College of Engineering Guindy, Anna University

Department of Computer Science and Engineering

CS6301 Machine Learning

Project Documentation

# Age and Gender Prediction using Convolutional Neural Network

| | |
|---|---|
| Srivatsav R | 2019103066 |
| Sachin Raghul T | 2019103573 |
| Sanjeev K M | 2019103576 |

## TABLE OF CONTENTS

## Objective

The main aim is to detect age and gender through the given data set. We will use simple python and Keras methods for detecting age and gender. The data set can be downloaded from UTKFace.

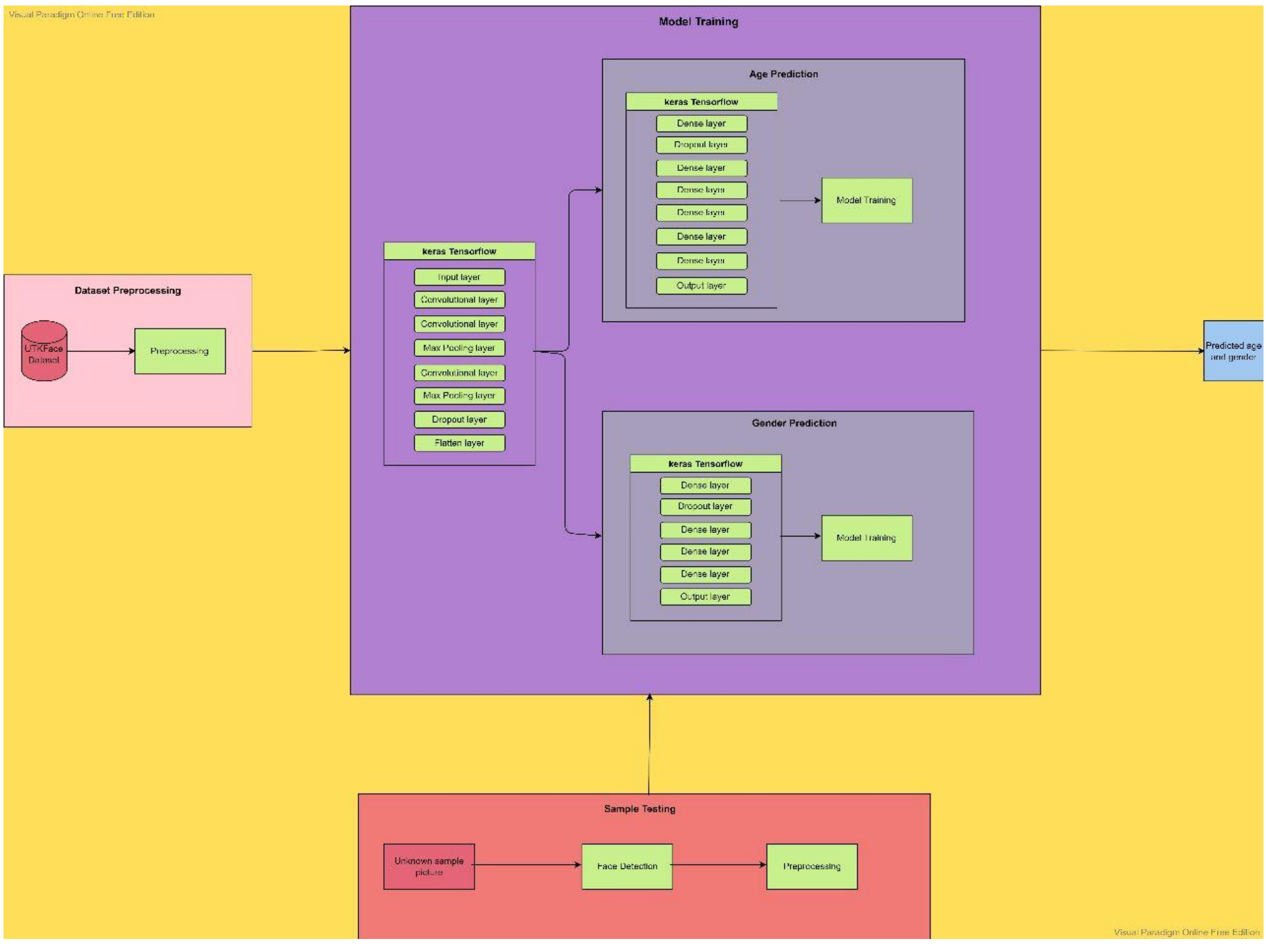
## Introduction

There are different sorts of procedures required for, just as the expulsion of the issue. In a Facial identification strategy: The articulations that the faces contain hold a great deal of data. At whatever point the individual associates with the other individual, there is an association of a ton of ideas.

The evolving of ideas helps in figuring certain boundaries. Age assessment is a multi-class issue in which the years; are categorized into classes. Individuals of various ages have various facials, so it is hard to assemble the pictures.

To identify the age and gender of several faces' procedures, are followed by several methods. From the neural network, features are taken by the convolution network. In light of the prepared models, the image is processed into one of the age classes and gender class.

## SYSTEM ARCHITECTURE:

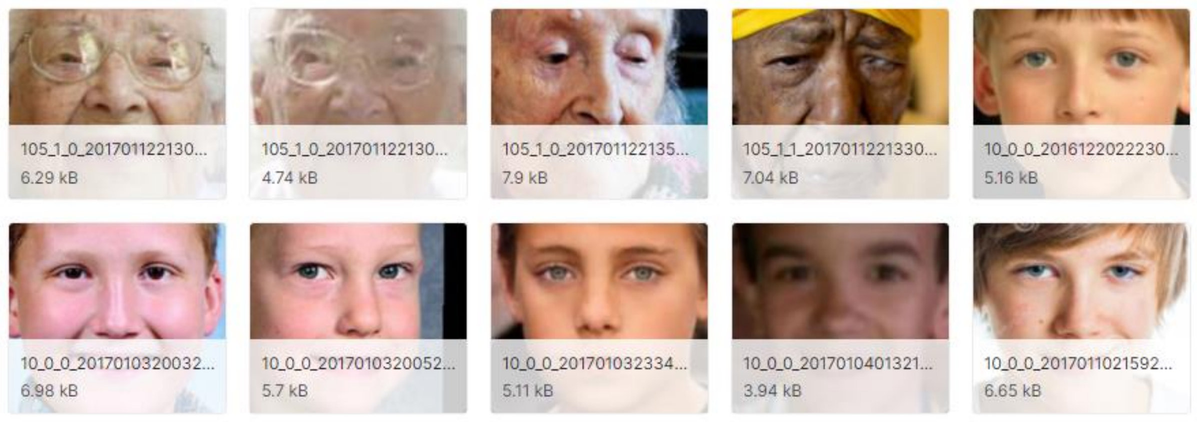


# MODULE 1:

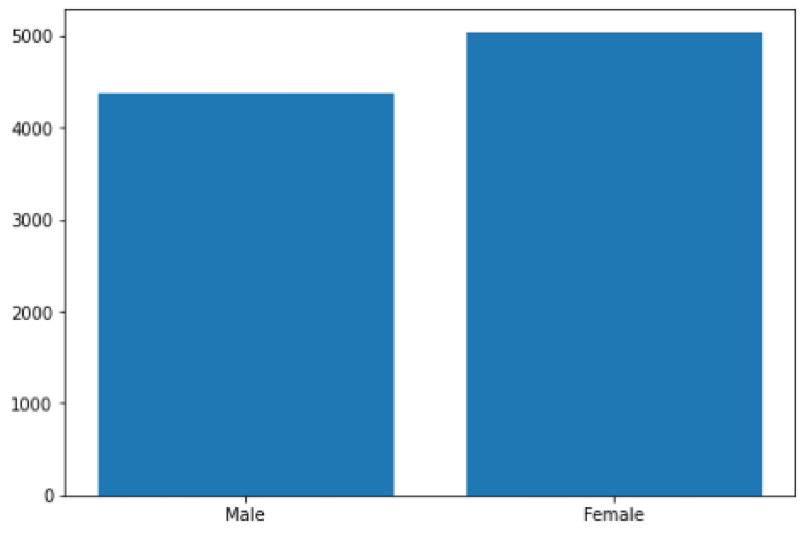
## The Dataset

UTK Dataset comprises age, gender, images, and pixels in .csv format. Age and gender detection according to the images have been researched for a long time. Different methodologies have been assumed control over the years to handle this issue. Presently we start with the assignment of recognizing age and gender utilizing the Python programming language.

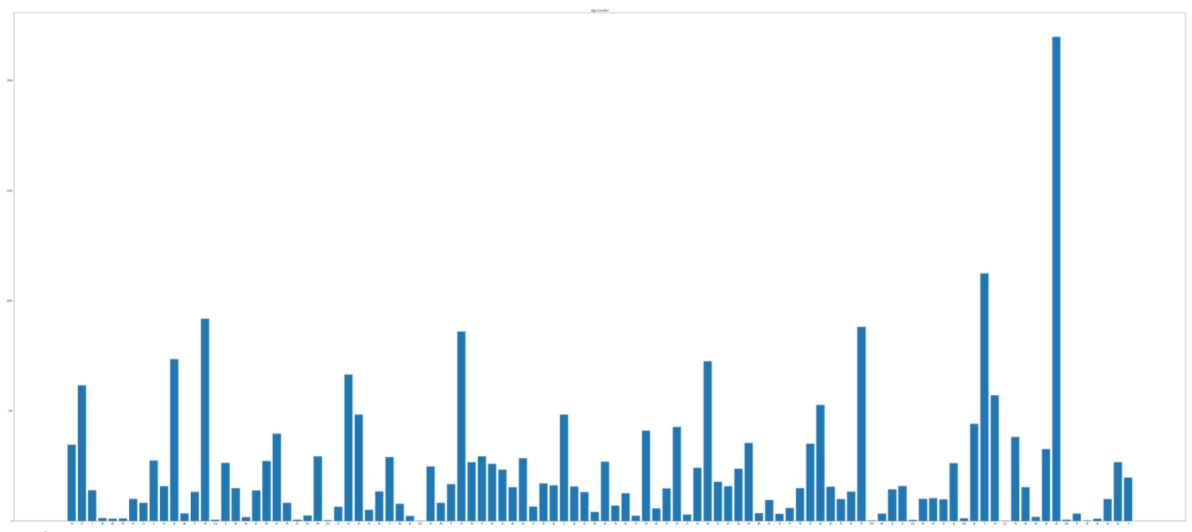
Keras is the interface for the TensorFlow library. Use Keras on the off chance that you need a profound learning library that allows simple and quick prototyping (through ease of use, seclusion, and extensibility). Support both convolutional networks and repetitive organizations, just as blends of the two. Run flawlessly on CPU and GPU.



