West Bengal State Electricity Distribution Company Limited.

A report on Machine Learning Applications of Power Systems

Name: Shreya Mukherjee
Enrollment No:21UEI075
Year: 3rd Year(5th Semester)
Institute: National Institute of Technology, Agartala

12-20-2023

Under the Guidance of

Mr. Jayanta Kumar Panda

D.E. (Tech)

Howrah Division I

WBSEDCL

CERTIFICATE

This is to certify that Miss Shreya Mukherjee (Enrollment No:21UEI075) student of B.Tech (Electronics and Instrumentation Engineering) 5th semester from NIT Agartala has successfully completed training at 33/11kv substation under Howrah Division –I ,owned by West Bengal State Electricity Distribution Company Limited (WBSEDCL),and various Machine Learning Aspects to ease the monitoring purpose of electricity produced from PVC.

Date:

Jayata Kumar Panda

Divisional Engineer(Technical)

Unit 1: Machine Learning

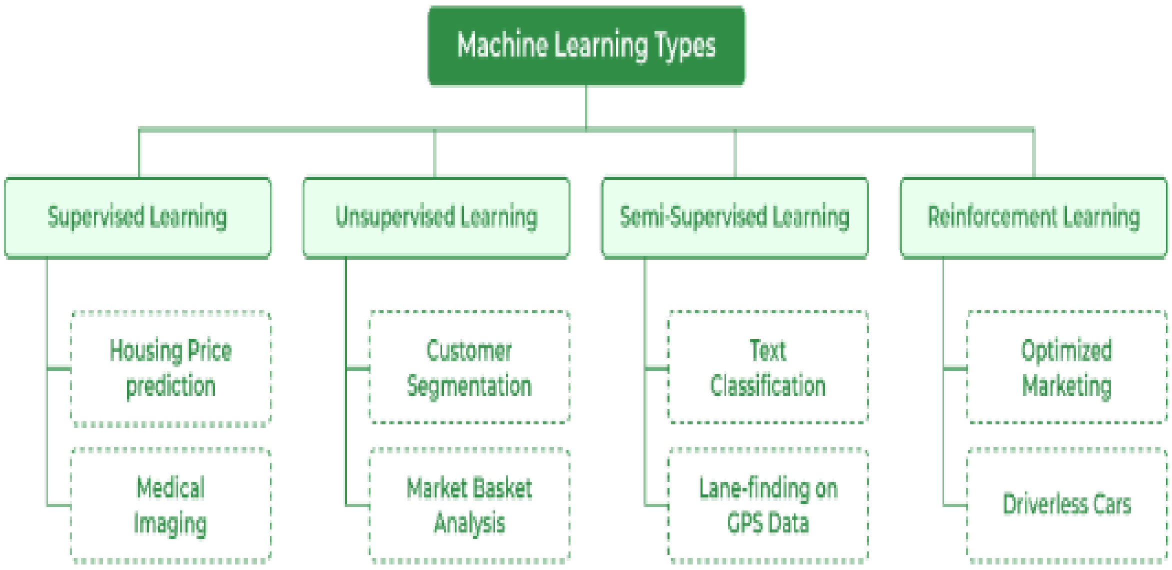
1.1.Introduction

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can effectively generalize and thus perform tasks without explicit instructions. Recently, generative artificial neural networks have been able to surpass many previous approaches in performance. Machine learning approaches have been applied to large language models, computer vision, speech recognition, email filtering, agriculture, and medicine, where it is too costly to develop algorithms to perform the needed tasks.

The mathematical foundations of ML are provided by mathematical optimization (mathematical programming) methods. Data mining is a related (parallel) field of study, focusing on exploratory data analysis through unsupervised learning.

ML is known in its application across business problems under the name predictive analytics. Although not all machine learning is statistically based, computational statistics is an important source of the field's methods.

2. Types of Machine Learning Algorithm



Supervised learning

A support-vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function allows the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Types of supervised-learning algorithms include active learning, classification and regression.

- **Classification algorithms are used when the outputs are restricted to a limited set of values, and**
- **Regression algorithms are used when the outputs may have any numerical value within a range.**

As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features. Unsupervised learning algorithms streamlined the process of survey and graph large indel based haplotypes of a gene of interest from pan-genome.

Clustering via Large Indel Permuted Slopes, CLIPS, turns the alignment image into a learning regression problem. The varied slope (b) estimates

between each pair of DNA segments enables to identify segments sharing the same set of indels.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In reinforcement learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcements learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP and are used when exact models are infeasible. Reinforcement learning

algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

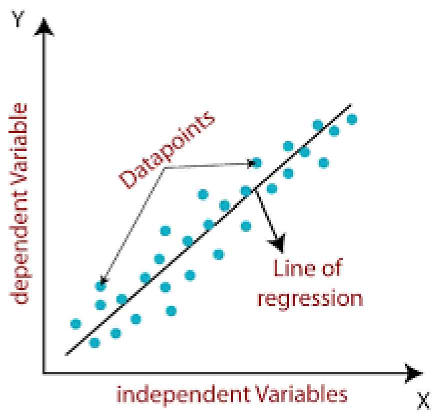
Dimensionality reduction

Dimensionality reduction is a process of reducing the number of random variables under consideration by obtaining a set of principal variables. In other words, it is a process of reducing the dimension of the feature set, also called the "number of features". Most of the dimensionality reduction techniques can be considered as either feature elimination or extraction. One of the popular methods of dimensionality reduction is principal component analysis (PCA). PCA involves changing higher-dimensional data (e.g., 3D) to a smaller space (e.g., 2D). This results in a smaller dimension of data (2D instead of 3D), while keeping all original variables in the model without changing the data.The manifold hypothesis proposes that high-dimensional data sets lie along low-dimensional manifolds, and many dimensionality reduction techniques make this assumption, leading to the area of manifold learning and manifold regularization.

3. Different Algorithms

Supervised Learning Algorithms:

1. **Linear Regression: Predicts continuous values based on input features by fitting a linear equation to the observed data.**



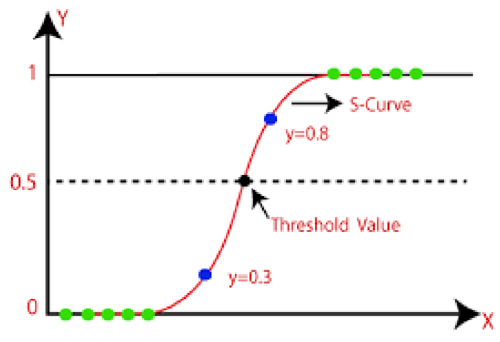
The equation for the regression line is:

$y= a_0+ a*x+ b$

Here, y= dependent variable

x= independent variable

2. **Logistic Regression: Used for binary classification by estimating probabilities using a logistic function.**



The odd is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur. so odd will be

$$\frac{p(x)}{1-p(x)} = e^z$$

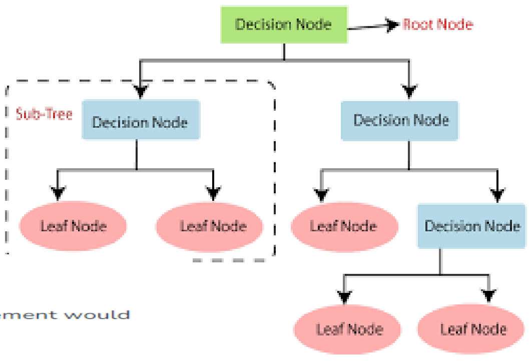
Applying natural log on odd. then log odd will be

$$\log\left[\frac{p(x)}{1-p(x)}\right] = z$$
$$\log\left[\frac{p(x)}{1-p(x)}\right] = w \cdot X + b$$

then the final logistic regression equation will be:

$$p(X;b,w) = \frac{e^{w \cdot X+b}}{1+e^{w \cdot X+b}} = \frac{1}{1+e^{-w \cdot X+b}}$$

3. **Decision Trees: Utilizes a tree-like model for decisions by splitting the dataset into branches based on feature values.**



Gini index

Gini Index $= 1 - \sum_j {}_j^2$

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower gini index should be preferred.
- Sklearn supports "gini" criteria for Gini Index and by default, it takes "gini" value.

Entropy

If a random variable x can take N different value, the i'value x_i with probability p_{ii} we can associate the following entropy with x :

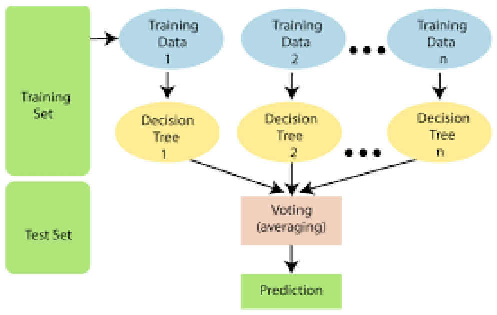
$H(x) = -\sum_{i=1}^{N} p(x_i) log_2 p(x_i)$

- Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy the more the information content.

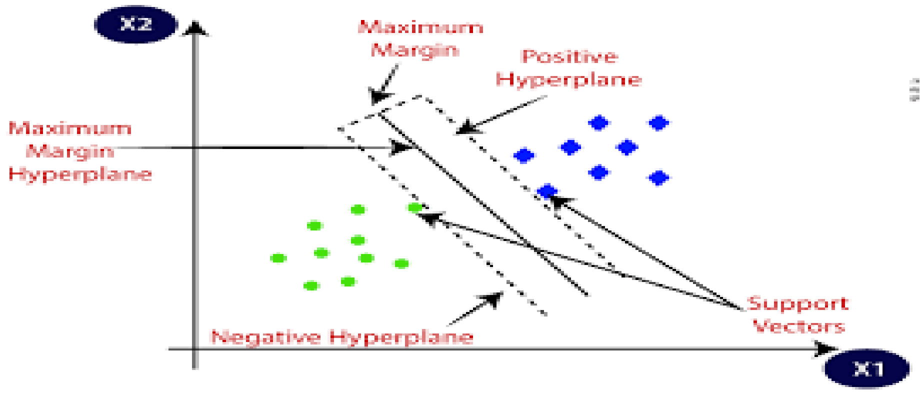
with A = v and Values(A) is the set of all possible of A, then

- The entropy typically changes when we use a node in a Python decision tree to partition the training instances into smaller subsets. Information gain is a measure of this change in entropy.
- Sklearn supports "entropy" criteria for Information Gain and if we want to use Information Gain method in sklearn then we have to mention it explicitly.

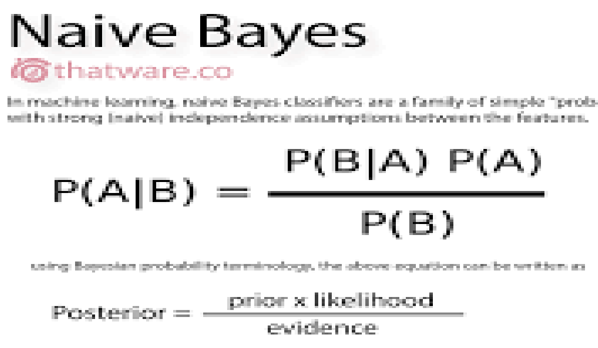
4. **Random Forest: An ensemble method that constructs multiple decision trees and merges their outputs for more accurate predictions.**



5. **Support Vector Machines (SVM): Maps data into a high dimensional space to find a hyperplane that best separates classes.**



6.Naive Bayes: Based on Bayes' theorem, calculates probabilities using the assumption that features are independent.



Naive Bayes

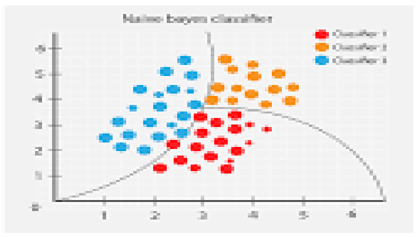
thatware.co

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

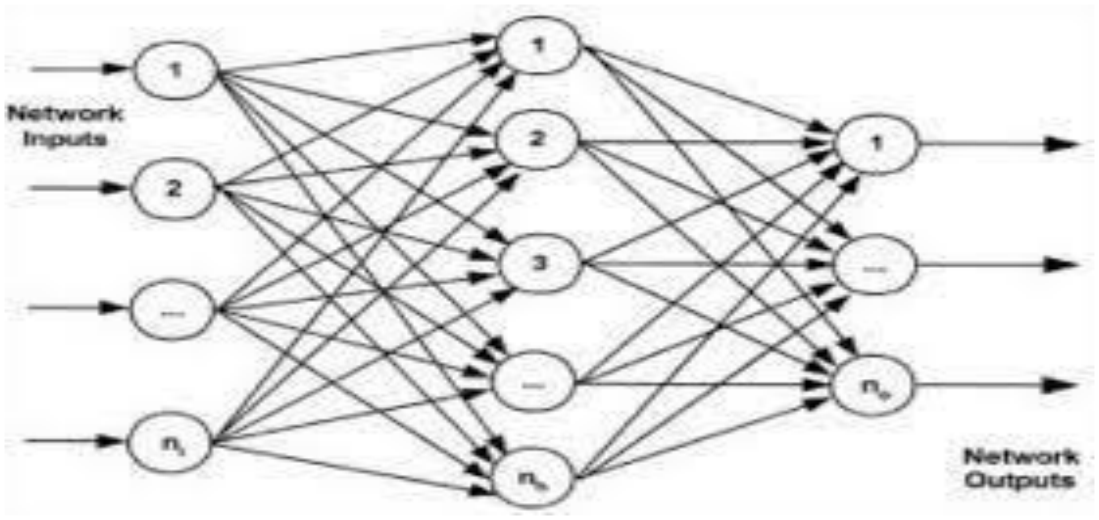
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

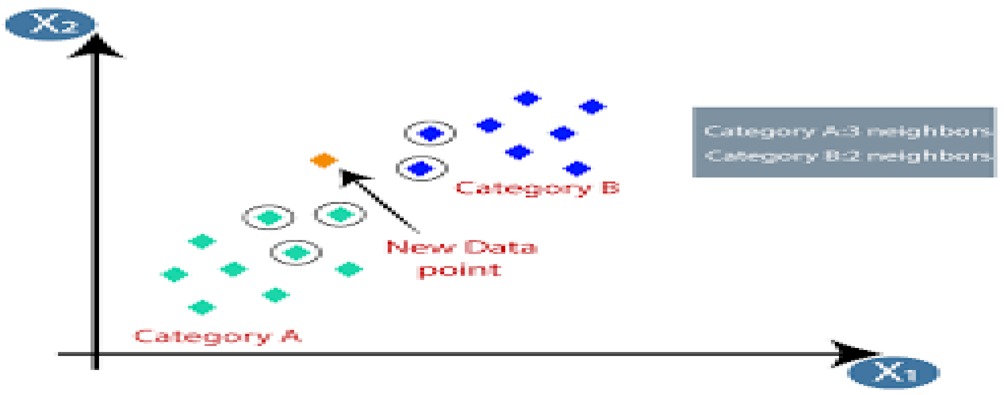
$$\text{Posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$



7.Neural Networks: Mimics the human brain's structure with interconnected nodes (neurons) organized in layers; includes architectures like feedforward, convolutional, and recurrent neural networks.

