|  |  |
| --- | --- |
| 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\_midpoint\_,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\_midpoint\_,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:**

1. Averaging filter: It is used in reduction of the detail in image. All coefficients are equal.
2. 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:**

1. Minimum filter: 0th percentile filter is the minimum filter. The value of the center is replaced by the smallest value in the window.
2. Maximum filter: 100th percentile filter is the maximum filter. The value of the center is replaced by the largest value in the window.
3. 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.

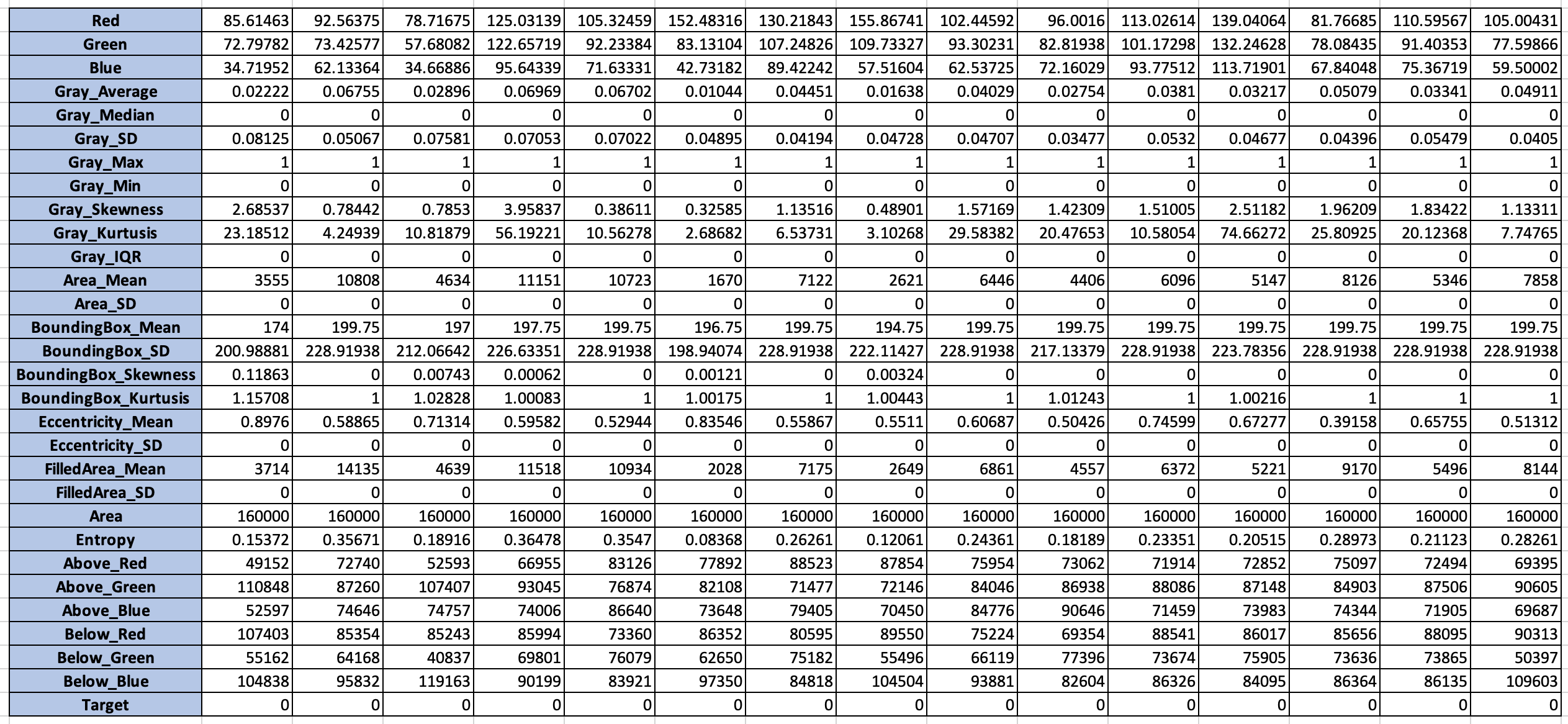
**Wavelet transform**

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**

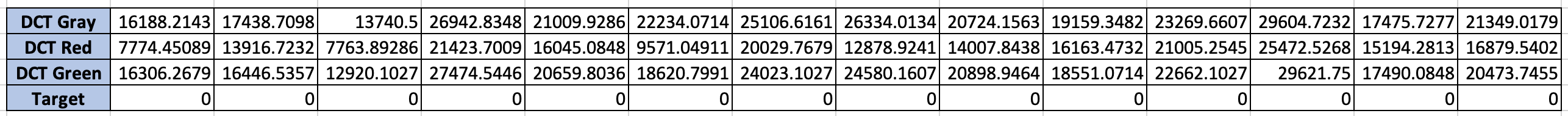


29 - Dependent Columns

1 - Independent column

**Frequency Features Extraction using – DCT**

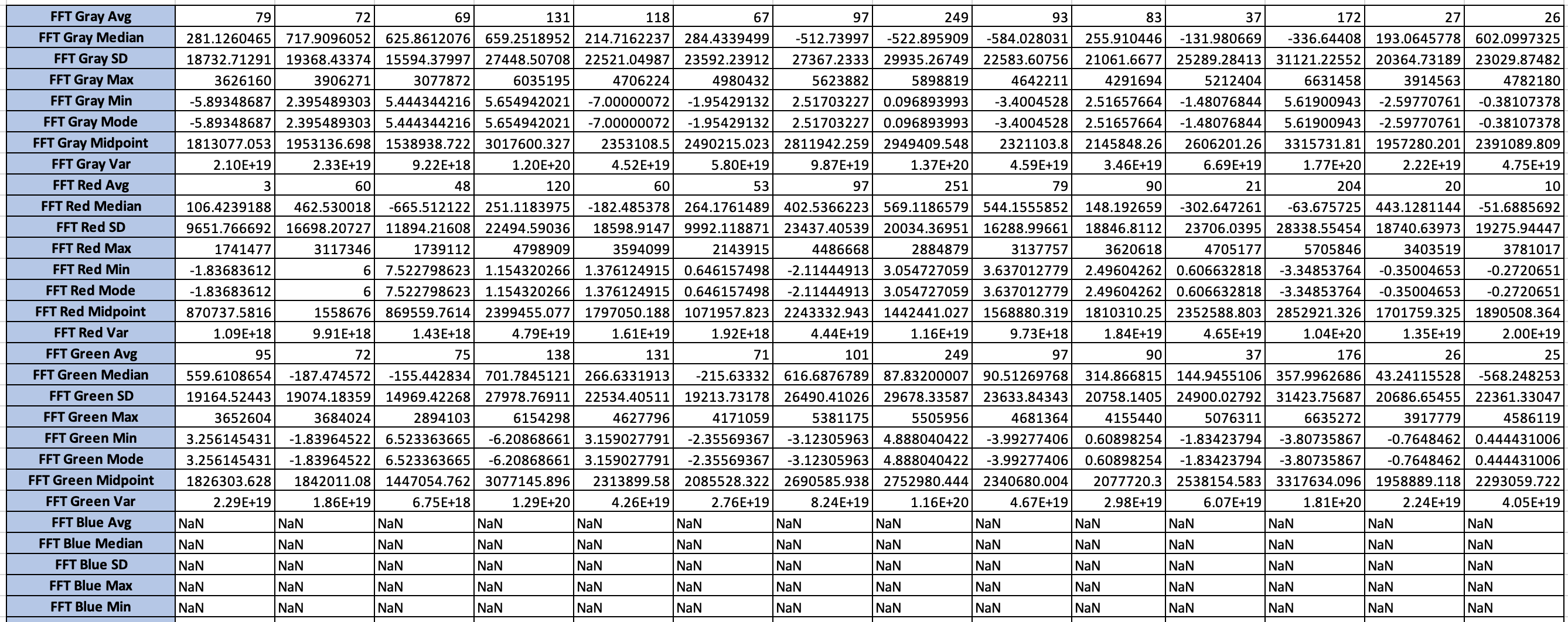
