Innovative Work in Mathematical Modelling and Simulation
Delhi Technological University

MATHEMATICAL M0DEL F0R FL00D ANALYSIS/ MANAGEMENT AL0NG INDIA's C0ASTLINE

Submitted to :
Dr.Nilam

Submitted by :
Sadhvi Mehra
2k19/MSCMAT/13
Nikita Kaushik
2k19/MSCMAT/33

ABSTRACT

India is exceptionally defenseless, as the majority of its topographical territory is inclined to yearly flooding. The high misfortunes and harms because of floods show the helpless variation and alleviation status of India and deficiency in calamity the board and readiness. It's important to take proper measures in order to avoid the disastrous outcomes of the flooding. It is important to minimise the L0SS 0f life and the loss of infrastructure. Both these factors affect the country and make it difficult for it's revival. So, for this it is essential to be prepared beforehand. Therefore, in this project we approach flood analysis and management along India's coastline with the help of a mathematical model.

ACKN0WLEDGEMENT

I wish to offer my earnest thanks to my Mathematical M0deling teacher Dr. Nilam whose important direction and proposals have been a significant supporter towards the culmination of the task.

I might want to thank my folks and my companions who have helped me with their significant proposals and have been useful in different periods of the fulfillment of this undertaking.

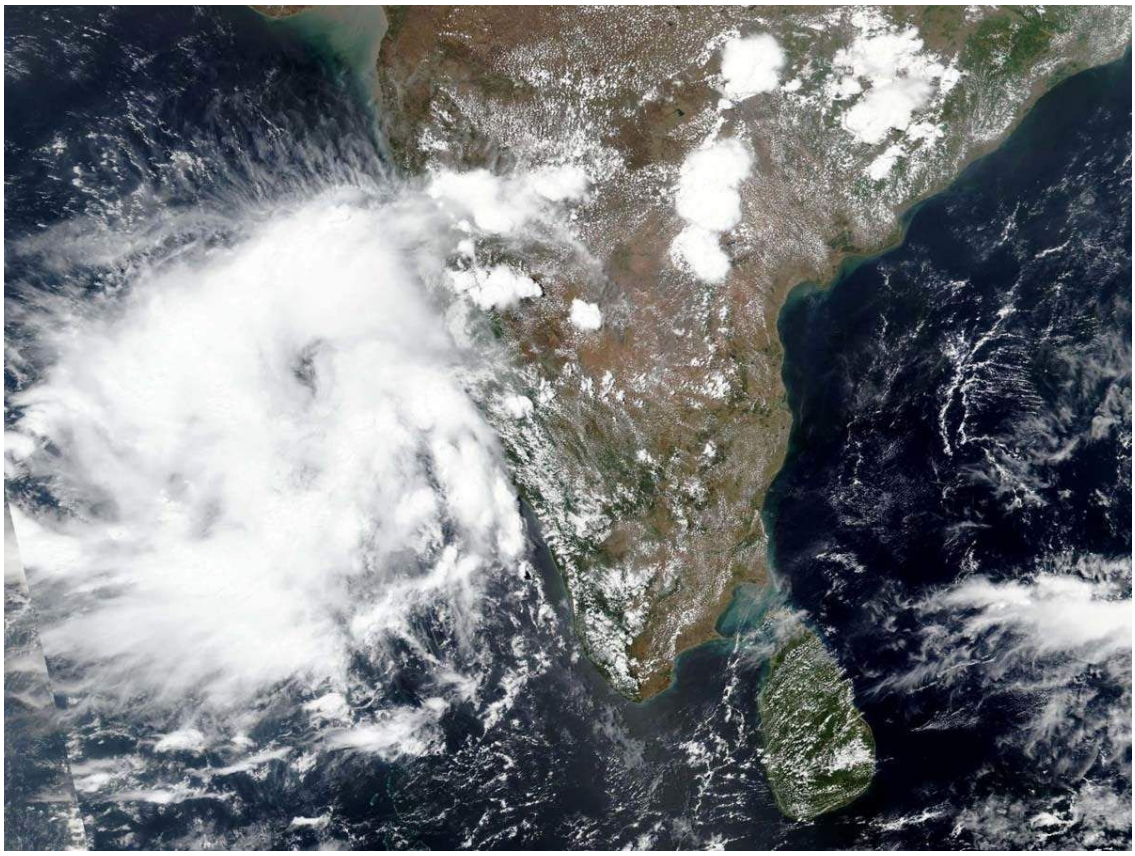
Last yet not the least I might want to thank my schoolmates who have consistently upheld and helped me a great deal.

TABLE 0F C0NTENTS

INTR0DUCTION [1]

Our project revolves around flood plone areas majorly across Indian coastline.The first thing that comes up in our mind after listening to the word coast is basically sea that is the areas in the vicinity of sea i.e a coastline: sea separated by a mass of land.So we begin by defining C0ASTAL FL00DING .

A seaside flood is the point at which the coast is flooded by the sea. A flood begins when waves move inland on an undefended drift or up and over or penetrate the waterfront safeguard works like hills and dikes.The causes incorporate ocean level ascent, topography, and quick seaside populace development joined by fast increment of human exercises that meddle with normal cycles.



Reclaimed Land: This is land that has been picked up from the ocean because of beach front administration. This land is low lying and level, so a little ascent in ocean level from a gentle tempest flood is sufficient to flood it and cause broad harm.

For instance, recovered land is involved by most of Mumbai populace making it powerless against seaside flooding.

Urbanization: The insufficient seepage framework and an incapable spatial arranging increment the flood risk.India's seaside areas, home to around 170 million of the nation's 1.4 billion individuals, are on the forefronts of a moving atmosphere, encountering ocean level ascent, disintegration, and cataclysmic events, for example, typhoons and twisters.

The most recent proof of this weakness happened in May 2020, as the most grounded storm recorded in a long time in the Bay of Bengal—Cyclone Amphan—hit, driving a few million individuals to clear.

It is important to be aware of the upcoming disasters and also to be prepared beforehand. This is necessary since it affects our lives in numerous ways. It costs us lives, infrastructure etc. A lot of the funds that are needed to be spent in the disaster hit areas can be avoided if there is proper palling and awareness. These funds can then be used for the actual growth and development of the areas.

As stated earlier Loss of lives and property are one of the major impacts of flood. Some others include Economic loss, Environmental impacts, coastal erosion, agricultural impact and social impact.

Natural Disasters can only be avoided with proper planning and implementation. Taking the case of Japan, it is hit with disastrous earthquakes at regular intervals. However planned urbanisation in Japan can withstand disasters which has proven to be a success in managing the impacts.

FL00D M0DELING AS A C0MPARTMENT M0DEL [2]

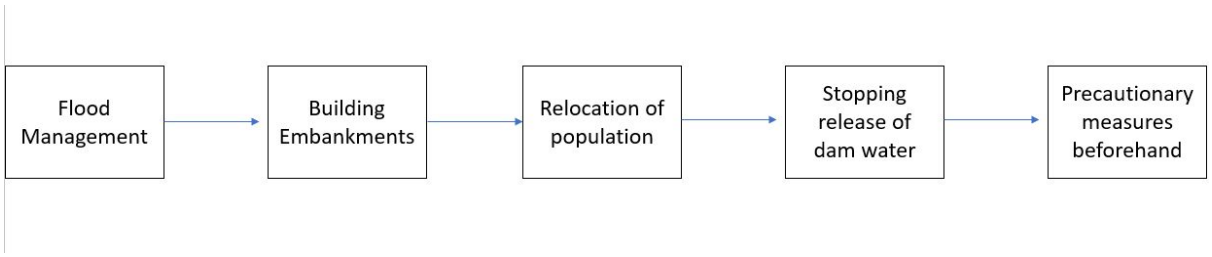
Compartment models are numerical models where every one of the state factors communicates a particular property of some piece of a framework, and these parts are alluded to as the compartments of the model. For the most part, this mirrors the presumption that the property of the compartment communicated by the state variable is homogeneously conveyed inside the compartment.

We have formed C0MPARTMENT M0DEL for the propest flood model as shown below. The first compartment model is of Flood analysis. The second compartment model is of flood management.

These are as follows -

The factors that will be considered in flood analysis will be Rainfall data we extracted, based upon which our analysis will be done.

Now the second compartment model is of flood management.



Embankments as shown in the figure

The first compartment consists of flood management. The measures that must be taken in consideration are given in the second and consecutive compartments. These include building of Embankments. This prevents the flood water from entering.

Other compartments include relocation of population, Stopping the release of dam water and adopting precautionary measures beforehand.

Before moving on to the conceptual, physical and logical model we take into consideration the **assumptions** for the proposed model.

ASSUMPTIONS

1.There doesn't occur any other natural disaster or calamity,other than flood during this period.

Here we have taken into consideration the fact that the time period during which we will obtain the data no other natural disaster other than flood will occur.

2.The state variables need to be independent i.e there can be little or no multicollinearity between independent variables i.e independent variables should not be too highly correlated with each other.

3.Being an instationary model,we assume all the sources don't get exhausted with time.

Instationary model as we know refers to a mathematical model that is independent of time.So here we have assumed that our model is an instationary model.

4.Since it is a Mechanistic model ,we presume the data to be exact and correct.

A Mechanistic model accepts that a perplexing framework can be perceived by looking at the functions of its individual parts and the way in which they are

coupled.

5.The number of deaths we take into account are only the ones caused by flood and not the natural deaths.Here we only take into consideration the deaths that are caused due to flooding.

Now we move on to the Conceptual, logical and physical model of the proposed flood modelling.

Fl00D management as a Conceptual,Logical and Physical model

C0NCEPTUAL M0DEL - An applied model which is a portrayal of a framework, made of the synthesis of ideas which are utilized to help individuals know, comprehend, or mimic a subject the model speaks to.

A theoretical model, when actualized appropriately, ought to fulfill four basic destinations.

1. Improve a person's comprehension of the delegate framework.
2. Encourage effective transport of framework subtleties between partners.
3. Give a perspective to framework architects to extricate framework particulars
4. Record the framework for future reference and give a way to cooperation.

This Model defines WHAT the system contains and what problem is to be worked upon. The question to be worked upon is **what to do to control flood and take preventive flood measures at best possible time before the repercussions turn hazardous.**

PR0P0SED S0LUTI0N AS L0GISTIC M0DEL -

In our model we make use of logistic regression .A dataset with the measure of precipitation and if a FL00D had happened in a specific region/state/city, in the earlier years, will be utilized. The dataset will have the precipitation information for a term of a year approx .

We likewise consider different components that take this normal information of precipitation, as contribution to our AI m0del and in the event that it causes a fl00d or not as the yield marks. We train 0ur m0del and test it.

We take this normal information of precipitation, as contribution to our AI model and on the off chance that it causes a flood 0r not as the yield marks.

We train 0ur m0del and save it. The state variable for the proposed numerical model are-

Annual rainfall-Annual rainfall or precipitation is the sum of rainfall in a year. Yearly precipitation is the n0rmal measure of absolute downpour that a sp0t for the most part gets.

When I say annual rainfall of my place is x mm, it does not imply that for a particular year the total rainfall my place received was x mm rather it means in general or on an average my place receives x mm of rainfall annually.

PR0BLEM STATEMENT:

Disaster prevention and Fl00D prediction using machine learning approach.

APPR0ACH : Given the input data, we give this data as an input, and let the model predict, if there is a possibility of flooding or not, by setting some threshold in the training data. Our basic approach for this problem is binary classification, using basic machine learning algorithms.

L0GISTIC REGRESSI0N

It is a prescient calculation utilizing free factors to foresee the reliant variable, much the same as Linear Regression, however with a distinction that the needy variable ought to be absolute variable.Independent factors can be numeric or clear cut factors, yet the needy variable will consistently be unmitigated.

Strategic relapse is a factual model that utilizes Logistic capacity to demonstrate the restrictive probability.For twofold relapse, we ascertain the contingent likelihood of the needy variable Y, given free factor X.It can be composed as P(Y=1|X) or P(Y=0|X)

This is perused as the restrictive likelihood of Y=1, given X or contingent likelihood of Y=0, given X.P(Y |X) is approximated as a sigmoid capacity applied to a direct blend of information highlights

L0GISTIC REGRESSI0N FUNCTI0N

Logistic regression uses logit function, also referred to as log-odds; it is the logarithm of odds. The odds ratio is the ratio of odds of an event A in the presence of the event B and the odds of event A in the absence of event B.

$$\ln\left(\frac{P}{1-P}\right) = \theta_1 + \theta_2 x + \mathrm{e}$$

$$\frac{P}{1-P} = \mathrm{e}^{\theta_1+\theta_2 x+\mathrm{e}}$$

$$P = \frac{1}{1+\mathrm{e}^{-}}(\theta_1 + \theta_2 x)$$

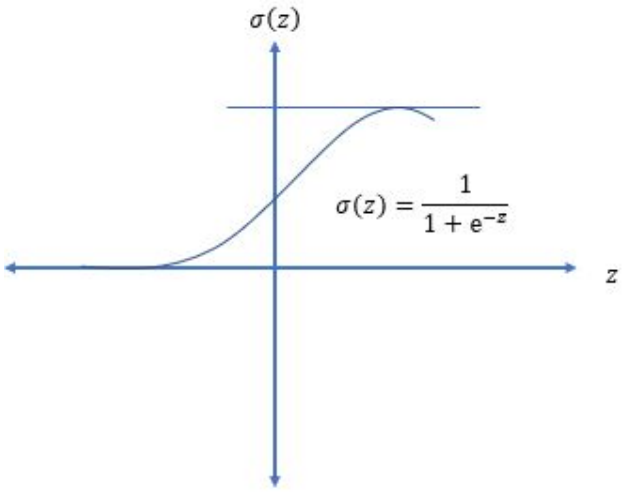
$$\sigma(z) = \frac{1}{1+\mathrm{e}^{-z}} \quad \text{Where} \quad z = \theta^T x$$

$$\theta^T \mathbf{x} = \sum_{i=1}^{m} \theta_i x_i = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m$$

logit or logistic function

- P is the probability that event Y occurs. P(Y=1)
- P/(1-P) is the odds ratio
- θ is a parameters of length m

Logit function estimates probabilities between 0 and 1, and hence logistic regression is a non-linear transformation that looks like S- function shown below.



In [42]:

```
import random
random.seed(10)
import numpy as np#linear Algebra
import pandas as pd#data processing
data=pd.read_csv('C:/Users/DELL/DESKTOP/kerala.csv')
print(data)
```

```
    SUBDIVISION  YEAR   JAN   FEB   MAR    APR    MAY     JUN     JUL     AUG  \
0        KERALA  1901  28.7  44.7  51.6  160.0  174.7   824.6   743.0   357.5
1        KERALA  1902   6.7   2.6  57.3   83.9  134.5   390.9  1205.0   315.8
2        KERALA  1903   3.2  18.6   3.1   83.6  249.7   558.6  1022.5   420.2
3        KERALA  1904  23.7   3.0  32.2   71.5  235.7  1098.2   725.5   351.8
4        KERALA  1905   1.2  22.3   9.4  105.9  263.3   850.2   520.5   293.6
..          ...   ...   ...   ...   ...    ...    ...     ...     ...     ...
113      KERALA  2014   4.6  10.3  17.9   95.7  251.0   454.4   677.8   733.9
114      KERALA  2015   3.1   5.8  50.1  214.1  201.8   563.6   406.0   252.2
115      KERALA  2016   2.4   3.8  35.9  143.0  186.4   522.2   412.3   325.5
116     KERALA   2017   1.9   6.8   8.9   43.6  173.5   498.5   319.6   531.8
117      KERALA  2018  29.1  52.1  48.6  116.4  183.8   625.4  1048.5  1398.9

       SEP    OCT    NOV    DEC   ANNUAL RAINFALL FLOODS
0    197.7  266.9  350.8   48.4            3248.6    YES
1    491.6  358.4  158.3  121.5            3326.6    YES
2    341.8  354.1  157.0   59.0            3271.2    YES
3    222.7  328.1   33.9    3.3            3129.7    YES
4    217.2  383.5   74.4    0.2            2741.6     NO
..     ...    ...    ...    ...               ...    ...
113  298.8  355.5   99.5   47.2            3046.4    YES
114  292.9  308.1  223.6   79.4            2600.6     NO
115  173.2  225.9  125.4   23.6            2176.6     NO
116  209.5  192.4   92.5   38.1            2117.1     NO
117  423.6  356.1  125.4   65.1            4473.0    YES

[118 rows x 16 columns]
```

In [43]:

```
data.head()
```

Out[43]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FLOODS
0	KERALA	1901	28.7	44.7	51.6	160.0	174.7	824.6	743.0	357.5	197.7	266.9	350.8	48.4	3248.6	YES
1	KERALA	1902	6.7	2.6	57.3	83.9	134.5	390.9	1205.0	315.8	491.6	358.4	158.3	121.5	3326.6	YES
2	KERALA	1903	3.2	18.6	3.1	83.6	249.7	558.6	1022.5	420.2	341.8	354.1	157.0	59.0	3271.2	YES
3	KERALA	1904	23.7	3.0	32.2	71.5	235.7	1098.2	725.5	351.8	222.7	328.1	33.9	3.3	3129.7	YES
4	KERALA	1905	1.2	22.3	9.4	105.9	263.3	850.2	520.5	293.6	217.2	383.5	74.4	0.2	2741.6	NO

In [44]:

```
data.tail()
```

Out[44]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FLOODS
113	KERALA	2014	4.6	10.3	17.9	95.7	251.0	454.4	677.8	733.9	298.8	355.5	99.5	47.2	3046.4	YES
114	KERALA	2015	3.1	5.8	50.1	214.1	201.8	563.6	406.0	252.2	292.9	308.1	223.6	79.4	2600.6	NO
115	KERALA	2016	2.4	3.8	35.9	143.0	186.4	522.2	412.3	325.5	173.2	225.9	125.4	23.6	2176.6	NO
116	KERALA	2017	1.9	6.8	8.9	43.6	173.5	498.5	319.6	531.8	209.5	192.4	92.5	38.1	2117.1	NO
117	KERALA	2018	29.1	52.1	48.6	116.4	183.8	625.4	1048.5	1398.9	423.6	356.1	125.4	65.1	4473.0	YES

ANNUAL

FINDING THE NUMBER OF MISSING VALUES

In [45]:

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

Out[45]:

```
SUBDIVISION          0
YEAR                 0
JAN                  0
FEB                  0
MAR                  0
APR                  0
MAY                  0
JUN                  0
JUL                  0
AUG                  0
SEP                  0
OCT                  0
NOV                  0
DEC                  0
 ANNUAL RAINFALL     0
FLOODS               0
dtype: int64
```

