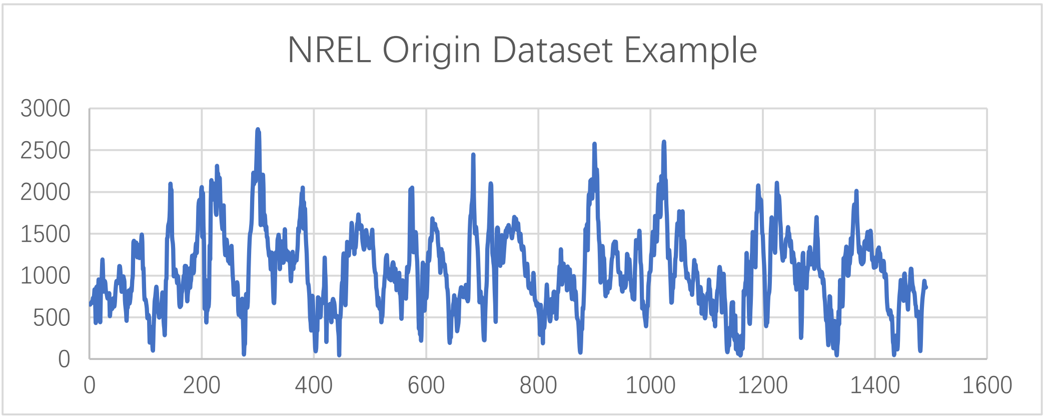
A Brief Report

1. Day head Forecast Model

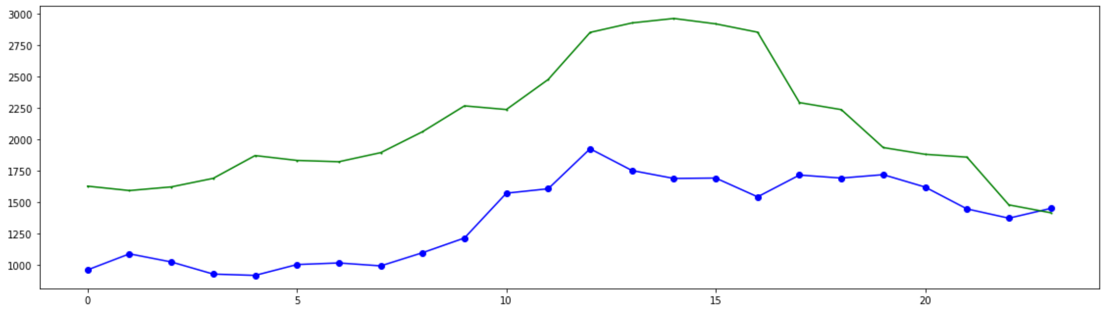
   NREL Wind Power



   # The Former Data
   # In this part, we will calculate the Data from N to 1 day(s) before the
   # forecasting time. The Value we get are R(t-24N)-R(t-24N-1), R(T-24N) and
   # R(T-24N-1)-R(T-24N-2).
   # In the test before, we are informed that the Delta Data are helpful to
   # gain a more precise predict. However, if we just input the Data like
   # R(t-24N-i), i=0,1,2 into the estimate machine, the constrains on the Data
   # are too much for the model or in another word, this method provides us too
   # much information, which can mislead the forecast machine.
   # And, significantly, the origin Dataset is from ReducedNoiseData.csv

   Result

   GBRT



   Not accurate enough.

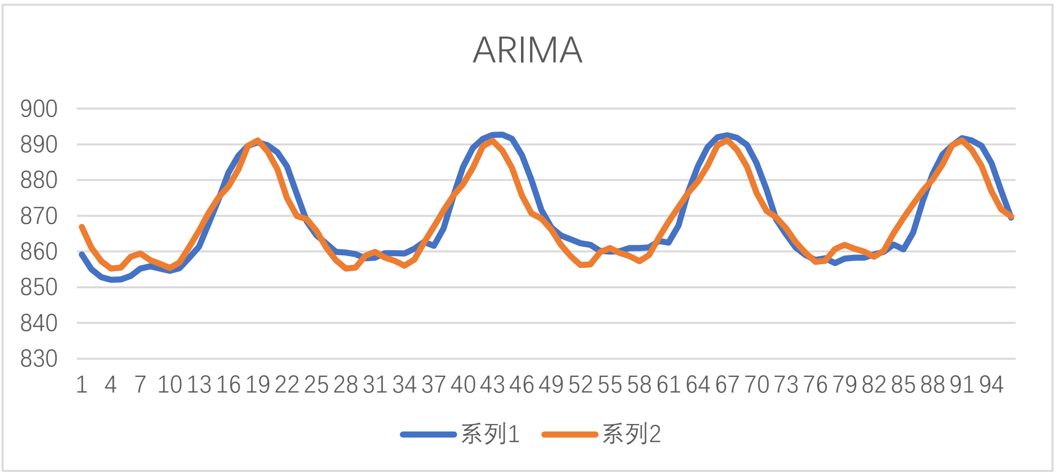
2. ARIMA Model

   ARIMA 模型的全称叫做自回归移动平均模型，全称是(ARIMA, Autoregressive Integrated Moving Average Model)。也记作 ARIMA(p,d,q)，是统计模型(statistic model)中最常见的一种用来进行时间序列预测的模型。

