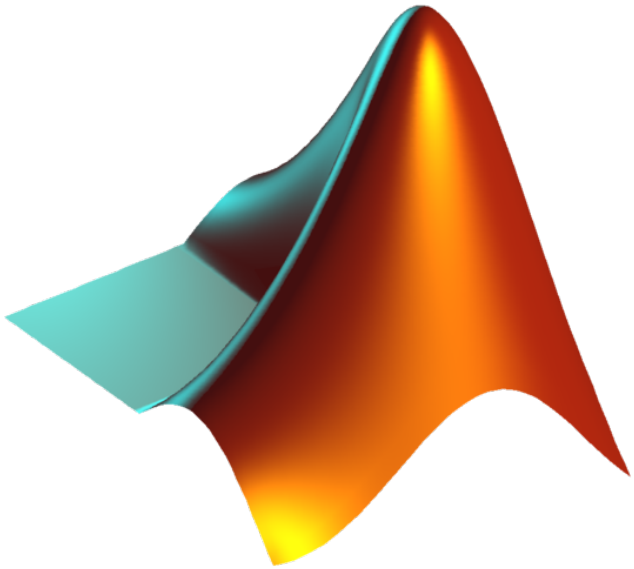


IMAGE & VIDEO ANALYTICS

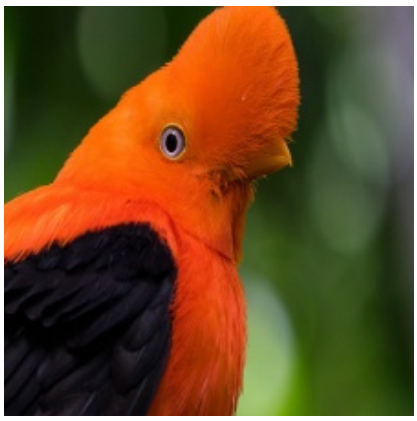
2048015 MANOJ KUMAR
2MDS

PROBLEM DESCRIPTION

From an ecological and environmental point of view, monitoring bird diversity is an important task. While bird monitoring is a well-established process, the observation is largely carried out manually which is time-consuming, and hence the scalability is low. This has motivated the use of machine learning methods to analyze bird images and sounds, using camera-trap data, recorded data or crowd-sourcing. In this challenge, the bird image classification task, especially for Himalayan birds, based on a limited but a diverse set of crowd-sourced data. Especially, the present challenge involves a fairly low amount of labelled data and may require transfer learning based approaches for effective classification.

ABOUT THE DATA

The model was trained and tested for 4 species of birds with the total images 532, it was divided into 372 training dataset and 160 testing images for test respectively and the model has shown a promising of 85% accuracy score when treated with refined datasets.

	
COCK OF THE ROCK	**COCKATOO**
	
CROW	**GOLDEN BROWED**

FUNCTION USED TO EXTRACT SPATIAL DOMAIN FEATURES

Importing shared dataset

```
mydir='/MATLAB Drive/Mini Project/Dataset/Class 1';
fileformat='*.jpg';

dd=dir(fullfile(mydir,fileformat));
assert(numel(dd) > 0, 'No file was found. Check that the path is
correct');
my_img = struct('img', cell(size(dd)));
k=numel(dd)+1;

for zz=1:numel(dd)
 my_img(zz).img = imread(fullfile(mydir,dd(zz).name));
end
```

Spatial Feature Extraction using MATLAB

```
for i=1:numel(dd)
    c_r_1=0;
    c_r_2=0;
    c_g_1=0;
    c_g_2=0;
    c_b_1=0;
    c_b_2=0;
    current=imresize(my_img(i).img,[400,400]);

%rgb means
    r=mean(mean(current(:,:,1)));
    g=mean(mean(current(:,:,2)));
    b=mean(mean(current(:,:,3)));

%Grayscale
    g_img1=rgb2gray(current);
    g_img=double(edge(g_img1,'Canny',0.2));

%Statistical measures
    av=mean(mean(g_img));
    med=median(median(g_img));
    st_dev=std(std(double(g_img)));
    max_=max(max(g_img));
    min_=min(min(g_img));

    sk_=real(skewness(skewness(g_img)));
    ku_=real(kurtosis(kurtosis(g_img)));
    iqr_=iqr(iqr(g_img));

    FeatureData = regionprops(g_img, 'all');

    % Area,
    % BoundingBox,
    % MajorAxisLength,
    % MinorAxisLength,
    % Eccentricity,
    % Orientation,
```

```
    % FilledArea,
    % EulerArea,
    % EquivDiameter,
    % Solidity,
    % perimeter

    Array = [FeatureData.Area];
    Area_Mean = mean(Array);
    Area_Std = std(Array);

    Array = [FeatureData.BoundingBox];
    BB_Mean = mean(Array);
    BB_Std = std(Array);
    BB_sk_=real(skewness(Array));
    BB_ku_=real(kurtosis(Array));

    Array = [FeatureData.Eccentricity];
    p_Mean = mean(Array);
    p_Std = std(Array);

    Array = [FeatureData.FilledArea];
    ea_Mean = mean(Array);
    ea_Std = std(Array);

    [M,N]=size(g_img);
    area=M*N;

%Entropy values
    e=entropy(g_img);

%Above & Below
    for i = 1:M
        for j = 1:N

            R=current(i,j,1);
            G=current(i,j,2);
            B=current(i,j,3);
            if(R>r)
                c_r_1=c_r_1+1;
            end
            if(G>g)
                c_g_1=c_g_1+1;
            end
            if(B>b)
                c_b_1=c_b_1+1;
            end
            if(R<r)
                c_r_2=c_r_2+1;
            end
            if(G<g)
                c_g_2=c_g_2+1;
            end
            if(B<b)
                c_b_2=c_b_2+1;
            end
        end
    end
```

```
    target = 0;

%Column Values
    rgb1=[r,g,b,av,med,st_dev,max_,min_,sk_,ku_,iqr_,...
        Area_Mean,Area_Std, BB_Mean, BB_Std, BB_sk_, BB_ku_,...
        p_Mean, p_Std, ea_Mean, ea_Std,...
        area,e,c_r_1,c_r_2,c_g_1,c_g_2,c_b_1,c_b_2,target];
%Writing into Excel Sheets
    writematrix(rgb1,'Spatial data.csv','WriteMode', 'append');
end
```

FUNCTION USED TO EXTRACT FREQUENCY DOMAIN FEATURES

Importing shared dataset

```
mydir='/MATLAB Drive/Mini Project/Dataset/Class 1';
fileformat='*.jpg';
dd=dir(fullfile(mydir,fileformat));
assert(numel(dd) > 0, 'No file was found. Check that the path is
correct');
my_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    my_img(zz).img = imread(fullfile(mydir,dd(zz).name));
end
```

```
%% CREATING A STRUCTURE FOR R COMPONENT

r_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    r_img(zz).img = my_img(zz).img(:,:,1);
end
%% CREATING A STRUCTURE FOR G COMPONENT

g_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    g_img(zz).img = my_img(zz).img(:,:,2);
end
%% CREATING A STRUCTURE FOR B COMPONENT

b_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    r_img(zz).img = my_img(zz).img(:,:,3);
end
%% CREATING A STRUCTURE FOR GRAY SCALE VERSION

gray_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    gray_img(zz).img = rgb2gray(my_img(zz).img);
end
%% CREATING A STRUCTURE FOR SHARPENED

edge_img = struct('img', cell(size(dd)));
for zz=1:numel(dd)
    edge_img(zz).img = fourrier(gray_img(zz).img,0.09,4);
end
```

