A Report on

Flood Extent on SAR Images Using a Supervised Classifier

For the
Image Processing (ECL - 410)

By

Mohd.Aarish Siddiqui (BT17ECE061)

Under the supervision of
Dr. Ghanapriya Singh

National Institute of Technology, Uttarakhand
Satellite Campus , MNIT , Jaipur – 302017 (India)

PROJECT TITLE

Flood Extent on SAR Images Using a Supervised Classifier

CONTENTS

INTRODUCTION:

The availability of a flood extent map becomes vital to assist the local authorities to plan rescue operations and evacuate the premises promptly. This project proposes a novel automatic way to rapidly map the flood extent using a supervised classifier. The methodology described in this project is fully automated since the image segmentation is done, extraction of features and training of the supervised classifier on the dataset comprising of pre flood and post flood Synthetic Aperture Radar (SAR) images. The entire flood mapping process, which consists of preprocessing the images, the extraction of the training dataset, and finally the classification, was assessed on flood events.

In the event of flooding, a clear cloud-free image acquired instantaneously is necessary to have a synoptic view of the affected area. In this context, remotely sensed images are suitable to map inundations, particularly when harsh climatic conditions are encountered and the access to the affected site is impractical. Regarding the dataset neededa pair of pre flood and post flood SAR image of same area is used.

MOTIVATION:

This project was chosen by us because it has lots of real life applications, if successfully implemented it will be very impactful to the society. In our country every year there is flood situation in some states. The novel method proposed in this project will prove capable of a quick and automatic mapping of the extent of the inundation, which can assist response authorities to prioritize during rescue operations. The rapidity of execution is a very important factor, especially when the main purpose of the flood maps is to support relief efforts at the time of the disaster.

Flood is causing devastating damage every year all over the world. One way to improve the readiness of the rescue team, policy makers and communities is by providing the flood extent maps, promptly after the disaster, preferably in an automated way and with the minimum number of satellite imagery to reduce cost.

METHODOLOGY:

- The Synthetic Aperture Radar (SAR) images of the pre flood and post flood areas of Kerela were taken from the ISRO website.
- Image Preprocessing steps were performed such as use of Gaussian Filter, contrast enhancement etc.
- Segmentation of images: partitioning of the image using one of the segmentation techniques called thresholding.
- Based on the threshold value the final segmented image was converted into the logical data type i.e. into the binary form.
- Extraction of features will be done using techniques such as gray level co-occurrence matrix (GLCM) and using that data to make two datasets namely A for pre flood and B for post flood images.
- Merging of two datasets and addition of one extra column of 0 and 1 for classification.
- Supervised classifier will be used to train the model and classify between the pre flood and post flood images.

DETAILED BLOCK DIAGRAM:

