Predicting The Energy Output Of Wind Turbine Based On Weather Condition

BY:

Senthil murugan S
Nalapriya D
Preethika S
Saranya E

Table of Contents

1.INTRODUCTION

1.1 Overview

Over the decade there has been rapid growth in wind generation of electricity.In the wind energy industry, it is of great importance to develop models that accurately forecast the power output of a wind turbine, as such predictions are used for wind farm location assessment , monitoring, and preventive maintenance.The energy output of a wind farm is highly dependent on the weather conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction.The goal of this project is to draw a insight from the given dataset and then use a RNN(Recurrent Neural Network) algorithm by datapreprocessing and embedding to the model.Recurrent Neural Network remembers the past and it's decisions are influenced by what it has learnt from the past.This method is believed to bring more information for predicting the energy output of wind turbine and in result have better determination of wind generation output based on weather condition.Thus wind power forecasting plays a key role in dealing with the challenges of balancing supply and demand in any electricity system, given the uncertainty associated with the wind farm power output. Hence the time series model is devoloped to predict the power output in wind farm based on weather condition.

1.2 Purpose

The capacity of wind energy production has been substantially increased during the last years. 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 energy production based on wind, and it may vary drastically over time. So the purpose of this project is to give accurate predictions and to avoid overproduction by coordinating the collaborative production of traditional power plants and weather-dependent energy sources.

2.LITERATURE SURVEY

2.1 Existing problem

The supervisory control and data acquisition (SCADA) data to model the wind turbine power curve (WTPC) existing model are:

- Kusiak, Zheng, and Song have shown how wind speed data may be used to predict the power output of a wind farm based on time-series prediction modeling. Neural networks are a very popular learning approach for wind power forecasting based on given time series. They provide an implicit model of the function that maps the given weather data to an energy output.

- Jursa and Rohrig have used particle swarm optimization and differential evolution to minimize the prediction error of neural networks for short-term windpower forecasting.

- Kramer and Gieseke used support vector regression for short term energy forecast and kernel methods and neural networks to analyze wind energy time series .

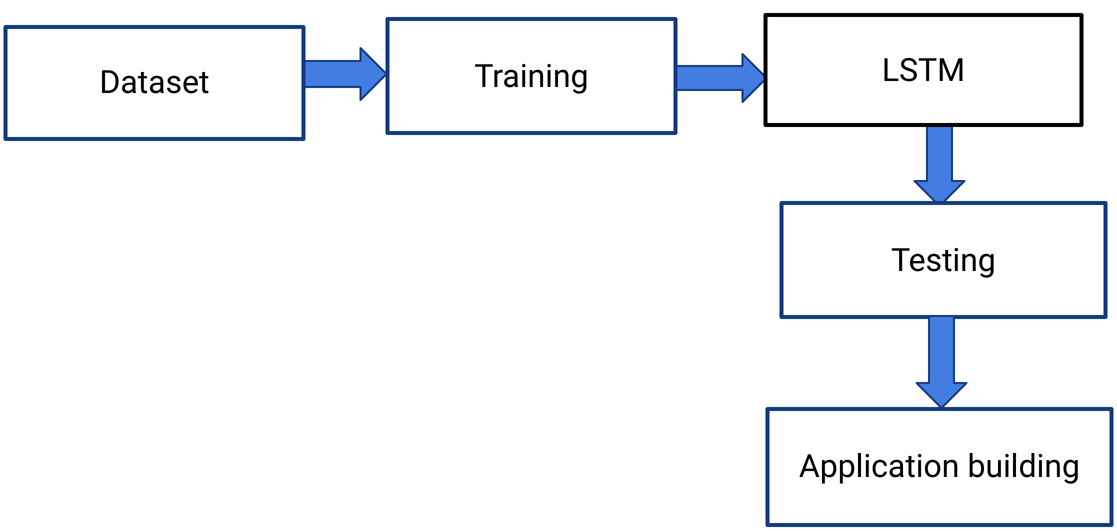
2.2 Proposed solution

The aim of this project is to use the SCADA data to check the feasibility of wind energy prediction and to identify the minimal subset of driving weather features that are significantly related to the wind energy output of the wind farm. At the modeling stage, we reduce the training data to the set of selected inputs using RNN algorithm to obtain models and also by plotting the graph, and model ensembles for predicting energy output.This time series model is devoloped to predict the future power output in wind farm based on weather condition and to give accurate prediction using flask API .

3.THEORITICAL ANALYSIS

In the statistical approach a vast amount of data is analyzed and meteorological processes are not explicitly represented. The link between historical power production and weather is determined and then used to forecast the future power output. Unlike physical methods, statistical methods involve only one-step to convert the input variables into power output.As soon as weather and energy data from different sources were put in an appropriate input-output form, we were able to apply a standard data-driven modeling approach to them. A good approach employs iterations among three stages: Data Collection/Reduction, Model Development, and Model Analysis and Variable Selection. In hard problems, many iterations are required to identify a subspace of minimal dimensionality where models of appropriate accuracy and complexity tradeoffs can be built.RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer. The neural network model is found to possess better performance than the regression model for turbine power curve estimation under complicated influence factors.

3.1 Block diagram



3.2 Hardware/Software designing

Hardware requirement:

1.Operating system: Windows 10
2.Hard disk: 500 GB
3.RAM: 2GB

Software requirement:

1.Anaconda navigator
2.Jupyter notebook
3.Spyder

4. EXPERIMENTAL INVESTIGATION:

4.1 Dataset Collection:

- The dataset was downloaded from the kaggle repository.
  https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset
- The dataset is in the format of .csv type file.
- It has 5 columns, We are using Theoretical Power column for building the model.

	A	B	C	D	E	F	G
1	Date/Tim	LV ActiveF	Wind Spe	Theoretic	Wind Direction (Â°)		
2	01 01 2018	380.0478	5.311336	416.3289	259.9949		
3	01 01 2018	453.7692	5.672167	519.9175	268.6411		
4	01 01 2018	306.3766	5.216037	390.9	272.5648		
5	01 01 2018	419.6459	5.659674	516.1276	271.2581		
6	01 01 2018	380.6507	5.577941	491.703	265.6743		
7	01 01 2018	402.392	5.604052	499.4364	264.5786		
8	01 01 2018	447.6057	5.793008	557.3724	266.1636		
9	01 01 2018	387.2422	5.30605	414.8982	257.9495		
10	01 01 2018	463.6512	5.584629	493.6777	253.4807		
11	01 01 2018	439.7257	5.523228	475.7068	258.7238		
12	01 01 2018	498.1817	5.724116	535.8414	251.851		
13	01 01 2018	526.8162	5.934199	603.0141	265.5047		
14	01 01 2018	710.5873	6.547414	824.6625	274.2329		
15	01 01 2018	655.1943	6.199746	693.4726	266.7332		
16	01 01 2018	754.7625	6.505383	808.0981	266.7604		
17	01 01 2018	790.1733	6.634116	859.459	270.4932		
18	01 01 2018	742.9853	6.378913	759.4345	266.5933		
19	01 01 2018	748.2296	6.446653	785.281	265.5718		
20	01 01 2018	736.6478	6.415083	773.1729	261.1587		
21	01 01 2018	787.2462	6.437531	781.7712	257.5602		
22	01 01 2018	722.8641	6.220024	700.7647	255.9265		
23	01 01 2018	935.0334	6.898026	970.7366	250.0129		
24	01 01 2018	1220.609	7.609711	1315.049	255.9857		
25	01 01 2018	1053.772	7.288356	1151.266	255.4446		
26	01 01 2018	1493.808	7.943102	1497.584	256.4074		
27	01 01 2018	1724.488	8.376162	1752.2	252.4126		
28	01 01 2018	1636.935	8.236958	1668.471	247.9794		
29	01 01 2018	1385.488	7.879591	1461.816	238.6096		
30	01 01 2018	1098.932	7.101376	1062.285	245.0956		

weather_train

Figure 4.1- Dataset

4.2 Importing Libraries:

- Keras is an open source neural-network library written in Python.
- numpy is used for performing numerical operations in Python.
- sklearn library provides many supervised and unsupervised learning algorithms and also used for preprocessing techniques like feature scaling.
- pandas library is used for data analysis.
- seaborn is used for data visualization.

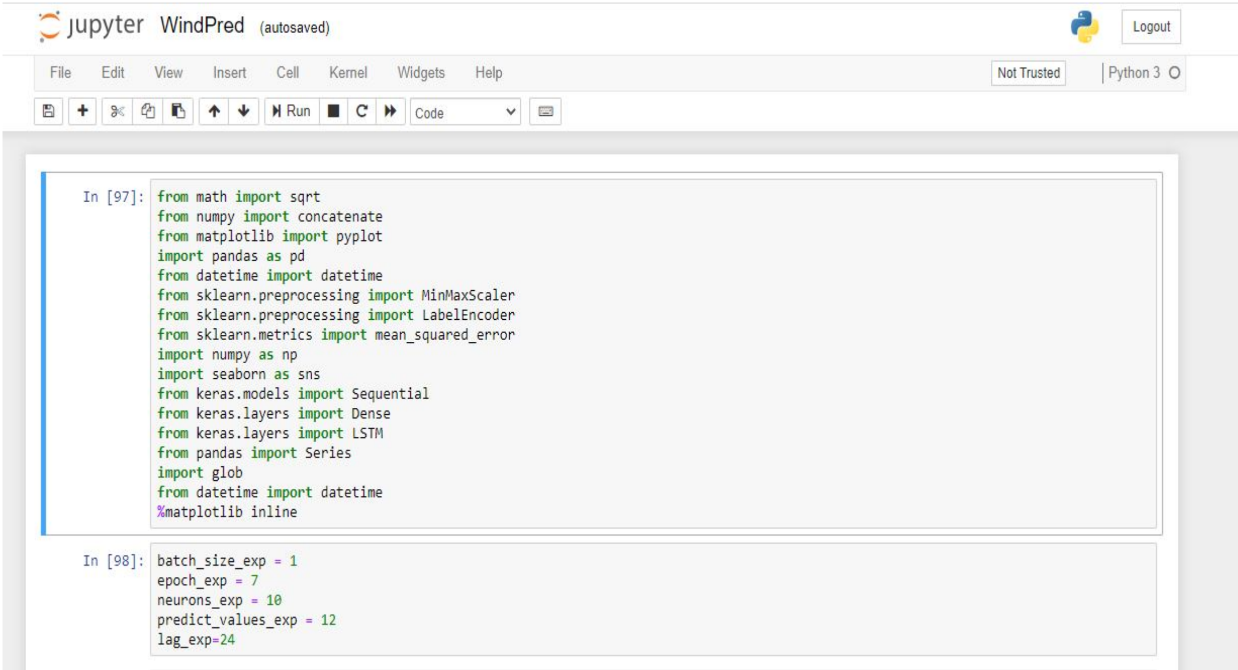


Figure 4.2- Importing Libraries

4.3 Model Building:

- Define the methods for ranging the iputs and outputs of a model.
- Apply the scaling functions on data.
- Initialize the model.
- Add the RNN(LSTM) layer.
- Add the Dense layer and give the output units.
- Compile the model by giving the loss as "mse".
- Fit the model.
- Predict the output by random prediction.
- Save the model .

4.4 Build an UI:

To create interactive webpage the Web technologies like HTML, CSS, JavaScript, PHP and JQuery are used.HTML - HTML is the standard markup language for creating Web pages. It describes the structure of a Web page. It consists of a series of elements. They tell the browser how to display the content. CSS - CSS stands for Cascading Style Sheets. It describes how HTML elements are to be displayed on screen, paper, or in other media. It can control the layout of multiple web pages all at once. Javascript - Javascript is the Programming language of HTML and the web.PHP - PHP is used for interaction with the files and for prediction call . JQuery - jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website.

Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of current module (__name__) as argument Pickle library to load the model file.

Open anaconda prompt from start menu.Navigate to the folder where your app.py resides.Now type “python app.py” command. It will show the local host where your app is running. Navigate to the localhost where you can view your web page. Enter the images and see the prediction on web page.

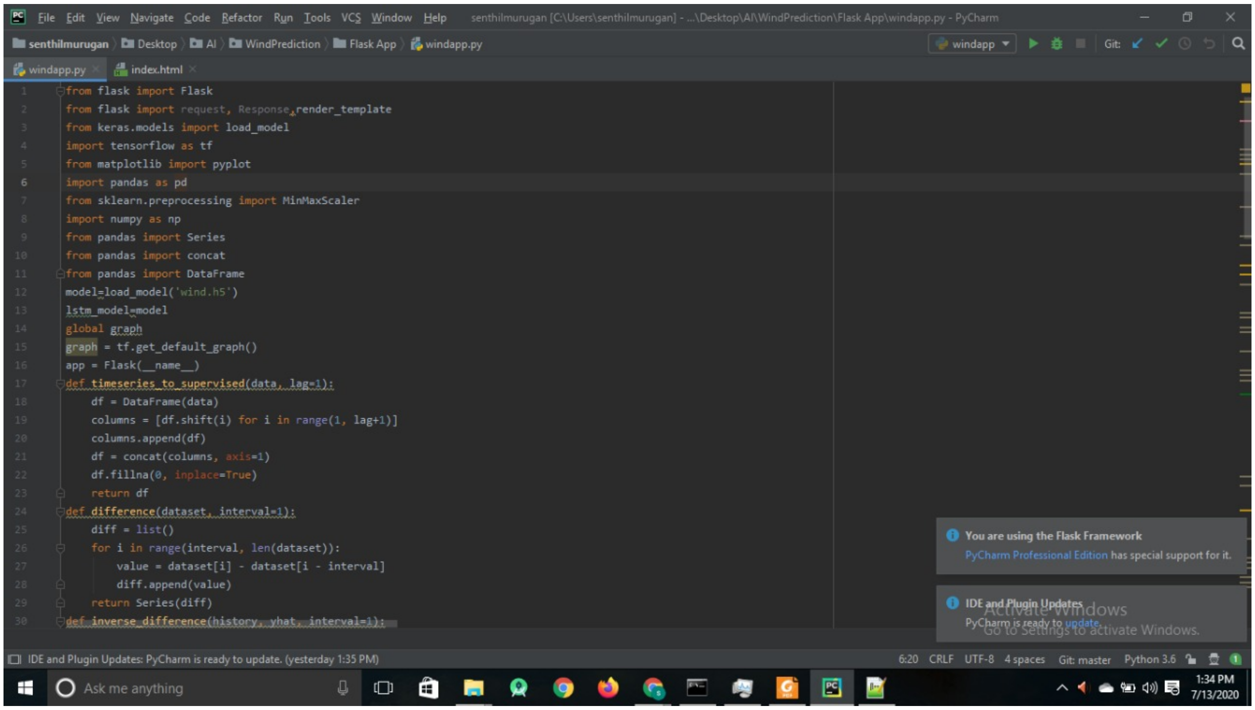


Figure 4.3- Flask code for UI

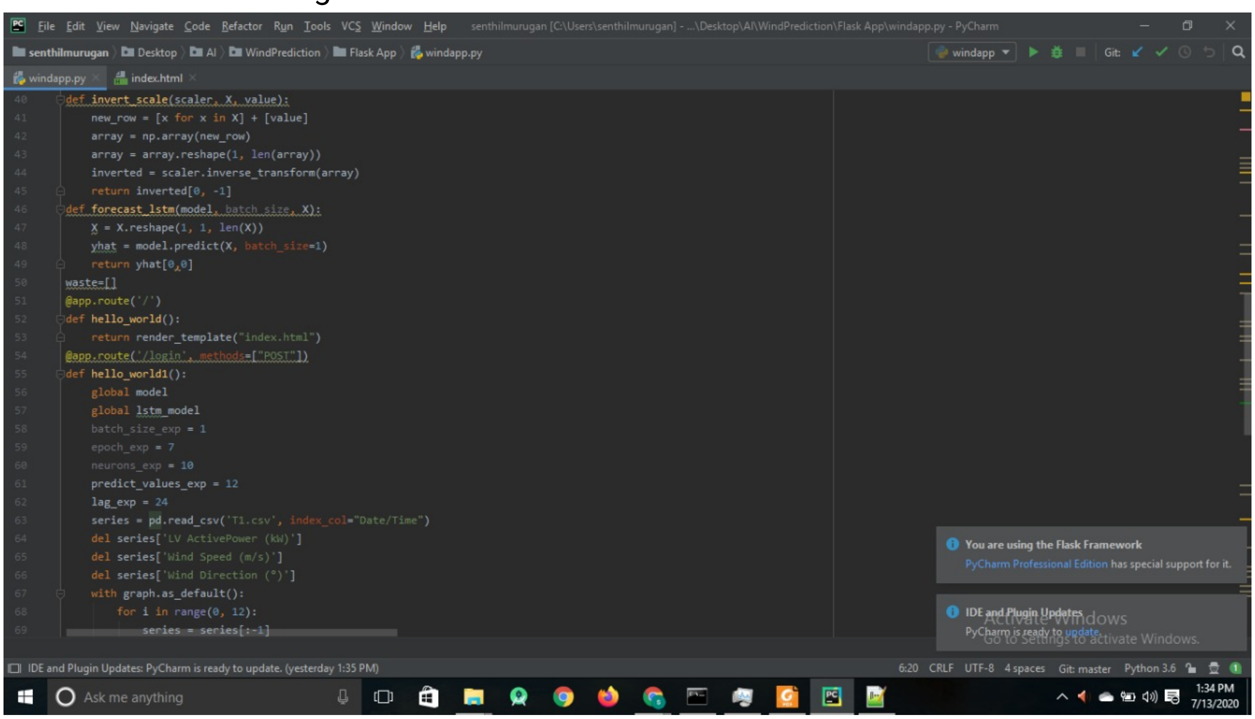


Figure 4.4 - Flask code for UI

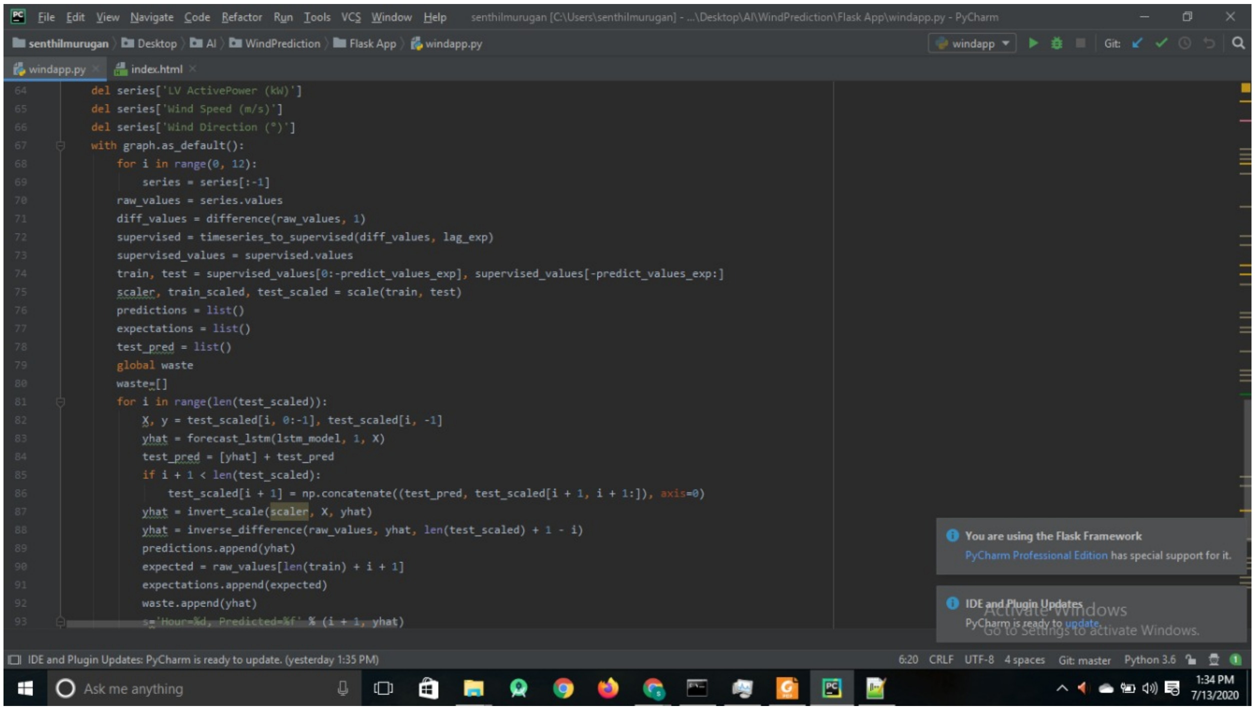


Figure 4.5 - Flask code for UI

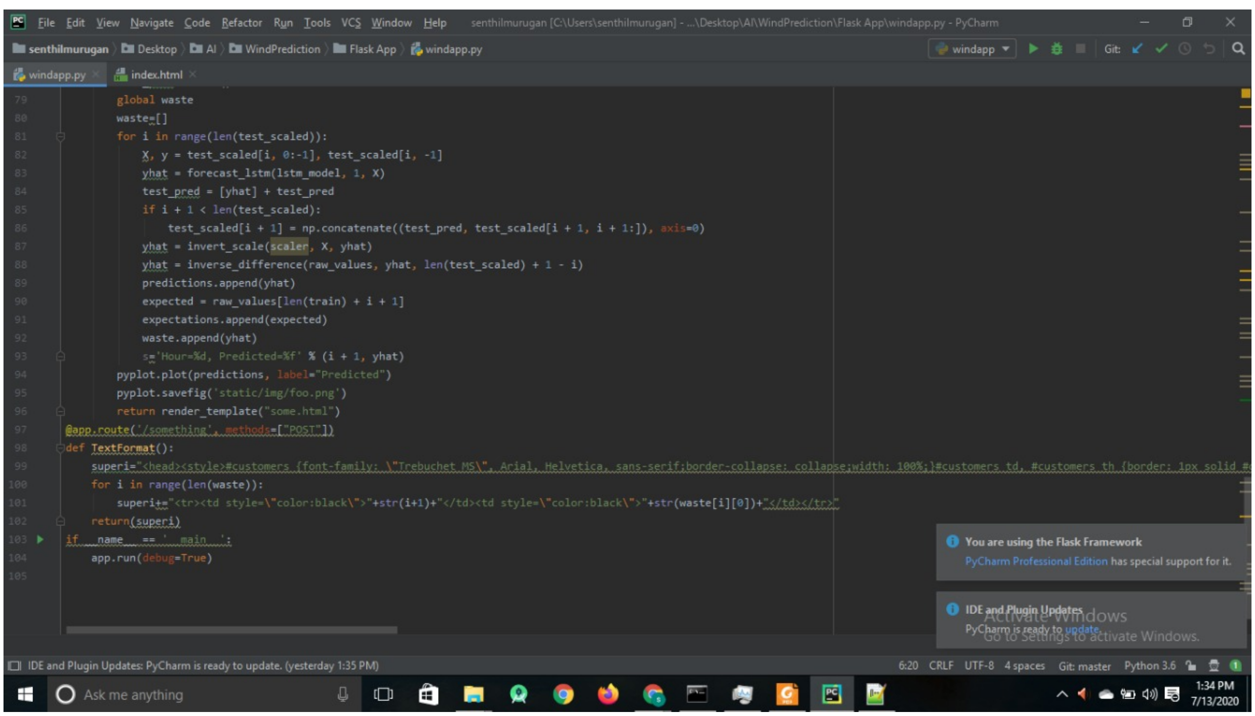


Figure 4.6 - Flask code for UI

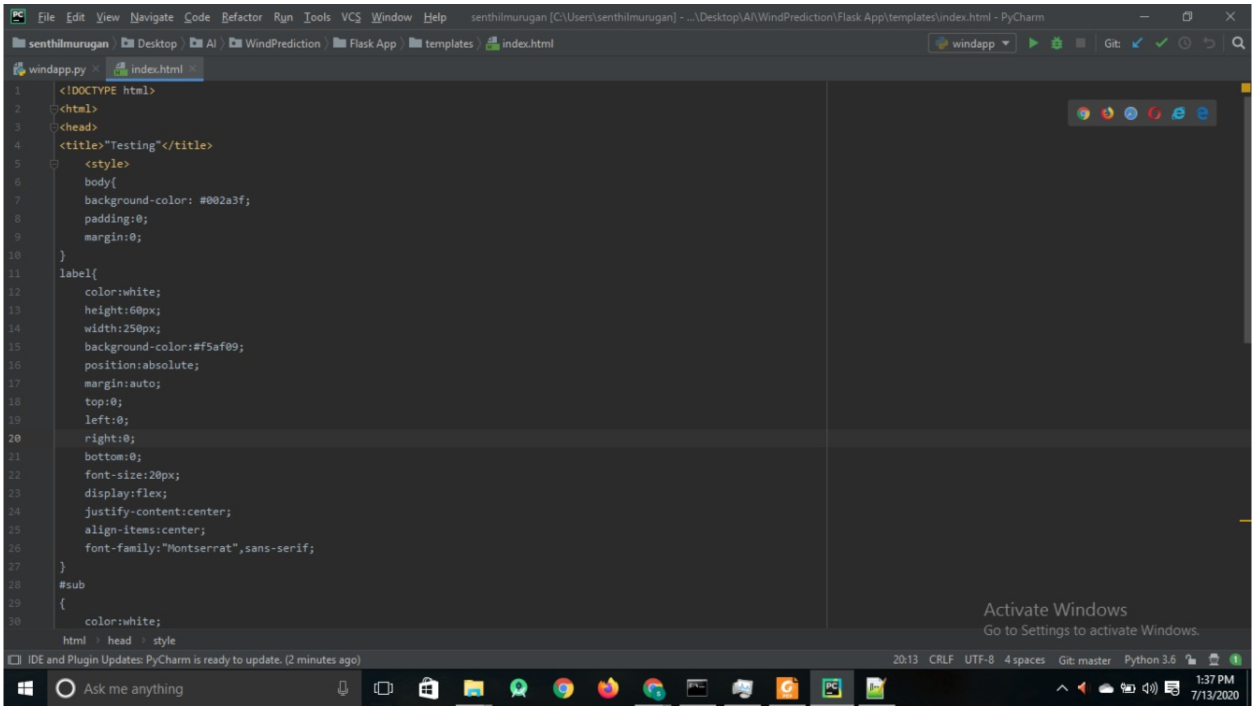


Figure 4.7 - HTML front end code for UI

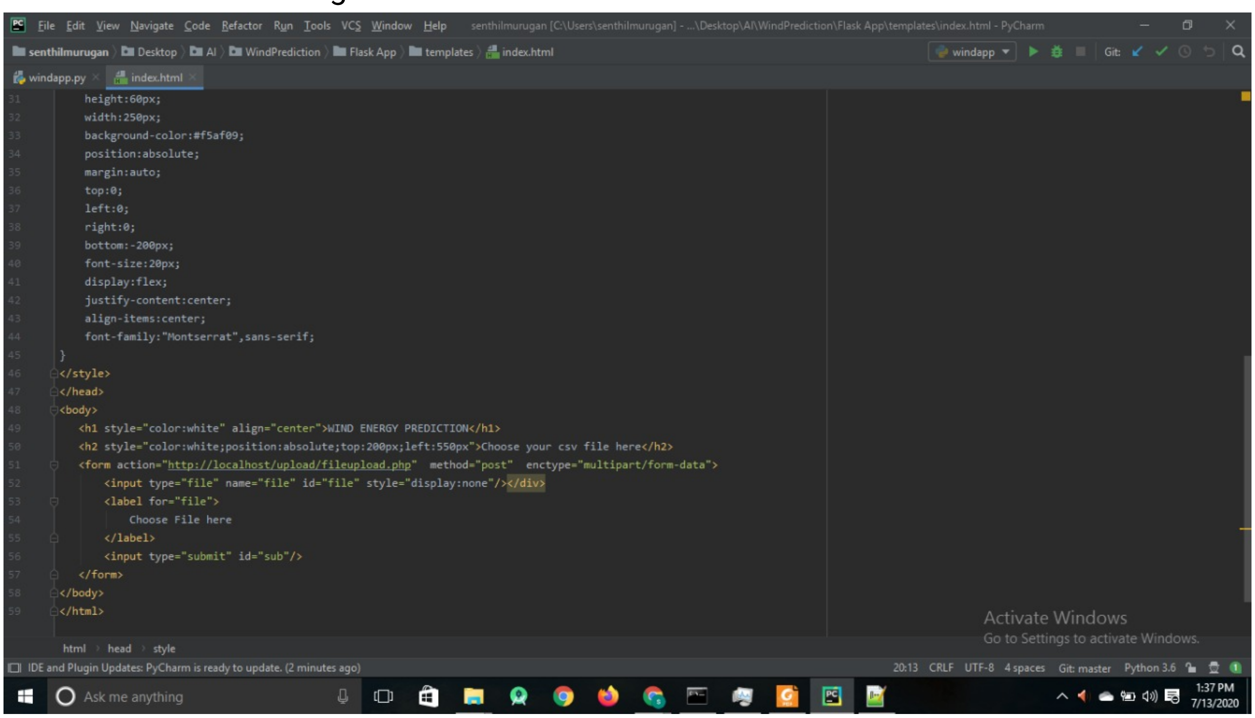
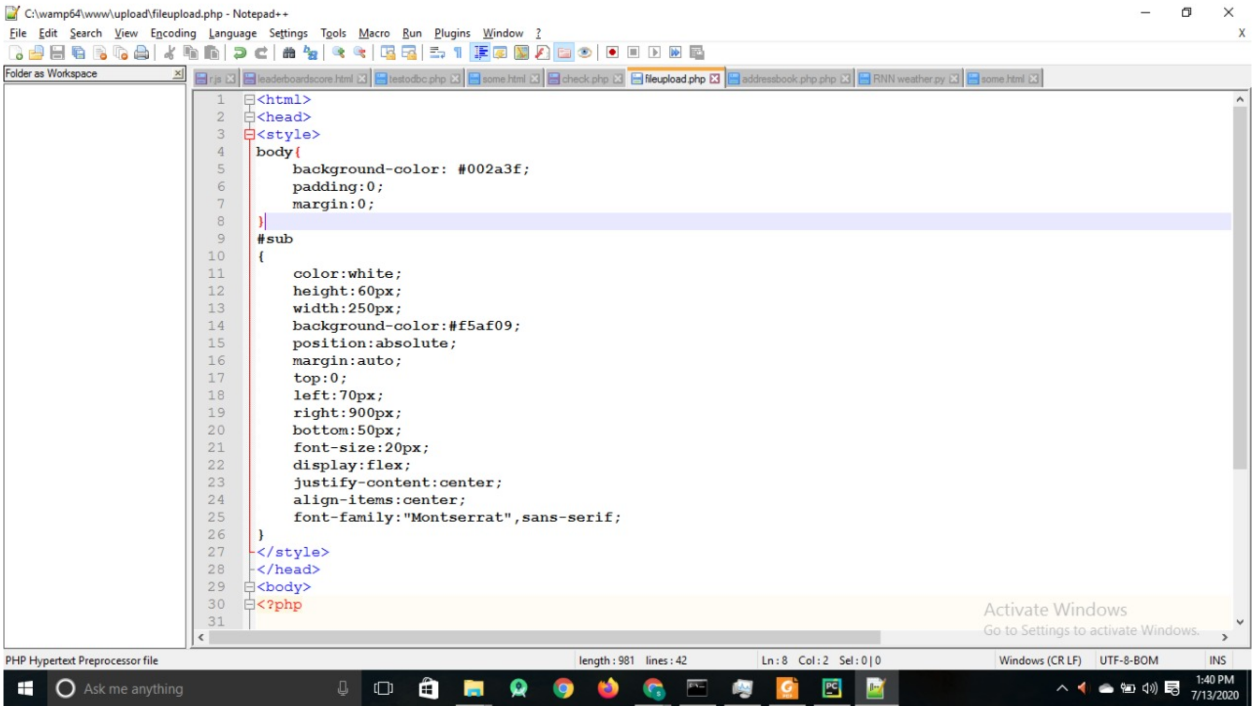
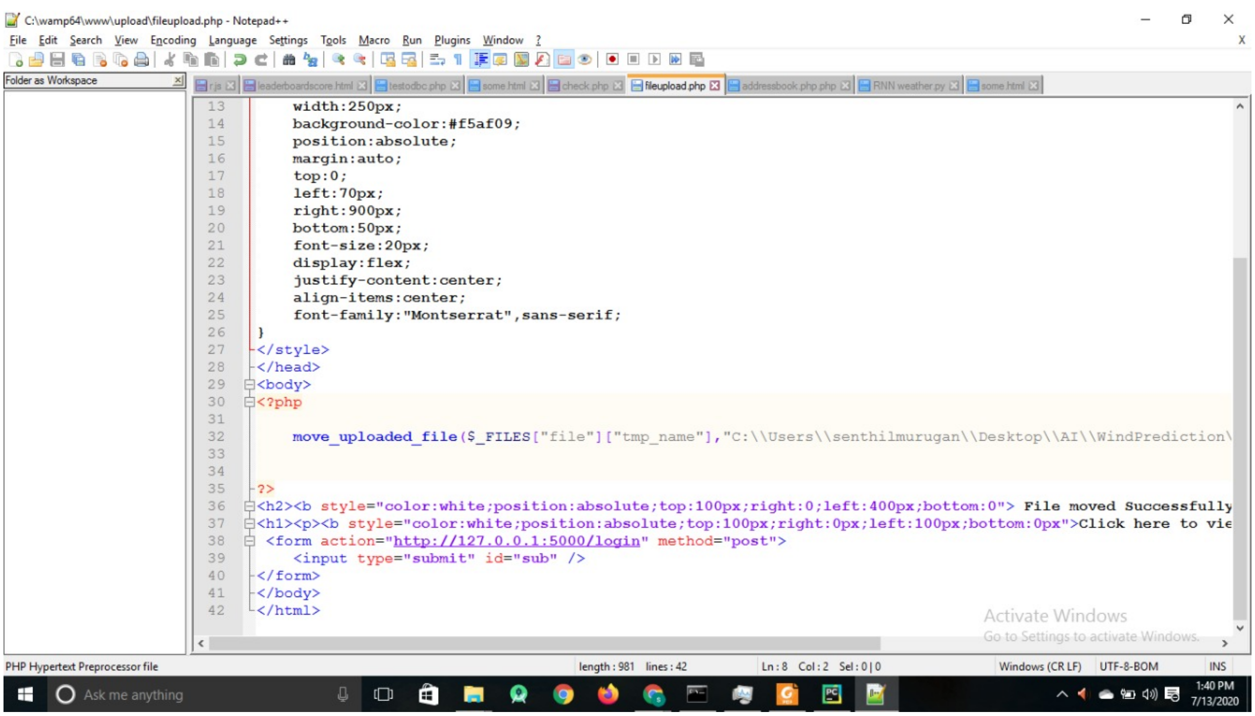


Figure 4.8 - HTML front end code for UI



```
<html>
<head>
<style>
body{
    background-color: #002a3f;
    padding:0;
    margin:0;
}
#sub
