**ABSTRACT**

Weather is a vital part of a person's life because it can tell us whether it will rain or be sunny. Weather forecasting is meteorologists' attempt to predict weather conditions in the future, as well as weather conditions that may be predicted. Temperature, pressure, humidity, dew point, rainfall, precipitation, wind speed, and dataset size are all used to calculate the climatic state parameters. To begin, the data must be educated. We can use 75-90% of the data from the data collection to train the data.

We'll use the Linear Regression Algorithm and the Nave Bayesian Classification Algorithm to make this prediction. Python, NumPy, Jupiter Notebook, Spyder, and Panda will be used in this project. The project is split into three separate Jupiter Notebooks: one to collect the weather data, inspect it, and clean it; a second to further refine the features and fit the data to a Linear Regression model and Naïve Bayesian model and a third to train and evaluate our output.

**ACKNOWLEDGEMENT**

* We’re pleased to acknowledge our sincere thanks to Board of Management of Sathyabama University for their kind encouragement in doing this hackathon project and for completing it successfully.
* We would like to express our sincere and deep sense of gratitude to our Project Guide Dr. S.Revathy, Associate Professor., for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

**INTRODUCTION**

The application of physics principles, augmented by a range of statistical and analytical methods, to predict the weather is known as weather forecasting. Weather forecasting provides forecasts of shifts in the Earth's surface temperature in addition to predictions of atmospheric phenomena. These shifts are influenced by atmospheric conditions such as snow and ice cover.

The foundation for weather prediction began with ancient Greek philosophers' theories and continued with Renaissance scientists.Any weather prediction requires a systematic compilation of weather records from different locations, as well as adequate data analysis and prediction.

**AGENDA**

Weather forecasting is the use of science and technology to predict the condition of the atmosphere at any given time span. Weather forecasting can be done in a variety of ways. The importance of weather warning warnings is that they can be used to avoid the loss of life and the degradation of the ecosystem.

Weather forecasting methods in ancient times were typically based on pattern recognition, or studying patterns of events. For example, if the previous day's sunset is especially red, the following day's weather is expected to be pleasant. However, none of the predictions turn out to be accurate.

Weather forecasting is the process of predicting future weather based on historical data such as temperature, humidity, dew, wind speed and direction, precipitation, haze and air quality, solar and terrestrial radiation, and so on. The weather forecast has a major impact on people's daily lives. It is educated after the data is collected.

**TRACK AND LANGUAGE USED:**

**Python** is a programming language that stands out from other programming languages because of its versatility, accessibility, and dependable tools for developing modern software. Python is the most suitable language for machine learning because it is consistent and based on simplicity. Python is the best programming language for machine learning because of its independent platform and widespread use.

**Python** is a programming language that is widely used because of its extensive functionality, applicability, and ease of use. Python is the best programming language for machine learning because of its independent platform and popularity among programmers.

**Machine learning** is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data.

**AIM / SCOPE:**

The goal of weather prediction is to provide information people and organizations can use to reduce weather-related losses and enhance societal benefits, including protection of life and property, public health and safety, and support of economic prosperity and quality of life.

In economic terms, the benefit of the investment in public weather forecasts and warnings is substantial: the estimated annualized benefit is about $31.5 billion, compared to the $5.1 billion cost of generating the information.

The Linear Regression algorithm, which is used to forecast weather using these data, is at the centre of this project. The higher the precision, the more parameters considered.

This project has the potential to assist a large number of people in predicting tomorrow's weather. Temperature, dew, pressure, and humidity are simply used to train the data in this project. These data are then used to train a prediction model using Linear Regression

**FORECAST DATA:**

* Between 1980 and 2009 - 96 weather disasters in the United States each caused at least $1 billion in damages, with total losses exceeding $700 billion (NCDC, 2010).
* Between 1999 and 2008 - there were an average of 629 direct weather fatalities per year (NWS, 2010).
* The annual impacts of adverse weather on the national highway system and roads are staggering:
* 1.5 million weather-related crashes with 7,400 deaths, more than 700,000 injuries, and $42 billion in economic losses (BTS, 2007).
* In addition, $4.2 billion is lost each year as a result of weather-related air traffic delays (NOAA, 2010).
* Weather is also a major factor in the complex set of interactions that determine air quality; more than 60,000 premature deaths each year are attributed to poor air quality (Schwartz and Dockery, 1992).

**PRACTICAL APPLICATIONS OF WEATHER FORECASTING:**

* Systematic weather records were kept after the invention of the instruments for measuring atmospheric conditions during the 17th century. Undoubtedly, these early records were employed mainly by those engaged in agriculture.
* Planting and harvesting can be planned better and carried out more efficiently if all the long-term weather patterns are estimated in advance.
* Weather warnings are a special kind of short-range forecasts. It is needed for the protection of human life from weather extremes. Weather warnings are issued by government and military organizations throughout the world for all kinds of threatening weather events like tropical storms which are called as hurricanes, typhoons, or tropical cyclones, depending on location.

