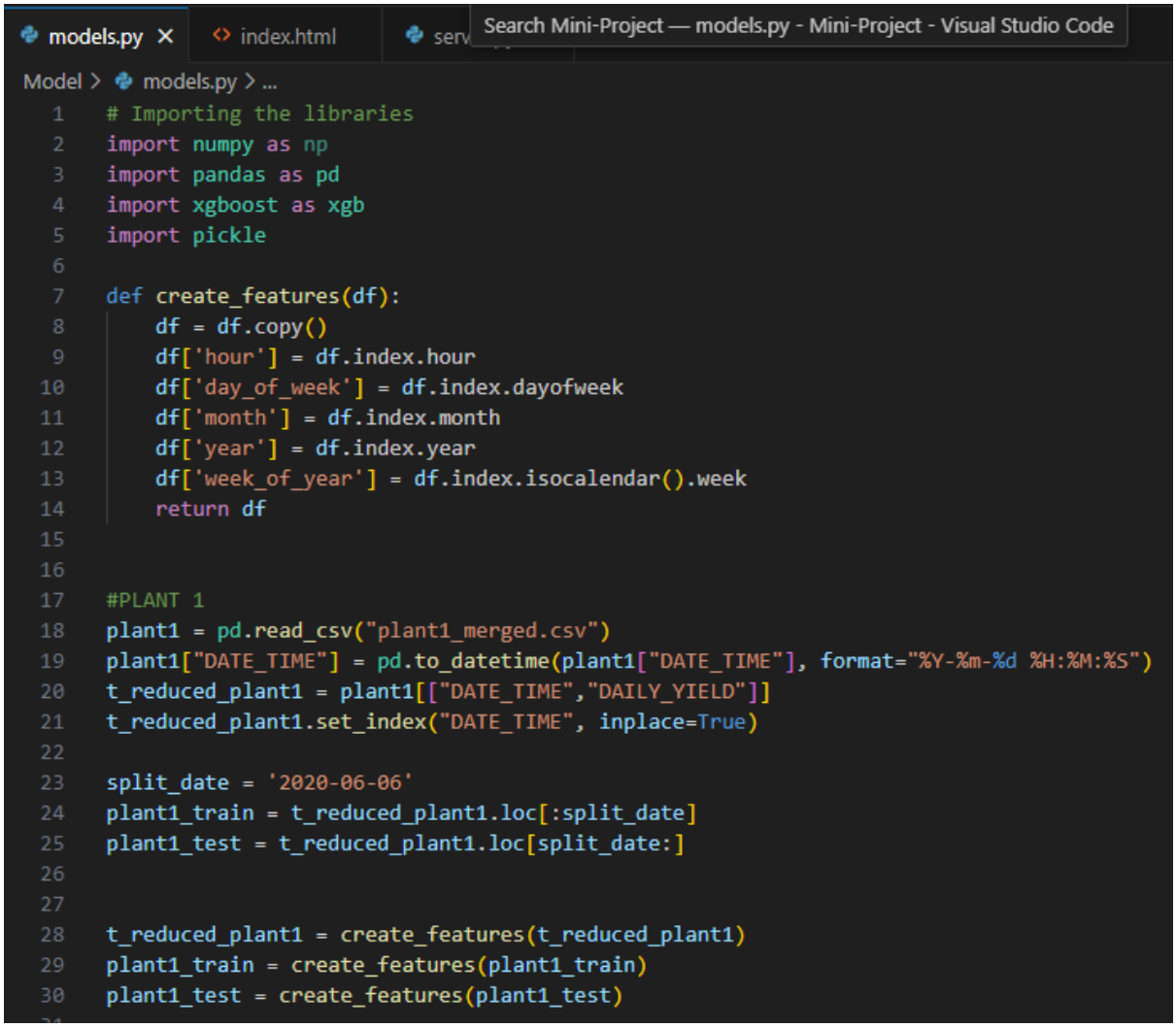
Department of Computer Science and Engineering (Data Science)

Model Deployment:

Code:




```python
# Importing the libraries
import numpy as np
import pandas as pd
import xgboost as xgb
import pickle

def create_features(df):
    df = df.copy()
    df['hour'] = df.index.hour
    df['day_of_week'] = df.index.dayofweek
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['week_of_year'] = df.index.isocalendar().week
    return df


#PLANT 1
plant1 = pd.read_csv("plant1_merged.csv")
plant1["DATE_TIME"] = pd.to_datetime(plant1["DATE_TIME"], format="%Y-%m-%d %H:%M:%S")
t_reduced_plant1 = plant1[["DATE_TIME","DAILY_YIELD"]]
t_reduced_plant1.set_index("DATE_TIME", inplace=True)

split_date = '2020-06-06'
plant1_train = t_reduced_plant1.loc[:split_date]
plant1_test = t_reduced_plant1.loc[split_date:]


t_reduced_plant1 = create_features(t_reduced_plant1)
plant1_train = create_features(plant1_train)
plant1_test = create_features(plant1_test)
```

Department of Computer Science and Engineering (Data Science)

```
X_p1_train_final = plant1_train.iloc[:, 1] # only hour data
y_p1_train_final = plant1_train.iloc[:, 0]

X_p1_test_final = plant1_test.iloc[:, 1]
y_p1_test_final = plant1_test.iloc[:, 0]

reg_final_p1 = xgb.XGBRegressor(n_estimators=1000, learning_rate=0.01)
reg_final_p1.fit(X_p1_train_final, y_p1_train_final)

predictions_final_p1 = reg_final_p1.predict(X_p1_test_final)
pickle.dump(reg_final_p1, open('model1.pkl','wb'))


#PLANT 2
plant2 = pd.read_csv("plant2_merged.csv")
plant2["DATE_TIME"] = pd.to_datetime(plant2["DATE_TIME"], format="%Y-%m-%d %H:%M:%S")

t_reduced_plant2 = plant2[["DATE_TIME","DAILY_YIELD"]]
t_reduced_plant2.set_index("DATE_TIME", inplace=True)

plant2_train = t_reduced_plant2.loc[:split_date]
plant2_test = t_reduced_plant2.loc[split_date:]


t_reduced_plant2 = create_features(t_reduced_plant2)
plant2_train = create_features(plant2_train)
plant2_test = create_features(plant2_test)
```

```
t_reduced_plant2 = create_features(t_reduced_plant2)
plant2_train = create_features(plant2_train)
plant2_test = create_features(plant2_test)

X_p2_train_final = plant2_train.drop(['year', 'month'], axis=1) # using week_of_year, day_of_week, hour
y_p2_train_final = plant2_train.iloc[:, 0]

X_p2_test_final = plant2_test.drop(['year', 'month'], axis=1)
y_p2_test_final = plant2_test.iloc[:, 0]

reg_final_p2 = xgb.XGBRegressor(n_estimators=1000, learning_rate=0.01)
reg_final_p2.fit(X_p2_train_final, y_p2_train_final)

predictions_final_p2 = reg_final_p2.predict(X_p2_test_final)
pickle.dump(reg_final_p2, open('model2.pkl','wb'))

```

Department of Computer Science and Engineering (Data Science)

models.py | index.html | server.py

Model > server.py > ...

```python
from flask import Flask, render_template, request, jsonify
import pickle
import numpy as np
import os

app = Flask(__name__)

# Get the directory of the current script
current_dir = os.path.dirname(os.path.abspath(__file__))

# Load the models
model1_path = os.path.join(current_dir, 'model1.pkl')
model2_path = os.path.join(current_dir, 'model2.pkl')

model1 = pickle.load(open(model1_path, 'rb'))
model2 = pickle.load(open(model2_path, 'rb'))

# Define route for rendering the index.html template
@app.route('/')
def index():
    return render_template('index.html')

# Define prediction endpoint for Model 1
@app.route('/predict_model1', methods=['POST'])
def predict_model1():
    # Get the JSON data from the request
    data = request.get_json()

    print("Received data for Model 1:", data)
    # Extract hour from the JSON data
    hour = int(data['hour'])
```

Department of Computer Science and Engineering (Data Science)

models.py | index.html | server.py

Model > server.py > ...

```python
    # Predict for model 1
    prediction = model1.predict(np.array([[hour]]))[0]

    # Create response JSON
    response = {
        'prediction': prediction.tolist()
    }

    return jsonify(response)

# Define prediction endpoint for Model 2
@app.route('/predict_model2', methods=['POST'])
def predict_model2():
    # Get the JSON data from the request
    data = request.get_json()

    print("Received data for Model 2:", data)
    # Extract features from the JSON data
    hour = int(data['hour'])
    dayOfWeek = int(data['dayOfWeek'])
    weekOfYear = int(data['weekOfYear'])

    # Predict for model 2
    prediction = model2.predict(np.array([[weekOfYear, dayOfWeek, hour]]))[0]

    # Create response JSON
    response = {
        'prediction': prediction.tolist()
    }

    return jsonify(response)
```

Department of Computer Science and Engineering (Data Science)

models.py | index.html | server.py

Model > templates > index.html > html > head > style

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Solar Power Generation Prediction</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 20px;
        }

        h1 {
            text-align: center;
            color: #333;
        }

        form {
            max-width: 400px;
            margin: 0 auto;
            padding: 30px;
            border: 1px solid #ccc;
            border-radius: 5px;
            background-color: #f9f9f9;
        }

        h2 {
            color: #333;
            margin-top: 0;
```

models.py | index.html | server.py

Model > templates > index.html > html > head > style

```
29      h2 {
30          color: #333;
31          margin-top: 0;
32      }
33
34      label {
35          display: block;
36          margin-bottom: 5px;
37          color: #666;
38      }
39
40      input[type="number"] {
41          width: 100%;
42          padding: 10px;
43          margin-bottom: 10px;
44          border: 1px solid #ccc;
45          border-radius: 5px;
46      }
47
48      button[type="submit"] {
49          width: 100%;
50          padding: 10px;
51          background-color: #4caf50;
52          color: #fff;
53          border: none;
54          border-radius: 5px;
55          cursor: pointer;
56          transition: background-color 0.3s;
57      }
58
```

Department of Computer Science and Engineering (Data Science)

models.py | index.html | server.py

Model > templates > index.html > html > body > form#prediction-form2

```
        button[type="submit"]:hover {
            background-color: #45a049;
        }

        #prediction {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
            background-color: #f9f9f9;
        }
    </style>
</head>

<body>
    <h1>Solar Power Generation Prediction</h1>

    <form id="prediction-form1">
        <h2>Plant 1 (Needs Hour)</h2>
        <label for="feature1">Hour:</label>
        <input type="number" id="feature1" name="feature1" required>
        <button type="submit">Predict Plant 1</button>
    </form>

    <form id="prediction-form2">
        <h2>Plant 2 (Needs Hour, Day of Week, Week of Year)</h2>
        <label for="feature2">Hour:</label>
        <input type="number" id="feature2" name="feature2" required>
        <label for="feature3">Day of Week:</label>
        <input type="number" id="feature3" name="feature3" required>
        <label for="feature4">Week of Year:</label>
```

Department of Computer Science and Engineering (Data Science)

models.py | index.html | server.py

Model > templates > index.html > html > body > script > submit() callback

```
87          <label for="feature3">Day of Week:</label>
88          <input type="number" id="feature3" name="feature3" required>
89          <label for="feature4">Week of Year:</label>
90          <input type="number" id="feature4" name="feature4" required>
91          <button type="submit">Predict Plant 2</button>
92      </form>
93
94      <p id="prediction"></p>
95
96      <script>
97          $("#prediction-form1").submit(function (event) {
98              event.preventDefault();
99              const hour = $("#feature1").val();
100             $.ajax({
101                 url: "/predict_model1",
102                 method: "POST",
103                 contentType: "application/json",
104                 data: JSON.stringify({ hour: hour }),
105                 success: function (response) {
106                     $("#prediction").text("Predicted value for Plant 1: " + response.prediction);
107                 },
108                 error: function (jqXHR, textStatus, errorThrown) {
109                     console.error("Error:", textStatus, errorThrown);
110                     $("#prediction").text("An error occurred. Please try again later.");
111                 },
112             });
113         });
114
```

models.py | index.html | server.py

Model > templates > index.html > html > body > script > submit() callback

```
110                     $("#prediction").text("An error occurred. Please try again later.");
111                 },
112             });
113         });
114
115         $("#prediction-form2").submit(function (event) {
116             event.preventDefault();
117             const hour = $("#feature2").val();
118             const dayOfWeek = parseInt($("#feature3").val());
119             const weekOfYear = parseInt($("#feature4").val());
120             $.ajax({
121                 url: "/predict_model2",
122                 method: "POST",
123                 contentType: "application/json",
124                 data: JSON.stringify({ hour: hour, dayOfWeek: dayOfWeek, weekOfYear: weekOfYear }),
125                 success: function (response) {
126                     $("#prediction").text("Predicted value for Plant 2: " + response.prediction);
127                 },
128                 error: function (jqXHR, textStatus, errorThrown) {
129                     console.error("Error:", textStatus, errorThrown);
130                     $("#prediction").text("An error occurred. Please try again later.");
131                 },
132             });
133         });
134
135     </script>
136 </body>
137
138 </html>
139
```

Department of Computer Science and Engineering (Data Science)

Flask App: