PCA(Principal Component Analysis)

AcadView

June 7, 2018

1 Overview

In real world data analysis tasks we analyze complex data i.e. multi dimensional data. We plot the data and find various patterns in it or use it to train some machine learning models. One way to think about dimensions is that suppose you have an data point x , if we consider this data point as a physical object then dimensions are merely a basis of view, like where is the data located when it is observed from horizontal axis or vertical axis.

As the dimensions of data increases, the difficulty to visualize it and perform computations on it also increases. So, how to reduce the dimensions of a data:

- Remove the redundant dimensions
- Only keep the most important dimensions

Principal Component Analysis (PCA) is a dimensionality-reduction technique that is often used to transform a high-dimensional dataset into a smaller-dimensional subspace prior to running a machine learning algorithm on the data.

2 When should you use PCA?

It is often helpful to use a dimensionality-reduction technique such as PCA prior to performing machine learning because:

- Reducing the dimensionality of the dataset reduces the size of the space on which k-nearest-neighbors (kNN) must calculate distance, which improve the performance of kNN.
- Reducing the dimensionality of the dataset reduces the number of degrees of freedom of the hypothesis, which reduces the risk of overfitting.
- Most algorithms will run significantly faster if they have fewer dimensions they need to look at.

• Reducing the dimensionality via PCA can simplify the dataset, facilitating description, visualization, and insight.

First try to understand some terms :

• **Variance :** It is a measure of the variability or it simply measures how spread the data set is. Mathematically, it is the average squared deviation from the mean score. We use the following formula to compute variance var(x).

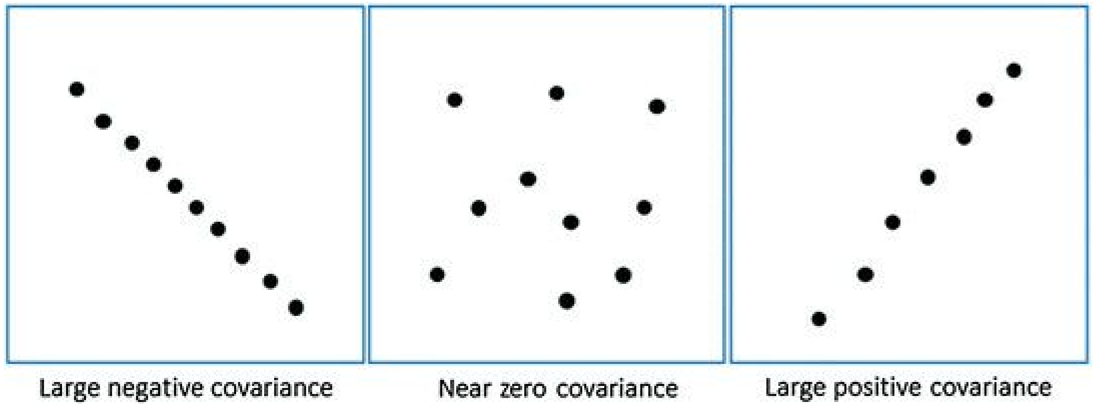
$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N} \qquad cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

• **Covariance :** It is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction. Formula is shown above denoted by *cov(x,y)* as the covariance of *x* and *y*.

Here, *xi* is the value of x in *ith* dimension.

x bar and *y bar* denote the corresponding mean values.

One way to observe the covariance is how interrelated two data sets are.

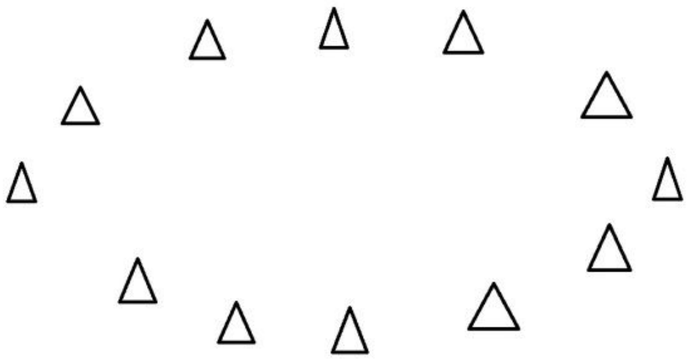


Positive covariance means X and Y are positively related i.e. as X increases Y also increases. Negative covariance depicts the exact opposite relation. However zero covariance means X and Y are not related.

