Name	**Sujal Sandeep Dingankar**
UID no.	**2024301005**
Experiment No.	**7**

AIM:	**Feature Reduction and Overfitting/Underfitting**
Program 1	
PROBLEM STATEMENT :	You are given a dataset with 10,000 records and 100 features. Your task is to perform feature extraction using PCA, SVD, and LDA after handling missing values and scaling the data. Steps to Perform: Data Preprocessing: Handling Missing Values: Identify and fill in any missing values in the dataset using one of the following techniques: Mean imputation Median imputation Dropping rows/columns Scaling the Data: Scale the dataset using StandardScaler from sklearn (or any other appropriate scaling method) to ensure all features are on the same scale. Principal Component Analysis (PCA): Apply PCA on the scaled dataset to reduce the dimensionality. Extract the top 5 principal components. Visualize: Plot the first two principal components in a 2D plot and explain how much variance is retained. Display: Show the explained variance ratio for the first few components and discuss the importance of each component. Singular Value Decomposition (SVD): Apply SVD to the scaled dataset to perform dimensionality reduction. Extract the singular values and right singular vectors. Visualize: Plot the singular values to show how they decay and how many are required to capture most of the variance.

	Linear Discriminant Analysis (LDA): Assume that you have a target label column (you can randomly generate a binary or multi-class label). Apply LDA to maximize class separability. Visualize: Plot the first two components derived from LDA and explain how it separates the data. Write a conclusion discussing the effectiveness of PCA, SVD, and LDA in dimensionality reduction and feature extraction. Also, discuss when to use each method and the pros and cons of each.

PROGRAM:	import numpy as np
	import pandas as pd
	import matplotlib.pyplot as plt
	import seaborn as sns
	from sklearn.preprocessing import StandardScaler
	from sklearn.decomposition import PCA, TruncatedSVD
	from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
	
	#Handling Missing Values using Mean Imputation Method:
	df = pd.read_csv('feature_extraction_10000x100 (1).csv')
	df.isnull().sum()
	df.isnull().values.any()
	df.fillna(df.mean(), inplace=True)
	print(df)
	
	#Scaling the data
	scaler = StandardScaler()
	scaled_data = scaler.fit_transform(df)
	#Converting the scaled data back to a Pandas DataFrame
	scaled_df = pd.DataFrame(scaled_data, columns=df.columns, index=df.index)
	display(scaled_df)
	
	#PCA
	pca = PCA(n_components=5)
	principalComponents = pca.fit_transform(scaled_data)
	principalDf = pd.DataFrame(data = principalComponents,columns = ['principal component 1', 'principal component 2', 'principal component 3', 'principal component 4', 'principal component 5'])

	plt.figure(figsize=(8, 6))
	plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'])
	plt.xlabel('Principal Component 1')
	plt.ylabel('Principal Component 2')
	plt.title('PCA: First Two Principal Components')
	plt.show()
	
	explained_variance_ratio = pca.explained_variance_ratio_
	#Bar plot of variance ratio
	plt.bar((range(explained_variance_ratio.shape[0])),explained_variance_ratio*100)
	
	plt.figure(figsize=(8,6))
	plt.plot(np.cumsum(pca.explained_variance_ratio_))
	plt.xlabel('Number of Components')
	plt.ylabel('Cumulative Explained Variance')
	plt.title('Explained Variance by PCA Components')
	plt.show()
	#Explained variance ratio
	print("Explained Variance Ratio:", explained_variance_ratio)
	#Cumulative explained variance
	cumulative_variance = np.cumsum(explained_variance_ratio)
	print("Cumulative Explained Variance:", cumulative_variance)
	
	#Apply SVD
	svd = TruncatedSVD(n_components=5)
	X_svd = svd.fit_transform(scaled_data)
	# Singular values and explained variance
	singular_values = svd.singular_values_
	# Plotting singular values
	plt.figure(figsize=(8,6))
	plt.plot(singular_values, marker='o')
	plt.title('Singular Values Decay (SVD)')
	plt.xlabel('Component')
	plt.ylabel('Singular Value')
	plt.show()
	
	#LDA
	y = np.random.randint(0, 3, size=(scaled_data.shape[0]))
	lda = LinearDiscriminantAnalysis(n_components=2)

```
X_lda = lda.fit_transform(scaled_data, y)
lda_df = pd.DataFrame(data=X_lda, columns=['LD1','LD2'])

# Plot LDA - First two components
plt.figure(figsize=(8,6))
plt.scatter(lda_df['LD1'], lda_df['LD2'], c=y, cmap='viridis', alpha=0.5)
plt.xlabel('First LDA Component')
plt.ylabel('Second LDA Component')
plt.title('2D LDA Projection')
plt.colorbar(label='Class')
plt.grid(True)
plt.show()
```

RESULT:

After handling missing values:

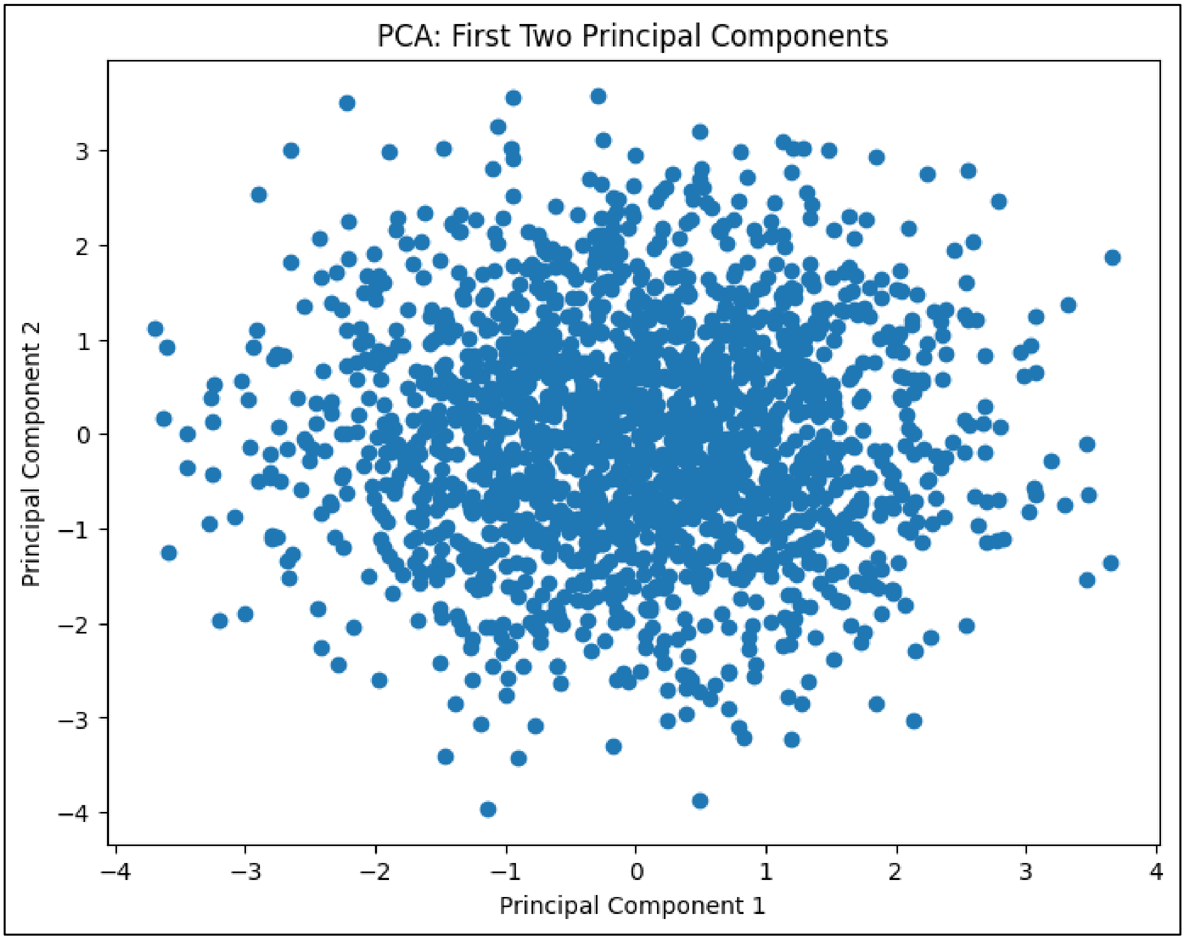
```
[4809 rows x 100 columns]
      Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  Feature_6  \
0      0.374540   0.950714   0.505553   0.598658   0.156019   0.155995
1      0.031429   0.498012   0.314356   0.508571   0.907566   0.249292
2      0.642032   0.084140   0.161629   0.898554   0.606429   0.009197
3      0.051682   0.531355   0.505553   0.637430   0.494933   0.503391
4      0.503356   0.902553   0.505252   0.826457   0.320050   0.895523
...         ...        ...        ...        ...        ...        ...
