGBoost Regression Score: 0.610395

We can see that XGBoost Regression model gave us the best score for our testing data. Therefore XGBoost Regression is best from all the above models.

RESULT ANALYSIS AND EVALUATION

MSE,MAE AND RMSE VALUES OF DIFFERENT MODELS

Linear Regressor gave an accuracy score of 0.57247 with following Error values:

- MEAN SQUARED ERROR(MSE): 1283022.2303069932
- MEAN ABSOLUTE ERROR(MAE): 839.1924227990719
- ROOT MEAN SQUARED ERROR(RMSE): 1132.705712136649

Ridge Regressor gave an accuracy score of 0.57247 with following Error values:

- MEAN SQUARED ERROR(MSE): 1283022.244564803
- MEAN ABSOLUTE ERROR(MAE): 839.191913653632
- ROOT MEAN SQUARED ERROR(RMSE): 1132.7057184303446

Random Forest Regressor gave an accuracy score of 0.602273 with following Error values:

- MEAN SQUARED ERROR(MSE): 1335343.740588017
- MEAN ABSOLUTE ERROR(MAE): 804.9766139683284
- ROOT MEAN SQUARED ERROR(RMSE): 1155.5707423554895

RESULT ANALYSIS AND EVALUATION

MSE,MAE AND RMSE VALUES OF DIFFERENT MODELS

Decision Tree Regressor gave an accuracy of 0.590234 with following Error values:

- MEAN SQUARED ERROR(MSE): 2392124.9911261336
- MEAN ABSOLUTE ERROR(MAE): 1081.8039454545453
- ROOT MEAN SQUARED ERROR(RMSE): 1546.6496019222109

XGBoost Regressor gave an accuracy of 0.610395 with following Error values:

- MEAN SQUARED ERROR(MSE) 1169207.4056684782
- MEAN ABSOLUTE ERROR(MAE) 759.989053429918
- ROOT MEAN SQUARED ERROR(RMSE) 1081.298943710054

From all the models that we used, we found out that XGBoost Regression model gave us the best score i.e 0.610395. We therefore end our analysis here and conclude XGBoost regression as our predictive model.

RESULT ANALYSIS AND EVALUATION

- So, using sales data, the methods Linear Regression, Ridge Regression, Decision Tree, Random Forest, and XGBoost are checked.
- RMSE (Root mean square error or root mean square deviation) which is one of the most often used methods for evaluating the quality of predictions is lower with XGBoost.
- It uses Euclidean distance to demonstrate how far predictions differ from measured true values.
- The residual (difference between prediction and truth) for each data point, the norm of the residual for each data point, the mean of residuals, and the square root of that mean is calculated to get the RMSE.
- When compared to other techniques, RMSE is often utilized in supervised learning applications since it employs and requires real measurements at each projected data point.
- R-squared values are higher for the XGBoost model than average. Therefore, the used model fits better and exhibits accuracy.
- It has been observed that increased efficiency is observed with XGBoost algorithms with lower RMSE rating.

RESULT ANALYSIS AND EVALUATION

- Multiple instance parameters and various other factors can also be employed to more creatively and successfully anticipate sales.
- When the parameters employed are enhanced, the accuracy of prediction systems can be greatly improved.
- Because profit is directly linked to the accuracy of sales estimates, the Big Marts strive for accuracy so that the company does not lose money.

CONCLUSION

- Every shopping mall in today's digitally linked world wants to anticipate client requests in order to avoid seasonal sales item shortages.
- Companies and shopping malls are getting better at anticipating product sales and consumer requests on a daily basis.
- The goal of this study is to use machine learning techniques to forecast future sales of Big Mart based on previous years' data.
- With conventional methods failing to assist businesses in increasing the revenue, the application of Machine Learning methodologies proves to be a significant factor in creating company plans, that take consumer purchasing habits into account.
- It is concluded that the Xgboost technique works better than existing models with lower MAE and RMSE.
- Predicting sales based on a variety of criteria, including past year's sales, allows firms to develop effective sales plans and enter the competitive market unafraid.

CONCLUSION

- Our predictions will help big marts to refine their methodologies and strategies which in turn helps them to increase their profit.
- This will also give them the idea for their new locations or Centres of Bigmart.
- Each and every company desires to know the demand of the customer in any season previously to avoid the shortage of products.
- As time passes by, the demand of the businesses needs to be more accurate about the predictions which will increase the sales exponentially.
- So, huge study is going on in this sector to make accurate predictions of sales.
- Better predictions are directly proportional to the profit made by the company.

SCOPE FOR FUTURE WORK

- The prediction's reach could be expanded in the future to include even greater scale data.
- To do so, you'll need to know the required parameters, their ideal values, and accurate sales forecasts.
- This concept can be used as a stepping stone for predicting outlet item sales in huge stores using deep learning, and it could and should be expanded upon in future studies.
- Multiple instance parameters and various other factors can also be employed to more creatively and successfully anticipate sales.
- When the parameters employed are raised, accuracy, which is important in prediction systems, can be considerably improved.
- In future, forecasting sales and building a sales plan can help to avoid unforeseen cash flow and to manage production, staff and financing needs more effectively.
- Using the transactional data, an efficient recommendation system can be built and hence the customers with similar liking will be suggested products that are available in the store.

BIBLIOGRAPHY/REFERENCES

- Nimit Jain,"Big Mart Sales Prediction Using Machine Learning",December 2021.
- Naveenraj R,Vinayaga Sundharam R, “Prediction of Big Mart Sales Using Machine Learning”,September 2021.
- Keyaben Patel,Navneet Kumar,Suraj Choudhari, “BigMart Sales Prediction Using Machine Learning”,September 2021.
- Rohit Sav,Pratiksha Shinde,Saurabh Gaikwad, “Big Mart Sales Prediction Using Machine Learning”,June 2021.
- Ranjitha P,Spandana M, “Predictive Analysis for Big Mart Sales Using Machine Learning Algorithms”,June 2021.
- Nikita Mallik,Karan Singh, “Sales Prediction Model For Big Mart”,June 2020.
- Purvika Bajaj,Renesa Ray,Shivani Shedge,Shravani Vidhate,Nikhilkumar Shardoor, “SALES PREDICTION USING MACHINE LEARNING ALGORITHMS”,June 2020.
- Akshay Godse,Poonam Pawar,Sairaj Sawant,Shirin Mujawar,Prashant Karkalle, “INTELLIGENT SALES PREDICTION USING MACHINE LEARNING TECHNIQUES”,December 2019.
- Gopal Behera,Neeta Nain, "A Comparative Study of Big Mart Sales Prediction", September 2019.

THANK YOU!