

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | U3256125  Yawen Wu |
| **Unit name** | Software Technology 1G |
| **Unit number** | 8995 |
| **Unit Tutor** | Linda Ma |
| **Assignment name** | ST1 Capstone Project – Semester 2 2023 |
| **Due date** | 29/10/2023 |
| **Date submitted** | 22/10/2023 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

**Signature of student: 文本

描述已自动生成 Date: 文本, 白板

描述已自动生成**

Table of Contents

[Introduction and Background 3](#_Toc148043639)

[Dataset Description 4](#_Toc148043640)

[Methodology 4](#_Toc148043641)

[Stage 1: Exploratory Data Analysis Stage 5](#_Toc148043642)

[Exploratory Data Analysis 5](#_Toc148043643)

[Stage 1: EDA (Exploratory Data Analytics) 5](#_Toc148043644)

[Stage 2: PDA (Predictive Data Analytics) 20](#_Toc148043645)

[Stage 2A With Google Colab or Pycharm) 20](#_Toc148043646)

[Stage 2B PDA with Teachable Machine withGoogle 29](#_Toc148043647)

[Stage 3A Deployment/Implemenation with Teachable Machine withGoogle 31](#_Toc148043648)

[Stage 3B Deployment/Implemenation with TkInter 33](#_Toc148043649)

[Conclusions 35](#_Toc148043650)

[References 36](#_Toc148043651)

[Appendix 1: Log Book 37](#_Toc148043652)

# Introduction and Background

This report describes the details of Python Capstone Project for ST1 unit within the scope of the project requirements provided in the assignment handout [1,2]. I have decided to work on the project developing a gender classification from public available dataset in Kaggle repository [1,2].

In recent years, computer technology has improved a lot. This project takes advantage of these improvements to make a tool that can tell the gender of celebrities in pictures. This can be helpful for things like checking online content, showing advertisements to the specific audience, facial recognization applications.

My project is all about making a tool that can guess the gender of famous people in photos. I’m using a big collection of 200,000 celebrity pictures to help our tool get really good at this. I want the tool to be great at telling if a person is a man or a woman when it looks at their picture. This can be very useful for different things.

This Capstone project report presents the details of prototype software technology platform ols, in terms of several Python software tools developed as part of this capstone project, based on a data driven scientific approach, involving exploratory data analysis, predictive analytics and implementation as a desktop Tkinter application for gender classfication. Rest of the report is organised as follows.

The details of the dataset used is described in Section 2 and the methodology used for development of software tool is presented in the Section 3. The details of code development, performance evaluation and deployment is discussed in Section 4. The report ends with conclusions and recommendations in Section 5 and Section 6 lists some key references and bibliography.

# Dataset Description

This dataset is used for Gender Classification with images. The dataset consists of almost 200K images which are almost 1.3GB in size[1]. Images cover large pose variations, background clutter, diverse people, supported by a large quantity of images and rich annotations[2].

Data Description:

* Type of data: 178x218 celeberity images.
* Data format: JPG.
* Number of images: 200000 images.
  + Of these, half are of male celebrities and the rest are female celebrities.
* Gender label
  + Male
  + Female
* Number of classes: two.
* Distribution of instances: Each of the two categories contains 100000 images.
* How data are acquired: Captured from websites by researchers at MMLAB, The Chinese University of Hong Kong.
* Applicability: This dataset is great for training and testing models for face detection, particularly for recognising facial attributes such as finding people with brown hair, are smiling, or wearing glasses.

# Methodology

The methodology used for developing the software platform involves 3 stages as outlined below:

1. Stage 1: Exploratory Data Analysis for celebrity images from the dataset using Google Colab.
2. Stage 2: Predictive analytics development using machine learning platform using teachable machine.
3. Stage 3: Implementation and Deployment of the software technology tool for real world field testing (using Python Tkinter)

## Stage 1: Exploratory Data Analysis Stage

Stage 1 is most important preliminary stage and the purpose of exploratory data analysis is to obtain a thorough understanding of data, and inform about the choice of predictive analytics algorithms to be used, and expected performance of the software tool in real world settings.

### Exploratory Data Analysis

The first phase of the software development activity involved understanding the data, basic exploratory data analysis and visualisation. Google Colab was chosen as the experimental environment. Before the exploratory data analysis can begin, some of steps required are:

* Downloading data subsets from Kaggle repository

(<https://www.kaggle.com/datasets/ashishjangra27/gender-recognition-200k-images-celeba> )

* Steps for EDA with Google Colab
  + Uploading the data to the Google drive
  + Mapping the Google drive to a new Google Colab notebook
  + Installation/Importing different python libraries for performing EDA
  + Changing the project working folder to the google drive location where project dataset is located.
  + These steps can be done using the following Python script.

## Stage 1: EDA (Exploratory Data Analytics)

## Step1: Understanding the Problem Statement from dataset source

1. This dataset consists of collection of celebrity images which are separated into male group and female group.
2. The colour Images corresponding to each celebrity category are of size 178 x 218 x 7 jpg format.
3. Data is separated into train, test and valid directories.
4. This dataset can be used for developed an automatic (computer based) gender classifier from images.

from google.colab import drive

drive.mount('/content/drive')

Mounted at /content/drive

import cv2

import matplotlib.pyplot as plt

%matplotlib inline

# Read the images

img\_path\_1 = '/content/drive/MyDrive/Gender Classification/dataset/test/female/161161.jpg'

img\_1 = cv2.imread(img\_path\_1)

img\_path\_2 = '/content/drive/MyDrive/Gender Classification/dataset/test/male/160058.jpg'

img\_2 = cv2.imread(img\_path\_2)

