IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 6, June 2022

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.118

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

Using Machine Learning Algorithms to Forecast Huge Big Mart Sales

P Poornima, K Tharun Teja

Assistant Professor, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology

Hyderabad, India

Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology

Hyderabad, India

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. The case of Big Mart, a one-stop-shopping centre, 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 take decisions to improve sales.

KEYWORDS: Machine Learning, Random Forest, Linear Regression, Decision Tree.

I. 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 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 by means of customer, and also data related to inventory management in a data warehouse. The data is thereafter refined in order to get accurate predictions and gather new as well as interesting results that shed a new light on our knowledge with respect to the task's data. This can then further be used for forecasting future sales by means of employing machine learning algorithms such as the random forests and linear regression model.

1.1 Problem Statement

"To find out what role certain properties of an item play and how they affect their sales by understanding Big Mart sales." In order to help Big Mart achieve this goal, a predictive model can be built to find out for every store, the key factors that can increase their sales and what changes could be made to the product or store's characteristics

1.2 Existing System

The BigMart sales analysis can be done by many algorithms but presently working algorithms computational time is very high and the predicted accuracy of these algorithms is very less and these algorithms are not good in finding patterns. So, we are finding new ways to predicting the accuracy of bigmart sales data and finding patterns in the given data.

1.3 Proposed System

For building a model to predict accurate results the dataset of Big Mart sales undergoes several sequence of steps and in this work, we propose a model using Xgboost technique. Every step plays a vital role for building the proposed model. In our model we have used 2013 Big mart dataset.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

After pre-processing and filling missing values, we used ensemble classifier using Decision trees, Linear regression, Ridge regression and Random forest regression. Both MSE and CVS 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 MSE and CVS. The details of the proposed method is explained in the following section.

II. LITERATURE SURVEY

1. A Forecast for Big Mart Sales Based on Random Forests and Multiple Linear Regression used Random Forest and Linear Regression for prediction analysis which gives less accuracy. To overcome this we can use XG boost Algorithm which will give more accuracy and will be more efficient.
2. Forecasting methods and applications contains Lack of Data and short life cycles. So some of the data like historical data, consumer-oriented markets face uncertain demands, can be prediction for accurate result.
3. Comparison of Different Machine Learning Algorithms for Multiple Regression on Black Friday Sales Data Used Neural Network for comparison of different algorithms. To overcome this Complex models like neural networks is used for comparison between different algorithms which is not efficient so we can use more simpler algorithm for prediction.
4. Prediction of retail sales of footwear using feed forward and recurrent neural networks used neural networks for prediction of sales. Using neural network for predicting of weekly retail sales, which is not efficient, So XG boost can work efficiently.

III. SYSTEM DESIGN OF BIGMART SALES PREDICTION

3.1 Dataset Information

The data scientists at Big Mart have collected 2013 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, Big Mart will try to understand the properties of products and stores which play a key role in increasing sales.

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

Fig 3.1 Attributes of the dataset

The above figure 3.1 shows the attributes included in the dataset

3.2 Data Preprocessing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behavior or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues.

Three common data preprocessing steps are:

- **Formatting**
- **Cleaning**

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

- **Sampling**

3.3. Label Encoding

In machine learning, we usually deal with datasets that contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labelled in words.

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

3.4. Onehot Encoding

A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

3.5. Machine Learning Algorithms Used:

The data available is increasing day by day and such a huge amount of unprocessed data is needed to be analyzed precisely, as it can give very informative and finely pure gradient results as per current standard requirements. It is not wrong to say as with the evolution of Artificial Intelligence (AI) over the past two decades, Machine Learning (ML) is also on a fast pace for its evolution. ML is an important mainstay of IT sector and with that, a rather central, albeit usually hidden, part of our life. As the technology progresses, the analysis and understanding of data to give good results will also increase as the data is very useful in current aspects. In machine learning, one deals with both supervised and unsupervised types of tasks and generally a classification type problem accounts as a resource for knowledge discovery

1. **Linear Regression Algorithm**

Regression can be termed as a parametric technique which is used to predict a continuous or dependent variable on basis of a provided set of independent variables. This technique is said to be parametric as different assumptions are made on basis of data set.

