

# Universitat de Barcelona

## Introduction to Machine Learning

---

### Work 2: Principal Component Analysis Exercise

---

Authors: Emer Rodriguez Formisano, Georgina Ballbè i Serra, Jorge Alexander

Supervisor: Maria Salamó

Date: 20th November 2017

<b>Introduction</b>	<b>1</b>
<b>Algorithm</b>	<b>1</b>
<b>Dataset Analysis</b>	<b>4</b>
Iris	4
Descriptive Statistics	4
Eigenvalues	5
Principle components	5
Wine	7
Descriptive statistics	7
Eigenvalues	9
Principal Components	10
Balance	12
Bupa	12
Dataset properties	12
Calculating PCA	13
<b>Conclusions</b>	<b>17</b>
<b>Code execution</b>	<b>18</b>

# Introduction

In this work implement the Principal Component Analysis (PCA) algorithm and we use it to analyze several data sets from the UCI repository. After this analysis, we compare the results with those obtained with those obtained using the function available in scikit learn: `sklearn.decomposition.PCA`.

## Algorithm

This section describes the PCA algorithm implemented for this study.

The main function which contains the algorithm for PCA in the code is:

```
def iml_pca(x_data,x_labels,k=None,standardized=False):  
    #this function implements the algorithm to do principal component analysis.
```

*x\_data* is the data to be analyzed, represented in a  $N \times d$  matrix ( $N$  input variables of dimension  $d$ )

*x\_labels* is a vector of dimension  $1 \times d$  which contains the names of each of the variable features.

*k* is the value of the dimension of the transformed data. By default is *None*, and it is calculated so that the total variance of the data is over 90%.

*standardized* if set to *True* *x\_data* will be divided by the standard deviation after subtracting the mean.

First we calculate the mean of each feature (dimension) and subtract it from the data:

```
#calculate the mean of each feature  
means_array = np.mean(x_data, axis=0)  
  
if standardized:  
    std_array = np.std(x_data, axis=0)  
    pca_data = (x_data - means_array) / std_array  
else:  
    #Subtract the mean of the feature to each feature value  
    pca_data=x_data-means_array
```

We plot the correlation matrix to support understanding of the dataset:

```
plot_correlation(pca_data)
```

We calculate the covariance matrix and print the values:

```
#Calculate covariance matrix
covars_matrix = np.cov(pca_data, rowvar=False)
print ("Covar matrix")
print covars_matrix
```

We calculate the eigenvalues and eigenvectors using the function available in numpy linear algebra library::

```
#Calculate the eigenvectors and eigenvalues
w, v = LA.eig(covars_matrix)
```

$w$  is a vector of  $d$  eigenvalues, and  $v$  is a matrix of size  $d \times d$ ,  $d$  eigenvectors of size  $d$ .  
We sort the eigenvalues in descending order, and the eigenvector accordingly:

```
#Sort the eigenvectors and eigenvalues
idx = w.argsort()[::-1]
eigenValues = w[idx]
eigenVectors = v[:,idx]
```

Plotting the eigenvalues we can see what components have a higher weight, more information and how the variance of the data is distributed among the components

```
plot_eigen_values(eigenValues)
```

Once we have this values we need to set how many values are relevant for our application.  $k$  can either be set as a hyperparameter or we can find it using a rule, in this case, keep the 90% of the total variance of the data. Although we implemented 90% as an automatic way of selecting  $k$  components, the correct value of  $k$  is that it will depend on the scope:

```
if not k:
    #How to chose k?
    for k in range(2,len(eigenVectors)):
        if (np.sum(eigenValues[0:k])/np.sum(eigenValues)) > 0.9:
            break

print "K value is: {}".format(k)
```

The eigenvalues and eigenvectors matrix are reduced to the new dimension, we keep the first  $k$  elements.

```
kEigenVectors=eigenVectors[:,0:k]
kEigenValues=eigenValues[0:k]
```

$kEigenVectors$  is a matrix of size  $d \times k$ , each column is an eigenvector.  
 $kEigenValues$  is a vector of size  $1 \times k$ .

With this new matrix we calculate the transformed dataset, by multiplying the first  $k$  eigenvectors and the variables of data matrix:

```
pca_data(dim= $n \times d$ ) .* kEigenvectors(dim= $d \times k$ )  $\rightarrow$  newDataSet(dim= $n \times k$ )
```

```
#derive new data set  
newDataSet=np.dot(pca_data, kEigenVectors)
```

Finally, we want to reconstruct the data. To do so, we need to multiply the  $k$ Eigenvectors with the transformed data (newDataSet):

```
newDataSet(dim= $n \times k$ ).*kEigenvectors' (dim= $k \times d$ )  $\rightarrow$  dataRecover(dim= $n \times d$ )
```

add the mean, and if standardized, multiply by the standard deviation.

```
#reconstruction  
dataRecover = np.dot(newDataSet, kEigenVectors.transpose()) + means_array  
if standardized:  
    dataRecover = dataRecover * std_array  
  
print ("Data reconstructed")  
plot_vars(x_labels, dataRecover, 0, 1)
```

In addition to this main function, we created auxiliary functions to plot the data:

```
plot_eigen_values(eig_values)  
#plots the eigenvalues and the % of the data variance accumulated in each component.
```

```
plot_components(proj, evec, idx=[0, 1], var_labels=None, scale=True)  
#PCA biplot which shows the individuals and the variables in two subplots for the  
selected components (idx). Similar to plot_vars but customised for PCA components.
```

```
descriptive_stats(X,var_labels)  
#plots descriptive statistics for the data: min, max, 1st 2nd and 3rd Qu, mean and  
standard deviation.
```

```
plot_vars(var_labels, var_data, var_1, var_2)  
#plots pairs of var_data features/dimensions (var_1, var_2)
```

```
plot_correlation(X)  
#plots the correlation matrix for the correlation values between the columns of matrix X.
```

The complete code is in the jupyter notebook file: Work2.ipynb. To obtain results run all the cells from top to bottom.

