

INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

KARRI VISHNU SAI RAMA REDDY

(Reg.No.-40110552)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12 B Status

by UGC | Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KARRI VISHNU SAI RAMA REDDY (40110552)** who carried out the project entitled "**RAIN FALL PREDICTION**" under my supervision from FEB 2023 to APRIL 2023.

Internal Guide

Dr. Nancy Noella, M.E., Ph.D.

Head of the Department

Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, KARRI VISHNU SAI RAMA REDDY hereby to declare that the project report entitled **RAINFALL PREDICTION** done by me under the guidance of **Dr. Nancy Noella, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, Dean, School of Computing, **Dr. L.Lakshmanan, M.E., Ph.D.**, and **Dr. S. Vigneshwari, M.E., Ph.D. Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Nancy Noella, M.E., Ph.D.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



Certificate of Training

is hereby granted to

**KARRI VISHNU SAI RAMA REDDY
(40110552)**

from Sathyabama Institute of Science and
Technology for successfully completing the 45 hours
professional training program on Machine Learning
conducted by Cognibot between 10th Feb, 2023 and
16th Apr, 2023


HARIHARASUDHAN
INSTRUCTOR, COGNIBOT

16th April, 2023

DATE

ABSTRACT

In India, Agriculture is the key point in survival. For agriculture , rainfall is most important. These days rainfall prediction has become a major problem. Prediction of rainfall gives awareness to people and know in advance about rainfall to take certain precautions to protect their crop from rainfall.

Many techniques came into existence to predict rainfall. Machine Learning algorithms are mostly useful in predicting rainfall. Some of the major Machine Learning algorithms are ARIMA Model(Auto-Regressive Integrated Moving Average), Artificial Neural Network, Logistic Regression, Support Vector Machine, Self Organizing Map, and Random Forest Classifier. Two commonly used models predict seasonal rainfall such as Linear and Non- Linear models. The first models are ARIMA Model. While using Artificial Neural Network(ANN) predicting rainfall can be done using Back Propagation NN, Cascade NN or Layer Recurrent Network. Artificial NN is same as Biological Neural Networks.

Random forest classifier is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forest classifier creates decision trees on randomly selected data samples, get prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications , such as recommendation engines, images classification and feature selection. It can be used to classify loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset. So that's why we have used Random Forest Classifier model in our project.

TABLE OF CONTENT		
CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iv
1.	INTRODUCTION 1.1 INTRODUCTION TO THE RAINFALL PREDICTION 1.2 NEED FOR THE RAINFALL PREDICTION 1.3 INTRODUCTION TO JUPYTER NOTEBOOK	1 2 2-3
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION 2.1 AIM OF THE PROJECT 2.2 SCOPE OF THE PROJECT	4 4
3.	EXPPERIMENTAL OR MATERIALS AND METHODS, ALGORITHM USED 3.1 VISUALIZATION USED 3.1.1 MAP VIEW 3.1.2 ALGORITHM USED 3.1.3 IMPORTING DATASET 3.1.4 DEALING WITH INVALID DATA 3.1.5 FEATURES SCALING AND TRANING MODEL 3.2 SOFTWARE REQUIREMENTS 3.3 HARDWARE REQUIREMENTS 3.4 SYSTEM IMPLEMENTATION 3.4.1 HISTORT OF JUPITER NOTEBOOK 3.4.2 FEATURES OF JUPITER NOTEBOOK	5 5 6 7-8 9-10 10-11 11-12 12 12 12-13 13-15

	3.4.3 JUPYTER NOTEBOOK WIDGETS	15-16
	3.5 CODE EDITOR	16-18
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	
	4.1 RESULTS	19
	4.2 FEASIBILITY STUDY	19
	4.2.1 BENEFITS	19
	4.3 IMPLEMENTATION	20
5.	SUMMARY AND CONCLUSIONS	21-24
	REFERENCES	25
	APPENDIX	26-30
	SOURCE CODE	31-32

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1.1.a	Map visual in Power Bi	5
3.1.1.b	Random Forest Classifier	6
3.1.2.a	Importing Libraries	7-8
3.1.3.a	Dealing with Invalid Data	9
3.1.4.a	More info about sklearn impute module	10
3.1.4.b	Features Scaling	10
3.1.5.b	Traning Model	11
3.5.a	Dataset	17
3.5.b	Column types	18

CHAPTER 1

1.INTRODUCTION

1.1 INTRODUCTION TO THE RAINFALL PREDICTION:

India's welfare is agriculture . The achievement of agriculture is dependent on rainfall. It also helps with water resources. Rainfall information in the past helps farmers better manages their crops, leading to economic growth in country.

Prediction of precipitation is beneficial to prevent flooding that saves people's lives and property. Fluctuation in the timing of precipitation and its amount makes forecasting of rainfall a problem for meterological scientists, Forecasting is one of the utmost challenges for researchers from a variety of fields, such as weather data mining, environmental machine learning, functional hydrology, and numerical forecasting, to create a predictive model for accurate rainfall. In these problems, a common question is how to infer the past predictions and make use of future predictions. A variety of sub-processes are typically composed of the substantial process in rainfall.it is at times not promising to predict the precipitation correctly by on its global system. Climate forecasting stands out for all countries around the globe in all the benefits of rainfall prediction meterological department.The job is very complicated because it needs specific numbers.

Accurate precipitation forecasting has been an important issue in hydrological science as early notice of stern weather can help avoid natural disaster injuries and damage if prompt and accurate forecasts are made. The theory of the modular model and the integration of different models has gained more interest in rainfall forecasting to address this challenge. A huge range of rainfall prediction methodologies are available in india.To solve all these problems we need machine learning technique and models to make accurate and timely predictions.So that's why we are using Random Forest Classifier model in this machine learning project.