$$Y = \beta o + \beta 1 X + \in (1)$$

Equation shown in eq.1 is used for simple linear regression. These parameters can be said as:

Y - Variable to be predicted

X - Variable(s) used for making a prediction

βo - When X=0, it is termed as prediction value or can be referred to as intercept term

β1 - when there is a change in X by 1 unit it denotes change in Y. It can also be said as slope term

∈ -The difference between the predicted and actual values is represented by this parameter and represents the residual value. However efficiently the model is trained, tested, and validated, there is always a difference between actual and predicted values which is irreducible error thus we cannot rely completely on the predicted results by the learning algorithm. Alternative methods given by Dieterich can be used for comparing learning algorithms.

2. **Decision Tree:**

Decision Tree is one of the most used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.

It is a tree-structured classifier with three types of nodes. The *Root Node* is the initial node which represents the entire sample and may get split further into further nodes. The *Interior Nodes* represent the features of a data set and the branches represent the decision rules. Finally, the *Leaf Node* represent the outcome. This algorithm is very useful for solving decision-related problems.

With a particular data point, it is run completely through the entirely tree by answering *True/False* questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.

3. **Random Forest Algorithm**

Random forest algorithm is a very accurate algorithm to be used for predicting sales. It is easy to use and understand for the purpose of predicting results of machine learning tasks. In sales prediction, random forest classifier is used because it has decision tree like hyperparameters. The tree model is same as decision tool. Fig.5 shows the relation between decision trees and random forest. To solve regression tasks of prediction by virtue of random forest, the

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

sklearn.ensemble library's random forest regressor class is used. The key role is played by the parameter termed as n_estimators which also comes under random forest regressor. Random forest can be referred to as a meta-estimator used to fit upon numerous decision trees (based on classification) by taking the dataset's different sub-samples. min_samples_split is taken as the minimum number when splitting an internal node if integer number of minimum samples are considered. A split's quality is measured using mse (mean squared error), which can also be termed as feature selection criterion. This also means reduction in variance mae (mean absolute error), which is another criterion for feature selection. Maximum tree depth, measured in integer terms, if equals one, then all leaves are pure or pruning for better model fitting is done for all leaves less than min_samples_split samples.

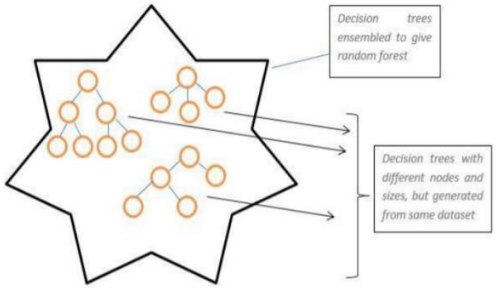

Fig 3.5.1 Trees generation

The above figure 3.5.1 shows the relation between the Decision Tree and the Random Forest models and how the trees are generated

IV. IMPLEMENTATION

1. Importing libraries and reading the dataset

```
In [7]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        %matplotlib inline
        warnings.filterwarnings('ignore')
```

Fig 4.1.1 Importing libraries

```
In [8]: df = pd.read_csv('Train.csv')
        df.head()
```

Out[8]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location
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	

Fig 4.1.2 Reading the dataset

```
In [9]: # statistical info
        df.describe()
```

Out[9]:

	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

Fig 4.1.3. Getting the statistical info of the dataset

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

2. Exploring the dataset

```
In [10]: # datatype of attributes
         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
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Fig 4.2.1 Datatype of attributes

```
In [11]: # check unique values in dataset
         df.apply(lambda x: len(x.unique()))

Out[11]: Item_Identifier              1559
         Item_Weight                   416
         Item_Fat_Content                5
         Item_Visibility              7880
         Item_Type                      16
         Item_MRP                     5938
         Outlet_Identifier              10
         Outlet_Establishment_Year       9
         Outlet_Size                     4
         Outlet_Location_Type            3
         Outlet_Type                     4
         Item_Outlet_Sales            3493
         dtype: int64
```

Fig 4.2.2 checking unique values

3. Data preprocessing

```
In [12]: # check for null values
         df.isnull().sum()

Out[12]: 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
```

Fig 4.3.1 Checking for null values

