BIGMART SALES PREDICTION
BY USING MACHINE LEARNING

A Mini-project Report

Submitted by

SOURABH GHAGHRE

in partial fulfilment of the requirements for the award of the degree of

Master of Technology

in

Mechanical Engineering

(Industrial Engineering & Management)

Department of Mechanical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

NIT CAMPUS PO, CALICUT

KERALA, INDIA 673601

May 2023

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: NITC **Signature:**

Date: **Name:**

Roll No.:

ABSTRACT

In recent times, the number of supermarkets and their franchises has significantly increased. To avoid losses and boost sales, these businesses require accurate sales forecasts, which can be a time-consuming and challenging task. To address this issue, a machine learning model that utilizes the Random Forest Regressor and Artificial Neural Network (ANN) to predict item sales. This model can help marts develop effective recruitment strategies, anticipate potential challenges, motivate their sales team, project revenue, support future marketing plans, and provide many other benefits. The study reveals that the ANN algorithm outperforms other methods in predicting outlet item sales in large marts. The mini-project can aid big marts' management in making decisions regarding maximizing item and outlet placement, enhancing customer experience, increasing sales, and driving revenue growth and business expansion.

Machine Learning is a technology that allows machines to make accurate predictions without being explicitly programmed to do so. It involves creating models and algorithms that analyze input data using statistics to predict an output, while also modifying the output based on new data. These models can be used in various areas and customized to meet specific goals. In this study, the focus is on Big Mart Shopping Centre, where machine learning was employed to forecast sales of different items and to examine the impact of various factors on sales. The research involved developing a predictive model using techniques such as Random Forest and Linear Regression on a dataset with multiple features, resulting in highly precise predictions. The results can be leveraged to make informed decisions to enhance sales.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 SALES FORECASTING:

Sales forecasting is a critical process for predicting future demand, allowing companies to allocate resources efficiently, estimate achievable revenue, and plan for future growth. Having access to the right data and drawing the right conclusions from it are crucial to effective forecasting. Accurate sales forecasts can aid in planning and reducing unnecessary costs. Sales forecasting is essential for retailing, logistics, manufacturing, marketing, and wholesaling businesses. The importance of forecasting lies in its ability to determine production volume, form the basis for budgeting, aid in deciding the extent of advertising, and assist in making decisions regarding expansion and production changes.

1.2 SCOPE

This study focuses on analyzing the sales data of Bigmart supermarket and is restricted to the item sales only. The approach used in this study relies on analyzing the purchasing behavior of customers and the aggregated data regarding the sales of products in Bigmart outlets.

1.3 RESEARCH OBJECTIVES:

The objective of the mini project is to develop an improved forecasting model that can predict the sales of outlet items at Bigmart. The study aims to compare the effectiveness of predictive models developed using machine learning and deep learning techniques, based on the stated objectives.

1. The duties include examining literature that pertains to sales forecasting at Big Mart by utilizing machine learning models and deep learning techniques, as well as analyzing the material.

2. Performing data preprocessing on the dataset.

3. Create predictive models using different learning techniques on the dataset, and then measure the performance of these models using appropriate metrics. Finally, select the model with the highest accuracy based on the chosen performance metric.

1.4 RESEARCH QUESTION

Which advanced machine learning algorithm and deep neural network have better predictive performance in forecasting sales of outlet items due to their ability to interpret complex relationships among the features?

1.5 OUTLINE OF THE REPORT

The report is organized as follows: Chapter 1 is the introduction, followed by a literature review in chapter 2, chapter 3 describe machine learning techniques, chapter 4 describe methodology and design, chapter 5 describe data pre-processing, chapter 6 describe evaluation metrics, chapter 7 describe model building, chapter 8 describe result and chapter 9 describe conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 BIG MART SALES PREDICTIONS

BMSP-ML: big mart sales prediction using different machine learning techniques, Ali et al. (2023) this paper elaborates the main objective is to identify the factors that can forecast sales patterns through this research and can gain insights into the data and employ various machine learning techniques to predict sales.

Magablah conducted a comparative study to predict big mart sales with the help of machine learning and deep learning techniques. The purpose of this thesis is to create a more precise predictive model for projecting the output sales in Bigmart. The research aims to evaluate and compare the effectiveness of various machine learning and deep learning techniques in achieving the desired objectives.

A comparative study of big mart sales prediction, Behera et al. (2019), this paper elaborates Forecasting forthcoming customer demand and adjusting inventory management accordingly.

Two-Level Statistical Model for Big Mart Sales Prediction, Pamula et al.(2018), this paper elaborate the method for projecting sales of a product from a specific outlet involves a two-step process

Big mart sales prediction and analysis, Choudhary et al.(2007), By utilizing different features of a dataset obtained from Big Mart, and following a specific approach for constructing a predictive model, extremely precise outcomes are produced.

CHAPTER 3
MACHINE LEARNING TECHNIQUES

Definition of Machine Learning: Machine learning is a branch of artificial intelligence that deals with the development of algorithms that can learn from data and improve their performance at a specific task over time. The machine learning approach involves training a computer program on a large amount of data and then using this experience to make predictions or decisions on new, unseen data. Machine learning algorithms can be used for a wide range of tasks, including image recognition, natural language processing, recommendation systems, fraud detection, and many others.

3.1 MACHINE LEARNING ALGORITHMS

Machine learning algorithms analyze and interpret data, and make predictions or decisions based on that analysis. Machine learning algorithms learn from data, meaning they improve their performance over time by adjusting their parameters based on the data they are fed for data set Supervised regression in machine learning is a task of predicting a continuous output variable based on one or more input variables. There are several machine learning algorithms that can be used for supervised regression, including:

3.1.1 Linear Regression: This algorithm models the relationship between the input variables and output variable as a linear equation.

3.1.2 Decision Tree Regression: To create a predictive model for the output variable, decision trees utilize input variables and recursively divide the data.

3.1.3 Random Forest Regression: Random Forest regression is an ensemble learning method that combines multiple decision trees to create a more accurate model.

3.1.4 Support Vector Regression: Support vector regression finds a hyperplane that separates the input and output variables with the maximum margin.

3.1.5 Gradient Boosting Regression: Gradient boosting regression builds a model by combining weak models in a stage-wise fashion, with each new model trying to improve on the errors of the previous model.

3.2 DEEP LEARNING

It is a subfield of machine learning that involves the use of artificial neural networks to learn and make predictions or decisions from complex data. Deep learning algorithms are designed to mimic the way the human brain works, with many layers of interconnected nodes that can recognize patterns and relationships in large datasets. The term "deep" in deep learning refers to the fact that these networks can have many layers, sometimes numbering in the hundreds or even thousands. These layers allow the network to learn increasingly complex representations of the input data, which can be used to make predictions or decisions with high accuracy. Figure 3.1 illustrate the ANN Algorithm.

