PRODUCT DEMAND PREDICTION WITH

MACHINE LEARNING

**Submitted by:**

**MALLAM SIDDHARTHA REDDY**

**au723921243030**

**phase-4 Development part-2**

**project :** PRODUCT DEMAND PREDICTION WITH

MACHINE LEARNING

**Phase-4 : Development part-2**

**Topic :**

**In this part you will continue building your project.**

**Continue building the product demand prediction model by:**

* **Feature engineering**
* **Model training**
* **Evaluation**.

**Email ID:** [**siddharthreddy213@gmail.com**](mailto:siddharthreddy213@gmail.com)

**Product Demand prediction With Machine Learning :**

Predicting product demand with machine learning is a valuable application in various industries. To get started, here's a high-level overview of the steps involved:

Data Collection :

Gather historical data on product sales, including factors that may influence demand, such as pricing, promotions, seasonality, and external events.

Clean and pre-process the data. This involves handling missing values, outliers, and encoding categorical variables.

Feature Selection/Engineering : Identify relevant features that can affect product demand. You may need to create new features or transform existing ones.

Data Splitting : Split the data into training, validation, and test sets to evaluate the model's performance.

Model Selection : Choose the appropriate machine learning model. Common choices include linear regression, decision trees, random forests, or more advanced techniques like neural networks. Model Training : Train the selected model using the training data.

Model Evaluation : Evaluate the model's performance using the validation set. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Hyper-parameter Tuning : Fine-tune the model's hyper-parameters to optimize its performance.

Model Validation : Assess the model's generalization performance on the test set to ensure it can make accurate predictions on unseen data.

Deployment : Once satisfied with the model's performance, deploy it in a real-world environment to make predictions on future demand.

**Data Loading anc Pre-processing** :

Data Sources : Identify the sources of your data, whether it's stored in databases, spreadsheets, text files, or obtained from APIs. Ensure you have access to the data you need.

Data Retrieval : Use appropriate libraries or tools to load your data into your analysis environment. For example, in Python, you can use libraries like Pandas to read data from various file formats or SQL databases.

Data Inspection: Once you've loaded the data, inspect the first few rows to get an initial understanding of its structure and contents. This step helps you verify that the data was loaded correctly.

Handling Missing Values : Identify and handle missing data. You can choose to remove rows with missing values, fill them with suitable values (e.g., mean, median), or use more advanced imputation techniques.

Dealing with Duplicates : Check for and remove duplicate records if they exist in the dataset.

Data Encoding : Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding. This is necessary for most machine learning algorithms that require numerical inputs.

**Visualization :**

Visualization is a powerful tool for understanding and communicating the results of product demand prediction. Here are some common types of visualizations you can use:

Time Series Plots : If your prediction involves time-dependent data, create time series plots to visualize historical demand and predicted demand over time. You can use line charts to show the actual demand and forecasted demand on the same graph.

Actual vs. Predicted Plots : Compare actual product demand with your model's predictions. Scatter plots or line charts can help you assess how closely your model's predictions align with the real data.

Residual Plots : Plot the residuals (the differences between actual and predicted values) over time. This can help you identify patterns or trends in prediction errors.

Histograms and Density Plots : Visualize the distribution of prediction errors to assess their normality and identify potential bias or skew in the predictions.

Box Plots : Use box plots to visualize the spread and distribution of errors, including outliers. This can help you identify areas where the model performs exceptionally well or poorly

**Evaluation Performance :**

Evaluation Metrics : These are quantitative measures used to assess how well a model or algorithm performs on a particular task. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), accuracy, precision, recall, F1 score, and many others. The choice of metric depends on the problem you are trying to solve. For example, classification tasks often use metrics like accuracy, while regression tasks use metrics like RMSE.

Training and Testing : In machine learning, you typically split your dataset into a training set and a testing set. The model is trained on the training set and then evaluated on the testing set to assess how well it generalizes to new, unseen data.

