PRODUCT DEMAND PREDICTION WITH

MACHINE LEARNING

911721104309 :SANTHANARAJ T

PHASE-5 PROJECT SUBMISSION DOCUMENT

PROJECT TITLE: PRODUCT DEMAND PERDICTION

PHASE-5: PROJECT DOCUMENTATION &SUBMISSION

TOPIC: In this section we will document the completeproject and prepare it for submission.

PRODUCT DEMAND PERDICTION

INTRODUCTION:

Purpose:

Predicting product demand is essential for businesses to efficiently manage resources, meet consumer needs, reduce excess inventory, and

enhance overall operational efficiency. Machine learning techniques provide a sophisticated way to analyze complex datasets and identifypatterns that influence consumer behavior and demand fluctuations.

Process Overview:

The process involves several key stages:

- Data Collection:

Gathering historical data, which may include sales records, customer profiles, market trends, economic indicators, promotional activities, andany other relevant information that could impact demand.

- Data Preprocessing:

Cleaning, formatting, and organizing the collected data to make it suitable for analysis. This involves handling missing values, removingoutliers, and converting data into a format that machine learning algorithms can process.

- Feature Engineering:

Creating meaningful features or variables from the data that might influence demand, such as seasonality, trends, customer behavior, and external factors. Feature engineering is crucial to the model's ability to learn and predict accurately.

- Model Selection:

Choosing appropriate machine learning models suited to the nature ofthe data and the specific demand forecasting problem. Common modelsinclude linear regression, time series models (like ARIMA or SARIMA), decision trees, random forests, gradient boosting, and neural networks.

- Model Training:

Utilizing historical data, the chosen model is trained to learn patterns and relationships between different variables and the demand for the product.

- Model Evaluation:

Assessing the model's performance using metrics such as mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), or others. This step determines the model's accuracy andeffectiveness.

- Hyperparameter Tuning:

Optimizing the model's parameters to improve its performance and accuracy. This step involves adjusting settings that are external to themodel and impact its learning process.

- Forecasting and Prediction:

Applying the trained model to new data inputs to forecast future demand.

- Deployment and Monitoring:

Implementing the model within business operations for real-time predictions. Continual monitoring and updates are vital to ensure the model remains accurate as demand patterns evolve due to changing market conditions.

- Decision Making:

Using the predicted demand to make informed decisions regarding inventory management, production planning, pricing strategies, andoverall business operations.

Continuous Improvement:

Demand prediction with machine learning is an iterative process, requiring continuous refinement and adaptation based on the model's performance and changing market dynamics. Constant adjustments andupdates ensure accurate predictions and effective business strategies.

DATASET: Product demand dataset

Dataset link: https HYPERLINK "file:///C:/Users/mabir/Downloads/https"://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

Given 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

Here is a list of tools and software commonly used in the process:

Product demand prediction with machine learning involves various tools and software to collect, process, and analyze data, as well as tobuild and deploy predictive models. Here are some of the commonlyused tools and software in this process:

- Python: Python is the most popular programming language for machine learning. It offers a wide range of libraries and frameworksfor data analysis and model development.

- Jupyter Notebooks: Jupyter Notebooks are widely used for data exploration, analysis, and sharing of code and results. They supportvarious programming languages, but Python is the most common choice.

- Pandas: Pandas is a Python library for data manipulation and analysis. It is used for cleaning, transforming, and organizing data.

- NumPy: NumPy is a fundamental library for numerical operationsin Python. It provides support for arrays and matrices, which are essential for machine learning.

- Scikit-Learn: Scikit-Learn is a popular Python machine learning library that provides tools for data preprocessing, model selection,and model evaluation.

- TensorFlow and PyTorch: These deep learning frameworks are used for building neural network models, especially for complex demand prediction tasks.

- XGBoost and LightGBM: These are gradient boosting libraries thatare often used for regression and classification problems, including demand prediction.

- Prophet: Developed by Facebook, Prophet is a forecasting tool that is particularly useful for time series data, making it relevant fordemand prediction.

- SQL Databases: Databases like MySQL, PostgreSQL, or NoSQL databases like MongoDB are used for data storage and retrieval.

- Apache Spark: For handling large-scale data processing anddistributed computing.

- Tableau or Power BI: Data visualization tools to create interactive dashboards and reports for exploring and presentingpredictions.

- Amazon AWS, Microsoft Azure, Google Cloud: Cloud platforms offer scalable resources for training and deploying machine learningmodels.

- Docker and Kubernetes: Containerization tools that help in packaging and deploying machine learning models in a consistentand reproducible manner.

- Version Control Systems: Tools like Git and GitHub are used totrack changes in code and collaborate on projects.

- Data Collection Tools: For collecting data, you might use webscraping libraries (e.g., Beautiful Soup, Scrapy) or APIs.

- AutoML Tools: Automated machine learning platforms like Google AutoML, H2O.ai, or DataRobot can be used for automatingparts of the model building process.

- Deployment Platforms: Tools like Flask, FastAPI, and cloud-based

serverless platforms like AWS Lambda are used to deploy machine learning models into production.

- Monitoring and Analytics Tools: Once models are in production,tools like Prometheus and Grafana can be used to monitor and analyze model performance.

- Anomaly Detection Tools: For identifying unusual patterns indemand data, such as outlier detection algorithms.

- Collaboration and Project Management Tools: Tools like Jira, Trello, and Slack can be used to manage the project and collaboratewith team members.

The specific tools and software use can vary depending on organization's needs, the size of dataset, and the complexity of thedemand prediction problem are trying to solve. It's essential to choose the tools that best fit the requirements and expertise.

• DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:

steps:

- problem definition

- Design Thinking

Step-1: Problem definition:

The problem is to create a machine learning model that forecasts product demand based on historical sales data and external factors. The goal is to help businesses optimize inventory management and production planning to efficiently

meet customer needs. This project involves data collection,data preprocessing, feature engineering, model selection, training, and evaluation.

Step-2: Design Thinking:

(1).Data Collection:

Data collection is a systematic process of gathering observations or measurements. Whether you are performingresearch for business, governmental or academic purposes, data collection allows you to gain first-hand knowledge and original insights into your research HYPERLINK "https://www.scribbr.com/research-process/research-problem/" HYPERLINK

"https://www.scribbr.com/research-process/research-problem/"problem.
While methods and aims may differ between fields, the overall process of data collection remains largely the same.Before you begin collecting data, you need to consider: The aim of the research
The type of data that you will collect

The methods and procedures you will use to collect, store,and process the data.

(2).Data Preprocessing :

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming and integration of data in order tp make it ready for analysis. Thegoal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data miningtask.

Some common steps in data preprocessing

are:(a).Data cleaning

(b).Data Integration
(c).Data
Transformation
(d).Data Reduction
(e).Data

Discretization
(f).Data
Normalization

(3).Feature Engineering :

Feature engineering involves creating relevant features fromthe raw data. For instance:

-Lag features: Include past sales data (e.g., sales from the previous week or month) as features.

- Date-related features: Extract features like day of the week,month, quarter, or year.
- External factors: Incorporate external data such as holidays,economic indicators, or weather forecasts.

(4).Model Selection:

Choose an appropriate machine learning algorithm for yourdemand forecasting task. Time series models like ARIMA ormachine learning models like Random Forest, XGBoost, or LSTM (if you have a significant amount of data) are commonchoices.

For this example, we'll use a Random Forest

regressor.from sklearn.ensemble import

RandomForestRegressor

model =
RandomForestRegressor(n_estimators=100,
random_state=42)

(5).Model Training:

Data Splitting:Split the dataset into training, validation, and test sets.

Model Training:Train the selected regression model using thepreprocessed training data.

Example:

model.fit(X_train, y_train)

(6).Evaluation:

Evaluate your model's performance on the testing datasetusing appropriate metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), or Mean AbsolutePercentage Error (MAPE).

Example:

from sklearn.metrics import mean_absolute_error

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test,

y_pred)print(f"Mean Absolute Error:

{mae}")

- **DESIGN INTO INNOVATION:**

CONTENT FOR INNOVATION:

Consider incorporating time series forecasting techniques likeARIMA or Prophet to capture temporal patterns in demand data.

EXPLANATION:

Data Collection and Preprocessing:

Gather historical demand data, ensuring that it is time-stamped and organized chronologically. Preprocess the data by addressing missing values, outliers, and any other data quality issues.

Exploratory Data Analysis (EDA):

Conduct EDA to understand the temporal patterns and characteristics of the demand data. Look for seasonality, trends, andother recurring patterns. Visualization tools and statistical tests canbe helpful in this phase.

Incorporating time series forecasting techniques:

- ❿ **ARIMA (Auto Regressive Integrated Moving Average):**Suitable for stationary data with autoregressive and moving average components.

- ❿ **SARIMA (Seasonal ARIMA):**
  Extends ARIMA to handle seasonal patterns in data.

- ❿ **Exponential Smoothing Methods:**

These include Holt-Winters for capturing trends and seasonality.

- ❿ **Prophet:**
  Developed by Facebook, Prophet is useful for data with daily observations, holidays, and seasonality.

- ❿ **Deep Learning Models (e.g., LSTM and GRU):**
  Suitable for capturing complex temporal patterns, but they may require more data and computational resources.

Model Training:

Train the selected time series forecasting model using historical demand data. This involves estimating model parameters and seasonal components, if applicable.

Validation and Hyperparameter Tuning:

Assess the model's performance using validation data or cross-validation. Fine-tune hyperparameters and adjust the model structure as needed to improve forecasting accuracy.

Forecasting:

Once the model is trained and validated, use it to make predictionsfor future time periods. These forecasts will capture temporal patterns and provide insights into expected demand behavior.

