Guzman Energy 2020-21 Winter Quant Internship - Assignment

By Yunfei (Sophie) Chen

Assignment 1: Power Calendar function

The purpose of this function is to correctly calculate which hours belong to a block given the ISO and peak type, on a daily / monthly / quarterly / yearly basis.

Things to take care of:

- Western market takes Monday to Saturday as weekdays, while Eastern market takes Monday to Friday as weekdays.

- Only MISO does not have the DST setting. According to Wikipedia, “starts on the second Sunday in March and ends on the first Sunday in November, with the time changes taking place at 2:00 a.m. local time.” This means “onpeak” and “2x16H” types are totally not influenced by DST as they only cover HE7-HE22.

- NERC holidays: New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving, Christmas Day.

Documentation of functions (Helper functions):

1. get_NERC_calendar(y)

Returns a NERC holidays calendar (a dictionary with key as holiday date and value as holiday name) given a specific year. Uses Python “holidays” package to fetch holidays and corresponding dates (need to install the package in advance).

2. isDSTStartDate(date)

Boolean. Identifies if the input date is the second Sunday in March of that year. Returns True is yes, otherwise False.

3. isDSTEndDate(date)

Boolean. Identifies if the input date is the first Sunday in November of that year. Returns True is yes, otherwise False.

4. get_onpeak_hour(date, area)

Returns the number of onpeak type hours of the input date, in the input market (“eastern” / “western”). Two factors to determine: if the date is a weekday in the input market and if the date is not a NERC holiday.

5. get_2x16H_hour(date, area)

Returns the number of 2x16H type hours of the input date, in the input market (“eastern” / “western”). Two factors to determine: if the date is a weekend and if the date is an NERC holiday.

6. get_period_type(period)

Returns the period flag (“daily” / “monthly” / “quarterly” / “yearly”) given the period string input (eg, “2018-2-3” as a daily, “2018Mar” as a monthly, “2018Q2” as a quarterly, “2018A” as an annually). Gives a message 'period type error' if none of the four patterns are matched to the input and returns None.

7. get_start_end(period)

Returns the start date & end date of the period input, in datetime.datetime formats.

Use the #6 Helper function to try to identify the period type firstly. If the period type can be identified, and if it is “daily”, then return the date directly (start=end);

if it is “monthly”, since the length of each month in different years can be different, I use the function *pd.date_range* here, and postpone the end date to the first day of the next month, and remove this first day in the result. This solves the issue perfectly;

If it is “quarterly”, we know clearly which months correspond to which quarter, and the month length issue is solved in the same way as described above;

If it is “yearly”, the month length issue is solved in the same way.

Gives the message “'cannot get start and end dates'” if any error is detected and returns None.

8. get_hours(iso, peak_type, period)

Returns the number of hours of the input iso, the input peak type and the input period.

Uses #1-7 Helper functions. Firstly, get the start date & end date of the period, then compute the number of hours and adjust for DST.

Gives the message “'cannot find the iso'” if the input iso is not among the 7 major ISOs, and returns None. Gives the message “cannot find the peak type” if the input peak type is not among the 4 peak types and returns None. Errors related to period and time has been defined in previous functions.

Some validation of the correctness of the functions:

```
print(get_hours('PJMISO', 'offpeak', '2020Mar'))
print(get_hours('MISO', 'offpeak', '2020Mar'))
print(get_hours('CAISO', 'onpeak', '2020Oct'))
# Counterexamples
print('\n')
print(get_hours('CAI', 'onpeak', '2020Oct'))   # iso error
print(get_hours('CAISO', 'peak', '2020Oct'))   # peak type error
print(get_hours('CAISO', 'onpeak', '2020ABC'))   # period time error
```

```
('PJMISO', 'offpeak', Timestamp('2020-03-01 00:00:00', freq='D'), Timestamp('2020-03-31 00:00:00', freq='D'), 391)
('MISO', 'offpeak', Timestamp('2020-03-01 00:00:00', freq='D'), Timestamp('2020-03-31 00:00:00', freq='D'), 392)
('CAISO', 'onpeak', Timestamp('2020-10-01 00:00:00', freq='D'), Timestamp('2020-10-31 00:00:00', freq='D'), 432)


cannot find the iso
[None, None, None, None, None]
cannot find the peak type
[None, None, None, None, None]
period type error
cannot get start and end dates
[None, None, None, None, None]
```

1. Firstly, the counterexamples check for the iso error, the peak type error, and the period time error. They are returned with None and the message describing the cause of the error. It works.

2. For those inputs that work well, check their results with the reference source: https://www.energygps.com/HomeTools/PowerCalendar.

- Eg1: PJMISO, offpeak, “2020Mar”. My function returns 391. Correct.

PJM Calendar - March 2020 to December 2022

Monthly Hours

Month	Year	Flat	Peak	Off-Peak	1x16H	Peak Days	Total Days	Holidays
03 - Mar	2020	743	352	391	80	22	31	0

NERC Holidays

Date	Holiday
Mon 5/25/2020	Memorial
Sat 7/4/2020	Independence

- Eg2: MISO, offpeak, “2020Mar”. My function returns 392.

The ref source does not provide the option of MISO, but we know MISO does not have the DST setting, and it does not deduct one hour at 2am on the second Sunday in March. Therefore, MISO should have one more offpeak hour in March than PJMISO, considering both ISOs are in the eastern market. So correct.

- Eg3: CAISO, onpeak, 2020Oct. My function returns 432. Correct.

CAISO Calendar - October 2020 to December 2022

Monthly Hours

Month	Year	Flat	Peak	Off-Peak	1x16H	Peak Days	Total Days	Holidays
10 - Oct	2020	744	432	312	64	27	31	0

NERC Holidays

Date	Holiday
Thu 11/26/2020	Thanksgiving
Fri 12/25/2020	Christmas

Assignment 2: Meter Data formatting

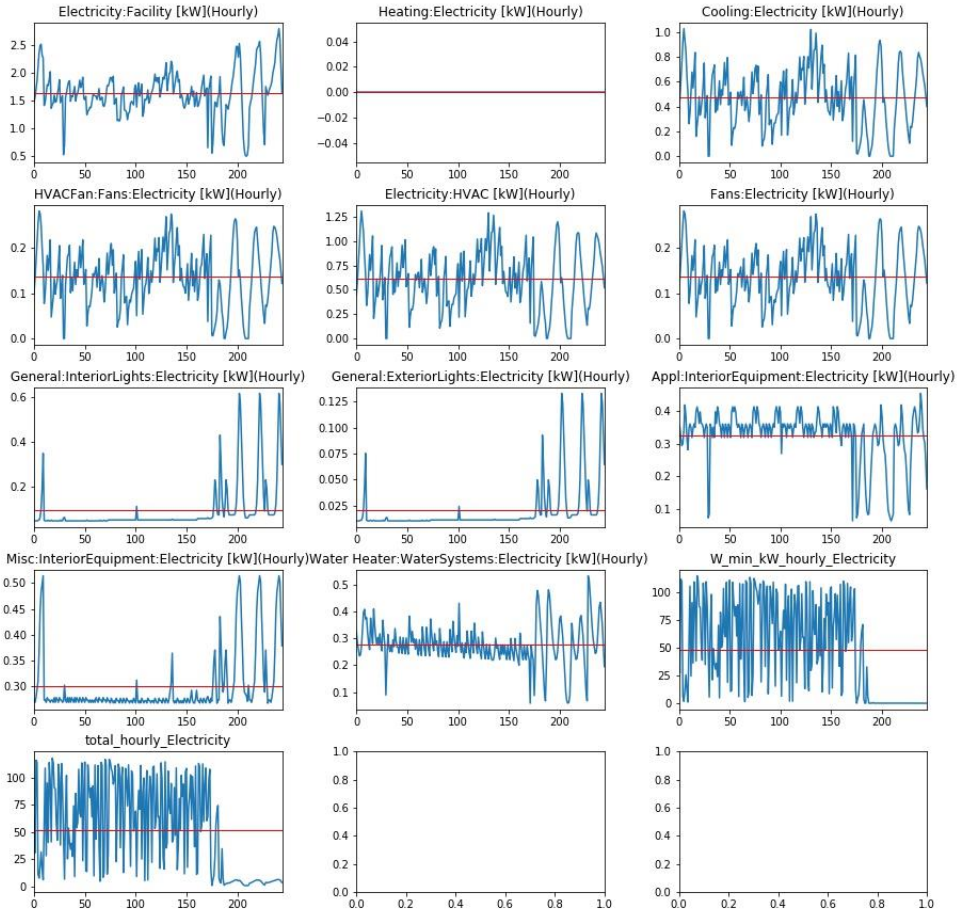
- To fit into the *dfply* package, I use the decorator function *@dfpipe* to turn some pandas methods to pipe methods for simplicity.