3 What does PCA do?

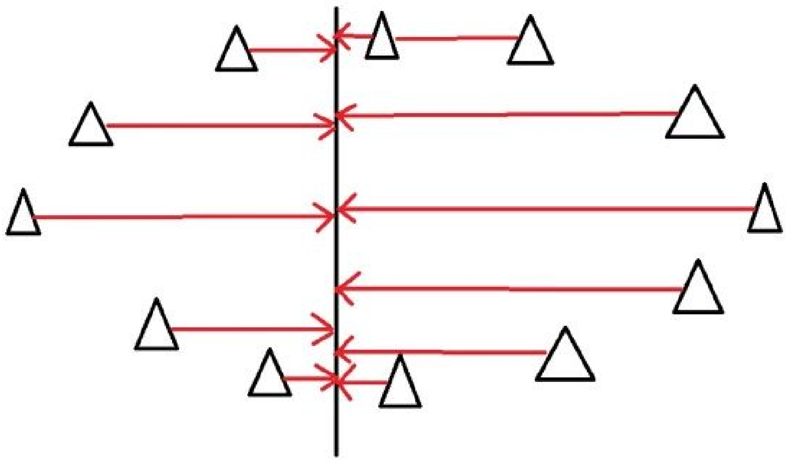
Principal Component Analysis does just what it advertises; it finds the principal components of the dataset. PCA transforms the data into a new, lower-dimensional subspace into a new coordinate system. In the new coordinate system, the first axis corresponds to the first principal component, which is the component that explains the greatest amount of the variance in the data.

It is often useful to measure data in terms of its principal components rather than on a normal x-y axis. So what are principal components then? They are the underlying structure in the data. They are the directions where there is the most variance, the directions where the data is most spread out. This is easiest to explain by way of example. Here's some triangles in the shape of an oval:



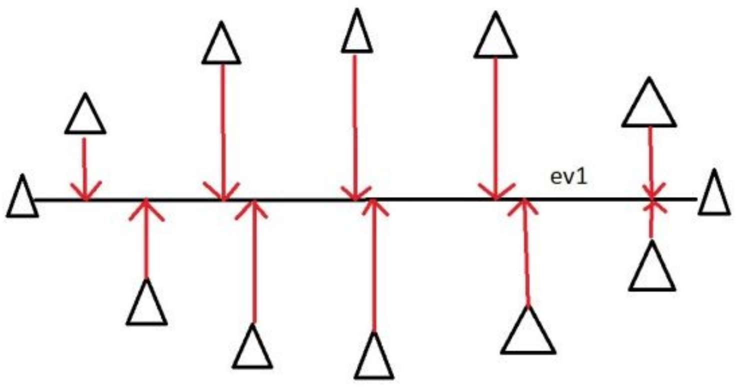
Imagine that the triangles are points of data. To find the direction where there is most variance, find the straight line where the data is most spread out when projected onto it.

A vertical straight line with the points projected on to it will look like this:



The data isn't very spread out here, therefore it doesn't have a large variance. It is probably not the principal component.

A horizontal line are with lines projected on will look like this:



On this line the data is way more spread out, it has a large variance. In fact there isn't a straight line you can draw that has a larger variance than a horizontal one. A horizontal line is therefore the principal component in this example.

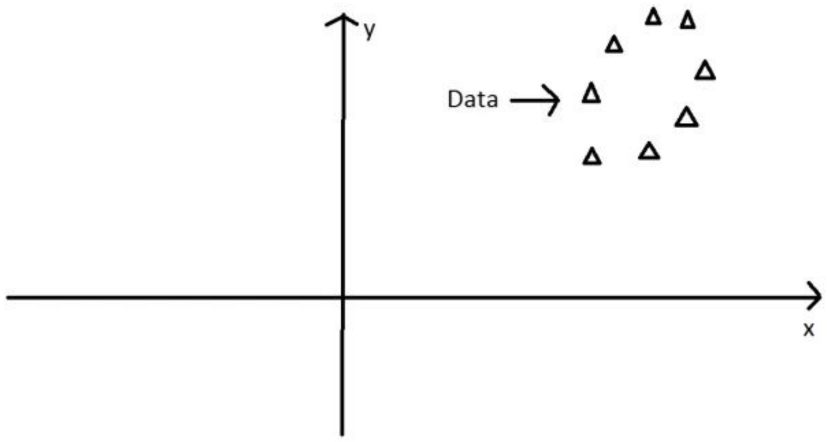
Luckily we can use maths to find the principal component rather than drawing lines and unevenly shaped triangles. This is where eigenvectors and eigenvalues come in.