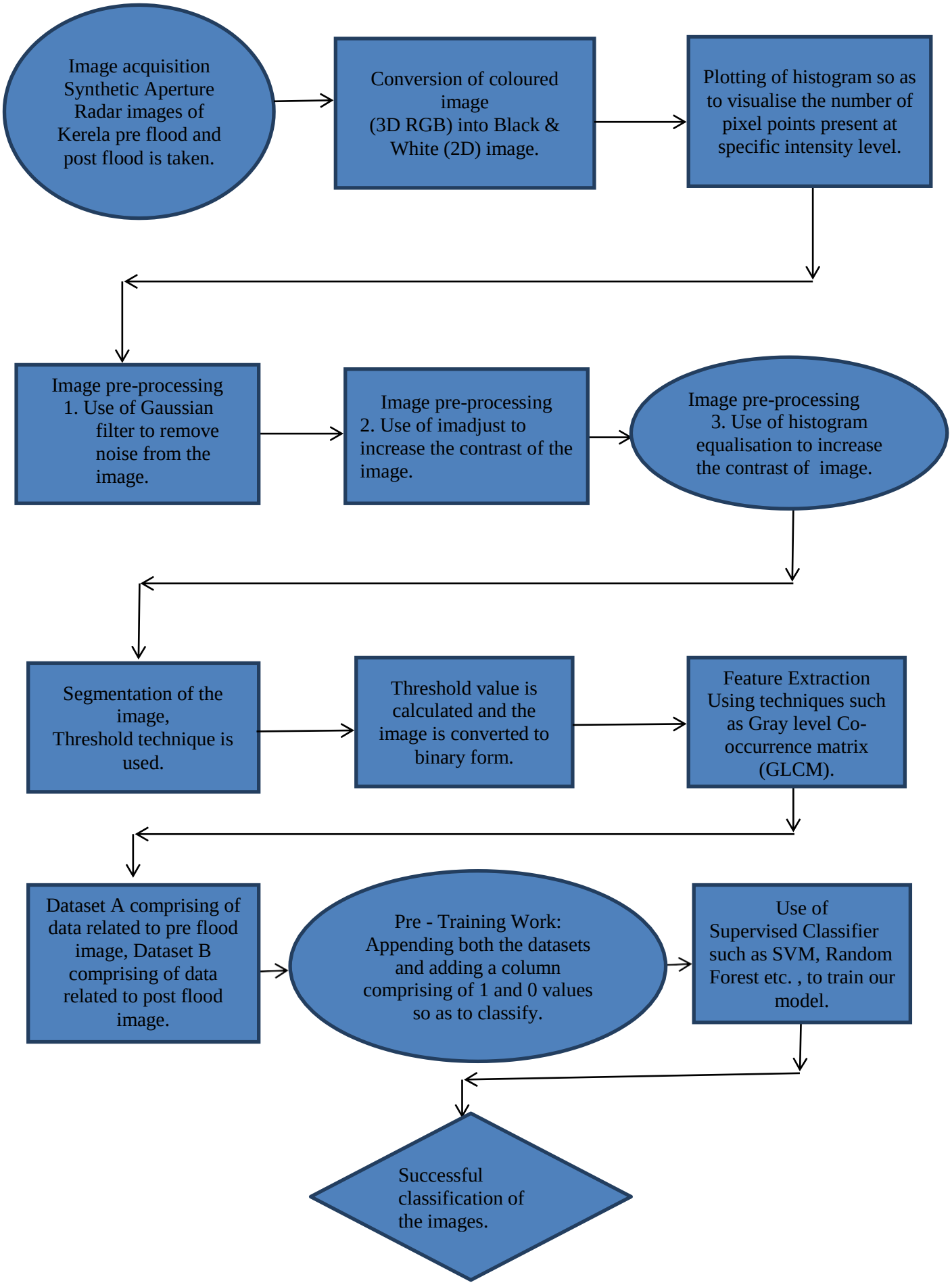


Image acquisition Synthetic Aperture Radar images of Kerela pre flood and post flood is taken.
Conversion of coloured image (3D RGB) into Black & White (2D) image.
Plotting of histogram so as to visualise the number of pixel points present at specific intensity level.
Image pre-processing 1. Use of Gaussian filter to remove noise from the image.
Image pre-processing 2. Use of imadjust to increase the contrast of the image.
Image pre-processing 3. Use of histogram equalisation to increase the contrast of image.
Segmentation of the image, Threshold technique is used.
Threshold value is calculated and the image is converted to binary form.
Feature Extraction Using techniques such as Gray level Co-occurrence matrix (GLCM).
Dataset A comprising of data related to pre flood image, Dataset B comprising of data related to post flood image.
Pre - Training Work: Appending both the datasets and adding a column comprising of 1 and 0 values so as to classify.
Use of Supervised Classifier such as SVM, Random Forest etc. , to train our model.
Successful classification of the images.

ALGORITHM USED & CONCEPTS:

Most of our work has been explained on the block diagram and it kind of gives idea about our work methodology so this section explains in depth analysis of the algorithm implemented in this project.

SEGMENTATION:

Image segmentation is the process of partitioning an image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries in images. The motive behind segmentation was to partition the SAR flood image into its primary components so that it becomes easy and meaningful for analysis of an image. More precisely we wanted to separate the image into two primary components one having water and one without water. We have used thresholding for the image which is most common and effective method for segmentation. In thresholding process we selected a threshold value to convert gray scale image to binary image, the threshold value was calculated using histogram equalization. graythresh function was used on histogram equalized image to get binary image or segmented image.

Result of segmentation has been added in out result section.

GRAY LEVEL CO-OCCURRENCE MATRIX:

Image texture can be extracted using complex visual patterns. Texture can be described as spatial arrangement of colors or intensities in an image or a sub region of an image.

GLCM is common method of texture analysis. In simple words it can explained as matrix which stores how often different combination of gray scale appear in original image. It is helpful in predicting variation of intensities in an image.

Properties of GLCM:

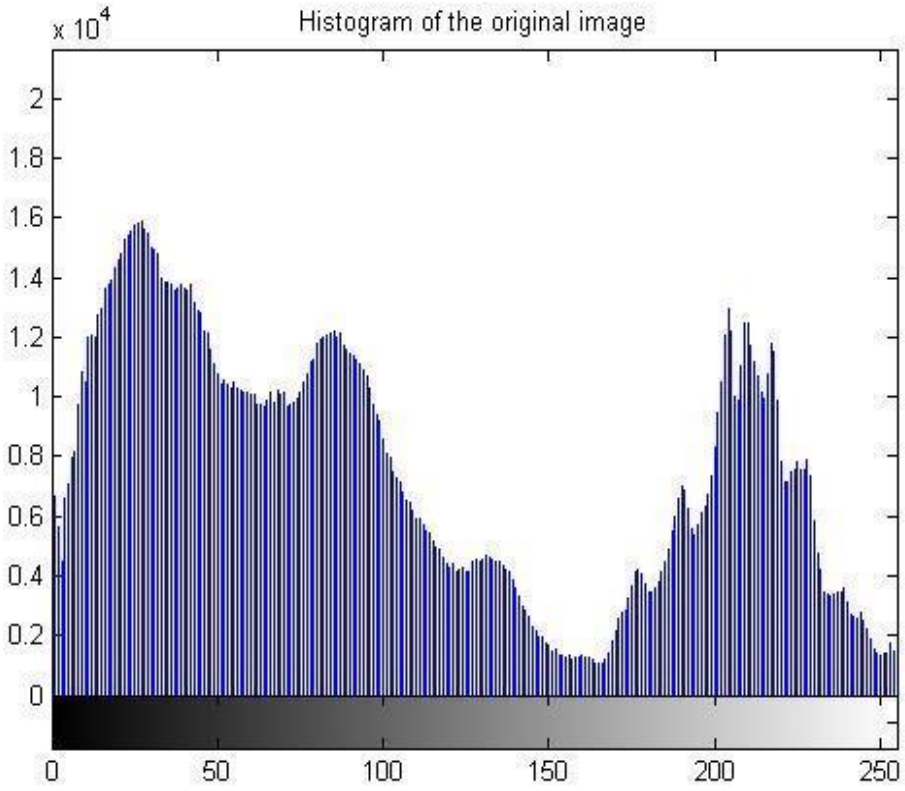
1. It is square matrix.
2. It is symmetric around diagonal.
3. Every element at (i , j) represents joint probabilities of gray scale at that position in original image.

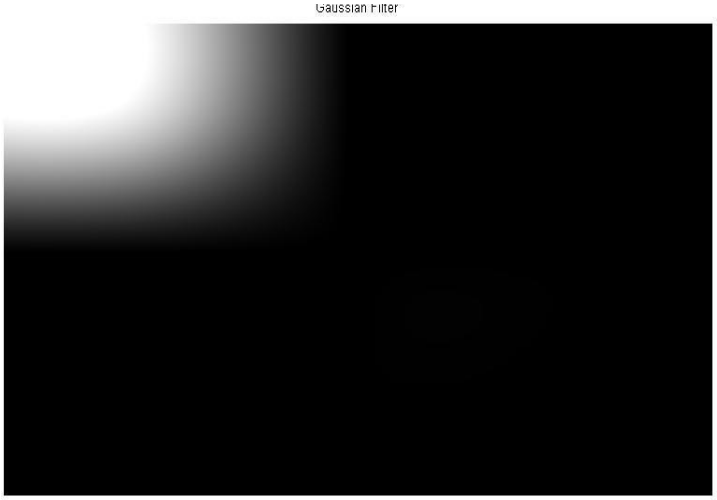
GLCM matrix can be calculated using MATLAB inbuilt functions. To make this image symmetric we simply add transpose to it. To further extract features we use statistical second order method of functions available in MATLAB like energy, dissimilarity, homogeneity etc. We made a dataset of 300 * 13 size by applying GLCM method on one or more images so as to make sure dataset is more balanced. 13 columns represent 13 features extracted from image and it was normalized in excel sheet by dividing every element by maximum value of every column. We then assigned 0 to pre flood image and 1 for post flood image in the dataset.

RANDOM FOREST CLASSIFIER : A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The dataset earlier formed was used to train our model using Random forest classifier. General steps of the algorithm were performed like splitting data set into train set and test set. Test set was chosen as 30%.Then feature scaling was performed and fitted random forest classifier on test set and train set. Then we found accuracy of our model to be around 65% by using confusion matrix. Model also worked on random input outside of datasets.

