Signs Of Ageing

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by
S.Sai Sravya, 121810301031

Under the esteemed guidance of

Verzeo Company

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM
(Deemed to be University)
VISAKHAPATNAM
Apr-May,2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)

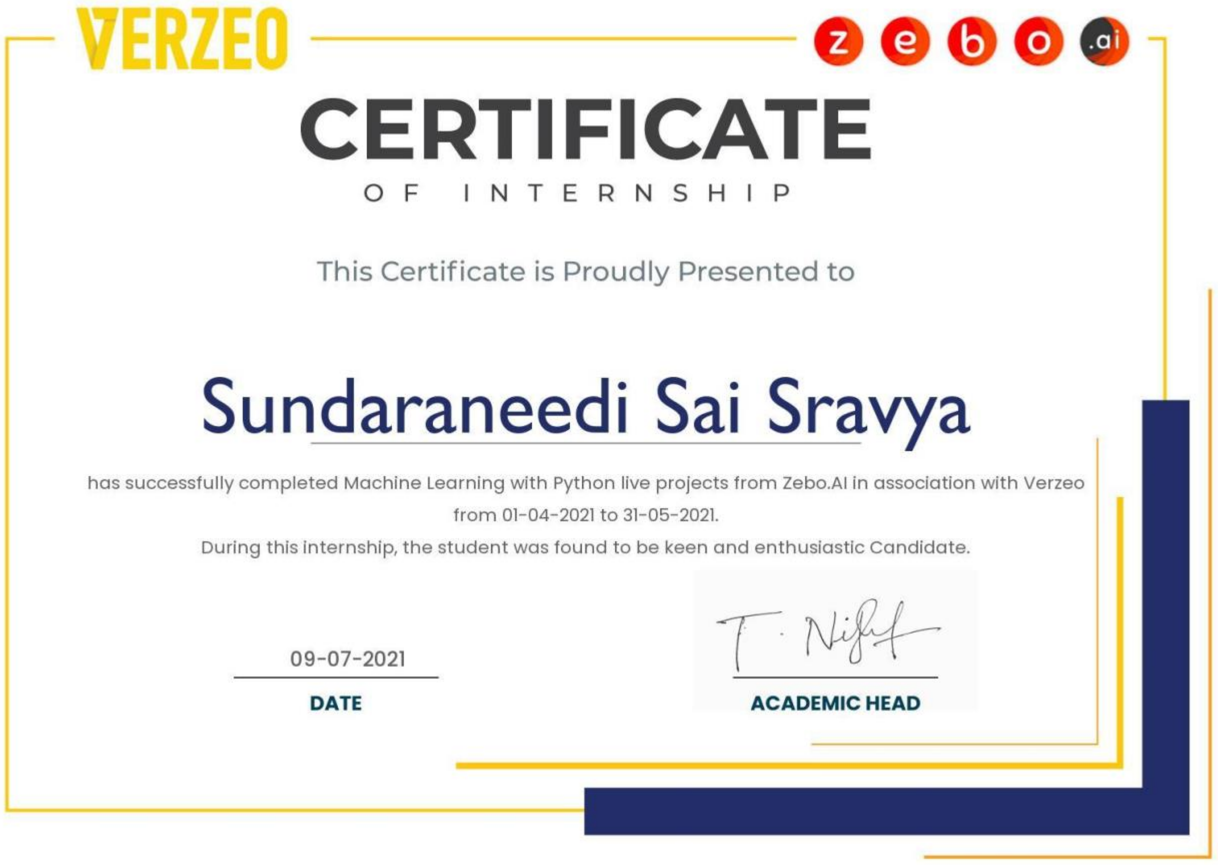
DECLARATION

I/We, hereby declare that the project report entitled "**Signs of Ageing**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:6/11/2021

Registration No(s).	**Name(s)**	**Signature(s)**
121810301031	**S.Sai Sravya**	

PROJECT COMPLETION CERTIFICATE FROM THE ORGANISATION/ COMPANY


VERZEO

zebo.ai

CERTIFICATE

OF INTERNSHIP

This Certificate is Proudly Presented to

Sundaraneedi Sai Sravya

has successfully completed Machine Learning with Python live projects from Zebo.AI in association with Verzeo from 01-04-2021 to 31-05-2021.

During this internship, the student was found to be keen and enthusiastic Candidate.

09-07-2021

DATE

[illegible]

ACADEMIC HEAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled **“Signs Of Ageing”** is a bonafide record of work carried out by

S.Sai Sravya, 121810301031

students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

INTERNSHIP REVIEWER 1	**INTERNSHIP REVIEWER 2**
P. Swarnalatha	**R. Sireesha**
Assistant Professor	**Professor**

TABLE OF CONTENTS

Abstract

Within the area of aging research, modern artificial intelligence (AI) algorithms have tremendous potential. A common characteristic of all living organisms, tissues, and cells is aging. The use of deep learning to develop age predictors opens up possibilities for dynamic and static data types that were previously incompatible. The use of AI biomarkers of aging allows a holistic view of biological processes as well as new methods for creating causal models-extracting the most significant biological targets and identifying significant adversarial networks.New developments in generative adversarial networks (GANs) and reinforcement learning (RL) enable the generation of diverse synthetic molecular and patient data, identification of novel biological targets, and generation of novel molecular compounds with desired properties and geroprotectors.These novel approaches may be combined into a unified, seamless process for biomarker development, target identification, drug discovery, and real-world evidence generation that may help accelerate and improve pharmaceutical research and development. Consequently, modern AI is expected to contribute to the credibility and prominence of longevity biotechnology in the healthcare and pharmaceutical industry, as well as to the convergence of numerous research fields.

In this project, there will be 1100 images that are given as input for the algorithm as datasets. The goal of this project is to tell us the probability percentage of ageing signs i.e. wrinkles, dark spots and puffy eyes of the image file that is inserted into the algorithm based on the datasets. The algorithm used here is linear regression algorithm.

About Organization

Verzeo is an upskilling e-learning platform headquartered in Bengaluru, India. It uses machine learning and artificial intelligence to assist students in gaining technological knowledge and skills required for employment.

Verzeo was founded in 2017 by Vungarala V Subrahmanyam. The idea behind Verzeo is to bridge the gap between the classroom environment and work-space environment by training students in new technological skills. The platform works online and offline, and provides training to undergraduate and postgraduate students in regional languages. Upskilling happens in the fields of cloud computing, artificial intelligence, cyber security, blockchain technology, augmented reality, web development, virtual reality, and other technology-related programs.

Post the learning, students get internship opportunities with multinational companies (MNCs). Their offline centers are in Mumbai, New Delhi, Bengaluru, Chennai, and Hyderabad, and focuses on students from Tier II and Tier III institutes.

In April 2018, Verzeo raised undisclosed seed funding and $5 million from high net-worth individuals in September of the same year.

Introduction

A machine that learns from experience without being programmed is called a machine learning system. Machine learning is a subtopic of Artificial Intelligence, or we can say that it is part of the field.

ML is the science of making computers take their own actions. Generally, a Machine Learning algorithm is a program that is designed to analyze and build models based on data. Users can use these models to perform various tasks.

Introduction To Machine Learning (ML)

Machine Learning has gained traction in the field of information technology over the past few years, and is also part of our everyday lives. All technological processes need strong and intelligent data analysis as data is increasing day by day.

Using machine learning, we can avoid having to write new code for every new application. Machine learning reduces the number of different types of problems to some extent by forming prototypes. Among the well-known applications we see around are speech recognition, self-driving cars, and web search recommendations.

Thus, the core notion of machine learning is to create computer programmes that execute specific activities (tasks) and can learn automatically from that data (experience) and enhance their performance when provided with data (performance).

With practise, this performance improves. It's a step-by-step procedure. The various types of problems that these ML algorithms are applied to produce solutions that are trustworthy and good enough to be used as a prototype and rely on results.

What Is Machine Learning?

Machine Learning is a branch of artificial intelligence that focuses on creating computer programmes that learn from their mistakes and make predictions.

Let's look at a real-world example to illustrate why machine learning is necessary. Various online retailers, such as Amazon, Flipkart, and eBay, leverage the user's previous purchasing history and previous watching history to entice them to purchase further things.

These sites will use this information to estimate what products users will buy and watch in the future. The idea is that these sites will evaluate purchases, wish lists, carts, and the opinions of other users who have similar interests. It is usually desirable to make this entire procedure automatic in order to avoid any guessing and so save a significant amount of time.

The example above demonstrates how important machine learning is in today's environment. Machine learning assists firms in extracting meaningful information from large amounts of data, allowing them to make critical business choices.

Machine Learning is used for activities that are too difficult or complex for a human to complete. These tasks are fed into machine learning algorithms, which explore and create models to achieve the desired outcomes.

Evolution Of ML

In the realm of computer games and artificial intelligence, Arthur Samuel coined the phrase "machine learning" in 1959.

"A computer programme is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, increases with the experience E," Tom Mitchell wrote later in 1997.

Initially, Machine Learning was only used to recognise patterns. It was also defined as a computer's ability to learn through an iterative process without being explicitly programmed.

Machine learning has taken a new turn with the ever-increasing amount of data and the invention of big data. Machine learning algorithms can now calculate exceedingly difficult calculations over large amounts of data automatically.

These mathematical calculations are performed at a high rate and with great precision. Fraud detection, online recommendations, and other key examples in this field.

Problem Identification & Objectives

Aging is a complex, time-dependent process that results in the loss of function, biological and physical damage, and the emergence of a variety of age-related disorders. Because of the hierarchical nature of living systems, ageing impacts most regulatory processes in a gradual manner. The human body is a multi-leveled, complicated system made up of billions of individual cells that form various tissues. These tissues are the building blocks for organs, and organs are arranged in distinct systems to do specific duties, such as the lymphatic, respiratory, digestive, urinary, or reproductive systems. The complicated interplay between genes and ageing can have an impact between environmental, mechanistic, biochemical and evolutionary constraints.

As a result, problems affecting merely a few basic processes within the cells of one or more organs might spread throughout the body. This explains why ageing cannot be fully understood or managed when only a few physiological processes are monitored. Aging appears to be the long-term result of the disturbance of different dynamical equilibriums created between antagonistic processes, rather than the result of single molecular processes or components having intrinsic negative effects. The multifactorial and systemic aspect of ageing explains why understanding its biology and causes is so difficult, and why ageing research continues to require multidisciplinary and global methods. This project identifies the signs of ageing and tell us to how much percent probability the signs are present.

Overview of Technologies

Software that can be used:

- **Python3** - Guido van Rossum created Python in the early 1990s, and the most recent version is 3.7.1, which we can refer to as Python3. In 2008, Python 3.0 was released. It's an interpreted language, which means it's not compiled and the interpreter will go through each line of code. This article can be used to understand the fundamentals of Python programming.

- **OpenCV** - OpenCV is a large open-source library for image processing, computer vision, and machine learning. Python, C++, Java, and other programming languages are supported by OpenCV. It can analyse photos and videos to recognise items, faces, and even human handwriting. When it's paired with other libraries, such as Numpy, a highly efficient library for numerical operations, the number of weapons in your arsenal grows, as all of the operations that Numpy can do may be merged with OpenCV.

- **Python libraries** - Various python libraries, frameworks, and modules have made it much easier and efficient in modern times compared to the olden days. Python is now one of the most popular programming languages for this activity, and it has largely displaced many other languages in the business, thanks to its extensive library. Python libraries that used in Machine Learning are:

→ Numpy
→ Scipy
→ Scikit-learn
→ Theano
→ TensorFlow

- → Keras
- → PyTorch
- → Pandas
- → Matplotlib

- **Jupyter notebook/google collab -** Jupyter Notebook is an open-source web software that lets you create and share documents with live code, equations, visualisations, and narrative text. Data cleaning and transformation, numerical simulation, statistical modelling, data visualisation, machine learning, and other applications are only a few examples. Python is one of the more than 40 programming languages supported by Jupyter. Python (version 3.3 or greater, or Python 2.7) is required to install the Jupyter Notebook.

Google Collab is the platform for you if you want to construct a machine learning model but don't have a computer that can handle the workload. Creating a local environment with Anaconda and installing packages and resolving installation issues is a chore even if you have a GPU or a decent PC. Collaboratory is a Google-provided free Jupyter notebook environment where you can leverage free GPUs and TPUs to solve all of these problems.

Operating system can be used:

- Windows/ubuntu

Hardware requirement:

- Laptop with basics hardware

Implementation

Coding & Testing :-

Steps taken for the project:

1)**Data Gathering**: -

To do this project first we have to look for desired datasets which are used in ageing face detection. The dataset contains various features of ageing people like puffy eyes, dark spot, wrinkles, dark circle and etc.

After this we need to import some libraries the necessary packages are:

- Pandas,
- Sklearn,
- Numpy,
- Matplotlib,
- Opencv,
- Seaborn.

2) **Performing EDA on the dataset**: -

In this section, we explore our dataset. First we will check the datatype and if there any missing values in data by using: Print(data.info)

So, we have no missing values in our dataset. We can now draw a pair diagram to get an overview of the relationship between all the entities.

Now, lets check whether our data is balanced or not because we need perfectly balanced data to perform binary classification task.

Data processing:

Now we need to balance our data, the easiest way to do is to randomly drop a number of instances of the overrepresented target function, this called random under sampling. Otherwise we could also create new synthetic data for the under-represented target class. This is called oversampling.

After all these above processes will train and test our model for ageing sign detection by using machine learning and image processing.

3) **Image Processing and Training of the dataset**: -

We can detect or extract the features of an image by using image processing and we can give those sample as an input to the regression algorithm.

There are many techniques that can be used:

- Features extraction
- Image segmentation
- Edge detection

4) **Testing and Predicting the dataset - AI and ML**: -

This is final step of our project, In this we can train our model By using regression algorithm we can detect deep features of input image then we can identify the ageing sign of the given input.

Packages needed to be installed :

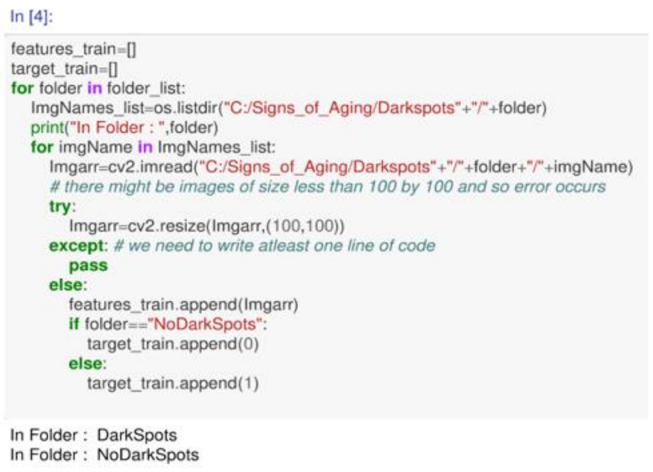
1. Tensorflow
2. Keras
3. Opencv
4. Pandas
5. Numpy
6. OS

STEPS TO RUN THE CODE:

→ Load the respective models and their corresponding weights.
→ The xml files named , 'haarcascade_frontal_default.xml' and 'haarcascade_eye.xml' must be downloaded and the path where the xml files are downloaded must be specified/changed in the codes.
→ Change the 'image_path' variable to the path of the image file that you want to test on.
→ To test on the image file run 'Wrinkles.ipynb' , 'PuffyEyes.ipynb' and 'DarkSpots.ipynb' first.
→ Then Run the 'prediction.ipynb'

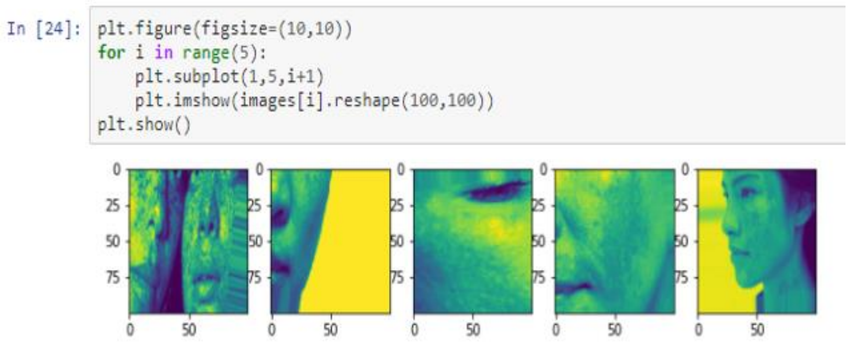
→ The image will pop up after the execution of ‘prediction.ipynb’ , click enter when the image shows and it will show the class and probabilities of the respective class in the given image.

Darkspots.ipynb :-



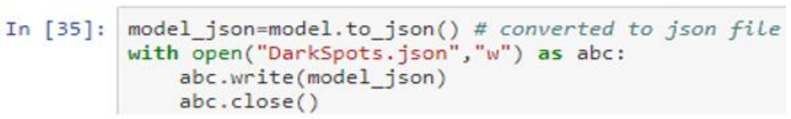
```
In [4]:
features_train=[]
target_train=[]
for folder in folder_list:
    ImgNames_list=os.listdir("C:/Signs_of_Aging/Darkspots"+"/"+folder)
    print("In Folder : ",folder)
    for imgName in ImgNames_list:
        Imgarr=cv2.imread("C:/Signs_of_Aging/Darkspots"+"/"+folder+"/"+imgName)
        # there might be images of size less than 100 by 100 and so error occurs
        try:
            Imgarr=cv2.resize(Imgarr,(100,100))
        except: # we need to write atleast one line of code
            pass
        else:
            features_train.append(Imgarr)
            if folder=="NoDarkSpots":
                target_train.append(0)
            else:
                target_train.append(1)

In Folder :  DarkSpots
In Folder :  NoDarkSpots
```

The images in the datasets that are stored as per label are appended with 0 or 1 depending on the type. In the above code, if the folder is no dark spots then 1 is appended and if the folder is dark spots 0 is appended.

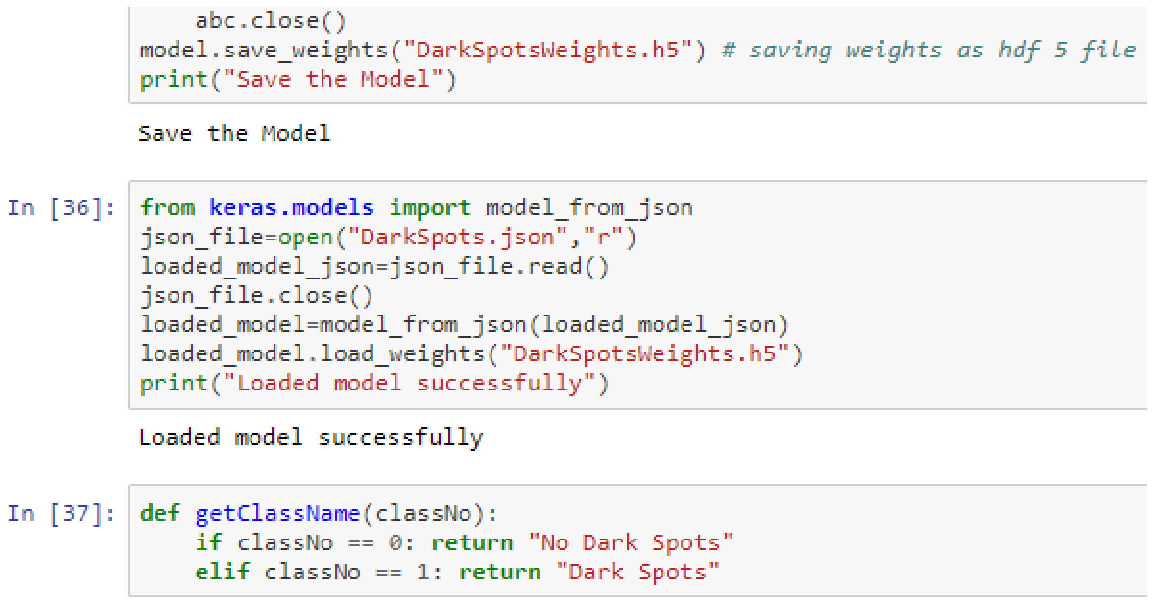


```
In [24]: plt.figure(figsize=(10,10))
         for i in range(5):
             plt.subplot(1,5,i+1)
             plt.imshow(images[i].reshape(100,100))
         plt.show()
```

The images are then sized and written to json files.



```
In [35]: model_json=model.to_json() # converted to json file
         with open("DarkSpots.json","w") as abc:
             abc.write(model_json)
             abc.close()
```



```
    abc.close()
model.save_weights("DarkSpotsWeights.h5") # saving weights as hdf 5 file
print("Save the Model")
```

```
Save the Model
```

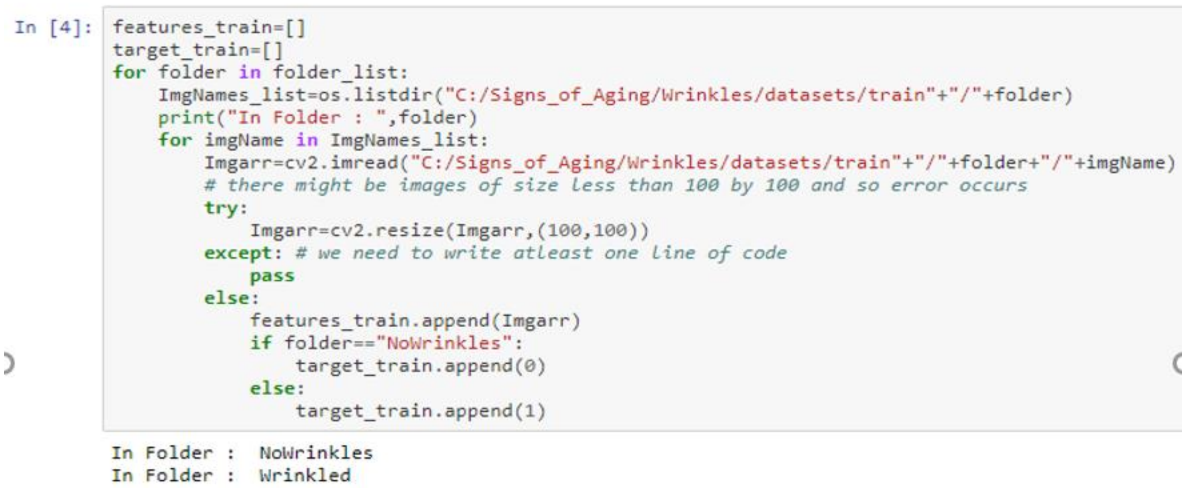
```
In [36]: from keras.models import model_from_json
         json_file=open("DarkSpots.json","r")
         loaded_model_json=json_file.read()
         json_file.close()
         loaded_model=model_from_json(loaded_model_json)
         loaded_model.load_weights("DarkSpotsWeights.h5")
         print("Loaded model successfully")
```

```
Loaded model successfully
```

```
In [37]: def getClassName(classNo):
             if classNo == 0: return "No Dark Spots"
             elif classNo == 1: return "Dark Spots"
```

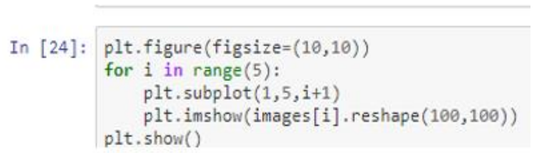
The same method is followed for the wrinkles and puffy eyes.

Wrinkles.ipynb:-

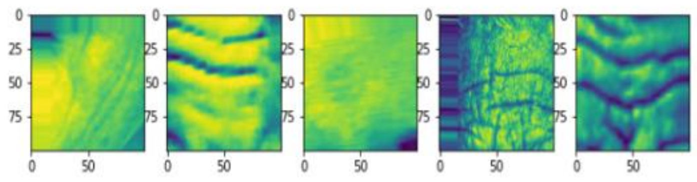


```
In [4]: features_train=[]
        target_train=[]
        for folder in folder_list:
            ImgNames_list=os.listdir("C:/Signs_of_Aging/Wrinkles/datasets/train"+"/"+folder)
            print("In Folder : ",folder)
            for imgName in ImgNames_list:
                Imgarr=cv2.imread("C:/Signs_of_Aging/Wrinkles/datasets/train"+"/"+folder+"/"+imgName)
                # there might be images of size less than 100 by 100 and so error occurs
                try:
                    Imgarr=cv2.resize(Imgarr,(100,100))
                except: # we need to write atleast one line of code
                    pass
                else:
                    features_train.append(Imgarr)
                    if folder=="NoWrinkles":
                        target_train.append(0)
                    else:
                        target_train.append(1)
```

```
In Folder :  NoWrinkles
In Folder :  Wrinkled
```



```
In [24]: plt.figure(figsize=(10,10))
         for i in range(5):
             plt.subplot(1,5,i+1)
             plt.imshow(images[i].reshape(100,100))
         plt.show()
```



```
In [35]: model_json=model.to_json() # converted to json file
         with open("Wrinkle.json","w") as abc:
             abc.write(model_json)
             abc.close()
         model.save_weights("WrinkleWeights.h5") # saving weights as hdf 5 file
         print("Save the Model")

         Save the Model
```

```
In [36]: from keras.models import model_from_json
         json_file=open("Wrinkle.json","r")
         loaded_model_json=json_file.read()
         json_file.close()
         loaded_model=model_from_json(loaded_model_json)
         loaded_model.load_weights("WrinkleWeights.h5")
         print("Loaded model successfully")

         Loaded model successfully
```

```
In [37]: def getClassName(classNo):
             if classNo == 0: return "No Wrinkles"
             elif classNo == 1: return "Wrinkled"
```

```
In [43]: import cv2
         image=cv2.imread('C:/Signs_of_Aging/Anthony_Fauci_2020.jpg')
```

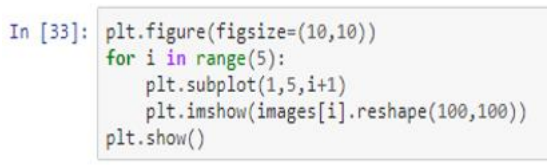
```
In [44]: cv2.imshow("Fauci",image)
         cv2.waitKey(5000)
         cv2.destroyAllWindows()
```

```
In [45]: face_cascade=cv2.CascadeClassifier("C:/xml files/haar-cascade-files-master/haarcascade_frontalface_default.xml")
```

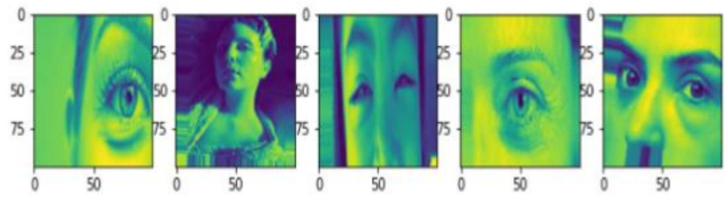
```
In [46]: gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
         faces=face_cascade.detectMultiScale(gray,1.05,15) # 5% reduction , iff the rectangle identifies the face 3 times
         for (x,y,w,h) in faces: # x,y->coordinates of rectangles on faces, w-> width of rectangle, h->height of rectangle
             cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
         img=image[x:x+w,y:y+h]
```

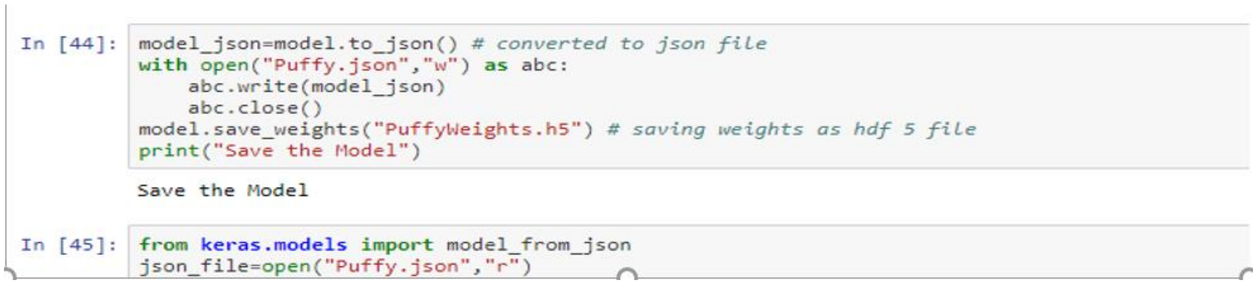
```
In [ ]: imagearr=cv2.resize(img,(100,100))
        imagearr=preprocessing(imagearr)
        imagearr=imagearr.reshape((1,100,100,1))
        predictions=loaded_model.predict(imagearr)
        classIndex=loaded_model.predict_classes(imagearr)
        probValue=np.amax(predictions)
        cv2.putText(image,"Class: ",(20,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
        cv2.putText(image,"Probability: ",(20,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
        if probValue>0.75:
            cv2.putText(image,getClassName(classIndex),(120,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
            cv2.putText(image,str(int(probValue*100))+" %",(200,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
        cv2.imshow("Model Prediction",image)
        key=cv2.waitKey(0)
        if key==ord("\r"):
            cv2.destroyAllWindows()
```

PuffyEyes.ipynb:-

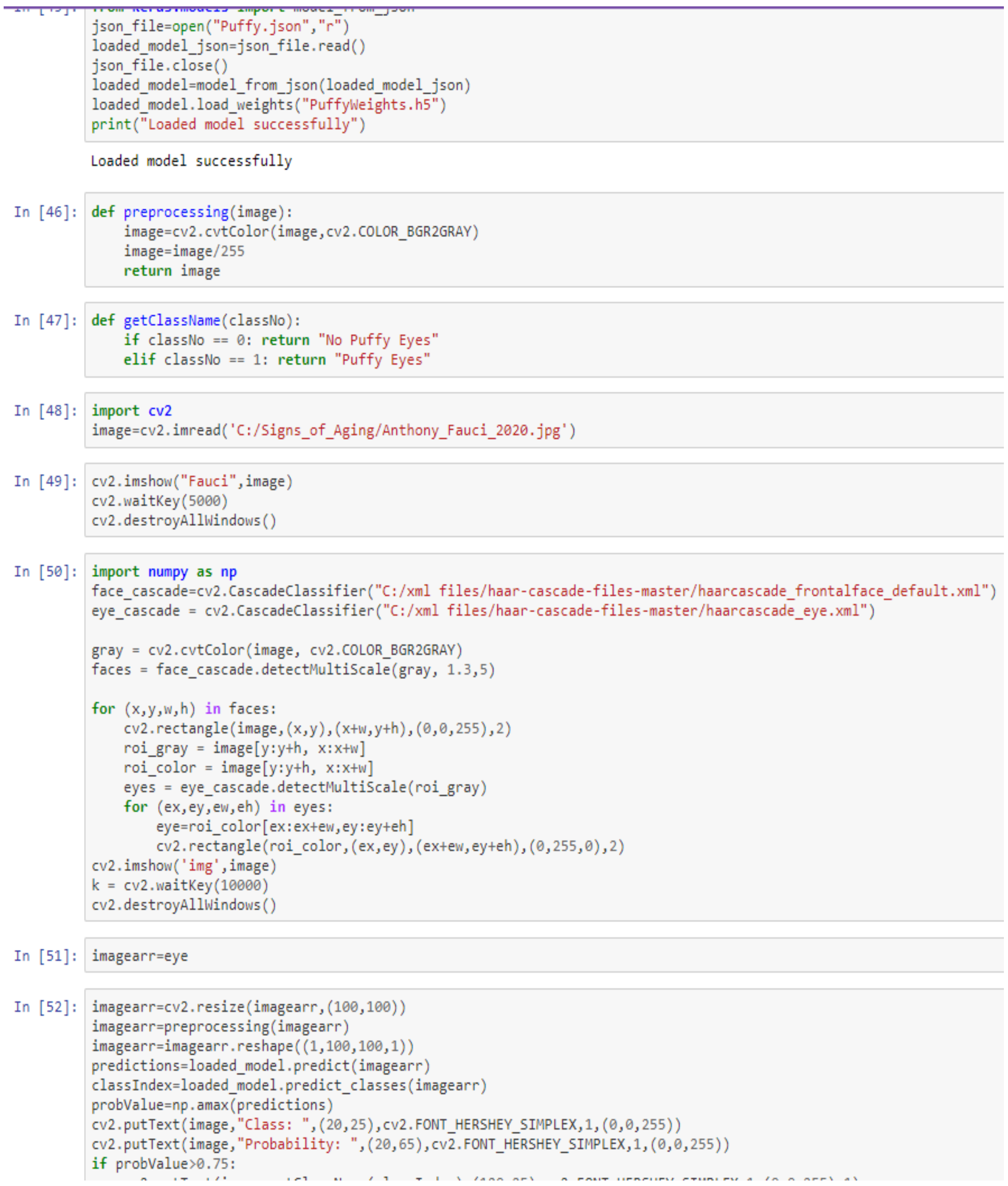


```
In [33]: plt.figure(figsize=(10,10))
         for i in range(5):
             plt.subplot(1,5,i+1)
             plt.imshow(images[i].reshape(100,100))
         plt.show()
```





```
In [44]: model_json=model.to_json() # converted to json file
         with open("Puffy.json","w") as abc:
             abc.write(model_json)
             abc.close()
         model.save_weights("PuffyWeights.h5") # saving weights as hdf 5 file
         print("Save the Model")

         Save the Model

In [45]: from keras.models import model_from_json
         json_file=open("Puffy.json","r")
```



```
json_file=open("Puffy.json","r")
loaded_model_json=json_file.read()
json_file.close()
loaded_model=model_from_json(loaded_model_json)
loaded_model.load_weights("PuffyWeights.h5")
print("Loaded model successfully")
```

```
Loaded model successfully
```

```
In [46]: def preprocessing(image):
             image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
             image=image/255
             return image
```

```
In [47]: def getClassName(classNo):
             if classNo == 0: return "No Puffy Eyes"
             elif classNo == 1: return "Puffy Eyes"
```

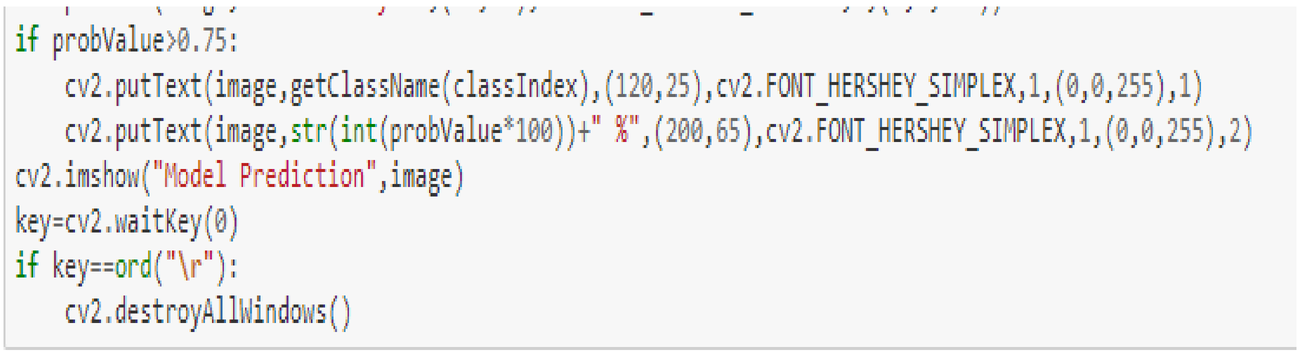
```
In [48]: import cv2
         image=cv2.imread('C:/Signs_of_Aging/Anthony_Fauci_2020.jpg')
```

```
In [49]: cv2.imshow("Fauci",image)
         cv2.waitKey(5000)
         cv2.destroyAllWindows()
```

```
In [50]: import numpy as np
         face_cascade=cv2.CascadeClassifier("C:/xml files/haar-cascade-files-master/haarcascade_frontalface_default.xml")
         eye_cascade = cv2.CascadeClassifier("C:/xml files/haar-cascade-files-master/haarcascade_eye.xml")

         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
         faces = face_cascade.detectMultiScale(gray, 1.3,5)

         for (x,y,w,h) in faces:
             cv2.rectangle(image,(x,y),(x+w,y+h),(0,0,255),2)
             roi_gray = image[y:y+h, x:x+w]
             roi_color = image[y:y+h, x:x+w]
             eyes = eye_cascade.detectMultiScale(roi_gray)
             for (ex,ey,ew,eh) in eyes:
                 eye=roi_color[ex:ex+ew,ey:ey+eh]
                 cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
         cv2.imshow('img',image)
         k = cv2.waitKey(10000)
         cv2.destroyAllWindows()
```

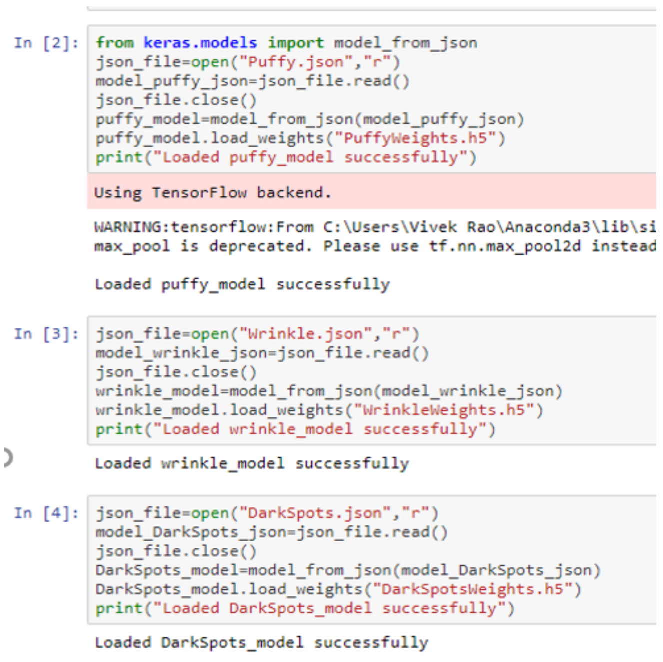
```
In [51]: imagearr=eye
```

```
In [52]: imagearr=cv2.resize(imagearr,(100,100))
         imagearr=preprocessing(imagearr)
         imagearr=imagearr.reshape((1,100,100,1))
         predictions=loaded_model.predict(imagearr)
         classIndex=loaded_model.predict_classes(imagearr)
         probValue=np.amax(predictions)
         cv2.putText(image,"Class: ",(20,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
         cv2.putText(image,"Probability: ",(20,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
         if probValue>0.75:
```

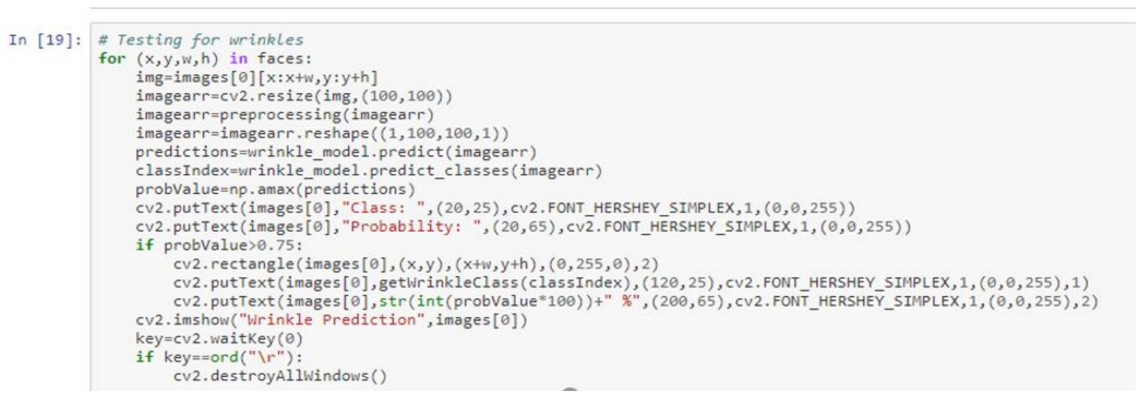


```
if probValue>0.75:
    cv2.putText(image,getClassName(classIndex),(120,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
    cv2.putText(image,str(int(probValue*100))+" %",(200,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
cv2.imshow("Model Prediction",image)
key=cv2.waitKey(0)
if key==ord("\r"):
    cv2.destroyAllWindows()
```

Prediction.ipynb:-



```
In [2]: from keras.models import model_from_json
        json_file=open("Puffy.json","r")
        model_puffy_json=json_file.read()
        json_file.close()
        puffy_model=model_from_json(model_puffy_json)
        puffy_model.load_weights("PuffyWeights.h5")
        print("Loaded puffy_model successfully")

        Using TensorFlow backend.

        WARNING:tensorflow:From C:\Users\Vivek Rao\Anaconda3\lib\si
        max_pool is deprecated. Please use tf.nn.max_pool2d instead

        Loaded puffy_model successfully

In [3]: json_file=open("Wrinkle.json","r")
        model_wrinkle_json=json_file.read()
        json_file.close()
        wrinkle_model=model_from_json(model_wrinkle_json)
        wrinkle_model.load_weights("WrinkleWeights.h5")
        print("Loaded wrinkle_model successfully")

        Loaded wrinkle_model successfully

In [4]: json_file=open("DarkSpots.json","r")
        model_DarkSpots_json=json_file.read()
        json_file.close()
        DarkSpots_model=model_from_json(model_DarkSpots_json)
        DarkSpots_model.load_weights("DarkSpotsWeights.h5")
        print("Loaded DarkSpots_model successfully")

        Loaded DarkSpots_model successfully
```



```
In [19]: # Testing for wrinkles
         for (x,y,w,h) in faces:
             img=images[0][x:x+w,y:y+h]
             imagearr=cv2.resize(img,(100,100))
             imagearr=preprocessing(imagearr)
             imagearr=imagearr.reshape((1,100,100,1))
             predictions=wrinkle_model.predict(imagearr)
             classIndex=wrinkle_model.predict_classes(imagearr)
             probValue=np.amax(predictions)
             cv2.putText(images[0],"Class: ",(20,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
             cv2.putText(images[0],"Probability: ",(20,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
             if probValue>0.75:
                 cv2.rectangle(images[0],(x,y),(x+w,y+h),(0,255,0),2)
                 cv2.putText(images[0],getWrinkleClass(classIndex),(120,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
                 cv2.putText(images[0],str(int(probValue*100))+" %",(200,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
             cv2.imshow("Wrinkle Prediction",images[0])
             key=cv2.waitKey(0)
             if key==ord("\r"):
                 cv2.destroyAllWindows()
```

```
WARNING:tensorflow:From C:\Users\Vivek Rao\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.glob
al_variables is deprecated. Please use tf.compat.v1.global_variables instead.
```

In [20]:

```
# Testing for puffy eyes
for (x,y,w,h) in faces:
    roi_gray = images[1][y:y+h, x:x+w]
    roi_color = images[1][y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        eye=roi_color[ex:ex+ew,ey:ey+eh]
        imagearr=eye
        imagearr=cv2.resize(imagearr,(100,100))
        imagearr=preprocessing(imagearr)
        imagearr=imagearr.reshape((1,100,100,1))
        predictions=puffy_model.predict(imagearr)
        classIndex=puffy_model.predict_classes(imagearr)
        probValue=np.amax(predictions)
        cv2.putText(images[1],"Class: ",(20,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
        cv2.putText(images[1],"Probability: ",(20,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
        if probValue>0.75:
            cv2.putText(images[1],getPuffyClass(classIndex),(120,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
            cv2.putText(images[1],str(int(probValue*100))+" %",(200,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
            for (ex,ey,ew,eh) in eyes:
                imgarr=roi_color[ex:ex+ew,ey:ey+eh]
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
            break
    cv2.imshow("Puffy Eyes Prediction",images[1])
    key = cv2.waitKey(0)
    if key==ord("\r"):
        cv2.destroyAllWindows()
```

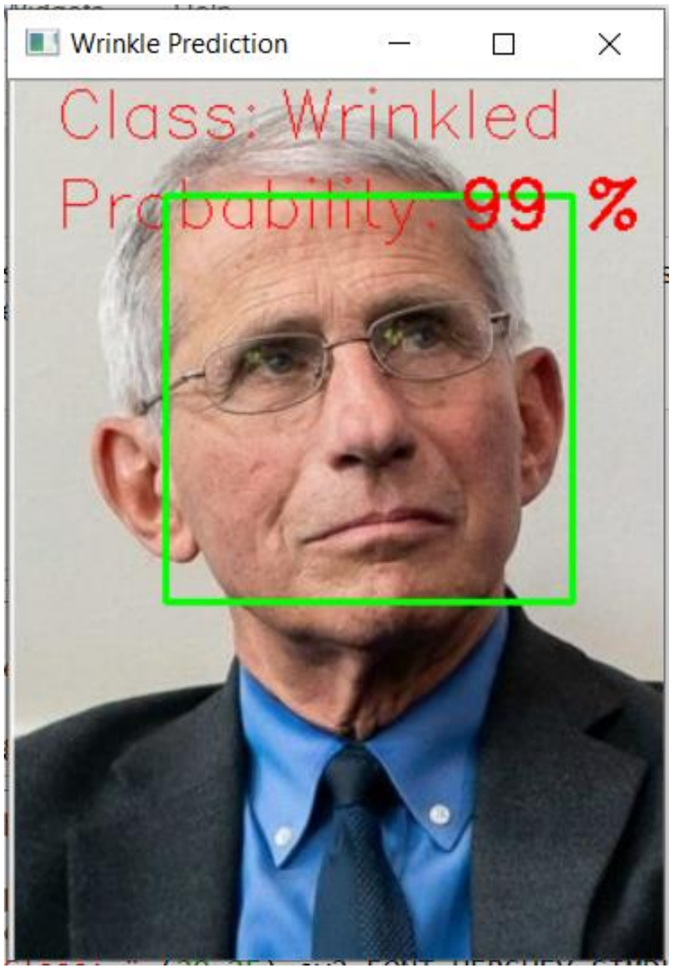
In [21]:

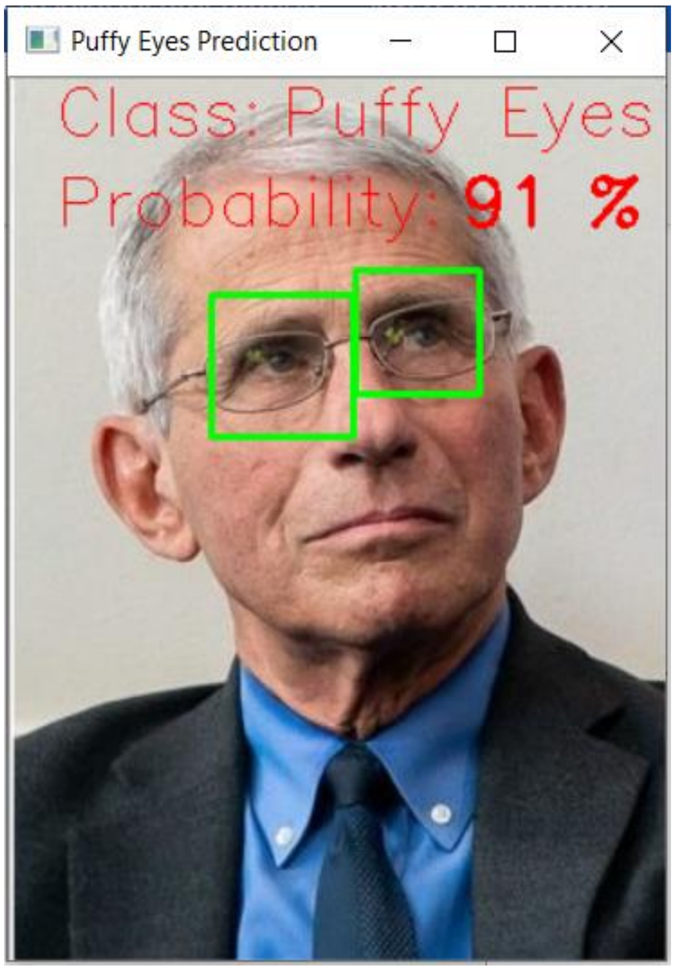
```
# Testing for Dark Spots
for (x,y,w,h) in faces:
    img=images[2][x:x+w,y:y+h]
    imagearr=cv2.resize(img,(100,100))
    imagearr=preprocessing(imagearr)
    imagearr=imagearr.reshape((1,100,100,1))
    predictions=DarkSpots_model.predict(imagearr)
    classIndex=DarkSpots_model.predict_classes(imagearr)
    probValue=np.amax(predictions)
    cv2.putText(images[2],"Class: ",(20,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
    cv2.putText(images[2],"Probability: ",(20,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255))
    if probValue>0.75:
        cv2.rectangle(images[2],(x,y),(x+w,y+h),(0,255,0),2)
        cv2.putText(images[2],getDarkSpotsClass(classIndex),(120,25),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),1)
        cv2.putText(images[2],str(int(probValue*100))+" %",(200,65),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
    cv2.imshow("Dark Spots Prediction",images[2])
    key=cv2.waitKey(0)
    if key==ord("\r"):
        cv2.destroyAllWindows()
```

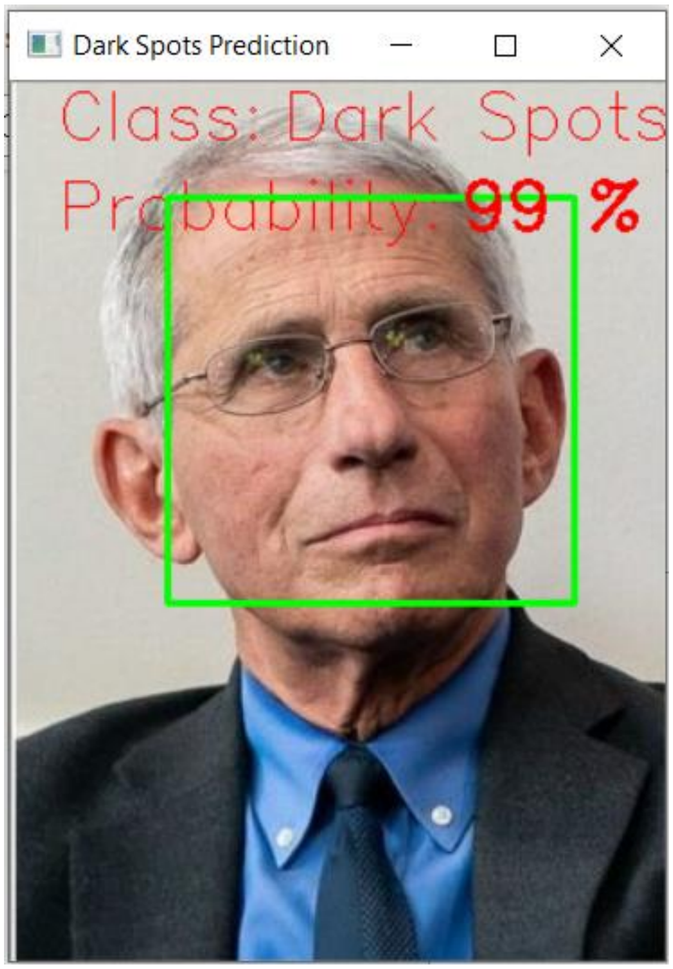
In [22]:

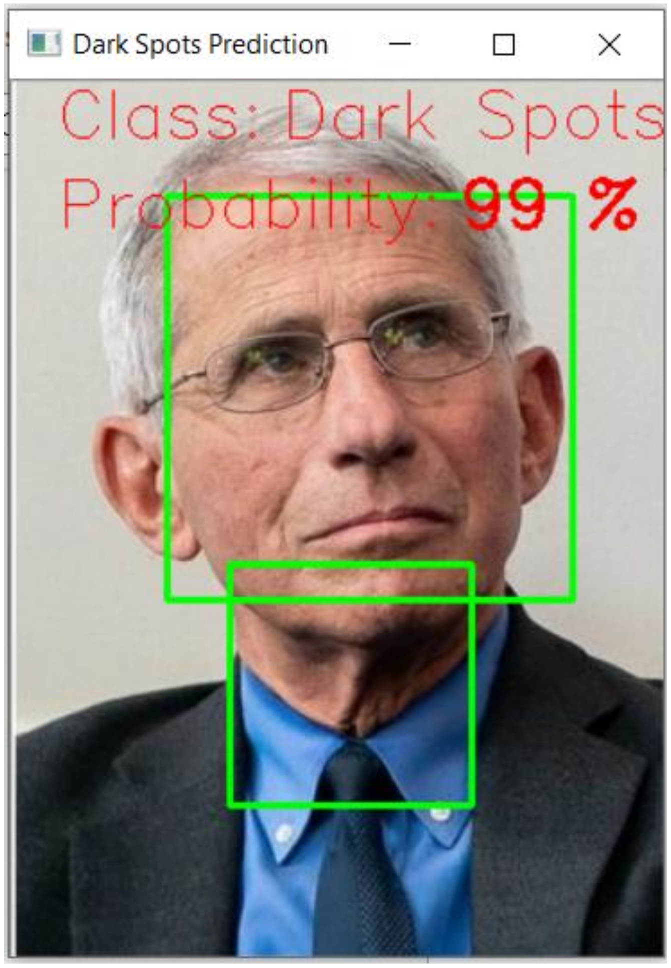
```
for i in range(3):
    cv2.imshow("Fauci",images[i])
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
```

Results




Puffy Eyes Prediction
Class: Puffy Eyes
Probability: 91 %


Dark Spots Prediction
Class: Dark Spots
Probability: 99 %



As we can see, the accuracy of the code is good as the algorithm recognizes the dark spots, wrinkles and puffy eyes of the image that has been appended in the code.

Conclusion

Machine Learning is a part of Artificial Intelligence which can make predictions using pattern and trends recognition in data. The ML algorithms have self-learning capabilities and do not require human interference for error calculation.

ML algorithms adapt themselves on their own and learn from the previous data to show results for the new data fed into the system and also identify the hidden trends and patterns in the data. It is an iterative process.

Artificial Intelligence is a field of computer science that makes artificial things such as computers, or other devices intelligent by feeding them with data and code. These devices are then able to behave like humans.

There are various machine learning algorithms and tools that are available for businesses to use. The only key here is that the business should know which is the right algorithm and the right tool to build a machine learning model for their organization’s benefit.

References

D. Kriegman, M.H. Yang and N. Ahuja, “Detecting faces in images: a survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, No 1, 2002, pp. 34-58.

E Hjelmas, “Face detection: a survey”, Computer vision and image understanding, vol. 83, No 3, 2001, pp. 236-274