Data Science - Industrial Production Forecasting (Internship Project)

To develop a time-series forecasting model using ARIMA and SARIMA to predict industrial production for 2 years. This example assumes you have a time-series dataset of industrial production.

Step 1: Import Libraries

First, let's import the necessary libraries.

```python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.tsa.arima_model import ARIMA

from statsmodels.tsa.statespace.sarimax import SARIMAX

from sklearn.metrics import mean_squared_error

For reproducibility

np.random.seed(42)

```Step 2: Load the Data

Assuming your dataset is in a CSV file with a column for dates and a column for production values.

```python

Load the dataset

data = pd.read_csv('industrial_production.csv', parse_dates=['Date'], index_col='Date')

data.head()

```

Step 3: Data Preprocessing

Let's visualize the data and check for stationarity.

```python
Visualize the time series data
plt.figure(figsize=(10, 6))
plt.plot(data, label='Industrial Production')
plt.title('Industrial Production Over Time')
plt.xlabel('Date')
plt.ylabel('Production')
plt.legend()
plt.show()
Check for stationarity using the Augmented Dickey-Fuller test
from statsmodels.tsa.stattools import adfuller
result = adfuller(data['Production'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])
```

Step 4: Differencing (if necessary)

If the data is not stationary, perform differencing.

```python
Differencing to make the data stationary
data_diff = data.diff().dropna()
Check stationarity again
result = adfuller(data_diff['Production'])
```

```
print('ADF Statistic:', result[0])
print('p-value:', result[1])
```

Step 5: Split Data into Train and Test

Split the data into training and testing sets.

```python
Split the data into train and test sets
train_size = int(len(data) * 0.8)
train, test = data[0:train_size], data[train_size:]
plt.figure(figsize=(10, 6))
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.title('Train and Test Data')
plt.xlabel('Date')
plt.ylabel('Production')
plt.legend()
plt.show()
```

Step 6: ARIMA Model

Fit the ARIMA model on the training data.

```python
Fit ARIMA model
arima_model = ARIMA(train, order=(5,1,0))
```

arima_result = arima_model.fit(disp=False)

print(arima_result.summary())

Forecast

arima_forecast = arima_result.forecast(steps=len(test))[0]

Plot the results

plt.figure(figsize=(10, 6))

plt.plot(train, label='Train')

plt.plot(test, label='Test')

plt.plot(test.index, arima_forecast, label='ARIMA Forecast')

plt.title('ARIMA Forecast')

plt.xlabel('Date')

plt.ylabel('Production')

plt.legend()

plt.show()

Calculate the error

arima_error = mean_squared_error(test, arima_forecast)

print(f'ARIMA Model Mean Squared Error: {arima_error}')

```

Step 7: SARIMA Model

Fit the SARIMA model on the training data.

```python

Fit SARIMA model

sarima_model = SARIMAX(train, order=(1,1,1), seasonal_order=(1,1,1,12))

sarima_result = sarima_model.fit(disp=False)

print(sarima_result.summary())

Forecast

sarima_forecast = sarima_result.forecast(steps=len(test))

Plot the results

plt.figure(figsize=(10, 6))

plt.plot(train, label='Train')

plt.plot(test, label='Test')

plt.plot(test.index, sarima_forecast, label='SARIMA Forecast')

plt.title('SARIMA Forecast')

plt.xlabel('Date')

plt.ylabel('Production')

plt.legend()

plt.show()

Calculate the error

sarima_error = mean_squared_error(test, sarima_forecast)

print(f'SARIMA Model Mean Squared Error: {sarima_error}')

```

Step 8: Evaluate the Model

Compare the performance of the ARIMA and SARIMA models.

```python

print(f'ARIMA Model Mean Squared Error: {arima_error}')

print(f'SARIMA Model Mean Squared Error: {sarima_error}')

Choose the best model based on the error

best_model = 'SARIMA' if sarima_error < arima_error else 'ARIMA'

print(f'The best model is: {best_model}')

```

Step 9: Forecast for the Next 2 Years

Use the best model to forecast for the next 2 years.

```python

Forecast for the next 2 years (assuming monthly data, so 24 steps)

if best_model == 'SARIMA':

future_forecast = sarima_result.forecast(steps=24)

else:

future_forecast = arima_result.forecast(steps=24)[0]

Plot the forecast

plt.figure(figsize=(10, 6))

plt.plot(data, label='Historical Data')

plt.plot(pd.date_range(start=data.index[-1], periods=25, freq='M')[1:], future_forecast, label='Future Forecast')

plt.title('Future Forecast for Industrial Production')

plt.xlabel('Date')

plt.ylabel('Production')

plt.legend()

plt.show()

```
```

This code provides a complete workflow for developing a time-series forecasting model using ARIMA and SARIMA to predict industrial production. Adjust the `order` and `seasonal_order` parameters based on your specific dataset for better results.