Time Series

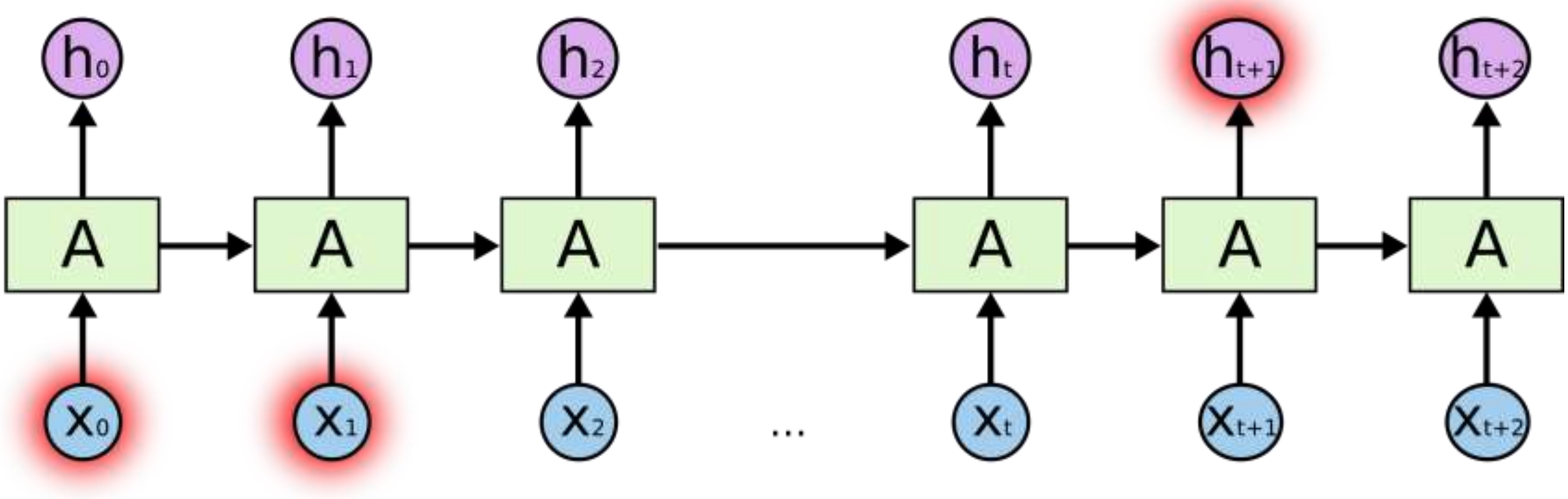
권지혜 윤빈나 윤재경
전혜민 정재원 최성웅

KU-BIG

Index

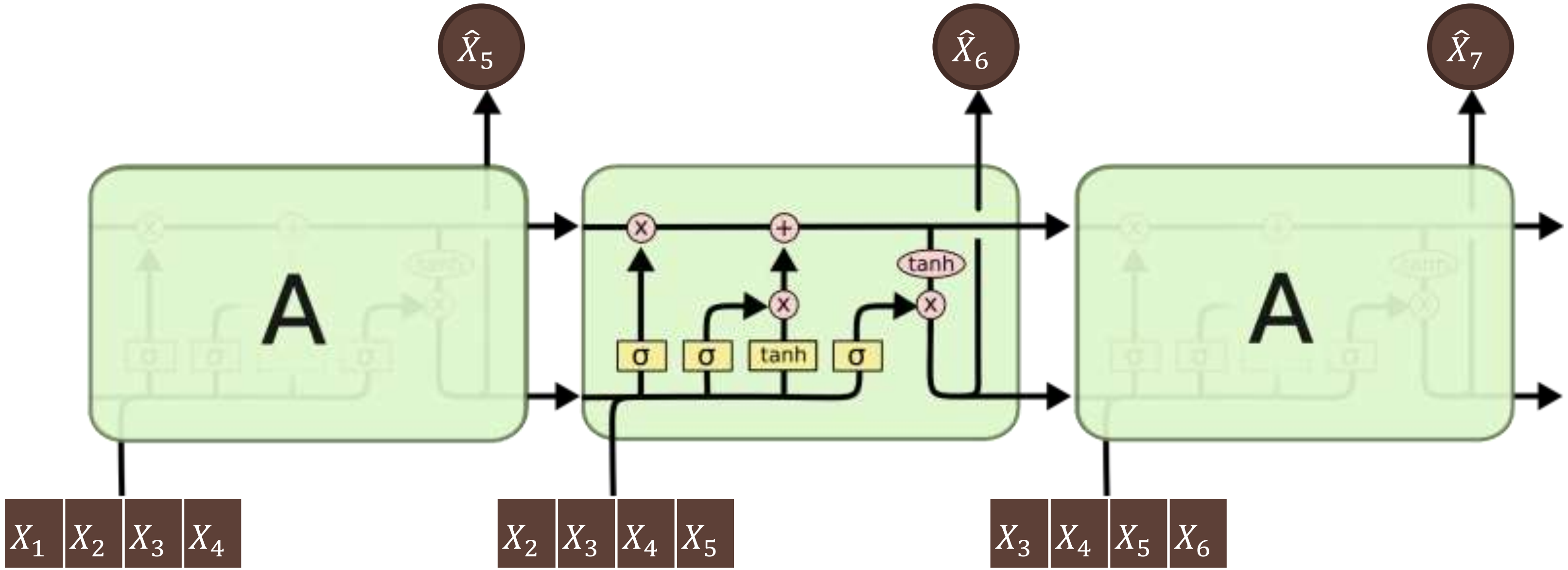
1. LSTM
2. Box-Jenkins Method

LSTM



RNN : Gradient Vanishing / Gradient Exploding problem

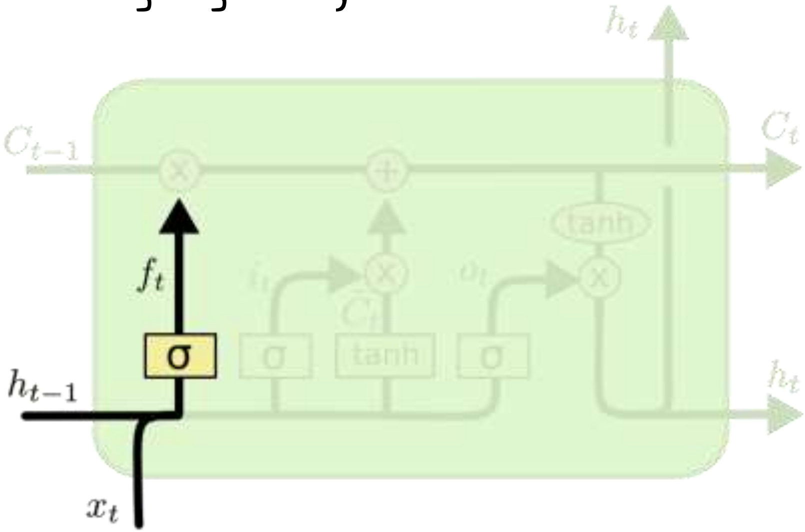
LSTM



$\hat{X}_5$
$\hat{X}_6$
$\hat{X}_7$
A
A
σ
σ
tanh
σ
tanh
X_1 X_2 X_3 X_4
X_2 X_3 X_4 X_5
X_3 X_4 X_5 X_6

LSTM

- Forget gate layer

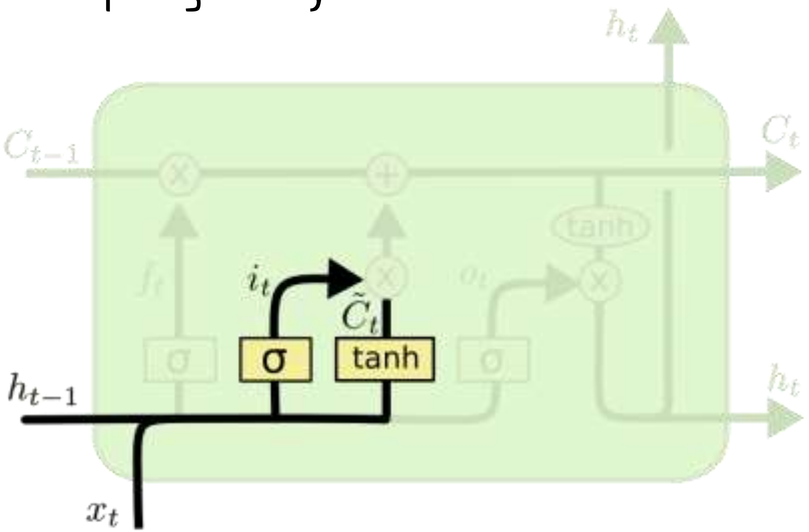


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

cell state로부터 어떤 정보를 버릴 것인지 정하는 단계의 gate

LSTM

- Input gate layer

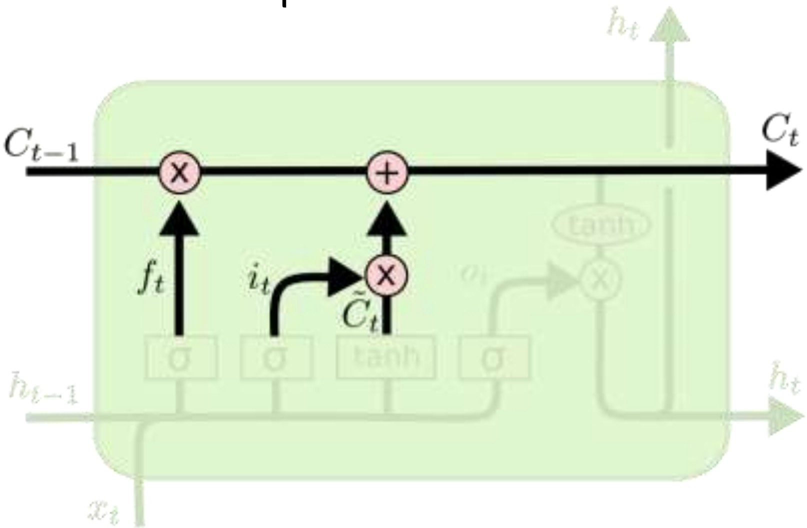


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

새로운 정보 중 어떤 것을 cell state에 저장할 것인지 결정

LSTM

- Cell state update

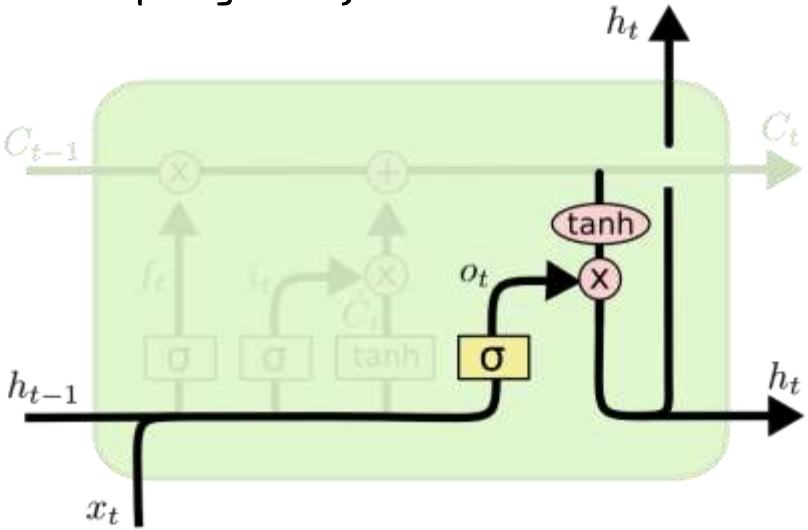


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

과거 state를 업데이트해서 새로운 cell state를 만듦

LSTM

- Output gate layer

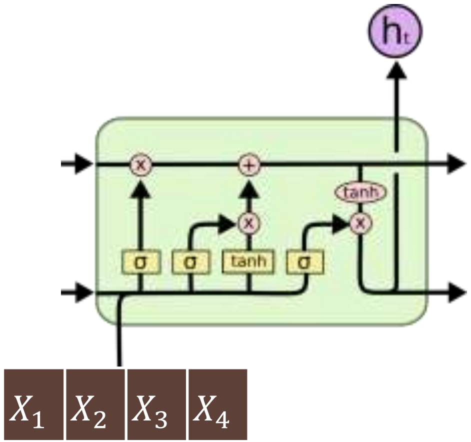


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

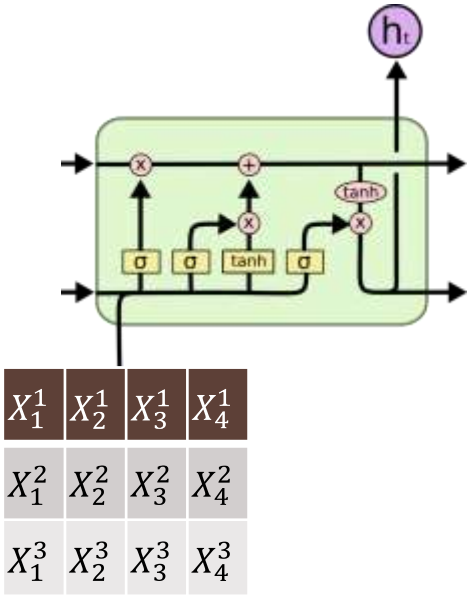
input 데이터를 태워서 cell state의 어느 부분을 output으로 내보낼지 결정함

LSTM

- Univariate



- Multivariate



LSTM - Code

timestamp	meter_reading	air_temperature	dew_temperature	precip_depth_1_hr	sea_level_pressure	wind_direction	wind_speed
2016-01-01 00:00:00	28.10	15.6	-5.6	0.0	1016.3	270.0	3.6
2016-01-01 01:00:00	26.57	13.9	-5.6	0.0	1015.6	270.0	4.1
2016-01-01 02:00:00	25.73	13.3	-5.6	0.0	1016.0	270.0	3.1
2016-01-01 03:00:00	25.96	12.2	-6.1	0.0	1016.6	280.0	3.1
2016-01-01 04:00:00	25.59	11.7	-6.7	0.0	1017.0	270.0	3.1

- Scale/Set window size

```python
# load dataset
values = dataset.values
# ensure all data is float
values = values.astype('float32')
# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

window_size=24

# frame as supervised learning
reframed = series_to_supervised(scaled, window_size, 1)

# drop columns we don't want to predict
reframed.drop(reframed.columns[[169,170,171,172,173,174]], axis=1, inplace=True)
print(reframed.head())
```

```
    var1(t-24)  var2(t-24)  var3(t-24)  var4(t-24)  var5(t-24)  var6(t-24)  \
24    0.090175    0.297778    0.286420    0.007519    0.565160    0.750000
25    0.082777    0.280000    0.286420    0.007519    0.565836    0.750000
26    0.078716    0.246667    0.286420    0.007519    0.580071    0.750000
27    0.079828    0.222222    0.274074    0.007519    0.601424    0.777778
28    0.078039    0.211111    0.259259    0.007519    0.615658    0.750000

    var7(t-24)  var1(t-23)  var2(t-23)  var3(t-23)    ...    var6(t-2)  \
24    0.233766    0.082777    0.280000    0.286420    ...     0.000000
25    0.266234    0.078716    0.246667    0.286420    ...     0.000000
26    0.201299    0.079828    0.222222    0.274074    ...     0.750000
27    0.201299    0.078039    0.211111    0.259259    ...     0.694444
28    0.201299    0.081520    0.173333    0.259259    ...     0.694444

    var7(t-2)  var1(t-1)  var2(t-1)  var3(t-1)  var4(t-1)  var5(t-1)  \
24   0.000000   0.094478   0.371111   0.409877   0.007519   0.629893
25   0.000000   0.087999   0.357778   0.409877   0.007519   0.640569
26   0.168831   0.081955   0.333333   0.424691   0.007519   0.654804
27   0.136364   0.079248   0.271111   0.439506   0.007519   0.683274
28   0.136364   0.079296   0.222222   0.439506   0.007519   0.693950

    var6(t-1)  var7(t-1)   var1(t)
24   0.000000   0.000000  0.087999
25   0.750000   0.168831  0.081955
26   0.694444   0.136364  0.079248
27   0.694444   0.136364  0.079296
28   0.583333   0.097403  0.079344

[5 rows x 169 columns]
```

LSTM - Code

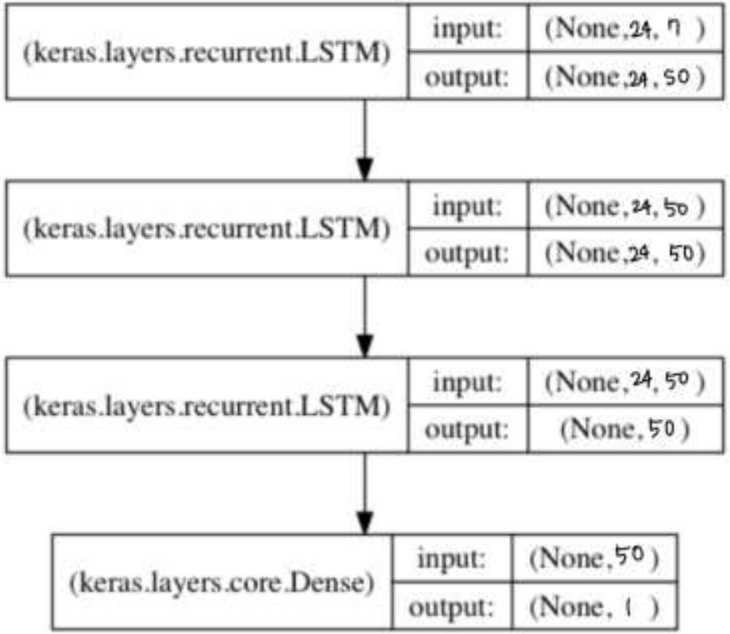
- Split data

```
#split train and test
values = reframed.values
n_train_hours = 200*24
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]
# split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]

# reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], window_size, int(train_X.shape[1]/window_size)))
test_X = test_X.reshape((test_X.shape[0], window_size, int(test_X.shape[1]/window_size)))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

```
(4800, 24, 7) (4800,) (3935, 24, 7) (3935,)
```

- Design LSTM model

```
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import LSTM
from tensorflow.keras import Sequential

# design network - 3 layered lstm
model  = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2]),return_sequences=True))
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2]),return_sequences=True))
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
```



LSTM - Code

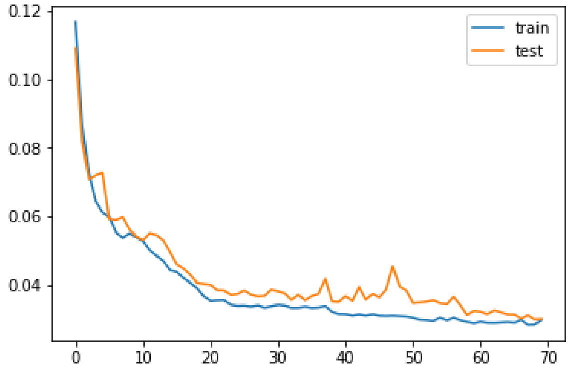
- Fit model

```
history = model.fit(train_X, train_y, epochs=70, batch_size=72,
                    validation_data=(test_X, test_y), verbose=2, shuffle=False)
```

```
Epoch 1/70
 - 16s - loss: 0.1167 - val_loss: 0.1090
Epoch 2/70
 - 9s - loss: 0.0867 - val_loss: 0.0819
Epoch 3/70
 - 9s - loss: 0.0725 - val_loss: 0.0707

. . .

Epoch 68/70
 - 9s - loss: 0.0284 - val_loss: 0.0312
Epoch 69/70
 - 9s - loss: 0.0285 - val_loss: 0.0300
Epoch 70/70
 - 9s - loss: 0.0298 - val_loss: 0.0301
```

- Loss plot

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```

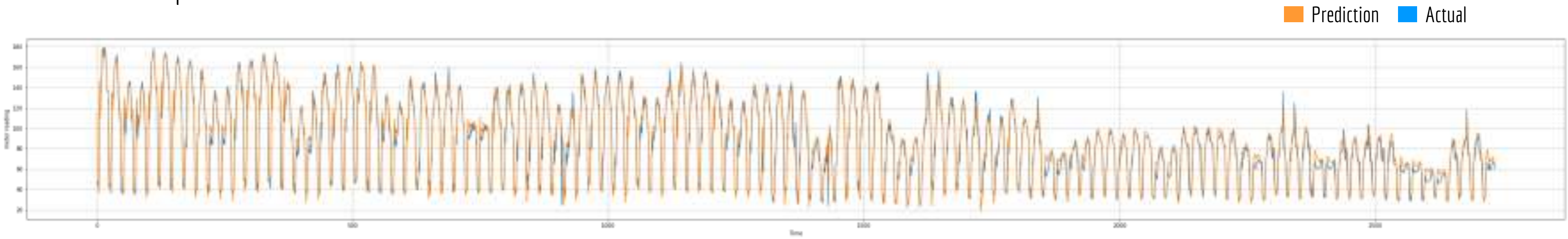


LSTM - Code

- RMSE