{
    color:white;
    height:60px;
    width:250px;
    background-color:#f5af09;
    position:absolute;
    margin:auto;
    top:0;
    left:70px;
    right:900px;
    bottom:50px;
    font-size:20px;
    display:flex;
    justify-content:center;
    align-items:center;
    font-family:"Montserrat",sans-serif;
}
</style>
</head>
<body>
<?php
```

Figure 4.9 - PHP code for UI



```
    width:250px;
    background-color:#f5af09;
    position:absolute;
    margin:auto;
    top:0;
    left:70px;
    right:900px;
    bottom:50px;
    font-size:20px;
    display:flex;
    justify-content:center;
    align-items:center;
    font-family:"Montserrat",sans-serif;
}
</style>
</head>
<body>
<?php

    move_uploaded_file($_FILES["file"]["tmp_name"],"C:\\Users\\senthilmurugan\\Desktop\\AI\\WindPrediction\


?>
<h2><b style="color:white;position:absolute;top:100px;right:0;left:400px;bottom:0"> File moved Successfully
<h1><p><b style="color:white;position:absolute;top:100px;right:0px;left:100px;bottom:0px">Click here to vie
 <form action="http://127.0.0.1:5000/login" method="post">
    <input type="submit" id="sub" />
</form>
</body>
</html>
```

Figure 4.10 - PHP code for calling prediction file

5.FLOWCHART:

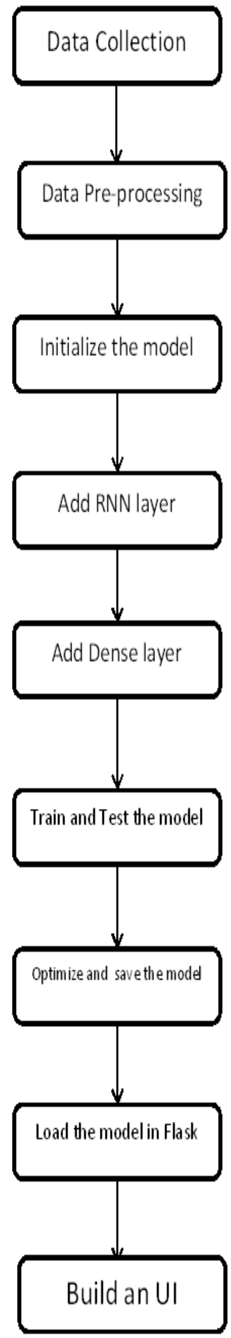


Figure 5.1- Flowchart

6.RESULTS:

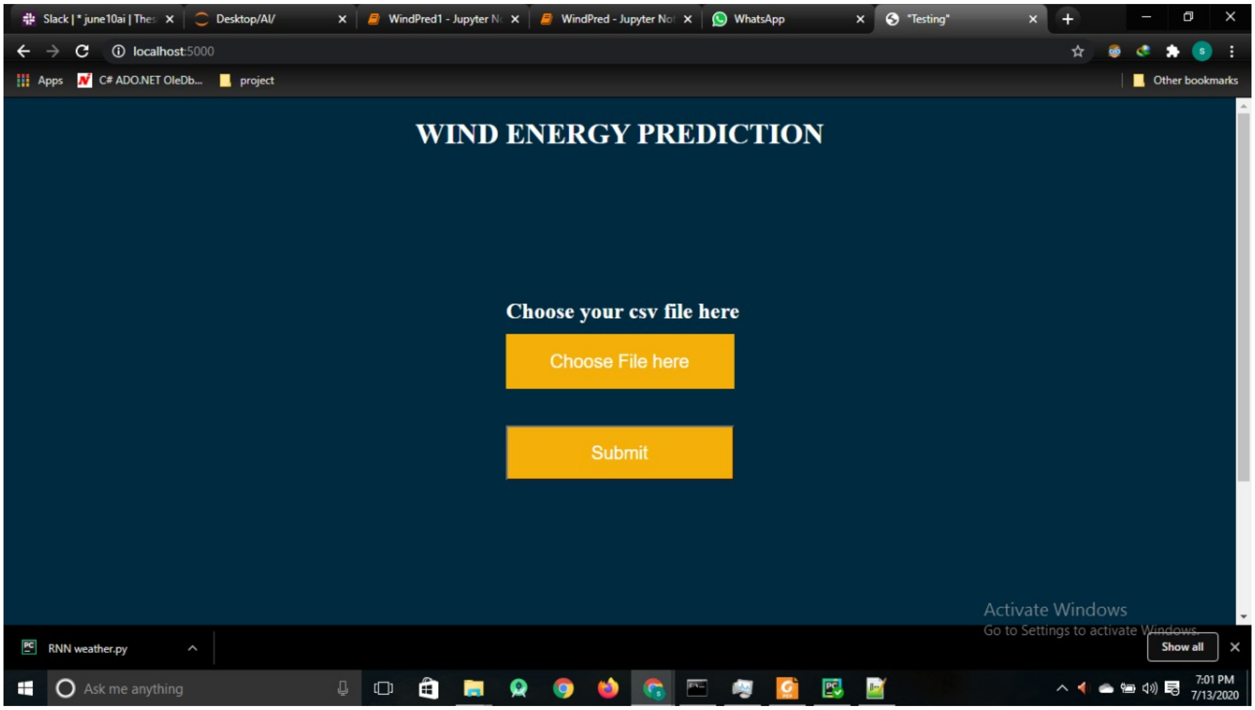


Figure 6.1- User Interface

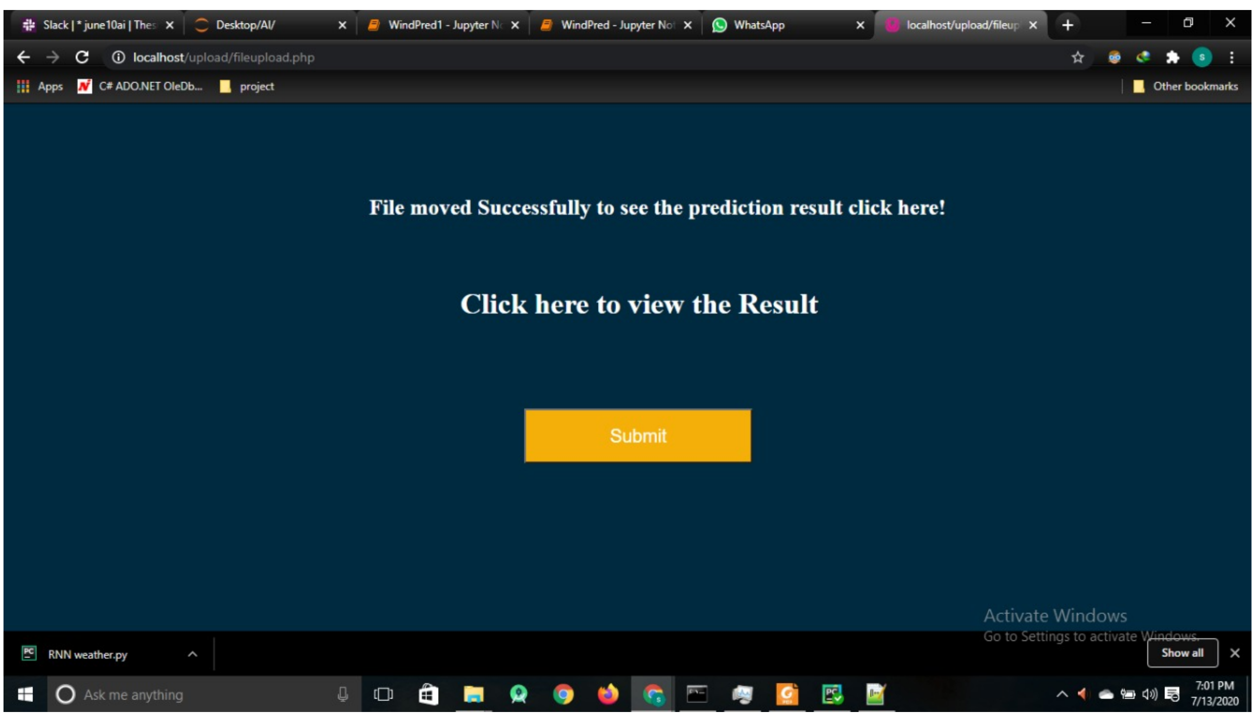


Figure 6.2- After choosing the file

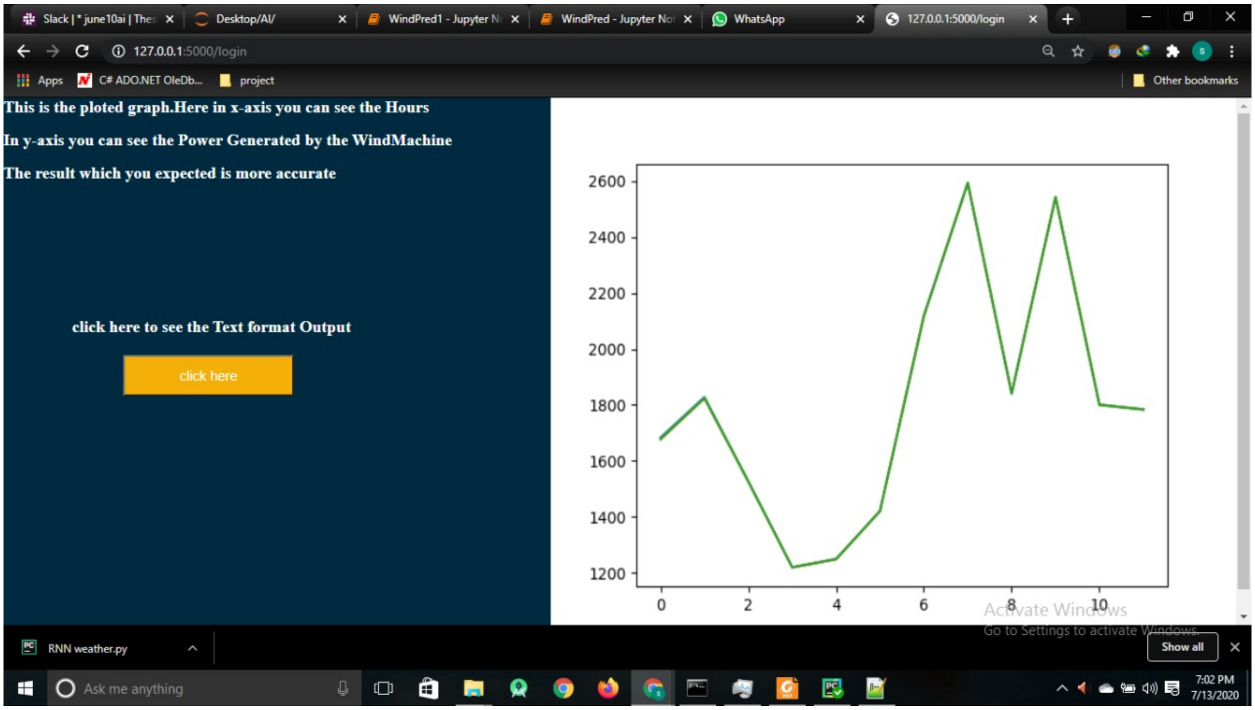
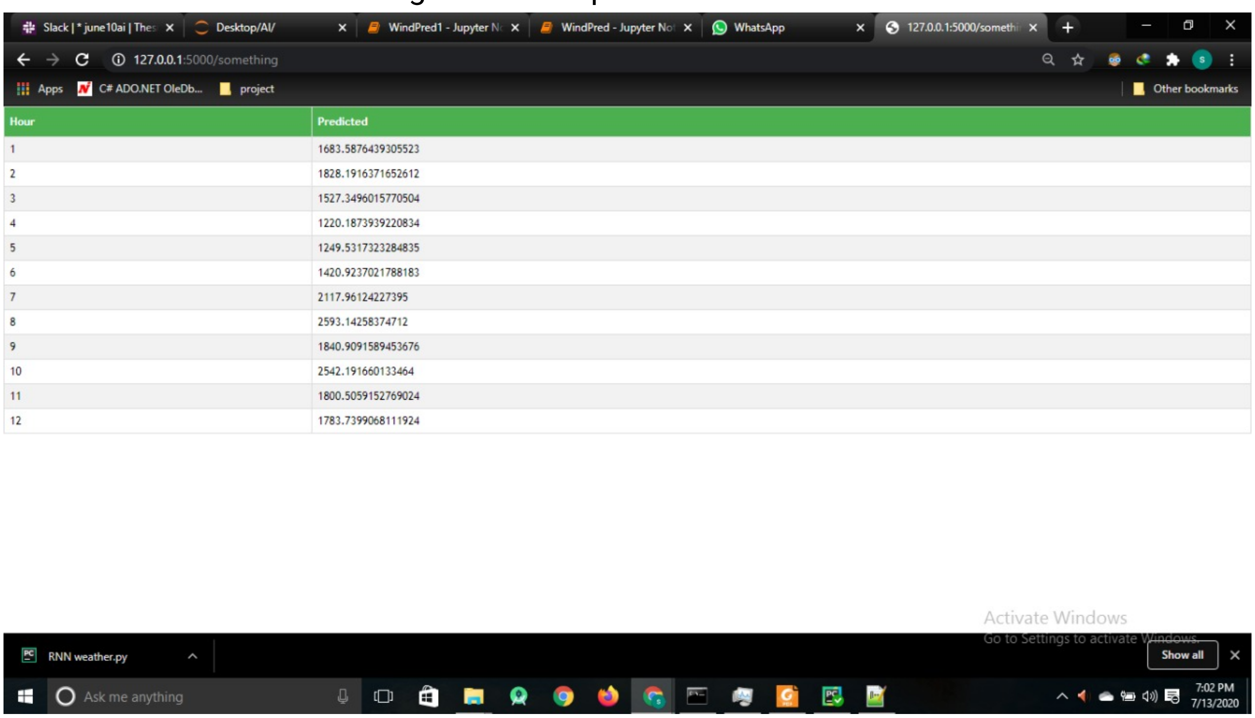


Figure 6.3- Output Prediction



Hour	Predicted
1	1683.5876439305523
2	1828.1916371652612
3	1527.3496015770504
4	1220.1873939220834
5	1249.5317323284835
6	1420.9237021788183
7	2117.96124227395
8	2593.14258374712
9	1840.9091589453676
10	2542.191660133464
11	1800.5059152769024
12	1783.7399068111924

Figure 6.4 - Result

7. ADVANTAGES AND DISADVANTAGES

Advantages of Recurrent Neural Network

- An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.

- Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

- Possibility of processing input of any length.

- Model size not increasing with size of input.

- Computation takes into account historical information.

- Weights are shared across time.

Disadvantages of Recurrent Neural Network

- Gradient vanishing and exploding problems.

- Training an RNN is a very difficult task.

- It cannot process very long sequences if using tanh or relu as an activation function.

- Computation being slow.

- Difficulty of accessing information from a long time ago.

- Cannot consider future input for the current state.

8. APPLICATIONS

RNNs are widely used in the following domains/ applications:

- Prediction problems
- Language Modelling and Generating Text
- Machine Translation
- Speech Recognition
- Generating Image Descriptions
- Video Tagging
- Text Summarization
- Call Center Analysis
- Face detection, OCR Applications as Image Recognition
- Other applications like Music composition

Text Generation

Generating text with recurrent neural networks is probably the most straightforward way of applying RNN in the context of the business operation .Text generation is valuable as a means for streamlining the workflow and minimizing the routine.Natural language generation relies on Recurrent Neural Networks predictive algorithms. Since the language is sequentially organized with grammar and bound into cohesion with semantics - it is relatively easy to train a model to produce generic text documents for multiple purposes.

Text Summarization

Text summarization is the process involves condensing the original text into a distillation of critical points and its subsequent reiteration into a cohesive summary. Summarization is used in project management to quickly onboard new members and keep an eye on the progress in general. This approach is also used to create news digests and streamline news article production pipeline.

Report Generation

Report Generation in this case, text generation serves as a form of data visualization. Except, instead of turning data into bars and charts and graphs, the text is transformed into a formatted document with template sentences covering key points. Here's an example of this kind of report: "There were 100 visitors on site during 24 hour period, which is two visitors more compared with the previous 24 hour period. Twenty-five visitors came from Facebook, 10 of which bounced off instantly, while the other 15 made from 5 to 20 clicks on the following page".

Speech-to-text application

Sound is another medium where content marketing can thrive. Due to a variety of reasons, not every user has time to read a blog post from start to finish, but they are likely to listen to it. However, recording read-outs with voice actors can be a bit too much on the budget. Hopefully, modern speech-to-text applications are capable of doing a serviceable and cost-effective job without calling much attention to its mechanistic nature. Such claims have sample banks with phonetic segments performed in different languages that are arranged in the form of the input text. Blogging platforms like Medium are currently trying out these features, and many separate services provide speech-to-text transformations, such as SpeechNote and VoiceNotebook.

Video tagging

RNNs can be used for video search where we can do image description of a video divided into numerous frames.

Face detection, OCR Applications as Image Recognition

Image recognition is one of the major applications of computer vision. It is also one of the most accessible form of RNN to explain.In its core, the algorithm is designed to consider one unit of image as input and produce the description of the image in the form of multiple groups of output .

