Machine Learning Project

Suryam Arnav Kalra

Procedure :

We have used dimensionality reduction techniques to reduce the feature dimension of the data and trained SVM classifiers on the reduced dimension feature space.

1) Importing Dataset

a) Imported the dataset from the diabetes.csv file provided
b) It contained 584 data samples, the data contained 8 features and 1 column for target values.
c) There were all real valued features.

2) Standardizing the Dataset

a) We standardized the data using the formula $X' = \frac{X - \mu}{\sigma}$, μ is the mean of feature values and σ is the standard deviation of the feature values.

3) Splitting the Dataset

a) We splitted the data randomly into 3 parts , 70% for training , 10% for validation and 20% for testing.

4) Performing PCA

a) For doing PCA we used sklearn.decomposition and imported PCA from it, in a function we passed the training data and generated a pca model using the function imported from sklearn.
b) We then gave our training data to it for fitting. After this we generated the principal components from our training dataset for further operation and also stored tha pca metrics generated here.
c) We stored the pca metrics generated to transform our validation and test data as mentioned in our assignment. We had to do pca only once and

generate pca metrics using that metrics only we had to transform our validation and test dataset.

d) The steps that the needed to perform PCA are:
   i) Compute the covariance matrix to identify correlations
   ii) Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
   iii) Create a feature vector to decide which principal components to keep.
   iv) Recast the data along the principal components axes.

5) Performing LDA

a) For doing LDA (linear discriminant analysis) we used sklearn.discriminant_analysis and imported LDA from it, in the function we passed the training data that was initially with us.
b) We then gave our 8 component training data with target values to it for fitting. After this we generated the 1 component from our 8 component training dataset for further operation and also stored the lda metrics generated here.
c) We stored the lda metrics generated to transform our validation and test data as mentioned in our assignment.
d) Steps that are needed to perform LDA are:
   i) Compute the d-dimensional mean vectors for all the different classes from the dataset.
   ii) Compute the scatter matrices . These matrices are computed in between class and within the class.
   iii) Compute the eigenvectors (e_1, e_2, ..., e_d) and corresponding eigenvalues (λ_1, λ_2, ..., λ_d) for the scatter matrices.
   iv) Sort the eigenvectors in the descending order of eigenvalues and choose k eigenvectors with the largest eigenvalues to make a d×k dimensional matrix say W
   v) Use this d×k eigenvector matrix to transform the samples onto the new subspace.

6) Performing SVM classification

a) Inputs for support vector machine classifier
   i) For PCA - We have a function called step_3_for_pca , from there we call the do_svm function to perform support vector machine classification we pass the 2 component training dataset generated after doing PCA, target values for this dataset, validation data, validation target values and the pca metrics generated from the PCA function.
   ii) For LDA - We have a function called step_3_for_lda this functions calls the do_svm function, since the lda metrics that was generated from the LDA function transforms 2 component data as mentioned in the assignment we first transform the validation data into 2 component data using pca metrics after this we pass the lda metrics and this 2 component validation data that will again we transformed to 1 component data using the lda metrics in the do_svm function.

b) Firstly we transform the validation data using the metrics passed in the function for PCA we transform the initial validation data into 2 component data for LDA we transform the 2 component data into 1 component data.

c) After this we perform support vector classification using 4 kernels, namely linear, polynomial, rbf and sigmoid.

d) For support vector classifier we have used the sklearn.svm and imported SVC from that

e) For a polynomial kernel we vary the degree and the coefficient after that we pass the name of the kernel as ‘poly’ along with the degree and the coefficient and use the training data and target values for fitting and compute accuracy using the results.

f) For a sigmoid kernel we vary only the coefficient after that we pass the name of the kernel as ‘sigmoid’ along with the coefficient and use the training data and target values for fitting and compute accuracy using the results.

g) For other kernels namely linear and rbf we only pass the name of the kernel to the SVC and use the training data and target values for fitting and compute accuracy using the results.
h) Using the above operations we calculate the best accuracy and the best kernel along with other best attributes associated with a particular kernel.
i) Support vector machines work by moving the data into relatively high dimensional space and finding a relatively high dimensional support vector classifier that can effectively classify the observations

7) Similarities between PCA and LDA

a) Both rank the new axes in order of importance
    i) PC1 (the first new axis that PCA creates) accounts for the most variation in the data , PC2 (the second new axis) does the second best job.
    ii) LD1 (the first new axis that LDA creates) accounts for the most variation between the categories, LD2 (the second new axis) does the second best job.
b) Both can let you dig in and see which components are driving the new axes
c) Linear Discriminant Analysis (LDA) is like PCA, but it focuses on maximizing the seperatibitly among the known categories.

