Image Process and Feature Matching

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.

LBP için resize eklenmiştir. Değişen bir durum olmamıştır. Metrikler aşağıdaki gibidir.

In [1]:

```
import os
import cv2
import glob
import logging
import numpy as np
import pandas as pd
from tqdm import tqdm
from sklearn import metrics
from skimage import feature
import matplotlib.pyplot as plt
```

In [2]:

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

In [3]:

```
def mkdir(path):
    try:
        if not os.path.exists(path):
            os.makedirs(path)
            logging.debug(f"Folder '{path}' created successfully.")
        else:
            raise RuntimeError(f"Folder '{path}' already exists.")
    except Exception as e:
        logging.error(f"Error creating folder '{path}': {e} You are going to
        raise
```

In [4]:

```
'''
Raporda belirtildigi gibi, girdi olarak net cekilen 1 adet yuz goruntusu ver

SIFT, SURF, distance vb. için tabi ki hazır fonksiyonları kullanacaksınız an
hazır kütüphane kullanamazsınız.

ibaresi proje ile alakali degildir. Projemiz yuz tespiti degil, girdi olarak
Girdi hatasi olmasi durumunda kodun yine de calismasi amaciyla hazir kutupha
'''

def face_extractor(source_dir, dest_dir, ext='jpg'):
    mkdir(dest_dir)
    haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascad

    pimgs = glob.glob(f"{source_dir}/*.{ext}")

    for pimg in tqdm(pimgs, desc="Cropping faces from wild images"):
        img=cv2.imread(pimg)
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces_rect = haar_cascade.detectMultiScale(gray_img, 1.2, 5)

        if faces_rect is ():
            logging.debug('No face detected')
            continue

        for (x, y, w, h) in faces_rect:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cropped_face = gray_img[y:y+h, x:x+w]
            img_name = os.path.splitext(os.path.basename(pimg))[0]
            cv2.imwrite(f'{dest_dir}/{img_name}_cropped.jpg', cropped_face)
```

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]:

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

In [7]:

```
test_source_dir='./utk_test'
test_dest_dir='./utk_test_cropped'
```

In [8]:

```
test_dir = './test_images'
```

Local Binary Patterns

In [9]:
```
class LocalBinaryPatterns:
    def __init__(self, numPoints, radius, r_shape=(96, 96)):
        self.numPoints = numPoints
        self.radius = radius
        self.r_shape = r_shape

    def describe(self, image, eps=1e-7):
        image = cv2.resize(image, self.r_shape)
        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 [10]:
```
def single_image_pipeline(img, desc, male_hist_path, female_hist_path):
    fimg = cv2.imread(img, 0)
    # cv2.COLOR_BGR2GRAY

    lbp_hist = desc.describe(fimg)
    lbp_hist = lbp_hist.astype(np.float32)

    male_lbp_hist = np.load(male_hist_path).astype(np.float32)
    female_lbp_hist = np.load(female_hist_path).astype(np.float32)

    male_distance = cv2.compareHist(lbp_hist, male_lbp_hist, cv2.HISTCMP_IN
    female_distance = cv2.compareHist(lbp_hist, female_lbp_hist, cv2.HISTCMP
    # HISTCMP_INTERSECT, HISTCMP_CORREL, HISTCMP_BHATTACHARYYA, HISTCMP_HELL

    return 0 if male_distance >= female_distance else 1
```

In [11]:
```
'''
UTK Train setindeki goruntulerden yuzler elde edilir ve dest_dir uzerine kay
'''
#face_extractor(source_dir, dest_dir)
```

Out[11]:
```
'\nUTK Train setindeki goruntulerden yuzler elde edilir ve dest_dir uzerin
e kaydedilir.\n'
```

In [12]:
```
'''
Elde edilen yuz goruntuleri cinsiyete gore ayristirilir.
'''

male_images=[]
female_images=[]
for pimg in tqdm(glob.glob(f"{dest_dir}/*.{ext}"), desc="Creating Male and F
    img = cv2.imread(pimg, cv2.COLOR_BGR2GRAY)
    female_images.append(img) if int(os.path.basename(pimg).split('_')[1]) e

```

```
Creating Male and Female Arrays: 100%|██████████████████████████████| 8
334/8334 [00:03<00:00, 2125.35it/s]
```

In [13]:
```
#print(f'Number of male images in trainset: {len(male_images)}\nNumber of fe
```

In [14]:
```
desc=LocalBinaryPatterns(24, 4) # define descriptor instance
```

In [15]:
```
'''
Male-Female LBP histogramlari elde edilir ve kaydedilir.
'''

male_histograms = [desc.describe(img) for img in tqdm(male_images, desc="Co
female_histograms = [desc.describe(img) for img in tqdm(female_images, desc

male_lbp_hist = np.mean(male_histograms, axis=0)
female_lbp_hist = np.mean(female_histograms, axis=0)

np.save('male_lbp_hist_resized.npy', male_lbp_hist)
np.save('female_lbp_hist_resized.npy', female_lbp_hist)
```

```
Computing LBP histogram - Male: 100%|██████████████████████████████████|
4512/4512 [00:15<00:00, 299.40it/s]
Computing LBP histogram - Female: 100%|████████████████████████████████|
3822/3822 [00:12<00:00, 303.03it/s]
```

In [16]:
```
#face_extractor(source_dir, dest_dir)
```

In [17]:
```
'''
Test veri setinden rastgele goruntu secilerek cinsiyet tahmini yapilir.
'''

random_image_pick = np.random.choice(glob.glob(test_dir + f'/*.{ext}'))
pred = single_image_pipeline(random_image_pick, desc, 'male_lbp_hist_resize
```

In [18]:
```
'''
Tum test verisi uzerinde tahminler gerceklestirilir.
'''


timgs = glob.glob(f"{test_dest_dir}/*.{ext}") #test images

actual, predicted = [], []
for timg in tqdm(timgs, desc="Inference in progress"):
    gender = single_image_pipeline(timg, desc, 'male_lbp_hist_resized.npy',
    label = int(os.path.basename(timg).split('_')[1])

    actual.append(label)
    predicted.append(gender)
```

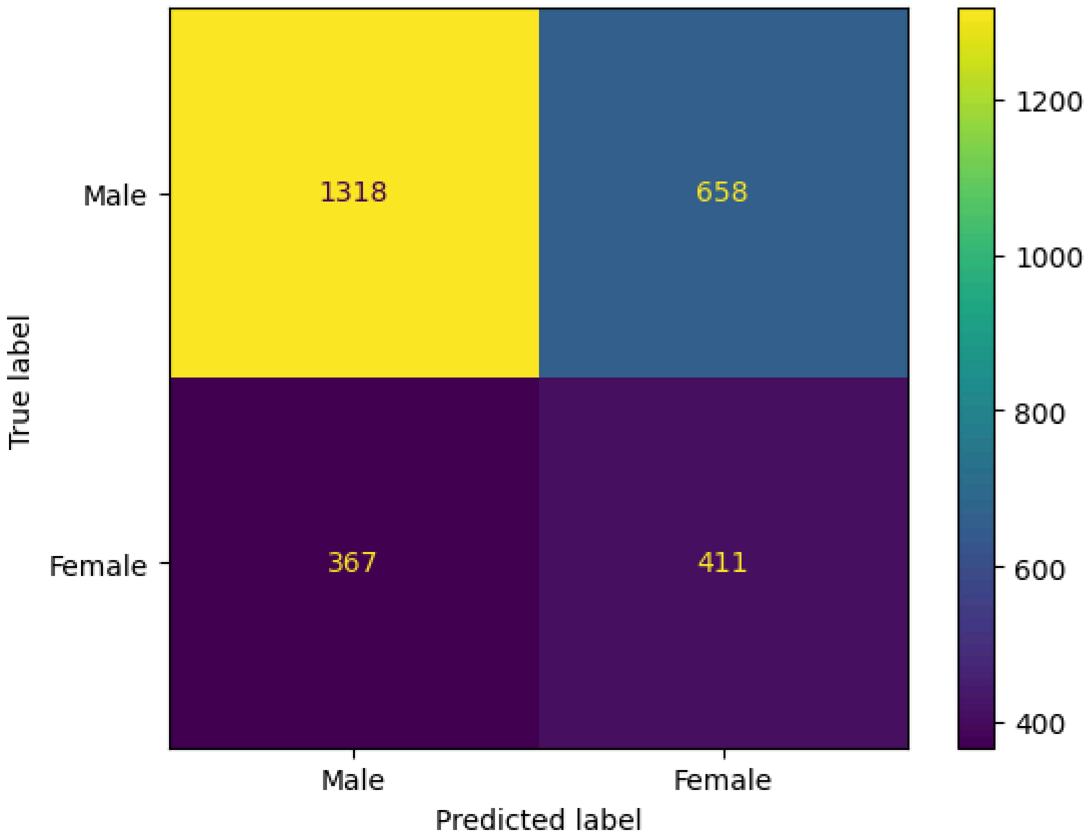
```
Inference in progress: 100%|███████████████████████████████████████████|
2754/2754 [00:10<00:00, 253.05it/s]
```

In [19]:
```
metric_report(actual, predicted)
```

Out[19]:
```
(0.6278140885984024,
 0.3844714686623012,
 0.5282776349614395,
 0.44504602057390363)
```

```
In [20]: confusion_matrix = metrics.confusion_matrix(actual, predicted)
```

```
In [21]: cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_mat
         cm_display.plot()
         plt.show()
```



```
In [22]: female_data = np.sum(actual); male_data = len(actual) - female_data
         female_data, male_data
```

```
Out[22]: (778, 1976)
```

```
In [23]: raise Exception("Eski Kod! Stop Here!")
```

```
---------------------------------------------------------------------------
-
Exception                                 Traceback (most recent call las
t)
Cell In[23], line 1
----> 1 raise Exception("Eski Kod! Stop Here!")

Exception: Eski Kod! Stop Here!
```