**Principal Component Analysis**

* **Unsupervised Learning + Dimensionality Reduction**

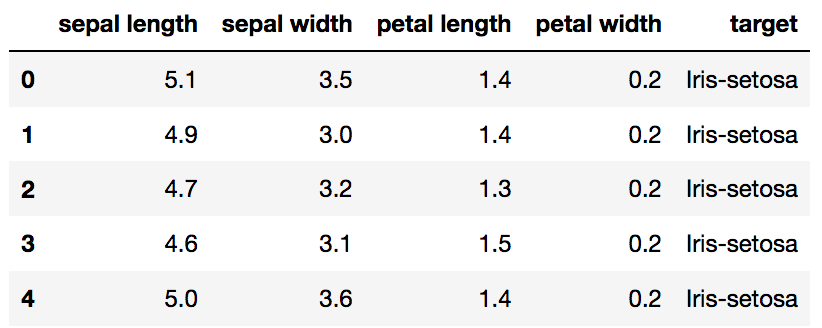
**PCA for Data Visualization**

For a lot of machine learning applications it helps to be able to visualize your data. Visualizing 2 or 3 dimensional data is not that challenging. However, even the Iris dataset used in this part of the tutorial is 4 dimensional. You can use PCA to reduce that 4 dimensional data into 2 or 3 dimensions so that you can plot and hopefully understand the data better.

**Load Iris Dataset**

The Iris dataset is one of datasets scikit-learn comes with that do not require the downloading of any file from some external website.

*load dataset into Pandas DataFrame  
df = pd.read\_csv(url, names=['sepal length','sepal width','petal length','petal width','target'])*

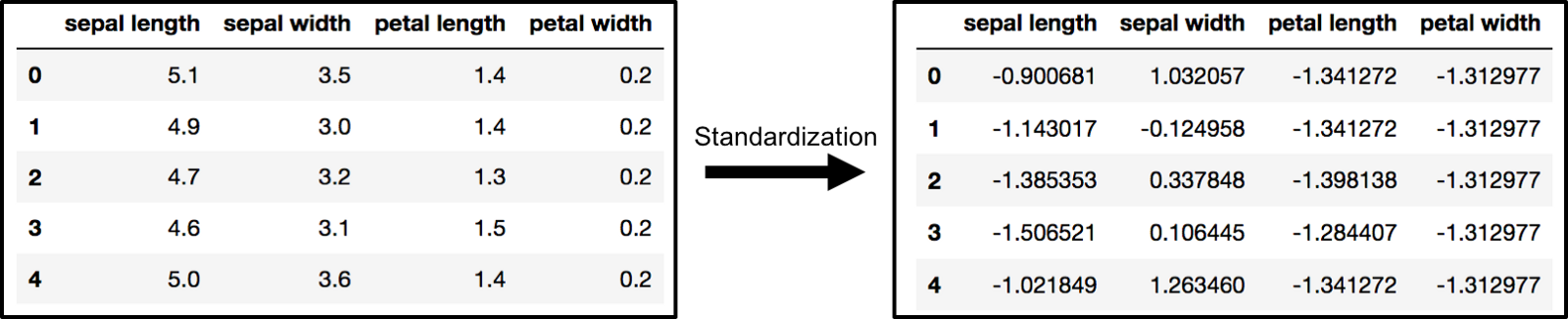


Original Pandas df (features + target)

**Standardize the Data**

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

*from sklearn.preprocessing import StandardScalerfeatures = ['sepal length', 'sepal width', 'petal length', 'petal width']# Separating out the features  
x = df.loc[:, features].values# Separating out the target  
y = df.loc[:,['target']].values# Standardizing the features  
x = StandardScaler().fit\_transform(x)*



The array **x** (visualized by a pandas dataframe) before and after standardization

