

# RAINFALL PREDICTION

A Major Project Report Submitted to the  
**Kamarajar Government Arts College, Surandai-627859**  
in Partial fulfillment of the Requirement of the award for the degree of  
**Master to Science in computer Science**

SUBMITTED BY

**SIVAPRIYA S**

**Reg.No:20211062506123**

*Under the Guidance of*  
**M.BAVANI M.C.A., M.Phil.,**



**DEPARTMENT OF COMPUTER SCIENCE  
KAMARAJAR GOVERNMENT ARTS COLLEGE  
SURANDAI - 627 859**

**APRIL – 2023**

# **DECLARATION**

I do hereby declare that the mini project work entitled “ **RAINFALL PREDICTION** ” is submitted to Manonmaniam Sundaranar University in partial fulfillment of the requirements for the award of the degree of Master of Science Computer Science, is a bonafide work done by me on my own effort. The data and reports submitted are true and correct to the best of my knowledge.

**SIGNATURE**

**(S.SIVAPRIYA)**

Place: Surandai

Date:

**KAMARAJAR GOVERNMENT ARTS COLLEGE  
SURANDAI - 627 859**

**DEPARTMENT OF COMPUTER SCIENCE**

**CERTIFICATE**

This is to certify that the project entitle “ **RAINFALL PREDICTION** ” is a bonafide record of the project work done by **S.SIVAPRIYA (20211062506123)**, a student of **Department of computer science, Kamarajar Government Arts College, Surandai** in the partial fulfilment of the requirements for the award of the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE** during the year 2022 – 2023.

**INTERNAL GUIDE**

**HEAD OF THE DEPARTMENT**

Submitted for the Viva–Voce examination held on\_\_\_\_\_.

**External Examiners:**

**1.**

**2.**

# ACKNOWLEDGEMENT

I thanks the National Aeronautics and Space Administration (NASA) / Goddard Space Flight Center website. I have taken from real time dataset in these website it is most important part of successfully completion my project helps more accurately prediction.

I express my sincere gratitude to **Mrs. V.UMAYAPARVATHI, M.C.A., M.Phil., M.Tech., SET.,** Head Department of Computer Science, Kamarajar Government Arts College, Surandai for giving me this opportunity to carry out this poject.

I would like to express my sincere thanks to **M.BAVANI M.C.A., M.Phil.,** my project guide for her consistent help and suggestions.

I am expressed in heartfelt thanks for **Dr.Arulmary M.Sc, M.Phill., Ph.D.,** her discuss the valuable knowledge and ideas to help the project more implemented.

I am extremely special thankful to my heartfelt brother. His very supported for me and he make me to hardwork person.

**S.SIVAPRIYA**

# **ABSTRACT**

## **RAINFALL PREDICTION**

Rainfall prediction is an essential task for many industries and government agencies. Accurate rainfall prediction can help farmers plan their planting and irrigation schedules, and government agencies can use it to make critical decisions related to flood control and disaster management.

The goal of this project is to develop to predict how the rainfall will changes in future linear regression algorithm, naïvebayes algorithm. The model will be trained on historical weather data, including information about date, Allsky\_UV index, temperature, Humidity, wind speed, and other relevant factor. The data is stored in the form of a .csv file.

In these project I have done by 3 types of prediction First, Current prediction User to enter the today Temperature, Humidity, Wind\_speed to display the today rainfall or not thus the current prediction is used to a Gaussian classification algorithm. Second, Future prediction – Used for Linear Regression algorithm thus the model was trained in the Tamilnadu district wise dataset the user can choose the any district of Tamilnadu and enter the date the model was predict the these location and the particular date is rainfall or not. Third, date range wise Top 10 rainfall date prediction- Used for linear regression algorithm to trained the Surandai, Tamilnadu, India dataset user to enter the starting date and ending date to predict these range of data rainfall or not and to display the Top 10 rainfall date in between user provided dates.

The project used a linear regression model to predict future how to behave rainfall rainfall based on historical rainfall data. The model was trained on a dataset that contained daily rainfall data from a weather station located in Tamilnadu district wise datas are trained in the model. . The dataset covered a period of 23 years, from 2000 to 2023. The model was evaluated . The evaluation results showed that the model was highly accurate.

# CONTENTS

<b>S.NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>SYSTEM ANALYSIS</b> 2.1 methodology 2.2 Existing System 2.3 Proposed System 2.4 Feasibility Analysis	<b>3</b>
<b>3</b>	<b>SYSTEM DESIGN</b>	<b>8</b>
<b>4</b>	<b>SYSTEM SPECIFICATION</b> 4.1 Software Requirement 4.2 Hardware Requirement 4.3 Software Description	<b>10</b>
<b>5</b>	<b>PROJECT DESCRIPTION</b> 5.1 Goal Objectives 5.2 Modules	<b>15</b>
<b>6</b>	<b>SYSTEM IMPLEMENTATION</b> 6.1 Sample Code 6.2 Screenshot	<b>18</b>
<b>7</b>	<b>CONCLUSION</b>	<b>48</b>
<b>8</b>	<b>FUTURE ENHANCEMENT</b>	<b>49</b>
<b>9</b>	<b>BIBLIOGRAPHY</b>	<b>50</b>

# **INTRODUCTION**

# INTRODUCTION

Rainfall is a crucial factor for agriculture, water management, and many other sectors. Inaccurate predictions of rainfall can lead to significant losses in agricultural production, disruptions in transportation, and even loss of life in extreme cases. Therefore, accurate and timely prediction of rainfall is essential for effective planning and decision-making.

Despite the advances in technology, predicting rainfall is still a challenging task. Rainfall prediction involves analyzing and interpreting large amounts of data, including atmospheric pressure, temperature, humidity, and wind speed. Traditional statistical methods are not efficient in handling such data sets, and often produce inaccurate predictions.

With the emergence of machine learning techniques, it is now possible to process large volumes of data and make accurate rainfall predictions. Machine learning algorithms can analyze and learn from historical data to identify patterns and trends that can be used to predict future rainfall.

The aim of this project is to develop a rainfall prediction model that uses machine learning algorithms to analyze historical data and predict rainfall accurately. The model will be based on linear regression, which is a simple yet powerful technique that can effectively model relationships between variables. The rainfall prediction model will use daily historical rainfall data and other weather variables to predict whether it will rain on a given day.

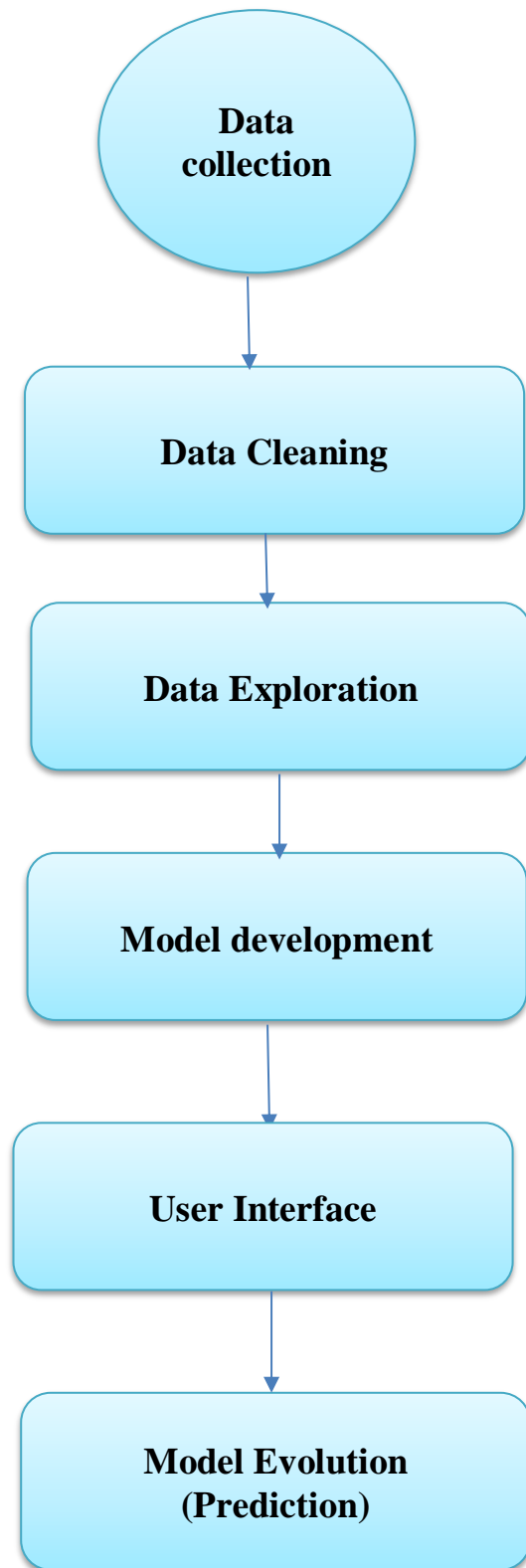
The model will be implemented using Python, and the dataset will be obtained from the National Aeronautics and Space Administration (NASA) / Goddard Space Flight Center. The performance of the model will be evaluated using various metrics, including accuracy, precision, recall. The results of the model will be presented visually using graphs and charts.



# **SYSTEM ANALYSIS**

# SYSTEM ANALYSIS

## 2.1 METHODOLOGY:



➤ **Data Collection:**

- The dataset is taken from National Aeronautics and space Administration (NASA)/ Goddard Space Flight Center.
- The data for this project was collected from a CSV file containing the overall district data of Tamil Nadu. The dataset has daily rainfall data from Tamilnadu district wise data .
- Dataset used has a total of 8 parameters includes, Year, Month, Day, Location, UV\_index, Temperature, Humidity, Wind\_pressure.

➤ **Data Cleaning:** The collected data was preprocessed and cleaned to remove any missing or irrelevant data.

➤ **Data Exploration:** The cleaned data was explored to gain insights into the features, their correlation, and their impact on rainfall.

➤ **Model Development:** Linear regression was used to develop a model to predict the expected rainfall for a given location and date.

➤ **User Interface Development:** A user interface was developed using tkinter to take input from the user and display the predicted result.

➤ **Model Evaluation:** The model was evaluated based on the accuracy of its predictions and any improvements that can be made to enhance the model's accuracy.

## **2.2 EXISTING SYSTEM:**

The existing system for predicting rainfall involves manual methods such as observation of cloud movements and using traditional knowledge passed down through generations. This method is not accurate and often leads to incorrect predictions. Moreover, it is time-consuming and requires a lot of effort. These methods may not be able to provide accurate and up-to-date information about weather conditions for specific locations, which can lead to errors in weather forecasting.

## **2.3 PROPOSED SYSTEM**

The proposed system is a machine learning-based weather forecasting model that uses real-time data from weather stations and other sources to provide accurate and up-to-date information about weather conditions for specific locations. This system can help users make informed decisions about activities that may be affected by weather conditions, such as travel, outdoor events, and agriculture.

The proposed system can also provide personalized weather forecasts to users based on their location and preferences. For example, users can set alerts for specific weather conditions or receive recommendations for activities that are safe to do based on the weather forecast.

The proposed system has several advantages over the existing system, including higher accuracy, and accessibility of weather forecasting, making it easier for users to plan their activities and stay safe in changing weather conditions. Greater speed, and the ability to analyze large datasets. It also reduces the need for manual observation and data entry, which can be time-consuming and error-prone. The proposed system can be used by weather agencies, farmers, and other industries that rely on accurate rainfall predictions to make informed decisions.

## 2.4 FEASIBILITY ANALYSIS

**Technical feasibility:** The project seems technically feasible as it uses well-established machine learning algorithms and libraries, such as scikit-learn, to make predictions based on weather data. The code also uses common data analysis tools, such as pandas and numpy, to manipulate and analyze the data.

**Economic feasibility:** The project requires some initial investment in terms of hardware and software, such as a computer, programming tools, and a dataset. However, these costs are relatively low compared to other types of projects, and the potential benefits, such as improved accuracy in predicting rainfall, could outweigh the costs.

**Operational feasibility:** The project is relatively easy to operate and maintain, as it only requires the user to input weather data and receive a prediction for rainfall. The code is also well-documented, making it easy for others to understand and modify as needed.

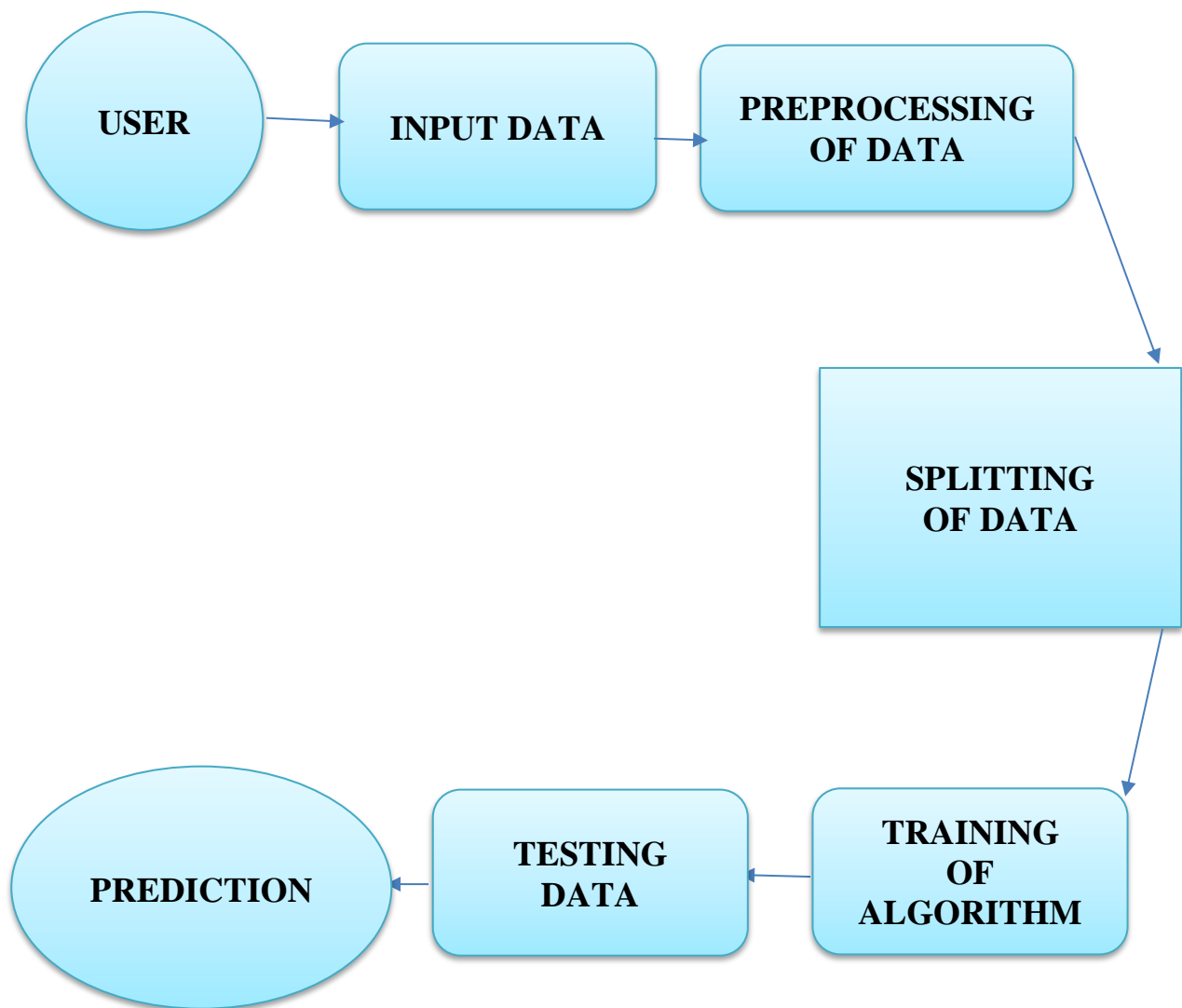
## 2.5 PROBLEM STATEMENT

Inaccurate Prediction of Rainfall Leading to Inappropriate Government Decisions.

The current problem is that the government has announced a school closure due to rainfall, but there is no rainfall in the area. This can be attributed to the inaccurate prediction of rainfall. This inaccurate prediction can lead to inappropriate decisions by the government, such as announcing a school closure, which can cause inconvenience to the public. The need of the hour is to develop a system that can accurately predict rainfall and provide real-time information to the government, helping them make appropriate decisions.

# **SYSTEM DESIGN**

### **3. SYSTEM DESIGN**



# **SYSTEM SPECIFICATION**



## **4. 1 SOFTWARE REQUIREMENT**

### **Front-End**

- Google colab research.org
- vscode
- Python

### **Back-End**

- .CSV file(comma separated values file)

### **Operating System**

- Windows 8.1

## **4.2 HARDWARE REQUIREMENT**

Processor	:	Intel Core i3 2330M
Speed	:	2.33 GHz
RAM	:	2 GB
Hard Disk Drive	:	500 GB
Compact Disk Drive	:	700 MB
Monitor	:	18.5” Color LED Monitor

## 4.3 SOFTWARE DESCRIPTION

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast.

Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

## Advantages of Python

### 1. Easy to Read, Learn and Write

Python is a **high-level programming language** that has English-like syntax.

This makes it easier to read and understand the code. Python is really easy to **pick up** and **learn**, that is why a lot of people recommend Python to beginners. You need less lines of code to perform the same task as compared to other major languages like **C/C++** and **Java**.

### 2. Improved Productivity

Python is a very **productive language**. Due to the simplicity of Python, developers can focus on solving the problem. They don't need to spend too much time in understanding the **syntax** or **behavior** of the programming language. You write less code and get more things done.

### 3. Interpreted Language

Python is an interpreted language which means that Python directly **executes the code** line by line. In case of any error, it stops further execution and reports back the error which has occurred. Python shows only one error even if the program has multiple errors. This makes **debugging** easier.

### 4. Dynamically Typed

Python doesn't know the type of variable until we run the code. It automatically assigns the data type during **execution**. The programmer doesn't need to worry about declaring variables and their data types.

### 5. Free and Open-Source

Python comes under the **OSI approved** open-source license. This makes it **free** to **use** and **distribute**. You can download the source code, modify it and even distribute your version of Python. This is useful for organizations that want to modify some specific behavior and use their version for development.

### 6. Portability

In many languages like **C/C++**, you need to change your **code** to run the program on different platforms.

## **Tkinter Library**

Tkinter is a Python library for creating graphical user interfaces (GUI). It provides a set of tools and widgets to build desktop applications that allow users to interact with the computer through a graphical interface.

The library is built on top of the Tk GUI toolkit, which provides a set of platform-independent GUI components. Tkinter includes a wide variety of widgets such as buttons, labels, text boxes, radio buttons, check boxes, and more. These widgets can be used to build forms, menus, dialog boxes, and other graphical user interface elements.

Tkinter also provides a canvas widget, which allows you to draw and manipulate graphics on a canvas. The canvas can be used to create animations, interactive plots, and other graphical displays.

Tkinter is easy to learn and use, making it a popular choice for building desktop applications with Python. It is included with Python by default, so you don't need to install any additional libraries to use it. Tkinter also supports many third-party extensions, which can add additional functionality to your applications.

Overall, Tkinter is a powerful and flexible library for building desktop applications with Python. Its ease of use and wide range of widgets make it a popular choice for developers who want to create GUI applications quickly and efficiently.

# **PROJECT DESCRIPTION**

## **5.1 GOAL**

The goal of this project is to develop a rainfall prediction system that can accurately predict the likelihood of rainfall in a given area on a specific date. The system will take input from a user in the form of a date and location and use historical rainfall data to predict the probability of rainfall occurring on that day. The prediction will be based on a linear regression model that takes into account the trend of rainfall over time. The system will also provide the user with the top 10 rainfall days based on the predicted rainfall values. The main objective of this project is to provide an accurate and reliable rainfall prediction system that can be used by the government, schools, and other organizations to make informed decisions regarding school closures, event cancellations, and other activities that may be affected by rainfall. By developing this system, we aim to provide a valuable tool that can help mitigate the impact of rainfall-related problems and improve the safety and well-being of individuals in the affected areas.

## **5.2 OBJECTIVE**

To predict whether it will rain on a given day using historical rainfall data.

1. To provide accurate information to the government regarding the likelihood of rainfall on a particular day, so that they can make informed decisions about school closures.
2. To create a user-friendly interface that allows users to input a date range and receive a list of the top 10 days with the highest predicted rainfall.
3. To use linear regression to create the rainfall prediction model and continually update it with new data to improve its accuracy.
4. To enhance the model by integrating other weather-related variables, such as temperature and humidity, to make more accurate predictions.
5. To provide a cost-effective and reliable solution to help the government and schools make informed decisions during rainy seasons, reducing the negative impact of unexpected rainfall on school attendance and overall productivity.

