INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 03 ISSUE: 08 | AUGUST -2019 ISSN: 2590-1892

FLOOD PREDICTION analysis using machine learning based on logistic regression algorithm by applying Python and Jupyter tool

[1]Prajna M R, [2]Deepti S, [3]Shrinidhi R , [4]Pooja J
PG scholars, Department of MCA, DSCE
Mahendra Kumar B, Asst.Prof, Dept of MCA

Abstract : Floods are among the most destructive natural disasters, which are highly complex to model. The research on the advancement of flood prediction models contributed to risk reduction, policy suggestion, minimization of the loss of human life, and reduction of the property damage associated with floods. [4]
In this paper, we developed a time series data mining approach to predict the flood in different locations with the help of causes, rise in water level and so on.

IndexTerms: **LogisticRegression[1],numpy,panda,matplotlib,accuracyscore**

INTRODUCTION

Among the natural disasters, floods are the most destructive, causing massive damage to human life, infrastructure, agriculture, and the socioeconomic system. Governments, therefore, are under pressure to develop reliable and accurate maps of flood risk areas and further plan for sustainable flood risk management focusing on prevention, protection, and preparedness.
Flood prediction models are of significant importance for hazard assessment and extreme event management. The applications in flood prediction can be classified according to flood resource variables,i.e., water level, river flood, soil moisture, rainfall–discharge, precipitation,[5] river inflow, peak flow,river flow, rainfall–runoff, flash flood, rainfall, streamflow, seasonal stream flow, flood peak discharge, urban flood, plain flood, groundwater level, rainfall stage, flood frequency analysis, flood quantiles,.surge level, extreme flow, storm surge, typhoon rainfall, and daily flows[6].

SPECIFICATIONS

- Python
- Machine learning Models
- [3]Jupyter Notebook(tool)
- Tableau(Visualization)

HARDWARE REQUIREMENTS

Processor : Pentium IV and above
Hard Disk : 250 GB
RAM : 4 GB

SOFTWARE REQUIREMENTS

Operating System: Windows 7/XP
Web server: Apache
Server Scripting: Python
Database Server: MySQL

STEPS INVOLVED IN THE PROCESS

STEP 1 : DEFINING THE PROBLEM
[2]Flood prediction is the use of forecasted precipitation and stream flow data in rainfall-runoff and stream flow routing models to forecast flow rates and water levels for periods ranging from few hours to days ahead, depending on the size of the watershed or river basin[7].

STEP 2 : COLLECTION OF DATA
The data was collected with the help of google according to the survey done on particular flood affected areas such as water level, altitude, rainfall, reasons etc

STEP 3 : PREPARE THE DATA
First you must upload the data to the [3]jupyter notebook for analysis. Check are there any null rows or columns, if any remove the rows or fill the columns with the mean value of that particular column.
Import all the packages and functions that are necessary for the analysis process.

STEP 4 : SPLITTING THE DATA INTO TRAINING AND TESTING
Here in this step we split the data into two separate tables training and testing respectively[8]. Training table is used for analysis and alteration purpose, whereas testing table is used to test the data in the

final step. We will also split the table data into 80% and 20%.

STEP 5 : ALGORITHM SELECTION
Here we select the appropriate algorithm/model that is necessary for the analysis purpose, we have selected [1]logistic regression model for processing the dataset.

STEP 6 : TRAINING THE ALGORITHM WITH DATA FOR MACHINE
Here the data is divided into xtrain, ytrain, xtest and ytest where 80% of data is taken as train data and remaining 20% of data is taken as test data.
Then if there are any nulls in the rows or column, remove those and clean the data and also if you have any high range values take the mean of those and reduce the range of those values.
Then the algorithm is trained with 80% cleaned data of xtrain and ytrain.