The image recognition framework includes:

1. Convolutional neural network that processes the image and recognizes the features of the pictures,
2. Recurrent neural networks that makes use of the known features to make sense of the image and put together a proper description of the input image.

Predictive Analytics

In a way, recurrent neural network stock prediction is one of the purest representations of RNN applications. It is all tweaking numbers to understand what the next figure might be.The critical term is time series prediction, which is a representation of the number figure fluctuation or transformation over time. Apps like Stock Market Sensei use this approach.The transformation includes a specific criterion that affected the changes (for example, the connection of the special price to the other expenses). The combination of the elements above is then taken into consideration upon calculation of the predictions.The predictions itself range by probability from the most to the least possible from the available data. As a result, the stock market trader gets more solid grounds for decision making and reduces the majority of risks.

9. CONCLUSION

Recurrent Neural Networks stand at the foundation of the modern-day marvels of artificial intelligence. They provide solid foundations for artificial intelligence applications to be more efficient, flexible in its accessibility and most importantly, more convenient to use.On the other hand, the results of recurrent neural network work show the real value of the information in this day and age. They show how many things can be extracted out of data and what this data can create in return. And this is incredibly inspiring.

In this study, we showed that wind energy output can be predicted from publicly available weather data with accuracy up to 80% R2 on the training range and up to 85, 5% on the unseen test data. We identified the smallest space of input variables where reported accuracy can be achieved, and provided clear trade-offs in prediction accuracy . We demonstrated that an off-the-shelf data modeling and variable selection tool can be used with mostly default settings to run the symbolic regression experiments as well as variable importance, variable contribution analysis, ensemble selection, and validation.

We are pleased that the presented framework is so simple that it can be used by literally everybody for predicting wind energy production on a smaller scale—for individual wind turbines on private farms or urban buildings, or for small wind farms. For future work, we are planning further study of the possibilities for longer-term wind energy forecasting.

10.FUTURE SCOPE

Wind energy is available without any cost and it does not emit any greenhouse gases. This makes it a great source of energy production for any developing state. The field of wind energy has tremendous scope for innovation, translating to real world applications and tremendous economic opportunity.

11.BIBILOGRAPHY

Wind Energy in the U.S.: A State By State Survey. (current). Washington, DC: American Wind Energy Association; 183 pp.

Spera, D.A. (May 1994). Wind Turbine Technology: Fundamental Concepts of Wind Turbine Engineering. 100368. Fairfield, NJ: American Society of Mechanical Engineers; 700 pp.

American Wind Energy Association. (1994). American Wind Energy Association's 1994 Membership Directory. Washington, DC: American Wind Energy Association; 42 pp.

Interstate Renewable Energy Council; Solar Energy Industries Association; Sandia National Laboratories. (1993). Procurement Guide for Renewable Energy Systems; 140 pp. Available from American Solar Energy Society, 2400 Central Avenue, G-1, Boulder, CO 80301.

Gipe, P. (1993). Wind Power for Home & Business. Post Mills, VT: Chelsea Green Publishing Company; 413 pp.

Recommended Practice for the Installation of Wind Energy Conversion Systems. (1989). A WEA Standard: AWEA 6.1-1989. Washington, DC: American Wind Energy Association; 48 pp.

Safety of Wind Turbine Generator Systems. (1994). Draft International Standard: TC-88. Geneva, Switzerland: International Electrotechnical Commission.

Wind Turbines: Peiformance Test Codes. (1989). ASMEJANSI PRC 42-1988. New York, NY:American Society of Mechanical Engineers; 61 pp.

Standard Peiformance Testing of Wind Energy Conversion Systems. (1988). AWEA Standard: A WEA 1.1-1988. Arlington, VA: American Wind Energy Association; 32 pp.

Recommended Practices for Wind Turbine Testing: 4. Acoustics. Measurement of Noise Emission from Wind Energy Conversion Systems (WECS); 2. Edition 1988. (1988). Edited by S. Ljunggren, and A. Gustafsson; 23 pp. Submitted to the Executive Committee of the International Energy Agency Program for Research and Development on Wind Energy Conversion Systems.

APPENDIX

Source code:

```
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
import pandas as pd
from datetime import datetime
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import numpy as np
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

```
from pandas import Series
import glob
from datetime import datetime
%matplotlib inline

batch_size_exp = 1
epoch_exp = 7
neurons_exp = 10
predict_values_exp = 12
lag_exp=24

def timeseries_to_supervised(data, lag=1):
    df = DataFrame(data)
    columns = [df.shift(i) for i in range(1, lag+1)]
    columns.append(df)
    df = concat(columns, axis=1)
    df.fillna(0, inplace=True)
    return df
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return Series(diff)
 def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back):
        a = dataset[i:(i + look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    print(len(dataY))
    return np.array(dataX), np.array(dataY)

def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]

def scale(train, test):
```

```
    # fit scaler
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaler = scaler.fit(train)
    # transform train
    train = train.reshape(train.shape[0], train.shape[1])
    train_scaled = scaler.transform(train)
    # transform test
    test = test.reshape(test.shape[0], test.shape[1])
    test_scaled = scaler.transform(test)
    return scaler, train_scaled, test_scaled

def invert_scale(scaler, X, value):
    new_row = [x for x in X] + [value]
    array = np.array(new_row)
    array = array.reshape(1, len(array))
    inverted = scaler.inverse_transform(array)
    return inverted[0, -1]

def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
        model.add(LSTM(neurons,   batch_input_shape=(batch_size,   X.shape[1],
X.shape[2]),      stateful=True))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=batch_size, verbose=1, shuffle=False)
        model.reset_states()
    return model

def forecast_lstm(model, batch_size, X):
    X = X.reshape(1, 1, len(X))
    yhat = model.predict(X, batch_size=1)
    return yhat[0,0]

series = pd.read_csv('T1.csv',index_col="Date/Time")
```

```
series.head()

del series['LV ActivePower (kW)']
del series['Wind Speed (m/s)']
del series['Wind Direction (°)']
series.head()


for i in range(0,12):
  series = series[:-1]
series.tail()

raw_values = series.values
diff_values = difference(raw_values, 1)

from pandas import concat
from pandas import datetime
from pandas import DataFrame

supervised = timeseries_to_supervised(diff_values, lag_exp)
supervised_values = supervised.values

train,test=supervised_values[0:-predict_values_exp],
supervised_values[-predict_values_exp:]

scaler, train_scaled, test_scaled = scale(train, test)

lstm_model = fit_lstm(train_scaled, batch_size_exp, epoch_exp, neurons_exp)

predictions = list()
expectations = list()
test_pred = list()
for i in range(len(test_scaled)):
    X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
    yhat = forecast_lstm(lstm_model, 1, X)
    test_pred = [yhat] + test_pred
    if i+1<len(test_scaled):
```

```
        test_scaled[i+1] = np.concatenate((test_pred, test_scaled[i+1, i+1:]),axis=0)
    yhat = invert_scale(scaler, X, yhat)
    yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
    predictions.append(yhat)
    expected = raw_values[len(train) + i + 1]
    expectations.append(expected)
    print('Hour=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))

expectations = np.array(expectations)
predictions = np.array(predictions)
print("Mean Absolute Percent Error: ",(np.mean(np.abs((expectations - predictions) / expectations))*100))

pyplot.plot(raw_values[-predict_values_exp:], label="True")
pyplot.plot(predictions, label="Predicted")
pyplot.legend(loc='upper right')
pyplot.xlabel("Number of hours")
pyplot.ylabel("Power generated by system (kW)")
pyplot.show()
```