Comparison of Feature Reduction Approaches and Classification Approaches for Pattern Recognition

Xiaoyang Rebecca LI, May,2016
Electrical and Computer Engineering Department, University of Houston
Houston, Texas
Xiaoyang.rebecca.li@gmail.com

***Abstract*— Pattern recognition on hyper-dimensional data could be divided as feature reduction and classification. Feature reduction projections and classifier models are learned by training dataset and applied to classify testing dataset. A few approaches of feature reduction have been compared in this paper: principle component analysis (PCA), linear discriminant analysis (LDA) and their kernel methods (KPCA,KLDA). Correspondingly, a few approaches of classification algorithm are implemented: Support Vector Machine (SVM), Gaussian Quadratic Maximum Likelihood and K-nearest neighbors (KNN) and Gaussian Mixture Model (GMM). Our experiments shows that SVM performed the most robust to the increasing of dimensional space, and SVM and LDA are more sensitive to noises. The Matlab Code is available in github.**

Keywords*— *Pattern Recognition; dimension reduction; classification; PCA; LDA; kernel; KNN; SVM; GMM

I. THE DICRIPTION OF THE PROJECT

We are provided with a hyperspectral dataset for this project. The samples in the dataset are pixels from a hyperspectral image. Hyperspectral images are data-cubes of size *m* by *n* by *d*, where *m* and *n* are spatial dimensions of the image, and *d* is the number of spectral channels in the image (e.g. in traditional color images, d=3). We have extracted pixels (samples) from a hyperspectral image for this project, and the data that is being provided has two components:

- Training data (of size: 750 samples/pixels by 144 channels/dimensions)
- Testing data (of size: 12197 samples/pixels by 144 channels/dimensions)

Our task is to setup and validate pattern recognition systems (including feature projection, and classification) that use training data (samples/pixels) to learn the models (e.g. feature projection projections, classifier models etc.), and apply them to classify testing data (samples/pixels), as Figure 1 shows. Since pixels have already been extracted and assigned to training and test data, the raw (input) feature space will be a *d*-dimensional space representing the spectral response across the *d* channels, and all training and test pixels will reside in this space.

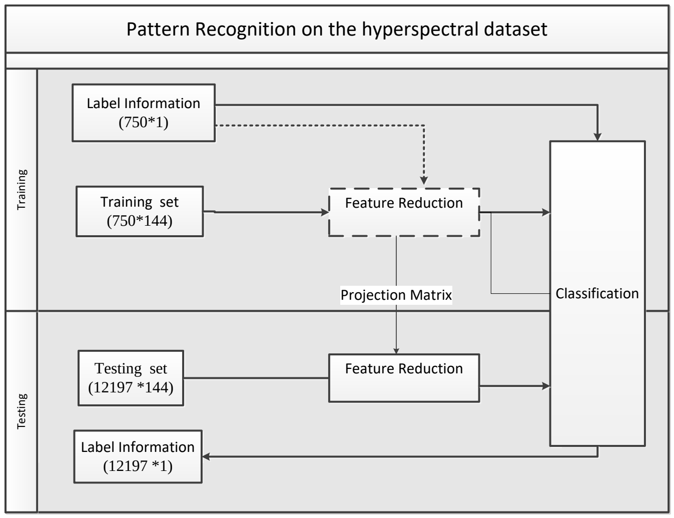


Fig. 1. The flow chart of Pattern Recognition in this project

We have applied a few feature reduction algorithms, classify the data and validate the results, as indicated in Figure 2. We have reported results with each combination of feature reduction and classification algorithm.

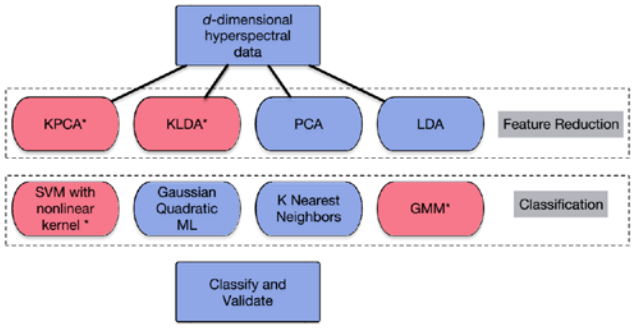


Fig. 2. The algorithms implemented in this project

II. Feature Reduction Algorithms

Feature reduction could be regarded as a preprocess procedure for input dataset (both training and testing data) to remove irrelevant and redundant features. Dimensionality reduction is needed when the following classification algorithms are not efficient or invalid to process high dimensional data. For example, in Gaussian parameter estimation classification algorithms including Gaussian maximum likelihood and Gaussian mixture model (GMM), the number of parameters (mean and covariance matrix to be estimated) learned from training set is highly restricted by the number of dimensions. Other classification algorithms such as K nearest neighborhood relied on distance measurements interpret weak neighbor relationship of each point in high dimensional Euclidean space. However, not all the classification methods need feature reduction. The efficiency of Neural Network learning algorithms are barely influenced by dimensionality of input data only determining the number of neurons in input layer. Support Vector Machine (SVM) only concerns about the support vectors near decision boundaries, so dimension of vectors has minor restriction on implementation.

Dimensionality reduction techniques can be divided into feature projection approaches and feature selection approaches. Projection approaches transform original features into a new feature space with lower dimensionality and the new constructed features are usually combinations of original features. Examples of feature projection techniques include Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA). Selection Approaches aim to select a small subset of features that minimize redundancy and maximize relevance to the target such as the class labels in classification. Representative feature selection techniques include Information Gain, Relief, Fisher Score and Lasso [1].

A. *Principle Component Analysis (PCA)*

Principle Component analysis aims to project the raw data into a new feature space that data spread out mostly in some directions. The projected directions are obtained by minimizing the mean square error (MSE) between projected data and original data. The criterion function in the MSE sense is minimized when the vectors of projected directions having the largest eigenvalues of the scatter matrix.

Consider a *D*-dimensional dataset *X* is projected to a *d*-dimensional space ($d \ll D$), the number of data in this dataset is *N*.

$$X^{\mathbb{R}^D} \rightarrow Y^{\mathbb{R}^d}$$

The procedures of PCA are as follows:

- Centralizing the data : Compute the mean value of dataset $M_{N\times D}$ and shift the data into center by subtracting mean value from original data:

$$Xctr_{N\times D} = X_{N\times D} - M_{N\times D} \quad (1)$$

- Generating scatter matrix : Obtain covariance matrix S buy computing dot product of centralized data.

$$S_{D\times D} = Xctr_{N\times D}^{t} \cdot Xctr_{N\times D} \quad (2)$$

- Computing Eigen value and Eigen vector: Decompose the scatter matrix to obtain Eigen vector matrix $U_{D\times D}$ and Eigen vector diagonal matrix $\Lambda_{D\times D}$ composed by eigen values .

$$S_{D\times D} \cdot U_{D\times D} = \Lambda_{D\times D} \cdot U_{D\times D}^{t} \quad (3)$$

  Eigen values represent the significance of the direction, and Eigen vectors indicate the corresponding unit directional vector. The projection matrix $U'_{D\times d}$ is obtained by cutting out the Eigen vectors to the correspond *d* largest value of Eigen values.

- Projecting the data: Project the data from *D*-dimensional space to *d*-dimensional space by using projected matrix.

$$Y_{N\times d} = X_{N\times D} \cdot U'_{D\times d} \quad (4)$$

Although PCA is a popular tool for dimensionality reduction, it is not optimal (in general) for pattern classification tasks because it is an unsupervised learning method that doesn't care about class label information.

B. *Linear Discriminant Analysis (LDA)*

In order to implement feature reduction related with classification, LDA is designed to find a projection matrix that maximizing between-class covariance and minimizing within-class covariance. Consider a *D*-dimensional dataset *X* is projected to a *d*-dimensional space ($d \ll D$), the number of data in this dataset is *N*. The number of class is C, the number of samples in each class subset D_i is n_i.

- Computing between-class scatter matrix S_b: Compute the mean value of each class $M_i$and mean value of all overall *M*.

$$S_b = \sum_{i=1}^{C} n_i (M_i - M) \cdot (M_i - M)^t \quad (5)$$

- Computing within-class Scatter Matrix S_b:

$$S_w = \sum_{i=1}^{C} \sum_{X\in D_i} (X - M_i) \cdot (X - M_i)^t \quad (6)$$

- Generating scatter matrix : Obtain covariance matrix S buy computing dot product of centralized data.

$$S_{D\times D} = S_w^{-1} \cdot S_b \quad (7)$$

The following steps are the same as PCA. The only difference of LDA and PCA is the generation of scatter matrix *S*. In PCA, as (1) and (2) indicating, the scatter matrix is generated from input data itself, while in LDA *S* is generated from input data and their class label information. Since only *C*-

1 of between-class scatter matrix is independent, the highest dimension of LDA to reduce is *C-1*.

C. *Kernel Principle Component Analysis (KPCA) and Kernel Linear Discriminant Analysis (KLDA)*

Linear representation methods like LDA and PCA have problem of projection to a new space when the data do not agree well with a linear representation model. To obtain a nonlinear generalization, kernel trick is designed to project data in to a nonlinear kernel space where data are linearly separable without explicitly mapping. It is implemented by replacing the inner product term of linear approach with a nonlinear kernel function, which defined by

$$K(X,X') = \Phi(x) \cdot \Phi(x)^t \quad (7)$$

In linear approaches, projection matrix is obtained by Eigen decomposition on scatter matrix $S_{D\times D}$, while in kernel approaches, the object of Eigen decomposition is *1/N* kernel matrix.

The only difference of KPCA and KLDA is that in KPCA, centralization is need after kernel matrix has been constructed, while in KLDA we don't.

III. Classificaiton Algorithms

A. *Gassian Maximum likelyhood (Gaussian ML)*

The idea of maximum likelihood estimation is to assign (estimate) a value of parameter θ which happens in the most possibility:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, l(\theta) \quad (8)$$

If we assume all the data x_k in dataset D are identical independently Gaussian distributed ($x_k \sim \mathcal{N}(\mu, \Sigma)$), then the probability that dataset D give observation of estimating parameter $\theta = \{\mu, \Sigma\}$ could be represented as:

$$l(\theta) = \ln[\textstyle\prod_{k=1}^{n} \mathcal{N}(x_k|\{\mu,\Sigma\})] \quad (9)$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \ln[\textstyle\prod_{k=1}^{n} \mathcal{N}(x_k|\{\mu,\Sigma\})] \quad (10)$$

By solving (10), the estimated value of μ, Σ could be represented as the mean of all samples and variance of values in samples to estimated mean. Considering a *D*-dimensional data, the discriminant function is generated by summing up logarithmic likelihood function according to estimated value of μ, Σ of each class and logarithmic prior probability of each class, as (11) indicating:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) - \frac{D}{2} ln 2\pi - \ln|\Sigma_i| + lnP(w_i) \quad (11)$$

We could even find from (11) that there are (D+D/2) parameters to be estimated for each class, which means that if the number of training sample is not large enough, the parameters cannot be well-estimated. This is one of the reasons why feature reduction is needed in classification problem, as we noted in the beginning of chap II.

B. *Gaussian Mixture Model(GMM)*

Gaussian Mixture Model (GMM) is a clustering algorithm to estimate probability function by regarding them us finite linear combination of Gaussian Model in which the weights of each Gaussian component is defined as prior probability of each component [3]. To obtain the optimal GM parameters we generally use Expectation-Maximum (EM) optimization approach rather that maximum likely hood estimation because it avoid infinitely possibility problem for some sparsely distributed single points .

Since GMM are widely used to cluster samples in to K clusters without given information of class label (C classes) of each data, in order to make use of the class label information, i.e. modify it to solving supervised classification problem, we cluster the data in each class individually. The procedure of supervised GMM is as follows:

1) *Divide the training dataset:* Obtain subset D_i that the samples in this subset all labeled in class *i*.
2) *Generate GMM optimal parameters for each subset* D_i*:*
   a) *Initalization:* Use K-means clustering method to find the center μ_0 for each component. Calculate Σ_o and priority possibility $P_0(c)$ according to the centers by K-means
   b) *Estimation step(E-step):* Use initalized parameters to calulate the possibility that data x comes from compent C: $P(Z_i = C)$
   c) *Maximization step(M-step):* Fix the $P(Z_i = C)$, find the GM parameters $\{ P(C), \mu_c, \Sigma_c \}$using expected likelihood
   d) *Iteration:* Take the estimiated GM parameters back to E-step until the result is convergence.
3) *Generate Discriminate functions according to Bayes Probability Rules:* Put the optimal GM parameters $\{ P(C), \mu_c, \Sigma_c \}$ into discriminate funciton (11), in which $P(C) = P(w_i)$.

The number of clusters K should be selected very carefully. If K is too small, the GMM is unable to well capture the distribution of data, (especially when K=1, GMM will degenerate to Gaussian ML). On the other hand, if K is too large, except for computation complexity, a severe overfitting problem will result, which means that we manually divide one component into several separated parts.

C. K- nearest neighborhood (KNN)

Nonparametric classification techniques are a type of estimation methods that bypass probability and go directly to posterior probability estimation rather than determine the posterior function by parameter estimation. The density P of a region R (volume is V) that there are K points fall in this region could be presented as *K/N*, in which N is the number of point in all regions. The density p_n(x) of a data point is *P/V, i.e. K/ (N*V).*

As one of the simplest algorithm of nonparametric classification techniques, K-nearest neighborhood method directly uses the approximated form p_n (x) to calculate posterior probability for class w_i:

$$p_n(w_i|\mathrm{x})=\frac{p_n(x|w_i)}{\sum_i^c p_n(x|w_i)}=\frac{K_i}{K_n} \qquad (12)$$

The intuitive interpretation of (12) is that we claim a new data as class w_i when the number of original samples labeled w_i is the most dominated for all K_n samples. K_n is an indicator that determine the range of neighborhood region. Specifically, we define $K_n = K_1\sqrt{n}$, in which K_1 is the initial size of analysis window that designated by user.

D. Support Vector Machine (SVM)

Support Vector Machine is a learning machine to find linear decision boundary for each class. Support vectors are the data points (marked in squares in Figure 1) that define the margin of largest separation between the two classes. We can see that we don't need to concern about all the data but only care about the support vectors that can completely determine the hyperplane and margins.

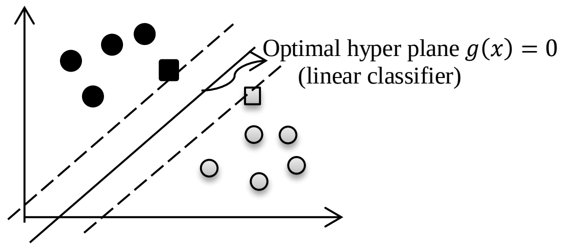


Fig. 3. Example of a a separable problem in a 2dimensional space

Suppose a set of data are linearly separable, so there is a hyper plane $g(x) = 0$ as a linear classifier that completely discriminate to two classes, in which discriminate function is defined as (12),

$$g(x) = \omega \cdot x + b \qquad (13)$$

It is positive when data is in class 1, negative for class 2. We then define a margin for a data *x* as *d(x),* it is obtained by

$$d(x) = \frac{|\omega \cdot x + b|}{\sqrt{||\omega|^2|}} \qquad (14)$$

There are many way to draw a linear classifier, but only the one that have maximum margin distance to both of the two classes is the optimal linear classifier. So the corresponding optimal margin is found by the minimum of margins. The discriminate function (12) could be completely determined by (15)

$$\underset{\omega,b}{\operatorname{argmax}} \min_{for\ all\ x_i} d(x_i) \qquad (15)$$

In more complex occasion , if the input vectors are non-linearly separable, kernel trick could be used to map original data to a high dimension feature space that data are linearly separable in that space. This occasion is the commonly happen in real data because it is quite often that data in the original space is not linearly separable, so if without extra notation, the template of SVM is default to kernel one.

IV. Experimental setup

We are using feature reduction methods as preprocessing to all the data including training data and testing data. Projected data then sent to classification procedure. As Figure 2 shows, 4 feature reduction method and 4 classification methods generated 16 pattern recognition approach combinations.

It is also worth to be mentioning that the kernel function of feature reduction and classification could not be the same because Kernel trick of feature reduction aim to project the raw data to a lower space that data are linearly separable as much as possible, while in pattern recognition is project to a higher space for the same goal. In our project, for Kernel function of KPCA and KLDA, we use polynomial function, and use Gaussian distribution as the kernel function for SVM.

In our experiment, we are observing the performance of classification by accuracy of correct classify, confusion matrix and receiver operating characteristics (ROC) for each feature reduction and classification approach combination.

A. The examine on dimensionality of feature reduction

As Chap.II and Chap.III narrates, feature reduction is needed especially for Gaussian classification approach including Gaussian ML and (GMM). Moreover, the number of training samples in our project (750 samples), highly restricted the largest dimensionality we could project. So in our project, the maximum dimensionality we select is 50, specifically, since largest dimensionality for LDA and KLDA method to be projected is *C-1*(C is the number of class, in our project is 15), the upper bounds for those two we select 14.

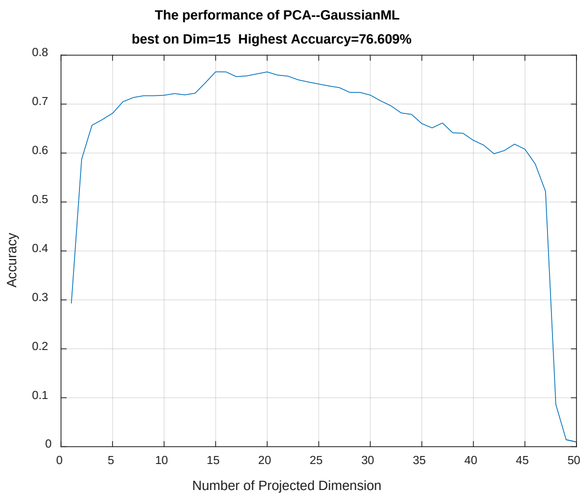


Fig 4.(1a)

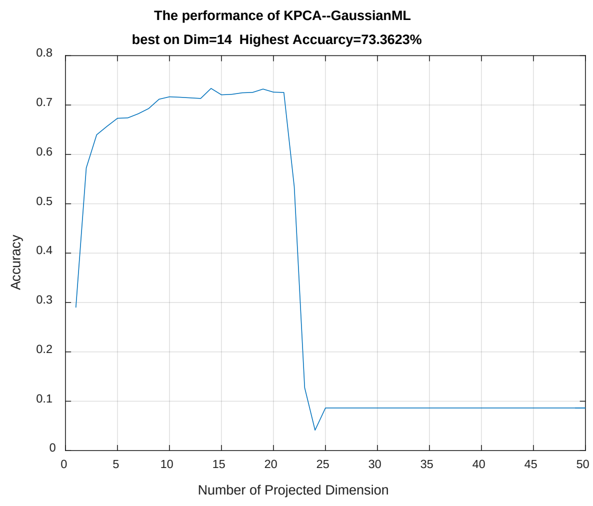


Fig 4.(2a)

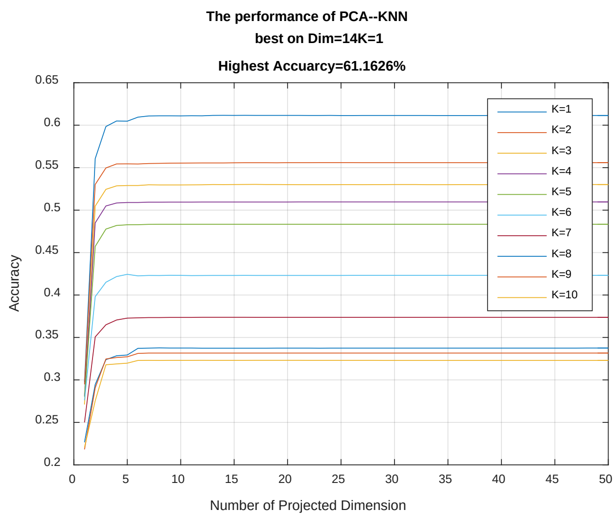


Fig 4.(1b)

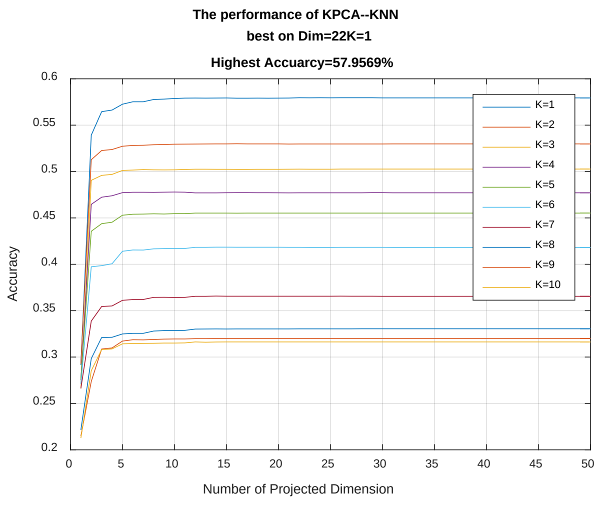


Fig 4.(2b)

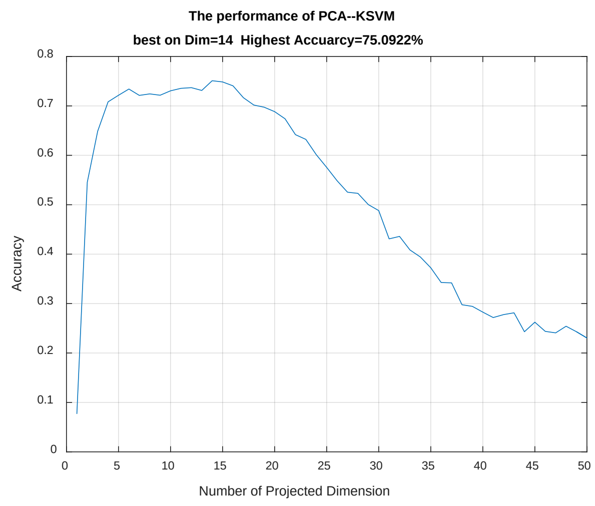


Fig 4.(1c)

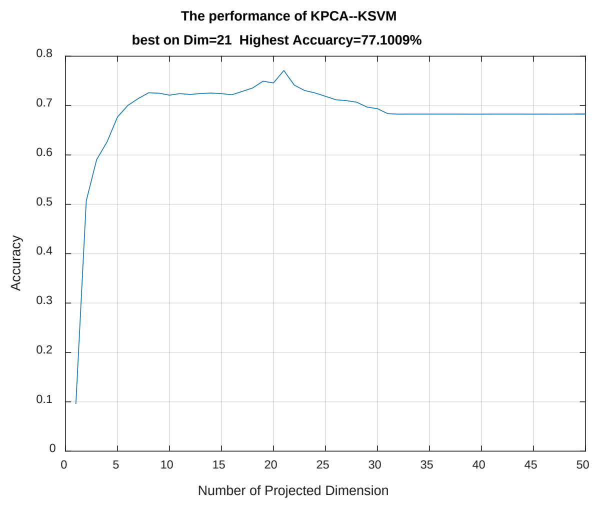


Fig 4.(2c)

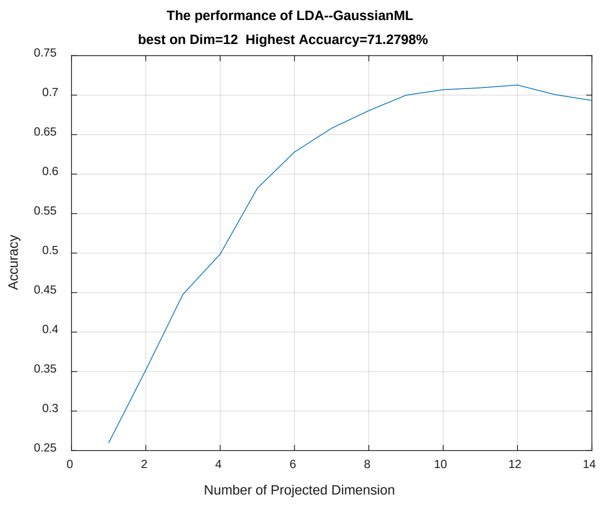


Fig 4.(3a)

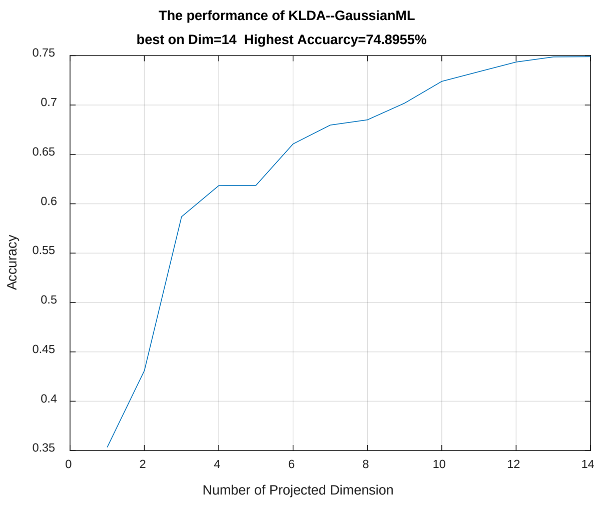


Fig 4.(4a)

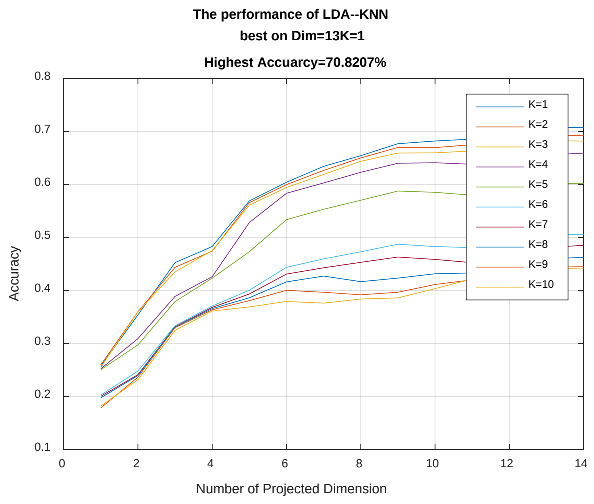


Fig 4.(3b)

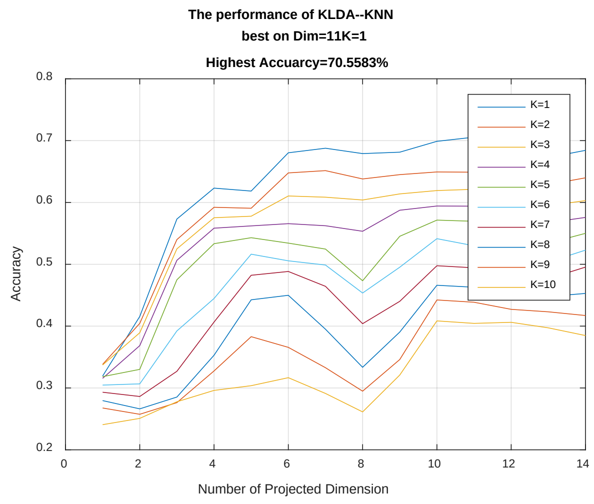


Fig 4.(4b)

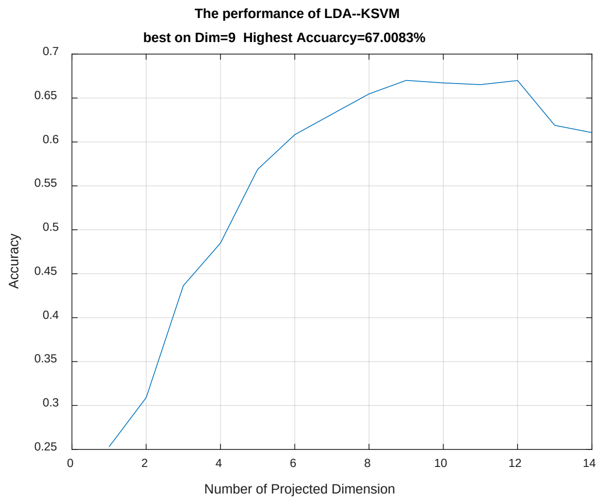


Fig 4.(3c)

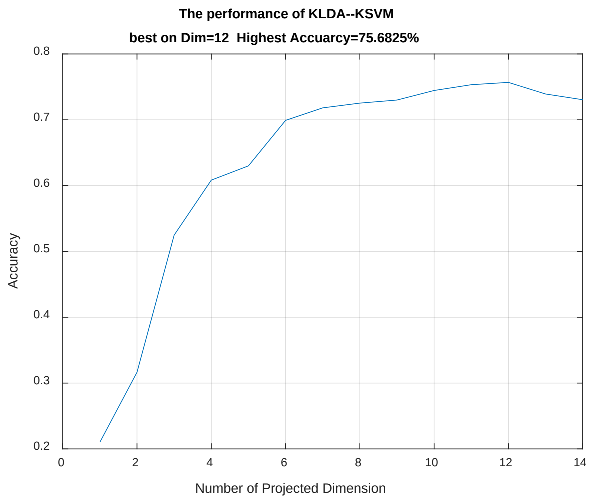


Fig 4.(4c)

Fig. 4. The adhoc analysis for all the feature reduction and classification approach combinations over all possibile projected dimensions .

From Figure 4 we can see that, all the feature selection approaches shows rising accuracy in low dimension region (from dimension 1 to 14), this is because the more dimension has been projected, the more information of original data has been conserved.

In higher dimensional space, on the one hand, the accuracy of Gaussian ML and (GMM) classification approaches encounter largely decrease, because the limited number of train samples is not sufficient enough to estimate the parameters and the singularity of covariance matrix is more severe in higher dimension, as Figure 4 (1a),(2a) showing . On the other hand, from Figure 4 (1c),(2c),(3c).(4c) SVM perform little difference on higher dimensional space. The reason is that SVM only concerns about the support vectors near decision boundaries, so dimensionality of vectors has minor restriction on implementation.

Another useful message from Figure 4 is that all the approach combinations perform well on dimension 14 compared to other dimension of the approach itself. So we will fix the dimensionality on 14 for following discussion.

B. *The effectiveness of feature reduction and classification approach combinations*

TABLE I. THE PERORMANCE OF APPROACH COMBINATIONS

	PCA	***KPCA***	***LDA***	***KLDA***
Gaussian ML	74.35%	73.36%	69.33%	74.90%
GMM	74.35%	73.36%	69.33%	74.90%
KNN	61.16%	57.92%	70.75%	68.43%
SVM*	75.09%	72.54%	61.06%	73.05%

*The accuracy of SVM without feature reduction is 73%

Table I shows the performance for approach combinations. The parameters of GMM and KNN are both set to 2, which we will further discussion in following part.

As Table I indicates, recognition combinations performs lower than 80% corrected classification, because the number of training samples in our project (750 samples) is not sufficient enough to learn the data model. Especially, the PCA /KPCA and KNN combination performs extremely disappointing. PCA/KPCA is designed to extract the dimensions that has highest "energy" (high covariance value) meanwhile possibly lose other use information like neighbor pertinence that is essential measurement to determine the nearest neighbor in Euclidean space in KNN. Generally speaking, the kernel feature reduction approaches have better performance over all classification approaches.

A detail performance analysis for each method combination to the elements in Table 1 could be seen from Appendix: Confusion matrix and (ROC) for feature reduction and classification approach combinations

C. *The discussion on projecting the training set*

In feature projection approaches, there are two ways to obtain the projected data: regarding training and testing dataset individually and jointly.

The former one is projecting the training data to learn the higher-to-lower dimensional mapping matrix, and then projecting the testing dataset by this mapping matrix. In this angle, only the configuration of training dataset has been captured and testing dataset generates feature reduced data as training data does.

The latter one treats training data and testing data equally, i.e., projecting the overall data in one time. The advantage of this way is that it encloses all the configuration/ distribution information of both training and testing dataset and there is no need to storage the projecting matrix. However, this operation only works for unsupervised feature projection approaches (e.g. PCA, KPCA) that the mapping matrixes don't use class information of training dataset.

In our project, we have tried both 2 way of projection training set for PCA and KPCA, the performance are barely no difference on accuracy.

D. *The discussion on some parameters of classication approaches*

We have examined the performance of different values of some parameters in KNN and GMM.

In KNN, K_1 is the initial size of analysis window that designated by user. Theoretically, the smaller K_1 is, the more precise distribution in local neighborhood of data could be captured.

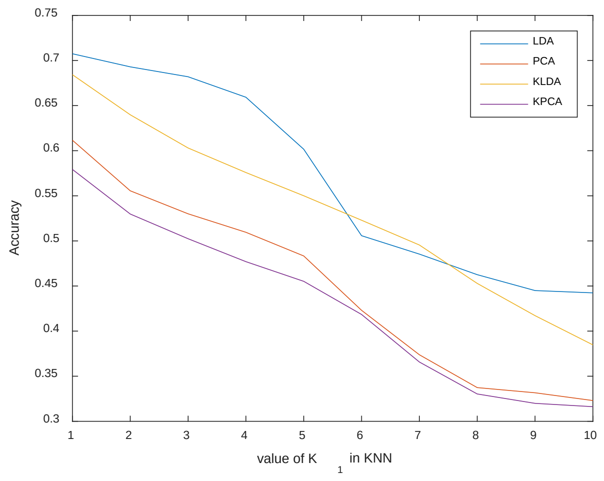


Fig. 5. The performance of KNN over different initial size of analysis window in feature reduction methods on projected dimension 14

Figure 5 even shows a clear decreasing trend for the increasing number of K_1 . Even from Figure 4. (1b), (2b), (3b), KNN classification performs best on K_1=1, corresponding to our statements.

In GMM model, K is the number of component we supposed in each class, too small of K results in inconspicuous advantage of mixture model while too big of K results in overfitting problem. K is optimal only when K is exactly the number of components (cluster) in the distribution of real data set.

TABLE II. THE PERORMANCE OF GMM OVER DIFFERENT VALUE OF K

	K=1	***K=2***	***K=3***	***K=4***	K=5
PCA	74.35%	65.74	50.00%	31.11%	4.14%
KPCA	73.36%	45.06%	36.74%	10.66%	*
LDA	69.33%	60.86%	41.00%	26.94%	8.79%
KLDA	74.90%	57.42%	44.95%	13.18%	*

*The performance of blank elements is ungiven because of the high computational amount

According to Table II, GMM classification performing best on K=1 indicates that the dataset doesn't have mulita-cluster distribution. So we will both 2 parameters to 1 for following discussion for controlling convenience.

E. The discussion on sample size of training set

To exam on how well the methods perform under varying sample size, we use (1) all available training data, (2) 10 samples per class, (3) 20samples per class, (4) 30 samples per class. In our project, the index for each class is randomly selected. Table 1 shows the overall dataset. Following tables shows the performance on selecting 10, 20, 30 samples per class respectively:

TABLE III. THE PERORMANCE OF 10 SAMPLES PER CLASS

	PCA	***KPCA***	***LDA***	***KLDA***
Gaussian ML	74.00%	72.00%	76.67%	74.67%
GMM	74.00%	72.00%	76.67%	74.67%
KNN	56.00%	55.33%	74.00%	70.67%
SVM	78.00%	72.00%	64.00%	76.67%

TABLE IV. THE PERORMANCE OF 20 SAMPLES PER CLASS

	PCA	***KPCA***	***LDA***	***KLDA***
Gaussian ML	77.00%	74.00%	72.00%	73.00%
GMM	77.00%	74.00%	72.00%	73.00%
KNN	60.00%	58.33%	73.00%	69.67%
SVM	73.33%	72.00%	61.33%	70.00%

TABLE V. THE PERORMANCE OF 30SAMPLES PER CLASS

	PCA	***KPCA***	***LDA***	***KLDA***
Gaussian ML	77.33%	73.56%	76.22%	75.78%
GMM	77.33%	73.56%	76.22%	75.78%
KNN	62.00%	59.33%	73.11%	66.67%
SVM	78.44%	75.78%	63.33%	72.67%

By comparing of the tables, all the models perform stably throughout size changing of testing dataset. To some methods including SVM and LDA, down sampling of training set even improve the classification performance. The reason is that the larger testing dataset, the more possible that data outside the configuration of learning model would appears.

V. CONCLUSION

In this project, we have implement and compared the 4 different feature reduction methods and 4 classification methods. According to the characteristic of the methods, several discussions have been proposed in experiment, turning out that:

- Feature reduction could help to improve the performance of classification, while the dimensionality should be selected carefully;
- The best dimensionality to project for all the classification method is around 14;
- Especially for SVM, even without feature reduction it performs quite well in accuracy;
- Kernel trick of feature reduction approaches would improve the performance than linear approaches;
- The optimal number of component in GMM classification is 1 indicating that the dataset doesn't have mulita-cluster distribution;
- All the models perform stably though size changing of training dataset.

ACKNOWLEDGMENT

This a term course report of the course "Digital Pattern Recognition Class". Deeply appreciate on the lecturer Professor Saurabh Prasad for the instruction and project design.

REFERENCES

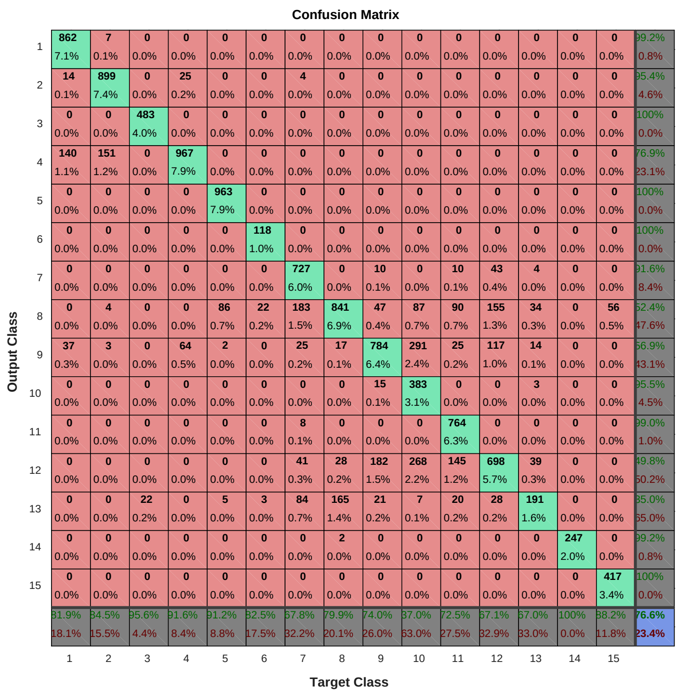
[1] Tang, Jiliang, Salem Alelyani, and Huan Liu. "Feature selection for classification: A review." Data Classification: Algorithms and Applications (2014): 37.

[2] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[3] http://www.eecs.tufts.edu/~mcao01/2010f/COMP-135.pdf

APPENDIX:

A. *Confusion matrix and (ROC) for feature reduction and classification approach combinations*



Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	**862** 7.1%	**7** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	99.2% 0.8%
2	**14** 0.1%	**899** 7.4%	**0** 0.0%	**25** 0.2%	**0** 0.0%	**0** 0.0%	**4** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	95.4% 4.6%
3	**0** 0.0%	**0** 0.0%	**483** 4.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
4	**140** 1.1%	**151** 1.2%	**0** 0.0%	**967** 7.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	76.9% 23.1%
5	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**963** 7.9%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
6	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**118** 1.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
7	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**727** 6.0%	**0** 0.0%	**10** 0.1%	**0** 0.0%	**10** 0.1%	**43** 0.4%	**4** 0.0%	**0** 0.0%	**0** 0.0%	91.6% 8.4%
8	**0** 0.0%	**4** 0.0%	**0** 0.0%	**0** 0.0%	**86** 0.7%	**22** 0.2%	**183** 1.5%	**841** 6.9%	**47** 0.4%	**87** 0.7%	**90** 0.7%	**155** 1.3%	**34** 0.3%	**0** 0.0%	**56** 0.5%	52.4% 47.6%
9	**37** 0.3%	**3** 0.0%	**0** 0.0%	**64** 0.5%	**2** 0.0%	**0** 0.0%	**25** 0.2%	**17** 0.1%	**784** 6.4%	**291** 2.4%	**25** 0.2%	**117** 1.0%	**14** 0.1%	**0** 0.0%	**0** 0.0%	56.9% 43.1%
10	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**15** 0.1%	**383** 3.1%	**0** 0.0%	**0** 0.0%	**3** 0.0%	**0** 0.0%	**0** 0.0%	95.5% 4.5%
11	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**8** 0.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**764** 6.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	99.0% 1.0%
12	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**41** 0.3%	**28** 0.2%	**182** 1.5%	**268** 2.2%	**145** 1.2%	**698** 5.7%	**39** 0.3%	**0** 0.0%	**0** 0.0%	49.8% 50.2%
13	**0** 0.0%	**0** 0.0%	**22** 0.2%	**0** 0.0%	**5** 0.0%	**3** 0.0%	**84** 0.7%	**165** 1.4%	**21** 0.2%	**7** 0.1%	**20** 0.2%	**28** 0.2%	**191** 1.6%	**0** 0.0%	**0** 0.0%	35.0% 65.0%
14	**0** 0.0%	**0** 0.0%	**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** 0.0%	**247** 2.0%	**0** 0.0%	99.2% 0.8%
15	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**417** 3.4%	100% 0.0%
	81.9% 18.1%	84.5% 15.5%	95.6% 4.4%	91.6% 8.4%	91.2% 8.8%	82.5% 17.5%	67.8% 32.2%	79.9% 20.1%	74.0% 26.0%	37.0% 63.0%	72.5% 27.5%	67.1% 32.9%	67.0% 33.0%	100% 0.0%	88.2% 11.8%	**76.6% 23.4%**

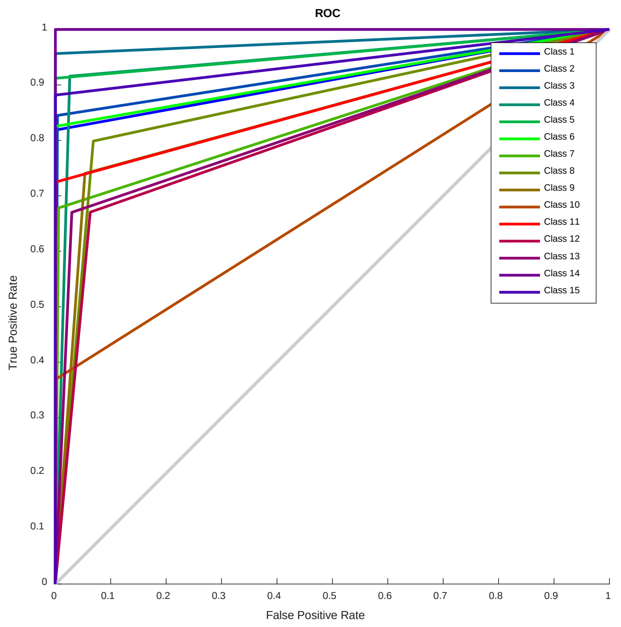
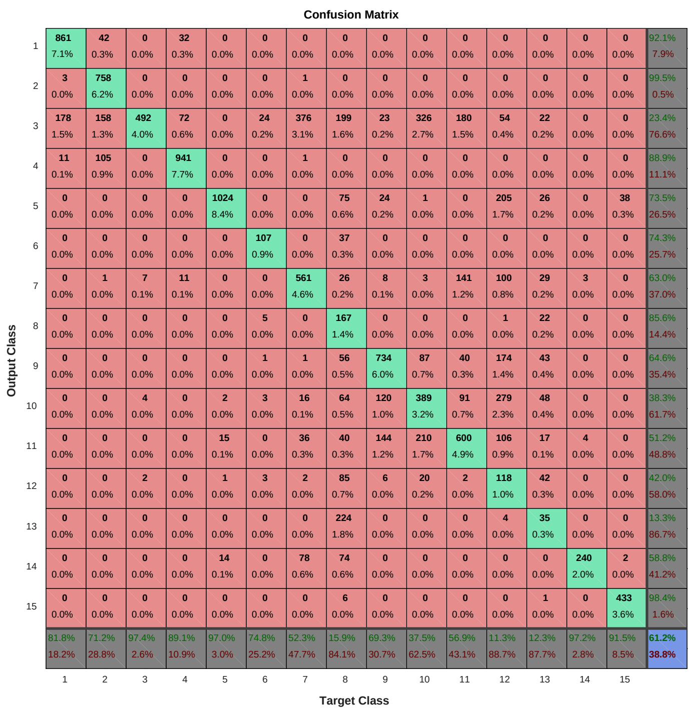


Fig . Confusion matrix and ROC of PCA-GaussianML method



Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	**861** 7.1%	**42** 0.3%	**0** 0.0%	**32** 0.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	92.1% 7.9%
2	**3** 0.0%	**758** 6.2%	**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** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	99.5% 0.5%
3	**178** 1.5%	**158** 1.3%	**492** 4.0%	**72** 0.6%	**0** 0.0%	**24** 0.2%	**376** 3.1%	**199** 1.6%	**23** 0.2%	**326** 2.7%	**180** 1.5%	**54** 0.4%	**22** 0.2%	**0** 0.0%	**0** 0.0%	23.4% 76.6%
4	**11** 0.1%	**105** 0.9%	**0** 0.0%	**941** 7.7%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	88.9% 11.1%
5	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1024** 8.4%	**0** 0.0%	**0** 0.0%	**75** 0.6%	**24** 0.2%	**1** 0.0%	**0** 0.0%	**205** 1.7%	**26** 0.2%	**0** 0.0%	**38** 0.3%	73.5% 26.5%
6	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**107** 0.9%	**0** 0.0%	**37** 0.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	74.3% 25.7%
7	**0** 0.0%	**1** 0.0%	**7** 0.1%	**11** 0.1%	**0** 0.0%	**0** 0.0%	**561** 4.6%	**26** 0.2%	**8** 0.1%	**3** 0.0%	**141** 1.2%	**100** 0.8%	**29** 0.2%	**3** 0.0%	**0** 0.0%	63.0% 37.0%
8	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**5** 0.0%	**0** 0.0%	**167** 1.4%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**22** 0.2%	**0** 0.0%	**0** 0.0%	85.6% 14.4%
9	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**1** 0.0%	**56** 0.5%	**734** 6.0%	**87** 0.7%	**40** 0.3%	**174** 1.4%	**43** 0.4%	**0** 0.0%	**0** 0.0%	64.6% 35.4%
10	**0** 0.0%	**0** 0.0%	**4** 0.0%	**0** 0.0%	**2** 0.0%	**3** 0.0%	**16** 0.1%	**64** 0.5%	**120** 1.0%	**389** 3.2%	**91** 0.7%	**279** 2.3%	**48** 0.4%	**0** 0.0%	**0** 0.0%	38.3% 61.7%
11	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**15** 0.1%	**0** 0.0%	**36** 0.3%	**40** 0.3%	**144** 1.2%	**210** 1.7%	**600** 4.9%	**106** 0.9%	**17** 0.1%	**4** 0.0%	**0** 0.0%	51.2% 48.8%
12	**0** 0.0%	**0** 0.0%	**2** 0.0%	**0** 0.0%	**1** 0.0%	**3** 0.0%	**2** 0.0%	**85** 0.7%	**6** 0.0%	**20** 0.2%	**2** 0.0%	**118** 1.0%	**42** 0.3%	**0** 0.0%	**0** 0.0%	42.0% 58.0%
13	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**224** 1.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**4** 0.0%	**35** 0.3%	**0** 0.0%	**0** 0.0%	13.3% 86.7%
14	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**14** 0.1%	**0** 0.0%	**78** 0.6%	**74** 0.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**240** 2.0%	**2** 0.0%	58.8% 41.2%
15	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**6** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.0%	**0** 0.0%	**433** 3.6%	98.4% 1.6%
	81.8% 18.2%	71.2% 28.8%	97.4% 2.6%	89.1% 10.9%	97.0% 3.0%	74.8% 25.2%	52.3% 47.7%	15.9% 84.1%	69.3% 30.7%	37.5% 62.5%	56.9% 43.1%	11.3% 88.7%	12.3% 87.7%	97.2% 2.8%	91.5% 8.5%	**61.2% 38.8%**

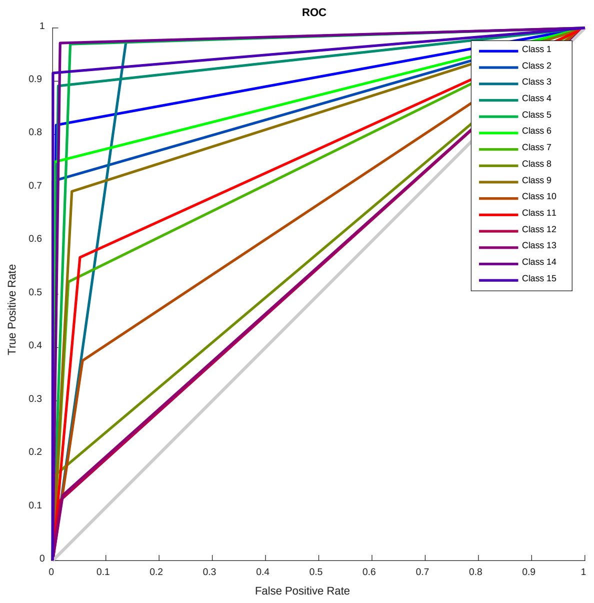
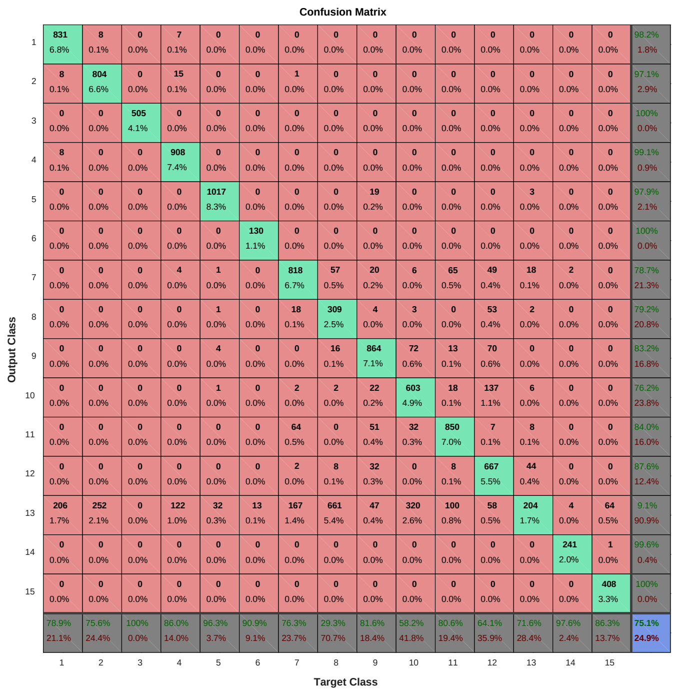


Fig. Confusion matrix and ROC of PCA - KNN method



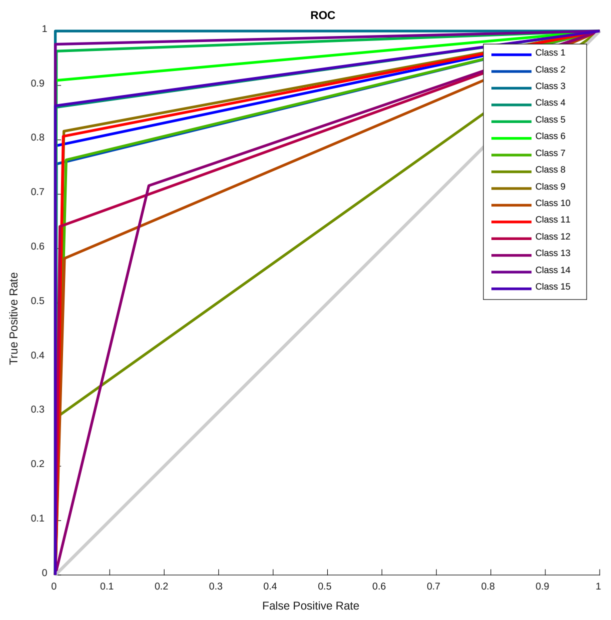
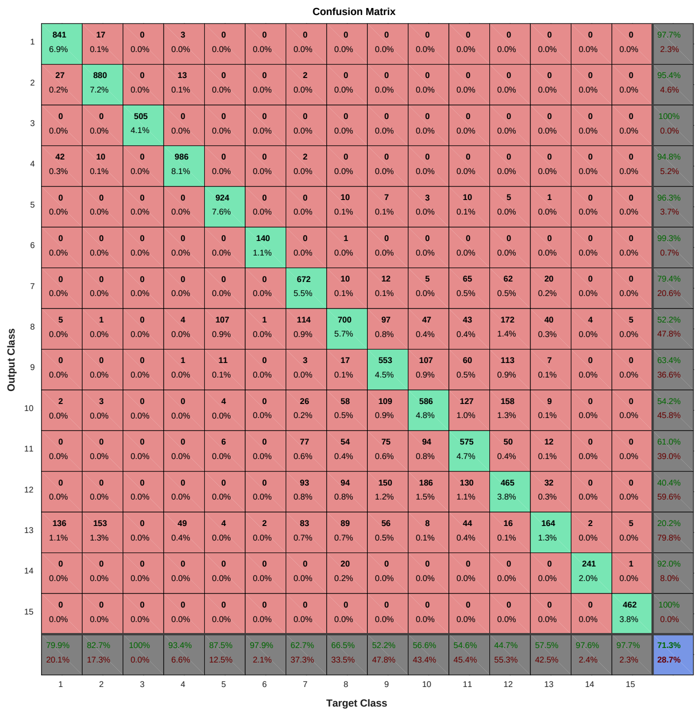


Fig. Confusion matrix and ROC of PCA - SVM method



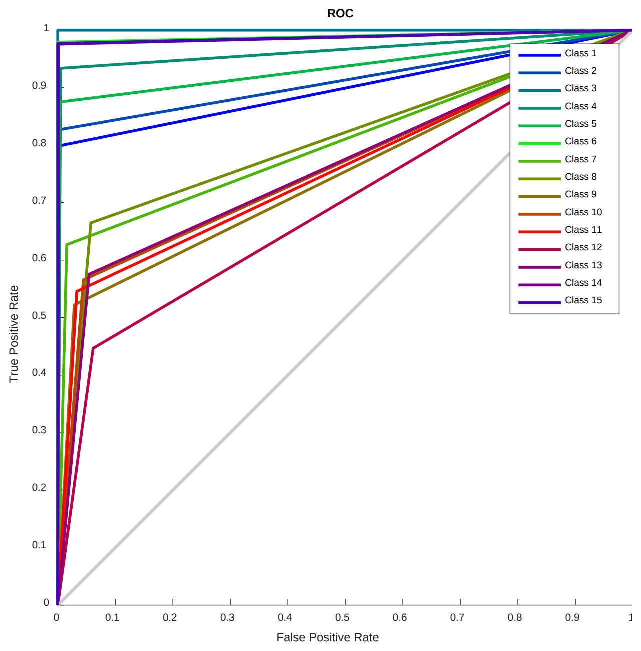
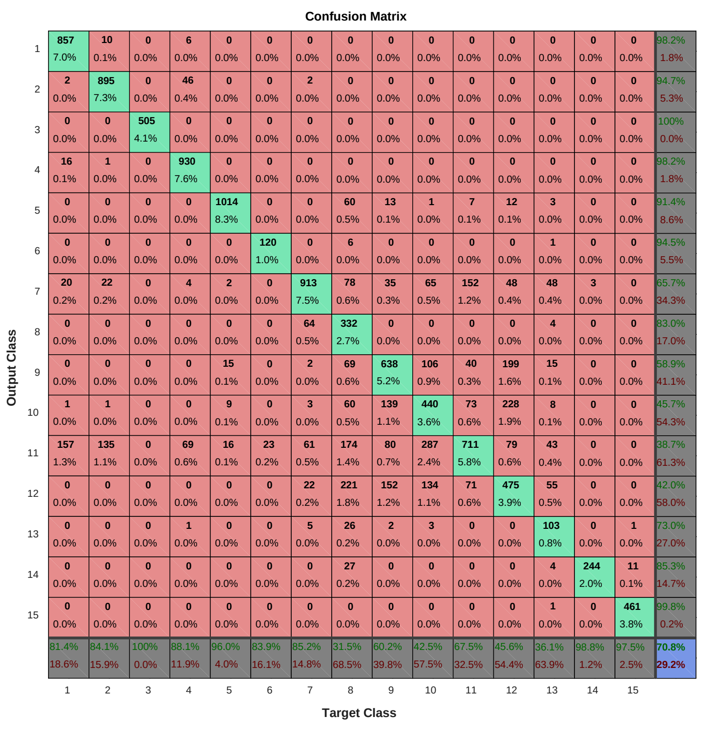


Fig . Confusion matrix and ROC of LDA-GaussianML method


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	857 7.0%	10 0.1%	0 0.0%	6 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.2% 1.8%
2	2 0.0%	895 7.3%	0 0.0%	46 0.4%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.7% 5.3%
3	0 0.0%	0 0.0%	505 4.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	16 0.1%	1 0.0%	0 0.0%	930 7.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.2% 1.8%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1014 8.3%	0 0.0%	0 0.0%	60 0.5%	13 0.1%	1 0.0%	7 0.1%	12 0.1%	3 0.0%	0 0.0%	0 0.0%	91.4% 8.6%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	120 1.0%	0 0.0%	6 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	94.5% 5.5%
7	20 0.2%	22 0.2%	0 0.0%	4 0.0%	2 0.0%	0 0.0%	913 7.5%	78 0.6%	35 0.3%	65 0.5%	152 1.2%	48 0.4%	48 0.4%	3 0.0%	0 0.0%	65.7% 34.3%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	64 0.5%	332 2.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.0%	0 0.0%	0 0.0%	83.0% 17.0%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.1%	0 0.0%	2 0.0%	69 0.6%	638 5.2%	106 0.9%	40 0.3%	199 1.6%	15 0.1%	0 0.0%	0 0.0%	58.9% 41.1%
10	1 0.0%	1 0.0%	0 0.0%	0 0.0%	9 0.1%	0 0.0%	3 0.0%	60 0.5%	139 1.1%	440 3.6%	73 0.6%	228 1.9%	8 0.1%	0 0.0%	0 0.0%	45.7% 54.3%
11	157 1.3%	135 1.1%	0 0.0%	69 0.6%	16 0.1%	23 0.2%	61 0.5%	174 1.4%	80 0.7%	287 2.4%	711 5.8%	79 0.6%	43 0.4%	0 0.0%	0 0.0%	38.7% 61.3%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	22 0.2%	221 1.8%	152 1.2%	134 1.1%	71 0.6%	475 3.9%	55 0.5%	0 0.0%	0 0.0%	42.0% 58.0%
13	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	5 0.0%	26 0.2%	2 0.0%	3 0.0%	0 0.0%	0 0.0%	103 0.8%	0 0.0%	1 0.0%	73.0% 27.0%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	27 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.0%	244 2.0%	11 0.1%	85.3% 14.7%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	461 3.8%	99.8% 0.2%
	81.4% 18.6%	84.1% 15.9%	100% 0.0%	88.1% 11.9%	96.0% 4.0%	83.9% 16.1%	85.2% 14.8%	31.5% 68.5%	60.2% 39.8%	42.5% 57.5%	67.5% 32.5%	45.6% 54.4%	36.1% 63.9%	98.8% 1.2%	97.5% 2.5%	70.8% 29.2%

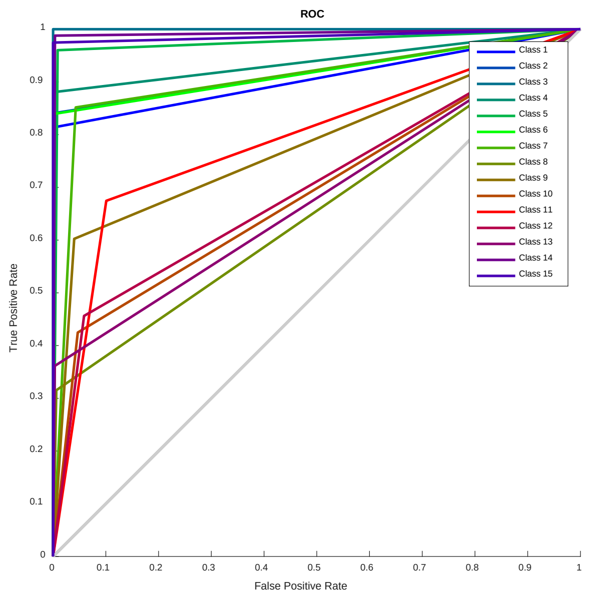
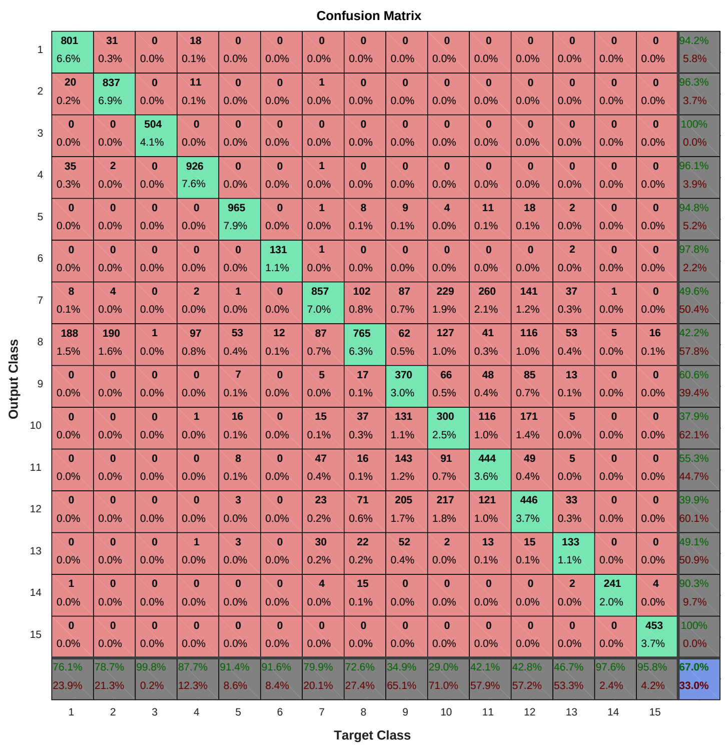


Fig . Confusion matrix and ROC of LDA-KNN method


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	801 6.6%	31 0.3%	0 0.0%	18 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.2% 5.8%
2	20 0.2%	837 6.9%	0 0.0%	11 0.1%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96.3% 3.7%
3	0 0.0%	0 0.0%	504 4.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	35 0.3%	2 0.0%	0 0.0%	926 7.6%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96.1% 3.9%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	965 7.9%	0 0.0%	1 0.0%	8 0.1%	9 0.1%	4 0.0%	11 0.1%	18 0.1%	2 0.0%	0 0.0%	0 0.0%	94.8% 5.2%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	131 1.1%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	97.8% 2.2%
7	8 0.1%	4 0.0%	0 0.0%	2 0.0%	1 0.0%	0 0.0%	857 7.0%	102 0.8%	87 0.7%	229 1.9%	260 2.1%	141 1.2%	37 0.3%	1 0.0%	0 0.0%	49.6% 50.4%
8	188 1.5%	190 1.6%	1 0.0%	97 0.8%	53 0.4%	12 0.1%	87 0.7%	765 6.3%	62 0.5%	127 1.0%	41 0.3%	116 1.0%	53 0.4%	5 0.0%	16 0.1%	42.2% 57.8%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	7 0.1%	0 0.0%	5 0.0%	17 0.1%	370 3.0%	66 0.5%	48 0.4%	85 0.7%	13 0.1%	0 0.0%	0 0.0%	60.6% 39.4%
10	0 0.0%	0 0.0%	0 0.0%	1 0.0%	16 0.1%	0 0.0%	15 0.1%	37 0.3%	131 1.1%	300 2.5%	116 1.0%	171 1.4%	5 0.0%	0 0.0%	0 0.0%	37.9% 62.1%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.1%	0 0.0%	47 0.4%	16 0.1%	143 1.2%	91 0.7%	444 3.6%	49 0.4%	5 0.0%	0 0.0%	0 0.0%	55.3% 44.7%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.0%	0 0.0%	23 0.2%	71 0.6%	205 1.7%	217 1.8%	121 1.0%	446 3.7%	33 0.3%	0 0.0%	0 0.0%	39.9% 60.1%
13	0 0.0%	0 0.0%	0 0.0%	1 0.0%	3 0.0%	0 0.0%	30 0.2%	22 0.2%	52 0.4%	2 0.0%	13 0.1%	15 0.1%	133 1.1%	0 0.0%	0 0.0%	49.1% 50.9%
14	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.0%	15 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	241 2.0%	4 0.0%	90.3% 9.7%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	453 3.7%	100% 0.0%
	76.1% 23.9%	78.7% 21.3%	99.8% 0.2%	87.7% 12.3%	91.4% 8.6%	91.6% 8.4%	79.9% 20.1%	72.6% 27.4%	34.9% 65.1%	29.0% 71.0%	42.1% 57.9%	42.8% 57.2%	46.7% 53.3%	97.6% 2.4%	95.8% 4.2%	67.0% 33.0%

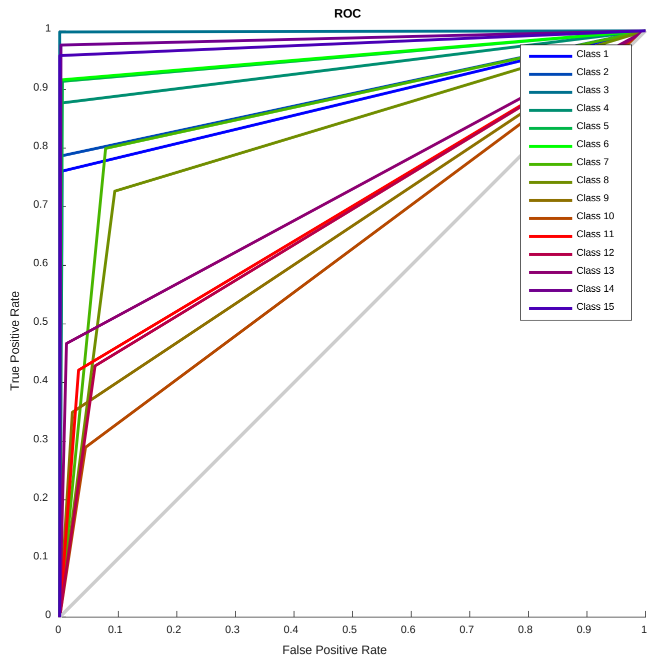
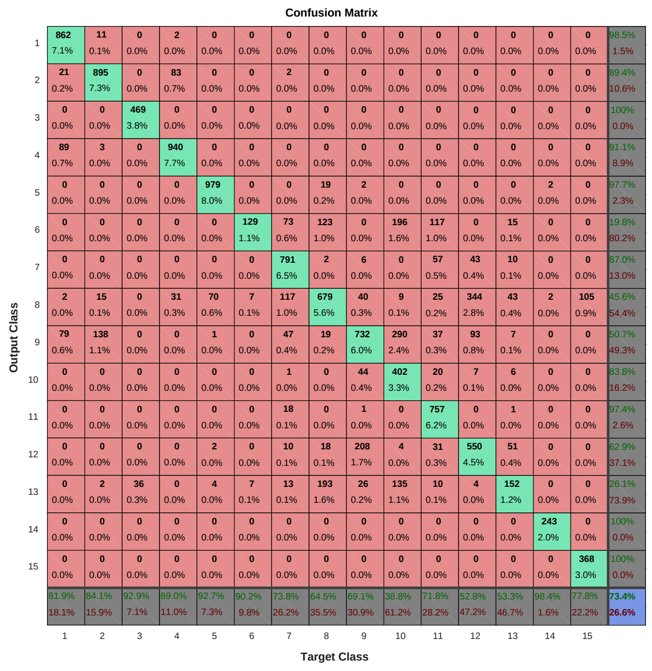


Fig . Confusion matrix and ROC of LDA-SVM method


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	862 7.1%	11 0.1%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.5% 1.5%
2	21 0.2%	895 7.3%	0 0.0%	83 0.7%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	89.4% 10.6%
3	0 0.0%	0 0.0%	469 3.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	89 0.7%	3 0.0%	0 0.0%	940 7.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	91.1% 8.9%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	979 8.0%	0 0.0%	0 0.0%	19 0.2%	2 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	97.7% 2.3%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	129 1.1%	73 0.6%	123 1.0%	0 0.0%	196 1.6%	117 1.0%	0 0.0%	15 0.1%	0 0.0%	0 0.0%	19.8% 80.2%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	791 6.5%	2 0.0%	6 0.0%	0 0.0%	57 0.5%	43 0.4%	10 0.1%	0 0.0%	0 0.0%	87.0% 13.0%
8	2 0.0%	15 0.1%	0 0.0%	31 0.3%	70 0.6%	7 0.1%	117 1.0%	679 5.6%	40 0.3%	9 0.1%	25 0.2%	344 2.8%	43 0.4%	2 0.0%	105 0.9%	45.6% 54.4%
9	79 0.6%	138 1.1%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	47 0.4%	19 0.2%	732 6.0%	290 2.4%	37 0.3%	93 0.8%	7 0.1%	0 0.0%	0 0.0%	50.7% 49.3%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	44 0.4%	402 3.3%	20 0.2%	7 0.1%	6 0.0%	0 0.0%	0 0.0%	83.8% 16.2%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	18 0.1%	0 0.0%	1 0.0%	0 0.0%	757 6.2%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	97.4% 2.6%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	10 0.1%	18 0.1%	208 1.7%	4 0.0%	31 0.3%	550 4.5%	51 0.4%	0 0.0%	0 0.0%	62.9% 37.1%
13	0 0.0%	2 0.0%	36 0.3%	0 0.0%	4 0.0%	7 0.1%	13 0.1%	193 1.6%	26 0.2%	135 1.1%	10 0.1%	4 0.0%	152 1.2%	0 0.0%	0 0.0%	26.1% 73.9%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	243 2.0%	0 0.0%	100% 0.0%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	368 3.0%	100% 0.0%
	81.9% 18.1%	84.1% 15.9%	92.9% 7.1%	89.0% 11.0%	92.7% 7.3%	90.2% 9.8%	73.8% 26.2%	64.5% 35.5%	69.1% 30.9%	38.8% 61.2%	71.8% 28.2%	52.8% 47.2%	53.3% 46.7%	98.4% 1.6%	77.8% 22.2%	73.4% 26.6%

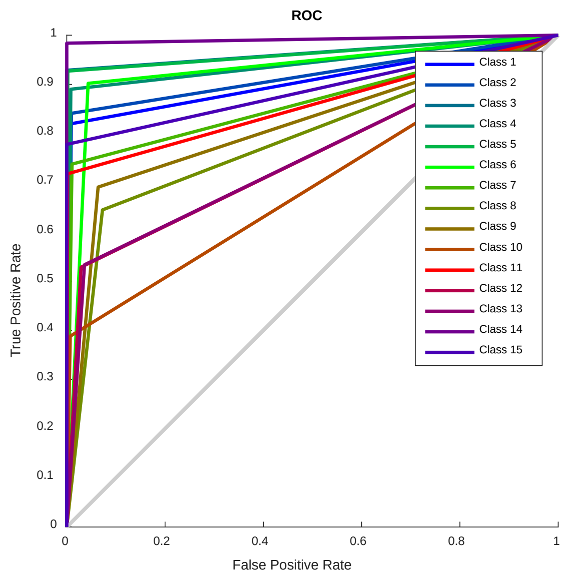
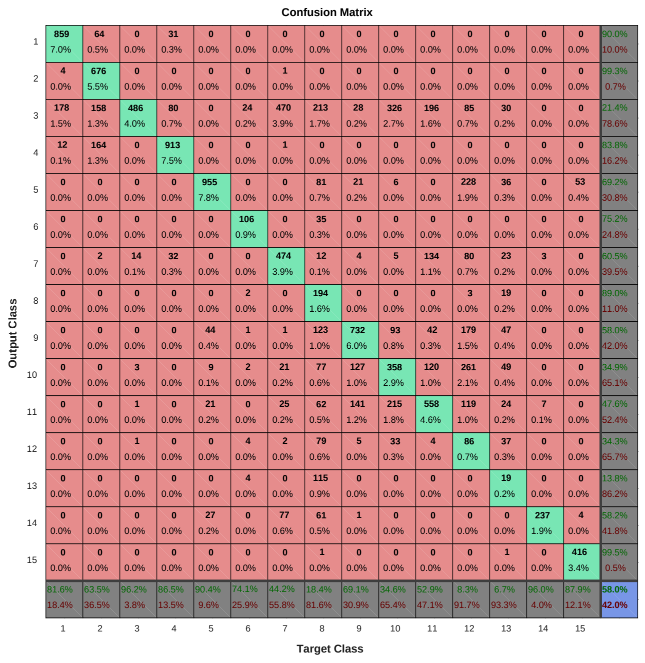


Fig . Confusion matrix and ROC of KPCA-GaussianML method


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	859 7.0%	64 0.5%	0 0.0%	31 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	90.0% 10.0%
2	4 0.0%	676 5.5%	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 0.0%	0 0.0%	0 0.0%	0 0.0%	99.3% 0.7%
3	178 1.5%	158 1.3%	486 4.0%	80 0.7%	0 0.0%	24 0.2%	470 3.9%	213 1.7%	28 0.2%	326 2.7%	196 1.6%	85 0.7%	30 0.2%	0 0.0%	0 0.0%	21.4% 78.6%
4	12 0.1%	164 1.3%	0 0.0%	913 7.5%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	83.8% 16.2%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	955 7.8%	0 0.0%	0 0.0%	81 0.7%	21 0.2%	6 0.0%	0 0.0%	228 1.9%	36 0.3%	0 0.0%	53 0.4%	69.2% 30.8%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	106 0.9%	0 0.0%	35 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.2% 24.8%
7	0 0.0%	2 0.0%	14 0.1%	32 0.3%	0 0.0%	0 0.0%	474 3.9%	12 0.1%	4 0.0%	5 0.0%	134 1.1%	80 0.7%	23 0.2%	3 0.0%	0 0.0%	60.5% 39.5%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	194 1.6%	0 0.0%	0 0.0%	0 0.0%	3 0.0%	19 0.2%	0 0.0%	0 0.0%	89.0% 11.0%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	44 0.4%	1 0.0%	1 0.0%	123 1.0%	732 6.0%	93 0.8%	42 0.3%	179 1.5%	47 0.4%	0 0.0%	0 0.0%	58.0% 42.0%
10	0 0.0%	0 0.0%	3 0.0%	0 0.0%	9 0.1%	2 0.0%	21 0.2%	77 0.6%	127 1.0%	358 2.9%	120 1.0%	261 2.1%	49 0.4%	0 0.0%	0 0.0%	34.9% 65.1%
11	0 0.0%	0 0.0%	1 0.0%	0 0.0%	21 0.2%	0 0.0%	25 0.2%	62 0.5%	141 1.2%	215 1.8%	558 4.6%	119 1.0%	24 0.2%	7 0.1%	0 0.0%	47.6% 52.4%
12	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	4 0.0%	2 0.0%	79 0.6%	5 0.0%	33 0.3%	4 0.0%	86 0.7%	37 0.3%	0 0.0%	0 0.0%	34.3% 65.7%
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.0%	0 0.0%	115 0.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	19 0.2%	0 0.0%	0 0.0%	13.8% 86.2%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	27 0.2%	0 0.0%	77 0.6%	61 0.5%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	237 1.9%	4 0.0%	58.2% 41.8%
15	0 0.0%	0 0.0%	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%	1 0.0%	0 0.0%	416 3.4%	99.5% 0.5%
	81.6% 18.4%	63.5% 36.5%	96.2% 3.8%	86.5% 13.5%	90.4% 9.6%	74.1% 25.9%	44.2% 55.8%	18.4% 81.6%	69.1% 30.9%	34.6% 65.4%	52.9% 47.1%	8.3% 91.7%	6.7% 93.3%	96.0% 4.0%	87.9% 12.1%	58.0% 42.0%

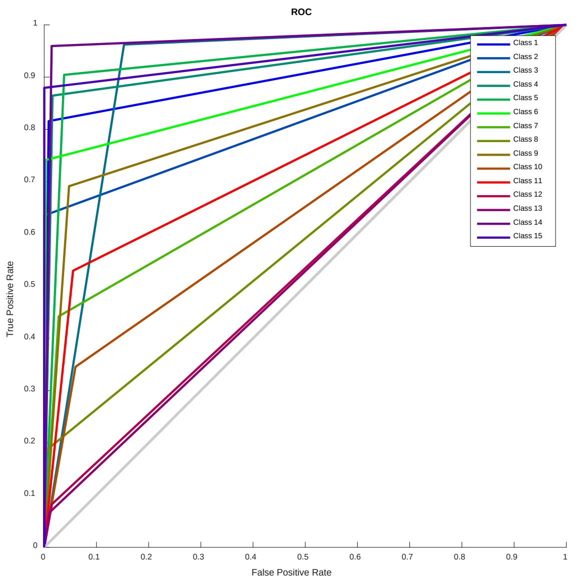
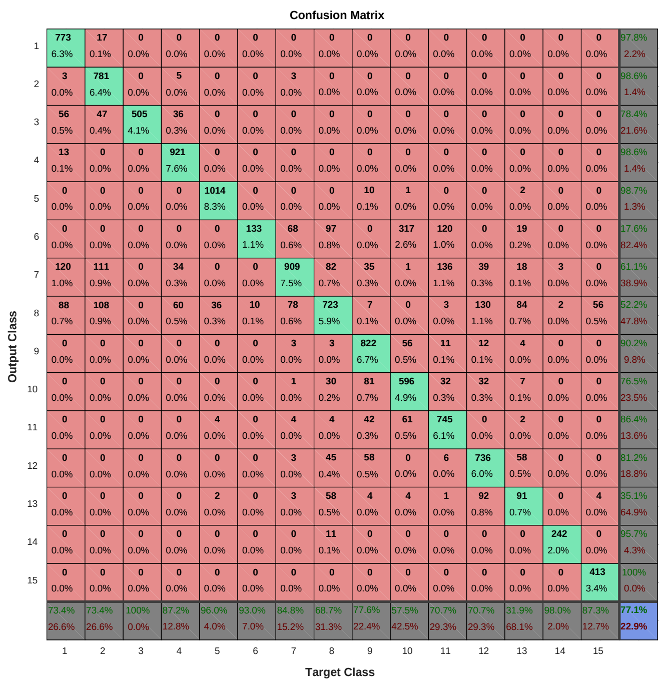


Fig . Confusion matrix and ROC of KPCA-KNN method


Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	773 6.3%	17 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	97.8% 2.2%
2	3 0.0%	781 6.4%	0 0.0%	5 0.0%	0 0.0%	0 0.0%	3 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.6% 1.4%
3	56 0.5%	47 0.4%	505 4.1%	36 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	78.4% 21.6%
4	13 0.1%	0 0.0%	0 0.0%	921 7.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.6% 1.4%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1014 8.3%	0 0.0%	0 0.0%	0 0.0%	10 0.1%	1 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	98.7% 1.3%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	133 1.1%	68 0.6%	97 0.8%	0 0.0%	317 2.6%	120 1.0%	0 0.0%	19 0.2%	0 0.0%	0 0.0%	17.6% 82.4%
7	120 1.0%	111 0.9%	0 0.0%	34 0.3%	0 0.0%	0 0.0%	909 7.5%	82 0.7%	35 0.3%	1 0.0%	136 1.1%	39 0.3%	18 0.1%	3 0.0%	0 0.0%	61.1% 38.9%
8	88 0.7%	108 0.9%	0 0.0%	60 0.5%	36 0.3%	10 0.1%	78 0.6%	723 5.9%	7 0.1%	0 0.0%	3 0.0%	130 1.1%	84 0.7%	2 0.0%	56 0.5%	52.2% 47.8%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.0%	3 0.0%	822 6.7%	56 0.5%	11 0.1%	12 0.1%	4 0.0%	0 0.0%	0 0.0%	90.2% 9.8%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	30 0.2%	81 0.7%	596 4.9%	32 0.3%	32 0.3%	7 0.1%	0 0.0%	0 0.0%	76.5% 23.5%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.0%	0 0.0%	4 0.0%	4 0.0%	42 0.3%	61 0.5%	745 6.1%	0 0.0%	2 0.0%	0 0.0%	0 0.0%	86.4% 13.6%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.0%	45 0.4%	58 0.5%	0 0.0%	6 0.0%	736 6.0%	58 0.5%	0 0.0%	0 0.0%	81.2% 18.8%
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	3 0.0%	58 0.5%	4 0.0%	4 0.0%	1 0.0%	92 0.8%	91 0.7%	0 0.0%	4 0.0%	35.1% 64.9%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	11 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	242 2.0%	0 0.0%	95.7% 4.3%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	413 3.4%	100% 0.0%
	73.4% 26.6%	73.4% 26.6%	100% 0.0%	87.2% 12.8%	96.0% 4.0%	93.0% 7.0%	84.8% 15.2%	68.7% 31.3%	77.6% 22.4%	57.5% 42.5%	70.7% 29.3%	70.7% 29.3%	31.9% 68.1%	98.0% 2.0%	87.3% 12.7%	77.1% 22.9%

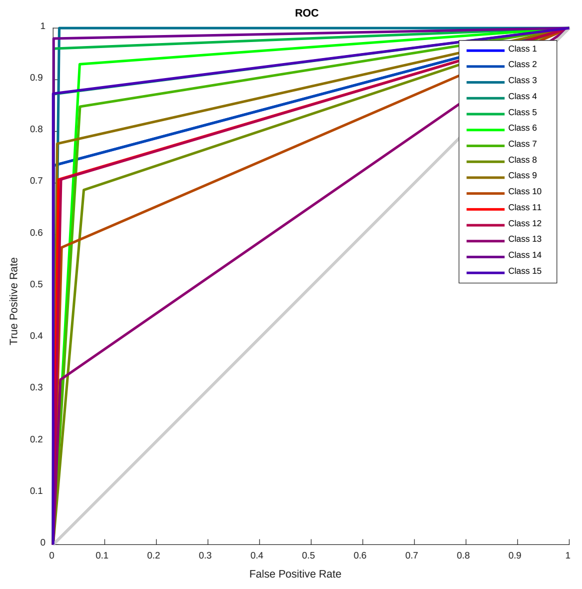
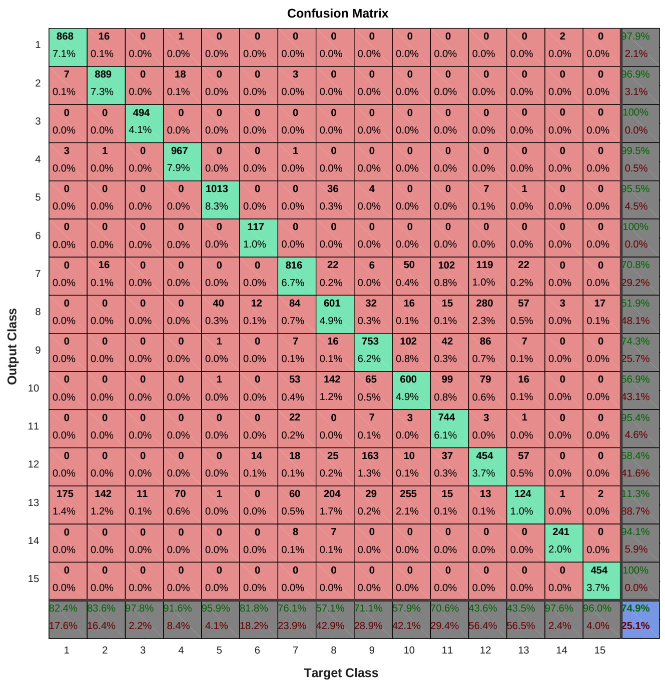


Fig . Confusion matrix and ROC of KPCA-SVM method


Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	868 7.1%	16 0.1%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	0 0.0%	97.9% 2.1%
2	7 0.1%	889 7.3%	0 0.0%	18 0.1%	0 0.0%	0 0.0%	3 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96.9% 3.1%
3	0 0.0%	0 0.0%	494 4.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	3 0.0%	1 0.0%	0 0.0%	967 7.9%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1013 8.3%	0 0.0%	0 0.0%	36 0.3%	4 0.0%	0 0.0%	0 0.0%	7 0.1%	1 0.0%	0 0.0%	0 0.0%	95.5% 4.5%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	117 1.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	16 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	816 6.7%	22 0.2%	6 0.0%	50 0.4%	102 0.8%	119 1.0%	22 0.2%	0 0.0%	0 0.0%	70.8% 29.2%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	40 0.3%	12 0.1%	84 0.7%	601 4.9%	32 0.3%	16 0.1%	15 0.1%	280 2.3%	57 0.5%	3 0.0%	17 0.1%	51.9% 48.1%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	7 0.1%	16 0.1%	753 6.2%	102 0.8%	42 0.3%	86 0.7%	7 0.1%	0 0.0%	0 0.0%	74.3% 25.7%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	53 0.4%	142 1.2%	65 0.5%	600 4.9%	99 0.8%	79 0.6%	16 0.1%	0 0.0%	0 0.0%	56.9% 43.1%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	22 0.2%	0 0.0%	7 0.1%	3 0.0%	744 6.1%	3 0.0%	1 0.0%	0 0.0%	0 0.0%	95.4% 4.6%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	14 0.1%	18 0.1%	25 0.2%	163 1.3%	10 0.1%	37 0.3%	454 3.7%	57 0.5%	0 0.0%	0 0.0%	58.4% 41.6%
13	175 1.4%	142 1.2%	11 0.1%	70 0.6%	1 0.0%	0 0.0%	60 0.5%	204 1.7%	29 0.2%	255 2.1%	15 0.1%	13 0.1%	124 1.0%	1 0.0%	2 0.0%	11.3% 88.7%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.1%	7 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	241 2.0%	0 0.0%	94.1% 5.9%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	454 3.7%	100% 0.0%
	82.4% 17.6%	83.6% 16.4%	97.8% 2.2%	91.6% 8.4%	95.9% 4.1%	81.8% 18.2%	76.1% 23.9%	57.1% 42.9%	71.1% 28.9%	57.9% 42.1%	70.6% 29.4%	43.6% 56.4%	43.5% 56.5%	97.6% 2.4%	96.0% 4.0%	74.9% 25.1%

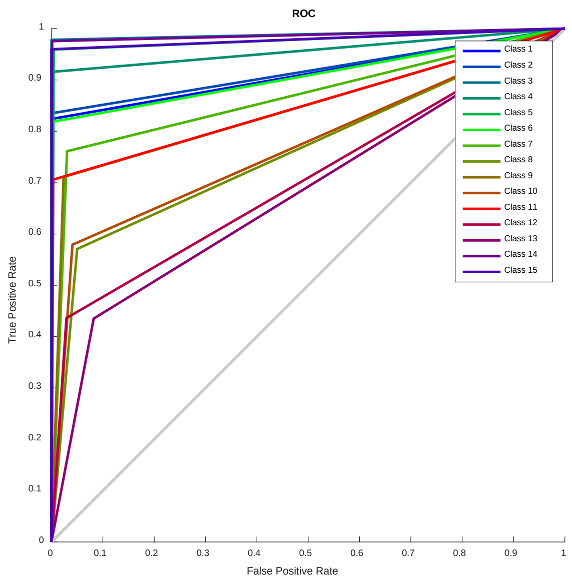
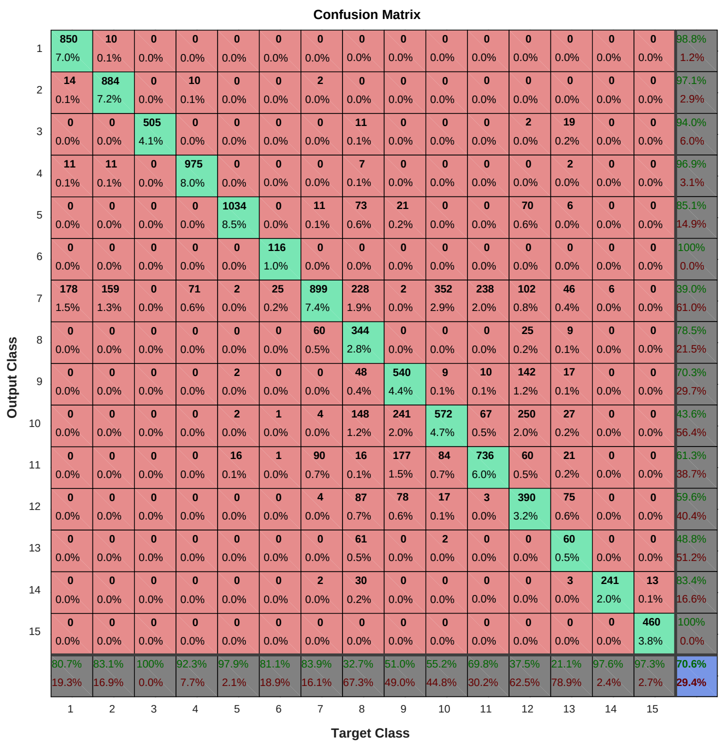


Fig . Confusion matrix and ROC of KLDA-Gassian ML method



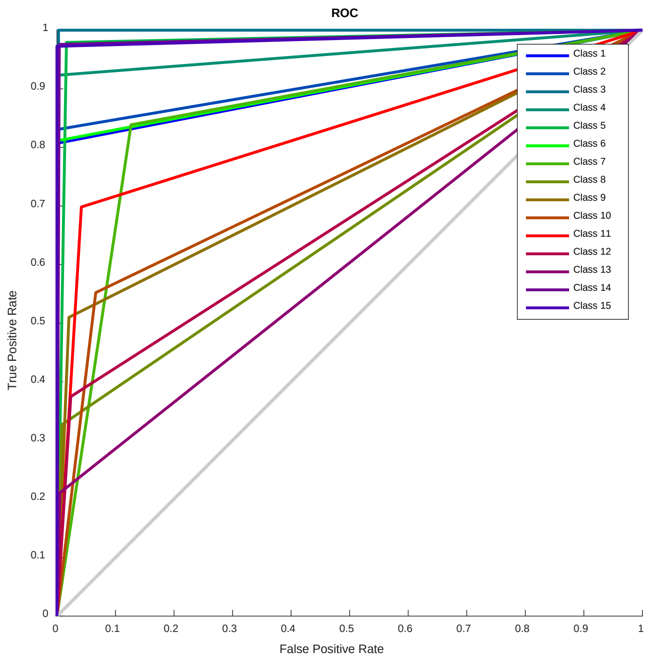
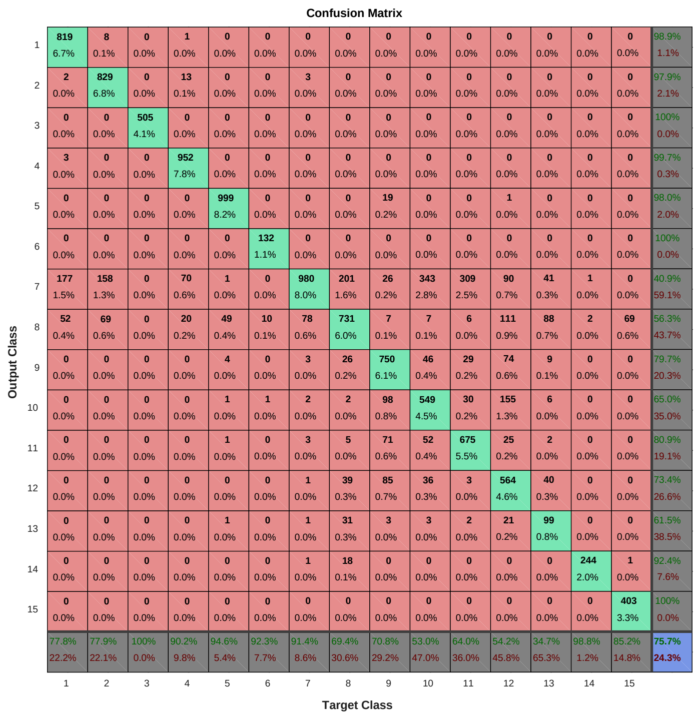


Fig . Confusion matrix and ROC of KLDA-KNN method



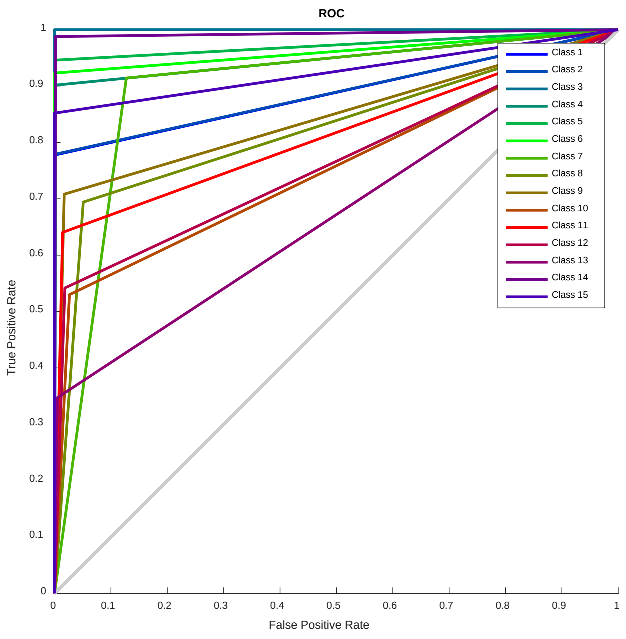


Fig . Confusion matrix and ROC of KLDA-SVM method