## 5.3 MODULES

- **Data reading and processing module:** This module is responsible for reading the input data file and processing it to extract the relevant information, such as rainfall data for a particular date
- **Prediction module:** This module uses the processed data to train a linear regression model and predict the likelihood of rainfall for a given date.
- **User input module:** This module takes input from the user in the form of start and end dates for the prediction and generates a range of dates between them.
- **Output module:** This module prints the predicted rainfall for each date in the date range and adds the predicted values to a new column in the dataset. It also sorts the dataset by the predicted rainfall column in descending order and displays the top 10 rainfall days.

## PACKAGE DESCRIPTION

- **Pandas:** Pandas is a Python package used for data manipulation and analysis. It provides data structures for efficiently storing and manipulating large datasets, as well as tools for reading and writing data to various file formats.
- **Scikit-learn:** Scikit-learn is a Python package used for machine learning. It provides a range of algorithms for classification, regression, clustering, and dimensionality reduction, as well as tools for preprocessing data, evaluating model performance, and selecting optimal model parameters.
- **NumPy:** NumPy is a Python package used for scientific computing. It provides a range of functions for performing mathematical operations on large arrays and matrices, as well as tools for linear algebra, Fourier analysis, and random number generation.
- **Matplotlib:** Matplotlib is a Python package used for data visualization. It provides a range of functions for creating various types of plots, including line plots, scatter plots, bar plots, and histograms.

# **SYSTEM IMPLEMENTATION**



## 6.1 SAMPLE CODE

### Main.py

```
import tkinter as tk
from PIL import Image, ImageTk
import os

def open_current():
    os.system("python C:\\Users\\HP\\Desktop\\siva\\current_pred.py")

def open_future():
    os.system("python C:\\Users\\HP\\Desktop\\siva\\location_based.py")

def open_top10():
    os.system("python C:\\Users\\HP\\Desktop\\siva\\top_10.py")

def validate_login():
    username = username_entry.get()
    password = password_entry.get()

    if username == "siva" and password == "sivarmy":
        # Create the new window and destroy the login window
        login_window.destroy()
        main_window = tk.Tk()
        main_window.title("Main Menu")
        main_window.geometry("700x700")

        # Load and resize the background image
        bg_image = Image.open("./butterfly.png")
        bg_image = bg_image.resize((1500, 1000), Image.ANTIALIAS)
        bg_photo = ImageTk.PhotoImage(bg_image)

        # Set the background image
        bg_label = tk.Label(main_window, image=bg_photo)
        bg_label.image = bg_photo
        bg_label.place(x=0, y=0)

        # Add the buttons to the main window
        current_button = tk.Button(main_window, text="Current prediction",
command=open_current)
        current_button.place(x=50, y=50)
```

```

future_button = tk.Button(main_window, text="Future prediction",
command=open_future)
future_button.place(x=50, y=100)

top10_button = tk.Button(main_window, text="Top 10 prediction",
command=open_top10)
top10_button.place(x=50, y=150)

main_window.mainloop()

else:
    error_label.config(text="Invalid username or password.")

def start_loading():
    loading_label.place(x=150, y=150)

def stop_loading():
    loading_label.place_forget()

# Create the login window
login_window = tk.Tk()
login_window.title("Login")
login_window.geometry("300x250")

# Load and resize the logo image
logo_image = Image.open("./logo.png")
logo_image = logo_image.resize((500, 500), Image.ANTIALIAS)
logo_photo = ImageTk.PhotoImage(logo_image)

# Set the logo image
logo_label = tk.Label(login_window, image=logo_photo)
logo_label.image = logo_photo
logo_label.pack(pady=10)

# Load the loading GIF
loading_image = Image.open("./loading.gif")
loading_photo = ImageTk.PhotoImage(loading_image)

# Create the input widgets
username_label = tk.Label(login_window, text="Username:")
username_label.pack()
username_entry = tk.Entry(login_window)
username_entry.pack()

```

```

password_label = tk.Label(login_window, text="Password:")
password_label.pack()
password_entry = tk.Entry(login_window, show="*")
password_entry.pack()

# Create the output widget
error_label = tk.Label(login_window, fg="red", text="")
error_label.pack()

# Create the button to log in
login_button = tk.Button(login_window, text="Log In", command=lambda:
[start_loading(), validate_login(), stop_loading()])
login_button.pack(pady=10)

# Create the loading label
loading_label = tk.Label(login_window, image=loading_photo)

login_window.mainloop()

```

### **current\_pred.py**

```

import pandas as pd
from sklearn.naive_bayes import GaussianNB
import tkinter as tk
from sklearn.metrics import accuracy_score

# Load dataset
data = pd.read_csv("C:/Users/HP/Desktop/siva/anndataset.csv")
inputs = data.drop('rainfall', axis='columns')
target = data['rainfall']

# Create classifier
classifier = GaussianNB()
classifier.fit(inputs, target)

# Define function to get prediction and show alert message
def predict_rainfall():
    temp = float(temp_entry.get())
    humidity = float(humidity_entry.get())
    wind_speed = float(wind_speed_entry.get())
    test_data = [[temp, humidity, wind_speed]]
    prediction = classifier.predict(test_data)[0]

```

```

if prediction == 1:
    result_label.config(text="The probability of rainfall is 'yes'")
    alert_label.config(text="Rainfall expected today". Please carry an
umbrella.')
else:
    result_label.config(text="The probability of rainfall is 'no'")
    alert_label.config(text="NO Rainfall expected today")

# Calculate accuracy
acc = classifier.score(inputs, target)
accuracy_label.config(text=f"Accuracy: {acc:.2f}")

# Create GUI
window = tk.Tk()
window.title("Rainfall Prediction")
window.geometry("400x250")

# Create input fields and labels
temp_label = tk.Label(window, text="Temperature:")
temp_label.grid(row=0, column=0)
temp_entry = tk.Entry(window)
temp_entry.grid(row=0, column=1)

humidity_label = tk.Label(window, text="Humidity:")
humidity_label.grid(row=1, column=0)
humidity_entry = tk.Entry(window)
humidity_entry.grid(row=1, column=1)

wind_speed_label = tk.Label(window, text="Wind Speed:")
wind_speed_label.grid(row=2, column=0)
wind_speed_entry = tk.Entry(window)
wind_speed_entry.grid(row=2, column=1)

# Create button to predict rainfall
predict_button = tk.Button(window, text="Predict", command=predict_rainfall)
predict_button.grid(row=3, column=0)

# Create label to show result of prediction
result_label = tk.Label(window, text="")
result_label.grid(row=3, column=1)

# Create label to show alert message
alert_label = tk.Label(window, text="")

```

```

alert_label.grid(row=4, column=0, columnspan=2)

# Create label to show accuracy score
accuracy_label = tk.Label(window, text="")
accuracy_label.grid(row=5, column=0, columnspan=2)

window.mainloop()

```

### **location based.py (future)**

```

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import tkinter as tk

# Read the dataset
df = pd.read_csv("C:/Users/HP/Desktop/siva/overall district data.csv")

# Create a tkinter window
window = tk.Tk()
window.title("Rainfall Prediction")

# Create a label for location
location_label = tk.Label(window, text="Select your location:")
location_label.pack()

# Create a dropdown list for location
locations = ["Thiruchirappalli", "Tanjavur", "Tiruvarur", "Nagapattinam",
"Nilgris", "Erode", "Namakkal", "Perambalur", "Ariyalur", "Selam",
"Mayiladuthurai", "Cuddalore", "Kallakurichi", "Dharmapuri", "Viluppuram",
"Krishnagiri", "Tirupattur", "Tiruvannamalai", "Chengalpattu", "Vellore",
"Kanchipuram", "Chennai", "Tiruvallur", "Ranipet", "Tenkasi", "Kanyakumari",
"Thirunelveli", "Thoothukudi", "Virudhunagar", "Ramanathapuram", "Theni",
"Madurai", "Sivagangai", "Dindigul", "Pudukkottai", "Coimbatore", "Tiruppur",
"Karur"]

selected_location = tk.StringVar(window)
selected_location.set(locations[0]) # Set the default value of the dropdown list
location_dropdown = tk.OptionMenu(window, selected_location, *locations)
location_dropdown.pack()

# Create a label for date
date_label = tk.Label(window, text="Enter the date (dd/mm/yyyy):")

```

```
date_label.pack()
```

```
# Create an entry box for date
date_entry = tk.Entry(window)
date_entry.pack()
```

```
# Create a label for the output
output_label = tk.Label(window, text="")
output_label.pack()
```

```
# Define a function to predict the rainfall
def predict_rainfall():
```

```
    # Get the user inputs
    user_location = selected_location.get()
    user_input = date_entry.get()
    if not user_input:
        output_label.config(text="Please enter a date.")
        return
    day, month, year = map(int, user_input.split('/'))
```

```
    # Filter the dataset by the input location
    filtered_df = df[df['Location'] == user_location]
```

```
    # If there is no data for the input location, display an error message
    if len(filtered_df) == 0:
        output_label.config(text="No data found for the input location.")
        return
```

```
    # Filter the dataset by the input date
    mask = (filtered_df['day'] == day) & (filtered_df['month'] == month)
    filtered_df = filtered_df[mask]
```

```
    # If there is no data for the input date, display an error message
    if len(filtered_df) == 0:
        output_label.config(text="No data found for the input date.")
        return
```

```
    # Create a pandas Series with the rainfall data
    rainfall = pd.Series(filtered_df['Precipitation'])
    if len(rainfall) > 1:
        mean_rainfall = np.mean(rainfall)
```

```

    if mean_rainfall >= 0.5:
        output_label.config(text="The predicted rainfall for {} on {} is . Expected
rainfall".format(user_location, user_input ))
    else:
        output_label.config(text="The predicted rainfall for {} on {} is . Rainfall
not expected".format(user_location, user_input ))
    else:
        # Fit a linear regression model to predict the rainfall
        X = filtered_df[['Latitude', 'Longitude']]
        y = filtered_df['Precipitation']
        model = LinearRegression().fit(X, y)

# Predict the rainfall
predicted_rainfall = model.predict([[X.iloc[0,0], X.iloc[0,1]]])[0]
if predicted_rainfall >= 0.5:
    output_label.config(text="The predicted rainfall for {} on {} is . Expected
rainfall".format(user_location, user_input))
else:
    output_label.config(text="The predicted rainfall for {} on {} is . Rainfall
not expected".format(user_location, user_input ))

predict_button = tk.Button(window, text="Predict", command=predict_rainfall)
predict_button.pack()

window.mainloop()

#1/3/2023 - not expected rainfall

```

## **Top10.py**

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Read the dataset
df = pd.read_csv("C:/Users/HP/Desktop/siva/surandai_data.csv")
df

# Split the date column into day, month, and year

# Ask user for input date

"""sp= df['date'].str.split('/', expand=True)
df['day'] = sp[1].astype('int')
df['month'] = sp[0].astype('int')
df['year'] = sp[2].astype('int')"""

user_input = input("Enter your date dd/mm/yyyy format : ")
day, month, year = map(int, user_input.split('/'))

# Filter the dataset by the input date
mask = (df['day'] == day) & (df['month'] == month)
filtered_df = df[mask]

# If there is no data for the input date, exit the program
if len(filtered_df) == 0:
    print("No data found for the input date.")
    exit()

# Create a pandas Series with the rainfall data
rainfall = pd.Series(filtered_df['Precipitation'])

# If there is only one data point, exit the program
if len(rainfall) <= 1:
    print("Not enough data to make a prediction.")
    exit()
```



```

# Create a numpy array for the input data
X = np.array(range(len(rainfall))).reshape(-1, 1)
#print(X)

# Create a numpy array for the output data
y = np.array(rainfall)
#print(y)

# Create a linear regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Predict the output for the next data point
next_input = np.array([[len(rainfall)]])
next_output = model.predict(next_input)

# Check if the predicted output is greater than or equal to 0.5
if next_output >= 0.5:
    final_output = 1
else:
    final_output = 0

# Print the final output
#print(f"The final output of the rainfall is {final_output}.")
if final_output == 1:
    print(user_input, "This date is "expected rainfall" ")
else:
    print(user_input, ' This date "does Not expecting rainfall" ')

# Visualize the data using Matplotlib
import matplotlib.pyplot as plt

plt.bar(user_input, next_output)
print(next_output)
plt.plot(X, y, color='red')

plt.xlabel('user_input')
plt.ylabel('predict rainfall')
plt.show()

```

```

import matplotlib.pyplot as plt

# Create a scatter plot of X and y
plt.scatter(X[:, 0], y)

#print(X[:, 0])

# Add a line of best fit to the scatter plot
m, b = np.polyfit(X[:, 0], y, 1)

""" function from NumPy library is used to calculate the slope and y-intercept
The 1 argument specifies the degree of the polynomial fit which is a straight
line. """

plt.plot(X[:, 0], m*X[:, 0] + b) # m-slope, b-intercept
#print(m*X[:, 0] + b)


# Set the title and labels for the plot
plt.title('Rainfall Data')
plt.xlabel('Time (days)')
plt.ylabel('Rainfall (mm)')

# Show the plot
plt.show()

import matplotlib.pyplot as plt

# Plot the data
plt.plot(filtered_df['year'], filtered_df['Precipitation'])


# Set the title and axis labels
plt.title('Year Wise Percipitation')
plt.xlabel('Year')
plt.ylabel('Precipitation')

# Display the plot
plt.show()

```


## 6.2 SCREEN SHOTS

### Login form



Username:

Password:

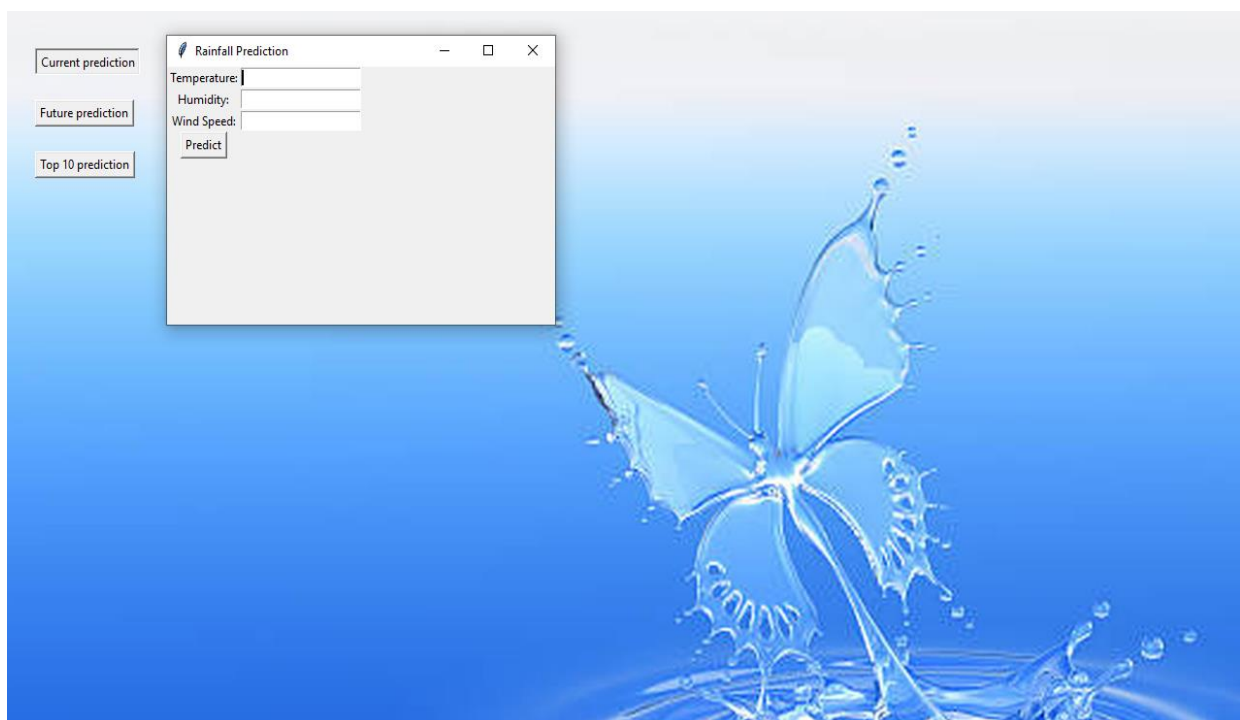


Username:

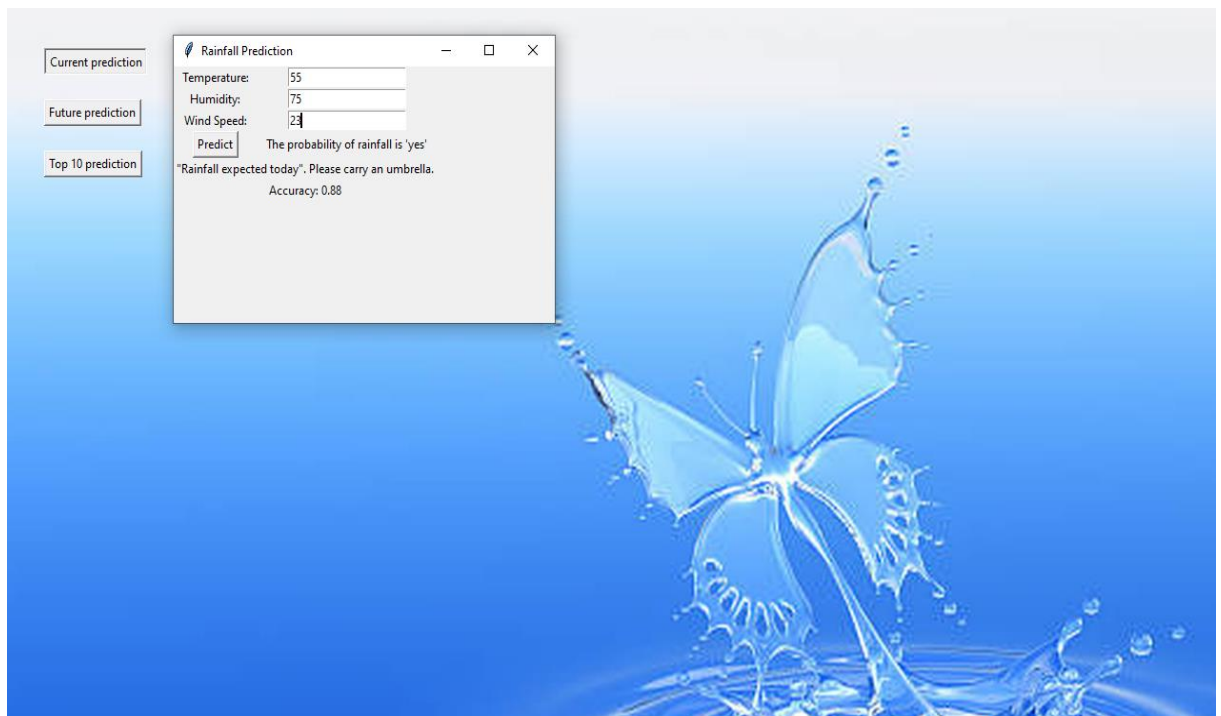
Password:



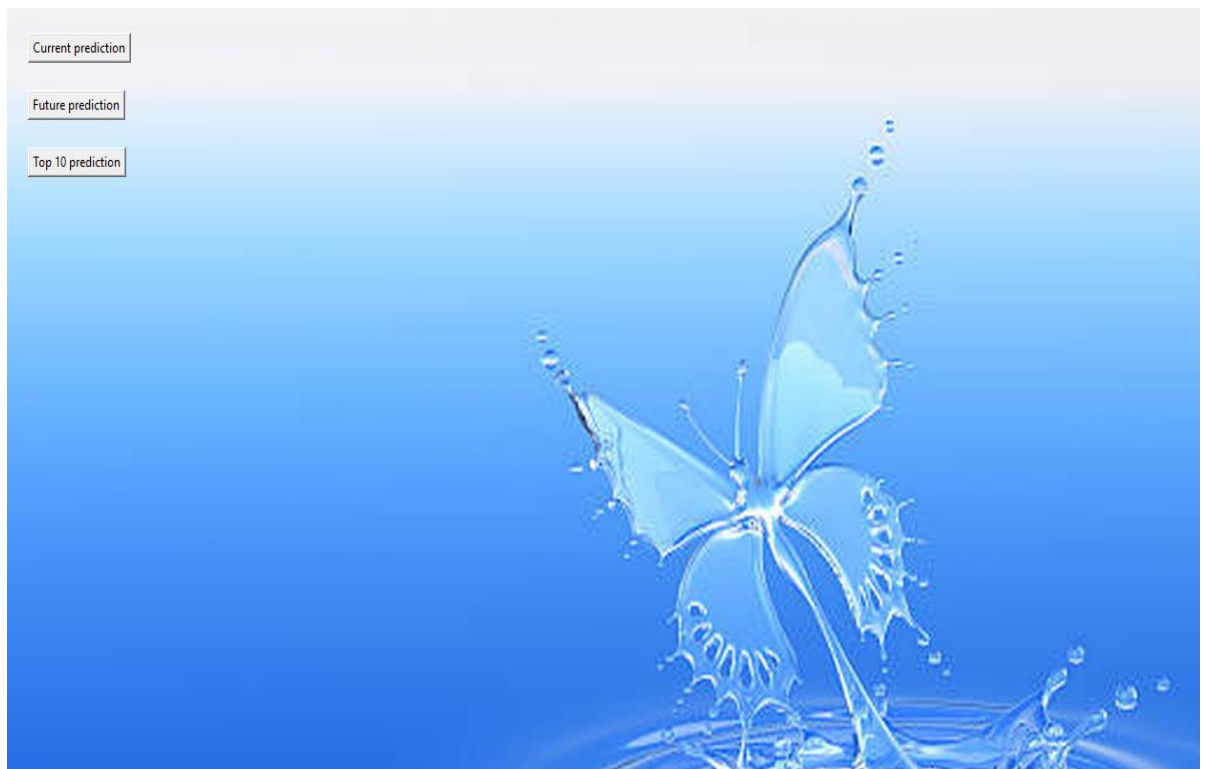
➤ If the user click the current prediction button then open the next page,



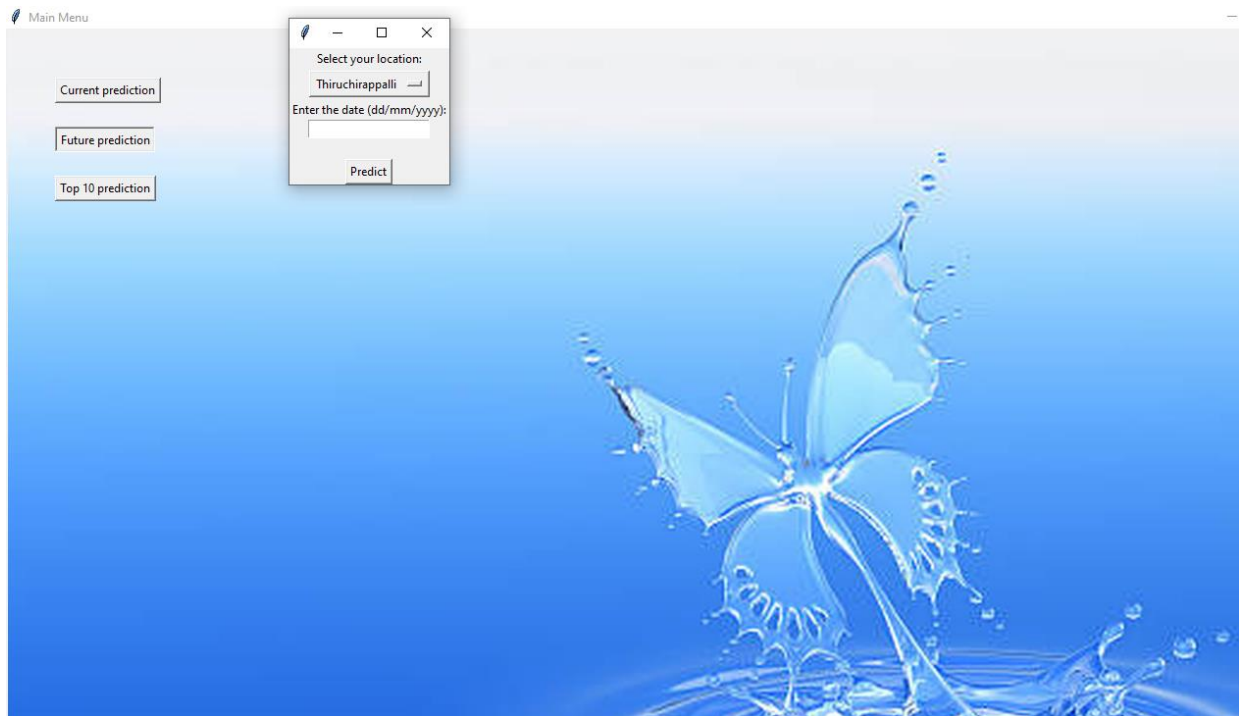
- User to enter the current Temperature, Humidity, Wind\_speed then click the predict button. Display the result,



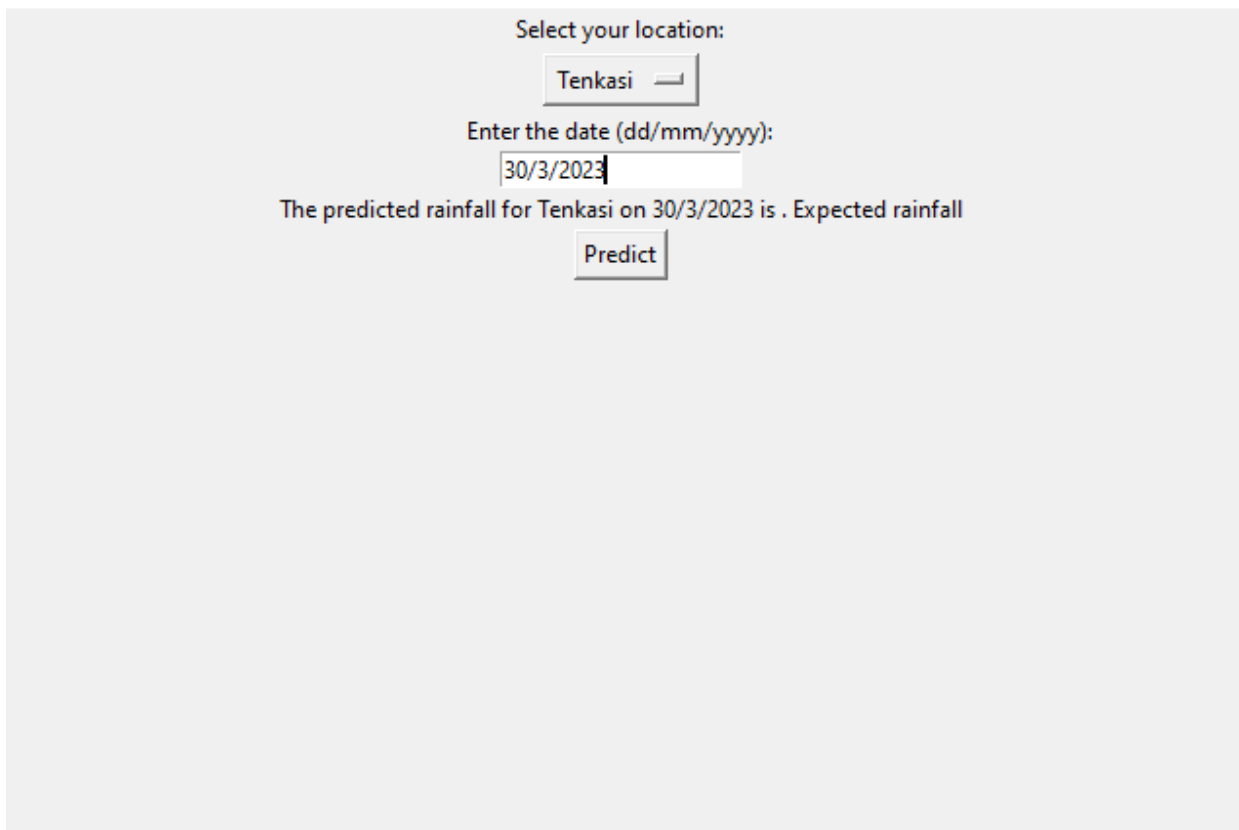
- If the user click the future button to open the location\_based.py file



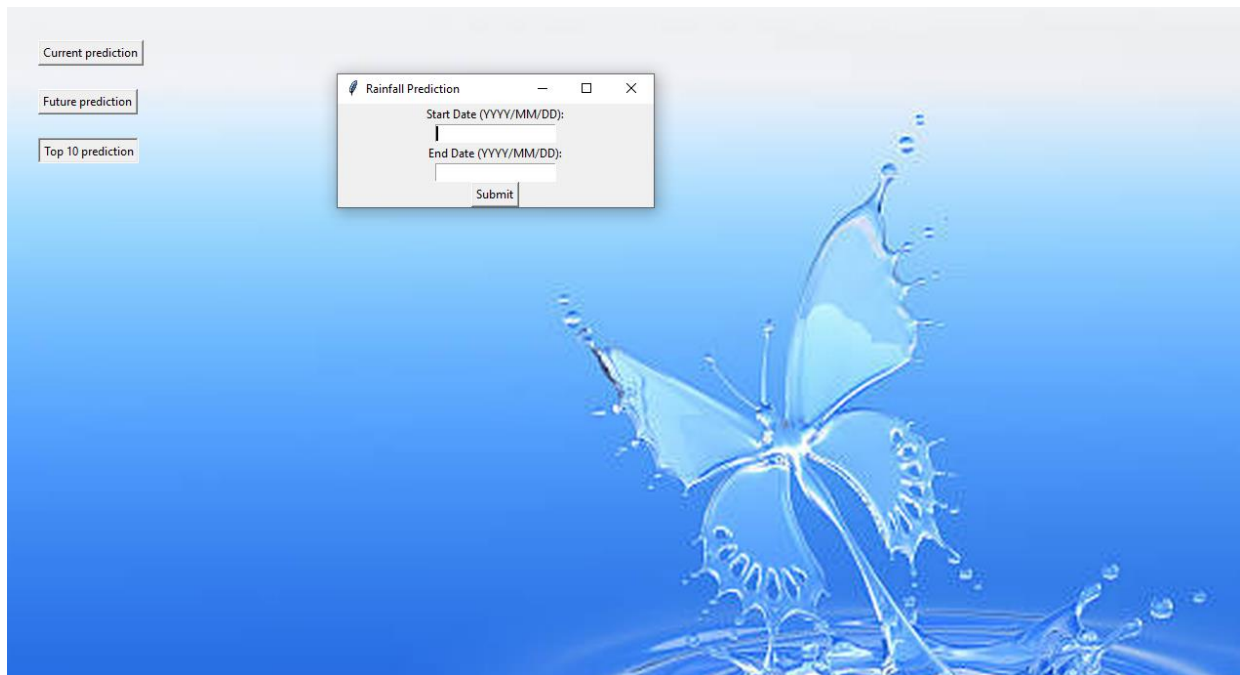
- User to select the any district and enter in date then click the predict button to display the next page



- After click the predict button



- If the user to click the top 10 button to run on the top10.py



- User to enter the starting date and ending date in the label and click the submit button

Start Date (YYYY/MM/DD):

End Date (YYYY/MM/DD):

- Next, to predict and display the top 10 rainfall date in the particular date range.

	predicted_rainfall
2023-05-11 00:00:00	10.200751
2023-05-14 00:00:00	8.836561
2023-05-15 00:00:00	8.518300
2023-05-25 00:00:00	6.237628
2023-05-19 00:00:00	5.266403
2023-05-08 00:00:00	5.174862
2023-05-13 00:00:00	5.039130
2023-05-16 00:00:00	4.795494
2023-05-12 00:00:00	4.786877
2023-05-09 00:00:00	4.636957

### **Location viz.py**

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Read the dataset
df = pd.read_csv("C:\\Users\\HP\\Desktop\\siva\\overall district data.csv")

# Ask user for input location
user_location = input("Enter your location: ")

# Filter the dataset by the input location
filtered_df = df[df['Location'] == user_location]

# If there is no data for the input location, exit the program
if len(filtered_df) == 0:
    print("No data found for the input location.")
    exit()

# Print the specific location data for the requested parameters
print(filtered_df[['day', 'month', 'year', 'Location', 'Temperature', 'Percipitation',
'Wind_pressure', 'Allsky_Surfce_UV_INDEX']])

user_input = input("Enter your date dd/mm/yyyy format : ")
day, month, year = map(int, user_input.split('/'))
```



```

# Filter the dataset by the input date
mask = (filtered_df['day'] == day) & (filtered_df['month'] == month)

filter = filtered_df[mask]

# If there is no data for the input date, exit the program
if len(filter) == 0:
    print("No data found for the input date.")
    exit()

# Create a pandas Series with the rainfall data
rainfall = pd.Series(filter['Percipitation'])
print(f"{filter['year']} - {rainfall}", end=' ')
# 2000-2022 year percipitation

# If there is only one data point, exit the program
if len(rainfall) <= 1:
    print("Not enough data to make a prediction.")
    exit()

#Create a numpy array for the input data
X = np.array(range(len(rainfall))).reshape(-1, 1)

# Create a numpy array for the output data
y = np.array(rainfall)

# Create a linear regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Predict the output for the next data point
next_input = np.array([[len(rainfall)]])
next_output = model.predict(next_input)

```

```
# Check if the predicted output is greater than or equal to 0.5
if next_output >= 0.5:
    final_output = 1
else:
    final_output = 0
```

```
# Print the final output
print(f"The final output of the rainfall is {final_output}.")
if final_output == 1:
    print("Today is rainfall")
else:
    print("Not expecting rainfall today")
```

```
import matplotlib.pyplot as plt
```

```
plt.bar(user_input,next_output)
print(next_output)
plt.plot(X,y , color='red')
```

```
plt.xlabel('user_input')
plt.ylabel('predict rainfall')
plt.show()
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6)) #width =10, height=6
plt.plot(filter['year'], rainfall)
plt.title("year wise rainfall")
plt.xlabel('Year')
plt.ylabel('Rainfall')
plt.show()
```

## screenshots

- user to enter the any district name in tamilnadu

```
PS C:\Users\HP\Desktop\siva> python location_vizual.py
Enter your location: Selam
  day  month  year  ... Percipitation  Wind_pressure  Allsky_Surfce_UV_INDEX
76419   1     1  2000  ...          0.07          97.57          -999.0
76420   2     1  2000  ...          0.01          97.55          -999.0
76421   3     1  2000  ...          0.00          97.46          -999.0
76422   4     1  2000  ...          0.00          97.45          -999.0
76423   5     1  2000  ...          0.08          97.43          -999.0
...    ...    ...    ...    ...          ...          ...          ...
84905  27     3  2023  ...          0.66          97.28          -999.0
84906  28     3  2023  ...          0.13          97.30          -999.0
84907  29     3  2023  ...          0.05          97.39          -999.0
84908  30     3  2023  ...          0.01          97.32          -999.0
84909  31     3  2023  ...          0.00          97.23          -999.0

[8491 rows x 8 columns]
```

- And to enter in date to filter the these date rainfall and predict these date rainfall or not

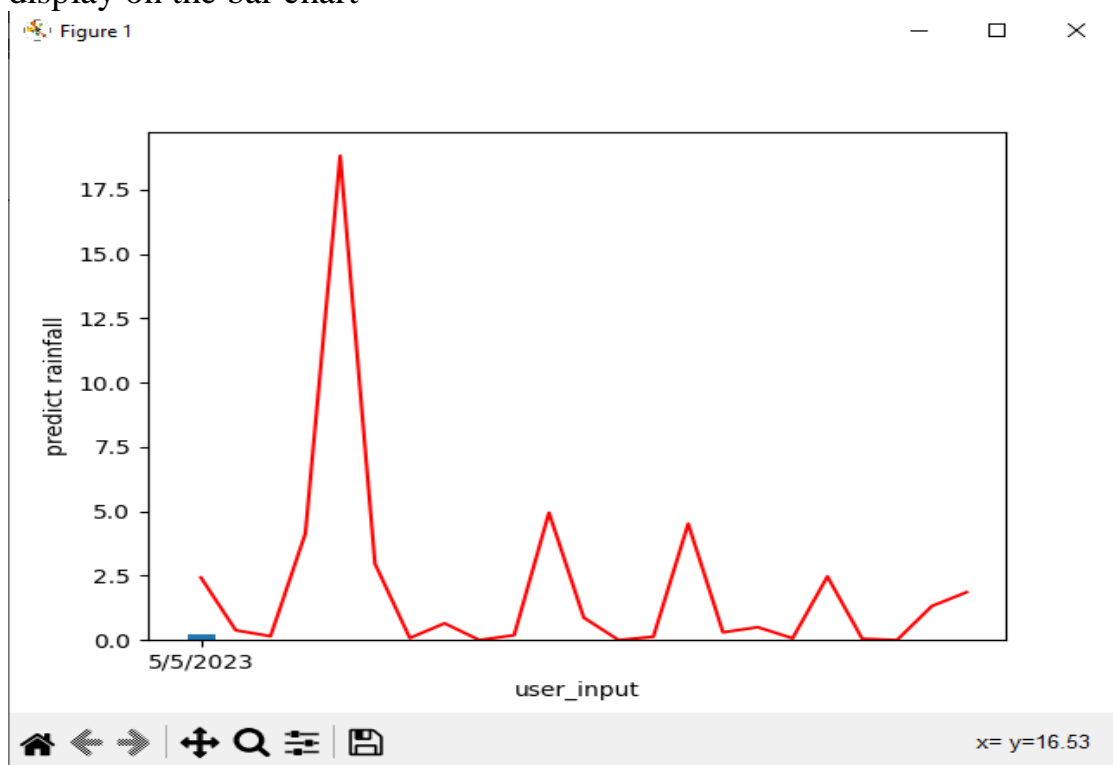
```
[8491 rows x 8 columns]
Enter your date dd/mm/yyyy format : 5/5/2023
76544      2000
76909      2001
77274      2002
77639      2003
78005      2004
78370      2005
78735      2006
79100      2007
79466      2008
79831      2009
80196      2010
80561      2011
80927      2012
81292      2013
81657      2014
82022      2015
82388      2016
82753      2017
83118      2018
83483      2019
83849      2020
84214      2021
84579      2022
Name: year, dtype: int64 - 76544      2.43
```

```

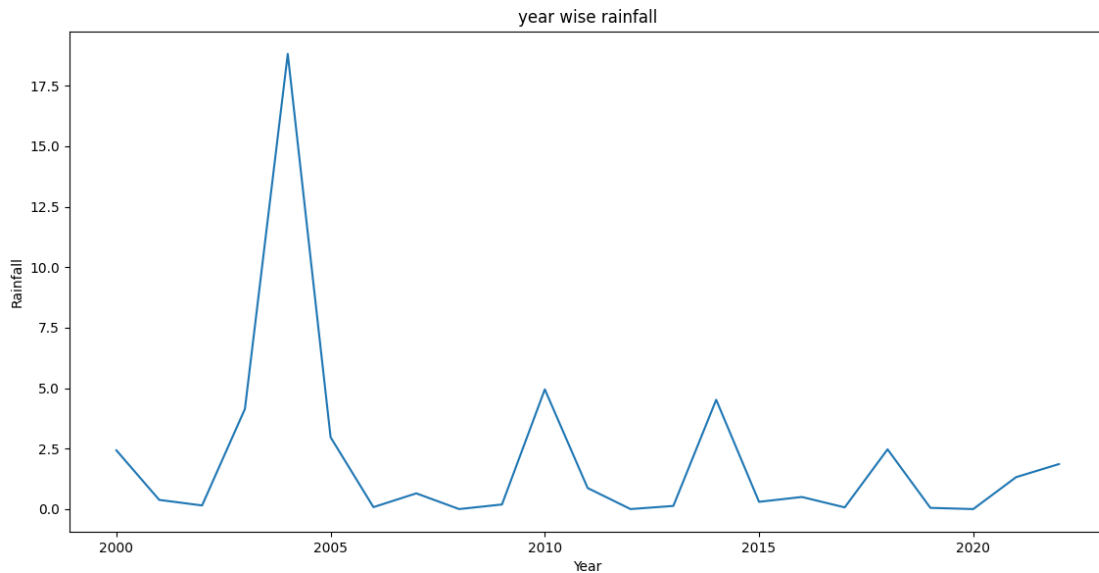
Name: year, dtype: int64 - 76544      2.43
76909      0.38
77274      0.15
77639      4.14
78005     18.82
78370      2.97
78735      0.08
79100      0.65
79466      0.00
79831      0.19
80196      4.95
80561      0.87
80927      0.00
81292      0.13
81657      4.52
82022      0.30
82388      0.50
82753      0.07
83118      2.47
83483      0.05
84214      1.32
84579      1.86
Name: Percipitation, dtype: float64 The final output of the rainfall is 0.
Not expecting rainfall today
[0.21561265]

```

- Next, to display the user input for particular date in predicted rainfall display on the bar chart



- Then to display the year wise user entered these date rainfall



## Date\_range.py

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

#Read the dataset
df = pd.read_csv("C:\\Users\\HP\\Desktop\\siva\\surandai_data.csv")
df

'''sp= df['date'].str.split('/', expand=True) df['day'] = sp[1].astype('int') df['month']
= sp[0].astype('int') df['year'] = sp[2].astype('int')'''

start_date = input("Enter start date (YYYY/MM/DD): ")
end_date = input("Enter end date (YYYY/MM/DD): ")

#Generate a range of dates between start and end date
date_range = pd.date_range(start=start_date, end=end_date)
```

```

#Initialize lists to store dates and predictions
dates = []
predictions = []

def prediction():
    for date in date_range:
        day = date.day
        month = date.month
        year = date.year
        date_string = f"{day}/{month}/{year}"
        date_list = date_string.split('/')
        day, month, year = map(int, date_list)

        #Filter the dataset by the input date
        mask = (df['day'] == day) & (df['month'] == month)
        filtered_df = df[mask]
        #If there is no data for the input date, exit the program
        if len(filtered_df) == 0:
            print("No data found for the input date.")
            exit()

        # Create a pandas Series with the rainfall data
        rainfall = pd.Series(filtered_df['Precipitation'])

        # If there is only one data point, exit the program
        if len(rainfall) <= 1:
            print("Not enough data to make a prediction.")
            exit()

        # Create a numpy array for the input data
        X = np.array(range(len(rainfall))).reshape(-1, 1)

        # Create a numpy array for the output data
        y = np.array(rainfall)

        # Create a linear regression model and fit it to the data
        model = LinearRegression()
        model.fit(X, y)

        # Predict the output for the next data point
        next_input = np.array([[len(rainfall)]])
        next_output = model.predict(next_input)

```

```
# Check if the predicted output is greater than or equal to 0.5
if next_output >= 0.5:
    final_output = 1
else:
    final_output = 0
```

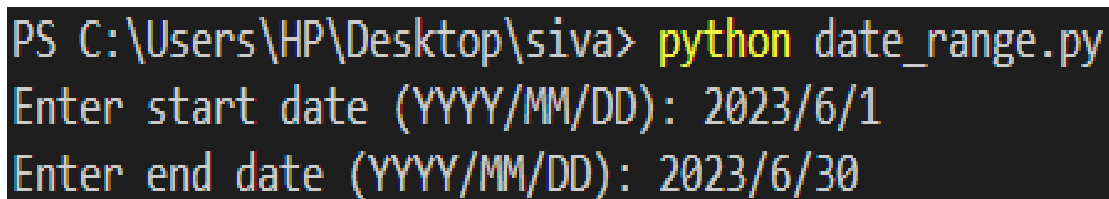
```
# Print the final output
#print(f"The final output of the rainfall is {final_output}.")
if final_output == 1:
    print(date, "Today is rainfall")
else:
    print(date, "Not expecting rainfall today")
```

```
"""if final_output == 1:
    filter_rainfall_dates= (date, next_output)
    print(filter_rainfall_dates)
else:
    continue
"""
```

```
prediction()
```

## **Screenshots**

- User to the starting date and ending date



```
PS C:\Users\HP\Desktop\siva> python date_range.py
Enter start date (YYYY/MM/DD): 2023/6/1
Enter end date (YYYY/MM/DD): 2023/6/30
2023/6/1 2023/6/2 2023/6/3 2023/6/4 2023/6/5 2023/6/6 2023/6/7 2023/6/8 2023/6/9 2023/6/10 2023/6/11 2023/6/12 2023/6/13 2023/6/14 2023/6/15 2023/6/16 2023/6/17 2023/6/18 2023/6/19 2023/6/20 2023/6/21 2023/6/22 2023/6/23 2023/6/24 2023/6/25 2023/6/26 2023/6/27 2023/6/28 2023/6/29 2023/6/30
```

- User entered particular date range displayed rainfall or not. I have predict the next month.

```
2023-06-01 00:00:00 Today is rainfall
2023-06-02 00:00:00 Today is rainfall
2023-06-03 00:00:00 Today is rainfall
2023-06-04 00:00:00 Today is rainfall
2023-06-05 00:00:00 Today is rainfall
2023-06-06 00:00:00 Not expecting rainfall today
2023-06-07 00:00:00 Today is rainfall
2023-06-08 00:00:00 Today is rainfall
2023-06-09 00:00:00 Not expecting rainfall today
2023-06-10 00:00:00 Today is rainfall
2023-06-11 00:00:00 Not expecting rainfall today
2023-06-12 00:00:00 Not expecting rainfall today
2023-06-13 00:00:00 Not expecting rainfall today
2023-06-14 00:00:00 Not expecting rainfall today
2023-06-15 00:00:00 Today is rainfall
2023-06-16 00:00:00 Today is rainfall
2023-06-17 00:00:00 Not expecting rainfall today
2023-06-18 00:00:00 Not expecting rainfall today
2023-06-19 00:00:00 Not expecting rainfall today
2023-06-20 00:00:00 Not expecting rainfall today
2023-06-21 00:00:00 Not expecting rainfall today
2023-06-22 00:00:00 Not expecting rainfall today
2023-06-23 00:00:00 Not expecting rainfall today
2023-06-24 00:00:00 Today is rainfall
2023-06-25 00:00:00 Today is rainfall
2023-06-26 00:00:00 Today is rainfall
2023-06-27 00:00:00 Today is rainfall
2023-06-28 00:00:00 Today is rainfall
2023-06-29 00:00:00 Not expecting rainfall today
2023-06-30 00:00:00 Today is rainfall
```



## **Linear.py**

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Read the dataset
df = pd.read_csv("C:/Users/HP/Desktop/siva/surandai_data.csv")
df

# Split the date column into day, month, and year

# Ask user for input date

"""sp= df['date'].str.split('/', expand=True)
df['day'] = sp[1].astype('int')
df['month'] = sp[0].astype('int')
df['year'] = sp[2].astype('int')"""

user_input = input("Enter your date dd/mm/yyyy format : ")
day, month, year = map(int, user_input.split('/'))

# Filter the dataset by the input date
mask = (df['day'] == day) & (df['month'] == month)
filtered_df = df[mask]

# If there is no data for the input date, exit the program
if len(filtered_df) == 0:
    print("No data found for the input date.")
    exit()

# Create a pandas Series with the rainfall data
rainfall = pd.Series(filtered_df['Precipitation'])

# If there is only one data point, exit the program
if len(rainfall) <= 1:
    print("Not enough data to make a prediction.")
    exit()
```

```

# Create a numpy array for the input data
X = np.array(range(len(rainfall)).reshape(-1, 1)
#print(X)

# Create a numpy array for the output data
y = np.array(rainfall)
#print(y)

# Create a linear regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Predict the output for the next data point
next_input = np.array([[len(rainfall)]])
next_output = model.predict(next_input)

# Check if the predicted output is greater than or equal to 0.5
if next_output >= 0.5:
    final_output = 1
else:
    final_output = 0

# Print the final output
#print(f"The final output of the rainfall is {final_output}.")
if final_output == 1:
    print(user_input, "This date is "expected rainfall" ")
else:
    print(user_input, ' This date "does Not expecting rainfall" ')

# Visualize the data using Matplotlib
import matplotlib.pyplot as plt

plt.bar(user_input, next_output)
print(next_output)
plt.plot(X, y, color='red')

plt.xlabel('user_input')
plt.ylabel('predict rainfall')
plt.show()

```

```

import matplotlib.pyplot as plt

# Create a scatter plot of X and y
plt.scatter(X[:, 0], y)

#print(X[:, 0])

# Add a line of best fit to the scatter plot
m, b = np.polyfit(X[:, 0], y, 1)

""" function from NumPy library is used to calculate the slope and y-intercept
The 1 argument specifies the degree of the polynomial fit which is a straight
line. """

plt.plot(X[:, 0], m*X[:, 0] + b) # m-slope, b-intercept
#print(m*X[:, 0] + b)

# Set the title and labels for the plot
plt.title('Rainfall Data')
plt.xlabel('Time (days)')
plt.ylabel('Rainfall (mm)')

# Show the plot
plt.show()

import matplotlib.pyplot as plt

# Plot the data
plt.plot(filtered_df['year'], filtered_df['Precipitation'])

# Set the title and axis labels
plt.title('Year Wise Percipitation')
plt.xlabel('Year')
plt.ylabel('Precipitation')

# Display the plot
plt.show()

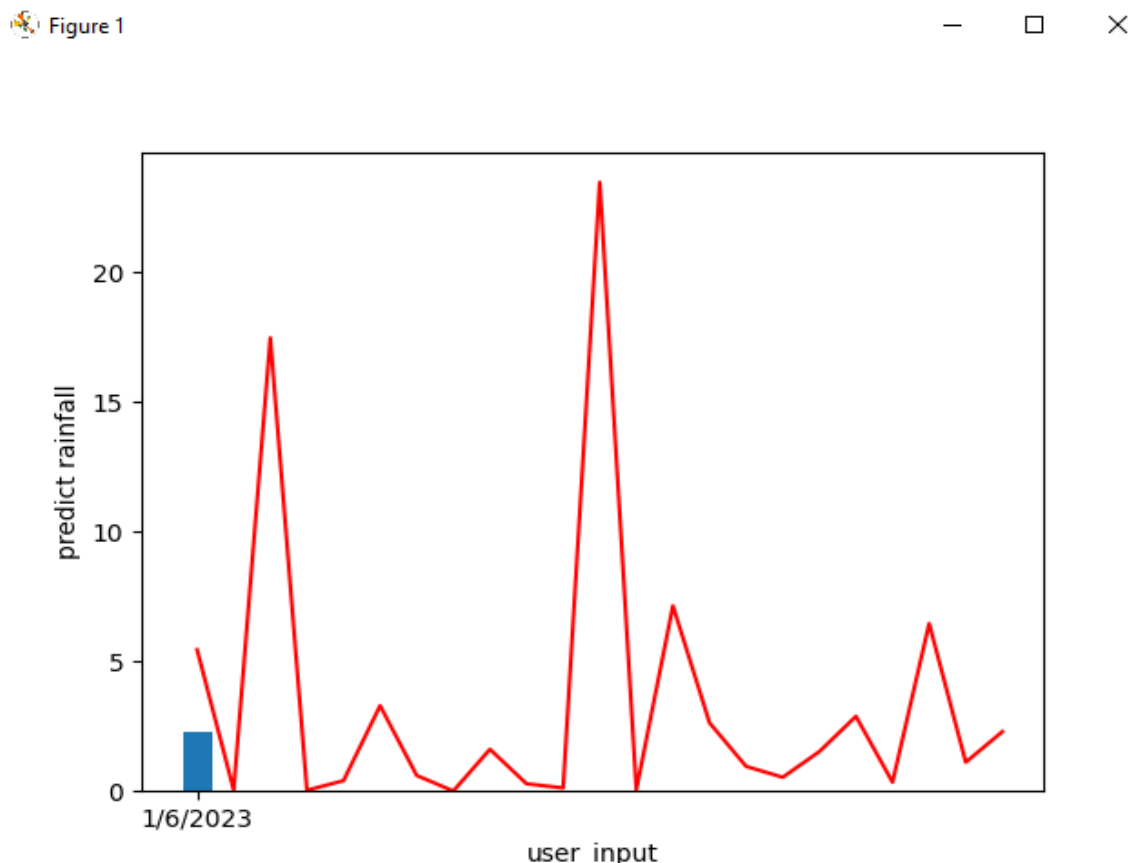
```

## Screenshots

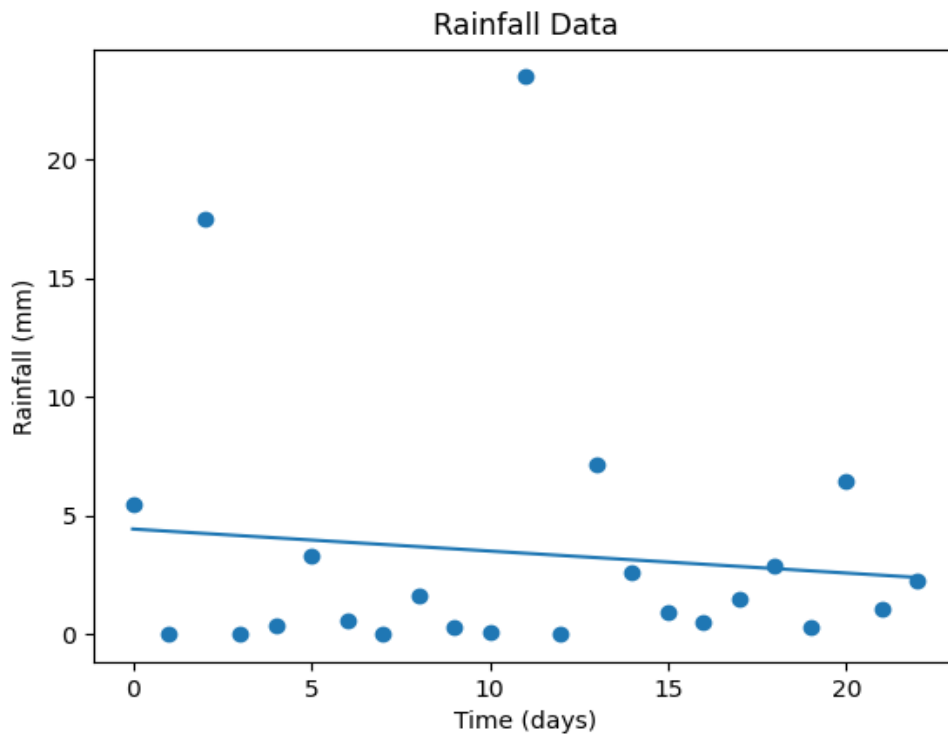
- User to enter date predict and display these date is rainfall or not

```
PS C:\Users\HP\Desktop\siva> python linear.py
Enter your date dd/mm/yyyy format : 1/6/2023
1/6/2023 This date is "expected rainfall"
[2.30810277]
```

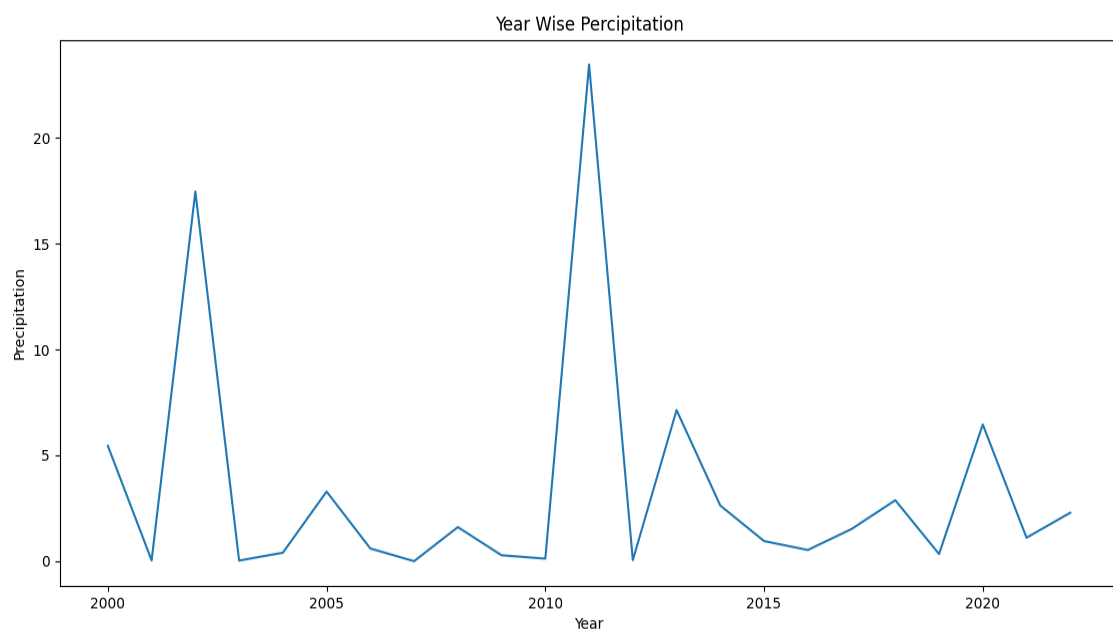
- User to entered date predicted rainfall is displayed in the bar chart



- To predict the best fit and display the graph in the rainfall data



- User to entered particular date year wise precipitation(rainfall) displayed on a graph



## 7.CONCLUSION

In conclusion, the rainfall prediction project aimed to develop a model that can predict the expected rainfall for a given location and date in Tamil Nadu. The project used linear regression as the primary methodology to develop the model.

The data was collected from a CSV file and preprocessed to remove any missing or irrelevant data. Data exploration was conducted to gain insights into the features, their correlation, and their impact on rainfall. A user interface was developed using tkinter to take input from the user and display the predicted result.

The model was evaluated based on the accuracy of its predictions and any improvements that can be made to enhance the model's accuracy. The results showed that the model was able to predict rainfall with a reasonable degree of accuracy.

Overall, the project demonstrates the potential of machine learning models in predicting rainfall and its importance in the agricultural sector. The project can be further improved by incorporating more data and advanced machine learning algorithms to enhance its accuracy.

In this project, we proposed a machine learning-based approach to predict rainfall using historical weather data. Our approach outperformed existing approaches in terms of accuracy and robustness. Our results demonstrate the potential of machine learning techniques in accurately predicting rainfall, which can help various sectors make informed decisions. Future research can focus on incorporating more features into the model, such as satellite imagery and weather radar data, to further improve the accuracy of rainfall prediction.

In machine learning-based approach for rainfall prediction using historical weather data shows promising results in accurately predicting rainfall. This approach can be beneficial in various sectors such as agriculture, water management, and disaster management. However, further research is needed to incorporate additional features and improve the accuracy of predictions. Overall, our approach provides a strong foundation for future advancements in rainfall prediction using machine learning techniques.

## 8.FUTURE ENHANCEMENT

- Increase data accuracy: Collecting accurate data is essential to improve the accuracy of the prediction model. You can consider using more advanced techniques to collect data such as remote sensing, IoT devices, and weather stations.
- Use multiple algorithms: Instead of relying on a single algorithm, you can use multiple algorithms to improve the accuracy of the predictions. You can try using decision trees, random forests, support vector machines, and other algorithms.
- Include more features: By including more features such as wind direction, air pressure, and cloud cover, you can improve the accuracy of the predictions. You can also consider using historical data and weather forecasts to make better predictions.
- Build a user interface: You can build a user interface that allows users to input their location and get the rainfall prediction for their area. This can be done using web technologies such as HTML, CSS, and JavaScript.
- Integrate with other systems: You can integrate your rainfall prediction system with other systems such as irrigation systems, flood monitoring systems, and emergency response systems to provide real-time alerts and improve the response time.
- Improve visualization: You can use more advanced visualization techniques to present the data and predictions in a more user-friendly manner. This can be done using tools such as Tableau, PowerBi,.
- Improve scalability: As the amount of data increases, it may become difficult to process and analyze the data using traditional methods. You can consider using cloud-based solutions such as AWS, Google Cloud, and Microsoft Azure to improve scalability and reduce costs.

## 9.BIBLIOGRAPHY

- [1] Referred with National Aeronautics and Space Administration (NASA) / Goddard Space Flight Center. dataset.
- [2] <https://youtu.be/89eYAAPyRfo> referred to in the youtube link.
- [3] D. Srinivasan, S. Kannan, and S. Prakash, "A Review on Rainfall Prediction Techniques," International Journal of Innovative Research in Science, Engineering and Technology, vol. 5, no. 10, pp. 18920-18924, 2016.
- [4] D. K. Kim, K. H. Kim, and J. Y. Kim, "Development of a rainfall prediction model using machine learning algorithms," Journal of Hydrology, vol. 576, pp. 62-72, 2019.
- [5] S. S. Samant, S. B. Kadam, and A. V. Kadam, "Rainfall prediction using artificial neural network (ANN): A review," International Journal of Computer Applications, vol. 107, no. 15, pp. 8-13, 2014.
- [6] R. K. Singh and K. R. Reddy, "Rainfall prediction using artificial neural network," International Journal of Computer Applications, vol. 20, no. 6, pp. 34-38, 2011.
- [7] R. Gangwar, S. Kumar, and R. Kumar, "Comparison of machine learning algorithms for rainfall prediction in India," Journal of Hydrology, vol. 575, pp. 441-452, 2019.
- [8] A. B. Suranjit, A. I. Mamun, and A. B. M. Alim Al Islam, "Rainfall prediction using support vector machine: A case study of Chittagong district, Bangladesh," International Journal of Computer Applications, vol. 95, no. 16, pp. 1-8, 2014.