# **PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING**

Name:K.Tamiloli

Regno:912421104050

Phase 5 submission document

Project title: product demand prediction

Introduction:

A product company plans to offer discounts on its product during the upcoming holiday season. The company wants to find the price at which its product can be a better deal compared to its competitors. For this task, the company provided a dataset of past changes in sales based on price changes. You need to train a model that can predict the demand for the product in the market with different price segments.

Demand forecasting is used to predict what customer demand will be for a product or service, with varying levels of specificity. Accurate, timely forecasts are invaluable for both businesses and their customers. There are many different methods, both qualitative and quantitative, for creating and improving forecasts.

Demand forecasting is particularly important for growing businesses, especially small and midsize ones. Businesses of stable size and sales don’t face the same risks and variation in outcomes that a company trying to grow quickly must prepare for, and mistakes in forecasting are more easily absorbed by a larger enterprise than a small one. Improper scaling is a major cause of failure among startups, and flawed demand forecasting can lead to just that by not preparing the company to fill a big order or by causing it to scale too rapidly to meet demand that doesn’t materialize.

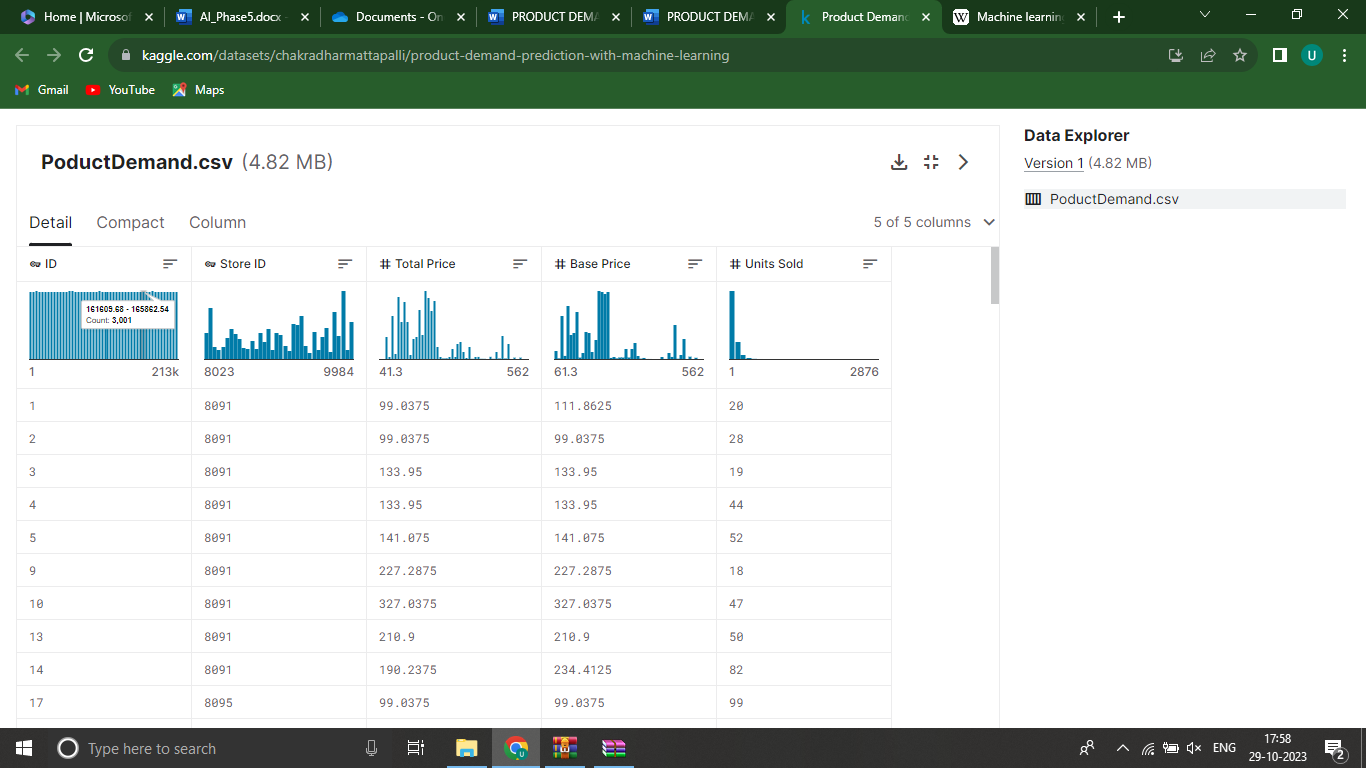
In demand forecasting, machine learning algorithms can analyze historical sales patterns and predict future trends. The first step is collecting data about past sales, such as product type, quantity sold, purchase frequency, seasonality, discounts, and more.

