University of Windsor
Faculty of Engineering

ELEC8900-76-R-2021F: Applied Machine Learning

Fall 2021

Project DR03

Ishita Joshi (110028901) and Sanjay Kumar Gengaiyan (110038152)

E-mails: {joshi14; gengaiy} @uwindsor.ca

Submitted to:

Professor: Roozbeh Razavi Far

Date: 29-11-2021

I. INTRODUCTION

This report summarizes the results of implementing dimensionality reduction techniques: Linear Discriminant Analysis (LDA), Sammon Mapping (SM) and Isomap Embedding (IM) on 13 given datasets with 17.5K, or 35K, or 3541 samples, 354 or 220 features, and 70 or 10 classes. The reduced datasets obtained using these algorithms are then passed on to four classifiers: Naive Bayes Classifier (NBC), Support Vector Machines (SVM), Decision Trees (DT), and Multi-layer Perceptron★ (MLP).

II. DIMENSIONALITY REDUCTION ALGORITHMS

Dimensionality reduction techniques are applied during data preprocessing and the goal is to project the dataset onto a lower dimension for better visualization and easier classification. The Dimensionality reductions algorithms used in the project are briefly explained in the following sections.

A. Linear Discriminant Analysis (LDA)

LDA is a supervised classification algorithm that is commonly used as a dimensionality reduction technique. It projects a d-dimensional feature space to a k-dimensional feature subspace. LDA is a linear transformation technique just like Principal Component Analysis (PCA), but it also maximizes separability between classes. Fisher [1] first introduced LDA for binary classification in 1936 and Rao [2] extended it for multiclass classification in 1948.

Conceptually, LDA creates a new axis and projects the data onto it in a way that the separation between means of each class is maximized and the scatter within each class is minimized. LDA extracts eigenvectors and eigenvalues from the scatter matrices or from covariance matrices. It then arranges the eigenvectors in decreasing order and chooses the top-k among them. The concept behind this is that features with eigenvalues nearer to zero are less informative and hence, can be discarded. Algorithm 1 represents the pseudo-code for LDA [3].

Algorithm 1 *LDA*

Input: Dataset X_i, lower dimension d
Output: k dimensional feature subspace Y_i such that (k<d)

1. Compute d-dimensional mean vectors $\mathbf{m_i}$

$$m_i = \frac{1}{N_i} \sum_{x \in D_i}^{N} x_k$$

2. Compute the within-class and between-class scatter matrices $\mathbf{S_W}$ and $\mathbf{S_B}$:

$$S_W = \sum_{i=1}^{c} S_i$$

Where S_i is the scatter matrix for every class

$$S_i = \sum_{x \in D_i}^{N} (x - m_i)(x - m_i)^T$$

And between-class scatter matrix

$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

Where **m** is the overall mean

3. Solve the matrix $S_w^{-1} S_b$ to generate eigenvectors and eigenvalues.
4. Sort the eigenvectors in decreasing order of eigenvalues, select the top-k among them and create a $d \times k$-dimensional transformation matrix **W**
5. Transform the data to new k-dimensional subspace $Y = X \times W$

B. Sammon Mapping

Sammon's mapping algorithm is a nonlinear mapping algorithm proposed by John W. Sammon in 1969 [4]. It maps the given dataset to a lower dimension in a way that the structure is preserved, and it does

★Initially assigned Recurrent Neural Network but changed to MLP.

so by conserving the Euclidean distance between the data points.
If we have L-dimensional data containing N data points Xi to be mapped on a d-dimensional subspace (d<L) Yi, then dij* and dij are the distances between vectors Xi and Xj in the L-subspace and d-subspace respectively. Distance between all these data points of Xi and Yi is calculated, and an error function E is defined that gives an idea about structure lost in the transformation.

$$E = \frac{1}{\sum_{i<j}[d_{ij}^*]} \sum_{i<j}^{N} \frac{[d_{ij}^* - d_{ij}]^2}{d_{ij}^*}$$

The goal here is to minimize this error using a steepest descent technique. Algorithm 2 describes the pseudo-code for Sammon Mapping.

***Algorithm 2** SM*

Input: Given data X_i, k-nearest neighbors, dimension of output d,
Output: Y_i in the lower dimension d.

1. Compute all pair-wise distances in the d-dimensional space $d_y(i,j)$.
2. Initialize Y_i either randomly or using a linear transformation like PCA.
3. Using the steepest descent technique, minimize error E.
4. Update the co-ordinates of all data points of Y_i
5. **Repeat** until E is optimized
6. **Return** Y_i

C. Isomap

Isomap is the isomap mapping algorithm proposed by Tenenbaum *et al*. in 2000 [5]. It is built upon the multidimensional scaling (MDS) which computes the low-dimensional representation based on pairwise geodesic distances. Geodesic is defined as the shortest path on the given surface. MDS computes low-dimensional representation which is usually measured using straight-line Euclidean distance whereas isomap mapping computes shortest path between points in the constructed neighbourhood graph.

The goal is to estimate the distance between the faraway point with the given input-space distances. The input-space distances provide a good approximation for nearby points and distance between the faraway points can be approximated by short-hops between the neighbouring points. After estimating the geodesic distances between all the pairs of points on the manifold classic MDS is applied to the matrix to construct a d-dimensional embedding that best preserves the intrinsic geometry. The coordinate vectors are chosen to minimize the cost function $E = ||\tau(D_G) - \tau(D_Y)||_{L^2}$ where D_Y is the matrix of Euclidean distances $D_Y(i,j) = \left|\left|y_i - y_j\right|\right|$ and $||A||_{L^2}$ is the matrix form of $\sqrt{\sum_{i,j} A_{ij}^2}$. The τ operator converts distances to inner products which characterize the geometry of data that supports efficient optimization. The global minimum is achieved by setting the coordinates $\mathbf{y_i}$ to the top d eigenvectors of the matrix $\tau(D_G)$. [5]

***Algorithm 3** IM*

Input: Dataset X with N data points, k nearest neighbours
Output: $Y = \{y_1, y_2, \ldots, y_n\}$

1. Construct a neighborhood graph. This step determines the neighbor points on the manifold M based on the distance $d_x(i,j)$ between the pairs of inputs i,j in the input space X. The set of neighbors are determined by k nearest neighbors or e-radius.
2. In the second step, the algorithm estimates the geodesic distances $d_M(i,j)$ between all pairs on the manifold M by computing the shortest path lengths $d_G(i,j)$ in graph G [6].

$$\boldsymbol{d_G(i,j) = d_x(i,j)} \textbf{ neighboring i,j}$$
$$\boldsymbol{d_G(i,j) = \infty} \textbf{ otherwise}$$

For each value of k=1,2,…,N, replace $d_G(i,j)$ by $\mathbf{min}\{\boldsymbol{d_G(i,j)}, \boldsymbol{d_G(i,k)} + \boldsymbol{d_G(k,j)}\}$. This gives the final values of $\boldsymbol{D_G = d_G(i,j)}$

3. Apply classical MDS to construct a lower dimensional embedding. In the given matrix D, MDS attempts to find n data points $y_1, y_2, \ldots y_n$ in d dimension such that, if $\hat{d}_{ij}$ is the Euclidean distance between y_i and y_j, then $\hat{D}$ is similar to D [7].
   MDS minimizes the following function:
$$\min_{y} \sum_{i=1}^{n}\sum_{i=1}^{n}(\boldsymbol{d_{ij}^{(X)} - d_{ij}^{(Y)}})^2 \text{ where}$$
$$d_{ij}^{(X)} = ||x_i - x_j||^2 \text{ and } d_{ij}^{(Y)} = ||y_i - y_j||^2$$
4. Obtain Gram Matrix by double centering [7]
   $X^TX = -\frac{1}{2}HD^XH$ where $H = I - \frac{1}{n}ee^T$ and
   e is a column vector of all 1s.
   The equation is reduced to
$$\boldsymbol{\min_{y}} \sum_{i=1}^{n}\sum_{i=1}^{n}(\boldsymbol{x_i^T x_j - y_i^T y_j})^2$$
5. Compute eigenvalue decomposition of Gram matrix $\boldsymbol{S = U\Lambda U^T}$.
   The output of the MDS is $\boldsymbol{Y = \Lambda^{1/2}\vee^T}$ where $\vee$ is the eigen vectors of X^TX corresponding to d eigenvalues and Λ is the top d eigenvalues of X^TX. [7]
6. A p-dimensional representation of Y is obtained by $\boldsymbol{\hat{X} = I_{PxN}\Lambda^{1/2}\vee^T}$ [7]

III. IMPLEMENTATION

This section will cover the languages and tools used, the implementation procedure and some challenges faced during implementation.

A. Language and tools used

- Python 3.8.8
- Scikit-learn 0.24.1
- Pandas 1.2.4
- NumPy 1.20.1
- Sammon mapping function by Tom Pollard [8]
- Editor used: Jupyter Notebook

B. Implementation Procedure

1. Import libraries and dataset.
2. Split dataset into input *X* and output *y*.
3. Perform baseline classification.
4. Apply DR technique (LDA, SM, IM) to extract 2 features (d=2) and compute variance of each feature.
5. Apply NBC classifier, check accuracy and F1-score.
6. Manipulate the value of d and find a value that gives highest accuracy and F1-score.
7. Compute variance of d extracted features.
8. Apply remaining classifiers SVM, DT and MLP to the reduced dataset.
9. Repeat for all datasets.

C. Challenges Faced

- Recurrent Neural Network models (even the lightest one) was too bulky and took hours to run. It was later replaced with MLP which was slow as well but better than RNN.
- Isomap and Sammon mapping algorithms are computationally heavy and had to be implemented on a sampled dataset.
- Isomap and Sammon mapping being distance-based algorithms, their explained variance does not

say much about the ‘goodness of fit’. Sklearn does not provide a function to calculate the explained variance of each feature.

- MLP needs a lot of fine-tuning for layer size, number of hidden layers, learning rate, optimizer, etc.

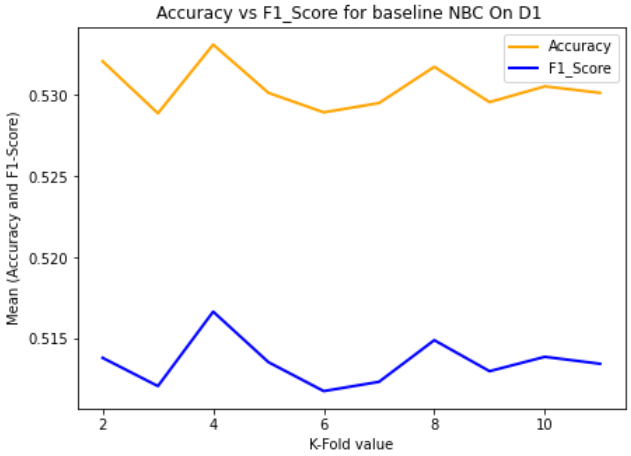
IV. Experimental Results

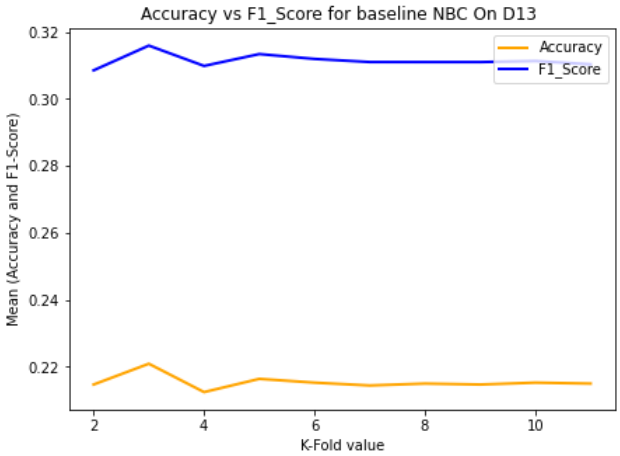
In this section, the graphical results generated during our experiments to find the best combination of hyperparameters will be discussed. Firstly, baseline classification was performed on our classifiers NBC, SVM, DT and MLP and generated graphs for one of the 17k, 35k datasets each and D13. Once the baseline classification results were ready, LDA was implemented followed by IM and SM. For dimensionality reduction, the most important parameter is the number of extracted features. We tried different values of *d* for each algorithm on D1 and implemented it on the remaining datasets.

A general pattern observed throughout our implementation process is that datasets with same number of samples have closely similar accuracies and require the same number of extracted features. The classifier results after applying dimensionality reduction algorithms give a reduced accuracy than the baseline except for dataset 13.

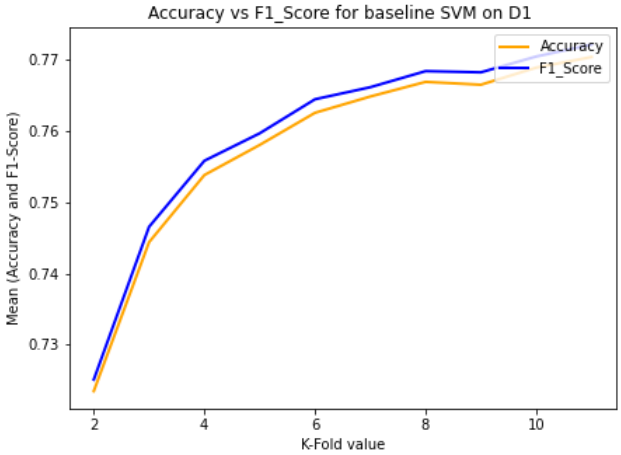
A. Baseline classification

i. Naive Bayes Classifier

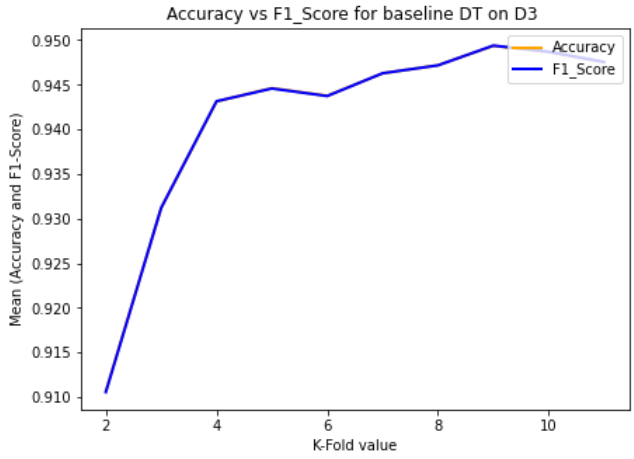


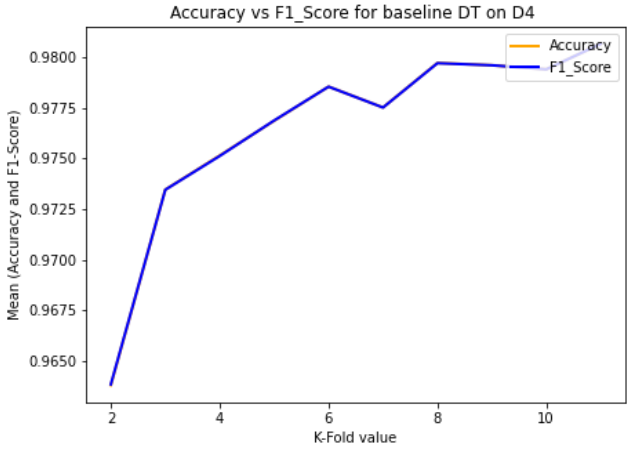


ii. Support Vector Machine

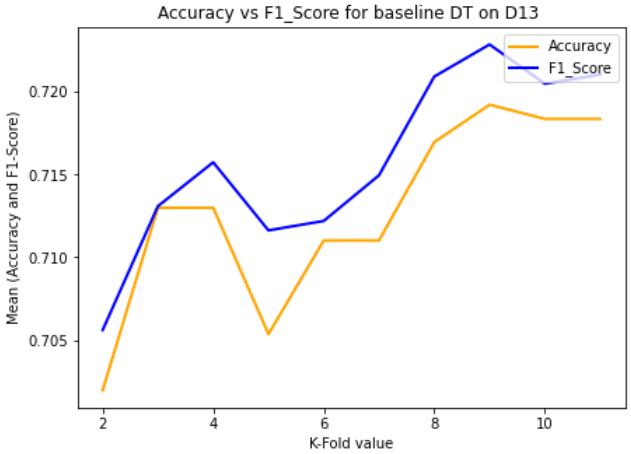


iii. Decision Tree Classifier





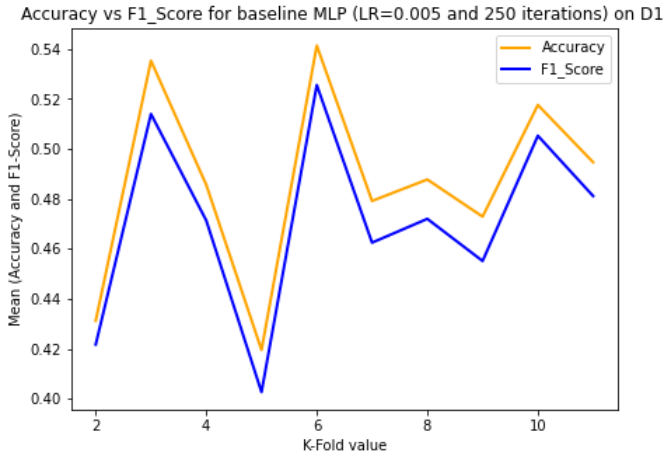
S

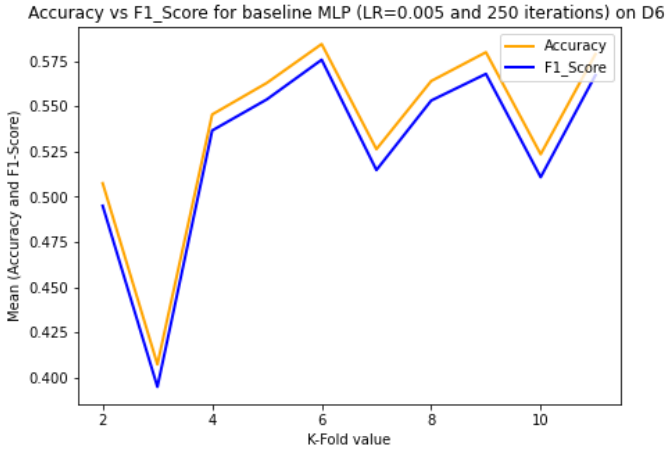


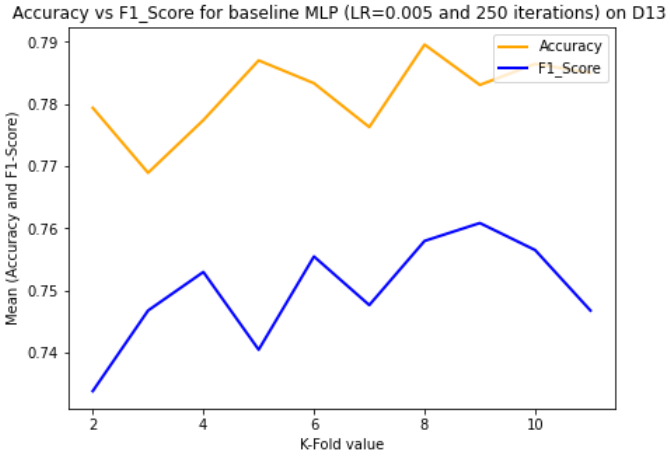
iv. Multi-layer Perceptron

For MLP, different combinations of hyperparameters were experimented on the data. The parameters of the final model are as mentioned:

- 3 hidden layers of sizes (256,128,128)
- Learning rate = 0.005
  Values 0.001 and 0.01 were tried during experimentation and these values were either too slow or too fast hence failed to converge within the maximum number of iterations.
- Optimizer: Adam
- Maximum iterations: 250

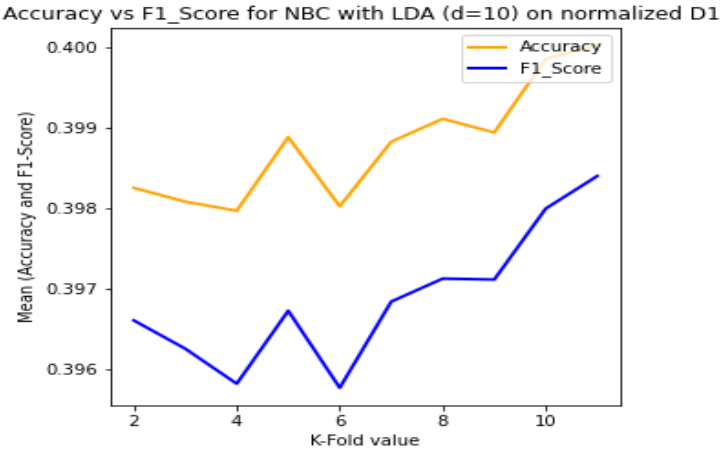


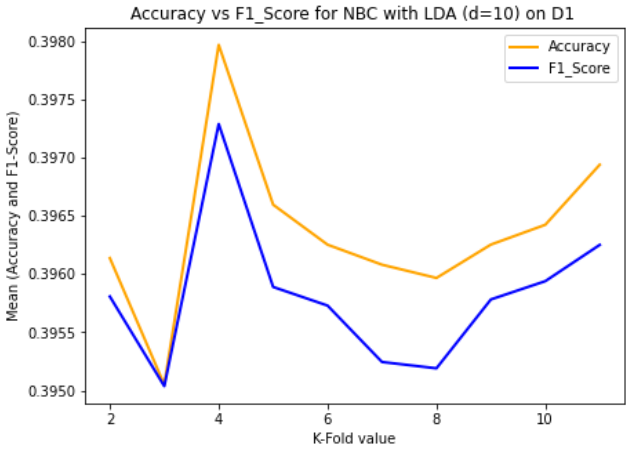




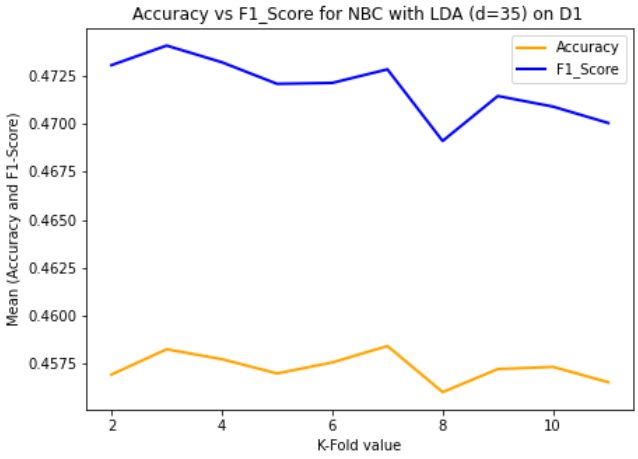
B. Linear Discriminant Analysis (LDA)

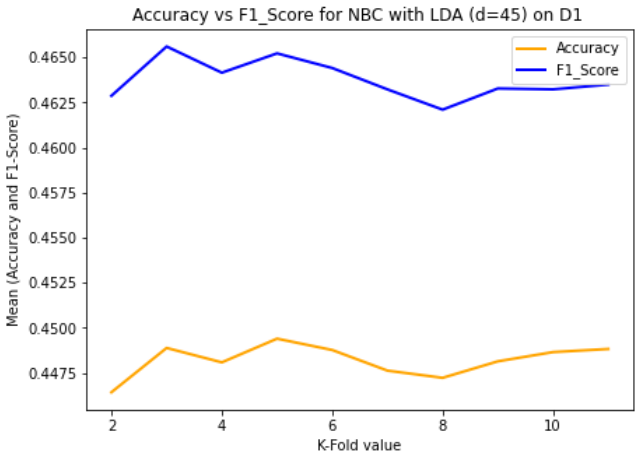
i. LDA with and without data normalization

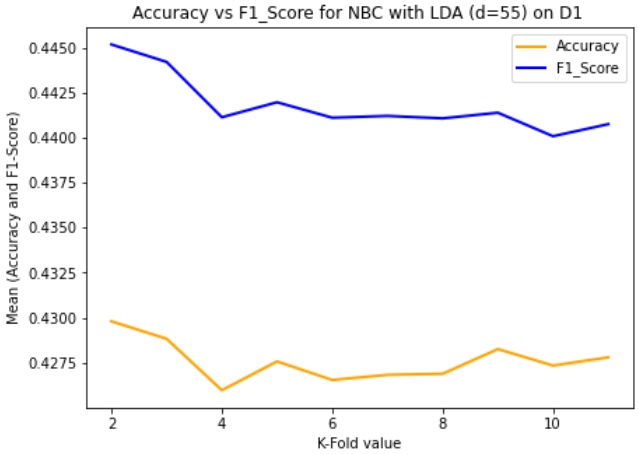


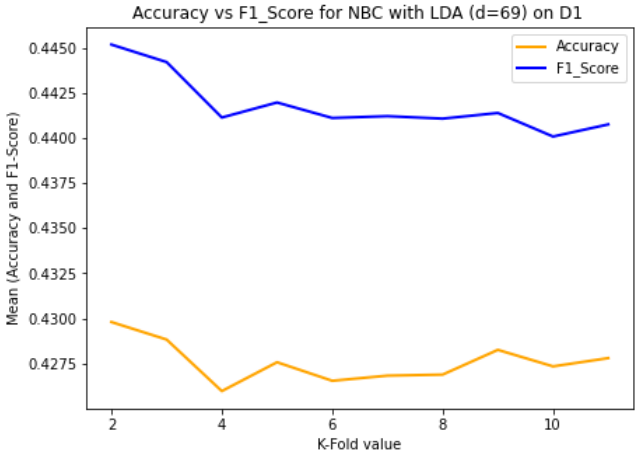


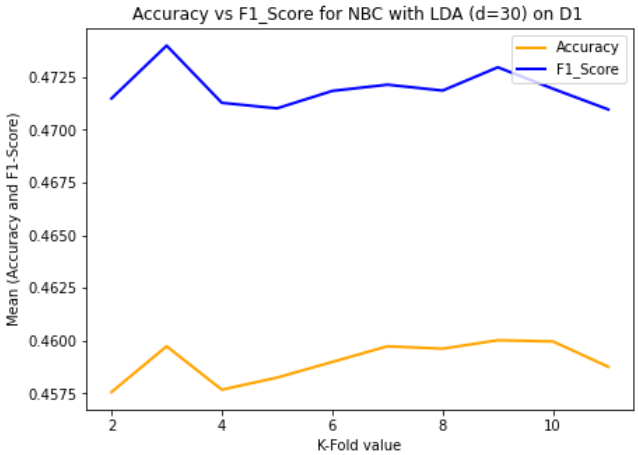
ii. LDA for different values of *d* with NBC on D1

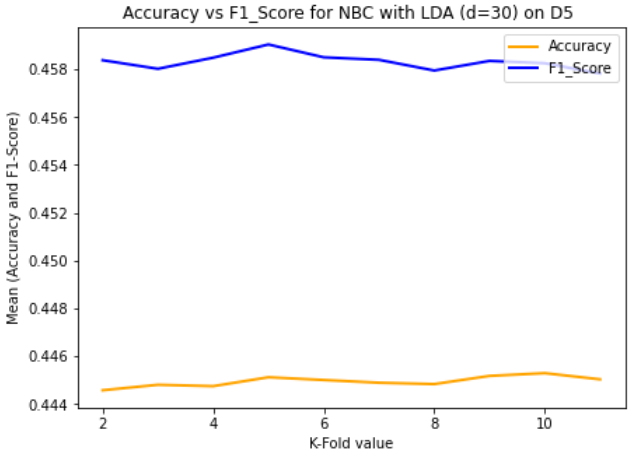


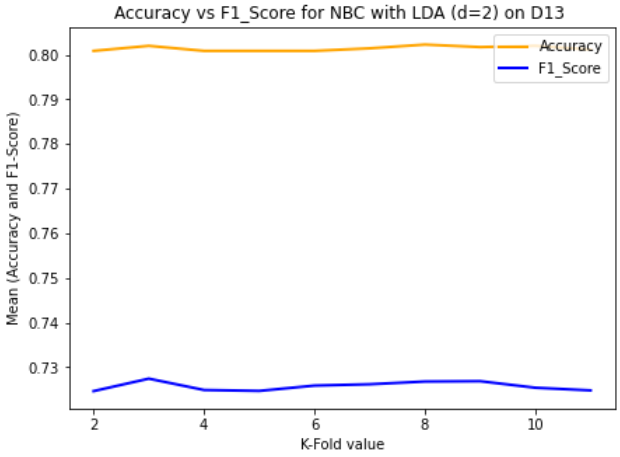



Accuracy vs F1_Score for NBC with LDA (d=55) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.4450
0.4425
0.4400
0.4375
0.4350
0.4325
0.4300
0.4275
2
4
6
8
10

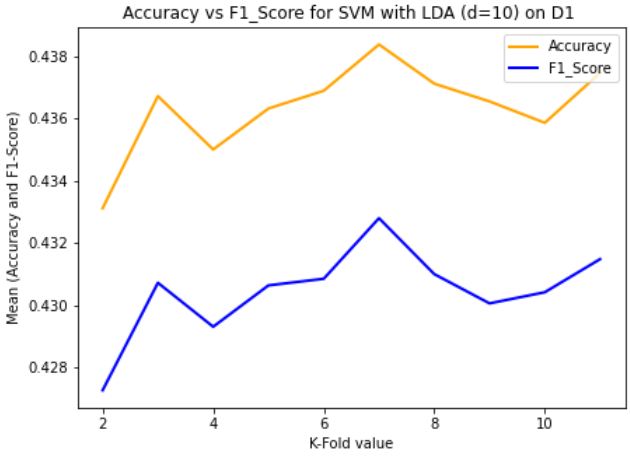

Accuracy vs F1_Score for NBC with LDA (d=69) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.4450
0.4425
0.4400
0.4375
0.4350
0.4325
0.4300
0.4275
2
4
6
8
10

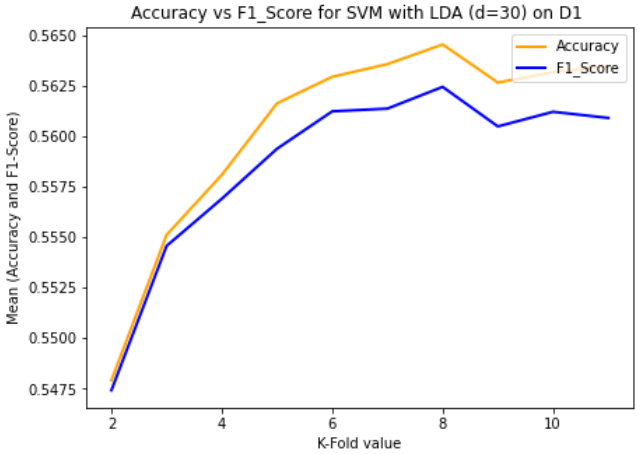

Accuracy vs F1_Score for NBC with LDA (d=30) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.4725
0.4700
0.4675
0.4650
0.4625
0.4600
0.4575
2
4
6
8
10

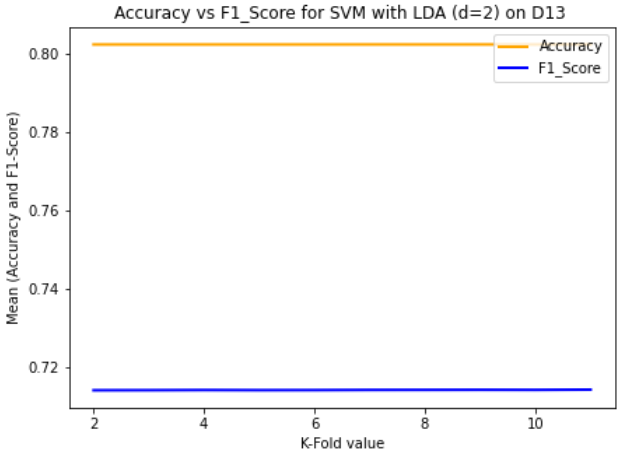




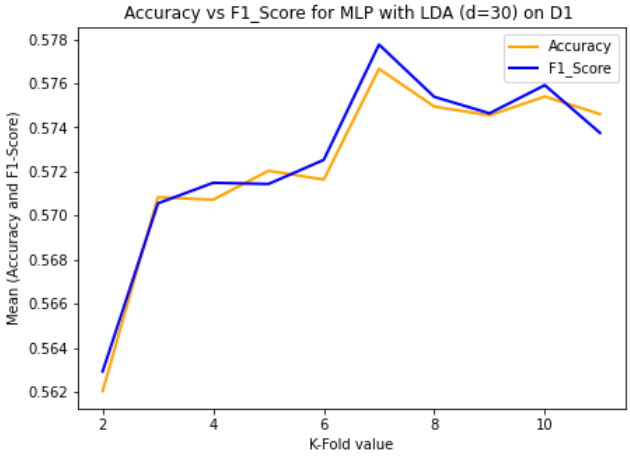
iii. LDA with SVM classifier

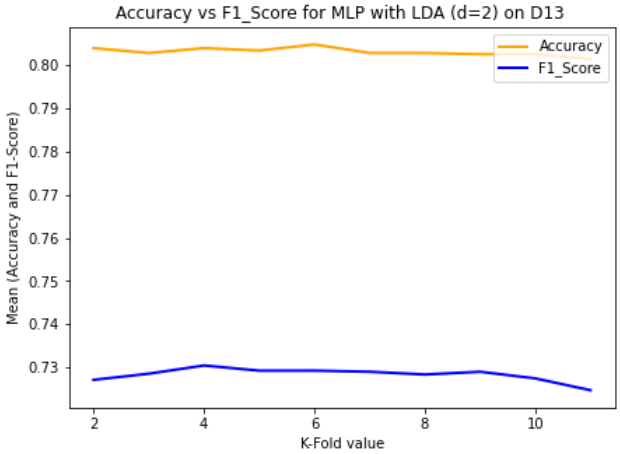




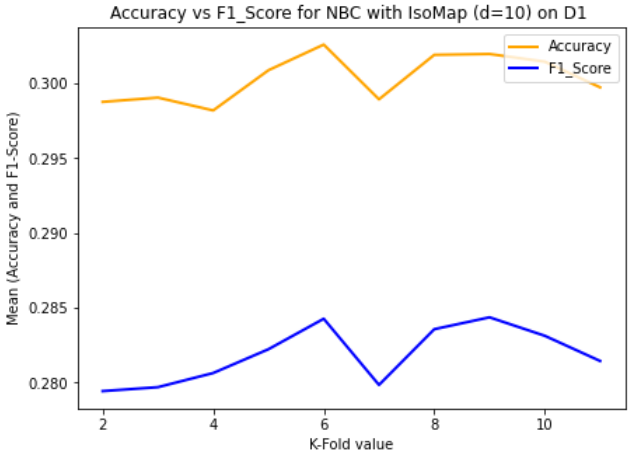


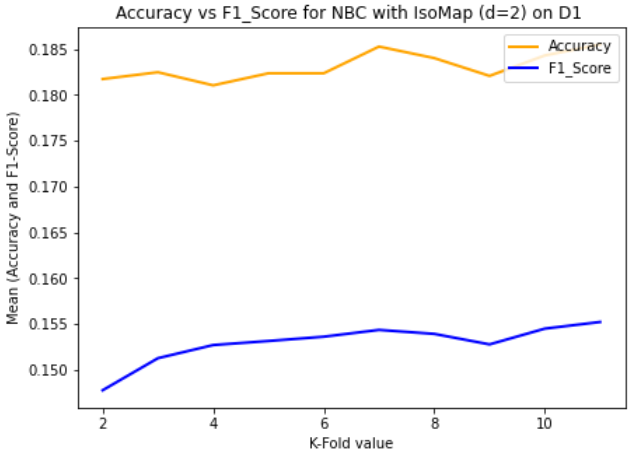
iv. LDA with MLP classifier

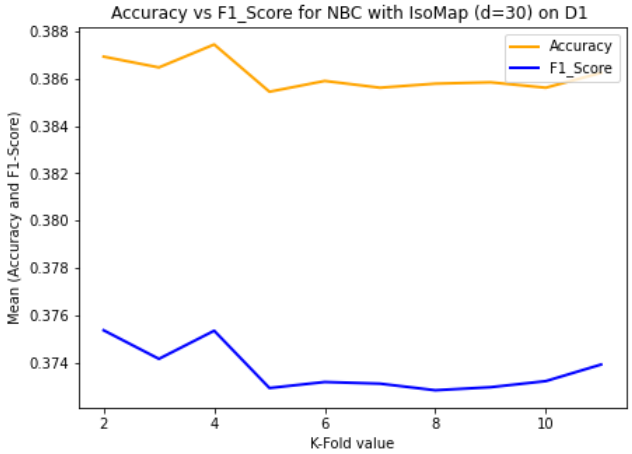


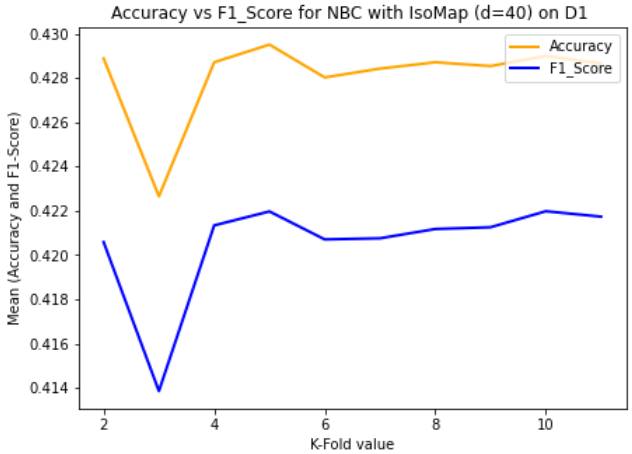


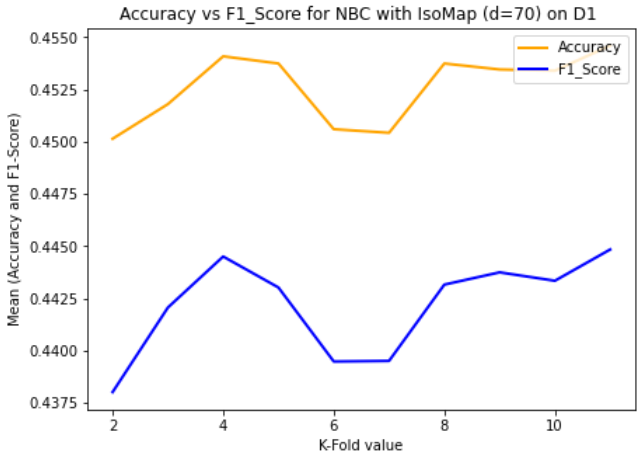
C. Isomap Embedding

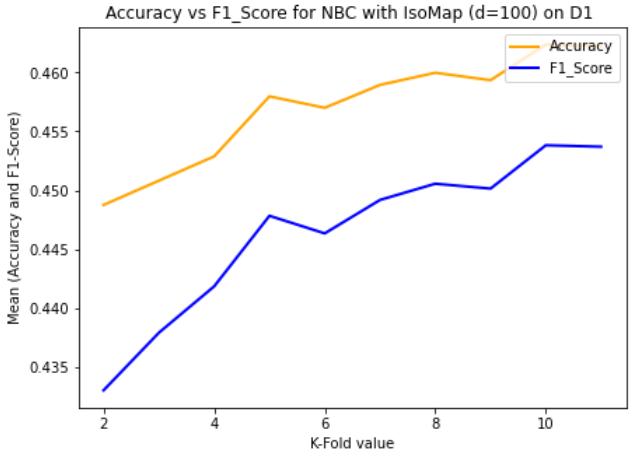


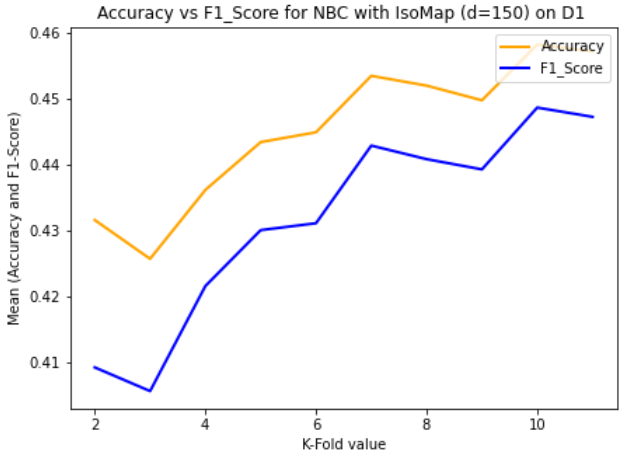


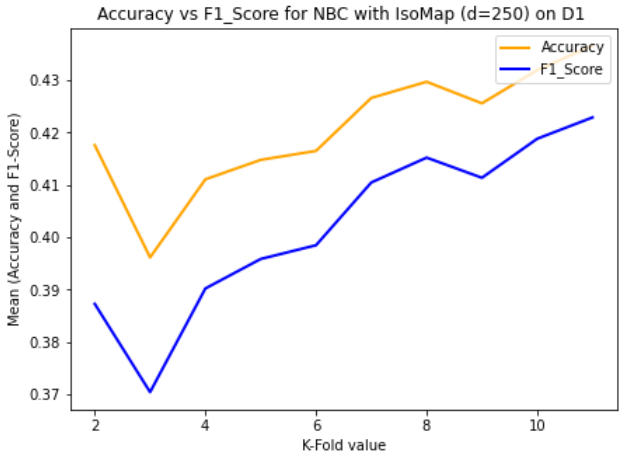

Accuracy vs F1_Score for NBC with IsoMap (d=30) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.388
0.386
0.384
0.382
0.380
0.378
0.376
0.374
2
4
6
8
10

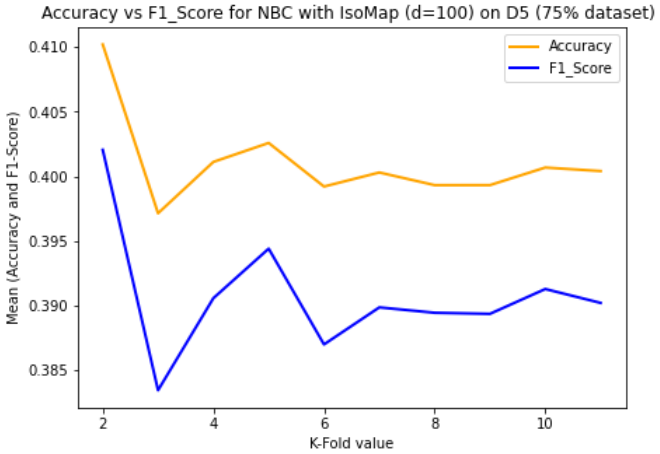

Accuracy vs F1_Score for NBC with IsoMap (d=40) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.430
0.428
0.426
0.424
0.422
0.420
0.418
0.416
0.414
2
4
6
8
10

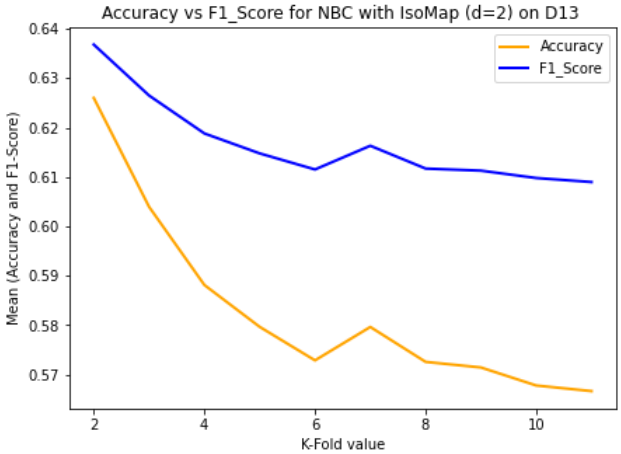

Accuracy vs F1_Score for NBC with IsoMap (d=70) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.4550
0.4525
0.4500
0.4475
0.4450
0.4425
0.4400
0.4375
2
4
6
8
10

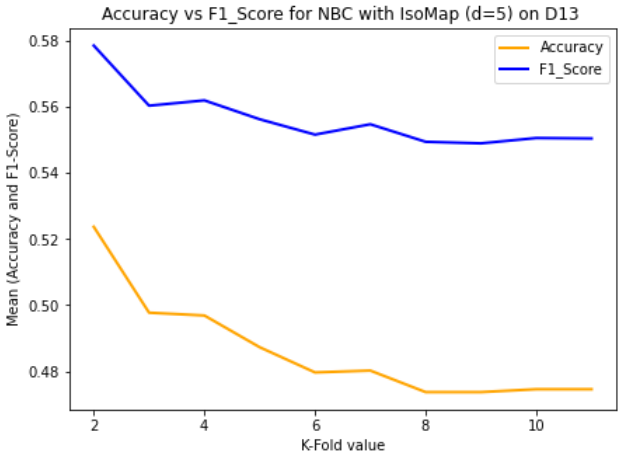

Accuracy vs F1_Score for NBC with IsoMap (d=100) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.460
0.455
0.450
0.445
0.440
0.435
2
4
6
8
10

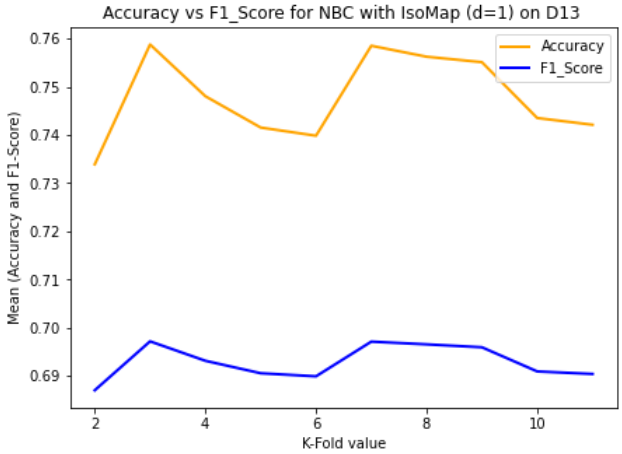

Accuracy vs F1_Score for NBC with IsoMap (d=150) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.46
0.45
0.44
0.43
0.42
0.41
2
4
6
8
10

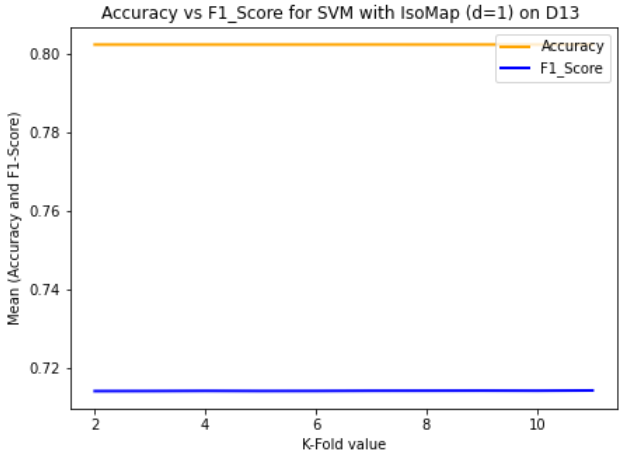

Accuracy vs F1_Score for NBC with IsoMap (d=250) on D1
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.43
0.42
0.41
0.40
0.39
0.38
0.37
2
4
6
8
10


Accuracy vs F1_Score for NBC with IsoMap (d=100) on D5 (75% dataset)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.410
0.405
0.400
0.395
0.390
0.385
2
4
6
8
10

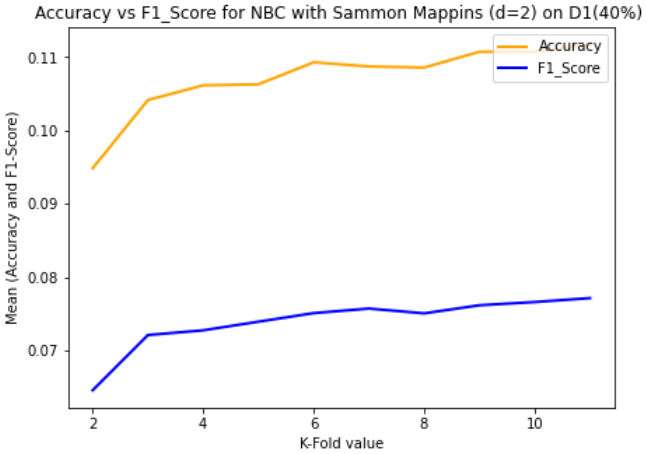

Accuracy vs F1_Score for NBC with IsoMap (d=2) on D13
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.64
0.63
0.62
0.61
0.60
0.59
0.58
0.57
2
4
6
8
10

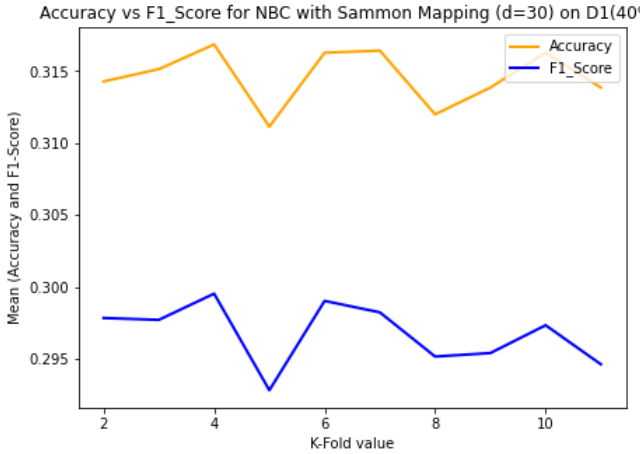

Accuracy vs F1_Score for NBC with IsoMap (d=5) on D13
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
K-Fold value
0.58
0.56
0.54
0.52
0.50
0.48
2
4
6
8
10

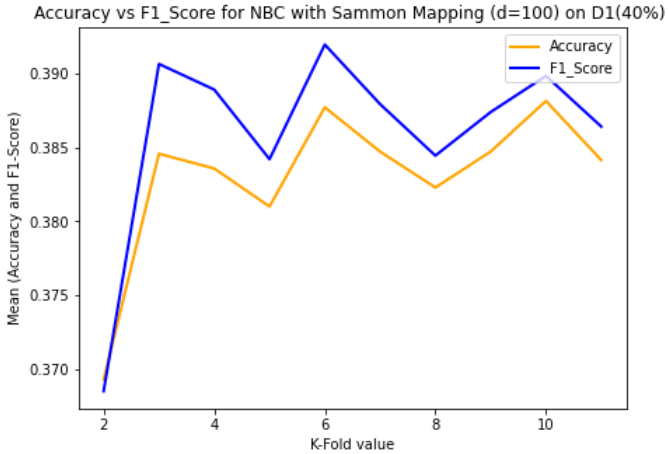


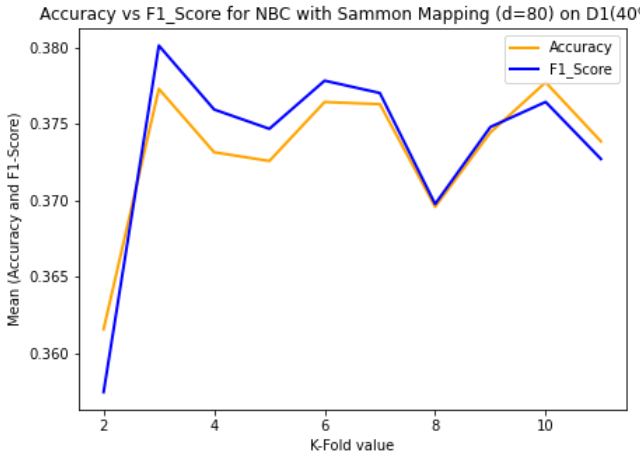


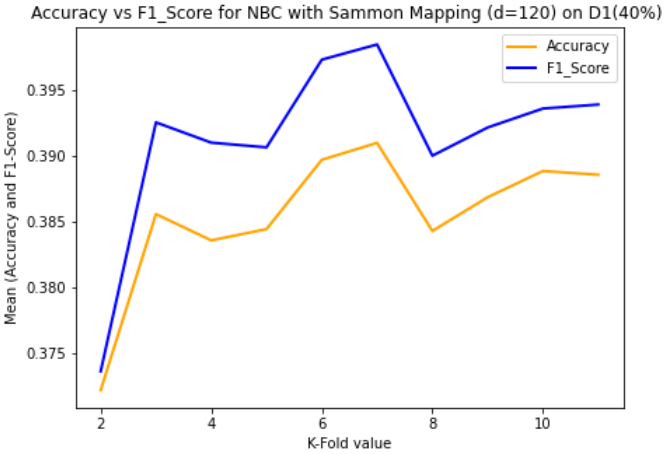
D. Sammon Mapping

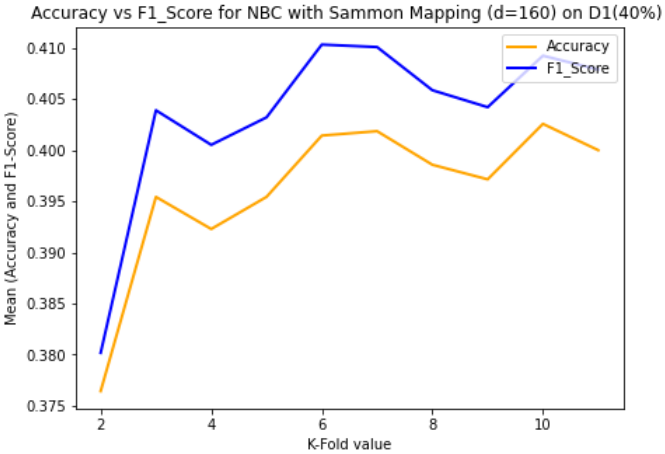



Accuracy vs F1_Score for NBC with Sammon Mapping (d=30) on D1(40%)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
0.315
0.310
0.305
0.300
0.295
2
4
6
8
10
K-Fold value


Accuracy vs F1_Score for NBC with Sammon Mapping (d=100) on D1(40%)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
0.390
0.385
0.380
0.375
0.370
2
4
6
8
10
K-Fold value


Accuracy vs F1_Score for NBC with Sammon Mapping (d=80) on D1(40%)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
0.380
0.375
0.370
0.365
0.360
2
4
6
8
10
K-Fold value


Accuracy vs F1_Score for NBC with Sammon Mapping (d=120) on D1(40%)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
0.395
0.390
0.385
0.380
0.375
2
4
6
8
10
K-Fold value


Accuracy vs F1_Score for NBC with Sammon Mapping (d=160) on D1(40%)
Accuracy
F1_Score
Mean (Accuracy and F1-Score)
0.410
0.405
0.400
0.395
0.390
0.385
0.380
0.375
2
4
6
8
10
K-Fold value

REFERENCES

[1] R.A. Fisher, "The Statistical Utilization of Multiple Measurements," Ann. Eugenics, vol. 8 pp. 376-386, 1938
[2] C.R. Rao, "The Utilization of Multiple Measurements in Problems of
Biological Classification," J. Royal Statistical Soc., B, vol. 10, pp. 159-203,
1948.
[3] S. Raschka, “Linear Discriminant Analysis,” sebastianraschka.com, 03-Aug-2014.
[4] J. W. Sammon, "A Nonlinear Mapping for Data Structure Analysis," in IEEE Transactions on Computers, vol. C-18, no. 5, pp. 401-409, May 1969, doi: 10.1109/T-C.1969.222678.
[5] J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” Science, vol. 290, no. 5500, pp. 2319–2323, 2000.
[6] O. Samko, A.D. Marshall, and P.L. Rosin, “Selection of the optimal parameter value for the Isomap algorithm,” Pattern Recognition Letters. 27. 968-979. 10.1016/j.patrec.2005.11.017.
[7] L. F. Terán Tamayo, Smartparticipation: A fuzzy-based recommender system for political community-building. Not Avail, 2014.
[8] Pollard, T. (2014, April 18). Sammon mapping in Python. computer software. Retrieved from https://github.com/tompollard/sammon.