PHASE-3

IBM NAN MUDHALVAN

Product demand prediction with machine learning

To load and preprocess a dataset for product demand prediction, you'll first need to download the dataset from the provided link. Assuming you have already downloaded the dataset, follow these steps to get started with the data loading and preprocessing:

1.Import Necessary Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the Dataset:

```
# Replace 'path_to_dataset' with the actual path to your downloaded dataset

data = pd.read_csv("path_to_dataset.csv")
```

3. Explore the Dataset:

Before preprocessing, it's important to get a sense of your data. This includes checking the first few rows, data types, and summary statistics.

print(data.head())

print(data.info())

print(data.describe())

4. Data Cleaning:

In this step, you should address issues such as missing values, duplicates, and outliers. Handle these issues as appropriate for your dataset.

# Check for missing values

print(data.isnull().sum())

# Remove duplicates if any

data.drop_duplicates(inplace=True)

# Handle missing values (e.g., by imputing or removing rows with missing values)

data.dropna(inplace=True)

5. Feature Engineering:

Create new features or preprocess existing ones as necessary. For demand prediction, you might want to extract date-related features, calculate moving averages, or decompose time series data if applicable.

```
# Example of date-related feature extraction (assuming a 'date' column exists)
data['date'] = pd.to_datetime(data['date'])
data['year'] = data['date'].dt.year
data['month'] = data['date'].dt.month
data['day'] = data['date'].dt.day
```

6. Data Splitting:

Split your dataset into training and testing sets. You can use techniques like time-based splitting or random splitting, depending on your specific use case.

```
From sklearn.model_selection import train_test_split

# Define your features and target variable
X = data.drop(columns=['demand'])  # Features
y = data['demand']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

7. Scaling and Normalization (if needed):

Depending on the algorithms you plan to use, you may need to scale or normalize your features. Common techniques include Standardization (mean=0, std=1) or Min-Max scaling.

8. Data Visualization (Optional):

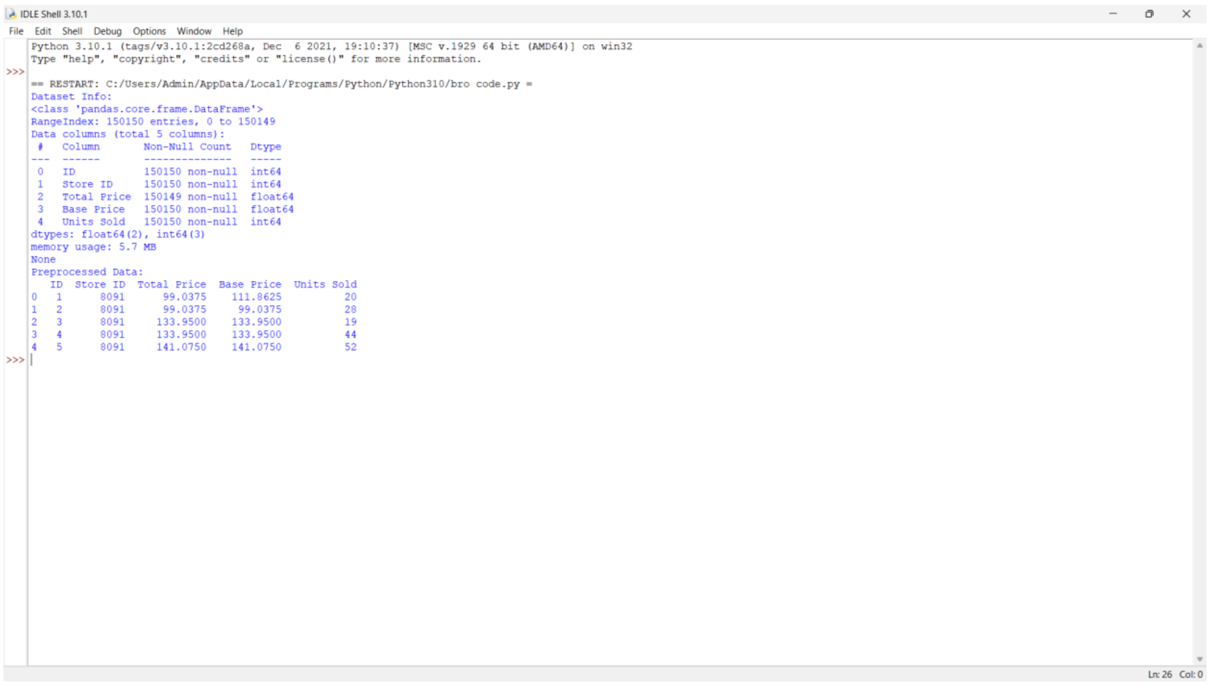
Visualize your data to gain insights into its distribution, patterns, and relationships. This can be helpful in understanding the characteristics of the dataset.

9. Encoding Categorical Variables (if needed):

If your dataset contains categorical variables, encode them into numerical format. You can use one-hot encoding, label encoding, or other methods.

Now you have loaded and preprocessed your dataset. The next steps involve selecting an appropriate model, training it, and evaluating its performance for demand prediction. The choice of model will depend on the nature of your data and specific project requirements. You can proceed with selecting and training a model based on your dataset and objectives.

10. OUTPUT



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/Admin/AppData/Local/Programs/Python/Python310/bro code.py =
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150150 entries, 0 to 150149
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   ID           150150 non-null  int64
 1   Store ID     150150 non-null  int64
 2   Total Price  150149 non-null  float64
 3   Base Price   150150 non-null  float64
 4   Units Sold   150150 non-null  int64
dtypes: float64(2), int64(3)
memory usage: 5.7 MB
None
Preprocessed Data:
   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
>>> |
```