#### The steps involved in pre-processing are: –

* **Features selection**

The data we have collected has many unwanted attributes which will not be needed in our project. Hence, we use the attributes which we need only.

* **Normalization**

The data we collected from internet should be first normalized. Normalization refers to rescaling real valued numeric attributes into the rage or 0 and 1. After the data are filtered it is then normalized.

* **Machine Learning**

Training a model is the process of iteratively improving your prediction equation by looping through the dataset multiple times, each time updating the weight and bias values in the direction indicated by the slope of the cost function (gradient). Training is complete when we reach an acceptable error threshold, or when subsequent training iterations fail to reduce our cost.

#### SOFTWARE REQUIREMENT

The software used in our project is:

* **Python 3.7**: Python is an interpreter, high level, general programming language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. It provides a vast library for data mining and predictions.
* **Python IDLE:** IDLE is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit.

#### FUNCTIONAL REQUIREMENTS

* The system must provide the predicted weather.
* The system must have an easy to use interface for using the system for all the users.
* The Admin must be able to update/modify the Dataset.
* The Dataset of the weather must be available for the system.

**DATA COLLECTION**

The data of weather forecast was obtained from Kaggle. We took about 4000 trained data and 800 test data.

**Parameters are:-**

* Temperature
* Pressure
* Humidity
* Dew point
* Rainfall
* Precipitation

**TEMPERATURE PREDICTION**

* We'll work on forecasting the average global land and ocean temperature using over 100 years of historical weather data. We'll pretend that we don't have access to any weather forecasts.
* Through a comparative study of weather data collected in Central Kerala from 2007 to 2015, we propose a system for temperature prediction using three machine learning models
* Multiple Linear Regression (MLR), Artificial Neural Network (ANN), and Support Vector Machine (SVM)
* We still have a century's worth of historical global temperature averages, including global maximum and minimum temperatures, as well as global land and ocean temperatures.
* Having all of this, we know that this is a supervised, regression machine learning problem.
* Mean Error (ME), Mean Absolute Error (MAE), and other metrics are used to assess the experimental outcomes.

**ADVANTAGES AND DISADVANTAGES OF WEATHER FORECASTING:**

|  |  |
| --- | --- |
| **WEATHER FORECASTING** | |
| ADVANTAGES | DISADVANTAGES |
| * Farmers can know when to plant or harvest their crops | * Weather is extremely difficult to forecast correctly |
| * People can choose where and when to take their holidays to take advantages of good weather | * It is expensive to monitor-so many variables from so many sources |
| * Surfers known when large waves are expected | * The computers needed to perform the millions of calculations necessary are expensive |
| * Regions can be evacuated if hurricanes or floods are expected | * The weather forecasters get blamed if the weather is different from the forecast |
| * Aircraft and shipping rely heavily on accurate weather forecasting | * Weather is extremely difficult to forecast correctly |

**APPLICATIONS IN VARIOUS FIELDS:**

There are a number of sectors with their own specific needs for weather forecasts and specialist services are provided to these users.

* 1. **AIR TRAFFIC:**
* Accurate weather forecasting is critical in the aviation industry because it is so weather-sensitive. Many planes are unable to land or take off due to fog or very low ceilings.  Turbulence and freezing are two other major in-flight dangers.
* Thunderstorms cause extreme turbulence due to updrafts and outflow boundaries, icing, and are a concern for all aircraft. Volcanic ash is also a significant issue for aviation, as ash clouds can cause aircraft to lose engine power.
* Airliners are diverted on a daily basis to take advantage of the jet stream tailwind to increase fuel quality. Prior to take off, aircrews are briefed on what to expect en route and at their destination.  Furthermore, airports often alter which runway is in use.

### MARINE

* Wind direction and intensity, wave periodicity and heights, tides, and precipitation can all restrict commercial and recreational use of waterways. Any of these factors may have an effect on the safety of marine transportation.
* As a result, a number of codes have been developed to effectively transmit comprehensive marine weather forecasts to vessel pilots via radio, such as the MAFOR (marine forecast) code.
* RTTY, Navtex, and Radiofax can all be used to receive standard weather forecasts at sea.

### AGRICULTURE

* Farmers use weather forecasts to determine what work they can do on any given day. Drying hay, for example, is only possible in dry weather. Dry spells will wreak havoc on cotton, wheat, and corn crops.
* Drought can destroy corn crops, but their dried remains, known as silage, can be used as a cattle feed substitute. Both in the spring and the fall, frosts and freezes wreak havoc on crops. A spring freeze, for example, will decimate the future peach crop of peach trees in full bloom. Orange groves can be severely harmed by frosts and freezes, regardless of when they occur.
  1. **FORESTRY**
* Wind, precipitation, and humidity forecasting are critical for preventing and managing wildfires.
* Various indices have been created, such as the Forest fire weather index and the Haines Index, to predict which areas are more likely to experience fire due to natural or human causes.

