ASSIGNMENT 2

ANDREW ID: parmenin

18-787: Data Analytics

2/13/23

Niyomwungeri Parmenide ISHIMWE

I, the undersigned, have read the entire contents of the syllabus for course 18-787 (Data Analytics) and agree with the terms and conditions of participating in this course, including adherence to CMU's AIV policy.

Signature: **Niyomwungeri Parmenide ISHIMWE**
Andrew ID: **parmenin**

Full Name: **Niyomwungeri Parmenide ISHIMWE**

LIBRARIES USED

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- from scipy import stats
- import statistics
- import warnings
- import statsmodels.api as sm
- from statsmodels.graphics.tsaplots import plot_acf
- from statsmodels.tsa.arima.model import ARIMA
- from sklearn.metrics import mean_absolute_error as mae
- from arch.unitroot import ADF, VarianceRatio

QUESTION 1:

It was asked to load into the environment (Jupyter notebook), and the intraday onshore wind power generation data was measured every hour for one year and it was done using the read_csv function from pandas. After loading them, the “Date” column is changed from string to datetime using the pandas’ function to_datetime; and the “Time” column is converted from integers into time delta or hours using the to_timedelta function from pandas and then the time delta hours are appended to date time in the “Date” column to form dates with hours as well. After that, checking for empty and Nan fields is done, and filled later on using linear interpolation with interpolate () function from pandas [1]. This formed date is used alongside the "Wind Generation" column to plot the hourly wind generation against the timestamps where the following graph is generated.

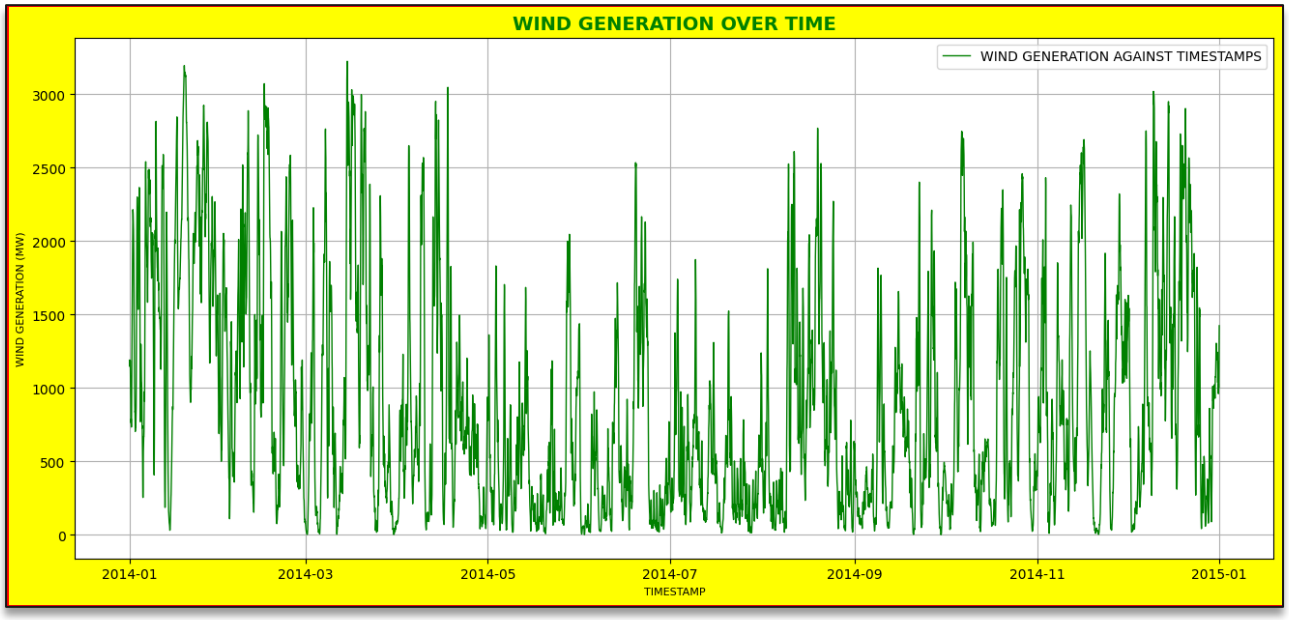


Figure 1: Hourly wind generation vs timestamps

After plotting wind generation against the timestamps (hourly wind generation), 4 more graphs are plotted to evaluate if there is any evidence of intra-annual seasonality. Those graphs were plotted using the wind generation on the y-axis and daily, weekly, monthly, or quarterly timestamps on the x-axis. This was done by first, setting the “Date” column as the DataFrame index and dropping the “Time” column as it is no longer needed, and then resampling this time series DataFrame using the pandas’ function called resample which takes the keyword (D for Daily, W for weekly, M for monthly, and Q for quarterly) for the corresponding sampling period. To better visualize the data, the resample function is used to resample time-series data

by taking the existing time-series data and changing the frequency of the data by up-sampling (increasing the frequency)[2]. This resampling is done using the mean function to take the average of the data in the given period. The resulting graphs are depicted below.

