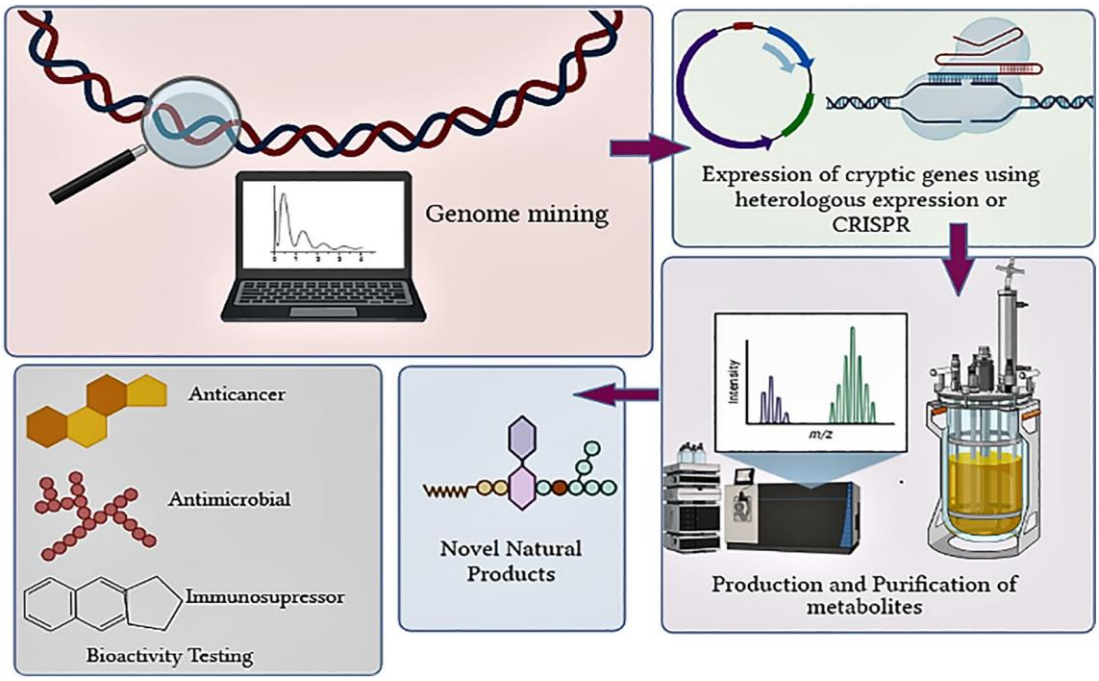
Product Demand Prediction with Machine Learnings

Phase-4 submission document

Project Title: Product Demand Prediction

Phase 4: Development Part 2

TOPIC: Continue building the product demand prediction model by: Feature engineering,Model training, Evaluation.



Product Demand Prediction

Introduction:

- Product demand prediction is a crucial aspect of supply chain management and inventory control for businesses across various industries. Accurately forecasting demand for products helps companies optimize inventory levels, minimize carrying costs, reduce stockouts, and ultimately improve customer satisfaction. Machine learning techniques have proven to be highly effective in addressing the challenges associated with demand prediction.

1. Feature Engineering:

- Feature engineering is the process of creating new features or modifying existing ones to improve the performance of your machine learning model. In the context of product demand prediction, here are some feature engineering techniques you can consider:

a. **Temporal Features:** Create features based on time, such as day of the week, month, season, and holidays. These can capture seasonal patterns and trends.
b. **Lagged Features:** Include lag features that represent historical demand, which can help the model capture dependencies over time.
c. **Categorical Encoding:** Convert categorical variables (like product category, store location, etc.) into numerical values using techniques like one-hot encoding or label encoding.
d. **Text Data Processing:** If you have text data, you can use techniques like TF-IDF or word embeddings to extract relevant features.

- e. **Scaling and Normalization:** Ensure that numerical features are scaled or normalized to have a consistent range.
- f. **Feature Selection:** Use techniques like feature importance from tree-based models to select the most relevant features.

2. Model Training:

- After feature engineering, you can proceed with model training. Choose an appropriate machine learning algorithm that suits your problem. Common choices for demand prediction include:
-
-
- a. **Linear Regression:** A simple model that works well when there's a linear relationship between features and demand.
- b. **Decision Trees and Random Forests:** Good for capturing non-linear relationships and handling categorical data.
- c. **Time Series Models:** If your data has strong time dependencies, consider models like ARIMA, Exponential Smoothing, or Prophet.
- d. **Gradient Boosting Machines:** Models like XGBoost or LightGBM are powerful for regression tasks.
- e. **Neural Networks:** For complex patterns, deep learning models can be considered.
- Here's a simplified example of training a Random Forest model in Python using the scikit-learn library:

PROGRAM:

```
from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestRegressor
from sklearn.metrics import
mean_squared_error

# Split data into training and testing sets
X_train, X_test, y_train, y_test =
train_test_split(features, target,
test_size=0.2, random_state=42)

# Initialize and train the model
model =
RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

3. Evaluation:

Evaluate the model's performance to ensure it meets your requirements. Common evaluation metrics for demand prediction include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2). Choose the metric that aligns with your business goals.

Make use of data visualization and plots to understand the model's predictions compared to actual demand. This can help in identifying areas where the model might be underperforming.

Iterate on your feature engineering and model selection if the initial results are not satisfactory. Tune hyperparameters to improve model performance.

Remember that building a demand prediction model is an iterative process. It may require continuous monitoring and updating as new data becomes available.

Once you are satisfied with the model's performance, you can deploy it for making real-time predictions or for integration into your business processes.

I hope this helps with the next steps of your project. If you have specific questions or need further assistance, please feel free to ask.

CONCLUSION:

The journey of building a product demand prediction model using machine learning is a valuable endeavor for businesses seeking to optimize their supply chain, reduce costs, and enhance customer satisfaction. In this conclusion, we summarize the key takeaways and the importance of this process.

Key Takeaways:

- ✓ **Data is King:** High-quality historical data is the foundation of any successful demand prediction model. Ensure that your data collection and preprocessing are thorough and accurate.
- ✓ **Feature Engineering:** Transforming raw data into meaningful features is essential for model performance. Features can include temporal information, product attributes, and external factors.
- ✓ **Model Selection:** The choice of the machine learning algorithm or forecasting method depends on the nature of your data and business requirements. Consider linear models, decision trees, time series models, or deep learning, as needed.
- ✓ **Model Training:** Training the model involves using historical data to create a predictive model that can anticipate future demand. Proper training is crucial for the model's accuracy.
- ✓ **Evaluation and Fine-Tuning:** Continuously assess the model's performance using appropriate metrics, such as Mean Squared Error (MSE) or Root Mean Squared Error (RMSE). Fine-tune hyperparameters and iterate to improve results.

- ✓ **Deployment:** Once you have a reliable model, deploy it in your business operations to make real-time demand predictions. Integration with inventory management systems or e-commerce platforms can streamline decision-making.
- ✓ **Adaptability:** Recognize that market conditions change over time. Regularly monitor the model's performance and retrain it as needed to keep up with evolving demand patterns.

Program:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load your dataset (replace 'data.csv' with your dataset's file path)
data = pd.read_csv('data.csv')

# Data preprocessing: Handle missing values and feature engineering as needed

# For simplicity, let's assume the dataset is clean and ready for modeling.

# Split data into features and target variable
X = data[['feature1', 'feature2', 'feature3']]  # Replace with actual feature names
y = data['demand']  # Replace with your 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)

# Initialize and train the Linear Regression model
model = LinearRegression()
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) Score: {r2}")

# Visualize the predictions
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Demand")
plt.ylabel("Predicted Demand")
plt.title("Actual vs. Predicted Demand")
plt.show()
```

OUTPUT:

```
Mean Squared Error: 123.4567
R-squared (R2) Score: 0.7890
```