Assignment 3 - Report

Shuxiang Sui

Exploratory data analysis:

For Assignment 3, I first spending a lot of time data studying:

The dataset has 14987 records and it starts from 2017-01-01 to 2018-09-17.

There are few missing data and outliers existed in this data, while some columns of data are in different formats, such as DATETIME, INT, float, and String.

```
In [3]: df.sort_values(by="HB_NORTH (RTLMP)", ascending=False)
Out[3]:
```

	DATETIME	HB_NORTH (RTLMP)	ERCOT (WIND_RTI)	ERCOT (GENERATION_SOLAR_RT)	ERCOT (RTLOAD)	HOURENDING	MARKETDAY	PEAKTYPE	MONTH	YEAR
9294	2018-01-23 07:00:00	2809.3575	2826.54	0.82	45679.060018	7	2018-01-23	WDPEAK	JANUARY	2018
12494	2018-06-05 16:00:00	2010.4625	2551.21	1204.49	65851.808693	16	2018-06-05	WDPEAK	JUNE	2018
13767	2018-07-28 17:00:00	1350.1875	3091.11	1183.07	67583.374865	17	2018-07-28	WEPEAK	JULY	2018
13766	2018-07-28 16:00:00	1248.5400	2779.95	1218.68	66711.027373	16	2018-07-28	WEPEAK	JULY	2018
14271	2018-08-18 17:00:00	1105.3150	4819.98	823.82	68399.665740	17	2018-08-18	WEPEAK	AUGUST	2018

Here are some superficial study of EDA:

1. Dataset:

```
In [7]: df.YEAR.value_counts()
Out[7]: 2017    8757
        2018    6224
        Name: YEAR, dtype: int64
```

There are more records in 2017 than in 2018.

```
In [8]: df.PEAKTYPE.value_counts()
Out[8]: WDPEAK     6964
        OFFPEAK    4993
        WEPEAK     3024
        Name: PEAKTYPE, dtype: int64
```

Weekday Peak

> Off Peak > Weekend Peek

```
In [9]: df.MONTH.value_counts()
Out[9]: JANUARY      1488
        MAY          1488
        JULY         1488
        AUGUST       1488
        MARCH        1485
        APRIL        1440
        JUNE         1440
        FEBRUARY     1344
        SEPTEMBER    1114
        OCTOBER       744
        DECEMBER      744
        NOVEMBER      718
        Name: MONTH, dtype: int64
```

The data stopped at September 2018, therefore, there are less data for October, November, and December.

Now, let's do a group by comparison:

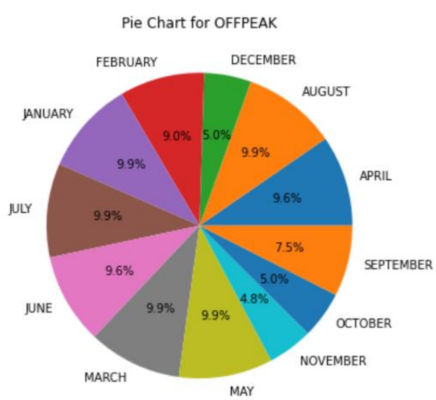
```
In [10]: df.groupby(['YEAR','PEAKTYPE']).size()

Out[10]: YEAR  PEAKTYPE
         2017  OFFPEAK     2917
               WDPEAK      4064
               WEPEAK      1776
         2018  OFFPEAK     2076
               WDPEAK      2900
               WEPEAK      1248
         dtype: int64
```

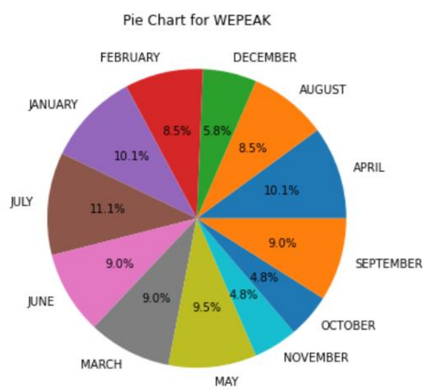
In 2017, there are almost 50% of records are Weekpeak, this trend keeps on in 2018.

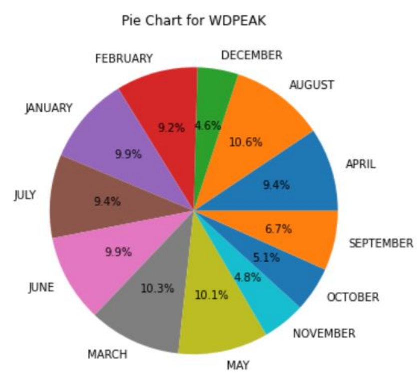
PEAKTYPE	MONTH	
OFFPEAK	APRIL	480
	AUGUST	496
	DECEMBER	248
	FEBRUARY	448
	JANUARY	496
	JULY	496
	JUNE	480
	MARCH	493
	MAY	496
	NOVEMBER	238
	OCTOBER	248
	SEPTEMBER	374
WDPEAK	APRIL	656
	AUGUST	736
	DECEMBER	320
	FEBRUARY	640
	JANUARY	688
	JULY	656
	JUNE	688
	MARCH	720
	MAY	704
	NOVEMBER	336
	OCTOBER	352
	SEPTEMBER	468
WEPEAK	APRIL	304
	AUGUST	256
	DECEMBER	176
	FEBRUARY	256
	JANUARY	304
	JULY	336
	JUNE	272
	MARCH	272
	MAY	288
	NOVEMBER	144
	OCTOBER	144
	SEPTEMBER	272

Here is the groupby for Peaks and Month, and the following are visualizations:

```
# create 3 pie chart for 3 peaktypes:
peaktypes = df['PEAKTYPE'].unique()

for peaktype in peaktypes:
    # filter data base on demand
    filtered_data = grouped_data.loc[peaktype]

    # draw
    plt.figure(figsize=(6, 6))  # 6*6
    plt.pie(filtered_data, labels=filtered_data.index, autopct='%1.1f%%')
    plt.title(f'Pie Chart for {peaktype}')
    plt.show()
```







We could observe a very clear trend for each month and their ratio to the year.

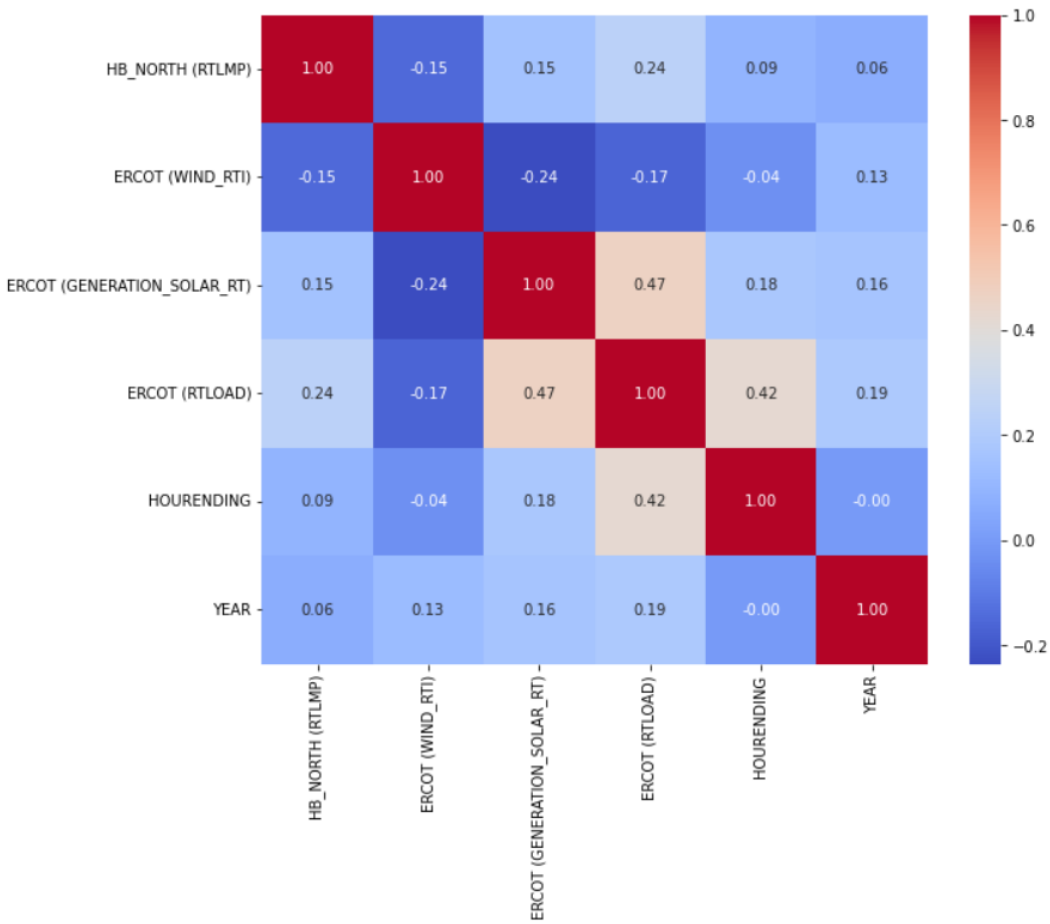
For example, the Weekend Real-time price "RTLMP" is higher in **July** and S**eptember** than any other month.

And it is also necessary to conduct a correlation analysis between variables.

In [14]:
```python
# Graph correlation between time series
corr_vars = ['HB_NORTH (RTLMP)', 'ERCOT (WIND_RTI)', 'ERCOT (GENERATION_SOLAR_RT)', 'ERCOT (RTLOAD)']
df[corr_vars].corr()
```

Out[14]:

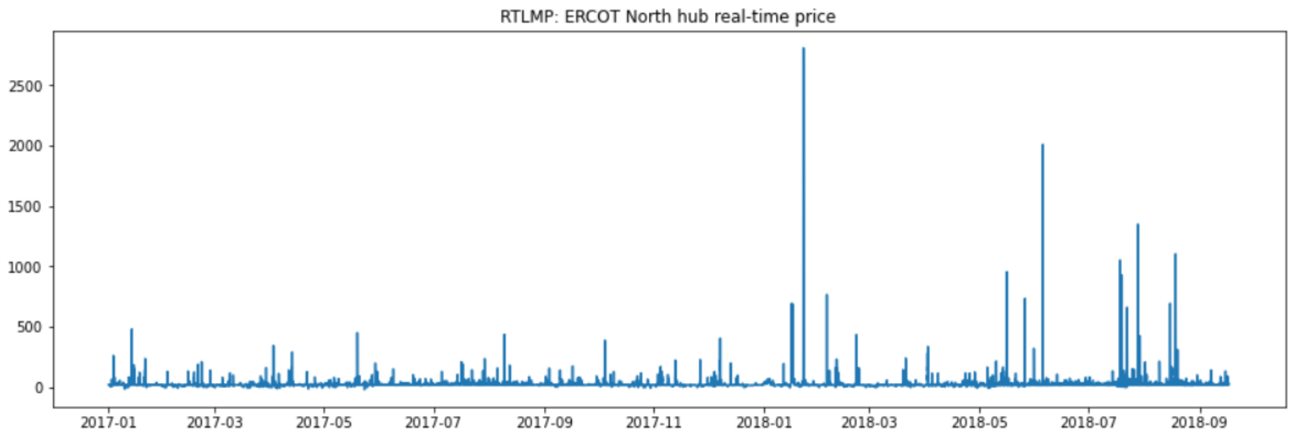
	HB_NORTH (RTLMP)	ERCOT (WIND_RTI)	ERCOT (GENERATION_SOLAR_RT)	ERCOT (RTLOAD)
HB_NORTH (RTLMP)	1.000000	-0.151156	0.151910	0.238481
ERCOT (WIND_RTI)	-0.151156	1.000000	-0.235325	-0.166710
ERCOT (GENERATION_SOLAR_RT)	0.151910	-0.235325	1.000000	0.466309
ERCOT (RTLOAD)	0.238481	-0.166710	0.466309	1.000000

