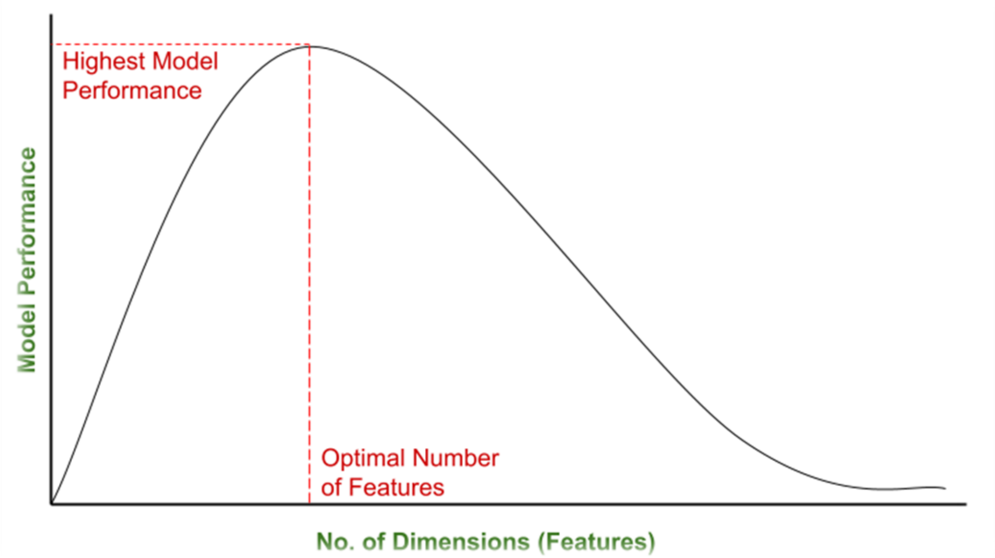
WEEK_9_DAY_3_AFTER_NOON

Dimensionality Reduction using PCA in python



The above graph represents the change in model performance with the increase in the number of dimensions of the dataset. It can be observed that the model performance is best only at an option dimension, beyond which it starts decreasing.

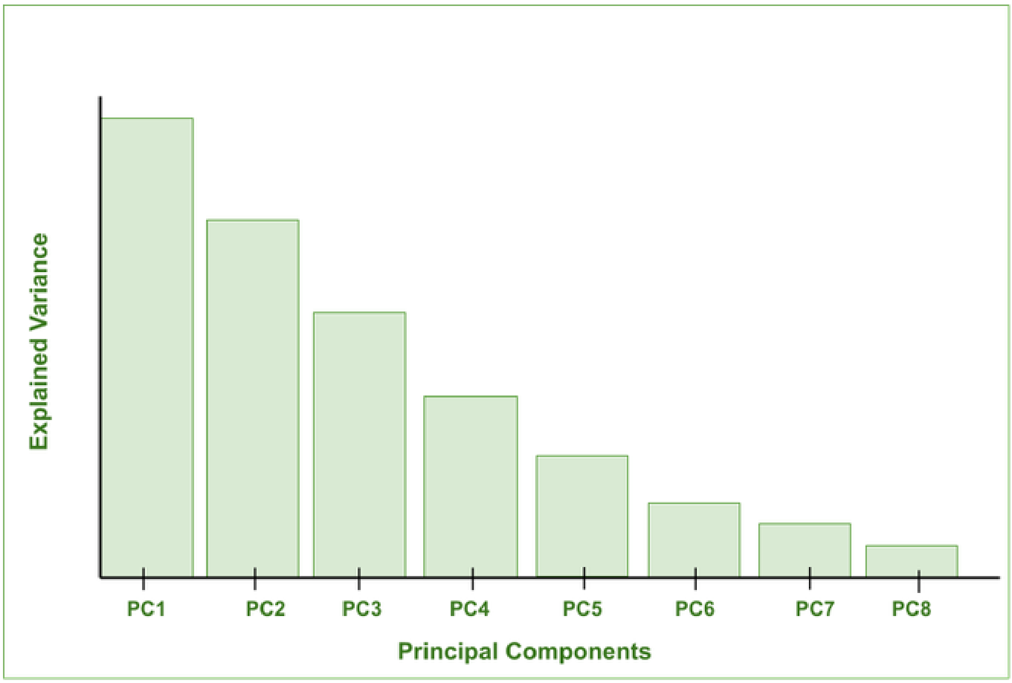
One of the most common ways to accomplish Dimensionality Reduction is Feature Extraction, where the number of dimensions is reduced by mapping a higher dimensional feature space to a lower-dimensional feature space.

