Singapore Household Electricity Consumption Prediction

Mak Kwok Fai

SG-DSI-18

Agena

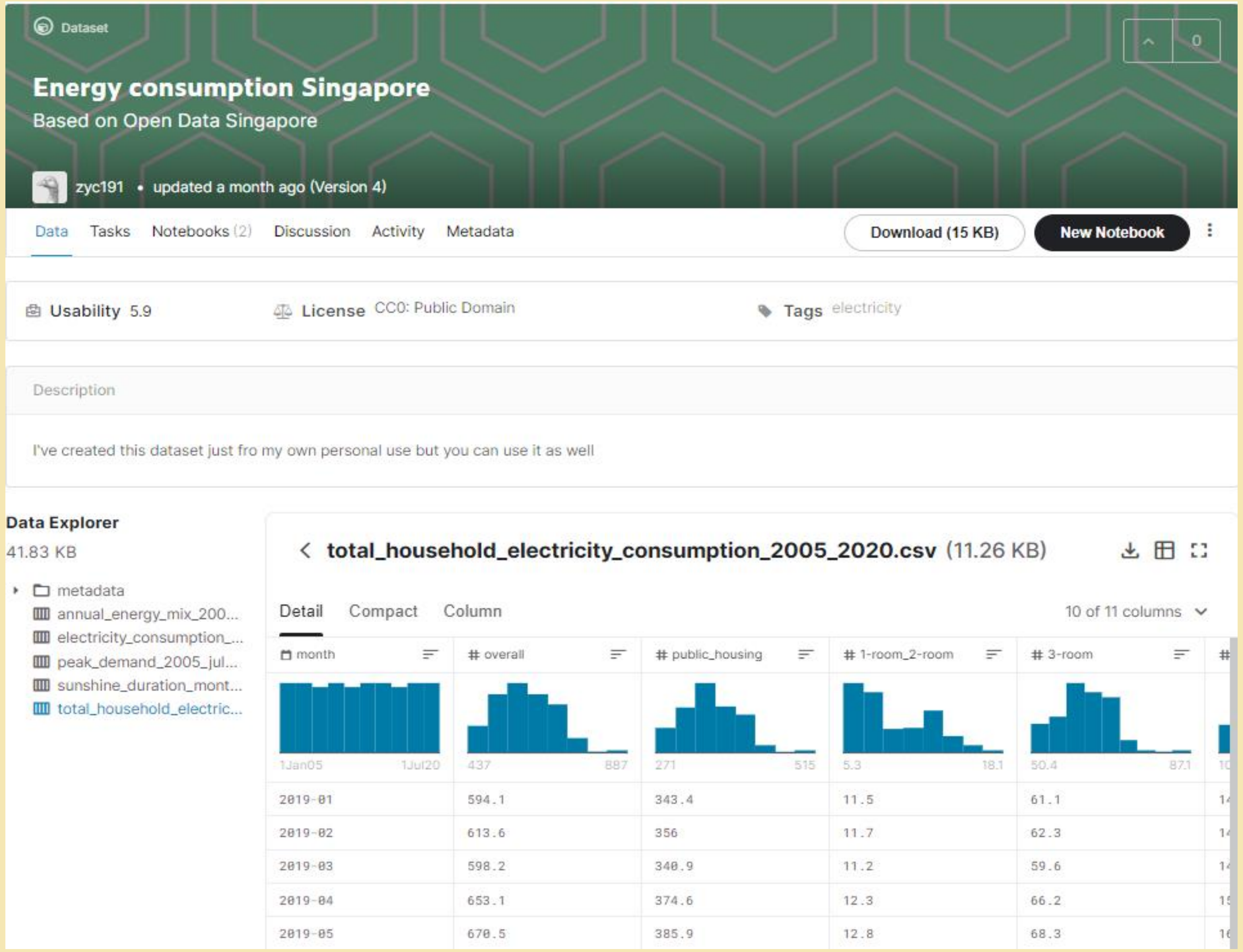
- Problem Statement
- Data & Preprocessing
- Differencing / Dickey Fuller Test
- Train / Test Split
- Modeling
- Reverse Differencing
- Conclusion

Problem Statement

As a data scientist engaged by Singapore Power, I am going to analyze the trend of household electricity consumption in Singapore.

Based on the data from 2005 to 2020, I am going to predict the household electricity consumption for the future.

Data Sources



Energy consumption Singapore

Based on Open Data Singapore

zyc191 • updated a month ago (Version 4)

Usability 5.9

License CC0: Public Domain

Tags electricity

Description

I've created this dataset just fro my own personal use but you can use it as well

Data Explorer

41.83 KB

- metadata
- annual_energy_mix_200...
- electricity_consumption_...
- peak_demand_2005_jul...
- sunshine_duration_mont...
- total_household_electric...

total_household_electricity_consumption_2005_2020.csv (11.26 KB)

month	# overall	# public_housing	# 1-room_2-room	# 3-room	#
1Jan05 1Jul20	437 887	271 515	5.3 18.1	50.4 87.1	10
2019-01	594.1	343.4	11.5	61.1	14
2019-02	613.6	356	11.7	62.3	14
2019-03	598.2	340.9	11.2	59.6	14
2019-04	653.1	374.6	12.3	66.2	15
2019-05	670.5	385.9	12.8	68.3	16

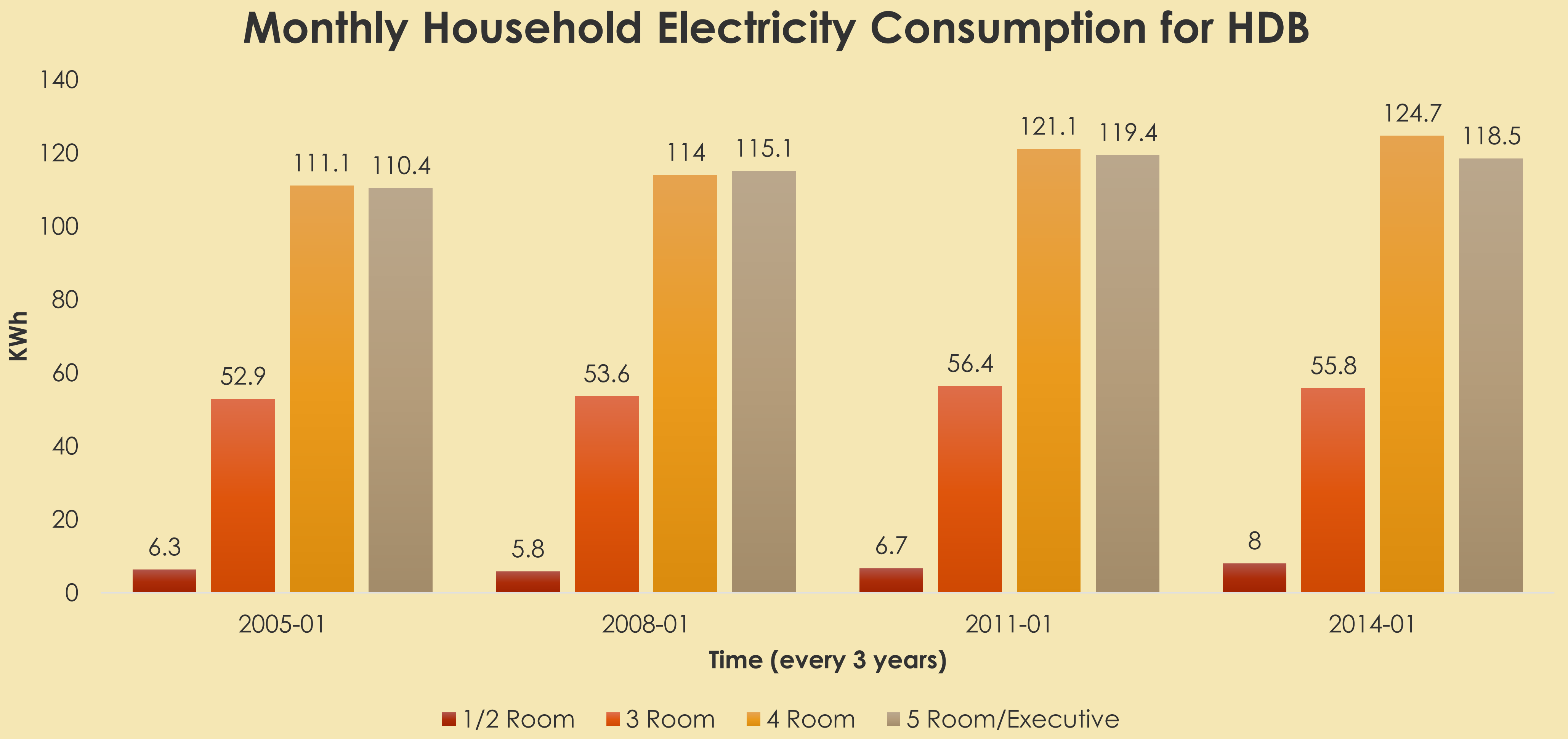
Data Sources

Data	Frequency	Duration
Sunshine	Monthly	1982 - 2020
Household Electricity Consumption	Monthly	2005 - 2020
Electricity Consumption for different Sectors	Yearly	2005 – 2020
Peak Electricity Demand	Yearly	2005 – 2020
Sources of Electricity	Yearly	2005 - 2020

