|  |
| --- |
| Vellore Institute of Technology, Chennai    DATABASE SYSTEMS: DESIGN AND IMPLEMENTATION  CSE5003  **Project on Wind Data Cleaning and Prediction**  Submitted by  Kalpalathika N 20MAI1006  Pattan Afrid Ahmed 20MAI1016  Poorvaja P 20MAI1009    MTECH CSE (AI and ML) |

Table of Contents

[Abstract 3](#_Toc56855841)

[Introduction 4](#_Toc56855842)

[Requirements 5](#_Toc56855843)

[Description of Data set 6](#_Toc56855844)

[Literature review 7](#_Toc56855845)

[Modules 8](#_Toc56855846)

[Wind data cleaning module 8](#_Toc56855847)

[Database module 10](#_Toc56855848)

[Prediction module 12](#_Toc56855849)

[Long Short-Term Memory (LSTM) 12](#_Toc56855850)

[Random Forest Regressor (RFR) 12](#_Toc56855851)

[Fbprophet (FP) 12](#_Toc56855852)

[Auto Regressive Integrated Moving Average (ARIMA) 13](#_Toc56855853)

[Conclusion 21](#_Toc56855854)

[References 22](#_Toc56855855)

# Abstract

Renewable energy resources are environmentally clean and found abundant in nature. To increase green energy penetration and reduce global warming, Wind energy is one of the promising technologies that is developing and promising to satisfy our future energy needs. In this project, the data collected from the mast are analyzed for missing or erroneous values. Data cleaning is done to improve the quality of that data and use the cleaned data for wind speed prediction to improve the accuracy and reliability in the result and to generate short-term forecast of the wind speed by using Autoregressive Integrated Moving Average (ARIMA), fbprophet, Long-Short Time Memory(LSTM) models.

# Introduction

Renewable energy sources are available in abundance naturally and can be utilized for power generation to satisfy industrial and commercial needs. The renewable sector is witnessing a phenomenal growth, and accurate prediction is essential. Wind energy is proposed as an important source of alternative energy in recent years. It has more advantages with respect to other sources in terms of installation and generation cost. In recent years, the renewable nature of wind power makes it more attractive. Nowadays, many countries face lots of problems with their energy sources due to many environmental factors and the uncertainties of origin. The power generated from wind energy is better than other energy sources. The wind turbine layout uses advanced technologies to decrease the cost of wind power generation and permit large-scale integration into the energy grid. The primary input needed for producing wind power generation is the natural wind. When the wind moves through wind turbines, the turbines convert the wind energy into mechanical energy. Then, the wind power is produced from that mechanical energy. The advantages of wind energy are combined with a few difficulties like high uncertainty, limited predictability and wind power energy is not entirely deliverable.

Accurate and reliable wind speed prediction is vital for wind farm planning and operational planning for electrical networks. To improve the accuracy of wind speed prediction, many forecasting approaches have been proposed; however, these models typically do not account for the importance of data pre-processing and are limited by the use of individual models. In this project, we used real-time data for cleaning and prediction. Dataset contains aesthetics like Date/time, pressure, and temperature, wind speed at different heights of 50m, 80m, and 100 m. We consider the mean of wind speed at different heights. For cleaning, we have used interpolation method. For prediction, we labeled the wind speed. And after that we are storing tit in the database and retrieving it for wind speed prediction.

# Requirements

|  |  |  |
| --- | --- | --- |
| Software | Python, PostgreSQL |  |
| Data | Renewable energy data (Wind data sources are getting identified) | 10 min of data for around 2 years are used. |

# Description of Data set

In this project, the data set used is from measurement mast, the wind data has 10 mins interval and contains Date and Time, mean and standard deviation of the wind speed at different heights, pressure and temperature. The period of the data is 2 year. (i.e.) from 2013 October to 2015 September.



**Figure 1 Photo of the mast structure (Source: Wikipedia)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date/Time** | **WS\_50\_mean** | **WS\_50\_stdev** | **WS\_80\_mean** | **WS\_80\_stdev** | **WS\_100\_mean** | **WS\_100\_stdev** | **Ta\_10\_mean** | **Pbaro\_5\_mean** |
| 1/10/2013 10:00 | 5.62 | 0.43 | 5.71 | 0.43 | 5.61 | 0.43 | 27.81 | 1004.2 |
| 1/10/2013 10:10 | 5.72 | 0.48 | 5.85 | 0.5 | 5.85 | 0.43 | 27.82 | 1004.1 |
| 1/10/2013 10:20 | 6 | 0.36 | 6.05 | 0.46 | 6.05 | 0.37 | 27.76 | 1003.9 |
| 1/10/2013 10:30 | 6.49 | 0.41 | 6.58 | 0.53 | 6.46 | 0.43 | 27.77 | 1003.9 |
| 1/10/2013 10:40 | 6.74 | 0.45 | 6.82 | 0.56 | 6.86 | 0.39 | 27.8 | 1003.8 |
| 1/10/2013 10:50 | 7.15 | 0.38 | 6.45 | 0.66 | 7.17 | 0.38 | 27.87 | 1003.6 |
| 1/10/2013 11:00 | 7.13 | 0.44 | 6.23 | 0.59 | 7.25 | 0.41 | 27.9 | 1003.5 |
| 1/10/2013  11:10 | 7.16 | 0.58 | 6.31 | 0.56 | 7.28 | 0.49 | 27.95 | 1003.4 |

**Table 1 Sample Mast Data**

# Literature review

1. In recent years, clean energies, such as wind power have been developed rapidly. Especially, wind power generation becomes a significant source of energy in some power grids. On the other hand, based on the uncertain and non-convex behaviour of wind speed, wind power generation forecasting and scheduling may be very difficult. In this paper, to improve the accuracy of forecasting the short-term wind speed, deep learning time series prediction based on Long Short Term Memory neural networks (LSTM). The proposed wind speed forecasting strategy is applied to real-life data from Sotavento that is located in the south-west of Europe, in Galicia, Spain, and Kerman that is located in the Middle East, in the southeast of Iran. The presented numerical results demonstrate the efficiency of the proposed method, compared to some other existing wind speed forecasting methods.
2. Wind speed forecasting models and their application to wind farm operations are attaining remarkable attention in the literature because of its benefits as a clean energy source. In this paper, we suggested the time series machine learning approach called random forest regression for predicting wind speed variations. The computed values of mutual information and auto-correlation shows that wind speed values depend on the past data up to 12 hours. The random forest model was trained using ensemble from two weeks data with previous 12 hours values as input for every value. The computed root mean square error shows that model trained with two weeks data can be employed to make reliable short-term predictions up to three years ahead.
3. One step ahead wind speed forecasting models were compared. A univariate model was developed using a linear autoregressive integrated moving average (ARIMA). This method’s performance is well studied for a large number of prediction problems. The main objective was to compare the impact of the various meteorological variables on the performance of the multivariate model (NARX) of wind speed prediction with respect to the high performance univariate linear model.

# Modules

## Wind data cleaning module

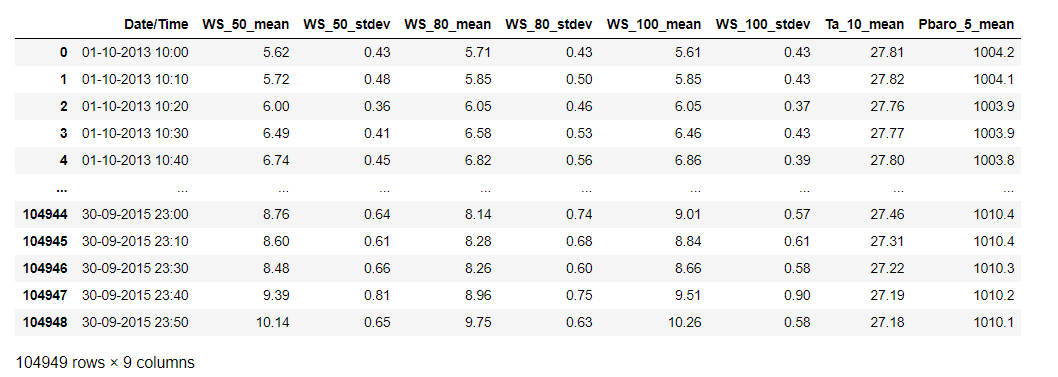
Aim of this project is to clean the missing values using pandas library in python. To deal with missing values, either we can use fillna(), interpolate(), imputer()methods.

*pandas* library has “interpolate()” function with *padding* method for missing values. Interpolation through padding means copying the value just before a missing/error entry.

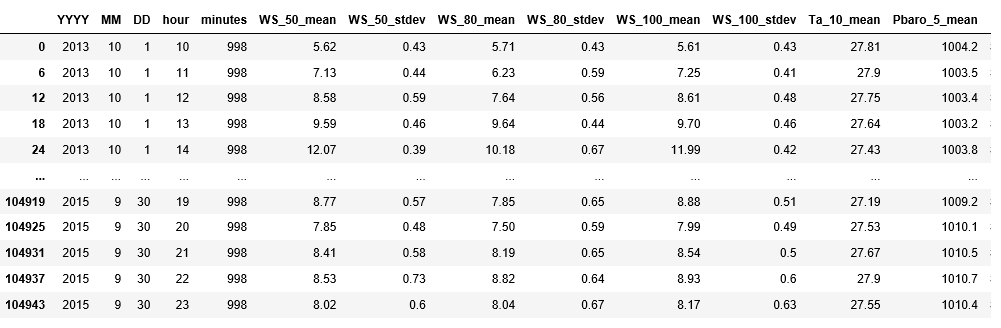
In first, loaded the excel file as dataset and storing the original data in the database. Then identify the columns with missing/error values and replaces it with the error code based on the error value.

Identified error rows with error status are stored separately in another table for future reference. Then performed interpolation on the dataset and the dataset is cleaned now.

Store the cleaned dataset into the database and retrieve it whenever needed to perform wind speed prediction for that dataset.



*Figure 2 Uncleaned Dataset*



*Figure 3 Error Log Dataset*

# 

*Figure 4 Cleaned Dataset*

## Database module

Using PostgreSQL for storing wind data, the tables are created and the values are inserted by connecting python and PostgreSQL.

Created three tables: 1. uncleaned data table

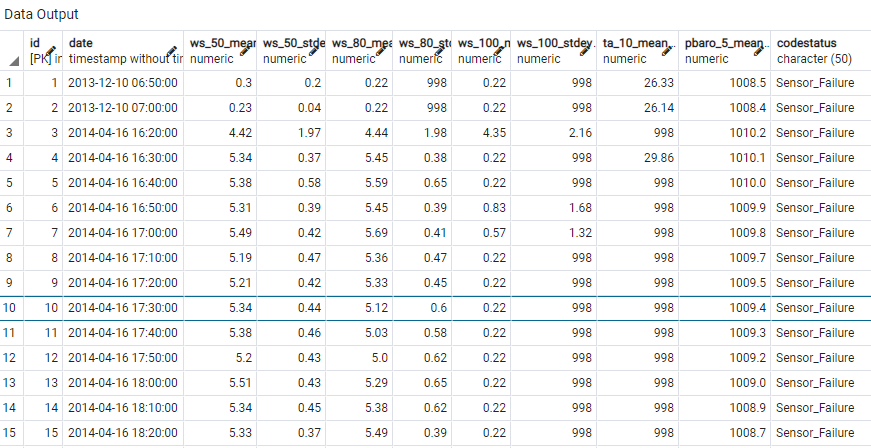
2. Error Log table and

3.cleaned data table

The Uncleaned data table consists of original data dataset which contains missing or error values. The Error log contains the error rows with error code status for future reference. The cleaned data table consists of the data with interpolated values.

## 

*Figure 5 Original Data Table*



*Figure 6 Error Log Data Table*

# 

*Figure 7 Cleaned Data Table*

## Prediction module

Mainly focusing on Autoregressive Integrated Moving Average (ARIMA), fbprophet and Long-short Time Memory (LSTM), Random Forest Regressor (RFR) prediction models.

* Fbprophet is improvement to the ARMA/ARIMA model.
* It has inbuilt functions for seasonality and trends.
* By training the model with proper tuning, we predict the wind speed.
* When it comes to LSTM, it’s special kind of RNN which overcome the vanishing gradient problem.

### Long Short-Term Memory (LSTM)

The long short-term memory network (LSTM), which is a variant of RNN. An LSTM unit shown in Figure 1 is composed of a cell, an input gate, an output gate, and a forget gate. The unique structure of LSTM can effectively solve the problems of gradient disappearance and gradient explosion problems in the training process of RNN.

### Random Forest Regressor (RFR)

Random forest is a non-parametric ensemble based learning technique used for both classification and regression problem. It is an extended version of decision tree algorithm which works on a set of rules and the possible outcomes to form a tree-like structure. For an incorrect rule adds the impurity to the subsequent nodes, a high risk of error propagation is always associated with decision trees.

Random forest algorithm eliminates error diffusion property inherent in decision trees by constructing multiple decision trees. Random samples of given data set are generated and fed to several tree-based learners to form a random forest. Splitting condition for each node in a tree is based on only the randomly selected predictor attributes which lower the error rate by avoiding the correlation among the trees.

Random forest regression is a non-parametric regression technique in which the functional relation between dependent and independent variables are captured from the features of the data.

In simple words random forest algorithm can be explained by a three-step procedure as follows:

1. From the given data select random samples with replacement

2. At each level, split node properly to get the best split until a maximum level of tree is obtained

3. Repeat the second step until a satisfied number of trees are generated.

### Fbprophet (FP)

To use Prophet for forecasting, first, a Prophet() object is defined and configured, then it is fit on the dataset by calling the fit() function and passing the data.

The fit() function takes a DataFrame of time series data. The DataFrame must have a specific format. The first column must have the name ‘ds‘ and contain the date-times. The second column must have the name ‘y‘ and contain the observations.

**Parameter Description**

yearly seasonality Fit yearly seasonality

weekly seasonality Fit weekly seasonality

daily seasonality Fit daily seasonality

change points List of dates at which to include potential changes.

The Prophet package provides intuitive parameters which are easy to tune. Even someone who lacks expertise in forecasting models can use this to make meaningful predictions for a variety of problems.

Using time as a regressor, Prophet is trying to fit several linear and non linear functions of time as components.

Usually, some popular error terms such Root Mean Square Error (RMSE) is used during the modeling evaluation.

### Auto Regressive Integrated Moving Average (ARIMA)

An ARIMA model is a class of statistical models for analyzing and predicting time series data.

It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

ARIMA is an acronym that stands for Auto Regressive Integrated Moving Average. It is a generalization of the simpler Auto Regressive Moving Average and adds the notion of integration.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA (p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

**p**: The number of lag observations included in the model, also called the lag order.

**d**: The number of times that the raw observations are differenced, also called the degree of differencing.

**q**: The size of the moving average window, also called the order of moving average.

**Root Mean Square Error** (**RMSE**) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; **RMSE** is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Based on a rule of thumb, it can be said that **RMSE** values between 0.2 and 0.5 shows that the model can relatively predict the data accurately.

****

**LSTM algorithm:**

# normalize the dataset

scaler = MinMaxScaler(feature\_range=(0, 1))

dataset = scaler.fit\_transform(dataset)

# split into train and test sets

train\_size = int(len(dataset) \* 0.80)

test\_size = len(dataset) - train\_size

train, test = dataset[0:train\_size,:], dataset[train\_size:len(dataset),:]

print(train, test)

# reshape into X=t and Y=t+1

look\_back = 6

trainX, trainY = create\_dataset(train, look\_back)

testX, testY = create\_dataset(test, look\_back)

# reshape input to be [samples, time steps, features]

trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# create and fit the LSTM network

epochs = 100

model = Sequential()

model.add(LSTM(units = 8, activation = 'relu', input\_shape = (1, look\_back)))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.fit(trainX, trainY, epochs=epochs, batch\_size=64, verbose=1)

# make predictions

trainPredict = model.predict(trainX, verbose = 1)

testPredict = model.predict(testX, verbose = 1)

print(trainPredict, testPredict)

# invert predictions

trainPredict = scaler.inverse\_transform(trainPredict)

trainYinv = scaler.inverse\_transform([trainY])

testPredict = scaler.inverse\_transform(testPredict)

testYinv = scaler.inverse\_transform([testY])

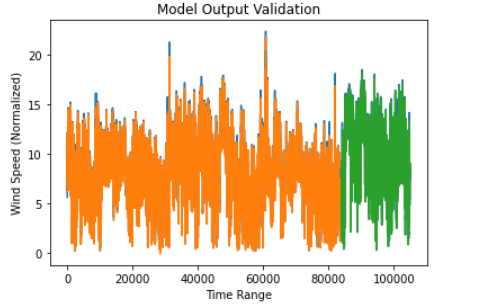
# calculate root mean squared error

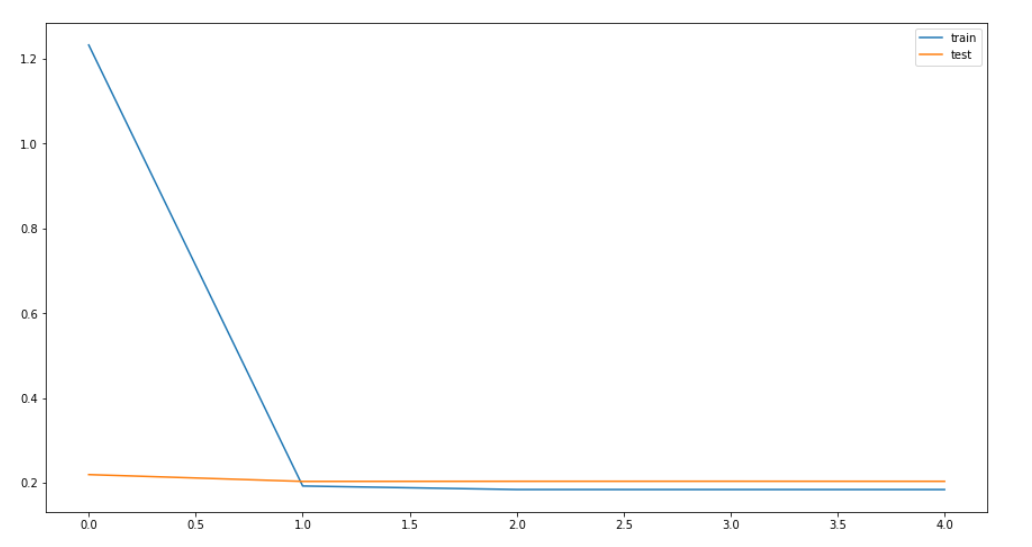
trainScore = math.sqrt(mean\_squared\_error(trainYinv[0], trainPredict[:,0]))

print('Train Score: %.3f RMSE' % (trainScore))

testScore = math.sqrt(mean\_squared\_error(testYinv[0], testPredict[:,0]))

print('Test Score: %.3f RMSE' % (testScore))





*Figure 8 LSTM Model*

**RandomForestRegressor:**

# input data for regressor. We split the wind speed data into five columns(T1, T2, T3, T4, T5)

training\_data = df[df.DATE\_TIME < pd.to\_datetime("30/04/2015")] # 2013 to 2015 apr

val\_mask = (df.DATE\_TIME >= pd.to\_datetime("01/05/2015")) & (df.DATE\_TIME < pd.to\_datetime("31/05/2015")) # 2015 april to may

test\_data = df[df.DATE\_TIME >= pd.to\_datetime("31/05/2015")]

clean\_train = training\_data[['T\_1','T\_2', 'T\_3', 'T\_4', 'T\_5','WS\_50\_mean']]

clean\_test = test\_data[['T\_1','T\_2', 'T\_3', 'T\_4', 'T\_5','WS\_50\_mean']]

clean\_val = val\_data[['T\_1','T\_2', 'T\_3', 'T\_4', 'T\_5','WS\_50\_mean']]

# train, test, validation data

X\_train,y\_train = clean\_train.drop(["WS\_50\_mean"],axis=1),clean\_train.WS\_50\_mean

X\_test,y\_test = clean\_test.drop(["WS\_50\_mean"],axis=1),clean\_test.WS\_50\_mean

X\_val,y\_val = clean\_val.drop(["WS\_50\_mean"],axis=1),clean\_val.WS\_50\_mean

# 100 estimators are divided, 50 parallel & fit for the RFR algorithm which classifies columns(T1, T2, T3, T4, T5)

scaler = StandardScaler()

rfr = RandomForestRegressor(random\_state=69,verbose=2,n\_jobs=50,n\_estimators=100)

X\_train\_scaled = scaler.fit\_transform(X\_train)

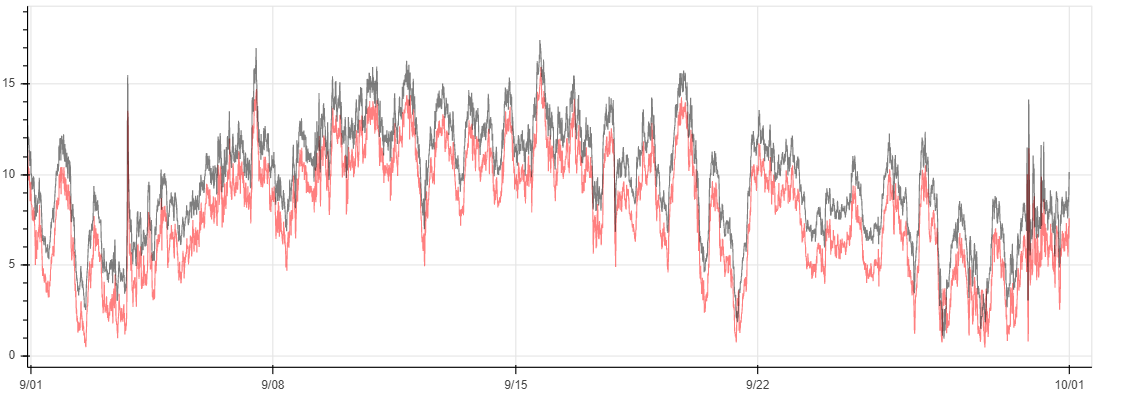
X\_test\_scaled = scaler.fit\_transform(X\_test)

X\_valid\_scaled = scaler.fit\_transform(X\_val)

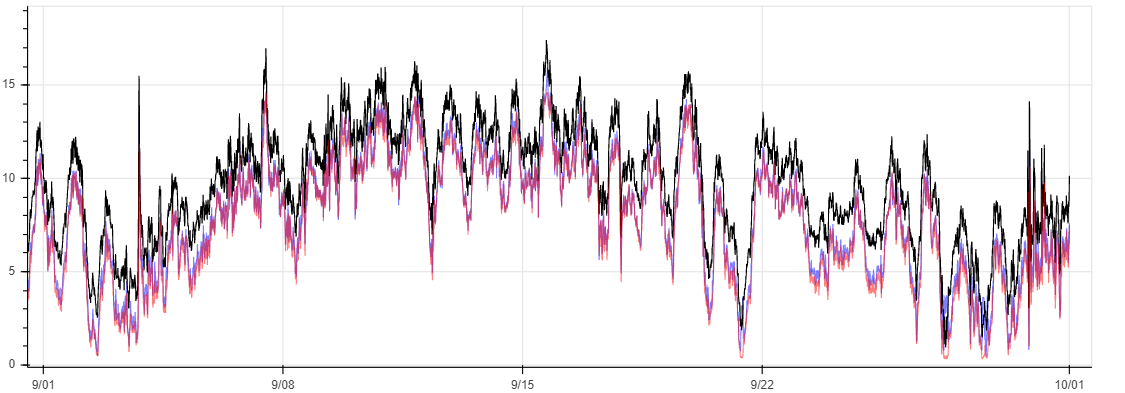
rfr.fit(X\_train\_scaled,y\_train)

# calculate RMSE

math.sqrt(mean\_squared\_error(X\_test\_scaled,X\_test)), math.sqrt(mean\_squared\_error(test\_data['RF\_PREDICTED'], test\_data['WS\_50\_mean']))

.

*Figure 9 Random Forest Regressor model*



*Figure 10 RFR Model vs. LSTM Model*

**fbprophet :**

# splitting the data and renaming *‘df’* and *‘y’*

dataset=df.reset\_index()[['Date/Time', 'WS\_50\_mean']].rename({'Date/Time':'ds', 'WS\_50\_mean':'y'}, axis='columns')

# split into train and test sets

train\_size = int(len(dataset) \* 0.80)

test\_size = len(dataset) - train\_size

train, test = dataset[0:train\_size], dataset[train\_size:len(dataset)]

# fitting the data and defining seasonalities

m = Prophet(interval\_width=0.95, yearly\_seasonality=True,

seasonality\_mode='additive',

weekly\_seasonality=True,

daily\_seasonality=True)

m.fit(train)

# predicting the values of *‘y’* column

future = m.make\_future\_dataframe(periods=5106, freq='H')

predict=m.predict(future)

fbp = predict['yhat']

m.plot(predict)

# calculate root mean square error

error\_missingData = mean\_squared\_error(dataset['y'], forecast['yhat'])

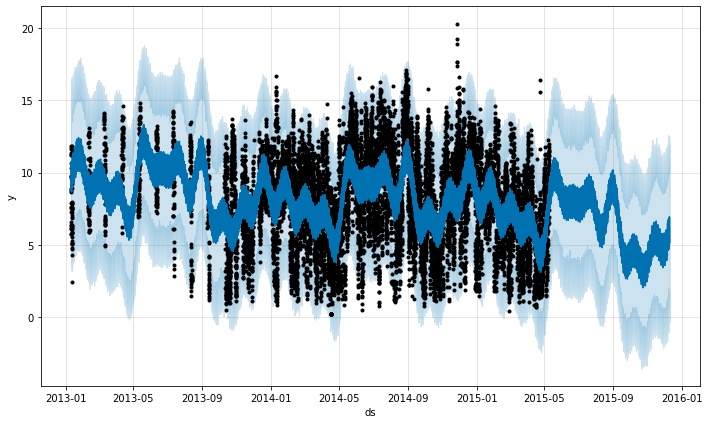
print("RMSE with missing data", error\_missingData)

m.plot\_components(forecast)

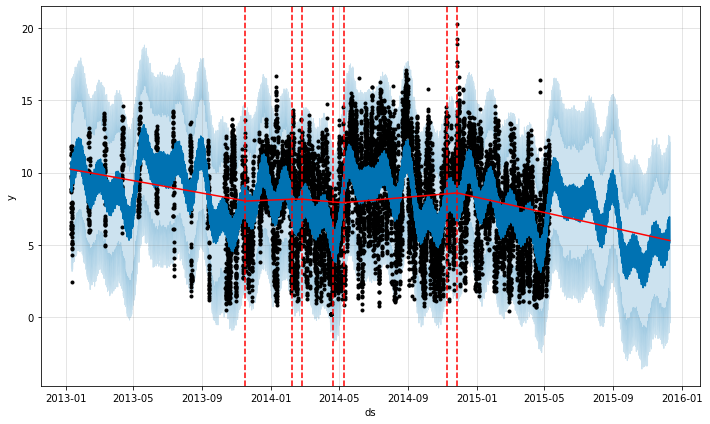
# changepoints of the windspeed

fig = m.plot(forecast)

add\_changepoints\_to\_plot(fig.gca(), m, forecast)