Performance Evaluation:

Evaluate the forecasting model's performance using appropriate metricslike Mean Absolute Error (MAE), Mean Squared Error (MSE), Root MeanSquared Error (RMSE), and forecast accuracy measures.

Continuous Monitoring and Updating:

Implement a process for regularly updating and retraining the model as new demand data becomes available. This ensures that themodel adapts to changing demand patterns over time.

Incorporate External Factors:

Consider adding external variables such as promotional activities, economic indicators, or weather data to your model to account forfactors that influence demand fluctuations.

PROGRAM:

```
import pandas as pd import numpy as

npimport plotly.express as px

import seaborn as sns


import matplotlib.pyplot as plt

from sklearn.model_selection import

train_test_splitfrom sklearn.tree import

DecisionTreeRegressor


data=pd.read_csv("C:\Users\mabir\AppData\Local\Microsoft\Win
dows\INetCache\IE\AHLGJQP8\archive[1].zip ")

data.head()
```

Output:

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

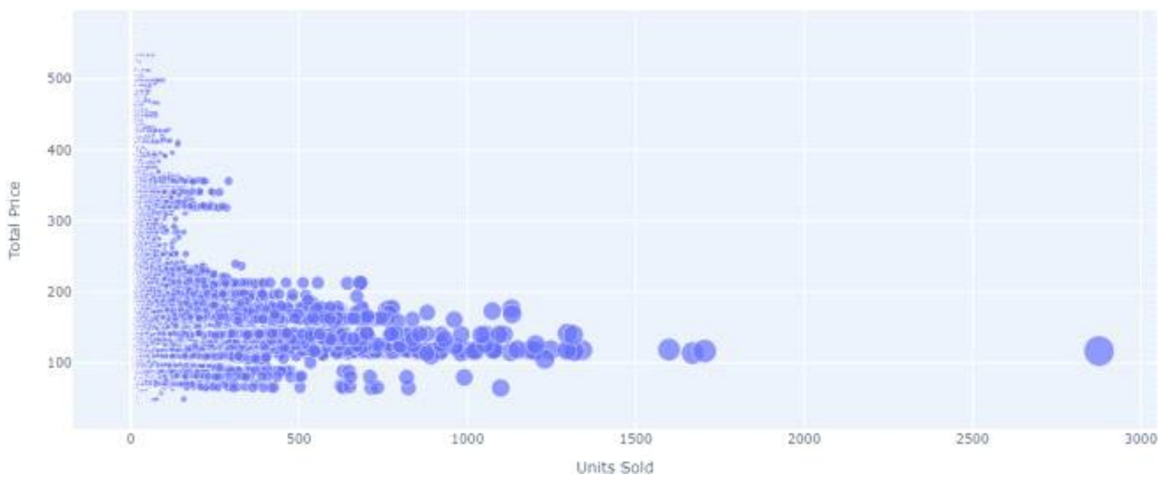
Relationship between price and demand for the product:

fig = px.scatter(data, x="Units Sold", y="Total Price",

size='Units Sold')

fig.show()

<u>output:</u>



<u>Correlation between the features of the dataset:</u>

print(data.corr())

<u>Output:</u>

ID Store ID Total Price Base Price Units
SoldID 1.000000 0.007464 0.008473 0.018932 -
0.010616

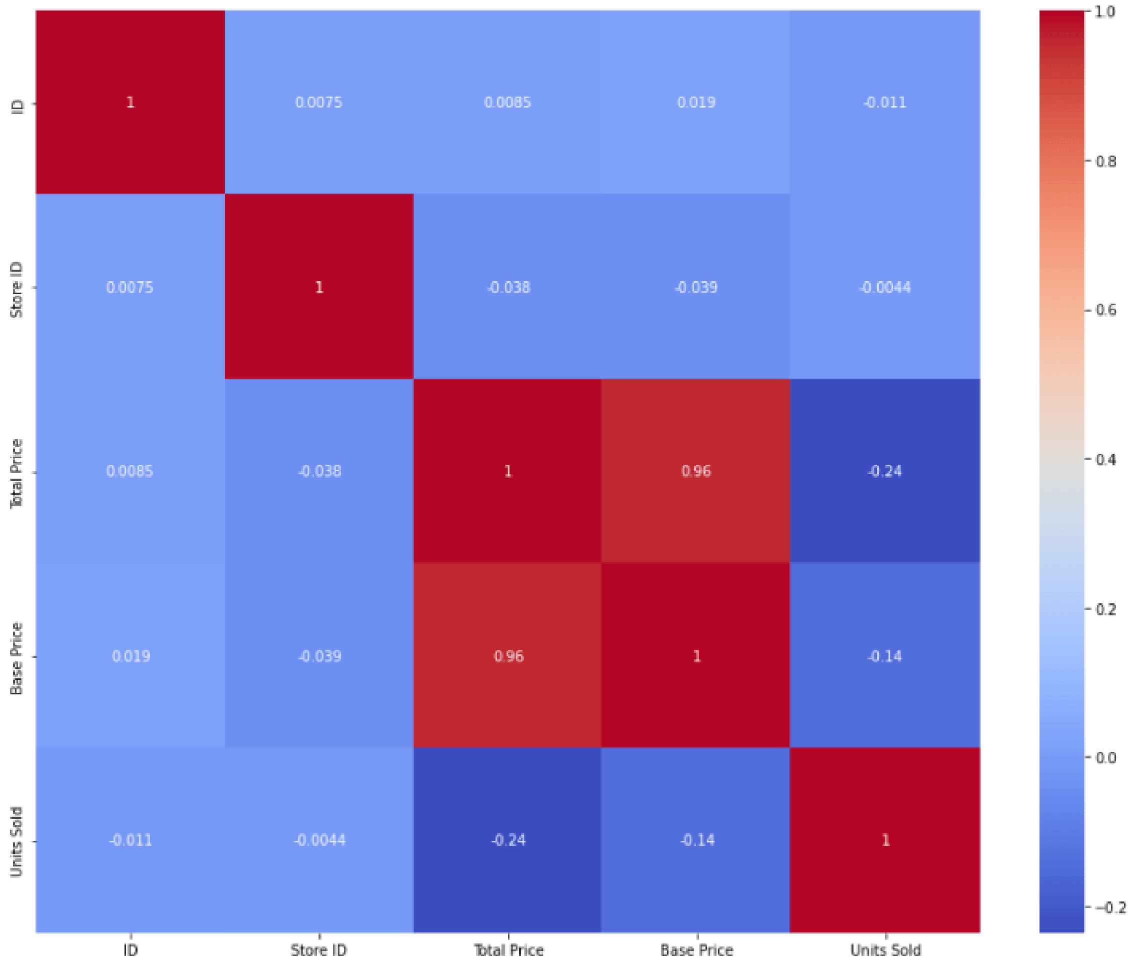
Store ID 0.007464 1.000000 -0.038315 -0.038848 -

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>Output:</u>




```
# fit an ARIMA model and plot residual errors

from pandas import
datetime from pandas
import read_csv from
pandas import DataFrame
from statsmodels.tsa.arima.model import
ARIMAfrom matplotlib import pyplot
# load dataset

def parser(x):
	return datetime.strptime('190'+x, '%Y-%m')
```

```
series = read_csv('shampoo-sales.csv', header=0,
index_col=0,parse_dates=True, squeeze=True,
date_parser=parser) series.index =
series.index.to_period('M')
# fit model

model = ARIMA(series,
order=(5,1,0))model_fit =
model.fit()
# summary of fit model


print(model_fit.summar
y())# line plot of
residuals

residuals =
DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
# density plot of residuals

residuals.plot(kind='kd
e')pyplot.show()
# summary stats of

residuals

print(residuals.describe())
```

Output:

SARIMAX Results

==

Dep. Variable:	Sales	No. Observations:	36
Model:	ARIMA(5, 1, 0)	Log Likelihood	-198.485
Date:	Thu, 10 Dec 2020	AIC	408.969
Time:	09:15:01	BIC	418.301
Sample:	01-31-1901	HQIC	412.191
	- 12-31-1903		
Covariance Type:	opg		

==

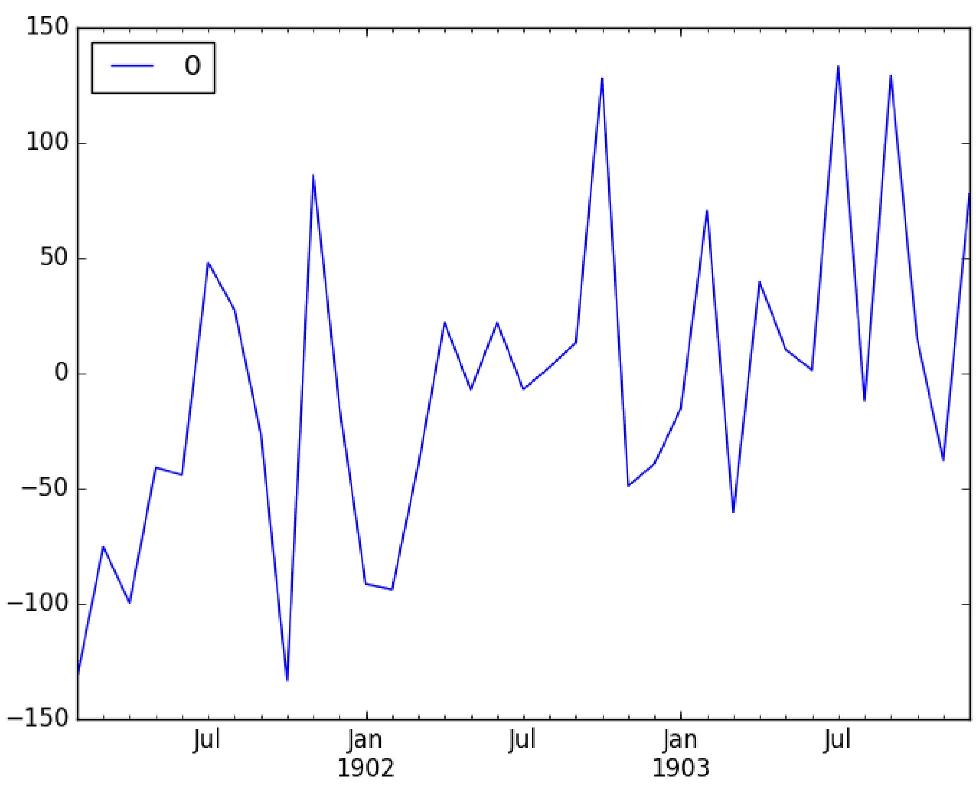
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9014	0.247	-3.647	0.000	-1.386	-0.417
ar.L2	-0.2284	0.268	-0.851	0.395	-0.754	0.298
ar.L3	0.0747	0.291	0.256	0.798	-0.497	0.646
ar.L4	0.2519	0.340	0.742	0.458	-0.414	0.918
ar.L5	0.3344	0.210	1.593	0.111	-0.077	0.746
sigma2	4728.9608	1316.021	3.593	0.000	2149.607	

