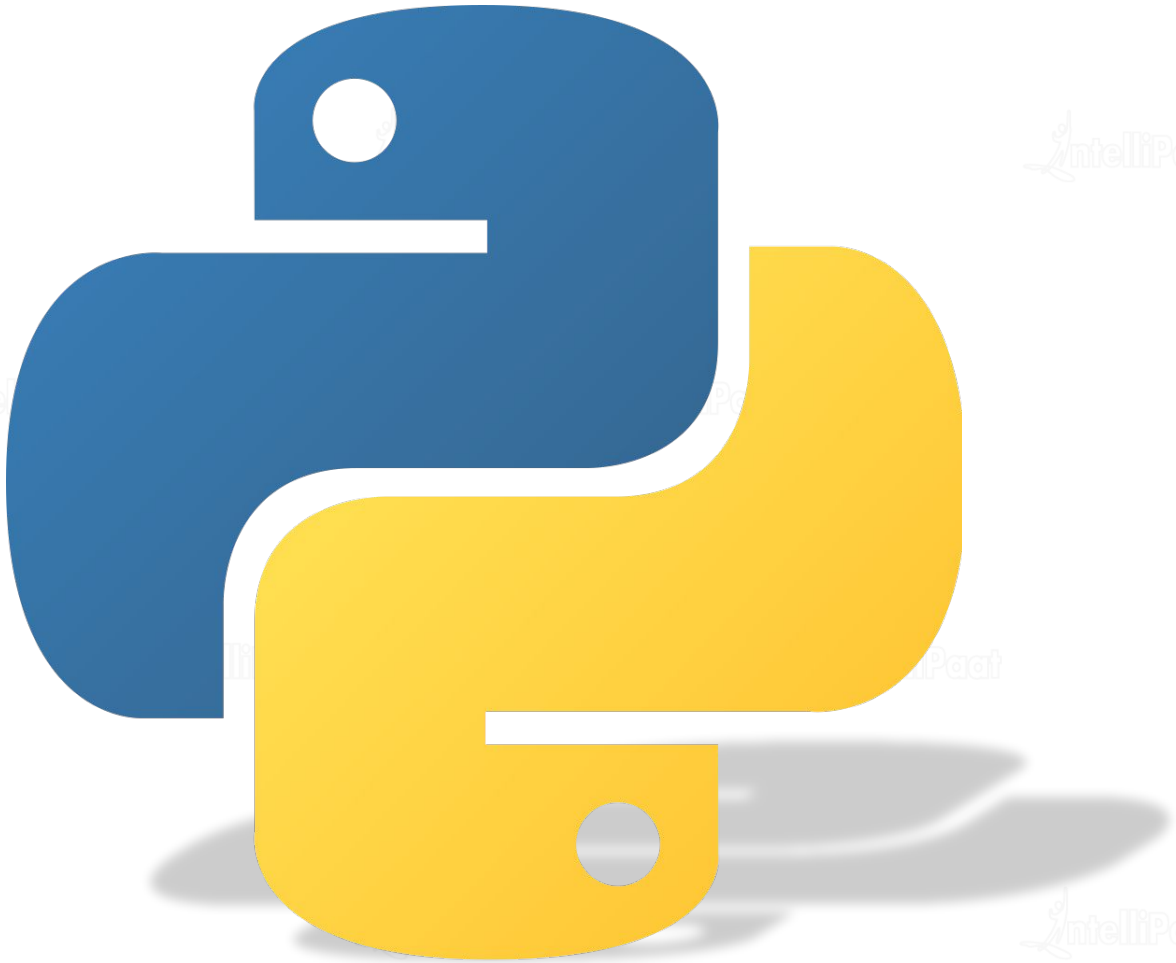
Linear Discriminant Analysis



Agenda

01 What is LDA?

02 Why LDA?

03 When to use LDA?

04 How does LDA Work?

05 Hands-on

What is LDA?

What is LDA?

LDA or Linear discriminant analysis is another way of how we can perform dimensionality reduction on a large dataset. Linear Discriminant Analysis is a supervised learning approach that uses class labels for training samples. Unlike Principal Component Analysis, the LDA focuses on maximizing the separation of the known categories in the target variables.

Why Use LDA?

Why use LDA?

Using LDA in data preprocessing reduces computation cost quite significantly.

Solves the setbacks of multi-class classification problems that are often mishandled by binary classification algorithms.

LDA makes assumptions about normally distributed classes and equal class covariance.

When to Use LDA?

When to use LDA?

Intended for classification problems where the target variables are categorical in nature

How does LDA Work?

How Does LDA Work?

Step - 1

Between-Class Variance:
The difference between the mean of different classes

Step - 2

Within-Class Variance:
Difference between mean and sample of each class

Step - 3

Construct Lower Dimensional Space:
It maximizes the between-class variance and minimizes the within-class variance.

Preparing Data for LDA

Step - 1

Output variable or target is categorical

Step - 2

Normally Distributed input variables

Step - 3

Handle Outliers in the data

Step - 4

It is a good practice to standardize the features before going for LDA.

Hands-On

Hands-on

Importing the necessary libraries

```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
```

Hands-on

Loading the dataset

```
digits = datasets.load_digits()
```

```
digits.target
```

```
array([0, 1, 2, ..., 8, 9, 8])
```

```
digits.target_names
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
digits.data.shape
```

```
(1797, 64)
```

```
X = digits.data
y = digits.target

X.shape
```

```
(1797, 64)
```

Hands-on

Splitting and standardizing the dataset

```
#splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

X_train.shape
```

```
(1437, 64)
```

Hands-on

Linear Discriminant Analysis

```
#LDA
lda = LinearDiscriminantAnalysis(n_components=9)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

X_train.shape
```

```
(1437, 9)
```

Hands-on

Building a Random Forest Model

```
#model Building
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
#prediction
y_pred = rf.predict(X_test)
```

Hands-on

Evaluating the model

```
#accuracy Score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

acc = accuracy_score(y_test, y_pred)
cf = confusion_matrix(y_test, y_pred)
clrep = classification_report(y_test, y_pred)
```

Hands-on

Evaluating the model

```
print(acc)
```

```
0.9583333333333334
```

```
print(cf)
```

```
[[32  0  0  0  1  0  0  0  0  0]
 [ 0 27  1  0  0  0  0  0  0  0]
 [ 0  0 31  2  0  0  0  0  0  0]
 [ 0  0  0 33  0  1  0  0  0  0]
 [ 0  0  0  0 45  0  0  1  0  0]
 [ 0  0  0  0  0 46  0  0  1  0]
 [ 0  0  0  0  1  0 34  0  0  0]
 [ 0  0  0  0  0  0  0 33  0  1]
 [ 0  2  0  0  0  0  0  0 27  1]
 [ 0  0  0  1  1  0  0  0  1 37]]
```

```
print(clrep)
```

	precision	recall	f1-score	support
0	1.00	0.97	0.98	33
1	0.93	0.96	0.95	28
2	0.97	0.94	0.95	33
3	0.92	0.97	0.94	34
4	0.94	0.98	0.96	46
5	0.98	0.98	0.98	47
6	1.00	0.97	0.99	35
7	0.97	0.97	0.97	34
8	0.93	0.90	0.92	30
9	0.95	0.93	0.94	40
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.96	0.96	0.96	360