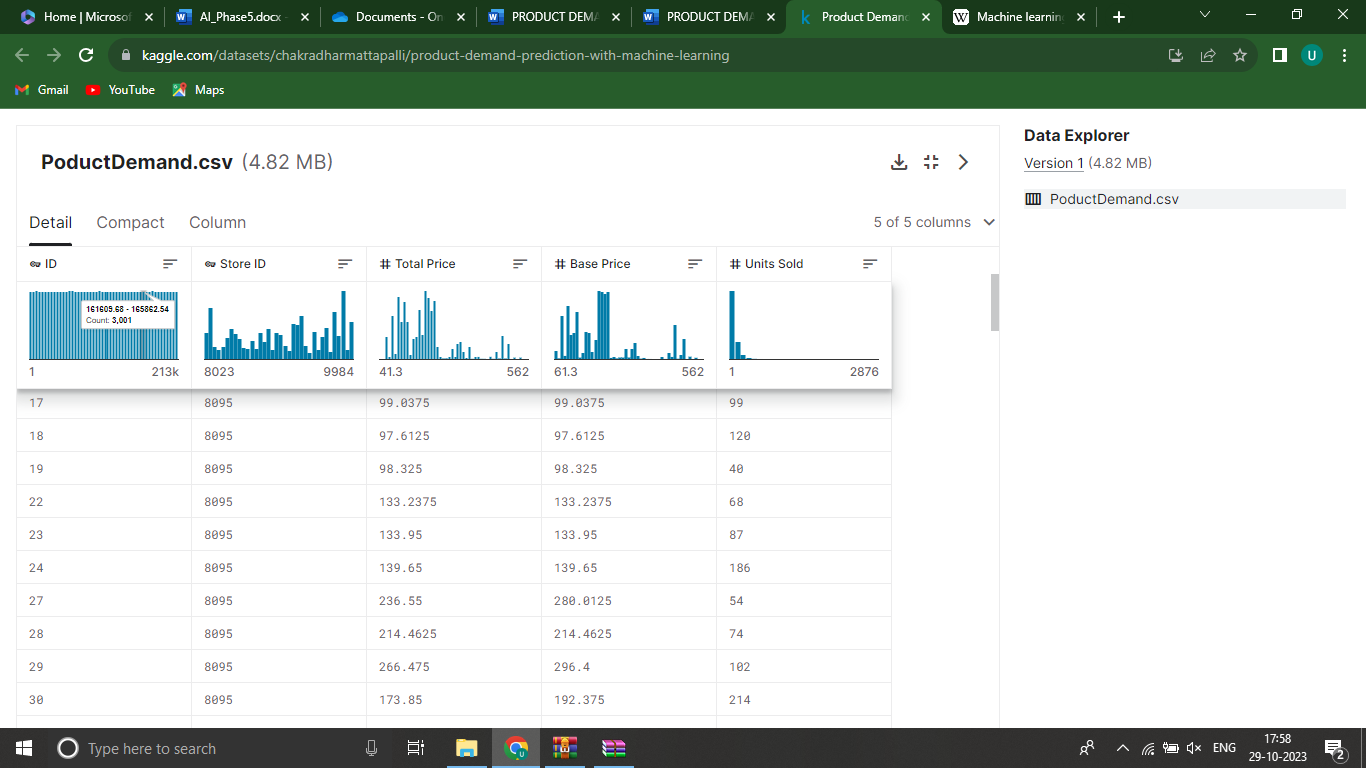
There are many different econometric models which differ depending on the analysis that managers wish to perform. The type of model that is chosen to forecast demand depends on many different aspects such as the type of data obtained or the number of observations. In this stage it is important to define the type of variables that will be used to forecast demand.

