ปฏิบัติการครั้งที่ 7 กระบวนวิชา 229351 Statistical Learning for Data Science

คำชี้แจง

ให้เริ่มทำปฏิบัติการจาก colab notebook หรือไฟล์ *.ipynb ที่กำหนดให้ จากนั้นบันทึกไว้เป็นไฟล์ *.pdf แล้วส่งใน Assignments

ดาวน์โหลดข้อมูลรถยนต์ชนิดต่างใน link ข้างล่างนี้
https://donlapark.pages.dev/229351/data/elecequip.csv

```
# uploading the csv file to colab

!wget -O elecequip.csv https://donlapark.pages.dev/229351/data/elecequip.csv
```

```
--2025-09-10 14:01:39--  https://donlapark.pages.dev/229351/data/elecequip.csv
Resolving donlapark.pages.dev (donlapark.pages.dev)... 172.66.47.56, 172.66.44.200, 2606:4700:310c::ac42:2f38, ...
Connecting to donlapark.pages.dev (donlapark.pages.dev)|172.66.47.56|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3844 (3.8K) [text/csv]
Saving to: ‘elecequip.csv’

elecequip.csv       100%[===================>]   3.75K  --.-KB/s    in 0s

2025-09-10 14:01:39 (22.8 MB/s) - ‘elecequip.csv’ saved [3844/3844]
```

```
# import module ที่ต้องใช้
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
```

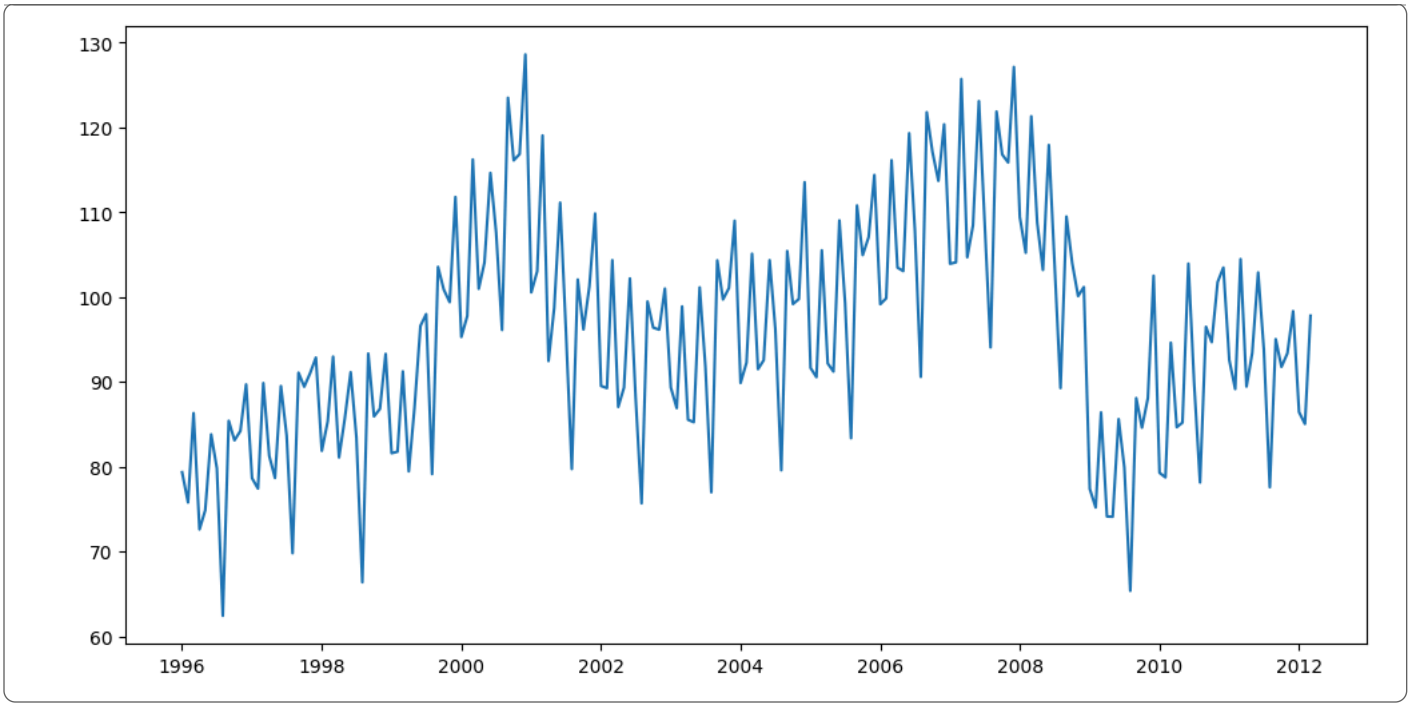
```
# parse_dates ชื่อของคอลัมน์ที่จะเปลี่ยนให้เป็น datetime
# index_col ชื่อของคอลัมน์ที่จะให้เป็น index
# date_parser ฟังก์ชันที่เปลี่ยน string ให้เป็น datetime
data = pd.read_csv('elecequip.csv', parse_dates=['time'],
                                    index_col='time',
                                    date_format='%Y-%m')

data
```

time	Unnamed: 0	value
1996-01-01	1	79.35
1996-02-01	2	75.78
1996-03-01	3	86.32
1996-04-01	4	72.60
1996-05-01	5	74.86
...	...	...
2011-11-01	191	93.37
2011-12-01	192	98.34
2012-01-01	193	86.44
2012-02-01	194	85.04
2012-03-01	195	97.80

195 rows × 2 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
plt.figure(figsize=(12,6))
plt.plot(data["value"]);
```




```
# subsetting data at specified date

data
```

time	Unnamed: 0	value
1996-01-01	1	79.35
1996-02-01	2	75.78
1996-03-01	3	86.32
1996-04-01	4	72.60
1996-05-01	5	74.86
...	...	...
2011-11-01	191	93.37
2011-12-01	192	98.34
2012-01-01	193	86.44
2012-02-01	194	85.04
2012-03-01	195	97.80

195 rows × 2 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
# Add or change values

data.loc['2012-03-02','value'] = 86

data
```

time	Unnamed: 0	value
1996-01-01	1.0	79.35
1996-02-01	2.0	75.78
1996-03-01	3.0	86.32
1996-04-01	4.0	72.60
1996-05-01	5.0	74.86
...	...	...
2011-12-01	192.0	98.34
2012-01-01	193.0	86.44
2012-02-01	194.0	85.04
2012-03-01	195.0	97.80
2012-03-02	NaN	86.00

196 rows × 2 columns

Next steps: Generate code with data | View recommended plots | New interactive sheet

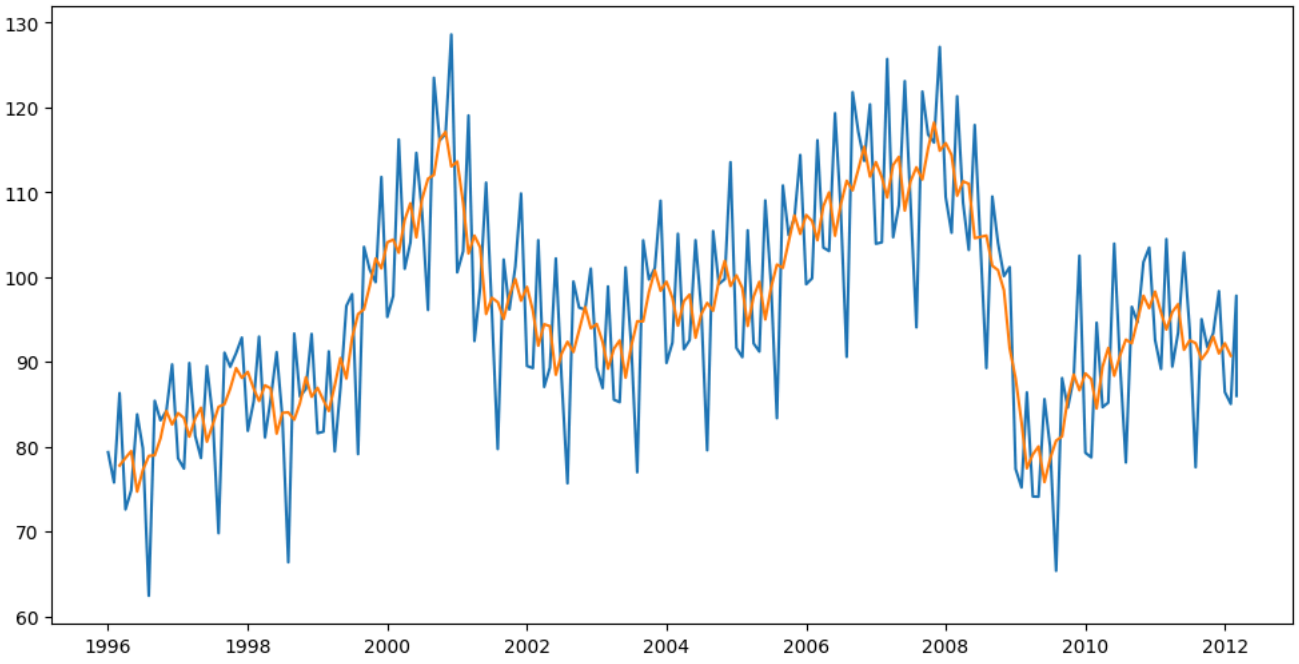
Moving average

```
data['MA'] = data['value'].rolling(window=5,center=True).mean()
#data['MA'] = data['value'].rolling(window=12,center=True).mean().rolling(window=2).mean().shift(-1)

data.head(12)
```

