Product-Demand-Prediction -with-Machine-Learnings

Phase 5

Submission document

Project title:
Product-Demand-Prediction-with-Machine-Learnings

Phase 5: final report

Topic: Start the prediction of Product-Demand-Prediction-with-Machine-Learnings by loading and pre-processing the dataset and do further for the dataset by evaluating the appropriate operations on the code or dataset.

Product Demand Forecasting using Machine Learning

We have intiated ,Demand forecasting is the process of making estimations about future customer demand over a defined period, using historical data and other information.

Usually organisations follow traditional forecasting techniques/algorithms such as Auto Arima, Auto Arima, Sarima, Simple moving average and many more

Our project:

The vendors who are selling everyday items need to keep their stock up to date so, that no customer returns from their shop empty hand.

Inventory Demand Forecasting using Machine Learning

In this article, we will try to implement a machine learning model which can predict the stock amount for the different products which are sold in different stores.

Goal :

Due to the recent boost in AI world, companies have started researching the possibility of using machine learning in place of traditional approach

Tuning traditional algorithms takes a significant amount of effort and domain expertise as well.

In this repo, we are trying to figure out a way to predict the same using machine learning algorithms.

Data :

The dataset comprised of units sold on a daily basis along with details regarding the sales, eg. SKU(product id), Store, price etc.

record_ID, week, store_id, sku_id, total_price, base_price, is_featured_sku, is_display_sku, units_sold
Python libraries make it easy for us to handle the data and perform typical and complex tasks with a single line of code.

Pandas – This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.
Numpy – Numpy arrays are very fast and can perform large computations in a very short time.
Matplotlib/Seaborn – This library is used to draw visualizations.
Sklearn – This module contains multiple libraries are having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.
XGBoost – This contains the eXtreme Gradient Boosting machine learning algorithm which is one of the algorithms which helps us to achieve high accuracy on predictions.

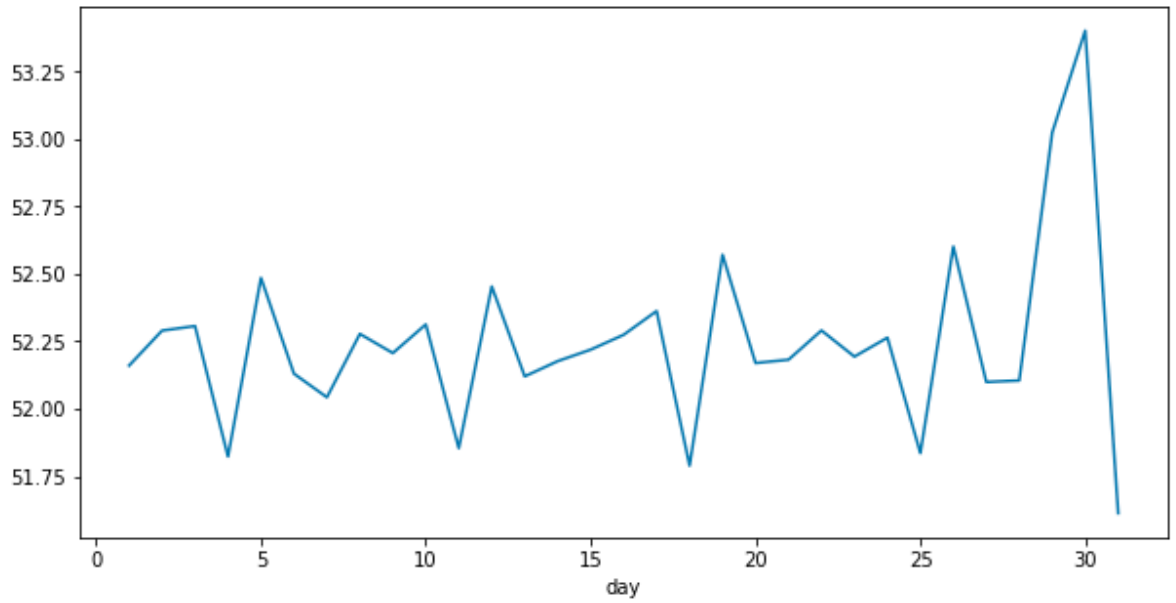
Workflow :