*Figure 11 Fbprophet model*



*Figure 12 Change points*

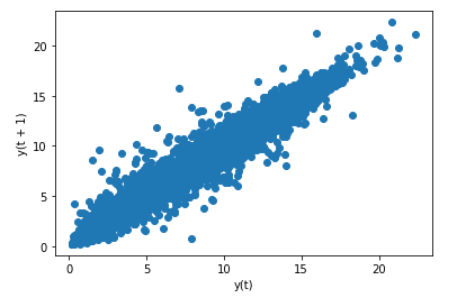
**Decompose :** The decomposition of time series is a statistical task that deconstructs a time series into several components, each representing one of the underlying categories of patterns. Used *statsmodels* pythonlibrary and *seasonal\_decompose*, *autocorrelation\_plot* methods.

lag\_plot(series)

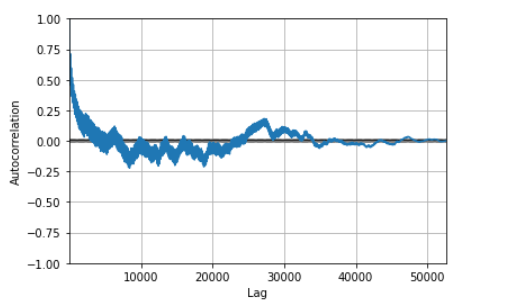
plt.show()

autocorrelation\_plot(series)

plt.show()



*Figure 13 a Decompose - correlation*



*Figure 13 b Decompose – correlation*

**ARIMA :**

# train the standardization

scaler = StandardScaler()

scaler = scaler.fit(values)

print('Mean: %f, StandardDeviation: %f' % (scaler.mean\_, sqrt(scaler.var\_)))

# standardization the dataset and print the first 5 rows

normalized = scaler.transform(values)

# preprocessing

normalized\_X = preprocessing.scale(series)

normalized\_X

# fitting the data for ARIMA

X=values

size = int(len(X) \* 0.66)

train, test = X[0:size], X[size:len(X)]

history = [x for x in train]

predictions = list()

for t in range(len(test)):

model = ARIMA(history, order=(10,1,0))

model\_fit = model.fit(disp=0)

output = model\_fit.forecast()

yhat = output[0]

predictions.append(yhat)

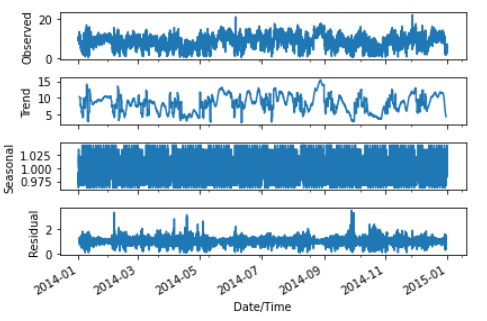
obs = test[t]

history.append(obs)

print('predicted=%f, expected=%f' % (yhat, obs))

# calculate the root mean square error

rmse= math.sqrt(mean\_squared\_error(test, predictions))



*Fig 14 ARIMA model*

# Conclusion

This project we have performed wind data cleaning using python and stored the cleaned dataset into the PostgreSQL database. The stored data is used to perform wind speed prediction. The results of LSTM, RFR, ARIMA and fbprophet approach for modeling and prediction of wind speed variations are compared. The model trained and tested with two years data shown remarkable prediction accuracy. The root mean square error calculated for four prediction model and the results are given below.

* LSTM with RMSE : 0.513
* fbprophet with RMSE : 0.768
* ARIMA with RMSE : 0.816
* RFR with RMSE : 1.895

From the results presented here, it is clear that LSTM is a suitable prediction model for an effective wind speed prediction and thereby helping an efficient, cost effective wind energy management.

# References

1. Wind speed forecast using random forest learning method(2017) - G. V. Drisyaa , Valsaraj P.a , K. Asokanb , K. Satheesh Kumara.
2. Prediction of Wind Speed using Machine Learning(2020) - Nabanita Mandal, Tanuja Sarode, IJCA(0975 – 8887)
3. Wind speed prediction using a univariate ARIMA model and a Multivariate NARX model(2016) - Erasmo Cadenas, Wilfrido Rivera, DOI: 10.3390/en9020109