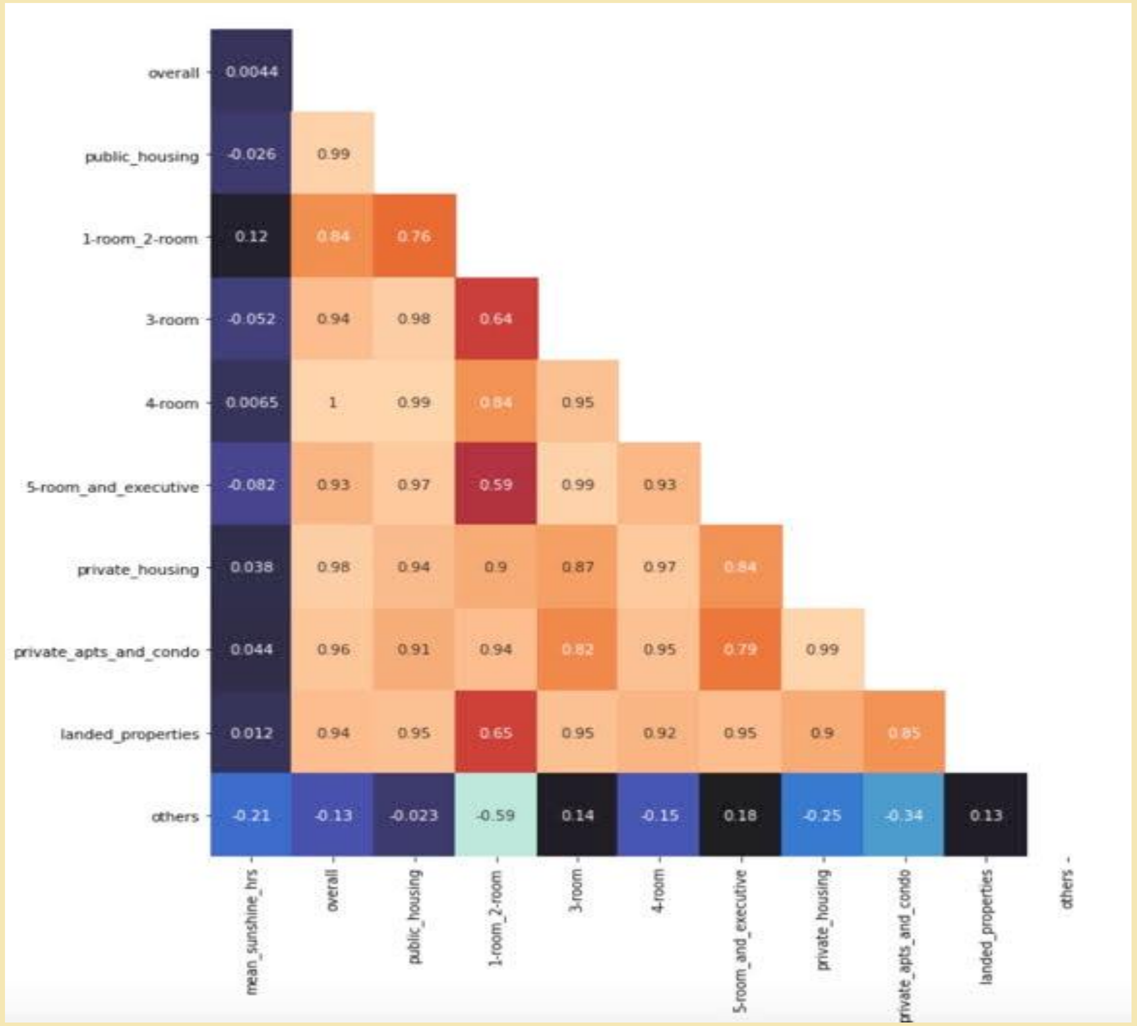
Data Sources

- Public Housing
  - 1-room_2-room
  - 3-room
  - 4-room
  - 5-room_and_executive
- Private Housing
  - Private_apts_and_condo
  - Landed_properties
- others

Data Sources

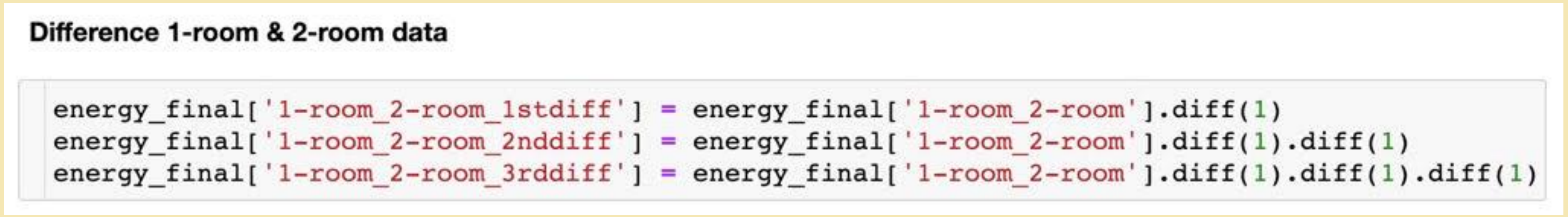


Data Cleaning



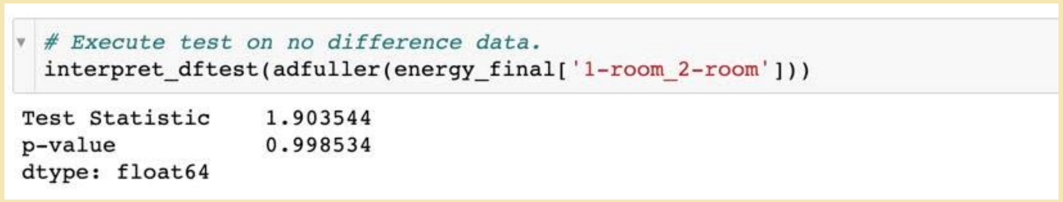
	mean_sunshine_hrs	overall	public_housing	1-room_2-room	3-room	4-room	5-room_and_executive	private_housing	private_apts_and_condo	landed_properties	others
overall	0.0044										
public_housing	-0.026	0.99									
1-room_2-room	0.12	0.84	0.76								
3-room	-0.052	0.94	0.98	0.64							
4-room	0.0065	1	0.99	0.84	0.95						
5-room_and_executive	-0.082	0.93	0.97	0.59	0.99	0.93					
private_housing	0.038	0.98	0.94	0.9	0.87	0.97	0.84				
private_apts_and_condo	0.044	0.96	0.91	0.94	0.82	0.95	0.79	0.99			
landed_properties	0.012	0.94	0.95	0.65	0.95	0.92	0.95	0.9	0.85		
others	-0.21	-0.13	-0.023	-0.59	0.14	-0.15	0.18	-0.25	-0.34	0.13	

Differencing / Dickey Fuller Test



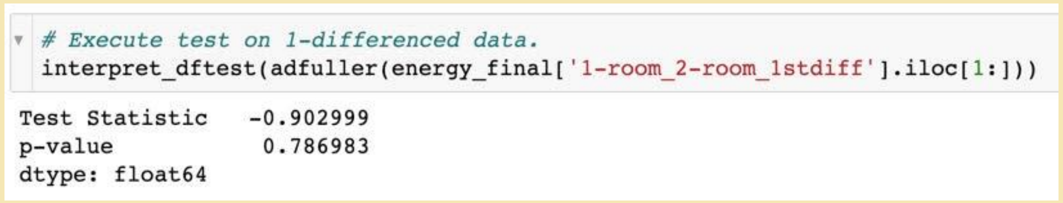
Difference 1-room & 2-room data

```
energy_final['1-room_2-room_1stdiff'] = energy_final['1-room_2-room'].diff(1)
energy_final['1-room_2-room_2nddiff'] = energy_final['1-room_2-room'].diff(1).diff(1)
energy_final['1-room_2-room_3rddiff'] = energy_final['1-room_2-room'].diff(1).diff(1).diff(1)
```



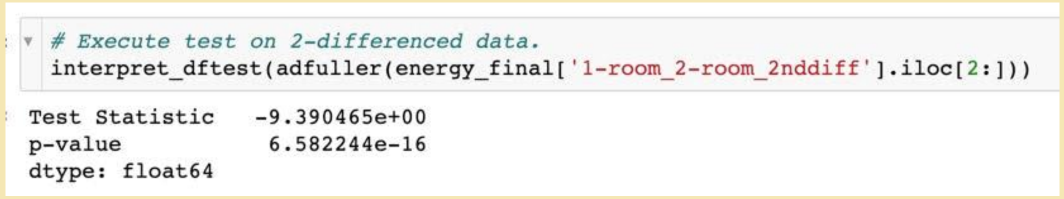
```
# Execute test on no difference data.
interpret_dftest(adfuller(energy_final['1-room_2-room']))
```

```
Test Statistic    1.903544
p-value           0.998534
dtype: float64
```



```
# Execute test on 1-differenced data.
interpret_dftest(adfuller(energy_final['1-room_2-room_1stdiff'].iloc[1:]))
```

```
Test Statistic   -0.902999
p-value           0.786983
dtype: float64
```

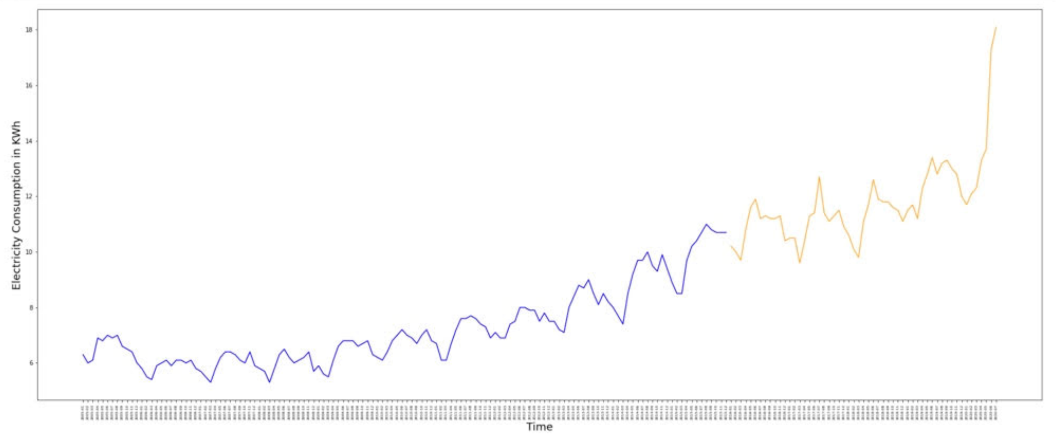


```
# Execute test on 2-differenced data.
interpret_dftest(adfuller(energy_final['1-room_2-room_2nddiff'].iloc[2:]))
```