```
In [13]: # check for categorical attributes
         cat_col = []
         for x in df.dtypes.index:
             if df.dtypes[x] == 'object':
                 cat_col.append(x)
         cat_col

Out[13]: ['Item_Identifier',
          'Item_Fat_Content',
          'Item_Type',
          'Outlet_Identifier',
          'Outlet_Size',
          'Outlet_Location_Type',
          'Outlet_Type']

In [14]: cat_col.remove('Item_Identifier')
         cat_col.remove('Outlet_Identifier')
         cat_col

Out[14]: ['Item_Fat_Content',
          'Item_Type',
          'Outlet_Size',
          'Outlet_Location_Type',
          'Outlet_Type']
```

Fig 4.3.2 Getting for categorical attributes

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

```
In [15]: # print the categorical columns
         for col in cat_col:
             print(col)
             print(df[col].value_counts())
             print()

Item_Fat_Content
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64

Item_Type
Fruits and Vegetables    1232
Snack Foods              1200
Household                 910
Frozen Foods              856
Dairy                     682
Canned                    649
Baking Goods              648
Health and Hygiene        520
Soft Drinks               445
Meat                      425
Breads                    251
Hard Drinks               214
Others                    169
Starchy Foods             148
Breakfast                 110
Seafood                    64
Name: Item_Type, dtype: int64

Outlet_Size
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64

Outlet_Location_Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64

Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
```

Fig 4.3.3 Printing the categorical columns

```
In [16]: # fill the missing values
         item_weight_mean = df.pivot_table(values = "Item_Weight", index = 'Item_Identifier')
         item_weight_mean
```

Out[16]:

Item_Identifier	Item_Weight
DRA12	11.600
DRA24	19.350
DRA59	8.270
DRB01	7.390
DRB13	6.115
...	...
NCZ30	6.590
NCZ41	19.850
NCZ42	10.500
NCZ53	9.600
NCZ54	14.650

1555 rows × 1 columns

```
In [17]: miss_bool = df['Item_Weight'].isnull()
         miss_bool

Out[17]: 0       False
         1       False
         2       False
         3       False
         4       False
                 ...
         8518    False
         8519    False
         8520    False
         8521    False
         8522    False
         Name: Item_Weight, Length: 8523, dtype: bool

In [18]: for i, item in enumerate(df['Item_Identifier']):
             if miss_bool[i]:
                 if item in item_weight_mean:
                     df['Item_Weight'][i] = item_weight_mean.loc[item]['Item_Weight']
                 else:
                     df['Item_Weight'][i] = np.mean(df['Item_Weight'])

In [19]: df['Item_Weight'].isnull().sum()
Out[19]: 0
```

Fig 4.3.4 Filling missing values for Item weights

```
In [20]: outlet_size_mode = df.pivot_table(values='Outlet_Size', columns='Outlet_Type', aggfunc=(lambda x: x.mode()[0]))
         outlet_size_mode
```

Out[20]:

Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Outlet_Size	Small	Small	Medium	Medium

```
In [21]: miss_bool = df['Outlet_Size'].isnull()
         df.loc[miss_bool, 'Outlet_Size'] = df.loc[miss_bool, 'Outlet_Type'].apply(lambda x: outlet_size_mode[x])

In [22]: df['Outlet_Size'].isnull().sum()
Out[22]: 0
```

Fig 4.3.5 Filling missing values for Outletsize

```
In [23]: sum(df['Item_Visibility']==0)
Out[23]: 526

In [24]: # replace zeros with mean
         df.loc[:, 'Item_Visibility'].replace([0], [df['Item_Visibility'].mean()], inplace=True)

In [25]: sum(df['Item_Visibility']==0)
Out[25]: 0
```

Fig 4.3.6 Filling missing values for Item visibility

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

```
In [26]: # combine item fat content
         df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({'LF':'Low Fat', 'reg':'Regular', 'low fat':'Low Fat'})
         df['Item_Fat_Content'].value_counts()

Out[26]: Low Fat    5517
         Regular    3006
         Name: Item_Fat_Content, dtype: int64
```

Fig 4.3.7 Combining item fat content

```
In [27]: df['New_Item_Type'] = df['Item_Identifier'].apply(lambda x: x[:2])
         df['New_Item_Type']