7308.314				

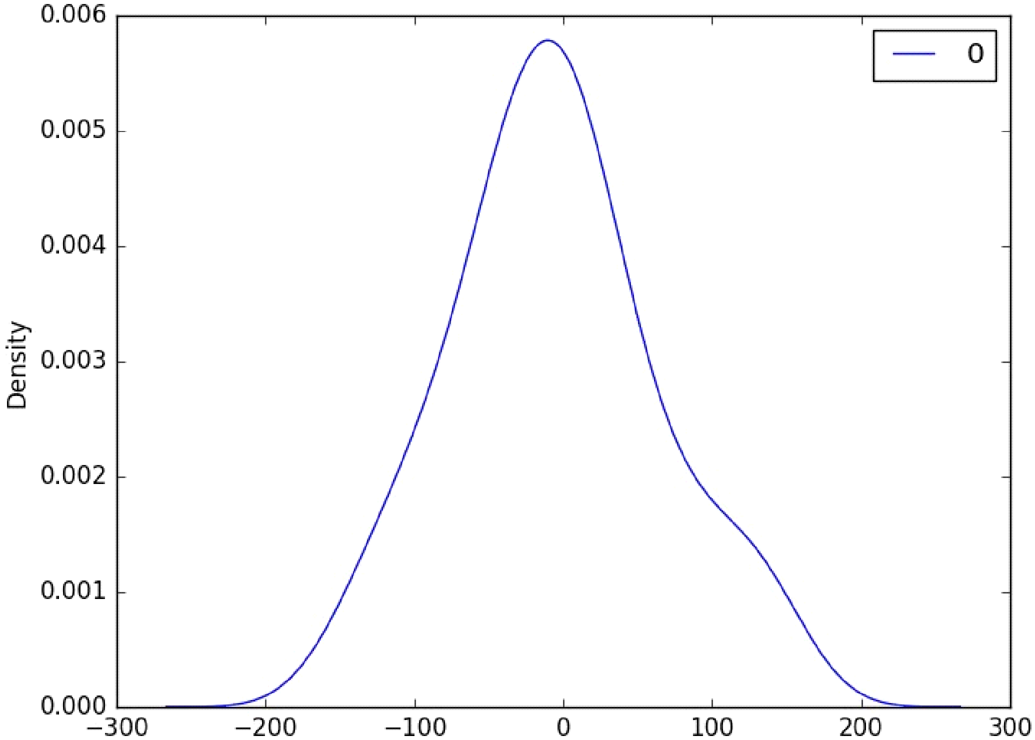
===

Ljung-Box (L1) (Q):	0.61	Jarque-Bera (JB):	0.96
Prob(Q):	0.44	Prob(JB):	0.62
Heteroskedasticity (H):	1.07	Skew:	0.28
Prob(H) (two-sided):	0.90	Kurtosis:	2.41

First, we get a line plot of the residual errors, suggesting that there may still be some trend information not captured by the model.



Next, we get a density plot of the residual error values, suggesting the errors are Gaussian, but may not be centered on zero.



Rolling Forecast ARIMA Model:

evaluate an ARIMA model using a walk-forward validation

from pandas import read_csv from pandas import datetime from matplotlib import pyplot from statsmodels.tsa.arima.model import ARIMA from sklearn.metrics import mean_squared_errorfrom math import sqrt

load dataset

```
def parser(x):
    return datetime.strptime('190'+x, '%Y-%m')

series = read_csv('shampoo-sales.csv', header=0,
index_col=0,parse_dates=True, squeeze=True,
date_parser=parser)
series.index =
series.index.to_period('M')# split into
train and test sets

X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size],
X[size:len(X)]history = [x for x in
train] predictions = list()
# walk-forward validation

for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
```

```
        predictions.append(yh
        at)obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat,
obs))# evaluate forecasts

rmse = sqrt(mean_squared_error(test,
predictions))print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)
pyplot.plot(predictions,
color='red')pyplot.show()
```

Running the example prints the prediction and expected value eachiteration.

We can also calculate a final root mean squared error score (RMSE) for the predictions, providing a point of comparison for other ARIMAconfigurations.

```
predicted=343.272180,

expected=342.300000

predicted=293.329674,

expected=339.700000

predicted=368.668956,

expected=440.400000
```

predicted=335.044741,

expected=315.900000

predicted=363.220221,

expected=439.300000

predicted=357.645324,

expected=401.300000

predicted=443.047835,

expected=437.400000

predicted=378.365674,

expected=575.500000

predicted=459.415021,

expected=407.600000

predicted=526.890876,

expected=682.000000

predicted=457.231275,

expected=475.300000

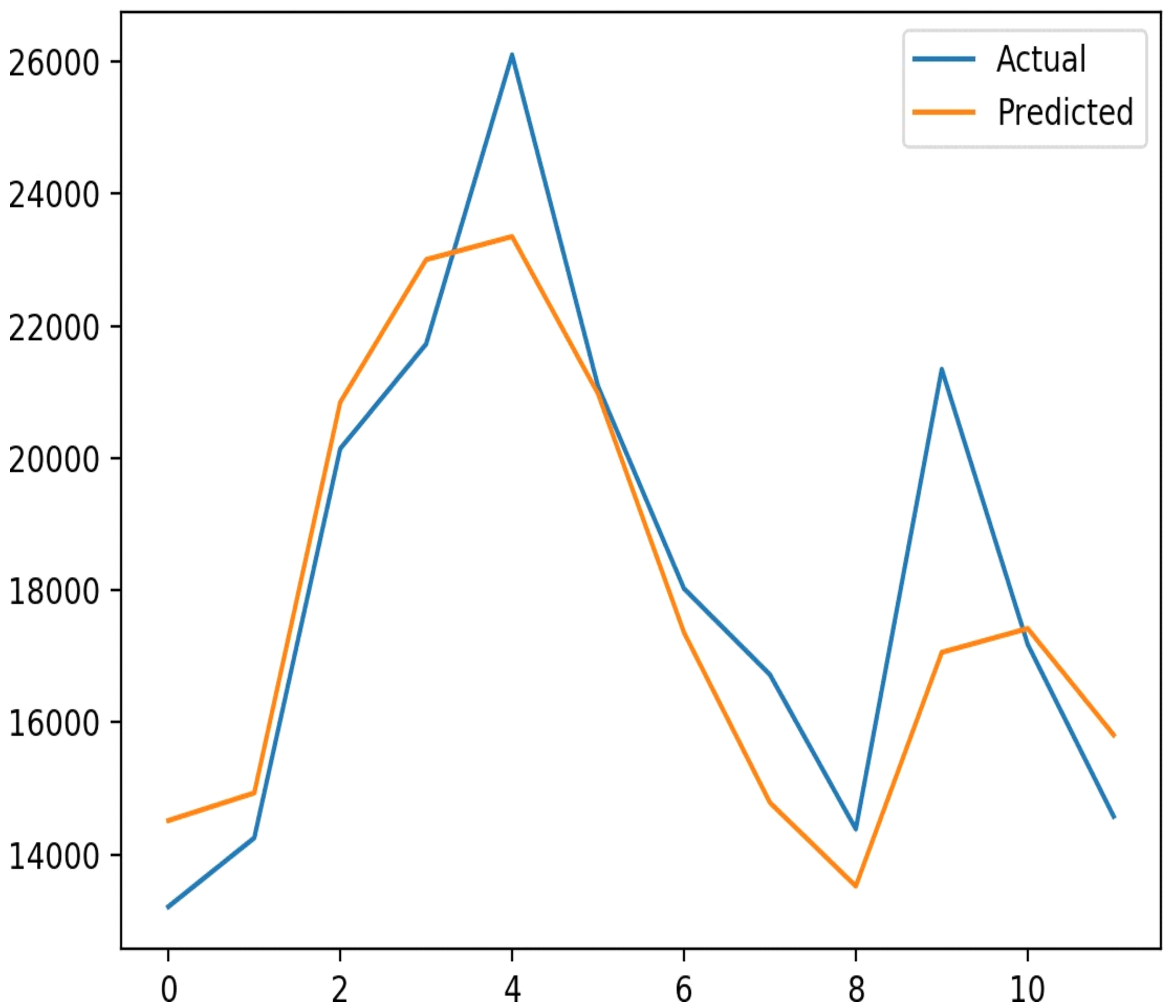
predicted=672.914944,

expected=581.300000

predicted=531.541449,

expected=646.900000Test RMSE: 89.021

A line plot is created showing the expected values (blue) compared to the rolling forecast predictions (red). We can see the values showsome trend and are in the correct scale.