time	Unnamed: 0	value	MA
1996-01-01	1.0	79.35	NaN
1996-02-01	2.0	75.78	NaN
1996-03-01	3.0	86.32	77.782
1996-04-01	4.0	72.60	78.674
1996-05-01	5.0	74.86	79.478
1996-06-01	6.0	83.81	74.696
1996-07-01	7.0	79.80	77.258
1996-08-01	8.0	62.41	78.908
1996-09-01	9.0	85.41	78.988
1996-10-01	10.0	83.11	80.968
1996-11-01	11.0	84.21	84.214
1996-12-01	12.0	89.70	82.616

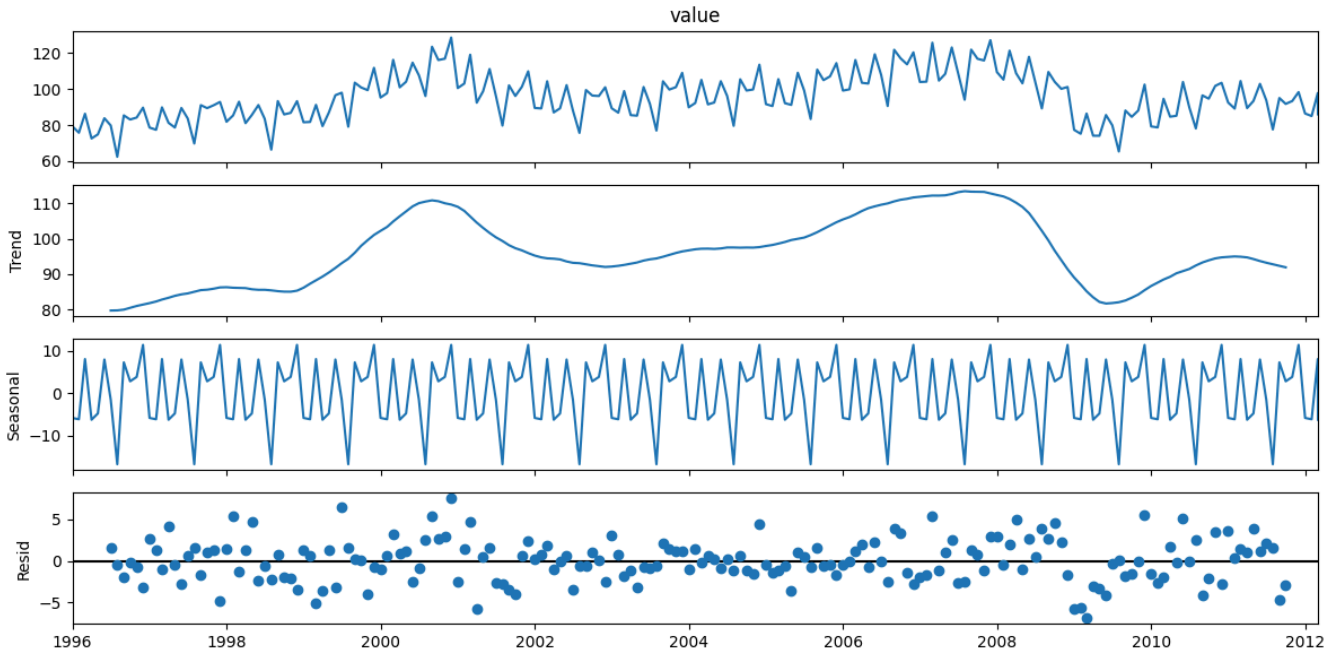
Next steps: Generate code with data | View recommended plots | New interactive sheet

```
plt.figure(figsize=(12,6))
plt.plot(data['value'])
plt.plot(data['MA']);
```