It contains 23k+ face data from various age, races and gender.



Dataset Distribution (ACROSS Gender)



```
Max value: 99
Min value: 1
```

Dataset Distribution (ACROSS AGES)

## MODULE 2:

### 1. Face Detection:

The first task that we perform is detecting faces in the image(photograph) or video stream. Now, we have to detect the face image from the source by haar cascading module and then from the extracted coordinates/location of the face, we extract this face for further processing .

### 2. Feature Extraction:

Now we have cropped out the face from the image, so we extract specific features from it. Here we are going to see how to use face embeddings to extract these features of the face. As we know a neural network takes an image of the face of the person as input and outputs a vector that represents the most important features of a face! In machine learning, this vector is nothing but called *embedding* and hence we call this vector ***face embedding.***

When we train the neural network, the network learns to output *similar vectors* for faces that look similar. Let us consider an example, if I have multiple images of faces within different timelapse, it's obvious that some features may change but not too much. So in this problem, the vectors associated with the faces are similar or we can say they are very close in the vector space.

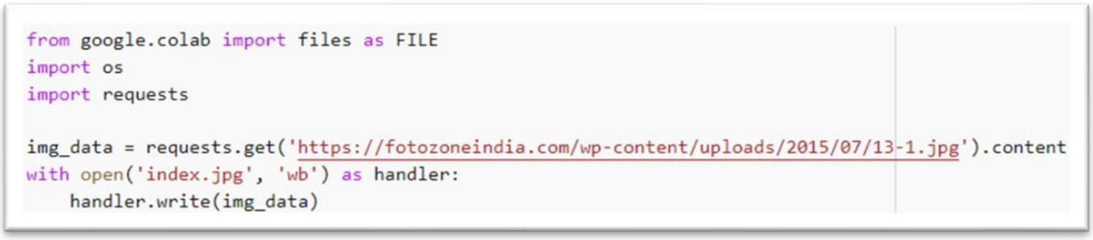
Up to this point, we came to know how this network works, let us see how to use this network on our own data. Here we pass all the images in our data to this pre-trained network to get the respective embeddings and save these embeddings in a file for the next step.

***OpenCV***

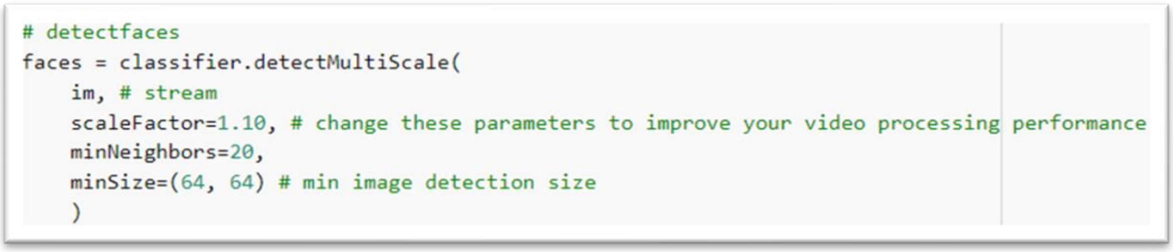
Various packages are available to perform machine learning, deep learning, and computer vision problems. So far, computer vision is the best module for such complex problems. ***OpenCV*** is an open-source library. It is supported by different programming languages such as R, Python, etc. It runs probably on most platforms such as Windows, Linux, and macOS.

IMPLEMETATION:

- Download the source  image that is to be tested against the Age and Gender Detection model from the url using python request module
- Using OpenCV detect, detect the cropped faces from the source image loaded and store them into the desired directory



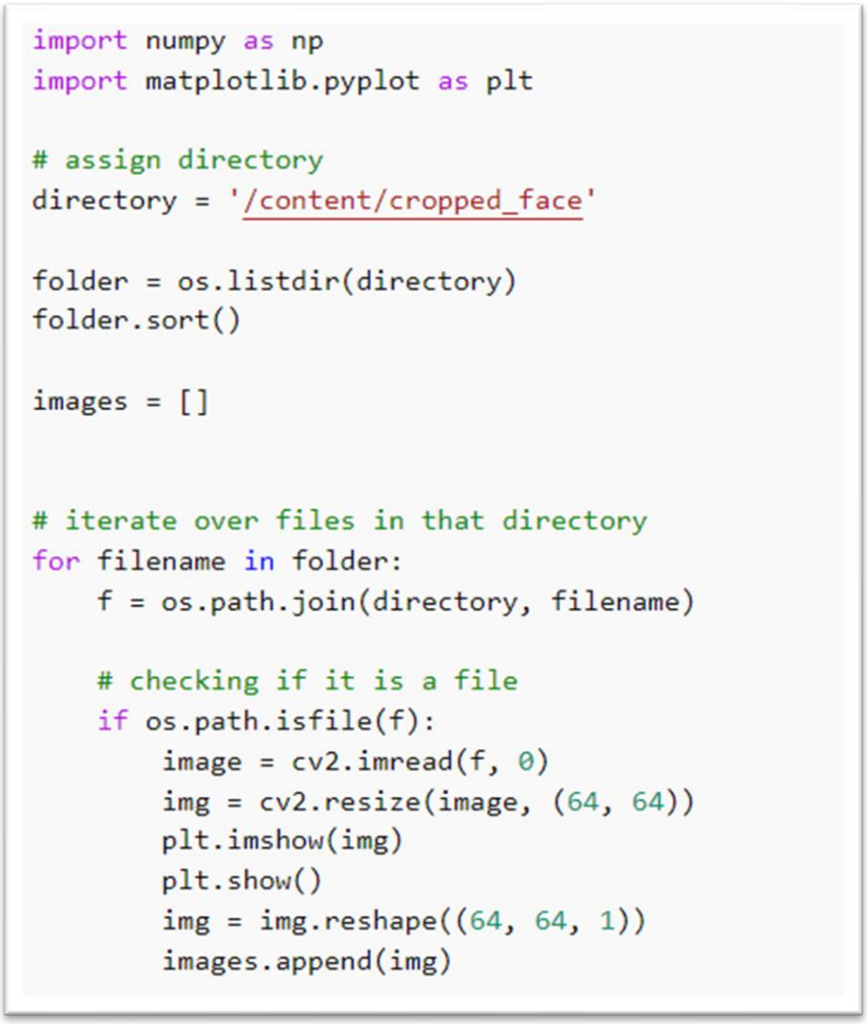
```
from google.colab import files as FILE
import os
import requests

img_data = requests.get('https://fotozoneindia.com/wp-content/uploads/2015/07/13-1.jpg').content
with open('index.jpg', 'wb') as handler:
    handler.write(img_data)
```



```
# detectfaces
faces = classifier.detectMultiScale(
    im, # stream
    scaleFactor=1.10, # change these parameters to improve your video processing performance
    minNeighbors=20,
    minSize=(64, 64) # min image detection size
    )
```

- Iterate through the cropped images stored in the directory using listdir module in the os library and do the following needed.
- Reduce and compress the extracted cropped image using resize module in OpenCV
- Convert the processed image to greyscale pixel data

- Append the image pixel values into a list each time you iterate a image in the directory present.



```
import numpy as np
import matplotlib.pyplot as plt

# assign directory
directory = '/content/cropped_face'

folder = os.listdir(directory)
folder.sort()

images = []


# iterate over files in that directory
for filename in folder:
    f = os.path.join(directory, filename)

    # checking if it is a file
    if os.path.isfile(f):
        image = cv2.imread(f, 0)
        img = cv2.resize(image, (64, 64))
        plt.imshow(img)
        plt.show()
        img = img.reshape((64, 64, 1))
        images.append(img)
```

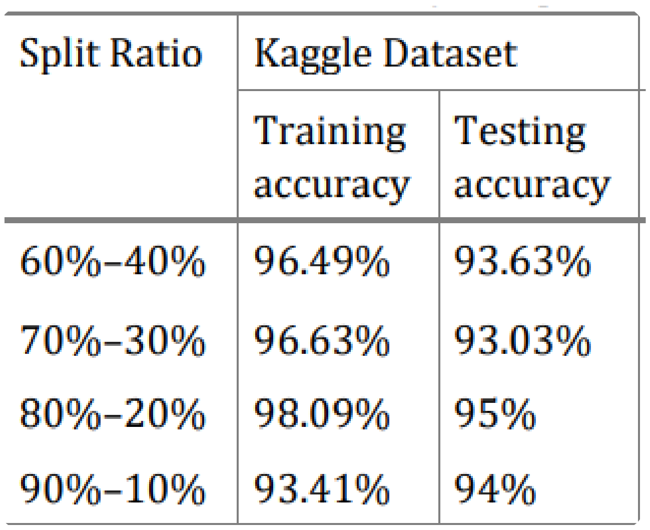
## MODULE 3:

### PREPROCESSING:

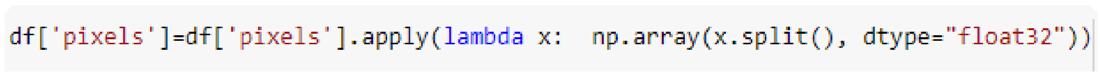
We will convert all the columns into an array by using the np.array and into dtype float. We will then split the data set into xTrain, yTrain, yTest, and xtest. In the end, we will apply the model sequential and test the predictions.