Out[27]: 0       FD
         1       DR
         2       FD
         3       FD
         4       NC
                 ..
         8518    FD
         8519    FD
         8520    NC
         8521    FD
         8522    DR
         Name: New_Item_Type, Length: 8523, dtype: object

In [28]: df['New_Item_Type'] = df['New_Item_Type'].map({'FD':'Food', 'NC':'Non-Consumable', 'DR':'Drinks'})
         df['New_Item_Type'].value_counts()

Out[28]: Food              6125
         Non-Consumable    1599
         Drinks             799
         Name: New_Item_Type, dtype: int64

In [29]: df.loc[df['New_Item_Type']=='Non-Consumable', 'Item_Fat_Content'] = 'Non-Edible'
         df['Item_Fat_Content'].value_counts()

Out[29]: Low Fat       3918
         Regular       3006
         Non-Edible    1599
         Name: Item_Fat_Content, dtype: int64
```

Fig 4.3.8New item type

```
In [30]: # create small values for establishment year
         df['Outlet_Years'] = 2013 - df['Outlet_Establishment_Year']

In [31]: df['Outlet_Years']

Out[31]: 0       14
         1        4
         2       14
         3       15
         4       26
                 ..
         8518    26
         8519    11
         8520     9
         8521     4
         8522    16
         Name: Outlet_Years, Length: 8523, dtype: int64

In [32]: df.head()
```

Out[32]:

em_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	New_Item_Type	Outlet_Years
Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380	Food	14
oft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228	Drinks	4
Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700	Food	14
ruits and egetables	182.0950	OUT010	1998	Small	Tier 3	Grocery Store	732.3800	Food	15
ousehold	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	Non-Consumable	26

Fig 4.3.9 Creating small values for establishment year

V. DATA VISUALISATION

```
In [33]: sns.distplot(df['Item_Weight'])

Out[33]: <AxesSubplot:xlabel='Item_Weight', ylabel='Density'>
```

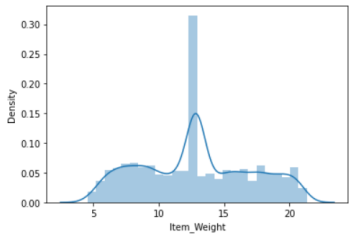

Fig 5.1 Item weight plot

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

```
In [34]: sns.distplot(df['Item_Visibility'])
Out[34]: <AxesSubplot:xlabel='Item_Visibility', ylabel='Density'>
```

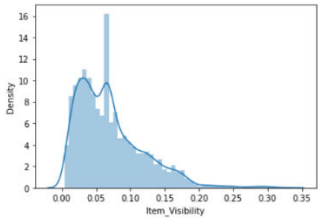

Fig 5.2 Item visibility plot

```
In [35]: sns.distplot(df['Item_MRP'])
Out[35]: <AxesSubplot:xlabel='Item_MRP', ylabel='Density'>
```

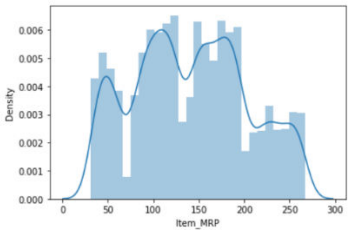

Fig 5.3 Item MRP plot

```
In [36]: sns.distplot(df['Item_Outlet_Sales'])
Out[36]: <AxesSubplot:xlabel='Item_Outlet_Sales', ylabel='Density'>
```

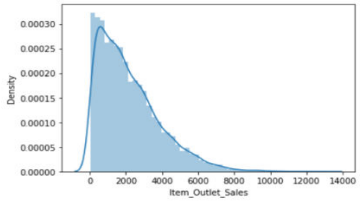

Fig 5.4 Item outlet sales plot

```
In [37]: # log transformation
         df['Item_Outlet_Sales'] = np.log(1+df['Item_Outlet_Sales'])
In [38]: sns.distplot(df['Item_Outlet_Sales'])
Out[38]: <AxesSubplot:xlabel='Item_Outlet_Sales', ylabel='Density'>
```

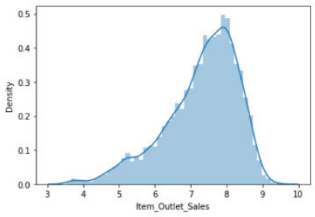

Fig 5.5 Log transformation plot

