Face Classification and Verification

Statistical Methods in Artificial Intelligence
Assignment-2
Monsoon 2019
Abhinav Gupta - 20171059

I. INTRODUCTION

This assignment deals with the problems of classification and verification. We use the popular ”face” problem space, that is, our dataset contains many images and these images belong to a particular class. Our objective is to get familiar with this problem space and be able to train learning models on the datasets so that we can classify a new image into its correct class.

We have been provided with three datasets:
1. IIIT Cartoons
2. IMFDB movie stars
3. Yale dataset

II. EIGEN FACES

A. What are eigen faces?

Eigen faces are the set of eigen vectors of the covariance matrix of the data face images, that are used in the problem of human face recognition. They are essentially the ”faces” that are RECONSTRUCTED after applying dimensionality reduction tchniques like PCA, on the given data set.

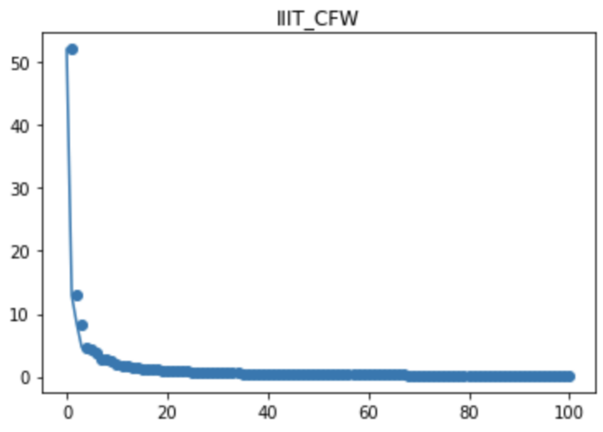
So we apply a feature transformation on the data set to reduce the dimensionality of the images. Then, when we reconstruct it back with the help of the eigen vectors, we get the ”reconstructed faces” or the ”eigen faces” back. Hence these eigen vectors are called eigen faces.

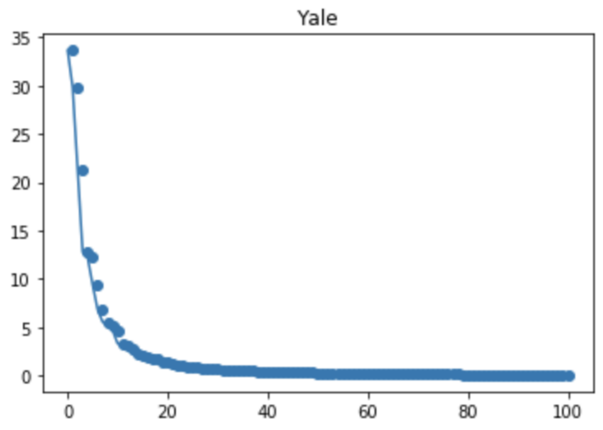
B. How many eigenvectors are required for satisfactory reconstruction?

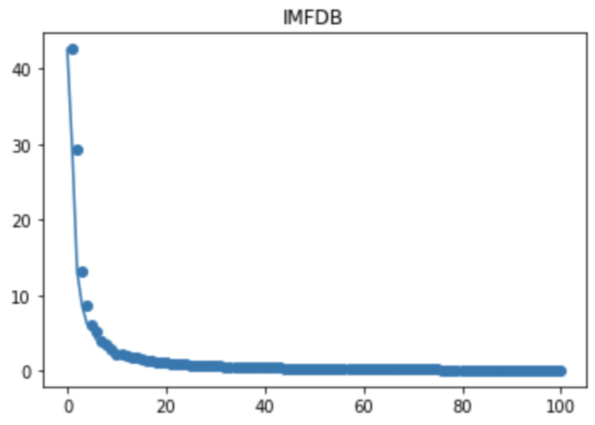
We plot the eigen value spectrum for all the three datasets. This graph contains the eigen values on the X axis and their corresponding values on the Y-axis. We notice that these 3 spectrums are pretty similar. We have also sorted these eigenvalues before plotting them.

