Guzman Assignment 3

Gu Wu UNI: gw2406

February 10, 2021

1. Prediction and EDA
   In this specific assignment, I unitized Recurrent Neural network. Theoretically, a huge RNN should be able to learn sequential data of arbitrary complexity; However, in the real world practical applications, vanilla Recurrent Neural Networks show their incapabilities of storing information that are far away,which limits their ability to model long-term dependent sequence data. The challenge here is known as Gradient Vanishing or Gradient Explosion. The issue here is that if the network's predictions are only based on the last few inputs, and these inputs were themselves predicted by the network, it has little chances to recover from the past mistakes. Therefore, for this reason, having a longer memory is remarkably desirable because even if the network cannot utilize the most recent information that is available to it, it can take a step back and seek for pieces of information that are further back in the past to formulate its forecasting. To overcome the drawback of error back-flow, Long Short-Term Memory (LSTM) that was equipped with a special gating mechanism was introduced by Hochreiter and Schmidhuber as an extension of the basic version of RNN.

   I first trained on the price time series by itself, and make the data suitable for the prediction. The code below essentially preprocessed the data and make it suitable for training.

```
def prepare_data(timeseries_data, n_features):
    X, y =[],[]
    for i in range(len(timeseries_data)):

        end_ix = i + n_features

        if end_ix > len(timeseries_data)-1:
            break

        seq_x, seq_y = timeseries_data[i:end_ix], timeseries_data[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

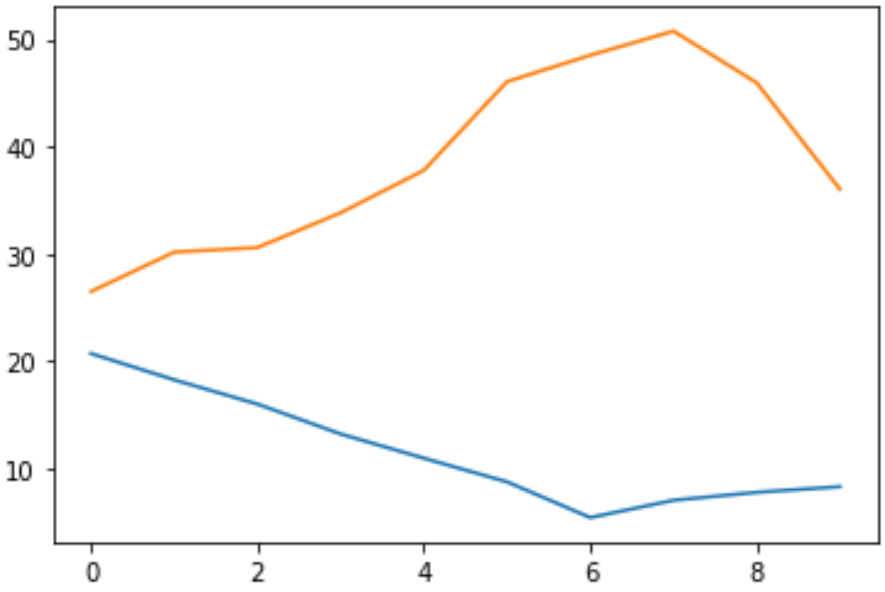
   I trained my model using 2 layer of LSTM cells and trained for 100 epoches