Discrete Wavelet Transform



4 - Dependent Columns

1 - Independent column

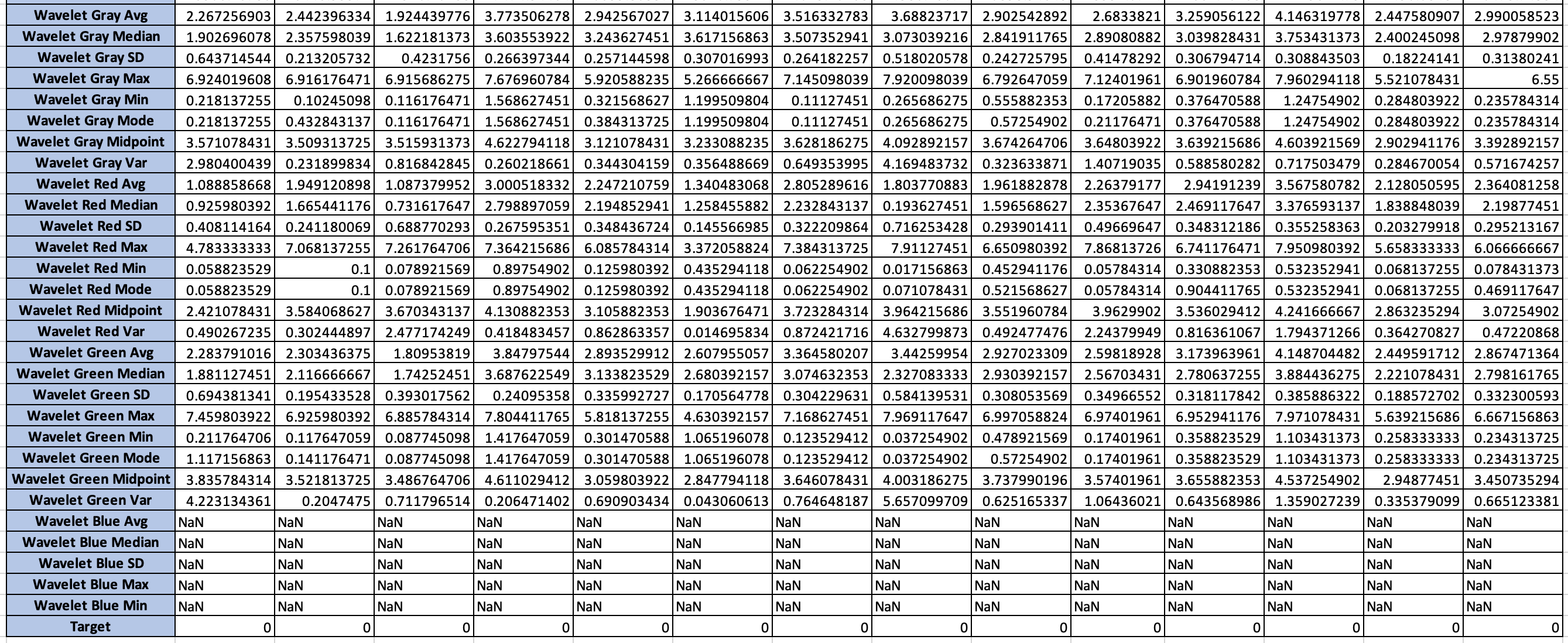
**Frequency Features Extraction using - FFT**



29 - Dependent Columns

1 - Independent column

**Frequency Features Extraction using - Wavelet**



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**

**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.

1. <https://www.dataquest.io/blog/sci-kit-learn-tutorial/>
2. <https://scikit-learn.org/stable/>