PROJECT REPORT

Date	18 November 2022
Team ID	PNT2022TMID49531
Project Name	Exploratory analysis of Rainfall data in India for Agriculture

Exploratory Analysis of Rainfall Data in India for Agriculture

1.INTRODUCTION:

1.1 Project Overview

Rainfall has been a major concern these days. Weather conditions have been changing for time being. Rainfall forecasting is important otherwise, it may lead to many disasters. Irregular heavy rainfall may lead to the destruction of crops, heavy floods that can cause harm to human life. It is important to exactly determine the rainfall for effective use of water resources, crop productivity, and pre-planning of water structures.

This comparative study is conducted concentrating on the following aspects: modeling inputs, Visualizing the data, modeling methods, and pre-processing techniques. The results provide a comparison of various evaluation metrics of these machine learning techniques and their reliability to predict rainfall by analyzing the weather data.

We will be using classification algorithms such as **Linear Regression , Lasso Model, Ridge Model, SVM and Random forest** . We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. Once the model is saved, we integrate it with flask application.

1.2 Purpose

The purpose of the study is the prediction of the rainfall using historical monthly data based on Machine learning methodologies such Linear Regression , Lasso Model, Ridge Model, SVM and Random forest. The extraction procedures/algorithms will produce the output by classification of the data according to the categories using SVM and Random forest . The similar data will be grouped for the accurate and precise information that will predict rainfall more correctly and with perfect figures. Accurate rainfall prediction can help to save lives and minimize property damage. It’s also crucial for agriculture, allowing farmers to when it’s best to plant or helping them protect their crops.

2.LITERATURE SURVEY

2.1 Existing problem

In current, Accurate rainfall prediction is more difficult due to the environmental condition. It is expensive to monitor-so many variables from so many sources, Works only for linear datasets and does not work for the non linear datasets. Works well only on small scale of datasets through which it was not able to predict the rainfall for larger datasets..

2.2 References

[1] Balamurugan, M. S., &Manojkumar, R. (2021). Study of short term rain forecasting using machine learning based approach. *Wireless Networks, 27*, 5429–5434

[2] Banadkooki, F. B., Ehteram, M., Ahmed, A. N., Fai, C. M., Afan, H. A., Ridwam, W. M., Sefelnasr, A., & El-Shafie, A. (2019). Precipitation forecasting using multilayer neural networkand support vector machine optimization based on flow regime algorithm taking into accountuncertainties of soft computing models. *Sustainability, 11*(23), 6681.

[3] Sefelnasr, A., & El-Shafie, A. (2019). Precipitation forecasting using multilayer neural networkand support vector machine optimization based on flow regime algorithm taking into accountuncertainties of soft computing models. *Sustainability, 11*(23), 6681.

[4] Bojang, P. O., Yang, T.-C., Pham, Q. B., & Yu, P.-S. (2020). Linking singular spectrum analysis and machine learning for monthly rainfall forecasting. *Applied Sciences, 10*(9), 3224.

[5] Boonyuen, K., Kaewprapha, P.,Weesakul, U., & Srivihok, P. (2019). Convolutional neural network inception-v3: A machine learning approach for leveling short-range rainfall forecast model from satellite image. In *International Conference on Swarm Intelligence* (pp. 105–115).

[6] Berlin:Springer Chattopadhyay, A., Hassanzadeh, P., & Pasha, S. (2020). Predicting clustered weather patterns: A test case for applications of

convolutional neural networks to spatio-temporal climate data. Sci.Rep. **10**(1), 1–13.

[7] Chen, L., Cao, Y.,Ma, L., & Zhang, J. (2020). A deep learning based methodology for precipitation nowcasting with radar. *Earth and Space Science, 7*, e2019EA000812.
[8] Diez-Sierra, J., & del Jesus, M. (2020). Long-term rainfall prediction using atmospheric synoptic patterns in semi-arid climates with statistical and machine learning methods. *Journal of Hydrology, 586*, 124789.

[9] Hussein, E., Ghaziasgar, M., & Thron, C. (2020). Regional rainfall prediction using support vector machine classification of large-scale precipitation maps. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)* (pp. 1–8). Piscataway: IEEE.

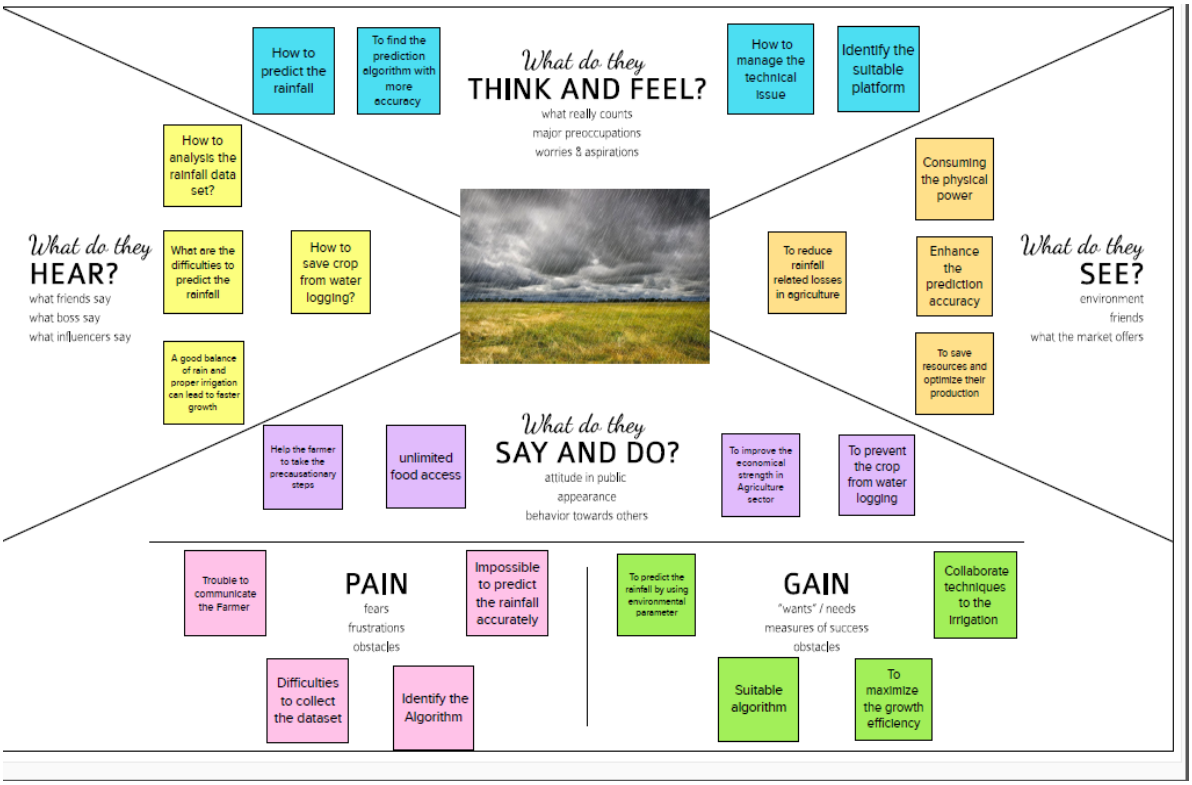
[10] Hussein, E. A., Ghaziasgar, M., Thron, C., Vaccari, M., & Bagula, A. (2021). Basic statistical estimation outperforms machine learning in monthly prediction of seasonal climatic parameters.
Atmosphere, 12(5), 539.

2.3 Problem Statement Definition

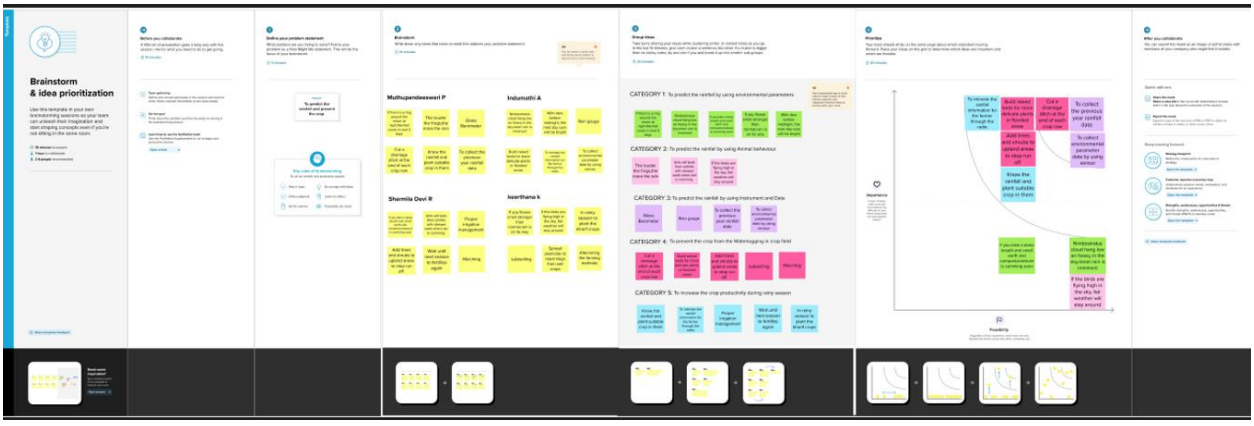
Rainfall is one of the most complex and difficult elements of the hydrology cycle to understand and to model due to the complexity of the atmospheric processes that generate rainfall and the tremendous range of variation over a wide range of scales both in space and time. Heavy rainfall prediction is a major problem for meteorological department as it is closely associated with the economy and life of human. It is a cause for natural disasters like flood and drought which are encountered by people across the globe every year. Accuracy of rainfall forecasting has great importance for countries like India whose economy is largely dependent on agriculture. Due to dynamic nature of atmosphere, Statistical techniques fail to provide good accuracy for rainfall forecasting. Thus, accurate rainfall prediction is one of the greatest challenges in operational hydrology. On a worldwide scale, large numbers of attempts have been made by different researchers to predict rainfall accurately using various techniques. But due to the nonlinear nature of rainfall, prediction accuracy obtained by these techniques is still below the satisfactory level.

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map canvas



3.2 Ideation & Brainstroming



3.3 Proposed Solution

Proposed Solution Template:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Erratic rainfall causes crop damage and reduced crop production
2.	Idea / Solution description	• To collect the previous year rainfall dataset • Data pre processing- It is a process of preparing the raw data and making it suitable for a machine learning model. • Modelling -After performing the tasks of pre- processing activities, the dataset is ready for the stage of classification, where training and test datasets are both given as input to four classification techniques • Evaluation -Accuracy, It may be defined as the number of correct predictions made as a ratio of all predictions made
3.	Novelty / Uniqueness	We will enhance the rainfall prediction accuracy.
4.	Social Impact / Customer Satisfaction	• It will help the farmers to take precautionary steps to minimize the losses. • To increase the crop production • Unlimited food access • Precautions should be taken due to heavy flooding
5.	Business Model (Revenue Model)	Our ideas is collaborated with the agriculture, it will improve the economical strength in agriculture sector.
6.	Scalability of the Solution	A trained model is adaptive According to datasets and environmental conditions.