```
Epoch 94/100
13990/13990 [==============================] - 18s 1ms/sample - loss: 1386.6478
Epoch 95/100
13990/13990 [==============================] - 18s 1ms/sample - loss: 1365.9617
Epoch 96/100
13990/13990 [==============================] - 18s 1ms/sample - loss: 1361.0785
Epoch 97/100
13990/13990 [==============================] - 18s 1ms/sample - loss: 1454.9136
Epoch 98/100
13990/13990 [==============================] - 17s 1ms/sample - loss: 1853.7286
Epoch 99/100
13990/13990 [==============================] - 16s 1ms/sample - loss: 1430.6251
Epoch 100/100
13990/13990 [==============================] - 16s 1ms/sample - loss: 1419.8911
```

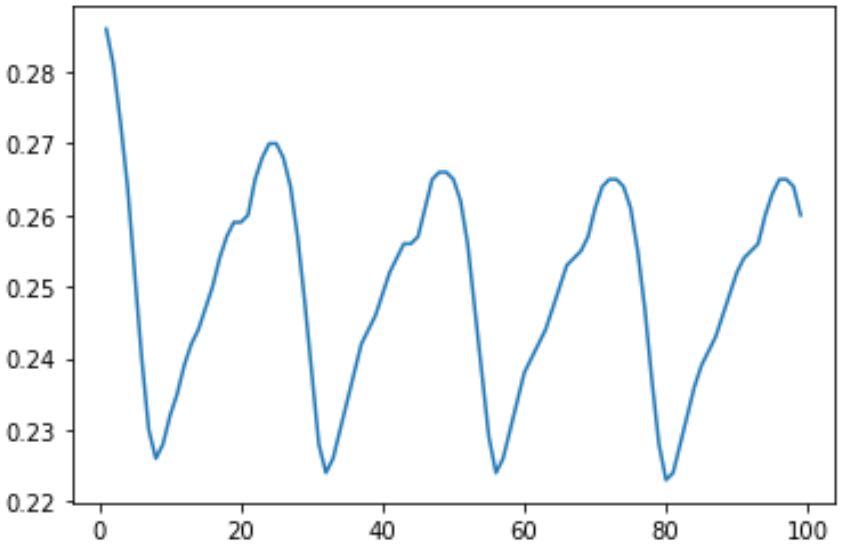
   and the code below describes how it is making prediction

```
x_input = np.array(df_new['HB_NORTH (RTLMP)'])[-987:-977]
temp_input=list(x_input)
lst_output=[]
i=0
while(i<10):

    if(len(temp_input)>10):
        x_input=np.array(temp_input[1:])

        x_input = x_input.reshape((1, n_steps, n_features))

        yhat = model.predict(x_input, verbose=0)

        temp_input.append(yhat[0][0])
        temp_input=temp_input[1:]

        lst_output.append(yhat[0][0])
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps, n_features))
        yhat = model.predict(x_input, verbose=0)

        temp_input.append(yhat[0][0])
        lst_output.append(yhat[0][0])
        i=i+1
print(lst_output)
```

As I plot the prediction graph, it showed that the prediction is not very good in terms of the quality because we can see clearly that there is an obvious gap between the prediction and actual value. There is definitely a couple of reasons to explain this situation. First of all, maybe I did not train for enough time therefore the model is not able to capture the underline patterns. Secondly, it is also possible that the time series itself involve high non-linearity and high complex. Finally, the time series is possible dependent on other various that is not involve in my modelling.



With this consideration, I ran a linear regression with power generation on the price. I also ran linear regression with different lags, and found interesting patterns.

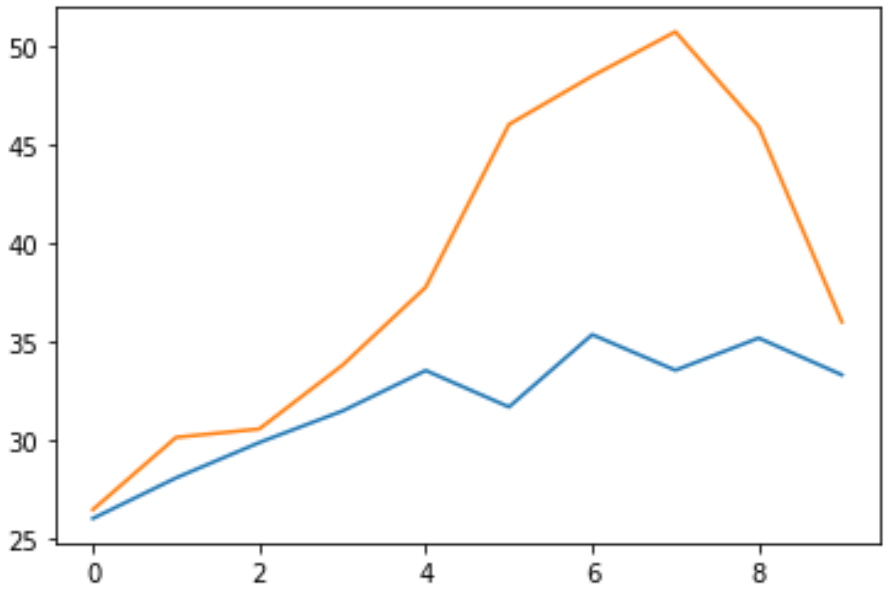


This tells us that the power generation has some effect on the price with a periodicity around 20 mins. With that said, I used LSTM to make prediction again but this time, I do it by conditioning on the features. I achieve this specifically by the code below

```
timeseries_data = np.array(df_new['HB_NORTH (RTLMP)'])[:14000]
n_steps = 10
X, y = prepare_data(timeseries_data, n_steps)
cc = np.array(df_new.loc[:,['ERCOT (WIND_RTI)', 'ERCOT (GENERATION_SOLAR_RT)','ERCOT (
    RTLOAD)']][10:14000])
X = np.hstack((X,cc))
X = X.reshape((X.shape[0], X.shape[1], n_features))
```

I trained my model for 100 epochs with 2 layers of LSTM cell and make prediction/

```
Epoch 90/100
13990/13990 [==============================] - 20s 1ms/sample - loss: 1529.6086
Epoch 91/100
13990/13990 [==============================] - 20s 1ms/sample - loss: 1724.1131
Epoch 92/100
13990/13990 [==============================] - 21s 1ms/sample - loss: 1891.6443
Epoch 93/100
13990/13990 [==============================] - 20s 1ms/sample - loss: 1868.3555
Epoch 94/100
13990/13990 [==============================] - 21s 2ms/sample - loss: 1838.2065
Epoch 95/100
13990/13990 [==============================] - 22s 2ms/sample - loss: 1799.1193
Epoch 96/100
13990/13990 [==============================] - 20s 1ms/sample - loss: 1747.8969