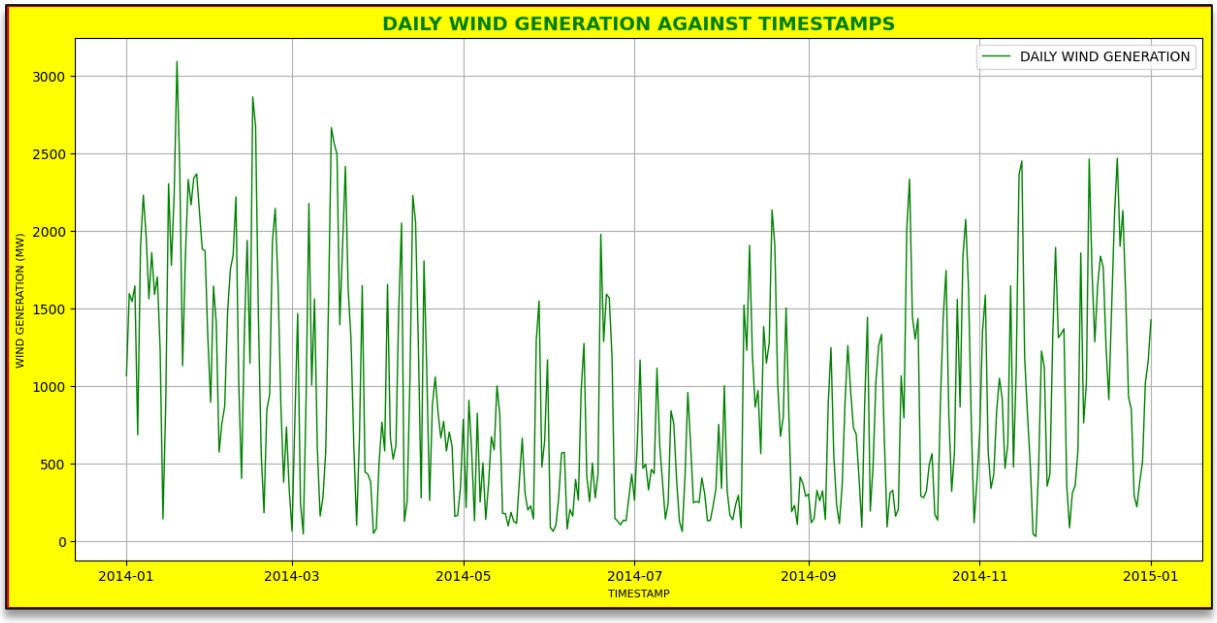


Figure 2: **Daily wind generation vs timestamps**

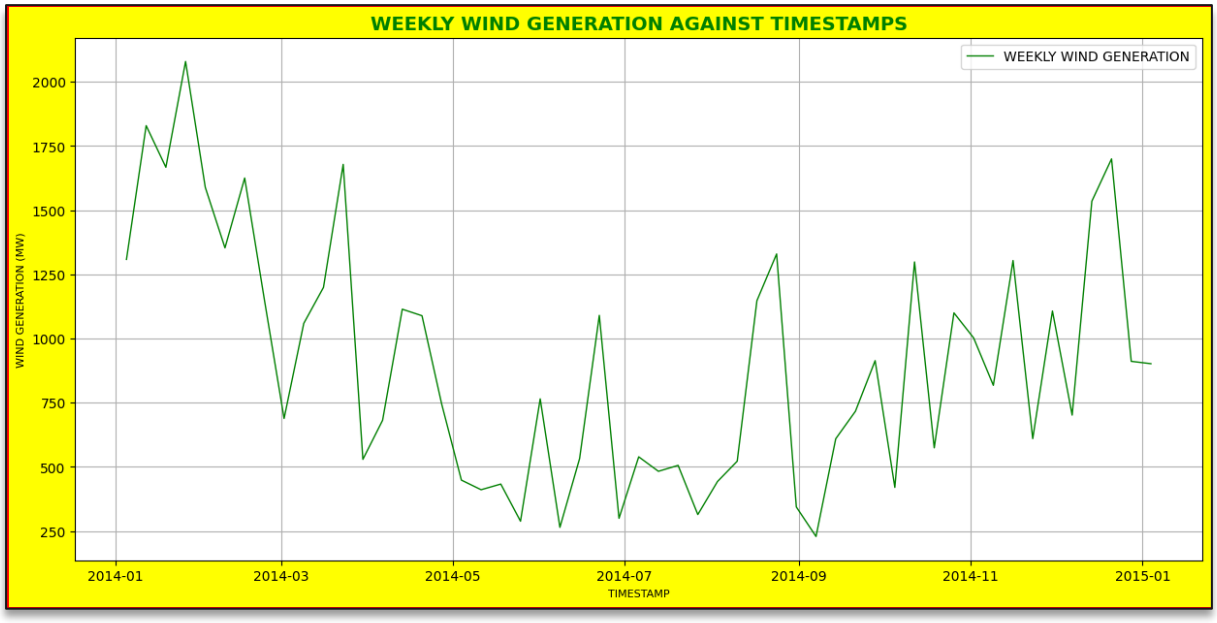


Figure 3: **Weekly wind generation vs timestamps**

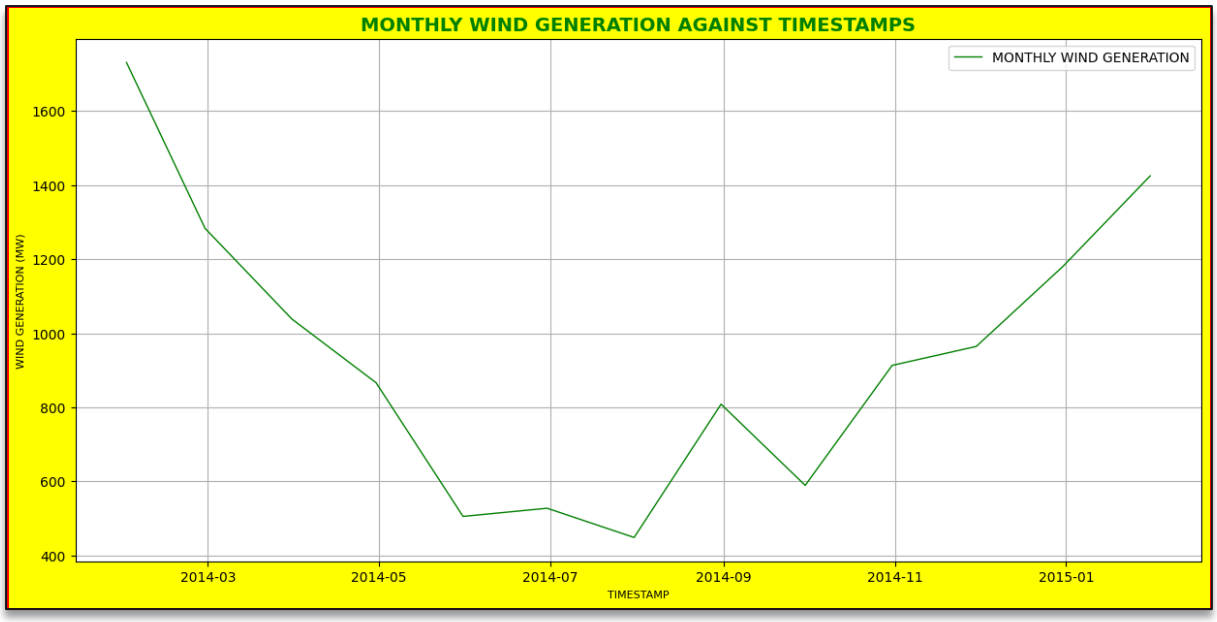


Figure 4: **Monthly wind generation vs timestamps**

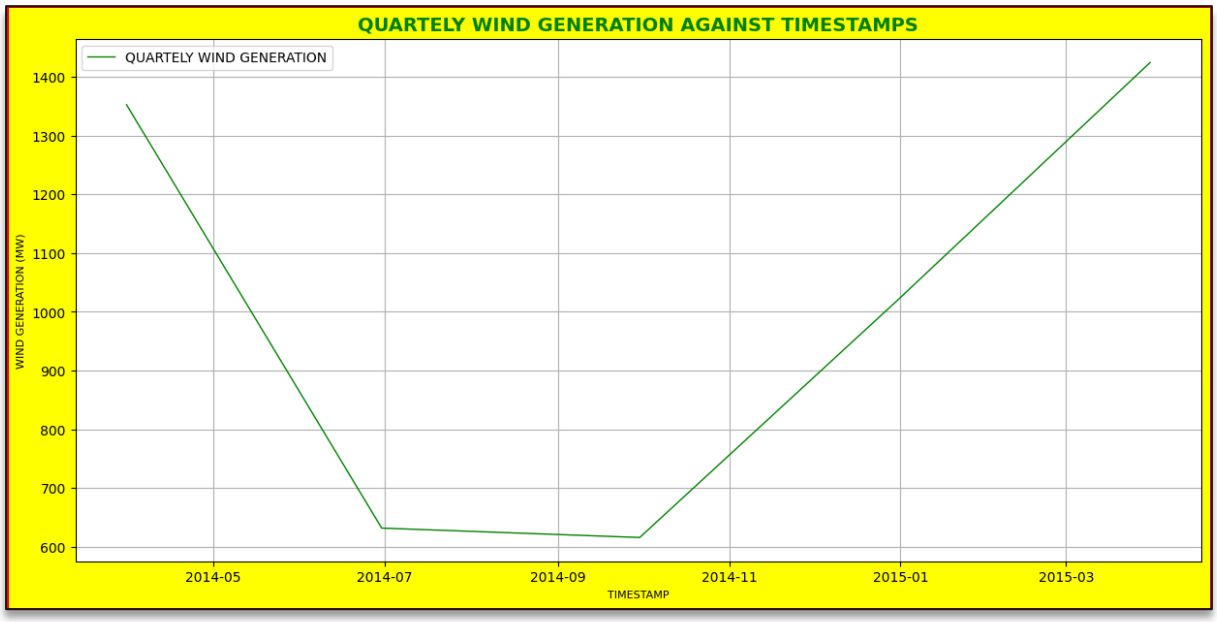


Figure 5: **Quarterly wind generation vs timestamps**

By looking closely at the graphs, it can be inferred that there is seasonality since there is a regular pattern in the graph i.e., regular ups and downs.

QUESTION 2:

The next step is to plot the change in wind generation over time as a percentage of the maximum generation, which was done by first finding the maximum wind generation using the max function. Next is to calculate the change in wind generation over time as a percentage of the maximum generation by taking the wind generation value for the current time, subtracting the previous time value for wind generation, then dividing by the maximum wind generation, and finally multiplying by 100. This is done for every row in the data frame. Furthermore, the results are plotted against the timestamps to produce the following graph.

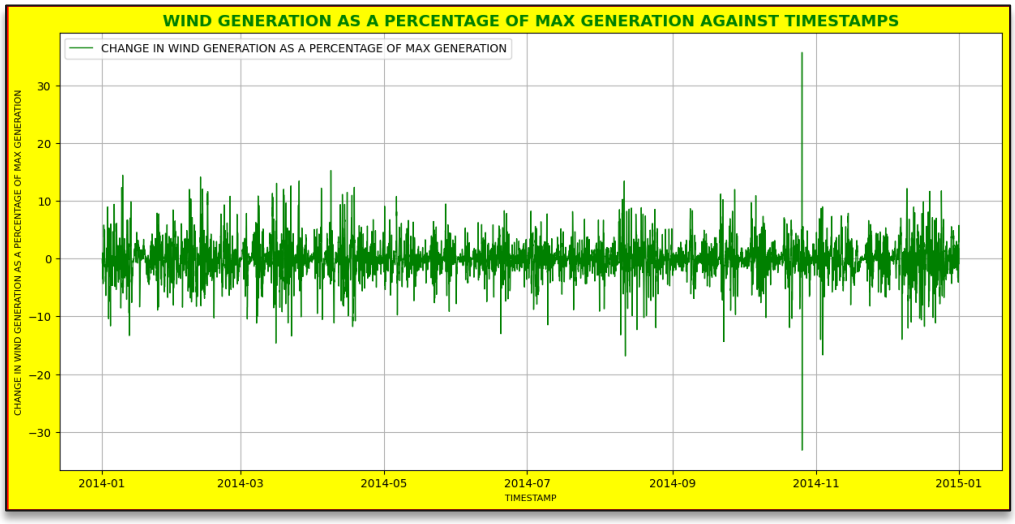


Figure 6: Wind generation as a percentage of maximum generation vs timestamps

To visualize annual seasonality, the data are resampled daily, weekly, monthly, and quarterly and the following plots are produced.

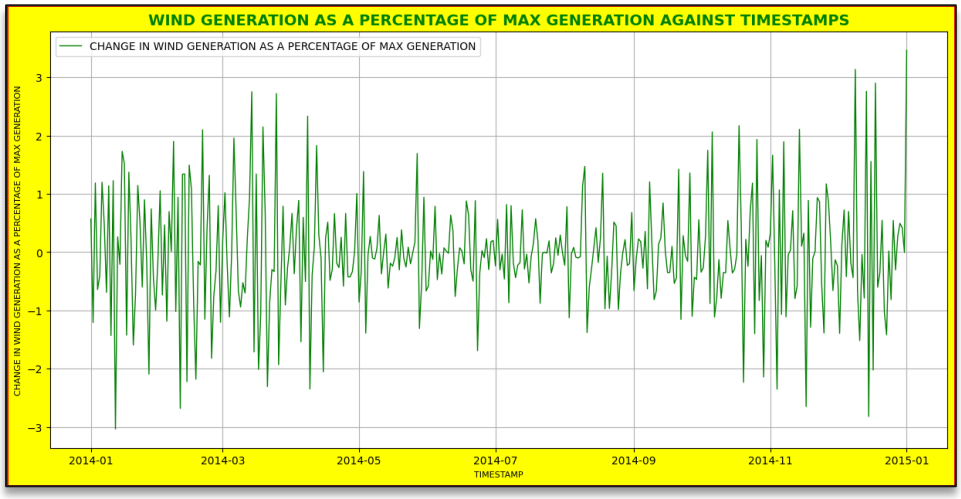


Figure 7: Daily wind generation as a percentage of maximum generation vs timestamps

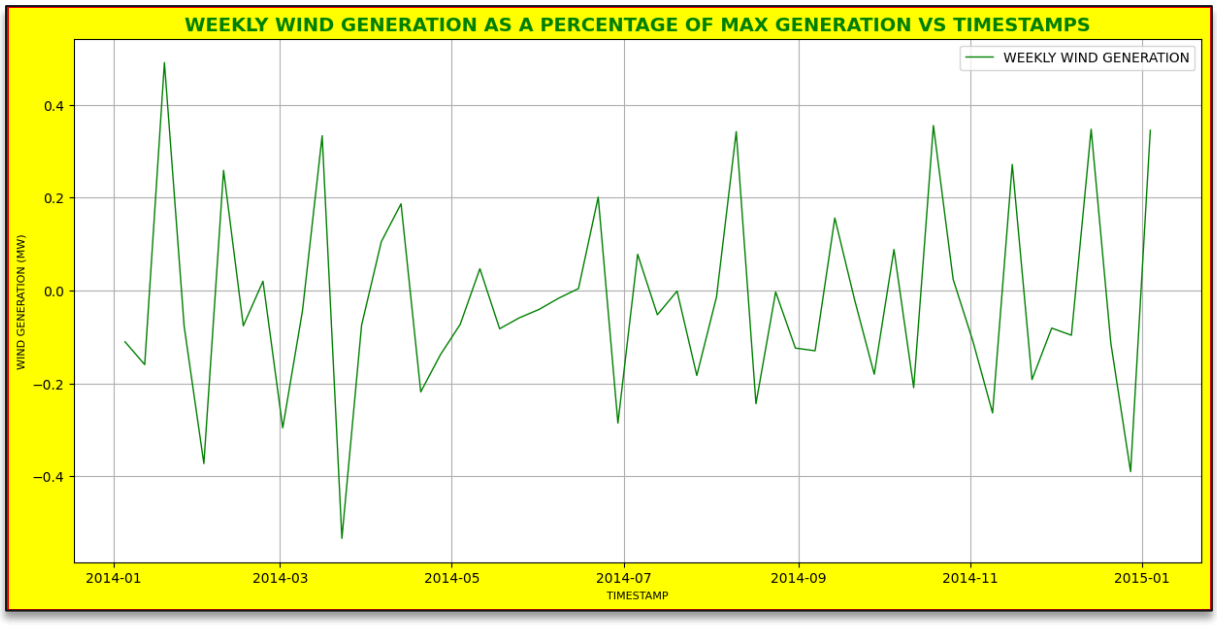


Figure 8: **Weekly wind generation as a percentage of maximum generation vs timestamps**

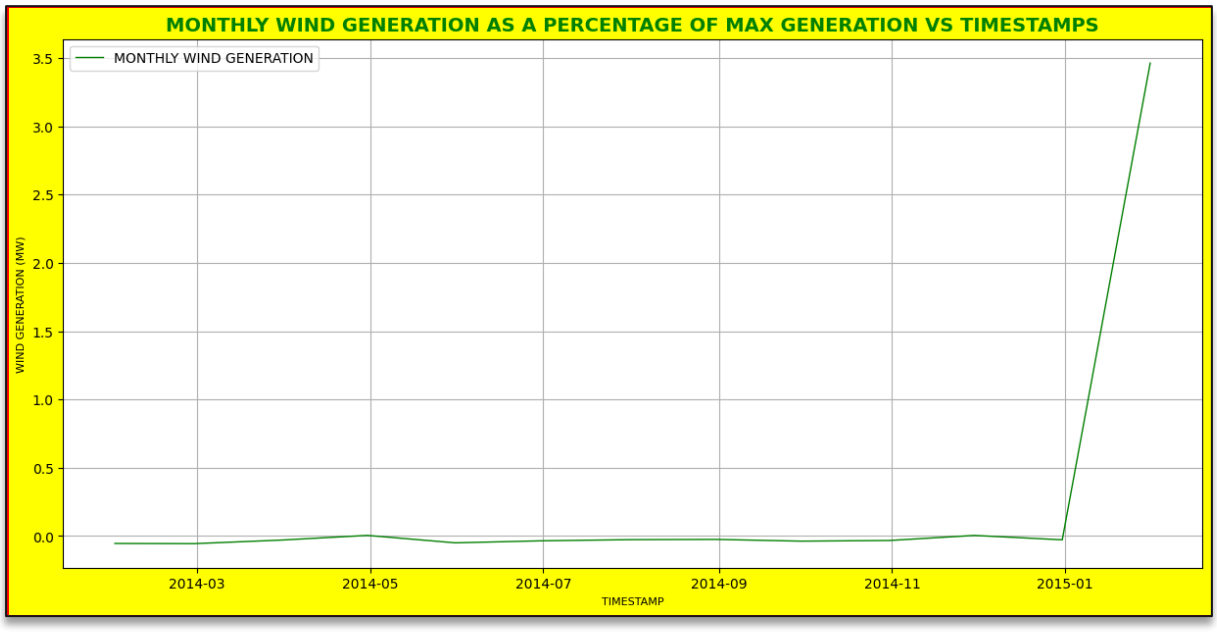
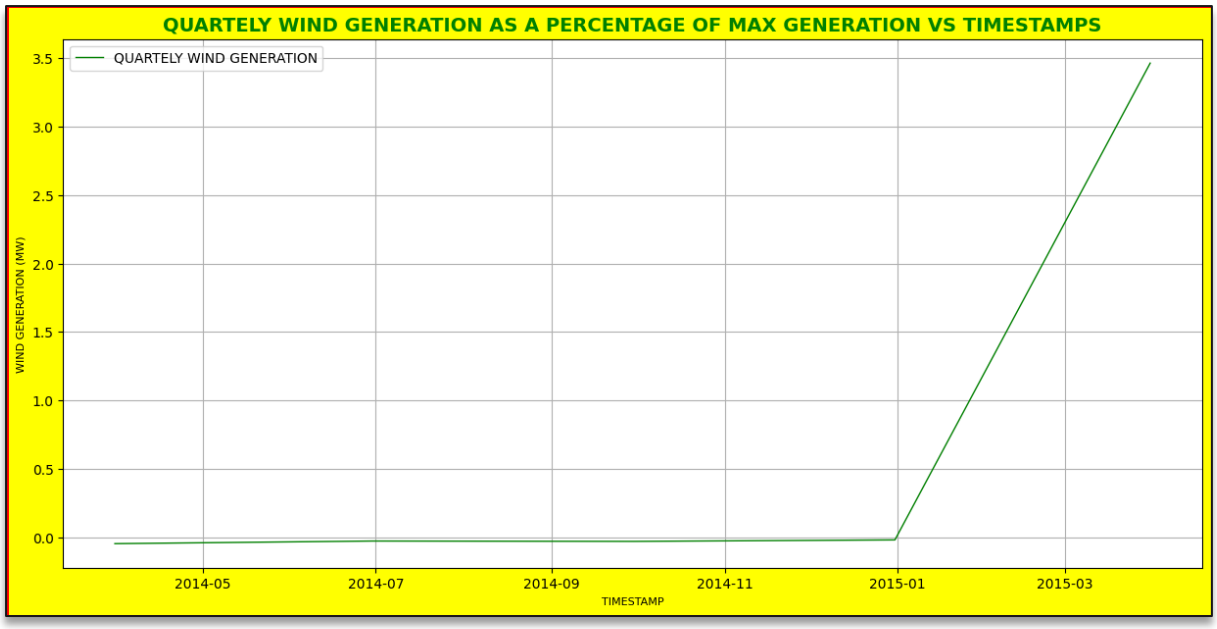


Figure 9: **Monthly wind generation as a percentage of max generation vs timestamps**



<u>Figure 10:</u> Quarterly Wind generation as a percentage of max generation vs timestamps

From the graphs above, it can be inferred that there is seasonality since there is a regular pattern in the graph i.e., regular ups and downs.

<u>QUESTION 3:</u>

By considering the positive and negative ramps in wind power generation, x(t), as a percentage of the maximum, over the hourly timescale, where an hourly ramp is defined as **r(t,d) = 100*[x(t+d)-x(t)]/max(x)** where **d=1** for an hourly sampling period, It is required to create empirical cumulative distribution functions (CDF) for both positive and negative ramps and plot them with probability on a vertical logarithmic axis. Furthermore, to depict the CDF for a normal distribution with mean zero and standard deviation from observations, and lastly to determine whether the normal distribution is a reasonable model for wind power extremes.

This was addressed by first calculating the ramps using the **r(t) = 100 * [x(t+1) - x(t)] / max(x)** formula since d=1. After that, positive and negative ramps are separated and sorted using their absolute values. The resulting data frame is used to plot a semilogy plot (vertical logarithmic axis) depicted below.