- Handling missing values
- Feature selection based on my previous experience in Supply chain domain
- Converting dataset into time series format to apply supervised learning approach.
- Regression Modeling
- Random Forest
- XGBoost
- SVM (future scope)
- Hyperparameter Tuning

Sample dataset:

- Data Pre-processing: Date Conversion: Convert the date column to a datetime object if it's not already in that format.
- Sorting: Sort the Data Frame by date if it's not already in chronological order.
- Feature Engineering: Create additional features if needed, such as moving averages, daily returns, or technical indicators. Normalization/Scaling: If you plan to use machine learning models, you might want to normalize or scale the data. Common techniques include Min-Max scaling or

standardization.



stock record

	record_ID	week	store_id	sku_id	total_price	base_price	is_featured_sku	is_display_sku	units_sold
0	1.0	17/01/11	8091.0	216418.0	99.0375	111.8625	0.0	0.0	20.0
1	2.0	17/01/11	8091.0	216419.0	99.0375	99.0375	0.0	0.0	28.0
2	3.0	17/01/11	8091.0	216425.0	133.9500	133.9500	0.0	0.0	19.0
3	4.0	17/01/11	8091.0	216233.0	133.9500	133.9500	0.0	0.0	44.0
4	5.0	17/01/11	8091.0	217390.0	141.0750	141.0750	0.0	0.0	52.0
...	...	...	...	...	...	...	...	...	...
150145	212638.0	09/07/13	9984.0	223245.0	235.8375	235.8375	0.0	0.0	38.0
150146	212639.0	09/07/13	9984.0	223153.0	235.8375	235.8375	0.0	0.0	30.0
150147	212642.0	09/07/13	9984.0	245338.0	357.6750	483.7875	1.0	1.0	31.0
150148	212643.0	09/07/13	9984.0	547934.0	141.7875	191.6625	0.0	1.0	12.0
150149	212644.0	09/07/13	9984.0	679023.0	234.4125	234.4125	0.0	0.0	15.0

150150 rows × 9 columns

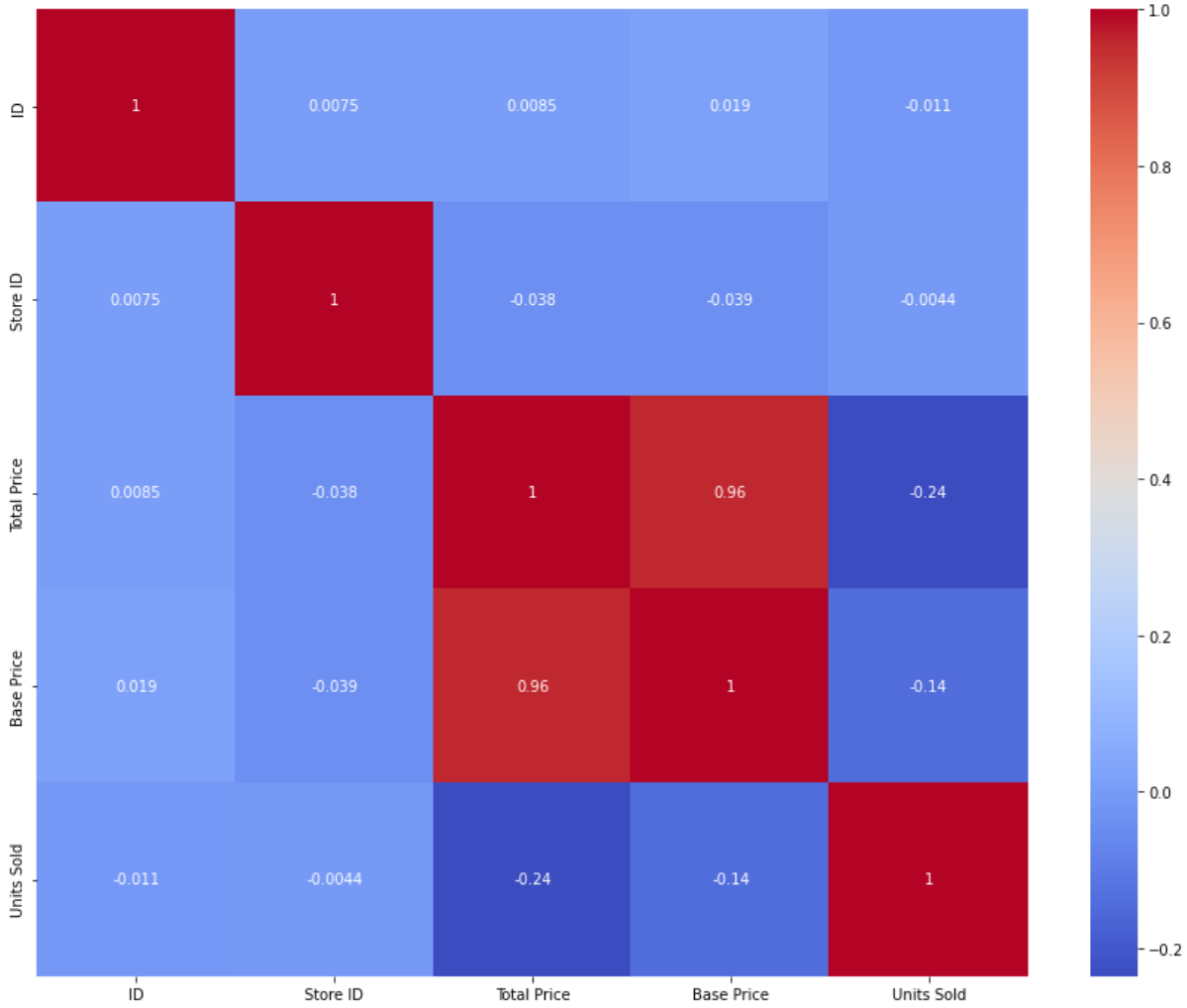
```
print(data.corr())

correlations = data.corr(method='pearson')

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

sns.heatmap(correlations,
cmap="coolwarm", annot=True)

plt.show()
```


ID
Store ID
Total Price
Base Price
Units Sold
1
0.0075
0.0085
0.019
-0.011
0.0075
1
-0.038
-0.039
-0.0044
0.0085
-0.038
1
0.96
-0.24
0.019
-0.039
0.96
1
-0.14
-0.011
-0.0044
-0.24
-0.14
1
ID
Store ID
Total Price
Base Price
Units Sold
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error as mae

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('StoreDemand.csv')
display(df.head())
display(df.tail())
```

OUTPUT:

	date	store	item	sales
0	2013-01-01	1	1	13
1	2013-01-02	1	1	11
2	2013-01-03	1	1	14
3	2013-01-04	1	1	13
4	2013-01-05	1	1	10

	date	store	item	sales
912995	2017-12-27	10	50	63
912996	2017-12-28	10	50	59
912997	2017-12-29	10	50	74
912998	2017-12-30	10	50	62
912999	2017-12-31	10	50	82

About Data:

Easypreprocessing is a library used for common ML related preprocessing activities.

In [2]:

prep = EasyPreProcessing('train.csv') //data

Data.csv

To create a detailed dataset for a product demand prediction project, you need to include various attributes that

can affect demand. Here's a more comprehensive dataset with explanations for each column:

1. Date or Timestamp:

- This column represents the date and time of each observation. It allows you to track how demand changes over time and identify seasonality trends.

2. Product Attributes:

- You may have various product-related attributes that can influence demand, such as:
    - Product ID or SKU: A unique identifier for each product.
    - Product Category: The category to which the product belongs (e.g., electronics, clothing, groceries).
    - Brand: The brand or manufacturer of the product.
    - Product Description: A brief description of the product.

3. Pricing Information:

- Pricing is a crucial factor in demand prediction:
    - Price: The price of the product.
    - Discounts or Promotions: Indicate whether any discounts or promotions are running during a specific time frame.

4. Seasonal Factors:

- Seasonal factors can significantly impact demand, so it's essential to include them:
    - Time of Year: You can include columns like "Month" and "Quarter" to capture seasonal trends.

- Holidays: Identify holidays or special events that might affect demand.

5. Marketing and Advertising:

- Data related to marketing and advertising efforts:
  - Advertising Spend: The amount spent on advertising campaigns.
  - Marketing Channels: The channels through which marketing is conducted (e.g., TV, social media, email).

6. Historical Sales Data:

- This is the target variable that you want to predict:
  - Sales or Demand: The number of units sold during a specific time period.

7. External Factors:

- Consider including external factors that can impact demand, such as:
  - Economic Indicators: Data on economic conditions (e.g., GDP, unemployment rate).
  - Competitor Data: Information about competitor products and pricing.

8. Customer Demographics:

- If available, you can include information about your customer base:
  - Customer Segmentation: Data on different customer segments.
  - Customer Behavior: Customer purchasing history or preferences.

9. Weather Data (If Applicable):

- For certain products, weather conditions can have a significant impact on demand. Include columns such as temperature, precipitation, and weather events.

10. Geographical Information (If Applicable):

- If demand varies by location, include geographical data such as region, city, or store location.

A well-structured and detailed dataset will allow you to build a more accurate demand prediction model. You should aim to collect historical data for an extended period to capture long-term trends and seasonality. Additionally, the quality of data, including the accuracy of timestamps and attributes, is critical for model performance.

Initialization Parameters

1. output - Set output variable/dependent variable

2. dates.features - Set datetime field names (optional)

For example:

1. output = 'column_name'
2. dates.features = ['date_field_1','date_field_2']

In [3]:

Prep.info //data

General Preprocessing

missing_values	Display missing value report
remove_blank()	Remove empty/blank columns
correction()	Display correction heatmap
standardize()	Standardize entire dataset except dependent variable
encode_output()	Encode dependent feature/output variable
over_sample()	Oversample dataset. Parameters {'smote': SMOTE, 'ros': RandomOverSample}
clustering.apply()	Cluster dataset using elbow plot

General Template

```
from easypreprocessing import EasyPreProcessing
prep = EasyPreProcessing('filename_here.csv')
prep.df
prep.output = 'output_variable_here'
prep.remove_blank()
prep.missing_values
prep.categorical.impute()
prep.numerical.impute()
prep.categorical.encode()
prep.correction()
prep.standardize()
X_train, X_test, y_train, y_test = prep.split()
```

Categorical Preprocessing

categorical.fields	Display all categorical field names
categorical.unique	Display unique/distinct categorical values
categorical.impute()	Handle categorical missing values. Parameters {'mean', 'medium', 'mode'}
categorical.encode()	Encode categorical features. Parameters {'le': LabelEncoding, 'ohe': OneHotEncoding}

Numerical Preprocessing

numerical.fields	Display all numerical field names
numerical.impute()	Handle numerical missing values. Parameters {'mean', 'medium', 'mode'}

Summing units_sold group by key.

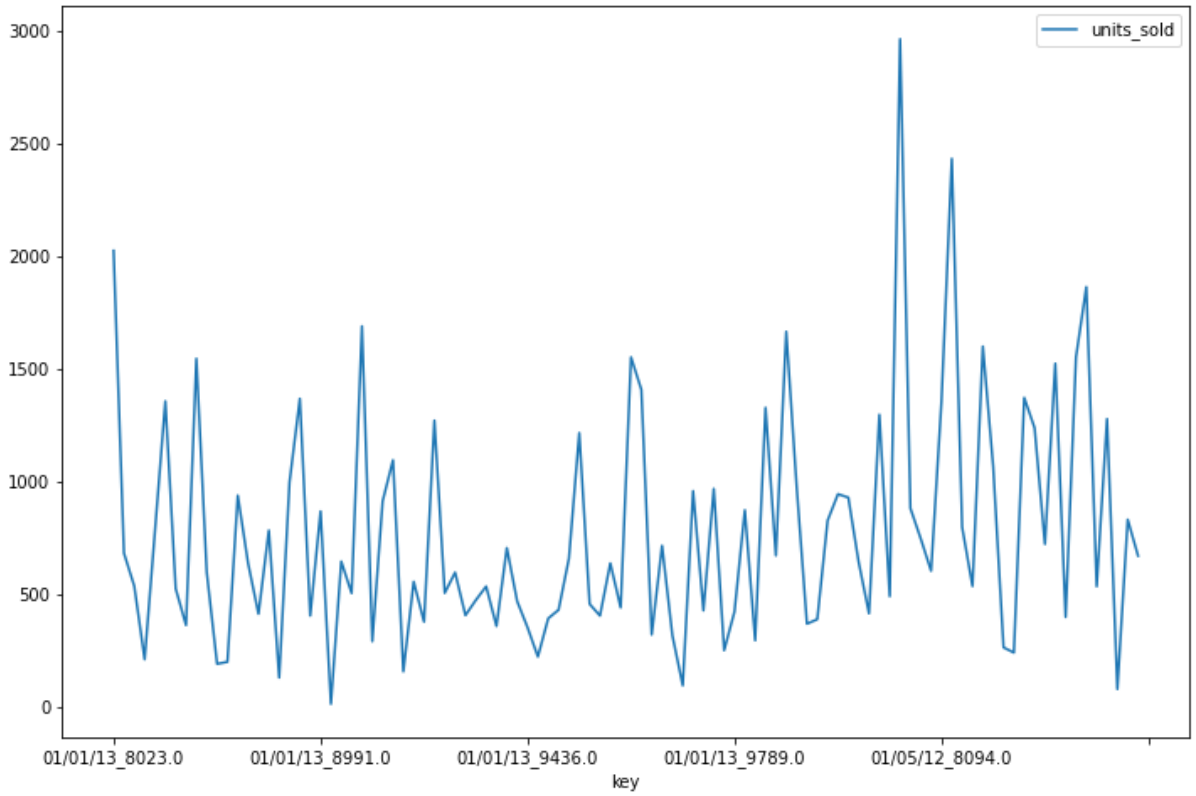
```
prep.dataset = prep.df.groupby('key').sum()
```

```
prep.df
```

key	units_sold
01/01/13_8023.0	2025.0
01/01/13_8058.0	682.0
01/01/13_8063.0	535.0
01/01/13_8091.0	210.0
01/01/13_8094.0	782.0
...	...
31/10/11_9890.0	531.0
31/10/11_9909.0	551.0
31/10/11_9954.0	431.0
31/10/11_9961.0	820.0
31/10/11_9984.0	506.0

9880 rows × 1 columns

Sample output:



Sample code using python:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

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

```
# Initialize and train the model
model = RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

Product Demand Prediction Model

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

```
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:

```
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 input the features (Total Price, Base Price) into the model and predict how much quantity can be demanded based on those values:

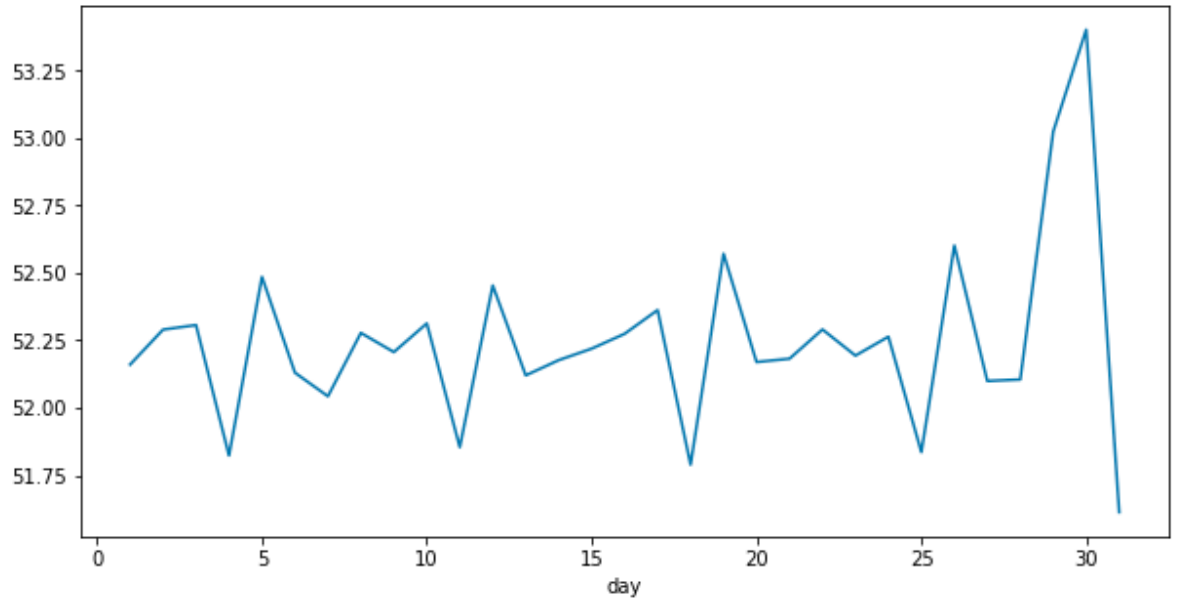
```
#features = [["Total Price", "Base Price"]]
features = np.array([[133.00, 140.00]])
model.predict(features)
```

array([27.])

Feature Engineering

There are times when multiple features are provided in the same feature or we have to derive some features from the existing ones. We will also try to include some extra features in our dataset so, that we can derive some interesting insights from the data we have. Also if the features derived are meaningful then they become a deciding factor in increasing the model's accuracy significantly.

```
parts = df["date"].str.split("-", n = 3, expand = True)
df["year"]= parts[0].astype('int')
df["month"]= parts[1].astype('int')
df["day"]= parts[2].astype('int')
df.head()
```



	date	store	item	sales	year	month	day
0	2013-01-01	1	1	13	2013	1	1
1	2013-01-02	1	1	11	2013	1	2
2	2013-01-03	1	1	14	2013	1	3
3	2013-01-04	1	1	13	2013	1	4
4	2013-01-05	1	1	10	2013	1	5

SUMMARY:

Creating a product demand prediction project with machine learning involves several steps, from data collection and preprocessing to model development and deployment. Below is an outline of the project, along with some sample Python code using popular libraries like Scikit-Learn and Pandas.

Project Outline:

1. Data Collection:

- Gather historical data on the product, including attributes such as price, promotions, seasonality, and historical sales. This data can be obtained from various sources like databases, CSV files, or APIs.

2. Data Preprocessing:

- Clean and prepare the data for analysis and modeling. This may involve handling missing values, encoding categorical variables, and feature engineering.

3. Exploratory Data Analysis (EDA):

- Explore the dataset to gain insights into the product's demand trends. Visualize data and identify correlations between features and demand.

4. Feature Selection:

- Identify which features are most relevant to demand prediction. You can use techniques like feature importance or correlation analysis.

5. Model Development:

- Choose a machine learning algorithm for demand prediction. Common choices include linear regression, decision trees, random forests, or gradient boosting.

6. Model Evaluation:

- Assess the model's performance using appropriate evaluation metrics like Mean Squared Error (MSE) or Root Mean Squared Error (RMSE).

