APPLIED DATA SCIENCE

IBM NAAN MUDHALVAN Phase-2

TEAM MEMBERS

- DHANULESH.S
- SURIYA.V
- NARENTHIRA KUMAR.G
- MAHESWAREN.R
- ARYAN REDDY.R

PROJECT TITLE : PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING

PHASE 3 : DEVOLPEMENT PART 1

TOPIC : Start building the product demand prediction model by loading and preprocessing the dataset. Collect and preprocess the historical sales data and external factors for analysis.

INTRODUCTION:

Data science preprocessing is the critical phase of preparing and cleaning data before analysis. It involves tasks such as data cleaning, handling missing values, scaling, encoding categorical variables, and feature selection.

Effective preprocessing is essential for accurate and reliable results in data analysis and machine learning.

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

1. **Import Libraries:**
    - Import the necessary Python libraries and modules, such as Pandas, NumPy, Matplotlib, Seaborn, Scikit-Learn, or any other libraries specific to your data analysis or machine learning tasks.
2. **Data Collection:**
    - Gather the raw data from various sources, such as databases, files, APIs, or sensors.
3. **Data Cleaning:**
    - Handle missing values: Identify and either impute missing data or remove rows/columns with missing values.
    - Handle duplicate records: Identify and remove duplicate entries.
    - Outlier detection and treatment: Identify and handle outliers, which may involve removal, transformation, or imputation.
    - Data type conversion: Ensure that data types are appropriate for analysis or modeling.
4. **Data Exploration:**
    - Explore the data to understand its distribution, statistics, and relationships among variables.
    - Visualization: Create plots and charts to visualize the data and identify patterns, trends, and potential issues.
5. **Feature Engineering:**
    - Select relevant features: Choose the most important variables for your analysis or model.
    - Create new features: Generate new variables that can capture meaningful information from existing ones.
    - Feature scaling: Normalize or standardize features to ensure they have a similar scale, especially for machine learning algorithms sensitive to feature magnitude.
6. **Data Transformation:**
    - One-Hot Encoding: Convert categorical variables into binary (0/1) indicators for each category.
    - Feature Scaling: Standardize or normalize numerical features to ensure they have a similar scale.
    - Dimensionality Reduction: Apply techniques like Principal Component Analysis (PCA) to reduce the

number of features while preserving relevant information.

7. **Data Splitting:**
    - Split the dataset into training, validation, and test sets. This is essential for model evaluation.
    - Ensure the data split maintains the same distribution of target values (stratified splitting in classification problems).

IMPORTANCE OF LOADING AND PROCESSING DATASET:

Importing Libraries:

- Importance: This step is essential as it allows you to load and use the libraries and tools needed for data analysis, manipulation, and modeling. Without the right libraries, you won't be able to perform subsequent steps effectively.

Data Collection:

- Importance: Collecting data is the foundation of any data analysis or modeling project. It ensures you have the raw material to work with, and the quality of the data you collect impacts the reliability and validity of your results.

Data Cleaning:

- Importance: Cleaning the data helps in removing errors, inconsistencies, and missing values, which can distort your analysis or lead to incorrect model outcomes. Clean data is essential for trustworthy results.

Data Exploration:

- Importance: Exploring the data helps you understand its characteristics, distributions, and relationships between variables. It provides insights into the data, guides feature selection, and helps in hypothesis generation.

Feature Engineering:

- Importance: Feature engineering involves selecting relevant features and creating new ones. Proper feature selection and engineering can improve model performance and reduce overfitting, as it focuses on the most informative variables.

Data Transformation:

- Importance: Data transformation steps like one-hot encoding, feature scaling, and dimensionality reduction prepare the data for modeling. They ensure that the data is in a format that can be

effectively used by machine learning algorithms.

Data Splitting:

- Importance: Splitting the data into training, validation, and test sets is crucial for model evaluation and avoiding overfitting. It allows you to assess the model's performance on unseen data, improving its generalizability.

Each of these steps is a critical component of the data preprocessing process, and they collectively contribute to the quality and reliability of your data analysis or machine learning model. Skipping or neglecting any of these steps can lead to incorrect results, biases, or issues that may not be immediately apparent but can have a significant impact on the final outcomes.

CHALLENGES IN LOADING AND DATA PREPROCESSING:

- Choosing the right libraries and ensuring their compatibility can be challenging. It may also be time-consuming to install and configure the necessary packages, especially in complex projects.

- Gathering high-quality and relevant data can be difficult. Data may be dispersed across different sources, and integrating it can be complex. There could also be privacy and ethical considerations when collecting data from various sources.

- Dealing with missing data, outliers, and errors can be a significant challenge. Deciding how to handle missing data, especially when it's a substantial portion of the dataset, requires careful consideration. Handling outliers may involve domain expertise.

- Large datasets can make data exploration computationally intensive and time-consuming. Understanding complex relationships between variables may also be challenging.

- Selecting the right features and engineering new ones require domain knowledge and creativity. The process can be time-consuming, and overengineering or selecting irrelevant features can harm model performance.

- Encoding categorical variables, scaling features, and reducing dimensionality must be done carefully. Choosing the right method and ensuring that the transformation doesn't introduce bias or loss of important information is a challenge.

- Ensuring that data splitting maintains the same distribution of target values, especially in imbalanced datasets, can be challenging. Additionally, determining the appropriate split ratios for training, validation, and test sets is not always straightforward.

PROGRAMS ARE WRITTEN BELOW

LOADING THE DATASET

PREPROCESSING THE DATASET

Import necessary libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
```

In [2]:
```
data = pd.read_csv("PoductDemand.csv")
data
```

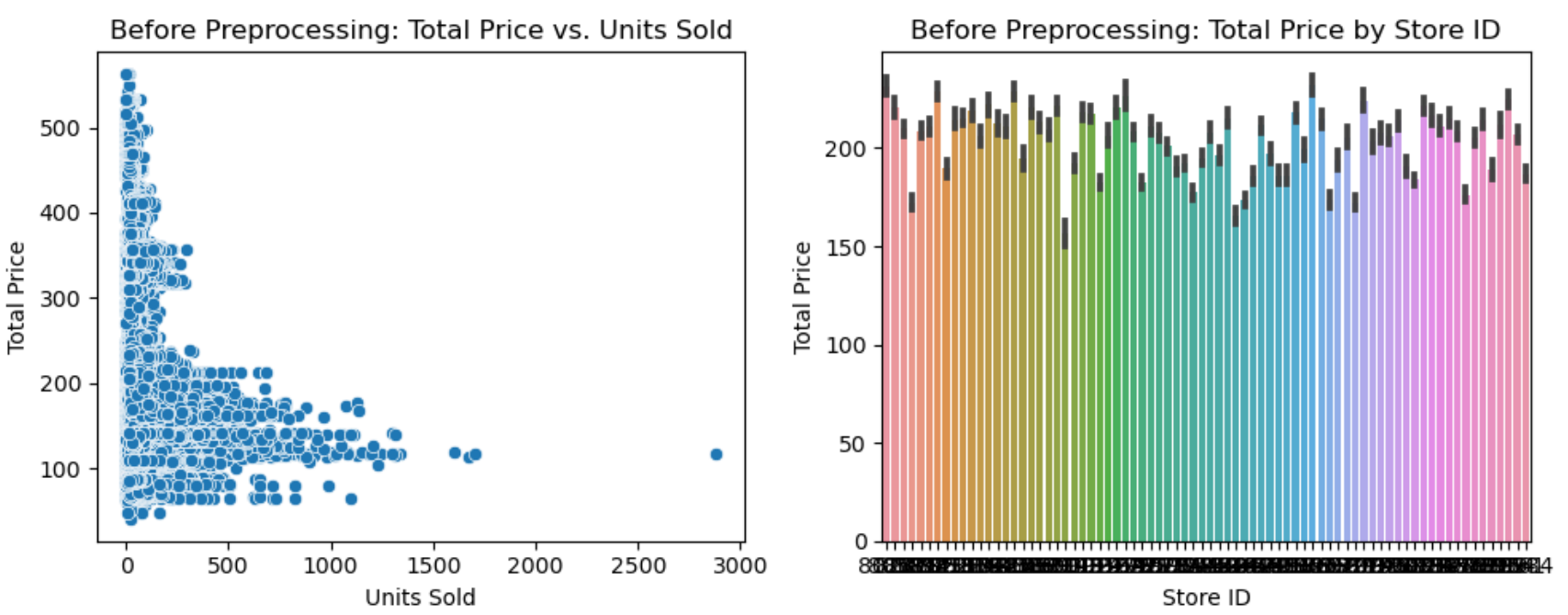
Out[2]:

	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
...	...	...	...	...	...
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

150150 rows × 5 columns

Data Visualization(before preprocessing)

In [4]:
```
plt.figure(figsize=(10, 4))