# Convert BGR to RGB

img\_1\_rgb = cv2.cvtColor(img\_1, cv2.COLOR\_BGR2RGB)

img\_2\_rgb = cv2.cvtColor(img\_2, cv2.COLOR\_BGR2RGB)

# Create a subplot and display the images

plt.figure(figsize=(10, 10))

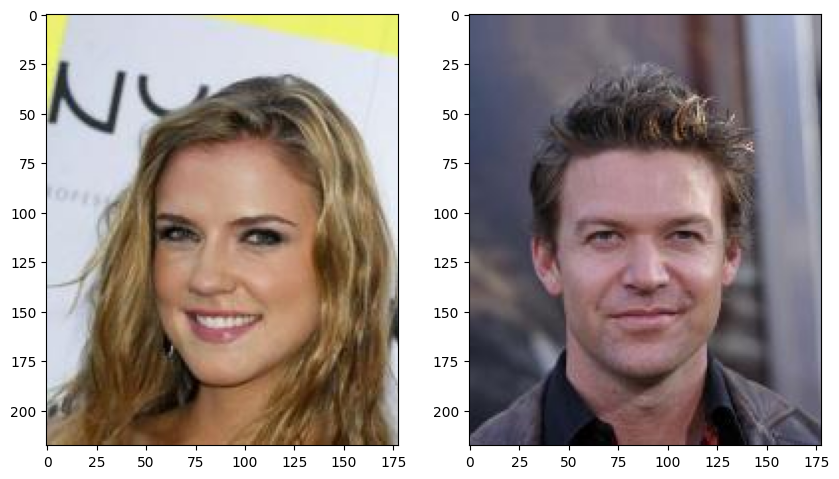
plt.subplot(121)

plt.imshow(img\_1\_rgb)

plt.subplot(122)

plt.imshow(img\_2\_rgb)

plt.show()



#Basic image manipulation (rotating/flipping)

flip\_img\_1=cv2.flip(img\_1\_rgb,0) # vertical flip

flip\_img\_2=cv2.flip(img\_2\_rgb,0) # vertical flip

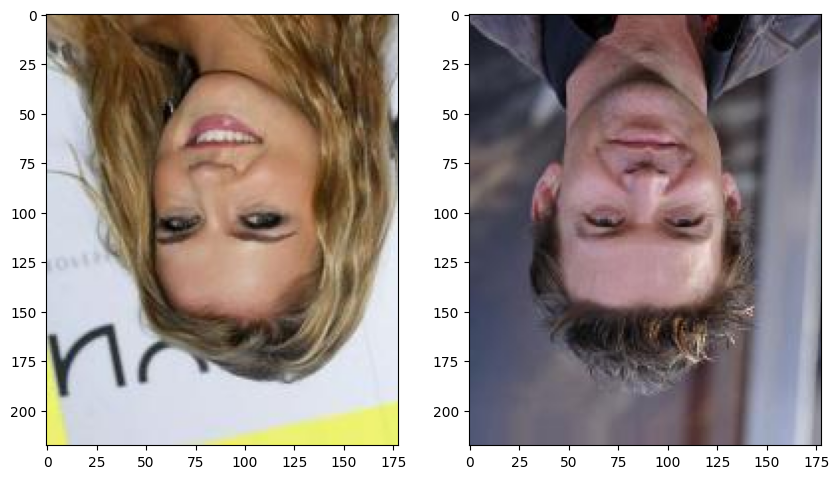
plt.figure(figsize=(10,10))

plt.subplot(121)

plt.imshow(flip\_img\_1)

plt.subplot(122)

plt.imshow(flip\_img\_2)



#horizontal flip

flip\_img\_1=cv2.flip(img\_1\_rgb,1) # horizontal flip

flip\_img\_2=cv2.flip(img\_2\_rgb,1) # horizontal flip

#horizontal flip

flip\_img\_1=cv2.flip(img\_1\_rgb,1) # horizontal flip

flip\_img\_2=cv2.flip(img\_2\_rgb,1) # horizontal flip

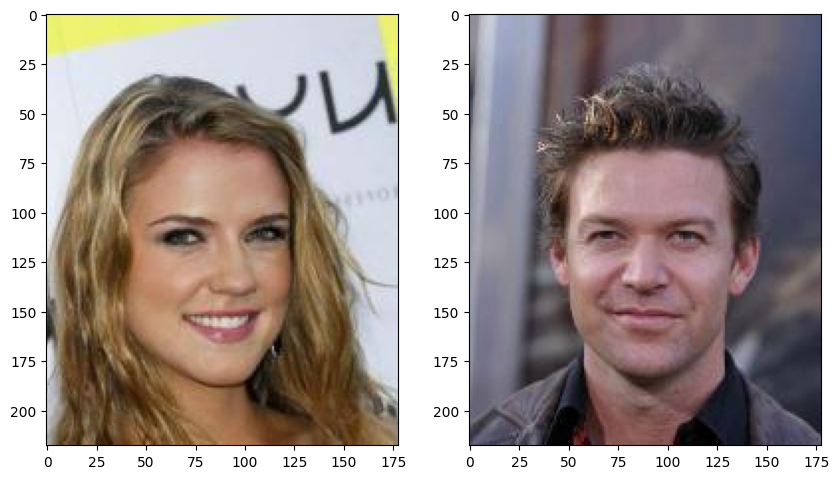
plt.figure(figsize=(10,10))

plt.subplot(121)

plt.imshow(flip\_img\_1)

plt.subplot(122)

plt.imshow(flip\_img\_2)



#transpose

transp\_img\_1=cv2.transpose(img\_1\_rgb,1) # transpose

transp\_img\_2=cv2.transpose(img\_2\_rgb,1) # transpose

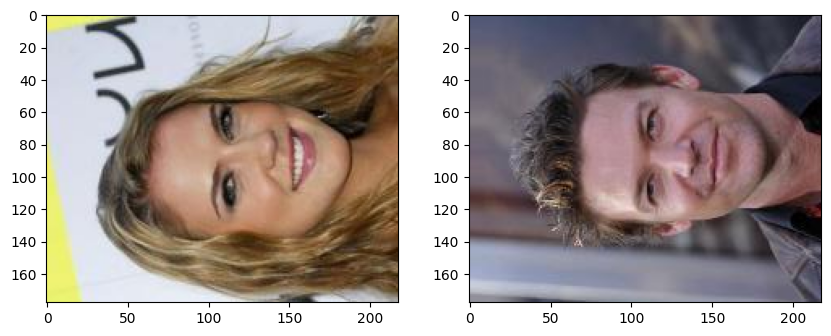
plt.figure(figsize=(10,10))

plt.subplot(121)

plt.imshow(transp\_img\_1)

plt.subplot(122)

plt.imshow(transp\_img\_2)



# **Minimum, maximum**

To find the min or max of a matrix, you can use minMaxLoc. This takes a single channel image from each file (it doesn't make much sense to take the max of a 3 channel image). So in the next code snippet you see a for loop, using python style image slicing, to look at each channel of the input image separately.

print("Female celebrity image min/max values and co-ordinates")

for i in range(0,3):

min\_value, max\_value, min\_location, max\_location=cv2.minMaxLoc(img\_1[:,:,i])

print("min {} is at {}, and max {} is at {}".format(min\_value, min\_location, max\_value, max\_location))

print("Male celebrity image min/max values and co-ordinates")

for i in range(0,3):

min\_value, max\_value, min\_location, max\_location=cv2.minMaxLoc(img\_2[:,:,i])

print("min {} is at {}, and max {} is at {}".format(min\_value, min\_location, max\_value, max\_location))

Female celebrity image min/max values and co-ordinates

min 0.0 is at (96, 51), and max 255.0 is at (158, 31)

min 0.0 is at (64, 109), and max 251.0 is at (0, 24)

min 4.0 is at (64, 109), and max 247.0 is at (0, 24)

Male celebrity image min/max values and co-ordinates

min 0.0 is at (98, 55), and max 222.0 is at (165, 0)

min 0.0 is at (98, 55), and max 216.0 is at (117, 91)

min 0.0 is at (127, 199), and max 247.0 is at (120, 99)

## Exracting Features in Visual Exploratory Data Analysis for visual model building

Features are image properties at locations that are used for computer based visual model development. Computer interprets the images based on these feature based Visual models. Some of the feature extraction techniques being attempted here are:

1. Making sense of images in presence of noise.
2. Extracting edges and Corners.
3. Understanding the impact of lighting and illumination artefacts.

#Conversion to Gray scale image needed for pre-processing

import numpy as np

import matplotlib.pyplot as plt

import skimage

import skimage.color as skic

import skimage.filters as skif

import skimage.data as skid

import skimage.util as sku

%matplotlib inline

fig, (ax1, ax2, ax3,ax4) = plt.subplots(1, 4, figsize=(10, 10))

ax1.imshow(img\_1\_rgb)

ax1.set\_title('Original Image\n Female celebrity')

ax1.set\_axis\_off()

ax2.imshow(skic.rgb2gray(img\_1), cmap ='gray')

ax2.set\_title('Gray scale image\n Female celebrity')

ax2.set\_axis\_off()

ax3.imshow(img\_2\_rgb)

ax3.set\_title('Original image\n Male celebrity')

ax3.set\_axis\_off()

ax4.imshow(skic.rgb2gray(img\_2),cmap = 'gray')

ax4.set\_title('Original image\n Male celebrity')

ax4.set\_axis\_off()



# We add Gaussian noise and denoise using denoise\_tv\_bregman approach

#for img\_1 and img\_2

img\_1\_n = sku.random\_noise(skic.rgb2gray(img\_1))

img\_1\_d = skimage.restoration.denoise\_tv\_bregman(img\_1\_n, 5.)

img\_2\_n = sku.random\_noise(skic.rgb2gray(img\_2))

img\_2\_d = skimage.restoration.denoise\_tv\_bregman(img\_2\_n, 5.)

fig, ((ax1, ax2,ax3), (ax4, ax5,ax6)) = plt.subplots(ncols=3,nrows=2,figsize=(10, 10))

ax1.imshow(img\_1\_rgb)

ax1.set\_title('Original Image\n Female celebrity')

ax1.set\_axis\_off()

ax2.imshow(img\_1\_n, cmap ='gray')

ax2.set\_title('Noisy image\n Female celebrity')

ax2.set\_axis\_off()

ax3.imshow(img\_1\_d, cmap = 'gray')

ax3.set\_title('Denoised image\n Female celebrity')

ax3.set\_axis\_off()

ax4.imshow(img\_2\_rgb)

ax4.set\_title('Original image\n Male celebrity')

ax4.set\_axis\_off()

ax5.imshow(img\_2\_n, cmap ='gray')