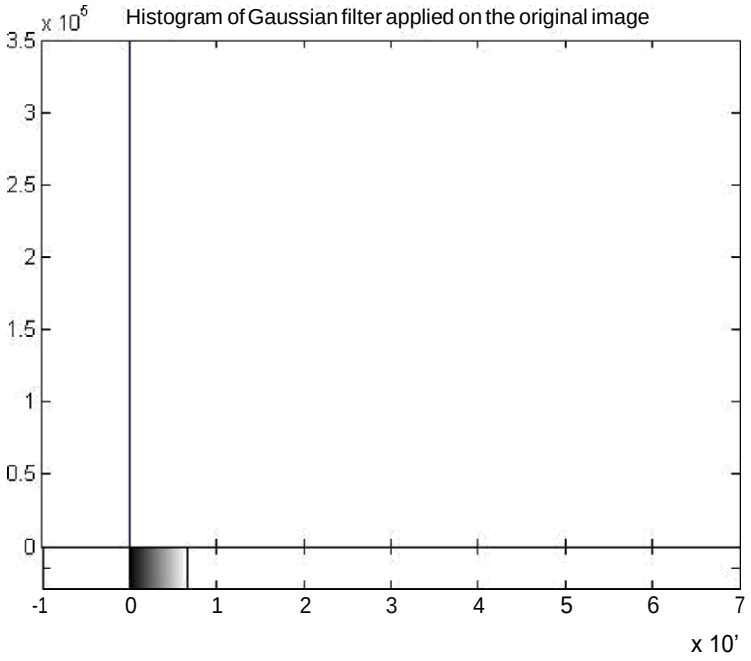
RESULTS:

Pre Flood Image :

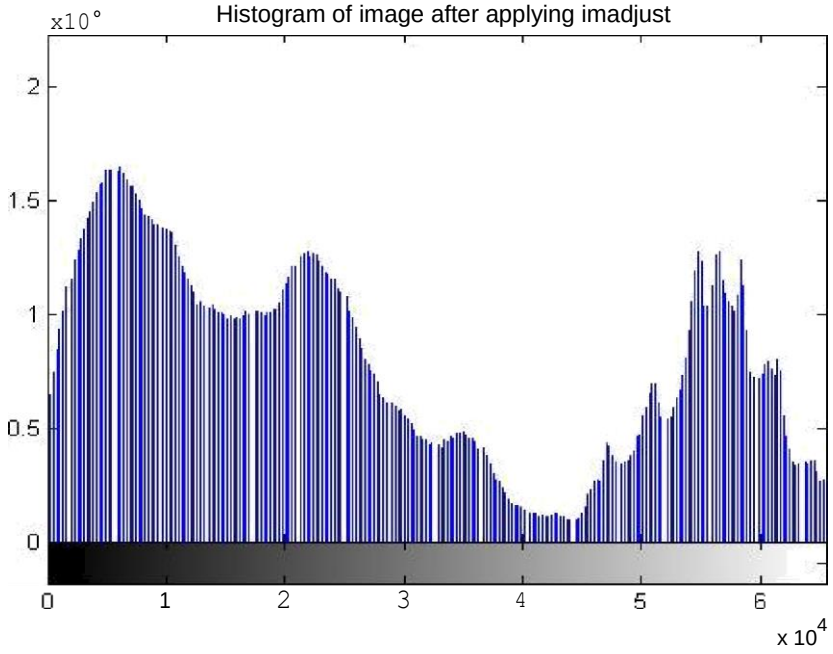


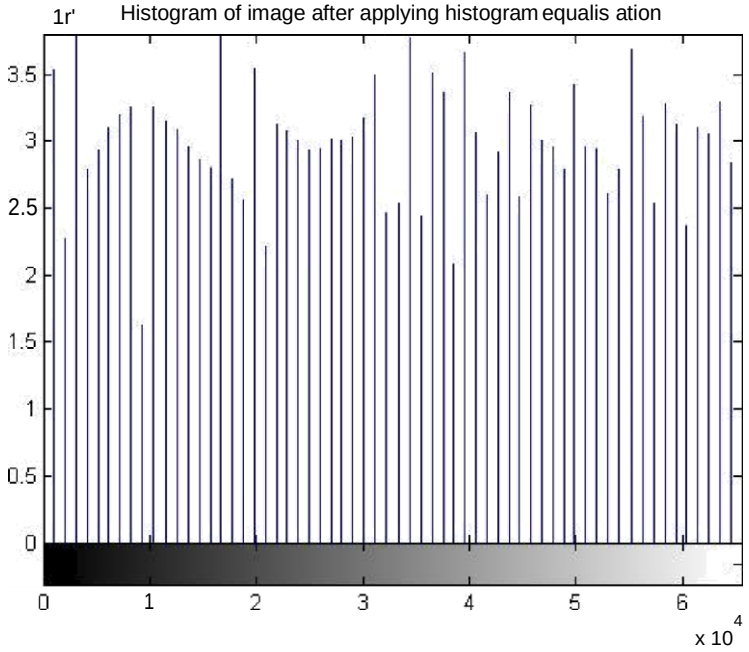


Histogram of Gaussian filter applied on the original image



Histogram of image after applying imadjust



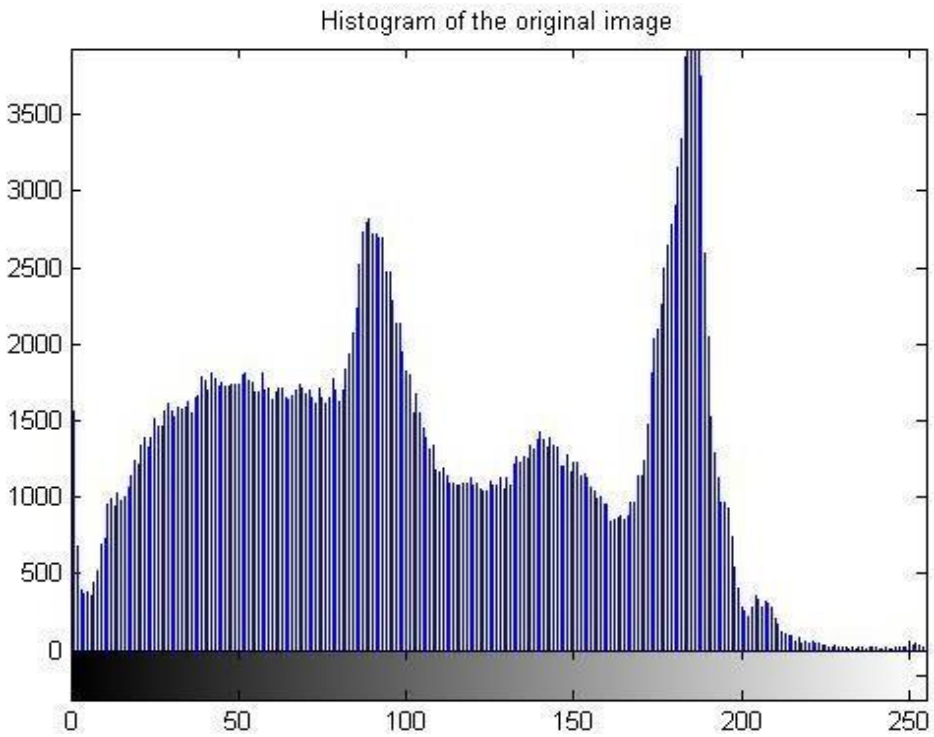


Original image

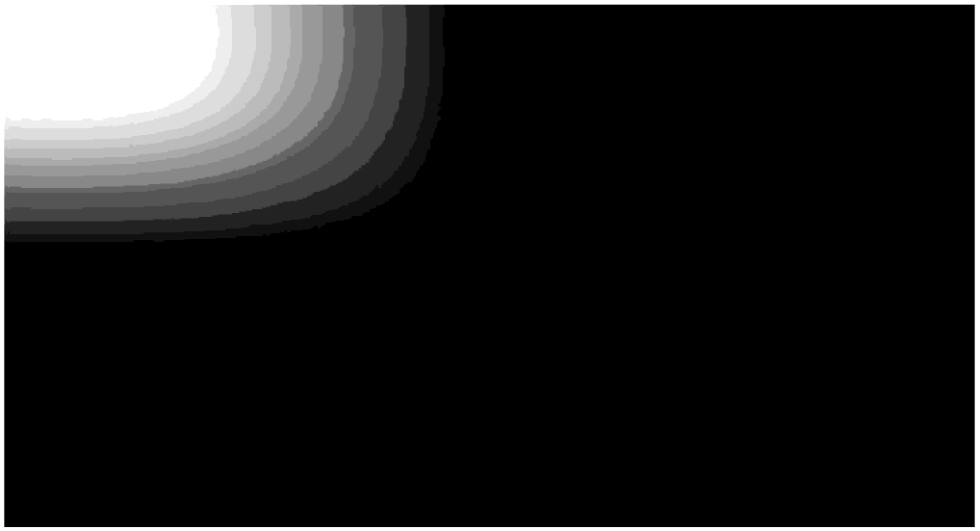
Image acer applying Histogram equalis ation

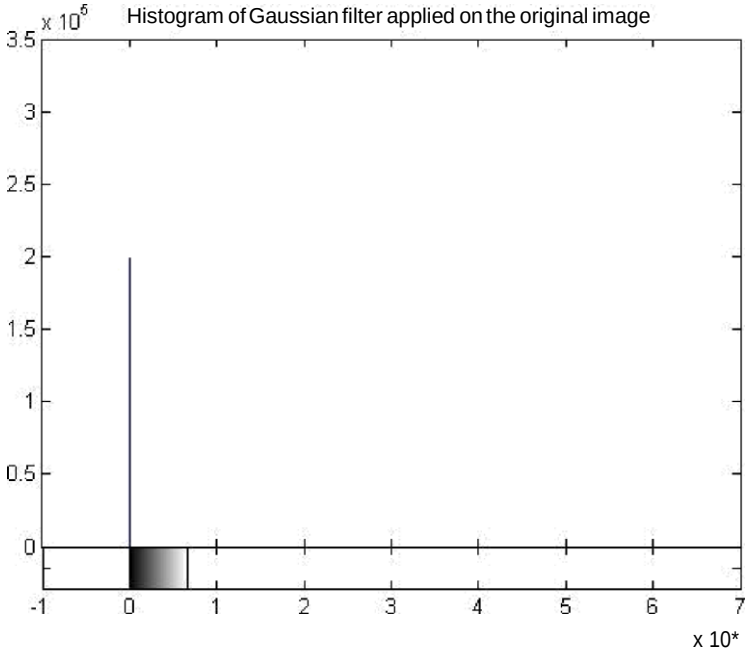
Segmented image

Post Flood :

