

Cycle-4

Aim: Dimensionality Reduction using Principal Component (PCA)

Analysis

Solution:

Dimensionality Reduction is a statistical/ML-based technique wherein we try to reduce the number of features in our dataset and obtain a dataset with an optimal number of dimensions.

One of the most common ways to accomplish Dimensionality Reduction is Feature Extraction, wherein we reduce the number of dimensions by mapping a higher dimensional feature space to a lower-dimensional feature space. The most popular technique of Feature Extraction is Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

Principal Component Analysis is a technique of feature extraction that maps a higher dimensional feature space to a lower-dimensional feature space. While reducing the number of dimensions, PCA ensures that maximum information of the original dataset is retained in the dataset with the reduced no. of dimensions and the correlation between the newly obtained Principal Components is minimum. The new features obtained after applying PCA are called Principal Components and are denoted as ***PC_i*** ($i=1,2,3\dots n$). Here, (Principal Component-1) PC1 captures the maximum information of the original dataset, followed by PC2, then PC3 and so on.

Source Code:

Step-1: Import necessary libraries

All the necessary libraries required to load the dataset, pre-process it and then apply PCA on it are mentioned below:

```
[27] # Import necessary libraries
      from sklearn import datasets # to retrieve the iris Dataset
      import pandas as pd # to load the dataframe
      from sklearn.preprocessing import StandardScaler # to standardize the features
      from sklearn.decomposition import PCA # to apply PCA
      import seaborn as sns # to plot the heat maps
```

Step-2: Load the dataset

After importing all the necessary libraries, we need to load the dataset. Now, the iris dataset is already present in sklearn. First, we will load it and then convert it into a pandas data frame for ease of use.

```

▶ #Load the Dataset
iris = datasets.load_iris()
#convert the dataset into a pandas data frame
df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
#display the head (first 5 rows) of the dataset
df.head()

```

Output:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Step 3: Standardize the Data

PCA is affected by scale, so you need to scale the features in your data before applying PCA. Use StandardScaler to help you standardize the data set's features onto unit scale (mean = 0 and variance = 1), which is a requirement for the optimal performance of many machine learning algorithms.

```

▶ #Standardize the features
#Create an object of StandardScaler which is present in sklearn.preprocessing
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit_transform(df)) #scaling the data
scaled_data.head()

```

Output:

	0	1	2	3
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

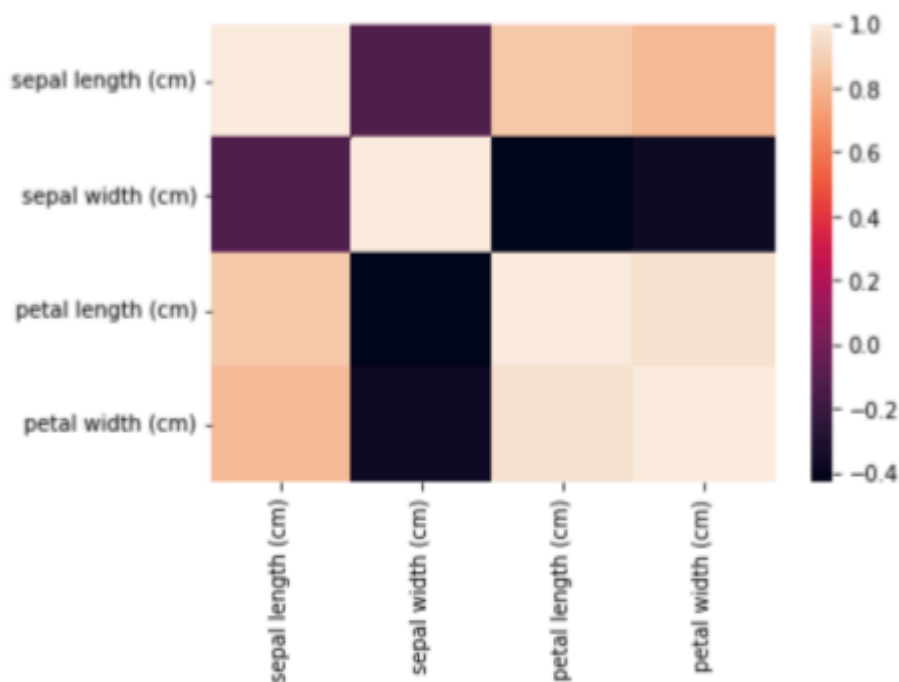
Step-4: Check the Co-relation between features without PCA (Optional)

Now, we will check the co-relation between our scaled dataset using a heat map. The correlation between various features is given by the `corr()` function and then the heat map is plotted by the `heatmap()` function. The colour scale on the side of the heatmap helps determine the magnitude of the co-relation.



```
#Check the Co-relation between features without PCA  
sns.heatmap(scaled_data.corr())
```

Output:



We can clearly see that a darker shade represents less co-relation while a lighter shade represents more co-relation. The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

Step-5: Applying Principal Component Analysis

Python offers yet another in-built class called PCA which is present in *sklearn.decomposition*, which we have already imported in step-1. We need to create an object of PCA and while doing so we also need to initialize `n_components` – which is the number of principal components we want in our final dataset. Here, we have taken `n_components = 3`, which means our final feature set will have 3 columns. We fit our scaled data to the PCA object which gives us our reduced dataset.

```

▶ #Applying PCA
#Taking no. of Principal Components as 3
pca = PCA(n_components = 3)
pca.fit(scaled_data)
data_pca = pca.transform(scaled_data)
data_pca = pd.DataFrame(data_pca, columns=['PC1', 'PC2', 'PC3'])
data_pca.head()

```

Output:

	PC1	PC2	PC3
0	-2.264703	0.480027	-0.127706
1	-2.080961	-0.674134	-0.234609
2	-2.364229	-0.341908	0.044201
3	-2.299384	-0.597395	0.091290
4	-2.389842	0.646835	0.015738

Step-6: Checking Co-relation between features after PCA

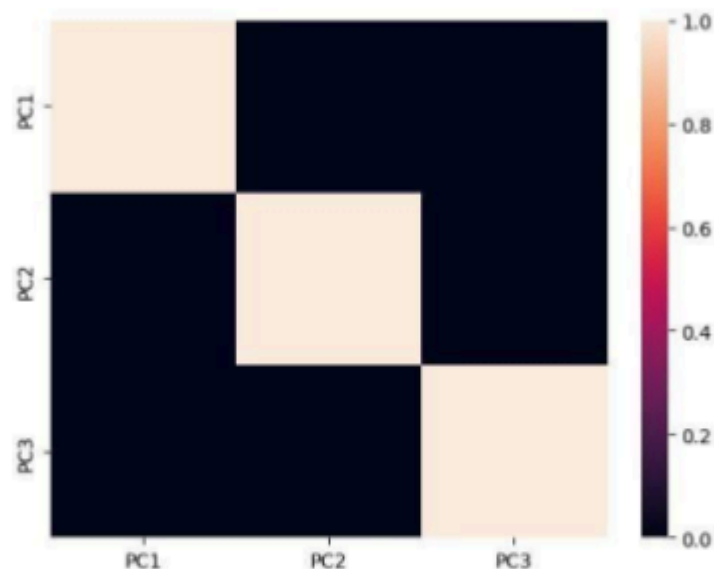
Now that we have applied PCA and obtained the reduced feature set, we will check the co-relation between various Principal Components, again by using a heatmap.

```

▶ #Checking Co-relation between features after PCA
sns.heatmap(data_pca.corr())

```

Output:



The above heatmap clearly depicts that there is no correlation between various obtained principal components (PC1, PC2, and PC3). Thus, we have moved from higher dimensional feature space to a lower-dimensional feature space while ensuring that there is no correlation between the so obtained PCs.