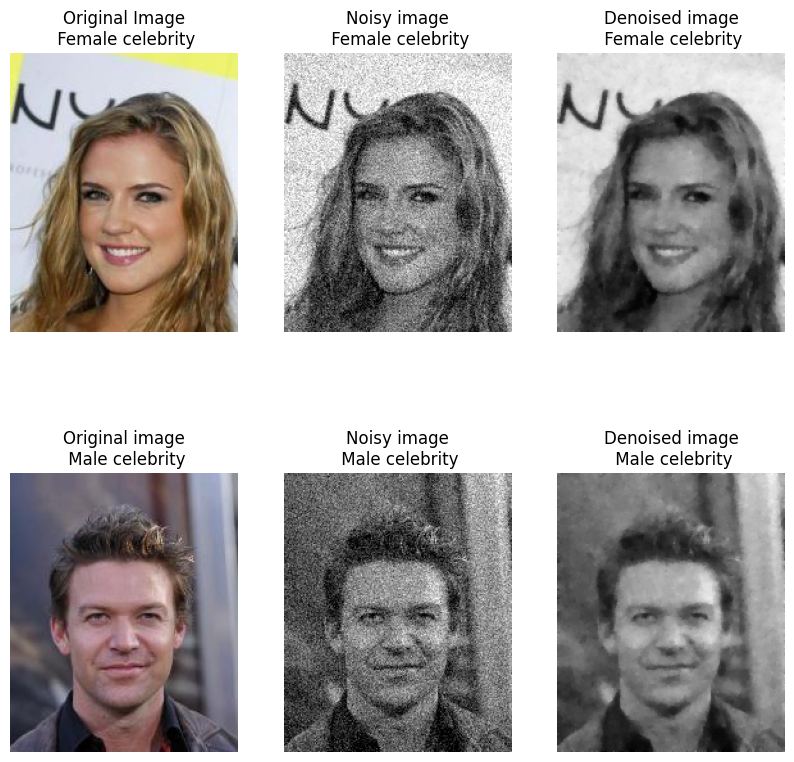
ax5.set\_title('Noisy image\n Male celebrity')

ax5.set\_axis\_off()

ax6.imshow(img\_2\_d, cmap = 'gray')

ax6.set\_title('Denoised image\n Male celebrity')

ax6.set\_axis\_off()



# **Noise reduction using Gaussian Blur**

Noise reduction usually involves blurring/smoothing an image using a Gaussian kernel. The width of the kernel (d) determines the amount of smoothing. Try with d=3 or d=6

d=3

img\_1\_blur3 = cv2.GaussianBlur(skic.rgb2gray(img\_1), (2\*d+1, 2\*d+1), -1)[d:-d,d:-d]

img\_2\_blur3 = cv2.GaussianBlur(skic.rgb2gray(img\_2), (2\*d+1, 2\*d+1), -1)[d:-d,d:-d]

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(ncols=2,nrows=2,figsize=(10, 10))

ax1.imshow(img\_1\_rgb)

ax1.set\_title('Original Image\n Female celebrity')

ax1.set\_axis\_off()

ax2.imshow(img\_1\_blur3, cmap ='gray')

ax2.set\_title('Blurred image\n Female celebrity')

ax2.set\_axis\_off()

ax3.imshow(img\_2\_rgb)

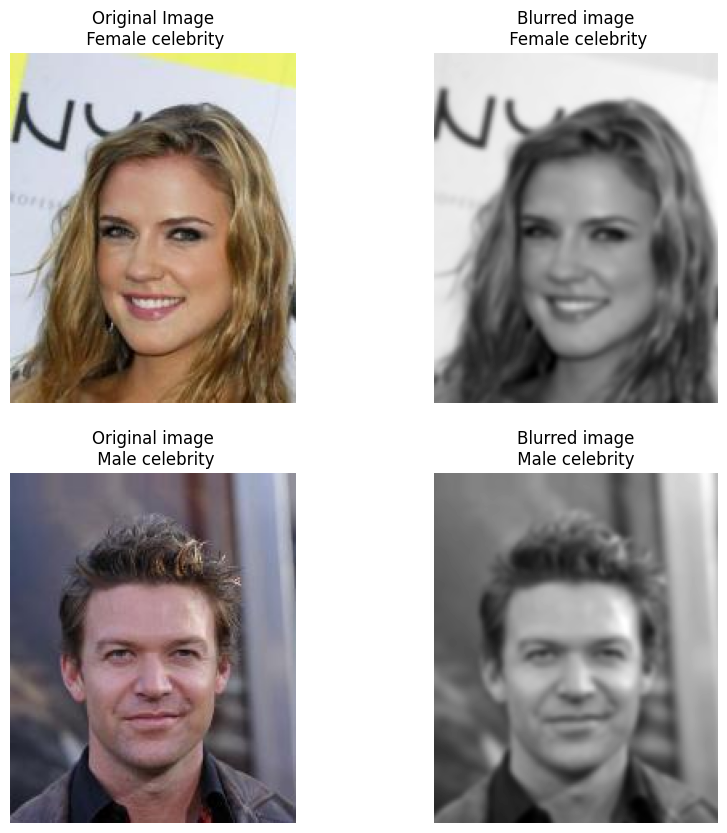
ax3.set\_title('Original image\n Male celebrity')

ax3.set\_axis\_off()

ax4.imshow(img\_2\_blur3, cmap = 'gray')

ax4.set\_title('Blurred image\n Male celebrity ')

ax4.set\_axis\_off()



# **Edge detection**

Edge is the final image processing technique for exploring the visual data.

Edge detection functions as a building block for building computer based detection models. Sobel's edge detector is one of the first truly successful edge detection (enhancement) technique and that involves convolution processing at its core.

#Sobel edge detector

#edge detector works on gray scale images

sobel\_img\_1=cv2.cvtColor(img\_1,cv2.COLOR\_BGR2GRAY)

sobel\_img\_2=cv2.cvtColor(img\_2,cv2.COLOR\_BGR2GRAY)

sobelx\_img\_1 = cv2.Sobel(sobel\_img\_1,cv2.CV\_64F,1,0,ksize=9)

sobely\_img\_1 = cv2.Sobel(sobel\_img\_1,cv2.CV\_64F,0,1,ksize=9)

sobelx\_img\_2 = cv2.Sobel(sobel\_img\_2,cv2.CV\_64F,1,0,ksize=9)

sobely\_img\_2 = cv2.Sobel(sobel\_img\_2,cv2.CV\_64F,0,1,ksize=9)

plt.figure(figsize=(10,10))

plt.subplot(221)

plt.imshow(sobelx\_img\_1, cmap = 'gray')

plt.subplot(222)

plt.imshow(sobely\_img\_1, cmap = 'gray')

plt.subplot(223)

plt.imshow(sobelx\_img\_2, cmap = 'gray')

plt.subplot(224)

plt.imshow(sobely\_img\_2, cmap = 'gray')

文本

描述已自动生成

# **Canny edge detection**

Canny edge detection is another winnning technique - it takes two thresholds. The first one determines how likely Canny is to find an edge, and the second determines how likely it is to follow that edge once it's found. Investigate the effect of these thresholds by altering the values below.

th1=30

th2=60

# Canny recommends threshold 2 is 3 times threshold 1

# you could try experimenting with this...

d=3

# gaussian blur

# this takes pixels in edgeresult where edge non-zero and colours them bright green

edgeresult\_1=img\_1.copy()

edgeresult\_1 = cv2.GaussianBlur(edgeresult\_1, (2\*d+1, 2\*d+1), -1)[d:-d,d:-d]

gray\_1 = cv2.cvtColor(edgeresult\_1, cv2.COLOR\_BGR2GRAY)

edge\_1 = cv2.Canny(gray\_1, th1, th2)

edgeresult\_1[edge\_1 != 0] = (0, 255, 0)

edgeresult\_2=img\_2.copy()

edgeresult\_2 = cv2.GaussianBlur(edgeresult\_2, (2\*d+1, 2\*d+1), -1)[d:-d,d:-d]

gray\_2 = cv2.cvtColor(edgeresult\_2, cv2.COLOR\_BGR2GRAY)

edge\_2 = cv2.Canny(gray\_2, th1, th2)

edgeresult\_2[edge\_2 != 0] = (0, 255, 0)

plt.figure(figsize=(10,10))

plt.subplot(221)

plt.imshow(cv2.cvtColor(img\_1, cv2.COLOR\_BGR2RGB))

plt.subplot(222)

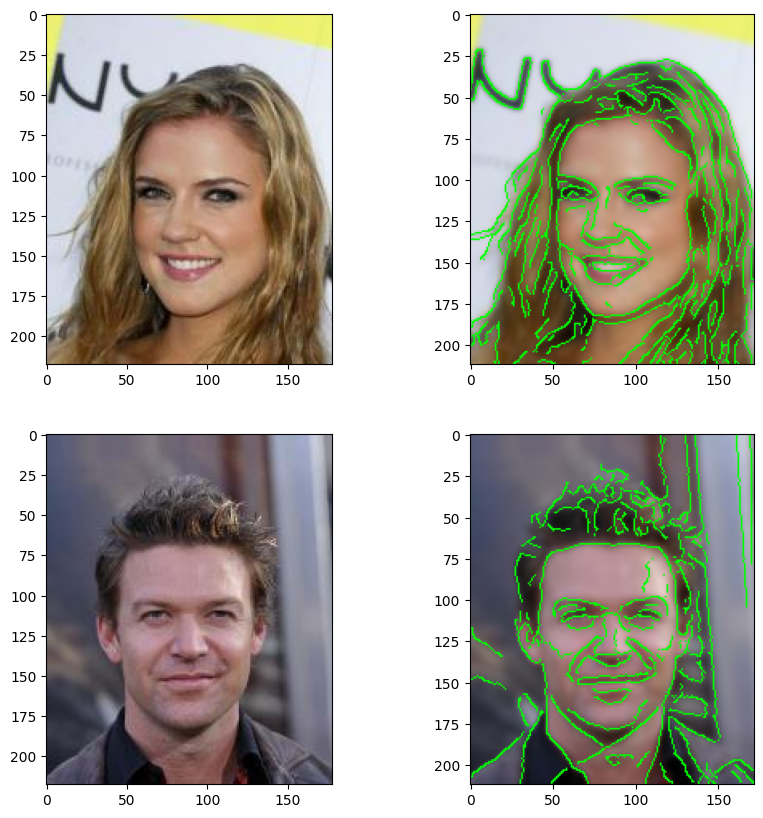
plt.imshow(cv2.cvtColor(edgeresult\_1, cv2.COLOR\_BGR2RGB))

plt.subplot(223)

plt.imshow(cv2.cvtColor(img\_2, cv2.COLOR\_BGR2RGB))

plt.subplot(224)

plt.imshow(cv2.cvtColor(edgeresult\_2, cv2.COLOR\_BGR2RGB))



# **Corner detectors**

If you think of edges as being lines, then corners are an obvious choice for features as they represent the intersection of two lines. One of the earlier corner detectors was introduced by Harris, and it is still a very effective corner detector that gets used quite a lot: it's reliable and it's fast. There's a tutorial explaining how Harris works on the OpenCV site here: [https://docs.opencv.org/3.0-beta/doc/py\_tutorials/py\_tutorials.html](https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fdocs.opencv.org%2F3.0-beta%2Fdoc%2Fpy_tutorials%2Fpy_tutorials.html)

# detecting corners for image\_1

harris\_1=img\_1.copy()

# greyscale it

gray = cv2.cvtColor(harris\_1,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

blocksize=4

kernel\_size=3 # sobel kernel: must be odd and fairly small

# run the harris corner detector

dst = cv2.cornerHarris(gray,blocksize,kernel\_size,0.05) # parameters are blocksize, Sobel parameter and Harris threshold

# result is dilated for marking the corners, this is visualisation related and just makes them bigger

dst = cv2.dilate(dst,None)

# we then plot these on the input image for visualisation purposes, using bright red

harris\_1[dst>0.01\*dst.max()]=[0,0,255]

# detecting corners for image\_2

harris\_2=img\_2.copy()

# greyscale it

gray = cv2.cvtColor(harris\_2,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

blocksize=4 #

kernel\_size=3 # sobel kernel: must be odd and fairly small

# run the harris corner detector

dst = cv2.cornerHarris(gray,blocksize,kernel\_size,0.05) # parameters are blocksize, Sobel parameter and Harris threshold

# result is dilated for marking the corners, this is visualisation related and just makes them bigger

dst = cv2.dilate(dst,None)

# we then plot these on the input image for visualisation purposes, using bright red

harris\_2[dst>0.01\*dst.max()]=[0,0,255]

plt.figure(figsize=(10,10))

plt.subplot(221)

plt.imshow(cv2.cvtColor(img\_1, cv2.COLOR\_BGR2RGB))

plt.subplot(222)

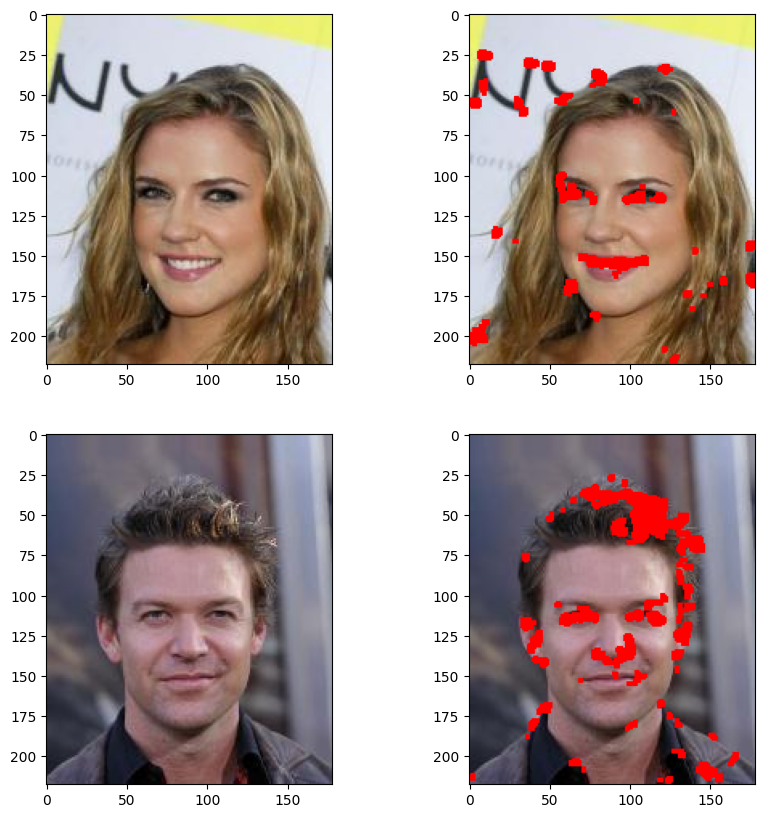
plt.imshow(cv2.cvtColor(harris\_1, cv2.COLOR\_BGR2RGB))

plt.subplot(223)

plt.imshow(cv2.cvtColor(img\_2, cv2.COLOR\_BGR2RGB))

plt.subplot(224)

plt.imshow(cv2.cvtColor(harris\_2, cv2.COLOR\_BGR2RGB))



# **Understanding illumination and lighting artefacts by examining the camera effects/exposure of an image**

import skimage.exposure as skie

%matplotlib inline

def show(img):

# Display the image.

fig, (ax1, ax2) = plt.subplots(1, 2,

figsize=(12, 3))

ax1.imshow(img, cmap=plt.cm.gray)

ax1.set\_axis\_off()

# Display the histogram.

ax2.hist(img.ravel(), lw=0, bins=256)

ax2.set\_xlim(0, img.max())

ax2.set\_yticks([])

plt.show()

show(img\_1\_rgb)

show(skie.equalize\_adapthist(img\_1\_rgb))

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

show(img\_2\_rgb)

show(skie.equalize\_adapthist(img\_2\_rgb))

图表, 直方图

描述已自动生成图表, 直方图

描述已自动生成

img = skic.rgb2gray(img\_1)

sobimg\_nheq= skif.sobel(img)

show(sobimg\_nheq)

img = skic.rgb2gray(skie.equalize\_adapthist(img\_1))

sobimg\_heq = skif.sobel(img)

show(sobimg\_heq)

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

img = skic.rgb2gray(img\_2)

sobimg\_nheq= skif.sobel(img)

show(sobimg\_nheq)

img = skic.rgb2gray(skie.equalize\_adapthist(img\_2))

sobimg\_heq = skif.sobel(img)

show(sobimg\_heq)

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

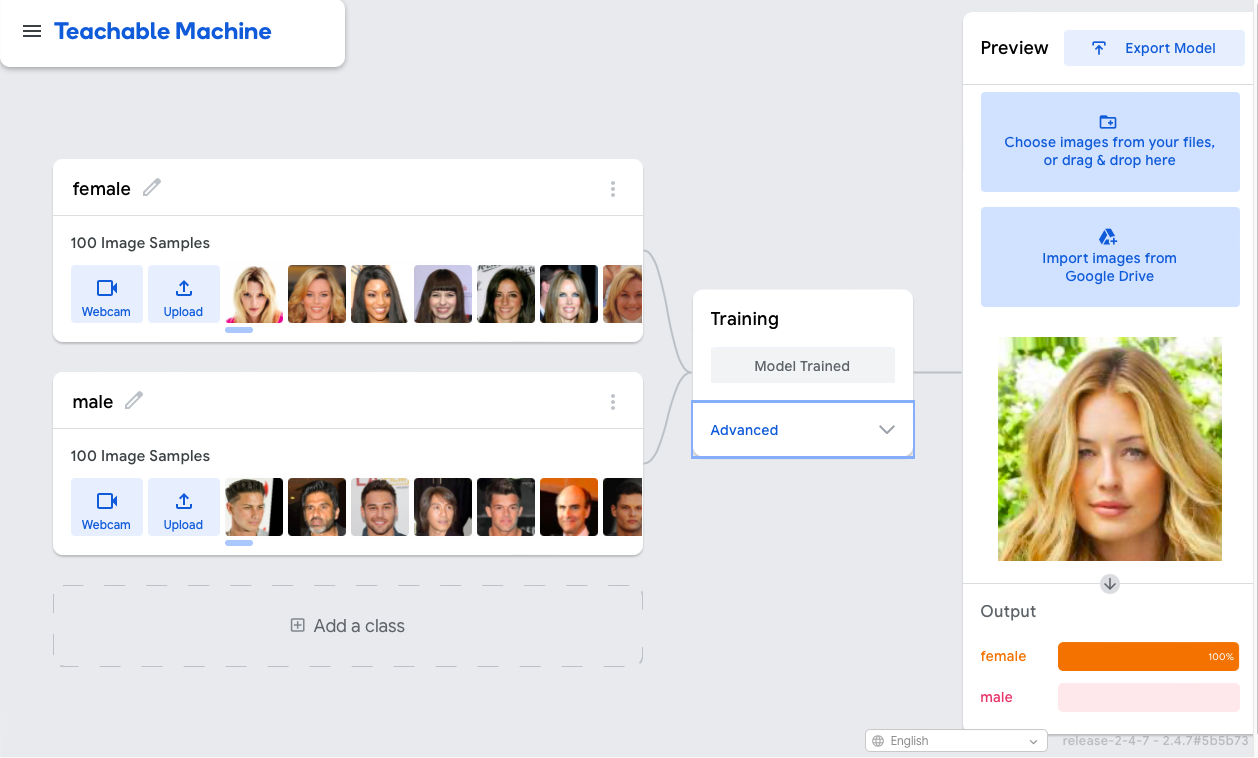
## Stage 2: PDA (Predictive Data Analytics)

## Stage 2B PDA with [Teachable Machine withGoogle](https://teachablemachine.withgoogle.com/)

1. Go through the following tutorial :

<https://medium.com/gopenai/teachable-machine-a-simple-way-to-create-your-own-machine-learning-models-45eb490957b7>

1. Upload the images for your capstone project, from your computer or your google drive into the Teachable Machine WithGoogle dasbboard
2. Build the model as per the instructions in (1) above and save the model in google drive or download the model to your machine.
3. Display the screenshot of the trained model from the dashboard and under the hood performance metrics:



图表

描述已自动生成图表, 折线图, 散点图

描述已自动生成

1. If you downloaded the teachable model, provide the name of the file and include it in your project submission

图片包含 文本

描述已自动生成

## Stage 3A Deployment/Implemenation with [Teachable Machine withGoogle](https://teachablemachine.withgoogle.com/)

* Click on Download my model and save the converted\_keras.zip in your google drive or Pycharm project folder
* Copy the model deployment Python code and paste it in either Google colab or in Pycharm Project folder.
* Unzip the converter\_keras.zip and change the path to appropriate model file (.h5 file), label file (labels.txt) and test image file (.jpg/.png) for test image in the model deployment Python code shown below:

from keras.models import load\_model # TensorFlow is required for Keras to work

from PIL import Image, ImageOps # Install pillow instead of PIL

import numpy as np

# Disable scientific notation for clarity

np.set\_printoptions(suppress=True)

# Load the model

model = load\_model("/content/drive/MyDrive/Gender Classification/converted\_keras/keras\_model.h5", compile=False)

# Load the labels

class\_names = open("/content/drive/MyDrive/Gender Classification/converted\_keras/labels.txt", "r").readlines()

# Create the array of the right shape to feed into the keras model

# The 'length' or number of images you can put into the array is

# determined by the first position in the shape tuple, in this case 1

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image

image = Image.open("/content/drive/MyDrive/Gender Classification/dataset/train/female/000093.jpg").convert("RGB")

# resizing the image to be at least 224x224 and then cropping from the center

size = (224, 224)

image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# turn the image into a numpy array

image\_array = np.asarray(image)

# Normalize the image

normalized\_image\_array = (image\_array.astype(np.float32) / 127.5) - 1

# Load the image into the array

data[0] = normalized\_image\_array

# Predicts the model

prediction = model.predict(data)

index = np.argmax(prediction)

class\_name = class\_names[index]

confidence\_score = prediction[0][index]

# Print prediction and confidence score

print("Class:", class\_name[2:], end="")

print("Confidence Score:", confidence\_score)

