# Predict Recommend the customer segment for a new wine based on the wine shop dataset by performing dimensionality reduction by using LDA:

To predict and recommend the customer segment for a new wine based on the wine shop dataset using Linear Discriminant Analysis (LDA), you need labeled data that includes information about different wines and their corresponding customer segments. LDA is a supervised dimensionality reduction technique that seeks to find a linear combination of features that maximally separates the classes in the data.

Here is a general step-by-step guide on how you can perform dimensionality reduction using LDA to predict and recommend customer segments for a new wine:

1. \*\*Prepare the dataset\*\*: Make sure your wine shop dataset is well-organized and contains relevant information about wines and customer segments. Each wine sample should have a set of features (e.g., flavor profile, acidity, sweetness) and a corresponding customer segment label.

2. \*\*Split the dataset\*\*: Split your dataset into training and testing sets. The training set will be used to train the LDA model, while the testing set will be used to evaluate its performance.

3. \*\*Normalize the features\*\*: It is important to normalize the features before applying LDA to ensure that they have similar scales. You can use standardization (subtracting the mean and dividing by the standard deviation) or other appropriate scaling methods.

4. \*\*Apply LDA\*\*: Train the LDA model on the training set using the labeled data. LDA seeks to find a projection that maximizes the separation between classes while minimizing the variance within each class.

5. \*\*Transform the data\*\*: Apply the trained LDA model to both the training and testing sets to project the data onto the lower-dimensional space.

6. \*\*Train a classifier\*\*: Once the data is transformed, you can use a classification algorithm (e.g., logistic regression, random forest, or support vector machines) to train a model on the transformed data.

7. \*\*Evaluate the model\*\*: Use the testing set to evaluate the performance of your trained model. You can use metrics such as accuracy, precision, recall, or F1 score to assess the model's predictive capabilities.

8. \*\*Predict the customer segment for a new wine\*\*: Once the model is trained and evaluated, you can use it to predict the customer segment for a new wine by applying the same feature normalization and LDA transformation on the new wine data. Then, use the trained classifier to make predictions based on the transformed data.

It's worth mentioning that the quality of your predictions will heavily depend on the quality and representativeness of your training data. Additionally, ensure that your dataset has enough samples for each customer segment to avoid biased predictions.

Remember that this is a general guide, and the actual implementation details may vary based on your specific dataset and programming language or tool you are using for analysis.

****Linear Discriminant Analysis (LDA)****

LDA is useful when we have more than two classes and when the number of observations is small. LDA is also more stable when distribution of the predictors are normal in each class.

> library(MASS)> lda.fit=lda(class~.,data=wine[-test,])> lda.fit

The last command will generate more details about the model as shown in Table 2.

IMG_256

Table 2. Average of each of 13 predictors for every class of wine

We then evaluate the performance of the model on test data:

> lda.pred=predict(lda.fit,wine[test,])> table(lda.pred$class,wine[test,]$class)

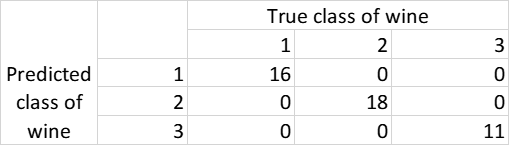


Table 3. Confusion matrix for LDA model

We can see from Table 3 that LDA has accuracy of 100% in predicting classes of test data.

We can also visualize classification of training data by LDA using below command and result is shown in Figure 1:

> plot(lda.fit)