7. Hyperparameter Tuning:

- Fine-tune the model's hyperparameters to optimize its performance.

8. Model Deployment:

- Deploy the model for real-time or batch predictions, depending on your use case. This can be done using web APIs, cloud services, or containerization tools like Docker.

9. Continuous Monitoring:

- Regularly monitor the model's performance and retrain it with new data as needed to ensure accurate demand predictions.

the actual implementation may require more detailed work, especially when dealing with large and complex datasets. Additionally, you should choose the appropriate algorithms and techniques based on your specific use case and data characteristics.

In a product demand prediction project, the sample output typically includes the predicted demand for a set of products or items based on historical data and the model we've developed. The output can be in various formats depending on our project's goals. Here's a simplified example:

Suppose we have developed a demand prediction model for a specific product, and we want to predict its demand for the next month.

Accuracy Metrics:

- Mean Absolute Error (MAE): 4.2 units

- Mean Squared Error (MSE): 22.1 units^2

- Root Mean Squared Error (RMSE): 4.7 units

Recommendations:

for us to meet the expected demand, consider increasing inventory for the first week of November and optimizing pricing and promotions during the same period.

We as a team recommend:

- Python
- Jupyter Notebook
- scikit-learn

Step 1: Data Generation In this example, we'll generate a synthetic dataset using Python's NumPy library. We can replace this with your real data.

```
import numpy as np
```

```
import pandas as pd



# Create a synthetic dataset

np.random.seed(0)

n_samples = 100

price = np.random.rand(n_samples) * 100

advertising_spend = np.random.rand(n_samples) * 10

demand = 30 * price - 5 * advertising_spend +
np.random.randn(n_samples) * 10



data = pd.DataFrame({'Price': price, 'Advertising': advertising_spend,
'Demand': demand})
```

Step 2: Data Exploration Wevcan explore the data using libraries like Pandas and Matplotlib to understand its characteristics and relationships.

Step 3: Data Preprocessing Preprocess the data as needed. In this example, the data is already generated, so there might not be much preprocessing necessary.

Step 4: Model Building Now, let's build a linear regression model to predict demand based on price and advertising spend.

```
python

Copy code


from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error



# Split the data into training and testing sets

X = data[['Price', 'Advertising']]

y = data['Demand']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)



# Create and train the model

model = LinearRegression()

model.fit(X_train, y_train)
```

Step 5: Model Evaluation Evaluate the model's performance using appropriate metrics.

```
python

Copy code


# Make predictions

y_pred = model.predict(X_test)



# Calculate mean squared error

mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
```

Step 6: Prediction We use the trained model to make predictions. For example, given a price of $60 and advertising spend of $3:

```
python
Copy code

predicted_demand = model.predict([[60, 3]])
print(f'Predicted Demand: {predicted_demand[0]:.2f}')
```

Output: The project will produce the Mean Squared Error as a measure of model performance and a predicted demand value for a specific set of input features.

This is a simplified example for educational purposes. In a real-world scenario, you would need to work with real data, handle more complex preprocessing, and potentially use more advanced machine learning algorithms for better prediction accuracy.

Sample Output:

```
yaml
Copy code

Product: ABC123
Prediction Date: 2023-11-01

Historical Demand:
```

- 2023-10-01: 100 units

- 2023-10-02: 105 units

- 2023-10-03: 110 units

...

Predicted Demand for November 2023:

- 2023-11-01: 115 units

- 2023-11-02: 120 units

- 2023-11-03: 118 units

...

Explanation:

The demand prediction for product ABC123 in November 2023 is based on historical data, pricing, promotions, and seasonality factors. The model forecasts an increase in demand due to a marketing campaign running in early November.

Accuracy Metrics:

- Mean Absolute Error (MAE): 4.2 units

- Mean Squared Error (MSE): 22.1 units^2

- Root Mean Squared Error (RMSE): 4.7 units