GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, Tiruttani

1103-GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PHASE 3

PROJECT TITLE

Product Demand Prediction With Machine Learning

Geethanjali V
3rd yr, 5th sem
Reg no.:110321104015
geethanjalibe202@gmail.com

PHASE 3: PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING

Predicting product demand with machine learning involves using historical data and various algorithms to forecast future demand for a particular product. Here's a simplified process:

1. Data Collection: Gather historical data on the product's sales, including factors that could affect demand like pricing, promotions, seasonality, and external events.

2. Data Preprocessing: Clean and prepare the data by handling missing values, outliers, and encoding categorical variables.

3. Feature Engineering: Create relevant features from the data, such as lag variables (past sales), seasonality indicators, and any external data that could impact demand.

4. Split Data: Divide the dataset into training and testing sets for model evaluation.

5. Model Selection: Choose an appropriate machine learning model, such as linear regression, decision trees, random forests, or more advanced techniques like time series forecasting with ARIMA or deep learning with LSTM.

6. Model Training: Train the selected model on the training data.

7. Model Evaluation: Assess the model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on the testing data.

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

9. Deployment: Once you have a satisfactory model, deploy it to predict future demand.

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

Remember that the effectiveness of your demand prediction model depends on the quality of your data, the choice of the right algorithm, and the continuous refinement of the model over time.

DATASET AND ITS DETAIL EXPLANATION AND IMPLEMENTATION OF PRODUCT DEMAND PREDICTIONWITH MACHINE LEARNING

Certainly, I can provide a simplified example of product demand prediction with machine learning using a sample dataset and Python. For a real-world application, you would need a more comprehensive dataset, but this example will illustrate the process. I'll use a synthetic dataset for demonstration purposes.

Sample Dataset Explanation:

Let's assume we have a dataset with the following columns:

- Date: The date of each sales record.
- ProductID: Unique identifier for each product.
- Price: The price of the product.
- Promotion: Indicates whether a product was on promotion (1 for yes, 0 for no).
- Demand: The number of units sold on that date.

Here's how you can implement product demand prediction:

python

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("product_demand_data.csv")

# Data preprocessing and feature engineering
data['Date'] = pd.to_datetime(data['Date'])
data['Month'] =  data['Date'].dt.month
data['Day'] = data['Date'].dt.day
data['Day_of_week'] = data['Date'].dt.dayofweek
```

```
# Split the data into training and testing sets
X = data[['Price', 'Promotion', 'Month', 'Day', 'Day_of_week']]
y = data['Demand']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a machine learning model (Random Forest)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Visualization (optional)
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Demand")
plt.ylabel("Predicted Demand")
plt.title("Actual vs. Predicted Demand")
plt.show()
```

In this example, we:

1. Load and preprocess the dataset.
2. Split the data into training and testing sets.
3. Choose a machine learning model (Random Forest) and train it.
4. Make predictions on the test set and evaluate the model's performance.
5. Optionally, visualize the actual vs. predicted demand.

In a real-world scenario, you would work with more complex datasets, perform hyperparameter tuning, and consider time series analysis if your data involves temporal patterns. Additionally, you may deploy the model for ongoing demand prediction.

BEGIN BUILDING THE PROJECT BY LOAD THE DATASET

To begin building the project for product demand prediction with machine learning, you should start by loading the dataset. Here's a step-by-step guide using Python and the Pandas library:

1. Import Libraries:

Start by importing the necessary libraries for data manipulation and analysis.

python

```
import pandas as pd
```

2. Load the Dataset:

Assuming you have a CSV file named "product_demand_data.csv" with the dataset in the same directory as your Python script, you can load it like this:

```
python
# Load the dataset
data = pd.read_csv("product_demand_data.csv")
```

If your dataset is in a different format (e.g., Excel, SQL database), Pandas provides functions to read those as well (e.g., `pd.read_excel()`, `pd.read_sql()`).

3. Explore the Dataset:

It's a good practice to take a quick look at the dataset to understand its structure. You can do this by examining the first few rows and some basic statistics:

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

# Get summary statistics
print(data.describe())
```

This will give you an initial understanding of the data and help you identify any missing values or outliers.

4. Data Preprocessing:

Depending on the dataset, you might need to perform data preprocessing tasks such as handling missing values, encoding categorical variables, and creating new features. You can use Pandas for these tasks, along with libraries like NumPy and Scikit-learn.

For example, to handle missing values, you can use:

python

```
# Check for missing values
print(data.isnull().sum())

# Handle missing values (e.g., fill with the mean)
data['Price'].fillna(data['Price'].mean(), inplace=True)
```

These are the initial steps for loading and exploring your dataset.

PREPROCESS DATASET

Preprocessing the dataset is a crucial step in preparing the data for machine learning. It involves tasks like handling missing values, encoding categorical variables, and feature engineering. Here's a simplified example of how to preprocess the dataset for product demand prediction:

```python
python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset
data = pd.read_csv("product_demand_data.csv")

# Handle missing values
data['Price'].fillna(data['Price'].mean(), inplace=True)

# Encode categorical variables (if applicable)
# For example, if 'Promotion' is a categorical variable with values 'Yes' and 'No':
data = pd.get_dummies(data, columns=['Promotion'], drop_first=True)

# Split the data into features (X) and the target (y)
X = data.drop('Demand', axis=1) # Features (exclude the 'Demand' column)
y = data['Demand']  # Target variable

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

In this code:

1. We handle missing values in the 'Price' column by filling them with the mean value.

2. If you have categorical variables (e.g., 'Promotion'), we use one-hot encoding to convert them into numerical format.

3. We split the dataset into features (X) and the target variable (y).

4. Finally, we split the data into training and testing sets to be used for building and evaluating the machine learning model.

Depending on your dataset, you may need to perform additional preprocessing steps, such as scaling numerical features, dealing with outliers, and performing more advanced feature engineering. Additionally, for time series data, you might need to consider time-based features and special handling.

PERFORMING DIFFERENT ANALYSIS NEEDED

1. Exploratory Data Analysis (EDA): EDA involves visualizing and summarizing your data to understand its characteristics. You can create various plots and statistics to identify trends, patterns, and outliers in your dataset. Common EDA tools include histograms, scatter plots, and box plots.

2. Correlation Analysis: Calculate correlations between different features and the target variable ('Demand'). This can help you identify which features are most strongly associated with demand. You can use techniques like Pearson correlation or create correlation matrices.

3. Time Series Analysis (if applicable): If your data involves time-based information (e.g., sales over time), consider time series analysis to identify seasonality and trends. Methods like decomposition and autocorrelation can be useful.

4. Feature Importance Analysis: If you're using a machine learning model, you can analyze feature importance scores to understand which features have the most impact on your predictions. Techniques like feature importance plots for tree-based models can be used.

5. Residual Analysis: If you're using regression models, analyze the residuals (the difference between predicted and actual values) to check if there's any pattern in the errors. Ideally, residuals should be normally distributed and randomly scattered around zero.

6. Model Validation: This involves assessing the performance of your machine learning model. Use metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared to evaluate how well your model predicts demand.

7. Hyperparameter Tuning: Optimize the hyperparameters of your machine learning model to improve its predictive performance. Techniques like grid search or random search can be employed.

8. Cross-Validation: Perform cross-validation to assess how well your model generalizes to new data. This helps you understand whether your model might be overfitting the training data.

9. Comparison of Multiple Models: Try different machine learning algorithms and compare their performance to select the one that best fits your dataset.

10. Feature Engineering: If necessary, experiment with creating new features that might improve the predictive power of your model.