**PRODUCT DEMAND PREDICTION USING MACHINE LEARNING**

In product demand prediction, 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. This data is fed into the algorithm to create models that identify sales patterns over time. Once these patterns are identified, they can be used to create an accurate forecast for future trends. Machine learning algorithms use data to learn how patterns and trends work for the datasets. The algorithms will generate an appropriate model based on this data. The more data added to the system over time, the more accurate the forecasting model becomes.

**Details About Columns :**

* **ID** : A dataset ID is a dataset's unique identifier. Any time you create a dataset with the Mapbox Studio dataset editor, it generates a dataset ID. You can use a dataset's ID to make requests related to the dataset using the Mapbox Datasets API.

* **STORE ID** : Store IDs can be important in demand forecasting and inventory management because they allow businesses to understand which stores are experiencing higher or lower demand for specific products. This information can be used to optimize inventory levels, restocking strategies, and product allocations to ensure that each store has the right products available to meet customer demand effectively.
* **TOTAL PRICE** : need a dataset that includes the prices and demand quantities for each product. Once you have that data, you can calculate the total price by multiplying the price of each product by its corresponding demand quantity and then summing up these values.

Here's a general formula:

**Total Price = (Price of Product 1 \* Demand of Product 1) + (Price of Product 2 \* Demand of Product 2) + ... + (Price of Product n \* Demand of Product n)**

* **BASE PRICE** : The term "base price" in the context of a dataset for product demand typically refers to the initial or starting price of a product before any discounts, promotions, or adjustments are applied. It is the price at which a product is initially offered to customers without any additional factors influencing its cost. In analyzing product demand data, the base price is a crucial factor to consider because changes in the base price can significantly impact consumer demand and sales.

* **UNITS SOLD** : "Units sold" in a dataset for product demand refers to the quantity of a particular product that has been purchased or sold during a specific time period. It represents the number of individual items or units of the product that have been bought by customers or moved through the supply chain. Analyzing units sold over time can help businesses understand trends, forecast future demand, and make informed decisions about inventory management, pricing, and production.

**LIBRARIES :**

* **NUMPY :** A popular Python library for numerical computations, using the Python package manager pip. Open your command prompt or terminal and use the following command:
* **pip install numpy**
* **PANDAS** : Use the following command to install Pandas using pip:
* **pip install pandas**

you can download it from the official Python website **(**[**https://www.python.org/downloads/**](https://www.python.org/downloads/)**).**

* **import matplotlib.pyplot as plt**

**from sklearn.model\_selection** :

1. \*Installing `matplotlib`\*:

To install `matplotlib`,

* **pip install matplotlib**

2. \*Installing `scikit-learn` (which includes `sklearn.model\_selection`)\*:

* **pip install scikit-learn**
* **import train\_test\_split**

**from sklearn.metrics :**

* \*Install scikit-learn\* if you haven't already. You can install it using pip
* **pip install scikit-learn**
* \*Import `train\_test\_split`\* from the

appropriate module:

* **python**

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

* **import f1\_score, accuracy\_score, confusion\_matrix ,classification\_report**

**from sklearn.ensemble :**

* Install scikit-learn (if not already installed)

using pip**:**

* **pip install scikit-learn**
* Import the necessary functions and classes

in your Python script or Jupyter Notebook using the following statements:

* **Python**

**from sklearn.metrics import f1\_score, accuracy\_score, confusion\_matrix, classification\_report**

* **Import RandomForestClassifier :**
* \*Install scikit-lear

**pip install scikit-learn**

2. \*Import `RandomForestClassifier`\* into your Python script or Jupyter Notebook:

**python**

**from sklearn.ensemble import RandomForestClassifier**

**TRAIN AND TEST DATASET :**

Creating train and test datasets is a fundamental step in building and evaluating machine learning models. Here's how you can create these datasets:

1. **Data Splitting** : Start with your complete dataset, which contains both the features (input variables) and the target variable (the variable you want to predict, like product demand).

2. **Randomization** : Shuffle the dataset randomly. This is important to ensure that there's no inherent order or bias in the data.

3. **Split Ratio** : Decide on a split ratio. A common choice is an 80/20 or 70/30 split, meaning you allocate 80% (or 70%) of the data to the training set and 20% (or 30%) to the test set. The training set is used to train the model, and the test set is used to evaluate its performance.

4. **Splitting** : Divide the shuffled dataset into the training and test sets based on the chosen ratio. Make sure both sets are representative of the overall data.

Where, X represents your features, y represents your target variable

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

In this code:

`X` is your feature data.

- `y` is your target variable.

- `test\_size` is set to 0.2, which means an 80/20 split.

- `random\_state` is used to ensure reproducibility. You can set it to any integer.

After splitting your data, you can use `X\_train` and `y\_train` to train your machine learning model, and `X\_test` and `y\_test` to evaluate its performance. Remember to keep your test set separate and only use it for evaluation to assess how well your model will perform on unseen data.

**REST OF DETAILS :**

**Objective:**

The main goal of this project is to predict the demand for a specific product or products based on historical data.

**Data Collection:**

Gather historical data related to the product(s) you want to predict demand for. This data should include features like sales volume, pricing, promotional activities, seasonality, and any external factors that may affect demand (e.g., holidays, economic indicators).

**Data Preprocessing:**

* Clean and preprocess the data to handle missing values, outliers.
* Convert categorical data into numerical format using techniques.
* Normalize or scale numerical features if needed.

**Exploratory Data Analysis (EDA):**

* Conduct exploratory data analysis to understand the distribution of data, correlations between variables, and identify any patterns or trends.
* Visualize data using plots and charts to gain insights.

**Feature Engineering:**

Create relevant features that can help improve the prediction model. For example, you can calculate moving averages, lag features, or derive new features from existing ones. Consider domain-specific features that might impact demand (e.g., weather data for a retail business).  
**Data Splitting:**

Split the dataset into training and testing sets. Common splits are 70/30 or 80/20 for training and testing, respectively. Ensure the data split maintains the temporal order if your data is time-series data.

**Model Selection:**

Choose an appropriate machine learning algorithm. Common choices include:

- Decision Trees

- Random Forests

- Gradient Boosting

**Model Training:**

Train the selected model(s) using the training dataset.

**Monitoring and Maintenance:**

- Continuously monitor the model's performance in production.

- Re-train the model periodically using new data to adapt to changing demand patterns.

**Documentation and Reporting:**

Document all aspects of the project, including data sources, preprocessing steps, model architecture, and deployment procedures.

Create reports or dashboards to communicate predictions and insights to stakeholders.  
**Legal and Ethical Considerations:**

Ensure compliance with data privacy regulations, especially if using customer data.

Address ethical considerations regarding the impact of predictions on pricing, inventory, and customer behaviour.  
**Feedback Loop:**

Establish a feedback loop to collect user feedback and incorporate it into model improvements.  
**Scaling and Optimization:**

Consider scaling up the solution to predict demand for more products or in additional markets. Optimize for efficiency and scalability as the project grows.Remember that the success of the project depends on data quality, feature engineering, and the choice of the right machine learning model. Regularly update and re-evaluate the model to maintain accurate predictions over time.

**The metrics used in the algorithm:**

1. \*Accuracy\*:

- Formula: (TP + TN) / (TP + TN + FP + FN)

Explanation: Accuracy measures the overall correctness of a classification model. It calculates the ratio of correctly predicted instances (both true positives and true negatives) to the total instances in the dataset. High accuracy indicates a good overall model performance, but it can be misleading in imbalanced datasets.

2. \*Precision\* (also known as Positive Predictive Value):

-

Formula: TP / (TP + FP)

- Explanation: Precision quantifies the accuracy of positive predictions made by the model. It calculates the ratio of true positives to the total predicted positives (true positives + false positives). Precision is valuable when the cost of false positives is high. High precision means that when the model predicts the positive class, it is often correct.

3. \*Recall\* (also known as Sensitivity or True Positive Rate):

- Formula: TP / (TP + FN)

- Explanation: Recall measures the ability of the model to identify all relevant instances of the positive class. It calculates the ratio of true positives to the total actual positives (true positives + false negatives). Recall is crucial when the cost of false negatives is high. High recall means the model can successfully capture most positive instances.

4. \*F-measure\* (also known as F1-score):

Formula: 2 \* (Precision \* Recall) / (Precision + Recall)

- Explanation: The F-measure is a harmonic mean of precision and recall. It provides a balance between these two metrics. A higher F-measure indicates a model that maintains a good trade-off between precision and recall. It is particularly useful when you want to find a balance between minimizing false positives and false negatives.

.