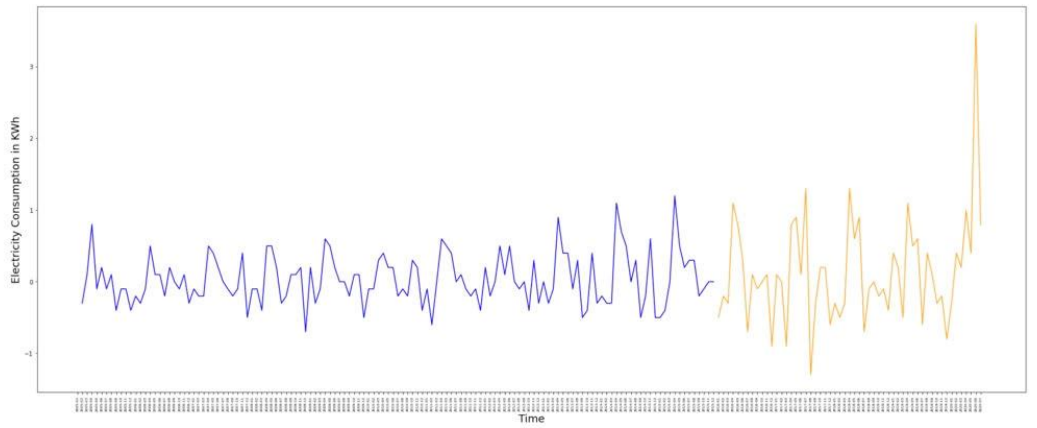
```
Test Statistic   -9.390465e+00
p-value           6.582244e-16
dtype: float64
```

Train/Test Split

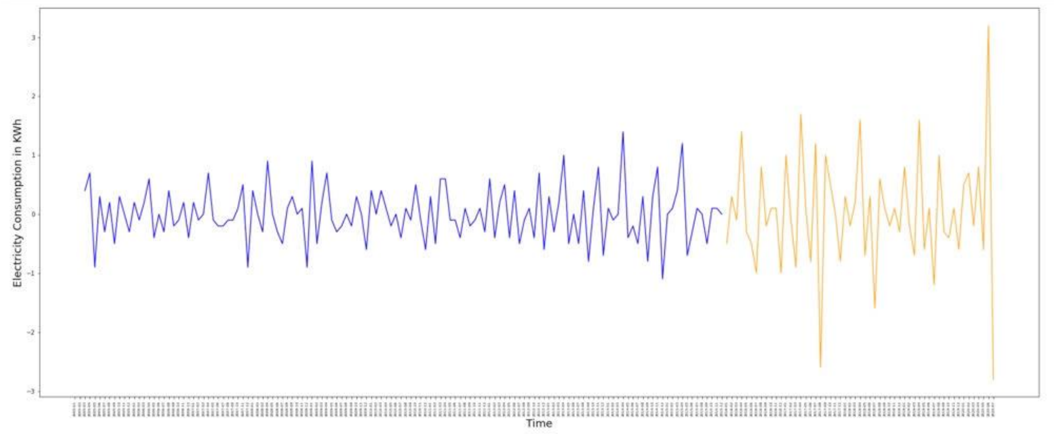
```
# Train/test split for original data

train_1room_2room = energy_final['1-room_2-room'].iloc[:132,]
test_1room_2room = energy_final['1-room_2-room'].iloc[132:,]

train_test_plot(train_1room_2room , test_1room_2room)
```



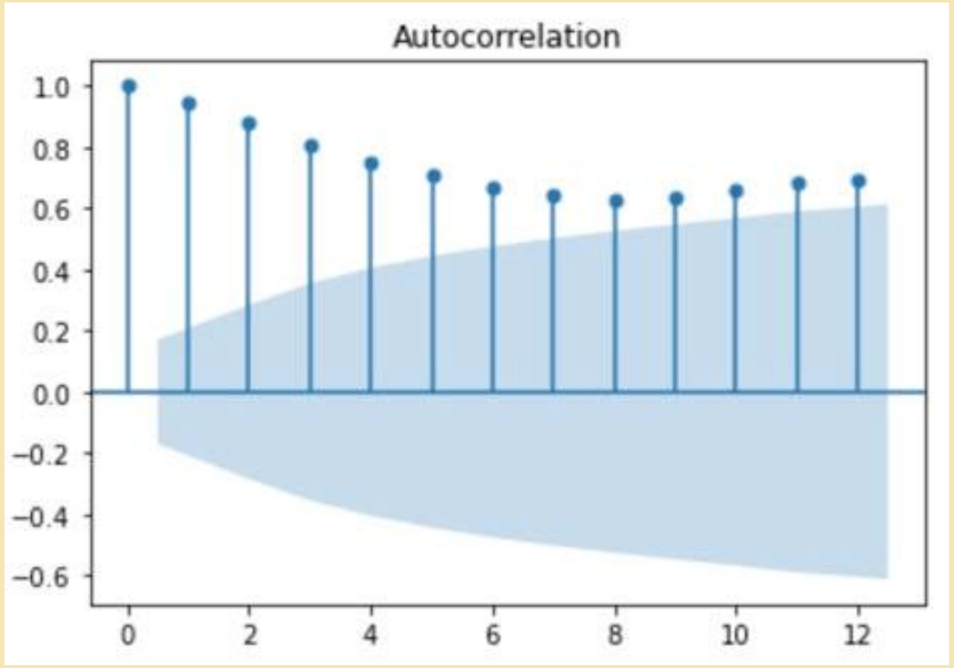
```
# Train/test split for 1st-differenced data

train_1room_2room_1stdiff = energy_final['1-room_2-room_1stdiff'].iloc[:132,]
test_1room_2room_1stdiff = energy_final['1-room_2-room_1stdiff'].iloc[132:,]

train_test_plot(train_1room_2room_1stdiff , test_1room_2room_1stdiff)
```

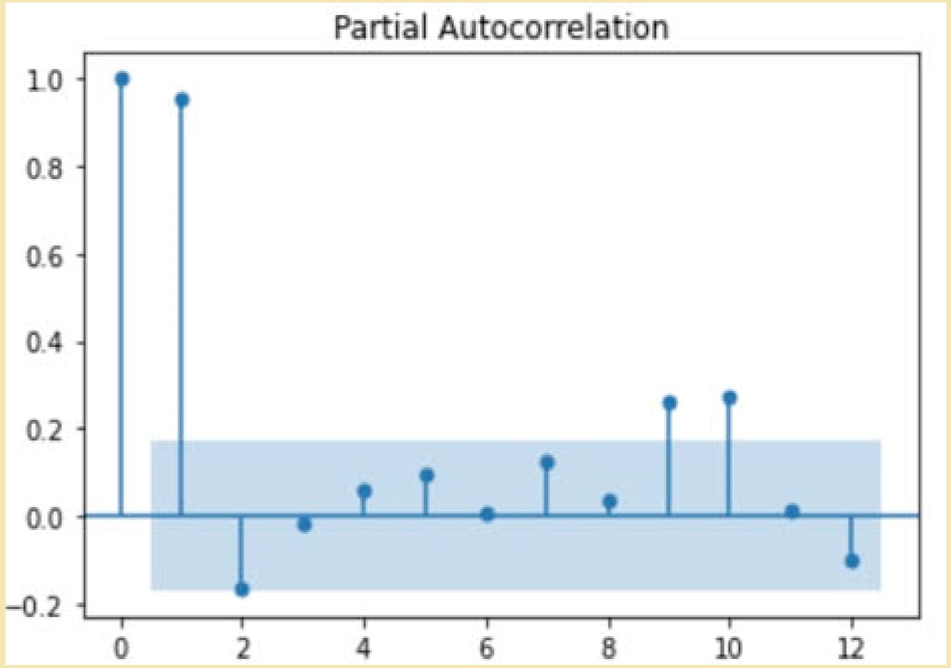


```
# Train/test split for 2nd-differenced data

train_1room_2room_2nddiff = energy_final['1-room_2-room_2nddiff'].iloc[:132,]
test_1room_2room_2nddiff = energy_final['1-room_2-room_2nddiff'].iloc[132:,]

train_test_plot(train_1room_2room_2nddiff , test_1room_2room_2nddiff)
```

