A

PROJECT REPORT

ON

“HUMAN IMAGE CLASSIFICATION MODEL”

SUBMITTED TO

VERZEO

IN PERTIAL FULFILLMENT OF TWO MONTHS FULL TIME IN ARTIFICIAL INTELLIGENCE WITH VERZEO

SUBMITTED BY

TEAM – B12

## NAME: - ASMITA NIMESH

## BATCH: - B12

## SUBJECT: - AI-MAJOR-FEB

**PROBLEM STATEMENT IS “CLASSIFICATION ON INDIAN HUMAN ORIGIN WITH OTHER’S”**

In [55]:



**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

*# Generate batches of tensor image data with real-time data augmentation.*

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

**import** pickle

​

In [2]:



gen **=** ImageDataGenerator()

batch\_size **=** 2186

​

*#Fetching Image \_\_\_\_\_\_\_\_*

​

*# Takes the path to a directory & generates batches of augmented data.*

​

flow **=** gen.flow\_from\_directory("C:\\Users\\Alwaz\Desktop\\Image dataset\\", color\_mode**=**"rgb", target\_size**=**(200, 200), batch\_size**=**batch\_size)

out **=** flow.next()

Found 2740 images belonging to 3 classes.

In [3]:



flow.class\_indices

Out[3]:

{'Foreigners image': 0, 'indian image': 1, 'non human images': 2}

In [4]:



X,y **=** out

print(X.shape)

print(y.shape)

X **=** X.reshape(X.shape[0],**-**1)

y **=**out[1].argmax(axis**=**1)

*# The numpy.argmax() function returns indices of the max element of the array in a particular axis.*

(2186, 200, 200, 3)

(2186, 3)

In [5]:



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

​

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.33, random\_state**=**42)

​

*# Reshaping \_\_\_\_\_*

​

X\_train**=**X\_train.reshape(X\_train.shape[0],**-**1)

X\_test**=**X\_test.reshape(X\_test.shape[0],**-**1)

​

print("X\_train shape :",X\_train.shape)

print("y\_train shape :",y\_train.shape)

print("X\_test shape :",X\_test.shape)

print("y\_test shape :",y\_test.shape)

X\_train shape : (1464, 120000)

y\_train shape : (1464,)

X\_test shape : (722, 120000)

y\_test shape : (722,)

**Using SVM to train model**

In [6]:



*# SVC without GRIDsearch \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2*

**from** sklearn.svm **import** SVC

​

*#A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems.*

​

**from** sklearn.metrics **import** accuracy\_score

class\_svc **=** SVC(kernel**=**'linear', random\_state**=**0)

class\_svc.fit(X\_train, y\_train)

​

y\_pred\_svc **=** class\_svc.predict(X\_test)

print(str(round(accuracy\_score(y\_test,y\_pred\_svc)**\***100,2)) **+** " % For SVC without Grid Search") *# SVC*

61.91 % For SVC without Grid Search

**Using pickle to save Model**

In [ ]:



**import** pickle

*# save the model to disk*

filename **=** 'finalized\_model.sav'

pickle.dump(class\_svc, open(filename, 'wb'))

​

In [36]:



*# load the model from disk*

loaded\_model **=** pickle.load(open(filename, 'rb'))

**Renaming Files**

In [17]:



**import** os

​

*# The OS module in Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality*

*# list dir - listdir() returns a list containing the names of the entries in the directory given by path.*

​

**def** renamed\_files(path, label):

"""

Rename files so they can contain "test n",

​

Arguments:

path -- path where images are located that will be renamed

label -- the label the images will be get

​

Output:

files will be renamed in the same folder

"""

i **=** 1

​

**for** filename **in** os.listdir(path):

**try**:

f, extension **=** os.path.splitext(path**+**filename)

src **=** path**+**filename

dst **=** path**+**label**+**str(i)**+**extension

os.rename(src, dst)

i **+=** 1

​

**except** Exception **as** e:

print(e)

i **+=** 1

​

​

path **=** "C:\\Users\\Alwaz\\Desktop\\Test set\\"

label **=** "test "

​

renamed\_files(path, label)

**Loading Files in Bulk to Predict and Filter**

In [11]:



​

**import** shutil

**from** keras.preprocessing **import** image

​

​

*# LOADING image to Filter \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

​

train\_image **=** []

​

**for** i **in** range(1,50):

**try**:

img **=** image.load\_img('{r}{u}{a}.jpg'.format(u**=**label,r**=**path,a**=**i), target\_size**=**(200,200),color\_mode **=** "rgb",interpolation**=**'nearest')

img **=** image.img\_to\_array(img) *# Converts a PIL Image instance to a Numpy array*

train\_image.append(img)

**except** Exception **as** e:

print(e)

**continue**

r **=** np.array(img)

r**=**r.reshape(1,**-**1)

*# Moving files from 'test set' to 'Filtered images' folder*

**if** loaded\_model.predict(r) **==** 1: *# 1 is for Indian*

original **=** r'C:\\Users\\Alwaz\\Desktop\\Test set\\test {a}.jpg'.format(a**=**i)

target **=** r'C:\\Users\\Alwaz\\Desktop\\Filtered Image\\test {a}.jpg'.format(a**=**i)

shutil.copyfile(original, target)

​

print("File is Filtered and moved to C:\\Users\\Alwaz\\Desktop\\Filtered Image")

File is Filtered and moved to C:\Users\Alwaz\Desktop\Filtered Image

**Loading Single Image Manually to predict**

In [54]:



**import** tqdm

**import** shutil

**from** keras.preprocessing **import** image

​

​

*# LOADING image to Filter \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

​

train\_image **=** []

​

**for** i **in** range(1,2): *# Used to test one image at a time*

**try**:

img **=** image.load\_img('C:\\Users\\Alwaz\\Desktop\\Individual Test set\\a ({a}).jpg'.format(a**=**i), target\_size**=**(200,200),color\_mode **=** "rgb",interpolation**=**'nearest')

img **=** image.img\_to\_array(img) *# Converts a PIL Image instance to a Numpy array*

train\_image.append(img)

**except** Exception **as** e:

print(e)

**continue**

r **=** np.array(img)

r**=**r.reshape(1,**-**1)

*# reading the image*

testImage **=** imgs.imread('C:\\Users\\Alwaz\\Desktop\\Individual Test set\\a ({a}).jpg'.format(a**=**i))

plt.imshow(testImage)

**if** loaded\_model.predict(r) **==** 0:

print("Its Foreigners image")

**if** loaded\_model.predict(r) **==** 1:

print("Its Indian image")

**if** loaded\_model.predict(r) **==** 2:

print("Its Non human image")

Out[] Its Non human image

