PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING

Phase 5 Document submission

Team member: Sudharson R

Naan mudhalvan id : au723721205045

INTRODUCTION:

Product demand prediction with machine learning is the process of leveraging advanced algorithms and data analysis techniques to forecast the future demand for specific products. This field has gained significant prominence due to its ability to provide more precise and data-driven insights into consumer behavior and market trends.

MAIN OBJECTIVES:

The main objective of product demand prediction with machine learning is to accurately forecast the future demand for a particular product or set of products. This helps businesses and organizations in several ways.

PROCESS PHASES:

1) Data Collection
2) Data Preprocessing
3) Feature Engineering
4) Visualization
5) Interpretation of Results

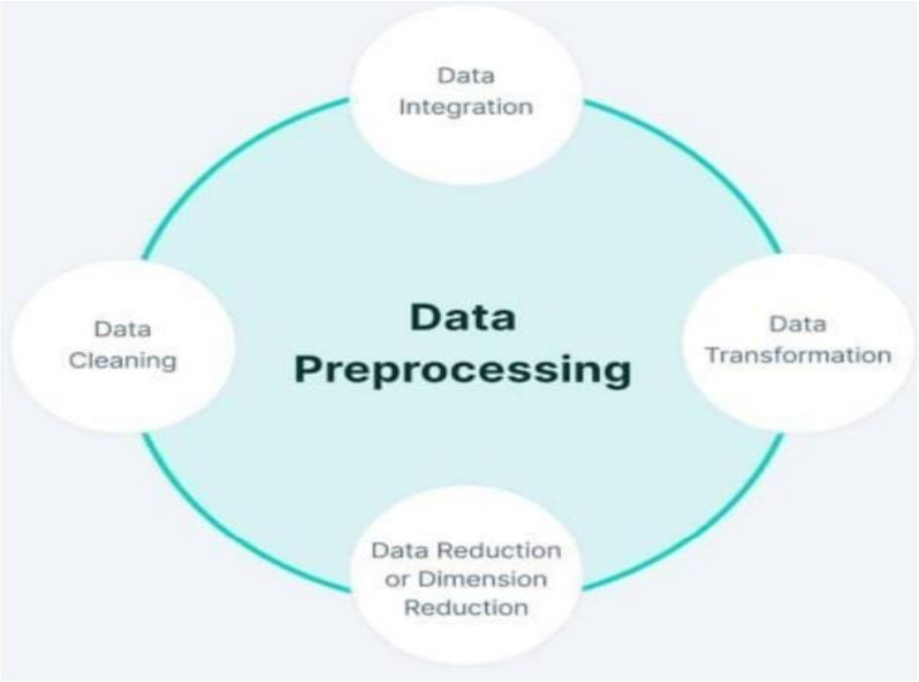
PHASE 1: DATA COLLECTION

- The data collection process in product demand prediction with machine learning is a crucial initial step to build accurate predictive models.
- Collecting the demand of the products should be included here
- Huge volumes of data are needed for the Data analysis process.
- Data collection contains data like store ID, Base price , Total price , Unit sold.

PHASE 2 : DATA PREPROCESSING

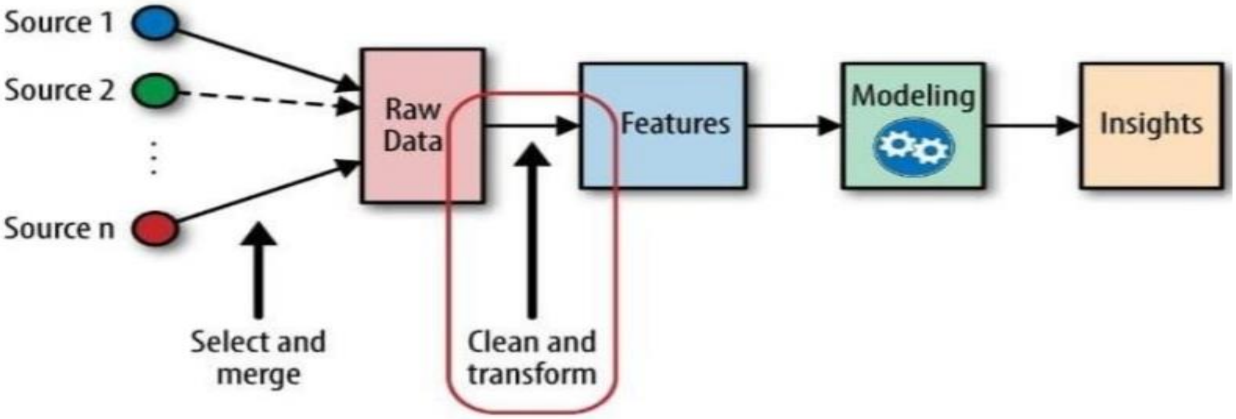
- After collecting the data that should be well prepared and clean

  for analysis.
- Handling the missing values, outliers, and data inconsistencies are should be including here.
- It transform the data through scaling, encoding categorical variables, and feature engineering.
- It integrate the data from different sources.
- Data cleaning, Data integration, Data transformation and Dimension reduction are the important factors in Data Preprocessing.



PHASE 3 : FEATURE ENGINEERING

- Create relevant features that will be used in the machine learning model. Features could include product attributes, time-related features , and variables related to external factors affecting demand.
- It reduce the dimensionality in case of the necessary situation.



PHASE 4 : VISUALIZATION

- Visualization is one of the key concept in data science which can be used for give the pictorial or virtual representation about the data.
- Several plots are used for visualize the customer segments.
- Plots example:
  - Bar Chart
  - Scatter Plot
  - Pie Plot
  - Line Plot
  - Histogram
- In python, Matplotlib library used for the visualization
- Using these charts we can easily visualize the product demand

PHASE 5 : INTERPRETATION OF RESULT

- The goal of this phase is to interpret the product demand and predict according to the data provided.
- It identifies the distinguishing characteristic of each segment.
- Profile each Product based on their Demand such as store ID , base price etc

ACCORDING TO THE DATASET

According to the dataset the product demand prediction contains

- **the product id;**
- **store id;**
- **total price at which product was sold;**
- **base price at which product was sold;**
- **Units sold (quantity demanded);**

The basic idea behind predict is to predict the demand between base price and total price along with unit sold

DATASET: https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

PROGRAM:

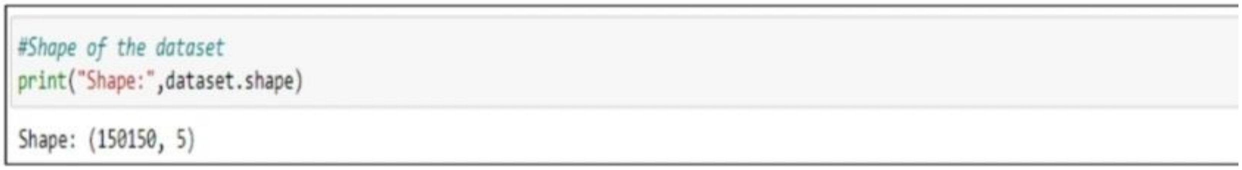
Let us start by importing necessary libraries and the dataset we need for the task of product demand prediction

```
import pandas as pd

import numpy as np

import plotly.express as px

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor
```

LOADING THE DATA:

```
  data = pd.read_csv("c://documents/productdemand.csv")
data.head()
```

Identify the shape of dataset

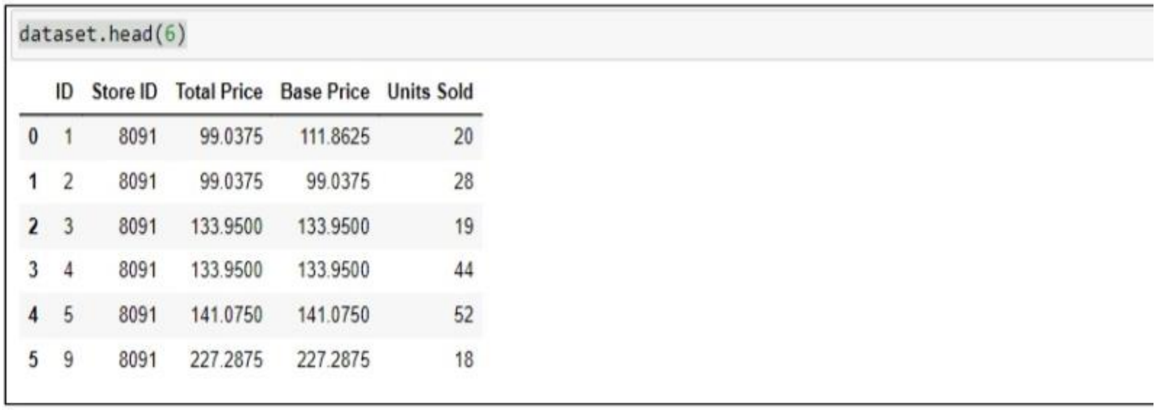


```
#Shape of the dataset
print("Shape:",dataset.shape)

Shape: (150150, 5)
```

The dataset contains **150150 rows** and **5 columns**

To check the head and tail of the dataset use the following

- **dataset.head**(contains the no. of rows)
- **dataset.tail**(contains the no. of rows)

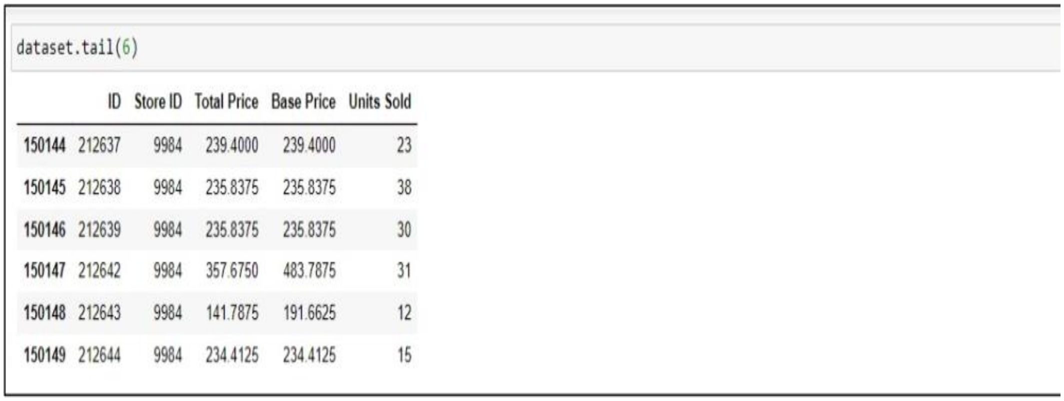


```
dataset.head(6)
```

	ID	Store ID	Total Price	Base Price	Units Sold
0	1	8091	99.0375	111.8625	20
1	2	8091	99.0375	99.0375	28
2	3	8091	133.9500	133.9500	19
3	4	8091	133.9500	133.9500	44
4	5	8091	141.0750	141.0750	52
5	9	8091	227.2875	227.2875	18

This shows the first 6 rows of the dataset.

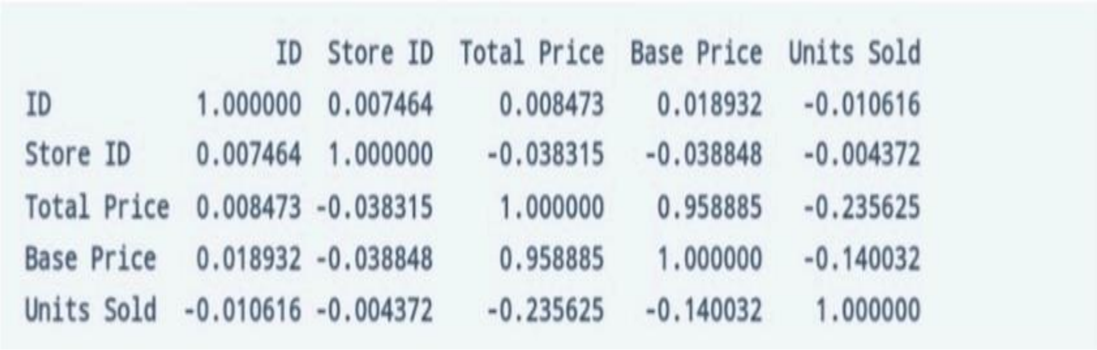
This shows the last 6 rows of the dataset



```
dataset.tail(6)
```

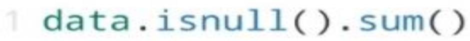
	ID	Store ID	Total Price	Base Price	Units Sold
150144	212637	9984	239.4000	239.4000	23
150145	212638	9984	235.8375	235.8375	38
150146	212639	9984	235.8375	235.8375	30
150147	212642	9984	357.6750	483.7875	31
150148	212643	9984	141.7875	191.6625	12
150149	212644	9984	234.4125	234.4125	15

We have the data **of Product ID , Store ID , Total Price , Base Price , Units Sold** in our dataset.There are **5 columns** and **150150 rows**.

