Homework of Principal Components Analysis Report

Polytechnic University of Turin

Computational linear algebra for large scale problems - 01TWYSM

Caffaro Fabio
*s*276842

1 Extraction of the Working dataset

The first part of the homework it's straightforward. It simply requires to load the complete dataset from the csv file *'fifa19datastats.csv'*, create the working dataset that we will use later and save it.

I have decide to perform this homework using python language, so to do this part I have used pandas library, and in particular pandas.load() and pandas.to_csv() functions.

In the first part is also required to extract a random subsample of the entire dataset using our university registration number as seed. To retrieve indexes avoiding repetition I haveve used the following function:

```
seed = 276842
ind = np.random.default_rng(seed=seed).choice(df.shape[0],
    ↪ size=10000, replace=False)    # random integers without
    ↪ repetition
```

Finally, I have extracted the column names of the dataset, separating the column names corresponding to the skills (columns from 6 to the end) to the others. This will be useful later to extract the skills values and to reconstruct the dataset.

2 Application of PCA

For this part of the homework, first I have retrieved the matrix containing the values of the skills using the skill's column names extracted before. Then, to choose the algorithm to apply I have decided to analyze the attributes in the matrix.

During the lessons, we have seen different versions of PCA algorithm. The main distinction among them are in the standardization method of the columns, using mean and standard deviation or the mean only, and in the method used to retrieve eigenvalues/eigenvectors of the variance-covariance matrix, using methods to do an eigenvalue decomposition or using a much more efficient SVD decomposition.

All the skills in the dataset are float numbers ranging from 0 to 99, so they all have the same scale. For this reason, my first thought was to use the version that consider only the mean and not the standard deviation. However, I thought that possibly even if they all have the same range, not all attributes follow the same distribution. For example, Goalkeeper's skills have a lower mean with respect to the other skills, due to the lower number of goalkeepers present in

the dataset (fig. 1). The same could have also been possible to the standard deviation.

As it is possible to see in the picture, in fact the skills have different distribution. For instance, the last 4 ones (referring to goalkeeper's attributes) are less skewed and with a much lower mean with respect to the other skills. However, analyzing further the distribution with the help of `pandas.DataFrame.describe()` function I've seen that despite the attributes have different means, all the standard deviations are quite similar ranging from around 10 to 20. For this reason I have decided to use the mean-only version of pca with SVD decomposition.

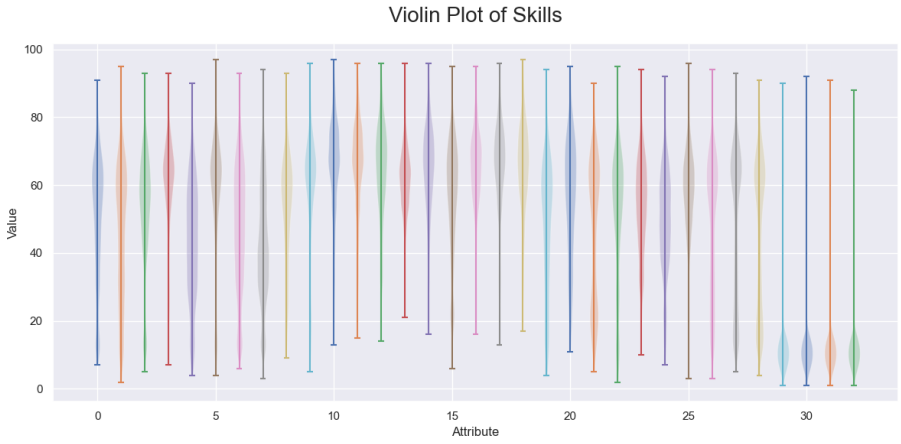

Figure 1: Violin plot of all the attributes. The last four are the one related to goalkeeper position

Note that with this choice we are weighting a bit more the attributes with more variance.

2.1 PCA implementation

In order to have a cleaner code I have decided to implement a `class PCA`, in which I have put all the functions to perform pca algorithm, approximate the dataset, restore the original dataset, etc. The class is structured in a scikit-learn fashion (`fit, transform, inv_transform`). Moreover, I have added some utility functions to plot graphs, compute the number of components and so on. I have defined a variable `PCA` disabling the usage of the standard deviation and setting the components to 0.9 variance explained (as required later). Then I've retrieved the reduced dataset with the new attributes corresponding to the principal components, using `fit_transform` function:

```
pca = PCA(n_components=0.9, use_mean= True, use_std=False)
W = pca.fit_transform(X)
```

3 Interpretation of main Principal Components

3.1 Selection of number of components

Regarding the first request of this section, I have defined PCA class to take `n_components` parameter both as int or as float as in scikitlearn. In the latter case, the value will be used as the minimum variance to explain, in order to compute the corresponding number of principal components needed to do that. This computation is implemented in an inner function of PCA class:

```
def __find_components(self, n_components):
    tot_variance = np.sum(self.lambdas)
    variance_explained = [x / tot_variance for x in self.lambdas]
    cum_variance = np.cumsum(variance_explained)

    if type(n_components) is int:
        self.n_components = n_components
    else:
        self.n_components = np.argmax(cum_variance >= n_components)
        ↪  + 1

    self.var_explained = cum_variance[self.n_components -1]
```

First I compute the total variance of the dataset summing all the singular values. Then I compute the percentage of variance explained of each singular value and the cumulative percentage of variance explained. If `n_component` is integer I save it and setting the variance explained as the corresponding cumulative sum. If `n_component` is float, I use `np.argmax` to find the first element of the cumulative sum vector larger or equal to the variance to explain. The plus one is to set the components to the real number of components, while the minus one to consider the starting index 0. Setting PCA with `n_components = 0.9`, the corresponding number of principal components required to have at least 0.9 variance explained are 7 (fig. 2).

3.2 Plotting bargraphs

To plot the horizontal bargrpahs I have implemented in the PCA class a wrapper function `plot_pc_horizontalbargraph`. It takes a `component` parameter to specify the component that you want to plot. The parameter is setted by default

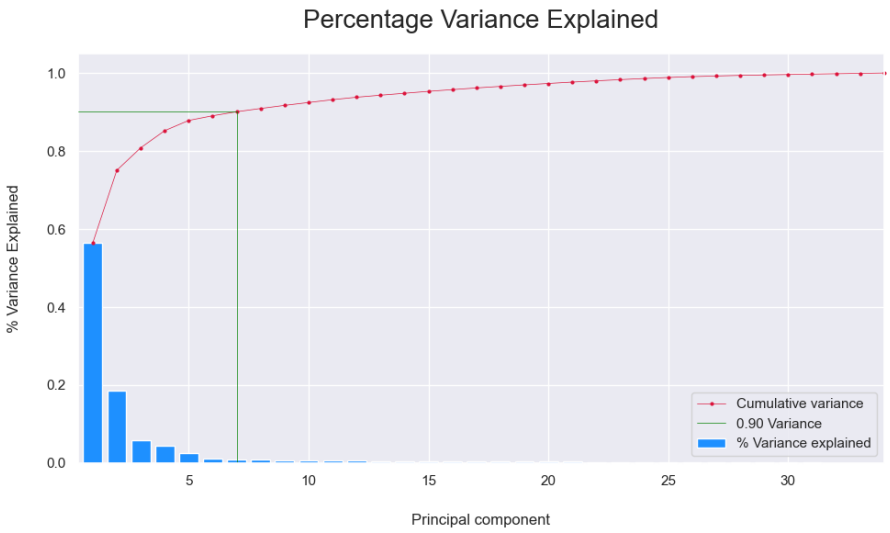

Figure 2: Variance explained ratio for each singular value and corresponding cumulative variance explained. Highlighted in green 0.9 variance, corresponding to the cumulative sum of the first 7 p.c.

to 'all', in this case all the components are plotted. The plots are constructed in the private function `__plot_pc`

3.3 Naming the Principal Components

Giving interpretation to the principal components is usually not a trivial task. Since principal components are a linear combination of the original attributes, sometimes is not even possible to clearly explain them. Consider for example a dataset containing health data, with attributes like age, height and temperature and a principal component that is a linear combination of these attributes, like for example $0.8 * age - 0.2 * height + 0.3 * temperature$. Is not easy to find a real concept that express this attribute. Nonetheless using bargraphs we can see how much each original attribute contribute positively or negatively to a principal component and this can give us insight on the data.

For example the first principal component is clearly separating goalkeepers to all the other roles (fig. 3). The last five attributes (**GKDIving, GKHandling, GKKicking, GKPositioning and GKReflexes**) contribute positively to this component, while all the other attributes negatively. This means that for high values of the first principal component we are looking at goalkeepers, while for low values we are looking at non-goalkeeprs (defenders, midfielders or strikers). Roughly speaking this is a 'goalkeeperness' attribute, so we can call it **'Goalkeeper'**. Note that, it's reasonable that this is the first principal component. In fact the major differences in the dataset are exactly between goalkeepers that have their own attributes and the rest of the players.