1.2 NEED FOR THE RAINFALL PREDICTION:

Rainfall forecasting is very important because heavy and irregular rainfall can have many impacts like destruction of crops and farms, damage of property so a better forecasting model is essential for an early warning that can minimize risks to life and property and also managing the agricultural farms in better way. The goal of rainfall prediction is to provide information people and organizations can use to restore weather related losses.

1.3 INTRODUCTION TO JUPYTER NOTEBOOK:

Jupyter Notebook (formerly IPython Notebooks) is a web based interactive computational environment for creating notebook documents. A Jupyter Notebook document is a browser based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media.

Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the “.ipynb” extension. Jupyter notebooks are built upon a number of popular open source libraries:

- IPython
- ZeroMQ
- jQuery
- Bootstrap
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Jupyter Notebook is a new user interface for users to do projects.

Key components of Jupyter Notebook:

Its two main components are the kernels and a dashboard. A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

- `%run`: Execute Python code
- `%load`: Insert code from an external script
- `%store`: Pass variables between notebooks
- `%writefile`: Saves the contents of a cell to an external file.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 AIM OF THE PROJECT:

The main aim of the project is to get the basic understanding of how Jupyter Notebook works practically. This application helps in programming the code according to the data set of Australia in the sector of Rainfall Prediction which can be understood easily.

2.2 SCOPE OF THE PROJECT:

Every project is carried out to achieve a set of goals with some conditions keeping in mind that it should be easy to use, feasible and user friendly. As the goal of this project is to Predict the Rainfall, this project will be designed keeping in mind the conditions stated above. The proposed project would cover:

- Developing a project that effectively works on predicting rainfall.
- Implementing it on a computer system.

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

3.1 VISUALIZATION USED:

3.1.1 MAP VIEW:

Created a Map visual to show regions on a map to show the locations from where the data is collected. Unlike the Shape map visual, Map view shows precise geographical locations of data points on a map.

Its main purpose is to compare regions on a map.

The Map visual is only available in Power BI Desktop and not in Jupyter Notebook. Since it is in preview, it must be enabled before you can use it.

To enable Map, select File > Options and Settings > Options > Preview Features, then select the Map visual checkbox. You'll need to restart Power BI Desktop after you make the selection.



Fig 3.1.1.a Shows the Map visual done in Power Bi

3.1.2 ALGORITHM USED:

We have used Random Forest Classifier Algorithm in our project for best accuracy. we have got an accuracy of 85%.

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.

For classification tasks, the output of the random forest is the class selected by most trees.

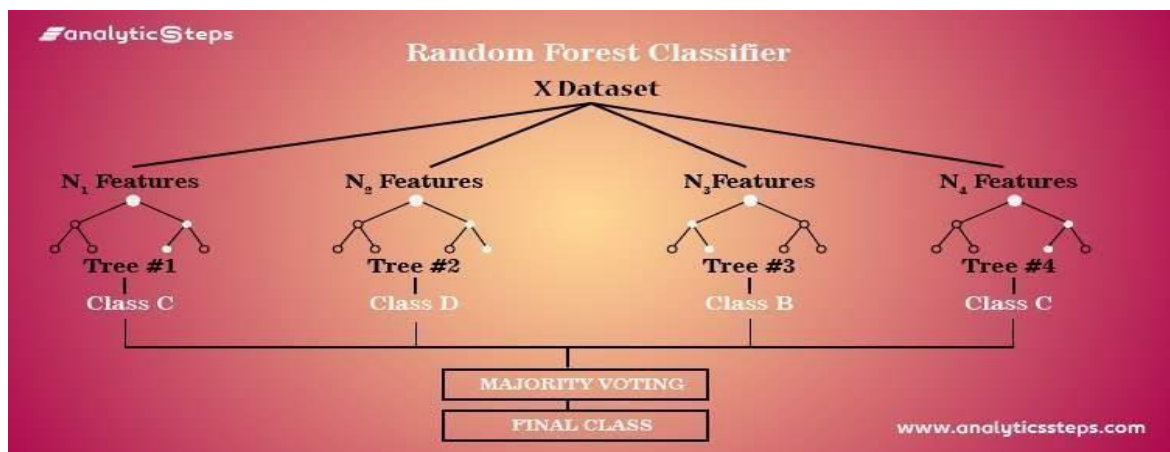


Fig 3.1.1.b Shows the diagram of Random Forest Classifier

3.1. 2 IMPORTING LIBRARIES:

1) To import Numpy and Pandas libraries in Jupyter Notebook.

Import numpy as np

Import pandas as pd

2) Read csv file (weatherAus) by using pandas.

IMPORTING LIBRARIES

```
[1]: import numpy as np  
import pandas as pd
```

Fig 3.1.2.a shows importing libraries

3.1.3 IMPORTING DATASET:

1) Import the dataset csv file by using pandas by giving the command:

```
dataset = pd.read_csv('weatherAUS.csv')
```

2) To view the top 5 rows of the DataFrame by using the following command:

```
dataset.head()
```

3) To display total Number of observations and Features of the DataFrame:

```
dataset.shape
```

4) To know the concise summary of the DataFrame:

```
dataset.info()
```

5) To know the descriptive statistics of the DataFrame:

```
dataset.describe()
```

6) To print X values from the dataset:

```
print(X)
```


7) To print Y values from the dataset:

```
print(Y)
```

8) To change the matrix shape of Y values:

```
Y=Y.reshape(-1,1)
```

IMPORTING DATASET

```
[2]: dataset = pd.read_csv('Rainfall Prediction.csv')
X = dataset.iloc[:, [1,2,3,4,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]].values
Y = dataset.iloc[:, -1].values
```

```
[3]: print(X)
```

```
[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Uluru' 5.4 26.9 ... 12.5 26.1 'No']
 ['Uluru' 7.8 27.0 ... 15.1 26.0 'No']
 ['Uluru' 14.9 nan ... 15.0 20.9 'No']]
```

```
[4]: print(Y)
```

```
['No' 'No' 'No' ... 'No' 'No' nan]
```

Fig 3.1.3.a Shows Importing Dataset.