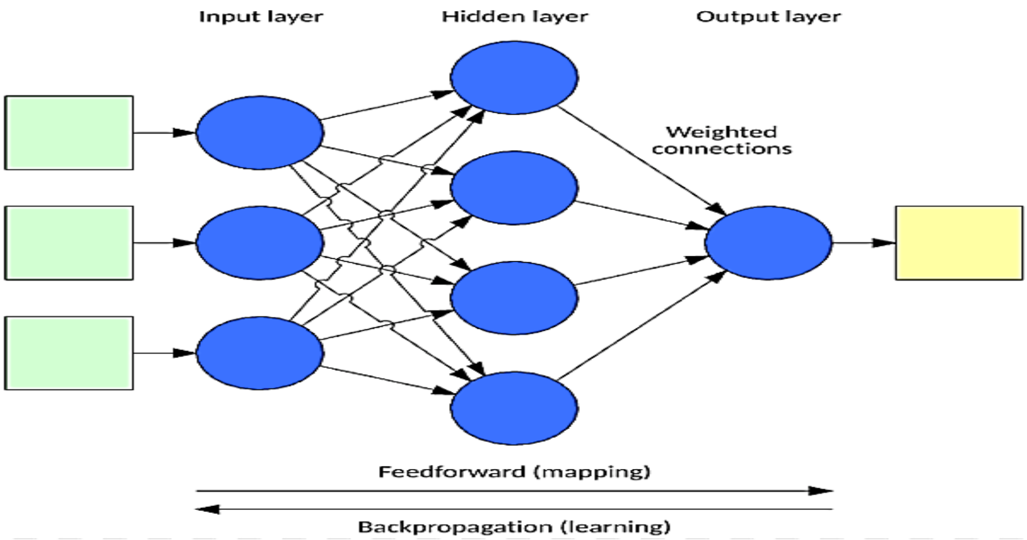


Figure 3.1. ANN Algorithm

CHAPTER 4
METHODOLOGY AND DESIGN

4.1 DATASET AND COLLECTION METHOD

A collection of data points that a computer may use to analyze and anticipate a situation as a whole. Internet information was gathered for the Kaggle.com website. The test data set used in this study comprises 8542 rows as well as 12 categories, which have been trained to deliver the most accurate prediction outcomes. A brief introduction of the dataset is shown in the table 4.1.

Table 4.1 Dataset Description

Column Name	Description
Item_Identifier	Unique product ID
Item_Weight	Weight of product
Item_Fat_Content	Checks the Concentration
Item_Fat_Content	The % of total display area
Type of Item	Category of product
MRP of Item	Maximum Retail Price for a product
Outlet_Identifier	ID of store
Establishment Year of outlet	Store was established
Size of outlet	The size of the store
Outlet Location	city Tiers
Type of outlet	Grocery store
Outlet Sales	Sales

4.2 METHODOLOGY:

The methodology for building machine learning models involves several steps:

1. Problem Definition: The problem to be solved and the goals to be achieved needs to be defined clearly.
2. Data Collection: Relevant data needs to be collected to for training and testing the model as the quality and quantity of data will have an impact on the model's performance.
3. Data Preparation: Clean and preprocess the data, which includes tasks such as handling missing values, handling outliers, normalizing data, and transforming data.
4. Feature Engineering: Select or create features that will be used to train the model. Feature engineering is a crucial step in building accurate and efficient models.
5. Model Selection: Choose the appropriate machine learning algorithm for the problem you want to solve.
6. Training of Model: The chosen model should be trained on the training data in order to grasp the underlying patterns and relationships that exist within the data.
7. Model Evaluation: Evaluate the performance of the model on a separate validation or test dataset. This helps to estimate the model's accuracy and identify potential issues.
8. Model Deployment: Once the model is trained and validated, deploy it to a production environment where it can be used to Perform predictions on novel data.
9. Monitoring and Maintenance: Continuously check the model's behavior and update the model as needed to maintain its accuracy and effectiveness over time. The flow diagram of Methodology is shown in Figure 4.1.

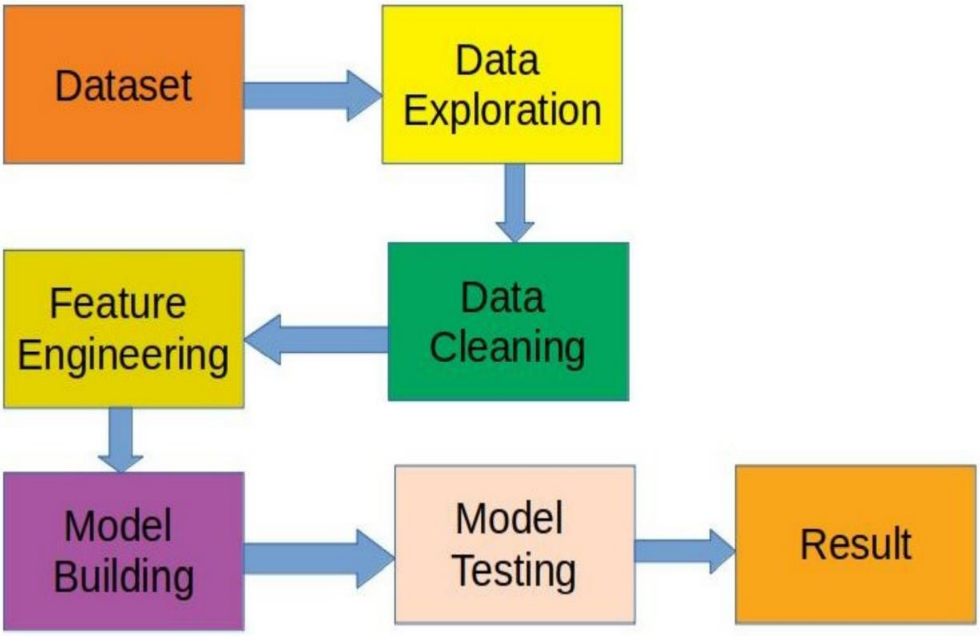


Figure 4.1. Steps followed for obtaining results

CHAPTER 5
DATA PREPROCESSING

5.1 FILLING MISSING VALUES:

we will find and fill the missing values with either mean, mode, or median.

- Mean used for the numerical dataset

```
df_train ['Item_Weight'].fillna(df_train ['Item_Weight'].mean(),inplace=True)
df_test ['Item_Weight'].fillna(df_test ['Item_Weight'].mean(),inplace=True)
```

- The mode used for the categorical dataset

```
df_train['Outlet_Size'].fillna(df_train['Outlet_Size'].mode()[0],inplace=True)
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0],inplace=True)
```

5.2 REMOVING UNWANTED ATTRIBUTES

Those attributes which will not involve in the prediction can be removed.

```
df_train.drop(['Item_Identifier','Outlet_Identifier'],axis=1, inplace=True)
df_test.drop(['Item_Identifier','Outlet_Identifier'],axis=1, inplace=True)
```

5.3 DATA VISUALIZATION

The Seaborn Python library is utilized to effectively visualize data by generating figures with code that showcase all the features, that is shown in all Figures.

```
sns.countplot(x='Outlet_Location_Type',data=df_train)
plt.show()
```