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via OTP
FR-3	user authentication	To verifying the identify of a user
FR-4	User Reporting	The Notification will be shown when the user access the account
FR-5	User Authorization level	To check the register form and then it will give the similar access to the user
FR-6	Dataset Details	It used to find weather condition relate the user location

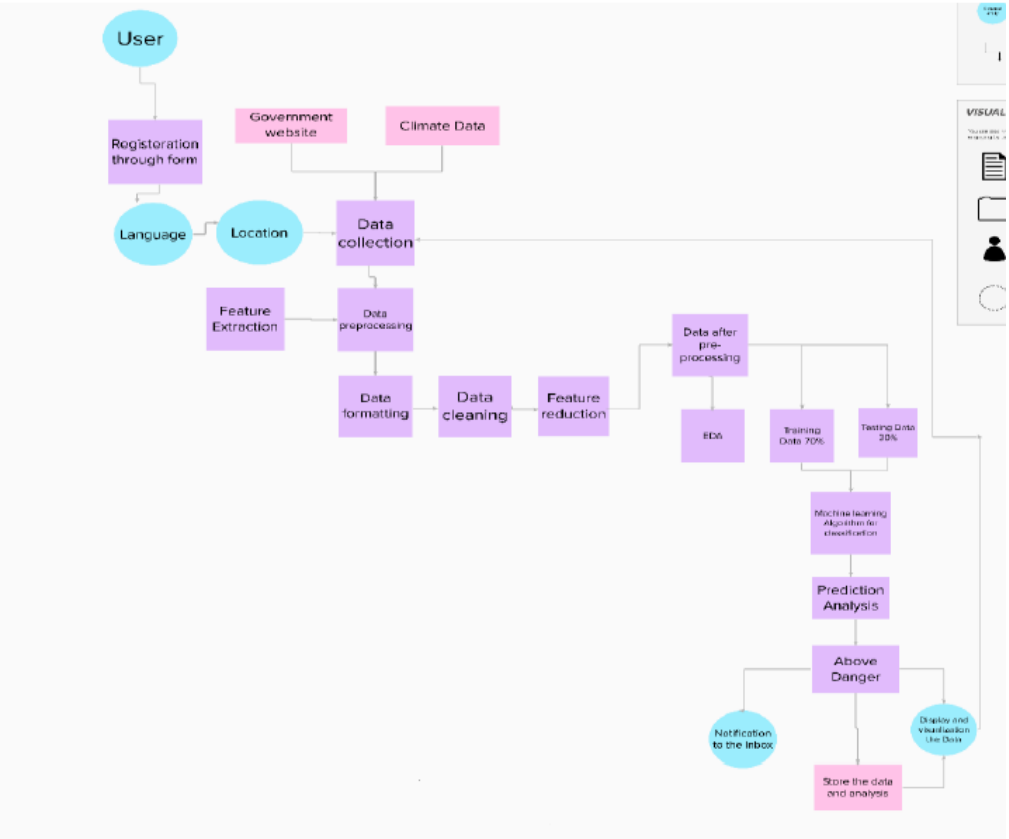
4.2 Non-Functional requirements

Non-functional Requirements:

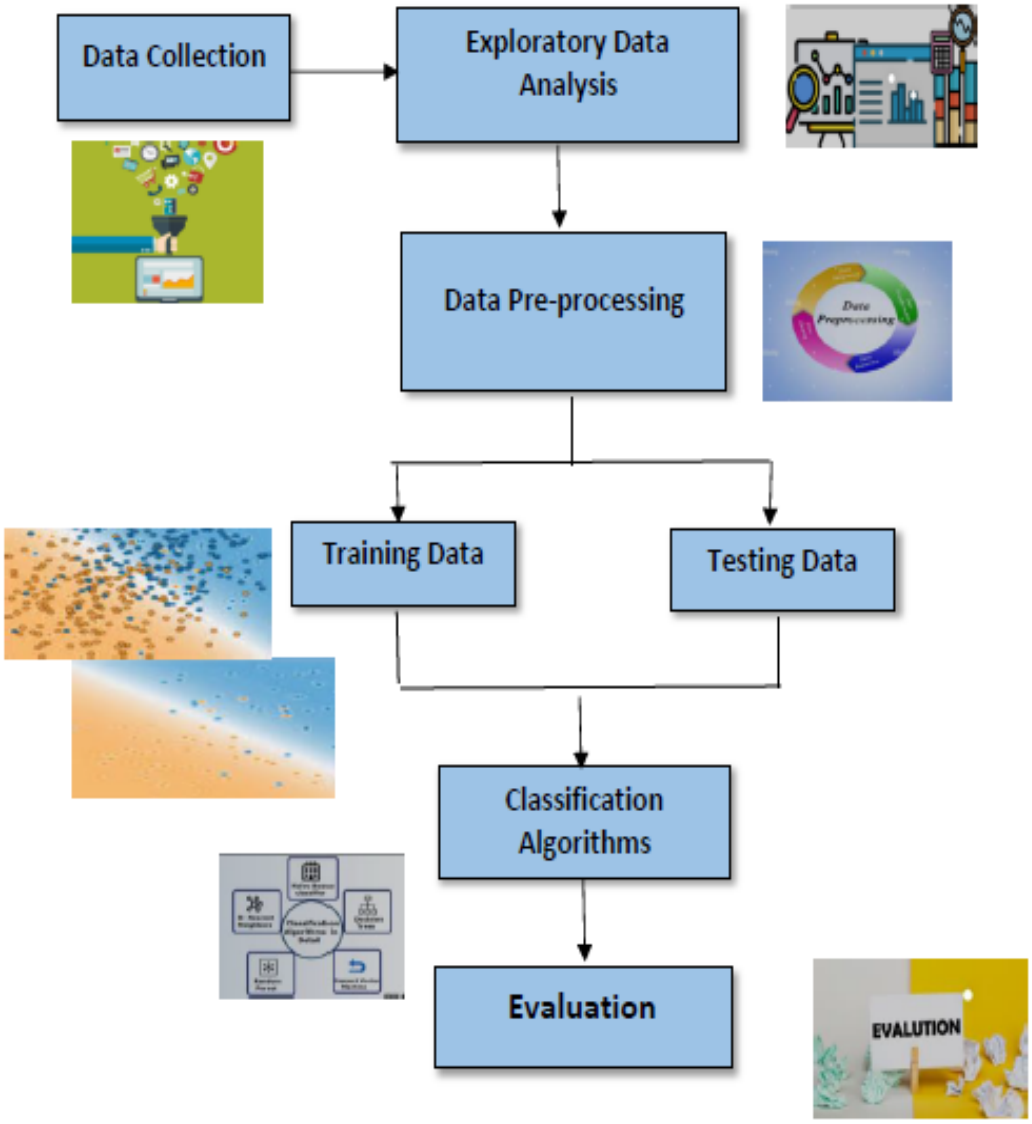
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Simple way to view all the location weather condition
NFR-2	**Security**	Without user permission, access the account then it will give the message to the user admin through the phone number
NFR-3	**Reliability**	The accuracy is better than previous accuracy in current prediction
NFR-4	**Performance**	It will respond and give the solution within three minutes after the request from the user .
NFR-5	**Availability**	User choose the multiple location and view the details
NFR-6	**Scalability**	The user may switch account in simple way

5.PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Medium	Sprint-1
		USN-3	As a user, I can register for the application through Gmail		High	Sprint-1
Customer (Travel agent)	To Know the Rainfall condition across the different cities	USN-4	I would like access to the Rainfall condition of different cities and states across the India.	I can access the forecast for five, ten and fifteen days at a location.	High	Sprint-1
Customer (Event Management Coordinator)	Accurate Rainfall prediction	USN-5	I would like to get accurate rainfall prediction so that I can plan open air events	I can visualize the rainfall predicted data earlier.	High	Sprit-1
Customer (Farmer)	Accurate Rainfall prediction	USN-6	I would like to get accurate rainfall prediction ,so that I can plan to sowing the seed	I can access the rainfall condition of different cities and states across the India.	High	Sprit-1
		USN-7	Avoid the waterlogging in crop field		High	Sprit-1

6.PROJECT PLANNING &SCHEDULING

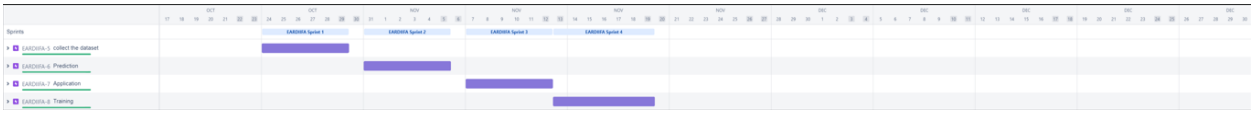
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Collect the dataset	USN-1	As a user, Analysis rainfall	2	High	Muthupandeeswari, Indumathi, Sharmila devi, Keerthana
Sprint-2	Prediction	USN-2	As a user, to build the model for prediction	3	High	Muthupandeeswari, Indumathi, Sharmila devi, Keerthana
Sprint-3	Application	USN-3	As a user, to build the application	2	Medium	Muthupandeeswari, Indumathi, Sharmila devi, Keerthana
Sprint-4	Training	USN-4	As a user, to train the model	3	High	Muthupandeeswari, Indumathi, Sharmila devi, Keerthana

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA



7.CODING & SOLUTIONING

CONNECTION TO HTML:

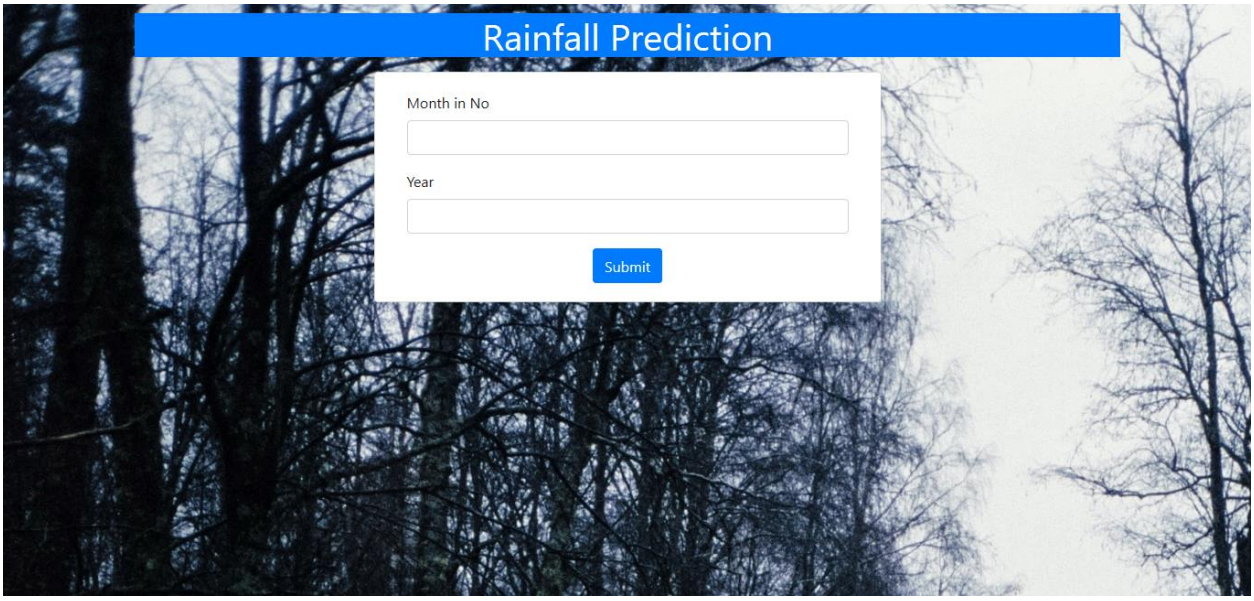
1. A user issues a request for a domain's root URL / to go to its index page
2. main.py maps the URL / to a Python function
3. The Python function finds a web template living in the templates/ folder.
4. A web template will look in the static/ folder for any images, CSSfiles it needs as it renders to HTML
5. Rendered HTML is sent back to main.py
6. main.py sends the HTML back to the browser

URL IN THE BROWSER AND BACKEND CONNECTION:

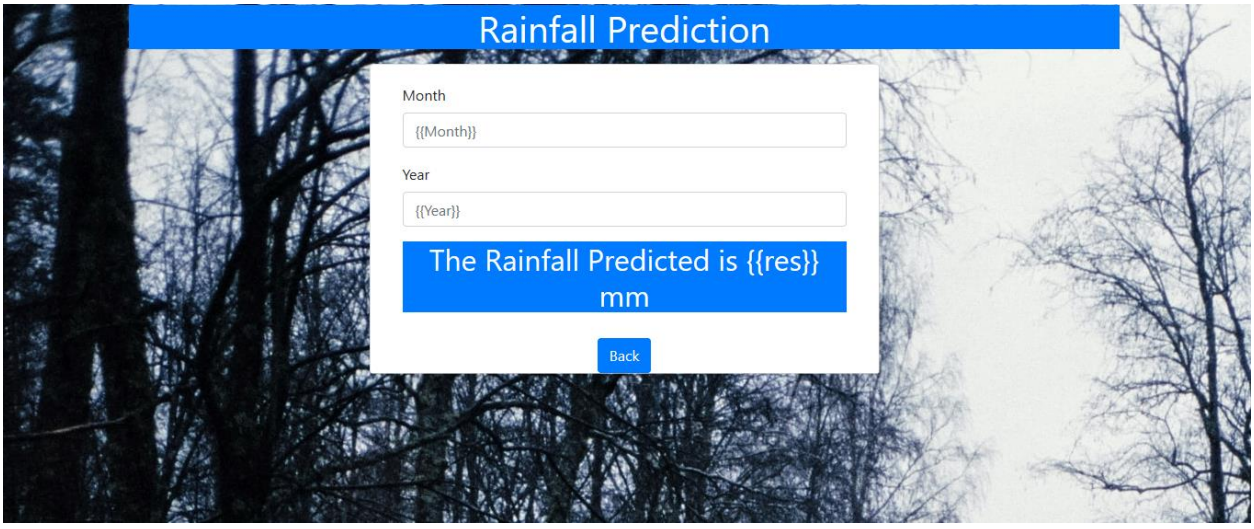
1. First. we imported the Flask class and a function render template.
2. Next, we created a new instance of the Flask class.
3. We then mapped the URL / to the function index(). Now, when someone visits this URL, the function index() will execute.
4. The function index() uses the Flask function render template() to render the index.html template we just created from the templates/ folder to the browser.
5. Finally, we use run() to run our app on a local server.

6. We'll set the debug flag to true, so we can view any applicable error messages if something goes wrong, and so that the local server automatically reloads after we've made changes to the code.
7. When we visited http://127.0.0.1:5000/, main.py had code in it, which mapped the URL / to the Python function index().
8. index() found the web template index.html in the templates/ folder, rendered it to HTML, and sent it back to the browser.

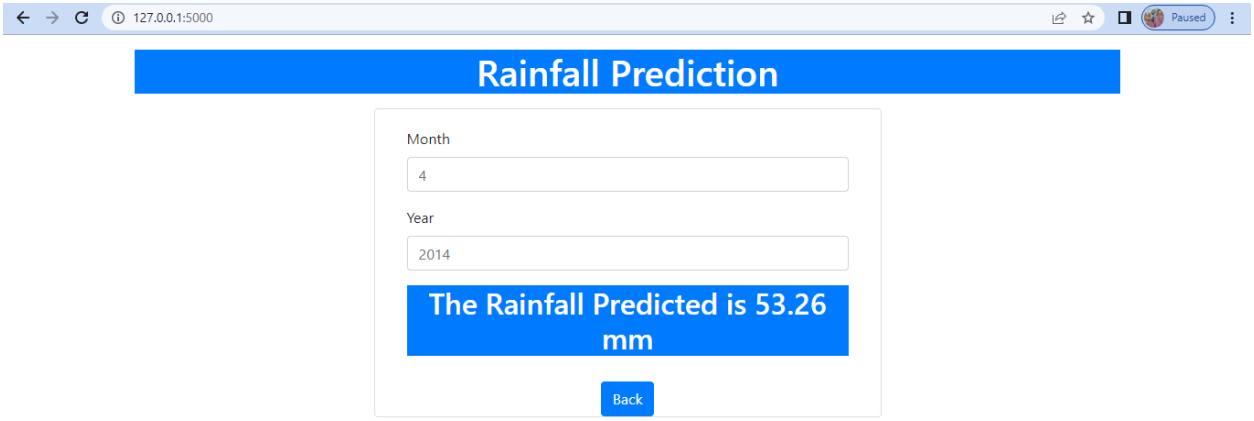
Index



Result



Output



8.TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets
LoginPage_TC_001	UI	Home Page	To predict the rainfall	Operating System: Windows 9x,2000,xp SP2 CPU:Celeron or Pentium class Processor RAM: 128 or 256MB Hard DiskSpace: 1GB	1.User enters the value of the parameter 2.the model must predict the rainfall by using the Linear model	http://127.0.0.1:5000/	Corresponding to the values entered the model must predict the rainfall by using Linear model	The rainfall value will not be obtained	Fail	–
LoginPage_TC_002	UI	Home page	To predict the rainfall	Operating System: Windows 9x,2000,xp SP2 CPU:Celeron or Pentium class Processor RAM: 128 or 256MB Hard DiskSpace: 1GB	1.User enters the value of the parameter 2.the model must predict the rainfall by using the Lasso Model	http://127.0.0.1:5000/	Corresponding to the values entered the model must predict the rainfall by using Lasso model	The rainfall value will not be obtained	Fail	–
LoginPage_TC_003	UI	Home page	To predict the rainfall	Operating System: Windows 9x,2000,xp SP2 CPU:Celeron or Pentium class Processor RAM: 128 or 256MB Hard DiskSpace: 1GB	1.User enters the value of the parameter 2.the model must predict the rainfall by using the Ridge model	http://127.0.0.1:5000/	Corresponding to the values entered the model must predict the rainfall by using Ridge model	The rainfall value will not be obtained	Fail	–
LoginPage_TC_004	UI	Home page	To predict the rainfall	Operating System: Windows 9x,2000,xp SP2 CPU:Celeron or Pentium class Processor RAM: 128 or 256MB Hard DiskSpace: 1GB	1.User enters the value of the parameter 2.the model must predict the rainfall by using the SVM Model	http://127.0.0.1:5000/	Corresponding to the values entered the model must predict the rainfall by using SVM Model	The rainfall value will not be obtained	Fail	
LoginPage_TC_005	UI	Home page	To predict the rainfall	Operating System: Windows 9x,2000,xp SP2 CPU:Celeron or Pentium class Processor RAM: 128 or 256MB Hard DiskSpace: 1GB	1.User enters the value of the parameter 2.the model must predict the rainfall by using the Random forest Model	http://127.0.0.1:5000/	Corresponding to the values entered the model must predict the rainfall by using Random Forest Model	The rainfall value will be obtained	Pass	

8.2 User Acceptance Testing

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	9	8	10	7	34
Duplicate	0	1	2	1	4
External	0	4	1	2	7
Fixed	10	9	7	10	36
Not Reproduced	0	1	1	0	2
Skipped	0	0	1	0	1
Won't Fix	1	4	1	0	6
Totals	20	27	23	20	90

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Client Application	48	0	0	48
Security	3	0	0	3
Outsource Shipping	4	0	0	4
Exception Reporting	7	0	0	7
Final Report Output	3	0	0	3
Version Control	1	0	0	1

9.RESULTS

9.1 Performance Metrics

S.No	Models	Training Accuracy	Testing Accuracy
1.	Linear regression Model	41.69999	33.1
2.	Lasso Model	41.699	33.30
3.	Ridge Model	41.699	33.30
4.	SVM Model	3.5000	1.700
5.	Random forest Model	72.6	42.1

10.ADVANTAGES AND DISADVANTAGES

Advantages:

1. It reduces overfitting in decision trees and helps to improve the accuracy
2. It is flexible to both classification and regression problems
3. It works well with both categorical and continuous values
4. It automates missing values present in the data
5. Normalising of data is not required as it uses a rule-based approach.

Disadvantages:

1. It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
2. It also requires much time for training as it combines a lot of decision trees to determine the class.
3. Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

11.CONCLUSION

A detailed survey on rainfall predictions using Random forest model over twenty-five years is done. From the survey it has been found that most of the researchers

used different models for rainfall prediction, but keras model of random forest gives significant results. Random forest is the model with least mean squared error and accurate prediction. The survey also gives a conclusion that the forecasting techniques like SVC, SVR of SVM are suitable to predict rainfall than other forecasting techniques such as statistical and numerical methods.

12.FUTURE SCOPE

Predicting the rainfall of a specific geographic location would be a challenge. Improvising the prediction model to predict the weather conditions and even predicting the loses of rainfall. Coping with the changing parameter values and making the code compatible for the changes in the parameter values. By using Random Forest instead of XGBoost

13.APPENDIX

Source Code

Import the libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import metrics
```

Read the Data

```
data = pd.read_csv("rainfall in india 1901-2015.csv")
data.head()
```

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.0
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.8

Data Exploration and pre processing

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SUBDIVISION  4116 non-null   object
 1   YEAR         4116 non-null   int64
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

```
data.isnull().sum()
```

```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

```
[5] data.duplicated().sum()
```

```
0
```

```
data['SUBDIVISION'].value_counts()
```

```
WEST MADHYA PRADESH                 115
EAST RAJASTHAN                      115
COASTAL KARNATAKA                   115
TAMIL NADU                          115
RAYALSEEMA                          115
TELANGANA                           115
COASTAL ANDHRA PRADESH              115
CHHATTISGARH                        115
VIDARBHA                            115
MATATHWADA                          115
MADHYA MAHARASHTRA                  115
KONKAN & GOA                        115
SAURASHTRA & KUTCH                  115
GUJARAT REGION                      115
EAST MADHYA PRADESH                 115
KERALA                              115
WEST RAJASTHAN                      115
SOUTH INTERIOR KARNATAKA            115
JAMMU & KASHMIR                     115
HIMACHAL PRADESH                    115
PUNJAB                              115
HARYANA DELHI & CHANDIGARH          115
UTTARAKHAND                         115
WEST UTTAR PRADESH                  115
```

```
MADHYA MAHARASHTRA                  115
KONKAN & GOA                        115
SAURASHTRA & KUTCH                  115
GUJARAT REGION                      115
EAST MADHYA PRADESH                 115
KERALA                              115
WEST RAJASTHAN                      115
SOUTH INTERIOR KARNATAKA            115
JAMMU & KASHMIR                     115
HIMACHAL PRADESH                    115
PUNJAB                              115
HARYANA DELHI & CHANDIGARH          115
UTTARAKHAND                         115
WEST UTTAR PRADESH                  115
EAST UTTAR PRADESH                  115
BIHAR                               115
JHARKHAND                           115
ORISSA                              115
GANGETIC WEST BENGAL                115
SUB HIMALAYAN WEST BENGAL & SIKKIM  115
NAGA MANI MIZO TRIPURA              115
ASSAM & MEGHALAYA                   115
NORTH INTERIOR KARNATAKA            115
LAKSHADWEEP                         114
ANDAMAN & NICOBAR ISLANDS           110
ARUNACHAL PRADESH                    97
Name: SUBDIVISION, dtype: int64
```

```
data.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fut
  """Entry point for launching an IPython kernel.
YEAR       1958.218659
JAN          18.957320
FEB          21.805325
MAR          27.359197
APR          43.127432
MAY          85.745417
JUN         230.234444
JUL         347.214334
AUG         290.263497
SEP         197.361922
OCT          95.507009
NOV          39.866163
DEC          18.870580
ANNUAL     1411.008900
Jan-Feb      40.747786
Mar-May     155.901753
Jun-Sep    1064.724769
Oct-Dec     154.100487
dtype: float64
```

```
[8] # filling na values with mean
    data = data.fillna(data.mean())
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fut
```

```
data.head(3)
```

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6

```
[10] data.isnull().any()
```

```
SUBDIVISION    False
YEAR           False
JAN            False
FEB            False
MAR            False
APR            False
MAY            False
JUN            False
JUL            False
AUG            False
SEP            False
OCT            False
NOV            False
DEC            False
ANNUAL         False
Jan-Feb        False
Mar-May        False
Jun-Sep        False
Oct-Dec        False
dtype: bool
```

```
[11] data.YEAR.unique()

array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1910, 1911, 1912,
       1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923,
       1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934,
       1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1946, 1947, 1949,
       1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960,
       1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971,
       1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982,
       1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
       1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004,
       2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
       1943, 1944, 1945, 1948, 1909])
```

```
[12] data.describe()
```

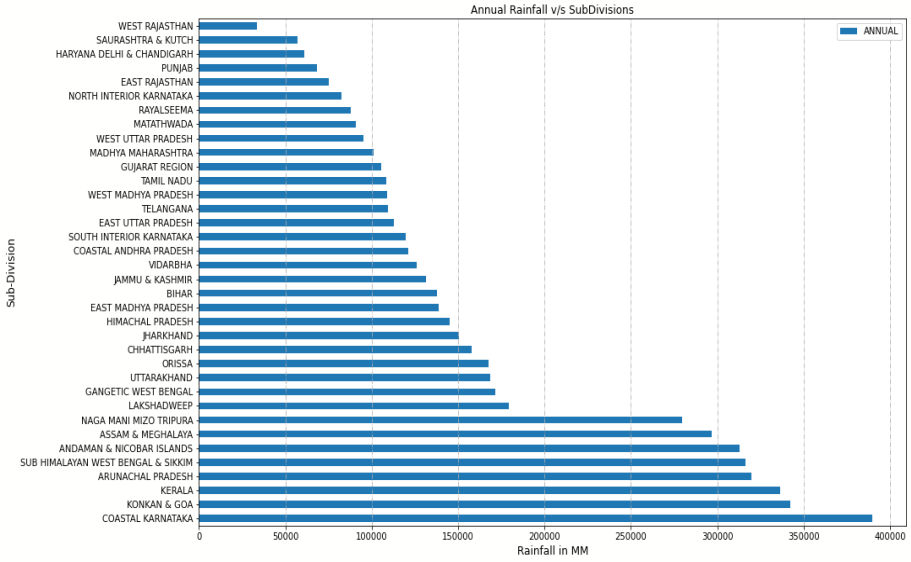
	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL
count	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000	4116.000000
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922	95.507009	39.866163	18.870580	1411.008900
std	33.140898	33.569044	35.896396	46.925176	67.798192	123.189974	234.568120	269.310313	188.678707	135.309591	99.434452	68.593545	42.318098	900.986632
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	0.000000	0.000000	0.000000	62.300000
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.475000	175.900000	156.150000	100.600000	14.600000	0.700000	0.100000	806.450000
50%	1958.000000	6.000000	6.700000	7.900000	15.700000	36.700000	138.900000	284.900000	259.500000	174.100000	65.750000	9.700000	3.100000	1125.450000
75%	1987.000000	22.125000	26.800000	31.225000	49.825000	96.825000	304.950000	418.225000	377.725000	265.725000	148.300000	45.825000	17.700000	1635.100000
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	948.300000	648.900000	617.500000	6331.100000

```
data.shape
```

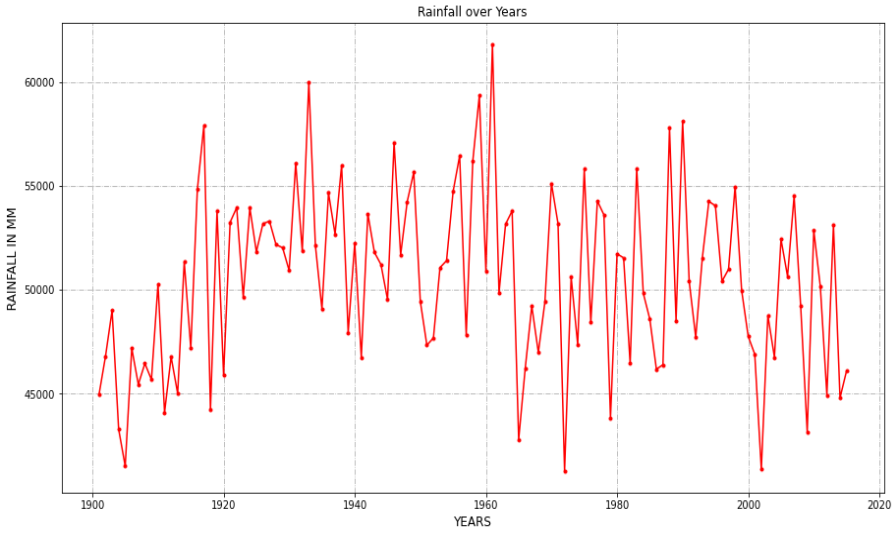
```
(4116, 19)
```

Data Visualization

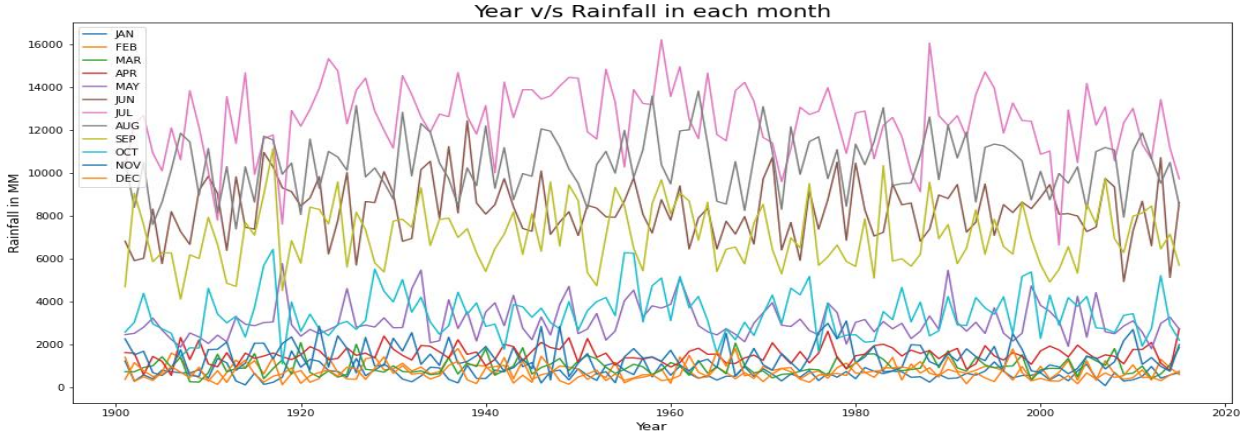
```
data[["SUBDIVISION","ANNUAL"]].groupby("SUBDIVISION").sum().sort_values(by
='ANNUAL',ascending=False).plot(kind='barh',stacked=True,figsize=(15,10))
plt.xlabel("Rainfall in MM",size=12)
plt.ylabel("Sub-Division",size=12)
plt.title("Annual Rainfall v/s SubDivisions")
plt.grid(axis="x",linestyle="-.")
plt.show()
```



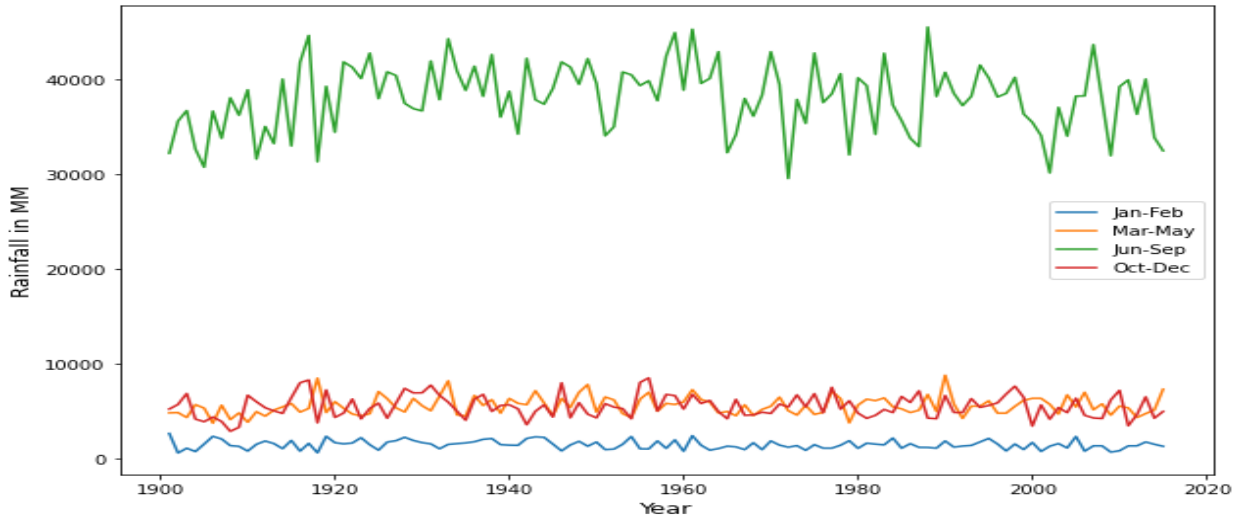
```
plt.figure(figsize=(15,8))
data.groupby("YEAR").sum()['ANNUAL'].plot(kind="line",color="r",marker="."
plt.xlabel("YEARS",size=12)
plt.ylabel("RAINFALL IN MM",size=12)
plt.grid(axis="both",linestyle="-.")
plt.title("Rainfall over Years")
plt.show()
```



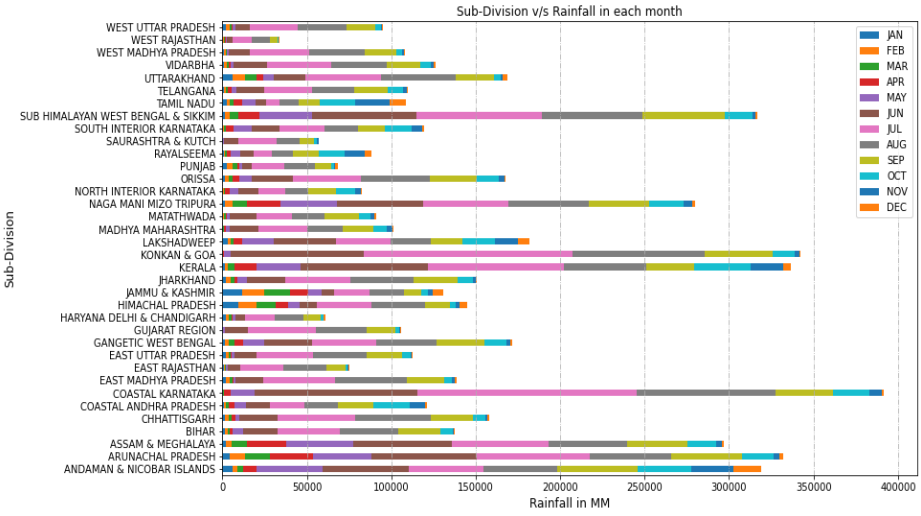

```
data[['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL','AUG', 'SEP'
,
       'OCT', 'NOV', 'DEC']].groupby("YEAR").sum().plot(kind="line",figsize
=(18,8))
plt.xlabel("Year",size=13)
plt.ylabel("Rainfall in MM",size=13)
plt.title("Year v/s Rainfall in each month",size=20)
plt.show()
```



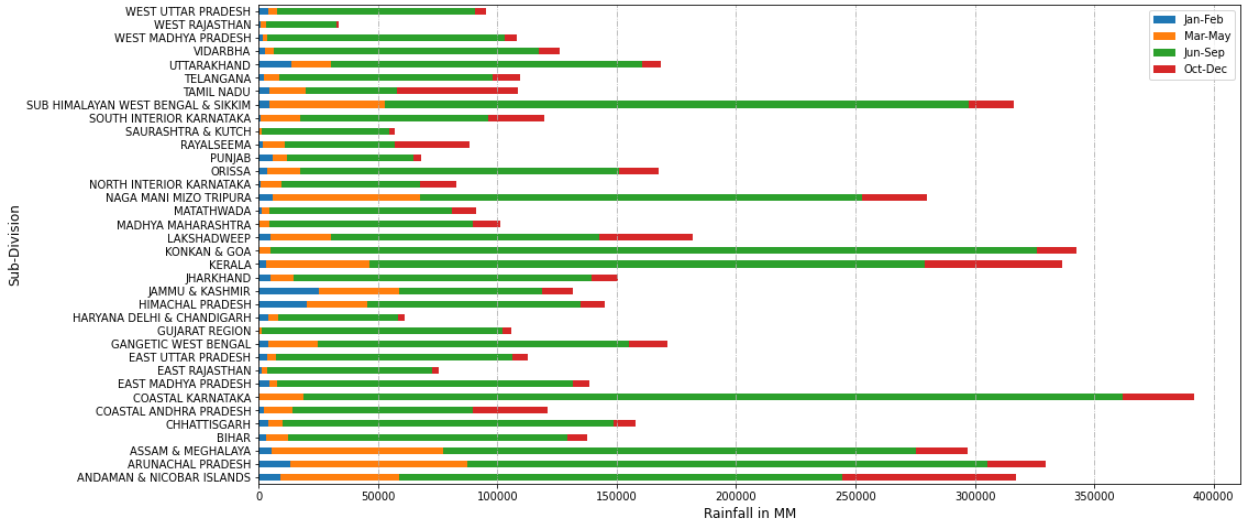
```
data[['YEAR','Jan-Feb', 'Mar-May','Jun-Sep', 'Oct-
Dec']].groupby("YEAR").sum().plot(figsize=(10,7))

plt.xlabel("Year",size=13)

plt.ylabel("Rainfall in MM",size=13)

plt.show()
```




```
data[['SUBDIVISION', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
       'AUG', 'SEP', 'OCT', 'NOV', 'DEC']].groupby("SUBDIVISION").sum().pl
ot(kind="barh",stacked=True,figsize=(13,8))
plt.title("Sub-Division v/s Rainfall in each month")
plt.xlabel("Rainfall in MM",size=12)
plt.ylabel("Sub-Division",size=12)
plt.grid(axis="x",linestyle="-.")
plt.show()
```



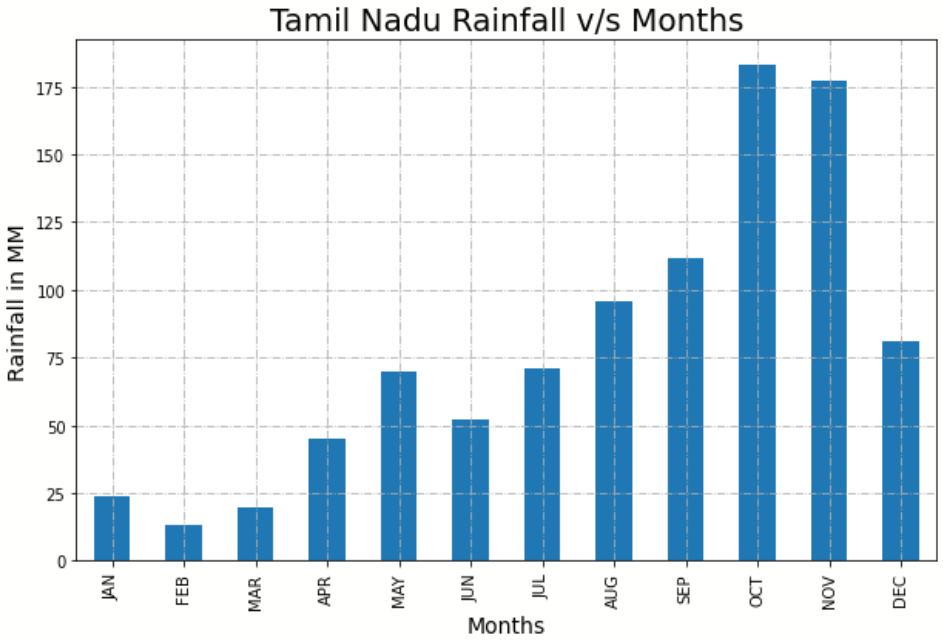
```
data[['SUBDIVISION', 'Jan-Feb', 'Mar-May',
       'Jun-Sep', 'Oct-
Dec']].groupby("SUBDIVISION").sum().plot(kind="barh",stacked=True,figsize=
(16,8))
plt.xlabel("Rainfall in MM",size=12)
plt.ylabel("Sub-Division",size=12)
plt.grid(axis="x",linestyle="-.")
plt.show()
```



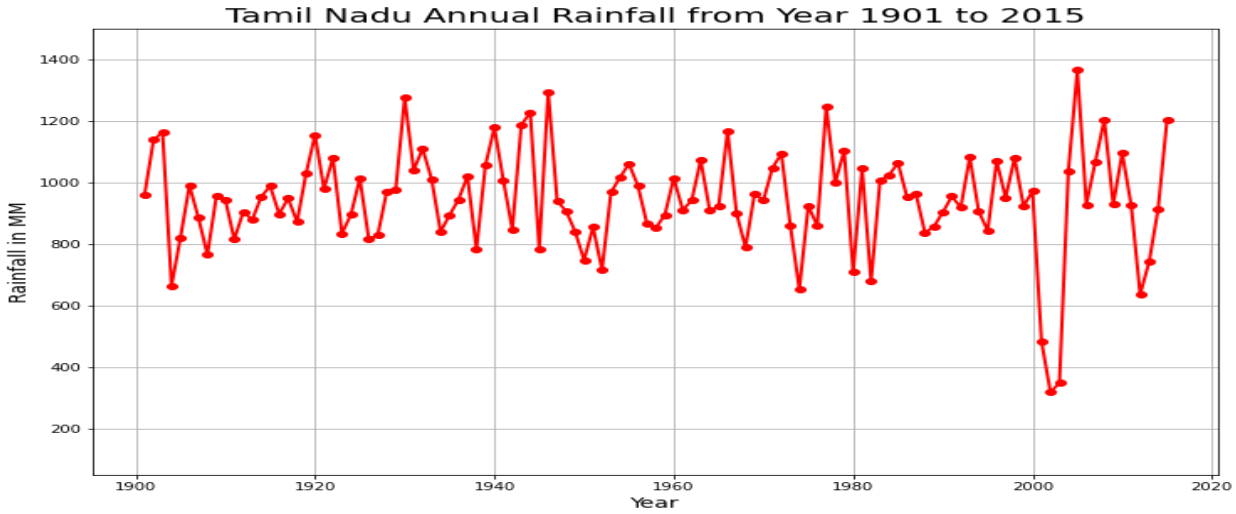

```
# Analysis of rainfall data of tamil nadu
TN = data.loc[((data['SUBDIVISION'] == 'TAMIL NADU'))]
TN.head(4)
```

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
3427	TAMIL NADU	1901	24.5	39.1	21.7	36.0	74.0	41.8	49.3	67.9	191.1	122.3	212.3	80.4	960.3	63.6	131.6	350.1	415.0
3428	TAMIL NADU	1902	67.2	9.8	25.1	21.9	84.7	39.3	55.1	113.8	98.6	282.2	174.9	165.8	1138.2	77.0	131.7	306.7	622.9
3429	TAMIL NADU	1903	19.3	7.8	1.7	18.2	128.5	58.5	72.6	115.0	210.4	128.1	200.5	203.2	1163.9	27.1	148.4	456.5	531.9
3430	TAMIL NADU	1904	35.2	0.1	0.7	19.5	121.9	34.9	89.0	40.4	85.7	163.2	23.6	49.1	663.1	35.3	142.1	249.9	235.8

```
plt.figure(figsize=(10,6))
TN[['JAN', 'FEB', 'MAR', 'APR','MAY', 'JUN','JUL','AUG', 'SEP', 'OCT','NOV
','DEC']].mean().plot(kind="bar",width=0.5,linewidth=2)
plt.title("Tamil Nadu Rainfall v/s Months",size=20)
plt.xlabel("Months",size=14)
plt.ylabel("Rainfall in MM",size=14)
plt.grid(axis="both",linestyle="-.")
plt.show()
```