```
Test RMSE: 8.678
```

- Prediction plot



Box-Jenkins Method

Classic Method

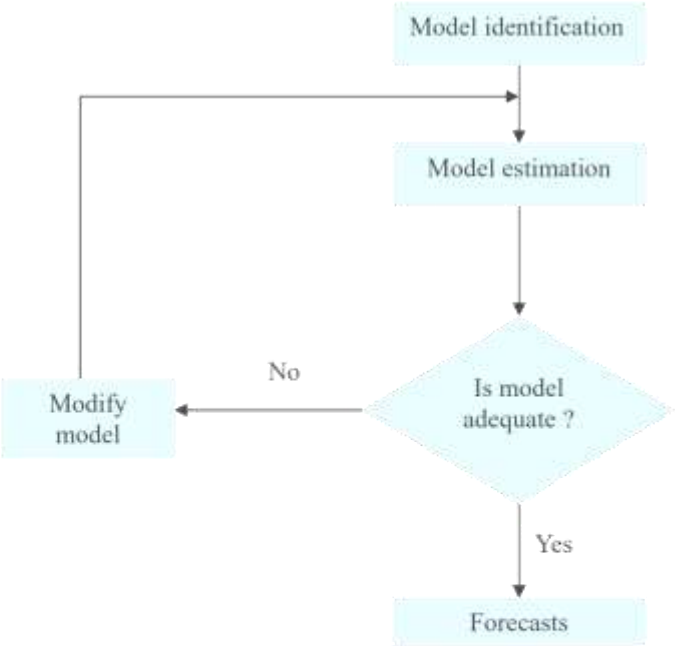
– 모형을 찾는 체계적인 방법이 없음

– Trial-and-error 로 모형을 찾아야 함

– 선택된 방법의 좁은 scope 내에서 찾게 됨

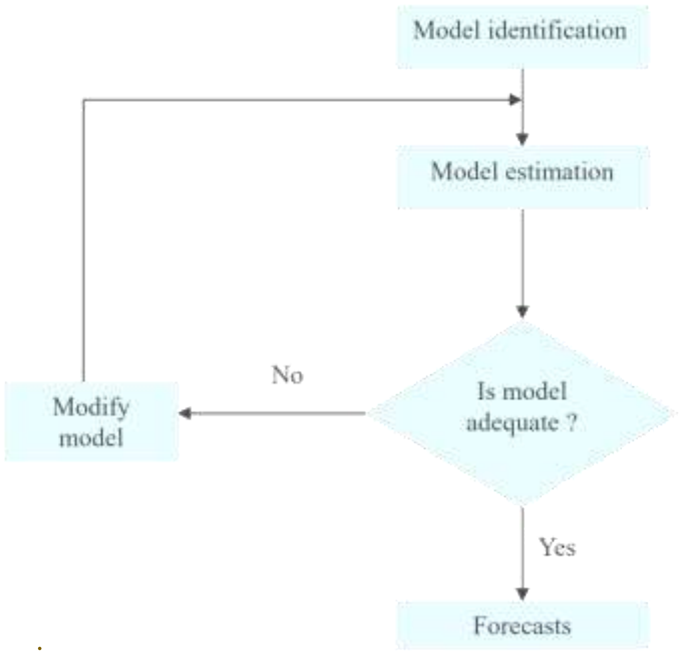
– 잘 작동하는 좋은 모형인지에 대한 (이론적) 검증이 어려움

Box-Jenkins Method



Box-Jenkins Method

Box-Jenkins Method



1. Model Identification
2. Model Estimation
3. Model Diagnostic
4. Modify or Forecast

1. Model Identification

- Is the Time Series Stationary?
- What Differencing will make it stationary?
- What Transforms will make it stationary?
- What Values of p and q are most promising?

Tools

- Time Series Plot
- Seasonal Decomposition
- Augmented Dicky-Fuller Test
- (Partial) Auto Correlation Function

1. Model Identification

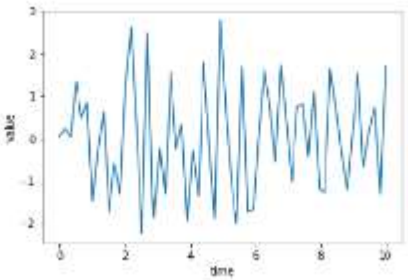
- Is the Time Series Stationary?
- What Differencing will make it stationary?
- What Transforms will make it stationary?
- What Values of p and q are most promising?

Tools

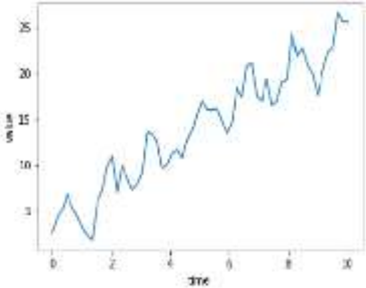
- Time Series Plot
- Seasonal Decomposition
- Augmented Dicky-Fuller Test
- (Partial) Auto Correlation Function

1. Model Identification Tools – Time Series Plot

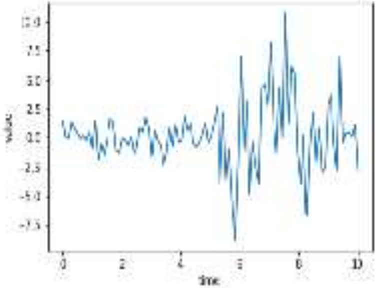
- Stationarity



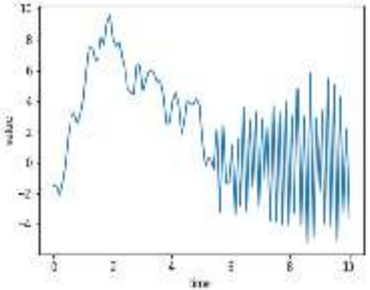
VS



Zero-Trend ?

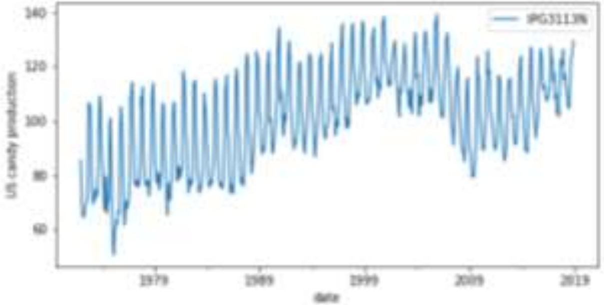


Constant Variance ?



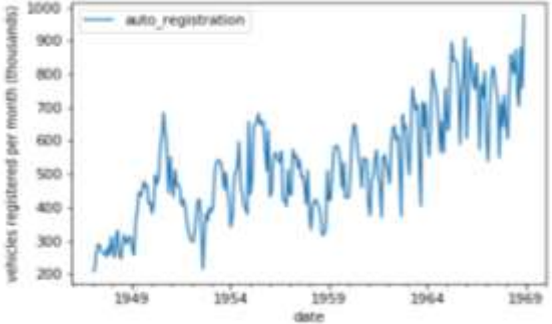
Constant AutoCorrelation ?

- Seasonality



Strong seasonal pattern

Use seasonal differencing

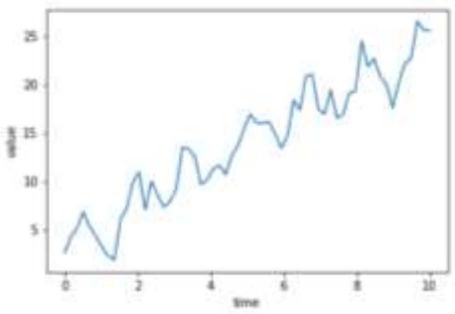


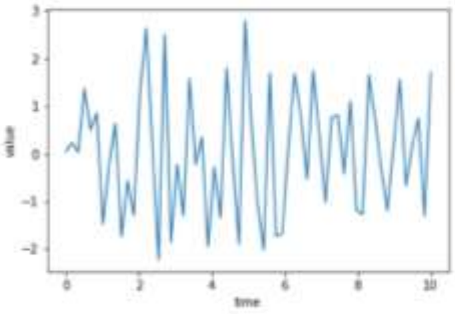
Weak seasonal pattern

Optional

1. Model Identification Tools – Time Series Plot

- Differencing





- (Variance Stabilizing) Transformation

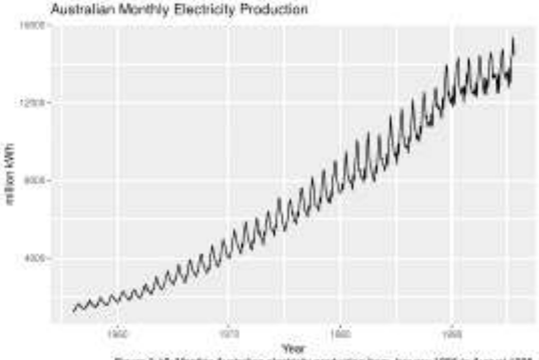
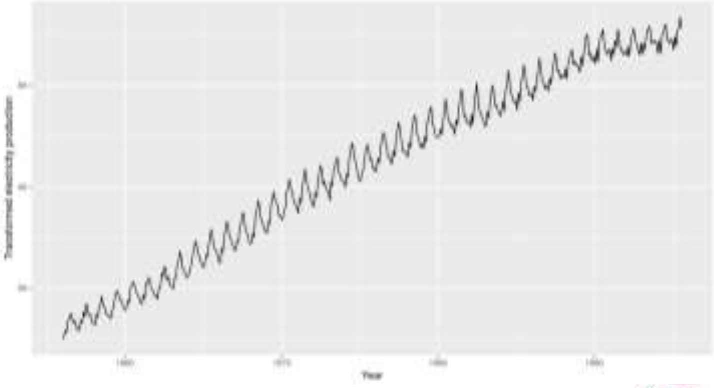


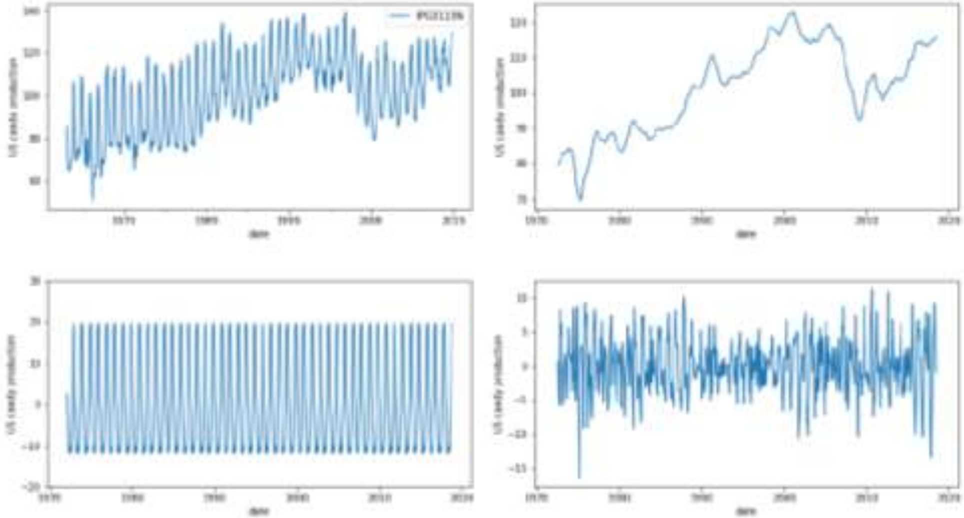
Figure 2-10: Monthly Australian electricity production from January 1956 to August 1995. Note the increasing variation as the level of the series increases.

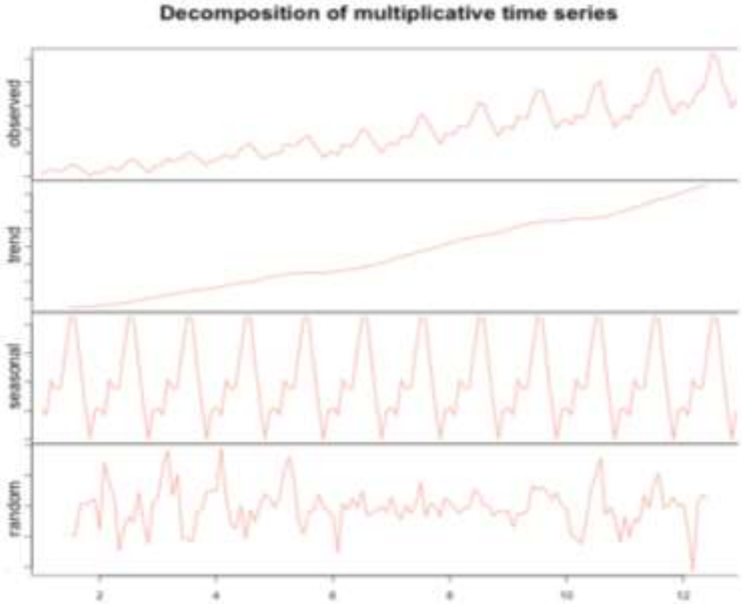
Box-Cox Transformation

$$w_t = \begin{cases} \log(y_t) & \text{if } \lambda = 0; \\ (y_t^\lambda - 1)/\lambda & \text{otherwise.} \end{cases}$$



1. Model Identification Tools – Seasonal Decomposition





Additive series = Trend + Season

Using Differencing

Multiplicative series = Trend x Season

Transformation ex> Log

1. Model Identification Tools- ADF Test

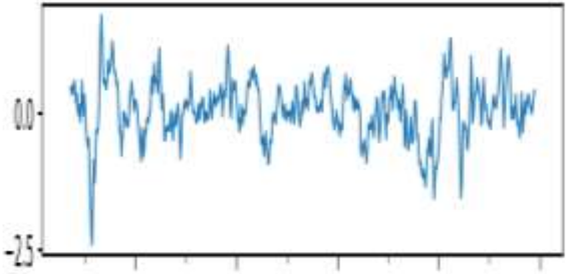
- Test for TREND Non-Stationary

- Null hypothesis is time series is non-stationary

- Test Statistic : More negative, more likely to be stationary

Non-stationary!

=> Differencing?

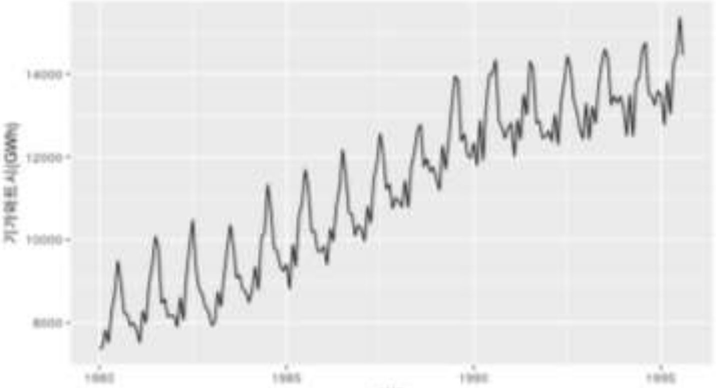


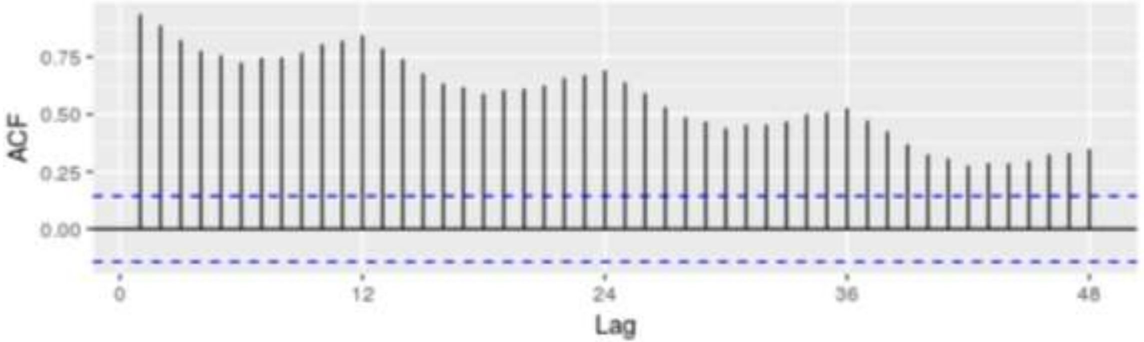

```
In [14]: from statsmodels.tsa.stattools import adfuller
         results = adfuller(candy['IPG3113N'])
         print('p-value is :', results[1],
          'test statistic is :', results[0])
p-value is : 0.3924327500713993 test statistic is : -1.77601530875016125
```

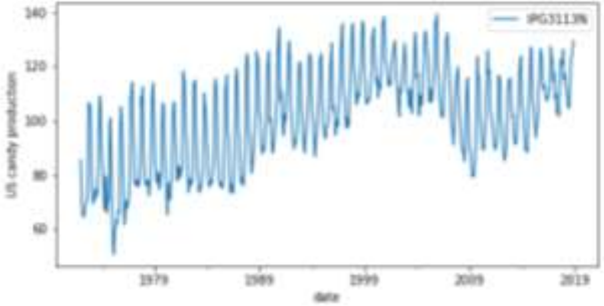
```
In [60]: candy_diff = candy.diff().dropna()
         results = adfuller(candy_diff['IPG3113N'])
         print('p-value is :', results[1],
          'test statistic is :', results[0])
p-value is : 6.631549159334366e-08 test statistic is : -6.175912489755692
```

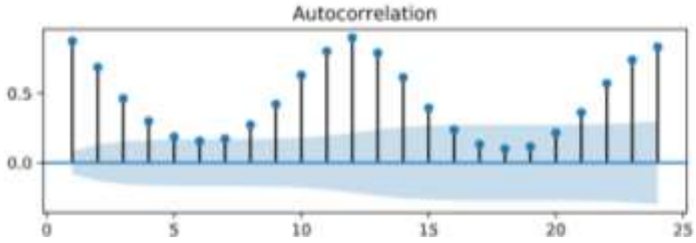
1. Model Identification Tools- ACF/PACF

- Finding Trend & Seasonality



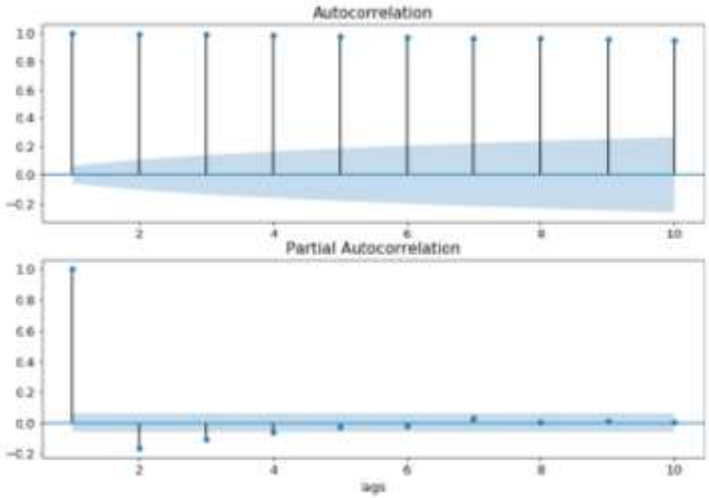


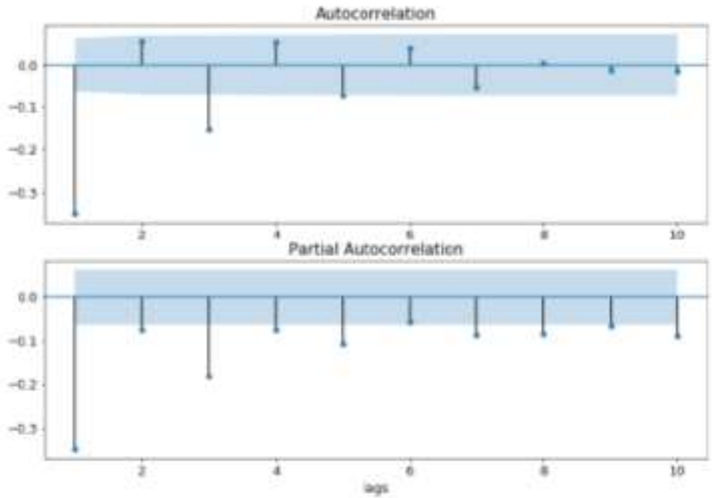




1. Model Identification Tools- ACF/PACF

- Under/Over Differencing





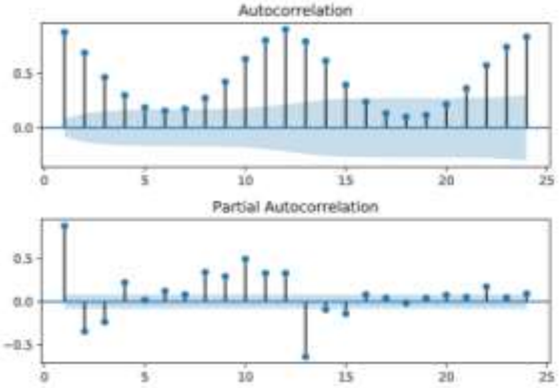
1. Model Identification Tools- ACF/PACF

- Finding Orders (p,d,q) (P,D,Q,)S

	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

1. Model Identification Tools- ACF/PACF

- Finding Orders (p,d,q) (P,D,Q,),S



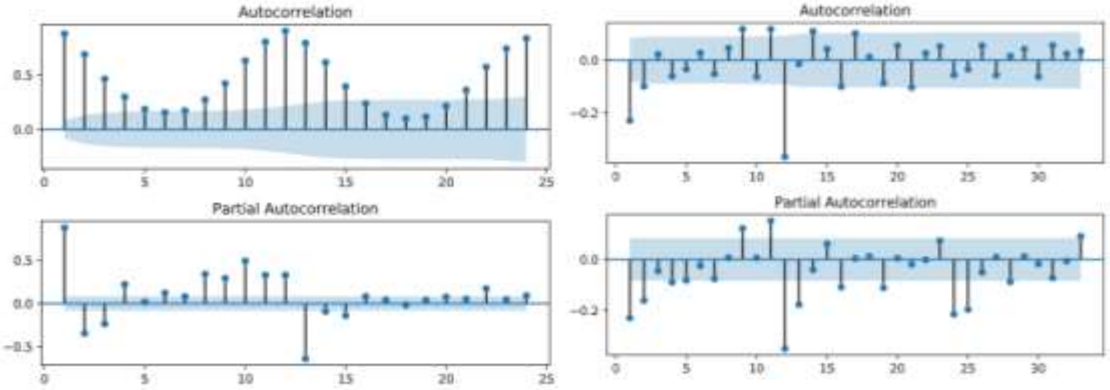
Trend & Seasonality exists -> Compare ADF test statistics of Seasonal Differenced Data and Double Differenced Data

-> Double Differenced Data has more negative statistic, thus more Stationary.

=> **S=12, D=1, d=1**

1. Model Identification Tools- ACF/PACF

- Finding Orders (p,d,q) (P,D,Q,)S



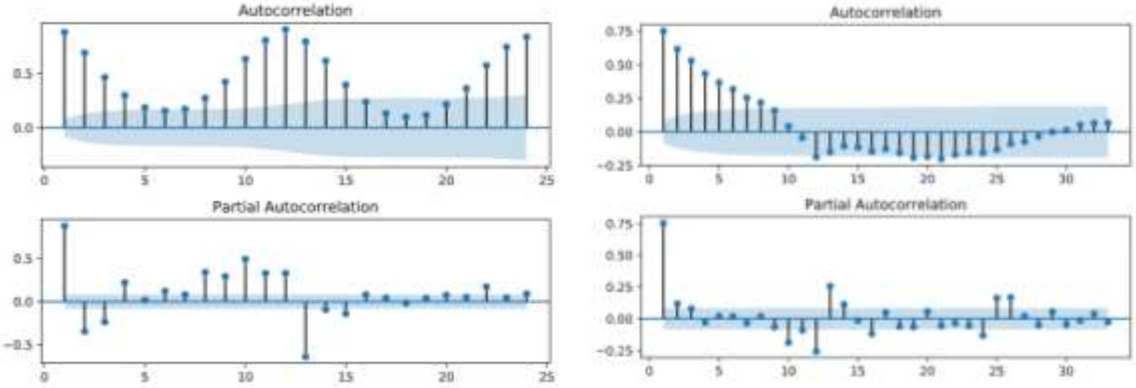
Normal ACF/PACF of Double Differenced Data

Negative First Autocorrelation -> Overdifferenced!

Using Seasonal Differenced Data! => ~~d=1~~ => **d=0**

1. Model Identification Tools- ACF/PACF

- Finding Orders (p,d,q) (P,D,Q,)S

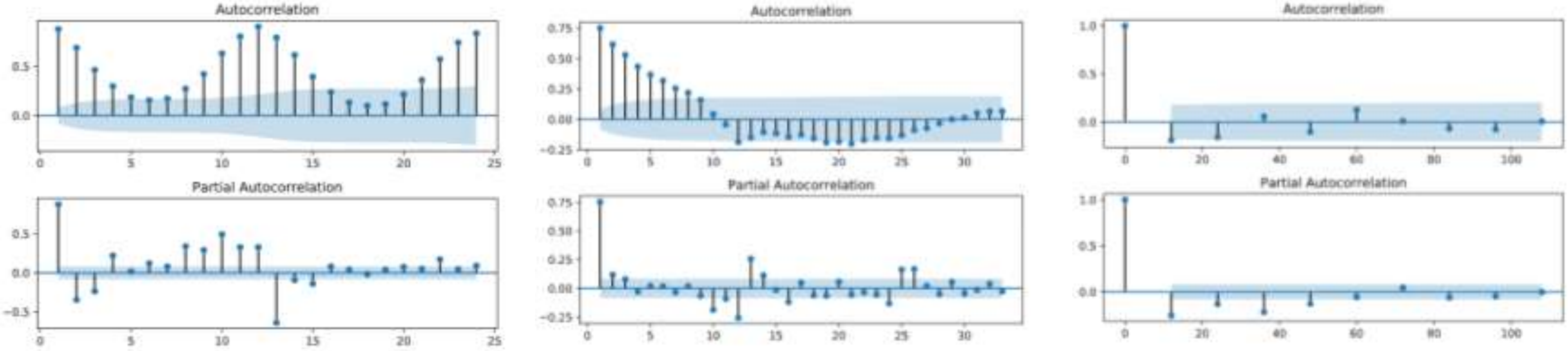


Normal ACF/PACF of Seasonal Differenced Data

Cut off after lag 1 in PACF & Tails off in ACF=> **p=1, q=0**

1. Model Identification Tools- ACF/PACF

- Finding Orders (p,d,q) (P,D,Q,)S



Seasonal ACF/PACF of Seasonal Differenced Data

Cut off after lag 1 in ACF & Tails off in PACF=> **P=0, Q=1**

Promising Order : non-seasonal order **(1,0,0)** / seasonal order **(0,1,1),12**

2. Model Estimation

- Parameter(Model Coefficients) Estimation
  using Train data set
- **Model(Order) Selection**

Tools

- Maximum Likelihood Estimation
- Un/Conditional Least Squares
- **AIC / BIC ; Model Selection**

2. Model Estimation – AIC & BIC

-**Lower** AIC indicates a **better** model

-AIC likes to choose simple models with lower order

-**AIC** is better at choosing **predictive models**

```
print(order_df.sort_values('aic'))
```

	p	q	P	Q	aic	bic
15	1	1	1	1	3112.918967	3134.376813
11	1	0	1	1	3129.972495	3147.138772
13	1	1	0	1	3183.475807	3200.642084
9	1	0	0	1	3190.917510	3203.792218
7	0	1	1	1	3339.591808	3356.758084
5	0	1	0	1	3369.698156	3382.572864
14	1	1	1	0	3426.025487	3443.191763
10	1	0	1	0	3432.552804	3445.427511
6	0	1	1	0	3614.584824	3627.459531
3	0	0	1	1	3624.548106	3637.422813
1	0	0	0	1	3638.128920	3646.712058
12	1	1	0	0	3669.730662	3682.605370
8	1	0	0	0	3670.421159	3679.004297
4	0	1	0	0	3813.022618	3821.605756
2	0	0	1	0	3893.520397	3902.103535
0	0	0	0	0	4055.250998	4059.542567

-**Lower** BIC indicates a **better** model

-BIC also likes to choose simple models with lower order, but **favors simpler model than AIC** does

-**BIC** is better at choosing **good explanatory models**

```
print(order_df.sort_values('bic'))
```

	p	q	P	Q	aic	bic
15	1	1	1	1	3112.918967	3134.376813
11	1	0	1	1	3129.972495	3147.138772
13	1	1	0	1	3183.475807	3200.642084
9	1	0	0	1	3190.917510	3203.792218
7	0	1	1	1	3339.591808	3356.758084
5	0	1	0	1	3369.698156	3382.572864
14	1	1	1	0	3426.025487	3443.191763
10	1	0	1	0	3432.552804	3445.427511
6	0	1	1	0	3614.584824	3627.459531
3	0	0	1	1	3624.548106	3637.422813
1	0	0	0	1	3638.128920	3646.712058
8	1	0	0	0	3670.421159	3679.004297
12	1	1	0	0	3669.730662	3682.605370
4	0	1	0	0	3813.022618	3821.605756
2	0	0	1	0	3893.520397	3902.103535
0	0	0	0	0	4055.250998	4059.542567

3. Model Diagnostic

- Checking Model adequacy with Residual

- Are the residuals uncorrelated?
- Are the residuals normally distributed?

Tools

- MAE / RMSE

- Diagnostics Plots

- Summary Statistics

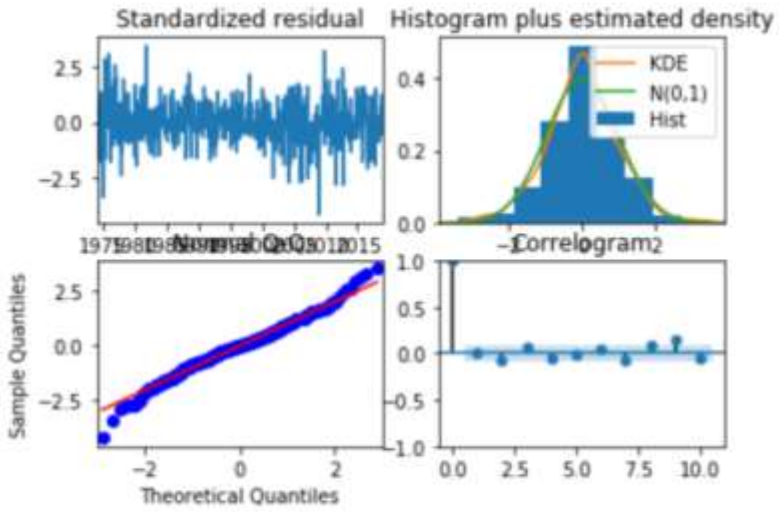
3. Model Diagnostic – Diagnostics Plots

```
model1 = SARIMAX(candy_sdiff, order=(1,0,1), seasonal_order=(1,1,1,12))
results = model1.fit()
```

```
print(mae)
```

3.2941547959168327

```
results.plot_diagnostics()
```



3. Model Diagnostic – Summary Statistics

Statespace Model Results

Dep. Variable:	IPG3113N	No. Observations:	552
Model:	SARIMAX(1, 0, 1)x(1, 1, 1, 12)	Log Likelihood	-1551.459
Date:	Wed, 27 Nov 2019	AIC	3112.919
Time:	16:19:58	BIC	3134.377
Sample:	01-01-1973	HQIC	3121.311
	- 12-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.8799	0.026	34.375	0.000	0.830	0.930
ma.L1	-0.2565	0.047	-5.421	0.000	-0.349	-0.164
ar.S.L12	-0.3626	0.039	-9.282	0.000	-0.439	-0.286
ma.S.L12	-0.9998	12.696	-0.079	0.937	-25.883	23.883
sigma2	16.5423	209.795	0.079	0.937	-394.649	427.734

Ljung-Box (Q):	152.00	Jarque-Bera (JB):	30.59
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	1.11	Skew:	-0.11
Prob(H) (two-sided):	0.48	Kurtosis:	4.14

Ljung-Box

- H0 : The data are independently distributed

- Applied to the residuals of fitted ARIMA model.

=> Test whether the residuals are autocorrelated.

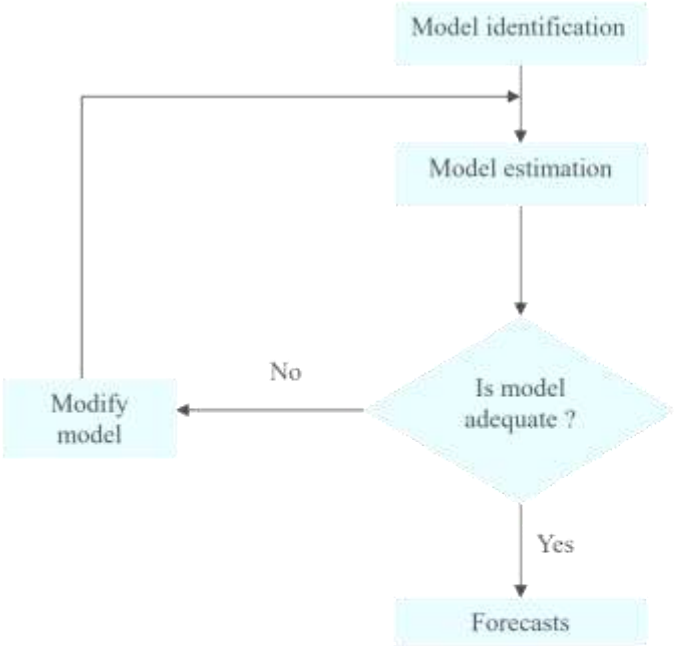
Jarque-Bera

- Goodness of fit test of whether sample data have the skewness and kurtosis matching a normal distribution.

- If it is far from zero, it means the data don't have a normal distribution

4. Modify or Forecast

Box-Jenkins Method

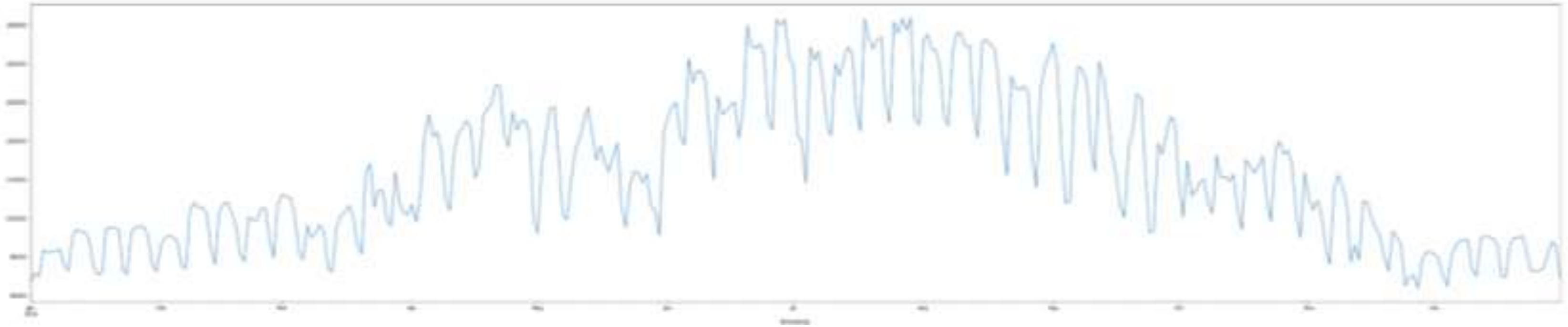


1. Model Identification – Time Series Plot

Site2, Office, Daily Sum

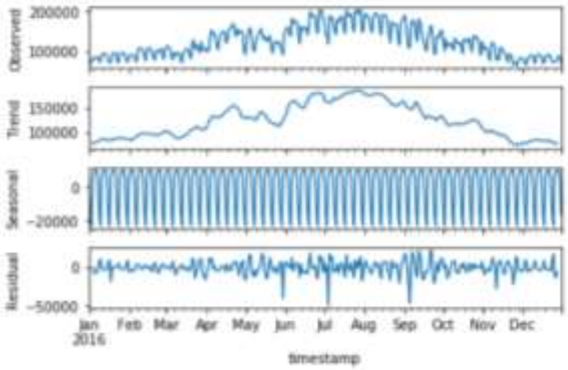
```python
# Time Plot
fig = plt.figure(figsize=(50,10))
daily_total2['meter_reading'].plot()
```

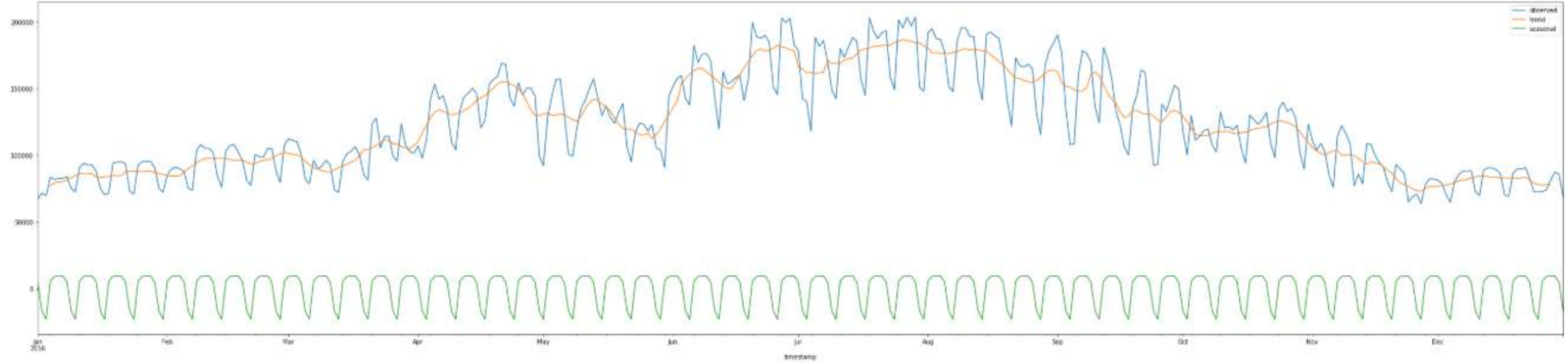
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a992b1f9e8>
```