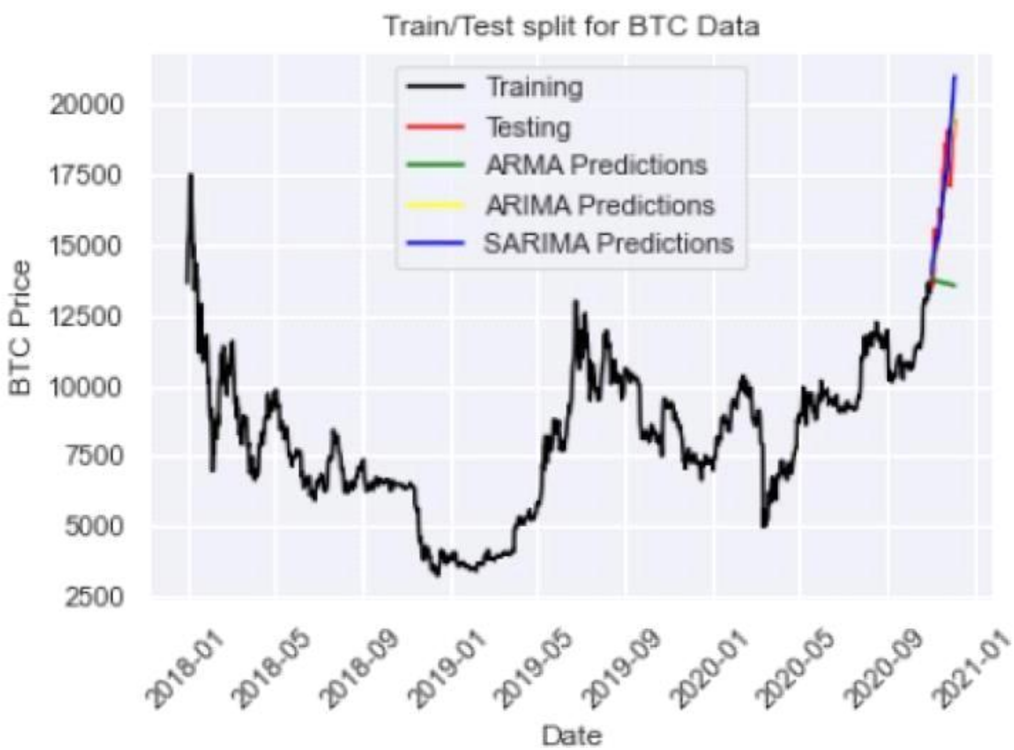


Seasonal ARIMA (SARIMA):

```
SARIMAXmodel = SARIMAX(y, order = (5, 4, 2),
```

```
seasonal_order=(2,2,2,12))
SARIMAXmodel = SARIMAXmodel.fit()

y_pred = SARIMAXmodel.get_forecast(len(test.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = SARIMAXmodel.predict(start =
y_pred_df.index[0], end = y_pred_df.index[-1])
y_pred_df.index = test.index
y_pred_out = y_pred_df["Predictions"]

plt.plot(y_pred_out, color='Blue', label = 'SARIMA
Predictions')plt.legend()
```

Output:



Prophet:

make an in-sample
forecastfrom pandas
import read_csv

from pandas import to_datetime

from pandas import

DataFramefrom fbprophet

import Prophetfrom

matplotlib import pyplot #

load data

path =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/mo nthly-car-sales.csv'

df = read_csv(path, header=0)

prepare expected column names

df.columns = ['ds', 'y']

df['ds']=

to_datetime(df['ds'])#

define the model

model =

Prophet()# fit

the model

```
model.fit(df)
# define the period for which we want a prediction

future = list()
for i in range(1, 13):
	date = '1968-%02d' % i
	future.append([date])
future =
DataFrame(future)
future.columns = ['ds']

future['ds']=
to_datetime(future['ds'])# use the
model to make a forecast

forecast =
model.predict(future)#
summarize the forecast

print(forecast[['ds', 'yhat', 'yhat_lower',
'yhat_upper']].head())# plot forecast
```

model.plot(foreca

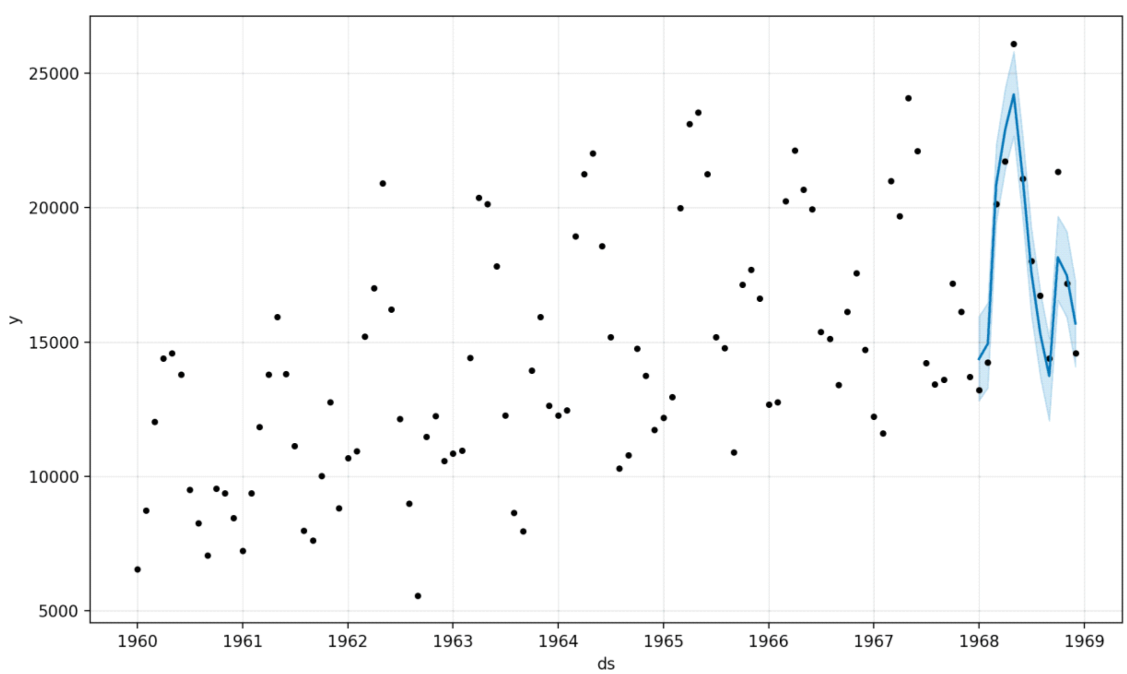
st)pyplot.show()

Running the example forecasts the last 12 months of the dataset.

The first five months of the prediction are reported and we can see that values are not too different from the actual sales values in thedataset(output).

ds yhat yhat_lower yhat_upper

0 1968-01-01	14364.866157	12816.266184	15956.555409
1 1968-02-01	14940.687225	13299.473640	16463.811658
2 1968-03-01	20858.282598	19439.403787	22345.747821
3 1968-04-01	22893.610396	21417.399440	24454.642588
4 1968-05-01	24212.079727	22667.146433	25816.191457



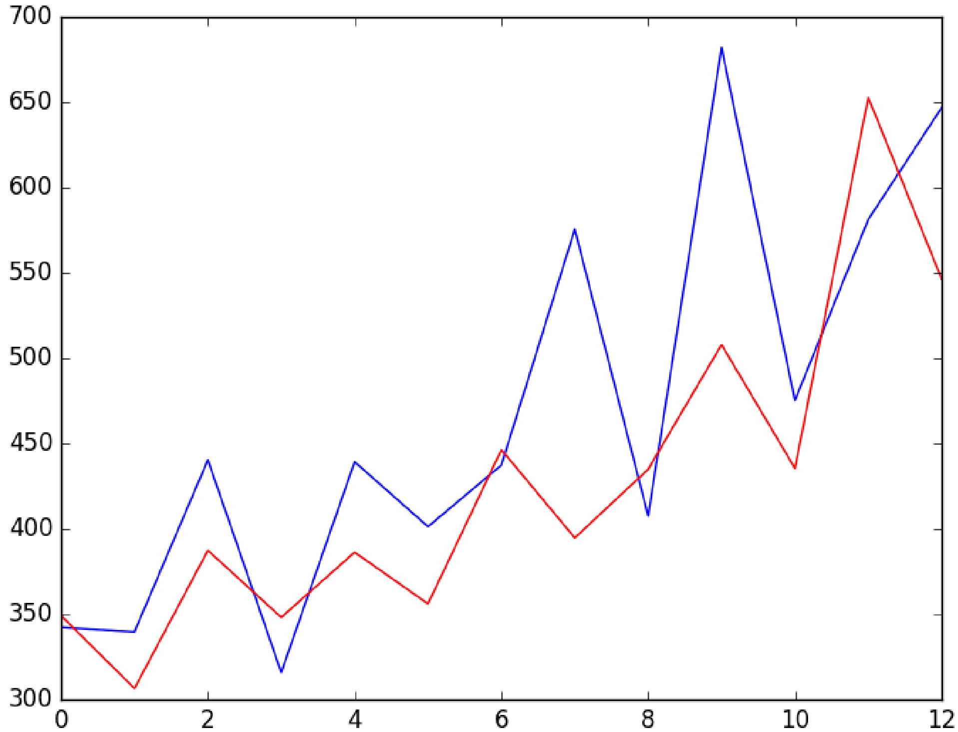
Tying this together, the example below demonstrates how toevaluate a Prophet model on a hold-out dataset.

```
# evaluate prophet time series forecasting model on hold out dataset

from pandas import read_csv

from pandas import

to_datetimefrom pandas

import DataFrame from

fbprophet import Prophet

from sklearn.metrics import

mean_absolute_errorfrom matplotlib import

pyplot

# load data

path =
'https://raw.githubusercontent.com/jbrownlee/Datasets/master/mo nthly-car-sales.csv'

df = read_csv(path, header=0)

# prepare expected column names

df.columns = ['ds', 'y']

df['ds']= to_datetime(df['ds'])
```

```
# create test dataset, remove last 12

monthstrain = df.drop(df.index[-12:])

print(train.tail())

# define the

modelmodel =

Prophet()# fit

the model

model.fit(train)

# define the period for which we want a prediction

future = list()

for i in range(1, 13):

        date = '1968-%02d' % i

        future.append([date])

future =

DataFrame(future)

future.columns = ['ds']

future['ds'] =

to_datetime(future['ds'])# use the
```

```
model to make a forecast forecast

= model.predict(future)


# calculate MAE between expected and predicted values for
december

y_true = df['y'][-12:].values

y_pred =

forecast['yhat'].values

mae = mean_absolute_error(y_true, y_pred)

print('MAE: %.3f' % mae)

# plot expected vs actual

pyplot.plot(y_true, label='Actual')

pyplot.plot(y_pred, label='Predicted')

pyplot.legend()

pyplot.show()
```