### 5) UTILITY COMPANIES

* Rain forecasts are used by electricity and gas firms to predict demand, which is heavily influenced by the weather. They calculate the degree day to see how much heating (heating degree day) or cooling (cooling degree day) can be used (cooling degree day).
* These figures are based on a daily average temperature of 65 degrees Fahrenheit (18 degrees Celsius).Heating degree days (one per degree Fahrenheit) are triggered by cooler temperatures, whereas cooling degree days are triggered by warmer temperatures.
* In the winter, when people turn up their heating, extreme cold weather can cause a surge in demand. Similarly, increased use of air conditioning systems in hot weather will lead to a rise in demand in the summer. By foreseeing a spike in demand.

### OTHER COMMERCIAL COMPANIES

### Private businesses are increasingly paying for weather forecasts tailored to their specific needs in order to maximise revenues or prevent major losses.

### Supermarket chains, for example, can adjust their shelf stocking in response to changing customer buying patterns and weather conditions. Weather forecasts can be used to invest in commodity futures such as oranges, corn, and wheat.

**Code Used:**

**Link for the Dataset Collection**

***https://github.com/Jesuschrist286/weather\_Prediction-codehackathon-/blob/main/Weather%20Data%20in%20India%20from%201901%20to%202017.csv***

**Api.py:**

|  |  |
| --- | --- |
|  | import tkinter as tk |
|  | import time |
|  |  |
|  | def getWeather(canvas): |
|  | city = textField.get() |
|  | api = "https://api.openweathermap.org/data/2.5/weather?q="+city+"&appid=06c921750b9a82d8f5d1294e1586276f" |
|  |  |
|  | json\_data = requests.get(api).json() |
|  | condition = json\_data['weather'][0]['main'] |
|  | temp = int(json\_data['main']['temp'] - 273.15) |
|  | min\_temp = int(json\_data['main']['temp\_min'] - 273.15) |
|  | max\_temp = int(json\_data['main']['temp\_max'] - 273.15) |
|  | pressure = json\_data['main']['pressure'] |
|  | humidity = json\_data['main']['humidity'] |
|  | wind = json\_data['wind']['speed'] |
|  | sunrise = time.strftime('%I:%M:%S', time.gmtime(json\_data['sys']['sunrise'] - 21600)) |
|  | sunset = time.strftime('%I:%M:%S', time.gmtime(json\_data['sys']['sunset'] - 21600)) |
|  |  |
|  | final\_info = condition + "\n" + str(temp) + "°C" |
|  | final\_data = "\n"+ "Min Temp: " + str(min\_temp) + "°C" + "\n" + "Max Temp: " + str(max\_temp) + "°C" +"\n" + "Pressure: " + str(pressure) + "\n" +"Humidity: " + str(humidity) + "\n" +"Wind Speed: " + str(wind) + "\n" + "Sunrise: " + sunrise + "\n" + "Sunset: " + sunset |
|  | label1.config(text = final\_info,foreground = "white",bg="#8C001A") |
|  | label2.config(text = final\_data,foreground = "white",bg="#8C001A") |
|  | canvas = tk.Tk() |
|  | canvas.geometry("600x500") |
|  | canvas.configure(bg='#8C001A') |
|  | canvas.title("Weather App") |
|  | f = ("poppins", 15, "bold") |
|  | t = ("poppins", 35, "bold") |
|  | label3 = tk.Label(canvas,borderwidth = 4,font=f) |
|  | label3.pack() |
|  | switch = "\n" + "Enter your city:" + "\n" |
|  | label3.config(text = switch,foreground = "white",bg="#8C001A") |
|  | textField = tk.Entry(canvas,justify='center', width = 20, font = t,foreground="white") |
|  | textField.configure(bg="#C04000", insertbackground='black') |
|  | textField.pack(pady = 20) |
|  | textField.focus() |
|  | textField.bind('<Return>', getWeather) |
|  |  |
|  | label1 = tk.Label(canvas, font=t) |
|  | label1.pack() |
|  | label2 = tk.Label(canvas, font=f) |
|  | label2.pack() |
|  | canvas.mainloop() |