1. Model Identification – Seasonal Decomposition

```
# Seasonal Decomposition
#from statsmodels.tsa.seasonal import seasonal_decompose

# Perform additive decomposition
decomp = seasonal_decompose(daily_total2['meter_reading'],freq=7,

# Plot decomposition
fig = plt.figure(figsize=(1000,10))
decomp.plot()
plt.show()
```





1. Model Identification– ADF Test

```
# ADF Test
#from statsmodels.tsa.stattools import adfuller
adf_results = adfuller(daily_total2['meter_reading'])
print('p-value is :', adf_results[1],
'test statistic is :', adf_results[0])
```

```
p-value is : 0.7261436282834549 test statistic is : -1.0719081742350394
```

```
# 1-order Differencing
daily_total2_diff = daily_total2[['meter_reading']].diff().dropna()
```

```
# ADF Test for Differenced Data
#from statsmodels.tsa.stattools import adfuller
adf_results = adfuller(daily_total2_diff['meter_reading'])
print('p-value is :', adf_results[1],
'test statistic is :', adf_results[0])
```

```
p-value is : 5.21821466096037e-07 test statistic is : -5.777493022727649
```

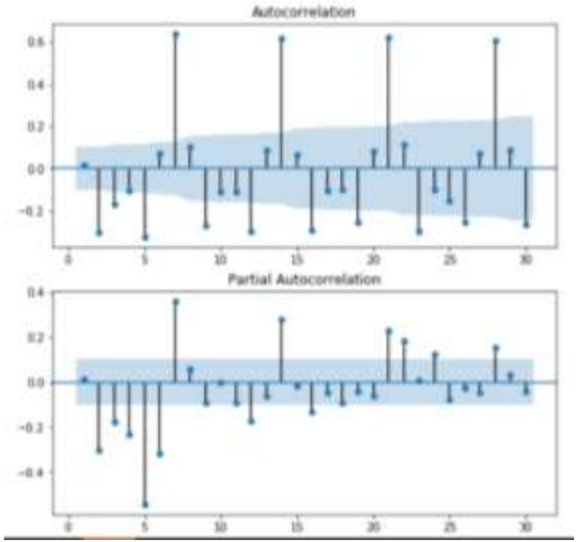
Non-Stationary

1st order Differencing

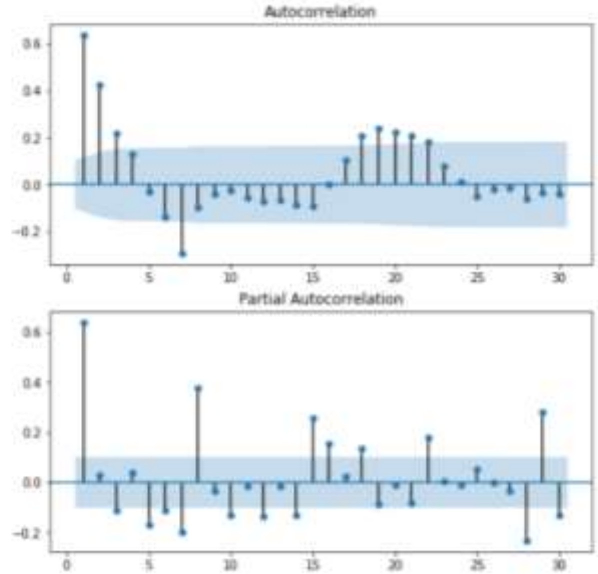
Stationary!

1. Model Identification- ACF/PACF

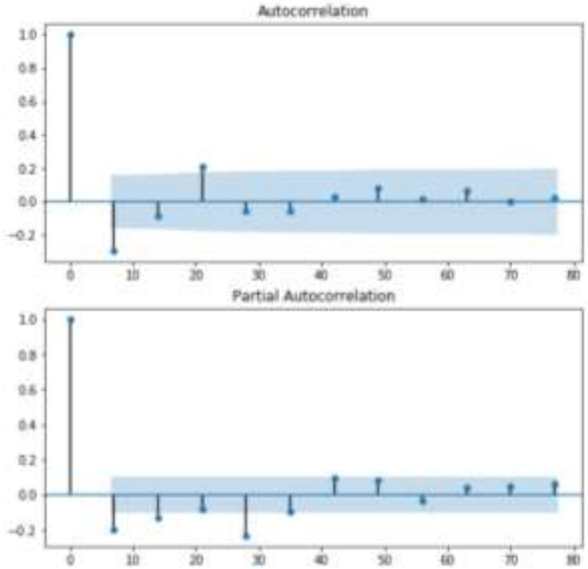
```python
# ACF/PACF of raw data
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

fig, (ax1, ax2) = plt.subplots(2,1,figsize=(8,8))

plot_acf(daily_total2_diff['meter_reading'], lags=30, zero=False,
plot_pacf(daily_total2_diff['meter_reading'], lags=30, zero=False,

plt.show()
```




```python
# Normal ACF/PACF of SDiff data
fig, (ax1, ax2) = plt.subplots(2,1,figsize=(8,8))

plot_acf(daily_total2_sdiff['meter_reading'], lags=30, zero=False
plot_pacf(daily_total2_sdiff['meter_reading'], lags=30, zero=Fals

plt.show()
```




```python
# Seasonal ACF/PACF of SDiff data
fig, (ax1, ax2) = plt.subplots(2,1,figsize=(8,8))

plot_acf(daily_total2_sdiff['meter_reading'], lags=[7*x for x in r
plot_pacf(daily_total2_sdiff['meter_reading'], lags=[7*x for x in

plt.show()
```



2. Model Estimation – AIC & BIC

```
f.sort_values(by='AIC').head()
```

	p	d	q	P	D	Q	AIC	BIC
609	2	1	1	1	0	2	-979.991806	-907.580127
615	2	1	1	2	0	1	-979.268989	-906.857310
372	1	1	1	2	0	1	-976.441646	-907.841108
258	1	0	0	1	0	2	-975.798797	-910.958578
456	1	1	2	2	0	2	-975.750983	-899.528163