So we notice that it’s only the first 50-60 eigenvalues that are non-zero. After that, most of them become zero. Hence, we could use around 50-60 eigenvectors for reconstruction, they should be satisfactory for IMFDB and Yale.

However, IIIT dataset, we notice that there are very few eigenvalues with high values. We know that we need atleast 95 percent of the data and for IIIT dataset, we calculate the threshold, that is, the number of eigen values that represent 95 percent of the data. This number comes out to be 145-160. Hence we need around 145-160 eigenvectors for the IIIT data set to be able to reconstruct satisfactorily. Here are the eigen value spectrum plots for the three datasets:

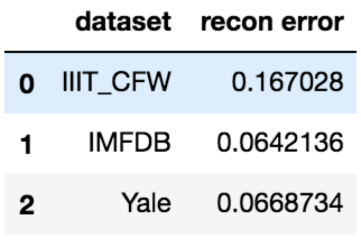






C. Which dataset is difficult to represent compactly with fewer eigen-vectors?

We apply PCA to all the datasets, and reconstruct the images. Here are the reconstruction errors we get.



	dataset	recon error
0	IIIT_CFW	0.167028
1	IMFDB	0.0642136
2	Yale	0.0668734

So we get the least error in the movies dataset.

D. Which person/identity is difficult to represent compactly with fewer eigen-vectors?

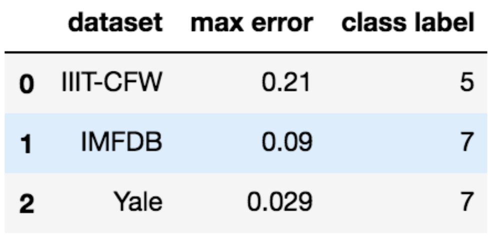
For this, we compute the reconstruction error for each label/class in each dataset. We do this to figure out which class is the most difficult to classify after applying PCA.

First we do this for IIIT dataset.
We get maximum error of 0.235 in class label 6

For IMFDB dataset.
We get maximum error of 0.121 in class label 7

For Yale dataset.
We get maximum error of 0.111 in class label 7

So it's most difficult to represent class 6 in IIIT dataset. This is if we apply PCA to reduce to 3 dimensions. Although if we change the number of dimensions in PCA, then we get varying results. For instance, if k=9, we get maximum error in class label 5 of IIIT data set. But mostly, the largest error belongs to IIIT dataset only.
Here are the results if k=9.



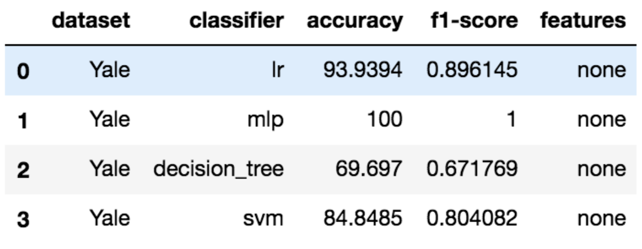
	dataset	max error	class label
0	IIIT-CFW	0.21	5
1	IMFDB	0.09	7
2	Yale	0.029	7

III. Classification of Data

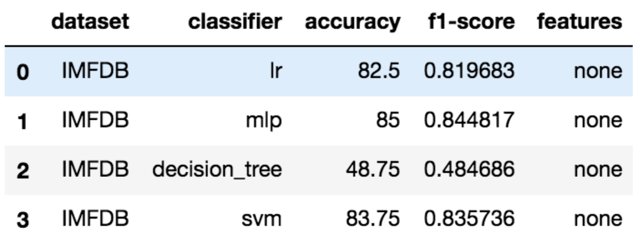
A. Using different classifiers

First, we use the MLP classifier to classify the data, and tabulate the results and accuracy of MLP on all the three datasets. Then, we use other classifiers, including SVM, logisitc regression and decision tree as well, so that we can compare them with MLP.