   Data: ISO NEW ENGLAND

   Results

   5000+96h

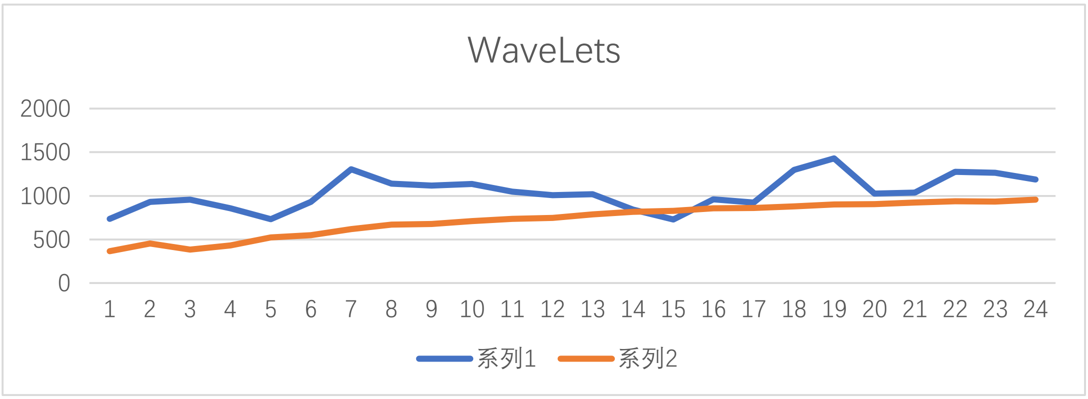


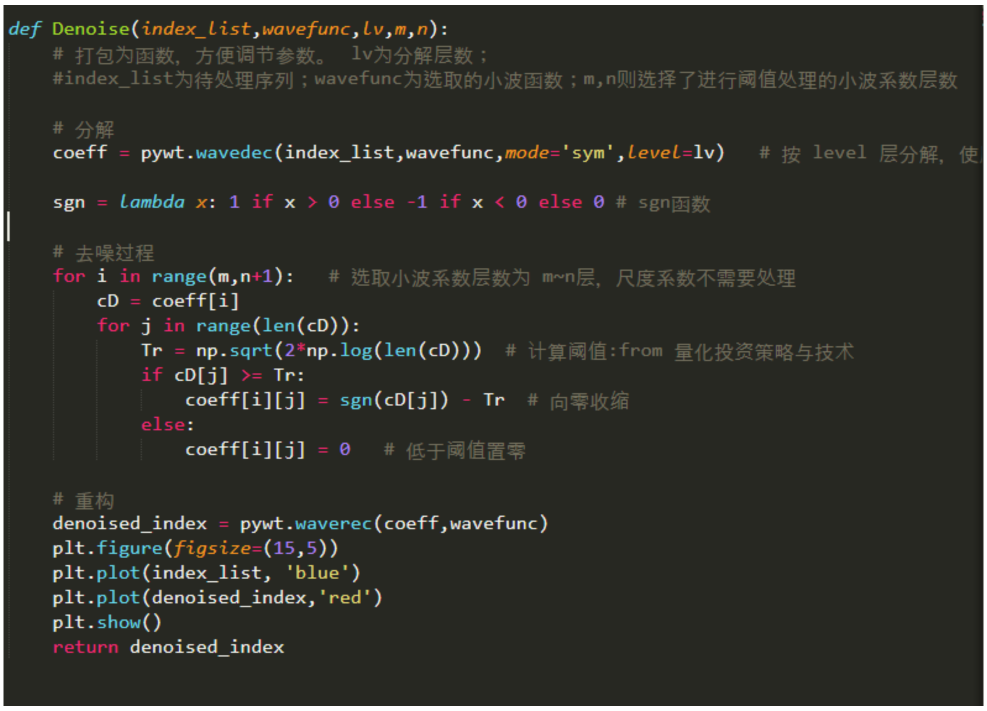
优点：比较准确

缺点：非常耗时；非常耗内存；数据稳定性依赖强；数据容易失稳。

3. Wavelet

   Failed to predict… but good for de-noise.

