Visualization Mini project-2

<https://prathak07.github.io/Visualization-Mini-Project-2/>

DATASET: **European Soccer Database** (Player Attributes for European Professional Football) containing **10849 rows and 36 columns**

Architecture of the Project:

1. The data is processed using Python code inside “**src**” folder : **GenVisData.py**
2. The python code process the ‘**Football\_Players.csv**’ inside “**data**” folder, which is passed as argument during runtime.
3. After processing it creates processed data in folder “**data/processed/**” containing all the files which can be used for plotting the various graphs using D3.js
4. Each graph in “**index.html**” can be plotted by clicking the respective buttons, and each graph plots random as well as stratified sample data in single plot. **(Random: Green ; Stratified: Red)**

Main Function of Python Code:



Figure 1 Main function of python code

**Task1: data clustering and decimation**



Figure 2 Code for Random and Stratified Sampling

♣ implement random sampling and stratified sampling

♣ the latter includes the need for k-means clustering (optimize k using elbow)

Random Sampling and Stratified Sampling is performed using python.

For stratified sampling first value of k is calculated using elbow method as below.

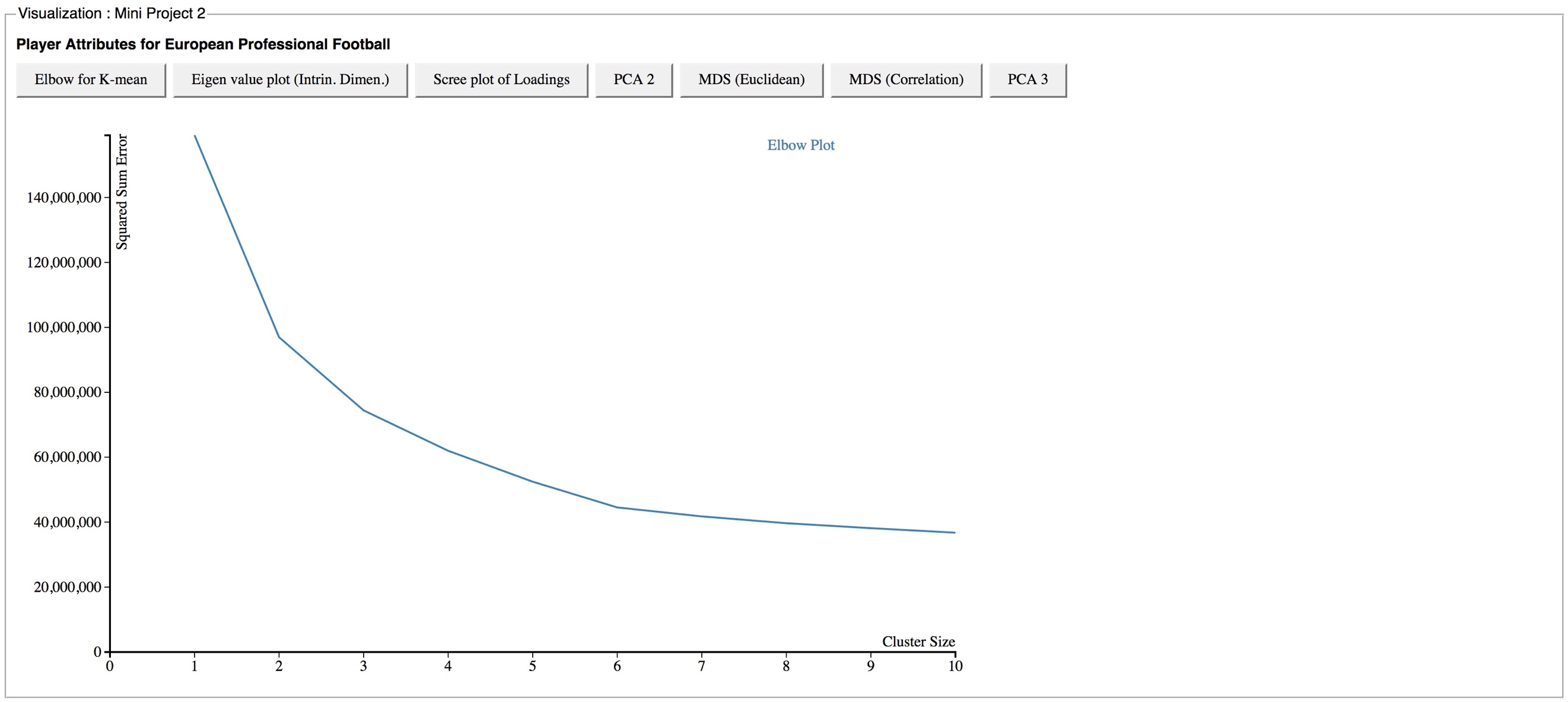


Figure 3 Elbow graph for calculating K for K-Mean clustering

From the above graph, it can be seen the number of clusters to be made is 3. Thus, for stratified sampling number of clusters is passed 3 and fraction of 0.05.

Task 2: dimension reduction (use decimated data)

♣ find the intrinsic dimensionality of the data using PCA

♣ produce scree plot visualization and mark the intrinsic dimensionality

♣ obtain the three attributes with highest PCA loadings



Figure 4 Python code for finding the eigen values for each dimension

Above code helps in plotting the Eigen values. Loadings for each of the dimensions of the data is then calculated from it. By checking the Eigen value plot Intrinsic Dimensionality of the data is 8 as at that point the graph becomes flat like x-axis. Below is the graph showing the Eigen Value plot.

