PJM HOURLY ENERGY CONSUMPTION

PROJECT - 01 : GROUP - 03

FINAL REVIEW PRESENTATION

GROUP MEMBERS -

DEEPESH, DEVA DHARSHINI, SUJAL,

SHIVA, VIVEK, SRAVYA.

BUSINESS OBJECTIVE :

ABOUT DATASET :-

PJM Interconnection LLC (PJM) is a regional transmission organization (RTO) in the United States. It is part of the Eastern Interconnection grid operating an electric transmission system serving all or parts of Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia, and the District of Columbia. The hourly power consumption data comes from PJM's website and are in megawatts (MW).

PROBLEM STATEMENT :-

1. Split the last year into a test set- can you build a model to predict energy consumption.
2. Find trends in energy consumption around hours of the day, holidays, or long term trends.
3. Understand how daily trends change depending of the time of year. Summer trends are very different than winter trends.
4. Forecast for next 30 days.

PROJECT FLOW :

1. Understanding the problem statement
2. EDA (Exploratory Data Analysis)
    → Data Pre-processing
    → Data cleaning
    → Visualization
    → Weekday, holiday and seasonal Trend
    → Outlier Detection
    → Outlier Removal
1. Model Building
    → 5 Models
    → Choosing The Final Model
    → Forecast For Next 30 Days
1. Model Deployment

PHASE - 1

EXPLORATORY DATA ANALYSIS (EDA)

Data Information

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143206 entries, 0 to 143205
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Datetime  143206 non-null  datetime64[ns]
 1   PJMW_MW   143206 non-null  float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 2.2 MB
```

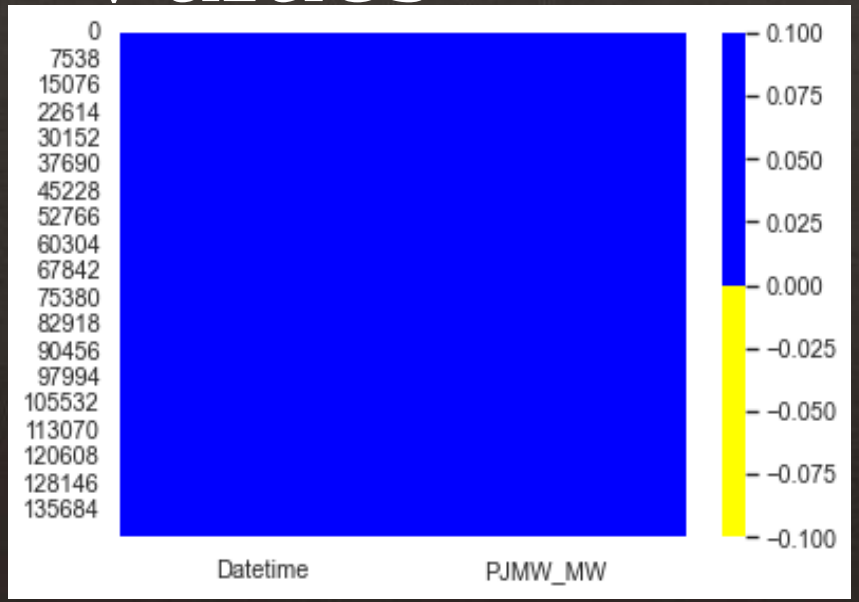
Data Description

```
In [9]: df.describe()
```

Out[9]:

	PJMW_MW
count	143206.000000
mean	5602.375089
std	979.142872
min	487.000000
25%	4907.000000
50%	5530.000000
75%	6252.000000
max	9594.000000

Checking for Null Values



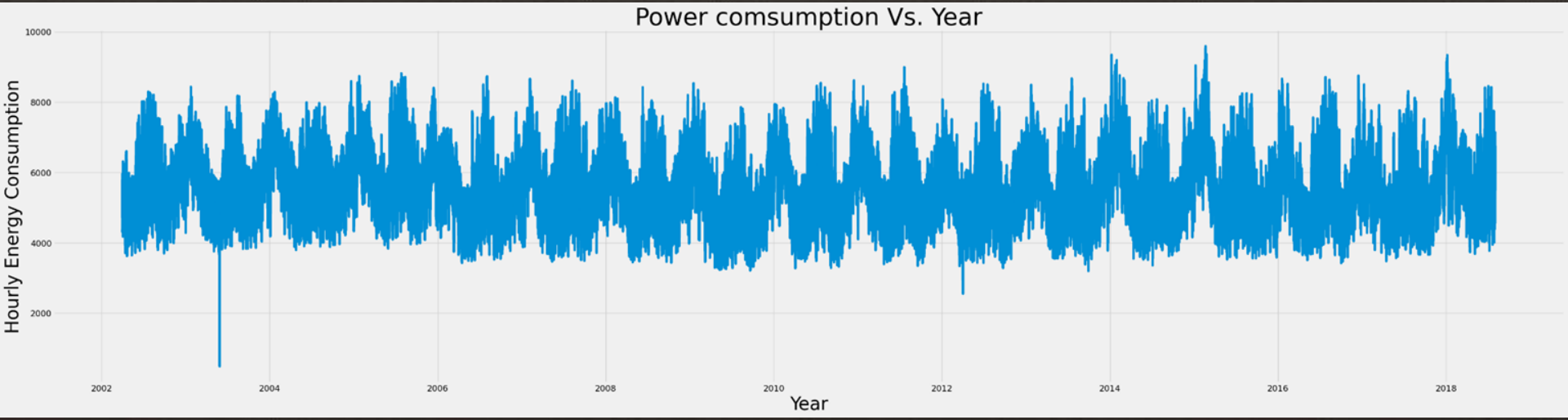

```
In [4]: df.duplicated().sum()

Out[4]: 0
```

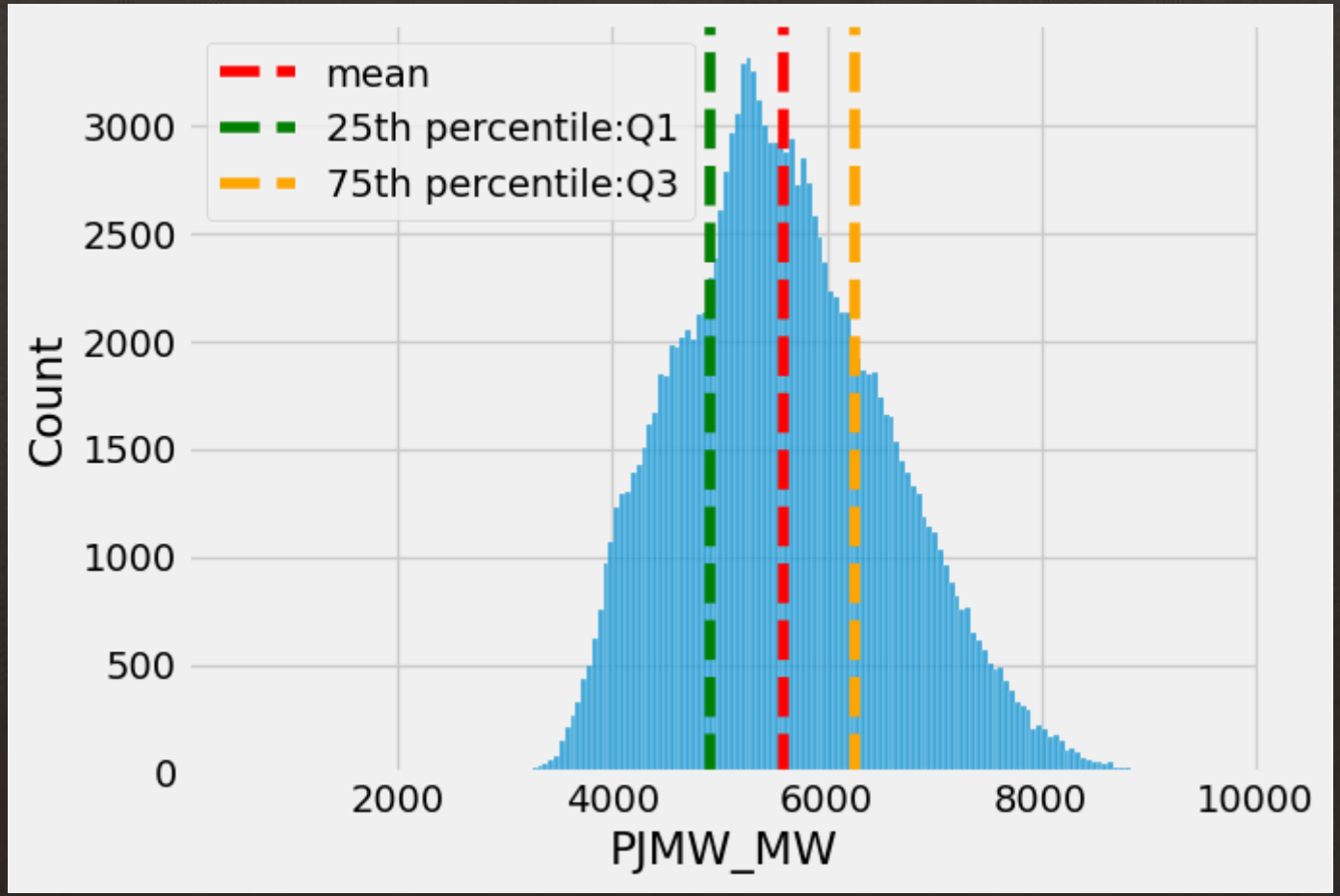
```
In [11]: df.isnull().sum()