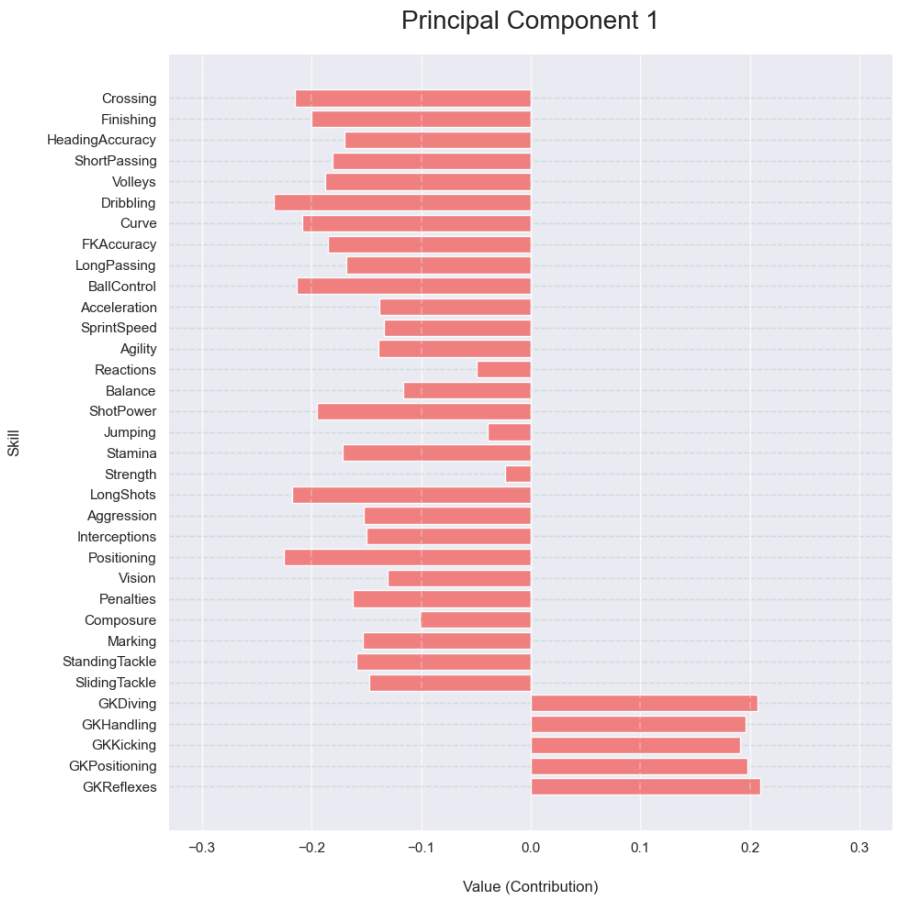

Figure 3: Horizontal bargraph of the first principal component

The second principal component has negative contribution from attributes like **Aggression**, **Interceptions**, **Marking**, **StandingTackle** and **SlidingTackle**. This are all clear **defensive attributes**.

On the other side, in the positive contributors we can see attributes as **Finishing**, **LongShots**, **Positioning**, **Penalties** and **Dribbling**. These instead, are all clear **offensive attributes**. So, for high positive values of the second principal component we are looking at more offensive players, while for high negative values we are looking at more defensive ones. In conclusion we can consider the second principal component as a **'Defensive/Offensive'** player discriminant.