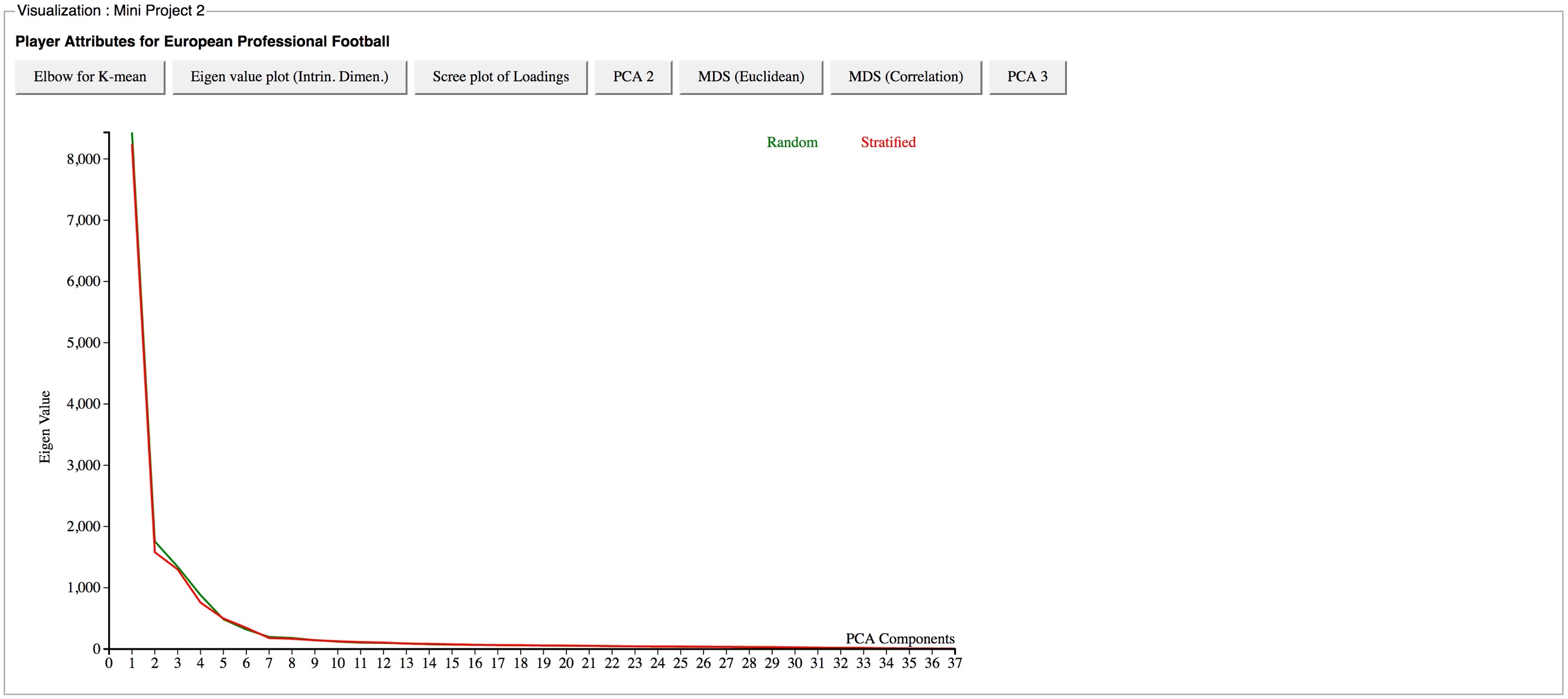


Figure 5 Eigen Value Plot

Below is the graph of Loadings for each of the dimensions of the data, where we can clearly see the first three dimensions corresponds to top 3 loadings.

