

ENERGY CONSUMPTION FORECASTING

TABLE OF CONTENTS

INTODUCTION	3
MACHINE LEARNING OVERVIEW	3
What is Machine Learning?	3
Types of Machine Learning	4
Supervised vs. Unsupervised Learning	4
MODEL SELECTION	5
Why Random Forest?	5
Comparison with Other Models	5
PROJECT WORKFLOW	6
Data Collection	6
Data Preprocessing	6
Feature Engineering	6
DATA DESCRIPTION	6
Dataset Overview	6
Data Sources	7
Feature Description	7
MODEL TRAINING	8
Splitting the Data	8
Training the Model	9
MODEL EVALUATION	9
Evaluation Metrics	9
Model Performance	10
WEB APPLICATION DEVELOPMENT	10
Front-end Design	10
Back-end Implementation	11
Integrating Machine Learning Model	11
RESULTS AND PREDICTIONS	11
ENERGY SAVING TIPS	12
CONCLUSION	12
REFERENCES	12
APPENDICES	13
Source Code	13

INTRODUCTION

The Energy Consumption Forecasting project leverages machine learning to predict energy consumption i.e., Electricity Consumption in Andhra Pradesh. Using historical data, weather patterns, and contextual factors to optimize energy production, distribution, and resource planning. the project aims to provide accurate forecasts for past, present, and future energy usage. This predictive analysis is crucial for efficient energy management, planning, and policy-making. By understanding consumption patterns, stakeholders can make informed decisions to optimize energy distribution and reduce wastage.

The Random Forest model was chosen for this task due to its robustness, accuracy, and ability to handle large datasets with complex interactions between features. The model was trained on historical energy consumption data, alongside weather and economic indicators, to capture the factors influencing energy use. The resulting forecasts can be used to anticipate demand, prepare for peak usage periods, and implement energy-saving measures.

The project also includes a user-friendly web application, allowing users to input specific parameters and receive real-time predictions. This tool is designed to aid policymakers, energy providers, and consumers in making data-driven decisions to enhance energy efficiency and sustainability.

MACHINE LEARNING OVERVIEW

What is Machine Learning?

Machine learning (ML) is a discipline of artificial intelligence (AI) that provides machines with the ability to automatically learn from data and past experiences while identifying patterns to make predictions with minimal human intervention.

Machine learning methods enable computers to operate autonomously without explicit programming. ML applications are fed with new data, and they can independently learn, grow, develop, and adapt.

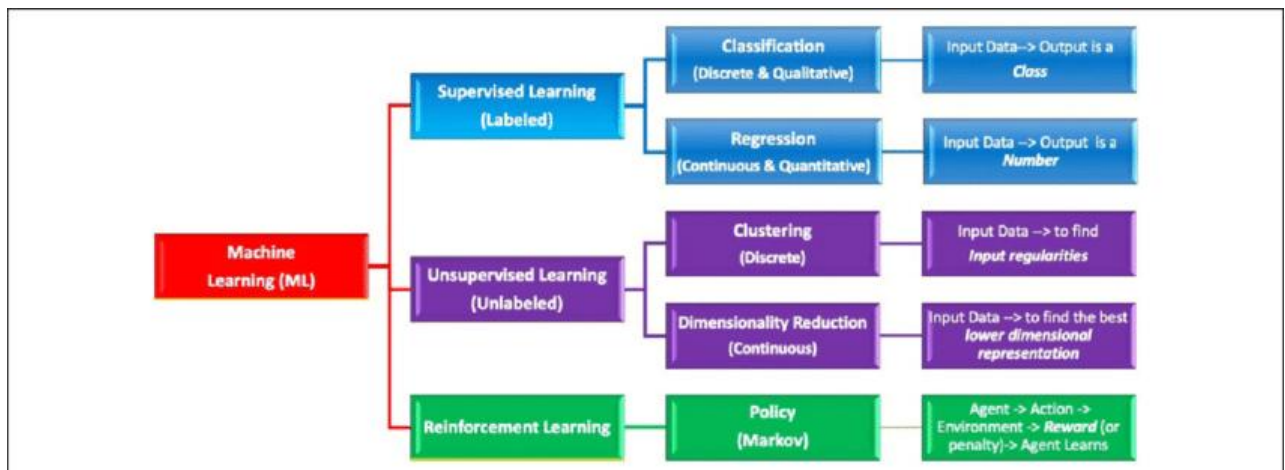


Fig: Machine Learning Overview

Types of Machine Learning

Supervised Learning: The model is trained on a labeled dataset, which means that each training example is paired with an output label. Examples include regression and classification tasks.

Unsupervised Learning: The model is given data without explicit instructions on what to do with it. The goal is to find hidden patterns or intrinsic structures in the input data. Examples include clustering and association.

Reinforcement Learning: The model learns by interacting with its environment, receiving rewards for performing actions that lead to positive outcomes.

Supervised vs. Unsupervised Learning

- Supervised Learning requires labeled data and is typically used for prediction tasks, such as classification (identifying the category of a given input) and regression (predicting a continuous value).
- Unsupervised Learning does not require labeled data and is used for tasks such as clustering (grouping similar items) and dimensionality reduction (reducing the number of random variables).



Supervised vs. Unsupervised Learning

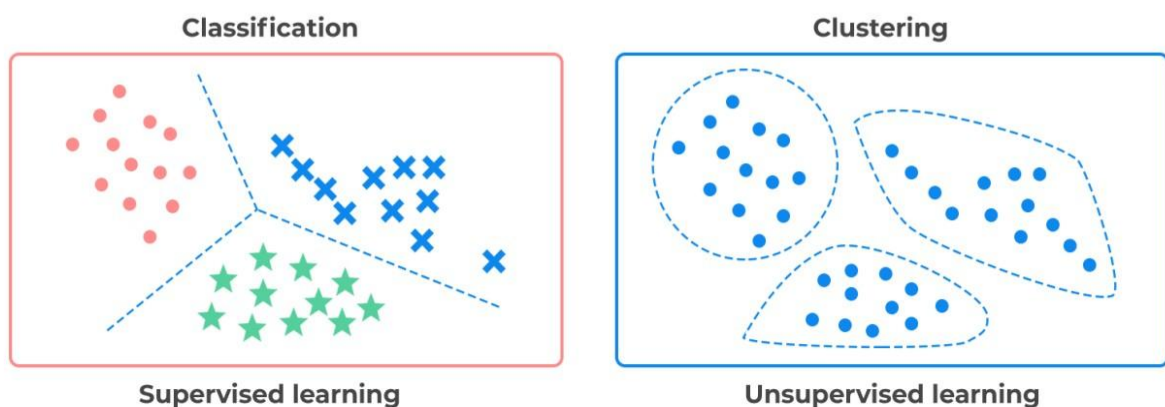


Fig: Supervised vs. Unsupervised Learning

MODEL SELECTION

Why Random Forest?

The Random Forest algorithm was chosen for this project due to its ability to handle large datasets and capture complex interactions between features. Random Forests:

- Combine the predictions of multiple decision trees to improve accuracy.
- Reduce the risk of overfitting compared to individual decision trees.
- Handle both numerical and categorical data effectively.

