Principal Component Analysis - PCA

Principal Component Analysis - PCA

- Noise filtering
- Visualization
- Feature Extraction
- Stock market predictions
- Gene data analysis

Principal Component Analysis - PCA

- Identify patterns in data
- Detect the correlation between variables

Principal Component Analysis - PCA

Reduce the dimensions of a d-dimensional dataset by projecting it onto a (k)-dimensional subspace (where k<d)

297. Principal Component Analysis (PCA) Intuition
Playback Rate
1.5x
1:26 / 3:49

Principal Component Analysis - PCA

- Standardize the data.
- Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
- Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace ($k \leq d$)/.
- Construct the projection matrix $\mathbf{W}$ from the selected k eigenvectors.
- Transform the original dataset $\mathbf{X}$ via $\mathbf{W}$ to obtain a k-dimensional feature subspace $\mathbf{Y}$

https://plot.ly/ipython-notebooks/principal-component-analysis/

setosa.io/ev/principal-component-analysis/

Explained Visually

By Victor Powell

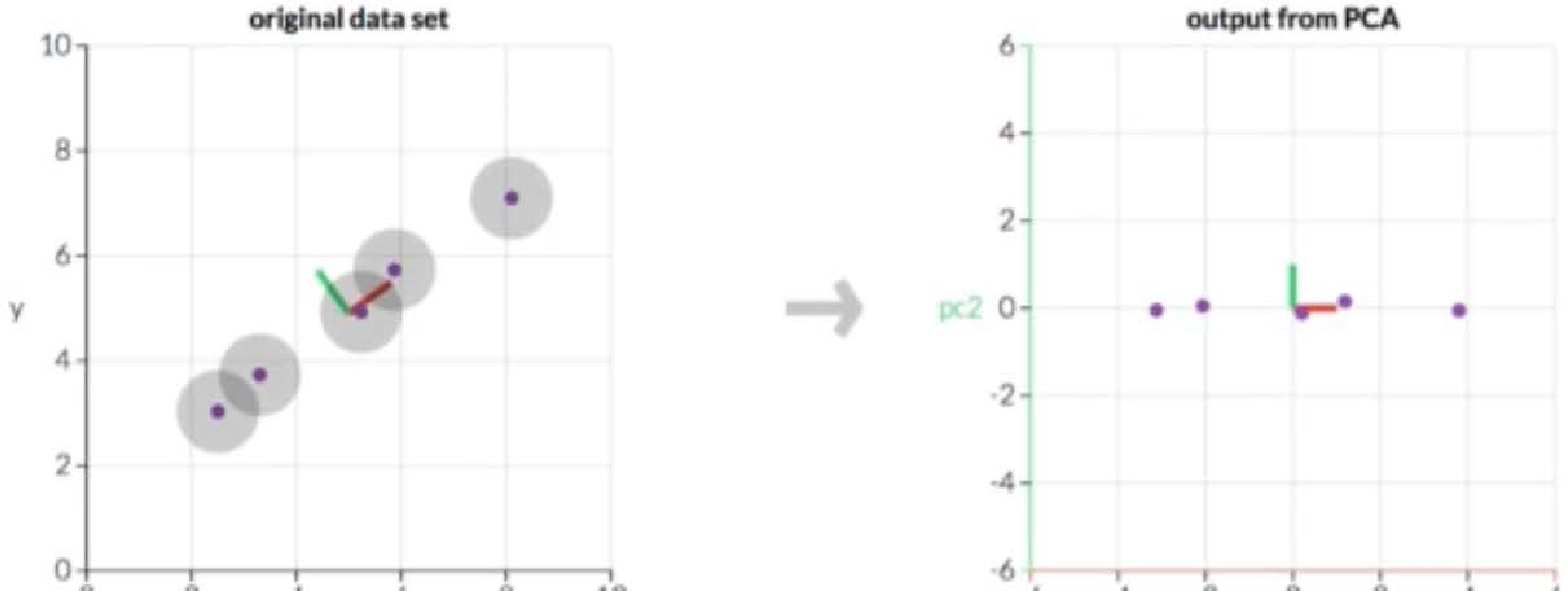
with text by Lewis Lehe

Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

2D example

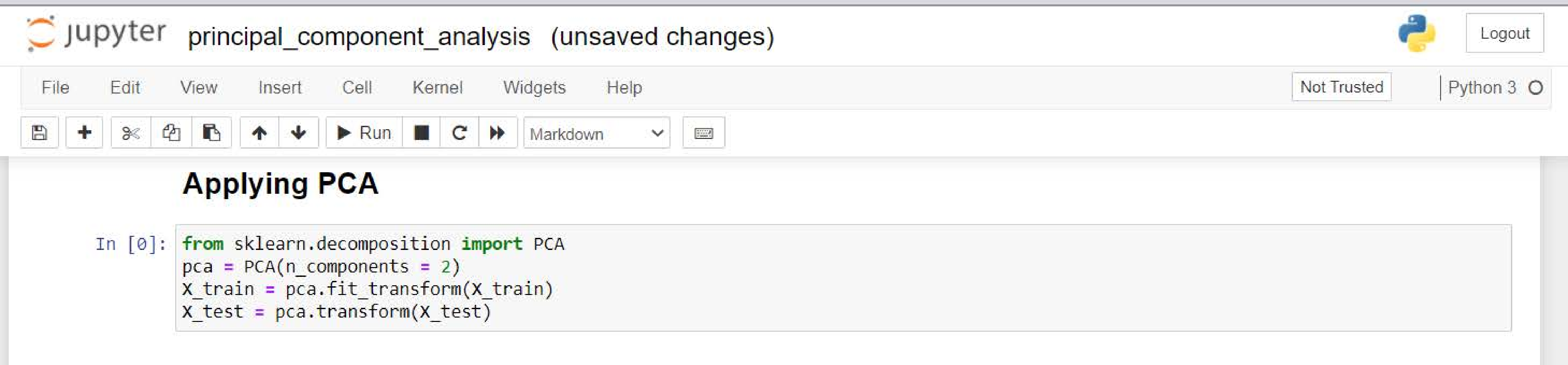
First, consider a dataset in only two dimensions, like (height, weight). This dataset can be plotted as points in a plane. But if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes don't actually mean anything physical; they're combinations of height and weight called "principal components" that are chosen to give one axes lots of variation.