**PCA Projection to 2D**

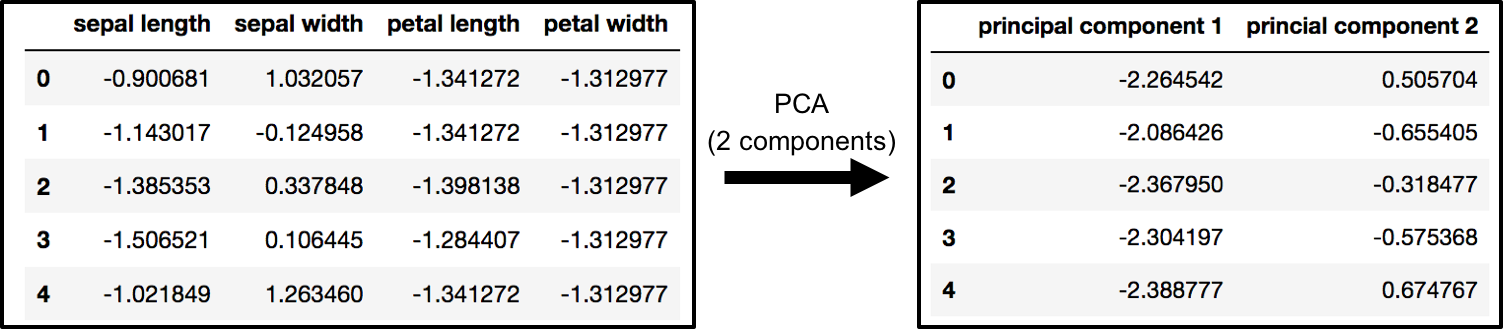
The original data has 4 columns (sepal length, sepal width, petal length, and petal width). In this section, the code projects the original data which is 4 dimensional into 2 dimensions. I should note that after dimensionality reduction, there usually isn’t a particular meaning assigned to each principal component. The new components are just the two main dimensions of variation.

*from sklearn.decomposition*

*import PCApca = PCA(n\_components=2)*

*principalComponents = pca.fit\_transform(x)*

*principalDf = pd.DataFrame(data = principalComponents  
 , columns = ['principal component 1', 'principal component 2'])*



PCA and Keeping the Top 2 Principal Components

*finalDf = pd.concat([principalDf, df[['target']]], axis = 1)*

*Concatenating DataFrame along axis = 1. finalDf is the final DataFrame before plotting the data.*

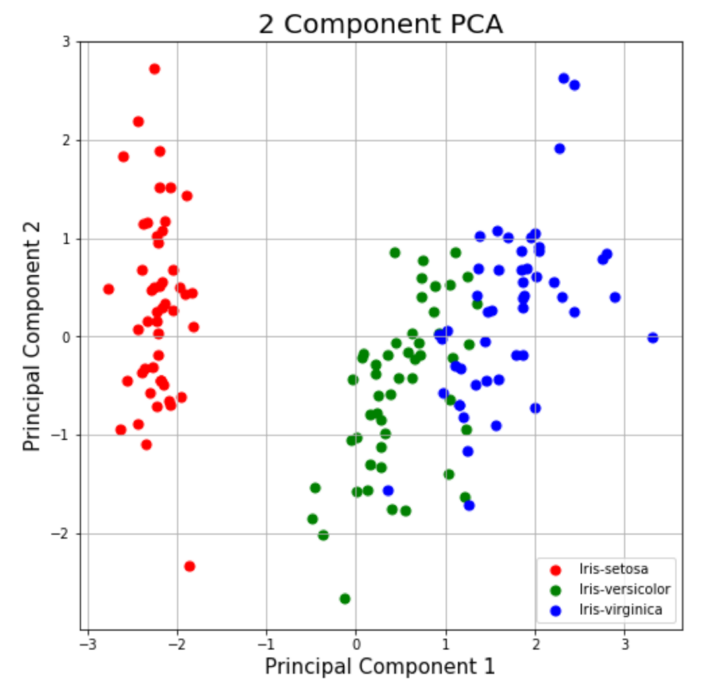


Concatenating dataframes along columns to make finalDf before graphing

**Visualize 2D Projection**

This section is just plotting 2 dimensional data. Notice on the graph below that the classes seem well separated from each other.

*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['target'] == target  
 ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']  
 , finalDf.loc[indicesToKeep, 'principal component 2']  
 , c = color  
 , s = 50)  
ax.legend(targets)  
ax.grid()*



**2 Component PCA Graph**

**Explained Variance**

The explained variance tells you how much information (variance) can be attributed to each of the principal components. This is important as while you can convert 4 dimensional space to 2 dimensional space, you lose some of the variance (information) when you do this. By using the attribute **explained\_variance\_ratio\_**, you can see that the first principal component contains 72.77% of the variance and the second principal component contains 23.03% of the variance. Together, the two components contain 95.80% of the information.

pca.explained\_variance\_ratio\_