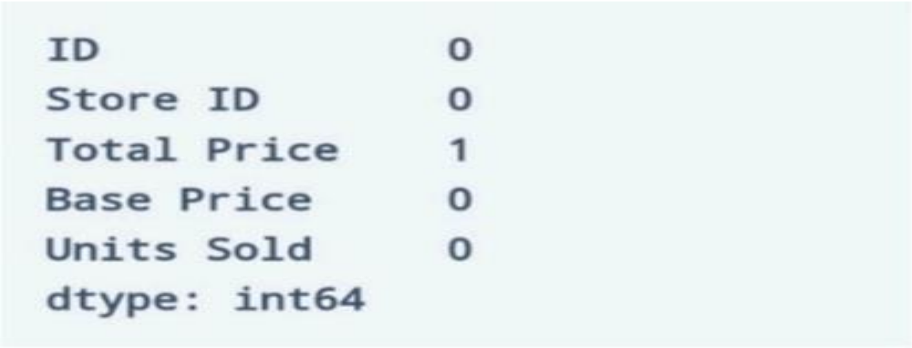


```
                   ID  Store ID  Total Price  Base Price  Units Sold
ID           1.000000  0.007464     0.008473    0.018932   -0.010616
Store ID     0.007464  1.000000    -0.038315   -0.038848   -0.004372
Total Price  0.008473 -0.038315     1.000000    0.958885   -0.235625
Base Price   0.018932 -0.038848     0.958885    1.000000   -0.140032
Units Sold  -0.010616 -0.004372    -0.235625   -0.140032    1.000000
```

Now lets have at whether this dataset contains null value or not

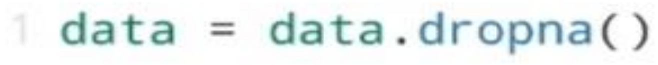


```
data.isnull().sum()
```



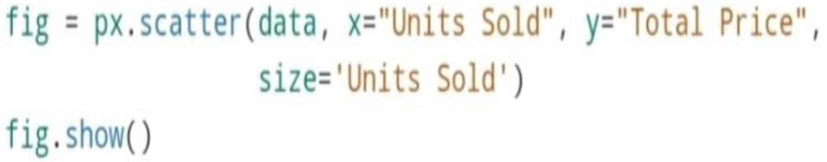
```
ID              0
Store ID        0
Total Price     1
Base Price      0
Units Sold      0
dtype: int64
```

So the dataset has only one missing value in the Total Price column, We remove that entire row for now:

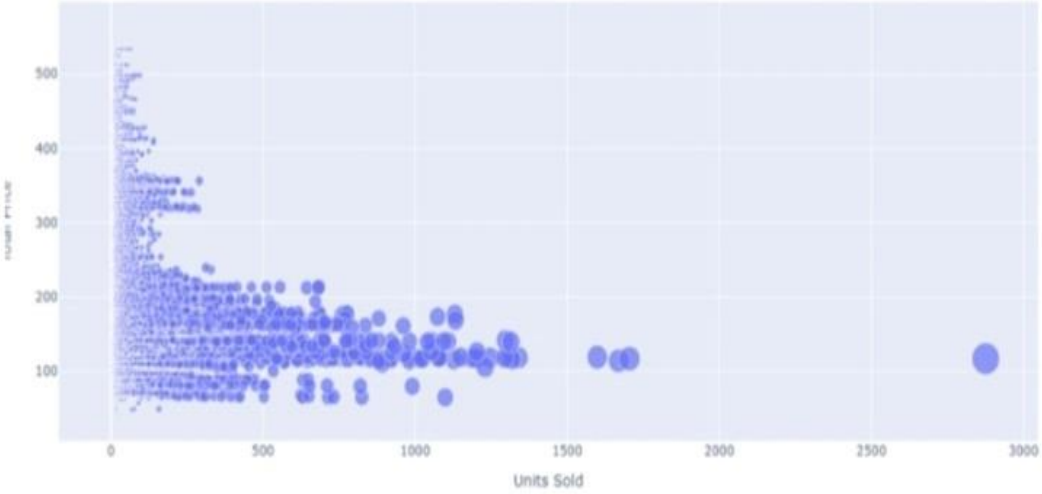


```
data = data.dropna()
```

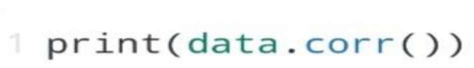
Let us now analyze the relationship between the price and the demand for the product. Here we use a scatter plot to see how the demand for the product varies with the price change



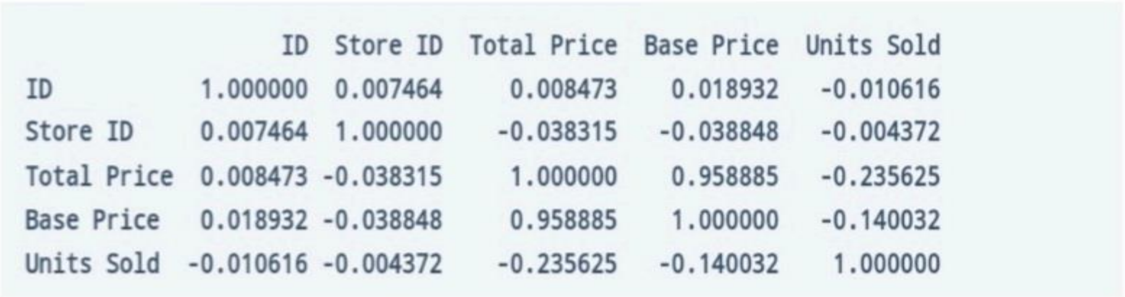
```
fig = px.scatter(data, x="Units Sold", y="Total Price",
                 size='Units Sold')
fig.show()
```



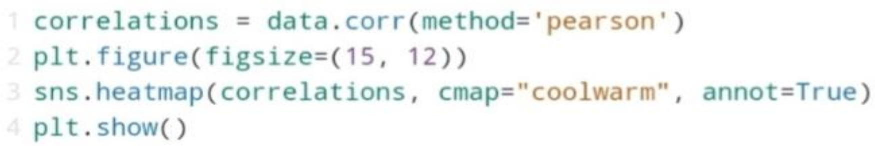
We can see that most of the data points show the sales of the product is increasing as the price is decreasing with some exceptions. Now let’s have a look at the correlation between the features of the dataset:



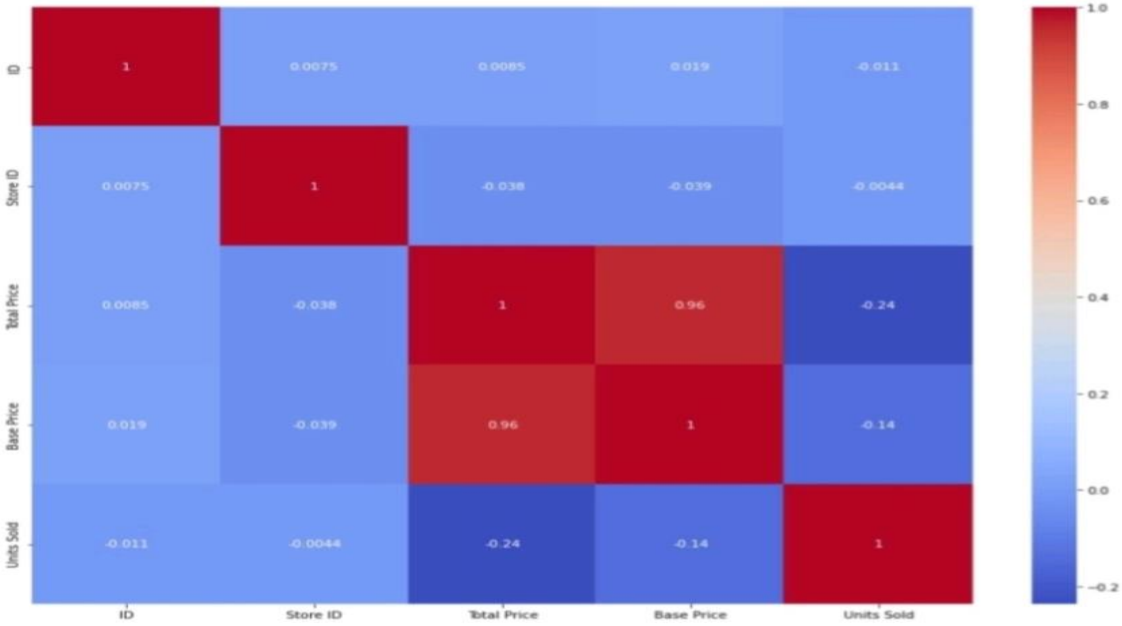
```python
print(data.corr())
```



```
                   ID  Store ID  Total Price  Base Price  Units Sold
ID           1.000000  0.007464     0.008473    0.018932   -0.010616
Store ID     0.007464  1.000000    -0.038315   -0.038848   -0.004372
Total Price  0.008473 -0.038315     1.000000    0.958885   -0.235625
Base Price   0.018932 -0.038848     0.958885    1.000000   -0.140032
Units Sold  -0.010616 -0.004372    -0.235625   -0.140032    1.000000
```

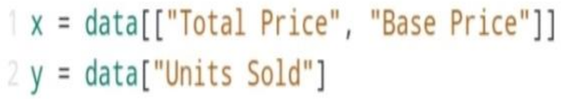


```python
correlations = data.corr(method='pearson')
plt.figure(figsize=(15, 12))
sns.heatmap(correlations, cmap="coolwarm", annot=True)
plt.show()
```



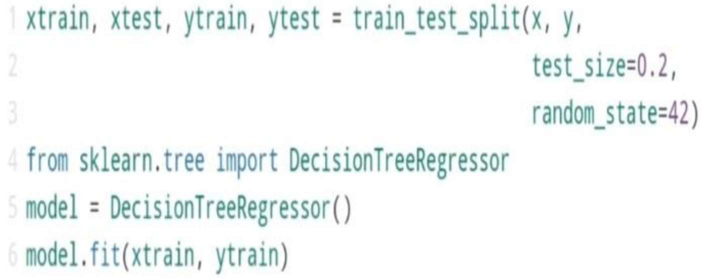
Now let’s move to the task of training a machine learning model to predict the demand for the product at different prices. We choose the Total Price and the Base

Price column as the features to train the model, and the Units Sold column as labels for the model:



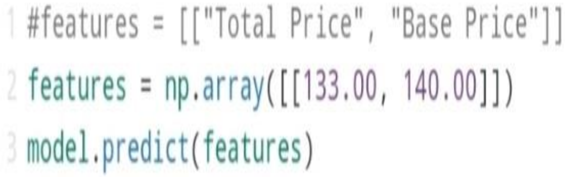
```python
x = data[["Total Price", "Base Price"]]
y = data["Units Sold"]
```

Now let’s split the data into training and test sets and use the decision tree regression algorithm to train our model

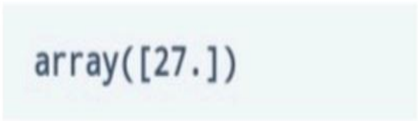


```python
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
```

Now let’s split the data into training and test sets and use the decision tree regression algorithm to train our model



```python
#features = [["Total Price", "Base Price"]]
features = np.array([[133.00, 140.00]])
model.predict(features)
```



```
array([27.])
```

DATA PREPROCESSING

IMPORT LIBRARY : Import the required libraries such as **PANDAS , NUMPY, MATPLOTLIB,** and **Sklearn**

NumPy : It is used for mathematical and numerical functions

PANDAS : It is a analysis tool used for data manipulation

MATPLOTLIB : It is used for data visualization

SKLEARN : To Implement Machine Learning model and Statistical Modeling

Handle Missing Data:

Missing Data can affect the performance of your machine learning models, and bias the conclusions derived from all the statistical analysis on the data.

So, it is necessary to handle the missing data before training our Machine Learning model.

Missing data can be handled in following ways:

- **Deleting rows**

  The row with the missing data can be removed if it have 70 – 75% of missing values

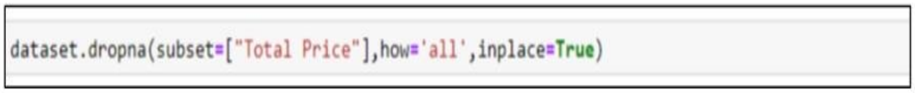
- **Replacing with Mean / Median / Mode**

i) We can calculate the mean, median or mode of the feature and replace it with the missing values