# Dataset Analysis

## Iris

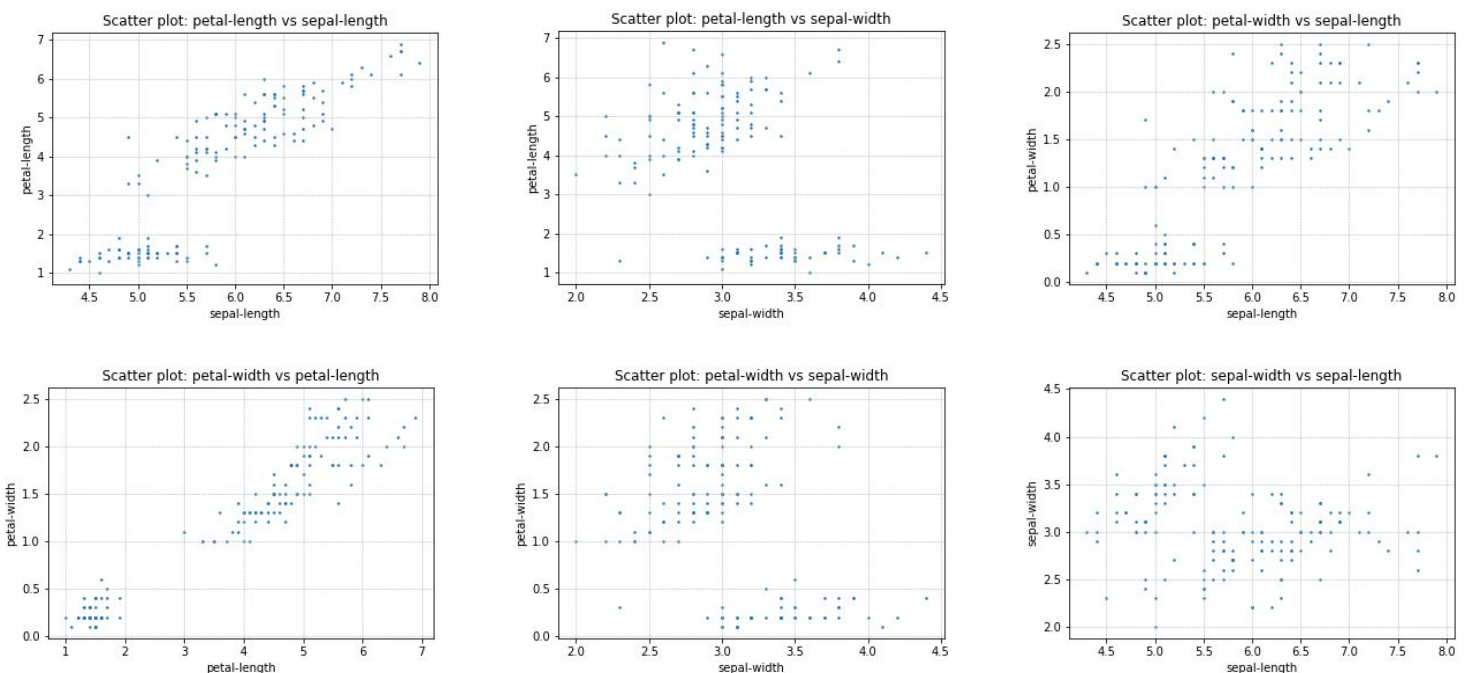
The initial dataset that was analysed was the Iris dataset. The iris dataset contains 150 measurements of flowers' *sepal-length*, *sepal-width*, *petal-length* and *petal-width*. Each flower is of one of 3 classes: *Iris-setosa*, *Iris-versicolor* or *Iris-virginica*.

## Descriptive Statistics

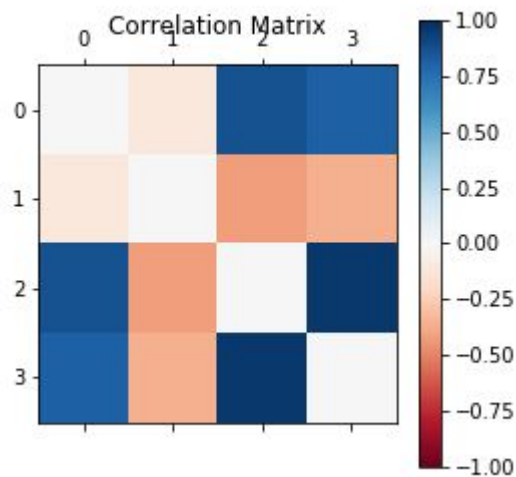
Statistical summary of the iris dataset:

var	Min.	1st Qu.	Median	Mean	Std.Dev	3rd Qu.	Max.
sepal-length	4.30	4.34	4.37	5.84	0.83	4.40	7.90
sepal-width	2.00	2.07	2.15	3.05	0.43	2.20	4.40
petal-length	1.00	1.04	1.07	3.76	1.76	1.11	6.90
petal-width	0.10	0.10	0.10	1.20	0.76	0.10	2.50

Scatter plots of each dimension against the others:



Correlation matrix:



Strong correlations are already observed, particularly between the petal-width and petal-length.

## Eigenvalues

Component	1	2	3	4
Eigen Vector	[ 0.36158968 -0.65653988 -0.58099728 0.31725455]	[-0.08226889 -0.72971237 0.59641809 -0.32409435]	[ 0.85657211 0.1757674 0.07252408 -0.47971899]	[ 0.35884393 0.07470647 0.54906091 0.75112056]
Absolute Variance Contribution	4.22484077	0.24224357	0.07852391	0.02368303
Percentage Variance Contribution	0.92461621	0.05301557	0.01718514	0.00518309

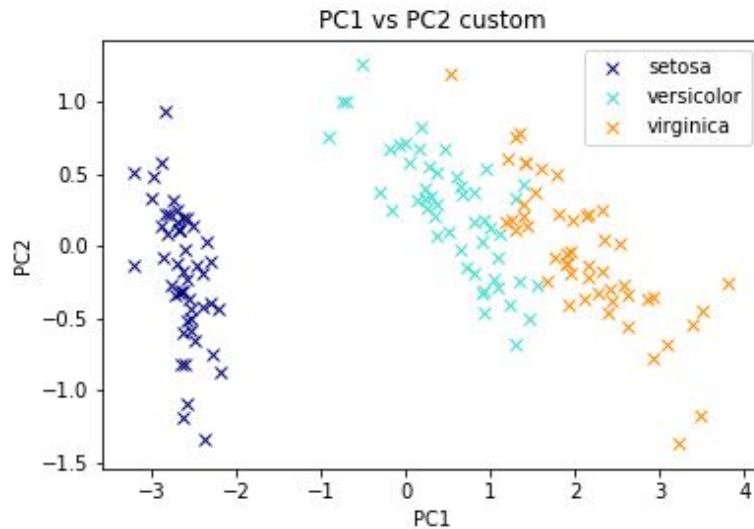
It can be seen that a single component of the Iris dataset, contributes 92% of the variance, and therefore the effect of adding more than one component for PCA is very small. Two will be used for the purpose of analysis however, for testing the custom implementation.

