**import** **tensorflow** **as** **tf**

**from** **tensorflow** **import** keras

**import** **cv2**

**import** **glob**

**import** **numpy** **as** **np**

norma=[cv2.imread(file) **for** file **in** glob.glob('E:/vinod/phd/normal/\*.jpeg')]

norma[0].shape

(1482, 1989, 3)

covid=[cv2.imread(file) **for** file **in** glob.glob('E:/vinod/phd/covid-chestxray-dataset-master/images/\*.jpeg')]

covid[0].shape

(1482, 1989, 3)

*#cv2.imshow("original", norma[0])*

resized\_images = cv2.resize(norma[0], (300, 300))

*#cv2.imshow("resized", resized\_images)*

mera\_dat = []

**for** i **in** range(60):

desired\_size = 368

im = norma[i]

old\_size = im.shape[:2] *# old\_size is in (height, width) format*

ratio = float(desired\_size)/max(old\_size)

new\_size = tuple([int(x\*ratio) **for** x **in** old\_size])

*# new\_size should be in (width, height) format*

im = cv2.resize(im, (new\_size[1], new\_size[0]))

delta\_w = desired\_size - new\_size[1]

delta\_h = desired\_size - new\_size[0]

top, bottom = delta\_h//2, delta\_h-(delta\_h//2)

left, right = delta\_w//2, delta\_w-(delta\_w//2)

color = [0, 0, 0]

new\_im = cv2.copyMakeBorder(im, top, bottom, left, right, cv2.BORDER\_CONSTANT,

value=color)

*# cv2.imshow("image", new\_im)*

*# cv2.waitKey(0)*

*# cv2.destroyAllWindows()*

*#*

mera\_dat.append(new\_im)

mera\_dat1 = []

**for** i **in** range(100):

desired\_size = 368

im = covid[i]

old\_size = im.shape[:2] *# old\_size is in (height, width) format*

ratio = float(desired\_size)/max(old\_size)

new\_size = tuple([int(x\*ratio) **for** x **in** old\_size])

*# new\_size should be in (width, height) format*

im = cv2.resize(im, (new\_size[1], new\_size[0]))

delta\_w = desired\_size - new\_size[1]

delta\_h = desired\_size - new\_size[0]

top, bottom = delta\_h//2, delta\_h-(delta\_h//2)

left, right = delta\_w//2, delta\_w-(delta\_w//2)

color = [0, 0, 0]

new\_im = cv2.copyMakeBorder(im, top, bottom, left, right, cv2.BORDER\_CONSTANT,

value=color)

*# cv2.imshow("image", new\_im)*

*# cv2.waitKey(0)*

*# cv2.destroyAllWindows()*

*#*

mera\_dat1.append(new\_im)

arr = np.array(mera\_dat)

ar1 = np.array(mera\_dat1)

**import** **matplotlib.pyplot** **as** **plt**

plt.suptitle('Normal X-Ray Reports')

**for** i **in** range(25):

plt.subplot(5, 5, i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(arr[i])

plt.show()

plt.suptitle('Covid Positive X-Ray Reports')

**for** i **in** range(25):

plt.subplot(5, 5, i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(ar1[i])

plt.show()

<Figure size 640x480 with 25 Axes>

<Figure size 640x480 with 25 Axes>

arr = arr.reshape((60, 406272))

ar1 = ar1.reshape((100, 406272))

*#Normalization*

arr = arr / 255

ar1 = ar1 / 255

**import** **pandas** **as** **pd**

dataset = pd.DataFrame(arr)

dataset['label'] = np.ones(60)

dataset.iloc[:, -1]

dataset\_1 = pd.DataFrame(ar1)

dataset\_1['label'] = np.zeros(100)

dataset\_1.iloc[:, -1]

dataset\_master = pd.concat([dataset, dataset\_1])

X = dataset\_master.iloc[:, :406272]

y = dataset\_master.iloc[:, -1]

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)

**from** **sklearn.tree** **import** DecisionTreeClassifier

dtf = DecisionTreeClassifier()

dtf.fit(X\_train, y\_train)

dtf.score(X\_test, y\_test)

0.875

dtf.score(X\_train, y\_train)

1.0

y\_pred = dtf.predict(X\_test)

**from** **sklearn.metrics** **import** precision\_score, recall\_score, f1\_score

precision\_score(y\_test, y\_pred)

0.8333333333333334

recall\_score(y\_test, y\_pred)

0.7692307692307693

f1\_score(y\_test, y\_pred)

0.8

test\_labels = ['Normal' **if** i == 1 **else** 'Covid Positive' **for** i **in** y\_pred]

test\_actuals = ['Normal' **if** i == 1 **else** 'Covid Positive' **for** i **in** y\_test]

X\_test\_plot = (X\_test.values).reshape((40, 368, 368, 3))

fig = plt.figure(figsize = (20, 20))

fig.suptitle('AI Model')

**for** i **in** range(9):

plt.subplot(3, 3, i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(X\_test\_plot[i])

plt.xlabel('Actual **{}** **\n**Predicted **{}**'.format(test\_actuals[i], test\_labels[i]), fontsize = 14, color='b')

plt.show()