Out[11]: Datetime    0
         PJMW_MW     0
         dtype: int64
```

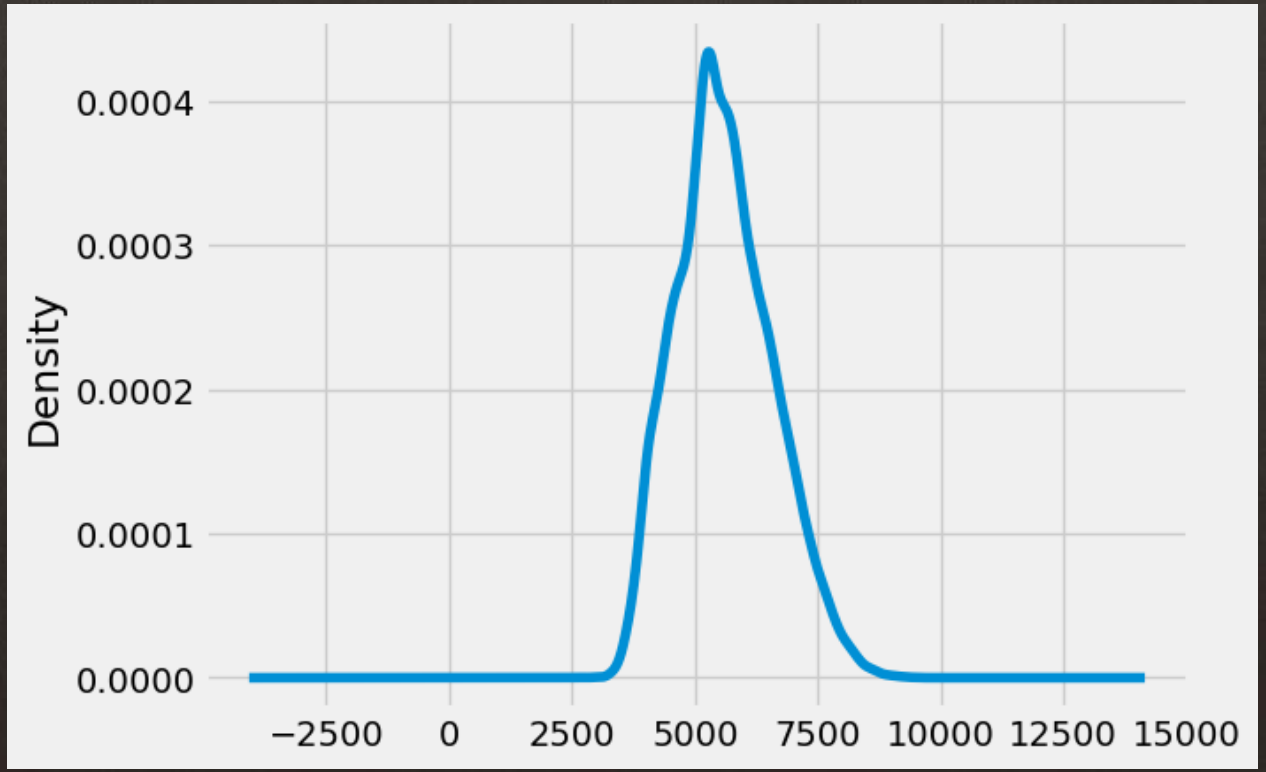
Plotting the data



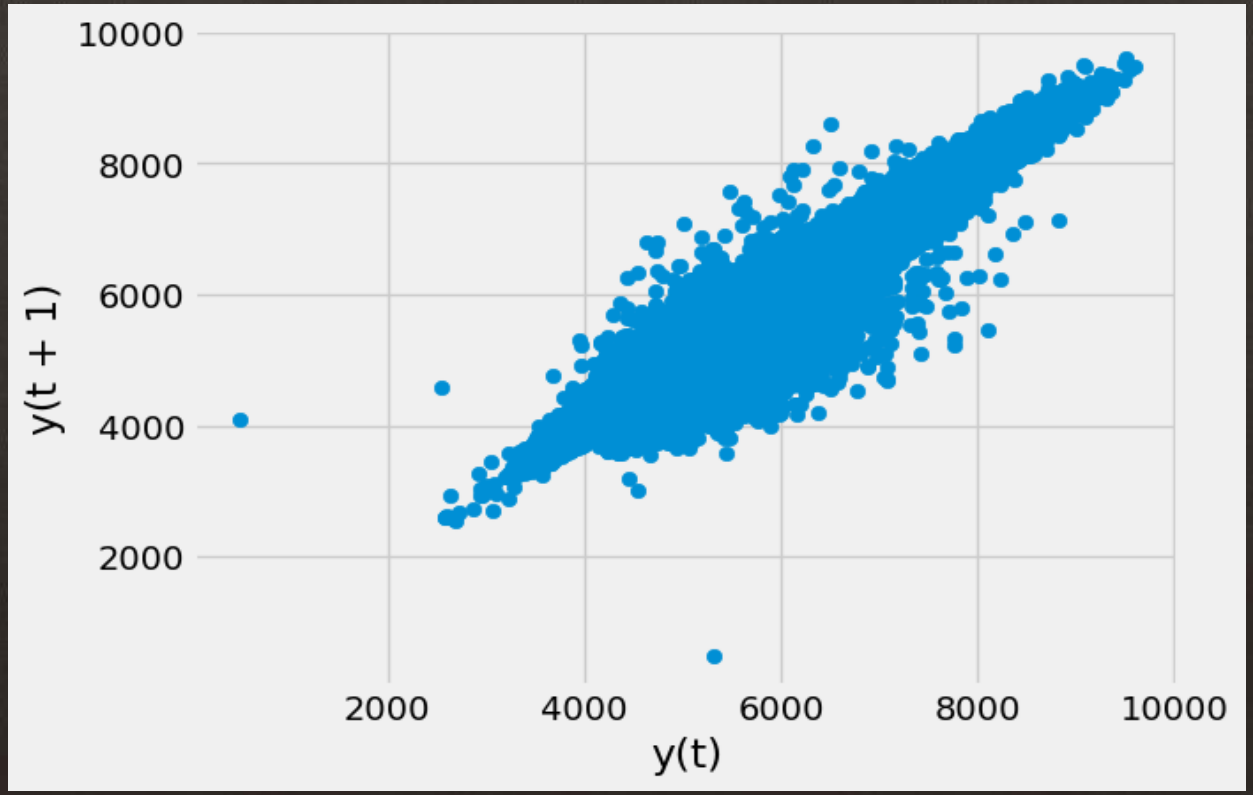
Histogram


mean
25th percentile:Q1
75th percentile:Q3
Count
3000
2500
2000
1500
1000
500
0
2000
4000
6000
8000
10000
PJMW_MW

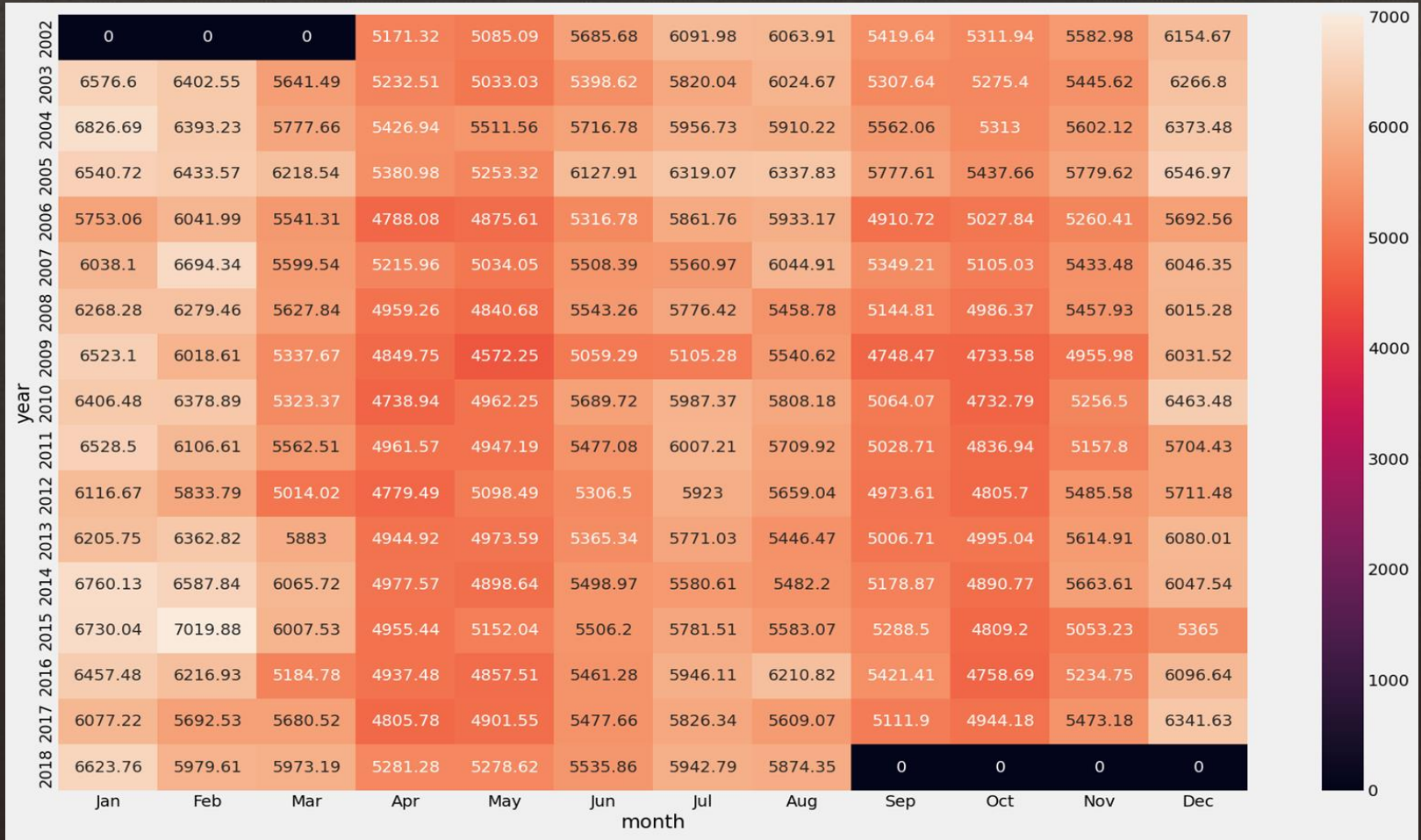
Density Plot



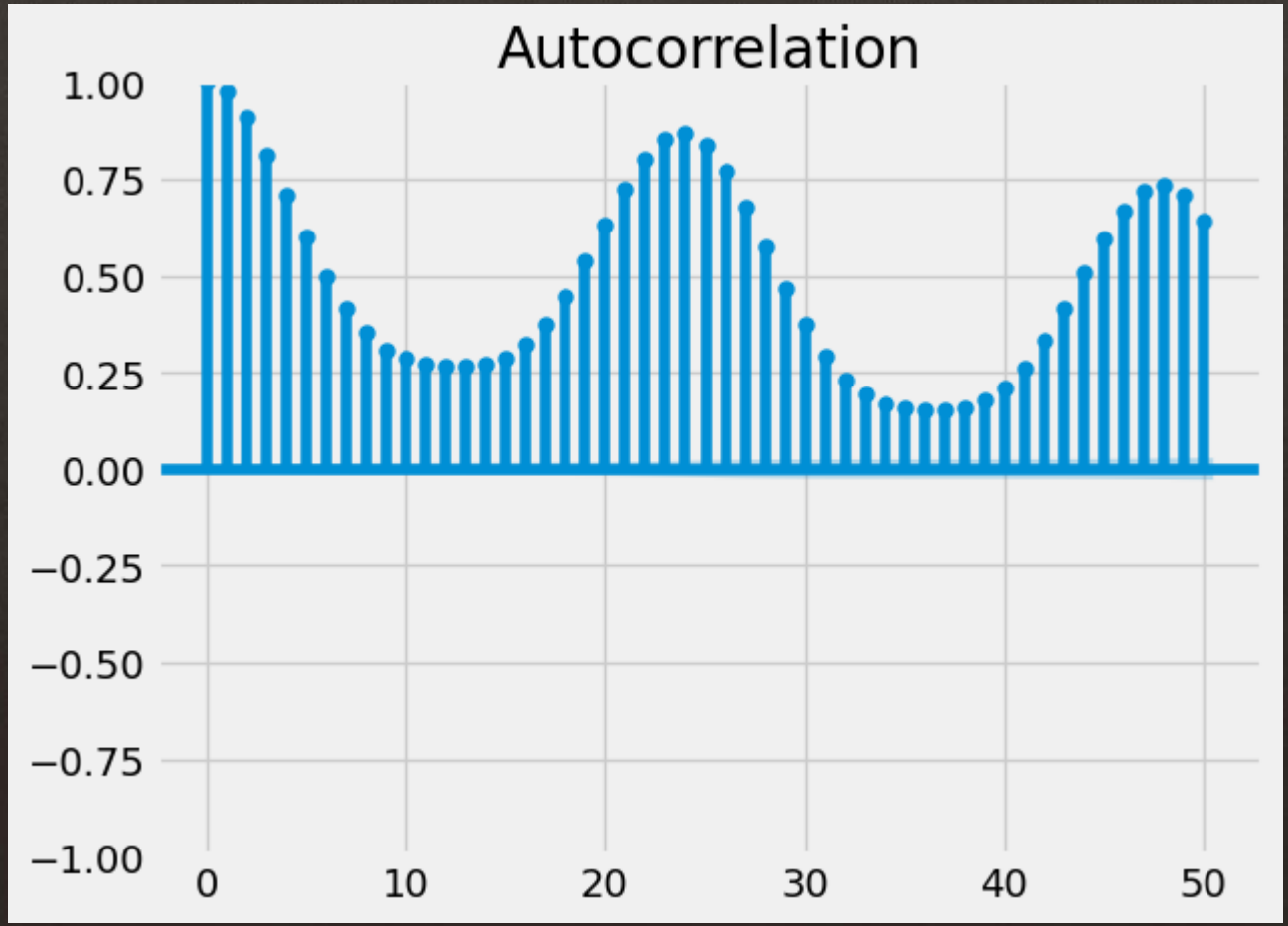
Lag Plot



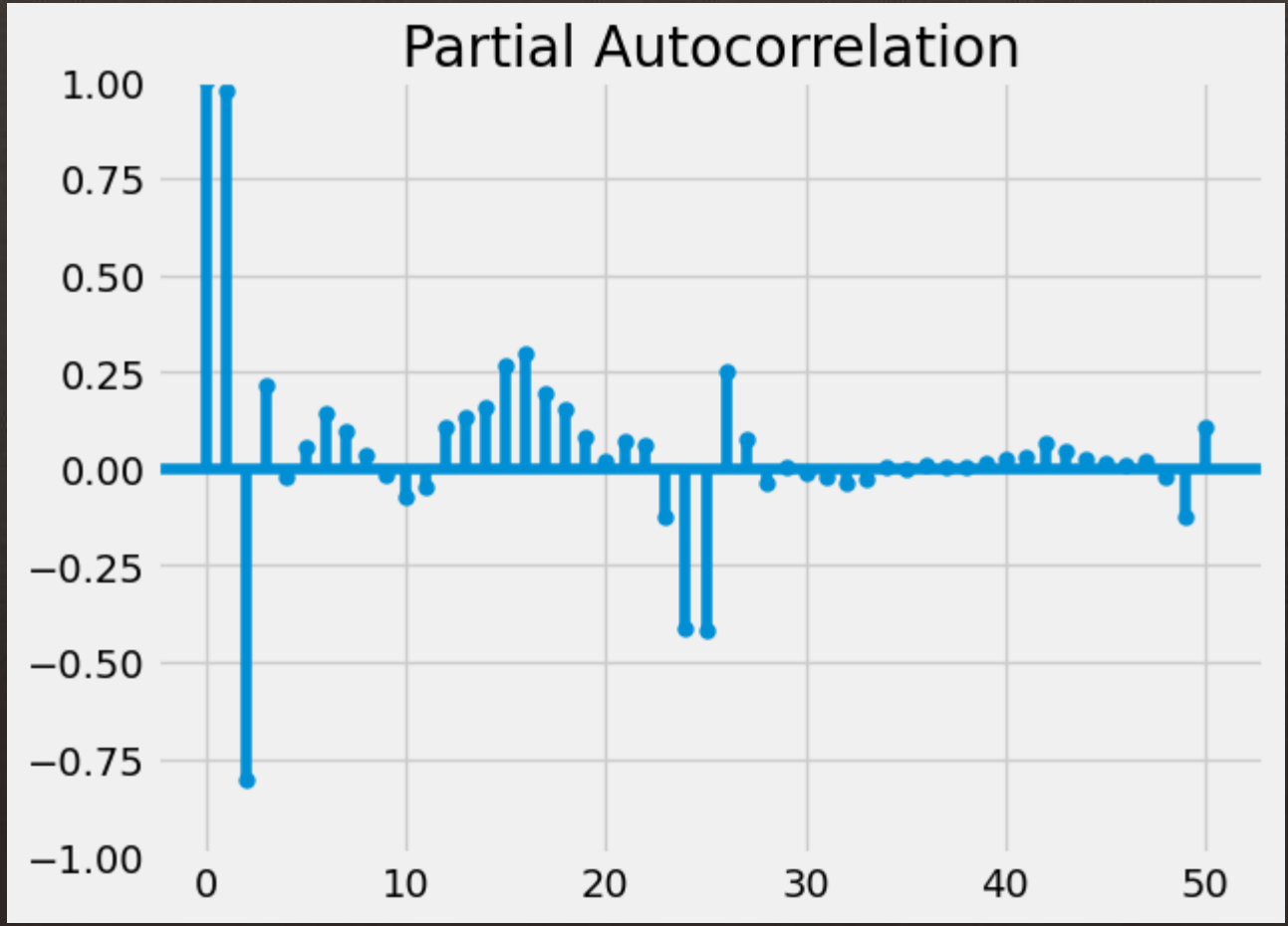
Heatmap


year
month
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2002 0 0 0 5171.32 5085.09 5685.68 6091.98 6063.91 5419.64 5311.94 5582.98 6154.67
2003 6576.6 6402.55 5641.49 5232.51 5033.03 5398.62 5820.04 6024.67 5307.64 5275.4 5445.62 6266.8
2004 6826.69 6393.23 5777.66 5426.94 5511.56 5716.78 5956.73 5910.22 5562.06 5313 5602.12 6373.48
2005 6540.72 6433.57 6218.54 5380.98 5253.32 6127.91 6319.07 6337.83 5777.61 5437.66 5779.62 6546.97
2006 5753.06 6041.99 5541.31 4788.08 4875.61 5316.78 5861.76 5933.17 4910.72 5027.84 5260.41 5692.56
2007 6038.1 6694.34 5599.54 5215.96 5034.05 5508.39 5560.97 6044.91 5349.21 5105.03 5433.48 6046.35
2008 6268.28 6279.46 5627.84 4959.26 4840.68 5543.26 5776.42 5458.78 5144.81 4986.37 5457.93 6015.28
2009 6523.1 6018.61 5337.67 4849.75 4572.25 5059.29 5105.28 5540.62 4748.47 4733.58 4955.98 6031.52
2010 6406.48 6378.89 5323.37 4738.94 4962.25 5689.72 5987.37 5808.18 5064.07 4732.79 5256.5 6463.48
2011 6528.5 6106.61 5562.51 4961.57 4947.19 5477.08 6007.21 5709.92 5028.71 4836.94 5157.8 5704.43
