Machine Learning for Solar Energy Prediction

BMS College of Engineering

Machine Learning

ELECTRONICS AND COMMUNICATIONS ENGINEERING

In association with

NOKIA NETWORKS, BANGALORE

Team Details: Priyanshu M; Roshan Nayak; K S Eshwar Subramanya Prasad; K Shivanithyanathan; Subramanya K; Tanay Somnani; Venkatesh Subramanya Iyer Giri

Mentor:

Renganayaki S,

System Specification Engineer, Nokia

Overview

Introduction

- Solar energy is going to play a measure role in the future global energy supply. Due to the fluctuating nature of solar energy, an efficient use is possible depending on reliable forecast information.
- We analyse how the different factors affect the prediction of energy production and how our dataset can be used to predict the expected output of sustainable sources.
- As a result, we facilitate users the necessary information to decide how much they wish to invest according to the desired energy output for their particular location.

Data Collection

- The Training dataset was taken from American Meteorological Society (AMS) 2013-2014 Solar Energy Prediction Contest.
- The dataset is of size 2.84 GB. The data are in **netCDF4** files with each file holding the grids for each ensemble member at every time step for a particular variable.
- Each netCDF4 file contains the latitude-longitude grid and time step values as well as metadata listing the full names of each variable and the associated units.
- Each netCDF4 file contains the total data for one of the model variables and are stored in a multidimensional array.

Data Preparation

- Weather features were contained in netCDF4 files stored in a multidimensional array.
- NetCDF4 libraries available in **R** language were used to convert netCDF4 format file for further data cleaning and processing.
- Missing values in time series weather features were replaced using **linear and spline interpolation** techniques.
- Daily dataset was prepared by averaging over the entire day (for sunlit hours) to represent each data by single data instance.

Data Preparation

	A	B	C	D	E	F	G	H	I	J
1	Date	Cloud coverage	Visibility	Temperature	Dew point	Relative humidity	Wind speed	Station pressure	Altimeter	Solar energy
2	2/1/2016	0.1	9.45	3.11	0.32	79.46	4.7	29.23	30.02	20256
3	2/2/2016	0.8	3.94	6.99	6.22	93.6	13.29	28.91	29.7	1761
4	2/3/2016	0.87	8.7	1.62	0.02	85	16.73	29.03	29.82	2775
5	2/4/2016	0.37	10	-2.47	-5.89	74.52	9.46	29.46	30.26	28695
6	2/5/2016	0.52	9.21	-2	-4.15	82.03	5.92	29.55	30.35	9517
7	2/6/2016	0.13	8.12	0.91	-1.62	81.03	5.48	29.44	30.24	26973
8	2/7/2016	0.21	10	4.24	0.54	73.8	12.71	29.09	29.88	22365
9	2/8/2016	0.87	7.84	-3.33	-5.05	83.53	14.46	28.96	29.75	4995

Figure: Prepared Daily Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Hour	Cloud covera	Visibility	Temperature	Dew point	Relative hum	Wind speed	Station press	Altimeter	Solar energy
2	31/01/2016	24	0.00	5.00	1.40	0.89	95.56	9.00	29.10	29.89	0.00
3	01/02/2016	1	0.00	7.88	1.16	0.62	91.04	7.04	29.11	29.90	0.00
4	01/02/2016	2	0.00	9.84	1.22	0.96	89.28	8.96	29.12	29.91	0.00
5	01/02/2016	3	0.00	9.84	1.02	0.61	89.12	6.36	29.14	29.93	0.00
6	01/02/2016	4	0.00	9.88	0.83	0.45	90.08	6.12	29.15	29.94	0.00
7	01/02/2016	5	0.00	9.84	0.77	0.10	85.44	5.08	29.16	29.95	0.00
8	01/02/2016	6	0.00	9.92	0.37	-0.01	89.12	4.72	29.19	29.98	0.00
9	01/02/2016	7	0.00	10.00	0.47	-0.04	90.08	6.00	29.20	29.99	84.29

Figure: Prepared Hourly Dataset

Exploratory Data Analysis

- Exploratory Data Analysis was performed on dataset for identifying the features, a number of observations, checking for null values or empty cells,etc.
- EDA was carried out for both daily data and hourly data to get the first impression of data before training

```
%matplotlib inline
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
```

```
df = pd.read_csv(r"Downloads/hourly-dataset_final2.csv")
```

```
df.drop('Unnamed: 11',inplace= True,axis = 1)     #removing unnecessary columns
```

```
df.head()
```