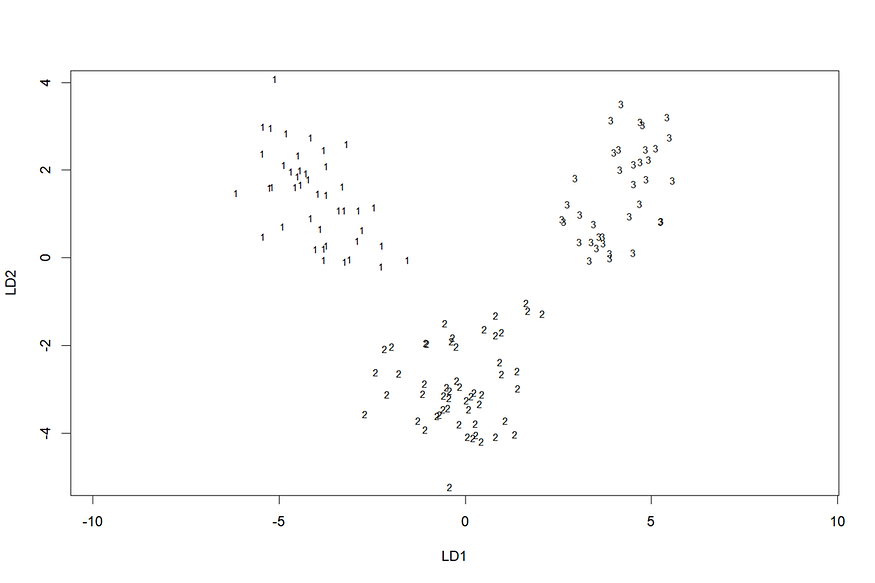


Figure 1. Classification of training data by LDA (Image by Author)

As there are three classes in data set, only two linear discriminants are needed to classify each observation. Figure 1 shows plot of training data on LD1 and LD2 space and the corresponding class for each data point. LD1 and LD2 values are computed based on coefficients of LDA model.

We can repeat the above process multiple times to get a more accurate estimate of the performance of LDA model by following these commands:

> for (i in 1:50) {+ test=sample(178,45)+ lda.fit=lda(class~.,data=wine[-test,])+ lda.pred=predict(lda.fit,wine[test,])+ Accuracy[i]=mean(lda.pred$class==wine[test,]$class)+ }> sum(Accuracy)/50[1] 0.9844444

([[6.4,0.1,4.3,3,690,0.8,0.75,0.43,1.41,0.3,0.7,1.56,10]])-1

([[6.4,0.1,4.3,3,690,0.8,0.75,0.43,4.41,0.3,03.7,41.56,1033]])-3

This code performs a classification task using Linear Discriminant Analysis (LDA) and Logistic Regression. Let's break down the code step by step:

1. Importing the necessary libraries: The code starts by importing the required libraries for data manipulation, visualization, and machine learning algorithms. These include numpy, pandas, matplotlib, seaborn, scikit-learn's LDA, confusion\_matrix, and metrics.

2. Importing the dataset: The code reads a CSV file named "wine.csv" using pandas and stores it in a DataFrame called 'df'. The dataset contains information about wine samples.

3. Correlation Matrix: The code calculates the correlation matrix of the dataset using the `corr()` function. It visualizes the correlation matrix using a heatmap created with the seaborn library.

4. Correlation for Customer Segment column: The code calculates the correlation values between the 'Customer\_Segment' column and other columns in the dataset. It sorts the correlation values in descending order to see which features are most correlated with the target variable.

5. Percentage of customer groups in the dataset: The code prints the number of samples belonging to each customer group in the 'Customer\_Segment' column.

6. Pie chart visualization: The code creates a pie chart to visualize the percentage of samples in each customer group.

7. Scatter plot: The code creates a scatter plot using the 'Ash\_Alcanity' and 'Color\_Intensity' columns from the dataset. It uses different colors to represent different customer segments.

8. Splitting the data into training set and test set: The code splits the dataset into training and test sets using the `train\_test\_split` function from scikit-learn. It assigns 80% of the data to the training set and 20% to the test set.

9. Feature Scaling: The code applies feature scaling to the training and test sets using the `StandardScaler` from scikit-learn. This step standardizes the features to have zero mean and unit variance.

10. Applying LDA: The code applies Linear Discriminant Analysis (LDA) to the training set. It reduces the dimensionality of the feature space to two dimensions using the `fit\_transform` method of LDA. It also transforms the test set using the learned LDA model.

11. Logistic Regression: The code creates a logistic regression classifier using the `LogisticRegression` class from scikit-learn. It trains the classifier on the LDA-transformed training set.

12. Predictions and Evaluations: The code uses the trained classifier to make predictions on the LDA-transformed test set. It calculates the confusion matrix and classification report to evaluate the model's performance.

13. Visualizing the Training and Test set results: The code creates scatter plots to visualize the predicted classes on the LDA-transformed training and test sets.

14. Model Saving: The code uses the `pickle` module to save the trained classifier (`classifier`), LDA model (`lda`), and feature scaler (`sc`) into separate files.

Overall, this code performs exploratory data analysis, applies LDA for dimensionality reduction, trains a logistic regression classifier, evaluates the model, and saves the trained model for future use.