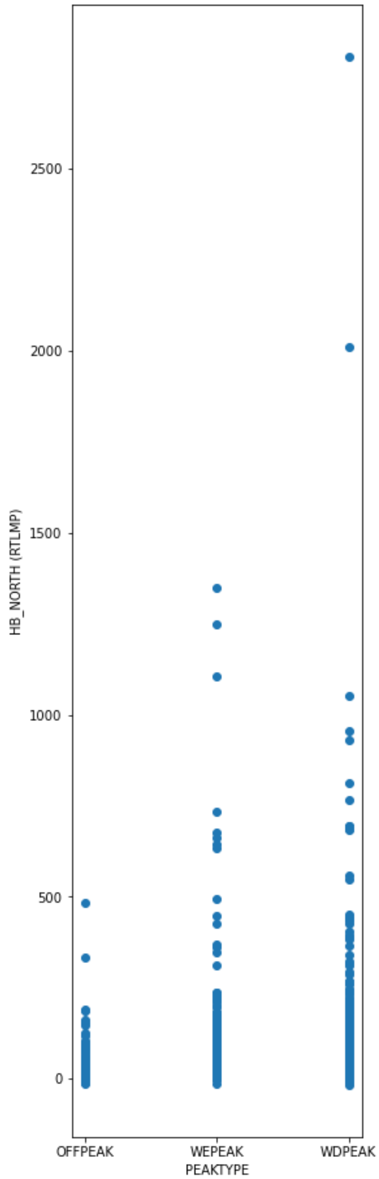


As we could observe from the map, the Strongest Correlation is 0.47, between ERCOT (GENERATION_SOLAR_RT) and ERCOT (RTLOAD).

The Second Strong correlation is 0.42, between Hour Ending and ERCOT (RTLOAD).

From the historical RTLMP price trend graph, we could see that there are many outliers throughout time, and the highest one even breaks the 2500.

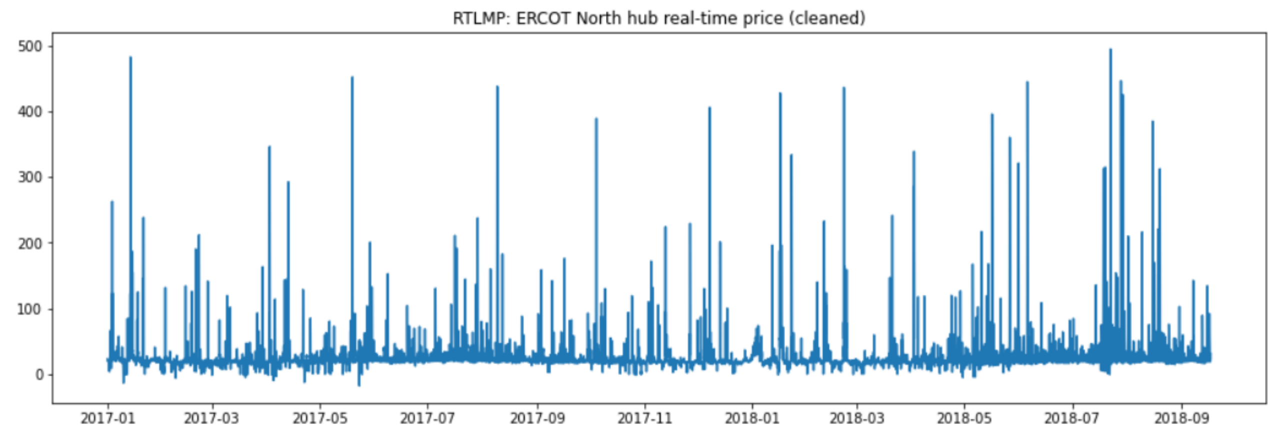




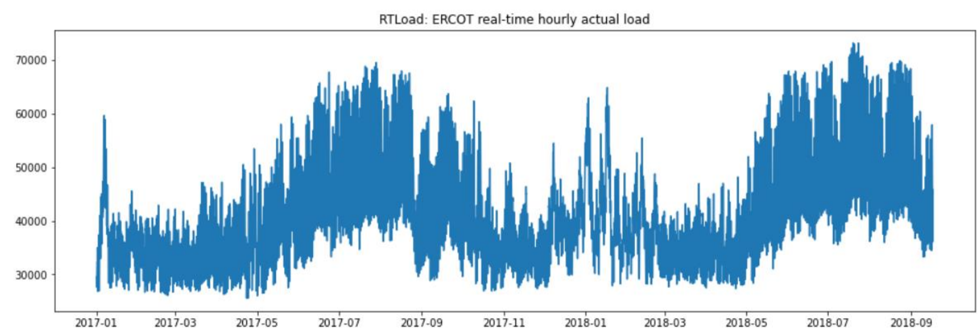
	DATETIME	HB_NORTH (RTLMP)	ERCOT (WIND_RTI)	ERCOT (GENERATION_SOLAR_RT)	ERCOT (RTLOAD)
13623	2018-07-22 17:00:00	494.7975	3019.56	1133.62	70945.990785
312	2017-01-14 01:00:00	482.9050	3393.45	0.11	33336.327325
3326	2017-05-19 16:00:00	452.2725	6048.78	658.84	53999.100555
13765	2018-07-28 15:00:00	446.6000	2487.60	1210.72	65040.478428
12493	2018-06-05 15:00:00	444.9850	2178.80	1218.70	63904.550605
...	...	...	...	...	...
2255	2017-04-05 01:00:00	-9.6625	11972.23	0.00	32575.858775
218	2017-01-10 03:00:00	-11.4800	11570.82	0.11	28449.908768
2673	2017-04-22 11:00:00	-12.5300	11554.00	187.24	35664.608470
217	2017-01-10 02:00:00	-13.6075	11832.67	0.11	28994.028000
3421	2017-05-23 15:00:00	-17.8600	10029.13	290.82	45378.964325