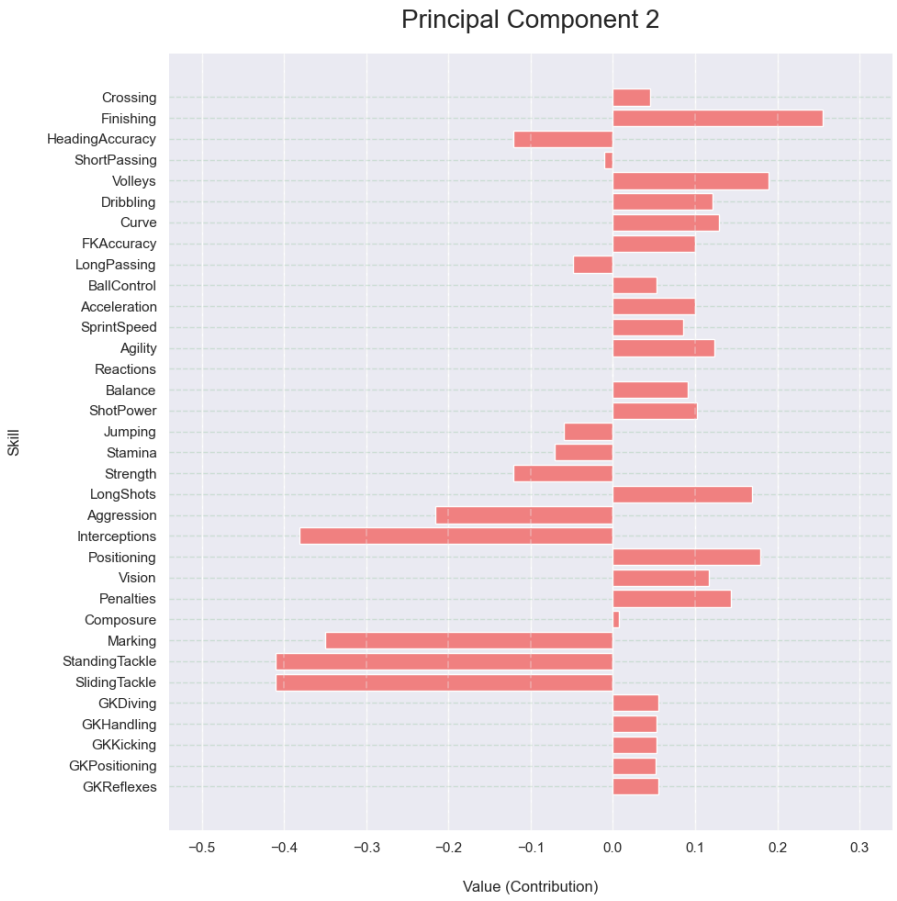

Figure 4: Horizontal bargraph of the second principal component

In the third principal component, we find that in the negative contribution side we have almost all the attributes, while from the positive ones stand out **Acceleration** and **SprintSpeed**. With the help of the image provided, we can see that these two attributes are referring to **Pace**. This component can be seen as a **'Pace'** attribute.

Note that there is a huge negative contribution by the goalkeper's attributes. GK are the only static players they have very low Pace.

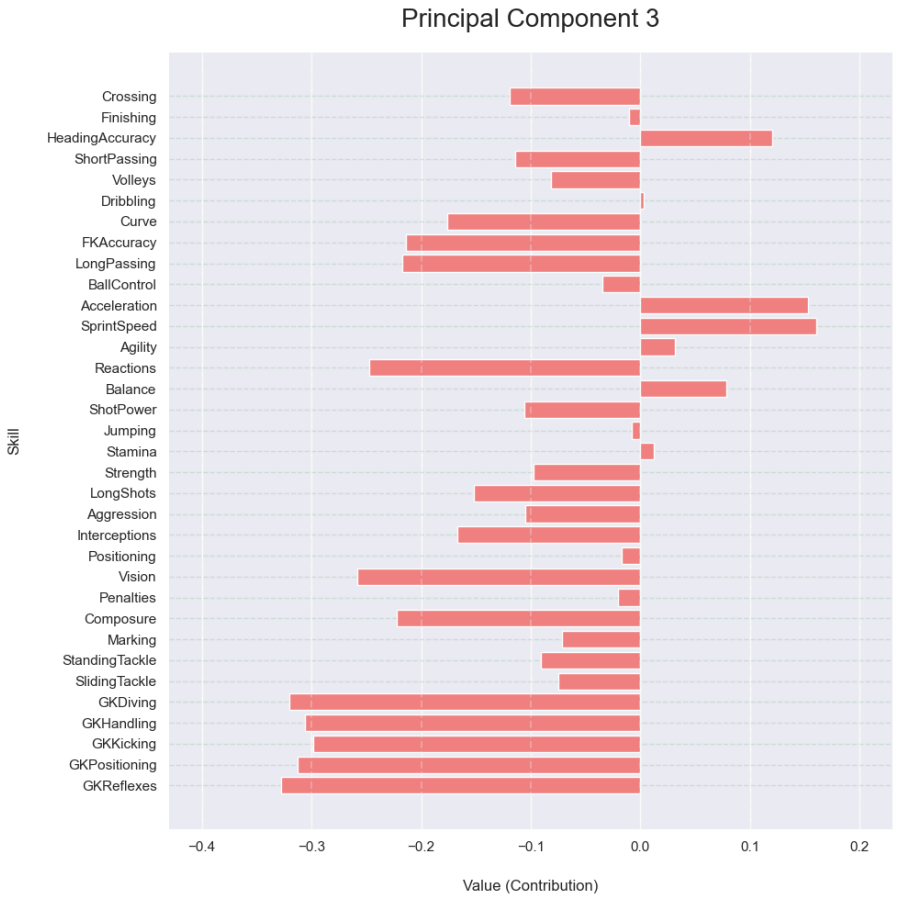

Figure 5: Horizontal bargraph of the third principal component

Principal component number four is characterized negatively by **Volleys, Strenght, Finishing, ShotPower, Penalties** and **HeadingAccuracy** that are all **Shooting attributes** and in particular attributes more related to **strikers**.

