IF540 - Gasal 2021-2022

Modul Lab 04 - Linear Discriminant Analysis

September 10, 2021

1 LDA technique for Dimensionality Reduction

Dataset Description: These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The initial data set had around 30 variables, but for some reason Only have the 13-dimensional version. The attributes are: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.

All attributes are continuous: No statistics available, but suggest to standardize the variables for certain uses (e.g. For use with classifiers which are NOT scaled invariant) NOTE: 1st attribute is the class identifier (1–3). I use the PCA technique for the Dimensionality Reduction of the wine dataset.

Part 1: Data Preprocessing

1.1 Import the Libraries In this step, we import three Libraries in Data Preprocessing part. A library is a tool that you can use to make a specific job. First of all, we import the numpy library used for multidimensional array then import the pandas library used to import the dataset and in last we import matplotlib library used for plotting the graph.

Code:

[1]:
```
# import the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Import the dataset In this step, we import the dataset to do that we use the pandas library. After import our dataset we define our Predictor and target attribute. we call 'X' predictor here and target attribute which we call 'y' here.

Code:

[2]:
```
# import the dataset
dataset = pd.read_csv('Wine.csv')
X = dataset.iloc[:, :13].values
Y = dataset.iloc[:, 13].values
```

1.3 Split the dataset for test and train In this step, we split our dataset into a test set and train set and an 80% dataset split for training and the remaining 20% for tests.

Code:

[3]:
```
# spliting the dataset into train set and test set
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)
```

Feature Scaling Feature Scaling is the most important part of data preprocessing. If we see our dataset then some attribute contains information in Numeric value some value very high andsome are very low if we see the age and estimated salary. This will cause some issues in our machinery model to solve that problem we set all values on the same scale there are two methods to solve that problem first one is Normalize and Second is Standard Scaler.

Here we use standard Scaler import from Sklearn Library.

Code:

[4]:
```
# feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X= StandardScaler()
X_train = sc_X.fit_transform(X_train)

X_test = sc_X.transform(X_test)
```

Part 2: Building a Linear Discriminant analysis for Dimensionality Reduction In this part, we use LDAA for Dimensionality Reduction.

2.1 Import the Libraries In this step, we import an LDA model from Scikit Learn Library.

Code:

[5]:
```
# import LDA model
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

2.2 Initialize our model In this step, we use the number of components = 2 which have high covariance

Code:

[6]:
```
# initialize the LDA
lda = LDA(n_components = 2)
```

2.3 Fitting the Model In this step, we fit the X data into the model.

Code:

[7]:
```
# fitting the LDA model
X_test = lda.fit_transform(X_test, y_test)
X_train = lda.transform(X_train)
```

3.1 Import the Libraries In this step, we are building our model to do this first we import a model from Scikit Learn Library.

Code:

[8]:
```
# import the Logistic Regression model from sklearn using the 2 variances with␣
 ↪the help of LDA
from  sklearn.linear_model import LogisticRegression
```

3.2 Initialize our Logistic Regression model In this step, we initialize our Logistic Regression model

Code:

[9]:
```
LG=LogisticRegression(random_state=0)
```

3.3 Fitting the Model In this step, we fit the training data into our model X_train, y_train is our training data.

Code:

[10]:
```
# fit the Logistic Regression model
LG.fit(X_train,y_train)
```

[10]:
```
LogisticRegression(random_state=0)
```

1.0.1 Part 4: Making a Prediction and Visualize the result

In this Part, we make a prediction of our test set dataset and visualizing the result using the matplotlib library.

4.1 Predict the test set Result In this step, we predict our test set result.

Code:

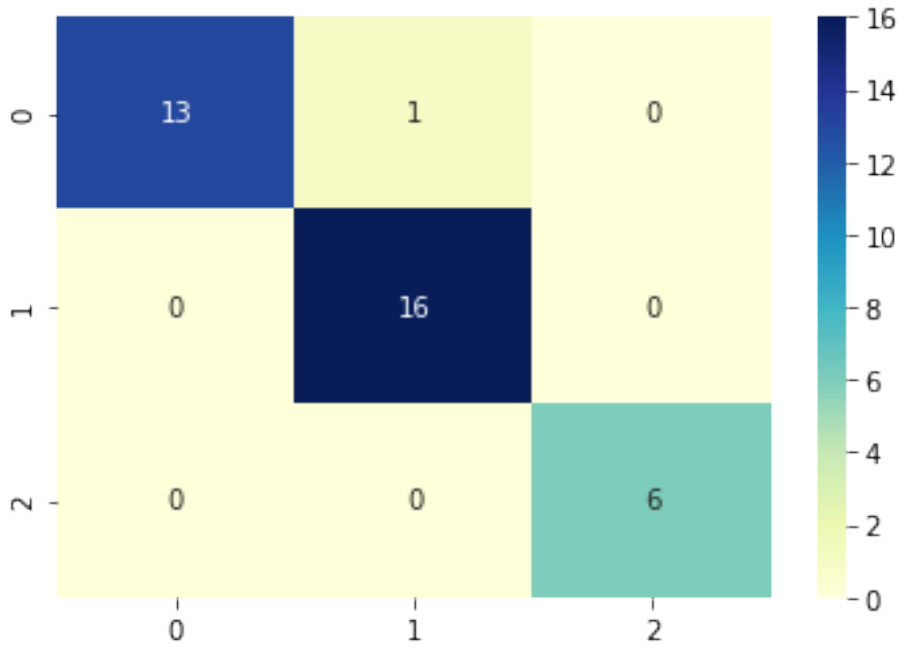
[11]:
```
# predict the Logistic regression model
y_pred=LG.predict(X_test)
```

4.2 Confusion Metric In this step we make a confusion metric of our test set result to do that we import confusion matrix from sklearn.metrics then in confusion matrix, we pass two parameters first is y_test which is the actual test set result and second is y_pred which predicted result.

Code:

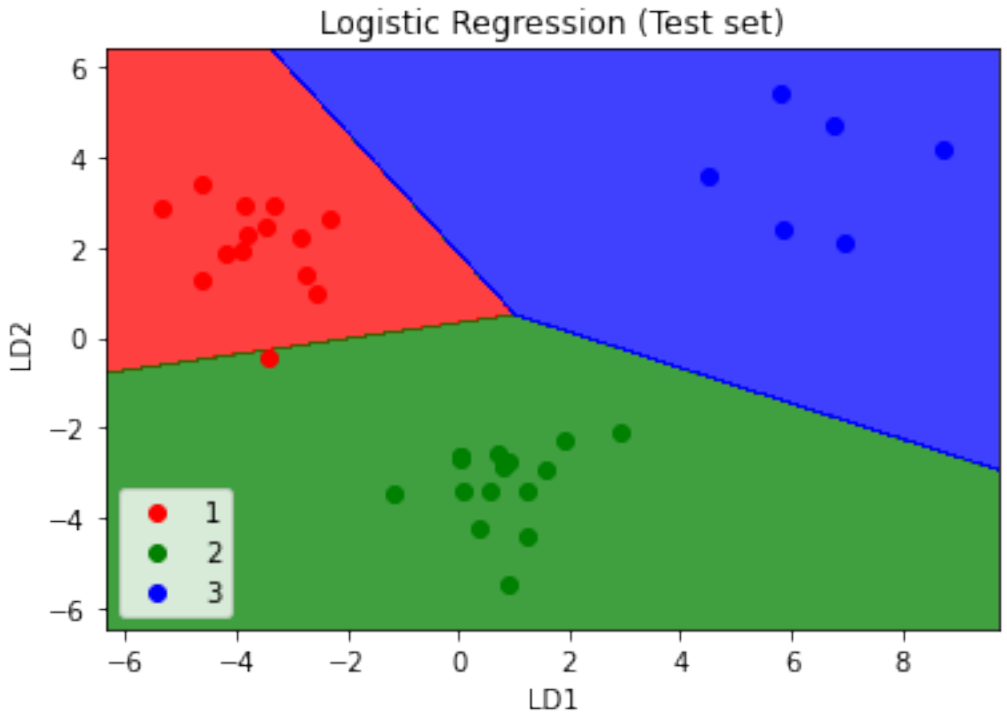
[14]:
```
# making a confusion metrics
from sklearn.metrics import confusion_matrix
confusion_matrix=confusion_matrix(y_test,y_pred)

sns.heatmap(confusion_matrix, annot=True, cmap="YlGnBu")
```

[14]: <AxesSubplot:>



4.3 Visualize our Test Set Result In this Step, we Visualize our test set result.

[13]:
```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].
 ↪max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].
 ↪max() + 1, step = 0.01))
plt.contourf(X1, X2, LG.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.
 ↪shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green', 'blue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.legend()
plt.show()
```



Source: https://medium.com/machine-learning-researcher/dimensionality-reduction-pca-and-lda-6be91734f567