Introduction to Data Science

PCA and K-Means

Assignment 3

Submitted by

Andras Csepreghy

xgj708

19th of February, 2019

UNIVERSITY OF
COPENHAGEN

Introduction

This assignment includes basic classification with K-Nearest Neighbors classifier classifier, a non-linear, non-parametric method for classification. Then applying cross-validation for model selection and standard data normalization for preprocessing.

Exercise 1

a) PCA

In the pca.py file I created the function called pca that takes in 3 arguments and returns 3 values. Slightly different from what was expected of us, but that is on purpose. I used numpy to create the necessary steps and sci-kit's PCA function was used sometimes to compare my results (it is no longer used in the code now)

The 3 arguments:

X:
Just the input data on which principal component analysis (PCA) will be performed
show_pc_plots:
A boolean that is a condition whether the function should create and show a plot about the explained variance and the cumulative variance of principal components. This is needed, because there is no reason to show ratios between principal components (PCs) when there is only two for the murder dataset, but I didn't want to clutter the __init__.py in which I mostly read data and call functions from different files.

with_std:
Centering the data made sense in all cases, however it wasn't very obvious when to standardize it. For instance we don't have too much information about the pesticide data outside the short description we got in the assignment. We can't be 100% certain that the data requires standardization or not, so I left the opportunity open to work with both standardized and non scaled data. The PCA gives completely different results

The 3 returned values:

unit_egenvectors:
We return the eigenvectors of the covariance matrices that are created from the input data matrix. Each eigenvector was converted to a unit vector by dividing it by its norm.

eigenvalues:
We return all the eigenvalues in descending order. The eigenvalues have the same order as their corresponding eigenvectors meaning each at every index you would find a correct eigenvector/eigenvalue pair.

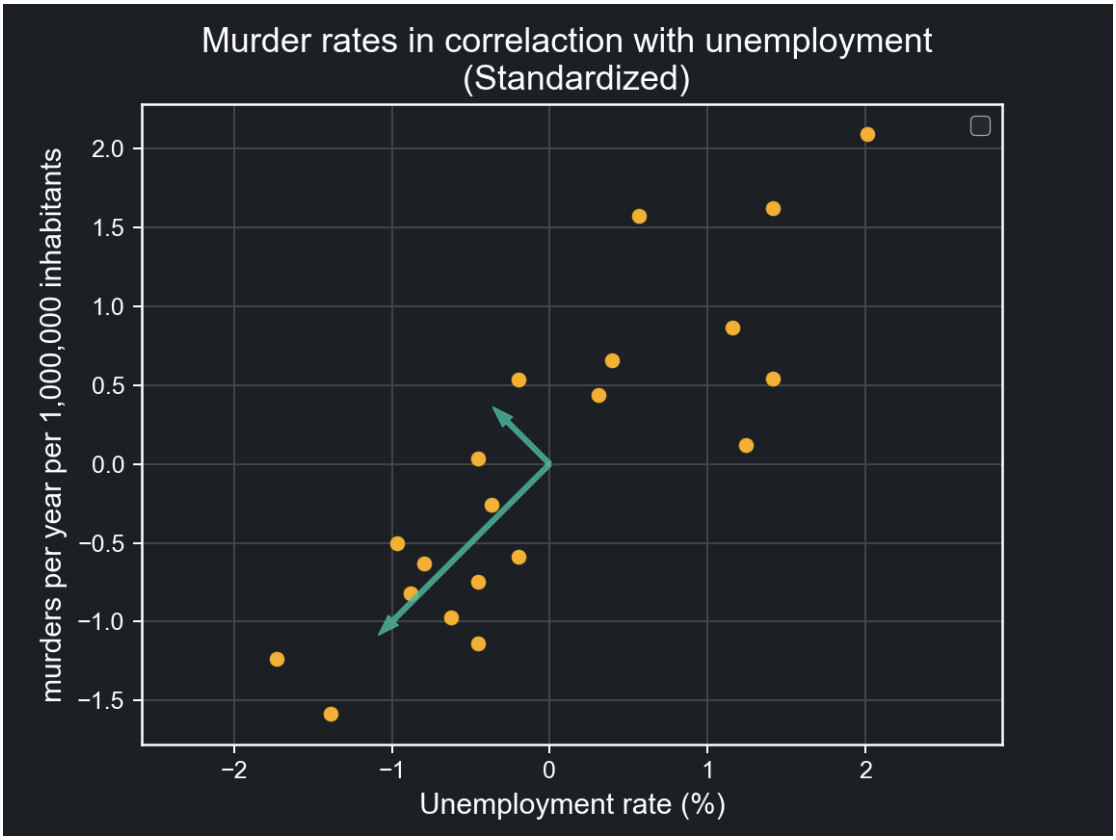
mean:
The mean of the input data matrix. For some plots, we need the mean.

