Report

On

Flood Prediction Using Machine Learning

By

M.Tech Modeling & Simulation

(CMS1905, CMS1914)

Under Guidance of

B. S. Pujari

UGC - Assistant Professor
Dept of Modeling and Simulation
Savitribai Phule University of Pune,

1. 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 the property damage associated with floods. To mimic the complex mathematical expressions of physical processes of floods, during the past two decades, machine learning (ML) methods contributed highly in the advancement of prediction systems providing better performance and cost-effective solutions. Due to the vast benefits and potential of ML, its popularity dramatically increased among hydrologists. Researchers through introducing novel ML methods and hybridizing of the existing ones aim at discovering more accurate and efficient prediction models. The main contribution of this project is to demonstrate the state of the art of ML models in flood prediction and to give insight into the most suitable models.

2. 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. However, the prediction of flood lead time and occurrence location is fundamentally complex due to the dynamic nature of climate condition. Therefore, today's major flood prediction models are mainly data-specific and involve various simplified assumptions.

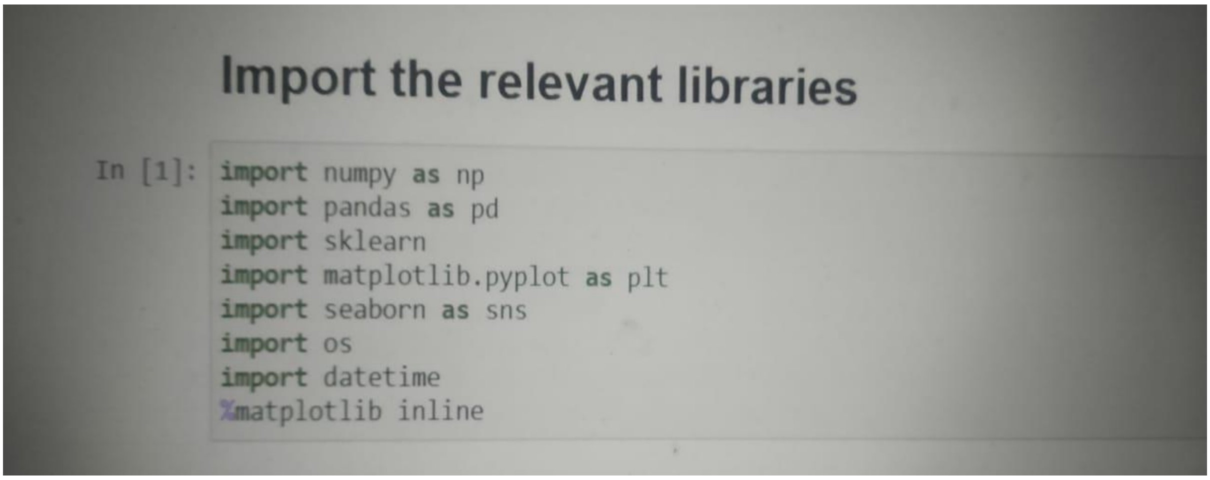
ML algorithms have important characteristics that need to be carefully taken into consideration. The first is that they are as good as their training, whereby the system learns the target task based on past data. If the data is scarce or does not cover varieties of the task, their learning falls short, and hence, they cannot perform well when they are put into work. The second aspect is the capability of each ML algorithm, which may vary across different types of tasks. This can also be called a "generalization problem", which indicates how well the trained system can predict cases it was not trained for, i.e., whether it can predict beyond the range of the training dataset.

For flood prediction, so many aspects are considered such as rainfall, weather forecast, hydraulic powerplants, climate change, etc. In our project, we are mainly working on rainfall. We use different ML models to predict the rainfall in certain areas, in certain months. And by setting the threshold value we get to know that where high rainfall, low rainfall occur and on the basis of that we predict the flood situation in certain areas. We are having the rainfall data from 1901 – 2015 for every month.

3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the process of visualizing and analyzing data to extract insights from it. In other words, EDA is the process of summarizing important characteristics of data in order to gain better understanding of the dataset.

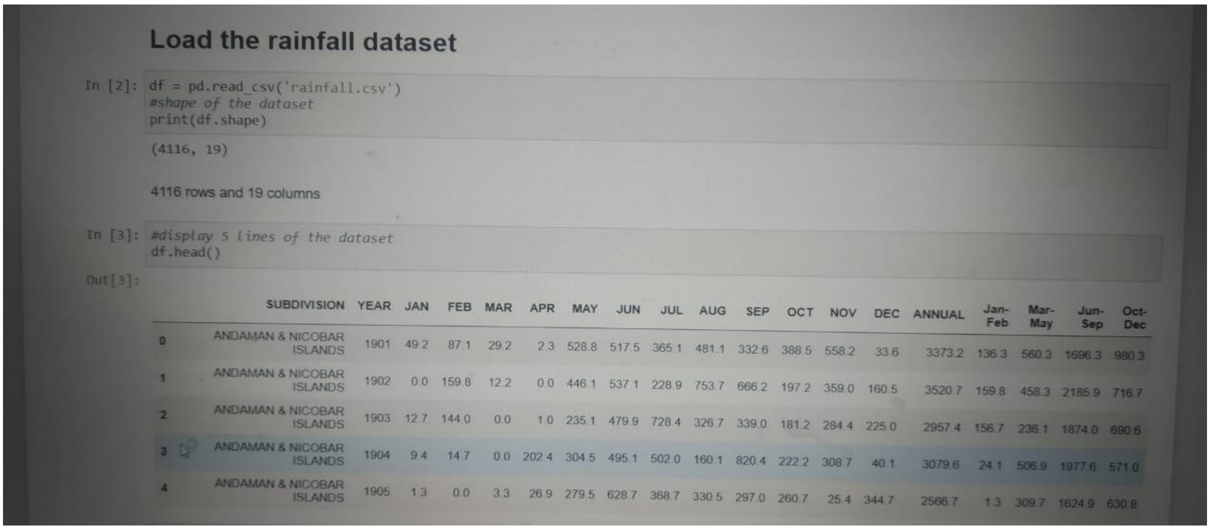
1. Import the relevant libraries



Import the relevant libraries

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import os
import datetime
%matplotlib inline
```

2. Loading the data



Load the rainfall dataset

In [2]:
```
df = pd.read_csv('rainfall.csv')
#shape of the dataset
print(df.shape)
```