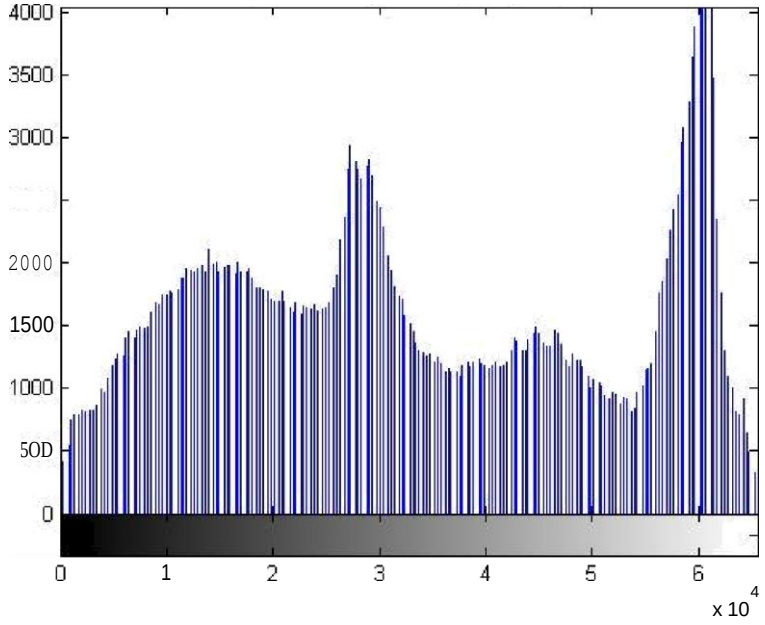


Gaus sian Filt el

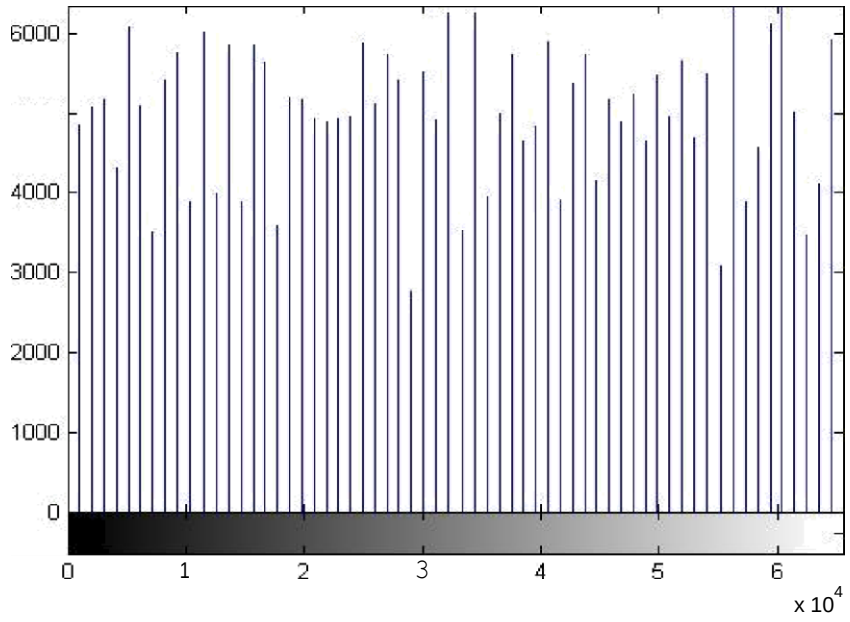




Histogram of image after applying imadjust



Histogram of image after applying histogram equalisation



Original image

Image after applying Histogram equalisation

Segmented image

DATASET USED:

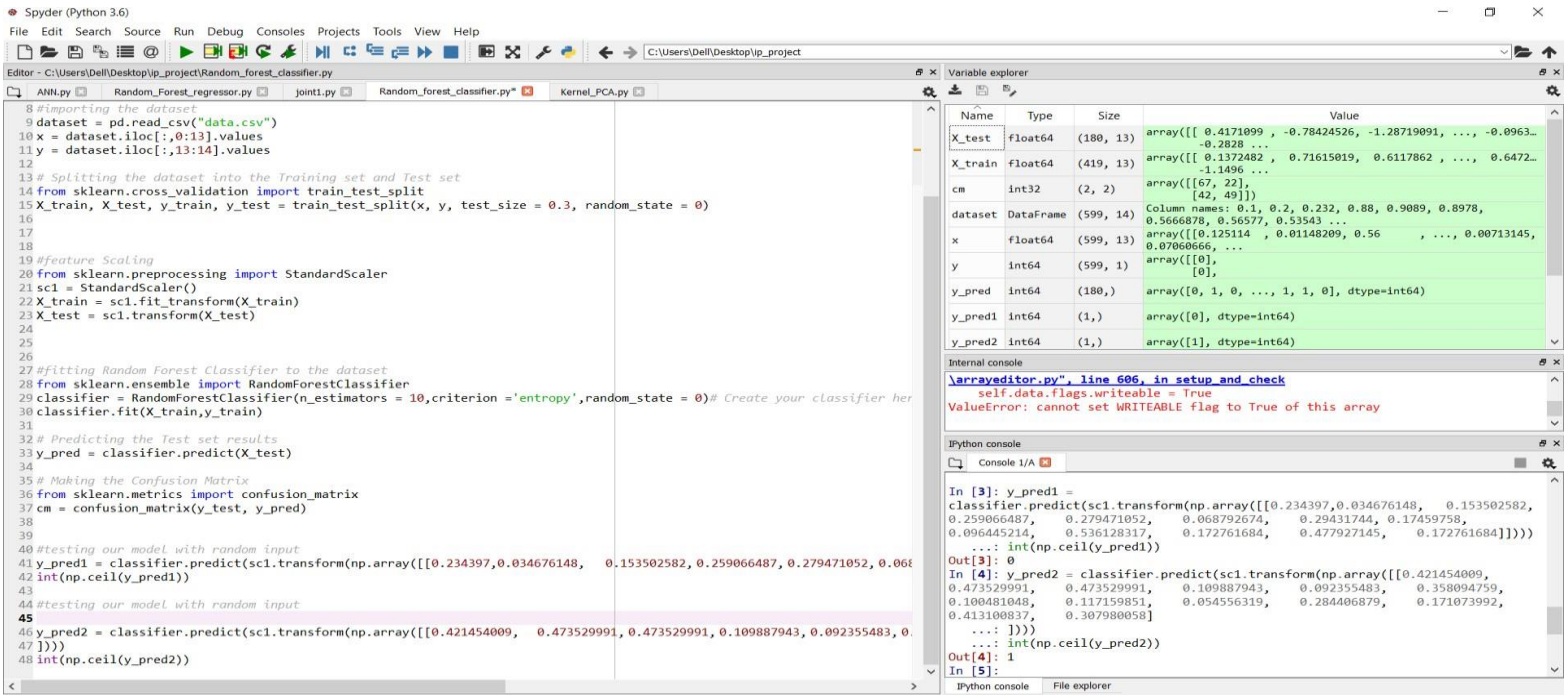
1) FLOOD IMAGE :