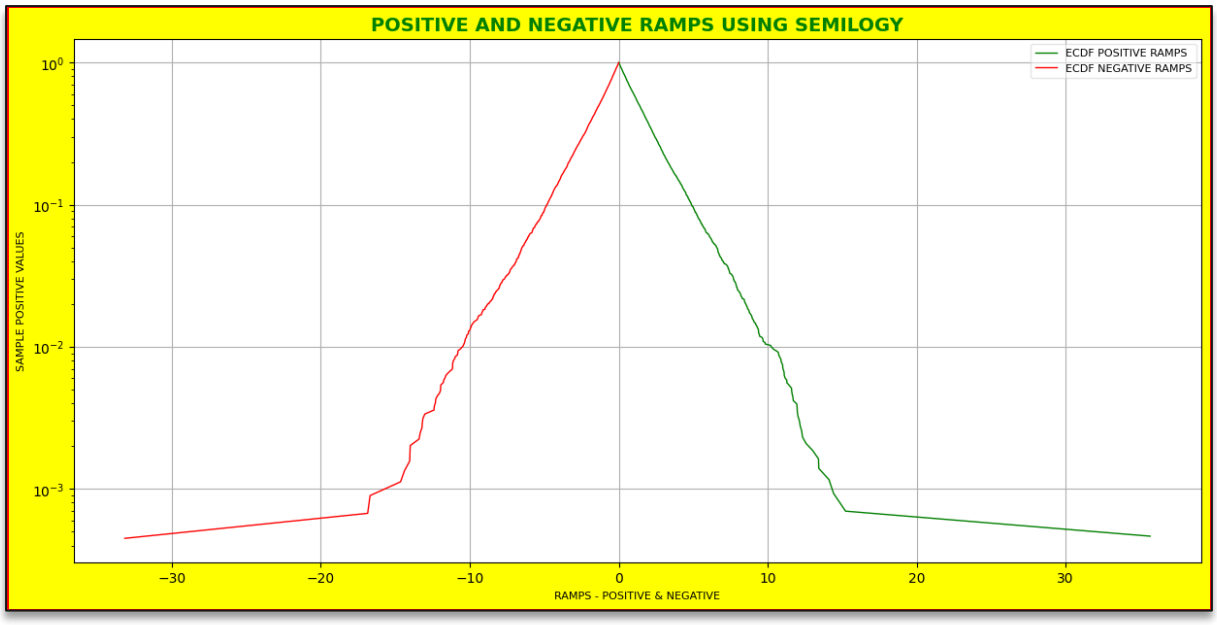


Figure 11: Plot of ramps using a semilogy plot (vertical logarithmic axis)

Next, a normal distribution CDF is plotted and depicted below.

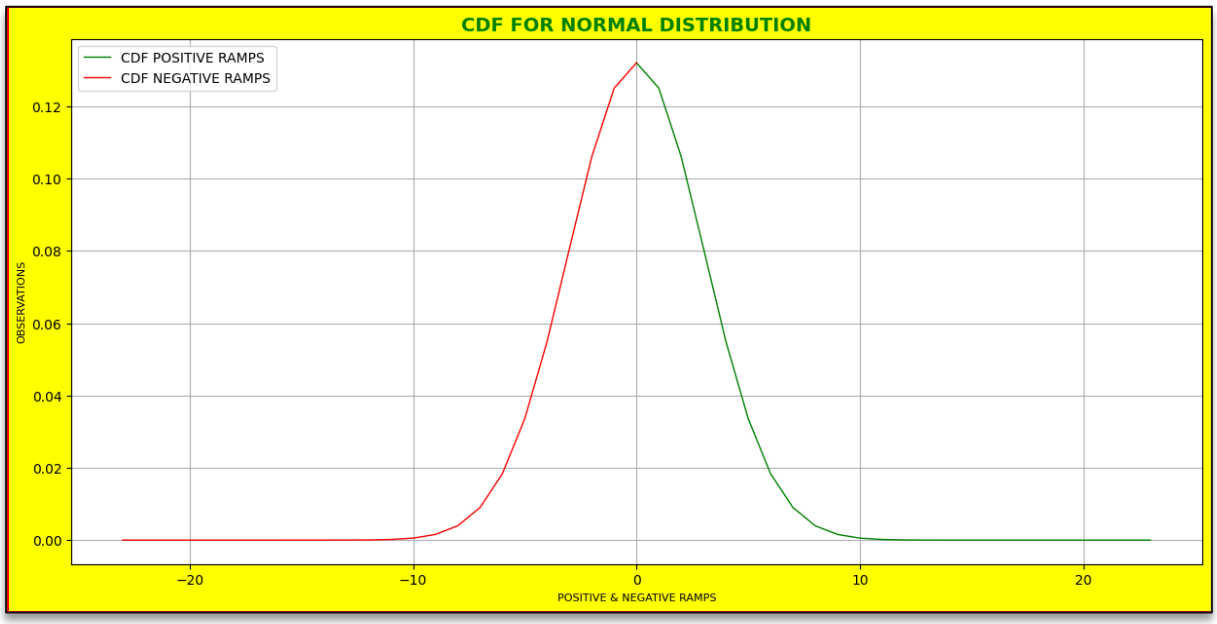
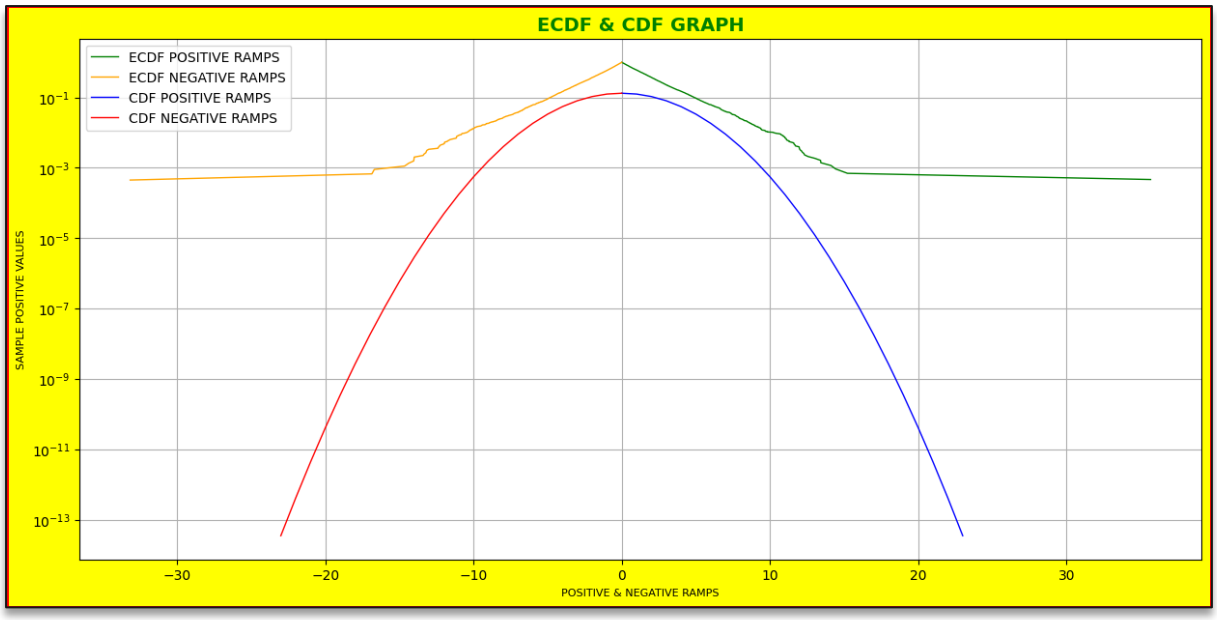


Figure 12: CDF for normal distribution graph

Lastly, both ramps using a semilogy and normal distribution CDF are plotted on the same plot.



<u>Figure 13:</u> Both ECDF and CDF on the same graph (4 curves)

It can be inferred from the graph above that the normal distribution is not a good model for wind power extremes.

<u>QUESTION 4:</u>

It is required to investigate variability over timescales from one hour to one day (1h, 2h, 3h, ..., 24h). This is addressed by using the formula r (t, d) = 100 * [x (t + d) -x(t)] / max(x), iterating over the d param using d=1...24, computing and plotting the 1%, 5%, 95%, and 99% percentiles for each d value and that resulted in the following table and graph.

	0.01	0.05	0.95	0.99
1h	-8.716841	-4.815062	4.880015	8.479010
2h	-16.060179	-9.059496	9.192968	16.316798
3h	-22.552397	-12.898090	12.910182	22.088826
4h	-27.915421	-16.268060	16.290538	27.687760
5h	-32.825107	-19.332486	19.254356	32.462981
6h	-36.398183	-22.313667	22.045018	36.205959
7h	-39.360730	-24.445030	24.267533	39.665995
8h	-42.256247	-27.089663	26.147145	41.597538
9h	-44.676846	-29.287840	27.863366	44.366745
10h	-46.464687	-31.066689	29.746388	46.685310
11h	-48.123458	-32.863831	31.411143	48.319960
12h	-49.138463	-34.074998	32.485893	51.162677
13h	-50.511781	-35.232529	33.481274	53.170087
14h	-51.630216	-36.212253	34.659422	54.191604
15h	-52.600329	-37.182985	35.664104	55.483847
16h	-53.816736	-37.763068	36.621194	56.060613
17h	-54.713865	-38.699541	37.468996	56.617722
18h	-55.060551	-39.260557	38.351832	57.769083
19h	-55.676815	-39.494481	39.206920	58.427885
20h	-55.510665	-39.534941	39.807156	58.312922
21h	-55.698084	-39.456967	40.442426	59.380015
22h	-56.383301	-39.480375	40.814163	59.125690
23h	-56.663453	-39.785453	41.467105	59.350902
24h	-57.091462	-39.971321	41.890153	59.804737

Figure 14: Percentile analysis table on the ramps for the timescales

As graphed below, 95% and 99% percentiles show the increase in power generation as the day grows while 1% and 5% percentiles show the decrease in power generation hour by hour the day.

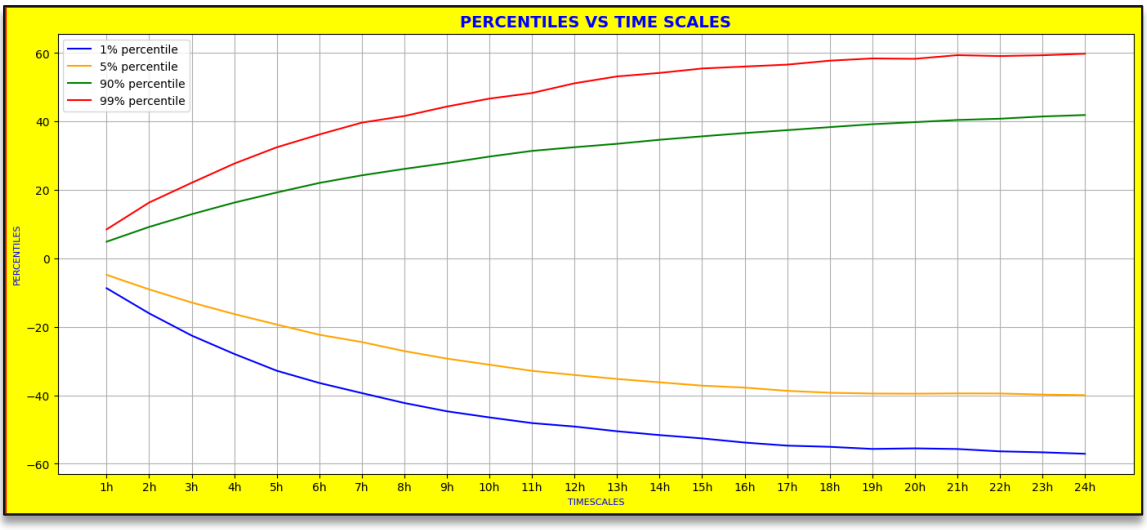


Figure 15: Percentile analysis graph on the ramps for the timescales

QUESTION 5:

For this question, it was required to calculate and plot the autocorrelation of wind generation for lags over 10 days. The steps used are first, calculating the autocorrelation of the wind generation for 10 days lags which are 240 lags in total (1 day * 24 hours * 10 days = 240 lags), done using the **tsa.acf** function from **statsmodels.api**[3]. The next step is plotting the autocorrelation of wind generation for lags over 10 days which is done using **plot_acf** function from **statsmodels.graphics.tsaplots**[4]. The resulting plot is depicted below.

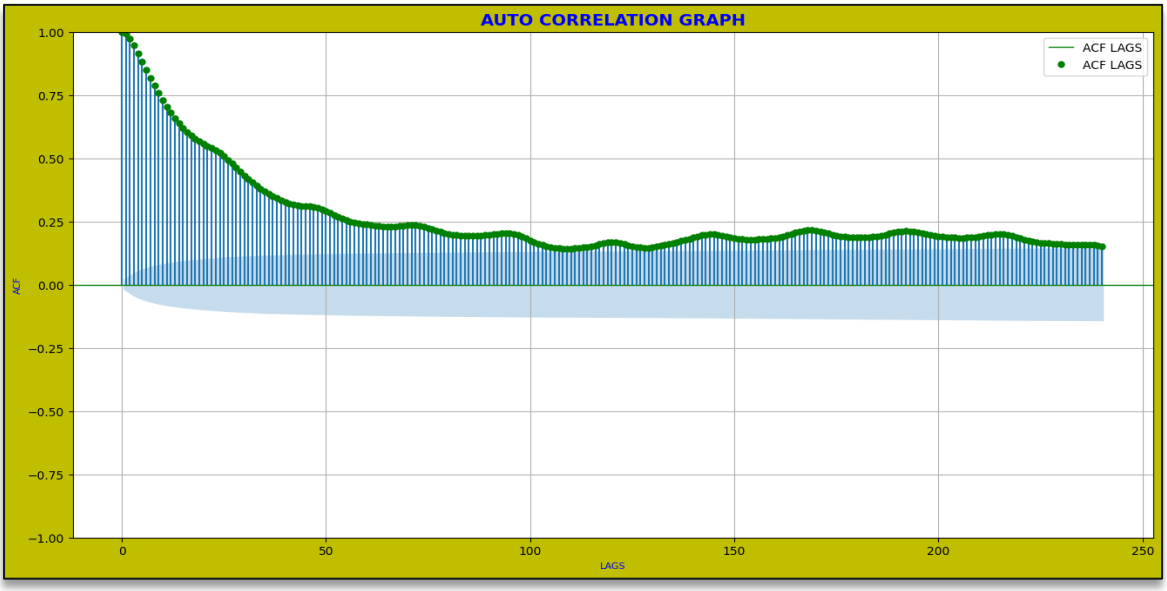


Figure 16: Autocorrelation of the wind generation (actual) for 10 days lags

The above plot shows a downward trend in the autocorrelation of wind generation over lags. It can be inferred that as the time lag between the wind generation values increases, their correlation decreases.

QUESTION 6:

It was required to calculate and plot the autocorrelation of **change in wind generation** for lags over 10 days by including the horizontal lines to detect statistically significant values ($p<0.05$). This was done where the first step was to calculate the autocorrelation of the change in wind generation for 10 days delays, for a total of 240 lags (1 day * 24 hours * 10 days = 240 lags) using the tsa.acf function from statsmodels.api. The plot_acf function from statsmodels.graphics.tsaplots is used to plot the autocorrelation of wind generation with delays greater than 10 days. The resulting plot is shown below.

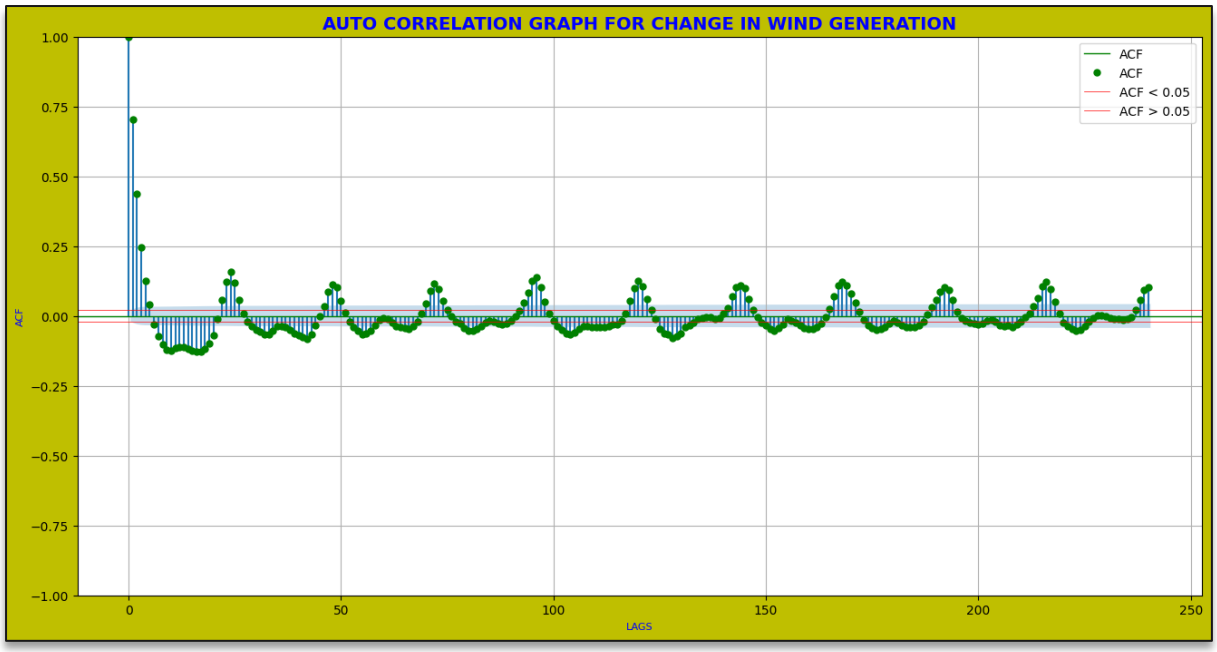


Figure 17: Autocorrelation of the change in wind generation for 10 days lags

The above plot shows that there is diurnal seasonality because it has a pattern of peaks and low points that repeat daily. Moreover, it can more appropriate to model the change in wind generation than the wind generation. This is because the variation in wind generation is a stationary time series, which means that its statistical features, such as mean and variance, remain constant throughout time.

QUESTION 7:

The aim here was to investigate the structure of the wind generation time series using a variance ratio test. This was done by first examining the variance ratio using the "**VarianceRatio**" function from the **arch.unitroot** library which is used to investigate the structure of the wind generation time series. That gave the following results.

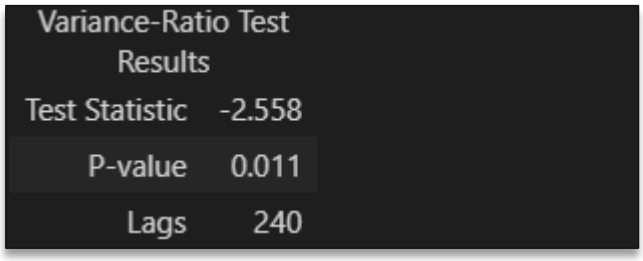


Figure 18: Variable ratio test results

Using the results above, it can be inferred that the null hypothesis of a random walk can be rejected. This is because the P-value of 0.011 is less than the significance level of 0.05, indicating that the result is statistically significant. A low P-value indicates that the observed difference in the test statistic is not due to chance, and the null hypothesis may be rejected.

In addition, using the Augmented Dickey-Fuller (ADF), the following results have been generated.

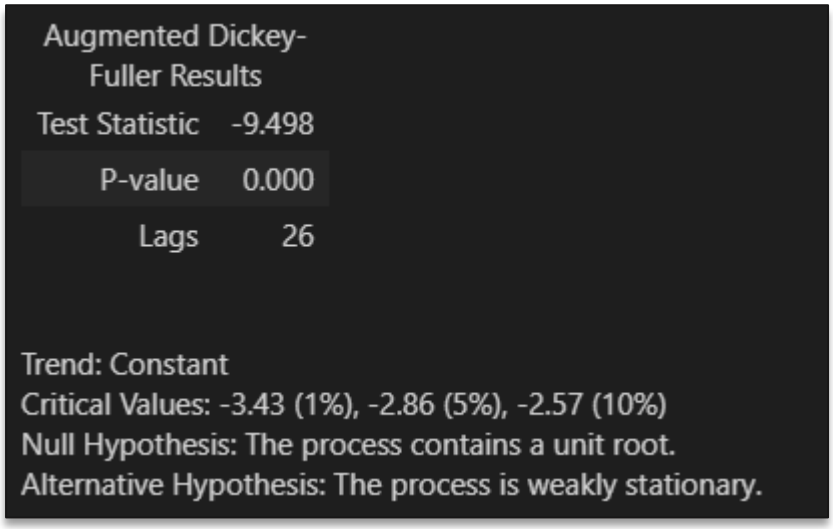


Figure 19: Augmented Dickey-Fuller (ADF) test

From the above test results, the test statistic value is -9.498, and the p-value is 0.000. The p-value is less than the significance level (0.05), which means that the null hypothesis of a unit root can be rejected. In other words, the time series has a stationary process.

Furthermore, the above ADF test findings imply that there is evidence of mean reversion because the test statistic is strongly negative, and the p-value is near zero (or zero). This signifies that the time series is stationary, and the mean does not change over time. Moreover, this suggests that there is no mean aversion because the mean of the time series does not change with time, indicating that it is stationary.

QUESTION 8:

It is required to estimate the optimal window for a simple moving average and determine if there is a simple benchmark that improves on the persistence benchmark. To address this, the simple moving average (SMA) is calculated for each window size from 1 to 24 using the rolling and mean functions from pandas[5]. After that, a data frame with window sizes' moving averages is made, null values are filled, the mean absolute error (MAE) between the SMA and the actual wind power is calculated and the results are presented in the following table.