**Grid.py,**

|  |
| --- |
|  |
|  | from tkinter import \* |
|  | def weather(): |
|  | import Api |
|  | Api.mains() |
|  | def tomorrow(): |
|  | import rainfall |
|  | rainfall.possible\_rain() |
|  | def analyze(): |
|  | import rainfall |
|  | rainfall.analyze() |
|  | root = Tk() #makes a blank popup, under the variable name 'root' |
|  | topFrame = Frame(root,bg="#837E7C") |
|  | root.geometry("1000x900") |
|  | topFrame.pack() |
|  | bottomFrame = Frame(root) |
|  | bottomFrame.pack(side=BOTTOM) |
|  | root.title("Weather App") |
|  | root.configure(bg='#2C3539') |
|  | f = ("poppins", 15, "bold") |
|  | t = ("poppins", 35, "bold") |
|  | label1 = Label(topFrame,borderwidth = 1,font = t) |
|  | label1.pack() |
|  | title = "\n" + "WEATHER REPORT " + "\n" |
|  | label1.config(text = title,foreground = "white",bg="#837E7C",width = 20) |
|  | button3 = Button(topFrame, text='Analyze avarage rainfall every month for india', fg='green',command = analyze,font = f,bg='#0C090A') |
|  | button1 = Button(topFrame, text='Current weather report for specific city..........',command = weather, fg='red',font = f,bg='#0C090A') |
|  | button2 = Button(topFrame, text='Do you need to predict rainfall tomorrow......?', fg='blue',command = tomorrow,font = f,bg='#0C090A') |
|  | #button4 = Button(topFrame, text='Button 4', fg='pink') |
|  | button2.pack(side=BOTTOM,padx=20, pady=20) |
|  | button3.pack(side=BOTTOM,padx=20, pady=20) |
|  | button1.pack(side=BOTTOM,padx=20, pady=20) |
|  | #button4.pack(side=BOTTOM) |
|  |  |
|  | root.mainloop() #loops the program forever until its closed |

**Image.py**

|  |
| --- |
|  |
|  |  |
|  | import tkinter as tk  def api\_image(): |
|  | import api |
|  | api.mains() |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | canvas = tk.Tk() |
|  | canvas.geometry("1200x1100") |
|  | canvas.title("Weather App") |
|  | #canvas.configure(bg='#6b8e23') |
|  | f = ("poppins", 15, "bold",) |
|  | t = ("poppins", 35, "bold") |
|  | button = Button(canvas, |
|  | text = 'apipross', |
|  | command = api\_image) |
|  | button.pack() |
|  |  |
|  | canvas.mainloop() |

**Main.py**

from PIL import Image, ImageTk

|  |
| --- |
|  |
|  | import tkinter as tk |
|  |  |
|  |  |
|  | def answer(canvas): |
|  | city = textField.get() |
|  | if city == '1': |
|  | import Api |
|  |  |
|  | elif city == '2': |
|  | import rainfall |
|  | rainfall.possible\_rain() |
|  | elif city == '3': |
|  | import rainfall |
|  | rainfall.analyze() |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  |  |
|  |  |
|  | canvas = tk.Tk() |
|  | canvas.title("Weather App") |
|  | canvas.configure(bg='#6b8e23') |
|  | f = ("poppins", 15, "bold",) |
|  | t = ("poppins", 35, "bold") |
|  | label1 = tk.Label(canvas,borderwidth = 1,font=f) |
|  | label1.pack() |
|  | title = "\n" + "WEATHER REPORT " + "\n" |
|  | swi = "\n" + "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" |
|  | switch = title + swi + "\n" + "1:Current weather report for specific city " + "\n" + swi + "\n" + "2:Do you need to predict rainfall tomorrow?" + "\n" + swi + "\n"+ "3:Analyze the avarage rainfall every month for india" |
|  | label1.config(text = switch,foreground = "white",bg="#6b8e23") |
|  |  |
|  | textField = tk.Entry(canvas, justify='center', width = 20, font = t, foreground="red") |
|  | textField.configure(bg="#C04000", insertbackground='black') |
|  | textField.pack(pady = 20) |
|  | textField.focus() |
|  | textField.bind('<Return>', answer) |
|  |  |
|  | #label2 = tk.Label(canvas, font=f) |
|  | #label2.pack() |
|  |  |
|  | canvas.mainloop() |

**Photo.png**

<https://github.com/Jesuschrist286/weather_Prediction-codehackathon-/blob/main/photo.png?raw=true>

**Rainfall.py**

import pandas as pd

|  |
| --- |
|  |
|  | import matplotlib.pyplot as plt |
|  | df = pd.read\_csv('weatherAUS.csv') |
|  | df['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True) |
|  | df['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True) |
|  | def possible\_rain(): |
|  |  |
|  | fig = plt.figure(figsize = (8,5)) |
|  | df.RainTomorrow.value\_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0) |
|  | plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset') |
|  | plt.show() |
|  |  |
|  | def datatype(): |
|  | fig, axarr = plt.subplots(1, 2, figsize=(20, 8)) |
|  |  |
|  | df.dtypes.value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,ax=axarr[1]) |
|  | axarr[1].set\_title("type of our data ", fontsize=18) |
|  |  |
|  | df.dtypes.value\_counts().plot(kind='bar',ax=axarr[0]) |
|  | plt.title('type of our data'); |
|  | axarr[0].set\_title("type of our data ", fontsize=18) |
|  | plt.show() |
|  |  |
|  | def analyze(): |
|  | data = pd.read\_csv("Weather Data in India from 1901 to 2017.csv") |
|  | # Average monthly rainfall in India |
|  | ax=data[['JAN', 'FEB', 'MAR', 'APR','MAY', 'JUN', 'AUG', 'SEP', 'OCT','NOV','DEC']].mean().plot.bar(width=0.5, linewidth=2, figsize=(16,10)) |
|  | plt.xlabel('Month',fontsize=30) |
|  | plt.ylabel('Monthly Rainfall (in mm)', fontsize=30) |
|  | plt.title('Monthly Rainfall in Subdivisions of India', fontsize=25) |
|  | ax.tick\_params(labelsize=10) |
|  | plt.grid() |
|  | plt.show() |

**Temperature.py**

|  |
| --- |
|  |
|  |  |
|  | import numpy as np ## For Linear Algebra |
|  | ## For visualizations I'll be using plotly package, this creates interesting and interective visualizations. |
|  | import plotly.express as px |
|  | import plotly.graph\_objects as go |
|  | from plotly.subplots import make\_subplots |
|  | from datetime import datetime ## Time Series analysis. |
|  | df = pd.read\_csv("weather.csv") |
|  | df=pd.read\_csv("weather.csv",index\_col=0) |
|  | df.head() |
|  | print(df.head()) |
|  | df1 = pd.melt(df, id\_vars='YEAR', value\_vars=df.columns[1:]) ## This will melt the date |
|  | df1['Date'] = df1['variable'] + ' ' + df1['YEAR'].astype(str) |
|  | df1.loc[:,'Date'] = df1['Date'].apply(lambda x : datetime.strptime(x, '%b %Y')) ## Converting String to datetime object |
|  | df1.head() |
|  |  |
|  |  |
|  | def timeline(): |
|  | df1.columns=['Year', 'Month', 'Temprature', 'Date'] |
|  | df1.sort\_values(by='Date', inplace=True) ## To get the time series right. |
|  | fig = go.Figure(layout = go.Layout(yaxis=dict(range=[0, df1['Temprature'].max()+1]))) |
|  | fig.add\_trace(go.Scatter(x=df1['Date'], y=df1['Temprature']), ) |
|  | fig.update\_layout(title='Temprature Throught Timeline:',xaxis\_title='Time', yaxis\_title='Temprature in Degrees') |
|  | fig.update\_layout(xaxis=go.layout.XAxis( |
|  | rangeselector=dict( |
|  | buttons=list([dict(label="Whole View", step="all"), |
|  | dict(count=1,label="One Year View",step="year",stepmode="todate") |
|  | ])), |
|  | rangeslider=dict(visible=True),type="date") |
|  | ) |
|  | fig.show() |
|  |  |
|  |  |
|  |  |
|  | def yearly\_temp(): |
|  | df['Yearly Mean'] = df.iloc[:,1:].mean(axis=1) ## Axis 1 for row wise and axis 0 for columns. |
|  | fig = go.Figure(data=[ |
|  | go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='lines'), |
|  | go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='markers') |
|  | ]) |
|  | fig.update\_layout(title='Yearly Mean Temprature :', |
|  | xaxis\_title='Time', yaxis\_title='Temprature in Degrees') |
|  | fig.show() |
|  |  |
|  | def season(): |
|  | df['Winter'] = df[['DEC', 'JAN', 'FEB']].mean(axis=1) |
|  | df['Summer'] = df[['MAR', 'APR', 'MAY']].mean(axis=1) |
|  | df['Monsoon'] = df[['JUN', 'JUL', 'AUG', 'SEP']].mean(axis=1) |
|  | df['Autumn'] = df[['OCT', 'NOV']].mean(axis=1) |
|  | seasonal\_df = df[['YEAR', 'Winter', 'Summer', 'Monsoon', 'Autumn']] |
|  | seasonal\_df = pd.melt(seasonal\_df, id\_vars='YEAR', value\_vars=seasonal\_df.columns[1:]) |
|  | seasonal\_df.columns=['Year', 'Season', 'Temprature'] |
|  |  |
|  | fig = px.scatter(seasonal\_df, 'Year', 'Temprature', facet\_col='Season', facet\_col\_wrap=2, trendline='ols') |
|  | fig.update\_layout(title='Seasonal mean tempratures throught years:') |
|  | fig.show() |
|  | yearly\_temp() |

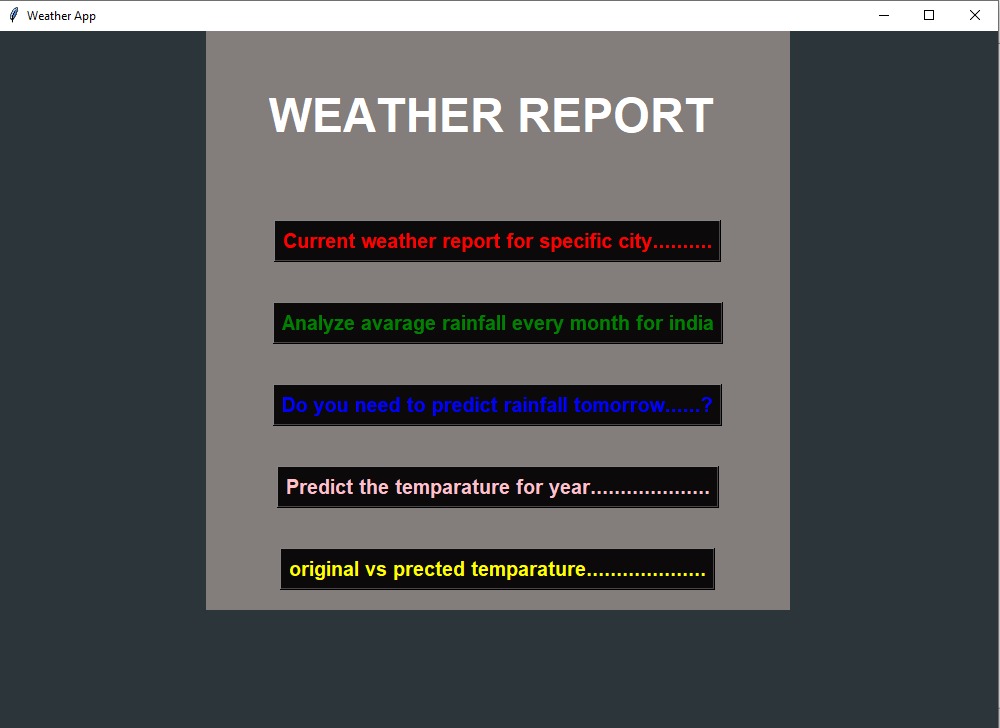
**Weat.png**

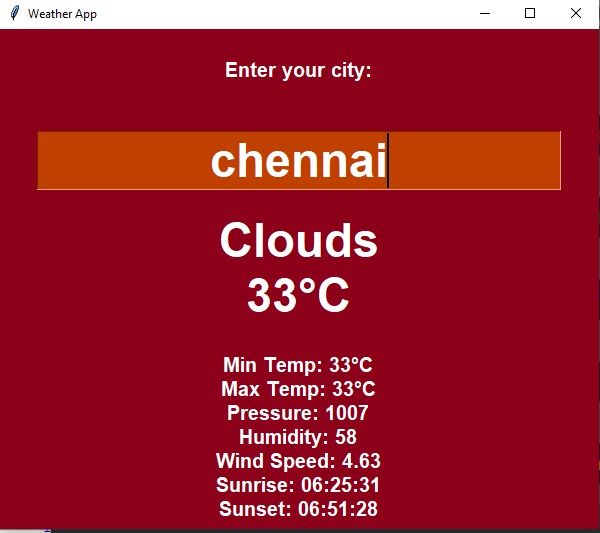
<https://github.com/Jesuschrist286/weather_Prediction-codehackathon-/blob/main/weat.png?raw=true>