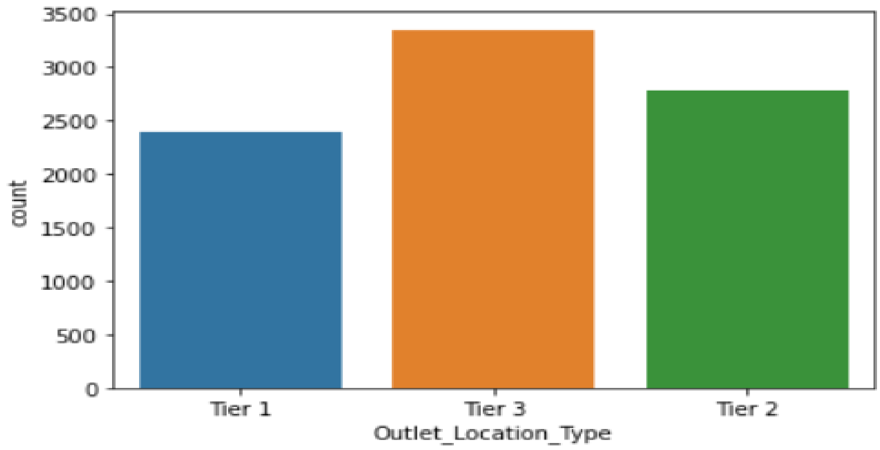


Figure 5.1 Outlet location type

```
df_train['Item_Fat_Content'].replace('low fat','Low Fat',inplace=True)
df_train['Item_Fat_Content'].replace('LF','Low Fat',inplace=True)
df_train['Item_Fat_Content'].replace('reg','Regular',inplace=True)
```

```
sns.countplot(x='Item_Fat_Content',data=df_train)
plt.show()
```

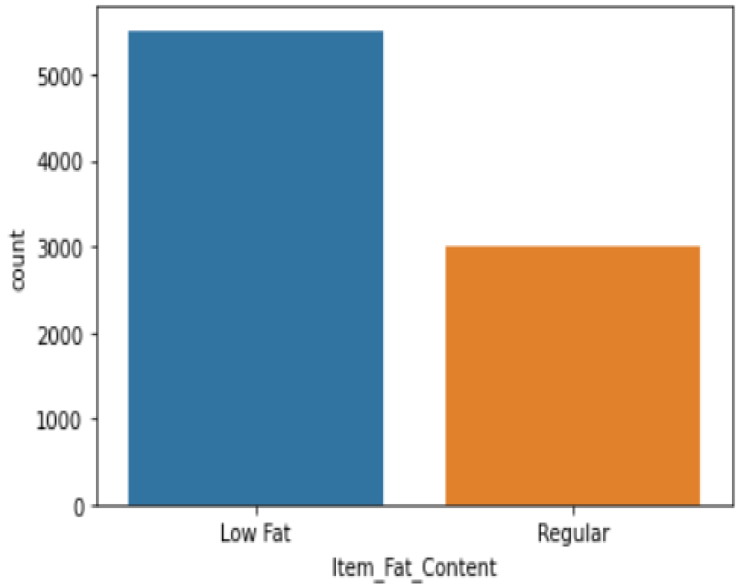


Figure 5.2 Item Fat Content

```python
plt.figure(figsize=(12,6))
sns.countplot(x='Outlet_Type', data = df_train)
plt.show()
```

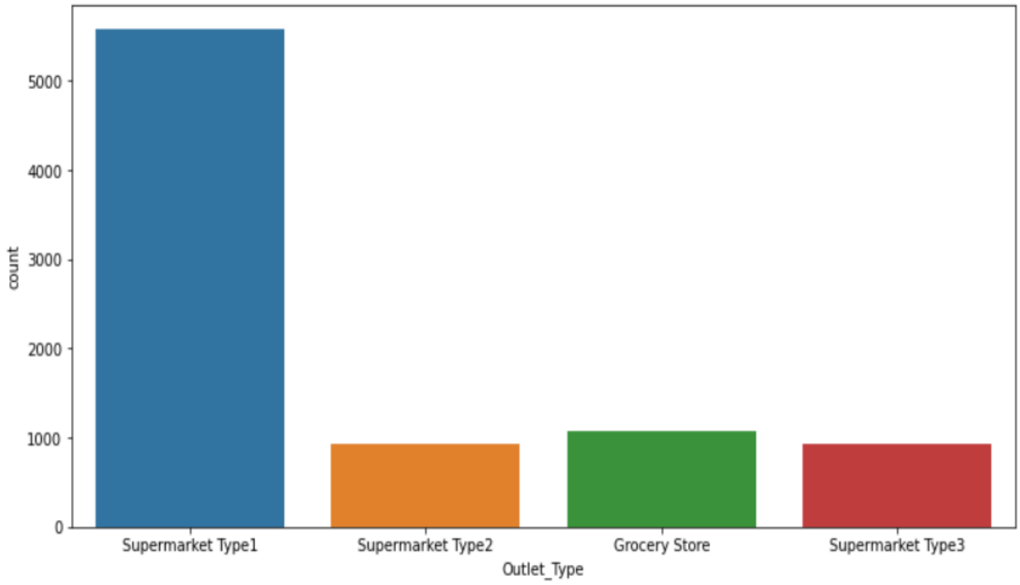


Figure 5.3 Outlet Type

```python
df_train['Item_Fat_Content'].replace('low fat','Low Fat',inplace=True)
df_train['Item_Fat_Content'].replace('LF','Low Fat',inplace=True)
df_train['Item_Fat_Content'].replace('reg','Regular',inplace=True)
```

```python
sns.countplot(x='Item_Fat_Content',data=df_train)
plt.show()
```

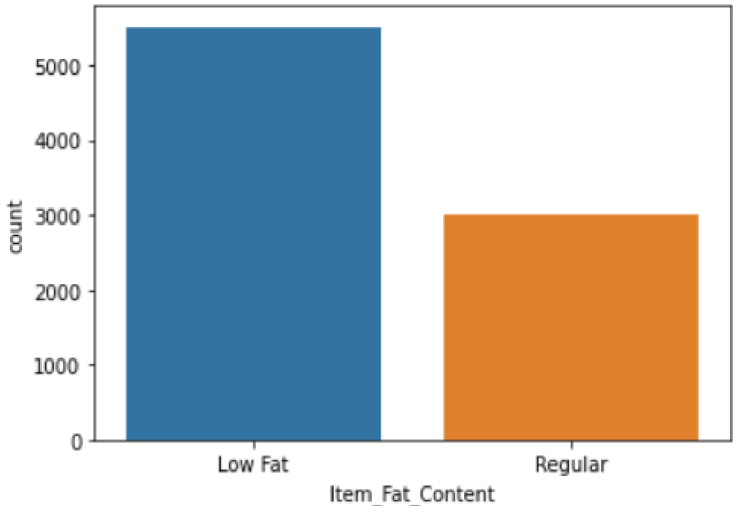


Fig.5.4. Item Fat (low and medium fat)

```
sns.countplot(x='Item_Fat_Content',data=df_train)
plt.show()
```

