IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⋇ Impact Factor 7.105 ⋇ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96127

WINE CLASSIFICATION USING LDA

Tikendrajit Sarmah[1], Sandarsh Gowda MM[2]

[1]Student, Dept. of MCA, Bangalore Institute of Technology, Bengaluru, India

[2]Assoc Professor, Dept. of MCA, Bangalore Institute of Technology, Bengaluru, India.

Abstract: Analysis of wine categories was the primary focus of this endeavour. Through empirical research, the wine dataset is reduced in dimension and grouped into clusters. It is necessary to do a factor analysis on the 13 variables in order to separate the complicated ones into 5 categories: the bitter tropic factor; the visual assessment; colour; pH; and mineral elements. Cluster analysis using K-means clustering was used to divide the data into three distinct groups. The features of each variety may be summarised according to the cluster centre. Using a series of empirical investigations, it is possible to draw a general picture of the wine's qualities and to group them into several groups.

INTRODUCTION

Linear Discriminant Analysis is a linear classification machine learning algorithm.
The Linear Discriminant Analysis (LDA) technique is developed to transform the features into a lower dimensional space, which maximizes the ratio of the between-class variance to the within-class variance, thereby guaranteeing maximum class separability.

$$J(W) = \frac{W^T S_B \, W}{W^T S_W \, W} \qquad \text{....(1)}$$

There are two types of LDA technique to deal with classes:

- class-dependent and class-independent. In the class-dependent LDA, one separate lower dimensional space is calculated for each class to project its data on it.
- In the class-independent LDA, each class will be considered as a separate class against the other classes. In this type, there is just one lower dimensional space for all classes to project their data on it.

THEORETICAL BACKGROUND TO LDA

Let the original data matrix X = {x1, x2, . . . , xN }, where xi represents the i th sample, pattern, or observation and N is the total number of samples. Each sample is represented by M features (xi ∈ RM). In other words, each sample is represented as a point in M-dimensional space. The data matrix is partitioned into c classes as follows,
X = [ω1, ω2, . . . , ωc]. Thus, Each class has ni samples. The total number of samples (N) is calculated as follows, N = Pc i=1 ni .

The goal of the LDA technique is to project the original data matrix onto a lower dimensional space. To achieve this goal, three steps needed to be performed.

1. The first step is to calculate the separability between different classes (i.e. the distance between the means of different classes), which is called the between-class variance or between-class matrix.

2. The second step is to calculate the distance between the mean and the samples of each class, which is called the within-class variance or within-class matrix.

3. The third step is to construct the lower dimensional space which maximizes the between-class variance and minimizes the within-class variance.

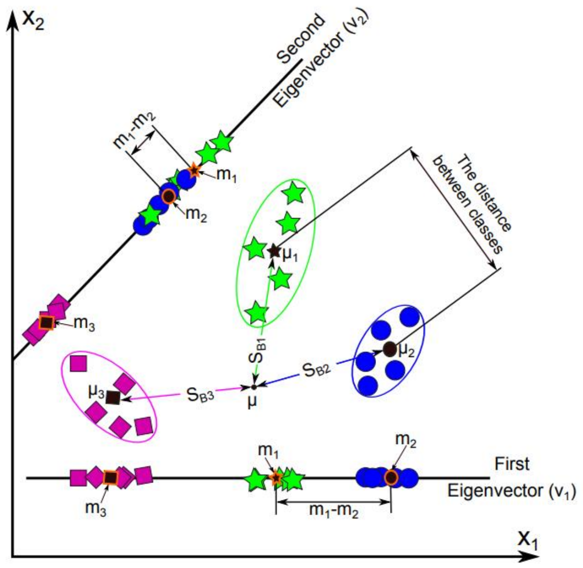
CALCULATING THE BETWEEN-CLASS VARIANCE (SB)

To calculate the between-class variance (SB), the separation distance between different classes which is denoted by (mi − m) will be calculated as follows,
(mi − m) 2 = (WT μi − WT μ) 2 = WT (μi − μ)(μi − μ) TW

- where, *mi* represents the projection of the mean of the i th class and it is calculated as follows, *mi = WT μi*
- *m* is the projection of the total mean of all classes and it is calculated as follows, $m = W^T \mu$.
- *W* represents the transformation matrix of LDA