- The unified unit of electricity is KW, so data in *new.app4.csv* has been changed to KW expression.

- Data in *USA_AL_Auburn-Opelika.AP.722284_TMY3_BASE.csv* contains the consumption of other types of energies (gas), which has been removed from the calculation.

Data Plotting & Anomaly & Pattern

1. By hour



- *Finding 1:*

Before Aug 2013, the data of *Electricity:Facility*, *Cooling:Electricity*, *HVACFan:Fans:Electricity*, *Electricity:HVAC*, and *Fans:Electricity* is volatile and fluctuates hour by hour. However, after Aug 2013, their data fluctuates much less and shows strong monthly seasonality. These seasonal patterns demonstrate a higher peak than historically, and a lower trough than historically. This seems to be abnormal.

- *Finding 2:*

The *Heating* appliance never used any electricity for the time period spanning from June 7 to September 17, 2013.

- *Finding 3:*

General:InteriorLights and *General:ExteriorLights* rarely consumed electricity before Aug 2013, but after that, these two appliances started to consume electricity and their consumption patterns show strong monthly seasonality.

- *Finding 4:*

Appl:InteriorEquipment, Misc:InteriorEquipment, Water Heater:WaterSystems data show stable hourly fluctuations within a fixed range of value respectively before Aug 2013, but is followed by a strong monthly seasonality afterwards.

- *Finding 5:*

W_min_kW_hourly_Electricity refers to the extra appliance's electricity consumption. Before Aug 2013, the appliance's consumption fluctuates hourly in the range of 0 to 115 kW. However, its consumption rapidly declined to 0 afterwards.

Also, the extra appliance's electricity consumption is much larger than that of all the other appliances.

2. By weekday

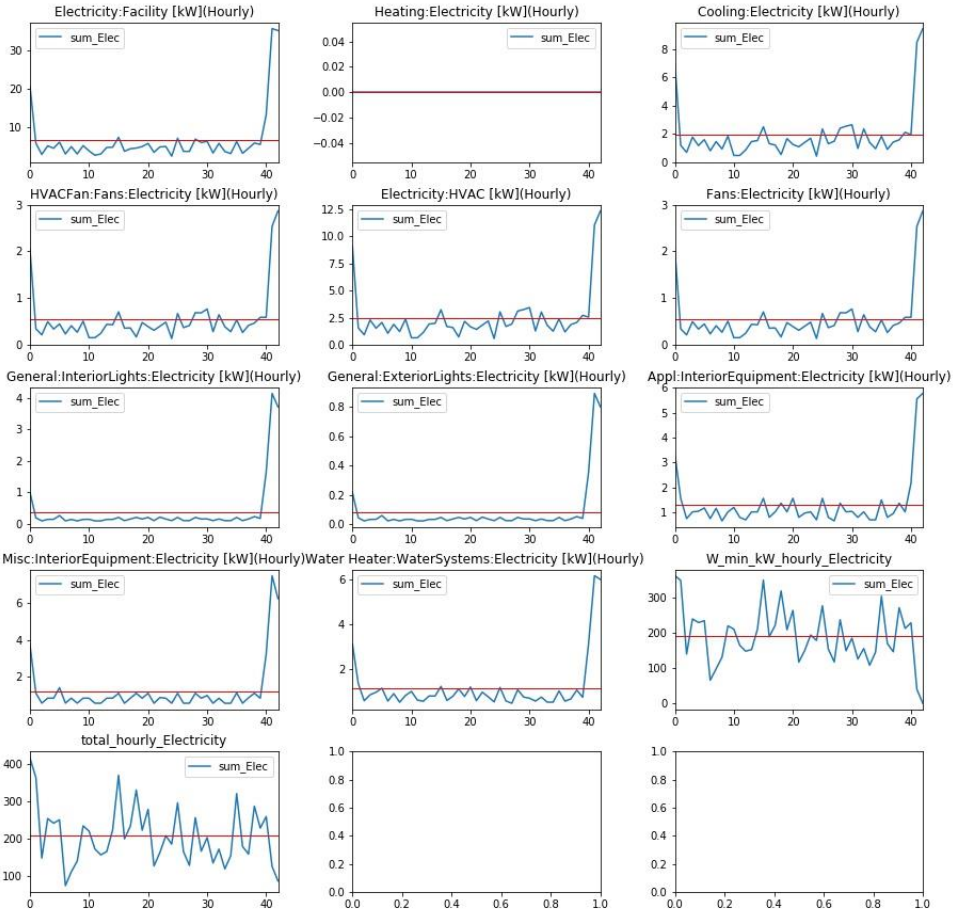
The hourly data is aggregated to daily basis.

- *Finding 1:*

Consistent with the previous findings, the data of all appliances except the extra appliance show a sudden rise in electricity consumption after Aug 2013, due to the abnormal, new strong monthly pattern.

- *Finding 2:*

The extra appliance sees a sharp drop to 0 after Aug, consistent with the previous finding as well. The average daily consumption of the appliance is in between 1.232 and 362.48 kW before it declines to 0.



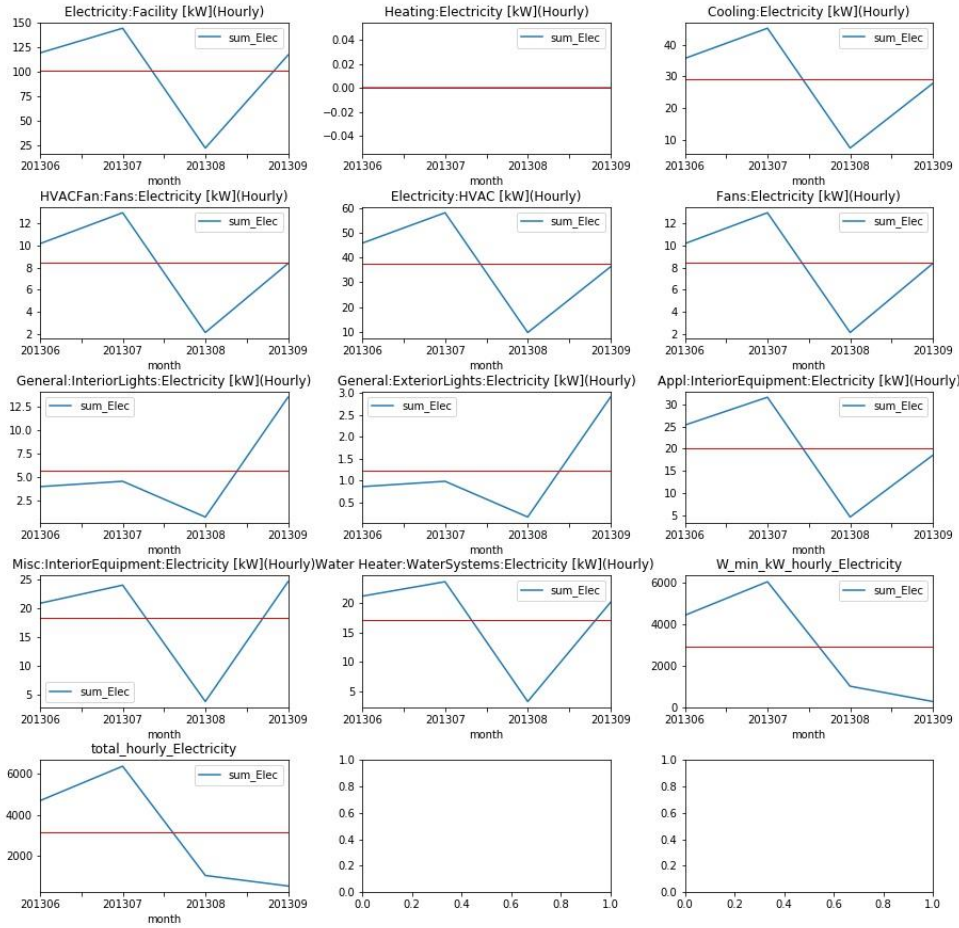
3. By month

The hourly data is aggregated to monthly basis.

