```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import warnings
```

```
df_bax_m = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/ARIMA_data.csv', index_col=0,parse_dates=True)
df_bax_m.head()
```

Date	Price	Open	High	Low	Vol.	Change %
2023-03-28	1879.66	1878.41	1883.26	1878.41	2740000.0	0.07
2023-03-29	1894.37	1872.81	1895.86	1872.27	8240000.0	0.78
2023-03-30	1886.61	1894.37	1898.59	1886.61	1390000.0	-0.41
2023-04-02	1889.72	1886.61	1889.90	1886.61	2280000.0	0.17
2023-04-03	1889.13	1889.24	1890.20	1888.22	2760000.0	-0.03

```
df = df_bax_m.copy()
series = df['Price']
```

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

```
# Fit ARIMA model
# We'll start with ARIMA(1,1,1) — adjust later
model = ARIMA(train, order=(2, 0, 1))  # p=1, d=1, q=1
model_fit = model.fit()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
```

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

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Predic
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In th
  return get_prediction_index(
```

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

```
ARIMA RMSE: 53.25
```

```
# Plot
plt.figure(figsize=(10, 4))
plt.plot(train.index, train, label='Train')
plt.plot(test.index, test, label='Actual')
plt.plot(test.index, preds, label='Forecast')
plt.title('ARIMA Forecast')
plt.legend()
plt.grid(True)
plt.show()
```

Grid Search for best p,d,q

```
# Grid Search over p, d, q
import itertools
p = d = q = range(0, 3)  # You can expand to range(0, 5) later
pdq_combinations = list(itertools.product(p, d, q))
```

```
best_rmse = float("inf")
best_order = None
```

```
print("Trying combinations of ARIMA(p,d,q):\n")

for order in pdq_combinations:
    try:
        model = ARIMA(train, order=order)
        model_fit = model.fit()
        preds = model_fit.forecast(steps=len(test))
        rmse = np.sqrt(mean_squared_error(test, preds))
        print(f"ARIMA{order} RMSE: {rmse:.2f}")

        if rmse < best_rmse:
            best_rmse = rmse
            best_order = order

    except:
        continue

print(f"\n✅ Best ARIMA Order: {best_order} with RMSE: {best_rmse:.2f}")
```

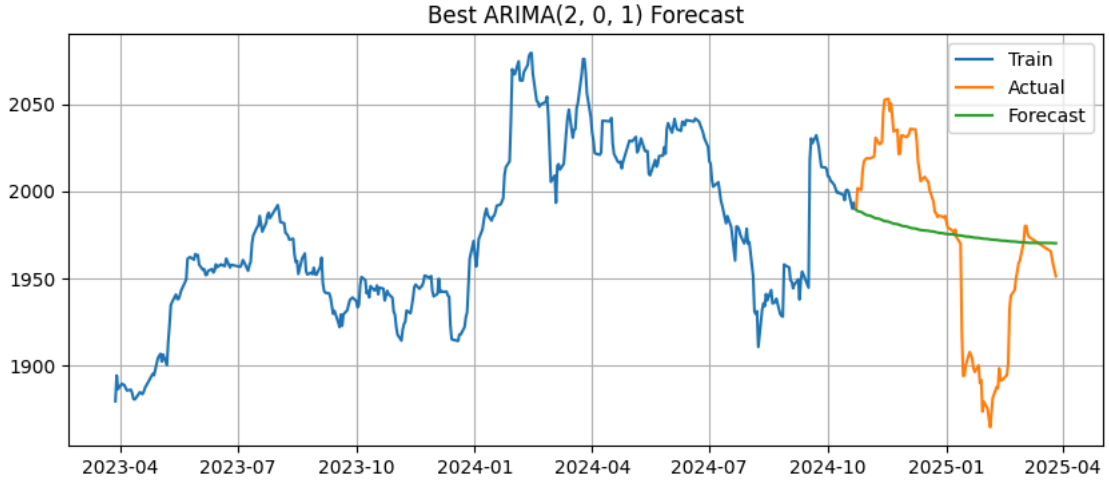
```
Trying combinations of ARIMA(p,d,q):

ARIMA(0, 0, 0) RMSE: 56.84
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Pred
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA paramet
  warn('Non-invertible starting MA parameters found.'
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Pred
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA paramet
```

```
  warn('Non-invertible starting MA parameters found.'
ARIMA(0, 0, 1) RMSE: 56.84
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Pred
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Pred
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but i
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Pred
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In
  return get_prediction_index(
```

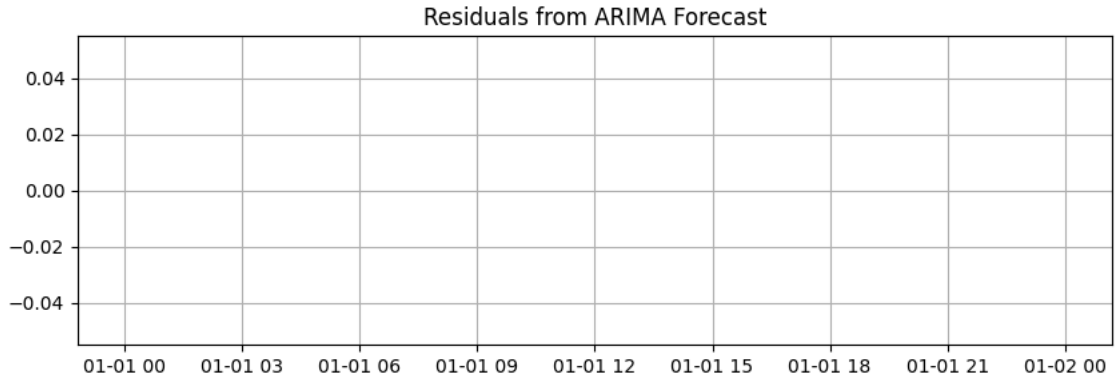
```
# Refit the best model and plot
model = ARIMA(train, order=best_order)
model_fit = model.fit()
preds = model_fit.forecast(steps=len(test))
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: A date index has been provided, but it
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Predic
  return get_prediction_index(
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In th
  return get_prediction_index(
```

```
# Plotting
plt.figure(figsize=(10, 4))
plt.plot(train.index, train, label='Train')
plt.plot(test.index, test, label='Actual')
plt.plot(test.index, preds, label='Forecast')
plt.title(f"Best ARIMA{best_order} Forecast")
plt.legend()
plt.grid(True)
plt.show()
```

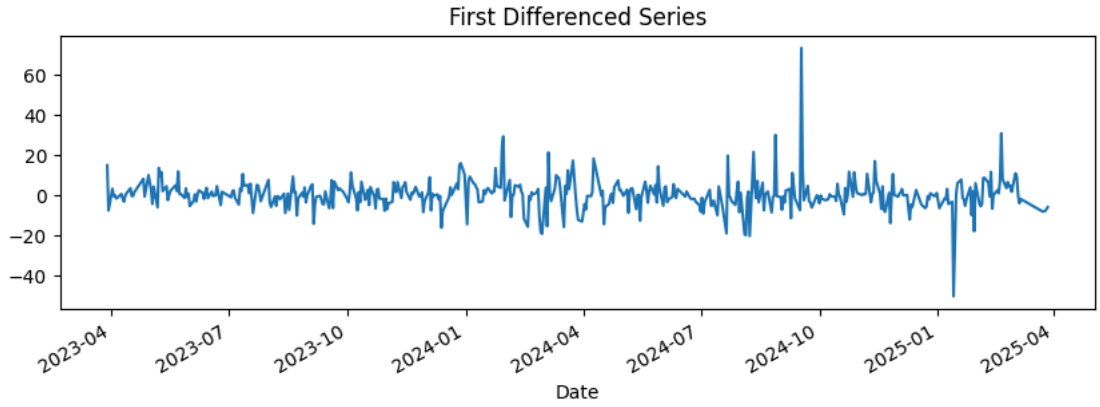



```
residuals = test - preds
plt.figure(figsize=(10, 3))
plt.plot(residuals)
plt.title("Residuals from ARIMA Forecast")
plt.grid(True)
plt.show()

```