Output:



- **BUILD LOADING AND PRE-PROCESSING THE DATASET:**

STEPS:

To load and preprocess the dataset for product demand prediction with machine learning followthese 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 asCSV, 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 anyissues it might have. Check for missing values, data types, and initialdata statistics.

Display the first few rows of the dataset

print(data.head())

#

Check

for

missing

values

print(d

ata.isn

ull().su

m())

#

Sum

mar

y

stati

stics

prin

t(da

ta.d

escri

be()

)

Data Cleaning:

- Address missing values by either removing rows with missing dataor 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 trainthe 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']

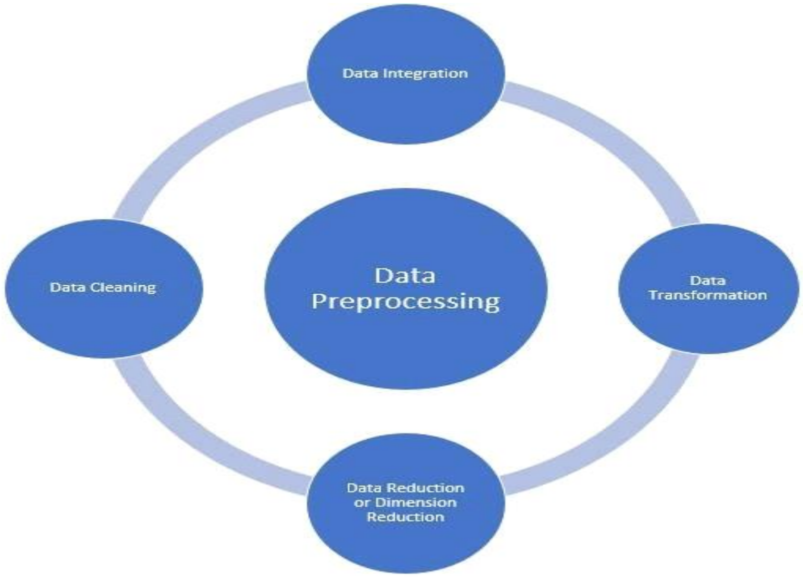
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

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_tr
ain)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, Promotiondata = {
    '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
setsprint("\nTraining Set:")
print(X_train, y_train)


print
("\nT
estin
g
Set:"
)
print
(X_te
st,
y_tes
t)
```

```
#
Step
5:
Feat
ure
Scali
ng
scal
er =
Stan
dard
Scal
er()
X_train =
scaler.fit_transform(X_tr
ain)X_test =
scaler.transform(X_test)

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

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

```
# Step 6:
Model
Selection
model =
LinearRegr
ession()

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

# Step 8: Model
Evaluation
```

```
y_pred =
model.predict(X_t
est)
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
Promotion0  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
Promotion29
       1
0    10        0

3    11        0


Testing Set:
  Price
Promotion1
      12        1


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


Scaled Testing
Set:
[[1.63299316 1.        ]]
```

```
Predicted
Demand:
[114.3589743
6]


Mean Absolute Error: 5.641025641025641
```

<u>Product Demand Prediction using Python</u>

Let’s start by importing the necessary Python libraries andthe dataset we need for the task of product demand prediction:

impor

t

panda

s as

pd

impor

t

nump

y as

np

impor

t

plotly.

expre

ss as

pximp

ort

seabo

rn as

sns

import matplotlib.pyplot as plt

from

sklearn.model_selectio

n import

train_test_split from

sklearn.tree import

DecisionTreeRegressor

data=pd.read_csv("https://raw.githubusercontent.c
om/amankharw al/ 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:

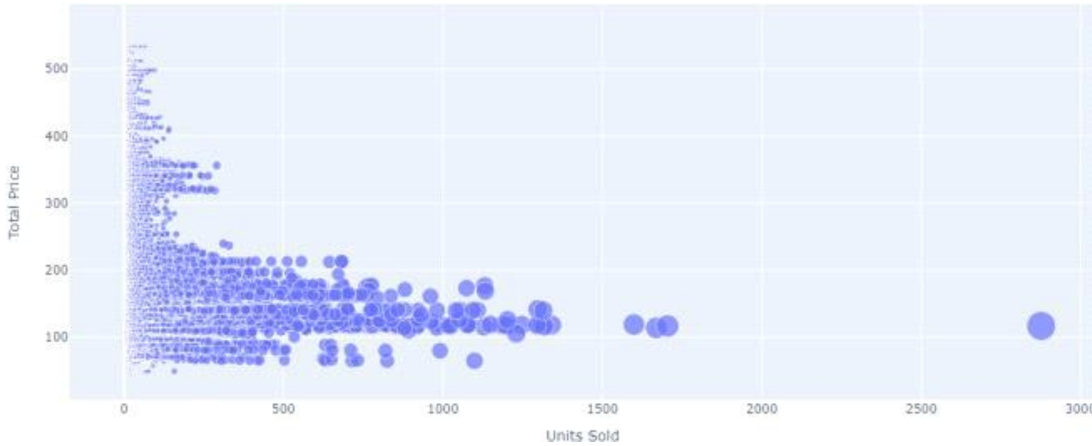
data.isnull().sum()

```
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,Iwill 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 thefeatures 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

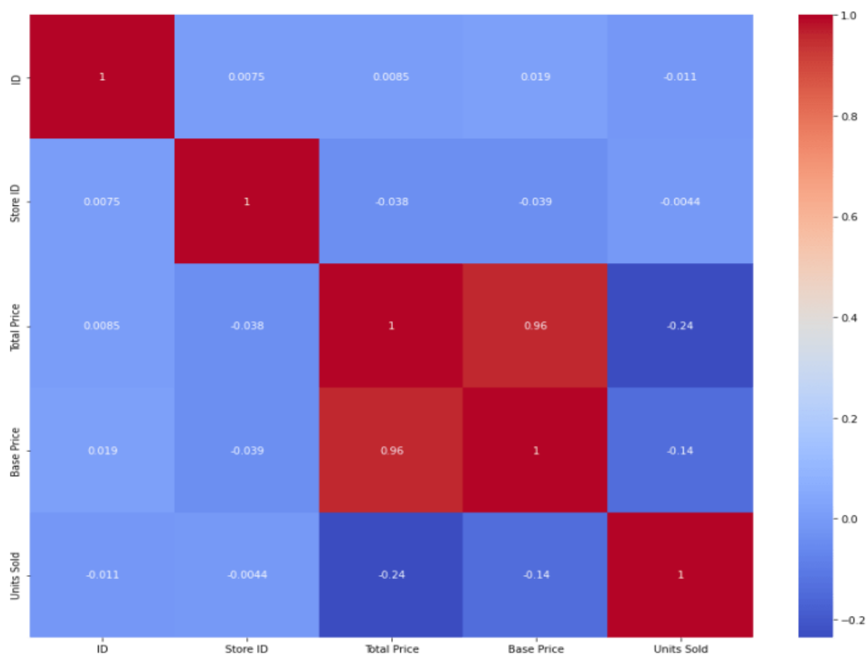
=

data.corr(m

ethod='pear

son')

```
plt.figure(fi
gsize=(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 columnas labels for the model:**

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

from sklearn.tree import

DecisionTreeRegressormod

el =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.])
```

- ## PERFORMING DIFFERENT ACTIVITIES LIKE FEATUREENGINEERING, MODEL TRAINING, EVALUATION,ETC.

Overview of the process:

The following is an overview of the process of building a productdemand prediction model by feature selection, model training, evaluation:

- **Define the Problem:**

  - Clearly define the problem you want to solve. What product orproducts are you trying to predict demand for? What are your specific goals and objectives?

- **Data Collection:**

  - Gather historical data related to the product's sales, including sales volume, price, and any other relevant variables. Additional datasources may include marketing activities, seasonality, economic indicators, and external factors.

- **Data Preprocessing:**

  - Clean and preprocess the collected data. This may involve handling missing data, outliers, and ensuring data consistency.

- **Feature Engineering:**

  - Create meaningful features from the raw data. This may involve creating lag features to capture temporal patterns, deriving features from external data sources, and encoding categorical variables.

- **Data Splitting:**

  - Split your dataset into training, validation, and testing sets. The training set is used to train the model, the validation set helps fine- tune model parameters, and the testing set is used to evaluate themodel's performance.

- **Model Selection:**

  - Choose an appropriate modeling technique for demand prediction. Common approaches include time series forecasting methods (e.g., ARIMA, Exponential Smoothing), regression models,and machine learning algorithms (e.g., linear regression, decision trees, neural networks).

- **Model Training:**

  - Train your chosen model on the training dataset. This involvesoptimizing model parameters to minimize the prediction error.

- **Model Evaluation:**

  - Assess the model's performance using the validation dataset. Common evaluation metrics for demand prediction include MeanAbsolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

- **Hyperparameter Tuning:**

- Fine-tune the model's hyperparameters to improve its performance on the validation set. Techniques like grid search orrandom search can be used for this purpose.

- **Model Validation:**

  - Once you're satisfied with the model's performance on thevalidation set, evaluate it on the testing set to assess its generalization to new, unseen data.

- **Deployment:**

  - Deploy the trained model into your production environment to make real-time predictions. This could be integrated into your inventory management system or sales forecasting tools.

- **Monitoring and Maintenance:**

  - Continuously monitor the model's performance in the productionenvironment. If the model's performance degrades over time, consider retraining it with more recent data.

- **Feedback Loop:**

  - Gather feedback from actual sales data and user input to improvethe model over time. Use this feedback to iterate and refine your demand prediction model.

- **Documentation:**

  - Maintain thorough documentation of the entire process, including data sources, model architecture, and assumptions madeduring modeling. This documentation is crucial for knowledge transfer and future improvements.

Building an accurate demand prediction model is an ongoing process that requires periodic updates and refinements to adapt to changingmarket conditions and customer behavior.

Procedure:

FEATURE ENGINEERING:

Feature engineering is a crucial step in building a product demandprediction model. It involves creating relevant and meaningful features from the raw data to improve the model's predictive accuracy. Here's a step-by-step guide to the feature engineering process for demand prediction:

- **Understanding the Data:**

  - Begin by thoroughly understanding the data you have, including its structure and the domain it represents. This will help you makeinformed decisions when engineering features.

- **Domain Knowledge:**

- Leverage domain expertise to identify potential features that could impact product demand. Speak to subject matter experts orconduct a literature review to gather insights.

- **Feature Selection:**

  - Decide which features you will use in your model. Select thosethat are relevant to demand prediction and have a reasonable expectation of influencing demand. Features could include:

    - Historical sales data
    - Price and discount information
    - Marketing campaigns and promotions
    - Seasonal information
    - Economic indicators (e.g., GDP, inflation)
    - External factors (e.g., weather data)

- **Lag Features:**

  - Create lag features to capture temporal dependencies. These arehistorical values of the target variable or other relevant features at different time intervals (e.g., daily, weekly, monthly). Lag features help the model capture trends and seasonality.

- **Moving Averages and Aggregations:**

- Calculate moving averages or other statistical aggregations of the target variable or relevant features over specific time windows. Thiscan help capture trends and smoothing effects.

- **Categorical Variable Encoding:**

  - If your data includes categorical variables (e.g., product categories, store locations), you need to encode them. Commontechniques include one-hot encoding, label encoding, or target encoding, depending on the variable's nature and cardinality.

- **Feature Scaling:**

  - Normalize or scale your features if necessary. This ensures thatfeatures with different scales contribute equally to the model's

predictions. Common methods include Min-Max scaling or z-scorenormalization.

- **External Data Integration:**

  - Incorporate external data sources that might impact product demand. For example, integrating weather data can be important forpredicting demand for seasonal products.

- **Text Data Processing:**

- If you have text data (e.g., customer reviews, product descriptions), you can use natural language processing techniques toextract relevant information. This might include sentiment analysis orkeyword extraction.

- **Feature Interactions:**

  - Create new features that represent interactions between existing features. For example, you can multiply sales with marketing budgetto capture the interaction effect.

- **Time-Related Features:**

  - Introduce time-related features such as day of the week, month,or holiday indicators. These can help capture day-of-week or seasonality effects.

- **Dimensionality Reduction:**

  - If your dataset has a large number of features, consider dimensionality reduction techniques like Principal Component

Analysis (PCA) to reduce the number of features while preservingimportant information.

- **Regularization Features:**

  - In some cases, you may create regularization features to penalize extreme values or trends that are not typical.

- **Feature Importance Analysis:**

  - Use feature importance techniques (e.g., feature importance scores from tree-based models) to identify which features have themost influence on the model's predictions. This can help refine feature selection.

- **Cross-Validation:**

  - When engineering features, ensure you use cross-validation toassess their impact on model performance and prevent overfitting.

- **Iterate:**

  - Feature engineering is often an iterative process. Keep refiningyour feature set based on the model's performance and domain knowledge.

Regularly reevaluate the feature engineering process as new databecomes available or business conditions change.

EXAMPLE PROGRAM CODE:

In [43]:
```python
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
featureScores
```

In [47]:
```python
featureScores
```

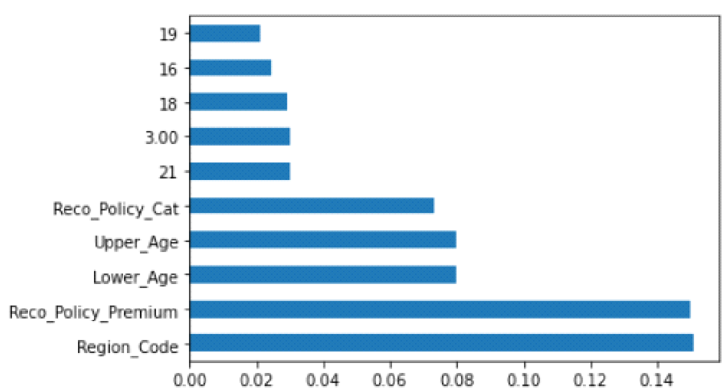
Out[47]:

	Specs	Score
0	City_Code	0.122643
1	Region_Code	74.805013
2	Accomodation_Type	0.756115
3	Reco_Insurance_Type	3.965972
4	Upper_Age	2.612166
5	Lower_Age	1.572930
6	Is_Spouse	0.632296
7	Health Indicator	0.453390
8	Holding_Policy_Duration	7.662646
9	Holding_Policy_Type	0.836189
10	Reco_Policy_Cat	1894.032997
11	Reco_Policy_Premium	9575.065324
12	diff_age	0.039347

In [189]:
```python
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(x,y)
print(model.feature_importances_)
```

Out[189]: ExtraTreesClassifier()

In [192]:
```python
feat_importances = pd.Series(model.feature_importances_, index=x.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



```
In [5]: from sklearn import preprocessing
        scalar=preprocessing.StandardScaler()
        mba1=scalar.fit_transform(mba)
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Tier 2	Tier 3	Supermarket Type1	Supermar Typ
count	6.818000e+03	6818.000000	6.818000e+03	6.818000e+03	6.818000e+03	6818.000000	6818.000000	6818.000000	6818.000000	6818.000
mean	1.704754e-16	0.355676	2.342737e-16	1.233002e-16	2.051747e-17	0.830302	0.326049	0.395571	0.651071	0.111
std	1.000073e+00	0.478753	1.000073e+00	1.000073e+00	1.000073e+00	0.598352	0.468800	0.489009	0.476667	0.314
min	-1.956094e+00	0.000000	-3.402972e+00	-1.759459e+00	-1.329746e+00	0.000000	0.000000	0.000000	0.000000	0.000
25%	-8.351767e-01	0.000000	-6.664603e-01	-7.589239e-01	-7.337084e-01	0.000000	0.000000	0.000000	0.000000	0.000
50%	5.250371e-03	0.000000	9.843446e-02	2.728679e-02	-1.376709e-01	1.000000	0.000000	0.000000	1.000000	0.000
75%	7.475728e-01	1.000000	7.696596e-01	7.091011e-01	1.292819e+00	1.000000	1.000000	1.000000	1.000000	0.000
max	2.011410e+00	1.000000	2.308161e+00	2.036689e+00	1.531234e+00	2.000000	1.000000	1.000000	1.000000	1.000

```
In [ ]: from sklearn.preprocessing import MinMaxScaler
        scale=MinMaxScaler()
        mba3=scale.fit(mba)
0    0.038462
1    0.038462
2    0.000000
3    0.000000
4    0.038462
Name: Car_MinMaxScale, dtype: float64
```

```
In [ ]: from sklearn.preprocessing import Normalizer
        scale=Normalizer()
        dataset=scale.fit_transform(dataset)
0    -1.106025
1    -0.086316
2    -1.106025
3    -1.106025
4    -0.086316
```

MODEL TRAINING:

The model training process for building a product demand predictionmodel involves preparing the data, selecting an appropriate modelingtechnique, training the model, and evaluating its performance. Here is a step-by-step guide for the model training process:

- **Data Preprocessing:**

- Before training your model, preprocess the data to ensure it's in a suitable format for modeling. Common preprocessing steps include

handling missing data, scaling or normalizing features, encoding categorical variables, and splitting the data into training and validation sets.

- **Select an Appropriate Model:**

•Choose a modeling technique that is suitable for your specificdemand prediction task. Common models used for demand prediction include:

•Time Series Models: such as ARIMA, Exponential Smoothing, orProphet for capturing time-dependent patterns.

•Regression Models: like linear regression, decision trees, randomforests, or gradient boosting for capturing linear and nonlinear relationships between features and demand.

•Machine Learning Models: such as neural networks (e.g., deep learning), support vector machines, or k-nearest neighbors, which can capture complex patterns and relationships in the data.

- **Train the Model:**

•Train the selected model on your training data. The steps involvedin training depend on the type of model:

•Time Series Models: You would typically estimate modelparameters using historical demand data.

- Regression Models: Use an optimization algorithm to find thebest coefficients that minimize the prediction error (e.g., mean squared error).
- Machine Learning Models: The training process involves adjusting the model's internal parameters to minimize a loss function, usually involving gradient descent or variations thereof.

- **Hyperparameter Tuning:**

  - Fine-tune the hyperparameters of your model to optimize its performance. You can use techniques like grid search, random search, or Bayesian optimization to find the best hyperparameters. This step is especially important for machine learning models.

- **Cross-Validation:**

  - Use cross-validation, such as k-fold cross-validation, to assess howwell your model generalizes to new data and to estimate its performance more accurately. This helps prevent overfitting.

- **Model Evaluation:**

  - Assess the model's performance using appropriate evaluation metrics. Common metrics for demand prediction include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Evaluate the model on both the training and validation datasets.

- **Feature Importance:**

- For machine learning models, determine the importance of individual features in making predictions. This information can help infeature selection and understanding the drivers of demand.

- **Model Interpretability:**

  - For complex models like neural networks, consider techniques formaking the model more interpretable, such as feature importance plots or SHAP (SHapley Additive exPlanations) values.

- **Model Selection:**

  - Compare the performance of different models and choose the onethat performs best on the validation data. Consider factors like interpretability, computational resources, and ease of implementation.

- **Final Model Training:**

  - Train the selected model on the entire training dataset, using the optimal hyperparameters, to create the final model that will be usedfor making predictions.

- **Save the Model:**

  - Save the trained model to a file or database so that it can be easily loaded and used for future predictions without having toretrain it.

- **Documentation:**

  - Maintain documentation that includes details of the selected model, its hyperparameters, and its performance on the training andvalidation datasets. This documentation is essential for model maintenance and future improvements.

It's important to continually monitor and update the model to ensure that it remains accurate and relevant for demand prediction.

EXAMPLE PROGRAM CODE:

```
import pandas as pd

import numpy as npimport matplotlib.pyplot as plt

%matplotlib inline

from matplotlib.pylab import rcParams
rcParams['figure.figsize']=20,10from keras.models import Sequential
from keras.layers import LSTM,Dropout,Densefrom
sklearn.preprocessing import MinMaxScaler

import pandas as pddf = pd.read_csv('aapl_stock_1yr.csv')


df.head()
```

OUTPUT:

	Date	Close/Last	Volume	Open	High	Low
0	09/15/2020	$115.54	184642000	$118.33	$118.829	$113.61
1	09/14/2020	$115.355	140150100	$114.72	$115.93	$112.8
2	09/11/2020	$112	180860300	$114.57	$115.23	$110
3	09/10/2020	$113.49	182274400	$120.36	$120.5	$112.5
4	09/09/2020	$117.32	176940500	$117.26	$119.14	$115.26

```
df.tail()
```

OUTPUT:

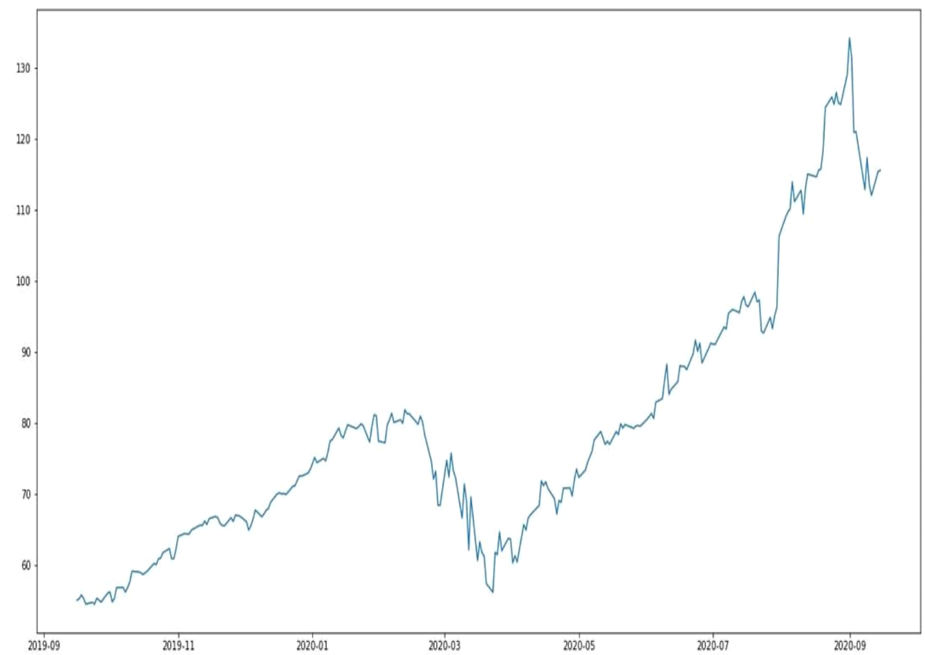
	Date	Close/Last	Volume	Open	High	Low
246	09/24/2019	$54.42	125737480	$55.2575	$55.6225	$54.2975
247	09/23/2019	$54.68	77678600	$54.7375	$54.96	$54.4125
248	09/20/2019	$54.4325	231908360	$55.345	$55.64	$54.3683
249	09/19/2019	$55.24	88751520	$55.5025	$55.94	$55.0925
250	09/18/2019	$55.6925	102572360	$55.265	$55.7125	$54.86
251	09/17/2019	$55.175	73545880	$54.99	$55.205	$54.78
252	09/16/2019	$54.975	84632560	$54.4325	$55.0325	$54.39

```
df = df[['Date', 'Close']]df.head()
```

OUTPUT:

```
plt.plot(df["Close"],label='Close Price history')
```

```
In [19]: plt.plot(df["Close"],label='AAPL Close Price history')
Out[19]: [<matplotlib.lines.Line2D at 0x14b306c50>]
```



MODEL EVALUATION:

The model evaluation process is a critical step in building a productdemand prediction model. It involves assessing the model's performance to determine how well it can accurately predict futureproduct demand. Here's a step-by-step guide for the model evaluation process:

- **Data Splitting:**

- Start by splitting your dataset into distinct subsets: a training set, avalidation set, and a testing set. A common split might be 70% for training, 15% for validation, and 15% for testing. The training set is used to train the model, the validation set helps fine-tune hyperparameters, and the testing set is reserved for final evaluation.

- **Choose Evaluation Metrics:**

•Select appropriate evaluation metrics that are relevant to yourdemand prediction task. Common metrics include:

•**Mean Absolute Error (MAE)**: Measures the average absolutedifference between predicted and actual demand.

•**Mean Squared Error (MSE)**: Measures the average squareddifference between predicted and actual demand, giving more weight to large errors.

•**Root Mean Squared Error (RMSE)**: The square root of MSE,providing a measure in the same units as the target variable.

•**R-squared (R^2)**: Indicates the proportion of variance in thetarget variable explained by the model. A higher R-squared value is generally better.

- **Model Evaluation on Validation Set:**

- Assess your model's performance on the validation set using the chosen evaluation metrics. This is an essential step for fine-tuning hyperparameters and making adjustments to the model if needed.

- **Hyperparameter Tuning:**

- If your model's performance on the validation set is not satisfactory, perform hyperparameter tuning. Adjust the model's hyperparameters and repeat the training and evaluation steps untilyou achieve the desired performance.

- **Cross-Validation:**

  - To obtain a more robust estimate of your model's performance and to prevent overfitting, you can use cross-validation techniques such as k-fold cross-validation. This involves splitting the data into multiple folds and training/evaluating the model multiple times.

- **Final Model Selection:**

  - After fine-tuning and optimizing your model on the validation set, select the best-performing model to move forward. You may choose the model with the lowest error or the highest R-squared, dependingon your specific goals.

- **Model Evaluation on the Testing Set:**

  - Once you have chosen your final model, evaluate its performance on the testing set. This provides an unbiased assessment of how wellthe model will perform on unseen data.

- **Visualizations:**

- Create visualizations such as time series plots, prediction vs. actual demand charts, and residual plots to gain insights into your model's behavior and errors.

- **Interpretability:**

  - If applicable, assess the model's interpretability. Depending on the model type, consider methods such as feature importance analysis or SHAP (Shapley Additive Explanations) values to understand which features drive predictions.

- **Benchmarking:**

  - Compare your model's performance to a simple baseline model(e.g., using historical average demand) to determine how much improvement your model provides.

- **Documentation and Reporting:**

  - Document the results of your model evaluation, including key metrics, findings, and any insights gained. This documentation is important for knowledge sharing and future reference.

- **Regular Monitoring and Reevaluation:**

  - After deploying the model in a production environment, continually monitor its performance. Reevaluate and update the model as needed with new data to ensure it remains accurate overtime.

It's important to maintain a robust evaluation framework to ensurethe model remains effective in practice.

EXAMPLE PROGRAM CODE:

```
df = df.sort_index(ascending=True,axis=0)data =
pd.DataFrame(index=range(0,len(df)),columns=['Date','Close'])for i in
range(0,len(data)):
    data["Date"][i]=df['Date'][i]

    data["Close"][i]=df["Close"][i]data.head()
```

OUTPUT:

Date	Date	Close
2019-09-16	2019-09-16	54.9750
2019-09-17	2019-09-17	55.1750
2019-09-18	2019-09-18	55.6925
2019-09-19	2019-09-19	55.2400
2019-09-20	2019-09-20	54.4325

Min-Max Scaler

```
scaler=MinMaxScaler(feature_range=(0,1))data.index=data.Date
data.drop("Date",axis=1,inplace=True)final_data = data.values
train_data=final_data[0:200,:]
valid_data=final_data[200:,:]scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(final_data)
x_train_data,y_train_data=[],[]
```

```
for i in range(60,len(train_data)):
    x_train_data.append(scaled_data[i-60:i,0])
    y_train_data.append(scaled_data[i,0])
```

LSTM Model

```
lstm_model=Sequential()
lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(
np.shape(x_train_data)[1],1)))
lstm_model.add(LSTM(units=50))
lstm_model.add(Dense(1))model_data=data[len(data)-len(valid_data)-
60:].values
model_data=model_data.reshape(-1,1)
model_data=scaler.transform(model_data)
```

Train and Test Data

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbo
se=2)X_test=[]
for i in range(60,model_data.shape[0]):
    X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
```

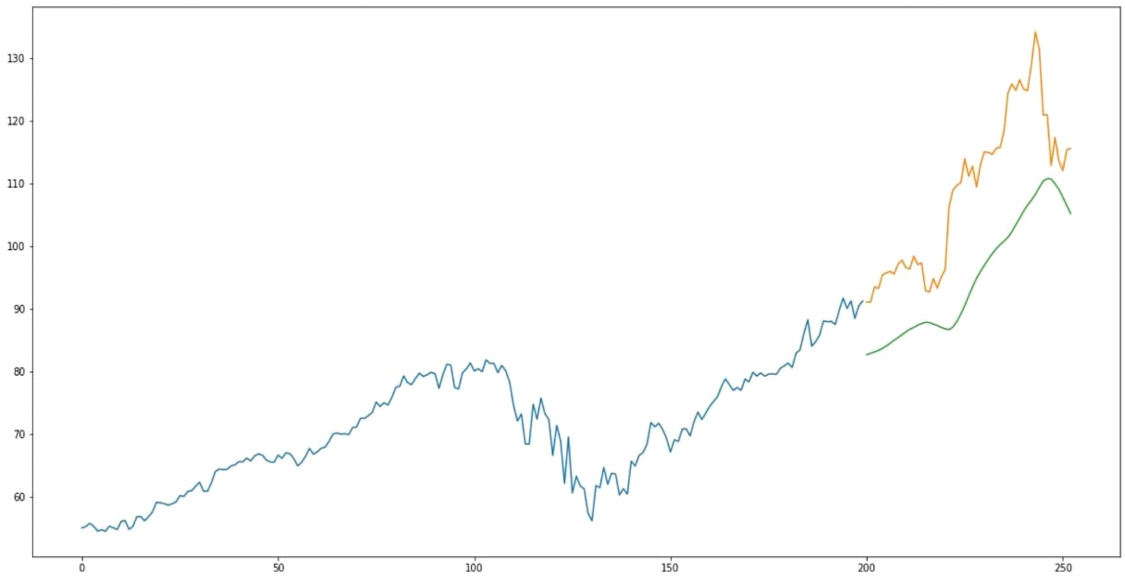
Prediction Function

```
predicted_stock_price=lstm_model.predict(X_test)
predicted_stock_price=scaler.inverse_transform(predicted_stock_price
)
```

Prediction Result

```
train_data=data[:200]
valid_data=data[200:]
valid_data['Predictions']=predicted_stock_price
plt.plot(train_data["Close"])
plt.plot(valid_data[['Close',"Predictions"]])
```

OUTPUT:



ADVANTAGES:

- **Accurate Forecasts:**

Machine learning models can analyze vast amounts of historical and real-time data to generate more accurate demand forecasts. This accuracy aids in better inventory management and reduces stockouts oroverstock situations.

- **Real-time Insights:**

With the ability to process and adapt to new data quickly, machine learning models provide real-time insights, enabling businesses to makerapid decisions based on the latest information.

- **Improved Inventory Management:**

Accurate demand prediction leads to optimized inventory levels. It minimizes holding costs by ensuring that products are available when needed, preventing overstock situations, and reducing excess inventory.

- **Customized Predictions:**

Machine learning models can be tailored to specific products, marketsegments, or geographic regions, allowing for more customized and granular demand predictions.

- **Enhanced Decision Making:**

Predictive models help in strategic decision-making by providing data-driven insights, allowing businesses to allocate resources efficiently andeffectively.

- **Adaptability to Various Factors:**

Machine learning models can consider multiple variables simultaneously, such as seasonal trends, market dynamics, promotionalactivities, and consumer behavior, resulting in more comprehensive andaccurate predictions.

- **Cost Savings:** By avoiding stockouts and overstock situations, businesses can save costs associated with excess inventory or lost salesdue to inadequate stock.

DISADVANTAGES:

- **Data Quality Dependency:**

Machine learning models heavily rely on the quality and relevance of the data used for training. Inaccurate, incomplete, or biased data can lead to flawed predictions, emphasizing the need for clean, representative, and high-quality datasets.

- **Complex Implementation:**

Developing, training, and maintaining machine learning models for demand prediction can be complex. It requires expertise in data scienceand machine learning, which might not be readily available within all organizations.

- **Interpretability:**

Some machine learning models, particularly complex ones like deep neural networks, lack interpretability. Understanding how the model arrives at specific predictions can be challenging, potentially leading to alack of transparency in decision-making processes.

- **Overfitting or Underfitting:**

ML models can suffer from overfitting (fitting too closely to historicaldata and performing poorly on new data) or underfitting (oversimplifying the model and missing important patterns), affectingthe accuracy and reliability of predictions.

- **External Factors and Unforeseen Events:**

Machine learning models might not account for unpredictable eventslike sudden market shifts, natural disasters, or changes in consumer behavior. They may struggle to accurately predict demand during unforeseen circumstances.

- **Continuous Maintenance and Updates:**

Models need continuous monitoring, retraining, and fine-tuning to remain relevant and effective. Without regular updates, their predictiveaccuracy may decline over time.

- **Resource Intensiveness:**

Implementing and maintaining machine learning systems can be resource-intensive, both in terms of computational power and thehuman expertise required to manage and update these models.

- **Ethical Considerations:**

There can be ethical implications regarding the use of data for predictions, especially in scenarios involving personal or sensitive information. Maintaining user privacy and ensuring ethical data use becomes crucial.

BENEFITS:

- **Improved Accuracy:**

Machine learning models can process vast amounts of data, identifying complex patterns and correlations that might be challenging for traditional statistical methods. This results in more accurate and precisedemand forecasts.

- **Enhanced Forecasting:**

By considering multiple variables such as seasonality, market trends, economic indicators, and consumer behavior, machine learning modelsimprove the accuracy of demand forecasts, aiding in better inventory management.

- **Real-time Insights:**

Machine learning models can be updated with new data in real-time, providing up-to-date insights for more responsive decision-making. Thisadaptability is particularly beneficial in rapidly changing markets.

- **Optimized Inventory Management:**

Accurate demand predictions lead to optimized inventory levels, reducing excess stock and minimizing the risk of stockouts. This resultsin cost savings by improving inventory turnover and reducing carrying costs.

- **Customized Solutions:**

Machine learning algorithms can be tailored to specific products, markets, or consumer segments, allowing for more personalized andadaptive demand forecasts.

- **Strategic Decision-making:**

Data-driven predictions enable businesses to make informed decisions.Predictive insights help in planning marketing strategies, pricing, and resource allocation effectively.

- **Cost Reduction:**

Accurate demand forecasts mitigate the need for excessive inventory, reducing costs associated with surplus goods and optimizing resources, ultimately improving the bottom line.

- **Automation and Efficiency:**

Machine learning models automate the demand prediction process, saving time and resources compared to traditional manual forecastingmethods.

- **Scalability:**

Once developed, machine learning models can be adapted and scaled tosuit various products or markets without significant additional costs, providing a scalable solution.

- **Competitive Edge:**

Companies leveraging machine learning for demand prediction gain a competitive advantage by better meeting customer needs, adapting to market changes swiftly, and optimizing their operations.

CONCLUSION:

- In conclusion, product demand prediction using machine learningoffers a promising approach for businesses seeking to optimize inventory management, enhance forecasting accuracy, and make

  data-driven decisions. The advantages of employing machine learning for demand prediction include improved accuracy, real-time insights, optimized inventory management, customization, and cost savings. These benefits empower companies to respond swiftly to market changes, allocate resources efficiently, and gain acompetitive edge.

- However, this approach comes with its own set of challenges.

Dependencies on data quality, the complexity of implementation,interpretability issues, and the potential for overfitting or underfitting are notable concerns. Unforeseen events, continuousmaintenance requirements, resource intensiveness, ethical considerations, model bias, and complexity for small businesses are additional factors that need to be addressed while utilizing machine learning for demand prediction.

- Notwithstanding these challenges, the benefits of machine learning in demand forecasting are substantial. The ability to provide more accurate predictions, real-time adaptability, and improved decision-making processes outweigh many of the limitations. As technology advances and methodologies improve,addressing these challenges becomes more achievable, especiallywith a focus on data quality, interpretability, and ethical considerations.

- Businesses willing to invest in the right infrastructure, data quality maintenance, and expertise can harness the power of machine learning for demand prediction. While it requires continuous monitoring, retraining, and fine-tuning, the potential for improvedinventory management, cost reduction, and a competitive advantage is substantial. Striking a balance between leveraging machine learning's strengths and addressing its limitations will be pivotal for successful adoption and implementation in the evolvinglandscape of demand forecasting and inventory management.