# Sport Analytics

Visual Analytics - University of Rome "La Sapienza"

Giovanbattista Abbate 1875271 • Fabio Di Spazio 1876540

February $17^{th}$, 2020

# 1 Introduction

Sports Analytics is a dashboard allowing the user to directly access data about the strongest football players over the world, visualizing the overall score and their gameplay skill values according to FIFA dataset. Our application is aimed at users who want to build their own "dream team", at "FantaEuropeo" players, at bookmakers or to any football lover. Sport Analytics helps the users by gathering all these useful informations into one single place resulting in time saved by the user.

# 2 Description

The app is composed by six main part organised as pictured in Figure 1.

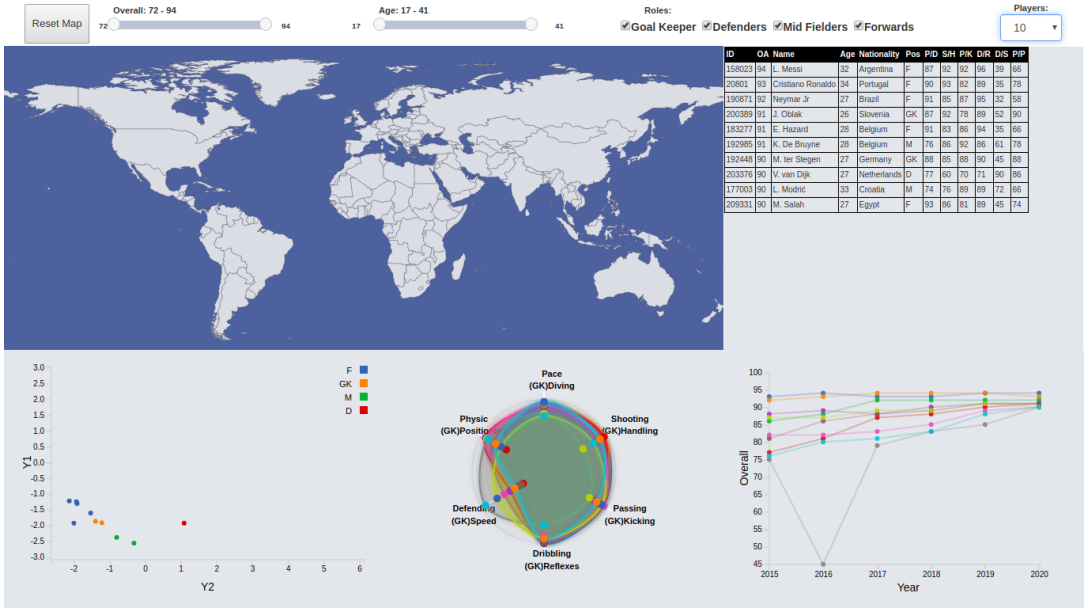


Figure 1: Full view of the page.

The elements are:

- **Header:** On the top of the page, it contains some basic filters.
- **Map:** The map filters the players according to their nationality.
- **Dataset Table:** The table contains all the values that matches the eventually setted filters (if any).

- **PCA Scatter Plot:** The scatter plot is the 2D graphical representation of the PCA.
- **Radar Chart:** The radar chart display players performance.
- **Line Chart:** The line chart shows players' overall over the years.

# 3 Implementation

## 3.1 Dataset and Preprocessing

The dataset used in this project is publicly available on Kaggle, at: https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset#players_20.csv. From the whole dataset, we used the files `players_15.csv`, `players_16.csv`, `players_17.csv`, `players_18.csv`, `players_19.csv`, `players_20.csv`. From the last file, during preprocessing we took the first 4000 players (out of 18000+), restricting attributes from 104 to 'sofifa_id', 'short_name', 'age', 'nationality', 'overall', 'player_positions', 'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic'. Moreover, since the last 6 attributes refer to outfield players, goalkeepers attributes were different from these so, we translated the attributes 'gk_diving', 'gk_handling', 'gk_kicking', 'gk_reflexes', 'gk_speed', 'gk_positioning' in order to fill the same 6 values in the data frame. About roles, we collapsed all the roles in 4 macro-roles: Defenders (D), Mid-fielders (M), Forwards (F) and Goalkeepers (GK). From the other files of the dataset we took the overall score of the previously mentioned 4000 players for each year, using them to plot the progress of these players from FIFA 15 to FIFA 20. Furthermore, we computed the PCA of the 6 main attributes of each player, performed some adjustment to the data and appended for each player an url containing his image.
The whole preprocessing is available at `/py/preprocessing.py`. We used `numpy` and `pandas` in order to manage the datasets, along with `sklearn` decomposition with whitening to compute PCA:

```
pca = PCA(n_components=6, whiten=True)
pca_result = pca.fit_transform(data.values)
```

The difference between original, normalized and whitened data is reported in Figure 2. The final dataset is stored in a JSON file, named `dataVeryFull.json`, where a total of 24 attributes for player are stored. The following parts are updated every time there is an update of the filtered data through the header or the map, and they are adaptive to the dimension of the window through refreshing the page.

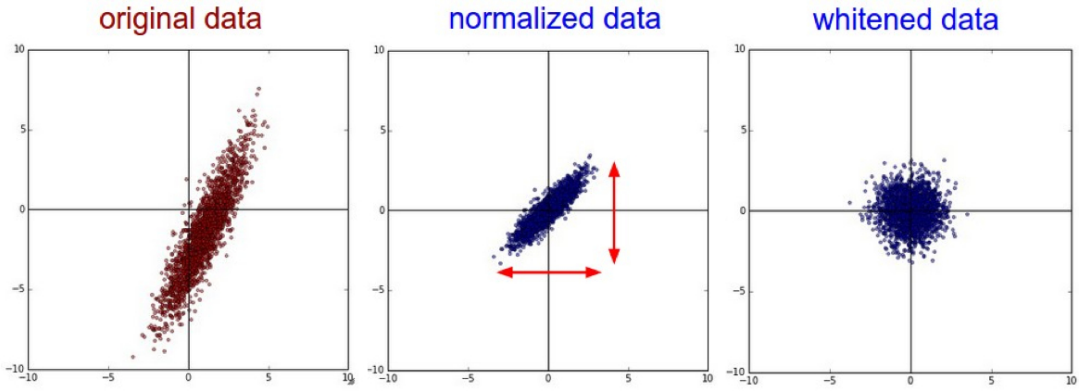


Figure 2: Difference between normalization and whitening.

## 3.2 Header

In the header there some items that through `d3.filter` function are used to restrict the data:

- **Button**: the "Reset Map" button. The interaction of this button is done through the `jQuery` library, in the `map.js` file. It resets the view of the map to the full planisphere, along with resetting values for all the visualizations in the page.
- **2 Sliders**: one for the overall, one for the age. The values of these sliders are managed through `jQuery` in the `main.js` file, and the data are filtered using them through `d3` library.
- **4 Check boxes**: these check boxes are useful to filter players by the four macro-roles defined in Section 3.1.
- **Drop-down Menu**: this menu allows the user to choose the number of players to show on all the visualization between 10, 100, 250, 500, 1000 and 2000. This is managed through `jQuery` after all the filters:

```
maxSlice = $("#var-select").val();
filteredData = filteredData.slice(0, maxSlice)
```

## 3.3 Map

The map used is taken from the Natural Earth site that allow to download the data in GeoJSON format (the maps are in the public domain). The GeoJson is a standard for representing geographic data using the JSON format and D3 uses GeoJSON to represent geographic features in JavaScript.

In order to use our map, the first thing to do is to define a map projection. Projections transform spherical polygonal geometry to planar polygonal geometry. D3 provides implementations of several classes of standard projections:

- Azimuthal
- Composite
- Conic
- Cylindrical

The projection that we chose is *Equi-Rectangular* that belongs to the *Cylindrical* class. Through `d3-geo` we can implement the projection:

```
var projection = d3.geoEquirectangular()
   .center([0, 15])
   .scale([w/(2*Math.PI)])
   .translate([w/2,h/2]);
```

Then, we need the geographic path generator, `d3.geoPath` generator which takes a defined projection and creates a SVG path data string.

```
var path = d3.geoPath()
             .projection(projection);
```

In order to draw the map, we need to create the SVG using D3 with:

```
var svg = d3
        .select("#map-holder")
        .append("svg")
        .attr("width", $("#map-holder").width())
        .attr("height", $("#map-holder").height())
        .call(zoom);
```

Now we need to load in our map data through `d3.json` and draw the map into the SVG. We put all the countries (map vectors) in the group `countriesGroup` and then for each country we need to draw a path. Within the JSON each country is defined as a *feature* so we bind `json.features` to `countriesGroup` and create a path for each feature. The shape of each country is returned by the `d` string. We also implemented two types of zoom: the D3 standard pan and zoom functionality and the function `boxZoom`. This function zooms into a clicked country. This is done by passing to the function the bounding box and the centroid of the country as well as a percentage of padding between the edges of the country and the displayed map. Moreover the `initiateZoom` function prevent the user to go beyond zooms limits, making the map too much small/big.

## 3.4 Dataset Table

Defined in `table.js`. Hovering with the mouse on the table's header, it is possible to see a legend of the abbreviations, while hovering on the players' name will show an image of the corresponding player. Each column of the table represents a different attribute of the players, while each row represents a player.

## 3.5 Scatter Plot

The scatter plot is implemented in `scatterPlot.js`. It represents the PCA computed above where are displayed the players that match the filters. Each player is represented by a dot and each role player has a different color (see the legend). The plot shows how the PCA has clustered players by placing similar player (depending on their skills) in neighbour places. Moreover the plot is responsiveness, in that each dot is clickable and the K-NN will be computed taking the neighbour dots; this is a nice feature, because in this way you can choose your best player for a certain role.

## 3.6 Radar Chart

The chart, coded in `radarChart.js`, has 6 axis representing the 6 main attributes of each player: Pace, Shooting, Passing, Dribbling, Defending and Physic for outfield players; Diving, Handling, Kicking, Reflexes, Speed and Positioning for the goalkeepers. Each blob represents a different player, and hovering on it returns a tooltip with the name of the corresponding player. This chart is made through `d3` library, using a scalable svg as basis and appending a circle for each player showed in the table. These circles glow during mouse hovering, and the intersection points between them and the axis represent players' values for that attribute.

## 3.7 Overalls Plot

Defined in `lineChart.js`, this plot follows the same principles of the previous one, defining through `d3` an svg and appending 6 values, coming from an array initialized on the dataset, defining the overalls of a specific player during the years 2015-2020. These points are then linearly interpolated two-by-two, and hovering over one of these lines, the name of the corresponding player is prompted and the other one are semi-hidden. Hovering over the points, it is possible to see the specific overall for a certain player in one of the 6 years.

### 3.8 Main Function

The `main.js` file contains the core of the project. It is mainly composed by a data filter, which considers all the components in the header and the Country selected by the map. It also contains all the variables not yet mentioned obtained through `jQuery` library, along with graphic configuration for the Radar Chart. It contains all the function to draw the objects in the page:

- **drawmap()** calls the file `map.js` and draws the first view of the map.
- **updateData()** is the core of the project. It removes previously computed K-NN on the PCA scatter plot, reads the dataset JSON file through `d3` library, and filters it according to the elements defined in Section 3.2 and to the country selected on the map. After this, the table, the radar chart, the overalls plot and the PCA scatter plot are updated through the functions:
  - **drawTable** which calls the function defined in `table.js`.
  - **RadarChart**, defined in `radarChart.js`.
  - **lineChart**, defined in the homonym js file.
  - **drawScatter**, defined in `scatterPlot.js`.

  All of these functions work basically in the same way, deleting the existing views and making new ones from the new `filteredData`.

## References

[1] Mike Bostock. Let's make a map. https://bost.ocks.org/mike/map/.

[2] Multiple Authors. Tutorials - d3/d3 wiki. https://github.com/d3/d3/wiki/Tutorials.

[3] jQuery Foundation. jquery. https://jquery.com/.

[4] Bootstrap Team. Bootstrap — the most popular html, css and js library in the world. https://getbootstrap.com.

[5] Nadieh Bremer. Radar chart redesign. http://bl.ocks.org/nbremer/21746a9668ffdf6d8242.

[6] soFIFA.com. Fifa 20 players. https://sofifa.com/.