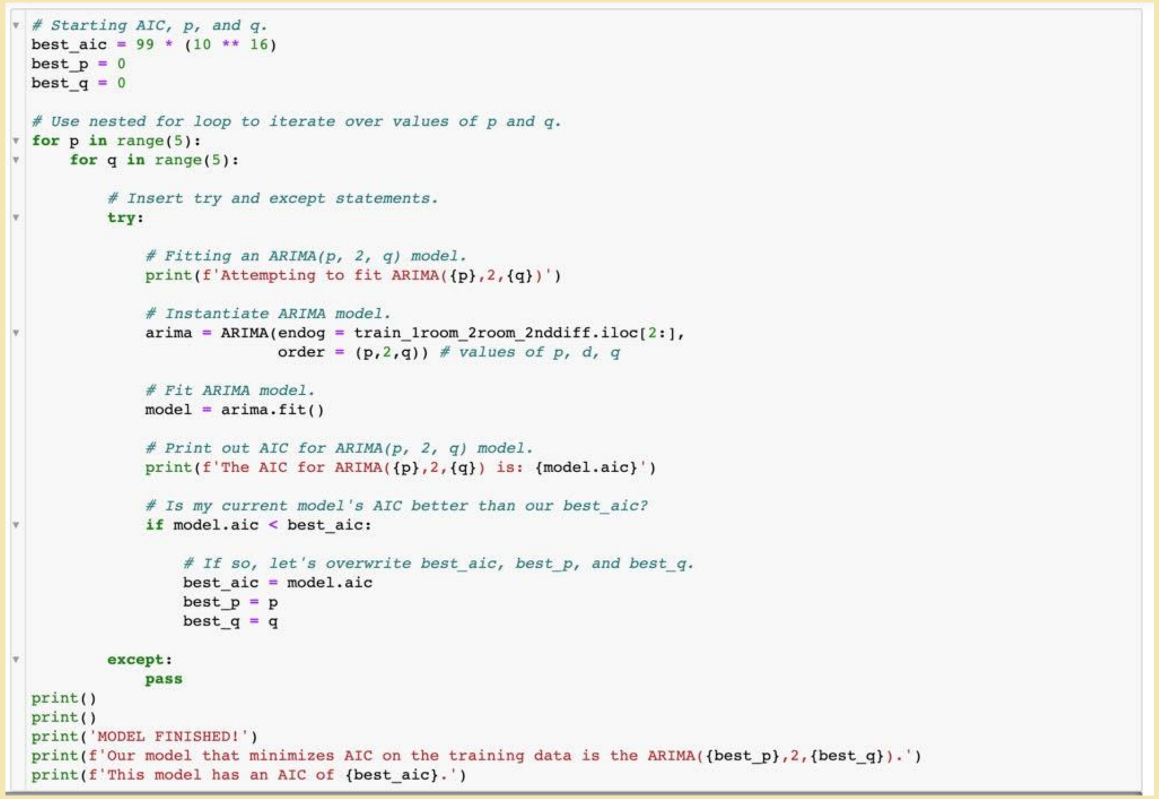


Modeling - ACF & PACF

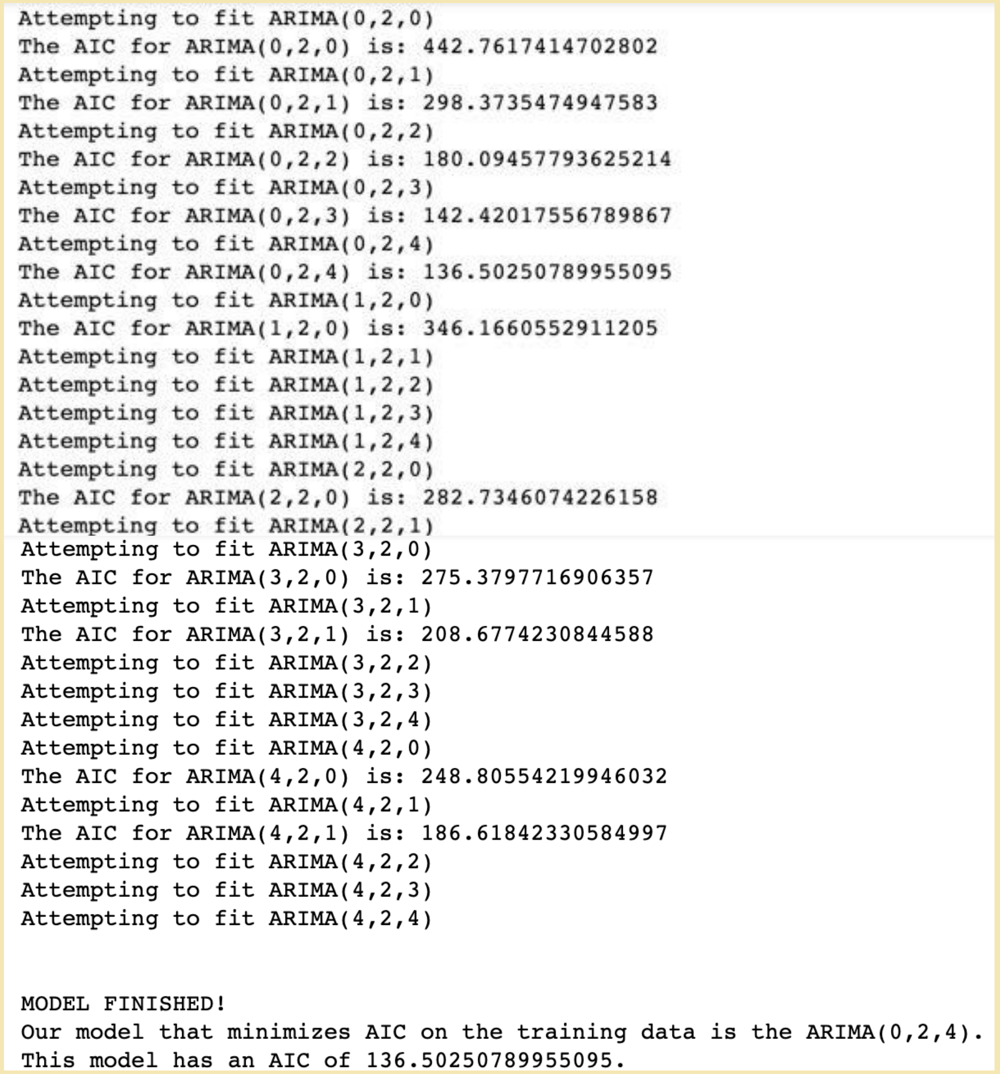




Modeling – for loop

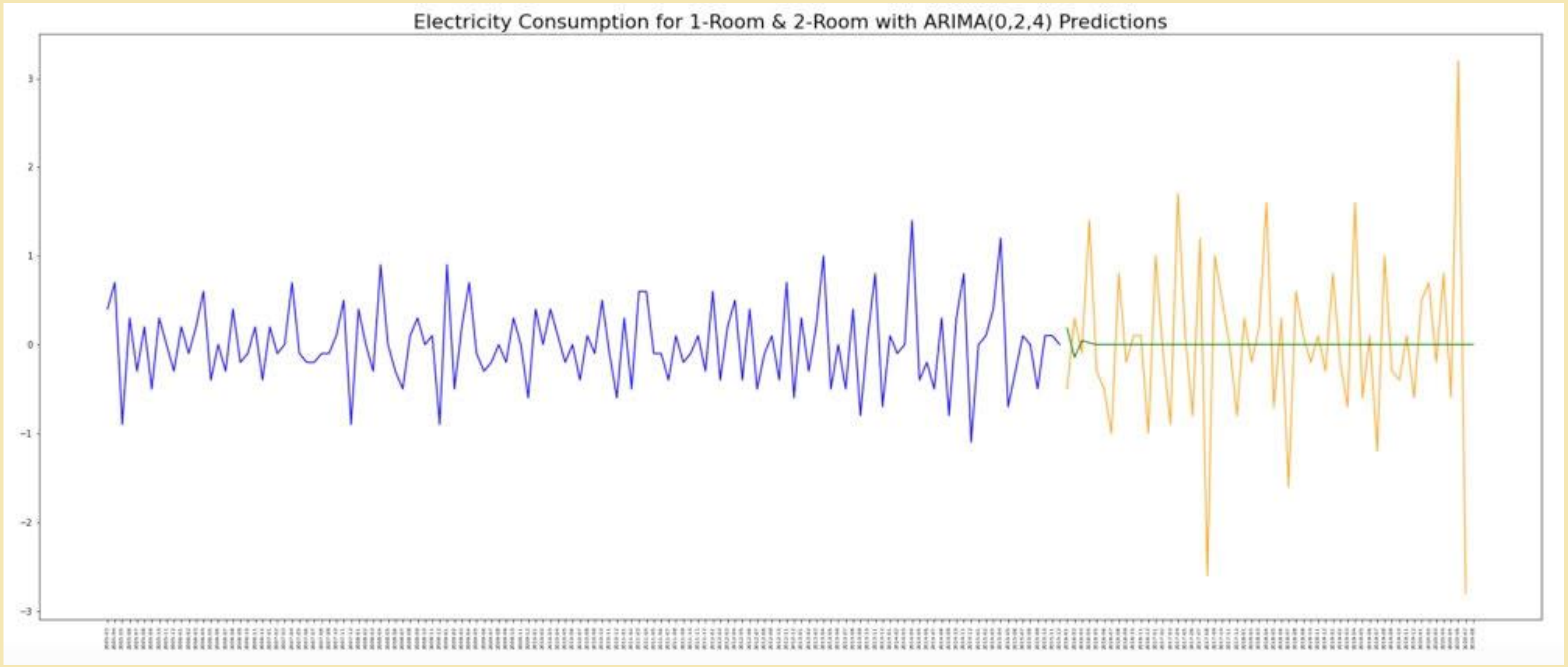


```python
# Starting AIC, p, and q.
best_aic = 99 * (10 ** 16)
best_p = 0
best_q = 0

# Use nested for loop to iterate over values of p and q.
for p in range(5):
    for q in range(5):

        # Insert try and except statements.
        try:

            # Fitting an ARIMA(p, 2, q) model.
            print(f'Attempting to fit ARIMA({p},2,{q})')

            # Instantiate ARIMA model.
            arima = ARIMA(endog = train_1room_2room_2nddiff.iloc[2:],
                          order = (p,2,q)) # values of p, d, q

            # Fit ARIMA model.
            model = arima.fit()

            # Print out AIC for ARIMA(p, 2, q) model.
            print(f'The AIC for ARIMA({p},2,{q}) is: {model.aic}')

            # Is my current model's AIC better than our best_aic?
            if model.aic < best_aic:

                # If so, let's overwrite best_aic, best_p, and best_q.
                best_aic = model.aic
                best_p = p
                best_q = q

        except:
            pass
print()
print()
print('MODEL FINISHED!')
print(f'Our model that minimizes AIC on the training data is the ARIMA({best_p},2,{best_q}).')
print(f'This model has an AIC of {best_aic}.')
```

