Profundiza más

Recurso de Profundización

Clase 7 - Reducción de dimensionalidad con análisis de componentes principales en *scikit-learn*

El siguiente código se divide en varias partes, primero cargamos las librerías, luego el *dataset*, después normalizamos los datos y aplicamos los componentes principales sobre los datos normalizados, eso nos crea un nuevo *dataset,* que contiene los valores de los componentes principales asociados a cada uno de los elementos del *dataset.*

```
# PCA
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
(X_cancer, y_cancer) = load_breast_cancer(return_X_y = True)

# each feature should be centered (zero mean) and with unit variance
X_normalized = StandardScaler().fit(X_cancer).transform(X_cancer)

pca = PCA(n_components = 2).fit(X_normalized)

X_pca = pca.transform(X_normalized)
print(X_cancer.shape, X_pca.shape)
```

```
(569, 30) (569, 2)
```

Implementación de *DBSCAN* en *scikit–learn*

El siguiente código está dividido en las siguientes partes. Primero cargamos las librerías, luego instanciamos los datos, posteriormente creamos una instancia de DBSCAN, luego aplicamos la instancia y visualizamos el resultado

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import DBSCAN

# 1. Generamos datos sintéticos con make_blobs
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.6, random_state=42)

# 2. Creamos la instancia del modelo DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)

# 3. Aplicamos el modelo a los datos
labels = dbscan.fit_predict(X)

# 4. Graficamos los clusters
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', edgecolors='k', s=50)
plt.title("Clustering con DBSCAN")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.colorbar(label="Cluster Label")
plt.show()
```