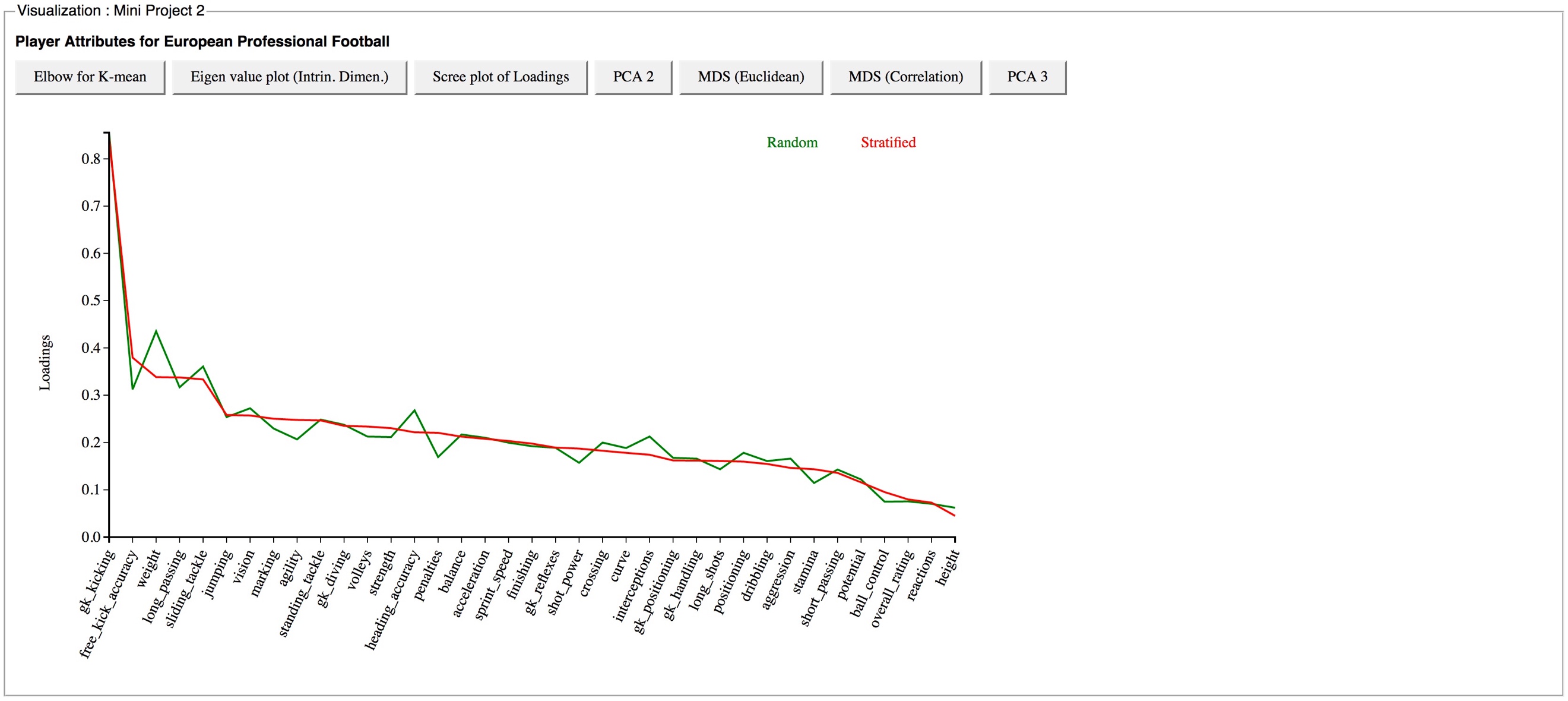


Figure 6 Scree-PLot showing Contribution by each Dimension of the Data

Python Code output showing k of k-means and intrinsic dimensionality…

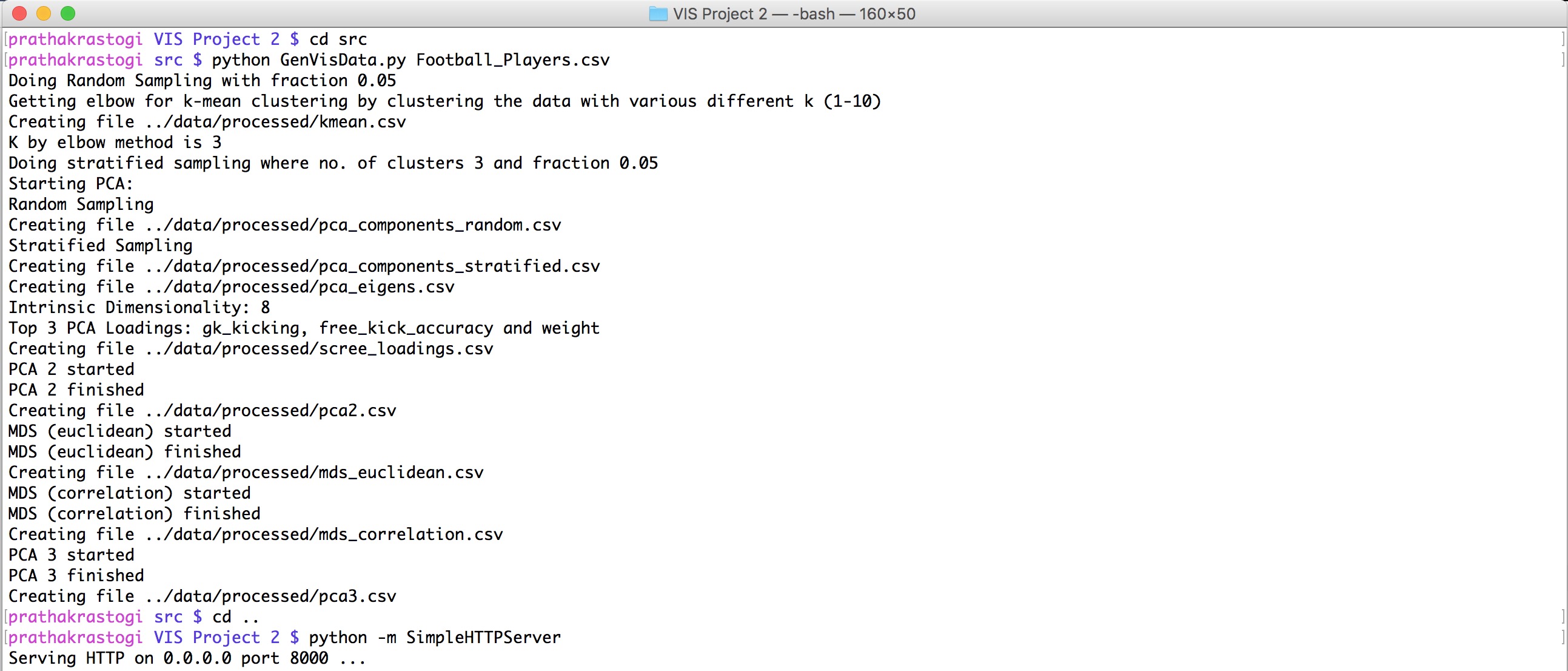


Figure 7 Output of Python Code

Task 3: visualization (use dimension reduced data)

♣ visualize data projected into the top two PCA vectors via 2D scatterplot

♣ visualize data via MDS (Euclidian & correlation distance) in 2D scatterplots

♣ visualize scatterplot matrix of the three highest PCA loaded attributes

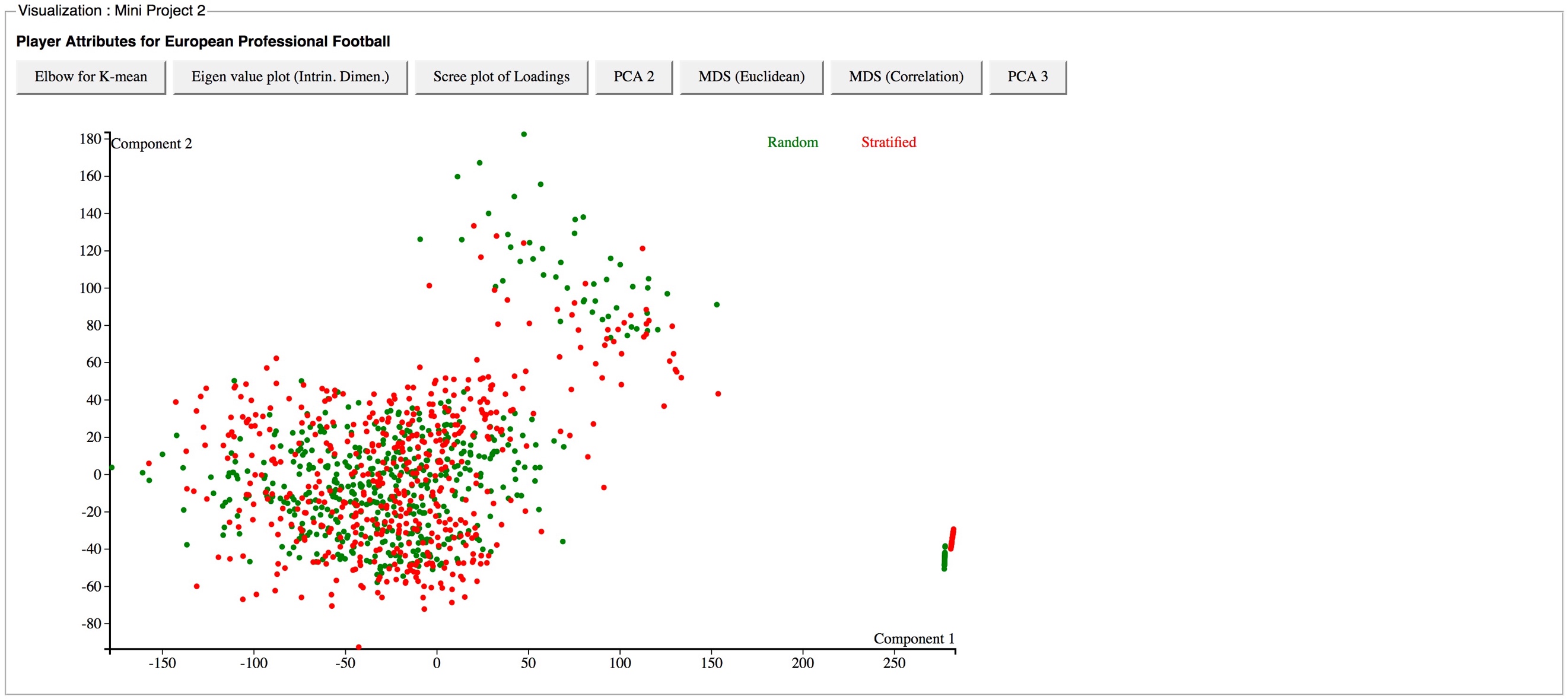


Figure 8 Data Projected on 2 PCA vectors



Figure 9 Python Code for Projecting data on 2 PCA Vectors

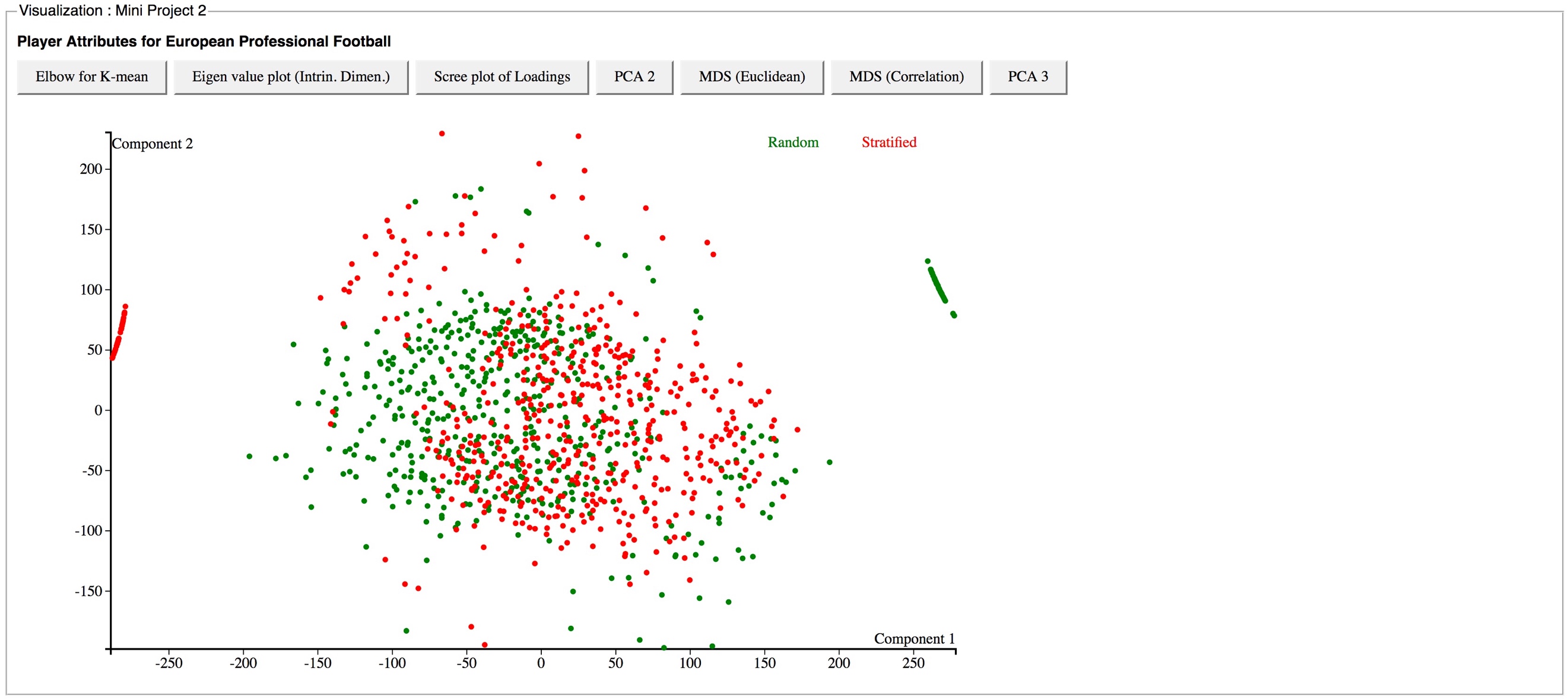


Figure 10 Data Projeced using MDS (euclidian distance)



Figure 11 Python code for projecting data using MDS (Euclidean Distance)



Figure 12 Data Projected using MDS (Correlation Distance)



Figure 13 Python Code for Projecting data using MDS (Correlation Distance)



Figure 14 Scatterplot Matrix of 3 of the highest loadings



Figure 15 Python code for generating the data for ScatterPlot Matrix

### Some Observations

1. Having a large data can provide more detailed info about the pattern in the data, but increase the amount of time it takes to process the data.
2. Having the small fraction of data by random sampling out of original data can significantly change the outlook the data provides.
3. In Football player data, all the fields had some importance as most of the columns are characteristic property of the player.
4. I replaced the blank part in data with 0, but if it is replaced by mean then top 3 loadings may change.
5. Various methods can be used to find the k of k-means like information criteria method, x-means clustering etc. each providing more efficient way to find k for k-means.

### References:

1. <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
2. <http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html>
3. <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>
4. <https://bl.ocks.org/mbostock/3213173>
5. <http://bl.ocks.org/weiglemc/6185069>
6. <https://bl.ocks.org/mbostock/3883245>
7. <http://bl.ocks.org/d3noob/b3ff6ae1c120eea654b5>
8. <http://bl.ocks.org/d3noob/8603837>