하이퍼파라미터

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from itertools import product
from sklearn.metrics import mean_absolute_error

# 데이터를 DataFrame으로 읽어옵니다.
data = pd.read_csv("C:/yelin/sp/항생제/시계열/merged_data.csv", parse_dates=["Date"])

# 연도와 월을 추출하여 새로운 열을 추가합니다.
data["Year"] = data["Date"].dt.year
data["Month"] = data["Date"].dt.month

# 'Date' 열을 이용하여 2013년 1월부터 2022년 12월까지의 데이터를 필터링합니다.
start_date = pd.Timestamp('2013-01-01')
end_date = pd.Timestamp('2022-12-31')
filtered_data = data[(data['Date'] >= start_date) & (data['Date'] <= end_date)]

# 월별로 데이터를 그룹화하고 'IMPL_CAPA'의 합계를 계산합니다.
monthly_sum = filtered_data.groupby(filtered_data['Date'].dt.to_period('M'))['IMPL_CAPA'].sum()

# 인덱스를 날짜 형태로 변환합니다.
monthly_sum.index = monthly_sum.index.to_timestamp()

# 하이퍼파라미터 그리드 설정
p_values = [0, 1, 2]  # AR 차수
d_values = [0, 1]     # 차분 차수
q_values = [0, 1, 2]  # MA 차수
P_values = [0, 1]     # 계절성 AR 차수
D_values = [0, 1]     # 계절성 차분 차수
Q_values = [0, 1]     # 계절성 MA 차수
s_values = [12]       # 계절 주기

# 최적 파라미터와 최적 성능 초기화
best_params = None
best_mae = float('inf')

# 하이퍼파라미터 그리드 탐색
for p, d, q, P, D, Q, s in product(p_values, d_values, q_values, P_values, D_values, Q_values, s_values):
    try:
        model = SARIMAX(monthly_sum, order=(p, d, q), seasonal_order=(P, D, Q, s))
        results = model.fit(disp=False)
        forecasted_caps = results.get_forecast(steps=12).predicted_mean
        mae = mean_absolute_error(monthly_sum[-12:], forecasted_caps)

        if mae < best_mae:
            best_mae = mae
            best_params = (p, d, q, P, D, Q, s)
    except:
        continue

# 최적 파라미터 출력
print("Best Parameters:", best_params)
```

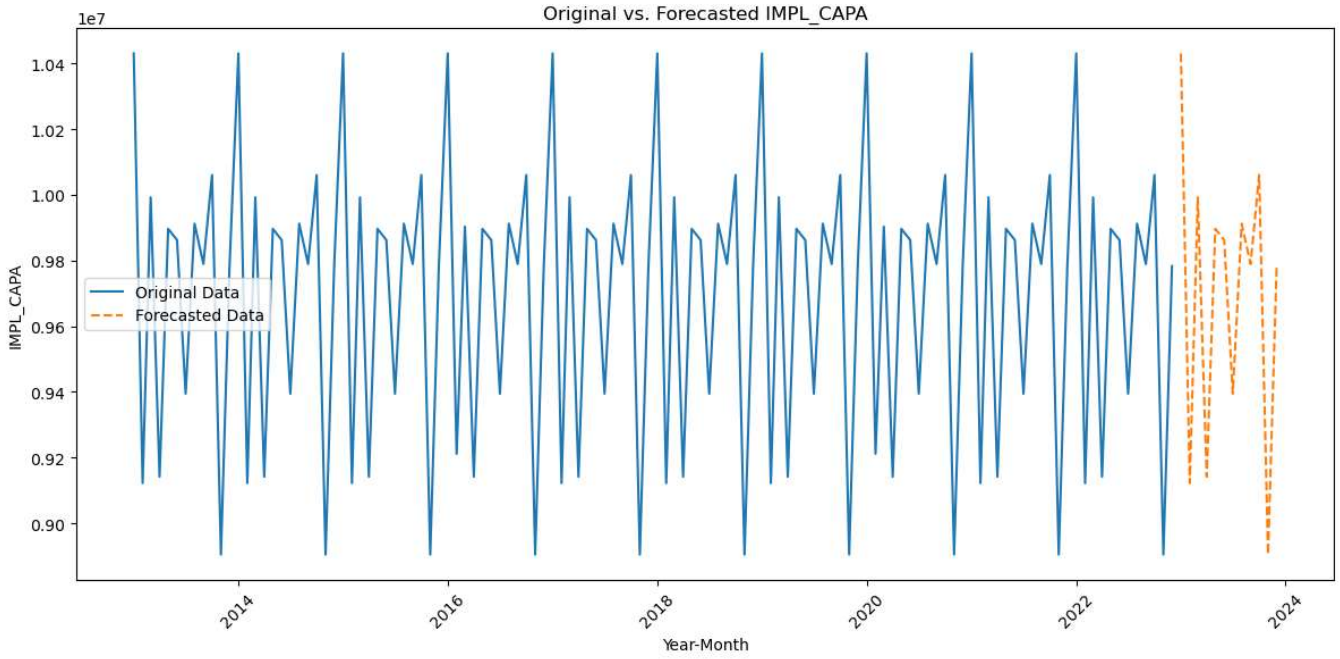
```
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:997: UserWarning: Non-stationary starting seasonal autoregr
  warn('Non-stationary starting seasonal autoregressive'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWbaseWmodel.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:997: UserWarning: Non-stationary starting seasonal autoregr
  warn('Non-stationary starting seasonal autoregressive'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWbaseWmodel.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWbaseWmodel.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "
C:WUsersWuserWanaconda3WlibWsite-packagesWstatsmodelsWtsaWstatespaceWsarimax.py:997: UserWarning: Non-stationary starting seasonal autoregr
  warn('Non-stationary starting seasonal autoregressive'
```

```
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Maximum Likelihood optimization failed to conv
  warnings.warn("Maximum Likelihood optimization failed to "
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
  warn('Non-invertible starting seasonal moving average'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters fou
  warn('Non-invertible starting MA parameters found.'
C:\Users\user\anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:1009: UserWarning: Non-invertible starting seasonal moving
```

10개년

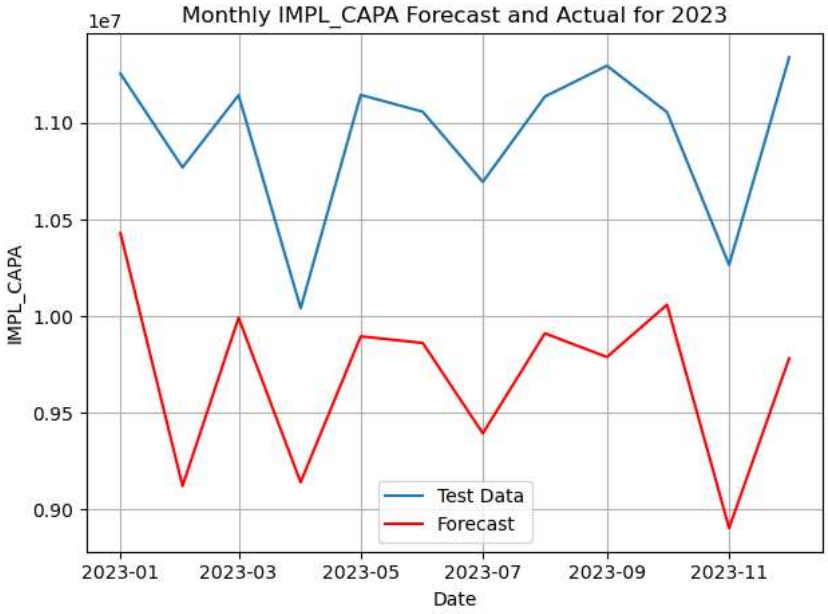
```
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pickle

# 데이터를 DataFrame으로 읽어옵니다.
data = pd.read_csv("C:/yelin/sp/항생제/시계열/merged_data.csv", parse_dates=["Date"])

# 연도와 월을 추출하여 새로운 열을 추가합니다.
data["Year"] = data["Date"].dt.year
data["Month"] = data["Date"].dt.month

# 'Date' 열을 이용하여 2022년 1월부터 2023년 12월까지의 데이터를 필터링합니다.
start_date = pd.Timestamp('2013-01-01')
end_date = pd.Timestamp('2022-12-31')
filtered_data = data[(data['Date'] >= start_date) & (data['Date'] <= end_date)]

# 월별로 데이터를 그룹화하고 'IMPL_CAPA'의 합계를 계산합니다.
monthly_sum = filtered_data.groupby(filtered_data['Date'].dt.to_period('M'))['IMPL_CAPA'].sum()

# 인덱스를 날짜 형태로 변환합니다.
monthly_sum.index = monthly_sum.index.to_timestamp()

# SARIMA 모델을 초기화하고 학습시킵니다.
best_params = (0, 0, 0, 0, 1, 0, 12)  # 최적 하이퍼파라미터 조합
model = SARIMAX(monthly_sum, order=best_params[:3], seasonal_order=best_params[3:])
results = model.fit()

# 모델을 파일로 저장
with open("C:/yelin/sp/항생제/시계열/차/best_sarima_model.pkl", "wb") as f:
    pickle.dump(results, f)

# 2024년의 월별 'IMPL_CAPA' 합계 예측
forecast = results.get_forecast(steps=12)
forecasted_caps = forecast.predicted_mean

# 예측 결과를 데이터프레임으로 만들고 CSV 파일로 저장
forecasted_data = pd.DataFrame({
    "Year-Month": forecasted_caps.index,
    "Forecasted IMPL_CAPA": forecasted_caps.values
})

forecasted_data.to_csv("C:/yelin/sp/항생제/시계열/차/best_forecasted_sum_2024.csv", index=False)


# 원본 데이터와 예측 데이터 시각화
plt.figure(figsize=(12, 6))
```

```
plt.plot(monthly_sum.index, monthly_sum.values, label='Original Data')
plt.plot(forecasted_data["Year-Month"], forecasted_data["Forecasted IMPL_CAPA"], label='Forecasted Data', linestyle='dashed')
plt.xlabel('Year-Month')
plt.ylabel('IMPL_CAPA')
plt.title('Original vs. Forecasted IMPL_CAPA')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pickle
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Test 데이터를 DataFrame으로 읽어옵니다.
test_data = pd.read_csv("C:/yelin/sp/항생제/시계열/test_data.csv", parse_dates=["Date"], encoding='cp949')

# 저장된 모델을 로드합니다.
with open("C:/yelin/sp/항생제/시계열/차/best_sarima_model.pkl", "rb") as f:
    loaded_model = pickle.load(f)

# Test 데이터 중 2023년 1월부터 12월까지의 데이터를 필터링합니다.
test_start_date = pd.Timestamp('2023-01-01')
test_end_date = pd.Timestamp('2023-12-31')
test_filtered_data = test_data[(test_data['Date'] >= test_start_date) & (test_data['Date'] <= test_end_date)]

# 월별로 Test 데이터를 그룹화하고 'IMPL_CAPA'의 총합을 계산합니다.
test_monthly_sum = test_filtered_data.groupby(test_filtered_data['Date'].dt.to_period('M'))['IMPL_CAPA'].sum()


# Test 데이터 예측
forecasted = loaded_model.get_forecast(steps=len(test_monthly_sum))
forecasted_caps = forecasted.predicted_mean

# 예측 결과를 데이터프레임으로 만듭니다.
forecasted_data = pd.DataFrame({
    "Date": test_monthly_sum.index,
    "Forecasted IMPL_CAPA": forecasted_caps.values
})

# 결과를 CSV 파일로 저장합니다.
forecasted_data.to_csv("C:/yelin/sp/항생제/시계열/차/test_forecasted_results_2023.csv", index=False)


plt.plot(test_monthly_sum.index.to_timestamp(), test_monthly_sum.values, label='Test Data')
plt.plot(test_monthly_sum.index.to_timestamp(), forecasted_caps, label='Forecast', color='red')
plt.title("Monthly IMPL_CAPA Forecast and Actual for 2023")
```

```
plt.xlabel("Date")
plt.legend()
plt.ylabel("IMPL_CAPA")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# 상대적 오차 계산
relative_errors = (test_monthly_sum.values - forecasted_caps) / test_monthly_sum.values

# 상대적 오차의 절대값을 취한 후 평균을 계산
mean_relative_error = np.mean(np.abs(relative_errors))

# 평균 상대적 오차를 백분율로 변환
accuracy_percentage = (1 - mean_relative_error) * 100

print(f"모델의 정확성: {accuracy_percentage:.2f}%")
```

```
모델의 정확성: 88.63%
```