```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/bax-5y/df_bax_cleaned_to_view_outliers_5y.csv
```

### Kaggle Run of SARIMA: Prathik Mohan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
import itertools
import warnings
warnings.filterwarnings("ignore")
```

```
# Picking 4 years of data for SARIMA - If 3 years of daily data = 726 rows, that’s only about 2.8 seasonal cycles (assuming ~252 tra
# For SARIMA to learn seasonal patterns well, you should ideally have at least 3 full cycles, i.e. ~750+ rows for yearly seasonality

# Load data (same as you did)
df = pd.read_csv('/kaggle/input/bax-5y/df_bax_cleaned_to_view_outliers_5y.csv', index_col=0, parse_dates=True)

# Make sure index is datetime
df.index = pd.to_datetime(df.index)

# Slice last 4 years of data
df_4y = df.last('4Y')

# Check result
print(f"Rows in 4 years of data: {len(df_4y)}")
df_4y
```

Rows in 4 years of data: 786

| | Price | Open | High | Low | Vol. | Change % |
|---|---|---|---|---|---|---|
| **Date** | | | | | | |
| **2022-01-03** | 1792.24 | 1797.10 | 1797.10 | 1792.24 | 635180.0 | -0.28 |
| **2022-01-04** | 1796.49 | 1790.33 | 1796.49 | 1789.90 | 2440000.0 | 0.24 |
| **2022-01-05** | 1796.08 | 1797.20 | 1799.26 | 1795.84 | 2130000.0 | -0.02 |
| **2022-01-06** | 1788.93 | 1796.08 | 1796.08 | 1788.93 | 974210.0 | -0.40 |
| **2022-01-09** | 1794.47 | 1788.38 | 1795.43 | 1788.14 | 1350000.0 | 0.31 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2025-03-05** | 1975.92 | 1980.24 | 1980.71 | 1974.02 | 818340.0 | -0.21 |
| **2025-03-06** | 1973.89 | 1975.92 | 1975.92 | 1973.89 | 294350.0 | -0.10 |
| **2025-03-23** | 1965.58 | 1962.09 | 1967.91 | 1955.49 | 611460.0 | -0.42 |
| **2025-03-25** | 1957.49 | 1951.62 | 1957.49 | 1942.88 | 1100000.0 | -0.41 |
| **2025-03-27** | 1951.36 | 1954.72 | 1954.72 | 1945.34 | 457910.0 | -0.31 |

786 rows × 6 columns

```
series = df_4y['Price']  # Assuming 'Price' column exists
```

```
# Train-test split (80% train, 20% test)
train_size = int(len(series) * 0.8)
train, test = series[:train_size], series[train_size:]
```

```
# ✅ Manually set SARIMA orders here:
p, d, q = 2, 0, 1          # ARIMA part
P, D, Q, s = 1, 1, 1, 252  # Seasonal part — try 252, 273, 365
```

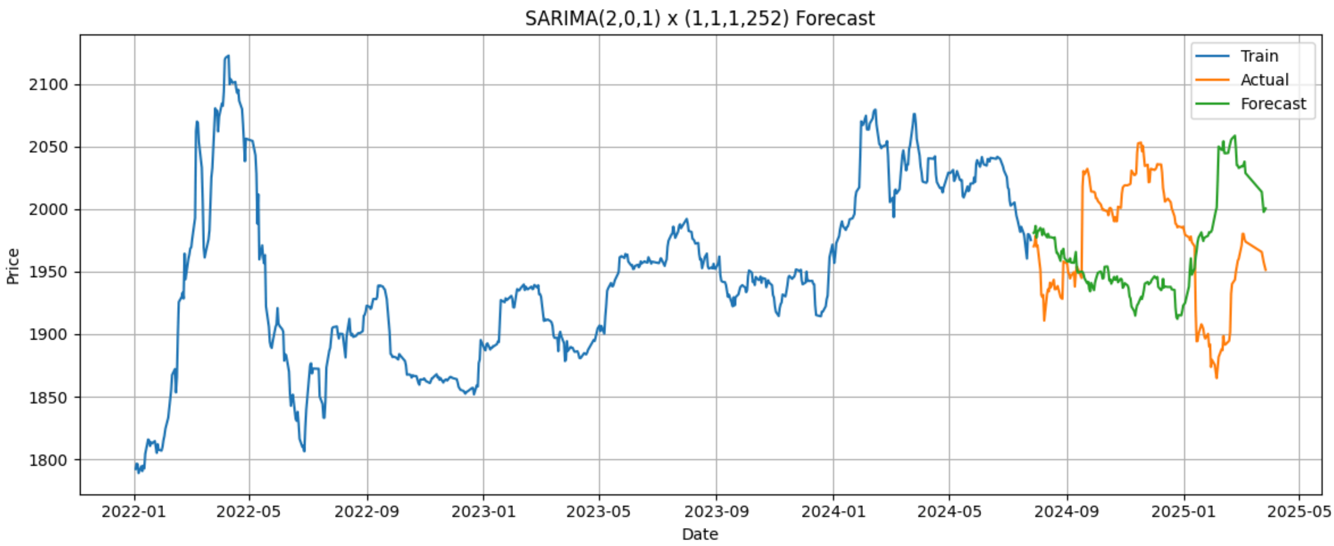
```
# Fit SARIMA model
model = SARIMAX(train,
                order=(p, d, q),
                seasonal_order=(P, D, Q, s),
                enforce_stationarity=False,
                enforce_invertibility=False)

model_fit = model.fit()
```

```
# Forecast
preds = model_fit.forecast(steps=len(test))
```

```
# Evaluate
rmse = np.sqrt(mean_squared_error(test, preds))
print(f"\nSARIMA({p},{d},{q}) x ({P},{D},{Q},{s}) RMSE: {rmse:.2f}")
```

```
SARIMA(2,0,1) x (1,1,1,252) RMSE: 79.40
```

```
# Plot
plt.figure(figsize=(12, 5))
plt.plot(train.index, train, label='Train')
plt.plot(test.index, test, label='Actual')
plt.plot(test.index, preds, label='Forecast')
plt.title(f"SARIMA({p},{d},{q}) x ({P},{D},{Q},{s}) Forecast")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
Start coding or generate with AI.
```