Operational guide

The Italian Women's National Volleyball Team: A Legacy of Excellence

Aurora Castelnovo, Nicole Gemelli, Daria Marinucci

As the title suggests, the subject of our project is the Italian women's national volleyball team. We aimed to analyse the selection process of athletes for the national team, considering tournaments from 2021 up to the last Olympic Games. After this analysis, we also attempted to predict the possible roster for the next World Championship, scheduled for September in Bangkok. In order to obtain this we have followed the data science pipeline:

- Data acquisition: Python
- Data storage: MongoDB
- Data profiling: Python
- Data preparation: Python, MongoDB, Knime
- Explorative data analysis: Python
- Data modelling: Knime
- Final conclusions

To have a clear understanding of the order to follow, first execute the codes on Python until the data exploration. After that move to MongoDB and execute all the query. In order to have a clear idea of the data, execute the exploratory data analyses on Python. Lastly see Knime workflow to discover the data modelling and final conclusions.

Data acquisition

At the beginning of this step, we considered what data was necessary and where to find it. The official website of the Italian league appeared to be the most reliable source for obtaining player statistics and a list of those with Italian nationality. This list is crucial as it represents the first selection criterion for determining eligible players. We also needed to know the list of the player of last tournaments, in particular:

- European 2021
- World 2022
- European 2023
- Olympics 2024

This part was the most difficult because it wasn't easy to find a page able to provide all teams, at the end we decided to use Wikipedia.

Data acquisition was performed using web scraping techniques. We used Python, specifically the BeautifulSoup library, to extract the necessary information. Given the nature of web scraping, we had to handle inconsistencies in data formatting and ensure the accuracy of extracted information. In some cases, manual verification was necessary to confirm player names and tournament rosters, reducing the risk of errors in subsequent analyses.

In order to obtain the statistics we wrote a function able to scrape different web pages and add the role to each one, we repeat this operation from season 21_22 till 24_24. We keep 5 different dataset for each role, because at the beginning we weren't sure if we need them separate or combined.

Moving to the Italian players, also here we wrote a function able to scrape pages containing the players from season 21_22 till 24_25. The function scrape the first URL and save all the needed data (Athlete,Role, Nationality, Height, Birth Year). The function moves to the second link, save the data and drop the duplicates, this is useful to have unique players. The cycle iterate all the URL and outputs the final dataset.

Lastly, the tournament. Here, we wrote a function able to scrape all the page related to the tournaments we were interested in and saved the team in datasets removing the trainer. As we wanted a unique one, we created a binary column for each tournament, where 1 highlights the participation and 0 the non-participation. After combining this dataset with the player information the final dataset contain: Player, Role, Year of birth and all the tournaments.

Data storage

This steps helps to store the data in a database, to have them. We decided to use MongoDB, with a referential method, we needed al collection together but we aimed to have the possibility to interact among them.

As we have performed scraping on Python we needed to connect it with Mongo, therefore we used the library pymongo to do it. Before we needed to do some transformation that will be better explained in the data integration part.

We have connected Python with our local server calling the collection Volleyball

We preferred to write the query directly on Mongo in the section Mongo DB Shell, a graphic interface that allow the interaction.

The following queries where performed.

First we need to transform the NaN into white spaces, we choose not to substitute anything because there was the risk to distort the analyses and as the missing values were greater than the 5% of the whole dataset remove them would be an error.

To have a complete view, we have decided to insert all the query in this section, even if they belong to other steps. All of them were performed on Mongo DB, this is the reason why they are here.

Tranformation of Nan values into " "

use Volleyball; *#selects the database*

const collections = db.getCollectionNames(); *#gets all the collections*

#for each collection we ask to iterate each document, checks if any value is equal to "NaN" and eventually transform it into " "

```
collections.forEach((collectionName) => {
   print(`working with collection n. : ${collectionName}`);
```

```
        db[collectionName].find().forEach((doc) => {
        let updates = {};
        Object.keys(doc).forEach((key) => {
            if (typeof doc[key] === "number" && doc[key] !== doc[key]) {
                updates[key] = "";
            }
        });
#if there are updates execute them
        if (Object.keys(updates).length > 0) {
            db[collectionName].updateOne(
                { _id: doc._id },
                { $set: updates }
            );
            print(`Updated  id: ${doc._id}`);
        }
    });
});
```

To see the last updates

```
db.local.find().pretty();
```

We wanted to change the column Height in Height_in_cm

```
db.athletes.updateMany(
  {},
  { $rename: { "Height": "Height_in_cm" } }
)
```

Merge statistics and Italian players

```
db.stat_set_2425.aggregate([ #starting from the setter dataset
    {
        $unionWith: {
            coll: "stat_opp_2425" #combine with the opposite hitter statistics
        }
    },
    {
        $unionWith: {
            coll: "stat_mb_2425" #union with the middle blocker
        }
    },
    {
        $unionWith: {
            coll: "stat_lib_2425" #union with the libero
        }
    },
    {
        $unionWith: {
            coll: "stat_spi_2425" #union with the spiker
        }
    },
#consider the new collection
    {
        $lookup: {
            from: "athletes", #collection to consider for join
            localField: "Athlete ", #attribute from the first collection
            foreignField: "Athlete", #attribute of the athlete collection
            as: "Athlete2425" #final document
        }
    },
    {
        $match: {
            "Athlete2425": { $ne: [] } #consider only the matching documents
        }
    },
#unwind in order to transform the array in a single document
    {
        $unwind: "$Athlete2425"
    },
    {
        $out: "stats2425" #output file
    }
]);
```

Data Profiling

Data profiling is a crucial preliminary step that helps us gain a deeper understanding of the dataset. We used Python to perform this analysis on three datasets: Italian players' personal information, the tournament team dataset, and the statistics dataset. Since all statistical datasets share the same structure, we focused only on the one related to the current season.