4804   0.038691   0.449889   0.505553   0.554887   0.753751   0.594528
4805   0.503356   0.924459   0.636846   0.446880   0.667462   0.703524
4806   0.955337   0.498012   0.975730   0.123995   0.191202   0.568255
4807   0.898904   0.578307   0.473285   0.498097   0.494933   0.907059
4808   0.592179   0.128293   0.205513   0.109997   0.768298   0.344372

      Feature_7  Feature_8  Feature_9  Feature_10  ...  Feature_91  \
0      0.058084   0.866176   0.601115    0.708073  ...    0.119594
1      0.410383   0.755551   0.228798    0.076980  ...    0.093103
2      0.101472   0.663502   0.005062    0.160808  ...    0.030500
3      0.516300   0.322956   0.495354    0.270832  ...    0.990505
4      0.389202   0.010838   0.905382    0.091287  ...    0.455657
...         ...        ...        ...         ...  ...         ...
4804   0.899872   0.605507   0.672089    0.603627  ...    0.217198
4805   0.854403   0.640505   0.495354    0.476905  ...    0.004997
4806   0.097628   0.808218   0.927500    0.845227  ...    0.978532
4807   0.888807   0.729333   0.242738    0.232541  ...    0.628932
4808   0.481381   0.142351   0.545908    0.166190  ...    0.496869
```

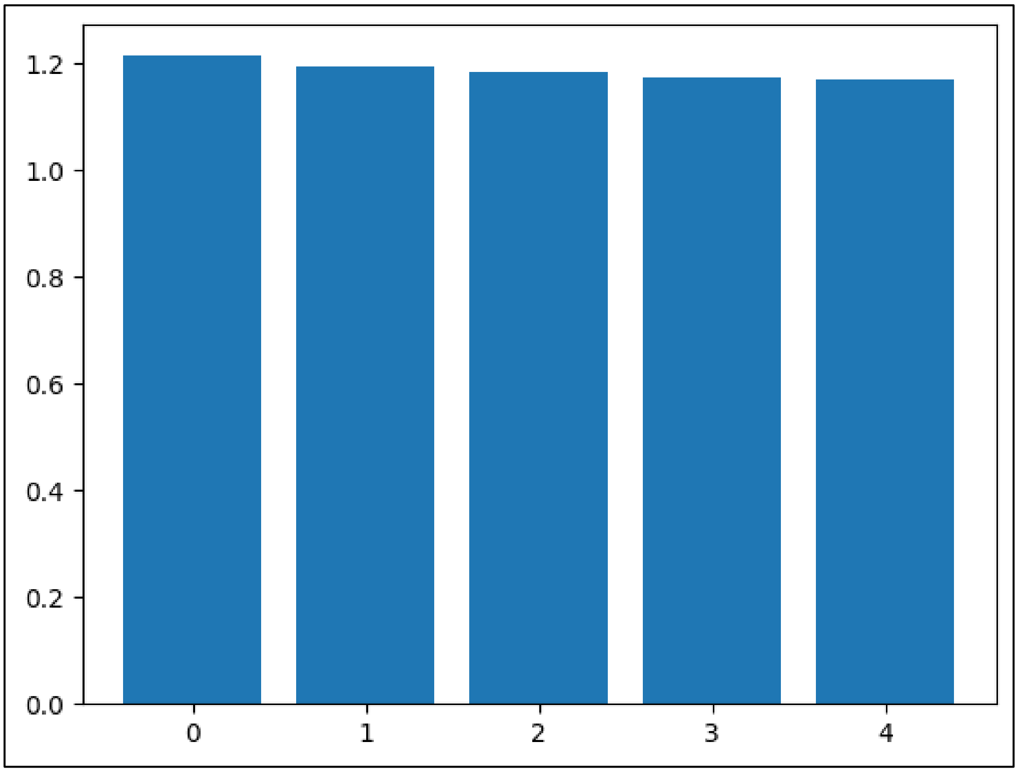
```
      Feature_92  Feature_93  Feature_94  Feature_95  Feature_96  Feature_97  \
0       0.713245    0.760785    0.561277    0.770967    0.493796    0.522733
1       0.897216    0.900418    0.633101    0.339030    0.503413    0.725956
2       0.037348    0.822601    0.360191    0.127061    0.522243    0.494640
3       0.412618    0.372018    0.776413    0.340804    0.930757    0.858413
4       0.620133    0.277381    0.188121    0.463698    0.353352    0.583656
...          ...         ...         ...         ...         ...         ...
4804    0.367487    0.277648    0.597489    0.098929    0.865825    0.511610
4805    0.496597    0.357395    0.656619    0.018497    0.407858    0.113797
4806    0.871272    0.927965    0.442182    0.094291    0.094392    0.043998
4807    0.238748    0.357297    0.262110    0.956766    0.363023    0.385133
4808    0.496597    0.500911    0.502117    0.492999    0.503413    0.494640

      Feature_98  Feature_99  Feature_100
0       0.427541    0.025419     0.107891
1       0.897110    0.887086     0.779876
2       0.215821    0.622890     0.500171
3       0.428994    0.750871     0.500171
4       0.077735    0.974395     0.986211
...          ...         ...          ...
4804    0.018809    0.768859     0.551069
4805    0.497196    0.648085     0.117194
4806    0.146290    0.527961     0.398020
4807    0.561084    0.521850     0.899141
4808    0.497196    0.499356     0.500171
```

