Group 6

# Wind Energy Forecasting (Inter-States)

30th April, 2021



| | | |
|---|---|---|
| Suyash Sharma | - | 2017B4A10627P |
| Saksham Upadhyay | - | 2017B4A30827P |
| Kshitij Gupta | - | 2019B4A30619P |
| Ahmad Faraaz | - | 2017B4A70558P |
| Vansh Bansal | - | 2019B4A70621P |
| Aman Gupta | - | 2017B4A20835P |
| Rajan Sahu | - | 2019B4A70572P |
| Rishi Garg | - | 2019B4A70642P |

# Introduction

Due to a rapid depletion of non-renewable resources, ever-increasing pollution and devastating climate change, it is a desideratum that the world briskly moves towards renewable energy sources like wind energy. One of the reasons why this is already not taking place at a large scale in India is because of the difficulty in its implementation due to its supposed seasonal and geographical irregularities.
The following report focuses on wind energy and the various factors that affect its efficiency. The study is only restricted to 4 key states of India, namely - Andhra Pradesh, Madhya Pradesh, Rajasthan, and Tamil Nadu. The efficacy of wind energy depends on various factors like temperature, weather, wind speed etc. of that particular region and this study aims to understand which parameters are the most relevant by performing descriptive statistical analysis and finding the correlation between them. We also perform a time-series analysis on the wind speed data to try to find any trend or seasonality in the data. Tools like AR, MA, ARMA, ARIMA, and SARIMA are also used to forecast weekly wind speed.

## Dataset

The given dataset contains hourly data of various factors affecting wind energy in Rajasthan, Madhya Pradesh, Andhra Pradesh and Tamil Nadu from the year 2000 to 2014. In every entry, it had the following attributes:
1. Timestamp of measurement
2. DHI and Clearsky DHI
3. DNI and Clearsky DNI
4. GHI and Clearsky GHI
5. Dew Point
6. Temperature
7. Pressure
8. Relative Humidity
9. Solar Zenith Angle
10. Wind Speed

**Definition**: A time series is a sequence of periodically recorded observations of a variable [2]. Throughout our discussion, we measure the quality of predictions with the MAPE metric (Mean Absolute Percentage Error) which is defined as follows:

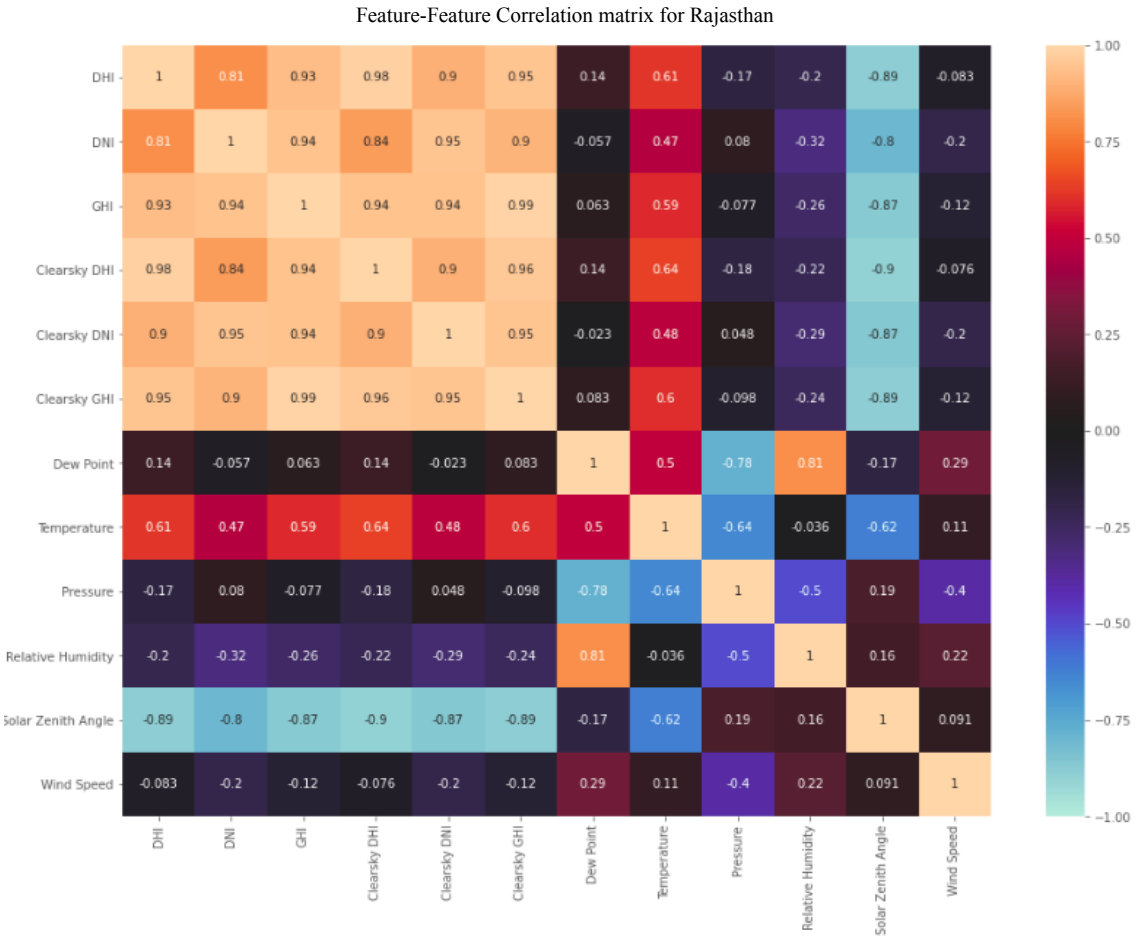
$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{Ft - Xt}{Xt}\right|$$

where n is the number of observations, $F_t$ is the forecast at time t, and $X_t$ is the actual observation at time t.

# CORRELATION

We obtained the feature-feature correlation maps for different states. For the sake of simplicity, only the correlation map of Rajasthan is shown here. Other states which have similar results have been shown in the appendix.

FIGURE 1 : CORRELATION HEATMAP FOR RAJASTHAN




The correlation coefficient represents the strength of linear relationship between two variables. Zero correlation means that the two variables are not related.

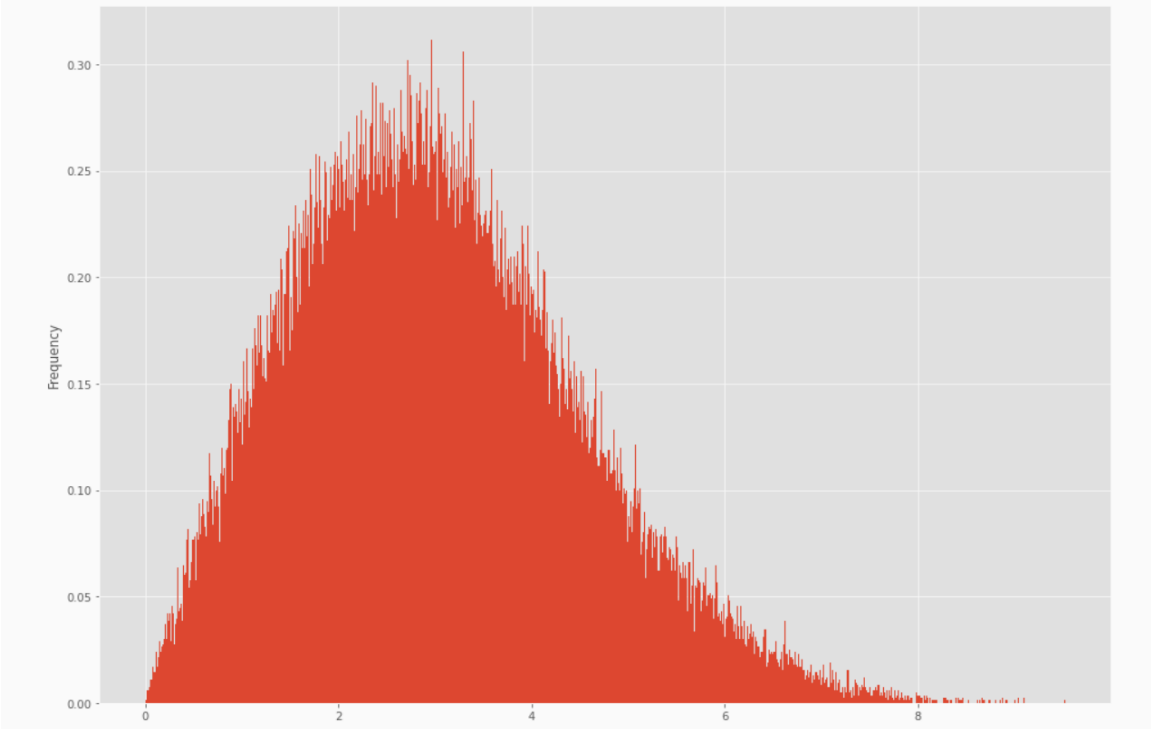
In Fig 1, the correlation factor between DNI and GHI is 0.94, between Clearsky GHI and GHI is 0.99, and Clearsky GHI and Clearsky DNI is 0.95. Hence DNI, DHI and GHI are highly correlated. Dew Point has almost zero correlation with DHI, DNI, GHI and the solar zenith angle. However, it has a strongly negative correlation with Pressure and a strongly positive correlation with Relative humidity. Temperature has a negligible correlation with relative humidity and wind speed. Pressure has a moderately negative correlation with temperature and relative humidity. It has little correlation with the wind speed. Wind speed has negligible correlation with all the factors, the highest correlation being 0.29 with Dew point. We preferred correlation matrix over covariance matrix, as the correlation values remain unaffected by change in dimension, scale and also quantifies the strength of relationship between two variables.

## Distribution Of Wind Speed Data Over A Location

First we checked the normality of Wind Speed Data by setting up the following Hypothesis test.
**$H_0$ : The Wind Speed data follows a Normal Distribution**
**$H_a$ : The Wind Speed data doesn't follow the Normal Distribution**
We took help of two well known tests to check if our hypothesis could be rejected or not.
The first test: **SHAPIRO WILK'S TEST** was conducted by using a simple pre defined python script. Although the Shapiro Wilkis test is known to be a reliable test, there were still suggestions from the statistics community that the test might be suitable for smaller data of samples. So we used the second test : **D'AGOSTINO'S K2 TEST**. While Shapiro's test evaluates the data sample and tells us how likely it is that the underlying distribution is a Normal/Gaussian distribution, on the other hand D'Agostino's K2 test calculates Kurtosis and Skewness.
Both the tests ended up with the conclusion that the underlying distribution wasn't normal. The p value obtained was negligible( Level Of Significance = 0.05 ).

The following Histogram could be also taken as a reference, which clearly showed a great degree of Skewness.

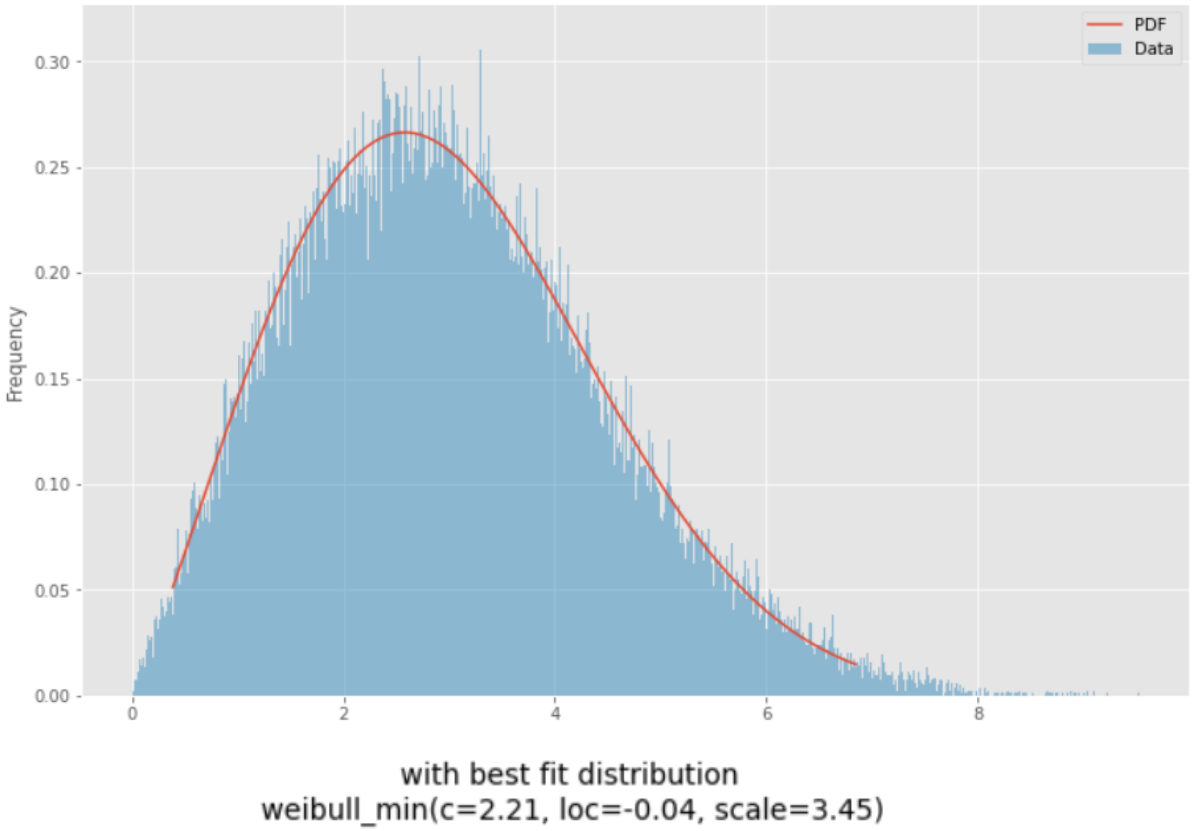
**FIGURE 2 : Daily Wind Speed Distribution , Wind (m/s)**




The further analysis was focussed on determining the best fit distribution for the data. The candidate distributions used were : Rayleigh, Exponential, Normal, Weibull, Gamma, Beta and Alpha. A python script was used to perform this(Appendix). The data was to be fitted into each type of distribution iteratively and the SSE was calculated. The following iterations compared the SSE value to the previous one until the best distribution with the minimum SSE was found. SSE being the squared sum of errors

gave us indication of what was the Calculated error in PDF fitted and the actual values of distribution. For Rajasthan the following results were retrieved :

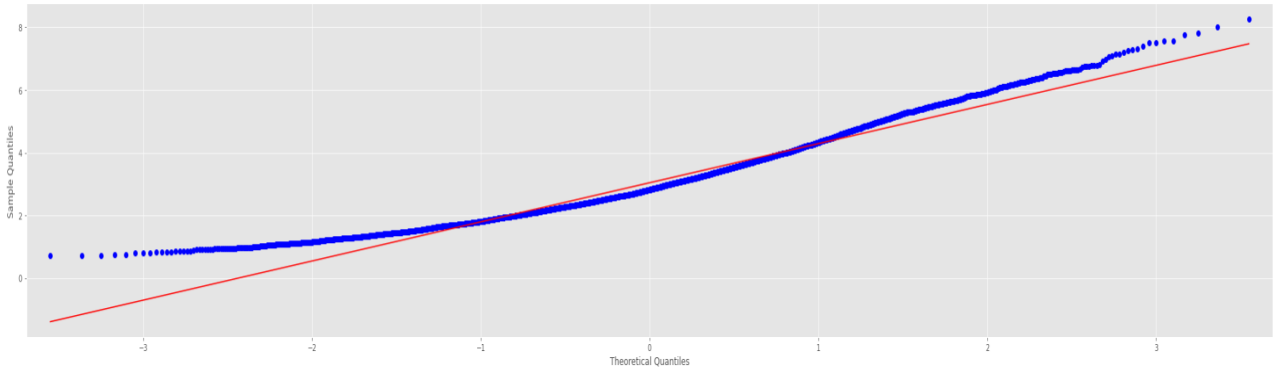
**FIGURE 3 : Best Fit Distribution on Rajasthan Wind Speed data, Wind (m/s)**




Similarly for other states, the distributions were more or less similar. However an interesting and an obvious observation was noted. In the case of Rajasthan and Madhya Pradesh, which aren't located along the coastline, the underlying distribution was found to be quite similar for both of these states. On the other hand Tamil Nadu and Andhra Pradesh had similar histogram plots. However Tamil Nadu data was best described by a gamma distribution, Andhra Pradesh data was best described through weibull max and Madhya Pradesh had a Weibull Min fitted over the data. In the case of Tamil Nadu and Andhra Pradesh, Kernel Density distributions might have provided better fit/results.

Apart from this QQ plots were also used as a final check to strengthen the notion of not having normal data . The following plot was obtained for Rajasthan :

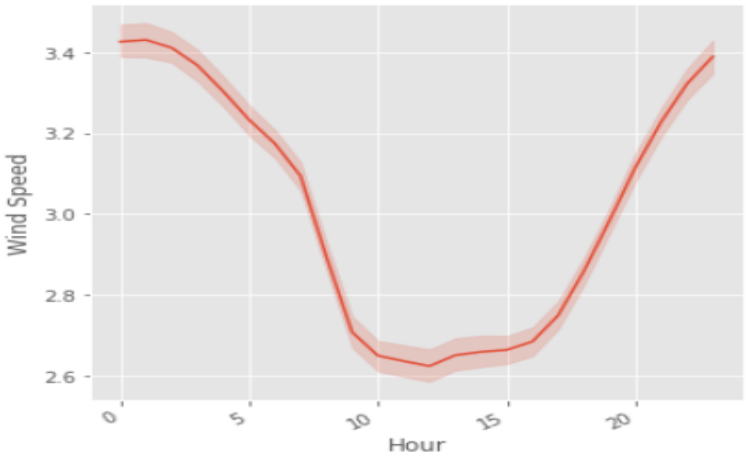
**FIGURE 4 : QQ PLOT FOR RAJASTHAN DAILY WIND DATA**



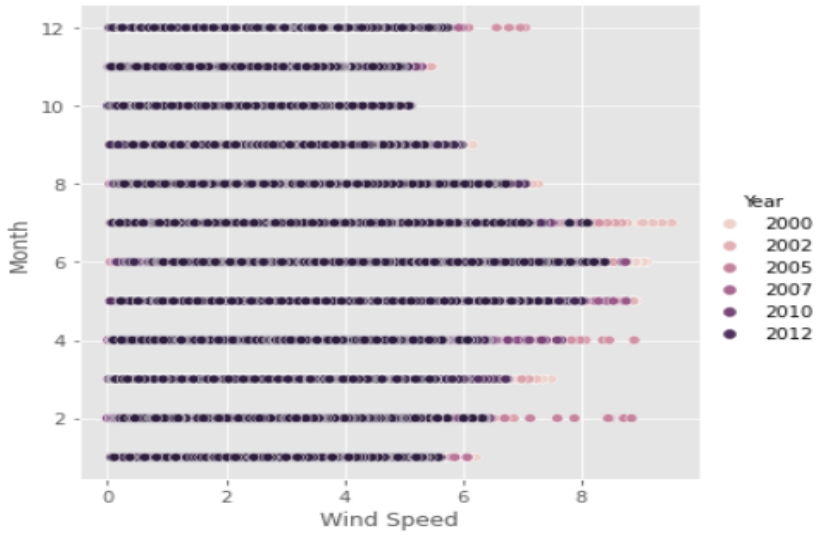

A line of dots on a 45-degree angle from the bottom left of the plot to the top right would indicate a perfect match for the distribution. The dots deviate from the grid, indicating a divergence from the predicted distribution, which is a normal distribution in this case.

## OTHER DESCRIPTIVE PLOTS :

**Hourly Variation of Wind Speed (m/s)**




**Annual Distribution of Wind Speed over significant years**



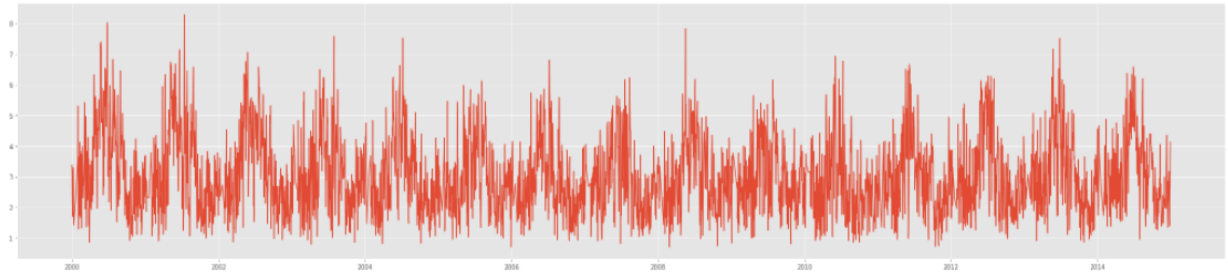

## STATIONARITY TESTS

**ADF Test**: To check the stationarity we performed the Augmented Dickey-Fuller test. The following hypothesis test was conducted

$H_0$: The series has a unit root
$H_a$: The series has no unit root

The given test is a unit root test. Unit root is a characteristic of a time series that makes it non-stationary. The test results gave a p-value of $3.33 \times 10^{-8}$. And for a given level of significance as 1%, we were able to reject the null hypothesis and conclude that our data waas stationary.

FIGURE 5: Wind Speed Data of Rajasthan over a time span of 15 years



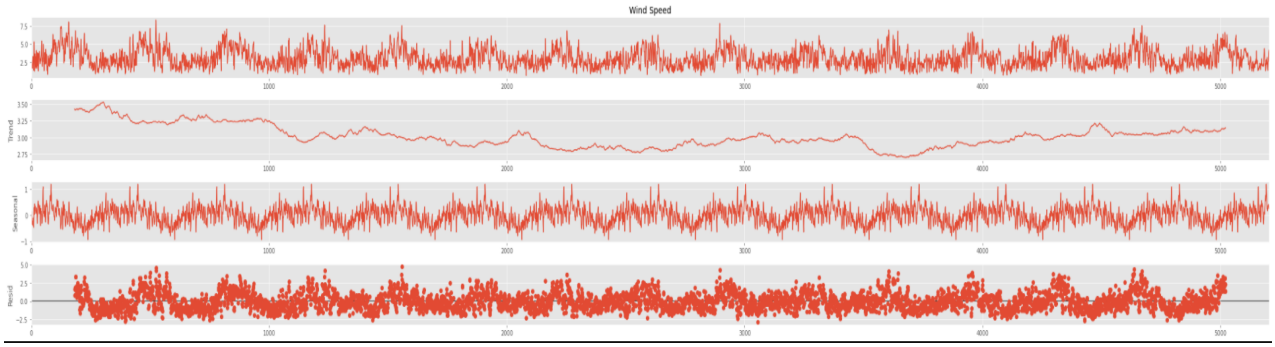
As could be seen from the above plot, the wind data doesn't show a general trend or growth in Wind Speed as time progresses.

## TIME SERIES ANALYSIS

The given data was an hourly based Wind Speed data. So in order to remove the unnecessary variations caused due to hourly fluctuations, or a sudden change in the weather conditions, it was important to average out the hourly data to a daily basis data. This also helped in reducing the size of the data and also helped in giving a reasonable estimate of the forecast.

Finally, since the data was found to be stationary, that is with the level of time series, the fluctuations didn't show a general trend, an additive model was adopted. The seasonal fluctuations didn't change over time.
The decomposition of the Time series model has been shown :

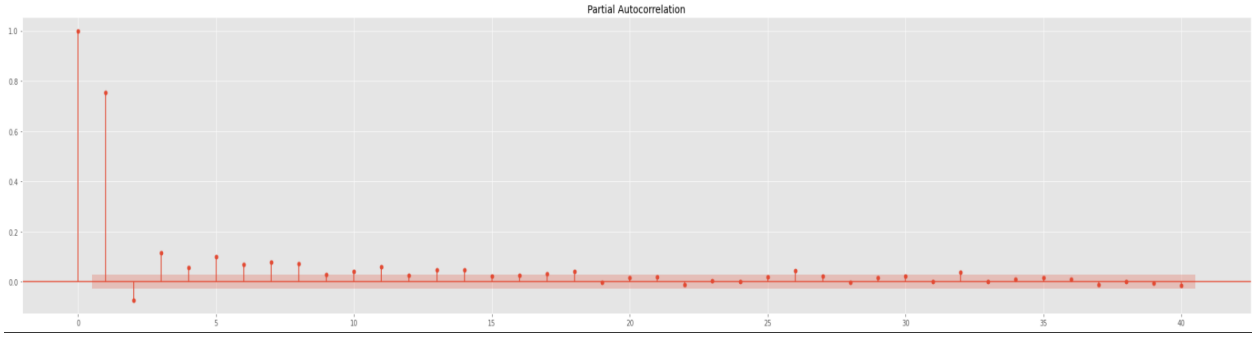
FIGURE 6 : TIME SERIES DECOMPOSITION OF WIND SPEED DATA (DAILY)




## PARTIAL AUTOCORRELATION FUNCTION (PACF)

Partial autocorrelation gives a summary of the relationship between an observation in a time series with observations at prior steps, while ignoring the relationships/indirect relations with the intervening observations. We don't want to keep so many associated features in our models because this can lead to multicollinearity problems. As a result, we just need to keep the features that are important.

FIGURE 7 : PACF PLOT



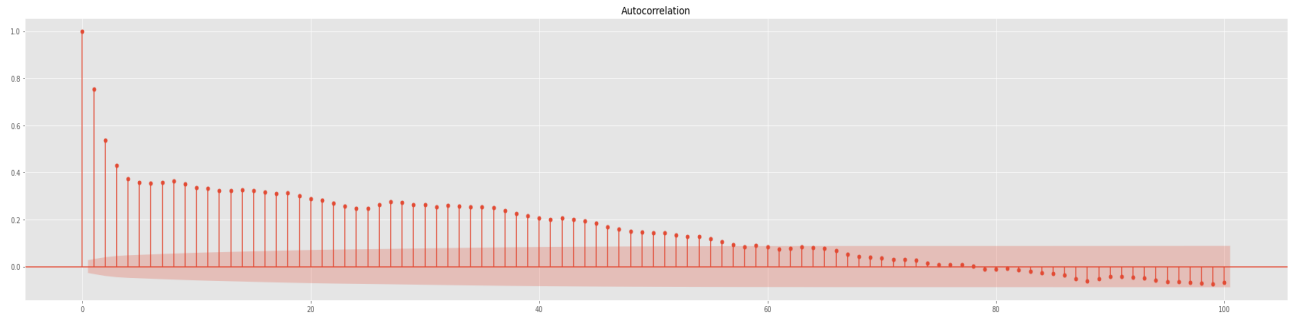

As stated above, PACF removes the indirect correlation between current value and the lagged value. Simply put, AR models will benefit from lags with PACF above the threshold. A potential value of p = 12 (for AR) in our case comes from a plot of PACF for the first 40 lags. Although a lag value of 15 could have also been significant, but to reduce computational complexities, a value of 12 also worked just well.

## AUTOCORRELATION FUNCTION (ACF)

ACF stands for auto-correlation function (complete), and it returns the auto-correlation values of any sequence with its lagged values. . In simplistic words, it explains how well the series' current value is compared to its previous values. Trend, seasonality, cyclic, and residual elements can all be seen in a time

series. Since ACF considers all of these factors when determining correlations, it is referred to as a "true auto-correlation series."

FIGURE 8 : ACF PLOT




Again, MA models will benefit from the lags with ACF above the threshold. Although the plot suggests that value of q = 60, would have been an ideal choice, owing to long computation time, a q value of 30 was selected for running the MA model.

## PREDICTION

**AR:** The AutoRegressive (AR) model is a type of multiple regression model in which the output variable is forecasted using a linear combination of previous values of the same variable in the time series. An AR model of order p, referred to as AR (p), can be written as

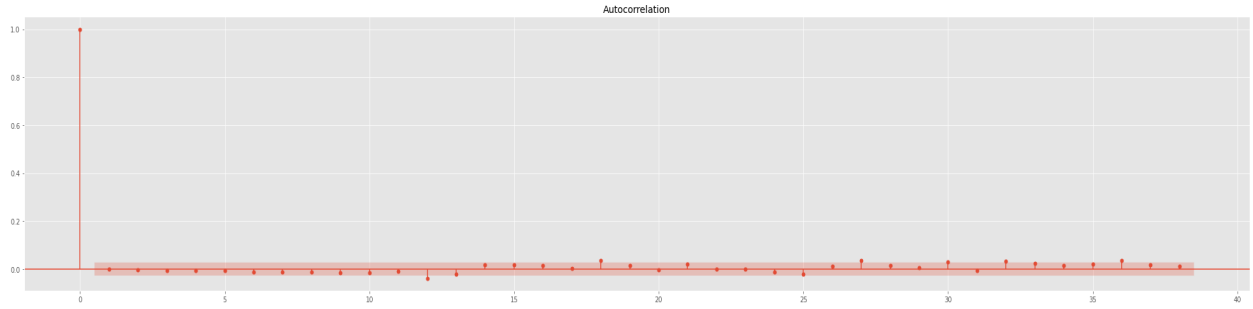
$$\theta_p(B)x_t = (1 - \alpha_1 B - \alpha_2 B^2 - .. - \alpha_p B^p)x_t = w_t$$

where B is the backshift operator and $w_t$ is white noise.

According to the AR model, a realisation at time t is a linear combination of the p previous realisations plus a noise expression. As stated above, the PACF plot suggested the use of lag value as 15, instead value of 12 was taken for analysis. The MAPE value obtained for Rajasthan came out to be 24.79%, and for Madhya Pradesh, Tamil Nadu and Andhra Pradesh the values obtained were 21.99%, 16.12% and 15.06% respectively. Interestingly the states situated away from the coastline had MAPE's in the 20s, and a more accurate result was obtained for those which were closer to the coastline. In case of Rajasthan data, the p-value(Check FIG.11) for the lag coefficients came out to be insignificant for certain lags, and the result could be also verified from the PACF plots too where the lag value wasn't that significant but still was taken into account.

An ACF plot of residuals was also obtained. Ideally all the points should be inside the threshold region, however very few points could be seen above the region indicating that the correlations aren't insignificant. This suggested that the AR model worked pretty well on the data, taking into consideration we had to adjust for lower lag values than the ideal one due to long computational times.

FIGURE 9 : ACF PLOT OF RESIDUALS ( AR )

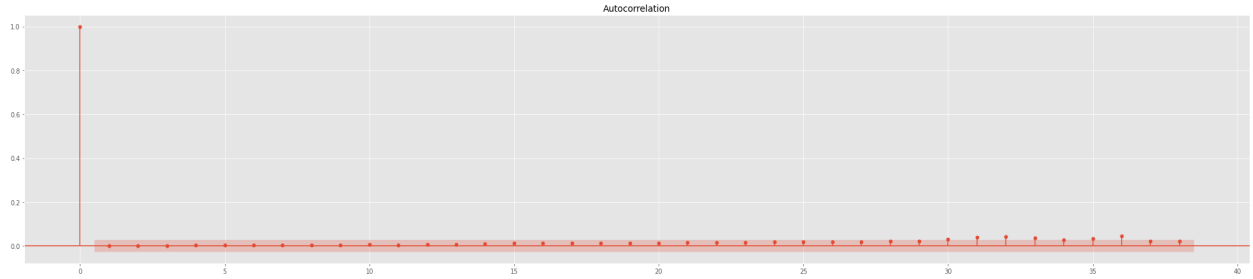



**MA**: As opposed to AR which uses past values of forecast variables in time series for regression analysis, the Moving Average(MA) model uses past forecast errors in a regression-like model. An MA(q) process can be expressed as follows

$$x_t = (1 + \beta_1 B + \beta_2 B^2 + ..\beta_q B^q)wt = \varphi_q(B)wt$$

where $\varphi_q$ is a polynomial of order q and B is the backshift operator. Here, each value of xt can be thought of as a weighted moving average of the past q forecast errors.
We found the value of q to be around 60 according to the ACF plot earlier, however this increases the complexity of our computation and leads to large training time. For simplification we are using the value of q to as 30. We found the MAPE to be 24.935%. And for other states as well, the MAPE values were found to differ from the MAPE values obtained in the AR model by +/- 0.5%. The following ACF plot of residuals signifies that all the points are well within the threshold region.

FIGURE 10 : ACF PLOT OF RESIDUALS ( MA )




The figure shown in the next page gives the summary statistics retrieved after fitting the MA model on the data.

FIGURE 11: SUMMARY STATISTICS FOR MA

SARIMAX Results

| | | | |
|---|---|---|---|
| Dep. Variable: | Wind Speed | No. Observations: | 5205 |
| Model: | ARIMA(0, 0, 30) | Log Likelihood | -6226.264 |
| Date: | Sun, 25 Apr 2021 | AIC | 12516.527 |
| Time: | 12:35:24 | BIC | 12726.363 |
| Sample: | 0 | HQIC | 12589.922 |
| | - 5205 | | |
| Covariance Type: | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 3.0388 | 0.062 | 48.703 | 0.000 | 2.916 | 3.161 |
| ma.L1 | 0.7836 | 0.013 | 61.649 | 0.000 | 0.759 | 0.809 |
| ma.L2 | 0.4459 | 0.016 | 27.144 | 0.000 | 0.414 | 0.478 |
| ma.L3 | 0.2859 | 0.018 | 15.721 | 0.000 | 0.250 | 0.322 |
| ma.L4 | 0.1826 | 0.018 | 9.937 | 0.000 | 0.147 | 0.219 |
| ma.L5 | 0.1542 | 0.019 | 8.326 | 0.000 | 0.118 | 0.190 |
| ma.L6 | 0.1451 | 0.018 | 7.913 | 0.000 | 0.109 | 0.181 |
| ma.L7 | 0.1436 | 0.019 | 7.672 | 0.000 | 0.107 | 0.180 |
| ma.L8 | 0.1766 | 0.019 | 9.343 | 0.000 | 0.140 | 0.214 |
| ma.L9 | 0.1749 | 0.019 | 9.022 | 0.000 | 0.137 | 0.213 |
| ma.L10 | 0.1397 | 0.019 | 7.229 | 0.000 | 0.102 | 0.178 |
| ma.L11 | 0.1507 | 0.019 | 7.740 | 0.000 | 0.113 | 0.189 |
| ma.L12 | 0.1380 | 0.020 | 7.025 | 0.000 | 0.099 | 0.177 |
| ma.L13 | 0.1220 | 0.019 | 6.299 | 0.000 | 0.084 | 0.160 |
| ma.L14 | 0.1379 | 0.020 | 7.030 | 0.000 | 0.099 | 0.176 |
| ma.L15 | 0.1378 | 0.019 | 7.096 | 0.000 | 0.100 | 0.176 |
| ma.L16 | 0.1294 | 0.020 | 6.622 | 0.000 | 0.091 | 0.168 |
| ma.L17 | 0.1111 | 0.020 | 5.660 | 0.000 | 0.073 | 0.150 |
| ma.L18 | 0.1276 | 0.019 | 6.614 | 0.000 | 0.090 | 0.165 |
| ma.L19 | 0.1206 | 0.019 | 6.219 | 0.000 | 0.083 | 0.159 |
| ma.L20 | 0.0976 | 0.019 | 5.138 | 0.000 | 0.060 | 0.135 |
| ma.L21 | 0.0996 | 0.019 | 5.249 | 0.000 | 0.062 | 0.137 |
| ma.L22 | 0.0825 | 0.019 | 4.250 | 0.000 | 0.044 | 0.121 |
| ma.L23 | 0.0675 | 0.019 | 3.490 | 0.000 | 0.030 | 0.105 |
| ma.L24 | 0.0418 | 0.019 | 2.200 | 0.028 | 0.005 | 0.079 |

The p values obtained for each coefficient could be seen from the fifth column. WE can draw a simple conclusion that the first 24 lag coefficients are significant for the model, which could be also verified from the ACF plot .

**ARMA:** AutoRegressive Moving Average or ARMA, as the name suggests, is a combination of the previous two models, AR and MA. It is used to describe a stationary time series in terms of two component polynomials, one of which corresponds to autoregression and the other to moving averages. The model has two parameters, p and q, for Auto-regression and Moving Averages respectively. ARMA (p,q) model is as follows

$$\theta_p(B)x_t = \varphi_q(B)w_t$$

where $\theta_p$ is a polynomial of order p and $\varphi_q$ is a polynomial of order q.

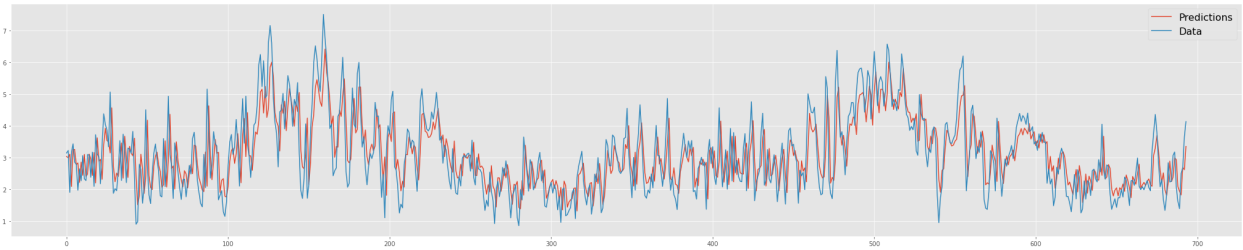
An iterative search for the parameters p and q was carried out each in the range of 1 to 20 and the best values obtained were 12 and 10 respectively. The model turned out to be a decent fit for the time series, as the p-values for most among the 22 coefficients were less than 0.05; this argument is further strengthened by the ACF plot of the residuals(Check Appendix).This was also suitable for both Rajasthan and Tamil Nadu.

For Andhra Pradesh and Madhya Pradesh, best fit values were obtained at (8,0,10) and (10,0,12) respectively. The MAPE for the following states were :

| | |
|---|---|
| Rajasthan | 18.429% |
| Tamil Nadu | 15.035% |
| Andhra Pradesh | 14.765% |
| Madhya Pradesh | 20.732% |

As discussed before, owing to heavy computational costs, the grid search for the best parameters for the rolling forecast was done in a much smaller space. This resulted in the values of p and q as 3 and 1 respectively. The daily forecasting gave us an MAPE of 18.429% while the weekly and monthly forecast gave MAPEs of 30.48% and 32.53% respectively.

FIGURE 12: DAILY FORECASTING (ARMA) (Rajasthan (12,0,10)




For monthly and weekly Forecasting, we split our data for training and testing purposes. The last two years of data was taken as testing data. Since Rolling forecast consumes a lot of time even for low values of p and q, hence Rolling forecast was carried out. The plots for weekly and Monthly ARMA prediction have been shown below:

FIGURE 13: WEEKLY FORECASTING (ARMA)

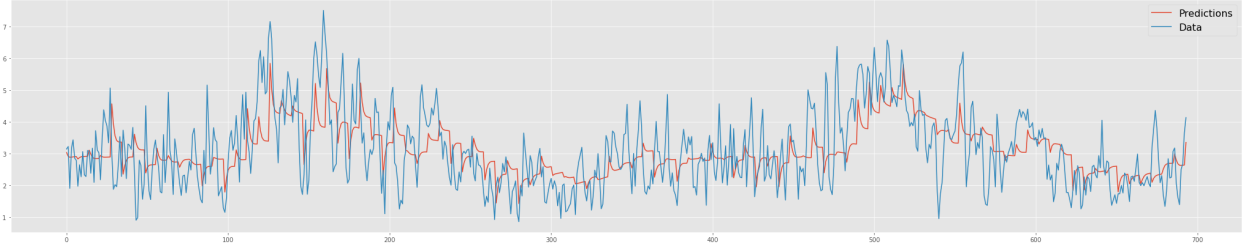
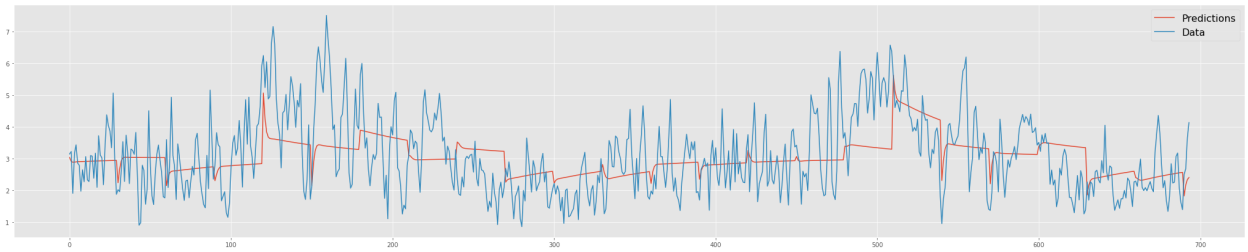


FIGURE 14 : MONTHLY FORECASTING (ARMA)




**ARIMA:** ARIMA is a generalisation of ARMA, which stands for AutoRegressive Integrated Moving Average. Unlike ARMA, which can only be used for stationary models, ARIMA is used for non-stationary models. It accomplishes this by converting the non-stationary model into a stationary one by differencing.. Mathematically, d order differencing is of the form

$$(1 - B)^d x_t$$

where B is the backshift operator.If differencing by order d produces white noise, the integrated series is of order d. ARIMA(p, d, q) performs ARMA(p, q) on data that has been integrated by order d, which signifies the added I - for integration. This model can be succinctly expressed by the equation

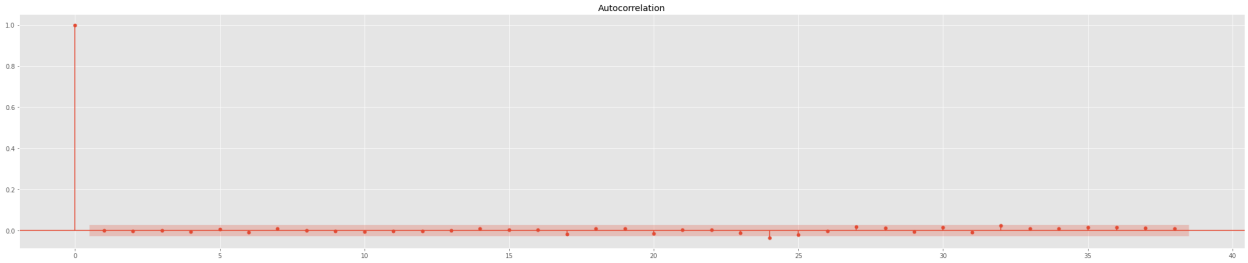
$$\theta_p(B)(1 - B)^d x_t = \varphi_q(B)w_t$$

where $\theta_p$ and $\varphi_q$ are polynomials of order p and q respectively.

Though the data is already confirmed to be stationary by our previous tests, we still performed ARIMA for testing purposes. We performed a grid search for parameters with $d \geq 1$ which best fit our model. We found that an order of (12, 1, 10) fit our model best(MAPE 24.725%). Similarly for Tamil Nadu, Andhra Pradesh and Madhya Pradesh, best orders were found to be (8,1,10), (8,1,10) and (12,1,10) respectively which resulted in insignificant correlation in the residuals.

The plot for ACF (Residuals) has been shown below:

FIGURE 15: ACF PLOT FOR RESIDUALS (ARIMA)




However, this model is too computationally intensive to use on a rolling forecast. Instead, we used an order of (3, 1, 1) for our rolling forecast, giving us an MAPE as following:

| States | Daily | Weekly | Monthly |
|---|---|---|---|
| Rajasthan | 24.725% | 31.65% | 33.791% |
| Tamil Nadu | 16.033% | 27.523% | 34.023% |
| Andhra Pradesh | 15.187% | 25.698% | 33.67% |
| Madhya Pradesh | 21.943% | 31.72% | 34.308% |

**SARIMA:** To remove additive seasonal effects, the seasonal ARIMA model uses differencing at a lag equal to the seasonality. At the same lag, it also adds autoregressive and moving average terms, resulting in the order being represented as (p, d, q)(P, D, Q)s, where s is the seasonality. This can be expressed mathematically as

$$\Theta_P(B^s)\theta_p(B)(1-B^s)^D(1-B)^d x_t = \Phi_Q(B^s)\varphi_q(B)w_t$$

where $\Theta_P$ , $\theta_p$, $\Phi_Q$, $\varphi_q$ are polynomials of orders P, p, Q and q respectively.

There are four seasonal elements that are not part of ARIMA that must be configured; they are:

- **P:** Seasonal autoregressive order.
- **D:** Seasonal difference order.
- **Q:** Seasonal moving average order.
- **m:** The number of time steps for a single seasonal period.

## Selection of Model and Analysis for Other States

SARIMA could be the best model for Time Series Forecasting of our data based on the MAPE values obtained for all of the above models. We simply couldn't afford regular time series prediction with SARIMA because of the incredibly high computing expense. As a result, we choose ARMA as the most suitable model and our preferred model for Time Series forecasting in other states.

## CONCLUSION

We used exploratory data analysis on the dataset and validated various regression models for forecasting in our analysis. In our research, we discovered that ARMA generated the best results

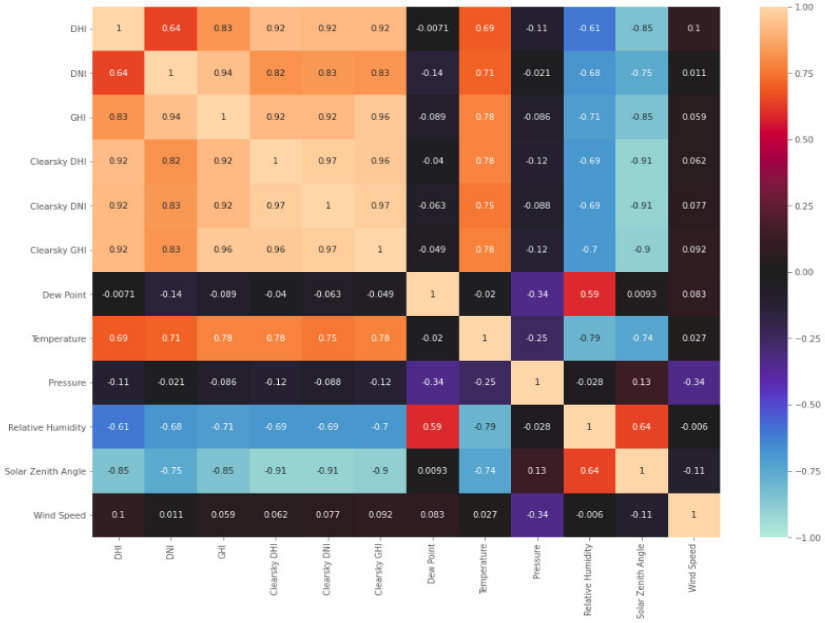
while remaining relatively simple to compute. We have tested our trained model on the rest of our dataset's states.

The findings were less than optimal, which was to be expected given the seasonal and climatic differences between the states. Based on the findings, we hope that the SARIMA model may be a better fit for our time series. However, owing to lack of adequate computer power to train a SARIMA model for regular forecasting, the modelling could not be completed due to the long duration.

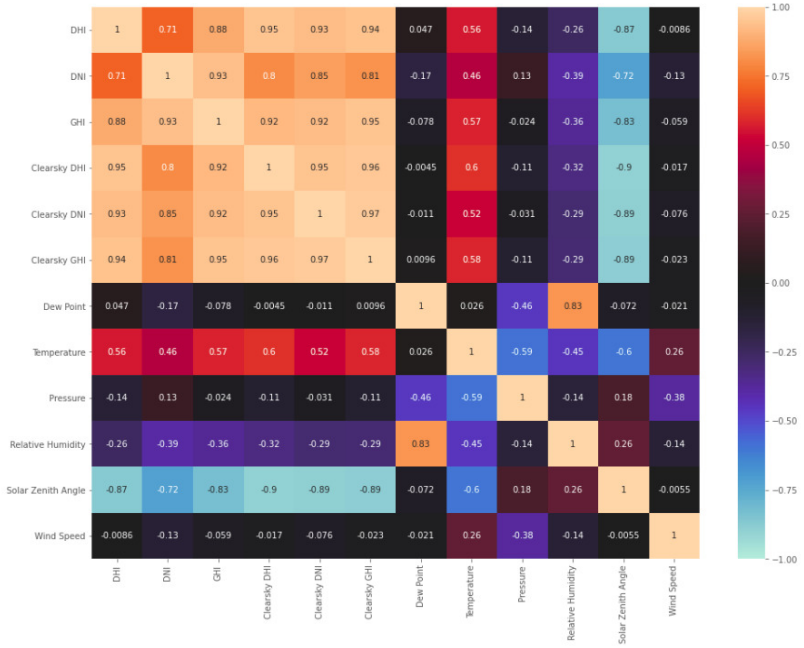
## APPENDIX

### CORRELATION HEATMAPS

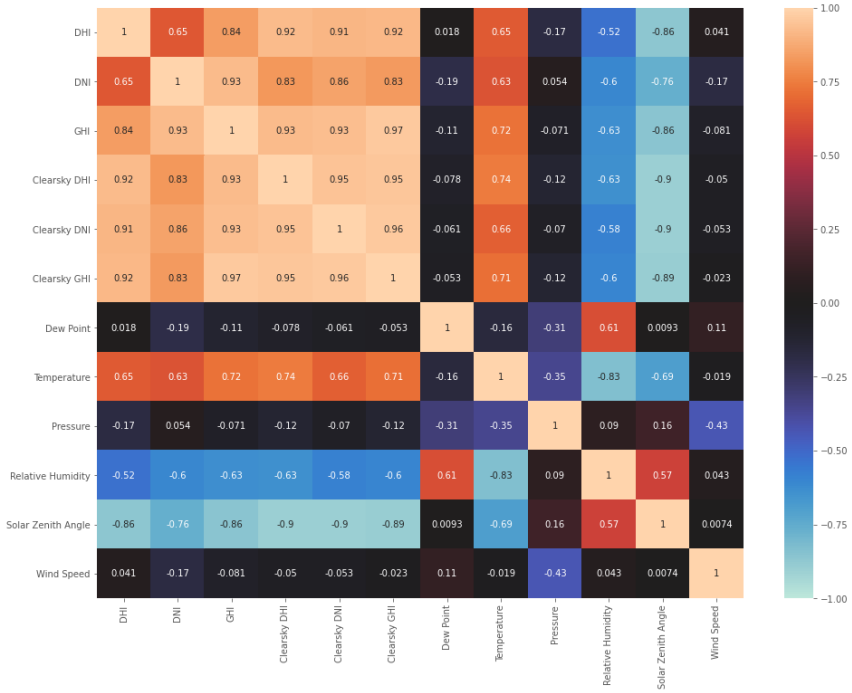
### TAMIL NADU




## MADHYA PRADESH

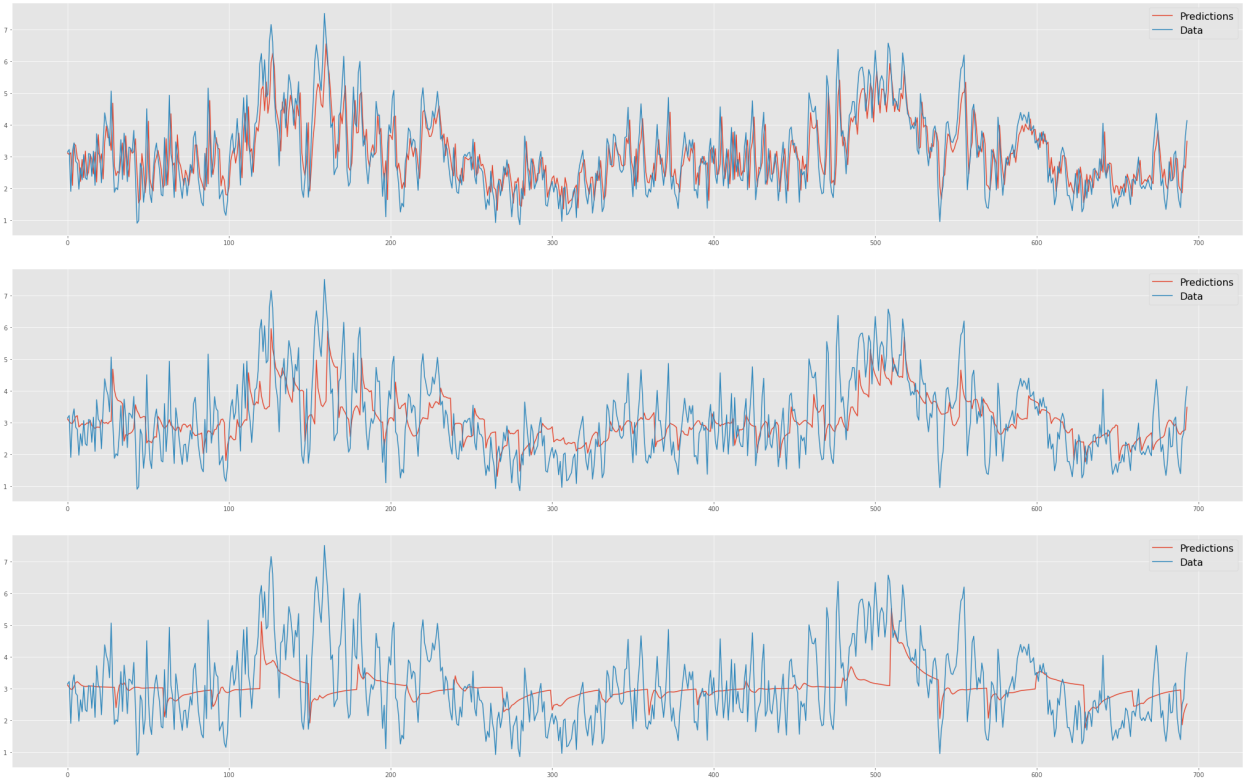


# ANDHRA PRADESH

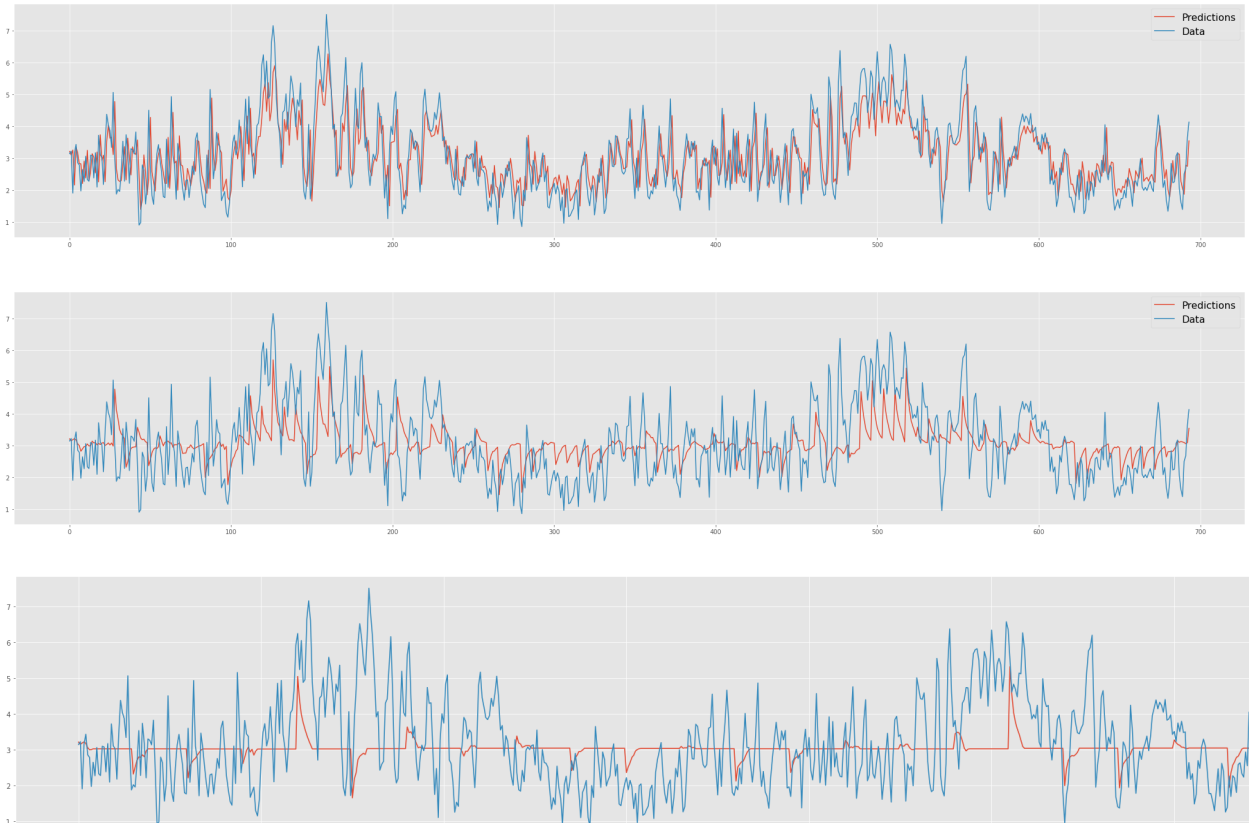



## Forecasts :

## AR : DAILY ,WEEKLY AND MONTHLY RAJASTHAN (12)

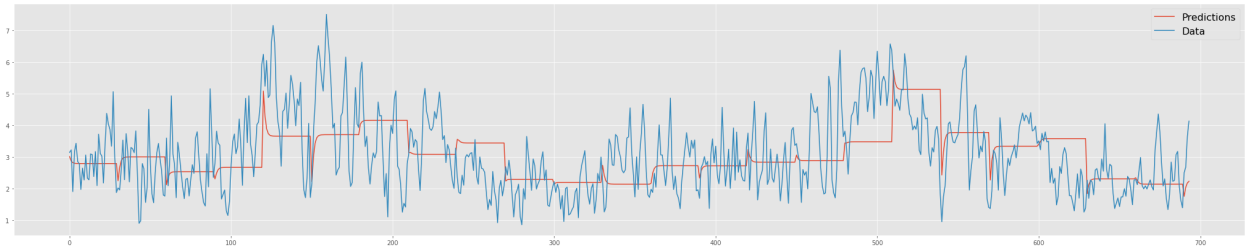


## MA : DAILY, WEEKLY AND MONTHLY  RAJASTHAN (30)

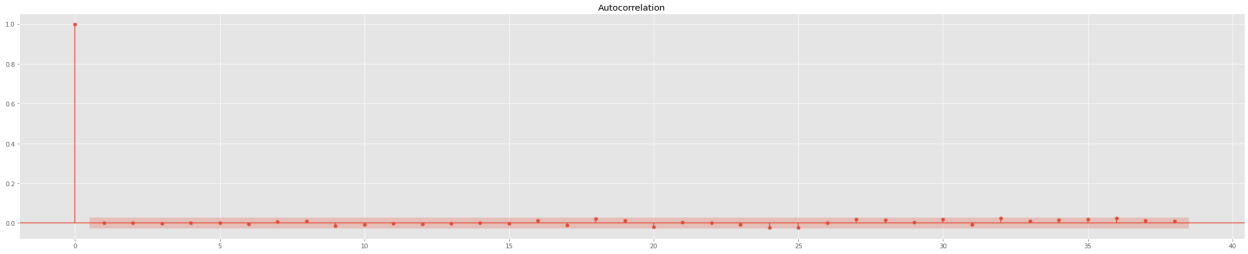





## ARIMA : DAILY, WEEKLY AND MONTHLY RAJASTHAN  (12,1,10)




**RESIDUAL PLOT FOR ARMA (RAJASTHAN)**




## CODE SNIPPET FOR ROLLING FORECASTING ( FOR DAILY DATA, ARIMA MODEL )

```
# Rolling forecast for ARIMA
history = daily_RJ['Wind Speed']

predictionsARIMA = []
weeklyARIMA = []
monthlyARIMA = []
for t in range(len(test_data)):
  modelARIMA = ARIMA(history[:train_data_len+t],order = (3,1,1))
  model_fitARIMA = modelARIMA.fit()

  if (t)%7==0:
    forecast = model_fitARIMA.predict(start=t+train_data_len, end=t+train_data_len+6)
    # print(forecast)
    for i in range(7):
      weeklyARIMA.append(forecast[train_data_len+t+i])

  if (t)%30==0:
    forecast = model_fitARIMA.predict(start=t+train_data_len, end=t+train_data_len+29)
    # print(forecast)
    for i in range(30):
      monthlyARIMA.append(forecast[train_data_len+t+i])

  output = model_fitARIMA.forecast()
  predVal = output[train_data_len+t]
  predictionsARIMA.append(predVal)
  # history[len(history)] = test_data[train_data_len+t]
```

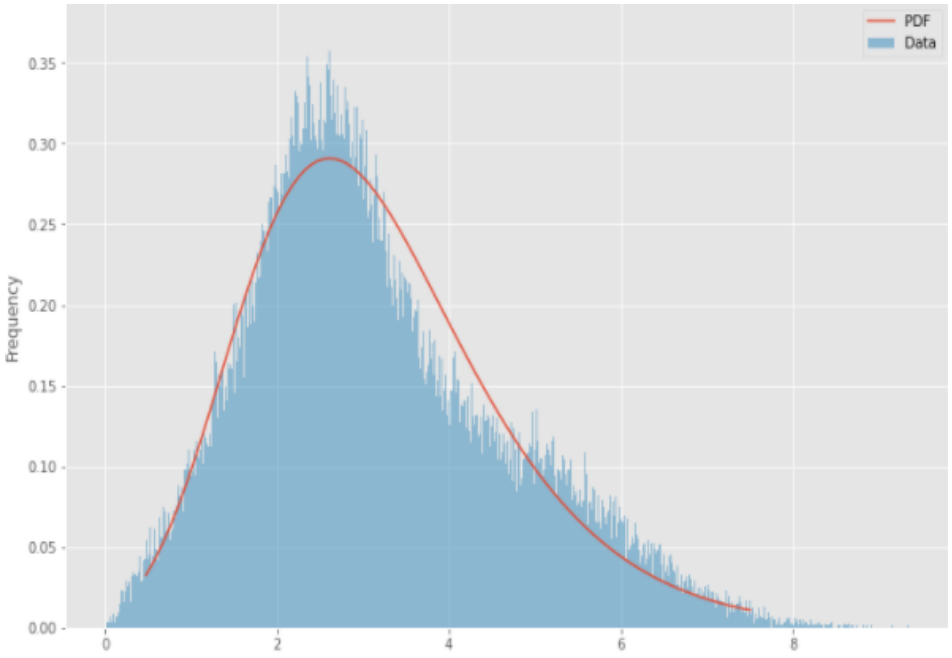
```
percentage = []
for i in range(len(predictionsARIMA)):
  percentage.append((abs(predictionsARIMA[i]-test_data[train_data_len+i])/(test_data[train_data_len+i]))*100)
mape = sum(percentage)/len(percentage)
print("Mean Absolute Percentage Error: {0} %".format(mape))

test_points = [x for x in test_data]

mse = mean_squared_error(test_points,predictionsARIMA)
print("Mean Sqaured Error: {0}".format(mse))
```

## CODE SNIPPET FOR CARRYING OUT BEST FIT DISTRIBUTION

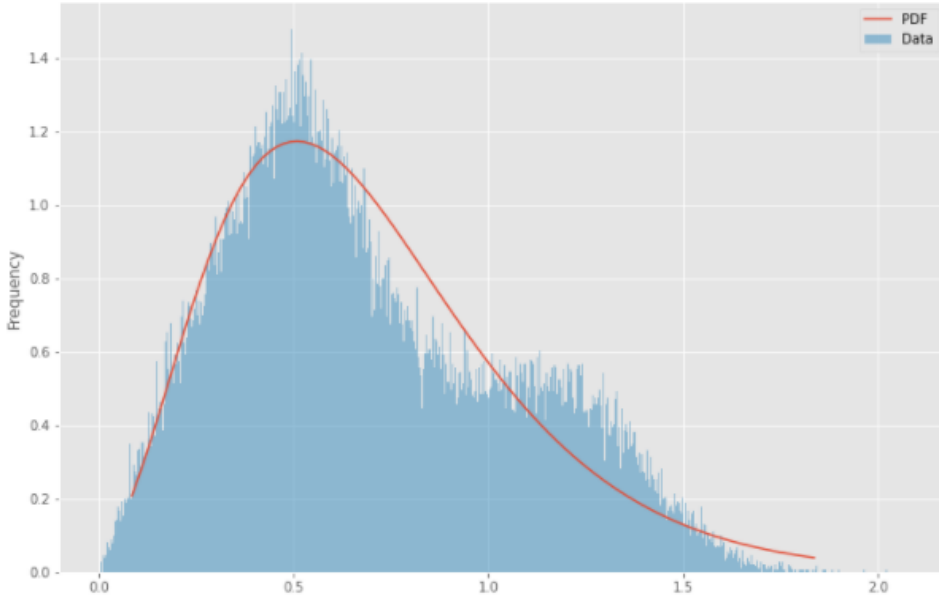
```
def best_fit_distribution(data, bins=200, ax=None):
    """Model data by finding best fit distribution to data"""
    # Get histogram of original data
    y, x = np.histogram(data, bins=bins, density=True)
    x = (x + np.roll(x, -1))[:-1] / 2.0
    # Distributions to check
    DISTRIBUTIONS = [
        st.dgamma,st.dweibull, st.alpha,st.weibull_min,st.gamma,st.norm,st.expon,st.rayleigh,st.weibull_max
    ]

    best_distribution = st.norm
    best_params = (0.0, 1.0)
    best_sse = np.inf
    for distribution in DISTRIBUTIONS:
        try:
            with warnings.catch_warnings():
                warnings.filterwarnings('ignore')
                params = distribution.fit(data)
                arg = params[:-2]
                loc = params[-2]
                scale = params[-1]
                # Calculate fitted PDF and error with fit in distribution
                pdf = distribution.pdf(x, loc=loc, scale=scale, *arg)
                sse = np.sum(np.power(y - pdf, 2.0))
                # if axis pass in add to plot
                try:
                    if ax:
                        pd.Series(pdf, x).plot(ax=ax)
                    end
                except Exception:
                    pass
                # identify if this distribution is better
                if best_sse > sse > 0:
                    best_distribution = distribution
                    best_params = params
                    best_sse = sse

        except Exception:
            pass

    return (best_distribution.name, best_params)

def make_pdf(dist, params, size=100):
    """Generate distributions's Probability Distribution Function """
    # Separate parts of parameters
    arg = params[:-2]
    loc = params[-2]
    scale = params[-1]
    # Get sane start and end points of distribution
    start = dist.ppf(0.01, *arg, loc=loc, scale=scale) if arg else dist.ppf(0.01, loc=loc, scale=scale)
    end = dist.ppf(0.99, *arg, loc=loc, scale=scale) if arg else dist.ppf(0.99, loc=loc, scale=scale)
    # Build PDF and turn into pandas Series
    x = np.linspace(start, end, size)
    y = dist.pdf(x, loc=loc, scale=scale, *arg)
    pdf = pd.Series(y, x)
    return pdf
```

**SUMMARY OF BEST FIT DISTRIBUTIONS FOR STATES**

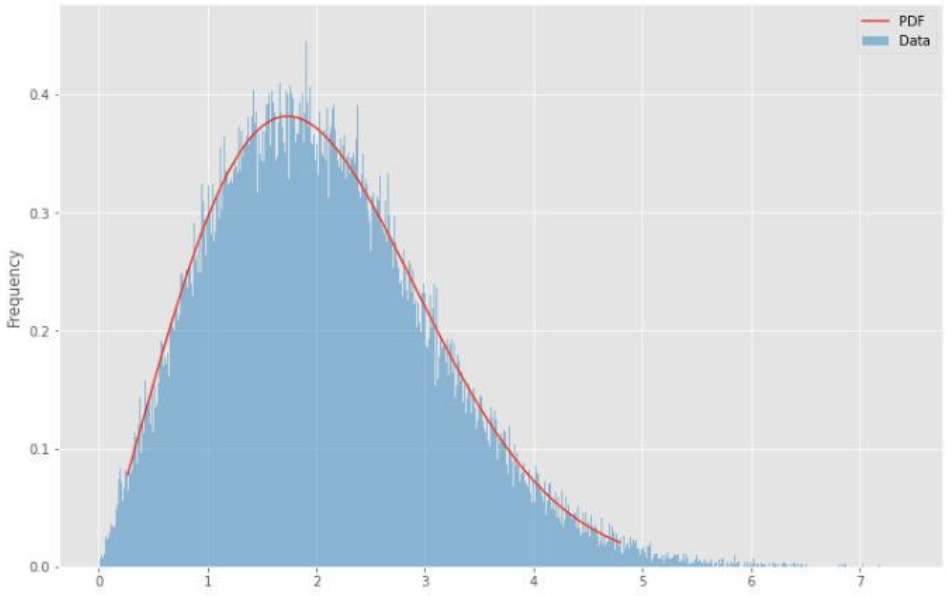
**1) ANDHRA PRADESH ( WEIBULL_MAX [c=14.28, loc=20.62, scale=18.11])**




**2) TAMIL NADU ( GAMMA[a=4.19, loc = -0.08 , scale = 0.19])**



**3) MADHYA PRADESH ( WEIBULL_MIN[c=2.13, loc=-0.01, scale=2.35])**



**REFERENCES**

**[1]** D.R. Anderson et al. *Statistics for Business & Economics*. Cengage Learning, 2013.

**[2]** National Renewable Energy Laboratory (NREL). Solar Resource Glossary. 2020.
URL : https://www.mnre.gov.in/

**[3]** Significance of ACF & PACF plots in Time Series Analysis.
URL:https://towardsdatascience.com/significance-of-acf-and-pacf-plots-in-time-series-analysis-2fa11a5d10a8

**[4]** Terence C. Mills *Applied Time Series Analysis : A Practical Guide to Modeling and Forecasting.* Academic Press, 2019. https://www.elsevier.com/books-and-journals

**[5]** *Finding empirical distribution to theoretical ones with Scipy (Python) ?*.
URL:https://stackoverflow.com/questions/6620471/fitting-empirical-distribution-to-theoretical-ones-with-scipy-python

**[6]** A gentle introduction to Partial Autocorrelation and AutoCorrelation .
URL: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

[7] Energy Education, Wind Power.
URL:https://energyeducation.ca/encyclopedia/Wind_power#:~:text=Several%20different%20factors%20influence%20the,air%20density%2C%20and%20blade%20radius.