DATA SET FOR FLOOD PREDICTION

Data set

ID	LOCATION	SEASON	YEAR	ABOVE SEA LEVEL	TEMPARATURE	REASON	RISE IN WATER LEVEL	FLOOD PREDICTION	DEATH RATE
1	Kodagu	Rainy	2018	900m	20	Heavy Rain	1277	Yes	200
2	Kerala	Monsoon	2017	2695m	19	Heavy Rain	1095	Yes	150
3	Bangalore	Monsoon	2018	900m	24	Rain	500	No	0
4	Mumbai	Monsoon	2019	450m	26	Lack of Vegitation	700	Yes	80
5	Oddisa	Spring	2019	1672m	22	Cyclone	1000	Yes	155
6	Waynad	Rainy	2017	888m	25	Rain	300	No	0
7	Kedharnath	Summer	2015	3582m	19	Melting if ice	900	Yes	300
8	Mantralaya	Winter	2019	950m	25	Heavy Rain	200	Yes	30
9	Delhi	Monsoon	2017	216m	10		100	No	0
10	Kolkata	Spring	2016	9.14m	25	Cyclone	150	No	0
11	KanyaKumari	Winter	2004	30m	23	Tsunami	250	Yes	50
12	Mangalore	Winter	2004	22m	28	Tsunami	320	Yes	126
13	Chennai	Winter	2015	6.7m	32	Heavy Rain	200	Yes	70
14	Kochi	Rainy	2018	9m	24	Cyclone	125	Yes	67
15	Andaman	Winter	2004	16m	26	Tsunami	380	Yes	135
16	Gujarat	Rainy	2015	150m	23	Cyclone	340	yes	81
17	Andheri	Rainy	2005	57.5m	25	Rain fall	644	yes	1094
18	Pondecherry	Winter	2015	530m	28	Cyclone	780	yes	500
19	Bhubaneshwar	Rainy	2015	500m	18	Rain fall	425	yes	600
20	Uttarakhand	Rainy	2013	90m	15	Rain fall	106	yes	5700
21	Banaskantha	Rainy	2011	40m	22	Rain fall	152	yes	80
22	Tripura	Mansoon	2018	50m	20	Heavy Rain	320	yes	176
23	Jammu and Kas	Mansoon	2018	557m	7	Heavy Rain	220	yes	67
24	Guwahati	Rainy	2016	700m	18	Heavy Rain	380	yes	15
25	Odissa	Rainy	2008	17.8m	15	Cyclone	330	Yes	96

STEP 7: EVALUATE TEST DATA
After the algorithm is trained with 80% of data, the algorithm is to be tested with remaining 20% of data.
Before that check there are any nulls in the test data, if present remove those nulls and obtain clean data. Then evaluate the algorithm with test data for predictions.

STEP 8: PARAMETER TUNING
A tuning is to be done for algorithm in order to control the behaviour of the algorithm. There are many tuning methods available, here we applied [1]logistic regression.

[1]Logistic regression is suitable for life expectancy because its most widely used predictive model.

STEP 9: START USING YOUR MODEL
After the algorithm is trained with test data, it gives the prediction that is it gives the accuracy of our model.
If the prediction is below 70% then the model is failed, there may be a mistake in choosing the data , cleaning the data or methods etc.

If the prediction is above 70% then the model is good and ready for usage.

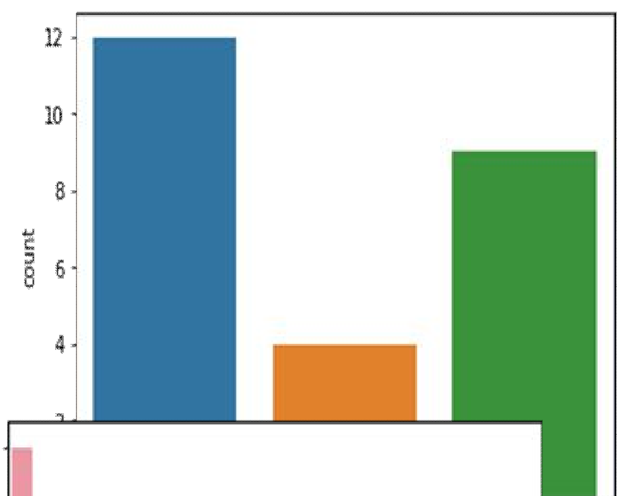
IMPLEMENTATION:
Tool used : [3]Jupyter Notebook