```
(4116, 19)
```

4116 rows and 19 columns

In [3]:
```
#display 5 lines of the dataset
df.head()
```

Out[3]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.0
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.8

```
In [4]: #Explore the columns and their datatypes

In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4116 entries, 0 to 4115
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SUBDIVISION  4116 non-null   object
 1   YEAR         4116 non-null   int64
 2   JAN          4112 non-null   float64
 3   FEB          4113 non-null   float64
 4   MAR          4110 non-null   float64
 5   APR          4112 non-null   float64
 6   MAY          4113 non-null   float64
 7   JUN          4111 non-null   float64
 8   JUL          4109 non-null   float64
 9   AUG          4112 non-null   float64
 10  SEP          4110 non-null   float64
 11  OCT          4109 non-null   float64
 12  NOV          4105 non-null   float64
 13  DEC          4106 non-null   float64
 14  ANNUAL       4090 non-null   float64
 15  Jan-Feb      4110 non-null   float64
 16  Mar-May      4107 non-null   float64
 17  Jun-Sep      4106 non-null   float64
 18  Oct-Dec      4103 non-null   float64
dtypes: float64(17), int64(1), object(1)
memory usage: 611.1+ KB
```

3. Summary of dataset

Summary of the dataset

```
In [6]: df.describe()
Out[6]:
```

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	
count	4116.000000	4112.000000	4113.000000	4110.000000	4112.000000	4113.000000	4111.000000	4109.000000	4112.000000	4110.000000	4109.000000	4105.0
mean	1958.218659	18.957320	21.805325	27.359197	43.127432	85.745417	230.234444	347.214334	290.263497	197.361922	95.507009	39.8
std	33.140898	33.585371	35.909488	46.959424	67.831168	123.234904	234.710758	269.539667	188.770477	135.408345	99.519134	68.6
min	1901.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	0.100000	0.000000	0.0
25%	1930.000000	0.600000	0.600000	1.000000	3.000000	8.600000	70.350000	175.600000	155.975000	100.525000	14.600000	0.7
50%	1958.000000	6.000000	6.700000	7.800000	15.700000	36.600000	138.700000	284.800000	259.400000	173.900000	65.200000	9.5
75%	1987.000000	22.200000	26.800000	31.300000	49.950000	97.200000	305.150000	418.400000	377.800000	265.800000	148.400000	46.1
max	2015.000000	583.700000	403.500000	605.600000	595.100000	1168.600000	1609.900000	2362.800000	1664.600000	1222.000000	948.300000	648.9

4. Selecting subdivision Madhya Maharashtra

Select SUBDIVISION 'Madhya Maharashtra'

```
In [10]: df_up = df[df.SUBDIVISION == 'MADHYA MAHARASHTRA']
         df_up
Out[10]:
```

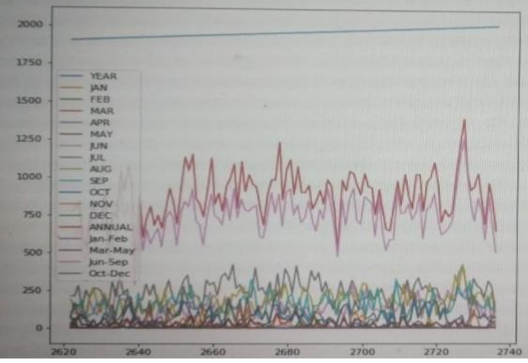
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
2622	MADHYA MAHARASHTRA	1901	18.8	0.6	7.7	36.6	30.4	107.7	215.9	194.1	83.7	68.7	4.4	0.5	769.0	19.4	74.7	601.4	73.5
2623	MADHYA MAHARASHTRA	1902	7.8	0.0	0.1	5.0	9.8	102.6	210.9	114.5	169.5	60.4	40.5	62.9	784.0	7.8	14.9	597.5	163.8
2624	MADHYA MAHARASHTRA	1903	7.6	0.0	0.0	3.2	77.2	86.3	281.8	155.5	142.3	74.2	7.6	2.2	837.9	7.6	80.4	665.9	84.1
2625	MADHYA MAHARASHTRA	1904	0.4	4.7	1.7	3.0	18.7	114.6	126.5	59.5	183.0	91.1	0.0	0.4	603.5	5.1	23.4	483.6	91.4
2626	MADHYA MAHARASHTRA	1905	0.0	1.2	0.0	2.3	23.6	65.0	252.8	79.0	52.6	52.9	8.3	0.0	537.8	1.2	25.9	449.5	61.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2732	MADHYA MAHARASHTRA	2011	0.0	0.3	0.3	5.0	2.9	133.3	261.4	238.1	148.4	62.8	0.0	0.0	852.6	0.3	8.2	781.3	62.8
2733	MADHYA MAHARASHTRA	2012	0.0	0.0	0.0	3.0	1.4	67.9	203.0	187.8	129.5	95.2	2.2	0.0	689.8	0.0	4.4	588.1	97.3
2734	MADHYA MAHARASHTRA	2013	0.1	5.3	0.8	5.7	6.0	212.4	311.8	147.0	210.3	57.8	4.0	1.3	962.4	5.3	12.4	881.5	63.1
2735	MADHYA MAHARASHTRA	2014	3.1	6.2	24.4	7.5	29.8	44.0	277.9	240.3	120.4	38.5	32.8	13.1	838.0	9.3	61.7	682.6	84.4
2736	MADHYA MAHARASHTRA	2015	1.4	0.8	41.2	9.6	24.4	177.0	111.7	67.2	146.6	48.3	16.2	0.1	644.5	2.2	75.3	502.5	64.5

115 rows × 19 columns

5. Graph showing trend of rainfall in Madhya Maharashtra

Plot the graph showing the trend of rainfall in Madhya Maharashtra

