SMAI 2019 (Monsoon)

Assignment 2 - Report

Kripa Anne

20171159

November 4, 2019

Abstract

Three datasets were used for the analysis presented in this report. All 3 were standard datasets related to 'face' classification and verification.

1 Indian Movie Face Database
2 IIIT Cartoon Face Dataset
3 Yale Face Database

Multiple Feature Extraction Methods were used.

1 Eigen Face
2 Kernel Face
3 Fisher Face
4 Kernel Fisher Face
5 VGG Features
6 ResNet Features

Various classification techniques were also used.

1 Multi Layer Perceptron Classifier
2 Logistic Regression Classifier
3 Support Vector Machine Classifier
4 Decision Tree Classifier
5 K-Nearest Neighbors Classifier

1 Feature Extraction

1.1 What are Eigen Faces?

Eigenfaces is the name given to a set of eigenvectors used in the computer vision problem of human face recognition such that they form an orthogonal basis set from which most, or even all faces can be constructed. It's a method that is useful for face recognition and detection by determining the variance of faces in a collection of face images and use those variances to encode and decode a face in a machine learning way without the full information reducing computation and space complexity.

Specifically, the Eigenfaces are the principal components of a distribution of faces, or equivalently, the eigenvectors of the covariance matrix of the set of face images. The idea is that any face can be reconstructed from a suitable linear combination of eigenfaces. In dimensionality reduction techniques like PCA, eigenfaces are essentially the principal components that can properly describe the image that maximizes the variance between them and minimizes reconstruction error.
The motivation of Eigenfaces is twofold:

Extract the relevant (facial) information.
Represent face images efficiently.

1.2 How many eigen vectors/faces are required to "satisfactorily" reconstruct a person in these three datasets?

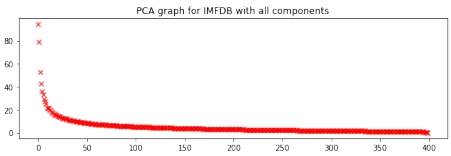

Figure 1: Eigen Spectrum of Indian Movie Face Dataset.

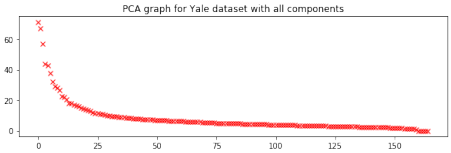

Figure 2: Eigen Spectrum of Yale Face Dataset.

In the Indian movie face dataset, the first few primary components are close to each other,and are more importance to represent a the samples.

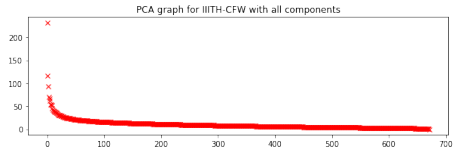

Figure 3: Eigen Spectrum of Cartoon Face Dataset.

The first couple of primary eigenvectors are able to capture a lot of the variance of the image. We need a reasonable number of principal components for satisfactory reconstruction. (around 200)

For the yale face dataset, it is a lot of the primary components show high variance, and imply a very high importance to multiple components. So fewer eigen vectors would be required for accurate reconstruction and representation. (around 120)

In the IIIT-CFW cartoon face dataset however, the disparity between the first few principal components is large, and drops quickly - we may need a large number of primary components for satisfactory reconstruction. The first component captures a lot of variance, but the remaining vectors do not. This can be attributed to the vast diveristy in images from this dataset of varying art styles, colours, non-uniform features etc. (around 300)

1.3 Which person/identity is difficult to represent compactly with fewer eigen vectors? Why is that?

	Class	Reconstruction Loss
0	Madhuri Dixit	0.014247
1	Kajol	0.018969
2	Shah Rukh Khan	0.017094
3	Shilpa Shetty	0.020135
4	Amitabh Bacchan	0.021857
5	Katrina Kaif	0.015487
6	Akshay Kumar	0.018697
7	Aamir Khan	0.020181

Figure 4: Reconstruction Losses Per Face in the IMFDB Dataset

As we see from Figure 4, the reconstruction loss for faces for an 90% representation of the original face, the loss is highest for Class 4 - Amitabh Bacchhan's face, followed very closely by Class 7 - Aamir Khan. It is easy to understand this empirically, and intuitively we can realise this by studying the data for these two actors.

The pictures for Amitabh Bacchan are very varied in the angles they are taken from. The faces are not well centred - and the centring of data is important in Eigen Face detection. They seem to have been taken in different lighting conditions, and from different movies. Similarly for Aamir Khan, his entire face is not often visible, and sometimes his hand appears in the image.

Interestingly, if we increase the number of components in the Eigen Face Representation to 50 (min(n_samples, n_features)=50), then Class 1 - Kajol becomes the largest reconstruction loss.

For the Yale dataset, Max reconstruction error at class = 11. For the IIIT-CFW dataset, Max reconstruction error at class = 4.

1.4 Which dataset is difficult to represent compactly with fewer eigen vectors? Why is it so?

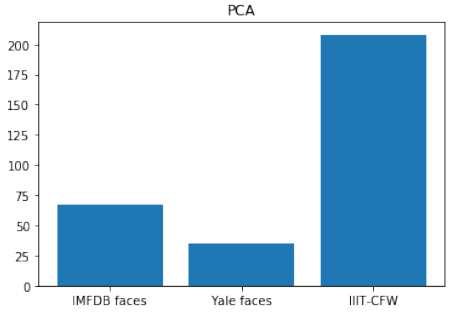

Figure 5: No of Components to Reconstruct Faces - PCA

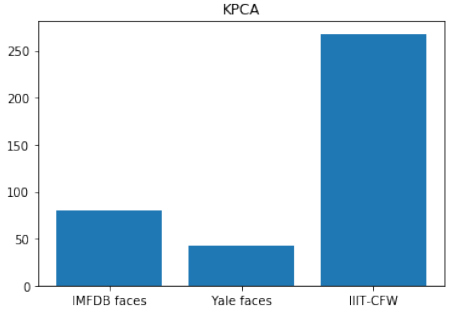

Figure 6: No of Components to Reconstruct Faces - KPCA

To achieve a representation of about 50% of the original image from the Indian Movie Face, we need the first 3 components, but to reach

a satisfactory 90% representation we need the first 70 components.

To achieve a representation of about 50% of the original image from the Yale Faces, we need the first 5 components, but to reach a satisfactory 85% representation we need the first 30 components.

For IIIT-CFW, because each separate component only manages to capture small variances, to achieve a representation of about 50% of the original cartoon face image we need the first 7 components, but to reach a satisfactory 90% representation we need over 180 components.

2 Classification

2.1 Use different classifiers and find the classification accuracy. Which method works well? Do a comparitive study.

We have employed 4 different classifiers - MLP, Logistic Regression, SVM and Decision tree classifiers. Pairing each of them with 8 different feature extraction methods (PCA, KPCA, LDA, KLDA, ResNet, VGG, PCA+KPCA, LDA+KLDA), the results for accuracies and F1 scores have been summarised in the tables below.

	Method	Dimensions	Classification Error	Accuracy	f1-score
0	pca with LR	67	0.1625	0.8375	0.837068
1	pca with SVM	67	0.2000	0.8000	0.815293
2	pca with MLP	67	0.2750	0.7250	0.755749
3	pca with Dtree	67	0.5375	0.4625	0.495102
4	kpca with LR	80	0.4375	0.5625	0.597754
5	kpca with SVM	80	0.9375	0.0625	0.014706
6	kpca with MLP	80	0.2375	0.7625	0.770836
7	kpca with Dtree	80	0.5250	0.4750	0.502350
8	lda with LR	6	0.0250	0.9750	0.965753
9	lda with SVM	6	0.0500	0.9500	0.955839
10	lda with MLP	6	0.0125	0.9875	0.989852
11	lda with Dtree	6	0.0375	0.9625	0.964110
12	klda with LR	6	0.0000	1.0000	1.000000
13	klda with SVM	6	0.0375	0.9625	0.962372
14	klda with MLP	6	0.0250	0.9750	0.970642
15	klda with Dtree	6	0.0375	0.9625	0.962500
16	vgg with LR	4096	0.0875	0.9125	0.903760
17	vgg with SVM	4096	0.1000	0.9000	0.908890
18	vgg with MLP	4096	0.1750	0.8250	0.834230
19	vgg with Dtree	4096	0.2375	0.7625	0.776894
20	resnet with LR	2048	0.0500	0.9500	0.950783
21	resnet with SVM	2048	0.0500	0.9500	0.947180
22	resnet with MLP	2048	0.0750	0.9250	0.928699
23	resnet with Dtree	2048	0.0625	0.9375	0.934598
24	pca+kpca with LR	147	0.2500	0.7500	0.763750
25	pca+kpca with SVM	147	0.2125	0.7875	0.792803
26	pca+kpca with MLP	147	0.2125	0.7875	0.786173
27	pca+kpca with Dtree	147	0.5125	0.4875	0.498318
28	lda+klda with LR	12	0.0125	0.9875	0.986805
29	lda+klda with SVM	12	0.0000	1.0000	1.000000
30	lda+klda with MLP	12	0.0250	0.9750	0.976607
31	lda+klda with Dtree	12	0.0625	0.9375	0.938128

Figure 7: Classification Results for Indian Movie Faces Dataset

	Method	Dimensions	Classification Error	Accuracy	f1-score
0	pca with LR	35	0.121212	0.878788	0.819068
1	pca with SVM	35	0.242424	0.757576	0.780957
2	pca with MLP	35	0.090909	0.909091	0.933036
3	pca with Dtree	35	0.454545	0.545455	0.536902
4	kpca with LR	42	0.575758	0.424242	0.554157
5	kpca with SVM	42	0.939394	0.060606	0.018997
6	kpca with MLP	42	0.090909	0.909091	0.877966
7	kpca with Dtree	42	0.303030	0.696970	0.571988
8	lda with LR	10	0.000000	1.000000	1.000000
9	lda with SVM	10	0.060606	0.939394	0.959954
10	lda with MLP	10	0.000000	1.000000	1.000000
11	lda with Dtree	10	0.090909	0.909091	0.928533
12	klda with LR	10	0.000000	1.000000	1.000000
13	klda with SVM	10	0.090909	0.909091	0.946307
14	klda with MLP	10	0.000000	1.000000	1.000000
15	klda with Dtree	10	0.151515	0.848485	0.830400
16	vgg with LR	4096	0.303030	0.696970	0.696466
17	vgg with SVM	4096	0.606061	0.393939	0.516955
18	vgg with MLP	4096	0.272727	0.727273	0.688710
19	vgg with Dtree	4096	0.515152	0.484848	0.475532
20	resnet with LR	2048	0.000000	1.000000	1.000000
21	resnet with SVM	2048	0.000000	1.000000	1.000000
22	resnet with MLP	2048	0.000000	1.000000	1.000000
23	resnet with Dtree	2048	0.151515	0.848485	0.862870
24	pca+kpca with LR	77	0.090909	0.909091	0.924791
25	pca+kpca with SVM	77	0.303030	0.696970	0.682581
26	pca+kpca with MLP	77	0.090909	0.909091	0.820929
27	pca+kpca with Dtree	77	0.424242	0.575758	0.596176
28	lda+klda with LR	20	0.000000	1.000000	1.000000
29	lda+klda with SVM	20	0.121212	0.878788	0.946092
30	lda+klda with MLP	20	0.000000	1.000000	1.000000
31	lda+klda with Dtree	20	0.181818	0.818182	0.794829

Figure 8: Classification Results for Yale Face Dataset

	Method	Dimensions	Classification Error	Accuracy	f1-score
0	pca with LR	208	0.525926	0.474074	0.473988
1	pca with SVM	208	0.407407	0.592593	0.608961
2	pca with MLP	208	0.496296	0.503704	0.482420
3	pca with Dtree	208	0.770370	0.229630	0.225856
4	kpca with LR	268	0.474074	0.525926	0.439344
5	kpca with SVM	268	0.881481	0.118519	0.026490
6	kpca with MLP	268	0.474074	0.525926	0.498910
7	kpca with Dtree	268	0.688889	0.311111	0.294273
8	lda with LR	6	0.022222	0.977778	0.974261
9	lda with SVM	6	0.022222	0.977778	0.973603
10	lda with MLP	6	0.022222	0.977778	0.973375
11	lda with Dtree	6	0.066667	0.933333	0.929014
12	klda with LR	6	0.037037	0.962963	0.955327
13	klda with SVM	6	0.044444	0.955556	0.955326
14	klda with MLP	6	0.088889	0.911111	0.905243
15	klda with Dtree	6	0.059259	0.940741	0.930565
16	vgg with LR	4096	0.281481	0.718519	0.719562
17	vgg with SVM	4096	0.355556	0.644444	0.579492
18	vgg with MLP	4096	0.362963	0.637037	0.588131
19	vgg with Dtree	4096	0.370370	0.629630	0.609769
20	resnet with LR	2048	0.014815	0.985185	0.975755
21	resnet with SVM	2048	0.037037	0.962963	0.966546
22	resnet with MLP	2048	0.059259	0.940741	0.924953
23	resnet with Dtree	2048	0.037037	0.962963	0.965402
24	pca+kpca with LR	476	0.459259	0.540741	0.553807
25	pca+kpca with SVM	476	0.392593	0.607407	0.620762
26	pca+kpca with MLP	476	0.414815	0.585185	0.566701
27	pca+kpca with Dtree	476	0.666667	0.333333	0.335189
28	lda+klda with LR	12	0.066667	0.933333	0.930978
29	lda+klda with SVM	12	0.037037	0.962963	0.957440
30	lda+klda with MLP	12	0.022222	0.977778	0.978396
31	lda+klda with Dtree	12	0.096296	0.903704	0.893511

Figure 9: Classification Results for Cartoon Face Dataset

2.2 For each dataset print the confusion matrix for the best model.

```
For IMFDB, LDA with LR:
[[12  0  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0  0]
 [ 0  0  6  0  0  0  0  0]
 [ 0  0  0 12  0  0  0  0]
 [ 0  0  0  0 13  0  0  0]
 [ 0  0  0  0  0  7  0  0]
 [ 0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0 10]]
```

```
For Yale, LDA+KLDA with LR:
[[4 0 0 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]
 [0 0 2 0 0 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 0 0 3 0 0 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 0 0 4 0 0 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 0 0 1 0 0 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 0 0 3 0 0 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 0 0 2 0 0]
 [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 0 0 2]]
```

```
For IIIT-CFW, resnet with LR:
[[ 9  0  0  0  0  0  0  0]
 [ 0  4  0  0  0  0  0  0]
 [ 0  0 14  0  0  0  1  0]
 [ 0  0  0 26  0  0  0  0]
 [ 1  0  0  0 18  0  0  0]
 [ 0  0  0  0  0 18  0  0]
 [ 0  0  0  0  0  0 20  0]
 [ 0  0  0  1  0  0  0 23]]
```

Figure 10: IMFDB - Logistic Regression + LDA (100% Accuracy)
Yale - Logistic Regression + (LDA+KLDA) (100% Accuracy) Cartoon - Logistic Regression + ResNet Features (97.8396% Accuracy

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other.

For the IMFDB Dataset, Logistic Regression + LDA gave perfect classification. For the Yale Dataset, LDA with SVM, KLDA with LR, KLDA with MLP, PCA+KPCA with LR, LDA+KLDA with LR, LDA+KLDA with MLP gave a 100% accuracy. For the Cartoon Dataset, Logistic Regression + ResNet Features gave the highest accuracy of 97.8396%.

3 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets.

3.1 Do t-SNE based visualization of faces.

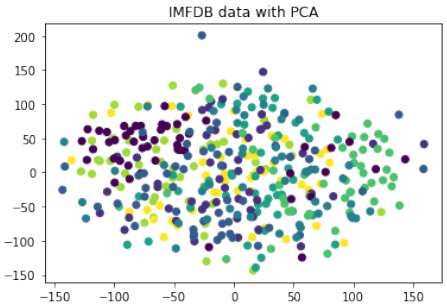

Figure 11: t-SNE 2D Plot of Eigen Faces for IMFDB Dataset

As we can see from these two plots, t-SNE is a method to group similar entities closer to one another. t-SNE benefits from better compression and feature representation - better feature representations are clearly grouped more closely.

The Eigen Faces (PCA) are sparsely grouped, while the Fisher Faces (Kernel LDA) are more closely grouped.

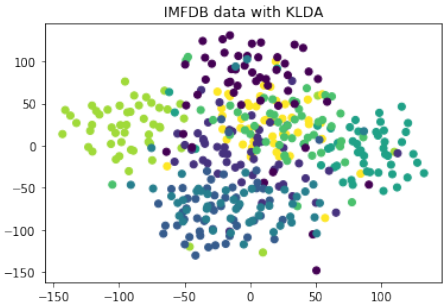

Figure 12: t-SNE 2D Plot of Fisher Faces for IMFDB Dataset

3.2 Does it make sense? Do you see similar people coming together? Or something else?

Yes, we can see similar faces clustering together in a sector, and these features allow us to perform classification with great accuracy.

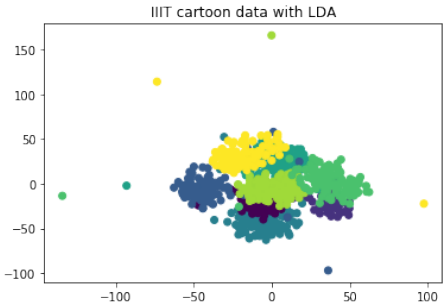

Figure 13: t-SNE 2D Plot for IIIT cartoon faces Dataset

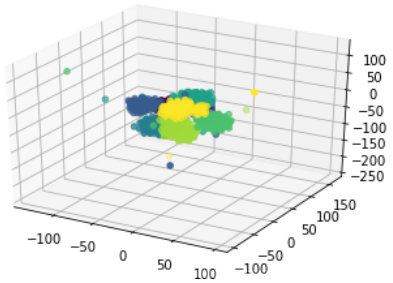

Figure 14: t-SNE 3D Plot for IIIT cartoon faces Dataset

We can clearly identify and separate 8 clusters, which are the data points from different classes grouping together in the t-SNE vector space.

It might not be very useful for some other cases where the data points cannot be very closely grouped. This can happen when people who look similar exist in the dataset. We see this happening in the Yale face dataset.

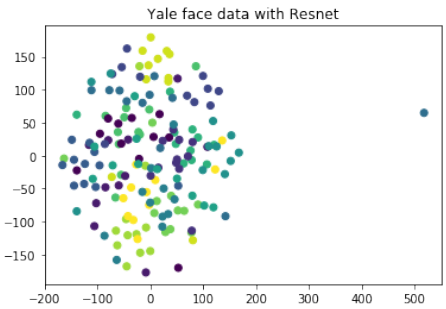

Figure 15: t-SNE 2D Plot for Yale face Dataset (ResNet)

4 Face Verification

4.1 How do we formulate the problem using KNN?

We can formulate this problem into one that a K-Nearest Neighbor Classifier can work with, and can perform face verification. We map our data points to feature spaces that can separate data with high variance (including some feature representations we have used earlier), and aggregate the k-closest data points using some similarity metric (Euclid's Distance, Jaccard Distance etc), and classify the validation data according to the nearest neighbors from the training set.

4.2 How do we analyze the performance? Suggest the metrics (like accuracy) that is appropriate for this task.

Face Verification is a critical biometric tool for security purposes, and minimising false negatives is of high priority. Due to this, metrics of precision is just as important as accuracy. We can also use F1 scores as a metric.
f1 = 2 * (precision * recall) / (precision + recall)

4.3 Show empirical results with all the representations.

	Method	Dimensions	Classification Error	Accuracy	Precision
0	pca with KNN at NN=3	67	0.4250	0.5750	0.5750
1	pca with KNN at NN=5	67	0.3625	0.6375	0.6375
2	pca with KNN at NN=17	67	0.4500	0.5500	0.5500
3	pca with KNN at NN=25	67	0.5625	0.4375	0.4375
4	kpca with KNN at NN=3	80	0.2875	0.7125	0.7125
5	kpca with KNN at NN=5	80	0.4000	0.6000	0.6000
6	kpca with KNN at NN=17	80	0.4500	0.5500	0.5500
7	kpca with KNN at NN=25	80	0.5250	0.4750	0.4750
8	lda with KNN at NN=3	6	0.0125	0.9875	0.9875
9	lda with KNN at NN=5	6	0.0250	0.9750	0.9750
10	lda with KNN at NN=17	6	0.0000	1.0000	1.0000
11	lda with KNN at NN=25	6	0.0500	0.9500	0.9500
12	klda with KNN at NN=3	6	0.0250	0.9750	0.9750
13	klda with KNN at NN=5	6	0.0250	0.9750	0.9750
14	klda with KNN at NN=17	6	0.0125	0.9875	0.9875
15	klda with KNN at NN=25	6	0.0250	0.9750	0.9750
16	vgg with KNN at NN=3	4096	0.0625	0.9375	0.9375
17	vgg with KNN at NN=5	4096	0.0875	0.9125	0.9125
18	vgg with KNN at NN=17	4096	0.1625	0.8375	0.8375
19	vgg with KNN at NN=25	4096	0.1000	0.9000	0.9000
20	resnet with KNN at NN=3	2048	0.0500	0.9500	0.9500
21	resnet with KNN at NN=5	2048	0.0500	0.9500	0.9500
22	resnet with KNN at NN=17	2048	0.0500	0.9500	0.9500
23	resnet with KNN at NN=25	2048	0.0125	0.9875	0.9875
24	pca+kpca with KNN at NN=3	147	0.3625	0.6375	0.6375
25	pca+kpca with KNN at NN=5	147	0.4000	0.6000	0.6000
26	pca+kpca with KNN at NN=17	147	0.4750	0.5250	0.5250
27	pca+kpca with KNN at NN=25	147	0.5000	0.5000	0.5000
28	lda+klda with KNN at NN=3	12	0.0375	0.9625	0.9625
29	lda+klda with KNN at NN=5	12	0.0000	1.0000	1.0000
30	lda+klda with KNN at NN=17	12	0.0250	0.9750	0.9750
31	lda+klda with KNN at NN=25	12	0.0375	0.9625	0.9625

Figure 16: KNN Results for Indian Movie Faces Dataset

The above table has classification results for KNN using different feature extraction methods

and with different values of k (3,5,27,25) for the IMFDB dataset. The remaining results for the remaining datasets can be seen in the notebook. K-Nearest Neighbors is not as suitable for classification as some of the other methods discussed earlier. Images represent larger spaces, and a lot of face images have a lot of overlapping data points. Picking the optimal K for K-nearest Neighbors becomes empirical for a good KNN Classifier.

5 Gender Classifier

5.1 What it is

This is a classifier that can identify and distinguish between male and female faces. It was trained on the IMFDB and the IIIT-CFW dataset and can distinguish the genders of cartoon and real faces. It is composed of several independent classification methods, all of which operate on different features extracted from the images, to reduced dimension spaces. It has an 80-20 split for training and test data respectively.

5.2 Qualitative Results

The best performing models were KLDA with SVM and KLDA with Logistic Regression with accuracy of 100

	Method	Dimensions	Classification Error	Accuracy	f1-score
0	pca with LR	175	0.144186	0.855814	0.830399
1	pca with SVM	175	0.120930	0.879070	0.837778
2	pca with MLP	175	0.130233	0.869767	0.826766
3	pca with Dtree	175	0.311628	0.688372	0.641998
4	kpca with LR	243	0.260465	0.739535	0.655223
5	kpca with SVM	243	0.274419	0.725581	0.420485
6	kpca with MLP	243	0.195349	0.804651	0.752189
7	kpca with Dtree	243	0.269767	0.730233	0.659510
8	lda with LR	1	0.013953	0.986047	0.982692
9	lda with SVM	1	0.004651	0.995349	0.993965
10	lda with MLP	1	0.009302	0.990698	0.988455
11	lda with Dtree	1	0.018605	0.981395	0.977752
12	klda with LR	1	0.000000	1.000000	1.000000
13	klda with SVM	1	0.000000	1.000000	1.000000
14	klda with MLP	1	0.013953	0.986047	0.982951
15	klda with Dtree	1	0.023256	0.976744	0.970986
16	pca+kpca with LR	418	0.246512	0.753488	0.712536
17	pca+kpca with SVM	418	0.148837	0.851163	0.784089
18	pca+kpca with MLP	418	0.116279	0.883721	0.856173
19	pca+kpca with Dtree	418	0.306977	0.693023	0.622669
20	lda+klda with LR	2	0.004651	0.995349	0.994541
21	lda+klda with SVM	2	0.004651	0.995349	0.994283
22	lda+klda with MLP	2	0.000000	1.000000	1.000000
23	lda+klda with Dtree	2	0.000000	1.000000	1.000000

Figure 17: Classification results

Their cross-validation scores:
At k=4:
Score with KLDA and SVM: 0.993477
Score with KLDA and LR: 0.993477
At k=30:
Score with KLDA and SVM: 0.993518
Score with KLDA and LR: 0.993518
The t-SNE visualisation presents a lot of information. We can see how the classes are differentiated and distinct in the t-SNE plot of the original data. We also see the benefits of transforming to a smaller space (like Kernel LDA) helps us further clearly define the classes.

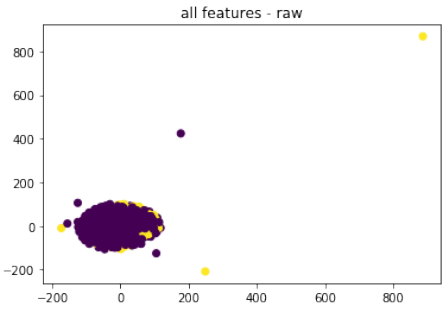

Figure 18: 2D t-SNE Plot on Original Data

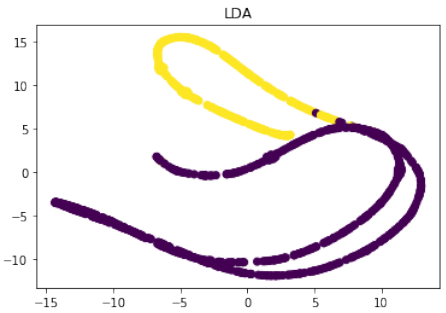

Figure 19: 2D t-SNE Plot on Fisher Face

5.3 Quantitative Results

Figure 20: Misclassified Data Point

Figure 21: Correctly Classified Data Point

The KLDA and Decision Tree model had an unsatisfactory accuracy, and images such as a drawing of Aishwarya Rai's partial face got misclassified as a male. However, pictures with more prominent features got classified correctly.

This problem is important as it is a vital part in the pipeline of a standard face-recognition algorithm which can be used for many purposes including security at public locations, monitoring, device security etc. It is not a trivial problem as the human face is diverse in features and there are no predominantly masculine or feminine features. Additionally, facial gestures and emotions can distort the face and reduce its accuracy. To improve our model, we would require a much larger and diverse training set with different emotions, different accessories on the face etc to help our model learn better.