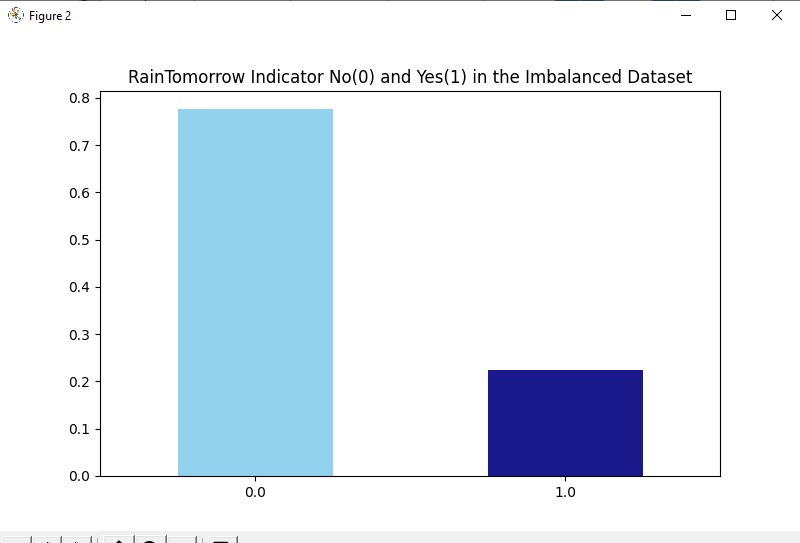
**Dataset For Weather**

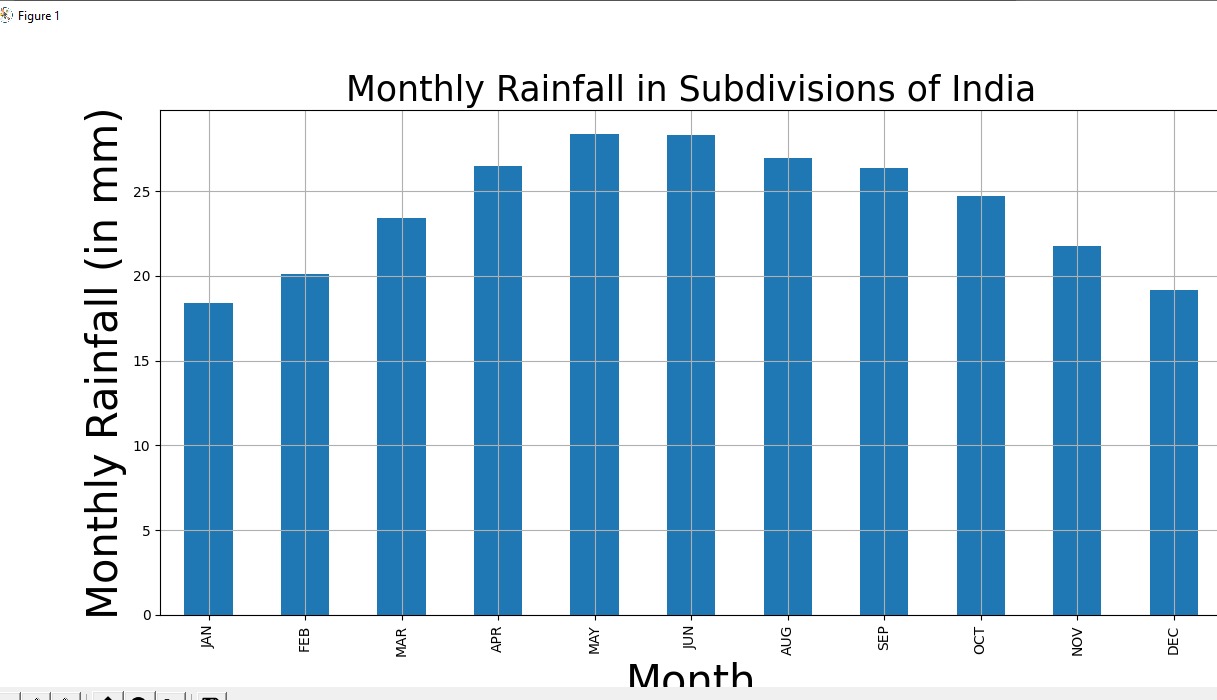
[**https://github.com/Jesuschrist286/weather\_Prediction-codehackathon-/blob/main/weather.csv**](https://github.com/Jesuschrist286/weather_Prediction-codehackathon-/blob/main/weather.csv)

