WIND's DANCE WITH ENSEMBLE

"THE DRIVERLESS WIND ENERGY FORECASTER"

INDEX

INTRODUCTION

Wind power has developed rapidly around the world as a promising renewable energy industry. The uncertainty of the wind power brings difficult challenges to the operation of the power system with the integration of wind farms into power grid.
Thus, accurate wind power prediction is increasingly important for stable operation of wind farms and the power grid.

Generally, the wind power prediction methods can be classified into deterministic prediction and probabilistic prediction. The former provides the predicted values for a specific time and the later provides the probability distribution corresponding to the predicted time.

In General, the wind power forecasting methods are summarized into three categories: physical methods, statistical methods and hybrid methods on computational intelligence. The difference between physical and statistical methods is whether the numerical weather prediction (predictions for the weather data like wind speed, temperature, pressure, humidity etc.) is applied. Specifically, the physical methods rely on the data from numerical weather prediction for further prediction. The power output of wind farm is calculated by the physical information of the mathematical model of wind turbine, after obtaining the information of wind farm location.

The statistical methods refer to predicting by mapping the relationship between historical meteorological data and historical power data. As a traditional statistical model, time series models are widely applied in meteorological field. However, the time series model lacks nonlinear fitting ability, which limits its prediction ability.

Project's Purpose

Concerns over global warming and the planetary effects of climate change continue to drive the need to restructure energy procurement approaches. The wind, which is available almost all the time, it has various behaviors according to its geographical location and time. Thus being readily available makes it suitable to be used for the generation of electrical energy. Wind Turbines first convert the kinetic energy of wind into the rotational/mechanical energy and finally into electrical energy.

But, we can't operate the turbines for all time as it will result in high maintenance costs other expenses. So it is necessary to devise a system that helps the owners in planning & scheduling the best operating time for the turbines.

Due to the highly volatile and chaotic nature of wind power, employing complex intelligent prediction tools is necessary. With accurate forecasts on the table, the planning team can devise appropriate generation, distribution, and maintenance strategies

Existing Problem

Electricity generation and consumption must be balanced across the entire grid because energy is consumed almost immediately after it is produced. A large failure in one part of the grid, unless quickly compensated for, can cause current to re-route itself to flow from the remaining generators to consumers over the transmission lines of insufficient capacity, causing further failures.
This means that if the wind stops blowing and a wind farm stops producing electricity, some other source of electricity has to pick up the slack.

So, it is necessary to know beforehand the approximate output of the wind farm. Because not only it helps in determining the best time of operation,its ecnomic viability but also provides the estimate of the energy required from other sources to meet the overall demand.

Proposed Solution

Thus, this project aims to use a combination of Statistical, Machine Learning, and Deep Learning Models to accurately predict wind power generated at a wind farm for the next 48 hours (i.e. for the next 2 days) based on weather conditions.

The Flask application is deployed on the IBM cloud, fetches the forecasts data in real-time when the user logs in. The developed application recommends /suggests the best time to utilize the energy from a single/combination of wind farms. The simple and convinient UI provides user the flexibility to switch across various pages of application to access the features.
Not only it provides the best time of operation and point forecasts of energy generated for the next 48 hours but also a complete report of the wind farm to the user. The report includes power share for the 5 largest farms in India, their wind speed forecasts, temperature forecasts, and a detailed comparison of various parameters via line graphs, pie charts, etc.

Because the app updates on regular intervals, automatically, it is termed as "Driverless". In other words, when the date changes at midnight, the previous data is removed after creating the

backup. New Data is fetched from the weather API, Newly Generated Forecast, reports are sent to the server without any human intervention.

THEORETICAL ANALYSIS

Mathematical Model

Wind turbines work by converting the kinetic the energy in the wind first into rotational kinetic the energy in the turbine and then electrical energy that can be supplied, via the power grid.

Under constant acceleration, the kinetic energy of an object having ***mass 'm'*** and ***velocity 'v'*** is equal to the ***work done 'W'*** in displacing that object from rest to a distance s under a ***force 'F'*** , i.e.