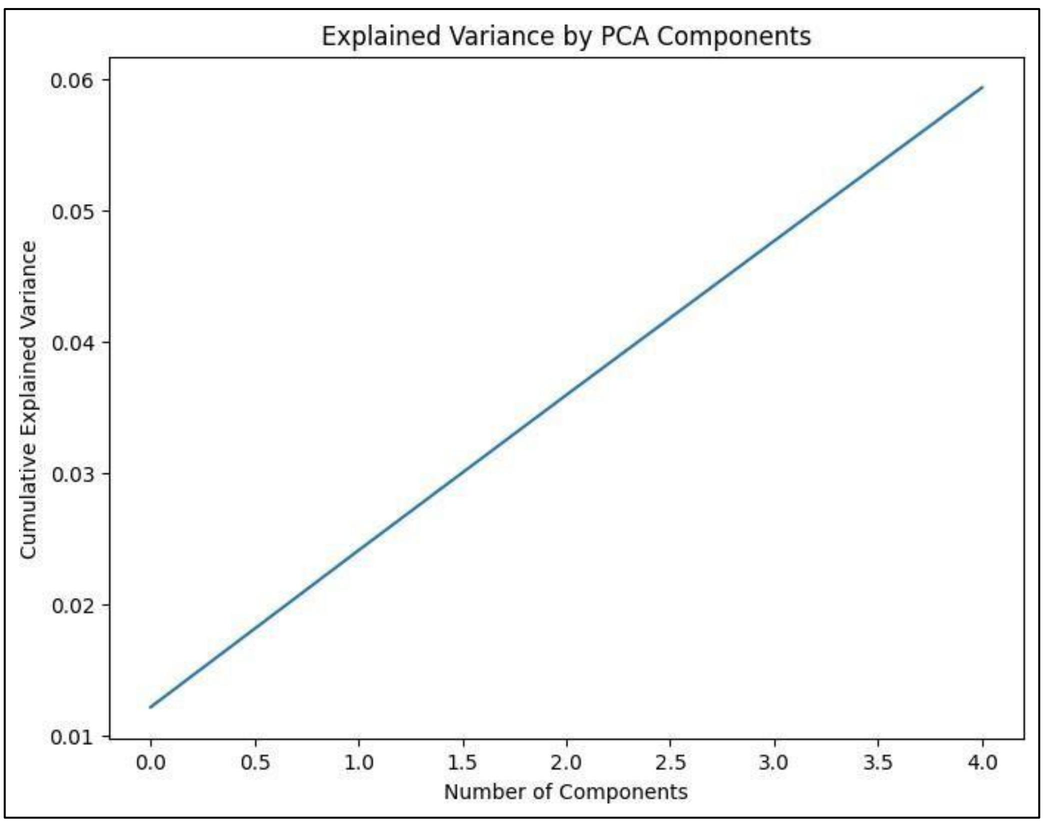
#After scaling the dataset

	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	Feature_6	Feature_7	Feature_8	Feature_9	Feature_10	...	Feature_91	Feature_92	Feature_93
0	-0.462263	1.664839	0.000000	3.260511e-01	-1.253236	-1.288662	-1.672282	1.314774	3.950666e-01	0.788637	...	-1.349826e+00	0.804814	0.889837
1	-1.707089	0.000000	-0.679697	1.114703e-03	1.481997	-0.940464	-0.352843	0.911398	-9.742386e-01	-1.498791	...	-1.445468e+00	1.468858	1.402919
2	0.508211	-1.491343	-1.238343	1.407741e+00	0.386018	-1.836529	-1.509785	0.575755	-1.797096e+00	-1.194951	...	-1.671482e+00	-1.634840	1.116978
3	-1.633612	0.137474	0.000000	4.658952e-01	0.000000	0.000000	0.043841	-0.665989	-2.041587e-16	-0.796163	...	1.794424e+00	-0.280302	-0.538690
4	0.000000	1.489428	0.018564	1.147696e+00	-0.656251	1.471349	-0.432172	-1.804079	1.514098e+00	-1.446935	...	-1.365394e-01	0.468724	-0.886434
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1798	0.143940	1.299922	1.114249	4.004448e-16	-1.365730	-0.498373	-1.280997	0.146248	-1.598514e+00	0.585994	...	-1.037509e+00	0.877053	0.672694
1799	-0.441328	-1.033846	-0.858257	1.299299e-01	-1.339144	1.103712	-1.238631	-0.515258	-1.261164e+00	1.308897	...	-1.504242e+00	0.000000	-0.732046
1800	-1.044375	0.940137	-0.763440	1.399622e+00	0.000000	-0.053296	0.000000	0.357088	3.702920e-01	0.302917	...	1.742895e-01	-0.216921	1.331910
1801	-0.660390	0.382149	-0.151342	1.154979e+00	0.000000	-1.486190	0.837187	0.562682	1.312314e+00	-0.742931	...	1.801612e+00	-0.997559	0.670351
1802	1.701447	0.000000	0.984855	-1.132655e+00	0.388211	0.655241	-0.225817	-0.631851	6.848540e-01	1.712648	...	-2.004119e-16	0.000000	0.000000

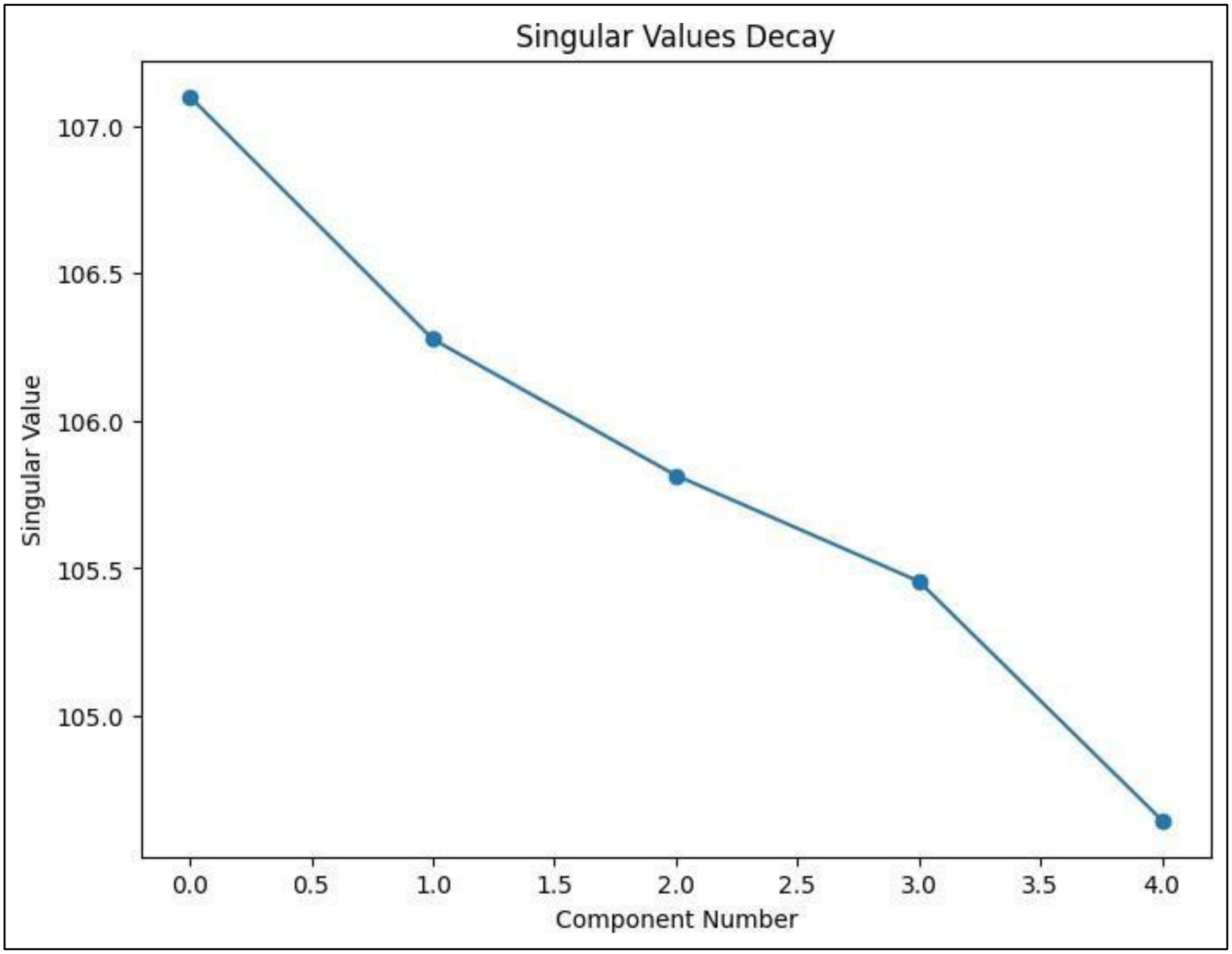
1803 rows × 100 columns

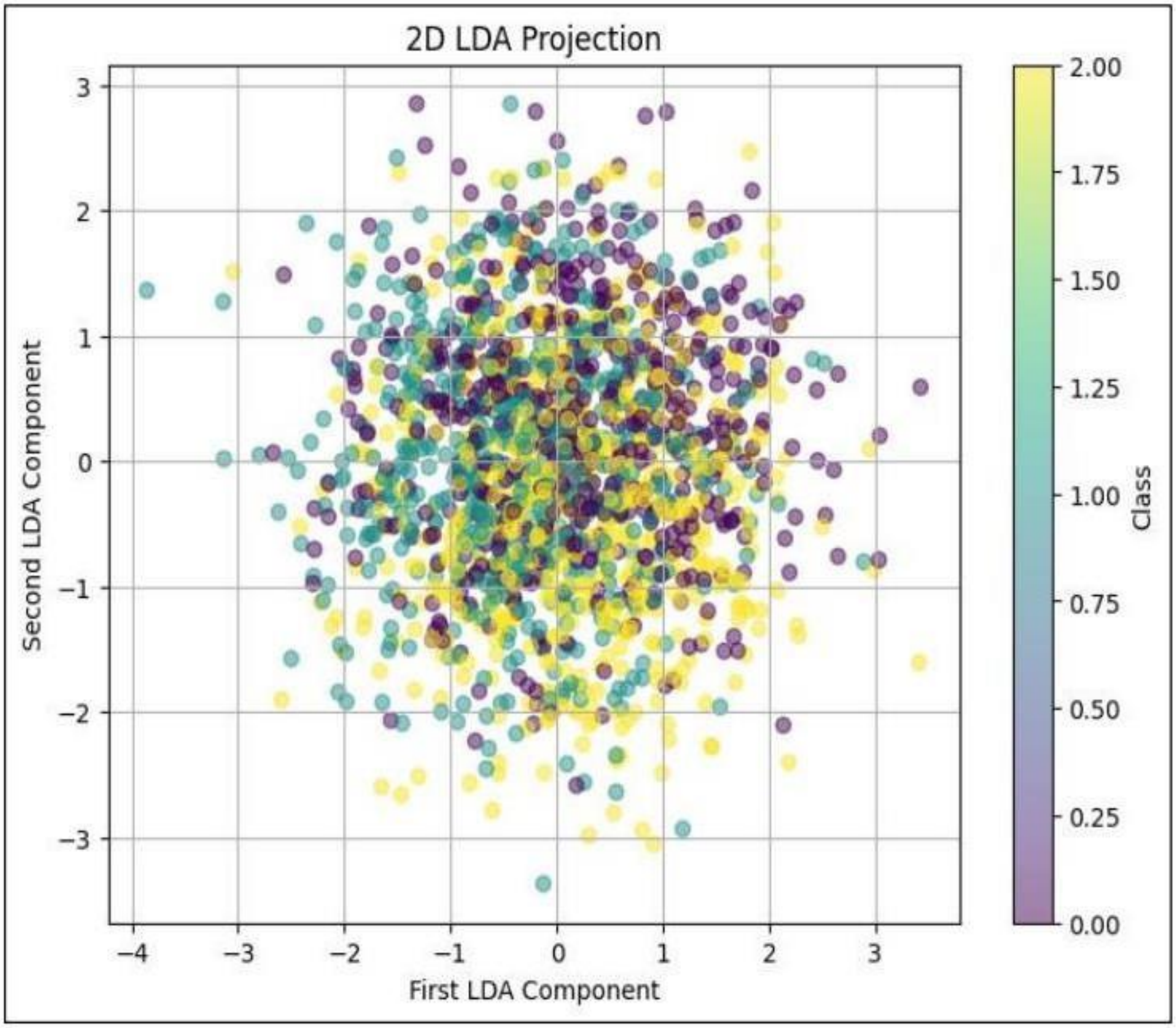

PCA: First Two Principal Components
Principal Component 1
Principal Component 2
−4
−3
−2
−1
0
1
2
3
4
3
2
1
0
−1
−2
−3
−4





Explained Variance Ratio: [0.01214242 0.01194801 0.01184605 0.01174268 0.01168956]
Cumulative Explained Variance: [0.01214242 0.02409043 0.03593648 0.04767916 0.05936872]


Singular Values Decay
107.0
106.5
106.0
105.5
105.0
Singular Value
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Component Number



CONCLUSION:	In this experiment, I learnt the concept of feature reduction and perform it with the help of PCA, SVD and LDA techniques. I was able to plot the graphs of each technique and understand the inferences derived from it. PCA captures the most variance in the data and makes datasets easier to visualize in lower dimensions. However, it is sensitive to outliers and noise in the data. SVD can capture the underlying structure of the data even when some information is missing or incomplete, but it is intensive for large datasets. LDA reduces dimensionality while preserving the features that are important for distinguishing between classes and it often improves the performance of algorithms. However, it is sensitive to overfitting if the dataset has very few samples.