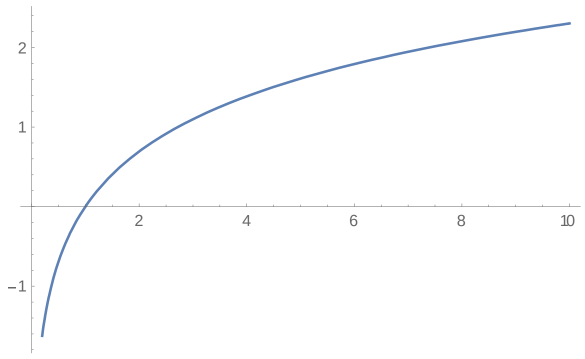




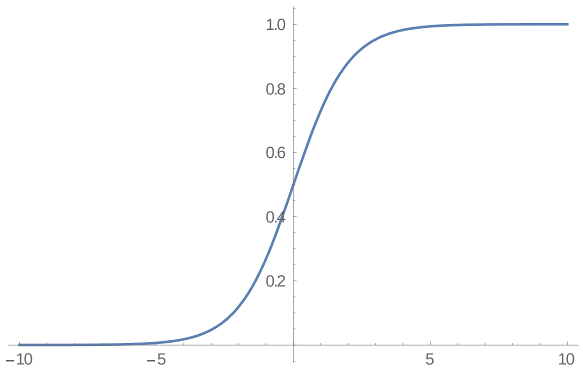
```python
def Denoise(index_list,wavefunc,lv,m,n):
    # 打包为函数, 方便调节参数。  lv为分解层数；
    #index_list为待处理序列；wavefunc为选取的小波函数；m,n则选择了进行阈值处理的小波系数层数

    # 分解
    coeff = pywt.wavedec(index_list,wavefunc,mode='sym',level=lv)   # 按 level 层分解, 使

    sgn = lambda x: 1 if x > 0 else -1 if x < 0 else 0 # sgn函数

    # 去噪过程
    for i in range(m,n+1):   # 选取小波系数层数为 m~n层, 尺度系数不需要处理
        cD = coeff[i]
        for j in range(len(cD)):
            Tr = np.sqrt(2*np.log(len(cD)))  # 计算阈值:from 量化投资策略与技术
            if cD[j] >= Tr:
                coeff[i][j] = sgn(cD[j]) - Tr  # 向零收缩
            else:
                coeff[i][j] = 0   # 低于阈值置零

    # 重构
    denoised_index = pywt.waverec(coeff,wavefunc)
    plt.figure(figsize=(15,5))
    plt.plot(index_list, 'blue')
    plt.plot(denoised_index,'red')
    plt.show()
    return denoised_index
```