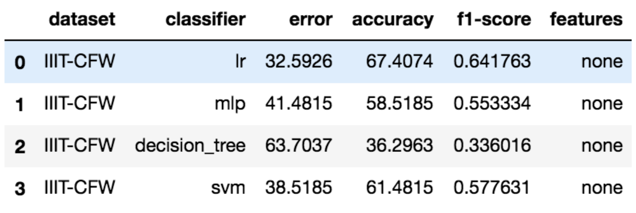
Here are three tables, that show the accuracies of the above 4 classifiers, on the 3 datasets:



	dataset	classifier	accuracy	f1-score	features
0	Yale	lr	93.9394	0.896145	none
1	Yale	mlp	100	1	none
2	Yale	decision_tree	69.697	0.671769	none
3	Yale	svm	84.8485	0.804082	none



	dataset	classifier	accuracy	f1-score	features
0	IMFDB	lr	82.5	0.819683	none
1	IMFDB	mlp	85	0.844817	none
2	IMFDB	decision_tree	48.75	0.484686	none
3	IMFDB	svm	83.75	0.835736	none

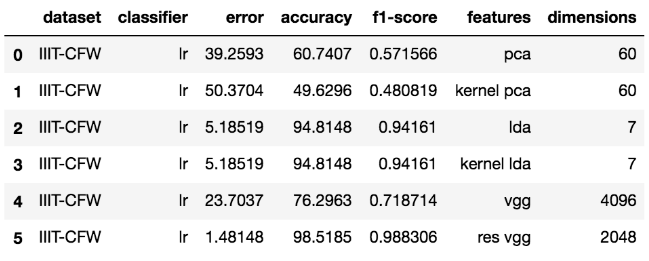


	dataset	classifier	error	accuracy	f1-score	features
0	IIIT-CFW	lr	32.5926	67.4074	0.641763	none
1	IIIT-CFW	mlp	41.4815	58.5185	0.553334	none
2	IIIT-CFW	decision_tree	63.7037	36.2963	0.336016	none
3	IIIT-CFW	svm	38.5185	61.4815	0.577631	none

IV. Using different feature transforms

Now next, we apply feature transformations like PCA and LDA on the data and then classify the tranformed data.

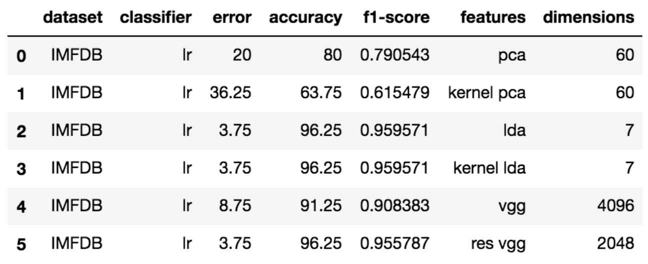
1) Using single feature transforms: We use logistic regression as our classifier, and apply 6 different feature transformations and analyse the results. Following are three tables, that show the accuracy results of logistic regression that has been used to classify data with different feature transforms, on the 3 datasets.



	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	IIIT-CFW	lr	39.2593	60.7407	0.571566	pca	60
1	IIIT-CFW	lr	50.3704	49.6296	0.480819	kernel pca	60
2	IIIT-CFW	lr	5.18519	94.8148	0.94161	lda	7
3	IIIT-CFW	lr	5.18519	94.8148	0.94161	kernel lda	7
4	IIIT-CFW	lr	23.7037	76.2963	0.718714	vgg	4096
5	IIIT-CFW	lr	1.48148	98.5185	0.988306	res vgg	2048

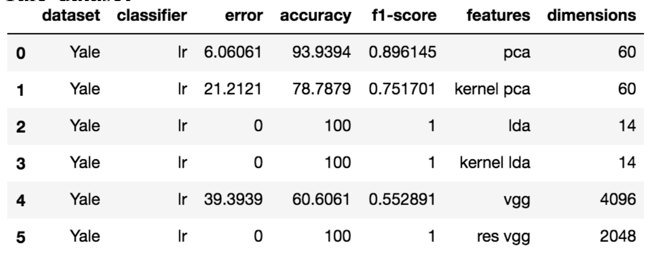
So as we can see, we have applied same classifier 'lr' logistic regression on the above IIIT dataset. The next three columns represent the error in classification, accuracy and the f1 score - our metrics for analysing performance. Feautures column show which feature transformation has been applied and dimensions show the number of dimensions reduced to, after applying the transformation.

