SOLAR POWER FORECASTING USING DEEP LEARNING TECHNIQUES

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

MOOLA MANOJ KUMAR, 221910305028
NANDGIRI SAI RAGHAVA KAPIL SHARMA, 221910305031
PATCHALA DEEPA SREE, 221910305036
PINJARI ADNAN HUSSAIN, 221910305038

Under the esteemed guidance of

Dr. M NISHANTH KARTHEEK
Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GITAM
(Deemed to be University)
HYDERABAD
November 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM
(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled **"SOLAR POWER FORECASTING USING DEEP LEARNING TECHNIQUES"** is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No.	**Name**	**Signature**
221910305028	MOOLA MANOJ KUMAR	
221910305031	NANDGIRI SAI RAGHAVA KAPIL SHARMA	
221910305036	PATCHALA DEEPA SREE	
221910305038	PINJARI ADNAN HUSSAIN	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM INSTITUTE OF TECHNOLOGY
GITAM
(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled **"SOLAR POWER FORECASTING USING DEEP LEARNING TECHNIQUES"** is a bonafide record of work carried out by **MOOLA MANOJ KUMAR (221910305028), NANDGIRI SAI RAGHAVA KAPIL SHARMA (221910305031), PATCHALA DEEPA SREE (221910305036), PINJARI ADNAN HUSSAIN (221910305038)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Project Coordinator**	**Head of the Department**
Dr. M NISHANTH KARTHEEK	Dr. S APARNA	Dr. S PHANI KUMAR
Assistant Professor	Assistant Professor	Professor

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. We would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honorable Pro-Vice Chancellor, **Prof. D Sambasiva Rao** for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to **Dr. M Nishanth Kartheek**, Assistant Professor, Department of Computer Science & Engineering, for the esteemed guidance, moral support and invaluable advice provided by her for the success of the project.

We are also thankful to all the staff members of the Computer Science and Engineering department who have cooperated in making our project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our project work.

Sincerely,

MOOLA MANOJ KUMAR, 221910305028
NANDGIRI SAI RAGHAVA KAPIL SHARMA, 221910305031
PATCHALA DEEPA SREE, 221910305036
PINJARI ADNAN HUSSAIN, 221910305038

TABLE OF CONTENTS

LIST OF FIGURE

ABSTRACT

The recent rapid and sudden growth of solar photovoltaic (PV) technology presents a future challenge for the electricity sector agents responsible for the coordination and distribution of electricity given the direct dependence of this type of technology on climatic and meteorological conditions. Therefore, the development of models that allow reliable future prediction, in the short term, of solar PV generation will be of paramount importance, in order to maintain a balanced and comprehensive operation. This work discusses a method for predicting the generated power, in the short term, of photovoltaic power plants, by means of deep learning techniques. To fulfill the above, a deep learning technique based on the KNN, ANN and LASSO regression algorithms are evaluated with respect to its ability to forecast solar power data. The prediction result shows that the DL techniques gives the best results for each category of days. Thus, it provides reliable information that enables more efficient operation of photovoltaic power plants in the future. The binomial formed by the concepts of deep learning and energy efficiency seems to have a promising future, especially regarding promoting energy sustainability, decarburization, and the digitization of the electricity sector.

1.INTRODUCTION

Renewable energy, especially solar PV, will gain prominence as a major source of energy in the future. But as their share of the energy mix grows, ensuring the safety, reliability, and profitability of power generation assets will be a top priority. Therefore, the successful integration of solar energy into the electrical grid requires an accurate prediction of the power generated by photovoltaic panels. Speaking of solar energy in particular, its unexpected behavior brings with it a series of problems when generating energy, such as voltage variations, power factor details, and stability. For this reason, these new tools are constantly being created that contribute to the prediction of future events, with the aim of reducing errors in predictions [1].

Auto-Regressive Integrated Moving Averages (ARIMA) models have proven to be extremely useful for the short-term prediction of highfrequency time series. In contrast to ARIMA models and statistical methods, artificial neural networks are more powerful, especially in representing complex relationships that exhibit nonlinear behaviors. In recent decades, the application of artificial neural networks (ANNs) in time series prediction has grown due to the ideal characteristics offered by ANNs for working with nonlinear models. Likewise, the development of applications that facilitate work when carrying out simulations with ANN continues to increase.

In [2], artificial neural networks are highlighted as one of the prediction methods for time series thanks to their great adaptability and capacity to solve nonlinear and complex problems. In recent years, as a result of the research on artificial

intelligence, deep learning based on ANN has come to light to become very popular due to its capability to accelerate the solution of some difficult computer problems. While multi-layer perceptron (MLP)-type ANNs can be used to model complex relationships, they are incapable of assimilating the long- and short-term dependencies present in historical data. These dependencies refer to the ability of an ANN to identify and remember behavior patterns from the distant past and the near past, respectively.

1.1 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- **Jupiter**
- **Google colab**

1.2 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

2. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high

demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.LITERATURE SURVEY

If intermittent renewable energy technologies such as those based on solar, wind, wave and tidal resources are eventually to supply significant shares of total energy supplies, it is crucial that the energy storage problem is solved. There are several (long-recognised) possibilities ahead including compressed air, pumped storage, further developments in batteries, regenerable fuel cells, 'super-capacitors' and so forth. But one that is being revisited extensively by industry and research establishments is the production and storage of hydrogen from electricity at off-peak times, and in times when there would be a surplus of renewable energy, for reuse in the electricity, gas and transport markets; short-term and even seasonal and longer-term storage is technically feasible with this option. This paper looks at the costs of the option both in the near-term and the long-term relative to the current costs of electricity and natural gas supplies. While the costs of hydrogen would necessarily be greater than those of natural gas (though not disruptively so), when used in conjunction with emerging technologies for decentralised generation and combined heat and power there is scope for appreciable economies in electricity supply. A lot will depend on innovation at the systems level, and on how we operate our electricity and gas grids and regulate our electricity and gas industries. We have also suggested that we now need to experiment more, at the commercial level, and in the laboratories, with the hydrogen option. [9]

Most of state of the art methods applied on time series consist of deep learning methods that are too complex to be interpreted. This lack of interpretability is a major drawback, as several applications in the real world are critical tasks, such as the medical field or the autonomous driving field. The explainability of models

applied on time series has not gather much attention compared to the computer vision or the natural language processing fields. In this paper, we present an overview of existing explainable AI (XAI) methods applied on time series and illustrate the type of explanations they produce. We also provide a reflection on the impact of these explanation methods to provide confidence and trust in the AI systems. [6]

