## Solarpanel Power Prediction using ML regression models

* Gowtham M (VCET, IIIrd Year)

**Problem Statement :**

Real-time solar panel power output prediction remains challenging due to varying weather conditions. Current methods often lack localization and real-time adaptability, leading to inaccurate power generation estimates.

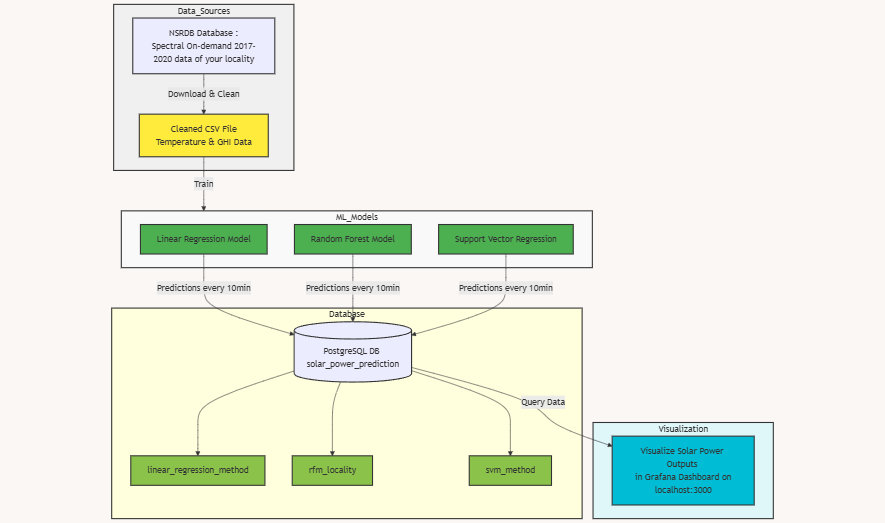
**Proposed Solution:**

I proposed a real-time ML-powered system designed to predict solar panel power output using multiple regression models. The system uses historical weather data from the NSRDB (https://nsrdb.nrel.gov/data-viewer), specific to your locality, to provide accurate and localized solar power forecasts. By analyzing key weather parameters like Temperature and Global Horizontal Irradiance (GHI), it ensures that the predictions reflect real-time weather conditions, delivering precise solar power generation estimates.

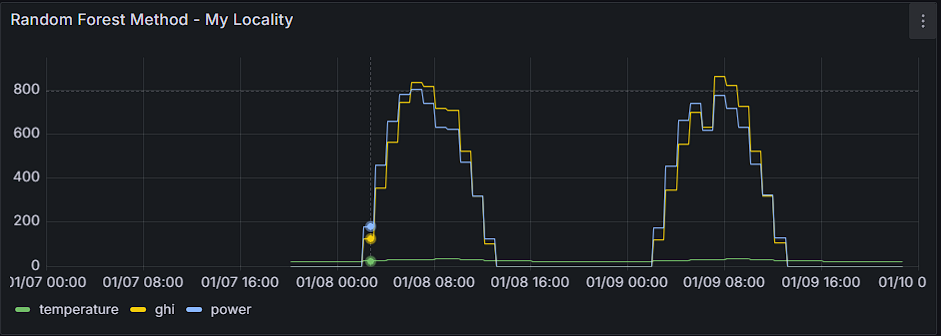
The system integrates three different regression models:

Linear Regression, Random Forest, and Support Vector Regression (SVR).

### System Architecture

****

**My Locality - Random Forest Method - Grafana Dashboard**



## Materials Required :

### Software Prerequisites

* **Python 3.11+**
* **PostgreSQL**
* **Grafana**

**Dataset :**

VisitNSRDB (<https://nsrdb.nrel.gov/data-viewer>) and enter your locality latitude & longitude coordinates. Choose the available dataset. You will receive a mail with the link to the dataset, valid for 24 hrs. Download before it expires!