Steps for PCA:

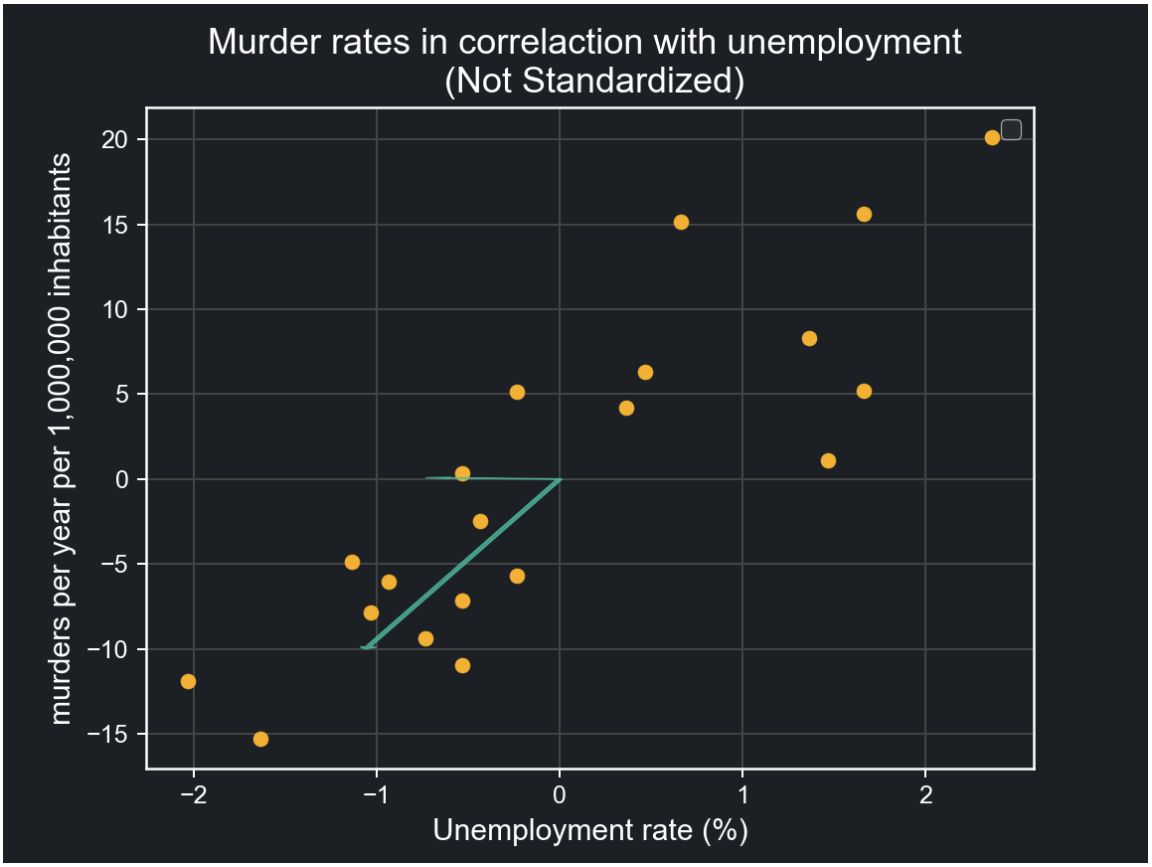
- Standardize and/or center the data (this step is not always applied)
- Create covariance matrix from the input data matrix
- Get eigenvector/eigenvalue pairs
- Sort them in descending order based on the eigenvalue (the larger the eigenvalue the more variance it represents)
- Create unit vectors from eigenvectors by dividing them by their norm
- Calculate and plot the cumulative variance versus the number of used PCs

b) Murder dataset

Here are the plots for both the standardized and the non-standardized murder dataset. The blue arrows are the eigenvectors with a length scaled by the standard deviation of the data projected onto that eigenvector.

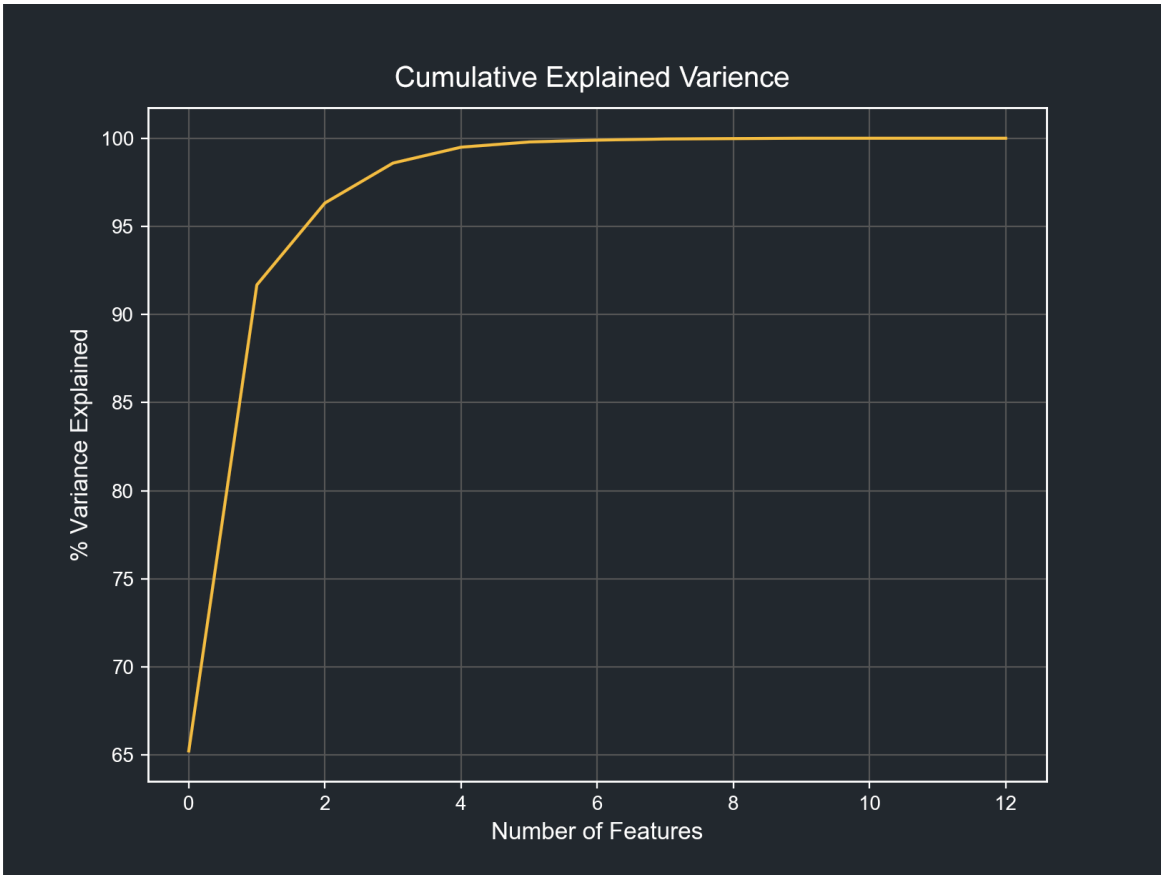


On plot below the blue arrows don't appear to be perpendicular however the vectors are indeed orthogonal. The distortion is caused by the different scale on different axes. plt.axis('equal') would correct it, but distorts the datapoints so much, that I decided to use this one. They show the same plot just in different scales.

