Image Process and Machine Learning

Dataset in use: https://susanqq.github.io/UTKFace/ (https://susanqq.github.io/UTKFace/)

In-the-wild Faces is used and part-2 is selected for train, part-3 is selected for test set.

In [1]:

```
import os
import cv2
import glob
import pickle
import logging
import numpy as np
from tqdm import tqdm
from sklearn import metrics
from skimage import feature
from sklearn.svm import SVC
import matplotlib.pyplot as plt
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

Histogram of Oriented Gradients (HoG)

In [3]:

```
hog = cv2.HOGDescriptor()
```

In [4]:

```
dest_dir='./utk_train_cropped'
ext='jpg'
```

In [5]:

```
def metric_report(actual, predicted):
    acc = metrics.accuracy_score(actual, predicted)
    precision = metrics.precision_score(actual, predicted)
    recall = metrics.recall_score(actual, predicted)
    f1 = metrics.f1_score(actual, predicted)
    return (acc, precision, recall, f1)
```

In [6]:

```
def hog_pattern_extractor(desc: cv2.HOGDescriptor, source_dir=dest_dir, ext=
    data = []
    labels = []

    for pimg in tqdm(glob.glob(f"{source_dir}/*.{ext}")):
        img = cv2.imread(pimg)
        img_resized = cv2.resize(img, r_shape)
        gray_img = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
        hist = hog.compute(gray_img)
        hist = hist.flatten()
        data.append(hist)
        labels.append(os.path.basename(pimg).split('_')[1])

    return data, labels
```

In [7]:
```
def hog_detector(model, target, ext=ext, r_shape=(64, 128)):

    if os.path.isdir(target):
        logging.debug(f"{target} is a directory full of image.")

        actual, predicted = [], []
        for pimg in tqdm(glob.glob(f"{target}/*.{ext}"), desc='SVM Inference
            img = cv2.imread(pimg)
            img_resized = cv2.resize(img, r_shape)
            gray_img = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
            hist = hog.compute(gray_img)
            hist = hist.flatten()
            pred = model.predict(hist.reshape(1, -1))[0]

            actual.append(int(os.path.basename(pimg).split('_')[1]))
            predicted.append(int(pred))

        return actual, predicted

    else:
        print(f"{target} does not exist.")
```

In [8]:
```
def hog_predict(model_instance, target, r_shape=(64, 128)):

    if os.path.isfile(target): logging.debug(f"{target} is a single image.")

    img = cv2.imread(target, 0)
    img_resized = cv2.resize(img, r_shape)
    hist = hog.compute(img_resized)
    hist = hist.flatten()

    pred = model_instance.predict(hist.reshape(1, -1))[0]

    print('Actual gender:', 'Female' if int(os.path.basename(target).split('
    print('Predicted gender:', 'Female' if int(pred) else 'Man')

    return pred
```

In [9]:
```
'''
Extract histograms and labels
'''

data_hog, labels_hog = hog_pattern_extractor(hog)
```

```
100%|██████████████████████████████████████████████████| 83
34/8334 [00:07<00:00, 1077.12it/s]
```

In [10]:
```
'''
Train SVM Classifier with HOG outputs
'''

model_hog = SVC(kernel='linear', random_state=42)
model_hog.fit(data_hog, labels_hog)
```

Out[10]:
```
 ▾            SVC          i ?
                           (https://scikit-
                           learn.org/1.4/modules/generated/sklearn.svm.SVC.l
 SVC(kernel='linear', random_state=42)
```

In [11]:
```
target_dir = './utk_test_cropped'
```

In [12]:
```
'''
Inference
'''

actual, predicted = hog_detector(model_hog, target_dir)
```

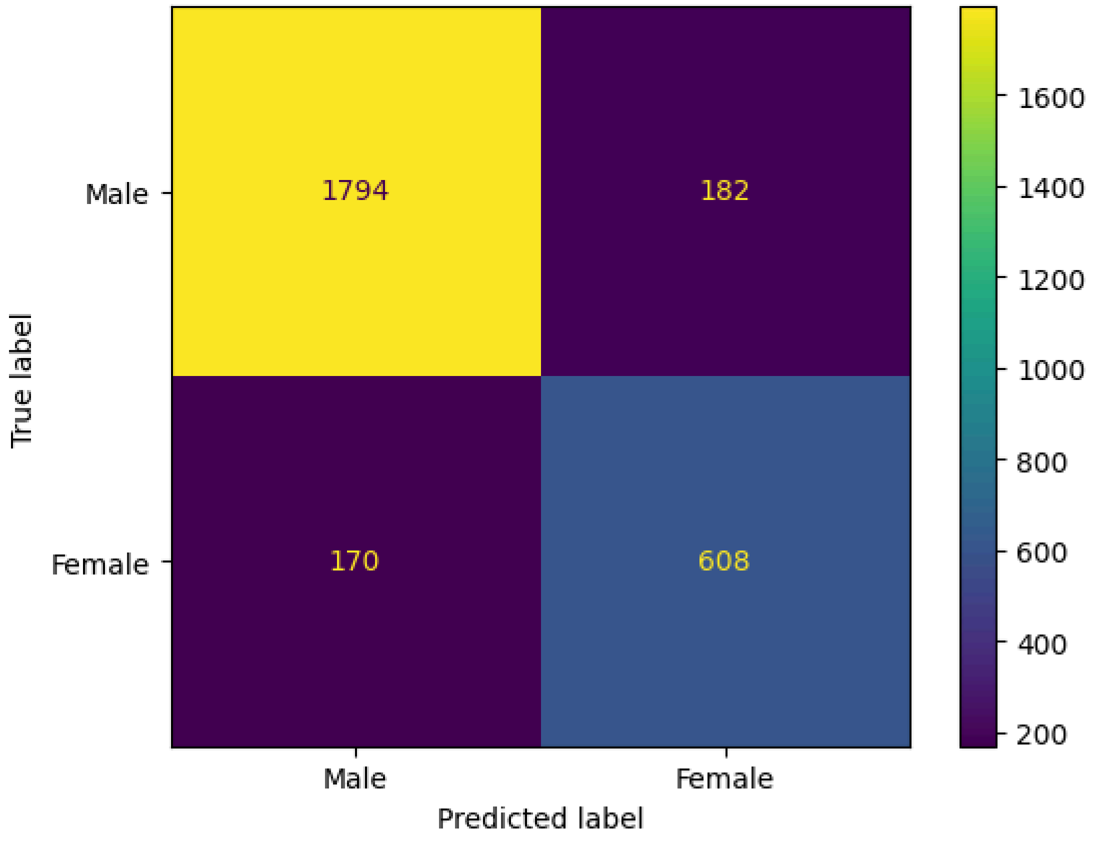
```
SVM Inference: 100%|████████████████████████████████████████| 2
754/2754 [00:19<00:00, 143.52it/s]
```

In [13]:
```
metric_report(actual, predicted)
```

Out[13]:
```
(0.8721859114015976, 0.769620253164557, 0.781491002570694, 0.77551020408163
26)
```

In [14]:
```
confusion_matrix = metrics.confusion_matrix(actual, predicted)
```

In [15]:
```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_mat
cm_display.plot()
plt.show()
```



In [16]:
```
fname = 'hog_svm_model.sav'
#pickle.dump(model_hog, open(fname, 'wb'))
```

In [17]:
```
loaded_model = pickle.load(open(fname, 'rb'))
```

In [18]:
```
# Verilen folder icindeki yuzlerden gender tespit etme
test_dir = './test_images'
actual_test, predicted_test = hog_detector(loaded_model, test_dir)
```

```
SVM Inference: 100%|████████████████████████████████████████
█| 10/10 [00:00<00:00, 156.24it/s]
```

In [19]:
```
metric_report(actual_test, predicted_test)
```

Out[19]:
```
(1.0, 1.0, 1.0, 1.0)
```

In [20]:
```
# Verilen goruntuden gender tespit etme
test_image = 'test_images/16_0_0_20170120133327900_cropped.jpg'
gender = hog_predict(loaded_model, test_image)
```

```
Actual gender: Man
Predicted gender: Man
```

Local Binary Patterns

In [21]:
```
class LocalBinaryPatterns:
    def __init__(self, numPoints, radius):
        self.numPoints = numPoints
        self.radius = radius

    def describe(self, image, eps=1e-7):
        lbp = feature.local_binary_pattern(image, self.numPoints, self.radiu
        (hist, _) = np.histogram(lbp.ravel(),
                                 bins=np.arange(0, self.numPoints + 3),
                                 range=(0, self.numPoints + 2))

        hist = hist.astype("float")
        hist /= (hist.sum() + eps)
        return hist
```

In [22]:
```
desc = LocalBinaryPatterns(24, 8)
```

In [23]:
```
def lbp_pattern_extractor(desc: LocalBinaryPatterns, source_dir=dest_dir, ex
    data = []
    labels = []

    for pimg in tqdm(glob.glob(f"{source_dir}/*.{ext}")):
        img = cv2.imread(pimg, 0)

        hist = desc.describe(img)
        hist = hist.flatten()

        data.append(hist)
        labels.append(os.path.basename(pimg).split('_')[1])

    return data, labels
```

In [24]:
```
def lbp_detector(model, target, ext=ext):

    if os.path.isdir(target):
        logging.debug(f"{target} is a directory full of image.")

        actual, predicted = [], []
        for pimg in tqdm(glob.glob(f"{target}/*.{ext}"), desc='SVM Inference
            img = cv2.imread(pimg, 0)
            hist = desc.describe(img)
            hist = hist.flatten()
            pred = model.predict(hist.reshape(1, -1))[0]

            actual.append(int(os.path.basename(pimg).split('_')[1]))
            predicted.append(int(pred))

        return actual, predicted

    else:
        print(f"{target} does not exist.")
```

In [25]:
```
def lbp_predict(model_instance, target):

    if os.path.isfile(target): logging.debug(f"{target} is a single image.")

    img = cv2.imread(target, 0)
    hist = desc.describe(img)
    hist = hist.flatten()

    pred = model_instance.predict(hist.reshape(1, -1))[0]

    print('Actual gender:', 'Female' if int(os.path.basename(target).split('
    print('Predicted gender:', 'Female' if int(pred) else 'Man')

    return pred
```

In [26]:
```
male_histograms = np.load('male_lbp_histograms.npy')
female_histograms = np.load('female_lbp_histograms.npy')
```

In [27]:
```
data = np.vstack((male_histograms, female_histograms))
labels = np.hstack( (np.zeros(len(male_histograms)), np.ones(len(female_hist
```

In [28]:
```
model_lbp = SVC(kernel='linear', random_state=42)
model_lbp.fit(data, labels)
```

Out[28]:
```
▾            SVC            ⓘ ?
                              (https://scikit-
                              learn.org/1.4/modules/generated/sklearn.svm.SVC.|
SVC(kernel='linear', random_state=42)
```

In [29]:
```
actual, predicted = lbp_detector(model_lbp, target_dir)
```

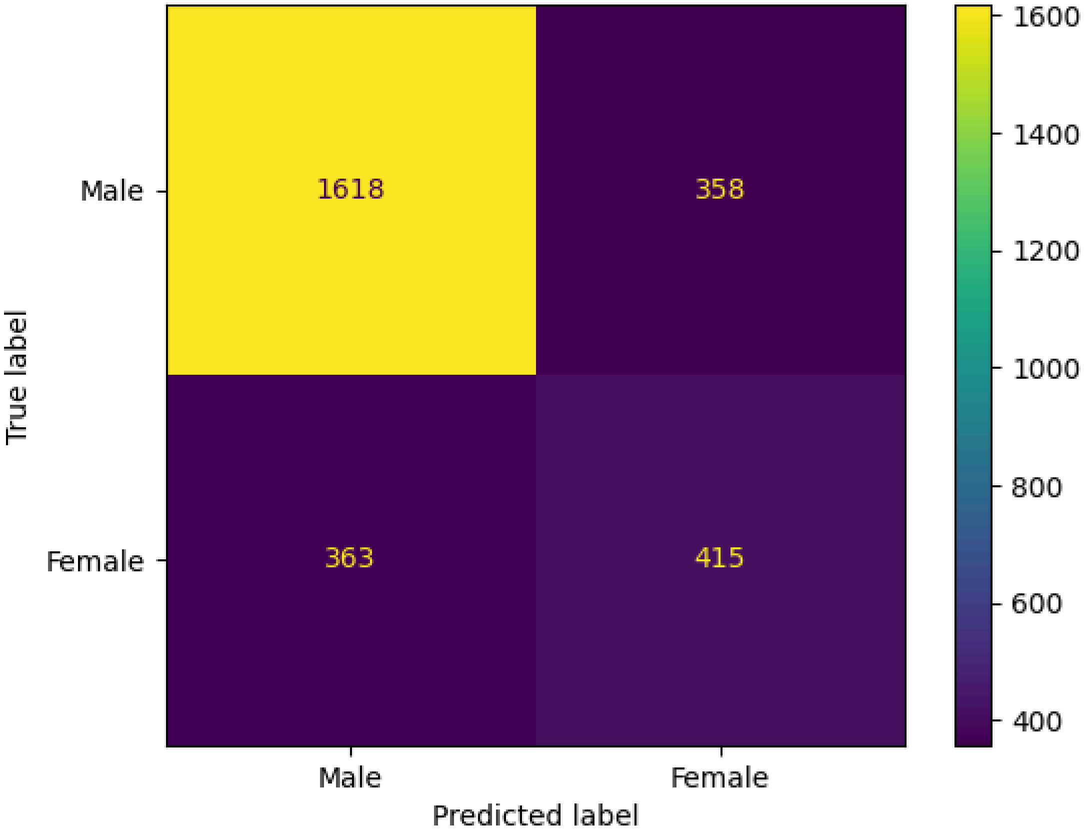
```
SVM Inference: 100%|██████████████████████████████████████████|
2754/2754 [01:12<00:00, 37.98it/s]
```

In [30]:
```
metric_report(actual, predicted)
```

Out[30]:
```
(0.7381989832970225, 0.536869340232859, 0.5334190231362468, 0.5351386202450
032)
```

In [31]:
```
confusion_matrix = metrics.confusion_matrix(actual, predicted)
```

In [32]:
```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_mat
cm_display.plot()
plt.show()
```



In [33]:
```
fname = 'lbp_svm_model.sav'
#pickle.dump(model_lbp, open(fname, 'wb'))
```

In [34]:
```
loaded_model = pickle.load(open(fname, 'rb'))
```

In [35]:
```
# Verilen folder icindeki yuzlerden gender tespit etme
test_dir = './test_images'
actual_test, predicted_test = lbp_detector(loaded_model, test_dir)
```

```
SVM Inference: 100%|██████████████████████████████████████████████
█| 10/10 [00:00<00:00, 35.19it/s]
```

In [36]:
```
metric_report(actual_test, predicted_test)
```

Out[36]:
```
(0.8, 0.7142857142857143, 1.0, 0.8333333333333334)
```

In [37]:

```
# Verilen goruntuden gender tespit etme
test_image = 'test_images/27_0_0_20170117013808240_cropped.jpg'
gender = lbp_predict(loaded_model, test_image)
```

```
Actual gender: Man
Predicted gender: Man
```

Scale Invariant Feature Transform (SIFT)

In [38]:

```
def extract_descriptors(image, extractor):
    gray = cv2.imread(image, 0)
    keypoints, descriptors = extractor.detectAndCompute(gray, None)
    return descriptors
```

In [39]:

```
def sift_feature_extractor():
    sift = cv2.SIFT_create()

    data = []
    labels = []

    for pimg in tqdm(glob.glob(f"{source_dir}/*.{ext}")):
        img = cv2.imread(pimg)
        img_resized = cv2.resize(img, r_shape)
        gray_img = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
        _, descriptors = sift.detectAndCompute(gray_img)
        data.append(descriptors)
        labels.append(os.path.basename(pimg).split('_')[1])

    return data, labels
```

In [40]:

```
## Bag of Words will be implemented (feedback)
```