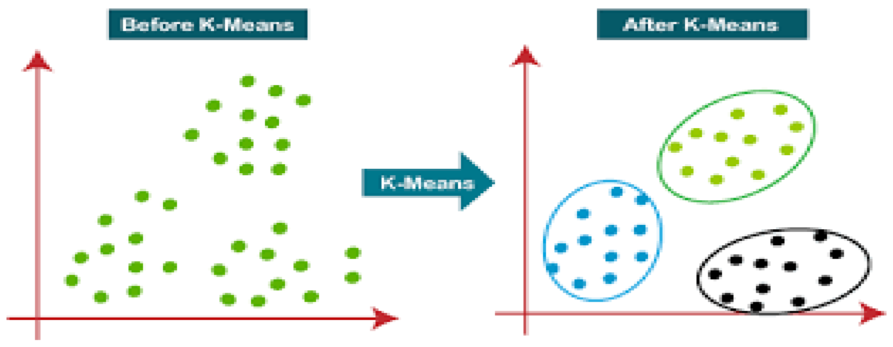


8. K-Nearest Neighbors (KNN): Classifies data based on the majority class among its k nearest neighbors in the feature space.

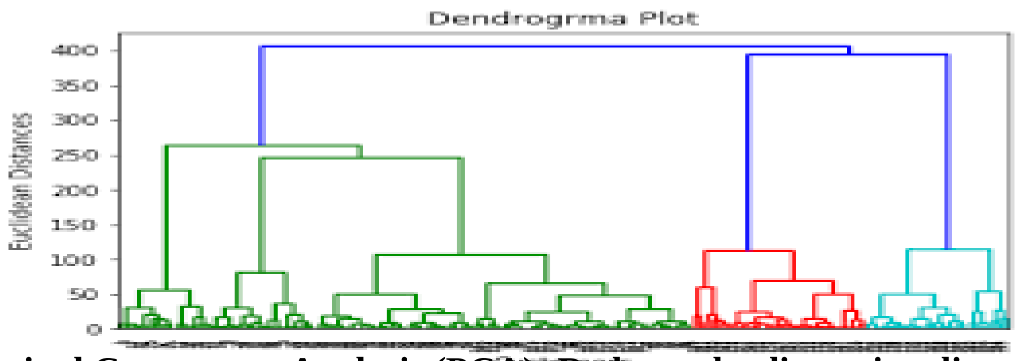


Unsupervised Learning Algorithms:

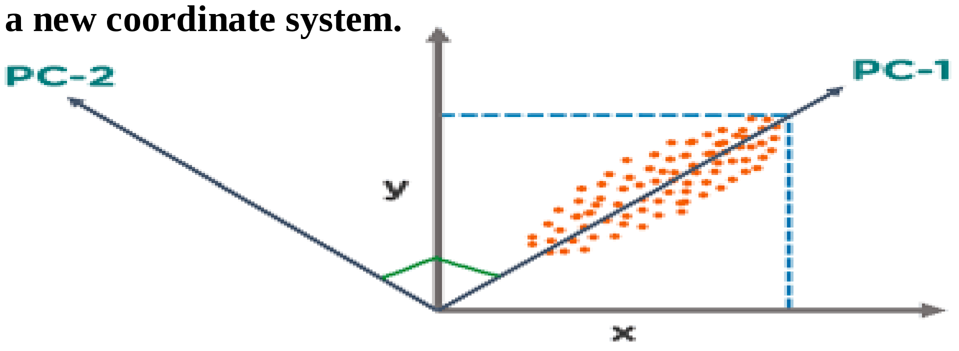
1. **K-Means Clustering: Divides data into 'k' clusters based on similarity or distance metrics.**



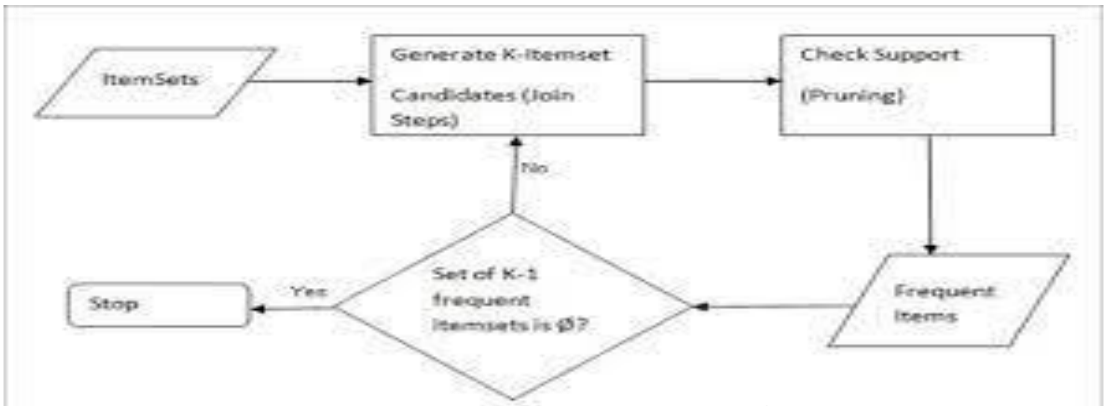
2. **Hierarchical Clustering: Forms clusters in a hierarchical tree-like structure to represent relationships.**



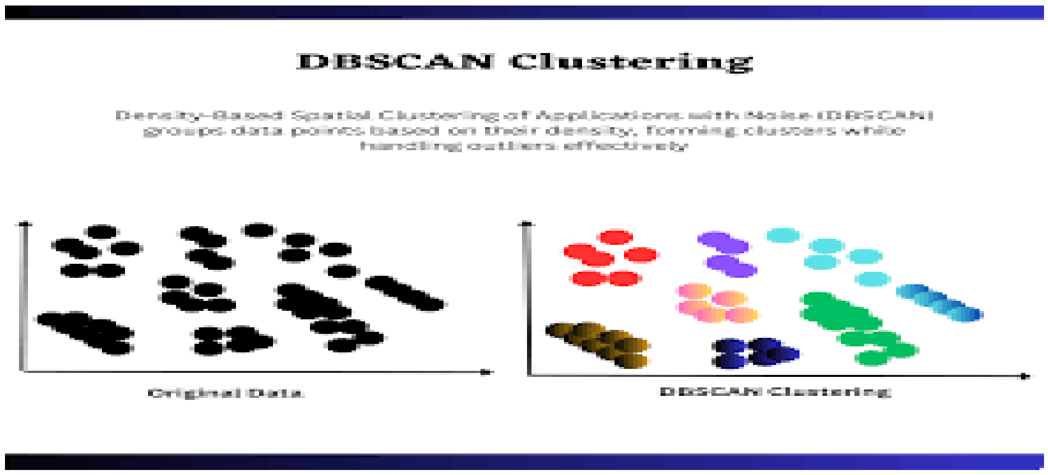
3. **Principal Component Analysis (PCA): Reduces the dimensionality of data while preserving important information by transforming it into a new coordinate system.**



4. **Apriori Algorithm: Identifies frequent itemsets in transactional databases to establish associations or patterns.**


ItemSets
Generate K-Itemset Candidates (Join Steps)
Check Support (Pruning)
No
Set of K-1 frequent itemsets is Ø?
Yes
Stop
Frequent Items

5. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Groups together points that are closely packed, marking outliers as noise.**

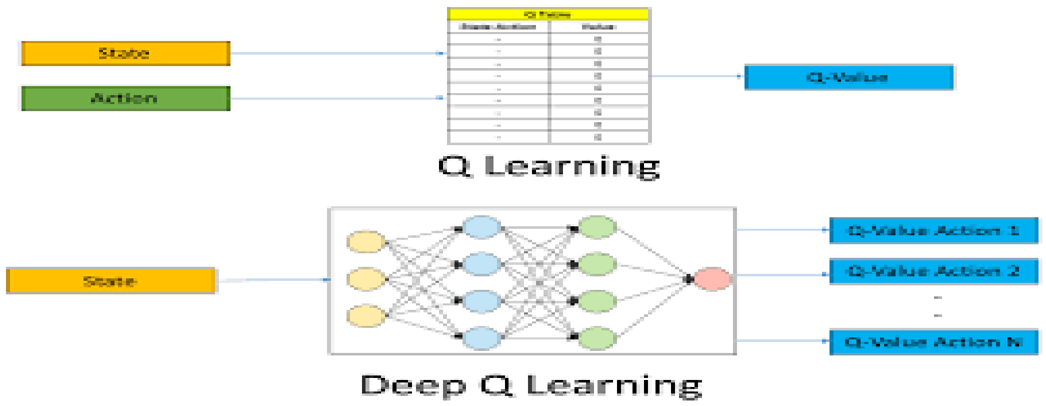


Semi-Supervised Learning Algorithms:

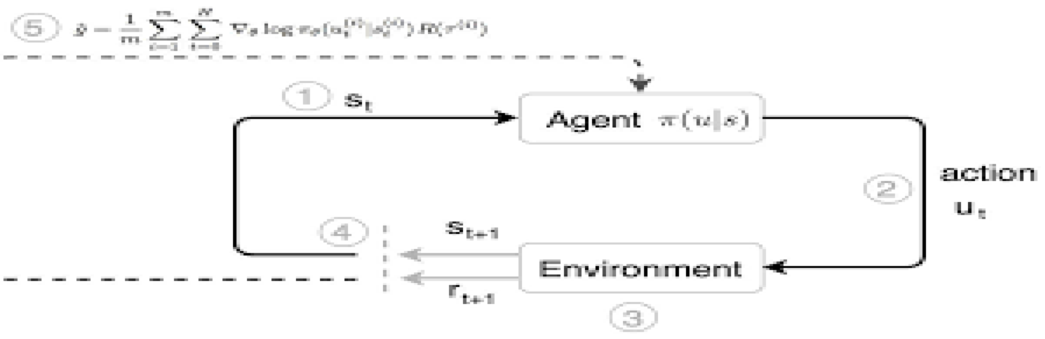
1. **Self-training: Trains a model on a small labeled dataset and then uses this model to label the remaining unlabeled data.**
2. **Semi-Supervised SVM: Extends traditional SVM to incorporate both labeled and unlabeled data in the learning process.**

Reinforcement Learning Algorithms:

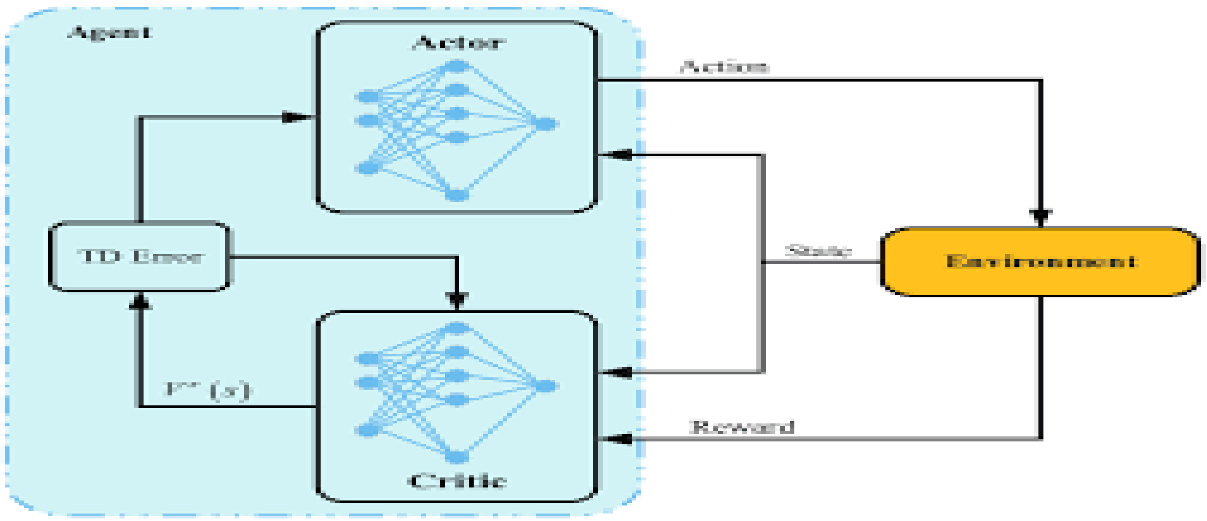
1. **Q-Learning: A model-free reinforcement learning algorithm where an agent learns to make decisions by maximizing an expected cumulative reward.**



2. **Deep Q Network (DQN): Utilizes neural networks to approximate the Q-values in Q-learning for handling complex state spaces.**
3. **Policy Gradient Methods: Directly optimize the policy function in reinforcement learning to find the best actions in an environment.**


5
$g = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})R(\tau^{(i)})$
1
s_t
Agent $\pi(u|s)$
2
action
u_t
4
s_{t+1}
Environment
r_{t+1}
3

4. **Actor-Critic: Combines value-based and policy-based methods by maintaining both value function and policy function to improve learning stability and efficiency.**



4. Application of Machine Learning Algorithm

Supervised Learning:

- **Image Classification: Identifying objects in images (e.g., cats vs. dogs).**
- **Speech Recognition: Transcribing spoken words into text.**

- **Predictive Analytics: Predicting sales, stock prices, etc., based on historical data.**
- **Medical Diagnosis: Identifying diseases based on symptoms.**

Unsupervised Learning:

- **Clustering: Grouping similar data points together (e.g., customer segmentation).**
- **Dimensionality Reduction: Reducing the number of features while preserving relevant information (e.g., principal component analysis).**
- **Anomaly Detection: Identifying outliers or anomalies in data (e.g., fraud detection).**

Semi-Supervised Learning:

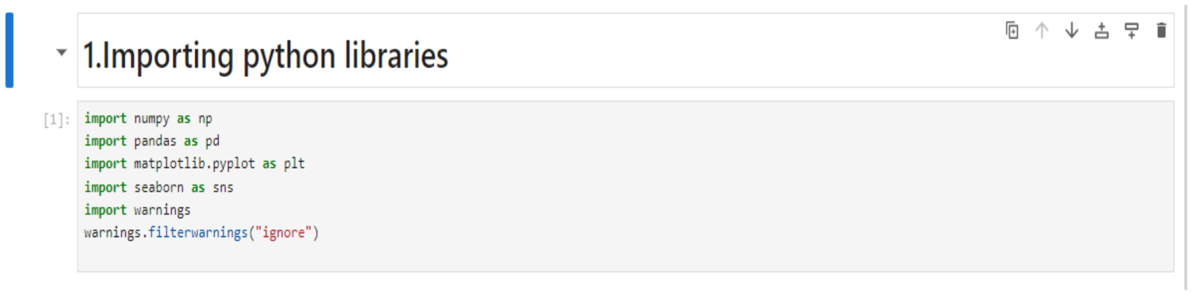
- **Text Analysis: Classifying documents when only a fraction of them are labeled.**
- **Image Recognition: Leveraging a small set of labeled images along with a larger set of unlabeled images for training.**

Reinforcement Learning:

- **Game Playing: Training AI agents to play games such as Chess, Go, or video games.**
- **Robotics: Teaching robots to perform tasks like walking, grasping objects, or navigation.**
- **Autonomous Driving: Teaching self-driving cars to navigate traffic and make decisions in real-time.**

5. Preparation of Data before fitting it into ML Models

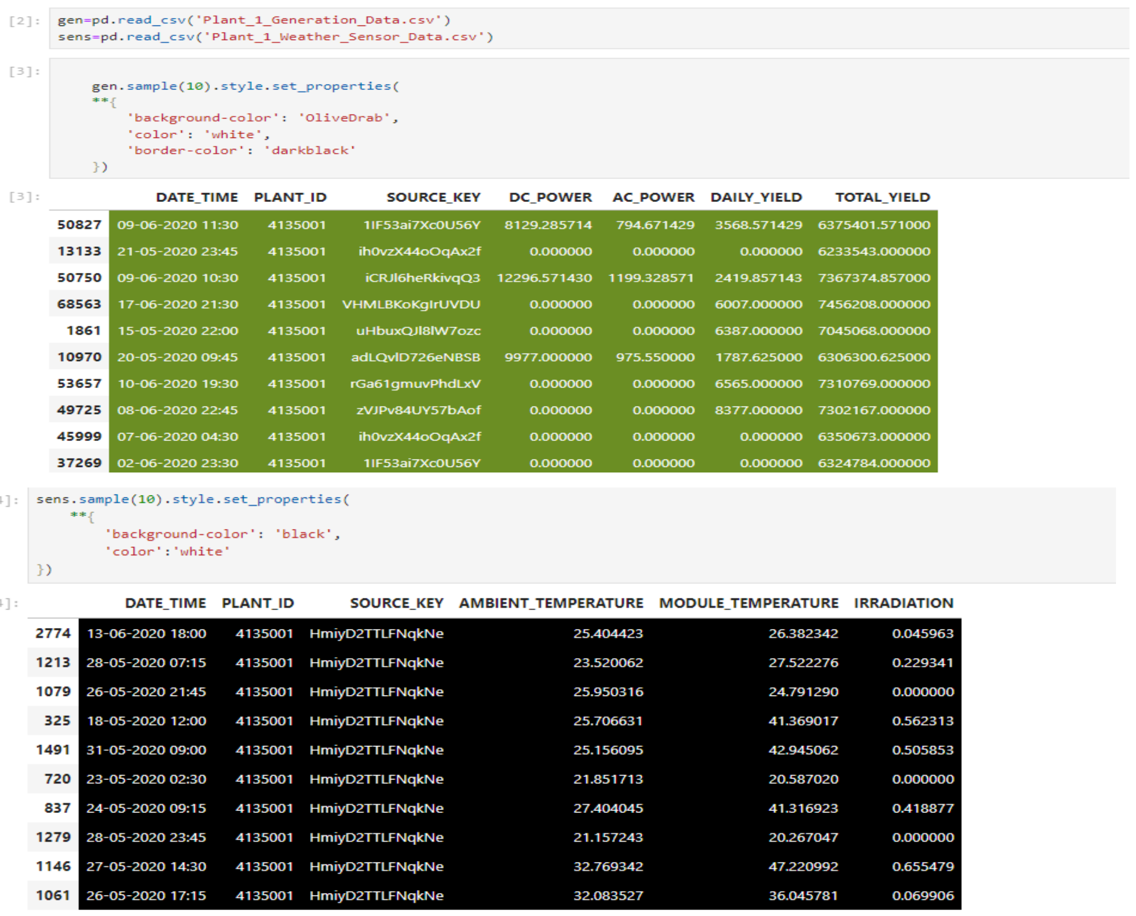
1. **Importing necessary libraries**


1.Importing python libraries

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

1. Data Collection:

- **Identify relevant data sources and gather the required datasets.**
- **Ensure the data collected aligns with the objectives and requirements of the machine learning problem.**



```
[2]: gen=pd.read_csv('Plant_1_Generation_Data.csv')
     sens=pd.read_csv('Plant_1_Weather_Sensor_Data.csv')
```

```
[3]:
     gen.sample(10).style.set_properties(
     **{
         'background-color': 'OliveDrab',
         'color': 'white',
         'border-color': 'darkblack'
     })
```

[3]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
50827	09-06-2020 11:30	4135001	1IF53ai7Xc0U56Y	8129.285714	794.671429	3568.571429	6375401.571000
13133	21-05-2020 23:45	4135001	ih0vzX44oOqAx2f	0.000000	0.000000	0.000000	6233543.000000
50750	09-06-2020 10:30	4135001	iCRJl6heRkivqQ3	12296.571430	1199.328571	2419.857143	7367374.857000
68563	17-06-2020 21:30	4135001	VHMLBKoKgIrUVDU	0.000000	0.000000	6007.000000	7456208.000000
1861	15-05-2020 22:00	4135001	uHbuxQJl8lW7ozc	0.000000	0.000000	6387.000000	7045068.000000
10970	20-05-2020 09:45	4135001	adLQvlD726eNBSB	9977.000000	975.550000	1787.625000	6306300.625000
53657	10-06-2020 19:30	4135001	rGa61gmuvPhdLxV	0.000000	0.000000	6565.000000	7310769.000000
49725	08-06-2020 22:45	4135001	zVJPv84UY57bAof	0.000000	0.000000	8377.000000	7302167.000000
45999	07-06-2020 04:30	4135001	ih0vzX44oOqAx2f	0.000000	0.000000	0.000000	6350673.000000
37269	02-06-2020 23:30	4135001	1IF53ai7Xc0U56Y	0.000000	0.000000	0.000000	6324784.000000

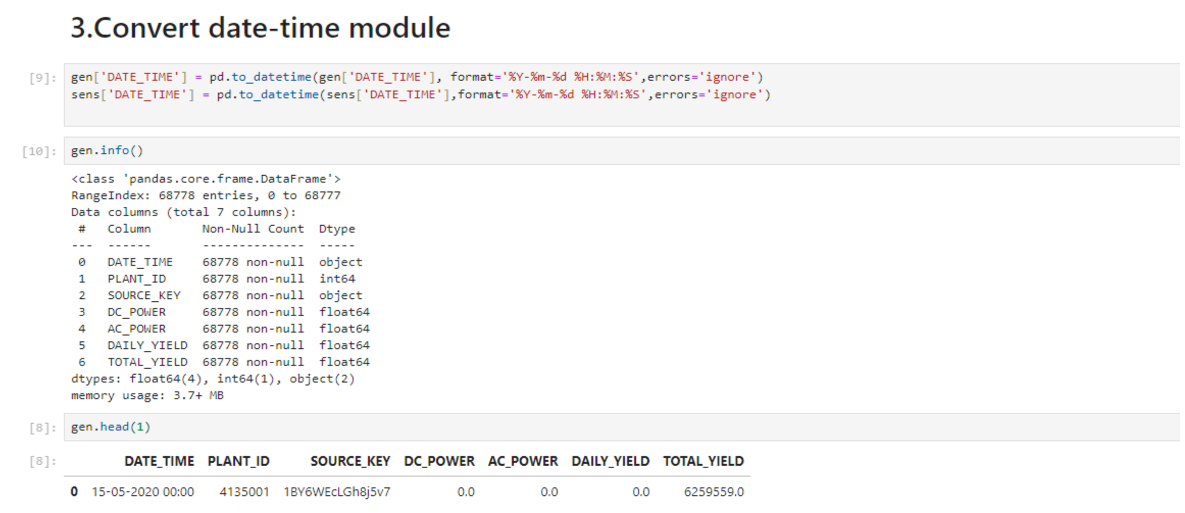
```
[4]: sens.sample(10).style.set_properties(
         **{
             'background-color': 'black',
             'color':'white'
     })
```

[4]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
2774	13-06-2020 18:00	4135001	HmiyD2TTLFNqkNe	25.404423	26.382342	0.045963
1213	28-05-2020 07:15	4135001	HmiyD2TTLFNqkNe	23.520062	27.522276	0.229341
1079	26-05-2020 21:45	4135001	HmiyD2TTLFNqkNe	25.950316	24.791290	0.000000
325	18-05-2020 12:00	4135001	HmiyD2TTLFNqkNe	25.706631	41.369017	0.562313
1491	31-05-2020 09:00	4135001	HmiyD2TTLFNqkNe	25.156095	42.945062	0.505853
720	23-05-2020 02:30	4135001	HmiyD2TTLFNqkNe	21.851713	20.587020	0.000000
837	24-05-2020 09:15	4135001	HmiyD2TTLFNqkNe	27.404045	41.316923	0.418877
1279	28-05-2020 23:45	4135001	HmiyD2TTLFNqkNe	21.157243	20.267047	0.000000
1146	27-05-2020 14:30	4135001	HmiyD2TTLFNqkNe	32.769342	47.220992	0.655479
1061	26-05-2020 17:15	4135001	HmiyD2TTLFNqkNe	32.083527	36.045781	0.069906

2. Data Cleaning:

- **Handle missing values: Identify and deal with missing data through imputation (using mean, median, or mode) or deletion if appropriate.**
- **Remove duplicate records if present in the dataset.**
- **Handle outliers: Decide whether to remove outliers, transform them, or use robust models that are less sensitive to outliers.**



3.Convert date-time module

```
[9]: gen['DATE_TIME'] = pd.to_datetime(gen['DATE_TIME'], format='%Y-%m-%d %H:%M:%S',errors='ignore')
     sens['DATE_TIME'] = pd.to_datetime(sens['DATE_TIME'],format='%Y-%m-%d %H:%M:%S',errors='ignore')
```

```
[10]: gen.info()
```