## Principle components

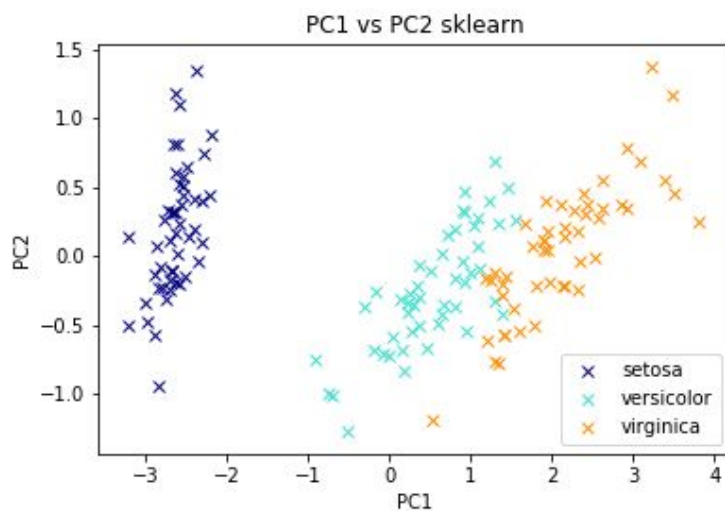
### Custom implementation

Below the two components from our PCA implementation are shown. It is now observed that

the *setosa* class is now very well separated from the rest and easy to classify. The *versicolor* are closer together, but there is a clear linear spread which also aids classification.



Sklearn implementation



The graph now shows the data points color-coded by their class label with each principle component each on an axis. It is observed how the sklearn implementation reverses the direction of the principle components, however this would not affect a classification process. This happens because scikit learn uses the SVD to get the eigenvectors and eigenvalues, instead of the numpy "linalg" library, which was used in the custom implementation.

# Wine

The third dataset analysed in the previous work was Wine dataset.

## Descriptive statistics

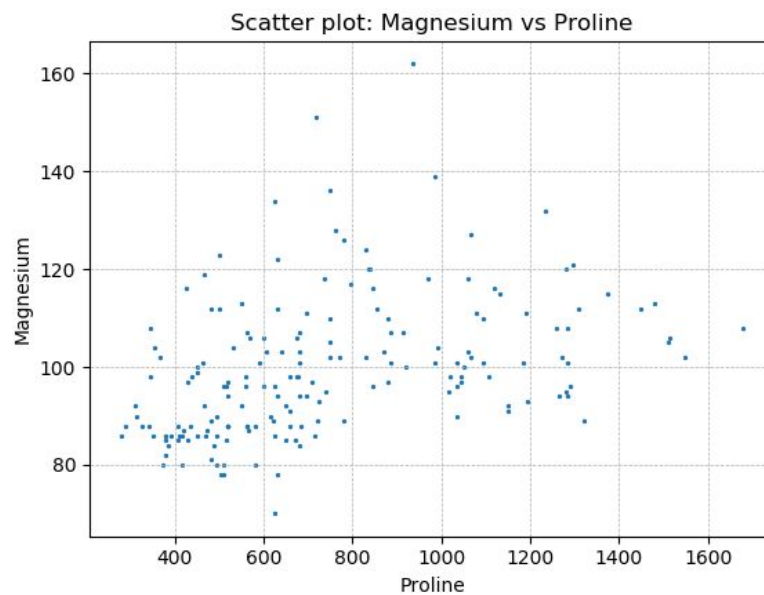
Before applying the PCA technique, a quick review of the variables is done by using descriptive statistics. The follow summarises them.

Idx.	Variable	Min.	1st Qu.	Median	Mean	Std.Dev	3rd Qu.	Max.
0	Alcohol	11.03	11.20	11.37	13.00	0.81	11.42	14.83
1	Malic acid	0.74	0.81	0.87	2.34	1.11	0.89	5.80
2	Ash	1.36	1.51	1.66	2.37	0.27	1.70	3.23
3	Alcalinity of ash	10.60	10.87	11.13	19.49	3.33	11.27	30.00
4	Magnesium	70.00	73.54	77.08	99.74	14.24	78.00	162.00
5	Total phenols	0.98	1.03	1.09	2.30	0.62	1.12	3.88
6	Flavanoids	0.34	0.40	0.46	2.03	1.00	0.47	5.08
7	Nonflavanoid phenols	0.13	0.13	0.14	0.36	0.12	0.14	0.66
8	Proanthocyanins	0.41	0.41	0.42	1.59	0.57	0.42	3.58
9	Color intensity	1.28	1.48	1.69	5.06	2.31	1.79	13.00
10	Hue	0.48	0.51	0.53	0.96	0.23	0.54	1.71
11	OD280/OD315 of diluted wines	1.27	1.28	1.29	2.61	0.71	1.29	4.00
12	Proline	278.00	283.31	288.62	746.89	314.02	297.20	1680.00

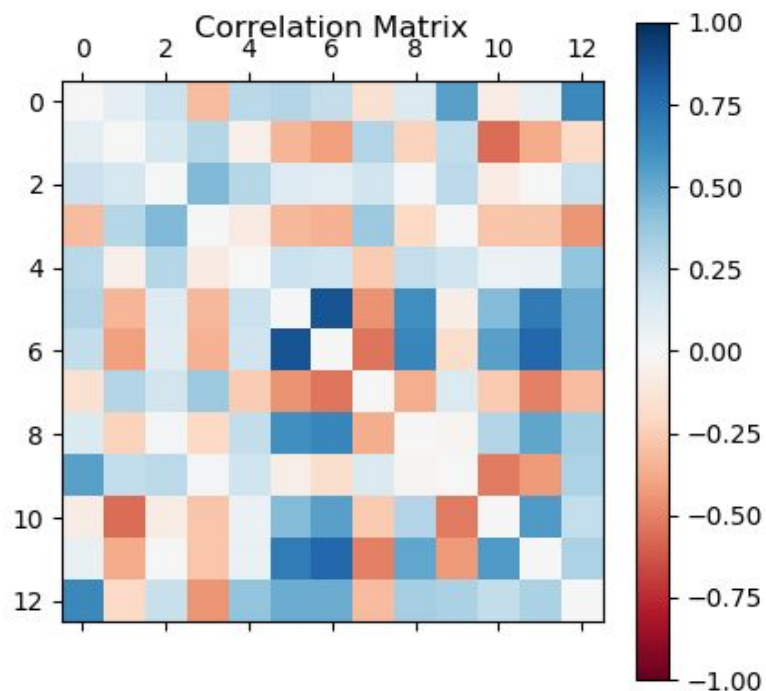
Some interesting ideas can be extracted which may condition the multivariate analysis.

- Having 13 variables supports the idea of using dimensionality reduction techniques such as PCA analysis.
- There is heterogeneous variables with diverse units variability. Thus, a preprocessing step is required in order to centralise and scale the variables. In this case, the standard score is used, that can be defined as  $z = \frac{y - \bar{x}}{\sigma}$ . The standardization has an extra property, useful for the interpretation of the axis, which is making the eigenvectors to have maximum length of 1.
- The table also shows that Magnesium and Proline variables have a high standard deviation. This fact is related to the high unit values. Both variables are dominant in the non scaled space but the standardization will remove the issue. The following scatter plot of both variables shows how the individuals are spread.





It is also a good practice to check the correlation between variables. The Correlation Matrix plotted in the following figure shows a tendency of strong correlation pattern among variables between 6 and 11. The PCA technique will be able to handle those as it will find new orthogonal and uncorrelated variables.



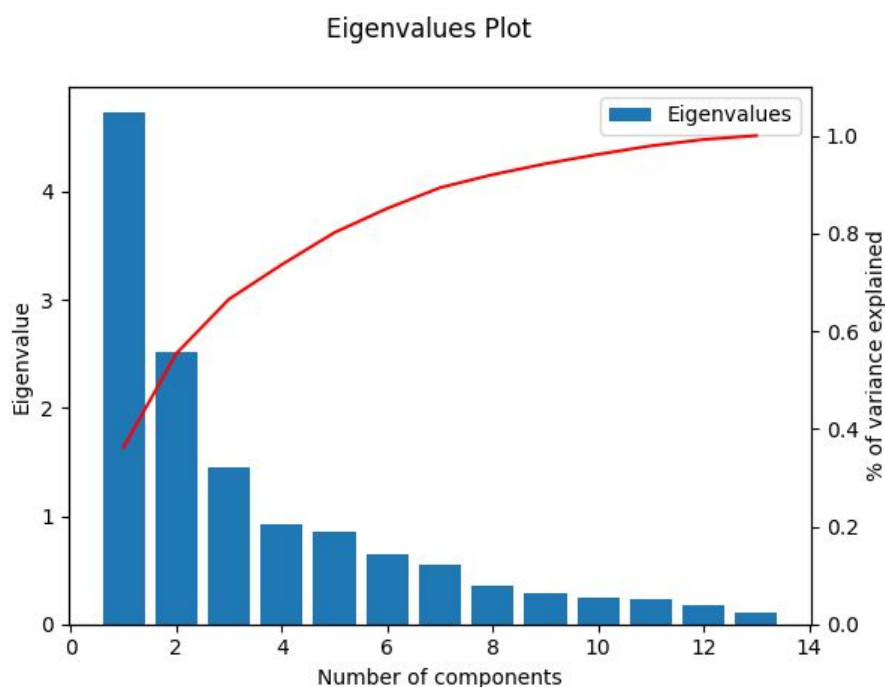
Please note that the diagonal of the correlation matrix has 1.0 coefficient and those were removed in order to improve the visualisation of the color scale.

## Eigenvalues

When applying the PCA algorithm, the first step is to obtain the eigenvalues and eigenvectors of the dataset containing the variables centered, and if necessary, also standardized.

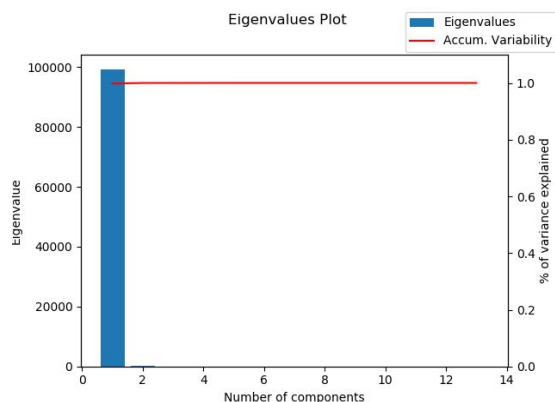
The eigenvalues can be interpreted as the percentage of the variance explained by the corresponding eigenvector. In this particular dataset, the first component explains 36.2% of the variability, the second 19.2% and the third 11.1%. In other words, by using the first 3 components (PC1, PC2 and PC3) the 66.5% of the variability is considered. As the aim of the analysis is to have a broad overview of the data and how the cloud of points is represented in a subspace using projections, the selection of 3 components for reducing the data dimensionality is quite sensible. Keep in mind that the extra 10 dimensions provides just one third of the remaining variability.

The next figure shows a visual representation of the eigenvalues. The first component is the most important as it captures more than a third of the variability indicated by the accumulated variability line in red. The second and the third are also relevant but from the fourth and on, the added information may not be worth the fact of adding a dimension. Again, this is a criteria depending on the objective of the analysis and there is no a hard rule to follow.



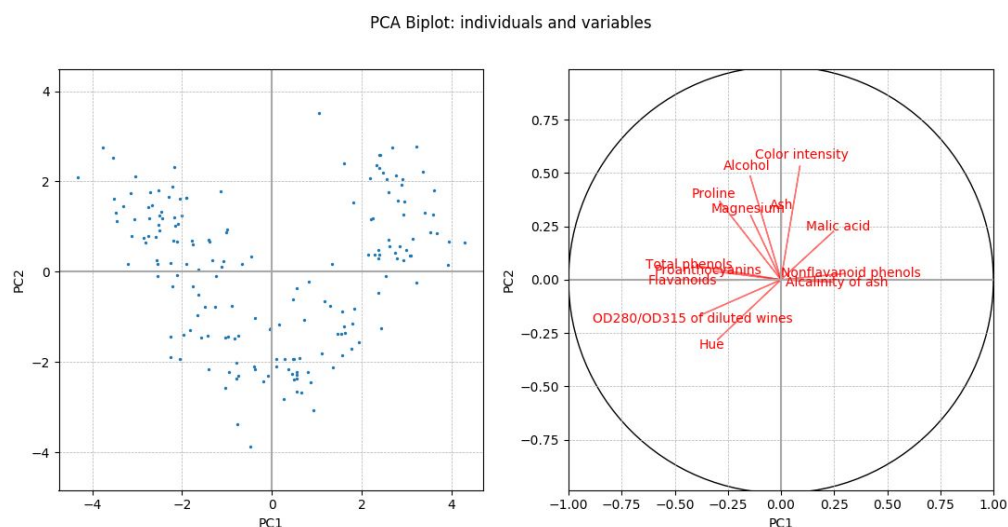
The selected eigenvectors will then forms the basis of the new space where the individuals will be projected.

The next figure shows, the same eigenvalue plot obtained if the standardization step was removed (variables only centered). The issue is that the variable with the highest variance due to unit scale influences negatively the entire analysis.



## Principal Components

After transforming the values of the individuals (wine samples) into the new space, the new coordinates can be plotted along with the eigenvectors. Next figure shows the results for the first two components.



As expected, the PC1 and PC2 capture most of the dispersion. The cloud of dots shows a V shape inside a cube which the horizontal axis captures the width and the vertical the high of the shape. Now we are staring the cube from the front face.

The variables plot on the right, helps to interpret the meaning of the axis which may form a latent variable, not originally captured in the dataset. The longer the variable line, the better the axis represents them. The angle between the lines (axis or other variables) represent the correlation between them.

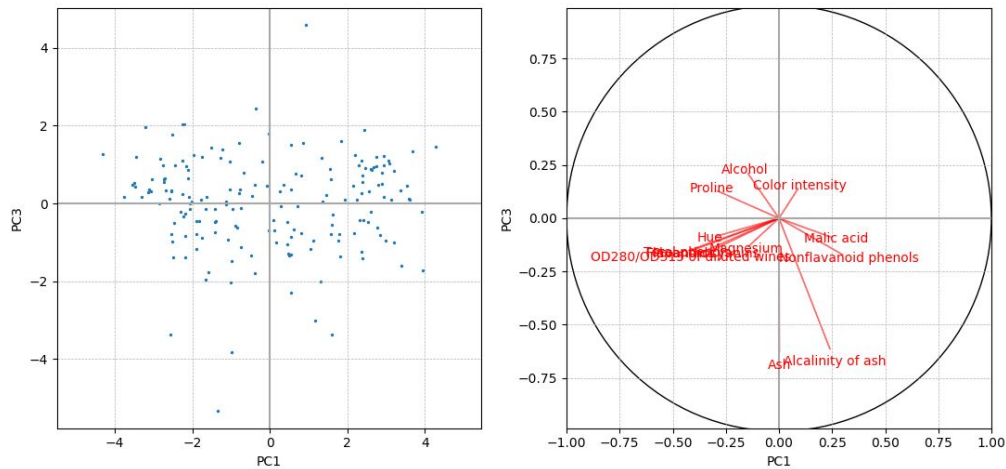
PC1 tend to differentiate the wines with low Total phenols, Flavanoids and Proanthocyanins from the ones with high Nonflavanoid phenols. On the other hand, PC2 tend to differentiate the wines with Alcohol and color intensity.

The next figure shows a second point of view of the cloud of points. In this case we are facing the imaginary cube from a top view. The dots tend to be concentrated in the arms of

the V shape rather than the center. This view helps to see that PC3 captures the differences Ash and Alkalinity of ash.

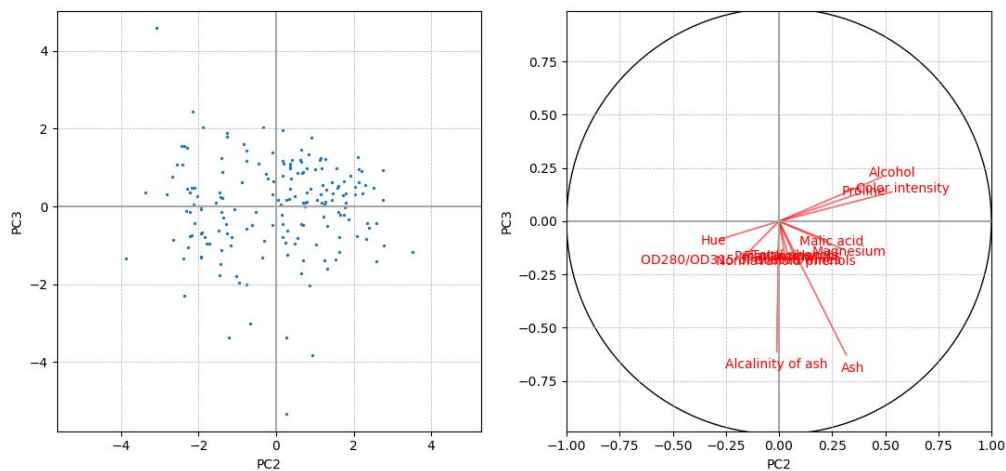
There are also 6 outliers which have PC3 values greater than 2.5 in absolute terms that must be good contributors to the overall variability.

PCA Biplot: individuals and variables



The remaining point of view of the imaginary box is the the side view (with the front facing down). It confirm the correlation between PC3 and the ash variables. It also confirm the relationship between the Alcohol and the Color intensity together with the Proline variable. The individual plots also shows the outliers but the rest of the samples are concentrated in the center.

PCA Biplot: individuals and variables



Although plots and the analysis helps to see the relationships between the samples, it would be good to have an expert of the field which could help the data miners to interpret the meaning of the relationships and any implicit variables.

## Balance

In the previous work we studied the k-means algorithm and variants performance with the Balance dataset. This data set is obtained from an experiment, variables are not correlated, therefore applying PCA was not interesting. Instead, we studied the Bupa dataset.

## Bupa

### Dataset properties

The bupa data set variables have the following features with the following characteristics:

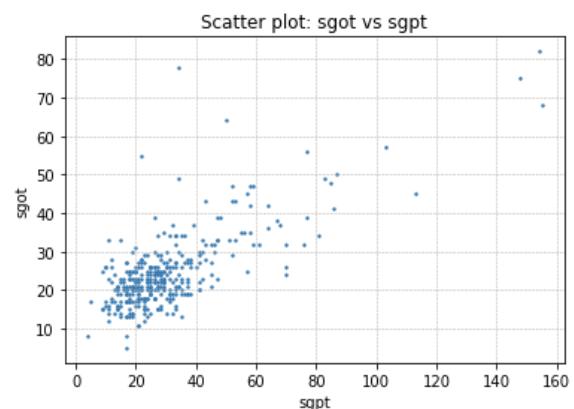
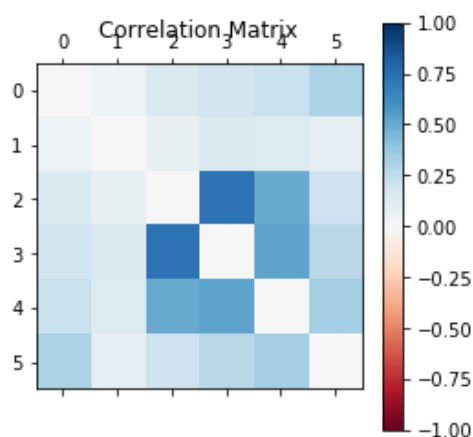
id	Features	Min.	Max	Median	Mean	Std. Dev
0	MCV	65.00	103.00	78.72	90.16	4.44
1	alkphos	23.00	138.00	35.72	69.87	18.32
2	sgpt	4.00	155.00	7.88	30.41	19.48
3	sgot	5.00	82.00	8.00	24.64	10.05
4	gammagt	5.00	297.00	5.00	38.28	39.20
5	drinks	0.00	20.00	0.00	3.46	3.33

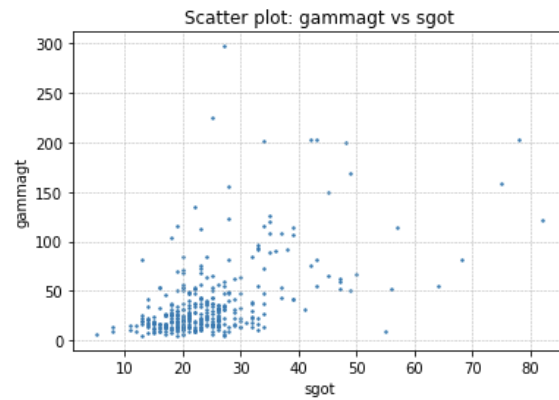
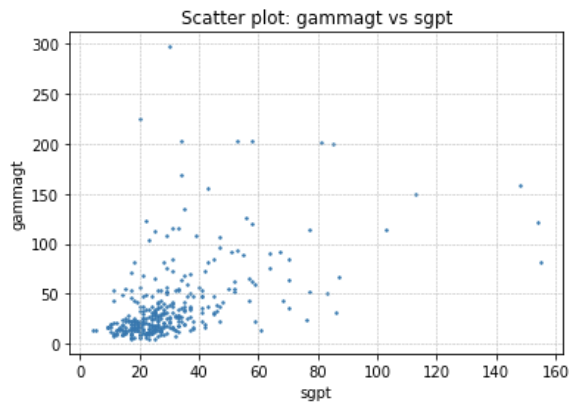
Each input of the data set has 6 features, applying PCA can allow to reduce dimensionality.

The values are not normalized, ranges for each feature vary. However, we will first run the PCA without normalizing, just subtracting the mean as described in the algorithm.

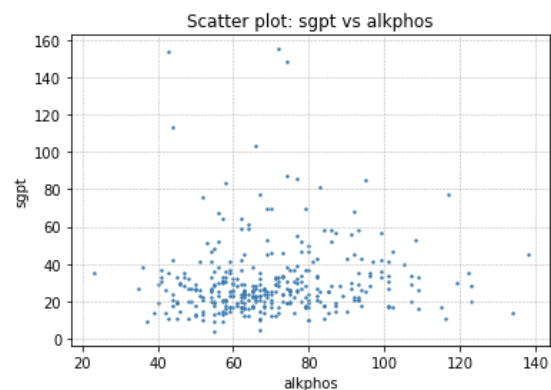
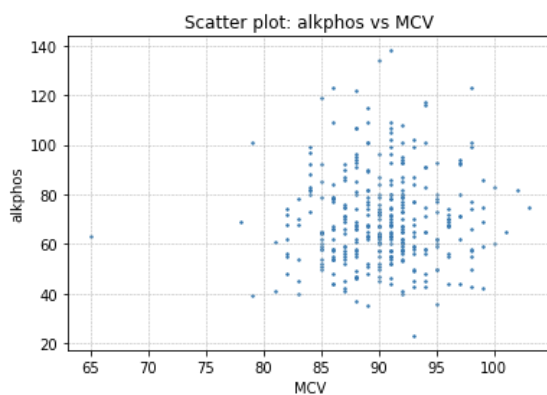
Gammagt, sgt and sgot might be the dominant components in PCA as they show higher standard deviation values.

The correlation matrix shows that all vars are positively related but there is only a high correlation between vars sgpt, sgot (2,3); sgpt,gammagt (2,4) and sgot,gammagt (3,4).





Features with lower correlation. MCV, alkphos(0,1) , alkphos,sgpt(1,2):



## Calculating PCA

In order to proceed with PCA, we first subtract the mean of each feature in the original data set and calculate the covariance matrix, using numpy function `np.cov(data,rowvar=0)`.

The covariance matrix values are:

```
[ 19.78   3.59  12.82   8.41  38.82   4.65 ]
[   3.60 336.64  27.28  26.97  95.89   6.17 ]
[  12.82  27.28 380.73 145.26 385.61  13.47 ]
[   8.40  26.97 145.26 101.29 208.45   9.39 ]
[  38.82  95.89 385.60 208.45 1540.92  44.71 ]
[   4.64   6.17  13.47   9.39  44.71  11.14 ]
```

Covariances of features (2,3), (2,4) and (3,4) have the highest covariance values, therefore they will be the dominant values of the PCA components (higher eigenvalues).

Then, we obtain the eigenvectors and the eigenvalues with the function available in numpy linear algebra module: `w, v = LA.eig(covars_matrix)`.

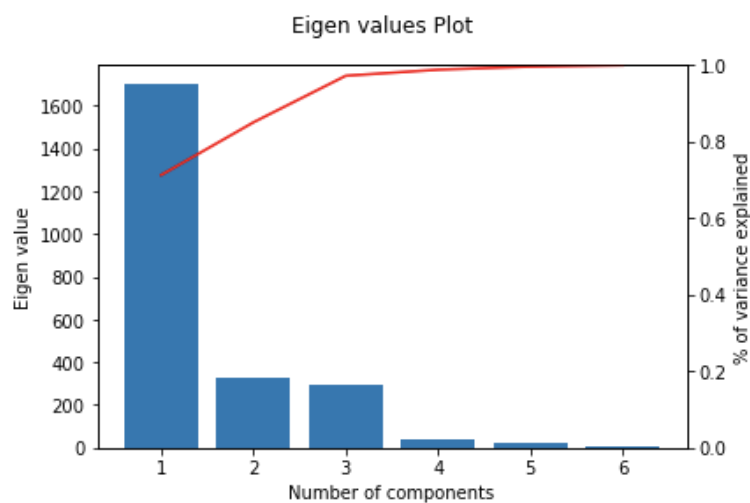
We now need to sort the eigenvalues (and the eigenvectors accordingly) from higher to lower values. This will give us an array with the components with the components with higher variance in the first positions.

We obtain the following Eigenvalues::

```
[ 1704.23  329.48  290.92  37.76  19.59  8.51]
```

and Eigenvectors:

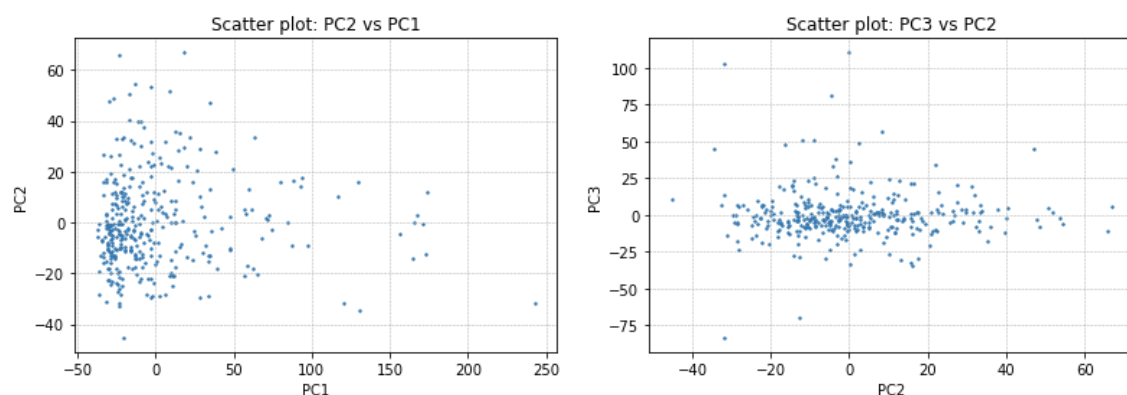
```
[ 2.48e-02  1.98e-03  4.94e-03  1.08e-01 -9.43e-01  3.11e-01]
[ 7.49e-02  9.95e-01 -3.81e-02 -4.14e-02  1.32e-03  6.73e-03]
[ 2.92e-01 -2.31e-03  8.89e-01 -3.49e-01 -3.21e-02 -1.31e-02]
[ 1.50e-01  3.88e-02  3.19e-01  9.24e-01  1.28e-01  5.10e-02]
[ 9.40e-01 -8.51e-02 -3.24e-01 -4.14e-02  2.33e-02  1.57e-02]
[ 2.83e-02  8.42e-03  8.68e-04  8.91e-02 -3.01e-01 -9.48e-01]
```



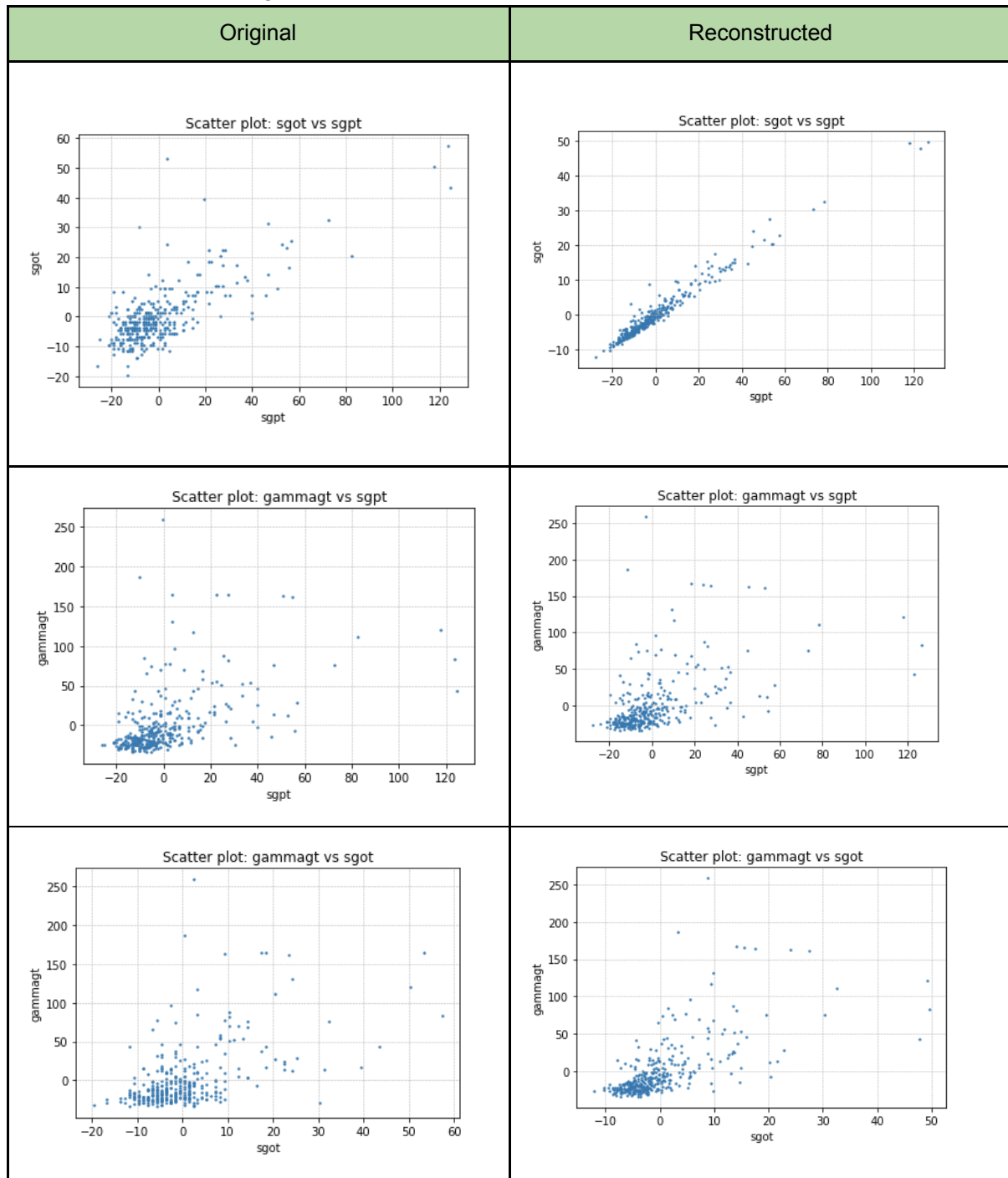
The figures shows that first Principal component is the most dominant with over a 70% of the total variance, components 2 and 3 also bring information on the variance of the data with values around 10-15% and the last three components variation is very small.

We can conclude that we can reduce the dimensionality of the data to 3 ( $k=3$ ) (total data variance >90%).