Python files :

[support vector regression.py](https://drive.google.com/open?id=1cj0c8tMAdK2bViaDYQ1LGhbEHOrwwCE6&usp=drive_copy)  
[support vector regression - plots & metrics.py](https://drive.google.com/open?id=1bNYkFwNulFZdO5OKXOWby-pACMvhSkYT&usp=drive_copy)  
[random forest.py](https://drive.google.com/open?id=1HaHcTfERa_fWXrWqi1AEkXO8y9vXGeH3&usp=drive_copy)  
[linear regression - plots & metrics.py](https://drive.google.com/open?id=1GQ8QUkvKPyRYhar2AD7jFUKwP1B_5IRj&usp=drive_copy)  
[random forest - metrics.py](https://drive.google.com/open?id=1Bg_7p5-QhkbzVk8HBBUlwKqmoXx7Cplr&usp=drive_copy)  
[linear regression.py](https://drive.google.com/open?id=15NAvf9mgww06lfhaqUsb_EOGOaCjytkd&usp=drive_copy)  
My Localilty dataset : [Temperature\_&\_GHI\_ 2017\_2020\_myLocality\_cleaned.csv](https://drive.google.com/open?id=12RhI7dkWBIe6bniRs6zA5ibPsCxGBayx&usp=drive_copy)

**Procedure :**

**Packages installation**

Install all packages using this : pip install numpy pandas scikit-learn matplotlib psycopg2

### Step 1: Clone the Repository

git clone https://github.com/g-wtham/solar\_power\_prediction.git  
cd solar\_power\_prediction

### Step 2: Set Up PostgreSQL

1. Create a PostgreSQL database:

* CREATE DATABASE solar\_power\_prediction;

1. Configure the database tables with the respective table names as per the in the random\_forest.py file given.

### Step 3: Connect Python to Postgres using Psycopg2 package

1. Define postgres database username and password (default-username: postgress; password: root)
2. psycopg2 is used for postgres to python connection

### Step 4: Install Grafana and select the data source (select postgresql)

1. Install Grafana (https://grafana.com/grafana/download) and set up username and password (default username & password: admin)
2. Navigate to locahost:3000, select postgresql as the data source and build dashboards by selecting the corresponding tables from the connected pg database.
3. Toggle the order setting ‘ON’ and set *custom limit*, as default is 50 and can hinder if more data points are plotted.
4. You can export the dashboards as JSON files as well for preserving the templates for external sharing.

### Step 5: Run the System

1. Train the model:

* python random\_forest.py

1. For getting the plots & metrics for the model:

* python random\_forest-metrics.py

1. Visualize Results in Grafana: View the predictions on the Grafana dashboard (localhost:3000)

**Explanation :**

**Data Preparation and Processing**

The system begins by importing weather data from a CSV file containing historical measurements from 2017-2020. The preprocessing steps include:

- Cleaning data by replacing zero values with NaN

- Calculating non-zero means for accurate weightage

- Splitting data into five input features plus temperature and GHI as target variables

**Model Architecture**

1. Linear Regression Implementation

- Utilizes sklearn's LinearRegression for both temperature and GHI prediction

- Performance metrics include MAE, MSE, RMSE, and R-squared values

- Visualizes results through kernel density estimation plots

2. Support Vector Regression Enhancement

- Implements RBF kernel for capturing non-linear relationships

- Features standardization using StandardScaler for better model performance

- Includes inverse transformation to convert standardized predictions back to original scale

3. Random Forest Approach

- Uses ensemble learning with multiple decision trees

- Separate models for temperature and GHI predictions

- Provides better handling of complex non-linear relationships

**Prediction Pipeline**

*Time Series Processing*

- System generates predictions every 10 minutes

- Converts datetime components into feature vectors

- Processes year, month, day, hour, and minute as separate features and passes those into PostgreSQL database

*Power Calculation Algorithm*

The solar power calculation follows a specific formula:

1. Initial power factor calculation: f = 0.18 × 7.4322 × GHI

2. Temperature difference: insi = Temperature - 25

3. Temperature coefficient: midd = 1 - 0.05 × insi

4. Final power output: Power = f × midd

Pass the power output value along with GHI, Temperature into the database.

*Database Integration*

- PostgreSQL database stores predictions

- Separate tables for each prediction method

- Stores timestamp, temperature, GHI, and calculated power

- Handles database connections with error management

*Performance Visualization*

- Implements kernel density estimation plots

- Compares actual vs predicted values for both temperature and GHI

- Provides visual assessment of model accuracy

**Results :**

## Prediction results visualized from various models, showcasing the actual vs. predicted values based on input parameters like GHI and Temperature.

Random Forest method performed better than linear regression and support regression model, as it captures the non-linearity of the dataset well. As the dataset contains more than 43,849 rows, SVR model struggles without performing additional preprocessing steps, while random forest achieves nearly 97.5% accuracy as R² score is 0.9755. Thus, out of 3 regression models, random forest gives us good performance to accuracy ratio.

### Random Forest Metrics :

Mean Absolute Error (MAE): 0.5528913519430765

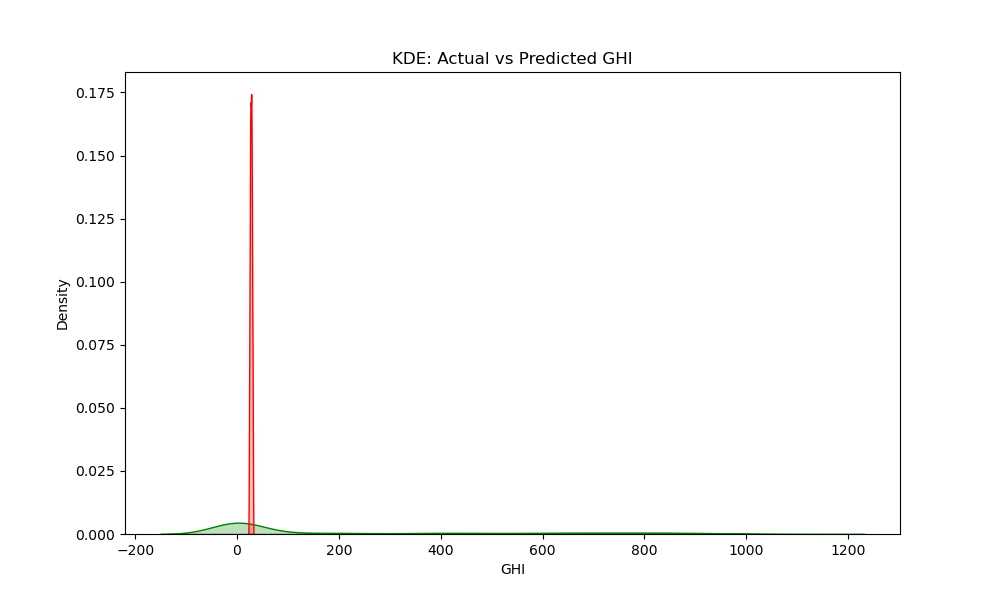
Mean Squared Error (MSE): 0.5644339387885424