Datasource: To predict the future, statistics utilize data from the past. That’s why statistical forecasting is often called *historical.* The common recommendation is collecting data on sales for at least two years.

**Dataset Link:** [**https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning**](https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning)

Given dataset:





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 |

1. **Data Preprocessing:**

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

1. **Data Preprocessing:**

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

Process Overview:

The process involves several key stages:

1. **Data Collection:**

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

2.Data Preprocessing in Machine learning

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So, for this, we use data preprocessing tasks.

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

Steps involved :

* **Getting the dataset**
* **Importing libraries**
* **Importing datasets**
* **Finding Missing Data**
* **Encoding Categorical Data**
* **Splitting dataset into training and test set**
* **Feature scaling**

**3.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.

**4.Model Selection:**

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

**5.Model Training:**

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

**6.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 and effectiveness.

**7.Hyperparameter Tuning:**

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

**8.Forecasting and Prediction:**

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

**9.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.

**10.Decision Making:**

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

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

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

1. Jupyter Notebooks: Jupyter Notebooks are widely used for data exploration, analysis, and sharing of code and results. They support various programming languages, but Python is the most common choice.
2. Pandas: Pandas is a Python library for data manipulation and analysis. It is used for cleaning, transforming, and organizing data

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

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

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

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

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

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

10.Apache Spark: For handling large-scale data processing and distributed computing.

11.Tableau or Power BI: Data visualization tools to create interactive dashboards and reports for exploring and presenting predictions.

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

1. Version Control Systems: Tools like Git and GitHub are used to track changes in code and collaborate on projects.

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

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

1. Deployment Platforms: Tools like Flask, FastAPI, and cloud-based serverless platforms like AWS Lambda are used to deploy machine learning models into production.

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

1. Anomaly Detection Tools: For identifying unusual patterns in demand data, such as outlier detection algorithms.

1. Collaboration and Project Management Tools: Tools like Jira,

Trello, and Slack can be used to manage the project and collaborate with team members.

**1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:**

**steps:**

1. problem definition

1. 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 performing research for business, governmental or academic purposes, data collection allows you to gain first-hand knowledge and original insights into your [research problem.](https://www.scribbr.com/research-process/research-problem/)

**(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. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

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 from the 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 your demand forecasting task. Time series models like ARIMA or machine learning models like Random Forest, XGBoost, or LSTM (if you have a significant amount of data) are common choices.

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 the preprocessed training data.

Example:

model.fit(X\_train, y\_train)

**(6).Evaluation:**

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

2.DESIGN INTO INNOVATION

**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, and other recurring patterns. Visualization tools and statistical tests can be helpful in this phase.

**Incorporating time series forecasting techniques:**

* **ARIMA (Auto Regressive Integrated Moving Average):**

Suitable for stationary data with autoregressive and moving average components.

* **SARIMA (Seasonal ARIMA):**

Extends ARIMA to handle seasonal patterns in data.

#  **Exponential Smoothing Methods:**

These include Holt-Winters for capturing trends and seasonality.

#  **Prophet:**

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

#  **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 predictions for 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 metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared 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 the model 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 for factors that influence demand fluctuations.

**PROGRAM:**

import pandas as pd import numpy as np import plotly.express as px import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeRegressor data=pd.read\_csv("C:\Users\mabir\AppData\Local\Microsoft\Windo ws\INetCache\IE\AHLGJQP8\archive[1].zip ")

data.head()

Output:

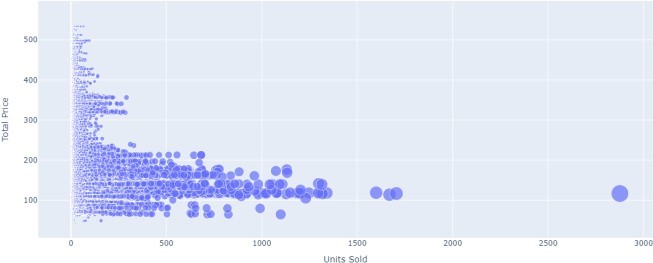
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID |  | Store ID | Total Price | Base Price | Units Sold |
|  | 1 | 8091 | 99.0375 | 111.8625 | 20 |
|  | 2 | 8091 | 99.0375 | 99.0375 | 28 |
|  | 3 | 8091 | 133.95 | 133.95 | 19 |
|  | 4 | 8091 | 133.95 | 133.95 | 44 |
|  | 5 | 8091 | 141.075 | 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() **output:**

  
**Correlation between the features of the dataset:**

print(data.corr()) **Output:**

**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**

1

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

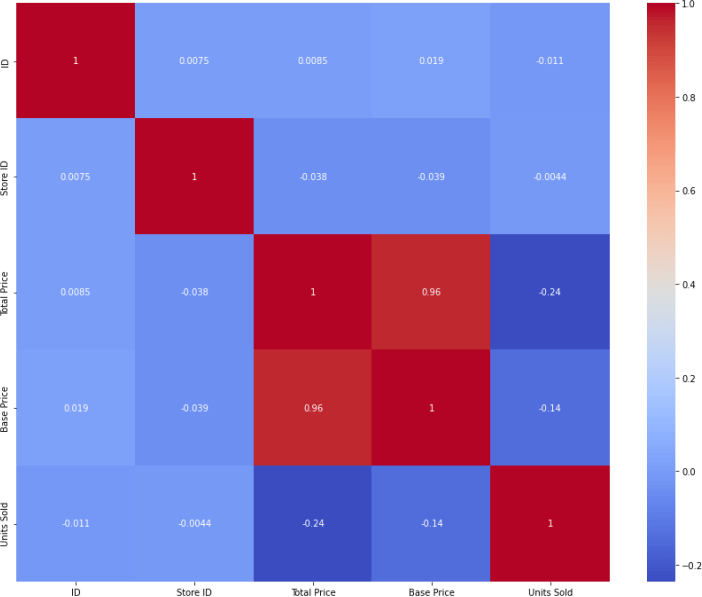
plt.figure(figsize=(15, 12))

3

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

4 plt.show()

**Output:**



# 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 ARIMA from 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

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

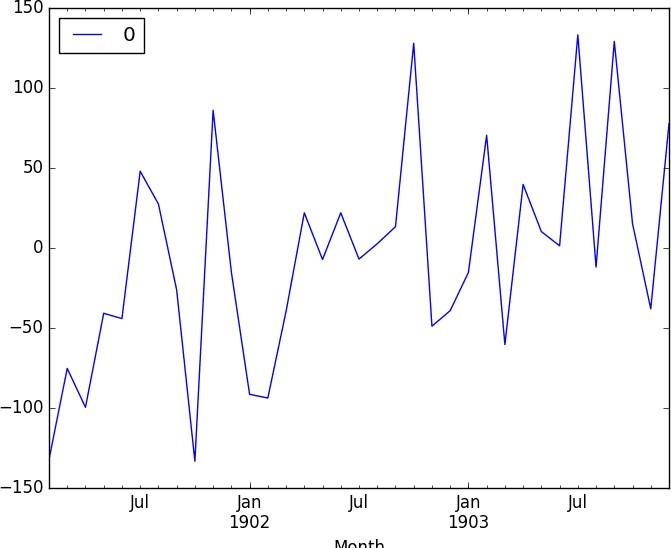
Prob(Q): 0.44 Prob(JB): 0.62

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.