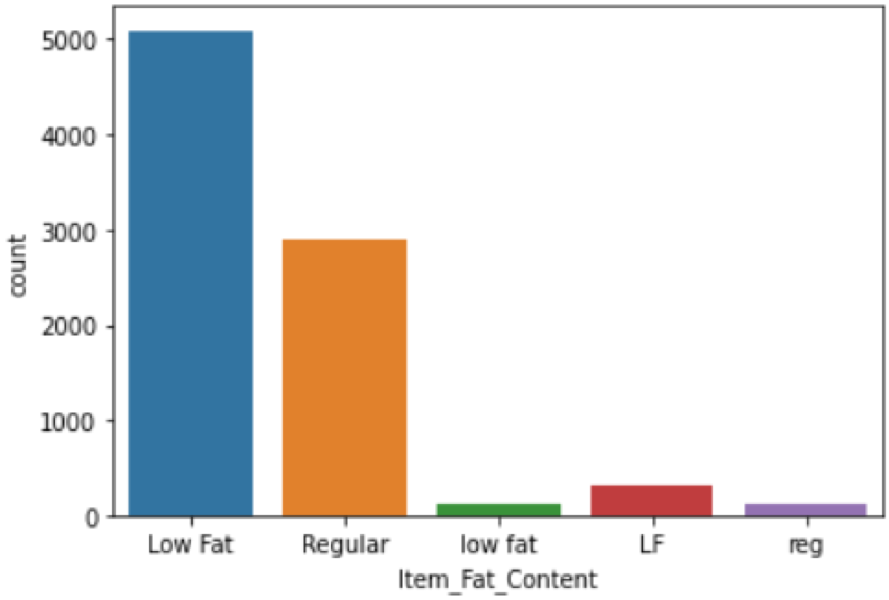


Fig.5.5. Item Fat (Different Fat Category)

```
sns.countplot(x='Outlet_Size',data=df_train)
plt.show()
```

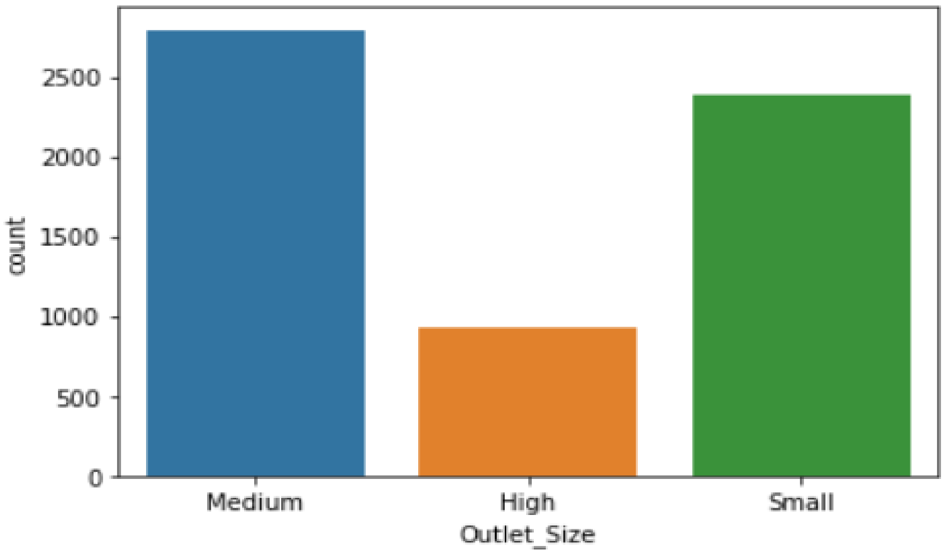


Fig.5.6. Outlet Size Category

```
plt.figure(figsize=(24,6))
sns.countplot(x='Item_Type',data=df_train)
plt.show()
```

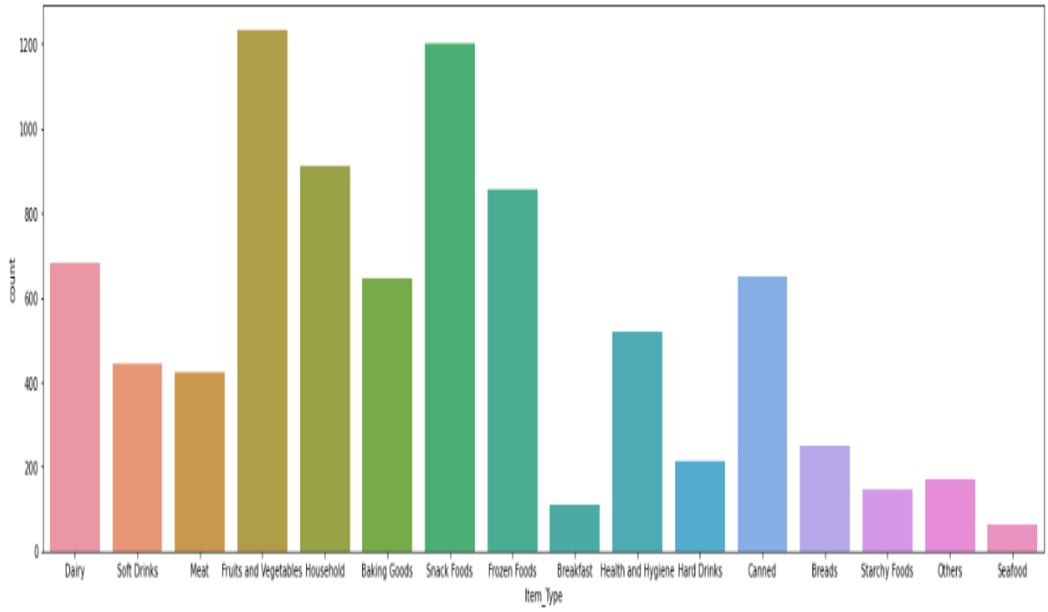


Fig.5.7. Item Type

5.4 EXPLORATORY DATA ANALYSIS

Exploratory data analysis is an essential step in conducting initial assessments of data to detect patterns, anomalies, and to verify assumptions and test hypotheses through the utilization of summary statistics and graphical depictions. By presenting data in a graphical format, such as graphs, data visualization enhances our comprehension of the data, enabling us to interpret it intuitively and detect trends, patterns, and anomalies in extensive datasets. For Exploratory data analysis (EDA) used some libraries:

5.4.1 Klib library: Figure 5.8 illustrate the bar graph of categorical data.

```
import klib
```

```
klib.cat_plot(df_train)
```

```
GridSpec(6, 5)
```

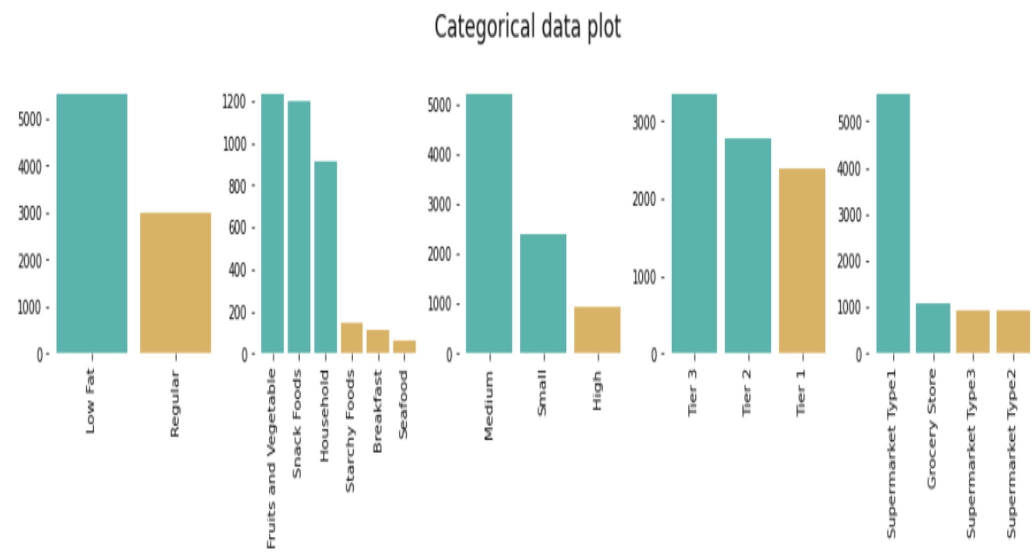
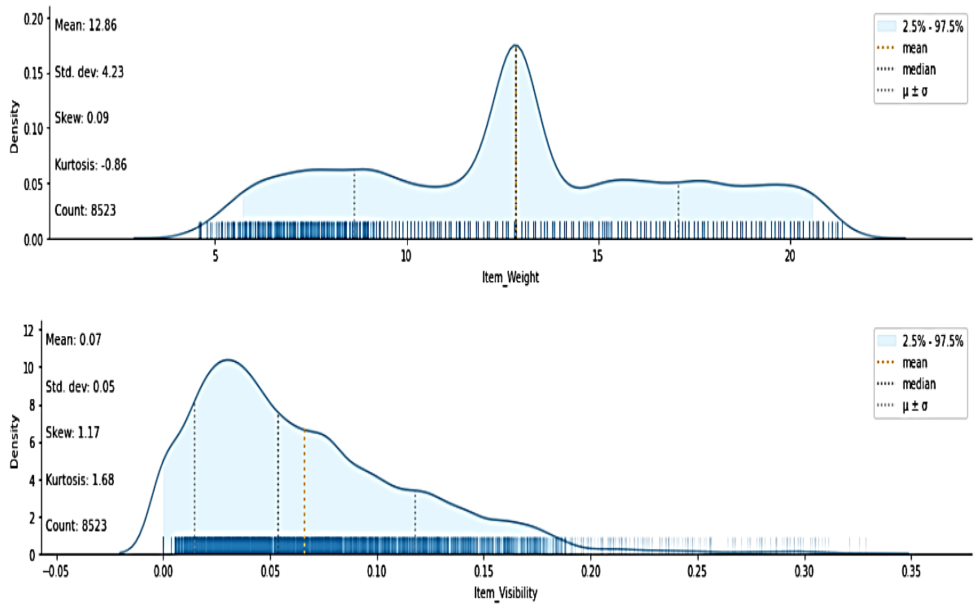


Fig.5.8 Categorical Data Plot

```
klib.dist_plot(df_train)
```

```
<AxesSubplot: xlabel='Item_Outlet_Sales', ylabel='Density'>
```



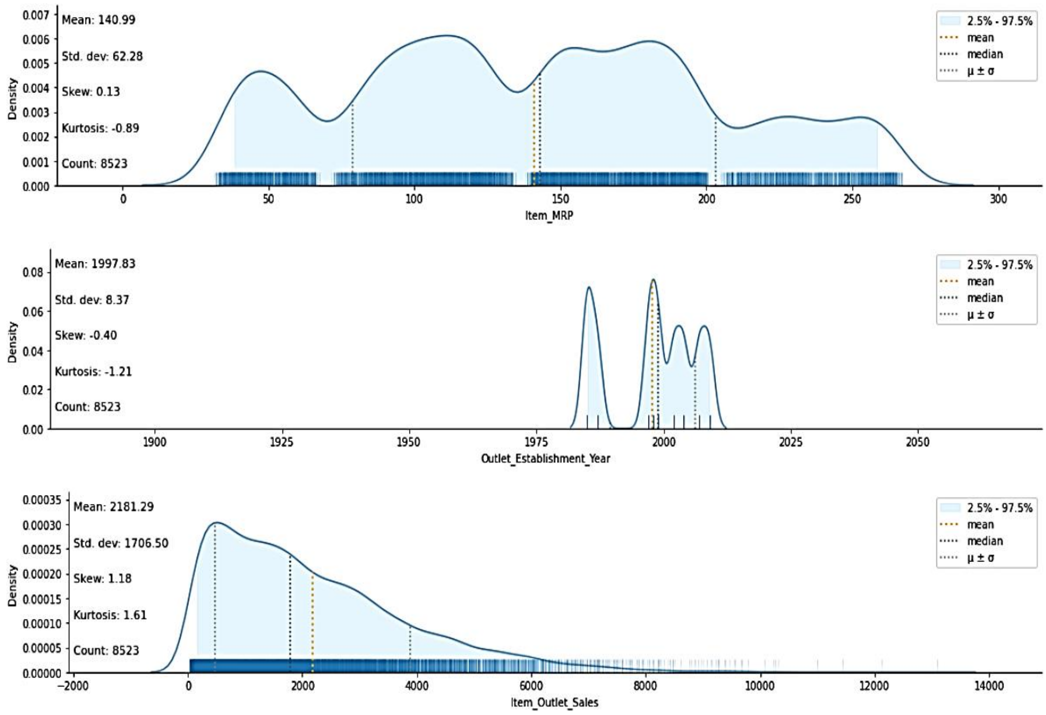


Fig.5.9 Distribution Graph

5.4.2 Correlation Heatmap: A correlation heatmap is a graphical representation of the correlation between different variables in a dataset. Figure 5.10 is a matrix that uses color-coded cells to visualize the correlation coefficients between pairs of variables. Each variable is represented by a row and column in the heatmap, and the correlation coefficient between the variables is indicated by the color of the cell at the intersection of the row and column. The color scheme typically ranges from red (indicating a strong positive correlation) to blue (indicating a strong negative correlation), with white or neutral colours representing no correlation.

The creation of a heatmap involves computing the correlation coefficient for each variable pair within the dataset. The correlation coefficient can range from -1 to +1, with -1 signifying a fully negative correlation, +1 indicating a complete positive correlation, and 0 indicating no correlation.

The resulting matrix of correlation coefficients is then plotted as a heatmap, with each cell colored according to the value of the correlation coefficient.

Typically, warm colours such as red are used to indicate positive correlations, while cool colours such as blue are used to indicate negative correlations.

```
plt.figure(figsize=(10,5))
sns.heatmap(df_train.corr(),annot=True)
plt.show()
```

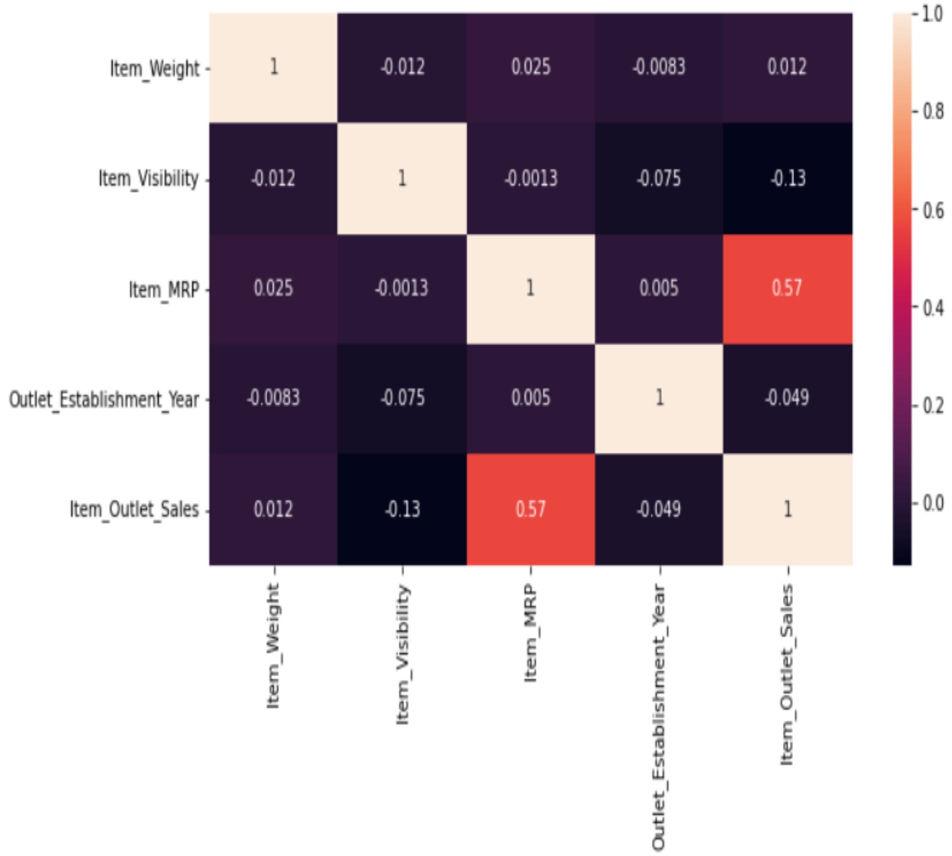
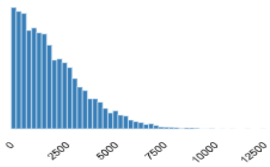


Fig.5.10. Correlation Heatmap

5.4.3 Pandas Profiling: It is showing statistics of all dataset in below fig. Figure

Item_Outlet_Sales
Real number (ℝ)

Distinct	3493	**Minimum**	33.29
Distinct (%)	41.0%	**Maximum**	13086.965
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	2181.2889	**Memory size**	66.7 KiB



5.11. Item Outlet Sales Statistics

Outlet_Size
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	66.7 KiB

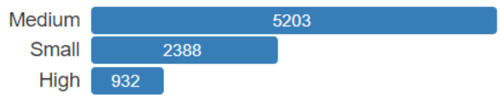


Figure 5.12. Outlet size Statistics

Outlet_Establishment_Year
Real number (ℝ)

Distinct	9	**Minimum**	1985
Distinct (%)	0.1%	**Maximum**	2009
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	1997.8319	**Memory size**	66.7 KiB

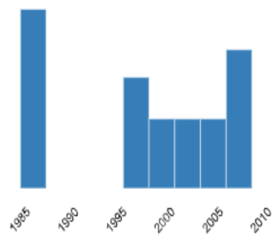


Figure 5.13. Outlet Establishment Year Statistics

Item_MRP

Real number (ℝ)

Distinct	5938	**Minimum**	31.29
Distinct (%)	69.7%	**Maximum**	266.8884
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	140.99278	**Memory size**	66.7 KiB

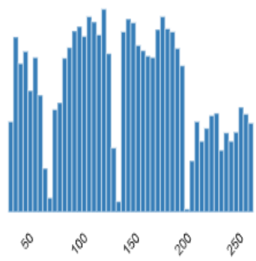


Figure 5.14. Item MRP Statistics

Item_Fat_Content

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	66.7 KiB

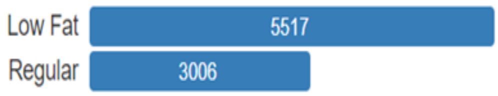
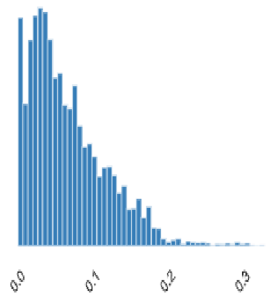


Figure 5.15. Item Fat Content Statistics

Item_Visibility

Real number (ℝ)

Distinct	7880	Minimum	0
Distinct (%)	92.5%	Maximum	0.32839095
Missing	0	Zeros	526
Missing (%)	0.0%	Zeros (%)	6.2%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.066132029	Memory size	66.7 KiB



More details

Item_Type

Categorical

Distinct	16
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	66.7 KiB

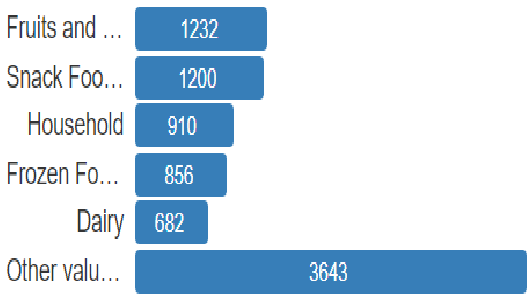


Figure 5.16 Item Visbility

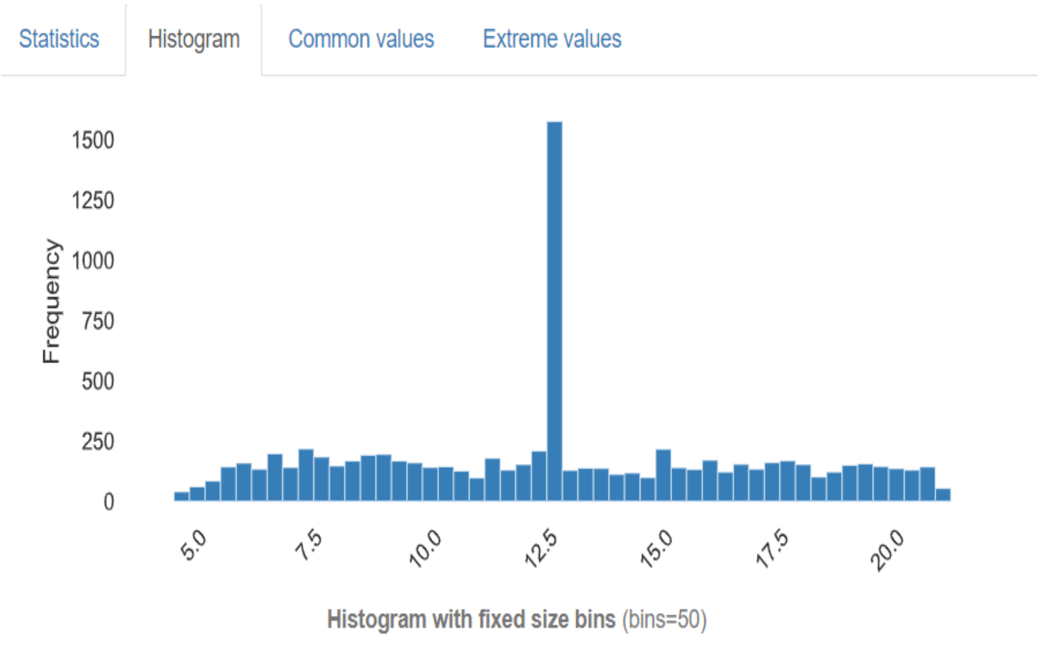


figure 5.17. For Whole Dataset Statistics Data

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	4.555
5-th percentile	6.13
Q1	9.31
median	12.857645
Q3	16
95-th percentile	20.19
Maximum	21.35
Range	16.795
Interquartile range (IQR)	6.69

Descriptive statistics

Standard deviation	4.2261237
Coefficient of variation (CV)	0.32868567
Kurtosis	-0.86029448
Mean	12.857645
Median Absolute Deviation (MAD)	3.3423548
Skewness	0.090561452
Sum	109585.71
Variance	17.860122
Monotonicity	Not monotonic

Overview

Overview | Alerts 7 | Reproduction

Dataset statistics

Number of variables	10
Number of observations	8523
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	666.0 KiB
Average record size in memory	80.0 B

Variable types

Numeric	5
Categorical	5

figure 5.18. Dataset Statistics

5.5 DATA CLEANING

Cleaning the dataset by making names of attributes from initial capital letters to small letters.

Drop the unwanted features like Item Identifier, Outlet Identifier.

```
df_train.drop(['Item_Identifier','Outlet_Identifier'],axis=1, inplace=True)
df_test.drop(['Item_Identifier','Outlet_Identifier'],axis=1, inplace=True)
```

5.6 LABEL ENCODING

Label encoding involves converting labels into a numerical format that can be understood by machines. This enables machine learning techniques to analyze and interpret these labels. In supervised learning with structured datasets, label

```
# LABEL ENCODING
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
df_train
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supermarket Type1	
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supermarket Type2	
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supermarket Type1	
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	Medium	Tier 3	Grocery Store	
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supermarket Type1	
...	...	...	...	...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supermarket Type1	

```
df_train['item_fat_content']=le.fit_transform(df_train['item_fat_content'])
df_train['item_type']=le.fit_transform(df_train['item_type'])
df_train['outlet_size']=le.fit_transform(df_train['outlet_size'])
df_train['outlet_location_type']=le.fit_transform(df_train['outlet_location_type'])
df_train['outlet_type']=le.fit_transform(df_train['outlet_type'])
```

encoding is an essential pre-processing step.

5.7 SPLITTING THE TRAINING AND TESTING DATA

Label encoding involves converting labels into a numerical format that can be understood by machines. This enables machine learning techniques to analyze and interpret these labels. In supervised learning with structured datasets, label encoding is an essential pre-processing step.

```
# spliting our data into train and test
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   item_weight                8523 non-null   float32
 1   item_fat_content           8523 non-null   int32
 2   item_visibility            8523 non-null   float32
 3   item_type                  8523 non-null   int32
 4   item_mrp                   8523 non-null   float32
 5   outlet_establishment_year  8523 non-null   int16
 6   outlet_size                8523 non-null   int32
 7   outlet_location_type       8523 non-null   int32
 8   outlet_type                8523 non-null   int32
 9   item_outlet_sales          8523 non-null   float32
dtypes: float32(4), int16(1), int32(5)
memory usage: 316.4 KB
```

```
x=df_train.drop('item_outlet_sales', axis=1)
```

```
y=df_train['item_outlet_sales']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, random_state=101, test_size=0.2)
```

5.8 STANDARDIZATION

Standardization is the process of rescaling data values between 0 and 1 by using the mean and standard deviation as a reference point. By doing this, the distribution of the data is adjusted to fit within a standard range. fit within a standard range.

```
# standardization
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

```
x_train_std = sc.fit_transform(x_train)
```

```
x_test_std= sc.transform(x_test)
```

```
x_train_std
```

```
array([[ 1.52290023, -0.74155088,  0.68469731, ..., -1.95699503,
         1.08786619, -0.25964107],
       [-1.239856  , -0.74155088, -0.09514746, ..., -0.28872895,
        -0.13870429, -0.25964107],
       [ 1.54667619,  1.34852514, -0.0083859 , ..., -0.28872895,
        -0.13870429, -0.25964107],
       ...,
       [-0.08197109, -0.74155088, -0.91916229, ...,  1.37953713,
        -1.36527477, -0.25964107],
       [-0.74888436,  1.34852514,  1.21363045, ..., -0.28872895,
        -0.13870429, -0.25964107],
       [ 0.67885675, -0.74155088,  1.83915361, ..., -0.28872895,
         1.08786619,  0.98524841]])
```

CHAPTER 6

EVALUATION METRICS

Evaluating the machine-learning model is a crucial step toward its success. This involves creating the model and analyzing its metrics until a high level of accuracy is achieved. To achieve this, evaluation metrics are used to describe the output of the model. The ability of these metrics to distinguish between various outputs is important. In this study, the Root Mean Squared Error (RMSE) metric was used for assessment. Additionally, R-squared was utilized as a statistical measure of how well the regression model fits the data, with a desired value of 1 indicating a better fit. Hence, improving the R-squared value towards 1 helps enhance the model's accuracy. some commonly used evaluation metrics for regression:

1. Mean squared error (MSE): measures the average squared difference between the predicted and actual values.

2. Root mean squared error (RMSE): the square root of the MSE.

3. Mean absolute error (MAE): measures the average absolute difference between the predicted and actual values.

4. R-squared (R2): measures the proportion of variance in the dependent variable that is explained by the independent variables.

5. Mean squared logarithmic error (MSLE): measures the average squared logarithmic difference between the predicted and actual values.

6. Explained variance score: measures the proportion of variance in the dependent variable that is explained by the model.

CHAPTER 7

MODEL BUILDING

Once the data preprocessing is complete, the dataset can be utilized to develop a predictive model. The algorithm is fed with the training data to train it to forecast the values. The model creates a target variable for prediction before being provided input from the testing data. There are various models that can be employed to build the predictive model.

A) Linear Regression: code and output of the dataset are shown below:

```
# linear regression
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train_std, y_train)
```

```
LinearRegression()
```

```
y_pred_lr=lr.predict(x_test_std)
```

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

```
print(r2_score(y_test,y_pred_lr))
print(mean_absolute_error(y_test,y_pred_lr))
print(np.sqrt(mean_squared_error(y_test,y_pred_lr)))
```

```
0.5040717498620855
880.9630372790444
1162.5770350328762
```

B) Lasso Regression- code and output of the dataset are shown below:

```
#Lasso
```

```
from sklearn.linear_model import Lasso
la= Lasso(alpha=1.0)
la.fit(x_train_std, y_train)
```

```
Lasso()
```

```
y_pred_la=la.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_la))
print(mean_absolute_error(y_test,y_pred_la))
print(np.sqrt(mean_squared_error(y_test,y_pred_la)))
```

```
0.5041279019642897
880.7557434185359
1162.5112160434733
```

C) Random Forest: code and output of the dataset are shown below

```
# random forest
```

```
from sklearn.ensemble import RandomForestRegressor
rf= RandomForestRegressor(n_estimators=1000)
rf.fit(x_train_std, y_train)
```

```
RandomForestRegressor(n_estimators=1000)
```

```
y_pred_rf=rf.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_rf))
print(mean_absolute_error(y_test,y_pred_rf))
print(np.sqrt(mean_squared_error(y_test,y_pred_rf)))
```

```
0.546573571346206
783.2672383717444
1111.6440064514652
```

D) Xgboost Regression: code and output of the dataset are shown below

```
#!pip install xgboost
```

```
from xgboost import XGBRegressor
model = XGBRegressor()
model.fit(x_train_std, y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=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,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
y_pred_model=model.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_model))
print(mean_absolute_error(y_test,y_pred_model))
print(np.sqrt(mean_squared_error(y_test,y_pred_model)))
```

```
0.5300847410940401
802.49927
1131.676
```