- *μj (1 × M)* represents the mean of the i th class *(μj = 1 nj P xi∈ωj xi)*, and *μ(1 × M)* is the total mean of all classes *(μ = 1 N PN i=1 xi = 1 c Pc j=1 μj)*, where c represents the total number of classes.
- The term (μi − μ)(μi − μ) T in Equation (2) represents the separation distance between the mean of the i th class (μi) and the total mean (μ), or simply it represents the between-class variance of the i th class (SBi).
- Substitute SBi into Equation (2) as follows, (mi − m) 2 = WT SBi W (3)
- The total between-class variance is calculated as follows,

$$\left(S_B = \sum_{i=1}^{c} n_i S_{B_i}\right)$$



CALCULATING THE WITHIN-CLASS VARIANCE (SW)

The within-class variance of the i^th class (S_{Wi}) represents the difference between the mean and the samples of that class. LDA technique searches for a lower-dimensional space, which is used to minimize the difference between the projected mean (m_i) and the projected samples of each class ($W^T x_i$), or simply minimizes the within-class variance.

$$\begin{aligned}\sum_{x_i \in \omega_j} (W^T x_i - m_j)^2 \\ &= \sum_{x_i \in \omega_j} (W^T x_i - W^T \mu_j)^2 \\ &= \sum_{x_i \in \omega_j} W^T (x_i - \mu_j)^2 W \\ &= \sum_{x_i \in \omega_j} W^T (x_i - \mu_j)(x_i - \mu_j)^T W \\ &= \sum_{x_i \in \omega_j} W^T S_{W_j} W\end{aligned} \qquad (4)$$

From Equation (4), the within-class variance for each class can be calculated as follows,

$S_{W_j} = d_j^T * d_j = \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T$ where xij represents the i th sample in the j th class as shown in Fig. (2, step (E, F)), and dj is the centering data of the j th class, i.e.

$d_j = \omega_j - \mu_j = \{x_i\}^{nj}_{i=1} - \mu j$

- Step (F) in the figure illustrates how the within-class variance of the first class (S_{Wi}) in our example is calculated.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

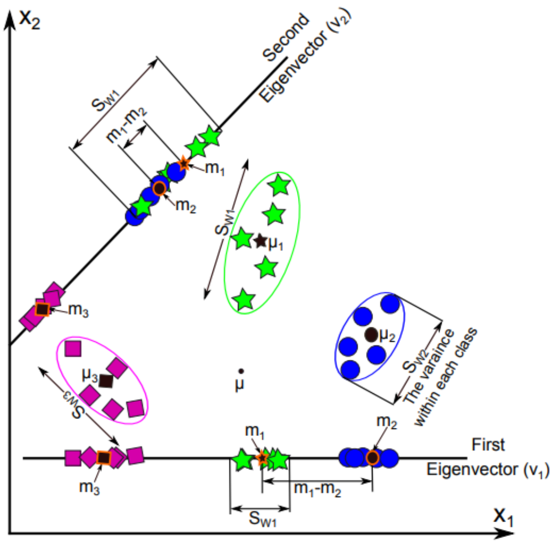
International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96127

• The total within-class variance represents the sum of all within-class matrices of all classes (see Fig. (2, step (F))), and it can be calculated as in Equation (5).

$$S_W = \sum_{i=1}^{c} S_{W_i} = \sum_{x_i \in \omega_1} (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{x_i \in \omega_2} (x_i - \mu_2)(x_i - \mu_2)^T + \cdots + \sum_{x_i \in \omega_c} (x_i - \mu_c)(x_i - \mu_c)^T \quad \ldots\ldots.(5)$$



CONSTRUCTING THE LOWER DIMENSIONAL SPACE

After calculating the between-class variance (SB) and within-class variance (SW), the transformation matrix (W) of the LDA technique can be calculated as in Equation (1), which can be reformulated as an optimization problem as in Equation (6).

$$S_W W = \lambda S_B W \quad \ldots\ldots\ldots\ldots(6)$$

where λ represents the Eigen values of the transformation matrix (W). The solution of this problem can be obtained by calculating the Eigen values (λ) and eigenvectors (V = {v1, v2,. . ., vM}) of
$W = S_w^{-1} S_B$, if S_W is non-singular.

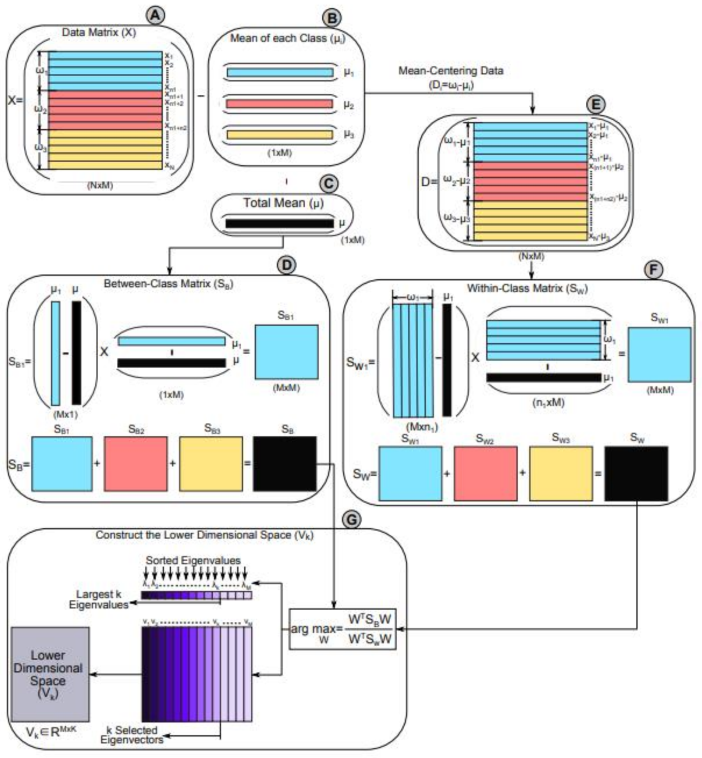
The Eigen values are scalar values, while the eigenvectors are non-zero vectors provides us with the information about the LDA space. The eigenvectors represent the directions of the new space, and the corresponding Eigen values represent the scaling factor, length, or the magnitude of the eigenvectors.
Thus, each eigenvector represents one axis of the LDA space and the associated Eigen value represents the robustness of this eigenvector. The robustness of the eigenvector reflects its ability to discriminate between different classes and decreases the within-class variance of each class, hence meets the LDA goal.
Thus, the eigenvectors with the k highest Eigen values are used to construct a lower dimensional space (V_k), while the other eigenvectors ({v_k+1, v_k+2, v_M}) are neglected.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⋇ Impact Factor 7.105 ⋇ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96127



PROJECTION ONTO THE LDA SPACE

The dimension of the original data matrix (X ∈ RN×M) is reduced by projecting it onto the lower dimensional space of LDA (Vk ∈ RM×k) as denoted in Equation (7).
The dimension of the data after projection is k; hence, M − k features are ignored or deleted from each sample. Each sample (xi) which was represented as a point a M-dimensional space will be represented in a k-dimensional space by projecting it onto the lower dimensional space (Vk) as follows, $y_i = x_i V_k$