2012 6116.67 5833.79 5014.02 4779.49 5098.49 5306.5 5923 5659.04 4973.61 4805.7 5485.58 5711.48
2013 6205.75 6362.82 5883 4944.92 4973.59 5365.34 5771.03 5446.47 5006.71 4995.04 5614.91 6080.01
2014 6760.13 6587.84 6065.72 4977.57 4898.64 5498.97 5580.61 5482.2 5178.87 4890.77 5663.61 6047.54
2015 6730.04 7019.88 6007.53 4955.44 5152.04 5506.2 5781.51 5583.07 5288.5 4809.2 5053.23 5365
2016 6457.48 6216.93 5184.78 4937.48 4857.51 5461.28 5946.11 6210.82 5421.41 4758.69 5234.75 6096.64
2017 6077.22 5692.53 5680.52 4805.78 4901.55 5477.66 5826.34 5609.07 5111.9 4944.18 5473.18 6341.63
2018 6623.76 5979.61 5973.19 5281.28 5278.62 5535.86 5942.79 5874.35 0 0 0 0
7000
6000
5000
4000
3000
2000
1000
0

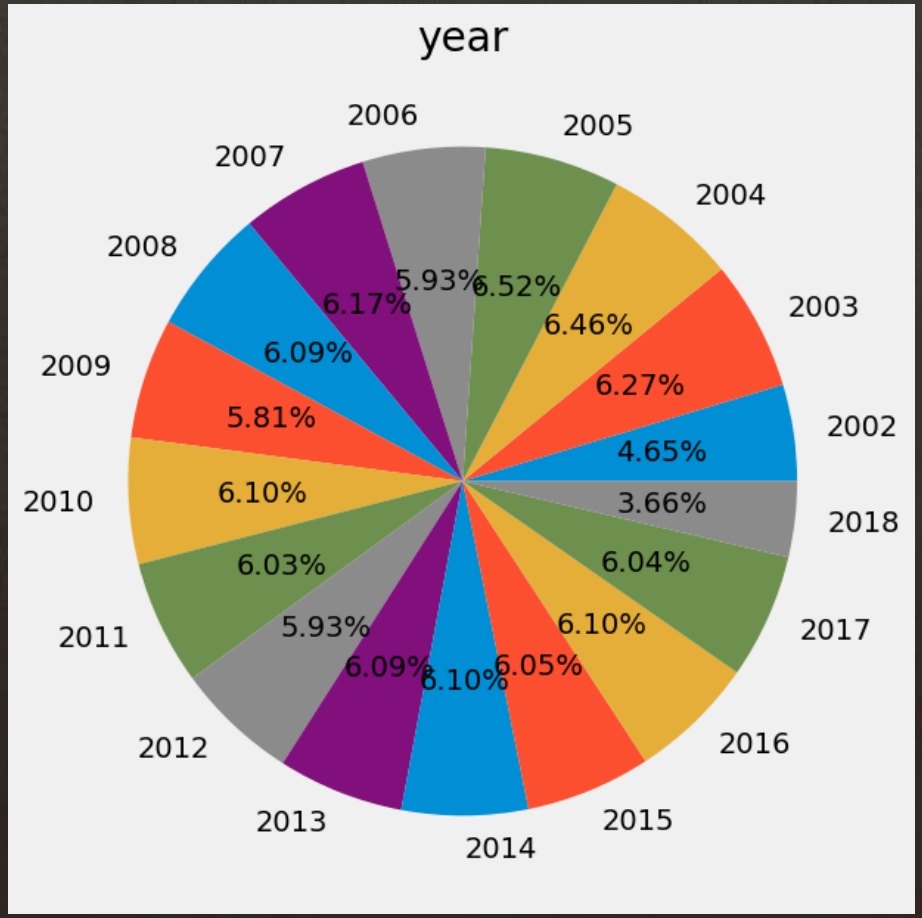
ACF Plot

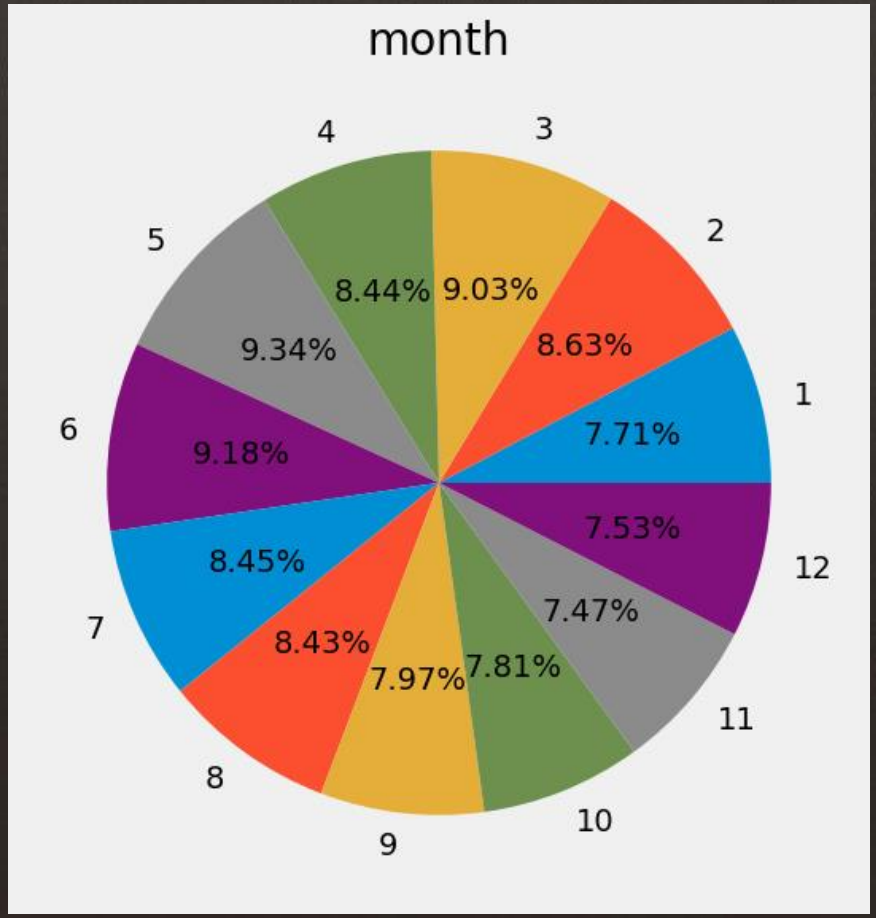

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50

PACF Plot

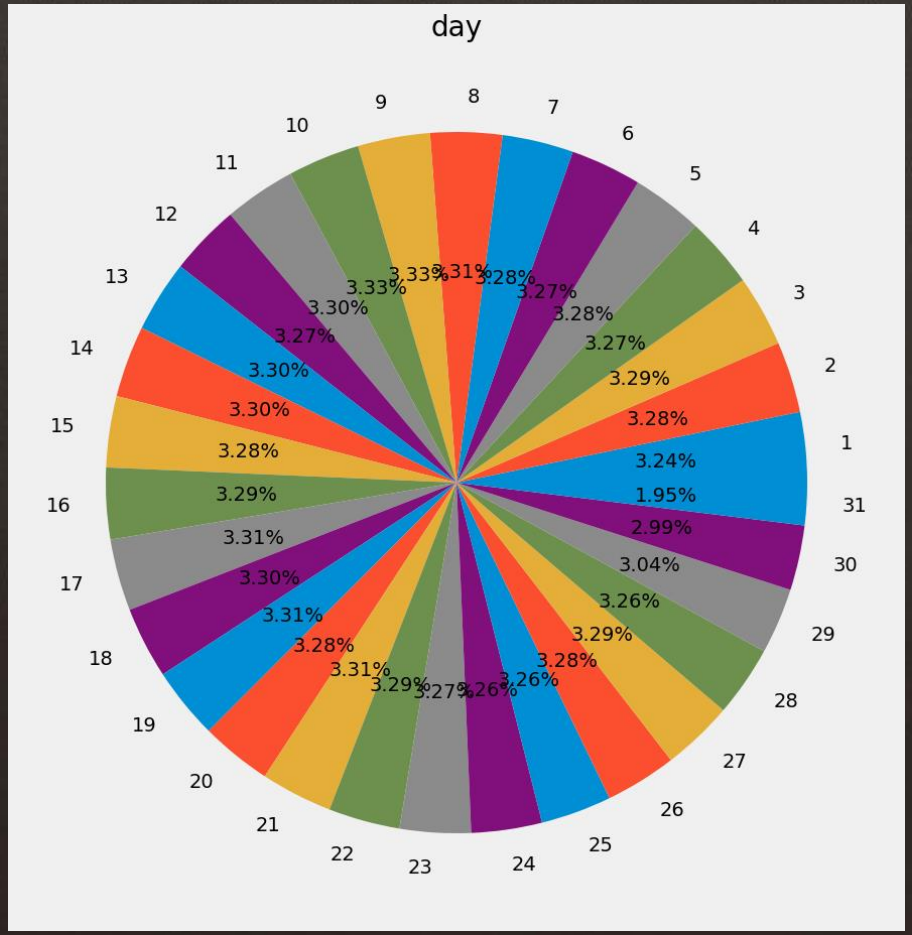

Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50

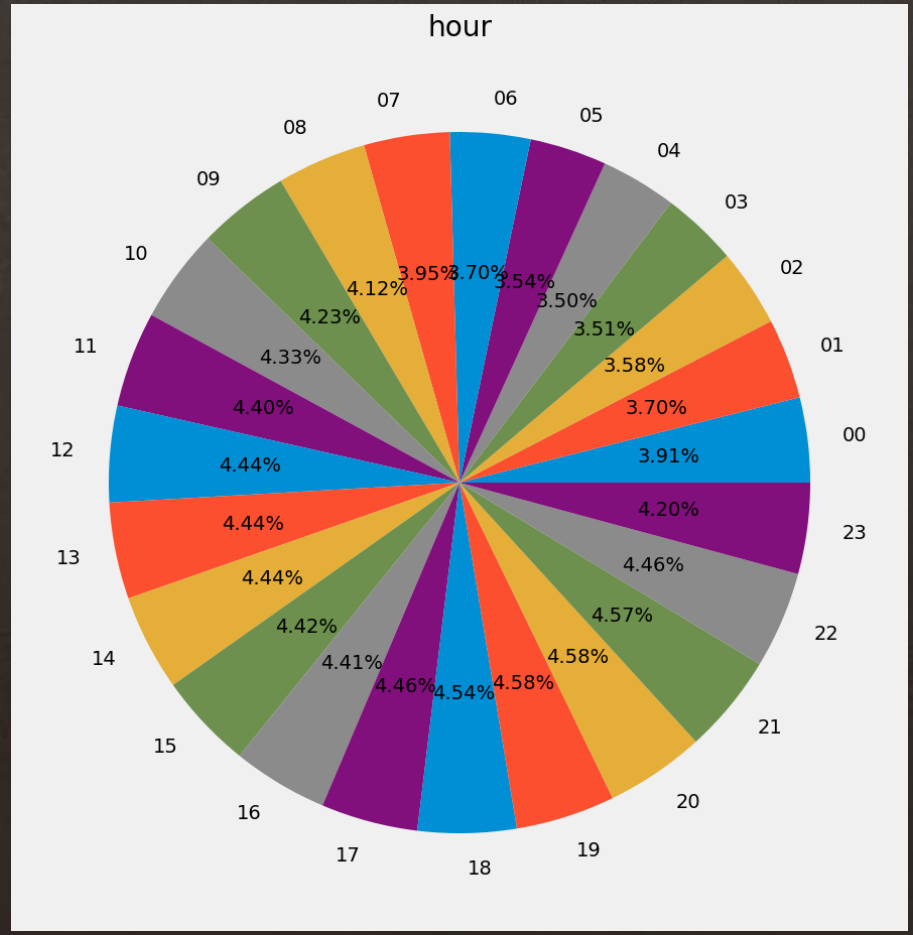
Pie Chart


year
2006
2005
2007
2004
2008
2003
2009
2002
2010
2018
2011
2017
2012
2016
2013
2015
2014
5.93%
6.52%
6.46%
6.17%
6.27%
6.09%
4.65%
5.81%
6.10%
3.66%
6.03%
6.04%
5.93%
6.10%
6.09%
6.05%
6.10%

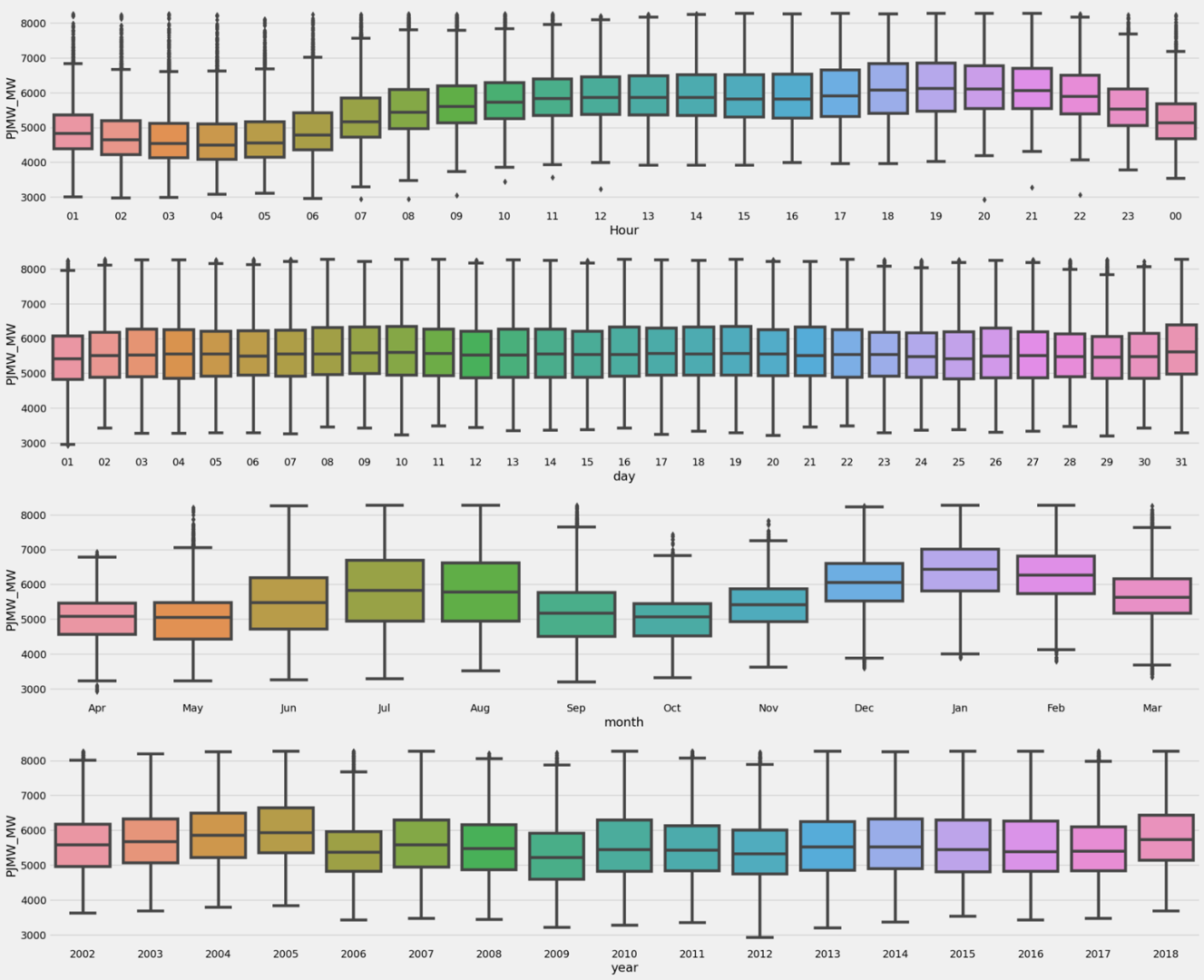

month
4
3
5
2
1
6
12
7
11
8
10
9
8.44%
9.03%
9.34%
8.63%
7.71%
9.18%
7.53%
8.45%
7.47%
8.43%
7.97%
7.81%

Pie Chart

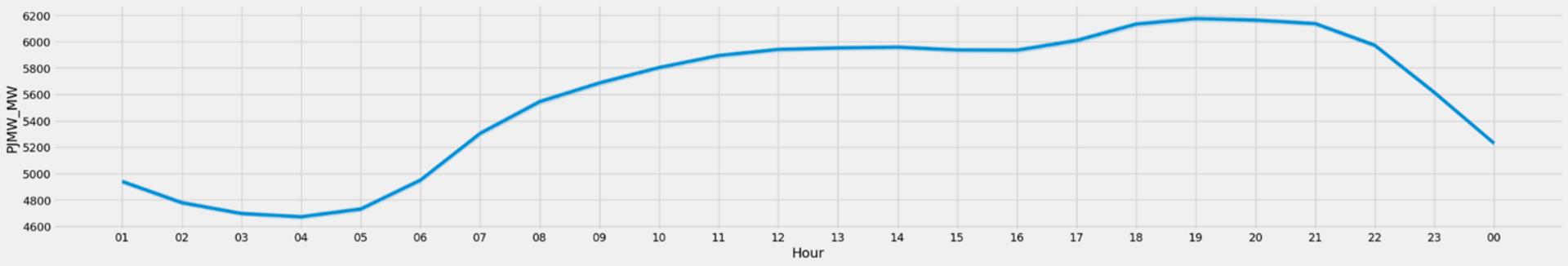

day
8
7
6
5
4
3
2
1