E) Adaboost Regression: code and output of the dataset are shown below

```
from sklearn.ensemble import AdaBoostRegressor
model = AdaBoostRegressor()
model.fit(x_train_std, y_train)
```

```
AdaBoostRegressor()
```

```
y_pred_model=model.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_model))
print(mean_absolute_error(y_test,y_pred_model))
print(np.sqrt(mean_squared_error(y_test,y_pred_model)))
```

```
0.46702625830119326
952.3147553000482
1205.2169259327748
```

F) K -Nearest Neighbors (KNN): code and output of the dataset are shown below

```
from sklearn.neighbors import KNeighborsRegressor
KNN = KNeighborsRegressor()
KNN.fit(x_train_std, y_train )
```

```
KNeighborsRegressor()
```

```
y_pred_KNN=KNN.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_KNN))
print(mean_absolute_error(y_test,y_pred_KNN))
print(np.sqrt(mean_squared_error(y_test,y_pred_KNN)))
```

```
0.5055493284555501
820.1612
1160.8439
```

G) Ridge Regression: code and output of the dataset are shown below

```
from sklearn.linear_model import Ridge
ri= Ridge(alpha=1.0)
ri.fit(x_train_std, y_train)
```

```
Ridge()
```

```
y_pred_ri=ri.predict(x_test_std)
```

```
print(r2_score(y_test,y_pred_ri))
print(mean_absolute_error(y_test,y_pred_ri))
print(np.sqrt(mean_squared_error(y_test,y_pred_ri)))
```

```
0.5040741163044216
880.9484305410501
1162.5742612699419
```

H) Artifical Neural Network(ANN): code and output of the dataset are shown below

```
#pip install tensorflow
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer
```

```
model = Sequential(
    [
        InputLayer(input_shape=(9,)),
        Dense(50, activation='relu'),
        Dense(10, activation='relu'),
        Dense(5, activation='relu'),
        Dense(2, activation='relu'),
        Dense(1, activation='linear')
    ]
)
```

```
model.compile(loss='mse',optimizer='adam',metrics=['mse','mae'])
model_history= model.fit(x_train_std,y_train,validation_data=(x_test_std,y_test),epochs=100)
```

```
Epoch 94/100
214/214 [==============================] - 1s 3ms/step - loss: 1169172.3750 - mse: 1169172.3750 - mae: 762.6365 - val_los
s: 1101601.1250 - val_mse: 1101601.1250 - val_mae: 744.8754
Epoch 95/100
214/214 [==============================] - 1s 3ms/step - loss: 1168762.6250 - mse: 1168762.6250 - mae: 761.6547 - val_los
s: 1105973.5000 - val_mse: 1105973.5000 - val_mae: 750.4374
Epoch 96/100
214/214 [==============================] - 1s 4ms/step - loss: 1168550.6250 - mse: 1168550.6250 - mae: 761.2982 - val_los
s: 1103067.6250 - val_mse: 1103067.6250 - val_mae: 747.3640
Epoch 97/100
214/214 [==============================] - 1s 4ms/step - loss: 1171173.0000 - mse: 1171173.0000 - mae: 762.5765 - val_los
s: 1110726.6250 - val_mse: 1110726.6250 - val_mae: 753.4868
Epoch 98/100
214/214 [==============================] - 1s 3ms/step - loss: 1166674.8750 - mse: 1166674.8750 - mae: 761.1324 - val_los
s: 1111453.1250 - val_mse: 1111453.1250 - val_mae: 754.1006
Epoch 99/100
214/214 [==============================] - 1s 3ms/step - loss: 1165607.0000 - mse: 1165607.0000 - mae: 759.1884 - val_los
```

```
model_predictions=model.predict(x_test_std)
```

```
54/54 [==============================] - 1s 4ms/step
```

```
model_predictions=model.predict(x_test_std)
```

```
54/54 [==============================] - 0s 3ms/step
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
print(mean_absolute_error(y_test,model_predictions))
print(mean_squared_error(y_test,model_predictions))
```

```
745.3458
1101562.0
```

```
model_predictions_train=model.predict(x_train_std)
print("X_train_std e2 score: ",r2_score(y_train,model_predictions_train))
```

```
214/214 [==============================] - 1s 2ms/step
X_train_std e2 score:  0.6067968191336988
```

```
model_predictions_test=model.predict(x_test_std)
print("X_test_std e2 score: ",r2_score(y_test,model_predictions_test))
```

```
54/54 [==============================] - 0s 3ms/step
X_test_std e2 score:  0.5958111916444011
```

CHAPTER 8

RESULT

The below table shows R^2 (%), mean absolute error, root mean squared error(RMSE) values in decreasing order-

A) Machine Learning Algorithm Result:

Table 8.1 Machine Learning Algorithm Result

S.N	MODEL	R^2(%)	MEAN ABSOLUTE ERROR	RMSE
1	Random Forest	54.81	782.356	1109.743
2	Xgboost	53.00	802.499	1131.676
3	KNN	50.55	820.16	1160.84
4	Lasso	50.41	880.75	1162.51
5	Linear Regression	50.40	880.96	1162.577
6	Ridge	50.40	880.94	1162.57
7	AdaBoost	49.95	901.15	1167.88

B) Artificial Neural Network Result (ANN):

Table 8.2 Artificial Neural Network Result

1	ANN	59.98	745.34	1049.55

CHAPTER 9

CONCLUSION

In today's highly competitive business landscape, accurate and reliable sales projections can make all the difference between success and failure. Random forest regressor and artificial neural networks (ANN) are two powerful machine learning techniques that have been shown to be highly effective in predicting sales and profitability. By leveraging the power of these advanced algorithms, businesses can gain valuable insights into market trends and consumer behavior, enabling them to make better decisions and stay ahead of the competition.

The benefits of accurate sales projections are clear. By understanding their sales and profitability, companies can optimize their methods and tactics, improve their product offerings, and develop effective marketing strategies that resonate with their target audience. This, in turn, can boost profitability, increase customer loyalty, and inspire the creation of more stores or branches, further expanding their reach and influence.

In conclusion, the combination of random forest regressor and ANN can provide businesses with a powerful tool for predicting sales and profitability, enabling them to stay ahead of the curve and outperform their competitors. By leveraging the insights provided by these advanced algorithms, companies can make better decisions, optimize their strategies, and ultimately achieve greater success in today's fast-paced and ever-changing market

REFERENCES

1.Thivakaran, T., K., Ramesh, M.(2022, July 5). Exploratory data analysis and sales forecasting of big mart dataset using supervised and ANN algorithms. Measurement: Sensors.

2. Magablah, B.(2020, August 25). A comparative study on machine learning and deep learning techniques for predicting big mart output sales. A thesis submitted for the degree of master of science in data analytics at Dublin business school.

3. Ali, R., F., Muneer, A., Almaghthawi, A., Alghamdi, A., Fati, S., M.(2023 June). BMSP-ML: big mart sales prediction using different machine learning techniques.IAES international journal of artificial intelligence (IJ-AI).

4. Pamula, R., Jain, P.,K.(2018, September 29). A two-level statistical model for big mart sales prediction.International conference on computing, power, and communication technologies (GUCON).

5. Behera, G., Nain, N.(2019) A comparative study of big mart sales prediction. conference paper 2019.