

PREDICTING THE ENERGY OUTPUT OF WIND TURBINE BASED ON WEATHER CONDITIONS





INDEX

- Introduction of Team members
- Road Map
- Advantages and Disadvantages
- Conclusion
- References

TEAM MEMBERS:

- Member 1 and Leader

Name:D.HEMANATHAN

College:HITECH,CBE

- Member 2

Name:GURRAM KUSUMA

College:HITECH,CBE

- Member 3

Name:D.HEMAMALINI

College:HITECH,CBE

- Member 4

Name:S.HEMNATH

College:HITECH,CBE

ROAD MAP

```
graph TD; A[1. Work Distribution] --> B[2. Data Collection]; B --> C[3. Data Pre-processing]; C --> D[4. Model Design]; D --> E[5. Application Design];
```

1. Work
Distribution

2. Data
Collection

4. Model
Design

3. Data
Pre-
processing

5. Application
Desig

DATA COLLECTION

- We used the wind turbine dataset from Kaggle provided in the description of the problem and downloaded the weather conditions data of the same place and of the same time from the web.

KAGGLE DATA

T1 - Excel

Sign in

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Paste Cut Copy Format Painter Clipboard

Calibri 11 Font

Wrap Text Alignment

General Number

Conditional Formatting Styles

Insert Delete Format Cells

AutoSum Fill Clear Editing

Sort & Filter Select

A1 Date/Time

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
|----|------------------|--------------|---------------|---------------|--------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Date/Time | LV ActivePov | Wind Speed (r | Theoretical_P | Wind Direction (°) | | | | | | | | | | | | | | | | |
| 2 | 01 01 2018 00:00 | 380.04779 | 5.31133604 | 416.328908 | 259.9949036 | | | | | | | | | | | | | | | | |
| 3 | 01 01 2018 00:10 | 453.7692 | 5.67216682 | 519.917511 | 268.6411133 | | | | | | | | | | | | | | | | |
| 4 | 01 01 2018 00:20 | 306.37659 | 5.2160368 | 390.900016 | 272.5647888 | | | | | | | | | | | | | | | | |
| 5 | 01 01 2018 00:30 | 419.6459 | 5.65967417 | 516.127569 | 271.2580872 | | | | | | | | | | | | | | | | |
| 6 | 01 01 2018 00:40 | 380.6507 | 5.57794094 | 491.702972 | 265.6742859 | | | | | | | | | | | | | | | | |
| 7 | 01 01 2018 00:50 | 402.392 | 5.60405207 | 499.436385 | 264.5786133 | | | | | | | | | | | | | | | | |
| 8 | 01 01 2018 01:00 | 447.60571 | 5.79300785 | 557.372363 | 266.1636047 | | | | | | | | | | | | | | | | |
| 9 | 01 01 2018 01:10 | 387.24219 | 5.30604982 | 414.898179 | 257.9494934 | | | | | | | | | | | | | | | | |
| 10 | 01 01 2018 01:20 | 463.65121 | 5.58462906 | 493.677652 | 253.4806976 | | | | | | | | | | | | | | | | |
| 11 | 01 01 2018 01:30 | 439.72571 | 5.52322817 | 475.706783 | 258.7237854 | | | | | | | | | | | | | | | | |
| 12 | 01 01 2018 01:40 | 498.1817 | 5.72411585 | 535.841397 | 251.8509979 | | | | | | | | | | | | | | | | |
| 13 | 01 01 2018 01:50 | 526.81622 | 5.93419886 | 603.014077 | 265.5046997 | | | | | | | | | | | | | | | | |
| 14 | 01 01 2018 02:00 | 710.58728 | 6.54741383 | 824.662514 | 274.2329102 | | | | | | | | | | | | | | | | |
| 15 | 01 01 2018 02:10 | 655.19427 | 6.19974613 | 693.472641 | 266.7331848 | | | | | | | | | | | | | | | | |
| 16 | 01 01 2018 02:20 | 754.76251 | 6.50538301 | 808.098138 | 266.7604065 | | | | | | | | | | | | | | | | |
| 17 | 01 01 2018 02:30 | 790.17328 | 6.63411617 | 859.459021 | 270.4931946 | | | | | | | | | | | | | | | | |
| 18 | 01 01 2018 02:40 | 742.98529 | 6.37891293 | 759.434537 | 266.5932922 | | | | | | | | | | | | | | | | |
| 19 | 01 01 2018 02:50 | 748.22961 | 6.44665289 | 785.28101 | 265.5718079 | | | | | | | | | | | | | | | | |
| 20 | 01 01 2018 03:00 | 736.64783 | 6.41508293 | 773.172863 | 261.1586914 | | | | | | | | | | | | | | | | |
| 21 | 01 01 2018 03:10 | 787.24622 | 6.43753099 | 781.771216 | 257.5602112 | | | | | | | | | | | | | | | | |
| 22 | 01 01 2018 03:20 | 722.86407 | 6.22002411 | 700.7647 | 255.9264984 | | | | | | | | | | | | | | | | |
| 23 | 01 01 2018 03:30 | 935.03339 | 6.89802599 | 970.736627 | 250.0128937 | | | | | | | | | | | | | | | | |
| 24 | 01 01 2018 03:40 | 1220.609 | 7.60971117 | 1315.04893 | 255.9857025 | | | | | | | | | | | | | | | | |
| 25 | 01 01 2018 03:50 | 1053.772 | 7.28835583 | 1151.26574 | 255.4445953 | | | | | | | | | | | | | | | | |
| 26 | 01 01 2018 04:00 | 1493.808 | 7.94310188 | 1497.58372 | 256.4074097 | | | | | | | | | | | | | | | | |
| 27 | 01 01 2018 04:10 | 1724.488 | 8.37616158 | 1752.19966 | 252.4125977 | | | | | | | | | | | | | | | | |
| 28 | 01 01 2018 04:20 | 1626.0251 | 8.22605755 | 1668.47071 | 247.0704006 | | | | | | | | | | | | | | | | |

Ready Accessibility: Unavailable

Type here to search

10:03 PM 11/14/2022

WEATHER DATA

| Clipboard | | Font | | Alignment | | Number | | Styles | | Cells | | Editing | | | | | |
|----------------------|----------|-------------|-----------|----------------------|------|------------|-----------|----------|---------------|---------------|---|---------|---|---|------------------|---|---|
| B6 | | X | ✓ | <i>f_x</i> | 45 F | | | | | | | | | | Full-screen Snap | | |
| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| 1 | Time | Temperature | Dew Point | Humidity | Wind | Wind Speed | Wind Gust | Pressure | Precipitation | Condition | | | | | | | |
| 2 | 12:20 AM | 43 F | 41 F | 93 % | SW | 7 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 3 | 12:50 AM | 43 F | 39 F | 87 % | WSW | 7 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 4 | 1:20 AM | 45 F | 39 F | 81 % | WSW | 7 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 5 | 1:50 AM | 45 F | 39 F | 81 % | SW | 7 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 6 | 2:20 AM | 45 F | 41 F | 87 % | SW | 8 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 7 | 3:50 AM | 45 F | 41 F | 87 % | SW | 6 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 8 | 4:20 AM | 45 F | 41 F | 87 % | SSW | 8 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 9 | 4:50 AM | 45 F | 41 F | 87 % | SW | 10 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 10 | 5:20 AM | 45 F | 41 F | 87 % | SW | 9 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 11 | 5:50 AM | 45 F | 39 F | 81 % | SW | 9 mph | 0 mph | 29.72 in | 0.0 in | Partly Cloudy | | | | | | | |
| 12 | 6:20 AM | 43 F | 39 F | 87 % | WSW | 7 mph | 0 mph | 29.75 in | 0.0 in | Partly Cloudy | | | | | | | |
| 13 | 6:50 AM | 41 F | 37 F | 87 % | SW | 7 mph | 0 mph | 29.72 in | 0.0 in | Partly Cloudy | | | | | | | |
| 14 | 7:20 AM | 39 F | 37 F | 93 % | S | 6 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 15 | 7:50 AM | 41 F | 37 F | 87 % | SSE | 5 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 16 | 8:20 AM | 41 F | 39 F | 93 % | SSW | 3 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 17 | 8:50 AM | 43 F | 39 F | 87 % | S | 5 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 18 | 9:20 AM | 45 F | 39 F | 81 % | SSW | 3 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 19 | 9:50 AM | 45 F | 37 F | 76 % | S | 6 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 20 | 10:20 AM | 46 F | 39 F | 76 % | S | 7 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 21 | 10:50 AM | 46 F | 39 F | 76 % | S | 7 mph | 0 mph | 29.75 in | 0.0 in | Fair | | | | | | | |
| 22 | 11:20 AM | 50 F | 39 F | 66 % | SSW | 8 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 23 | 11:50 AM | 50 F | 37 F | 62 % | SSW | 8 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 24 | 12:20 PM | 50 F | 37 F | 62 % | SSW | 8 mph | 0 mph | 29.72 in | 0.0 in | Fair | | | | | | | |
| 25 | 12:50 PM | 50 F | 37 F | 62 % | SW | 10 mph | 0 mph | 29.70 in | 0.0 in | Fair | | | | | | | |
| 26 | 1:20 PM | 52 F | 37 F | 58 % | WSW | 10 mph | 0 mph | 29.70 in | 0.0 in | Fair | | | | | | | |
| 27 | 1:50 PM | 52 F | 37 F | 58 % | SW | 13 mph | 0 mph | 29.70 in | 0.0 in | Fair | | | | | | | |
| 28 | 2:20 PM | 52 F | 39 F | 62 % | WSW | 15 mph | 0 mph | 29.70 in | 0.0 in | Fair | | | | | | | |
| Sheet1 Sheet2 Sheet3 | | | | | | | | | | | | | | | | | |

100%

Type here to search

Hackathon Video Ps Pendik Turke Anaconda Pr Weather Dat 00:54

Data pre-processing

- Activity 1 : Taking care of the missing data
- Solution - The missing data is less than 3% of the total data so we have filled the missing data by the average value of the respective column data.
- Activity 2 : Feature scaling
- Solution - Weather data has a scale of 30 minutes while the wind turbine data from Kaggle has a scale of 10 minutes so in order to merge both the data's we have converted each of them into the same scale (i.e. scale of 30 minutes). Then we merged both the data's into one and obtained our final data.

Handle Missing Data and Merging the data

DATA PRE-PROCESSING

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

```
In [3]: path = "T1.csv"
df = pd.read_csv(path)
df.rename(columns={"Date/Time": "Time",
                  "LV ActivePower (kW)": "ActivePower(KW)",
                  "Wind Speed (m/s)": "WindSpeed(m/s)",
                  "Wind Direction(°)": "Wind_Direction"},
         inplace=True)
```

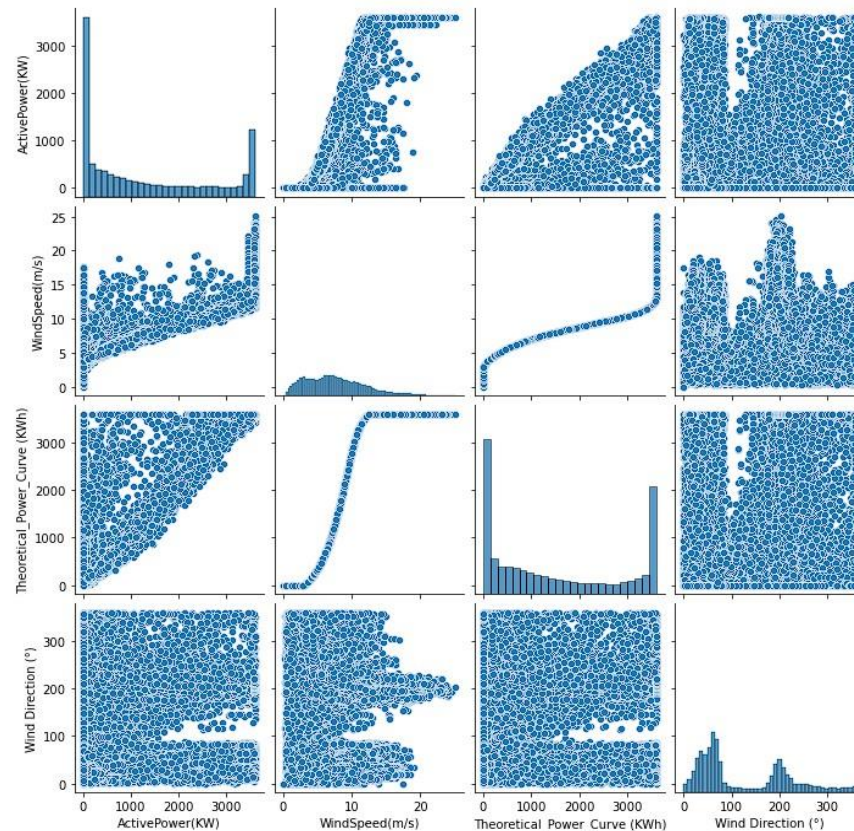
```
In [4]: df.isnull().sum()
```

```
Out[4]: Time                0
ActivePower(KW)            0
WindSpeed(m/s)             0
Theoretical_Power_Curve (KWh) 0
Wind Direction (°)         0
dtype: int64
```

- **Activity 3:** Visualization of the Data
- **Solution-** We analyzed the data and studied the impact of different features on output (active power). With the results we achieved, we eradicated all the unnecessary features from the model

```
In [4]: sns.pairplot(df)
```

```
Out[4]: <seaborn.axisgrid.PairGrid at 0x155c46e1d00>
```



```
In [5]: plt.figure(figsize=(10, 8))
corr = df.corr()
ax = sns.heatmap(corr, vmin = -1, vmax = 1, annot = True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)

print(corr)
```

```
ActivePower(KW)    WindSpeed(m/s) \
ActivePower(KW)    1.000000    0.912774
WindSpeed(m/s)    0.912774    1.000000
Theoretical_Power_Curve (KWh)    0.949918    0.944209
Wind Direction (°)    -0.062702    -0.077188
```

```
Theoretical_Power_Curve (KWh) \
ActivePower(KW)    0.949918
WindSpeed(m/s)    0.944209
Theoretical_Power_Curve (KWh)    1.000000
Wind Direction (°)    -0.099076
```

```
Wind Direction (°)
ActivePower(KW)    -0.062702
WindSpeed(m/s)    -0.077188
Theoretical_Power_Curve (KWh)    -0.099076
Wind Direction (°)    1.000000
```



Model Design

- **Activity 1:** Deciding parameters or features
- **Solution-** With the results we achieved we have selected following as our features for the model :
 - 1.Wind Speed
 - 2.Temperature
 - 3.Humidity
 - 4.Pressure
- **Activity 2:** Separating Training and Testing data from the data set
- **Solution-** This process will be done inside the model file itself by using the `train_test_split` library in the `sklearn` package of python.
- **Activity 3:** Developing the model
- **Solution-** We use the boosted trees regressor model to train and evaluate the model using TensorFlow. We decided to train the model using 120 trees to optimize the model and preventing it from underfit also we have used entire batch of training data to train so that model can accurately be trained.

APPLICATION DESIGN

- **Our initial plans included building the website in Html, but we wanted to provide our end-user the solution in single click, . We used Flask and Html framework for designing the UI of the website and Jupyter Python for the back-end.**

- **Activity 1:** Build the UI

- **Solution-** We used Flask and Html to design our Website. Flask reduces code development time to huge extent because of its UI and html access. It allows seeing the applied changes almost instantly, without even losing the current application state. Apart from this, development in Flask is very easy because it uses the unique Widget tree model. Everything in Flask UI is just a Libraries. "App route" widget is the root widget of the website and all the other libraries become its children in hierarchial pattern.

FLASK DISPLAYING HIERARCHY

```
In [1]: import numpy as np
        from flask import Flask, request, jsonify, render_template
        import joblib
        import requests
```

```
In [2]: app = Flask(__name__)
        model = joblib.load("power_prediction.sav")
```

```
In [3]: @app.route('/')
        def home():
            return render_template("intro.html")

        @app.route('/predict')
        def predict():
            return render_template("predict.html")

        @app.route('/windapi', methods=['POST'])
        def windapi():
            city = request.form.get('city')
            apikey = "d8484354b9e388875c48dae8d0d09cd1"
            url = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "&appid=" + apikey
            resp = requests.get(url)
            resp = resp.json()
            temp = str(resp["main"]["temp"]) + " °C"
            humid = str(resp["main"]["humidity"]) + "%"
            pressure = str(resp["main"]["pressure"]) + " mmHG"
            speed = str(resp["wind"]["speed"]) + " m/s"
            return render_template('predict.html', temp=temp, humid = humid, pressure=pressure, speed = speed)
```

```
In [4]: @app.route('/y_predict', methods = ['POST'])
        def y_predict():
            x_test = [[float(x) for x in request.form.values()]]

            prediction = model.predict(x_test)
            print(prediction)
            output = prediction[0]
            return render_template('predict.html', prediction_text = 'The energy predicted is {:.2f} KWh'.format(output))
```

```
In [ ]: if __name__ == "__main__":
        app.run(debug=False)

        * Serving Flask app "__main__" (lazy loading)
        * Environment: production
        WARNING: This is a development server. Do not use it in a production deployment.
        Use a production WSGI server instead.
        * Debug mode: off

        * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- Flask also has a huge library support for design and performance along with detailed documentation of each on Flask Libraries. It has no database
- A great emphasis has been put on developing the elegant UI of the website. We have used Brown theme. Through our Website we can predict the power generated by windmill for next 72 hours (on hourly basis) on any user input coordinates and current weather condition.
- Also a database of 50 cities in the India has been fed to the app which can be exploited using "**Search for weather condition**" button. User can search for the name of the cities and find the power that can be generated there. The app also provides the next 72 hour prediction of weather conditions at that particular coordinate.

HTML CODE FOR INTRO