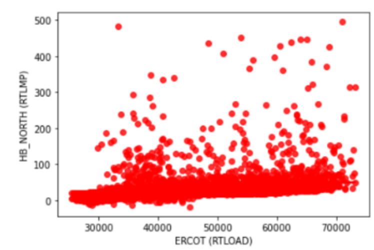
14960 rows × 10 columns

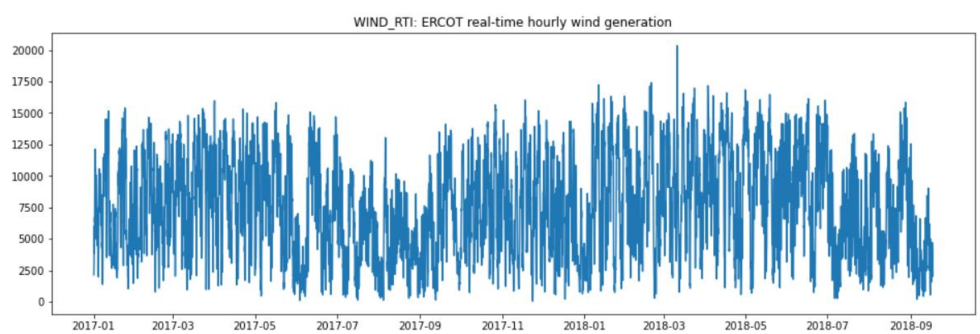
After dropping the outliers that above 500, we could observe a relatively fluctuating but normal distribution for RTLMP.

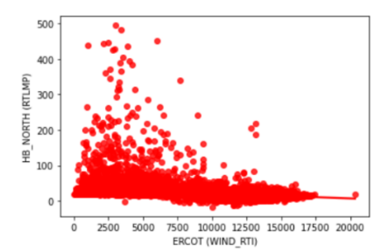


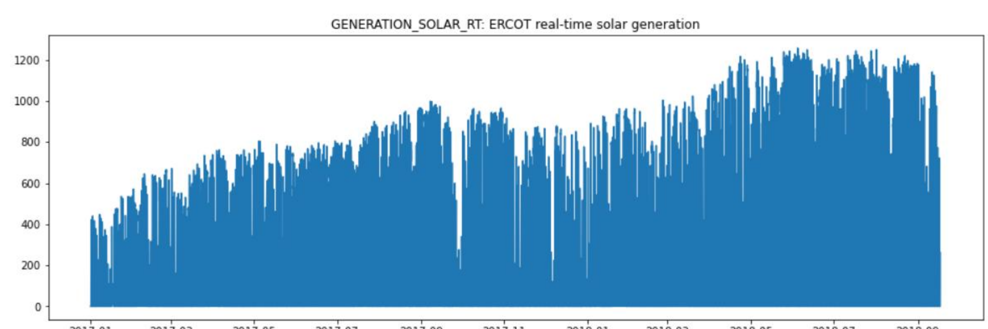
By this chance, the other trend graph are listed in the following:

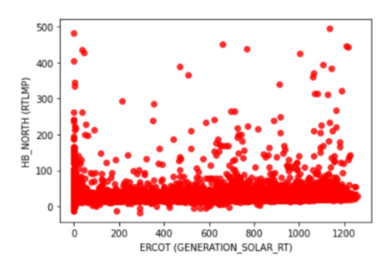








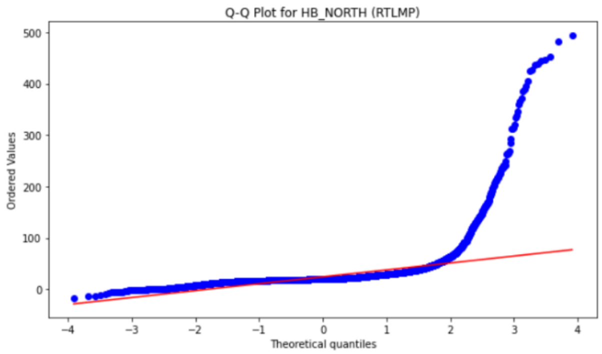


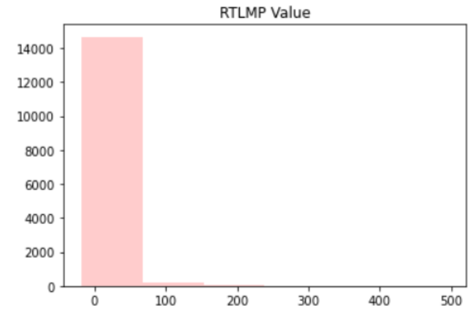


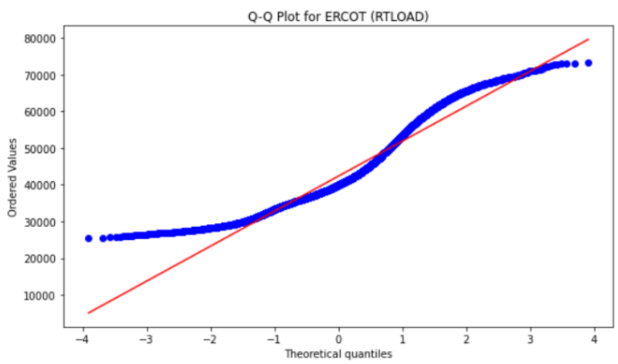
There are some obvious **seasonal patterns** that indicate the price and generation peek are usually in the summer and decline by the winter.

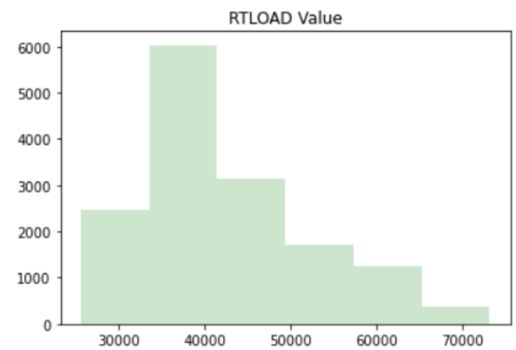
Besides, from the QQ plot and historical graph, it seems that it is not normally distributed data.

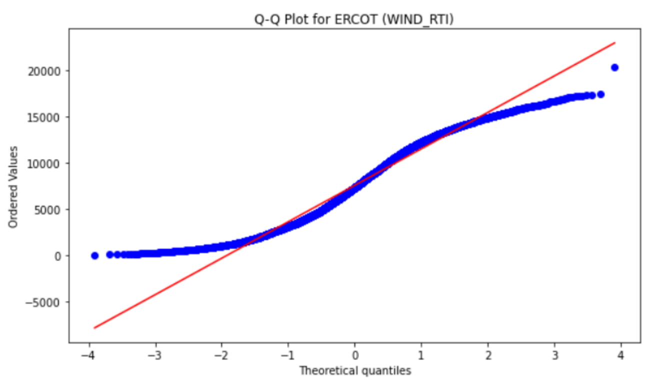
```python
#histogram
plt.hist(df['HB_NORTH (RTLMP)'], bins=6, alpha=0.2, color='red')

plt.title("RTLMP Value")

plt.show()
```

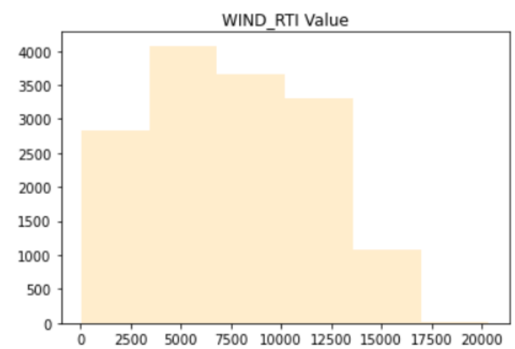


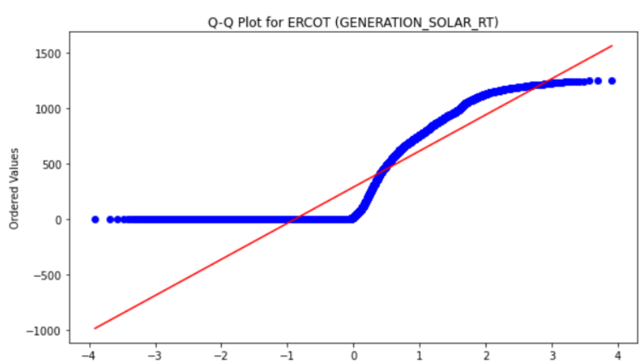


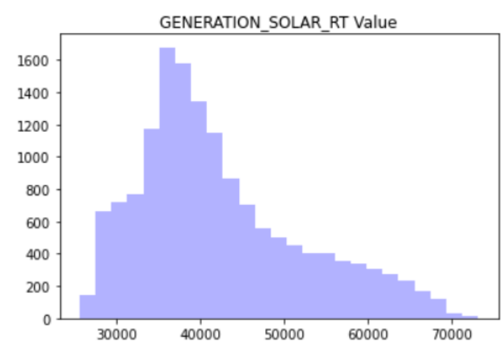












```
ADF Statistic: -16.383217491416932
p-value: 2.738741383731422e-29
Critical Values:
	1%: -3.4307881607085093
	5%: -2.861733648008762
	10%: -2.566873074173142
```

```
ADF Statistic: -5.135010613400731
p-value: 1.19159087346091e-05
Critical Values:
	1%: -3.4307884544915375
	5%: -2.861733777838518
	10%: -2.566873143280059
```

```
ADF Statistic: -11.37478614270305
p-value: 8.799833858895371e-21
Critical Values:
	1%: -3.4307884544915375
	5%: -2.861733777838518
	10%: -2.566873143280059
```

```
ADF Statistic: -7.635977820138659
p-value: 1.9519948521410252e-11
Critical Values:
	1%: -3.4307884544915375
	5%: -2.861733777838518
	10%: -2.566873143280059
```

```python
import statsmodels
#note: I found out that only numpy 1.23 support statsmodels
from statsmodels.tsa.stattools import adfuller

#took only the timeseries data
ts_RTLMP = df['HB_NORTH (RTLMP)']
ts_RTLOAD = df['ERCOT (RTLOAD)']
ts_WIND_RTI = df['ERCOT (WIND_RTI)']
ts_SOLAR_RT = df['ERCOT (GENERATION_SOLAR_RT)']

#def adf test function to test the stationarity.

def adf_test(data):
    result = adfuller(data)
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t{}: {}'.format(key, value))

# test
adf_test(ts_RTLMP)
```