4 Eigenvectors and Eigenvalues(Optional)

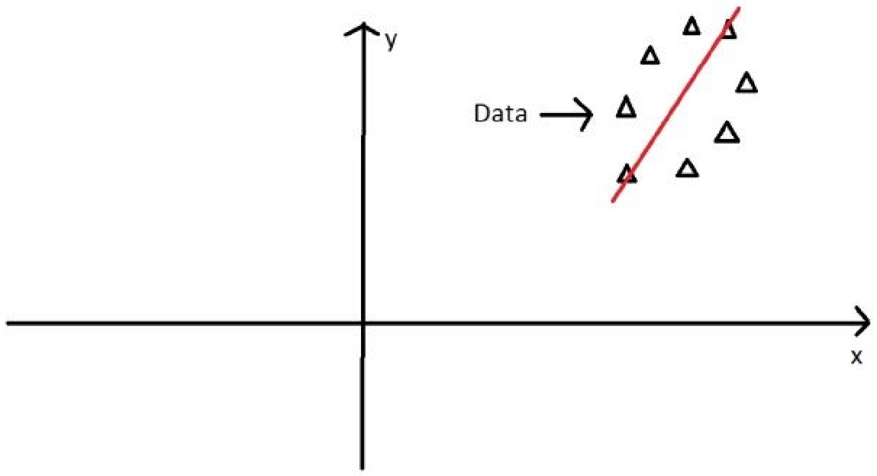
When we get a set of data points, like the triangles above, we can deconstruct the set into eigenvectors and eigenvalues. Eigenvectors and values exist in pairs: every eigenvector has a corresponding eigenvalue. An eigenvector is a direction, in the example above the eigenvector was the direction of the line (vertical, horizontal, 45 degrees etc.) . An eigenvalue is a number, telling you how much variance there is in the data in that direction, in the example above the eigenvalue is a number telling us how spread out the data is on the line. The eigenvector with the highest eigenvalue is therefore the principal component.

Okay, so even though in the last example we could point my line in any direction, it turns out there are not many eigenvectors/values in a data set. In fact the amount of eigenvectors/values that exist equals the number of dimensions the data set has. Say we are measuring age and hours on the internet. there are 2 variables, its a 2 dimensional data set, therefore there are 2 eigenvectors/values. If I'm measuring age, hours on internet and hours on mobile phone there's 3 variables, 3-D data set, so 3 eigenvectors/values. The reason for this is that eigenvectors put the data into a new set of dimensions, and these new dimensions have to be equal to the original amount of dimensions. This sounds complicated, but again an example should make it clear.

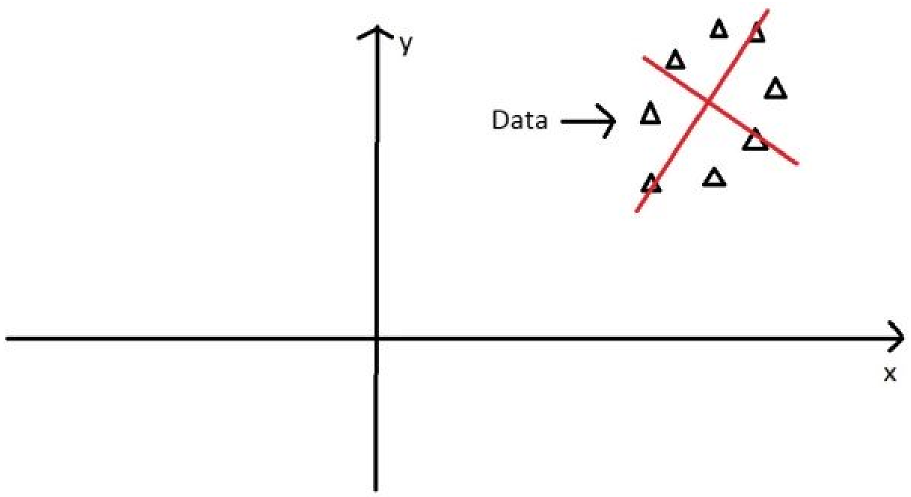
Here's a graph with the oval:



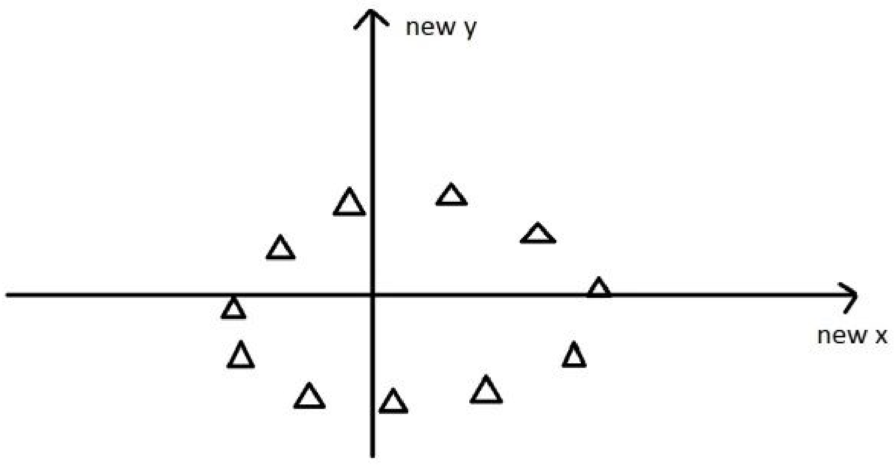
At the moment the oval is on an x-y axis. x could be age and y hours on the internet. These are the two dimensions that my data set is currently being measured in. Now remember that the principal component of the oval was a line splitting it longways:



It turns out the other eigenvector (remember there are only two of them as its a 2-D problem) is perpendicular to the principal component. As we said, the eigenvectors have to be able to span the whole x-y area, in order to do this (most effectively), the two directions need to be orthogonal (i.e. 90 degrees) to one another. This why the x and y axis are orthogonal to each other in the first place. It would be really awkward if the y axis was at 45 degrees to the x axis. So the second eigenvector would look like this:



The eigenvectors have given us a much more useful axis to frame the data in. We can now re-frame the data in these new dimensions. It would look like this:

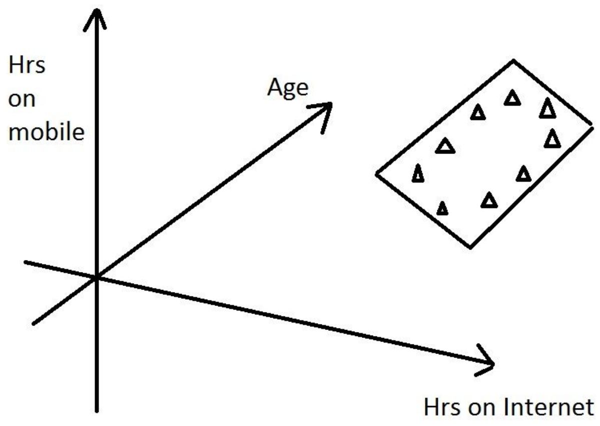


Note that nothing has been done to the data itself. Were just looking at it from a different angle. So getting the eigenvectors gets you from one set of axes to another. These axes are much more intuitive to the shape of the data now. These directions are where there is most variation, and that is where there is more information (think about this the reverse way round. If there was no variation in the data [e.g. everything was equal to 1] there would be no information, it's a very boring statistic in this scenario the eigenvalue for that dimension would equal zero, because there is no variation).