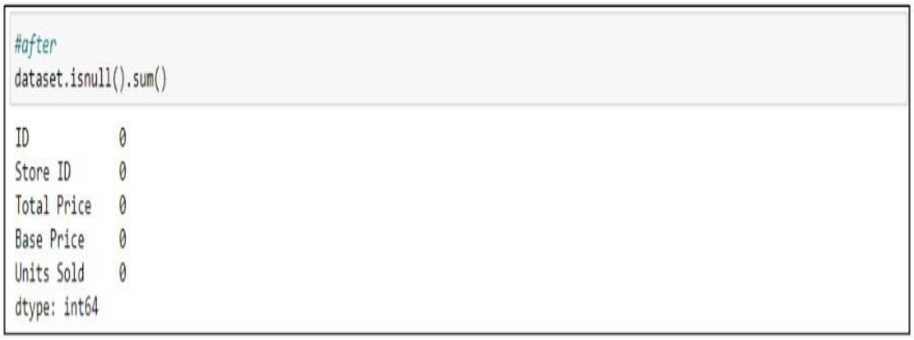
ii) Replacing with the above three approximations are a statistical approach of handling the missing values

It seems to be 1 missing value in 'TOTAL PRICE' in our data set

We have to remove the value by a method called dropna() method



```
dataset.dropna(subset=["Total Price"],how='all',inplace=True)
```



```
#after
dataset.isnull().sum()
```

```
ID             0
Store ID       0
Total Price    0
Base Price     0
Units Sold     0
dtype: int64
```

Now there are no missing value in the dataset.

Splitting the Dataset:

The dataset should be split into Train data and Test data for the following reasons

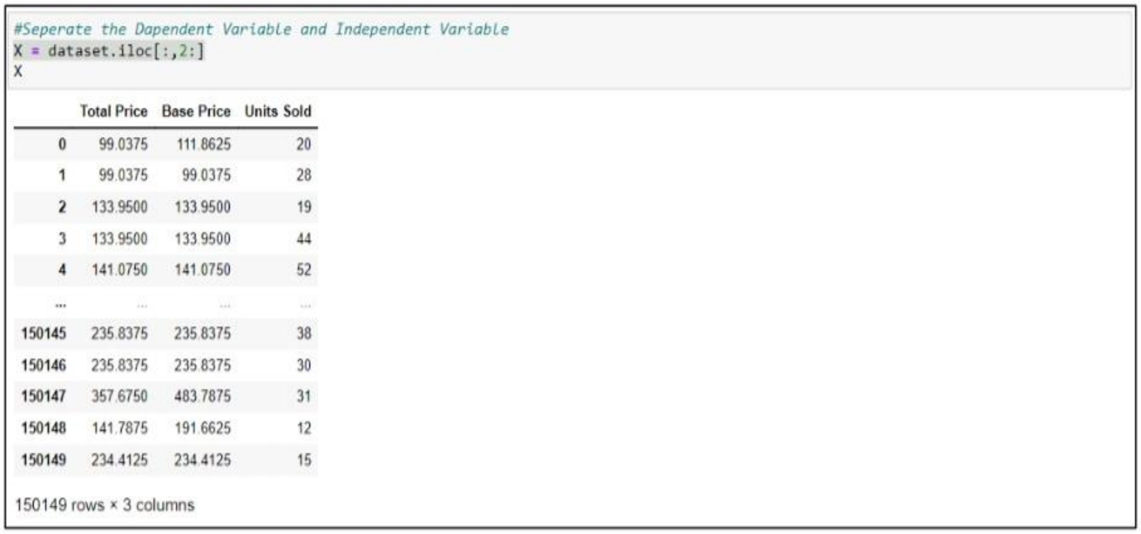
- To estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications
- To measure the accuracy of our model

Before splitting our dataset we should separate the Dependent variable and Independent variable

In our dataset the Dependent variable are '**Total Price**' , '**Base Price**' , '**Units Sold**' and the Independent variable is '**Product ID**'.

Dependent Variable - The dependent variable is the variable about which predictions or explanations are being sought.

Independent Variable - Independent variables (also referred to as Features) are the input for the prediction in our model which is not dependent on other variable.



```
#Seperate the Dapendent Variable and Independent Variable
X = dataset.iloc[:,2:]
X
```

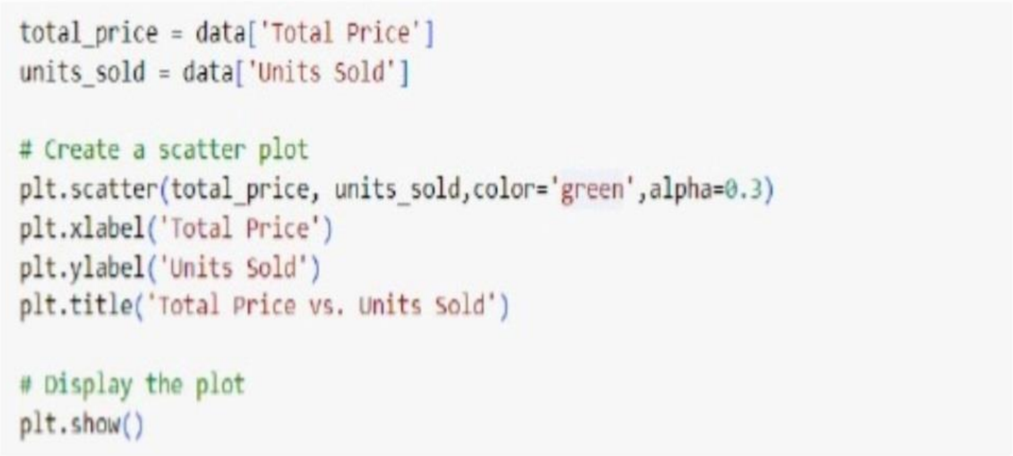
	Total Price	Base Price	Units Sold
0	99.0375	111.8625	20
1	99.0375	99.0375	28
2	133.9500	133.9500	19
3	133.9500	133.9500	44
4	141.0750	141.0750	52
...	...	...	...
150145	235.8375	235.8375	38
150146	235.8375	235.8375	30
150147	357.6750	483.7875	31
150148	141.7875	191.6625	12
150149	234.4125	234.4125	15