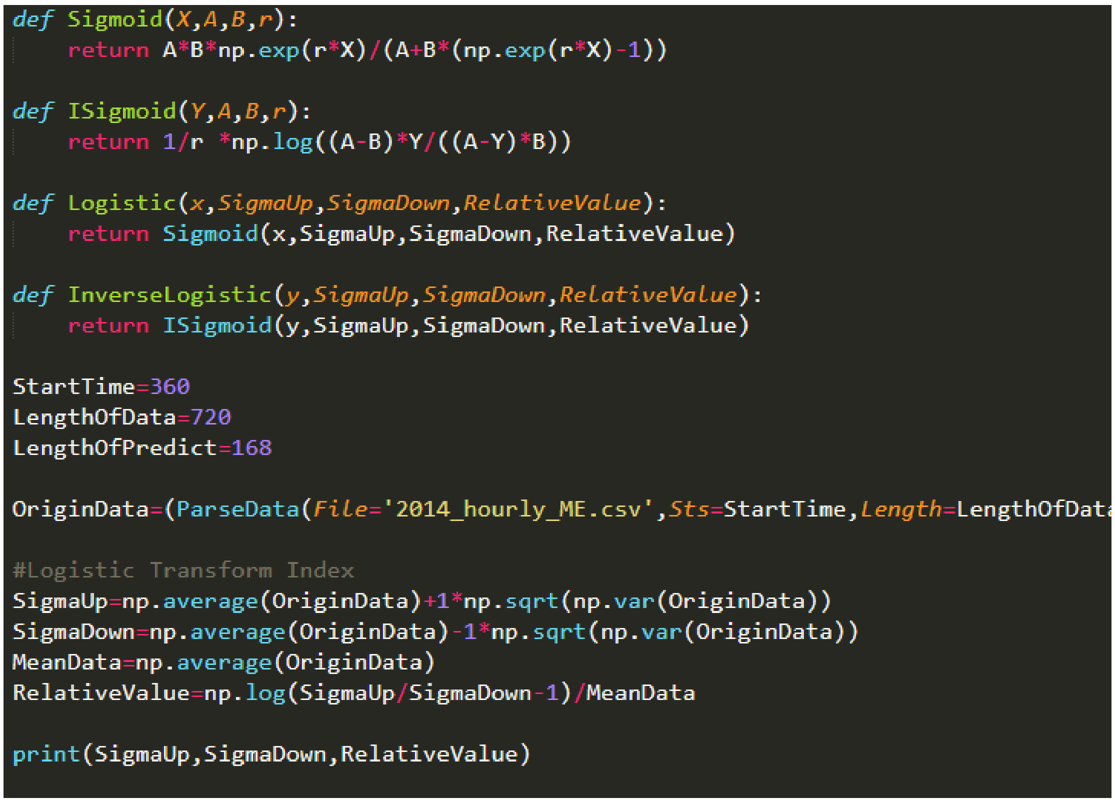
4. Punish

Log Punish



Logistic Punish

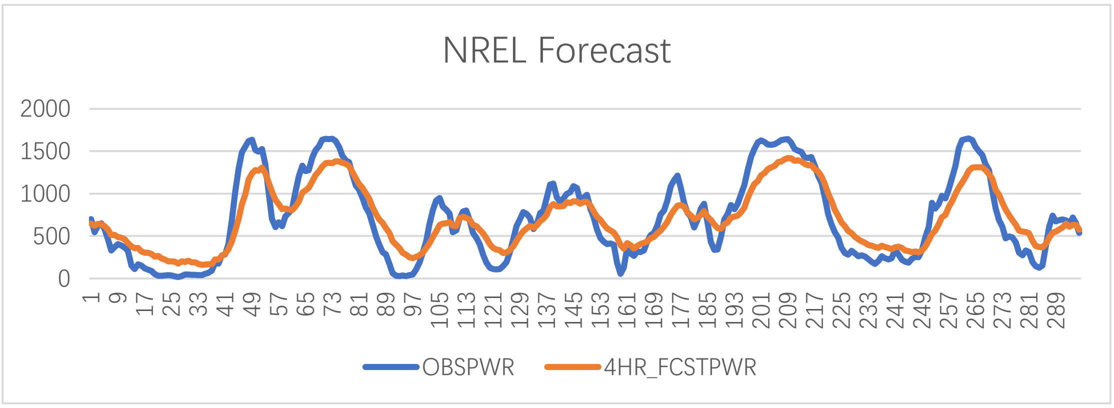




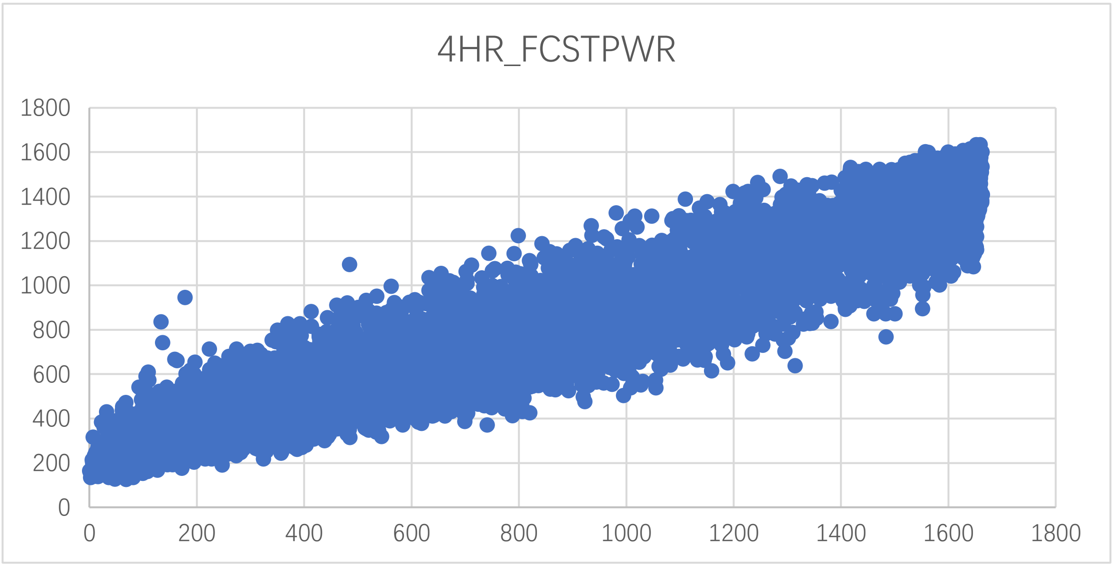
```python
def Sigmoid(X,A,B,r):
    return A*B*np.exp(r*X)/(A+B*(np.exp(r*X)-1))

def ISigmoid(Y,A,B,r):
    return 1/r *np.log((A-B)*Y/((A-Y)*B))

def Logistic(x,SigmaUp,SigmaDown,RelativeValue):
    return Sigmoid(x,SigmaUp,SigmaDown,RelativeValue)

def InverseLogistic(y,SigmaUp,SigmaDown,RelativeValue):
    return ISigmoid(y,SigmaUp,SigmaDown,RelativeValue)

StartTime=360
LengthOfData=720
LengthOfPredict=168

OriginData=(ParseData(File='2014_hourly_ME.csv',Sts=StartTime,Length=LengthOfDat

#Logistic Transform Index
SigmaUp=np.average(OriginData)+1*np.sqrt(np.var(OriginData))
SigmaDown=np.average(OriginData)-1*np.sqrt(np.var(OriginData))
MeanData=np.average(OriginData)
RelativeValue=np.log(SigmaUp/SigmaDown-1)/MeanData

print(SigmaUp,SigmaDown,RelativeValue)
```