In [46]:

```
data.describe()
```

Out[46]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	
count	118.000000	118.000000	118.000000	118.000000	118.000000	118.000000	118.000000	118.000000	118.000000	118.000000	1
mean	1959.500000	12.218644	15.633898	36.670339	110.330508	228.644915	651.617797	698.220339	430.369492	246.207627	2
std	34.207699	15.473766	16.406290	30.063862	44.633452	147.548778	186.181363	228.988966	181.980463	121.901131	
min	1901.000000	0.000000	0.000000	0.100000	13.100000	53.400000	196.800000	167.500000	178.600000	41.300000	
25%	1930.250000	2.175000	4.700000	18.100000	74.350000	125.050000	535.550000	533.200000	316.725000	155.425000	2
50%	1959.500000	5.800000	8.350000	28.400000	110.400000	184.600000	625.600000	691.650000	386.250000	223.550000	2
75%	1988.750000	18.175000	21.400000	49.825000	136.450000	264.875000	786.975000	832.425000	500.100000	334.500000	3
max	2018.000000	83.500000	79.000000	217.200000	238.000000	738.800000	1098.200000	1526.500000	1398.900000	526.700000	5

In [47]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118 entries, 0 to 117
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   SUBDIVISION       118 non-null    object
 1   YEAR              118 non-null    int64
 2   JAN               118 non-null    float64
 3   FEB               118 non-null    float64
 4   MAR               118 non-null    float64
 5   APR               118 non-null    float64
 6   MAY               118 non-null    float64
 7   JUN               118 non-null    float64
 8   JUL               118 non-null    float64
 9   AUG               118 non-null    float64
 10  SEP               118 non-null    float64
 11  OCT               118 non-null    float64
 12  NOV               118 non-null    float64
 13  DEC               118 non-null    float64
```

```
 15  DEC                118 non-null    float64
 14   ANNUAL RAINFALL   118 non-null    float64
 15  FLOODS             118 non-null    object
dtypes: float64(13), int64(1), object(2)
memory usage: 14.9+ KB
```

In [48]:

```
#replacing the yes/no in flood coloumn by 1/0
data['FLOODS'].replace(['YES','NO'],[1,0],inplace=True)
```

In [49]:

```
data.head()
```

Out[49]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL RAINFALL	FLOODS
0	KERALA	1901	28.7	44.7	51.6	160.0	174.7	824.6	743.0	357.5	197.7	266.9	350.8	48.4	3248.6	1
1	KERALA	1902	6.7	2.6	57.3	83.9	134.5	390.9	1205.0	315.8	491.6	358.4	158.3	121.5	3326.6	1
2	KERALA	1903	3.2	18.6	3.1	83.6	249.7	558.6	1022.5	420.2	341.8	354.1	157.0	59.0	3271.2	1
3	KERALA	1904	23.7	3.0	32.2	71.5	235.7	1098.2	725.5	351.8	222.7	328.1	33.9	3.3	3129.7	1
4	KERALA	1905	1.2	22.3	9.4	105.9	263.3	850.2	520.5	293.6	217.2	383.5	74.4	0.2	2741.6	0

In [50]:

```
#seperating x and y
x=data.iloc[:,1:14]
x.head()
```

Out[50]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
0	1901	28.7	44.7	51.6	160.0	174.7	824.6	743.0	357.5	197.7	266.9	350.8	48.4
1	1902	6.7	2.6	57.3	83.9	134.5	390.9	1205.0	315.8	491.6	358.4	158.3	121.5
2	1903	3.2	18.6	3.1	83.6	249.7	558.6	1022.5	420.2	341.8	354.1	157.0	59.0
3	1904	23.7	3.0	32.2	71.5	235.7	1098.2	725.5	351.8	222.7	328.1	33.9	3.3
4	1905	1.2	22.3	9.4	105.9	263.3	850.2	520.5	293.6	217.2	383.5	74.4	0.2

In [51]:

```
y=data.iloc[:,-1]
y
```

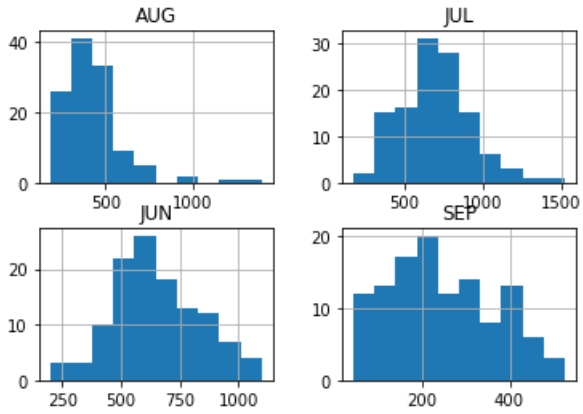
Out[51]:

```
0      1
1      1
2      1
3      1
4      0
      ..
113    1
114    0
115    0
116    0
117    1
Name: FLOODS, Length: 118, dtype: int64
```

PLOTTING THE DATA

In [52]:

```
import matplotlib.pyplot as plt
#set the backend of matplotlib to inline backend
%matplotlib inline
c=data[['JUN','JUL','AUG','SEP']]
c.hist()
plt.show()
#how the rainfall index vary each season
```



In [53]:

```
ax=data[['JAN','FEB','MAR','APR','MAY','JUN','JUL','AUG','SEP','OCT','NOV','DEC']].mean()
print(ax)
```

```
JAN     12.218644
FEB     15.633898
MAR     36.670339
APR    110.330508
MAY    228.644915
JUN    651.617797
JUL    698.220339
AUG    430.369492
SEP    246.207627
OCT    293.207627
NOV    162.311017
DEC     40.009322
dtype: float64
```

USING SKLEARN TO DEVELOP THE ML MODEL

In [54]:

```
from sklearn import preprocessing
minmax=preprocessing.MinMaxScaler(feature_range=(0,1))
minmax.fit(x).transform(x)
```

Out[54]:

```
array([[0.        , 0.34371257, 0.56582278, ..., 0.39727673, 0.95570189,
        0.2388724 ],
       [0.00854701, 0.08023952, 0.03291139, ..., 0.5804966 , 0.37952709,
        0.60039565],
       [0.01709402, 0.03832335, 0.23544304, ..., 0.57188626, 0.37563604,
        0.29129575],
       ...,
       [0.98290598, 0.02874251, 0.04810127, ..., 0.31517821, 0.28105358,
        0.11622156],
       [0.99145299, 0.02275449, 0.08607595, ..., 0.24809772, 0.18258007,
        0.18793274],
       [1.        , 0.34850299, 0.65949367, ..., 0.57589107, 0.28105358,
        0.3214639 ]])
```

In [55]:

```
from sklearn import model_selection,neighbors
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
x_train
```

Out[55]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
41	1942	2.4	4.7	23.2	180.3	191.9	813.6	828.8	329.3	99.8	374.4	84.7	117.9
3	1904	23.7	3.0	32.2	71.5	235.7	1098.2	725.5	351.8	222.7	328.1	33.9	3.3
116	2017	1.9	6.8	8.9	43.6	173.5	498.5	319.6	531.8	209.5	192.4	92.5	38.1
110	2011	20.5	45.7	24.1	165.2	124.2	788.5	536.8	492.7	391.2	227.2	169.7	49.5
49	1950	0.1	53.7	31.1	68.5	242.0	638.3	905.7	387.3	411.6	250.4	149.2	8.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...
94	1995	10.3	6.5	37.3	134.9	355.6	493.4	702.5	457.3	280.0	198.3	182.6	0.1
62	1963	30.2	24.8	69.8	96.3	157.1	393.3	720.2	511.0	223.9	282.6	93.4	48.4
28	1929	12.8	29.8	58.9	210.7	148.0	946.6	844.0	293.9	268.9	350.4	158.2	39.4
33	1934	74.5	1.7	47.7	92.4	106.7	852.9	415.0	337.2	48.4	335.9	93.4	4.9
56	1957	1.2	16.0	25.7	70.2	381.2	872.0	835.3	358.8	41.3	280.1	192.5	28.9

94 rows × 13 columns

In [26]:

```
x_test
```

Out[26]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
18	1919	43.0	6.1	33.9	65.9	247.0	636.8	648.0	484.2	255.9	249.2	280.1	53.0
41	1942	2.4	4.7	23.2	180.3	191.9	813.6	828.8	329.3	99.8	374.4	84.7	117.9
49	1950	0.1	53.7	31.1	68.5	242.0	638.3	905.7	387.3	411.6	250.4	149.2	8.8
13	1914	0.7	6.8	18.1	32.7	164.2	565.3	857.7	402.2	241.0	374.4	100.9	135.2
19	1920	35.2	5.5	24.1	172.0	87.7	964.3	940.8	235.0	178.0	350.1	302.3	8.2
28	1929	12.8	29.8	58.9	210.7	148.0	946.6	844.0	293.9	268.9	350.4	158.2	39.4
108	2009	3.3	1.5	62.6	69.0	191.6	438.2	924.9	269.3	326.5	205.2	274.4	44.2
101	2002	4.7	8.7	35.7	117.3	330.8	503.1	318.7	438.2	99.0	511.7	137.5	2.1
16	1917	2.9	47.6	79.4	38.1	122.9	703.7	342.7	335.1	470.3	264.1	256.4	41.6
37	1938	0.3	79.0	53.3	164.5	179.6	681.6	648.6	287.9	223.2	223.7	69.5	22.9
71	1972	2.6	7.5	2.5	87.5	436.0	401.8	714.4	294.9	185.7	351.5	140.5	114.3
38	1939	13.6	3.6	24.9	172.8	105.1	625.8	749.6	459.9	134.1	339.8	298.1	10.2
63	1964	1.1	7.8	67.2	83.3	94.8	379.4	754.2	548.0	398.2	325.7	191.7	17.8
109	2010	18.6	1.0	31.4	138.9	190.6	667.5	629.0	356.0	275.6	441.4	335.1	46.8
99	2000	11.7	57.8	21.5	96.3	124.5	633.8	343.2	566.5	195.8	214.2	78.1	69.1
0	1901	28.7	44.7	51.6	160.0	174.7	824.6	743.0	357.5	197.7	266.9	350.8	48.4
50	1951	6.6	6.5	41.6	175.9	148.5	774.1	544.6	190.6	313.8	250.6	229.6	23.2
32	1933	1.0	9.3	36.9	139.5	738.8	859.3	773.4	479.5	469.7	397.0	126.1	42.3
10	1911	3.0	4.3	18.2	51.0	180.6	990.0	705.3	178.6	60.2	302.3	145.7	87.6
89	1990	14.9	4.8	18.0	41.8	488.5	528.6	635.4	370.8	103.3	323.2	158.8	5.2
74	1975	5.2	21.4	63.4	123.8	162.2	864.4	531.3	675.9	457.7	368.9	204.3	19.9
115	2016	2.4	3.8	35.9	143.0	186.4	522.2	412.3	325.5	173.2	225.9	125.4	23.6
58	1959	3.0	21.4	6.3	150.7	347.2	872.8	1155.7	397.3	405.5	200.4	151.9	34.0
33	1934	74.5	1.7	47.7	92.4	106.7	852.9	415.0	337.2	48.4	335.9	93.4	4.9

In [58]:

```
y_train=y_train.astype('int')
y_train
```

Out[58]:

```
41     1
3      1
116    0
110    1
49     1
      ..
94     0
62     0
28     1
33     0
56     1
Name: FLOODS, Length: 94, dtype: int32
```

In [59]:

```
y_test=y_test.astype('int')
y_test
```

Out[59]:

```
111    0
4      0
23     1
12     0
11     1
106    1
82     0
26     1
7      0
78     0
10     0
20     0
98     0
59     1
42     1
38     1
46     1
29     1
89     0
36     0
22     1
0      1
65     0
100    0
Name: FLOODS, dtype: int32
```

LOGISTIC REGRESSION TO PREDICT

In [60]:

```
x_train_std=minmax.fit_transform(x_train)
print(x_train_std)
```

```
[[0.34482759 0.03221477 0.05949367 ... 0.64440699 0.15923376 0.58259149]
 [0.01724138 0.31812081 0.03797468 ... 0.54687171 0.00718348 0.01582591]
 [0.99137931 0.02550336 0.08607595 ... 0.26100695 0.18258007 0.18793274]
 ...
 [0.23275862 0.17181208 0.37721519 ... 0.59384875 0.37922778 0.19436202]
 [0.27586207 1.         0.02151899 ... 0.56330314 0.18527387 0.02373887]
 [0.47413793 0.01610738 0.20253165 ... 0.44575521 0.48189165 0.14243323]]
```

In [61]:

```
from sklearn.model_selection import cross_val_score,cross_val_predict
from sklearn.linear_model import LogisticRegression
```

In [62]:

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr_acc=cross_val_score(lr,x_train_std,y_train,cv=3,scoring='accuracy',n_jobs=-1)
lr_proba=cross_val_predict(lr,x_train_std,y_train,cv=3,method='predict_proba')
```

```
D:\VS\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [63]:

```
lr_acc
```

Out[63]:

```
array([0.8125    , 0.83870968, 0.80645161])
```

In [64]:

```
lr_proba
```

Out[64]:

```
array([[0.5897063 , 0.4102937 ],
       [0.2983309 , 0.7016691 ],
       [0.7696297 , 0.2303703 ],
       [0.44531597, 0.55468403],
       [0.26823571, 0.73176429],
       [0.40929148, 0.59070852],
       [0.51189829, 0.48810171],
       [0.3701461 , 0.6298539 ],
       [0.34282428, 0.65717572],
       [0.74538501, 0.25461499],
       [0.26658433, 0.73341567],
       [0.5794107 , 0.4205893 ],
       [0.62864766, 0.37135234],
       [0.70132746, 0.29867254],
       [0.51877166, 0.48122834],
       [0.4212901 , 0.5787099 ],
       [0.60523352, 0.39476648],
       [0.28544897, 0.71455103],
       [0.2222035 , 0.7777965 ],
       [0.63110828, 0.36889172],
       [0.64612429, 0.35387571],
       [0.23803918, 0.76196082],
       [0.79365041, 0.20634959],
       [0.4608258 , 0.5391742 ],
       [0.16799877, 0.83200123],
       [0.29082509, 0.70917491],
       [0.21044871, 0.78955129],
       [0.50505329, 0.49494671],
       [0.29093853, 0.70906147],
       [0.19675157, 0.80324843],
       [0.36634537, 0.63365463],
       [0.66206249, 0.33793751],
       [0.6478595 , 0.3521405 ],
       [0.41825489, 0.58174511],
       [0.5854239 , 0.4145761 ],
       [0.42123915, 0.57876085],
       [0.25041668, 0.74958332],
       [0.52094545, 0.47905455],
       [0.63082882, 0.36917118],
       [0.55842348, 0.44157652],
       [0.58690323, 0.41309677],
       [0.41777754, 0.58222246],
       [0.55180961, 0.44819039],
```

```
       [0.55180961, 0.44819039],
       [0.3685319 , 0.6314681 ],
       [0.43906199, 0.56093801],
       [0.39552828, 0.60447172],
       [0.55685095, 0.44314905],
       [0.3597661 , 0.6402339 ],
       [0.63648782, 0.36351218],
       [0.50554485, 0.49445515],
       [0.38612614, 0.61387386],
       [0.65178899, 0.34821101],
       [0.31367991, 0.68632009],
       [0.35041733, 0.64958267],
       [0.80284545, 0.19715455],
       [0.73304443, 0.26695557],
       [0.12399122, 0.87600878],
       [0.4427548 , 0.5572452 ],
       [0.35043592, 0.64956408],
       [0.1104236 , 0.8895764 ],
       [0.60851426, 0.39148574],
       [0.34405261, 0.65594739],
       [0.28880323, 0.71119677],
       [0.69033552, 0.30966448],
       [0.49944819, 0.50055181],
       [0.4526393 , 0.5473607 ],
       [0.32459535, 0.67540465],
       [0.56282557, 0.43717443],
       [0.54045756, 0.45954244],
       [0.31023891, 0.68976109],
       [0.50136083, 0.49863917],
       [0.36920971, 0.63079029],
       [0.60927864, 0.39072136],
       [0.42430765, 0.57569235],
       [0.50860645, 0.49139355],
       [0.7301504 , 0.2698496 ],
       [0.76393421, 0.23606579],
       [0.54424892, 0.45575108],
       [0.3640628 , 0.6359372 ],
       [0.49745604, 0.50254396],
       [0.35323594, 0.64676406],
       [0.33866478, 0.66133522],
       [0.76455335, 0.23544665],
       [0.41792628, 0.58207372],
       [0.52476405, 0.47523595],
       [0.51795823, 0.48204177],
       [0.85716281, 0.14283719],
       [0.40378452, 0.59621548],
       [0.52360758, 0.47639242],
       [0.46585207, 0.53414793],
       [0.67078037, 0.32921963],
       [0.30943521, 0.69056479],
       [0.58139604, 0.41860396],
       [0.4209228 , 0.5790772 ]])
```

In [65]:

```
y_pred=lr.predict(x_test)
y_pred
```

Out[65]:

```
array([0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1])
```

In [66]:

```
print(y_test.values)
```

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

In [67]:

```
print("List of Predicted Values:")
print(y_pred)
```

```
List of Predicted Values:
[0 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 1]
```

In [68]:

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
print("\naccuracy score:%f"%(accuracy_score(y_test,y_pred)*100))
print("\nrecall score:%f"%(recall_score(y_test,y_pred)*100))
print("\nroc score:%f"%(roc_auc_score(y_test,y_pred)*100))
print(confusion_matrix(y_test,y_pred))
```

```
accuracy score:95.833333

recall score:100.000000

roc score:96.153846
[[12  1]
 [ 0 11]]
```

In [ ]:

```
```

PHYSICAL M0DEL

A physical model is a structure of thoughts and ideas from which we decipher our perceptions and test results.

If the rainfall along the coastal areas increases rapidly, the water level will increase and cause an alarming situation. It is wise to know beforehand about the upcoming hazard so as to act upon it accordingly.

The following measures must be taken -
1) The people living near the coastline must be relocated.
2) The water from the dam must not be released during this time period.
3) Embankments must be constructed.

It is important to relocate the people who must be residing near the coastline to avoid unnecessary loss of life. Through the data it can be easily predicted if we might be facing an alarming situation or not.

Therefore it is necessary to be prepared beforehand. so the residents must be notified before any uncontrollable situation arises so that innocent lives can be saved.

It is also important to issue a red alert to the fishermen and boatmen to stay away from the sea until the situation is under control. Strict implementation of this must be done by the authorities.

It is also important that the water from the dam must not be released during this time. Already the level of water reaches an alarming point and release of water from the dam can only add to the problem. It is crucial to not release the water for the time being.

BIBLI0GRAPHY

- Books referred - 1) Introduction for Scientists and Engineers (Kai Veltan)

  2)Mathematical Modelling with case studies (Belinda Barnes and Fulford)

- Kerala Rainfall Data extracted from Kaggle.com