

PCA_LDA_T-SNE_COMPARISON

February 19, 2025

```
[63]: data_url = 'https://raw.githubusercontent.com/pkmlong/  
↳Breast-Cancer-Wisconsin-Diagnostic-DataSet/master/data.csv'
```

```
[64]: import numpy as np  
import pandas as pd  
import seaborn as sns  
sns.set()  
import matplotlib.pyplot as plt
```

```
[65]: df = pd.read_csv(data_url)  
df.head()
```

```
[65]:      id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \  
0    842302      M      17.99      10.38      122.80      1001.0  \  
1    842517      M      20.57      17.77      132.90      1326.0  \  
2    84300903     M      19.69      21.25      130.00      1203.0  \  
3    84348301     M      11.42      20.38       77.58       386.1  \  
4    84358402     M      20.29      14.34      135.10      1297.0  \  
  
      smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \  
0          0.11840      0.27760          0.3001          0.14710  \  
1          0.08474      0.07864          0.0869          0.07017  \  
2          0.10960      0.15990          0.1974          0.12790  \  
3          0.14250      0.28390          0.2414          0.10520  \  
4          0.10030      0.13280          0.1980          0.10430  \  
  
      ... texture_worst  perimeter_worst  area_worst  smoothness_worst  \  
0      ...      17.33      184.60      2019.0      0.1622  \  
1      ...      23.41      158.80      1956.0      0.1238  \  
2      ...      25.53      152.50      1709.0      0.1444  \  
3      ...      26.50      98.87      567.7      0.2098  \  
4      ...      16.67      152.20      1575.0      0.1374  \  
  
      compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \  
0          0.6656      0.7119          0.2654          0.4601  \  
1          0.1866      0.2416          0.1860          0.2750  \  
2          0.4245      0.4504          0.2430          0.3613  \  
3          0.8663      0.6869          0.2575          0.6638
```

4	0.2050	0.4000	0.1625	0.2364
---	--------	--------	--------	--------

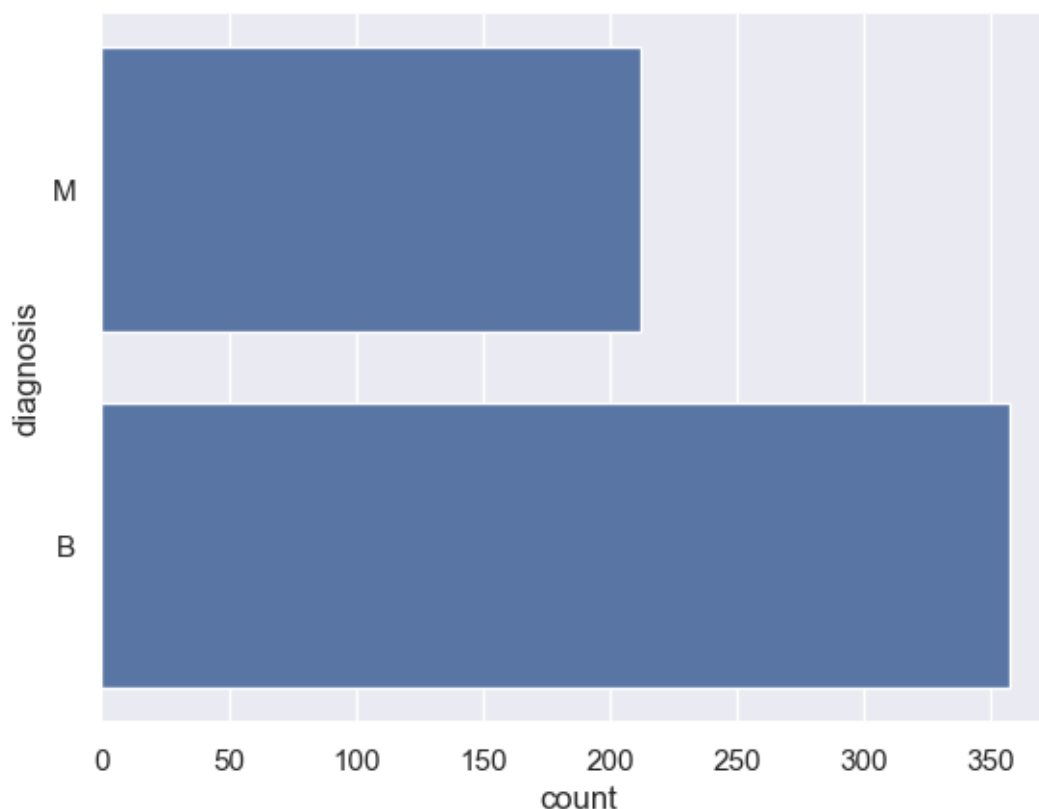
	fractal_dimension_worst	Unnamed: 32
0	0.11890	NaN
1	0.08902	NaN
2	0.08758	NaN
3	0.17300	NaN
4	0.07678	NaN

[5 rows x 33 columns]

```
[66]: df.columns
```

```
[66]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
          'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
          'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
          'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
          'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
          'fractal_dimension_se', 'radius_worst', 'texture_worst',
          'perimeter_worst', 'area_worst', 'smoothness_worst',
          'compactness_worst', 'concavity_worst', 'concave points_worst',
          'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
          dtype='object')
```

```
[67]: sns.countplot(df['diagnosis'])
plt.show()
```



```
[68]: df.drop(['Unnamed: 32'], axis = 1, inplace = True)
```

```
[69]: df.head()
```

```
[69]:
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave	points_mean	\
0	0.11840	0.27760	0.3001		0.14710	
1	0.08474	0.07864	0.0869		0.07017	
2	0.10960	0.15990	0.1974		0.12790	
3	0.14250	0.28390	0.2414		0.10520	
4	0.10030	0.13280	0.1980		0.10430	

	...	radius_worst	texture_worst	perimeter_worst	area_worst	\
0	...	25.38	17.33	184.60	2019.0	
1	...	24.99	23.41	158.80	1956.0	

2	...	23.57	25.53	152.50	1709.0
3	...	14.91	26.50	98.87	567.7
4	...	22.54	16.67	152.20	1575.0

	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	\
0	0.1622	0.6656	0.7119		0.2654
1	0.1238	0.1866	0.2416		0.1860
2	0.1444	0.4245	0.4504		0.2430
3	0.2098	0.8663	0.6869		0.2575
4	0.1374	0.2050	0.4000		0.1625

	symmetry_worst	fractal_dimension_worst
0	0.4601	0.11890
1	0.2750	0.08902
2	0.3613	0.08758
3	0.6638	0.17300
4	0.2364	0.07678

[5 rows x 32 columns]

```
[70]: df.drop(['id'], axis = 1, inplace = True)
```

```
[71]: df.head()
```

```
[71]:
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	M	17.99	10.38	122.80	1001.0	
1	M	20.57	17.77	132.90	1326.0	
2	M	19.69	21.25	130.00	1203.0	
3	M	11.42	20.38	77.58	386.1	
4	M	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.3001		0.14710
1	0.08474	0.07864	0.0869		0.07017
2	0.10960	0.15990	0.1974		0.12790
3	0.14250	0.28390	0.2414		0.10520
4	0.10030	0.13280	0.1980		0.10430

	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst	\
0	0.2419	...	25.38	17.33	184.60	
1	0.1812	...	24.99	23.41	158.80	
2	0.2069	...	23.57	25.53	152.50	
3	0.2597	...	14.91	26.50	98.87	
4	0.1809	...	22.54	16.67	152.20	

	area_worst	smoothness_worst	compactness_worst	concavity_worst	\
0	2019.0	0.1622	0.6656	0.7119	

1	1956.0	0.1238	0.1866	0.2416
2	1709.0	0.1444	0.4245	0.4504
3	567.7	0.2098	0.8663	0.6869
4	1575.0	0.1374	0.2050	0.4000

	concave points_worst	symmetry_worst	fractal_dimension_worst
0	0.2654	0.4601	0.11890
1	0.1860	0.2750	0.08902
2	0.2430	0.3613	0.08758
3	0.2575	0.6638	0.17300
4	0.1625	0.2364	0.07678