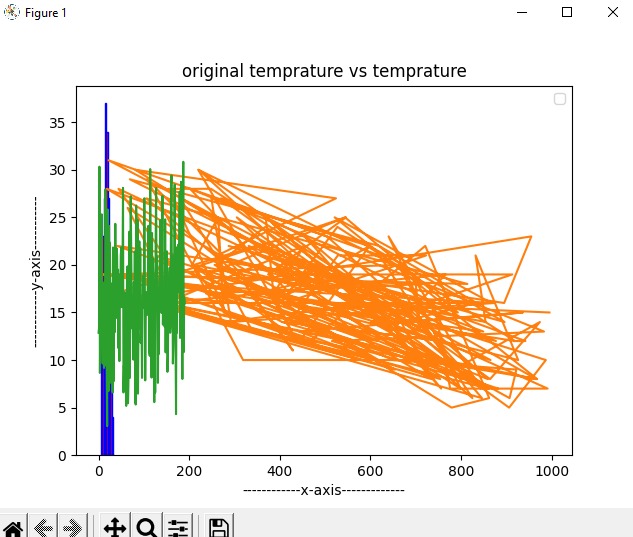
**OUTPUT:**

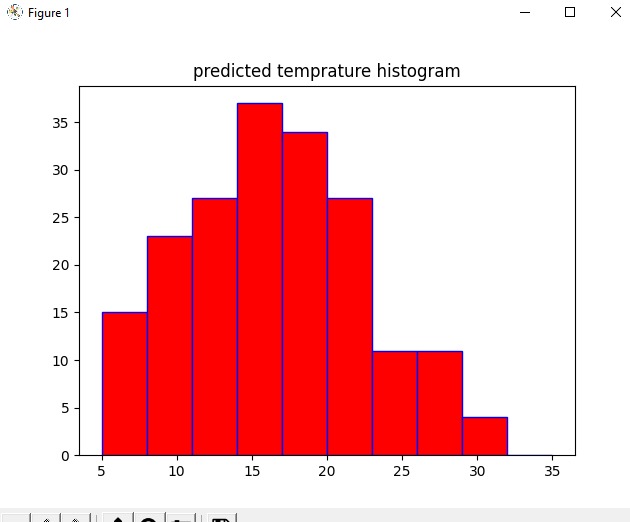
****

****

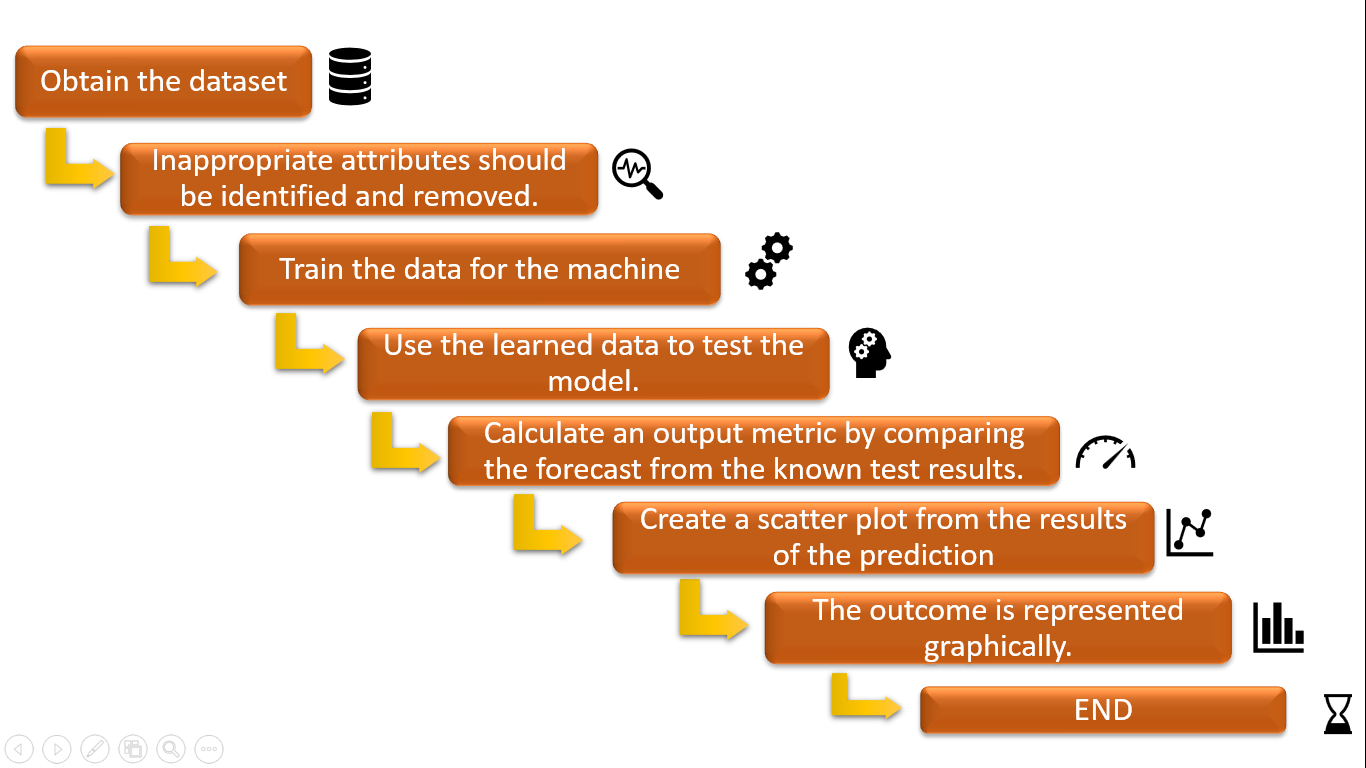
****

****

****

****

**WORKFLOW OF THE MODEL:**

**

**RESULT:**

The prediction method is in good working order. Many of the attribute values had been correctly preprocessed.

The model was implemented and trained using train data after all of the preprocessing was completed. Tkinter was used to build the system's user interface.

Pycharm/Spyder was used for the coding. We linked the front-end to the back-end after completing all of the steps. Our precision was discovered to be around.

We linked the front-end to the back-end after completing all of the steps. The accuracy of our predictions was found to be about 82 percent.

#### CONCLUSION

The weather prediction done using linear regression algorithm and Naïve Bayes algorithm are very essential for improving the future performance for the people.

For predicting the weather, the linear regression algorithm and Naïve Bayes algorithm was applied to the datasets of the weather. We made a model to predict the weather using some selected input variables collected from Kaggle.

The problem with current weather scenario is that we are not able to prepare our self and not able to do some important works.

So, for knowing the weather scenario at high accuracy considering every factor that affects in the weather scenario, this model is created.