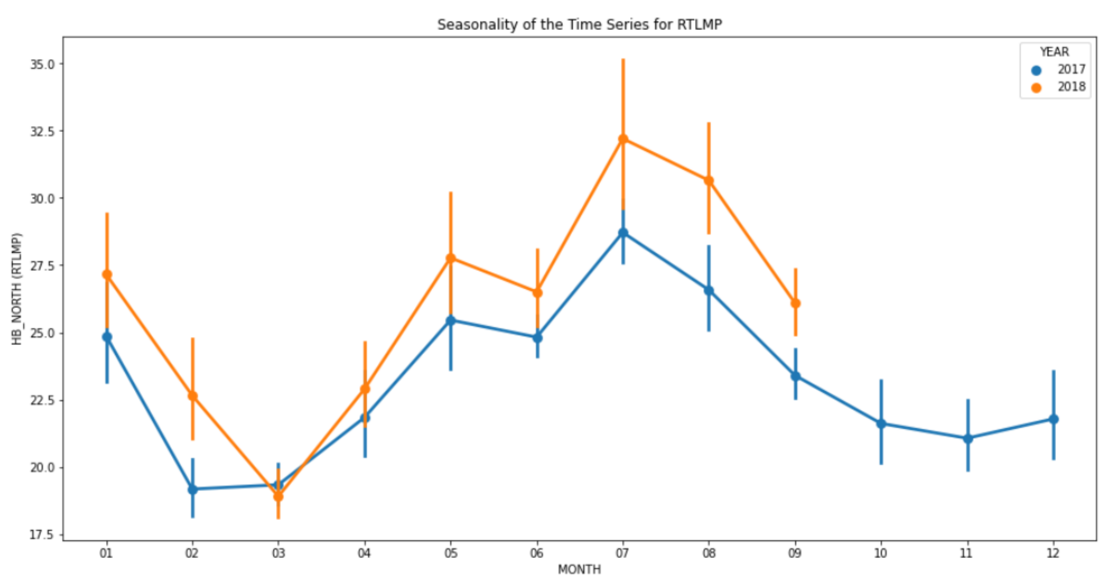
Therefore, I take advantage of Adfuller to test the the hypothesis and check the stationarity.

All of the tested variables are stationary.

After confirming the stationarity, I selected the variable and conduct the **seasonal trend visualization**:

```python
df_ts = df.copy()
columns_to_drop = ["DATETIME", "HOURENDING", "MARKETDAY"]
df_ts = df_ts.drop(columns=columns_to_drop)
df_ts

plt.figure(figsize=(16,8))
plt.title('Seasonality of the Time Series for RTLMP')
sns.pointplot(x='MONTH',y='HB_NORTH (RTLMP)',hue='YEAR',data=df_ts)
```

```
3]: <AxesSubplot:title={'center':'Seasonality of the Time Series for RTLMP'}, xlabel='MONTH', ylabel='HB_NORTH (RTLMP)'>
```



2. Prediction and analysis

Rename the data and change the DateTime to continuous timestamp data:

```
df.rename(columns={'HB_NORTH (RTLMP)': 'RTLMP',
                   'ERCOT (WIND_RTI)': 'WIND_RTI',
                   'ERCOT (GENERATION_SOLAR_RT)': 'GENERATION_SOLAR_RT',
                   'ERCOT (RTLOAD)': 'RTLOAD'},  inplace=True)
df.head()
```

0]:

	DATETIME	RTLMP	WIND_RTI	GENERATION_SOLAR_RT	RTLOAD	HOURENDING	MARKETDAY	PEAKTYPE	MONTH	YEAR
0	2017-01-01 01:00:00	23.3575	2155.31	0.0	29485.791355	1	2017-01-01	OFFPEAK	01	2017
1	2017-01-01 02:00:00	21.4650	2313.81	0.0	28911.565913	2	2017-01-01	OFFPEAK	01	2017
2	2017-01-01 03:00:00	20.7350	2587.68	0.0	28238.258175	3	2017-01-01	OFFPEAK	01	2017
3	2017-01-01 04:00:00	20.2700	2748.65	0.0	27821.000513	4	2017-01-01	OFFPEAK	01	2017
4	2017-01-01 05:00:00	20.1200	2757.49	0.0	27646.942413	5	2017-01-01	OFFPEAK	01	2017

```
#turn datetime to a continuous variable:
df['Timestamp'] = (df['DATETIME'] - pd.Timestamp("2017-01-01 00:00:00")) // pd.Timedelta('1h')
df.drop(['DATETIME', 'MARKETDAY'], axis=1, inplace=True)
df.tail(5)
```

41]:

	RTLMP	WIND_RTI	GENERATION_SOLAR_RT	RTLOAD	HOURENDING	PEAKTYPE	MONTH	YEAR	Timestamp
14980	20.3950	3094.87	0.00	37923.34	6	OFFPEAK	09	2018	14982
14981	20.8800	3325.27	0.00	40936.18	7	WDPEAK	09	2018	14983
14982	20.8600	3195.52	2.04	41902.24	8	WDPEAK	09	2018	14984
14983	22.7675	2605.50	111.59	43014.37	9	WDPEAK	09	2018	14985
14984	31.0600	2034.80	261.65	45782.55	10	WDPEAK	09	2018	14986