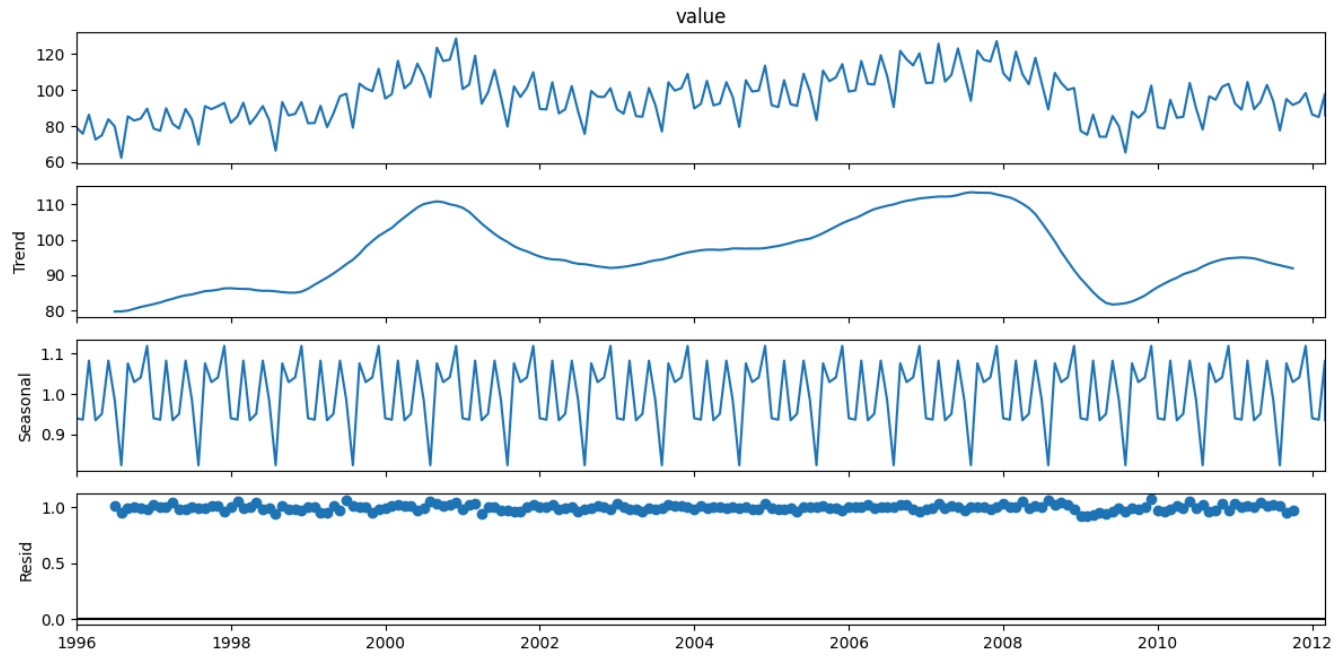


Classical decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib as mpl

mpl.rc("figure", figsize=(12,6))
result_add = seasonal_decompose(data['value'], model='additive', period=12)

result_add.plot();
```




```
result_mul = seasonal_decompose(data['value'], model='multiplicative', period=12)

result_mul.plot();
```



เรียกดูแต่ละส่วน

```
print(result_add.trend)
print(result_add.seasonal)
print(result_add.resid)
print(result_add.observed)
```

```
time
1996-01-01   NaN
1996-02-01   NaN
1996-03-01   NaN
1996-04-01   NaN
1996-05-01   NaN
              ..
2011-12-01   NaN
2012-01-01   NaN
2012-02-01   NaN
2012-03-01   NaN
2012-03-02   NaN
Name: trend, Length: 196, dtype: float64
time
1996-01-01    -5.870942
1996-02-01    -6.182553
1996-03-01     8.099891
1996-04-01    -6.298248
1996-05-01    -4.801748
                ...
2011-12-01    11.464224
2012-01-01    -5.870942
2012-02-01    -6.182553
2012-03-01     8.099891
2012-03-02    -6.298248
Name: seasonal, Length: 196, dtype: float64
time
1996-01-01   NaN
1996-02-01   NaN
1996-03-01   NaN
1996-04-01   NaN
1996-05-01   NaN
              ..
2011-12-01   NaN
2012-01-01   NaN
2012-02-01   NaN
2012-03-01   NaN
2012-03-02   NaN
Name: resid, Length: 196, dtype: float64
```

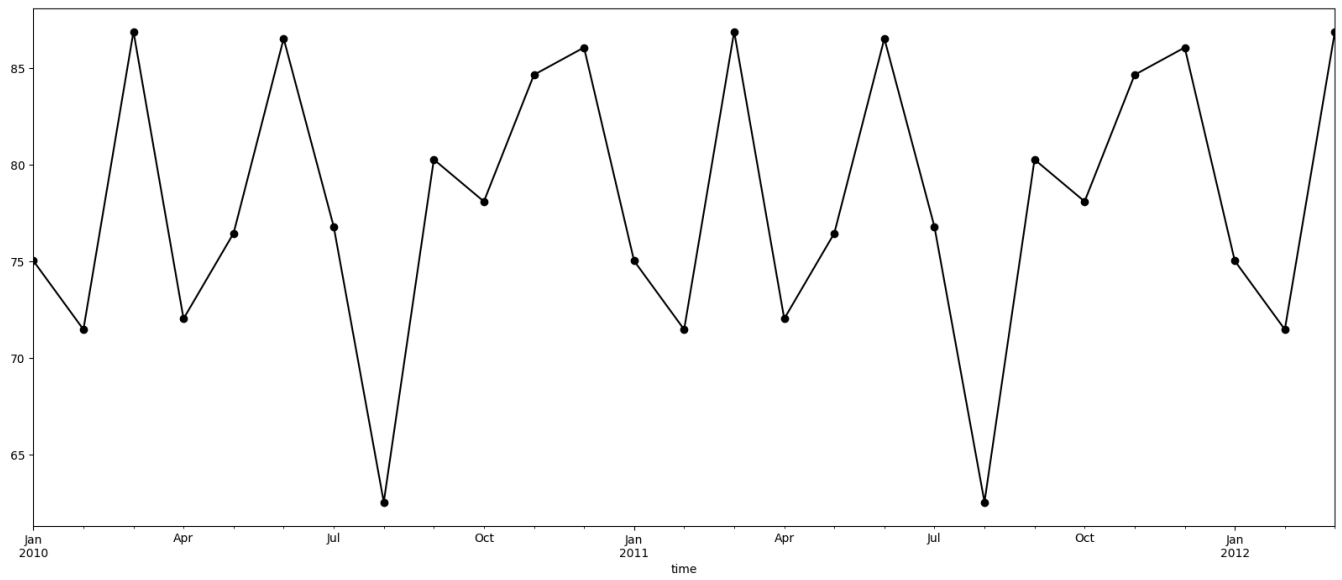
```
time
1996-01-01    79.35
1996-02-01    75.78
1996-03-01    86.32
1996-04-01    72.60
1996-05-01    74.86
              ...
2011-12-01    98.34
2012-01-01    86.44
2012-02-01    85.04
2012-03-01    97.80
2012-03-02    86.00
Name: value, Length: 196, dtype: float64
```

ปฏิบัติการครั้งที่ 7

1. สร้างโมเดลเพื่อการทำนายดังนี้ 1.1 แบ่งข้อมูลออกเป็น 2 ส่วน
   - training set: วันที่ 1996-01-01 ถึง 2009-12-01
   - test set: วันที่ 2010-01-01 ถึง 2012-03-01
     1.2 แยกส่วนประกอบ $y_t = S_t + T_t + R_t$ บน training set 1.3 สร้าง time series ชุดใหม่ที่แสดงถึงทำนายค่าบน test set โดยนำค่า $T_t + R_t$ ของวันล่าสุดใน training set ที่มีค่า T_t มาบวกกับแต่ละค่าใน S_t จากวันที่ วันที่ 2010-01-01 ถึง 2012-03-01
2. คำนวณ RMSE โดยใช้ฟังก์ชัน `rmse` ข้างล่าง
3. แสดงแผนภาพข้อมูล elecequip และค่าทำนายที่ได้

```
def rmse(y_true,y_pred):
  # Inputs
  # y_true: actual value, y_pred: predicted values
  # Output: RMSE between y_true and y_pred
  return np.sqrt(np.mean(np.square(y_true-y_pred)))
```

```
ax = ts_new.plot(marker='o', color='black', figsize=(20,8))
```



Reasoning: Split the data into training and test sets based on the specified date ranges and display the head and tail of each to verify the split.

```
train_data = data.loc['1996-01-01':'2009-12-01'].copy()
test_data = data.loc['2010-01-01':'2012-03-01'].copy()
```

```
print("Train Data Head:")
display(train_data.head())
print("\nTrain Data Tail:")
display(train_data.tail())

print("\nTest Data Head:")
display(test_data.head())
print("\nTest Data Tail:")
display(test_data.tail())
```

```
Train Data Head:
```

time	Unnamed: 0	value	MA
1996-01-01	1.0	79.35	NaN
1996-02-01	2.0	75.78	NaN
1996-03-01	3.0	86.32	77.782
1996-04-01	4.0	72.60	78.674
1996-05-01	5.0	74.86	79.478

```
Train Data Tail:
```

time	Unnamed: 0	value	MA
2009-08-01	164.0	65.36	80.712
2009-09-01	165.0	88.09	81.208
2009-10-01	166.0	84.60	85.732
2009-11-01	167.0	88.09	88.516
2009-12-01	168.0	102.52	86.646

```
Test Data Head:
```

time	Unnamed: 0	value	MA
2010-01-01	169.0	79.28	88.650
2010-02-01	170.0	78.74	87.964
2010-03-01	171.0	94.62	84.500
2010-04-01	172.0	84.66	89.432
2010-05-01	173.0	85.20	91.658

```
Test Data Tail:
```

time	Unnamed: 0	value	MA
2011-11-01	191.0	93.37	92.992
2011-12-01	192.0	98.34	90.992
2012-01-01	193.0	86.44	92.198
2012-02-01	194.0	85.04	90.724
2012-03-01	195.0	97.80	NaN

Decompose the training set

```
from statsmodels.tsa.seasonal import seasonal_decompose

result_train_add = seasonal_decompose(train_data['value'], model='additive', period=12)

print("Additive Decomposition of Training Data:")
print("Trend:")
print(result_train_add.trend.head())
print("\nSeasonal:")
print(result_train_add.seasonal.head())
```

```
print("\nResidual:")
print(result_train_add.resid.head())
print("\nObserved:")
print(result_train_add.observed.head())
```

```
Additive Decomposition of Training Data:
Trend:
time
1996-01-01   NaN
1996-02-01   NaN
1996-03-01   NaN
1996-04-01   NaN
1996-05-01   NaN
Name: trend, dtype: float64

Seasonal:
time
1996-01-01   -5.985857
1996-02-01   -5.957941
1996-03-01    8.183245
1996-04-01   -6.462428
1996-05-01   -5.041178
Name: seasonal, dtype: float64

Residual:
time
1996-01-01   NaN
1996-02-01   NaN
1996-03-01   NaN
1996-04-01   NaN
1996-05-01   NaN
Name: resid, dtype: float64

Observed:
time
1996-01-01    79.35
1996-02-01    75.78
1996-03-01    86.32
1996-04-01    72.60
1996-05-01    74.86
Name: value, dtype: float64
```

Generate predictions

```
last_trend = result_train_add.trend.dropna().iloc[-1]
last_resid = result_train_add.resid.dropna().iloc[-1]
last_tr_sum = last_trend + last_resid
print(f'Last non-null trend value: {last_trend}')
print(f'Last non-null residual value: {last_resid}')
print(f'Sum of last trend and residual: {last_tr_sum}')

result_test_add = seasonal_decompose(test_data['value'], model='additive', period=12)
seasonal_test = result_test_add.seasonal

ts_new = seasonal_test + last_tr_sum

print("\nPredicted time series (ts_new) head:")
print(ts_new.head())
```

```
Last non-null trend value: 81.725
Last non-null residual value: -3.6679887820512764
Sum of last trend and residual: 78.05701121794871

Predicted time series (ts_new) head:
time
2010-01-01    75.016838
2010-02-01    71.467671
2010-03-01    86.871838
2010-04-01    72.023921
2010-05-01    76.445171
Name: seasonal, dtype: float64
```

Calculate rmse

```
rmse_value = rmse(test_data['value'], ts_new)
print(f'RMSE: {rmse_value}')
```

```
RMSE: 14.395922913666372
```

Visualize results

```
plt.figure(figsize=(12,6))
plt.plot(data['value'], label='Original Data')
plt.plot(train_data['value'], label='Training Data')
plt.plot(test_data['value'], label='Test Data')
plt.plot(ts_new, label='Predictions')
plt.title('Original Data, Training Data, Test Data, and Predictions')
plt.xlabel('Time')
plt.ylabel('Value')
plt.legend()
plt.show()
```