BIG MART SALES PRDICTION USING MACHINE LEARNING

Project Report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

By 1.Kharan Jagabathula (12107230)

2.Ayush Singh Bhati (12107726)

Section : K21ML

Under the supervision of

Dr. Dhanpratap Singh (25706)

Transforming Education Transforming India

Department of Intelligent Systems School of Computer Science Engineering Lovely Professional University, Jalandhar

ABSTRACT

Nowadays many shopping malls keep track of individual item sales data to forecast future client demand and adjust inventory management. To be ahead of the competition and earn more profit one needs to create a model which will help to predict and find out the sales of the various product present in the particular store. So to predict out the sales for the big mart one need to use the very important tool i.e. Machine Learning (ML). ML is that field of computer science which gives machines i.e. computers the ability to learn without doing any type of programming. Using the concepts of machine and basics of data science one can build a model which can help to predict the sales of the big mart. Because of increasing competition among various shopping complex, one needs to have some predictive model which could help to gain some useful insights to maximize the profit and be ahead of the competitors.

INTRODUCTION

The daily competition between different malls as well as big malls is becoming more and more intense because of the rapid rise of international supermarkets and online shopping. Every mall or mart tries to provide personal and short-term donations or benefits to attract more and more customers daily, such as the sales price of everything which is usually predicted to be managed through different ways such as corporate asset management, logistics, and transportation service, etc. Current machine learning algorithms that are very complex and provide strategies for predicting or predicting long-term demand for a company's sales, which now also help in overcoming budget and computer programs.

In this report, we basically discuss the subject of specifying a large mart sale or predicting an item for a customer’s future need in a few supermarkets in various locations and products that support the previous record. Various ML algorithms such as linear regression, random forest, etc. are used to predict sales volume. As we know, good marketing is probably the lifeblood of all organizations, so sales forecasting now plays an important role in any shopping mall. It is always helpful to predict the best, and develop business strategies about useful markets and to improve market knowledge. Regular sales forecasting research can help in-depth analysis of pre-existing conditions and conditions and then, assumptions are often used in terms of customer acquisition, lack of funding, and strength before setting budgets and marketing plans for the coming year.

In other words, sales forecasts are predicted on existing services of the past. In-depth knowledge of the past is required to develop and enhance market opportunities no matter what the circumstances, especially the external environment, which allows to prepare for the future 2 needs of the business. Extensive research is ongoing in the retailer's domain to predict longterm sales demand. An important and effective method used to predict the sale of a mathematical method, also called the conventional method, but these methods take more time to predict sales. And these methods could not manage indirect data so to overcome these problems in traditional methods the machine learning techniques used. ML methods can handle not only indirect data but also large data sets well.

PROBLEM STATEMENT

Due to increasing competition many malls and bigmart are trying their best to stay ahead in competition. In order to find out what are the various factors which affect the sales of bigmart and what strategies one needs to employ in order to gain more profit one need to have some model on which they can rely. So a predictive model can be made which could help to gain useful information and increase profit.

OBJECTIVES

Objectives of these project are:

a) Predicting future sales from a given dataset.

b) To understand the key features that are responsible for the sale of a particular product.

c) Find the best algorithm that will predict sales with the greatest accuracy.

PREDICTING SALES USING MACHINE LEARNING

Machine learning is a powerful tool that can be used to predict sales. Machine learning algorithms can be trained on historical sales data to learn the relationships between different factors and sales. Once trained, the machine learning model can be used to predict sales for future periods.

Machine learning algorithms can be used to predict sales in several ways. One common approach is to use a supervised learning algorithm. Supervised learning algorithms are trained on a dataset of historical sales data, where each input example is paired with a corresponding output value (i.e., the sales value). The algorithm learns to predict the output value for new input examples.

Another approach to sales prediction using machine learning is to use an unsupervised learning algorithm. Unsupervised learning algorithms are trained on a dataset of unlabelled data, meaning that the output values are not known. The algorithm learns to identify patterns in the data and group similar data points together. Unsupervised learning algorithms can be used to identify customer segments or product categories that are likely to have high sales

Benefits of using machine learning for sales prediction

- More accurate sales predictions
- Improved customer satisfaction
- Reduced costs
- Increased revenue
- Competitive advantage

MODELS USED TO PREDICT SALES

There are several different machine learning algorithms that can be used to predict sales. Some of the most common algorithms include:

- **Linear regression**: Linear regression is a simple but effective algorithm for predicting sales. The algorithm learns a linear relationship between the input features and the sales value.
- **Random forest**: Random forests are a type of ensemble learning algorithm. Ensemble learning algorithms combine the predictions of multiple individual models to produce a more accurate prediction. Random forests are particularly effective for predicting sales because they are able to learn complex relationships between the input features and the sales value.
- **XG Boost**: XG Boost is another type of ensemble learning algorithm that is often used for sales prediction. XG Boost is similar to random forests, but it is able to learn more complex relationships between the input features and the sales value.

