

CHAPTER-1

1.0 INTRODUCTION:

Due to two main factors-the industry's orientation toward privatizations (reforms) and the shift in electricity generation toward clean, pollution-free renewable energy sources-the electricity sector, particularly in the supply industry, has undergone numerous structural and systematic changes over the past several years throughout the world. Electricity forecasting becomes one of the most crucial tasks in managing the power systems in this dynamic climate. In operation planning, scheduling, and real-time power system balance, forecasting is crucial. In today's power systems, electricity load, pricing, and renewable energy sources are the three main forecasting concerns. The solar power industry has experienced significant growth and has assumed a leadership position among the recently developed renewable sources of energy (solar energy).

Additionally, the penetration of electricity produced by renewable energy sources into the power system is growing quickly, which is seen as an alternative source of energy. Solar energy has had enormous growth among new renewable energy sources in recent years; in many nations, it represents a real replacement for fossil fuels. Additionally, the capability for producing solar energy is stochastic, intermittent, and connected to the production of other ramp occurrences. Despite this, it is a free and pollution-free source of energy, making it a popular alternative to conventional energy sources and one of the most well-established renewable energy sources.

1.1 DATA SCIENCE:

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term "data science" was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyse large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

1.2 ARTIFICIAL INTELLIGENCE:

Artificial Intelligence (AI) is a branch of computer science focused on creating systems that can perform tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, understanding natural language, and interacting with the environment.

AI systems are designed to emulate human cognitive functions, allowing them to analyze large amounts of data, recognize patterns, and make decisions or

predictions based on that data. AI algorithms can be broadly categorized into three types:

1. Narrow AI (Weak AI): Narrow AI refers to AI systems that are designed and trained for a specific task or set of tasks. These systems excel at performing a specific function, such as playing chess, driving a car, or recognizing speech. Narrow AI is the most common form of AI currently in use, powering applications like virtual assistants, recommendation systems, and image recognition software.

2. General AI (Strong AI): General AI, also known as strong AI, refers to AI systems that possess the ability to understand, learn, and apply knowledge across a wide range of tasks, similar to human intelligence

3. Artificial Superintelligence (ASI): Artificial superintelligence is an advanced form of AI that surpasses human intelligence in virtually every aspect. ASI would possess cognitive abilities far beyond those of humans and would potentially be capable of solving complex problems and making groundbreaking discoveries at an exponential rate. Achieving ASI raises ethical, societal, and existential questions about the impact of such powerful AI systems on humanity.

AI technologies rely on various techniques and approaches, including machine learning, deep learning, natural language processing, computer vision, and robotics.

The proposed method was experimentally tested using an outdoor platform with a dual-axis tracking system with bifacial modules. The effective irradiance was calculated using the novel method presented nRMSE of 2.88%, relative to the results obtained using the consolidated method. The bifacial gains obtained were 6.24% and 6.69%, respectively, using the proposed and traditional calculation methods.

CHAPTER-2

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms.

Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning.

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig 2.1 Machine Learning process

Supervised Machine Learning **is the** majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output **is $y = f(X)$.**

The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression, multi-class classification, Decision Trees and support vector machines etc.**

Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables.

Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

Machine learning algorithms can be used for predictive analytics to forecast future outcomes based on historical data. In this approach, the model is trained on unlabeled data and tasked with finding patterns or structure within the data.

CHAPTER-3

3.0 PREPARING THE DATASET:

This Dataset contains 8760 records from various weather features which can be analyzed and used as predictors. It Contains

- 1. System Power Generated:** Power generating systems are generally treated as heat engines to convert heat input into work, hence to produce electricity at a sustained rate. Heat input is supplied by burning fossil fuels (coal, oil and natural) and biomass, or processing nuclear fuel, or harvesting thermal energy from renewable energy sources.
- 2. Solar Speed:** At a distance of more than a few solar radii from the Sun, the solar wind reaches speeds of 250–750 km/s and is supersonic, meaning it moves faster than the speed of the fast magnetosonic wave. The flow of the solar wind is no longer supersonic at the termination shock.
- 3. Solar Direction:** Since the sun is always in the southern half of the sky (in the northern hemisphere), solar panels that face south will receive the most direct sunlight and, therefore, is the best direction for solar panels.
- 4. Pressure:** Solar radiation pressure is due to the Sun's radiation at closer distances, thus especially within the Solar System. (The radiation pressure of sunlight on Earth is very small: it is equivalent to that exerted by about a milligram on an area of 1 square metre, or $10 \mu\text{N/m}^2$, or 10^{-10} atmospheres.)
- 5. Air Temperature:** The temperature of the air is a measure of how quickly the molecules are moving. The more energy of motion the molecules have, the higher the temperature you feel in the air. Air temperature is measured with thermometers. Common thermometers consist of a glass rod with a very thin tube in it.

CHAPTER-4

4.0 PROPOSED SYSTEM:

Solar radiation is the Earth's primary source of energy and has an important role in the surface radiation balance, hydrological cycles, and vegetation photosynthesis, and weather and climate extremes. Scikit-learn is a popular machine learning library in Python, but it primarily focuses on traditional machine learning algorithms and does not directly provide functionality for solar power generation or solar radiation prediction. However, you can use scikit-learn in conjunction with other libraries and techniques to build models for solar power generation and solar radiation prediction. Gather historical solar power generation data and solar radiation data from reliable sources. Make sure the data includes relevant features such as time of day, weather conditions, temperature, humidity, and any other factors that might affect solar power generation or solar radiation. Machine learning models for solar power generation or solar radiation prediction. Scikit-learn offers a wide range of models, including regression models, ensemble methods (such as random forests or gradient boosting), or even neural networks. It's worth mentioning that while scikit-learn can be used for certain aspects of the pipeline, such as data pre-processing, model training, and evaluation, other libraries or techniques may be more suitable for solar power generation and solar radiation prediction.

MERITS:

- We compared more than a two algorithms to getting better accuracy level.
- We figured out solar power generation & radiation.
- We deploy the model into production level application.
- We improved the accuracy level and performance level.
- We implement machine learning techniques for effective manner.

CHAPTER-5

5.0 LITERATURE SURVEY:

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization or reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Solar energy currently plays a significant role in supplying clean and renewable electric energy worldwide. Harnessing solar energy through PV plants requires problems such as site selection to be solved, for which long-term solar resource assessment and photovoltaic energy forecasting are fundamental issues. This paper proposes a fast-track methodology to address these two critical requirements when exploring a vast area to locate, in a first approximation, potential sites to build PV plants.

Then, PV energy is calculated, including heat losses. Finally, PV energy forecasting is accomplished by constructing the P50 and P95 estimations of the mean yearly PV energy. A case study in Mexico fully demonstrates the

methodology using hourly data from 2000 to 2020 from NSRDB. The proposed methodology is validated by comparison with actual PV plant generation throughout the country

India is a fast developing country and in population we are nearer to China, In India there will be global future of the solar energy markets. Energy sources classify into renewable energy and nonrenewable energy sources due to hazardous effect on environment in recent years the Government of India is promoting the production of green energy through National Solar Energy Mission-2020. India 2020 energy policy reports states that India is implementing the use of solar energy for electricity generation with wind energy. It is considered that India is one of the countries in the world who ensure full access to electricity, bringing power to more than 700 million people since 2000.

Now India is planning to reduce load on conventional power plant by using solar thermal system like photovoltaic system. India can make its significant progress in reducing the use of traditional biomass in cooking and traditional fuel for electricity generation. Indian government promotes the use of solar pump for irrigation and solar roof system for electricity generation which works on solar energy. This paper gives literature review on solar energy systems across the universe and its advantages to the customer of renewable energy sources.

CHAPTER-6

SYSTEM STUDY:

6.1 Aim:

Every grid operator has the same issue with their successful integration to meet current demand grid operator now also has to forecast the availability of solar and solar generation facilities in the upcoming hour, day. So, this project can easily find out the Solar Power Generation.

6.2 Objectives:

The goal is to develop a machine learning model for solar power generation prediction, to potentially replace the updatable supervised machine Learning Classification models by predicting results in the form of best accuracy by comparing supervised algorithms.

6.3 Feasibility study:

- 1) **Data Wrangling:** In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis.
- 2) **Data collection:** This Dataset contains 8760 records from various weather features which can be analysed and used as predictors. It Contains
 - System Power Generated
 - Solar Speed
 - Solar Direction
 - Pressure
 - Air Temperature

6.4 Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. The outliers have to be removed and also variable conversion need to be done.

6.5 Building the regression model

The prediction of tesla stock price, a high accuracy prediction model is effective because of the following reasons: It provides better results in regressions problem.

- It is strong in preprocessing outliers, irrelevant variables,
- It produces out of bag estimate .

6.6 Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then Tuned model involved by tuned time to time with improving the accuracy.

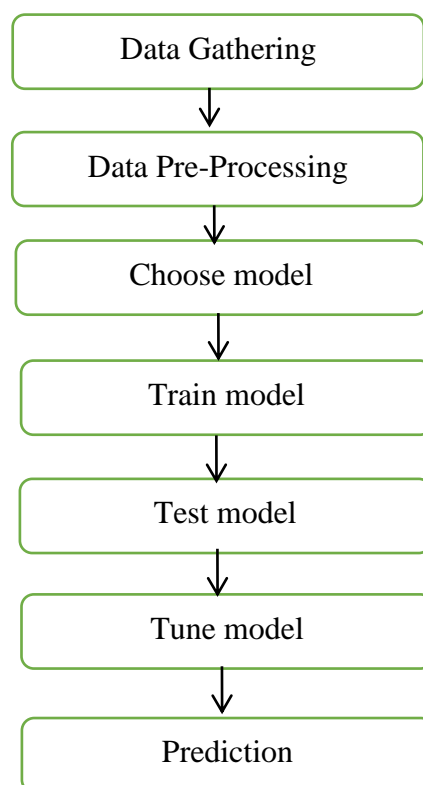


Fig6.7 Process of Dataflow Diagram

CHAPTER-7

7.0 PROJECT REQUIREMENTS:

GENERAL:

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements

A. Hardware requirements

B. software requirements

7.1 Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

7.2 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

7.3 Environmental Requirements:

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3 or later

Hard disk : minimum 10 GB

RAM : minimum 4 GB

CHAPTER-8

8.1 SOFTWARE DESCRIPTION

Anaconda is a free and open-source of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently Pip packages provide many of the features of conda packages and in most cases they can work together. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

8.2 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

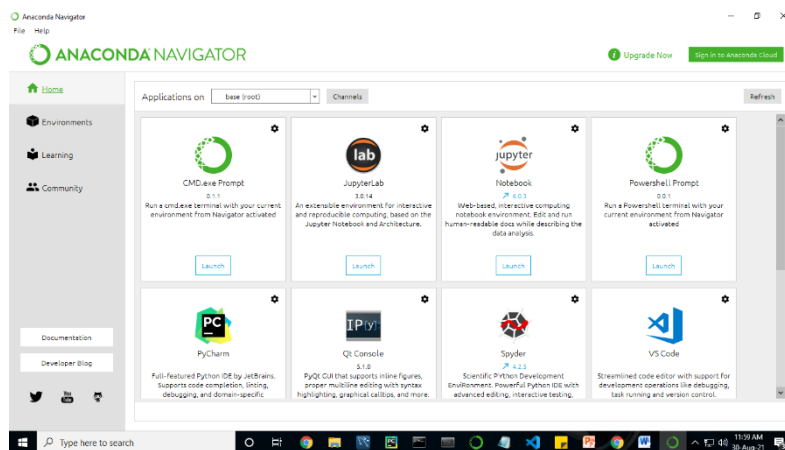


FIG.8.2 Anaconda navigator

8.3 Conda:

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are

doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. These softwares have great graphical user interface and these will make your work easy to do. You can also use it to run your python script. These are the software carried by anaconda navigator.

8.3 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

Running the Jupyter Notebook

Launching *Jupyter Notebook App*: The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app
- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- Click on the menu *Help* -> *User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing *shift* + *enter*.
- You can run the whole notebook in a single step by clicking on the menu *Cell* -> *Run All*.
- To restart the kernel (i.e. the computational engine), click on the menu *Kernel* -> *Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

8.4 JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser.

The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

Kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.

Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

8.5 Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.

- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

- Installing the Python anaconda platform.
- Loading the dataset.
- Summarizing the dataset.
- Visualizing the dataset.
- Evaluating some algorithms.
- Making some predictions.

CHAPTER-9

PYTHON PROGRAM:

9.1 Introduction:

Python is a high-level, interpreted programming language known for its simplicity, versatility, and readability. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability with its clear and expressive syntax, making it an ideal language for beginners and experienced developers alike.

Here's a brief introduction to some key aspects of Python:

1. Readability: Python's syntax is designed to be intuitive and easy to read, with a focus on simplicity and clarity. This makes Python code highly maintainable and accessible to developers of all skill levels.

2. Interpreted: Python is an interpreted language, which means that code is executed line by line by an interpreter without the need for compilation. This allows for rapid development and prototyping, as code changes can be immediately tested without a lengthy compilation step.

3. Multiparadigm: Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the most appropriate approach for solving a particular problem.

4. Dynamic typing: Python is dynamically typed, meaning that variable types are determined at runtime. This allows for greater flexibility and simplicity in coding, but it also requires careful attention to type compatibility to avoid runtime errors.

5. Extensive standard library: Python comes with a comprehensive standard library that provides support for a wide range of tasks, from file I/O and networking to data processing and web development. This rich set of modules and packages simplifies development by providing ready-to-use solutions for common programming tasks.

6. Large ecosystem: In addition to the standard library, Python has a vast ecosystem of third-party libraries and frameworks developed by the community. These libraries cover a wide range of domains, including web development (e.g., Django, Flask), scientific computing (e.g., NumPy, pandas), machine learning (e.g., TensorFlow, scikit-learn), and more.

7. Cross-platform: Python is cross-platform, meaning that Python code can run on various operating systems without modification. This makes it a versatile choice for developing applications that need to run on different platforms, such as Windows, macOS, and Linux.

Overall, Python's simplicity, versatility, and robust ecosystem make it an excellent choice for a wide range of programming tasks, from simple scripts to complex web applications and machine learning models.

9.2 Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

9.3 Statements and control flow:

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

9.4 PYTHON EXPRESSION:

In Python, an expression is a combination of values, variables, operators, and function calls that are evaluated to produce a single value. Expressions can be simple or complex and can involve arithmetic operations, comparison operations, logical operations, function calls, and more.

Here's a breakdown of the components typically found in a Python expression:

1. **Values:** These are the basic building blocks of expressions. Values can be literals (such as numbers or strings) or variables that hold values.
2. **Operators:** Operators are symbols that perform operations on one or more operands. Python supports various types of operators, including arithmetic operators (+, -, *, /), comparison operators (==, !=, <, >), logical operators (and, or, not), assignment operators (=, +=, -=), and more.
3. **Variables:** Variables are used to store values that can be referenced and manipulated in expressions. They can hold different types of data, such as numbers, strings, lists, or objects.

4. **Function Calls:** Functions are reusable blocks of code that perform specific tasks. Function calls can be included in expressions to perform computations or return values based on input arguments.
5. **Grouping Parentheses:** Parentheses can be used to group sub-expressions and specify the order of evaluation in complex expressions. They help clarify the intended meaning and precedence of operations.

In Python, expressions are evaluated using the Python interpreter, which follows specific rules for operator precedence and associativity to determine the order of evaluation. The result of evaluating an expression is a single value, which can then be used in assignments, function calls, conditions, and other parts of a Python program.

9.5 Methods:

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called *dunder methods* due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

Typing:

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite

being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `Spam Class()` or `Eggs Class()`), and the classes are instances of the meta class type (itself an instance of itself), allowing meta-programming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, C ,Python. An experimental optional static type checker named mypy supports compile-time type checking.

CHAPTER-10

10.0 SYSTEM ARCHITECTURE:

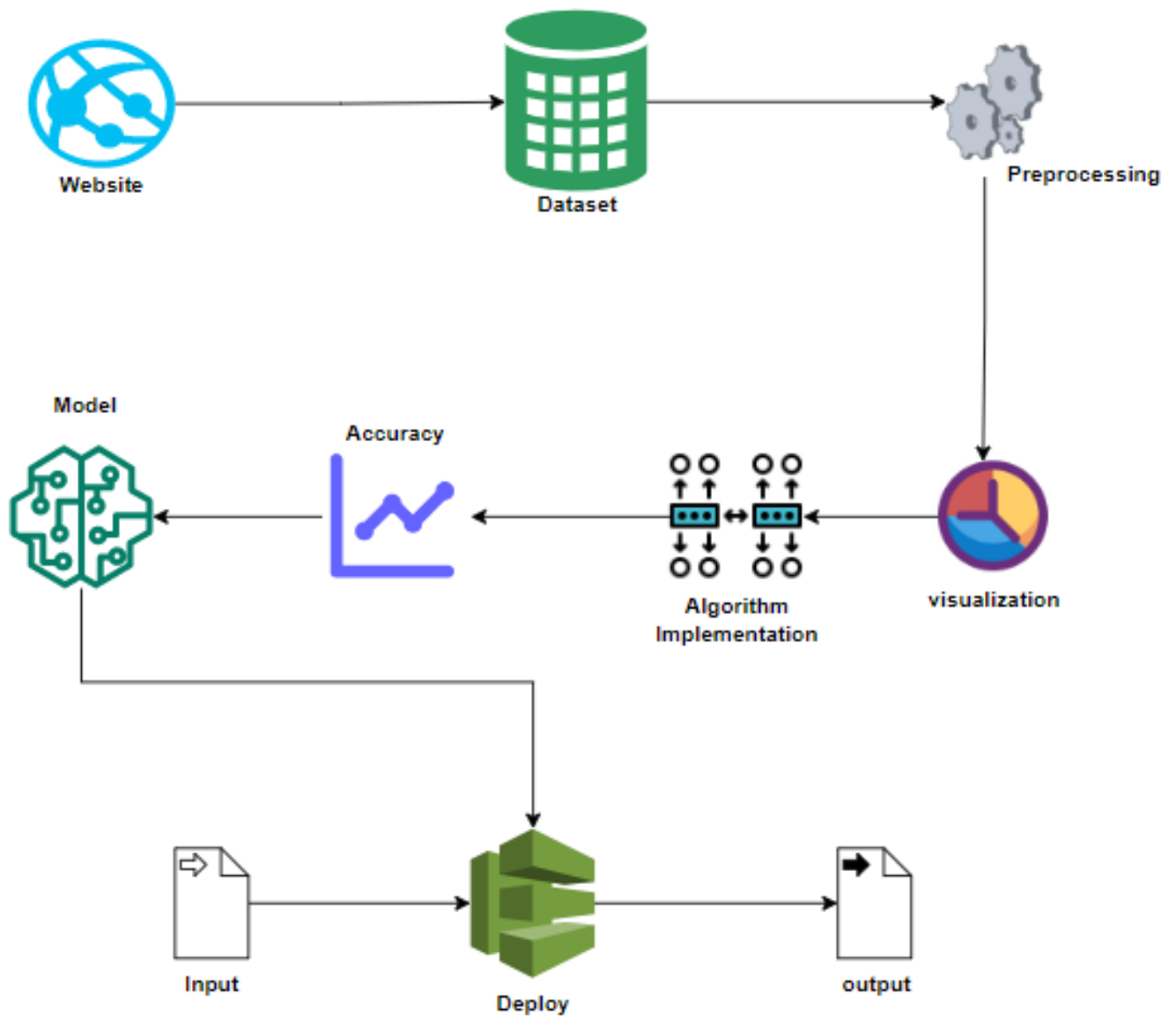


FIG.10.0 System Architecture

10.2 WORK FLOW DIAGRAM

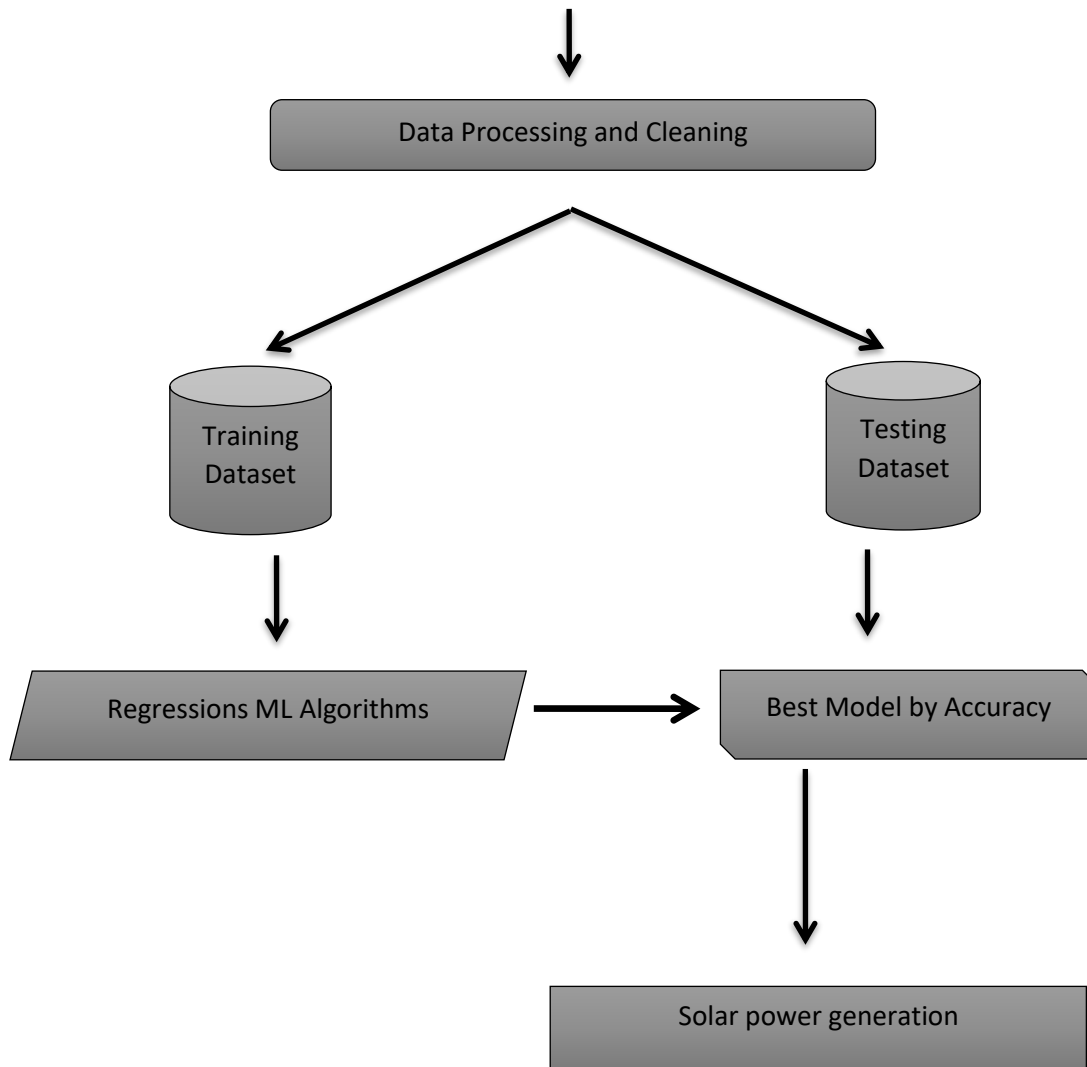


FIG.10.2 Workflow Diagram

10.3 CLASS DIAGRAM

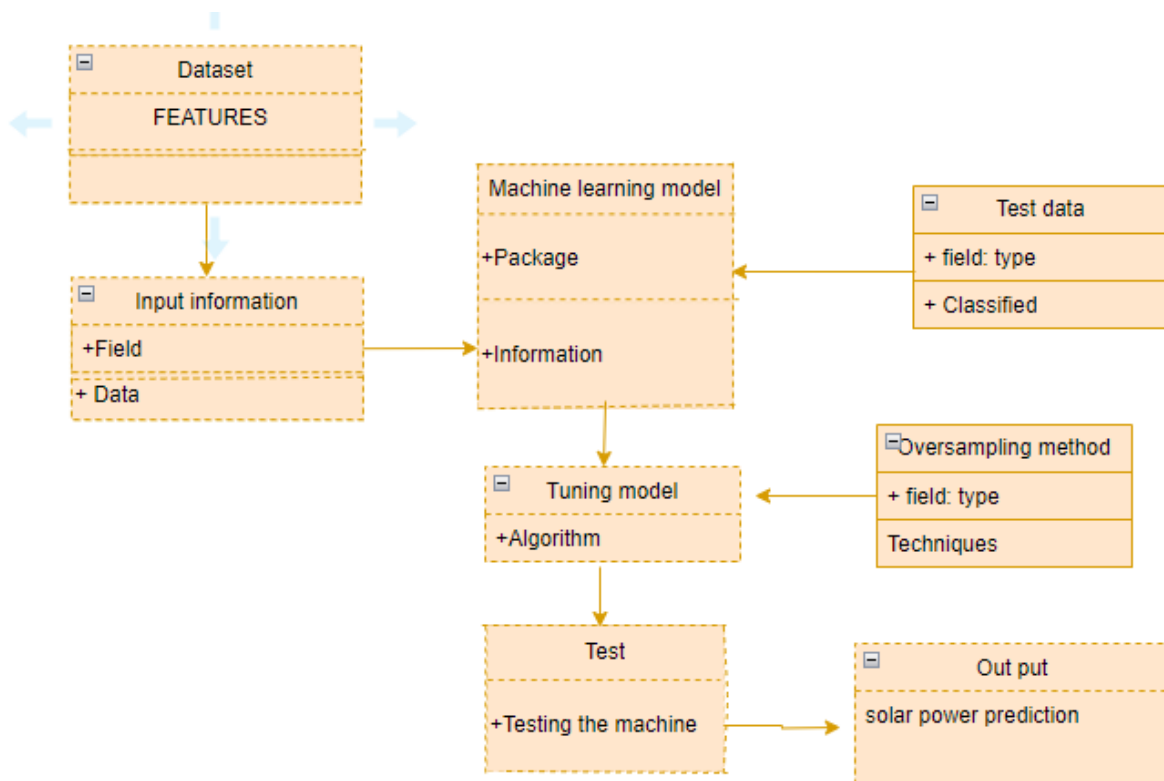


FIG.10.3 Class Diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

10.4 SEQUENCE DIAGRAM:

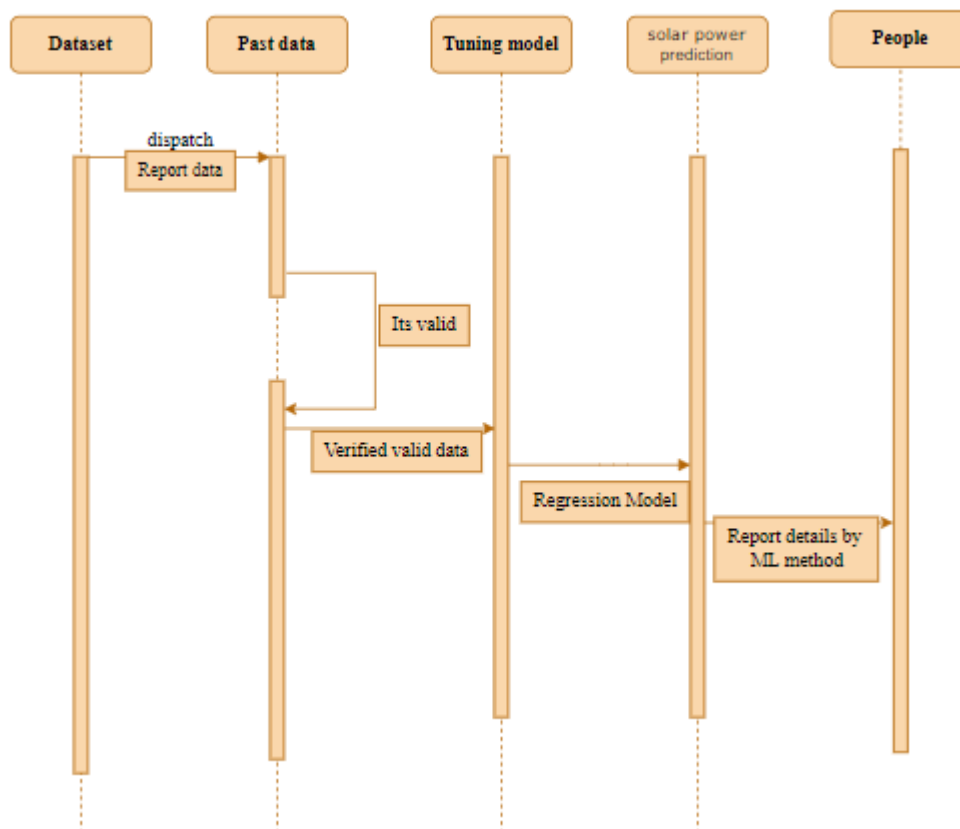


Fig 10.4 Sequence Diagram

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

10.5 Entity Relationship Diagram (ERD):

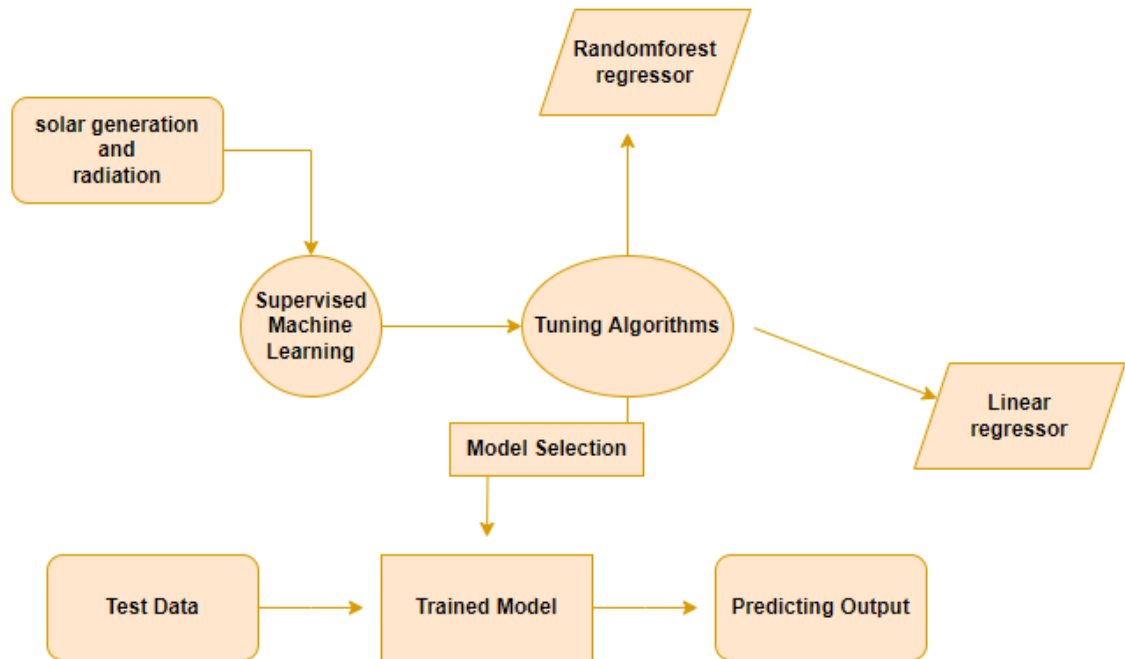


Fig.10.5 ER-Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

CHAPTER-11

11.0 LIS OF MODULE

- Data Pre-processing
- Data visualization
- Algorithm implementation

11.1 DATA PRE-PROCESSING:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different **data cleaning** tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more quickly clean data**. It wants to **spend less time cleaning data**, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame

- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

	Day of Year	Year	Month	Day	First Hour of Period	Distance to Solar Noon	Average Temperature (Day)	Average Wind Direction (Day)	Average Wind Speed (Day)	Sky Cover	Visibility
count	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000
mean	183.282631	2008.665982	6.524495	15.715999	11.501542	0.503327	58.468996	24.957862	10.099486	1.987667	9.559609
std	105.751253	0.471727	3.448038	8.795825	6.875714	0.298069	6.842318	6.912203	4.837128	1.412220	1.380290
min	1.000000	2008.000000	1.000000	1.000000	1.000000	0.050401	42.000000	1.000000	1.100000	0.000000	0.000000
25%	92.000000	2008.000000	4.000000	8.000000	5.500000	0.232061	53.000000	25.000000	6.600000	1.000000	10.000000
50%	183.000000	2009.000000	7.000000	16.000000	13.000000	0.479241	59.000000	27.000000	10.000000	2.000000	10.000000
75%	275.000000	2009.000000	10.000000	23.000000	17.500000	0.739559	63.000000	29.000000	13.100000	3.000000	10.000000
max	366.000000	2009.000000	12.000000	31.000000	22.000000	1.141361	78.000000	36.000000	26.600000	4.000000	10.000000

Fig.11.1 previous Data

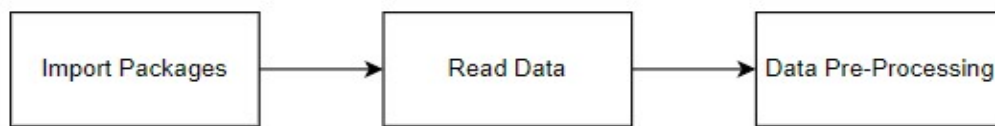
```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 0 to 2919
Data columns (total 16 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Day of Year                             2919 non-null   int64
1   Year                                    2919 non-null   int64
2   Month                                  2919 non-null   int64
3   Day                                    2919 non-null   int64
4   First Hour of Period                    2919 non-null   int64
5   Is Daylight                             2919 non-null   bool
6   Distance to Solar Noon                  2919 non-null   float64
7   Average Temperature (Day)               2919 non-null   int64
8   Average Wind Direction (Day)            2919 non-null   int64
9   Average Wind Speed (Day)                2919 non-null   float64
10  Sky Cover                               2919 non-null   int64
11  Visibility                              2919 non-null   float64
12  Relative Humidity                       2919 non-null   int64
13  Average Wind Speed (Period)              2919 non-null   float64
14  Average Barometric Pressure (Period)     2919 non-null   float64
15  Power Generated                          2919 non-null   int64
dtypes: bool(1), float64(5), int64(10)
memory usage: 367.7 KB

```

Fig.11.1 Data pre-processing

MODULE DIAGRAM:



11.1:2 GIVEN INPUT EXPECTED OUTPUT

Input : Data

Output : Removing Noisy Data

11.2 DATA VISUALIZATION:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

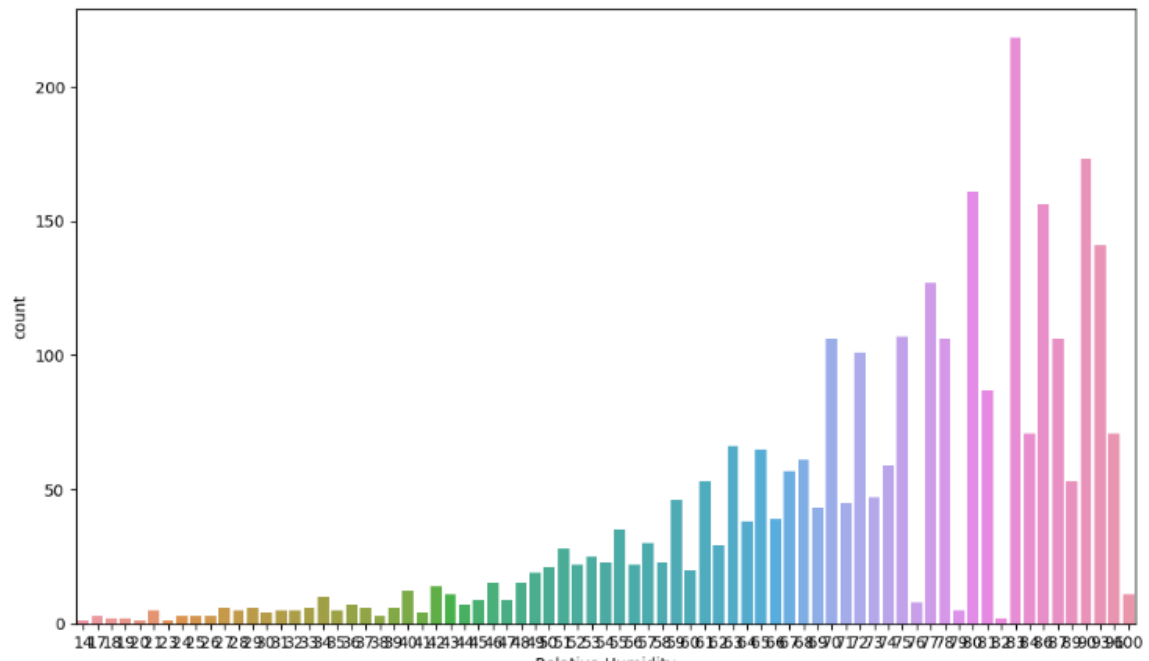
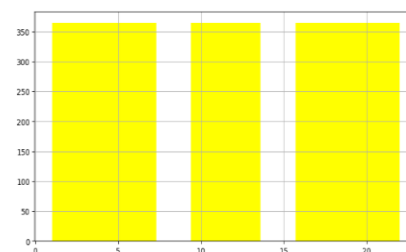
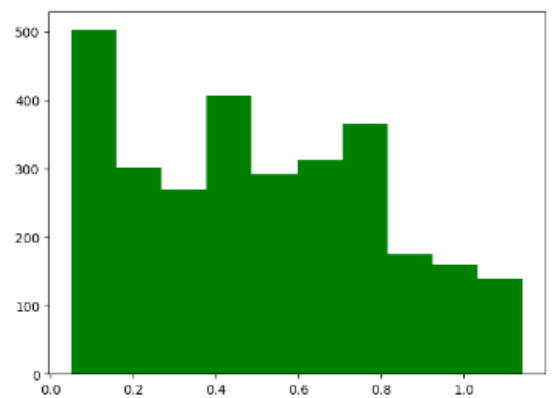
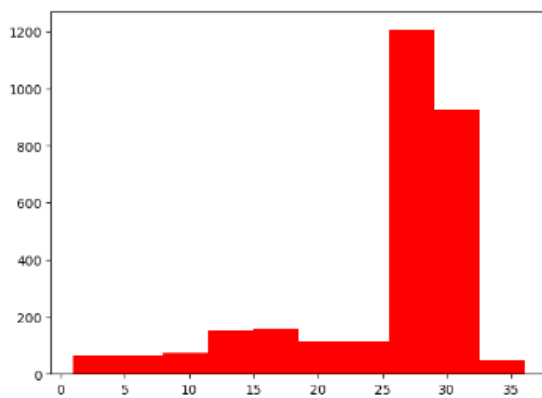
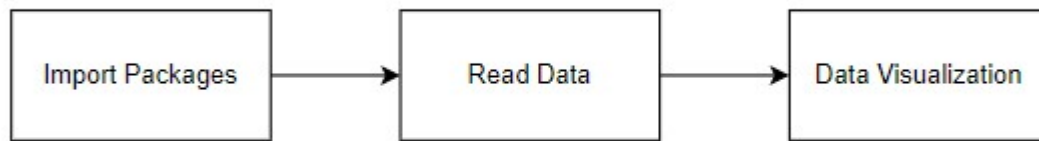


Fig.11.2 Data visualization

<BarContainer object of 10 artists>



MODULE DIAGRAM:



12.2:1GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

CHAPTER-12

12.0 Algorithm implementation:

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize.

A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

12.1 Performance Metrics to calculate:

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class is no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F-Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

The below 2 different algorithms are compared:

- Random Forest Regression
- Linear Regression

12.2 RANDOM FOREST:

Random Forest is an ensemble learning algorithm that can be used for both classification and regression tasks. It builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Here's a step-by-step explanation of how Random Forest works:

Bootstrapping: Random Forest starts by creating multiple subsets of the original dataset through a process called bootstrapping. This involves randomly sampling the data with replacement, creating new datasets of the same size as the original.

Building Decision Trees: For each subset, a decision tree is constructed. Decision trees are built by selecting the best feature from a random subset of features at each node, considering various criteria such as Gini impurity for classification or mean squared error for regression.

Reducing Overfitting: One of the key advantages of Random Forest is that it reduces overfitting. Each tree in the forest is trained on a different subset of the data, and by averaging or voting, the model becomes more robust and less prone to the noise present in individual trees.

Feature Importance: Random Forest provides a measure of feature importance. Features that are more frequently used for splitting in the trees are considered more important. This information can be valuable for understanding the underlying patterns in the data.

Tuning Parameters: Random Forest has several hyperparameters that can be tuned to optimize its performance, including the number of trees in the forest, the maximum depth of each tree, and the number of features considered at each split.

Advantages of Random Forest:

- Reduces overfitting.
- Handles missing values well.
- Provides feature importance.

- Can handle large datasets with high dimensionality.

1. Decision Trees:

- Random Forest is built upon the foundation of decision trees. Decision trees are simple models that recursively split the data based on features to make predictions.

2. Ensemble Learning:

- Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve overall performance and robustness.

3. Bagging (Bootstrap Aggregating):

- Random Forest uses bagging to build diverse trees. It randomly selects subsets of the training data (with replacement) to train each tree. This helps in reducing overfitting and capturing different patterns in the data.

4. Random Feature Selection:

- In addition to using random subsets of data, Random Forest also randomly selects a subset of features for each split in a tree. This introduces further randomness, prevents overfitting, and ensures that each tree focuses on different aspects of the data.

5. Decision Tree Training:

- Each decision tree in the Random Forest is trained independently on its subset of data. The training process involves recursively splitting the data based on features, considering the best split at each node according to a specified criterion (commonly Gini impurity or information gain).

6. Voting (Classification):

- For classification tasks, each tree in the forest "votes" for a class. The class with the most votes becomes the final prediction of the Random Forest.

7. Averaging (Regression):

- For regression tasks, each tree predicts a numerical value, and the final prediction is the average of these values.

8. Out-of-Bag (OOB) Score:

- Because each tree is trained on a subset of the data, some data points are not included (out-of-bag) in the training of certain trees. The OOB score is calculated by evaluating each tree on the data it did not see during training. This provides an estimate of the model's performance without the need for a separate validation set.

9. Hyperparameter Tuning:

- Random Forest has hyperparameters such as the number of trees, the depth of each tree, and the number of features considered at each split. Proper hyperparameter tuning is crucial for achieving optimal performance.

10. Feature Importance:

- Random Forest provides a feature importance score, indicating the contribution of each feature to the model's predictions. This information is valuable for feature selection and understanding the impact of variables on the model.

11. Parallel Training:

- Random Forest is suitable for parallel processing, as each tree can be trained independently. This makes it computationally efficient and scalable.

12. Robustness to Overfitting:

- The ensemble nature of Random Forest, combined with the randomness introduced during training, makes it less prone to overfitting compared to individual decision trees.

13. Applications:

- Random Forest is widely used in various domains, including finance, healthcare, and image classification. Its versatility and ability to handle high-dimensional data make it a popular choice for machine learning tasks.

14. Scikit-learn Implementation:

- In Python, the scikit-learn library provides a Random Forest Classifier implementation

Disadvantages:

- Can be computationally expensive.
- The model may become less interpretable with a large number of trees.
- Random Forest is a versatile and powerful algorithm that is widely used in practice, especially in situations where interpretability is not the primary concern and high predictive accuracy is desired.

```
from sklearn.metrics import r2_score
R2 = r2_score(y_test,predicted)
print("THE R2 SCORE OF RandomForestRegressor IS :",R2*100)

THE R2 SCORE OF RandomForestRegressor IS : 90.5396819430449

from sklearn.metrics import rand_score
RAND = rand_score(y_test,predicted)
print("THE ACCURACY SCORE OF RandomForestRegressor IS :",RAND*100)

THE ACCURACY SCORE OF RandomForestRegressor IS : 90.29093281148076

from sklearn.metrics import explained_variance_score
EVS = explained_variance_score(y_test,predicted)
print("THE EXPLAINED VARIANCE SCORE OF RandomForestRegressor IS :",EVS)

THE EXPLAINED VARIANCE SCORE OF RandomForestRegressor IS : 0.9054034946192723

from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,predicted)
print("THE MEAN SQUARED ERROR SCORE OF RandomForestRegressor IS :",MSE)

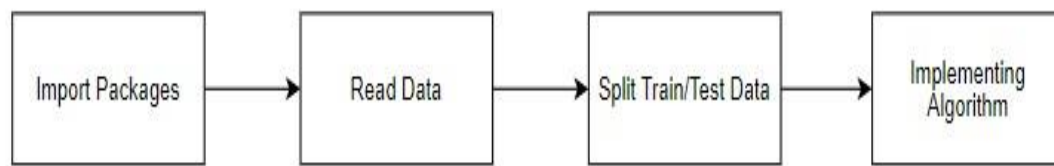
THE MEAN SQUARED ERROR SCORE OF RandomForestRegressor IS : 9964273.478743378

from sklearn.metrics import median_absolute_error
MAE = median_absolute_error(y_test,predicted)
print("THE MEAN ABSOLUTE ERROR SCORE OF RandomForestRegressor IS :",MAE)

THE MEAN ABSOLUTE ERROR SCORE OF RandomForestRegressor IS : 236.72500000000002
```

Fig.12..2 Sckit-learn Implementation

MODULE DIAGRAM:



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

12.3 LINEAR REGRESSOR:

Linear regression is a fundamental supervised learning algorithm used for predicting a continuous target variable based on one or more independent features.

1. Simple Linear Regression: In simple linear regression, there is one independent variable (feature) and one dependent variable (target). The relationship is modeled as a straight line.

2. Multiple Linear Regression: In multiple linear regression, there are multiple independent variables. The relationship is modeled as: $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$ where $\beta_0, \beta_1, \beta_2, \dots, \beta_n$ are the coefficients, and x_1, x_2, \dots, x_n are the independent variables.

3. Objective Function (Cost Function): The goal of linear regression is to find the values of the coefficients that minimize the difference between the predicted values and the actual target values. This is typically done by minimizing the mean squared error (MSE) or mean absolute error (MAE) between the predicted and actual values.

4. Training Process: The training process involves adjusting the coefficients to minimize the cost function. This is often done using optimization algorithms like gradient descent. Gradient descent iteratively updates the coefficients in the direction that reduces the cost.

5. Gradient Descent: Gradient descent is an iterative optimization algorithm used to find the minimum of a function. In the context of linear regression, it adjusts the coefficients by moving in the direction of the steepest decrease in the cost function.

6. Feature Scaling: Feature scaling can be important in linear regression. Scaling features ensures that each feature contributes proportionally to the optimization process. Common methods include standardization (subtracting mean and dividing by standard deviation) or normalization (scaling to a specific range).

7. Assumptions of Linear Regression: Linear regression assumes that there is a linear relationship between the features and the target variable. It also assumes that the residuals (the differences between predicted and actual values) are normally distributed and have constant variance.

8. Regularization (Optional): Regularization techniques such as Ridge (L2 regularization) or Lasso (L1 regularization) can be applied to prevent overfitting and improve the generalization of the model.

9. Evaluation Metrics: Common metrics for evaluating the performance of a linear regression model include Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 (coefficient of determination).

10. Applications: Linear regression is widely used in various fields, including finance, economics, biology, and social sciences. It is often used for predicting house prices, stock prices, sales, and many other continuous variables.

```
from sklearn.metrics import r2_score
R2 = r2_score(y_test,predicted)
print("THE R2 SCORE OF LINEAR REGRESSER IS :",R2*100)
```

THE R2 SCORE OF LINEAR REGRESSER IS : 55.79949826470872

```
from sklearn.metrics import rand_score
RAND = rand_score(y_test,predicted)
print("THE ACCURACY SCORE OF LINEAR REGRESSER IS :",RAND*100)
```

THE ACCURACY SCORE OF LINEAR REGRESSER IS : 97.75685414128289

```
from sklearn.metrics import explained_variance_score
EVS = explained_variance_score(y_test,predicted)
print("THE EXPLAINED VARIANCE SCORE OF LINEAR REGRESSER IS :",EVS)
```

THE EXPLAINED VARIANCE SCORE OF LINEAR REGRESSER IS : 0.5581262629005517

```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,predicted)
print("THE MEAN SQUARED ERROR SCORE OF LINEAR REGRESSER IS :",MSE)
```

THE MEAN SQUARED ERROR SCORE OF LINEAR REGRESSER IS : 43911.78347656334

```
from sklearn.metrics import median_absolute_error
MAE = median_absolute_error(y_test,predicted)
print("THE MEAN ABSOLUTE ERROR SCORE OF LINEAR REGRESSER IS :",MAE)
```

THE MEAN ABSOLUTE ERROR SCORE OF LINEAR REGRESSER IS : 124.14957117180617

Fig 12.6 Linear Regressor

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

CHAPTER-13

13.1 DEPLOYMENT:

Django (Web Framework) :

Django is an extremely popular and fully featured server-side web framework, written in Python. This module shows you why Django is one of the most popular web server frameworks, how to set up a development environment, and how to start using it to create your own web applications. In this first Django article we answer the question "What is Django?" and give you an overview of what makes this web framework special. We'll outline the main features, including some advanced functionality that we won't have time to cover in detail in this module. Now that you know what Django is for, we'll show you how to set up and test a Django development environment on Windows, Linux (Ubuntu), and macOS . Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-

supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

13.2 HTML Introduction

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



Fig.13.2 HTML Tag

`<!DOCTYPE html>` — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

`<html>` — This tag signals that from here on we are going to write in HTML code.

`<head>` — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the **content of the page** goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

Add HTML Headings To Web Page

In HTML, headings are written in the following elements:

- `<h1>`
- `<h2>`
- `<h3>`
- `<h4>`
- `<h5>`
- `<h6>`

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

A link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the `<a>` tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href=http://www.google.com>Google</a>
```

Image Tag:

In today's modern digital world, images are everything. The `` tag has everything you need to display images on your site. Much like the `<a>` anchor element, `` also contains an attribute

13.3 CSS:

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS SYNTAX:

```
Selector {  
  
Property 1 : value;  
  
Property 2 : value;  
  
Property 3 : value;  
  
}
```

For example:

```
h1 {  
  Color: red;  
  Text-align: center;  
}  
  
#unique {  
  Color: green;  
}
```

- Selector: selects the element you want to target
- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don't render on the browser
- Helps to debug our code
- Two ways to comment:
 - Single line

CSS How-To

There are 3 ways to write CSS in our HTML file.

- Inline CSS
- Internal CSS
- External CSS
- Priority order
 - Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost
- Example:

```
<h3 style = "color:red"> Have a great day </h3>
```

```
<p style = "color:green"> I did this, I did that </p>
```

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document

Example:

```
<style>
```

```
H1{
```

```
Color:red;
```

```
}
```

```
</style>
```

```
<h3> Have a great day </h3>
```

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
 - Element Selector
 - Id Selector
 - Class Selector
- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

CSS Background

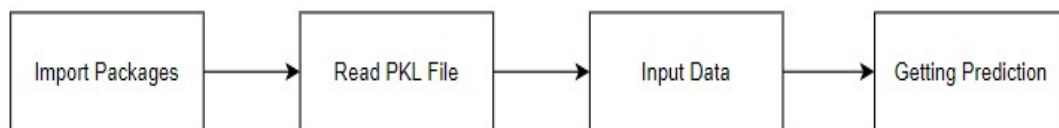
- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
 - Color – used to set the color of the background
 - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
 - Image – used to set an image as the background
 - Position – used to determine the position of the image

Attachment – It basically helps in controlling the mechanism of scrolling.

Deploying the model predicting output:

In this module the trained machine learning model is converted into pickle data format file (.pkl file) which is then deployed for providing better user interface and predicting the output of Predicting the solarpower generation .

MODULE DIAGRAM:



CHAPTER-14

14.1 Module – I

Data Preprocessing:

```
# Data Pre-Processing
```

```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')
In [ ]:
df = pd.read_csv('GENERATION.csv')
df.head()
In [ ]:
df.tail()
In [ ]:
df.shape
In [ ]:
df.size
In [ ]:
df.columns
In [ ]:
df.isnull()
```

In []:

```

In [:
df = df.dropna()
In [:
df['Power Generated'].unique()
In [:
df.describe()
In [:
df.corr()
In [:
df.info()
In [:
pd.crosstab(df["Relative Humidity"], df["Power Generated"])
In [:
df.groupby(["Visibility", "Average Wind Speed (Period)"]).groups
In [:
df["Power Generated"].value_counts()
In [:
pd.Categorical(df["Average Temperature (Day)"]).describe()
In [:
df.duplicated()
In [:
sum(df.duplicated())

```

14.2 Module - II

Data visualization:

```
# Data Visualization
```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
In [:
df = pd.read_csv('GENERATION.csv')
df.head()
In [:
df.columns
In [:
plt.figure(figsize=(12,7))
sns.countplot(x='Relative Humidity',data=df)
In [:
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
plt.hist(df['Average Wind Direction (Day)'],color='red')

plt.subplot(1,2,2)
plt.hist(df['Distance to Solar Noon'],color='green')
In [:
df.hist(figsize=(15,55),layout=(15,4), color='blue')
plt.show()

```

In [:

In [:


```

In []:
df['First Hour of Period'].hist(figsize=(10,5),color='yellow')
In []:
sns.ecdfplot(df['Visibility'], color='orange') # scatter, plot, triplot,
stackplot
In []:
sns.kdeplot(df['Average Barometric Pressure (Period)'], color='purple')
In []:
df['Month'].plot(kind='density')
In []:
sns.displot(df['Day'], color='purple')
# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot,
histplot, kdeplot, pointplot, violinplot, stripplot
In []:
sns.ecdfplot(df['Power Generated'], color='coral') # residplot, scatterplot
In []:
fig, ax = plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot = True, fmt='0.2%',cmap = 'autumn',ax=ax)
In []:
def plot(df, variable):
    dataframe_pie = df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(9,9), autopct='%1.2f%%', fontsize
= 10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)

plot(df, 'Power Generated')
In []:

```

14.3 Module – III

IMPLEMENTING RFR Algorithm

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
In []:
df = pd.read_csv('GENERATION.csv')

del df['Day of Year']
del df['Year']
del df['Month']
del df['Day']
del df['Is Daylight']
del df['First Hour of Period']

df.head()
In []:
df=df.dropna()
In []:
df.columns
In []:

```

```

df.rename(columns={'Distance to Solar Noon':
'Distance_to_Solar_Noon', 'Average Temperature
(Day)': 'Average_Temperature_Day', 'Average Wind Direction
(Day)': 'Average_Wind_Direction_Day',
                'Average Wind Speed (Day)': 'Average_Wind_Speed_Day', 'Sky
Cover': 'Sky_Cover', 'Relative Humidity': 'Relative_Humidity',
                'Average Wind Speed
(Period)': 'Average_Wind_Speed_Period', 'Average Barometric Pressure
(Period)': 'Average_Barometric_Pressure_Period', 'Power
Generated': 'Power_Generated'}, inplace=True)
In [ ]:
df.columns
In [ ]:
x = df.drop(labels='Power Generated', axis=1)
y = df.loc[:, 'Power Generated']
In [ ]:
x
In [ ]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30,
random_state=42)
print("NUMBER OF TRAIN DATASET      : ", len(x_train))
print("NUMBER OF TEST DATASET       : ", len(x_test))
print("TOTAL NUMBER OF DATASET      : ", len(x_train)+len(x_test))
In [ ]:
print("NUMBER OF TRAIN DATASET      : ", len(y_train))
print("NUMBER OF TEST DATASET       : ", len(y_test))
print("TOTAL NUMBER OF DATASET      : ", len(y_train)+len(y_test))
In [ ]:
from sklearn.ensemble import RandomForestRegressor
In [ ]:
RFR = RandomForestRegressor()
RFR.fit(x_train,y_train)
In [ ]:
predicted = RFR.predict(x_test)
In [ ]:
from sklearn.metrics import r2_score
R2 = r2_score(y_test,predicted)
print("THE R2 SCORE OF RandomForestRegressor IS :",R2*100)
In [ ]:
from sklearn.metrics import rand_score
RAND = rand_score(y_test,predicted)
print("THE ACCURACY SCORE OF RandomForestRegressor IS :",RAND*100)
In [ ]:
from sklearn.metrics import explained_variance_score
EVS = explained_variance_score(y_test,predicted)
print("THE EXPLAINED VARIENCE SCORE OF RandomForestRegressor IS :",EVS)
In [ ]:
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,predicted)
print("THE MEAN SQUARED ERROR SCORE OF RandomForestRegressor IS :",MSE)
In [ ]:
from sklearn.metrics import median_absolute_error
MAE = median_absolute_error(y_test,predicted)
print("THE MEAN ABSOLUTE ERROR SCORE OF RandomForestRegressor IS :",MAE)
In [ ]:
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test

```

```

df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed',
color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed',
color='green')
plt.show()
In [ ]:
import joblib
joblib.dump(RFR, 'GENERATION2.pkl')

```

14.4 Module – IV

IMPLEMENTING LINEAR REGRESSOR

```

# Linear Regression

```

In []:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

```

In []:

```

import warnings
warnings.filterwarnings('ignore')
In [ ]:
df = pd.read_csv('RADIATION.csv')
df.head()
In [ ]:
del df['UNIXTime']
del df['Data']
del df['Time']
del df['TimeSunRise']
del df['TimeSunSet']
In [ ]:
df.head()
In [ ]:
df.rename(columns={'WindDirection(Degrees)': 'WindDirection_Degrees'})
In [ ]:
df=df.dropna()
In [ ]:
df.columns
In [ ]:
x = df.drop(labels='Radiation', axis=1)
y = df.loc[:, 'Radiation']
In [ ]:
x
In [ ]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42)
print("NUMBER OF TRAIN DATASET      : ", len(x_train))
print("NUMBER OF TEST DATASET       : ", len(x_test))
print("TOTAL NUMBER OF DATASET      : ", len(x_train)+len(x_test))
In [ ]:
print("NUMBER OF TRAIN DATASET      : ", len(y_train))

```

```

print("NUMBER OF TEST DATASET      : ", len(y_test))
print("TOTAL NUMBER OF DATASET    : ", len(y_train)+len(y_test))
In [ ]:
from sklearn.linear_model import LinearRegression
In [ ]:
LR = LinearRegression()
LR.fit(x_train,y_train)
In [ ]:
predicted = LR.predict(x_test)
In [ ]:
from sklearn.metrics import r2_score
R2 = r2_score(y_test,predicted)
print("THE R2 SCORE OF LINEAR REGRESSER IS :",R2*100)
In [ ]:
from sklearn.metrics import rand_score
RAND = rand_score(y_test,predicted)
print("THE ACCURACY SCORE OF LINEAR REGRESSER IS :",RAND*100)
In [ ]:
from sklearn.metrics import explained_variance_score
EVS = explained_variance_score(y_test,predicted)
print("THE EXPLAINED VARIANCE SCORE OF LINEAR REGRESSER IS :",EVS)
In [ ]:
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,predicted)
print("THE MEAN SQUARED ERROR SCORE OF LINEAR REGRESSER IS :",MSE)
In [ ]:
from sklearn.metrics import median_absolute_error
MAE = median_absolute_error(y_test,predicted)
print("THE MEAN ABSOLUTE ERROR SCORE OF LINEAR REGRESSER IS :",MAE)
In [ ]:
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed',
color='red')
plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed',
color='green')
plt.show()
In [ ]:import joblib
joblib.dump(LR, 'RADIATION2.pkl')

```

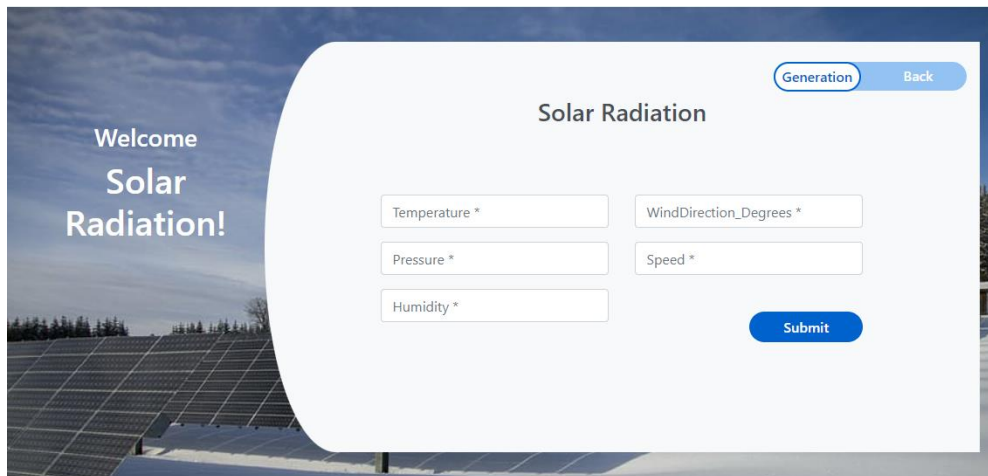
14.5 OUTPUT SCREENSHOT:



Fig 14.5 Output

14.6 SOLAR RADIATION:

Solar radiation, often called the solar resource or just sunlight, is a general term for the electromagnetic radiation emitted by the sun. Solar radiation can be captured and turned into useful forms of energy, such as heat and electricity, using a variety of technologies.



The screenshot shows a web application interface for 'Solar Radiation'. On the left, there is a blue sidebar with the text 'Welcome Solar Radiation!'. The main content area has a light blue header with 'Solar Radiation' and two buttons: 'Generation' and 'Back'. Below the header, there are five input fields arranged in two columns: 'Temperature *', 'WindDirection_Degrees *', 'Pressure *', 'Speed *', and 'Humidity *'. A blue 'Submit' button is located at the bottom right of the input fields. The background of the interface features a photograph of solar panels.

Fig.14.6 Solar Radiation

14.7 SOLAR GENERATION:

Solar power works by converting energy from the sun into power. There are two forms of energy generated from the sun for our use – electricity and heat. Both are generated through the use of solar panels, which range in size from residential rooftops to 'solar farms' stretching over acres of rural land.

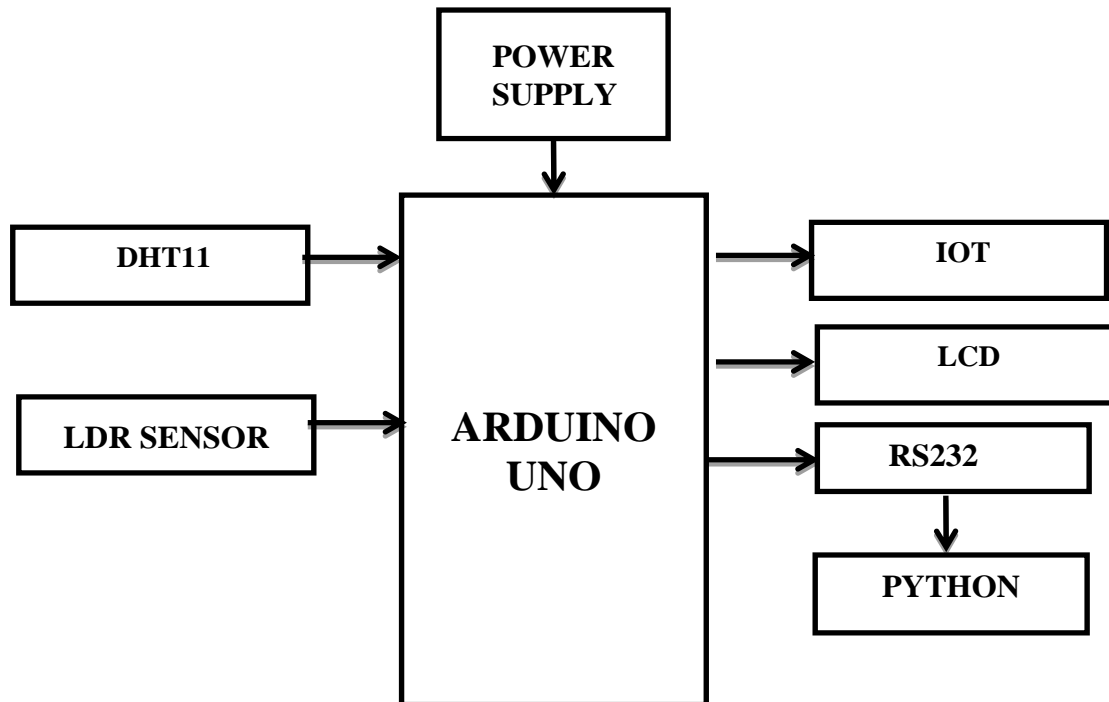


The screenshot shows a web application interface for 'Solar Generation'. On the left, there is a blue sidebar with the text 'Welcome Solar Generation' and a link 'Back to HomePage'. The main content area has a light blue header with 'Solar Generation' and two buttons: 'Generation' and 'Radiation'. Below the header, there are eight input fields arranged in two columns: 'Distance_to_Solar_Noon *', 'Sky_Cover *', 'Average_Temperature_Day *', 'Visibility *', 'Average_Wind_Direction_Day *', 'Relative_Humidity *', 'Average_Wind_Speed_Day *', 'Average_Wind_Speed_Period *', and 'Average_Barometric_Pressure_Period *'. A blue 'Submit' button is located at the bottom right of the input fields.

Fig.14.7 Solar Generation Output

CHAPTER-15

BLOCK DIAGRAM:



Solar radiation is the Earth's primary source of energy and has an important role in the surface radiation balance, hydrological cycles, and vegetation photosynthesis, and weather and climate extremes. The accurate prediction of solar radiation is therefore very important in both the solar industry and climate research. Scikit-learn is a popular machine learning library in Python, but it primarily focuses on traditional machine learning algorithms and does not directly provide functionality for solar power generation or solar radiation prediction. However, you can use scikit-learn in conjunction with other libraries and techniques from reliable source side range of models, including regression models, ensemble methods (such as random forests or gradient boosting), or even neural networks. It's worth mentioning that while scikit-learn can be used for certain aspects of the pipeline, such as data pre-processing, model training, and evaluation, other libraries or techniques may be more suitable for solar power generation and solar radiation prediction.

15.1 POWER SUPPLY:

AC socket. Some Arduino boards like UNO, MEGA and DUE, come with an AC socket that can be used to power the boards and to supply additional voltage if needed. A power supply adapter that provides from 7 to 12V (Volts) of DC (Direct Current) is required

15.2 ARDUINO UNO:

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.

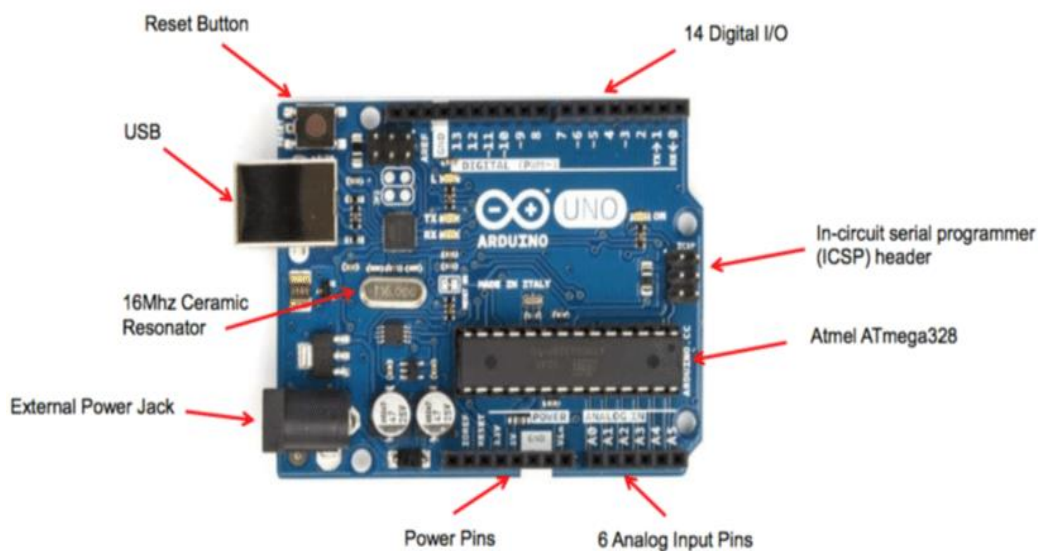


Fig.15.2 Arduino uno

15.3 TEMPERATURE SENSOR- (DHT11):

DHT11 can measure temperature from 0°C to 50°C with a $\pm 2.0^\circ\text{C}$ accuracy, and humidity from 20 to 80% with a 5% accuracy. Note that the DHT11 has a sampling rate of 1Hz, which means it can provide new data once every second.

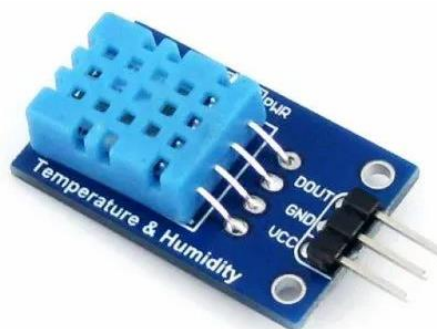


Fig.15.3 DHT11 SENSOR

15.4 LDR SENSOR:

A Light Dependent Resistor (LDR) is a type of passive electronic sensor used to detect light. It's made up of two conductors separated by an insulator which becomes more conducting when exposed to high levels of light intensity, forming a variable resistor in the circuit.

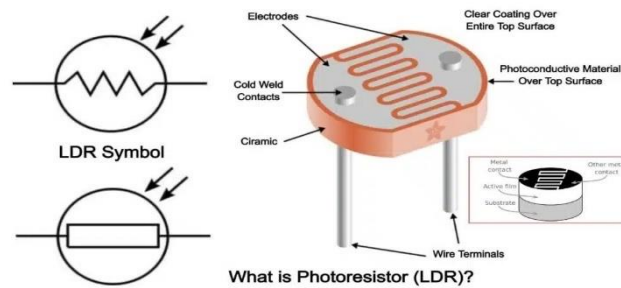


Fig.15.4 LDR Sensor

15.5 IOT

IoT Sensors are electronic chipsets or modules that sense the ambient or system conditions and transmit that data to the Internet through a gateway. These different sensors can function through physical contact, radiation, or magnetic fields.

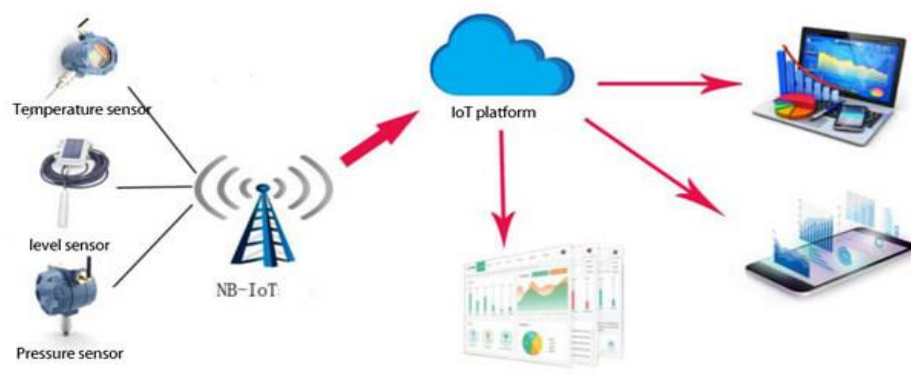


Fig.15.5 IOT

15.6 LCD DISPLAY:

LCD display is a liquid crystal display that can show 16 characters in each of its two rows, providing a total of 32 characters of information. It's commonly used to display alphanumeric information in various electronic devices.



Fig.15.6 LCD Display

15.7 RS232:

An RS-232 serial port is most often used by repair technicians to perform diagnostics and service updates. It may also be used to control a device when connected to a computer running a home automation system or Custom Integrated Audio/Video (A/V) system, such as the CAV-M1000ES Multi-room A/V Distribution System.



ADVANTAGES:

- We compared more than a two algorithms to getting better accuracy level.
- We figured out solar power generation & radiation.
- We deploy the model into production level application.
- We improved the accuracy level and performance level.
- We implement machine learning techniques for effective manner.

DISADVANTAG:

- They collect only solar array panel data for analysis purpose.
- They did not do any machine learning techniques.
- They analyze only power generation.
- Their working terminologies are quite complex to understand.
- They did not deploy the model into environment.

CHAPTER -16

16.1 Conclusion:

The energy balance of solar energy is exceptionally certain. The energy consumed in the entire chain of solar plants is recuperated in a few normal functional months. The correlation of solar energy with ordinary innovations features the ecological benefits of solar energy. The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The Best accuracy on public test set is higher accuracy score is will be find out. This application can help out to find the Solar Power Prediction.

Future Work:

- Solar Power Generation Prediction to connect the AI Model.
- To Optimize the Work to implement in AI Environment.