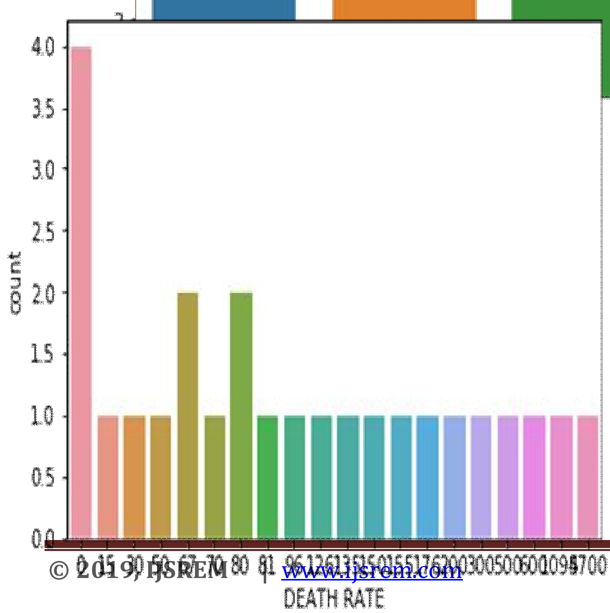
```
# coding: utf-8

    # In[282]:


    import numpy
    import pandas
    import sklearn
    import seaborn
    from sklearn.model_selection import
train_test_split
    from sklearn.linear_model import
LogisticRegression [1]
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import
mean_squared_error,r2_score
    from sklearn import preprocessing
    import matplotlib.pyplot as plt
    get_ipython().run_line_magic('matplotlib',
'inline')


    # In[283]:


    fp =
pandas.read_csv('C:/Users/Prajna/Downloads/PRO
JECT data.csv')
    fp


    # In[284]:


    fp1 = fp[['LOCATION','ABOVE SEA
LEVEL','TEMPARATURE','RISE IN WATER
LEVEL','FLOOD PREDICTION','DEATH
RATE']]


    # In[285]:


    seaborn.countplot(x='FLOOD
PREDICTION',data=fp1)


    # In[286]:

    fp1.isnull().any()


    # In[287]:

    fp1.isnull().sum()


    # In[288]:


    fp_data_X = fp1[['LOCATION','ABOVE SEA
LEVEL','TEMPARATURE','RISE IN WATER
LEVEL','DEATH RATE']]
    fp_data_Y = fp1[['FLOOD PREDICTION']]

    # In[289]:

    X_train, X_test, Y_train, Y_test =
train_test_split(fp_data_X,fp_data_Y,test_size=0.2
0)
    # In[290]:

    encoder = preprocessing.LabelEncoder()

    # In[291]:

    X_train['LOCATION']=encoder.fit_transform(
X_train['LOCATION'])

    # In[292]:

    data = X_train['RISE IN WATER LEVEL']
    price_frame = pandas.DataFrame(data)
    min_max_normalizer =
preprocessing.scale(price_frame)
    price_frame_normalized =
pandas.DataFrame(min_max_normalizer)
    price_frame_normalized

    # In[293]:

    X_train['RISE IN WATER
LEVEL']=preprocessing.scale(X_train['RISE IN
WATER LEVEL'])

    # In[294]:

    data = X_train['ABOVE SEA LEVEL']
    sea = pandas.DataFrame(data)
    min_max_normalizer =
preprocessing.scale(sea)
    price_frame_normalized =
pandas.DataFrame(min_max_normalizer)
    price_frame_normalized

    # In[295]:

    X_train['ABOVE SEA
LEVEL']=preprocessing.scale(X_train['ABOVE
SEA LEVEL'])


    # In[296]:
```

```
data = X_train['TEMPARATURE']
temp = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(temp)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[297]:

X_train['TEMPARATURE']=preprocessing.scale(X_train['TEMPARATURE'])

# In[298]:

data = X_train['DEATH RATE']
price_frame = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(price_frame)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[299]:

X_train['DEATH RATE']=preprocessing.scale(X_train['DEATH RATE'])

# In[300]:

X_test['LOCATION']=encoder.fit_transform(X_test['LOCATION'])

# In[301]:
data = X_test['RISE IN WATER LEVEL']
price_frame = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(price_frame)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[302]:

X_test['RISE IN WATER LEVEL']=preprocessing.scale(X_test['RISE IN WATER LEVEL'])

# In[303]:

data = X_test['ABOVE SEA LEVEL']
sea = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(sea)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[304]:

X_test['ABOVE SEA LEVEL']=preprocessing.scale(X_test['ABOVE SEA LEVEL'])

# In[305]:

data = X_test['TEMPARATURE']
temp = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(temp)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[306]:

X_test['TEMPARATURE']=preprocessing.scale(X_test['TEMPARATURE'])

# In[307]:

data = X_test['DEATH RATE']
price_frame = pandas.DataFrame(data)
min_max_normalizer = preprocessing.scale(price_frame)
price_frame_normalized = pandas.DataFrame(min_max_normalizer)
price_frame_normalized

# In[308]:

X_test['DEATH RATE']=preprocessing.scale(X_test['DEATH RATE'])

# In[309]:

logistic_regression = LogisticRegression() [1]
```

```
# In[310]:

logistic_regression.fit(X_train,Y_train) [1]

# In[311]:

y_pred = logistic_regression.predict(X_test) [1]

# In[312]:

print(accuracy_score(Y_test,y_pred))

# In[219]:

plt.scatter(fp1['RISE IN WATER LEVEL'],fp1['FLOOD PREDICTION'])
plt.xlabel('RISE IN WATER LEVEL')
plt.ylabel('FLOOD PREDICTION')
plt.title('RISE IN WATER LEVEL - FLOOD PREDICTION')
```

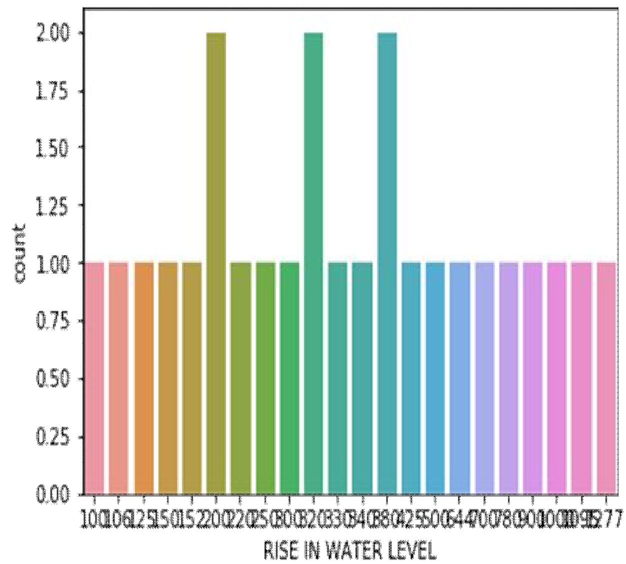
VISUALIZATION





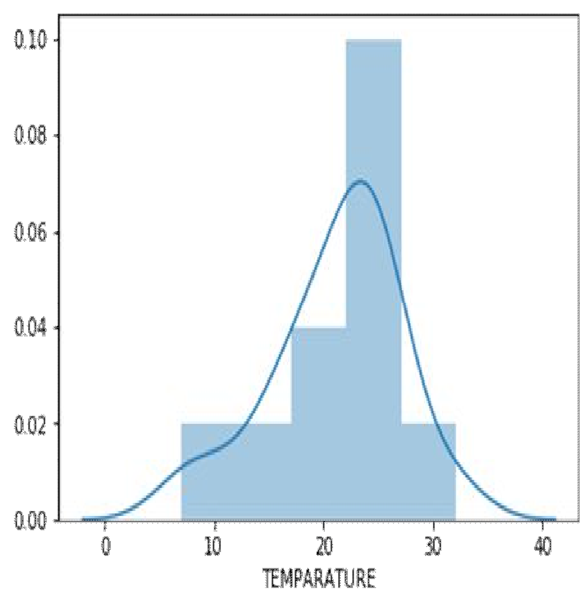
Flood Prediction

This graph shows the flood prediction column which is mentioned in the dataset. It shows the possibility of flood happening in particular area taking into consideration of rise in water level .
This graph shows the number of people died in the flood in particular area and in particular year. The data considered for this graph is taken by the cleaned dataset.



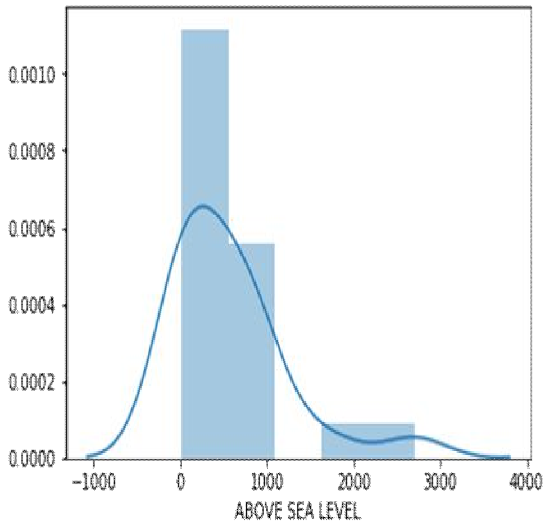
Rise in water level

This graph shows the rise in water level which causes the flood in particular area and in particular year. The data considered for this graph is taken by the cleaned dataset.



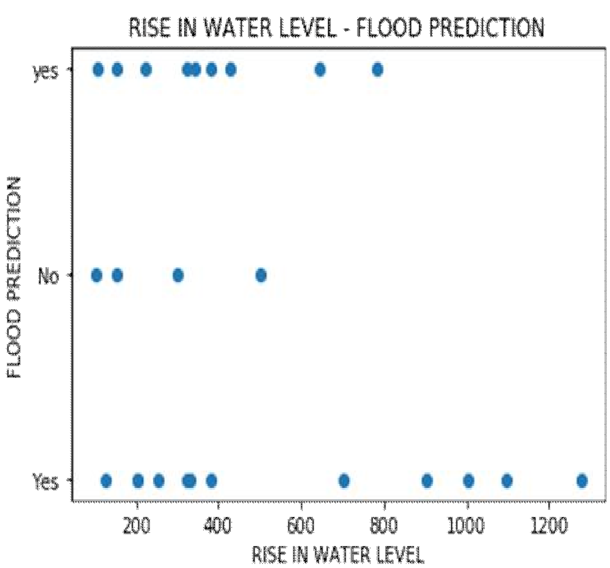
Temperature

This graph shows the temperature in particular area and in particular year where the flood happened. The data considered for this graph is taken by the cleaned dataset.



Above sea level

This graph show that how much a particular are is above sea level. It helps in predicting the cause and the probability of flood happening.



Scatter plot

This scatter plot will show how the prediction is done on the basis of rise in water level in particular area. Here we can see that at what level of rise in water can cause the flood. We can predict whether the flood happens or not considering the statistics of rise in water level.

CONCLUSION:

Flood occurs most commonly from heavy rain fall when natural watercourses do not have capacity to excess water.
In costal areas, water inundation can be caused by a storm, tsunami or high tide coinciding with higher than normal river levels.

References:

[1] Linear Logistic regression : an introduction, author:T Haifley. Cypress Semicond. Corp., San Jose, CA, USA .
[2]Flood Prediction Using Machine Learning Models:Literature Review, Author: Amir Mosavi, Pinar Qzturk, Kwok-wing Chau.
[3]Pysnippet: Accelerating Exploratory Data Analysis In Jupyter Notebook Through Facilitated Access To Example Code, Author: Alex Watson, Scott Bateman and Suprio Ray.
[4] web reference- www.mdpi.com
[5] web reference- www.preprints.org
[6] Journal paper- Amir Mosavi, Pinar Ozturk, Kwok-wing Chau. "Flood Prediction Using Machine Learning Models: Literature Review" , Water, 2018
[7] Journal paper- Paper: Submitted to Victoria University College.
[8] Journal paper- Puneet Mathur. "Machine Learning Applications Using Python" , Springer Nature, 2019
[9] Journal paper- Ramesh Naidu Laveti, Swetha Kuppili, Janaki Ch, Supriya N Pal, N. Sarat Chandra Babu. "Implementation of learning analytics framework for MOOCs using state-of-the-art in-memory computing" , 2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH), 2017