Wk12_seasonal decomposition

January 12, 2021

0.1 Seasonal Decomposition

Amin Mansouri 18097367

0.1.1 alle imports

[17]:
```
import random
import time
import torch
import torchvision
import torchvision.transforms as transforms
import numpy as np
import pandas as pd
import math
import random

import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import seaborn as sns
import matplotlib.pyplot as plt

from tqdm import tqdm
from IPython.display import clear_output
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.metrics import mean_absolute_error
import wandb

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.seasonal import STL
```

```
import warnings
import copy
from pylab import rcParams
```

[18]:
```
# CUDA initialisation
ngpu = torch.cuda.device_count() # number of available gpus
device = torch.device("cuda:4") if (torch.cuda.is_available() and ngpu > 0)␣
 ↪else "cpu" #cuda:0 for gpu 0, cuda:4 for gpu 5
torch.backends.cudnn.benchmark=True # Uses cudnn auto-tuner to find the best␣
 ↪algorithm to use for your hardware
```

0.1.2 Dataframe

[19]:
```
df = pd.read_pickle('28_df_consumption_weekday_dagterug')
df = df.drop('day_0', axis=1)
df = df['2019-09-02':'2019-11-29']
temp = pd.DataFrame()
temp['consumption'] = df['consumption']
temp['cons_T-24'] = df['cons_T-24']
temp['cons_T-25'] = df['cons_T-25']
temp['cons_T-26'] = df['cons_T-26']
temp['cons_T-48'] = df['cons_T-38']
df = pd.DataFrame()
df = temp
df.head()
```

[19]:

	consumption	cons_T-24	cons_T-25	cons_T-26	cons_T-48
2019-09-02 00:00:00	0.183	0.170	0.228	0.281	0.745
2019-09-02 01:00:00	0.239	0.171	0.170	0.228	0.378
2019-09-02 02:00:00	0.290	0.246	0.171	0.170	0.461
2019-09-02 03:00:00	0.302	0.422	0.246	0.171	0.333
2019-09-02 04:00:00	0.234	1.188	0.422	0.246	2.080

[20]:
```
df.tail()
```

[20]:

	consumption	cons_T-24	cons_T-25	cons_T-26	cons_T-48
2019-11-29 19:00:00	0.558	0.602	0.813	0.721	0.400
2019-11-29 20:00:00	0.819	0.632	0.602	0.813	0.599
2019-11-29 21:00:00	0.638	0.718	0.632	0.602	0.398
2019-11-29 22:00:00	0.591	0.561	0.718	0.632	0.418
2019-11-29 23:00:00	1.365	0.465	0.561	0.718	0.337

[21]:
```
#scale the data
#X:
scalerx = StandardScaler()
scalerx.fit(df.loc[:,~df.columns.isin(["consumption"])])
```

```
scaled_dataX = scalerx.transform(df.loc[:,~df.columns.isin(["consumption"])]).
 ↪tolist()
#Y:
scalery = StandardScaler()
scalery.fit(df.loc[:,df.columns.isin(["consumption"])])
datay = scalery.transform(df.loc[:,df.columns.isin(["consumption"])])

#split the data
train_X = scaled_dataX[0:24*7*4]
train_y = datay[0:24*7*4].reshape(-1,1)

valid_X = scaled_dataX[24*7*4:24*7*5]
valid_y = datay[24*7*4:24*7*5].reshape(-1,1)

test_X = scaled_dataX[24*7*5:24*7*6]
test_y = datay[24*7*5:24*7*6].reshape(-1,1)

#Make tensors from the numpy arrays.
train_X_t = torch.from_numpy(np.array(train_X)).to(device).float()
train_y_t = torch.from_numpy(np.array(train_y)).to(device).float()

valid_X_t = torch.from_numpy(np.array(valid_X)).to(device).float()
valid_y_t = torch.from_numpy(np.array(valid_y)).to(device).float()

test_X_t = torch.from_numpy(np.array(test_X)).to(device).float()
test_y_t = torch.from_numpy(np.array(test_y)).to(device).float()

#Tensor Datasets
train_set = torch.utils.data.TensorDataset(train_X_t, train_y_t)
test_set = torch.utils.data.TensorDataset(valid_y_t, valid_X_t)

#Tensor DataLoaders
train_loader = torch.utils.data.DataLoader(train_set, batch_size=24,␣
 ↪shuffle=False, num_workers = 0)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=24,␣
 ↪shuffle=False, num_workers = 0)
```