On the other hand, there are **Crossing, Curve, Long/ShortPassing** that are **Passing attributes** and **Agility, Balance, Dribbling** that are **Dribbling attributes**. These latter two categories are more related to **midfielders**.

Then, for high negative values of the fourth component we are looking at strikers, while for high positive values at midfielders. We can interpret this principal component as a **'Forward/Midfielder'** attribute

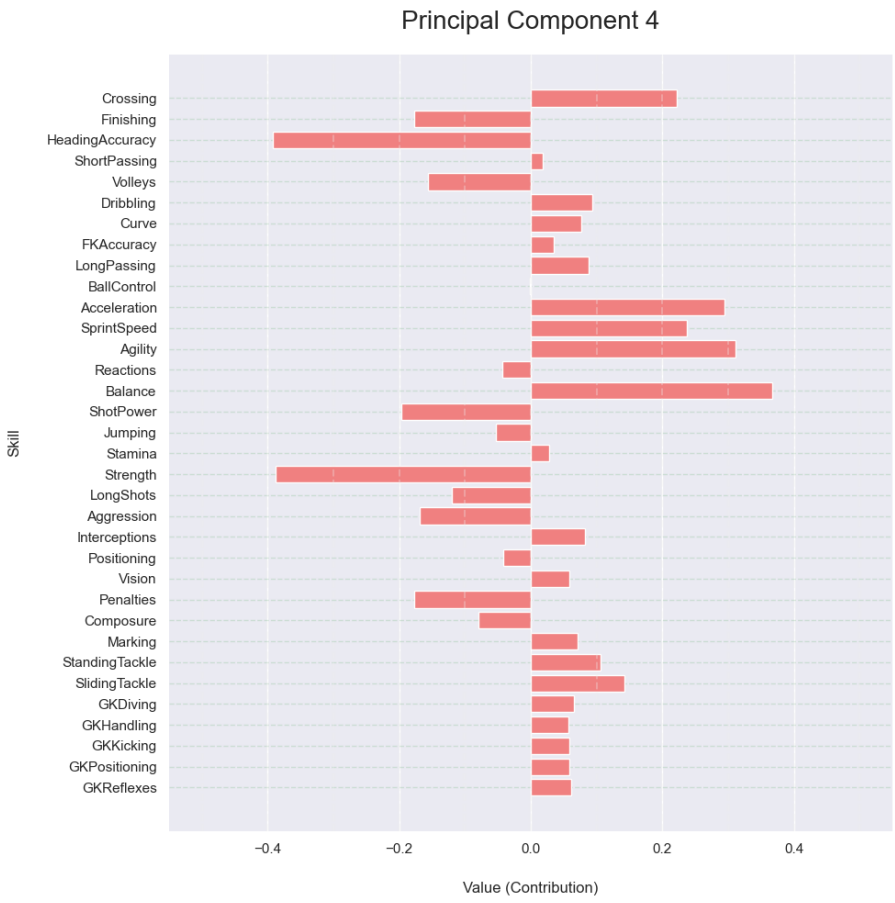

Figure 6: Horizontal bargraph of the fourth principal component

Starting from the fifth principal component, things appear to be more hazy. In this case we have **Crossing, FreeKickAccuracy, Curve. LongPassing** and **ShortPassing** on the positive side. This are clearly all Passing and **Technical attributes**.

On the negative side instead, there are **Jumping, Stamina, Strength, Acceleration** and **SpintSpeed** that are all **Physical attributes**. We can define this component as a **'Physical/Technical'** attribute.

Note how, strangely also here are present with a non negligible negative contribution, all the goalkepeer's attributes. Maybe the reason to this is that typically goalkeepers are more physical players and are less able at passing, so we can say less technical.

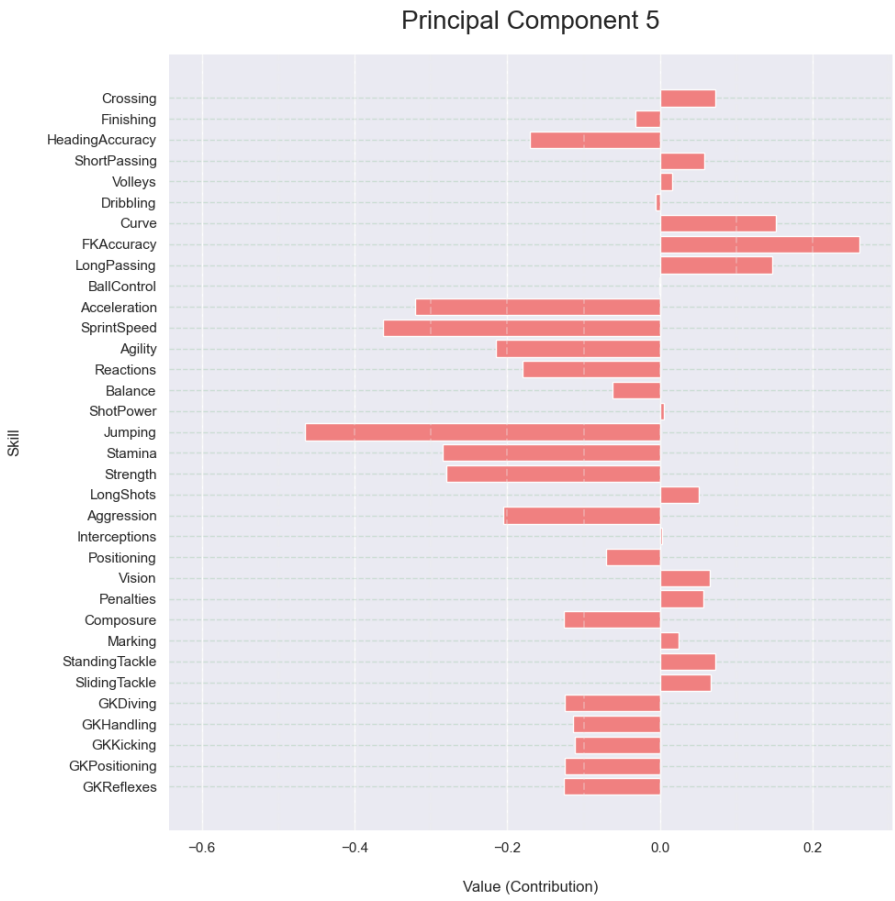

Figure 7: Horizontal bargraph of the fifth principal component

This was the most difficult one, since there are no clear correlations with the original attributes. The contributions are quite mixed. In the positive side the three dominant attributes are **Jumping** (**Physical**), **Balance** (**Physical** and **Dribbling**) and **FreeKickAccuracy** (**Techincal** and **Passing**).

On the negative side there are attributes of all types: and **Passing attributes** as **Crossing** and **LongPassing**, **Dribbling attributes** as **Dribbling** and **BallControl**, **Pace attributes** as **Accelleration** and , **Mental attributes** as **Positioning** and **Vision**. I couldn't find a fit choice so I opted to the attribute with the highest contribution: **'Jumping'**

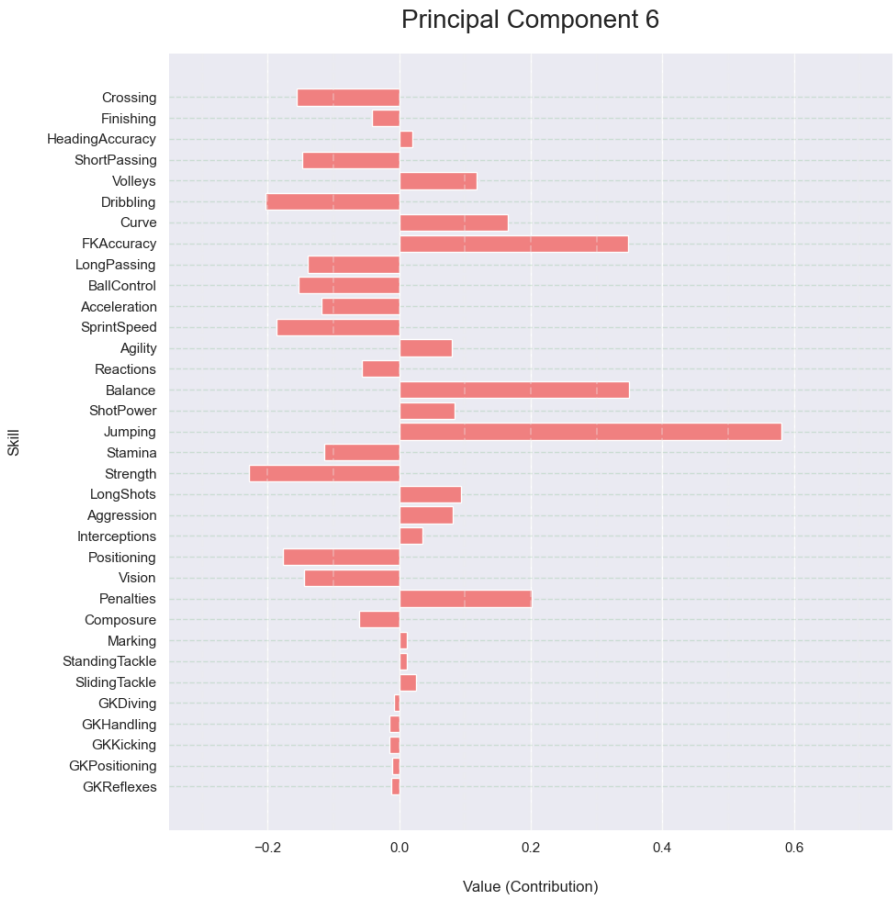

Figure 8: Horizontal bargraph of the sixth principal component

In the last principal component we can see a negative contribution of attributes like **ShortPassing, Dribbling, LongPassing, BallControl** and **HeadingAccuracy**. These are **Technical attributes**. Moreover, there is an important contribution of attributes like **Vision, Composure** that are **Mental attributes**.

On the other hand we have **FKAccuracy, Curve** and **Crossing** that are **Passing attributes** and **SpintSpeed, Acceleration, Strength** that are **Physical attributes**.

Also this, component was not so easy to interpret. However analyzing further the contributions, I have realized that skills like Vision and LongPassing (negative contributors) are usually associated with Center Defensive Midfielders (CDM) also called 'playmakers'. These players have the role to start the actions and dictate the rythm of the game, so usually are referred as Tactical players. I have called this component **'Tactical/Physical'**, but I think it specifically refers to more defensive or more offensive midfielders

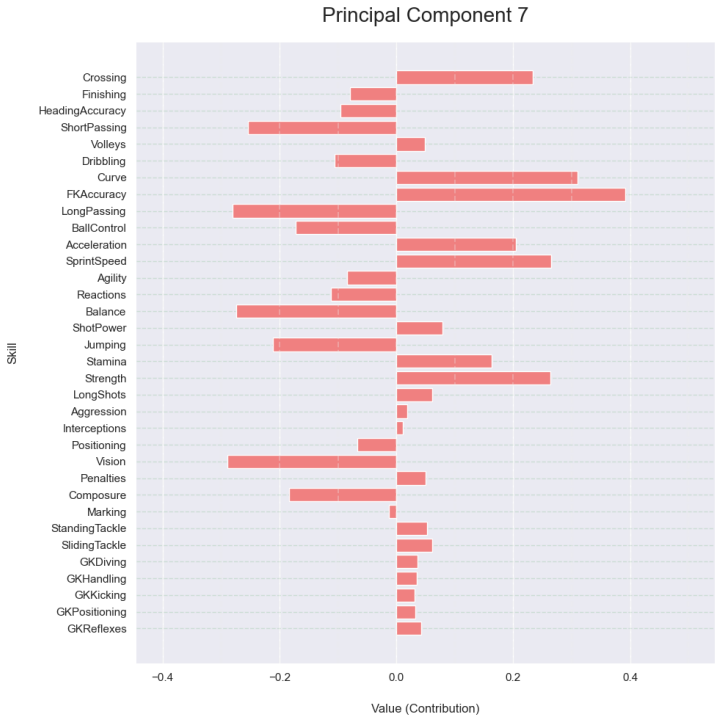

Figure 9: Horizontal bargraph of the seventh principal component

3.4 Scatter Plots

For the last request of the third part of the homework I have plotted 4 graphs (fig. 10). In the upper-left subplot (fig. 10 a) I have scattered the first component (Goalkeeper) against the second (Defensive/Offensive). As told before the first component defines a net separation between the goalkeepers and all the other players. This can be clearly seen in these two pictures, where the red dots representing the goalkeepers are well separated from the rest of the points. If this was a classification task, a good choice would have been to use the first component to filter all the GK players and then analyze further all the remaining players.

An additional considerations: in the upper left subplot we can also asses the correctness of the second component (Defensive/Offensive). In fact, we see a clear vertical distribution, starting from around -100, where are located principally defenders, to 100 where are located mostly forwards. Midfielders are located around 0 and so are goalkeepers.

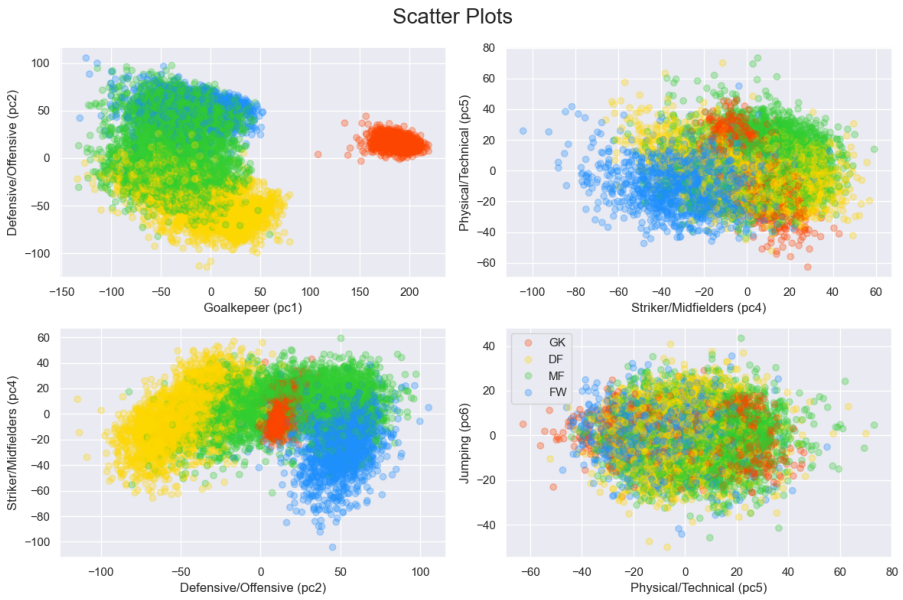

Figure 10: Scatter plots of some principal components. (a) On the top-left pc1 against pc2, (b) on the top-right pc4 against pc5, (c) on the bottom left pc2 against pc4 and (d) in the bottom-right pc5 against pc6

In the upper-right subplot (fig. 10 b), I have plotted pc number 4 (Striker/Midfielder) against pc number 5 (Physical/Technical). In this plot, we can appreciate better the distinction made by pc4 between strikers and midfielders. On

the third quadrant (Strikers and Physical) we find a predominance of Strikers. Instead, on the first quadrant (Midfielders and Technical) there is a prelevance of midfielders. While both strikers and midfielders lie on the main diagonal, defenders are distributed orthogonally to them. We can distinguish two clusters of defenders. The first one (Striker and Technical) probably correspond to wing-backs (LB or RB). These are players that plays on the lateral sides of the pitch and have the role to help during the attacking phase. Typically they have good passing, crossing and shooting abilities, so somehow associated with strikers. The second cluster is in the bottom-right of the plot (Midfielders and Physical). These are the defenders that plays in the centre of the defence (CB). They usually are taller and stronger than the wing-backs, or in other words more physical.

In the lower-left subplot (fig. 10 c), I have scattered the second component 'Defensive/Offensive' against the fourth 'Forward/Midfielder'. In this case we can see a good separation between all the classes. In fact, defenders are positioned on the upper-left part of the chart ('Defensive' and 'Midfielder'). Midfielders are more concentrated on the upper-right ('Offensive' and 'Midfielders'), while the forwards are on the lower-right ('Offensive' and 'Striker'). Goalkeepers are around the (0,0) coordinate as expected, since these are all attributes that don't concern this category.

In the last figure (fig. 10 d), I have plotted the fifth component ('Physical/Technical') against the sixth ('Jumping'). In this case I wanted to display, as stated before, that starting from the fifth component the separation is blurrier. While along the x axis we can see a sort of distribution (from the left to the right we see progressively FW-DF-MF) along the y axis there is no separation at all.

4 Computation of the Mean Relative Error and saving the results

The last part of the homework is quite similar to the first one. The most interesting part is the computation of the Mean Relative Error of the approximated dataset with respect to the original one. To do this, I have used a lambda function combined with a list comprehension:

```
MRE = lambda Xreal, Xpred: np.sum([la.norm(Xreal[:,j] -
↪ Xpred[:,j])/la.norm(Xreal[:,j]) for j in
↪ range(Xreal.shape[1])])/Xreal.shape[0]
```

The results are exactly the same both for my implementation and for the implementation using scikit-learn PCA. In this part I have also tried to implement

the version with the standard deviation to compare the results. The relative mean error in this latter case is a bit smaller. However, in this case 8 principal components are needed

```
Dataset approximated with the first 7 principal components. Mean Relative Error = 0.0003494999988202315
Dataset approximated with sklearn implementation (0.9 var explained/7 components). Mean Relative Error = 0.00034949999882023157
Dataset approximated with use of std (0.902256282158481 var explained/8 components). Mean Relative Error = 0.00034278559586929403
```