```
<html>
  <head>
    <title>Wind Energy Prediction</title>
    <style>

      .header {
        top:0px;
        margin:0px;
        left: 0px;
        right: 0px;
        position: fixed;
        background: #6c493a;
        color: white;
        overflow: hidden;
        padding-bottom: 30px;
        font-size: 2.25vw;
        width: 100%;
        padding-left:0px;
        text-align: center;
        padding-top:20px;
      }
      .second{
        top:80px;
        bottom:0px;
        margin:0px;
        left: 0px;
        right: 0px;
        position: fixed;
        padding: 0px;
        width: 100%;
        background-image:url(https://c1.wallpaperflare.com/preview/623/531/630/596ca965a2e3f.jpg);
        background-repeat:no-repeat;
        background-size: contain;
      }
      .inside{
        top:80px;
        bottom:0px;
        margin:0px;
        left: 45%;
        right: 0%;
        position: fixed;
        padding-left: 40px;
```

```

        position: fixed;
        padding-left: 40px;
        padding-top: 8%;
        padding-right: 40px;
        background-color: #F2D19A;
        font-family: Georgia, serif;
        color: black;
        font-size: 20px;
        text-align: justify;
    }
    .myButton{
        border: none;
        text-align: center;
        cursor: pointer;
        text-transform: uppercase;
        outline: none;
        overflow: hidden;
        color: #fff;
        font-weight: 700;
        font-size: 12px;
        background-color: #6c493a;
        padding: 10px 15px;
        margin: 0 auto;
        box-shadow: 0 5px 15px rgba(0,0,0,0.20);
    }
</style>
</head>
<body>

    <div class="header">Predicting The Energy Output Of Wind Turbine Based On Weather Condition</div>
    <div class="second">
        <div class="inside">Renewable energy, such as wind and solar energy, plays an increasing role in the supply of energy worldwide. This trend will continue because glo
However, levels of production of wind energy are hard to predict as they rely on potentially unstable weather conditions present at the wind farm. In particular, wind speed is crucial for e
        <br><br><br>
        <a href="{{url_for('predict')}}"><button type="button" class="myButton" >Want to predict the energy??</button></a>
        </div>

    </div>
</body>
</html>

```

Advantages and Disadvantages

ADVANTAGES

- Weather Underground Services provide very accurate Historical Weather Data which increased the accuracy of model.
- Website is more convenient to use due to zero storage.
- With Choosing city, Website can accurately predict power output using weather condition.

DISADVANTAGES

- Weather API is paid and the free version provide limited API requests per day.
- Android Website can be deployed on IBM Cloud.
- No free server available on IBM Cloud for deploying Backend.

Conclusion

- In this project, we used Weather Underground services (subsidiary of IBM) to get accurate historical weather data. For merging this data with Windmill data we learned some Data Analysis concepts. We analyzed several ML models, and chose Random Tree Regressor to develop this model. This project gave us deep insight about the Flutter framework. We integrated the app with model and Weather API using REST API and Flask Back-end. The accuracy of Random Tree Regressor model for this project is 85%.

References

- Wind Power Data:
<https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset>
- Weather Data
<https://www.wunderground.com>
- Data Science
<https://www.youtube.com/watch?v=CmorAWRsCAw&list=PLeo1K3hjS3uuASpe-1LjfG5f14Bnozjwy>
- Climacell API
<https://www.climacell.co/weather-api/>
- Flask
<https://flask.palletsprojects.com/en/1.1.x/>
- Flutter
<https://flutter.dev/>