```
In [39]: sns.countplot(df["Item_Fat_Content"])
Out[39]: <AxesSubplot:xlabel='Item_Fat_Content', ylabel='count'>
```

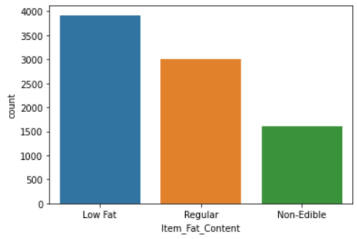

Fig 5.6 Item fat content plot

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

```
In [40]: # plt.figure(figsize=(15,5))
         l = list(df['Item_Type'].unique())
         chart = sns.countplot(df["Item_Type"])
         chart.set_xticklabels(labels=l, rotation=90)

Out[40]: [Text(0, 0, 'Dairy'),
          Text(1, 0, 'Soft Drinks'),
          Text(2, 0, 'Meat'),
          Text(3, 0, 'Fruits and Vegetables'),
          Text(4, 0, 'Household'),
          Text(5, 0, 'Baking Goods'),
          Text(6, 0, 'Snack Foods'),
          Text(7, 0, 'Frozen Foods'),
          Text(8, 0, 'Breakfast'),
          Text(9, 0, 'Health and Hygiene'),
          Text(10, 0, 'Hard Drinks'),
          Text(11, 0, 'Canned'),
          Text(12, 0, 'Breads'),
          Text(13, 0, 'Starchy Foods'),
          Text(14, 0, 'Others'),
          Text(15, 0, 'Seafood')]
```

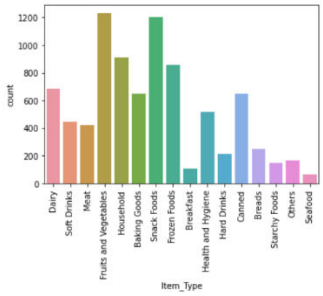

Fig 5.7 Count plot of item type

```
In [41]: sns.countplot(df['Outlet_Establishment_Year'])
Out[41]: <AxesSubplot:xlabel='Outlet_Establishment_Year', ylabel='count'>
```

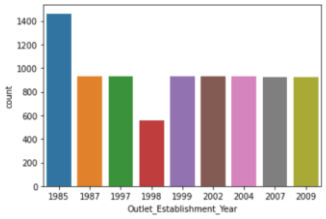

Fig 5.8 Outlet establishment year plot

```
In [42]: sns.countplot(df['Outlet_Size'])
Out[42]: <AxesSubplot:xlabel='Outlet_Size', ylabel='count'>
```

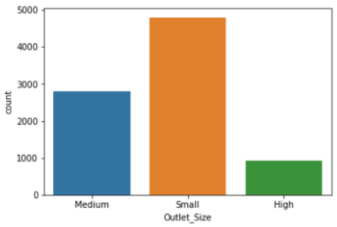

Fig 5.9 Outlet size plot.

```
In [43]: sns.countplot(df['Outlet_Location_Type'])
Out[43]: <AxesSubplot:xlabel='Outlet_Location_Type', ylabel='count'>
```

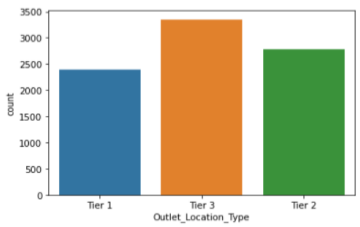

Fig5.10 Outlet location type plot

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

```
In [44]: sns.countplot(df['Outlet_Type'])
Out[44]: <AxesSubplot:xlabel='Outlet_Type', ylabel='count'>
```

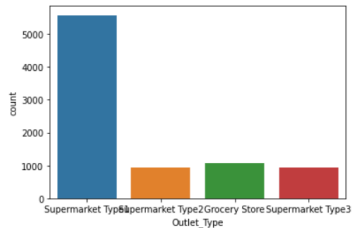

Fig 5.11 Outlet type plot