- *Finding 1:*

The monthly data is much less granular and contains less information. The abnormal pattern demonstrated in hourly and weekday data is hardly observed here, except for the *General:InteriorLights* and *General:ExteriorLights*.

The sudden decline in the extra appliance's consumption can still be observed in monthly data. The monthly electricity consumption of this appliance before Aug 2013 is in between 0 and 6000 kW.



Assignment 3: EDA and forecast model

The task is to predict RTLMP with explanatory variables including WIND_RTI, GENERATION_SOLAR_RT, RTLoad, etc. For prediction, I create both a linear regression model and a CART (Classification and Regression Tree) model, train the models on in-sample data, and evaluate their performances on the out-of-sample data. I also introduce dummy variables and lagged dependent variables into the models according to my findings from EDA.

The dataset focuses on ERCOT, Electric Reliability Council of Texas, that manages the flow of electric power to more than 26 million Texas customers -- representing about 90 percent of the state's electric load.

1. EDA

Firstly, I explore the range of value for each column in the dataset. One interesting finding is the real-time electricity price (RTLMP) can be negative. This is because produced electricity must go

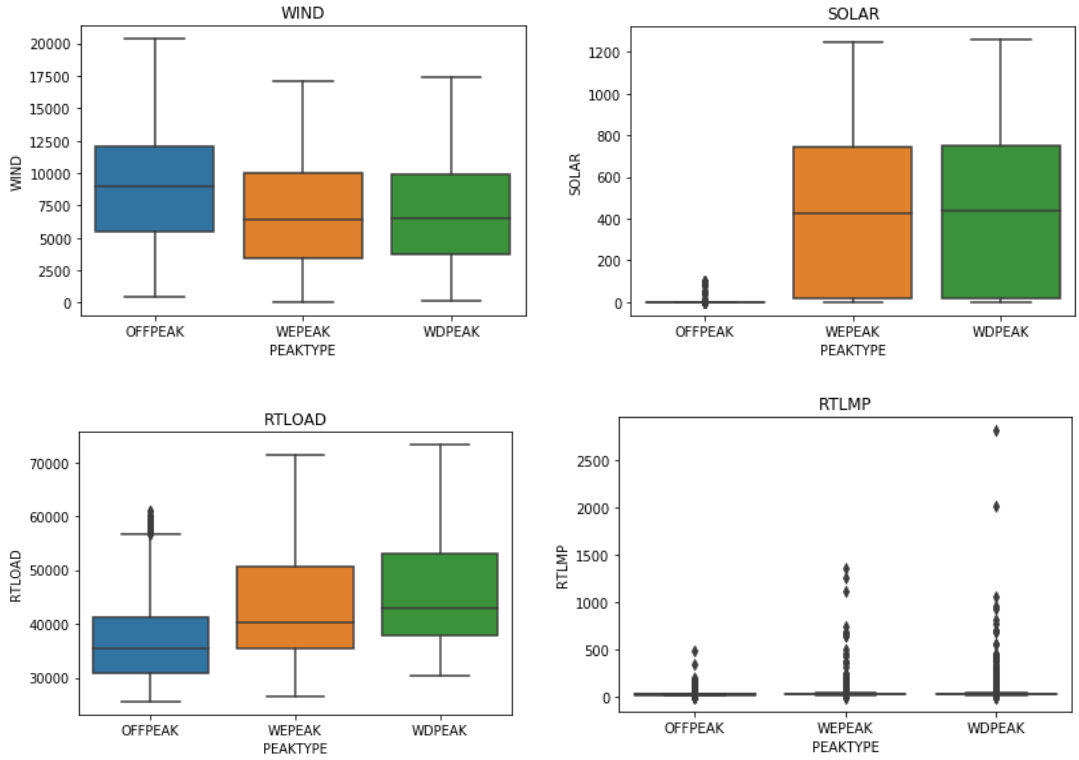
somewhere, which is a unique property of electricity compared to other energies like gas and coal, so during times of low demand or sudden windy times, the electricity is produced at a loss.

The time period of the dataset is from 2017/01/01 to 2018/09/17.

	min	max
RTLMP	-17.86	2809.36
WIND	54.44	20350.4
SOLAR	0	1257.54
RTLOAD	25566.5	73264.7
HOURENDING	1	24
MARKETDAY	2017-01-01 00:00:00	2018-09-17 00:00:00

Next, I plot the boxplots of each variable to see their data distribution. I think the pattern might be different for various peak types, so I categorize the dataset by peak type. There are three peak types: OFFPEAK, WEPEAK, WDPEAK.

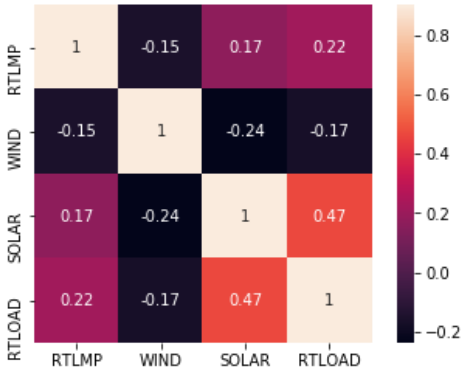
As shown by the plots, hourly Wind generation is significantly higher during OFFPEAK. Hourly Solar generation is significantly lower during OFFPEAK. Hourly actual load is increasing in the order: OFFPEAK < WEPEAK < WDPEAK. The North Hub real-time price is increasing in the order: OFFPEAK < WEPEAK < WDPEAK, in both price level and volatility.



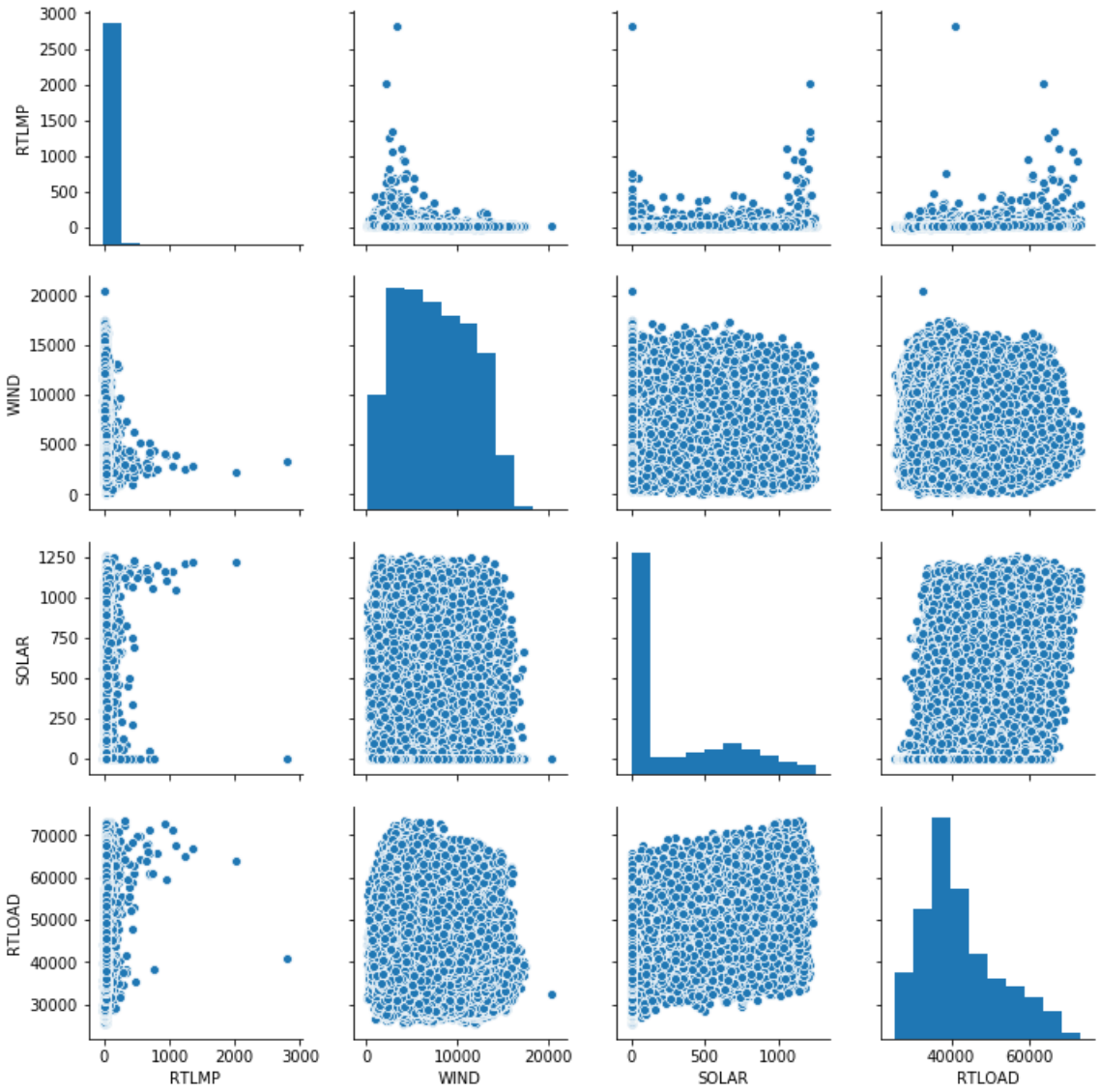
Thirdly, I plot the correlation between variables in order to see how they are correlated and check for possible multicollinearity issue.

For prediction purpose, the dependent variable here is shifted backward in time by one step. For example, the correlation observed between RTLMP and WIND is actually the correlation of the current Wind generation and the future North hub price.

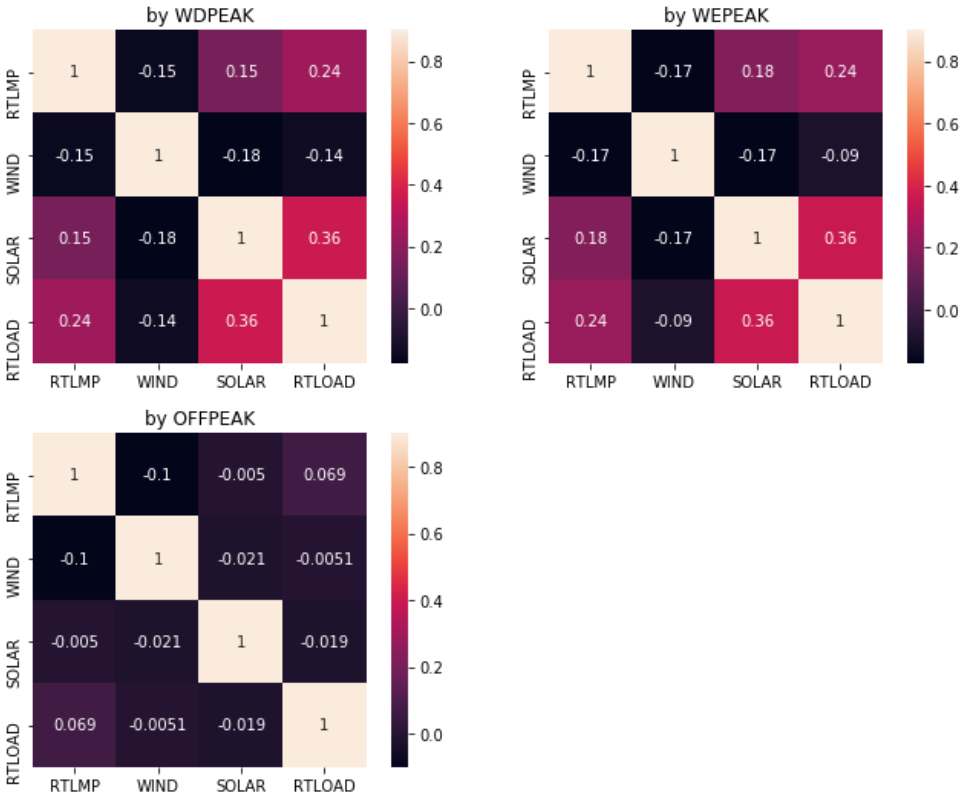
According to the heatmap, WIND is negatively correlated with all the other variables except itself. Solar generation and hourly actual load are the most correlated, with a correlation of 0.47. Since multicollinearity usually refers to a correlation over 0.7, there should be no multicollinearity concern at this stage.



According to the pairplot, the data of the dependent variable is highly concentrated and does not follow a normal distribution. The scatter plots between the dependent variable and the others indicate there could be unequal variance (Heteroscedasticity) problem. What's more, there seems to be outliers in the data (for instance, the two data points with RTMLP over 2000).

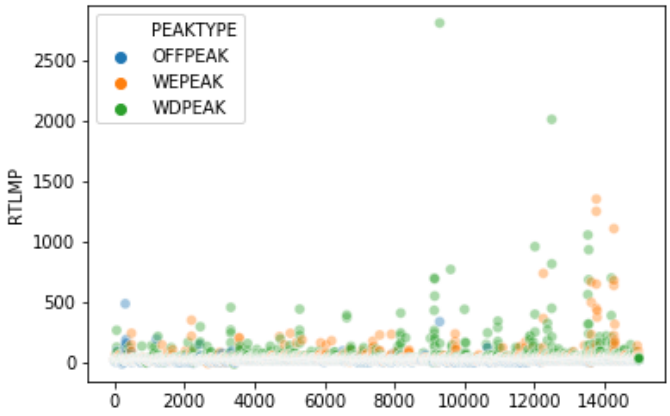


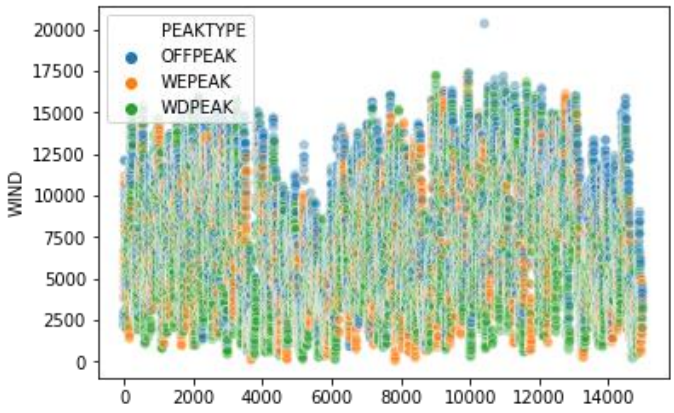
With respect to different peak types, I also plot the heatmaps to see whether the correlation patterns could vary. The heatmaps of WEPEAK and WDPEAK are similar to the original plot, while that of OFFPEAK is somehow different in that the variables become much less correlated, except for WIND and RTLMP.

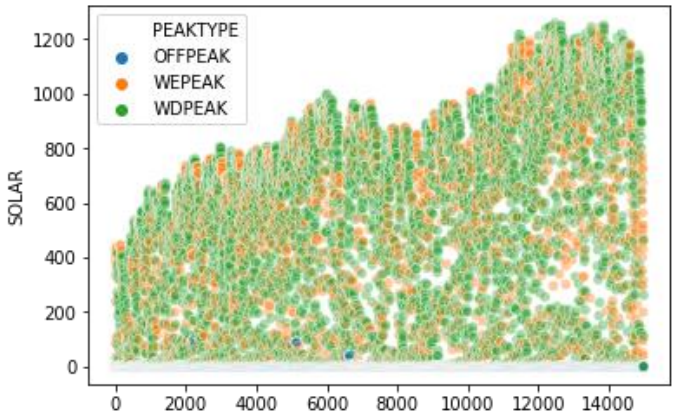


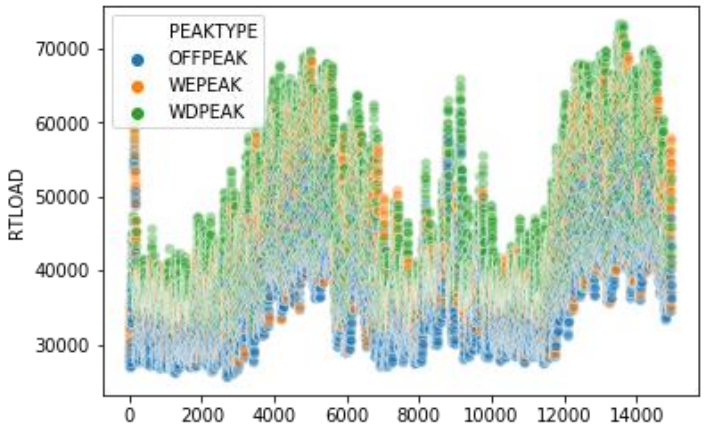
What's more, I scatterplot the data of each variable using different colors for different peak types. As shown by the color distribution, most of the outliers or extremely high values in the price come from WDPEAK and WEPEAK types. OFFPEAK wind generation is the highest, consistent with the boxplots. Only at the bottom side of SOLAR plot can we observe a few blue dots (OFFPEAK), indicating low solar generation during OFFPEAK, consistent with the boxplots as well. RTLOAD is highest for WDPEAK and lowest for OFFPEAK.

