COMPUTER SCIENCE AND ENGINEERING(CSE), ISLAMIC UNIVERSITY OF TECHNOLOGY(IUT)

Gender Classification Application using Machine Learning

CSE 4710: MACHINE LEARNING LAB

Md Sultanul Islam Ovi 160041036
Nafil Mahmud 160041031
Abida Taskin 160041035
Novera Tasnuba Aura 160041041
Eksan Ahmed 160041067

I. Introduction

A gender classification system uses the face of a person from a given image to tell the gender (male/female) of the given person. A successful gender classification approach can boost the performance of many other applications including face recognition and smart human-computer interface.

A. *Problem Statement*

In this project, we design and implement an application that can successfully classify males and females from an image or a video by using Machine Learning Approach. Here, we use a support vector machine (SVM), one of the most popular and efficient machine learning algorithms to classify gender.

B. *Application Area*

This project falls under the area of image processing and computer vision.

Image processing is a method to perform some operations on an image, to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or features associated with that image. To get an optimized workflow and to avoid losing time, it is important to process images after the capture, in a post-processing step.

Computer vision is an interdisciplinary scientific field that deals with how computers can gain a high-level understanding of digital images or videos. It is the process of understanding digital images and videos using computers. It seeks to automate tasks that human vision can achieve. This involves methods of acquiring, processing, analyzing, and understanding digital images, and extraction of data from the real world to produce information.

C. *Challenges*

There were a few challenges we needed to overcome to complete our tasks and optimize our classifier to the best we can

- Converting height and width all the images to an equal size.

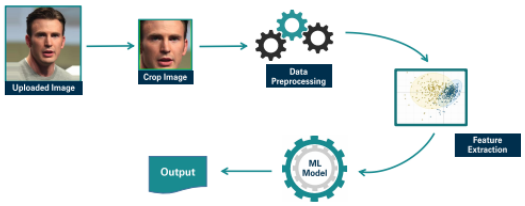


Fig. 1. Project Overview

- Handling missing values.
- Choosing an appropriate classifier algorithm
- Reducing dimensions to optimize.

D. *Motivation*

Face recognition is one of the greatest applications of machine learning. We have reached a good saturated state when it comes to recognizing a face. Now we can apply these methods into different applications like gender classification, age prediction, emotion prediction, etc. Here in our project we try to classify people into male and female by their face and using machine learning. We focus on how to approach the problem, how to process those images, how to tune our hyperparameters to make our classifier efficient.

II. Background Study

A. *Project Overview*

Our project is divided into six modules.

- In our first module, we load our data-set. We convert those images into gray-scale. Then we detect faces from those images using haar cascade classifier. Then we crop those faces and save them.
- In our second module, we load our cropped-faces data-set. We analyze those images and remove images that are too small and make the rest of the images to the same size.
- In our third module, we pre-process our data. We handle missing values and perform normalization.

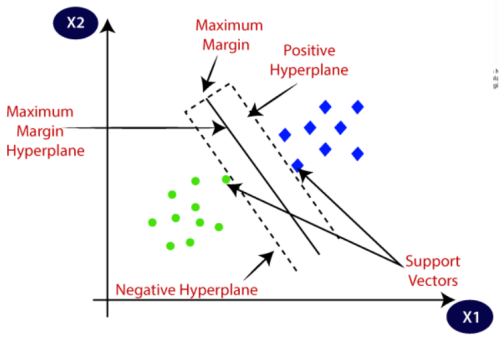


Fig. 2. SVM

- In our fourth module, we generate eigen images from our data-set. We perform a PCA analysis to reduce dimensions.
- In our fifth module, we train our SVM model. We perform different Model Evaluation and Hyper Parameter Tuning
- In our sixth module, we create the pipeline model which loads all the modules and generates the output

B. *Data collection and feature analysis*

We have collected our data from this website [1]. This data-set contains 7000 images of females and 7000 images of males. These images are of different sizes. We need to shrink some of the images and also enlarge some of them to get all of them to the same size. Then we crop out the faces in the shape of rectangular and store them.