```
In [45]: corr = df.corr()
         sns.heatmap(corr, annot=True, cmap='coolwarm')
Out[45]: <AxesSubplot:>
```

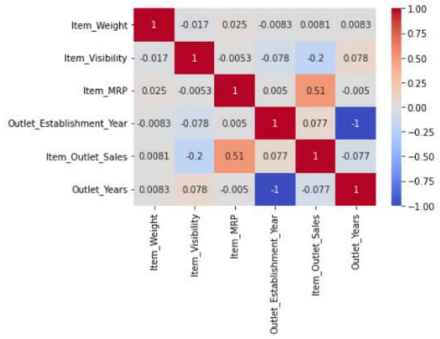

Fig 5.12 Correlation matrix

VI. EXPERIMENTAL RESULTS

6.1 Preprocessing for categorical attributes

Label encoding

```
In [47]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         df['Outlet'] = le.fit_transform(df['Outlet_Identifier'])
         cat_col = ['Item_Fat_Content', 'Item_Type', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type', 'New_Item_Type']
         for col in cat_col:
             df[col] = le.fit_transform(df[col])
```

Onehot Encoding

```
In [48]: df = pd.get_dummies(df, columns=['Item_Fat_Content', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type', 'New_Item_Type'])
         df.head()
Out[48]:
```

Outlet_Years	Outlet	...	Outlet_Location_Type_0	Outlet_Location_Type_1	Outlet_Location_Type_2	Outlet_Type_0	Outlet_Type_1	Outlet_Type_2	Outlet_Type_3	N
14	9	...	1	0	0	0	1	0	0	
4	3	...	0	0	1	0	0	1	0	
14	9	...	1	0	0	0	1	0	0	
15	0	...	0	0	1	1	0	0	0	
26	1	...	0	0	1	0	1	0	0	

Fig 6.1 label, Onehot encoding

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

6.2 Model training and splitting the dataset

```
In [49]: X = df.drop(columns=['Outlet_Establishment_Year', 'Item_Identifier', 'Outlet_Identifier', 'Item_Outlet_Sales'])
         y = df['Item_Outlet_Sales']
```

Model Training

```
In [62]: from sklearn.model_selection import cross_val_score
         from sklearn.metrics import mean_squared_error,confusion_matrix
         def train(model, X, y):
             # train the model
             model.fit(X, y)

             # predict the training set
             pred = model.predict(X)

             # perform cross-validation
             cv_score = cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv=5)
             cv_score = np.abs(np.mean(cv_score))

             print("Model Report")
             print("MSE:",mean_squared_error(y,pred))
             print("CV Score:", cv_score)
```

Fig 6.2 Dataset split and model training

6.3 Linear regression

```
In [63]: from sklearn.linear_model import LinearRegression
         model = LinearRegression(normalize=True)
         train(model, X, y)
         coef = pd.Series(model.coef_, X.columns).sort_values()
         coef.plot(kind='bar', title="Model Coefficients")
```

```
Model Report
MSE: 0.2880787839956559
CV Score: 0.28920602795798417
```

```
Out[63]: <AxesSubplot:title={'center':'Model Coefficients'}>
```

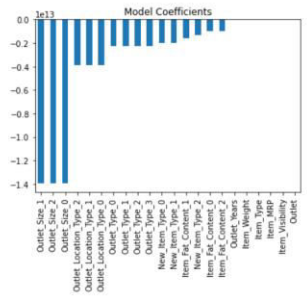

Fig 6.3 Linear Regression

The above figure 6.3 shows prediction of linear regression and taken title as Model Coefficient where we have got the MSE = 0.2880787833 and CV score = 0.28920602795. MSE and CV score is very minimal so that this model trained and predicted efficiently. The model normalizes the cofficients so that when x increases, y decreases or y increases, x decreases. So it is proven the basic models also predict efficiently.

6.4 Decision Tree Regressor

```
In [47]: from sklearn.tree import DecisionTreeRegressor
         model = DecisionTreeRegressor()
         train(model, X, y)
         coef = pd.Series(model.feature_importances_, X.columns).sort_values(ascending=False)
         coef.plot(kind='bar', title="Feature Importance")
```

```
Model Report
MSE: 5.5534030638578795e-34
CV Score: 0.575091932907843
```

```
Out[47]: <AxesSubplot:title={'center':'Feature Importance'}>
```

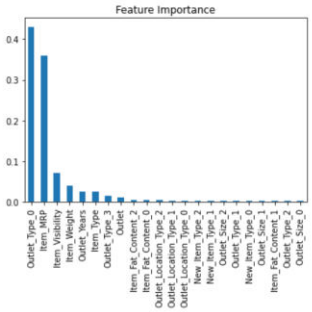

Fig 6.4 Decision Tree Regressor

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

The above figure 6.4 shows prediction of linear regression and taken title as Feature Importance where we have got the MSE = 5.55340306385e-34 and CV score = 0.5750919329. MSE is minimal and CV score is low so that this model trained efficiently and it predicted well.

6.5 Random Forest Regressor

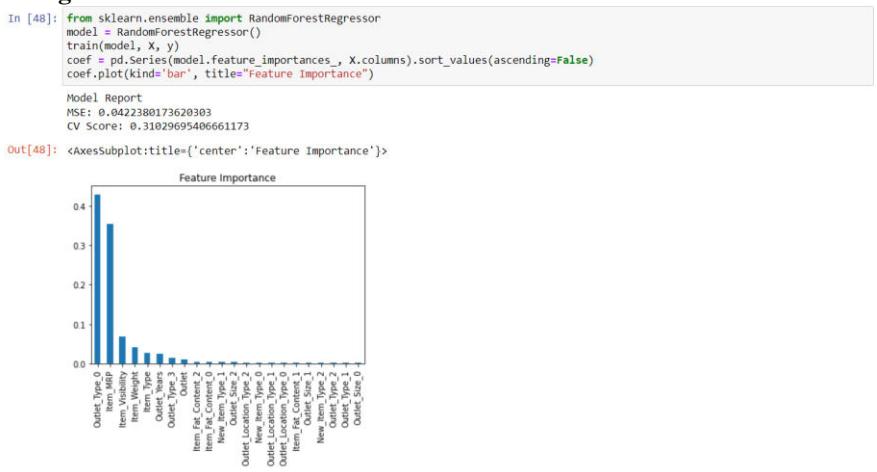

Fig 6.5 Random Forest Regressor

The above figure 5.5 shows prediction of linear regression and taken title as Feature Importance where we have got the MSE = 0.0422380173 and CV score = 0.3102969540. MSE is minimal and CV score is low so that this model trained efficiently and also it predicted well.

VII. CONCLUSION AND FUTURE SCOPE

The basics of machine learning and the associated data processing and modeling algorithms are described and 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.

Multiple instances parameters and various factors can be used to make this sales prediction more innovative and successful. Accuracy, which plays a key role in prediction-based systems, can be significantly increased as the number of parameters used are increased. Also, a look into how the sub-models work can lead to increase in productivity of system. The paper can be further collaborated in a web-based application or in any device supported with an in-built intelligence by virtue of 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 corporation's tasks on the whole.

REFERENCES

[1] Cerrada, M., & Aguilar, J. (2008). Reinforcement learning in system identification.
[2] Downey, A. B. (2011). Think stats. "O'Reilly Media, Inc.".
[3] Géron, A. (2019). Hands-On Machine Learning with ScikitLearn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.
[4] Learning, M. (1994). Neural and Statistical Classification. Editors D. Mitchie et. al, 350.
[5] MacKay, D. J., & Mac Kay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press.
[6] Mitchell, T. M. (1999). Machine learning and data mining. Communications of the ACM, 42(11), 30-36. Cambridge University, UK, 32, 34.
[7] Saltz, J. S., & Stanton, J. M. (2017). An introduction to data science. Sage Publications.
[8] Shashua, A. (2009). Introduction to machine learning: Class notes 67577. arXiv preprint arXiv:0904.3664.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.118|

||Volume 11, Issue 6, June 2022||

| DOI:10.15680/IJIRSET.2022.1106182 |

[9] Smola, A., & Vishwanathan, S. V. N. (2008). Introduction to machine learning.
[10] Quinlan, J. R. (2014). C4. 5: programs for machine learning. Elsevier. In Reinforcement Learning. IntechOpen.
[11] Welling, M. (2011). A first encounter with Machine Learning. Irvine, CA.: University of California, 12.

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 8.118

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

Scan to save the contact details