	Date	Hour	Cloud coverage	Visibility	Temperature	Dew point	Relative humidity	Wind speed	Station pressure	Altimeter	Solar energy
0	31-01-2016	24	0.0	5.00	1.40	0.89	95.56	9.00	29.10	29.89	0.0
1	01-02-2016	1	0.0	7.88	1.16	0.62	91.04	7.04	29.11	29.90	0.0
2	01-02-2016	2	0.0	9.84	1.22	0.96	89.28	8.96	29.12	29.91	0.0
3	01-02-2016	3	0.0	9.84	1.02	0.61	89.12	6.36	29.14	29.93	0.0
4	01-02-2016	4	0.0	9.88	0.83	0.45	90.08	6.12	29.15	29.94	0.0

Figure: EDA on Hourly dataset using Pandas Library

Exploratory Data Analysis

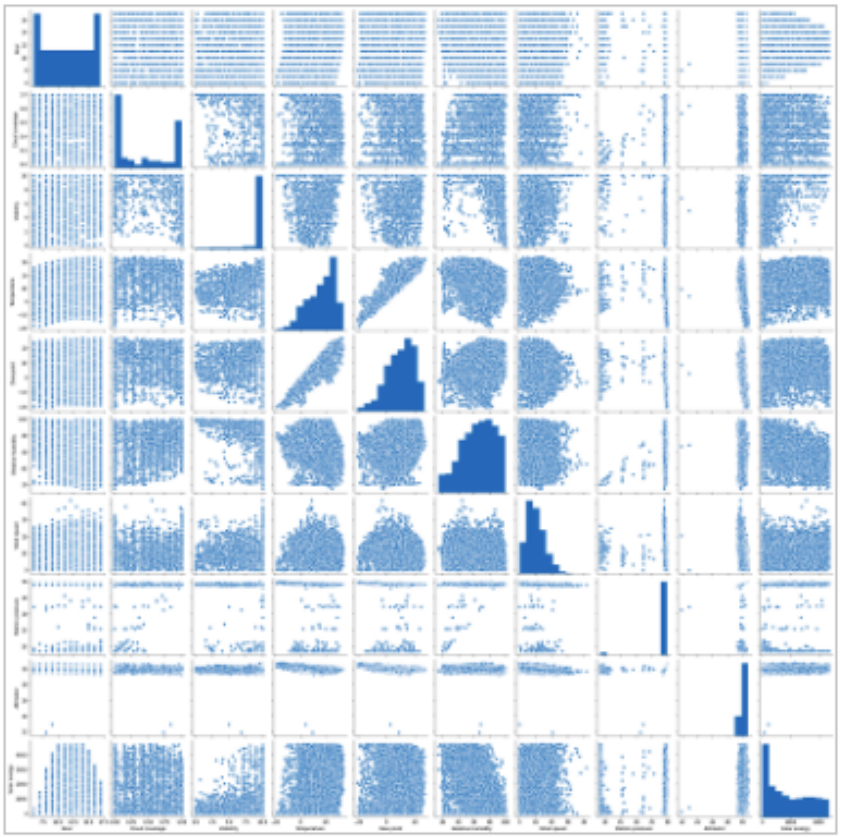

Figure: Pairplot

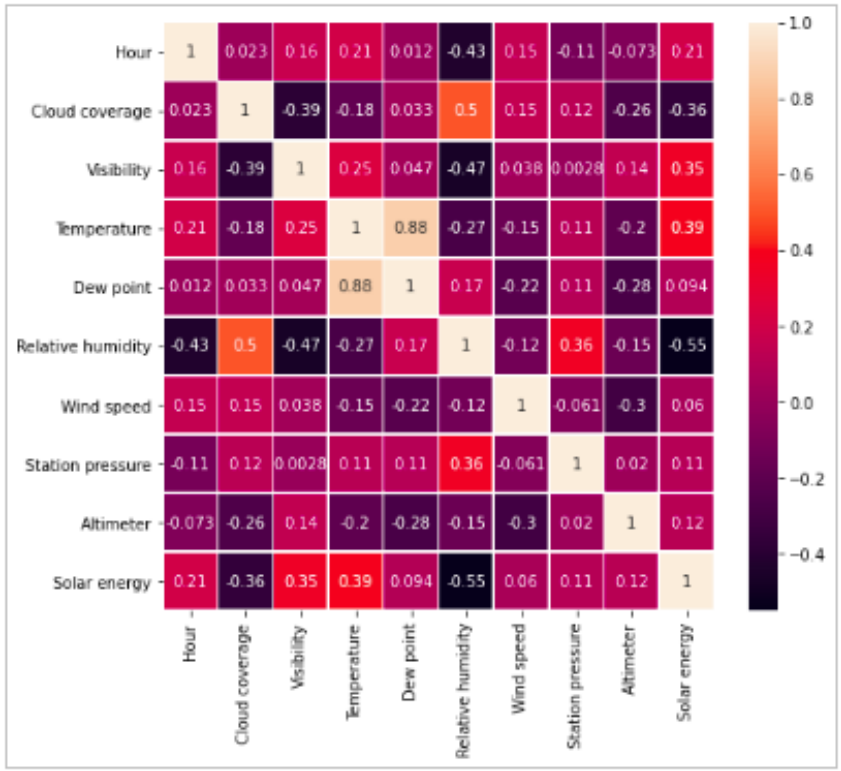

Figure: Correlation Heat Map

Dimensionality Reduction

- Dimensionality reduction refers to the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data.
- Principal Component Analysis (PCA): PCA was used to provide low dimensional visualization (2D) from high dimensional space.

Principal Component Analysis (PCA)

```
scaler = MinMaxScaler()
#scalar = StandardScaler()

df = pd.DataFrame(scaler.fit_transform(df.iloc[:,1:]), columns=df.columns[1:])

x = df[['Cloud coverage','Visibility','Temperature','Dew point','Relative humidity','Wind speed',
