Pattern Recognition and Machine Learning

(Winter 2022)

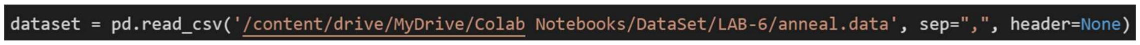
Assignment 6: Dimensionality Reduction

Report

Question - 1:

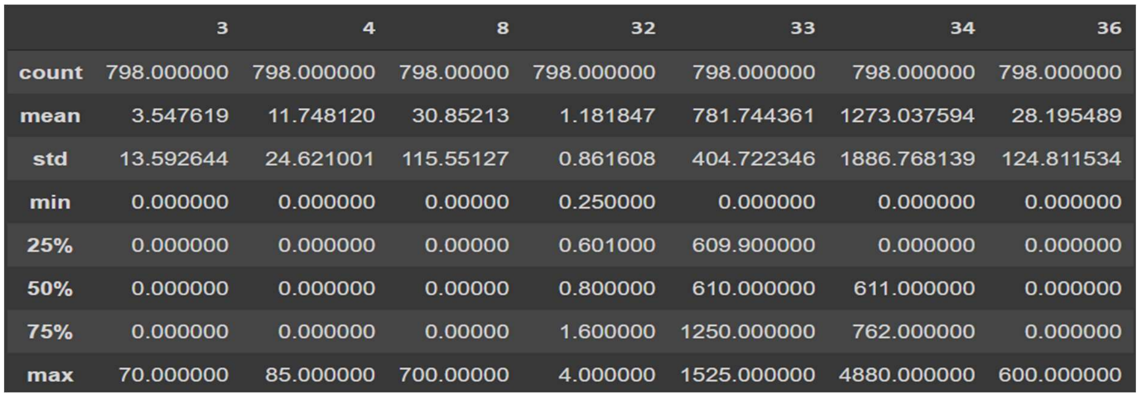
Task - 1: From the given link, download "anneal.data", "anneal.names" and "anneal.test", convert them into a readable format (Ex: txt, csv, etc....) and do meaningful Exploratory Data Analysis.

For this task, since we have to convert the "anneal.data" into a readable format. I loaded the given "anneal.data" file using pd.read_csv() function.



```
dataset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/DataSet/LAB-6/anneal.data', sep=",", header=None)
```

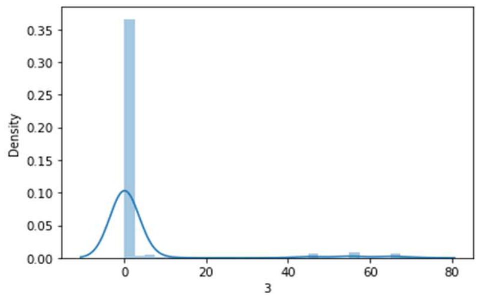
For, the exploratory data analysis of the data. I first checked the information of continuous columns in the given dataset.



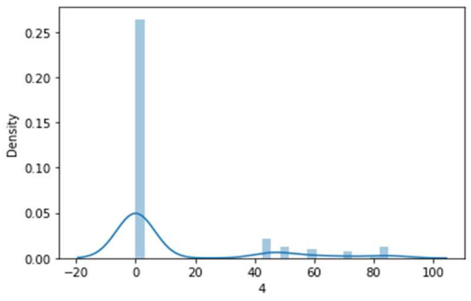
	3	4	8	32	33	34	36
count	798.000000	798.000000	798.00000	798.000000	798.000000	798.000000	798.000000
mean	3.547619	11.748120	30.85213	1.181847	781.744361	1273.037594	28.195489
std	13.592644	24.621001	115.55127	0.861608	404.722346	1886.768139	124.811534
min	0.000000	0.000000	0.00000	0.250000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.00000	0.601000	609.900000	0.000000	0.000000
50%	0.000000	0.000000	0.00000	0.800000	610.000000	611.000000	0.000000
75%	0.000000	0.000000	0.00000	1.600000	1250.000000	762.000000	0.000000
max	70.000000	85.000000	700.00000	4.000000	1525.000000	4880.000000	600.000000

I plotted the graphs for visualizing the distribution of the continuous columns present in the given dataset.

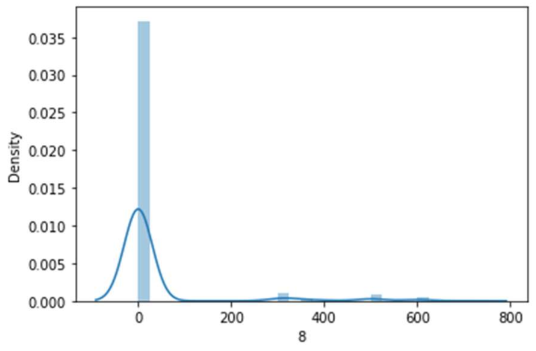
Plot: Density v/s feature '3'



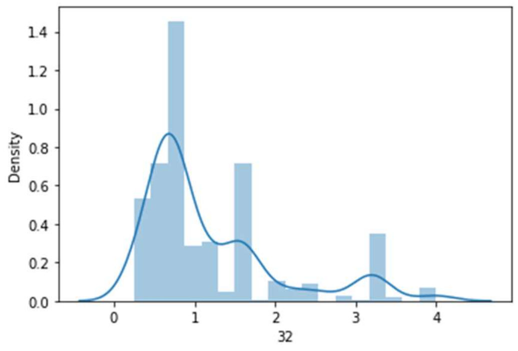
Plot: Density v/s feature '4'



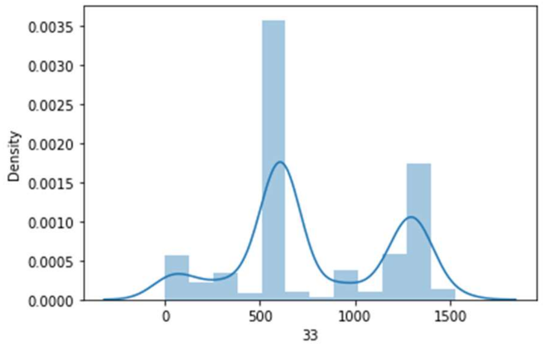
Plot: Density v/s feature ‘8’



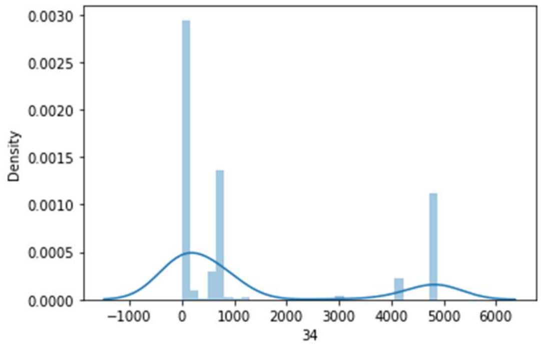
Plot: Density v/s feature ‘32’



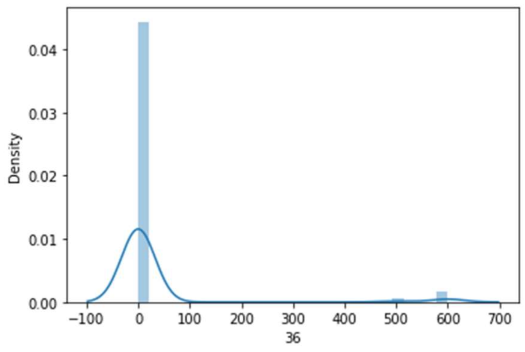
Plot: Density v/s feature ‘33’



Plot: Density v/s feature ‘34’

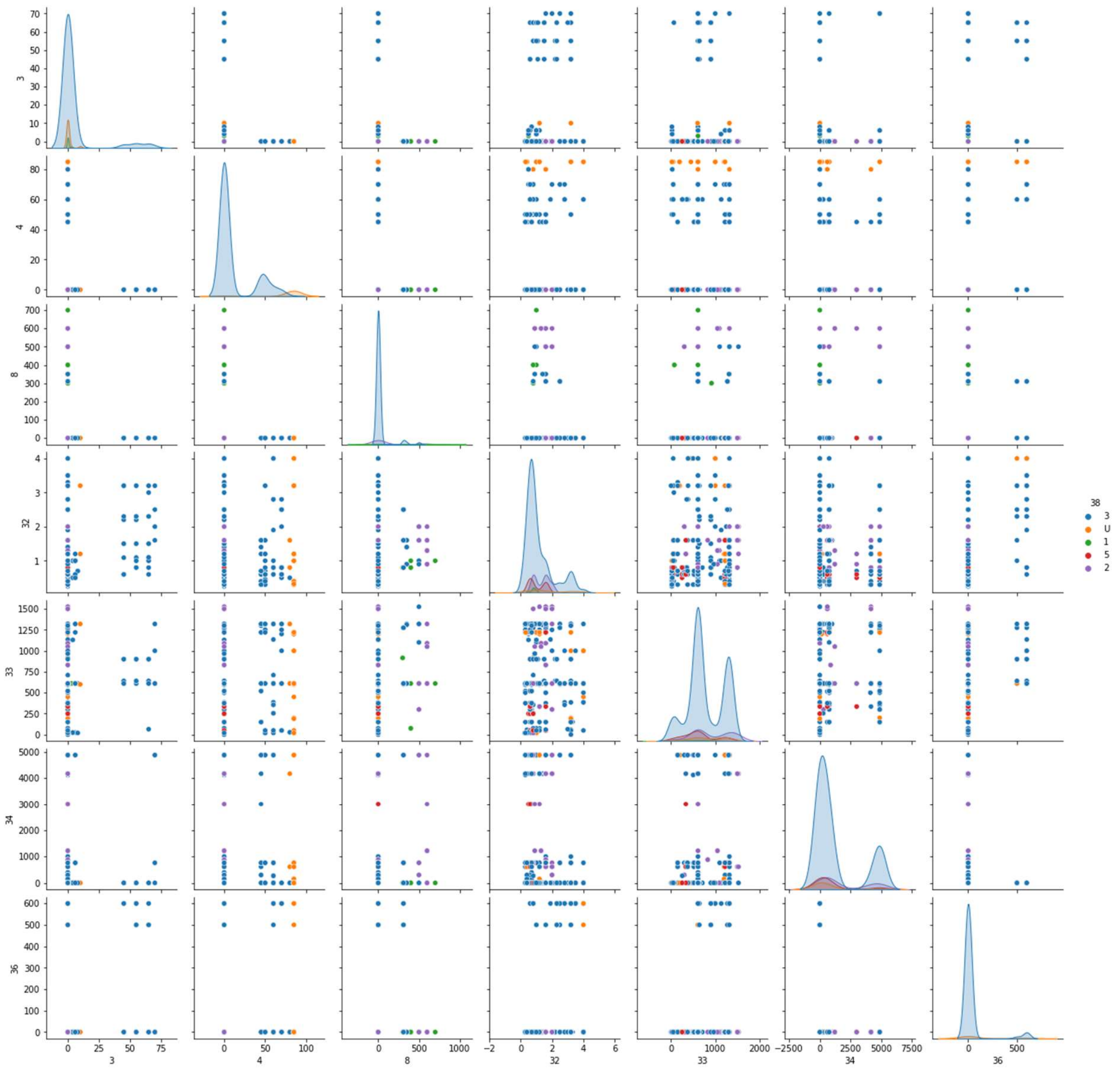


Plot: Density v/s feature ‘36’



I also plotted the pair plot, which helps visualize the pairwise relationships in a dataset.

Finally, I also took the input of the “anneal.test” dataset using the same pd.read_csv() function for the final testing.



<u>Task - 2:</u> Pre-process the data (If any discrepancies/errors, handle them as well) and split the data into [65:35].

For this part, I first checked for those columns that contains missing values. So, I made a list that will store indexes of those columns in which missing values are greater than 600. I also added column '1' in this list because it contains only single unique value which has no significance. Then, I dropped all these columns from the dataset.

Then, I handled the case of columns that contains categorical data. So, for these I just applied OneHotEncoding to these columns. Hence, finally there are 31 feature columns present in the dataset. I also Label Encoded the target column of the dataset.

Then finally I split the data into 65:35 sets using the train_test_split function/method by setting train_size = 0.65. Finally, I got the training, validation, and testing sets for this dataset.

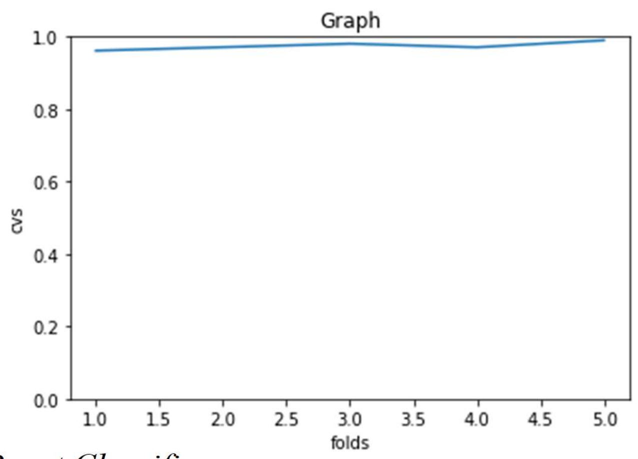
<u>**Task - 3:**</u> **Train 2-3 Classification Models (studied and implemented so far) with the proper reasoning of choosing them and show 5-Fold Cross-Validation Plots as well for comparison.**

For this task, I took 3 Classification Models i.e., Decision Tree Classifier, XGBoost Classifier and Gaussian Naïve Bayes Classifier. I chose these three models because this task is a classification task. So, I took these models as these are helpful in classifying the data.

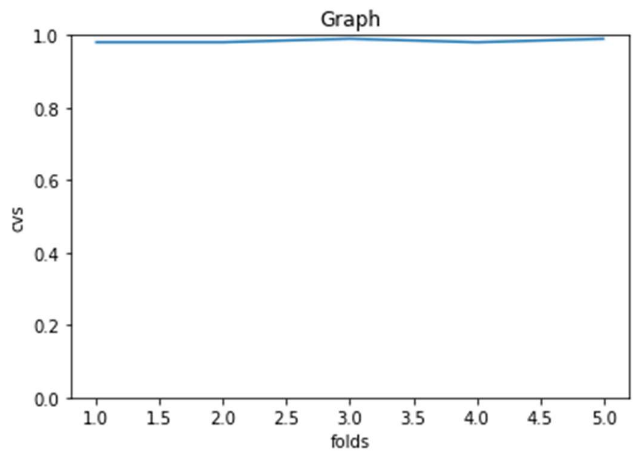
For each model, I instantiate the model first, then fitted the training set to the model and then calculated the accuracy of the validation set.

I also plotted the 5-Fold Cross Validation Plots for each model for comparison.

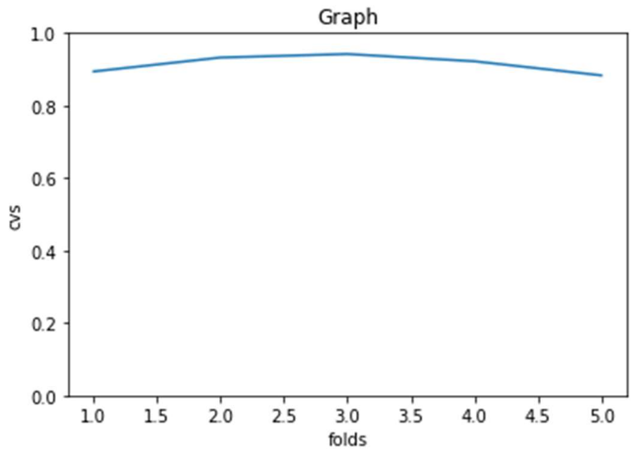
- For *Decision Tree Classifier*:



- For *XG Boost Classifier*:



- For *Gaussian Naïve Bayes Classifier*:



The accuracies and the cross-validation score that I got for each model are:

- For *Decision Tree Classifier*:

Accuracy: 99.28571428571429%

Cross-Validation Scores: [0.96153846 0.97115385 0.98076923 0.97087379 0.99029126]

Mean Cross-Validation Score: 97.49253174010455

- For *XGBoost Classifier*:

Accuracy: 98.57142857142858

Cross-Validation Scores: [0.98076923 0.98076923 0.99038462 0.98058252 0.99029126]

Mean Cross-Validation Score: 98.45593726661687

- For *Gaussian Naïve Bayes Classifier*:

Accuracy: 86.07142857142858

Cross-Validation Scores: [0.89423077 0.93269231 0.94230769 0.9223301 0.88349515]

Mean Cross-Validation Score: 91.50112023898431

Task - 4: Implement Principal Component Analysis from scratch, with sub-tasks as following:-

a. **Centralize the Data via feature-wise means and standard deviations. Write the code for deriving the covariance matrix from scratch.**

For this task, I first calculated the mean and variance of each of the columns present in the dataset. Then I centralized the Data as:

Data = (Data-mean)/(standard deviation)

Then, for calculating the covariance matrix from scratch. I implemented the formula for the covariance matrix i.e.,

Covariance matrix = (xTx)/(N-1)

Hence, I was able to calculate the covariance matrix of size feature*feature.

b. **Make a function Singular_Value_Decomp from scratch in order to compute Eigenvectors, Eigenvalues and Principal Components.**

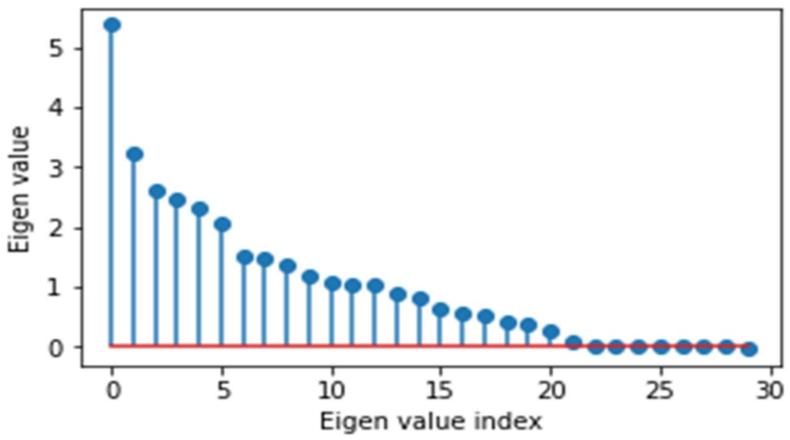
For this task, I made a function Singular_Value_Decomp that takes in parameter the covariance matrix and returns the eigenvalues and eigenvectors for that matrix.

I wrote the entire function form scratch in order to calculate the eigenvalues and eigenvectors without using the np.linalg functions. I used the concepts for calculating the eigenvalues and eigenvectors of a matrix to implement the function from scratch. I applied the principle of QR decomposition technique inorder to calculate the eigenvalues and eigenvectors. Hence, the implemented function was able to compute the eigenvectors, eigenvalues and principal components.

Task - 5: Use the above-made PCA to reduce the data upto a chosen dimension/principal-components and train 2-3 chosen classification models alongside 5-Fold Cross-Validation Plots.

For this task, I first calculated the eigenvalues and eigenvectors of the covariance matrix the using the implemented function "Singular_Value_Decomp". Then, I sorted the eigenvalues and eigenvectors. I also made a dictionary that takes eigen values as keys and their corresponding eigenvectors as values.

I also plotted a graph that represents the eigenvalues of the features of the dataset.



Hence, observing the graph and computing the variance. I decided to take the first 16 principal components out of the initial 31. Hence, I projected the centralized dataset onto the first 16 principal components with help of their eigenvectors. Finally, I was able to get the reduced dimensionality dataset after applying the PCA concept.

Hence, now for testing the dataset, I trained 3 Classification models that I took earlier (i.e., before applying PCA) and fitted the training dataset to it.

So, I trained the decision tree classifier, xgboost classifier and gaussian naïve bayes classifier with the reduced dimensionality dataset.

Then, I predicted the target values and computed the accuracy of each model on the training dataset.

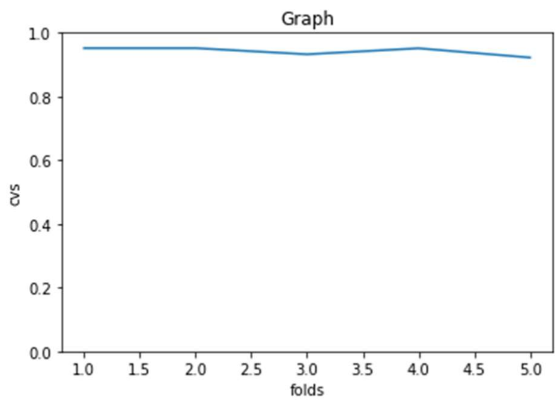
- For *Decision Tree Classifier*:

Accuracy: 95.0%

Cross-Validation Scores: [0.95192308 0.95192308 0.93269231 0.95145631 0.9223301]

Mean Cross-Validation Score: 94.20649738610905%

Graph:



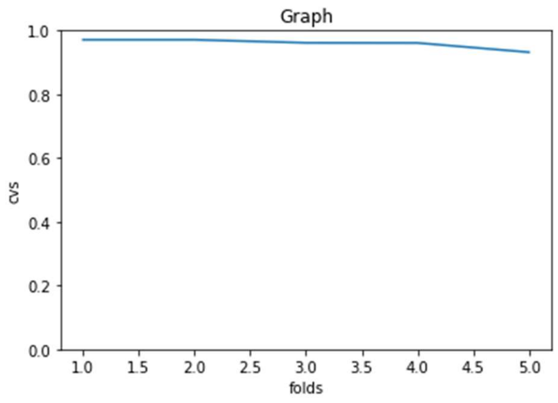
- For *XGBoost Classifier*:

Accuracy: 97.0%

Cross-Validation Scores: [0.97115385 0.97115385 0.96153846 0.96116505 0.93203883]

Mean Cross-Validation Score: 95.94100074682599

Graph:



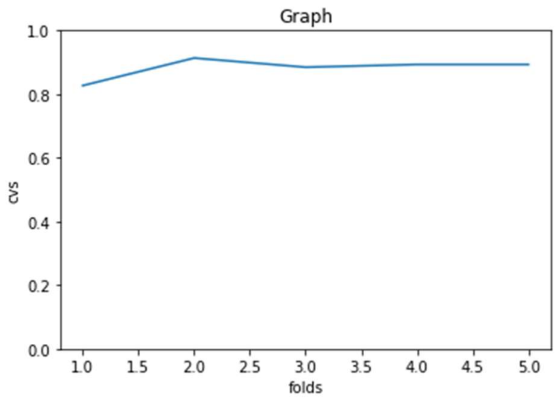
- For *Gaussian Naïve Bayes Classifier*:

Accuracy: 90.0%

Cross-Validation Scores: [0.82692308 0.91346154 0.88461538 0.89320388 0.89320388]

Mean Cross-Validation Score: 88.22815533980582%

Graph:



Task - 6: Show the Test results of Classification Models on both types of datasets (Before and After PCA), via 2-3 Evaluation Metrics of choice (Ex:- Accuracy, Sensitivity, F1-Score, etc.) with the proper Reasonings.

For this task, I took 2 evaluation metrics i.e., accuracy and classification report for evaluating the models. I took these two as evaluation metrics because accuracy_score in multilabel classification computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true. Hence, helps evaluating the model. The next is classification report it Builds a text report showing the main classification metrics. It contains f1_score, precision and recall in single report class wise. Hence, I took these two as my evaluation metrics and evaluated the model before PCA and after implementing PCA.

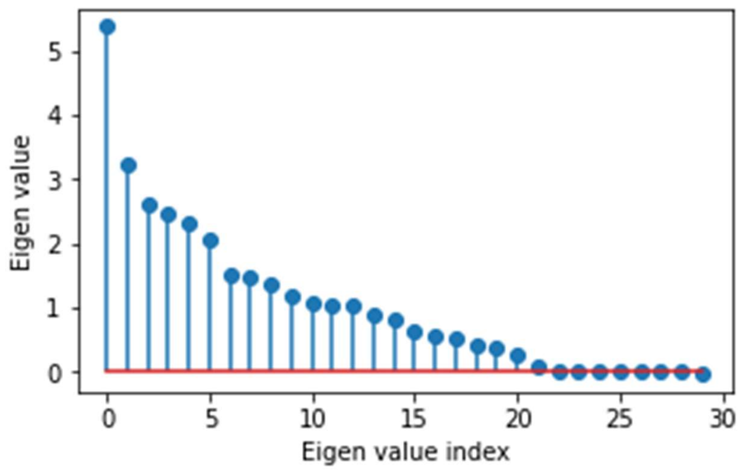
I have shown all these results in my collab file.

Task - 7: Were any changes observed before and after implementing PCA, with respect to the distribution of the dataset? Also, make any suitable graph through which the optimal number of principal components can be decided for optimal results.

Yes, there were changes observed before and after implementing PCA, with respect to the distribution of the dataset. After implementing PCA the data was approximately evenly (normal) distributed. I have also plotted the distribution of principal components for the reduced dimensionality dataset (after PCA) to observe the dataset distribution in my code (collab) file.

Also, to determine the optimal number of principal components. I made a graph that plots eigenvalues v/s eigen index, I even calculated principal components based upon the percentage of variance.

I got the graph of eigenvalues as:



Question - 2:

- **Data Pre-Processing.**

I first took the input of the given dataset. Then I performed the data pre-processing on the given dataset similar to the question 1, but this time I applied the Label Encode to encode the categorical columns of the dataset. Finally, I got the training and testing sets for this dataset.

Task - 1: Implement Linear Discriminant Analysis from scratch with the following subtasks:-

a. A function for computing within class and between class scatter matrices.

For this task, I first computed the mean vectors for each class. Now for computing the *within class scatter matrix*, I first created a matrix of size train.shape[1]*train.shape[1] and initialized it with zeroes. Then, I just implemented the formula of computing the within class scatter matrix for 'c' classes i.e.,

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

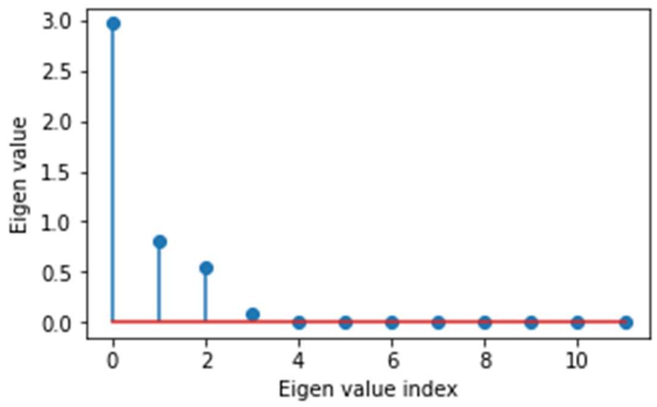
Now, for computing the *between class scatter matrix*, I first created a matrix of size train.shape[1]*train.shape[1] and initialized it with zeroes. Then, I just implemented the formula of computing the between class scatter matrix for 'c' classes i.e.,

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

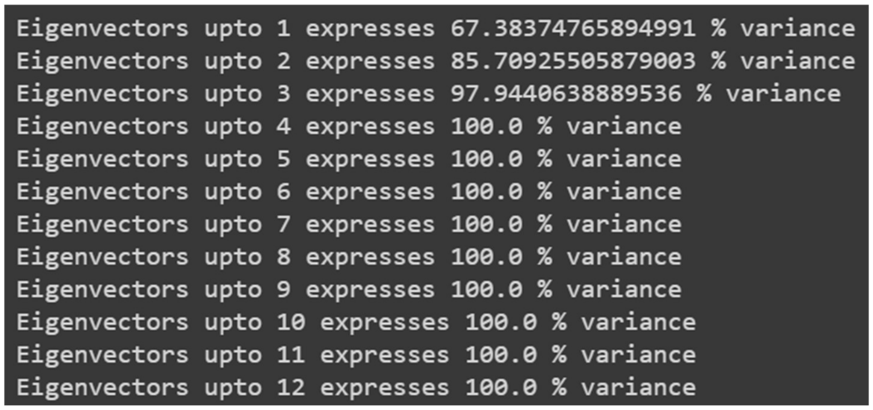
$$\text{where } \mu = \frac{1}{N}\sum_{\forall x} x = \frac{1}{N}\sum_{x \in \omega_i} N_i \mu_i$$

b. A function that will automatically select the number of linear discriminants based upon the percentage of variance that needs to be conserved

For this task, I computed first (SW^-1)(SB). Then, I computed the eigenvalues and eigenvectors of this (SW^-1)(SB) matrix. Then, I sorted the eigenvalues and eigenvectors. I also made a dictionary that takes eigen values as keys and their corresponding eigenvectors as values. I plotted the graph for eigenvalues vs eigen index. The plot that I got is:



Then, for determining the number of linear discriminants based upon the percentage of variance that needs to be conserved. I calculated the variance and selected 3 linear discriminants based upon the percentage of variance. I got the variance as:



```
Eigenvectors upto 1 expresses 67.38374765894991 % variance
Eigenvectors upto 2 expresses 85.70925505879003 % variance
Eigenvectors upto 3 expresses 97.9440638889536 % variance
Eigenvectors upto 4 expresses 100.0 % variance
Eigenvectors upto 5 expresses 100.0 % variance
Eigenvectors upto 6 expresses 100.0 % variance
Eigenvectors upto 7 expresses 100.0 % variance
Eigenvectors upto 8 expresses 100.0 % variance
Eigenvectors upto 9 expresses 100.0 % variance
Eigenvectors upto 10 expresses 100.0 % variance
Eigenvectors upto 11 expresses 100.0 % variance
Eigenvectors upto 12 expresses 100.0 % variance
```

Since, up to 3 linear discriminants the variance was close to 97%. So, I just took 3 linear discriminants.

Task - 2: Perform PCA and compare the results with LDA.

For this task, I trained the 3 classification models that I trained for PCA in the previous question and computed the accuracies for each of the model (i.e., Decision Tree Classifier, XGBoost Classifier, Gaussian Naïve Bayes Classifier).

On observing the accuracies that I got with LDA feature extraction method were much better as compared to the PCA implementation of feature extraction method.

Hence, I can say that the LDA feature extraction method worked well on this dataset as compared to the PCA feature extraction method.

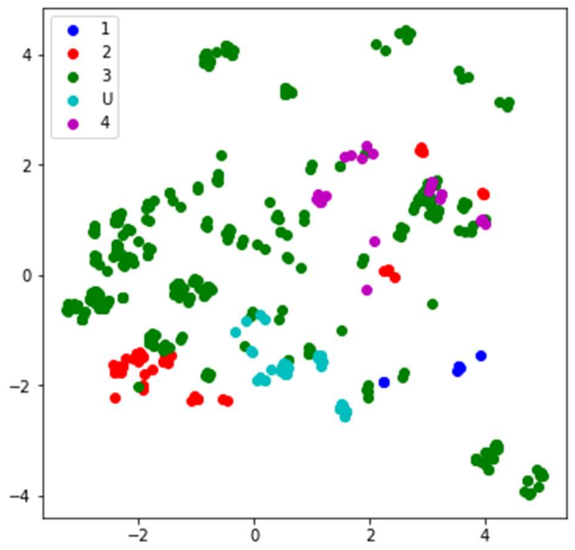
Also, we can say that this dataset model the difference between the classes of data. Hence, LDA works better on this dataset as compared to PCA.

Task - 3: Identify features having a high impact on classification tasks using both PCA and LDA and visualize the sample space using the first two principal components and first two linear discriminants and comment your observations.

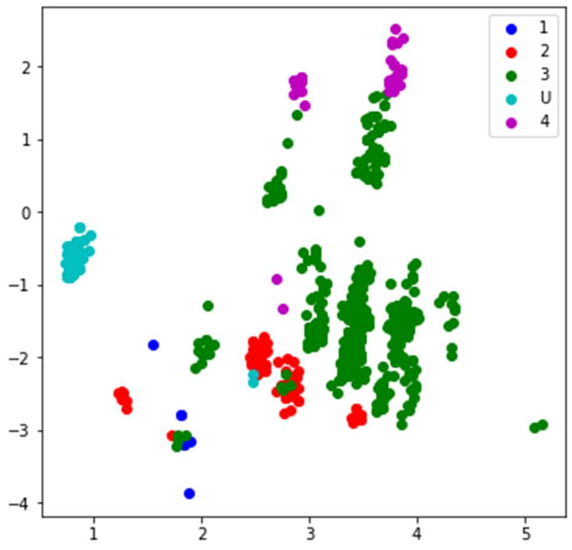
For this task, I returned the features that were having a high impact on classification task using both PCA and LDA.

For visualization of the sample space using the first two principal components and first two linear discriminants. I plotted the scatter plots. The plots that I got are:

For the first two principal components:



For the first two linear discriminants

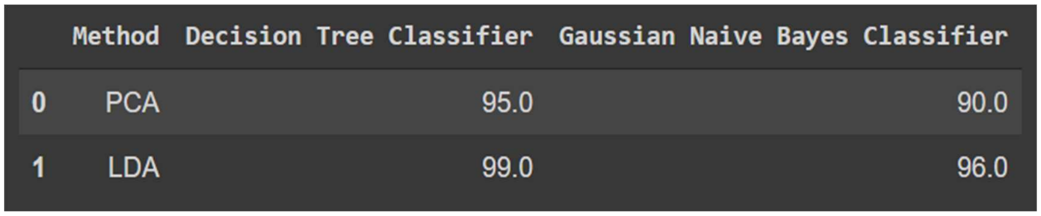


<u>Task - 4:</u> Using any 2 classification techniques make a 2 * 2 table with column headers as classification techniques used and row headers as feature extraction methods used. The values inside the table should be the accuracy achieved in each case. Compare the results of the table.

For this task, I took the two classification models as Decision Tree Classifier and Gaussian Naïve Bayes Classifier and made a 2*2 table with column headers as classification techniques used and row headers as feature extraction methods used.

The values that I inserted in this table is the accuracies that I achieved in each classification technique with each of the feature extraction methos used in this assignment.

The table that I got for this task is:

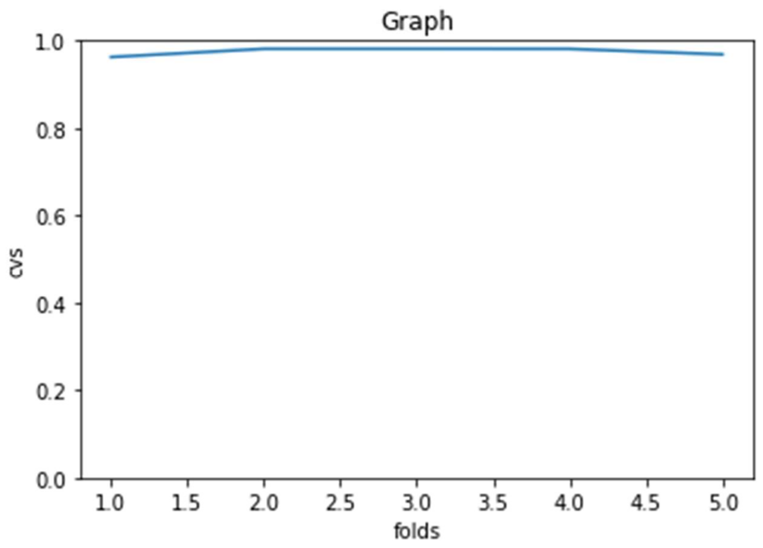


	Method	Decision Tree Classifier	Gaussian Naive Bayes Classifier
0	PCA	95.0	90.0
1	LDA	99.0	96.0

<u>Task - 5:</u> Using LDA as a classifier, perform 5-fold cross-validation and plot ROC and compute AUC for each fold from scratch.

For this task, I used LDA as classifier and performed 5-fold cross validation. I also plotted the cross-validation graph between cross validation scores and folds.

The curve that I got for 5-fold cross validation is:



Then, I computed predict probability using the testing dataset, inorder to compute the auc_roc_score. Hence, I got the auc_roc_score as 0.7070395800853001.