[5 rows x 31 columns]

```
[72]: df.columns
```

```
[72]: Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
          'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
          'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
          'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
          'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
          'fractal_dimension_se', 'radius_worst', 'texture_worst',
          'perimeter_worst', 'area_worst', 'smoothness_worst',
          'compactness_worst', 'concavity_worst', 'concave points_worst',
          'symmetry_worst', 'fractal_dimension_worst'],
          dtype='object')
```

```
[73]: X = df.iloc[:, 1:].values
      y = df['diagnosis'].values
```

```
[74]: X.shape
```

```
[74]: (569, 30)
```

```
[75]: y.shape
```

```
[75]: (569,)
```

```
[76]: from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state= 0)
```

0.0.1 Feature Scaling

```
[77]: from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

0.1 PCA (Principal Component Analysis)

```
[84]: from sklearn.decomposition import PCA

pca = PCA(n_components = 1)

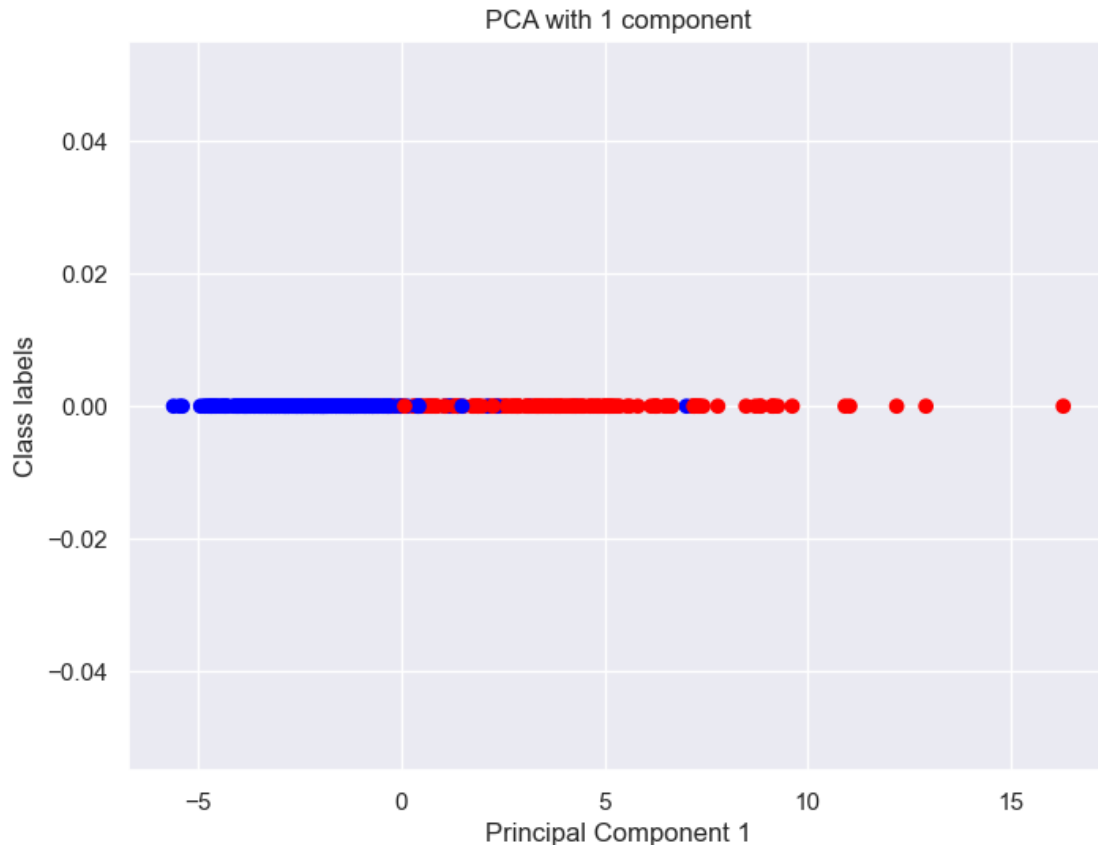
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

```
[86]: import matplotlib.pyplot as plt
import numpy as np

# Map class labels to colors
colors = {'B': 'blue', 'M': 'red'}

# Create a list of colors based on the class labels
c = [colors[label] for label in y_train]

plt.figure(figsize=(8, 6))
plt.scatter(X_train, np.zeros(len(X_train)), c=c)
plt.title('PCA with 1 component')
plt.xlabel('Principal Component 1')
plt.ylabel('Class labels')
plt.show()
```