C. *Classification model used*

Here for our project, we have used the SVM model. SVM is a supervised machine learning algorithm that can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs. Support vectors are data points that are closer to the hyper-plane and influence the position and orientation of the hyper-plane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyper-plane. These are the points that help us build our SVM.

D. *Explanation of the parameters or hyperparameters*

There are mainly 3 hyperparameters of SVM classifier.

- The main hyperparameter of the SVM is the kernel. It maps the observations into some feature space. Ideally, the observations are more easily (linearly) separable after this transformation. There are multiple standard kernels for these transformations, e.g. the linear kernel, the polynomial kernel, and the radial kernel. The choice of the kernel and their hyperparameters affect greatly the separability of the classes (in classification) and the performance of the algorithm.
- The Penalty Parameter(C): It tells the algorithm how much you care about misclassified points. SVMs, in general, seek to find the maximum margin hyperplane. A high value for C tells the algorithm that you care more about classifying all of the training points correctly than leaving wiggle room for future data.
- Gamma: It is used when we use the Gaussian RBF kernel. if you use linear or polynomial kernel then you do not need gamma only you need C hyperparameter. Somewhere it is also used as sigma. Gamma is a hyperparameter that we have to set before training the model. Gamma decides how much curvature we want in a decision boundary. Gamma high means more curvature. Gamma low means less curvature.

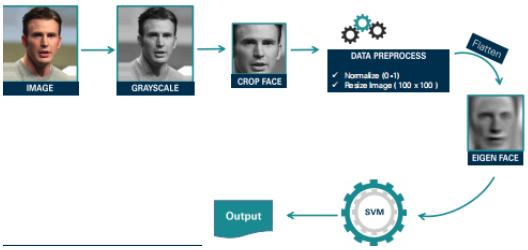


Fig. 3. Detailed Project Overview

III. Implementation

A. *Overview of the experiment*

We perform our implementation on anaconda distribution of the Python. We use the OpenCV library to detect faces. The other libraries that we used are numpy, scipy, matplotlib, pandas, jupyter, sklearn.

We use image processing techniques in OpenCV and the concepts behind the images. We also do the necessary image analysis and required prepossessing steps for images. For the preprocess images, we will extract features from the images, ie. computing Eigen images using principal component analysis. With Eigen images, we will train the Machine learning model and also learn to test our model before deploying, to get the best results from the model we will tune with the Grid search method for best hyperparameters. After that, our machine learning model is ready to deploy.

B. *Feature engineering*

We have analyzed the dataset and removed some of the images that are very small in size. Then we resize the rest of the images to 100X100 format.

Then we searched for missing values and removed them by setting them equal to 0.

Then we normalized the whole dataset to get the values in between 0 to 1

C. *Training and Testing set generation*

We split our dataset into the training set and testing set in 0.8 ratio in random order meaning we train our model with 80% of the data and we test our model with the remaining 20% of the data.

D. *Running the classifier*

We run our SVM classifier using rbf kernel where we use 1 as C value and 0.01 as gamma value. In this case, we get an output of 82.89% which is good enough but we can still do some tuning to the hyperparameters to get better results.

IV. Result Analysis

A. *Initial Analysis*

- With our current model, we get 82.89% accuracy on our test dataset.
- We have generated a confusion matrix that summarizes the output. Here the model can detect 323 males out of 442 males and 570 females out of 651 females.
- We have generated the classification report which gives us precision, recall, f1-score, support scores.
- We have generated the kappa score which is 0.63
- We have generated ROC and AUC where AUC score is 0.89

B. *Hyperparameter Tuning*

To get the best results from the SVM model we tune the Hyperparameters with Grid search method for best hyperparameters. After Grid search method we found that we do not need to change C value and kernel. We only need to change the gamma value to 0.03

C. *Model Evaluation*

- After tuning, accuracy increased from 82.89% to 84.26%.
- In the confusion matrix, we can see that the model now can detect 331 males out of 442 males and 590 females out of 651 females.
- After tuning, we have seen improvement in precision, recall, f1-score, support scores in the classification report.
- After tuning, the kappa score has increased to 0.66 from 0.63
- After tuning, the AUC score has increased to 0.91 from 0.89

V. Conclusion

This project demonstrates how we can apply machine learning to classify gender from images and videos. Although the system is not perfect it is quite capable. We can further improve the performance of the system by using different classifiers and also by introducing ensemble methods. This will improve the system's performance. This model can be modified or improved to predict age or sentiment from images. And we can further improve this model to work with real-time videos.

Required Libraries

```
import numpy as np
import pandas as pd
import cv2
import sklearn
import pickle
import matplotlib.pyplot as plt
import matplotlib.image as mat_image
%matplotlib inline
from PIL import Image
from glob import glob
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
# load haar cascade classifier
haar = cv2.CascadeClassifier('./model/haarcascade_frontalface_default.xml')

pd.options.mode.chained_assignment = None  # default='warn'
```

Fig. 4. Required Libraries

```
#loading images from dataset
femalepath = glob('./data/female/*.jpg')
malepath = glob('./data/male/*.jpg')
a = len(femalepath)
b = len(malepath)
print(a,b)

# let us consider one image
path = femalepath[0]
img = cv2.imread(path)

plt.imshow(img)
plt.show()
```

```
7000 7000
```

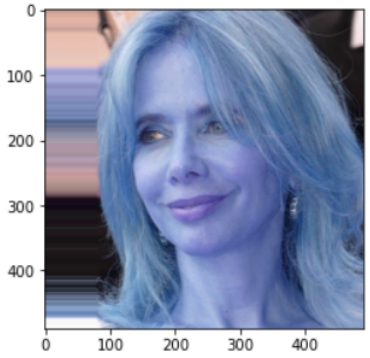


Fig. 5. Loading Images

References

[1] https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

Appendix

Code Snippets

```
# convert image into grayscale
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
print(gray.shape)
plt.imshow(gray,cmap='gray')
```

```
(489, 490)
```

```
<matplotlib.image.AxesImage at 0x17b4d3ffbe0>
```

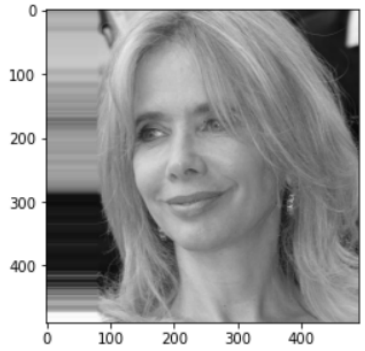


Fig. 6. Converting image into Grayscale

```
faces = haar.detectMultiScale(gray,1.5,5)
print(faces)

for x,y,w,h in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,255),5)
plt.imshow(img)
```

```
[[ 86  86 273 273]]
```

```
<matplotlib.image.AxesImage at 0x17b4d7714c0>
```

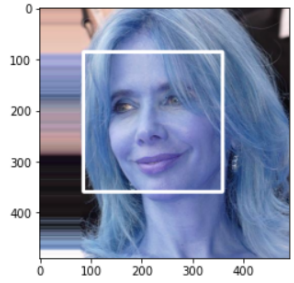
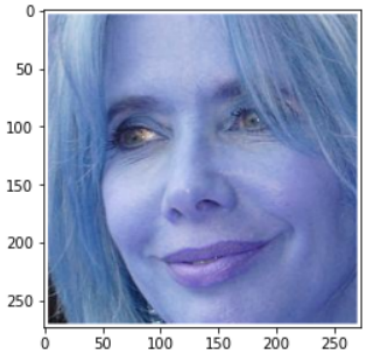


Fig. 7. Face Detection

```
# crop the image
crop_img = img[y:y+h,x:x+h]
plt.imshow(crop_img)
```

```
<matplotlib.image.AxesImage at 0x17b5c3bce80>
```




```
# save the image
cv2.imwrite('f_01.png',crop_img)
```

```
True
```

Fig. 8. Cropping the image

```
# Apply to all the images
def extract_images(path,gender,i):
    img = cv2.imread(path)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = haar.detectMultiScale(gray,1.5,5)
    for x,y,w,h in faces:
        roi = img[y:y+h,x:x+w]
        if gender == 'male':
            cv2.imwrite('./data/crop/male_crop/{}_{}.png'.format(gender,i),roi)
        else:
            cv2.imwrite('./data/crop/female_crop/{}_{}.png'.format(gender,i),roi)
```

```
extract_images(femalepath[0],'female',1)
```

```
for i,path in enumerate(femalepath):
    try:

        extract_images(path,'female',i)
        print('INFO: {}/{} processed sucessfully'.format(i,len(femalepath)))

    except:
        print('INFO: {}/{} cannot be processed'.format(i,len(femalepath)))
```

```
for i,path in enumerate(malepath):
    try:

        extract_images(path,'male',i)
        print('INFO: {}/{} processed sucessfully'.format(i,len(malepath)))

    except:
        print('INFO: {}/{} cannot be processed'.format(i,len(malepath)))
```

Fig. 9. Applying to all the images

```
def resize_img(path_to_resize):
    try:
        # step - 1: read image
        img = cv2.imread(path_to_resize)
        # step - 2: convert into grayscale
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        # step -3: resize into 100 x 100 array
        size = gray.shape[0]
        if size >= 100: #shrink
            gray_re = cv2.resize(gray,(100,100),cv2.INTER_AREA) # SHRINK
        else: # enLarge
            gray_re = cv2.resize(gray,(100,100),cv2.INTER_CUBIC) # ENLARGE
        # step -4: Flatten Image (1x10,000)
        return gray_re.flatten()
    except:
        return None
```

Fig. 10. Resizing the images

```
df_new['gender'] = df_new['path'].apply(gender)

# structuring function
df_new['structure_data'] = df_new['path'].apply(resize_img)

# copy and expand their columns
df1 = df_new['structure_data'].apply(pd.Series)
df1 = df1.astype(float)

df2 = pd.concat((df_new['gender'],df1),axis=1)

plt.imshow(df2.loc[0][1:].values.reshape(100,100).astype('int'),cmap='gray')
plt.title("Label: "+df2.loc[0]['gender'])
plt.show()
```

Fig. 11. Providing structure to dataframes

```
# removing missing values
df.dropna(axis=0,inplace=True)
df.isnull().sum()
```

```
gender    0
0         0
1         0
2         0
3         0
         ..
9995      0
9996      0
9997      0
9998      0
9999      0
Length: 10001, dtype: int64
```

Fig. 12. Removing missing values

```
Xnorm = X / X.max()
Xnorm
```

```
array([[0.7372549 , 0.70588235, 0.72156863, ..., 0.47058824
        0.43529412],
       [0.1254902 , 0.09411765, 0.1254902 , ..., 0.14117647
        0.06666667],
       [0.08627451, 0.11764706, 0.15294118, ..., 0.74509804
        0.69411765],
       ...,
       [0.09803922, 0.09803922, 0.10196078, ..., 0.11764706
        0.13333333],
       [0.08235294, 0.10588235, 0.12156863, ..., 0.07843137
        0.09803922],
       [0.01568627, 0.01176471, 0.00784314, ..., 0.35294118
        0.36470588]])
```

```
# female = 1, male = 0
y_norm = np.where(y=='female',1,0)
```

```
# save x and y in numpy zip
np.savez('./data/data_10000_norm.npz',Xnorm,y_norm)
```

Fig. 13. Data Normalization

```
# load numpy zip
data = np.load('./data/data_10000_norm.npz')

X  = data['arr_0'] # independent features
y = data['arr_1'] # target value

print(X.shape, y.shape)

X1 = X - X.mean(axis=0)
pca = PCA(n_components=None,whiten=True,svd_solver='auto')
x_pca = pca.fit_transform(X1)
print(x_pca.shape)

eigen_ratio = pca.explained_variance_ratio_
eigen_ratio_cum = np.cumsum(eigen_ratio)
```

Fig. 14. PCA part-01

```
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(eigen_ratio[:200],'r>--')
plt.xlabel('no. of components')
plt.ylabel('Explanied Variance ratio')

plt.subplot(1,2,2)
plt.plot(eigen_ratio_cum[:200],'r>--')
plt.xlabel('no. of components')
plt.ylabel('Cumulative Explanied Variance ratio')

plt.show()
```

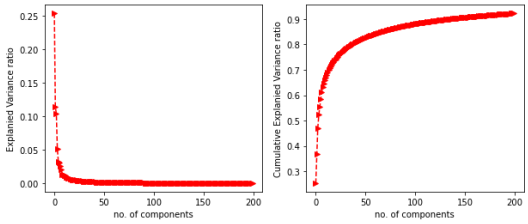


Fig. 15. PCA part-02

```
pca_50 = PCA(n_components=50,whiten=True,svd_solver='auto')
x_pca_50 = pca_50.fit_transform(X1)
```

```
x_pca_50.shape
```

```
(5463, 50)
```

```
# saving pca
import pickle
```

```
pickle.dump(pca_50,open('./model/pca_50.pickle','wb'))
```

```
# consider 50 compnent and inverse transform
x_pca_inv = pca_50.inverse_transform(x_pca_50)
```

```
x_pca_inv.shape
```

```
(5463, 10000)
```

Fig. 16. PCA part-03

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify=y)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((4370, 50), (1093, 50), (4370,), (1093,))
```

Fig. 17. Train Test split

```
from sklearn.svm import SVC
```

```
model = SVC(C=1.0,kernel='rbf',
    gamma=0.01,probability=True)
```

```
model.fit(x_train,y_train)
print('model trained sucessfully')
```

```
model trained sucessfully
```

```
# score
model.score(x_train,y_train)
```

```
0.8578947368421053
```

```
# score
model.score(x_test,y_test)
```

```
0.8289112534309241
```

Fig. 18. Training the ML Model

```
from sklearn import metrics
```

```
y_pred = model.predict(x_test)
y_prob = model.predict_proba(x_test) # proability
```

```
cm = metrics.confusion_matrix(y_test,y_pred)
cm = np.concatenate((cm,cm.sum(axis=0).reshape(1,-1)),
                    axis=0)
cm = np.concatenate((cm,cm.sum(axis=1).reshape(-1,1)),
                    axis=1)
plt.imshow(cm)
for i in range(3):
    for j in range(3):
        plt.text(j,i,'%d'%cm[i,j])

plt.xticks([0,1])
plt.yticks([0,1])
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```

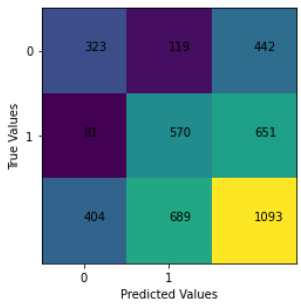


Fig. 19. Confusion Matrix

```
# classification report
cr = metrics.classification_report(y_test,y_pred,
target_names=['male','female'],output_dict=True)
pd.DataFrame(cr).T
```

	precision	recall	f1-score	support
male	0.824427	0.733032	0.776048	442.000000
female	0.831429	0.894009	0.861584	651.000000
accuracy	0.828911	0.828911	0.828911	0.828911
macro avg	0.827928	0.813520	0.818816	1093.000000
weighted avg	0.828597	0.828911	0.826994	1093.000000

Fig. 20. Classification Report

```
# kappa
metrics.cohen_kappa_score(y_test,y_pred)
```

```
0.6384015370380527
```

Fig. 21. Kappa Score

```
# roc for female
fpr,tpr,thresh = metrics.roc_curve(y_test,y_prob[:,1])
auc_s = metrics.auc(fpr,tpr)
plt.figure(figsize=(7,5))
plt.plot(fpr,tpr,'-.')
plt.plot([0,1],[0,1],'b--')
for i in range(0,len(thresh),20):
    plt.plot(fpr[i],tpr[i],'^')
    plt.text(fpr[i],tpr[i],"%0.2f"%thresh[i])

plt.legend(['AUC Score = %0.2f'%auc_s])

plt.xlabel('False Positve Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characterstics')
plt.show()
```

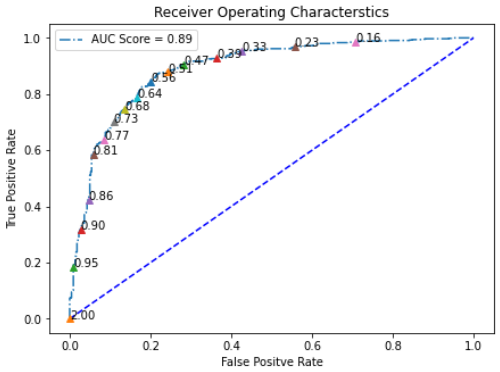


Fig. 22. ROC and AUC

```
model_tune = SVC()
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'C':[1,10],
             'kernel':['rbf','poly'],
             'gamma':[0.03,0.04,0.035],
             'coef0':[0,1],
             }
model_grid = GridSearchCV(model_tune,
param_grid=param_grid,scoring='accuracy',cv=10,verbose=2)
model_grid.fit(X,y)
```

```
Fitting 10 folds for each of 24 candidates, totalling 240 fits
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 co
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.3s re
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
[CV] ............ C=1, coef0=0, gamma=0.03, kernel=rbf, total=
[CV] C=1, coef0=0, gamma=0.03, kernel=rbf ....................
```

```
model_grid.best_params_
```

```
{'C': 1, 'coef0': 0, 'gamma': 0.03, 'kernel': 'rbf'}
```

Fig. 23. Hyperparameter Tuning

```
# WIth best parameter buil ML Model
model_best = SVC(C=1.0,kernel='rbf',
    gamma= 0.03,coef0= 0,probability=True)
model_best.fit(x_train,y_train)
model_best.score(x_test,y_test)
```

```
0.8426349496797805
```

Fig. 24. Training with best parameter

```
y_pred = model_best.predict(x_test)
y_prob = model_best.predict_proba(x_test) # proability
```

```
cm = metrics.confusion_matrix(y_test,y_pred)
cm = np.concatenate((cm,cm.sum(axis=0).reshape(1,-1)),
                    axis=0)
cm = np.concatenate((cm,cm.sum(axis=1).reshape(-1,1)),
                    axis=1)
plt.imshow(cm)
for i in range(3):
    for j in range(3):
        plt.text(j,i,'%d'%cm[i,j])

plt.xticks([0,1])
plt.yticks([0,1])
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```

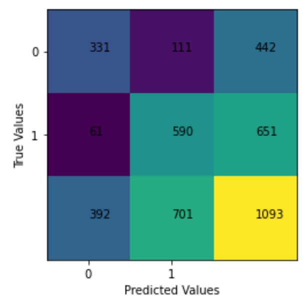


Fig. 25. Confusion Matrix with best parameter

```
# classification report
cr = metrics.classification_report(y_test,y_pred,
    target_names=['male','female'],output_dict=True)
pd.DataFrame(cr).T
```

	precision	recall	f1-score	support
male	0.844388	0.748869	0.793765	442.000000
female	0.841655	0.906298	0.872781	651.000000
accuracy	0.842635	0.842635	0.842635	0.842635
macro avg	0.843021	0.827583	0.833273	1093.000000
weighted avg	0.842760	0.842635	0.840828	1093.000000

Fig. 26. Classification Report with best parameter

```
# kappa
metrics.cohen_kappa_score(y_test,y_pred)
```

```
0.6672837386776724
```

Fig. 27. Kappa Score with best parameter

```python
# roc for female
fpr,tpr,thresh = metrics.roc_curve(y_test,y_prob[:,1])
auc_s = metrics.auc(fpr,tpr)
plt.figure(figsize=(10,6))
plt.plot(fpr,tpr,'-.')
plt.plot([0,1],[0,1],'b--')
for i in range(0,len(thresh),20):
    plt.plot(fpr[i],tpr[i],'^')
    plt.text(fpr[i],tpr[i],"%0.2f"%thresh[i])

plt.legend(['AUC Score = %0.2f'%auc_s])

plt.xlabel('False Positve Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characterstics')
plt.show()
```

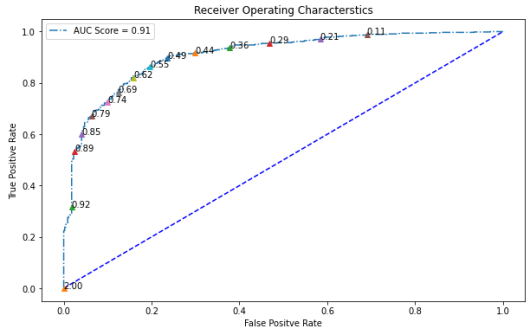


Fig. 28. ROC and AUC with best parameter

```python
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import pickle
import cv2
%matplotlib inline
from PIL import Image
import matplotlib.image as mat_image
```

```python
# Load all the models
haar = cv2.CascadeClassifier('./model/haarcascade_frontalface_default.xml')
# pickle files
mean  = pickle.load(open('./model/mean_preprocess.pickle','rb'))
model_svm  = pickle.load(open('./model/model_svm.pickle','rb'))
model_pca  = pickle.load(open('./model/pca_50.pickle','rb'))

print('Model loaded sucessfully')
```

```
Model loaded sucessfully
```

Fig. 29. All model loading

```python
gender_pre = ['Male','Female']
font = cv2.FONT_HERSHEY_SIMPLEX
```

```python
# test data
test_data_path = './img/s7.jpg'
color = 'bgr'
# step-1: read image
img = cv2.imread(test_data_path)
# step-2: convert into gray scale
if color == 'bgr':
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
else:
    gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
# step-3: crop the face (using haar cascase classifier)
faces = haar.detectMultiScale(gray,1.5,3)
for x,y,w,h in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,255),2) # drawing rectangle
    roi = gray[y:y+h,x:x+w] # crop image
    # step - 4: normalization (0-1)
    roi = roi / 255.0
    # step-5: resize images (100,100)
    if roi.shape[1] > 100:
        roi_resize = cv2.resize(roi,(100,100),cv2.INTER_AREA)
    else:
        roi_resize = cv2.resize(roi,(100,100),cv2.INTER_CUBIC)
    # step-6: Flattening (1x10000)
    roi_reshape = roi_resize.reshape(1,10000) # 1,-1
    # step-7: subptract with mean
    roi_mean = roi_reshape - mean
    # step -8: get eigen image
    eigen_image = model_pca.transform(roi_mean)
    # step -9: pass to ml model (svm)
    results = model_svm.predict_proba(eigen_image)[0]
    # step -10:
    predict = results.argmax() # 0 or 1
    score = results[predict]
    # step -11:
    text = "%s : %0.2f"%(gender_pre[predict],score)
    cv2.putText(img,text,(x,y),font,1,(0,255,255),2)
cv2.imshow('Gender Detector',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 30. Gender classifier Module

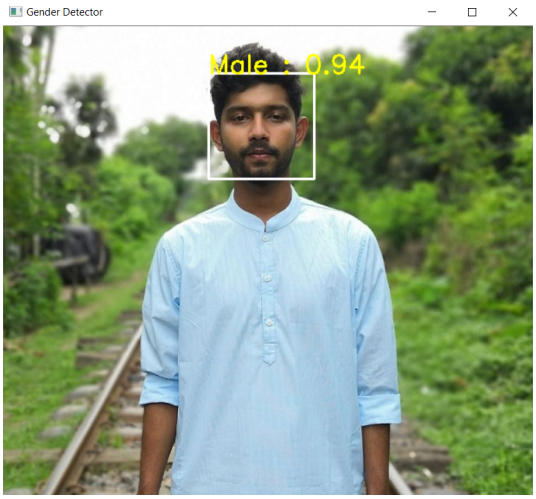


Fig. 31. Example output 01

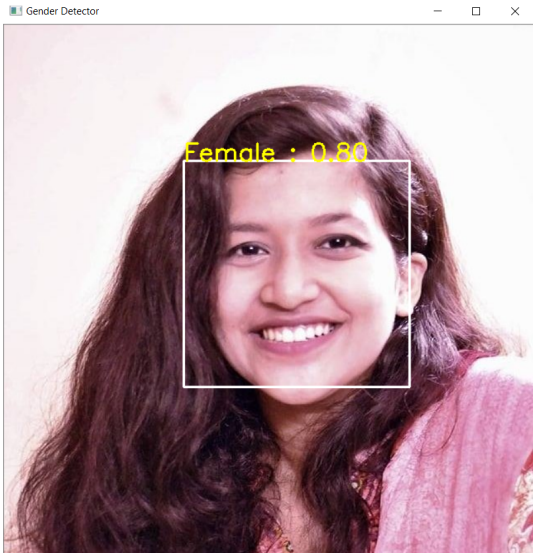


Fig. 32. Example output 02

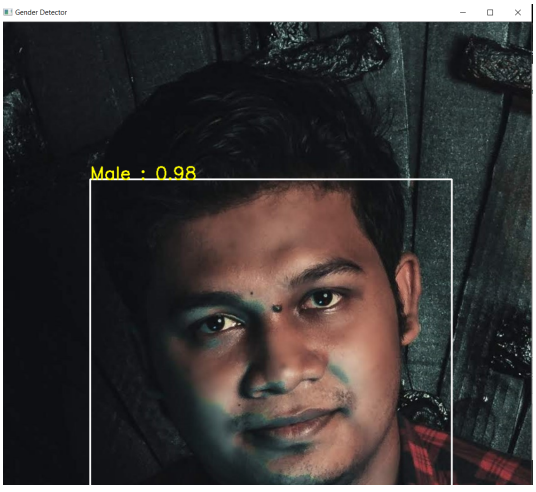


Fig. 33. Example output 03

Fig. 34. Example output 04

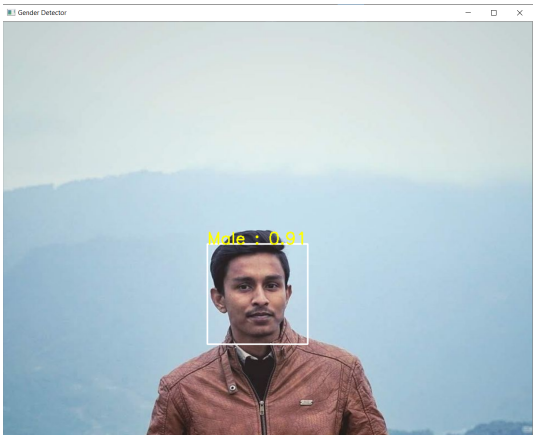


Fig. 35. Example output 05

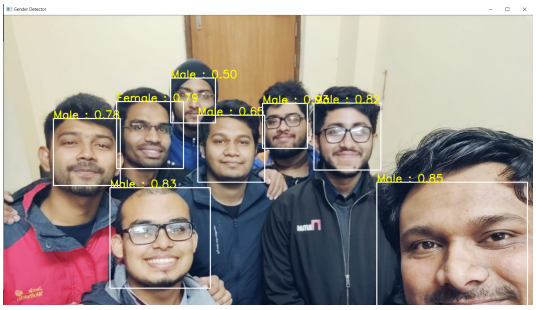


Fig. 36. Example output 06