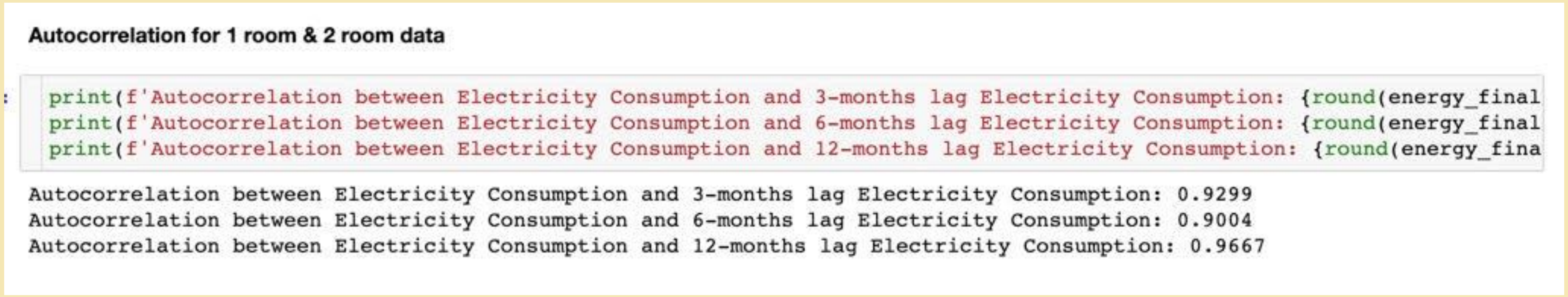


```
Attempting to fit ARIMA(0,2,0)
The AIC for ARIMA(0,2,0) is: 442.7617414702802
Attempting to fit ARIMA(0,2,1)
The AIC for ARIMA(0,2,1) is: 298.3735474947583
Attempting to fit ARIMA(0,2,2)
The AIC for ARIMA(0,2,2) is: 180.09457793625214
Attempting to fit ARIMA(0,2,3)
The AIC for ARIMA(0,2,3) is: 142.42017556789867
Attempting to fit ARIMA(0,2,4)
The AIC for ARIMA(0,2,4) is: 136.50250789955095
Attempting to fit ARIMA(1,2,0)
The AIC for ARIMA(1,2,0) is: 346.1660552911205
Attempting to fit ARIMA(1,2,1)
Attempting to fit ARIMA(1,2,2)
Attempting to fit ARIMA(1,2,3)
Attempting to fit ARIMA(1,2,4)
Attempting to fit ARIMA(2,2,0)
The AIC for ARIMA(2,2,0) is: 282.7346074226158
Attempting to fit ARIMA(2,2,1)
Attempting to fit ARIMA(3,2,0)
The AIC for ARIMA(3,2,0) is: 275.3797716906357
Attempting to fit ARIMA(3,2,1)
The AIC for ARIMA(3,2,1) is: 208.6774230844588
Attempting to fit ARIMA(3,2,2)
Attempting to fit ARIMA(3,2,3)
Attempting to fit ARIMA(3,2,4)
Attempting to fit ARIMA(4,2,0)
The AIC for ARIMA(4,2,0) is: 248.80554219946032
Attempting to fit ARIMA(4,2,1)
The AIC for ARIMA(4,2,1) is: 186.61842330584997
Attempting to fit ARIMA(4,2,2)
Attempting to fit ARIMA(4,2,3)
Attempting to fit ARIMA(4,2,4)


MODEL FINISHED!
Our model that minimizes AIC on the training data is the ARIMA(0,2,4).
This model has an AIC of 136.50250789955095.
```

Modeling – ARIMA (0, 2, 4)



Modeling – Seasonal Factor

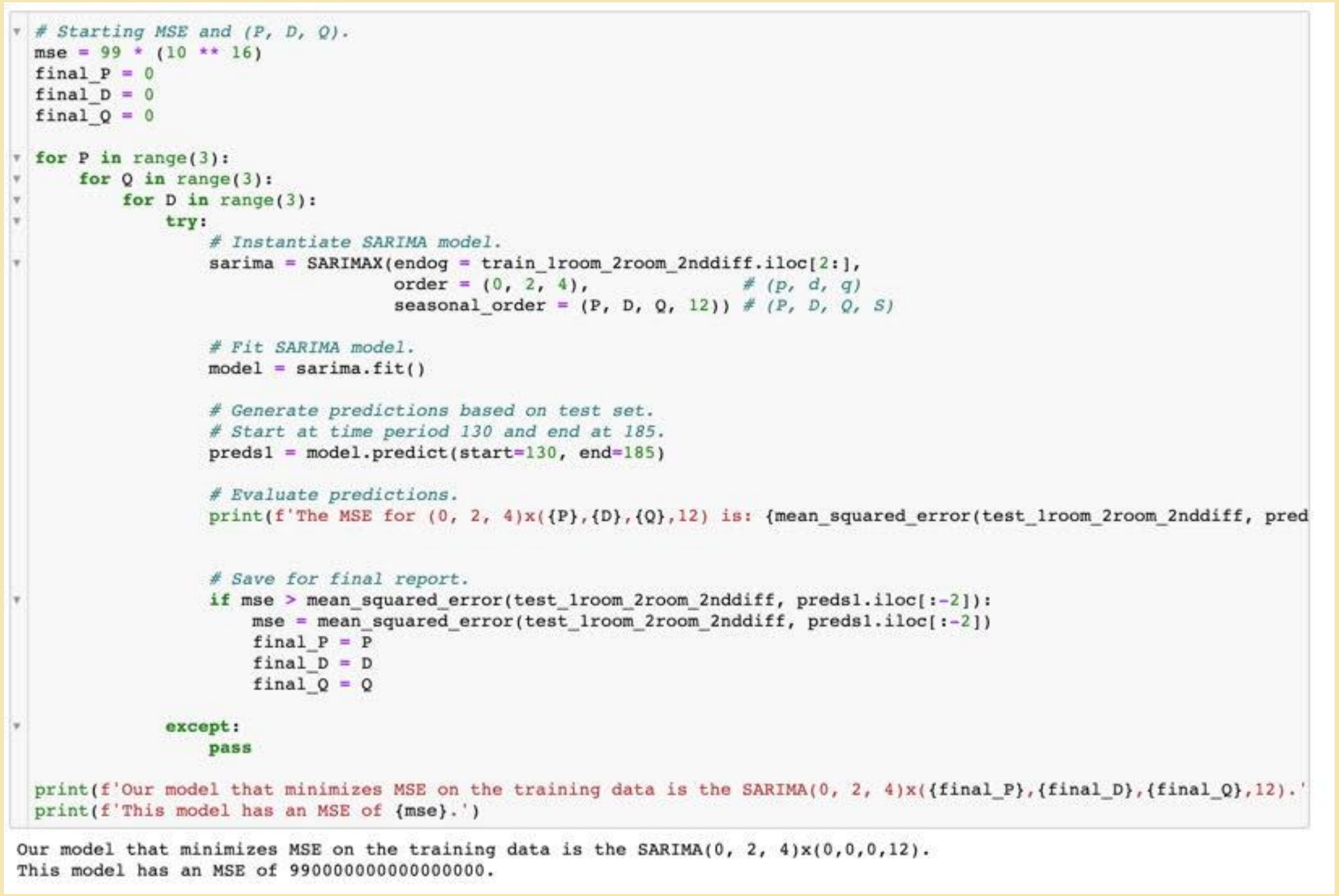


Autocorrelation for 1 room & 2 room data

```
print(f'Autocorrelation between Electricity Consumption and 3-months lag Electricity Consumption: {round(energy_final
print(f'Autocorrelation between Electricity Consumption and 6-months lag Electricity Consumption: {round(energy_final
print(f'Autocorrelation between Electricity Consumption and 12-months lag Electricity Consumption: {round(energy_fina
```

```
Autocorrelation between Electricity Consumption and 3-months lag Electricity Consumption: 0.9299
Autocorrelation between Electricity Consumption and 6-months lag Electricity Consumption: 0.9004
Autocorrelation between Electricity Consumption and 12-months lag Electricity Consumption: 0.9667
```

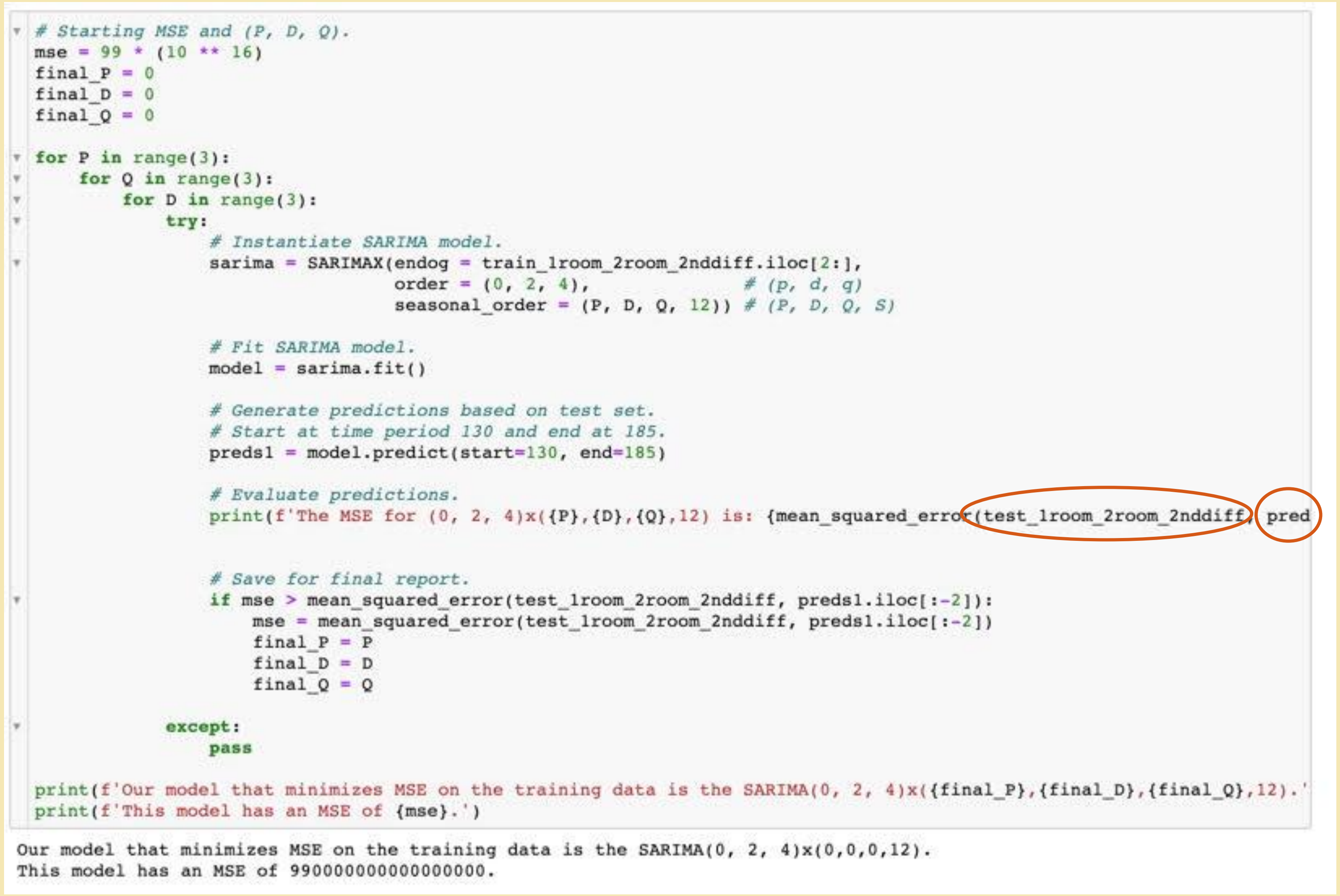
Modeling – for loop



```
# Starting MSE and (P, D, Q).
mse = 99 * (10 ** 16)
final_P = 0
final_D = 0
final_Q = 0

for P in range(3):
    for Q in range(3):
        for D in range(3):
            try:
                # Instantiate SARIMA model.
                sarima = SARIMAX(endog = train_1room_2room_2nddiff.iloc[2:],
                                 order = (0, 2, 4),                # (p, d, q)
                                 seasonal_order = (P, D, Q, 12)) # (P, D, Q, S)

                # Fit SARIMA model.
                model = sarima.fit()

                # Generate predictions based on test set.
                # Start at time period 130 and end at 185.
                preds1 = model.predict(start=130, end=185)

                # Evaluate predictions.
                print(f'The MSE for (0, 2, 4)x({P},{D},{Q},12) is: {mean_squared_error(test_1room_2room_2nddiff, pred


                # Save for final report.
                if mse > mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2]):
                    mse = mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2])
                    final_P = P
                    final_D = D
                    final_Q = Q

            except:
                pass

print(f'Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x({final_P},{final_D},{final_Q},12).'
print(f'This model has an MSE of {mse}.')
```

```
Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x(0,0,0,12).
This model has an MSE of 990000000000000000.
```

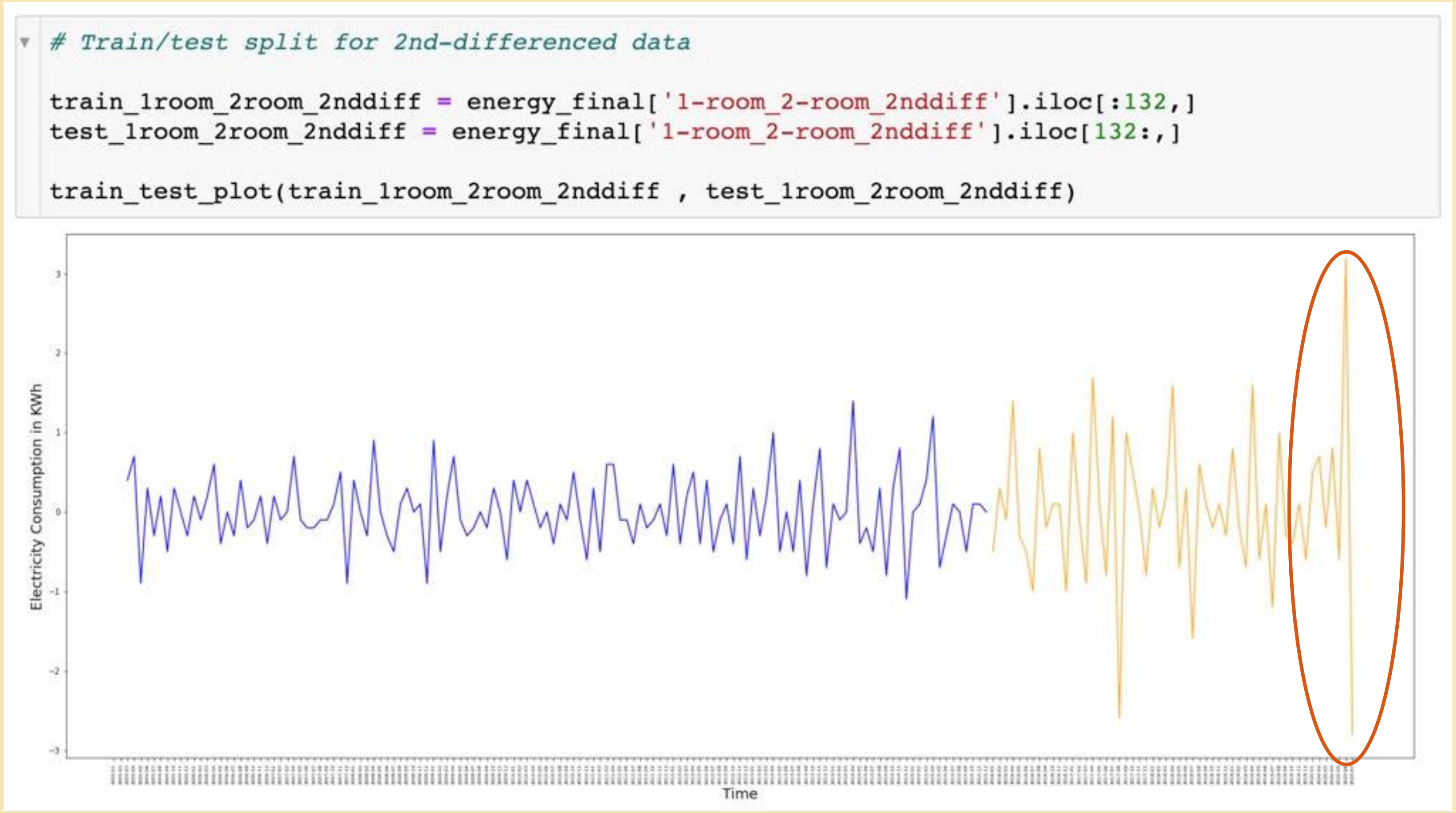
Modeling – for loop



```
# Starting MSE and (P, D, Q).
mse = 99 * (10 ** 16)
final_P = 0
final_D = 0
final_Q = 0

for P in range(3):
    for Q in range(3):
        for D in range(3):
            try:
                # Instantiate SARIMA model.
                sarima = SARIMAX(endog = train_1room_2room_2nddiff.iloc[2:],
                                 order = (0, 2, 4),               # (p, d, q)
                                 seasonal_order = (P, D, Q, 12)) # (P, D, Q, S)

                # Fit SARIMA model.
                model = sarima.fit()

                # Generate predictions based on test set.
                # Start at time period 130 and end at 185.
                preds1 = model.predict(start=130, end=185)

                # Evaluate predictions.
                print(f'The MSE for (0, 2, 4)x({P},{D},{Q},12) is: {mean_squared_error(test_1room_2room_2nddiff, pred


                # Save for final report.
                if mse > mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2]):
                    mse = mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2])
                    final_P = P
                    final_D = D
                    final_Q = Q

            except:
                pass

print(f'Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x({final_P},{final_D},{final_Q},12).'
print(f'This model has an MSE of {mse}.')
```

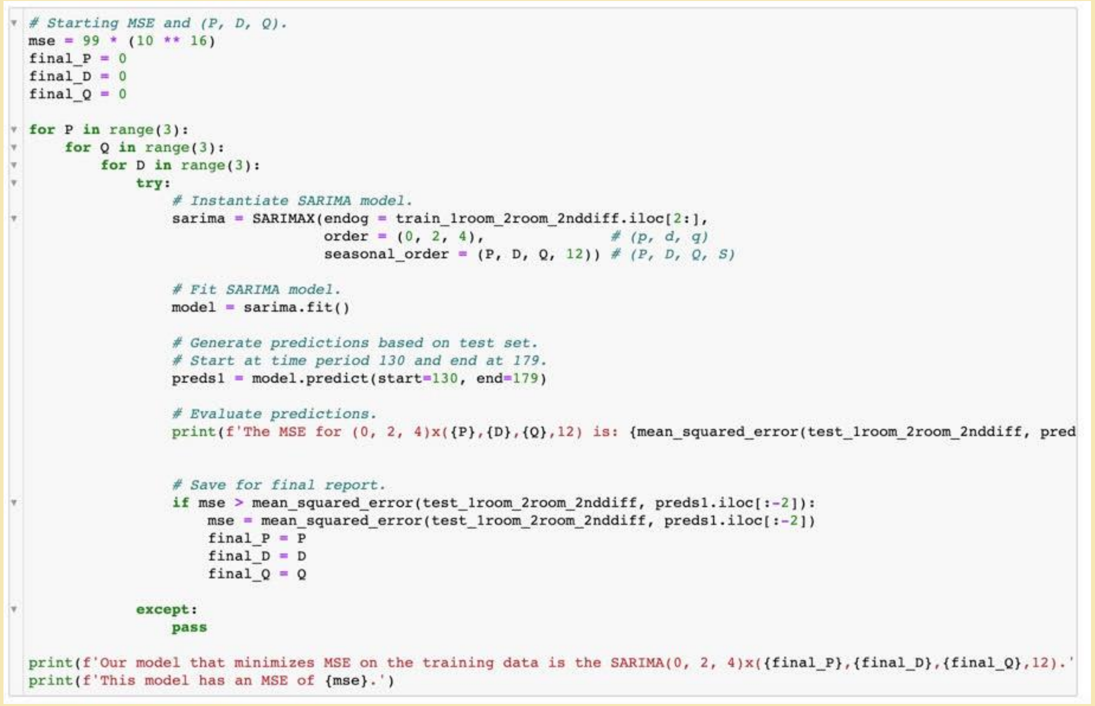
```
Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x(0,0,0,12).
This model has an MSE of 990000000000000000.
```