$$Y = XV_k \qquad \text{.........(7)}$$

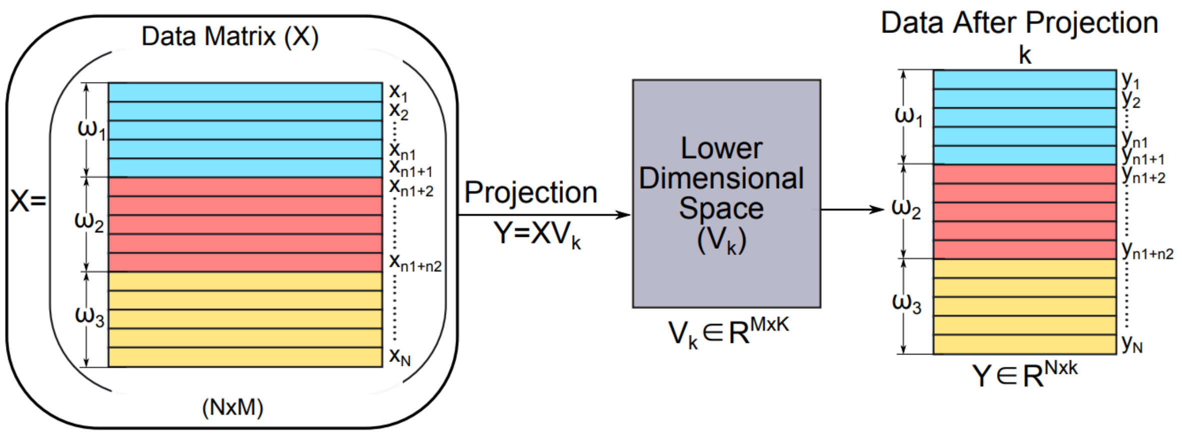


Figure: Projection of the original samples (i.e. data matrix) on the lower dimensional space of LDA (Vk)

IMPLEMENTATION OF LDA

LDA tries to reduce dimensions of the feature set while retaining the information that discriminates output classes. LDA tries to find a decision boundary around each cluster of a class. It then projects the data points to new dimensions in a way that the clusters are as separate from each other as possible and the individual elements within a cluster are as close to the centroid of the cluster as possible. The new dimensions are ranked on the basis of their ability to maximize

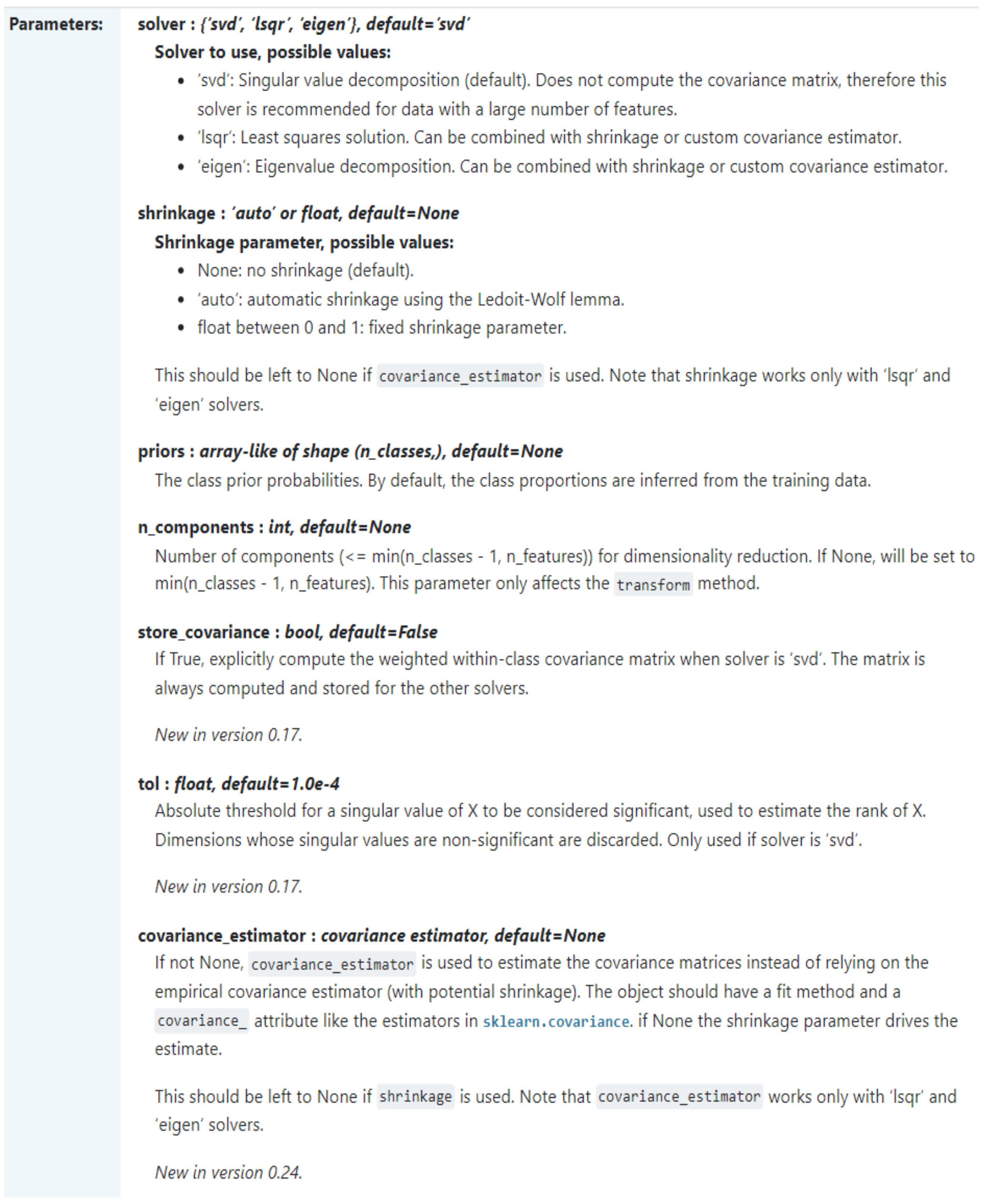
DOI: 10.17148/IARJSET.2022.96127

the distance between the clusters and minimize the distance between the data points within a cluster and their centroids. These new dimensions form the linear discriminants of the feature set.

IMPLEMENTING LDA WITH SCIKIT-LEARN

A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.
The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions, using the transform method.

PARAMETERS



Parameters:

solver : *{'svd', 'lsqr', 'eigen'}, default='svd'*

Solver to use, possible values:

- 'svd': Singular value decomposition (default). Does not compute the covariance matrix, therefore this solver is recommended for data with a large number of features.
- 'lsqr': Least squares solution. Can be combined with shrinkage or custom covariance estimator.
- 'eigen': Eigenvalue decomposition. Can be combined with shrinkage or custom covariance estimator.

shrinkage : *'auto' or float, default=None*

Shrinkage parameter, possible values:

- None: no shrinkage (default).
- 'auto': automatic shrinkage using the Ledoit-Wolf lemma.
- float between 0 and 1: fixed shrinkage parameter.

This should be left to None if `covariance_estimator` is used. Note that shrinkage works only with 'lsqr' and 'eigen' solvers.

priors : *array-like of shape (n_classes,), default=None*

The class prior probabilities. By default, the class proportions are inferred from the training data.

n_components : *int, default=None*

Number of components (<= min(n_classes - 1, n_features)) for dimensionality reduction. If None, will be set to min(n_classes - 1, n_features). This parameter only affects the `transform` method.

store_covariance : *bool, default=False*

If True, explicitly compute the weighted within-class covariance matrix when solver is 'svd'. The matrix is always computed and stored for the other solvers.

New in version 0.17.

tol : *float, default=1.0e-4*

Absolute threshold for a singular value of X to be considered significant, used to estimate the rank of X. Dimensions whose singular values are non-significant are discarded. Only used if solver is 'svd'.

New in version 0.17.

covariance_estimator : *covariance estimator, default=None*

If not None, `covariance_estimator` is used to estimate the covariance matrices instead of relying on the empirical covariance estimator (with potential shrinkage). The object should have a fit method and a `covariance_` attribute like the estimators in `sklearn.covariance`. if None the shrinkage parameter drives the estimate.

This should be left to None if `shrinkage` is used. Note that `covariance_estimator` works only with 'lsqr' and 'eigen' solvers.

New in version 0.24.