In detail, first, we read the CSV file containing five columns age, ethnicity, gender, img_name, and pixels by using pandas, read_csv function. The first five rows got by using DataFrame.head() method. We converted the column-named pixels into an array by using the NumPy library and Reshaping them into dimensions 64, 64 by using the openCv. We also converted the values in the float.

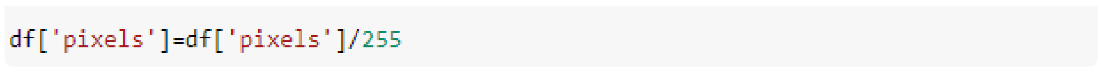
We divided the values further by 255.



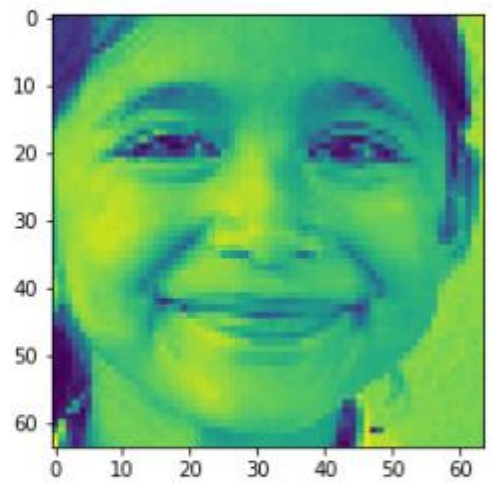
| Split Ratio | Kaggle Dataset | |
|---|---|---|
| | Training accuracy | Testing accuracy |
| 60%–40% | 96.49% | 93.63% |
| 70%–30% | 96.63% | 93.03% |
| 80%–20% | 98.09% | 95% |
| 90%–10% | 93.41% | 94% |



```
df['pixels']=df['pixels'].apply(lambda x:  np.array(x.split(), dtype="float32"))
```



```
df['pixels']=df['pixels']/255
```



```
Gender: Female & Age: 8
```

Image after reshaping

## And we group ages into 4 classes:

```python
def age_group(age):
    if age >=0 and age < 18:
        return 1
    elif age < 30:
        return 2
    elif age < 80:
        return 3
    else:
        return 4 #unknown
```

# MODULE 4:

## TRAINING THE MODEL:

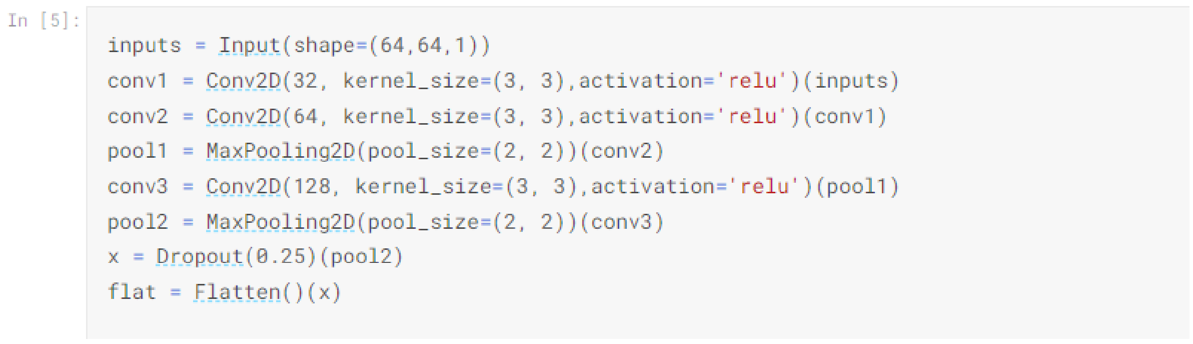
### Keras

Keras is an open-source Neural Network library. It is written in Python and is sufficiently fit to run on Theano, TensorFlow, or CNTK, developed by one of the Google engineers, Francois Chollet. It is made easy to understand, extensible, and particular for quicker experimentation with profound neural organizations.

- The Age and Gender Prediction keras model is split up into three submodels
  - Model to feed the input to other models (Input layer model)
  - Model to detect the age (Age Model)
  - Model to detect the gender (Gender Model)

### INPUT LAYER MODEL:

- Input model contains of an input layer, 3 convolutional layer, 2 MaxPooling layer, a dropout and the flatten layer.
- The images are of size 64 x 64. You convert the image matrix to an array, rescale it between 0 and 1, reshape it so that it's of size 64 x 64 x 1, and feed this as an input to the network.
- We are using two convolutional 2D layers:
- The first layer will have 32-3 x 3 filters,
- The second layer will have 64-3 x 3 filters and
- In addition, there are two max-pooling layers each of size 2 x 2.
- Also the model contains Flatten and Dropout layers in the sequential module such that the model is best trained for what is required.

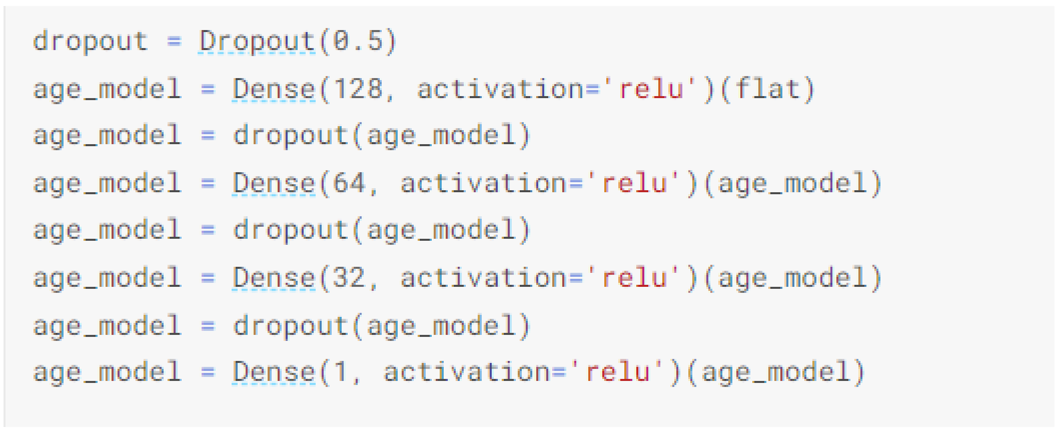


```
In [5]:
inputs = Input(shape=(64,64,1))
conv1 = Conv2D(32, kernel_size=(3, 3),activation='relu')(inputs)
conv2 = Conv2D(64, kernel_size=(3, 3),activation='relu')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(128, kernel_size=(3, 3),activation='relu')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv3)
x = Dropout(0.25)(pool2)
flat = Flatten()(x)
```

Input layer model

## AGE MODEL:

- Age model has an sequence of dropout and Dense layers and reducing the neural layers in each step.
- We use relu activation since age_detection involves multi class classification



```
dropout = Dropout(0.5)
age_model = Dense(128, activation='relu')(flat)
age_model = dropout(age_model)
age_model = Dense(64, activation='relu')(age_model)
age_model = dropout(age_model)
age_model = Dense(32, activation='relu')(age_model)
age_model = dropout(age_model)
age_model = Dense(1, activation='relu')(age_model)
```

Age model

## GENDER MODEL:

- Gender model has an sequence of dropout and Dense layers and reducing the neural layers in each step.
- We use sigmoid activation since gender_detection involves only two single class classification (M & F)

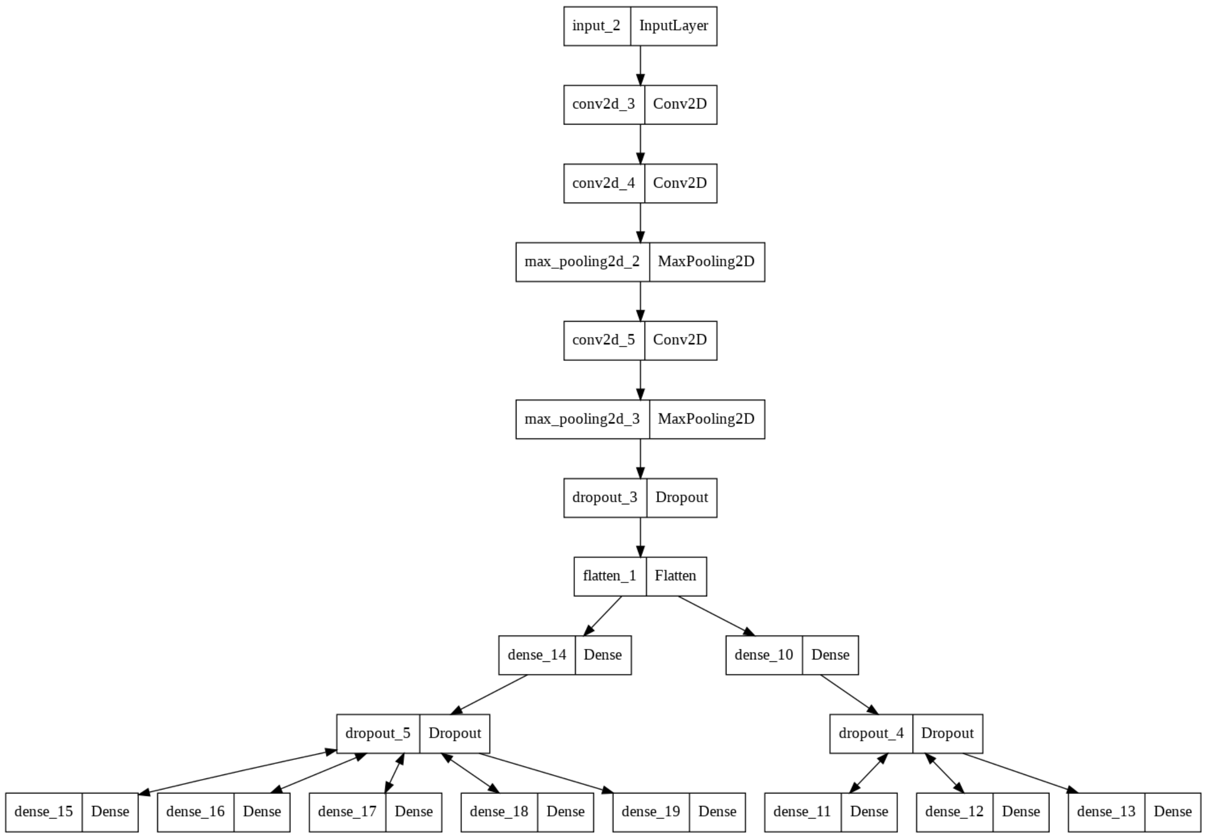
```
dropout = Dropout(0.5)
gender_model = Dense(128, activation='relu')(flat)
gender_model = dropout(gender_model)
gender_model = Dense(64, activation='relu')(gender_model)
gender_model = dropout(gender_model)
gender_model = Dense(32, activation='relu')(gender_model)
gender_model = dropout(gender_model)
gender_model = Dense(16, activation='relu')(gender_model)
gender_model = dropout(gender_model)
gender_model = Dense(8, activation='relu')(gender_model)
gender_model = dropout(gender_model)
gender_model = Dense(1, activation='sigmoid')(gender_model)
```

Gender model

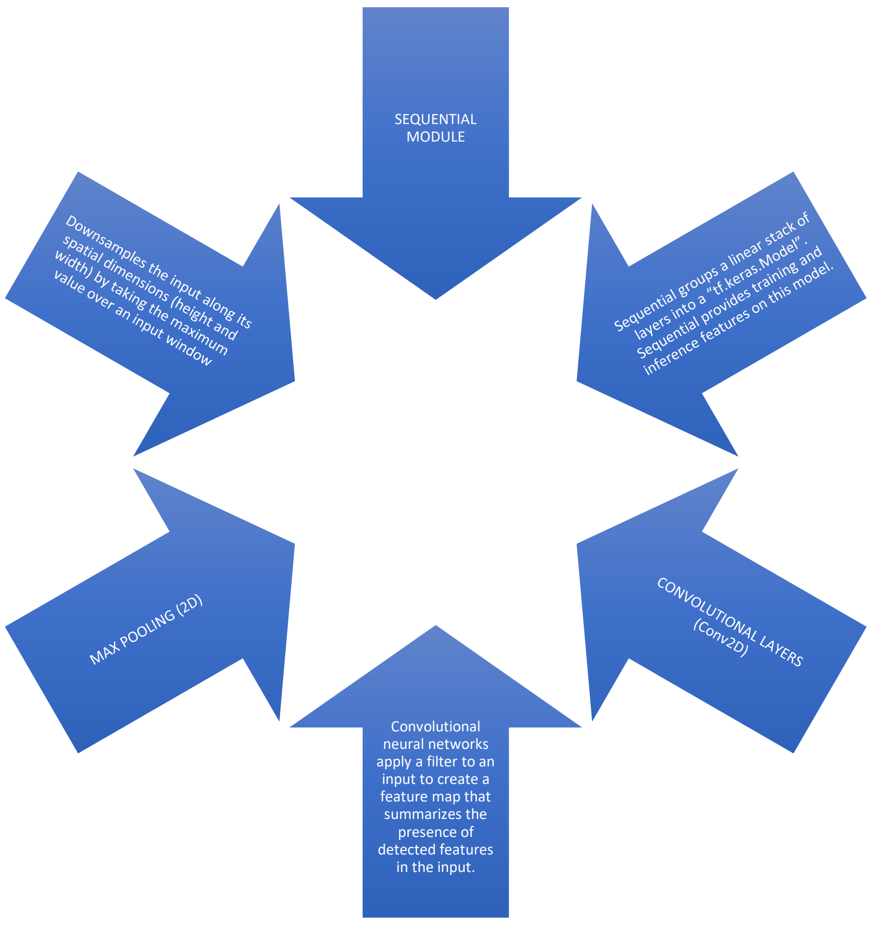
Finally all three models are made to work in unison under a single model

```
model = Model(inputs=inputs, outputs=[age_model,gender_model])
model.compile(optimizer = 'adam', loss =['mse','binary_crossentropy'],metrics=['accuracy'])
```

## MODEL SUMMARY



## MODULES EXLAINED:


SEQUENTIAL MODULE
Sequential groups a linear stack of layers into a "tf.keras.Model". Sequential provides training and inference features on this model.
CONVOLUTIONAL LAYERS (Conv2D)
Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input.
MAX POOLING (2D)
Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window

CONVOLUTIONAL LAYERS (Conv2D)

- Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input.

MAX POOLING (2D)

- Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window

FLATTEN LAYER

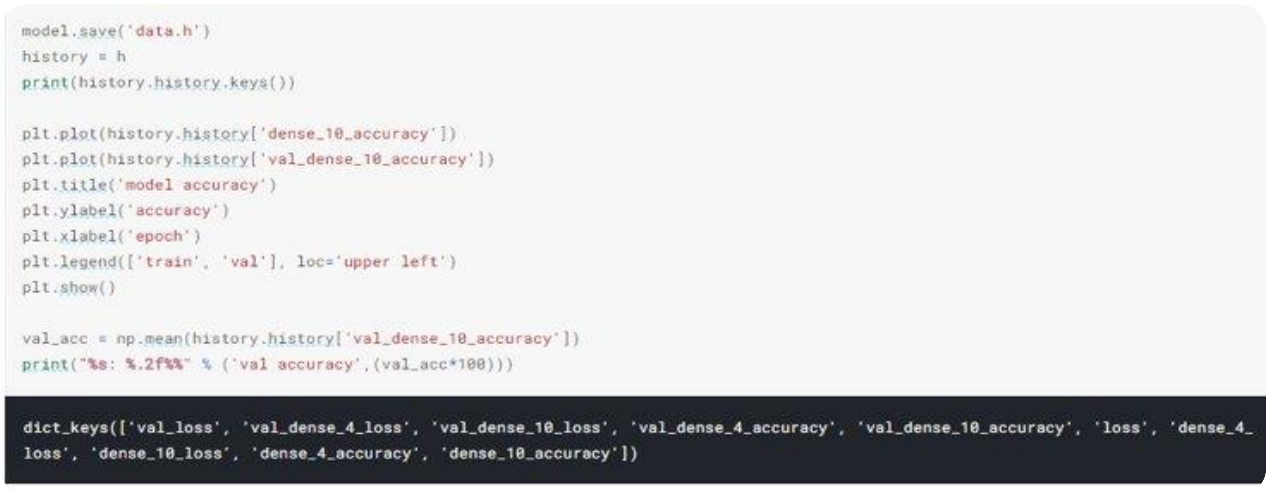
- A flatten operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension.

## DROPOUT LAYER

- The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

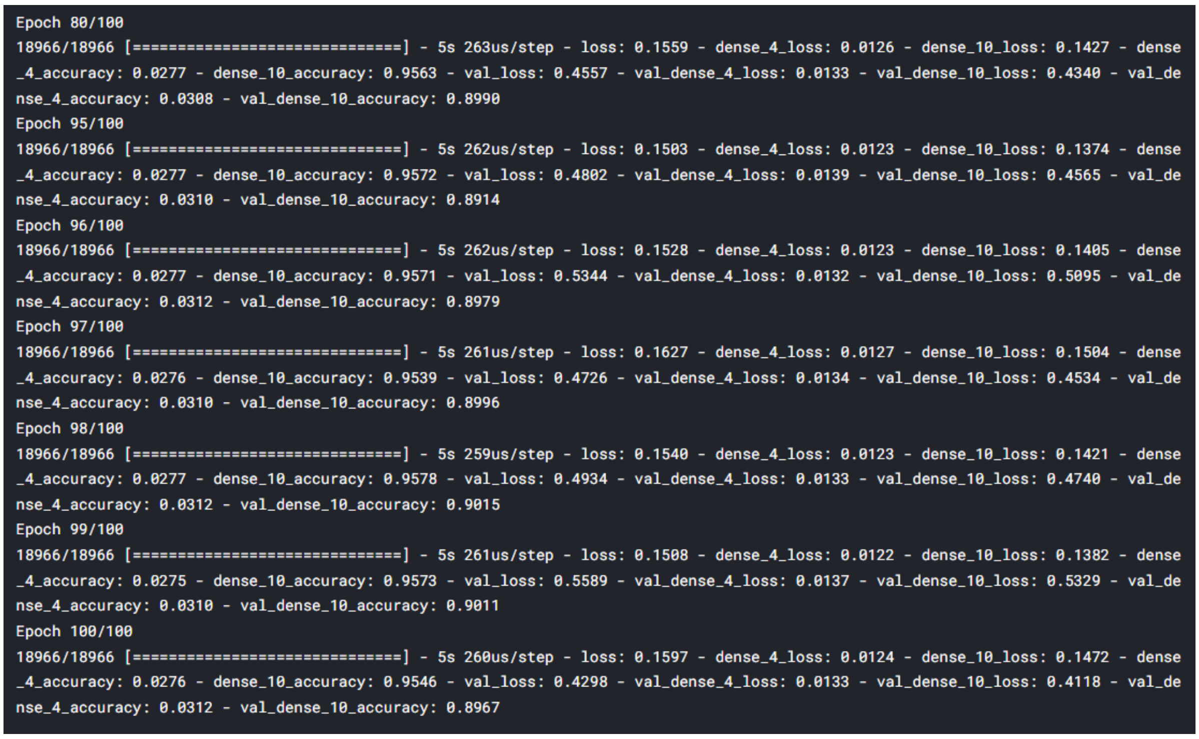
## EPOCHES AND BATCH SIZE

- The model is trained using batch size of 128. Batch sizes of 64, 128, 256 etc.. are preferable. But in our case, batch size of 128 gives better result than 64 or 256.
- Batch size of 64 is biased and batch size of 256 was highly variant in our case, whereas 64 was proved to be the best.
- It contributes massively  in determining the learning parameters and also it affects the prediction accuracy. So, this model is trained using 100 epochs so that the model either overfits nor underfits.