For each dataset, we examined the attribute types using **.dtypes** and identified missing values with **.isnull().sum().** Among them, the statistics dataset had the highest number of missing values. To assess the extent of this issue, we calculated the percentage of missing values and identified the five columns with the most missing data. Given that this dataset also contains the largest number of numeric columns, we used **.describe()** to gain insights into key statistical measures.

In the personal information dataset, we determined the oldest and youngest players. Additionally, we analysed the distribution of athletes across different roles, visualizing it with a bar plot, thanks to the help of the library **Seaborn**.

For the tournament dataset, our primary interest was understanding the distribution of convocated athletes per tournament, as this insight will be valuable for future predictions.

Data Preparation

This step contains data cleaning, integration and enrichment. It was performed principally on Python, but something was done also on Mongo and Knime. We saw immediately that name and surname were different for each dataset, so we need to choose one format and transform in it.

Data cleaning

Statistics data set had the column athlete composed by surname and name, we decided to use it as baseline. We needed to add the column role to each dataset as data were stored in different pages. As we were aware of the problems related with having "," instead of "." For separate decimal part we have decided to sunstitute them, with a replace function:

df.replace(',', '.', regex=True)

Moving to the list of the Italian players those information were saved in two different columns we used the following function to correct it:

new_df["Athlete"] = new_df["Surname"].fillna("") + " " + new_df["Name"].fillna("").

We have combine the surname and the name of each player to have a complete column. After that, we needed to translate the role from Italian to English. Wehave written the transformation in a dictionary "role_translation" and applied the function:

new_df["Role"] = new_df["Role"].replace(role_translation) #replacement of the italian role

Another change was done in the column "Height", where we have removed the string "cm" in order to have a numerical attribute.

```
updated_df['Height']=updated_df['Height'].str.replace(' cm', '')
```

To have a better understanding on MongoDB, we have decided to change the name of the column Height in Height _in_cm [query **change the column Height in Height_in_cm**].

Finally, in the tournament team we saw that name and surname had different format with respect to our base line, we need to change something. We used the following function in order to solve that problem:

```
def invert_name(name):
    name = re.sub(r"\[\d+\]", "", name).strip()  # Remove references like [1], [2]
    parts = name.split()
    if len(parts) > 2:
        return f"{' '.join(parts[1:])} {parts[0]}"
    elif len(parts) == 2:
        return f"{parts[1]} {parts[0]}"
    else:
        return name
```

The roles were saved with the initial letter, for example C is Centrale, which means middle blocker. We wrote a function to translate them.

```
def translate_role(role):
    translations = {
        "S": "Spiker",
        "C": "Middle Blocker",
        "P": "Setter",
        "L": "Libero",
        "O": "Opposite"
    }
    return translations.get(role, role)
```

In the original data, there was a column “date” with the full birth data. In the other datasets we have just the year this is the reason why we have decided to extract it.

```
def extract_year(date):
    return date.split()[-1]
```

On mongoDB we have replaced all the Nan values with white spaces. [Query **Tranformation of Nan values into “ “**]

We substitute the Nan values with white spaces on MongoDB using the node “Missing value” .

Data integration and enrichment

Firstly, we have created the “total_ds” the one used to perform machine learning techniques for prediction. On Python we have combined the role datasets for each season adding the column “Year”, with the objective to differentiate them. We have combined all the season together and performed an inner join with the Italian players list, so that we were able to know the statistics and the personal information of the players that could be convocated in the national team.

On MongoDB, we have combined together all the role for season 24_25 in a single dataset and joined it with the Italian volleyball player list, this was helpful to understand among who we can choose. [query **Merge statistics and Italian players**]

On Knime, we have joined the list of the players with the national volleyball team, adding the binary column In list, where 1 means that the player was at least in one team and 0 that she was in none of the teams.

Explorative data analysis

Exploratory Data Analysis (EDA) was conducted utilizing both Knime (to a lesser extent) and Python.

Within the Knime platform, the Statistics View node facilitated a comprehensive overview of key statistical metrics for each variable within the 'total_ds' and 'team_turn' datasets. Furthermore, the Linear Correlation node was employed to quantify correlations between variables across these two datasets.

The primary EDA activities were performed in Python, where specific relationships between variables of interest to our objective were examined and analyzed.

Data modelling

This phase was executed entirely within the Knime platform. Following the upload and processing of the 'total_ds' and 'team_turn' datasets, and after conducting a general Exploratory Data Analysis (EDA) on both, the data modeling phase was initiated.

Initially, the 'total_ds' dataset was normalized using the “Normalizer” node to ensure that all variables had the same range of values. Subsequently, it was partitioned into five subsets, each containing observations pertaining to a unique role. The "SelfOrganizingMap" node, implementing Kohonen's Self Organizing Map algorithm for unsupervised clustering, was applied to each subset. Subsequently, the “Weka Cluster Assigner” node was utilized to assign the imported data to the corresponding clusters based on the cluster model generated previously.

For each subset, only the most relevant variables for each role were employed in the clustering model.

Upon identifying the clusters exhibiting the best performance for each role, the focus was narrowed to the 2022-23 and 2023-24 seasons to pinpoint the players associated with these top performances. Subsequently, the required number of players per role was selected (e.g., two players were selected for the libero role). In instances of performance ties, the player exhibiting the strongest performance in the current 2024-25 season was chosen.

To validate the selected players, a check was conducted to ensure their participation in the current season, thereby excluding players who are no longer active.

Finally, a comparison was made between the selected players and those who participated in recent European Championships, World Championships, Olympics, and the last two VNLs. This comparison provides insights into the efficacy of our selection process and identifies which and how many players could be classified as "emerging talents."