UPES

UNIVERSITY OF TOMORROW

Elements of AIML

ASSIGNMENT 2

Name: Somya Vats

Batch: 11

SAP: 500119012

Roll no: R2142230349

FLASK BASED UI FOR PREDICTING SOLAR PLANT YIELD USING MACHINE LEARNING ON WEATHER AND POWER DATA

- *GITHUB LINK*

Introduction

As climate change concerns continue to grow, global reliance on renewable sources has been increasingly sought out in order to not only follow a greener path, but to specifically address Sustainable Development Goals – SDG 7: Affordable and Clean Energy, and SDG 13: Climate Action. Solar energy provides clean energy, but it remains a hard area to maximise due to the uncontrollable factors of nature, which ultimately leads to variability in solar generation. All these factors make solar power forecasting an indispensable task when it comes to efficient energy management systems, plannings, and resource utilizations.

In this project, we focus on estimating how much energy a solar plant can produce by looking at past generation of the plant and weather conditions. Through the use of python ML models, the objective is to build a model that manages to forecast solar generation for a given weather scenario. Such models are a useful piece of machinery for solar plant operators as they can enhance the decision making by improving resource distribution and assisting in sustainable energy consumption as well.

This task is dependent on using Machine Learning given its sophistication in recognizing intricate structures within extensive amounts of data, self adjusting in real time, and providing forecast abilities which would be impossible otherwise. This project shows the other side of ML: how it changes the game for the renewable business.

Project Overview

This project utilizes machine learning to predict solar plant energy yield based on historical weather and power data. Key factors such as sunlight intensity, cloud cover, temperature, and humidity are used to forecast power output, aiding solar plant operators in efficient energy distribution and storage planning

---

Solution Approach

We trained our model on time-series data, analyzing atmospheric and energy trends to develop predictive insights. Our solution uses Python's Scikit-learn for modeling and Flask as the backend for an interactive web interface, with HTML/CSS for the front-end. The model forecasts future solar generation based on real-time weather data.

Tech Stack

This application was developed with the help of HTML/CSS and Flask as a framework for process-oriented model training and interaction with users, along with the combination of several libraries of the Python programming language.

Python Libraries:

I. Pandas and NumPy were used for data handling and storage as well as for statistical data analysis, that required dealing with big amounts of data.
II. Scikit-learn was utilized for performing most of the machine learning algorithms, model training and hyperparameters tuning.
III. Matplotlib and seaborn libraries aimed to illustrate data and models results in graphical representations.

Flask (Backend): As the backend framework application, Flask offered the simplest integration tool to the developers as minimalistic API routes existed in order to make predictions and query data.

HTML/CSS (Frontend): Used to construct model visualizations and model forecast outputs allowing users to perform prediction tasks and other engaged procedures actively and simply and look at descriptive parts of the data such as images.

CODE DESCRIPTION

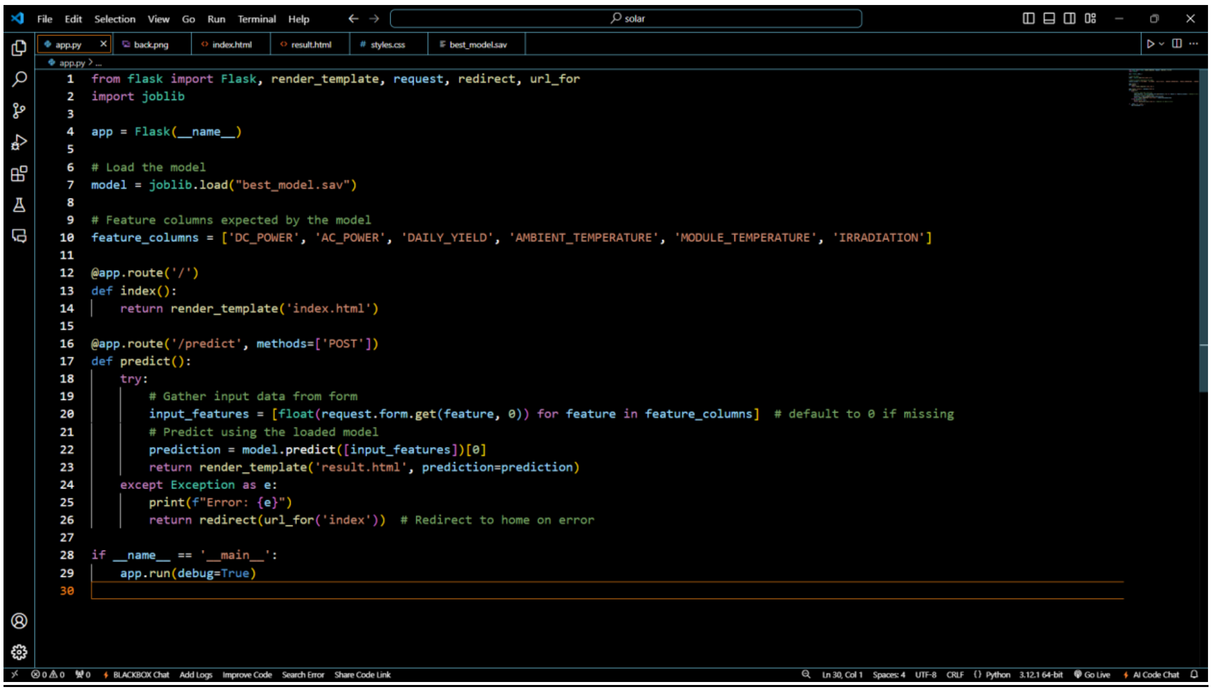
- *GITHUB LINK*

Structure of the folder:

```
Solar
├── app.py              # Main Flask application file
├── best_model.sav        # Trained ML model saved for predictions
└── templates            # Folder for HTML templates and CSS
    ├── index.html        # Home page for input
    ├── result.html       # Results page for displaying predictions
    └── style.css        # Styling for HTML templates
```

App.py:

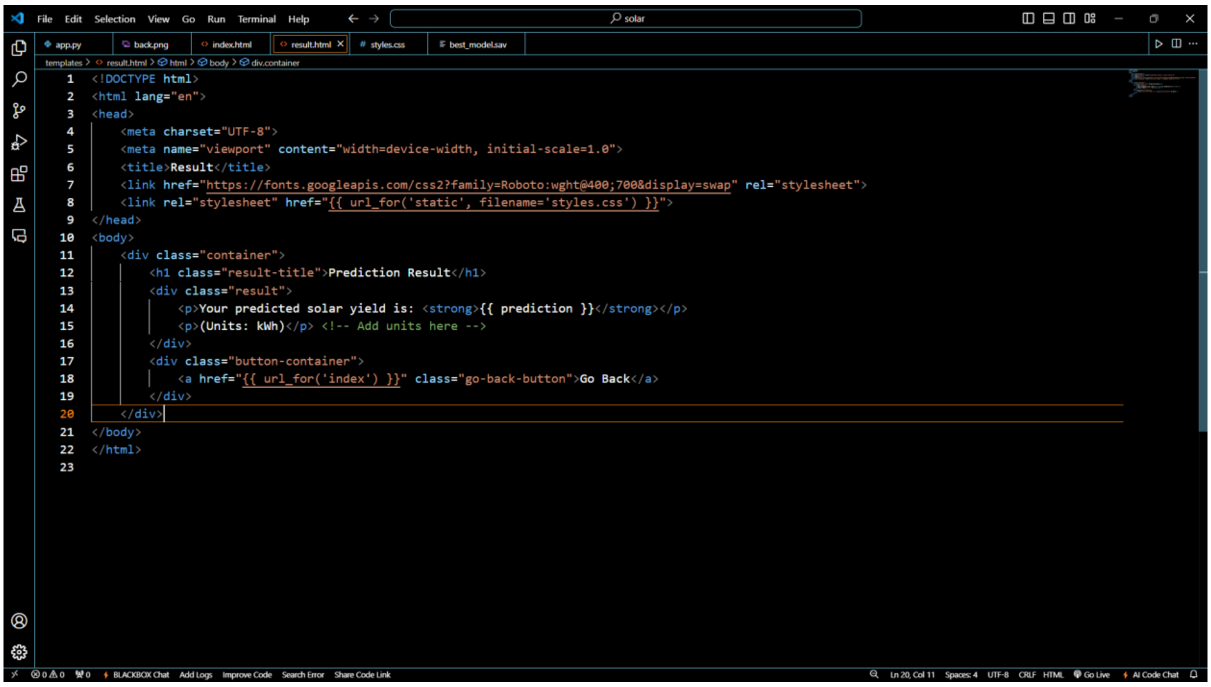
Flask is a lightweight and flexible web framework for Python that allows developers to create web applications quickly with minimal overhead. In the provided code, Flask is used to build a simple web application that accepts user input, processes it, and uses a pre-trained machine learning model to make predictions. The `Flask` object is initialized and routes are defined to handle different web pages, such as the homepage where the user inputs data and the result page where the prediction is displayed. Flask's simplicity and minimal setup make it ideal for small to medium-sized applications, especially when you need to integrate machine learning models, like in this case where a saved model (`best_model.sav`) is loaded and used to predict outcomes based on user inputs. The application gathers user data through a form, uses that data to generate predictions, and renders an HTML template to show the results. Flask is important because it allows developers to focus on the logic and functionality of their application without worrying about complex configurations, making it especially useful for rapid prototyping and machine learning integrations. It also supports easy routing, template rendering, and error handling, which makes managing web applications straightforward and efficient. By using Flask, developers can create dynamic, data-driven web applications that are flexible enough to scale as needed while keeping the development process simple and streamlined.



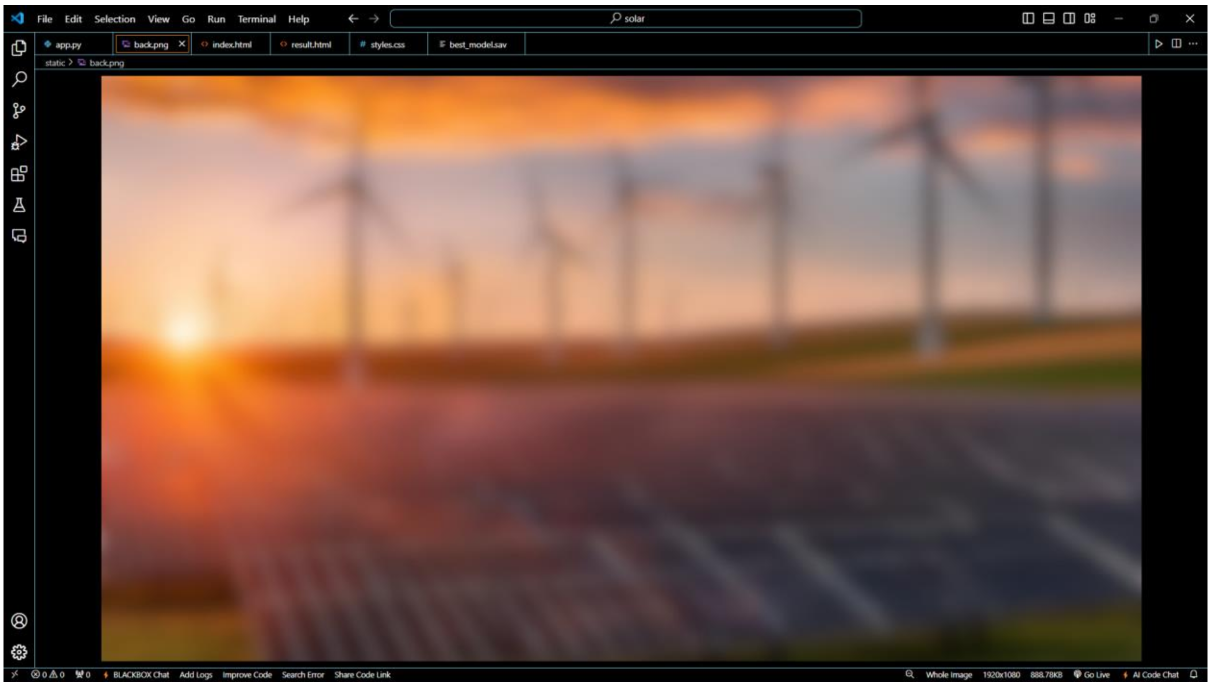
```python
from flask import Flask, render_template, request, redirect, url_for
import joblib

app = Flask(__name__)

# Load the model
model = joblib.load("best_model.sav")

# Feature columns expected by the model
feature_columns = ['DC_POWER', 'AC_POWER', 'DAILY_YIELD', 'AMBIENT_TEMPERATURE', 'MODULE_TEMPERATURE', 'IRRADIATION']

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Gather input data from form
        input_features = [float(request.form.get(feature, 0)) for feature in feature_columns]  # default to 0 if missing
        # Predict using the loaded model
        prediction = model.predict([input_features])[0]
        return render_template('result.html', prediction=prediction)
    except Exception as e:
        print(f"Error: {e}")
        return redirect(url_for('index'))  # Redirect to home on error

if __name__ == '__main__':
    app.run(debug=True)
```

Frontend:

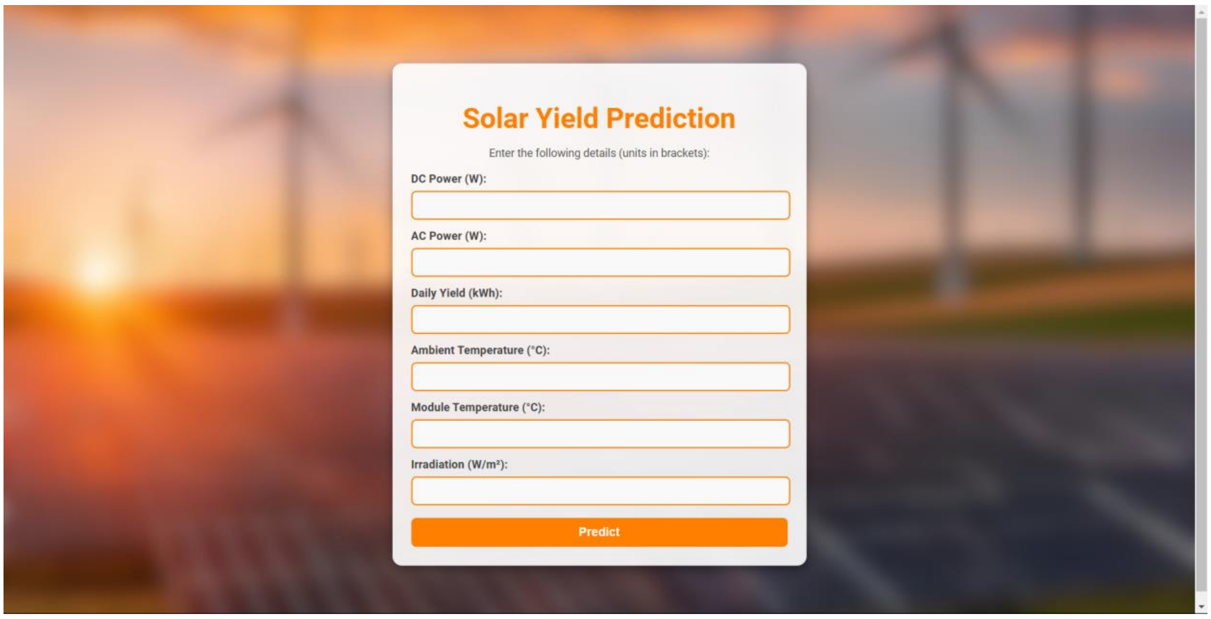
For the frontend of this Solar Yield Prediction application, the goal is to create a user-friendly and visually appealing interface that enhances user experience. The design focuses on simplicity and clarity, ensuring that users can easily input their data and view the prediction results. The HTML structure includes a well-organized form where users enter six key parameters related to solar power production, such as DC Power, AC Power, Daily Yield, Ambient Temperature, Module Temperature, and Irradiation. To make the form visually attractive, it incorporates modern design elements like rounded input fields, bold labels, and a clear call-to-action button for submitting the form. The CSS styles focus on creating a clean and intuitive layout, with a background image and transparent containers that give a sense of depth, as well as hover effects to make the interface interactive. The result page, displayed after the prediction, highlights the solar yield prediction with a prominent result display, and offers a “Go Back” button for easy navigation. The overall approach ensures that users can interact with the application smoothly, while the combination of design and functionality helps present the prediction in an aesthetically pleasing and user-friendly manner.


File Edit Selection View Go Run Terminal Help
solar
app.py
back.png
index.html
result.html
styles.css
best_model.sav
templates > result.html > html > body > div.container
Ln 20, Col 11 Spaces: 4 UTF-8 CRLF HTML Go Live

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result</title>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1 class="result-title">Prediction Result</h1>
        <div class="result">
            <p>Your predicted solar yield is: <strong>{{ prediction }}</strong></p>
            <p>(Units: kWh)</p> <!-- Add units here -->
        </div>
        <div class="button-container">
            <a href="{{ url_for('index') }}" class="go-back-button">Go Back</a>
        </div>
    </div>
</body>
</html>
```


File Edit Selection View Go Run Terminal Help
solar
app.py
back.png
index.html
result.html
styles.css
best_model.sav
static > back.png
Whole Image 1920x1080 888.78KB Go Live

Final result:


Solar Yield Prediction

Enter the following details (units in brackets):

DC Power (W):

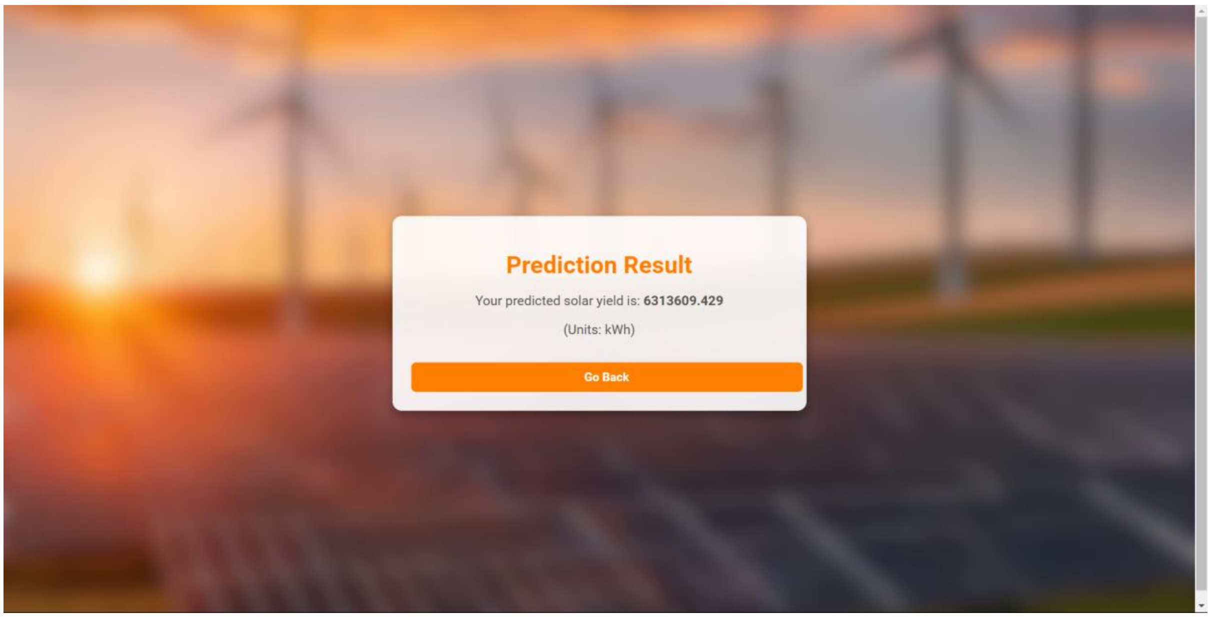
AC Power (W):

Daily Yield (kWh):

Ambient Temperature (°C):

Module Temperature (°C):

Irradiation (W/m²):


Prediction Result

Your predicted solar yield is: **6313609.429**

(Units: kWh)

Practical Applications

The model for predicting solar energy output has practical uses in the renewable energy industry as it supplies the operators, energy companies and policymakers involved in energy production with the necessary foresight. Such predictions ensure effective and efficient energy production. The main advantages are the following:

1. **Higher Efficiency:** When the operators are able to determine the amount of energy to be generated accurately, they can change the production plans in such a way as to optimize efficiency and reduce wastage of energy resources.
2. **Cost Saving:** Factors such as overproduction or underproduction have great operational cost implications due to the need to install energy storage systems or balance the power grid.
3. **Improved Grid Orchestration:** Accurate analyses also help install solar power plants scattered all over the grid as their supporting roles, enhancing grid stability and reducing fossil energy uses.
4. **Sustainability Orientation**: The constructs of the solution endorse energy companies to attain the sustainability agenda as they adopt the low carbon emission options which make solar energy competitive in the energy mix. Such a predictive solution also has a significant impact to other sectors that are not directly associated with renewable energy.

For instance, hospitals that mostly rely on renewable energy sources for emergency needs can be assured of reliability in supply due to the accurate estimation of yield. This model also has applicability such as in government agencies by which its use can determine the performance of solar PV capacity of a nation and thus provide guidance on renewable energy's policies and investment opportunities.

Key Takeaways and Challenges

Through this project, I gained insights into:

- **Technical Skills**: Enhanced understanding of data preprocessing, feature selection, and working with regression models to address real-world problems.
- **Conceptual Insights**: The project emphasized the connection between weather patterns and energy production, illustrating how ML models can leverage these insights.

Key challenges included:

1. **Data Quality**: Cleaning and preprocessing data to handle missing values and inconsistencies.
2. **Model Accuracy**: Initial predictions were suboptimal, requiring model experimentation and hyperparameter tuning to improve accuracy.

Acknowledgements

I am grateful for my mentor’s guidance throughout this project, helping me understand core concepts in machine learning and solar energy forecasting. I also appreciate my peers for reviewing my work and providing valuable feedback.

Name: Somya Vats

Batch: 11

SAP: 500119012

Roll no: R2142230349