IMFDB dataset:



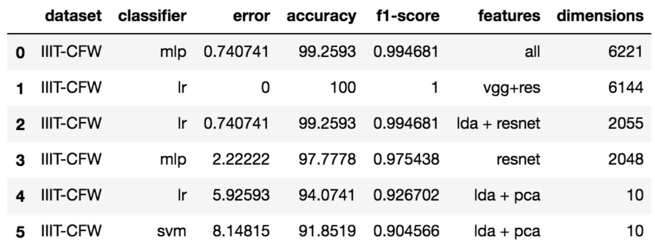
	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	IMFDB	lr	20	80	0.790543	pca	60
1	IMFDB	lr	36.25	63.75	0.615479	kernel pca	60
2	IMFDB	lr	3.75	96.25	0.959571	lda	7
3	IMFDB	lr	3.75	96.25	0.959571	kernel lda	7
4	IMFDB	lr	8.75	91.25	0.908383	vgg	4096
5	IMFDB	lr	3.75	96.25	0.955787	res vgg	2048

Yale dataset



	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	Yale	lr	6.06061	93.9394	0.896145	pca	60
1	Yale	lr	21.2121	78.7879	0.751701	kernel pca	60
2	Yale	lr	0	100	1	lda	14
3	Yale	lr	0	100	1	kernel lda	14
4	Yale	lr	39.3939	60.6061	0.552891	vgg	4096
5	Yale	lr	0	100	1	res vgg	2048

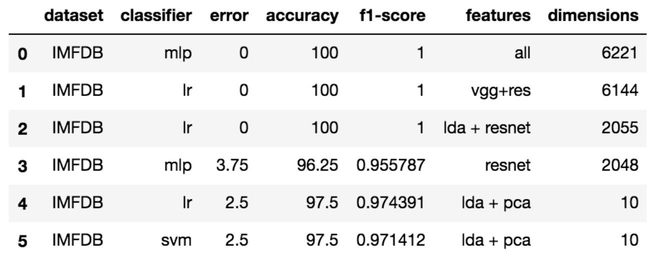
2) Combination of features: Now we pply a combination of feature transformations on the dataset for classification. Here are the results for the different datasets.



	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	IIIT-CFW	mlp	0.740741	99.2593	0.994681	all	6221
1	IIIT-CFW	lr	0	100	1	vgg+res	6144
2	IIIT-CFW	lr	0.740741	99.2593	0.994681	lda + resnet	2055
3	IIIT-CFW	mlp	2.22222	97.7778	0.975438	resnet	2048
4	IIIT-CFW	lr	5.92593	94.0741	0.926702	lda + pca	10
5	IIIT-CFW	svm	8.14815	91.8519	0.904566	lda + pca	10

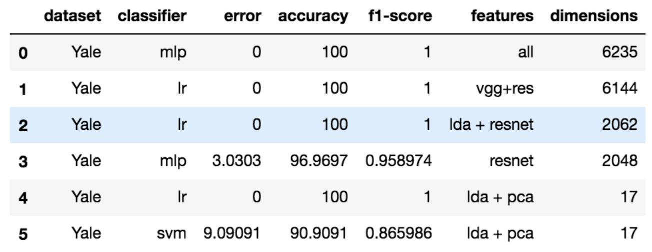
VGG + ResNet gives us a great accuracy!

IMFDB dataset:



	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	IMFDB	mlp	0	100	1	all	6221
1	IMFDB	lr	0	100	1	vgg+res	6144
2	IMFDB	lr	0	100	1	lda + resnet	2055
3	IMFDB	mlp	3.75	96.25	0.955787	resnet	2048
4	IMFDB	lr	2.5	97.5	0.974391	lda + pca	10
5	IMFDB	svm	2.5	97.5	0.971412	lda + pca	10

Yale dataset

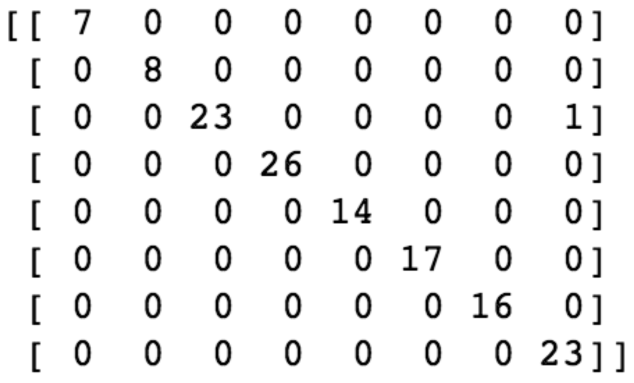


	dataset	classifier	error	accuracy	f1-score	features	dimensions
0	Yale	mlp	0	100	1	all	6235
1	Yale	lr	0	100	1	vgg+res	6144
2	Yale	lr	0	100	1	lda + resnet	2062
3	Yale	mlp	3.0303	96.9697	0.958974	resnet	2048
4	Yale	lr	0	100	1	lda + pca	17
5	Yale	svm	9.09091	90.9091	0.865986	lda + pca	17

Combination of all features and LDA+ResNet are good combinations, but VGG+ResNet steals the show here, with 100 accuracy in all 3 datasets!

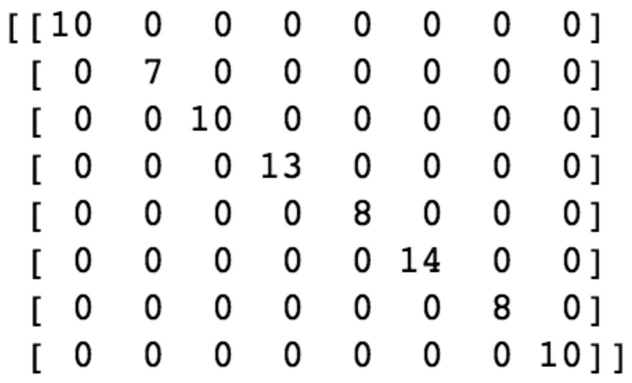
3) Confusion Matrices: We print the confusion matrices for the best model for eafh dataset.

1. IIIT



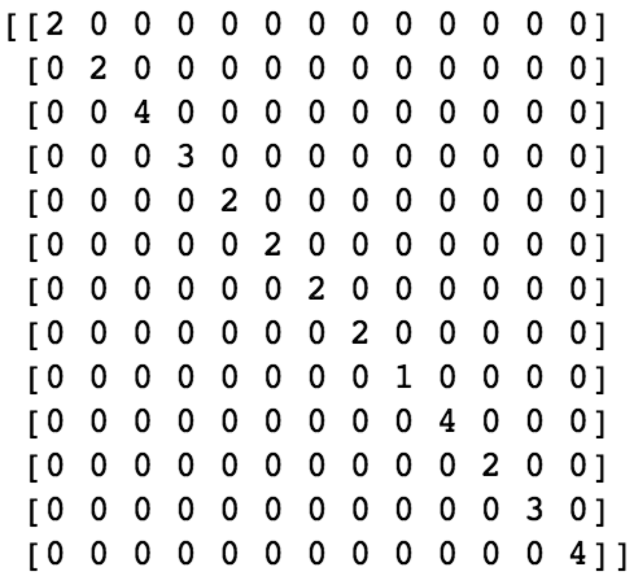
```
[[ 7  0  0  0  0  0  0  0]
 [ 0  8  0  0  0  0  0  0]
 [ 0  0 23  0  0  0  0  1]
 [ 0  0  0 26  0  0  0  0]
 [ 0  0  0  0 14  0  0  0]
 [ 0  0  0  0  0 17  0  0]
 [ 0  0  0  0  0  0 16  0]
 [ 0  0  0  0  0  0  0 23]]
```

2. IMFDB



```
[[10  0  0  0  0  0  0  0]
 [ 0  7  0  0  0  0  0  0]
 [ 0  0 10  0  0  0  0  0]
 [ 0  0  0 13  0  0  0  0]
 [ 0  0  0  0  8  0  0  0]
 [ 0  0  0  0  0 14  0  0]
 [ 0  0  0  0  0  0  8  0]
 [ 0  0  0  0  0  0  0 10]]
```

3. Yale



```
[[2 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 4 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 4 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 3 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 4]]
```

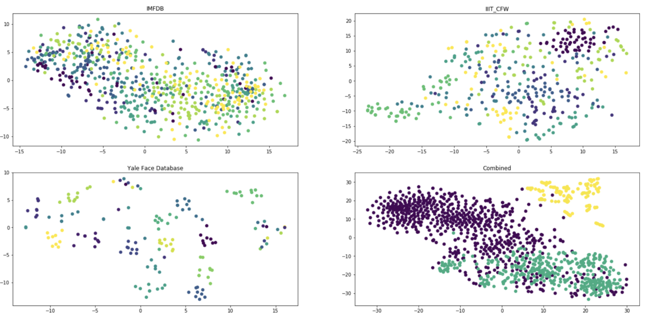
For 100 accuracy, the confusion matrix would be a diagonal matrix!

V. t-SNE based visualisation

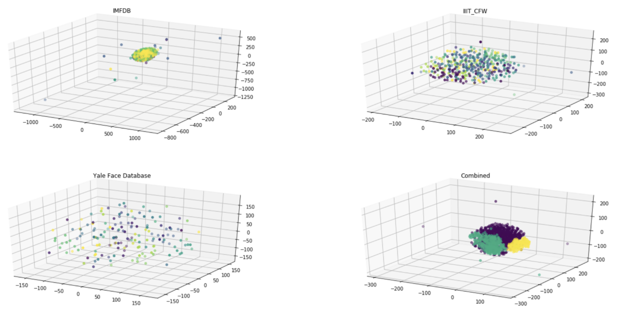
A. Visualising t-SNE, datawise and combined

Here are the reults after doing a t-SNE based visualisation on the three datasets individually and then combined.

Two dimensional:



Three dimensional:



B. Do we notice similar people coming together?

YES, the faces with similar classes do tend to come closer! This is evident from the visualisation plots above. Also, if we plot the faces in all the datasets we see formation of different clusters, hinting that faces with similar classes are closer to one another.

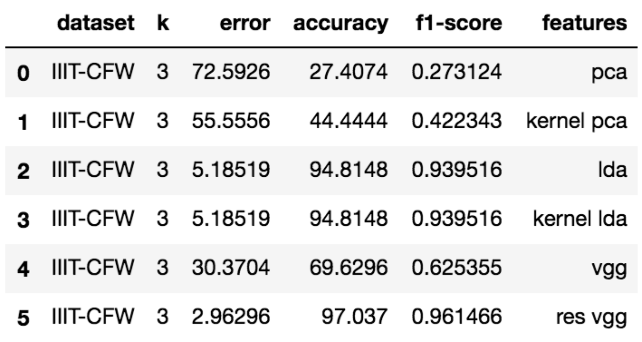
VI. Face Verification

Problem is defined as follows: input is face ID and the face image, and output is yes/no. It means that given a face image and a classID, our model should determine whether it belongs to that class and verify it.

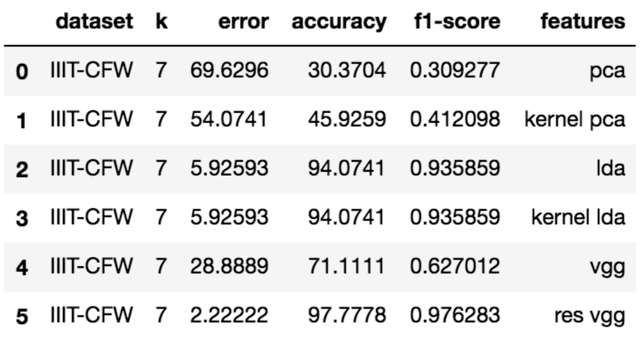
A. KNN Formulation

We now formulate this problem using KNN. We use a KNN classifier to verify the faces. We analyse the performance of face verification using Accuracy and f1 scores.

Here are the results, for IIIT dataset, with k=3 and k=7.



	dataset	k	error	accuracy	f1-score	features
0	IIIT-CFW	3	72.5926	27.4074	0.273124	pca
1	IIIT-CFW	3	55.5556	44.4444	0.422343	kernel pca
2	IIIT-CFW	3	5.18519	94.8148	0.939516	lda
3	IIIT-CFW	3	5.18519	94.8148	0.939516	kernel lda
4	IIIT-CFW	3	30.3704	69.6296	0.625355	vgg
5	IIIT-CFW	3	2.96296	97.037	0.961466	res vgg



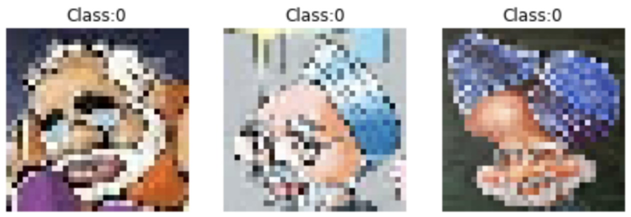
	dataset	k	error	accuracy	f1-score	features
0	IIIT-CFW	7	69.6296	30.3704	0.309277	pca
1	IIIT-CFW	7	54.0741	45.9259	0.412098	kernel pca
2	IIIT-CFW	7	5.92593	94.0741	0.935859	lda
3	IIIT-CFW	7	5.92593	94.0741	0.935859	kernel lda
4	IIIT-CFW	7	28.8889	71.1111	0.627012	vgg
5	IIIT-CFW	7	2.22222	97.7778	0.976283	res vgg

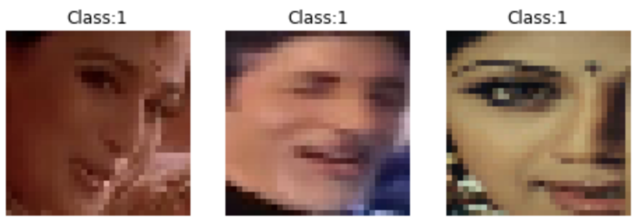
VII. Face Recognition on an Extended Dataset

We take a combination of all the 3 datasets given to us: IIIT-CFW, IMFDB and Yale, to form the new extended dataset.
This data set has many face images, but some of them are faces of real peeople whereas some are in the animated form (from the cartoon dataset). Therefore, our dataset has two classes: animated and not animated (real).

1) Real Life applications:

- This would be highly essential to check the legitimacy of people's faces and ensure that the person is real. For instance, if someone is applying for a passport, visa or any other legal document, this would ensure that no one uploads fake cartoon images.
- This would also be very helpful as filters in a search engine. So if someone searches for say, "Bon Jovi cartoon", the search engine must display those images of Bon Jovi that are in the animated form. So we must be able to differentiate between animated and real images.
- This problem is certainly not trivial, because even animated images could deceive us. There can also be multiple variations of the cartoon of the same person.





2) Implementation Pipeline:

- First, we simply combine all the three datasets. We then reassign the labels - 0 to all the animated images and 1 to the real ones.
- So we have a dataset with two clases: animated whose label is zero, and real with label 1.
- We them split this data into training set and test set.
- We apply PCA on the training set, and the fitted weights that it returns are then used to transform the test set.
- Now we are ready to classify the data! We declare an object instance of the classifier, and train the classifier on the data.
- Once training is complete, we use the trained classifier to predict the classes for the validation set.
- And finally, we compare the classifier's predictions with the actual class labels and compute accuracies!

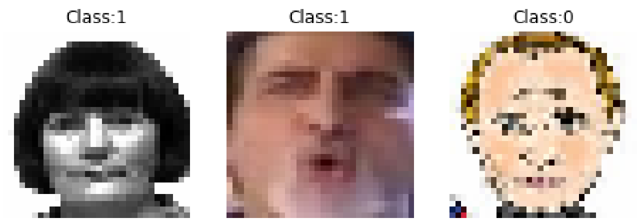
The Metrics that were used to analyse the performance of the classifer were accuracy, precision and f1 scores. Here are the results of the classification.

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       135
           1       0.96      1.00      0.98       113

    accuracy                           0.98       248
   macro avg       0.98      0.98      0.98       248
weighted avg       0.98      0.98      0.98       248
```

Here are the correctly and wrongly classified images:



worongly classified: