LAB MANUAL

Unit III – Machine Learning

Unit III – Machine Learning

Lab 1. Predicting Solar Power Output Using Linear Regression

Objective

To implement solar power output prediction using Linear Regression, a general approach to how you can tackle the solar power output prediction problem using linear regression, along with a sample implementation using Python and common libraries like Pandas, Scikit-learn, and Matplotlib. The model will be trained on historical data and will predict the solar power output for given values of the input features. The performance of the model will be evaluated using standard metrics such as Mean Squared Error (MSE) and R-squared (R^2).

Problem

The global shift towards sustainable energy has led to an increasing reliance on renewable energy sources such as solar power. Accurate predictions of solar power output are crucial for effective grid management, energy storage, and optimizing the use of solar energy. we aim to predict the solar power output based on certain environmental factors, which include temperature, humidity, solar_irradiance and wind_speed. The goal is to develop a predictive model using Linear Regression, a simple yet powerful algorithm, to estimate the solar power output from these features.

Solution

To predict solar power output using linear regression and save the model, we'll go through the following steps:

1. Import required libraries
2. Prepare the dataset
    1. Load dataset
    2. Select Features and target variable
3. Train a linear regression model
    1. Split the dataset into training and testing sets (80% train, 20% test)
    2. Initialize and train the linear regression model

4. Evaluate the model
    1. Predicting the solar power output on the test set
    2. Evaluate the model performance
5. Visualization
6. Save the model for later use
7. Load and use the Model

Procedures

1. Import required libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import joblib
```

2. Prepare the dataset

Features:

- temperature: in the range of 10-35°C to represent different climates.
- humidity: In between 20% and 100% as humidity varies across regions.
- solar_irradiance: In between 100 W/m² and 1000 W/m² (typical range for solar radiation).
- wind_speed: In between 0 m/s and 10 m/s, representing wind conditions.

Target Variable (solar_power_output): This is calculated as a function of the other features, with random noise added to make the data more realistic. The relationship assumes that higher irradiance and temperature increase power output, while high humidity reduces efficiency. Wind speed has a small positive impact.

2.1 Load Dataset

```
df = pd.read_csv('solar_power_output.csv')
df.head()
```

	temperature	humidity	solar_irradiance	wind_speed	solar_power_output
0	19.363503	75.852937	266.619636	5.190818	128.101772
1	33.767858	62.887709	587.710853	4.791819	290.911789
2	28.299849	44.762209	885.651252	0.256421	442.336390
3	24.966462	85.103602	759.002398	3.412478	380.261988
4	13.900466	74.778494	825.905033	3.801956	415.931953

2.2 Select Features and target variable

```
# Features and target variable
X = df[['temperature','humidity','solar_irradiance','wind_speed']]
y = df['solar_power_output']
```

3. Train a linear regression model

Now, we'll split the dataset into training and testing sets, train the linear regression model, and evaluate its performance.

3.1 Split the dataset into training and testing sets (80% train, 20% test)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.2 Initialize and train the linear regression model

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

4. Evaluate the model

We will now predict the solar power output using the test data and evaluate the performance using metrics like Mean Squared Error (MSE) and R-squared (R^2).

4.1 Predicting the solar power output on the test set

```
# Predicting the solar power output on the test set
y_pred = model.predict(X_test)
```

Predicted outputs of test dataset,

```
y_pred
```

```
array([145.57713107,  58.80813607, 162.02529472, 389.82145928,
       499.91950594, 201.0282157 , 280.32773311, 343.94995963,
       207.65845171, 274.08855807, 270.4758993 ,  71.45181397,
       429.2717113 , 474.93452177, 131.00879232, 488.72382531,
       351.97648481,  66.01156114, 286.53039285, 315.26996554,
       156.25649583, 217.68415494, 328.50926609, 456.09724416,
       224.5996492 , 343.33441394, 258.65661936, 253.27035924,
       487.33886964, 488.13938702, 352.87341423, 139.35221981,
       486.2457442 , 166.59050109, 443.99714231, 498.93693591,
       440.23818941, 184.21480104, 497.81182991, 239.22808162,
       345.26650268, 281.34107991, 215.84679473, 472.53738059,
       105.19383876, 353.92961078, 132.62179771, 496.30985098,
       472.03357268, 464.32390435,  68.39576647, 337.05784428,
       271.45779664, 142.00782194, 330.57420011, 209.16463829,
       431.33413852, 460.5254782 , 271.68335513, 148.17216826,
       322.32496321, 189.54364635, 148.80585848,  54.32626697,
       477.68708932, 193.09857442, 327.78450176, 155.94903389,
       486.49330331,  81.90369754, 249.1025331 , 257.83534344,
       494.00652549, 228.57425188, 379.22282528, 370.0548375 ,
       320.35938503, 181.0579938 , 469.88218569,  72.05722784,
       329.76422063, 237.83607462, 394.78456359, 336.13041583,
       123.75761569, 474.89224919, 482.2744928 , 289.41809558,
       296.55227317, 113.97247379, 226.40379581, 443.74657838,
       330.25021256, 490.55448237, 281.92908425, 214.43598619,
       328.26740398, 241.67895954, 142.67690408, 467.08839039])
```

4.2 Evaluate the model performance

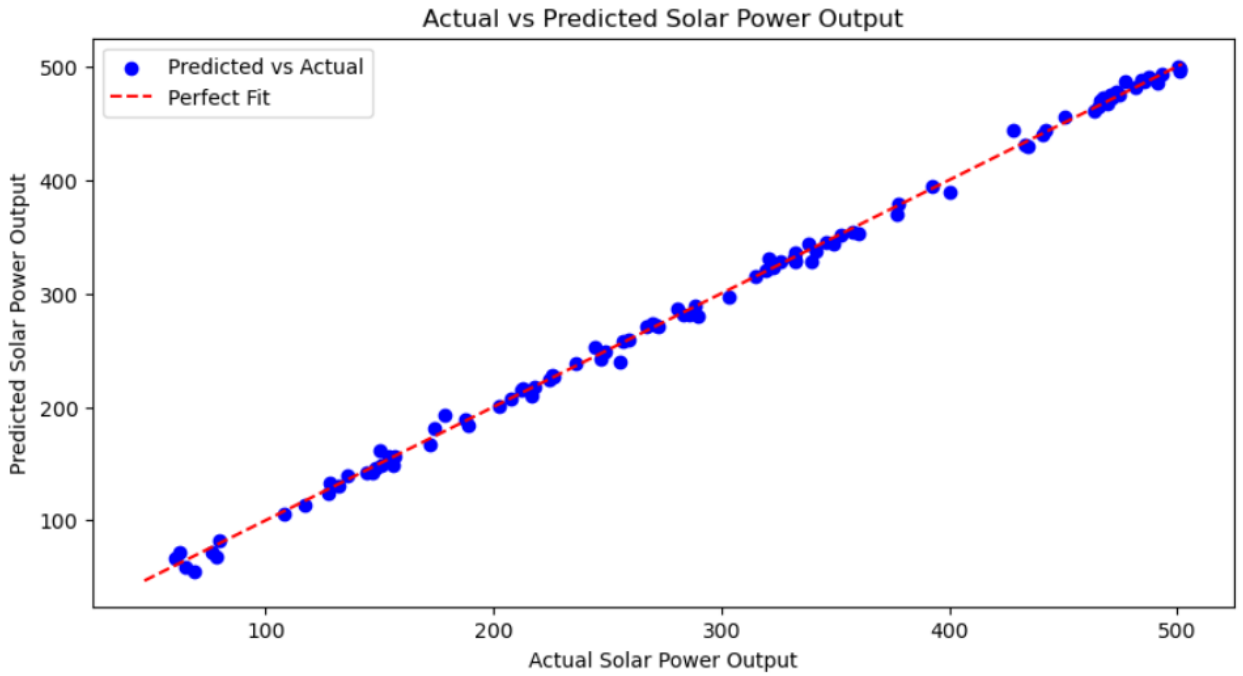
```
# Evaluate the model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 29.462345968246467
R-squared: 0.9983255901416407
```

5. Visualization

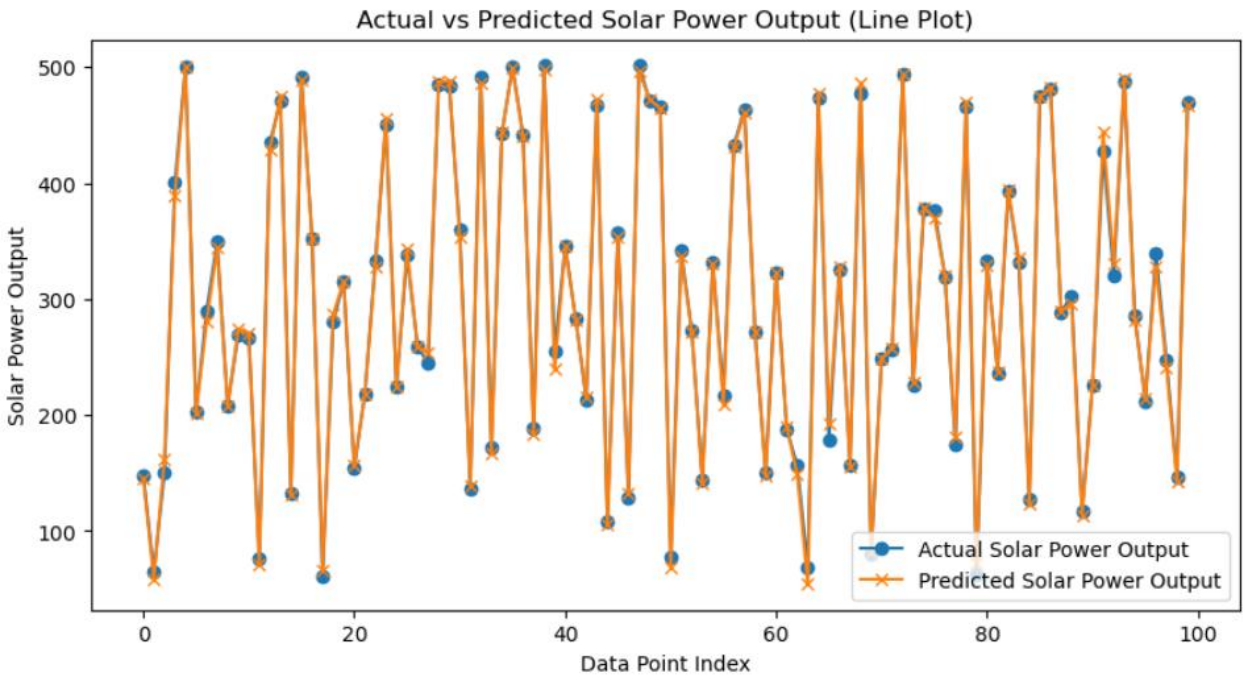
```
import matplotlib.pyplot as plt

# Scatter plot to compare actual vs predicted values
plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
# Diagonal line for perfect predictions
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--', label='Perfect Fit')
plt.xlabel("Actual Solar Power Output")
plt.ylabel("Predicted Solar Power Output")
plt.title("Actual vs Predicted Solar Power Output")
plt.legend()
plt.show()
```



- Each blue dot represents a comparison between an actual and predicted value.
- The red dashed line represents a perfect fit line, where predicted values would exactly match actual values if the model were perfect.
- The closer the points are to this line, the better the model's accuracy.

```
# Line plot to show predictions and actual values
plt.figure(figsize=(10, 5))
plt.plot(range(len(y_test)), y_test, label="Actual Solar Power Output", marker='o')
plt.plot(range(len(y_test)), y_pred, label="Predicted Solar Power Output", marker='x')
plt.xlabel("Data Point Index")
plt.ylabel("Solar Power Output")
plt.title("Actual vs Predicted Solar Power Output (Line Plot)")
plt.legend()
plt.show()
```



- Plots both actual and predicted solar power outputs for each test data point index.
- Helps visualize how predictions follow (or deviate from) the actual trend across the dataset.

6. Save the model for later use

```
# Save the trained model
joblib.dump(model, 'solar_power_prediction_model.pkl')
```

```
['solar_power_prediction_model.pkl']
```

7. Load and use the Model

```
# Load the saved model
model = joblib.load('solar_power_prediction_model.pkl')

# New input data for prediction
# Example input (temperature,humidity,solar_irradiance,wind_speed)
new_data = np.array([[19.36, 75.85,266.61,5.19]])

# Predict solar power output
predicted_output = model.predict(new_data)
print(f"Predicted Solar Power Output: {predicted_output[0]} watts")
```

```
Predicted Solar Power Output: 132.6167057318902 watts
```