Project DR06

Elhamsadat Hejazi

I. Literature Review

Real-world data for many machine-learning or data-mining projects are usually high-dimensional or consist of a large number of features which is hard to be managed [1]. To avoid overfitting, wasting time and storage for saving data, and to improve the performance of the machine-learning model, there are some algorithms that scientists use to achieve low-dimensional data out of the original dataset provided that it retains most of the important attributes, called intrinsic, of the main dataset [2]. This transformation is called Dimensionality Reduction (DR).

The DR algorithms listed in the project DR06 are Probabilistic Principal Component Analysis (PPCA), Landmark ISOMAP [LIM], and Local Linear Embedding [LLE] that will be discussed in the following.

A. Probabilistic PCA [PPCA]

The major difference between the PPCA technique and the well-known PCA technique is that it deals with the missing data and in addition to it, the probabilistic approach enables the model to analyze a wide range of possible extensions in the future [3].

Algorithm 1 PPCA

Assume a centered dataset with $\boldsymbol{d}$ dimensions as $Y = [y_1, \dots, y_n]^T$ where $\boldsymbol{n}$ is the number of observations. Considering Gaussian noise, the relationship between this dataset and the embedded $\boldsymbol{q}$-dimensional latent-space data (i.e., X_n) will be as follows:

$$y_n = W_{X_n} + n \quad (1)$$

Where $\boldsymbol{y_n}$ is the set of a specific observation, $\boldsymbol{W}$ is a $\boldsymbol{d}\times\boldsymbol{q}$ matrix ($\boldsymbol{q}<\boldsymbol{d}$), and the noise is in Gaussian distribution with the mean of 0 and COV of $\boldsymbol{\beta^{-1}I}$.

Therefore, the likelihood will be calculated as follows:

$$P(y_n|X_n, W, \beta) = N\left(y_n|W_{X_n}, \beta^{-1}I\right) \quad (2)$$

Where $\boldsymbol{N}$ defines as a normal distribution function of multivariate of y_n with the mean of W_{X_n} and COV of $\beta^{-1}I$.

The next step would be defining the marginal likelihood which is as follows:

$$p(y_n|W, b) = \int p(y_n|X_n, W, b)p(X_n)dX_n \quad (3)$$

That necessitates identifying a prior distribution over X_n that for PPCA this prior distribution (i.e., $p(X_n)$) has a mean of 0, Gaussian distribution, and a unit COV which turns the equation (3) into the following:

$$p(y_n|W, \beta) = N(y_n|0, WW^T, \beta^{-1}I) \quad (4)$$

Then if the data be considered independent and identically distributed, the product of individual probabilities will result in the likelihood of the whole set:

$$p(Y|W, \beta) = \prod_{n=1}^{N} p(y_n|W, \beta) \quad (5)$$

The solution of $\boldsymbol{W}$ will be obtained by maximizing equation (5). Since the result of this model is calculated by maximizing the probability, it can be called the probabilistic PCA.

(1) Assume $x_n = Wz_n + \epsilon_n$, which x_n is the observation n, W is the mapping matrix $W = [w_1, \dots, w_n]$, and $\epsilon_n \sim N(0, \sigma^2)$ is the Gaussian noise. **(2)** since ϵ_n is considered Gaussian, the conditional distribution of x_n would be: $p(x_n|z_n, W, \sigma^2) = N(Wz_n, \sigma^2 I_D)$. **(3)** Assume a Gaussian prior on z_n as $P(z_n) = N(0, I_K)$. (4) integrating out latent variables: $p(x_n|W, \sigma^2) = N(0, WW^T, \sigma^2 I_D)$.

Algorithm 1: Sudoku of Probabilistic PCA

B. Landmark Isomap [LIM]

One of the well-established non-linear dimensionality reduction techniques is called Isomap. However, one of the issues of this technique is its scalability that has been improved with another technique named Landmark-Isomap (L-Isomap).

Algorithm 2 LIM

In the Isomap, the algorithm has input data space $X \in R^{D\times N}$ with all N data samples and D dimensions that it tries to embed the input samples into a lower-dimensional space $X \in R^{d\times N}$, In such a way that the geodetic distance between the points is maintained. In this algorithm, the starting point is to create an indirect neighborhood graph $G = (V, E)$, where each node $Vi \in V$, corresponding to the point $x_i \in X$, is connected with its k-nearest neighbor, In such a way that each point node $v_i \in V$ is connected to the point $x_i \in X$ by its k-nearest neighbor and also each edge $e_{ij} \in E$ is appointed to weight D_{ij} that shows the Euclidean distance between points x_i and x_j. In the second, the Isomap algorithm calculates the shortest path between every two points in a graph to estimate the geodesic

distance D_{ij}^{G} that uses the Dijkstra's or Floyd's shortest-path algorithm. matrix D_G is the geodesic distance between all the data points. Finally, Isomap runs MDS on matrix D_G to find the low-dimensional embedding. This method works well for many different fields, but unfortunately, if there is a large data space N, the time consumed by this algorithm to find the shortest path is not optimal in ($O(kN^2 \log N)$) and the MDS eigenvalue decomposition ($O(N^3)$) [5,6]. To improve the speed of these two values, L-Isomap is presented. In L-Isomap, a series of points are considered random points. Instead of calculating the shortest path between all available data, the distance between points and random points, called Landmark points, is calculated. The classical MDS is then applied to the result obtained from the $n \times N$ geodesic distance matrix to embed the low dimensions of the Landmark points. The embeddings of the remaining points are achieved by their geodesic distance to the landmark points by a fixed linear change. In this way, the time complexities of calculating the shortest path and the MDS computation are decreased to $O(knN \log N)$ and $O(n^2N)$.

Landmark Candidate is a very significant procedure in the L-Isomap is to create the k-nearest neighborhood graph. Many points are similar because of shared data, so some points can be omitted to make the neighborhood graph and diagram easier. For this purpose, we select candidate landmark points using a simple neighborhood graph. This algorithm is summarized in Algorithm 1 and it can run in time complexities $O(N \log N)$.

We apply Algorithm 1 to achieve the neighborhood subset $\Omega* = \{S_{i_1}, \dots, S_{i_v}\}$ and landmark candidates set $X_c = \{x_{i_1}, \dots, x_{i_v}\}$ where x_{i_r} defines each neighborhood S_{i_r} in $\Omega*$ and r={1, ..., v}.

(1) Let $\Omega* = \Phi$. (2) If $u_{S_i} = 0$ for all i then stop: $\Omega*$ is the cover, where $i = 1, 2, \dots, N$. Otherwise, find a subscript $i_r \in \{1, 2, \dots, N\}$, maximizing the ratio $u_{S_i}/c(S_i)$ and proceed to Step (3). (3) Add S_{i_r} to $\Omega*$, replace each Si by $S_i - S_{i_r}$ and return to Step (1).

Algorithm 2: Sudoku of landmark Isomap

B. Locally Linear Embedding [LLE]

LLE can be mentioned as one of the helpful techniques in the area of dimensional reduction. In comparison with the Isomap technique, there is a slight difference between them. While both compute the distance between points as small linear neighborhoods in the nonlinear manifold, Isomap uses the geodesic distance to obtain the graph traversal, but LLE finds the weights that execute the local linear interpolations [7].

Algorithm 3 LLE

The LLE algorithm consists of three main stages:

1) Discovering K-Nearest Neighbors of each point via Euclidean distance. 2) Calculating the local interpolation weight matrix of each sample point. For this purpose, the error function must be minimized as follows:

$$\min \varepsilon(W) = \sum_{i=1}^{N} \left| x_i - \sum_{j=1}^{K} w_j^i x_{ij} \right|^2 \quad (6)$$

Such that.

$$\sum_{j=1}^{K} w_j^i = 1 \quad (7)$$

Where the k neighbor of x_i are defined as x_{ij} ($j=1,\dots,k$), and the weights between x_i and x_{ij} are presented as w_j^i. A local COV matrix also should be created for W matrix as follows:

$$Q_{jm}^i = \left(x_i - x_{ij}\right)^T (x_i - x_{im}) \quad (8)$$

By combining the last two equations, the local optimization reconstruction weight matrix can be obtained:

$$w_j^i = \frac{\sum_{m=1}^{k} (Q^i)_{jm}^{-1}}{\sum_{p=1}^{k}\sum_{q=1}^{k} (Q^i)_{pq}^{-1}} \quad (9)$$