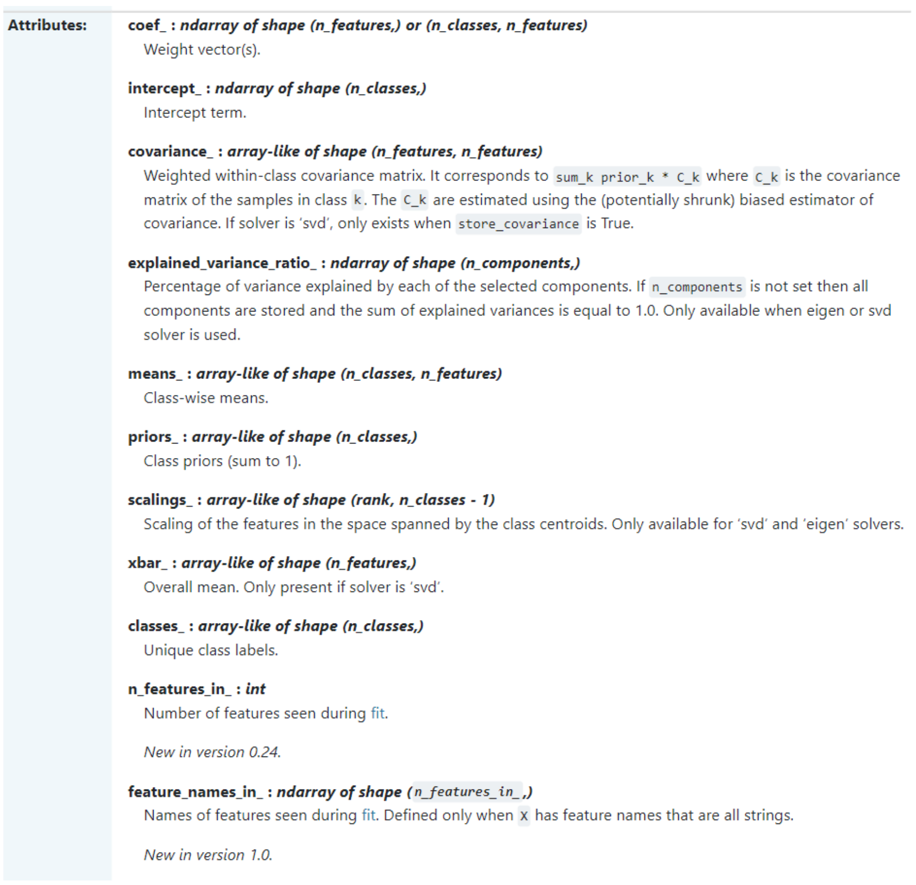
IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 6, June 2022

DOI: 10.17148/IARJSET.2022.96127

ATTRIBUTES



Attributes:

coef_ : *ndarray of shape (n_features,) or (n_classes, n_features)*
Weight vector(s).

intercept_ : *ndarray of shape (n_classes,)*
Intercept term.

covariance_ : *array-like of shape (n_features, n_features)*
Weighted within-class covariance matrix. It corresponds to `sum_k prior_k * C_k` where `C_k` is the covariance matrix of the samples in class `k`. The `C_k` are estimated using the (potentially shrunk) biased estimator of covariance. If solver is 'svd', only exists when `store_covariance` is True.

explained_variance_ratio_ : *ndarray of shape (n_components,)*
Percentage of variance explained by each of the selected components. If `n_components` is not set then all components are stored and the sum of explained variances is equal to 1.0. Only available when eigen or svd solver is used.

means_ : *array-like of shape (n_classes, n_features)*
Class-wise means.

priors_ : *array-like of shape (n_classes,)*
Class priors (sum to 1).

scalings_ : *array-like of shape (rank, n_classes - 1)*
Scaling of the features in the space spanned by the class centroids. Only available for 'svd' and 'eigen' solvers.

xbar_ : *array-like of shape (n_features,)*
Overall mean. Only present if solver is 'svd'.

classes_ : *array-like of shape (n_classes,)*
Unique class labels.

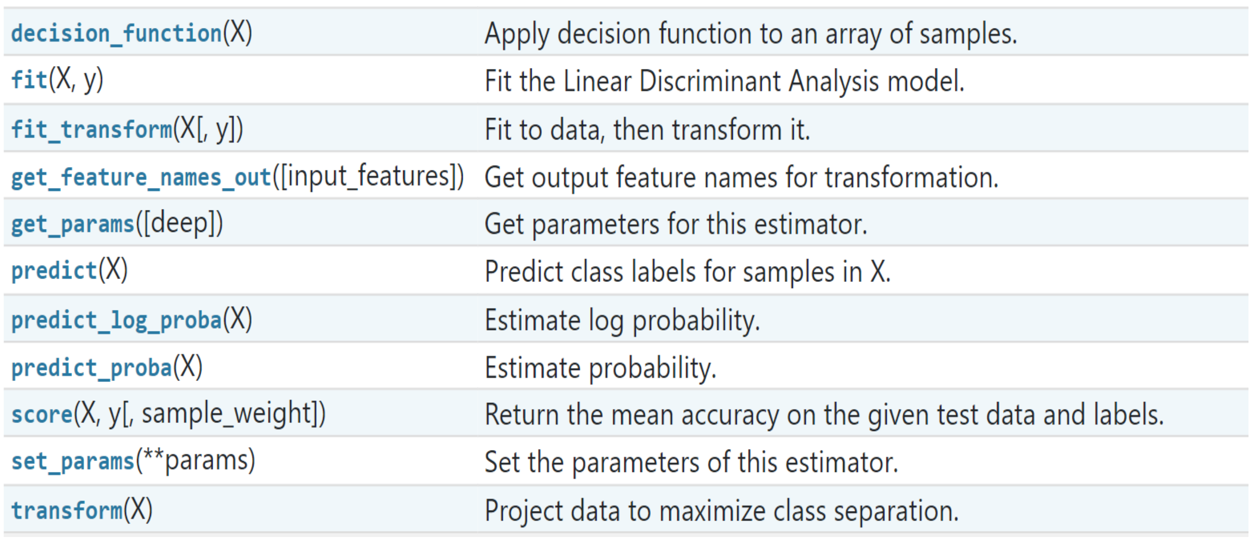
n_features_in_ : *int*
Number of features seen during fit.

New in version 0.24.

feature_names_in_ : *ndarray of shape (`n_features_in_`,)*
Names of features seen during fit. Defined only when `X` has feature names that are all strings.

New in version 1.0.

METHODS



`decision_function`(X)	Apply decision function to an array of samples.
`fit`(X, y)	Fit the Linear Discriminant Analysis model.
`fit_transform`(X[, y])	Fit to data, then transform it.
`get_feature_names_out`([input_features])	Get output feature names for transformation.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict class labels for samples in X.
`predict_log_proba`(X)	Estimate log probability.
`predict_proba`(X)	Estimate probability.
`score`(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
`set_params`(**params)	Set the parameters of this estimator.
`transform`(X)	Project data to maximize class separation.

In my project, a classifier is constructed which determines the Customer Segment to which a specific wine sample belongs. Each sample consists of a vector $\mathbf{v}$ of 13 attributes of the wine, that is $\mathbf{v} \in \mathbb{R}^{13}$. The attributes are as follows:

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash

5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

Based on these attributes, the goal is to identify from which of three Customer Segment, the data is originated. The data set is available at the UCI Machine Learning Repository. The following code block shows three rows from the dataset.

In my project, The first 13 columns contain the sample features and last column denotes the target class while the remaining. This data can be read into a matrix using the pandas function.
Below code are used to read and display csv file of the project-

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#Import dataset
dataset = pd.read_csv("C:\\Users\TIKEN\Desktop\Wine.csv")
print(dataset.head())
```

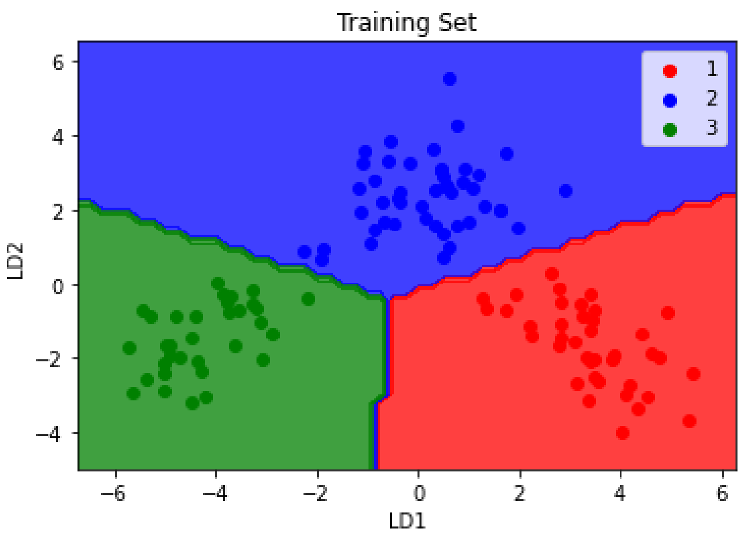
The first five row of the dataset is exactly looks below-

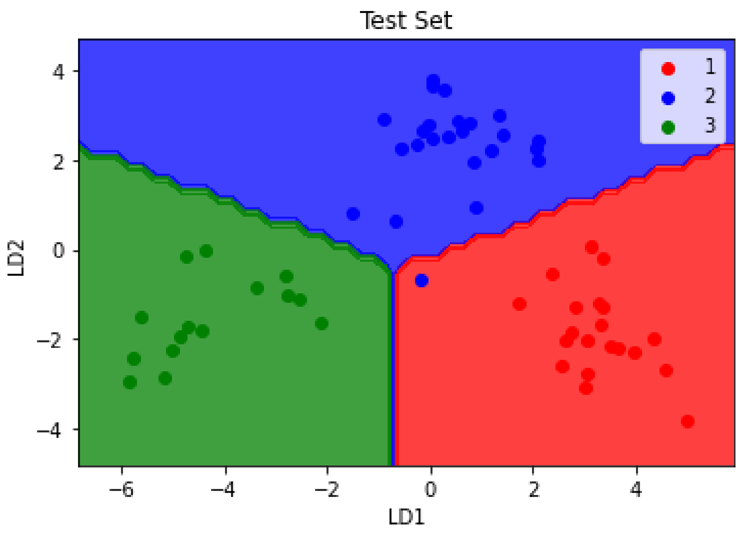
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1

After that, I splited the dataset into training set and test set using train_test_split() function from sci-kit learn model_selection
I also import Linear Discriminant Analysis from sklearn.discriminant_analysis to analyze these data sets.
Then I used contourf and scatter plot from matplotlib.pyplot to visualize the difference of above mentioned data sets.

RESULTS

At the end, the result comes out that the result set of test dataset is not accurate with the trained dataset. The difference of the result sets are clearly visible in below mentioned Scatter plot -





CONCLUSION

In this experiment A Linear Discriminant Analysis model was proposed to be developed fromscratch without using any pre-trained models. The model performed pretty well on our dataset collected from UCI Machine Learning repository achieving about 98 % accuracy on test data by the model which is quite amazing. Though LDA achieves leading results in potato plant disease image analysis tasks and there is still scope for development. We intend to develop a more efficient LDA structure to classify wine consumers dataset.

REFERENCES:

1. Linear Discriminant Analysis January 2015 Authors: Alaa Tharwat, Fachhochschule Bielefeld
2. Hastie, Trevor, et al. "The elements of statistical learning: data mining, inference and prediction." The Mathematical Intelligencer 27.2 (2005): 83-85.
3. Wine Classification Using Linear Discriminant Analysis https://nicholastsmith.wordpress.com/2016/02/13/wine-classification-using-linear-discriminant-analysis-with-python-and-scikit-learn/
4. M. D. Cabezudo, M. Herraiz, and E. F Gorostiza, "On the main analytical characteristics for solving enological problems," Process Biochem., no. 18, pp. 17–23, 1983.
5. The dataset is collected from https://archive.ics.uci.edu/ml/datasets/Wine
6. Tutorial hints from "https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/#:~:text=Linear%20Discriminant%20Analysis%20or%20Normal,separating%20two%20or%20more%20classes."
7. McLachlan, G. J. (2004). Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience

DOI: 10.17148/IARJSET.2022.96127

8. Lachenbruch, P. A. (1975). Discriminant analysis. NY: Hafner
9. Wilson K.M. Wong, Vinod Thorat, Mugdha V. Joglekar, Charlotte X. Dong et al. "Machine learning algorithms in big data analyses identify determinants of insulin gene transcription" , Cold Spring Harbor Laboratory, 2021
10. From scikit-learn.org https://scikit-learn.org/stable/modules/preprocessing.html
11. K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, San Diego, California, 1990.
12. S. Axler, Linear Algebra Done Right, Springer-Verlag New York Inc., New York, New York,1995.
13. P.N. Belhumeour, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711–720, 1997.
14. Agarwal S, Mittal N, Sureka A (2018) Potholes and bad road conditions- mining Twitter to extract information on killer roads. In: CoDS-COMAD '18 Proceedings of the ACM India joint international conference on data science and management of data. ACM, pp 67–77
15. Agarwal S, Sureka A (2017) Investigating the role of Twitter in E-governance by extracting information on citizen complaints and grievances reports. In: Reddy P, Sureka A, Chakravarthy S, Bhalla S (eds) Big data analytics. BDA 2017. Lecture Notes in Computer Science, vol 10721.Springer, Cham