$$E = W = Fs$$

According to Newton's Law, we have:

$$F = ma$$

Hence, $$E = mas \quad (1)$$

Using the third equation of motion:

$$v^2 = u^2 + 2as$$

we get: $$a = (v^2 - u^2)/2s$$

Since the initial velocity of the object is zero, i.e. u = 0 , we get:

$$a = v^2 / 2$$

Substituting it in equation (1), we get that the kinetic energy of a mass in motions is:

$$E = 1/2 * m * v^2 \quad (2)$$

The ***power(P)*** in the wind is given by the rate of change of energy:

$$P = dE / dt = 1/2 * v^2 * dm/dt \quad (3)$$

As mass flow rate is given by:

$$dm/dt = \rho A * dx/dt$$

The rate of change of distance is given by:

$$dx/dt = v$$

Replacing velocity with ***v***:

$$dm/dt = \rho A v$$

Hence, from equation (3), the power can be defined as:

$$P = 1/2\ \rho * A * v^3 \quad (4)$$

BETZ LIMIT

A German physicist Albert Betz concluded in 1919 that no wind turbine can convert more than 16/27 (59.3%) of the kinetic energy of the wind into mechanical energy turning a rotor. To this day, this is known as the **Betz Limit** or **Betz' Law**. The theoretical maximum power efficiency of any design of wind turbine is **0.59** (i.e. no more than 59% of the energy carried by the wind can be extracted by a wind turbine). This is called the "**power coefficient**" and is defined as:

$$Cp(max) = 0.59$$

The Cp value is unique to each turbine type and is a function of wind speed that the turbine is operating in. By the time we take into account the other factors in a complete wind turbine system - e.g. the gearbox, bearings, generator and so on - only 10-30% of the power of the wind is ever actually converted into usable electricity.

Hence, the power coefficient needs to be factored in equation (4) and the extractable power from the wind is given by:

$$P = 1/2\ \rho * A * v^3 * Cp \quad (5)$$

where, ρ = Density (kg/m3)
A = Swept Area (m2)
v = Wind Speed (m/s)
Cp = Power Coefficient
P = Power (W)