Modeling – Seasonal Factor

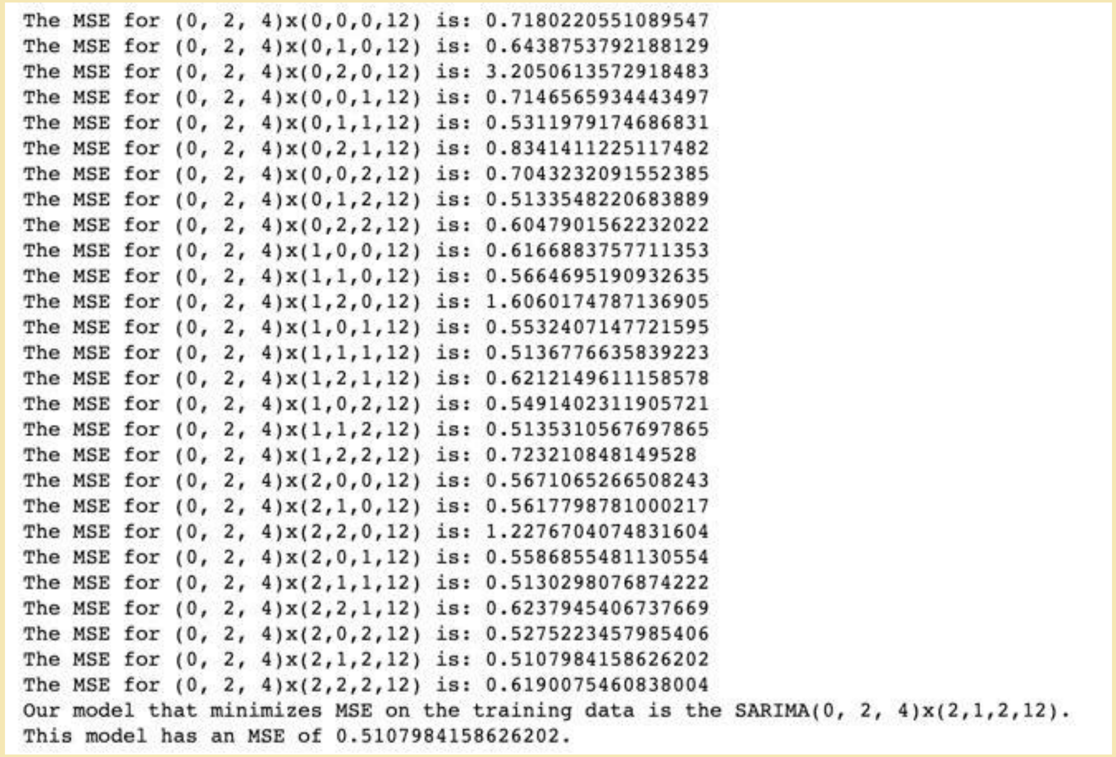


```
# Train/test split for 2nd-differenced data

train_1room_2room_2nddiff = energy_final['1-room_2-room_2nddiff'].iloc[:132,]
test_1room_2room_2nddiff = energy_final['1-room_2-room_2nddiff'].iloc[132:,]

train_test_plot(train_1room_2room_2nddiff , test_1room_2room_2nddiff)
```

Modeling – for loop

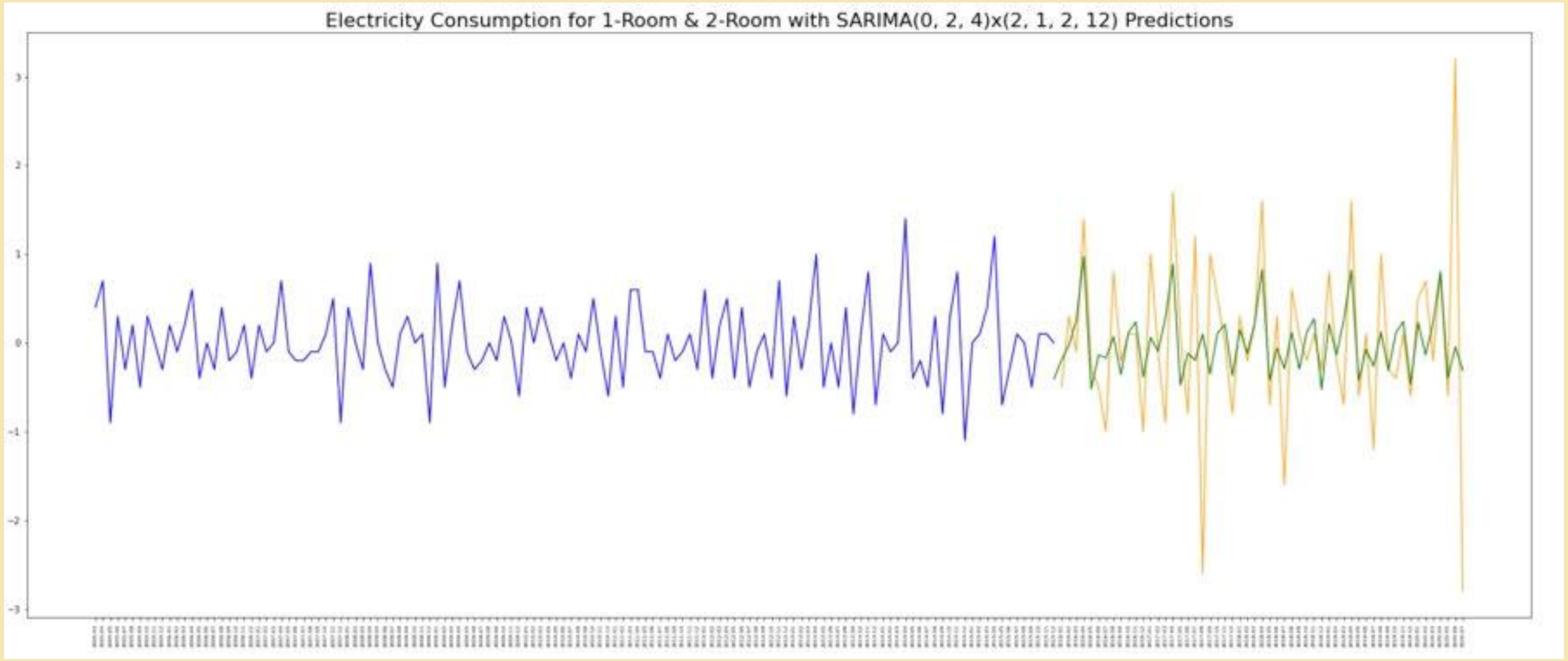


```python
# Starting MSE and (P, D, Q).
mse = 99 * (10 ** 16)
final_P = 0
final_D = 0
final_Q = 0

for P in range(3):
    for Q in range(3):
        for D in range(3):
            try:
                # Instantiate SARIMA model.
                sarima = SARIMAX(endog = train_1room_2room_2nddiff.iloc[2:],
                                 order = (0, 2, 4),              # (p, d, q)
                                 seasonal_order = (P, D, Q, 12)) # (P, D, Q, S)

                # Fit SARIMA model.
                model = sarima.fit()

                # Generate predictions based on test set.
                # Start at time period 130 and end at 179.
                preds1 = model.predict(start=130, end=179)

                # Evaluate predictions.
                print(f'The MSE for (0, 2, 4)x({P},{D},{Q},12) is: {mean_squared_error(test_1room_2room_2nddiff, pred


                # Save for final report.
                if mse > mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2]):
                    mse = mean_squared_error(test_1room_2room_2nddiff, preds1.iloc[:-2])
                    final_P = P
                    final_D = D
                    final_Q = Q

            except:
                pass

print(f'Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x({final_P},{final_D},{final_Q},12).'
print(f'This model has an MSE of {mse}.')
```

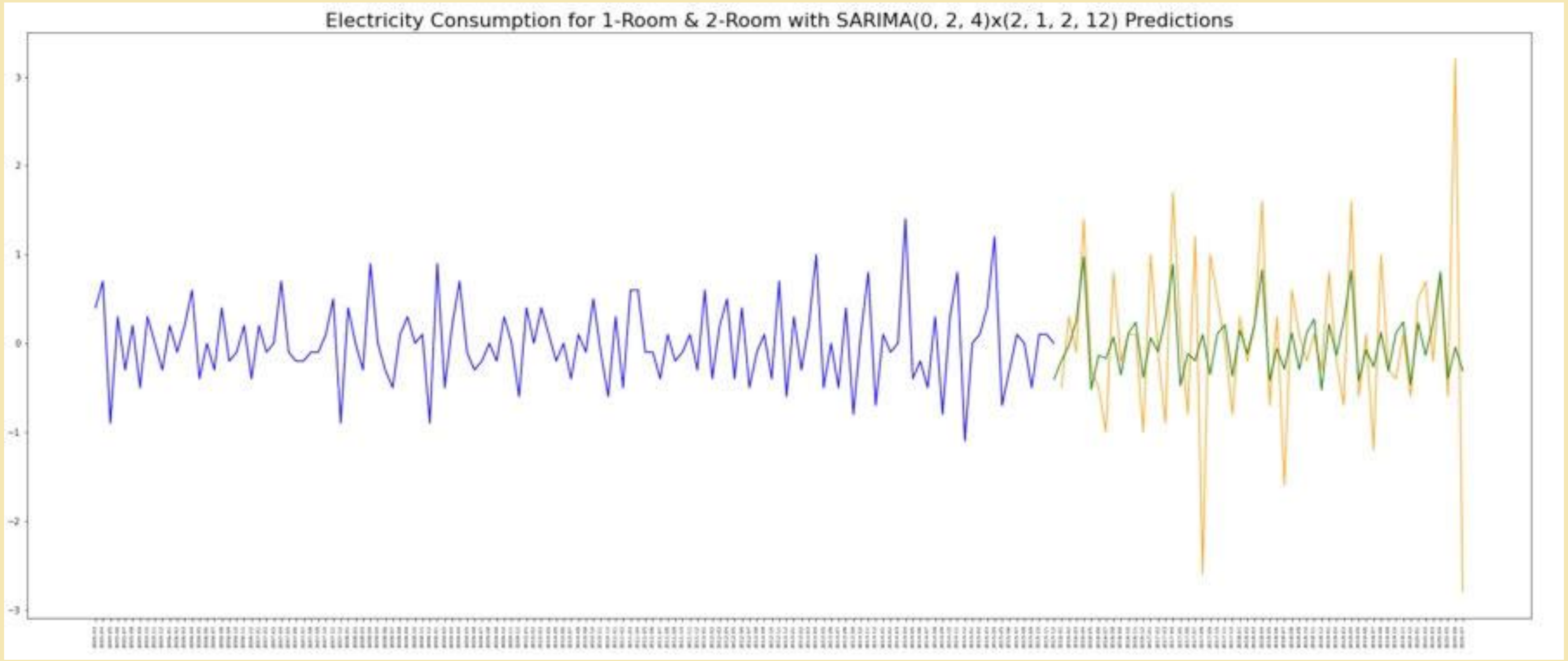


```
The MSE for (0, 2, 4)x(0,0,0,12) is: 0.7180220551089547
The MSE for (0, 2, 4)x(0,1,0,12) is: 0.6438753792188129
The MSE for (0, 2, 4)x(0,2,0,12) is: 3.2050613572918483
The MSE for (0, 2, 4)x(0,0,1,12) is: 0.7146565934443497
The MSE for (0, 2, 4)x(0,1,1,12) is: 0.5311979174686831
The MSE for (0, 2, 4)x(0,2,1,12) is: 0.8341411225117482
The MSE for (0, 2, 4)x(0,0,2,12) is: 0.7043232091552385
The MSE for (0, 2, 4)x(0,1,2,12) is: 0.5133548220683889
The MSE for (0, 2, 4)x(0,2,2,12) is: 0.6047901562232022
The MSE for (0, 2, 4)x(1,0,0,12) is: 0.6166883757711353
The MSE for (0, 2, 4)x(1,1,0,12) is: 0.5664695190932635
The MSE for (0, 2, 4)x(1,2,0,12) is: 1.6060174787136905
The MSE for (0, 2, 4)x(1,0,1,12) is: 0.5532407147721595
The MSE for (0, 2, 4)x(1,1,1,12) is: 0.5136776635839223
The MSE for (0, 2, 4)x(1,2,1,12) is: 0.6212149611158578
The MSE for (0, 2, 4)x(1,0,2,12) is: 0.5491402311905721
The MSE for (0, 2, 4)x(1,1,2,12) is: 0.5135310567697865
The MSE for (0, 2, 4)x(1,2,2,12) is: 0.723210848149528
The MSE for (0, 2, 4)x(2,0,0,12) is: 0.5671065266508243
The MSE for (0, 2, 4)x(2,1,0,12) is: 0.5617798781000217
The MSE for (0, 2, 4)x(2,2,0,12) is: 1.2276704074831604
The MSE for (0, 2, 4)x(2,0,1,12) is: 0.5586855481130554
The MSE for (0, 2, 4)x(2,1,1,12) is: 0.5130298076874222
The MSE for (0, 2, 4)x(2,2,1,12) is: 0.6237945406737669
The MSE for (0, 2, 4)x(2,0,2,12) is: 0.5275223457985406
The MSE for (0, 2, 4)x(2,1,2,12) is: 0.5107984158626202
The MSE for (0, 2, 4)x(2,2,2,12) is: 0.6190075460838004
Our model that minimizes MSE on the training data is the SARIMA(0, 2, 4)x(2,1,2,12).
This model has an MSE of 0.5107984158626202.
```

Modeling – SARIMA (0, 2, 4) x (2, 1, 2, 12)



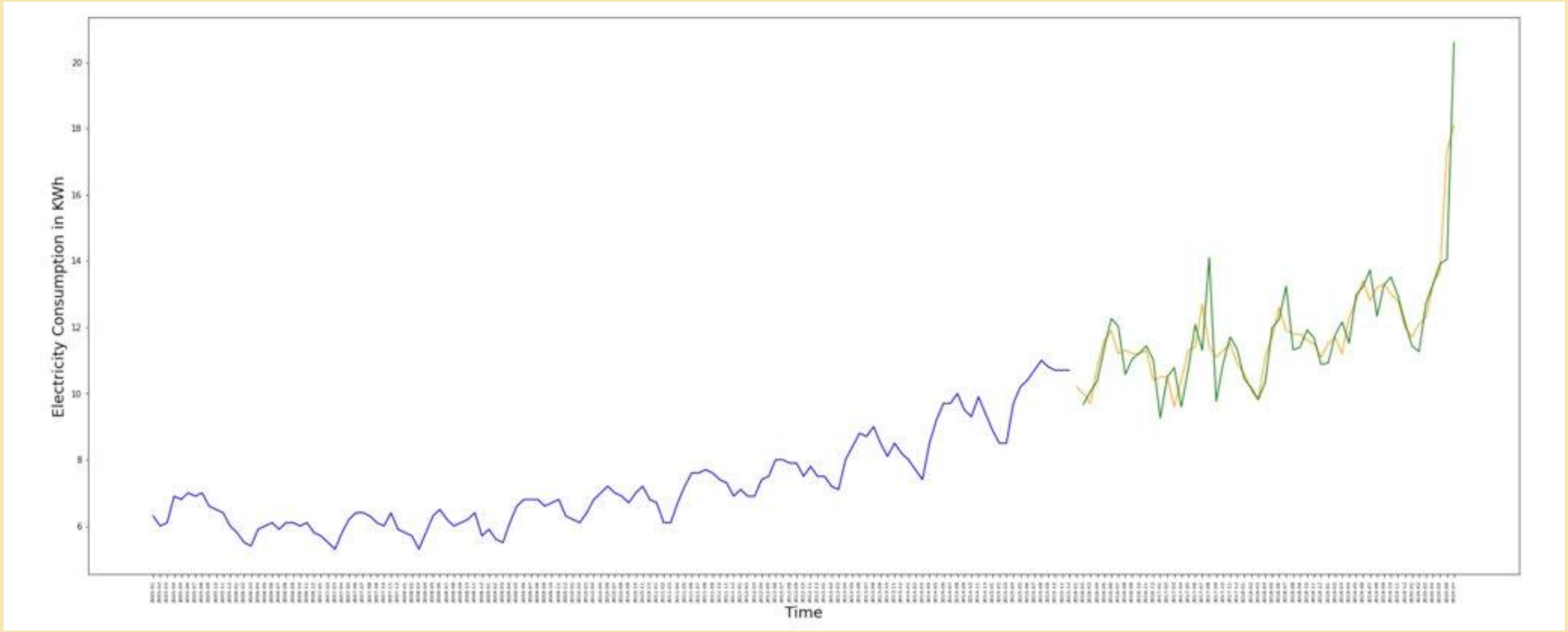
Modeling – Reverse Differencing



Reverse Differencing

Original	Y_t
1-time Difference	$Y_t' = Y_t - Y_{t-1}$
2-times Differences	$Y''_t = Y'_t - Y'_{t-1}$
	$Y''_t = Y_t - 2Y_{t-1} + Y_{t-2}$
	$Y_t \quad = Y''_t + 2Y_{t-1} - Y_{t-2}$

Reverse Differencing



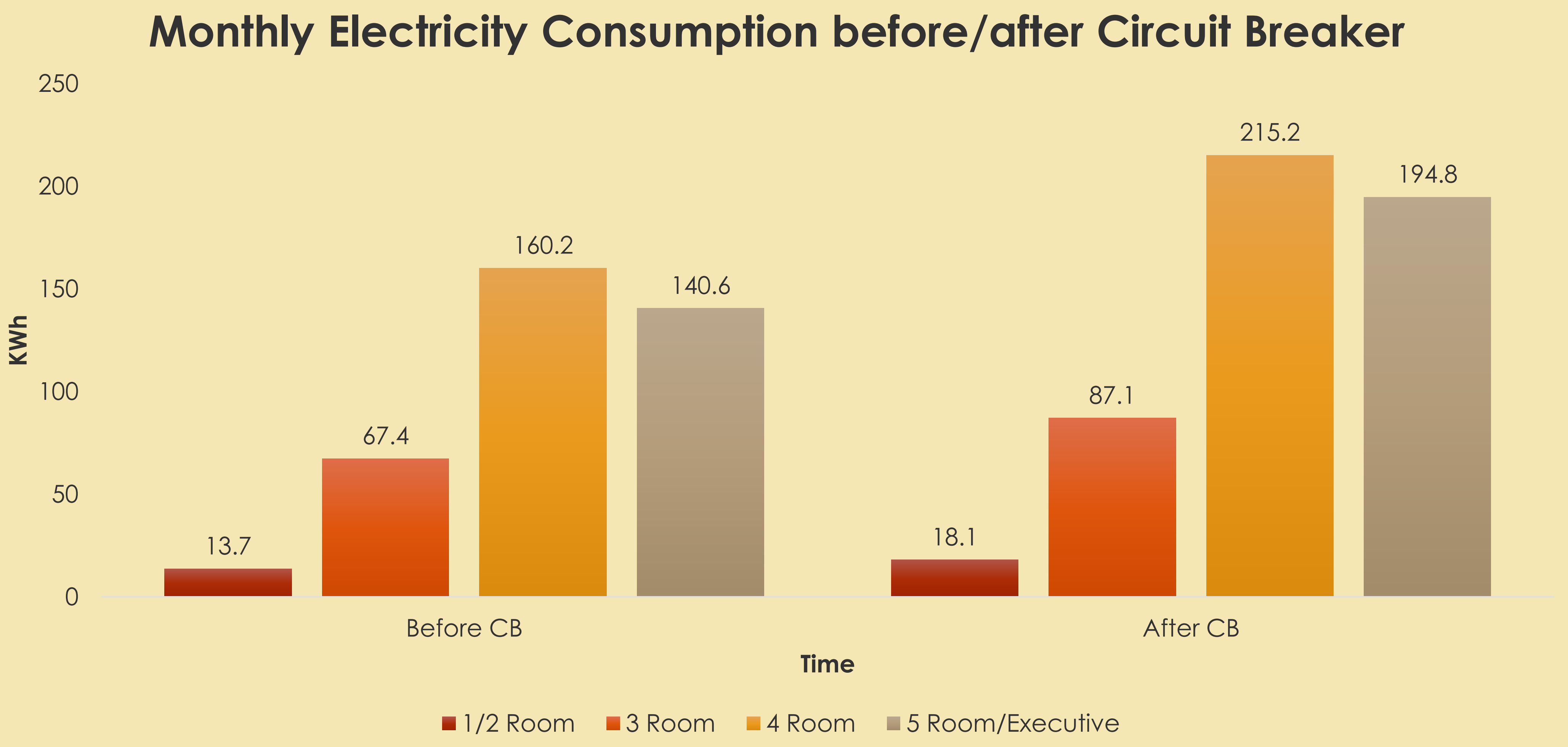
Conclusion

As Singapore enters phase 3 of circuit-breaker, a lot of people still Work from Home and Home-Based Learning.

We foresee that the rate of household electricity consumption of 2021 is higher than usual, and it is quite difficult to predict the actual rate because the usage pattern is different from past years.

Thank you & let's save electricity

Appendix



Monthly Electricity Consumption before/after Circuit Breaker
250
200
150
100
50
0
KWh
13.7
67.4
160.2
140.6
18.1
87.1
215.2
194.8
Before CB
After CB
Time
1/2 Room
3 Room
4 Room
5 Room/Executive

Appendix

Incandescent Light
73 KWh / month

LED Light
11.7 KWh / month