RAINFALL PREDICTION USING RANDOM FOREST REGRESSION

**1.INTRODUCTION**

* Rainfall prediction remains a serious concern and has attracted the attention of governments, industries, risk management entities, as well as the scientific community.
* To this extent, rainfall prediction is essential since this variable is the one with the highest correlation with adverse natural events such as landslides, flooding, mass movements and avalanches. These incidents have affected society for years.
* Therefore, having an appropriate approach for rainfall prediction makes it possible to take preventive and mitigation measures for these natural phenomena.
* To solve this uncertainty, we used various machine learning techniques and models to make accurate and timely predictions.
* Rainfall prediction is one of the challenging and uncertain tasks which has a significant impact on human society.
* Timely and accurate predictions can help to proactively reduce human and financial loss .
* Rainfall prediction is helpful to avoid flood which save lives and properties of humans. Moreover, it helps in managing resources of water.

**2.PROBLEM STATEMENT**

To predict the rainfall with the help of the attributes MinTemp , MaxTemp , Rainfall, Evaporation, Sunshine, WindGustDir , WindGustSpeed for various machine learning techniques and models to make accurate and timely predictions .

**3. OBJECTIVE**

* Explore and visualize the data
* Build a classification model to predict the rainfall with the help of changes accordingly in the environment .
* Optimize the data with appropriate techniques and models .
* Generate the model with the help of various attributes checking the possibility of the rainfall .
* Finally with the help of the project , we can predict the rainfall.

**4. PROPOSED ALGORITHM**

**RANDOM FOREST CLASSIFICATION**

* The Random Forest Algorithm is composed of different decision trees, each with the same nodes, but using different data that leads to different leaves.
* It merges the decisions of multiple decision trees in order to find an answer, which represents the average of all these decision trees.
* The random forest algorithm is a supervised learning model; it uses labeled data to “learn” how to classify unlabeled data.
* This is the opposite of the K-means Cluster algorithm, which we learned in a past article was an unsupervised learning model
* The Random Forest Algorithm is used to solve both regression and classification problems, making it a diverse model that is widely used by engineers.

**5. ALGORITHM OF RANDOM FOREST CLASSIFICATION**

**Step 1** **:** Select random K data points from the training set.

**Step 2 :** Build the decision trees associated with the selected data points(Subsets)

**Step 3 :** Choose the number N for decision trees that you want to build.

**Step 4 :** Repeat Step 1 and 2

**Step 5 :** For new data points , find the prediction of each decision tree,and assign the new data points to the category that wins the majority votes.

**6. MATHEMATICAL CALCULATION**

Training time = O(log(nd)\*k)

Run time=O(depth\*k)

Space = O(store each DT\*K)

As the number of base model increases, training run time increases so always use Cross- validation to find optimal hyperparameter

For each decision tree, Scikit-learn calculates a nodes importance using Gini Importance, assuming only two child nodes (binary tree):

**nij= wjCj – wleft(j)Cleft (j) – wright(j)Cright(j)**

ni sub(j)= the importance of node j

w sub(j) = weighted number of samples reaching node j

C sub(j)= the impurity value of node j

left(j) = child node from left split on node j

right(j) = child node from right split on node j

sub() is being used as subscript isn’t available in Medium

The importance for each feature on a decision tree is then calculated as:

fii = Ʃj: node j splits on feature inij / ƩkЄ all nodes nik

fi sub(i)= the importance of feature i

ni sub(j)= the importance of node j

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

Normfi = fii / ƩjЄ all features of fi

The final feature importance, at the Random Forest level, is it’s average over all the trees.

The sum of the feature’s importance value on each trees is calculated and divided by the total number of trees:

**RFfi = Ʃj Єall trees normfi ij / T**

RFfi sub(i)= the importance of feature i calculated from all trees in the Random Forest model

normfi sub(ij)= the normalized feature importance for i in tree j

T = total number of trees

**Gini index = 1- Ʃ i=1 to n (Pi)^2**

**=1-[(** P+ **)^2+(P\_)^2]**

Where P+ is the probability of a positive class and P\_ is the probability of a negative class.