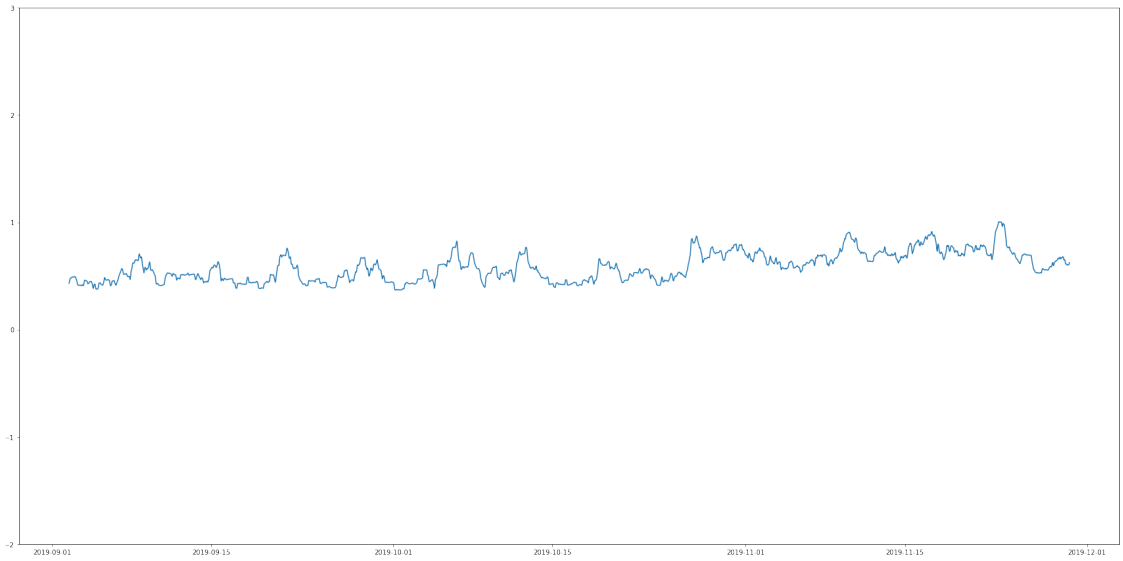
Code Seasonal Decomposition

[22]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

[23]:
```
result = seasonal_decompose(df["consumption"], model='additive',period = 24)
```

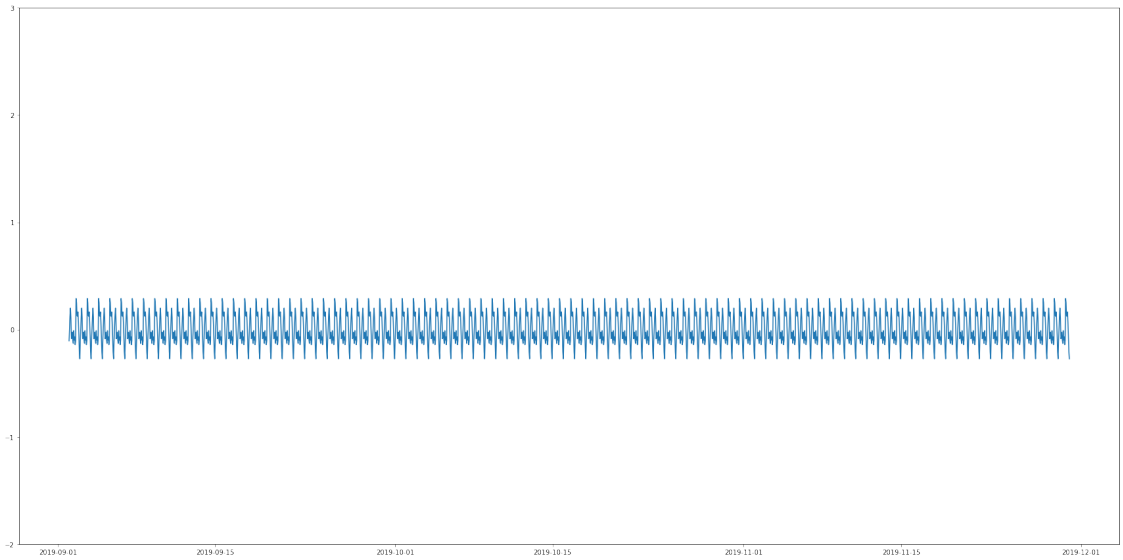
[32]:
```
plt.subplots(figsize=(30,15))
plt.plot(result.trend)
plt.ylim([-2,3])
```

[32]: (-2.0, 3.0)



[28]:
```
plt.subplots(figsize=(30,15))
plt.plot(result.seasonal)
plt.ylim([-2,3])
```