Artificial intelligence-powered medical technologies are rapidly evolving into applicable solutions for clinical practice. Deep learning algorithms can deal with increasing amounts of data provided by wearables, smartphones, and other mobile monitoring sensors in different areas of medicine. Currently, only very specific settings in clinical practice benefit from the application of artificial intelligence, such as the detection of atrial fibrillation, epilepsy seizures, and hypoglycemia, or the diagnosis of disease based on histopathological examination or medical imaging. The implementation of augmented medicine is long-awaited by patients because it allows for a greater autonomy and a more personalized treatment, however, it is met with resistance from physicians which were not prepared for such an evolution of clinical practice. This phenomenon also creates the need to validate these modern tools with traditional clinical trials, debate the educational upgrade of the medical curriculum in light of digital medicine as well as ethical consideration of the ongoing connected monitoring. The aim of this paper is to discuss recent scientific literature and provide a perspective on the benefits, future opportunities and risks of established artificial intelligence applications in clinical practice on physicians, healthcare institutions, medical education, and bioethics. [3]

The increasing world population and availability of energy hungry smart devices are major reasons for alarmingly high electricity consumption in the current times. So far, various simulation tools, engineering and Artificial Intelligence based methods are being used to perform optimal electricity demand forecasting. While engineering methods use dynamic equations to forecast, the AI-based methods use historical data to predict future demand. However, modeling of nonlinear electricity demand patterns is still underdeveloped for robust solutions as the existing methods are useful only for handling short-term dependencies. Moreover, the existing methods are static in nature because they are purely historical data driven. In this paper, we propose a deep learning based framework to forecast electricity demand by taking care of long-term historical dependencies. Initially, the cluster analysis is performed on the electricity consumption data of all months to generate season based segmented data. Subsequently, load trend characterization is carried out to have a deeper insight of metadata falling into each of the clusters. Further, Long Short Term Memory network multi-input multi-output models are trained to forecast electricity demand based upon the season, day and interval data. In the present work, we have also incorporated the concept of moving window based active learning to improve prediction results. To demonstrate the applicability and effectiveness of the proposed approach, it is applied to the electricity consumption data of Union Territory Chandigarh, India. Performance of the proposed approach is evaluated by comparing the prediction results with Artificial Neural Network, Recurrent Neural Network and Support Vector Regression models. [4]

In the last few years, the deep learning (DL) computing paradigm has been deemed the Gold Standard in the machine learning (ML) community. Moreover, it has gradually become the most widely used computational approach in the field of ML,

thus achieving outstanding results on several complex cognitive tasks, matching or even beating those provided by human performance. One of the benefits of DL is the ability to learn massive amounts of data. The DL field has grown fast in the last few years and it has been extensively used to successfully address a wide range of traditional applications. More importantly, DL has outperformed well-known ML techniques in many domains, e.g., cybersecurity, natural language processing, bioinformatics, robotics and control, and medical information processing, among many others. Despite it has been contributed several works reviewing the State-of-the-Art on DL, all of them only tackled one aspect of the DL, which leads to an overall lack of knowledge about it. Therefore, in this contribution, we propose using a more holistic approach in order to provide a more suitable starting point from which to develop a full understanding of DL. Specifically, this review attempts to provide a more comprehensive survey of the most important aspects of DL and including those enhancements recently added to the field. In particular, this paper outlines the importance of DL, presents the types of DL techniques and networks. It then presents convolutional neural networks (CNNs) which the most utilized DL network type and describes the development of CNNs architectures together with their main features, e.g., starting with the AlexNet network and closing with the High-Resolution network (HR.Net). Finally, we further present the challenges and suggested solutions to help researchers understand the existing research gaps. It is followed by a list of the major DL applications. Computational tools including FPGA, GPU, and CPU are summarized along with a description of their influence on DL. The paper ends with the evolution matrix, benchmark datasets, and summary and conclusion. [8]

4.SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

The prediction of solar radiation is a fundamental key to increasing the supply of electrical energy generated from this medium to the distribution networks. In the energy market, when an electric energy producer does not comply with the programmed offer, they are penalized with a proportional relationship between the energy actually produced and what is stated in the contract. The integration of these renewable energy sources intensifies the complexity of managing the grid and the ongoing balance between electricity consumption and production due to its unpredictable and intermittent nature. Several tools and methodologies have been developed for the prediction of solar energy at different horizons.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. Complexity of managing the grid
2. Unpredictable and intermittent nature

4.2 Proposed System:

This work proposes a strategy for using deep learning to estimate the short-term generated power of photovoltaic power facilities. To achieve the aforementioned, the forecasting capabilities of a deep learning technique based on the KNN, ANN, and LASSO regression algorithms are assessed. The outcome of the forecast demonstrates that for each group of days, DL approaches produce the greatest outcomes. Thus, it offers trustworthy data that enables future photovoltaic power plants to run more effectively. Deep learning and energy efficiency create a combination that appears to have a bright future, especially in terms of boosting energy sustainability, decarburization, and the digitization of the electrical sector.

4.2.1 Advantages of proposed system:

1. It offers trustworthy data
2. Enables future photovoltaic power plants to run more effectively

4.3 FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogsDescription of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

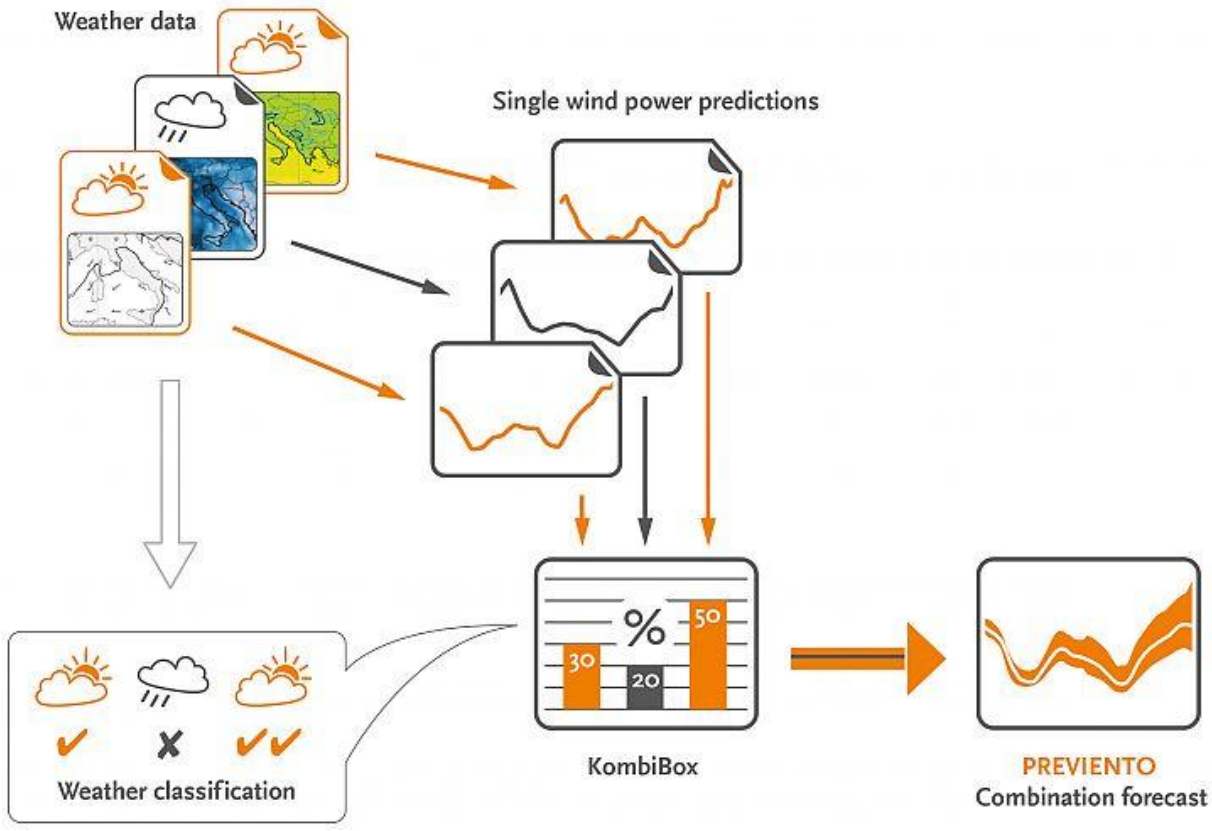


Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

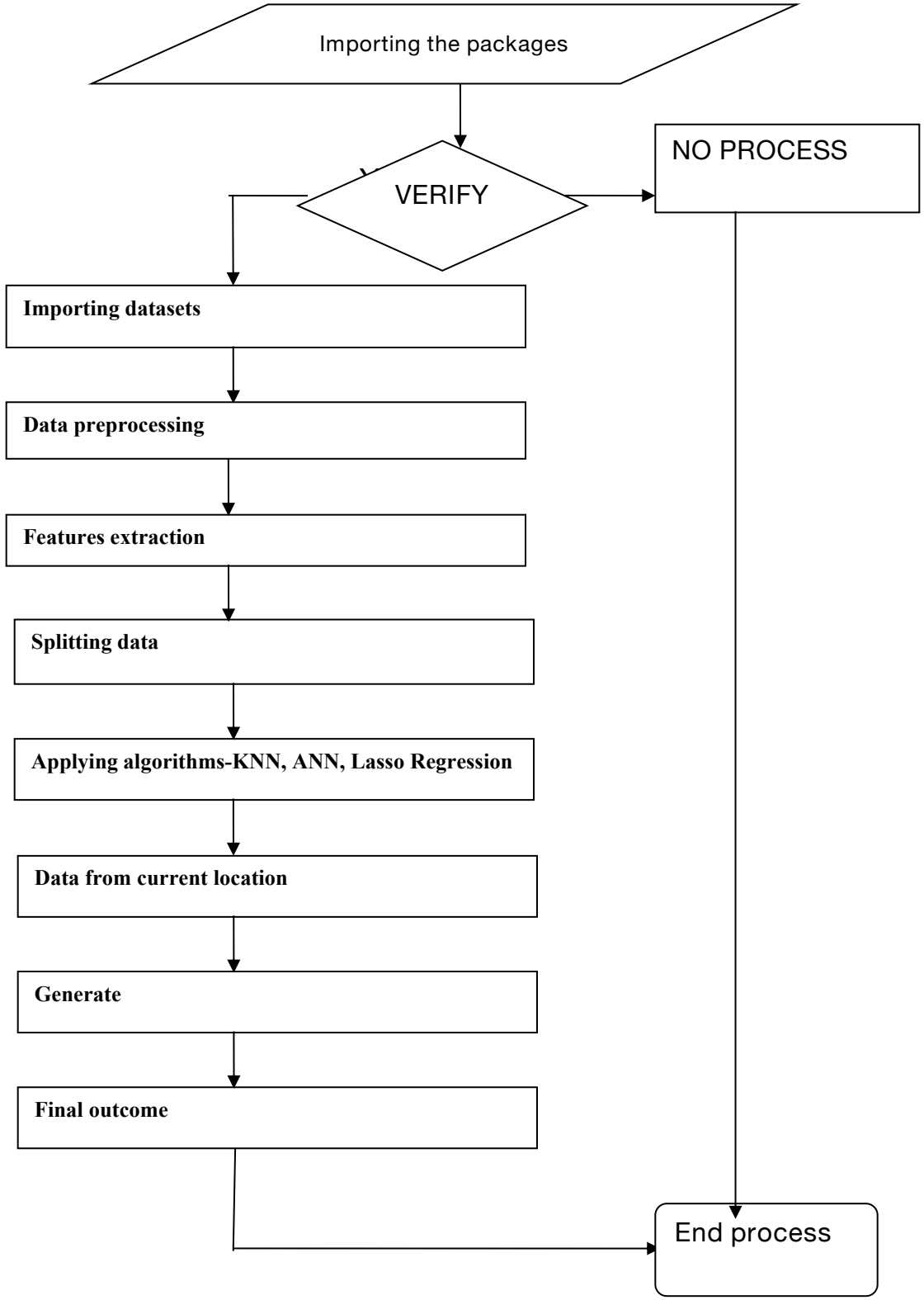


Fig.5.1.3 Dataflow diagram

5.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.

4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

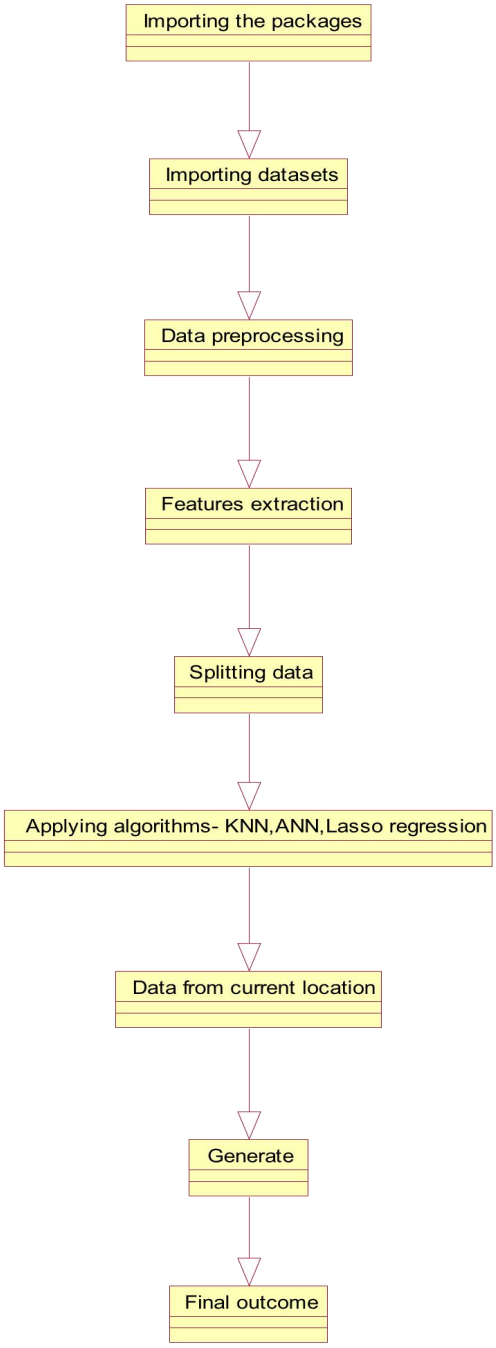


Figure 5.2.1. Class Diagram

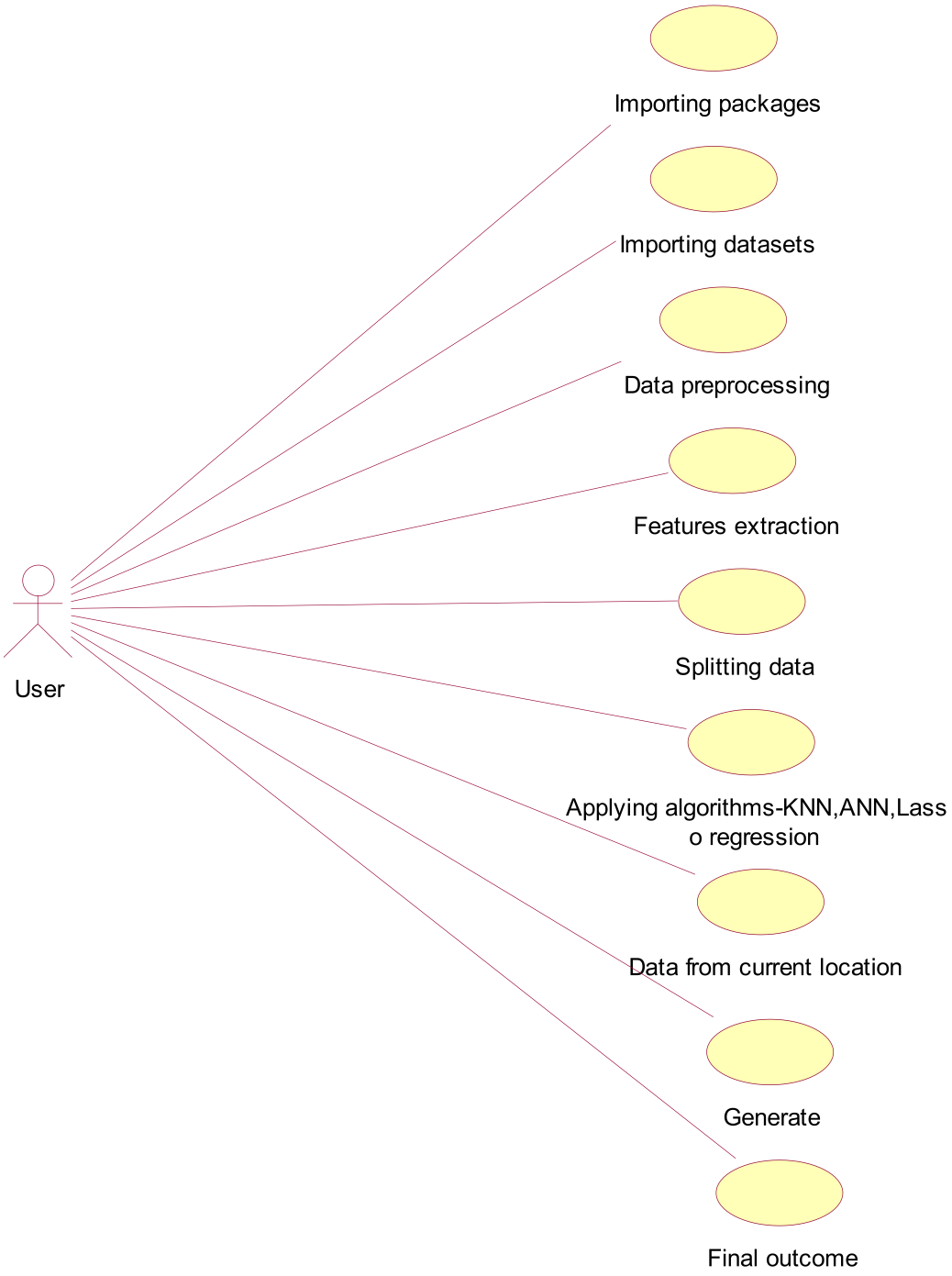


Fig.5.2.2 Class diagram

Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

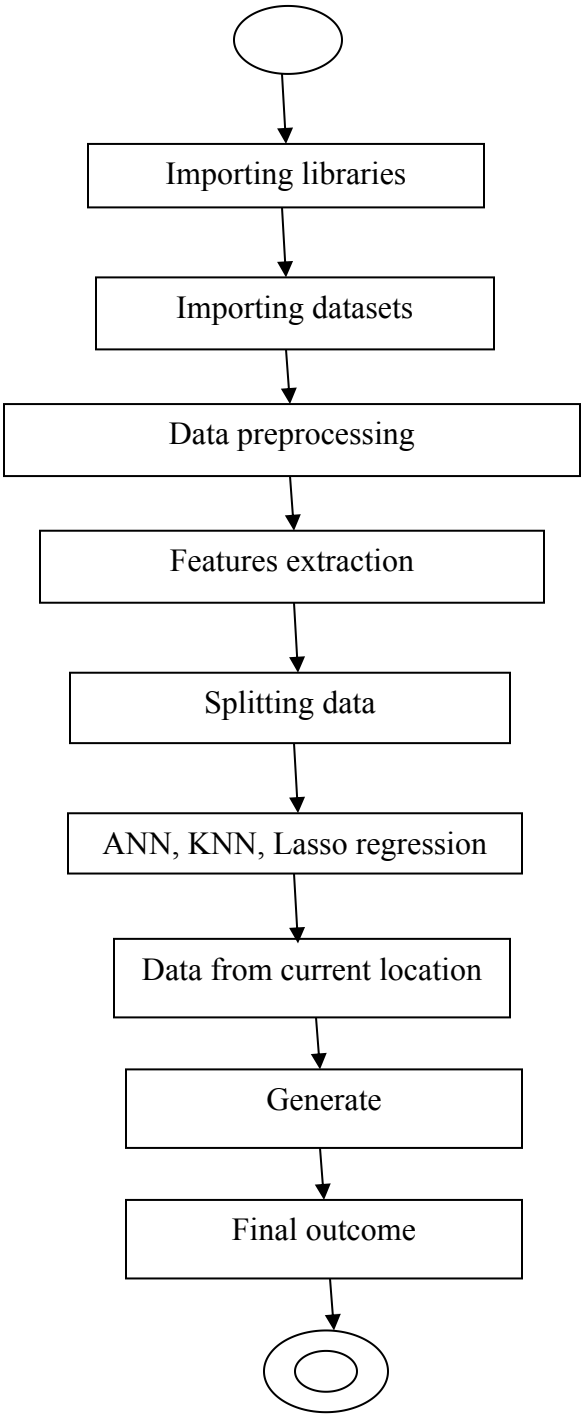


Fig.5.2.3 Activity diagram

Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

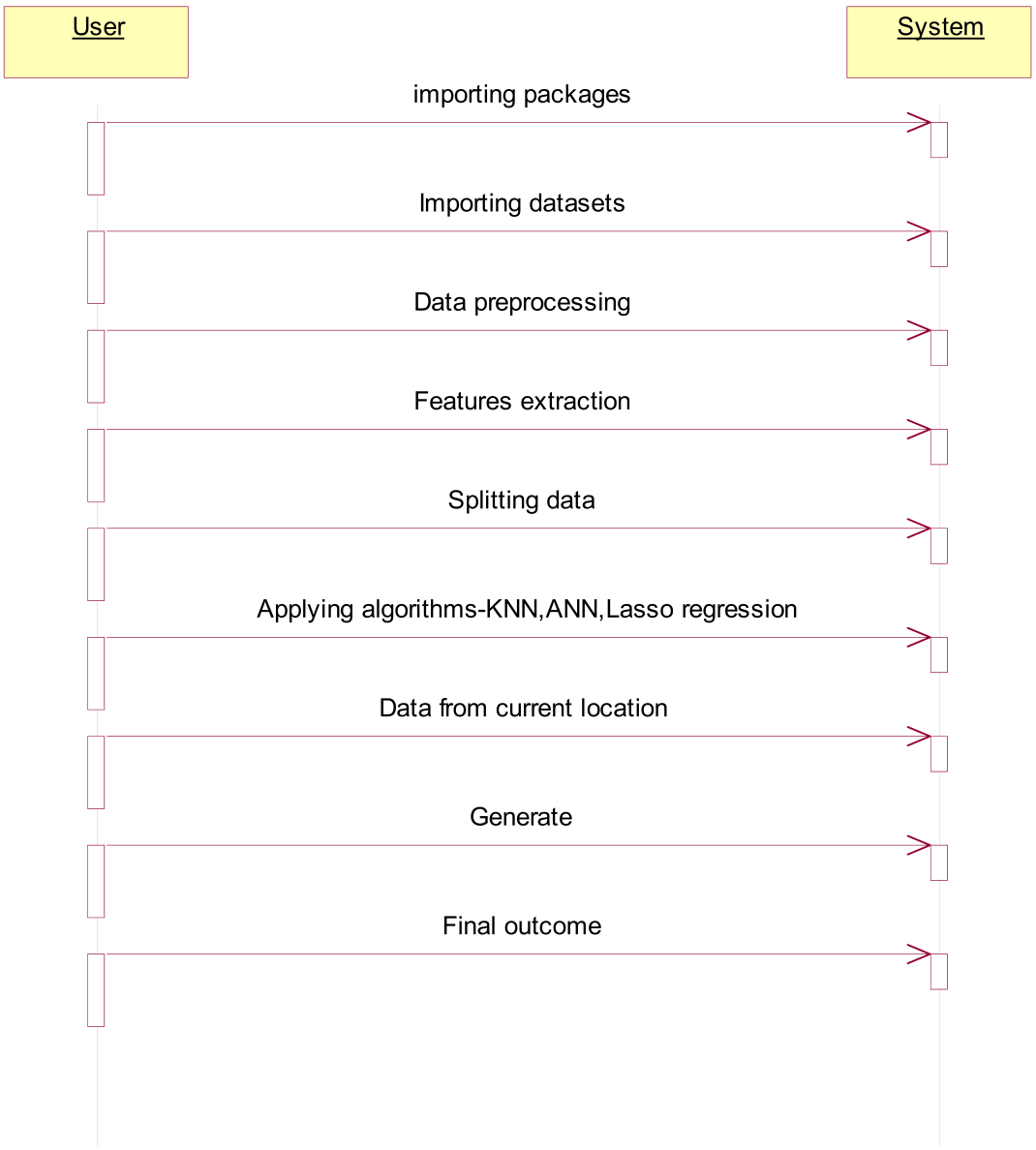


Fig.5.2.4 Sequence diagram

Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions.

The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

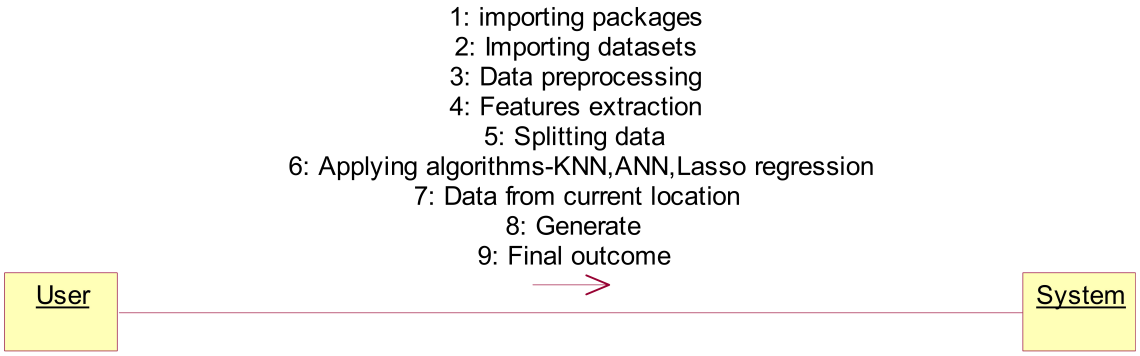


Fig.5.2.5 Collaboration diagram

Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

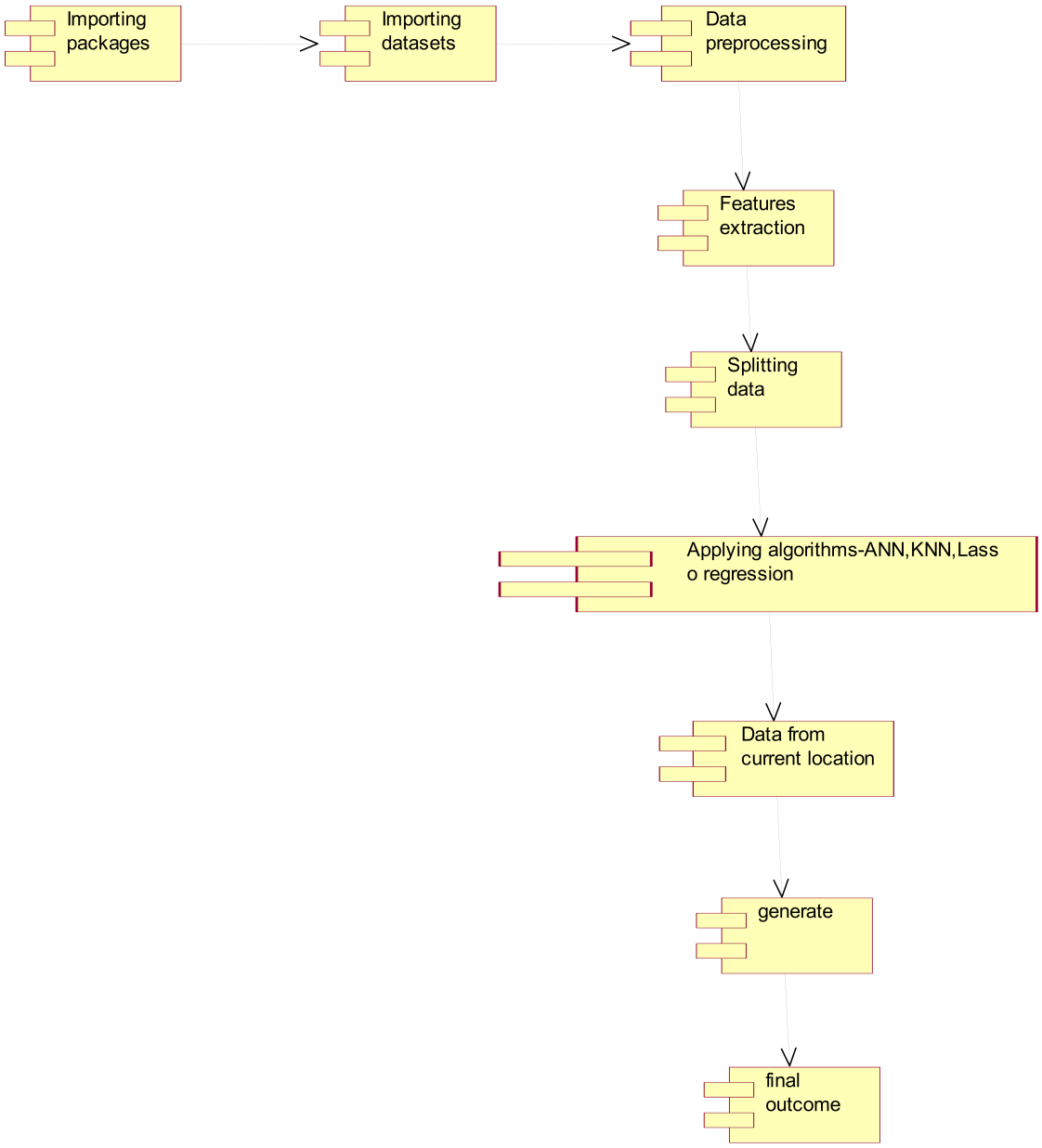


Fig.5.2.6 Component diagram

Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

Fig.5.2.7 Deployment diagram

5. IMPLEMENTATION

MODULES:

1) Importing Libraries : Using this will import necessary packages from our program
2) Importing datasets: Using this we will upload our data for processing
3) Applying algorithms: using this generate algorithms
    1) KNN
    2) ANN
    3) Lasso Regression

    Final outcome is displayed in frontend with help of Flask Framework

ALGORITHMS:

KNN: K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

ANN: Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions. The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

LASSO REGRESSION: In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both

variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. It was originally introduced in geophysics, and later by Robert Tibshirani, who coined the term. Lasso was originally formulated for linear regression models. This simple case reveals a substantial amount about the estimator. These include its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates do not need to be unique if covariates are collinear.

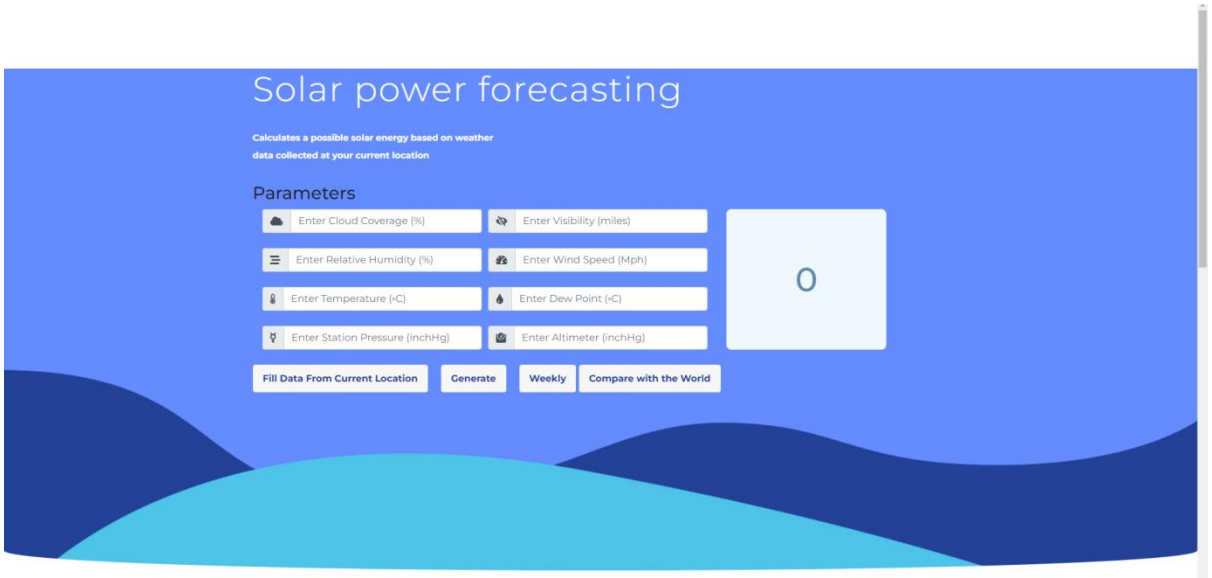
6.2 SAMPLE CODE:

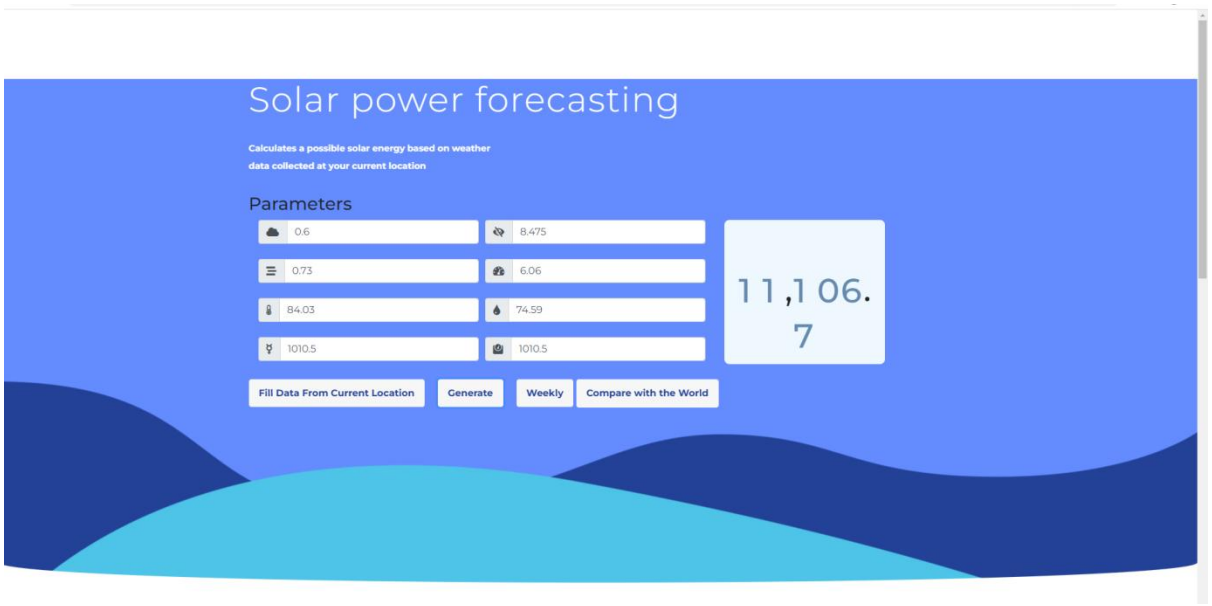
```
import pandas as pd
import numpy as np
import tensorflow as tf
from keras.layers import Dense, Activation, BatchNormalization, Dropout
from keras import regularizers
from keras.optimizers import RMSprop, Adam, SGD
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
#data_path = r'drive/My Drive/Proj/S.P.F./solarpowergeneration.csv'
dts = pd.read_csv('data/solarpowergeneration.csv')
dts.head(10)
X = dts.iloc[:, :-1].values
y = dts.iloc[:, -1].values
print(X.shape, y.shape)
y = np.reshape(y, (-1,1))
y.shape
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
print("Train Shape: {} {} \nTest Shape: {} {}".format(X_train.shape,
y_train.shape, X_test.shape, y_test.shape))
from sklearn.preprocessing import StandardScaler
# input scaling
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

# outcome scaling:
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)
y_test = sc_y.transform(y_test)
def create_spfnet(n_layers, n_activation, kernels):
  model = tf.keras.models.Sequential()
  for i, nodes in enumerate(n_layers):
    if i==0:
      model.add(Dense(nodes, kernel_initializer=kernels,
activation=n_activation, input_dim=X_train.shape[1]))
```

```
      #model.add(Dropout(0.3))
    else:
      model.add(Dense(nodes, activation=n_activation,
kernel_initializer=kernels))
      #model.add(Dropout(0.3))

  model.add(Dense(1))
  model.compile(loss='mse',
                optimizer='adam',
                metrics=[tf.keras.metrics.RootMeanSquaredError()])
  return model
spfnet = create_spfnet([32, 64], 'relu', 'normal')
spfnet.summary()
from keras.utils.vis_utils import plot_model
plot_model(spfnet, to_file='spfnet_model.png', show_shapes=True,
show_layer_names=True)
hist = spfnet.fit(X_train, y_train, batch_size=32, validation_data=(X_test,
y_test),epochs=150, verbose=2)
plt.plot(hist.history['root_mean_squared_error'])
#plt.plot(hist.history['val_root_mean_squared_error'])
plt.title('Root Mean Squares Error')
plt.xlabel('Epochs')
plt.ylabel('error')
plt.show()
```




Solar power forecasting
Calculates a possible solar energy based on weather data collected at your current location
Parameters
0.6
8.475
0.73
6.06
84.03
74.59
1010.5
1010.5
11,106.7
Fill Data From Current Location
Generate
Weekly
Compare with the World

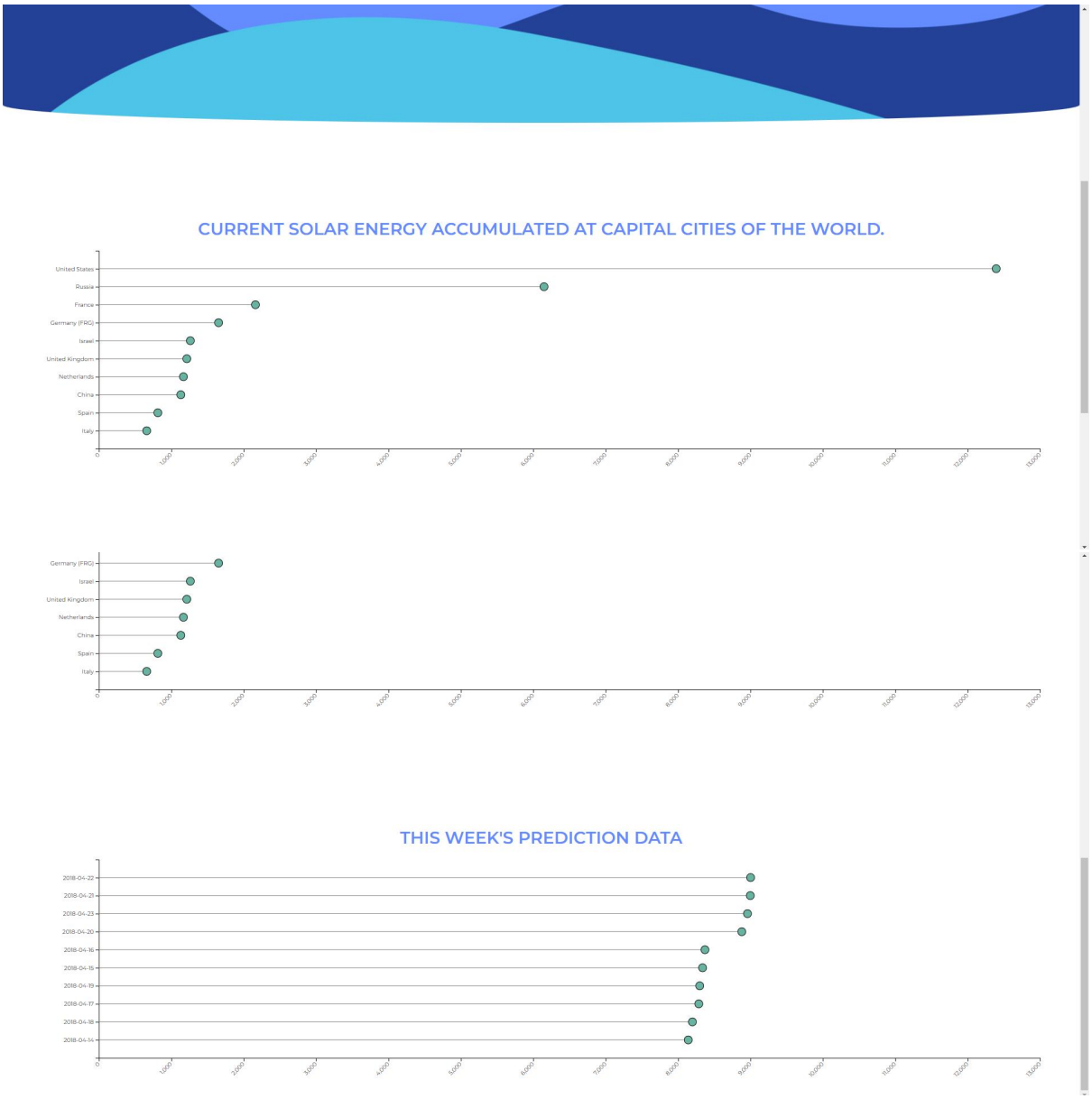


Figure 5.2.3 - THE OUTPUT

7.SOFTWARE ENVIRONMENT

What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

Strenghts :

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python

has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

  Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where

Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

8. SYSTEM TESTING

8.1 TESTING STRATEGIES

UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to determine if there are issues by the developer himself.. it is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing Techniques:

Black Box Testing - Using which the user interface, input and output are tested.

White Box Testing –Used to test each one of those functions behavior is tested.

DATA FLOW TESTING

Data flow testing is a family of testing strategies based on selecting paths through the program's control flow in order to explore sequence of events related to the status of Variables or data object. Dataflow Testing focuses on the points at which variables receive and the points at which these values are used.

INTEGRATION TESTING

Integration Testing done upon completion of unit testing, the units or modules are to be integrated which gives raise too integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

BIG BANG INTEGRATION TESTING

Big Bang Integration Testing is an integration testing Strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

USER INTERFACE TESTING

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by Graphical User interface [GUI].

8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	Data from current location	Data taken from the location	There is no process
2	Generate	Data generated	There is no process

10.CONCLUSION

The proposed model gives trustworthy data that will allow photovoltaic power plants to operate more efficiently in the future. The combination formed by the concepts of artificial intelligence and energy efficiency appears to have a bright future, particularly in terms of boosting energy sustainability, decarburization, and electrical sector digitization. The importance of data processing (time series) utilized to train models was highlighted in this study. A specific time series may perform better with one model, while the same model may have poorer performance with another time series model. This reason suggests focusing on research and development in multiple models in order to arrive at predictions with high suitability. In addition, the preprocessing approaches used to reduce noise, eliminate outliers, and reduce errors from prediction models should be taken into account. When used to predict solar energy, artificial intelligence algorithms have demonstrated their aptitude and superiority in obtaining favorable outcomes. However, obtaining such results would necessitate a significant amount of hyperparameter adjustment. Furthermore, the quality of the data, particularly the outliers, has a substantial impact on the prediction model's performance. Several time series prediction approaches can be implemented using the methodology provided in this work.

REFERENCES

[1] D. Anderson and M. Leach, ‘‘Harvesting and redistributing renewable energy: On the role of gas and electricity grids to overcome intermittency through the generation and storage of hydrogen,’’ Energy Policy, vol. 32, no. 14, pp. 1603–1614, Sep. 2004.

[2] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, ‘‘Explainable artificial intelligence (XAI) on TimeSeries data: A survey,’’ 2021, arXiv:2104.00950.

[3] G. Briganti and O. Le Moine, ‘‘Artificial intelligence in medicine: Today and tomorrow,’’ Frontiers Med., vol. 7, p. 27, Feb. 2020.

[4] J. Bedi and D. Toshniwal, ‘‘Deep learning framework to forecast electricity demand,’’ Appl. Energy, vol. 238, pp. 1312–1326, Mar. 2019.

[5] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, ‘‘Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions,’’ J. Big Data, vol. 8, no. 1, pp. 1–74, Dec. 2021.

[6] S. Hochreiter, ‘‘The vanishing gradient problem during learning recurrent neural nets and problem solutions,’’ Int. J. Uncertainty, Fuzziness Knowl.- Based Syst., vol. 6, no. 2, pp. 107–116, Apr. 1998.

[7] S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://deeplearning.cs.cmu.edu/pdfs/Hochreiter97-lstm.pdf

[8] S. A. Kalogirou, ‘‘Artificial neural networks in renewable energy systems applications: A review,’’ Renew. Sustain. Energy Rev., vol. 5, no. 4, pp. 373–401, Dec. 2001.

[9] M. Elsaraiti and A. Merabet, ‘‘A comparative analysis of the ARIMA and LSTM predictive models and their effectiveness for predicting wind speed,’’ Energies, vol. 14, no. 20, p. 6782, Oct. 2021, doi: 10.3390/en14206782.

[10] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, ‘‘Optimizing deep learning hyper-parameters through an evolutionary algorithm,’’ in Proc. Workshop Mach. Learn. High-Perform. Comput. Environ., Nov. 2015, pp. 1–5.