Putting the values of a left split in the formula we get:

Gini Index = 1- Σ i=1 to n (Pi)^2

                   =1-[( P+ )^2+(P\_)^2]

                   =1-[(1/3)^2+(2/3)^2]

                  =1-[0.1089+0.4356]

                  =1-0.5445

                  =0.4555

For the right split the Gini index will be:

Gini Index = 1- Σ i=1 to n (Pi)^2

                   =1-[( P+ )^2+(P\_)^2]

                   =1-[(1/2)^2+(1/2)^2]

                  =1-[0.25+0.25]

                  =1-0.5

                  =0.5

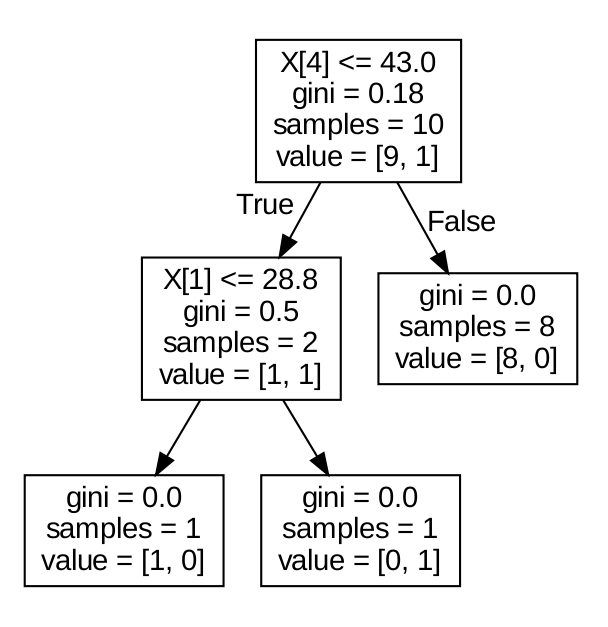
[Text(0.6, 0.8333333333333334, 'X[4] <= 43.0\ngini = 0.18\nsamples = 10\nvalue = [9, 1]'),

Text(0.4, 0.5, 'X[3] <= 13.0\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),

Text(0.2, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),

Text(0.6, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),

Text(0.8, 0.5, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]')]



**Fig 1** : **Process of the mathematical calculation of Random forest classifier**

**7.DATASET ATTRIBUTES**

* The data set under consideration contains daily weather observa-tions from numerous  weather stations

**Feature**  **Description**

MinTemp The minimum temperature in degrees celsius

MaxTemp The maximum temperature in degrees celsius

Rainfall The amount of rainfall recorded for the day in mm

Evaporation The so-called Class A pan evaporation (mm) in the 24 hours to 9am

Sunshine The number of hours of bright sunshine in the day.

WindGustDir The direction of the strongest wind gust in the 24 hours to midnight

WindGustSpeed The speed (km/h) of the strongest wind gust in the 24 hours to

midnight

WindDir9am Direction of the wind at 9am

WindDir3pm Direction of the wind at 3pm

WindSpeed9am Wind speed averaged over 10 minutes prior to 9am

WindSpeed3pm Wind speed averaged over 10 minutes prior to 3pm

Humidity9am Humidity (percent) at 9am

Humidity3pm Humidity (percent) at 3pm

Pressure9am Atmospheric pressure (hpa) reduced to mean sea level at 9am

Pressure3pm Atmospheric pressure (hpa) reduced to mean sea level at 3pm

Cloud9am Fraction of sky obscured by cloud at 9am.

Cloud3pm Fraction of sky obscured by cloud at 3pm.

Temp9am Temperature (degrees C) at 9am

Temp3pm Temperature (degrees C) at 3pm.

**8. SOURCE CODE FOR PREDICTING RAINFALL**

import numpy as np

import pandas as pd

import os

import missingno as msno

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

from scipy import stats

from sklearn.linear\_model import LogisticRegression

from imblearn.over\_sampling import SMOTE

from collections import Counter

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score, f1\_score

from xgboost import XGBClassifier

from sklearn.ensemble import RandomForestRegressor

from sklearn.naive\_bayes import BernoulliNB

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestRegressor

import warnings

warnings.filterwarnings("ignore")

rain = pd.read\_csv('/content/weatherAUS.csv')

rain.head(10)

print(f'The number of rows are {rain.shape[0] } and the number of columns are {rain.shape[1]}')

rain.info()

categorical\_col, contin\_val=[],[]

for i in rain.columns:

if rain[i].dtype == 'object':

categorical\_col.append(i)

else:

contin\_val.append(i)

print(categorical\_col)

print(contin\_val)

rain.nunique()

rain.isnull().sum()

msno.matrix(rain)

msno.bar(rain, sort='ascending')

msno.heatmap(rain)

plt.figure(figsize=(17,15))

ax = sns.heatmap(rain.corr(), square=True, annot=True, fmt='.2f')

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=90)

plt.show()

rain['RainTomorrow'] = rain['RainTomorrow'].map({'Yes': 1, 'No': 0})

rain['RainToday'] = rain['RainToday'].map({'Yes': 1, 'No': 0})

print(rain.RainToday)

print(rain.RainTomorrow)

#Checking percentage of missing data in every column

(rain.isnull().sum()/len(rain))\*100

for label,content in rain.items():

if pd.api.types.is\_numeric\_dtype(content):

if pd.isnull(content).sum():

print(label)

for label,content in rain.items():

if pd.api.types.is\_numeric\_dtype(content):

if pd.isnull(content).sum():

rain[label+"\_is\_missing"]=pd.isnull(content)

rain[label]=content.fillna(content.median())

rain.head()

fig, ax =plt.subplots(1,2)

print(rain.RainToday.value\_counts())

print(rain.RainTomorrow.value\_counts())

plt.figure(figsize=(20,20))

sns.countplot(data=rain,x='RainToday',ax=ax[0])

sns.countplot(data=rain,x='RainTomorrow',ax=ax[1])

fig, ax =plt.subplots(3,1)

plt.figure(figsize=(10,10))

sns.countplot(data=rain,x='WindDir9am',ax=ax[0])

sns.countplot(data=rain,x='WindDir3pm',ax=ax[1])

sns.countplot(data=rain,x='WindGustDir',ax=ax[2])

fig.tight\_layout()

#Dropping date column

rain=rain.iloc[:,1:]

rain

le = preprocessing.LabelEncoder()

rain['WindDir9am'] = le.fit\_transform(rain['WindDir9am'])

rain['WindDir3pm'] = le.fit\_transform(rain['WindDir3pm'])

rain['WindGustDir'] = le.fit\_transform(rain['WindGustDir'])

rain.head()

fig, ax =plt.subplots(2,1)

plt.figure(figsize=(10,10))

sns.boxplot(rain['Pressure3pm'],orient='v',color='c',ax=ax[0])

sns.boxplot(rain['Pressure9am'],orient='v',color='c',ax=ax[1])

fig.tight\_layout()

sns.violinplot(x='RainToday',y='MaxTemp',data=rain,hue='RainTomorrow')

sns.violinplot(x='RainToday',y='MinTemp',data=rain,hue='RainTomorrow')

X1=rain[['MinTemp','MaxTemp','Rainfall','WindSpeed9am','Humidity9am','Pressure9am']]

Y1=rain['RainTomorrow']

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()

rf\_model = rf.fit(X1,Y1)

print(rf\_model)

pip install gradio

import gradio as gr

def rain(MinTemp,MaxTemp,Rainfall,WindSpeed9am,Humidity9am,Pressure9am):

x=np.array([MinTemp,MaxTemp,Rainfall,WindSpeed9am,Humidity9am,Pressure9am])

ypred1= rf\_model.predict x.reshape(1,-1))

if ypred1<0.5:

return "LESS POSSIBLE"

else:

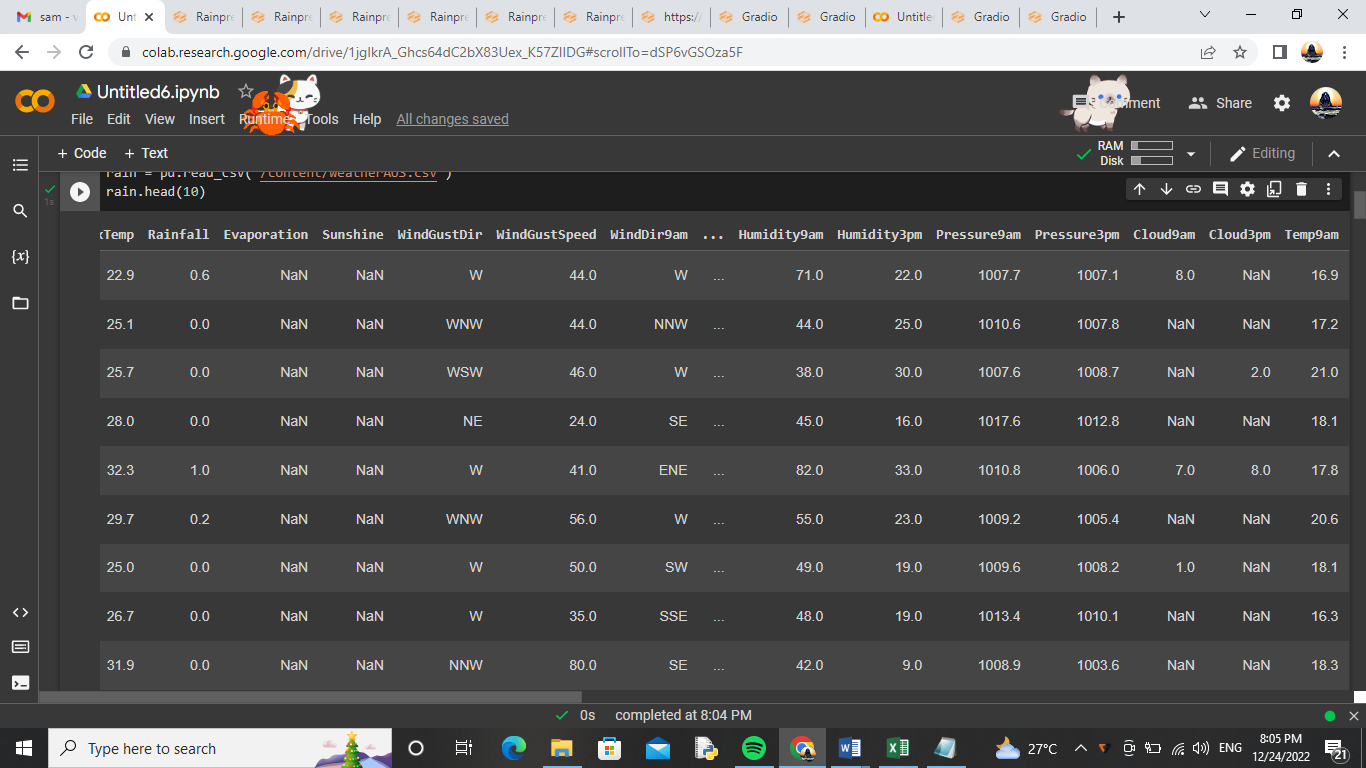
return "THERE IS A CHANCE OF RAINFALL"

outputs = gr.outputs.Textbox()

app=gr.Interface(fn=rain,inputs=['number','number','number','number','number','number'],outputs=outputs,description="\*MinTemp\* in degree celsius,\*Maxtemp\* in degrees celsius,\*Rainfall\* in mm, \*WindSpeed9am\* day average over 10 mins,\*Humidity9am\* at percent,\*Pressure9am\*pressure (hpa) reduced to mean sea level ",title="Rainprediction",examples=[[13.4,22.9,0.6,20,71,1007.7],[13.5,22.9,16.8,6,80,1005.8]])

app.launch()

**9. OUTPUT**



**Fig 2 :** **Rainfall dataset used for prediction**

* This is a dataset of rainfall taken from Kaggle
* The dataset contains MinTemp,MaxTemp,Rainfall,WindSpeed9am,Humidity9am and Pressure9am

# 

# Fig 3 : Visualizing the missing values

# This graph represents the missing values for the rain

# The black colour represents the presence of value in the given dataset

# 

# ****Fig 4 :**** Sorting of rain

# This graph represents the ascending for the missing values rain graph .

# 

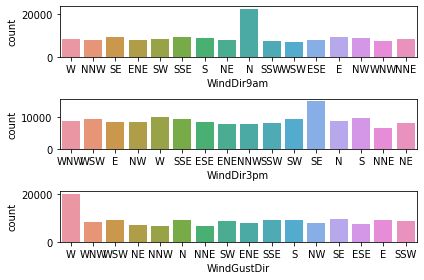
# 

# Fig 5 : Correlation heatmap of 16 attributes

# This graph shows the correlation heatmap of Mintemp , Maxtemp,Rainfall,Sunshine,Evaporation,Windgust speed,Windgust dir etc.,

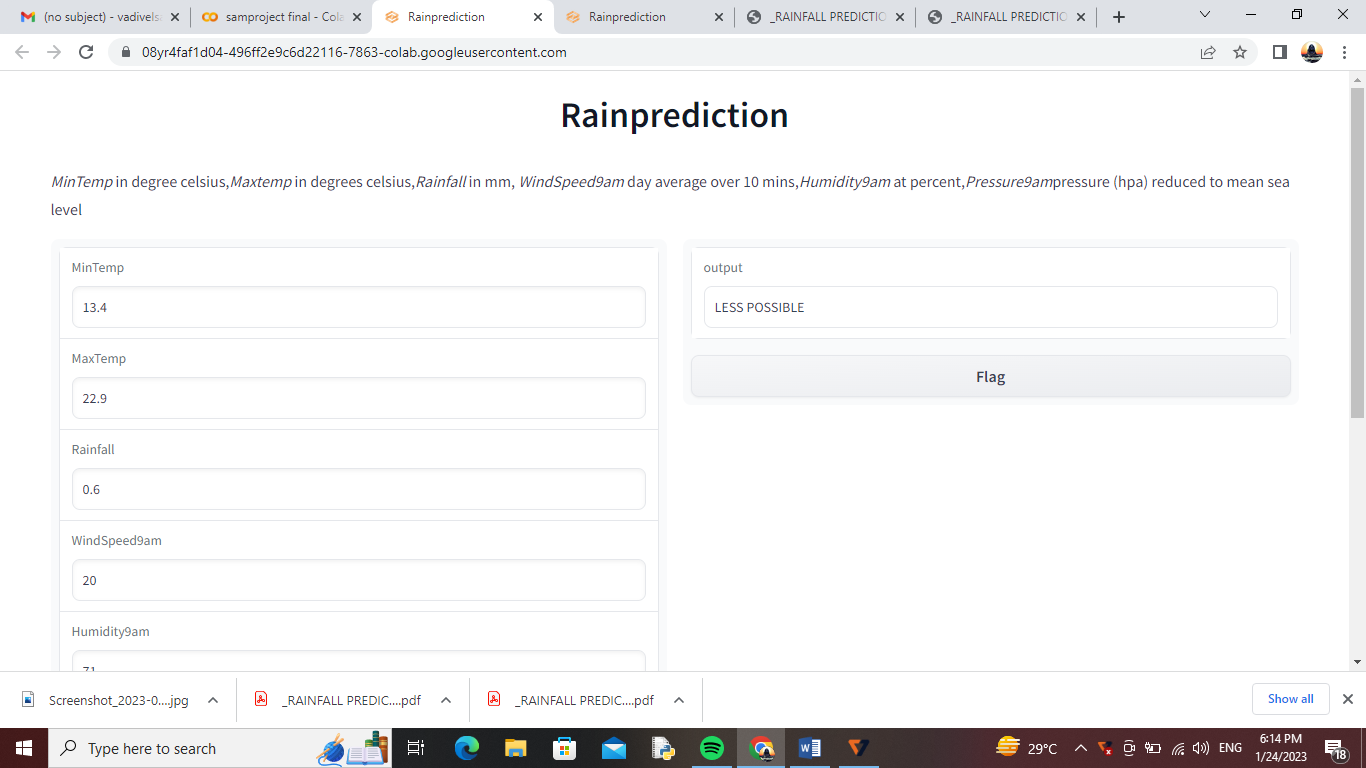
# The darkest colour has the highest relation between the attributes

# The light colur has the lowest relation between the attributes



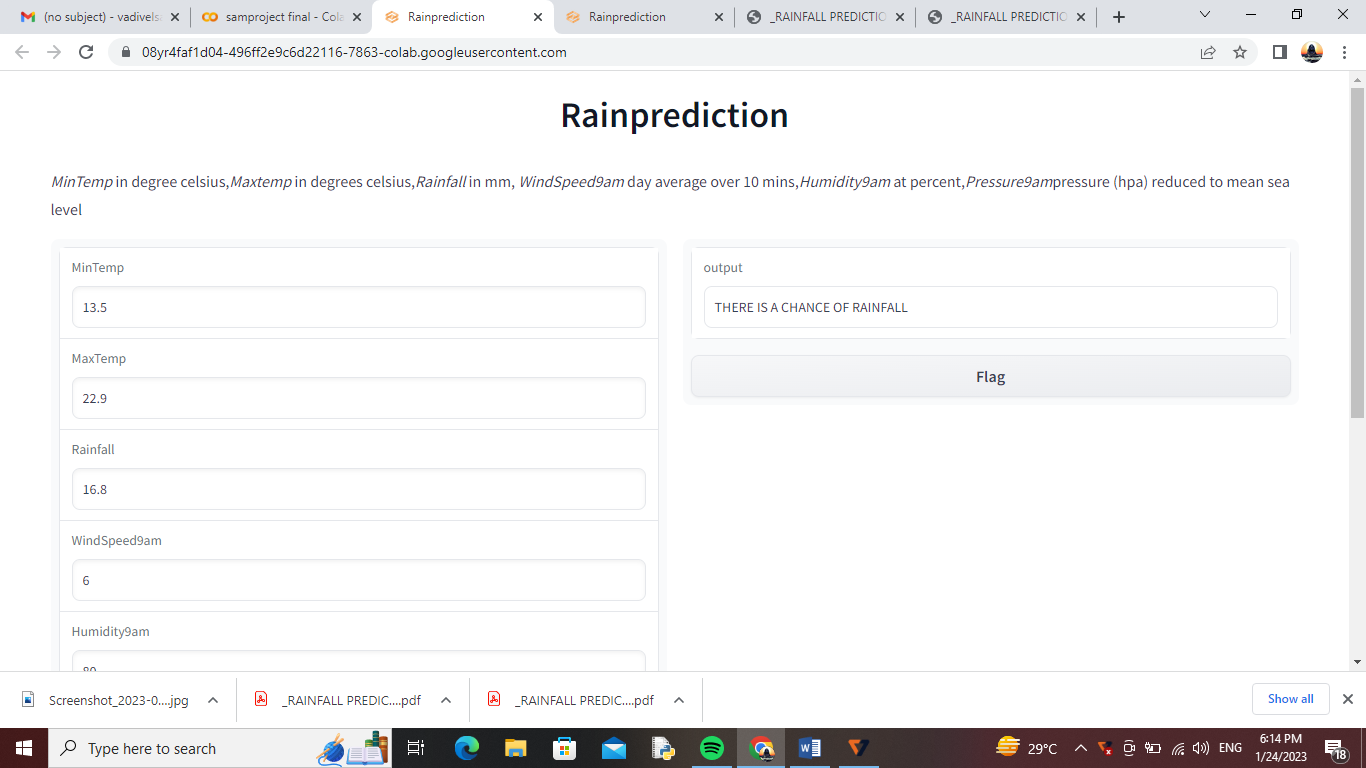
**Fig 6 : Highest direction of the wind**

* This figure represents the direction of wind flowing at 9am and 3pm
* Peach colour in the figure represents the West , Green colour in the figure represents the North , Pale green in the figure represents the South and the Purple colour in the figure represents the East and the remaining colours represnts the flow of directions in the poles



**Fig 7 : Prediction from user’s input**

* This is gradio’s GUI , user should give the needed inputs and they can get their results whether there is a possibility of getting Rainfall or not
* This picture represents the **Less possibility of Rainfall**

****

**Fig 8 :** **Prediction from user’s input**

* This is gradio’s GUI , user should give the needed inputs and they can get their results whether there is a possibility of getting Rainfall or not
* This picture represents the **Less possibility of Rainfall**

**10. CONCLUSION**

For predicting rainfall, here this project uses three boosting techniques and one artificial neural network using forward and back propagation.. After the training dataset have been trained, they have tested it by predicting some unseen day’s temperature and found accurate results. It is more suited to predict the tomorrow’s rainfall.

**11. RESULT**

We have successfully predicted the possibility of getting Rainfall with Linear regression algorithm using Python code in google colaboratory.