150149 rows × 3 columns

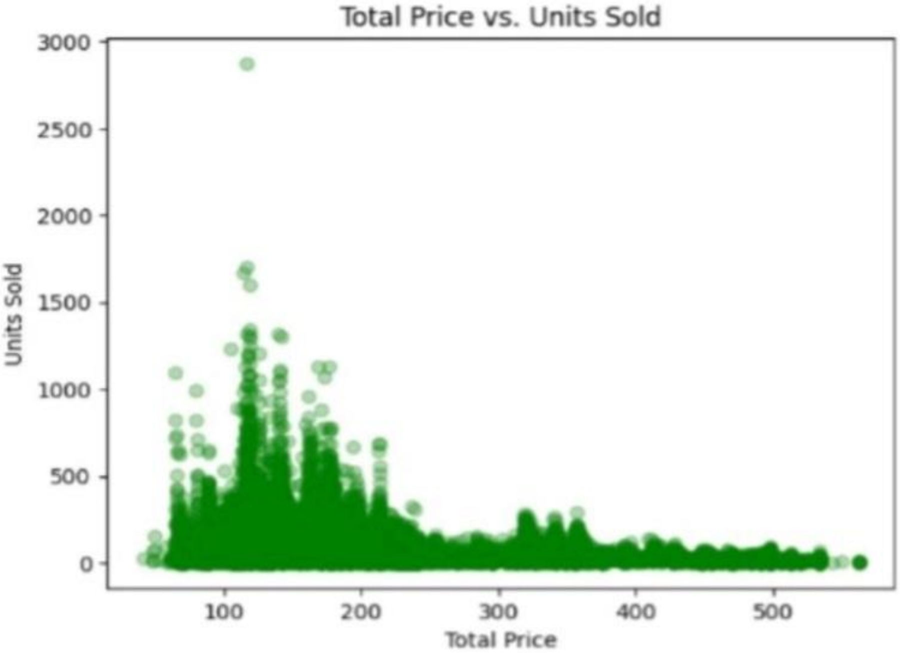
VISUALIZATION

■ Scatterplot Visualization

The follwing contains the visualization of dataset using scatterplot

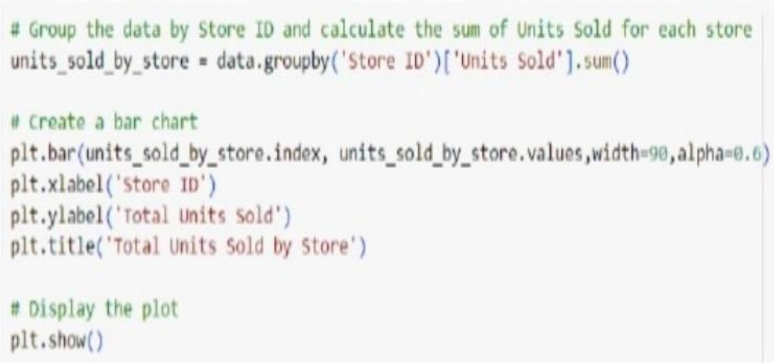


```
total_price = data['Total Price']
units_sold = data['Units Sold']

# Create a scatter plot
plt.scatter(total_price, units_sold,color='green',alpha=0.3)
plt.xlabel('Total Price')
plt.ylabel('Units Sold')
plt.title('Total Price vs. Units Sold')

# Display the plot
plt.show()
```



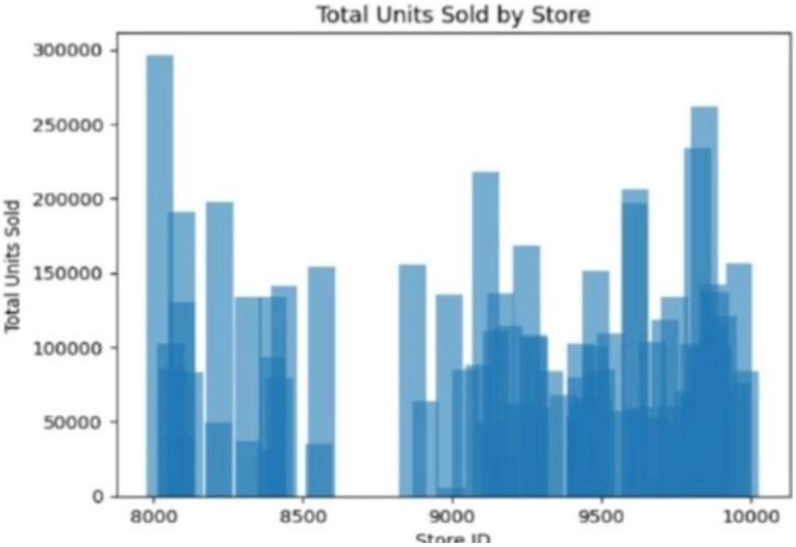
■ Barchat visualization

The following contains the visualizatio of bar chart



```
# Group the data by Store ID and calculate the sum of Units Sold for each store
units_sold_by_store = data.groupby('Store ID')['Units Sold'].sum()

# Create a bar chart
plt.bar(units_sold_by_store.index, units_sold_by_store.values,width=90,alpha=0.6)
plt.xlabel('Store ID')
plt.ylabel('Total Units Sold')
plt.title('Total Units Sold by Store')

# Display the plot
plt.show()
```

The output barchat visualization is

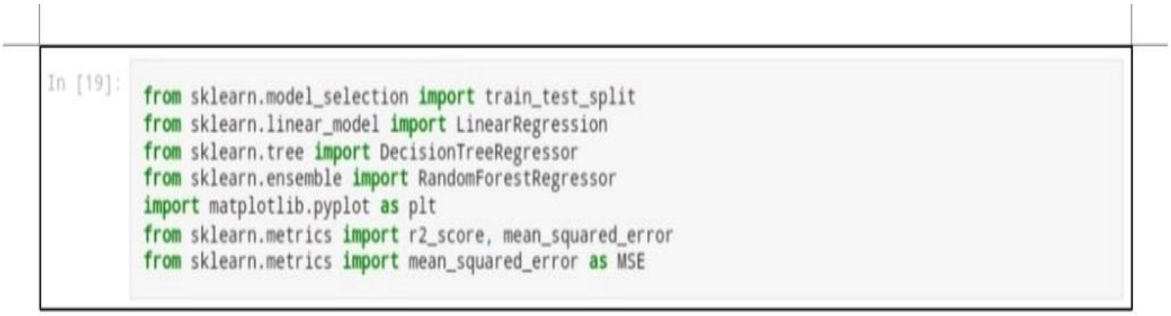


MODEL EVALUATION:

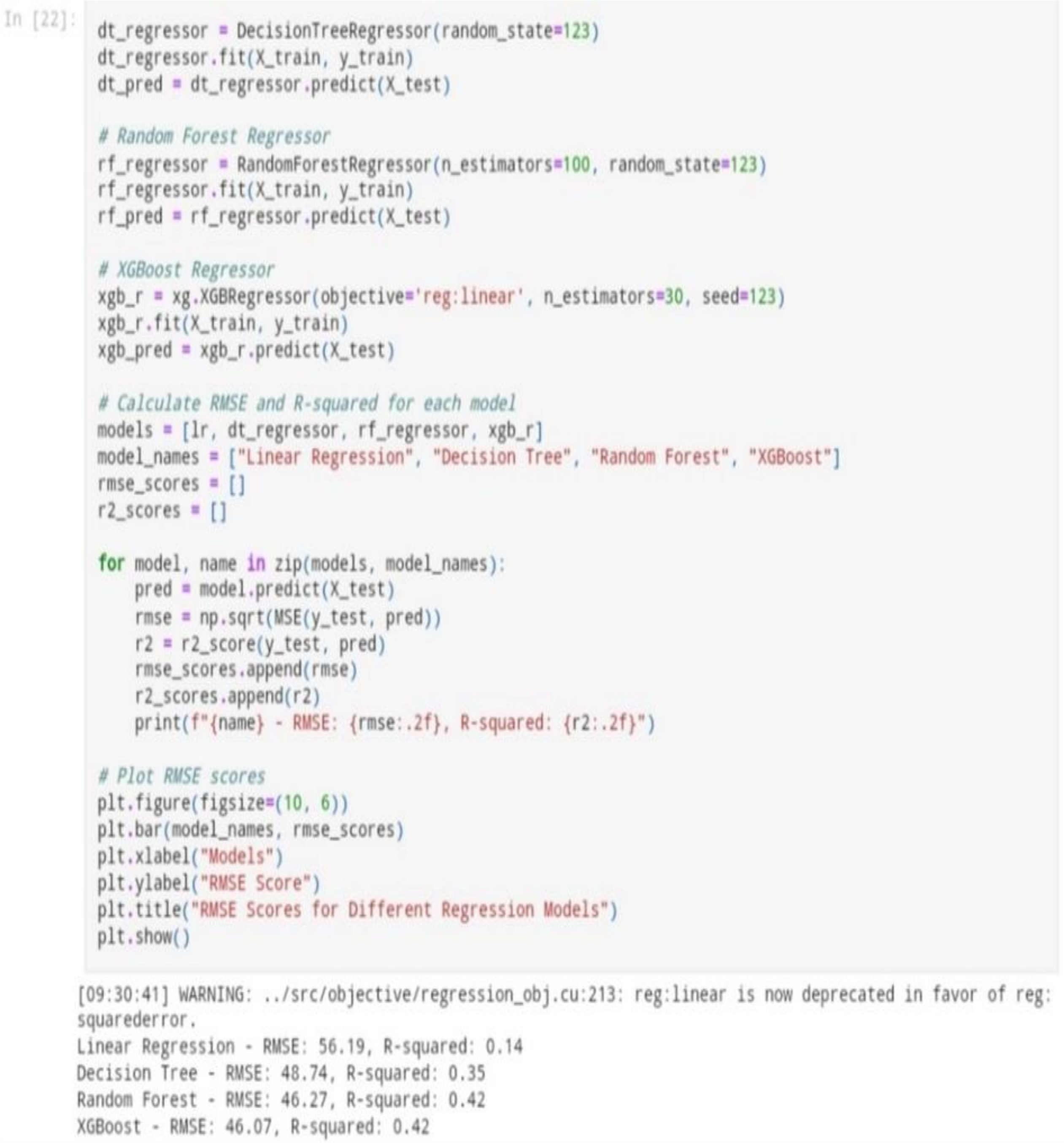
For Model Evaluation we use multiple models to check which one performs well and best in data. Because the linear regression did not perform well enough

Here we use methods like DecisionTree , RandomForest , LinearRegression , XGboost.

- **LinearRegression:**It is used to predict the value of a variable of a variable based on the value of another variable.
- **RandomForest:** It is a commonly used machine learning algorithm used for classification and regression.
- **DecisionTree:** It is a tree-like decision support tool, displaying decisions and their outcomes.
- **XGboost:** It is a machine learning algorithm that uses an ensemble of decision trees and gradient boosting to make predictions.



```
In [19]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
         import matplotlib.pyplot as plt
         from sklearn.metrics import r2_score, mean_squared_error
         from sklearn.metrics import mean_squared_error as MSE
```

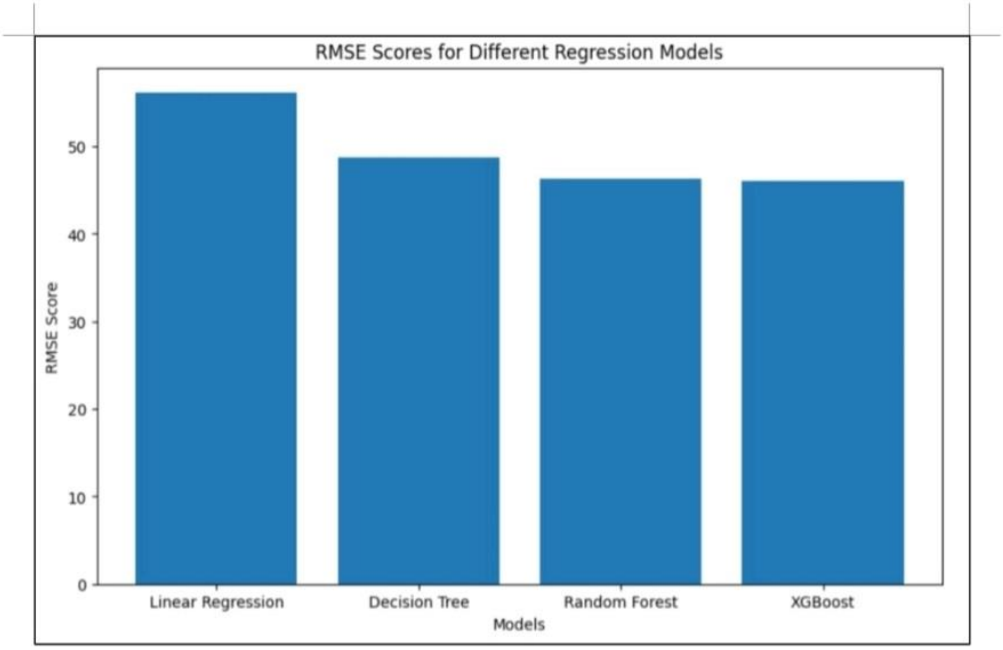


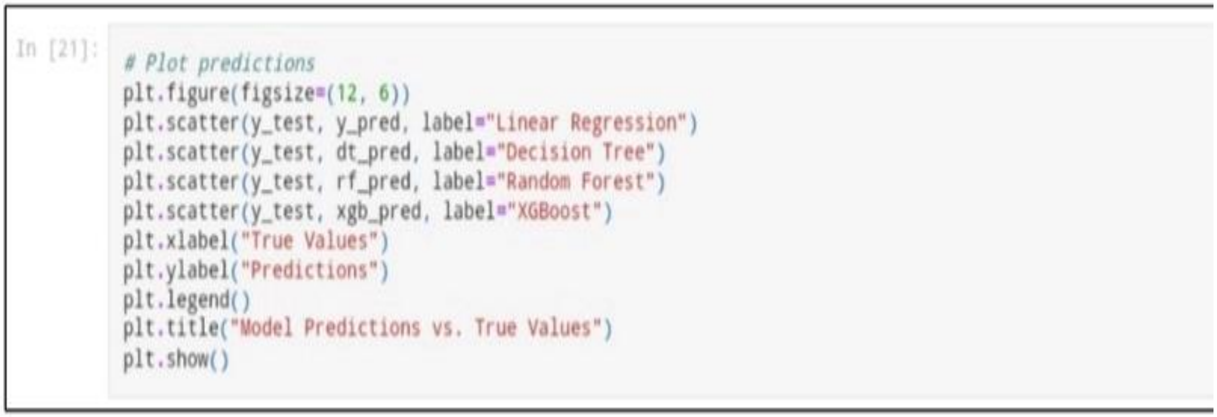
```
In [22]: dt_regressor = DecisionTreeRegressor(random_state=123)
         dt_regressor.fit(X_train, y_train)
         dt_pred = dt_regressor.predict(X_test)

         # Random Forest Regressor
         rf_regressor = RandomForestRegressor(n_estimators=100, random_state=123)
         rf_regressor.fit(X_train, y_train)
         rf_pred = rf_regressor.predict(X_test)

         # XGBoost Regressor
         xgb_r = xg.XGBRegressor(objective='reg:linear', n_estimators=30, seed=123)
         xgb_r.fit(X_train, y_train)
         xgb_pred = xgb_r.predict(X_test)

         # Calculate RMSE and R-squared for each model
         models = [lr, dt_regressor, rf_regressor, xgb_r]
         model_names = ["Linear Regression", "Decision Tree", "Random Forest", "XGBoost"]
         rmse_scores = []
         r2_scores = []

         for model, name in zip(models, model_names):
             pred = model.predict(X_test)
             rmse = np.sqrt(MSE(y_test, pred))
             r2 = r2_score(y_test, pred)
             rmse_scores.append(rmse)
             r2_scores.append(r2)
             print(f"{name} - RMSE: {rmse:.2f}, R-squared: {r2:.2f}")

         # Plot RMSE scores
         plt.figure(figsize=(10, 6))
         plt.bar(model_names, rmse_scores)
         plt.xlabel("Models")
         plt.ylabel("RMSE Score")
         plt.title("RMSE Scores for Different Regression Models")
         plt.show()
```

```
[09:30:41] WARNING: ../src/objective/regression_obj.cu:213: reg:linear is now deprecated in favor of reg:
squarederror.
Linear Regression - RMSE: 56.19, R-squared: 0.14
Decision Tree - RMSE: 48.74, R-squared: 0.35
Random Forest - RMSE: 46.27, R-squared: 0.42
XGBoost - RMSE: 46.07, R-squared: 0.42
```

BAR CHART:

The bar chart represents the difference between regression models and their RMSE

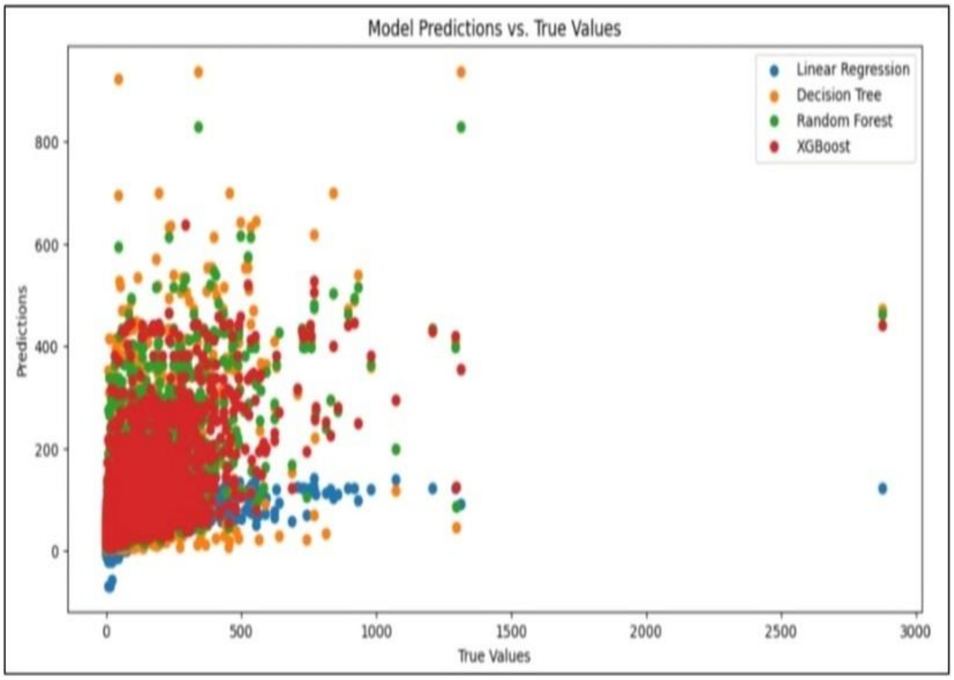




```
# Plot predictions
plt.figure(figsize=(12, 6))
plt.scatter(y_test, y_pred, label="Linear Regression")
plt.scatter(y_test, dt_pred, label="Decision Tree")
plt.scatter(y_test, rf_pred, label="Random Forest")
plt.scatter(y_test, xgb_pred, label="XGBoost")
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.legend()
plt.title("Model Predictions vs. True Values")
plt.show()
```

SCATTERPLOT:

The Scatter Plot represents the Actual predicts vs the Model Predictions and how their values differ



CONCLUSION:

In conclusion, product demand prediction with machine learning is a powerful tool for businesses seeking to optimize their operations and meet customer needs more effectively.

By leveraging advanced machine learning techniques, historical data, and relevant features, organizations can develop accurate predictive models to anticipate future demand for their products.

This predictive capability has far-reaching implications, from improved inventory management and supply chain optimization to better production planning and targeted marketing strategies. It enables businesses to stay agile and responsive in a dynamic market environment, ultimately leading to enhanced customer satisfaction and a competitive edge.

As machine learning continues to advance, product demand prediction will become an increasingly vital component of business strategy, allowing companies to adapt to changing market conditions and make data-driven decisions that drive success.