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

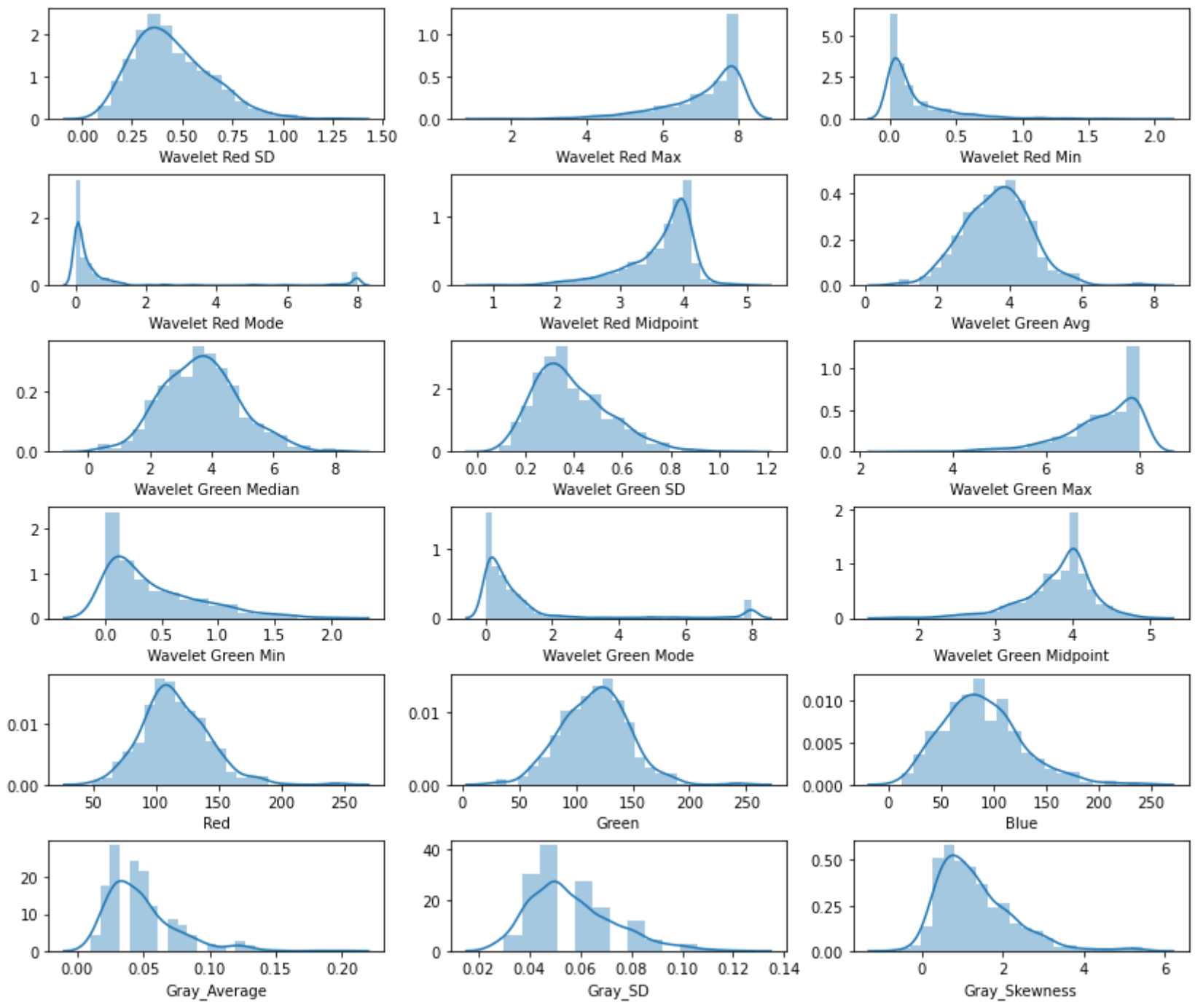
In [10]:
```
df.to_csv('Shuffled_df.csv', header=True, index=False)
```

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)
```



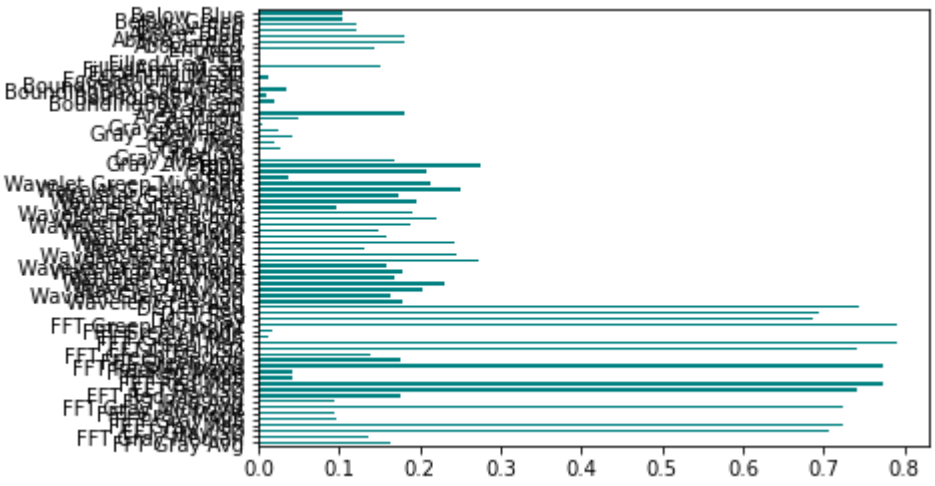
FEATURE SELECTION

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()
```