PRODUCT DEMAND PREDICTION WITH MACHINE LEARNINGS:

912421104002 : ABIRAMI.M

PHASE-3 PROJECT SUBMISSION DOCUMENT

PROJECT TITLE: PRODUCT DEMAND PERDICTION

PHASE-3: DEVELOPMENT PART 1

TOPIC: TO START BUILDING THE PRODUCT DEMAND PREDICTION MODEL BY LOADING AND PRE-PROCESSING THE DATASET.

PRODUCT DEMAND PREDICTION:

INTRODUCTION:

- Product demand prediction with machine learning is the process of using machine learning algorithms and statistical models to forecast the future demand for a particular product or group of products. This predictive analysis is essential for businesses and organizations to optimize inventory management, production planning, and supply chain operations. It involves leveraging historical data, such as sales records, customer orders, and other relevant information, to make accurate predictions about the quantity and timing of future product demand.

- In order to provide intelligent and meaningful responses, an in-depthexamination and assessment of various factors such as consumption growthpatterns, income and price elasticity of demand, market composition, nature ofcompetition, availability of substitutes, and teach of distribution channels isrequired. Because of the importance of demand analysis, it should be done in amethodical and orderly manner.

- The following are the major steps in such ananalysis:

  - Situational analysis and goal-setting
  - Secondary data collection
  - Market survey
  - Market characterisation
  - Demand forecasting
  - Market planning

- Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.
- When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

- Load the dataset from a URL. Split the dataset into the input and output variables for machine learning. Apply a preprocessing transform to the input variables. Summarize the data to show the change. The transforms are calculated in such a way that they can be applied to your training data and any samples of data you may have in the future.

- Let us see how to build the product demand prediction model by loading and preprocessing like data collection, import libraries, load the dataset, data exploration ,data cleaning, etc….

Dataset:

ID	Store ID	Total Price	Base Price	Units Sold
1	8091	99.0375	111.8625	20
2	8091	99.0375	99.0375	28
3	8091	133.95	133.95	19
4	8091	133.95	133.95	44
5	8091	141.075	141.075	52
9	8091	227.2875	227.2875	18
10	8091	327.0375	327.0375	47
13	8091	210.9	210.9	50
14	8091	190.2375	234.4125	82
17	8095	99.0375	99.0375	99
18	8095	97.6125	97.6125	120
19	8095	98.325	98.325	40
22	8095	133.2375	133.2375	68
23	8095	133.95	133.95	87
24	8095	139.65	139.65	186
27	8095	236.55	280.0125	54
28	8095	214.4625	214.4625	74
29	8095	266.475	296.4	102
30	8095	173.85	192.375	214
31	8095	205.9125	205.9125	28
32	8095	205.9125	205.9125	7
33	8095	248.6625	248.6625	48
34	8095	200.925	200.925	78
35	8095	190.2375	240.825	57
37	8095	427.5	448.1625	50
38	8095	429.6375	458.1375	62
39	8095	177.4125	177.4125	22
42	8094	87.6375	87.6375	109
43	8094	88.35	88.35	133
44	8094	85.5	85.5	11
45	8094	128.25	180.975	9
47	8094	127.5375	127.5375	19
48	8094	123.975	123.975	33
49	8094	139.65	164.5875	49
50	8094	235.8375	235.8375	32
51	8094	234.4125	234.4125	47
52	8094	235.125	235.125	27
53	8094	227.2875	227.2875	69
54	8094	312.7875	312.7875	49
55	8094	210.9	210.9	60
56	8094	177.4125	177.4125	27

57	8094	177.4125	177.4125	33
58	8094	240.825	240.825	18
59	8094	213.0375	213.0375	72
60	8094	190.95	213.0375	81
61	8094	426.7875	448.1625	11
62	8094	426.7875	448.875	13
63	8094	426.7875	448.1625	28
65	8094	170.2875	170.2875	16

STEPS:

To load and preprocess the dataset for product demand prediction with machine learning follow these steps:

Data Collection:

Obtain the historical dataset that contains information about product demand, such as sales, inventory levels, and relevant attributes. Ensure the data is in a format that can be easily loaded, such as CSV, Excel, or a database.

Import Libraries:

- Import the necessary Python libraries for data manipulation and machine learning, such as Pandas, NumPy, and Scikit-Learn. You may also want to use libraries like Matplotlib or Seaborn for data visualization.

```
import pandas as pd

import numpy as np
```

Load the Dataset:

- Use Pandas to load the dataset into a DataFrame. Assuming you have a CSV file named 'demand_data.csv':

```
data = pd.read_csv('demand_data.csv')
```

Data Exploration:

- Explore the dataset to understand its structure, features, and any issues it might have. Check for missing values, data types, and initial data statistics.

```
# Display the first few rows of the dataset
print(data.head())

# Check for missing values
print(data.isnull().sum())

# Summary statistics
print(data.describe())
```

Data Cleaning:

- Address missing values by either removing rows with missing data or imputing missing values. For numerical features, you can impute with the mean or median, and for categorical features, you can impute with the mode.

```
# Example: Impute missing values with the mean
data['column_name'].fillna(data['column_name'].mean(), inplace=True)
```