1/1 [==============================] - 2s 2s/step

Class: female

Confidence Score: 0.9999963

from warnings import filterwarnings

import tensorflow as tf

from tensorflow import io

from tensorflow import image

from matplotlib import pyplot as plt

filterwarnings("ignore")

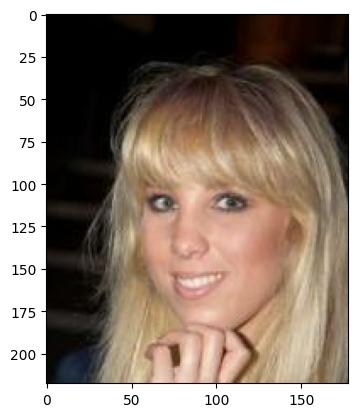
tf\_img = io.read\_file("/content/drive/MyDrive/Gender Classification/dataset/validation/female/180030.jpg")

tf\_img = image.decode\_png(tf\_img, channels=3)

print(tf\_img.dtype)

plt.imshow(tf\_img)

# plt.show()



## Stage 3B Deployment/Implemenation with TkInter

from tkinter import filedialog  
from tkinter import \*  
import tkinter as tk  
import keras  
from PIL import ImageTk, Image, ImageOps *# Install pillow instead of PIL*from keras.models import load\_model *# TensorFlow is required for Keras to work*import numpy as np  
  
*#load the trained model*model = load\_model('keras\_model.h5')  
*# Load the labels*label\_path='labels.txt'  
class\_names = open(label\_path, "r").readlines()  
  
test\_image\_path = '180117.jpg'  
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)  
  
*#initialize GUI*top=tk.Tk()  
top.geometry('800x600')  
top.title('gender classification')  
top.configure(background='#CDCDCD')  
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))  
output\_image = Label(top)  
  
def classify(test\_image\_path):  
 global label\_packed  
 disp\_string= ''  
 image = Image.open(test\_image\_path).convert("RGB")  
  
 *# resizing the image to be at least 224x224 and then cropping from the center* size = (224, 224)  
 image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)  
  
 *# turn the image into a numpy array* image\_array = np.asarray(image)  
  
 *# Normalize the image* normalized\_image\_array = (image\_array.astype(np.float32) / 127.5) - 1  
  
 *# Load the image into the array* data[0] = normalized\_image\_array  
  
 *# Predicts the model* prediction = model.predict(data)  
 index = np.argmax(prediction)  
 class\_name = class\_names[index]  
 confidence\_score = prediction[0][index]  
  
 *# Display prediction and confidence score* disp\_string+= "\nClass:"+ str(class\_name[2:])  
 disp\_string+= "\nConfidence Score:"+ str(confidence\_score)  
  
 *#label.configure(foreground='#011638', text=class\_name)* label.configure(foreground='#011638', text=disp\_string)  
  
def show\_classify\_button(file\_path):  
 classify\_b=Button(top,text="Classify Image",command=lambda: classify(file\_path),padx=10,pady=5)  
 classify\_b.configure(background='#364156', foreground='black',font=('arial',10,'bold'))  
 classify\_b.place(relx=0.79,rely=0.46)  
  
def upload\_image():  
 try:  
 file\_path=filedialog.askopenfilename()  
 uploaded=Image.open(file\_path)  
 uploaded.thumbnail(((top.winfo\_width()/2.25),(top.winfo\_height()/2.25)))  
 im=ImageTk.PhotoImage(uploaded)  
 output\_image.configure(image=im)  
 output\_image.image=im  
 label.configure(text='')  
 show\_classify\_button(file\_path)  
 except:  
 pass  
  
upload=Button(top,text="Upload an image",command=upload\_image,padx=10,pady=5)  
upload.configure(background='#364156', foreground='black',font=('arial',10,'bold'))  
upload.pack(side=BOTTOM,pady=50)  
output\_image.pack(side=BOTTOM,expand=True)  
label.pack(side=BOTTOM,expand=True)  
heading = Label(top, text="Gender Classifier",pady=20, font=('arial',20,'bold'))  
heading.configure(background='#CDCDCD',foreground='#364156')  
heading.pack()  
top.mainloop()

图形用户界面, 应用程序, Teams

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

## Conclusions

This report presents the work done towards the ST1 capstone project for design, development, implementation and deployment of data driven gender classifier software app using Python using a comprehensive exploratory data analysis in stage 1, followed predictive model development in stage 2 using Teachable Machine with Google, and model deployment and implementation of app in Tkinter. As can be seen from model performance, the confidence of gender classification from just the images is close to 99.9 % , and can be of immense use to identify the gender of a person or a photo.

## References

1. (<https://www.kaggle.com/datasets/ashishjangra27/gender-recognition-200k-images-celeba> )
2. (<https://www.kaggle.com/datasets/jessicali9530/celeba-dataset> )

## Appendix 1: Log Book

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Planned  Activities | Tasks  Completed | Problems  Faced | Further Comments |
| Week 1-2 | Chanllenge1 | Completed | No | No |
| Week 3 | Chanllenge2,3,4,5 | Completed | No | No |
| Week 4 | Chanllenge6 | Completed | No | No |
| Week 5 | Chanllenge7,8 | Completed | No | No |
| Week 6 | Chanllenge9,10 | Completed | No | No |
| Week 7 | Chanllenge11 | Completed | No | No |
| Week 9 | Chanllenge12 | Completed | No | No |
| Week 10 | Chanllenge13 | Completed | No | No |
| Week 12 | Cap-stage1,2,3 | Completed | No | No |
| Week 13 | Cap-report,slides | Completed | No | No |