Wind direction prediction for Windmill data

Computer Science and Engineering

By

Utkarsh Bakshi 181284

Under the supervision of

Renu Sri Gandham

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

DECLARATION

We on our behalf declare that the project that we made has been made by us under the direction of **(**Renu Sri Gandham**)** the Infosys Limited. We also announce that neither this project nor anypart of this project has been submitted elsewhere for any purpose.

Supervised to:
Renu Sri Gandham
Senior Lead Analyst | Data Scientist | Educator at
Infosys Limited

Submitted by:
Utkarsh Bakshi 181284
Computer Science & Engineering Department
Jaypee University of Information Technology

CERTIFICATE

This is to certify that the project which is being represented in the project report named **“**Wind Direction Prediction of windmill data” in partial fufilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineeringand represented to the Department of Computer Science and Engineerin(CSE), Jaypee University of Information Technology, Waknaghatis an authentic record of work done by Utkarsh Bakshi (181284) during the period from January 2022 to May 2022 under the superviision of Renu Sri Gandham, Senior Lead Analyst | Data Scientist | Educator at Infosys Limited.

Utkarsh Bakshi 181284

The above statement made is correct to the best of my knowledge.

Renu Sri Gandham

Senior Lead Analyst | Data Scientist | Educator at Infosys Limited

ACKNOWLEDGEMENT

Firstly, I am gratefulness to Almighty God and thankful for his divine grace that made it possible to successfully complete the project work in a smooth way.

I am grateful and wish my profound indebtedness to supervisor Renu Sri Gandham, Senior Lead Analyst | Data Scientist | Educator at Infosys Limited . where she was very helpful and guided us throughout the thesis to carry out this project. Her endless scholarly guidance, patience, continual encouragment, const scholarly guidance and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude and thankful to to Renu Sri Gandham**,** Department of CSE, for her considerate nature and kind help to finishmy project on time.

I would also generously thank each of the individuals who have helped me continuously and straight forwardly or in a roundabout way to make this project a success. In this odd and strange situation, I also want to thank the various staff individuals, both educating and non-educating, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Utkarsh Bakshi 181284

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

As a renewable energy source, wind turbine generators are considered to be important generation alternatives in electric power systems because of their non-exhaustible nature. As wind power penetration increases, power forecasting is crucially important for integrating wind power in a conventional power grid. This machine learning model will help in predicting the current alignment error of the wind turbine (Direction_Wind) based on the current and historic operational characteristics for a given turbine and nacelle, to maximize the power production under different weather conditions.

CHAPTER 01

INTRODUCTION

Introduction
Problem Statement
Objective
Methodology
Organization

1.1 INTRODUCTION

Wind direction changes over time and space, which has an impact on wind turbines. Many publications have been published that look at the effect of wind direction on the efficiency or power generation of a single wind turbine or multiple wind turbines in a wind farm. However, there has been minimal research into the influence of wind flow direction fluctuation on the loads on wind turbines in a wind farm.

The structural, engine, and control system dynamics all contribute to a wind turbine's total dynamic performance. The structural dynamics that deal with the motion of the blades and tower have an impact on the transient stresses on the wind turbine structure, and hence its fatigue life. The drivetrain dynamics affect the transient loads on drivetrain components such as the gearbox, as well as the control system for a pitch-regulated machine.

The generation of wind power is determined by wind speed and its derivatives, such as wind speed and direction. This study addresses three primary difficulties, taking into account the stochastic nature of wind power: It first goes through the current state of energy forecasting, with a focus on wind power forecasting. It gives an overview of the numerous elements that influence wind power generation and describes the major features of the forecasting model design framework.

This machine learning model will help in predicting the current alignment error of the wind turbine (Direction_Wind) based on the current and historic operational characteristics for a given turbine and nacelle, to maximize the power production under different weather conditions.

1.2 PROBLEM STATEMENT

Build a machine learning model which would help predicting the current alignment error of the wind turbine (Column Direction_Wind) based on the current and historic operational characteristics for a given turbine and nacelle, to maximize the power production under different weather conditions

1.3 OBJECTIVE

An energy domain company has decided to automate the process of adjusting the wind turbine angle/direction for power optimization. Sensor has collected huge amount of real time data and is available in **"sensor_wind_data.csv"**. The company has given the data and has advised the team to build a model which can predict the wind direction.

1.4 METHODOLOGY

- ➢ Linear Regression
- ➢ Gradient Boosting
- ➢ Grid Search

- **Linear Regression**
  - ★ Linear Regression is a machine learning algorithm based on supervised learning.
  - ★ Regression models a target prediction value based on independent variables.
  - ★ It is mostly used for finding out the relationship between variables and forecasting.
  - ★ In this project with the help of highly correlated columns the target column 'Direction_wind' is predicted.

- **Gradient Boosting Algorithm**
  - ➢ In gradient boosting, each predictor corrects its predecessor's error.
  - ➢ In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.
  - ➢ The class of the gradient boosting regression in scikit-learn is GradientBoostingRegressor.
  - ➢ Gradient boosting works including the loss function, weak learners and the additive model.
  - ➢ Gradient boosting plays an important role in improving the performance over the base algorithm with various regularization schemes.

- **Grid search**
  - ★ Grid searching is a method to find the best possible combination of hyper-parameters at which the model achieves the highest accuracy.
  - ★ Before applying Grid Searching on any algorithm, Data is divided into training and validation sets, a validation set is used to validate the models.
  - ★ Grid Searching plays a vital role in tuning hyperparameters for the mathematically complex models.
  - ★ In this project the accuracy has been improved with the help of grid search.

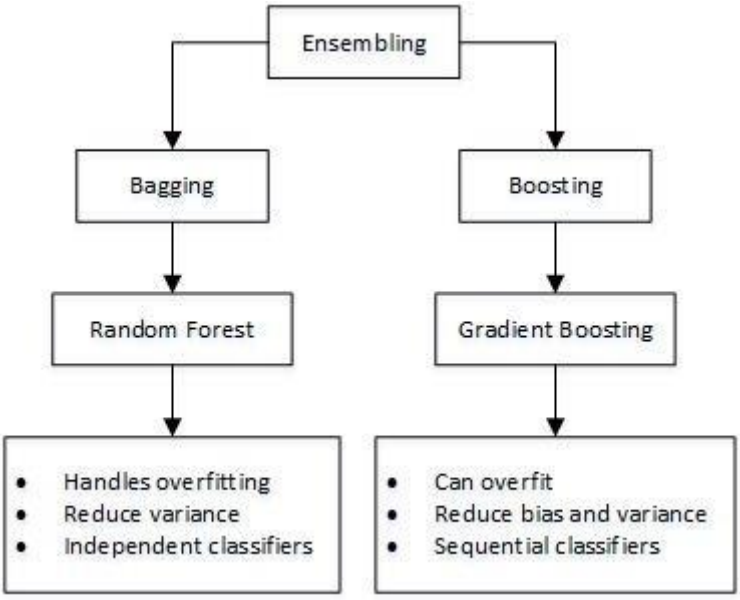


Fig 1.1 Random Forest v/s Gradient Boost

1.5 ORGANIZATION

Chapter 1 contains the Introduction, Problem Statement, Objective, Methodology of the Project or System.

Chapter 2 contains the literature survey in which some of the previous works are studied and compared in order to make this model.

Chapter 3 discusses the system development as it contains all the computational, mathematical, analytical, theoretical data.

Chapter 4 is all about the performance analysis in which different models' results are compared, with algorithm improvement and outputs.

Chapter 5 discusses the conclusions, future scope of the model, its applications in real world, advantages and limitations.

Then we have the References section.

Last is the Appendices section which includes the source code of model.

CHAPTER 02

LITERATURE SURVEY

As per the information given in Chapter 1, the movement tracking of objects can have a lot of applications in real world. In this section, some of the previous works are discussed which were studied and compared in order to make this model.

There are many different ways to find out optical flow of objects out of which Gradient method is basic one. But due to its aperture problem, gradient method cannot give the complete optical flow fields solution. After survying , we got to know two different Differential techniques named Lucas - Kanade algorithm and Horn - Schunck algorithm and compared them based on their results.

Bhumika Gupta (2017) [1] the acquisition of the proposed object by a well-known computer technology associated with computer vision and image processing focused to find its objects or conditions for a particular category (e.g. people, flowers, animals) in digital photos and videos. There are various applications for acquisition of assets well researched which includes facial recognition, character recognition, and vehicle calculator. Object acquisition can be used for a variety of purposes including retrieval and surveillance. Different basic concepts were used in this study object discovery while using the OpenCV library of python 2.7, improves the efficiency and accuracy of the object discovery introduced

Kartik Umesh Sharma (2017) [2], proposed something the discovery system detects real-world objects be it in digital photography or video, where the object can for any category of objects namely people, cars, etc. to get something in a photo or video in the system it needs to have a few parts to complete visual function, they are a model database, a feature detector, hypothesiser and hypothesiser confirmation. This paper provides an overview of the variety techniques used to find an object, localize an object, separate object, remove features, appearance information, and more, in photos and videos. I ideas are drawn based on textbooks as well important issues are also identified as appropriate to the object adoption. Information about source and online codes data sets are provided to help the new researcher enter a place to find something. An idea about a possible solution of the discovery of a multi-class object object was also introduced. This the paper is suitable for beginner researchers on this domain.

Mukesh Tiwari (2017) [3] introduced object discovery again tracking is one of the most important areas of research for a common change in object movement and scene variability size, occlusions, appearance differences, and ego-motion as well light changes. In particular, choosing a feature is an important role in tracking an object. It has to do with a lot of real time applications such as car view, video surveillance and and so on. To win the issue of adoption, track related to the movement and appearance of an object. Most of this The algorithm focuses on the tracking algorithm to smooth it out video sequence. On the other hand, only a limited number of methods are used pre-existing information about object shape, color, texture and more. Tracking algorithm integrates The parameters

mentioned above are also discussed analyzed in this study. The purpose of this paper is to analyze and update the previous way of looking at an object tracking and identification using video sequencing different categories. Also, point to a gap and suggest a new one how to improve video object tracking frame.

Rupesh Kumar Rout
[4] developed feedback based object detection algorithm. It adopts dual layer updating model to update the background and segment the foreground with an adaptive threshold method and object tracking is treated as a object matching algorithm.

CHAPTER 03

SYSTEM DEVELOPMENT

Methods used
Theoretical and Mathematical development
Computational development

Methods / Tools used

• We saw how data science requires a vast array of tools. The tools for data science are for analyzing data, creating aesthetic and interactive visualizations and creating powerful predictive models using machine learning algorithms.

JUPYTER

• Project **Jupyter** is an open-source tool based on IPython for helping developers in making open-source software and experiences interactive computing. Jupyter supports multiple languages like Julia, **Python**, and R.

• It is a web-application tool used for writing live code, visualizations, and presentations. Jupyter is a widely popular tool that is designed to address the requirements of Data Science.

• It is an interactable environment through which Data Scientists can perform all of their responsibilities. It is also a powerful tool for storytelling as various presentation features are present in it.

• Using Jupyter Notebooks, one can perform data cleaning, statistical computation, visualization and create predictive **machine learning models**. It is 100% open-source and is, therefore, free of cost.

• There is an online Jupyter environment called Collaboratory which runs on the cloud and stores the data in Google Drive.

LIBRARIES USED

• A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer.Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc. Python libraries that used in Machine Learning are:

- **Numpy**
- **Pandas**
- **Matplotlib**
- **Seaborn**
- **Scikit-learn**

NUMPY

• NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow use NumPy internally for manipulation of Tensors.

PANDAS

• Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and a wide variety of tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

MATPLOTLIB

• Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc.

SEABORN

• Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

• Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for the same variables for better understanding of the dataset.

SCIKIT-LEARN

•Scikit-learn is one of the most popular Machine Learning libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool for those starting out with Machine Learning.

Theoretical and Mathematical development

Windmill_notebook1.ipynb

Exploratory Data Analysis:

- Importing the libraries.
- Reading the data file.
- Displaying the first five rows.
- Describing the data set.
- Checking for the null values.
- Performing Correlation.
- After performing the columns that are greater than 0.9 have been dropped.
- After performing there only 24 columns remain out of 41 columns that are present in the data set.
- To visualize how columns are correlated to each other we plotted the graph in the form of a heat map.
- To see how input variables are contributing to outcome variables(i.e "Wind_Direction") we have plotted the graph.
- To observe if the remaining 24 columns correlate or not we have plotted the pair plot and observed that they are not correlated and the data points are randomly distributed.
- We have plotted the box plot and observed that many values are in oulier range we cannot remove them because they are lying very closely but Angle_Blade1 contains outlier so we need to remove it.

Building the linear regression model:

- We have split the data set and trained the model.
- After finding the score of the model the accuracy is 18 percent which is not applicable because the accuracy should be above 70% .

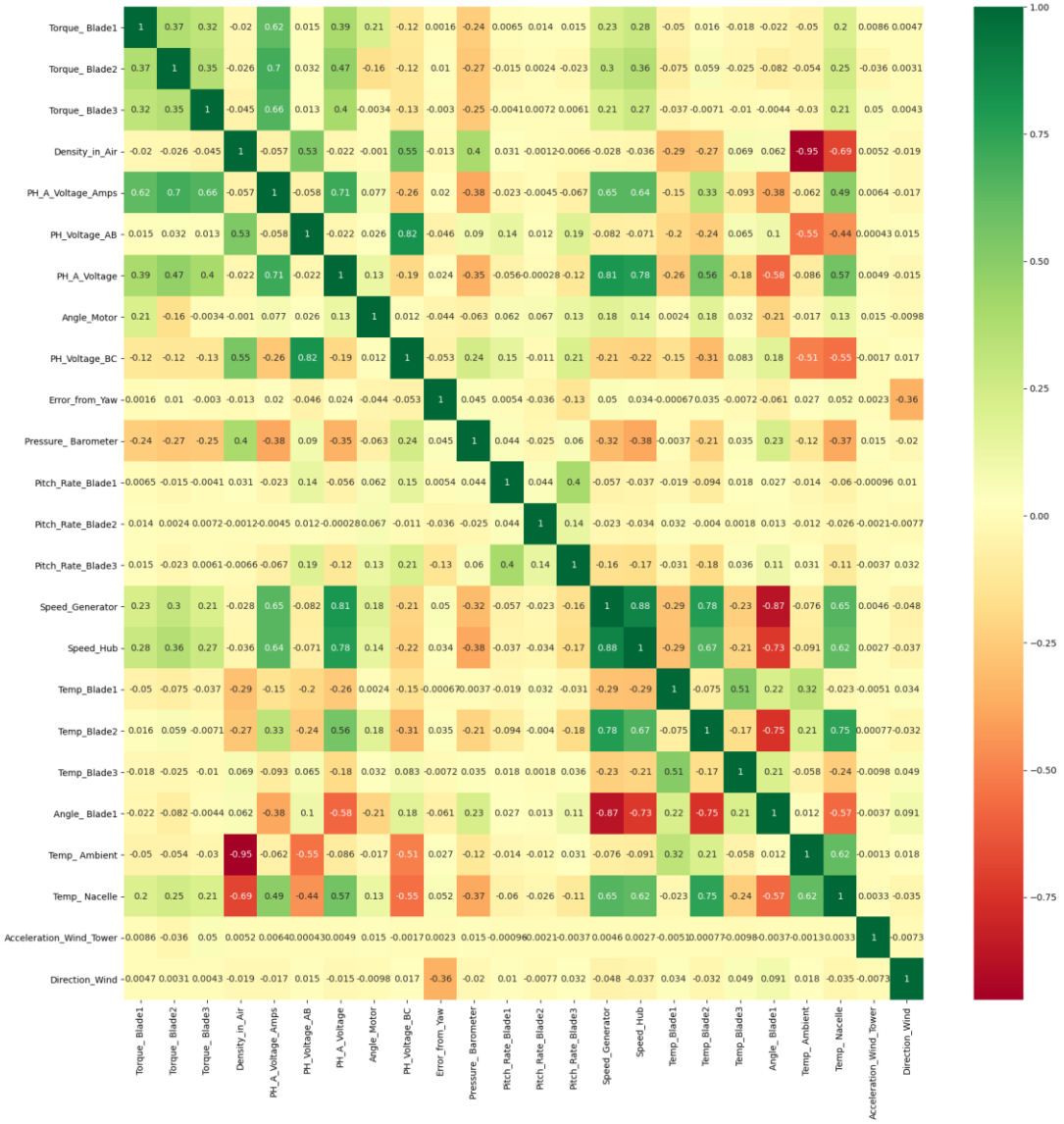


Fig 3.1 2-D Heat Map

- **Building a linear regression model:**

Windmill_notebook2.ipynb

❖ After not getting the desired accuracy we performed different methods
  - ➢ Took the log of different columns -Speed_Hub,Speed_Wind ,we plotted a graph to see how both variables are related then we took a log to see the difference .
  - ➢ Performed task of derived variable -took square,cube on different columns
  - ➢ Performed differencing task ,took difference of speed_hub column and speed_wind column

➢ All the above task was performed on selected columns - Power_PH_C,Pitch_Rate_Blade1,Pitch_Rate_Blade2,Pitch_Rate_Blade3,Speed_wind,Speed_Hub,Angle_Blade1,Angle_Blade2,Angle_Blade3.

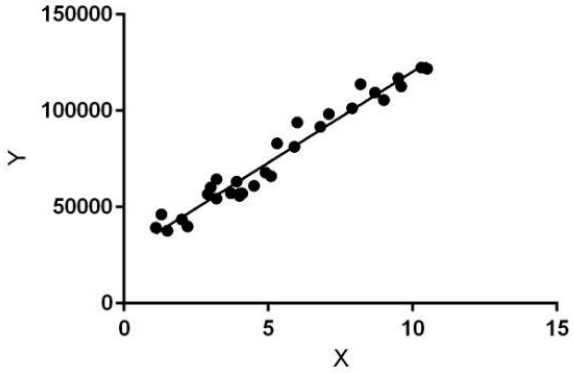
➢ After performing all the above we got a maximum accuracy of 20%.

➢ We changed columns once again. Now we took the following columns- Error_from_Yaw,Speed_wind,Speed_Hub,Angle_Blade1,Angle_Blade2,Angle_Blade3.

➢ Still we didn't get good accuracy ,so finally we came to a conclusion that the LINEAR REGRESSION model is not suited for our prediction ,so now we performed Gradient Boosting Regression.

Linear Regression Algorithm

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task**.** Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

Fig 3.2 Regression Model

Computational Development

Code Breakage

- We will import all the required libraries for the implementation of the project.

Reading the datafile

```
In [3]: wind_data = pd.read_csv("sensor_wind_data.csv")
```

Dataset

```
In [4]: wind_data.head()
```

Out[4]:

	Torque_ Blade1	Torque_ Blade2	Torque_ Blade3	Density_in_Air	PH_A_Voltage_Amps	PH_Voltage_AB	Amps_Voltage_Motor	PH_Voltage_AN	PH_A_Voltage	Angle_Motor	...	T
0	0.488559	2.982536	0.872205	1.661788	459.047948	0.755399	1383.780584	0.434857	10.554844	179.283151	...	
1	2.086069	3.022398	1.376642	1.663072	590.868528	0.755706	1775.154500	0.434727	7.620256	171.540307	...	
2	4.481245	5.839726	1.021304	1.663500	607.994237	0.756013	1826.079321	0.434597	8.701336	204.708207	...	
3	-0.959085	2.053630	4.795327	1.661740	506.361961	0.756320	1523.516573	0.434510	7.057755	172.020813	...	
4	3.922131	0.068994	2.731100	1.660629	548.720577	0.756627	1649.772005	0.434718	9.987469	205.016184	...	

5 rows × 41 columns

Fig 3.3 Dataset

- Performing Eda

Performing EDA: Checking for null values

```
In [7]: wind_data.isna().any().sum()
Out[7]: 0
```

Correlation Matrix

```
In [8]: corr= wind_data.corr()

        import numpy as np
        corr1 = np.round(corr, decimals=2)

In [9]: corr1.to_csv("file1.csv")

In [10]: from sklearn.model_selection import train_test_split
         from sklearn.svm import SVC
         from sklearn.metrics import confusion_matrix
         np.random.seed(123)
```

Here we have dropped all the columns having correlation greater than 0.9

Fig 3.4 EDA

- To take the input point which is one of the main part of the project we use this logic in order to get things done.

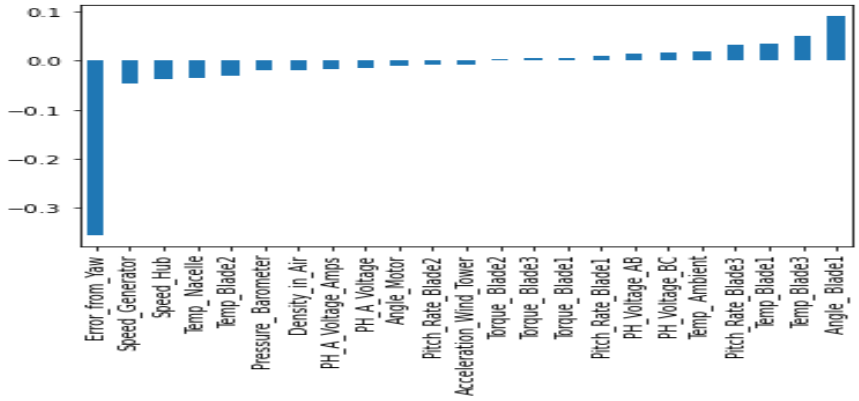


Fig 3.5 Plot

Correlation Matrix

```
[20]: corr= wind_data.corr()

      import numpy as np
      corr1 = np.round(corr, decimals=2)
```

```
[21]: corr1.to_csv("file1.csv")
```

```
[22]: from sklearn.model_selection import train_test_split
      from sklearn.svm import SVC
      from sklearn.metrics import confusion_matrix
      np.random.seed(123)
```

Here we have dropped all the columns having correlation greater than 0.9

```
[23]: corr = wind_data.corr()
      columns = np.full((corr.shape[0],), True, dtype=bool)
      for i in range(corr.shape[0]):
          for j in range(i+1, corr.shape[0]):
              if corr.iloc[i,j] >= 0.9:
                  if columns[j]:
                      columns[j] = False
      selected_columns = wind_data.columns[columns]
      wind_data = wind_data[selected_columns]
```

Correlation Matrix

Fig 3.6 Correlation Matrix

- Plotting the chart to visualize how columns are correlated with each other in the form of Heat Map

```
: %matplotlib notebook
  from itertools import combinations
  import matplotlib.pyplot as plt

  fig, axes = plt.subplots(len(wind_data.columns)//3, 3, figsize=(12, 48))

  i = 0
  for triaxis in axes:
      for axis in triaxis:
          wind_data.hist(column = wind_data.columns[i], bins = 100, ax=axis)
          i = i+1
```

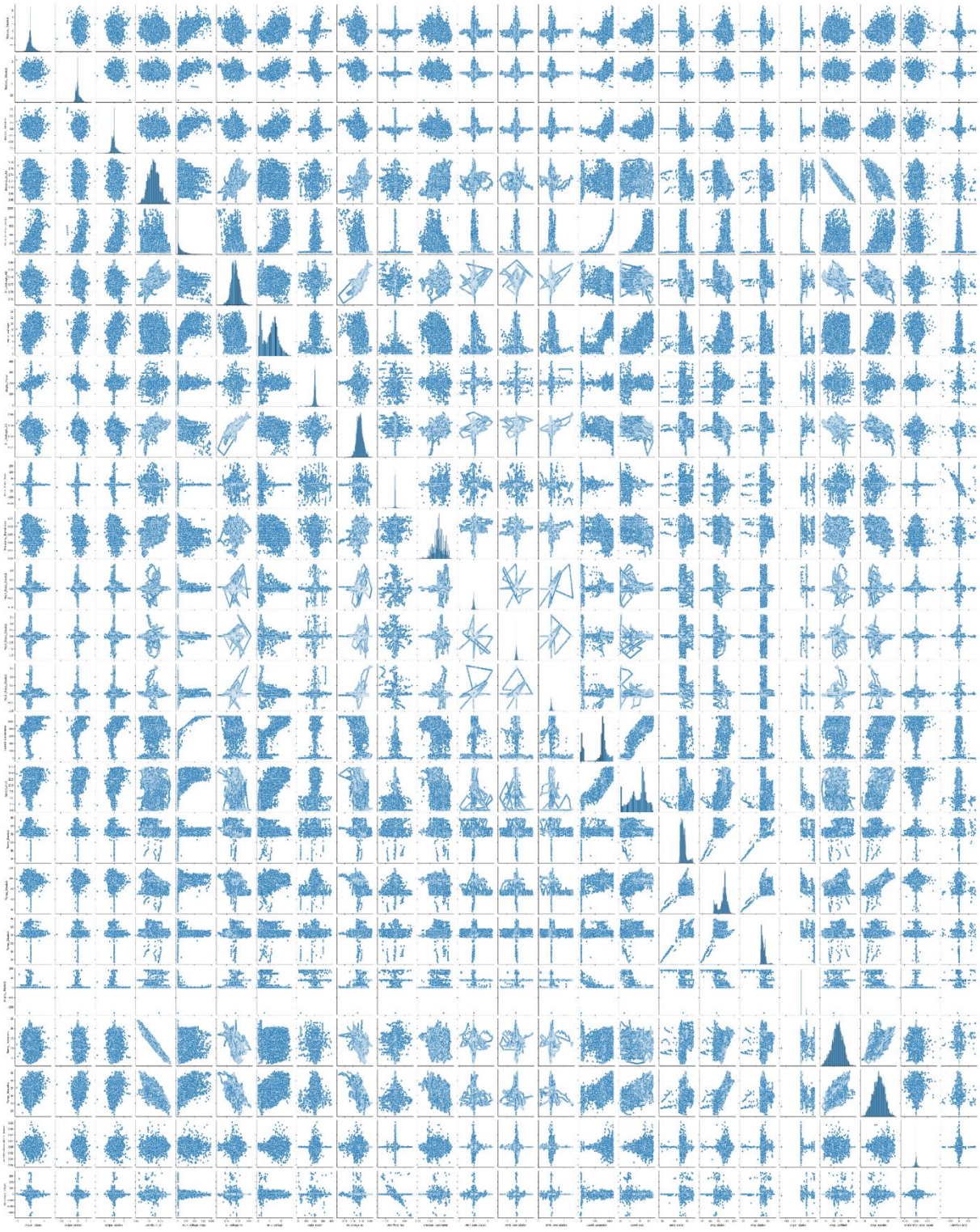


Fig 3.7 Sub PLots

- From the above pair plot we observe that none of the columns are correlated and data points are randomly distibuted so we have successfully dropped the correlated columns.
- From the boxplot many values are in outlier range but we cannot remove them because they are lying very closely ,Angle_Blade1 contains outlier so we need to remove it.

Splitting the dataset and Training the Linear model:

Splitting the dataset

```
X= wind_data.drop(['Direction_Wind'],axis=1)
Y=wind_data['Direction_Wind']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=42)
```

Training the model

```
from sklearn import  linear_model
```

```
model = linear_model.LinearRegression()
```

```
model.fit(X_train,Y_train)
```

```
LinearRegression()
```

Score of the model

```
score = model.score(X_test,Y_test)
score
```

```
0.188988190913282
```

We have split the data set and trained the model.
After finding the score of the model the accuracy is 18 percent which is not applicable because the accuracy should be above 70% .

General Additive Model

- A Generalized additive model (GAM) is a generalized linear model in which the linear response variable depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions.
- Created derived variables
- Tried different combination by squaring and cubing of variables.

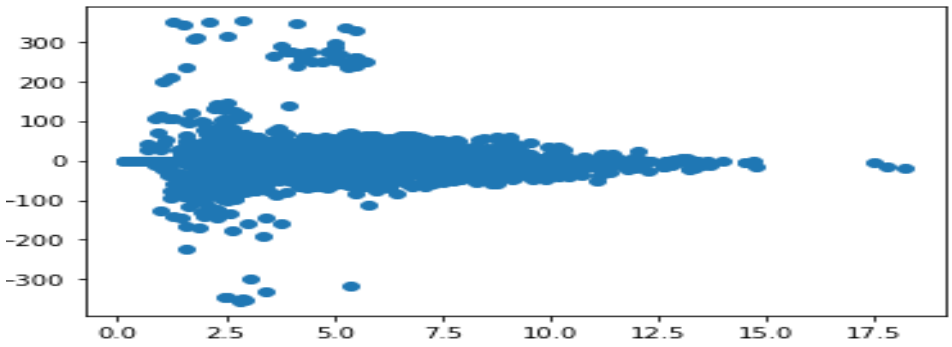


Fig 3.8 Scatter Plot

Taking particular columns

Taking particular columns namely - Power_PH_C","Pitch_Rate_Blade1","Pitch_Rate_Blade2","Pitch_Rate_Blade3","Speed_Hub","Speed_Wind","Angle_ Blade1","Angle_ Blade2","Angle_ Blade3

```
X = wind_data[["Power_PH_C","Pitch_Rate_Blade1","Pitch_Rate_Blade2","Pitch_Rate_Blade3","Speed_Hub","Speed_Wind","Angle_ Blade1",
Y=wind_data["Direction_Wind"]
```

Pitch_Rate_Blade1,Pitch_Rate_Blade2,Pitch_Rate_Blade3 contains negative value more than 5000 so cannot drop the columns,replace it with different ways like- mean,median,square of values.

	Power_PH_C	Pitch_Rate_Blade1	Pitch_Rate_Blade2	Pitch_Rate_Blade3	Speed_Hub	Speed_Wind	Angle_ Blade1	Angle_ Blade2	Angle_ Blade3
0	500.733513	0.011941	0.089392	0.0	15.833939	8.903327	1.083084	1.092598	1.117342
1	643.476705	0.011537	0.092245	0.0	15.834342	9.870556	1.400603	1.390493	1.409885
2	661.680912	0.011147	0.095097	0.0	15.834746	10.715178	1.734305	1.736561	1.764734
3	552.182798	0.011412	0.097926	0.0	15.835150	9.167437	1.236629	1.241662	1.230212

Model Building

Linear Regression Model

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3)
```

```
model = linear_model.LinearRegression()
model.fit(X_train,Y_train)
```

```
LinearRegression()
```

```
score = model.score(X_train,Y_train)
score
```

```
0.03286909027231455
```

Tranfroming Variales

Taking log of different variables

```
test1=pd.read_csv("sensor_wind_data.csv")
test1.head()
```

	Torque_ Blade1	Torque_ Blade2	Torque_ Blade3	Density_in_Air	PH_A_Voltage_Amps	PH_Voltage_AB	Amps_Voltage_Motor	PH_Voltage_AN	PH_A_Voltage
0	0.488559	2.982536	0.872205	1.661788	459.047948	0.755399	1383.780584	0.434857	10.554844
1	2.086069	3.022398	1.376642	1.663072	590.868528	0.755706	1775.154500	0.434727	7.620256
2	4.481245	5.839726	1.021304	1.663500	607.994237	0.756013	1826.079321	0.434597	8.701336

- Performed transforming variables for better results.

- Data set splitting.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

Gradient Boosting Algorithm

Windmill_notebook3.ipynb

- **GBA**
  - ➢ In gradient boosting, each predictor corrects its predecessor's error.
  - ➢ In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.
  - ➢ In this project first search for missing values if you find any impute with the median values.
  - ➢ First necessary dependent variables and target variables are assigned.
  - ➢ After that data splitting is done.
  - ➢ Now Testing data is 30% and Training data is 70%.
  - ➢ Next gradient boosting parameters are assigned using grid search
  - ➢ Then the model is trained.
  - ➢ Accuracy was found out to be 91% and 96% in the training and testing dataset respectively.
  - ➢ The accuracy has been greatly increased from 62% to 91% by using gradient boosting.
  - ➢ Finally mean square error(MSE) is calculated and was found to be 27.47 which is optimum.

Model Building

```
x = df[["Error_from_Yaw", 'Speed_Hub','Speed_Wind','Angle_ Blade1','Angle_ Blade2','Angle_ Blade3']]
```

```
y=df["Direction_Wind"]
```

Splitting data into 70-30 for traning and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

```
sc = StandardScaler()
x_train_std = sc.fit_transform(x_train)
x_test_std = sc.transform(x_test)
```

- Parameters for Gradient Boosting Algorithm

```
gbr_params = {'n_estimators': 1000,
              'max_depth': 8,
              'min_samples_split': 5,
              'learning_rate': 0.01,
              'loss': 'ls'}
```

- Model Accuracy:

```
print("Model Accuracy: %.3f" % gbr.score(x_train_std, y_train))
```

```
Model Accuracy: 0.933
```

```
print("Model Accuracy: %.3f" % gbr.score(x_test_std, y_test))
```

```
Model Accuracy: 0.964
```

- Results from Grid Search

The best estimator across ALL searched params:
GradientBoostingRegressor(learning_rate=0.01, max_depth=10, n_estimators=800,
subsample=0.5)

The best score across ALL searched params:
0.38386950597864566

The best parameters across ALL searched params:
{'learning_rate': 0.01, 'max_depth': 10, 'n_estimators': 800, 'subsample': 0.5}

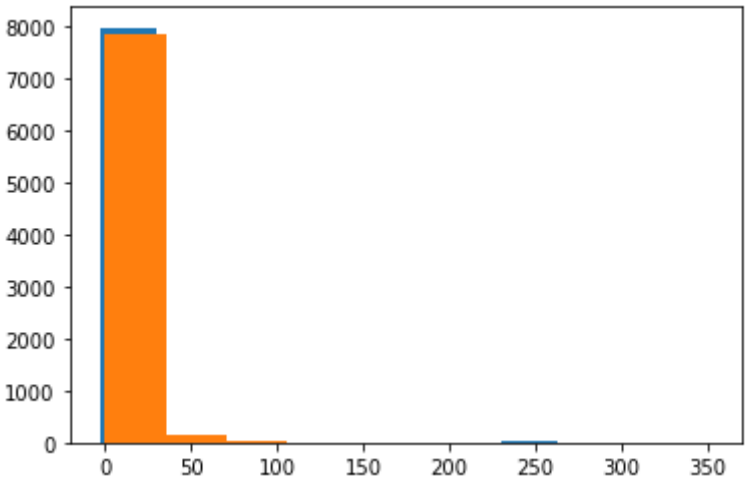


Fig 3.9 Prediction for the test data

Predictions for the test data

	Actual	Predicted
1981	3.144520	4.826248
11030	3.713587	4.809721
3996	3.713587	4.917551
2179	3.713587	4.161358
586	3.713587	6.988173
...	...	...
2195	3.713587	4.809540
11337	3.713587	6.941195
10005	3.713587	12.570249
2918	3.713587	7.095098
146	13.772180	7.001308

- After doing all preprocessing part on data we build Linear Regression and General Additive model but we didn't get the accuracy so we finally concluded that linear model is not suited for our dataset after that we performed Gradient Boosting Algorithm and found best parameters using Grid search and we finally got an accuracy of 93.3%.

We used following tools and technologies to make this model:

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

- Processor type : Pentium III-compatible processor or faster.
- Processor speed : 1.4 GHz or more 32 bit/64 bit
- RAM : 4GB or more
- HARD DISK : 16GB or more
- Monitor : VGA or higher resolution 800x600 or higher resolution
- Pointing device : Microsoft Mouse or compatible pointing device

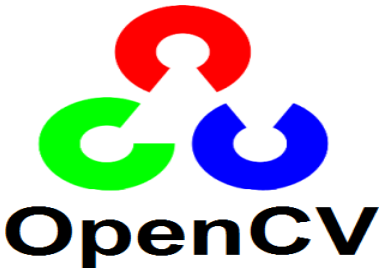
SOFTWARE REQUIREMENTS

- Application software Framework : python
- Operating System : Windows 7 /10

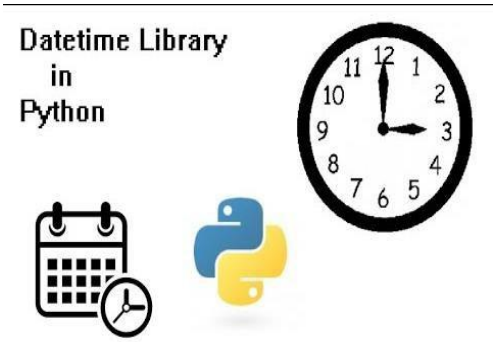
TOOLS USED

- Python programming language
- OpenCV in Python
- NumPy in Python
- Jupyter Notebook


python


OpenCV


NumPy


Datetime Library
in
Python


jupyter

Python:

Python is a robust programming language with a wide range of capabilities.
Its broad features make working with targeted programs (including meta-programming and meta-objects) simple, and it fully supports object-oriented programming. Many additional paradigms, such as contract generation and logic programming, are supposed through extensions. Python takes advantage of power typing as well as the integration of reference computation and waste management waste collecting. lt also supports advanced word processing (late binding), which binds the way the words change during the process.

Patches to less essential sections of C Python that can give a minor improvement in performance at an obvious price are rejected by Python developers who try to prevent premature execution. When speed is crucial, the Python program developer can use mod-written modules in C-languages or PyPy, a timely compiler, to submit time-sensitive jobs. Python is a Python interpreter that converts Python scripts to C and invokes the Python interpreter directly from the C-1evel API. Python developers strive to make the language enjoyable to use. Python's architecture supports Lisp culture in terms of functionality. Filters, maps, and job reduction, as well as a list comprehension, dictionaries, sets, and generator expressions, are all included.

Two modules (itertools and functools) in the standard library use realistic Haskell and Standard ML tools.

Why Python?

We used the python language since it is the only language which can be used on the maximum platforms that we can work on like Win, Linux, Mac, etc. Itis a free language having all the libraries for free . It is easy to use and is just like the common language that we use to communicate.

Python supports library collections and has a straightforward language structure like English while java and C ++ have complex codes. Python programs have fewer lines than other programming languages. That is why we use Python language to know human-made, AI, and deal with a large amount of knowledge. Python is the default language for an article.

Libraries Used:

• A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer.Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc. Python libraries that used in Machine Learning are:

- **Numpy**
- **Pandas**
- **Matplotlib**
- **Seaborn**
- **Scikit-learn**

To import the required libraries, we'll first use the keyword import and then library name.

Importing Libraries

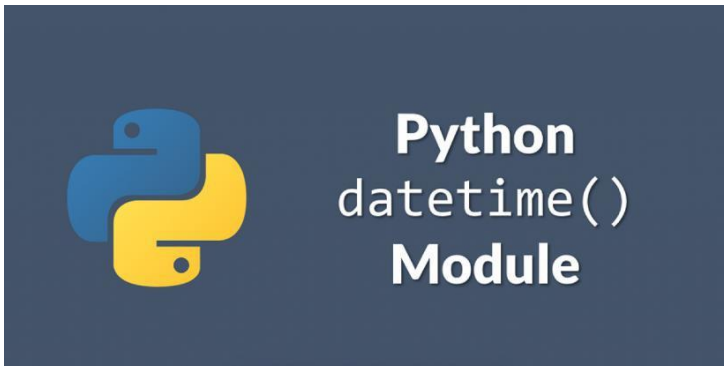
```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import tree
from sklearn.model_selection import GridSearchCV
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

NumPy:

Numpy is a standard purpose processing package. It provides high performance for multidimensional object, and tools for working with these components. It is a basic computer science package via Python. NumPy, which represents Numerical Python, is a multidisciplinary library and a group of methods for filtering those members. Using NumPy, maths and logical operations in arrays can be determined. This study describes the basics of NumPy as its structure and surrounding. It also discusses the various functions of similar members, types of references, etc. An introduction to Matplotlib was also provided. All of this is explained with the help of examples of better understanding.

In addition to its obvious scientific uses, Numpy could be used as an efficient multi-sided container of standard data.

DateTime:



At Python, date & time is not the data itself, but the module that is called date may be imported to work with date and time. The Python Datetime module will be built into the Python, so there is no need to install it outside. The Python Datetime module provides classes to work with the date and time. These classes offer several activities to deal with days, times and intervals. Date and time of day are in Python, so if you fool yourself, you are actually changing things and not the character unit or time stamps.

Jupyter Notebook:

One of the Project Jupiter projects is that the Jupiter notebook. Jupiter notebook is an open source web application that equips users with coding tools to urge various results like data clearing, data visibility, editing, retransmission, mathematical modeling, machine learning, etc. The Jupyter notebook originally contained only three main languages, Julia, Python, and R which, when combined, gave birth to the name Jupyter. Currently, the jupyter notebook supports over 40 editing languages. These scripts are often easily shared via github or sent to a colleague via dropbox.

CHAPTER 04

PERFORMANCE ANALYSIS

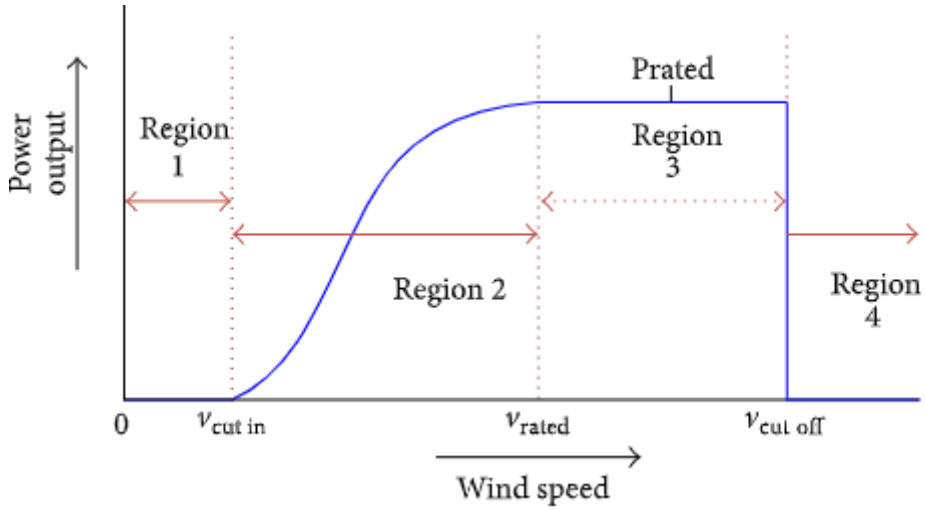
Algorithm improvements:

Better accuracy: Gradient Boosting Regression generally provides better accuracy. When we compare the accuracy of GBR with other regression techniques like Linear Regression, GBR is mostly winner all the time. This is why GBR is being used in most of the online hackathon and competitions.

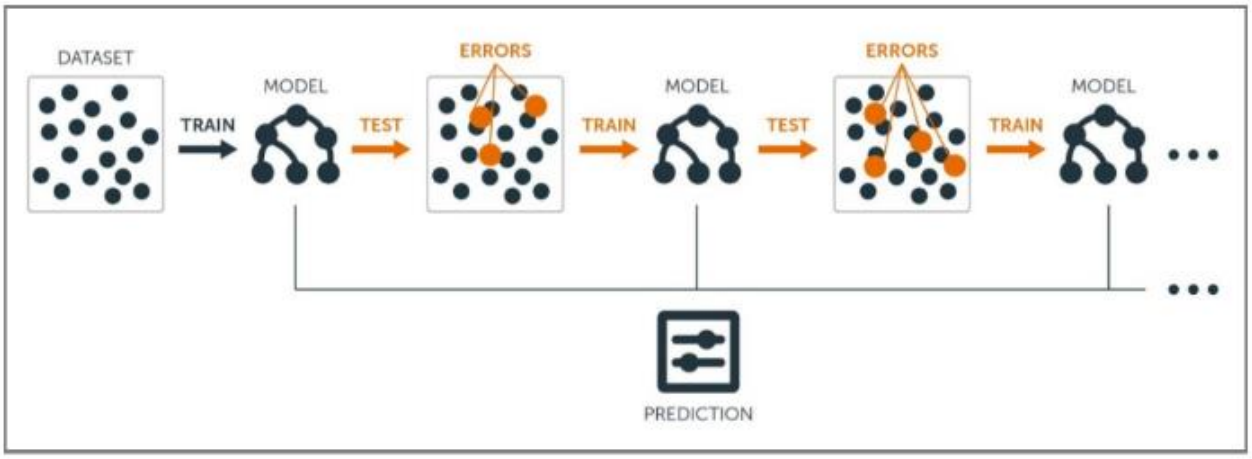
Less pre-processing: As we know that data pre processing is one of the vital steps in machine learning workflow, and if we do not do it properly then it affects our model accuracy. However, Gradient Boosting Regression requires minimal data preprocessing, which helps us in implementing this model faster with lesser complexity. Though pre-processing is not mandatory here we should note that we can improve model performance by spending time in pre-processing the data.

Higher flexibility: Gradient Boosting Regression provides can be used with many hyper-parameter and loss functions. This makes the model highly flexible and it can be used to solve a wide variety of problems.

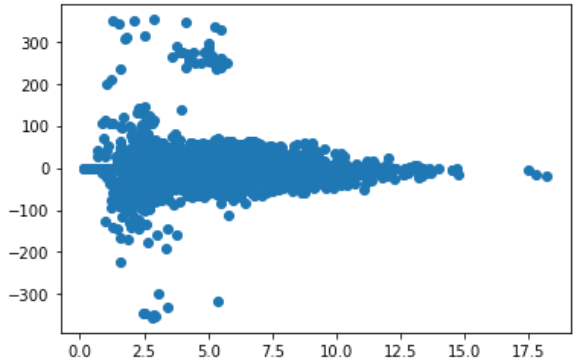
Missing data: Missing data is one of the issue while training a model. Gradient Boosting Regression handles the missing data on its own and does not require us to handle it explicitly. This is clearly a great win over other similar algorithms. In this algorithm the missing values are treated as containing information. Thus during tree building, splitting decisions for node are decided by minimizing the loss function and treating missing values as a separate category that can go either left or right.

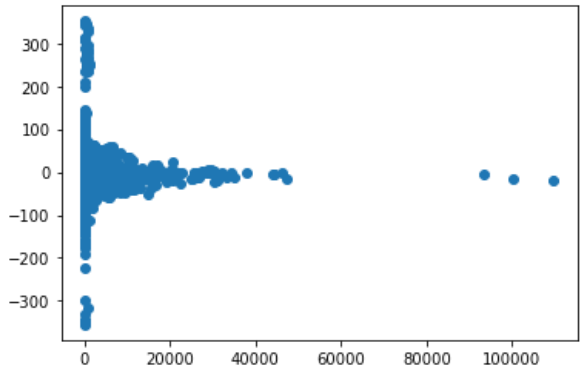


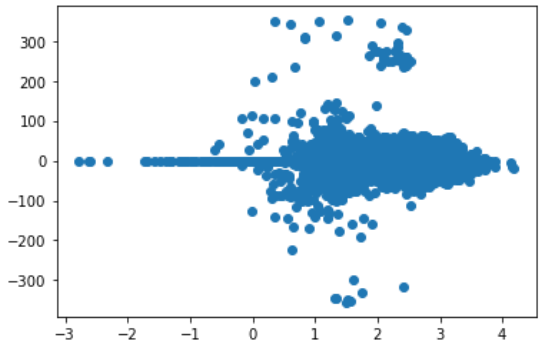
How does GBA work:

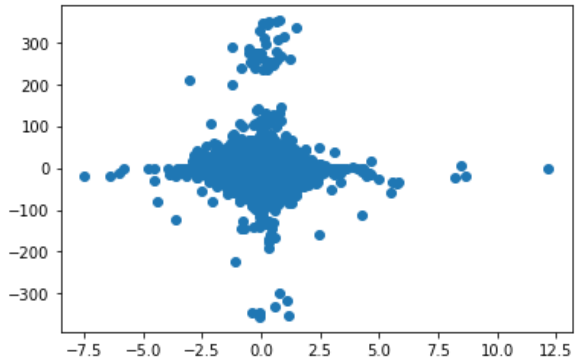


Scatter plots:









CHAPTER 05

CONCLUSIONS

Conclusion
Applications
Future Scope
Advantages
Limitations

Conclusion

After doing all preprocessing part on data we build Linear Regression and General Additive model but we didn't get the accuracy so we finally concluded that linear model is not suited for our dataset after that we performed Gradient Boosting Algorithm and found best parameters using Grid search and we finally got an accuracy of 93.3%.

Applications

This project can have numerous applications in real world.

Wind power is playing a pivotal part in global energy growth as it is clean and pollution-free. To maximize profits, economic scheduling, dispatching, and planning the unit commitment, there is a great demand for wind forecasting techniques. This drives the researchers and electric utility planners in the direction of more advanced approaches to forecast over broader time horizons. Key prediction techniques use physical, statistical approaches, artificial intelligence techniques, and hybrid methods. An extensive review of the current forecasting techniques, as well as their performance evaluation, is here presented. The techniques used for improving the prediction accuracy, methods to overcome major forecasting problems, evolving trends, and further advanced applications in future research are explored.

FUTURE SCOPE

After doing all preprocessing part on data we build Linear Regression and General Additive model but we didn't get the accuracy so we finally concluded that linear model is not suited for our dataset after that we performed Gradient Boosting Algorithm and found best parameters using Grid search and we finally got an accuracy of 93.3%.
We will be trying advance algorithm like Yaw control and others.

ADVANTAGES

Wind direction **helps in identifying, tracking and predicting weather patterns**, such as determining the direction of movement of depressions and the transfer of precipitation from one region to another.

LIMITATIONS

Wind energy has a drawback that **it is not a constant energy source**. Wind turbines generate noise and visual pollution. Birds have been killed by flying into spinning turbine blades. The cost of travel and maintenance on the turbines increases and is time-consuming.

REFERENCE

[1] Bhumika Gupta (2017) et al
https://www.researchgate.net/profile/PriyankaBadhani/publication/317058859_Study_of_Twitter_Sentiment_Analysis_using_Machine_Learning_Algorithms_on_Python/links/60fbebe50c2bfa282af92131/Study-of-Twitter-Sentiment-Analysis-using-Machine-Learning-Algorithms-on-Python.pdf

[2] Kartik Umesh Sharma (2017) et al
https://www.inderscienceonline.com/doi/abs/10.1504/IJCVR.2017.081234

[3] Mukesh Tiwari (2017) et al
http://www.ripublication.com/ijcir17/ijcirv13n5_07.pdf

[4] Rupesh Kumar Rout (2015) et al
http://ethesis.nitrkl.ac.in/4836/1/211CS1049.pdf

Dealt with bugs and errors with the help of YouTube and Stack Overflow.

Studied about the required libraries/modules from GeeksforGeeks.

APPENDICES

This section includes the source code:

Importing Libraries ¶

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the datafile

```
wind_data = pd.read_csv("sensor_wind_data.csv")
```

Dataset

```
wind_data.head()
```

	Torque_ Blade1	Torque_ Blade2	Torque_ Blade3	Density_in_Air	PH_A_Voltage_Amps	PH_Voltage_AB	Amps_Voltage_Motor	PH_Voltage_AN	PH_A_Voltage	Angle_Motor	...	T
0	0.488559	2.982536	0.872205	1.661788	459.047948	0.755399	1383.780584	0.434857	10.554844	179.283151	...	
1	2.086069	3.022398	1.376642	1.663072	590.868528	0.755706	1775.154500	0.434727	7.620256	171.540307	...	
2	4.481245	5.839726	1.021304	1.663500	607.994237	0.756013	1826.079321	0.434597	8.701336	204.708207		

Performing EDA: Checking for null values

In [7]:
```
wind_data.isna().any().sum()
```

Out[7]: 0

Correlation Matrix

In [8]:
```
corr= wind_data.corr()

import numpy as np
corr1 = np.round(corr, decimals=2)
```

In [9]:
```
corr1.to_csv("file1.csv")
```

In [10]:
```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
np.random.seed(123)
```

Here we have dropped all the columns having correlation greater than 0.9

After dropping we are now left with 24 columns out of 41

In [15]:
```
wind_data.head()
```

Out[15]:

	Torque_ Blade1	Torque_ Blade2	Torque_ Blade3	Density_in_Air	PH_A_Voltage_Amps	PH_Voltage_AB	PH_A_Voltage	Angle_Mo
0	0.488559	2.982536	0.872205	1.661788	459.047948	0.755399	10.554844	179.283
1	2.086069	3.022398	1.376642	1.663072	590.868528	0.755706	7.620256	171.540
2	4.481245	5.839726	1.021304	1.663500	607.994237	0.756013	8.701336	204.708
3	-0.959085	2.053630	4.795327	1.661740	506.361961	0.756320	7.057755	172.020
4	3.922131	0.068994	2.731100	1.660629	548.720577	0.756627	9.987469	205.016

5 rows × 24 columns

Splitting the dataset

```
X= wind_data.drop(['Direction_Wind'],axis=1)
Y=wind_data['Direction_Wind']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=42)
```

Training the model

```
from sklearn import  linear_model
```

```
model = linear_model.LinearRegression()
```

```
model.fit(X_train,Y_train)
```

```
LinearRegression()
```

Score of the model

```
score = model.score(X_test,Y_test)
score
```

```
0.18898819091328167
```

Taking particular columns

Taking particular columns namely - Power_PH_C","Pitch_Rate_Blade1","Pitch_Rate_Blade2","Pitch_Rate_Blade3","Speed_Hub","Speed_Wind","Angle_ Blade1","Angle_ Blade2","Angle_ Blade3

```
X = wind_data[["Power_PH_C","Pitch_Rate_Blade1","Pitch_Rate_Blade2","Pitch_Rate_Blade3","Speed_Hub","Speed_Wind","Angle_ Blade1",
Y=wind_data["Direction_Wind"]
```

Tranfroming Variales

Taking log of different variables

```
test1=pd.read_csv("sensor_wind_data.csv")
test1.head()
```

	Torque_ Blade1	Torque_ Blade2	Torque_ Blade3	Density_in_Air	PH_A_Voltage_Amps	PH_Voltage_AB	Amps_Voltage_Motor	PH_Voltage_AN	PH_A_Voltage
0	0.488559	2.982536	0.872205	1.661788	459.047948	0.755399	1383.780584	0.434857	10.554844
1	2.086069	3.022398	1.376642	1.663072	590.868528	0.755706	1775.154500	0.434727	7.620256
2	4.481245	5.839726	1.021304	1.663500	607.994237	0.756013	1826.079321	0.434597	8.701336

Model Building

```
x = df[["Error_from_Yaw", 'Speed_Hub','Speed_Wind','Angle_ Blade1','Angle_ Blade2','Angle_ Blade3']]
```

```
y=df["Direction_Wind"]
```

Splitting data into 70-30 for traning and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

```
sc = StandardScaler()
x_train_std = sc.fit_transform(x_train)
x_test_std = sc.transform(x_test)
```

Model Building

```
x = df[["Error_from_Yaw", 'Speed_Hub','Speed_Wind','Angle_ Blade1','Angle_ Blade2','Ang
```

```
y=df["Direction_Wind"]
```

Splitting data into 70-30 for traning and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

```
sc = StandardScaler()
x_train_std = sc.fit_transform(x_train)
x_test_std = sc.transform(x_test)
```

Parameters for Gradient Boosting Algorithm

```
gbr_params = {'n_estimators': 1000,
          'max_depth': 8,
          'min_samples_split': 5,
          'learning_rate': 0.01,
          'loss': 'ls'}
```

```
gbr = GradientBoostingRegressor(**gbr_params)
```

```
gbr_params = {'n_estimators': 1000,
              'max_depth': 8,
              'min_samples_split': 5,
              'learning_rate': 0.01,
              'loss': 'ls'}
```

```
print("Model Accuracy: %.3f" % gbr.score(x_train_std, y_train))
```

```
Model Accuracy: 0.933
```

```
print("Model Accuracy: %.3f" % gbr.score(x_test_std, y_test))
```

```
Model Accuracy: 0.964
```