```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skimage
from skimage.io import imshow, imread
%matplotlib inline
```

```
import matplotlib.pyplot as plt
import os
import glob
from skimage import io, color
from skimage.feature.texture import greycomatrix, greycoprops
import numpy as np
import pandas as pd
from scipy.stats import kurtosis
from scipy.stats import skew
from scipy.stats import entropy
import cv2
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, callbacks
import PIL
import skimage
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
import os, sys
from IPython.display import display
from IPython.display import Image as _Imgdis
from PIL import Image
import numpy as np
from time import time
from time import sleep
```

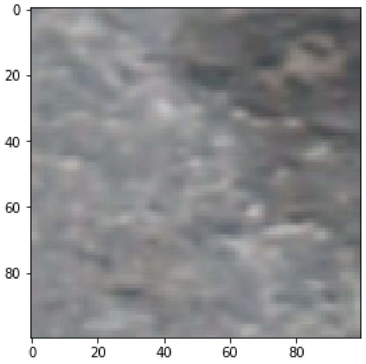
```
train_path='/content/drive/MyDrive/mod_ravelling_dataset/train'
```

```
Non_raveling_img_path='/content/drive/MyDrive/mod_ravelling_dataset/train/Non_raveling/image1.jpg'
Reveling_img_path='/content/drive/MyDrive/mod_ravelling_dataset/train/Raveling/image100.jpg'
```

```
categories=['Non_raveling','Raveling']
```

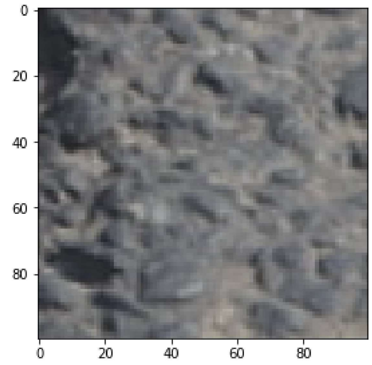
```
from skimage import io,color
image = io.imread(Non_raveling_img_path)
imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7f025af38160>
```

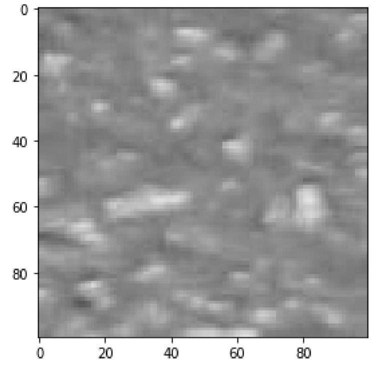


```
image = io.imread(Reveling_img_path)
imshow(image)
```

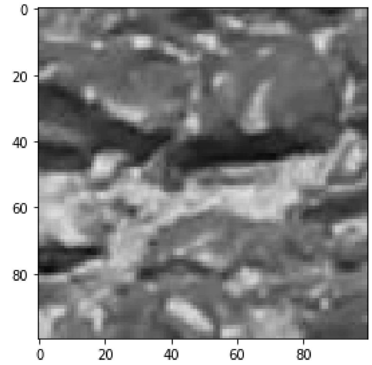
```
<matplotlib.image.AxesImage at 0x7f025826cd60>
```



```
import os
NonRaveling_directory = '/content/drive/MyDrive/raveling-detection-ce784a-2023/mod_ravelling_dataset/train/Non_raveling'
for filename in os.listdir(NonRaveling_directory):
  if filename.endswith(".jpg") or filename.endswith(".png"):
    filepath = os.path.join(NonRaveling_directory, filename)
    image = io.imread(filepath)
    io.imshow(image)
```



```
import os
Raveling_directory = '/content/drive/MyDrive/raveling-detection-ce784a-2023/mod_ravelling_dataset/train/Raveling'
for filename in os.listdir(Raveling_directory):
  if filename.endswith(".jpg") or filename.endswith(".png"):
    filepath = os.path.join(Raveling_directory, filename)
    image = io.imread(filepath)
    io.imshow(image)
```



```
#convert images to grayscale
# from PIL import Image
# import os

# def convert_to_grayscale(path1):
#     for filename in os.listdir(path1):
```

```
#         with Image.open(f"{path1}/{filename}") as im:
#             im = im.convert("L")
#             im.save(f"{path1}/{filename}")
# convert_to_grayscale("/content/drive/MyDrive/raveling-detection-ce784a-2023/mod_ravelling_dataset/train/Non_raveling")


#convert images to grayscale

# def convert_to_grayscale(path2):
#     for filename in os.listdir(path2):
#         with Image.open(f"{path2}/{filename}") as im:
#             im = im.convert("L")
#             im.save(f"{path2}/{filename}")
# convert_to_grayscale("/content/drive/MyDrive/raveling-detection-ce784a-2023/mod_ravelling_dataset/train/Raveling")


features = {}

features['redMean'] = []
features['blueMean'] = []
features['greenMean'] = []

features['redStd'] = []
features['blueStd'] = []
features['greenStd'] = []

features['redSkew'] = []
features['blueSkew'] = []
features['greenSkew'] = []

features['redKurt'] = []
features['blueKurt'] = []
features['greenKurt'] = []

features['Entropy0'] = []
# features['Entropy1'] = []
# features['Entropy2'] = []
# features['Entropy3'] = []
# features['Entropy4'] = []
# features['Entropy5'] = []
# features['Entropy6'] = []
# features['Entropy7'] = []
# features['blueEntropy'] = []
# features['greenEntropy'] = []



features['Classes'] = []


for i in range(len(categories)):
  path = os.path.join(train_path,categories[i],'*')
  path = glob.glob(path)

  for p in path:
    features['Classes'].append(i)
    image = io.imread(p)


    img_gs = cv2.imread(p,cv2.IMREAD_GRAYSCALE)
    img_gs = skimage.feature.greycomatrix(img_gs, [1], [np.pi/2])
    features['Entropy0'].append(skimage.measure.shannon_entropy(np.reshape(img_gs,(256,256))))

    imgR = image[:,:,0]
    features['redMean'].append(np.mean(imgR))
    features['redStd'].append(np.std(imgR))
    features['redSkew'].append(np.mean(skew(imgR)))
    features['redKurt'].append(np.mean(kurtosis(imgR)))

    imgB = image[:,:,1]
    features['blueMean'].append(np.mean(imgB))
    features['blueStd'].append(np.std(imgB))
    features['blueSkew'].append(np.mean(skew(imgB)))
    features['blueKurt'].append(np.mean(kurtosis(imgB)))

    imgG = image[:,:,2]
    features['greenMean'].append(np.mean(imgG))
    features['greenStd'].append(np.std(imgG))
    features['greenSkew'].append(np.mean(skew(imgG)))
    features['greenKurt'].append(np.mean(kurtosis(imgG)))

train_dataFrame = pd.DataFrame.from_dict(features)
```

```
train_dataFrame.head()
```

	redMean	blueMean	greenMean	redStd	blueStd	greenStd	redSkew	blueSkew	greenSkew	redKurt
0	96.6887	99.7035	104.2558	12.577575	11.161585	10.243074	1.435075	1.372324	1.197007	3.328718
1	131.9528	132.7625	133.1749	30.126762	30.161474	30.040721	0.769982	0.778120	0.770225	0.715707
2	127.2294	132.1558	136.5937	23.110586	20.868625	19.783271	0.223935	0.179741	0.066978	0.025122
3	126.4984	126.8334	126.2913	23.424577	21.329089	19.565184	0.548684	0.595919	0.641238	0.242112
4	123.5977	124.9423	125.9379	18.149156	18.759424	18.432288	0.456985	0.470760	0.490505	1.518233

```
X_train = train_dataFrame.drop('Classes', axis=1)
y_train = train_dataFrame.Classes
```

```
input_shape = [X_train.shape[1]]
```

```
model = keras.models.Sequential()

model.add(tf.keras.layers.BatchNormalization(input_shape = input_shape))

model.add(tf.keras.layers.Dense(128,activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.BatchNormalization())


model.add(tf.keras.layers.Dense(64,activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(32,activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(16,activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(8,activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.1))
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(1,activation = 'sigmoid'))


model.compile(optimizer=tf.keras.optimizers.Adam(),loss="binary_crossentropy",metrics=['accuracy'])
model.fit(X_train,y_train,batch_size = 20,epochs = 1000)
```

```
Epoch 537/1000
35/35 [==============================] - 0s 6ms/step - loss: 0.2700 - accuracy: 0.8857
Epoch 538/1000
35/35 [==============================] - 0s 6ms/step - loss: 0.2192 - accuracy: 0.9071
Epoch 539/1000
35/35 [==============================] - 0s 6ms/step - loss: 0.2205 - accuracy: 0.9143
Epoch 540/1000
35/35 [==============================] - 0s 6ms/step - loss: 0.2111 - accuracy: 0.9214
Epoch 541/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2516 - accuracy: 0.8971
Epoch 542/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2282 - accuracy: 0.9057
Epoch 543/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2175 - accuracy: 0.9129
Epoch 544/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2132 - accuracy: 0.9157
Epoch 545/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2301 - accuracy: 0.9200
Epoch 546/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2794 - accuracy: 0.8986
Epoch 547/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2396 - accuracy: 0.9029
Epoch 548/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2623 - accuracy: 0.8886
Epoch 549/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2685 - accuracy: 0.8943
Epoch 550/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2649 - accuracy: 0.9000
Epoch 551/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2073 - accuracy: 0.9157
Epoch 552/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2225 - accuracy: 0.9143
Epoch 553/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2579 - accuracy: 0.9086
Epoch 554/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2083 - accuracy: 0.9214
Epoch 555/1000
35/35 [==============================] - 0s 4ms/step - loss: 0.2763 - accuracy: 0.8814
Epoch 556/1000
35/35 [==============================] - 0s 5ms/step - loss: 0.2588 - accuracy: 0.9000
```

```
test_path = '/content/drive/MyDrive/mod_ravelling_dataset/test'


featuresTest = {}

featuresTest['redMean'] = []
featuresTest['blueMean'] = []
featuresTest['greenMean'] = []

featuresTest['redStd'] = []
featuresTest['blueStd'] = []
featuresTest['greenStd'] = []

featuresTest['redSkew'] = []
featuresTest['blueSkew'] = []
featuresTest['greenSkew'] = []

featuresTest['redKurt'] = []
featuresTest['blueKurt'] = []
featuresTest['greenKurt'] = []

featuresTest['Entropy0'] = []
# featuresTest['Entropy1'] = []
# featuresTest['Entropy2'] = []
# featuresTest['Entropy3'] = []
# features['Entropy2'] = []
# features['Entropy3'] = []
# features['Entropy4'] = []
# features['Entropy5'] = []
# features['Entropy6'] = []
# features['Entropy7'] = []
#  featuresTest['blueEntropy'] = []
#  featuresTest['greenEntropy'] = []

featuresTest['filename'] = []


for img in (os.listdir(test_path)):
  p = os.path.join(test_path,img)
  #path = glob.glob(path)
  image = io.imread(p)


  img_gs = cv2.imread(p,cv2.IMREAD_GRAYSCALE)
  img_gs = skimage.feature.greycomatrix(img_gs, [1], [np.pi/2])

  featuresTest['Entropy0'].append(skimage.measure.shannon_entropy(np.reshape(img_gs,(256,256))))
  # featuresTest['Entropy1'].append(skimage.measure.shannon_entropy(np.reshape(img_gs,(256,256))))
  # featuresTest['Entropy2'].append(skimage.measure.shannon_entropy(np.reshape(img_gs,(256,256))))
  # featuresTest['Entropy3'].append(skimage.measure.shannon_entropy(np.reshape(img_gs,(256,256))))

  imgR = image[:,:,0]
  featuresTest['redMean'].append(np.mean(imgR))
  featuresTest['redStd'].append(np.std(imgR))
  featuresTest['redSkew'].append(np.mean(skew(imgR)))
  featuresTest['redKurt'].append(np.mean(kurtosis(imgR)))

  imgB = image[:,:,1]
  featuresTest['blueMean'].append(np.mean(imgB))
  featuresTest['blueStd'].append(np.std(imgB))
  featuresTest['blueSkew'].append(np.mean(skew(imgB)))
  featuresTest['blueKurt'].append(np.mean(kurtosis(imgB)))

  imgG = image[:,:,2]
  featuresTest['greenMean'].append(np.mean(imgG))
  featuresTest['greenStd'].append(np.std(imgG))
  featuresTest['greenSkew'].append(np.mean(skew(imgG)))
  featuresTest['greenKurt'].append(np.mean(kurtosis(imgG)))

  featuresTest['filename'].append(img)

test_dataFrame = pd.DataFrame.from_dict(featuresTest)
```

```
test_dataFrame.head()
```

	redMean	blueMean	greenMean	redStd	blueStd	greenStd	redSkew	blueSkew	greenSkew	redKu
0	127.3126	128.2218	127.9632	35.946294	31.602579	27.632822	0.311149	0.198299	-0.055981	-0.4459
1	136.7490	134.8682	130.5521	43.689308	41.688996	39.067212	-0.224221	-0.221600	-0.229627	-0.6538
2	129.2429	126.6656	125.1148	35.356410	34.518206	32.500277	0.349248	0.393751	0.414430	-0.2476
3	106.2044	105.7387	104.1821	27.319788	25.921628	23.716322	0.825809	0.855422	0.816398	0.5150
4	134.2944	131.7906	127.8910	21.932440	20.634669	18.014914	-0.704286	-0.687406	-0.602224	0.1877

```
X_test = test_dataFrame.drop(['filename'], axis = 1)
```

```
predictions = model.predict(X_test)
```

```
    10/10 [==============================] - 0s 2ms/step
```

```
classes = []
for pred in predictions:
  if(pred < 0.5): classes.append('Non_raveling')
  else: classes.append('Raveling')
```

```
test_dataFrame['class'] = classes
```

```
df_sub = test_dataFrame[['filename','class']]
```

```
df_sub.head()
```

	filename	class
0	101.jpg	Raveling
1	108.jpg	Raveling
2	110.jpg	Non_raveling
3	114.jpg	Non_raveling

```
df_sub.to_csv('Submission', index = False)
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:14 PM