We use the three first principal components to calculate a the transformed data:



If we reconstruct the original data we obtain:

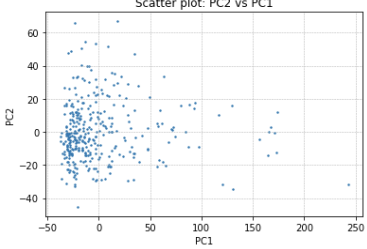
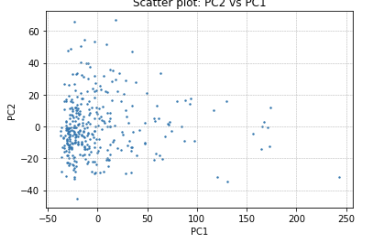
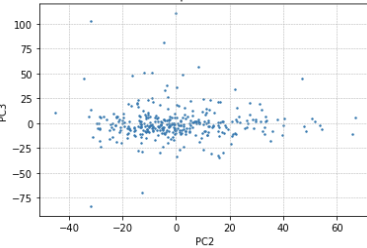
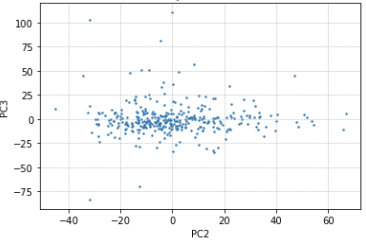
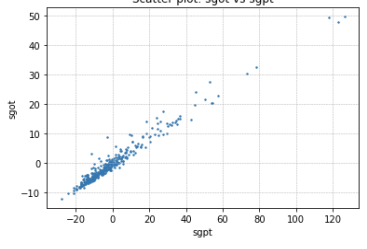
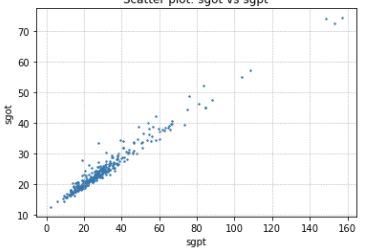
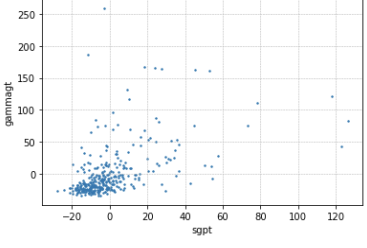
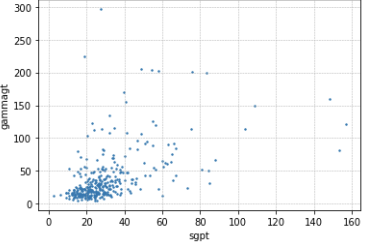
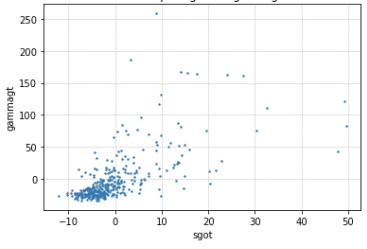
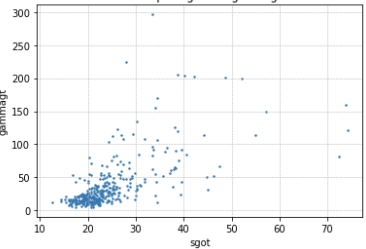


In the first pair of images we can see how the transformation has linearized the relation between the two features.

In for the other two pairs we can see the data the points are a bit more compressed. (The shift of -20 for the first two pairs and -10 for the last one, with regards of the figures in the first section is due to subtracting the mean value of the features).

Finally, we compare the results with the ones obtained with `sklearn.decomposition.PCA`,  $K$  is set to 3.



IML-PCA	sklearn.pca
Principal components	
	
	
Reconstructed data	
	
	
	
Eigenvalues	
$\lambda_1=1704.23$ $\lambda_2= 329.48$ $\lambda_3= 290.92$	$\lambda_1=1704.23$ $\lambda_2= 329.48$ $\lambda_3= 290.92$

# Conclusions

PCA allows us to reduce the dimensions of our variables by finding the correlation between the set of features and transforming them into a set of uncorrelated components. Then, it keeps the components that capture most of the information. There is a trade off between the variability explained and the number of dimensions selected, the right value will depend on the aim of the analysis.

The PCA analysis showed how our custom implementation differed from the SKlearn one, due to the different method of calculating the eigenvectors, our implementation uses the function `numpy.linalg.eig` whereas the `sklearn.decomposition.PCA` uses `scipy.linalg.svd`<sup>1</sup>. Specifically, we found a difference in the sign of the eigenvectors, same values but opposite sign for each vector component.

The iris data-set was small and low dimensional and had 92% of its variance captured in a single PCA component. This would allow the reduction of the dimensions from 4 to 1, although we used 2 components for the purpose of analysis. It did not give more advice for knowing the underlying information as the data-set was simple to classify from the beginning, and the few dimensions already showed a strong correlation.

The wine dataset showed the importance of preprocessing the data as it was crucial to standardize the variables before applying PCA. As the scope of the analysis was to have an overview of the data and observe how the cloud of points can be represented in a subspace using projections, 3 components were selected. Those captured two thirds of the total variability. In short, PC1 tends to differentiate the wines with low Total phenols, Flavanoids and Proanthocyanins from the ones with high Nonflavanoid phenols. On the other hand, PC2 tends to differentiate the wines with Alcohol and color intensity while PC3 captured the differences in Ash and Alcalinity of ash variables. Although analysis helped to see the differences between the samples, an expert of the field is required to interpret the meaning of the relationships and any implicit or latent variables.

For the Bupa dataset, we could see a strong correlation among three of its six features, and a low correlation among the rest. After calculating the eigenvalues we could see a predominant component which contained 70% of the total variance, followed by two more components with 10-15% of the variance. From this information we decided to reduce dimensionality to three. Comparing the results of our implementation with the sklearn `pca` function we obtained the same eigenvalues. When analyzing the previous dataset we observed a different sign in one of the eigenvectors, in this case however we did not find this difference, the three eigenvectors associated to the higher eigenvalues matched. In order to find the reason for this difference, we ran the algorithm with `k=6`, and we could see that the last eigenvector was the one with the difference in the sign.

---

<sup>1</sup> <https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/decomposition/pca.py#L436>

# Code execution

File: Work2.ipynb

Code can be executed using the scripts in Work2.ipynb. The code was implemented using Jupyter notebook. Run all the cells from top to bottom to obtain results. Make sure the datasets folder is present.