- *Other sales prediction models include*:

1. Time series analysis
2. Neural networks
3. Support vector machines

ALGORITHMS USED

1. LINEAR REGRESSION (LR)

As we know Regression can be termed as a parametric technique which means we can predict a continuous or dependent variable on the basis of a provided datasets of independent variables.

The Equation of simple LR is:

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

Y : It is basically the variable which we used as a predicted value.

X : It is a variable(s) which is used for making a prediction.

βo : It is said to be a prediction value when X=0.

β1 : when there is a change in X value by 1 unit then Y value is also changed. It can also be said as slope.

2. XGBOOST REGRESSION

XGBoost stands for extreme Gradient Boosting. The implementation of an algorithm designed for the efficient operation of computer time and memory resources. Boosting is a sequential process based on the principle of the ensemble. This includes a collection of lower learners as well improves the accuracy of forecasts .No model prices n heavy for any minute t, based on the results of the previous t-speed. Well-calculated results are given less weight, and the wrong ones are weighed down. With this algorithm system 11 The XGBoost model uses stepwise, ridge regression internally, automatically selecting features as well as deleting multicollinearity.

CODE

In [54]:
```
import numpy as np                                      # for creating numpy arrays
import pandas as pd                                     # for creating pandas dataframe
import matplotlib.pyplot as plt                         # for Visualization
import seaborn as sns                                   # for Visualization
from sklearn.preprocessing import LabelEncoder          # for Label Encoding on categorical data
from sklearn.model_selection import train_test_split    # for splitting data
from xgboost import XGBRegressor                        # ML Model
from sklearn.linear_model import LinearRegression       # ML Model
from sklearn import metrics                             # to find performance of ML model
```

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

In [3]:
```
df
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	(
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	NaN	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	

In [4]:
```
# first 5 rows of the dataframe
df.head()
```

Out[4]:

	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	

In [5]:
```
# number of data points(Rows) & number of features(Columns)
df.shape
```

Out[5]:
```
(8523, 12)
```

In [6]:
```
# getting some information about null values and data types
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
```

In [7]:
```
# checking for missing values
df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
df.describe()
```

Out[8]:

	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

In [63]:
```
df.dropna(inplace = True)
```

In [64]:
```
df.shape
```

Out[64]:
```
(7060, 12)
```

Data Pre-processing

In [65]:
```
#Data Pre-Processing
```

In [66]:
```
df.head()
```

Out[66]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_
0	156	9.30	1	0.016047	4	249.8092	9	1999	1	
1	8	5.92	2	0.019278	14	48.2692	3	2009	1	
2	662	17.50	1	0.016760	10	141.6180	9	1999	1	
3	1121	19.20	2	0.000000	6	182.0950	0	1998	3	
4	1297	8.93	1	0.000000	9	53.8614	1	1987	0	

In [67]:
```
df['Item_Fat_Content'].value_counts()
```

Out[67]:
```
1    4222
2    2388
0     260
4     106
3      84
Name: Item_Fat_Content, dtype: int64
```

Label Encoding

In [68]:
```
encoder = LabelEncoder()
```

In [69]:
```
df['Item_Identifier'] = encoder.fit_transform(df['Item_Identifier'])

df['Item_Fat_Content'] = encoder.fit_transform(df['Item_Fat_Content'])

df['Item_Type'] = encoder.fit_transform(df['Item_Type'])

df['Outlet_Identifier'] = encoder.fit_transform(df['Outlet_Identifier'])

df['Outlet_Size'] = encoder.fit_transform(df['Outlet_Size'])

df['Outlet_Location_Type'] = encoder.fit_transform(df['Outlet_Location_Type'])

df['Outlet_Type'] = encoder.fit_transform(df['Outlet_Type'])
```

In [70]:
```
df.head()
```

Out[70]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_
0	156	9.30	1	0.016047	4	249.8092	7	1999	1	
1	8	5.92	2	0.019278	14	48.2692	3	2009	1	
2	660	17.50	1	0.016760	10	141.6180	7	1999	1	
3	1117	19.20	2	0.000000	6	182.0950	0	1998	3	
4	1293	8.93	1	0.000000	9	53.8614	1	1987	0	

In [71]:
```
X = df.drop(columns='Item_Outlet_Sales', axis=1)
Y = df['Item_Outlet_Sales']
```

In [72]:
```
print(X)
```

