IBM Applied Data Science Project

Phase 2: Innovation

Rohit Kumar Pandey

Problem Definition:

The problem at hand is to develop a machine learning model that can accurately forecast product demand based on historical sales data and external factors. This project aims to assist businesses in optimizing their inventory management and production planning processes to efficiently meet customer demands. To achieve this, we will undertake several key steps, including data collection, data preprocessing, feature engineering, model selection, model training, and evaluation.

Innovation Phase: Incorporating ARIMA Model for Demand Forecasting

Objective:
In this phase, we aim to enhance our demand forecasting capabilities by incorporating advanced time series forecasting techniques like ARIMA (AutoRegressive Integrated Moving Average). By leveraging ARIMA, we can capture temporal patterns and improve the accuracy of our demand predictions.

Implementation Steps:

1. Data Transformation:
- Convert the historical sales data into a time series format.
- Define the time intervals (e.g., monthly, weekly) that suit the demand data's temporal nature.

2. ARIMA Model Configuration:
- Determine the appropriate ARIMA order parameters (p, d, q).
- Select a suitable seasonal order if seasonal patterns are present (P, D, Q). - Conduct model diagnostics to ensure the stationarity of the time series.

3. **Model Training:**
- Split the time series data into training and validation sets.
- Train the ARIMA model on the training data.
- Validate the model's performance using out-of-sample forecasting.

4. **Model Evaluation:**
- Assess the ARIMA model's performance using time series-specific metrics such as: - AIC (Akaike Information Criterion)

- BIC (Bayesian Information Criterion)
- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

5. **Integration with Existing Models:**
- Combine the ARIMA forecasted values with the predictions from our existing demand

forecasting models.
- Explore ensemble techniques (e.g., weighted averaging) to aggregate forecasts for

improved accuracy

6. **Continuous Improvement**:
- Implement a robust monitoring and maintenance plan for the ARIMA model. - Regularly retrain the ARIMA model to adapt to changing demand patterns.

Expected Outcomes:
- Improved accuracy in demand forecasting due to the capture of temporal patterns. - Enhanced model reliability through the integration of ARIMA with existing models. - Better adaptability to changes in demand dynamics, resulting in more informed decision-making.

Conclusion:-

By incorporating ARIMA into our demand forecasting framework, we take a significant step towards achieving more precise and reliable predictions. This innovation aligns with our commitment to data-driven excellence and continuous improvement in addressing complex demand patterns.

Implementation of ARIMA Model :

#importing the libraries

```
import pandas as pd

import numpy as np

from statsmodels.tsa.arima_model import ARIMA

from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt
```

 # Load your training and testing data

```
train_data = pd.read_csv('train_data.csv')

test_data = pd.read_csv('test_data.csv')
```

 # Extract the demand column from the datasets

train_demand = train_data['demand']

test_demand = test_data['demand']

 # Define ARIMA parameters (p, d, q)

p, d, q = 1, 1, 1

Fit the ARIMA model to the training data

model = ARIMA(train_demand, order=(p, d, q))

model_fit = model.fit(disp=0)

 # Make predictions for the testing data

predictions, _, _ = model_fit.forecast(steps=len(test_demand))

 # Calculate the Mean Squared Error (MSE) to evaluate the model

mse = mean_squared_error(test_demand, predictions)

print('Mean Squared Error:', mse)

 # Visualize the actual vs. predicted demand

plt.plot(test_demand, label='Actual Demand')

plt.plot(predictions, color='red', label='ARIMA Forecast')

plt.legend()

plt.xlabel('Time')

plt.ylabel('Demand')

plt.title('ARIMA Model - Product Demand Prediction')

plt.show()