Drag the points around in the following visualization to see PC coordinate system adjusts.



Principal Component Analysis - PCA

- Learn about the relationship between X and Y values
- Find list of principal axes


jupyter principal_component_analysis (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Applying PCA

```
In [0]: from sklearn.decomposition import PCA
        pca = PCA(n_components = 2)
        X_train = pca.fit_transform(X_train)
        X_test = pca.transform(X_test)
```

Linear Discriminant Analysis - LDA

Linear Discriminant Analysis - LDA

- Used as a dimensionality reduction technique
- Used in the pre-processing step for pattern classification
- Has the goal to project a dataset onto a lower-dimensional space

Linear Discriminant Analysis - LDA

- Sounds similar to PCA right?

LDA differs because in addition to finding the component axises with LDA we Are interested in the axes that maximize the separation between multiple classes.

Linear Discriminant Analysis- LDA

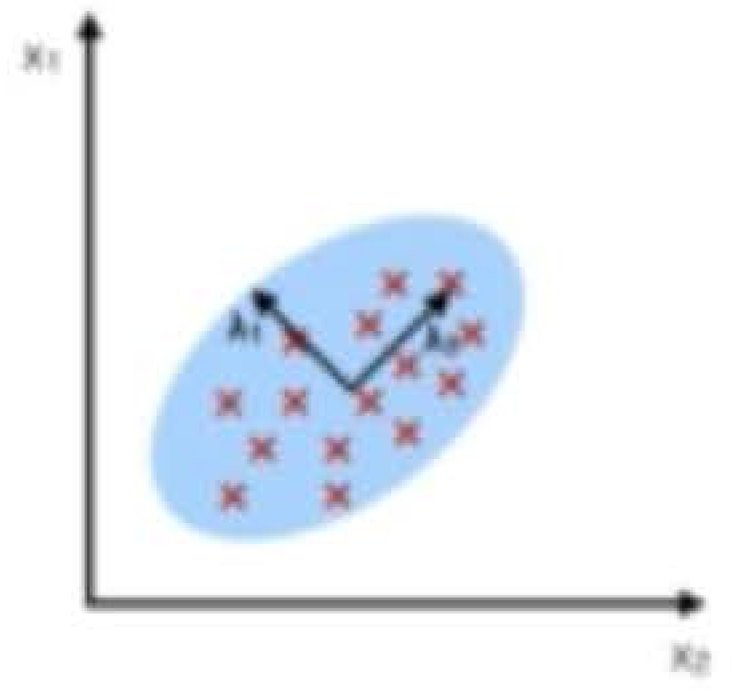
Breaking it down further:

The goal of LDA is to project a feature space (a dataset n-dimensional samples) onto a small subspace subspace k(where $k \leq n-1$) while maintaining the class-discriminatory information.

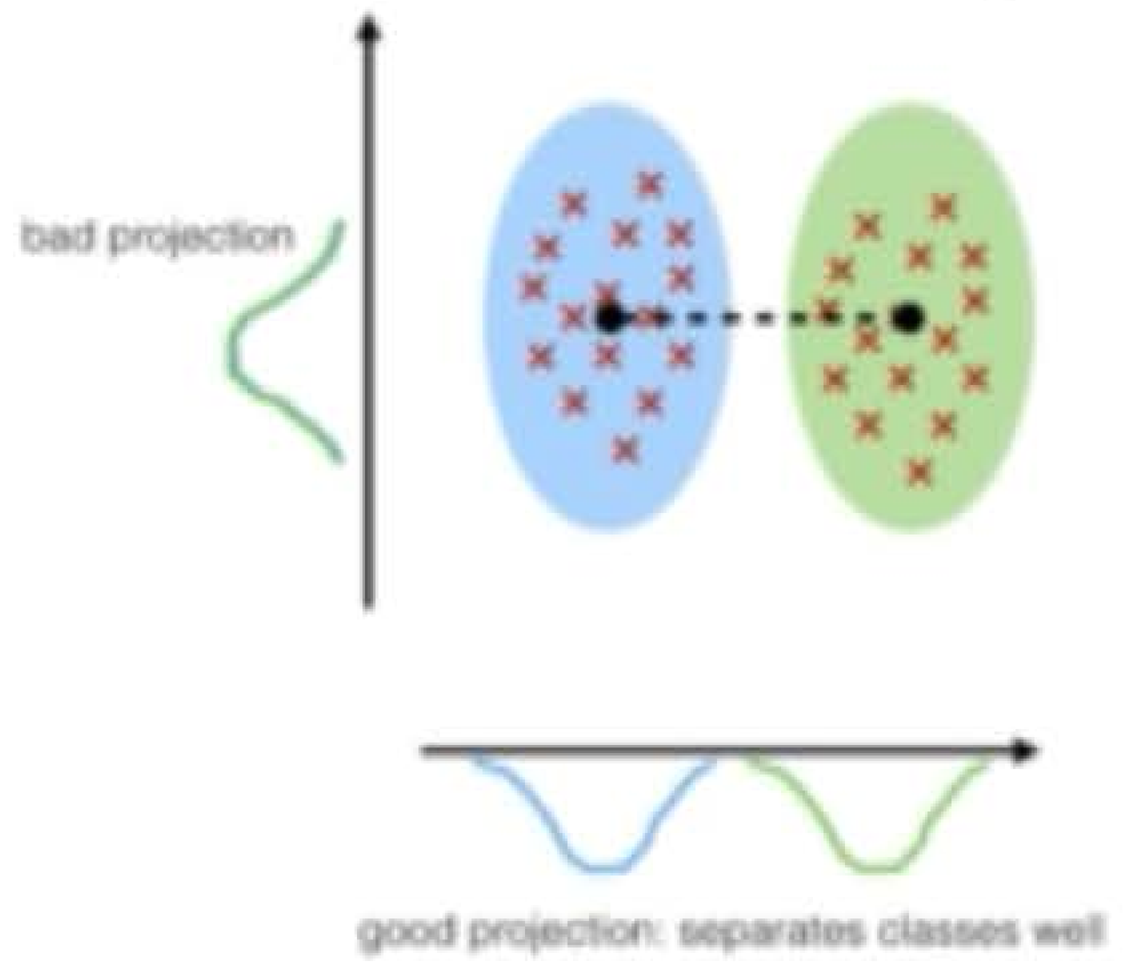
Both PCA and LDA are linear transformation techniques used for dimensional reduction. PCA is described as unsupervised but LDA is supervised because of the relation to the dependent variable.

Linear Discriminant Analysis - LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation

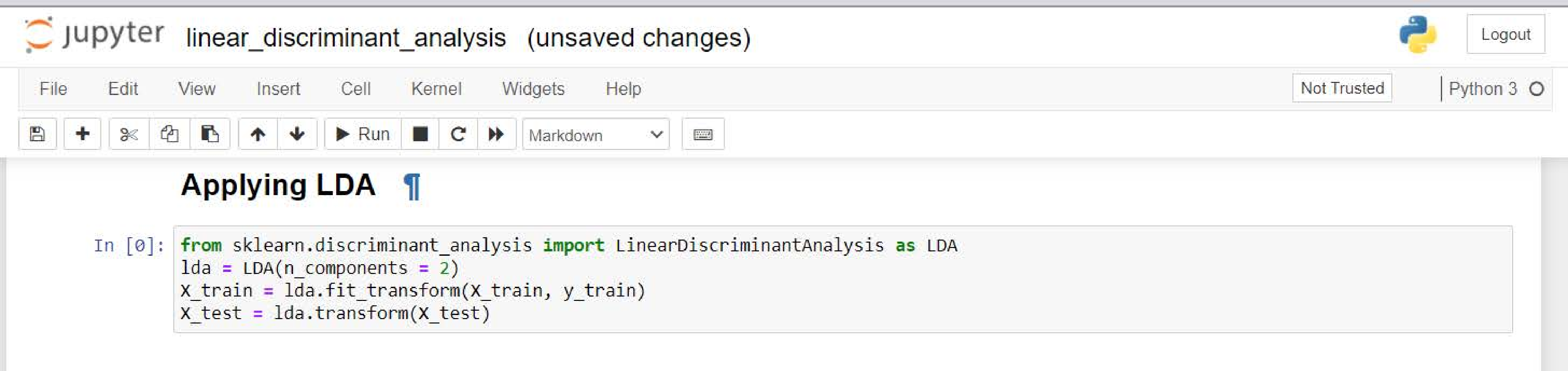


LDA: https://sebastianraschka.com/Articles/2014_python_lda.html

Linear Discriminant Analysis - LDA

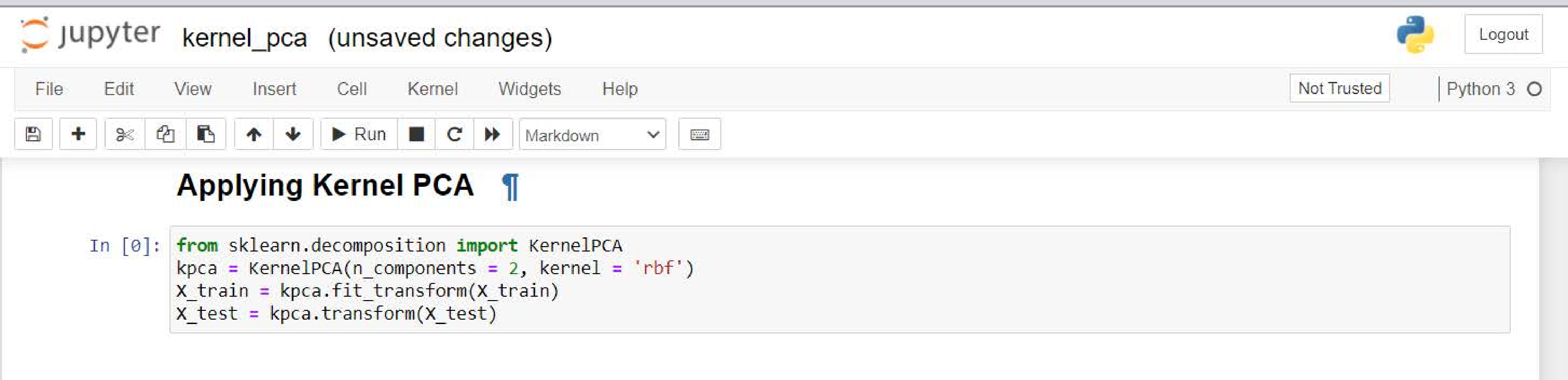
1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

https://sebastianraschka.com/Articles/2014_python_lda.html



Applying LDA ¶

In [0]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 2)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```


jupyter kernel_pca (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Applying Kernel PCA ¶

In [0]:
```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components = 2, kernel = 'rbf')
X_train = kpca.fit_transform(X_train)
X_test = kpca.transform(X_test)
```