WIND is stable within the range of 0 to 17,500. SOLAR shows both an upward trend and an annual seasonality in data. Annual seasonality in RTLOAD is strong, and the peaks appear at around June – July each year; RTLOAD is peaked at 70,000 and bottomed at 25,000.



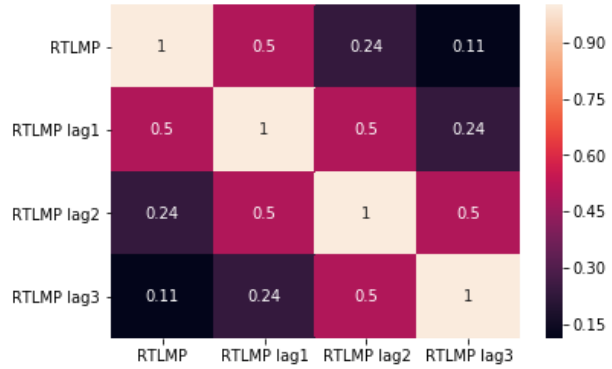


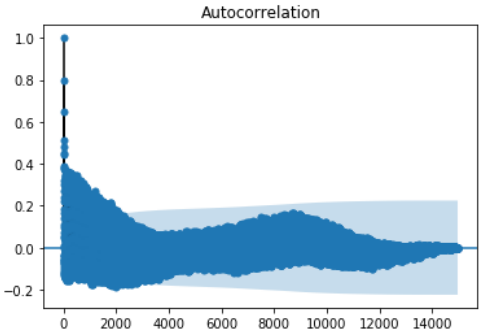


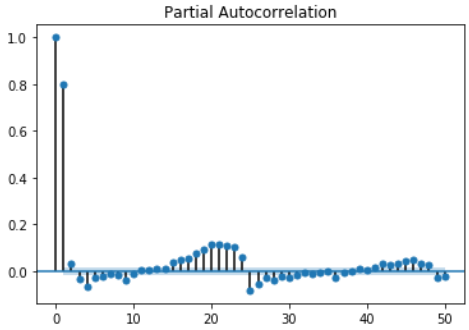


Lastly, but most importantly, since we are dealing with time series, I would like to check the relationship between the dependent variable and its lags up to 3, to see if previous prices are predictive of the future prices. The correlation between RTLMP and RTLMP lag1 is 0.5, while the number is only 0.24 between RTLMP and its lag2, and a mere 0.11 between RTLMP and its lag3, suggesting a gradually weakening pattern.

As suggested by the ACF and PACF plots, I would like to use RTLMP up to lag 1 in the prediction models.



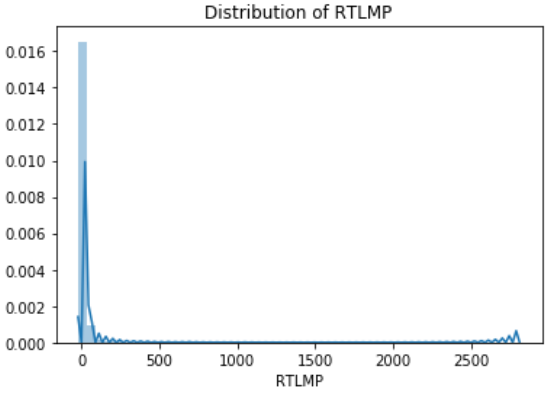


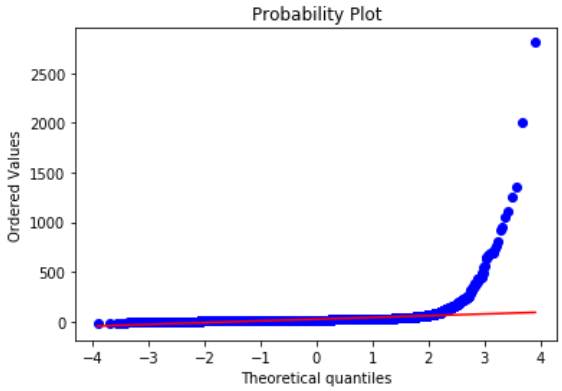


2. Data Preprocessing

First is outlier treatment. As shown in the pairplot, there is evidence of the existence of outliers in data. Therefore, I remove all data with RTLMP over 2,000 or with WIND over 20,000. Altogether three datapoints are removed.

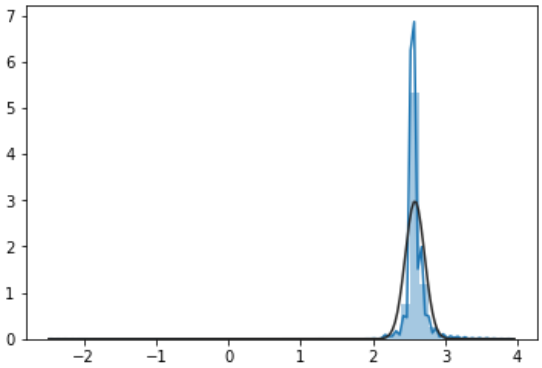
Originally, RTLMP has a distribution as below, which is right-skewed and has fat tail. The plot on RHS is the corresponding normal probability plot.