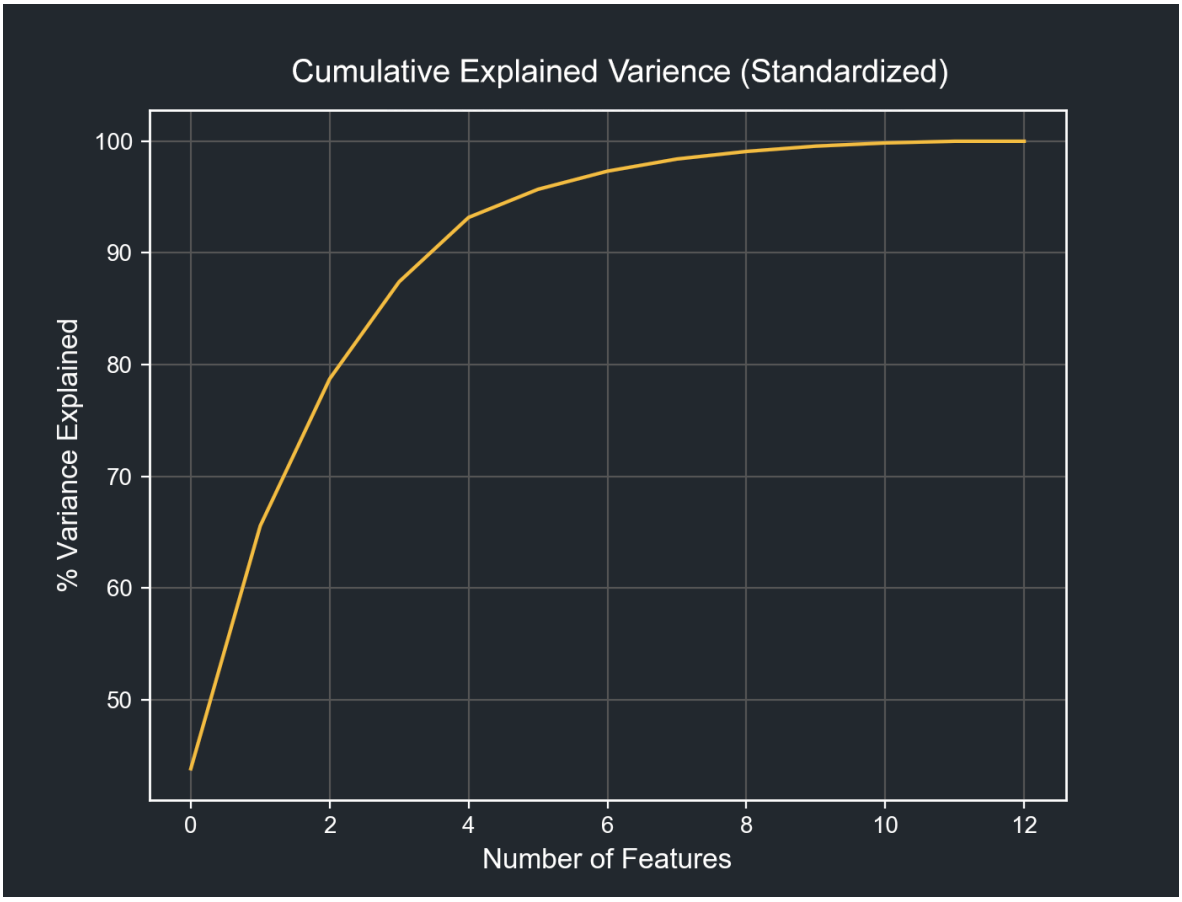


c) Pesticide dataset

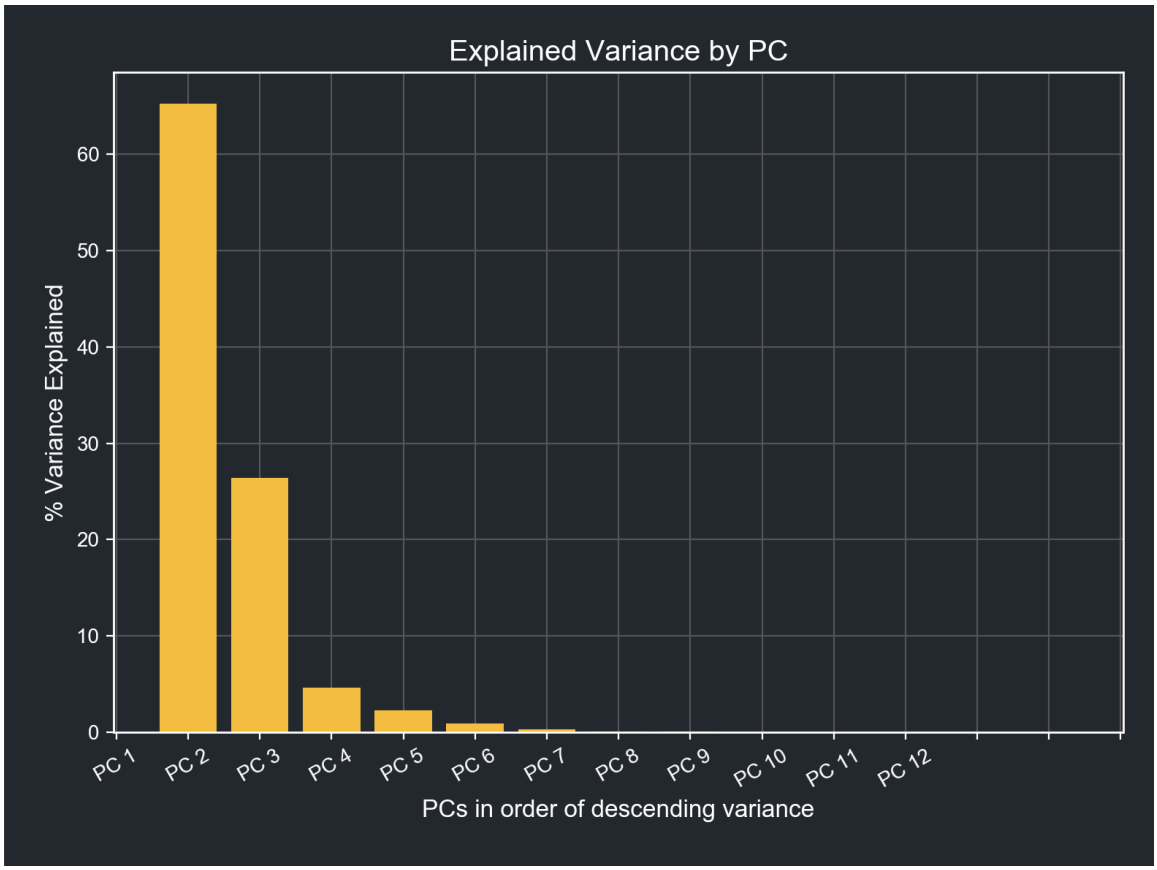
Having done PCA on this dataset we most of the variance captured by the first few dimensions. That is different (as explained later) when standardizing the data. I did standardization out of experimenting. On the first 2 plots we see the cumulative variance explained versus the number of dimensions.



