Chapter 16: Demo PCA Visualization

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter16_PCA/'
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

In [3]:
```
# Load dataset into Pandas DataFrame
df = pd.read_excel("Iris.xls")
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepallength    150 non-null float64
sepalwidth     150 non-null float64
petallength    150 non-null float64
petalwidth     150 non-null float64
iris           150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
df.head()
```

Out[5]:

	sepallength	sepalwidth	petallength	petalwidth	iris
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Standardize the Data

Since PCA yields a feature subspace that maximizes the variance along the axes, it makes sense to standardize the data, especially, if it was measured on different scales. Although, all features in the Iris dataset were measured in centimeters, let us continue with the transformation of the data onto unit scale (mean=0 and variance=1), which is a requirement for the optimal performance of many machine learning algorithms.

In [7]:
```
features = ['sepallength', 'sepalwidth', 'petallength', 'petalwidth']
x = df.loc[:, features].values
```

In [8]:
```
y = df.loc[:,['iris']].values
```

In [9]:
```
# Scaler
x = StandardScaler().fit_transform(x)
```

In [10]:
```
pd.DataFrame(data = x, columns = features).head(3)
```

Out[10]:

	sepallength	sepalwidth	petallength	petalwidth
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977

In [11]:
```
pca = PCA(n_components=2) # trực quan
```

In [12]:
```
principalComponents = pca.fit_transform(x)
```

In [13]:
```
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1',
                          'principal component 2'])
```

In [14]:
```
principalDf.head(5)
```

Out[14]:

	principal component 1	principal component 2
0	-2.264542	0.505704
1	-2.086426	-0.655405
2	-2.367950	-0.318477
3	-2.304197	-0.575368
4	-2.388777	0.674767

In [15]:
```
df[['iris']].head()
```

Out[15]:

	iris
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa

In [16]:
```
finalDf = pd.concat([principalDf, df[['iris']]], axis = 1)
finalDf.head(5)
```

Out[16]:

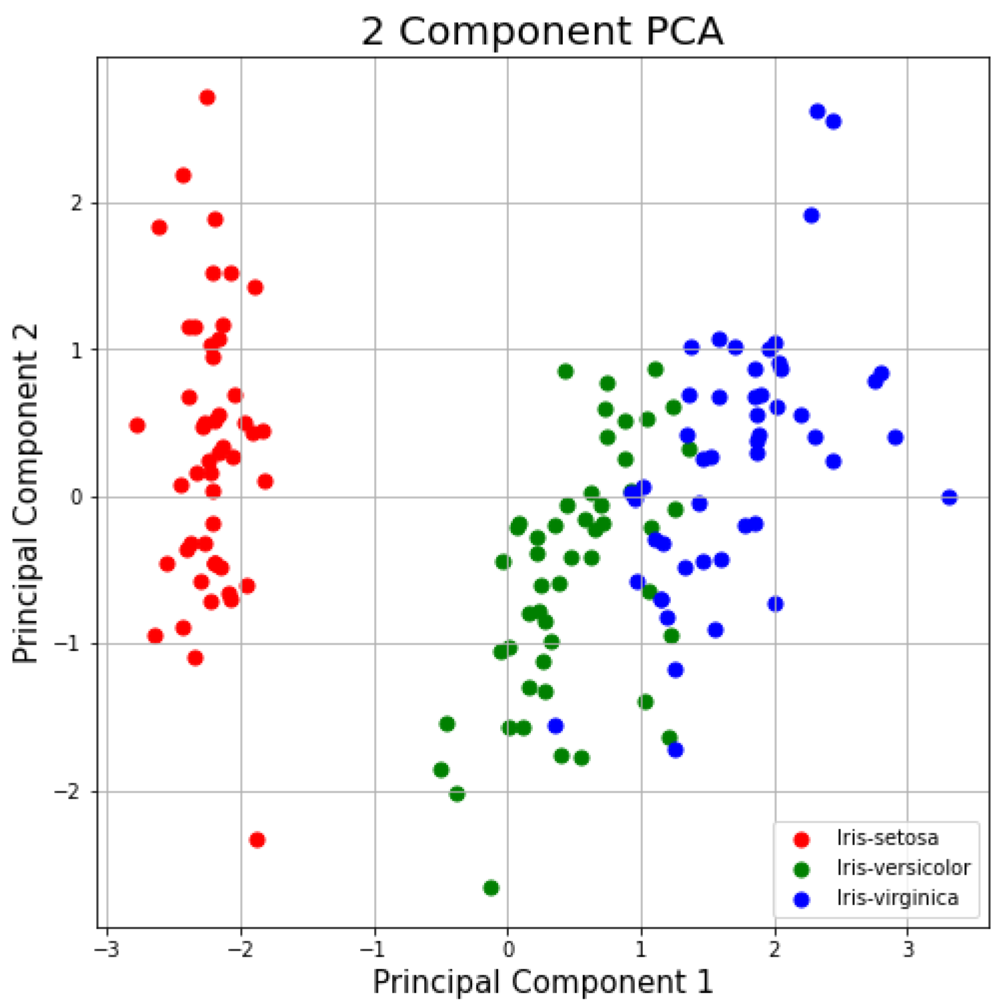
	principal component 1	principal component 2	iris
0	-2.264542	0.505704	Iris-setosa
1	-2.086426	-0.655405	Iris-setosa
2	-2.367950	-0.318477	Iris-setosa
3	-2.304197	-0.575368	Iris-setosa
4	-2.388777	0.674767	Iris-setosa

Visualize 2D Projection

Use a PCA projection to 2d to visualize the entire data set. You should plot different classes using different colors or shapes. Do the classes seem well-separated from each other?

In [17]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 Component PCA', fontsize = 20)


targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['iris'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



Explained Variance

The explained variance tells us how much information (variance) can be attributed to each of the principal components.

In [18]:
```
pca.explained_variance_ratio_
```

Out[18]:
```
array([0.72770452, 0.23030523])
```

Together, the first two principal components contain 95.80% of the information. The first principal component contains 72.77% of the variance and the second principal component contains 23.03% of the variance. The third and fourth principal component contained the rest of the variance of the dataset.

In [ ]:
```

```