	MAE
1	0.000000
2	33.382471
3	62.862934
4	89.390713
5	113.502877
6	135.513893
7	155.782852
8	174.581157
9	191.861547
10	207.697341
11	222.125052
12	235.404059
13	247.846811
14	259.396548
15	270.069150
16	279.936994
17	289.123206
18	297.732165
19	305.644117
20	312.853296
21	319.455683
22	325.480773
23	331.071757
24	336.425749

Figure 20: MAE between the SMA and the actual wind power for each window size

From the above table, the first window size with n=1 is the one with the least MAE which is zero.

There are various benchmarks that can improve on the persistence benchmark, including ARIMA and SARIMA, mean, median, and moving average benchmarks. These approaches employ historical data to make predictions and are frequently used in time series analysis to increase prediction accuracy.

QUESTION 9:

It is required to evaluate the mean-Absolute-error (MAE) performance of the persistence benchmark forecast over forecast horizons from one hour to one day and plot MAE as a percentage of the maximum generation for the persistence benchmark. This was done by first, calculating persistence for every window size, filling in missing values, calculating the mean absolute error (MAE) between the predicted wind power and the actual wind power, and plotting the MAE as a percentage of the maximum generation for the persistence benchmark which has given the following graphic.

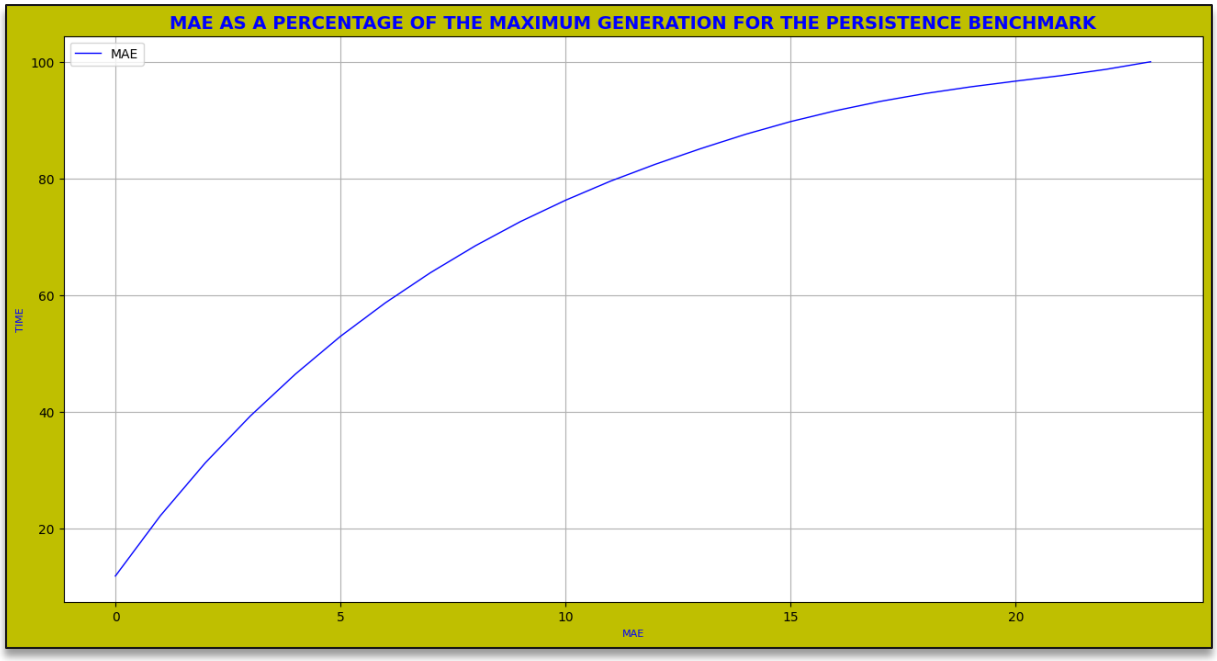


Figure 21: MAE as a percentage of the max. generation for the persistence benchmark

QUESTION 10:

It is required to loop over the number of parameters using an ARIMA model for describing wind generation and using information criteria (AIC and BIC) to find the optimal ARIMA model. An AutoRegressive Integrated Moving Average (ARIMA) is a statistical method used for analyzing and modeling time series data. It models the past values of a time series to predict future values[5]. This is addressed by looping through a range of parameters (p and q = [1:4]), passing the wind generation data frame along with p, d, q parameters to the ARIMA model, fitting and returning the model estimates, calculating the AIC and BIC from the estimation and store the results in a detailed table which is shown below.

P Parameter	Q Parameter	AIC Criterion	BIC Criterion
1.0	1.0	98978.479903	98999.713414
1.0	2.0	98964.857960	98993.169308
1.0	3.0	98966.576021	99001.965206
1.0	4.0	98966.154330	99008.621352
2.0	1.0	98854.032745	98882.344093
2.0	2.0	98784.948487	98820.337672
2.0	3.0	98968.839175	99011.306197
2.0	4.0	98812.833183	98862.378042
3.0	1.0	98782.501139	98817.890324
3.0	2.0	98784.926424	98827.393446
3.0	3.0	98788.930764	98838.475623
3.0	4.0	98787.230446	98843.853142
4.0	1.0	98966.168754	99008.635776
4.0	2.0	98786.617048	98836.161907
4.0	3.0	98952.987202	99009.609898
4.0	4.0	98981.306584	99045.007117

THE OPTIMAL PARAMETERS ARE

P Parameter	3.000000
Q Parameter	1.000000
AIC Criterion	98782.501139
BIC Criterion	98817.890324

Figure 22: ARIMA model selection of optimal parameters

There is improvement in the model's performance as the AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are smaller for the optimal parameters (P Parameter = 3.0, Q Parameter = 1.0, AIC Criterion = 98782.501139, BIC Criterion = 98817.890324).

Smaller AIC and BIC values indicate better model performance as they measure the goodness of fit of the model and consider the complexity of the model.

There is an improvement in model performance because the AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) for the ideal parameters decreased (P parameter = 3.0, Q parameter = 1.0, AIC Criterion = 98782.501139, BIC Criterion = 98817.890324). Smaller AIC and BIC values suggest higher model performance since they assess the model's goodness of fit and complexity.

From the table above it can be inferred that the parameters at **p=3** and **q=1** are optimal since they have the lowest AIC and BIC values of **98782.501139** and **98817.890324** respectively.

REFERENCES

[1] ‘pandas.DataFrame.interpolate — pandas 1.5.3 documentation’. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html (accessed Jan. 23, 2023).

[2] ‘Python | Pandas dataframe.resample() - GeeksforGeeks’. https://www.geeksforgeeks.org/python-pandas-dataframe-resample/ (accessed Feb. 11, 2023).

[3] ‘statsmodels.api.tsa.acf Example’. https://programtalk.com/python-more-examples/statsmodels.api.tsa.acf/ (accessed Feb. 12, 2023).

[4] K. Drelczuk, ‘ACF (autocorrelation function) — simple explanation with Python example’, *Medium*, May 15, 2020. https://medium.com/@krzysztofdrelczuk/acf-autocorrelation-function-simple-explanation-with-python-example-492484c32711 (accessed Feb. 12, 2023).

[5] J. Brownlee, ‘How to Create an ARIMA Model for Time Series Forecasting in Python’, *MachineLearningMastery.com*, Jan. 08, 2017. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ (accessed Feb. 12, 2023).