CODE TO EXTRACT FEACTURES IN THE FREQUENCY DOMAIN AFTER APPLYING FFT

```
%Fast Fourier transform

for i=1:numel(dd)
    current=gray_img(i).img;

%Fourier transform
    fft_img=fft2(current);

%Statistical measures
    gray_av=real(mean(mean(fft_img)));            % Avg
    gray_med=real(median(median(fft_img)));       % Median
    gray_st_dev=real(std(std(double(fft_img))));  % S.D
    gray_max_=real(max(max(fft_img)));            % MAX
    gray_min_=real(min(min(fft_img)));            % MIN
    gray_mode_=real(mode(mode(fft_img)));
    gray_midpoint_=real(0.5*(gray_max_+gray_min_));
    gray_var_=real(var(var(fft_img)));

    current=r_img(i).img;
%Fourier transform
    fft_img=fft2(current);
%Statistical measures
    r_av=real(mean(mean(fft_img)));            % Avg
    r_med=real(median(median(fft_img)));       % Median
    r_st_dev=real(std(std(double(fft_img))));  % S.D
    r_max_=real(max(max(fft_img)));            % MAX
    r_min_=real(min(min(fft_img)));            % MIN
    r_mode_=real(mode(mode(fft_img)));
    r_midpoint_=real(0.5*(r_max_+r_min_));
    r_var_=real(var(var(fft_img)));

    current=g_img(i).img;
%Fourier transform
    fft_img=fft2(current);
%Statistical measures
    g_av=real(mean(mean(fft_img)));            % Avg
    g_med=real(median(median(fft_img)));       % Median
    g_st_dev=real(std(std(double(fft_img))));  % S.D
    g_max_=real(max(max(fft_img)));            % MAX
    g_min_=real(min(min(fft_img)));            % MIN
    g_mode_=real(mode(mode(fft_img)));
    g_midpoint_=real(0.5*(g_max_+g_min_));
    g_var_=real(var(var(fft_img)));

    current=b_img(i).img;
%Fourier transform
    fft_img=fft2(current);
%Statistical measures
    b_av=real(mean(mean(fft_img)));            % Avg
    b_med=real(median(median(fft_img)));       % Median
    b_st_dev=real(std(std(double(fft_img))));  % S.D
    b_max_=real(max(max(fft_img)));            % MAX
    b_min_=real(min(min(fft_img)));            % MIN
    b_mode_=real(mode(mode(fft_img)));
```

```
    b_midpoint_=real(0.5*(b_max_+b_min_));
    b_var_=real(var(var(fft_img)));

    target = 0;

%Column Values

rgb=[gray_av,gray_med,gray_st_dev,gray_max_,gray_min_,gray_mode_,gray_m
idpoint_,gray_var_,...

r_av,r_med,r_st_dev,r_max_,r_min_,r_mode_,r_midpoint_,r_var_,...
        g_av,g_med,g_st_dev,g_max_,g_min_,g_mode_,g_midpoint_,g_var_
,...

b_av,b_med,b_st_dev,b_max_,b_min_,b_mode_,b_midpoint_,b_var_,target];
    writematrix(rgb,'FFT data.csv','WriteMode', 'append');
end
```

CODE TO EXTRACT FEACTURES IN THE FREQUENCY DOMAIN AFTER APPLYING DCT

```
%Discrete cosine transform

%gray, red, green, blue
for i=1:numel(dd)

    current=gray_img(i).img;    %r_img(i).img g_img(i).img b_img(i).img
    %DCT
    dct_img=dct2(current);
    gray_dct=dct_img(1,1);

    current=r_img(i).img;    %r_img(i).img g_img(i).img b_img(i).img
    %DCT
    dct_img=dct2(current);
    r_dct=dct_img(1,1);

    current=g_img(i).img;
    %DCT
    dct_img=dct2(current);
    g_dct=dct_img(1,1);

    target = 0;

    dc = [gray_dct,r_dct,g_dct, target]
    writematrix(dc,'DCT.csv','WriteMode', 'append')
end
```

CODE TO EXTRACT FEACTURES IN THE FREQUENCY DOMAIN AFTER APPLYING WAVELET TRANSFORM

```
%Wavelet
%gray
for i=1:numel(dd)
    current=gray_img(i).img; %r_img(i).img

    %WAVELET transform
```

```
    wave_img=wave(current,'haar',3);

%Statistical measures
    gray_av=real(mean(mean(wave_img)));            % Avg
    gray_med=real(median(median(wave_img)));       % Median
    gray_st_dev=real(std(std(double(wave_img))));  % S.D
    gray_max_=real(max(max(wave_img)));            % MAX
    gray_min_=real(min(min(wave_img)));            % MIN
    gray_mode_=real(mode(mode(wave_img)));
    gray_midpoint_=real(0.5*(gray_max_+gray_min_));
    gray_var_=real(var(var(wave_img)));

    current=r_img(i).img;
%WAVELET transform
    wave_img=wave(current,'haar',3);
%Statistical measures
    r_av=real(mean(mean(wave_img)));            % Avg
    r_med=real(median(median(wave_img)));       % Median
    r_st_dev=real(std(std(double(wave_img))));  % S.D
    r_max_=real(max(max(wave_img)));            % MAX
    r_min_=real(min(min(wave_img)));            % MIN
    r_mode_=real(mode(mode(wave_img)));
    r_midpoint_=real(0.5*(r_max_+r_min_));
    r_var_=real(var(var(wave_img)));

    current=g_img(i).img;
%WAVELET transform
    wave_img=wave(current,'haar',3);
%Statistical measures
    g_av=real(mean(mean(wave_img)));            % Avg
    g_med=real(median(median(wave_img)));       % Median
    g_st_dev=real(std(std(double(wave_img))));  % S.D
    g_max_=real(max(max(wave_img)));            % MAX
    g_min_=real(min(min(wave_img)));            % MIN
    g_mode_=real(mode(mode(wave_img)));
    g_midpoint_=real(0.5*(g_max_+g_min_));
    g_var_=real(var(var(wave_img)));

    current=b_img(i).img;

%WAVELET transform
    wave_img=wave(current,'haar',3);
%Statistical measures
    b_av=real(mean(mean(wave_img)));            % Avg
    b_med=real(median(median(wave_img)));       % Median
    b_st_dev=real(std(std(double(wave_img))));  % S.D
    b_max_=real(max(max(wave_img)));            % MAX
    b_min_=real(min(min(wave_img)));            % MIN
    b_mode_=real(mode(mode(wave_img)));
    b_midpoint_=real(0.5*(b_max_+b_min_));
    b_var_=real(var(var(wave_img)));

    target = 0;

%Column Values
```

```
rgb=[gray_av,gray_med,gray_st_dev,gray_max_,gray_min_,gray_mode_,gray_m
idpoint_,gray_var_,...

r_av,r_med,r_st_dev,r_max_,r_min_,r_mode_,r_midpoint_,r_var_,...
        g_av,g_med,g_st_dev,g_max_,g_min_,g_mode_,g_midpoint_,g_var_
,...

b_av,b_med,b_st_dev,b_max_,b_min_,b_mode_,b_midpoint_,b_var_,target];
    writematrix(rgb,'Wavelet.csv','WriteMode', 'append');
end
```

NORMALIZATION

```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

EXPLANATION ABOUT EACH FEATURES

Spatial Domain Features

Mean Filter:

Linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. The idea is replacing the value of every pixel in an image by the average of the grey levels in the neighborhood define by the filter mask.

Types of Mean filter:

(i) Averaging filter: It is used in reduction of the detail in image. All coefficients are equal.

(ii) Weighted averaging filter: In this, pixels are multiplied by different coefficients. Center pixel is multiplied by a higher value than average filter.

Order Statistics Filter:

It is based on the ordering the pixels contained in the image area encompassed by the filter. It replaces the value of the center pixel with the value determined by the ranking result. Edges are better preserved in this filtering.

Types of Order statistics filter:

(i) Minimum filter: 0th percentile filter is the minimum filter. The value of the center is replaced by the smallest value in the window.

(ii) Maximum filter: 100th percentile filter is the maximum filter. The value of the center is replaced by the largest value in the window.

(iii) Median filter: Each pixel in the image is considered. First neighboring pixels are sorted and original values of the pixel is replaced by the median of the list.

Canny Edge detection is an image processing method used to detect edges in an image while suppressing noise.

Discrete Cosine Transform filter.

The Basic Operation of the Discrete Cosine Transform (DCT) is as follows: • The input image is N by M;

- f(i,j) is the intensity of the pixel in row i and column j; F(u,v) is the DCT coefficient in row k1 and column k2 of the DCT matrix.
- For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT.
- Compression is achieved since the lower right values represent higher frequencies, and are often small – small enough to be neglected with little visible distortion.
- The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level; 8-bit pixels have levels from 0 to 255. FFT filter:
- The algorithm helps in such a way that it allows us to split the input signal that is spread in time (Like in the image above) into the number of frequencies of length, amplitude and phase so that all these frequencies together can reform the original signal.

So, it actually converts the data information of time domain into domain of frequencies and also backwards.

Wavelet transform

Wavelet transforms are mathematical tools for analysing data where features vary over different scales. For signals, features can be frequencies varying over time, transients, or slowly varying trends. For images, features include edges and textures. Wavelet transforms were primarily created to address limitations of the Fourier transform. While Fourier analysis consists of decomposing a signal into sine waves of specific frequencies, wavelet analysis is based on decomposing signals into shifted and scaled versions of a wavelet. A wavelet, unlike a sine wave, is a rapidly decaying, wave-like oscillation. This enables wavelets to represent data across multiple scales. Wavelet transforms can be classified into two broad classes: the continuous wavelet transform (CWT) and the discrete wavelet transform (DWT). The

continuous wavelet transform is a time-frequency transform, which is ideal for analysis of non-stationary signals. A signal being nonstationary means that its frequency-domain representation changes over time. CWT is similar to the short-time Fourier transform (STFT). The STFT uses a fixed window to create a local frequency analysis, while CWT tiles the time-frequency plane with variable-sized windows. The window widens in time, making it suitable for low-frequency phenomena, and narrows for high-frequency phenomena. The continuous wavelet transform can be used to analyse transient behaviour, rapidly changing frequencies, and slowly varying behaviour.

Here, mainly focuses on spatial and frequency filters so, order statistics filters play an important role in classification.

EXTRACTED FEATURES

Spatial Features Extraction using MATLAB

Red	85.61463	92.56375	78.71675	125.03139	105.32459	152.48316	130.21843	155.86741	102.44592	96.0016	113.02614	139.04064	81.76685	110.59567	105.00431
Green	72.79782	73.42577	57.68082	122.65719	92.23384	83.13104	107.24826	109.73327	93.30231	82.81938	101.17298	132.24628	78.08435	91.40353	77.59866
Blue	34.71952	62.13364	34.66886	95.64339	71.63331	42.73182	89.42242	57.51604	62.53725	72.16029	93.77512	113.71901	67.84048	75.36719	59.50002
Gray_Average	0.02222	0.06755	0.02896	0.06969	0.06702	0.01044	0.04451	0.01638	0.04029	0.02754	0.0381	0.03217	0.05079	0.03341	0.04911
Gray_Median	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gray_SD	0.08125	0.05067	0.07581	0.07053	0.07022	0.04895	0.04194	0.04728	0.04707	0.03477	0.0532	0.04677	0.04396	0.05479	0.0405
Gray_Max	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Gray_Min	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gray_Skewness	2.68537	0.78442	0.7853	3.95837	0.38611	0.32585	1.13516	0.48901	1.57169	1.42309	1.51005	2.51182	1.96209	1.83422	1.13311
Gray_Kurtusis	23.18512	4.24939	10.81879	56.19221	10.56278	2.68682	6.53731	3.10268	29.58382	20.47653	10.58054	74.66272	25.80925	20.12368	7.74765
Gray_IQR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Area_Mean	3555	10808	4634	11151	10723	1670	7122	2621	6446	4406	6096	5147	8126	5346	7858
Area_SD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BoundingBox_Mean	174	199.75	197	197.75	199.75	196.75	199.75	194.75	199.75	199.75	199.75	199.75	199.75	199.75	199.75
BoundingBox_SD	200.98881	228.91938	212.06642	226.63351	228.91938	198.94074	228.91938	222.11427	228.91938	217.13379	228.91938	223.78356	228.91938	228.91938	228.91938
BoundingBox_Skewness	0.11863	0	0.00743	0.00062	0	0.00121	0	0.00324	0	0	0	0	0	0	0
BoundingBox_Kurtusis	1.15708	1	1.02828	1.00083	1	1.00175	1	1.00443	1	1.01243	1	1.00216	1	1	1
Eccentricity_Mean	0.8976	0.58865	0.71314	0.59582	0.52944	0.83546	0.55867	0.5511	0.60687	0.50426	0.74599	0.67277	0.39158	0.65755	0.51312
Eccentricity_SD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FilledArea_Mean	3714	14135	4639	11518	10934	2028	7175	2649	6861	4557	6372	5221	9170	5496	8144
FilledArea_SD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Area	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000	160000
Entropy	0.15372	0.35671	0.18916	0.36478	0.3547	0.08368	0.26261	0.12061	0.24361	0.18189	0.23351	0.20515	0.28973	0.21123	0.28261
Above_Red	49152	72740	52593	66955	83126	77892	88523	87854	75954	73062	71914	72852	75097	72494	69395
Above_Green	110848	87260	107407	93045	76874	82108	71477	72146	84046	86938	88086	87148	84903	87506	90605
Above_Blue	52597	74646	74757	74006	86640	73648	79405	70450	84776	90646	71459	73983	74344	71905	69687
Below_Red	107403	85354	85243	85994	73360	86352	80595	89550	75224	69354	88541	86017	85656	88095	90313
Below_Green	55162	64168	40837	69801	76079	62650	75182	55496	66119	77396	73674	75905	73636	73865	50397
Below_Blue	104838	95832	119163	90199	83921	97350	84818	104504	93881	82604	86326	84095	86364	86135	109603
Target	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29 - Dependent Columns
1 - Independent column

Frequency Features Extraction using – DCT

Discrete Wavelet Transform

DCT Gray	16188.2143	17438.7098	13740.5	26942.8348	21009.9286	22234.0714	25106.6161	26334.0134	20724.1563	19159.3482	23269.6607	29604.7232	17475.7277	21349.0179
DCT Red	7774.45089	13916.7232	7763.89286	21423.7009	16045.0848	9571.04911	20029.7679	12878.9241	14007.8438	16163.4732	21005.2545	25472.5268	15194.2813	16879.5402
DCT Green	16306.2679	16446.5357	12920.1027	27474.5446	20659.8036	18620.7991	24023.1027	24580.1607	20898.9464	18551.0714	22662.1027	29621.75	17490.0848	20473.7455
Target	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4 - Dependent Columns
1 - Independent column

Frequency Features Extraction using - FFT

FFT Gray Avg	79	72	69	131	118	67	97	249	93	83	37	172	27	26
FFT Gray Median	281.1260465	717.9096052	625.8612076	659.2518952	214.7162237	284.4339499	-512.73997	-522.895909	-584.028031	255.910446	-131.980669	-336.64408	193.0645778	602.0997325
FFT Gray SD	18732.71291	19368.43374	15594.37997	27448.50708	22521.04987	23592.23912	27367.2333	29935.26749	22583.60756	21061.6677	25289.28413	31121.22552	20364.73189	23029.87482
FFT Gray Max	3626160	3906271	3077872	6035195	4706224	4980432	5623882	5898819	4642211	4291694	5212404	6631458	3914563	4782180
FFT Gray Min	-5.89348687	2.395489303	5.444344216	5.654942021	-7.00000072	-1.95429132	2.51703227	0.096893993	-3.4004528	2.51657664	-1.48076844	5.61900943	-2.59770761	-0.38107378
FFT Gray Mode	-5.89348687	2.395489303	5.444344216	5.654942021	-7.00000072	-1.95429132	2.51703227	0.096893993	-3.4004528	2.51657664	-1.48076844	5.61900943	-2.59770761	-0.38107378
FFT Gray Midpoint	1813077.053	1953136.698	1538938.722	3017600.327	2353108.5	2490215.023	2811942.259	2949409.548	2321103.8	2145848.26	2606201.26	3315731.81	1957280.201	2391089.809
FFT Gray Var	2.10E+19	2.33E+19	9.22E+18	1.20E+20	4.52E+19	5.80E+19	9.87E+19	1.37E+20	4.59E+19	3.46E+19	6.69E+19	1.77E+20	2.22E+19	4.75E+19
FFT Red Avg	3	60	48	120	60	53	97	251	79	90	21	204	20	10
FFT Red Median	106.4239188	462.530018	-665.512122	251.1183975	-182.485378	264.1761489	402.5366223	569.1186579	544.1555852	148.192659	-302.647261	-63.675725	443.1281144	-51.6885692
FFT Red SD	9651.766692	16698.20727	11894.21608	22494.59036	18598.9147	9992.118871	23437.40539	20034.36951	16288.99661	18846.8112	23706.0395	28338.55454	18740.63973	19275.94447
FFT Red Max	1741477	3117346	1739112	4798909	3594099	2143915	4486668	2884879	3137757	3620618	4705177	5705846	3403519	3781017
FFT Red Min	-1.83683612	6	7.522798623	1.154320266	1.376124915	0.646157498	-2.11444913	3.054727059	3.637012779	2.49604262	0.606632818	-3.34853764	-0.35004653	-0.2720651
FFT Red Mode	-1.83683612	6	7.522798623	1.154320266	1.376124915	0.646157498	-2.11444913	3.054727059	3.637012779	2.49604262	0.606632818	-3.34853764	-0.35004653	-0.2720651
FFT Red Midpoint	870737.5816	1558676	869559.7614	2399455.077	1797050.188	1071957.823	2243332.943	1442441.027	1568880.319	1810310.25	2352588.803	2852921.326	1701759.325	1890508.364
FFT Red Var	1.09E+18	9.91E+18	1.43E+18	4.79E+19	1.61E+19	1.92E+18	4.44E+19	1.16E+19	9.73E+18	1.84E+19	4.65E+19	1.04E+20	1.35E+19	2.00E+19
FFT Green Avg	95	72	75	138	131	71	101	249	97	90	37	176	26	25
FFT Green Median	559.6108654	-187.474572	-155.442834	701.7845121	266.6331913	-215.63332	616.6876789	87.83200007	90.51269768	314.866815	144.9455106	357.9962686	43.24115528	-568.248253
FFT Green SD	19164.52443	19074.18359	14969.42268	27978.76911	22534.40511	19213.73178	26490.41026	29678.33587	23633.84343	20758.1405	24900.02792	31423.75687	20686.65455	22361.33047
FFT Green Max	3652604	3684024	2894103	6154298	4627796	4171059	5381175	5505956	4681364	4155440	5076311	6635272	3917779	4586119
FFT Green Min	3.256145431	-1.83964522	6.523363665	-6.20868661	3.159027791	-2.35569367	-3.12305963	4.888040422	-3.99277406	0.60898254	-1.83423794	-3.80735867	-0.7648462	0.444431006
FFT Green Mode	3.256145431	-1.83964522	6.523363665	-6.20868661	3.159027791	-2.35569367	-3.12305963	4.888040422	-3.99277406	0.60898254	-1.83423794	-3.80735867	-0.7648462	0.444431006
FFT Green Midpoint	1826303.628	1842011.08	1447054.762	3077145.896	2313899.58	2085528.322	2690585.938	2752980.444	2340680.004	2077720.3	2538154.583	3317634.096	1958889.118	2293059.722
FFT Green Var	2.29E+19	1.86E+19	6.75E+18	1.29E+20	4.26E+19	2.76E+19	8.24E+19	1.16E+20	4.67E+19	2.98E+19	6.07E+19	1.81E+20	2.24E+19	4.05E+19
FFT Blue Avg	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FFT Blue Median	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FFT Blue SD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FFT Blue Max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
FFT Blue Min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

29 - Dependent Columns
1 - Independent column

Frequency Features Extraction using - Wavelet

Wavelet Gray Avg	2.267256903	2.442396334	1.924439776	3.773506278	2.942567027	3.114015606	3.516332783	3.68823717	2.902542892	2.6833821	3.259056122	4.146319778	2.447580907	2.990058523
Wavelet Gray Median	1.902696078	2.357598039	1.622181373	3.603553922	3.243627451	3.617156863	3.507352941	3.073039216	2.841911765	2.89080882	3.039828431	3.753431373	2.400245098	2.97879902
Wavelet Gray SD	0.643714544	0.213205732	0.4231756	0.266397344	0.257144598	0.307016993	0.264182257	0.518020578	0.242725795	0.41478292	0.306794714	0.308843503	0.18224141	0.31380241
Wavelet Gray Max	6.924019608	6.916176471	6.915686275	7.676960784	5.920588235	5.266666667	7.145098039	7.920098039	6.792647059	7.12401961	6.901960784	7.960294118	5.521078431	6.55
Wavelet Gray Min	0.218137255	0.10245098	0.116176471	1.568627451	0.321568627	1.199509804	0.11127451	0.265686275	0.555882353	0.17205882	0.376470588	1.24754902	0.284803922	0.235784314
Wavelet Gray Mode	0.218137255	0.432843137	0.116176471	1.568627451	0.384313725	1.199509804	0.11127451	0.265686275	0.57254902	0.21176471	0.376470588	1.24754902	0.284803922	0.235784314
Wavelet Gray Midpoint	3.571078431	3.509313725	3.515931373	4.622794118	3.121078431	3.233088235	3.628186275	4.092892157	3.674264706	3.64803922	3.639215686	4.603921569	2.902941176	3.392892157
Wavelet Gray Var	2.980400439	0.231899834	0.816842845	0.260218661	0.344304159	0.356488669	0.649353995	4.169483732	0.323633871	1.40719035	0.588580282	0.717503479	0.284670054	0.571674257
Wavelet Red Avg	1.088858668	1.949120898	1.087379952	3.000518332	2.247210759	1.340483068	2.805289616	1.803770883	1.961882878	2.26379177	2.94191239	3.567580782	2.128050595	2.364081258
Wavelet Red Median	0.925980392	1.665441176	0.731617647	2.798897059	2.194852941	1.258455882	2.232843137	0.193627451	1.596568627	2.35367647	2.469117647	3.376593137	1.838848039	2.19877451
Wavelet Red SD	0.408114164	0.241180069	0.688770293	0.267595351	0.348436724	0.145566985	0.322209864	0.716253428	0.293901411	0.49669647	0.348312186	0.355258363	0.203279918	0.295213167
Wavelet Red Max	4.783333333	7.068137255	7.261764706	7.364215686	6.085784314	3.372058824	7.384313725	7.91127451	6.650980392	7.86813726	6.741176471	7.950980392	5.658333333	6.066666667
Wavelet Red Min	0.058823529	0.1	0.078921569	0.89754902	0.125980392	0.435294118	0.062254902	0.017156863	0.452941176	0.05784314	0.330882353	0.532352941	0.068137255	0.078431373
Wavelet Red Mode	0.058823529	0.1	0.078921569	0.89754902	0.125980392	0.435294118	0.062254902	0.071078431	0.521568627	0.05784314	0.904411765	0.532352941	0.068137255	0.469117647
Wavelet Red Midpoint	2.421078431	3.584068627	3.670343137	4.130882353	3.105882353	1.903676471	3.723284314	3.964215686	3.551960784	3.9629902	3.536029412	4.241666667	2.863235294	3.07254902
Wavelet Red Var	0.490267235	0.302444897	2.477174249	0.418483457	0.862863357	0.014695834	0.872421716	4.632799873	0.492477476	2.24379949	0.816361067	1.794371266	0.364270827	0.47220868
Wavelet Green Avg	2.283791016	2.303436375	1.80953819	3.847975444	2.893529912	2.607955057	3.364580207	3.44259954	2.927023309	2.59818928	3.173963961	4.148704482	2.449591712	2.867471364
Wavelet Green Median	1.881127451	2.116666667	1.74252451	3.687622549	3.133823529	2.680392157	3.074632353	2.327083333	2.930392157	2.56703431	2.780637255	3.884436275	2.221078431	2.798161765
Wavelet Green SD	0.694381341	0.195433528	0.393017562	0.24095358	0.335992727	0.170564778	0.304229631	0.584139531	0.308053569	0.34966552	0.318117842	0.385886322	0.188572702	0.332300593
Wavelet Green Max	7.459803922	6.925980392	6.885784314	7.804411765	5.818137255	4.630392157	7.168627451	7.969117647	6.997058824	6.97401961	6.952941176	7.971078431	5.639215686	6.667156863
Wavelet Green Min	0.211764706	0.117647059	0.087745098	1.417647059	0.301470588	1.065196078	0.123529412	0.037254902	0.478921569	0.17401961	0.358823529	1.103431373	0.258333333	0.234313725
Wavelet Green Mode	1.117156863	0.141176471	0.087745098	1.417647059	0.301470588	1.065196078	0.123529412	0.037254902	0.57254902	0.17401961	0.358823529	1.103431373	0.258333333	0.234313725
Wavelet Green Midpoint	3.835784314	3.521813725	3.486764706	4.611029412	3.059803922	2.847794118	3.646078431	4.003186275	3.737990196	3.57401961	3.655882353	4.537254902	2.94877451	3.450735294
Wavelet Green Var	4.223134361	0.2047475	0.711796514	0.206471402	0.690903434	0.043060613	0.764648187	5.657099709	0.625165337	1.06436021	0.643568986	1.359027239	0.335379099	0.665123381
Wavelet Blue Avg	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wavelet Blue Median	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wavelet Blue SD	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wavelet Blue Max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wavelet Blue Min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Target	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29 - Dependent Columns
1 - Independent column

Finally, together 532*91, rows and columns were collected and pre-processed before the model building phase. Missing data, Outliers and Datatypes were aliened and refined in the Exploratory Data Analysis phase.

Important features were technically selected and reduced from 91 columns to appropriate group to find the best features. Above mentioned process were handled in Python programming language, since it was an open source and rich in Machine learning libraries.

CLASSIFICATION AND RESULT ANALYSIS

Importing Required libraries

In [1]:
```
import pandas as pd
import numpy as np
import time
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
```

Importing Required Dataset

In [2]:
```
spatial=pd.read_csv("Spatial_data.csv").drop_duplicates()
frequency=pd.read_csv("Frequency_data.csv").drop_duplicates()
DataFrame=pd.merge(frequency,spatial,left_index=True,right_index=True)
```

In [3]:
```
#Check the shape
print(DataFrame.shape)
```

```
(532, 91)
```

In [4]:
```
DataFrame.head(25)
```

Out[4]:

	FFT Gray Avg	FFT Gray Median	FFT Gray SD	FFT Gray Max	FFT Gray Min	FFT Gray Mode	FFT Gray Midpoint	FFT Gray Var	FFT Red Avg	FFT Red Median	...	FilledArea_SD	Area	Entropy
0	79.0	281.13	18732.71	3626160.0	-5.89	-5.89	1813077.05	21000000000000000000.00	3.0	106.42	...	0.0	160000.0	0.15
1	72.0	717.91	19368.43	3906271.0	2.40	2.40	1953136.70	23300000000000000000.00	60.0	462.53	...	0.0	160000.0	0.36
2	69.0	625.86	15594.38	3077872.0	5.44	5.44	1538938.72	9220000000000000000.00	48.0	-665.51	...	0.0	160000.0	0.19
3	131.0	659.25	27448.51	6035195.0	5.65	5.65	3017600.33	120000000000000000000.00	120.0	251.12	...	0.0	160000.0	0.36
4	118.0	214.72	22521.05	4706224.0	-7.00	-7.00	2353108.50	45200000000000000000.00	60.0	-182.49	...	0.0	160000.0	0.35
5	67.0	284.43	23592.24	4980432.0	-1.95	-1.95	2490215.02	58000000000000000000.00	53.0	264.18	...	0.0	160000.0	0.08
6	97.0	-512.74	27367.23	5623882.0	2.52	2.52	2811942.26	98700000000000000000.00	97.0	402.54	...	0.0	160000.0	0.26
7	249.0	-522.90	29935.27	5898819.0	0.10	0.10	2949409.55	137000000000000000000.00	251.0	569.12	...	0.0	160000.0	0.12
8	93.0	-584.03	22583.61	4642211.0	-3.40	-3.40	2321103.80	45900000000000000000.00	79.0	544.16	...	0.0	160000.0	0.24
9	83.0	255.91	21061.67	4291694.0	2.52	2.52	2145848.26	34600000000000000000.00	90.0	148.19	...	0.0	160000.0	0.18
10	37.0	-131.98	25289.28	5212404.0	-1.48	-1.48	2606201.26	66900000000000000000.00	21.0	-302.65	...	0.0	160000.0	0.23
11	172.0	-336.64	31121.23	6631458.0	5.62	5.62	3315731.81	177000000000000000000.00	204.0	-63.68	...	0.0	160000.0	0.21
12	27.0	193.06	20364.73	3914563.0	-2.60	-2.60	1957280.20	22200000000000000000.00	20.0	443.13	...	0.0	160000.0	0.29
13	26.0	602.10	23029.87	4782180.0	-0.38	-0.38	2391089.81	47500000000000000000.00	10.0	-51.69	...	0.0	160000.0	0.21
14	32.0	-292.82	21785.27	4201323.0	-0.20	-0.20	2100661.40	31900000000000000000.00	25.0	-249.64	...	0.0	160000.0	0.28
15	118.0	-785.86	16489.70	3466186.0	7.22	7.22	1733096.61	13900000000000000000.00	91.0	330.27	...	0.0	160000.0	0.39
16	112.0	-60.30	18768.88	3863265.0	2.19	2.19	1931633.60	21700000000000000000.00	114.0	325.48	...	0.0	160000.0	0.11
17	117.0	120.04	27275.41	5664014.0	-4.04	-4.04	2832004.98	96000000000000000000.00	73.0	294.84	...	0.0	160000.0	0.24
18	51.0	-224.28	9916.76	2068978.0	2.21	2.21	1034490.11	18500000000000000000.00	51.0	-75.78	...	0.0	160000.0	0.21
19	149.0	3.29	20690.84	4234871.0	0.25	0.25	2117435.62	35000000000000000000.00	140.0	57.88	...	0.0	160000.0	0.12
20	105.0	-156.58	21365.04	4171184.0	0.29	0.29	2085592.14	37300000000000000000.00	50.0	852.37	...	0.0	160000.0	0.41
21	80.0	-135.00	26630.25	5283815.0	-0.20	-0.20	2641907.40	80800000000000000000.00	4.0	269.79	...	0.0	160000.0	0.31
22	157.0	-1029.75	24431.18	5149970.0	4.35	4.35	2574987.17	66200000000000000000.00	152.0	255.52	...	0.0	160000.0	0.28
23	129.0	-349.07	22606.96	4399923.0	-3.56	-3.56	2199959.72	38300000000000000000.00	115.0	-281.63	...	0.0	160000.0	0.17
24	141.0	-319.09	27245.78	5824453.0	-3.22	-3.22	2912224.89	105000000000000000000.00	83.0	743.41	...	0.0	160000.0	0.29

25 rows × 91 columns

In [5]:
```
DataFrame.keys()
```

Out[5]:
```
Index(['FFT Gray Avg', 'FFT Gray Median', 'FFT Gray SD', 'FFT Gray Max',
       'FFT Gray Min', 'FFT Gray Mode', 'FFT Gray Midpoint', 'FFT Gray Var',
       'FFT Red Avg', 'FFT Red Median', 'FFT Red SD', 'FFT Red Max',
       'FFT Red Min', 'FFT Red Mode', 'FFT Red Midpoint', 'FFT Red Var',
       'FFT Green Avg', 'FFT Green Median', 'FFT Green SD', 'FFT Green Max',
       'FFT Green Min', 'FFT Green Mode', 'FFT Green Midpoint',
       'FFT Green Var', 'FFT Blue Avg', 'FFT Blue Median', 'FFT Blue SD',
       'FFT Blue Max', 'FFT Blue Min', 'DCT Gray', 'DCT Red', 'DCT Green',
       'Wavelet Gray Avg', 'Wavelet Gray Median', 'Wavelet Gray SD',
       'Wavelet Gray Max', 'Wavelet Gray Min', 'Wavelet Gray Mode',
       'Wavelet Gray Midpoint', 'Wavelet Gray Var', 'Wavelet Red Avg',
       'Wavelet Red Median', 'Wavelet Red SD', 'Wavelet Red Max',
       'Wavelet Red Min', 'Wavelet Red Mode', 'Wavelet Red Midpoint',
       'Wavelet Red Var', 'Wavelet Green Avg', 'Wavelet Green Median',
       'Wavelet Green SD', 'Wavelet Green Max', 'Wavelet Green Min',
       'Wavelet Green Mode', 'Wavelet Green Midpoint', 'Wavelet Green Var',
       'Wavelet Blue Avg', 'Wavelet Blue Median', 'Wavelet Blue SD',
       'Wavelet Blue Max', 'Wavelet Blue Min', 'Red', 'Green', 'Blue',
       'Gray_Average', 'Gray_Median', 'Gray_SD', 'Gray_Max', 'Gray_Min',
       'Gray_Skewness', 'Gray_Kurtusis', 'Gray_IQR', 'Area_Mean', 'Area_SD',
       'BoundingBox_Mean', 'BoundingBox_SD', 'BoundingBox_Skewness',
       'BoundingBox_Kurtusis', 'Eccentricity_Mean', 'Eccentricity_SD',
       'FilledArea_Mean', 'FilledArea_SD', 'Area', 'Entropy', 'Above_Red',
       'Above_Green', 'Above_Blue', 'Below_Red', 'Below_Green', 'Below_Blue',
       'Target'],
      dtype='object')
```

Unique values throughout the dataset

In [6]:
```
DataFrame=DataFrame.drop(['FFT Gray Var','FFT Red Var','FFT Green Var','FFT Red Var',
                          'Wavelet Green Var','Wavelet Gray Var','Wavelet Red Var'],axis=1)
```

Remove Nan

In [7]:
```
DataFrame=DataFrame.drop(['Wavelet Blue Avg','Wavelet Blue Max','Wavelet Blue Min',
                          'Wavelet Blue Median','Wavelet Blue SD','FFT Blue Max', 'FFT Blue Min',
                          'FFT Blue SD','FFT Blue Avg', 'FFT Blue Median'],axis=1)
```

In [8]:
```
df = DataFrame.sample(frac = 1)
```

In [9]:
```
df.tail(15)
```

Out[9]:

	FFT Gray Avg	FFT Gray Median	FFT Gray SD	FFT Gray Max	FFT Gray Min	FFT Gray Mode	FFT Gray Midpoint	FFT Red Avg	FFT Red Median	FFT Red SD	...	FilledArea_SD	Area	Entropy	Above_Re
358	168.0	-208.63	27258.22	5259792.0	5.47	5.47	2629898.74	124.0	353.64	22121.32	...	0.0	160000.0	0.16	85308
356	162.0	230.50	30938.26	6414037.0	14.44	14.44	3207025.72	84.0	-848.42	20926.64	...	0.0	160000.0	0.26	88556
400	133.0	-200.22	27235.26	5916350.0	-1.49	10.00	2958174.26	78.0	-156.44	18771.06	...	0.0	160000.0	0.32	72904
155	126.0	25.30	74258.94	38495084.0	5.70	5.70	19247544.85	112.0	2597.98	63102.67	...	0.0	160000.0	0.26	80859
345	121.0	200.64	30675.70	6726728.0	2.96	2.96	3363365.48	71.0	-1269.29	19533.29	...	0.0	160000.0	0.71	97019
112	30.0	403.36	23036.65	4598251.0	-3.00	-3.00	2299124.00	0.0	-406.67	10175.64	...	0.0	160000.0	0.23	65070
126	26.0	-1095.59	67032.65	30039173.0	-0.96	-0.96	15019586.02	15.0	-527.19	63912.72	...	0.0	160000.0	0.21	67413
496	107.0	649.01	23590.08	5182652.0	0.33	0.33	2591326.17	68.0	-1009.97	15504.37	...	0.0	160000.0	0.30	71658
223	157.0	826.85	146964.62	115672263.0	1.70	1.70	57836132.35	21.0	-580.76	124375.23	...	0.0	160000.0	0.24	101154
35	164.0	-499.31	28975.07	6191576.0	-3.62	-3.62	3095786.19	160.0	396.72	21271.67	...	0.0	160000.0	0.20	79488
104	98.0	-640.11	19922.31	4152853.0	8.24	8.24	2076430.62	52.0	280.93	11742.76	...	0.0	160000.0	0.26	60752
393	77.0	104.34	18602.13	3902515.0	-0.07	-0.07	1951257.46	45.0	-259.69	15456.76	...	0.0	160000.0	0.23	86223
279	61.0	449.37	57331.35	21763962.0	0.51	0.51	10881981.25	34.0	-494.24	52050.39	...	0.0	160000.0	0.44	53021
406	54.0	-474.83	25479.48	5548606.0	-1.81	-1.81	2774302.10	22.0	864.61	19225.62	...	0.0	160000.0	0.31	75862
163	72.0	-25.96	131537.96	125350865.0	-0.57	-0.57	62675432.22	47.0	105.34	120956.98	...	0.0	160000.0	0.25	61136

15 rows × 75 columns

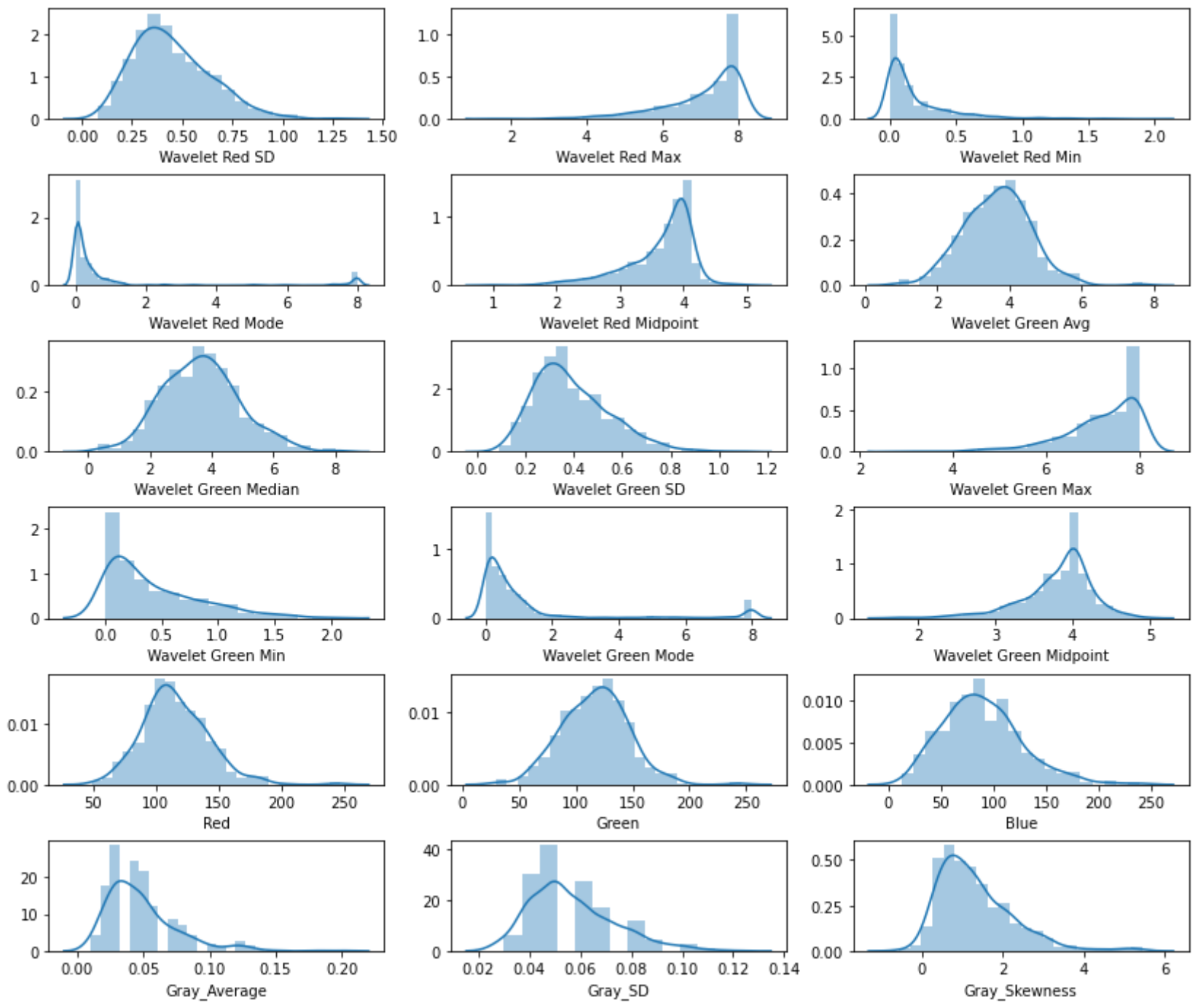
In [10]:
```
df.to_csv('Shuffled_df.csv', header=True, index=False)
```

EXPLORATORY DATA ANALYSIS

In [11]:
```
numerical_features = []
categorical_features = []

for i in df.columns:
    if df[i].nunique()>7:
        numerical_features.append(i)
    else:
        categorical_features.append(i)
fig, axes = plt.subplots(nrows=17, ncols=3, figsize=(14,35))
fig.subplots_adjust(hspace=0.5)

for ax, feats in zip(axes.flatten(), numerical_features):
    sns.distplot(a=df[feats], ax=ax)
```



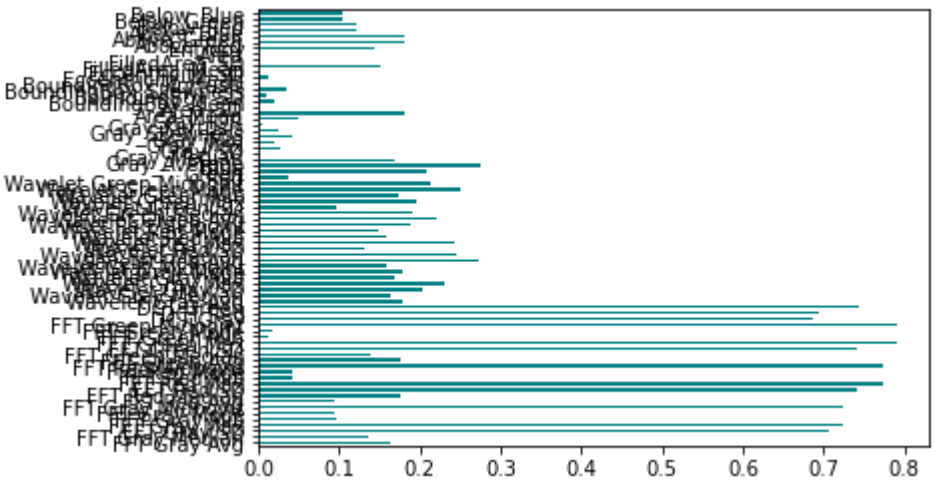
FEATURE SELECTION

Filter Method

In [12]:
```
from sklearn.feature_selection import mutual_info_classif

# generate dataset
X = df.iloc[:,:-1].values
Y = df.iloc[:,-1].values
```

In [13]:
```
importanes = mutual_info_classif(X,Y)
feat_importanes = pd.Series(importanes, df.columns[0:len(df.columns)-1])
feat_importanes.plot(kind='barh', color='teal')
plt.show()
```



In [14]:
```
filter_df = pd.DataFrame(feat_importanes)
filter_df.columns = ['Values']
filter_df = filter_df.sort_values(by = 'Values', ascending = False)
cm = sns.light_palette("black", as_cmap=True)
filter_df.style.background_gradient(cmap=cm)
```

Out[14]:

	Values
FFT Green Midpoint	0.791423
FFT Green Max	0.791423
FFT Red Max	0.773070
FFT Red Midpoint	0.773070
DCT Green	0.743557
FFT Red SD	0.740922
FFT Green SD	0.740416
FFT Gray Max	0.724994
FFT Gray Midpoint	0.724994
FFT Gray SD	0.707771
DCT Red	0.694280
DCT Gray	0.686313
Blue	0.274708
Wavelet Red Avg	0.271913
Wavelet Green Mode	0.251248
Wavelet Red Median	0.245469
Wavelet Red Max	0.241688
Wavelet Gray Max	0.229340
Wavelet Green Avg	0.220754
Wavelet Green Midpoint	0.213751
Green	0.206966
Wavelet Gray SD	0.202157
Wavelet Green Max	0.196136
Wavelet Green Median	0.190341
Wavelet Red Midpoint	0.188129
Above_Red	0.181566
Above_Green	0.180958
Area_Mean	0.180952
Wavelet Gray Avg	0.177979
Wavelet Gray Mode	0.177083
FFT Red Median	0.176312
FFT Green Avg	0.176198
Wavelet Green Min	0.173872
Gray_Average	0.169141
Wavelet Gray Min	0.168065
Wavelet Gray Median	0.163085
FFT Gray Avg	0.162383
Wavelet Gray Midpoint	0.157856
Wavelet Red Min	0.157211
FilledArea_Mean	0.151300
Wavelet Red Mode	0.148053
Entropy	0.144413
FFT Green Median	0.137605
FFT Gray Median	0.135413
Wavelet Red SD	0.130401
Below_Red	0.121605
Above_Blue	0.121605
Below_Blue	0.104972
Below_Green	0.104107
Wavelet Green SD	0.097196

	Values
FFT Gray Min	0.095294
FFT Red Avg	0.095018
FFT Gray Mode	0.094334
Gray_IQR	0.048078
FFT Red Mode	0.040682
FFT Red Min	0.040605
Gray_Min	0.040510
Red	0.036258
BoundingBox_Skewness	0.034021
Gray_SD	0.026896
Gray_Skewness	0.024022
BoundingBox_Mean	0.018782
Gray_Max	0.018329
FFT Green Mode	0.015710
Eccentricity_Mean	0.013095
FFT Green Min	0.012310
BoundingBox_SD	0.009356
Gray_Kurtusis	0.004262
FilledArea_SD	0.000848
Area	0.000251
Area_SD	0.000000
Eccentricity_SD	0.000000
Gray_Median	0.000000
BoundingBox_Kurtusis	0.000000

In [15]:
```
filter_df.describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
Values	74.0	0.220173	0.243309	0.0	0.040624	0.160119	0.219003	0.791423

In [16]:
```
filter_df = filter_df[filter_df['Values'] > 0.5]
filter_df = filter_df.sort_values(by = 'Values', ascending = False)
cm = sns.light_palette("red", as_cmap=True)
filter_df.style.background_gradient(cmap=cm)
```

Out[16]:

	Values
FFT Green Midpoint	0.791423
FFT Green Max	0.791423
FFT Red Max	0.773070
FFT Red Midpoint	0.773070
DCT Green	0.743557
FFT Red SD	0.740922
FFT Green SD	0.740416
FFT Gray Max	0.724994
FFT Gray Midpoint	0.724994
FFT Gray SD	0.707771
DCT Red	0.694280
DCT Gray	0.686313

In [17]:
```
sel = filter_df.index.values
len(sel)
```

Out[17]: 12

In [18]:
```
selected_feature1=[]
for i in sel:
    selected_feature1.append(i)

selected_feature1
```

Out[18]:
```
['FFT Green Midpoint',
 'FFT Green Max',
 'FFT Red Max',
 'FFT Red Midpoint',
 'DCT Green',
 'FFT Red SD',
 'FFT Green SD',
 'FFT Gray Max',
 'FFT Gray Midpoint',
 'FFT Gray SD',
 'DCT Red',
 'DCT Gray']
```

Sequential Forward Selection

In [19]:
```
X = df.drop(['Target'],axis=1)
y = df['Target']

# importing the necessary libraries
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LinearRegression

# Sequential Forward Selection(sfs)
sfs = SFS(LinearRegression(),k_features=10,forward=True,floating=False,scoring = 'r2',cv = 0)
sfs.fit(X, y)
selected_features = sfs.k_feature_names_ # to get the final set of features
selected_feature2=[]
for i in selected_features:
    selected_feature2.append(i)

selected_feature2
```

Out[19]:
```
['FFT Red Avg',
 'Wavelet Gray Avg',
 'Wavelet Gray Mode',
 'Wavelet Green Avg',
 'Wavelet Green Min',
 'Gray_Kurtusis',
 'BoundingBox_SD',
 'Entropy',
 'Above_Red',
 'Above_Blue']
```

MODEL BUILDING

Importing required libraries form sklearn

In [20]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [21]:
```
X = df[selected_feature1]
y = df['Target']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 4658)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(372, 12)
(160, 12)
(372,)
(160,)
```

In [22]:
```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Decision Tree

In [23]:
```
start_dt1 = time.time()
clf1 = DecisionTreeClassifier()

dt_model1 = clf1.fit(X_train, y_train)
end_dt1 = time.time()
final_dt1 = end_dt1 - start_dt1
final_dt1 = round(final_dt1,3)

y_pred_dt1 = dt_model1.predict(X_test)
accuracy_dt1=accuracy_score(y_test, y_pred_dt1)
print("Accuracy of Decision Tree is :", accuracy_dt1)
print("Computation time {} - Sec".format(final_dt1))
```

```
Accuracy of Decision Tree is : 0.84375
Computation time 0.003 - Sec
```

Random Forest

In [24]:
```
start_rf1 = time.time()

rf1=RandomForestClassifier()
rf_model1 = rf1.fit(X_train, y_train)
end_rf1 = time.time()
final_rf1 = end_rf1 - start_rf1
final_rf1 = round(final_rf1,3)

y_pred_rf1 = rf_model1.predict(X_test)
accuracy_rf1=accuracy_score(y_test, y_pred_rf1)
print("Accuracy of Random Forests model is :", accuracy_rf1)
print("Computation time {} - Sec".format(final_rf1))
```

```
Accuracy of Random Forests model is : 0.83125
Computation time 0.167 - Sec
```

Gaussian Naive Bayes

In [25]:
```
start_nb1 = time.time()
gnb1 = GaussianNB()
gnb1.fit(X_train, y_train)
nb_model = gnb1.fit(X_train, y_train)
end_nb1 = time.time()
final_nb1 = end_nb1 - start_nb1
final_nb1 = round(final_nb1,3)
final_nb1

y_pred_nb1 = gnb1.predict(X_test)

#display_confusion_matrix(y_test, y_pred_nb)
accuracy_nb1=accuracy_score(y_test, y_pred_nb1)
print("Gaussian Naive Bayes model accuracy :", accuracy_nb1)
print("Computation time {} - Sec".format(final_nb1))
```

```
Gaussian Naive Bayes model accuracy : 0.7625
Computation time 0.003 - Sec
```

Logistic Regression

In [26]:
```
start_lr1 = time.time()
lr1 = LogisticRegression()
lr1.fit(X_train, y_train)
end_lr1 = time.time()
final_lr1 = end_lr1 - start_lr1
final_lr1 = round(final_lr1,3)

y_pred_lr1 = lr1.predict(X_test)
accuracy_lr1=accuracy_score(y_test, y_pred_lr1)
print("Accuracy of Logistic Regression is :", accuracy_lr1)
print("Computation time {} - Sec".format(final_lr1))
```

```
Accuracy of Logistic Regression is : 0.8
Computation time 0.025 - Sec
```

Support Vector Classifer

In [27]:
```
start_svm1 = time.time()
svm1 = SVC()

svm1.fit(X_train, y_train)

end_svm1 = time.time()
final_svm1 = end_svm1 - start_svm1
final_svm1 = round(final_svm1,3)

y_pred_svm1 = svm1.predict(X_test)
accuracy_svm1=accuracy_score(y_test, y_pred_svm1)
print("Accuracy of Support Vector Machine is :", accuracy_svm1)
print("Computation time {} - Sec".format(final_svm1))
```

```
Accuracy of Support Vector Machine is : 0.7375
Computation time 0.007 - Sec
```

Artifical Neural Network

In [28]:
```
start_mlp1 = time.time()
mlp1 = MLPClassifier()
mlp1.fit(X_train, y_train.values.ravel())

end_mlp1 = time.time()
final_mlp1 = end_mlp1 - start_mlp1
final_mlp1 = round(final_mlp1,3)

y_pred_mlp1= mlp1.predict(X_test)
accuracy_mlp1=accuracy_score(y_test, y_pred_mlp1)
print("Accuracy of Artificial neural network  is :", accuracy_mlp1)
print("Computation time {} - Sec".format(final_mlp1))
```

```
Accuracy of Artificial neural network  is : 0.775
Computation time 0.325 - Sec

/opt/anaconda3/lib/python3.8/site-packages/sklearn/neural_network/_multilayer_perceptron.py:582: ConvergenceW
arning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

KNN

In [29]:
```
start_knn1 = time.time()
model11 = KNeighborsClassifier(n_jobs=-1)
model11.fit(X_train, y_train)
end_knn1 = time.time()
final_knn1 = end_knn1 - start_knn1
final_knn1 = round(final_knn1,3)

y_pred_knn1 = model11.predict(X_test)
accuracy_knn1=accuracy_score(y_test, y_pred_knn1)
print("\nAccuracy of k Nearest Neighbors is :", accuracy_knn1)
print("Computation time {} - Sec".format(final_knn1))
```

```
Accuracy of k Nearest Neighbors is : 0.80625
Computation time 0.002 - Sec
```

Conclusion

In [30]:
```
models = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest', 'Guassian Naive Bayes','Logistic Regression',
              'Support Vector Machines', 'Artificial neural network','K - Nearest Neighbors'],
    'Score': [accuracy_dt1,accuracy_rf1,accuracy_nb1,accuracy_lr1,
              accuracy_svm1,accuracy_mlp1,accuracy_knn1]})
models.sort_values(by='Score', ascending=False)
```

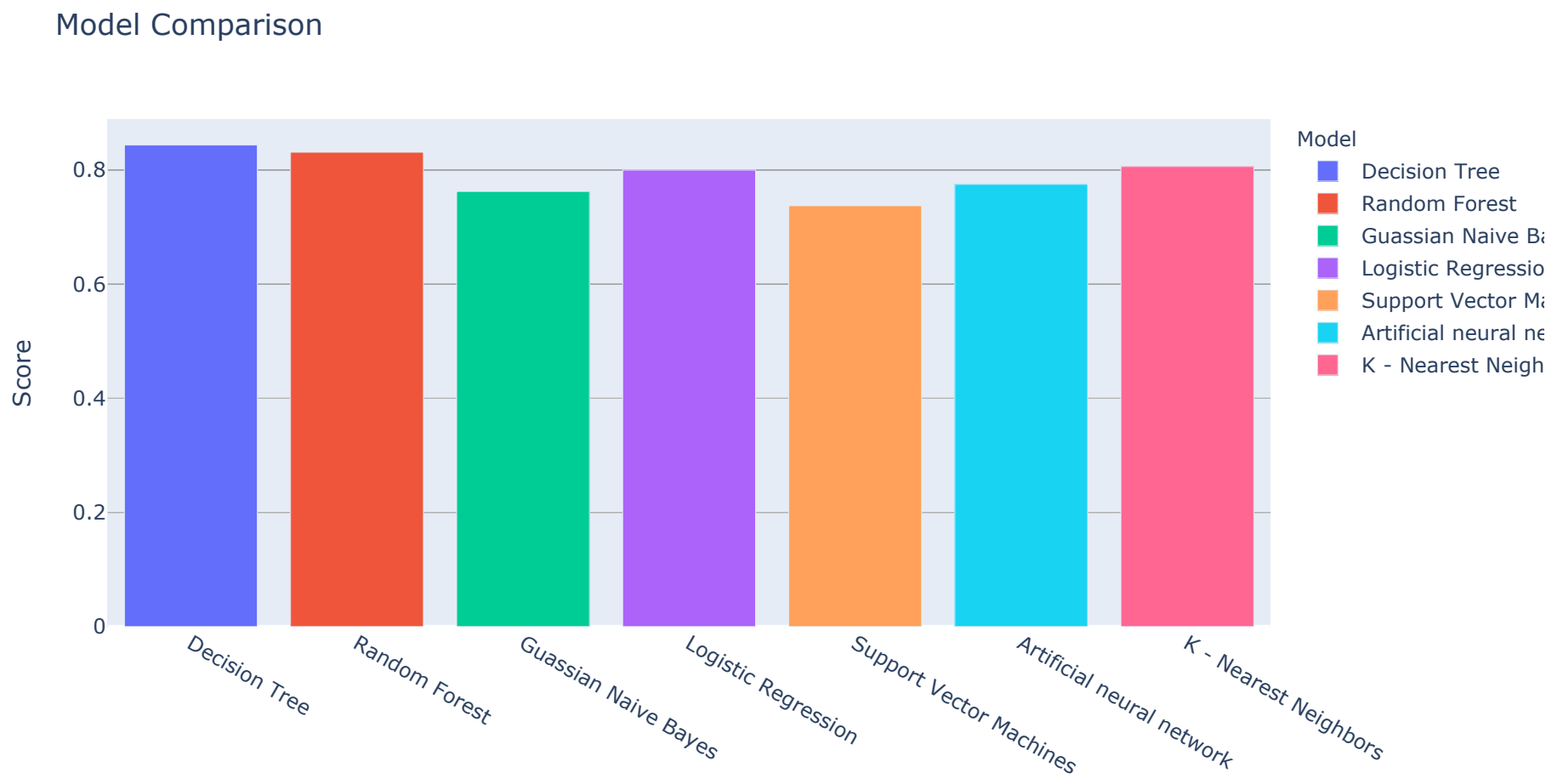
Out[30]:

	Model	Score
0	Decision Tree	0.84375
1	Random Forest	0.83125
6	K - Nearest Neighbors	0.80625
3	Logistic Regression	0.80000
5	Artificial neural network	0.77500
2	Guassian Naive Bayes	0.76250
4	Support Vector Machines	0.73750

In [31]:
```
models.to_csv('Result2.csv', header=True, index=False)
```

In [32]:
```
import plotly.express as px
import plotly.graph_objects as go

fig = px.bar(models, x='Model', y='Score', color="Model", title="Model Comparison")
fig.show()
```



REFERENCES

[1] https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

[2] https://blogs.mathworks.com/community/

[3] Steven Van Vaerenbergh (2021). Sklearn matlab (https://github.com/steven2358/sklearn-matlab), GitHub. Retrieved May 8, 2021.

[4] https://www.dataquest.io/blog/sci-kit-learn-tutorial/

[5] https://scikit-learn.org/stable/