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
3.31%
3.28%
3.27%
3.28%
3.27%
3.29%
3.28%
3.24%
1.95%
2.99%
3.04%
3.26%
3.29%
3.28%
3.26%
3.27%
3.29%
3.31%
3.28%
3.30%
3.31%
3.30%
3.29%
3.28%
3.30%
3.30%
3.27%
3.30%
3.33%
3.33%

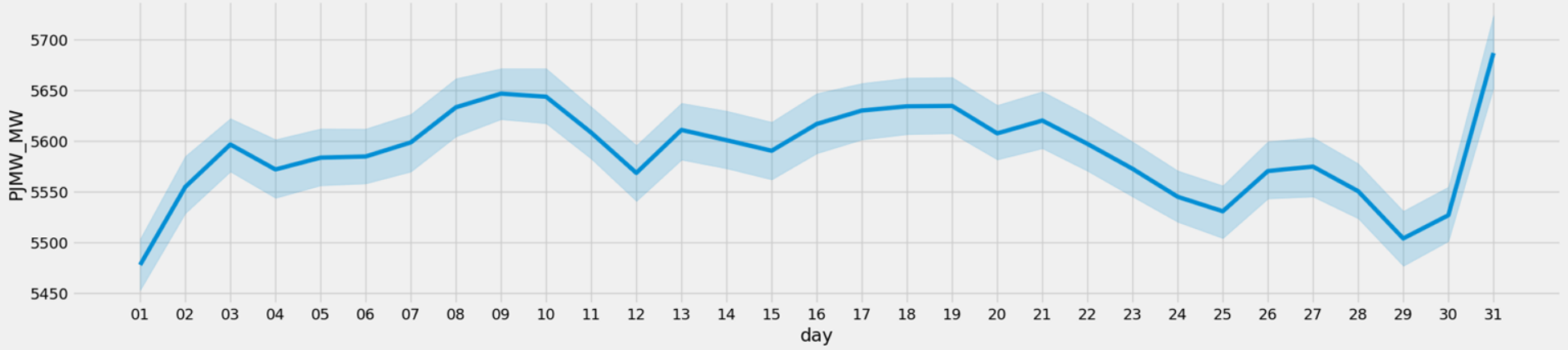

hour
00
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
3.91%
3.70%
3.58%
3.51%
3.50%
3.54%
3.70%
3.95%
4.12%
4.23%
4.33%
4.40%
4.44%
4.44%
4.44%
4.42%
4.41%
4.46%
4.54%
4.58%
4.58%
4.57%
4.46%
4.20%

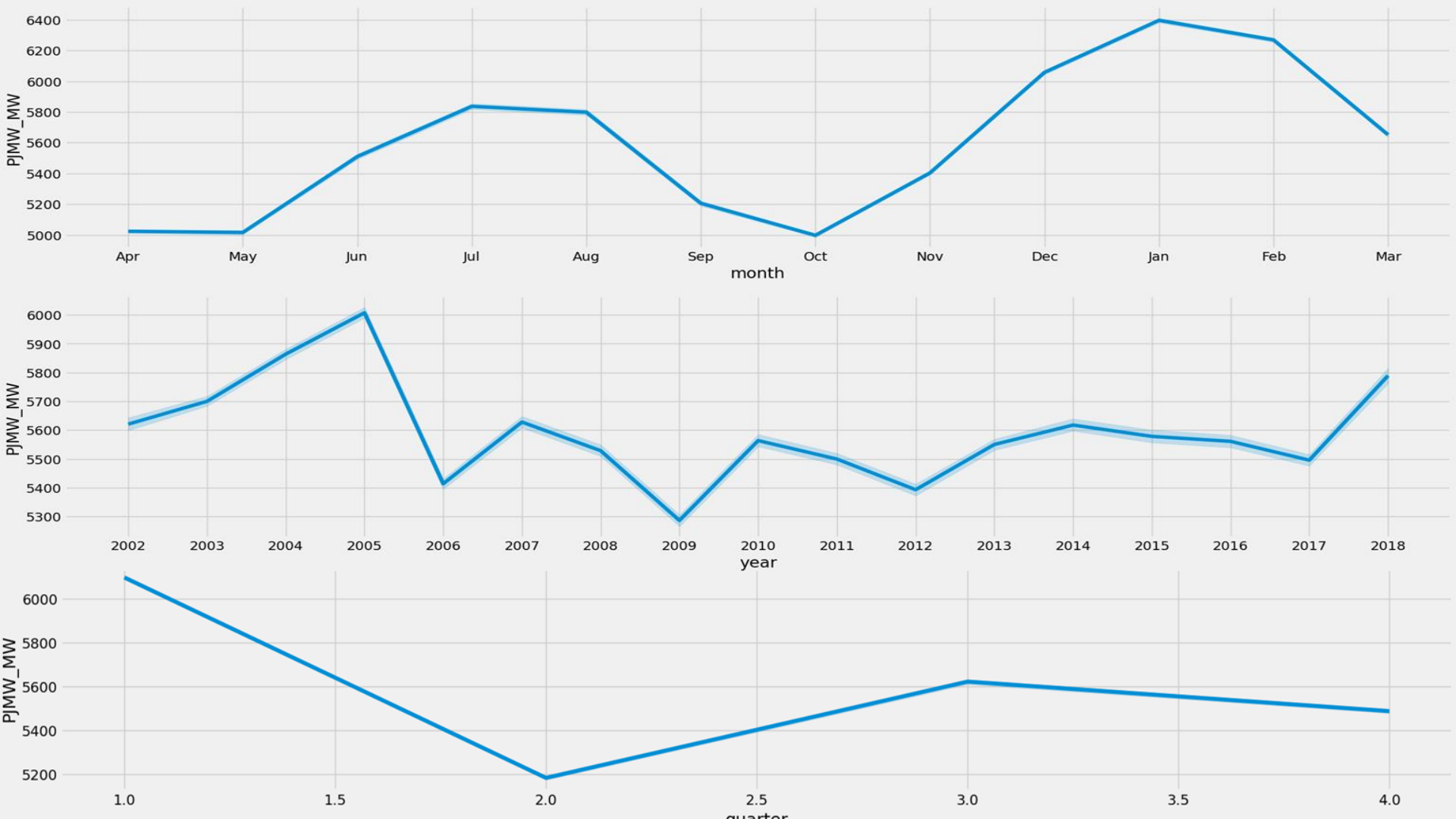
Box Plot


PJMW_MW
Hour
day
month
year
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar

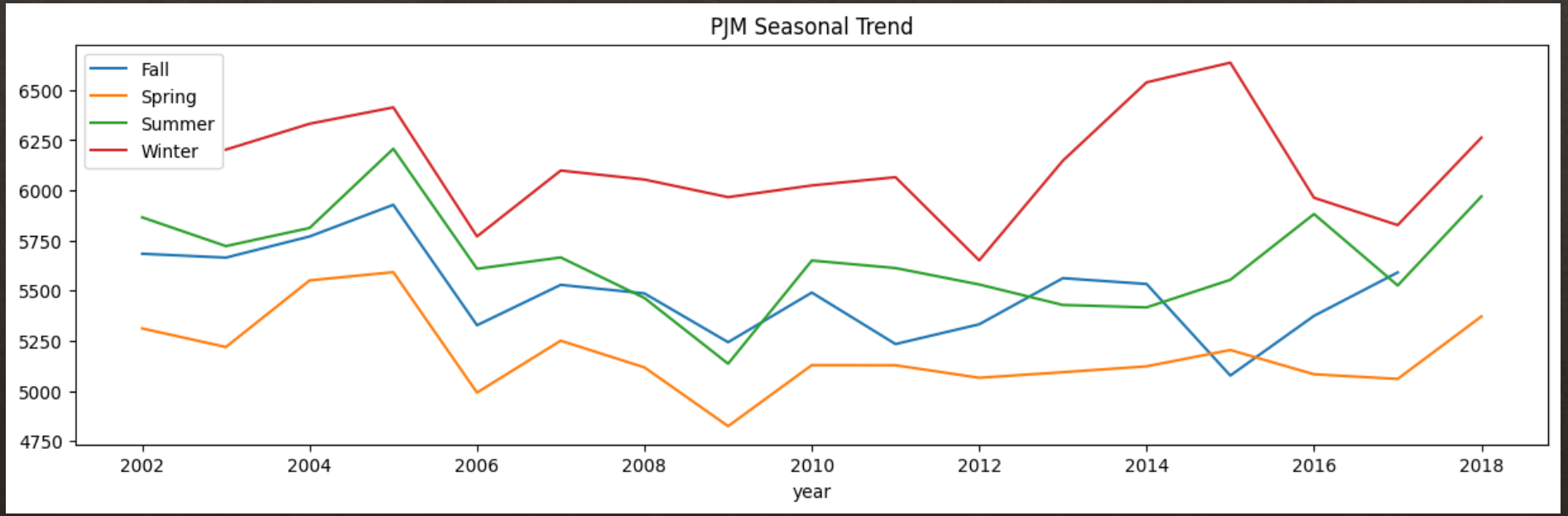
Line Plot


PJMW_MW
6200
6000
5800
5600
5400
5200
5000
4800
4600
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 00
Hour

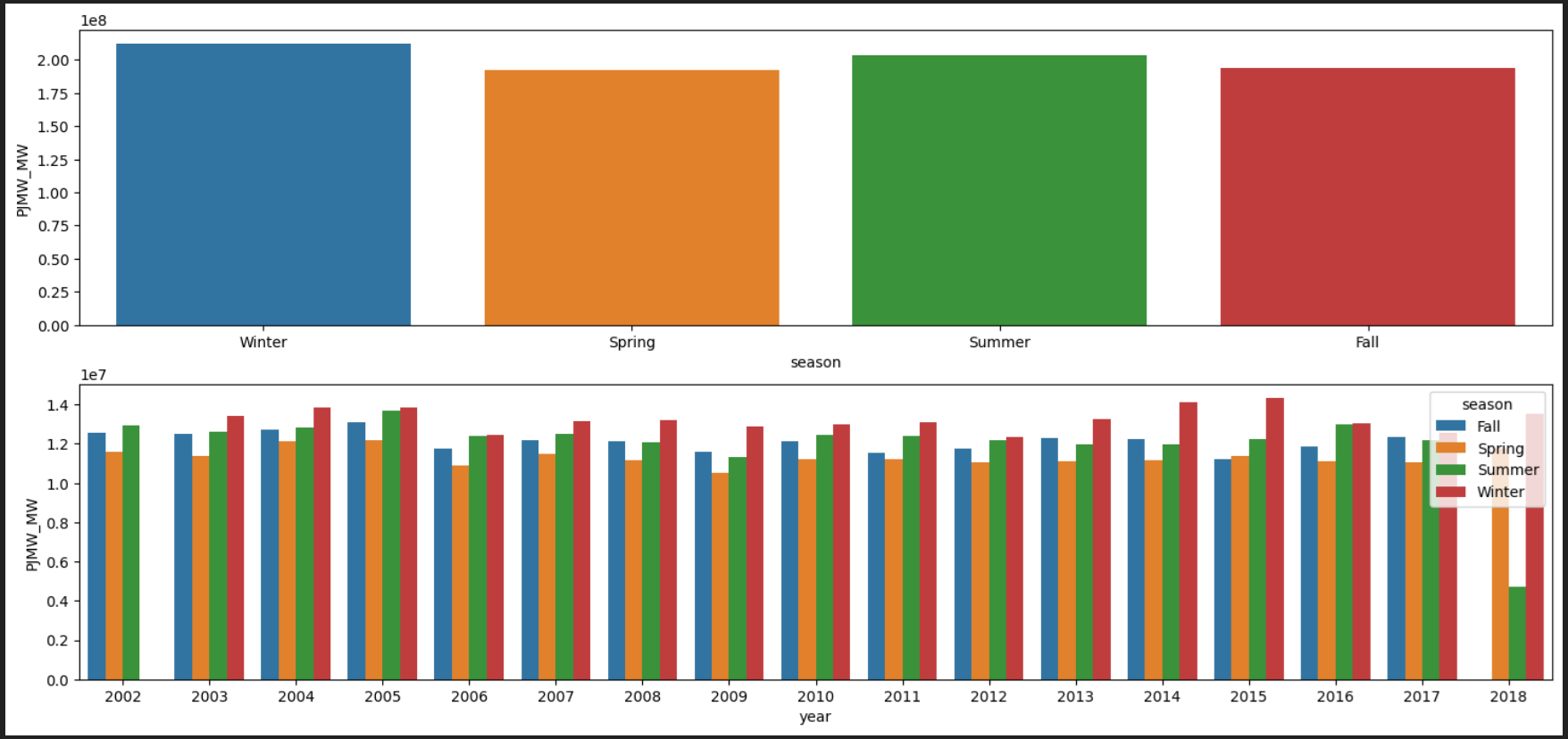

PJMW_MW
5700
5650
5600
5550
5500
5450
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
day


PJMW_MW
6400
6200
6000
5800
5600
5400
5200
5000
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
month
PJMW_MW
6000
5900
5800
5700
5600
5500
5400
5300
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
year
PJMW_MW
6000
5800
5600
5400
5200
1.0
1.5
2.0
2.5
3.0
3.5
4.0

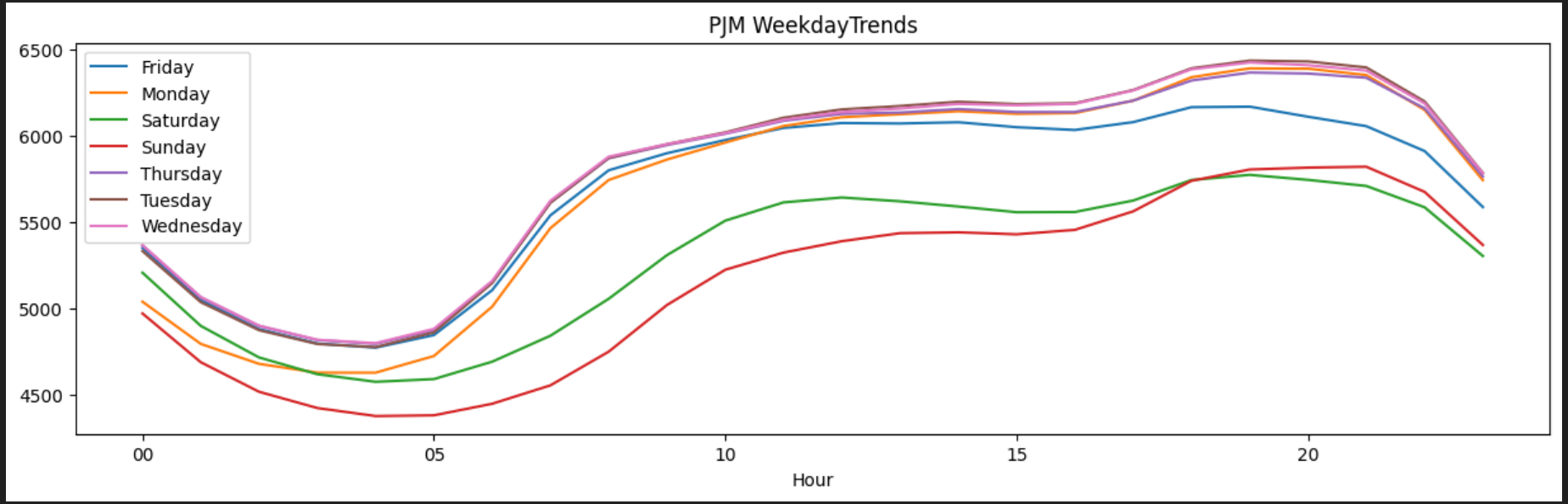
Seasonal Plot



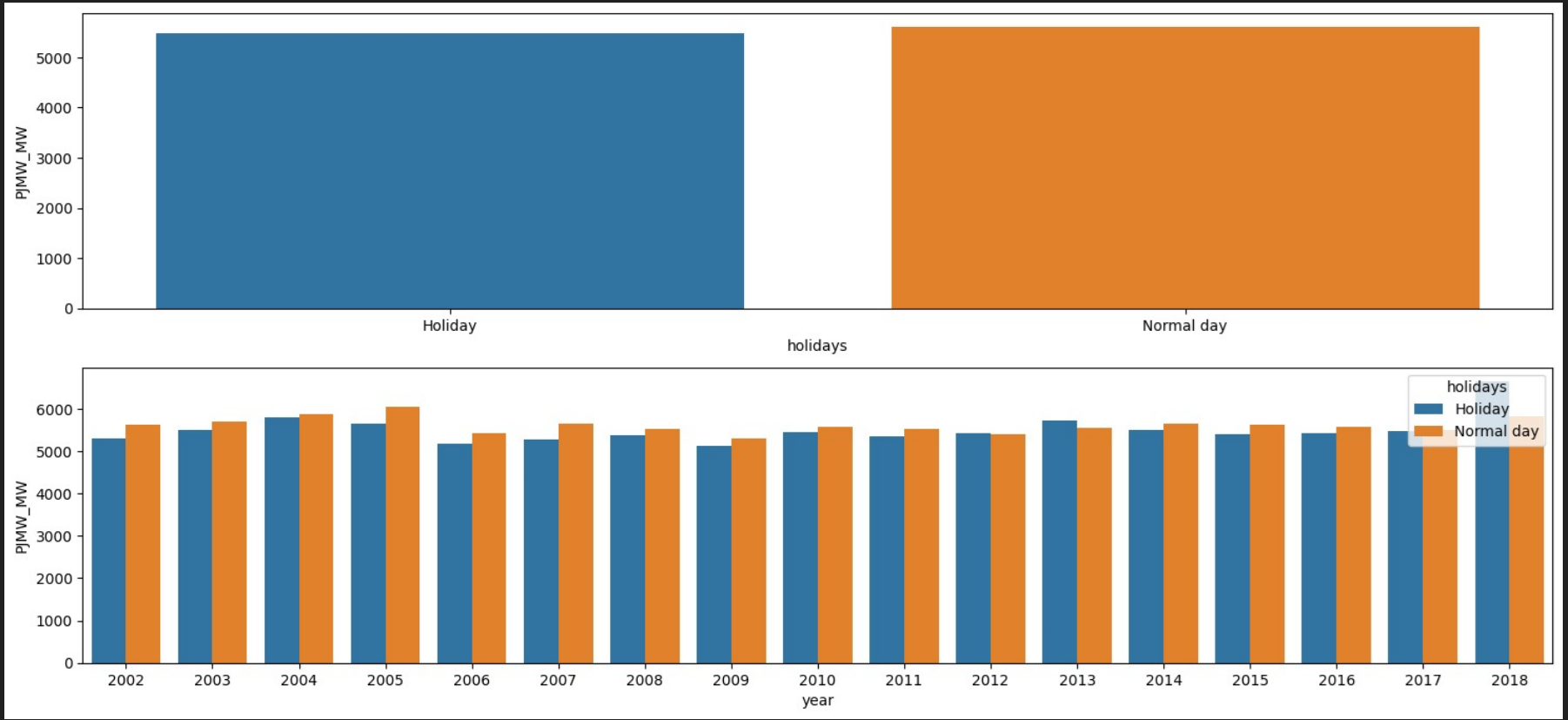
Seasonal Trend



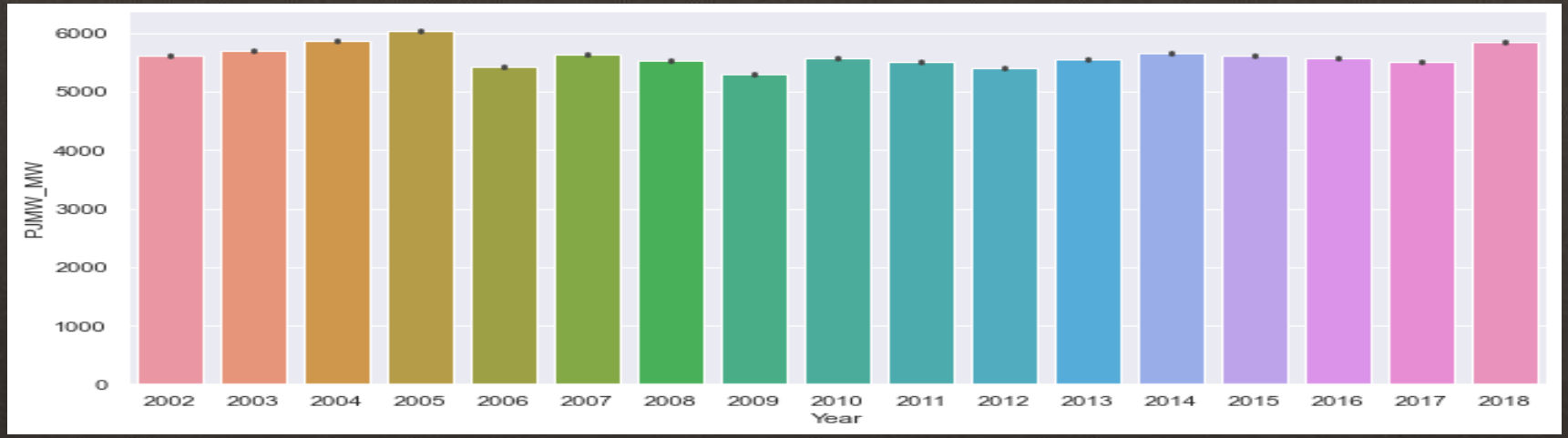
Weekday Trends

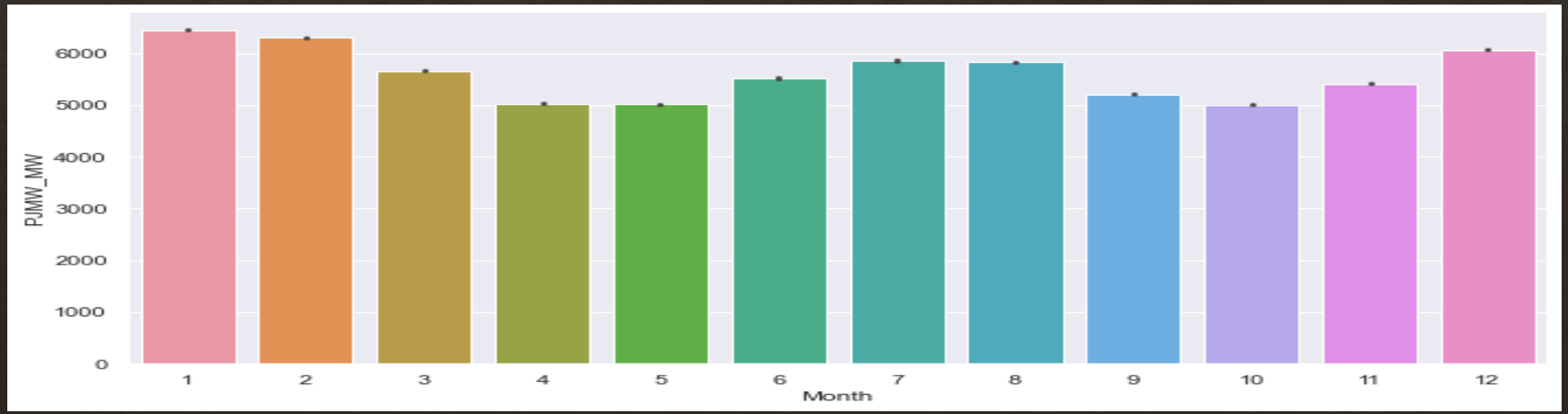


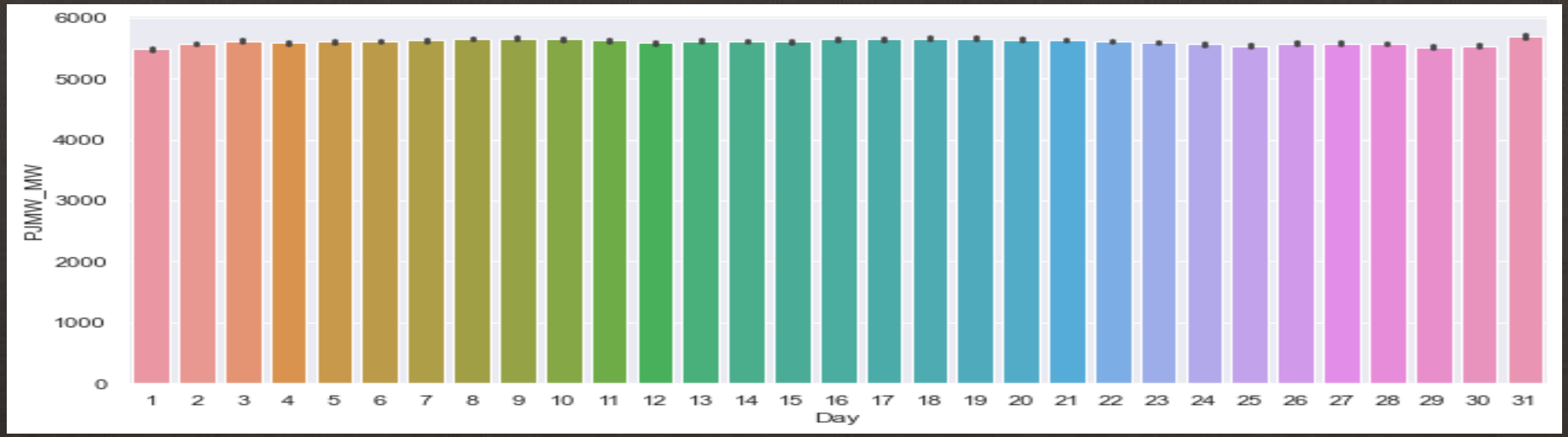