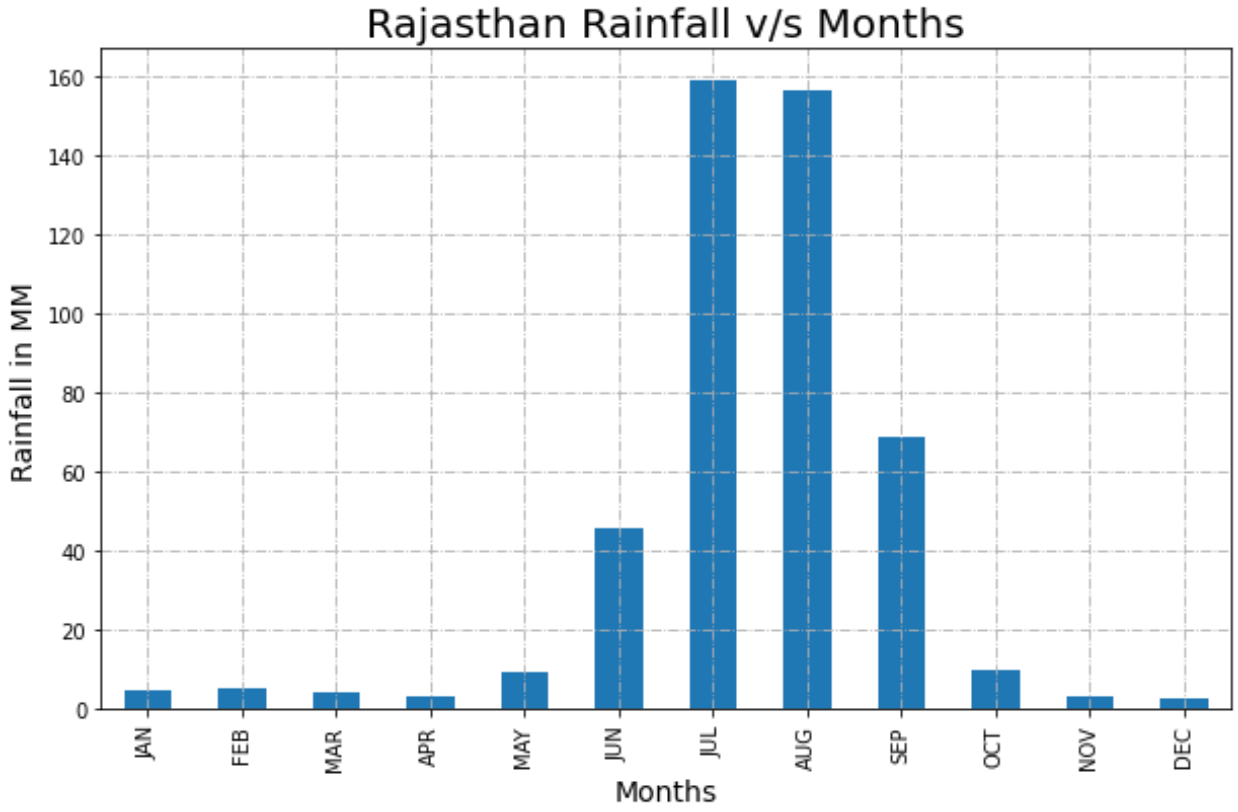

```
TN.groupby("YEAR").sum()['ANNUAL'].plot(ylim=(50,1500),color='r',marker='o
',linestyle='-',linewidth=2,figsize=(12,8));
plt.xlabel('Year',size=14)
plt.ylabel('Rainfall in MM',size=14)
plt.title('Tamil Nadu Annual Rainfall from Year 1901 to 2015',size=20)
plt.grid()
plt.show()
```



```
# analysis of rainfall data of rajasthan
Rajasthan = data.loc[((data['SUBDIVISION'] == 'WEST RAJASTHAN') | (data['SUBDIVISION'] == 'EAST RAJASTHAN'))]
Rajasthan.head()
```

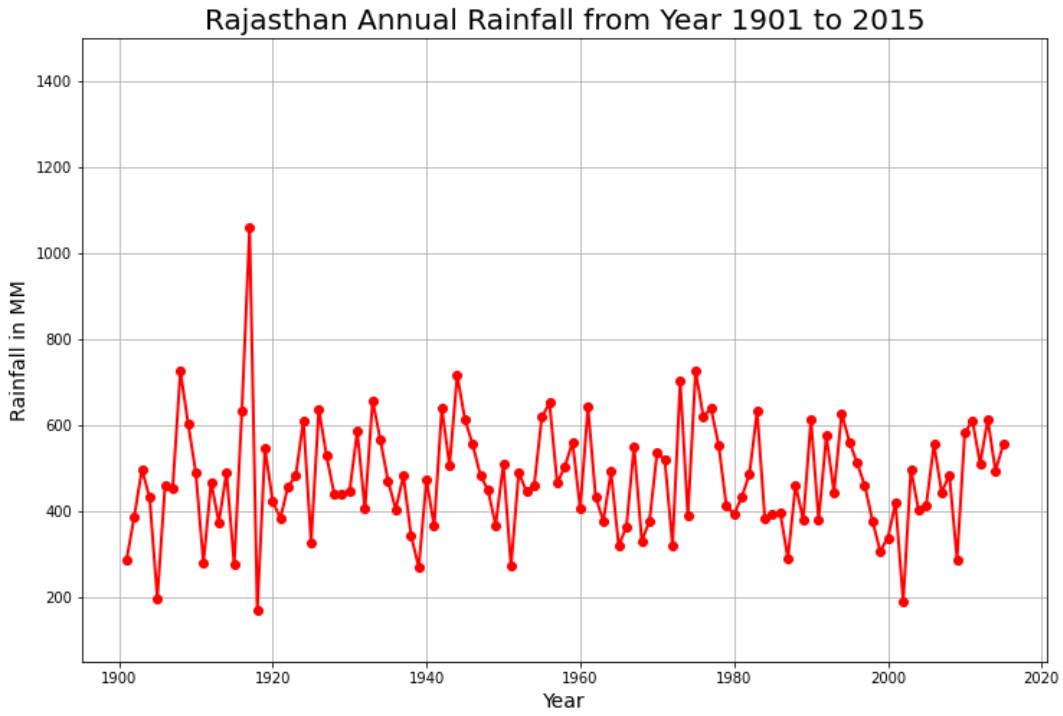
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
1817	WEST RAJASTHAN	1901	6.7	0.0	1.1	0.0	6.1	3.0	79.0	59.2	1.0	2.1	0.0	0.6	158.9	6.7	7.2	142.2	2.7
1818	WEST RAJASTHAN	1902	0.0	0.0	0.0	0.5	4.0	49.1	27.0	71.3	41.8	1.8	0.0	0.0	195.6	0.0	4.5	189.2	1.8
1819	WEST RAJASTHAN	1903	1.7	1.3	5.5	0.0	4.2	2.7	154.8	87.1	49.3	0.1	0.0	0.5	307.0	3.0	9.7	293.8	0.5
1820	WEST RAJASTHAN	1904	3.8	2.9	16.3	0.7	11.4	14.6	39.8	45.6	21.4	1.4	2.9	7.1	167.9	6.6	28.5	121.4	11.4
1821	WEST RAJASTHAN	1905	6.3	4.8	0.7	1.3	0.3	4.9	30.1	0.6	64.5	0.0	0.0	0.9	114.4	11.0	2.4	100.1	0.9

```
plt.figure(figsize=(10,6))
Rajasthan[['JAN', 'FEB', 'MAR', 'APR','MAY', 'JUN','JUL','AUG', 'SEP', 'OCT','NOV','DEC']].mean().plot(kind="bar",width=0.5,linewidth=2)
plt.title("Rajasthan Rainfall v/s Months",size=20)
plt.xlabel("Months",size=14)
plt.ylabel("Rainfall in MM",size=14)
plt.grid(axis="both",linestyle="-.")
plt.show()
```

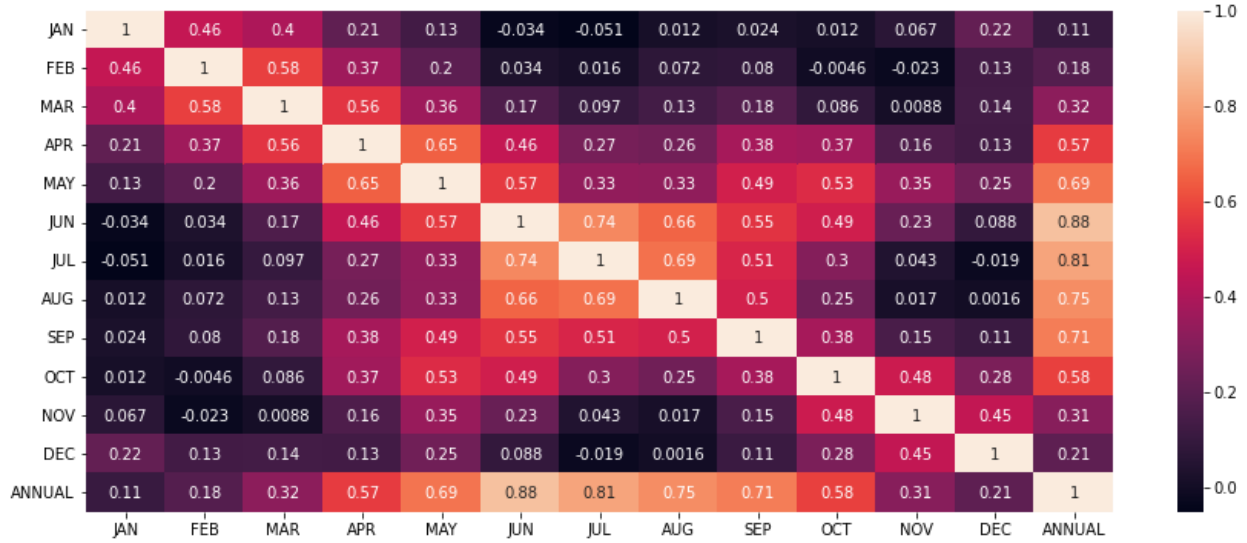



```
Rajasthan.groupby("YEAR").mean()['ANNUAL'].plot(ylim=(50,1500),color='r',marker='o',linestyle='-',linewidth=2,figsize=(12,8));
plt.xlabel('Year',size=14)
plt.ylabel('Rainfall in MM',size=14)
plt.title('Rajasthan Annual Rainfall from Year 1901 to 2015',size=20)
plt.grid()
```

```
plt.show()
```




```
# correlation b/w each numeric attribute
plt.figure(figsize=(15,6))
sns.heatmap(data[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC','ANNUAL']].corr(),annot=True)
plt.show()
```



Modelling

```
data["SUBDIVISION"].nunique()
36
```

```
group = data.groupby('SUBDIVISION')['YEAR','JAN','FEB','MAR','APR','MAY','
JUN','JUL','AUG','SEP','OCT','NOV','DEC']
data=group.get_group(('TAMIL NADU'))
data.head
```

[ ]

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
3427	1901	24.5	39.1	21.7	36.0	74.0	41.8	49.3	67.9	191.1	122.3	212.3	80.4
3428	1902	67.2	9.8	25.1	21.9	84.7	39.3	55.1	113.8	98.6	282.2	174.9	165.8
3429	1903	19.3	7.8	1.7	18.2	128.5	58.5	72.6	115.0	210.4	128.1	200.5	203.2
3430	1904	35.2	0.1	0.7	19.5	121.9	34.9	89.0	40.4	85.7	163.2	23.6	49.1
3431	1905	6.5	7.5	17.2	64.8	83.7	49.8	39.0	101.8	73.5	250.4	123.7	3.2

```
df=data.melt(['YEAR']).reset_index()
df.head()
```

	index	YEAR	variable	value
0	0	1901	JAN	24.5
1	1	1902	JAN	67.2
2	2	1903	JAN	19.3
3	3	1904	JAN	35.2
4	4	1905	JAN	6.5

```
[ ] df= df[['YEAR','variable','value']].reset_index().sort_values(by=['YEAR','index'])
    df.head()
```

	index	YEAR	variable	value
0	0	1901	JAN	24.5
115	115	1901	FEB	39.1
230	230	1901	MAR	21.7
345	345	1901	APR	36.0
460	460	1901	MAY	74.0

```
df.YEAR.unique()
```

```
array([1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911,
       1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
       1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933,
       1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944,
       1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955,
       1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966,
       1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977,
       1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
       1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
       2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015])
```

```
[ ] df.columns=['Index','Year','Month','Avg_Rainfall']
```

```
df.head()
```