0.1.1 Conclusion

PCA has no concern with the class labels. It summarizes the feature set without considering the output. PCA tries to find the directions of the maximum variance in the dataset. In a high cardinality feature set, there are possibilities of duplicate features which would add redundancy to the dataset, increase the computation cost and add unnecessary model complexity. **The role of PCA is to find such highly correlated or duplicate features and to come up with a new feature set where there is minimum correlation between the features or in other words feature set with maximum variance between the features.**

0.2 t-SNE (t-Distributed Stochastic Neighbor)

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,
↳random_state= 0)
```

```
[ ]: from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
[ ]: from sklearn.manifold import TSNE
```

```
[ ]: tsne = TSNE(n_components = 2, random_state = 0)
```

```
[ ]: tsne_obj = tsne.fit_transform(X_train)
```

```
c:\Users\Neil\anaconda3\Lib\site-  
packages\joblib\externals\loky\backend\context.py:136: UserWarning: Could not  
find the number of physical cores for the following reason:  
[WinError 2] The system cannot find the file specified  
Returning the number of logical cores instead. You can silence this warning by  
setting LOKY_MAX_CPU_COUNT to the number of cores you want to use.  
    warnings.warn(  
        File "c:\Users\Neil\anaconda3\Lib\site-  
packages\joblib\externals\loky\backend\context.py", line 257, in  
_count_physical_cores  
    cpu_info = subprocess.run(  
        ~~~~~~  
File "c:\Users\Neil\anaconda3\Lib\subprocess.py", line 548, in run  
    with Popen(*popenargs, **kwargs) as process:  
        ~~~~~~  
File "c:\Users\Neil\anaconda3\Lib\subprocess.py", line 1026, in __init__  
    self._execute_child(args, executable, preexec_fn, close_fds,  
File "c:\Users\Neil\anaconda3\Lib\subprocess.py", line 1538, in _execute_child  
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,  
        ~~~~~~
```

```
[ ]: tsne_df = pd.DataFrame({'X' : tsne_obj[:,0],  
                            'Y' : tsne_obj[:,1],  
                            'classification' : y_train  
                            })
```

```
[ ]: tsne_df.head()
```

```
[ ]:      X      Y classification  
0 -7.053003 -2.806510          B  
1 -8.142869 -9.293004          B  
2 -13.934480  8.887429          B  
3 -2.931662 -9.876837          B  
4 -9.162365 -12.554125          B
```

