ml1

November 3, 2024

1 Feature Transformation: Apply LDA Algorithm on Iris Dataset and classify which species a given flower belongs to. Dataset Link: https://www.kaggle.com/datasets/uciml/iris

[7]:
```
import numpy as  np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import␣
 ↪confusion_matrix,accuracy_score,classification_report
```

[8]:
```
df=pd.read_csv(r"C:\Users\dell\Desktop\DMV and ML\ML Datasets\Iris.csv")
```

[9]:
```
df
```

[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	\
0	1	5.1	3.5	1.4	0.2	
1	2	4.9	3.0	1.4	0.2	
2	3	4.7	3.2	1.3	0.2	
3	4	4.6	3.1	1.5	0.2	
4	5	5.0	3.6	1.4	0.2	
..	…	…	…	…	…	
145	146	6.7	3.0	5.2	2.3	
146	147	6.3	2.5	5.0	1.9	
147	148	6.5	3.0	5.2	2.0	
148	149	6.2	3.4	5.4	2.3	
149	150	5.9	3.0	5.1	1.8	

	Class Label
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa

```
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[10]:
```
df.describe()
```

[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

[11]:
```
# x=df.iloc[:,:-1] # All columns except the last one
x=df.drop(["Class Label"],axis=1)
y=df["Class Label"]
```

[12]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪2,random_state=100)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[12]:
```
((120, 5), (30, 5), (120,), (30,))
```

[13]:
```
sc= StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

[14]:
```
lda= LinearDiscriminantAnalysis()
x_train= lda.fit_transform(x_train,y_train)
x_test=lda.transform(x_test)
```

[15]:
```
x_train.shape, x_test.shape
```

[15]:
```
((120, 2), (30, 2))
```

[16]:
```
model = LogisticRegression()
model.fit(x_train,y_train)
```

[16]:
```
LogisticRegression()
```

[17]:
```
y_pred=model.predict(x_test)
```

[18]:
```
accuracy_score(y_test , y_pred)
```

[18]:
```
1.0
```

[19]:
```
confusion_matrix(y_test , y_pred)
```

[19]:
```
array([[11,  0,  0],
       [ 0,  6,  0],
       [ 0,  0, 13]], dtype=int64)
```

[20]:
```
print(classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00         6
 Iris-virginica       1.00      1.00      1.00        13

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

The StandardScaler is a tool from the sklearn.preprocessing module in Python that standardizes features by removing the mean and scaling to unit variance. It transforms data so that it has a mean of 0 and a standard deviation of 1. This is commonly done to ensure features contribute equally to a model, especially when they vary in scale.

StandardScaler() will normalize the features i.e. each column of X, INDIVIDUALLY, so that each column/feature/variable will have $= 0$ and $= 1$.

PCA is unsupervised and finds directions of maximum variance. Useful for visualization and compression. LDA is supervised and finds directions of maximum separation between classes. Useful for classification and pattern recognition.

• useful when the dataset has multiple classes and the goal is to find a linear combination of features that best separates the classes. • LDA works by projecting the data onto a lower-dimensional space that maximizes the separation between the classes. • Maximizes class separability while minimizing variance within classes • LDA finds the direction that best separates the two classes.