```
      Item_Identifier  Item_Weight  Item_Fat_Content  Item_Visibility  \
0                 156        9.300                 1         0.016047
1                   8        5.920                 2         0.019278
2                 660       17.500                 1         0.016760
3                1117       19.200                 2         0.000000
4                1293        8.930                 1         0.000000
...               ...          ...               ...              ...
8518              369        6.865                 1         0.056783
8519              893        8.380                 2         0.046982
8520             1353       10.600                 1         0.035186
8521              679        7.210                 2         0.145221
8522               50       14.800                 1         0.044878

      Item_Type  Item_MRP  Outlet_Identifier  Outlet_Establishment_Year  \
0             4  249.8092                  7                       1999
1            14   48.2692                  3                       2009
2            10  141.6180                  7                       1999
3             6  182.0950                  0                       1998
4             9   53.8614                  1                       1987
...         ...       ...                ...                        ...
8518         13  214.5218                  1                       1987
```

In [73]:
```
print(Y)
```

```
0       3735.1380
1        443.4228
2       2097.2700
3        732.3800
4        994.7052
          ...
8518    2778.3834
8519     549.2850
8520    1193.1136
8521    1845.5976
8522     765.6700
Name: Item_Outlet_Sales, Length: 7060, dtype: float64
```

Model Training

In [74]:
```
#Machine Learning Model Training
```

In [75]:
```
#Linear Regressor
```

In [76]:
```
regressor = LinearRegression()
```

In [77]:
```

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

In [78]:
```
regressor.fit(X_train, Y_train)
```

Out[78]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [79]:
```
LinearRegression()
```

Out[79]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [80]:
```python
# prediction on training data
training_data_prediction = regressor.predict(X_train)
```

In [81]:
```python
# R squared Value
r2_train = metrics.r2_score(Y_train, training_data_prediction)
```

In [82]:
```python
print('R Squared value = ', r2_train)
```

```
R Squared value =  0.4647954241224679
```

In [83]:
```python
# prediction on test data
test_data_prediction = regressor.predict(X_test)
```

In [84]:
```python
# R squared Value
r2_test = metrics.r2_score(Y_test, test_data_prediction)
```

In [85]:
```python
print('R Squared value = ', r2_test)
```

```
R Squared value =  0.4673130731538986
```

XGboost

In [95]:
```python
!pip install xgboost
```

```
   ---------------------------------------- 98.5/99.7 MB 9.2 MB/s eta 0:00:01
   ---------------------------------------- 99.0/99.7 MB 9.2 MB/s eta 0:00:01
   ---------------------------------------- 99.6/99.7 MB 10.9 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 10.7 MB/s eta 0:00:01
   ---------------------------------------- 99.7/99.7 MB 6.5 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-2.0.1

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [96]:
```python
from xgboost import XGBRegressor
```

In [97]:
```python
#XGBoost Regressor
```

In [98]:
```python
regressor = XGBRegressor()
```

In [99]:
```python
regressor.fit(X_train, Y_train)
```

Out[99]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [101]: # prediction on training data
          training_data_prediction = regressor.predict(X_train)

In [102]: # R squared Value
          r2_train = metrics.r2_score(Y_train, training_data_prediction)

In [103]: print('R Squared value = ', r2_train)

          R Squared value =  0.8636824834453669

In [104]: # prediction on test data
          test_data_prediction = regressor.predict(X_test)

In [105]: # R squared Value
          r2_test = metrics.r2_score(Y_test, test_data_prediction)

In [106]: print('R Squared value = ', r2_test)

          R Squared value =  0.431570074014475

In [107]: #As we can clearly see Linear Regression performs slighlty better
```

Conclusion

So from this project we conclude that a smart sales forecasting program is required to manage vast volumes of knowledge for business organizations. The Algorithms which are presented in this report, Linear Regression and XGBoost regression provide an effective method for data sharing as well as decision-making and provide new approaches that are used for better identifying consumer needs and formulate marketing plans that are going to be implemented. The outcomes of ML algorithms which are done in this project will help us to pick the foremost suitable demand prediction algorithm and with the aid of which BigMart will prepare its marketing campaigns.