```
# 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-sarima/df_bax_cleaned_to_view_outliers_5y.csv
```

SARIMAX: PRATHIK MOHAN

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

# Load 5-year data
df = pd.read_csv('/kaggle/input/bax-5y-sarima/df_bax_cleaned_to_view_outliers_5y.csv', index_col=0, parse_dates=True)
df.index = pd.to_datetime(df.index)

# Slice last 4 years
df_4y = df.last('4YE')
print(f"Rows in 4 years: {len(df_4y)}")

# Preview
df_4y.head()
```

```
Rows in 4 years: 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

```
# Check correlation with Price
correlations = df_4y.corr()
price_corr = correlations['Price'].sort_values(ascending=False)
print(price_corr)
```

```
Price       1.000000
Low         0.995698
High        0.995506
Open        0.990376
Vol.        0.052195
Change %    0.044944
Name: Price, dtype: float64
```

```
df_4y['Vol.'] = np.log1p(df_4y['Vol.'])  # log(1 + volume)
df_4y.head()
```

	Price	Open	High	Low	Vol.	Change %
Date						
2022-01-03	1792.24	1797.10	1797.10	1792.24	13.361665	-0.28
2022-01-04	1796.49	1790.33	1796.49	1789.90	14.707509	0.24
2022-01-05	1796.08	1797.20	1799.26	1795.84	14.571633	-0.02
2022-01-06	1788.93	1796.08	1796.08	1788.93	13.789383	-0.40
2022-01-09	1794.47	1788.38	1795.43	1788.14	14.115616	0.31

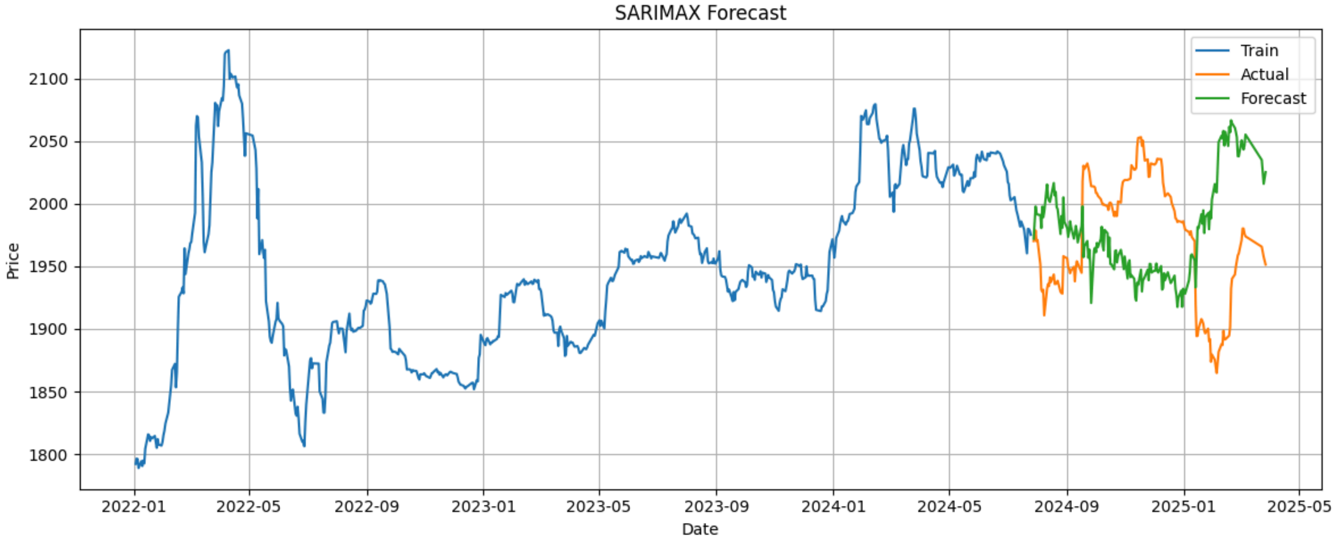
```
# Target
target = df_4y['Price']

# Exogenous variables – based on your correlation decision
exog = df_4y[['Vol.', 'Change %']]  # ✅ adjust based on previous step

# Train-test split (80/20)
train_size = int(len(target) * 0.8)
train_y, test_y = target[:train_size], target[train_size:]
train_X, test_X = exog[:train_size], exog[train_size:]

# Define SARIMAX model
model = SARIMAX(train_y,
                exog=train_X,
                order=(2, 0, 1),
                seasonal_order=(1, 1, 1, 252),
                enforce_stationarity=False,
                enforce_invertibility=False)

# Fit model
model_fit = model.fit()

# Forecast with exog input
preds = model_fit.forecast(steps=len(test_y), exog=test_X)

# Evaluate
rmse = np.sqrt(mean_squared_error(test_y, preds))
print(f"\nSARIMAX RMSE: {rmse:.2f}")
```

```
SARIMAX RMSE: 79.94
```

```
print(df_4y[['Vol.', 'Change %']].dtypes)
print(df_4y[['Vol.', 'Change %']].isna().sum())
```

```
Vol.        float64
Change %    float64
dtype: object
Vol.        0
Change %    0
dtype: int64
```

```
plt.figure(figsize=(12, 5))
plt.plot(train_y.index, train_y, label='Train')
plt.plot(test_y.index, test_y, label='Actual')
plt.plot(test_y.index, preds, label='Forecast')
plt.title('SARIMAX Forecast')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```


SARIMAX Forecast
Train
Actual
Forecast
Price
Date
2100
2050
2000
1950
1900
1850
1800
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05