```
In [16]: df_up.plot(figsize = (8,8), legend=True)
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x16947f93648>
```



6. 10 subdivisions with less rainfall in 2005

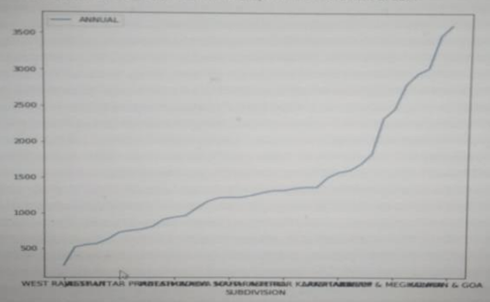
Which are the 10 SUBDIVISION with less rainfall in the 2005? ¶

```
In [12]: df2000 = df.groupby('YEAR').get_group(2005)
         less_rainfall10 = df2000[['SUBDIVISION','YEAR','ANNUAL']].sort_values('ANNUAL')
         less_rainfall10.head(15)
```

Out[12]:

	SUBDIVISION	YEAR	ANNUAL
1921	WEST RAJASTHAN	2005	260.8
1461	HARYANA DELHI & CHANDIGARH	2005	519.5
1576	PUNJAB	2005	552.0
2036	EAST RAJASTHAN	2005	567.9
2496	SAURASHTRA & KUTCH	2005	631.6
1231	WEST UTTAR PRADESH	2005	725.7
2151	WEST MADHYA PRADESH	2005	752.2
1116	EAST UTTAR PRADESH	2005	770.3
3761	NORTH INTERIOR KARNATAKA	2005	807.5
1001	BIHAR	2005	907.8
2841	MATATHWADA	2005	940.5
686	JHARKHAND	2005	960.4
3416	RAYALSEEMA	2005	1066.4
3301	TELANGANA	2005	1163.1
2956	VIDARBHA	2005	1214.6

```
In [18]: less_rainfall10.plot(x= 'SUBDIVISION', y='ANNUAL', figsize=(8,8))
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x169487d7588>
```



7. 10 subdivisions with high rainfall in 2005

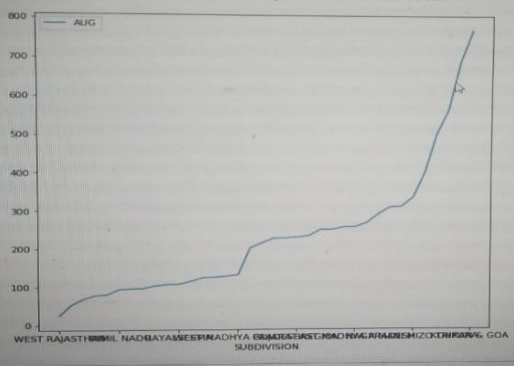
Which are the 10 SUBDIVISION with high rainfall in the 2005?

```
In [14]: df2000 = df.groupby('YEAR').get_group(2005)
         high_rainfall10 = df2000[['SUBDIVISION','YEAR','ANNUAL','AUG']].sort_values('AUG')
         high_rainfall10.tail(15)
```

Out[14]:

	SUBDIVISION	YEAR	ANNUAL	AUG
3876	SOUTH INTERIOR KARNATAKA	2005	1319.3	233.2
1346	UTTARAKHAND	2005	1348.2	249.2
2956	VIDARBHA	2005	1214.6	249.2
2726	MADHYA MAHARASHTRA	2005	1221.3	256.1
2266	EAST MADHYA PRADESH	2005	1366.4	256.6
3071	CHHATTISGARH	2005	1287.2	267.7
3991	KERALA	2005	3031.1	291.0
1001	BIHAR	2005	907.8	309.1
99	ANDAMAN & NICOBAR ISLANDS	2005	2954.7	311.1
426	NAGA MANI MIZO TRIPURA	2005	1836.7	334.8
196	ARUNACHAL PRADESH	2005	2335.5	400.0
311	ASSAM & MEGHALAYA	2005	2468.3	501.9
541	SUB HIMALAYAN WEST BENGAL & SIKKIM	2005	2807.7	564.5
3646	COASTAL KARNATAKA	2005	3470.9	692.0
2611	KONKAN & GOA	2005	3615.7	772.2

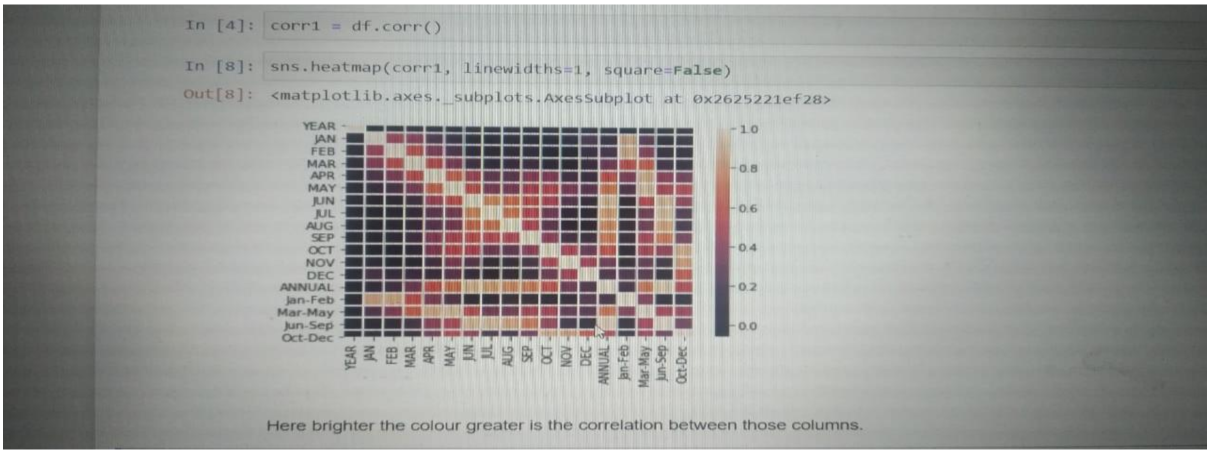
```
In [19]: high_rainfall10.plot(x= 'SUBDIVISION', y='AUG', figsize=(8,8))
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x16948851348>
```



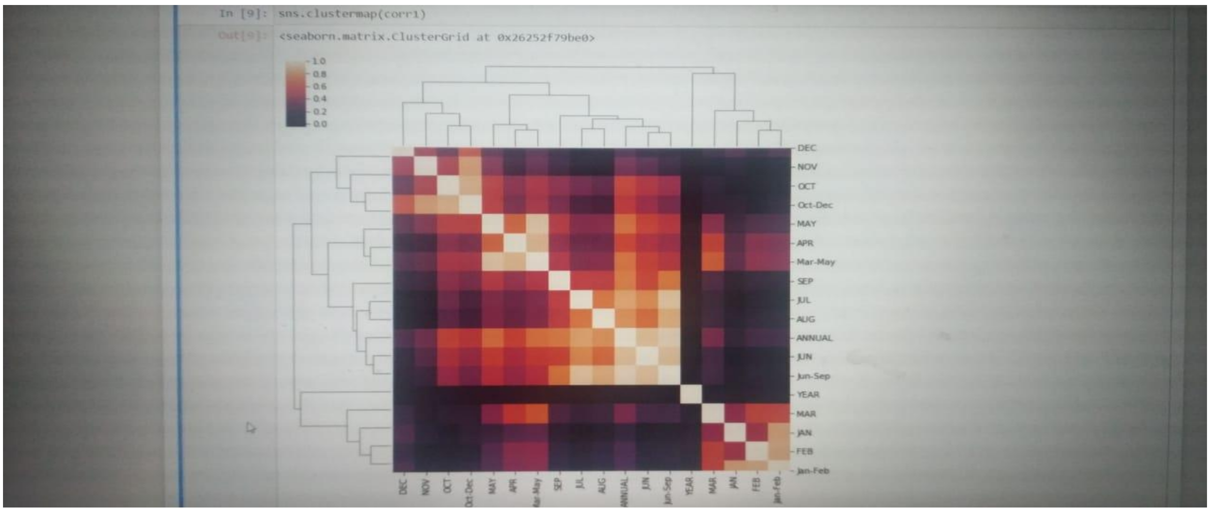
3. Data Preprocessing

Correlation :-

The strength of the linear association between two variables is quantified by the correlation coefficient. The correlation coefficient always takes a value between -1 and 1, with 1 or -1 indicating perfect correlation (all points would lie along a straight line in this case). Here we are finding the correlation between the all columns with each other.

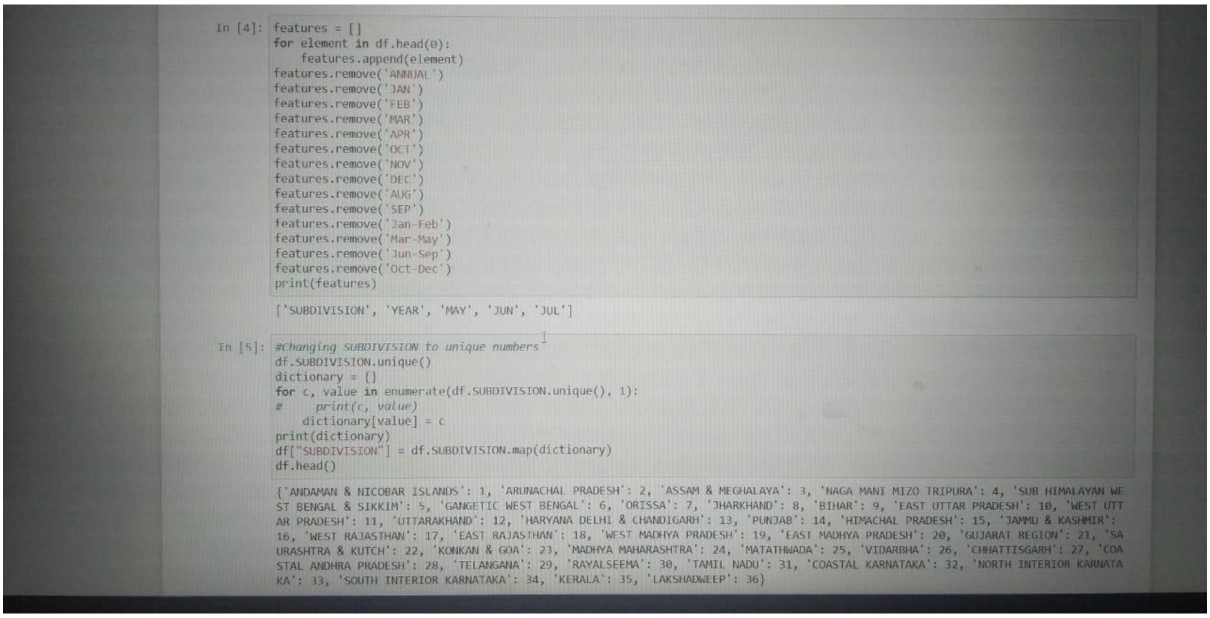


Now we are using cluster map format to find out the relation between every column with each other. Cluster map is basically unsupervised concept. Here we are using dendrograms to find out the relation.



Data Preparation :-

We are preparing the supervised data to get the predictions as we needed. We are removing NA values which are less than 5%. We are now splitting the data into training data and test data. Here we are taking subdivisions as 1, 2, …. ,36.

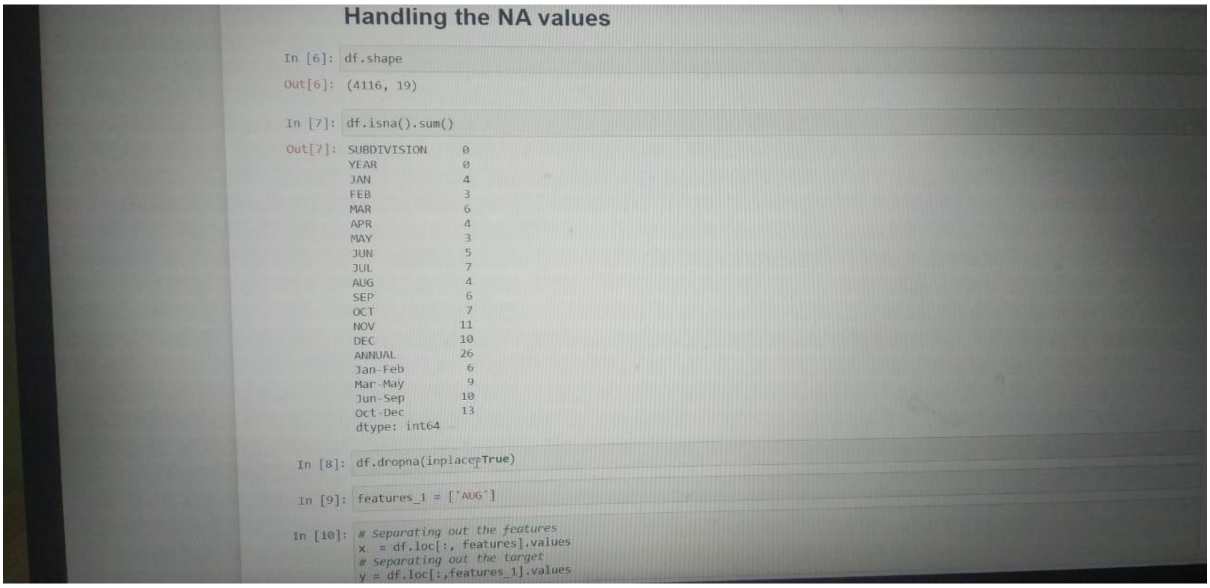


```
In [4]: features = []
        for element in df.head(0):
            features.append(element)
        features.remove('ANNUAL')
        features.remove('JAN')
        features.remove('FEB')
        features.remove('MAR')
        features.remove('APR')
        features.remove('OCT')
        features.remove('NOV')
        features.remove('DEC')
        features.remove('AUG')
        features.remove('SEP')
        features.remove('Jan-Feb')
        features.remove('Mar-May')
        features.remove('Jun-Sep')
        features.remove('Oct-Dec')
        print(features)

        ['SUBDIVISION', 'YEAR', 'MAY', 'JUN', 'JUL']

In [5]: #Changing SUBDIVISION to unique numbers
        df.SUBDIVISION.unique()
        dictionary = {}
        for c, value in enumerate(df.SUBDIVISION.unique(), 1):
        #     print(c, value)
            dictionary[value] = c
        print(dictionary)
        df["SUBDIVISION"] = df.SUBDIVISION.map(dictionary)
        df.head()

        {'ANDAMAN & NICOBAR ISLANDS': 1, 'ARUNACHAL PRADESH': 2, 'ASSAM & MEGHALAYA': 3, 'NAGA MANI MIZO TRIPURA': 4, 'SUB HIMALAYAN WE
        ST BENGAL & SIKKIM': 5, 'GANGETIC WEST BENGAL': 6, 'ORISSA': 7, 'JHARKHAND': 8, 'BIHAR': 9, 'EAST UTTAR PRADESH': 10, 'WEST UTT
        AR PRADESH': 11, 'UTTARAKHAND': 12, 'HARYANA DELHI & CHANDIGARH': 13, 'PUNJAB': 14, 'HIMACHAL PRADESH': 15, 'JAMMU & KASHMIR':
        16, 'WEST RAJASTHAN': 17, 'EAST RAJASTHAN': 18, 'WEST MADHYA PRADESH': 19, 'EAST MADHYA PRADESH': 20, 'GUJARAT REGION': 21, 'SA
        URASHTRA & KUTCH': 22, 'KONKAN & GOA': 23, 'MADHYA MAHARASHTRA': 24, 'MATATHWADA': 25, 'VIDARBHA': 26, 'CHHATTISGARH': 27, 'COA
        STAL ANDHRA PRADESH': 28, 'TELANGANA': 29, 'RAYALSEEMA': 30, 'TAMIL NADU': 31, 'COASTAL KARNATAKA': 32, 'NORTH INTERIOR KARNATA
        KA': 33, 'SOUTH INTERIOR KARNATAKA': 34, 'KERALA': 35, 'LAKSHADWEEP': 36}
```



Handling the NA values

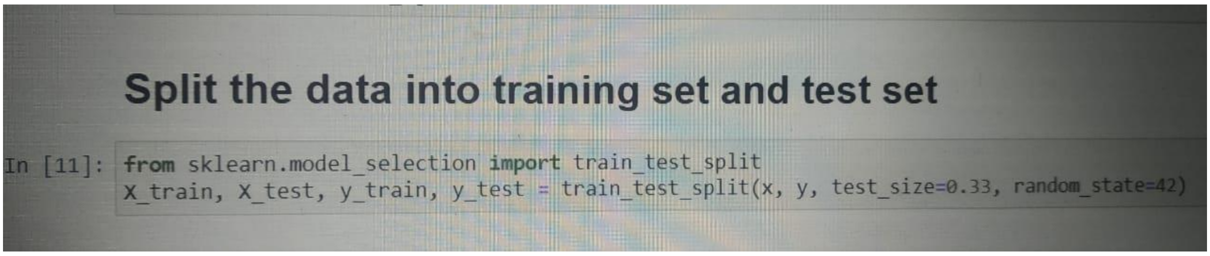
```
In [6]: df.shape
Out[6]: (4116, 19)