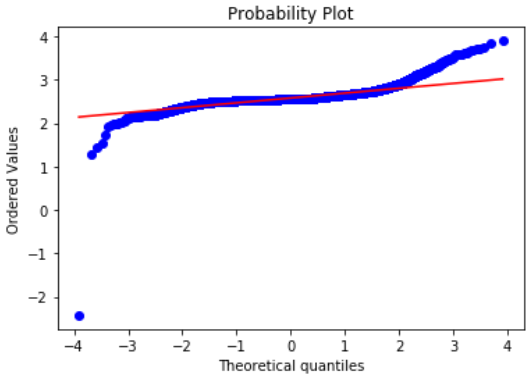




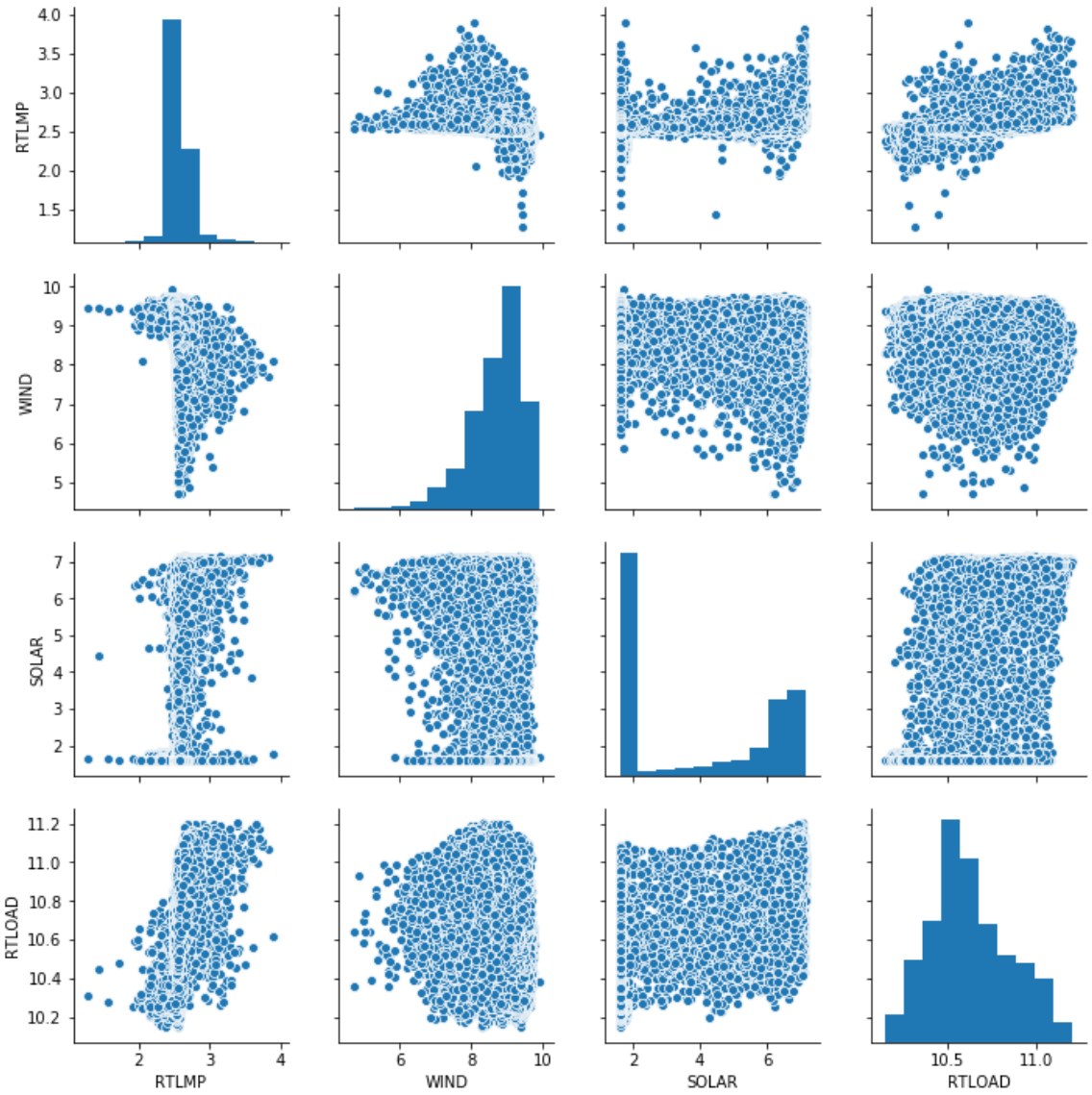
Then I apply a Box-Cox transformation on RTLMP data to remedy the non-normality distribution problem. The best lambda found is -0.207. The transformation formula is $y'_\lambda = \frac{y^\lambda - 1}{\lambda}$.

After transformation, the distribution of RTLMP is as below.





Plot again the heatmap and the pairplot.



3. Predictive Model

3.1 Model 1 – Linear Regression estimated by WLS

I split the data into in-sample and out-of-sample datasets, respectively for training and test purposes. The in-sample set accounts for 80% of the entire dataset.

Thanks to the changing patterns found in the variables during different peak types, I introduce two dummy variables, peak_dummy_WDPEAK and peak_dummy_WEPEAK to account for it.

$$peak\ dummy\ WDPEAK = \begin{cases} 1 & if\ PEAK\ TYPE = WDPEAK \\ 0 & otherwise \end{cases},$$

$$peak\ dummy\ WEPEAK = \begin{cases} 1 & if\ PEAK\ TYPE = WEPEAK \\ 0 & otherwise \end{cases},$$

If both dummies are zero-valued, that means the current peak type is OFFPEAK.

The regression model is

$$RTLMP_{t+1} = \beta_0 + \beta_1 WIND_t + \beta_2 SOLAR_t + \beta_3 RTLOAD_t + \beta_4\, peak\ dummy\ WDPEAK + \beta_5\, peak\ dummy\ WEPEAK + \beta_6 RTLMP_t + \beta_7 RTLMP_{t-1} + \epsilon_t,$$

where ϵ_t is IID and follows N(0, 1). Note the model introduces the dependent variable up to lag1.

Due to the possible heteroscedasticity observed from the pairplot, instead of OLS, I decide to use WLS to remedy this issue. The heteroscedasticity is mainly observed in the scatterplot between SOLAR and RTLMP, that the variance of price is larger for relatively large or small solar generations. Therefore, the weight is set to be the inverse of the absolute value of SOLAR.

The regression result is as below:

WLS Regression Results

Dep. Variable:	RTLMP	R-squared:	0.575
Model:	WLS	Adj. R-squared:	0.575
Method:	Least Squares	F-statistic:	2318.
Date:	Tue, 17 Nov 2020	Prob (F-statistic):	0.00
Time:	12:13:06	Log-Likelihood:	12862.
No. Observations:	11983	AIC:	-2.571e+04
Df Residuals:	11975	BIC:	-2.565e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4804	0.044	10.838	0.000	0.394	0.567
WIND	-0.0229	0.001	-19.612	0.000	-0.025	-0.021
SOLAR	0.0040	0.000	8.494	0.000	0.003	0.005
RTLOAD	0.0707	0.005	15.279	0.000	0.062	0.080
peak_dummy_WDPEAK	-0.0003	0.002	-0.141	0.888	-0.004	0.003
peak_dummy_WEPEAK	0.0044	0.002	1.955	0.051	-1.2e-05	0.009
RTLMP current	0.5207	0.008	61.404	0.000	0.504	0.537
RTLMP previous	0.0731	0.008	8.845	0.000	0.057	0.089

Omnibus:	11018.127	Durbin-Watson:	1.793
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3117638.328
Skew:	3.704	Prob(JB):	0.00
Kurtosis:	81.672	Cond. No.	939.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Several conclusions:

1. All the independent variables are significant at a 95% level except for the peak_dummy_WDPEAK, according to their p values.

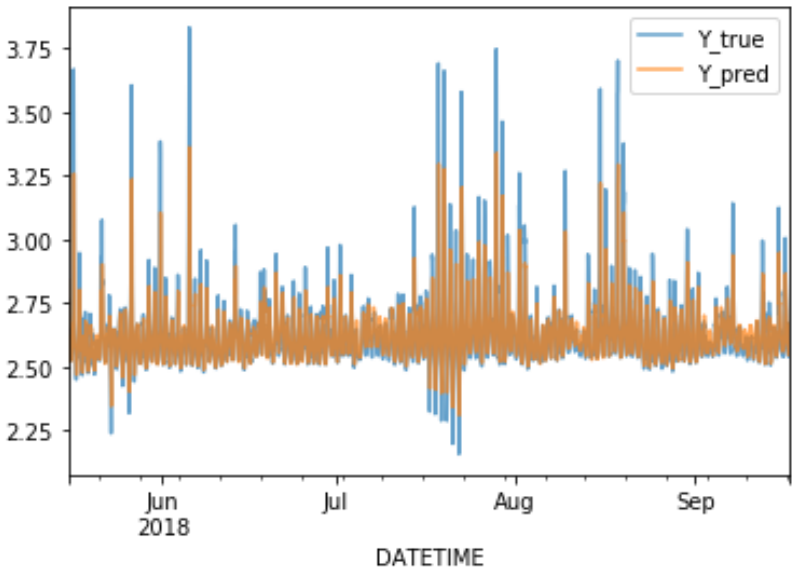
2. Holding everything else unchanged, a one-unit increase in wind generation would drive down the North hub electricity price by 0.023 on average. This is consistent with the results and guess from EDA.

3. For predicting $RTLMP_{t+1}$, the variable $RTLMP_t$ is playing an important role as the estimated coefficient of it is as high as 0.5207, suggesting that if current price is high/low, the future price is likely to follow the momentum and keeps high/low.

4. Both the R squared and adjusted R squared are 0.575, which is nice. R squared measures how much of the variation in the dependent variable is explained by the variations in the independent variables.

With the fitted linear regression model, I test the model's performance on the out-of-sample dataset. A comparison of the predicted prices and the true prices is plotted. The overall prediction capacity of my linear model is nice, as it's quite good at capturing the upward / downward trends in the price, probably due to the introduction of lagged prices. The predicted time series fluctuates similarly to the real price series.

One weakness of the model is that it's not able to capture the extreme values, namely the extreme peaks / troughs as shown by the blue curve. On the test set, this model has a MSE of 0.005569 and a MAE of 0.036438.