0.454	0.1232	0.123212	0.133243	0.343214	0.324324	0.32424	0.43534	0.345435	0.324234	0.634535	0.345456	0.46546	1
0.471921	0.305494	0.305494	0.149477	0.21046	0.274463	0.304675	0.171354	0.404049	0.114103	0.141252	0.048428	0.081721	1
0.005748	0.555148	0.555148	0.360993	0.160039	0.307648	0.06938	0.198188	0.190045	0.000657	0.356151	0.215236	0.160496	1
0.19234	0.14172	0.14172	0.150133	0.118261	0.838172	0.105092	0.17503	0.069325	0.153201	0.252077	0.100941	0.123865	1
0.197094	0.23417	0.23417	0.668574	0.00762	0.195699	0.037457	0.068884	0.123044	0.058902	0.480891	0.249219	0.367431	1
0.047157	0.129449	0.129449	0.481341	0.253789	0.200522	0.147366	0.0047	0.155769	0.181891	0.185376	0.031707	0.02023	1
0.321512	0.050926	0.050926	0.48175	0.411161	0.245364	0.218779	0.024745	0.040304	0.16135	0.301797	0.124314	0.377885	1
0.125672	0.111406	0.111406	0.229247	0.224646	0.012122	0.118586	0.24556	0.237821	0.155655	0.146097	0.05134	0.152905	1
0.154717	0.393411	0.393411	0.242499	0.164173	0.378416	0.26087	0.202517	0.109274	0.10874	0.071137	0.333249	0.040641	1
0.130797	0.124595	0.124595	0.204538	0.08524	0.061601	0.336568	0.088314	0.515516	0.166891	0.387257	0.01623	0.064158	1
0.208366	0.564965	0.564965	0.187395	0.030406	0.229184	0.092928	0.51141	0.334459	0.14942	0.031396	0.329525	0.120723	1
0.273638	0.258465	0.258465	0.230337	0.333468	0.161476	0.027933	0.05169	0.087848	0.056852	0.169971	0.142813	0.140842	1
0.445222	0.23123	0.23123	0.292025	0.048324	0.228916	0.177599	0.046657	0.299099	0.159395	0.390995	0.190334	0.213017	1
0.00729	0.086703	0.086703	0.189018	0.465607	0.358433	0.06608	0.296361	0.296371	0.079175	0.035508	0.515118	0.186433	1
0.346758	0.014705	0.014705	0.009739	0.575028	0.250187	0.281156	0.271334	0.128797	0.319582	0.381552	0.700691	0.379131	1
0.253909	0.398219	0.398219	0.1881	0.615759	0.345121	0.163656	0.132014	0.305431	0.348446	0.249183	0.183532	0.173414	1
0.196894	0.068606	0.068606	0.188674	0.046123	0.243533	0.221143	0.481999	0.454067	0.105207	0.119645	0.116172	0.068635	1
0.197012	0.053592	0.053592	0.213101	0.137515	0.147616	0.077407	0.004391	0.107777	0.071557	0.045441	0.237814	0.015618	1
0.211336	0.238367	0.238367	0.055882	0.460279	0.139527	0.195934	0.233906	0.229403	0.039907	0.107594	0.06772	0.066748	1
0.286978	0.183019	0.183019	0.26831	0.088878	0.078238	0.079635	0.004017	0.165518	0.236968	0.179715	0.062804	0.361554	1
0.202861	0.255384	0.255384	0.336222	0.022796	0.383262	0.098463	0.322027	0.123832	0.074265	0.128187	0.164822	0.164367	1
0.186024	0.090776	0.090776	0.116775	0.234581	0.294978	0.506485	0.592184	0.393932	0.12496	0.046901	0.080171	0.107011	1
0.044143	0.670549	0.670549	0.144129	0.195313	0.091614	0.181075	0.282152	0.001637	0.369324	0.213642	0.17078	0.166654	1
0.215545	0.16213	0.16213	0.135515	0.62039	0.048641	0.672617	0.099241	0.679468	0.048555	0.155382	0.093854	0.035898	1
0.054871	0.02714	0.02714	0.176904	0.337119	0.558219	0.215245	0.172031	0.001203	0.148431	0.059346	0.014466	0.22393	1
0.495388	0.20477	0.20477	0.742884	0.288337	0.274088	0.021633	0.003298	0.08803	0.134325	0.192592	0.504811	0.142875	1
0.093834	0.595642	0.595642	0.317542	0.259021	0.588461	0.072796	0.307553	0.258014	0.0699	0.008354	0.531093	0.010963	1
0.13432	0.395962	0.395962	0.199796	0.113173	0.062907	0.403572	0.013904	0.586411	0.283866	0.280115	0.394417	0.142799	1
0.241988	0.321832	0.321832	0.036858	0.050035	0.304898	0.004986	0.16421	0.340734	0.097666	0.052309	0.169592	0.218151	1
0.053276	0.001688	0.001688	0.137858	0.098315	0.011932	0.306343	0.289583	0.086354	0.260992	0.576332	0.401734	0.207984	1
0.205626	0.0218	0.0218	0.236037	0.077758	0.367664	0.19303	0.030262	0.201214	0.247778	0.168477	0.18315	0.055631	1
0.077906	0.265999	0.265999	0.694255	0.104768	0.263601	0.306615	0.157605	0.405323	0.035152	0.261375	0.029854	0.113213	1

2) NORMAL IMAGE (WITHOUT FLOOD) :

0.1	0.2	0.232	0.88	0.9089	0.8978	0.566688	0.56577	0.53543	0.3434	0.24324	0.123213	0.345345	0
0.125114	0.011482	0.56	0.158479	0.9089	0.063205	0.022214	0.009737	0.125554	0.116652	0.007131	0.070607	0.184539	0
0.475526	0.122738	0.09844	0.12179	0.9089	0.111126	0.254731	0.027969	0.233445	0.083776	0.144329	0.077331	0.495071	0
0.271475	0.197359	0.215818	0.350205	0.9089	0.018777	0.391138	0.347278	0.032215	0.14694	0.39792	0.175592	0.295943	0
0.01324	0.003282	0.007298	0.315723	0.9089	0.025322	0.209336	0.133617	0.025069	0.23898	0.032076	0.19287	0.315926	0
0.200958	0.194327	0.205386	0.069659	0.9089	0.173979	0.337454	0.216449	0.067713	0.036162	0.587988	0.199631	0.254189	0
0.122327	0.132338	0.567282	0.010254	0.9089	0.064647	0.07174	0.222122	0.232443	0.036797	0.103151	0.464327	0.736005	0
0.164145	0.091198	0.435601	0.538373	0.9089	0.119764	0.352941	0.326163	0.212423	0.170096	0.145208	0.320066	0.149641	0
0.194331	0.661077	0.019121	0.140524	0.9089	0.241152	0.232849	0.211125	0.144119	0.048951	0.245144	0.004055	0.103766	0
0.058729	0.164101	0.00621	0.352828	0.9089	0.206858	0.087517	0.296324	0.086041	0.563844	0.035263	0.118427	0.238125	0
0.266926	0.021587	0.280299	0.210528	0.160793	0.167896	0.019869	0.196961	0.274	0.315626	0.097636	0.40912	0.044578	0
0.072824	0.113243	0.697532	0.317472	0.336151	0.175265	0.226525	0.397235	0.17655	0.287042	0.020974	0.220938	0.376725	0
0.094579	0.142273	0.183646	0.001154	0.148014	0.210073	0.328717	0.291444	0.077877	0.167767	0.302945	0.017797	0.339648	0
0.038748	0.065609	0.174684	0.274423	0.151703	0.051077	0.097564	0.005295	0.272572	0.375202	0.175596	0.017793	0.135715	0
0.280002	0.418323	0.23399	0.286492	0.191377	0.260344	0.014962	0.07106	0.121802	0.010356	0.043039	0.084539	0.339846	0
0.116803	0.047581	0.302192	0.081447	0.30094	0.193689	0.386717	0.140066	0.255632	0.194768	0.024127	0.023338	0.238829	0
0.182235	0.281239	0.219623	0.280555	0.550592	0.029752	0.326765	0.071796	0.276516	0.359175	0.062066	0.056597	0.100778	0
0.02157	0.258984	0.302079	0.330433	0.018814	0.10721	0.498294	0.046155	0.494921	0.47245	0.357589	0.031381	0.144612	0
0.014555	0.282195	0.106262	0.485127	0.199414	0.383234	0.299389	0.487224	0.28709	0.349545	0.034823	0.060245	0.044534	0
0.099762	0.368882	0.175586	0.347034	0.021095	0.05509	0.134106	0.068398	0.037211	0.032349	0.05119	0.205928	0.155356	0
0.33244	0.178331	0.11097	0.032431	0.420331	0.608792	0.055269	0.011376	0.033005	0.067235	0.044692	0.286393	0.109649	0
0.208327	0.175918	0.283841	0.254653	0.402332	0.148073	0.016073	0.130178	0.259414	0.313105	0.262998	0.176801	0.26795	0
0.192653	0.208257	0.015534	0.258042	0.186997	0.266393	0.022522	0.215361	0.44175	0.437783	0.100386	0.046222	0.174923	0
0.099483	0.03646	0.381816	0.298941	0.238063	0.010675	0.130608	0.028644	0.265832	0.449612	0.363486	0.160178	0.270254	0
0.221756	0.230257	0.529238	0.004252	0.23568	0.064865	0.122736	0.182981	0.178214	0.19677	0.274581	0.381639	0.148207	0
0.120387	0.030585	0.31912	0.293794	0.292136	0.161379	0.054898	0.147074	0.256719	0.816377	0.205693	0.342889	0.102316	0
0.181461	0.486326	0.19369	0.038108	0.168242	0.281711	0.084158	0.243212	0.13587	0.073151	0.40825	0.19977	0.045762	0
0.10176	0.390095	0.102046	0.131469	0.280774	0.087674	0.196494	0.300134	0.063734	0.155823	0.003912	0.477034	0.160351	0
0.136322	0.028891	0.215475	0.169308	0.115923	0.100946	0.032093	0.189086	0.169561	0.129645	0.192027	0.420322	0.123506	0
0.01194	0.327067	0.193444	0.320909	0.508001	0.12221	0.131964	0.060373	0.278215	0.014727	0.11345	0.099886	0.009171	0
0.092561	0.303647	0.001221	0.099382	0.092014	0.424822	0.087625	0.232574	0.114223	0.525278	0.345697	0.113702	0.137817	0
0.091471	0.122269	0.068068	0.259578	0.219917	0.068356	0.000442	0.073033	0.076852	0.032254	0.219647	0.249918	0.094282	0

3) MACHINE LEARNING RESULT



4) PREDICTED RESULT FOR RANDOM INPUT

```
In [3]: y_pred1 =
classifier.predict(sc1.transform(np.array([[0.234397,0.034676148,   0.153502582,
0.259066487,    0.279471052,    0.068792674,    0.29431744, 0.17459758,
0.096445214,    0.536128317,    0.172761684,    0.477927145,    0.172761684]])))
   ...: int(np.ceil(y_pred1))
Out[3]: 0
In [4]: y_pred2 = classifier.predict(sc1.transform(np.array([[0.421454009,
0.473529991,    0.473529991,    0.109887943,    0.092355483,    0.358094759,
0.100481048,    0.117159851,    0.054556319,    0.284406879,    0.171073992,
0.413100837,    0.307980058]
   ...: ])))
   ...: int(np.ceil(y_pred2))
Out[4]: 1
In [5]:
```

CONCLUSION:

By performing this project of image processing we got to learn many important concepts related to images and machine learning. We learnt to apply histogram equalization on an image to change the contrast of the image and thus increased the dynamic range of our image which helped in doing thresholding technique for image segmentation effectively and the features were extracted from images efficiently using GLCM. And thus dataset made for this algorithm helped in classification of an image using Random forest classifier. Overall we learnt how to process an image, extract features, and how to create dataset out of that for ML algorithm, so it was quite good experience.

FUTURE SCOPE:

1) Deep learning algorithm can be implemented instead of Random Forest Classifier to give more accurate results.
2) Thresholding technique can give some offset so we can opt for better technique for image segmentation.

BIBLIOGRAPHY :

Took help from 'Digital Image Processing' 3rd Edition by R.C Gonzalez and Richard E. Woods and 'NPTEL'.

APPENDIX :

MATLAB CODE:

PRE FLOOD:

```
% Read the pre flood image.
originalImage = imread('C:\Users\Dell\Desktop\ip_project\unflood1.jpg');

% Conversion of coloured (3D Image) to Black and White Image(2D image).
I = rgb2gray(originalImage);

% Conversion to double data type
db = uint16(I);

% Creating a Gaussian filter of size 3*3 and sigma = 2
smooth = fspecial('gaussian',[3,3],2);

% Applying that filter onto the image
filt = imfilter(db,smooth,'replicate');

% Increasing the contrast of the image
img1 = imadjust(filt);

% Increasing the contrast of the image using histogram equalisation of
% remaining part
img = histeq(img1);

% Calculating the threshold value
T = graythresh(img);

% Conversion of image in binary form based on the threshold value
S = im2bw(img,T);

% Taking the filter in frequency domain so as to see how filter was made
filtf=fft2(smooth);

% Resizing the filter to visualise it properly
filtf_1=imresize(filtf,[422,759]);

% Plotting of histogram to visualise no. of pixel points vs. intensity
% values.
figure(1);
```

```
imhist(I);
title('Histogram of the original image');

% Displaying the filter
figure(2);
imshow(filtf_1);
title('Gaussian Filter');

% Plotting the histogram of output of mask applied on the image
figure(3);
imhist(filt);
axis([-1*(10e4),7*(10e4),0*(10e4),3.5*(10e4)]);
title('Histogram of Gaussian filter applied on the original image');

% Displaying of the histogram of increased contrast image
figure(4);
imhist(img1);
title('Histogram of image after applying imadjust');

% Displaying of the histogram of increased contrast image
figure(5);
imhist(img);
title('Histogram of image after applying histogram equalisation');

% Displaying the results
figure(6);
imshow(I);
title('Original image');
figure(7);
imshow(img1);
title('Image after applying imadjust');
figure(8);
imshow(img);
title('Image after applying Histogram equalisation');
figure(9);
imshow(S);
title('Segmented image');
```

POST FLOOD:

```
% Read the post flood image.
originalImage = imread('C:\Users\Dell\Desktop\ip_project\flood4.jpg');

% Conversion of coloured (3D Image) to Black and White Image(2D image).
I = rgb2gray(originalImage);

% Conversion to double data type
db = uint16(I);

% Creating a Gaussian filter of size 3*3 and sigma = 2
smooth = fspecial('gaussian',[3,3],2);

% Applying that filter onto the image
filt = imfilter(db,smooth,'replicate');

% Increasing the contrast of the image
img1 = imadjust(filt);

% Increasing the contrast of the image using histogram equalisation of
% remaining part
img = histeq(img1);

% Calculating the threshold value
T = graythresh(img);
```

```
% Conversion of image in binary form based on the threshold value
S = im2bw(img,T);

% Taking the filter in frequency domain so as to see how filter was made
filtf=fft2(smooth);

% Resizing the filter to visualise it properly
filtf_1=imresize(filtf,[422,759]);

% Plotting of histogram to visualise no. of pixel points vs. intensity
% values.
figure(1);
imhist(I);
title('Histogram of the original image');

% Displaying the filter
figure(2);
imshow(filtf_1);
title('Gaussian Filter');

% Plotting the histogram of output of mask applied on the image
figure(3);
imhist(filt);
axis([-1*(10e4),7*(10e4),0*(10e4),3.5*(10e4)]);
title('Histogram of Gaussian filter applied on the original image');

% Displaying of the histogram of increased contrast image
figure(4);
imhist(img1);
title('Histogram of image after applying imadjust');

% Displaying of the histogram of increased contrast image
figure(5);
imhist(img);
title('Histogram of image after applying histogram equalisation');

% Displaying the results
figure(6);
imshow(I);
title('Original image');
figure(7);
imshow(img1);
title('Image after applying imadjust');
figure(8);
imshow(img);
title('Image after applying Histogram equalisation');
figure(9);
imshow(S);
title('Segmented image');
```

CODE FOR GLCM AND FEATURE EXTRACTION:

```
clc;
clearall;
closeall;

% texture analysis using GLCM and then feature extraction

% folder in which your images exists
location = ('C:\Users\Dell\Desktop\ip_project\seg_beforeflood.PNG');

% describing the data and specifying how to read the data from the datastore.
ds = imageDatastore(location);

% quantifying spatial intensity using one of the parameter of GLCM
offsets = [0 1; -1 1;-1 0;-1 -1];
```

```
% defining the empty matrix
  G=[];

% hasdata returns 1 if there is data available to read from the datastore specified by ds .
whilehasdata(ds)

% read image from datastore
     i = read(ds) ;

% storing the dimensions of our image
     [rows, columns, numberOfColorChannels] = size(i);

% checking no of planes in image
ifnumberOfColorChannels> 1

% Convert it to gray scale by taking only the one channel.
          i = i(:, :, 2);
end


    glcm1 = graycomatrix(i,'numlevels',16,'offset',[0 1])

% extracting features using the 2nd order statistical functions like
% dissimilarity,entropyetc is used
   [out] = GLCM_Features1(glcm1);

A=[out.autocout.controut.corrmout.corrpout.cpromout.cshadout.dissiout.energout.entroout.homomout.homopout.maxprout.sosvhout.savghout.svarhout.senthout.dvarhout.denth out.inf1h out.inf2h out.indncout.idmnc];

   glcm2 = graycomatrix(i,'numlevels',16,'offset',[-1,1]);

% extracting features using the 2nd order statistical functions like
% dissimilarity,entropyetc is used
   [out1] = GLCM_Features1(glcm2);
   B=[out1.autoc out1.contr out1.corrm out1.corrp out1.cprom out1.cshad out1.dissi out1.energ out1.entro out1.homom out1.homop out1.maxpr out1.sosvh out1.savgh out1.svarh out1.senth out1.dvarh out1.denth out1.inf1h out1.inf2h out1.indnc out1.idmnc];

   glcm3 = graycomatrix(i,'numlevels',16,'offset',[-1,0]);
% extracting features using the 2nd order statistical functions like
% dissimilarity,entropyetc is used
  [out2] = GLCM_Features1(glcm3);
    C=[out2.autoc out2.contr out2.corrm out2.corrp out2.cprom out2.cshad out2.dissi out2.energ out2.entro out2.homom out2.homop out2.maxpr out2.sosvh out2.savgh out2.svarh out2.senth out2.dvarh out2.denth out2.inf1h out2.inf2h out2.indnc out2.idmnc];

   glcm4= graycomatrix(i,'numlevels',16,'offset',[-1,0]);
% extracting features using the 2nd order statistical functions like
% dissimilarity,entropyetc is used
   [out3] = GLCM_Features1(glcm4);
   D=[out3.autoc out3.contr out3.corrm out3.corrp out3.cprom out3.cshad out3.dissi out3.energ out3.entro out3.homom out3.homop out3.maxpr out3.sosvh out3.savgh out3.svarh out3.senth out3.dvarh out3.denth out3.inf1h out3.inf2h out3.indnc out3.idmnc];
   M=[ A ,B,C,D];

% the final matrix formed
    G=[G;M];


end
```

CODE FOR RANDOM FOREST CLASSIFIER:

```
# Importing the libraries
importnumpy as np
importmatplotlib.pyplot as plt
import pandas as pd


#importing the dataset
dataset = pd.read_csv("data.csv")
x = dataset.iloc[:,0:13].values
y = dataset.iloc[:,13:14].values

# Splitting the dataset into the Training set and Test set
fromsklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)



#feature Scaling
fromsklearn.preprocessing import StandardScaler
sc1 = StandardScaler()
X_train = sc1.fit_transform(X_train)
X_test = sc1.transform(X_test)



#fitting Random Forest Classifier to the dataset
fromsklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10,criterion ='entropy',random_state = 0)# Create your classifier here
classifier.fit(X_train,y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
fromsklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)


#testing our model with random input
y_pred1 = classifier.predict(sc1.transform(np.array([[0.234397,0.034676148,    0.153502582,        0.259066487,
            0.279471052,        0.068792674,        0.29431744,         0.17459758,         0.096445214,        0.536128317,
            0.172761684,        0.477927145,        0.172761684]])))
int(np.ceil(y_pred1))

#testing our model with random input

y_pred2 = classifier.predict(sc1.transform(np.array([[0.421454009, 0.473529991,         0.473529991,        0.109887943,
            0.092355483,        0.358094759,        0.100481048,        0.117159851,        0.054556319,        0.284406879,
            0.171073992,        0.413100837,        0.307980058]
])))
int(np.ceil(y_pred2))
```