In [7]: df.isna().sum()
Out[7]: SUBDIVISION     0
        YEAR            0
        JAN             4
        FEB             3
        MAR             6
        APR             4
        MAY             3
        JUN             5
        JUL             7
        AUG             4
        SEP             6
        OCT             7
        NOV            11
        DEC            10
        ANNUAL         26
        Jan-Feb         6
        Mar-May         9
        Jun-Sep        10
        Oct-Dec        13
        dtype: int64

In [8]: df.dropna(inplace=True)

In [9]: features_1 = ['AUG']

In [10]: # Separating out the features
         x  = df.loc[:, features].values
         # Separating out the target
         y = df.loc[:,features_1].values
```



Split the data into training set and test set

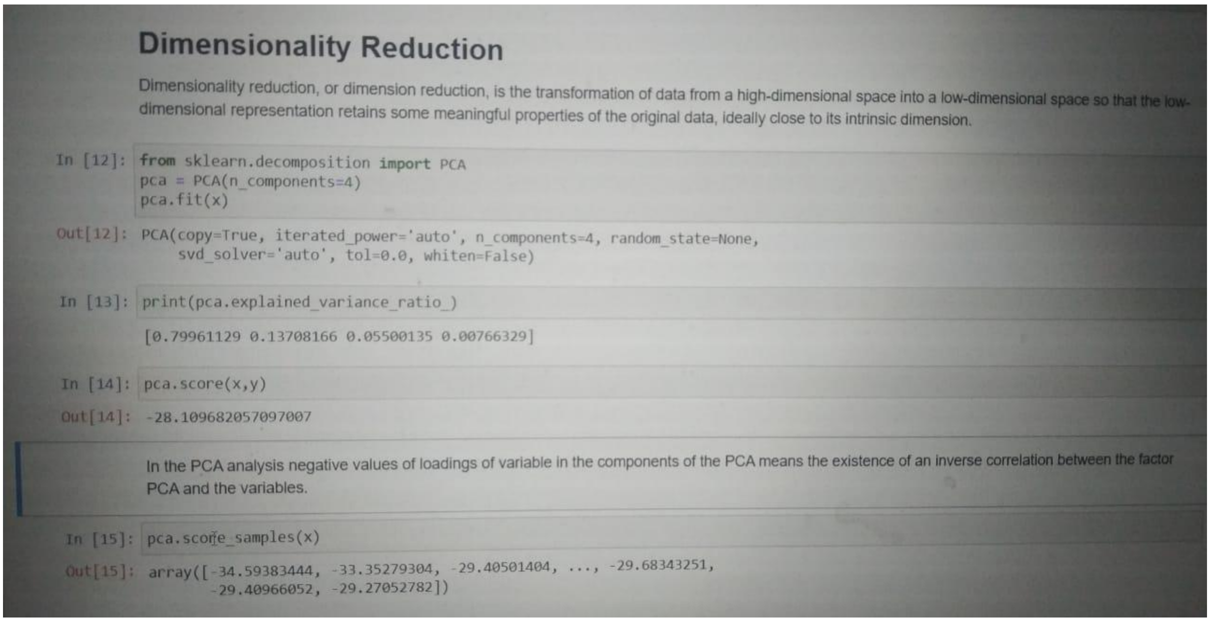
```
In [11]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

Dimensionality Reduction :-

Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data. This is called dimensionality reduction.

The most common approach to dimensionality reduction is called principal components analysis or PCA. This is a technique that comes from the field of linear algebra and can be used as a data preparation technique to create a projection of a dataset prior to fitting a model. In the PCA analysis negative values of loadings of variable in the components of the PCA means the existence of an inverse correlation between the factor PCA and the variables.



4. Machine Learning Models

1. Support Vector Machine :-

The support vector (SV) as a nonlinear search algorithm using statistical learning theory. Later, the SVM was introduced as a class of SV, used to minimize over-fitting and reduce the expected error of learning machines. SVM is greatly popular in flood modelling; it is a supervised learning machine which works based on the statistical learning theory and the structural risk minimization rule. The training algorithm of SVM builds models that assign new non-probabilistic binary linear classifiers, which minimize the empirical classification error and maximize the geometric margin via inverse problem solving. SVM is used to predict a quantity forward in time based on training from past data. Over the past two decades, the SVM was also extended as a regression tool, known as support vector regression (SVR). Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample.

2. Random Forest :-

Random forest is a Supervised Learning Algorithm which uses ensemble learning method for classification and regression. Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees.

3. Multi Linear Regression :-

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable. Multiple linear regression (MLR) is used to determine a mathematical relationship among a number of random variables. In other terms, MLR examines how multiple independent variables are related to one dependent variable. Once each of the independent factors has been determined to predict the dependent variable, the information on the multiple variables can be used to create an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

4. Multilayer Perceptron (MLP) :-

The MLP—an advanced representation of ANNs— recently gained popularity. The MLP is a class of FFNN which utilizes the supervised learning of BP for training the network of interconnected nodes of multiple layers. Simplicity, nonlinear activation, and a high number of layers are characteristics of the MLP. Due to these characteristics, the model was widely used in flood prediction and other complex hydrogeological models. In an assessment of ANN classes used in flood modelling, MLP models were reported to be more efficient with better generalization ability. Nevertheless, the MLP is generally found to be more difficult to optimize. Back-percolation learning algorithms are used to individually calculate the propagation error in hidden network nodes for a more advanced modelling approach. Here, it is worth mentioning that the MLP, more than any other variation of ANNs (e.g., FFNN, BPNN, and FNN), gained popularity among hydrologists.

5. Result

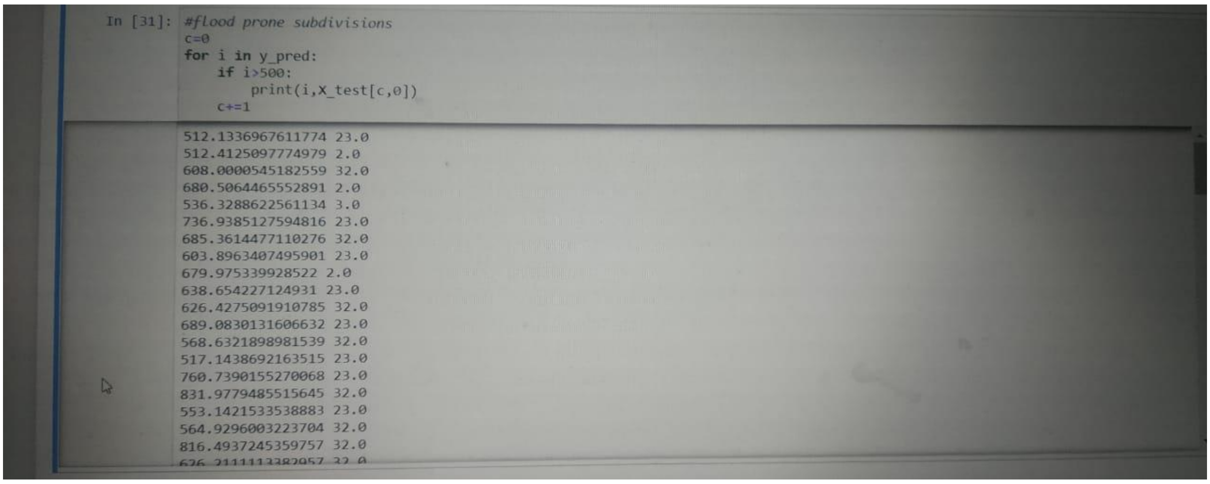
1. Table :-

	SVM	Random Forest	Multi Linear Regression	MLP Regressor
Score	0.3428042979348 1726	0.514737774882588 5	0.550955783669908 2	0.551376852897723
Mean Absolute Error	109.85604996109 512	96.58520467449426	91.40146397442292	91.4226284545516
Mean Squared Error	23083.990469639 375	17044.82933270201 5	15772.67224608601 8	15757.88219494692 4

2. Flood Prone Subdivisions :-

Now we are considering the 500 mm is the threshold value. When we predict the rainfall for certain month for certain subdivision that time the rainfall is more than 500 mm then that subdivision is considered as the flood prone area.

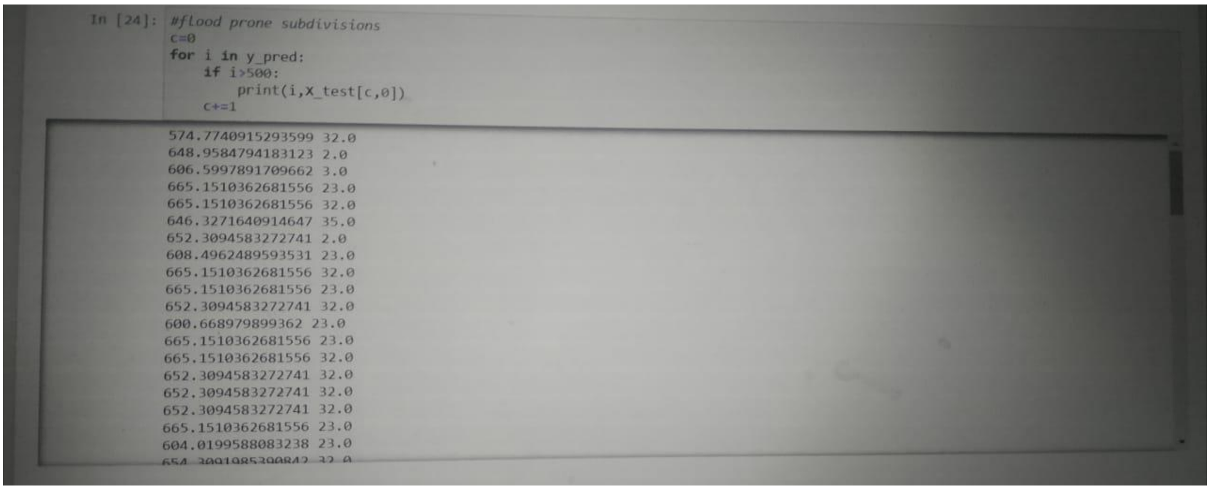
1. SVM



```
In [31]: #flood prone subdivisions
         c=0
         for i in y_pred:
             if i>500:
                 print(i,X_test[c,0])
             c+=1

512.1336967611774 23.0
512.4125097774979 2.0
608.0000545182559 32.0
680.5064465552891 2.0
536.3288622561134 3.0
736.9385127594816 23.0
685.3614477110276 32.0
603.8963407495901 23.0
679.975339928522 2.0
638.654227124931 23.0
626.4275091910785 32.0
689.0830131606632 23.0
568.6321898981539 32.0
517.1438692163515 23.0
760.7390155270068 23.0
831.9779485515645 32.0
553.1421533538883 23.0
564.9296003223704 32.0
816.4937245359757 32.0
```

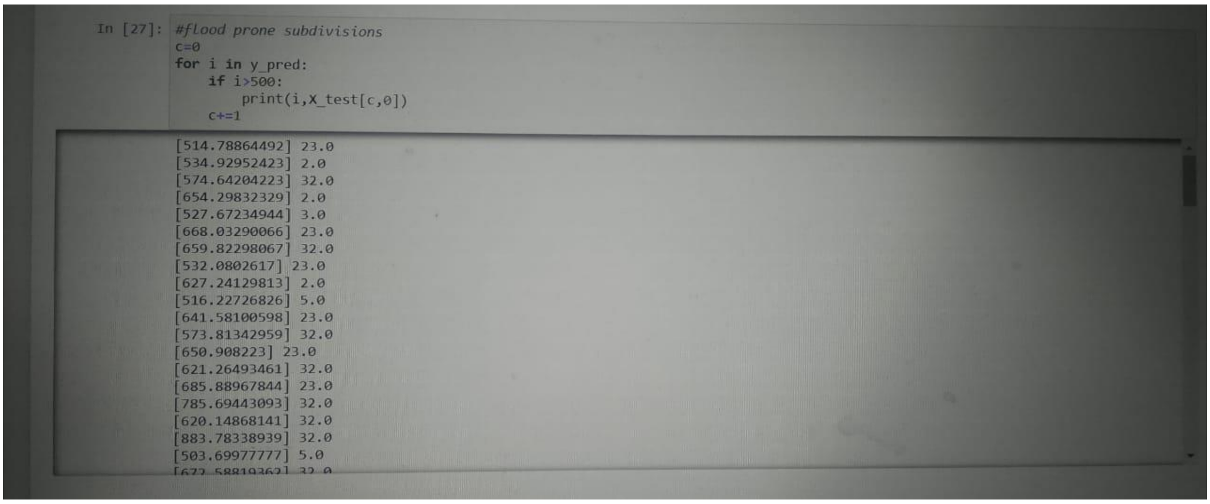
2. Random forest



```
In [24]: #flood prone subdivisions
         c=0
         for i in y_pred:
             if i>500:
                 print(i,X_test[c,0])
             c+=1

574.7740915293599 32.0
648.9584794183123 2.0
606.5997891709662 3.0
665.1510362681556 23.0
665.1510362681556 32.0
646.3271640914647 35.0
652.3094583272741 2.0
608.4962489593531 23.0
665.1510362681556 32.0
665.1510362681556 23.0
652.3094583272741 32.0
600.668979899362 23.0
665.1510362681556 23.0
665.1510362681556 32.0
652.3094583272741 32.0
652.3094583272741 32.0
652.3094583272741 32.0
665.1510362681556 23.0
604.0199588083238 23.0
```

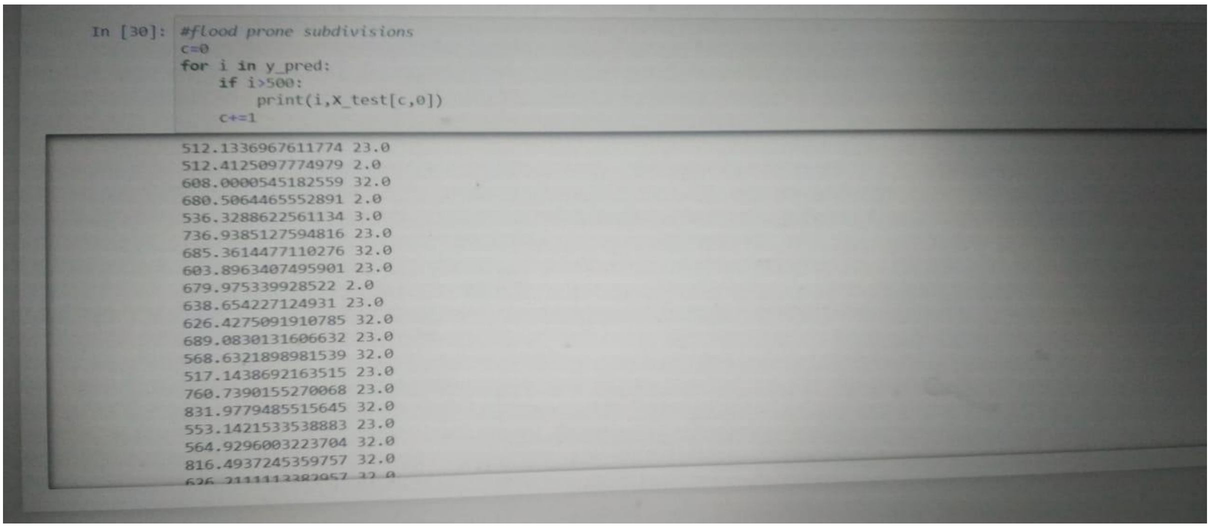
3. Multi Linear Regression



```
In [27]: #flood prone subdivisions
         c=0
         for i in y_pred:
             if i>500:
                 print(i,X_test[c,0])
             c+=1

[514.78864492] 23.0
[534.92952423] 2.0
[574.64204223] 32.0
[654.29832329] 2.0
[527.67234944] 3.0
[668.03290066] 23.0
[659.82298067] 32.0
[532.0802617] 23.0
[627.24129813] 2.0
[516.22726826] 5.0
[641.58100598] 23.0
[573.81342959] 32.0
[650.908223] 23.0
[621.26493461] 32.0
[685.88967844] 23.0
[785.69443093] 32.0
[620.14868141] 32.0
[883.78338939] 32.0
[503.69977777] 5.0
```

4. MLP Regressor



6. Conclusion

The current state of ML modelling for flood prediction is quite young and in the early stage of advancement.

Here we are targeting the rainfall prediction which leads us towards the flood prone areas. In this project, we are mainly focusing on four ML models viz., Support Vector Machine (SVM), Random Forest (RF), Multi Linear Regression and Multilayer Perceptron (MLP).

According to our result, MLP and Multi Linear Regression are more effective than SVM and RF.

According to our main aim of our project, we are concluded that the high rainfall subdivisions having more possibilities of Flood.