CHAPTER 6

IMPLEMENTATION

Statistical Properties of Colour Channels

```
from PIL import Image
importnumpy as np
importscipy
fromscipy import stats
import cv2
importskimage
fromskimage.measure import shannon_entropy
importmatplotlib.pyplot as plt
import pandas as pd
classsmp_values():
defmean_properties(image):
    #image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image_1.jpg")
b,g,r=cv2.split(image)
rgb_img=cv2.merge([r,g,b])
    #plt.imshow(rgb_img)
    #plt.imshow(image)
red=image[:,:,2]
    Mean=list()
    Range=list()
```

```
        Deviation=list()

        Entropy=list()

Skewness=list()

        Kurtosis=list()

        r1=r.mean()

Mean.append(r1)

        r2=skimage.measure.shannon_entropy(r,base=10)

Entropy.append(r2)

        r3=r.std()

Deviation.append(r3)

        r4=np.ptp(r)

Range.append(r4)

        t=scipy.stats.skew(r)

        #print(b)

        r5=t.mean()

Skewness.append(r5)

        c=scipy.stats.kurtosis(r,axis=0,fisher=False)

        #print(c)

        r6=c.mean()

Kurtosis.append(r6)

        g1=g.mean()

Mean.append(g1)

        g2=skimage.measure.shannon_entropy(g,base=10)
```

```
Entropy.append(g2)

        g3=g.std()

Deviation.append(g3)

        g4=np.ptp(g)

Range.append(g4)

        e=scipy.stats.skew(g)

        #print(b)

        g5=e.mean()

Skewness.append(g5)

        f=scipy.stats.kurtosis(g,axis=0,fisher=False)

        #print(c)

        g6=f.mean()

Kurtosis.append(g6)

        b1=b.mean()

Mean.append(b1)

        b2=skimage.measure.shannon_entropy(b,base=10)

Entropy.append(b2)

        b3=b.std()

Deviation.append(b3)

        b4=np.ptp(b)

Range.append(b4)

        h=scipy.stats.skew(b)

        #print(b)
```

```
    b5=h.mean()
Skewness.append(b5)
    i=scipy.stats.kurtosis(b,axis=0,fisher=False)
    #print(c)
    b6=i.mean()
Kurtosis.append(b6)
    #print(Mean)
    #print(Entropy)
    #print(Deviation)
    #print(Range)
    #print(Skewness)
    #print(Kurtosis)
    #df1= pd.DataFrame(Mean, columns = ['Mean_R', 'Mean_G','Mean_B'])
    df1= pd.DataFrame([Mean])
    df1.columns =['Mean_R', 'Mean_G','Mean_B']
    df2= pd.DataFrame([Deviation])
    df2.columns =['Standard_deviation_R', 'Standard_deviation_G','Standard_deviation_B']
    df3= pd.DataFrame([Skewness])
    df3.columns =['Skewness_R', 'Skewness_G','Skewness_B']
    df4= pd.DataFrame([Kurtosis])
    df4.columns =['Kurtosis_R', 'Kurtosis_G','Kurtosis_B']
    df5= pd.DataFrame([Entropy])
    df5.columns =['Entropy_R', 'Entropy_G','Entropy_B']
```

```
        df6= pd.DataFrame([Range])

        df6.columns =['Range_R', 'Range_G','Range_B']

frames = [df1, df2, df3, df4, df5, df6]

result = pd.concat(frames, axis=1, sort=False)

return(result)
```

Gray Level Co-Occurrence Matrix (GLCM)

```
importnumpy as np

import cv2

importskimage

fromskimage.feature import greycomatrix, greycoprops

import pandas as pd

classglcm_prop():

defglclm_properties(image):

        #image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image_1.jpg")

result=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        #print(image)

        #print(result)

        #print(result.max())

angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]

entropy=[]

ASM_val=[]

        Contrast=[]
```

```
Corrleation=[]

new_angle=0

for i in range(4):

new_angle=angles[i]

g_e= greycomatrix(result, [1],[new_angle], 256, symmetric=False, normed=True)

entr = skimage.measure.shannon_entropy(g_e)

correlation = greycoprops(g_e, 'correlation')

        ASM = greycoprops(g_e, 'ASM')

contrast = greycoprops(g_e, 'contrast')

entropy.append(entr)

ASM_val.append(ASM[0][0])

Contrast.append(contrast[0][0])

Corrleation.append(correlation[0][0])

     #asm,contrast,corrleation,entropy

     dummy0=[]

dummy0.append(ASM_val[0])

dummy0.append(Contrast[0])

dummy0.append(Corrleation[0])

dummy0.append(entropy[0])

     dummy45=[]

dummy45.append(ASM_val[1])

dummy45.append(Contrast[1])

dummy45.append(Corrleation[1])
```

```
dummy45.append(entropy[1])

        dummy90=[]

dummy90.append(ASM_val[2])

dummy90.append(Contrast[2])

dummy90.append(Corrleation[2])

dummy90.append(entropy[2])

        dummy135=[]

dummy135.append(ASM_val[3])

dummy135.append(Contrast[3])

dummy135.append(Corrleation[3])

dummy135.append(entropy[3])

        df1= pd.DataFrame([dummy0])

        df1.columns =['ASM_0', 'Contrast_0','Corrleation_0','entropy_0']

        df2= pd.DataFrame([dummy45])

        df2.columns =['ASM_45', 'Contrast_45','Corrleation_45','entropy_45']

        df3= pd.DataFrame([dummy90])

        df3.columns =['ASM_90', 'Contrast_90','Corrleation_90','entropy_90']

        df4= pd.DataFrame([dummy135])

        df4.columns =['ASM_135', 'Contrast_135','Corrleation_135','entropy_135']

frames = [df1, df2, df3, df4]

result_glcm = pd.concat(frames, axis=1, sort=False)

return(result_glcm)
```

Gray Level Run Lengths (GLRL)

```
importSimpleITK as sitk
importnumpy as np
import cv2
fromskimage.color import rgb2gray
fromskimage.feature import greycomatrix, greycoprops
import pandas as pd

classglrlm_prop():
defglrlm_properties(image):
        #image=cv2.imread(r"C:\Users\Pavan\Desktop\Project\image_1.jpg")

grayscale = rgb2gray(image)

grayscale = np.array(grayscale)
im = sitk.GetImageFromArray(grayscale)
test_arr = np.ones((grayscale.shape), dtype='uint8')
ma = sitk.GetImageFromArray(test_arr)

        # Store to nrrd:
sitk.WriteImage(im, 'image.nrrd')
sitk.WriteImage(ma, 'mask.nrrd', True)  # enable compression to save disk space

        # or extract features:
fromradiomics import featureextractor
extractor = featureextractor.RadiomicsFeatureExtractor(r'path/to/params.yml')
features = extractor.execute(im, im, label=1)

glrlm=[]

glrlm.append(float(features['original_glrlm_ShortRunEmphasis']))
glrlm.append(float(features['original_glrlm_LongRunEmphasis']))
glrlm.append(float(features['original_glrlm_GrayLevelNonUniformity']))
glrlm.append(float(features['original_glrlm_RunLengthNonUniformity']))
glrlm.append(float(features['original_glrlm_RunPercentage']))
glrlm.append(float(features['original_glrlm_LowGrayLevelRunEmphasis']))
glrlm.append(float(features['original_glrlm_HighGrayLevelRunEmphasis']))
glrlm.append(float(features['original_glrlm_ShortRunLowGrayLevelEmphasis']))
glrlm.append(float(features['original_glrlm_ShortRunHighGrayLevelEmphasis']))
glrlm.append(float(features['original_glrlm_LongRunLowGrayLevelEmphasis']))
glrlm.append(float(features['original_glrlm_LongRunHighGrayLevelEmphasis']))

        df1= pd.DataFrame([glrlm])
        df1.columns =['Short_run_emphasis_0',
'Long_run_emphasis_0','Gray_level_nonuniformity_0'
,'Run_length_nonuniformity_0','Run_percentage_0','Low_gray_level_run_emphasis_0','High
_gray_level_run_emphasis_0','Short_run_Low_gray_level_emphasis_0','Short_run_High_gra
y_level_emphasis_0','Long_run_Low_gray_level_emphasis_0','Long_run_High_gray_level_
emphasis_0']

        df2= pd.DataFrame([glrlm])
        df2.columns =['Short_run_emphasis_45',
'Long_run_emphasis_45','Gray_level_nonuniformity_45'
,'Run_length_nonuniformity_45','Run_percentage_45','Low_gray_level_run_emphasis_45','H
igh_gray_level_run_emphasis_45','Short_run_Low_gray_level_emphasis_45','Short_run_Hig
h_gray_level_emphasis_45','Long_run_Low_gray_level_emphasis_45','Long_run_High_gray
_level_emphasis_45']

        df3= pd.DataFrame([glrlm])
        df3.columns =['Short_run_emphasis_90',
'Long_run_emphasis_90','Gray_level_nonuniformity_90'
,'Run_length_nonuniformity_90','Run_percentage_90','Low_gray_level_run_emphasis_90','H
igh_gray_level_run_emphasis_90','Short_run_Low_gray_level_emphasis_90','Short_run_Hig
h_gray_level_emphasis_90','Long_run_Low_gray_level_emphasis_90','Long_run_High_gray
_level_emphasis_90']
```

```
        df4= pd.DataFrame([glrlm])
        df4.columns =['Short_run_emphasis_135',
'Long_run_emphasis_135','Gray_level_nonuniformity_135'
,'Run_length_nonuniformity_135','Run_percentage_135','Low_gray_level_run_emphasis_13
5','High_gray_level_run_emphasis_135','Short_run_Low_gray_level_emphasis_135','Short_r
un_High_gray_level_emphasis_135','Long_run_Low_gray_level_emphasis_135','Long_run_
High_gray_level_emphasis_135']

frames = [df1, df2, df3, df4]
result_glrlm = pd.concat(frames, axis=1, sort=False)

return(result_glrlm)
```

Support Vector Machine (SVM)

```
import pickle

header_list =['Mean_R', 'Mean_G','Mean_B','Standard_deviation_R',
'Standard_deviation_G','Standard_deviation_B','Skewness_R',
'Skewness_G','Skewness_B','Kurtosis_R', 'Kurtosis_G','Kurtosis_B','Entropy_R',
'Entropy_G','Entropy_B','Range_R', 'Range_G','Range_B','ASM_0',
'Contrast_0','Corrleation_0','entropy_0','ASM_45',
'Contrast_45','Corrleation_45','entropy_45','ASM_90',
'Contrast_90','Corrleation_90','entropy_90','ASM_135',
'Contrast_135','Corrleation_135','entropy_135','Short_run_emphasis_0',
'Long_run_emphasis_0','Gray_level_nonuniformity_0'
,'Run_length_nonuniformity_0','Run_percentage_0','Low_gray_level_run_emphasis_0','High
_gray_level_run_emphasis_0','Short_run_Low_gray_level_emphasis_0','Short_run_High_gra
y_level_emphasis_0','Long_run_Low_gray_level_emphasis_0','Long_run_High_gray_level_
emphasis_0','Short_run_emphasis_45',
'Long_run_emphasis_45','Gray_level_nonuniformity_45'
,'Run_length_nonuniformity_45','Run_percentage_45','Low_gray_level_run_emphasis_45','H
igh_gray_level_run_emphasis_45','Short_run_Low_gray_level_emphasis_45','Short_run_Hig
h_gray_level_emphasis_45','Long_run_Low_gray_level_emphasis_45','Long_run_High_gray
_level_emphasis_45','Short_run_emphasis_90',
'Long_run_emphasis_90','Gray_level_nonuniformity_90'
,'Run_length_nonuniformity_90','Run_percentage_90','Low_gray_level_run_emphasis_90','H
igh_gray_level_run_emphasis_90','Short_run_Low_gray_level_emphasis_90','Short_run_Hig
h_gray_level_emphasis_90','Long_run_Low_gray_level_emphasis_90','Long_run_High_gray
_level_emphasis_90','Short_run_emphasis_135',
'Long_run_emphasis_135','Gray_level_nonuniformity_135'
,'Run_length_nonuniformity_135','Run_percentage_135','Low_gray_level_run_emphasis_13
5','High_gray_level_run_emphasis_135','Short_run_Low_gray_level_emphasis_135','Short_r
un_High_gray_level_emphasis_135','Long_run_Low_gray_level_emphasis_135','Long_run_
High_gray_level_emphasis_135','Result']

import pandas as pd
fromsklearn.model_selection import train_test_split
df = pd.read_csv("dataset.csv", names=header_list)

X_train_1=df.iloc[:,:-1]
Y_train_1=df.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X_train_1,Y_train_1, test_size=0.2)
fromsklearn import svm
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, y_train)

#predicting Y values for test data
Y_pred = classifier.predict(X_test)

#confusion matrix for tested datatsets
fromsklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, Y_pred)

#Store the trainined SVM Model for future use
pickle.dump(classifier,open(r'C:\Users\Pavan\Desktop\Project\corrosion_detection\predictor\
model\model.pkl', 'wb'))
```