  
**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\_error from math import sqrt

# load dataset

def parser(x):

return datetime.strptime('190'+x, '%Y-%m')

eries = 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(yhat) obs = test[t] history.append(obs) print('predicted=%f, expected=%f' % (yhat, obs))

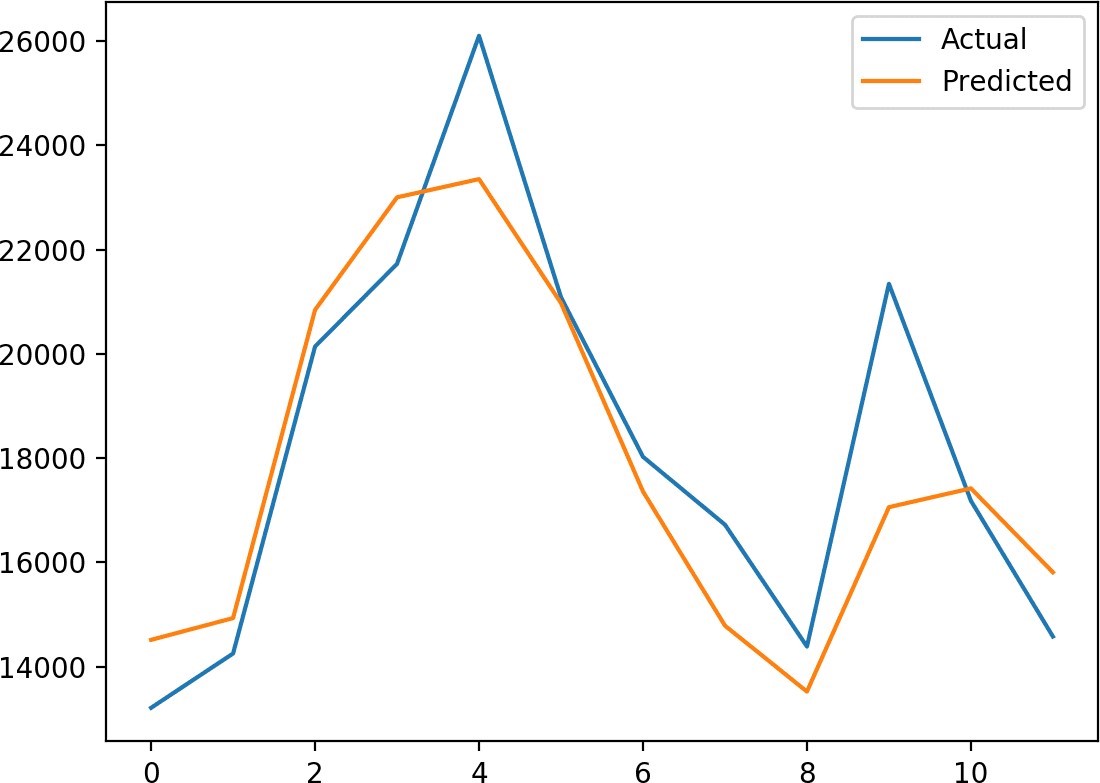
# evaluate forecasts

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()

Output:



**Prophet:**

# make an in-sample forecast from pandas import read\_csv from pandas import to\_datetime from pandas import DataFrame from fbprophet import Prophet from 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'])

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

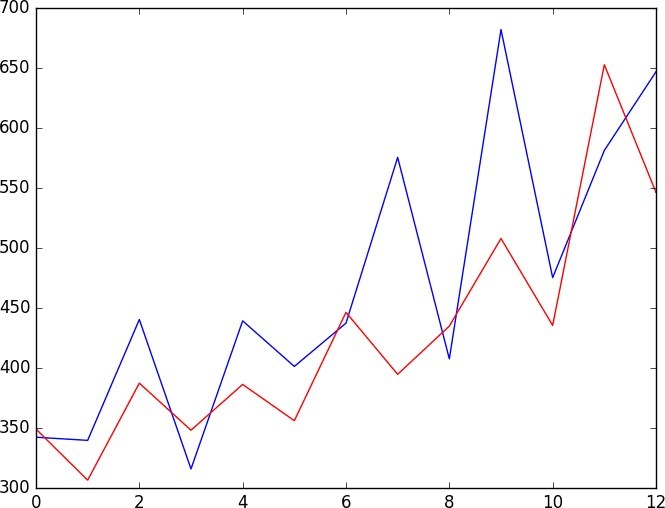
model.plot(forecast) pyplot.show()

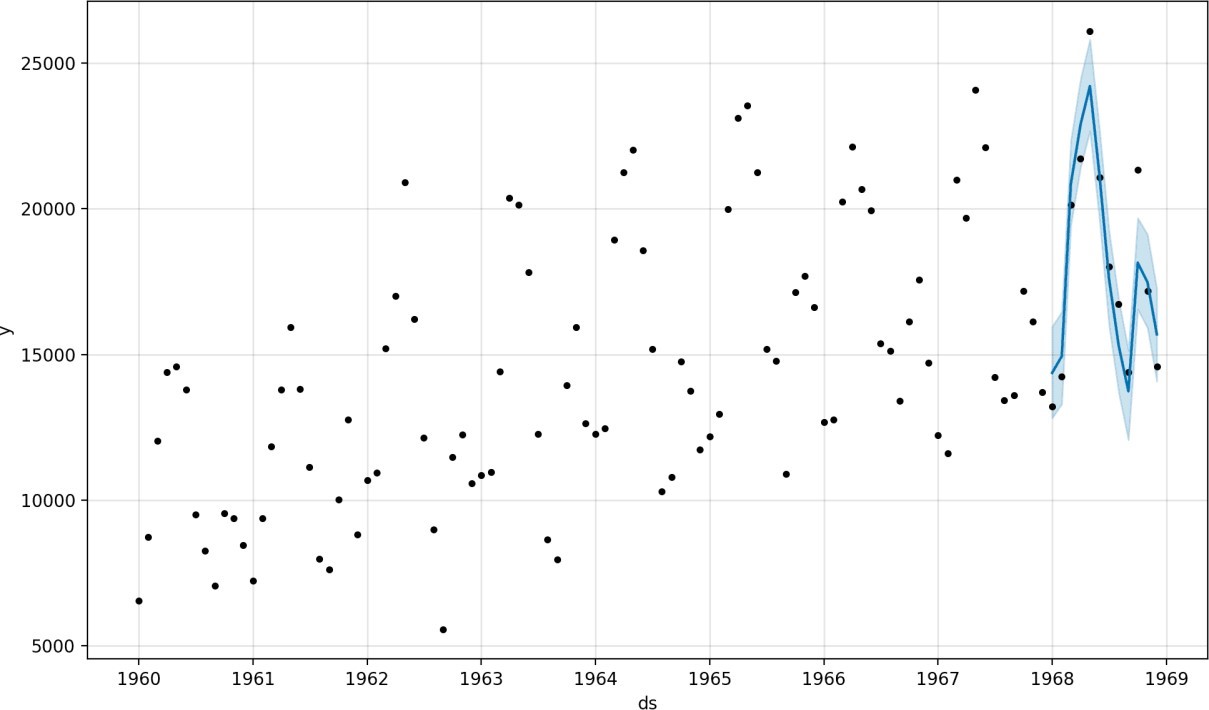
Output:

ds yhat yhat\_lower yhat\_upper

1. 1968-01-01 14364.866157 12816.266184 15956.555409
2. 1968-02-01 14940.687225 13299.473640 16463.811658
3. 1968-03-01 20858.282598 19439.403787 22345.747821
4. 1968-04-01 22893.610396 21417.399440 24454.642588
5. 1968-05-01 24212.079727 22667.146433 25816.191457

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:**

1. 



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

**STEPS:**

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

* **Data Collection:**

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

* **Import Libraries:**

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

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

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

**Data Cleaning:**

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

data['column\_name'].fillna(data['column\_name'].mean(), inplace=True)

**Feature Engineering:**

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

# Example: Create a 'month' feature from a date column data['month'] = pd.to\_datetime(data['date\_column']).dt.month

* + **Data Splitting:**

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

from sklearn.model\_selection import train\_test\_split

X = data.drop('target\_column', axis=1) y = data['target\_column']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

F,{e6a69e9e-34b2-4806-a164-424600ed6a15}{101},0.6666666666666666,0.6666666666666666**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.

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

import pandas as pd import numpy as np import plotly.express as pximport seaborn as sns import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeRegressor

data=pd.read\_csv("https://raw.githubusercontent.com/amankharwal/ Website-data/master/demand.csv") data.head()

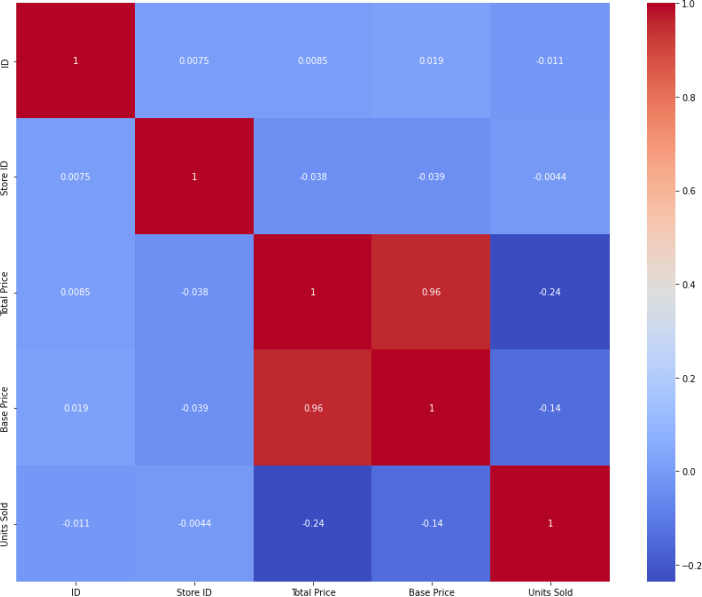
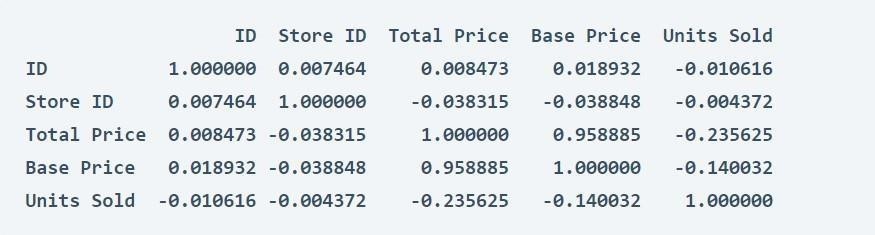
**Look at whether this dataset contains any null values or not:**

data.isnull().sum()

**We can see that most of the data points show the sales of theproduct is increasing as the price is decreasing with some**

**exceptions. Now let’s have a look at the correlation between the features of the dataset:**

print(data.corr())

correlations =

data.corr(method='pearson') plt.figure(figsize=(15, 12))

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

4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION,ETC.

**Overview of the process:**

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

1. **Define the Problem:** 
   * Clearly define the problem you want to solve. What product or products are you trying to predict demand for? What are your specific goals and objectives?

**2.Data Collection:**  Gather historical data related to the product's sales, including sales volume, price, and any other relevant variables. Additional data sources may include marketing activities, seasonality, economic indicators, and external factors.

1. **Data Preprocessing:** 
   * Clean and preprocess the collected data. This may involve handling missing data, outliers, and ensuring data consistency.

1. **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.
2. **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 the model's performance.

6.Model training:

Train your chosen model on the training dataset. This involves optimizing model parameters to minimize the prediction error.

1. **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).
2. **Model Evaluation:** 
   * Assess the model's performance using the validation dataset.

Common evaluation metrics for demand prediction include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

1. **Hyperparameter Tuning:** 
   * Fine-tune the model's hyperparameters to improve its performance on the validation set. Techniques like grid search or random search can be used for this purpose.
2. **Model Validation:** 
   * Once you're satisfied with the model's performance on the validation set, evaluate it on the testing set to assess its generalization to new, unseen data.

1. **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.

1. **Monitoring and Maintenance:**

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

**Feature engineering:**

**Domain Knowledge:**

* + Leverage domain expertise to identify potential features that could impact product demand. Speak to subject matter experts or conduct a literature review to gather insights.

**Feature Selection:**

* + Decide which features you will use in your model. Select those that 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 are historical 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.
  + **Model training:**
  + **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:**

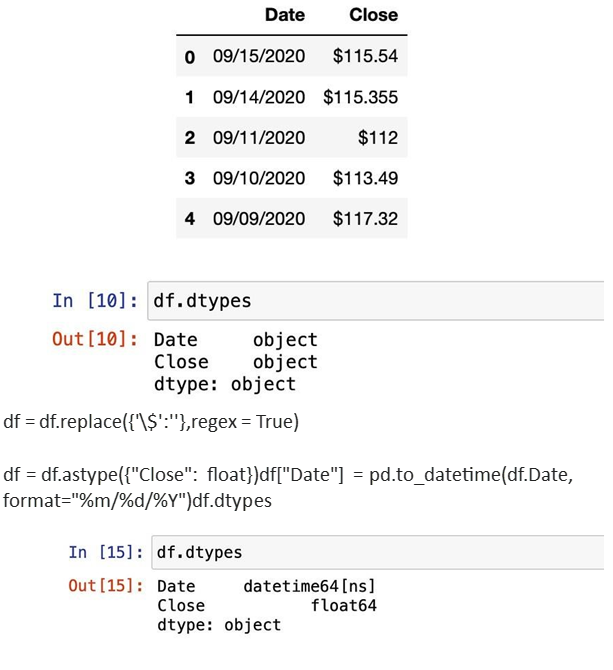
  
df.tail()

**OUTPUT:**



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

**OUTPUT:**



MODEL EVALUATION:

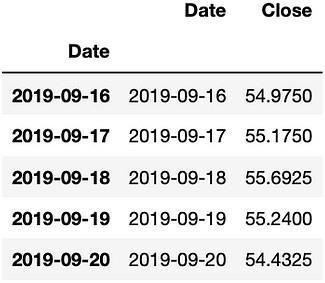
**EXAMPLE PROGRAM CODE:**

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:**



**ADVANTAGES:**

1. **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 or overstock situations.

1. **Real-time Insights:**

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

1. **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.

**DISADVANTAGES:**

**1.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.

Benefits:

**1.Complex Implementation:**

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

1. **Enhanced Forecasting:**

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

1. **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. This adaptability is particularly beneficial in rapidly changing markets.

1. **Optimized Inventory Management:**

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

1. **Customized Solutions:**

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

1. **Strategic Decision-making:**

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

1. **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.

**CONCLUSION:**

* + In conclusion, product demand prediction using machine learning offers 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, realtime insights, optimized inventory management, customization, and cost savings. These benefits empower companies to respond swiftly to market changes, allocate resources efficiently, and gain a competitive edge.