**Loading and Importing the libraries**

import numpy as np

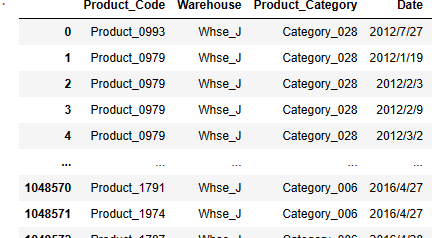
import pandas as pd

import matplotlib.pyplot as plt

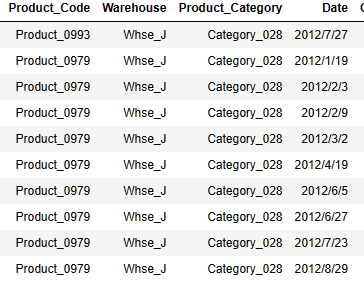
import seaborn as sns

data = pd.read\_csv("Historical Product Demand.csv")

data



data.head(10)



data.tail(10)



data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1048575 entries, 0 to 1048574

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Product\_Code 1048575 non-null object

1 Warehouse 1048575 non-null object

2 Product\_Category 1048575 non-null object

3 Date 1037336 non-null object

4 Order\_Demand 1048575 non-null object

dtypes: object(5)

memory usage: 40.0+ MB

data.isna().sum()

Product\_Code 0

Warehouse 0

Product\_Category 0

Date 11239

Order\_Demand 0

dtype: int64

data.dropna(axis = 0, inplace = True)

data.isna().sum()

Product\_Code 0

Warehouse 0

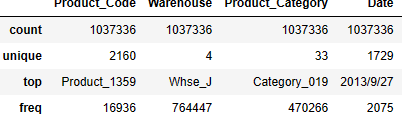
Product\_Category 0

Date 0

Order\_Demand 0

dtype: int64

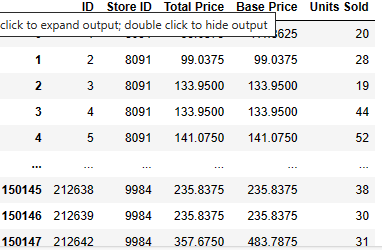
data.describe(include= 'all')



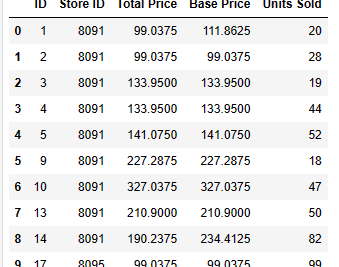
**Loading The Data:**

data1 = pd.read\_csv("PoductDemand.csv")

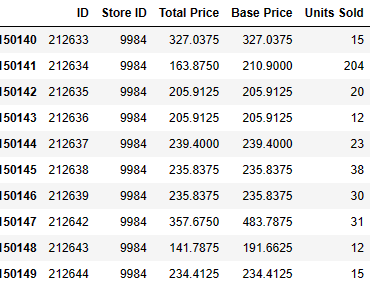
data1



data1.head(10)



data1.tail(10)



**Future Engineering and pre-processing visualization:**

data1.isna().sum()

ID 0

Store ID 0

Total Price 1

Base Price 0

Units Sold 0

dtype: int64

data1.dropna(axis = 0, inplace = True)

data1.isna().sum()

ID 0

Store ID 0

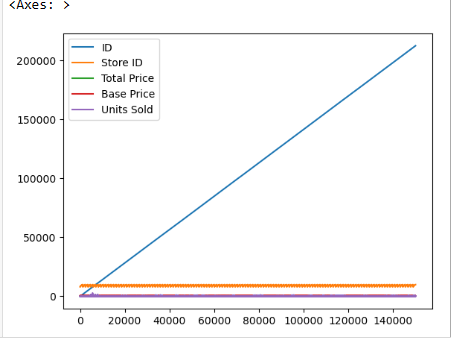
Total Price 0

Base Price 0

Units Sold 0

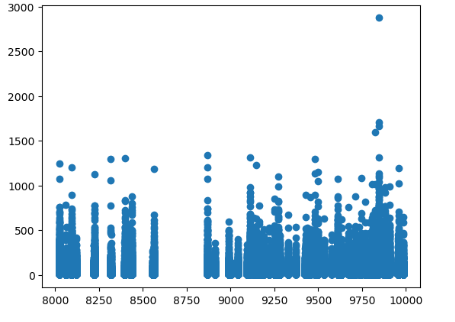
dtype: int64

data1.plot()



plt.scatter(data1["Store ID"], data1["Units Sold"])

plt.show()



data1.mean()

ID 106270.971795

Store ID 9199.420935

Total Price 206.626751

Base Price 219.424262

Units Sold 51.674543

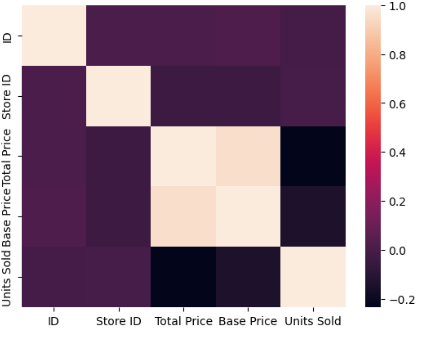
dtype: float64

data1.corr()