3.1.4 DEALING WITH INVALID DATA:

To deal with invalid data:

- 1) use sklearn.impute module.
- 2) from sklearn.impute module import class SimpleImputer.
- 3) create an instance for class SimpleImputer as:
`imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')`
- 4) now replace x and y values with most frequent values from the imputer instance as:

```
X = imputer.fit_transform(X)
```

```
Y = imputer.fit_transform(Y)
```

- 5) now print X and Y most frequent values:

```
print(X)
```

```
print(Y)
```

DEALING WITH INVALID DATASET

```
[7]: from sklearn.impute import SimpleImputer
      imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
      X = imputer.fit_transform(X)
      Y = imputer.fit_transform(Y)
```

```
[8]: print(X)

[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Uluru' 5.4 26.9 ... 12.5 26.1 'No']
 ['Uluru' 7.8 27.0 ... 15.1 26.0 'No']
 ['Uluru' 14.9 20.0 ... 15.0 20.9 'No']]
```

```
[9]: print(Y)

[['No']
 ['No']
 ['No']
 ...
 ['No']
 ['No']
 ['No']]
```

Fig 3.1.4 a Shows Dealing with Invalid Data.

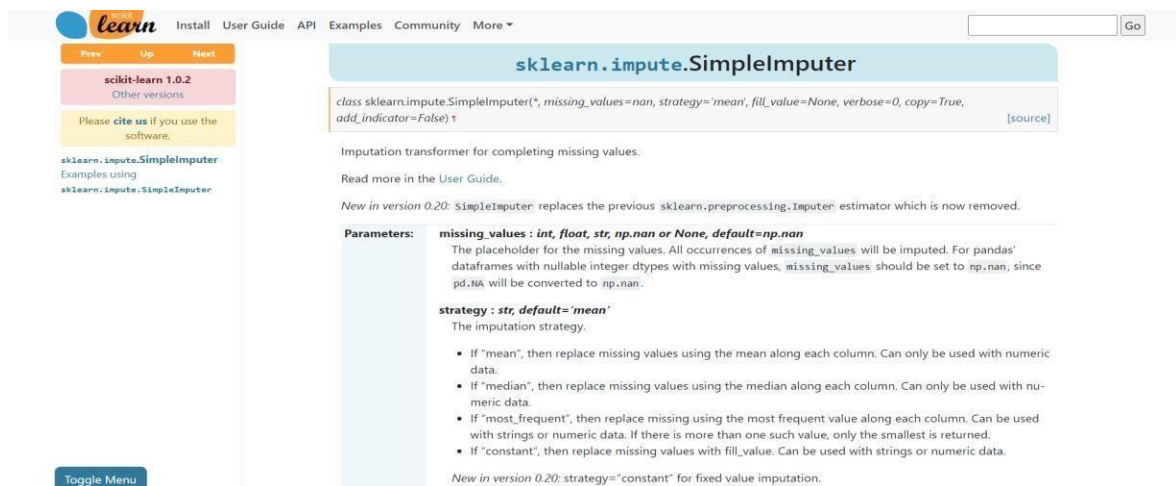


Fig 3.1.4 b Shows more info about sklearn.impute module.

3.1.5 FEATURE SCALING AND TRAINING MODEL:

FEATURES SCALING

```
[14]: from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X = sc.fit_transform(X)

[15]: print(X)

[[-1.53166617  0.19132753 -0.04135977 ... -0.01407077  0.02310362
  -0.52979545]
 [-1.53166617 -0.75105231  0.26874452 ...  0.03244663  0.387799
  -0.52979545]
 [-1.53166617  0.11279588  0.35331842 ...  0.62166712  0.22733303
  -0.52979545]
 ...
 [ 1.20928479 -1.06517892  0.52246622 ... -0.69632607  0.65037966
  -0.52979545]
 [ 1.20928479 -0.68822699  0.53656187 ... -0.29317521  0.63579185
  -0.52979545]
 [ 1.20928479  0.42692249 -0.45013361 ... -0.30868102 -0.10818671
  -0.52979545]]
```

Fig 3.1.5 a Shows Features Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

TRAINING MODEL

```
[19]: from sklearn.ensemble import RandomForestClassifier
      classifier = RandomForestClassifier(n_estimators=100, random_state=0)
      classifier.fit(X_train, Y_train)
```

Fig 3.1.5 b Shows Training Model

Model training is at the heart of the data science development lifecycle where the data science team works to fit the best weights and biases to an algorithm to minimize the loss function over prediction range. Loss functions define how to optimize the ML algorithms. A data science team may use different types of loss functions depending on the project objectives, the type of data used and type of algorithm.

When a supervised learning technique is used. Model training creates mathematical representation of the relationship between the data features and a target label. In unsupervised learning. It creates a mathematical representation among the data features themselves.

Importance of Model Training:

Model training is the primary step in machine learning, resulting in a working model that can then be validated, tested and deployed. The model's performance during training will determine how well it will work.

3.2 SOFTWARE REQUIREMENTS:

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

The software requirements for this project are:

- Operating system: - Windows
- Dataset: - Microsoft Excel

3.3 HARDWARE REQUIREMENTS:

The most common set of requirements defined by an operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL list is tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following listed discuss the various aspects of hardware requirements.

