**Product demand prediction using machine learning**

**Definition:**

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.

The dataset that we have for this task contains data about:

the product id;

store id;

total price at which product was sold;

base price at which product was sold;

Units sold (quantity demanded);

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 px

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.tree import DecisionTreeRegressor

DecisionTreeRegresso

data = pd.read\_csv("https://raw.githubusercontent.com/amankharwal/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

**Steps:**

Predicting product demand using machine learning involves analyzing historical data to forecast future demand. Here's a simplified overview of the steps involved:

1. \*\*Data Collection:\*\* Gather historical data on product sales, including factors like time of sale, product attributes, pricing, promotions, and external variables (e.g., economic indicators, weather).

2. \*\*Data Preprocessing:\*\* Clean the data, handle missing values, and convert categorical variables into numerical format.

3. \*\*Feature Engineering:\*\* Create relevant features or attributes that can help the model better understand the data, e.g., seasonality, trends, and lag features.

4. \*\*Data Split:\*\* Divide the data into training and testing datasets. Cross-validation can also be used for model evaluation.

5. \*\*Model Selection:\*\* Choose an appropriate machine learning algorithm for demand prediction. Common choices include linear regression, decision trees, random forests, or more advanced models like neural networks.

6. \*\*Model Training:\*\* Train the selected model on the training data, where it learns patterns and relationships between features and product demand.

7. \*\*Model Evaluation:\*\* Evaluate the model's performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on the test dataset.

8. \*\*Hyperparameter Tuning:\*\* Fine-tune the model's hyperparameters to optimize its performance.

9. \*\*Deployment:\*\* Once a satisfactory model is developed, deploy it in a production environment where it can make real-time predictions or periodic forecasts.

10. \*\*Monitoring and Updating:\*\* Continuously monitor the model's performance and update it as new data becomes available.

Now analyze the relationship between the price and the demand for the product. Here I will use a scatter plot to see how the demand for the product varies with the price change:

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

size='Units Sold')

fig.show()

Product Demand Prediction Model

Now let’s move to the task of training a machine learning model to predict the demand for the product at different prices. I will choose the Total Price and the Base Price column as the features to train the model, and the Units Sold column as labels for the model:

x = data[["Total Price", "Base Price"]]

y = data["Units Sold"]

Now let’s split the data into training and test sets and use the decision tree regression algorithm to train our model:

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y,

test\_size=0.2,

random\_state=42)

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()

model.fit(xtrain, ytrain)

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

#features = [["Total Price", "Base Price"]]

features = np.array([[133.00, 140.00]])

model.predict(features)

**Output**:

array([27.])