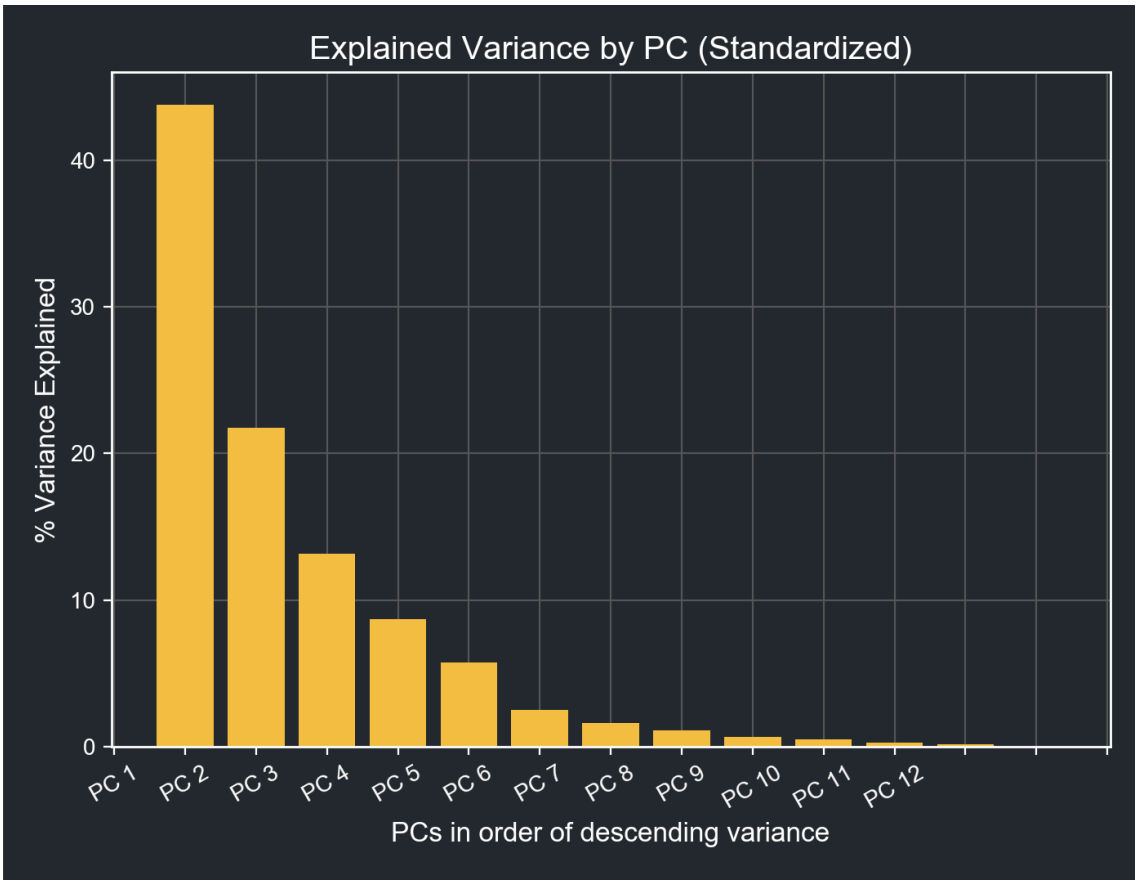
Takes significantly more dimensions to reach 90% of the variance captured when standardized.



These two plots show the variance captured by each principal component in descending order (multiplied to capture %). These two bar plots can be seen as the same concept from a different view point to the previous two plots.



Again, same difference between standardized and non-standardized data.