Trend Of Energy Consumption During Holidays

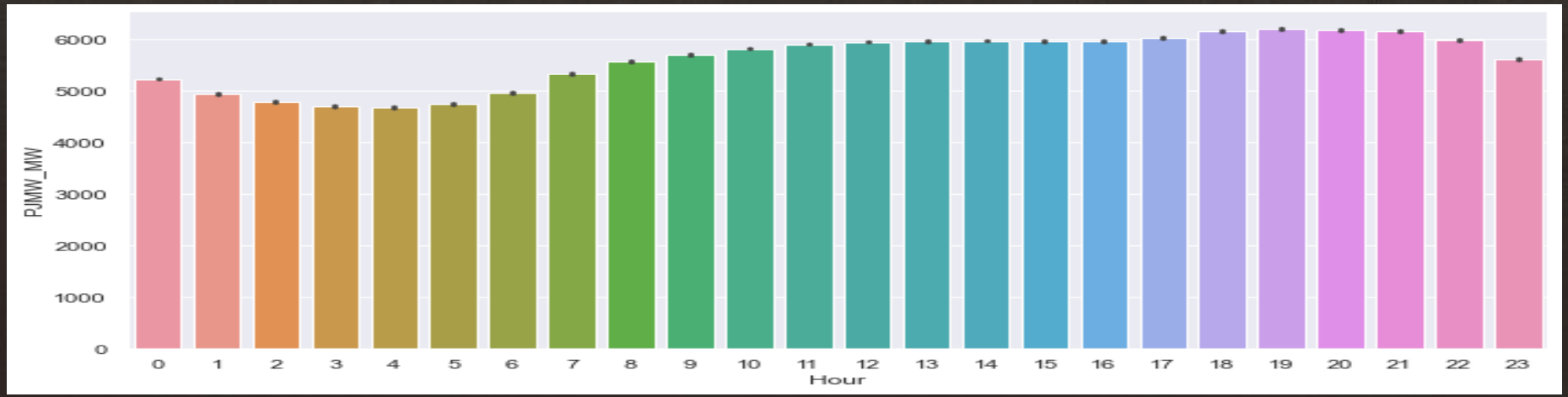


Cat Plots

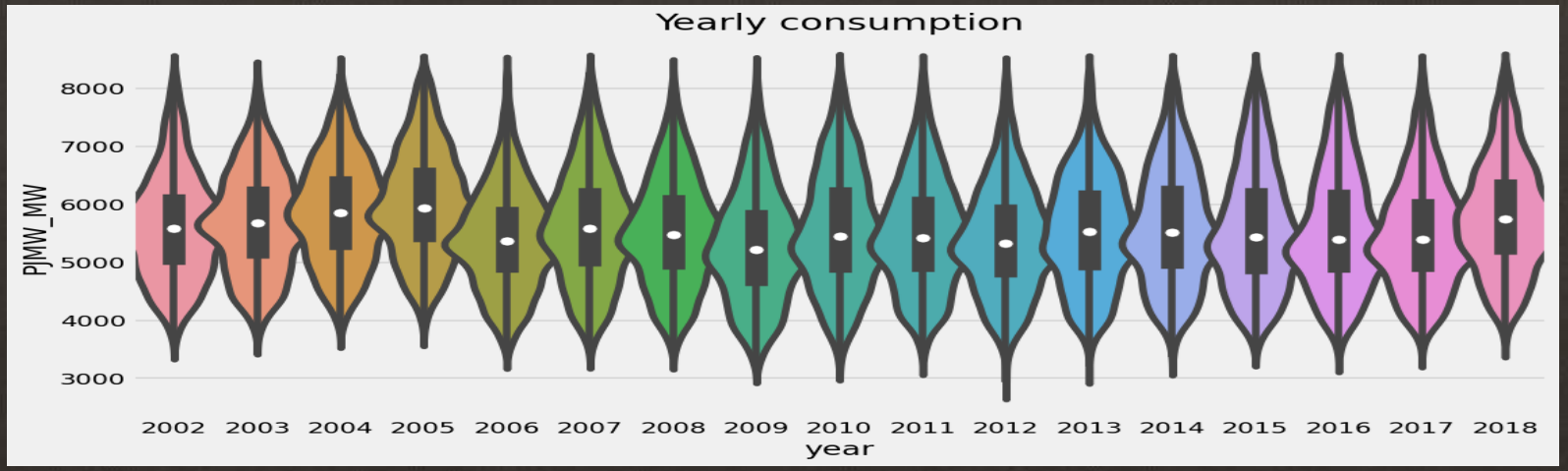

PJMW_MW
6000
5000
4000
3000
2000
1000
0
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Year

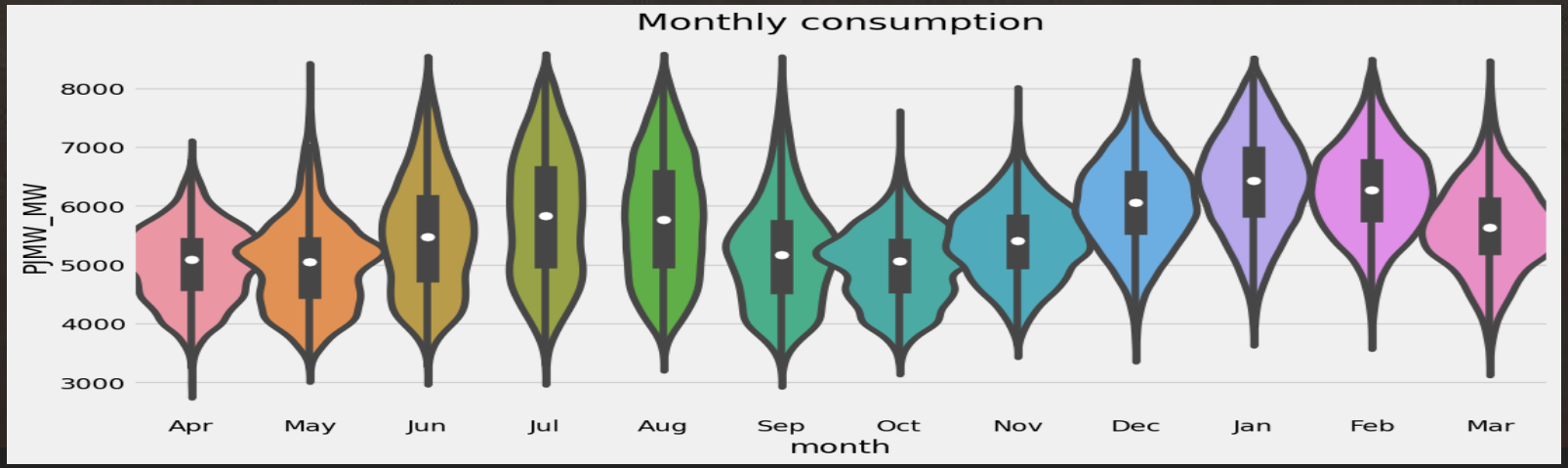

PJMW_MW
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12
Month

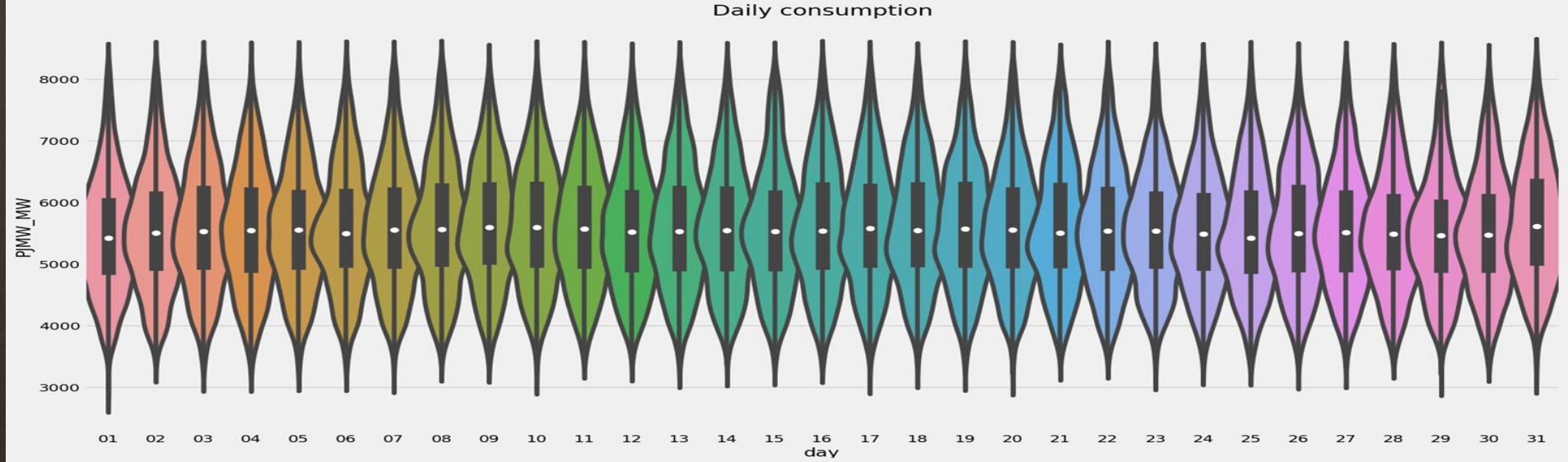

PJMW_MW
6000
5000
4000
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Day

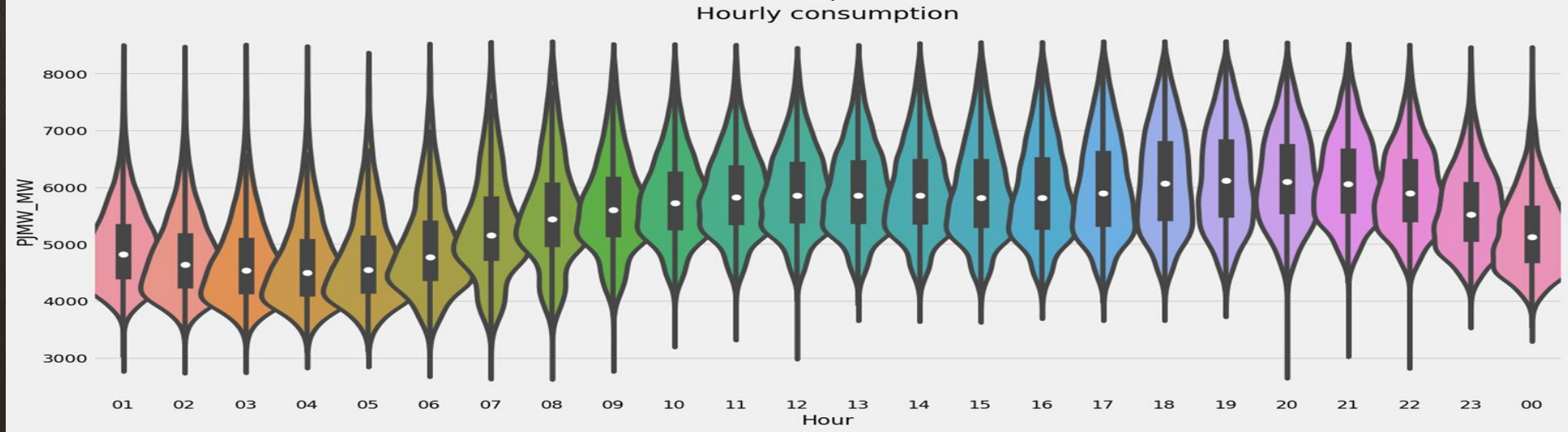

PJMW_MW
6000
5000
4000
3000
2000
1000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour

Violin Plot

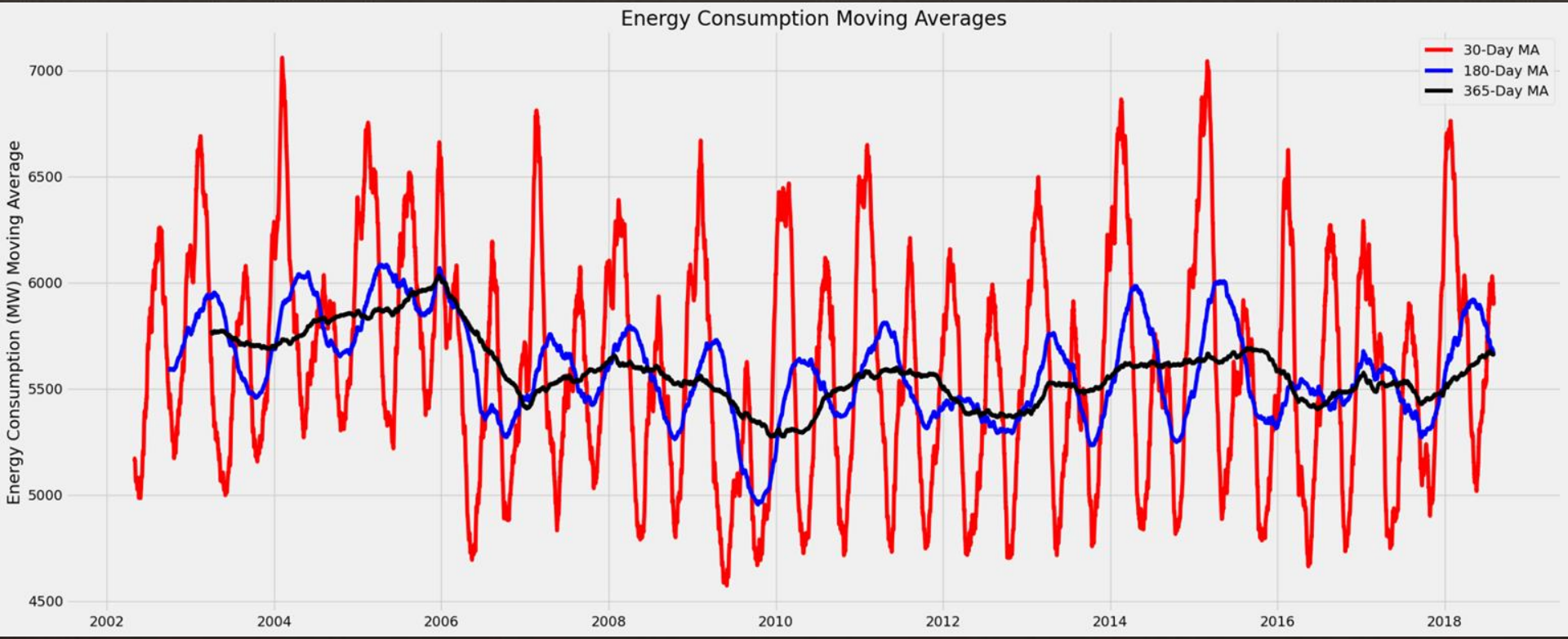

Yearly consumption
PJMW_MW
8000
7000
6000
5000
4000
3000
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
year


Monthly consumption
PJMW_MW
8000
7000
6000
5000
4000
3000
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
month

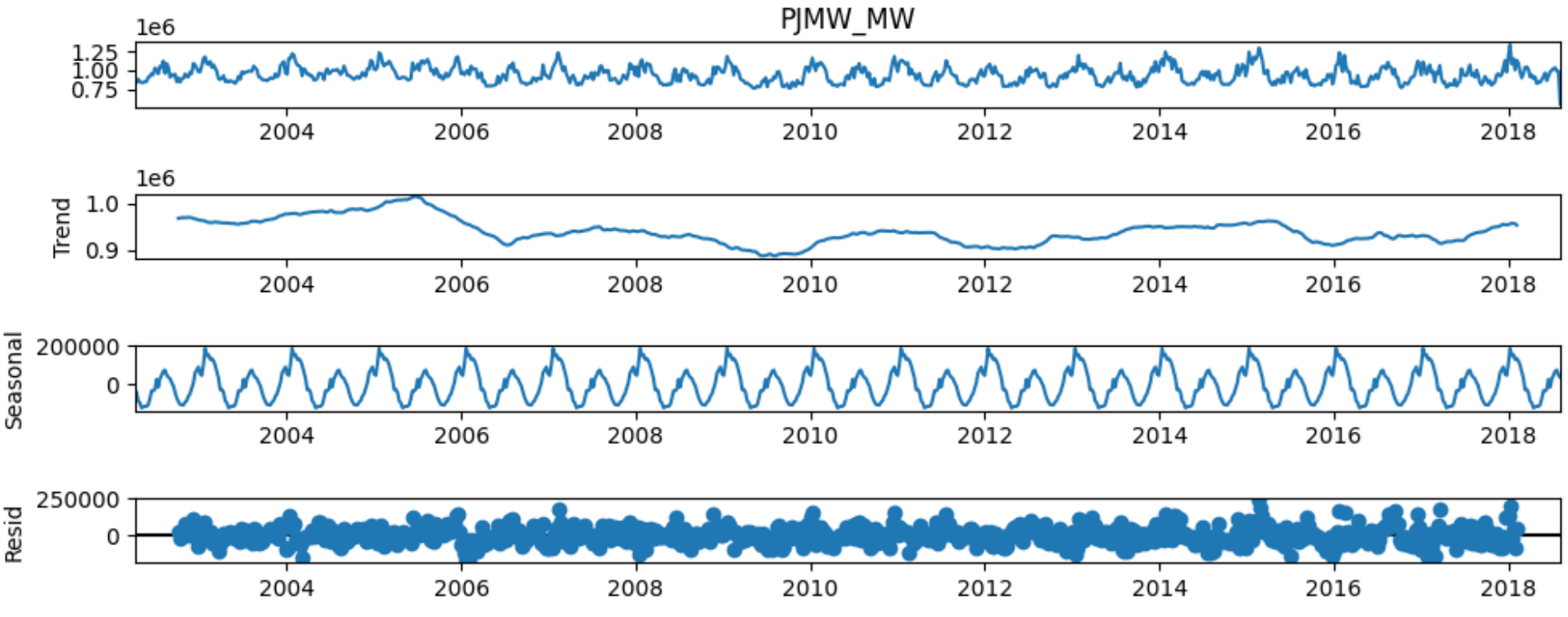

Daily consumption
PJMW_MW
8000
7000
6000
5000
4000
3000
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
day


Hourly consumption
PJMW_MW
8000
7000
6000
5000
4000
3000
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 00
Hour

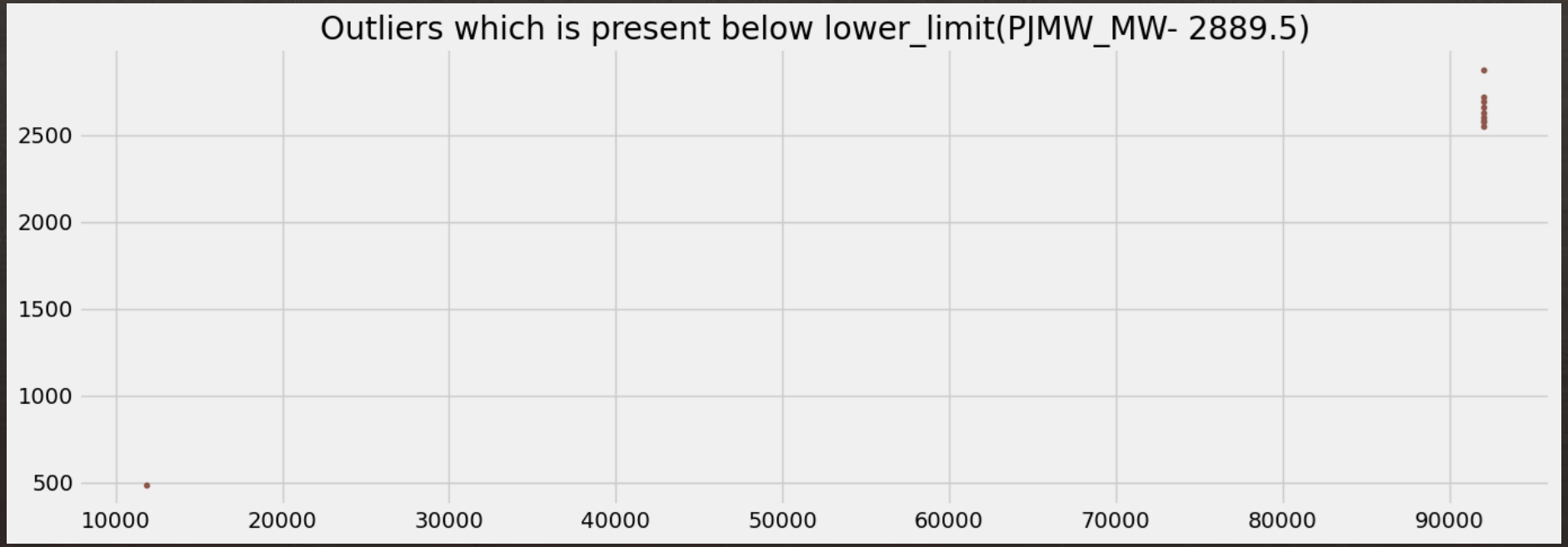
Moving Average

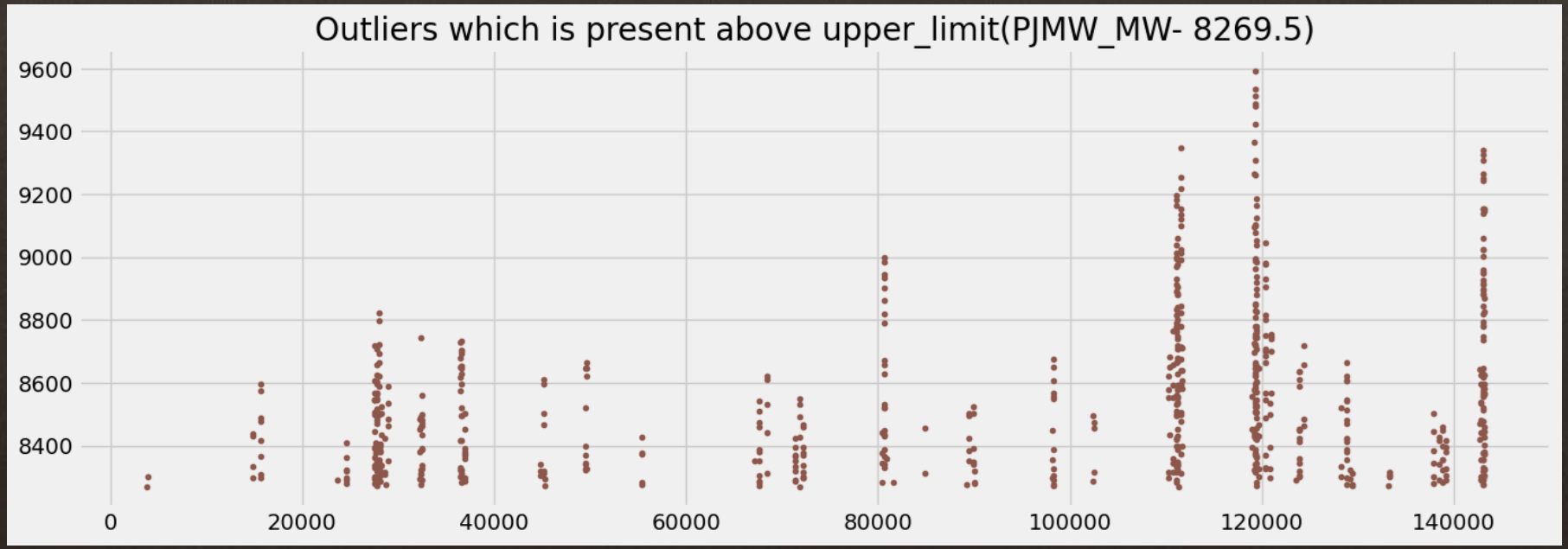


Time Series Decomposition



Outlier Detection




Outliers which is present above upper_limit(PJMW_MW- 8269.5)
9600
9400
9200
9000
8800
8600
8400
0
20000
40000
60000
80000
100000
120000
140000

Outlier Removal

```
In [28]: Q1= df.PJMW_MW.quantile(0.25)
         Q3= df.PJMW_MW.quantile(0.75)
         Q1, Q3

Out[28]: (4907.0, 6252.0)

In [29]: IQR= Q3- Q1
         IQR

Out[29]: 1345.0

In [30]: lower_limit= Q1 - 1.5*IQR
         upper_limit= Q3 + 1.5*IQR
         lower_limit, upper_limit

Out[30]: (2889.5, 8269.5)

In [31]: df[(df.PJMW_MW< lower_limit)|(df.PJMW_MW> upper_limit)]
         df.shape

Out[31]: (143206, 1)

In [32]: df= df[(df.PJMW_MW> lower_limit)&(df.PJMW_MW< upper_limit)]
         df.shape

Out[32]: (142503, 1)
```

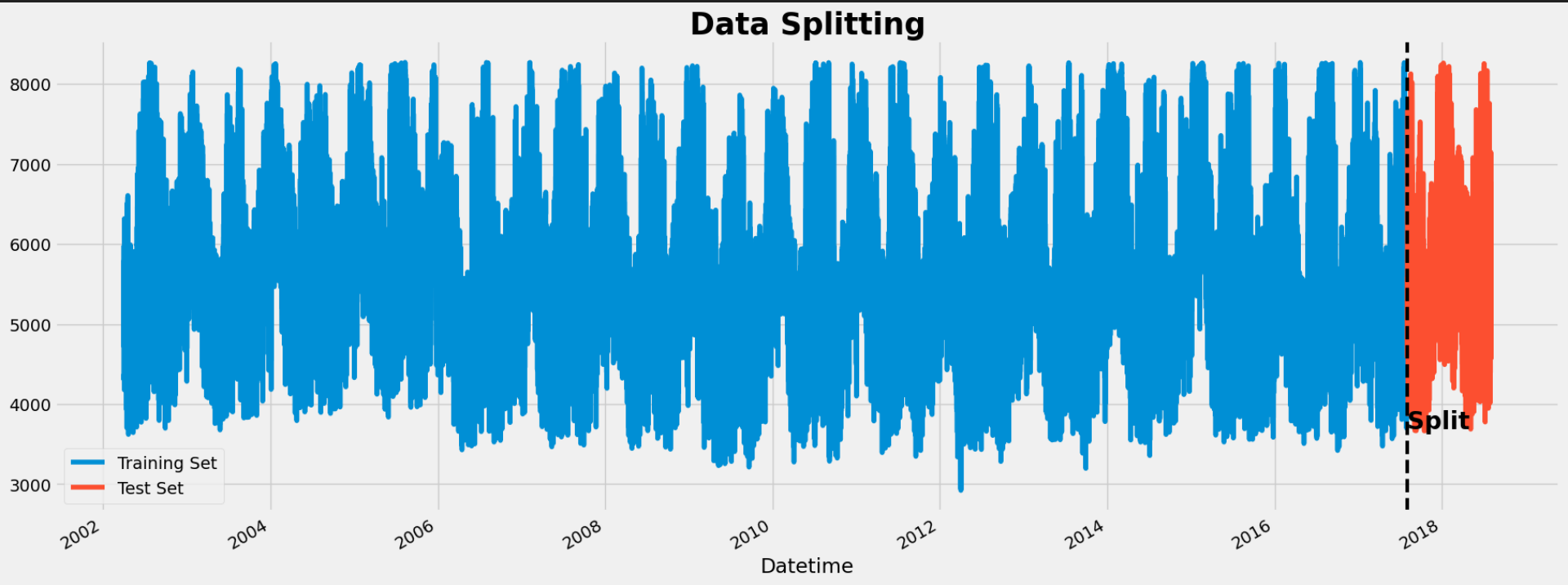
PHASE - 2

MODEL BUILDING

MODELS BUILT:

1. PROPHET MODEL
2. DEEP NEURAL NETWORK
3. XGBOOST
4. LINEAR REGRESSION
5. RANDOM FOREST

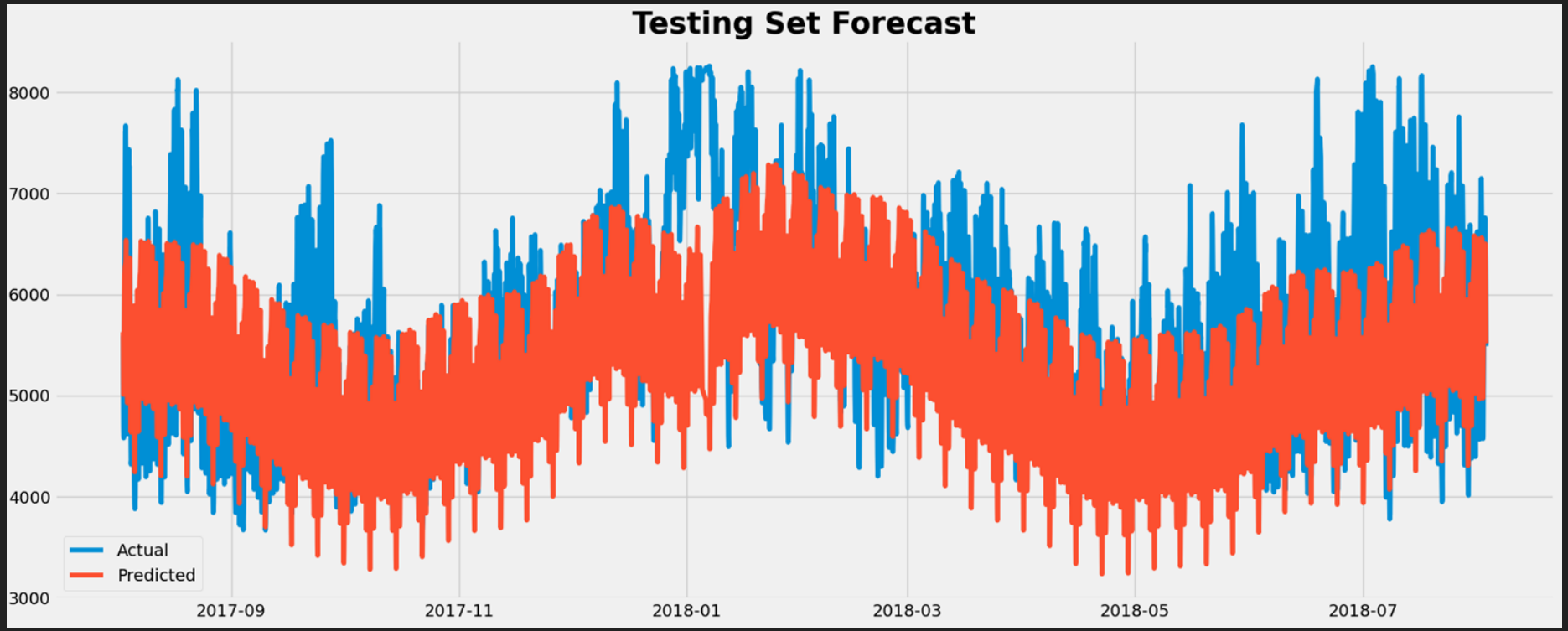
Splitting the data set