```
diff_series = series.diff().dropna()
diff_series.plot(figsize=(10, 3), title="First Differenced Series")

```

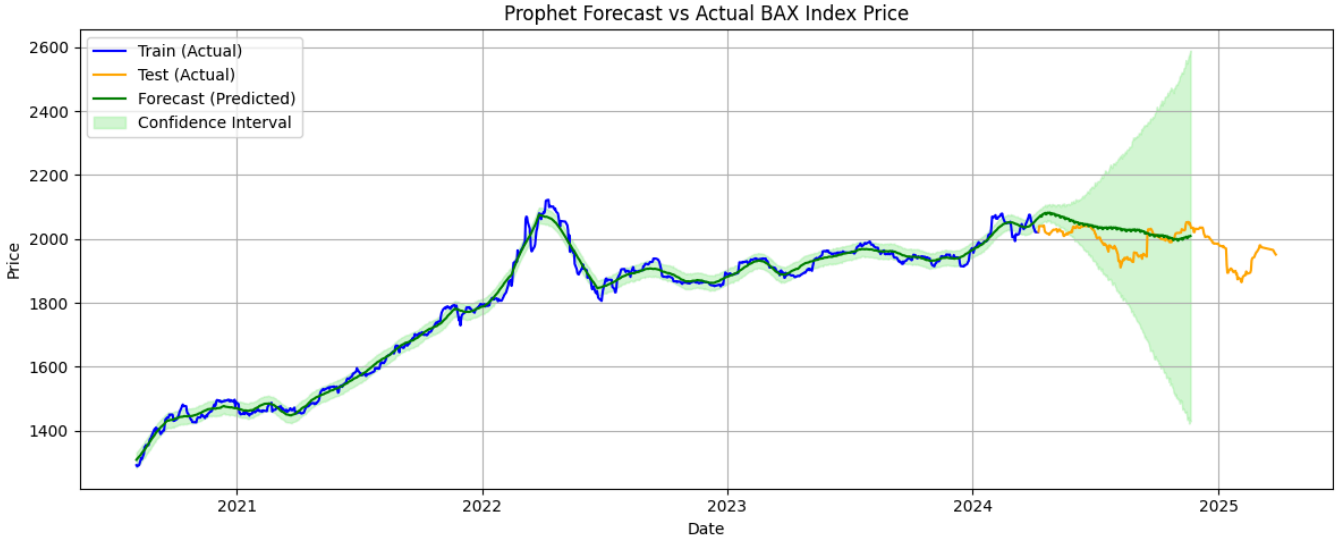
```
<Axes: title={'center': 'First Differenced Series'}, xlabel='Date'>
```



Prophet

```
from prophet import Prophet

df = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Prophet_data.csv', index_col=0,parse_dates=True)

df = df.reset_index()
df = df[['Date', 'Price']].rename(columns={'Date': 'ds', 'Price': 'y'})

# Train-test split
train_size = int(len(df) * 0.8)
train_df = df.iloc[:train_size]
test_df = df.iloc[train_size:]

# Fit Prophet model
m = Prophet(daily_seasonality=True)
m.fit(train_df)

# Forecast into the future
future = m.make_future_dataframe(periods=len(test_df))
forecast = m.predict(future)

# Plot
plt.figure(figsize=(12, 5))
plt.plot(train_df['ds'], train_df['y'], label='Train (Actual)', color='blue')
plt.plot(test_df['ds'], test_df['y'], label='Test (Actual)', color='orange')
plt.plot(forecast['ds'], forecast['yhat'], label='Forecast (Predicted)', color='green')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'],
                 color='lightgreen', alpha=0.4, label='Confidence Interval')

plt.title('Prophet Forecast vs Actual BAX Index Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/x8t56ivb.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/zefdu3s8.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=99509
02:14:14 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
02:14:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```




```
# Extract only forecasted values for the test period
forecast_test = forecast.iloc[-len(test_df):]  # last N predictions

# Actual and predicted values
y_true = test_df['y'].values
y_pred = forecast_test['yhat'].values

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
print(f'Prophet RMSE: {rmse:.2f}')
```

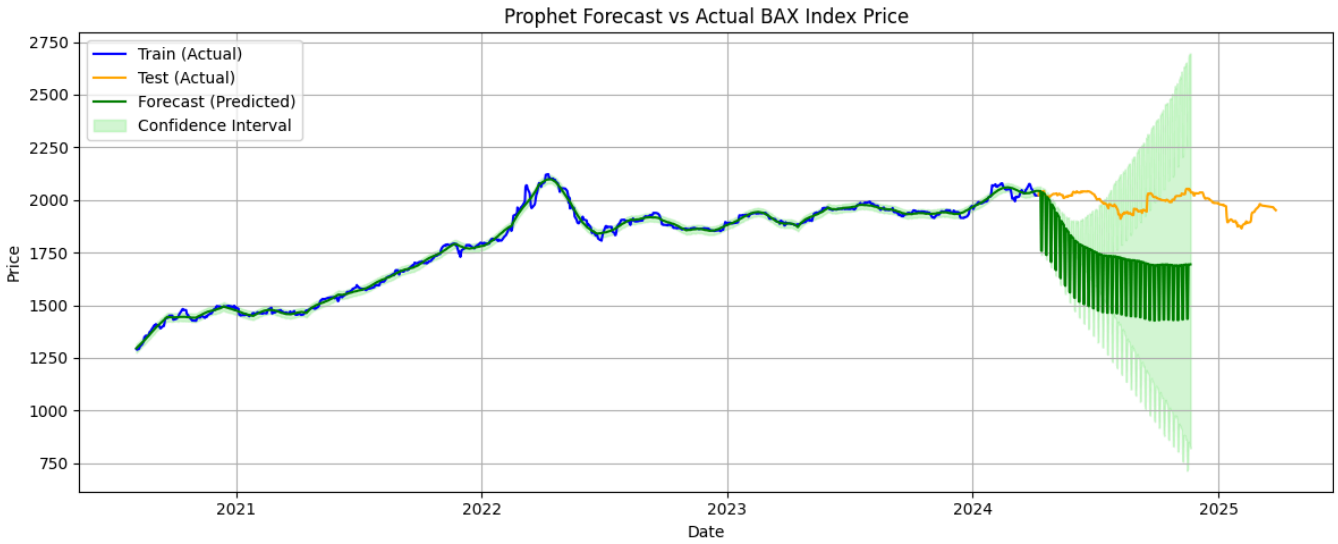
```
Prophet RMSE: 62.88
```

Multiplicative in Prophet

```
from prophet import Prophet

df = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Prophet_data.csv', index_col=0,parse_dates=True)

df = df.reset_index()
df = df[['Date', 'Price']].rename(columns={'Date': 'ds', 'Price': 'y'})

# Train-test split
train_size = int(len(df) * 0.8)
train_df = df.iloc[:train_size]
test_df = df.iloc[train_size:]

# Fit Prophet model
m = Prophet(seasonality_mode='multiplicative', daily_seasonality=True)
m.fit(train_df)

# Forecast into the future
future = m.make_future_dataframe(periods=len(test_df))
forecast = m.predict(future)

# Plot
plt.figure(figsize=(12, 5))
plt.plot(train_df['ds'], train_df['y'], label='Train (Actual)', color='blue')
plt.plot(test_df['ds'], test_df['y'], label='Test (Actual)', color='orange')
plt.plot(forecast['ds'], forecast['yhat'], label='Forecast (Predicted)', color='green')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'],
                 color='lightgreen', alpha=0.4, label='Confidence Interval')

```

```
plt.title('Prophet Forecast vs Actual BAX Index Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/nmdsek2j.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/okrjoetg.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=12946
02:25:22 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
02:25:24 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```




```
# Extract only forecasted values for the test period
forecast_test = forecast.iloc[-len(test_df):]  # last N predictions

# Actual and predicted values
y_true = test_df['y'].values
y_pred = forecast_test['yhat'].values

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
print(f'Prophet RMSE: {rmse:.2f}')
```

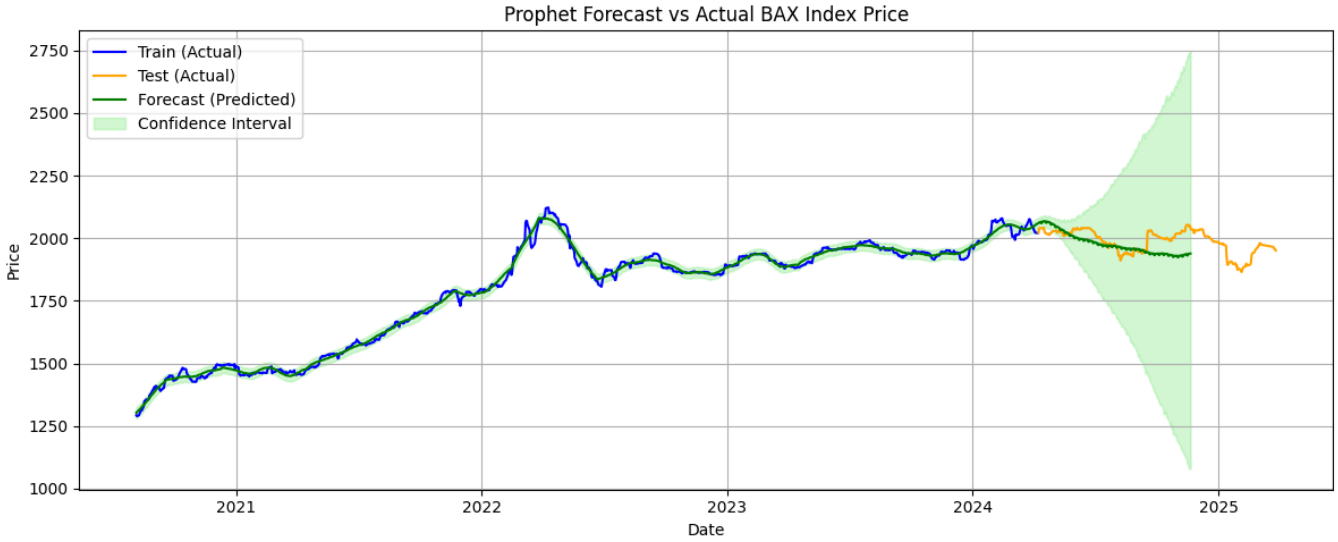
```
Prophet RMSE: 325.42
```

With Changepoint - 0.1

```
from prophet import Prophet

df = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Prophet_data.csv', index_col=0,parse_dates=True)

df = df.reset_index()
df = df[['Date', 'Price']].rename(columns={'Date': 'ds', 'Price': 'y'})

# Train-test split
train_size = int(len(df) * 0.8)
train_df = df.iloc[:train_size]
test_df = df.iloc[train_size:]

# Fit Prophet model
m = Prophet(daily_seasonality=True, changepoint_prior_scale=0.1)  # 0.1-0.5, 0.1 is giving least rmse
m.fit(train_df)

# Forecast into the future
future = m.make_future_dataframe(periods=len(test_df))
forecast = m.predict(future)

# Plot
```

```
plt.figure(figsize=(12, 5))
plt.plot(train_df['ds'], train_df['y'], label='Train (Actual)', color='blue')
plt.plot(test_df['ds'], test_df['y'], label='Test (Actual)', color='orange')
plt.plot(forecast['ds'], forecast['yhat'], label='Forecast (Predicted)', color='green')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'],
                 color='lightgreen', alpha=0.4, label='Confidence Interval')

plt.title('Prophet Forecast vs Actual BAX Index Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/1jbj0dfg.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/t1b7533p.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=1661
02:27:08 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
02:27:09 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```




```
# Extract only forecasted values for the test period
forecast_test = forecast.iloc[-len(test_df):]  # last N predictions

# Actual and predicted values
y_true = test_df['y'].values
y_pred = forecast_test['yhat'].values

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
print(f'Prophet RMSE: {rmse:.2f}')
```

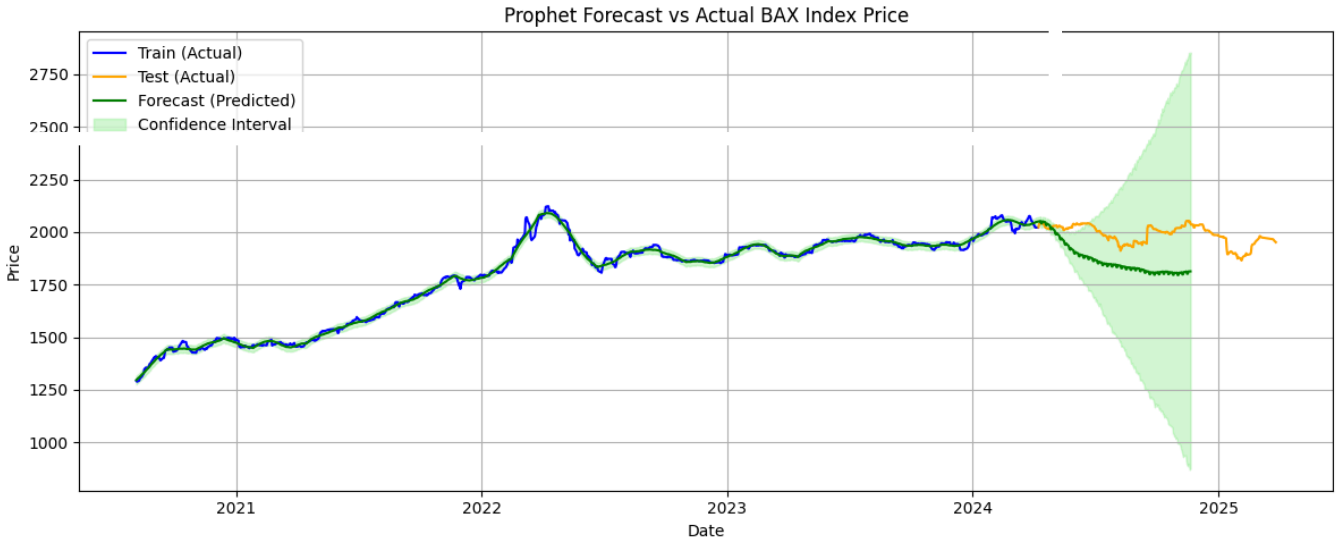
```
Prophet RMSE: 45.83
```

Changepoint 0.2-0.5

```
from prophet import Prophet

df = pd.read_csv(r'/content/drive/MyDrive/PRN23039142546/Prophet_data.csv', index_col=0,parse_dates=True

df = df.reset_index()
df = df[['Date', 'Price']].rename(columns={'Date': 'ds', 'Price': 'y'})

# Train-test split
train_size = int(len(df) * 0.8)
train_df = df.iloc[:train_size]
test_df = df.iloc[train_size:]

# Fit Prophet model
m = Prophet(daily_seasonality=True, changepoint_prior_scale=0.4)
```

```
m.fit(train_df)

# Forecast into the future
future = m.make_future_dataframe(periods=len(test_df))
forecast = m.predict(future)

# Plot
plt.figure(figsize=(12, 5))
plt.plot(train_df['ds'], train_df['y'], label='Train (Actual)', color='blue')
plt.plot(test_df['ds'], test_df['y'], label='Test (Actual)', color='orange')
plt.plot(forecast['ds'], forecast['yhat'], label='Forecast (Predicted)', color='green')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'],
                 color='lightgreen', alpha=0.4, label='Confidence Interval')

plt.title('Prophet Forecast vs Actual BAX Index Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.tight_layout()
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/po5yz61i.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp4giq92fm/uqs3wy2l.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_mode .bin', 'random', 'seed=9579
02:29:03 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
02:29:05 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```




```
# Extract only forecasted values for the test period
forecast_test = forecast.iloc[-len(test_df):]  # last N predictions

# Actual and predicted values
y_true = test_df['y'].values
y_pred = forecast_test['yhat'].values

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
print(f'Prophet RMSE: {rmse:.2f}')
```

```
Prophet RMSE: 133.74
```

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