Conduct a regression analysis while making Peaktype and Month as dummies:

```
lm_hr = smf.ols(formula = 'RTLMP ~ Timestamp + WIND_RTI + GENERATION_SOLAR_RT + RTLOAD + PEAKTYPE + MONTH + HOURENDING'
print (lm_hr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  RTLMP   R-squared:                       0.206
Model:                            OLS   Adj. R-squared:                  0.205
Method:                 Least Squares   F-statistic:                     215.1
Date:                Tue, 23 May 2023   Prob (F-statistic):               0.00
Time:                        01:10:01   Log-Likelihood:                -66271.
No. Observations:               14960   AIC:                         1.326e+05
Df Residuals:                   14941   BIC:                         1.327e+05
Df Model:                          18
Covariance Type:            nonrobust
```

```
======================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
Intercept            -7.6103      1.147     -6.633      0.000      -9.859      -5.361
PEAKTYPE[T.WDPEAK]   -2.3391      0.503     -4.651      0.000      -3.325      -1.353
PEAKTYPE[T.WEPEAK]   -0.2404      0.561     -0.428      0.669      -1.341       0.860
MONTH[T.02]          -1.0150      0.773     -1.313      0.189      -2.530       0.501
MONTH[T.03]          -0.3142      0.767     -0.410      0.682      -1.818       1.189
MONTH[T.04]           2.1392      0.776      2.758      0.006       0.619       3.660
MONTH[T.05]          -2.5552      0.761     -3.356      0.001      -4.048      -1.063
MONTH[T.06]         -12.2775      0.811    -15.135      0.000     -13.868     -10.687
MONTH[T.07]         -14.4562      0.854    -16.931      0.000     -16.130     -12.783
MONTH[T.08]         -13.8663      0.837    -16.576      0.000     -15.506     -12.227
MONTH[T.09]         -12.1868      0.850    -14.339      0.000     -13.853     -10.521
MONTH[T.10]          -4.0373      0.920     -4.390      0.000      -5.840      -2.234
MONTH[T.11]          -1.5428      0.941     -1.639      0.101      -3.387       0.302
MONTH[T.12]          -6.2140      0.930     -6.681      0.000      -8.037      -4.391
Timestamp           1.29e-05   4.31e-05      0.299      0.765   -7.15e-05    9.73e-05
WIND_RTI             -0.0014   4.64e-05    -29.820      0.000      -0.001      -0.001
GENERATION_SOLAR_RT  -0.0005      0.001     -0.769      0.442      -0.002       0.001
RTLOAD                0.0012    3.1e-05     39.189      0.000       0.001       0.001
HOURENDING           -0.1726      0.030     -5.761      0.000      -0.231      -0.114
==============================================================================
Omnibus:                24820.299   Durbin-Watson:                   1.130
Prob(Omnibus):              0.000   Jarque-Bera (JB):         18724449.047
Skew:                      11.282   Prob(JB):                         0.00
Kurtosis:                 174.843   Cond. No.                     5.31e+05
==============================================================================
```

For most of created dummies, they are stastitically significant at $p< 0.05$.

From the Linear Model result, we could understand that the RTLMP on Weekday will be 2.33 USD higher than Off Peak, and for June, July, August, and September, the price will be 12.28/14.45/13.86/12.18 higher than the omitted month January.

However, this also shows RTLMP doesn't have a linear Trend to these factors, and therefore I will move to other methods when making the predictions.

I think it is necessary to keep the Peektypes and therefore I recode the peektype:

```python
from sklearn import preprocessing

#replace the Peaktype by value
df['PEAKTYPE'] = df['PEAKTYPE'].replace({'OFFPEAK': 0, 'WEPEAK': 1, 'WDPEAK': 2})
# rescale the data for the better predictions:
drop_list=["RTLMP", "HOURENDING", "MONTH", "YEAR"]

#spliting data prepare
y = pd.DataFrame(df["RTLMP"])
X = df.drop([col for col in drop_list if col in df], axis=1)
```

From previous OLS model, the RTLMP for Weekday Peak is highest, and then for the Weekend Peak and Off Peak, therefore I recode them to 2, 1, and 0.

```python
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
```

I split the train and test data with 80/20 while dropping any unwanted data from the drop list.

```python
def split_train_test(X, y, test_size=0.2):
    # Determine the index at which to split the data
    split_index = int(len(X) * (1 - test_size))

    # Split the data into training and test sets
    X_train, X_test = X[:split_index], X[split_index:]
    y_train, y_test = y[:split_index], y[split_index:]

    return X_train, X_test, y_train, y_test

# Split the X and y data into train and test sets
X_train, X_test, y_train, y_test = split_train_test(X, y, test_size=0.2)

# Print the shapes of the resulting datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (11968, 5)
X_test shape: (2992, 5)
y_train shape: (11968, 1)
y_test shape: (2992, 1)
```

Then, I scaled the data to make them more continuous and it will be easier to discover the trend.

And then, I developed 2 methods in fitting and predicting these data:

First, using a neural network model:

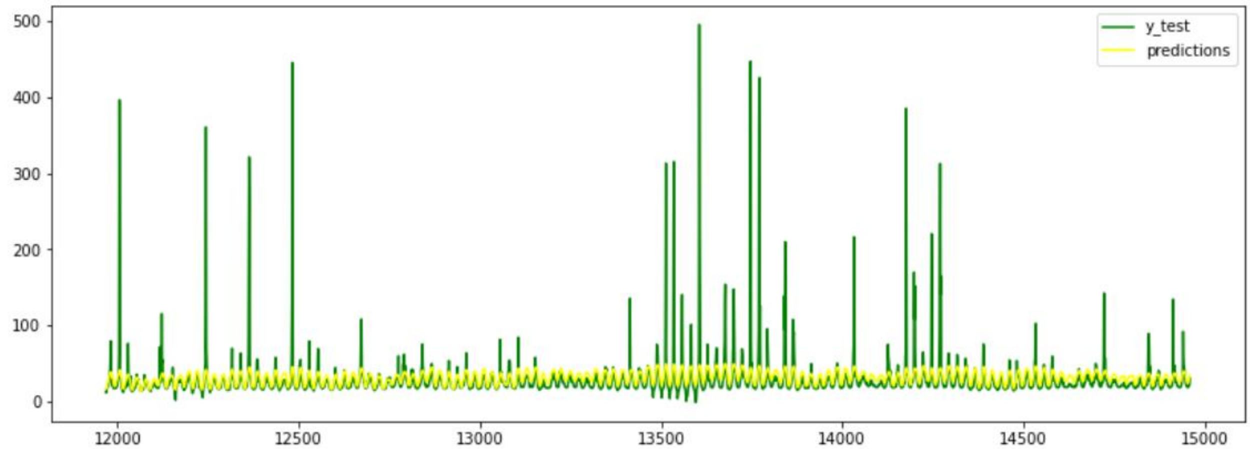
This model applied 1 input layer and 2 hidden layers while setting the learning rate to 0.01.

```
Epoch 1/10
374/374 [==============================] - 2s 5ms/step - loss: 8.1186 - accuracy: 0.1292
Epoch 2/10
374/374 [==============================] - 2s 4ms/step - loss: 3.9793 - accuracy: 0.1451
Epoch 3/10
374/374 [==============================] - 2s 4ms/step - loss: 3.3130 - accuracy: 0.1591
Epoch 4/10
374/374 [==============================] - 2s 4ms/step - loss: 3.1218 - accuracy: 0.1704
Epoch 5/10
374/374 [==============================] - 2s 4ms/step - loss: 2.9897 - accuracy: 0.1827
Epoch 6/10
374/374 [==============================] - 2s 4ms/step - loss: 2.8943 - accuracy: 0.1930
Epoch 7/10
374/374 [==============================] - 2s 4ms/step - loss: 2.8262 - accuracy: 0.1957
Epoch 8/10
374/374 [==============================] - 2s 4ms/step - loss: 2.7730 - accuracy: 0.1979
Epoch 9/10
374/374 [==============================] - 2s 4ms/step - loss: 2.7332 - accuracy: 0.1969
Epoch 10/10
374/374 [==============================] - 2s 4ms/step - loss: 2.6990 - accuracy: 0.2011
94/94 [==============================] - 0s 3ms/step - loss: 3.1088 - accuracy: 0.1237
[3.1087710857391357, 0.12366310507059097]
```

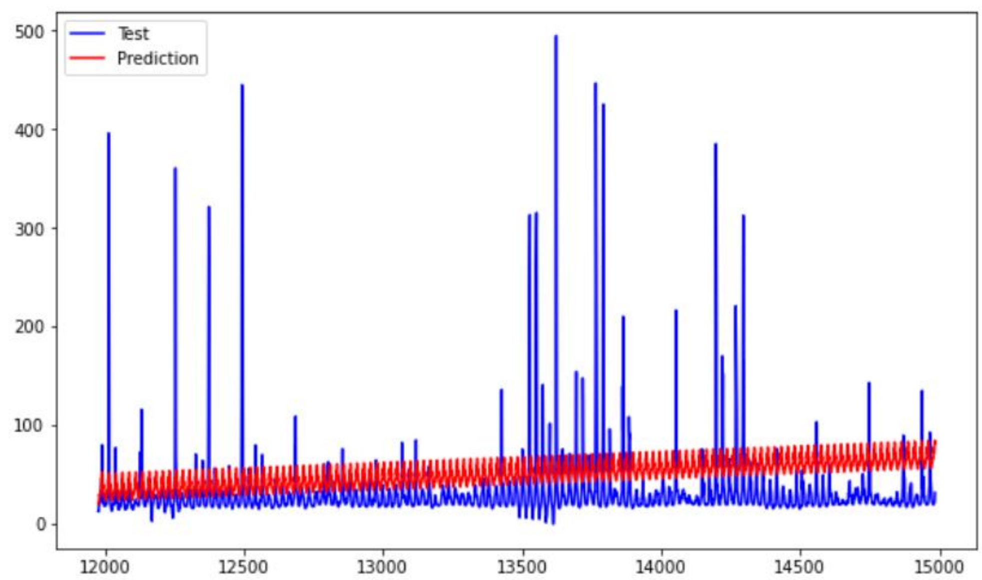
This gave us an output of 0.12366.

Now, let's apply this model to the prediction:

The model shows a rolling-matching trend, however, it didn't predict the price fluctuation timely, I believe this might be due to the reason that my neural network model wasn't optimized and self-adjusting, and I think it might work better if I add more layers and try to find a way to minimize the loss and increase accuracy.



I also attempted to use Arima in this prediction, however, it seems that the prediction are gradually over estimated and I think the reason might be found on the method that I treated the seasonal data, I think avoiding the +/- 1day between month would be helpful to solve this issue.



Dear Interviewer,

I am grateful for the opportunity you afforded me to showcase my skills through this assignment. Although I didn't fully complete Assignment 3 to my satisfaction, it served as a valuable learning experience. This was my initial venture into applying machine learning methods to time series data, and I've made significant strides throughout the process.

Despite the challenges, my confidence remains unshaken. I am keen to delve deeper into time series and quantitative data analysis during the upcoming summer. I believe these skills will enable me to excel not just in the field of data science, but also in Energy Economics.

As an Economics graduate with Distinction from Purdue University, I have developed a profound interest in Energy Economics. I yearn for an opportunity to bring to the fore my unique insights and skills in this field, leveraging my academic background and inherent passion.

In conclusion, I express my heartfelt appreciation for this opportunity. I eagerly anticipate our future interactions and the chance to discuss the potential synergy between my skill set and the needs of your esteemed organization.

Best Regards,

Shuxiang Sui