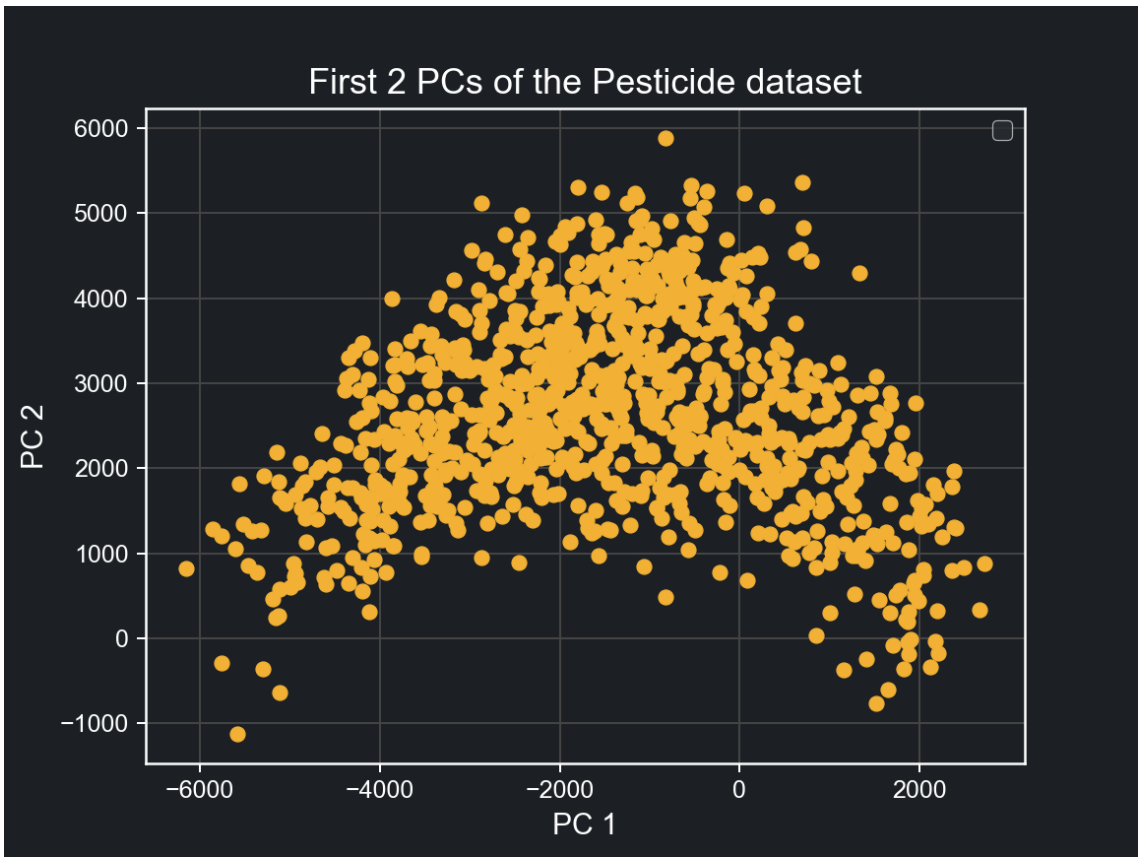


How many PCs (dimensions) do you need to capture 90% of the variance in your dataset? 95%?
This is irrelevant for the murder dataset, because it's only 2 dimensions.
For the pesticide dataset, based on printing the cumulative_explain_variences, the first two PCs give 91.67693% of the variance and the first 3 give 96.32517% of it without scaling the data. However if the data is standardized before doing the PCA the results are very different. As expected the amount of variance differences between dimensions decreases. In this case the first 5 dimensions will give 93.17742% of the variance and the first 6 give 95.69540% of it. This is not surprising as standardizing the data removes variance between features.

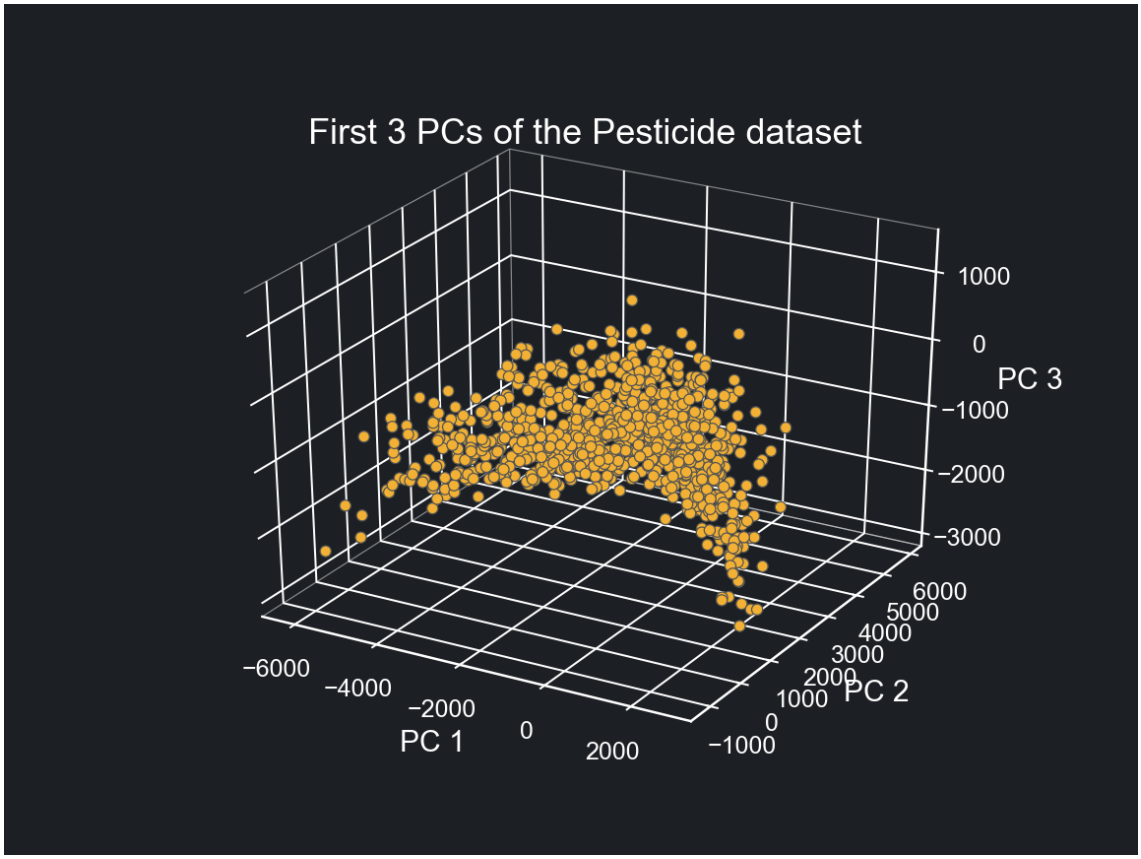
For PCA in the pesticide dataset I only considered the first 13 numbers in each vector because they correspond to features of the dataset while the last number is a binary categorical value that doesn't have continuous nature and is the dependent variable.

Exercise 2

In exercise 2 I did a multidimensional scaling on the pesticide dataset using the PCA functions created in exercise 1. For the fun of it I did It both for 2D and 3D to have some nice-looking plots. First plot shows the dataset projected onto the first 2 principal components.



And the second one is with using the same function but projecting onto the first 3 PCs and creating a 3D plot out of it.



Exercise 3

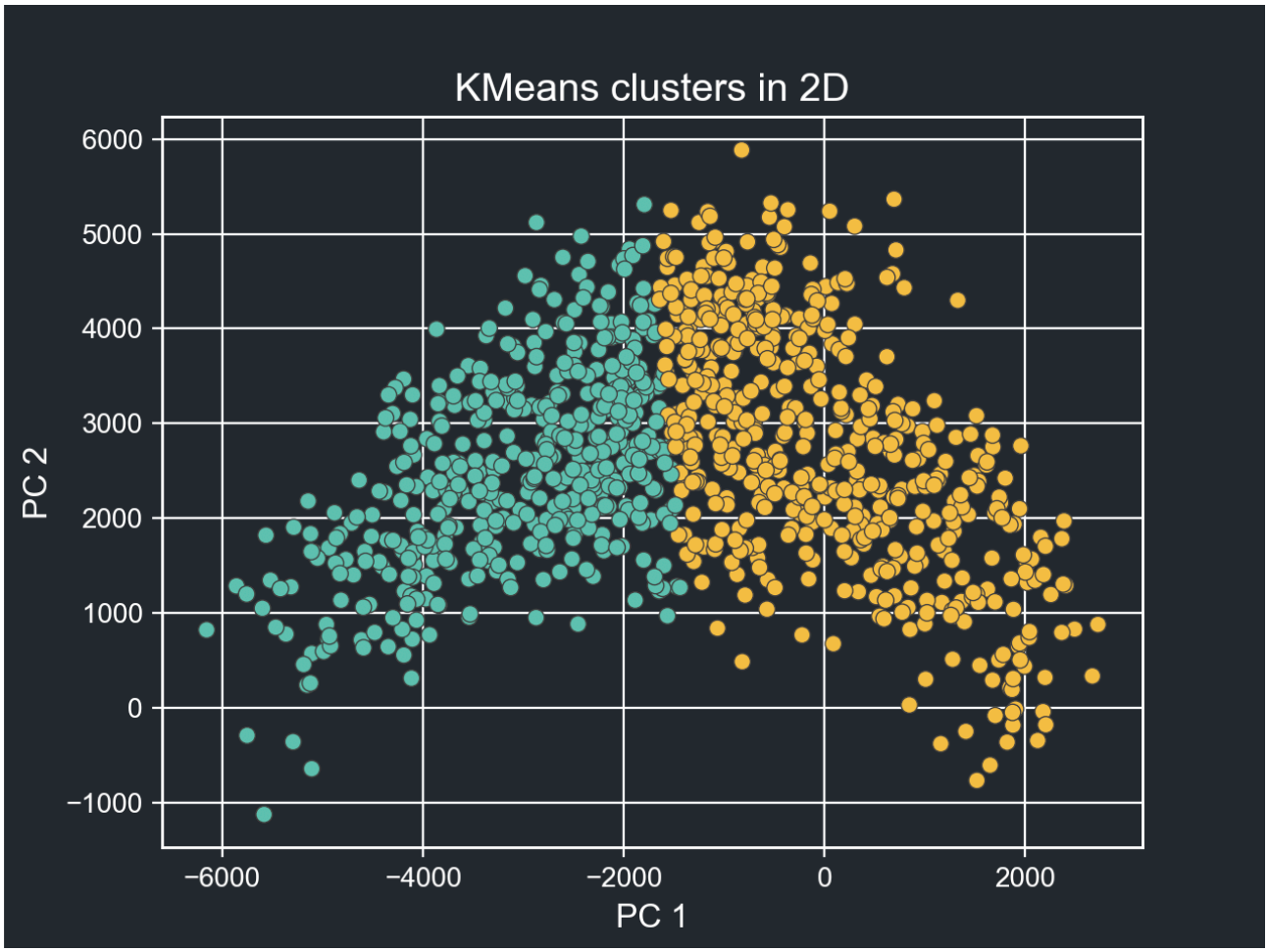
There is not much theory to discuss here. The assignment basically describes exactly what I did, and what steps I followed. I used sci-kit learn's Kmeans class and I initialized it as told, with the first datapoint in the dataset. Which is generally a bad idea. Made k = 2, which is the number of clusters the algorithm is going to produce by minimizing the mean distance to the cluster centroids to all the points that belong to that cluster.

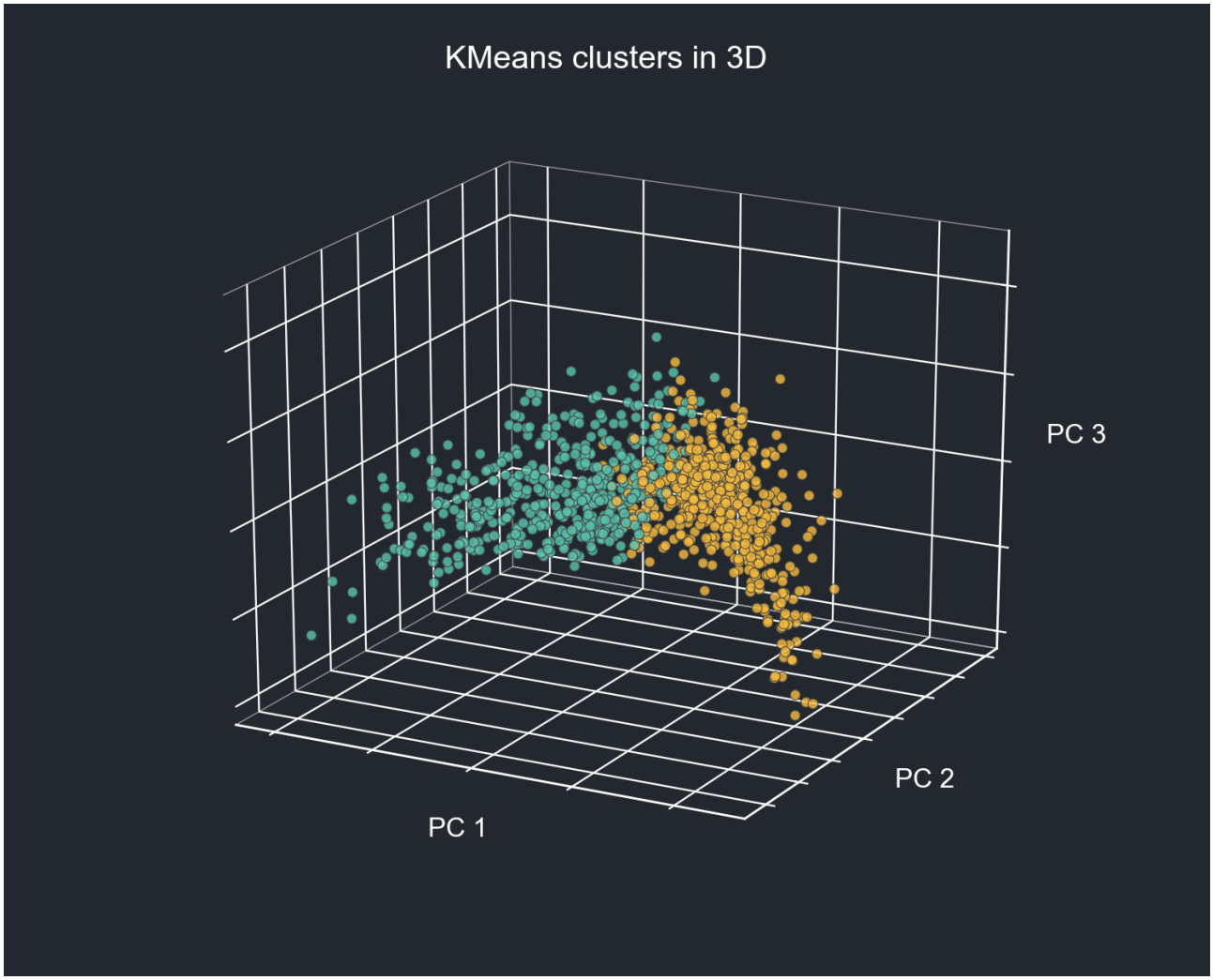
The cluster centers should be the last print my code makes, however here are the two centers as numpy arrays:

[5.68220339e+00 4.93389831e+01 7.90923729e+02 3.84509110e+03
 3.38647246e+03 1.36161229e+03 2.94523305e+02 1.31936441e+02
 7.07457627e+01 3.95805085e+01 1.94152542e+01 4.22669492e+00]

[2.19507576e+00 1.37007576e+01 1.70367424e+02 1.39206250e+03
 3.18763636e+03 2.62546970e+03 1.00435985e+03 6.33471591e+02
 4.96619318e+02 2.95941288e+02 1.46073864e+02 2.92537879e+01]

And as some bonus exercise here is the pesticide dataset projected onto 2 PCs and 3PCs with the clusters colored differently. This took me hours...




KMeans clusters in 3D
PC 3
PC 2
PC 1