3.2 Model 2 – CART optimized with grid search

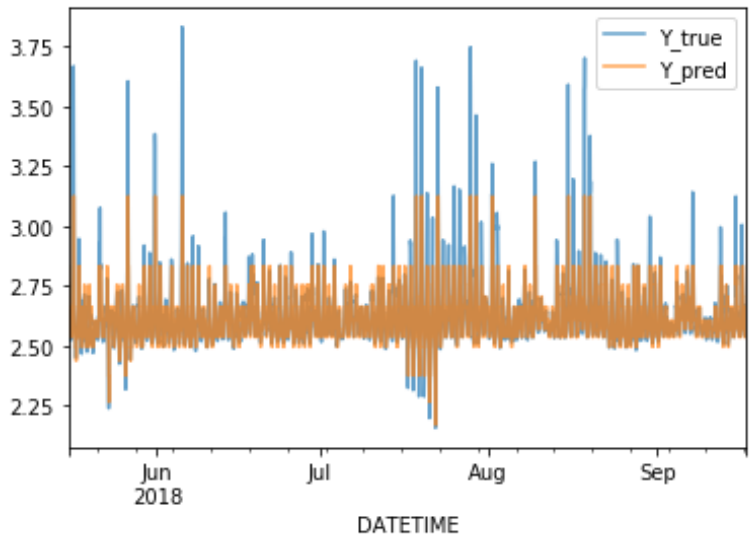
The linear model works well, but I am also curious about the performance of a non-linear model on the electricity price dataset.

I use a decision tree regressor and optimize the hyperparameters with grid search. The optimal parameters of the tree are:

max_depth: 4, *min_samples_leaf*: 5, *min_samples_split*: 3.

The tree is trained and evaluated in the same way as Model 1. On the test set it achieves a MSE of 0.006324 and a MAE of 0.040408, slightly worse than Model 1.

As shown by the plot, the prediction by Model 2 is not as flexible and changeable as Model 1, and does not match the real prices as well as Model 1. I think Model 1 has better generalization performance on the out-of-sample data.



Assignment 4 (Optional): Learn products of Futures

We are offered three futures products here:

- ERN: ERCOT North 345KV Real-Time Peak Fixed Price Future

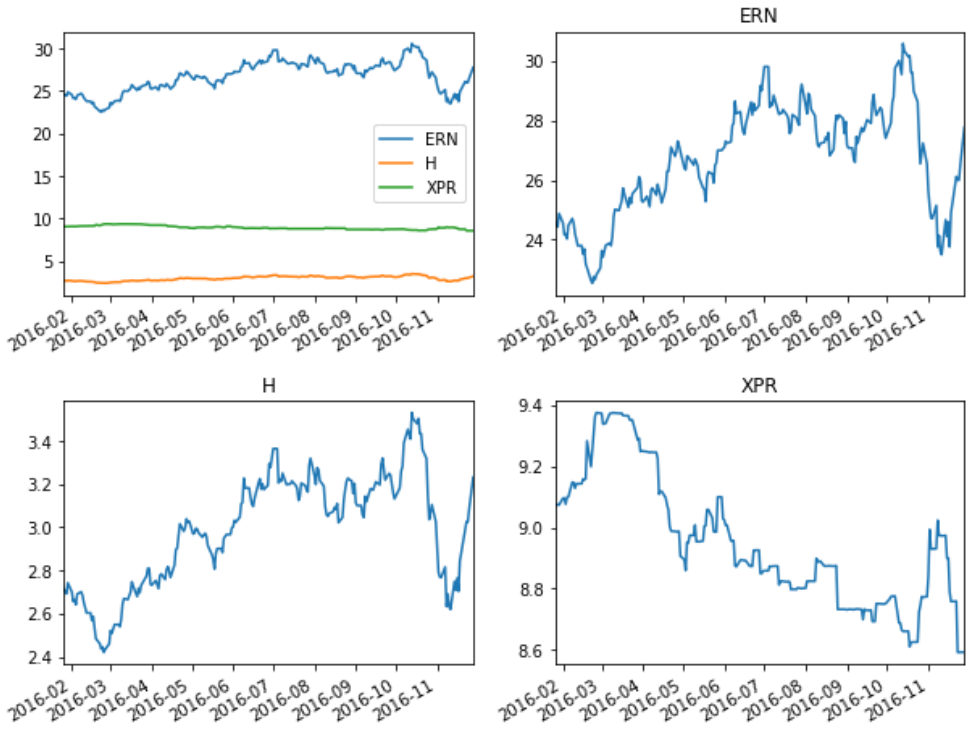
- H: Henry LD1 Fixed Price Future

- XPR: ERCOT North 345kV Physical HR Peak HE 0700-2200

We are holding the physical power and can only use a combination of H and XPR to hedge the exposure. A high-level understanding of these contracts is that ERN is the power futures, H is the natural gas futures, and XPR is the heat rate futures. Heat rate is "a measure of the efficiency at which a plant converts natural gas into electricity as expressed in millions of British thermal units (MMBtu) per megawatt hour (MWh)", and it is "the current price of power divided by the price of natural gas for a specific term and location" (https://business.directenergy.com/blog/2013/April/4-Key-Things-You-Should-Know-About-Energy-Heat-Rates).

$$XPR = \frac{ERN}{H}$$

The dataset contains daily futures prices from 2016-01-26 to 2016-11-28.



Assumptions:

- Due to lack of data, take ERN price as an approximation for physical power prices in the market

- Assume that we hold 1,000 MWH of physical power; the spot month limit of H is 4,000, here the limit is also exceeded

- MARGIN account is not considered here (including initial margin and maintenance margin) for simplicity

- Hold a fixed number of contracts in XPR, and rebalancing H on a weekly basis

- Two goals of the hedging: first, avoid large losses in daily and final PnL; second, try to keep the volatility of daily PnL small

- Two hyperparameters in hedging strategy: N_prev, Adjust_ratio

N_prev refers to the number of dates to trace back for the average heat rate level.

Adjust_ratio refers to the adjustment factor applied on the number of contracts in H, which adjusts for the imperfect correlation between power price and has price. It's a value in the range of 0 to 1.

- At the end of each month we close current positions in H and XPR, and roll into new monthly H and XPR contracts.

Description of hedging strategies:

I use one dataframe to store the positions and another to compute the PnLs.

The fixed number of contracts in XPR is pre-determined. The contract size of ERN is 1 MW·h, and that of XPR is 50 MW per hour. Originally, if the liquidation of ERN is good, we are supposed to hedge our physical power exposure by shorting 1,000 ERN contracts to offset any losses in power prices. Now instead, I short $\frac{1000}{50} = 20$ contracts of XPR.

H is dynamically rebalanced to adjust for the change in heat rate over time. For example, 2016-04-20 is one of the dates to rebalance H, then the strategy looks back N_prev number of days to compute an average historical heat rate, denoted as $\overline{heat\ rate_{N_{prev}}}$. Then the number of contracts into H on 2016-04-20 would be

$$N_H = \overline{heat\ rate_{N_{prev}}} * N_{XPR} * \frac{Multiplier_{XPR}}{Multiplier_H} = 0.4 * heat\ rate_{N_{prev}} ,$$

Where $N_{XPR} = 20$ is known, $Multiplier_{XPR} = 50$ and $Multiplier_H = 2{,}500$ are written in the contract.

What's more, after some experimental tests, I find that the amount of $0.4 * heat\ rate_{N_{prev}}$ could lead to over-hedging in some scenarios, for instance, when the power price plummets and gas price rallies, causing large losses. Therefore, I apply a parameter $Adjust\_ratio$ on N_H to alleviate the influence of it:

$$N_H = N_H * Adjust\_ratio .$$

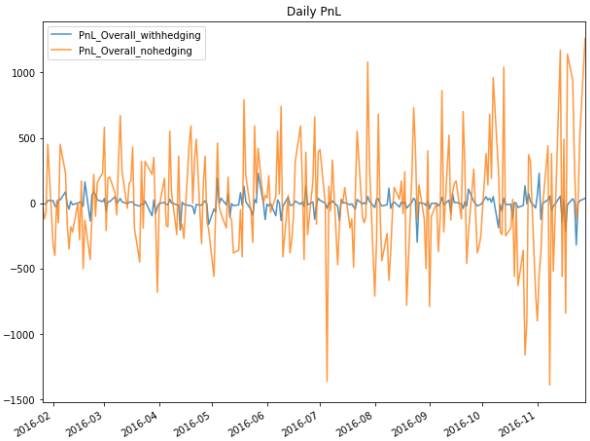
The choices of hyperparameters $Adjust\_ratio$ and N_prev are to be discussed later. After the positions of H contracts are decided, I compute the PnL of power price, H and XPR on a daily basis. Performances would be shown later.

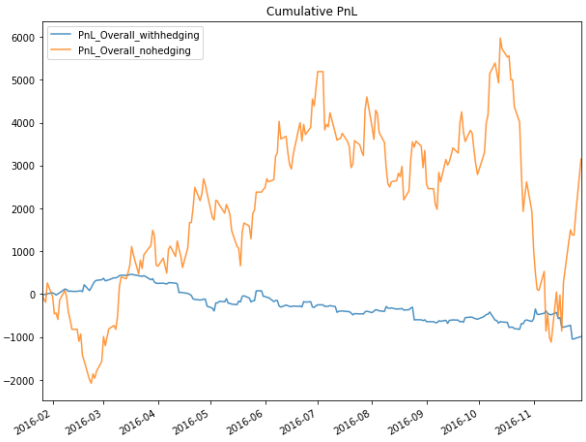
Going back the searching for the optimal hyperparameters, I implement a grid search on all combinations of $N_{prev} \in [1, 100]$ and $Adjust\ ratio \in [0.9, 1]$. The first optimization focuses on minimizing the standard deviation of the daily PnL, with a constraint of keeping the final losses no larger than $1,000. The second optimization conducted prioritizes maximizing the final PnL, with a constraint of keeping the overall PnL standard deviation under 70.

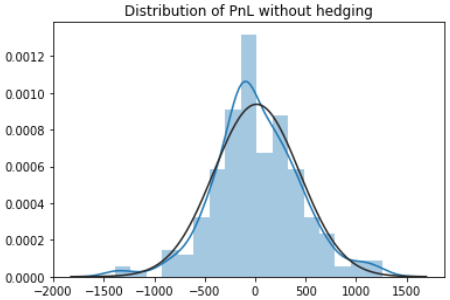
The first optimization suggests adopting $N_{prev} = 1,\ Adjust\ ratio = 0.95$ (Result 1).

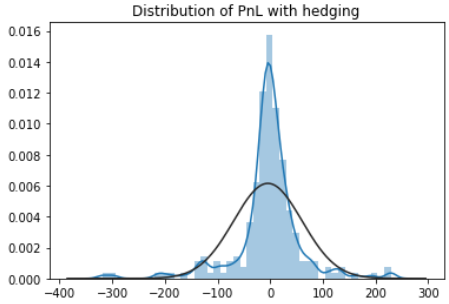
The second optimization suggests adopting $N_{prev} = 56,\ Adjust\ ratio = 0.9$ (Result 2).

Performance of Result 1 is as below.



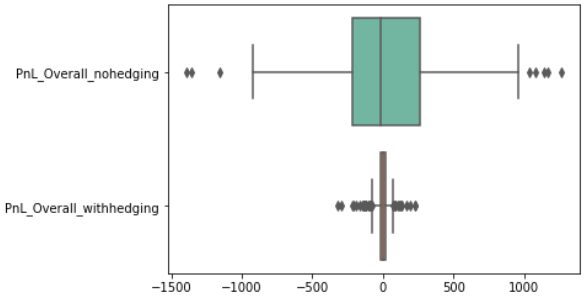




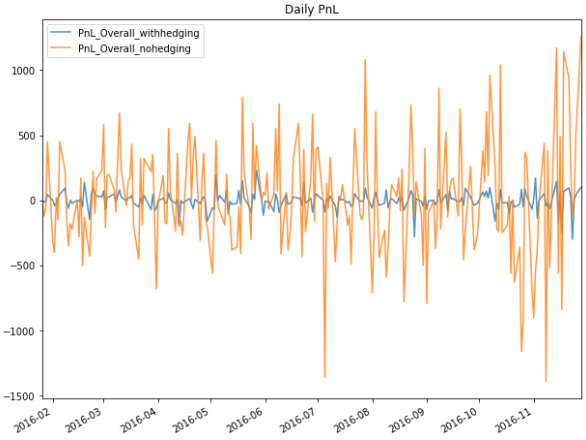


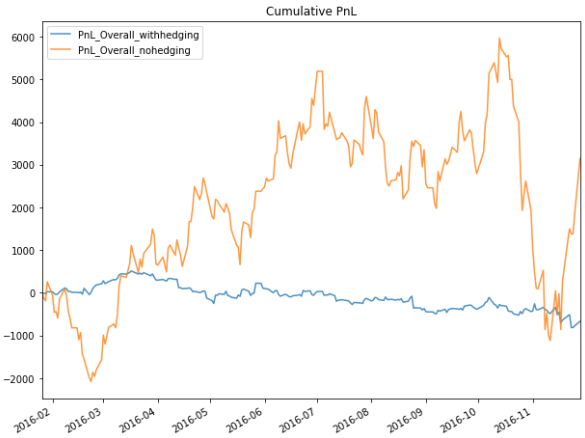

```
count      219.000000
mean        14.383562
std        426.434932
min      -1390.000000
25%       -220.000000
50%        -20.000000
75%        260.000000
max       1260.000000
Name: PnL_Overall_nohedging, dtype: float64
```

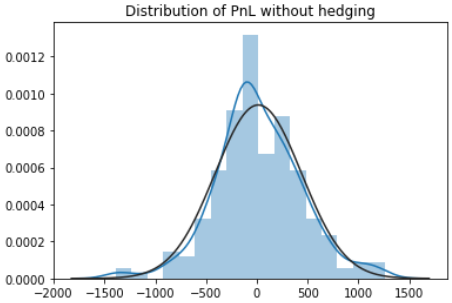
```
count    219.000000
mean      -4.519868
std       65.042444
min     -319.114406
25%      -18.987302
50%       -2.420323
75%       18.622294
max      228.859834
Name: PnL_Overall_withhedging, dtype: float64
```

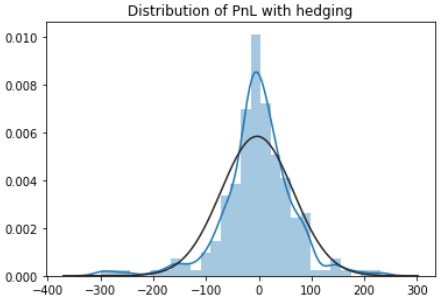


Performance of Result 2 is as below.



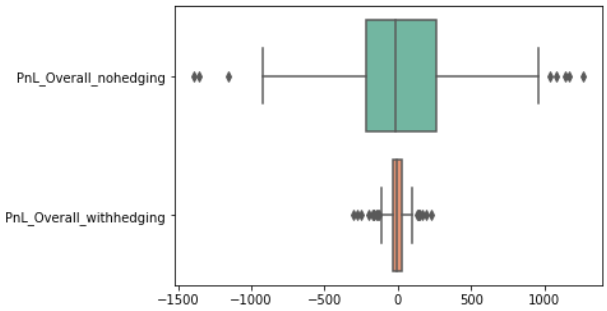







```
count      219.000000
mean        14.383562
std        426.434932
min      -1390.000000
25%       -220.000000
50%        -20.000000
75%        260.000000
max       1260.000000
Name: PnL_Overall_nohedging, dtype: float64
```

```
count      219.000000
mean        -3.056346
std         68.612674
min       -300.242377
25%        -29.298189
50%         -3.493697
75%         29.533501
max        231.616862
Name: PnL_Overall_withhedging, dtype: float64
```



Comparing Result 1 & 2, Result 1 hedges more accurately, stabilizes the daily PnL, but is slightly less profitable than Result 2. Result 2 is slightly more volatile than Result 1 in terms of PnL flow, but is more profitable at the end. This is probably due to the rally in power price from Mar to Oct 2016. The **conclusion** is If the price plummets in a year, Result 1 with minimal std is preferred, but if the market rallies in a year, Result 2 with maximal gains might be preferred.