```
Train:    133852
Test:     8651
```

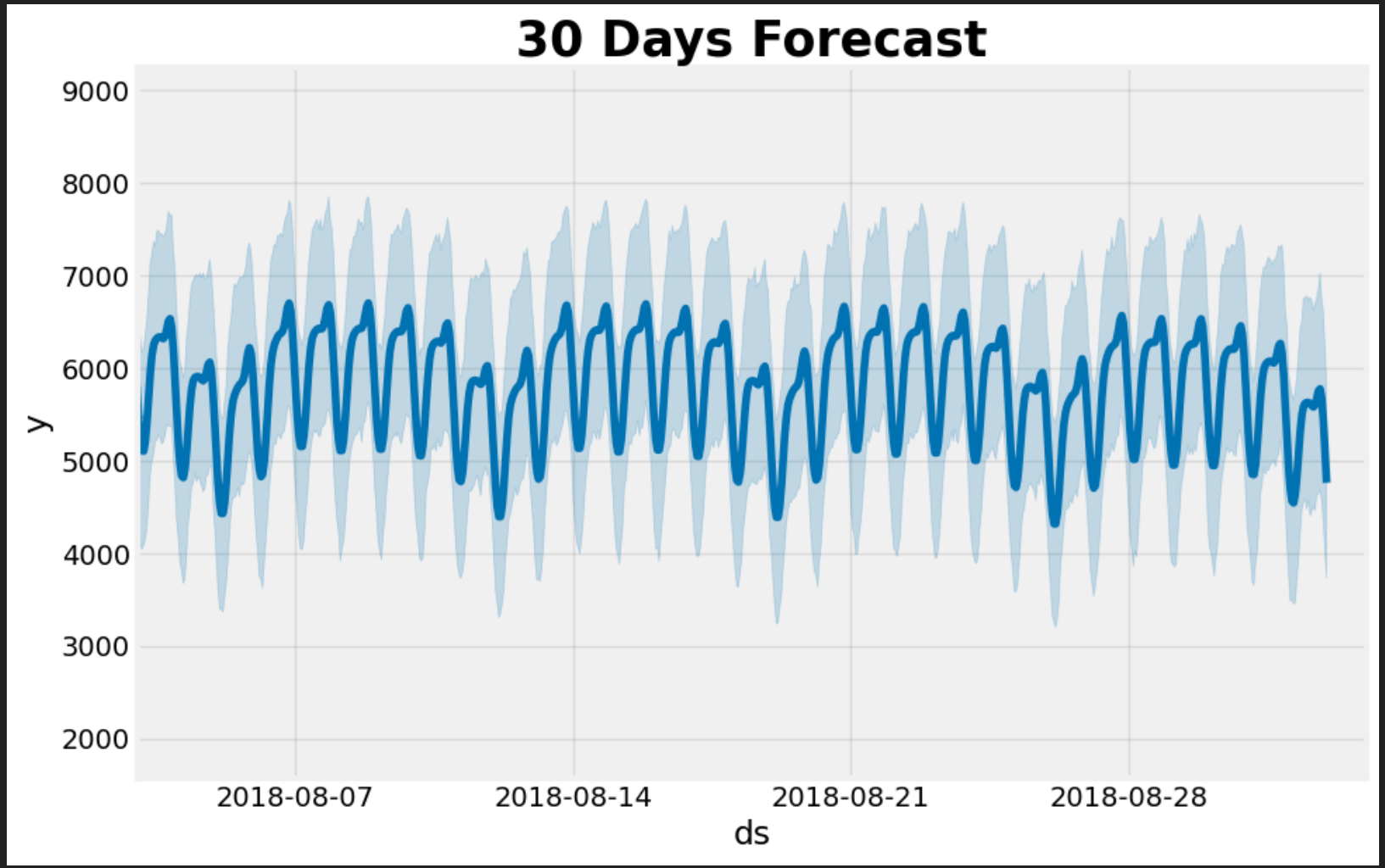
1. PROPHET MODEL

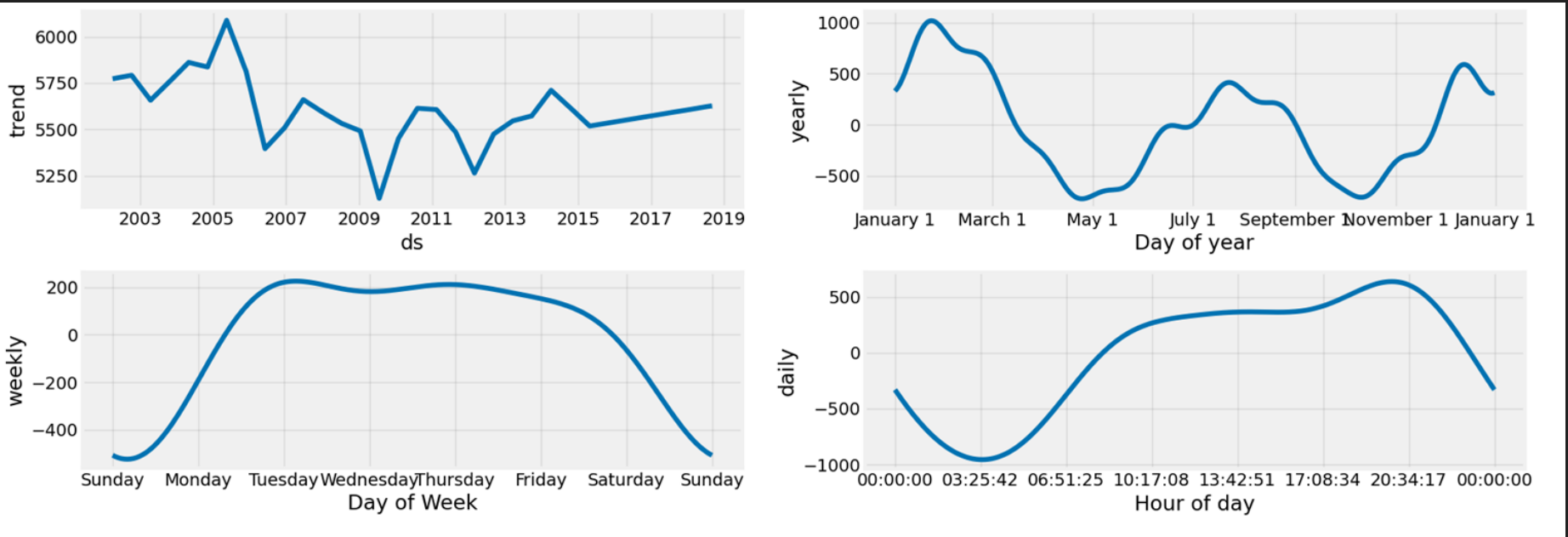
Prophet is an open-source library for univariate (one variable) time series forecasting developed by Facebook. It works best with time series that have strong seasonal effects and several seasons of historical data.



MAE: 545.605

RMSE Score on Test Set: 721.23

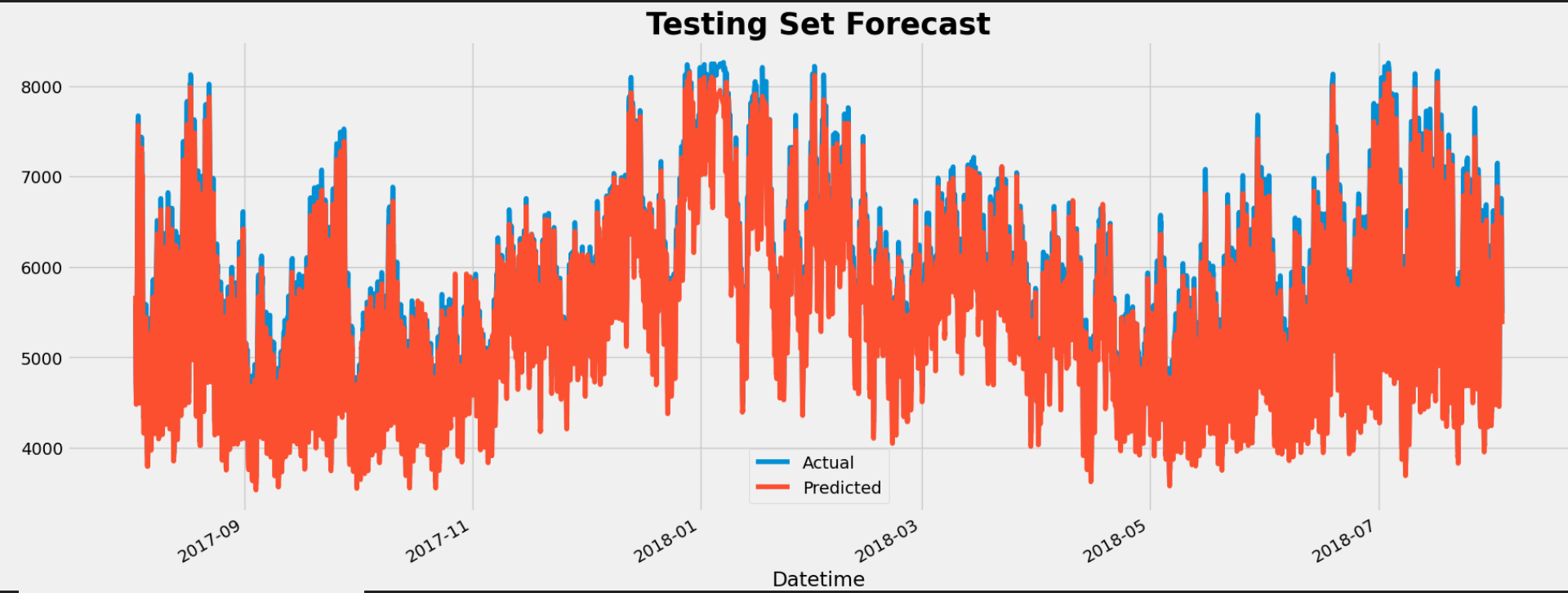

30 Days Forecast
9000
8000
7000
6000
5000
4000
3000
2000
y
2018-08-07
2018-08-14
2018-08-21
2018-08-28
ds


trend
6000
5750
5500
5250
2003
2005
2007
2009
2011
2013
2015
2017
2019
ds
yearly
1000
500
0
−500
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year
weekly
200
0
−200
−400
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of Week
daily
500
0
−500
−1000
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day

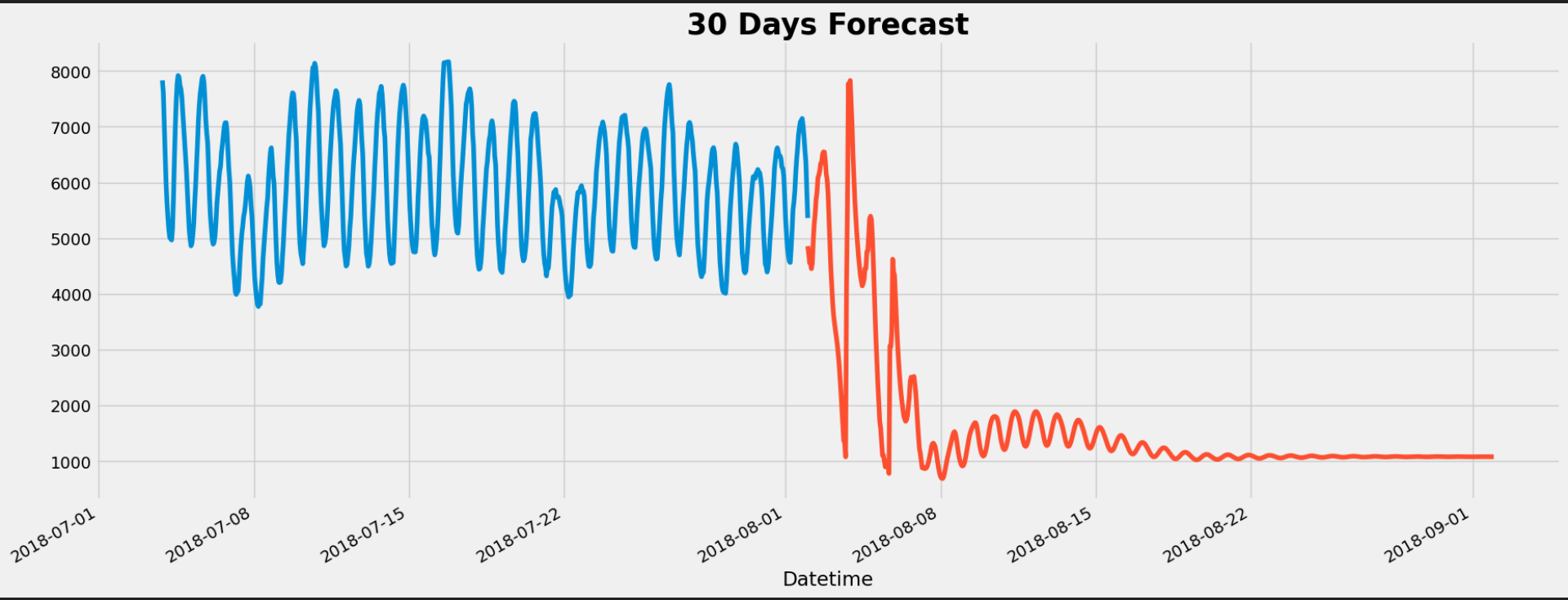
2. DEEP NEURAL NETWORK

The Deep Learning framework we are using is Tensorflow. Before feeding the data into Neural Network, we have to do some modification to the data so they can be accepted by the model.

We are going to use windowing technique which basically group the data into feature and label. The label will be the next value. You can take a look at the next few cells to give an idea what we are going to do


Testing Set Forecast
8000
7000
6000
5000
4000
Actual
Predicted
2017-09
2017-11
2018-01
2018-03
2018-05
2018-07
Datetime

MAE: 193.143



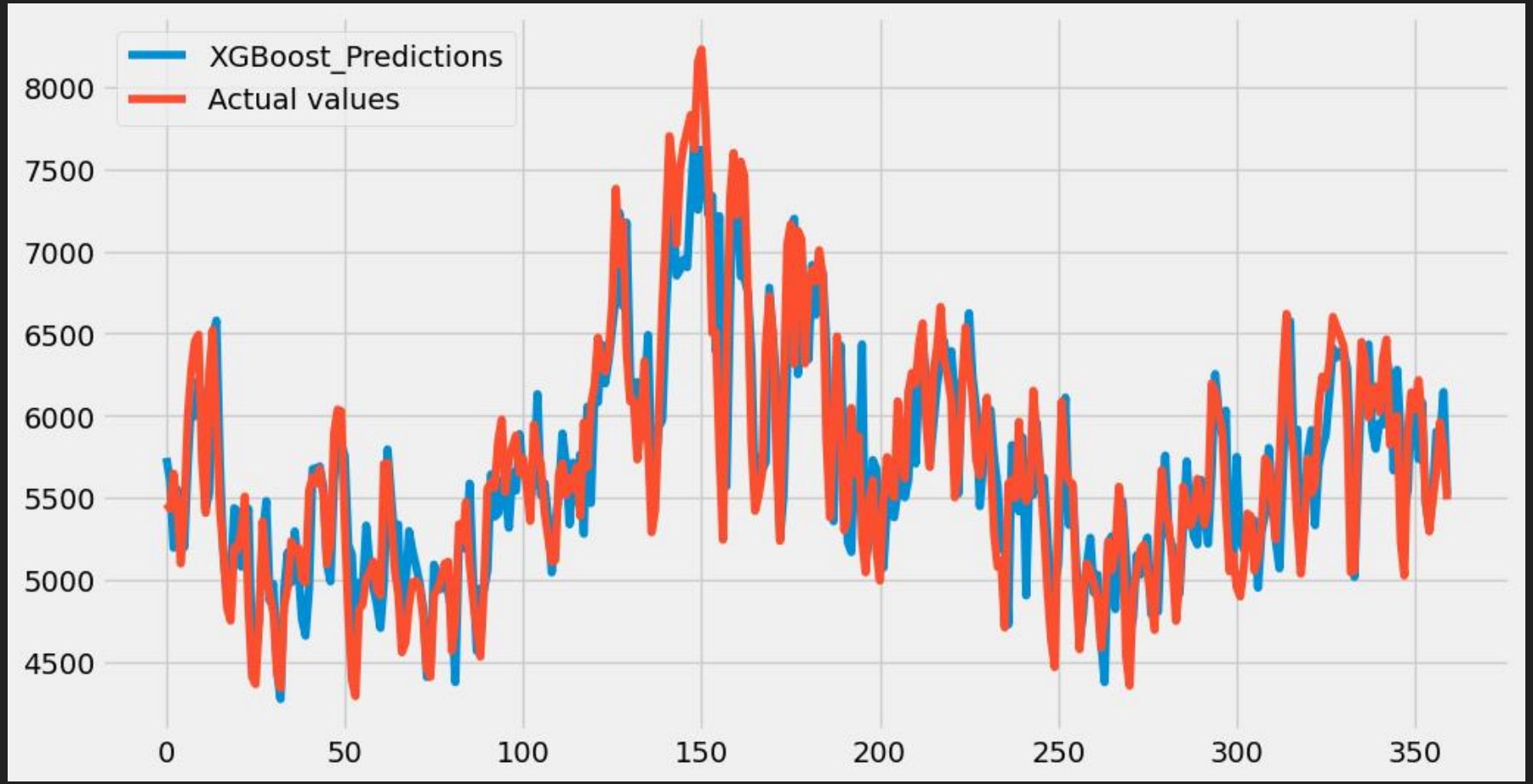
The Forecast of Deep Neural Network(DNN) model is not satisfactory.

3. XGBOOST MODEL

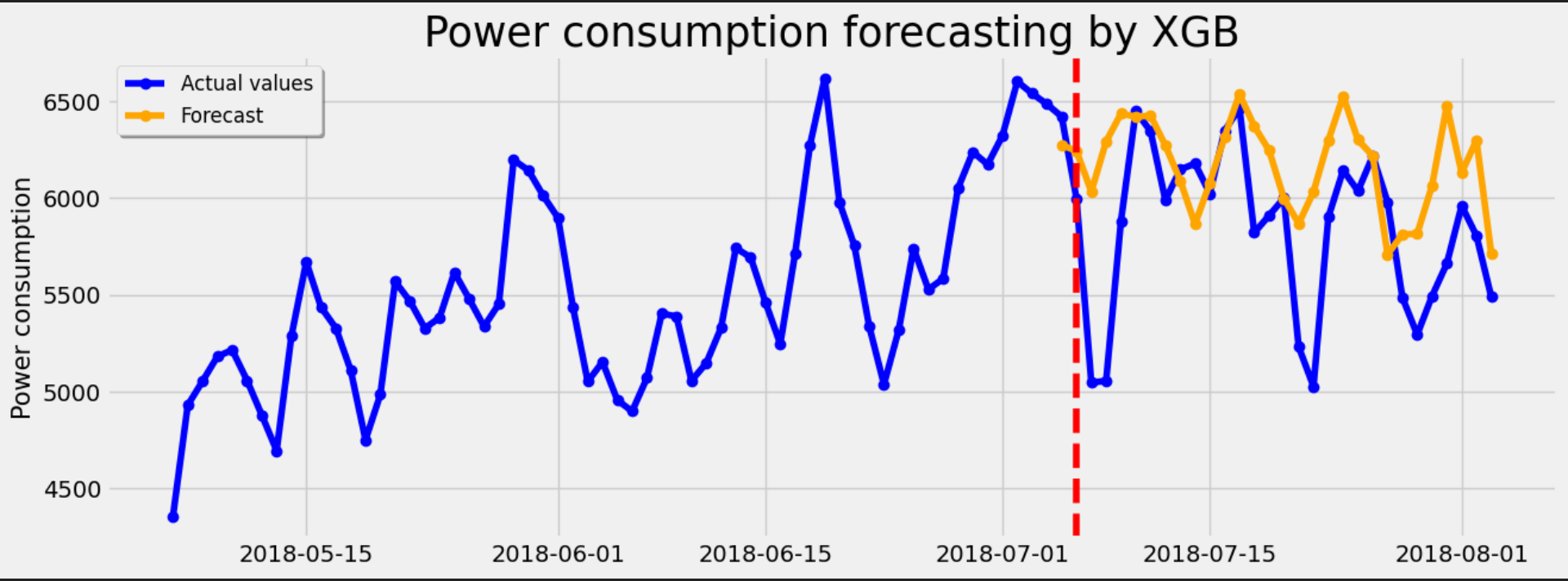
XGBoost is short for Extreme Gradient Boosting and is an efficient implementation of the stochastic gradient boosting machine learning algorithm. The stochastic gradient boosting algorithm, also called gradient boosting machines or tree boosting, is a powerful machine learning technique that performs well or even best on a wide range of challenging machine learning problems.

To be able to use XGBoost for time series forecasting, the data should be transformed into supervised learning before feeding it into the model.

Prediction has been done with test data



Forecasting



Predicted RMSE,MAE,MAPE

```
RMSE for XG Boost Model is: 353.39044761279615
MAE for XG Boos Model is: 270.524717518854
MAPE for XG Boos Model is: 4.708
```

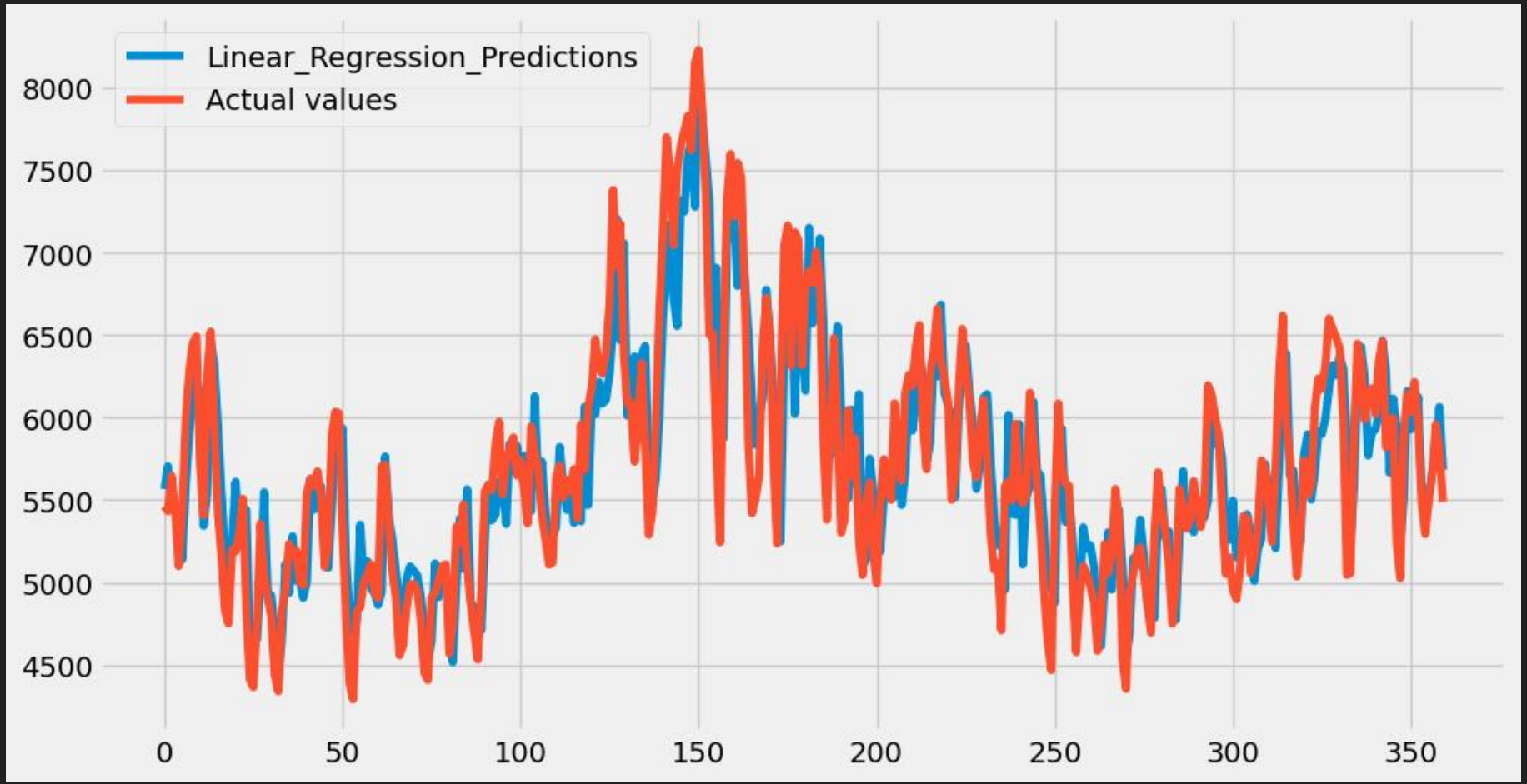
Forecasted RMSE,MAE,MAPE