Hardware requirements for this project:

Processor: Intel processor

i3/i5/i7 Ram: 2 GB hard

CPU: 1 gigahertz (GHz) 64-bit (x64) processor or better recommended

3.4 SYSTEM IMPLEMENTATION:

3.4.1 HISTORY OF JUPYTER NOTEBOOK:

In 2014, Fernando Perez announced a spin off project from IPython called Project Jupyter. IPython continues to exist as a Python shell and kernel for Jupyter, while the notebook and other language agnostic and it supports execution environments. In 2015 GitHub and the Jupyter Project announced native rendering of Jupyter notebooks file format (.ipynb files) on the GitHub platform. Project Jupyter's operating philosophy is to support interactive data science and scientific computing across all programming languages via the development of open source software. According to the Project Jupyter website, "Jupyter will always be 100% open source software, free for all to use and released under the liberal terms of the modified BSD license.

The Jupyter Notebook has become a popular user interface for nvloud computing, and major cloud providers have adopted the Jupyter Notebook or derivative tools as a fronted interface for cloud users. Examples include Amazon's SageMaker Notebooks, Google's Colaboratory and Microsoft's Azure Notebook.

Google Colaboratory (also known as Colab) is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive. Colab was originally an internal Google project; an attempt was made tom open source all the code and work more directly upstream, leading to the development of the "Open in Colab".

As of October 2019, the Colaboratory UI allows for the creation of notebooks withn Python 2 and Python 3 kernels; however, an existing notebook whose kernelspec is IR or Swift will also work.

Since both R and Swift are installed in the container. Julia language can also work on Colab (with e.g. Python and GPUs; Google's tensor processing units also work with Julia on Colab).

3.4.2 FEATURES OF JUPYTER NOTEBOOK:

The unique features of Jupyter Notebook are as follows:

3.4.2.1 RANGE OF ATTRACTIVE FEATURES:

Having familiarized ourselves with the interface of two platforms for running Notebooks (Jupyter Notebook and JupyterLab), we are ready to start writing and running our project in Jupyter Notebook. Jupyter has many appealing core features that make efficient Python programming. These include an assortment of things to do.

In Jupyter Notebook you can use features such as:

- Run
- Importing numpy and getting the arrange docstring
- Getting the Python sorted function docstring
- We can pull up a list of the available functions on an object
- We can click an empty code cell in the Tab Completion section
- We can list the available modules when importing external libraries
- We can list the available modules of imported external libraries
- We can perform function and variable completion
- We can list the available magic commands
- `Plt.show()`
- Can declare the a variable
- Getting the runtime for the entire cell
- Getting the runtime for one line
- We can check the average results of multiple runs
- Listing the contents of the working directory with `ls`
- Listing the working directory with `pwd`
- We can run the `%load_ext sql` cell to load the external command into the notebook

3.4.2.2 GETTING DATA:

Getting Data in Jupyter Notebook, users to select from a range of data sources. The data sources are anywhere in the spectrum from on-premise to cloud-based, unstructured to structured. New data sources are added every month.

Some of the latest available data sources are as follows:

- Jupyter Notebook datasets
- Excel
- Power BI dataflows
- SQL Server
- MySQL database

- Analysis Services
- Azure
- Text/CSV
- Oracle
- PDF
- Access
- XML
- JSON

3.4.2.3 CUSTOMIZABLE DASHBOARD:

The dashboard layout extension is an add on for Jupyter Notebook. It lets you arrange your notebook outputs (text, plots, widgets, ...) in grid or report like layouts. it saves information about your layouts in your notebook document.

Other people with the extension can view your layouts.

3.4.3 JUPYTER NOTEBOOK WIDGETS:

Widgets are eventful python objects that have a representation in the browser, often as a control like a slider, text box, etc. you can use widgets to build interactive GUIs for your notebooks.

We present these widgets:

- IntSlider
- FloatSlider
- FloatLogSlider
- IntRangeSlider
- FloatRangeSlider
- IntProgress
- FloatProgress
- BoundedIntText
- BoundedFloatText
- IntText

- FloatText
- ToggleButton
- Checkbox
- Valid
- Dropdown
- RadioButtons
- Select
- SelectionSlider, etc

3.5 CODE EDITOR:

Notebooks have always been a tool for the incremental development of software ideas. Data scientists use Jupyter to journal their work, explore and experiment with novel algorithms, quickly sketch new approaches and immediately observe the outcomes. Moreover, Jupyter is moving towards becoming a full-fledged IDE; just not an IDE you are used to.

With its great extensions and libraries like `kale` and `nbdev`, it is certainly capable of doing much more than just drafting an idea. Check the editor carefully to find out more.

However, once every blue moon, we may want to edit a `.py` file. This file may hold some utility functions we import in the Notebook or define our models' classes. It's a good practice to work like that.

But the text editor bundled with Jupyter is just that: a simple, featureless text editor. So, what can we do? There are efforts like this one, which tries to integrate the `monaco` editor (the code editor which powers VS Code) into Jupyter.

The first plugin we will install enables folding. Have you ever seen the `numpy` source code? Usually, the docstring of a function takes up the whole space. Let's fold it by default.

Data used in this project looks likes:

Fig 3.5.a Shows the Dataset

You can see all the columns in your dataset along with the column type. For instance, the Name column has type ABC, which means the column contains the text. You can actually click on the column type to see all the column types and their names as shown in the following screenshot.

Rainfall Prediction.csv - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Clipboard Font Alignment Number Styles Cells Editing

B1 Location

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporatic	Sunshine	WindGust	WindGust	WindDir9	WindDir3	WindSpee	WindSpee	Humidity9	Humidity3	Press
2	12/1/2008	Albury	13.4	22.9	0.6	NA	NA	W	44	W	WNW	20	24	71	22	10
3	12/2/2008	Albury	7.4	25.1	0	NA	NA	WNW	44	NNW	WSW	4	22	44	25	10
4	12/3/2008	Albury	12.9	25.7	0	NA	NA	WSW	46	W	WSW	19	26	38	30	10
5	12/4/2008	Albury	9.2	28	0	NA	NA	NE	24	SE	E	11	9	45	16	10
6	12/5/2008	Albury	17.5	32.3	1	NA	NA	W	41	ENE	NW	7	20	82	33	10
7	12/6/2008	Albury	14.6	29.7	0.2	NA	NA	WNW	56	W	W	19	24	55	23	10
8	12/7/2008	Albury	14.3	25	0	NA	NA	W	50	SW	W	20	24	49	19	10
9	12/8/2008	Albury	7.7	26.7	0	NA	NA	W	35	SSE	W	6	17	48	19	10
10	12/9/2008	Albury	9.7	31.9	0	NA	NA	NNW	80	SE	NW	7	28	42	9	10
11	#####	Albury	13.1	30.1	1.4	NA	NA	W	28	S	SSE	15	11	58	27	1
12	#####	Albury	13.4	30.4	0	NA	NA	N	30	SSE	ESE	17	6	48	22	10
13	#####	Albury	15.9	21.7	2.2	NA	NA	NNE	31	NE	ENE	15	13	89	91	10
14	#####	Albury	15.9	18.6	15.6	NA	NA	W	61	NNW	NNW	28	28	76	93	9
15	#####	Albury	12.6	21	3.6	NA	NA	SW	44	W	SSW	24	20	65	43	10
16	#####	Albury	8.4	24.6	0	NA	NA	NA	NA	S	WNW	4	30	57	32	10

Rainfall Prediction

Fig 3.5. b Shows the column types

CHAPTER 4

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 RESULTS:

Jupyter Notebook is a programming based application which we used in our project to Predict Rainfall using Machine Learning Algorithm.

The dataset is successfully imported to Jupyter Notebook and we have completed our project. From this project we can easily Predict Rainfall.

4.2 FEASIBILITY STUDY:

Jupyter Notebook is developed by Professor Fernando Perez in February 2015 and it was first released to the general public on February 20, 2018. Jupyter Notebook is a programming based application, from wide range of data sources, which can be used for Application development and Machine Learning.

Jupyter Notebook is user friendly and simple that power users and business analysts can work with it. Jupyter Notebook is a powerful tool that can be used by developers in enterprise systems for modeling scenarios and complex data mash- up.

4.2.1 BENEFITS:

- Free.
- Easy start up, just type jupyter notebook in your terminal.
- Visualization display.
- Text editing.
- SQL adaptability

4.3 IMPLEMENTATION:

- We have designed our project by using Jupyter Notebook application.
- We collected data called WeatherAus dataset from Kaggle .com.
- Requirements for this project : -
- Software: Jupyter Notebook, Excel, Google.
- Hardware: PC or Desktop.
- For constructing of successful project we have to collect good data.
- For our project we have collected our data source from kaggle.com.
- After collecting dataset, we have to imported it to Jupyter Notebook.
- After importing the dataset code our machine lesrning algorithm.
- Now we have to collect our data source, we need to examine the quality of data.
- The easiest way to check the quality of data is to importing the data to Jupyter Notebook and checking the quality of data running data filtration option.

CHAPTER 5

SUMMARY AND CONCLUSIONS

5.1 SUMMARY:

JUPYTER NOTEBOOK:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more..

According to the official website of [Jupyter](https://jupyter.org), Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages..

FUNCTIONALITIES:

One major feature of the Jupyter notebook is the ability to display plots that are the output of running code cells. The IPython kernel is designed to work seamlessly with the [matplotlib](https://matplotlib.org/) plotting library to provide this functionality. Specific plotting library integration is a feature of the kernel.

MAIN FEATURES OF WEB APPLICATIONS:

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Displaying the result of computation using rich media

- Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.
- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.
- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.
- Some of the main functions of Jupyter Notebook are

Starting the notebook server:

You can start running a notebook server from the command line using the following command:

This will print some information about the notebook server in your console, and open a web browser to the URL of the web application (by default, <http://127.0.0.1:8888>).

The landing page of the Jupyter notebook web application, the dashboard, shows the notebooks currently available in the notebook directory (by default, the directory from which the notebook server was started).

You can create new notebooks from the dashboard with the New Notebook button, or open existing ones by clicking on their name. You can also drag and drop ipynb notebooks and standard .py Python source code files into the notebook list area.

When starting a notebook server from the command line, you can also open a particular notebook directly, bypassing the dashboard, with `jupyter notebook my_notebook.ipynb`. The ipynb extension is assumed if no extension is given.

When you are inside an open notebook, the File | Open... menu option will open the dashboard in a new browser tab, to allow you to open another notebook from the notebook directory or to create a new notebook.

JUPYTER NOTEBOOK :

DOWNLOAD OF JUPYTER NOTEBOOK:

Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download [Anaconda](#). We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook. To run the notebook:

```
jupyter notebook
```

As an existing Python user, you may wish to install Jupyter using Python's package manager, [pip](#), instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

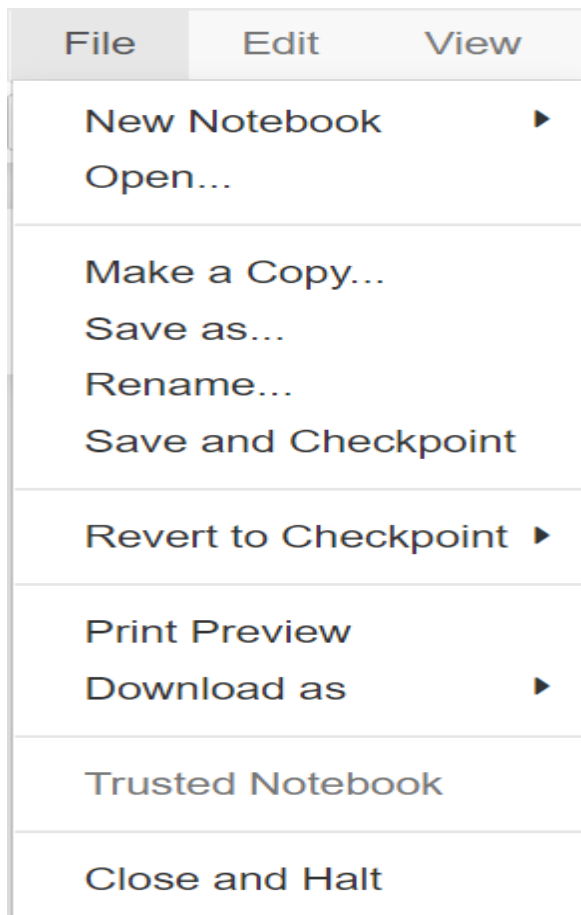
Then install the Jupyter Notebook using:

```
pip3 install jupyter
```

While Jupyter runs code in many programming languages, **Python** is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook.

We recommend using the [Anaconda](#) distribution to install Python and Jupyter. We'll go through its installation in the next section.

files of jupyter notebook



5.2 CONCLUSION:

- By using jupyter notebook the Prediction of data for Rainfall data set is implemented.
- We have achieved the project upto our limit.
- We are unable to do the project on a certain country because it requires meta data which is beyond our scope.

REFERENCES

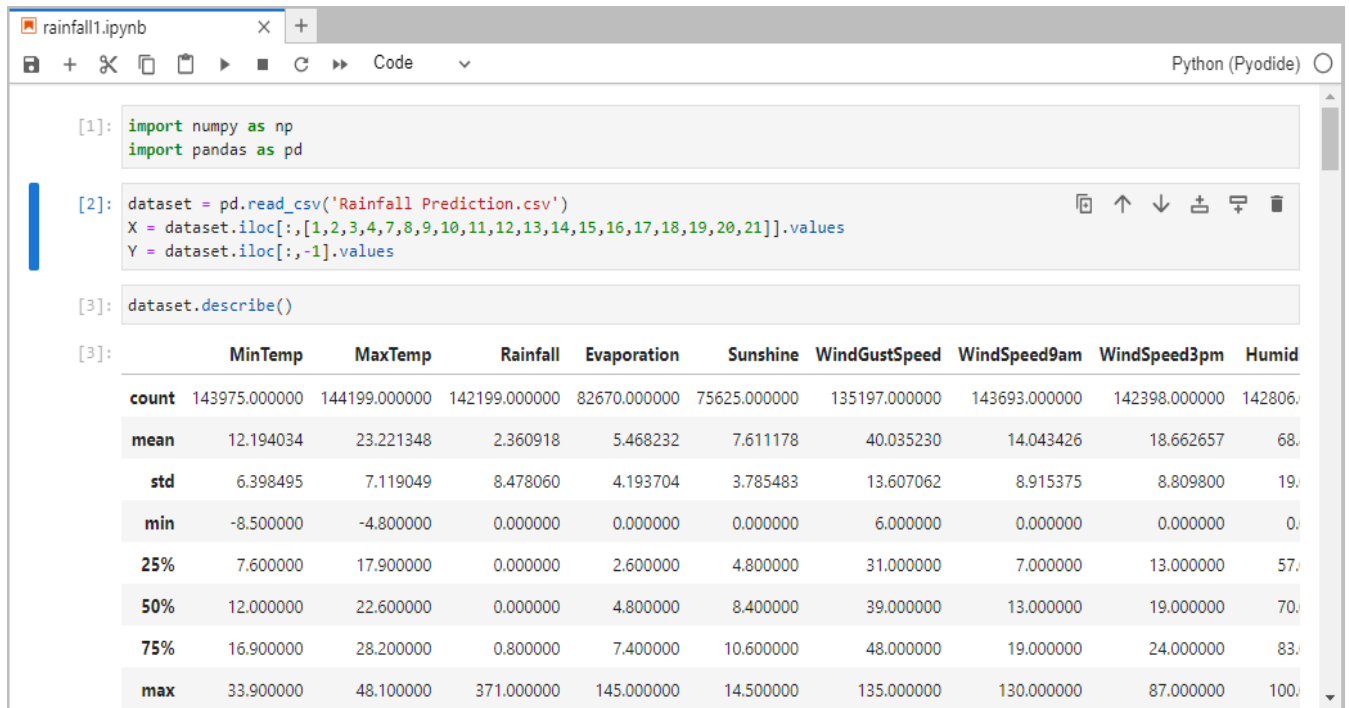
- ***Data Reference from:***
Kaggle datasets download -d jsphyg/weather-dataset-rattle-package
- ***Machine Learning coding reference from:***
Sklearn.impute. SimpleImputer
- On collage I learnt machine learning from Sathyabama mam and my guide Ms.K.Dhanalakshmi mam Helped me a lot during this project.

WEBSITES:

- <http://www.bom.gov.au/climate/data>
- <http://www.bom.gov.au/climate/dwo/IDCJDW2801.latest.sht>
- <http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml>
- <http://www.bom.gov.au/climate/dwo/> and <http://www.bom.gov.au/climate/data>.