```
[ ]: tsne_df['classification'].value_counts()
```

```
[ ]: classification  
B    290  
M    165  
Name: count, dtype: int64
```



```
[ ]: plt.figure(figsize = (10,10))
sns.scatterplot(x = "X", y = 'Y', hue = 'classification', legend = 'full', data=
↪ tsne_df)
plt.show()
```



0.2.1 Conclusion

t-SNE is an unsupervised dimensionality reduction technique that does not take class labels into account. It focuses on preserving the local structure of the data rather than maximizing variance. t-SNE maps high-dimensional data into a lower-dimensional space, typically for visualization purposes, by minimizing the divergence between pairwise similarities in the original and reduced space. In datasets with high cardinality, complex structures can make it difficult to interpret relationships between data points. The role of t-SNE is to capture and reveal hidden structures in high-dimensional data by clustering similar points together while maintaining meaningful separa-

tions, making it particularly useful for visualizing patterns and groupings in complex datasets.

0.3 LDA (Linear Discriminant Analysis)

0.3.1 Performing LDA (Linear Discriminant Analysis)

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,
    ↪random_state= 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

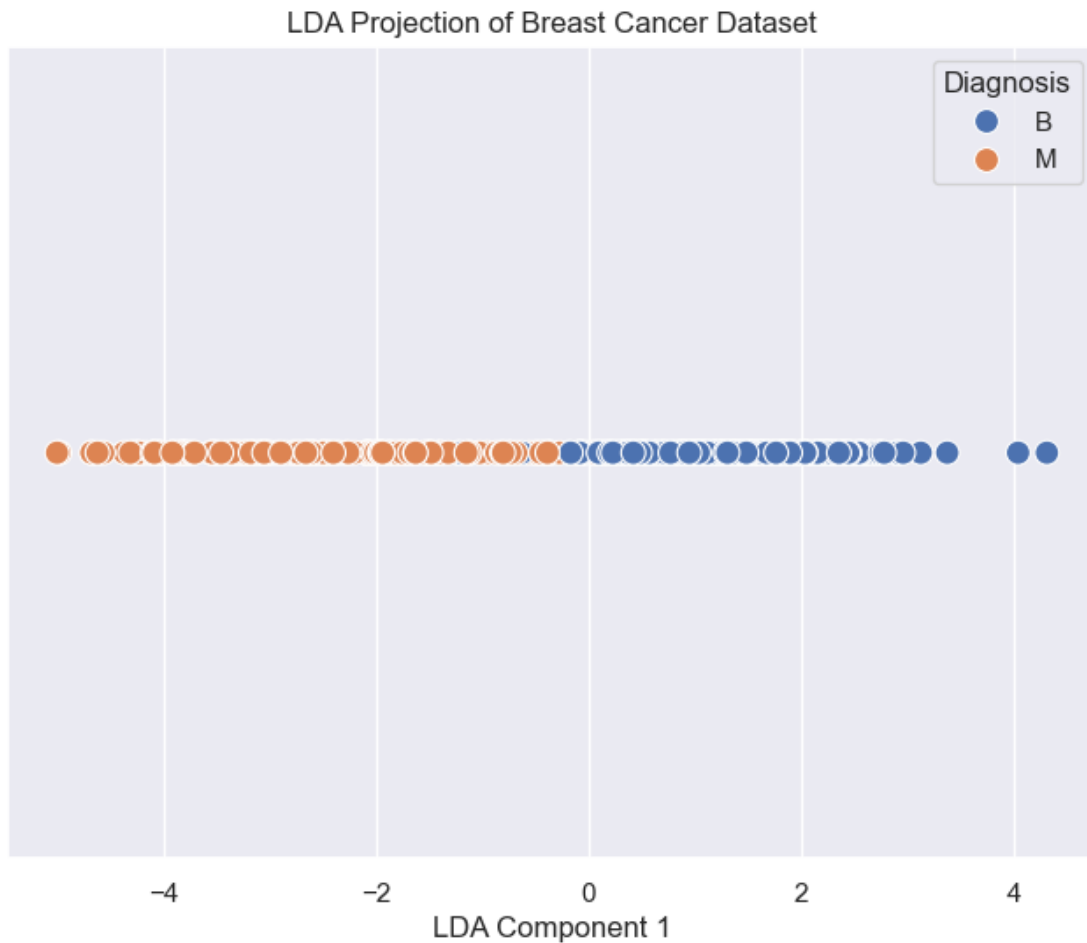
[ ]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components = 1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

[ ]: import seaborn as sns
import matplotlib.pyplot as plt

# Create a DataFrame for plotting
import pandas as pd
df_lda = pd.DataFrame({'LDA1': X_train[:, 0], 'Diagnosis': y_train})

# Plot using Seaborn
plt.figure(figsize=(8,6))
sns.scatterplot(x=df_lda['LDA1'], y=[0] * len(df_lda), hue=df_lda['Diagnosis'],
    ↪palette="deep", s=100)
plt.xlabel('LDA Component 1')
plt.title('LDA Projection of Breast Cancer Dataset')
plt.yticks([]) # Remove y-axis labels since it's 1D
plt.legend(title='Diagnosis')
plt.show()
```



1 What's the difference from PCA?

LDA tries to reduce the dimensionality by taking into consideration the information that discriminates the output classes. LDA tries to find the decision boundary around each cluster of class. It projects the data points to new dimension in a way that the clusters are as separate from each other as possible and individual elements within a class are as close to the centroid as possible. **In other words, the inter-class separability is increased in LDA. Intra-class separability is reduced.** The new dimensions are the linear discriminants of the feature set.

Experiment: Principal Component Analysis (PCA) vs Linear Discriminant Analysis (LDA) vs T-distributed Stochastic Neighbor Embedding (t-SNE)

Title:

Implementation of Linear Discriminant Analysis and Principal Component Analysis and T-distributed Stochastic Neighbor Embedding (t-SNE)

Aim:

Comparing the results of PCA with LDA and t-SNE for better suitability

Objective:

Students will learn:

- The implementation of the principal component analysis and Linear Discriminant analysis and T-distributed stochastic neighbor embedding on a dataset.
 - Visualization and interpretation of results.
-

Problem Statement

APPLY AND IMPLEMENT T-SNE ALGORITHM ON A SPECIFIC DATASET OF YOUR CHOICE AND COMPARE THE OUTCOMES WITH PCA AND LDA FOR THE SAME

Explanation / Stepwise Procedure / Algorithm

Principal Component Analysis (PCA)

PCA is a widely used dimensionality reduction technique that transforms high-dimensional data into lower-dimensional data while retaining most of the information. The goal of PCA is to identify the directions (principal components) in which the data varies the most and project the data onto those directions.

Here's how PCA works:

1. Standardize the data by subtracting the mean and dividing by the standard deviation.
2. Calculate the covariance matrix of the standardized data.

3. Compute the eigenvectors and eigenvalues of the covariance matrix.
4. Select the top k eigenvectors corresponding to the largest eigenvalues.
5. Project the original data onto the selected eigenvectors to obtain the lower-dimensional representation.

PCA is useful for:

- Reducing the dimensionality of high-dimensional data
- Identifying patterns and correlations in the data
- Improving the performance of machine learning algorithms

Linear Discriminant Analysis (LDA)

LDA is a supervised dimensionality reduction technique that aims to find a linear combination of features that separates classes of data. LDA is commonly used for classification problems and is particularly useful when the number of features is high and the number of samples is small.

Here's how LDA works:

1. Standardize the data by subtracting the mean and dividing by the standard deviation.
2. Compute the within-class scatter matrix and the between-class scatter matrix.
3. Compute the eigenvectors and eigenvalues of the scatter matrices.
4. Select the top k eigenvectors corresponding to the largest eigenvalues.
5. Project the original data onto the selected eigenvectors to obtain the lower-dimensional representation.

LDA is useful for:

- Reducing the dimensionality of high-dimensional data
- Improving the performance of classification algorithms
- Identifying the most discriminative features for classification

t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a non-linear dimensionality reduction technique that maps high-dimensional data to lower-dimensional data while preserving local structure. t-SNE is particularly useful for

visualizing high-dimensional data and identifying patterns or clusters that may not be apparent in the original data.

Here's how t-SNE works:

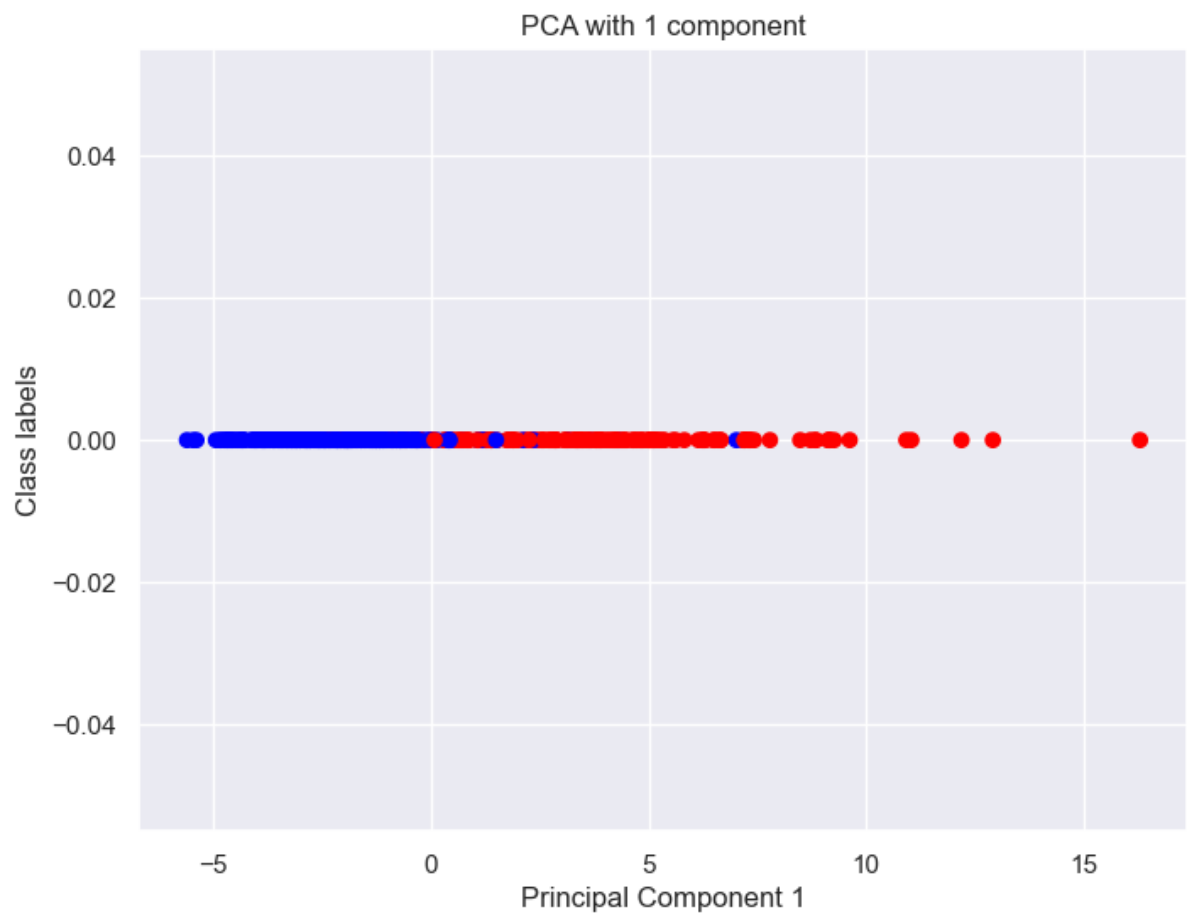
1. Compute the similarity matrix of the high-dimensional data using a Gaussian kernel.
2. Convert the similarity matrix into a joint probability distribution.
3. Define a cost function that measures the difference between the joint probability distribution and the joint probability distribution of the lower-dimensional data.
4. Minimize the cost function using gradient descent to obtain the lower-dimensional representation.

t-SNE is useful for:

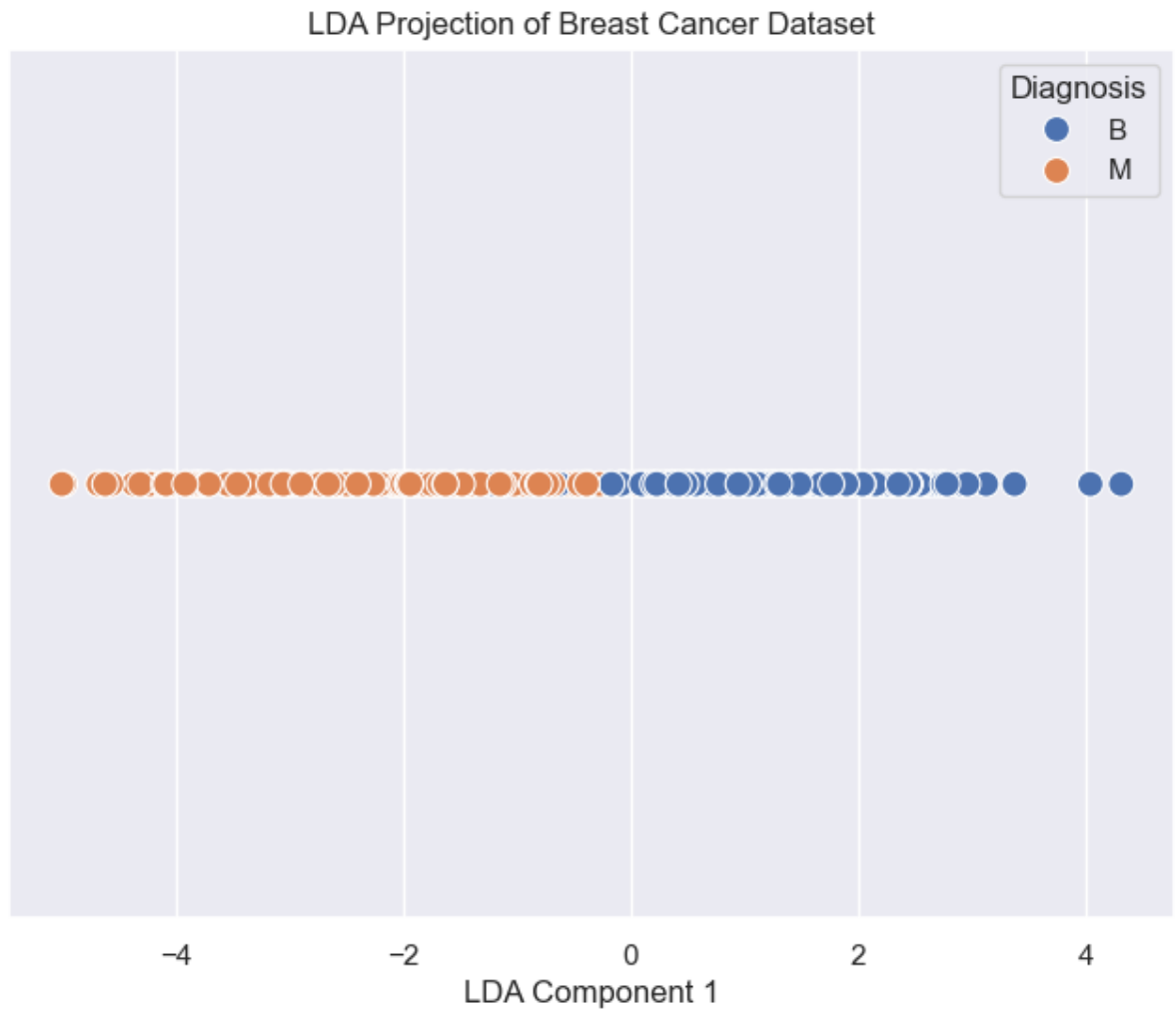
- Visualizing high-dimensional data in a lower-dimensional space
- Identifying patterns and clusters in the data
- Reducing the dimensionality of high-dimensional data while preserving local structure

Figures/Diagrams

- LDA and PCA and t-SNE plots plotted for the dataset.
- Comparison between LDA and PCA and t-SNE.







Challenges Encountered

1. T-SNE needs careful selection of parameters like perplexity and learning rate for good results, or else the outcome may not be accurate.
 2. T-SNE can be slow and expensive for large datasets, making it challenging to apply to big data and get quick results.
 3. Unlike PCA, which preserves global structure, T-SNE focuses on local structure, making it harder to understand the results, especially for complex high-dimensional data.
 4. Comparing T-SNE with LDA can be challenging because LDA is a supervised method that relies on class labels, whereas T-SNE is unsupervised, making it difficult to evaluate their performance on the same dataset.
-

Conclusion

- In conclusion, T-SNE is a powerful tool for dimensionality reduction, but it requires careful parameter selection and can be slow for large datasets.

- Unlike PCA, which preserves global structure, T-SNE focuses on local structure, providing a unique perspective on the data.
- While LDA is a supervised method that excels in certain tasks, T-SNE's unsupervised nature makes it a valuable addition to any data analyst's toolkit.
- Ultimately, the choice between T-SNE, PCA, and LDA depends on the specific needs of the project and the characteristics of the data.