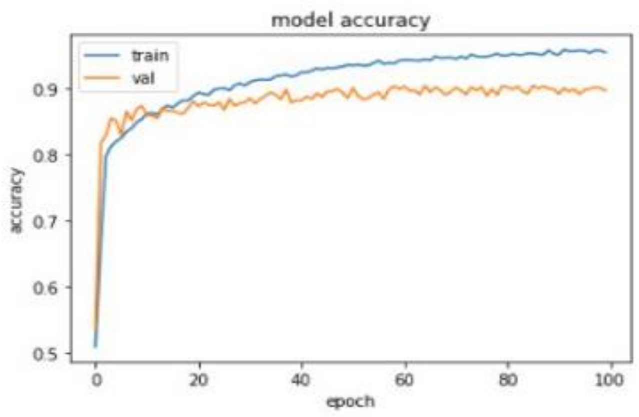
```
model.save('data.h')
history = h
print(history.history.keys())

plt.plot(history.history['dense_10_accuracy'])
plt.plot(history.history['val_dense_10_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

val_acc = np.mean(history.history['val_dense_10_accuracy'])
print("%s: %.2f%%" % ('val accuracy',(val_acc*100)))
```

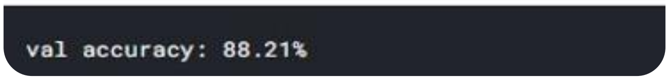
```
dict_keys(['val_loss', 'val_dense_4_loss', 'val_dense_10_loss', 'val_dense_4_accuracy', 'val_dense_10_accuracy', 'loss', 'dense_4_
loss', 'dense_10_loss', 'dense_4_accuracy', 'dense_10_accuracy'])
```



```
Epoch 80/100
18966/18966 [==============================] - 5s 263us/step - loss: 0.1559 - dense_4_loss: 0.0126 - dense_10_loss: 0.1427 - dense
_4_accuracy: 0.0277 - dense_10_accuracy: 0.9563 - val_loss: 0.4557 - val_dense_4_loss: 0.0133 - val_dense_10_loss: 0.4340 - val_de
nse_4_accuracy: 0.0308 - val_dense_10_accuracy: 0.8990
Epoch 95/100
18966/18966 [==============================] - 5s 262us/step - loss: 0.1503 - dense_4_loss: 0.0123 - dense_10_loss: 0.1374 - dense
_4_accuracy: 0.0277 - dense_10_accuracy: 0.9572 - val_loss: 0.4802 - val_dense_4_loss: 0.0139 - val_dense_10_loss: 0.4565 - val_de
nse_4_accuracy: 0.0310 - val_dense_10_accuracy: 0.8914
Epoch 96/100
18966/18966 [==============================] - 5s 262us/step - loss: 0.1528 - dense_4_loss: 0.0123 - dense_10_loss: 0.1405 - dense
_4_accuracy: 0.0277 - dense_10_accuracy: 0.9571 - val_loss: 0.5344 - val_dense_4_loss: 0.0132 - val_dense_10_loss: 0.5095 - val_de
nse_4_accuracy: 0.0312 - val_dense_10_accuracy: 0.8979
Epoch 97/100
18966/18966 [==============================] - 5s 261us/step - loss: 0.1627 - dense_4_loss: 0.0127 - dense_10_loss: 0.1504 - dense
_4_accuracy: 0.0276 - dense_10_accuracy: 0.9539 - val_loss: 0.4726 - val_dense_4_loss: 0.0134 - val_dense_10_loss: 0.4534 - val_de
nse_4_accuracy: 0.0310 - val_dense_10_accuracy: 0.8996
Epoch 98/100
18966/18966 [==============================] - 5s 259us/step - loss: 0.1540 - dense_4_loss: 0.0123 - dense_10_loss: 0.1421 - dense
_4_accuracy: 0.0277 - dense_10_accuracy: 0.9578 - val_loss: 0.4934 - val_dense_4_loss: 0.0133 - val_dense_10_loss: 0.4740 - val_de
nse_4_accuracy: 0.0312 - val_dense_10_accuracy: 0.9015
Epoch 99/100
18966/18966 [==============================] - 5s 261us/step - loss: 0.1508 - dense_4_loss: 0.0122 - dense_10_loss: 0.1382 - dense
_4_accuracy: 0.0275 - dense_10_accuracy: 0.9573 - val_loss: 0.5589 - val_dense_4_loss: 0.0137 - val_dense_10_loss: 0.5329 - val_de
nse_4_accuracy: 0.0310 - val_dense_10_accuracy: 0.9011
Epoch 100/100
18966/18966 [==============================] - 5s 260us/step - loss: 0.1597 - dense_4_loss: 0.0124 - dense_10_loss: 0.1472 - dense
_4_accuracy: 0.0276 - dense_10_accuracy: 0.9546 - val_loss: 0.4298 - val_dense_4_loss: 0.0133 - val_dense_10_loss: 0.4118 - val_de
nse_4_accuracy: 0.0312 - val_dense_10_accuracy: 0.8967
```

Model training



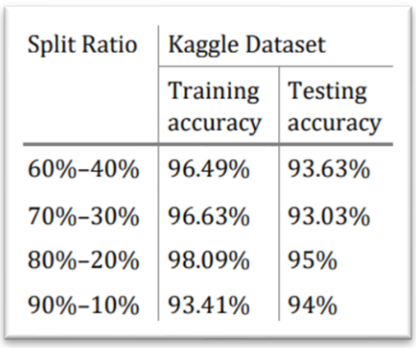


```
val accuracy: 88.21%
```

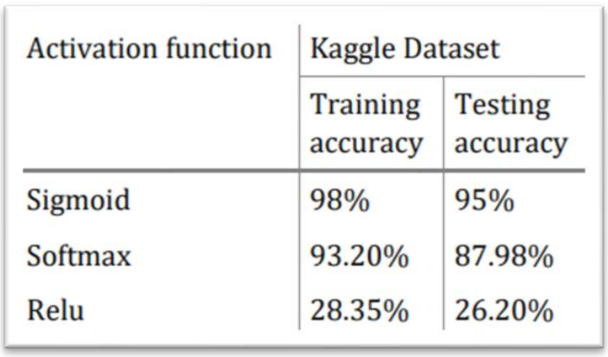
**Model Accuracy: 88.21%**

## RESEARCHS INVOLVED:

- First, on splitting the dataset for training and testing, a variety of combinations were tested and among those, 0.8 : 0.2 (train : test) was the best.
- Next on choosing the number of convolutional layers, having two conv 2D and doing maxpooling resulted good before training the model for age and gender.
- And, on selection of batch sizes and epoch, it was batch size of 128 and 100 epoch which was the one that neither extremely overfit nor underfit.
- Sigmoid function gives higher accuracy, but on considering only doing sigmoid will overfit the model, first the model is trained by relu and then finally it was trained using sigmoid activation function.



| Split Ratio | Kaggle Dataset | |
|---|---|---|
| | Training accuracy | Testing accuracy |
| 60%-40% | 96.49% | 93.63% |
| 70%-30% | 96.63% | 93.03% |
| 80%-20% | 98.09% | 95% |
| 90%-10% | 93.41% | 94% |

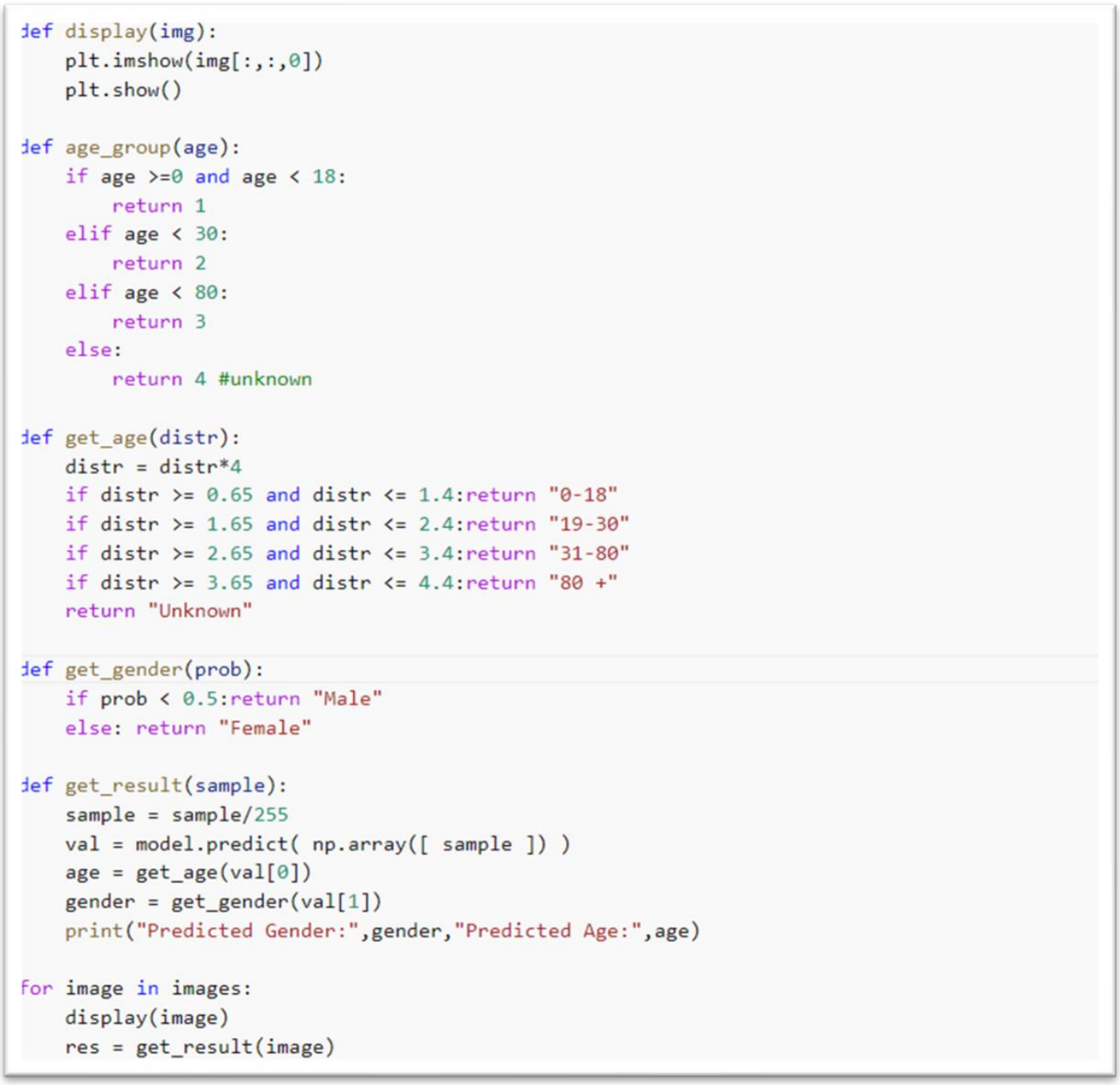


| Activation function | Kaggle Dataset | |
|---|---|---|
| | Training accuracy | Testing accuracy |
| Sigmoid | 98% | 95% |
| Softmax | 93.20% | 87.98% |
| Relu | 28.35% | 26.20% |

## MODULE 5:

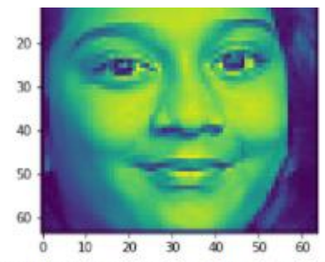
## AGE AND GENDER PREDCTION:

- After preprocessing the raw data (images), the processed data ie. The feature vector is fit into our trained model. So, that the age and gender is estimated accordingly.
- We classify the age based on the distortion vallue provided by the trained model.

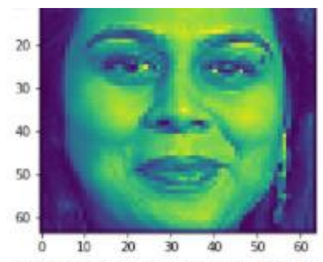


```
def display(img):
    plt.imshow(img[:,:,0])
    plt.show()

def age_group(age):
    if age >=0 and age < 18:
        return 1
    elif age < 30:
        return 2
    elif age < 80:
        return 3
    else:
        return 4 #unknown

def get_age(distr):
    distr = distr*4
    if distr >= 0.65 and distr <= 1.4:return "0-18"
    if distr >= 1.65 and distr <= 2.4:return "19-30"
    if distr >= 2.65 and distr <= 3.4:return "31-80"
    if distr >= 3.65 and distr <= 4.4:return "80 +"
    return "Unknown"

def get_gender(prob):
    if prob < 0.5:return "Male"
    else: return "Female"

def get_result(sample):
    sample = sample/255
    val = model.predict( np.array([ sample ]) )
    age = get_age(val[0])
    gender = get_gender(val[1])
    print("Predicted Gender:",gender,"Predicted Age:",age)

for image in images:
    display(image)
    res = get_result(image)
```

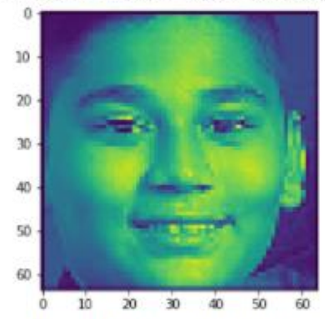
- Final prediction of age and gender of the detected faces from the given source image.



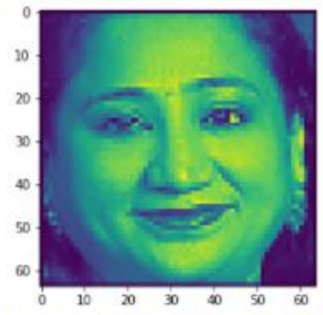
```
Predicted Gender: Female Predicted Age: 0-18
```



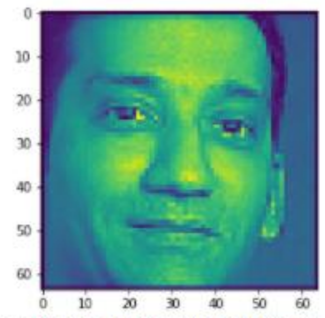
```
Predicted Gender: Female Predicted Age: 19-30
```



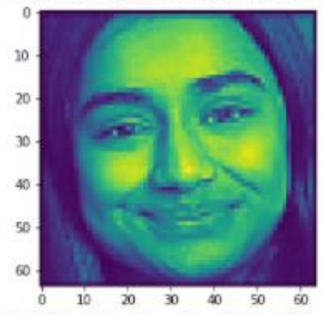
```
Predicted Gender: Male Predicted Age: 0-18
```



```
Predicted Gender: Female Predicted Age: 31-80
```



```
Predicted Gender: Male Predicted Age: 19-30
```

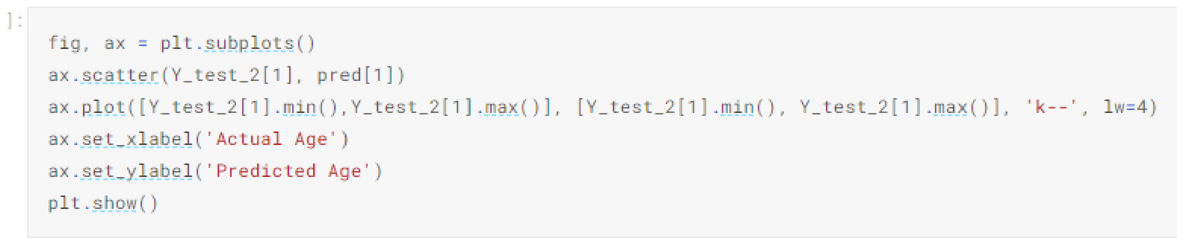


```
Predicted Gender: Female Predicted Age: 19-30
```

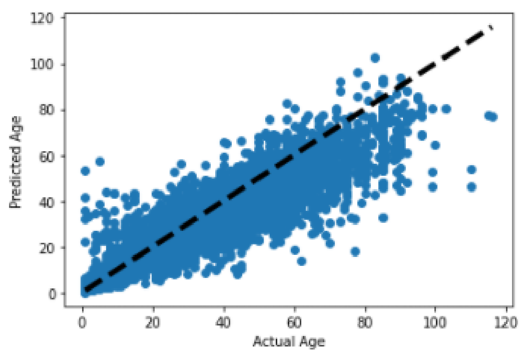
## PERFORMANCES:

### PERFORMANCE ON AGE:

Actual age vs Predicted age Distortion



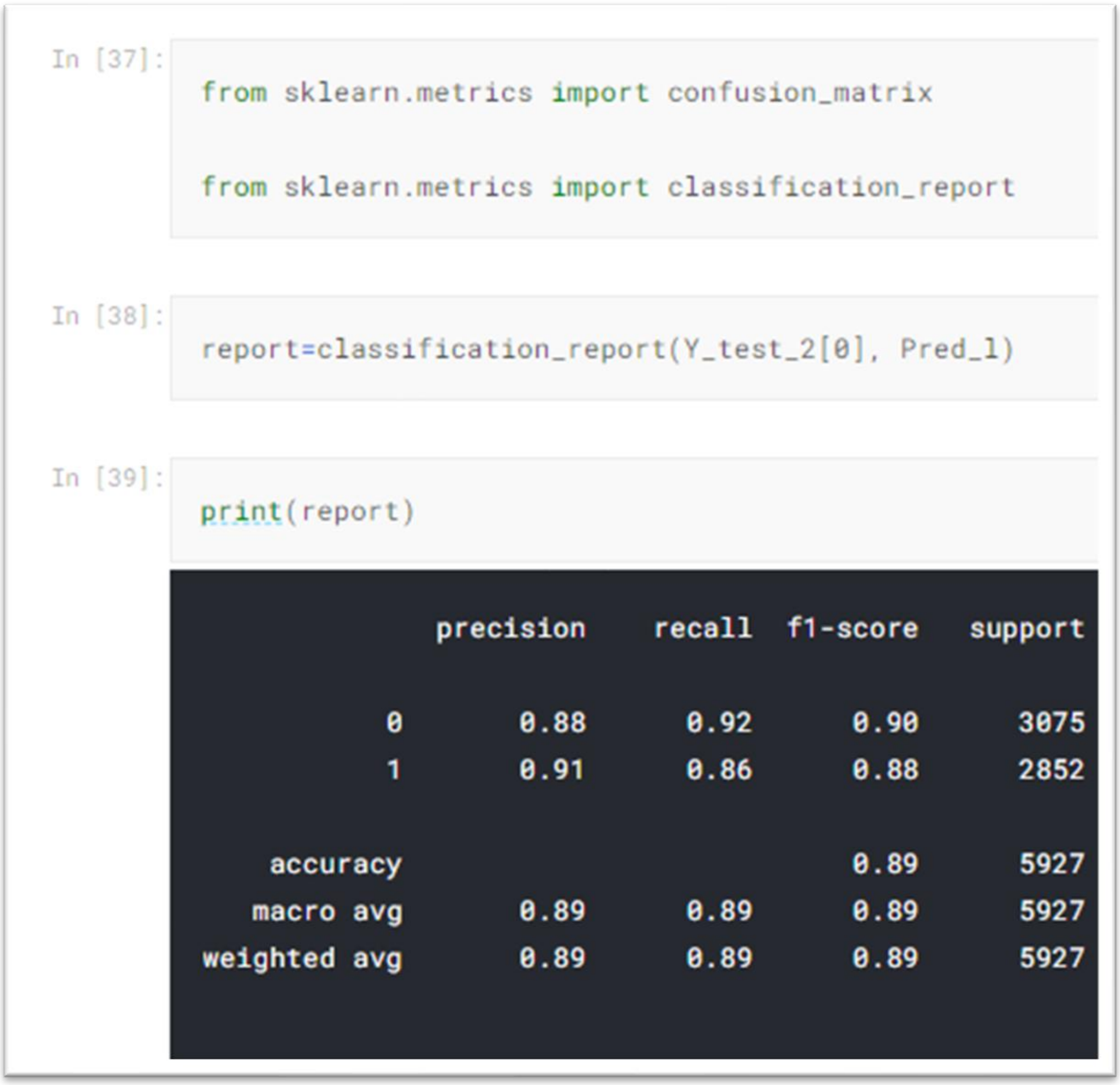
```
]:
fig, ax = plt.subplots()
ax.scatter(Y_test_2[1], pred[1])
ax.plot([Y_test_2[1].min(),Y_test_2[1].max()], [Y_test_2[1].min(), Y_test_2[1].max()], 'k--', lw=4)
ax.set_xlabel('Actual Age')
ax.set_ylabel('Predicted Age')
plt.show()
```



Actual age vs Predicted age Distortion

## PERFORMANCE METRICS ON GENDER:  :

- The trained model has good performance in test dataset.
- It has high precision recall and f1-score, thus model generalizes the data well



```
In [37]: from sklearn.metrics import confusion_matrix

         from sklearn.metrics import classification_report
```

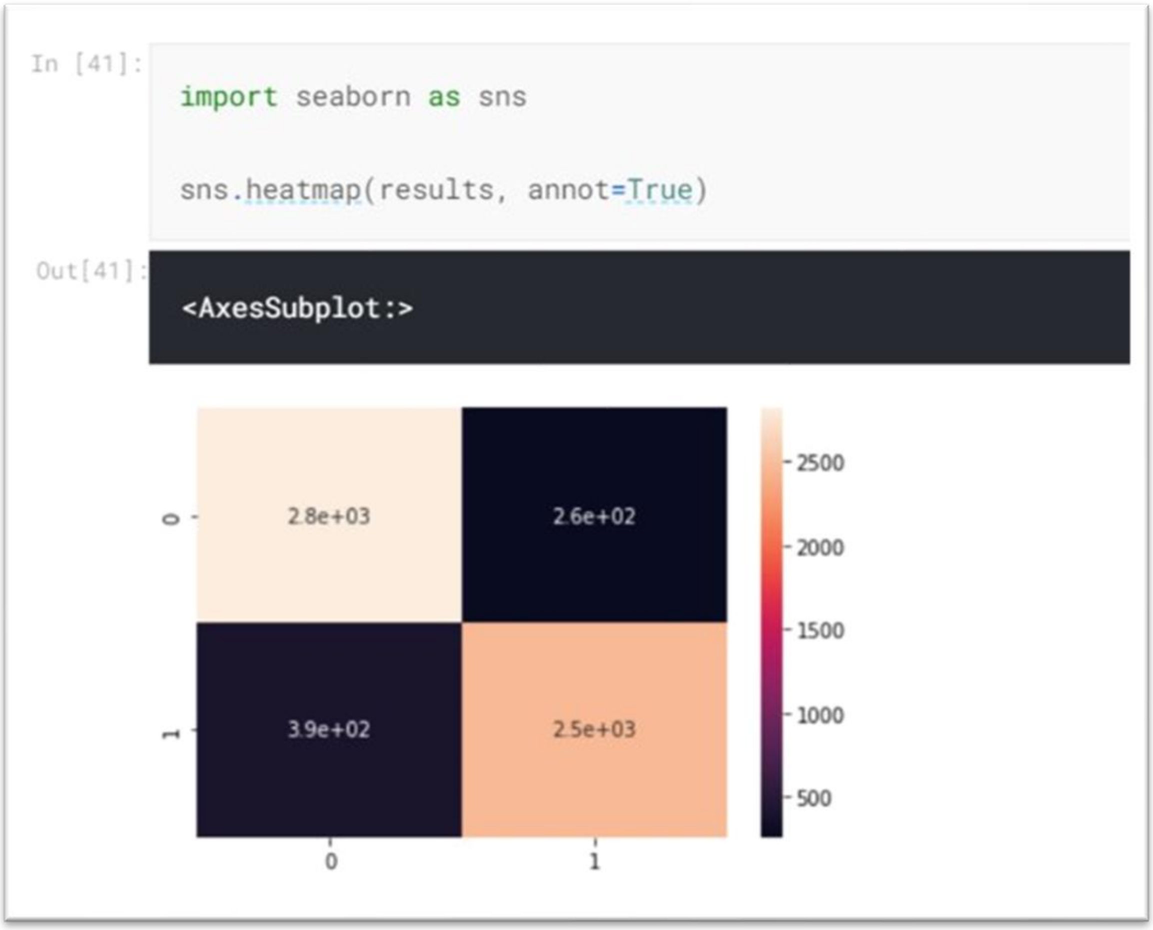
```
In [38]: report=classification_report(Y_test_2[0], Pred_l)
```

```
In [39]: print(report)
```

```
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      3075
           1       0.91      0.86      0.88      2852

    accuracy                           0.89      5927
   macro avg       0.89      0.89      0.89      5927
weighted avg       0.89      0.89      0.89      5927
```

**Classification Report**

**Confusion matrix**



```
In [41]:
import seaborn as sns

sns.heatmap(results, annot=True)
```

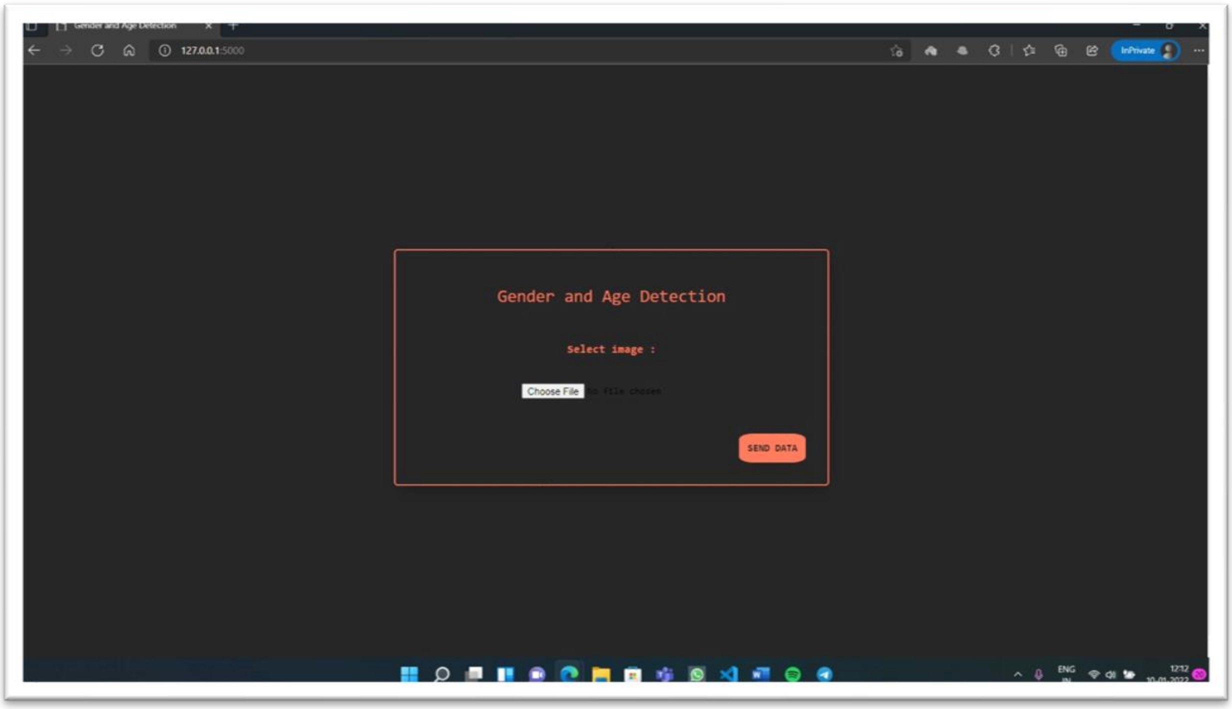
```
Out[41]:
<AxesSubplot:>
```

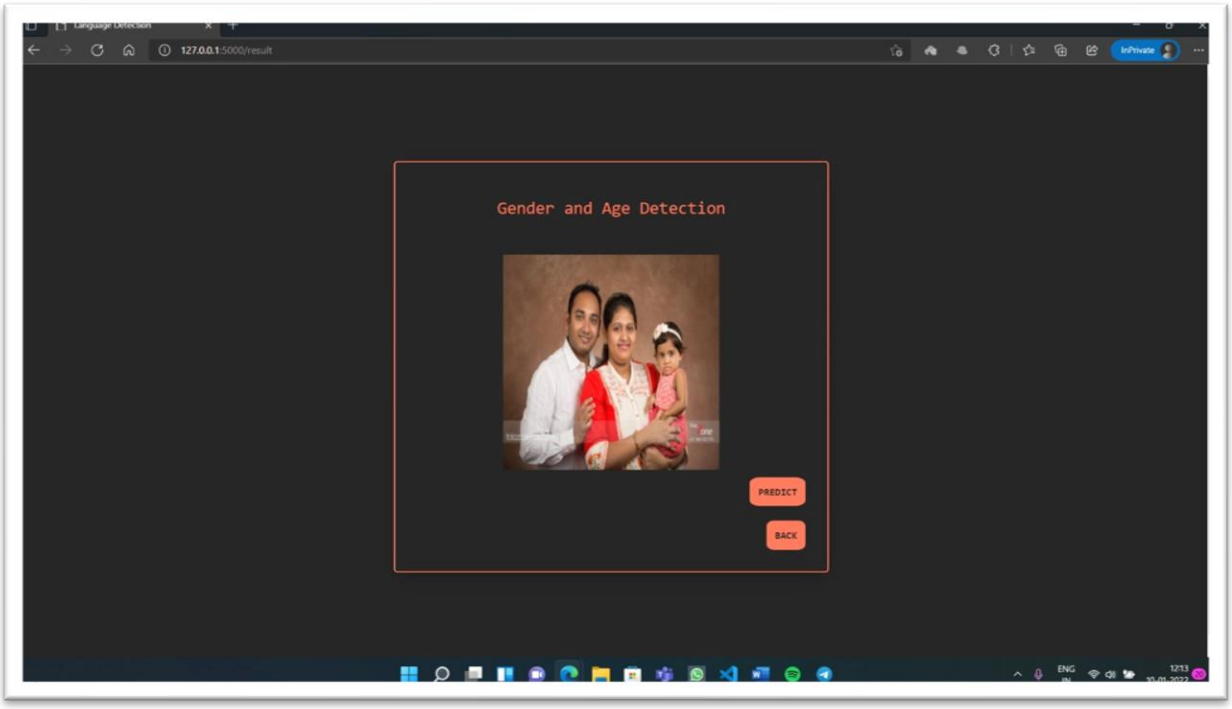
## Conclusion

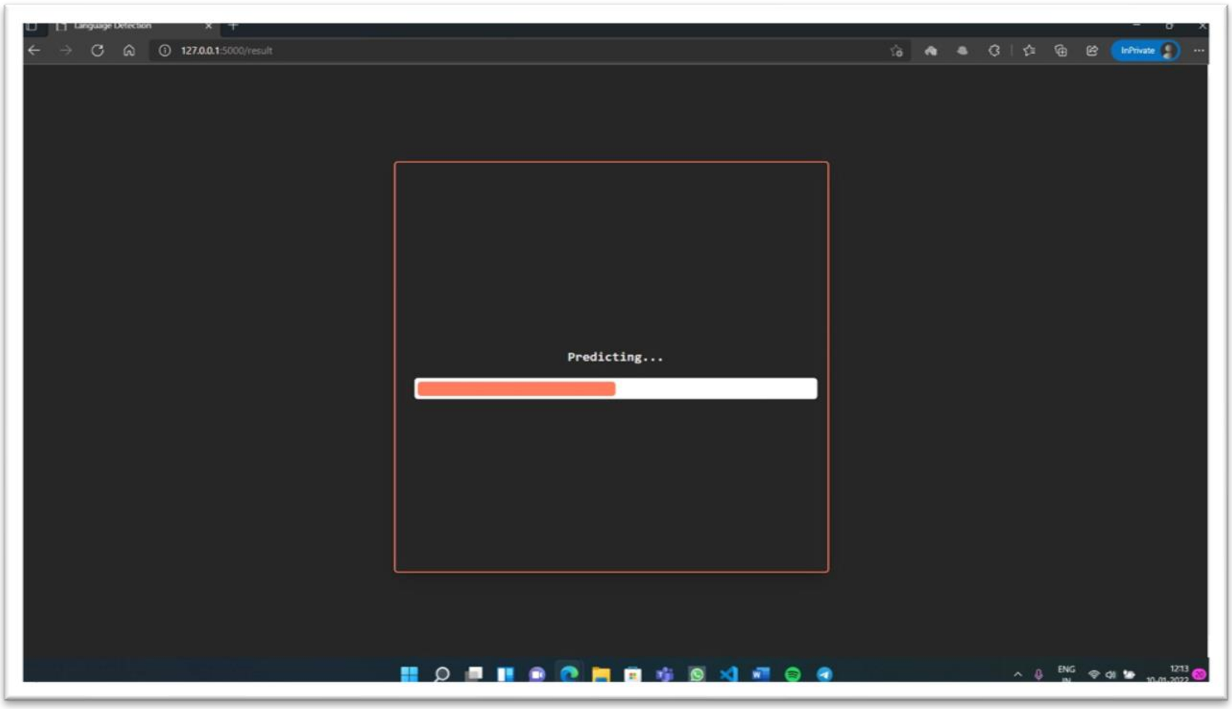
We tackled the classification of age group and gender of unfiltered real-world face images. We posed the task as a multiclass classification problem and, as such, train the model with a classification-based loss function as training targets. The image pre-processing algorithm, handles some of the variability observed in typical unfiltered real-world faces, and this confirms the model applicability for age group and gender classification in-the-wild. Finally, we investigate the classification accuracy for age and gender; our proposed method achieves the state-of-the-art performance, in both age group and gender classification. For future works, we will consider a deeper CNN architecture and a more robust image processing algorithm for exact age estimation. Also, the apparent age estimation of human's face will be interesting research to investigate in the future.

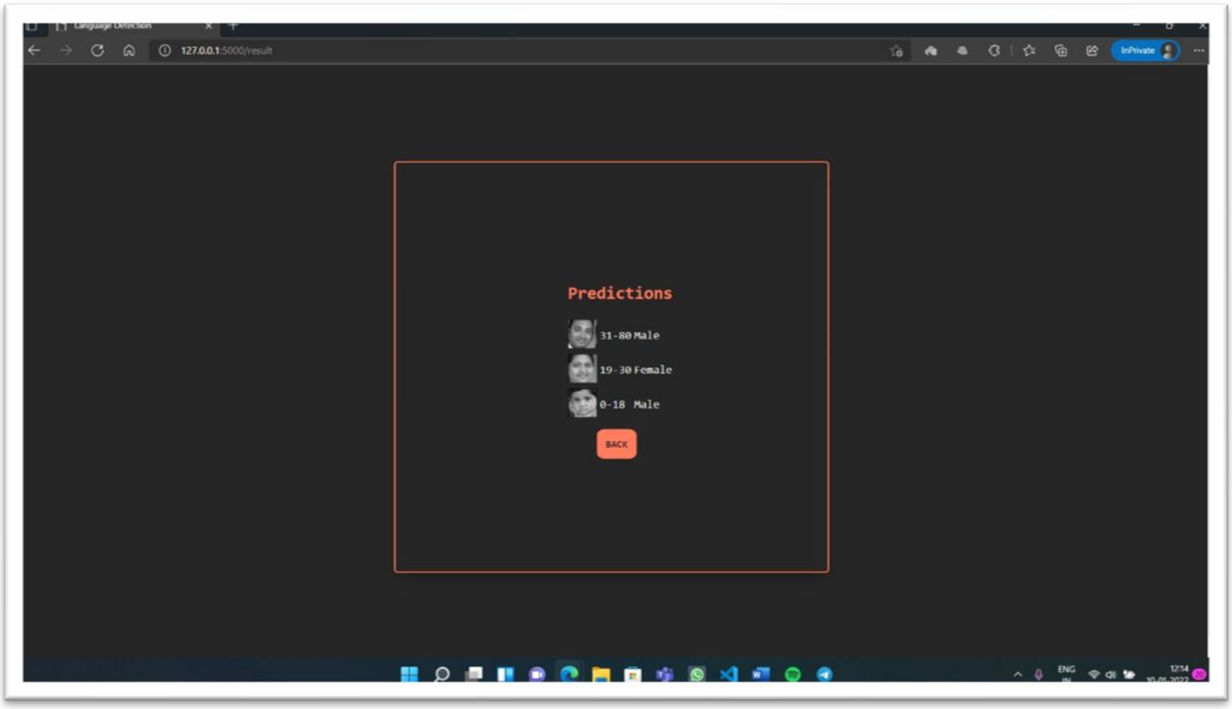
## WEB APP:

The trained model is integrated with frontend using Flask framework






Language Detection
127.0.0.1:5000/result
InPrivate
Predicting...


Language Detection
127.0.0.1:5000/result
InPrivate
Predictions
31-80 Male
19-30 Female
0-18 Male
BACK

## REFERNCES:

### Project source files:

Model Training

Testing interface

Dataset

### Base paper:

Human Gender And Age Detection

### References:

- ✓ G. Levi, and T. Hassner,” Age and Gender Classification Using Convolutional Neural Networks,” IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, 2015.
- ✓ Developer guides (keras.io)
- ✓ Module: tf | TensorFlow Core v2.7.0