APPENDIX

A. Screenshots

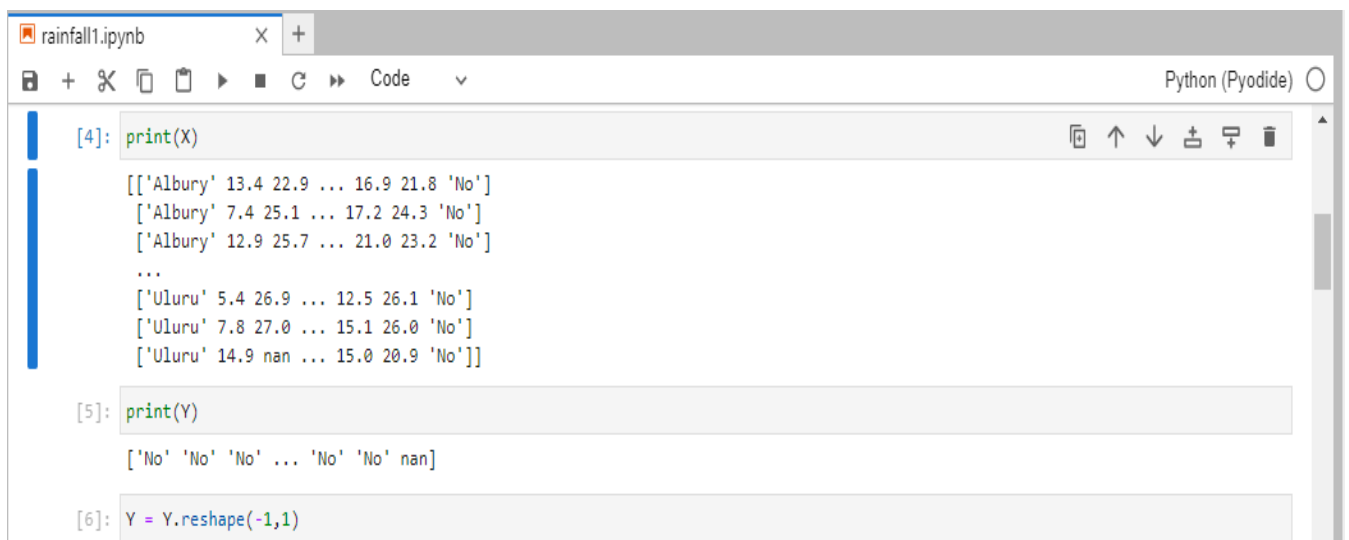


```
[1]: import numpy as np
import pandas as pd

[2]: dataset = pd.read_csv('Rainfall Prediction.csv')
X = dataset.iloc[:, [1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]].values
Y = dataset.iloc[:, -1].values

[3]: dataset.describe()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humid
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000000	143693.000000	142398.000000	142806.000000
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035230	14.043426	18.662657	68.000000
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607062	8.915375	8.809800	19.000000
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000



```
[4]: print(X)

[['Albury' 13.4 22.9 ... 16.9 21.8 'No']
 ['Albury' 7.4 25.1 ... 17.2 24.3 'No']
 ['Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 ['Uluru' 5.4 26.9 ... 12.5 26.1 'No']
 ['Uluru' 7.8 27.0 ... 15.1 26.0 'No']
 ['Uluru' 14.9 nan ... 15.0 20.9 'No']]

[5]: print(Y)

['No' 'No' 'No' ... 'No' 'No' nan]

[6]: Y = Y.reshape(-1,1)
```

DEALING WITH INVALID DATASET

```
rainfall1.ipynb Python (Pyodide)
```

```
[7]: from sklearn.impute import SimpleImputer
      imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
      X = imputer.fit_transform(X)
      Y = imputer.fit_transform(Y)
```

```
[8]: print(X)
[[ 'Albury' 13.4 22.9 ... 16.9 21.8 'No']
 [ 'Albury'  7.4 25.1 ... 17.2 24.3 'No']
 [ 'Albury' 12.9 25.7 ... 21.0 23.2 'No']
 ...
 [ 'Uluru'  5.4 26.9 ... 12.5 26.1 'No']
 [ 'Uluru'  7.8 27.0 ... 15.1 26.0 'No']
 [ 'Uluru' 14.9 20.0 ... 15.0 20.9 'No']]
```

```
[9]: print(Y)
[['No']
 ['No']
 ['No']
 ...
 ['No']
 ['No']
 ['No']]
```

ENCODING DATASET

```
rainfall1.ipynb Python (Pyodide)
```

```
[10]: from sklearn.preprocessing import LabelEncoder
      le1 = LabelEncoder()
      X[:,0] = le1.fit_transform(X[:,0])
      le2 = LabelEncoder()
      X[:,4] = le2.fit_transform(X[:,4])
      le3 = LabelEncoder()
      X[:,6] = le3.fit_transform(X[:,6])
      le4 = LabelEncoder()
      X[:,7] = le4.fit_transform(X[:,7])
      le5 = LabelEncoder()
      X[:, -1] = le5.fit_transform(X[:, -1])
      le6 = LabelEncoder()
      Y[:, -1] = le6.fit_transform(Y[:, -1])
```

```
[11]: print(X)
[[2 13.4 22.9 ... 16.9 21.8 0]
 [2  7.4 25.1 ... 17.2 24.3 0]
 [2 12.9 25.7 ... 21.0 23.2 0]
 ...
 [41 5.4 26.9 ... 12.5 26.1 0]
 [41 7.8 27.0 ... 15.1 26.0 0]
 [41 14.9 20.0 ... 15.0 20.9 0]]
```

```
rainfall1.ipynb x +
Python (Pyodide)

[12]: print(Y)

[[0]
 [0]
 [0]
 ...
 [0]
 [0]
 [0]]

[13]: Y = np.array(Y, dtype=float)
      print(Y)

[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]
 [0.]]
```

FEATURE SCALING

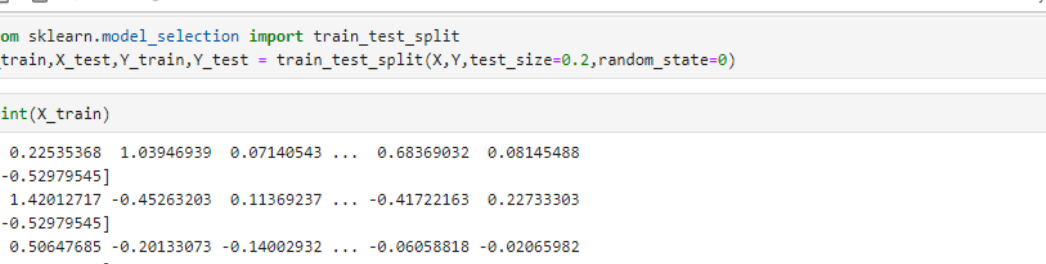
```
rainfall1.ipynb x +
Python (Pyodide)

[14]: from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X = sc.fit_transform(X)

[15]: print(X)

[[-1.53166617  0.19132753 -0.04135977 ... -0.01407077  0.02310362
 -0.52979545]
 [-1.53166617 -0.75105231  0.26874452 ...  0.03244663  0.387799
 -0.52979545]
 [-1.53166617  0.11279588  0.35331842 ...  0.62166712  0.22733303
 -0.52979545]
 ...
 [ 1.20928479 -1.06517892  0.52246622 ... -0.69632607  0.65037966
 -0.52979545]
 [ 1.20928479 -0.68822699  0.53656187 ... -0.29317521  0.63579185
 -0.52979545]
 [ 1.20928479  0.42692249 -0.45013361 ... -0.30868102 -0.10818671
 -0.52979545]]
```

SPLITTING DATASET INTO TRAINING SET AND TEST SET



```
rainfall1.ipynb
```

Python (Pyodide)

```
[16]: from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
[17]: print(X_train)
```

```
[[ 0.22535368  1.03946939  0.07140543 ...  0.68369032  0.08145488
-0.52979545]
 [ 1.42012717 -0.45263203  0.11369237 ... -0.41722163  0.22733303
-0.52979545]
 [ 0.50647685 -0.20133073 -0.14002932 ... -0.06058818 -0.02065982
 1.88752093]
 ...
 [ 1.0687232  0.75675544  0.93124006 ...  1.10234698  1.07342629
-0.52979545]
 [ 0.57675765 -0.04426743 -0.16822062 ...  0.01694083 -0.28324049
 1.88752093]
 [ 1.63096955 -0.0285611 -0.91529006 ... -0.35519842 -0.76463838
-0.52979545]]
```

```
[18]: print(Y_train)
```

```
[[1.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
```

TRAINING MODEL

```
rainfall1.ipynb
```

```
[19]: from sklearn.ensemble import RandomForestClassifier
      classifier = RandomForestClassifier(n_estimators=100, random_state=0)
      classifier.fit(X_train, Y_train)

      <ipython-input-19-84bb09ba651d>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
      classifier.fit(X_train, Y_train)

[19]: > RandomForestClassifier

[20]: classifier.score(X_train, Y_train)

[20]: 0.9999312525780283

[21]: y_pred = le6.inverse_transform(np.array(classifier.predict(X_test), dtype=int))
      Y_test = le6.inverse_transform(np.array(Y_test, dtype=int))

      /lib/python3.11/site-packages/sklearn/preprocessing/_label.py:155: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
      y = column_or_1d(y, warn=True)

[22]: print(y_pred)

      ['No' 'No' 'No' ... 'No' 'No' 'No']

[23]: print(Y_test)

      ['Yes' 'Yes' 'No' ... 'Yes' 'No' 'No']
```

```
rainfall1.ipynb x +
Python (Pyodide)

[24]: y_pred = y_pred.reshape(-1,1)
      Y_test = Y_test.reshape(-1,1)

[25]: df = np.concatenate((Y_test,y_pred),axis=1)
      dataframe = pd.DataFrame(df,columns=['Rain on Tommorrow','Predition of Rain'])

[26]: print(dataframe)

      Rain on Tommorrow Predition of Rain
0                Yes          No
1                Yes          No
2                No           No
3                No           Yes
4                No           No
...              ...           ...
29087             No           Yes
29088             No           No
29089             Yes          No
29090             No           No
29091             No           No

[29092 rows x 2 columns]
```

CALCULATING ACCURACY

```
rainfall1.ipynb x +
Python (Pyodide)

[27]: from sklearn.metrics import accuracy_score
      accuracy_score(Y_test,y_pred)

[27]: 0.8521930427608965

[ ]:
```

SOURCE CODE :

```
import numpy as np
import pandas as pd

dataset = pd.read_csv('Rainfall Prediction.csv')
X = dataset.iloc[:, [1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]].values
Y = dataset.iloc[:, -1].values

dataset.describe()

print(X)

print(Y)

Y = Y.reshape(-1, 1)

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
X = imputer.fit_transform(X)
Y = imputer.fit_transform(Y)

print(X)

print(Y)

from sklearn.preprocessing import LabelEncoder
le1 = LabelEncoder()
X[:, 0] = le1.fit_transform(X[:, 0])
le2 = LabelEncoder()
X[:, 4] = le2.fit_transform(X[:, 4])
le3 = LabelEncoder()
X[:, 6] = le3.fit_transform(X[:, 6])
le4 = LabelEncoder()
X[:, 7] = le4.fit_transform(X[:, 7])
le5 = LabelEncoder()
X[:, -1] = le5.fit_transform(X[:, -1])
le6 = LabelEncoder()
Y[:, -1] = le6.fit_transform(Y[:, -1])

print(X)
```



```

print(Y)

Y = np.array(Y,dtype=float)
print(Y)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)

print(X)

from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)

print(X_train)

print(Y_train)

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100,random_state=0)
classifier.fit(X_train,Y_train)

classifier.score(X_train,Y_train)

y_pred = le6.inverse_transform(np.array(classifier.predict(X_test),dtype=int))
Y_test = le6.inverse_transform(np.array(Y_test,dtype=int))

print(y_pred)

print(Y_test)

y_pred = y_pred.reshape(-1,1)
Y_test = Y_test.reshape(-1,1)

df = np.concatenate((Y_test,y_pred),axis=1)
dataframe = pd.DataFrame(df,columns=['Rain on Tommorrow','Predition of Rain'])

print(dataframe)

from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)

```