```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import os

data1=os.listdir('/kaggle/input/raveling-detection-ce784a-2023/
mod_ravelling_dataset/train/Non_raveling')
data2=os.listdir('/kaggle/input/raveling-detection-ce784a-2023/mod_rav
elling_dataset/train/Raveling')

data1array=np.array(data1)
data2array=np.array(data2)

test_data=os.listdir('/kaggle/input/raveling-detection-ce784a-2023/
mod_ravelling_dataset/test')
test_data=np.array(test_data)

# Concatenating Non_raveling and raveling array
data=np.concatenate([data1,data2])
type(data)
```

```
numpy.ndarray
```

```
from skimage.io import imread, imshow
image_data=[]
test_image_data=[]
for i in data1array:

image=imread('/kaggle/input/raveling-detection-ce784a-2023/mod_ravelli
ng_dataset/train/Non_raveling/'+i)
    image_data.append(image)

for j in data2array:

image1=imread('/kaggle/input/raveling-detection-ce784a-2023/mod_ravell
ing_dataset/train/Raveling/'+j)
    image_data.append(image1)
# Storing the test images in the list
for k in test_data:

image2=imread('/kaggle/input/raveling-detection-ce784a-2023/mod_ravell
ing_dataset/test/'+k)
    test_image_data.append(image2)

image_data=np.array(image_data)

image_data.shape
```

```
(700, 100, 100, 3)
```

```
image_data
```

```
array([[[[107, 118, 122],
         [109, 120, 124],
         [114, 123, 128],
         ...,
         [105, 105, 107],
         [106, 106, 108],
         [105, 105, 107]],

        [[120, 130, 132],
         [121, 131, 133],
         [122, 130, 133],
         ...,
         [106, 106, 108],
         [105, 105, 107],
         [104, 104, 106]],

        [[133, 138, 141],
         [133, 138, 141],
         [130, 135, 138],
         ...,
         [106, 105, 110],
         [105, 104, 109],
         [102, 101, 106]],

        ...,

        [[ 82,  86,  87],
         [ 82,  86,  87],
         [ 83,  87,  88],
         ...,
         [ 77,  81,  84],
         [ 76,  80,  83],
         [ 76,  80,  83]],

        [[ 84,  85,  87],
         [ 84,  85,  87],
         [ 86,  87,  89],
         ...,
         [ 78,  82,  85],
         [ 77,  81,  84],
         [ 76,  80,  83]],

        [[ 85,  86,  90],
         [ 85,  86,  90],
         [ 87,  88,  92],
         ...,
         [ 78,  82,  85],
         [ 77,  81,  84],
```

```
  [ 76,  80,  83]]],


[[[ 95,  98, 103],
  [ 95,  98, 103],
  [ 93,  96, 101],
  ...,
  [ 92,  95, 100],
  [ 89,  92,  97],
  [ 86,  89,  94]],

 [[ 96,  99, 104],
  [ 96,  99, 104],
  [ 95,  98, 103],
  ...,
  [ 89,  92,  97],
  [ 86,  89,  94],
  [ 84,  87,  92]],

 [[ 93,  96, 101],
  [ 93,  96, 101],
  [ 93,  96, 101],
  ...,
  [ 88,  91,  96],
  [ 86,  89,  94],
  [ 84,  87,  92]],

 ...,

 [[ 91,  92,  96],
  [ 90,  91,  95],
  [ 88,  89,  93],
  ...,
  [105, 106, 110],
  [105, 106, 110],
  [105, 106, 110]],

 [[ 90,  91,  95],
  [ 91,  92,  96],
  [ 91,  92,  96],
  ...,
  [102, 103, 107],
  [104, 105, 109],
  [105, 106, 110]],

 [[ 90,  91,  95],
  [ 92,  93,  97],
  [ 95,  96, 100],
  ...,
```

```
  [100, 101, 105],
  [103, 104, 108],
  [104, 105, 109]]],


[[[115, 119, 128],
  [110, 114, 123],
  [105, 112, 118],
  ...,
  [ 84,  82,  83],
  [ 85,  83,  84],
  [113, 111, 112]],

 [[117, 121, 130],
  [111, 115, 124],
  [106, 113, 119],
  ...,
  [ 78,  76,  79],
  [ 85,  83,  86],
  [102, 100, 103]],

 [[114, 119, 125],
  [112, 117, 123],
  [104, 109, 115],
  ...,
  [ 76,  75,  80],
  [ 84,  83,  88],
  [ 89,  88,  93]],

 ...,

 [[ 77,  86,  95],
  [ 89,  98, 107],
  [104, 113, 122],
  ...,
  [178, 168, 166],
  [169, 159, 157],
  [153, 143, 141]],

 [[ 92, 101, 110],
  [100, 109, 118],
  [112, 120, 131],
  ...,
  [171, 157, 148],
  [168, 154, 145],
  [163, 149, 140]],

 [[105, 113, 124],
  [109, 117, 128],
```

```
  [117, 125, 136],
  ...,
  [177, 157, 148],
  [176, 158, 148],
  [177, 159, 149]]],


...,


[[[ 97,  99,  98],
  [113, 115, 114],
  [129, 131, 130],
  ...,
  [ 68,  69,  73],
  [ 81,  82,  86],
  [ 96,  97, 101]],

 [[ 83,  83,  85],
  [ 75,  75,  77],
  [ 75,  75,  77],
  ...,
  [ 84,  85,  89],
  [ 98,  99, 103],
  [111, 112, 116]],

 [[ 81,  80,  85],
  [ 66,  65,  70],
  [ 55,  55,  57],
  ...,
  [ 97,  96, 101],
  [114, 113, 118],
  [129, 128, 133]],

 ...,

 [[ 56,  57,  61],
  [ 75,  76,  78],
  [ 99,  99, 101],
  ...,
  [183, 182, 177],
  [149, 148, 143],
  [150, 149, 144]],

 [[ 54,  55,  57],
  [ 74,  75,  77],
  [104, 104, 106],
  ...,
  [162, 161, 156],
```

```
  [156, 155, 150],
  [167, 166, 161]],

 [[ 45,  46,  48],
  [ 60,  61,  63],
  [ 89,  89,  91],
  ...,
  [162, 162, 154],
  [165, 165, 157],
  [162, 162, 154]]],


[[[220, 214, 198],
  [212, 206, 192],
  [223, 217, 203],
  ...,
  [ 74,  83,  88],
  [ 77,  86,  91],
  [ 79,  86,  92]],

 [[188, 182, 168],
  [198, 192, 178],
  [198, 192, 178],
  ...,
  [ 75,  84,  89],
  [ 79,  86,  92],
  [ 79,  87,  90]],

 [[151, 145, 133],
  [148, 142, 130],
  [150, 144, 132],
  ...,
  [ 72,  80,  83],
  [ 75,  83,  86],
  [ 77,  85,  88]],

 ...,

 [[107, 108, 112],
  [117, 118, 122],
  [126, 126, 128],
  ...,
  [104, 108, 107],
  [ 94,  98,  97],
  [100, 104, 103]],

 [[107, 108, 110],
  [114, 115, 117],
  [122, 124, 123],
```

```
  ...,
  [ 97, 101, 100],
  [ 85,  89,  88],
  [ 90,  94,  93]],

 [[106, 110, 109],
  [113, 117, 116],
  [119, 123, 122],
  ...,
  [ 85,  89,  88],
  [ 75,  79,  78],
  [ 81,  85,  84]]],


[[[ 93, 101, 103],
  [ 81,  89,  91],
  [ 69,  81,  81],
  ...,
  [113, 124, 130],
  [116, 127, 133],
  [119, 128, 133]],

 [[132, 136, 137],
  [114, 118, 119],
  [ 92,  98,  98],
  ...,
  [115, 126, 132],
  [118, 129, 133],
  [121, 130, 135]],

 [[163, 162, 160],
  [147, 146, 142],
  [126, 127, 122],
  ...,
  [117, 128, 132],
  [118, 129, 133],
  [121, 130, 135]],

 ...,

 [[123, 138, 145],
  [126, 139, 145],
  [136, 146, 148],
  ...,
  [187, 183, 172],
  [179, 175, 164],
  [177, 173, 162]],

 [[120, 137, 144],
```

```
          [125, 140, 145],
          [129, 140, 142],
          ...,
          [188, 184, 172],
          [178, 174, 162],
          [175, 171, 159]],

         [[113, 133, 140],
          [125, 142, 149],
          [130, 144, 145],
          ...,
          [193, 191, 176],
          [183, 181, 166],
          [178, 176, 161]]], dtype=uint8)
```

```
#Creating features - mean, standard deviation, skew, kurtosis
mean_pix_val=[]
for i in image_data:
    m=np.mean(i, axis=(0,1,2))
    mean_pix_val.append(m)
# forming list of mean pixel values for test dataset
test_mean_pix_val=[]
for i in test_image_data:
    mt=np.mean(i, axis=(0,1,2))
    test_mean_pix_val.append(mt)

mean_pix_val=np.array(mean_pix_val)
test_mean_pix_val=np.array(test_mean_pix_val)
mean_pix_val.shape
```

```
(700,)
```

```
# forming list of std. dev of pixel values

std_pix_val=[]
for i in image_data:
    s=np.std(i, axis=(0,1,2))
    std_pix_val.append(s)
std_pix_val=np.array(std_pix_val)
# forming list of std. dev of pixel values for test dataset
test_std_pix_val=[]
for i in test_image_data:
    st=np.std(i, axis=(0,1,2))
    test_std_pix_val.append(st)
test_std_pix_val=np.array(test_std_pix_val)

# forming list of skewness of pixel values
from scipy.stats import skew,kurtosis
skew_pix_val=[]
for i in image_data:
    skw=skew(i, axis=None)
```

```
    skew_pix_val.append(skw)
skew_pix_val=np.array(skew_pix_val)

# forming list of skewness of pixel values for test dataset
test_skew_pix_val=[]
for i in test_image_data:
    skwt=skew(i, axis=None)
    test_skew_pix_val.append(skwt)
test_skew_pix_val=np.array(test_skew_pix_val)

# forming list of kurtosis of pixel values
kurtosis_pix_val=[]
for i in image_data:
    kurt=kurtosis(i, axis=None)
    kurtosis_pix_val.append(kurt)
kurtosis_pix_val=np.array(kurtosis_pix_val)

# forming list of kurtosis of pixel values for test dataset
test_kurtosis_pix_val=[]
for i in test_image_data:
    kurt=kurtosis(i, axis=None)
    test_kurtosis_pix_val.append(kurt)
test_kurtosis_pix_val=np.array(test_kurtosis_pix_val)

# Creating a dataframe combining all the features
new_df=pd.DataFrame(data=[mean_pix_val,std_pix_val,skew_pix_val,kurtos
is_pix_val])
new_df=new_df.T
new_df
```

	0	1	2	3
0	100.982767	15.326150	0.813725	2.229210
1	113.600933	23.875414	1.311332	2.442279
2	128.915033	32.323276	0.324928	-0.160714
3	109.174133	27.663724	-0.242535	1.301507
4	92.311467	4.986681	1.065992	5.230708
..	...	...	...	...
695	130.099400	20.460961	-0.686226	0.585171
696	124.379367	33.208679	0.736603	-0.309305
697	103.750467	37.631947	0.214915	0.201746
698	128.136367	35.750421	0.421898	-0.427010
699	148.229667	30.649320	-0.188027	-0.291086

[700 rows x 4 columns]

```
# Creating a dataframe combining all the features for test dataset
new_test_df=pd.DataFrame(data=[test_mean_pix_val,test_std_pix_val,test
_skew_pix_val,test_kurtosis_pix_val])
new_test_df=new_test_df.T
new_test_df
```

```
              0          1         2          3
0    102.415900  32.463761  0.123269   0.361713
1    132.515533  32.248923 -0.068662  -0.596367
2    117.461800  32.603269  0.428191  -0.270314
3    160.333533  22.389330  0.015995  -0.083226
4     84.447200  13.063507  2.659176  11.627158
..          ...        ...       ...        ...
295  123.722200  23.080805 -0.727106   0.670490
296  120.257000  21.438355  0.738814   2.374624
297  151.217233  22.851005  0.074730  -0.415092
298  127.935133  40.267035  0.237494  -0.761378
299  142.330067  35.715390 -0.527356   0.015707

[300 rows x 4 columns]
```

```
new_df.columns=['mean','std','skew','kurtosis']
new_df
```

```
           mean        std      skew  kurtosis
0    100.982767  15.326150  0.813725  2.229210
1    113.600933  23.875414  1.311332  2.442279
2    128.915033  32.323276  0.324928 -0.160714
3    109.174133  27.663724 -0.242535  1.301507
4     92.311467   4.986681  1.065992  5.230708
..          ...        ...       ...       ...
695  130.099400  20.460961 -0.686226  0.585171
696  124.379367  33.208679  0.736603 -0.309305
697  103.750467  37.631947  0.214915  0.201746
698  128.136367  35.750421  0.421898 -0.427010
699  148.229667  30.649320 -0.188027 -0.291086

[700 rows x 4 columns]
```

```
# Finally naming the columns of the test dataframe
new_test_df.columns=['mean','std','skew','kurtosis']
new_test_df
```

```
           mean        std      skew   kurtosis
0    102.415900  32.463761  0.123269   0.361713
1    132.515533  32.248923 -0.068662  -0.596367
2    117.461800  32.603269  0.428191  -0.270314
3    160.333533  22.389330  0.015995  -0.083226
4     84.447200  13.063507  2.659176  11.627158
..          ...        ...       ...        ...
295  123.722200  23.080805 -0.727106   0.670490
296  120.257000  21.438355  0.738814   2.374624
297  151.217233  22.851005  0.074730  -0.415092
298  127.935133  40.267035  0.237494  -0.761378
299  142.330067  35.715390 -0.527356   0.015707

[300 rows x 4 columns]
```

```
# Adding a column for which 0 represent Non raveling and 1 represent
raveling
y1=np.zeros(350)
y2=np.ones(350)
y=np.concatenate([y1,y2])
y=pd.DataFrame(y,columns=['y'])
new_df=pd.concat([new_df,y],axis=1)
new_df
```

	mean	std	skew	kurtosis	y
0	100.982767	15.326150	0.813725	2.229210	0.0
1	113.600933	23.875414	1.311332	2.442279	0.0
2	128.915033	32.323276	0.324928	-0.160714	0.0
3	109.174133	27.663724	-0.242535	1.301507	0.0
4	92.311467	4.986681	1.065992	5.230708	0.0
..	...	...	...	...	...
695	130.099400	20.460961	-0.686226	0.585171	1.0
696	124.379367	33.208679	0.736603	-0.309305	1.0
697	103.750467	37.631947	0.214915	0.201746	1.0
698	128.136367	35.750421	0.421898	-0.427010	1.0
699	148.229667	30.649320	-0.188027	-0.291086	1.0

[700 rows x 5 columns]

```
# Applying Train-test split and breaking data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split( new_df.drop(columns='y',axis=1), new_df['y'],
test_size=0.7,shuffle=True, random_state=42)
X_train.shape
y_train.shape
```

(210,)

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
```

StandardScaler()

StandardScaler()

```
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)
from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()
logmodel.fit(X_train,y_train)
LogisticRegression()
```

LogisticRegression()

```
#predicting the data
pred=logmodel.predict(X_test)
pred
```

```
array([0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1.,
1.,
       1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0.,
0.,
       0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
1.,
       1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0.,
1.,
       0., 1., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0.,
0.,
       1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1.,
0.,
       1., 0., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1.,
1.,
       0., 1., 1., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
0.,
       0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0.,
1.,
       1., 0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 0., 0.,
1.,
       1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 0., 1., 1., 0., 1., 1.,
1.,
       0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0.,
0.,
       0., 0., 1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 1., 0.,
1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0.,
0.,
       1., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1., 1., 0., 1.,
0.,
       1., 0., 1., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1.,
0.,
       0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 0.,
1.,
       1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1.,
1.,
       1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.,
1.,
       0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0., 1., 1.,
0.,
       0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1.,
0.,
       1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1.,
0.,
```

```
       0., 1., 1., 0., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1.,
0.,
       1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1.,
1.,
       1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1.,
1.,
       1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1.,
0.,
       1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1.,
1.,
       0., 1., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1.])
```

```
# checking the accuracy
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.67      0.77       253
         1.0       0.72      0.93      0.81       237

    accuracy                           0.79       490
   macro avg       0.82      0.80      0.79       490
weighted avg       0.82      0.79      0.79       490
```

```
# Final Prediction for test
test_pred=logmodel.predict(new_test_df)
test_pred
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/base.py:444:
UserWarning: X has feature names, but LogisticRegression was fitted
without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted
without"
```

```
array([1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
```

```
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
0.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```python
# Creating .CSV in the give submission format
test_pred=pd.DataFrame(test_pred)
test_data=pd.DataFrame(test_data)
test_pred_1=pd.concat([test_data,test_pred],axis=1)

test_pred_1.columns=['filename','class']
test_pred_1=test_pred_1.set_index('filename')

test_pred_1['class']=np.where(test_pred_1['class']==1,'Raveling','Nonr
aveling')
test_pred_1
```

```
                class
filename
208.jpg      Raveling
45.jpg       Raveling
56.jpg       Raveling
89.jpg       Raveling
20.jpg    Nonraveling
...               ...
213.jpg      Raveling
136.jpg      Raveling
90.jpg       Raveling
25.jpg       Raveling
147.jpg      Raveling

[300 rows x 1 columns]
```