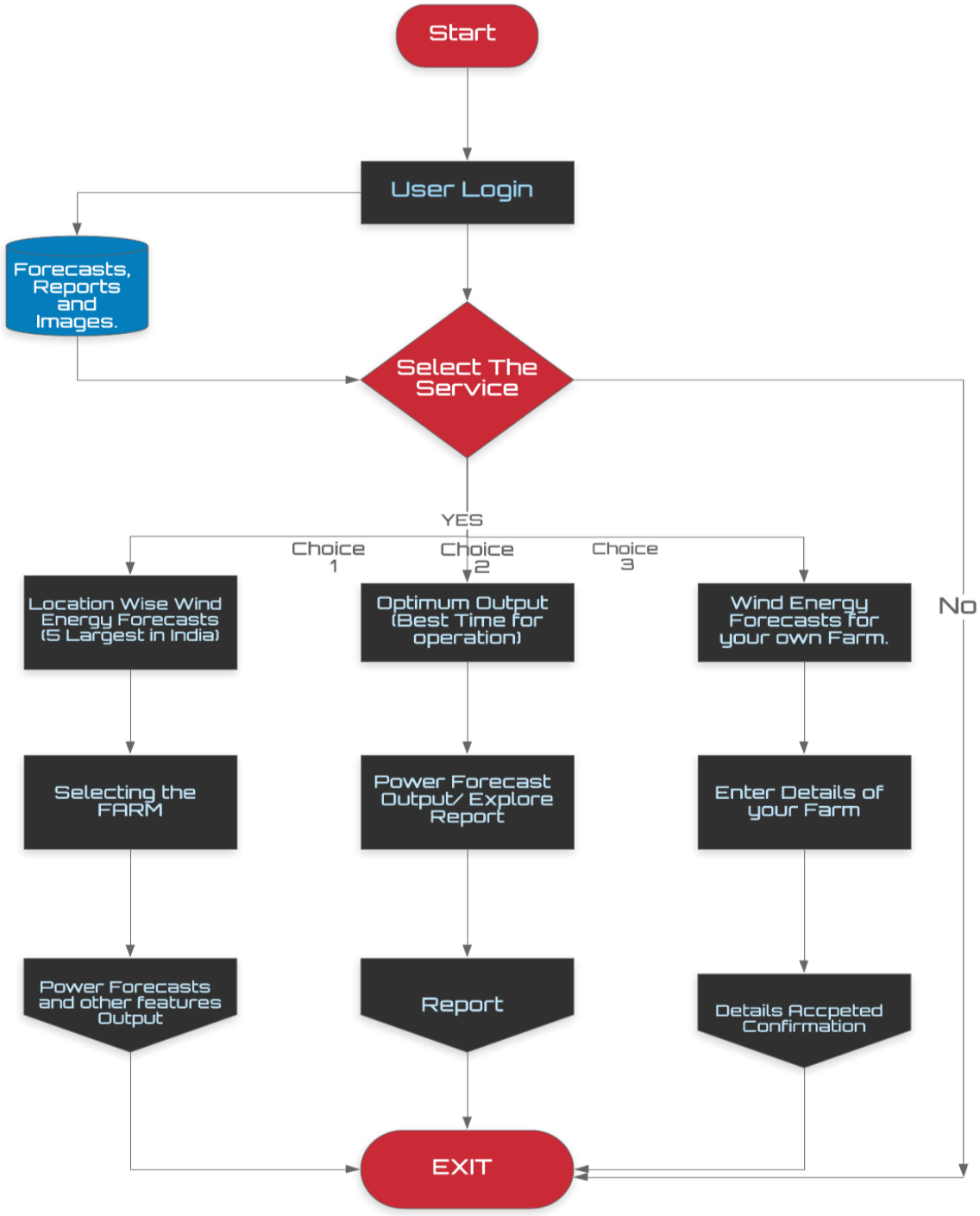
FLOW CHARTS

Application Workflow

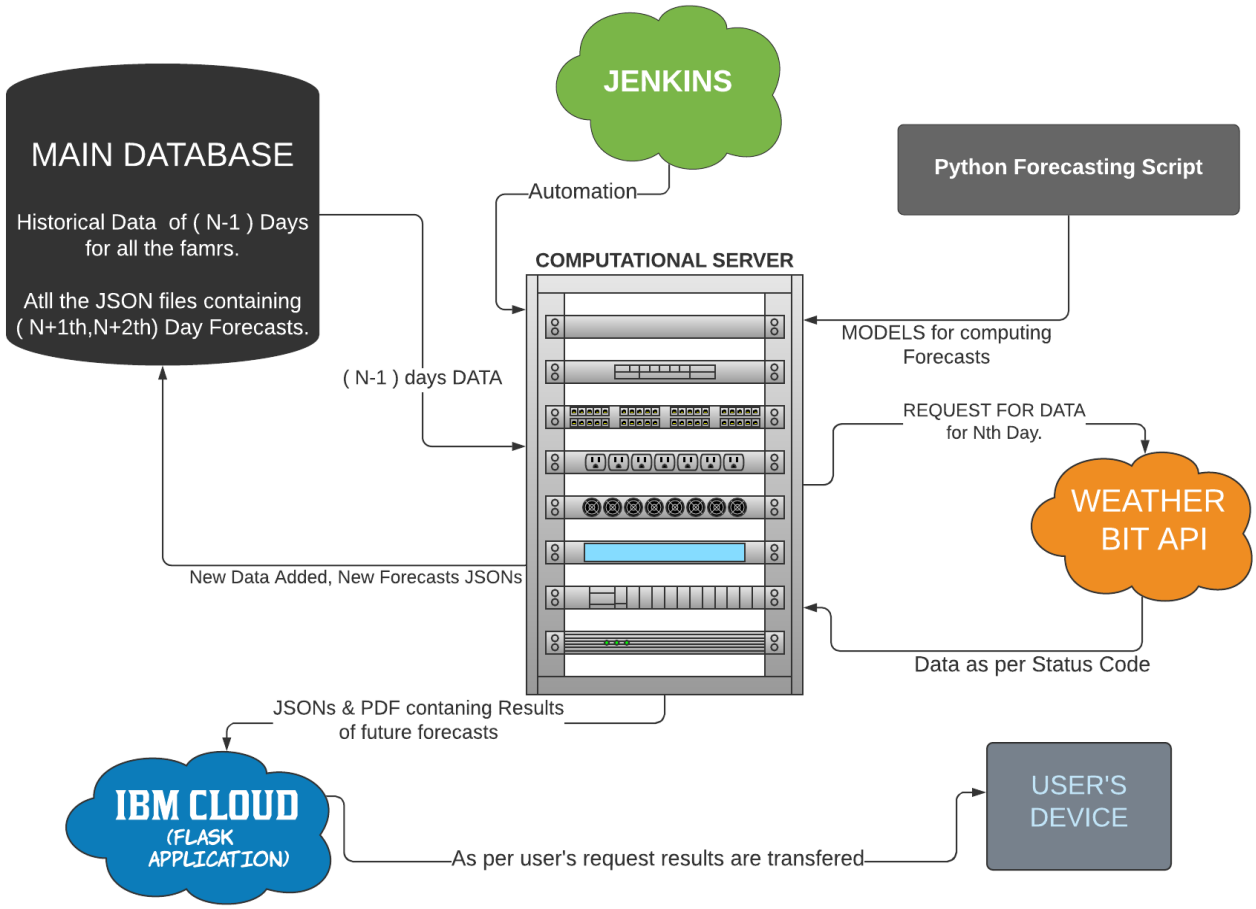
The application workflow provides the basic idea about how the user will be able access various features of the application.

- Before accessing any feature, the user need to login into the application. If he/she does not posses any account then they are welcomed to create one.
- After the login, the user can make 3 choices, either they can see the farm wise forecasts, or check the optimum value forecasts or if they wish to have forecasts for their own farm then they can fill the form with correct details.
- Inside every choice, the user inturn gets many choice to visualize and observe the predictions. The details are kept as simple as possible so that it can be easily understood by people from non technical background as well like planning managers, operations manager etc.