        'Station pressure','Altimeter']]
pca = PCA().fit(x)
z = pca.transform(x)
print(z)
plt.scatter(z[:,0],z[:,1])
```

Figure: PCA on Hourly dataset

Dimensionality Reduction: PCA

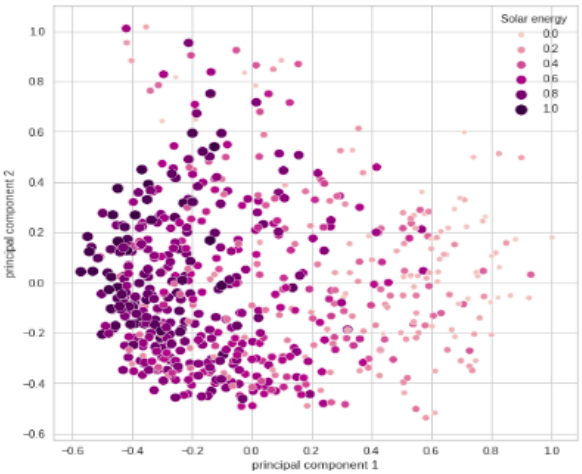

Figure: 2 component PCA visualization (colour coded on normalized Solar output)

```
from sklearn.model_selection import KFold

crossvalidation = KFold(n_splits=5, shuffle=True)
model = LinearRegression().fit(df_pca.iloc[:,0:2],df_pca.iloc[:,-1])

n_scores = cross_val_score(model,df_pca.iloc[:,0:2],df_pca.iloc[:,-1], scoring="r2"
                           ,cv=crossvalidation, n_jobs=1)

print(str(n_scores))
print('Mean r2_score: %.3f' % (np.mean(n_scores)))
```

```
[0.47294389 0.52498537 0.50114031 0.54833157 0.59913496]
Mean r2_score: 0.529
```

Figure: Two component PCA used as predictors to fit Multiple regression model

Model Selection

Several Models were build and model performance was analyzed

Random forest

- Random forests operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the mean/average prediction (regression) of the individual trees.
- We used a model with 100 trees.