# Plot 1: Total Price vs. Units Sold using Seaborn
plt.subplot(1, 2, 1)
sns.scatterplot(x='Units Sold', y='Total Price', data=data)
plt.title('Before Preprocessing: Total Price vs. Units Sold')

# Plot 2: Total Price by Store ID using Seaborn
plt.subplot(1, 2, 2)
sns.barplot(x='Store ID', y='Total Price', data=data)
plt.title('Before Preprocessing: Total Price by Store ID')

plt.tight_layout()
plt.show()
```



1.Data Cleaning

In [5]:
```
data.fillna(0, inplace=True)
```

2.Data Encoding

In [6]:
```
label_encoder = LabelEncoder()
data['Store ID'] = label_encoder.fit_transform(data['Store ID'])
```

3.Data Scaling and Normalization

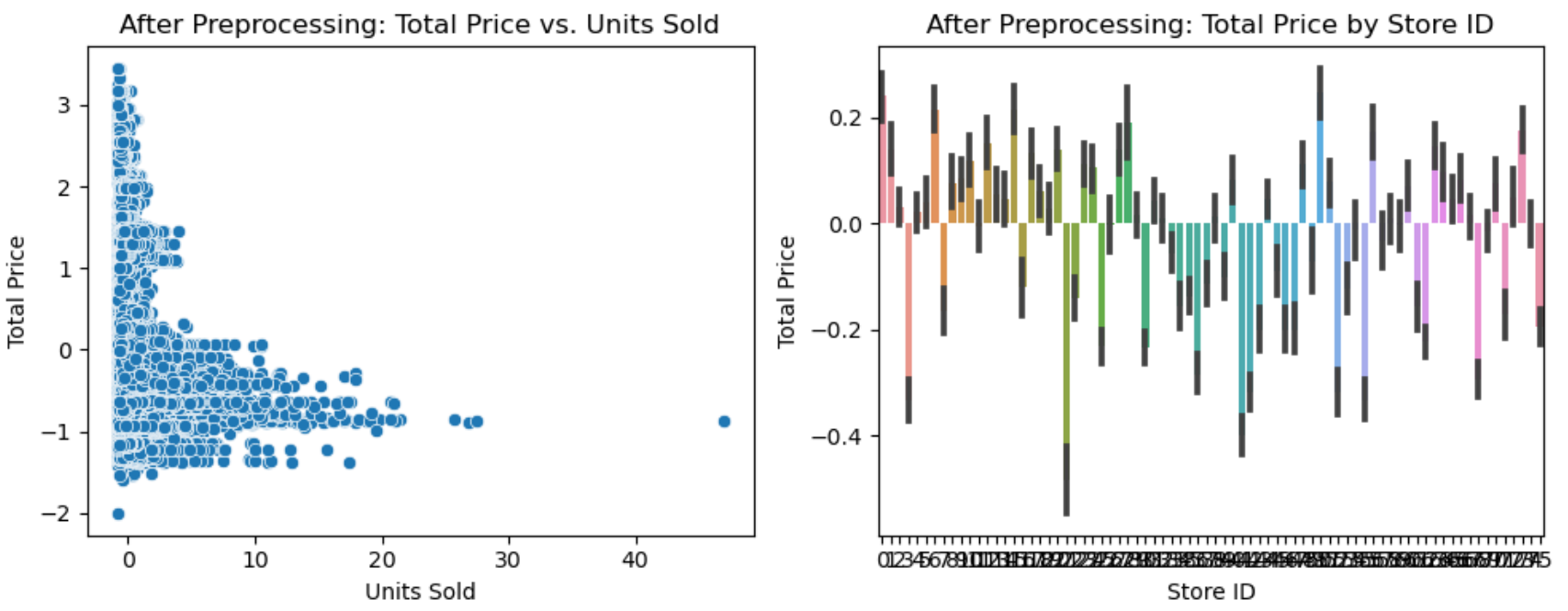
In [7]:
```
scaler = StandardScaler()
data[['Total Price', 'Base Price', 'Units Sold']] = scaler.fit_transform(data[['Total Price', 'Base Price', '
```

4.Feature Selection

In [8]:
```
selected_features = data[['Store ID', 'Total Price', 'Base Price', 'Units Sold']]
```

Data Visualization after Preprocessing

In [9]:
```
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
sns.scatterplot(x='Units Sold', y='Total Price', data=data)
plt.title('After Preprocessing: Total Price vs. Units Sold')

