```
X_rec_origin_train.shape, X_rec_origin_test.shape, y_rec_origin_train.shape, y_rec_
```

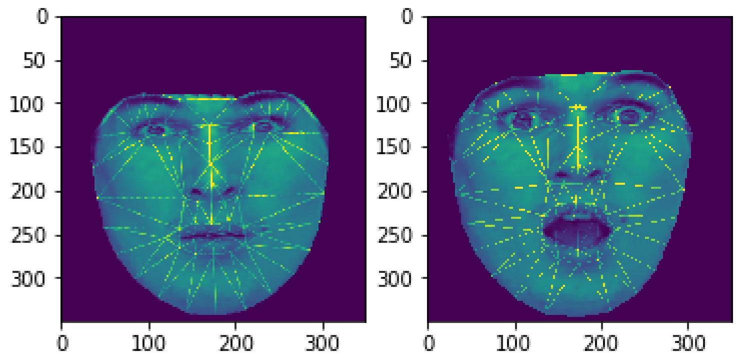
```
((3350, 2), (1650, 2), (3350,), (1650,))
```

```
len(set(y_rec_origin_train.to_numpy()))
```

```
118
```

```
pairPlot(images_df[['neutralised_image']].iloc[205].to_numpy()[0], images_df[['orig
```

```
<Figure size 432x288 with 0 Axes>
```



```
array([ 68,  10,  68, ..., 115,  62,  56], dtype=int32)
```

Fisher Face

```
fishface = cv2.face.FisherFaceRecognizer_create() #Initialize fisher face classifie
fishface.train(X_rec_origin_train['neutralised_image'].to_numpy(), y_rec_origin_tra


correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['original_image']):
    pred, conf = fishface.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st

print('Emotion Acc: ', (100*correct)/(correct + incorrect))

correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['neutralised_image']):
    pred, conf = fishface.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st

print('Emotionless Acc: ', (100*correct)/(correct + incorrect))
```

```
[2019-12-10 19:20:06] 0.0% is done
[2019-12-10 19:20:08] 30.303030303030305% is done
[2019-12-10 19:20:10] 60.60606060606061% is done
[2019-12-10 19:20:13] 90.9090909090909% is done
Emotion Acc:  99.81818181818181
[2019-12-10 19:20:13] 0.0% is done
[2019-12-10 19:20:16] 30.303030303030305% is done
[2019-12-10 19:20:18] 60.60606060606061% is done
[2019-12-10 19:20:21] 90.9090909090909% is done
Emotionless Acc:  99.33333333333333
```

```
print(fishface.predict(X_rec_origin_test['original_image'][10]), y_rec_origin_test[
      fishface.predict(X_rec_origin_test['neutralised_image'][10]), y_rec_origin_te
```

```
(90, 4577.445127442707) 55 (55, 134.1073278241769) 55
```

Eigen Face

```
eigenface = cv2.face.EigenFaceRecognizer_create()
eigenface.train(X_rec_origin_train['neutralised_image'].to_numpy(), y_rec_origin_tr


correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['original_image']):
    pred, conf = eigenface.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st

print('Emotion Acc: ', (100*correct)/(correct + incorrect))

correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['neutralised_image']):
    pred, conf = eigenface.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st

print('Emotionless Acc: ', (100*correct)/(correct + incorrect))
```

```
[2019-12-09 21:12:13] 0.0% is done
[2019-12-09 21:13:46] 30.303030303030305% is done
[2019-12-09 21:15:18] 60.60606060606061% is done
[2019-12-09 21:16:50] 90.9090909090909% is done
Emotion Acc:  99.03030303030303
[2019-12-09 21:17:18] 0.0% is done
[2019-12-09 21:18:51] 30.303030303030305% is done
[2019-12-09 21:20:23] 60.60606060606061% is done
[2019-12-09 21:21:56] 90.9090909090909% is done
Emotionless Acc:  99.15151515151516
```

```
print(eigenface.predict(X_rec_origin_test['original_image'][10]), y_rec_origin_test
      eigenface.predict(X_rec_origin_test['neutralised_image'][10]), y_rec_origin_t
```

```
(90, 12562.90669242794) 55 (55, 1924.4147285173933) 55
```

LBHP

```
lbph_face = cv2.face.LBPHFaceRecognizer_create()
lbph_face.train(X_rec_origin_train['original_image'], y_rec_origin_train.to_numpy(

correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['original_image']):
    pred, conf = lbph_face.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st
print('Emotion Acc: ', (100*correct)/(correct + incorrect))

correct = 0
incorrect = 0
for cnt, image in enumerate(X_rec_origin_test['neutralised_image']):
    pred, conf = lbph_face.predict(image)
    if pred == np.array(y_rec_origin_test)[cnt]:
        correct += 1
    else:
        incorrect += 1

    if cnt % 500 == 0:
        print("[{}] {}% is done".format(strftime("%Y-%m-%d %H:%M:%S", gmtime()), st
print('Emotionless Acc: ', (100*correct)/(correct + incorrect))
```

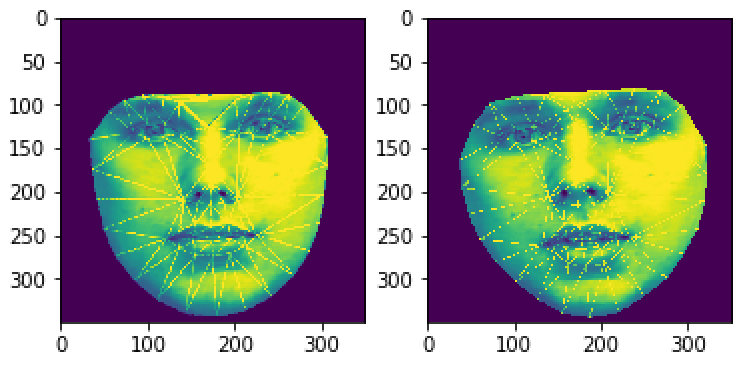
```
[2019-12-09 19:22:26] 0.0% is done
[2019-12-09 19:24:29] 30.303030303030305% is done
[2019-12-09 19:26:33] 60.60606060606061% is done
[2019-12-09 19:28:36] 90.9090909090909% is done
Emotion Acc:  99.87878787878788
[2019-12-09 19:29:13] 0.0% is done
[2019-12-09 19:31:16] 30.303030303030305% is done
[2019-12-09 19:33:19] 60.60606060606061% is done
[2019-12-09 19:35:22] 90.9090909090909% is done
Emotionless Acc:  99.81818181818181
```

```
print(lbph_face.predict(X_rec_origin_test['original_image'][10]), y_rec_origin_test
      lbph_face.predict(X_rec_origin_test['neutralised_image'][10]), y_rec_origin_t
```

```
(73, 4.596833089278591) 73 (73, 4.734876506819591) 73
```

```
im_indx = 355
pairPlot(images_df.iloc[im_indx]['neutralised_image'],images_df.iloc[im_indx]['orig
```

```
<Figure size 432x288 with 0 Axes>
```



```
print(lbph_face.predict(X_rec_origin_test['original_image'][im_indx]), y_rec_origi
      lbph_face.predict(X_rec_origin_test['neutralised_image'][im_indx]), y_rec_or
```

```
(75, 4.857063873657447) 75 (75, 14.4775203870867) 75
```