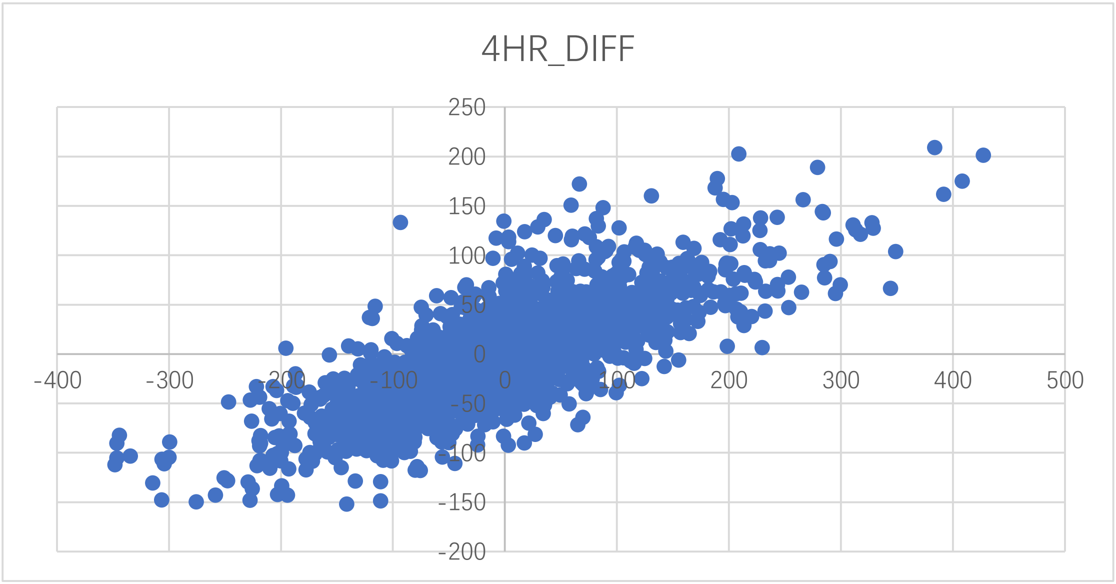
5. Enhancement

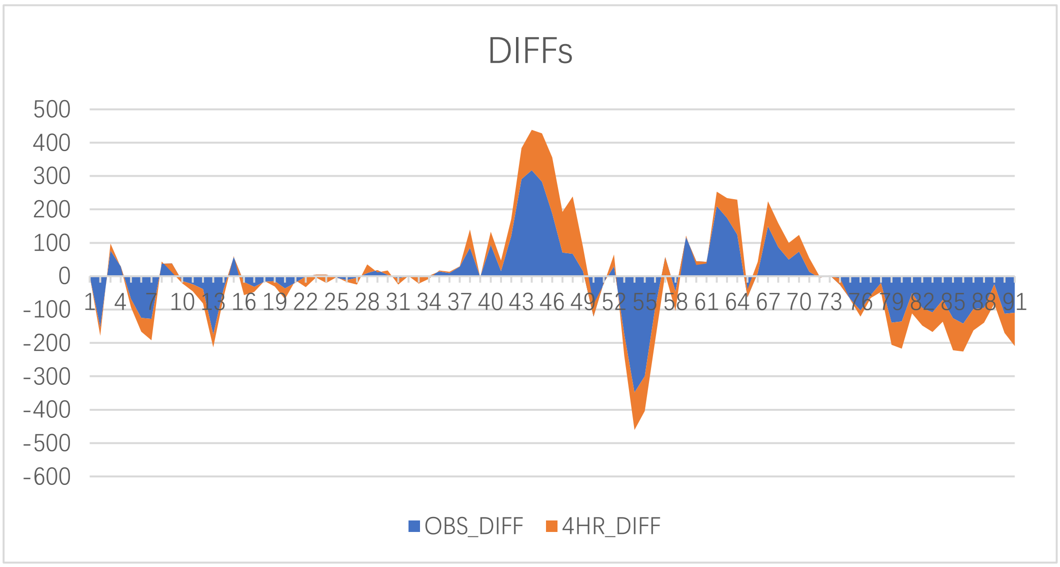
Dataset

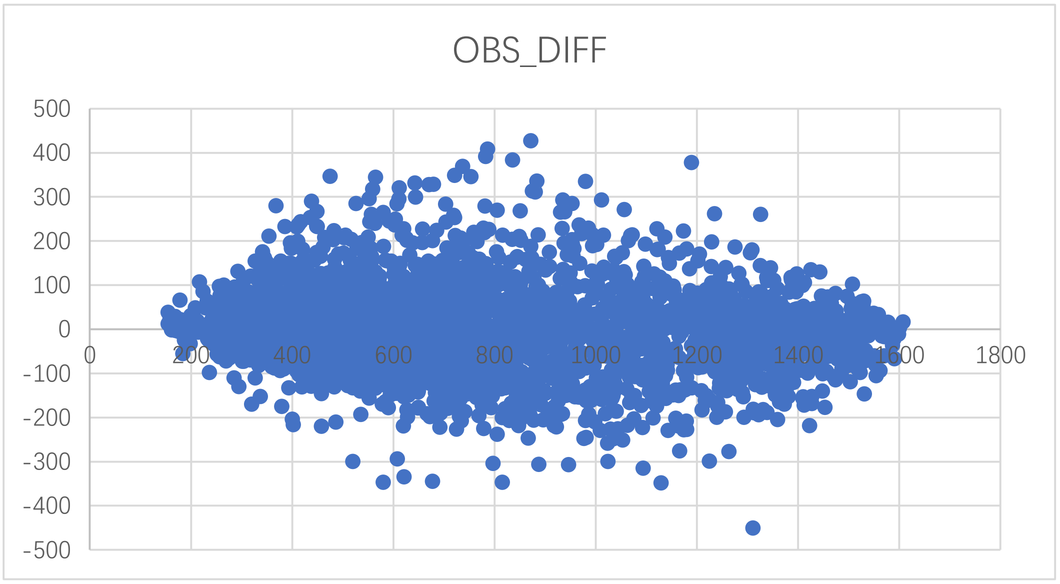


Aim: Enhance the data to eliminate the error. (To build unbiased estimator)

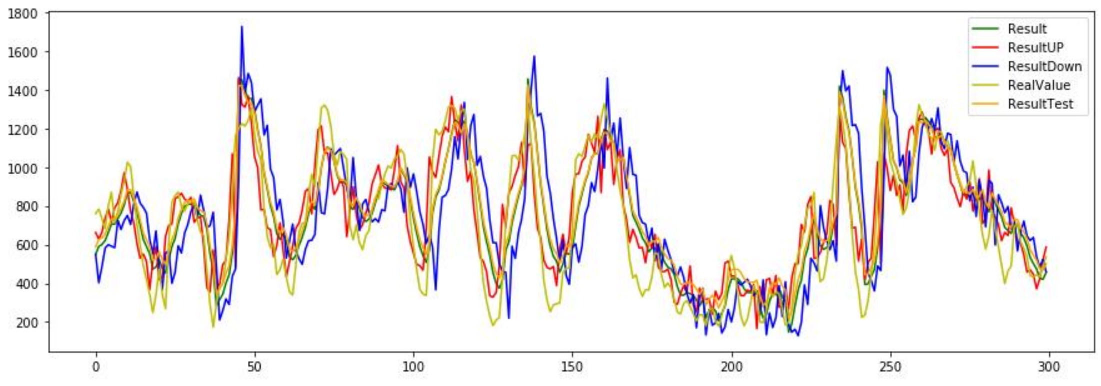








Result



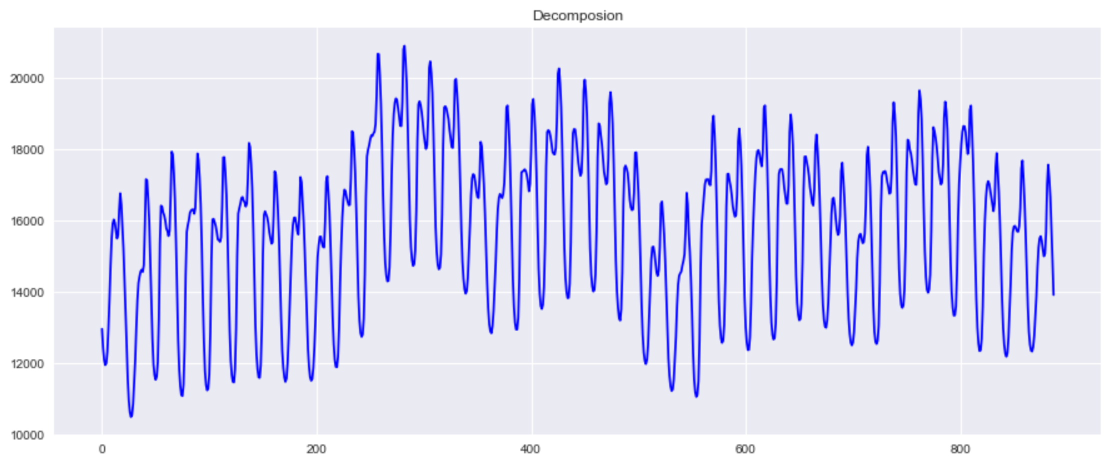
Suspended…

6. A brief Report

1. Dataset

The System Load Data are derived from ISO New England Control Area within year of 2014.

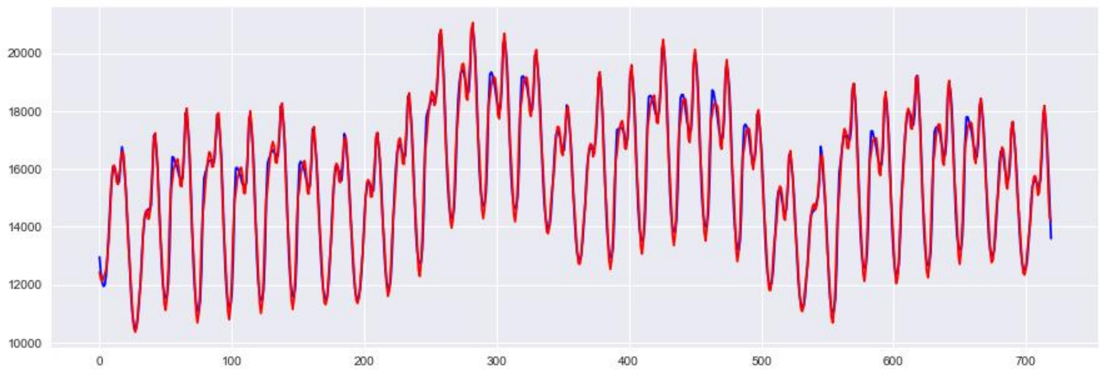
The graph below shows the actual hourly data from 2014/01/11 0:00 and followed with 720 + 168 hours in order to build the model and predict.



2. Steps of the algorithm

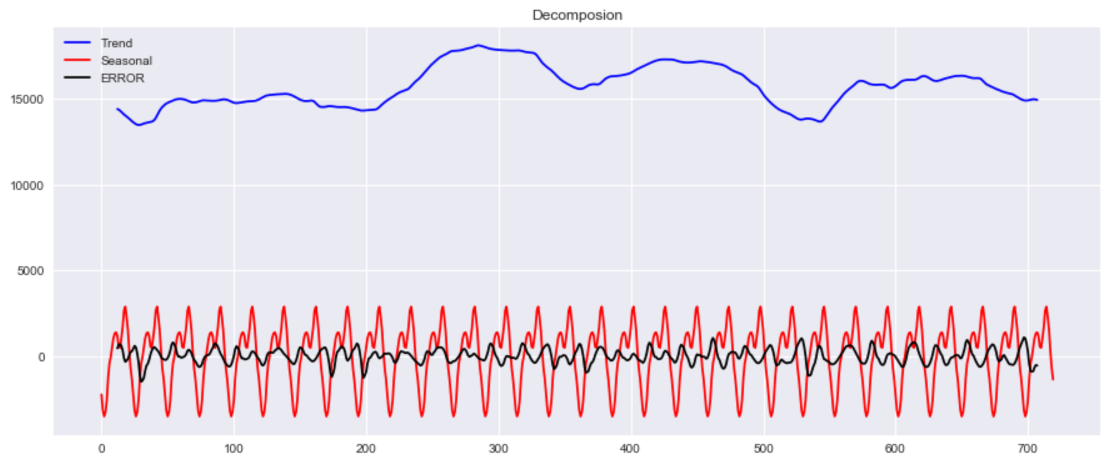
a) De-noise applied with wavelet

Though it is not essential here since the data is smooth enough, I do it otherwise.



b) Make a seasonal decompose 1st time

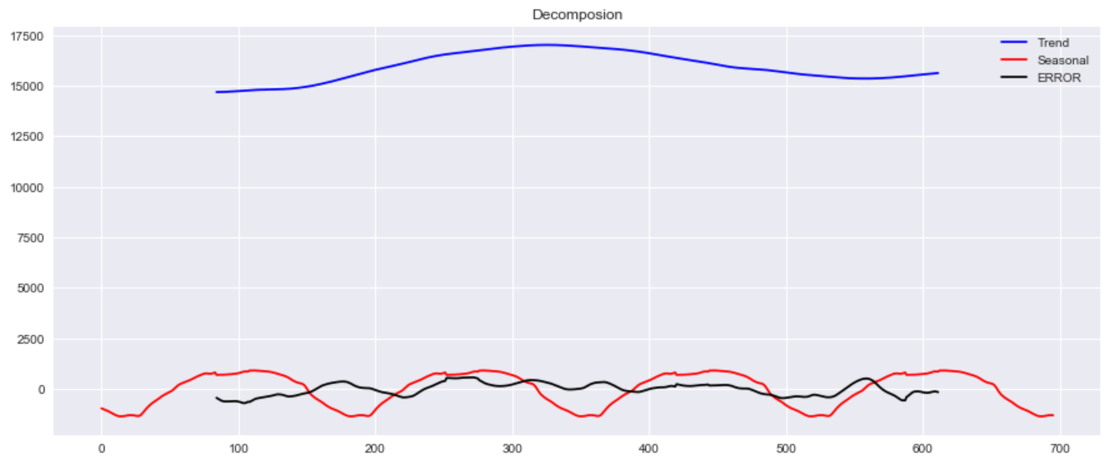
After the decomposing process, we gain the trends, the seasonal character and the error. I set the window to 24H, since it is obvious that there is any periodical fluctuation here.



Fortunately, The Error is not that large.

c) The 2nd Decomposing

There should be some information hiding in the trend curves. i.e., the 7*24H periodicity. Decomposing for the 2nd time!



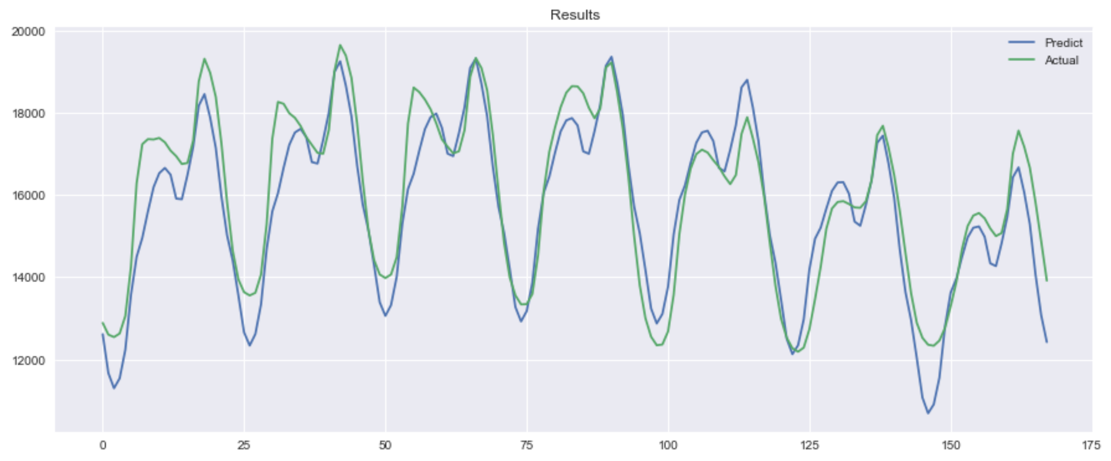
The result is acceptable, and the 7*24H periodicity lies here. People work mostly on weekdays.

d) Make a naïve predict of the trend

If the interval of the predicted time is not that long, we can naively assume that the change of the trend is linear. Thus, just Draw a line.

e) Assemble these features

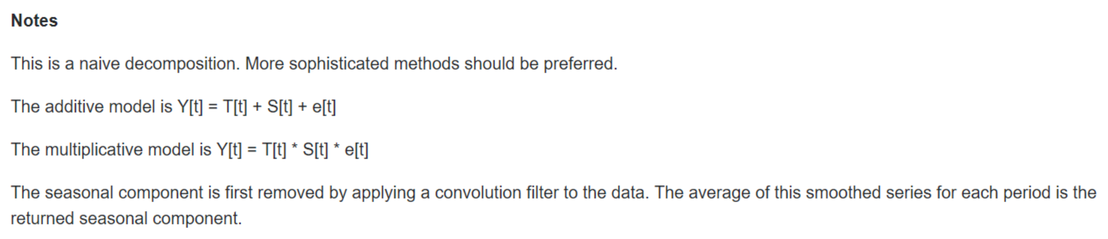
Sum them up.



f) Notes

The decomposing process using the function **seasonal_decompose()** from **statsmodels.tsa.seasonal.**

The author says



Notes

This is a naive decomposition. More sophisticated methods should be preferred.

The additive model is Y[t] = T[t] + S[t] + e[t]

The multiplicative model is Y[t] = T[t] * S[t] * e[t]

The seasonal component is first removed by applying a convolution filter to the data. The average of this smoothed series for each period is the returned seasonal component.

3. Error

4.5%