ADP_Red\stats\7.time_series.py

```python
## Time Series Analysis
# %% 0. Load Libraries and Dataset
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import scipy.stats as stats

# data encoding type: 'utf-8', 'euc-kr'
df = pd.read_csv('../../ADP_Python/data/서울특별시 코로나19.csv')
date_column = '날짜'

# Check data properties
print(df.head())
print(df.info())

# # Change datatype to datetime
df[date_column] = pd.to_datetime(df[date_column], format='%Y-%m-%d')
df.set_index(date_column, inplace=True)

print(df.dtypes)
print(df.head())

# EDA Visualization
plt.plot(df)
plt.show()


# %% 1. Time Series Decomposition
# (Trend, Seasonality, Residual)
# - 'additive'
# - 'multiplicative'

from statsmodels.tsa.seasonal import seasonal_decompose

decomp_add = seasonal_decompose(df, model='additive')
decomp_mul = seasonal_decompose(df, model='multiplicative')

decomp_add.plot()
decomp_mul.plot()
plt.show()


# %% 2. Stationarize the Series
# %% 2-1. Durbin-Watson Test
from statsmodels.stats.stattools import durbin_watson

print(durbin_watson(df))

# %% 2-2. Augmented Dickey-Fuller Test (d)
# Stationary Test
from statsmodels.tsa.stattools import adfuller

# train, test data split
df_train = df[:'2016-12-01']
df_test = df.drop(df_train.index)
```

```
print(df_train)
print(df_test)

adf = adfuller(df_train, regression='ct')

print(f'ADF Statistic: {adf[0]}')
print(f'p-value: {adf[1]}')

if adf[1] < 0.05:
    print('stationary time-series data')
else:
    print('WARNING: non-stationary time-serie data')
    print('WARNING: differentiation or log transformation needed')

# %% 2-3. Differentiation
# First-order differentiation
df_diff1 = df_train.diff(1)
df_diff1 = df_diff1.dropna()

df_diff1.plot()
plt.show()

adf1 = adfuller(df_diff1)

print(f'ADF Statistic: {adf1[0]}')
print(f'p-value: {adf1[1]}')

# Second-order differentiation
df_diff2 = df_train.diff(2)
df_diff2 = df_diff2.dropna()

df_diff2.plot()
plt.show()

adf2 = adfuller(df_diff2)

print(f'ADF Statistic: {adf2[0]}')
print(f'p-value: {adf2[1]}')

# 2-3. Log Transformation
# 2-4. Box-Cos Transformation


# %% 3. Plot ACF/PACF Charts and Find Optimal Parameters
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# %% 3-1. AR (Auto Regressive) Model: AR(p)
# PACF (p)
plot_pacf(df_diff1)
plt.show()

# %% 3-2. MA (Moving Average) Model: MA(q)
# ACF (q)
plot_acf(df_diff1)
plt.show()

# %% 3-3. Grid Search: p, q
from pmdarima import auto_arima

```

```python
auto_arima_model = auto_arima(df_train,
                              start_p=0, max_p=5,
                              start_q=0, max_q=5,
                              seasonal=True,
                              d=1,
                              trace=True,
                              error_action='ignore',
                              suppress_warnings=True,
                              stepwise=False)


# %% 4. Build the ARIMA Model
# %% 4-0. ARMA Model: AR(p) + MA(q)
# %% 4-1. ARIMA Model: AR(p) + differentiation(d) + MA(q)
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(df_train, order=(5,1,0))
result = model.fit()
result.summary()


# %% 4-2. SARIMA Model

# %% 5. Make Predictions
# %% 5-1. Model Prediction
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12,6))

valid_y = result.predict()
axes[0].plot(valid_y, label='prediction')
axes[0].plot(df_train, label='target')

axes[0].legend(loc='upper left')

# 학습데이터 세트로부터 테스트 데이터 길이(len(df_test))만큼 예측
pred_y = result.forecast(steps=len(df_test), alpha=0.05)

axes[1].plot(pred_y, label='prediction')
axes[1].plot(df_test, label='target')
axes[1].legend(loc='upper right')

plt.tight_layout()
plt.show()

# %% 5-2. Model Evaluation
from sklearn.metrics import mean_squared_error, r2_score

print(f'r2_score: {r2_score(df_test, pred_y)}')                  # R^2
print(f'RMSE: {np.sqrt(mean_squared_error(df_test, pred_y))}')  # Root Mean Squared Error

# %%
true_index = list(df.index)
predict_index = list(df_test.index)

true_value = np.array(list(df.price))

# plot

plt.plot(true_index, true_value, label='True')
plt.plot(predict_index, pred_y, label='Prediction')
plt.vlines(pd.Timestamp('2017-01-01'), 0, 10000, linestyle='--')
```

```
plt.show()

# %% References
# [[머신러닝][시계열] AR, MA, ARMA, ARIMA의 모든 것 - 개념편](https://velog.io/@euisuk-chung/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D%EC%8B%9C%EA%B3%84%EC%97%B4-AR-MA-ARMA-ARIMA%EC%9D%98-%EB%AA%A8%EB%93%A0-%EA%B2%83-%EA%B0%9C%EB%85%90%ED%8E%B8)
# [[머신러닝][시계열] AR, MA, ARMA, ARIMA의 모든 것 - 실습편](https://velog.io/@euisuk-chung/%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D%EC%8B%9C%EA%B3%84%EC%97%B4-AR-MA-ARMA-ARIMA%EC%9D%98-%EB%AA%A8%EB%93%A0-%EA%B2%83-%EC%8B%A4%EC%8A%B5%ED%8E%B8)
```