Gradient Boosting Regressor

- In boosting, each new tree is a fit on a errors made by the previous tree. Hence here the input to the current tree depends on the outputs of the previous tree.
  Model Parameters: learning_rate=0.01, max_depth=8, max_features='sqrt', min_samples_split=4, n_estimators=900

RandomizedSearchCV

- RandomizedSearchCV is used for hyper parameter tuning, we create a grid and input values for all parameters we want to experiment with then the algorithm returns the best model with minimum error from those combinations of parameters we entered
- The randomizedsearchcv chooses the best RF model with the lowest error, from the parameters we entered.

```
n_e=[int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
max_f=['auto','sqrt']
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
```

```
random_grid = {'n_estimators': n_e,
               'max_features': max_f,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

Figure: parameters used for RandomizedSearchCV

RandomizedSearchCV

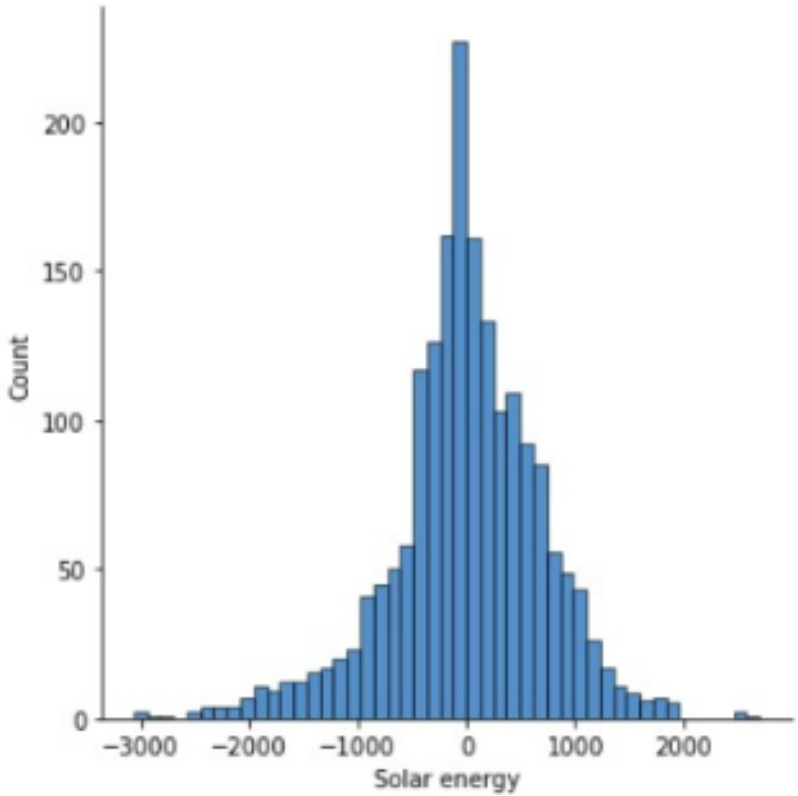

Figure: Error distribution plot

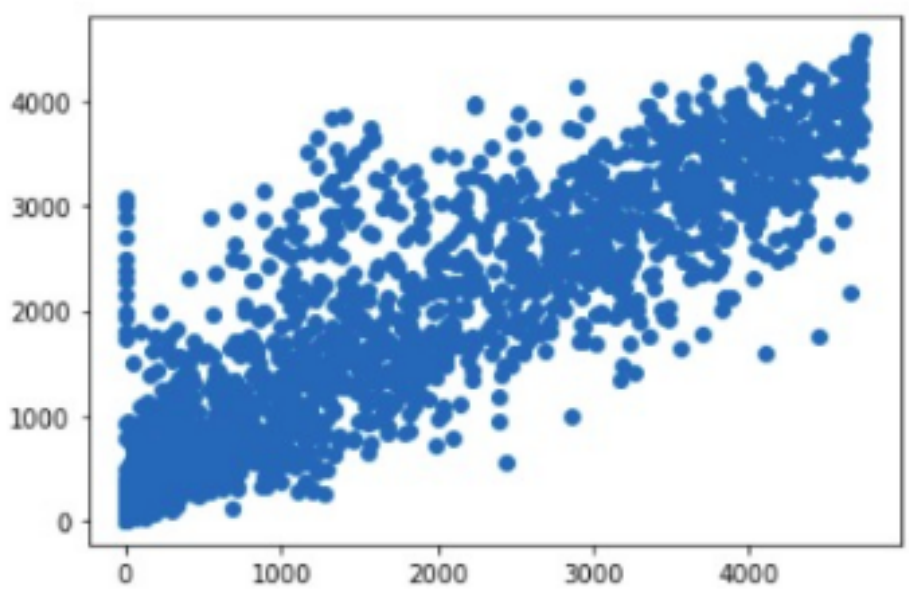

Figure: y true v/s y predict

Gradient Boosted Regression(GBR)