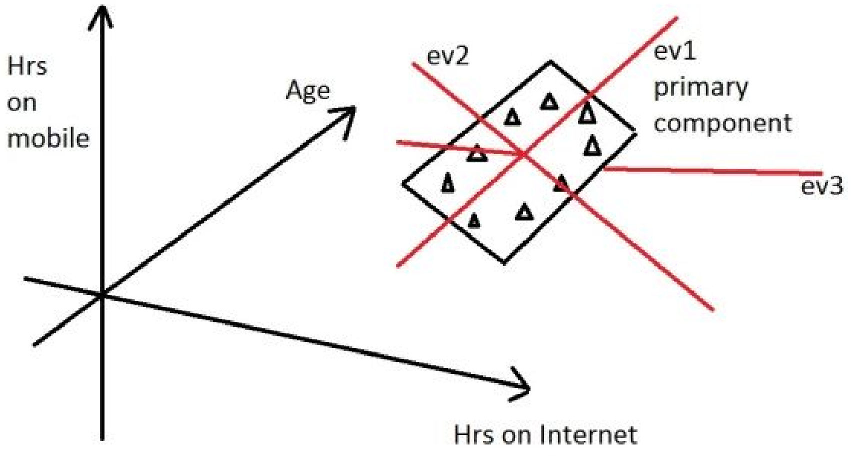
5 Dimension Reduction

PCA can be used to reduce the dimensions of a data set. Dimension reduction is analogous to being philosophically reductionist: It reduces the data down into its basic components, stripping away any unnecessary parts.

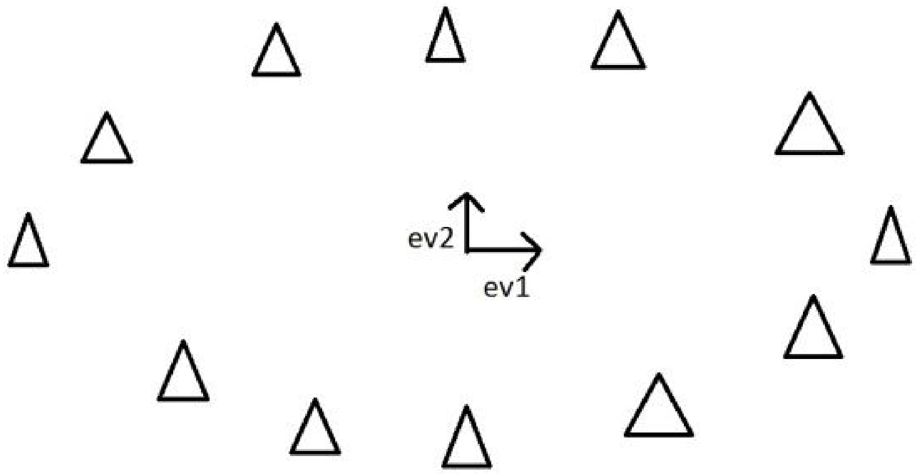
Let's say you are measuring three things: age, hours on internet and hours on mobile. There are 3 variables so it is a 3D data set. 3 dimensions is an x,y and z graph, It measure width, depth and height (like the dimensions in the real world). Now imagine that the data forms into an oval like the ones above, but that this oval is on a plane. i.e. all the data points lie on a piece of paper within this 3D graph (having width and depth, but no height). Like this:



When we find the 3 eigenvectors/values of the data set (remember 3D problem = 3 eigenvectors), 2 of the eigenvectors will have large eigenvalues, and one of the eigenvectors will have an eigenvalue of zero. The first two eigenvectors will show the width and depth of the data, but because there is no height on the data (it is on a piece of paper) the third eigenvalue will be zero. On the picture below ev1 is the first eigenvector (the one with the biggest eigenvalue, the principal component), ev2 is the second eigenvector (which has a non-zero eigenvalue) and ev3 is the third eigenvector, which has an eigenvalue of zero.



We can now rearrange our axes to be along the eigenvectors, rather than age, hours on internet and hours on mobile. However we know that the ev3, the third eigenvector, is pretty useless. Therefore instead of representing the data in 3 dimensions, we can get rid of the useless direction and only represent it in 2 dimensions, like before:



This is dimension reduction. We have reduced the problem from a 3D to a 2D problem, getting rid of a dimension. Reducing dimensions helps to simplify the data and makes it easier to visualize.