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.

PCA ensures that maximum information of the original dataset is retained in the dataset with the reduced no. of dimensions and the co-relation between the newly obtained Principal Components is minimum.

The new features obtained after applying PCA are called Principal Components and are denoted as ***PCi (i=1,2,3…n)***. Here, (Principal Component-1) PC1 captures the maximum information of the original dataset, followed by PC2, then PC3 and so on.



The above bar graph depicts the amount of Explained Variance captured by various Principal Components. (The Explained Variance defines the amount of information captured by the Principal Components).

Steps to Apply PCA in Python for Dimensionality Reduction

Step-1: Import necessary libraries

```
# 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

```
#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()
```

	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 features

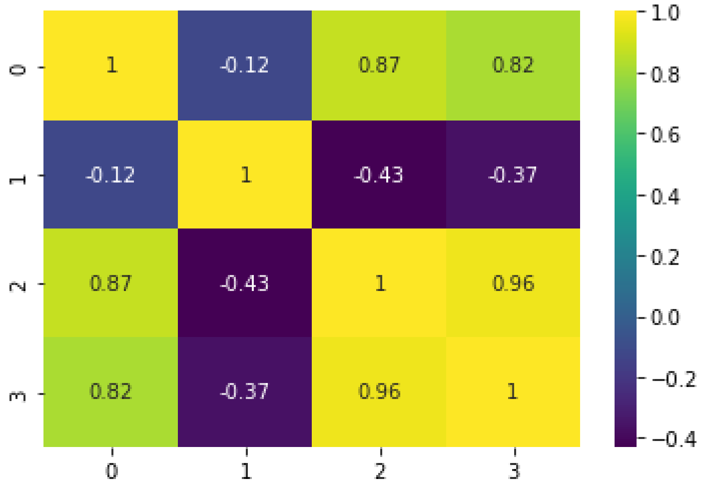
```
#Standardize the features
#Create an object of StandardScaler which is present in sklearn.preprocessing
scalar = StandardScaler()
scaled_data= pd.DataFrame(scalar.fit_transform(df)) #scaling the data
scaled_data
```

	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
...	...	...	...	...
145	1.038005	-0.131979	0.819596	1.448832
146	0.553333	-1.282963	0.705921	0.922303
147	0.795669	-0.131979	0.819596	1.053935
148	0.432165	0.788808	0.933271	1.448832
149	0.068662	-0.131979	0.762758	0.790671

150 rows × 4 columns

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

```
#Check the Co-relation between features without PCA
sns.heatmap(scaled_data.corr(),cmap='viridis',annot=True)
```



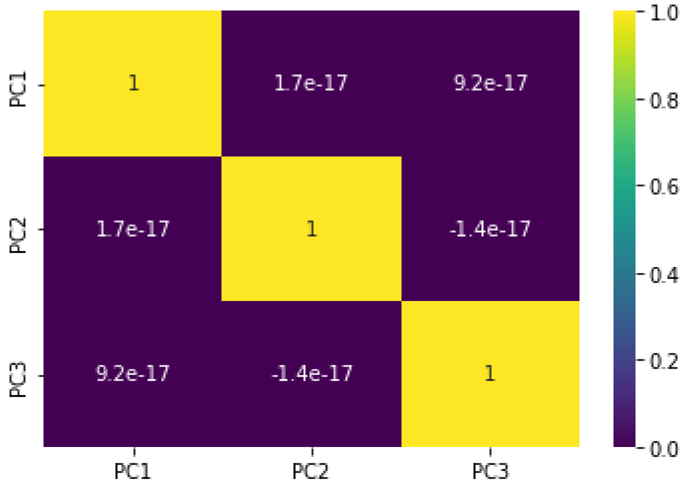
Step-5: Applying Principal Component Analysis

```
#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()
```

	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

```
#Checking Co-relation between features after PCA
sns.heatmap(data_pca.corr(),cmap='viridis',annot=True)
```


1
1.7e-17
9.2e-17
1.7e-17
1
-1.4e-17
9.2e-17
-1.4e-17
1
PC1
PC2
PC3
1.0
0.8
0.6
0.4
0.2
0.0