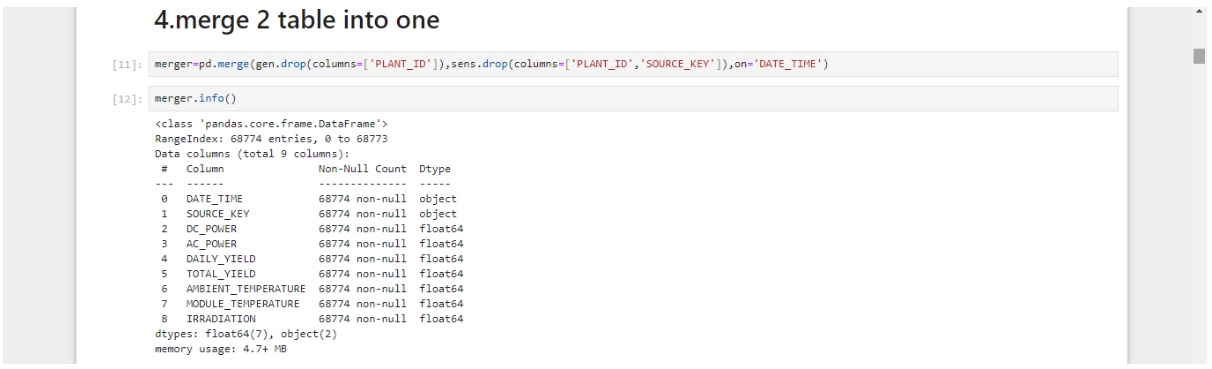
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68778 entries, 0 to 68777
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   DATE_TIME    68778 non-null  object
 1   PLANT_ID     68778 non-null  int64
 2   SOURCE_KEY   68778 non-null  object
 3   DC_POWER     68778 non-null  float64
 4   AC_POWER     68778 non-null  float64
 5   DAILY_YIELD  68778 non-null  float64
 6   TOTAL_YIELD  68778 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.7+ MB
```

```
[8]: gen.head(1)
```

[8]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0

Merging generation and sensor table data into one

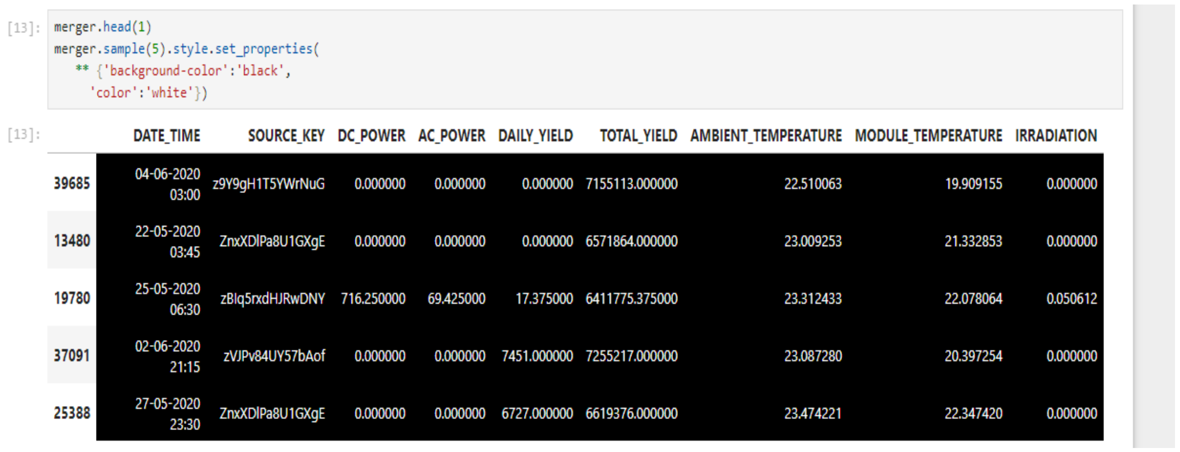


4.merge 2 table into one

```
[11]: merger=pd.merge(gen.drop(columns=['PLANT_ID']),sens.drop(columns=['PLANT_ID','SOURCE_KEY']),on='DATE_TIME')
```

```
[12]: merger.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68774 entries, 0 to 68773
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   DATE_TIME            68774 non-null  object
 1   SOURCE_KEY           68774 non-null  object
 2   DC_POWER             68774 non-null  float64
 3   AC_POWER             68774 non-null  float64
 4   DAILY_YIELD          68774 non-null  float64
 5   TOTAL_YIELD          68774 non-null  float64
 6   AMBIENT_TEMPERATURE  68774 non-null  float64
 7   MODULE_TEMPERATURE   68774 non-null  float64
 8   IRRADIATION          68774 non-null  float64
dtypes: float64(7), object(2)
memory usage: 4.7+ MB
```

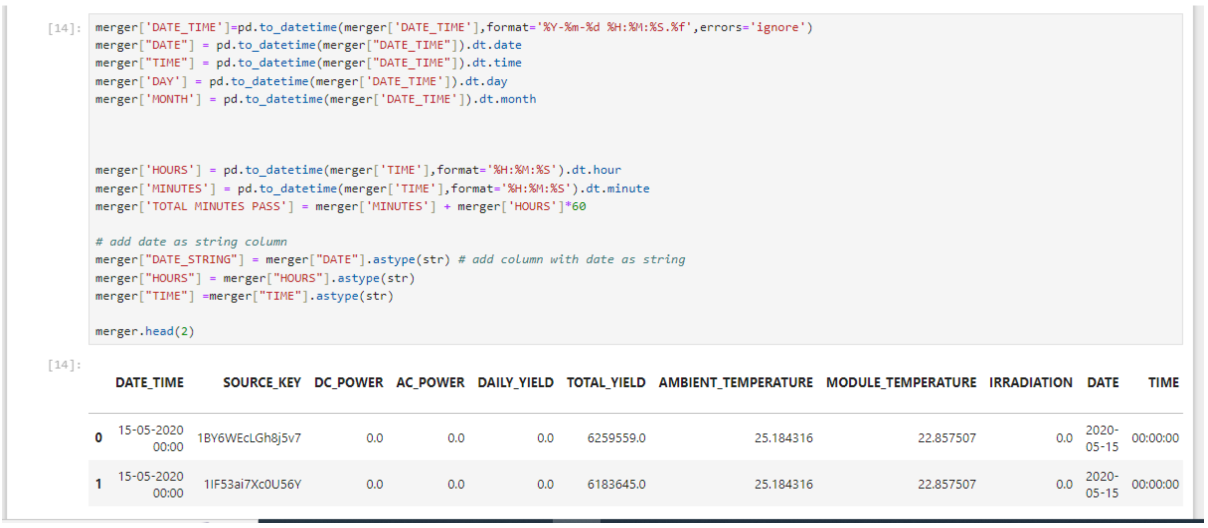


```
[13]: merger.head(1)
      merger.sample(5).style.set_properties(
         ** {'background-color':'black',
          'color':'white'})
```

[13]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
39685	04-06-2020 03:00	z9Y9gH1T5YWrNuG	0.000000	0.000000	0.000000	7155113.000000	22.510063	19.909155	0.000000
13480	22-05-2020 03:45	ZnxXDlPa8U1GXgE	0.000000	0.000000	0.000000	6571864.000000	23.009253	21.332853	0.000000
19780	25-05-2020 06:30	zBIq5rxdHJRwDNY	716.250000	69.425000	17.375000	6411775.375000	23.312433	22.078064	0.050612
37091	02-06-2020 21:15	zVJPv84UY57bAof	0.000000	0.000000	7451.000000	7255217.000000	23.087280	20.397254	0.000000
25388	27-05-2020 23:30	ZnxXDlPa8U1GXgE	0.000000	0.000000	6727.000000	6619376.000000	23.474221	22.347420	0.000000

Changing the datatype of “DATE_TIME” column to datetime

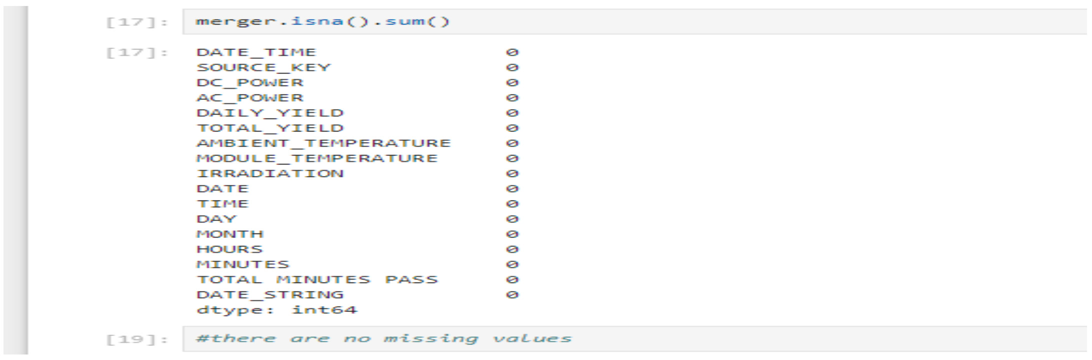


```
[14]: merger['DATE_TIME']=pd.to_datetime(merger['DATE_TIME'],format='%Y-%m-%d %H:%M:%S.%f',errors='ignore')
      merger["DATE"] = pd.to_datetime(merger["DATE_TIME"]).dt.date
      merger["TIME"] = pd.to_datetime(merger["DATE_TIME"]).dt.time
      merger['DAY'] = pd.to_datetime(merger['DATE_TIME']).dt.day
      merger['MONTH'] = pd.to_datetime(merger['DATE_TIME']).dt.month



      merger['HOURS'] = pd.to_datetime(merger['TIME'],format='%H:%M:%S').dt.hour
      merger['MINUTES'] = pd.to_datetime(merger['TIME'],format='%H:%M:%S').dt.minute
      merger['TOTAL MINUTES PASS'] = merger['MINUTES'] + merger['HOURS']*60

      # add date as string column
      merger["DATE_STRING"] = merger["DATE"].astype(str) # add column with date as string
      merger["HOURS"] = merger["HOURS"].astype(str)
      merger["TIME"] =merger["TIME"].astype(str)

      merger.head(2)
```

[14]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DATE	TIME
0	15-05-2020 00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0	25.184316	22.857507	0.0	2020-05-15	00:00:00
1	15-05-2020 00:00	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0	25.184316	22.857507	0.0	2020-05-15	00:00:00

Identification of missing values

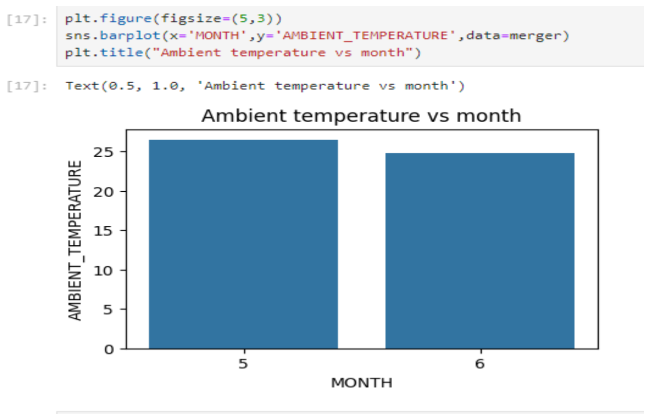


```
[17]: merger.isna().sum()

[17]: DATE_TIME                0
      SOURCE_KEY               0
      DC_POWER                 0
      AC_POWER                 0
      DAILY_YIELD              0
      TOTAL_YIELD              0
      AMBIENT_TEMPERATURE      0
      MODULE_TEMPERATURE       0
      IRRADIATION              0
      DATE                     0
      TIME                     0
      DAY                      0
      MONTH                    0
      HOURS                    0
      MINUTES                  0
      TOTAL MINUTES PASS       0
      DATE_STRING              0
      dtype: int64

[19]: #there are no missing values
```

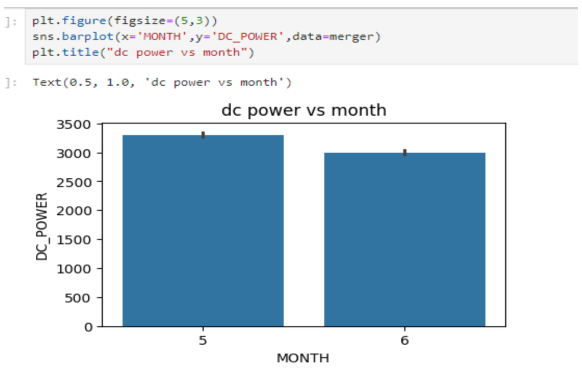
3. Data Exploration and Visualization:

- **Perform exploratory data analysis (EDA) to understand the data's characteristics, distributions, correlations, and relationships between features**
- **Visualize data using histograms, scatter plots, box plots, etc., to gain insights and identify patterns.**

Relationship between Ambient Temperature and DC Power with month



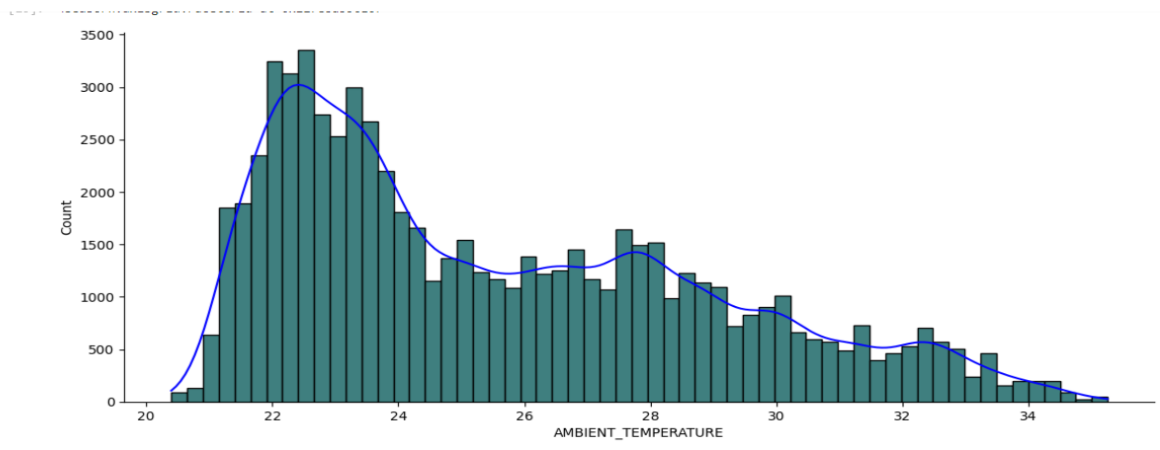
```
[17]: plt.figure(figsize=(5,3))
      sns.barplot(x='MONTH',y='AMBIENT_TEMPERATURE',data=merger)
      plt.title("Ambient temperature vs month")

[17]: Text(0.5, 1.0, 'Ambient temperature vs month')
```



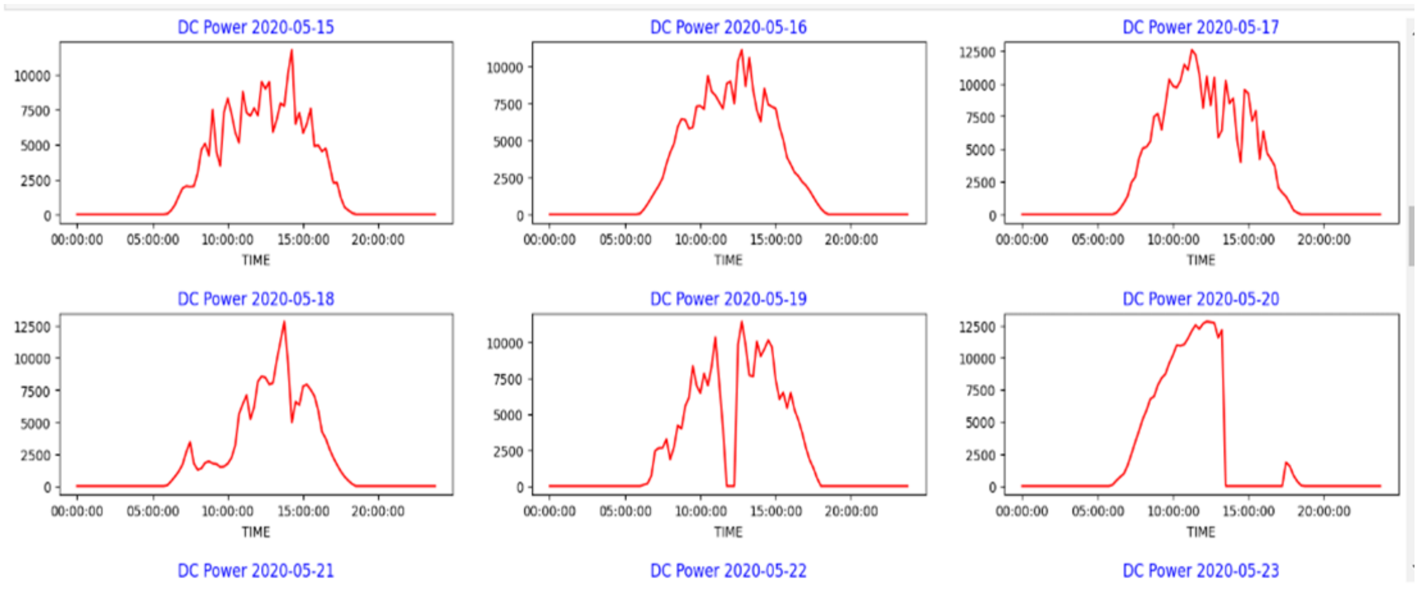
```
]: plt.figure(figsize=(5,3))
   sns.barplot(x='MONTH',y='DC_POWER',data=merger)
   plt.title("dc power vs month")

]: Text(0.5, 1.0, 'dc power vs month')
```

It is observed that ambient temperature is higher in month of May and hence high is dc power.

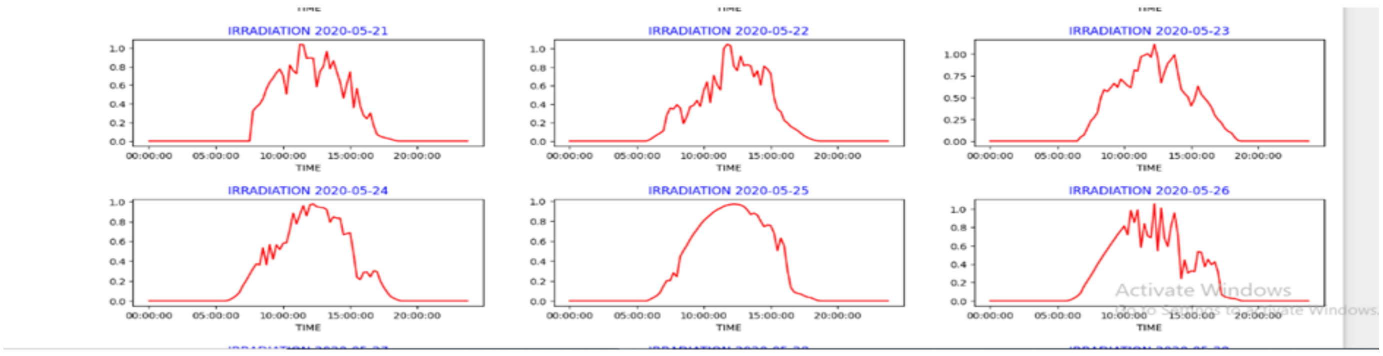
Ambient Temperature Distribution



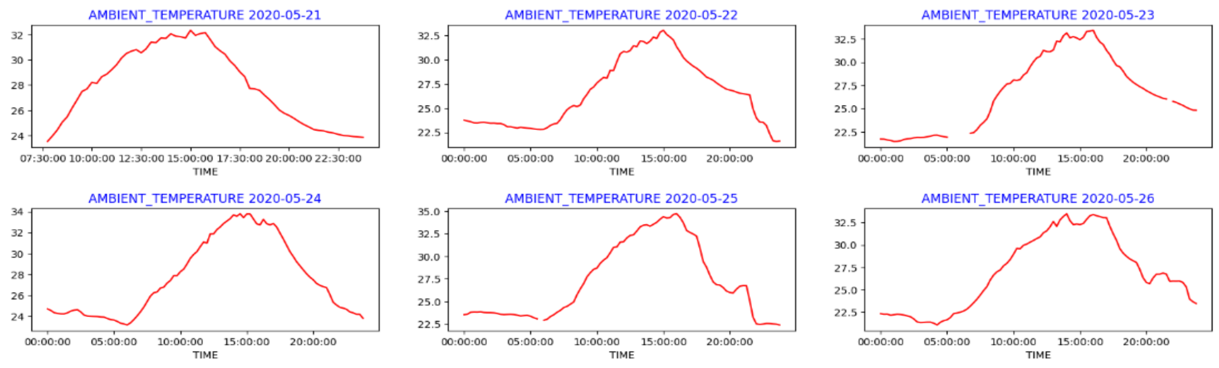
Multiple Plotting on time basis to determine abnormality



IRRADIATION



AMBIENT TEMPERATURE



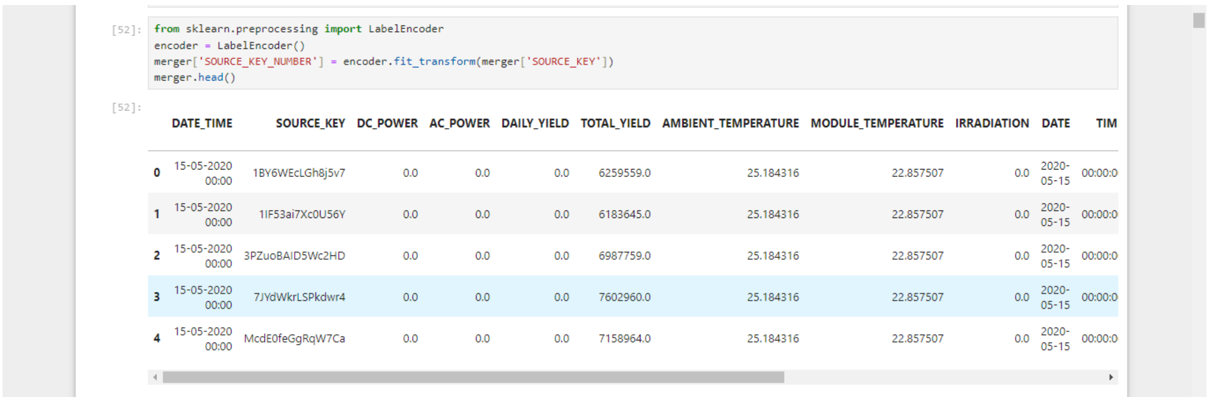
4. Feature Selection and Engineering:

- **Select relevant features that contribute most to the prediction task to reduce dimensionality and noise in the dataset.**

- **Create new features through techniques like scaling, normalization, one-hot encoding, or polynomial transformations.**
- **Use domain knowledge to engineer features that might improve model performance.**

5. Data Transformation and Encoding:

- **Encode categorical variables into numerical representations using techniques like one-hot encoding, label encoding, or embedding.**



```
[52]: from sklearn.preprocessing import LabelEncoder
      encoder = LabelEncoder()
      merger['SOURCE_KEY_NUMBER'] = encoder.fit_transform(merger['SOURCE_KEY'])
      merger.head()
```

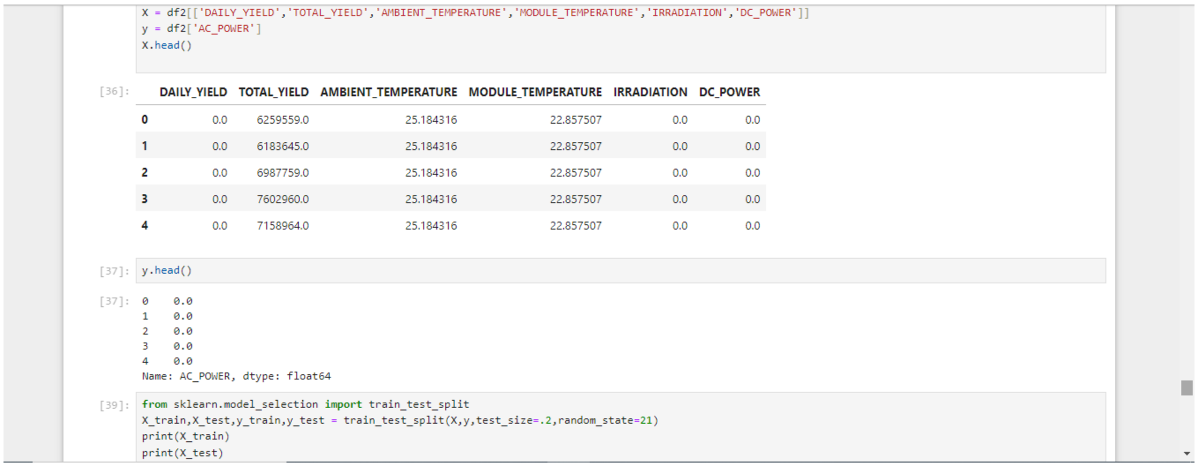
[52]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DATE	TIM
0	15-05-2020 00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0	25.184316	22.857507	0.0	2020-05-15	00:00:0
1	15-05-2020 00:00	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0	25.184316	22.857507	0.0	2020-05-15	00:00:0
2	15-05-2020 00:00	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0	25.184316	22.857507	0.0	2020-05-15	00:00:0
3	15-05-2020 00:00	7JYdWkrLSPkdwr4	0.0	0.0	0.0	7602960.0	25.184316	22.857507	0.0	2020-05-15	00:00:0
4	15-05-2020 00:00	McdE0feGgRqW7Ca	0.0	0.0	0.0	7158964.0	25.184316	22.857507	0.0	2020-05-15	00:00:0

- **Scale numerical features to a similar range using methods like standardization (scaling to a mean of 0 and standard deviation of 1) or min-max scaling.**

6. Splitting Data:

- **Split the dataset into training, validation, and test sets. Common splits include 70-30 or 80-20 for training and validation/test, respectively.**
- **Ensure that the splitting maintains the same distribution of classes or patterns present in the original dataset.**

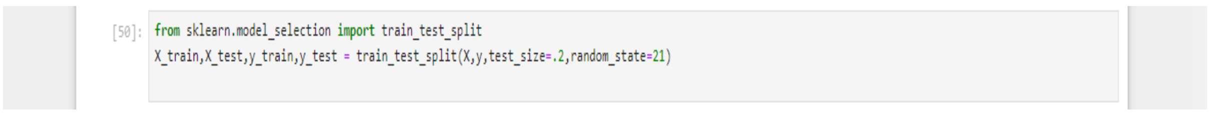


```
X = df2[['DAILY_YIELD','TOTAL_YIELD','AMBIENT_TEMPERATURE','MODULE_TEMPERATURE','IRRADIATION','DC_POWER']]
y = df2['AC_POWER']
X.head()
```

[36]:

	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DC_POWER
0	0.0	6259559.0	25.184316	22.857507	0.0	0.0
1	0.0	6183645.0	25.184316	22.857507	0.0	0.0
2	0.0	6987759.0	25.184316	22.857507	0.0	0.0
3	0.0	7602960.0	25.184316	22.857507	0.0	0.0
4	0.0	7158964.0	25.184316	22.857507	0.0	0.0

```
[37]: y.head()

[37]: 0    0.0
      1    0.0
      2    0.0
      3    0.0
      4    0.0
      Name: AC_POWER, dtype: float64

[39]: from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.2,random_state=21)
      print(X_train)
      print(X_test)
```



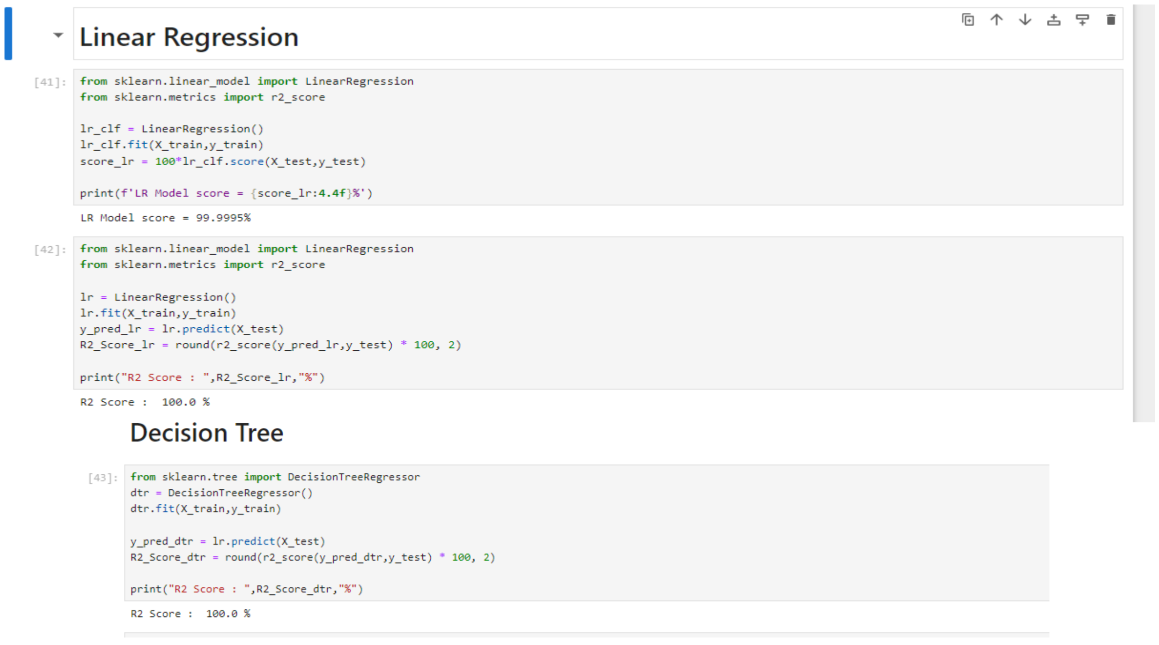
```
[50]: from sklearn.model_selection import train_test_split
      X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.2,random_state=21)
```

10. Preprocessing Pipeline:

- **Create a preprocessing pipeline to encapsulate and automate the data preparation steps, ensuring consistency across training, validation, and test data.**

11. Feature Importance Analysis:

- **Use feature importance techniques like Linear Regression, Decision Tree feature importances or L1 regularization to identify the most influential features for the model.**



Linear Regression

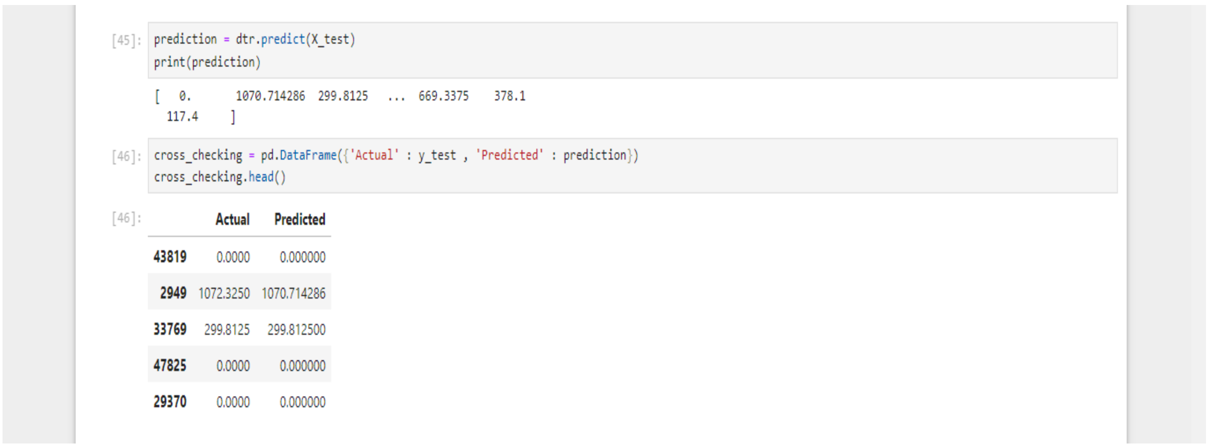
```
[41]: from sklearn.linear_model import LinearRegression
      from sklearn.metrics import r2_score

      lr_clf = LinearRegression()
      lr_clf.fit(X_train,y_train)
      score_lr = 100*lr_clf.score(X_test,y_test)

      print(f'LR Model score = {score_lr:4.4f}%')

      LR Model score = 99.9995%

[42]: from sklearn.linear_model import LinearRegression
      from sklearn.metrics import r2_score

      lr = LinearRegression()
      lr.fit(X_train,y_train)
      y_pred_lr = lr.predict(X_test)
      R2_Score_lr = round(r2_score(y_pred_lr,y_test) * 100, 2)

      print("R2 Score : ",R2_Score_lr,"%")

      R2 Score :  100.0 %
```

Decision Tree

```
[43]: from sklearn.tree import DecisionTreeRegressor
      dtr = DecisionTreeRegressor()
      dtr.fit(X_train,y_train)

      y_pred_dtr = lr.predict(X_test)
      R2_Score_dtr = round(r2_score(y_pred_dtr,y_test) * 100, 2)

      print("R2 Score : ",R2_Score_dtr,"%")

      R2 Score :  100.0 %
```

By following these steps, you can ensure that your data is well-prepared, clean, and ready to be fed into machine learning models for training and evaluation, leading to more accurate and reliable results.

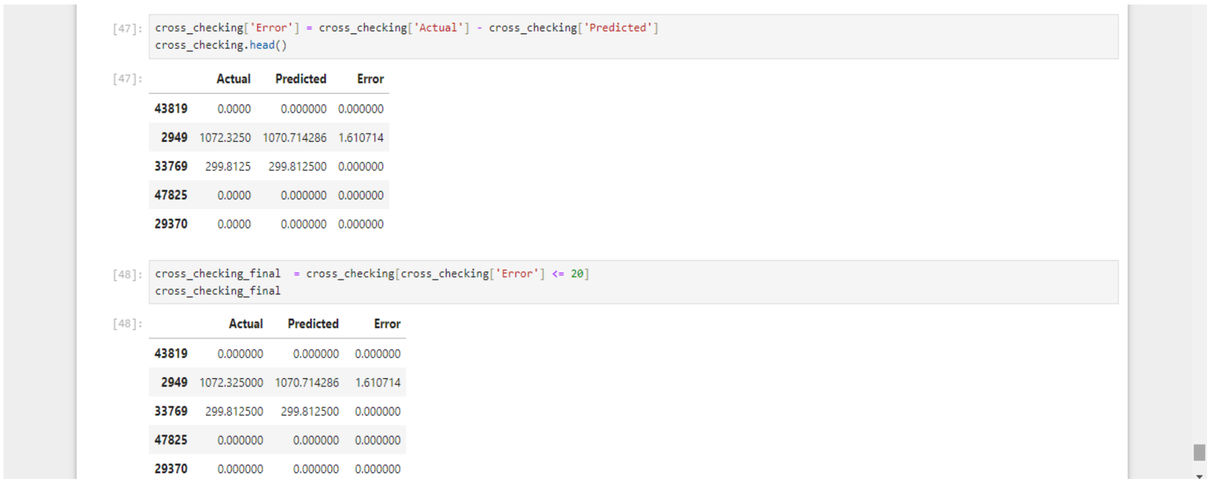
Prediction of AC POWER by decision tree algorithm



```
[45]: prediction = dtr.predict(X_test)
      print(prediction)

      [   0.        1070.714286  299.8125   ...  669.3375    378.1
        117.4     ]

[46]: cross_checking = pd.DataFrame({'Actual' : y_test , 'Predicted' : prediction})
      cross_checking.head()
```

[46]:

	Actual	Predicted
43819	0.0000	0.000000
2949	1072.3250	1070.714286
33769	299.8125	299.812500
47825	0.0000	0.000000
29370	0.0000	0.000000



```
[47]: cross_checking['Error'] = cross_checking['Actual'] - cross_checking['Predicted']
      cross_checking.head()
```

```
[47]:          Actual    Predicted     Error
      43819    0.0000     0.000000  0.000000
       2949 1072.3250  1070.714286  1.610714
      33769  299.8125   299.812500  0.000000
      47825    0.0000     0.000000  0.000000
      29370    0.0000     0.000000  0.000000
```

```
[48]: cross_checking_final  = cross_checking[cross_checking['Error'] <= 20]
      cross_checking_final
```

```
[48]:            Actual    Predicted     Error
      43819    0.000000     0.000000  0.000000
       2949 1072.325000  1070.714286  1.610714
      33769  299.812500   299.812500  0.000000
      47825    0.000000     0.000000  0.000000
      29370    0.000000     0.000000  0.000000
```

The entire process can be well understood by the code .

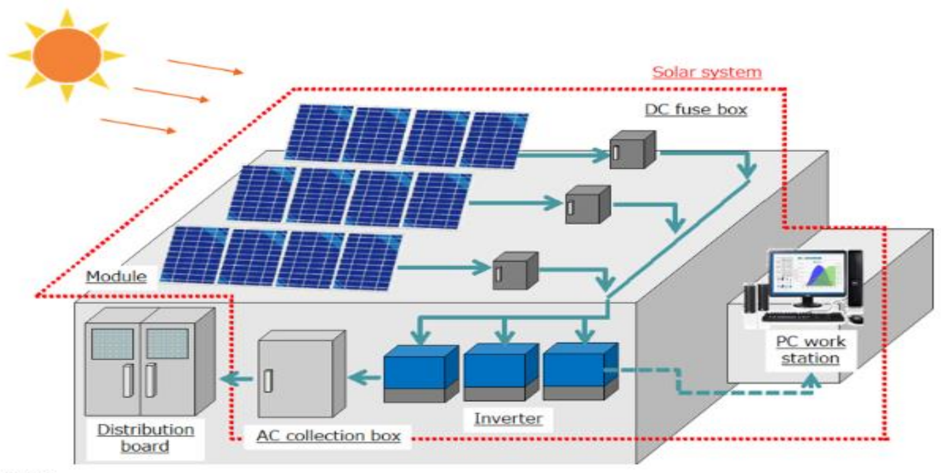
Github link: https://github.com/shreyaamukherjee/Solar-Panel-Prediction

Solar Power Generation Forecast

PV Solar Power Plant:

Photo Voltaic Solar Power has emerged as the best source of green energy in recent past in a country like India which gets a good amount of solar insolation. With the continuous development of efficient PV modules, Battery storage and Smart Grid etc. Power Generation through PV Solar Plant has gained the momentum further and has a very promising future.

The solar power plant is also known as the Photovoltaic (PV) power plant. It is a large-scale PV plant designed to produce bulk electrical power from solar radiation. The solar power plant uses solar energy to produce electrical power. Therefore, it is a conventional power plant. Solar energy can be used directly to produce electrical energy using solar PV panels.Hence, to produce electrical power on a large scale, solar PV panels are used. Below is the layout plan of photovoltaic power plant.



</center>

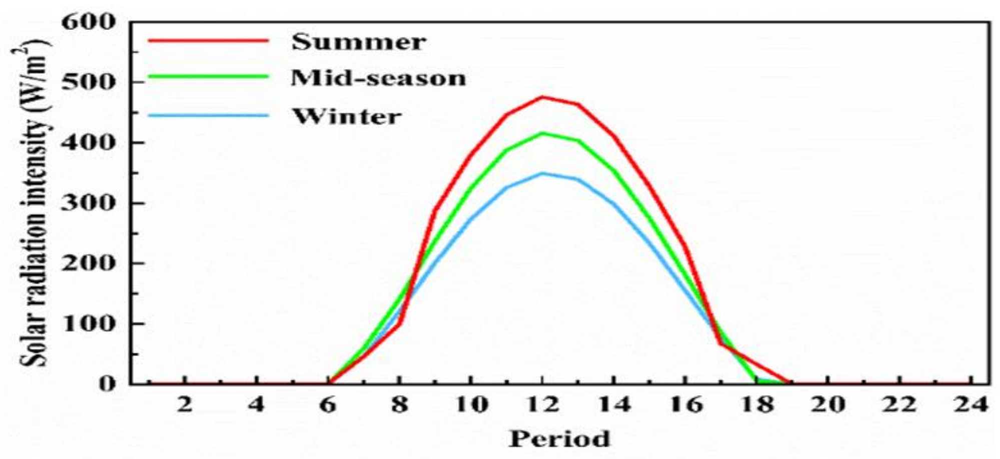
The above picture shows a typical structure of a solar power plant. Sunlight falls on PV modules, generates DC Power which is fed to the Inverters (through some Junction Box and String Monitoring Box), Inverters convert DC Power to AC Power, AC Power is stepped up through Transformers to match Grid Voltage and finally fed to the Grid through some Switchgear.

Data Description

We have at hand solar power generation data and weather data of a solar plower plants. Let's explore the given data, draw some insights, try to meet our challenges and predict/forecast the plant output to the extent possible which can be used for a better Grid Management/Stability.

Solar radiation terminology :

Peak sun hours (PSH): Daily irradiation is commonly referred to as daily PSH (or full sun hours). The number of PSH for the day is the number of hours for which power at the rate of 1kW/m2 would give an equivalent amount of energy to the total energy for that day. The terms peak sunlight hours and peak sunshine hours may also be used. Irradiation: The total quantity of radiant solar energy per unit area received over a given period, e.g. daily, monthly or annually. Insolation: Another term for irradiation. The amount of solar radiation, incident on the surface over a period of time, Peak sun hours (kWh/m2/ day) are a measurement of daily insolation. Irradiance: The solar radiation incident on a surface at any particular point in time measured in W/m2.

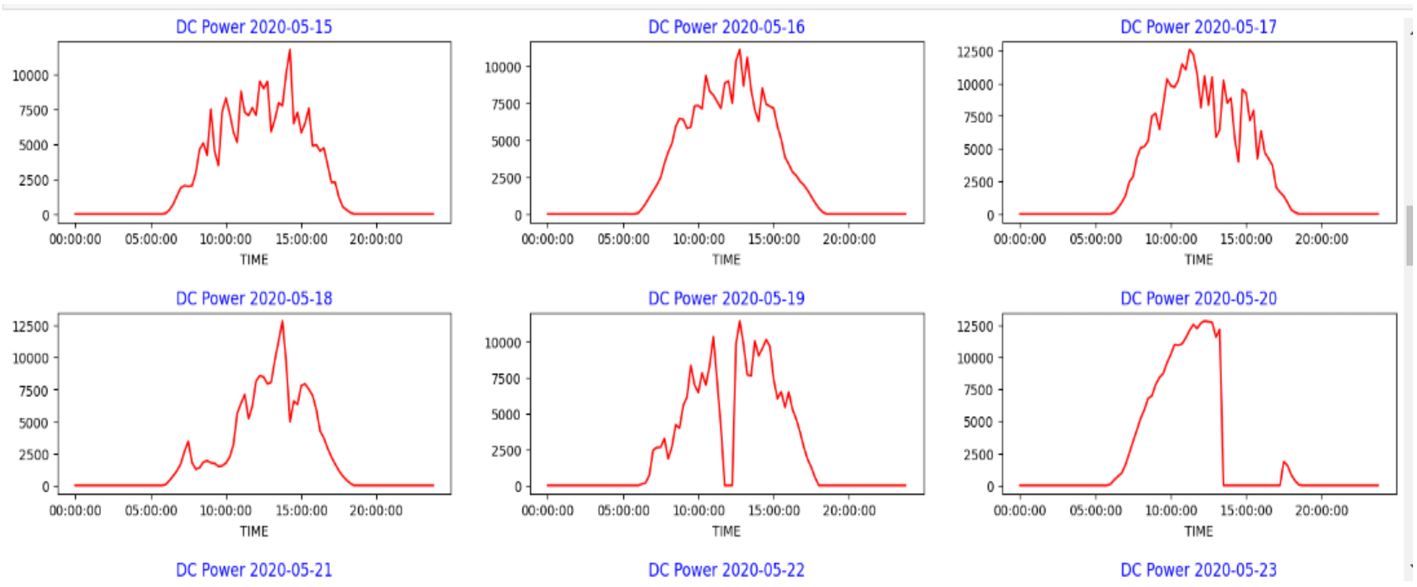


Faults & Abnormalities detection in solar power plant generation

The reason we get this shape is based on the angle of the sun to your panels. In the early morning, when the sun is still rising, it's essentially at its lowest point in the sky. In order for the sunlight to reach your panels at this time of the day, it has to cut through the most amount of atmosphere. This is important to note, because energy from the sun is absorbed when it travels through the atmosphere. So, this means that the more atmosphere it travels through, the more energy is absorbed and the less that is available for your panels to convert into electricity. At midday, when the sun is directly overhead, sunlight doesn't need to travel through as much atmosphere. Therefore, your panels will absorb more sunlight during this time.

In winter, the same concept applies. However, on gloomier winter days there may be less sunlight for your panels to absorb.

This can simply be explained by the sun being positioned lower in the sky than it is during summer.



Form the per day DC_POWER generation graph we can find that, most of the days there is a some fluctuation in the power generation.

Less Fluctuation in DC_POWER generation is observed in these days.

1. **2020-05-16**

2. **2020-05-24**
3. **2020-06-12**
4. **2020-06-13**
5. **2020-05-26**

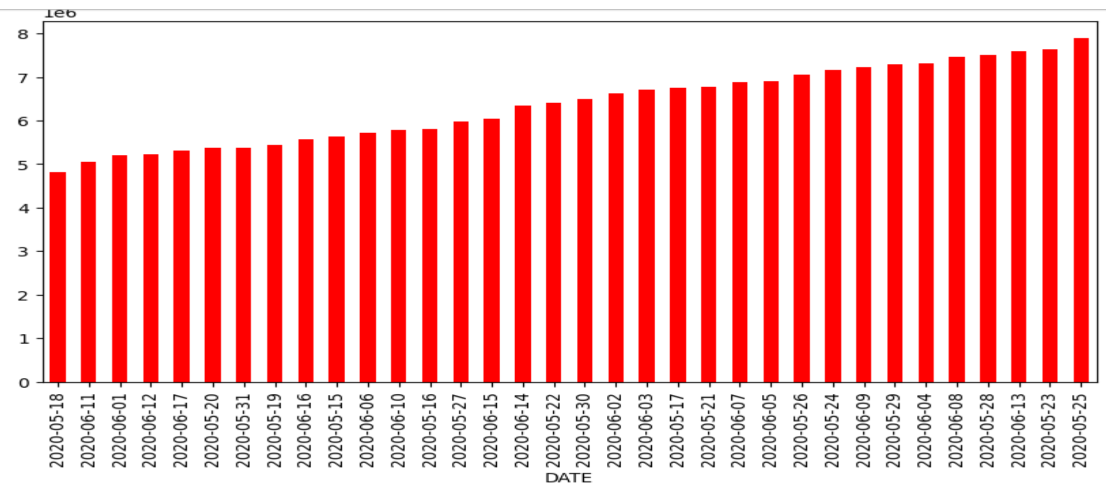
High Fluctuation in DC_POWER generation is observed in these days.

1. **2020-05-15**
2. **2020-05-17**
3. **2020-05-18**
4. **2020-05-26**
5. **2020-05-31**
6. **2020-06-01**
7. **2020-06-07**
8. **2020-06-06**
9. **2020-06-05**
10. **2020-06-10**
11. **2020-05-19**

Very High Fluctuation & Reduction in DC_POWER generation is observed in these days.

1. **2020-06-16**
2. **2020-06-15**
3. **2020-05-20**

Note: Reason for very high Fluctuation & Reduction in DC_POWER generation is due to fault in the system or may be fluctuation in weather or due to clouds etc. which need to be analyse further

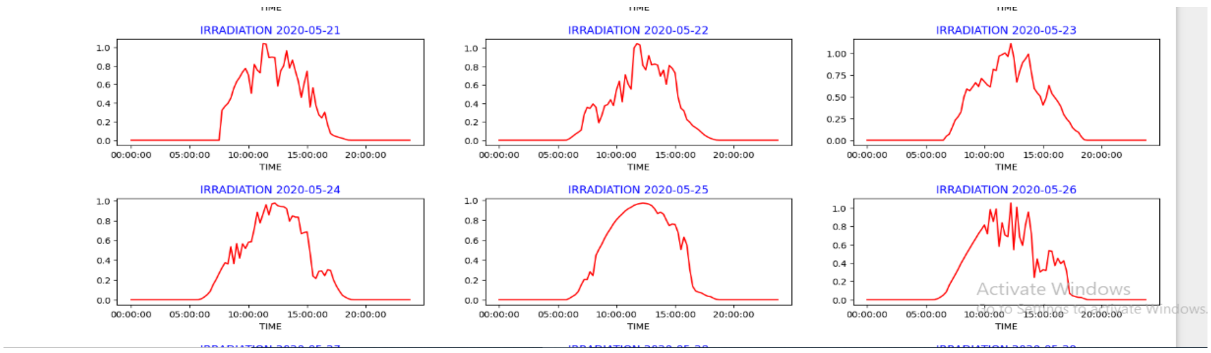


Form the per day DC_POWER generation graph we can find the average power generation per day.

***Highest average DC_POWER Generation is on*: 2020-05-25**

***Lowest average DC_POWER Generation is on* : 2020-05-18**

NOTE: This Large variation in the DC_POWER generation is due to the fault in the system or due to weather change, which needs to study further. But from this bar plot we find the day on which there is highest DC_POWER is generated and the day with the lowest DC_POWER generated.



IRRADIATION graph pattern is looking very similar to the corresponding DC_POWER generation on per day basis.

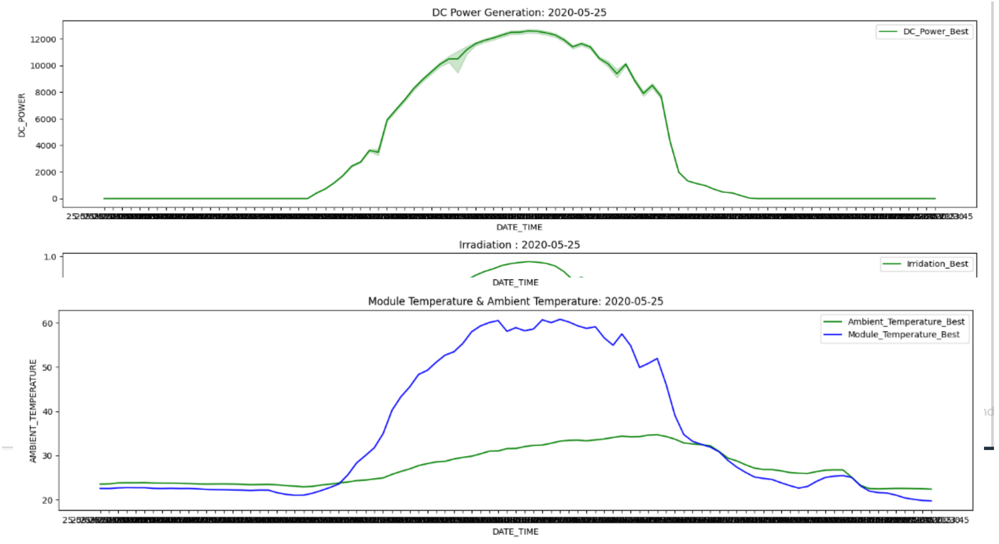
In solar power plant DC_POWER or Output power is mostly depends on IRRADIATION .Or it is not wrong to say that it's directly proportional.

Best and Worst Power generation comparision:

Major Environmental Factors affecting the of solar power generation are.

1. **The thickness of clouds is also a factor in how much sunlight your solar panels can soak up. We may see thicker clouds in winter too and this is something else to look out for. It's hard for sunlight to travel through thick clouds, which will affect your solar power system's output.**
2. **While we've looked at the sun's positioning and how it can affect output, there's another factor to consider when your system may not be performing at its maximum… even at midday.**
3. **Solar panel temperature is the number one reason behind your solar power system not achieving peak performance**
4. **Solar power generation is directly depends on Irradiation comming from the sun.**

NOTE: Both DC_POWER graph and IRRADIATION graph is almost looking like an ideal graph which is explained earlier. Weather is also looking good, and there is no cloud is in the sky because there is very less variation in IRRADIATION and temperature of the solar panel and ambient temperature.



Possible Reasons for these large fluctuation in the DC_POWER, IRRADIATION, Ambient temperature, Module temperature:

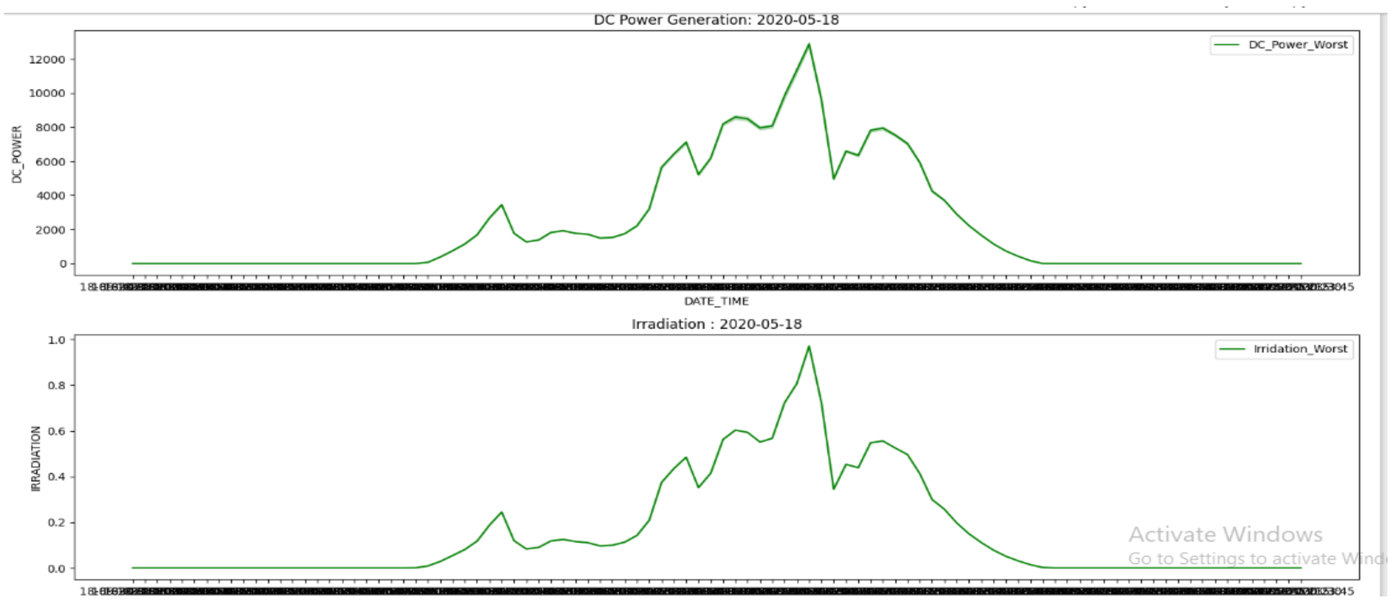
At about 12 O'clock there is a sharp decline in the DC_POWER generation from 700 to almost 20 KWatt.

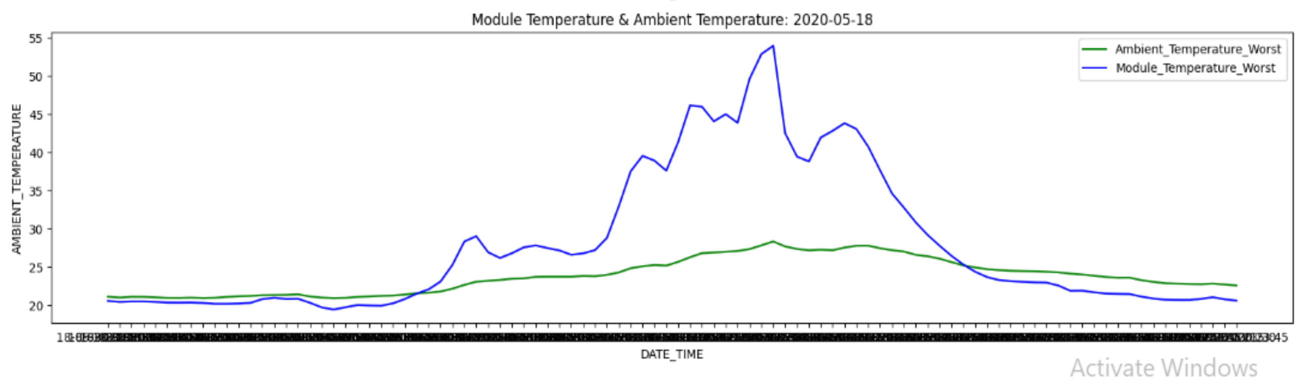
And at the same time IRRADIATION fall from 0.6 to 0.3 almost half.

Ambient temperature and Module temperature also fall drastically. Module temperature from 45 C to 35 C & Ambient temperature is also reduced.

linkcode

The possible reason for this reduction is due to may be heavy rain and heavily clouded sky and bad weather. There is almost very less possibility of any fault in the system





Solar Power Plant Inverter Efficiency Calculation

- **In fact, we shall discuss here the general power inverter efficiency whether it's solar inverter or pure sine wave inverter or even modified sine wave inverter.**
- **The inverter efficiency refers to how much dc power will be converted to ac power, as some of power will be lost during this transition in two forms:**

Heat loss.

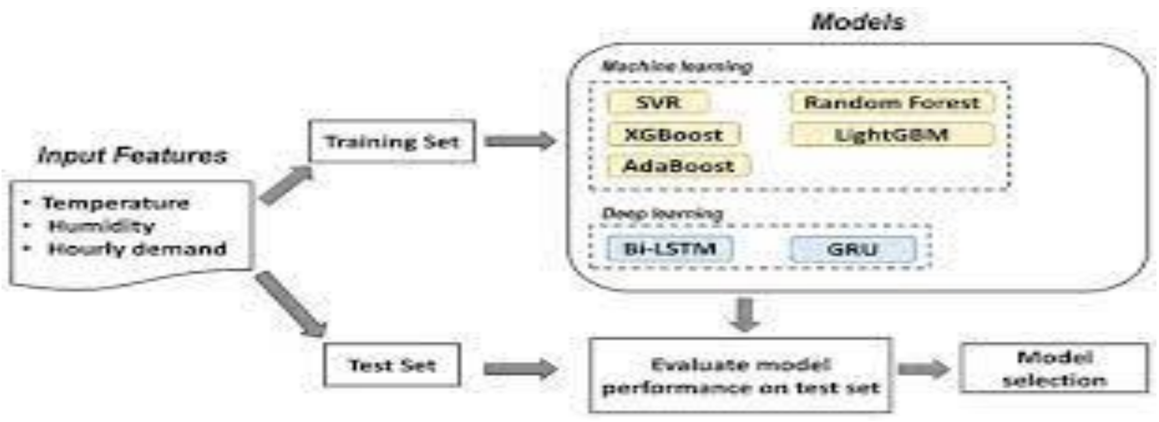
- **Stand-by power which consumed just to keep the inverter in power mode. Also, we can refer to it as inverter power consumption at no load condition.**
- **Hence, inverter efficiency = pac/pdc where pac refers to ac output power in watt and pdc refers to dc input power in watts.**

7. AI/ML APPLICATIONS IN POWER SYSTEMS

The integration of Artificial Intelligence (AI) and Machine Learning (ML) in power systems has revolutionized the way the industry operates, optimizing efficiency, reliability, and sustainability. These technologies enable power systems to adapt to dynamic demands, predict faults, and enhance decision-making processes. This report delves into various AI/ML applications within power systems, highlighting their significance and impact.

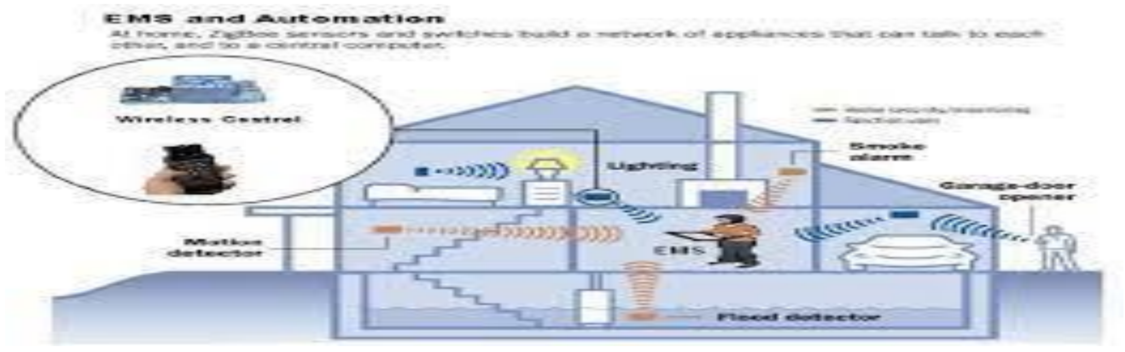
1. Load Forecasting

Load forecasting utilizes AI/ML algorithms to predict electricity demand accurately. These algorithms analyze historical consumption patterns, weather data, and socioeconomic factors to forecast future demand. This assists utilities in optimizing resource allocation, reducing wastage, and



2. Fault Detection and Diagnosis

AI/ML models enable quick detection and diagnosis of faults in power systems by analyzing real-time data. These algorithms detect anomalies, localize faults, and predict potential failures, allowing for proactive maintenance. This minimizes downtime, improves reliability, and ensures continuous power supply.

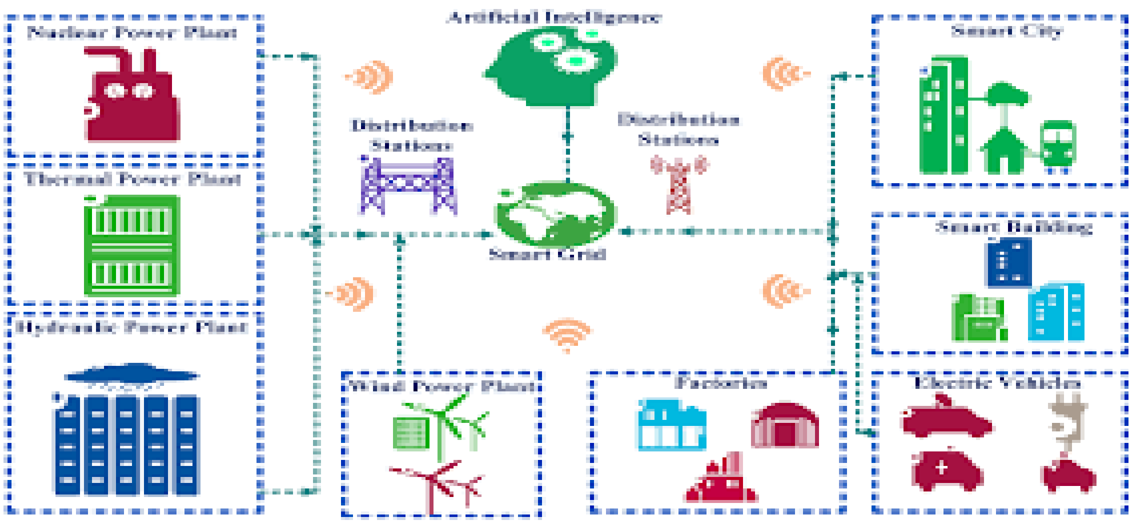

EMS and Automation
At home, ZigBee sensors and switches build a network of appliances that can talk to each other, and to a central computer.
Wireless Control
Lighting
Smoke alarm
Garage-door opener
Motion detector
EMS
Flood detector

3. Energy Management Systems

AI/ML-based energy management systems optimize energy distribution by monitoring and analyzing data from multiple sources. These systems enable dynamic pricing, demand response, and efficient utilization of renewable resources. This contributes to a more sustainable energy ecosystem by maximizing resource utilization and reducing waste.

4. Smart Grid Optimization

AI/ML algorithms optimize smart grid operations by managing power flow, voltage fluctuations, and congestion points. These technologies enhance grid resilience, enabling seamless integration of renewable energy sources and improving overall system reliability.



5. Predictive Maintenance

ML models analyze equipment data to predict potential failures, enabling proactive maintenance scheduling. This reduces maintenance costs, extends equipment lifespan, and minimizes unplanned outages, enhancing overall system reliability and performance.

6. Power Quality Enhancement

AI/ML algorithms monitor and control voltage levels, harmonic distortions, and frequency variations. These technologies ensure consistent and high-quality power delivery, improving power reliability and minimizing disruptions for consumers.

7. Renewable Energy Integration

AI/ML plays a crucial role in optimizing the integration of renewable energy sources like solar and wind power. These algorithms forecast generation patterns, enabling better grid management and maximizing the utilization of clean energy.

8. Market Price Forecasting

AI/ML models predict electricity prices based on various factors such as demand, weather, and market conditions. These forecasts assist stakeholders in making informed decisions regarding energy trading, investment, and resource allocation.

8.CONCLUSION

The incorporation of AI/ML in power systems represents a significant advancement in the energy industry. These applications optimize grid operations, enhance efficiency, and contribute to the transition towards a more sustainable energy landscape. As AI/ML technologies continue to evolve, their role in shaping the power industry will become increasingly crucial, fostering innovation and driving the adoption of smarter, more resilient, and greener energy systems.

9. REFERENCE

1. **Load Forecasting**

- **Zhang, Y., Yang, H., & Huang, X. (2018). Machine Learning Applications in Load Forecasting: A Literature Review. *IEEE Transactions on Smart Grid, 9*(4), 3991-4008. DOI: 10.1109/TSG.2017.2786383**

2. **Fault Detection and Diagnosis**

- **Li, C., Gao, W., & Zhang, X. (2020). Application of Artificial Intelligence in Fault Diagnosis of Power System Equipment. *Journal of Electrical Engineering & Technology, 15*(2), 629-643. DOI: 10.1007/s42835-020-00519-1**

3. **Energy Management Systems**

- **Chen, X., Zhang, J., & Li, P. (2019). Energy Management Systems in Smart Grids: A Review. *IEEE Access, 7*, 61848-61866. DOI: 10.1109/ACCESS.2019.2917469**

4. **Smart Grid Optimization**

- **Mahmud, A., & Chowdhury, M. E. H. (2021). Optimization of Smart Grids Using Machine Learning Techniques: A Comprehensive Review. *Sustainability, 13*(7), 3896. DOI: 10.3390/su13073896**

5. **Predictive Maintenance**

- **Wang, H., Huang, S., & Wang, F. (2019). Machine Learning Applications in Power System Maintenance: A Comprehensive Review. *Energies, 12*(18), 3594. DOI: 10.3390/en12183594**

6. **Power Quality Enhancement**

- **Mekri, F., & Zerizer, A. (2018). Power Quality Improvement Using Artificial Intelligence Techniques: A Comprehensive Review. *Energies, 11*(12), 3422. DOI: 10.3390/en11123422**

9. Machine Learning

- **https://en.wikipedia.org/wiki/Machine_learning**