Functions:

1) **import_data**: Imports the data from the given 'diabetes.csv' file and converts it to a numpy array with the elements being in float data type.
2) **label_encoding**: This function does the encoding of the categorical attributes in the format {a:1, b:2, ...} for each column which contains categorical values.
3) **one_hot_encoder**: This function does one hot encoding of the columns which contain categorical attributes.
4) **standardize**: This function is used to standardize the column values using the formula $X' = \frac{X - \mu}{\sigma}$
5) **get_train_test_validation_split**: This function randomly shuffles the data and does a 70:10:20 split to get the training, validation and testing data. The data is further divided into train_X, train_Y, test_X, test_Y, validate_X, validate_Y sets which denote respectively the training attributes, target values for training, testing attributes and target values for testing, validation attributes and target values for validation.
6) **get_plot_pca**: This function is used to plot the data using the 2 principal components on a 2-D graph in which the data points of the same class have the same color.
7) **get_plot_lda**: This function is used to plot the data using the 1 component obtained after doing LDA on a 2-D graph in which data points of the same class have the same color and the y-coordinate of each point is assumed to be 0.
8) **get_2pca_plot**: This function calls the get_plot_pca() function with the required parameters to get the plot.
9) **get_2lda_plot**: This function calls the get_plot_lda() function with the required parameters to get the plot.
10) **do_pca**: This function is used to do the principal component analysis with 2 components and returns the transformed training data and the projection matrix.
11) **get_accuracy**: This function is used to get the accuracy score of the predicted values against the target values.

12) **do_svm**: This function is used to implement support vector machines for different kernels with varying hyperparameters and returns the best kernel and hyperparameter for which the validation accuracy is the maximum.
13) **get_test_accuracy**: This function is used to get the accuracy on the test data by implementing a svm classifier on the best kernel and hyper parameters obtained.
14) **do_LDA**: This function is used to perform Linear Discriminant Analysis of the data and returns the transformed training data and the projection matrix.
15) **step_3_for_pca**: This function calls the do_svm() function with the data obtained after 2 component PCA and gets the best kernel and hyperparameters and then calls the get_test_accuracy() function to report the accuracy on the test data.
16) **step_3_for_lda**: This function calls the do_svm() function with the data obtained after 1 component LDA and gets the best kernel and hyperparameters and then calls the get_test_accuracy() function to report the accuracy on the test data.

Data Analysis and Cleaning:

- Since the missing values in the data were filled with 0 so we let the values be 0 there and each and every attribute had real floating point values.
- Since we were doing PCA and LDA it was necessary to do data standardization. Here's the formula that we used for normalization $X' = \frac{X - \mu}{\sigma}$

Results:

1) Analyze impact of varying kernel and hyperparameters of the SVM for 2-component PCA

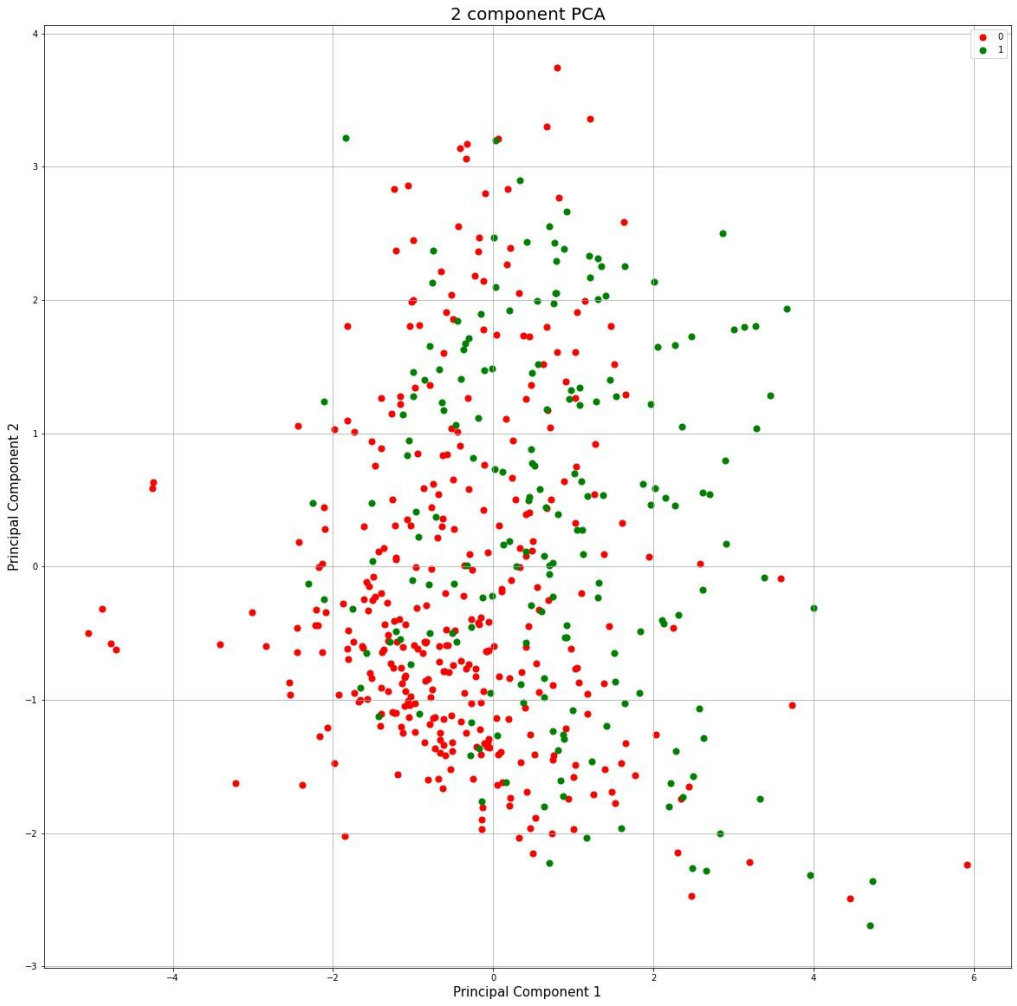

Fig a) Graph of data points obtained from 2-component PCA

2 component PCA			
Kernel	**Hyperparameters**		**Validation accuracy**
linear	NA		77.92207792207793
poly	Degree	Bias coefficient	
	1	1	77.92207792207793
	1	2	77.92207792207793
	1	3	77.92207792207793
	1	4	77.92207792207793
	2	1	81.81818181818183
	2	2	81.81818181818183
	2	3	81.81818181818183
	2	4	81.81818181818183
	3	1	77.92207792207793
	3	2	77.92207792207793
	3	3	77.92207792207793
	3	4	77.92207792207793
	4	1	81.81818181818183
	4	2	81.81818181818183
	4	3	81.81818181818183
	4	4	81.81818181818183
rbf	NA		80.51948051948052
sigmoid	Bias coefficient		
	1		70.12987012987013
	2		68.83116883116884
	3		67.53246753246754
	4		76.62337662337663

```
Best kernel = poly
Best validation accuracy = 81.81818181818183
Best degree = 2
Best bias coefficient = 1

Test accuracy = 72.72727272727273
```

2) Analyze impact of varying kernel and hyperparameters of the SVM for 1-component LDA

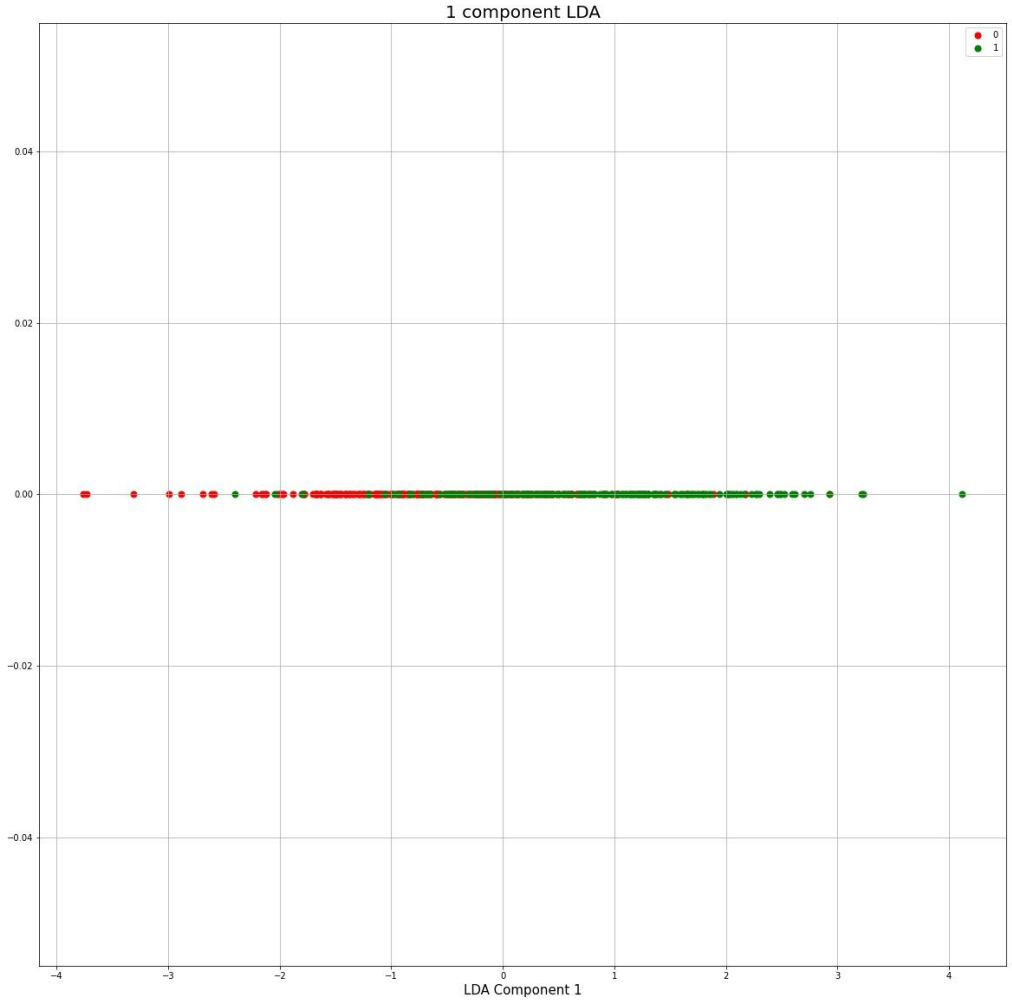

Fig b) Graph of data points obtained from 1-component LDA

1 component LDA			
Kernel	**Hyperparameters**		**Validation accuracy**
linear	NA		85.71428571428571
poly	Degree	Bias coefficient	
	1	1	85.71428571428571
	1	2	85.71428571428571
	1	3	85.71428571428571
	1	4	85.71428571428571
	2	1	84.4155844155844
	2	2	84.4155844155844
	2	3	84.4155844155844
	2	4	84.4155844155844
	3	1	84.4155844155844
	3	2	84.4155844155844
	3	3	84.4155844155844
	3	4	84.4155844155844
	4	1	84.4155844155844
	4	2	84.4155844155844
	4	3	84.4155844155844
	4	4	84.4155844155844
rbf	NA		84.4155844155844
sigmoid	Bias coefficient		
	1		80.51948051948052
	2		62.33766233766234

	3	59.74025974025974
	4	68.83116883116884

```
Best kernel = linear
Best validation accuracy = 85.71428571428571

Test accuracy =  76.62337662337663
```

3) Effect of 2-component PCA and 1-component LDA on the test accuracy

a) We can see that there are not many significant differences between the validation as well as the test accuracies obtained from the 2-component PCA and the 1-component LDA.

b) Majority of the time the accuracy shown by 1-component LDA is slightly higher or comparable to the accuracy shown by the 2-component PCA.

c) In terms of the best kernel, we can see some differences but majority of the time either poly or rbf kernel was the best for both of them.

d) These differences were expected due to the varying number of components in both the techniques.

e) The major difference is that PCA calculates the best discriminating components without foreknowledge about groups, whereas linear discriminant analysis calculates the best discriminating components for groups that are defined by the user.

f) So we can expect LDA to perform slightly better than PCA as can be observed on the validation and test accuracies.

g) We observed that increasing the number of components for both the PCA and the LDA led to better accuracies denoting that accuracy is somewhat proportional to the number of components used.

My own insights about the data

a) It is evident from the two graphs attached above, that we get a better splitting of data after using the 2-component PCA as compared to the 1-component LDA.
b) Most of the data after PCA is well separated along the two dimensions with some outliers in data.
c) Most of the data after LDA is well separated along the one dimension which has a high discriminating component for the groups of the data.
d) Since the data is not categorical in nature , we did not have to do any label encoding or one hot encoding , the codes of which are there in our main source file since at first we were given categorical data only.

Folder hierarchy:

1. main.py - This file contains all the functions necessary for this assignment.
2. requirements.txt - This contains all the packages required for this code to run
3. Report.pdf - This contains a report for this assignment.
4. 2_component_PCA.jpeg - 2 component PCA graph of data points
5. 1_component_LDA.jpeg - 1 component LDA graph of data points

How to run the code:

1. Download this directory into your local machine
2. Copy the ‘diabetes.csv’ file in the Source Code Directory
3. Ensure all the necessary dependencies with required version and latest version of Python3 are available (verify with requirements.txt)

```
pip3 install -r requirements.txt
```

4. Run the source code with the command

```
python3 main.py
```

5.

- **If you face any difficulty in installing packages, you can run the code on google collab.**