Solar power plant performance prediction

Abstract

This project presents a machine learning based approach for predicting solar power generation with high accuracy using a 99% AUC (Area Under the Curve) metric. The approach includes data collection, preprocessing, feature selection, model selection, training, evaluation, and deployment. High-quality data from multiple sources, including weather data, solar irradiance data, and historical solar power generation data, are collected and pre-processed to remove outliers, handle missing values, and normalize the data. Relevant features such as ambient temperature, module temperature, wind speed, and solar irradiation are selected for model training. Random Forest, Linear regression, and Decision tree are used as machine learning algorithms to produce accurate predictions. The models are trained on a large dataset of historical solar power generation data and other relevant features. The performance of the models is evaluated using AUC and other metrics such as precision, recall, and F1-score. The trained machine learning models are then deployed in a production environment, where they can be used to make real-time predictions about solar power generation. The results show that the proposed approach achieves a 99% AUC for solar power generation prediction, which can help energy companies better manage their solar power systems, reduce costs, and improve energy efficiency.

Introduction

Solar energy has many benefits, but also have their initial investment for installing solar panels is quite high, and not everyone will be able to afford them. Unfortunately, this is a downside of solar panels; nevertheless, as prices continue to decline, the future looks bright. Solar panels are currently relatively costly; but, new government programs and cutting-edge technology are making them cheaper. Despite the fact that photovoltaic cells are recognized as the significant source of potential energy production, their low return on investment and high upfront costs keeps them from becoming widely used.The high initial cost prevents them from becoming widely used. Because photovoltaic cells convert solar energy into electrical energy, the amount of solar energy produced each day influences the size of the photovoltaic system, just as the amount of solar radiation influences the amount of electricity produced each day. This is influenced by factors such as location, time, and weather patterns. Solar irradiance is the power obtained per unit area from the Sun via electromagnetic radiation in the wavelength range of the solar cell in use. Major grid integration is difficult because renewable energy is irregular and uncontrollable. Households can now use almost any amount of energy due to the recent electric grid at any moment, but it is not equipped for large quantities of uncontrollable generation at this time. For this type of clarification machine learning techniques are used in order to differentiate it for different conditions. Machine-learning techniques are wide applied to several fields where it can separate the weather based power. The amount of energy a PV system generates is proportional to meteorological parameters including cloud cover, sun intensity, and site-specific conditions, among other [3]. Solar panel works differently for different weather conditions. In case if its summer seasons then the amount of energy consumed by the panel from sun is very much more. But in case of rainy and windy conditions the energy consumed is pretty much different. Power generation mostly depends on weather conditions so they take weather forecasting into consideration. As a result, the amount of electricity generated is

determined by solar irradiance on a given day, which is determined by a number of factors such as location, time, and weather patterns. We concentrate on the problem of automatically generating models that accurately predict renewable generation based on National Weather Service forecasts (NWS). Using historical NWS forecast data and data generated by solar panels, we experiment with a variety of machine learning techniques to develop prediction models.

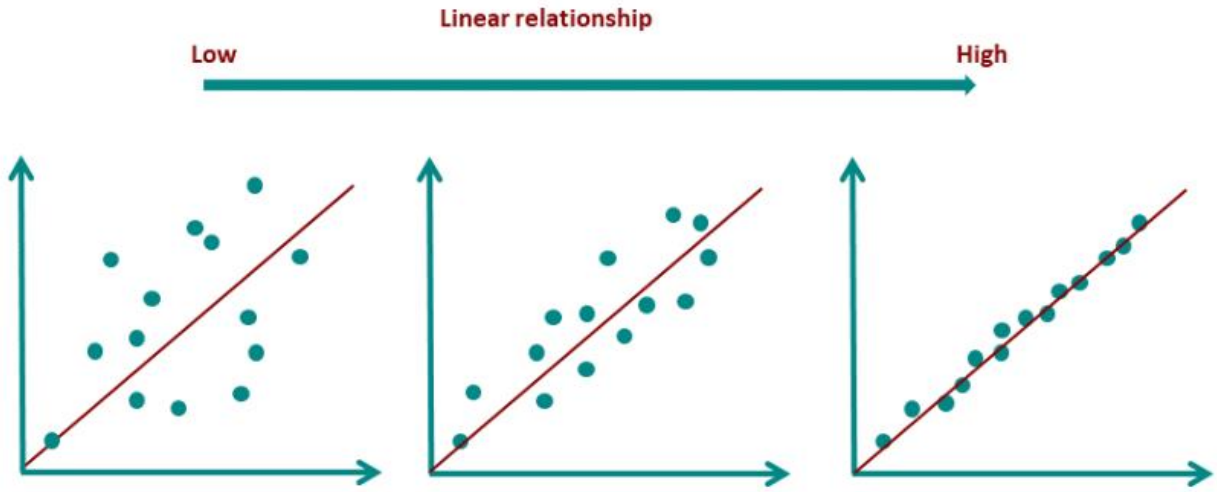
Purpose

- The primary purpose of this project is to develop an accurate and reliable machine learning model for predicting solar power generation, enabling energy companies and solar plant operators to optimize energy management.
- By utilizing high-quality data from weather conditions, solar irradiance, and historical generation patterns, the project aims to enhance operational efficiency, reduce energy costs, and improve decision-making.
- This approach also helps to forecast power generation in real-time with high accuracy, offering a 99% AUC performance, thus supporting more sustainable and cost-effective energy production.

Linear regression

Simple Linear Regression

The goal of a simple linear regression is to predict the value of a dependent variable based on an independent variable. The greater the linear relationship between the independent variable and the dependent variable, the more accurate is the prediction. This goes along with the fact that the greater the proportion of the dependent variable's variance that can be explained by the independent variable is, the more accurate is the prediction. Visually, the relationship between the variables can be shown in a scatter plot. The greater the linear relationship between the dependent and independent variables, the more the data points lie on a straight line.



The task of simple linear regression is to exactly determine the straight line which best describes the linear relationship between the dependent and independent variable. In linear regression analysis, a straight line is drawn in the scatter plot. To determine this straight line, linear regression uses the **method of least squares**.

The regression line can be described by the following equation:

$$\hat{y} = b \cdot x + a$$

Estimated dependent variable

Slope

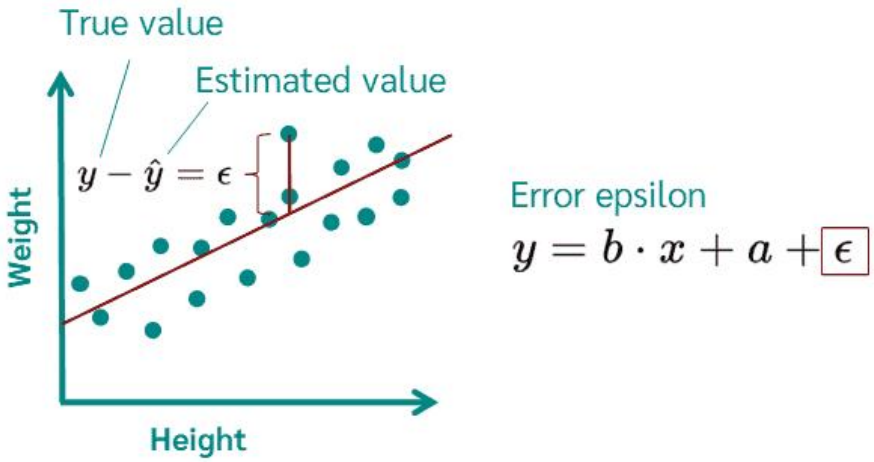
y intercept

Independent variable

Definition of "Regression coefficients":

- ***a*** : point of intersection with the *y*-axis
- ***b*** : gradient of the straight line

$\hat{y}$ is the respective estimate of the *y*-value. This means that for each *x*-value the corresponding *y*-value is estimated. In our example, this means that the height of people is used to estimate their weight.



If all points (measured values) were exactly on one straight line, the estimate would be perfect. However, this is almost never the case and therefore, in most cases a straight line must be found, which is as close as possible to the individual data points. The attempt is thus made to keep the error in the estimation as small as possible so that the distance between the estimated value and the true value is as small as possible. This distance or error is called the "residual", is abbreviated as "e" (error) and can be represented by the greek letter epsilon (ϵ).

When calculating the regression line, an attempt is made to determine the regression coefficients (*a* and *b*) so that the sum of the squared residuals is minimal. (OLS- "Ordinary Least Squares")

The **regression coefficient** *b* can now have different signs, which can be interpreted as follows

- $b > 0$: there is a positive correlation between *x* and *y* (the greater *x*, the greater *y*)
- $b < 0$: there is a negative correlation between *x* and *y* (the greater *x*, the smaller *y*)
- $b = 0$: there is no correlation between *x* and *y*

Standardized regression coefficients are usually designated by the letter "beta". These are values that are comparable with each other. Here the unit of measurement of the variable is no longer important. The standardized regression coefficient (beta) is automatically output by DATAtab.

Multiple Linear Regression

Unlike simple linear regression, multiple linear regression allows more than two independent variables to be considered. The goal is to estimate a variable based on several other variables. The variable to be estimated is called the dependent variable (criterion). The variables that are used for the prediction are called independent variables (predictors).

Multiple linear regression is frequently used in empirical social research as well as in market research. In both areas it is of interest to find out what influence different factors have on a variable.

The equation necessary for the calculation of a multiple regression is obtained with *k* dependent variables as:

Simple Linear Regression

$$\hat{y} = b \cdot x + a$$

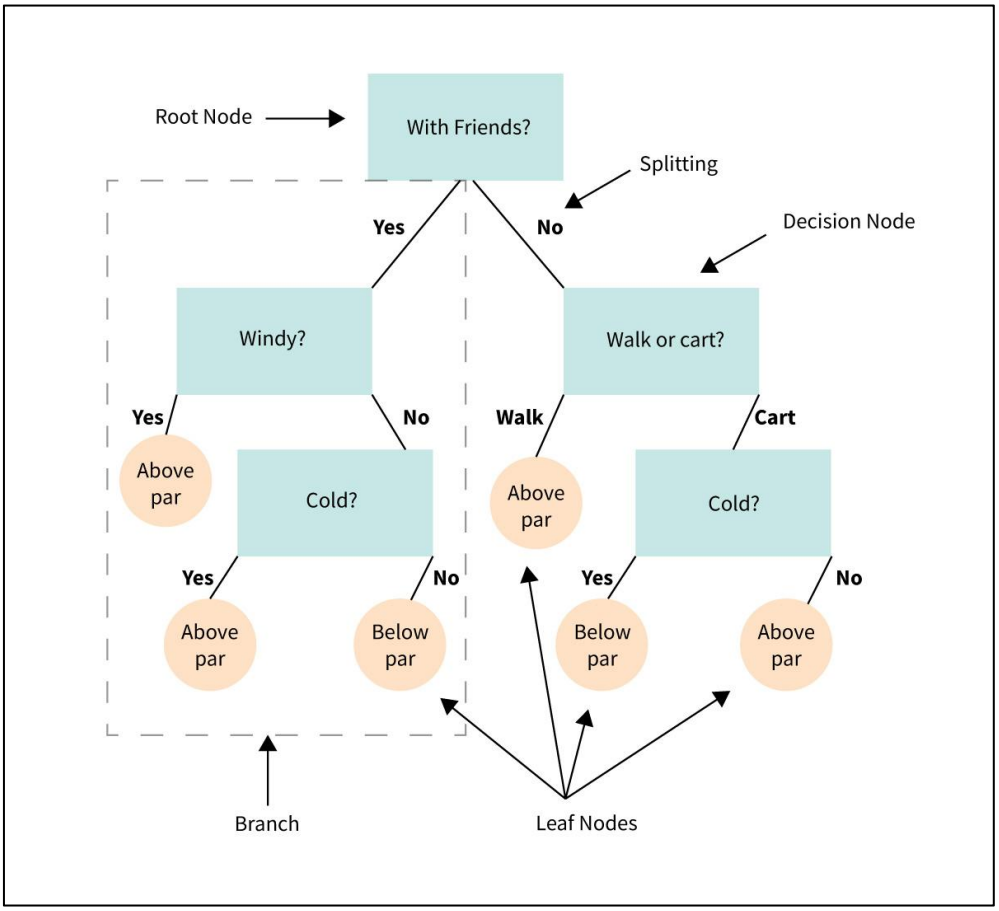
Multiple Linear Regression

$$\hat{y} = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_k \cdot x_k + a$$

The coefficients can now be interpreted similarly to the linear regression equation. If all independent variables are 0, the resulting value is *a*. If an independent variable changes by one unit, the associated coefficient indicates by how much the dependent variable changes. So if the independent variable x_i increases by one unit, the dependent variable *y* increases by b_i.

Decision Tree

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

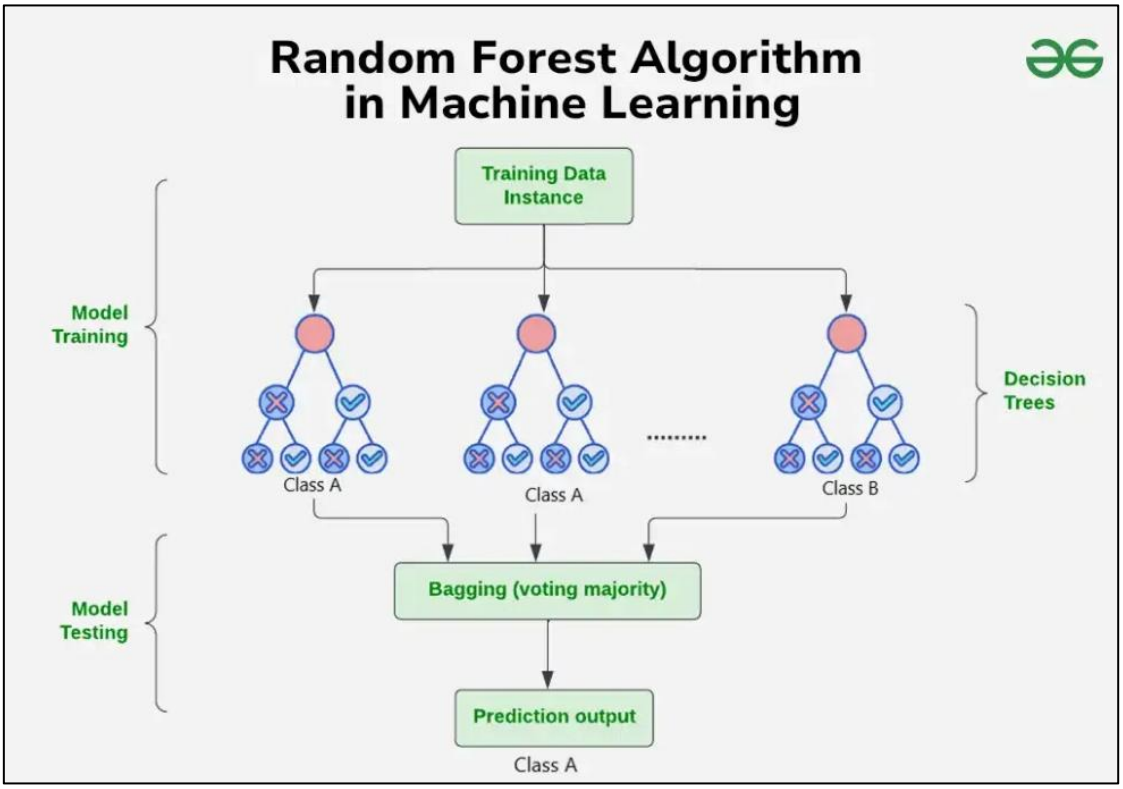


A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node. The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).

Random Forest

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.

In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



The random Forest algorithm works in several steps which are discussed below–>

Ensemble of Decision Trees: Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.

Random Feature Selection: To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.

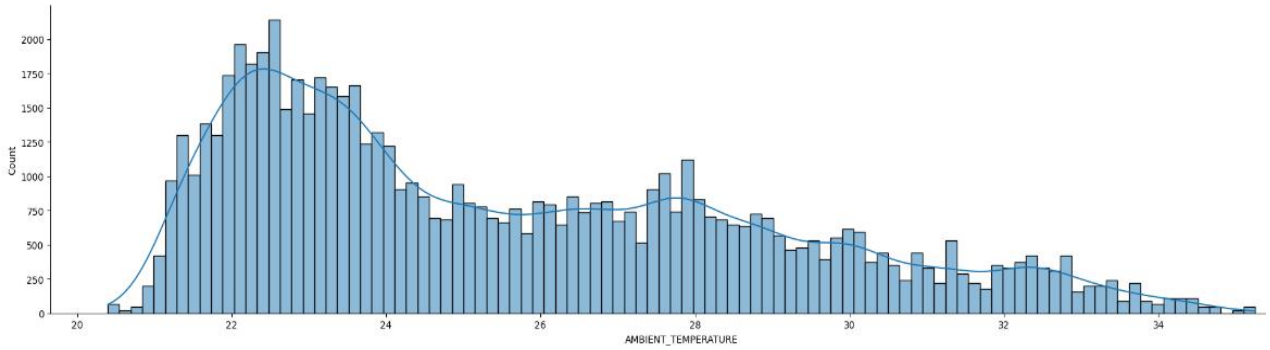
Bootstrap Aggregating or Bagging: The technique of bagging is a cornerstone of Random Forest's training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.

Decision Making and Voting: When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the mode (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

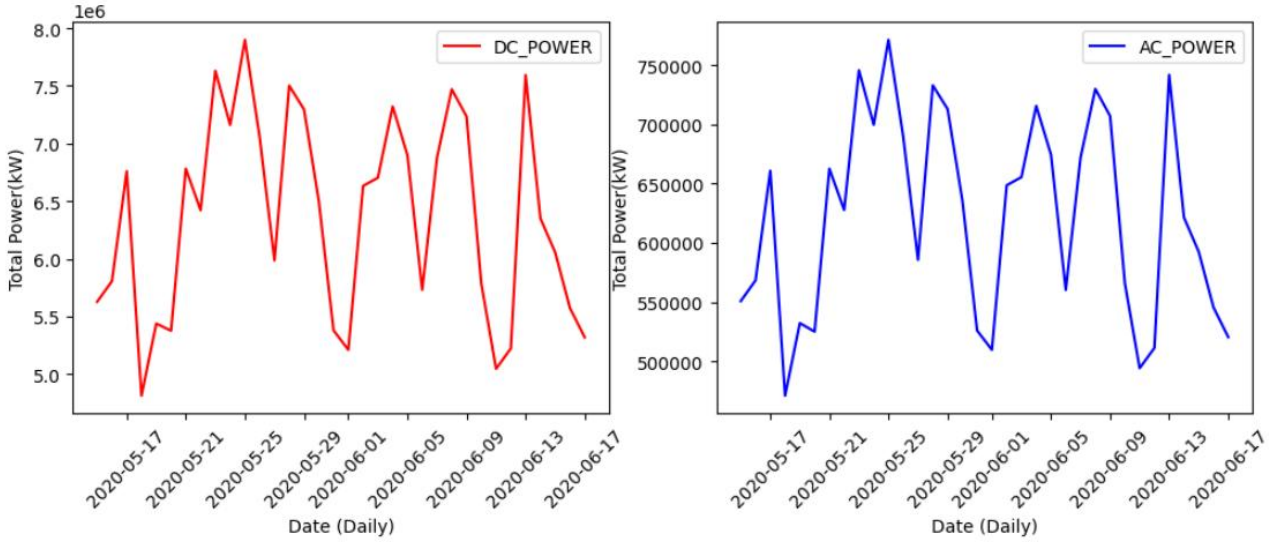
Exploratory data analysis :

❖ Distribution of Ambient temperature

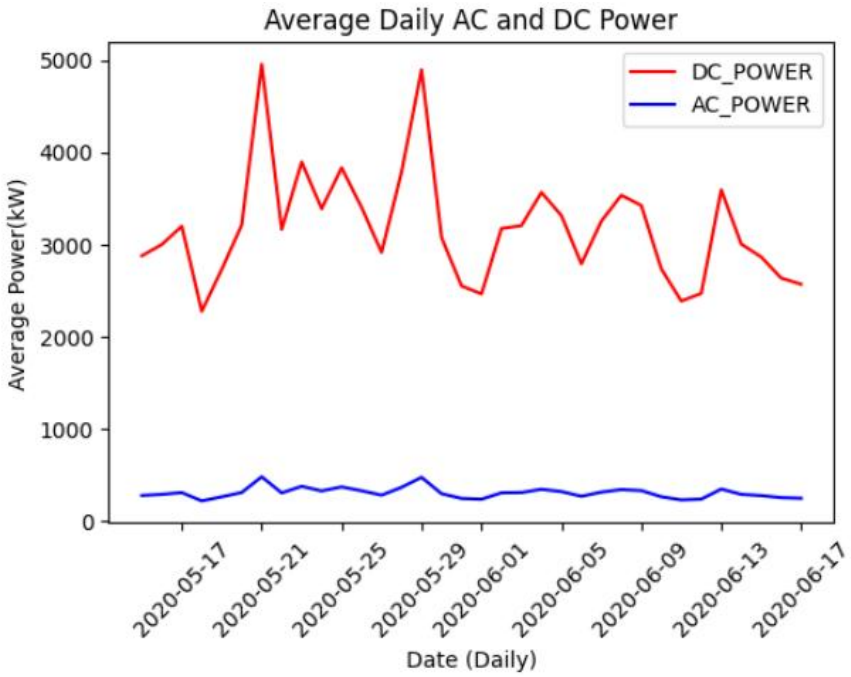
```
219]: sns.displot(data= merged, x="AMBIENT_TEMPERATURE", kde= True, bins = 100, height=5, aspect=4)
219]: <seaborn.axisgrid.FacetGrid at 0x21538045190>
```



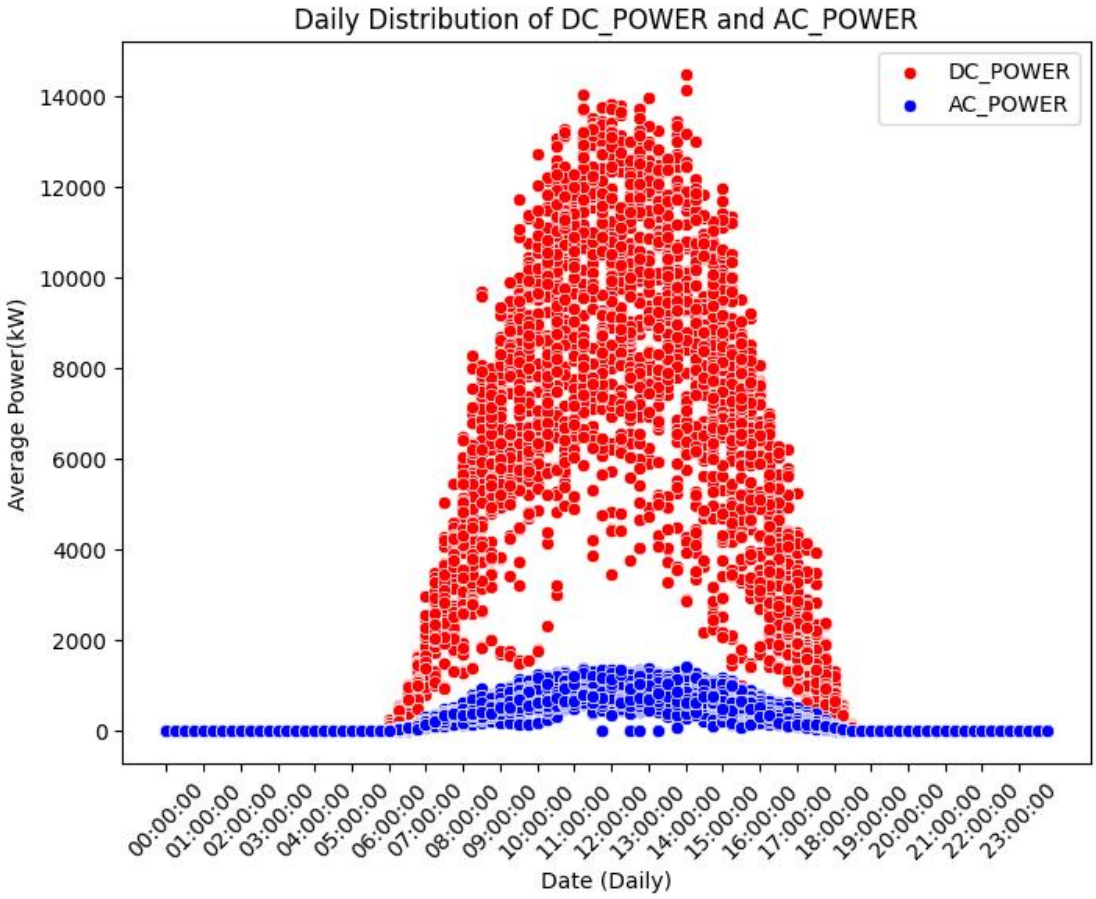
❖ Daily DC power and AC power generated



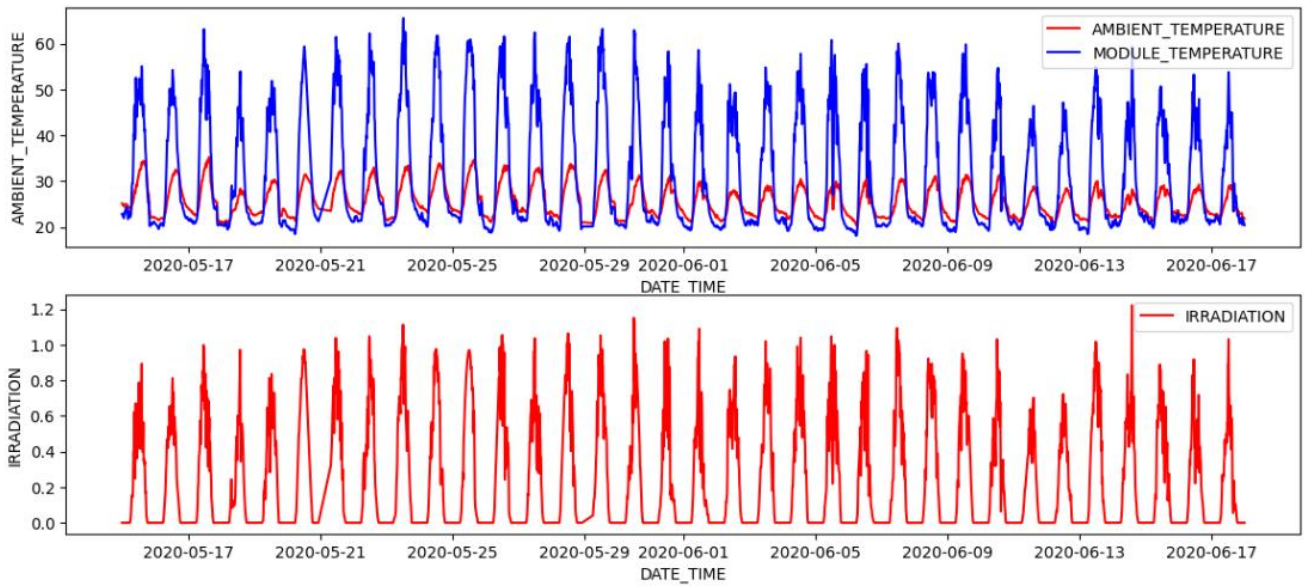
❖ Daily average power generated



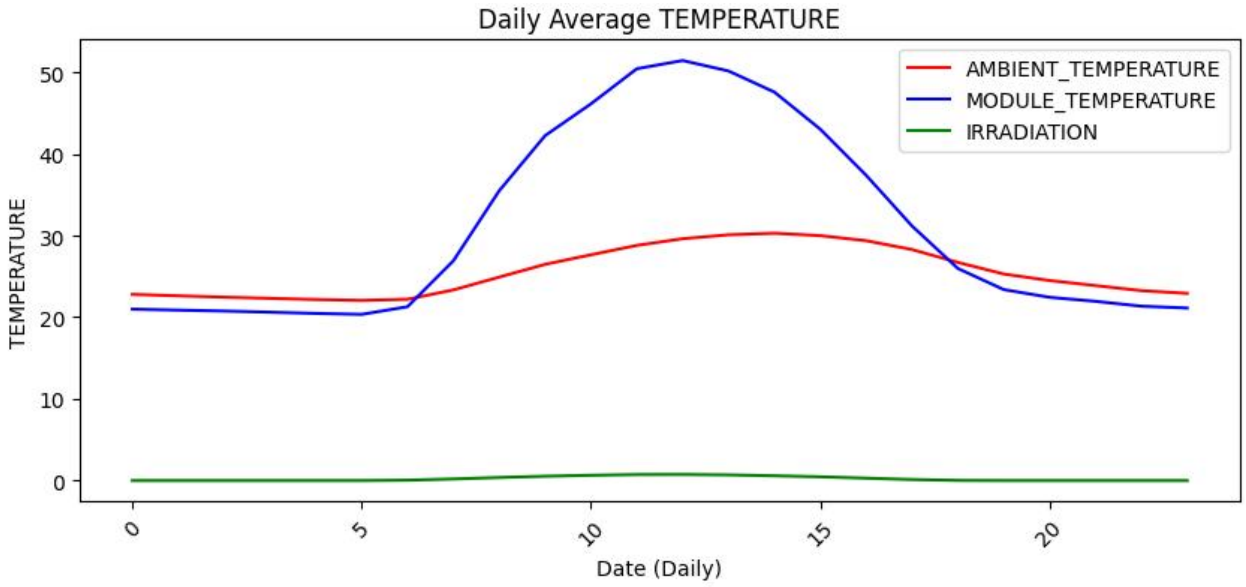
❖ Daily distribution of ac and dc power



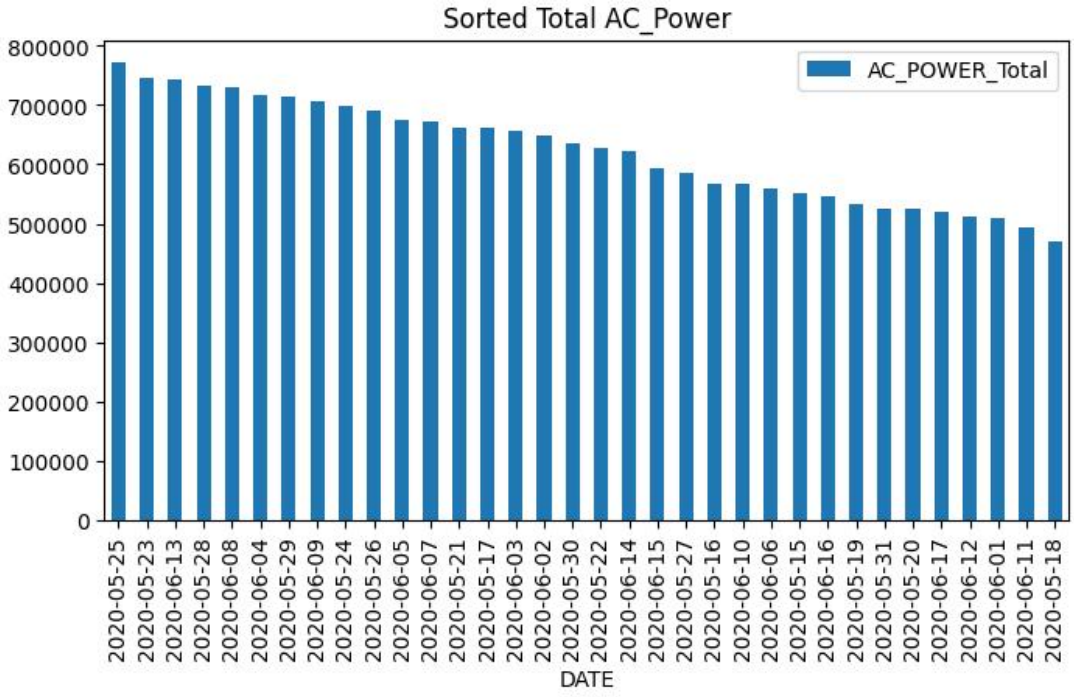
❖ Ambient temperature And Irradiation vs DATE_TIME



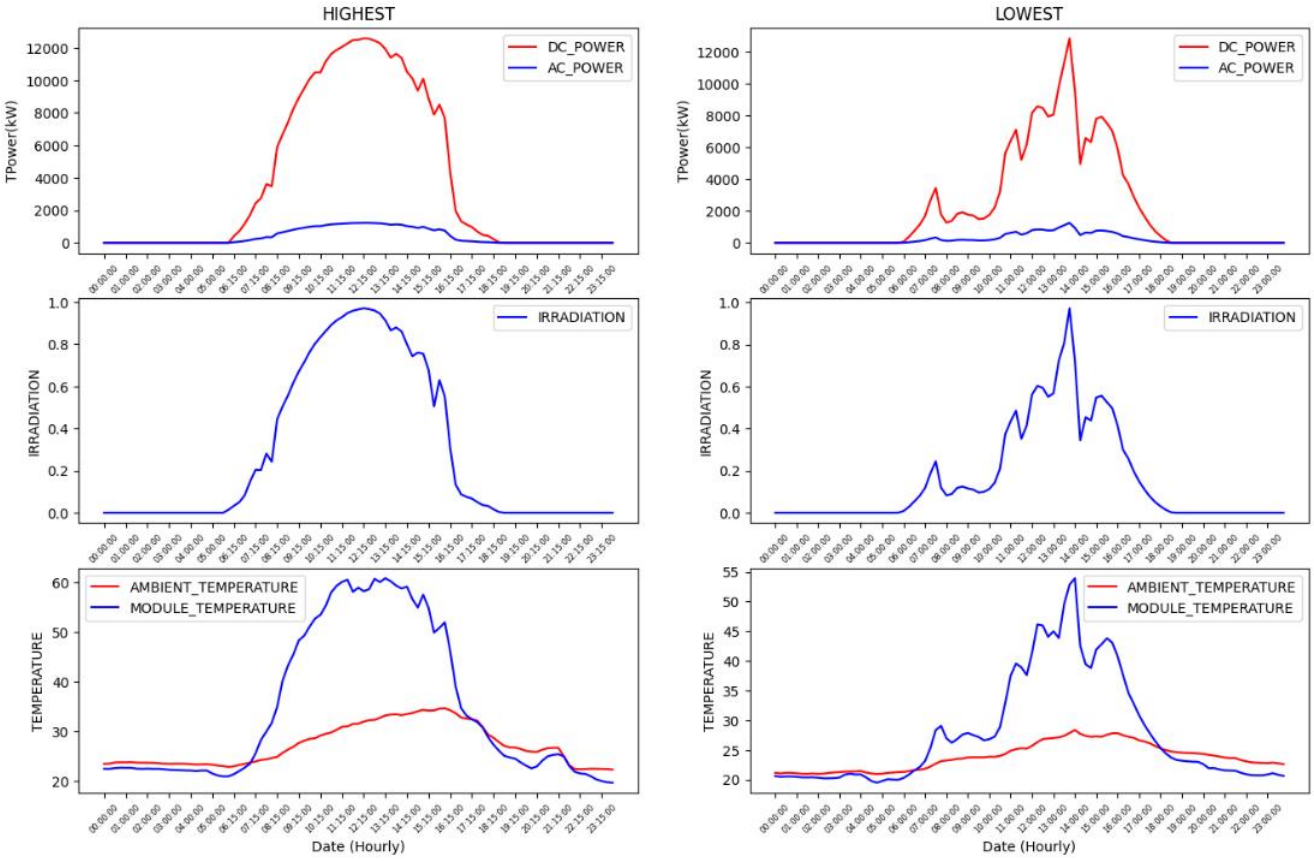
❖ Daily average temperature



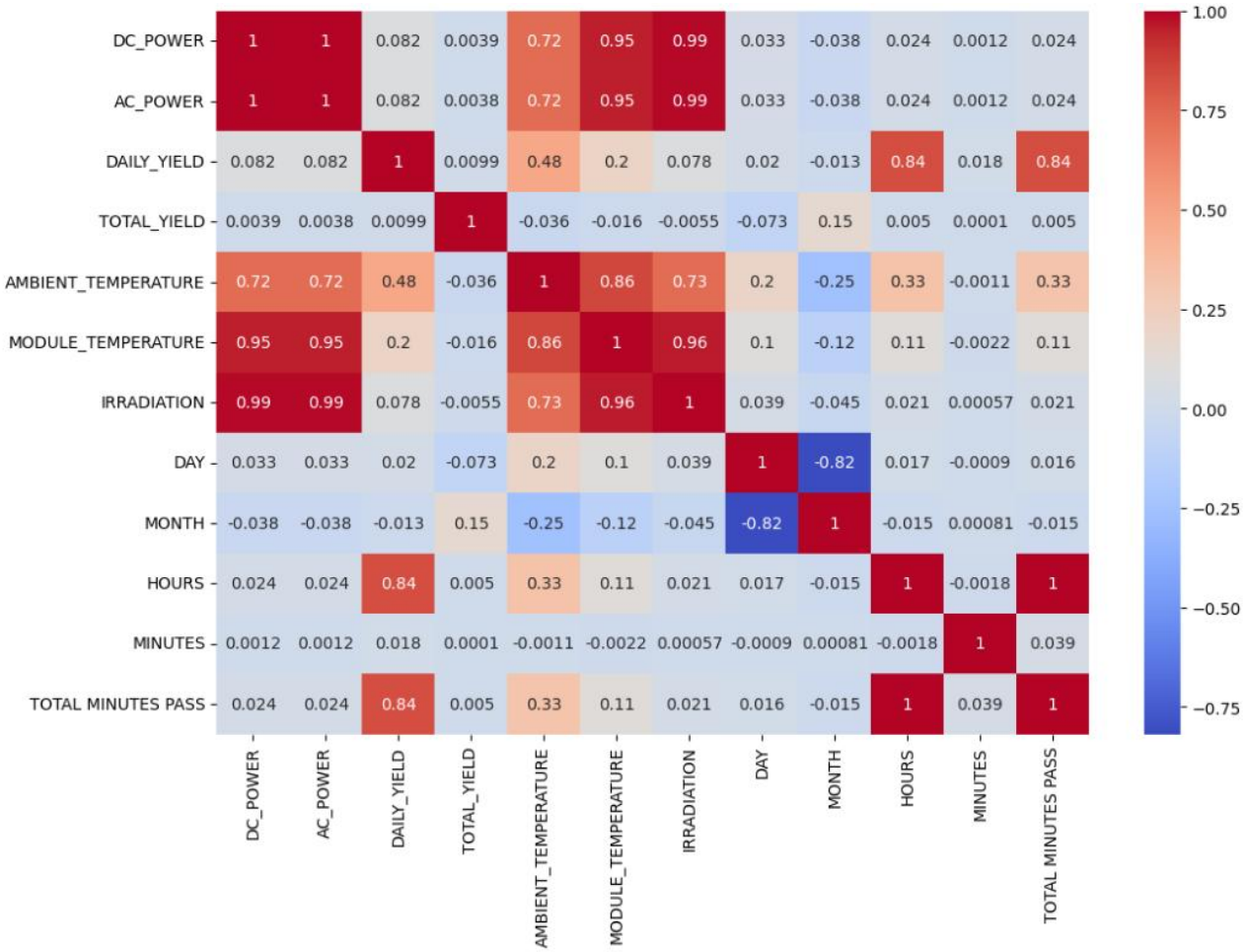
❖ Sorted Total_AC_power



❖ Comparison between highest and lowest



❖ Correlation analysis



RESULTS

1. Linear Regression :

[278]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
y_pred_lr = lr.predict(x_test)
R2_Score_lr = (r2_score(y_pred_lr,y_test) * 100)
MAE_lr = mean_absolute_error(y_pred_lr, y_test)
#Make Normalization

print("R2 Score : ",R2_Score_lr,"%")
print("MAE : ",MAE_lr,"")
```

```
R2 Score :  99.99940219845737 %
MAE :  0.6513600904748844
```

[280]:

	Actual	Predicted
0	0.000000	0.141637
1	1190.087500	1193.038723
2	1088.828571	1089.684390
3	134.675000	135.574417
4	1059.371429	1061.572642

2. Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
dtr.fit(x_train,y_train)

y_pred_dtr = rfr.predict(x_test)
R2_Score_dtr = (r2_score(y_pred_dtr,y_test) * 100)
MAE_dtr = mean_absolute_error(y_pred_dtr, y_test)

print("R2 Score : ",R2_Score_dtr,"%")
print("MAE : ",MAE_dtr,"")
```

```
R2 Score :  99.9999145285789 %
MAE :  0.12497124994444442
```

[283]:

	Actual	Predicted
0	0.000000	0.0000
1	1190.087500	1190.5000
2	1088.828571	1088.4375
3	134.675000	134.9000
4	1059.371429	1058.3000

3. Random forest

[275]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
y_pred_rfr = rfr.predict(x_test)
R2_Score_rfr = (r2_score(y_pred_rfr,y_test) * 100)
MAE_rfr = mean_absolute_error(y_pred_rfr, y_test)

print("R2 Score : ",R2_Score_rfr,"%")
print("MAE : ",MAE_rfr,"")
```

```
R2 Score :  99.9999145285789 %
MAE :  0.12497124994444442
```

[277]:

	Actual	Predicted
0	0.000000	0.000000
1	1190.087500	1190.502857
2	1088.828571	1088.760310
3	134.675000	134.694554
4	1059.371429	1059.524911
5	957.585714	958.007786
6	89.314286	89.318714
7	0.000000	0.000000
8	597.275000	597.256268
9	3.600000	3.572821