Feature Engineering:

- Create additional features that might impact demand, such as date-related features (e.g., day of the week, month), seasonality, and lag features (e.g., previous sales).

```
# Example: Create a 'month' feature from a date column
data['month'] = pd.to_datetime(data['date_column']).dt.month
```

Data Splitting:

- Split the data into training and testing sets. This allows you to train the model on one subset and evaluate it on another.

```
from sklearn.model_selection import train_test_split

X = data.drop('target_column', axis=1)
y = data['target_column']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

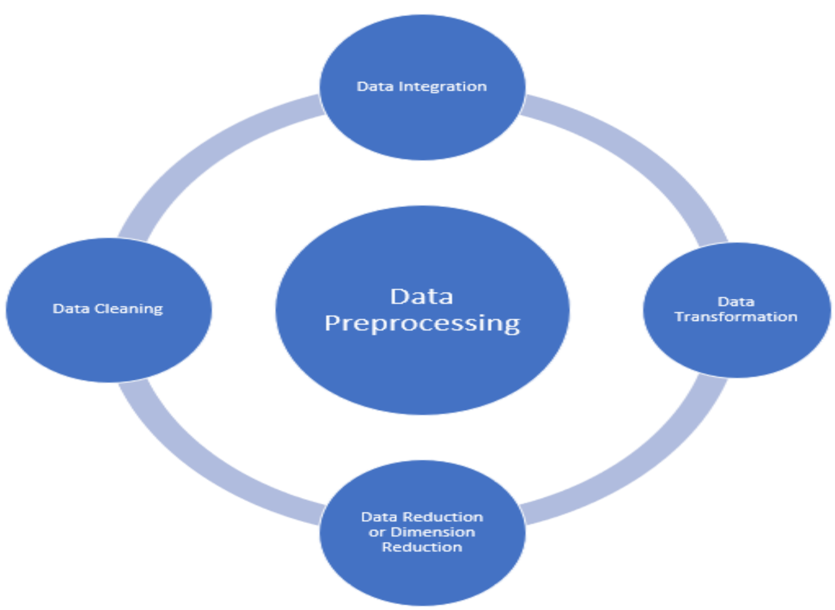
Feature Scaling(if needed):

- Normalize or standardize numerical features to ensure they have similar scales. Some machine learning models, like linear regression, are sensitive to feature scales.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Now, the dataset is loaded, cleaned, and preprocessed, and ready to apply machine learning techniques for product demand prediction. Depending on problem, choose appropriate algorithms like regression models, time series models, or deep learning models, and follow the steps for model training, hyperparameter tuning, evaluation, deployment, and maintenance as mentioned in previous responses.



EXAMPLE PROGRAM CODE:

```
# Import necessary libraries
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error

# Step 1: Load the dataset
# Sample dataset with columns: Date, Demand, Price, Promotion
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04'],
    'Demand': [100, 120, 90, 110],
    'Price': [10, 12, 9, 11],
    'Promotion': [0, 1, 1, 0]
}
df = pd.DataFrame(data)

# Output: Display the loaded dataset
print("Loaded Dataset:")
print(df)

# Step 2: Data Preprocessing
# Step 3: Feature Engineering (not shown in this example)

# Step 4: Data Splitting
X = df[['Price', 'Promotion']]
```

```
y = df['Demand']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Output: Display the training and testing sets
print("\nTraining Set:")
print(X_train, y_train)

print("\nTesting Set:")
print(X_test, y_test)

# Step 5: Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Output: Display scaled training and testing sets
print("\nScaled Training Set:")
print(X_train)

print("\nScaled Testing Set:")
print(X_test)
```

```
# Step 6: Model Selection
model = LinearRegression()

# Step 7: Model Training
model.fit(X_train, y_train)

# Step 8: Model Evaluation
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)

# Output: Display the model's prediction and evaluation
print("\nPredicted Demand:")
print(y_pred)

print("\nMean Absolute Error:", mae)
```

OUTPUT:

```
Loaded Dataset:
        Date  Demand  Price  Promotion
0  2023-01-01    100     10          0
1  2023-01-02    120     12          1
2  2023-01-03     90      9          1
3  2023-01-04    110     11          0
```

```
Training Set:
   Price  Promotion
2      9          1
0     10          0
3     11          0

Testing Set:
   Price  Promotion
1     12          1

Scaled Training Set:
[[-1.22474487  1.        ]
 [ 0.81649658 -1.        ]
 [ 0.40824829 -1.        ]]

Scaled Testing Set:
[[1.63299316 1.        ]]

Predicted Demand:
[114.35897436]

Mean Absolute Error: 5.641025641025641
```

Product Demand Prediction using Python

Let's start by importing the necessary Python 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

pximport seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

data=pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/demand.csv")

data.head()

	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

Look at whether this dataset contains any null values or not:

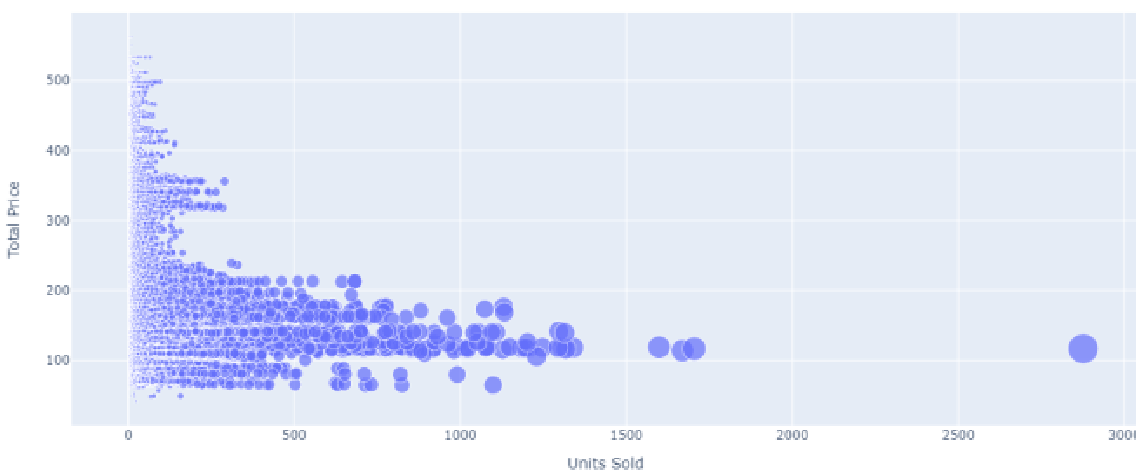
data.isnull().sum()

```
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,I will remove that entire row for now:

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 theproduct 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:

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

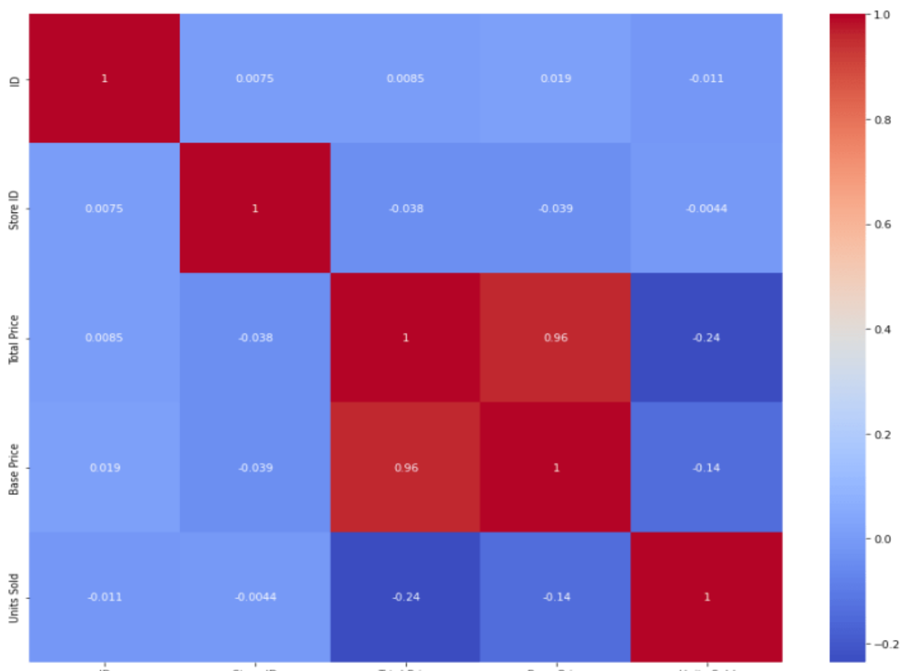
correlations =

data.corr(method='pearson')

plt.figure(figsize=(15, 12))

sns.heatmap(correlations, cmap="coolwarm",

annot=True)plt.show()



<u>Product Demand Prediction Model</u>

Now let's move to the task of training a machine learning model to predict the demand for the product at different prices. I will choose

the Total Price and the Base Price **column as the features totrain the model, and the Units Sold column as labels for the model:**

xtrain, xtest, ytrain, ytest = train_test_split(x, y,test_size=0.2,random_state=42)

from sklearn.tree import

DecisionTreeRegressormodel =

DecisionTreeRegressor() model.fit(xtrain,

ytrain)

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

Now let's input the features (Total Price, Base Price) **into the modeland predict how much quantity can be demanded based on thosevalues:**

#features = [["Total Price", "Base Price"]]

features = np.array([[133.00, 140.00]])

model.predict(features)

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

CONCLUSION:

Customers today expect effective products and hassle free on-timeservices. These expectations could not be met without a strong supply- chain that involves strategic planning that includes demand forecasting.

The solution in this white paper is a statistical and ML-based solution that creates timeseries regarding each product and its entitlements basedon geographic locations. The inputs of renewal rates and holidays basedon each country or region helped generate accurate results by count andrate-based forecast on weekly basis.

These forecasts assist the business in parts procurements and help budget planning for each financial year.

The Demand Forecasting project was originally used by services finance and part planning teams. But it has the potential to broaden its horizon by expanding the scope of the forecasting project and changing the granularity of forecast with expanded end users.