```
learning_rate = [0.001, 0.01, 0.1, 0.2]


n_estimators=list(range(500,1000,100))

max_depth=list(range(4,9,4))

min_samples_split=list(range(4,9,2))

min_samples_leaf=[1,2,5,7]

max_features=['auto','sqrt']

param_grid = {"learning_rate":learning_rate,
              "n_estimators":n_estimators,
              "max_depth":max_depth,
              "min_samples_split":min_samples_split,
              "min_samples_leaf":min_samples_leaf,
              "max_features":max_features}
```

Figure: parameters used for GBR

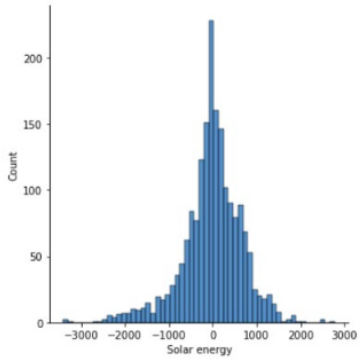

Figure: Error distribution plot

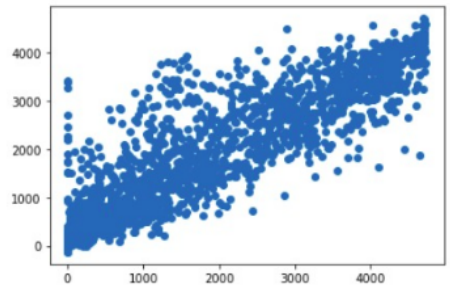

Figure: y true v/s y predict

Model Selection

XGBOOST Regressor

- This algorithm is based on the Gradient Boosting technique and decision tree as weak learners. But its faster than Gradient Boosting Regressor through parallel processing, tree-pruning, and regularization to avoid overfit/bias.
  Model Parameters: n_estimators=800, eta=0.025, max_depth=8, subsamples=0.75

LightGBM Regressor

- This algorithm is based on Gradient Boosting technique as weak learners. It is fast by making use of parallel computing. Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.
  Model Parameters: bootsting_type=dart, n_estimators=2000,learning_rate=0.1,subsample=0.5,num_leaves=100

Model Stacking

- Base estimators and a final estimator are used.
- Another layer of learning are introduced in which the model can learn how to use the outputs of the base models to predict the final output instead of taking the average of the outputs of the base estimators.
- performs better than other ensembling techniques like random forest.

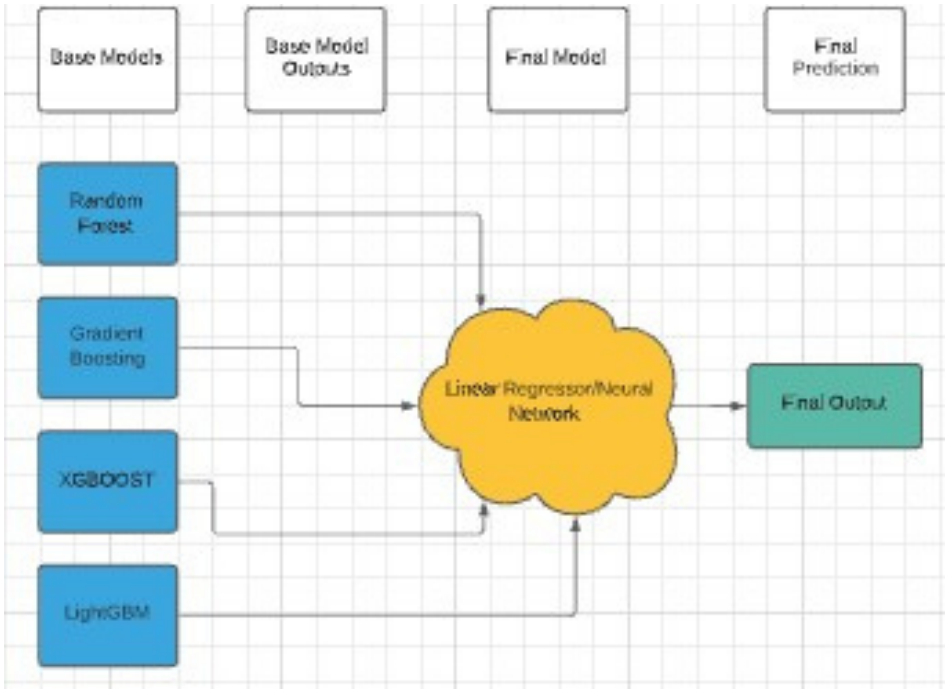

Figure: Stacking Illustration

Model Evaluation

Model	R2 Score
Linear regression	0.44
SVR	0.46
Linear regression on PCA	0.53
NN with 2 hidden layer	0.78
Random forest	0.82
Gradient boosting regressor	0.82
LightGBM regressor	0.83
Stack model 1	0.83
XGBOOST regressor	0.84
Stack model 2	**0.85**

Table: Model comparison and evaluation

- Stack Model 1: Four base estimator and Linear Regressor as Final estimator
- Stack Model 2: Four base estimator and Neural Network with one hidden layer and relu as activation function

Time Series Forecasting

- Forecasting in general is the process of making predictions based on past and present data and most commonly by analysis of trends.
- In Machine Learning we Forecast Time series data by training the model on past values. Using the technique we can forecast future data which can be helpful.
- In this project we have forecasted solar energy data by analyzing past solar energy values.
- We use **LSTM** i.e., Long short term memory. It is a type of RNN but it solves many drawbacks which RNN has.

Time Series Forecasting

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 9, 50)             10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 9, 50)             20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
```

Figure: Model Architecture: 3 LSTMs and one Dense layer for our model

```
Epoch 95/100
177/177 [==============================] - 2s 9ms/step - loss: 0.0098 - val_loss: 0.0098
Epoch 96/100
177/177 [==============================] - 1s 8ms/step - loss: 0.0100 - val_loss: 0.0095
Epoch 97/100
177/177 [==============================] - 1s 8ms/step - loss: 0.0100 - val_loss: 0.0097
Epoch 98/100
177/177 [==============================] - 1s 8ms/step - loss: 0.0096 - val_loss: 0.0099
Epoch 99/100
177/177 [==============================] - 1s 8ms/step - loss: 0.0092 - val_loss: 0.0100
Epoch 100/100
177/177 [==============================] - 1s 8ms/step - loss: 0.0093 - val_loss: 0.0106
```

Figure: Model Training: val loss and training loss for 100 epoch

Application Development

- We developed a web application to help the user to utilize our machine learning model and predict the total solar energy generation at a particular location, if they want to set up a solar energy plant.
- Web application is deployed on Heroku: Cloud Application Platform

Tech Stack of Web Application

- Flask web server
- Materialize CSS
- Leaflet.js
- Chart.js

Data Collection using API

- **Weatherbit**: Weatherbit API is used to retrieve current weather observations from over 47,000 live weather stations. We provide the city name, from which the latitude and longitude is retrieved and used in the API, and the API provides the weather information required.
- **Opentopodata**: It is an elevation API, which is used for the "Altimeter" feature.The elevation is provided using the city name or the latitude and longitude.
- **Timezonedb**: To get time in given latitude and longitude.
- **Opencagedata**: The OpenCage Geocoding API provides reverse (lat/long to text) and forward (text to lat/long) geocoding via a RESTful API.

Backend Functional Process Chart

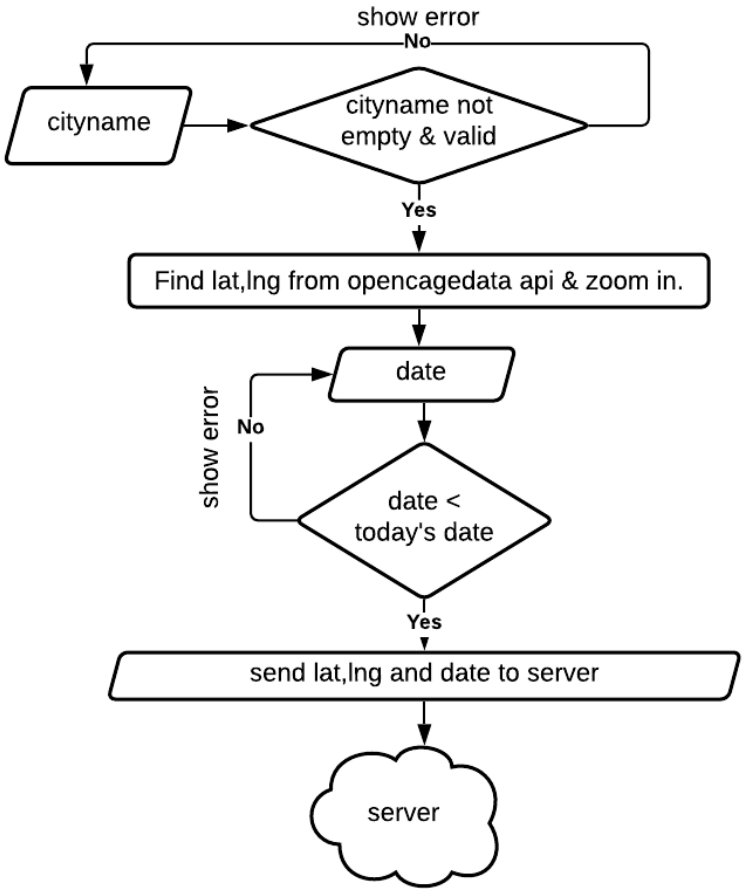

Figure: Backend Process Chart

Application UI Backend

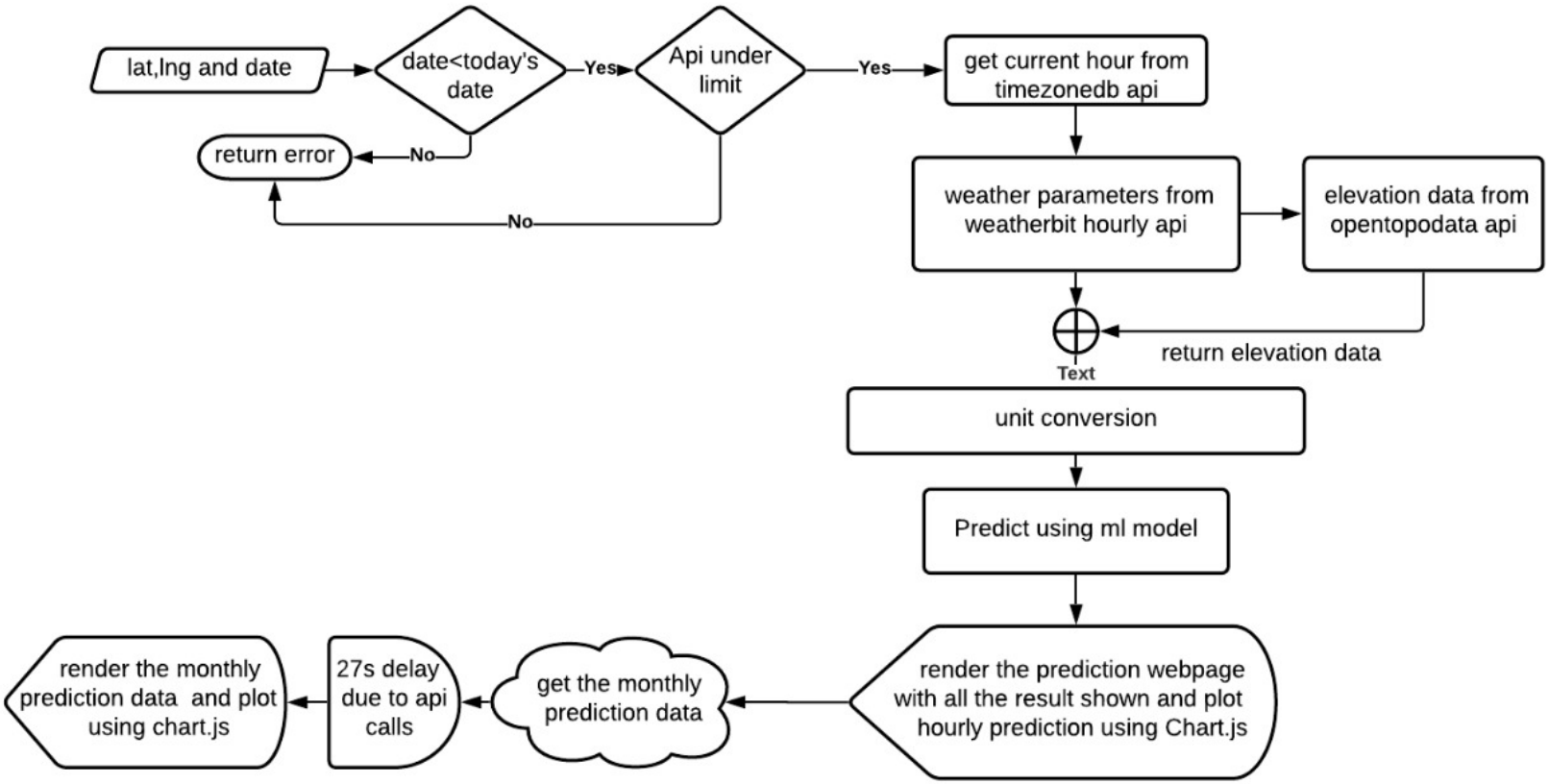

Figure: Flow Chart for Hourly Prediction results

Application UI Backend

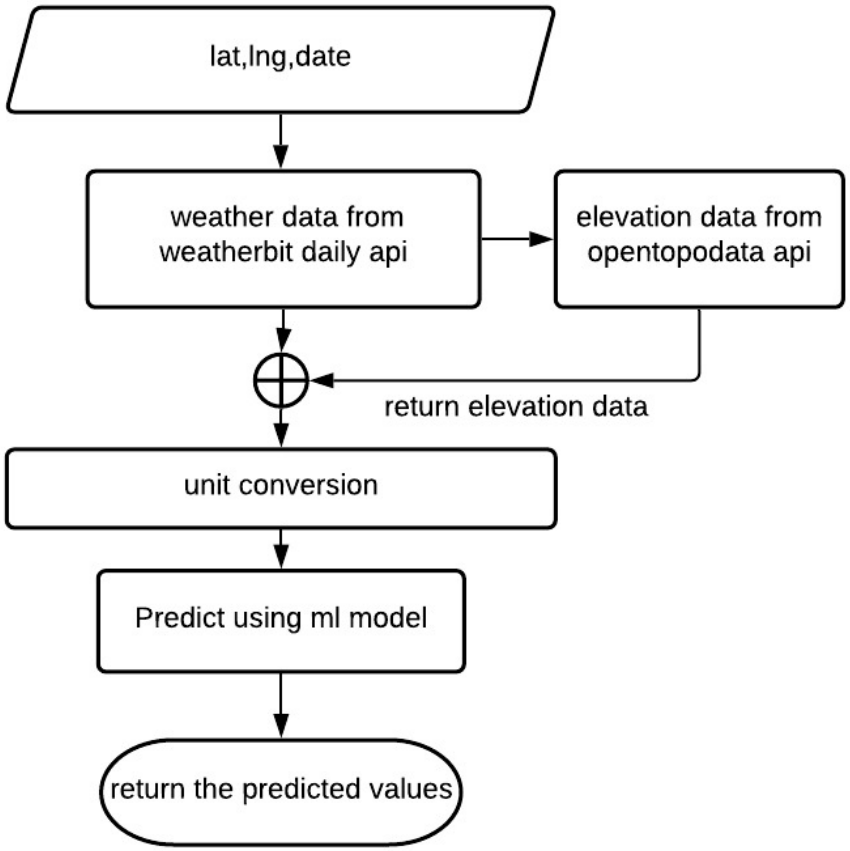

Figure: Flow Chart for Daily Prediction

Future Scope and Conclusion

- Model will aid users for solar Power forecasting at any particular location in making practical design decisions.
- This forecasting approach is a key step towards the development of a data-driven decision-making framework for power system operation.
- Similar study and conclusion can be drawn on large scale national solar grid and wind farms' energy forecasting models.

Team

Priyanshu M

K Shivanithyanathan

Roshan Nayak

Subramanya K

Tanay Somnani

K S Eshwar Subramanya Prasad

Venkatesh Subramanya Iyer Giri

The End