PROJECT REPORT

Team Name : Wildcard Coders

Link to the GitHub Repository:
https://github.com/sreyans01/Power_Prediction_in_Wind_Farms

APPENDIX

Introduction

Overview

The core of civilization lies in the production and distribution of energy. From industry to household, cost-effective energy availability plays a significant role in the development of society. The cost-effectiveness could be achieved only if we deal with some of the significant issues in the production and distribution of energy in the form of electricity. As we enter a phase where we are shifting from non-renewable energy sources to renewable energy sources, it becomes essential for the energy distributor to extract optimally and in the best possible way for establishing cheap production as well as minimum production so that it suffices to the need of the users.

One of the primary renewable energy sources is Wind Energy. Wind energy is a form of solar energy. Wind energy (or wind power) is the process by which wind is used to generate electricity. Wind turbines convert the kinetic energy in the wind into mechanical power. A generator can convert mechanical strength into electricity.

The electricity generated from the various sources of electricity reaches a shared power grid wherein the distribution and transmission are decided according to the need and availability of the energy from multiple sources. The wind being an unreliable source, the production of electricity from it could lead to two major problems:

1. The shortage of energy in the power grid due to relying on future weather conditions.
2. The costly overproduction of the wind energy if necessary, planning of energy availability isn’t done.

This factors not only increase the cost of energy production but also increases uncertainty among the distributors in the power grid. Knowing the availability of electricity is important to suffice the excess need of industries or households along with managing overproduction.

Overview of the Problem

Rapid Growth in Wind Generation of Electricity

Over the last decade, there has an enormous increase in the production of electrical energy through wind farms, which also leads to huge wastage of energy because of variable nature of wind.

Brainstorm ideas

As wind is inherently variable, wind power is a fluctuating source of electrical energy.Short term forecasts(ranging from 1 h upto 72h) are useful in power system planning for unit commitment and dispatch, and for electricity trading in certain electricity markets where wind power and storage can be traded or hedged.

Building a deep learning model

The problem can be solved by building a deep learning model which can predict the next 72 hour Power Outputs based on the weather forecast.

Creating an Android Application

A standalone app will give the end user the Power output data, and users can see a complete graph of Power Output vs Time to get more insights of the data.

Testing and Deploying the Model

The deep learning model, which was built to give the predictions, will then be tested through the application for different locations and values.

Making the application ready for use

Once, the deep learning model is well tested for different values, our application is now ready for use.

Purpose

To tackle the problem such as uncertainty in Active Wind Power extraction or costly overproduction, an application needs to be built for accurate prediction of the active wind power output from the wind farm. So, in order to create sustainable energy system, our world needs new technology to step into, to ensure more efficient and reliable production of wind energy.

According to many reports also, it is estimated that wind energy will constitute to more than 40% of the total renewable energy production of the world in about 10-20 years. But due to insufficient technology, and frameworks, there occurs a huge wastage in wind energy generation due to the variable nature of wind.

To counter this huge wastage of resources, we can actually bring digital technologies that can effectively make the energy production easy and profitable. The Mobile Applications are easiest to do so, as they have turned out to be used by most of the people nowadays. We also aim at reducing the chances of overproduction of energy because this ultimately results in a problem for the power grid managers. We aim at building a mobile application that can effectively be used by the power grid to tackle all there uncertainty towards Wind Turbine and its energy production.

Existing Problem

Wind energy is a key player in the field of renewable energy. 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. Energy suppliers are interested in accurate predictions, as they can avoid overproduction by coordinating the collaborative production of traditional power plants and weather-dependent energy sources.

Countries like India which is still majorly dependent on non-renewable energy has the big problem of managing the energy demand of the consumers without the reliability on the energy source such as wind energy. Most of power grids are unknown about the capacity of the wind production of the wind farm and certainly face problem such as under-production which not only increases the dependence on renewable energy sources but also increases the cost of production as lost of free wind energy are being not utilized.

A similar problem occurs with the over-production of energy if many energy sources (both renewable and non-renewable sources) are being utilized for energy production due to lack of wind power forecast. This is a major challenge as installation of wind farms also turns out to be costly and it would be really hard to produce energy without proper forecast as it wouldn't be economical for production. These are the major challenges we are aiming to solve.

Proposed Solution

The generation of electric power from the wind relies on atmospheric processes. The power output of a single wind turbine is a direct function of the strength of the wind over the rotor swept area. Coarsely simplifying the meteorological aspects involved, winds originate from the movement of air masses from high to low pressure areas: the larger the difference in pressure, the stronger the resulting winds. On top of that come the boundary layer effects, complexing wind behavior due to natural obstacles, friction effects, the nature of the surface itself, temperature gradients, etc. The boundary layer is formally defined as the lower part of the atmosphere where wind speed is affected by the surface. From experimental studies, it is clear there is a clear relation of the wind power with the weather at the farm. This relation goes as follows:

$$P = \frac{1}{2}\rho A V^3$$

As the problem demands forecasting of the Active Power Output. We build a Deep Learning model on Keras framework to predict the active power based on the wind condition prevalent at wind farm site. Artificial Neural Networks do considerably great job in finding in complex relation between parameters and output. The Deep Learning model is then deployed on the IBM Cloud to provide the access to the application using API calls.

For the end user, we create an Android application for the forecast of Active Power Output for the next 72 hours. This application will be continuously collecting weather data such as wind speed, wind direction, and other useful information according to the detected user's location or the location entered by the user using the Application Programming Interface (API). This incorporates a direct link of the android application and the web server of OpenWeatherMap.org, and this data will be continuously synced with the IBM Cloud base server and then back to the ML model for predicting the output. The data returned by the ML model will be fetched and updated back into the app to the end user. To inform the user about the best time slot to run the wind turbine we also create a functionality which integrates different time slot and returns an timeslot with maximum benefit of running the wind turbine.

Block Diagram

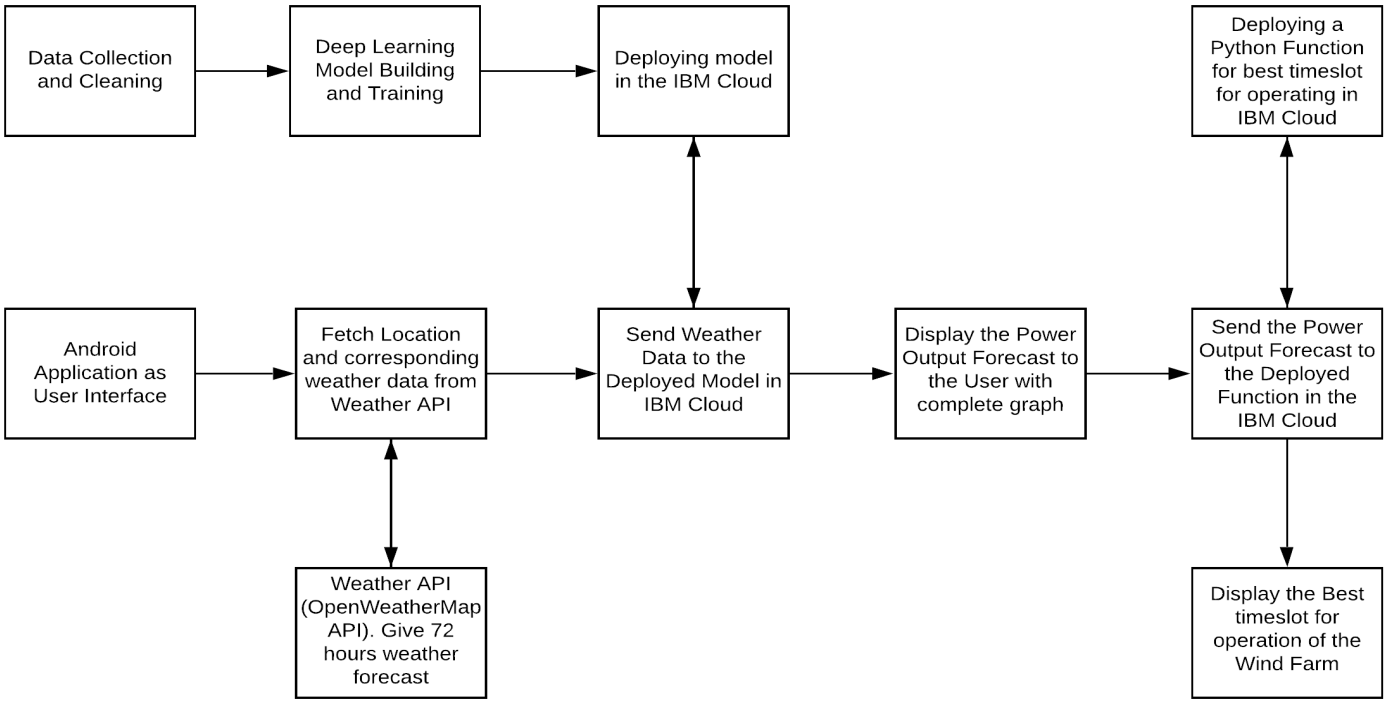


Fig 1: The figure depicts the block diagram of the project-work

Software Designing

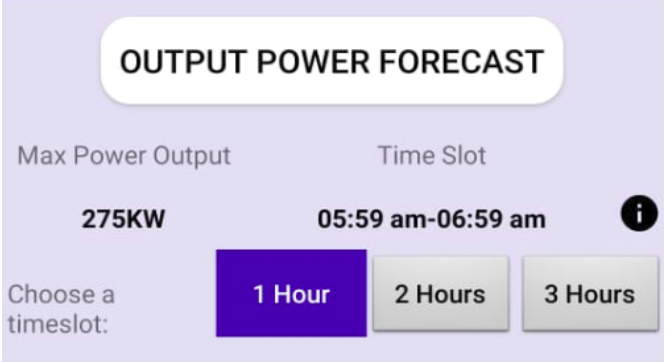
User Story:

As a customer, I need to be able to:
i. Get my weather conditions such as wind speed and wind direction
ii. Change the location for the accurate prediction
iii. Get complete details of the next 72 hrs power output
iv. View a graph of Power Output vs Time
v. View the best time to run the wind farm for the maximum power generation
vi. Chose my own time slot for the running of wind farm
vii. Chose any time duration between 5h to 72h

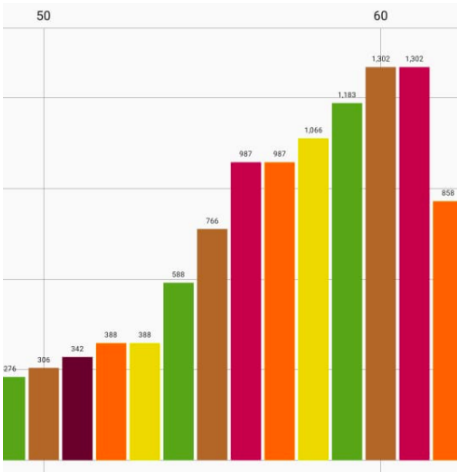
Features:

The major features of the Application must include:
i. A weather forecast card which informs about factors such as weather condition, wind speed and direction which is important for operating the wind farm.

ii. A power output forecast card which displays the timeslot and the time with the maximum power output.



iii. The graphical representation of the forecast.



Requirements/Overview:

1. The one of the most important application is to show its users the prediction of the Power output in the wind farm of a particular location for the next 72 hours with a proper representation of the data in a graphical format which will enhance the user experience.

2. An function needs to built which will integrate a line curve of the prediction produced by the model over a length of timeslot chosen by the user.

3. This function returns the particular time where the power is maximum (maximum integrated value).

4. To cite out an example for a timeslot of 2 hrs, the function outputs a time which produces maximum power for the next 72 hrs.

5. A deep learning model needs to be trained on a SCADA dataset to predict the active power forecast.

6. With proper hyper tuning and regularization the model needs to be trained for accurate result.
7. A Machine Learning service needs to be created so that an API could be deployed which will hold the model and Function.
Watson Studio provides platform to run, train and deploy model on the Watson Notebooks.

Experimental Investigation

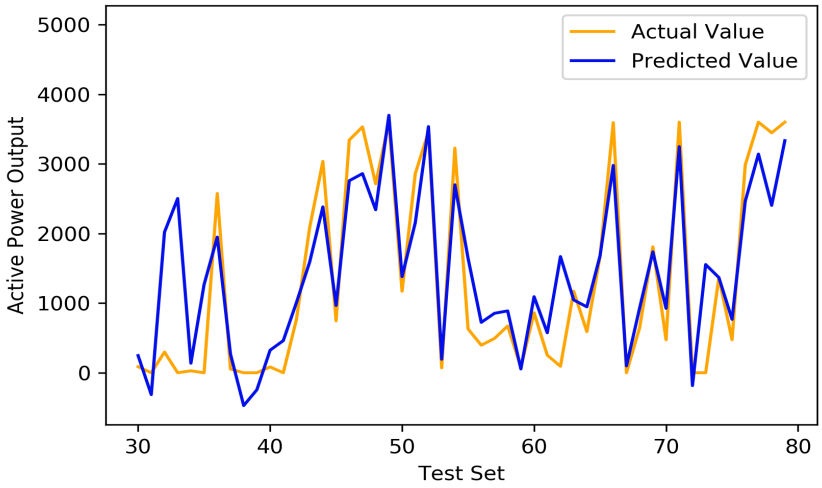
The core of the application lies in the prediction of the Active Power Output from the given weather data. This includes building a Machine Learning model for the prediction. A Neural Network based model has been used here to predict the Active Power Output of the Wind Farm. Neural Networks are the algorithms or architecture that have multiple layers with each layer getting fed into the other to form a chain of layers which finally processes and find relations from the data. Artificial Neural Networks are capable of building complex relationships between the input and the output features.

For the training the model, a public dataset is chosen available at Kaggle. The details of the Dataset are as follows:
The dataset has been collected from a SCADA system installed in the wind turbine at turkey. The exact location of wind turbine is in Turkey, Yalova. The coordinates are X:668478 Y:4494833 UTM ED 50, 6 degree.
The dataset has recorded the Active Power Output for every 10 mins of interval for a span of 1 year. It includes features such as wind speed, wind direction and the Theoretically Expected Power Output. The latter is from a ideal theoretical curve which has considerable errors as it only takes wind speed into consideration.

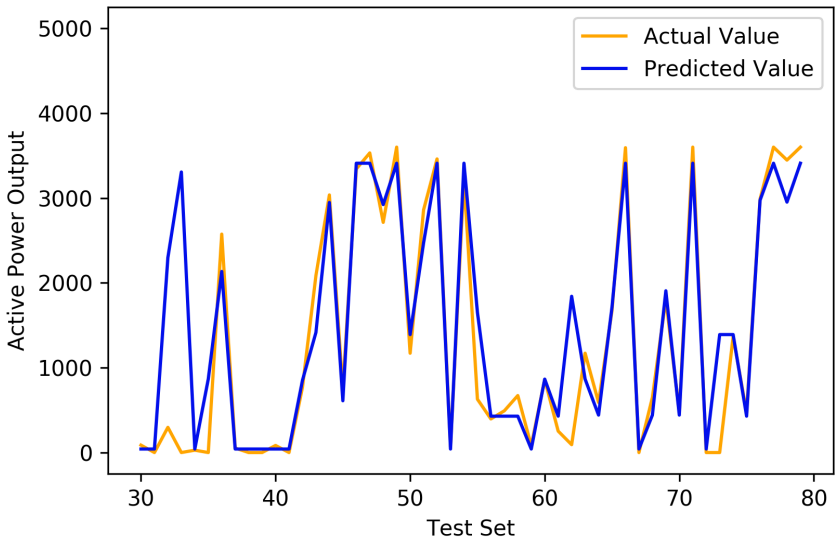
Standard Machine Learning Algorithms vs Neural Networks

A number of regression techniques has been used for the prediction and the results of the same are as follows:

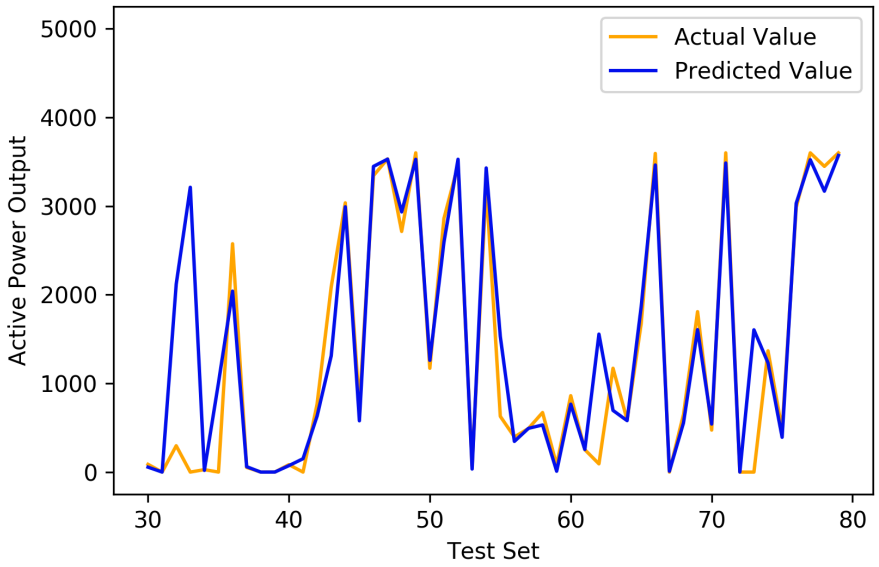
Multivariate Linear Regression:
The input features included the wind speed and wind direction(normalized).
Test Set Mean Absolute Error: 388.6834758143794
This the real value vs predicted value in the test set:



Random Forest Regressor :
(Maximum Depth of the Tree had been tuned to remove over fitting)
Test Set Mean Absolute Error: 192.75887749214806
This the real value vs predicted value in the test set:

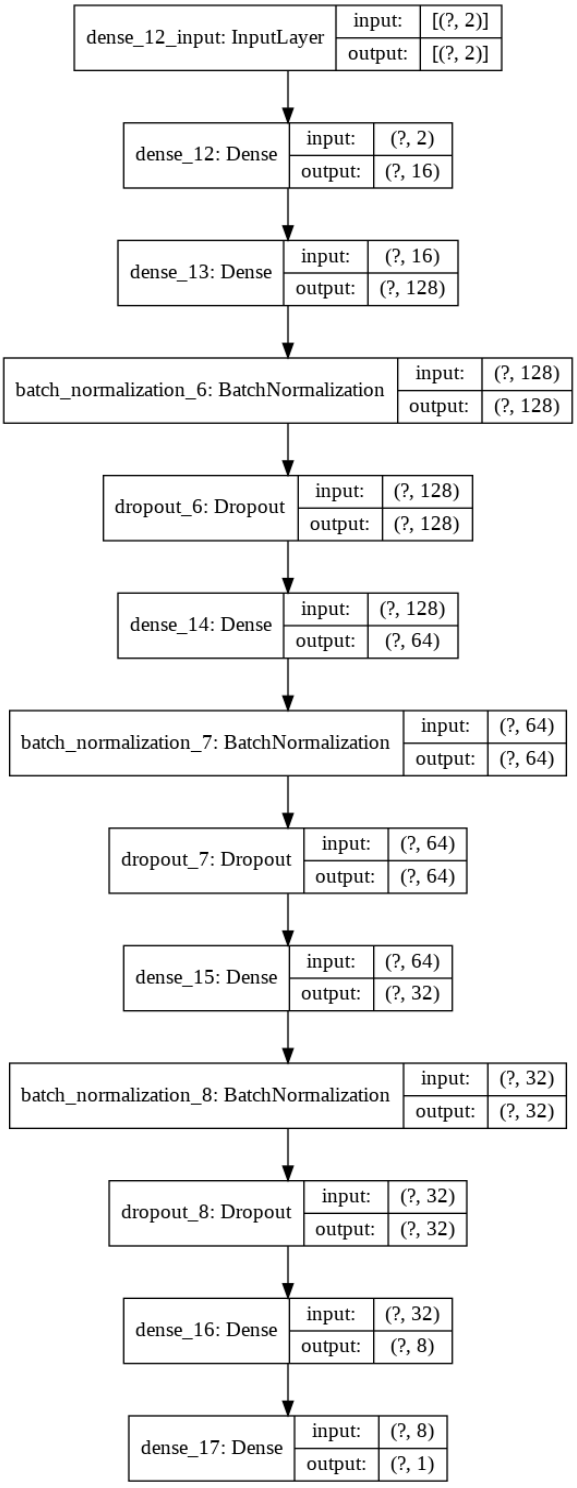


Artificial Neural Network:
(Simple Architecture without Regularization)
Test Set Mean Absolute Error: 185.13300537517875
This the real value vs predicted value in the test set:



The above architecture was a simple one with 5 layers and without any regularization , hence it

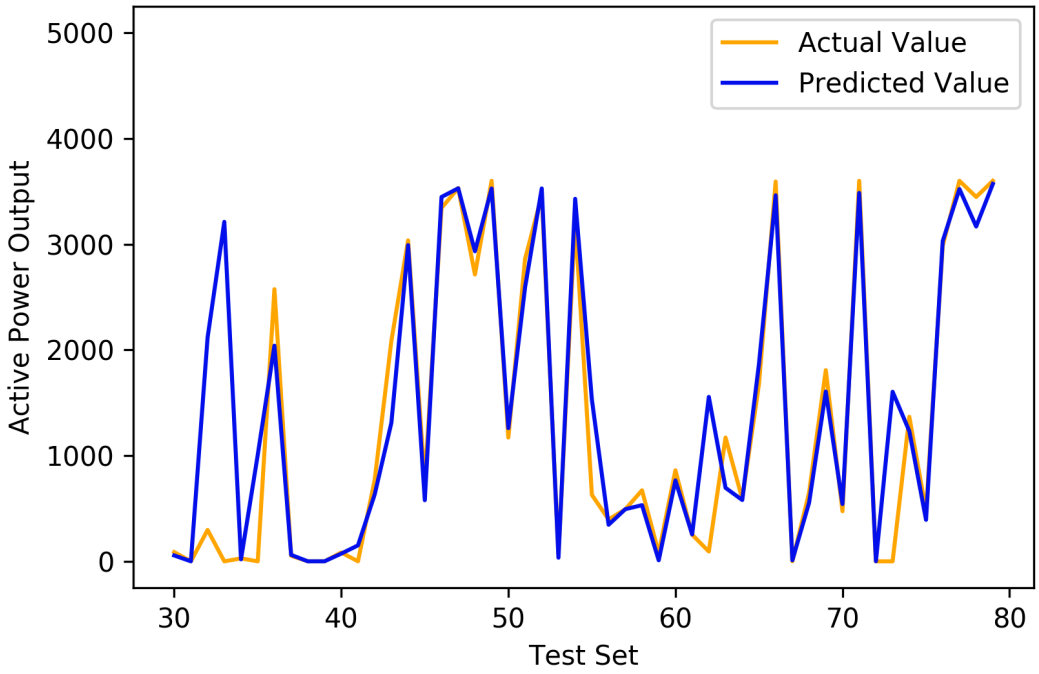
overfitted. But the simple architecture provided a considerable improvement in mean absolute error on test set. Hence using ANN would be a better option rather going for simple ML regression models.
Model overfitting is generally a problem when there is no regularization in the model. The model performs well on the train set but it fails to generalize well to the new data.
We experimented on the model with adding Dropouts (A method of regularization) and Batch Normalization. The model summary is as follows:
The block diagram represents the final model architecture:



The above model is modified with extra layers such as Batch Normalization Layer and Dropout for better test accuracy.

Artificial Neural Network (With Batch Normalization and Dropouts) :

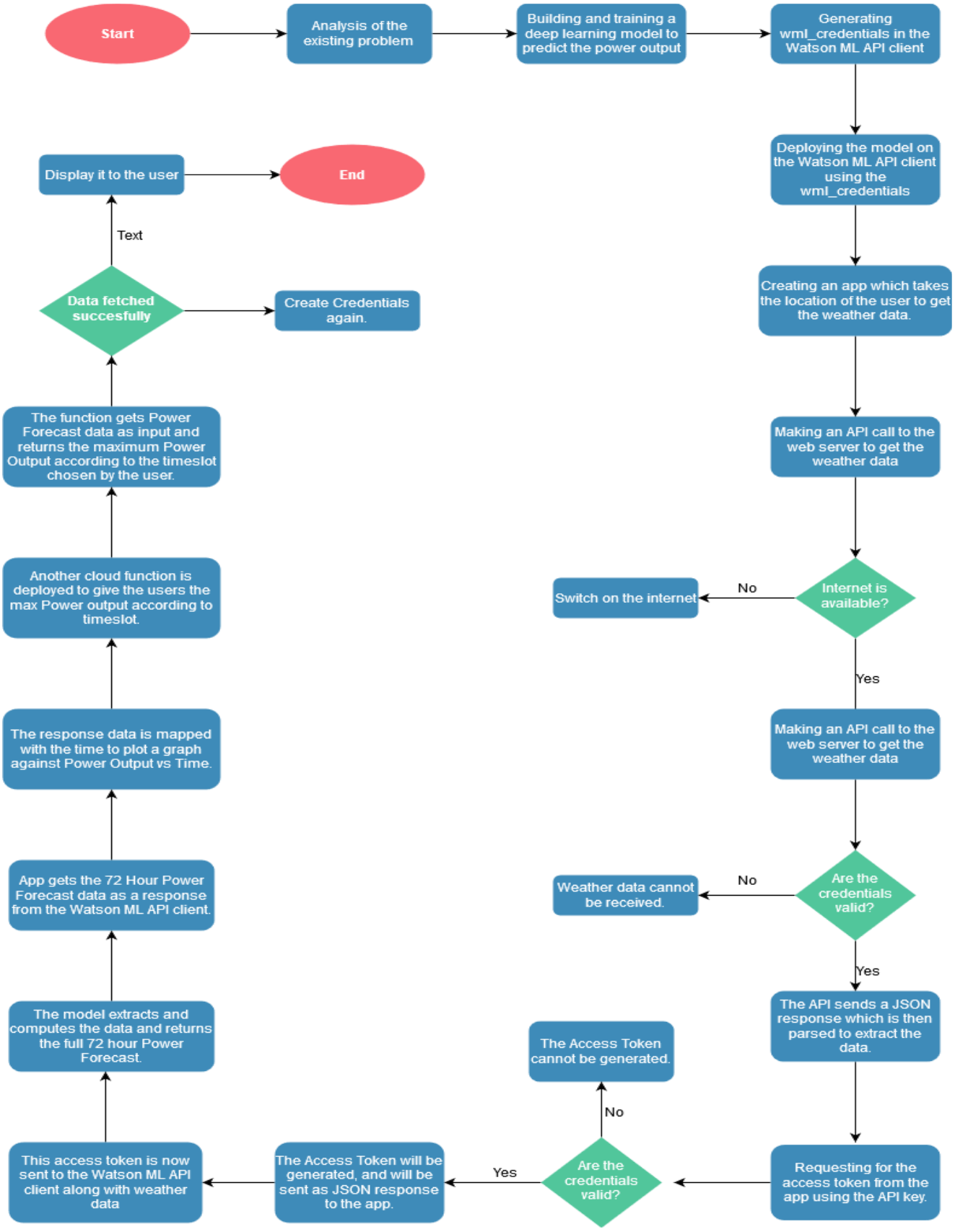
Test Set Mean Absolute Error: 160.1541670167519
This the real value vs predicted value in the test set:



This is the final performance on test set. It performs considerably better than the other models hence we would go for this model for deployments.

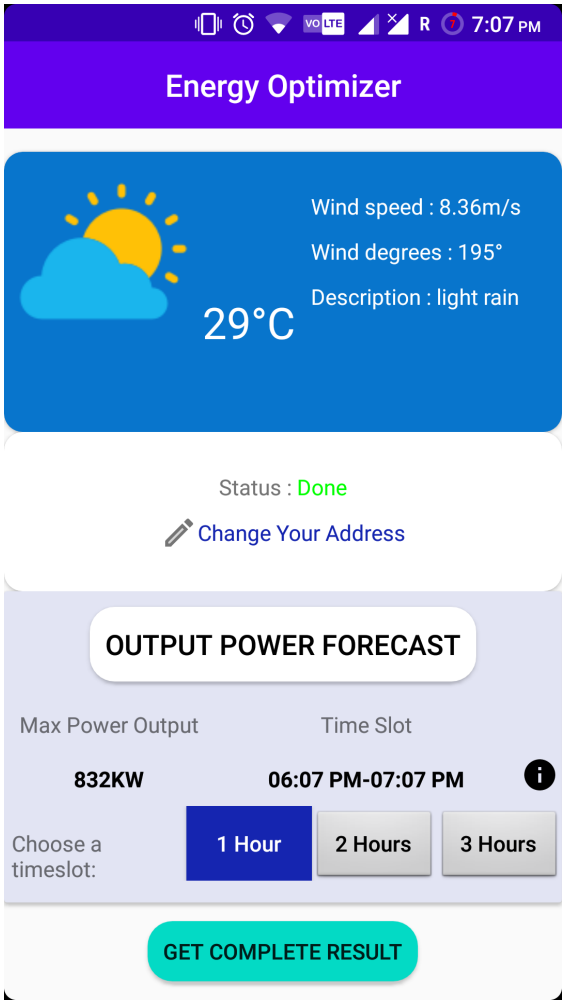
Hence after experimentally trying out various model we arrive with a optimum model with good test accuracy.

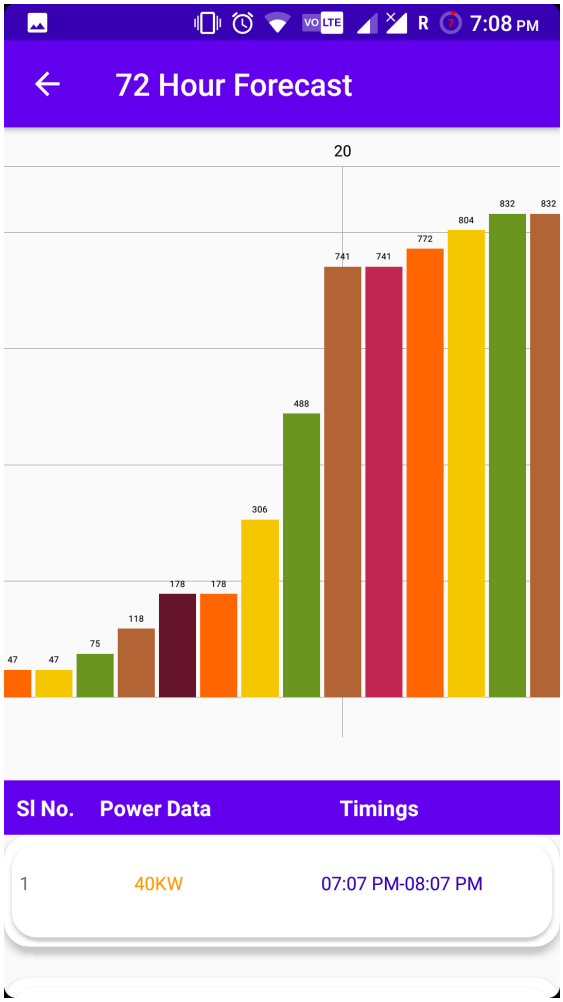
Flowchart

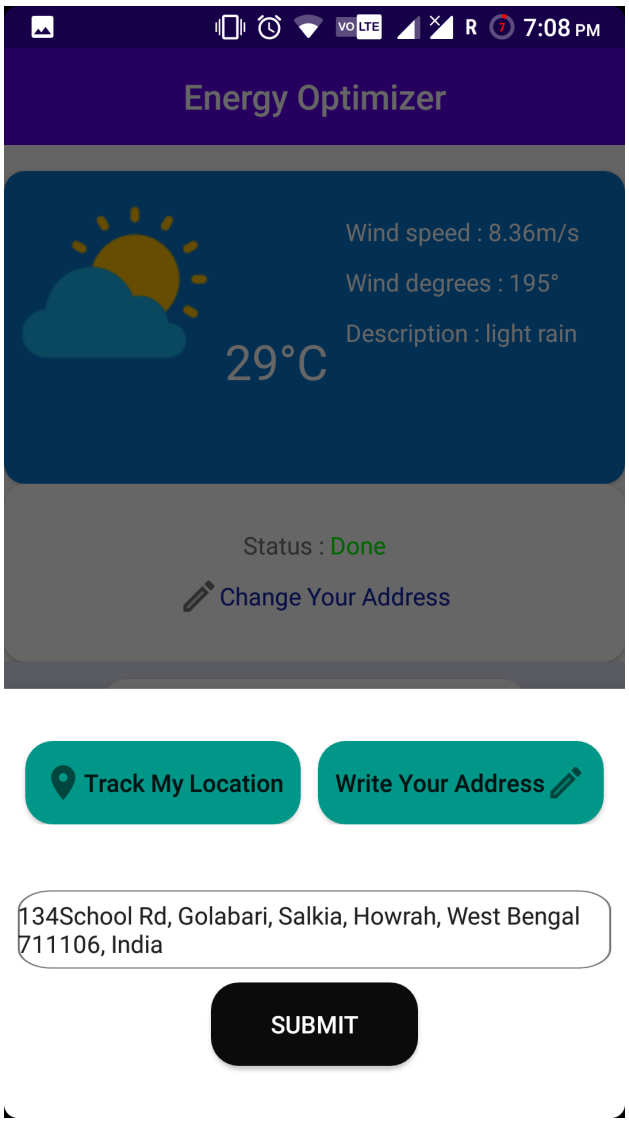

Start
Analysis of the existing problem
Building and training a deep learning model to predict the power output
Generating wml_credentials in the Watson ML API client
Deploying the model on the Watson ML API client using the wml_credentials
Creating an app which takes the location of the user to get the weather data.
Making an API call to the web server to get the weather data
Internet is available?
No
Switch on the internet
Yes
Making an API call to the web server to get the weather data
Are the credentials valid?
No
Weather data cannot be received.
Yes
The API sends a JSON response which is then parsed to extract the data.
Requesting for the access token from the app using the API key.
Are the credentials valid?
No
The Access Token cannot be generated.
Yes
The Access Token will be generated, and will be sent as JSON response to the app.
This access token is now sent to the Watson ML API client along with weather data
The model extracts and computes the data and returns the full 72 hour Power Forecast.
App gets the 72 Hour Power Forecast data as a response from the Watson ML API client.
The response data is mapped with the time to plot a graph against Power Output vs Time.
Another cloud function is deployed to give the users the max Power output according to timeslot.
The function gets Power Forecast data as input and returns the maximum Power Output according to the timeslot chosen by the user.
Data fetched succesfully
Create Credentials again.
Text
Display it to the user
End

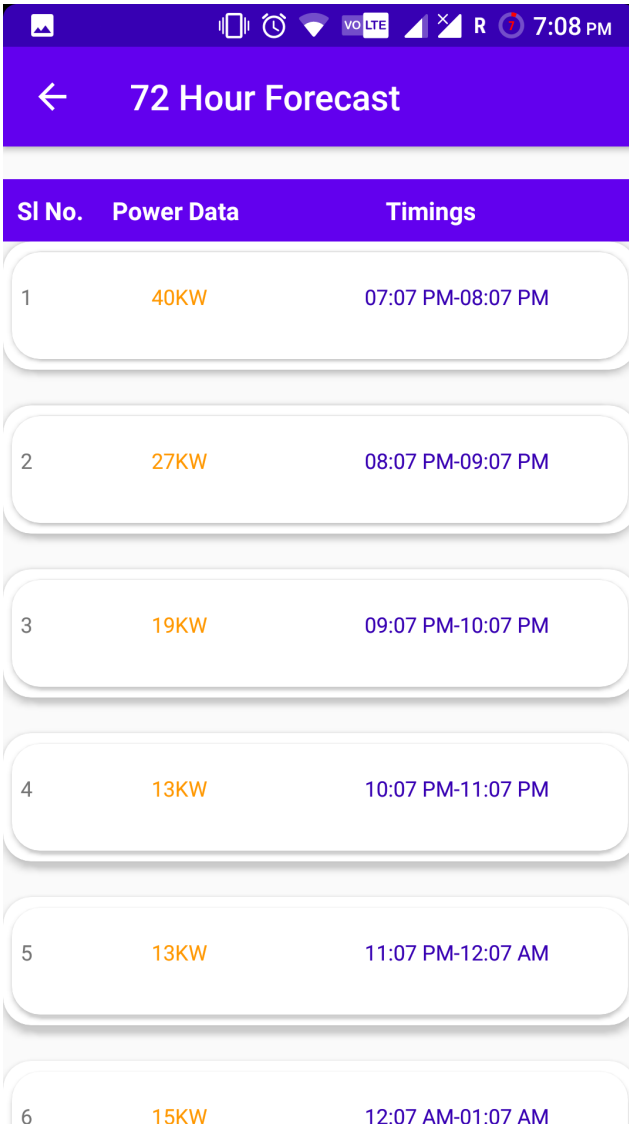
Result

The whole project was built using the IBM Cloud Services, more specifically IBM Watson Studio Services. The building of model included various stages like data analysis, data visualization etc. which were all made easy by the help of IBM Watson Studio Notebook. More specifically the training and the hyperparameter tuning included various iteration of training. But the IBM Watson Studio Notebook provided good processing power to train the model. We also created an instance of the Machine Learning Service which would help us to deploy the trained model. We used the Watson-Machine-Learning API Client to deploy the model into the ML service created in the cloud. Another Function was also deployed to find the maximum power output for a specific time slot using integration under the line curve. This function was completely written on Python and was deployed in the cloud in a similar manner as we deployed the deep learning model in the cloud.
Finally from the Android Application side we used HTTP request to access the model and Function in the cloud.
After lot of optimization this is the Final interface of the application:






7:08 PM
Energy Optimizer
Wind speed : 8.36m/s
Wind degrees : 195°
Description : light rain
29°C
Status : Done
Change Your Address
Track My Location
Write Your Address
134School Rd, Golabari, Salkia, Howrah, West Bengal 711106, India
SUBMIT


7:08 PM
72 Hour Forecast
Sl No.
Power Data
Timings
1
40KW
07:07 PM-08:07 PM
2
27KW
08:07 PM-09:07 PM
3
19KW
09:07 PM-10:07 PM
4
13KW
10:07 PM-11:07 PM
5
13KW
11:07 PM-12:07 AM
6
15KW
12:07 AM-01:07 AM

Advantages and Disadvantages

Advantages:

1. The regular usage of application can help the power grid controller to manage the wind mill more efficiently and more productively.
2. The application provides all the necessary weather data and the forecast of Active Power Output which can help user to visualize the production analysis for the next three days.
3. The application also provides the best time to operate the wind farm which would immensely help user as he is not needed to make calculation regarding the operation.
4. In long term this will lead to less dependency on non-renewable resources which can be extracted as per needs but more confidence on renewable energy with the forecast which was unreliable earlier.

Disadvantages:

1. The mobile application takes weather data from the Weather API which may or may not be fully accurate.
2. There might be slight variations in the prediction results shown in the app, and the actual results.
3. There are several server calls by the mobile application for the prediction of the results, and any loss in internet connection may result in restarting the app.
4. As the complete application is built on the concept that all the connections and transmission are ideal which is not always the case. There would be an extra loss of power due to physical connections and efficiency of conversion of mechanical to electrical energy.

Applications

- Accurate wind forecast is essential for integration of wind farms to power systems
- The impact of the wind uncertainty on the operation of gas plants was investigated.
- The android application can affect the wind energy sector significantly. This can help all the power grid user to decide the operation frequency as well the exact time for operation.
- Suppose the power grid user is planning for a collaborative extraction of wind energy with other sources which are more likely be extracted with proper planning, to achieve maximum efficiency they can use the application.
- The cost of overproduction can be reduced by accurate planning based on the forecast.
- On long term usage farms of the application the power grid controller could also help the wind farm manger to find faults and disconnections if expected power output is not extracted as per the prediction. In future it may help the user to find ways to decreases all the physical losses during transmission as he would have a perfect goal to reach every time he improves the system.
- Users can also get the weather forecast in the application which would ultimately help the user for the future risk of operation during natural calamities like thunderstorms and lightning.
- The app is designed to provide one of the best user experience with less complexity and more information. It can also be used by users who were earlier not comfortable with digital interface.

Conclusion

The application **Energy Optimizer** will not only ease the process for the power grid operators but also turns out to be a changing point for the people to see Wind Farm as reliable or more predictable source of energy than before. This would not only reduce dependency on non-renewable energy but also help the wind farm market to grow profitably.
The application was completely built with the help of IBM cloud services which not only provided a platform to run the model but also test it using some UI options like the node-red. This actually helped us to experiment and iterate to enhance the features and its user experience. We plan to improve on more features as we take feedback from the users

Future Work

In an era of accelerating change, the imperative to limit climate change and achieve sustainable growth is strengthening the momentum of the global energy transformation. The rapid decline in renewable energy costs, improving energy efficiency, widespread electrification, increasingly “smart” technologies, continual technological breakthroughs and well-informed policy making all drive this shift, bringing a sustainable energy future within reach.

While the transformation is gaining momentum, it must happen faster. Around two-thirds of global greenhouse gas emissions stem from energy production and use, which are at the core of efforts to combat climate change. To meet climate goals, progress in the power sector needs to accelerate further, while the decarbonisation of transport and heating must pick up steam.

The energy system, consequently, requires rapid, immediate and sustained change. The deployment of renewable must increase at least six-fold compared to the levels set out in current plans. The share of electricity in total energy use must double, with substantial electrification of transport and heat. Renewable would then make up two-thirds of energy consumption and 85% of power generation. Together with energy efficiency, this could deliver over 90% of the climate mitigation needed to maintain a 2°C limit.

The world’s actions today will be crucial to create a sustainable energy system. Ultimately, the path to secure a better future depends on pursuing a positive, inclusive, economically, socially and environmentally beneficial energy transformation.

Subsequently we plan to build a time series model from the data so that it can predict the future values based on the values entered by the user for the last 24 hours. This would not only remove all the affect of the losses due to the external sources but also could improve over time with more and more user data. We also plan to remove the location specificity and the dependency of accuracy on the location as it would make the application more versatile and robust to external factors. We also plan to build an anomaly or fault detection if sufficient continuous information is provided by the user. We even plan to work on the flexibility of time slot with limited effect on user interface.

Bibliography

1. Wind Turbine SCADA Dataset
https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset

2. Predicting the Energy Output of Wind Farms Based on Weather Data: Important Variables and their Correlation
https://hpi.de/friedrich/docs/paper/RE1.pdf

3. OpenEI Wind Energy
https://openei.org/wiki/Wind_energy

4. Wind Energy: Forecasting Challenges for Its Operational Management https://www.researchgate.net/publication/259441184_Wind_Energy_Forecasting_Challenges_for_Its_Operational_Management

5. Optimization of Short-term Overproduction Response of Variable Speed Wind Turbines https://www.researchgate.net/publication/323460742_Optimization_of_Short-term_Overproduction_Response_of_Variable_Speed_Wind_Turbines

6. Open Weather Map
https://openweathermap.org/api

7. Android Design GuideLines
https://developer.android.com/design

8. Material Design Guidelines by google
https://material.io/design/guidelines-overview

Appendix

Link to the GitHub Repository:
https://github.com/sreyans01/Power_Prediction_in_Wind_Farms

Our Video Demonstration consists of two parts -
Part 1 - Overall explanation of the app, purpose,overview of the Problem Statement.

Part 2 - Code explanation which we have written to complete our project.

Part 1 Link -
https://drive.google.com/file/d/1tbmipGPuaXkLn1Yc6XmwGYPAFEH-O-st/view?usp=sharing
Part 2 Link -
https://drive.google.com/file/d/1QkWWSGlQvAWUmxm90t35dwKfg8l1USwr/view?usp=sharing

SOURCE CODE

Python Function for Maximum Power Output

Importing Libraries:

```
import numpy as np
import matplotlib.pyplot as plt
import random
```

Function:

```
def deploy_function():
    def score(payload):
        str123 = payload["values"][0]
        slot = 1
        str_ar = str123.split(' ')
        str_ar1 = []
        for i in range(len(str_ar)):
            str_ar1.append(int(str_ar[i]))
        ar = str_ar1
        total_length = len(ar)
        def float_range(start,stop,step):
            lst = []
            i = start
            while i < stop:
                lst.append(i)
                i = i + step
            return lst

        def calc_area_trap(first_height, second_height, base):
            rect_area =  min(first_height,second_height) * base
            triangle_area = 0.5 *
  (max(first_height,second_height) -
 min(first_height,second_height)) * base
            return rect_area + triangle_area

        def calc_height(prevh,nexth,frac):
```

```
            if nexth >= prevh:
                h = prevh + (nexth-prevh) * frac
            else:
                h = nexth + (prevh - nexth) * (1-frac)
            return abs(h)

        return_str = ' '
        for slot in range(1,4):
            max_energy = -1
            start_time = -1
            for start_point in float_range(0,total_length - 1 - slot, 0.05):
                energy = 0
                for i in range(slot):
                    esp = start_point + i
                    frac = esp - int(esp)
                    energy += calc_area_trap(calc_height(ar[int(esp)],ar[int(esp)+1],frac),ar[int(esp)+1],1-frac)
                    energy += calc_area_trap(ar[int(esp) + 1],calc_height(ar[int(esp) + 1],ar[int(esp) + 2],frac),frac)

                if(energy > max_energy):
                    max_energy = energy
                    start_time = start_point

            start_time1 = int(int(start_time) * 60 + ((start_time - int(start_time)) * 3)/0.05)
            return_str +=  str(max_energy/slot) + ' '+ str(start_time1) + ' '
        return {"Result" : return_str}

    return score
```

Function Deployment Using Watson Cloud Services:

```
wml_credentials = {
  "apikey": "rWBYMMAvXLpT0poX4fMQBdWofu7mWvKRg2zMaAdvjV9z",
```

```
  "iam_apikey_description": "Auto-generated for key
  3f4a5270-1f86-4809-bc4d-da42cd90a0ab",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn":
  "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn":
  "crn:v1:bluemix:public:iam-identity::a/5c1ce8fd5a6e46a49cc3f2514
  4e7c0ab::serviceid:ServiceId-b2a5850b-6647-4a97-8110-4ba3c1ef69e
  0",
  "instance_id": "89def234-5ebd-4907-a37b-e59d988f9286",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
from watson_machine_learning_client import
WatsonMachineLearningAPIClient
client = WatsonMachineLearningAPIClient(wml_credentials)
meta_data = {client.repository.FunctionMetaNames.NAME :
'Maximum_Power_Output'}
function_details =
client.repository.store_function(meta_props=meta_data,
function=deploy_function)
function_id = function_details['metadata']['guid']
function_id
function_deployment_details =
client.deployments.create(artifact_uid=function_id, name=
'Maximum_Power_Output')
```

Deep Learning Model for Active Power Output forecast

Importing Libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from keras import Sequential,Input
from keras.layers import Dense, BatchNormalization, Dropout
from sklearn.metrics import mean_absolute_error
import keras #version 2.2.5
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

Importing the dataset:

```
def __iter__(self): return 0

client_3906e0246d7b4b60ba65f993ab87759c =
ibm_boto3.client(service_name='s3',
 ibm_api_key_id='VHv1ef_wW9NQ8kyFlLjlmRYT_4t5sjbzVL5jYTLL9_k0',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
endpoint_url='https://s3.eu-geo.objectstorage.service.networklay
er.com')
body =
client_3906e0246d7b4b60ba65f993ab87759c.get_object(Bucket='ibmha
ck2020-donotdelete-pr-2luk1rdgnwhitd',Key='T1.csv')['Body']
if not hasattr(body, "__iter__"): body.__iter__=
types.MethodType( __iter__, body )

df = pd.read_csv(body)
df.head()
```

Data Analysis and Visualization:

```
df.drop('Date/Time',axis = 1, inplace=True)
df.hist(bins = 50, figsize=(20,15))
df["Wind Speed (m/s)"].describe()
df.plot(kind = 'scatter', x = 'Wind Speed (m/s)', y = 'LV
ActivePower (kW)',alpha = 0.4)

df[df["Wind Speed (m/s)"] >= 10].plot(kind = 'scatter', x =
'Wind Direction (Â°)', y = 'LV ActivePower (kW)',alpha = 0.5)
df.plot(kind = 'scatter', x = 'Theoretical_Power_Curve (KWh)', y
= 'LV ActivePower (kW)',alpha = 0.4)
df.plot(kind = 'scatter', x = 'Wind Speed (m/s)',
y = 'Theoretical_Power_Curve (KWh)',alpha = 0.4)
df2 = df.drop(axis = 1, labels =['LV ActivePower (kW)','Wind
Direction (Â°)'])
df2.sort_values(by=['Wind Speed (m/s)'],inplace=True)
plt.plot(df2['Wind Speed (m/s)'].iloc[:].values,
df2['Theoretical_Power_Curve (KWh)'].iloc[:].values)
```

Data Preprocessing:

```
X_wind_speed = df2['Wind Speed (m/s)'].iloc[:].values
Y_TPC = df2['Theoretical_Power_Curve (KWh)'].iloc[:].values
#The below line estimates the expected output for the given wind
speed(here 10m/s)
np.interp(10, X_wind_speed, Y_TPC)
X_data = df.drop(axis = 1, labels=['LV ActivePower
(kW)','Theoretical_Power_Curve (KWh)']).iloc[:].values
Y_data = df['LV ActivePower (kW)'].iloc[:].values.reshape(-1,1)
X_train, X_test, Y_train, Y_test = train_test_split(X_data,
Y_data, test_size = 0.2, random_state = 42)

wind_speed_max = max(X_train[:,0])
theoretical_max_output = 3600
active_power_max = max(Y_train[:,0])
degrees = 360
X_train[:,0] = np.clip(X_train[:,0], a_min=0,
a_max=wind_speed_max)/wind_speed_max
X_train[:,1] = X_train[:,1]/degrees
```

```
Y_train = np.clip(Y_train,a_min=0,a_max =
  theoretical_max_output)/theoretical_max_output
X_test[:,0] = np.clip(X_test[:,0], a_min=0,
  a_max=wind_speed_max)/wind_speed_max
X_test[:,1] = X_test[:,1]/degrees
Y_test = np.clip(Y_test,a_min=0,a_max =
  theoretical_max_output)/theoretical_max_output
```

Model Development:

```
def model():
    model = Sequential([
                Dense(16, activation = 'relu', input_shape = (2,)),
                Dense(128,activation = 'relu'),
                BatchNormalization(),
                Dropout(0.2),
                Dense(64,activation = 'relu'),
                BatchNormalization(),
                Dropout(0.2),
                Dense(32,activation = 'relu'),
                BatchNormalization(),
                Dropout(0.2),
                Dense(8,activation = 'relu'),
                Dense(1, activation = 'sigmoid')])
    model.compile(metrics =
  ['mae','mse'],optimizer=keras.optimizers.Adam(lr = 0.001), loss
  = 'mse')
    return model

model = model()
model.summary()
```

Model Training and Hyper tuning:

```
history = model.fit(X_train,Y_train,epochs = 100, validation_data
  = (X_test, Y_test),batch_size=2048)
plt.plot(history.history['mean_absolute_error'])
plt.plot(history.history['val_mean_absolute_error'])
story.history['loss'])
plt.plot(history.history['val_loss'])
```

Model Testing and Model Deployment using Watson Cloud Services:

```
pred = model.predict(X_test) * theoretical_max_output
model_result_path = "keras_model.h5"
model.save(model_result_path)
get_ipython().system("tar -zcvf keras_model.tar.gz
'keras_model.h5'")
from watson_machine_learning_client import
WatsonMachineLearningAPIClient
wml_credential = {
  "apikey": "oBhAZ0BMAHKtMjP1pCksazYUhv27eGVTJvfKlyd_GjXo",
  "iam_apikey_description": "Auto-generated for key
823b37a4-8b52-491b-a2e7-5f6cf8fb6210",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn":
"crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn":
"crn:v1:bluemix:public:iam-identity::a/63753724baeb4b4d9c46990d4
d7706ea::serviceid:ServiceId-57adf678-b945-4f6a-9f72-a4d6a83b62b
4",
  "instance_id": "e8b2e1c3-9068-458f-92ff-7b1b413da3af",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
client = WatsonMachineLearningAPIClient(wml_credential)
metadata = {
    client.repository.ModelMetaNames.AUTHOR_NAME : 'Piyush',
    client.repository.ModelMetaNames.AUTHOR_EMAIL :
'piyushkumarbehera.18je0596@cse.iitism.ac.in',
    client.repository.ModelMetaNames.NAME :
'ActiveWindPowerPrediction',
    client.repository.ModelMetaNames.FRAMEWORK_NAME:
'tensorflow',
    client.repository.ModelMetaNames.FRAMEWORK_VERSION: '1.15',

client.repository.ModelMetaNames.FRAMEWORK_LIBRARIES:[{'name':'k
eras', 'version': '2.2.4'}],
    client.repository.ModelMetaNames.RUNTIME_NAME      :
'python',
    client.repository.ModelMetaNames.RUNTIME_VERSION   : '3.6'
```

```
25 }
26 stored_data = client.repository.store_model(model =
   'keras_model.tar.gz',meta_props = metadata)
27 guid = client.repository.get_model_uid(stored_data)
28 deploy = client.deployments.create(artifact_uid=guid)
29 scoring_endpoint = client.deployments.get_scoring_url(deploy)
```