Root Mean Squared Error (RMSE): 0.7512881862431635

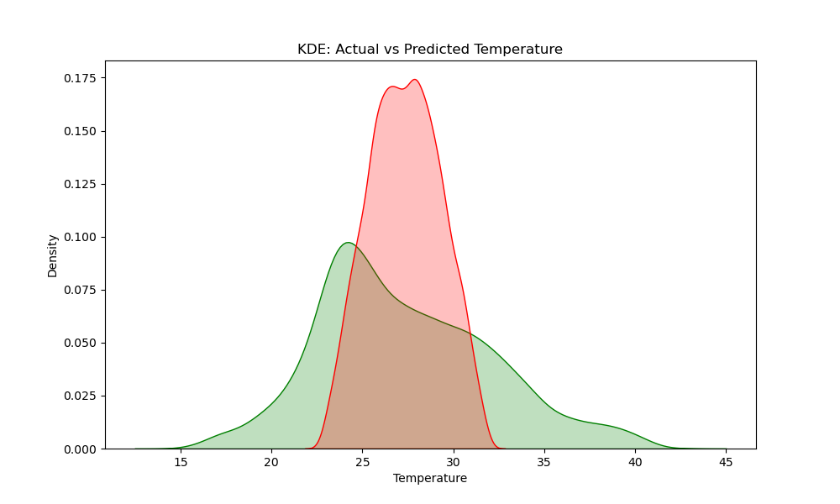
R-squared (R²): 0.9755113005363533

### 1. Linear Regression

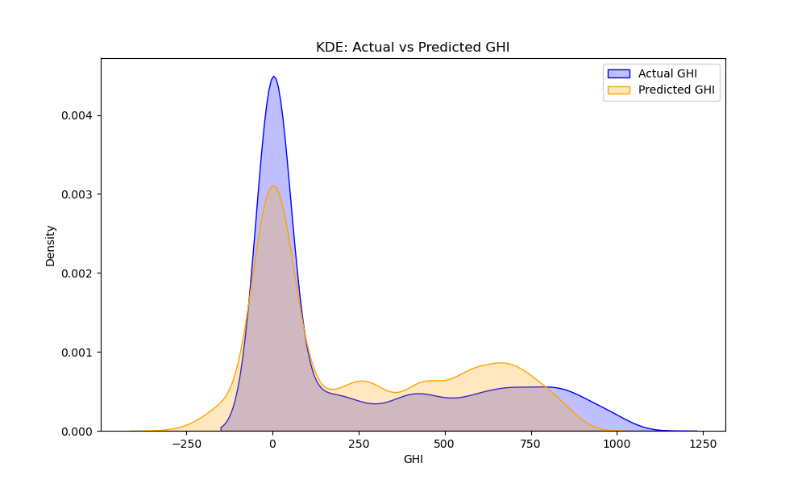
* **Actual vs Predicted GHI**



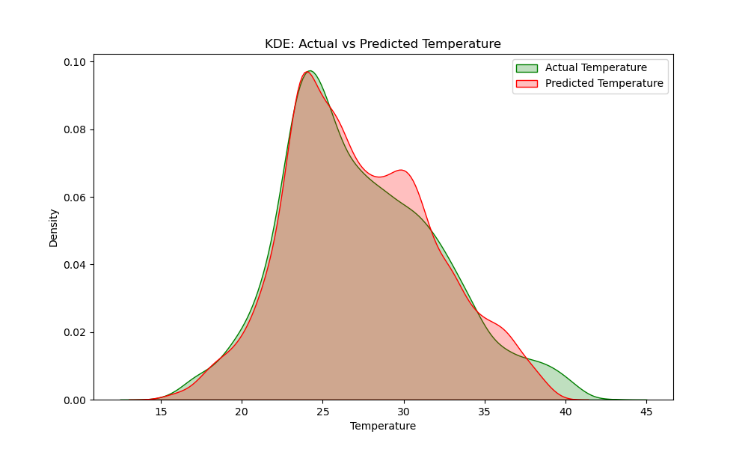
* **Actual vs Predicted Temperature**

  
**2. Support Vector Regression (SVR)**

* **Actual vs Predicted GHI**

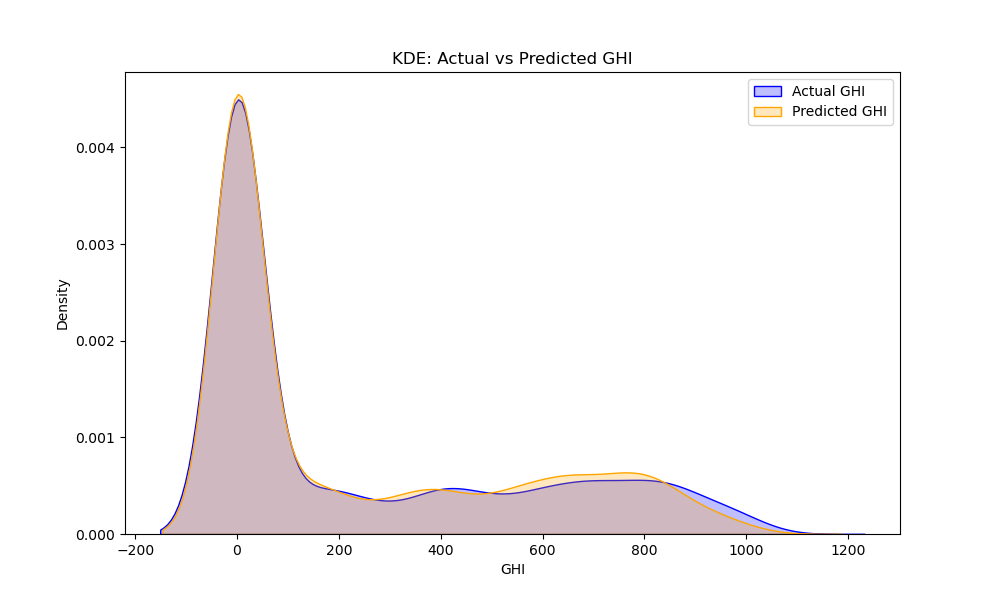


* **Actual vs Predicted Temperature**

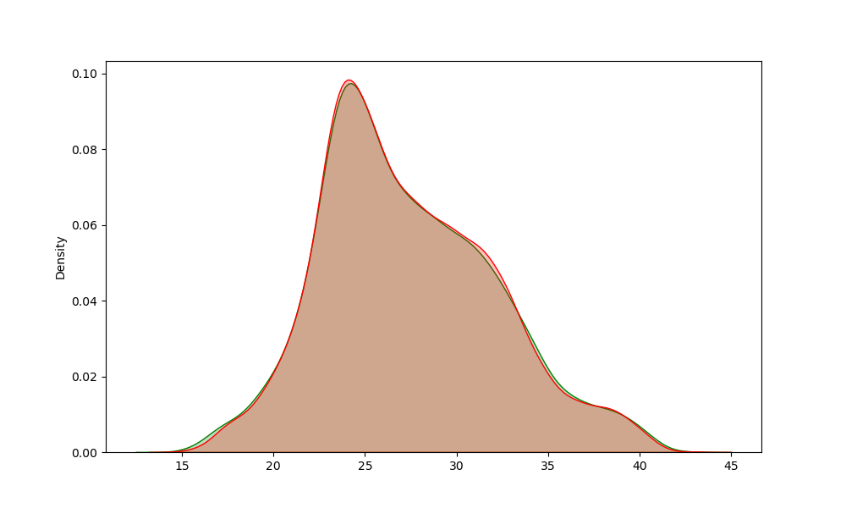


### 3. Random Forest

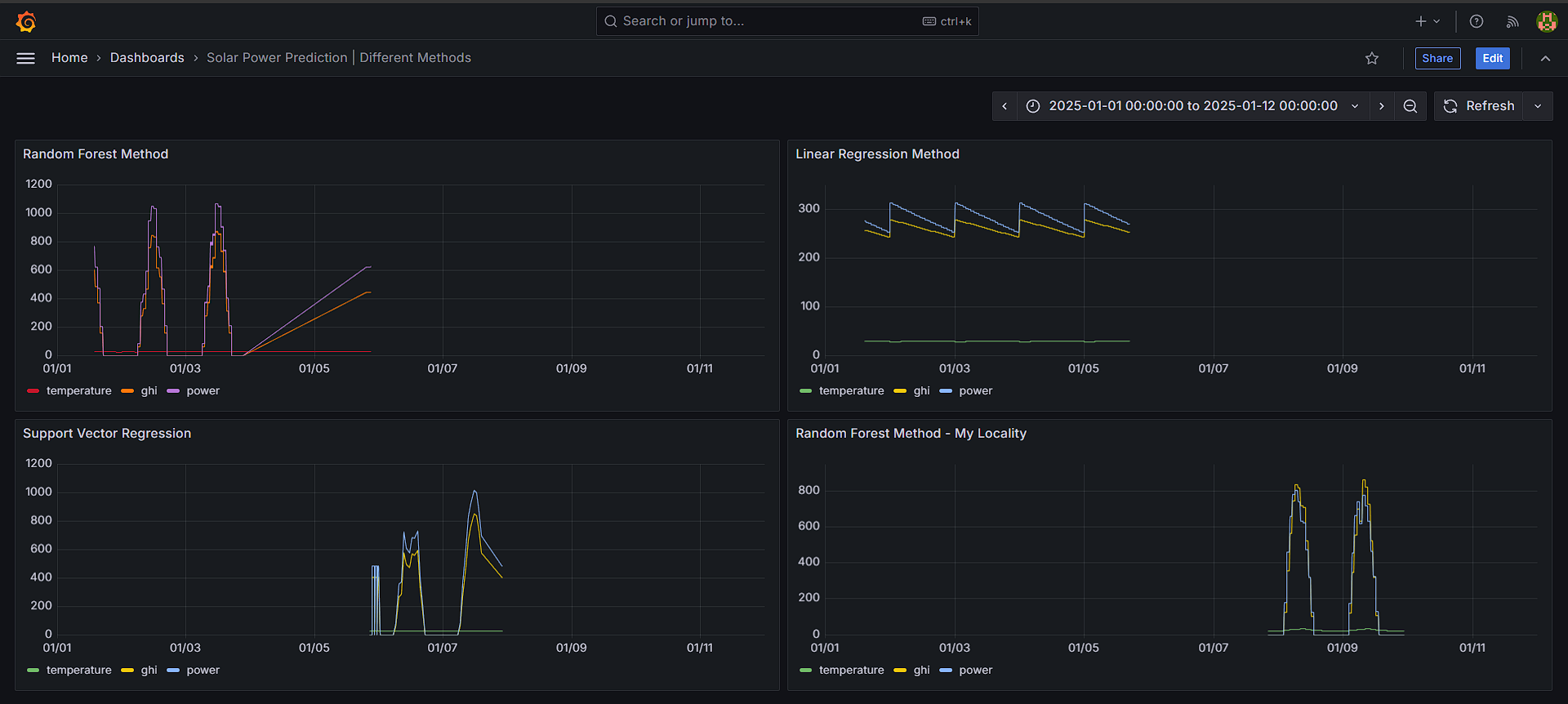
* **Actual vs Predicted GHI**



* **Actual vs Predicted Temperature**



### 4. Grafana Dashboard

**All methods dashboard**