609) SARIMA(2,1,1)(1,0,2)7

```
f.sort_values(by='BIC').head()
```

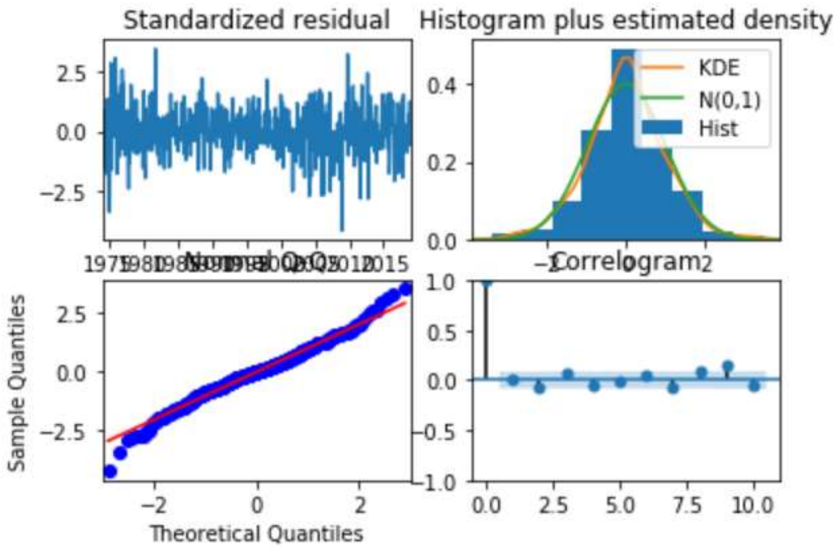
	p	d	q	P	D	Q	AIC	BIC
255	1	0	0	1	0	1	-973.818925	-912.792836
258	1	0	0	1	0	2	-975.798797	-910.958578
264	1	0	0	2	0	1	-975.450944	-910.610725
336	1	0	1	1	0	1	-974.297230	-909.457011
432	1	1	2	0	0	0	-970.244300	-909.266044

255) SARIMA(1,0,0)(1,0,1)7

3. Model Diagnostic – Diagnostics Plots

609) SARIMA(2,1,1)(1,0,2)7

```
results.plot_diagnostics()
```



3. Model Diagnostic – Summary Statistics

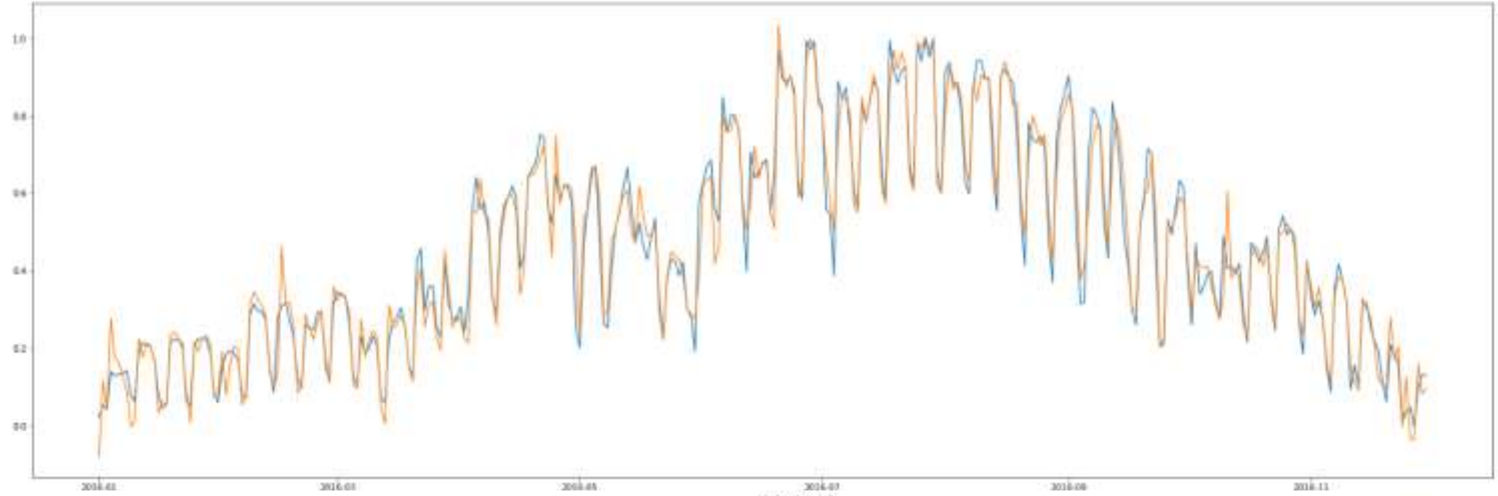
609) SARIMA(2,1,1)(1,0,2)7

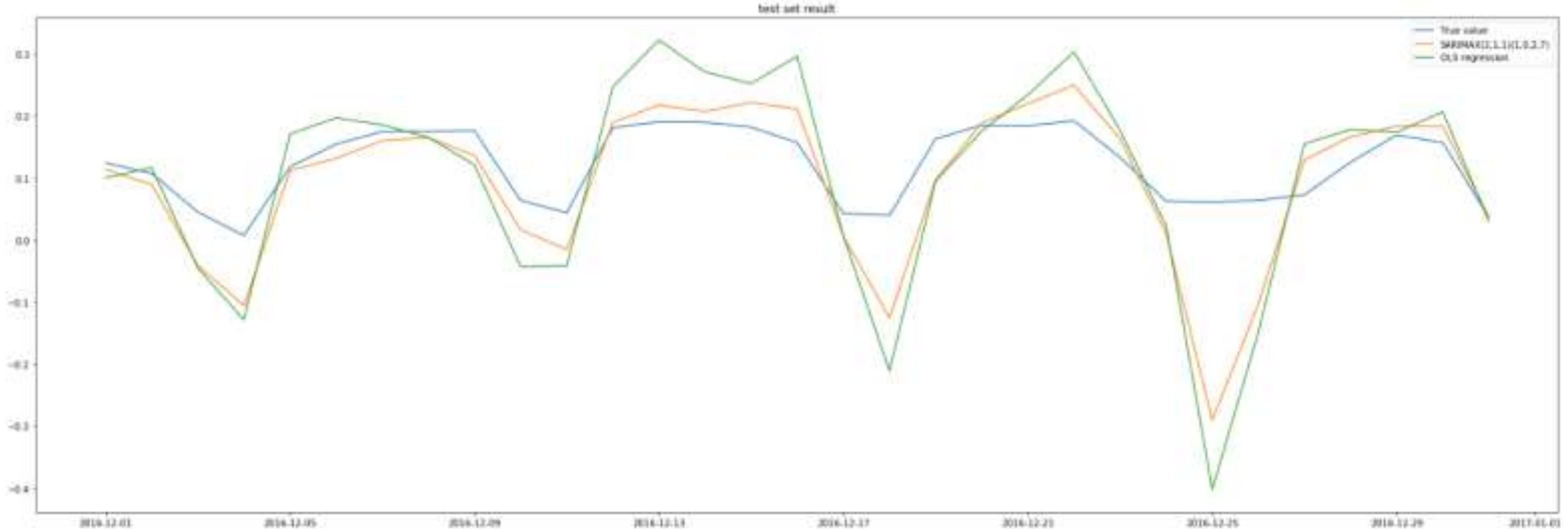
Statespace Model Results

Dep. Variable:	meter_reading	**No. Observations:**	335
Model:	SARIMAX(2, 1, 1)x(1, 0, 2, 7)	**Log Likelihood**	508.996
Date:	Wed, 27 Nov 2019	**AIC**	-979.992
Time:	13:40:22	**BIC**	-907.580
Sample:	01-01-2016	**HQIC**	-951.120
	- 11-30-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
air_temperature	0.5216	0.059	8.769	0.000	0.405	0.638
weekend	-0.1164	0.008	-15.136	0.000	-0.131	-0.101
holiday	-0.1651	0.009	-18.962	0.000	-0.182	-0.148
dew_temperature	0.1449	0.026	5.656	0.000	0.095	0.195
sea_level_pressure	0.0697	0.039	1.802	0.072	-0.006	0.145
hot_temperature^2	0.2845	0.027	10.687	0.000	0.232	0.337
hot	-0.1874	0.016	-11.418	0.000	-0.220	-0.155
Sunday	-0.0919	0.006	-15.152	0.000	-0.104	-0.080
Saturday	-0.0488	0.009	-5.523	0.000	-0.066	-0.031
Tuesday	0.0033	0.010	0.336	0.737	-0.016	0.022
cold	0.0177	0.026	0.666	0.505	-0.034	0.070
Friday	-0.0273	0.015	-1.878	0.060	-0.056	0.001
ar.L1	0.6430	0.086	7.495	0.000	0.475	0.811
ar.L2	-0.1042	0.065	-1.595	0.111	-0.232	0.024
ma.L1	-0.8283	0.080	-10.354	0.000	-0.985	-0.671
ar.S.L7	0.7255	0.172	4.217	0.000	0.388	1.063
ma.S.L7	-0.7096	0.177	-3.999	0.000	-1.057	-0.362
ma.S.L14	0.1314	0.064	2.061	0.039	0.006	0.256
sigma2	0.0028	0.000	15.935	0.000	0.002	0.003

Ljung-Box (Q):	38.47	**Jarque-Bera (JB):**	63.61


1.0
0.8
0.6
0.4
0.2
0.0


test set result
True value
OLS regression
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
-0.4

Time se


Do you
have any
question?
Thank you
for your attention.