plt.subplot(1, 2, 2)
sns.barplot(x='Store ID', y='Total Price', data=data)
plt.title('After Preprocessing: Total Price by Store ID')

plt.tight_layout()
plt.show()
```



5.Data plitting

In [10]:
```
X = selected_features
y = data['Total Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Importing Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
```

Load csv File

In [2]:
```
data = pd.read_csv("ProductDemand.csv")
data
```

Out[2]:

	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
...	...	...	...	...	...
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

150150 rows × 5 columns

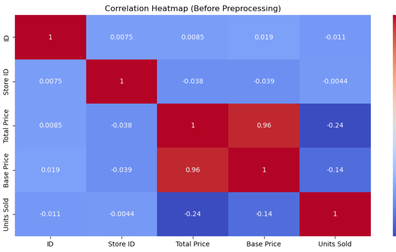
Data Inspection

In [3]:
```
print(data.head())
print(data.info())
```

```
   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
<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
```

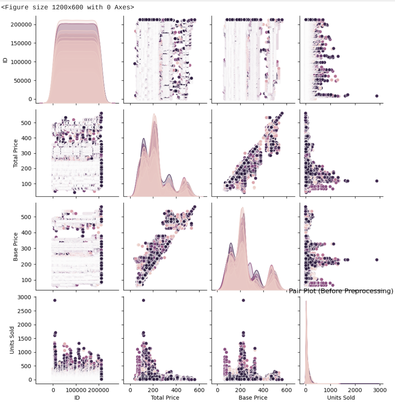
Data Visualization

In [4]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap (Before Preprocessing)")
plt.show()
```



In [6]:
```
plt.figure(figsize=(12, 6))
sns.pairplot(data, hue='Store ID')
plt.title("Pair Plot (Before Preprocessing)")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



Data Cleaning

Removal of Data Duplicates

In [7]:
```
data = data.drop_duplicates()
```

Handle Missing Values

In [8]:
```
data = data.dropna()
```

Data Tranformation

In [9]:
```
numeric_features = [ "Base Price", "Units Sold", "Store ID"]
```

Data Splitting

In [10]:
```
X = data[numeric_features]
y = data["Total Price"]
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Feature Transformation Pipeline

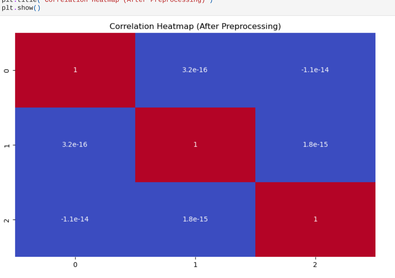
In [12]:
```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features)
    ])
```

Dimensionality Reduction with PCA

In [13]:
```
pca = PCA(n_components=3)
data_preprocessed_pca = pca.fit_transform(preprocessor.fit_transform(X))
```

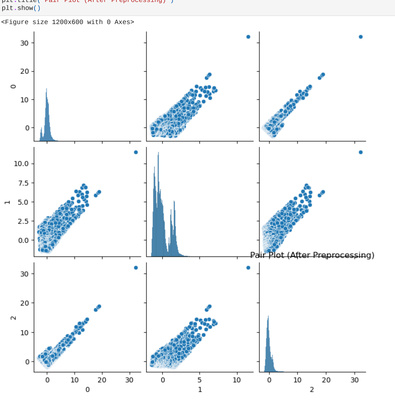
Data Visualization

In [14]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(pd.DataFrame(data_preprocessed_pca).corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap (After Preprocessing)")
plt.show()
```



In [15]:
```
plt.figure(figsize=(12, 6))
sns.pairplot(pd.DataFrame(data_preprocessed_pca))
plt.title("Pair Plot (After Preprocessing)")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



CONCLUSION

- Data preprocessing is the essential groundwork for data analysis and machine learning. It focuses on enhancing data quality, optimizing features, and building reliable models. While it comes with challenges, it is a critical phase that underpins meaningful insights and the effectiveness of machine learning models, making it a necessary investment in any data-driven project.

- Data preprocessing streamlines the dataset, making it more efficient in terms of storage and computation. This is particularly important when working with large datasets or deploying models in resource-constrained environments.
- Through proper preprocessing, models are better equipped to generalize from the training data to make accurate predictions or classifications on new, unseen data, ensuring the model's real-world applicability.

.