```
RMSE for XGB forecast check is: 485.2427674054524
MAE for XGB forecast check is: 368.6199620225695
MAPE for XGB forecast check is: 6.668
```

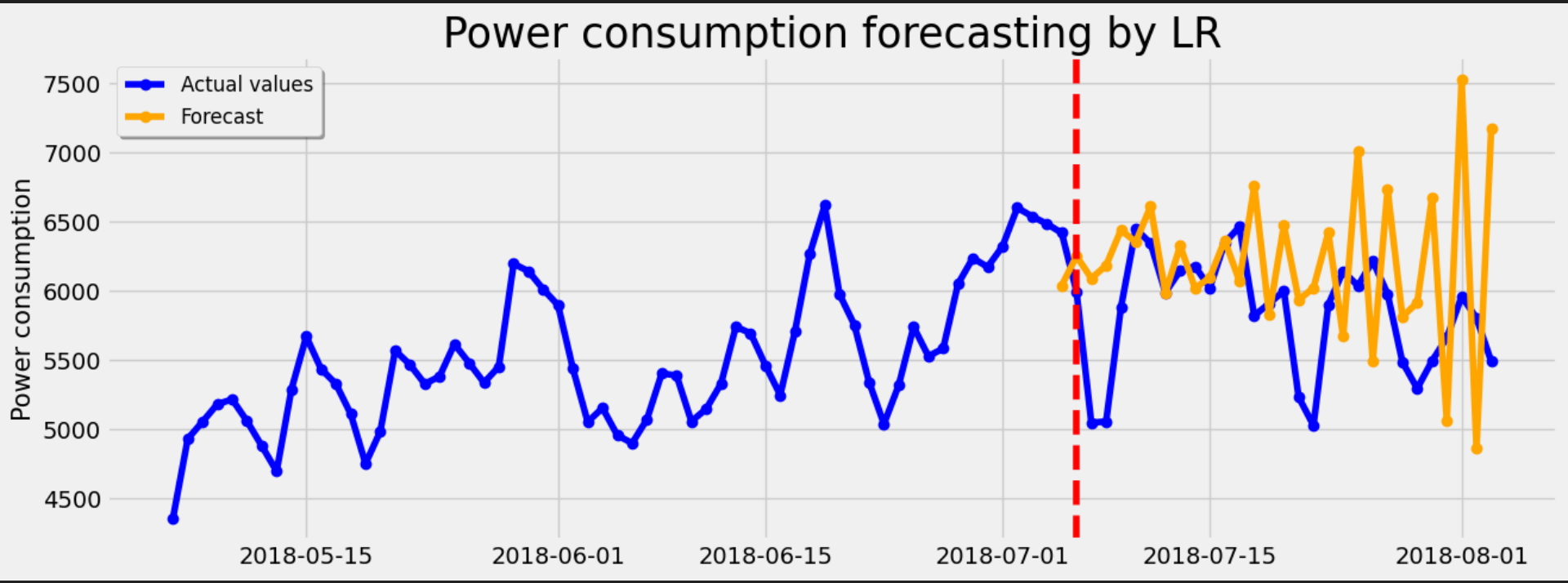
4. LINEAR REGRESSION

Linear regression is a quiet and the simplest statistical regression method used for predictive analysis in machine learning. Linear regression shows the linear relationship between the independent(predictor) variable i.e. X-axis and the dependent(output) variable i.e. Y-axis, called linear regression. If there is a single input variable X(independent variable), such linear regression is called *simple linear regression*.

Prediction has been done with test data



Forecasting



Predicted RMSE,MAE,MAPE

```
RMSE for Linear Regression Model is: 342.5745881638847
MAE for Linear Regression Model is: 269.31156194432845
MAPE for Linear Regression Model is: 4.703
```

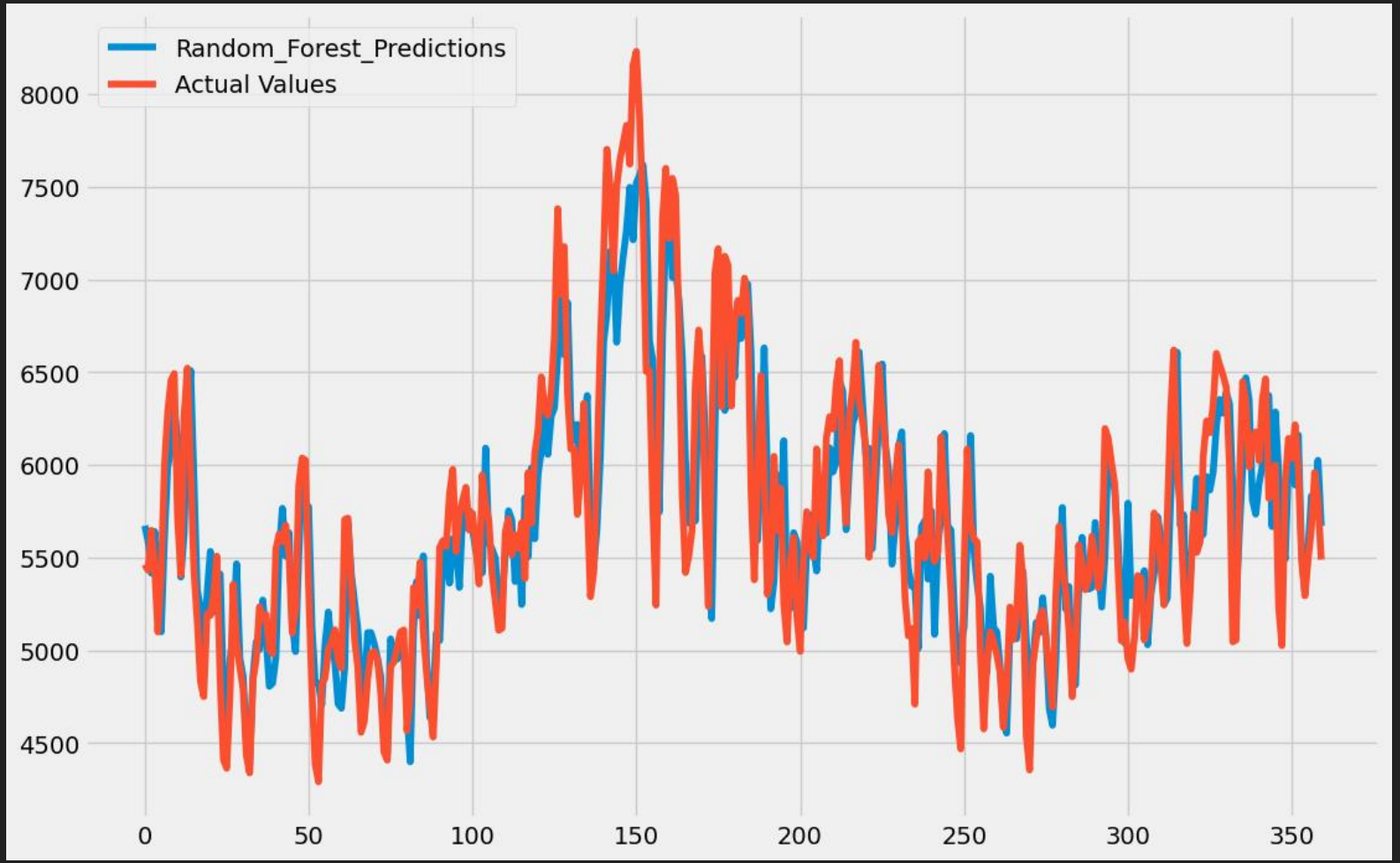
Forecasted RMSE,MAE,MAPE

```
RMSE for forecast check is: 748.1299433652413
MAE for forecast check is: 604.5203961263184
MAPE for forecast check is: 10.681
```

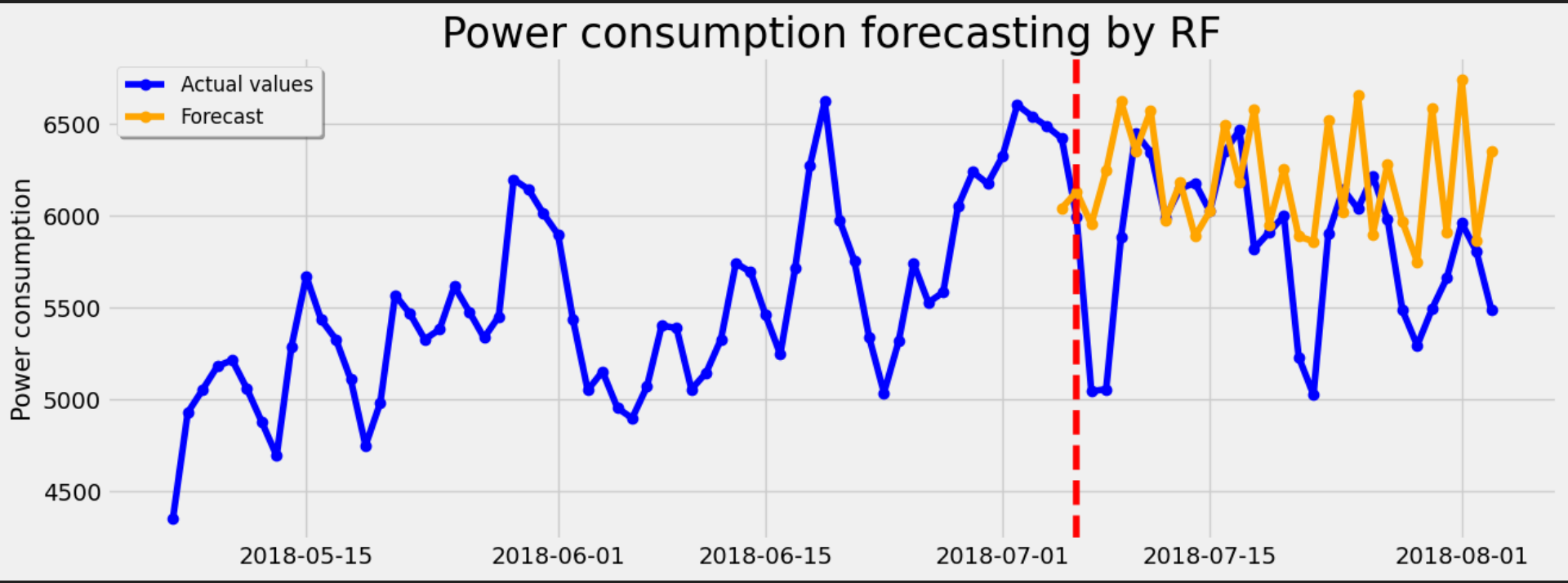
5. RANDOM FOREST

Random forest is a *Supervised Machine Learning Algorithm* that is *used widely in Classification and Regression problems*. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing *continuous variables,* as in the case of regression, and *categorical variables,* as in the case of classification. It performs better for classification and regression tasks.

Prediction has been done with test data



Forecasting



Predicted RMSE,MAE,MAPE

```
RMSE for Random Forest Model is: 346.4975114729343
MAE for Random Forest Model is: 273.41029119512467
MAPE for Random Forest Model is: 4.741
```

Forecasted RMSE,MAE,MAPE

```
RMSE for RF forecast check is: 546.8498682853016
MAE for RF forecast check is: 431.2064304612366
MAPE for RF forecast check is: 7.694
```

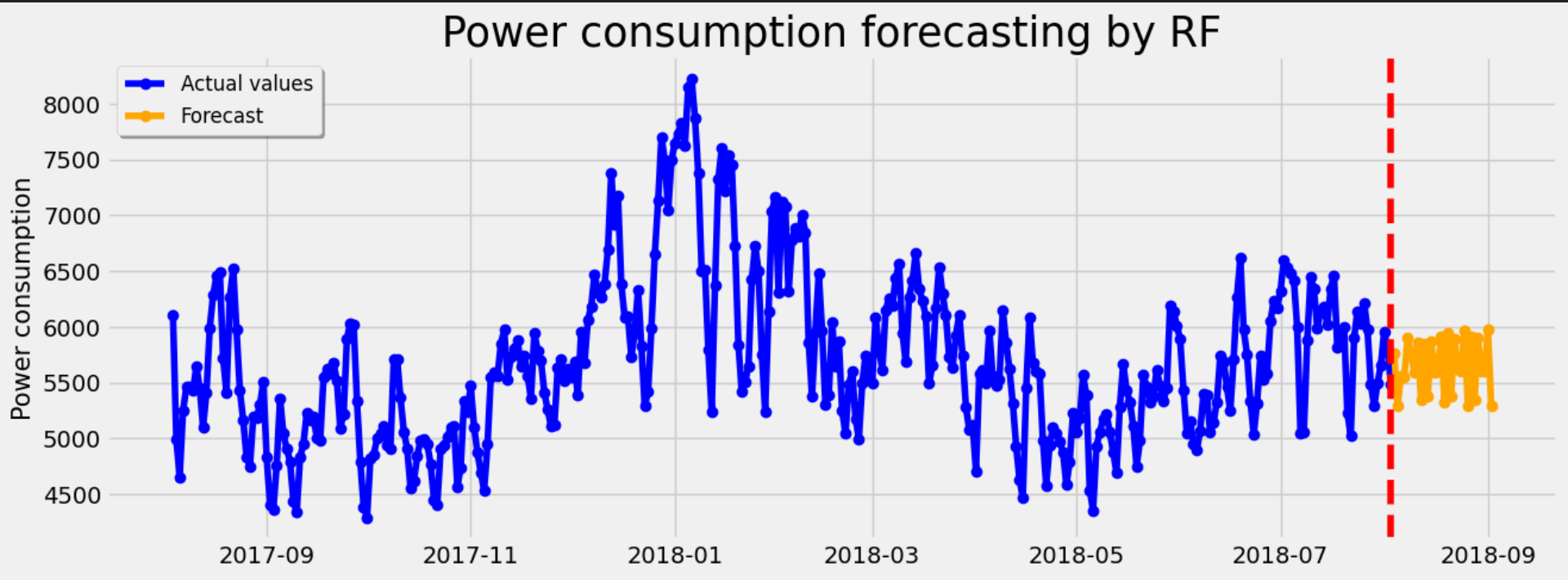
Findings

Based on our finding, we can see RANDOM FOREST model has less errors (RMSE, MAE, MAPE).

So forecasted power consumption for next 30 days from 2018-08-03 using RANDOM FOREST.

	future_energy
2018-08-04	5805.478750
2018-08-05	5250.396250
2018-08-06	5549.295942
2018-08-07	5510.736667
2018-08-08	5840.070000
2018-08-09	5759.415417
2018-08-10	5566.587138
2018-08-11	5990.728333
2018-08-12	5282.426250
2018-08-13	5849.890000
2018-08-14	5319.908333
2018-08-15	5782.970833
2018-08-16	5701.998333
2018-08-17	5539.374638
2018-08-18	6063.485417
2018-08-19	5293.225000
2018-08-20	5848.776667
2018-08-21	5237.801286
2018-08-22	5781.370000
2018-08-23	5712.072083
2018-08-24	5502.784167
2018-08-25	6005.816250
2018-08-26	5266.590833
2018-08-27	5932.142083
2018-08-28	5241.819583
2018-08-29	5781.027917
2018-08-30	5719.370833
2018-08-31	5523.631830
2018-09-01	6040.474583
2018-09-02	5221.987917

Forecasting for next 30 days



PHASE - 3

MODEL DEPLOYMENT

DEPLOYED MODEL :

The model is deployed on Streamlit framework and the Integrated Development Environment (IDE) used for its development was Spider.

Select the number of days to Forecast from 2018-Aug-4

Days

30 – +

Power consumption for 30th day

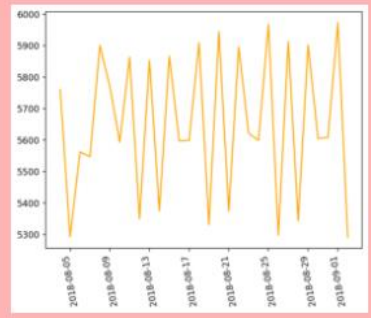
	Power Consumption
2018-09-02 00:00:00	5,290.475

Forecast power consumption data

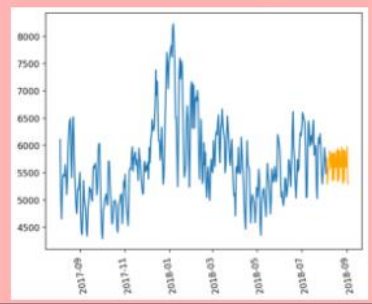
Power Consumption Forecasted for 30 days

	Power Consumption
2018-08-04 00:00:00	5,760.9104
2018-08-05 00:00:00	5,292.6054
2018-08-06 00:00:00	5,562.3671
2018-08-07 00:00:00	5,546.9054
2018-08-08 00:00:00	5,900.8813
2018-08-09 00:00:00	5,771.2787
2018-08-10 00:00:00	5,593.3129
2018-08-11 00:00:00	5,863.6356
2018-08-12 00:00:00	5,349.9837
2018-08-13 00:00:00	5,853.5342

Forecasted Graph



Actual Vs Forecast plot




Thank
You