3) computing the result of the sample point by its weight matrix and the nearest neighbors.

$$\min \varepsilon(Y) = \sum_{i=1}^{N} \left| y_i - \sum_{j=1}^{K} w_j^i y_{ij} \right|^2 \quad (10)$$

Such that.

$$\sum_{i=1}^{N} y_i = 0 \quad (11)$$

$$\frac{1}{N} \sum_{i=1}^{N} y_i y_i^T = I \quad ; (I \text{ is a unit matrix}) \quad (12)$$

(1) finding K nearest neighbors for each $\vec{x}_i$ **(2)** consider weight matrix w and find the Residual Sum of Squares as: $RSS(w) \equiv \sum_{i=1}^{n} \|\vec{x}_i - \sum_{j \neq i} w_{ij} \vec{x}_j\|^2$ where $w_{ij} = 0$ unless $\vec{x}_j$ is one of the $\vec{x}_i$'s k nearnest neighbors and for each i, $\sum_j w_{ij} = 1$. **(3)** finding the coordinates Y: $\phi(Y) = \sum_{i=1}^{n} \|\vec{y}_i - \sum_{j \neq i} w_{ij} \vec{y}_j\|^2$ where $\sum_i Y_{ij} = 0$ for each j, and $YY^T = I$.

Algorithm 3: Sudoku of Locally Linear Embedding

II. Reference

[1]: Zhao, J., & Jiang, Q. (2006). Probabilistic PCA for t distributions. Neurocomputing, 69(16-18), 2217-2226.
[2]: Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res*, *10*(66-71), 13.
[3]: Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61(3), 611-622.
[4]: Shi, H., Yin, B., Kang, Y., Shao, C., & Gui, J. (2017). Robust l-Isomap with a novel landmark selection method. Mathematical Problems in Engineering, 2017.
[5]: E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[6]: R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, article 345, 1962.
[7]: Ventura, D. (2008). Manifold Learning Examples–PCA, LLE, and ISOMAP. Department of Computer Science, Brigham Young University, Provo, UT, USA.

III. Implementing

In this section, we will explain about the coding section and out procedure in coding. We had followed the below steps:

a) Dividing each dataset into train and test sections stablished in DatasourceReader class.
b) Dividing each dataset into 10 folds stablished in DatasourceReader class.
c) Creating functions that apply GridsearchCV on train and test sections to optimize the parameters of each classifier.
d) Creating function for each classifier KNN, RF, MLP, and RBF which are stablished in Utils class. For RBFNN we customized the original class to be able to use it in Python.
e) Creating function for each dimensionality reduction algorithms with each classifier separately for each fold, stablished in Calculation Class.
f) Creating function that includes each DR algorithm with each of the Classifiers will be called on each original dataset to calculate the explained variance (each DR algorithm and classifier is again optimized).

IV. Experimental

In this section, we presented the results obtained from implementing the DR and Classification methods in Python in Visual Studio Code software. The listed DR algorithms in out project code is as Probabilistic PCA, Landmark Isomap, and Locally Linear Embedding. The classifiers consist of KNN, RF, MLP, and RBF neural network.

There are 13 datasets available for us that dataset 1 to 13 had almost the same behavior. Therefore, we will present the results of analyzation of dataset 1 and 13 for their significant different performance. for splitting the data more efficiently, K-fold cross-validation has been used as a strong factor for validation. In the K-fold cross-validation all the entries in the original training dataset are managed for training as well as validation. In each dataset, there were some required tasks such as:

a) Analyzing the performance of each classifier for each dataset in the "Baseline" section, through two factors: F-measure and Accuracy. This has been done by considering 10 folds on each dataset.

b) calculating the variability explained by each extracted feature. For this purpose, we created a function that analyzes the dimensionality reduction algorithms by classifier to generate the optimal reduced-dimension matrix for calculating the ratio of variance of a feature to the variance of the features of the original data set without considering K-fold. The summation of these ratios should be then provided in the section "overall variability explained".

c) Calculating the performance of each classifier on each dimensionality reduction algorithms in the "with dimensionality reduction" section through two factors of F-measure and Accuracy on each 10 folds of the dataset. In this regard, we designed a function that runs each classifier on each dimensionality reduction algorithms, we simultaneously calculated the ratio of the variance of each feature to the variance of the original dataset, and the optimized matrix of features. This process was so time-consuming that although different values of parameters has been tested, but we couldn't achieve a good output.

One important thing that should be mentioned is that for implementing the classifiers we take advantage of the function "GridSearchCV" to find the optimal parameter for each classifier which makes the coding superior to the case where this information is entered manually or by try and error.

Since we implemented these steps in Python and we had the problem of lack of available information, we faced some challenge for applying PPCA, RBFNN, and LIM. For solving this issue for the PPCA and RBFNN, we decided to customize the class of these methods and consequently we had limited input parameters which resulted in a very complex calculation. On the other hand, we couldn't find any information for LIM in Python so we tried to implement it in Octave, but because of not being

familiar to this software we couldn't achieve any result.

The output of each section will be provided and explained in detail in the following.

i. Baseline Section

In the baseline section of the project, the performance of each classifier will be analyzed through two factors: F-measure and Accuracy. F-measure is a measure of a model's accuracy on a dataset. By comparing the results in the first dataset (see figure 1-a and 1-b) it can be concluded that Random Forest (RF) is performing better than the rest of the algorithms with the average accuracy and f-measure of 99%. we have considered different important parameters in coding section, in order to achieve the best or the optimal value of maximum depth which was equal to 100. On the other hand, KNN performed constantly in each fold with the average accuracy of 70% with the optimum K value of 10.

The poor performance of MLP that was common in almost all the datasets. Even by changing the numbers of hidden layers in coding section, with the max iteration of 200, the performance remained stable.

The results obtained by RBF Neural Network classifier presents the average f-measure and accuracy of 50% in dataset 1. We have implemented this classifier with the dense range of 70, and we have considered 100 epochs for each fold reporting accuracy, f-measure, and loss value to find the best/optimal value of accuracy and the lowest value of loss.

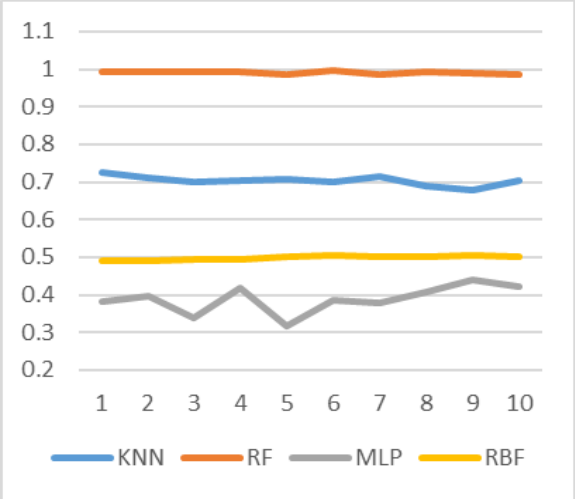

Figure 1-a. F-measure in baseline dataset 1

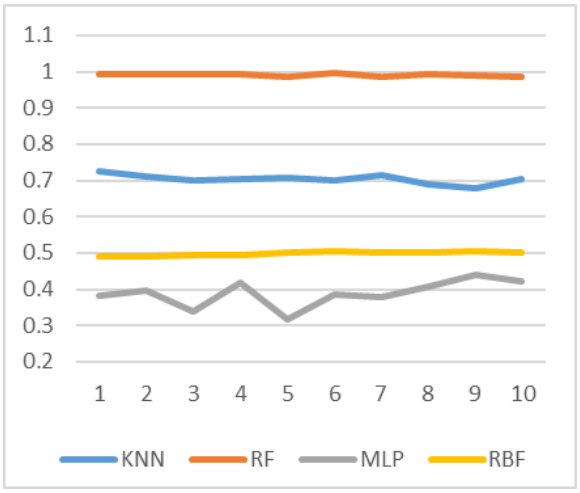

Figure 1-b. F-measure in baseline dataset 1

In data set 13, we had some particular changes in the performance of the classifiers (see figure 2-a and 2-b). This time RBF had the highest performance with the average accuracy of 81% and the minimum loss value of 0.1%. For the KNN classifier we had provided the range of {5, 10, 15} for the K value. For the dataset 13, this algorithm chose 10 and 15 for all folds and resulted the average accuracy of 80%. RF classifier performed almost the same as KNN with the average accuracy of 80%. The last classifier, MLP, performed similarly to RF and KNN with the average accuracy of 80%. Except for the RBF algorithm, the common point between the classifiers was the significant difference between the value of f-measure and accuracy. The average F-measure obtained from RF, KNN, and MLP was around 10-15% that could mean that the abnormality of the data scatter varies the values of accuracy and f-measure.

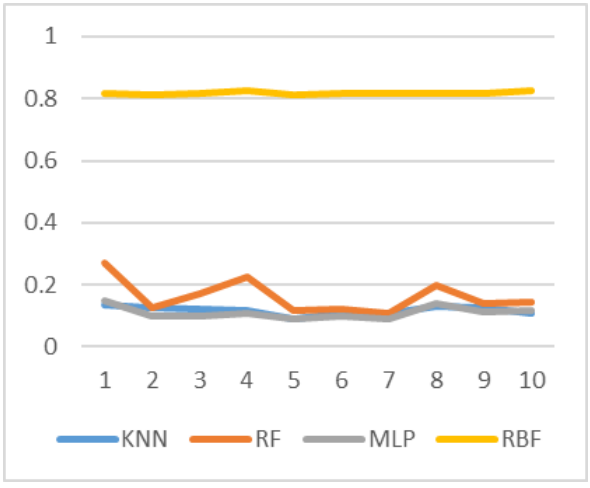

Figure 2-a. F-measure in baseline dataset 13

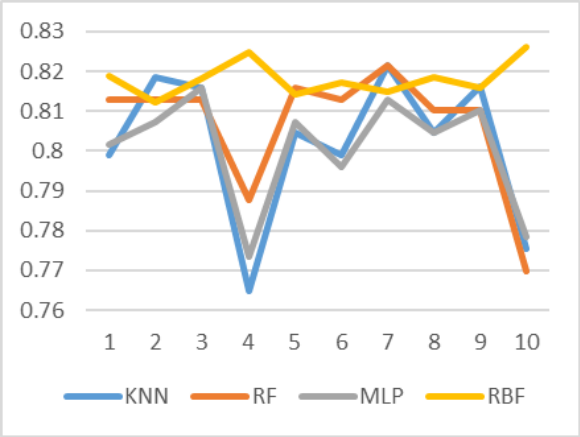

Figure 2-b. F-measure in baseline dataset 13

ii. Variability Explained by Each Extracted Feature Section

the proportion of the summation of the variance of each selected feature to the original dataset's variance is called ratio. The summation of ratio of the dataset 1 and dataset 13 for each classifier is presented in tables 1 and 2, respectively.

Overall Variability Explained	
PPCA	LLE
KNN	
0.0675	0.0544
RF	
0.0647	0.2398
MLP	
0.0299	0.0068

Table 1. overall variability explained dataset 1

Overall Variability Explained	
PPCA	LLE
KNN	
0.0168	0.0947
RF	
0.0458	0.0426
MLP	
0.0801	0.064055723

Table 2. overall variability explained dataset 13

iii. With Dimensionality Reduction Section

In this section, we presented a comparison to evaluate the difference caused on F-measure and Accuracy of the classifier output, in both situation of using and not using DR methods. The studied classifiers are KNN, RF, MLP. This assessment is done on two dataset 1 and 13 which are presented in Table 3 and 4, respectively.

For the dataset 1, KNN worked with K-neighbor of 10 and 15, RF resulted better with max-depth of 100, and MLP had the max-iteration of 200.

Dataset 1						
KNN	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	71.62%	72.40%	1.65%	1.77%	1.84%	2.86%
fold 2	70.80%	71.09%	2.72%	3.47%	0.68%	0.69%
fold 3	71.29%	69.89%	6.07%	5.52%	3.46%	3.09%
fold 4	70.64%	70.23%	27.19%	28.40%	1.24%	2.86%
fold 5	70.58%	70.69%	2.33%	2.70%	1.53%	2.34%
fold 6	72.03%	69.89%	5.23%	6.61%	1.00%	2.00%
fold 7	72.08%	71.43%	2.00%	2.30%	0.54%	0.57%
fold 8	70.40%	69.09%	4.55%	4.55%	0.79%	0.91%
fold 9	70.03%	67.94%	1.73%	1.79%	1.04%	2.34%
fold 10	70.86%	70.34%	2.75%	3.33%	0.74%	0.86%
RF	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	99.17%	99.14%	2.13%	2.17%	0.19%	0.63%
fold 2	99.22%	99.20%	4.13%	4.57%	0.10%	0.06%
fold 3	99.13%	99.14%	2.96%	3.03%	1.77%	5.20%
fold 4	99.06%	99.14%	7.13%	7.43%	0.79%	2.69%
fold 5	98.80%	98.74%	31.72%	34.69%	1.11%	1.60%
fold 6	99.57%	99.54%	12.12%	13.14%	0.18%	0.29%
fold 7	98.70%	98.74%	16.23%	14.15%	0.09%	0.17%
fold 8	99.20%	99.20%	10.23%	17.15%	0.10%	1.26%
fold 9	99.05%	99.03%	18.23%	2.36%	2.15%	4.06%
fold 10	98.78%	98.74%	2.82%	32.15%	0.18%	0.63%
MLP	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	33.84%	38.11%	14.00%	16.84%	0.99%	0.94%
fold 2	35.25%	39.71%	14.24%	9.86%	1.01%	0.93%
fold 3	30.50%	33.71%	6.62%	8.24%	1.08%	0.74%
fold 4	37.30%	41.77%	14.87%	5.45%	0.96%	1.02%
fold 5	28.21%	31.54%	15.45%	16.77%	1.10%	1.11%
fold 6	37.19%	38.63%	13.48%	19.82%	1.05%	0.94%
fold 7	33.98%	37.66%	14.92%	14.09%	0.98%	1.02%
fold 8	37.84%	40.80%	17.11%	17.03%	0.91%	1.21%
fold 9	41.02%	44.06%	7.03%	15.42%	1.08%	1.18%
fold 10	37.78%	42.17%	13.88%	18.48%	1.23%	1.00%

Table 3. accuracy of dataset 1 with/without DR

In this process, numerous numbers of parameters have been examined for DR algorithms that improved the model. One of the most important parameters in DR examination is N-Neighbors that changing this parameter effects differently on accuracy and f-score. The optimal value of this parameter turned to be N=10. However, although this was the optimal value, the accuracy, f-measure, and ratio didn't seem efficient in our results. But with these results, PPCA works better on all the classifiers comparing with LLE, as you can see in Table 3. In addition to this, we once again have evaluated the optimal parameters for each classifier.

Figures 3-a to 3-b shows a comparison on the performance of each DR on the output of each classifier in each fold in dataset 1.

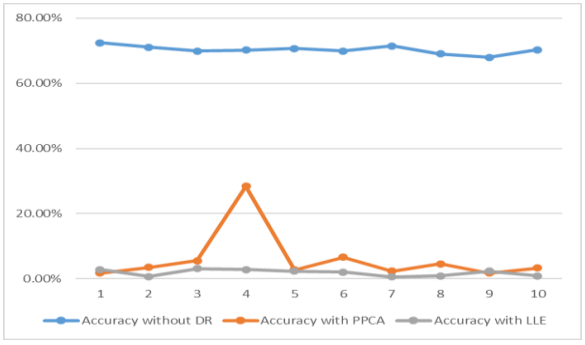

Figure 3-a. performance of KNN without/with PPCA and LLE on accuracy - dataset 1

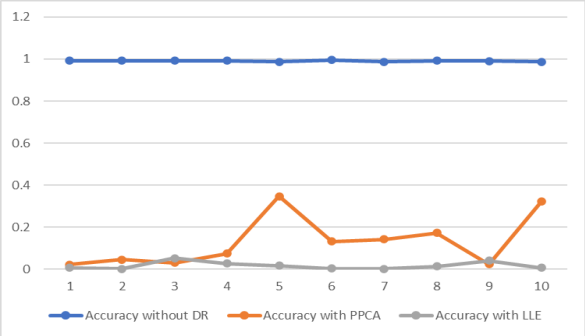

Figure 3-b. performance of RF without/with PPCA and LLE on accuracy - dataset 1

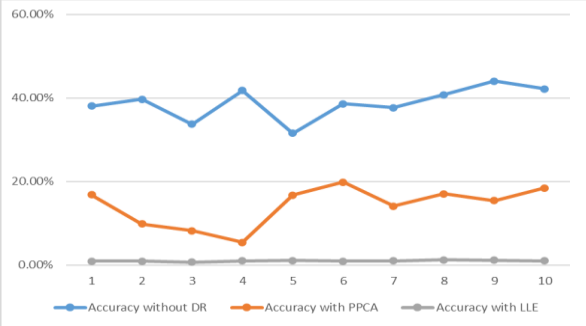

Figure 3-c. performance of MLP without/with PPCA and LLE on accuracy - dataset 1

You can find the same results for dataset 13 in the following. For the dataset 13, KNN worked with K-neighbor of 10, RF resulted better with max-depth of 100, and MLP had the max-iteration of 200.

Dataset 13						
KNN	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	13.65%	79.89%	11.01%	78.47%	11.06%	79.32%
fold 2	12.78%	81.87%	9.84%	79.04%	9.91%	80.45%
fold 3	12.32%	81.59%	9.94%	80.74%	9.91%	80.45%
fold 4	11.64%	76.49%	9.98%	81.59%	11.47%	75.35%
fold 5	8.94%	80.45%	12.23%	72.52%	8.90%	80.17%
fold 6	10.64%	79.89%	9.86%	81.02%	9.93%	80.74%
fold 7	10.86%	82.15%	9.89%	80.17%	8.88%	79.89%
fold 8	13.17%	80.45%	8.97%	81.30%	12.08%	76.49%
fold 9	12.68%	81.61%	8.92%	80.45%	9.96%	80.74%
fold 10	10.96%	77.56%	9.79%	78.75%	10.92%	77.56%
RF	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	27.27%	81.30%	11.06%	79.32%	11.01%	78.75%
fold 2	12.58%	81.30%	9.95%	81.02%	9.93%	80.74%
fold 3	17.26%	81.30%	9.98%	81.59%	9.98%	81.59%
fold 4	22.64%	78.75%	10.88%	77.05%	12.21%	73.37%
fold 5	11.50%	81.59%	8.92%	80.45%	8.91%	79.89%
fold 6	12.36%	81.30%	9.93%	80.74%	10.20%	76.77%
fold 7	10.79%	82.15%	8.99%	81.59%	8.97%	81.02%
fold 8	19.74%	81.02%	11.15%	80.45%	11.15%	80.45%
fold 9	13.85%	81.02%	9.97%	81.30%	10.80%	79.60%
fold 10	14.19%	76.99%	10.92%	77.56%	11.73%	76.70%
MLP	Without DR		With PPCA		With LLE	
	F-measure	Accuracy	F-measure	Accuracy	F-measure	Accuracy
fold 1	14.90%	80.17%	11.71%	76.49%	11.06%	79.32%
fold 2	9.93%	80.74%	12.50%	77.05%	9.95%	81.02%
fold 3	9.98%	81.59%	9.84%	79.04%	9.98%	81.59%
fold 4	10.90%	77.34%	11.52%	70.25%	10.90%	77.34%
fold 5	8.93%	80.74%	9.86%	75.07%	8.93%	80.74%
fold 6	9.85%	79.60%	9.30%	71.95%	9.93%	80.74%
fold 7	9.00%	81.30%	9.99%	81.02%	8.99%	81.59%
fold 8	13.92%	80.45%	11.82%	73.37%	11.15%	80.45%
fold 9	11.13%	81.02%	9.91%	80.45%	9.97%	81.30%
fold 10	11.74%	77.84%	10.55%	71.31%	10.92%	77.56%

Table 4. Accuracy of dataset 13 with/without DR

As for the dataset 1, numerous numbers of parameters have been examined for DR algorithms that improved the model for dataset 13 as well. The optimal value of parameter N-Neighbor is again N=10. Therefore, in this dataset the resulted accuracy of the performance classifiers with dimensionality reduction, presented better values compared to dataset 1 and, in some folds, even performed better than the accuracy without dimensionality reduction.

Figures 4-a to 4-b shows a comparison on the performance of each DR on the output of each classifier in each fold in dataset 13.

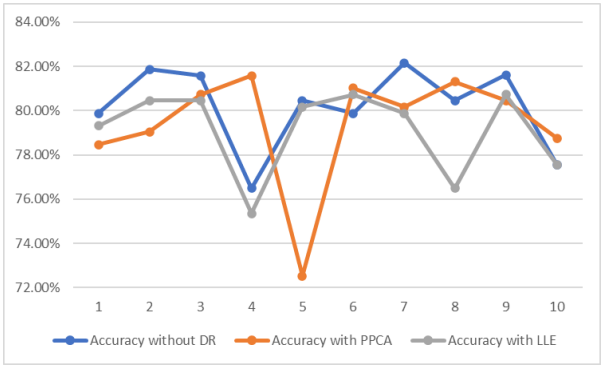

Figure 4-a. performance of KNN without/with PPCA and LLE on accuracy - dataset 13

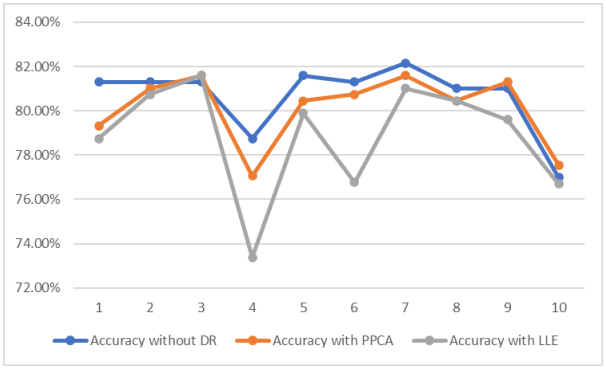

Figure 4-b. performance of RF without/with PPCA and LLE on accuracy - dataset 13

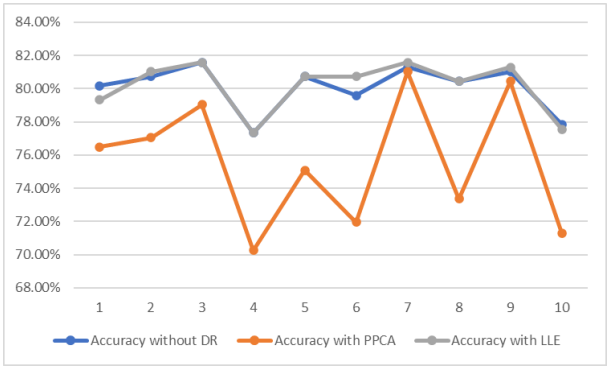

Figure 4-c. performance of MLP without/with PPCA and LLE on accuracy - dataset 13

V. Conclusion

Dimensionality Reduction (DR) methods can be used in data pre-processing to achieve efficient data reduction. Using the dimensionality reduction methods, will probably improve the model performance. The reduced size of samples in dimensionality reduction leads to less calculation and much faster pace in performing the algorithm.

Although using DR algorithms on different datasets should have better results comparing to not using them, in our project, depending on the dimensions of the dataset, DR showed different accuracy. In other words, for large scale datasets we didn't see a good performance and for small scale datasets, had a relatively good performance.