Epoch 97/100
13990/13990 [==============================] - 20s 1ms/sample - loss: 1743.1698
Epoch 98/100
```

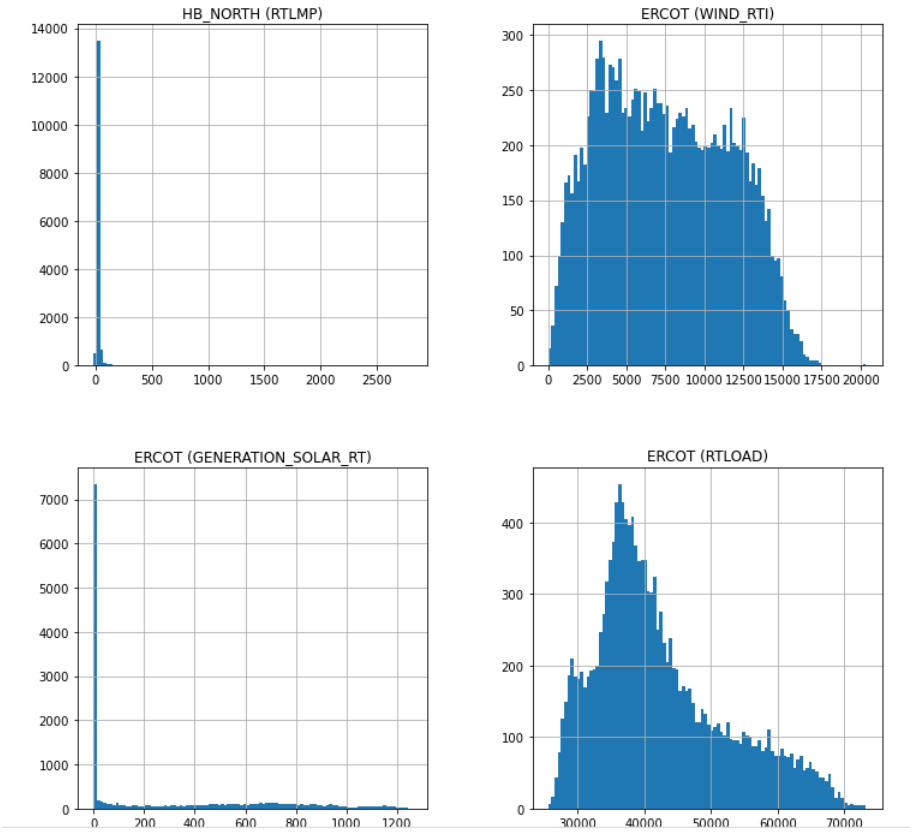


This result actually confirms our initial assumption that the power generation has effect on the price, and this time, we are able to make better prediction in terms of quality.

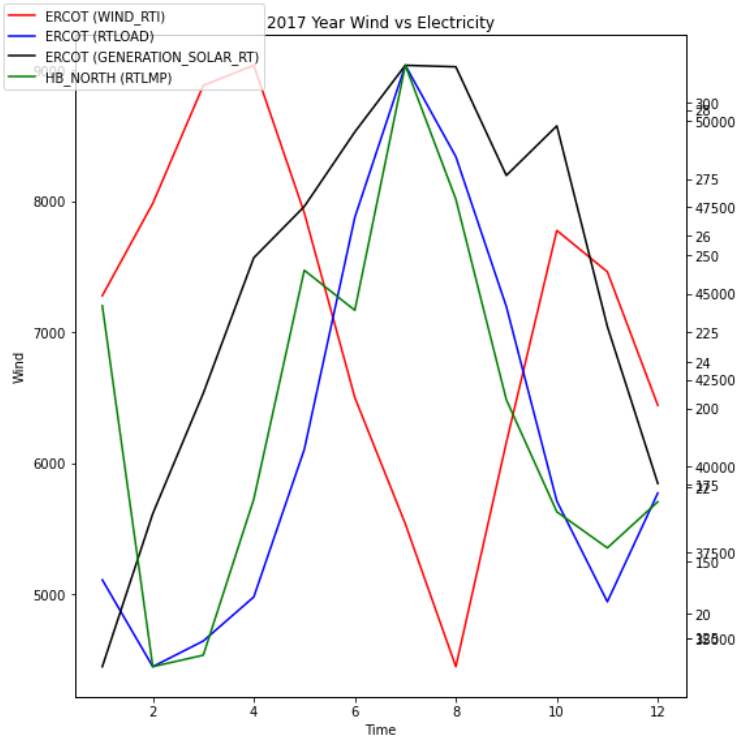
The first table below is a statistical summary about our data.

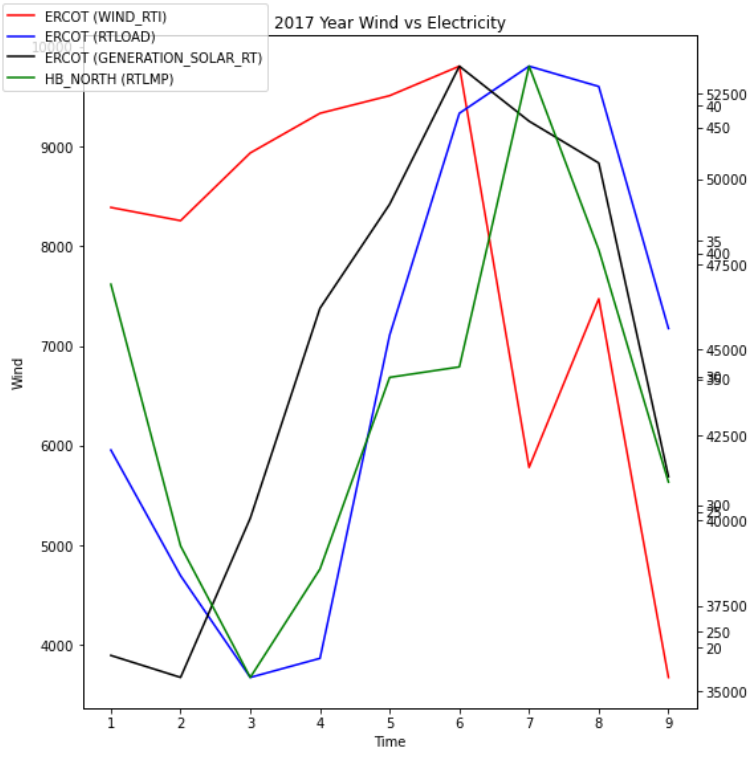
	HB_NORTH (RTLMP)	ERCOT (WIND_RTI)	ERCOT (GENERATION_SOLAR_RT)	ERCOT (RTLOAD)
count	14987.000000	14987.000000	14987.000000	14987.000000
mean	25.766417	7529.923293	291.911783	42371.673703
std	46.361945	3994.588069	370.895763	9874.339631
min	-17.860000	0.000000	0.000000	25566.511248
25%	18.041250	4133.925000	0.000000	35431.636526
50%	20.057500	7278.960000	21.930000	39934.007113
75%	25.030000	10851.280000	608.580000	47873.100786
max	2809.357500	20350.400000	1257.540000	73264.662123

The graphs below tells us the distribution of our data, and we can observe that the price is very small generally except that there exists some spikes during the time period and same as solar power generation. The distribution wind power and electricity load looks more normal than the two distribution we just went over.

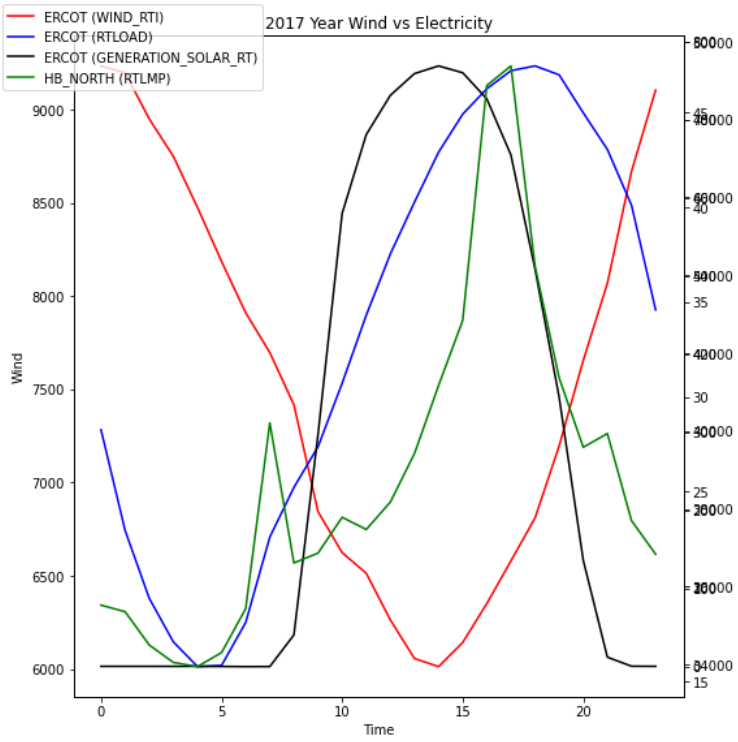


The graphs below tells us the time series of the average value of each time series for each month in the year of 2017 and 2018 respectively. As we can see from the figure, that power load is positively correlated with the price and it also makes sense because when we need more electricity and we have to load more power which also leads a higher price. Another observation we can make is that wind generation is negatively correlated with electricity load.





A similar observation can also be made if the time series is for each hour.


ERCOT (WIND_RTI)
ERCOT (RTLOAD)
ERCOT (GENERATION_SOLAR_RT)
HB_NORTH (RTLMP)
2017 Year Wind vs Electricity
Wind
Time
9000
8500
8000
7500
7000
6500
6000
0
5
10
15
20
60000
50000
40000
30000
25
20
15