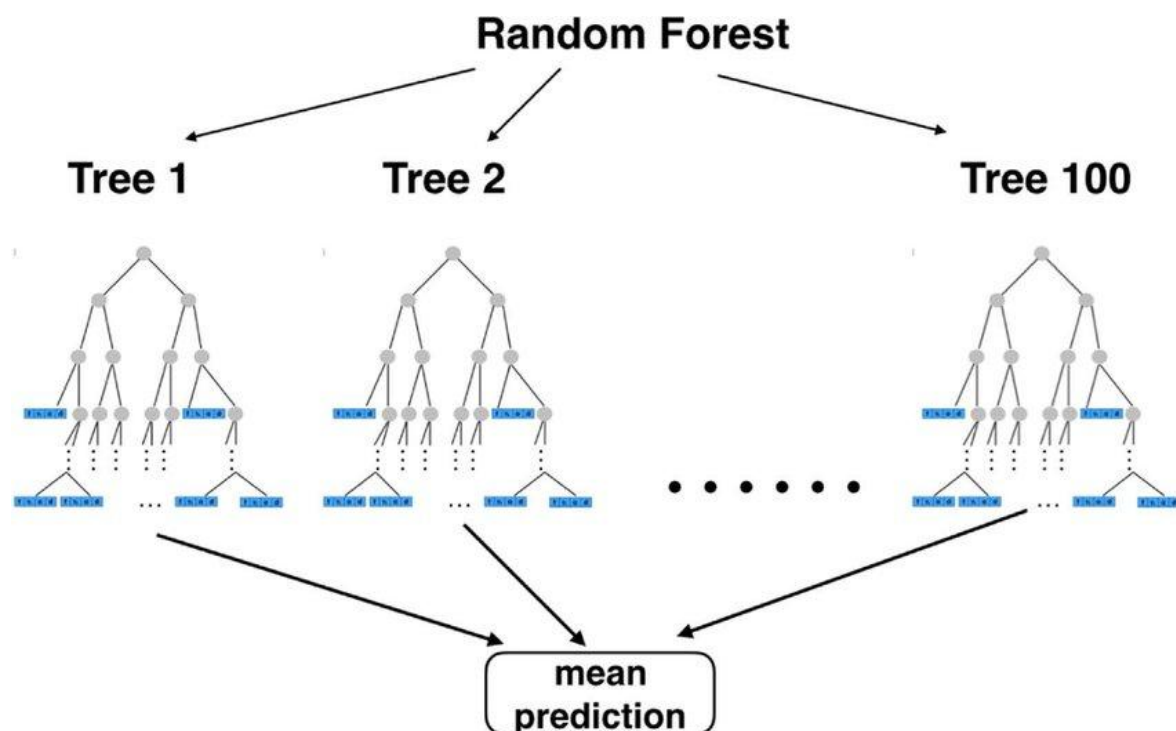


Fig: Random Forest Regressor Algorithm

Comparison with Other Models

Other models considered included:

- **Linear Regression:** Simple and interpretable but may not capture complex relationships.
- **Support Vector Machines (SVM):** Effective for high-dimensional spaces but computationally intensive.
- **Neural Networks:** Powerful but require large amounts of data and computational resources.
- **Gradient Boosting Machines (GBM):** Highly accurate but more prone to overfitting and require careful tuning.

PROJECT WORKFLOW

Data Collection

The dataset was sourced from Kaggle, a well-known platform for data science and machine learning datasets. After extensive searching through various datasets, the Andhra Pradesh Energy Consumption dataset was selected. This dataset provides comprehensive historical energy consumption data specific to the region of Andhra Pradesh.

Link : [Dataset Link](#)

Data Preprocessing

Data preprocessing steps included:

- Handling missing values.
- Normalizing numerical features.
- Encoding categorical variables.
- Splitting the data into training and testing sets.

Feature Engineering

Feature engineering involved creating new features from raw data to improve the predictive power of the model. This included:

- Transforming seasonal data for rainfall and temperature.
- Encoding holidays as a binary feature.
- Creating lag features to capture temporal dependencies.

DATA DESCRIPTION

Dataset Overview

The Andhra Pradesh Energy Consumption dataset, available on Kaggle, encompasses historical data on energy consumption from January 1, 2015, to May 14, 2023. It includes detailed records such as daily energy consumption in megawatt-hours (MWh), alongside various influencing factors like weather conditions (temperature, humidity, wind speed) and economic indicators. This comprehensive dataset facilitates analysis of energy usage patterns and the impact of external factors on energy consumption in Andhra Pradesh.

Data Sources

Data sources include:

- Historical energy consumption records from Andhra Pradesh's energy department.
- Weather data from meteorological departments.
- Economic data from financial reports.

Feature Description

Date: The specific date of the record.

Rain (mm): The amount of rainfall recorded.

Temperature (°C): The temperature recorded.

Inflation Rate: The economic inflation rate.

Is Holiday: A binary indicator of whether the date is a holiday.

data_final							
	Date	Energy Required (MU)	temp	rain	inflation	day	Holiday
0	2015-01-01	131.501	28.000000	0.100000	0.052	Thursday	Work
1	2015-01-02	135.684	28.000000	0.400000	0.052	Friday	Work
2	2015-01-03	136.575	27.000000	1.100000	0.052	Saturday	Work
3	2015-01-04	136.887	27.000000	0.400000	0.052	Sunday	Work
4	2015-01-05	138.566	27.000000	0.000000	0.052	Monday	Work
...
3010	2023-05-10	218.349	37.245505	0.293171	0.043	Wednesday	Work
3011	2023-05-11	219.604	37.272888	0.000000	0.043	Thursday	Work
3012	2023-05-12	224.523	38.218507	0.814892	0.043	Friday	Work
3013	2023-05-13	224.030	38.145824	0.000000	0.043	Saturday	Work
3014	2023-05-14	226.349	39.622750	0.000000	0.043	Sunday	Work

3015 rows × 7 columns

Fig: AP Dataset

MODEL TRAINING

Splitting the Data

To assess the performance of the model accurately, the dataset was divided into training and testing sets. Following common practice, 77% of the data was allocated for training, while the remaining 23% was designated for testing. The `train_test_split` function from the `scikit-learn` library was utilized for this purpose, ensuring a randomized split while maintaining consistency across evaluations. The random state was set to 42 to facilitate reproducibility and consistency in subsequent analyses.

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
```

```
X = final_data.drop(columns = ['Energy Required (MU)'])  
y = final_data['Energy Required (MU)']
```

```
X.head()
```

	temp	rain	inflation	Day_of_Week	Month	Year	Season	Is_Holiday
0	28.0	0.1	0.052	3	1	2015	0	0
1	28.0	0.4	0.052	4	1	2015	0	0
2	27.0	1.1	0.052	5	1	2015	0	0
3	27.0	0.4	0.052	6	1	2015	0	0
4	27.0	0.0	0.052	0	1	2015	0	0

```
y.head()
```

```
0    131.501  
1    135.684  
2    136.575  
3    136.887  
4    138.566
```

```
Name: Energy Required (MU), dtype: float64
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.23, random_state=42)
```

Fig: Data Splitting

Training the Model

The Random Forest model was trained on the training dataset. Key steps included:

- Initializing the model with a set number of trees.
- Fitting the model to the training data.
- Predicting energy consumption on the test set to evaluate performance.

```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)

rf_reg.fit(X_train, y_train)

RandomForestRegressor
RandomForestRegressor(random_state=42)

y_pred = rf_reg.predict(X_test)
```

Fig: Training the Random Forest Regressor

MODEL EVALUATION

Evaluation Metrics

Evaluation metrics used to assess the model's performance included:

- **Mean Squared Error (MSE):** The average squared difference between predicted and actual values.
- **R-squared (R^2):** The proportion of variance in the dependent variable predictable from the independent variables.

Checking for Model Accuracy

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Mean Squared Error: 59.19752572828895

```
r2 = r2_score(y_test, y_pred)
print("R-squared (R2) Score:", r2)
```

R-squared (R2) Score: 0.9045887165325278

Fig: Checking Accuracy

Model Performance

The model's performance was evaluated based on the above metrics, with a focus on minimizing errors and maximizing the R^2 value.

Predicting a new result with Random Forest Regression

```
user_input = [[42,0,0.054,1,6,2024,1,1]]
rf_reg.predict(user_input)

array([224.03913])
```

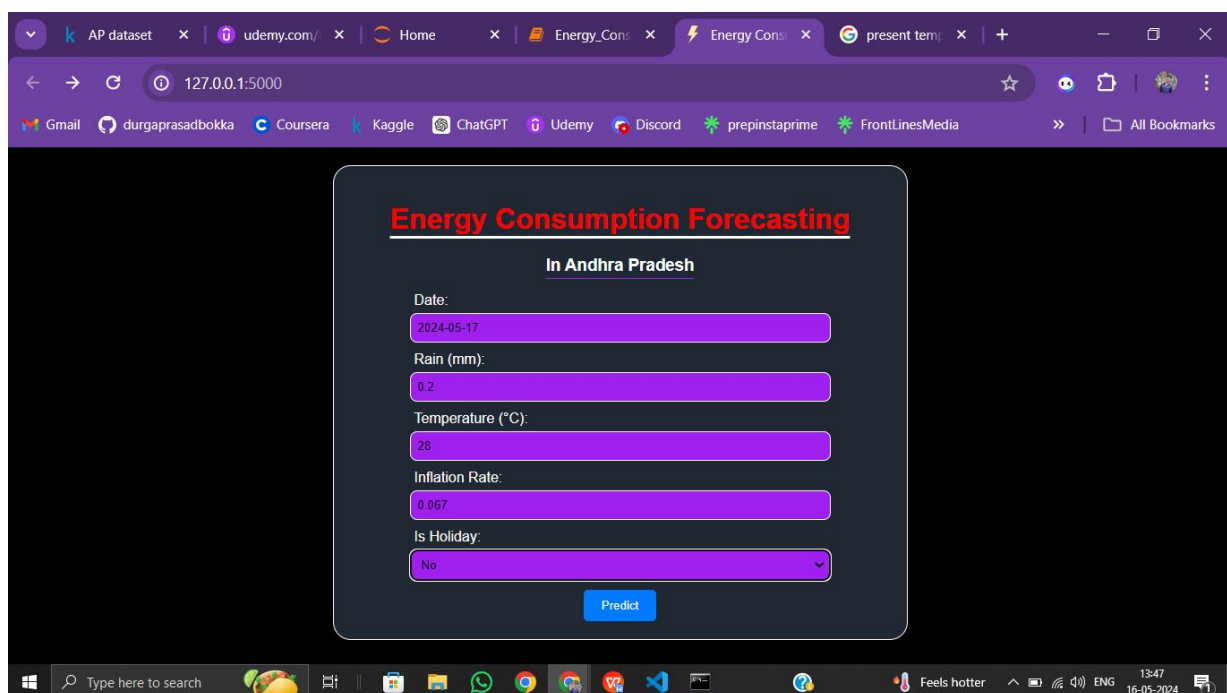
Fig: Model Performance

WEB APPLICATION DEVELOPMENT

Front-end Design

The front-end of the web application was designed using HTML and CSS. Key components included:

- A form for inputting prediction parameters (date, rain, temperature, inflation rate, holiday status).
- Flatpickr for date selection.
- Input validation for numerical fields.



The screenshot shows a web browser window with the URL 127.0.0.1:5000. The page displays a form titled "Energy Consumption Forecasting" with a subtitle "In Andhra Pradesh". The form contains several input fields: "Date" (set to 2024-05-17), "Rain (mm)" (set to 0.2), "Temperature (°C)" (set to 28), "Inflation Rate" (set to 0.067), and "Is Holiday" (set to No). A "Predict" button is located at the bottom of the form. The browser's address bar and tabs are visible at the top, and the Windows taskbar is at the bottom.

Fig: Front-end

Back-end Implementation

The back-end was developed using Flask, a lightweight Python web framework. Key functionalities included:

- Handling form submissions and input validation.
- Loading the trained Random Forest model.
- Making predictions based on user inputs.
- Rendering results on the prediction result page.

Integrating Machine Learning Model

The machine learning model was integrated into the web application to provide real-time predictions. This involved:

- Loading the pre-trained model.
- Processing user inputs to match the model's expected input format.
- Generating predictions and displaying results on the web interface.

RESULTS AND PREDICTIONS

The results from the model are displayed on the web application, showing the predicted energy consumption for the given inputs. The predictions are dynamically generated based on the latest model and data.

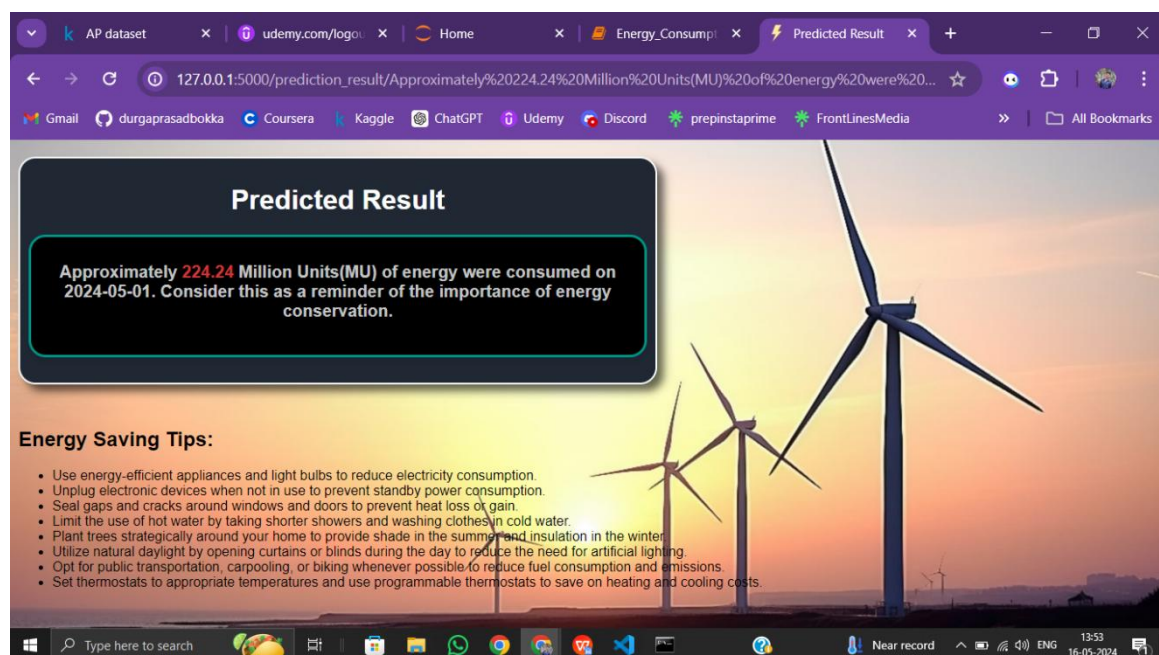


Fig: Result Page

ENERGY SAVING TIPS

The application includes a section on energy-saving tips to educate users on reducing their energy consumption. Tips provided include:

- Using energy-efficient appliances and light bulbs.
- Unplugging devices when not in use.
- Setting thermostats to appropriate temperatures.
- Sealing gaps around windows and doors.
- Limiting hot water use.
- Utilizing natural daylight.
- Opting for public transportation.
- Planting trees for natural insulation.

CONCLUSION

The Energy Consumption Forecasting project successfully predicts future energy consumption i.e., Electricity consumption in Andhra Pradesh using a Random Forest model. The integration of a user-friendly web application allows for easy access to predictions and promotes energy-saving practices. This project demonstrates the effectiveness of machine learning in addressing real-world problems and highlights the importance of efficient energy management.

REFERENCES

Include references to all data sources, libraries, and tools used in the project. Examples:

- Scikit-learn documentation for Random Forest implementation.
- Flask documentation for web development.
- Data sources for historical energy consumption, weather, and economic data.

APPENDICES

Source Code

In this section, we include the full source code used for the project. This comprises the main Python script for training and predicting energy consumption using the Random Forest model, as well as the code for the web application backend.

main.py

```
from flask import Flask, render_template, request, redirect, url_for
import joblib
from datetime import datetime
import re

app = Flask(__name__)
```

```
# Load the pre-trained energy consumption prediction model
model = joblib.load('model/energy_consumption_model.pkl')
```

```
# Home page
```

```
@app.route('/')
```

```
def index():
    return render_template('index.html')
```

```
# Define the route to handle prediction requests
```

```
@app.route('/predict', methods=['POST'])
```

```
def predict():
    if request.method == 'POST':
        # Get the current date
        current_date = datetime.now().date()

        # Get the input data from the form
        date_str = request.form['date']
        date = datetime.strptime(date_str, '%Y-%m-%d').date()
        rain = float(request.form['rain'])
        temp = float(request.form['temp'])
```

```

inflation = float(request.form['inflation'])
holiday = str(request.form['holiday'])

# Convert holiday input to a binary variable
if holiday == 'yes':
    is_holiday = 1
else:
    is_holiday = 0

```

```

day_of_week = date.weekday()
month = date.month
year = date.year

# Function to determine the season based on the month
def get_season(month):
    if month in [10, 11, 12, 1, 2]:
        return 0 # Winter
    elif month in [3, 4, 5, 6]:
        return 1 # Summer
    else:
        return 2 # Rainy

season = get_season(month)

# Make the energy consumption prediction using the model
predicted_consumption = model.predict([[temp, rain, inflation, day_of_week, month,
year, season, is_holiday]])
consumption = round(predicted_consumption[0], 2)

# Generate an appropriate message based on whether the date is past, future, or today
if date < current_date:
    message = f"Approximately {consumption} Million Units(MU) of energy were
consumed on {date.strftime('%Y-%m-%d')}. Consider this as a reminder of the importance of
energy conservation."
elif date > current_date:
    message = f"Based on current projections, approximately {consumption} Million
Units(MU) of energy will be required on {date.strftime('%Y-%m-%d')}. Let's plan ahead and
make conscious efforts to conserve energy."
else:
    message = f"Today, approximately {consumption} Million Units(MU) of energy
are projected to be required. This serves as a call to action for us all to utilize energy wisely
and adopt energy-saving practices."

# Redirect to the prediction result page with the message
return redirect(url_for('prediction_result', prediction=message))

```

```

# Define the route to display the prediction result
@app.route('/prediction_result/<prediction>')

```

```
def prediction_result(prediction):
    # Use regex to find the numerical part of the prediction
    match = re.search(r'\d+(\.\d+)?', prediction)
    if match:
        consumption_value = match.group()
        # Highlight the consumption value in the prediction message
        highlighted_prediction = prediction.replace(consumption_value, f'<span
class="consumption">{consumption_value}</span>')
        return render_template('prediction_result.html', prediction=highlighted_prediction)
    else:
        return render_template('prediction_result.html', prediction=prediction)
```

```
# Run the Flask app in debug mode
if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Energy Consumption Forecasting</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/flatpickr/dist/flatpickr.min.css">
    <link rel="icon" href="{{ url_for('static',
filename='Images/favicon.ico') }}">
</head>
<body>
    <div class="container">
        <h1>Energy Consumption Forecasting</h1>
        <h3>In Andhra Pradesh</h3>
        <form id="prediction-form" action="/predict" method="post">
            <div class="form-group">
                <label for="date">Date:</label>
                <input type="text" id="date" name="date" placeholder="Select
Date" required>
            </div>
            <div class="form-group">
                <label for="rain">Rain (mm):</label>
                <input type="number" id="rain" name="rain" placeholder="For
Future prediction please give as per the Season" step="0.01" required>
            </div>
```

```

        <div class="form-group">
            <label for="temp">Temperature (°C):</label>
            <input type="number" id="temp" name="temp" placeholder="For
Future prediction please give as per the Season" step="0.1" required>
        </div>
        <div class="form-group">
            <label for="inflation">Inflation Rate:</label>
            <input type="number" id="inflation" name="inflation"
placeholder="For Future prediction please give as per present rate"
step="0.001" required>
        </div>
        <div class="form-group">
            <label for="holiday">Is Holiday:</label>
            <select id="holiday" name="holiday" required>
                <option value="none">-none-</option>
                <option value="yes">Yes</option>
                <option value="no">No</option>
            </select>
        </div>
        <button type="submit">Predict</button>
    </form>
</div>
<script src="https://cdn.jsdelivr.net/npm/flatpickr"></script>
<script>
    let calendarInstance = null;

    document.addEventListener('DOMContentLoaded', function () {
        calendarInstance = flatpickr("#date", {
            enableTime: false,
            dateFormat: "Y-m-d",
            clickOpens: false
        });
        document.getElementById('date').addEventListener('click', function
() {
            if (calendarInstance.isOpen) {
                calendarInstance.close();
            } else {
                calendarInstance.open();
            }
        });
    });
</script>
</body>
</html>

```


prediction_result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Predicted Result</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
  <link rel="icon" href="{{ url_for('static',
filename='Images/favicon.ico') }}">
</head>
<body id="bg-img">
  <div class="container2">
    <h1 id='result'>Predicted Result</h1>
    <div class="result">
      <p>{{ prediction | safe }}</p>
    </div>
  </div>
  <div class="energy-saving-container">
    <div class="energy-saving-tips">
      <h2>Energy Saving Tips:</h2>
      <ul>
        <li>Use energy-efficient appliances and light bulbs to reduce
electricity consumption.</li>
        <li>Unplug electronic devices when not in use to prevent
standby power consumption.</li>
        <li>Seal gaps and cracks around windows and doors to prevent
heat loss or gain.</li>
        <li>Limit the use of hot water by taking shorter showers and
washing clothes in cold water.</li>
        <li>Plant trees strategically around your home to provide
shade in the summer and insulation in the winter.</li>
        <li>Utilize natural daylight by opening curtains or blinds
during the day to reduce the need for artificial lighting.</li>
        <li>Opt for public transportation, carpooling, or biking
whenever possible to reduce fuel consumption and emissions.</li>
      </ul>
    </div>
  </div>
</body>
</html>
```

```

        <li>Set thermostats to appropriate temperatures and use
programmable thermostats to save on heating and cooling costs.</li>
    </ul>
</div>
</div>
</body>
</html>

```

styles.css

```

body {
    font-family: Arial, sans-serif;
    background-color: #000000;
}

.container {
    max-width: 600px;
    margin: 20px auto;
    padding: 20px;
    border: 1px solid #ffffff;
    border-radius: 20px;
    background-color: #1f2833;
    position: relative;
}

```

```

.container h1 {
    color: #ff0000;
    text-align: center;
    text-decoration: underline;
    text-decoration-color: white;
    text-decoration-skip-ink: none;
    text-underline-offset: 8px;
    animation: transform 0.3s ease;
}

```

```

.container h1:hover{
    transform: scale(1.05);
}

```

```

.container h3{
    text-align: center;
    color: #ffffff;
    text-decoration: underline;
    text-decoration-color: #A020F0;
}

```

```
text-decoration-skip-ink: none;
text-underline-offset: 8px;
```

```
}
```

```
.container2 {
  max-width: 730px;
  margin: 20px auto;
  padding: 10px;
  border: 2px solid #ffffff;
  border-radius: 20px;
  background-color: #1f2833;
  position: absolute;
  top: 0;
  left: 10px;
  box-shadow: 8px 8px 10px rgba(0, 0, 0, 0.6);
}
```

```
.energy-saving-container{
  max-width: 100%;
  margin: 20px auto;
  padding: 10px;
  position: absolute;
  top: 290px;
  left: 0;
}
```

```
label {
  display: block;
  margin-bottom: 5px;
  margin-left: 70px;
}
.form-group label{
  color: white;
}
input, select {
  width: calc(80% - 10px);
  margin: 0 auto 10px;
  padding: 8px;
  background-color: #A020F0;
  border: 1px solid #ffffff;
  border-radius: 8px;
  display: block;
  box-sizing: border-box;
}
```

```
input:hover, select:hover {
```

```
background-color: #fff;
border-color: #A020F0;
outline: none;
}
input::placeholder {
  color: #393838;
}
```

```
button {
  padding: 10px 20px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  display: block;
  margin: 0 auto;
  animation: transform 0.3s ease;
}
```

```
button:hover {
  transform: scale(1.05);
}
```

```
button:active {
  transform: scale(0.95);
}
```

```
.form-group:last-child {
  margin-bottom: 20px;
}
```

```
.result {
  background-color: #000000;
  padding: 20px;
  border-radius: 20px;
  border: 3px solid #009688;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  animation: fadeIn 0.5s ease;
  margin: 20px auto;
  margin-top: 20px;
}
@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(-20px);
  }
}
```

```
    to {
      opacity: 1;
      transform: translateY(0);
    }
  }
  .result h2 {
    color: #fc0000;
    text-align: center;
  }
  .result p {
    font-weight: bold;
    font-size: 20px;
    color: #c5c6c7;
    margin-top: 10px;
    text-align: center;
  }
  .result p .consumption {
    color: rgb(208, 57, 57);
    font-weight: bold;
    font-size: 20px;
  }
}
```

```
#result{
  text-align: center;
  color: #ffffff;
}
```

```
#bg-img{
  background-image: url('/static/Images/bg.jpg');
  background-color: #000000;
  background-size: cover;
  background-repeat: no-repeat;
  height: 100vh;
  margin: 0;
  padding: 0;
}
```

The End

-----@-----