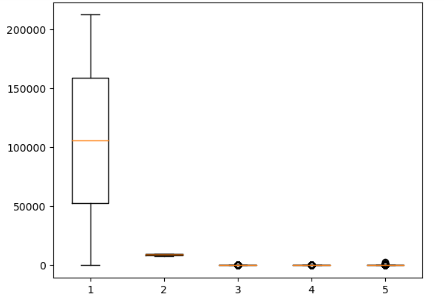


sns.heatmap(data1.corr())

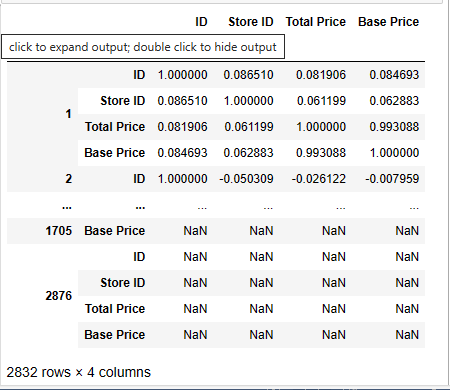
plt.show()



plt.boxplot(data1)



data1.groupby('Units Sold').corr()



data1.mean()

ID 106270.971795

Store ID 9199.420935

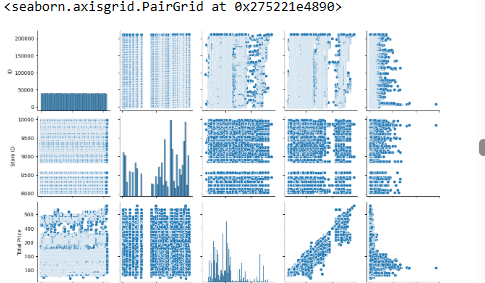
Total Price 206.626751

Base Price 219.424262

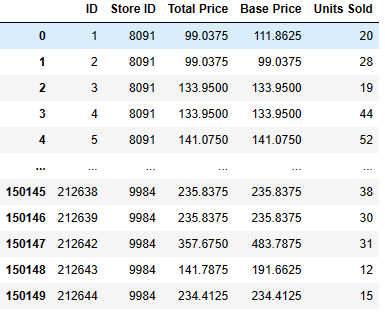
Units Sold 51.674543

dtype: float64

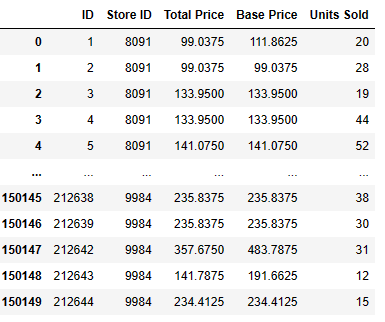
sns.pairplot(data1)



data1.dropna()



data1.fillna(0)



**Linear Regression:**

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

model = lr.fit(x\_train, y\_train)

model.intercept\_

66.30203634254558

model.coef\_

array([ -4.39017685, 0.95330134, -342.45636915, 268.59705875])

y\_pred = model.predict(x\_test)

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import r2\_score

mean\_absolute\_error(y\_test, y\_pred)

33.233948162416986

mean\_squared\_error(y\_test, y\_pred)

2935.7936428617845

r2\_score(y\_test, y\_pred)

0.1607003705784842

y\_pred.mean()

51.573538982561075

def generateX(N):

x = np.random.random(N)\*2\*np.pi

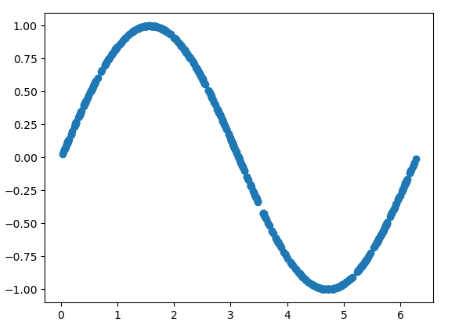
yd = np.sin(x)

return x, yd

x, y = generateX(500)

plt.scatter(x, y)

plt.show()



def plotmodel(x, y, yd):

i = x.argsort()

plt.figure()

plt.plot(x[i], y[i], "g-o")

plt.plot(x[i], yd[i], "r-o")

plt.ylabel("F(X)")

plt.xlabel("X")

plt.legend(["estimated", "True"])

plt.title("Comparision")

plt.show()

plotmodel(x, y\_pred, y)