APPLICATION WORKFLOW



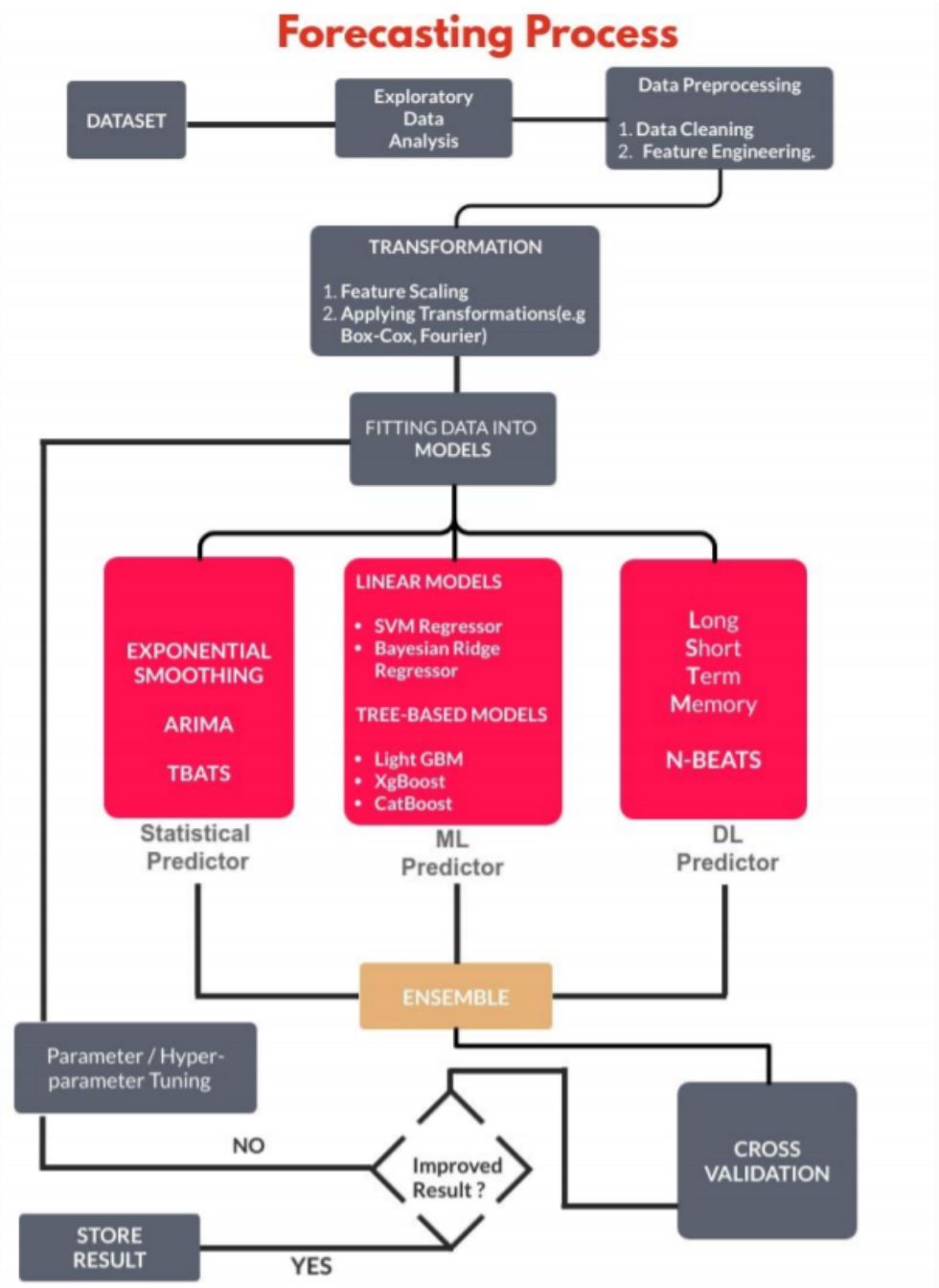
Back-End Flow of the Application



- Above figure shows how data travels between resources to obtain the final wind energy forecasts.
- The jenkins application is responsible for automation of the whole process. It is responsible for invoking the other resources.
- Jenkins, at first, invokes the python forecasting script which retrievesthe historical data from the database and then requests for the current data from the WEATHER BIT API via the api call. The data once recieved is concatenated with the old data and the new data is then fed to the models.
- ML, DL and Statistical models then compute forecasts for the next 24 hours. The result is then sent to the IBM cloud where the Flask Application is responsible for displaying the output to the user.

Flow of Forecasting Model

The flow demonstrates how forecasts are generated from the raw data fetched from the API/Dataset.



ENVIRONMENT DETAILS

Hardware Details:

1. The Python Script was simulated on a 64-bit windows 10 OS machine with 16 GB RAM having Intel (R) 8th gen i-7, 8685U processor (1.8 Ghz base, 4.8 Ghz turbo boost), 8 cores, Nvidia GeForce GTX 1050 (GPU training for Neural Network).

2. The Android Application Script was created on a 64-bit windows 10 OS machine with 8GB RAM having 7th gen i-5, 7200U processor (2.5 Ghz) and 8 cores.

3. The Computational Server on which Jenkis performs forecasting with Models is a Ubuntu 18.04 machine with 32GB RAM having Intel (R) Xeon(R) E5-2676 v3 processor (2.40 Ghz) and 8 cores.

Software Details:

The Python Script was simulated on PyCharm and Google Collaboratory.
Language: Python 3.6
Deep Learning Frameworks: Keras, PyTorch and Tensorflow.
Statistical Libraries: Statsmodel, Scipy, pmdarima.
Data Visualisation Libraries: WindRose, Seaborn,Matplotlib, Plotly Express.

The android script was simulated on andriod studio**.**
Flask: Python framework to create Restful API to deploy the predicting model on server.
Flutter: Android Framework by Google for integrating the hybrid app with real-time monitoring of data.

DEVELOPMENT

Account/ Environment Setup:

We created accounts for all the team members, required/will be required during the project for accessing various resources. E.g., IBM Academic Initiative Account for accessing IBM's cloud and other facilities, Android Studio account for creating an android application, H2O.ai's account for accessing the Driver less AI, etc.

Data Extraction & Cleaning:

- We first extracted the Historical Observations of weather conditions from the Meteosat API by providing the appropriate coordinates and credentials. The data thus obtained is used as the training data for the forecasting models.

- The data obtained from API is in JSON format with a dictionary-like structure, so to make data easy to handle and process, we convert it into a data frame using python's PANDAS library. The ***Time*** feature of the data is used as an index. The data frame is then sorted as per the index values.

- We then check if the data contains any null value or not. We observed that, the feature ***Wind Direction*** has null values but the ***Wind Speed*** is never zero for the data under observation and this is theoretically not possible. We replace the null values with the mean value of the feature.

FEATURE ENGINEERING

In this step we manually generated features that we thought couldhelp the algorithms to understand the patterns in a more effective manner. With the use of relevant features, the complexity of the algorithms reduces. Even if we somehow misjudged the best fit algorithm that is not ideal for the situation, the results would still be accurate.

Types of Features:

1. Lag Features: Lags are essentially the delays in signal value(Time Signals) by a certain amount of time. These lag features helps us define the autocorrelation i.e. how similar a time series is with itself. The Power Generation time series which is highly dependent on the weather conditions like Temperature, Wind Speed, Pressure etc., will correlate positively with itself on

daily basis (day/night temperature drop) as well as yearly basis(summer/winter temperatures).

2. Date related Features: Date related features are very effective in time series modelling as they have indirect or direct relation with the target value. Because the weather conditions are strongly governed by the time in a year/month, introducing date realted features to the algorithms is likely to help in achieving better results.

3. Rolling Mean/Std: Rolling mean / Moving average is generally used as a data preparation technique to create a smoothed version of the original dataset.Smoothing is useful as it can reduce the random variation in the observations and better expose the structure of the underlying causal processes.

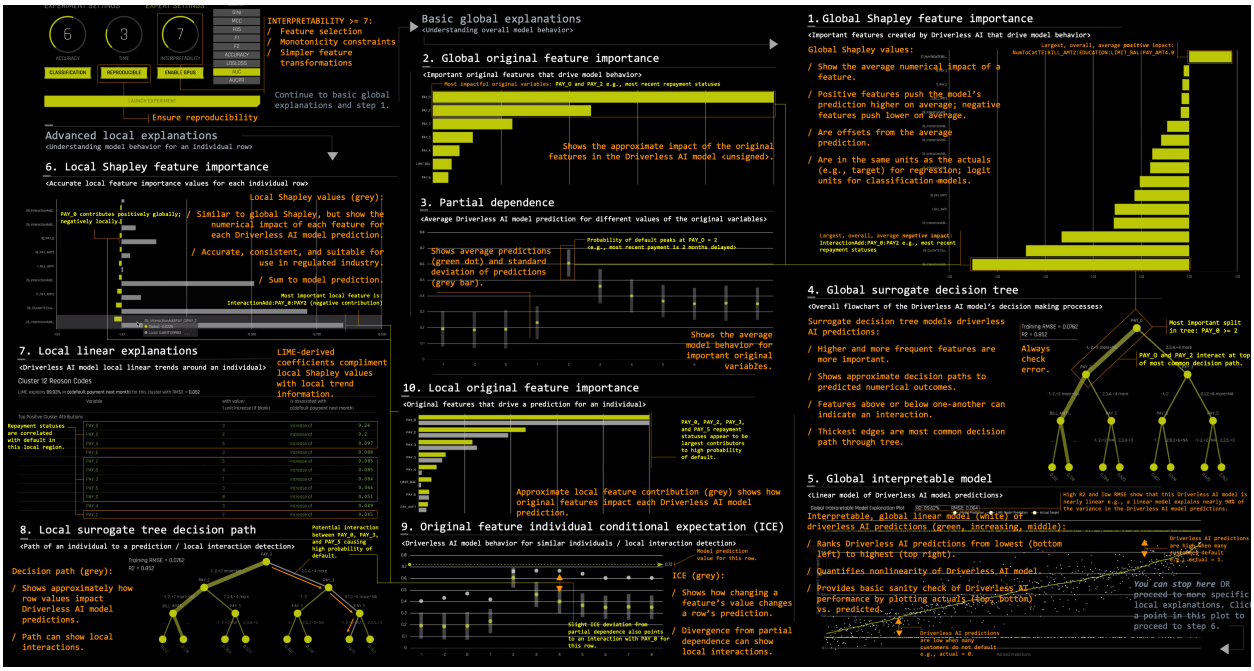
4. Target Encoding: Instead of relying on the information that a feature is related to the target, we explicitly encode the feature in such a way that its presence is directly related tot the target value hence the name TARGET ENCODING.

H20.i's Driverless AI for Time Series

In order to gain some insights into the data, feature relations, distribution of data, we performed an AutoAi experiment on H2O's Driverless Ai for Time Series.

Following are some of the key points of the experiments:

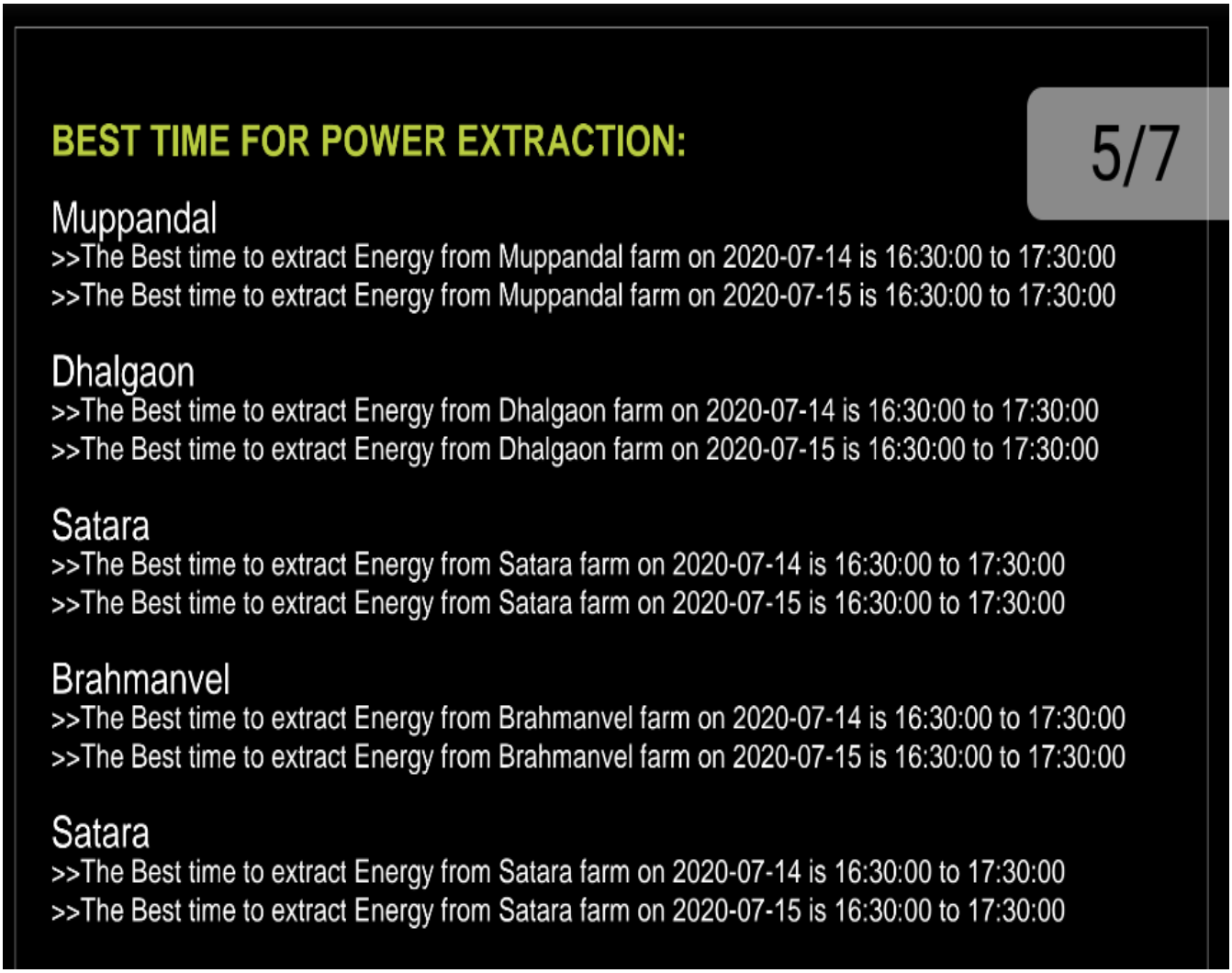
1. The experiment showed that for the linear split in data for training, validation and testing many features had different distributions in all the 3 sets.
2. Power and Log transformed features were contributing highly in the target prediction.
3. Linear and polynomial interaction between Wind Speed, Pressure and Temperature was also bringing significant improvement



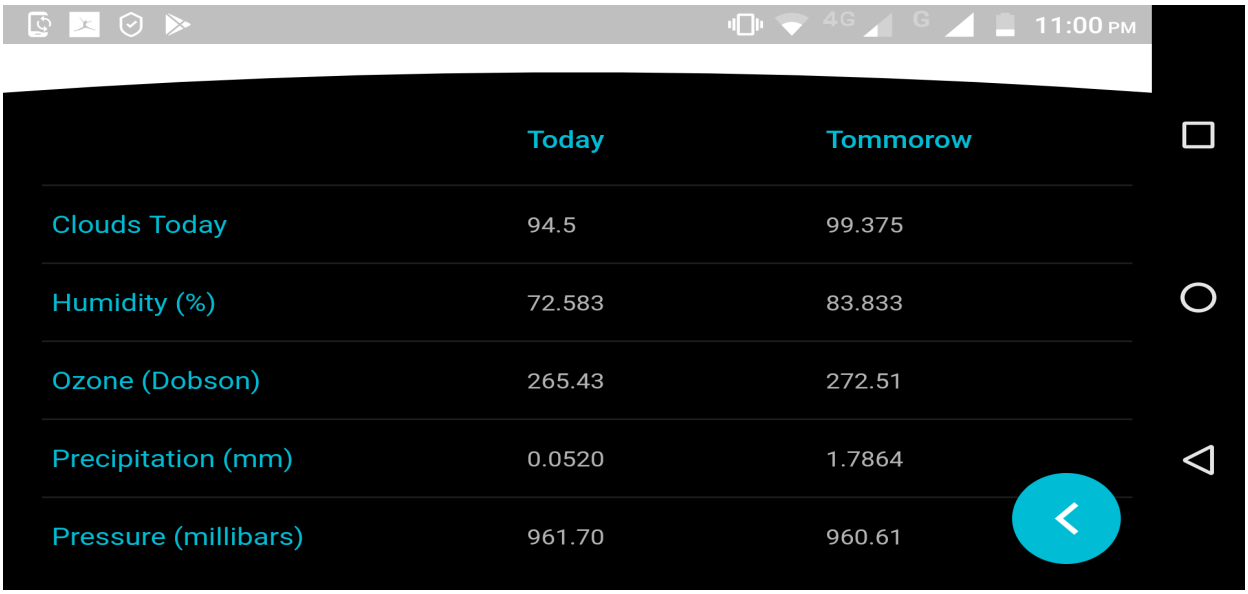
RESULT

After training the data on all the models, the result from all the models is combined and is stored in file. The final file is then sent to the application via IBM Cloud. The app then displays result on the UI.

BEST TIME TO EXTRACT ENERGY FROM WIND FARM OUTPUT



Features for next 24 hours(Today) and next 48 hours (Tommorow)



- From the output screens of the the user interface it very easy to intrepet the results without having any prior technical knowledge.
- Thus, the output and observtions can be used to effectively schedule the operation time of a turbine so as to generate maximum energy.
- The best time for extracting the energy from farm is also explicitly provided in the output which can be of great importance for the farm owners.

CONCLUSION

The wind energy forecasting application is successfully deployed using IBM Cloud. The application provides details like the power ouput of 5 largest wind farms of India for the next 48 hours and the best time to extract energy from them.

Advantages

1. The output of the application is very easy to interpret.
2. The application will not incur any kind of data loss in case of any kind of failure be it hardware or software relate. Because after every compuation i.e. whenever new files are generated, the data is safely backed up.
3. The app is "driverless" i.e it operates on its own and whenever the date changes, new data is fetched, new forecasts are generated, new reports and results are generated, all necessary files are backed up and the final output is sent to the application.

Limitations

1. Due to the complex algorithms and methodologies used in the development the application, the code may be hard to debug.
2. Due to real-time fetching some functionality of the application may not perform as desired under slow/weak internet connections.
3. The application depends on historical data for training the models so if in any case the API from which the data is being fetched goes down then the application might not be able to produce accurate forecasts.

FUTURE SCOPE

1. For determining the best time to extract the energy from the wind farm, instead of using the naive approach used in this project, we can use more advanced algorithms like 'Maximum Sub-Array Problem' to determine the longest most productive time period for wind farm operation.

2. Addition of powerfull models that can map non-linear data ver well can make the application more stable and reliable.
3. Due to computational and other resource related limitations some good performing models were dropped from the final ensemble. Thus, in large computational environment the existing approach can give better results.
4. More detailed analysis of the Energy supply and demand operation can be done if the daily electricity demand data is taken into account.

REFERENCES

1. http://web.mit.edu/windenergy/windweek/Presentations/Wind%20Energy%20101.pdf - for Mathematical Modelling.

2. https://www.raeng.org.uk/publications/other/23-wind-turbine - Mathematical Modelling.

3. https://sites.google.com/view/raybellwaves/blog/using-xgboost-and-hyperopt-in-a-kaggle-comp - Hyperopt for XgBoost and LighGBM parameter search.

4. https://gluon.mxnet.io/chapter12_time-series/issm-scratch.html - Innovation State Space Model.

5. https://arxiv.org/abs/1905.10437 - Neural basis expansion analysis for interpretable time series forecasting(N-BEATS).

BIBLIOGRAPHY

Names: Adhyansh Bhardwaj, Anant Jakhmola

College Name: Maharaja Agrasen Institute of Technology, Graphic Era University

Work Title: Generating Wind Energy Forecasts for determining the best time to extract energy from the Wind Farms.

APPENDIX

1. **Weather BIT API :** https://www.weatherbit.io/api
2. **IBM CLOUD:** https://www.ibm.com/in-en/cloud

Codes:

API REQUEST FOR FETCHING DATA:

```
import requests
import json
key_meteostat = '### key for meteostat API ###'
latitude = 8.150        #Co-ordinates of Muppandal Wind Farm
longitude = 77.3524
url =
f"https://api.meteostat.net/v2/point/hourly?lat={latitude}&
lon={longitude}&alt=104&start={current_date}&end={day_e}"
header = {'x-api-key': key_meteostat}
response = requests.get(url, headers=header)

weather = json.loads(response.text)
```

Lag Features code:

```
df['lag_1'] = df['Energy'].shift(1)
df['lag_2'] = df['Energy'].shift(2)
df['lag_3'] = df['Energy'].shift(3)
df['lag_12'] =  df['Energy'].shift(24)
```

Date and calendar related features code:

```
df['quarter'] = pd.DatetimeIndex(df.index).quarter
df['week_of_year'] = pd.DatetimeIndex(df.index).weekofyear
df['day_of_week'] = pd.DatetimeIndex(df.index).dayofweek
df['day_of_month'] = pd.DatetimeIndex(df.index).day
df['hour_of_day'] = pd.DatetimeIndex(df.index).hour
```

Rolling mean code:

```
df['rolling_mean_1'] = df['Energy'].rolling(window=1).mean()
df['rolling_mean_2'] = df['Energy'].rolling(window=2).mean()
df['rolling_mean_3'] = df['Energy'].rolling(window=3).mean()
df['rolling_mean_6'] = df['Energy'].rolling(window=6).mean()
for i in [36,78,144]:
    print('Rolling period:', i)
    df['rolling_mean_'+str(i)] = df['Energy'].transform(lambda x: x.shift(1).rolling(i).mean())
    df['rolling_std_'+str(i)]  = df['Energy'].transform(lambda x: x.shift(1).rolling(i).std())
```

Target Encoding code:

```
def mean_encoding(col_to_encode,target,new_name):
  encode = df.groupby(col_to_encode)[target].mean()
  df.loc[:,new_name] = df[col_to_encode].map(encode)

df['Month'] = pd.DatetimeIndex(df.index).month
encod_month = df.groupby('Month')['Energy'].mean()
df.loc[:, 'month_mean_enc'] = df['Month'].map(encod_month)

def wind_direction_text(num):
  val=int((num/22.5)+.5)
  arr=["N","NNE","NE","ENE","E","ESE", "SE" ,"SSE" ,"S" ,"SSW",
```