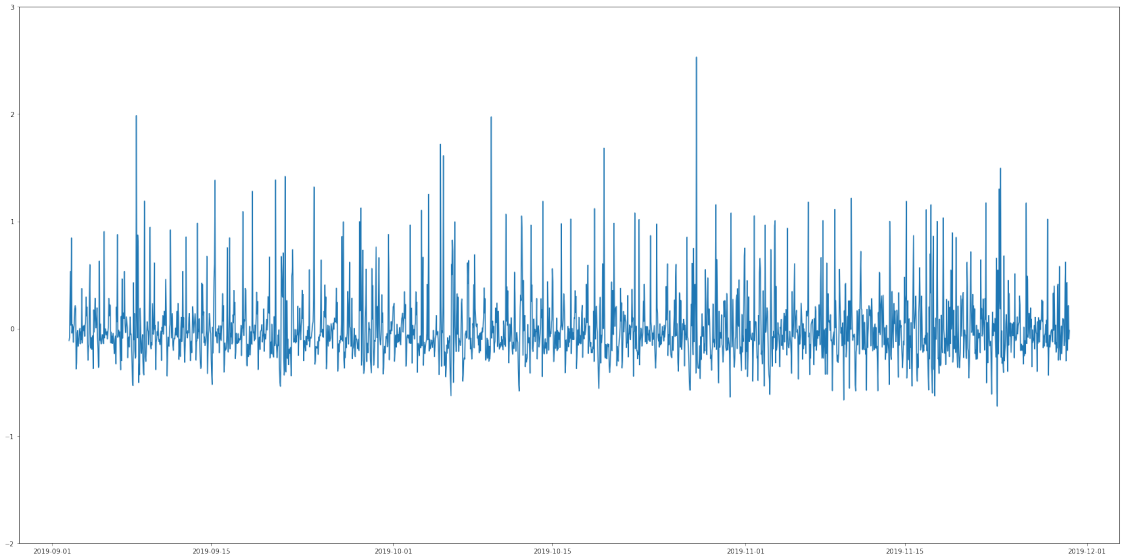
[28]: (-2.0, 3.0)



[29]:
```
plt.subplots(figsize=(30,15))
plt.plot(result.resid)
```

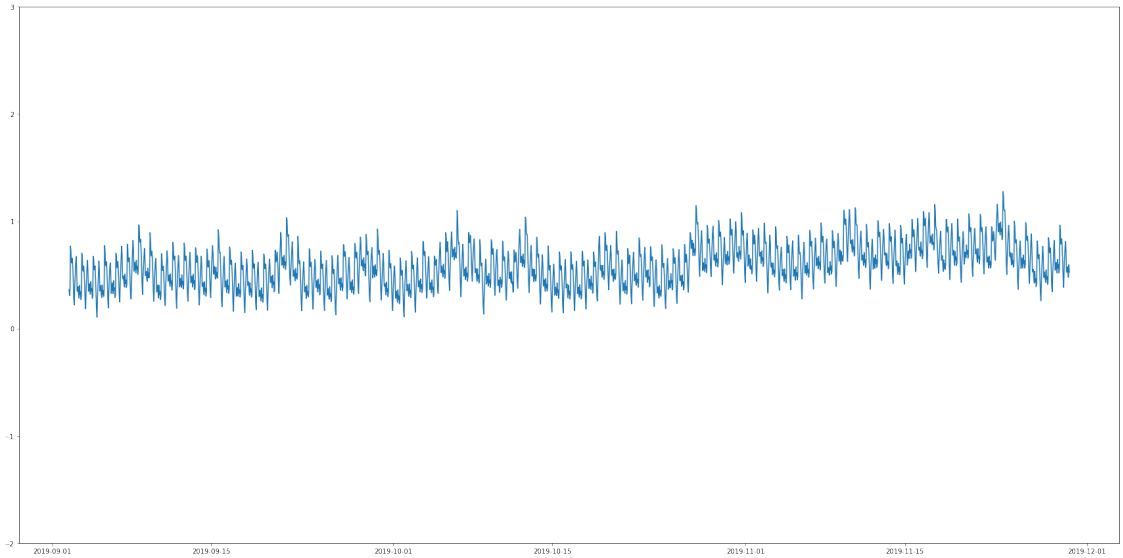
```
plt.ylim([-2,3])
```

[29]: (-2.0, 3.0)



[31]:
```
plt.subplots(figsize=(30,15))
plt.plot(result.seasonal+result.trend)
plt.ylim([-2,3])
```

[31]: (-2.0, 3.0)



[ ]: