Procedure and Results

- **Members (GR-G, 65)**
  - ➔ **Name: Surajit Kundu, Roll No: 21MM91R09**
  - ➔ **Name: Ankur Kumar Jaiswal, Roll No: 21MM62R08**

❖ Introduction

We have performed dimensionality reduction techniques like Principal Component Analysis(PCA) and Linear Discriminant Analysis(LDA) on the given Diabetes dataset. After dimensionality reduction, we have implemented an SVM classifier((sklearn.svm.SVC)) to the reduced dimensionality data generated from PCA and LDA.

We have used the Python programming language. Furthermore, the following libraries are used like NumPy, Pandas, matplotlib, sklearn, etc. We have calculated the accuracy on the validation set with different kernels like Linear, Polynomial, Sigmoid, and Gaussian radial basis functions.

We have downloaded the Diabetes dataset from https://www.kaggle.com/mathchi/diabetes-data-set. There are 768 rows and 9 columns in the dataset. Out of all columns, 8 columns are input attributes, and the last column i.e. ”Outcome” is the target value. And the other column names are ‘Pregnancies’, ‘Glucose’, ‘Blood pressure, ‘SkinThickness’, ‘Insulin’, ‘BMI’, ‘DiabetesPedigreeFunction’, ‘Age’. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

❖ Read the dataset

To read the Diabetes dataset, we have used the **read_csv()** function from the pandas’ library. We read the CSV file into the data frame format. In the dataset, there are 768 rows and nine columns, containing information regarding Diabetes, Digestive, and Kidney Diseases. The “Outcome” column is the target value that indicates whether the patient has diabetes or not, and other columns are input attributes of the given dataset. The output column has only two unique values, i.e., 0or 1. The value “1” implies that the patient has diabetes and “0” implies that the patient has no diabetes. There are a total of 500 zero’s and 268 ones in the output column.

❖ Data Preprocessing

We have dropped the “Outcome” column from the diabetes dataset using the **drop()** function from the panda’s data frame. And then, we have taken input feature attributes as X and target values as Y. Then, we scale the input data using **StandardScaler()** function from the sklearn library. StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance.

❖ Dimensionality Reduction

➢ Principal Component Analysis

A dimensionality-reduction method is often used to reduce the dimensionality of large data sets by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
PCA can be explained in five simple steps:

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along the axes of the principal component.

We have used the **PCA()** function from the sklearn library with 2 components. We have applied the dimensionality technique to the data stored in the X variable. We have used the **Matplotlib** library for the data visualization that we got from PCA.

➢ Linear Discriminant Analysis

It is also a dimensionality reduction technique. The goal of LDA is to discriminate different classes in low dimensional space by retaining the components containing feature values that have the best separation across classes. This can also be explained in six simple steps:

1. Compute mean vectors of each class of dependent variable.
2. Computers with-in-class and between-class scatter matrices.
3. Computes eigenvalues and eigenvector for SW(Scatter matrix within the class) and SB (scatter matrix between class).
4. Sorts the eigenvalues in descending order and selects the top k.
5. Creates a new matrix containing eigenvectors that map to the k eigenvalues.
6. Obtains the new features (i.e. linear discriminants) by taking the dot product of the data and the matrix.

We have imported **LDA()** function from the sklearn library with one component. We have applied the dimensionality technique to the data stored in the X variable. We have also used the fit_transform() function to scale the data and to learn the scaling parameters of the data. We have used the **Matplotlib** library for the data visualization that we got from LDA.

❖ Splitting the dataset

For splitting the diabetes dataset, we have to import the **train_test_split()** function from the scikit learn library. With the help of this function, we have split the dataset into a 70:10:20 ratio keeping the random state 42. We have used this function two times. Firstly, we have split the dataset into 70:30 where 70 % is given to the training dataset. Secondly, we have split the remaining 30 % into a

10:20 ratio where 10 % is given to the validation dataset and the rest to the test dataset.

❖ Support Vector Machine Classifier

We have imported **SVC()** classifier from the sklearn library to train the model. Support Vector Machine(SVM) classifier is a supervised machine learning algorithm that can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.
We have trained the SVM classifier with the training dataset that we got after dimensionality reduction through PCA and LDA. SVM classifier contains different kernels like sigmoid, linear, polynomial, and Gaussian radial basis function. kernels take a low-dimensional input space and transform it into a higher-dimensional space.

❖ Accuracy

For calculating the accuracy, we have imported the **accuracy_score()** function from the sklearn library. We have calculated the accuracy score of the validation dataset with different kernels using this function.
When we have trained the SVM classifier with PCA output. We have used many kernels functions like linear, polynomial, sigmoid, radial basis function, etc. We have also calculated the validation accuracy of different kernels by varying the hyperparameters of the classifier. The hyperparameters of the classifier that we have used are gamma and C. Out of all kernels, We have the **highest validation accuracy** of 70.12 % with the **linear kernel**. And the **test accuracy** with the **linear kernel** is 70.12 %.
When we have trained the SVM classifier with LDA output. We have used many kernels functions like linear, polynomial, sigmoid, radial basis function, etc. We have also calculated the validation accuracy of different kernels by varying the hyperparameters of the classifier. The hyperparameters of the classifier that we

have used are gamma and C. Out of all kernels, We have the **highest validation accuracy** of 79.22 % with the **linear kernel**. And the **test accuracy** with the **linear kernel** is 76.62 %.

❖ Data Visualization

For visualizing, we have used the **matplotlib** library. After reducing the dataset with PCA and LDA techniques and splitting the dataset, we have plotted the training dataset using this function.

❖ Difference between the results of PCA and LDA

After the test accuracy results that we got from the SVM classifier that was implemented on the reduced dimensionality of input data, we can conclude that the performance of dimensionality reduced LDA is better than dimensionality reduced PCA. The main reason might be the loss of information which is important for classification. And we also that LDA performance is better in a large dataset with multiple classes.

❖ Support vector Machine with Hyper-parameter tuning

For training the SVM classifier with hyperparameter tuning, we have imported **GridSearchCV()** function from the sklearn library. The classifier's hyperparameter that we have used is the kernel, C, and gamma. We have defined the parameter's range. We have trained the hyper-tunned SVM classifier with the training dataset. Then, we have printed the parameter that gives the best results after tuning. We have performed a grid search in both the PCA outputs and LDA outputs. In PCA output trained SVM, the parameters gamma = 1 and C = 0.1 is giving the highest validation accuracy with linear kernel. In LDA output trained SVM, the parameters gamma = 0.1 and C =100 is giving the highest validation accuracy with radial basis function kernel.