Note that we can reduce dimensions even if there isn't a zero eigenvalue. Imagine we did the example again, except instead of the oval being on a 2D plane, it had a tiny amount of height to it. There would still be 3 eigenvectors, however this time all the eigenvalues would not be zero. The values would be something like 10, 8 and 0.1. The eigenvectors corresponding to 10 and 8 are the dimensions where there is a lot of information, the eigenvector corresponding to 0.1 will not have much information at all, so we can therefore discard the third eigenvector again in order to make the data set more simple.

5.1 One final word of caution

When performing PCA, it is typically a good idea to normalize the data first. Because PCA seeks to identify the principal components with the highest variance, if the data are not properly normalized, attributes with large values and large variances (in absolute terms) will end up dominating the first principal component when they should not. Normalizing the data gets each attribute onto more or less the same scale, so that each attribute has an opportunity to contribute to the principal component analysis.

6 Implementation of PCA using scikit-learn

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.

6.1 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. The code below will load the iris dataset.

```
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"

# load dataset into Pandas DataFrame
df = pd.read_csv(url, names=['sepal length','sepal width','petal
length','petal width','target'])
```

	sepal length	sepal width	petal length	petal width	target
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

6.2 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 datasets features onto unit scale (mean = 0 and variance = 1) which is a requirement for the optimal performance of many machine learning algorithms. If you want to see the negative effect not scaling your data can have, scikit-learn has a section on the effects of not standardizing your data.

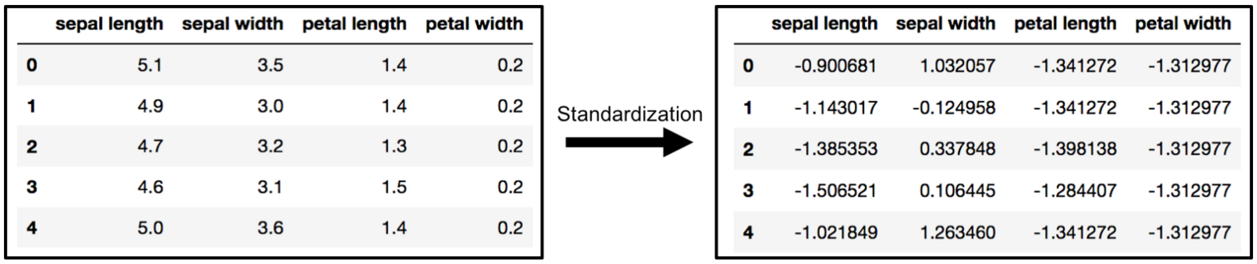
```
from sklearn.preprocessing import StandardScaler

features = ['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)
```


Standardization

	sepal length	sepal width	petal length	petal width
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

	sepal length	sepal width	petal length	petal width
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
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

6.3 PCA Projection to 2D

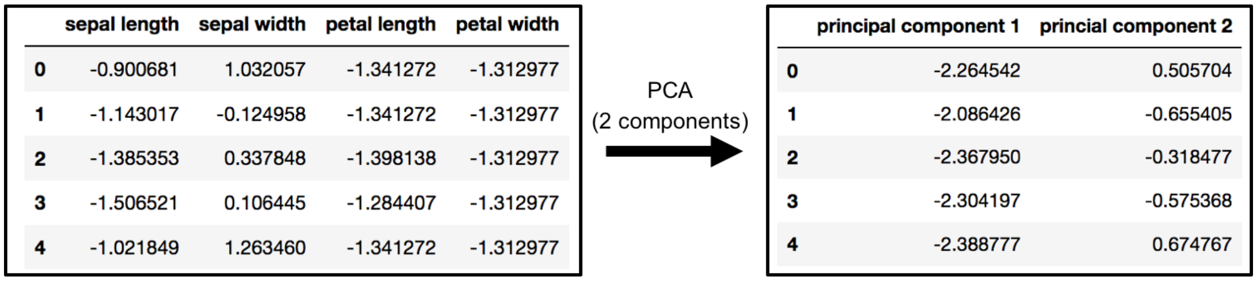
The original data has 4 columns (sepal length, sepal width, petal length, and petal width). In this section, the code projects the original data which is 4 dimensional into 2 dimensions. I should note that after dimensionality reduction, there usually isn't a particular meaning assigned to each principal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

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

principalComponents = pca.fit_transform(x)

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

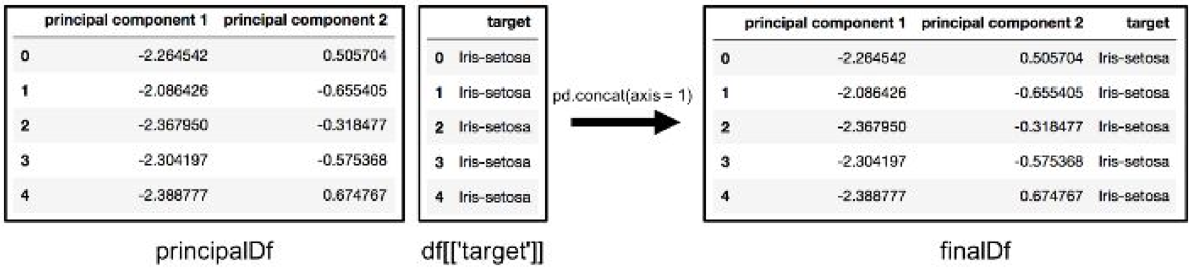

PCA
(2 components)

	sepal length	sepal width	petal length	petal width
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
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

	principal component 1	princial 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

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

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


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

	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

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

	principal component 1	principal component 2	target
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

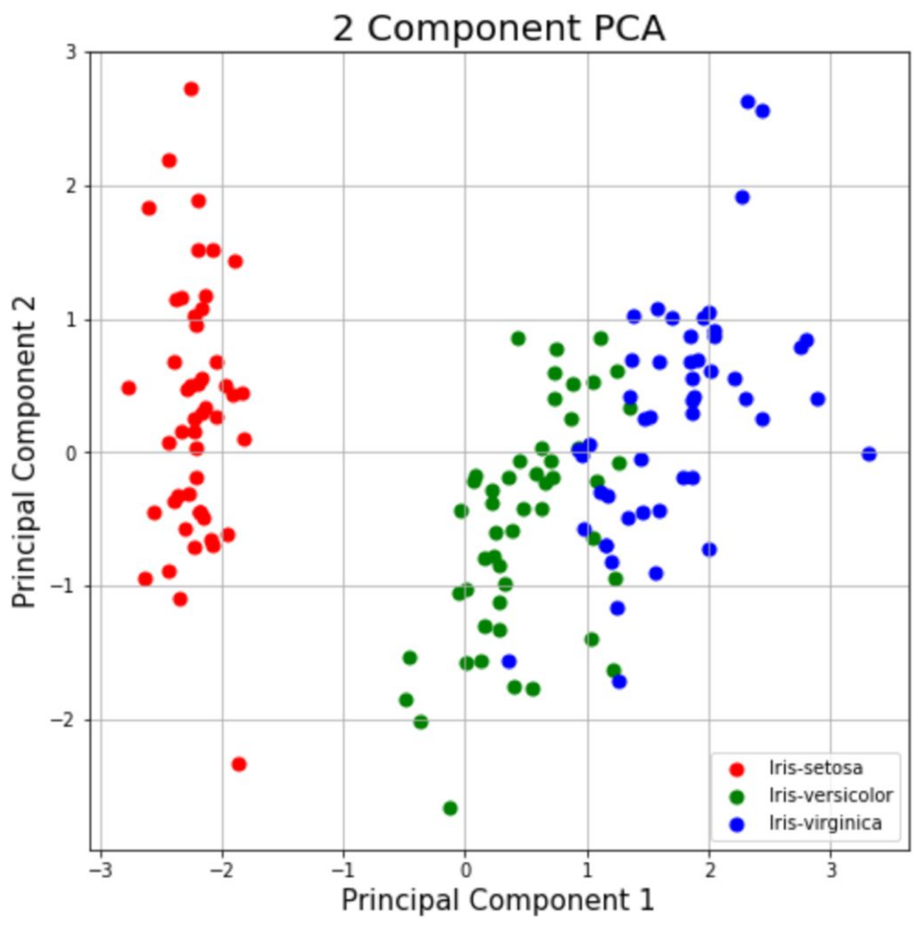
Concatenating dataframes along columns to make finalDf before graphing

6.4 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
Principal Component 1
Principal Component 2
Iris-setosa
Iris-versicolor
Iris-virginica