	Index	Year	Month	Avg_Rainfall
0	0	1901	JAN	24.5
115	115	1901	FEB	39.1
230	230	1901	MAR	21.7
345	345	1901	APR	36.0
460	460	1901	MAY	74.0

```
[ ] Month_map={'JAN':1,'FEB':2,'MAR' :3,'APR':4,'MAY':5,'JUN':6,'JUL':7,'AUG':8,'SEP':9,
    'OCT':10,'NOV':11,'DEC':12}
df['Month']=df['Month'].map(Month_map)
df.head(12)
```

	Index	Year	Month	Avg_Rainfall
0	0	1901	1	24.5
115	115	1901	2	39.1
230	230	1901	3	21.7
345	345	1901	4	36.0
460	460	1901	5	74.0

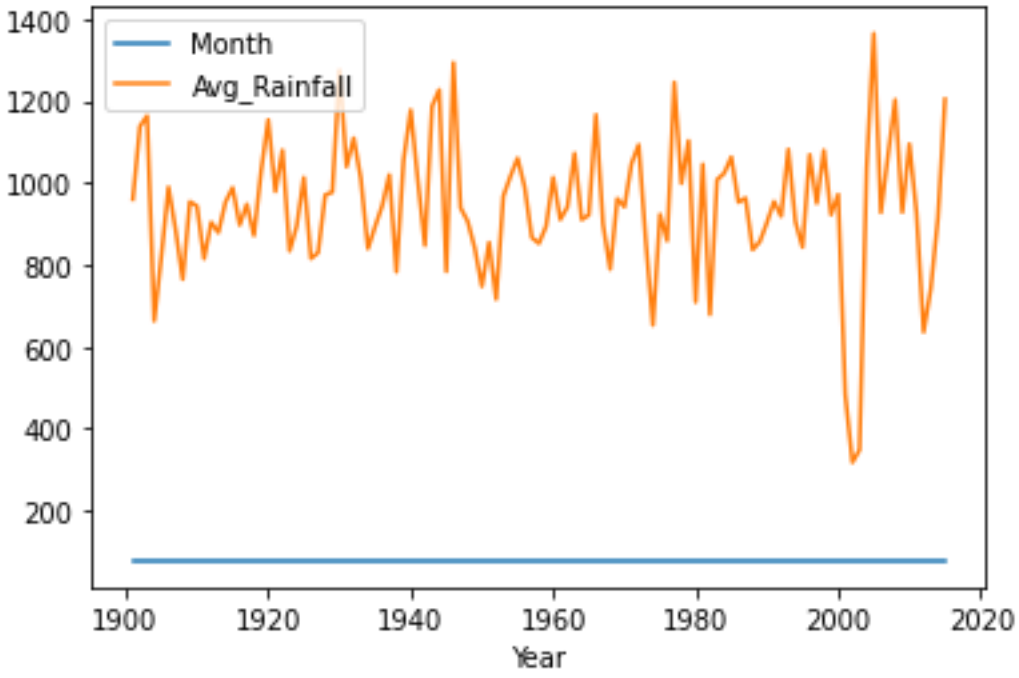
[ ]	**805**	805	1901	8	67.9
	920	920	1901	9	191.1
	1035	1035	1901	10	122.3
	1150	1150	1901	11	212.3
	1265	1265	1901	12	80.4

```
[ ] df.drop(columns="Index",inplace=True)
```

```
df.head(2)
```

	Year	Month	Avg_Rainfall
0	1901	1	24.5
115	1901	2	39.1

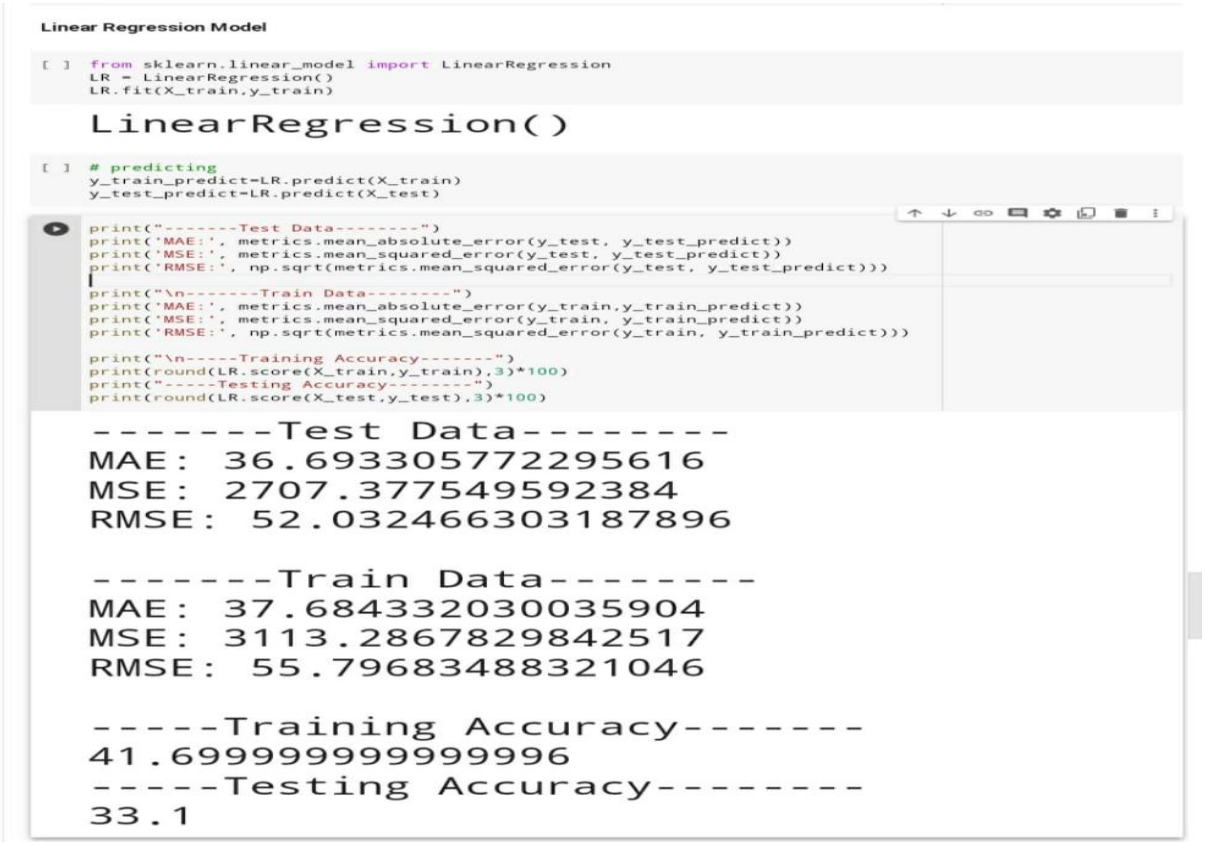
```
[ ] df.groupby("Year").sum().plot()
plt.show()
```




```
X=np.asanyarray(df[['Year','Month']]).astype('int')
y=np.asanyarray(df['Avg_Rainfall']).astype('int')
print(X.shape)
```

```
print(y.shape)
(1380, 2)
(1380,)

# splitting the dataset into training and testing
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r
andom_state=10)
```

Linear Regression Model



Lasso Model

Lasso Model

```
from sklearn import metrics
print("-------Test Data--------")
print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

print("\n-------Train Data--------")
print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))

print("\n-----Training Accuracy-------")
print(round(lasso.score(X_train,y_train),3)*100)
print("-----Testing Accuracy--------")
print(round(lasso.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 41.774633175550605
MSE: 3011.4820490350985
RMSE: 54.87697193755408

-------Train Data--------
MAE: 46.667686894462854
MSE: 3948.760899348929
RMSE: 62.839166921188

-----Training Accuracy-------
26.1
-----Testing Accuracy--------
25.6
```

Ridge Model

Ridge Model

```
[ ] from sklearn.linear_model import Ridge
    from sklearn.model_selection import GridSearchCV

    ridge=Ridge()
    parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
    ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
    ridge_regressor.fit(X_train,y_train)

    print(ridge_regressor.best_params_)
    print(ridge_regressor.best_score_)
```

```
{'alpha': 1e-15}
-3139.0798658992653
```

```
[ ] print("Best Parameter for Ridge:",ridge_regressor.best_estimator_)
```

```
Best Parameter for Ridge: Ridge(alpha=1e-
```

```
[ ] ridge=Ridge(alpha=100.0)

    # fit into the object
    ridge.fit(X_train,y_train)
```

```
Ridge(alpha=100.0)
```

```
[ ] # predicting the train and test values
    y_train_predict=ridge.predict(X_train)
    y_test_predict=ridge.predict(X_test)
```

```
[ ] from sklearn import metrics
    print("-------Test Data--------")
    print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
    print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

    print("\n-------Train Data--------")
    print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
    print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))

    print("\n-----Training Accuracy-------")
    print(round(ridge.score(X_train,y_train),3)*100)
    print("-----Testing Accuracy--------")
    print(round(ridge.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 36.694264997117806
MSE: 2700.404122847211
RMSE: 51.96541275547815

-------Train Data--------
MAE: 37.71478463865123
MSE: 3113.4499194422324
RMSE: 55.798296743200254

-----Training Accuracy-------
41.699999999999996
-----Testing Accuracy--------
33.300000000000004
```

SVM Model

Svm Model

```
[ ] from sklearn import preprocessing
    from sklearn import svm

    svm_regr = svm.SVC(kernel='rbf')
    svm_regr.fit(X_train, y_train)
```

```
SVC()
```

```
[ ] y_test_predict = svm_regr.predict(X_test)
    y_train_predict = svm_regr.predict(X_train)
```

SVM Model

```
[ ] from sklearn import metrics
    print("-------Test Data--------")
    print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
    print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

    print("\n-------Train Data--------")
    print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
    print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))


    print("\n-----Training Accuracy-------")
    print(round(svm_regr.score(X_train,y_train),3)*100)
    print("-----Testing Accuracy--------")
    print(round(svm_regr.score(X_test,y_test),3)*100)
```

```
-------Test Data--------
MAE: 76.73671497584542
MSE: 9936.306763285023
RMSE: 99.68102509146374

-------Train Data--------
MAE: 78.82815734989649
MSE: 11555.623188405798
RMSE: 107.4970845577023

-----Training Accuracy-------
3.5000000000000004
-----Testing Accuracy--------
1.7000000000000002
```

Random Forest Model

Random Forest Model

```
[ ] from sklearn.ensemble import RandomForestRegressor
    random_forest_model = RandomForestRegressor(max_depth=100, max_features='sqrt', min_samples_leaf=4,
                          min_samples_split=10, n_estimators=800)
    random_forest_model.fit(X_train, y_train)
```

```
RandomForestRegressor(max_depth=100,
max_features='sqrt', min_samples_leaf=4,

min_samples_split=10, n_estimators=800)
```

```
[ ] y_train_predict=random_forest_model.predict(X_train)
    y_test_predict=random_forest_model.predict(X_test)
```

```
[ ] print("-------Test Data--------")
    print('MAE:', metrics.mean_absolute_error(y_test, y_test_predict))
    print('MSE:', metrics.mean_squared_error(y_test, y_test_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_predict)))

    print("\n-------Train Data--------")
    print('MAE:', metrics.mean_absolute_error(y_train,y_train_predict))
    print('MSE:', metrics.mean_squared_error(y_train, y_train_predict))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_train, y_train_predict)))
```

```
-------Test Data--------
MAE: 34.05103074139167
MSE: 2341.6039068475443
RMSE: 48.3901219966177

-------Train Data--------
MAE: 25.907838970434035
MSE: 1456.8105348382642
RMSE: 38.16818747122090 4
```

```
[ ]
    print("-----------Training Accuracy-----------")
    print(round(random_forest_model.score(X_train,y_train),3)*100)
    print("-----------Testing Accuracy-----------")
    print(round(random_forest_model.score(X_test,y_test),3)*100)
```

```
-----------Training Accuracy-----------
72.7
-----------Testing Accuracy-----------
42.199999999999996
```

```
predicted = random_forest_model.predict([[2016,11]])
predicted
array([153.88717274])
```

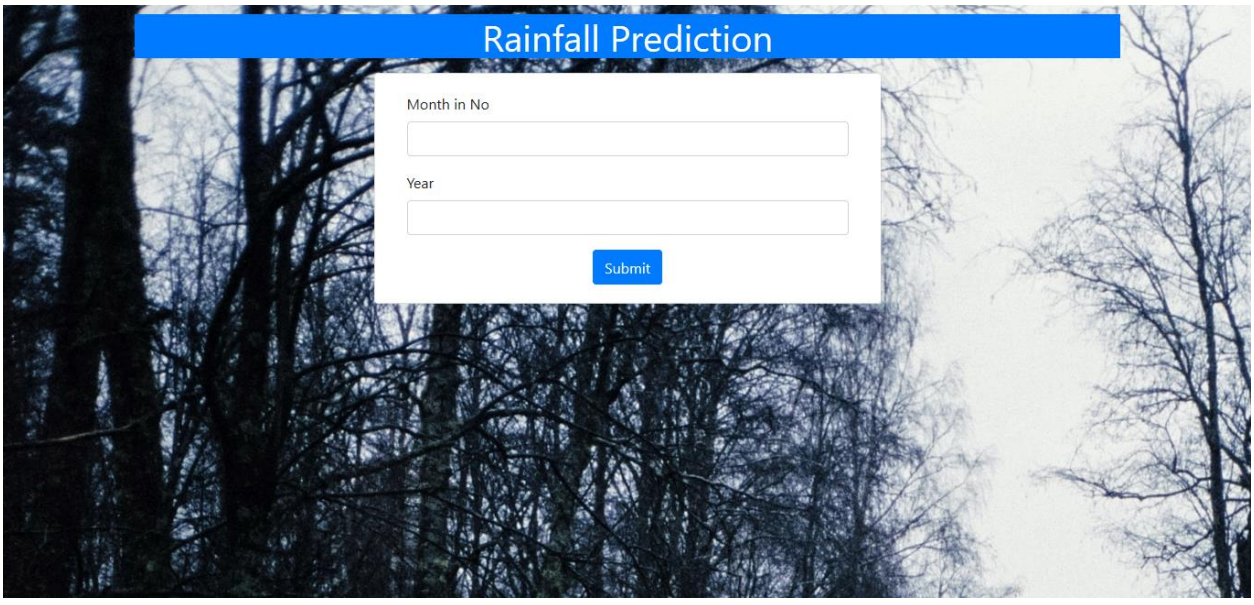
Save the Model

```
import pickle
pickle.dump(random_forest_model ,open("Rainfall_Prediction.pkl","wb"))
```

INDEX HTML CODE

```
<!DOCTYPE
html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <link rel="stylesheet"
            href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
            integrity="sha384-
            ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
            crossorigin="anonymous">
                <title>Rainfall-Prediction</title>
            </head>
            <body style="background-image: url('https://imgur.com/BAGXL6M.jpg')" >
             <div class="container">
                <h1 class="text-center m-3 badge-primary text-wrap">
                    Tamil Nadu Rainfall Prediction
                  </h1>
                    <div class="card container" style="width: 50%; ">

                        <div class="card-body">

                            <form action="/" method="post">
                                <div class="form-group">
                                  <label for="formGroupExampleInput1">Month in No</label>
                                  <input
                                    type="text"
                                    class="form-control"
                                    id="formGroupExampleInput1"
                                    name="Month"
                                    required
                                  />
                                </div>
```

```
                        <div class="form-group">
                          <label for="formGroupExampleInput2">Year</label>
                          <input
                            type="text"
                            class="form-control"
                            id="formGroupExampleInput2"
                            name="Year"
                            required
                          />
                        </div>
                        <center><button type="submit" class="btn btn-
primary">Submit</button></center>
                    </form>
                </div>
            </div>
        </div>


</body>
</html>
```

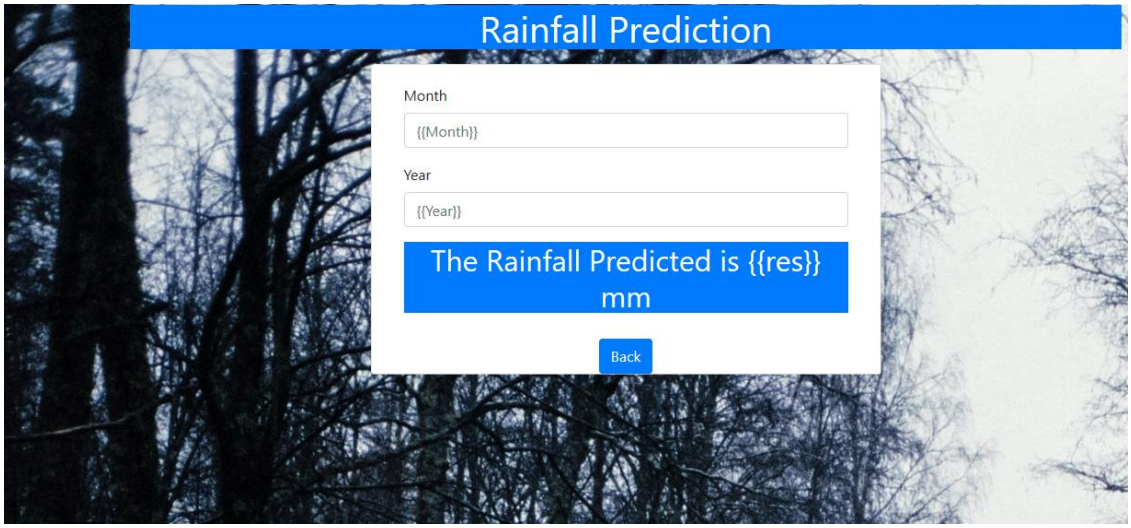


RESULT HTML CODE

```
<!DOCTY
PE
html>
            <html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <title>Rainfall-Prediction</title>
</head>
<body style="background-image: url('https://imgur.com/BAGXL6M.jpg')">
    <div class="container">
    <h1 class="text-center m-3 badge-primary text-wrap">Tamil Nadu rainfall
Prediction</h1>
        <div class="card container" style="width: 50%;">
            <div class="card-body">
                    <form action="/" method="post">
                        <div class="form-group">
                          <label for="formGroupExampleInput1">Month</label>
                          <input
                            type="text"
                            class="form-control"
                            id="formGroupExampleInput1"
                            name="Month"
                            placeholder="{{Month}}"
                            required
                          />
                        </div>
                        <div class="form-group">
                          <label for="formGroupExampleInput2">Year</label>
                          <input
                            type="text"
                            class="form-control"
                            id="formGroupExampleInput2"
                            name="Year"
                            placeholder="{{Year}}"
                            required
                          />
                        </div>
                        <h2 class="text-center  badge-primary text-wrap">The
Rainfall Predicted is {{res}} mm</h2>

                      </form>
                    </div>
```

```
                        <center><a href="/"><button type="submit" class="btn btn-
primary">Back</button></a></center>
                    </div>


    </div>


</body>
</html>
```

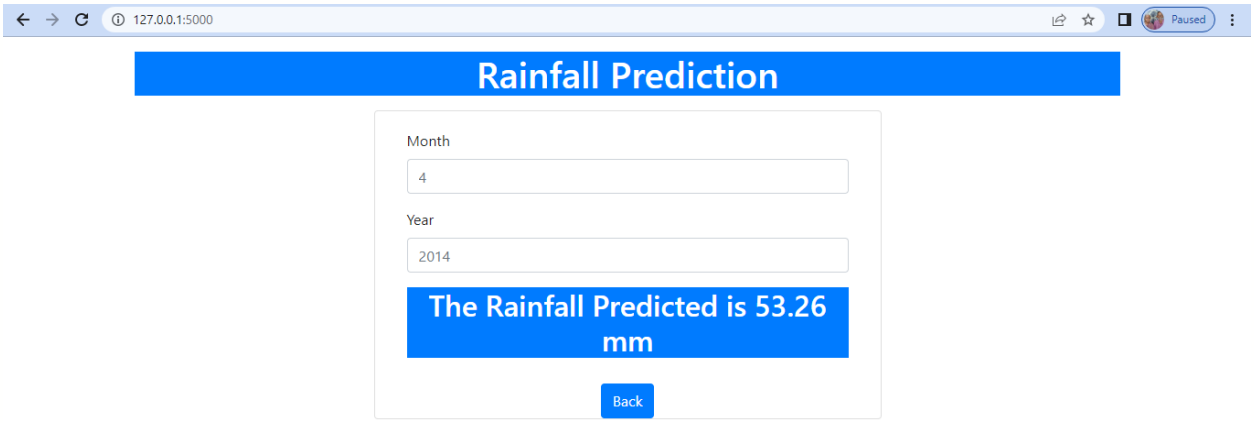


PYTHON SCRIPT

```
from flask import
render_template,Flask,request
                            import pickle

                            app=Flask(__name__)
                            file=open("model.pkl","rb")
                            random_Forest=pickle.load(file)
                            file.close()


                            @app.route("/", methods=["GET","POST"])
                            def home():
                                if request.method=="POST":
                                    myDict = request.form
                                    Month = int(myDict["Month"])
                                    Year = int(myDict["Year"])
```

```
        pred = [Year,Month]
        res=random_Forest.predict([pred])[0]
        res=round(res,2)
        return
render_template('result.html',Month=Month,Year=Year,res=res)
    return render_template('index.html')


if __name__ == "__main__":
    app.run(debug=True)
```

OUTPUT:



PROJECT DEMO LINK:

https://drive.google.com/file/d/11EAtbCM21kQzhBzK9loBhyZkiT9M4rAC/view?usp=share_link

GITHUB LINK

IBM-EPBL/IBM-Project-20731-1664169979