MDSAA

Master Degree Program in
Data Science and Advanced Analytics

Machine Learning in Sports Industry

Discover promising athletes for the future of Handball

Miguel Coutinho Nunes

Project Work

presented as partial requirement for obtaining the Master Degree Program in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

MACHINE LEARNING IN SPORTS INDUSTRY: DISCOVER PROMISING ATHLETES FOR THE FUTURE OF HANDBALL

by

Miguel Coutinho Nunes

Project Work presented as partial requirement for obtaining the Master's degree in Advanced Analytics, with a Specialization in Business Analytics

Supervisor: Vítor Manuel Pereira Duarte dos Santos, PhD

Advisor: Ricardo Andorinho

November 2022

DEDICATION

Dedico inteiramente esta tese à minha família e namorada que sempre me apoiaram e contribuíram para que conseguisse atingir esta etapa de minha vida, sendo uma das mais importantes. Obrigado a todos!

AKNOWLEDGMENT

Gostaria de começar por agradecer o apoio e colaboração do Prof. Dr. Vítor Duarte dos Santos na orientação e disponibilidade prestada ao longo do desenvolvimento da tese. A sua influência e encorajamento foram essenciais para que este projeto fosse finalizado.

Quero também agradecer à Federação Portuguesa de Andebol e seus representantes, Ricardo Andorinho e João Sousa, por providenciarem os dados, esclarecerem dúvidas e contribuírem com ideias para que conseguisse avançar com o estudo.

Não poderia deixar de mencionar o agradecimento pela ajuda do meu amigo Rui Sousa na criação do programa em VBA para pré-preparação dos dados e não menos importante, um reconhecimento especial pelo apoio da minha namorada, Madalena Cabaço, ao incentivar-me nos momentos finais de entrega.

ABSTRACT

Nowadays, technology applied in industries has increased day by day. As a result, different ways of collecting and processing data have been investigated. Machine Learning is one of those ways where, in addition to its applicability in all sectors, it has been increasingly explored in different sports. Furthermore, the analysis of data at a visual level helps in the interpretation and understanding of them.

These types of procedures always seek to support in the decision-making work done by coaches, managers and scouting. It can be inherent to any sport and handball is obviously included.

This specific investigation addresses methods to create advantages for Handball, introducing predictive analytics. The discovery of promising athletes based on collected variables is one of the biggest challenges in this sport and although the data provided by the Federação de Andebol de Portugal are limited, this study demonstrates a 'direction' of how it can be done based on a single variable.

In addition to working in the collection and pre-preparation of sports data, examples of visual presentations such as vertical/horizontal bar graphs and maps are exposed. Finally, Machine Learning algorithms with and without default parameters are used to predict if the player is promising. From this perspective, it can be concluded that, based on formation years, models score is slightly better for Support Vector Machines algorithms despite the proximity of the results. It is important to point out that relevant conclusions were also drawn from the graphs.

KEYWORDS

Machine Learning; Promising forecasting; Predictive Analysis; Sports industry; Performance metrics; Handball

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
CRIPS-DM	Cross Industry Standard Process for Data Mining
CSV	Comma-Separated Values
FAP	Federação de Andebol de Portugal
ID	Identification Number
IT	Information Technology
JSON	JavaScripct Object Notation
KNN	K-Nearest Neighbors
ML	Machine Learning
N/A	Not Applicable
SQL	Structured Query Language
SVM	Support Vector Machines
VBA	Visual Basic for Applications

1. INTRODUCTION

1.1. Context

Artificial Intelligence (AI) and Machine Learning (ML) have become increasingly incident in various fields and contexts (*Machine Learning: What It Is and Why It Matters* | *SAS*, n.d.). Artificial Intelligence technology has been widely used in the last decade in almost every field of science and our daily life in areas ranging from education to healthcare and space research. Their corresponding technologies have gained importance as they provide effective and robust solutions to problems with high accuracy and few errors (Cristina et al., 2020). As a field of AI, Machine Learning is also applied within these fields, and its reliability has been proven. ML uses previous experiences to establish a link to the future, and the success of the model is highly related to the characteristics of the dataset. In recent years, several experiments using different ML models have been conducted in sports (Oytun et al., 2020). In the current world, sports produce considerable statistical information about each player, team, games, and seasons. Traditional sports science believed science to be owned by experts, coaches, team managers, and analysers. However, sports organizations have recently realized the abundant science available in their data and sought to take advantage of that science using ML techniques. Sports ML assists coaches and managers in result prediction, player performance assessment, player injury prediction, game strategy evaluation and promising athletes identification (Langaroudi & Yamaghani, 2019).

Every country wants to get fame at the global level in different sports. To achieve fame, countries are investing in sports and games to enhance the performance of their teams and players. The players have progressed from time to time and there has been an increased popularity for every sport. The performance of athletes in their respective sport has also improved in recent times. Whatever may be the sport, the competition nowadays is much high. To discover real talent among all players, it's crucial to store, manage and update time-to-time data to have a record of their numbers. This is where Machine Learning comes into play (Sri Harsha Vardhan Goud et al., 2019).

In handball, during the years practicing the sport, the improvement of athletes is constantly occurring until reaches the highest level and become a promising player (Biscaia et al., 2018). In each stage, coaches should consider specific goals based on players' characteristics such as previous experiences, levels of maturation, growth, and development (Menezes et al., 2018).

Importantly, handball is known as a sport that requires strength, coordination, power, and a discontinuous pace, with intermittent game characteristics that involve fast defence and offense. In addition, the game is performance-oriented and contains technical, tactical, and psychological elements (Oytun et al., 2020). Therefore is required years to learn, develop and specialize to become differentiate from others.

In sum, sports statistics offer a reasonable and recognizable approach to search for ways to discover promising players and how they are developed. Nevertheless, the problem lies in deciding which statistics should be used. There are popular measures such as goals or assists, but they fail to capture some critical indicators. Therefore, one of the most important questions for today's coaches is to know both the variables that influence player performance and the relative contribution of these variables to become promising (Romero et al., 2020).

1.2. Motivation

There is a need to take advantage of how promising athletes can be discovered in high competition sports. Within the national sports, handball represents one of the sports where these concepts are becoming more used (Biscaia et al., 2018) and, therefore, this may be a starting point in the development of models to promote good Machine Learning practices. With the application of ML algorithms, we can give coaches, technical staff and scouting people the possibility to analyze the principal factors that identifies promising players.

One of the main tasks is to define a variable that are sufficiently supported to be applied in the algorithms. There are some features that can be found in the database of Federação Portuguesa de Andebol (FAP).

However, it is important to note that identifying variables and building models in a sports environment is a challenging task. Indicators to identifying promising athletes are affected by many factors in different sports, and it is not easy to estimate which are most important and decisive. In this specific study, database provided by FAP contains some variables that can be crucial for the discovery.

In conclusion, despite the challenge, it would be interesting to draw conclusions about FAP players database and build predictive algorithms with the ability to identify promising handball athletes.

1.3. Objectives

The research goal is to build a comprehensive model for the use of machine learning techniques on the discovery of promising handball athletes.

In order to achieve this goal, the following intermediate objectives were defined:

- Perform a study about handball, namely about the athlete's main characteristics and performance indicators that can be analysed in this database;
- Understand how to pre-preparate and analyse sports data graphically;
- Apply Machine Learning algorithm techniques to predict promising handball players;
- Evaluate analysed data and performance of tested algorithms;
- Define conclusions regarding data and predictive analysis.

1.4. Study importance and relevance

The discovery of promising players based on several different sport factors is something that all coaches and scouting panels wanted to understand in order to select the best athletes for the team. Applying this need while exploiting the application of advanced statistics could be quite complex. That is why this study is relevant for the handball sport since variables will be detected with the goal of analysing a particular athlete and detecting whether he or she is someone promising or not through algorithmic models.

Another important aspect refers to the fact that this research may become an open door to apply Machine Learning techniques in the sports sector and national handball since is becoming more explored in this industry (Schwenkreis, 2020).

2. LITERATURE REVIEW

2.1. Handball

2.1.1. Game Description & Rules

Today team handball is played indoors on a field of 20 by 40 meters. It is played by two teams consisting of 7 players out of which one player is the (optional) goalkeeper. There is a goal on each side of the field and a penalty area in front of the goal. Only the goalkeeper of the defending team is allowed inside the penalty area (*England Handball Association Rules and Regulations 2018.19*, 2018; “IX. Rules of the Game a) Indoor Handball,” 2022; *Official USHA Handball Rulebook with Interpretations, One-Wall and Three-Wall Addendums*, 2015).

After a throw-off by one of the teams, the team possessing the ball is trying to score a goal which is called an attack. After 60 minutes match time, the team that has scored most goals wins the match. An attack might end with the following outcomes, which means that the attack ends, and the ball possession changes:

- The team throws the ball at the goal and scores a goal.
- The team throws the ball and misses the goal, or the goalkeeper saves the ball.
- During the attack, the attacking team loses the ball due to a ball handling error of one of the players.
- During the attack, the attacking team loses the ball due to a violation of the rules of the game (“IX. Rules of the Game a) Indoor Handball,” 2022).

The game dynamics in handball arise from a series of phases that are organised and based on ball possession, and performance indicators may be able to reflect the internal structure of this possession in different game sequences. It is important to note that in handball, setting up a goal is the result of direct and indirect actions, as well as the use of free spaces and passes, but note that different objectives must be met during different game phases (“IX. Rules of the Game a) Indoor Handball,” 2022). These game phases: a) attack, characterised by a team attempting to score a goal; b) defense, which aims to recover possession and prevent the opposing team from scoring a goal; and c) goalkeeper, where a player tries to prevent a goal within the goal area. These dimensions are combined to achieve match success and take place in different areas of the field (Debanne & Volossovitch, 2022; "IX. Rules of the Game a) Indoor Handball," 2022; Póvoas et al., 2012). Given the specific features of handball, match indicators should include defense, attack, and the role of the goalkeeper (Skarbalius et al., 2013).

2.1.2. Performance indicators

Focusing now on performance indicators of players, although handball is known as a team sport, we must consider that the team's performance also comes from the individual performance of athletes (Ibrahim Akl & Hassan, 2017; Romero & Angulo, 2020). As in all other sports, there is a specific set of predominant characteristics that are essential to the practice of handball and to achieve success. Some of the most important performance indicators are coordination and agility, strength and power, constitution and social factors (Wagner et al., 2014).

Besides these performance indicators, the number of years practicing the sport (formation years) is crucial to identify possible promising players. Considering players experiences, their development occurs in different stages, from their initiation in handball (around 12 years old, in a diversified environment with the promotion of free and creative play) until reaching adult teams, with a high level of specialization (Biscaia et al., 2018; Menezes et al., 2018). There are studies mentioning that the sport-specific development age are under-12, under-14, under-16 and under-18 teams (Menezes et al., 2018). Others suggest that is required a period of eight years to prepare children and juniors in handball (Cezar et al., 2013) or between seven and ten years in order to increase the performance during the game (Biscaia et al., 2018).

Essentially, practicing handball in a properly oriented way, from an early age leads to the superior development of better understanding the game and tactical situations, at a given moment (Baker et al., 2003; Biscaia et al., 2018).

During this data science study development, a function based on the formation years of steps 'minis', 'sub-14', 'sub-16' and 'sub-18' will be built for target/dependent variable creation used on modelling analysis.

2.1.3. Challenges and Opportunities

The accomplishment of a team is measured by the victories and won trophies during the season and the position in the league (Debanne & Volossovitch, 2022; Schwenkreis, 2020). There are four areas that can be distinguished which contribute to the objective:

- The preparation of a season focusing on finding the best players and consequently build a team with the same mindset for the year;
- The preparation of matches during a season / competition, studying and analysing the opponents in detail by checking the individualities and tactics;
- Understand how team and players performance can be increased during the league;
- Constantly analyse the team and players performance after matches as a retrospective (Schwenkreis, 2020).

If the coaching team could convert these contributions into something actionable, the team performance will somehow increase and leading to victories.

2.2. Study tools and concepts

2.2.1. Docker

Docker is an open-source platform for developing, publishing and running applications. It is a tool that allows the segregation between the application and its infrastructure, making software delivery faster (Bui, 2015; *Docker Overview | Docker Documentation*, 2022; Rad et al., 2017).

The technology offered by this tool has provided the virtualization of applications using the concept of containers. As it is a relatively recent platform, it is constantly updated by the people who use it (Bui, 2015; *Docker Overview | Docker Documentation*, 2022). Docker Container is a way to "package" all the code and its dependencies of an application in the standard format, respecting the language in which it was developed, allowing its fast execution, whether on a local computer, in virtual machines or cloud service providers (*Docker Overview | Docker Documentation*, 2022; Rad et al., 2017). Also, the Container works in isolation, in terms of disk, memory, processing and network. This separation allows for more agility in the creation, execution and orchestration of containers, in addition to different environments being able to use the same host, without any problem (Bui, 2015; *Docker Overview | Docker Documentation*, 2022; Rad et al., 2017).

During the collection of data for study analysis, a docker container called 'mymongo' was created to restore all data provided by FAP into MongoDB Compass.

2.2.2. MongoDB

MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables (Chodorow, 2013).

A document-oriented database replaces the concept of a "row" with a more flexible model, the "document." By allowing embedded documents and arrays, the document- oriented approach makes it possible to represent complex hierarchical relationships with a single record. Every document has a special key, "_id", that is unique within a collection. Also, a collection can be thought of as a table with a dynamic schema and a single instance of MongoDB can host multiple independent databases, each of which contains its own collections (Banker et al., 2016; *O Que É O MongoDB? | MongoDB*, 2022). MongoDB documents or collections of documents are formatted as JSON (Java Script Object Notation), a popular scheme for storing arbitrary data structures (Banker et al., 2016).

Additionally, MongoDB contains some components as such MongoDB Compass. This component is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment. Being free to use and available for MacOS, Linux and Windows (*What Is MongoDB Compass? — MongoDB Compass*, 2022).

Focusing on the study of discovering promising handball players, all FPA information is stored in infra-structural MongoDB database, where the collection process started from there to excel file.

2.2.3. Array and Array of objects

Since data of FAP is stored in MongoDB with JSON format, is important to reference array and array of objects data type that are important for the study improvements. Array is a collection of data and a data structure that is stored in a sequence of memory locations. The array can store all the primitive data types (like integer, float, string, and boolean). An array is represented as elements enclosed within square brackets "[ ]," and the elements of an array are accessed using their index values which start from 0 for the first element and go up in number for every next element in the array (Eljinini, 2022; T.Bray, 2014).

On the other hand, array of objects stores multiple objects in the memory in sequential order. The objects are non-sequential memory locations initialized under one identifier that can store all types of values. Objects have properties defined by key-value pairs instead of elements. The complete object is wrapped with a pair of curly brackets "{ }" (Eljinini, 2022; Rauschmayer, 2022).

On this study, the columns 'registrations' and 'history' retrieved from MongoDB are characterized by array of objects. Column 'history' is storing information about all games played including 'gameNumber', 'homeTeam', 'league', 'goals', etc. Column 'registrations' is referencing data about the all-career years containing 'clubName', 'seasonDesc', 'associationDesc', 'stepDesc', etc. Aditionally, column 'roles' is considered an array containing all individuals responsabilities.

2.2.4. Microsoft Excel

The application Microsoft Excel is a software program created by Microsoft that uses spreadsheets to organize numbers and data with formulas and functions. Excel analysis is ubiquitous around the world and used by businesses of all sizes to perform all kind of analysis (*Microsoft Excel Spreadsheet Software | Microsoft 365*, 2022). Inclusively, is the most important software tools for new hires being consistent in different areas and across all experience levels (Lee et al., 2018).

Additionally, Excel can be enriched when is used different Excel features pivot tables, advanced filtering, and formatting capabilities as approaches to structure data and managerial decision-making (Jacobs et al., 2016). By organizing data using software like Excel, data analysts and other users can make information easier to view as information is added or changed. Excel contains a large number of boxes called cells that are ordered in rows and columns. Data is placed in these cells (*Microsoft Excel Spreadsheet Software | Microsoft 365*, 2022). Combining Microsoft Excel with VBA code can be very comfortable for developers (Cirujano & Zhu, 2013).

In the current study, this application is mainly used to store content, coming from MongoDB CSV files, applying VBA code program for data structure and pre-prepraration.

2.2.5. Visual Basic for Applications – VBA code

The Visual Basic for Applications (VBA) consists in an event driven programming language used for Microsoft Office applications, including Excel. It gives 100% accurate results and requires a very little time (Abidin et al., 2015). VBA is applied generally for algebraic calculations just like creating variables, interacting with different equations and creating results (Bernard et al., 2018). The great power is that it can customized applications and solutions to enhance the capabilities of those. For example, the automation of repetitive tasks is one of the most common uses of VBA (*Getting Started with VBA in Office | Microsoft Learn*, n.d.). Another advantage consist in the fact that less skilled employees can easily run VBA in excel and operate many tasks and applications getting accurate results since inexistent complexity while working with it (Chaudhry et al., 2021).

On data science scope, VBA code was used in order to recreate and execute the process for migration of data and its analysis being quite faster and flexible (Poznan & Poznan, 2019). Specifically, in this study VBA came as a solution to deconstruct the array of objects into a new table thinking on analyses needs.

2.2.6. Anaconda Navigator and Jupyter Notebook

Anaconda Navigator is a desktop graphical user interface (GUI) that allows you to launch applications and manage packages, environments, and channels. In Anaconda it's possible to search for packages and is available for Windows, macOS, and Linux. The great advantage of this desktop is the fact that we can work with applications, packages and use multiple environments without using any commands in a terminal window (*Anaconda Navigator — Anaconda Documentation*, 2022).
One of the available applications inside Anaconda Navigator desktop is called Jupyter Notebook. This notebook can be considered mostly as an interactive application console suitable for capturing the whole computation process such as developing, documenting, and executing code, as well as communicating the results (*The Jupyter Notebook — Jupyter Notebook 7.0.0a7 Documentation*, 2022). The Jupyter notebook combines two components:

- web application: creation of documents containing computations and explanatory text from an interactive tool from browser;
- notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, images, etc.

During this prediction study, Juptyer Notebook and, consequently, Anaconda Navigator were used to analyse generated excel tables applying Python language.

2.2.7. Python language

Python consists in a high-level, simple and powerful programming language with excellent functionality for process linguistic data. It is mainly used for in web development (server side), software development, mathematics, Machine Learning applications and system scripting (Bird et al., 2009).

Among all the different applications, Python have several advantages. Is known for being very user-friendly, forgiving of errors giving flexibility to the programmer. This language can support different styles of programming including structural and object-oriented providing the ability to use modular components that were designed in other programming languages (Srinath, 2017).

2.3. Predictive Analytics and Machine Learning

2.3.1. Predictive Analytics

Predictive analytics is the branch of the advanced analytics which is used to make predictions about unknown future events. It uses many techniques from historical data combined with statistical modeling, data mining techniques and machine learning. The patterns found in historical and transactional data can be used to identify risks and opportunities for future (Kelleher et al., 2015; Swamynathan, 2019). This concept is often associated with big data and data science. Predictive analytics models capture relationships among many factors to assess risk with a particular set of conditions to assign a score. To gain insights from this data, data scientists use deep learning and machine learning algorithms to find patterns, train the models and make predictions (Swamynathan, 2019).

2.3.2. Machine Learning

Machine learning consists in a collection of algorithms and techniques used to create computational systems that learn from data in order to make predictions and inferences, that is programming computers to optimize a performance using example data (Han et al., 2012; Kelleher et al., 2015; Ozdemir, 2016; Swamynathan, 2019). This data is used to apply statistical model/algorithms to capture relationships between various datasets and further predict the likelihood of an event. Machine Learning models are the main responsibles for this action (Swamynathan, 2019).

There are different ways to train machine learning algorithms, having their own advantages and disadvantages. For this action is important to understand that are two kinds of data:

labeled data and unlabeled data. Labeled data has both the input and output parameters in a completely machine-readable pattern but requires a lot of human labor to label the data. Unlabeled data only has one or none of the parameters in a machine-readable form. This negates the need for human labor but requires more complex solutions (Han et al., 2012; Ozdemir, 2016). Following this explanation, there are three machine learning subcategories (Ozdemir, 2016; Swamynathan, 2019):

- Supervised learning
- Unsupervised learning
- Reinforcement learning

This study will focus on the Supervised learning subcategory since is the only applied one.

2.3.2.1. Supervised learning

Basically, supervised learning is a subcategory of Machine Learning that finds associations between features of a dataset and a target variable (Kelleher et al., 2015). These associations allow supervised models to make predictions based on past examples. It is defined by its use of labeled datasets to train algorithms that classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately (Swamynathan, 2019). Specifically, supervised learning works using parts of the data to predict another part. First, we must separate data into two parts, as follows:

- The independent variables, which are the columns that will be used to make our prediction;
- The dependent variables, which is the column that we wish to predict. This is sometimes called target.

Supervised learning attempts to find a relationship between the independent and dependent variables in order to make a prediction (Ozdemir, 2016).
For example, in this specific master thesis the supervised learning models will try to find the association between the number of years played on each step (minis, sub-14, sub-16 and sub-18) and that player's classification being promising, inconclusive or not promising (the target variable).

Following this concept, there are two types of commonly used supervised learning algorithms: Regression and Classification.

2.3.2.2. Regression

Regression models attempt to predict a continuous number in relevance with a given input dataset. This means that the response can take on a range of infinite values (Han et al., 2012). Regression analysis is a statistical methodology that is most often used for numeric prediction, although other methods exist as well. Regression also encompasses the identification of distribution trends based on the available data (Ozdemir, 2016; Swamynathan, 2019).

2.3.2.3. Classification

Classification attempts to predict the actual or the probability of class labels and the number of class labels can be two or more. That is, it recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined (Ozdemir, 2016; Swamynathan, 2019). To achieve it, the model are derived based on the analysis of a set of training data (i.e., data objects for which the class labels are known) (Han et al., 2012).
For this project, classification algorithm is applied since we are dealing with classification of Handball athletes. However, is important to note that these classes are more than two ('promising', 'inconclusive' and 'not promising'). Consequently, multi-class classification task is the used one for predictive analysis.

2.3.2.4. Multi-class classification

As mentioned above, multi-class classification contains more than two classes in our target/dependent variable making the assumption that each sample is assigned to one and only one label i.e., a player can be either promising or inconclusive but not both at the same time (Aceto et al., 2018; Tsoumakas & Katakis, 2009).

2.3.2.5. Machine Learning algorithms

Support Vector Machines

From several Machine Learning classification algorithms, Support Vector Machines (SVMs) are a powerful supervised learning method used in data analysis and pattern recognition, which also have been widely adopted for prediction in a variety of sport domains (Soto Valero, 2016). This algorithm can either be used for classification or regression study, being widely used for classification (Aoudi & Barbar, 2016).

The method performs a nonlinear mapping in order to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal hyperplane which separates the instances of one class from another, that is, will find the hyper plane by employing vectors (training dataset) and margins (defined by vectors) (Langaroudi & Yamaghani, 2019; Loss et al., 2022).

A SVM model represents the data as points in space, focusing on dividing the different classes by a clear gap that is as wide as possible. It takes more time among other algorithms, however is believed to have high accuracy (Langaroudi & Yamaghani, 2019).

K-Nearest Neighbors

The k-Nearest Neighbors algorithm (k-NN) is one of the most popular instance-based learning algorithms, being non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point (Claudino et al., 2019). Among all machine learning algorithms K-NN benefits from simplicity, easily programmable, comprehensible, robustness for search space and usually gives high prediction accuracy (Patel & Thakur, 2019). However, is computationally very expensive and requires lot of memory (Horvat et al., 2018).

This algorithm currently is used for sports analysis for prediction studies. The process by doing a comparison of an instance with the similar training data. All training instances are stored in an n-dimensional pattern space representing points (Soto Valero, 2016). While the algorithm is doing the prediction of an instance, searches the pattern space for the k training instances that are closest to the unknown instance. These k instances are the parameter ‘n_neighbors’ (nearest neighbors) of the unknown instance. This is the most fundamental parameter of KNN model since it’s regulating how many neighbours should be checked when an item is being classified (*Sklearn.Neighbors.KNeighborsClassifier — Scikit-Learn 1.1.3 Documentation*, 2022).

2.3.2.6. Training and Validation data

Typically, when you’re building a model, you split your labeled dataset into training and validation sets (Munro & Mirsharif, n.d.). The data from the training set are used to train and fit the model. A model is often trained through an iterative process, since is measuring the performance on predicting the outcome. This measure is used to update the model parameters in order to reduce the model error when applied to the data in the training set. The model parameters are a set of variables associated with the model, and their values are learned during the training process (Bishop, 2006; Munro & Mirsharif, n.d.).

After training the model, data from the validation set are used to evaluate the model for ability to generalize (the performance on unseen data). Usually, model hyperparameters are tuned and the evaluation becomes more biased as skill is incorporated into model configuration. However, the model occasionally sees the data, but never does it “Learn” from this. Therefore, the validation set affects a model, but only indirectly. After this, the best predictive model is selected (Bishop, 2006; Maleki et al., 2020).

In this case study, the training data is used to train the different classification models and the validation data is used to compare their performances and decide which one to take via performance characteristics such as accuracy, recall, precision, and F1-score. While training and validation set are split, stratify parameter is implemented making the division so that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify (Fernandes et al., 2018).

2.3.2.7. Performance metrics

Precision

Starting with precision value, also called positive prediction value, is defined as the relative amount of positive class predictions that actually belong to the positive class (Powers, 2007). Applying the logic on the current study, a precision value of 1 signifies that every player classified with the label ‘promising’, truly is ‘promising’. However, it is important to note that this has no influence on the amount of players classified as ‘inconclusive’ and ‘not promising’ that are truly ‘promising’ (Powers, 2007).

Recall

Regarding recall value, quantifies the number of positive class predictions made out of all positive examples in the dataset. REF Putting into words the study scope, out of the players that actually were considered as ‘Promising’, the model predicted perfectly this outcome for all of those players (Powers, 2007).

Accuracy

Accuracy consists in a performance metric that makes ratio between the number of correctly classified samples and the overall number of samples, where many researchers think is the most reasonable (Wang et al., 2007). This measure, by definition, it also works when labels are more than two, that is multiclass classification (Powers, n.d.). However, accuracy could not be considered as a reliable metric when the quantity of samples in one class is bigger than the quantity of samples in the other classes (unbalanced dataset), because it provides an overoptimistic estimation of the classifier ability on the majority class (Akosa, 2017).

F1 Score

On the other hand, F1 Score has been used in cases where data sizes are unbalanced being one of the most suitable evaluation metric. This metric calculates the relation between precision and recall. Therefore, if the result of F1 score is high, both precision and recall of the classifier indicate good results, as it normally emphasizes the lower value (Cristina et al., 2020).

Score

Score is known to return the mean accuracy on the given test data and labels. In multi-label classification, this is the subset accuracy which is a harsh metric since you require for each sample that each label set be correctly predicted (*Sklearn.Svm.SVC — Scikit-Learn 1.1.3 Documentation*, 2022).

3. METHODOLOGY

The research methodology discusses the applied methods during thesis elaboration and is composed by 3 main phases, which are Exploration Phase, Analytical Phase and Conclusive Phase.

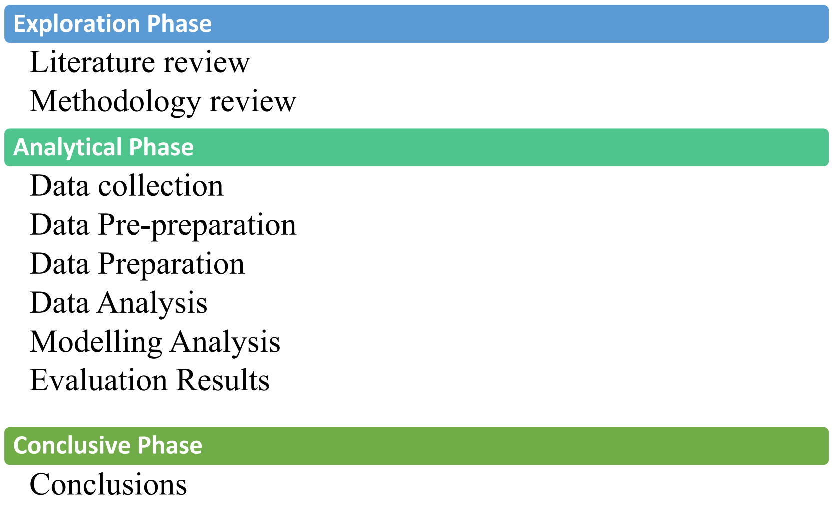


Figure 1 – Research methodology phases

First, the exploration phase starts with a literature review demonstrating knowledge and understanding of the academic literature on several topics such as Handball game, athletes' performance indicators, study tools and machine learning algorithms. And it ends with methodology review itself.

The second stage called analytical phase is supported on CRISP-DM – Cross Industry Standard Process for Data Mining - that will be used to describe and represent the different phases of this project. CRISP-DM consists in an industry-proven way to guide your data mining efforts, characterized by a process model with six phases providing an overview of the data mining life cycle. The methodology of this model will include description of the typical tasks involved with each step.

Initially, is crucial to dig deeper on how the data will be collected from the source to start the analyse of variables description. Understanding the meaning of all data will trigger the steps of searching ways to extract the essential content and pre-structure it. This step will be proceeded via Microsoft Excel using VBA code, extracting two columns ‘history’ and ‘registrations’ characterized by array of objects into two new tables.

Subsequently of digging and pre-preparate data, some specific variables will be explored, selected and cleaned on main table ‘Folha 1 – players’. The selection task will be proceeded also on ‘Folha History’ and ‘Folha Registrations’ in order to construct pivot_tables for moving forward with study analysis. At the same time the missing values will be filled or checked. All this process can be defined as data preparation where will be developed through Python language applied on Jupyter Notebook.

Following the preparation of data, each table or pivot_table will be analysed to gather future conclusions regarding the involvement of handball individuals registered on FAP. Data analysis it’s mainly composed by graphical representations, consisting in a phase which development effort is essential.

Starting with the most important step of all process, modelling analysis will apply two machine learning algorithms spread into four created models with different parameters focusing on a classification approach of athletes. The inherent dataset is splitted having 75% of training data and 25% for validation of the prediction models. Performance metrics will be used to measure the quality of those algorithms.

Lastly, the step ending up the cycle will be based on the evaluation and discussion of results obtained from both data analysis and modelling analysis tasks.

Finally, to close the 3 main phases of research methodology, the outcomes are responsible to establish conclusions that can make sense for the board management of FAP. Additionally, from there the improvements of future work can be discussed.

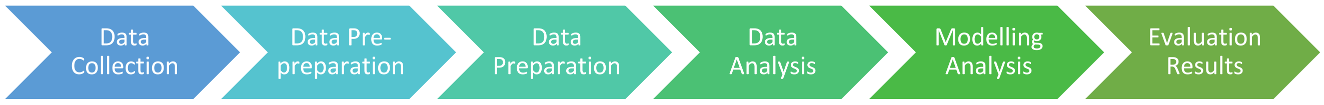


Figure 2 – CRISP-DM process model

4. STUDY DEVELOPMENT

4.1. Data Collection

The Federação de Andebol de Portugal (FAP) has a portal responsible for storing data related with several components of the sport that are essential for its functioning and evolution. This same portal corresponds to the Administrative and Sports Management System present in si.fpa.pt.

The data was made available by João Sousa, an IT member of FAP, through a link that contained a MongoDB dump with a password required to access and time limit for doing so. The database dump has records of the data structures that were used for this specific purpose for publication and future study analysis.

The mentioned link is below:
https://sousadax.myqnapcloud.com/share.cgi?ssid=83dc0e9a4b9649c981d276d4d7c35924

To obtain the information through the link, it was necessary to perform some **steps**:

1st Install and open docker (https://docs.docker.com/desktop/mac/install/);

2nd Download information and store in a folder called 'fpa-mongo-dump' (Appendix 1);

3rd Run command '*docker run -it -v /Users/miguelcoutinhonunes/Downloads/fpa-mongo-dump:/tmp/dump -p 27018:27017 --name mymongo -d mongo*';

4th Run command '*docker exec mymongo mongorestore /tmp/dump –gzip*' (Appendix 2);

5th Install MongoDB Compass (https://www.mongodb.com/try/download/compass);

6th Use defect connection string ('*mongodb://localhost:27018')* (Appendix 4 and 5).

After performing the steps mentioned above, it was possible to access the data through MongoDB Compass. The database named 'fpaDBV2-dev' contained about 16 collections:

- Associations (24.58 kB);
- Clubs (102.4 kB);
- Devices (147.46 kB);
- Favorites (57.34 kB);
- Fixturepublishes (28.67 kB);
- Gamedetails (137.03 MB);
- Games (19.72 MB);
- Ipaddresses (4.10 kB);
- Leagues (348.16 kB);
- NamefpaDB (4.10 kB);
- **Players (33.48 MB);**
- Rounds (3.19 MB);
- Seasons (20.48 kB);
- Steps (20.48 kB);
- Teams (2.86 MB);
- Users (7.94 MB);

Representative details are in Appendix 6, 7 and 8.

Among all the collections represented, the one that made the most sense to select was ‘Players’ database. This same choice was made for three reasons: 1) the purpose and theme of the study analysis; 2) being one of the collections with the largest storage size containing approximately 49000 documents; 3) this database contained information about the remaining 15 collections as can be seen in the metadata analysis (topic *Variables Description*).

Throughout MongoDB Compass, the 'Players' dataset was exported in CSV format performing the following steps:

1. Select ‘Export Collection’ (Appendix 9);
2. Choose ‘Export Full Collection’ (Appendix 10);
3. Select ‘CSV’ export file type (Appendix 11);
4. Do ‘Export’ (Appendix 11).

Variables Description

The CSV file from 'Players' database collection is representing information about players, referees, officials and technicians containing about 25 columns, where two of which are characterized by array of objects, 'history' and 'registrations', and one defined as array named 'roles'. In the following points we have the description of each one of the variables:

Variables	**Description**
__v	N/A
_id	ID of player, referee, official or technician
birthday	Birth date
cipa	Identification number of each person
createdAt	Data creation date
data.CIP_ARBITRO_TISN	Referee nominated for game
data.CIP_ATLETAS_TISN	Player CIPA available for game in the system
data.CIP_DATA_NASCIMENTO	Birth date
data.CIP_NOME	Name of player, referee, official or technician
data.CIP_NUMERO	Identification number of each person
data.CIP_OFICIAL_TISN	Official available for game in the system
data.CIP_TECNICO_TISN	Technician available for game in the system
data.EXTENSAO	Information extension
data.ID_ANEXO	Attachment ID
data.ID_CIPA	CIPA ID
data.ID_OBJECTO	CIPA ID
data.URL_FOTO	Photo of player URL
history	Array of information about each game played
id	CIPA ID

imgURL	Photo of player URL
name	Name of player, referee, official or technician
registrations	Array of information about each season played
role	Roles in Handball
roles	Array of roles in Handball
updatedAt	Data update date

Table 1 – Variables description of main table 'Players'

Since array of objects 'history' and 'registrations' will be crucial for the research study development, it makes sense to going deeper into constituent's instances.

Starting with 'history', as mentioned above, it's covering information about games played focusing on the most important aspects. This array of objects contains 14 variables, following the description of each:

Variables	**Description**
gameNumber	Game ID
homeTeam	Team playing at home
Result	Final result
awayTeam	Team playing away
date	Game date
league	League played
goals	Number of goals scored
yellow	Received yellow card
twom1	Two-minute card received for the first time
twom2	Two-minute card received for the second time
twom3	Two-minute card received for the third time[1]
red1	Red card received for two-minute card accumulation[1]
red2	Red card received directly
tp	N/A

Table 2 – Variables description of 'history' array of objects

[1] If two-minute card is received for the third time this means that player will consequently have red card ('red1' variable)

Regarding ‘registrations’ instance, addresses data about each season played. If is referencing an individual that is not a player, columns ‘refereeCategoryId’ and ‘refereeCategoryName’ or ‘oficialId’ and ‘officialName’ will be filled. There are 17 variables characterizing this array, which are:

Variables	**Description**
clubId	Club ID
clubName	Club name
seasonId	Season ID
seasonDesc	Season year
associationId	Association ID
associationDesc	Association name
stepId	Step ID
stepDesc	Step name
refereeCategoryId	Referee category ID
refereeCategoryName	Referee category name
cipaId	Identification number of each player
oficialId	Oficial ID
officialName	Oficial name
registrationType	Type of registration
nextStepId	Next step ID
nextStepName	Next step name
_id	Old ID

Table 3 - Variables description of ‘registrations’ array of objects

4.2. Data Pre-preparation

Considering the main purpose of the study and the variable description presented above, to discover the promising athletes was truly necessary to deconstruct ‘history’ and ‘registrations’ array of objects as a table for analysis.

To prepare that, the CSV file containing ‘Players’ database with all content was converted to an excel file (players_database.xlsm). From there, a VBA code was built with the objective to have isolated tables from array of objects, ‘history’ and ‘registrations’ in a new excel sheet (‘Folha Registrations’ and Folha History’).

Firstly, two functions were created to save all information about data and columns as a collection making this process simpler in next steps.

For data collection function (‘SaveDataToCollection’) is required to replace the string with ‘$oid’ inside to empty data in order to not interfere with save process. Besides that, was also necessary to replace the string of several team names because of quotation marks where is specifically explained on *Limitations* topic. On the construction of the function itself, firstly a loop for each character of string is created to save data into collection considering the count of existing quotation marks. If it is the third quotation mark or one that subtracted by 3 and divided by 4 gives 0 as remainder ((countAspas - 3) Mod 4 = 0), this means that the data can be saved into the collection. On the other hand, if it is the fourth quotation mark or one that subtracted by 4 and divided by 4 gives 0 as remainder ((countAspas - 4) Mod 4 = 0), will stop saving the string. Without considering counting the quotation marks, the word will be built from all characters of data string. (Appendix page 85)

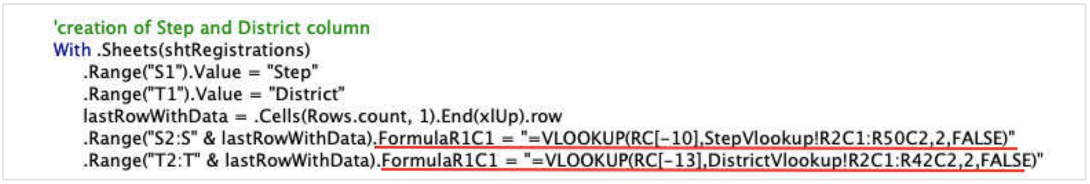
Regarding columns collection function (‘SaveColumnsToCollection’), same logic is applied without the need of replace strings at the beginning. (Appendix page 88)
For ‘ExtractJSON()’ program, an excel sheet is created (‘Folha Registrations’ or ‘Folha History’), if not existing, and columns from collection are stored in line one using for each loop. For ‘history’, inside of string data the number of columns ‘gamenumber’ are counted to discover how many rows will be created and, while ‘ID’ data is added to the rows that will be fulfilled. Also, a data range is set to select the place where we will copy our data. Finally, a for each loop is applied to cover and paste that data on all cells. This paste process was constructed only for this type of structure, not being adaptable for others. (Appendix page 88)
Considering future analysis, was also important to create manually two additional sheets on excel called ‘StepVlookup’ and ‘DistrictVlookup’. Regarding ‘StepVlookup’, consists in a table with all existent step names (‘stepDesc’ column) from ‘Registrations’ being renamed following the official article published by FAP at 2021 (https://portal.fpa.pt/wp-content/uploads/2021/03/COM.-20-21-N.-55.pdf). With this change the step names are now corrected (Appendix table 4). Concerning ‘DistrictVlookup’, contains all associations of FAP

in the first column (‘associationDesc’) and correspondent districts in the second (Appendix table 5). There are 18 districts in Portugal but the 2 islands, Madeira and Açores, were included also in the scope.

Inside ‘ExtractJSON()’ VBA code both tables ‘StepVlookup’ and ‘DistrictVlookup’ are used in ‘Folha Registrations’ generation, creating two additional columns called ‘Step’ and ‘District’ via VLOOKUP function macros for each (Appendix 12). Those macros were recorded by applying a simple action of pasting a function to the first cell of new column. Below is the example function responsible for ‘Step’ column:

=VLOOKUP([@stepDesc];StepVlookup!$A$2:$B$50;2;FALSE)

After this, the VBA ‘ExtractJSON()’ code dependable to build ‘Step’ and ‘District’ columns was created using those macros.



```
'creation of Step and District column
With .Sheets(shtRegistrations)
    .Range("S1").Value = "Step"
    .Range("T1").Value = "District"
    lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
    .Range("S2:S" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-10],StepVlookup!R2C1:R50C2,2,FALSE)"
    .Range("T2:T" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-13],DistrictVlookup!R2C1:R42C2,2,FALSE)"
```

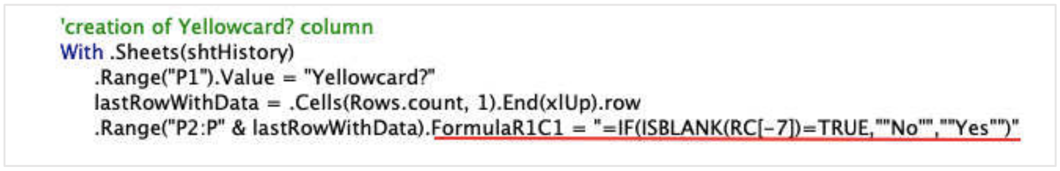
Figure 3 – ‘ExtractJSON()’ code using marked macros

On ‘Folha History’, was also created a new column called ‘Yellowcard?’ based on ‘yellow’ original one. Instead of having values with ‘X’, the new column was built with values ‘Yes’ or ‘No’ mentioning if player received a yellow card or not, respectively. As a result of this explanation is exemplified the function of new column creation:

=IF(ISBLANK([@yellow])=TRUE; "No"; "Yes")

Following the logic explained for ‘Step’ and ‘District’ columns, the column ‘Yellowcard?’ was also built through a recorded macro (Appendix 13) who is pasting the function represented above.

Accordingly, ‘ExtractJSON()’ program code will contain the following code for this new column production:



```
'creation of Yellowcard? column
With .Sheets(shtHistory)
    .Range("P1").Value = "Yellowcard?"
    lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
    .Range("P2:P" & lastRowWithData).FormulaR1C1 = "=IF(ISBLANK(RC[-7])=TRUE,""No"",""Yes"")"
```

Figure 4 - ‘ExtractJSON()’ code using marked macro

Furthermore, two additional macros were recorded executing the vertical and horizontal alignment and autofitting of ‘Folha Registrations’ and ‘Folha History’(Appendix 14). Those macros are also added on ‘ExtractJSON()’ (Appendix 15) having the VBA code ready for run.

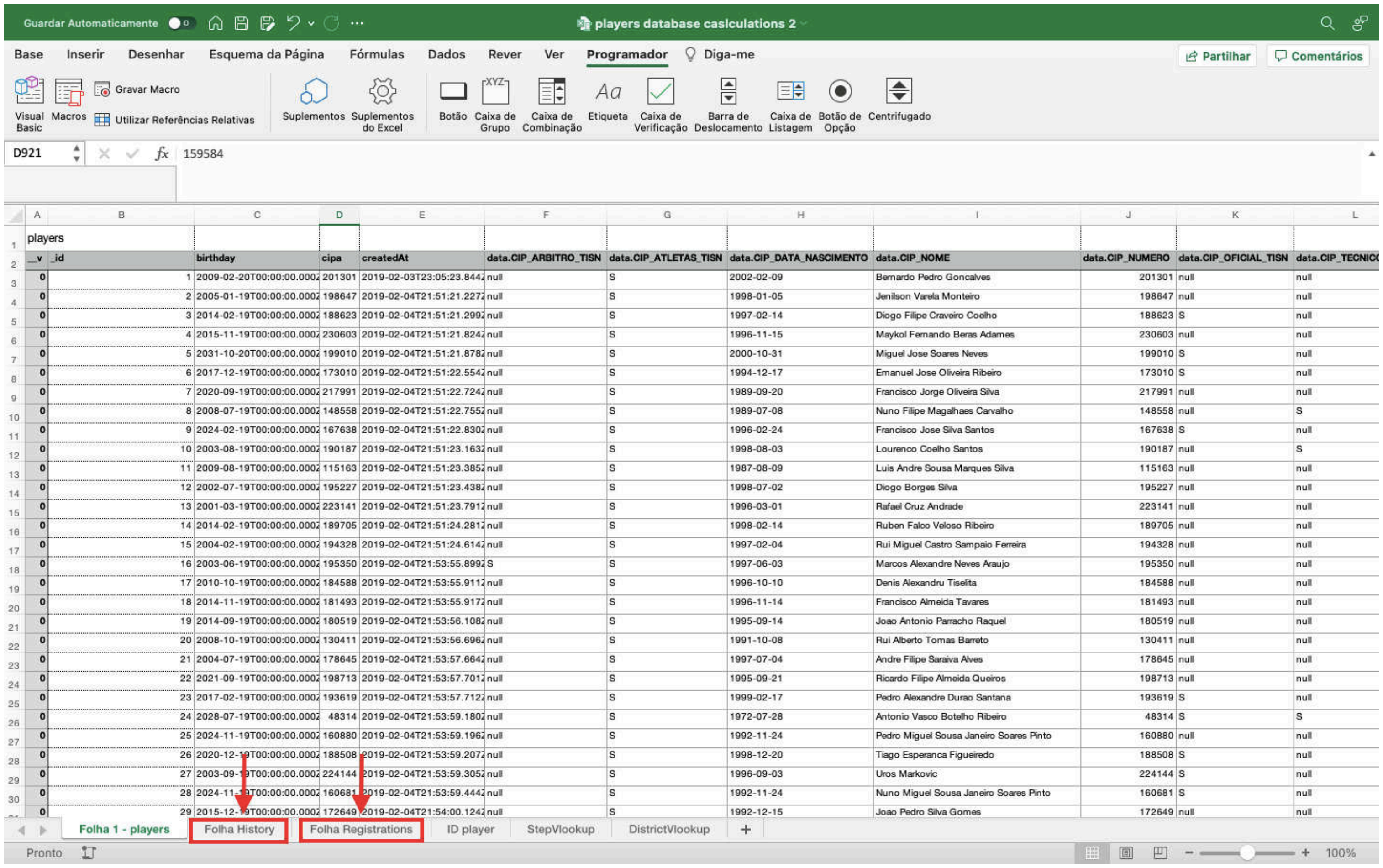
As mentioned before, having the structure of ‘Players’ database (‘Folha 1 – players’ sheet) the code described above can be ran following the next steps:

1st Go to ‘Programador’ option on excel;

2nd Select ‘Visual Basic’ (Appendix 16);

3rd Click on Play icon ‘Executar Sub/FormulárioDeUtilizador’ (Appendix 17).

The code takes approximately 35 minutes to run. After this, ‘Folha History’ and ‘Folha Registrations’ are created and ready to be used.



players											
__v	_id	birthday	cipa	createdAt	data.CIP_ARBITRO_TISN	data.CIP_ATLETAS_TISN	data.CIP_DATA_NASCIMENTO	data.CIP_NOME	data.CIP_NUMERO	data.CIP_OFICIAL_TISN	data.CIP_TECNIC
0	1	2009-02-20T00:00:00.000Z	201301	2019-02-03T23:05:23.844Z	null	S	2002-02-09	Bernardo Pedro Goncalves	201301	null	null
0	2	2005-01-19T00:00:00.000Z	198647	2019-02-04T21:51:21.227Z	null	S	1998-01-05	Jenilson Varela Monteiro	198647	null	null
0	3	2014-02-19T00:00:00.000Z	188623	2019-02-04T21:51:21.299Z	null	S	1997-02-14	Diogo Filipe Craveiro Coelho	188623	S	null
0	4	2015-11-19T00:00:00.000Z	230603	2019-02-04T21:51:21.824Z	null	S	1996-11-15	Maykol Fernando Beras Adames	230603	null	null
0	5	2031-10-20T00:00:00.000Z	199010	2019-02-04T21:51:21.878Z	null	S	2000-10-31	Miguel Jose Soares Neves	199010	S	null
0	6	2017-12-19T00:00:00.000Z	173010	2019-02-04T21:51:22.554Z	null	S	1994-12-17	Emanuel Jose Oliveira Ribeiro	173010	S	null
0	7	2020-09-19T00:00:00.000Z	217991	2019-02-04T21:51:22.724Z	null	S	1989-09-20	Francisco Jorge Oliveira Silva	217991	null	null
0	8	2008-07-19T00:00:00.000Z	148558	2019-02-04T21:51:22.755Z	null	S	1989-07-08	Nuno Filipe Magalhaes Carvalho	148558	null	S
0	9	2024-02-19T00:00:00.000Z	167638	2019-02-04T21:51:22.830Z	null	S	1996-02-24	Francisco Jose Silva Santos	167638	S	null
0	10	2003-08-19T00:00:00.000Z	190187	2019-02-04T21:51:23.163Z	null	S	1998-08-03	Lourenco Coelho Santos	190187	null	S
0	11	2009-08-19T00:00:00.000Z	115163	2019-02-04T21:51:23.385Z	null	S	1987-08-09	Luis Andre Sousa Marques Silva	115163	null	null
0	12	2002-07-19T00:00:00.000Z	195227	2019-02-04T21:51:23.438Z	null	S	1998-07-02	Diogo Borges Silva	195227	null	null
0	13	2001-03-19T00:00:00.000Z	223141	2019-02-04T21:51:23.791Z	null	S	1996-03-01	Rafael Cruz Andrade	223141	null	null
0	14	2014-02-19T00:00:00.000Z	189705	2019-02-04T21:51:24.281Z	null	S	1998-02-14	Ruben Falco Veloso Ribeiro	189705	null	null
0	15	2004-02-19T00:00:00.000Z	194328	2019-02-04T21:51:24.614Z	null	S	1997-02-04	Rui Miguel Castro Sampaio Ferreira	194328	null	null
0	16	2003-06-19T00:00:00.000Z	195350	2019-02-04T21:53:55.899Z	S	S	1997-06-03	Marcos Alexandre Neves Araujo	195350	null	null
0	17	2010-10-19T00:00:00.000Z	184588	2019-02-04T21:53:55.911Z	null	S	1996-10-10	Denis Alexandru Tiselita	184588	null	null
0	18	2014-11-19T00:00:00.000Z	181493	2019-02-04T21:53:55.917Z	null	S	1996-11-14	Francisco Almeida Tavares	181493	null	null
0	19	2014-09-19T00:00:00.000Z	180519	2019-02-04T21:53:56.108Z	null	S	1995-09-14	Joao Antonio Parracho Raquel	180519	null	null
0	20	2008-10-19T00:00:00.000Z	130411	2019-02-04T21:53:56.696Z	null	S	1991-10-08	Rui Alberto Tomas Barreto	130411	null	null
0	21	2004-07-19T00:00:00.000Z	178645	2019-02-04T21:53:57.664Z	null	S	1997-07-04	Andre Filipe Saraiva Alves	178645	null	null
0	22	2021-09-19T00:00:00.000Z	198713	2019-02-04T21:53:57.701Z	null	S	1995-09-21	Ricardo Filipe Almeida Queiros	198713	null	null
0	23	2017-02-19T00:00:00.000Z	193619	2019-02-04T21:53:57.712Z	null	S	1999-02-17	Pedro Alexandre Durao Santana	193619	S	null
0	24	2028-07-19T00:00:00.000Z	48314	2019-02-04T21:53:59.180Z	null	S	1972-07-28	Antonio Vasco Botelho Ribeiro	48314	S	S
0	25	2024-11-19T00:00:00.000Z	160880	2019-02-04T21:53:59.196Z	null	S	1992-11-24	Pedro Miguel Sousa Janeiro Soares Pinto	160880	null	null
0	26	2020-12-19T00:00:00.000Z	188508	2019-02-04T21:53:59.207Z	null	S	1998-12-20	Tiago Esperanca Figueiredo	188508	S	null
0	27	2003-09-19T00:00:00.000Z	224144	2019-02-04T21:53:59.305Z	null	S	1996-09-03	Uros Markovic	224144	S	null
0	28	2024-11-19T00:00:00.000Z	160681	2019-02-04T21:53:59.444Z	null	S	1992-11-24	Nuno Miguel Sousa Janeiro Soares Pinto	160681	S	null
0	29	2015-12-19T00:00:00.000Z	172649	2019-02-04T21:54:00.124Z	null	S	1992-12-15	Joao Pedro Silva Gomes	172649	null	null

Figure 5 – 'Folha History' and 'Folha Registrations' created sheets

4.3. Data Preparation

Succeeding all phases described above, the data preparation starts at Jupyter Notebook script called 'Study code' using Python language. Jupyter Notebook is launched from Anaconda Navigator desktop as explained before *Literature Review* topic.

After the pre-preparation on excel file running VBA code, three sheets were read on 'Study code' script: 'Folha 1 – players', 'Folha History' and 'Folha Registrations'.

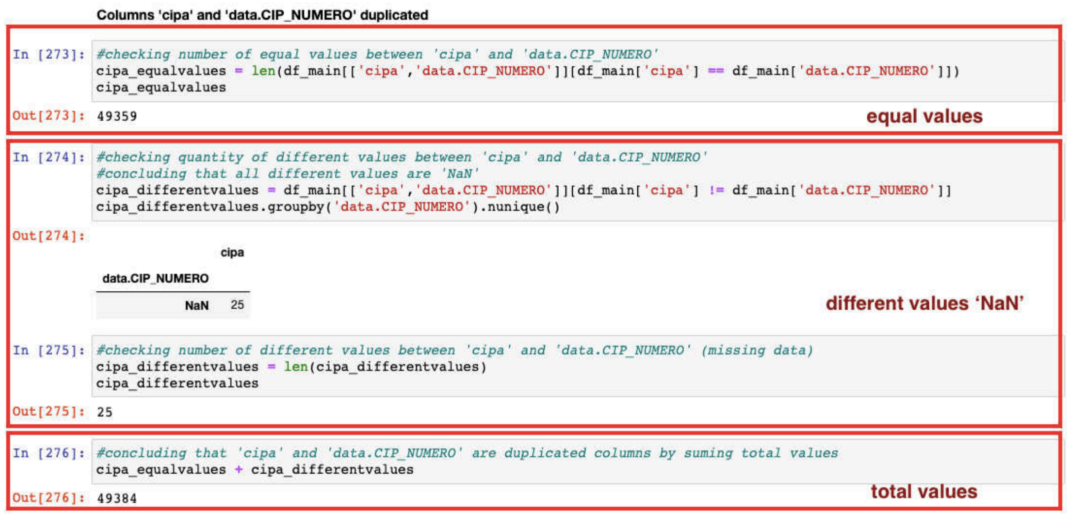
<u>‘Folha 1 – players’</u>

Firstly, was prepared ‘Folha 1 – players’ sheet, where ‘registrations’ and ‘history’ data are stored as an array of objects. The total number of rows for this main table was 49384 (Appendix 18).

Concerning duplicated data, was discovered four sets of repeated columns which is unnecessary for the analysis. The sets are the following:

- ‘cipa’ and ‘data.CIP_NUMERO’;
- ‘data.ID_CIPA’, ‘data.ID_OBJECTO’ and ‘id’;
- ‘data.URL_FOTO’ and ‘imgURL’;
- ‘data.CIP_NOME’ and ‘name’.

From python script ‘Study code’, for each set was verified the number of equal and different values between columns. The different values on all sets were always ‘NaN’, so these cannot be considered as different due to missing values. After, the total of those different values and equals was summed giving 49384 instances, which is representing the total rows of dataset. This situation occurred for all sets, therefore with this sum can be concluded that the explored columns are duplicated. On Figure 6 is present one set of repeated columns investigation example:


Columns 'cipa' and 'data.CIP_NUMERO' duplicated

```
In [273]: #checking number of equal values between 'cipa' and 'data.CIP_NUMERO'
          cipa_equalvalues = len(df_main[['cipa','data.CIP_NUMERO']][df_main['cipa'] == df_main['data.CIP_NUMERO']])
          cipa_equalvalues
Out[273]: 49359
```

```
In [274]: #checking quantity of different values between 'cipa' and 'data.CIP_NUMERO'
          #concluding that all different values are 'NaN'
          cipa_differentvalues = df_main[['cipa','data.CIP_NUMERO']][df_main['cipa'] != df_main['data.CIP_NUMERO']]
          cipa_differentvalues.groupby('data.CIP_NUMERO').nunique()
Out[274]:
                        cipa
          data.CIP_NUMERO
                    NaN   25
```

```
In [275]: #checking number of different values between 'cipa' and 'data.CIP_NUMERO' (missing data)
          cipa_differentvalues = len(cipa_differentvalues)
          cipa_differentvalues
Out[275]: 25
```

```
In [276]: #concluding that 'cipa' and 'data.CIP_NUMERO' are duplicated columns by suming total values
          cipa_equalvalues + cipa_differentvalues
Out[276]: 49384
```

Figure 6 – Duplicated columns example

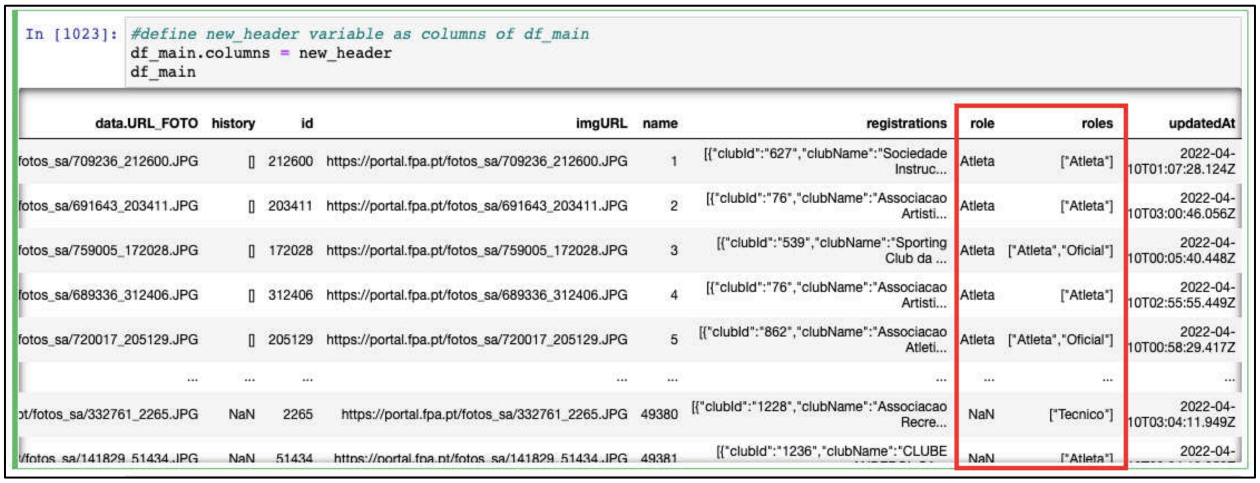
Afterwards, come up the exploration of ‘birthday’ column that contained the birthday date of athletes, referees, officials and technicians. While starting the search, ‘birthday_year’ column was created from first four characters of 'birthday' resulting just the birth year. Basic condition came to be employed on this column to check if there were registered individuals with birth years superior to 2018. It gave a result of 19972 people, representing 40%, which doesn't make sense since that are people registered with 3 years old or less (Appendix 19).

Additionally, there were a relationship between ‘birthday_year’ and ‘data.CIP_DATA_NASCIMENTO’ column where 'data.CIP_DATA_NASCIMENTO' day is always equal to the last two digits of 'birthday_year', independently of sample investigated (Appendix 20). Consequently, taking into account this investigation, 'birthday' and 'birthday_year' columns were dropped (Appendix 21).

Furthermore, column 'data.CIP_DATA_NASCIMENTO' was also checked. Also containing the birthday date, it was retrieved only the birth year and was applied the exact same condition as for ‘birthday_year’ analysis. This time the outcome was just 7 individuals that were accordingly removed from main table (Appendix 22). In order to avoid noisy data, the condition investigating people who born before 1900 year was also created giving a result of 28 individuals. These people came to be eliminated (Appendix 23) giving more simplicity on the graphical representations constructed on *Data Analysis* topic.

Plus, missing data of columns called ‘data.CIP_ATLETAS_TISN’, 'data.CIP_ARBITRO_TISN', 'data.CIP_OFICIAL_TISN' and 'data.CIP_TECNICO_TISN' were investigated using ‘value_counts()’ fuction (Appendix 25). The last three columns mentioned before came to be removed due to have approximately 100% of missing data (‘NaN’ values).

Focusing now on the relation between column ‘roles’ and ‘role’. As you can see in the below image those columns are quite similar approaching the same content:



```
In [1023]: #define new_header variable as columns of df_main
           df_main.columns = new_header
           df_main
```

data.URL_FOTO	history	id	imgURL	name	registrations	role	roles	updatedAt
fotos_sa/709236_212600.JPG	[]	212600	https://portal.fpa.pt/fotos_sa/709236_212600.JPG	1	[{"clubId":"627","clubName":"Sociedade Instruc...	Atleta	["Atleta"]	2022-04-10T01:07:28.124Z
fotos_sa/691643_203411.JPG	[]	203411	https://portal.fpa.pt/fotos_sa/691643_203411.JPG	2	[{"clubId":"76","clubName":"Associacao Artisti...	Atleta	["Atleta"]	2022-04-10T03:00:46.056Z
fotos_sa/759005_172028.JPG	[]	172028	https://portal.fpa.pt/fotos_sa/759005_172028.JPG	3	[{"clubId":"539","clubName":"Sporting Club da ...	Atleta	["Atleta","Oficial"]	2022-04-10T00:05:40.448Z
fotos_sa/689336_312406.JPG	[]	312406	https://portal.fpa.pt/fotos_sa/689336_312406.JPG	4	[{"clubId":"76","clubName":"Associacao Artisti...	Atleta	["Atleta"]	2022-04-10T02:55:55.449Z
fotos_sa/720017_205129.JPG	[]	205129	https://portal.fpa.pt/fotos_sa/720017_205129.JPG	5	[{"clubId":"862","clubName":"Associacao Atleti...	Atleta	["Atleta","Oficial"]	2022-04-10T00:58:29.417Z
...	...	...	...	...	...	...	...	...
pt/fotos_sa/332761_2265.JPG	NaN	2265	https://portal.fpa.pt/fotos_sa/332761_2265.JPG	49380	[{"clubId":"1228","clubName":"Associacao Recre...	NaN	["Tecnico"]	2022-04-10T03:04:11.949Z
/fotos_sa/141829_51434.JPG	NaN	51434	https://portal.fpa.pt/fotos_sa/141829_51434.JPG	49381	[{"clubId":"1236","clubName":"CLUBE	NaN	["Atleta"]	2022-04-

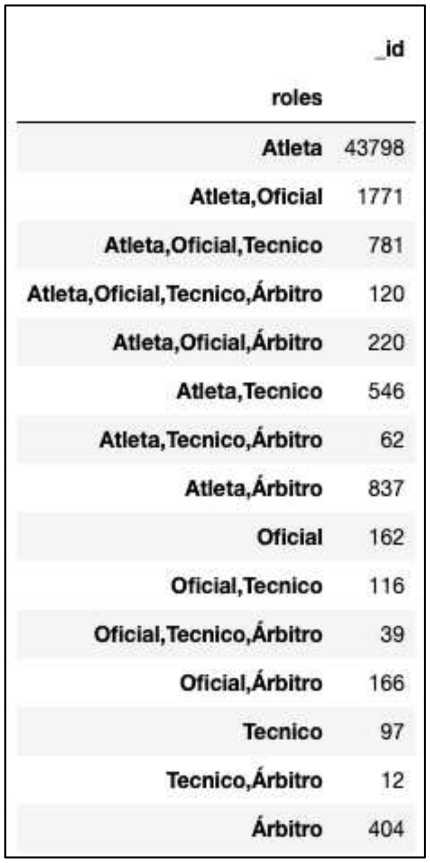
Figure 7 – 'roles' and 'role' columns similarity

To decide which one should be selected for analysis, a simple verification was made. A sum of missing data for each column gave the following results (Appendix 26):

- Column 'role': 26456 missing values;
- Column 'roles': 6 missing values.

Based on this calculation, was decided to select 'roles' variable since 'role' column missing data is representing more than 50% of it. Also, missing values rows of 'roles' were eliminated from table.

Since this column came from MongoDB as an array, it contains all values with quotation marks and square brackets, i.e. ["Atleta"]. Therefore, these characters were removed with strip and replace functions (Appendix 27). From there was created a pivot_table (Appendix 28) counting the existent types of 'roles':

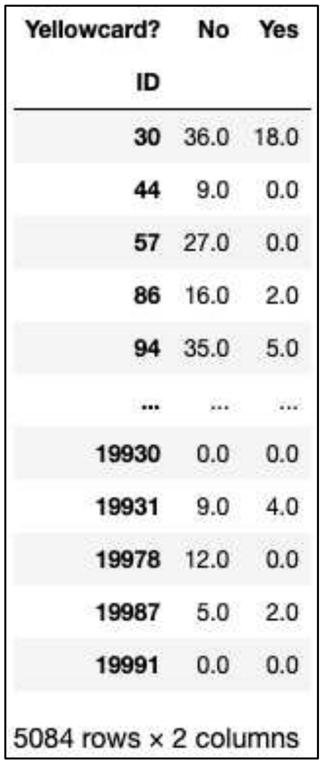


roles	_id
Atleta	43798
Atleta,Oficial	1771
Atleta,Oficial,Tecnico	781
Atleta,Oficial,Tecnico,Árbitro	120
Atleta,Oficial,Árbitro	220
Atleta,Tecnico	546
Atleta,Tecnico,Árbitro	62
Atleta,Árbitro	837
Oficial	162
Oficial,Tecnico	116
Oficial,Tecnico,Árbitro	39
Oficial,Árbitro	166
Tecnico	97
Tecnico,Árbitro	12
Árbitro	404

Figure 8 – Pivot_table 'Roles'

<u>**'Folha History'**</u>

Secondly 'Folha History' sheet, generated from VBA code, has been explored emphasizing in the relation between the number of goals scored by players having yellow card received or not. After filling missing data with 0 value on column 'goals', a pivot_table called 'yellowcard' was created grouping the total of goals by 'ID' player for having yellow card (column 'Yes') or not (column 'No') as can be seen in Figure 9:



```
Yellowcard?   No   Yes
         ID
         30  36.0  18.0
         44   9.0   0.0
         57  27.0   0.0
         86  16.0   2.0
         94  35.0   5.0
        ...   ...   ...
      19930   0.0   0.0
      19931   9.0   4.0
      19978  12.0   0.0
      19987   5.0   2.0
      19991   0.0   0.0

5084 rows × 2 columns
```

Figure 9 – Pivot_table 'yellowcard'

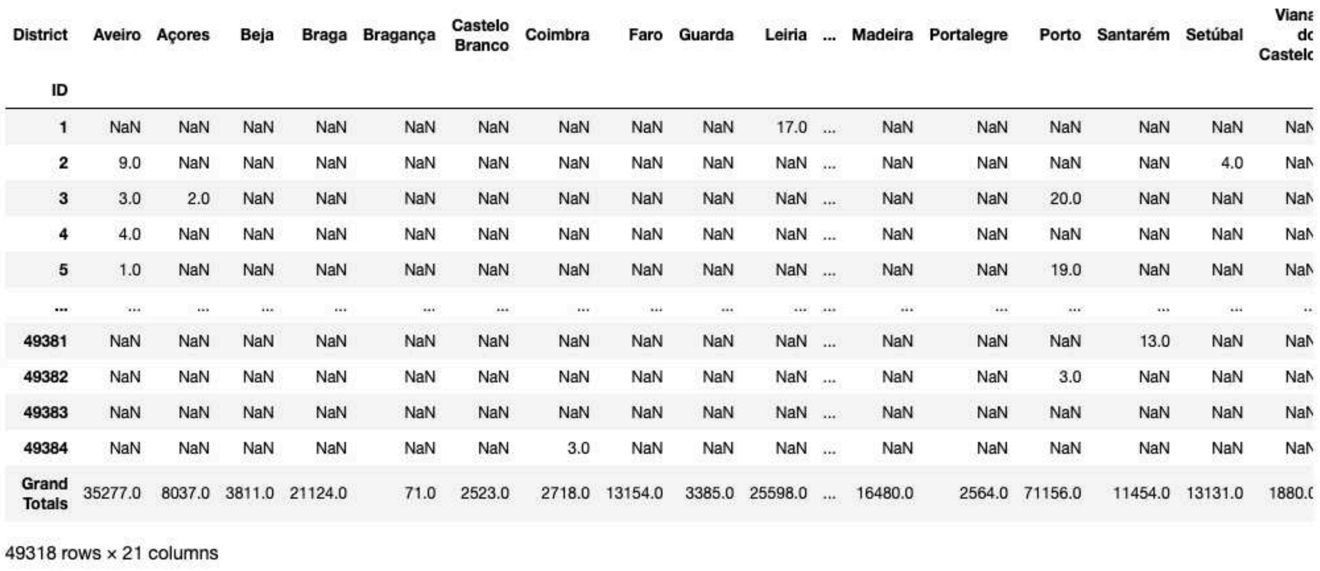
'Folha Registrations'

Before starting to create pivot_tables from 'Folha Registrations' table was fundamental to explore some variables. As described on *Variables description* topic, table of registrations contain information about players, referees and officials.
Considering this, column 'refereeCategoryName' came to be checked using a pivot_table called 'referee_checkstep' and a simple function. With 'referee_checkstep' was verified the number of 'refereeCategoryName' with step name (column 'Step') and the function got the same information as 'referee_checkstep' but specifying which rows are (Appendix 31 and 32). Therefore, results in a total of 12 rows that were removed from the table due to referees having 'Step' which doesn't make sense since only players can own it. The same was applied for column 'officialName' however officials with 'Step' were not giving any result because they were also removed on referee analysis, being part of same index (Appendix 33 and 34). Taking into consideration the previous explanation, when analysing data aggerated somehow with 'Step' column will be guaranteed that we are dealing only with players.

Finally, on ‘Folha Registrations’ sheet, there were two more pivot_tables constructed: ‘district’ and ‘promising’.

Pivot_table ‘district’

Pivot_table ‘district’ is representing each player aggregation count of ‘Step’ per district (Appendix 35). This means that the count of ‘Step’ column is representing the number of years that a player spent in a certain district. Summing up the previous details, Figure 10 is representing the pivot_table.



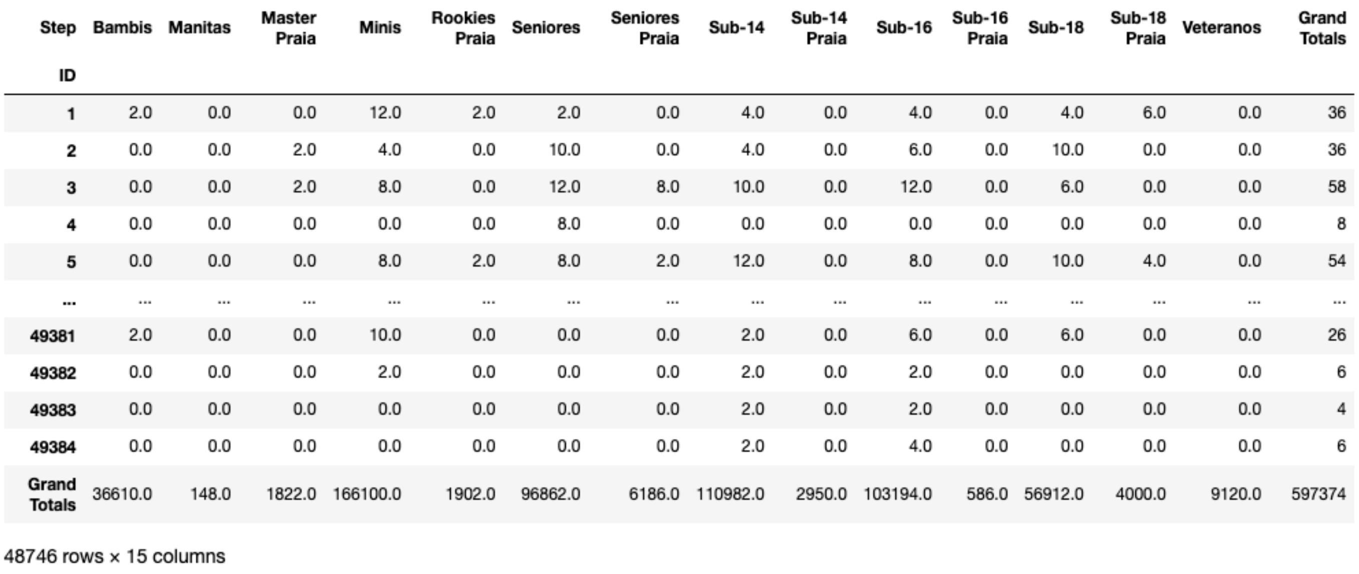
District	Aveiro	Açores	Beja	Braga	Bragança	Castelo Branco	Coimbra	Faro	Guarda	Leiria	...	Madeira	Portalegre	Porto	Santarém	Setúbal	Vian d Castel
ID																	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	17.0	...	NaN	NaN	NaN	NaN	NaN	Na
2	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	4.0	Na
3	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	20.0	NaN	NaN	Na
4	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	Na
5	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	19.0	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
49381	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	13.0	NaN	Na
49382	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	3.0	NaN	NaN	Na
49383	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	Na
49384	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	Na
Grand Totals	35277.0	8037.0	3811.0	21124.0	71.0	2523.0	2718.0	13154.0	3385.0	25598.0	...	16480.0	2564.0	71156.0	11454.0	13131.0	1880.

49318 rows × 21 columns

Figure 10 – Pivot_table ‘district’

Pivot_table ‘promising’

In this specific case, the data preparation phase for ‘promising’ pivot_table happens for Modelling Analysis focusing on the main goal of this study, discover promising athletes of handball. Thus, this pivot_table is generated by counting the number of ‘seasonDesc’ (season year) for each player stored in ‘Step’ columns, as can be seen below:



Step	Bambis	Manitas	Master Praia	Minis	Rookies Praia	Seniores	Seniores Praia	Sub-14	Sub-14 Praia	Sub-16	Sub-16 Praia	Sub-18	Sub-18 Praia	Veteranos	Grand Totals
ID															
1	2.0	0.0	0.0	12.0	2.0	2.0	0.0	4.0	0.0	4.0	0.0	4.0	6.0	0.0	36
2	0.0	0.0	2.0	4.0	0.0	10.0	0.0	4.0	0.0	6.0	0.0	10.0	0.0	0.0	36
3	0.0	0.0	2.0	8.0	0.0	12.0	8.0	10.0	0.0	12.0	0.0	6.0	0.0	0.0	58
4	0.0	0.0	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8
5	0.0	0.0	0.0	8.0	2.0	8.0	2.0	12.0	0.0	8.0	0.0	10.0	4.0	0.0	54
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49381	2.0	0.0	0.0	10.0	0.0	0.0	0.0	2.0	0.0	6.0	0.0	6.0	0.0	0.0	26
49382	0.0	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	2.0	0.0	0.0	0.0	0.0	6
49383	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.0	0.0	0.0	0.0	0.0	4
49384	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	4.0	0.0	0.0	0.0	0.0	6
Grand Totals	36610.0	148.0	1822.0	166100.0	1902.0	96862.0	6186.0	110982.0	2950.0	103194.0	586.0	56912.0	4000.0	9120.0	597374

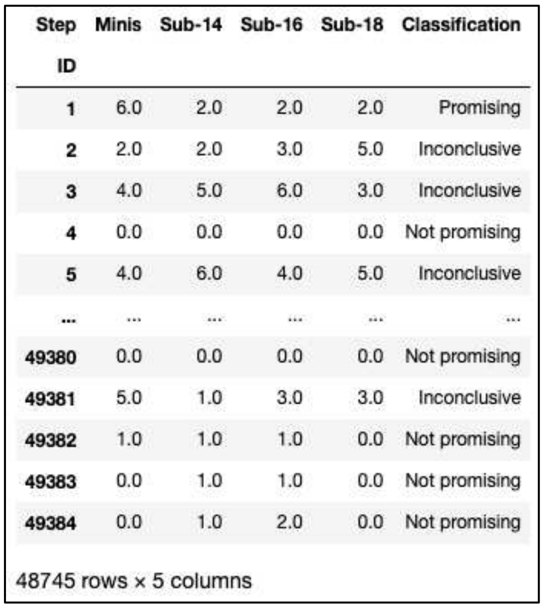
48746 rows × 15 columns

Figure 11 – Pivot_table 'promising'

After filling the missing values from pivot_table with '0' since they don't contain any information for the count, a function called 'f' is built. Function 'f' approaches classification conditions for the number of years played on each Step 'Minis', 'Sub-14', 'Sub-16' and 'Sub-18'. Discussing together with members of FAP and following the idea discussed on Performance Indicators topic of *Literature Review*, was decided that a promising player would staying at least two years at 'Minis' and two exact years at 'Sub-14', 'sub-16' and 'Sub-18'. On the other hand, a not promising athlete will have at least three steps, among the analyzed four, equal or less than one year. We are talking about exact years at step 'Sub-14', 'sub-16' and 'Sub-18' since is the maximum of years possible that a player can be. If he/she has more than two years on these steps, then will be considered as inconclusive (Appendix 42 and 43). Below is represented 'f' python code function:

```
def f(row):
    if row['Minis'] >= 2 and row['Sub-14'] == 2 and row['Sub-16'] == 2 and row['Sub-18'] == 2:
        val = 'Promising'
    elif (row['Minis'] <= 1 and row['Sub-14'] <= 1 and row['Sub-16'] <= 1 and row['Sub-18'] == 2) or (row['Minis'] <= 1 and row['Sub-14'] == 2 and row['Sub-16'] <= 1 and row['Sub-18'] <= 1) or (row['Minis'] <= 1 and row['Sub-14'] <= 1 and row['Sub-16'] == 2 and row['Sub-18'] <= 1) or (row['Minis'] >= 2 and row['Sub-14'] <= 1 and row['Sub-16'] <= 1 and row['Sub-18'] <= 1) or (row['Minis'] <= 1 and row['Sub-14'] <= 1 and row['Sub-16'] <= 1 and row['Sub-18'] <= 1):
        val = 'Not promising'
    else:
        val = 'Inconclusive'
    return val
```

Succeeding the creation of 'f' function, this one was applied on 'promising' pivot_table by adding a column named as 'Classification' (Appendix 44). Then all 'Step' columns that were not being part of 'f' creation were deleted (Appendix 45). Then, before starting Modelling Analysis, we get the following table:



Step	Minis	Sub-14	Sub-16	Sub-18	Classification
ID					
1	6.0	2.0	2.0	2.0	Promising
2	2.0	2.0	3.0	5.0	Inconclusive
3	4.0	5.0	6.0	3.0	Inconclusive
4	0.0	0.0	0.0	0.0	Not promising
5	4.0	6.0	4.0	5.0	Inconclusive
...	...	...	...	...	...
49380	0.0	0.0	0.0	0.0	Not promising
49381	5.0	1.0	3.0	3.0	Inconclusive
49382	1.0	1.0	1.0	0.0	Not promising
49383	0.0	1.0	1.0	0.0	Not promising
49384	0.0	1.0	2.0	0.0	Not promising

48745 rows × 5 columns

Figure 12 – Table 'promising' for modelling analysis

4.4. Data Analysis

Following the previous *Data Preparation* phase, script 'Study code' will be used for the data analysis focusing on visual representations in order to evaluate the results and take some conclusions more easily in the next steps. Those representations will require different packages as plotly and matplotlib. Tables or pivot_tables from 'Folha 1 – players', 'Folha History' and 'Folha Registrations' will be analysed.

‘Folha 1 – players’

Regarding ‘Folha 1 – players’, after cleaning 'data.CIP_DATA_NASCIMENTO' column having birth years superior to 1900 and inferior to 2018, a variable called ‘birthday_year’ was created from a value count function of column mentioned. The results of this code function had all birth years as index and quantity of those as values. Therefore, from ‘birthday_year’ variable,
a bar plot was built showing the number of individuals for each birth year (Appendix 24):

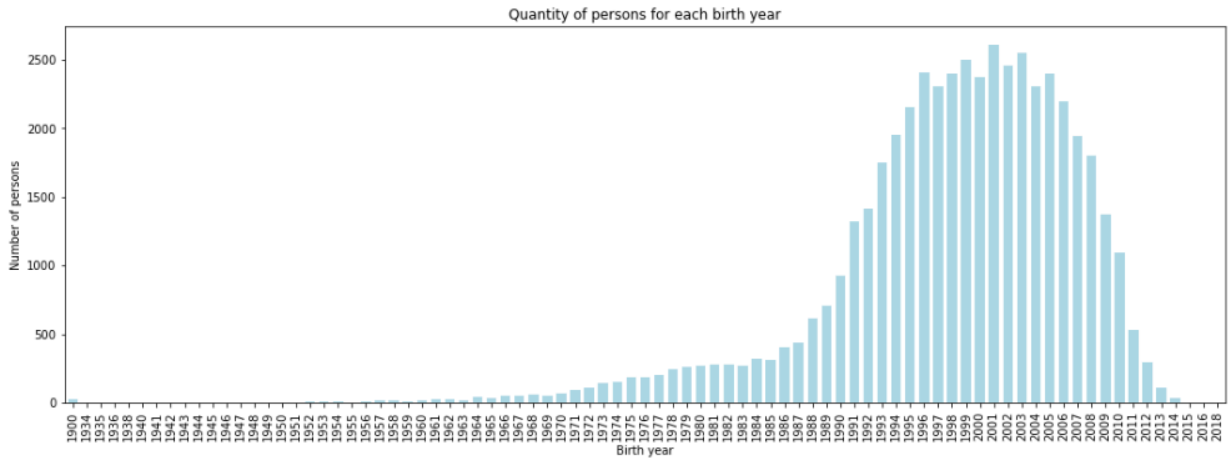


Figure 13 – Number of individuals for each birth year

The figure above was plotted using matplotlib python tool where values are insert on y-axis and birth year are sorted respecting the timeline.

About ‘Roles’ pivot_table, having ‘_id’ column as values, is applied count aggregation function meaning that Figure 8 is structuring the quantity of people for each type of role. The line ‘Grand Totals’ and blank were dropped from ‘Roles’ pivot_table being ready for graphical schematization (Appendix 29). By analysing this figure, we can directly say that are 15 different sets of roles in this database either with one, two or three roles in the same set. As a result, this pivot_table was plotted using matplotlib package, making two horizontal bar plots containing sorted outcome. The first graphic was employed having all different types of roles:

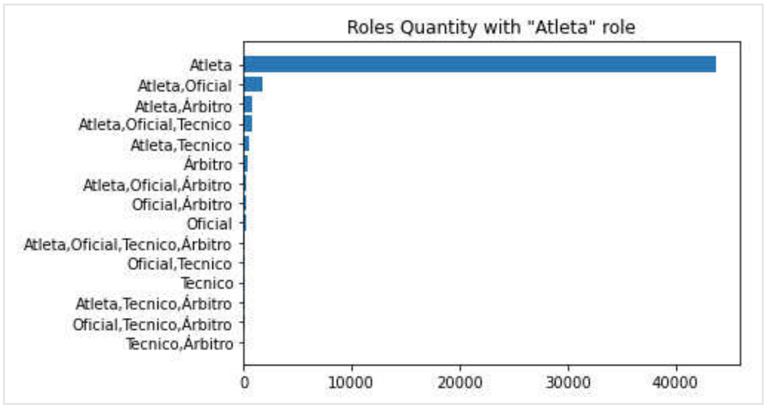


Figure 14 – Number of role sets

However, considering the huge difference between the quantity of 'Atleta' role and the remaining ones, a second plot was generated without it. This time some colours were implemented on all bars and the associated values from '_id' column placed in front of each one, being straightforwardly interpreted:

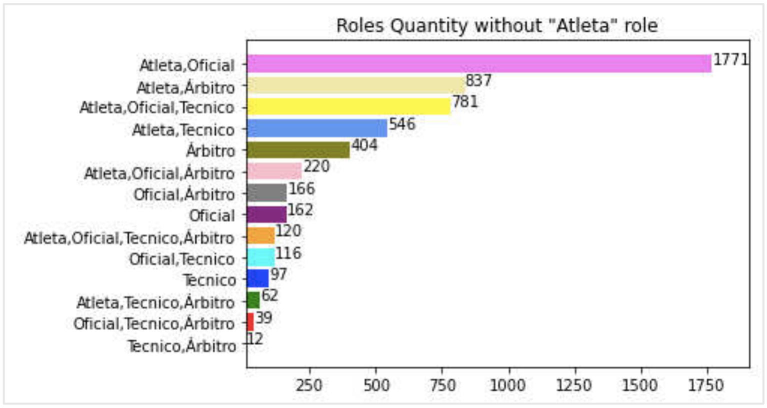


Figure 15 - Number of roles sets without 'Atleta' role

<u>‘Folha History’</u>

Considering ‘Folha History’, as mentioned before, ‘yellowcard’ pivot_table from Figure 9 is applying on each player a sum aggregation function using ‘goals’ as values for the case of yellow card received or not. After creating it, ‘Grand totals’ information are deleted and ‘NaN’ values had to be replaced by 0 to be able to generate the graph properly (Appendix 30). Based on this pivot_table, a stacked bar chart came to be built representing the number of goals scored with and without yellow card always using a sample of 30 players:

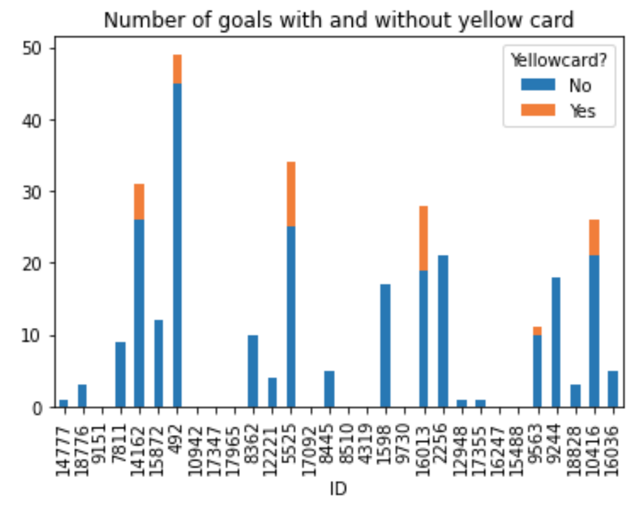
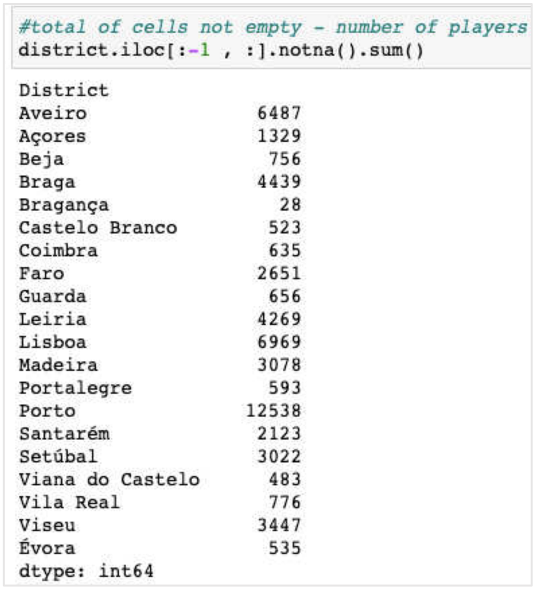


Figure 16 – Stacked bar chart

<u>‘Folha Registrations’</u>

While analysing ‘district’ pivot_table in Figure 10, and following the explanation gave in Data Preparation topic, was defined the objective to calculate the average years of formation per district and map/plot the results. We are dealing with 18 Portuguese districts and 2 island (Madeira and Açores). To achieve this average calculation was required to extract:

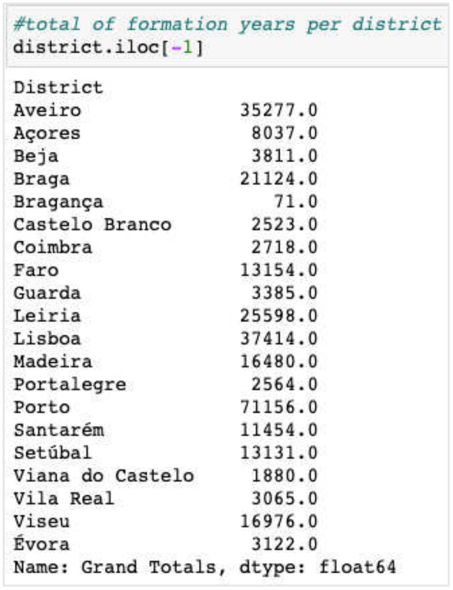
- Total number of players that were being part of formation of each district:



```
#total of cells not empty - number of players
district.iloc[:-1 , :].notna().sum()

District
Aveiro              6487
Açores              1329
Beja                 756
Braga               4439
Bragança              28
Castelo Branco       523
Coimbra              635
Faro                2651
Guarda               656
Leiria              4269
Lisboa              6969
Madeira             3078
Portalegre           593
Porto              12538
Santarém            2123
Setúbal             3022
Viana do Castelo     483
Vila Real            776
Viseu               3447
Évora                535
dtype: int64
```

Figure 17 – Number of players per district

- Total of formation years of each district:

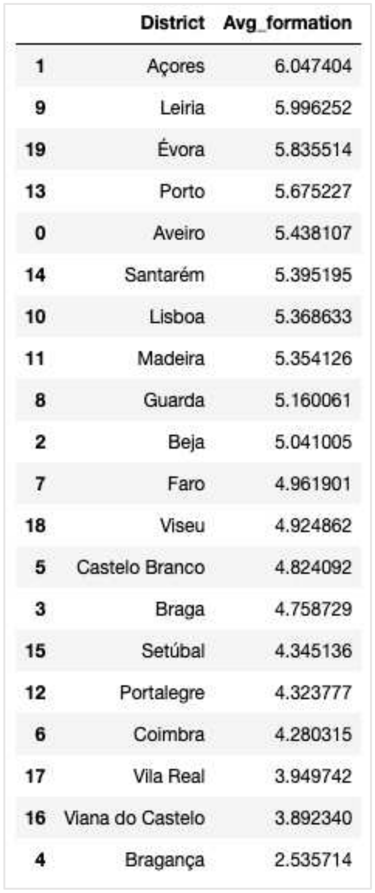


```
#total of formation years per district
district.iloc[-1]

District
Aveiro              35277.0
Açores               8037.0
Beja                 3811.0
Braga               21124.0
Bragança               71.0
Castelo Branco       2523.0
Coimbra              2718.0
Faro                13154.0
Guarda               3385.0
Leiria              25598.0
Lisboa              37414.0
Madeira             16480.0
Portalegre           2564.0
Porto               71156.0
Santarém            11454.0
Setúbal             13131.0
Viana do Castelo     1880.0
Vila Real            3065.0
Viseu               16976.0
Évora                3122.0
Name: Grand Totals, dtype: float64
```

Figure 18 – Formation years total per district

After this extraction, the calculation formula applied was (Appendix 36):

$$\text{Average of formation years per district} = \frac{\text{Total of formation years of each district}}{\text{Total number of players that were being part of formation of each district}}$$



	District	Avg_formation
1	Açores	6.047404
9	Leiria	5.996252
19	Évora	5.835514
13	Porto	5.675227
0	Aveiro	5.438107
14	Santarém	5.395195
10	Lisboa	5.368633
11	Madeira	5.354126
8	Guarda	5.160061
2	Beja	5.041005
7	Faro	4.961901
18	Viseu	4.924862
5	Castelo Branco	4.824092
3	Braga	4.758729
15	Setúbal	4.345136
12	Portalegre	4.323777
6	Coimbra	4.280315
17	Vila Real	3.949742
16	Viana do Castelo	3.892340
4	Bragança	2.535714

Figure 19 – Average formation years table

As a result, a table with average formation years (column ‘Avg_formation’) for each district (column ‘District’) has been structured as can be seen in the Figure 19 above. Therefore, two graphical representations, map and bar chart, were made from the same content both using plotly package.

Portugal districts map

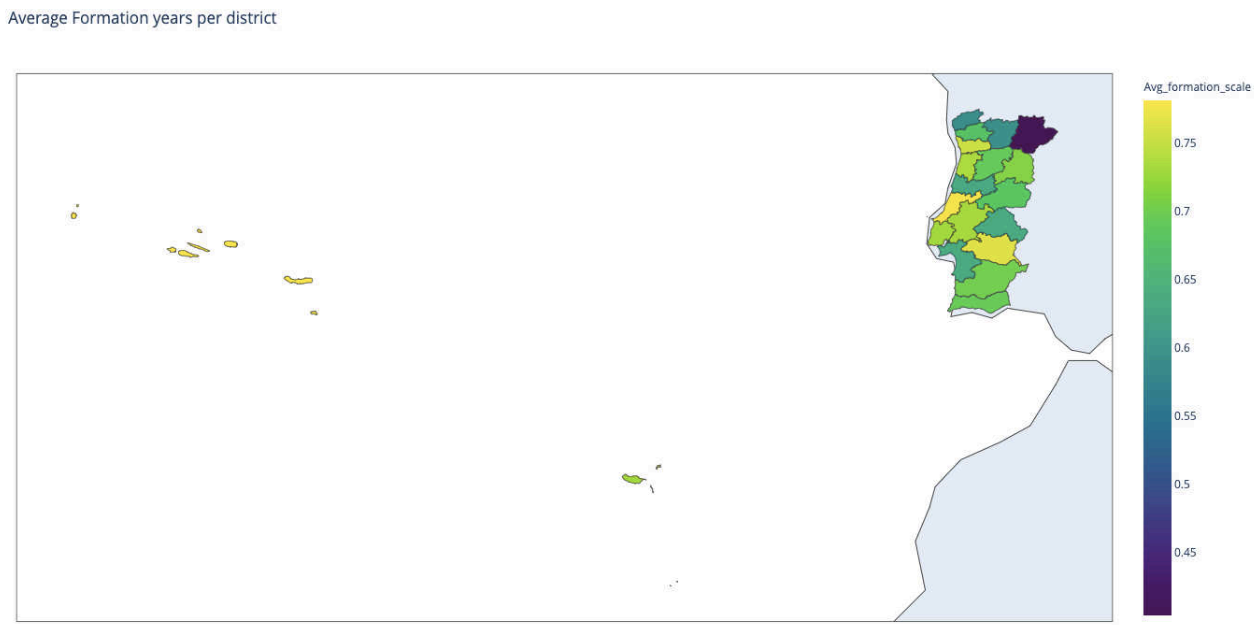


Figure 20 – Average formation years per district map

Regarding Portugal districts map, there was some complexity to accomplish this. First, became necessary to extract the district coordinates with respective code (‘dis_code’) as JSON language. After a dictionary was generated to store ‘dis_code’ referent to each district name (Appendix 37). This dictionary served to create the ‘id’ column into ‘district’ pivot_table (Appendix 38). Additionally, ‘Avg_formation_scale’ column was built from the logarithm values of ‘Avg_formation’. This new column is used for the purpose of map representation have more colour variations due to values being closer to each other. On Appendix topic can be find two plots of ‘Avg_formation_scale’ and ‘Avg_formation’ proving this sentence (Appendix 39 and 40).

Inside of function responsible to generate the map, px.choropleth(), column ‘id’ was mentioned inside instance ‘locations’ and column ‘Avg_formation_scale’ in instance ‘color’ (Appendix 41). Finally, Portugal came to be plotted as showed on Figure 20.

Bar chart

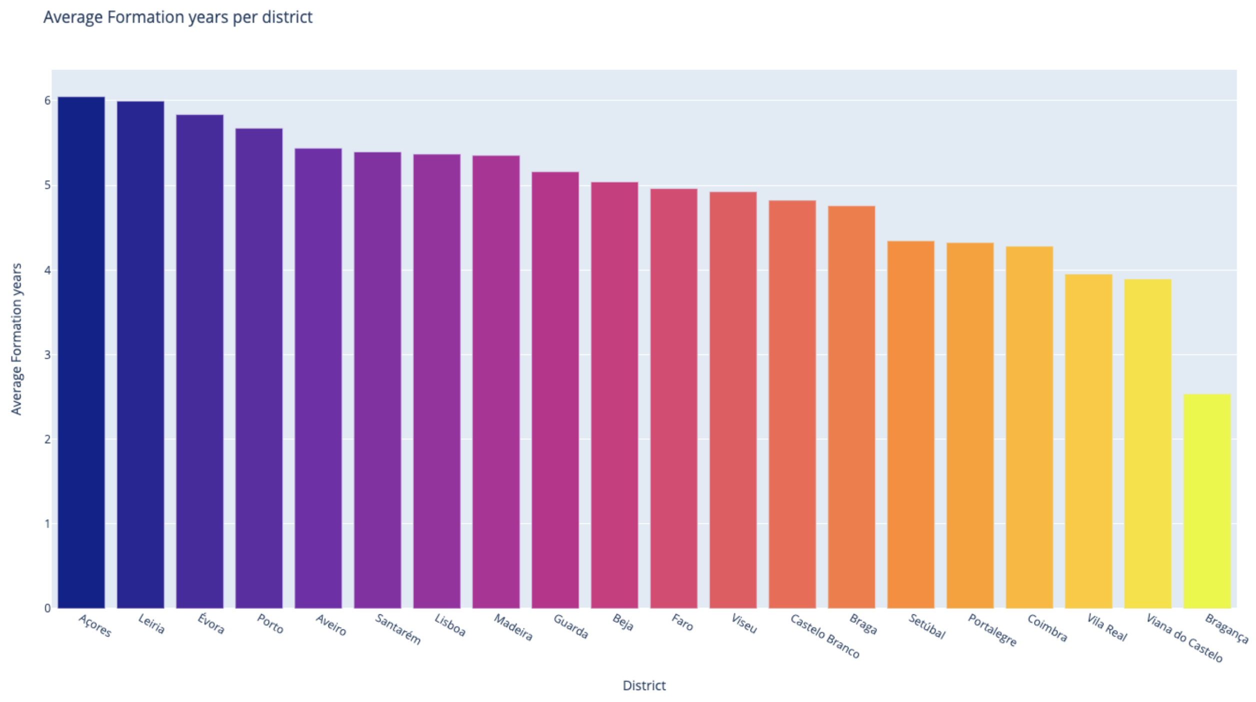


Figure 21 – Average formation years per district bar chart

4.5. Modelling Analysis

The 5th phase of methodology workflow process will take place on 'Study code' script, as Data Preparation and Data Analysis steps, concentrating on measure performance metrics: precision, recall, f1-score, accuracy and score. The Modelling Analysis will be applied over 'promising' pivot_table generated from 'Folha Registrations' sheet.

By analyzing Figure 11, pivot_table 'promising' was ready to start the prediction study. The implemented algorithms were Support Vector Machine (SVM) and K-Nearest Neighbors (KNN). Before starting, 'Classification' column has been assigned as dependent variable called 'target' and a stratified split of 'promising' dataset came to be done containing a validation size of 25% (Appendix 46).

Regarding Support Vector Machine, two models were constructed called ‘modelSVM’ and ‘modelSVM_100’. First, ‘modelSVM’ was created with default parameters giving a score of 99,9% for both training and validation data. The classification report of validation gave 100% for metric accuracy, precision, recall and f1-score except for recall of ‘Promising’ players that got 99%. This means that predicted model ‘modelSVM’ is almost perfect only with the exception that, out of all the players that actually were considered as ‘Promising’, the outcome was correctly predicted for 99% of those (Appendix 47).

Secondly on ‘modelSVM_100’, is implemented with ‘C’ parameter equal to 100 which means we are looking for a smaller-margin hyperplane to get all the training points classified correctly. As a result, we obtained a score of 100% on both training and validation data. The validation data classification report got 100% on all metrics, concluding that there weren’t existent misclassifications (Appendix48).

Respecting K-Nearest Neighbors algorithm, also two models were created: ‘modelKNN’ and ‘modelKNN50’. Start by ‘modelKNN’ with default parameters, for both training and validation data was achieved a score of 99,9%. The classification report results on 100% on all metrics, excluding recall that also result on 99%. Therefore, is applied the exact same logic as ‘modelSVM’ (Appendix 47).
Finally, the ‘modelKNN50’ containing 50 neighbors to check when a player is being classified (‘n_neighbors’ parameter), gave 99,8% score on training and validation data. Following the classification report, all metrics of ‘Inconclusive’ and ‘Not promising’ items got 100% with an exception for ‘Promising’ classification. Besides the 100% of accuracy and precision, evaluating validation data, ‘Promising’ item got 89% on recall which means that among the promising players, the model predicted correctly 89% of them. Metric F1-score result on 94% meaning that ‘modelKNN50’ is good on predicting if the athlete is classified as ‘Promising’ or not but could be better. The 89% of recall is the reason for f1-score 94% metric since this is defined as harmonic mean of precision and recall.
(Appendix 49).

4.6. Evaluation Results

Finally, we reach the last phase of the study, evaluating the results of all visual illustrations from 'Folha 1 – players', 'Folha History' and 'Folha Registrations'. Interpretating and debating these outcomes will surely clear some information that was somehow covered before. The results evaluated are from *Data Analysis* and *Modelling Analysis* chapters.

'Folha 1 – players'

The evaluation results start by verifying Figure 13 representing the number of individuals for each birth year. We can directly visualize that from 1970, the number of people began to increase gradually until 2001, and from that year onwards there was a decrease in the same direction until the most recent days. It is worth noting the fact that the x-axis has between 1934 and 2018 values of almost every year of birth, making it certainly a more appealing graphic result. Most people were born in 2001, following shortly after the year 2003 and 1999 among others. Focusing on the recent years, there are 153 people with less than 10 years old are which means that is 0,31% of data (Appendix 50).

Sum, the largest percentage of people registered in the FAP database is 21 years old and most individuals are between 12 and 32 years old, representing about 82% (40216 people) of the database (Appendix 51). On the other hand, people who born before 1972 and after 2014, having more than 50 years and less than 8, are just indicating 1,5% of all people (Appendix 52).

Secondly, while analysing Figure 14, visually 'Atleta' was by far the biggest role having 43798 people among the 15 different types of roles. This quantity characterizes 89% of entirely information. It should be noted that the first 5 sets of roles all have the responsibility of 'atleta', which means that once you are an athlete, there is a certain possibility of exercising another role later on. On the other hand, the 'arbitro' is present in the last 3 sets, realizing that for these cases it is highly unlikely that someone who was a referee, for example, will also become a coach or coach and official.

As mentioned on Data Analysis, 'Atleta' role was removed from the graphic due to the considerable quantity difference among others. Consequently, Figure 15 plot was constructed. By cheching this new graph, 'Atleta,Oficial' role was the higher with 1771 people and 'Técnico,Árbitro' the lowest having only 12 individuals. In a conclusion, if we remove 'Atleta' role from the database, it is possible to verify that when you are an athlete, there is a 33% chance of becoming an official (Appendix 53). In contrast, only 0.2% of those who were referees become coaches (Appendix 54).

<u>'Folha History'</u>

Interpreting Figure 16, between the 30 different players ID, when don't have yellow card during the game they are willing to score more goals, and this happens independently of which people are present in the sample. Checking ID 492 as an example, on a total of around 49 goals, this player scored approximately 45 goals without yellow card and just 4 with it. When looking at all goals scored by yellow and non-yellowed players in the same game, it's easy to see that 'No' yellow card goals are always superior i.e., ID 14162, 492, 5525, 16013, etc. Accordingly, having a yellow card can negatively influence the number of goals scored on an athlete.

<u>'Folha Registrations'</u>

<u>Pivot_table 'district'</u>

As can be analysed on the Figure 20 and 21 'Açores' district had the highest average of formation time with 6,05 years. Right after it came 'Leiria' and 'Évora', being central districts, also close to 6 years. Unfortunately, 'Bragança' was the lowest location and with a certain distance among others. This district had 2,5 formation years and the nearer one, 'Viana do Castelo', obtained 3,9 years.

<u>Pivot_table 'promising'</u>

Finally, by evaluating Figure 22, we have a comparison of 4 different score metrics between the analyzed models from 2 classification algorithms (KNN and SVM). The 'modelKNN' and 'modelKNN50' are K-nearest neighbors model algorithms. For Support Vector Machines we have 'modelSVM' and 'modelSVM_100'. The evaluation was focus on validation data since is measuring the performance of the unseen data.
Thus, between the four models 'modelSVM_100' is the preferable one containing 100% of score, meaning that looking for a smaller-margin hyperplane (C parameter) is a better approach. In opposition, the lowest score belongs to 'modelKNN50' having 99,8% which means that even the worst model can be considered almost as perfect on predicting.

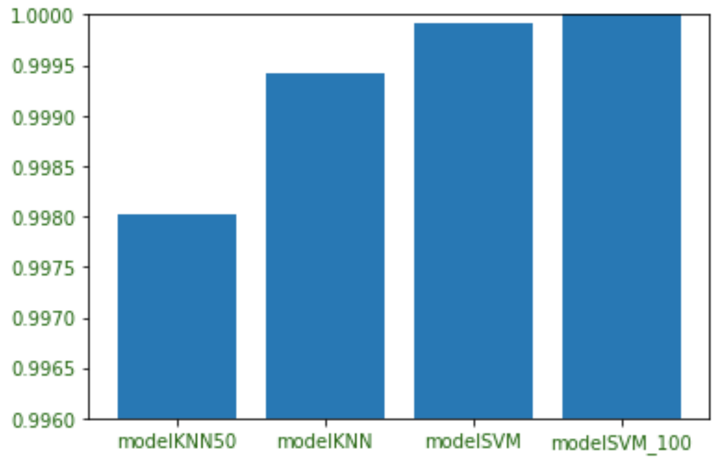


Figure 22 – Comparison bar plot of predictive models score

Discussion

The main findings from visual presentations evaluated:

1. From Figure 13 and Appendix 50 there are only 153 people under 10 years old, representing 0.31% of the database, concluding that the practice of this sport starts a bit later;
2. Evaluating Figure 14, it is concluded that when you are an athlete there is a greater predisposition to assume another role in the future;
3. Interpreting Figure 16, it is accomplished that not receiving the yellow card will positively influence the number of goals scored;

4. By checking Figure 20, is important to notice that the three northernmost districts of Portugal (Bragança, Vila Real and Viana do Castelo) were the ones with lowest average.

Regarding Modelling Analysis findings interpreted:

1. Models that don't have default parameters got the highest and lowest score result;
2. In this study, is important to notice that SVM algorithm models are better compared to KNN;
3. Taking into account the condition applied about formation years by step, promising athletes are representing 1% of the prediction.

5. CONCLUSIONS

This chapter concludes the developed work of this study by reflecting the most important conclusions, limitations, and future work. It also allows an understanding of whether the specified objectives were achieved or not.

5.1. Synthesis of the developed work

Initially, after FAP providing a cloud link, 'Players' database was collected using a Docker container. Following this, there was a pre-preparation of the data using a created VBA code program. Starting with python language, a script came to be prepared by cleaning missing values and creating pivot_tables with information for the visual presentations applied on each excel sheet below:

- 'Folha 1 – players': 3 bar charts;
- 'Folha History': 1 stacked bar chart;
- 'Folha Registrations': 1 bar chart and 1 map.

Afterwards, a Modelling Analysis for predicting promising athletes was performed on 'Folha Registrations' and study results were evaluated and discussed.

Concluding, following the main study objective, promising handball athletes were successfully predicted based on the number of formation years by building classification models. The intermediate objectives came to be also achieved through collecting, preparating and analysing graphically the dataset resulting on outcomes evaluation. From here, conclusions and main findings were defined on *Discussion* topic.

5.2. Limitations

While working on the research study, there were encountered some limitations and aspects that could be improved.

The first limitation came after providence of link with the MongoDB dump and downloading the zip, where some difficulties were felt while installing MongoDB Macbook to access the data. During the execution of the steps to download the information, when using the default connection (step 6) no data appeared in MongoDB Compass (Appendix 3), although used the suggested code. After a call with João Sousa, we understood that it was necessary to stop the container created in docker ('mymongo') and eliminate it. After carrying out the same steps, but with localhost 27018, instead of the 27017 used initially, it was possible to verify all the information in MongoDB Compass (Appendix 4).

Secondly, other limitation happened while creating a function responsible to save data as collection in VBA code, several team names had quotation marks inside e.g. 'AC FAFE \"B\"' or 'CD S.Bernardo \"A\"'. As mentioned before on *Data Pre-preparation* topic the counting of quotation marks was essential to save data strings. If the team's name itself contains these marks ("") consequently the store of information into new sheet ('Folha History') will not be correct, having the following structure present on Appendix 55. Therefore, in order to solve this issue was fundamental to replace in VBA code the string of team names without having quotation marks inside, as an example for 'AC FAFE \"B\"':

str = Replace(str, "A. C. FAFE \" & chr(34) & "B\" & chr(34), "A. C. FAFE \B\")

This action was made for 187 different club names (Appendix page 85).

Finally, the great preventive limitation consisted on the lack of enough variables for the players classification through predictive algorithms to be sufficiently plausible. Consequently, the only base variable to classify promising athletes was the number of formation years.

5.3. Future work

As for the next steps, the VBA code program must be updated for all types of databases retrieved from MongoDB, as it is built specifically for the 'Players' table. As a practical example of the current code, we have the creation of the column structure for future store where line 32 is being referenced because it is the first one containing arrays in the column 'history' and 'registrations'.

Specifying the use of the data, an analysis can also be developed dividing and highlighting the differences between Indoor Handball and Beach Handball. In addition, a study comparing male and female athletes can be prepared.

In the future, it is advisable for FAP to collect more variables data that includes the psychological component, strength, power and constitution, thus having more useful performance indicators to identify promising athletes in the sport.

REFERENCES

Abidin, I. Z., Juahir, H., Azid, A., Mustafa, A. D., & Azaman, F. (2015). Application of Excel-VBA for Computation of Water Quality Index and Air Pollutant Index. *Malaysian Journal of Analytical Sciences*, *19*(5), 1056–1064.

Aceto, G., Ciuonzo, D., Montieri, A., & Pescapé, A. (2018). Multi-classification approaches for classifying mobile app traffic. *Journal of Network and Computer Applications*, *103*, 131–145. https://doi.org/10.1016/j.jnca.2017.11.007

Akosa, J. (2017). *Predictive Accuracy : A Misleading Performance Measure for Highly Imbalanced Data*.

Anaconda Navigator — Anaconda documentation. (2022). https://docs.anaconda.com/navigator/

Aoudi, W., & Barbar, A. M. (2016). Support Vector Machines : A Distance-Based Approach to Multi-Class Classification. *IEEE International Multidisciplinary Conference on Engineering Technology*.

Baker, J., Horton, S., Robertson-Wilson, J., & Wall, M. (2003). Nurturing sport expertise: Factors influencing the development of elite athlete. *Journal of Sports Science and Medicine*, *2*, 1–9.

Banker, K., Hawkins, T., Bakkum, P., Verch, S., & Garrett, D. (2016). *MongoDB In Action* (J. B. Susan Conant, J. H. Brian Hanafee, & W. Thielen (Eds.); Second). Manning Publications Co.

Bernard, M., Dwi Minarti, E., & Hutajulu, M. (2018). Constructing Student's Mathematical Understanding Skills and Self Confidence: Math Game with Visual Basic Application for Microsoft Excel in Learning Phytagoras at Junior High School. *International Journal of Engineering & Technology*, *7*(3.2), 732–736. https://doi.org/10.14419/ijet.v7i3.2.18738

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python* (J. Steele & L. Dimant (Eds.); First). O'Reilly Media, Inc.

Biscaia, P., Coelho, E., Hernández-Mendo, A., & Alves, J. (2018). Processamento da Informação e Antecipação em Jogadoras de Andebol de Elite: Da Formação ao Alto Nível. *Revista Iberoamericana de Psicología Del Ejercicio y El Deporte*, *13*(2), 179–191.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* (M. Jordan, J. Kleinberg, & B. Scho (Eds.)). Springer Science+Business Media, LLC

Bui, T. (2015). Analysis of Docker Security. *Aalto University T-110.5291 Seminar on Network Security*. http://arxiv.org/abs/1501.02967

Cezar, H., Cristina, N., Cristina, H., & Narcis, N. (2013). Formation Strategy for the Young Handball Players. *Procedia - Social and Behavioral Sciences*, *93*, 1936–1939. https://doi.org/10.1016/j.sbspro.2013.10.144

Chaudhry, A. K., Kalwar, M. A., Khan, M. A., & Shaikh, S. A. (2021). Improving the Efficiency of Small Management Information System by Using VBA. *International Journal of Science and Engineering Investigations*, *10*(111), 7–13.

Chodorow, K. (2013). *MongoDB: The Definitive Guide* (A. Spencer & K. Ebrahim (Eds.); Seconda). O'Reilly Media, Inc.

Cirujano, J., & Zhu, Z. (2013). Automatic reporting for manpower resources. *Proceedings, Annual Conference - Canadian Society for Civil Engineering*.

Claudino, J. G., Capanema, D.-O., De-Souza, T.-V., Serrão, J. C., Machado Pereira, A.-C., & Nassis, G.-P. (2019). Current Approaches to the Use of Artificial Intelligence for Injury Risk Assessment and Performance Prediction in Team Sports: a Systematic Review. *Sports Medicine - Open*, *5*(28), 2–12.

Cristina, A., Rodrigues, N., Pereira, A. S., Manuel, R., Mendes, S., Araújo, A. G., Couceiro, M. S., & Figueiredo, A. J. (2020). Using Artificial Intelligence for Pattern Recognition in a Sports Context. *Sensors*, *20*(3040), 1–18. https://doi.org/10.3390/s20113040

Debanne, T., & Volossovitch, A. (2022). Team Regulatory Strategies and Performance in Elite Handball. *Research Quarterly for Exercise and Sport*, *00*(00), 1–12. https://doi.org/10.1080/02701367.2021.1948955

Docker overview | Docker Documentation. (2022). https://docs.docker.com/get-started/overview/

Eljinini, M. A. H. (2022). *Practical Data Structures* (First). M.A. Eljinini.

England Handball Association Rules and Regulations 2018.19 (Issue June). (2018).

Fernandes, D. D., Matuck, G. R., Montini, D. A., Alberto, L., & Dias, V. (2018). *A Stratified Sampling Algorithm for Artificial Neural Networks*. https://doi.org/10.1007/978-3-319-77028-4

Getting started with VBA in Office | Microsoft Learn. (n.d.). Retrieved November 2, 2022, from https://learn.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* (Third). Elsevier Inc.

Horvat, T., Job, J., & Medved, V. (2018). Prediction of Euroleague Games based on Supervised Classification Algorithm k -Nearest Neighbours. *Proceedings Ofthe 6th International Congress on Sport Sciences Research AndTechnology Support*, 203–207. https://doi.org/10.5220/0006893502030207

Ibrahim Akl, A.-R., & Hassan, A. A. (2017). An Artificial Neural Network Approach for Predicting Kinematics in Handball Throws. *American Journal of Sports Science*, *5*(5), 35–39. https://doi.org/10.11648/j.ajss.20170505.13

IX. Rules of the Game a) Indoor Handball. (2022). In *Paper Knowledge . Toward a Media History of Documents* (1 March 20).

Jacobs, A., Robinson, D. F., & De Paolo, C. A. (2016). Using excel to make strategic managerial decisions. *Journal of Information Systems Education*, *27*(2), 93–98.

Kelleher, J. D., Namee, B. Mac, & D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics*.

Langaroudi, M. K., & Yamaghani, M. R. (2019a). *Learning and Computational Intelligence*. *5*(1).

Langaroudi, M. K., & Yamaghani, M. R. (2019b). Sports Result Prediction Based on Machine Learning and Computational Intelligence Approaches: A Survey. *J. ADV COMP ENG TECHNOL*, *5*(1), 27–36.

Lee, L., Kerler, W., & Ivancevich, D. (2018). Beyond Excel: Software Tools and the Accounting Curriculum. *AIS Educator Journal*, *13*(1), 44–61. https://doi.org/10.3194/1935-8156-13.1.44

Loss, L. S., Wang, H., Shao, Y., Zhou, S., Zhang, C., & Xiu, N. (2022). Support Vector Machine Classifier via L0/1 Soft-Margin Loss. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, *44*(10), 7253–7265.

Machine Learning: What it is and why it matters | SAS. (n.d.). Retrieved July 20, 2021, from https://www.sas.com/en_us/insights/analytics/machine-learning.html

Maleki, F., Muthukrishnan, N., & Ovens, K. (2020). Machine Learning Algorithm Validation From Essentials to Advanced Applications and Implications for Regulatory Certification and Deployment. *Neuroimaging Clinics of NA*, *30*, 433–445. https://doi.org/10.1016/j.nic.2020.08.004

Menezes, R. P., Ramos, N. C., Marques, R. F. R., & Nunomura, M. (2018). Teaching handball to U-16 and U-18 women's teams: Coaches' perspective on the long-term. *Motriz. Revista de Educacao Fisica*, *24*(4), 1–9. https://doi.org/10.1590/S1980-6574201800040001

Microsoft Excel Spreadsheet Software | Microsoft 365. (2022).https://www.microsoft.com/en-us/microsoft-365/excel

Munro, R., & Mirsharif, Q. (n.d.). *The Essential Guide to Training Data*. Figure Eight.

O Que É O MongoDB? | MongoDB. (2022). https://www.mongodb.com/pt-br/what-is-mongodb

Official USHA Handball Rulebook with Interpretations, One-Wall and Three-Wall Addendums. (2015). United States Handball Association.

Oytun, M., Tinazci, C., Sekeroglu, B., Acikada, C., & Yavuz, H. U. (2020). Performance Prediction and Evaluation in Female Handball Players Using Machine Learning Models. *IEEE Access*, *8*, 116321–116335. https://doi.org/10.1109/ACCESS.2020.3004182

Ozdemir, S. (2016). *Principles of Data Science* (S. Vernekar, S. Gonsalves, & A. A. Tendulkar (Eds.)). Packt Publishing Ltd.

Patel, H., & Thakur, G. S. (2019). An Improved Fuzzy K-Nearest Neighbor Algorithm for Imbalanced Data using Adaptive Approach. *IETE Journal of Research*, *65*(6), 780–789. https://doi.org/10.1080/03772063.2018.1462109

Póvoas, S. C. A., Seabra, A. F. T., Ascensão, A. A. M. R., Magalhães, J., Soares, J. M. C., & Rebelo, A. N. C. (2012). Physical and Physiological Demands of Elite Team Handball. *Journal OfStrength and Conditioning Research*, *26*(12), 3365–3375.

Powers, D. M. W. (n.d.). *What the F-measure doesn't measure Features, Flaws, Fallacies and Fixes*.

Powers, D. M. W. (2007). *Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation* (Issue January 2008).

Poznan, G. B., & Poznan, M. W. (2019). Automation of the Process of Reporting the Compliance of the Production Plan with Its Execution Based on Integration of SAP ERP System In Connection With Excel Spreadsheet and VBA Application. *Digitalization of Supply Chains*, 101–116. https://doi.org/10.17270/b.m.978-83-66017-86-3

Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, *17*(3), 228–235. https://www.researchgate.net/publication/318816158

Rauschmayer, D. A. (2022). *JavaScript For Impatient Programmers* (ECMAScript). Fran Caye.

Romero, F. P., & Angulo, E. (2020). *Finding outstanding performance in handball players based on statistical analysis*.

Romero, F. P., Angulo, E., Serrano-Guerrero, J., & Olivas, J. A. (2020). A fuzzy framework to evaluate players' performance in handball. *International Journal of Computational Intelligence Systems*, *13*(1), 549–558. https://doi.org/10.2991/ijcis.d.200416.001

Schwenkreis, F. (2020). Why the Concept of Shopping Baskets helps to analyze Team-Handball. *2020 International Conference on Intelligent Data Science Technologies and Applications*, 4–10. https://doi.org/10.1109/IDSTA50958.2020.9264068

Skarbalius, A., Pukėnas, K., & Vidūnaitė, G. (2013). Sport Performance Profile in Men ' S European Modern Handball : Discriminant Analysis. *Education. Physical Training. Sport*, *3*(3), 44–54.

sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.1.3 documentation. (2022). https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

sklearn.svm.SVC — scikit-learn 1.1.3 documentation. (2022). https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Soto Valero, C. (2016). Predicting win-loss outcomes in MLB regular season games - A comparative study using data mining methods. *International Journal of Computer Science in Sport*, *15*(2), 91–112. https://doi.org/10.1515/ijcss-2016-0007

Sri Harsha Vardhan Goud, P., Mohana Roopa, Y., & Padmaja, B. (2019). Player performance analysis in sports: With fusion of machine learning and wearable technology. *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, *Iccmc*, 600–603. https://doi.org/10.1109/ICCMC.2019.8819815

Srinath, K. R. (2017). Python -The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology*, *4*(12), 354–357. www.irjet.net

Swamynathan, M. (2019). *Mastering Machine Learning with Python in Six Steps* (C. S. John, J. Markham, A. Mirashi, & Cover (Eds.); Second). APRESS.

T.Bray. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. *ECMA International*, *1*(October). http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf%5Cnhttps://www.rfc-editor.org/info/rfc7158

The Jupyter Notebook — Jupyter Notebook 7.0.0a7 documentation. (2022). https://jupyter-notebook.readthedocs.io/en/latest/notebook.html

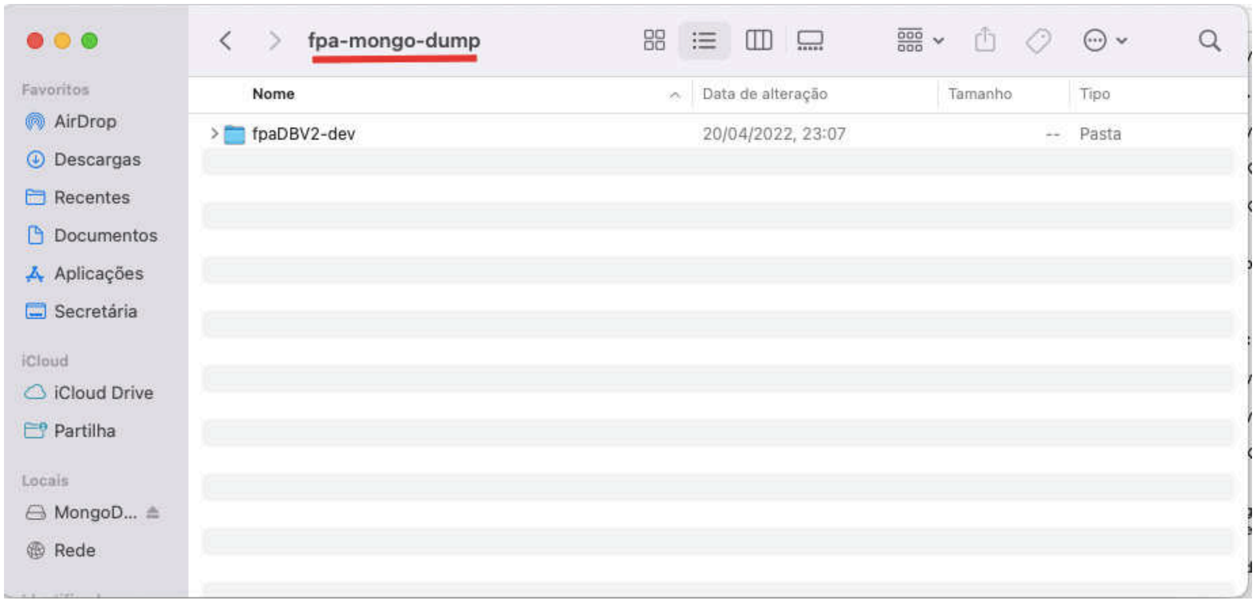
Tsoumakas, G., & Katakis, I. (2009). Multi-Label Classification : An Overview. *Journal of Data Warehousing and Mining*. https://doi.org/10.4018/jdwm.2007070101

Wagner, H., Finkenzeller, T., Würth, S., & Von Duvillard, S. P. (2014). Individual and Team Performance in Team-Handball: A Review. *Journal of Sports Science and Medicine*, *13*(4), 808–816.

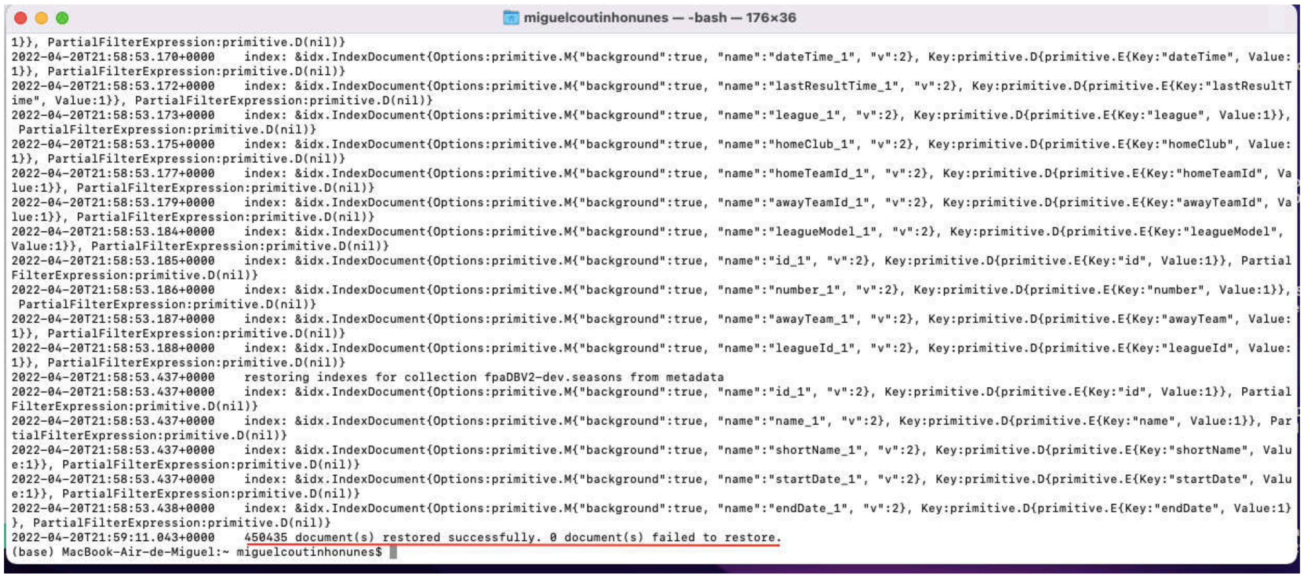
Wang, L., Chu, F., & Xie, W. (2007). Accurate Cancer Classification Using Expressions of Very Few Genes. *IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS*, *4*(1), 40–53.

What is MongoDB Compass? — MongoDB Compass. (2022). https://www.mongodb.com/docs/compass/current/

APPENDIXES

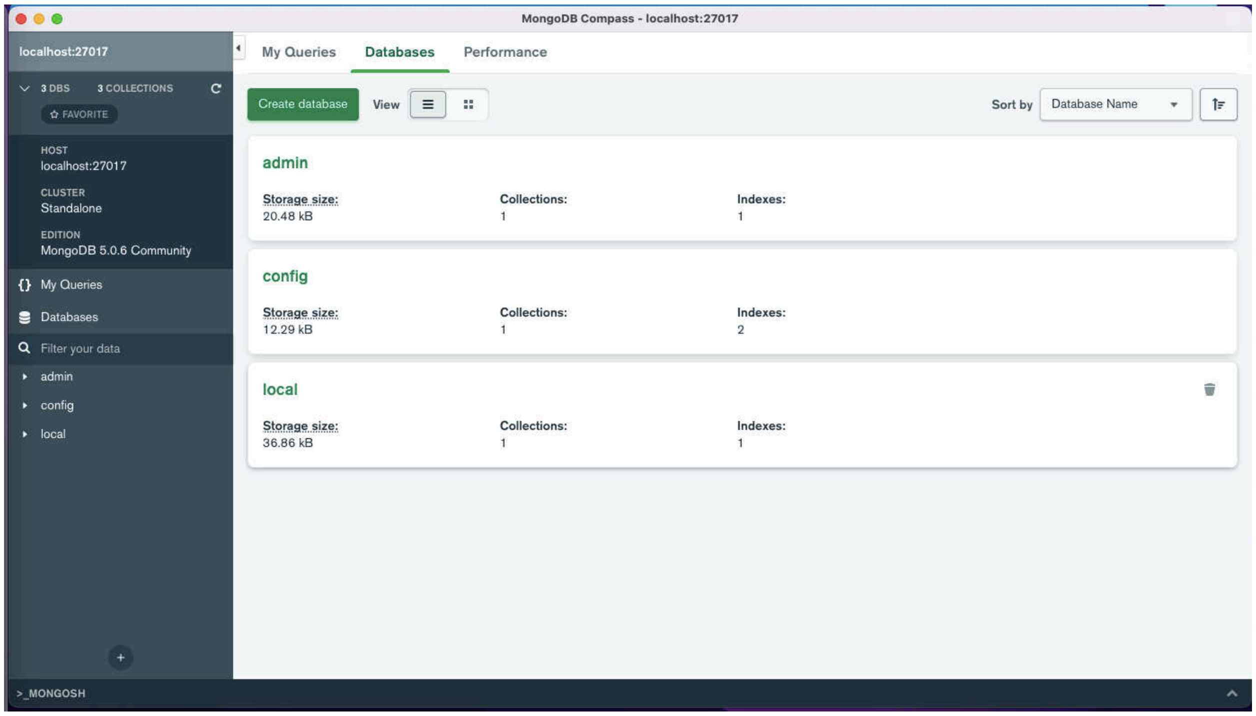


Appendix 1 - 'fpa-mongo-dump' folder



```
1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.170+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"dateTime_1", "v":2}, Key:primitive.D{primitive.E{Key:"dateTime", Value:
1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.172+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"lastResultTime_1", "v":2}, Key:primitive.D{primitive.E{Key:"lastResultT
ime", Value:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.173+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"league_1", "v":2}, Key:primitive.D{primitive.E{Key:"league", Value:1}},
 PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.175+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"homeClub_1", "v":2}, Key:primitive.D{primitive.E{Key:"homeClub", Value:
1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.177+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"homeTeamId_1", "v":2}, Key:primitive.D{primitive.E{Key:"homeTeamId", Va
lue:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.179+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"awayTeamId_1", "v":2}, Key:primitive.D{primitive.E{Key:"awayTeamId", Va
lue:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.184+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"leagueModel_1", "v":2}, Key:primitive.D{primitive.E{Key:"leagueModel",
Value:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.185+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"id_1", "v":2}, Key:primitive.D{primitive.E{Key:"id", Value:1}}, Partial
FilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.186+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"number_1", "v":2}, Key:primitive.D{primitive.E{Key:"number", Value:1}},
 PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.187+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"awayTeam_1", "v":2}, Key:primitive.D{primitive.E{Key:"awayTeam", Value:
1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.188+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"leagueId_1", "v":2}, Key:primitive.D{primitive.E{Key:"leagueId", Value:
1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.437+0000    restoring indexes for collection fpaDBV2-dev.seasons from metadata
2022-04-20T21:58:53.437+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"id_1", "v":2}, Key:primitive.D{primitive.E{Key:"id", Value:1}}, Partial
FilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.437+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"name_1", "v":2}, Key:primitive.D{primitive.E{Key:"name", Value:1}}, Par
tialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.437+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"shortName_1", "v":2}, Key:primitive.D{primitive.E{Key:"shortName", Valu
e:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.437+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"startDate_1", "v":2}, Key:primitive.D{primitive.E{Key:"startDate", Valu
e:1}}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:58:53.438+0000    index: &idx.IndexDocument{Options:primitive.M{"background":true, "name":"endDate_1", "v":2}, Key:primitive.D{primitive.E{Key:"endDate", Value:1}
}, PartialFilterExpression:primitive.D(nil)}
2022-04-20T21:59:11.043+0000    450435 document(s) restored successfully. 0 document(s) failed to restore.
(base) MacBook-Air-de-Miguel:~ miguelcoutinhonunes$ 
```

Appendix 2 - Terminal code with documents restored


MongoDB Compass - localhost:27017

localhost:27017

3 DBS 3 COLLECTIONS

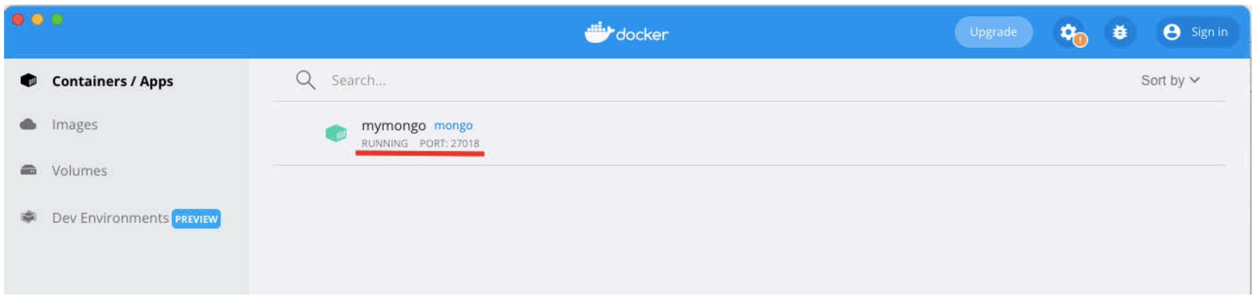
HOST
localhost:27017

CLUSTER
Standalone

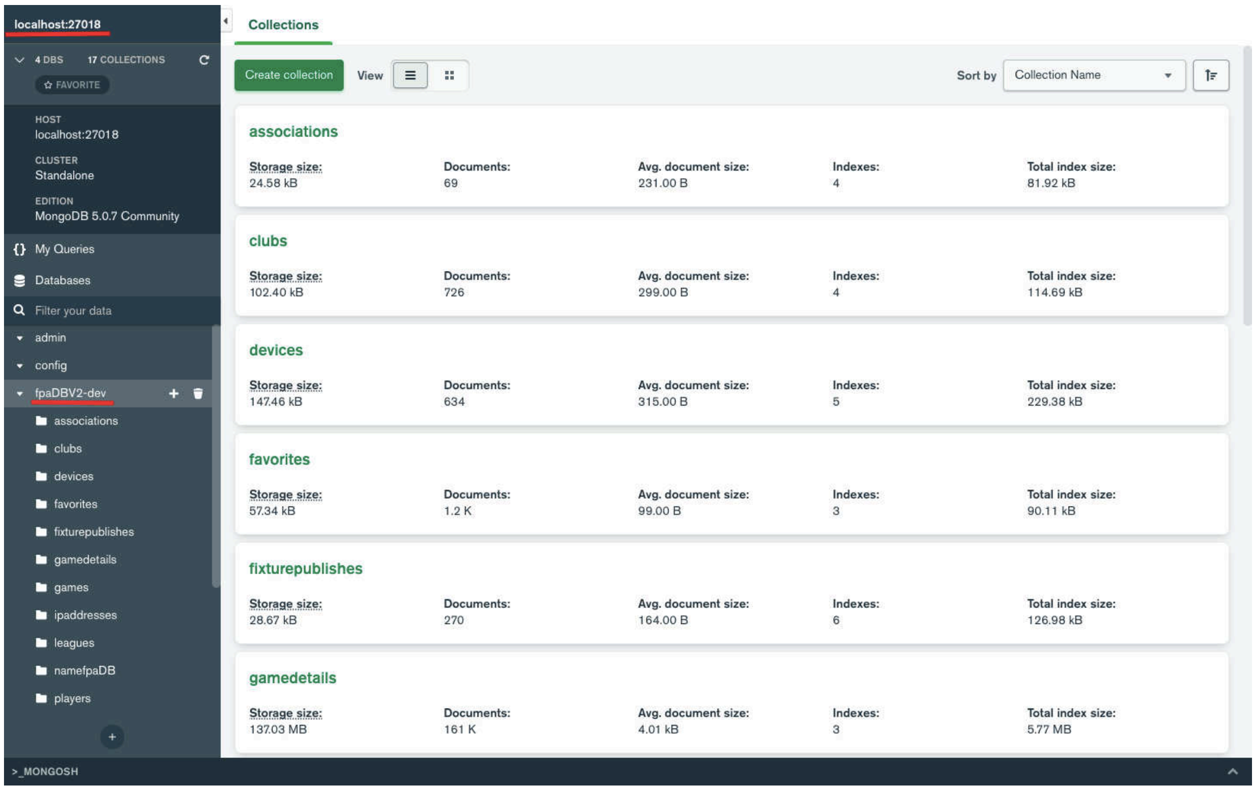
EDITION
MongoDB 5.0.6 Community

admin	Storage size: 20.48 kB	Collections: 1	Indexes: 1
config	Storage size: 12.29 kB	Collections: 1	Indexes: 2
local	Storage size: 36.86 kB	Collections: 1	Indexes: 1

Appendix 3 - MongoDB Compass with no content


mymongo mongo
RUNNING PORT: 27018

Appendix 4 - Container 'mymongo' localhost 27018



associations
Storage size: 24.58 kB | Documents: 69 | Avg. document size: 231.00 B | Indexes: 4 | Total index size: 81.92 kB

clubs
Storage size: 102.40 kB | Documents: 726 | Avg. document size: 299.00 B | Indexes: 4 | Total index size: 114.69 kB

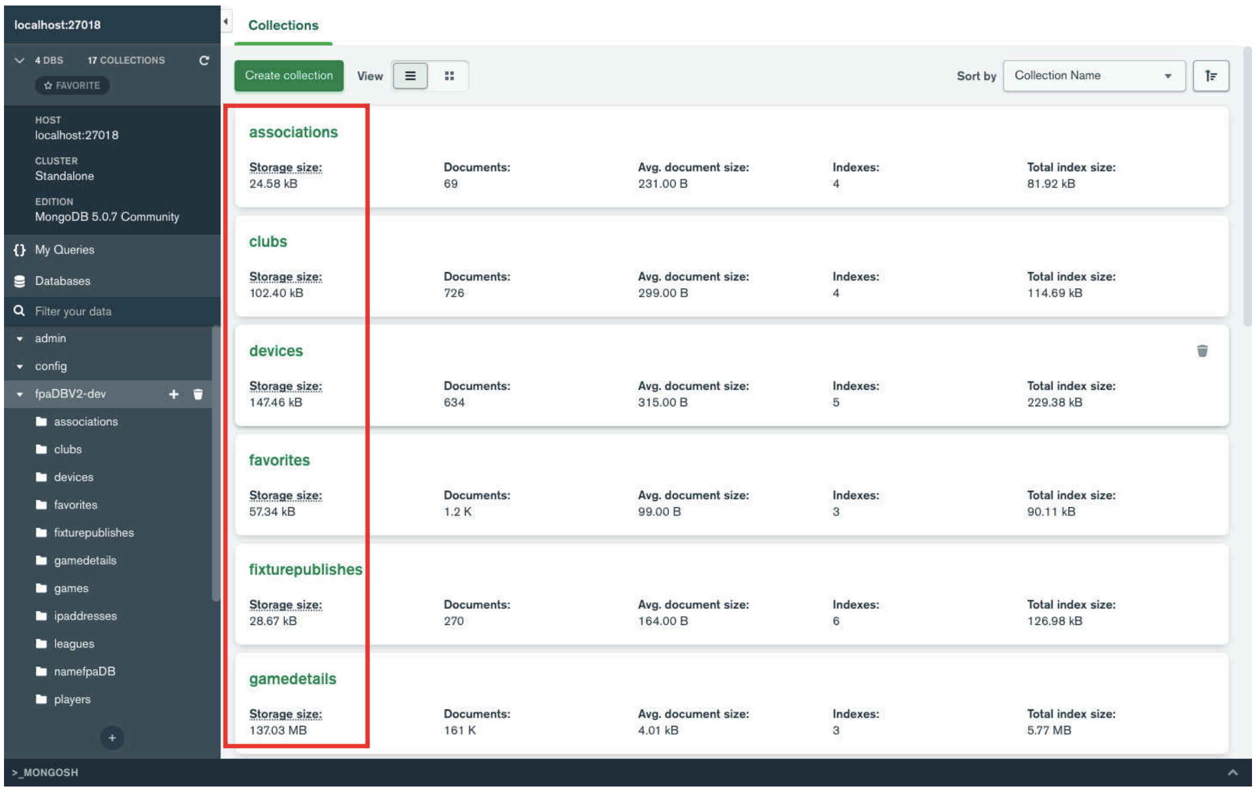
devices
Storage size: 147.46 kB | Documents: 634 | Avg. document size: 315.00 B | Indexes: 5 | Total index size: 229.38 kB

favorites
Storage size: 57.34 kB | Documents: 1.2 K | Avg. document size: 99.00 B | Indexes: 3 | Total index size: 90.11 kB

fixturepublishes
Storage size: 28.67 kB | Documents: 270 | Avg. document size: 164.00 B | Indexes: 6 | Total index size: 126.98 kB

gamedetails
Storage size: 137.03 MB | Documents: 161 K | Avg. document size: 4.01 kB | Indexes: 3 | Total index size: 5.77 MB

Appendix 5 - MongoDB Compass content with localhost 27018



associations
Storage size: 24.58 kB | Documents: 69 | Avg. document size: 231.00 B | Indexes: 4 | Total index size: 81.92 kB

clubs
Storage size: 102.40 kB | Documents: 726 | Avg. document size: 299.00 B | Indexes: 4 | Total index size: 114.69 kB

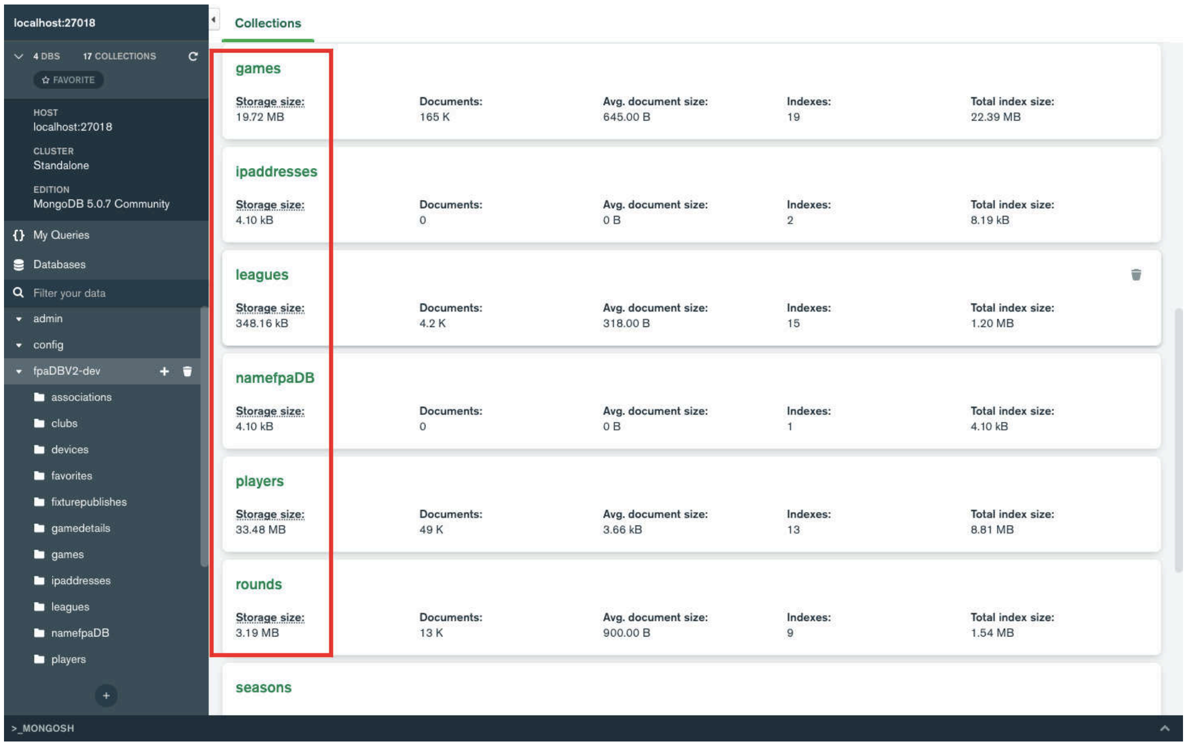
devices
Storage size: 147.46 kB | Documents: 634 | Avg. document size: 315.00 B | Indexes: 5 | Total index size: 229.38 kB

favorites
Storage size: 57.34 kB | Documents: 1.2 K | Avg. document size: 99.00 B | Indexes: 3 | Total index size: 90.11 kB

fixturepublishes
Storage size: 28.67 kB | Documents: 270 | Avg. document size: 164.00 B | Indexes: 6 | Total index size: 126.98 kB

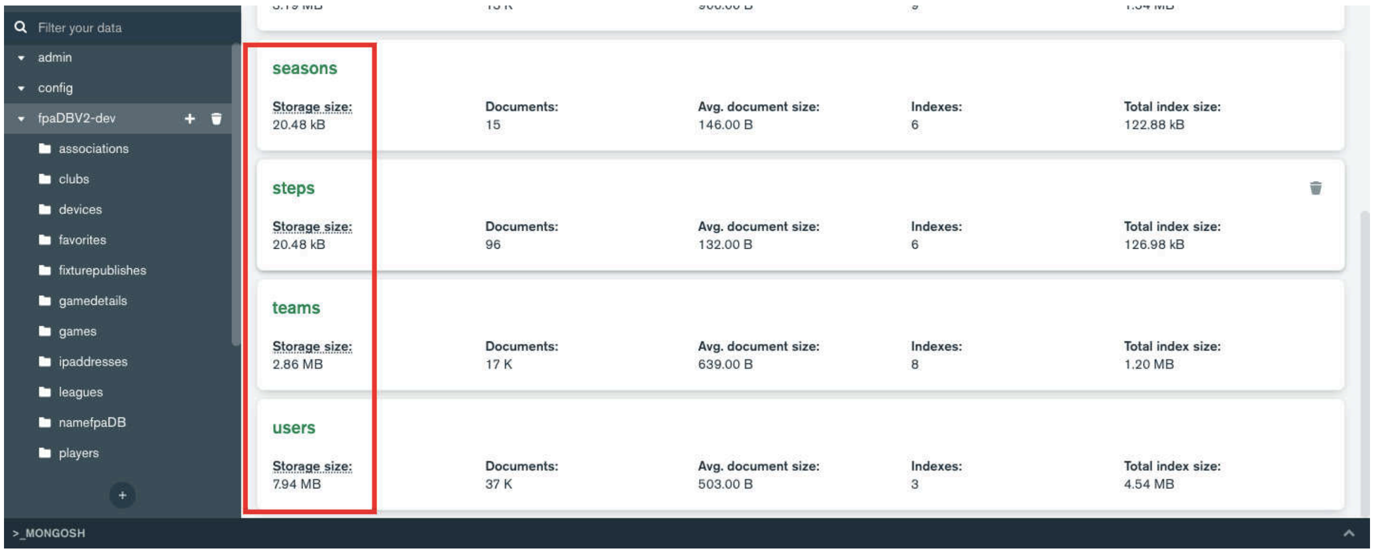
gamedetails
Storage size: 137.03 MB | Documents: 161 K | Avg. document size: 4.01 kB | Indexes: 3 | Total index size: 5.77 MB

Appendix 6 - Collections of 'fpaDBV2-dev' database



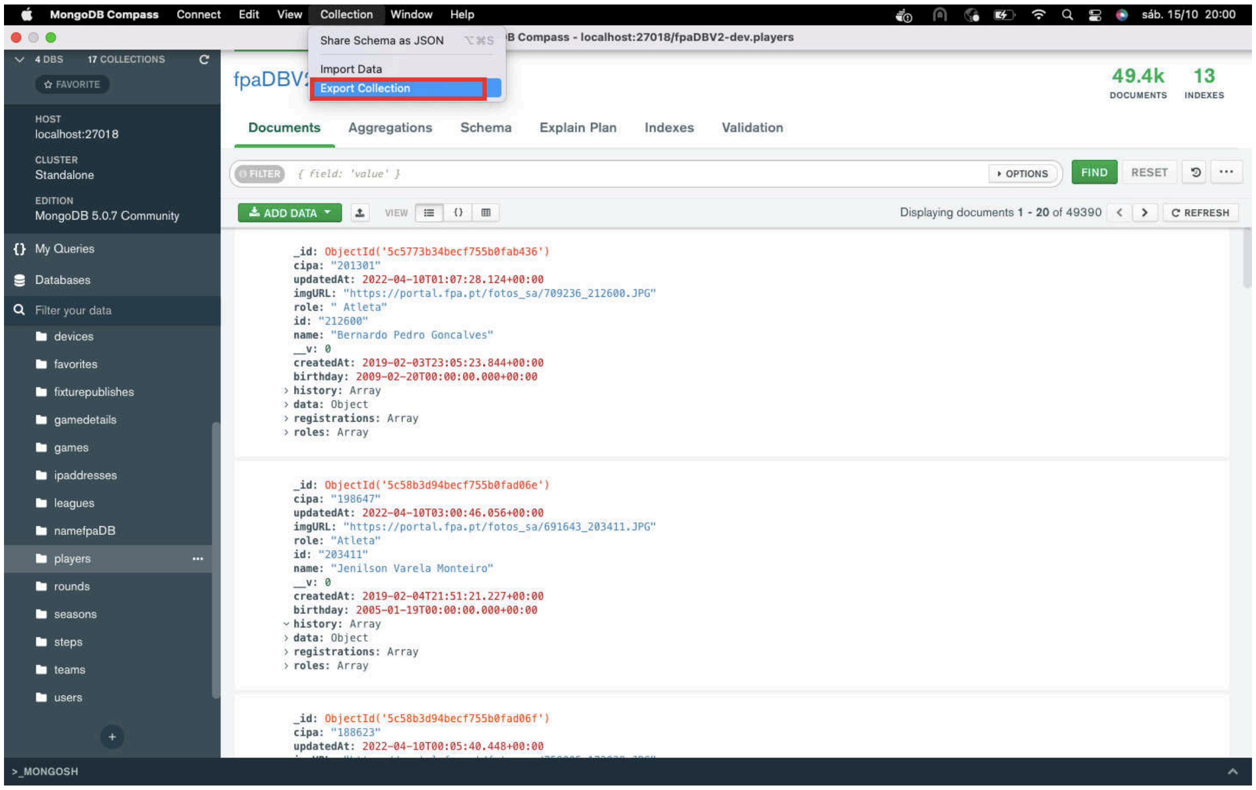
	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
games	19.72 MB	165 K	645.00 B	19	22.39 MB
ipaddresses	4.10 kB	0	0 B	2	8.19 kB
leagues	348.16 kB	4.2 K	318.00 B	15	1.20 MB
namefpaDB	4.10 kB	0	0 B	1	4.10 kB
players	33.48 MB	49 K	3.66 kB	13	8.81 MB
rounds	3.19 MB	13 K	900.00 B	9	1.54 MB
seasons					

Appendix 7 - Collections of 'fpaDBV2-dev' database



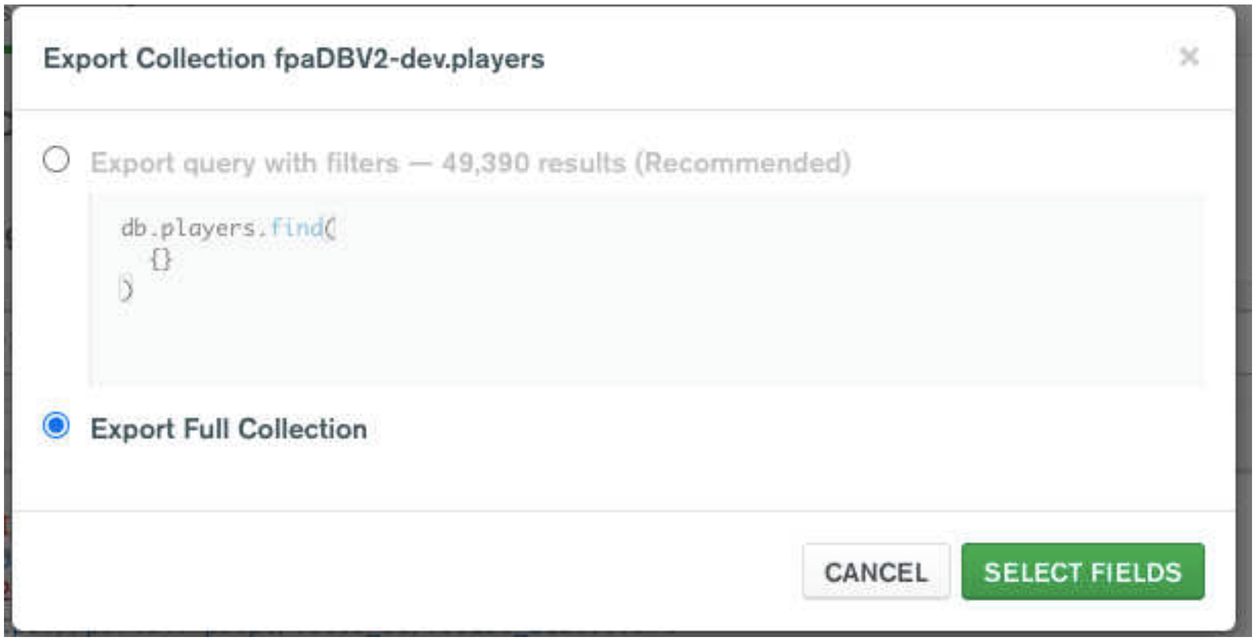
	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
seasons	20.48 kB	15	146.00 B	6	122.88 kB
steps	20.48 kB	96	132.00 B	6	126.98 kB
teams	2.86 MB	17 K	639.00 B	8	1.20 MB
users	7.94 MB	37 K	503.00 B	3	4.54 MB

Appendix 8 - Collections of 'fpaDBV2-dev' database



```
_id: ObjectId('5c5773b34becf755b0fab436')
cipa: "201301"
updatedAt: 2022-04-10T01:07:28.124+00:00
imgURL: "https://portal.fpa.pt/fotos_sa/709236_212600.JPG"
role: " Atleta"
id: "212600"
name: "Bernardo Pedro Goncalves"
__v: 0
createdAt: 2019-02-03T23:05:23.844+00:00
birthday: 2009-02-20T00:00:00.000+00:00
> history: Array
> data: Object
> registrations: Array
> roles: Array

_id: ObjectId('5c58b3d94becf755b0fad06e')
cipa: "198647"
updatedAt: 2022-04-10T03:00:46.056+00:00
imgURL: "https://portal.fpa.pt/fotos_sa/691643_203411.JPG"
role: "Atleta"
id: "203411"
name: "Jenilson Varela Monteiro"
__v: 0
createdAt: 2019-02-04T21:51:21.227+00:00
birthday: 2005-01-19T00:00:00.000+00:00
v history: Array
> data: Object
> registrations: Array
> roles: Array

_id: ObjectId('5c58b3d94becf755b0fad06f')
cipa: "188623"
updatedAt: 2022-04-10T00:05:40.448+00:00
```

Appendix 9 - Select 'Export Collection'



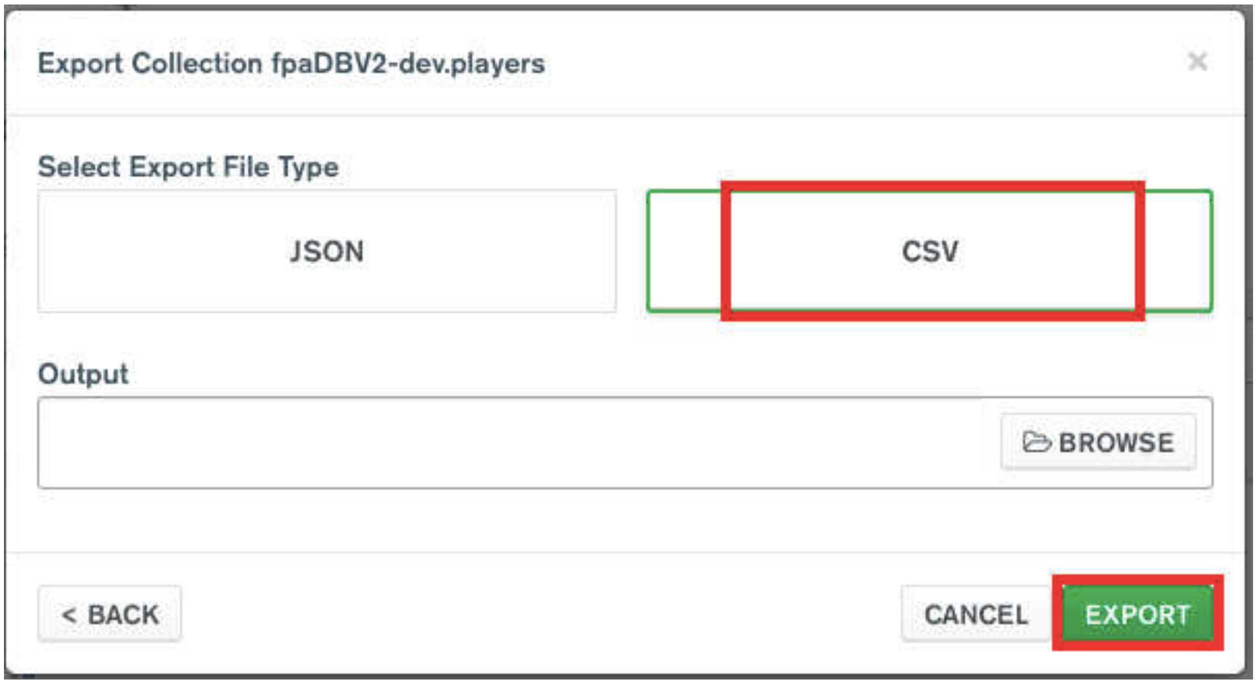
Export Collection fpaDBV2-dev.players

Export query with filters — 49,390 results (Recommended)

```
db.players.find(
  {}
)
```

[✓ Export Full Collection]

Appendix 10 - Choose 'Export Full Collection'



Appendix 11 - Select 'CSV' export file type and 'Export'

Table ‘StepVlookup’:

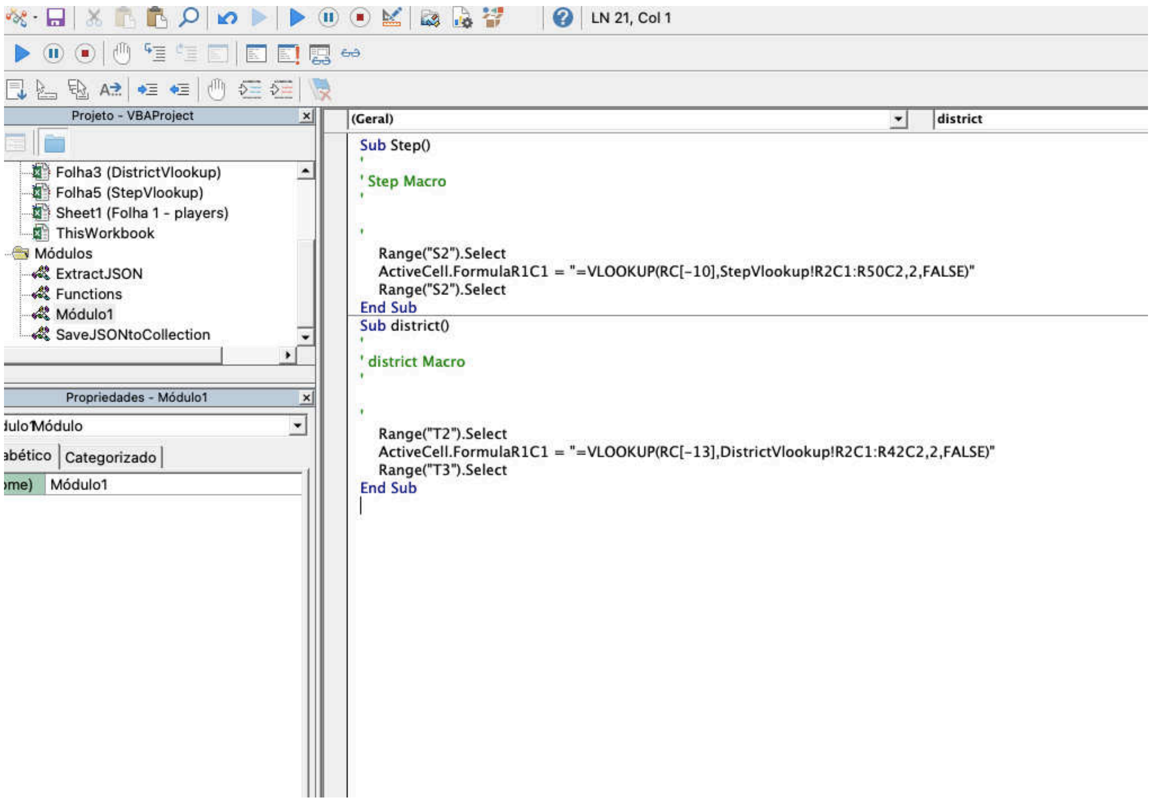
stepDesc	**Step**
SUB-20 M	Seniores
SUB-19 M	Sub-18
Juniores M	Sub-18
Sub 18 M - And Praia	Sub-18 Praia
Juvenis M	Sub-16
Iniciados M	Sub-14
Rookies Praia Masc	Rookies Praia
Infantis M	Minis
Minis M	Minis
Bambis M	Bambis
Seniores M	Seniores
Master Praia Masc	Master Praia
Seniores M - And. Praia	Seniores Praia
null	N/A
Veteranos M	Veteranos
Sub 15 M - And Praia	Sub-14 Praia
SUB-18 M	Sub-18
Sub 17 M - And Praia	Sub-16 Praia
SUB-17 M	Sub-16
Seniores F	Seniores
Master Praia Fem	Master Praia
Juniores F	Sub-18
Juvenis F	Sub-16
Rookies Praia Fem	Rookies Praia
Iniciados F	Sub-14
Infantis F	Minis
Minis F	Minis
Veteranos F	Veteranos
Bambis F	Bambis
SUB-20 F	Seniores
Sub 17 F - And Praia	Sub-16 Praia
SUB-18 F	Sub-18
Sub 15 F - And Praia	Sub-14 Praia
Sub 18 F - And Praia	Sub-18 Praia
SUB-17 F	Sub-16
SUB-16 F	Sub-16
SUB-12 F	Minis
SUB-15 F	Sub-14
SUB-14 F	Sub-14
SUB-13 F	Minis
Seniores F - And. Praia	Seniores Praia
SUB-16 M	Sub-16
SUB-15 M	Sub-14
Sub 14 M - And Praia	Sub-14 Praia
Sub 14 F - And Praia	Sub-14 Praia
SUB-14 M	Sub-14
SUB-13 M	Minis
Manitas M	Manitas
Manitas F	Manitas

Table 4 - ‘StepVlookup’

Table ‘DistrictVlookup’:

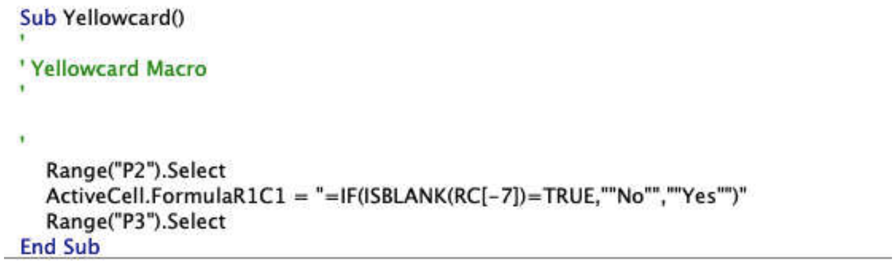
associationDesc	**District**
A.A. Leiria	Leiria
Leiria A Praia	Leiria
F.A.P.	N/A
A.A. Aveiro	Aveiro
Aveiro A Praia	Aveiro
A.A. Viseu	Viseu
A.A. Setubal	Setúbal
A.A. Ilha Faial	Açores
Porto A Praia	Porto
A.A. Porto	Porto
Andebol de Praia	N/A
A.A. Braga	Braga
L.P.A.	N/A
Lisboa A Praia	Lisboa
A.A. Lisboa	Lisboa
A.A. Algarve	Faro
Setubal A Praia	Setúbal
A.A. Santarem	Santarém
A.A. Madeira	Madeira
A.A. Vila Real	Vila Real
A.A. Guarda	Guarda
A.A. Ilha S. Maria	Açores
A.A. Castelo Branco	Castelo Branco
Madeira A Praia	Madeira
A.A. Coimbra	Coimbra
Viseu A Praia	Viseu
A.A. Evora	Évora
A.A. V. Castelo	Viana do Castelo
Braga A Praia	Braga
A.A. Beja	Beja
A.A. Ilha S. Miguel	Açores
A.A. Portalegre	Portalegre
Santarém A Praia	Santarém
Leiria - A 4ALL	Leiria
A.A. Ilha Terceira	Açores
A.A. Ilha Graciosa	Açores
Setúbal - A 4ALL	Setúbal
A.A. Braganca	Bragança
Aveiro - A 4ALL	Aveiro
Porto - A 4LL	Porto
UAAA	N/A

Table 5 – ‘DistrictVlookup’



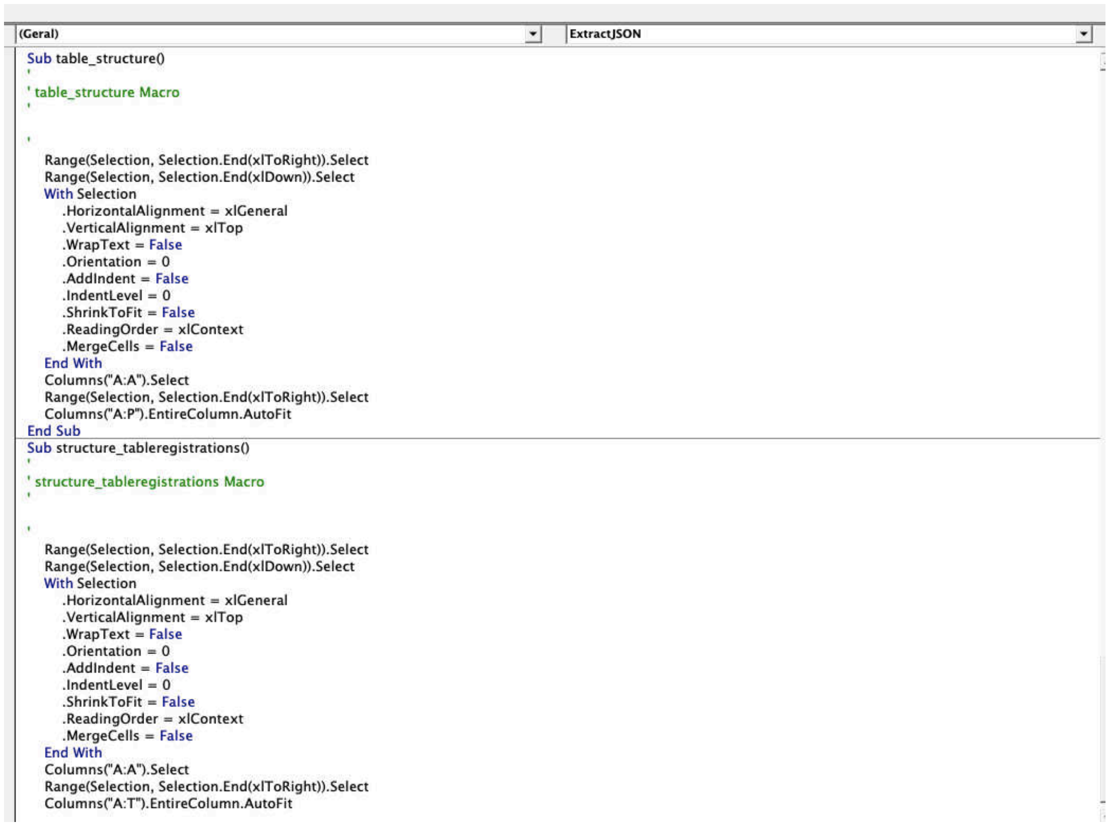
```
Sub Step()
'
' Step Macro
'

'
    Range("S2").Select
    ActiveCell.FormulaR1C1 = "=VLOOKUP(RC[-10],StepVlookup!R2C1:R50C2,2,FALSE)"
    Range("S2").Select
End Sub
Sub district()
'
' district Macro
'

'
    Range("T2").Select
    ActiveCell.FormulaR1C1 = "=VLOOKUP(RC[-13],DistrictVlookup!R2C1:R42C2,2,FALSE)"
    Range("T3").Select
End Sub
```

Appendix 12 - Vlookup macros for 'Step' and 'District' columns making



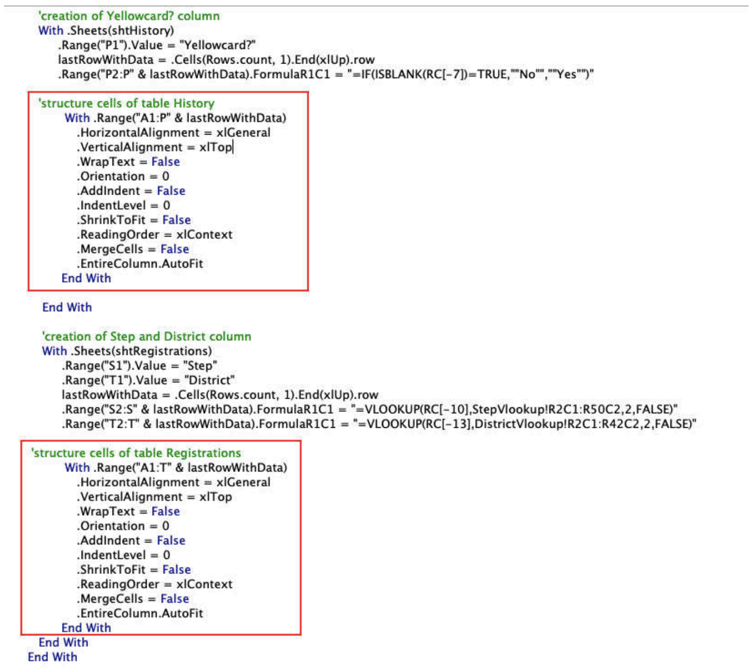
```
Sub Yellowcard()
'
' Yellowcard Macro
'

'
    Range("P2").Select
    ActiveCell.FormulaR1C1 = "=IF(ISBLANK(RC[-7])=TRUE,""No"",""Yes"")"
    Range("P3").Select
End Sub
```

Appendix 13 - Macro for 'Yellowcard?' column making



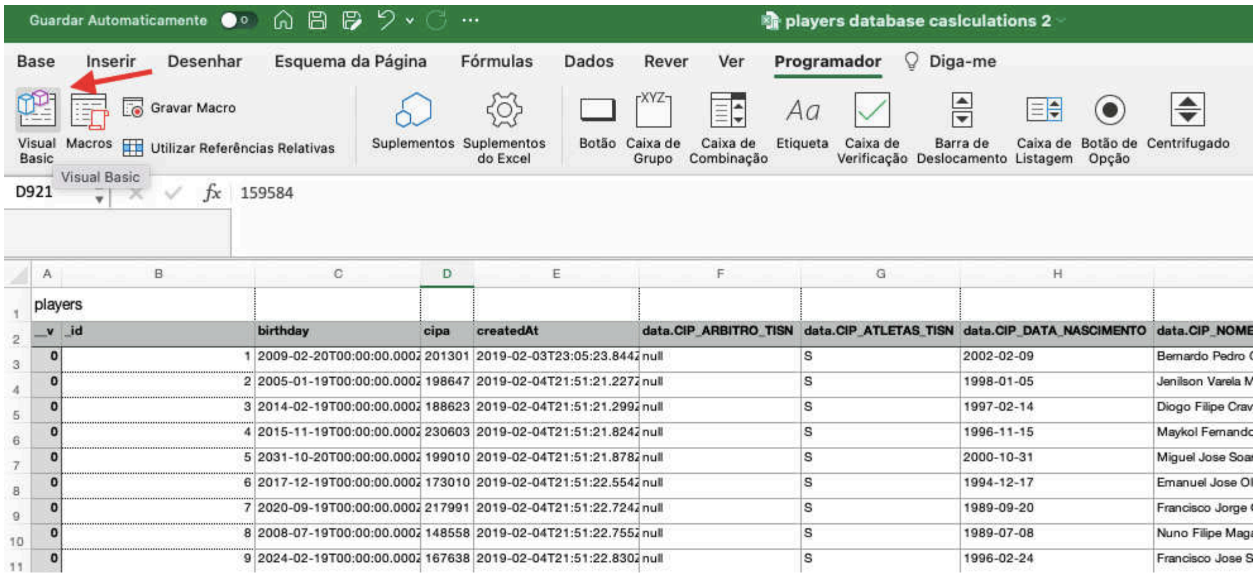
```
Sub table_structure()
'
' table_structure Macro
'

'
    Range(Selection, Selection.End(xlToRight)).Select
    Range(Selection, Selection.End(xlDown)).Select
    With Selection
        .HorizontalAlignment = xlGeneral
        .VerticalAlignment = xlTop
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Columns("A:A").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Columns("A:P").EntireColumn.AutoFit
End Sub
Sub structure_tableregistrations()
'
' structure_tableregistrations Macro
'

'
    Range(Selection, Selection.End(xlToRight)).Select
    Range(Selection, Selection.End(xlDown)).Select
    With Selection
        .HorizontalAlignment = xlGeneral
        .VerticalAlignment = xlTop
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Columns("A:A").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Columns("A:T").EntireColumn.AutoFit
```

Appendix 14 - Macros structuring 'Folha Registrations' and 'Folha History'



```
  'creation of Yellowcard? column
  With .Sheets(shtHistory)
      .Range("P1").Value = "Yellowcard?"
      lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
      .Range("P2:P" & lastRowWithData).FormulaR1C1 = "=IF(ISBLANK(RC[-7])=TRUE,""No"",""Yes"")"

  'structure cells of table History
        With .Range("A1:P" & lastRowWithData)
           .HorizontalAlignment = xlGeneral
           .VerticalAlignment = xlTop
           .WrapText = False
           .Orientation = 0
           .AddIndent = False
           .IndentLevel = 0
           .ShrinkToFit = False
           .ReadingOrder = xlContext
           .MergeCells = False
           .EntireColumn.AutoFit
       End With

   End With

   'creation of Step and District column
   With .Sheets(shtRegistrations)
       .Range("S1").Value = "Step"
       .Range("T1").Value = "District"
       lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
       .Range("S2:S" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-10],StepVlookup!R2C1:R50C2,2,FALSE)"
       .Range("T2:T" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-13],DistrictVlookup!R2C1:R42C2,2,FALSE)"

  'structure cells of table Registrations
        With .Range("A1:T" & lastRowWithData)
           .HorizontalAlignment = xlGeneral
           .VerticalAlignment = xlTop
           .WrapText = False
           .Orientation = 0
           .AddIndent = False
           .IndentLevel = 0
           .ShrinkToFit = False
           .ReadingOrder = xlContext
           .MergeCells = False
           .EntireColumn.AutoFit
       End With
  End With
End With
```

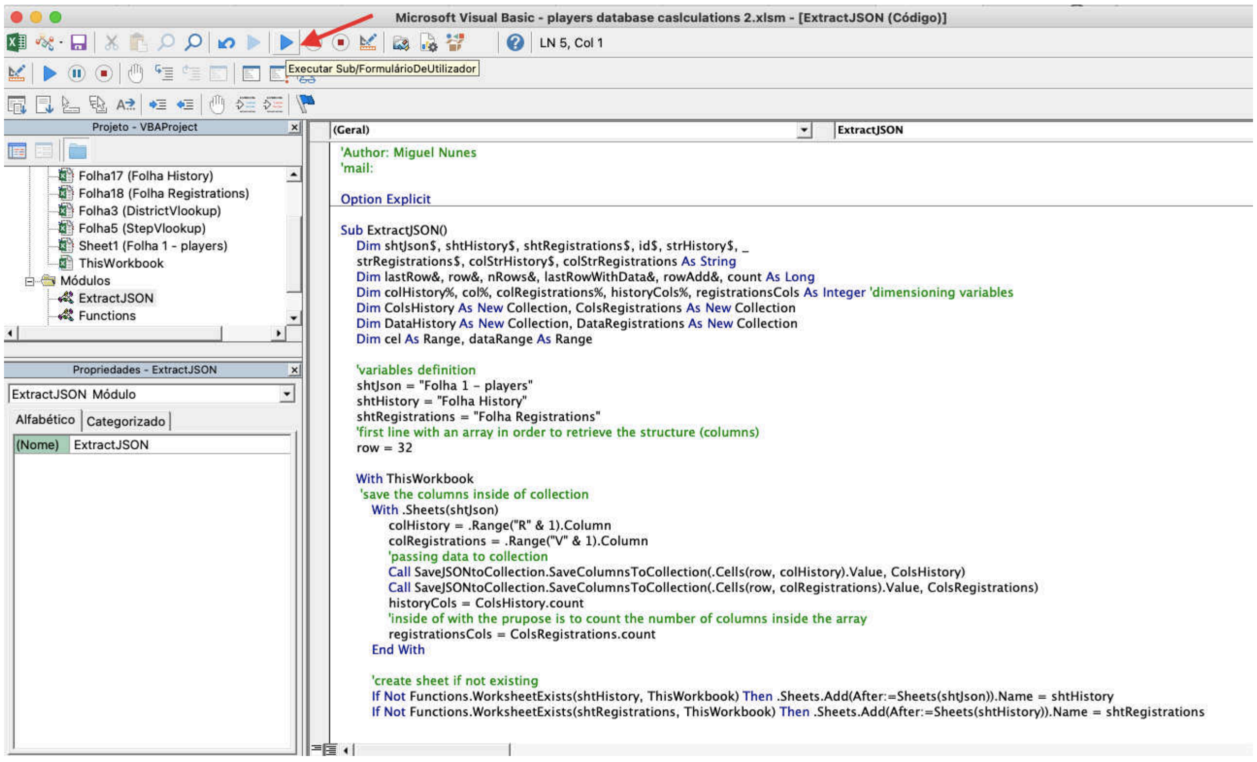
Appendix 15 - Structure macros added on 'ExtractJSON()' code



D921 159584

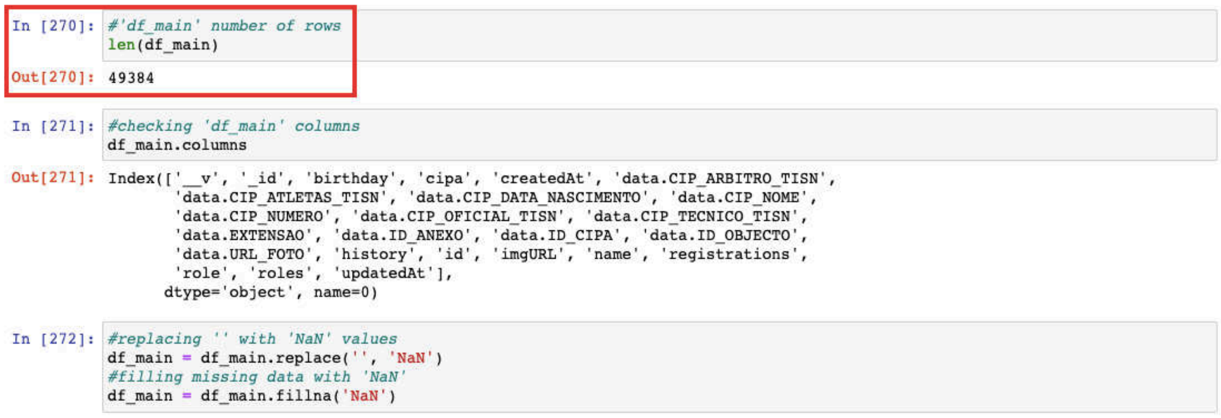
	A	B	C	D	E	F	G	H	
1	players								
2	_v	_id	birthday	cipa	createdAt	data.CIP_ARBITRO_TISN	data.CIP_ATLETAS_TISN	data.CIP_DATA_NASCIMENTO	data.CIP_NOME
3	0	1	2009-02-20T00:00:00.000Z	201301	2019-02-03T23:05:23.844Z	null	S	2002-02-09	Bernardo Pedro
4	0	2	2005-01-19T00:00:00.000Z	198647	2019-02-04T21:51:21.227Z	null	S	1998-01-05	Jenilson Varela M
5	0	3	2014-02-19T00:00:00.000Z	188623	2019-02-04T21:51:21.299Z	null	S	1997-02-14	Diogo Filipe Crav
6	0	4	2015-11-19T00:00:00.000Z	230603	2019-02-04T21:51:21.824Z	null	S	1996-11-15	Maykol Fernando
7	0	5	2031-10-20T00:00:00.000Z	199010	2019-02-04T21:51:21.878Z	null	S	2000-10-31	Miguel Jose Soa
8	0	6	2017-12-19T00:00:00.000Z	173010	2019-02-04T21:51:22.554Z	null	S	1994-12-17	Emanuel Jose Ol
9	0	7	2020-09-19T00:00:00.000Z	217991	2019-02-04T21:51:22.724Z	null	S	1989-09-20	Francisco Jorge
10	0	8	2008-07-19T00:00:00.000Z	148558	2019-02-04T21:51:22.755Z	null	S	1989-07-08	Nuno Filipe Mag
11	0	9	2024-02-19T00:00:00.000Z	167638	2019-02-04T21:51:22.830Z	null	S	1996-02-24	Francisco Jose S

Appendix 16 – 'Visual Basic' option on 'Programador'



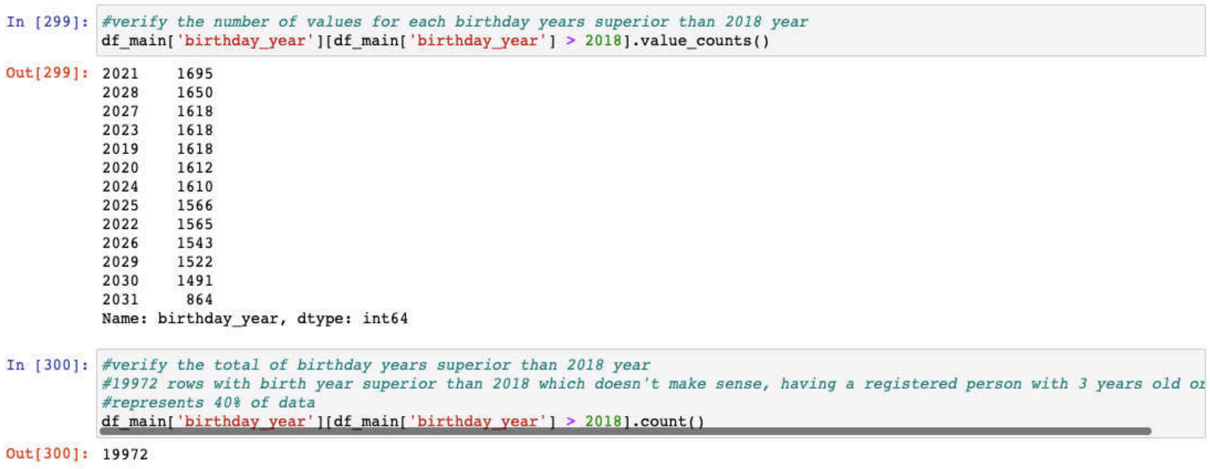
```
'Author: Miguel Nunes
'mail:

Option Explicit

Sub ExtractJSON()
    Dim shtJson$, shtHistory$, shtRegistrations$, id$, strHistory$, _
    strRegistrations$, colStrHistory$, colStrRegistrations As String
    Dim lastRow&, row&, nRows&, lastRowWithData&, rowAdd&, count As Long
    Dim colHistory%, col%, colRegistrations%, historyCols%, registrationsCols As Integer 'dimensioning variables
    Dim ColsHistory As New Collection, ColsRegistrations As New Collection
    Dim DataHistory As New Collection, DataRegistrations As New Collection
    Dim cel As Range, dataRange As Range

    'variables definition
    shtJson = "Folha 1 - players"
    shtHistory = "Folha History"
    shtRegistrations = "Folha Registrations"
    'first line with an array in order to retrieve the structure (columns)
    row = 32

    With ThisWorkbook
     'save the columns inside of collection
       With .Sheets(shtJson)
          colHistory = .Range("R" & 1).Column
          colRegistrations = .Range("V" & 1).Column
          'passing data to collection
          Call SaveJSONtoCollection.SaveColumnsToCollection(.Cells(row, colHistory).Value, ColsHistory)
          Call SaveJSONtoCollection.SaveColumnsToCollection(.Cells(row, colRegistrations).Value, ColsRegistrations)
          historyCols = ColsHistory.count
          'inside of with the prupose is to count the number of columns inside the array
          registrationsCols = ColsRegistrations.count
       End With

       'create sheet if not existing
       If Not Functions.WorksheetExists(shtHistory, ThisWorkbook) Then .Sheets.Add(After:=Sheets(shtJson)).Name = shtHistory
       If Not Functions.WorksheetExists(shtRegistrations, ThisWorkbook) Then .Sheets.Add(After:=Sheets(shtHistory)).Name = shtRegistrations
```

Appendix 17 – 'Play' icon to start VBA program



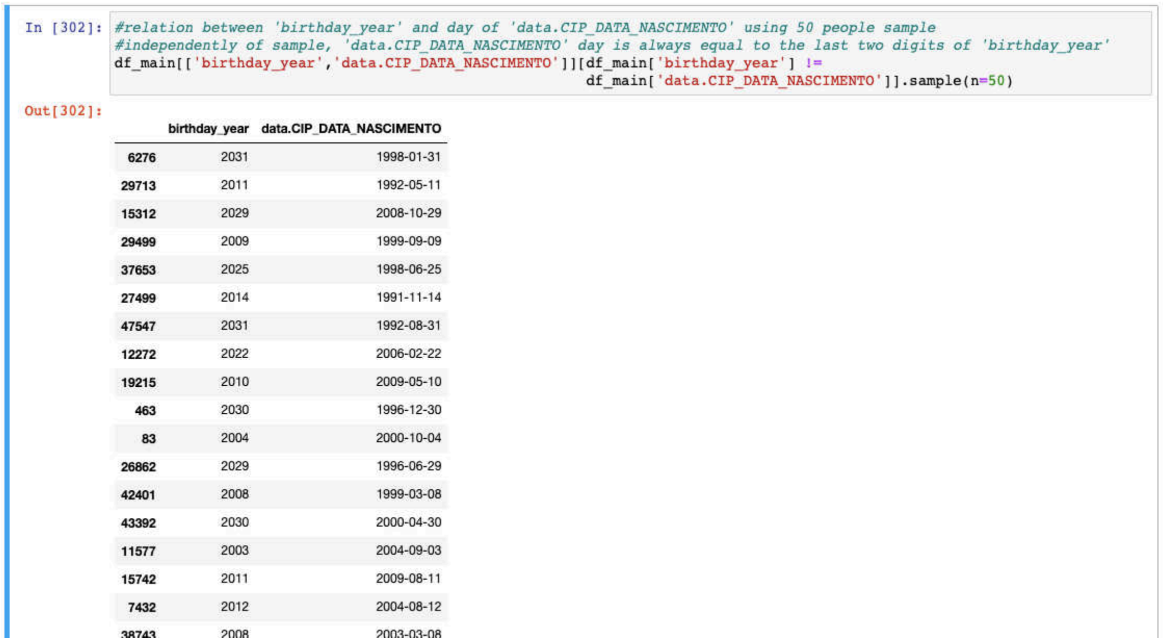
```
In [270]: #'df_main' number of rows
          len(df_main)

Out[270]: 49384

In [271]: #checking 'df_main' columns
          df_main.columns

Out[271]: Index(['__v', '_id', 'birthday', 'cipa', 'createdAt', 'data.CIP_ARBITRO_TISN',
                 'data.CIP_ATLETAS_TISN', 'data.CIP_DATA_NASCIMENTO', 'data.CIP_NOME',
                 'data.CIP_NUMERO', 'data.CIP_OFICIAL_TISN', 'data.CIP_TECNICO_TISN',
                 'data.EXTENSAO', 'data.ID_ANEXO', 'data.ID_CIPA', 'data.ID_OBJECTO',
                 'data.URL_FOTO', 'history', 'id', 'imgURL', 'name', 'registrations',
                 'role', 'roles', 'updatedAt'],
                dtype='object', name=0)

In [272]: #replacing '' with 'NaN' values
          df_main = df_main.replace('', 'NaN')
          #filling missing data with 'NaN'
          df_main = df_main.fillna('NaN')
```

Appendix 18 - Main table total rows



```
In [299]: #verify the number of values for each birthday years superior than 2018 year
          df_main['birthday_year'][df_main['birthday_year'] > 2018].value_counts()

Out[299]: 2021    1695
          2028    1650
          2027    1618
          2023    1618
          2019    1618
          2020    1612
          2024    1610
          2025    1566
          2022    1565
          2026    1543
          2029    1522
          2030    1491
          2031     864
          Name: birthday_year, dtype: int64

In [300]: #verify the total of birthday years superior than 2018 year
          #19972 rows with birth year superior than 2018 which doesn't make sense, having a registered person with 3 years old or
          #represents 40% of data
          df_main['birthday_year'][df_main['birthday_year'] > 2018].count()

Out[300]: 19972
```

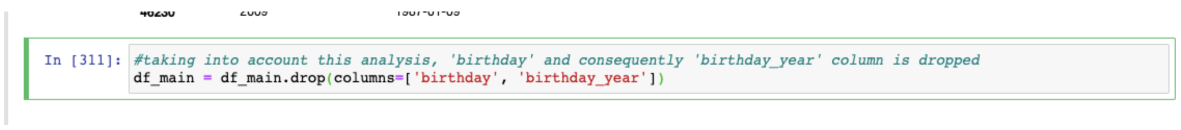
Appendix 19 – Birth year superior to 2018



```
In [302]: #relation between 'birthday_year' and day of 'data.CIP_DATA_NASCIMENTO' using 50 people sample
          #independently of sample, 'data.CIP_DATA_NASCIMENTO' day is always equal to the last two digits of 'birthday_year'
          df_main[['birthday_year','data.CIP_DATA_NASCIMENTO']][df_main['birthday_year'] !=
                                                                df_main['data.CIP_DATA_NASCIMENTO']].sample(n=50)

Out[302]:
```

	birthday_year	data.CIP_DATA_NASCIMENTO
6276	2031	1998-01-31
29713	2011	1992-05-11
15312	2029	2008-10-29
29499	2009	1999-09-09
37653	2025	1998-06-25
27499	2014	1991-11-14
47547	2031	1992-08-31
12272	2022	2006-02-22
19215	2010	2009-05-10
463	2030	1996-12-30
83	2004	2000-10-04
26862	2029	1996-06-29
42401	2008	1999-03-08
43392	2030	2000-04-30
11577	2003	2004-09-03
15742	2011	2009-08-11
7432	2012	2004-08-12
38743	2008	2003-03-08

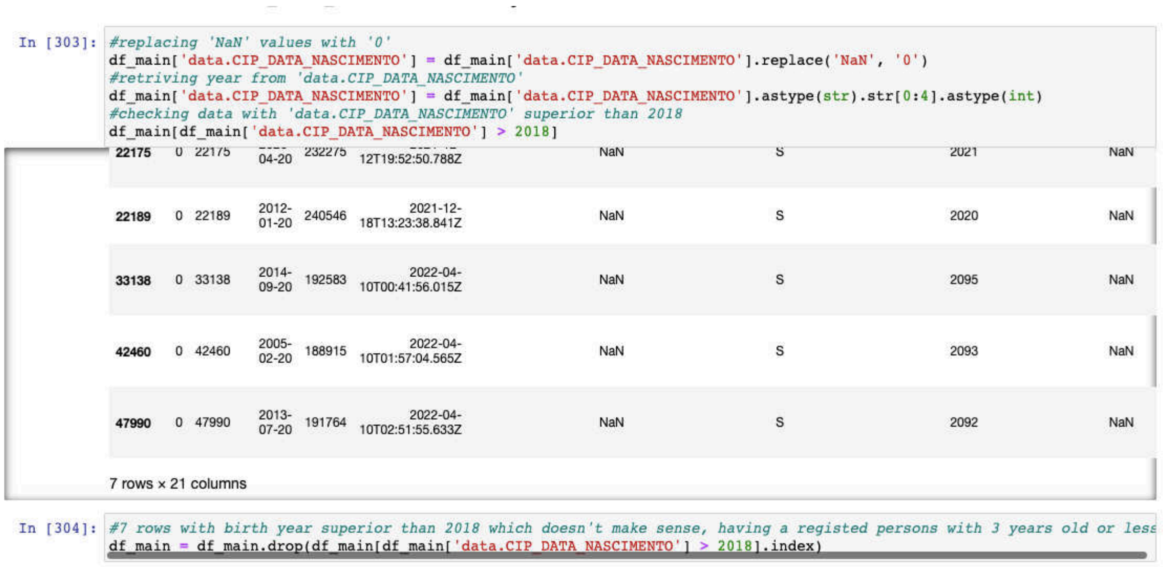
Appendix 20 – Relation of 'birthday_year' and 'data.CIP_DATA_NASCIMENTO' with 50 people sample



46230	2009	1987-01-09

```
In [311]: #taking into account this analysis, 'birthday' and consequently 'birthday_year' column is dropped
          df_main = df_main.drop(columns=['birthday', 'birthday_year'])
```

Appendix 21 – Drop of 'birthday' and 'birthday_year' colums



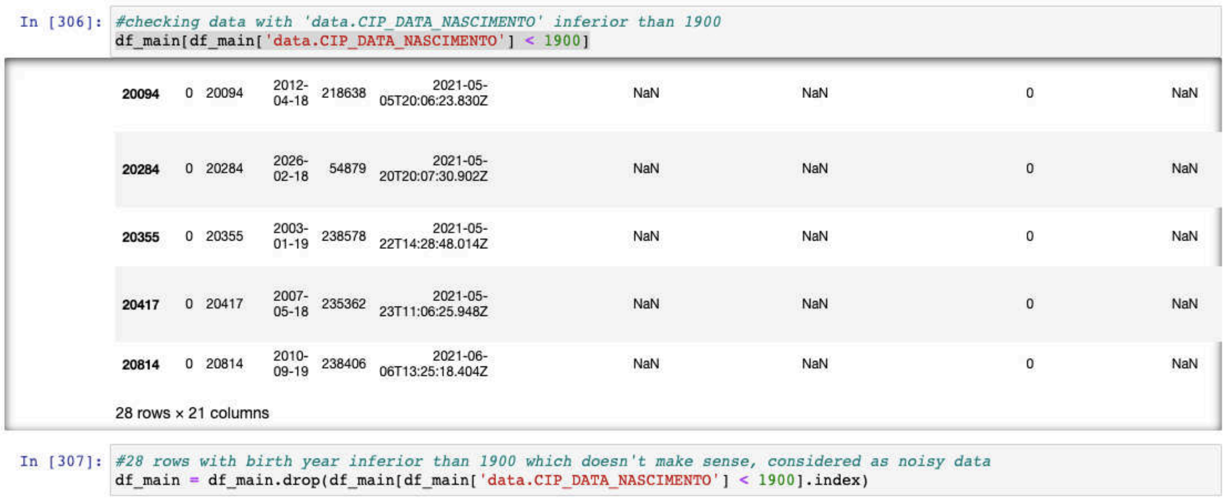
```
In [303]: #replacing 'NaN' values with '0'
          df_main['data.CIP_DATA_NASCIMENTO'] = df_main['data.CIP_DATA_NASCIMENTO'].replace('NaN', '0')
          #retriving year from 'data.CIP_DATA_NASCIMENTO'
          df_main['data.CIP_DATA_NASCIMENTO'] = df_main['data.CIP_DATA_NASCIMENTO'].astype(str).str[0:4].astype(int)
          #checking data with 'data.CIP_DATA_NASCIMENTO' superior than 2018
          df_main[df_main['data.CIP_DATA_NASCIMENTO'] > 2018]
```

22175	0	22175	04-20	232275	12T19:52:50.788Z	NaN	S	2021	NaN
22189	0	22189	2012-01-20	240546	2021-12-18T13:23:38.841Z	NaN	S	2020	NaN
33138	0	33138	2014-09-20	192583	2022-04-10T00:41:56.015Z	NaN	S	2095	NaN
42460	0	42460	2005-02-20	188915	2022-04-10T01:57:04.565Z	NaN	S	2093	NaN
47990	0	47990	2013-07-20	191764	2022-04-10T02:51:55.633Z	NaN	S	2092	NaN

7 rows × 21 columns

```
In [304]: #7 rows with birth year superior than 2018 which doesn't make sense, having a registed persons with 3 years old or less
          df_main = df_main.drop(df_main[df_main['data.CIP_DATA_NASCIMENTO'] > 2018].index)
```

Appendix 22 – Drop of 7 people who had birth year superior to 2018



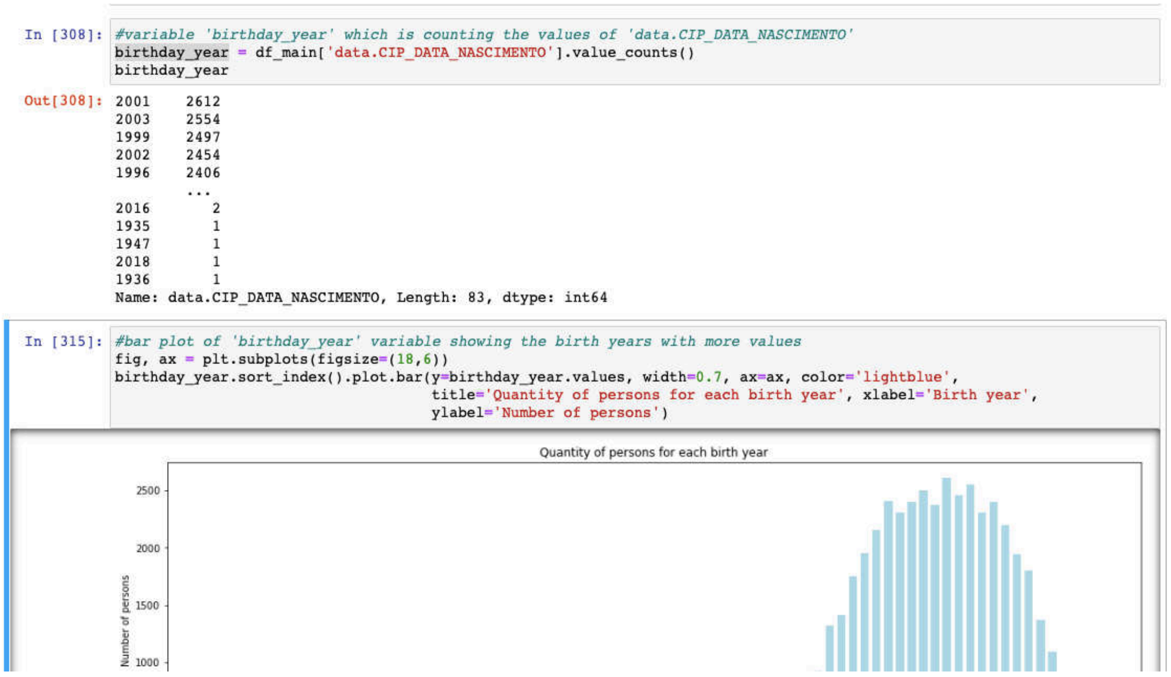
```
In [306]: #checking data with 'data.CIP_DATA_NASCIMENTO' inferior than 1900
          df_main[df_main['data.CIP_DATA_NASCIMENTO'] < 1900]
```

20094	0	20094	2012-04-18	218638	2021-05-05T20:06:23.830Z	NaN	NaN	0	NaN
20284	0	20284	2026-02-18	54879	2021-05-20T20:07:30.902Z	NaN	NaN	0	NaN
20355	0	20355	2003-01-19	238578	2021-05-22T14:28:48.014Z	NaN	NaN	0	NaN
20417	0	20417	2007-05-18	235362	2021-05-23T11:06:25.948Z	NaN	NaN	0	NaN
20814	0	20814	2010-09-19	238406	2021-06-06T13:25:18.404Z	NaN	NaN	0	NaN

28 rows × 21 columns

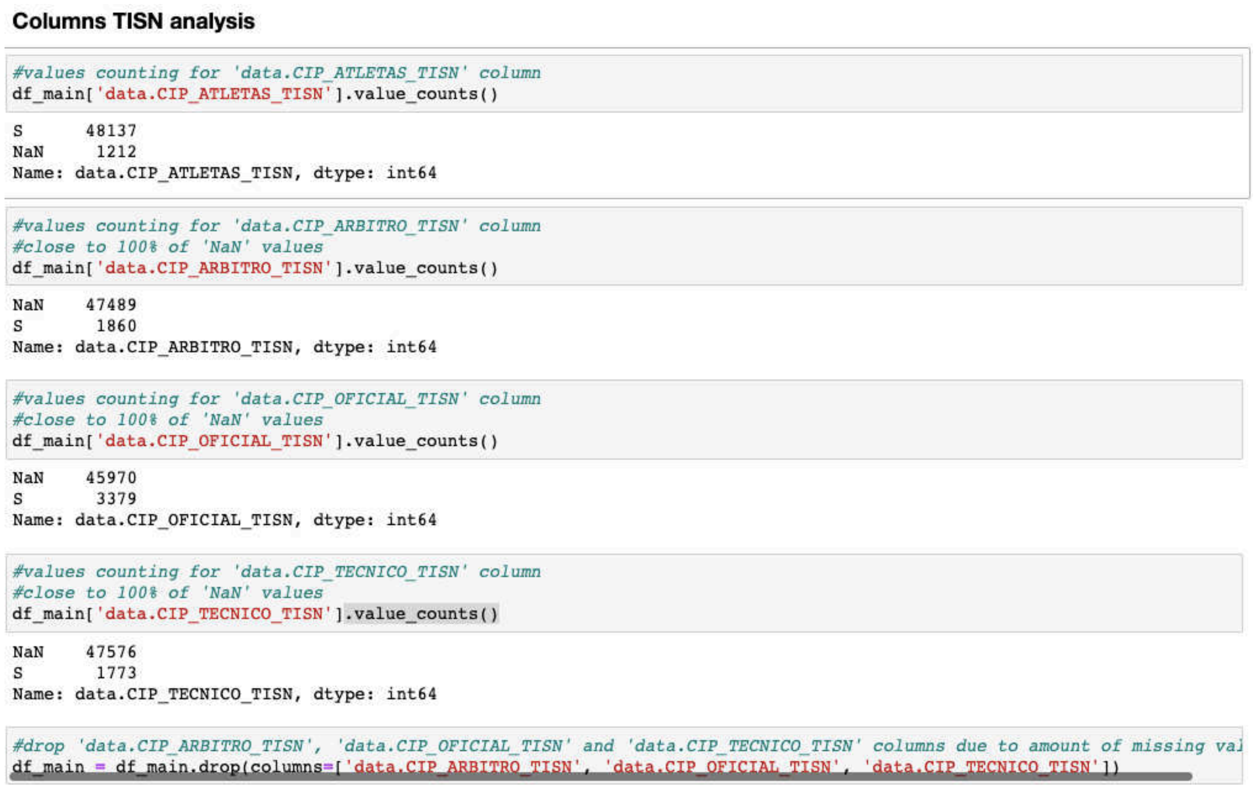
```
In [307]: #28 rows with birth year inferior than 1900 which doesn't make sense, considered as noisy data
          df_main = df_main.drop(df_main[df_main['data.CIP_DATA_NASCIMENTO'] < 1900].index)
```

Appendix 23 – Drop of 28 people who had birth year inferior to 2018



```
In [308]: #variable 'birthday_year' which is counting the values of 'data.CIP_DATA_NASCIMENTO'
          birthday_year = df_main['data.CIP_DATA_NASCIMENTO'].value_counts()
          birthday_year

Out[308]: 2001    2612
          2003    2554
          1999    2497
          2002    2454
          1996    2406
                  ...
          2016       2
          1935       1
          1947       1
          2018       1
          1936       1
          Name: data.CIP_DATA_NASCIMENTO, Length: 83, dtype: int64

In [315]: #bar plot of 'birthday_year' variable showing the birth years with more values
          fig, ax = plt.subplots(figsize=(18,6))
          birthday_year.sort_index().plot.bar(y=birthday_year.values, width=0.7, ax=ax, color='lightblue',
                                               title='Quantity of persons for each birth year', xlabel='Birth year',
                                               ylabel='Number of persons')
```

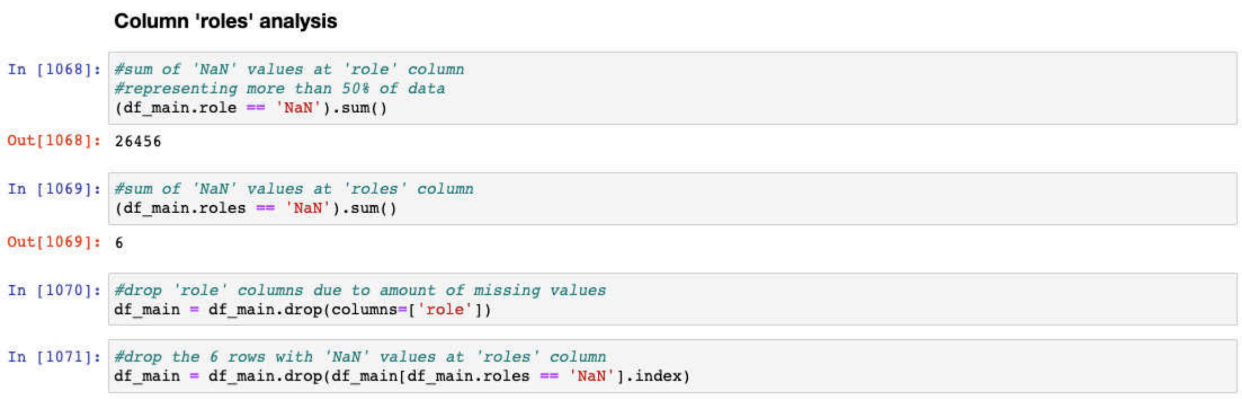
Appendix 24 – Bar plot code showing the number of individuals for each birth year



Columns TISN analysis

```
#values counting for 'data.CIP_ATLETAS_TISN' column
df_main['data.CIP_ATLETAS_TISN'].value_counts()

S      48137
NaN     1212
Name: data.CIP_ATLETAS_TISN, dtype: int64

#values counting for 'data.CIP_ARBITRO_TISN' column
#close to 100% of 'NaN' values
df_main['data.CIP_ARBITRO_TISN'].value_counts()

NaN    47489
S       1860
Name: data.CIP_ARBITRO_TISN, dtype: int64

#values counting for 'data.CIP_OFICIAL_TISN' column
#close to 100% of 'NaN' values
df_main['data.CIP_OFICIAL_TISN'].value_counts()

NaN    45970
S       3379
Name: data.CIP_OFICIAL_TISN, dtype: int64

#values counting for 'data.CIP_TECNICO_TISN' column
#close to 100% of 'NaN' values
df_main['data.CIP_TECNICO_TISN'].value_counts()

NaN    47576
S       1773
Name: data.CIP_TECNICO_TISN, dtype: int64

#drop 'data.CIP_ARBITRO_TISN', 'data.CIP_OFICIAL_TISN' and 'data.CIP_TECNICO_TISN' columns due to amount of missing va
df_main = df_main.drop(columns=['data.CIP_ARBITRO_TISN', 'data.CIP_OFICIAL_TISN', 'data.CIP_TECNICO_TISN'])
```

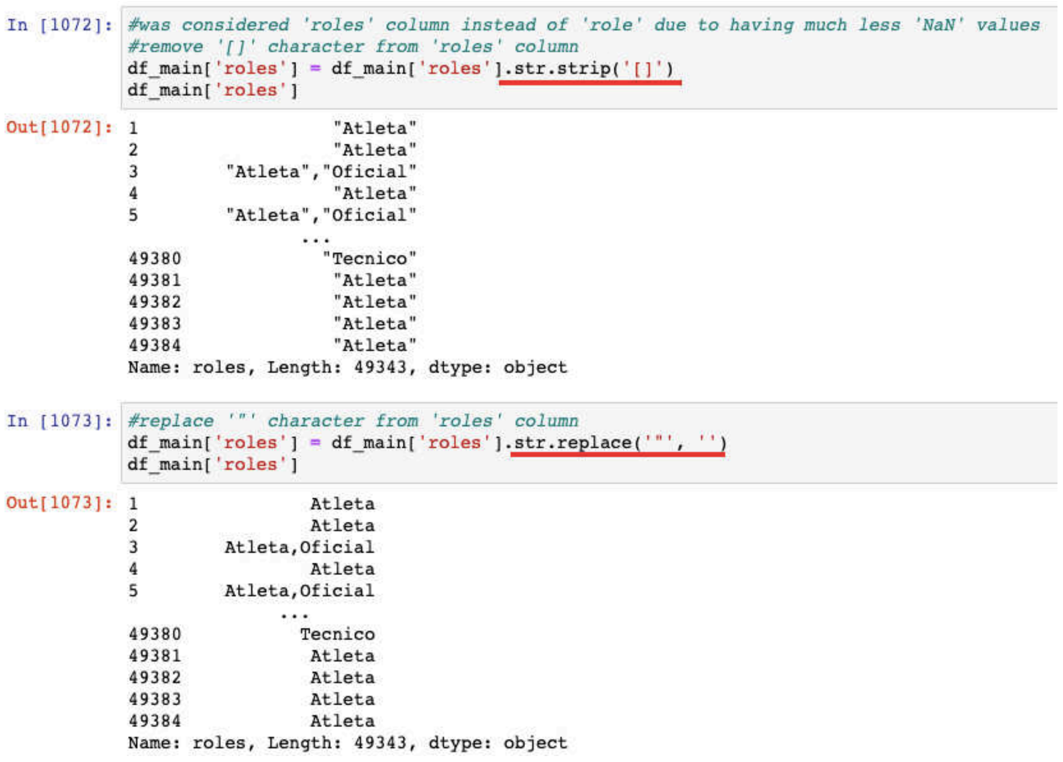
Appendix 25 – Missing values of TISN columns and drop



Column 'roles' analysis

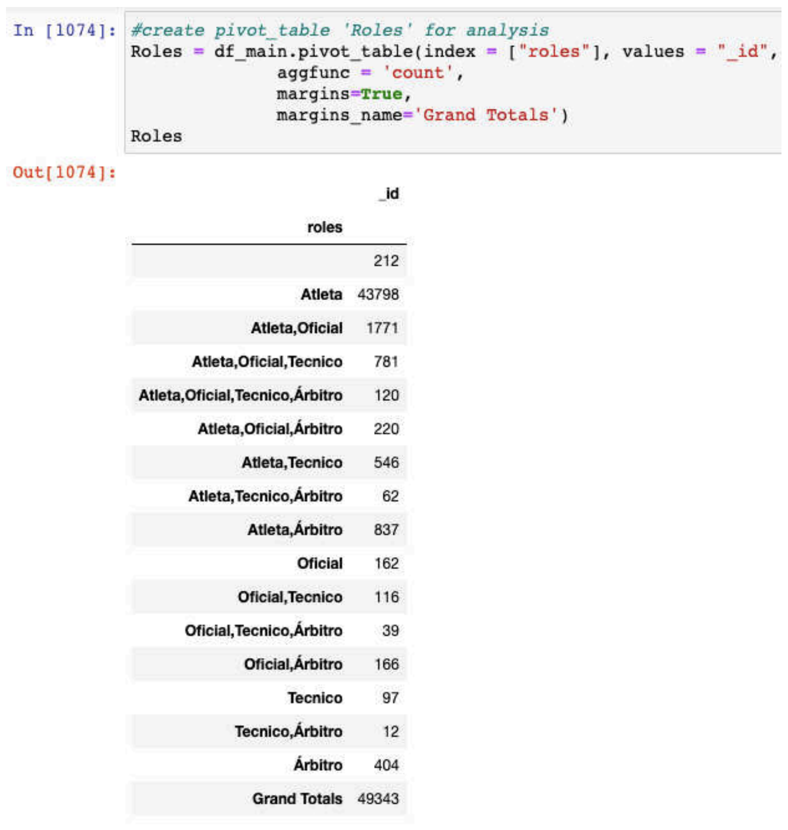
```
In [1068]: #sum of 'NaN' values at 'role' column
           #representing more than 50% of data
           (df_main.role == 'NaN').sum()

Out[1068]: 26456

In [1069]: #sum of 'NaN' values at 'roles' column
           (df_main.roles == 'NaN').sum()

Out[1069]: 6

In [1070]: #drop 'role' columns due to amount of missing values
           df_main = df_main.drop(columns=['role'])

In [1071]: #drop the 6 rows with 'NaN' values at 'roles' column
           df_main = df_main.drop(df_main[df_main.roles == 'NaN'].index)
```

Appendix 26 – 'role' and 'roles' columns missing values check, drop of 'role' and drop missing values rows of 'roles'



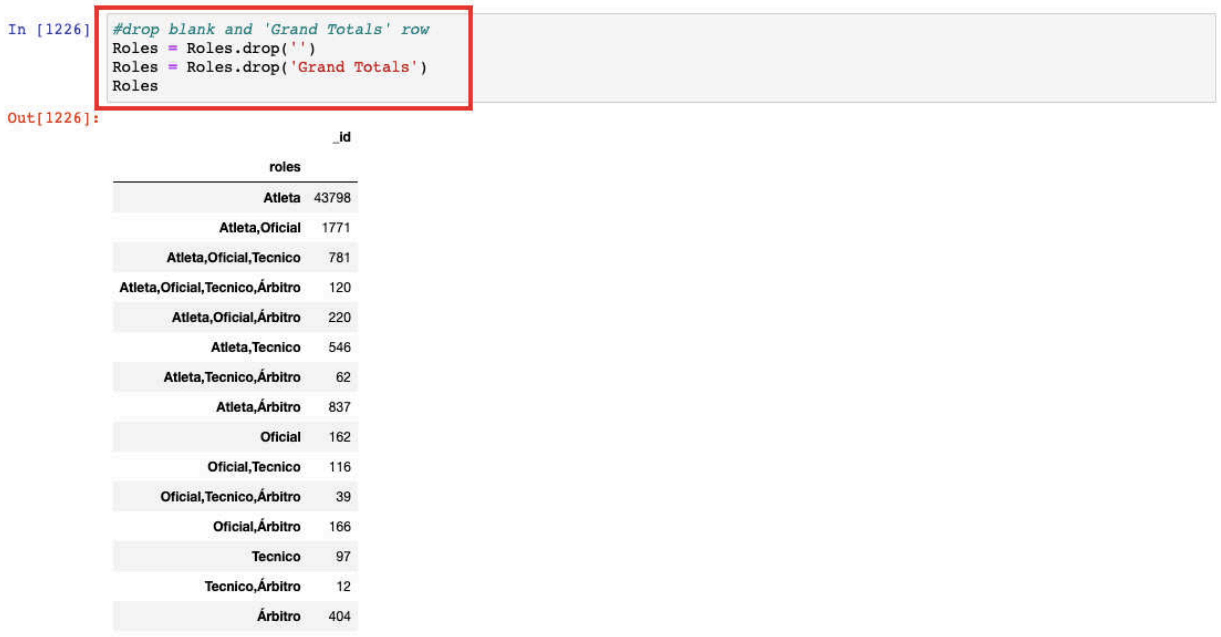
```
In [1072]: #was considered 'roles' column instead of 'role' due to having much less 'NaN' values
           #remove '[]' character from 'roles' column
           df_main['roles'] = df_main['roles'].str.strip('[]')
           df_main['roles']

Out[1072]: 1                  "Atleta"
           2                  "Atleta"
           3         "Atleta","Oficial"
           4                  "Atleta"
           5         "Atleta","Oficial"
                          ...
           49380             "Tecnico"
           49381              "Atleta"
           49382              "Atleta"
           49383              "Atleta"
           49384              "Atleta"
           Name: roles, Length: 49343, dtype: object

In [1073]: #replace '"' character from 'roles' column
           df_main['roles'] = df_main['roles'].str.replace('"', '')
           df_main['roles']

Out[1073]: 1                 Atleta
           2                 Atleta
           3         Atleta,Oficial
           4                 Atleta
           5         Atleta,Oficial
                          ...
           49380            Tecnico
           49381             Atleta
           49382             Atleta
           49383             Atleta
           49384             Atleta
           Name: roles, Length: 49343, dtype: object
```

Appendix 27 – Characters removed with 'strip' and 'replace' functions



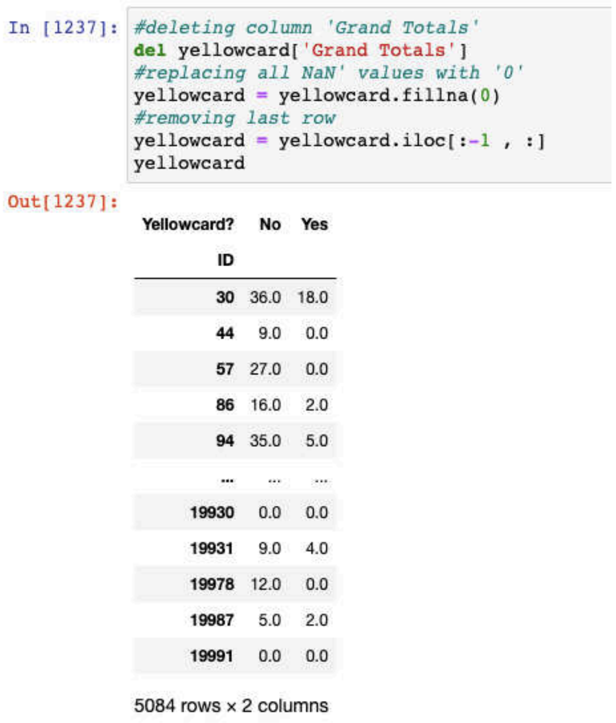
```
In [1074]: #create pivot_table 'Roles' for analysis
            Roles = df_main.pivot_table(index = ["roles"], values = "_id",
                            aggfunc = 'count',
                            margins=True,
                            margins_name='Grand Totals')
            Roles
Out[1074]:
                                              _id
                                   roles
                                                212
                                  Atleta      43798
                          Atleta,Oficial       1771
                  Atleta,Oficial,Tecnico        781
          Atleta,Oficial,Tecnico,Árbitro        120
                  Atleta,Oficial,Árbitro        220
                          Atleta,Tecnico        546
                  Atleta,Tecnico,Árbitro         62
                          Atleta,Árbitro        837
                                 Oficial        162
                         Oficial,Tecnico        116
                 Oficial,Tecnico,Árbitro         39
                         Oficial,Árbitro        166
                                 Tecnico         97
                         Tecnico,Árbitro         12
                                 Árbitro        404
                            Grand Totals      49343
```

Appendix 28 – 'Roles' pivot_table



```
In [1226]: #drop blank and 'Grand Totals' row
           Roles = Roles.drop('')
           Roles = Roles.drop('Grand Totals')
           Roles
Out[1226]:
                                              _id
                                   roles
                                  Atleta      43798
                          Atleta,Oficial       1771
                  Atleta,Oficial,Tecnico        781
          Atleta,Oficial,Tecnico,Árbitro        120
                  Atleta,Oficial,Árbitro        220
                          Atleta,Tecnico        546
                  Atleta,Tecnico,Árbitro         62
                          Atleta,Árbitro        837
                                 Oficial        162
                         Oficial,Tecnico        116
                 Oficial,Tecnico,Árbitro         39
                         Oficial,Árbitro        166
                                 Tecnico         97
                         Tecnico,Árbitro         12
                                 Árbitro        404
```

Appendix 29 – Blank and 'Grand Totals' line removed from pivot_table



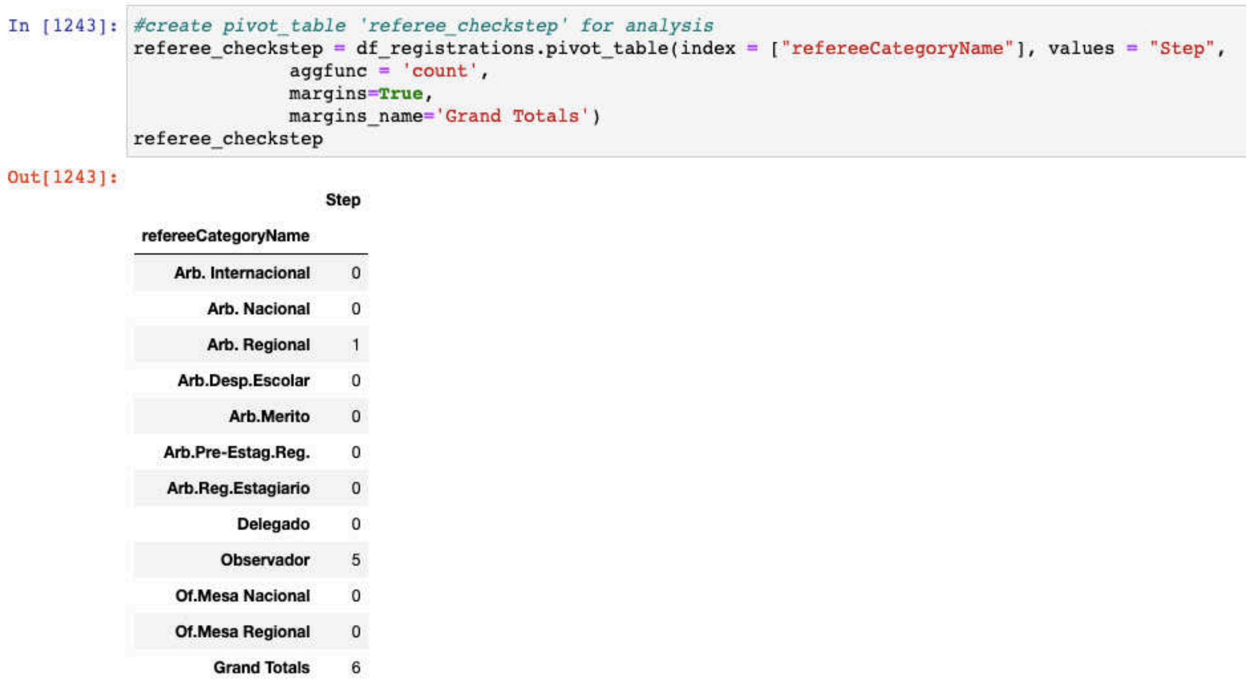
```
In [1237]: #deleting column 'Grand Totals'
           del yellowcard['Grand Totals']
           #replacing all NaN' values with '0'
           yellowcard = yellowcard.fillna(0)
           #removing last row
           yellowcard = yellowcard.iloc[:-1 , :]
           yellowcard
```

Out[1237]:

Yellowcard?	No	Yes
ID		
30	36.0	18.0
44	9.0	0.0
57	27.0	0.0
86	16.0	2.0
94	35.0	5.0
...	...	...
19930	0.0	0.0
19931	9.0	4.0
19978	12.0	0.0
19987	5.0	2.0
19991	0.0	0.0

5084 rows × 2 columns

Appendix 30 – Pivot_table 'yellowcard' cleaning

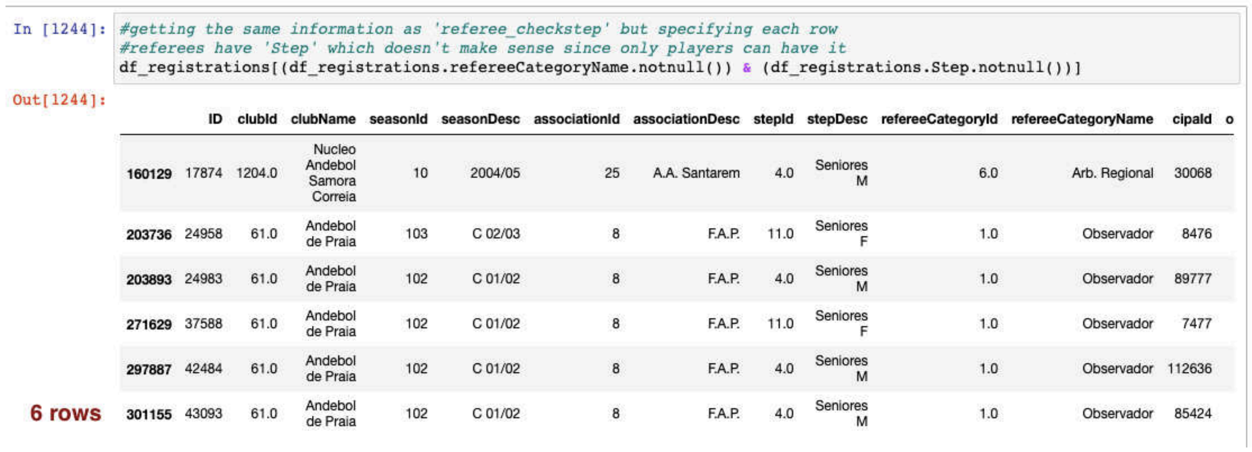


```
In [1243]: #create pivot_table 'referee_checkstep' for analysis
           referee_checkstep = df_registrations.pivot_table(index = ["refereeCategoryName"], values = "Step",
                           aggfunc = 'count',
                           margins=True,
                           margins_name='Grand Totals')
           referee_checkstep
```

Out[1243]:

	Step
refereeCategoryName	
Arb. Internacional	0
Arb. Nacional	0
Arb. Regional	1
Arb.Desp.Escolar	0
Arb.Merito	0
Arb.Pre-Estag.Reg.	0
Arb.Reg.Estagiario	0
Delegado	0
Observador	5
Of.Mesa Nacional	0
Of.Mesa Regional	0
Grand Totals	6

Appendix 31 – 'referee_checkstep' pivot_table

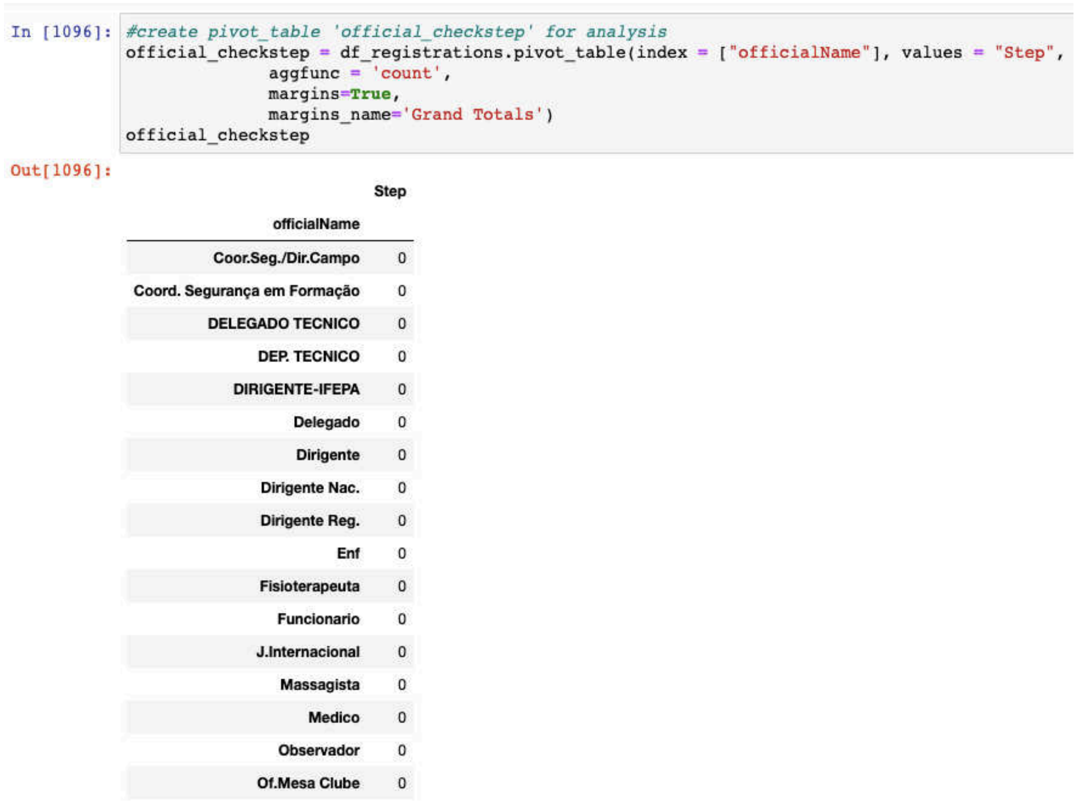


```
In [1244]: #getting the same information as 'referee_checkstep' but specifying each row
            #referees have 'Step' which doesn't make sense since only players can have it
            df_registrations[(df_registrations.refereeCategoryName.notnull()) & (df_registrations.Step.notnull())]
Out[1244]:
```

	ID	clubId	clubName	seasonId	seasonDesc	associationId	associationDesc	stepId	stepDesc	refereeCategoryId	refereeCategoryName	cipaId	o
160129	17874	1204.0	Nucleo Andebol Samora Correia	10	2004/05	25	A.A. Santarem	4.0	Seniores M	6.0	Arb. Regional	30068	
203736	24958	61.0	Andebol de Praia	103	C 02/03	8	F.A.P.	11.0	Seniores F	1.0	Observador	8476	
203893	24983	61.0	Andebol de Praia	102	C 01/02	8	F.A.P.	4.0	Seniores M	1.0	Observador	89777	
271629	37588	61.0	Andebol de Praia	102	C 01/02	8	F.A.P.	11.0	Seniores F	1.0	Observador	7477	
297887	42484	61.0	Andebol de Praia	102	C 01/02	8	F.A.P.	4.0	Seniores M	1.0	Observador	112636	
301155	43093	61.0	Andebol de Praia	102	C 01/02	8	F.A.P.	4.0	Seniores M	1.0	Observador	85424	

6 rows

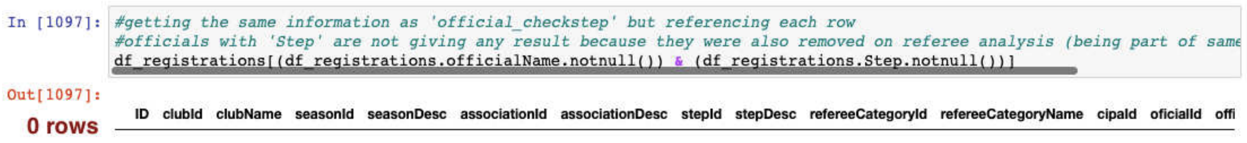
Appendix 32 – Specifiying each row mentioned on 'referee_checkstep' pivot_table



```
In [1096]: #create pivot_table 'official_checkstep' for analysis
           official_checkstep = df_registrations.pivot_table(index = ["officialName"], values = "Step",
                         aggfunc = 'count',
                         margins=True,
                         margins_name='Grand Totals')
           official_checkstep
Out[1096]:
```

officialName	Step
Coor.Seg./Dir.Campo	0
Coord. Segurança em Formação	0
DELEGADO TECNICO	0
DEP. TECNICO	0
DIRIGENTE-IFEPA	0
Delegado	0
Dirigente	0
Dirigente Nac.	0
Dirigente Reg.	0
Enf	0
Fisioterapeuta	0
Funcionario	0
J.Internacional	0
Massagista	0
Medico	0
Observador	0
Of.Mesa Clube	0

Appendix 33 – 'official_checkstep' pivot_table

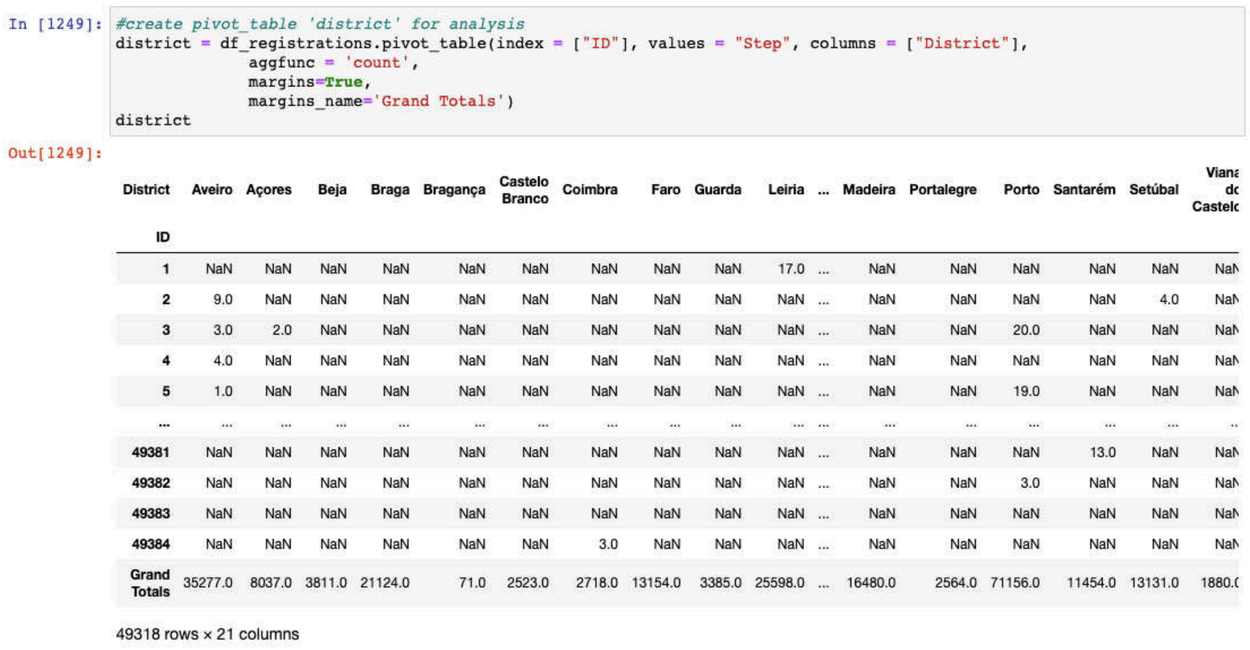


```
In [1097]: #getting the same information as 'official_checkstep' but referencing each row
           #officials with 'Step' are not giving any result because they were also removed on referee analysis (being part of same
           df_registrations[(df_registrations.officialName.notnull()) & (df_registrations.Step.notnull())]
Out[1097]:
```

ID	clubId	clubName	seasonId	seasonDesc	associationId	associationDesc	stepId	stepDesc	refereeCategoryId	refereeCategoryName	cipaId	oficialId	offi

0 rows

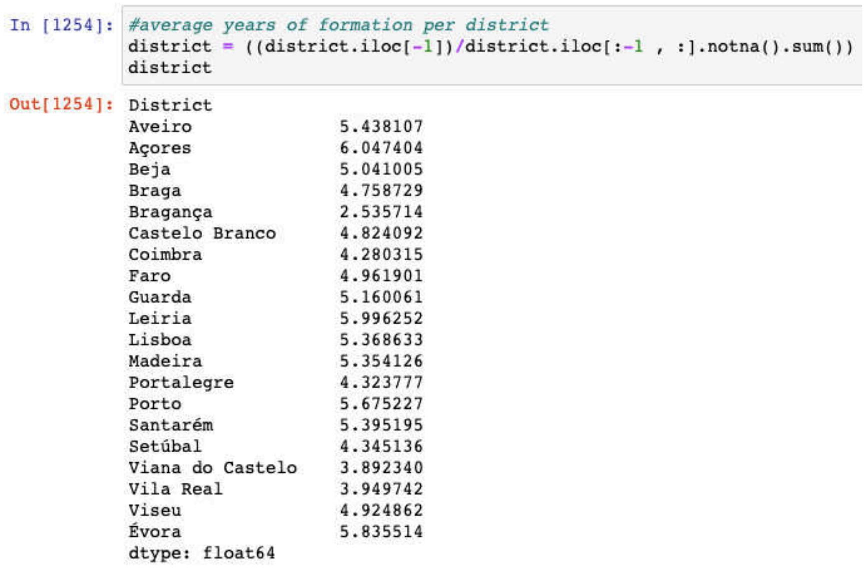
Appendix 34 – No rows existent from 'official_checkstep' pivot_table


```
In [1249]: #create pivot_table 'district' for analysis
           district = df_registrations.pivot_table(index = ["ID"], values = "Step", columns = ["District"],
                         aggfunc = 'count',
                         margins=True,
                         margins_name='Grand Totals')
           district
Out[1249]:
```

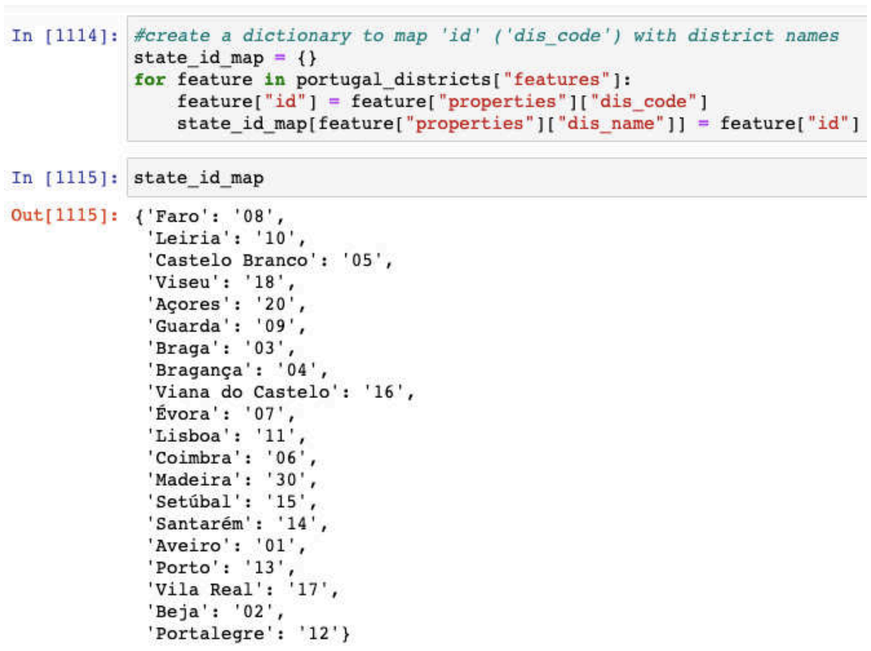
District	Aveiro	Açores	Beja	Braga	Bragança	Castelo Branco	Coimbra	Faro	Guarda	Leiria	...	Madeira	Portalegre	Porto	Santarém	Setúbal	Viana dc Castelc
ID																	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	17.0	...	NaN	NaN	NaN	NaN	NaN	NaN
2	9.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	4.0	NaN
3	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	20.0	NaN	NaN	NaN
4	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
5	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	19.0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
49381	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	13.0	NaN	NaN
49382	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	3.0	NaN	NaN	NaN
49383	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
49384	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
Grand Totals	35277.0	8037.0	3811.0	21124.0	71.0	2523.0	2718.0	13154.0	3385.0	25598.0	...	16480.0	2564.0	71156.0	11454.0	13131.0	1880.

49318 rows × 21 columns

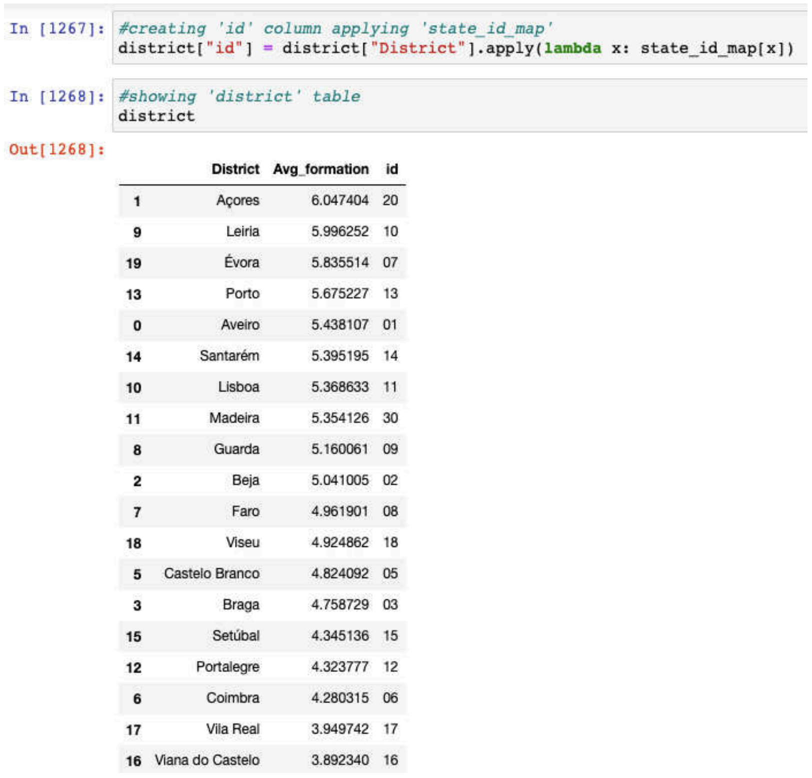
Appendix 35 – 'district' pivot_table


```
In [1254]: #average years of formation per district
           district = ((district.iloc[-1])/district.iloc[:-1 , :].notna().sum())
           district

Out[1254]: District
           Aveiro              5.438107
           Açores              6.047404
           Beja                5.041005
           Braga               4.758729
           Bragança            2.535714
           Castelo Branco      4.824092
           Coimbra             4.280315
           Faro                4.961901
           Guarda              5.160061
           Leiria              5.996252
           Lisboa              5.368633
           Madeira             5.354126
           Portalegre          4.323777
           Porto               5.675227
           Santarém            5.395195
           Setúbal             4.345136
           Viana do Castelo    3.892340
           Vila Real           3.949742
           Viseu               4.924862
           Évora               5.835514
           dtype: float64
```

Appendix 36 – Calculation of average formation year per district


```
In [1114]: #create a dictionary to map 'id' ('dis_code') with district names
            state_id_map = {}
            for feature in portugal_districts["features"]:
                feature["id"] = feature["properties"]["dis_code"]
                state_id_map[feature["properties"]["dis_name"]] = feature["id"]

In [1115]: state_id_map

Out[1115]: {'Faro': '08',
             'Leiria': '10',
             'Castelo Branco': '05',
             'Viseu': '18',
             'Açores': '20',
             'Guarda': '09',
             'Braga': '03',
             'Bragança': '04',
             'Viana do Castelo': '16',
             'Évora': '07',
             'Lisboa': '11',
             'Coimbra': '06',
             'Madeira': '30',
             'Setúbal': '15',
             'Santarém': '14',
             'Aveiro': '01',
             'Porto': '13',
             'Vila Real': '17',
             'Beja': '02',
             'Portalegre': '12'}
```

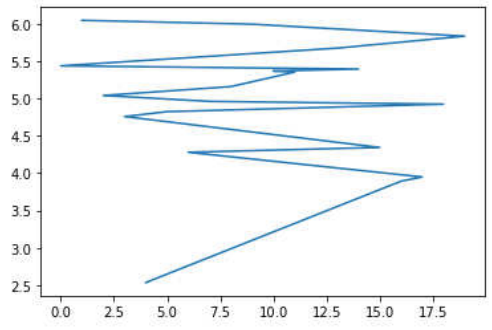
Appendix 37 – Dictionary to map 'id' district


```
In [1267]: #creating 'id' column applying 'state_id_map'
           district["id"] = district["District"].apply(lambda x: state_id_map[x])

In [1268]: #showing 'district' table
           district

Out[1268]:
```

	District	Avg_formation	id
1	Açores	6.047404	20
9	Leiria	5.996252	10
19	Évora	5.835514	07
13	Porto	5.675227	13
0	Aveiro	5.438107	01
14	Santarém	5.395195	14
10	Lisboa	5.368633	11
11	Madeira	5.354126	30
8	Guarda	5.160061	09
2	Beja	5.041005	02
7	Faro	4.961901	08
18	Viseu	4.924862	18
5	Castelo Branco	4.824092	05
3	Braga	4.758729	03
15	Setúbal	4.345136	15
12	Portalegre	4.323777	12
6	Coimbra	4.280315	06
17	Vila Real	3.949742	17
16	Viana do Castelo	3.892340	16

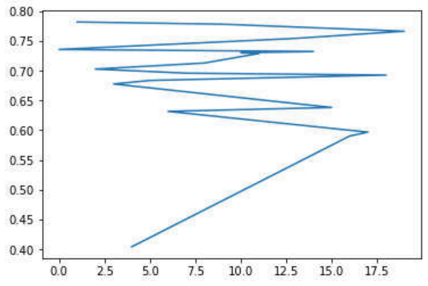
Appendix 38 – 'district' table having 'id' column added

```
In [1270]: #plotting 'Avg_formation' column
           district["Avg_formation"].plot()

Out[1270]: <AxesSubplot:>
```



Appendix 39 – Plot 'Avg_formation' column

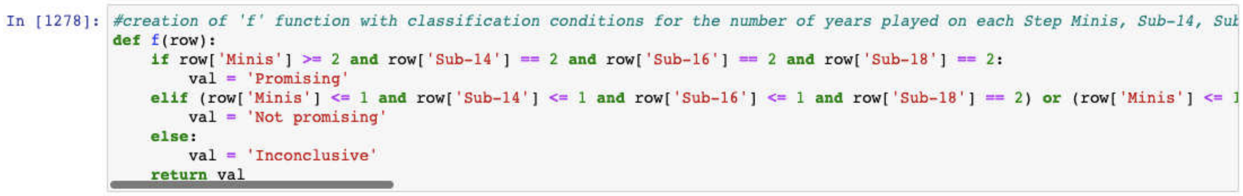
```
In [1271]: #creating 'Avg_formation_scale' with logorithm values of 'Avg_formation'
           #this column will be use for map draw color having more colour variations due to values being more close to each other
           district["Avg_formation_scale"] = np.log10(district["Avg_formation"])
           ##plotting 'Avg_formation_scale' column
           district["Avg_formation_scale"].plot()

Out[1271]: <AxesSubplot:>
```



Appendix 40 – Plot 'Avg_formation_scale' column

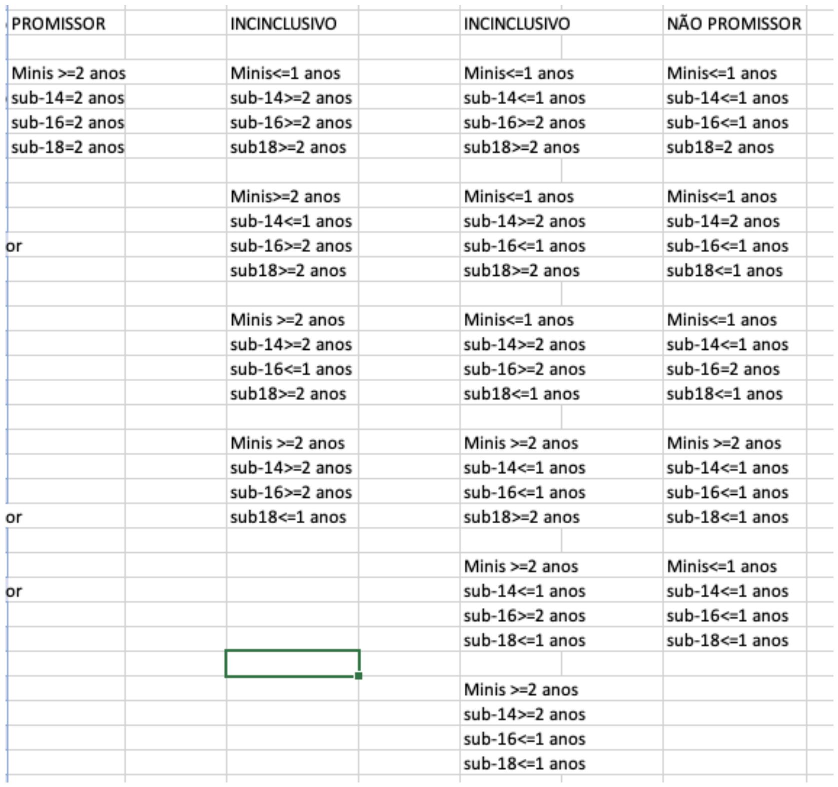
```
In [1272]: #install required package for map
           !pip install geojson-rewind

           Requirement already satisfied: geojson-rewind in /Users/miguelcoutinhonunes/opt/anaconda3/lib/python3.8/site-packages
           (1.0.3)
           WARNING: You are using pip version 21.0.1; however, version 22.3 is available.
           You should consider upgrading via the '/Users/miguelcoutinhonunes/opt/anaconda3/bin/python -m pip install --upgrade p
           ip' command.

In [1273]: #a Python library for enforcing polygon ring winding order in GeoJSON, that is reorganizing portugal districts coordina
           from geojson_rewind import rewind
           portugal_districts_rewound = rewind(portugal_districts,rfc7946=False)

In [1274]: #map Portugal districts taking into account the average formation years
           fig = px.choropleth(
               district,
               locations="id",
               geojson=portugal_districts_rewound,
               color="Avg_formation_scale",
               color_continuous_scale="Viridis",
               hover_name="District",
               hover_data=["Avg_formation"],
               title="Average Formation years per district",
           )
           fig.update_geos(fitbounds="locations", visible=True)
           fig.show()
```

Appendix 41 – Average formation year per district map code



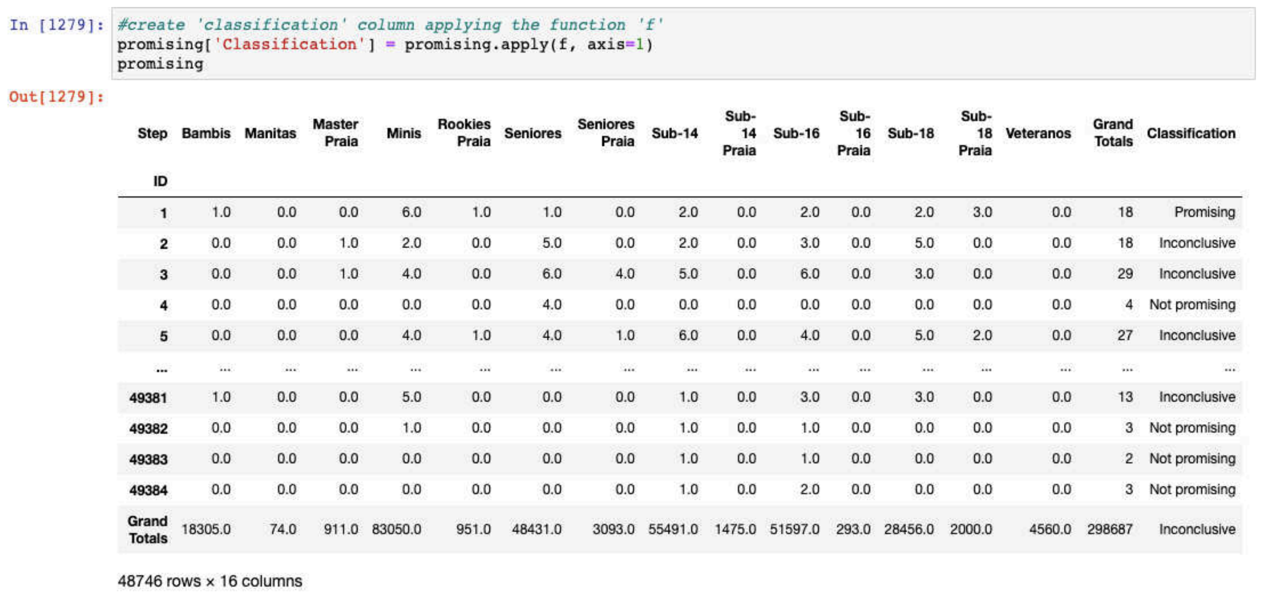
```
In [1278]: #creation of 'f' function with classification conditions for the number of years played on each Step Minis, Sub-14, Sub
def f(row):
    if row['Minis'] >= 2 and row['Sub-14'] == 2 and row['Sub-16'] == 2 and row['Sub-18'] == 2:
        val = 'Promising'
    elif (row['Minis'] <= 1 and row['Sub-14'] <= 1 and row['Sub-16'] <= 1 and row['Sub-18'] == 2) or (row['Minis'] <= 1
        val = 'Not promising'
    else:
        val = 'Inconclusive'
    return val
```

Appendix 42 – Function 'f' on python code



PROMISSOR	INCINCLUSIVO	INCINCLUSIVO	NÃO PROMISSOR
Minis >=2 anos	Minis<=1 anos	Minis<=1 anos	Minis<=1 anos
sub-14=2 anos	sub-14>=2 anos	sub-14<=1 anos	sub-14<=1 anos
sub-16=2 anos	sub-16>=2 anos	sub-16>=2 anos	sub-16<=1 anos
sub-18=2 anos	sub18>=2 anos	sub18>=2 anos	sub18=2 anos
	Minis>=2 anos	Minis<=1 anos	Minis<=1 anos
	sub-14<=1 anos	sub-14>=2 anos	sub-14=2 anos
or	sub-16>=2 anos	sub-16<=1 anos	sub-16<=1 anos
	sub18>=2 anos	sub18>=2 anos	sub18<=1 anos
	Minis >=2 anos	Minis<=1 anos	Minis<=1 anos
	sub-14>=2 anos	sub-14>=2 anos	sub-14<=1 anos
	sub-16<=1 anos	sub-16>=2 anos	sub-16=2 anos
	sub18>=2 anos	sub18<=1 anos	sub18<=1 anos
	Minis >=2 anos	Minis >=2 anos	Minis >=2 anos
	sub-14>=2 anos	sub-14<=1 anos	sub-14<=1 anos
	sub-16>=2 anos	sub-16<=1 anos	sub-16<=1 anos
or	sub18<=1 anos	sub18>=2 anos	sub-18<=1 anos
		Minis >=2 anos	Minis<=1 anos
or		sub-14<=1 anos	sub-14<=1 anos
		sub-16>=2 anos	sub-16<=1 anos
		sub-18<=1 anos	sub-18<=1 anos
		Minis >=2 anos	
		sub-14>=2 anos	
		sub-16<=1 anos	
		sub-18<=1 anos	

Appendix 43 – Conditions of function 'f'



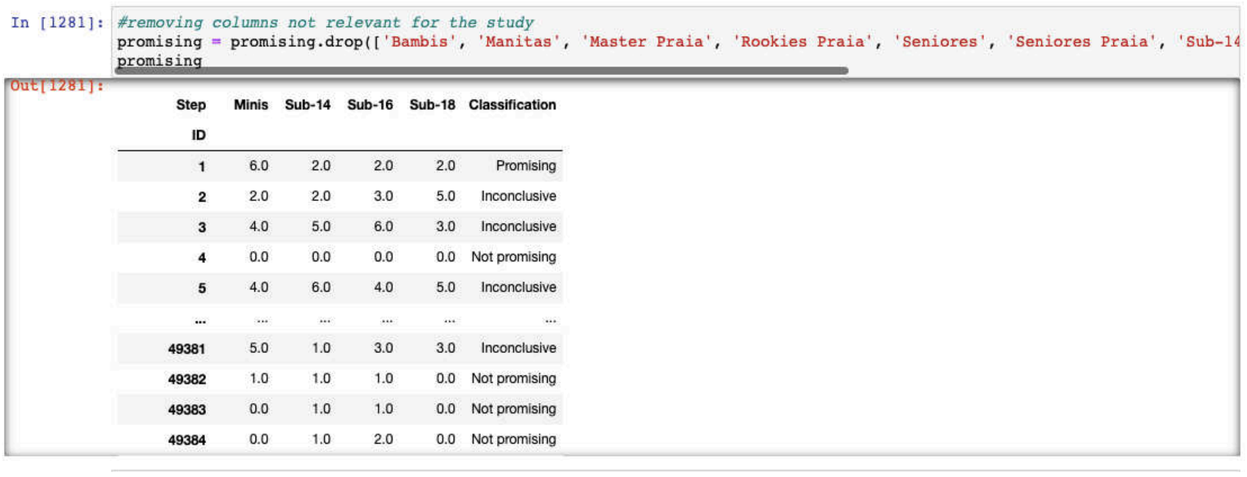
```
In [1279]: #create 'classification' column applying the function 'f'
           promising['Classification'] = promising.apply(f, axis=1)
           promising
```

Out[1279]:

Step	Bambis	Manitas	Master Praia	Minis	Rookies Praia	Seniores	Seniores Praia	Sub-14	Sub-14 Praia	Sub-16	Sub-16 Praia	Sub-18	Sub-18 Praia	Veteranos	Grand Totals	Classification
ID																
1	1.0	0.0	0.0	6.0	1.0	1.0	0.0	2.0	0.0	2.0	0.0	2.0	3.0	0.0	18	Promising
2	0.0	0.0	1.0	2.0	0.0	5.0	0.0	2.0	0.0	3.0	0.0	5.0	0.0	0.0	18	Inconclusive
3	0.0	0.0	1.0	4.0	0.0	6.0	4.0	5.0	0.0	6.0	0.0	3.0	0.0	0.0	29	Inconclusive
4	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4	Not promising
5	0.0	0.0	0.0	4.0	1.0	4.0	1.0	6.0	0.0	4.0	0.0	5.0	2.0	0.0	27	Inconclusive
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49381	1.0	0.0	0.0	5.0	0.0	0.0	0.0	1.0	0.0	3.0	0.0	3.0	0.0	0.0	13	Inconclusive
49382	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	3	Not promising
49383	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	2	Not promising
49384	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	2.0	0.0	0.0	0.0	0.0	3	Not promising
Grand Totals	18305.0	74.0	911.0	83050.0	951.0	48431.0	3093.0	55491.0	1475.0	51597.0	293.0	28456.0	2000.0	4560.0	298687	Inconclusive

48746 rows × 16 columns

Appendix 44 – Column 'Classification' creation from 'f' function

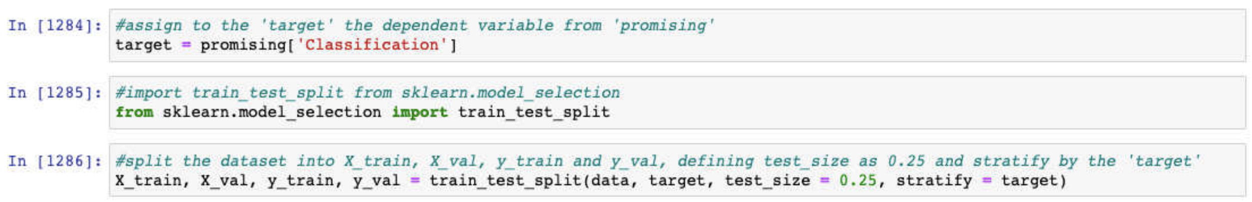


```
In [1281]: #removing columns not relevant for the study
           promising = promising.drop(['Bambis', 'Manitas', 'Master Praia', 'Rookies Praia', 'Seniores', 'Seniores Praia', 'Sub-14
           promising
```

Out[1281]:

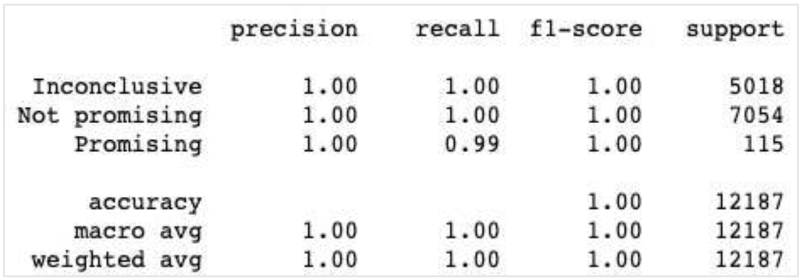
Step	Minis	Sub-14	Sub-16	Sub-18	Classification
ID					
1	6.0	2.0	2.0	2.0	Promising
2	2.0	2.0	3.0	5.0	Inconclusive
3	4.0	5.0	6.0	3.0	Inconclusive
4	0.0	0.0	0.0	0.0	Not promising
5	4.0	6.0	4.0	5.0	Inconclusive
...	...	...	...	...	...
49381	5.0	1.0	3.0	3.0	Inconclusive
49382	1.0	1.0	1.0	0.0	Not promising
49383	0.0	1.0	1.0	0.0	Not promising
49384	0.0	1.0	2.0	0.0	Not promising

Appendix 45 – Delete unnecessary columns for predictive analysis



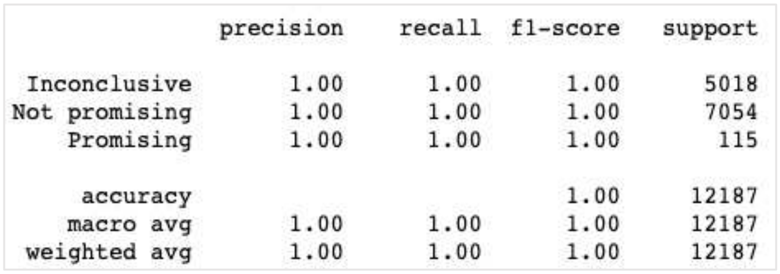
```
In [1284]: #assign to the 'target' the dependent variable from 'promising'
           target = promising['Classification']

In [1285]: #import train_test_split from sklearn.model_selection
           from sklearn.model_selection import train_test_split

In [1286]: #split the dataset into X_train, X_val, y_train and y_val, defining test_size as 0.25 and stratify by the 'target'
           X_train, X_val, y_train, y_val = train_test_split(data, target, test_size = 0.25, stratify = target)
```

Appendix 46 – 'Classification' column as target and split to 25% of validation data



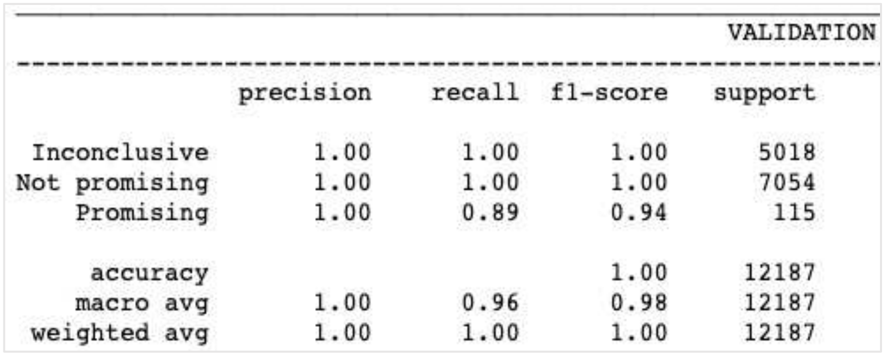
```
              precision    recall  f1-score   support

 Inconclusive       1.00      1.00      1.00      5018
Not promising       1.00      1.00      1.00      7054
    Promising       1.00      0.99      1.00       115

     accuracy                           1.00     12187
    macro avg       1.00      1.00      1.00     12187
 weighted avg       1.00      1.00      1.00     12187
```

Appendix 47 – Classification report of 'modelSVM' and 'modelKNN'



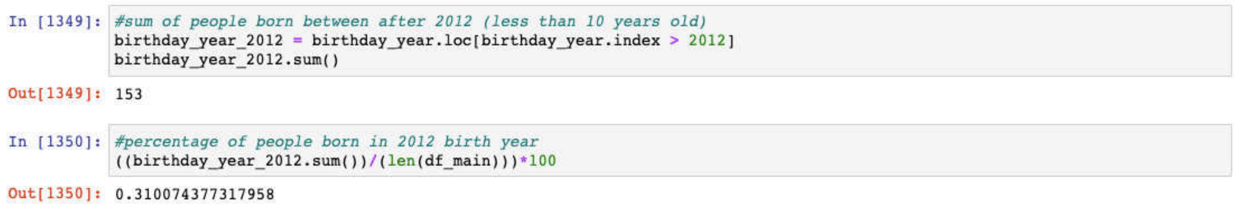
```
              precision    recall  f1-score   support

 Inconclusive       1.00      1.00      1.00      5018
Not promising       1.00      1.00      1.00      7054
    Promising       1.00      1.00      1.00       115

     accuracy                           1.00     12187
    macro avg       1.00      1.00      1.00     12187
 weighted avg       1.00      1.00      1.00     12187
```

Appendix 48 - Classification report of 'modelSVM_100'



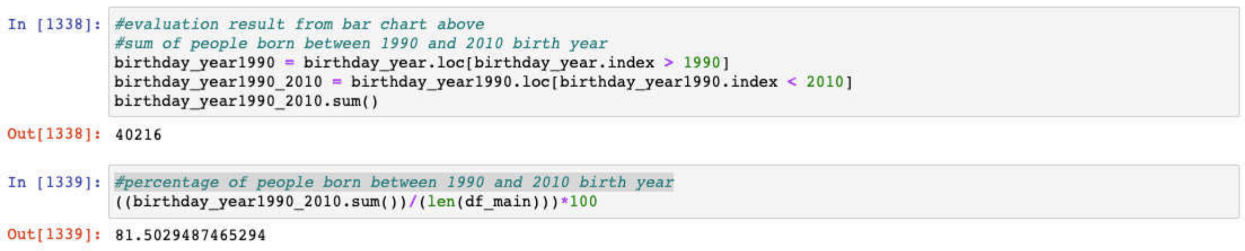
```
----------------------------------------------------------
                                               VALIDATION
----------------------------------------------------------
              precision    recall  f1-score   support

 Inconclusive       1.00      1.00      1.00      5018
Not promising       1.00      1.00      1.00      7054
    Promising       1.00      0.89      0.94       115

     accuracy                           1.00     12187
    macro avg       1.00      0.96      0.98     12187
 weighted avg       1.00      1.00      1.00     12187
```

Appendix 49 - Classification report of 'modelKNN50'



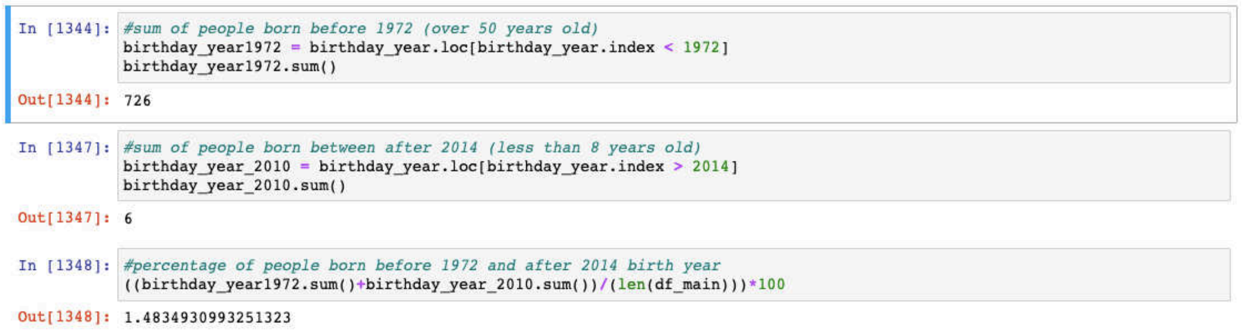
```
In [1349]: #sum of people born between after 2012 (less than 10 years old)
           birthday_year_2012 = birthday_year.loc[birthday_year.index > 2012]
           birthday_year_2012.sum()

Out[1349]: 153

In [1350]: #percentage of people born in 2012 birth year
           ((birthday_year_2012.sum())/(len(df_main)))*100

Out[1350]: 0.310074377317958
```

Appendix 50 – Percentage of people less than 10 years old



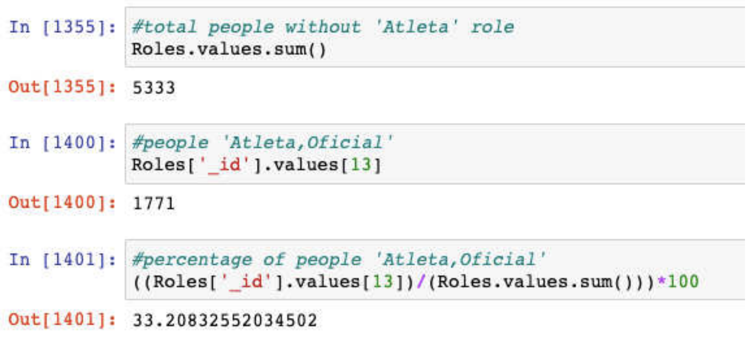
```
In [1338]: #evaluation result from bar chart above
           #sum of people born between 1990 and 2010 birth year
           birthday_year1990 = birthday_year.loc[birthday_year.index > 1990]
           birthday_year1990_2010 = birthday_year1990.loc[birthday_year1990.index < 2010]
           birthday_year1990_2010.sum()

Out[1338]: 40216

In [1339]: #percentage of people born between 1990 and 2010 birth year
           ((birthday_year1990_2010.sum())/(len(df_main)))*100

Out[1339]: 81.5029487465294
```

Appendix 51 – Percentage of people between 12 and 32 years old



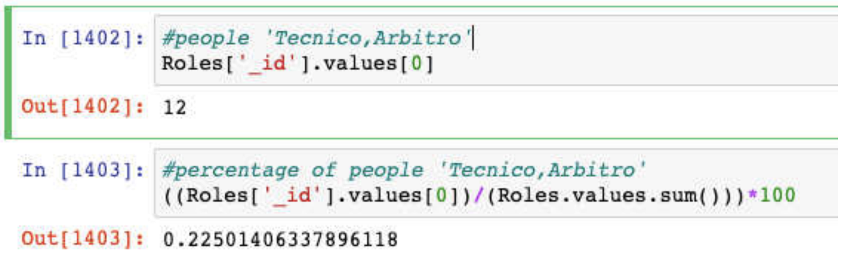
```
In [1344]: #sum of people born before 1972 (over 50 years old)
           birthday_year1972 = birthday_year.loc[birthday_year.index < 1972]
           birthday_year1972.sum()

Out[1344]: 726

In [1347]: #sum of people born between after 2014 (less than 8 years old)
           birthday_year_2010 = birthday_year.loc[birthday_year.index > 2014]
           birthday_year_2010.sum()

Out[1347]: 6

In [1348]: #percentage of people born before 1972 and after 2014 birth year
           ((birthday_year1972.sum()+birthday_year_2010.sum())/(len(df_main)))*100

Out[1348]: 1.4834930993251323
```

Appendix 52 – Percentage of people above 50 years old and less than 8 years old



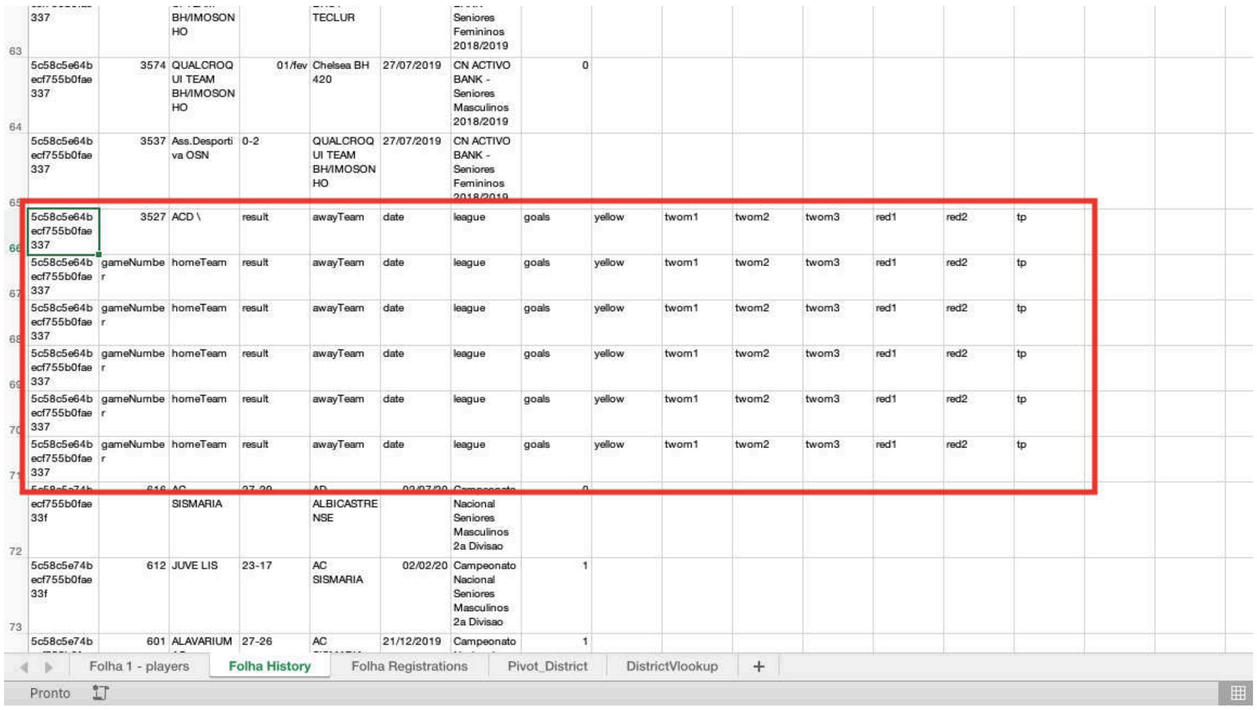
```
In [1355]: #total people without 'Atleta' role
           Roles.values.sum()

Out[1355]: 5333

In [1400]: #people 'Atleta,Oficial'
           Roles['_id'].values[13]

Out[1400]: 1771

In [1401]: #percentage of people 'Atleta,Oficial'
           ((Roles['_id'].values[13])/(Roles.values.sum()))*100

Out[1401]: 33.20832552034502
```

Appendix 53 – Percentage of role 'Atleta,Oficial'



```
In [1402]: #people 'Tecnico,Arbitro'
           Roles['_id'].values[0]

Out[1402]: 12

In [1403]: #percentage of people 'Tecnico,Arbitro'
           ((Roles['_id'].values[0])/(Roles.values.sum()))*100

Out[1403]: 0.22501406337896118
```

Appendix 54 – Percentage of role 'Tecnico,Arbitro'



63	337		BH/IMOSON HO		TECLUR		Seniores Femininos 2018/2019								
64	5c58c5e64b ecf755b0fae 337	3574	QUALCROQ UI TEAM BH/IMOSON HO	01/fev	Chelsea BH 420	27/07/2019	CN ACTIVO BANK - Seniores Masculinos 2018/2019	0							
65	5c58c5e64b ecf755b0fae 337	3537	Ass.Desporti va OSN	0-2	QUALCROQ UI TEAM BH/IMOSON HO	27/07/2019	CN ACTIVO BANK - Seniores Femininos 2018/2019								
66	5c58c5e64b ecf755b0fae 337	3527	ACD \	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
67	5c58c5e64b ecf755b0fae 337	gameNumbe r	homeTeam	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
68	5c58c5e64b ecf755b0fae 337	gameNumbe r	homeTeam	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
69	5c58c5e64b ecf755b0fae 337	gameNumbe r	homeTeam	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
70	5c58c5e64b ecf755b0fae 337	gameNumbe r	homeTeam	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
71	5c58c5e64b ecf755b0fae 337	gameNumbe r	homeTeam	result	awayTeam	date	league	goals	yellow	twom1	twom2	twom3	red1	red2	tp
72	5c58c5e74b ecf755b0fae 33f	616	AC SISMARIA	27-20	AD ALBICASTRE NSE	02/07/20	Campeonato Nacional Seniores Masculinos 2a Divisao	0							
73	5c58c5e74b ecf755b0fae 33f	612	JUVE LIS	23-17	AC SISMARIA	02/02/20	Campeonato Nacional Seniores Masculinos 2a Divisao	1							
	5c58c5e74b	601	ALAVARIUM	27-26	AC	21/12/2019	Campeonato	1							

Appendix 55 – Incorrect stored information in 'Folha History' sheet

VBA code function ‘SaveDataToCollection’:

```
Function SaveDataToCollection(str As String, ByRef Cols As Collection)
    Dim chrIndex&, countAspas As Long
    Dim chrJson$, colName As String
    Dim saveChr As Boolean

    str = Replace(str, chr(34) & "$oid" & chr(34) & ":", "")
    str = Replace(str, "ACD \" & chr(34) & " O SOT?O \" & chr(34) & " / THIS IS TEAM", "ACD \ O SOT?O \ / THIS IS
TEAM")
    str = Replace(str, "A. C. FAFE \" & chr(34) & "A\" & chr(34), "A. C. FAFE \A\")
    str = Replace(str, "A. C. FAFE \" & chr(34) & "B\" & chr(34), "A. C. FAFE \B\")
    str = Replace(str, "ACD \" & chr(34) & " O SOTÃO \" & chr(34) & " / THIS IS TEAM", "ACD \ O SOTÃO \ / THIS
IS TEAM")
    str = Replace(str, "ACD ACD \" & chr(34) & " O SOTÃO \" & chr(34) & " / TIT ROOKIES", "ACD ACD \ O SOTÃO
\ / TIT ROOKIES")
    str = Replace(str, "ADA MAIA / ISMAI \" & chr(34) & "A\" & chr(34), "ADA MAIA / ISMAI \A\")
    str = Replace(str, "ADA MAIA / ISMAI \" & chr(34) & "B\" & chr(34), "ADA MAIA / ISMAI \B\")
    str = Replace(str, "Ass.Desportiva OSN \" & chr(34) & "A\" & chr(34), "Ass.Desportiva OSN \A\")
    str = Replace(str, "Ass.Desportiva OSN \" & chr(34) & "B\" & chr(34), "Ass.Desportiva OSN \B\")
    str = Replace(str, "AD SANJOANENSE \" & chr(34) & "A\" & chr(34) & "/MAIS ÓPTICA", "AD SANJOANENSE
\A\/MAIS ÓPTICA")
    str = Replace(str, "AD SANJOANENSE \" & chr(34) & "\" & chr(34) & "A\" & chr(34) & "/DELBA", "AD
SANJOANENSE \A\/DELBA")
    str = Replace(str, "AD SANJOANENSE \" & chr(34) & "A\" & chr(34) & "/DELBA", "AD SANJOANENSE
\A\/DELBA")
    str = Replace(str, "AD SANJOANENSE \" & chr(34) & "B\" & chr(34) & "/DELBA", "AD SANJOANENSE
\B\/DELBA")
    str = Replace(str, "AD Sanjoanense \" & chr(34) & "A\" & chr(34), "AD Sanjoanense \A\")
    str = Replace(str, "AD Sanjoanense \" & chr(34) & "B\" & chr(34), "AD Sanjoanense \B\")
    str = Replace(str, "FC PORTO  \" & chr(34) & "A\" & chr(34), "FC PORTO  \A\")
    str = Replace(str, "FC PORTO  \" & chr(34) & "B\" & chr(34), "FC PORTO  \B\")
    str = Replace(str, "FC PORTO \" & chr(34) & "A\" & chr(34), "FC PORTO  \A\")
    str = Replace(str, "FC PORTO \" & chr(34) & "B\" & chr(34), "FC PORTO  \B\")
    str = Replace(str, "SIM PORTO SALVO\" & chr(34) & "A\" & chr(34), "SIM PORTO SALVO\A\")
    str = Replace(str, "SIM PORTO SALVO\" & chr(34) & "B\" & chr(34), "SIM PORTO SALVO\B\")
    str = Replace(str, "GINASIO C SUL\" & chr(34) & "A\" & chr(34), "GINASIO C SUL\A\")
    str = Replace(str, "GINASIO C SUL\" & chr(34) & "B\" & chr(34), "GINASIO C SUL\B\")
    str = Replace(str, "AD AMARANTE \" & chr(34) & "A\" & chr(34), "AD AMARANTE \A\")
    str = Replace(str, "AD AMARANTE \" & chr(34) & "B\" & chr(34), "AD AMARANTE \B\")
    str = Replace(str, "POVOA ANDEBOL CLUBE / BODEGÃO \" & chr(34) & "A\" & chr(34), "POVOA ANDEBOL
CLUBE / BODEGÃO \A\")
    str = Replace(str, "POVOA ANDEBOL CLUBE / BODEGÃO \" & chr(34) & "B\" & chr(34), "POVOA ANDEBOL
CLUBE / BODEGÃO \B\")
    str = Replace(str, "Estrela e Vigorosa Sport \" & chr(34) & "A\" & chr(34), "Estrela e Vigorosa Sport \A\")
    str = Replace(str, "Estrela e Vigorosa Sport \" & chr(34) & "B\" & chr(34), "Estrela e Vigorosa Sport \B\")
    str = Replace(str, "ESTRELA VIGOROSA SPORT \" & chr(34) & "A\" & chr(34), "ESTRELA VIGOROSA SPORT
\A\")
    str = Replace(str, "ESTRELA VIGOROSA SPORT \" & chr(34) & "B\" & chr(34), "ESTRELA VIGOROSA SPORT
\B\")
    str = Replace(str, "FC GAIA \" & chr(34) & "A\" & chr(34), "FC GAIA \A\")
    str = Replace(str, "FC GAIA \" & chr(34) & "B\" & chr(34), "FC GAIA \B\")
    str = Replace(str, "FC GAIA \" & chr(34) & "C\" & chr(34), "FC GAIA \C\")
    str = Replace(str, "SL BENFICA\" & chr(34) & "A\" & chr(34), "SL BENFICA\A\")
    str = Replace(str, "SL BENFICA\" & chr(34) & "B\" & chr(34), "SL BENFICA\B\")
    str = Replace(str, "ACD ACD \" & chr(34) & " O SOTÃO \" & chr(34) & " / TIT 15", "ACD ACD \ O SOTÃO \ / TIT
15")
    str = Replace(str, "ACD \" & chr(34) & " O SOTÃO \" & chr(34) & " / TIT ZÁAS", "ACD \ O SOTÃO \ / TIT ZÁAS")
    str = Replace(str, "ACD ACD \" & chr(34) & " O SOTÃO \" & chr(34) & " / TIT ESTRELAS", "ACD ACD \ O
SOTÃO \ / TIT ESTRELAS")
    str = Replace(str, "CF SASSOEIROS \" & chr(34) & "A\" & chr(34), "CF SASSOEIROS \A\")
    str = Replace(str, "CF SASSOEIROS \" & chr(34) & "B\" & chr(34), "CF SASSOEIROS \B\")
    str = Replace(str, "ND SANTA JOANA - MAIA \" & chr(34) & "A\" & chr(34), "ND SANTA JOANA - MAIA \A\")
    str = Replace(str, "ND SANTA JOANA - MAIA \" & chr(34) & "B\" & chr(34), "ND SANTA JOANA - MAIA \B\")
    str = Replace(str, "BECA \" & chr(34) & "A\" & chr(34), "BECA \A\")
    str = Replace(str, "BECA \" & chr(34) & "B\" & chr(34), "BECA \B\")
    str = Replace(str, "B.E.C.A. \" & chr(34) & "A\" & chr(34), "BECA \A\")
    str = Replace(str, "B.E.C.A. \" & chr(34) & "B\" & chr(34), "BECA \B\")
    str = Replace(str, "ARSENAL C. DEVESA \" & chr(34) & "A\" & chr(34), "ARSENAL C. DEVESA \A\")
    str = Replace(str, "ARSENAL C. DEVESA \" & chr(34) & "B\" & chr(34), "ARSENAL C. DEVESA \B\")
    str = Replace(str, "GC SANTO TIRSO \" & chr(34) & "A\" & chr(34), "GC SANTO TIRSO \A\")
    str = Replace(str, "GC SANTO TIRSO \" & chr(34) & "B\" & chr(34), "GC SANTO TIRSO \B\")
    str = Replace(str, "GC SANTO TIRSO  \" & chr(34) & "A\" & chr(34), "GC SANTO TIRSO \A\")
    str = Replace(str, "GC SANTO TIRSO  \" & chr(34) & "B\" & chr(34), "GC SANTO TIRSO \B\")
    str = Replace(str, "GC SANTO TIRSO  \" & chr(34) & "C\" & chr(34), "GC SANTO TIRSO \C\")
    str = Replace(str, "SPORTING CP\" & chr(34) & "A\" & chr(34), "SPORTING CP\A\")
    str = Replace(str, "SPORTING CP\" & chr(34) & "B\" & chr(34), "SPORTING CP\B\")
    str = Replace(str, "GM 1º DEZEMBRO\" & chr(34) & "A\" & chr(34), "GM 1º DEZEMBRO\A\")
    str = Replace(str, "GM 1º DEZEMBRO\" & chr(34) & "B\" & chr(34), "GM 1º DEZEMBRO\B\")
    str = Replace(str, "CA LEÇA \" & chr(34) & "A\" & chr(34), "CA LEÇA \A\")
    str = Replace(str, "CA LEÇA \" & chr(34) & "B\" & chr(34), "CA LEÇA \B\")
    str = Replace(str, "CA Leça \" & chr(34) & "A\" & chr(34), "CA LEÇA \A\")
    str = Replace(str, "CA Leça \" & chr(34) & "B\" & chr(34), "CA LEÇA \B\")
    str = Replace(str, "ABC BRAGA \" & chr(34) & "A\" & chr(34), "ABC BRAGA \A\")
    str = Replace(str, "ABC BRAGA \" & chr(34) & "B\" & chr(34), "ABC BRAGA \B\")
    str = Replace(str, "CCR FERMENTOES \" & chr(34) & "A\" & chr(34), "CCR FERMENTOES \A\")
    str = Replace(str, "CCR FERMENTOES \" & chr(34) & "B\" & chr(34), "CCR FERMENTOES \B\")
    str = Replace(str, "CCR FERMENTOES \" & chr(34) & "C\" & chr(34), "CCR FERMENTOES \C\")
    str = Replace(str, "CCR FERMENTÕES \" & chr(34) & "A\" & chr(34), "CCR FERMENTÕES \A\")
    str = Replace(str, "CCR FERMENTÕES \" & chr(34) & "B\" & chr(34), "CCR FERMENTÕES \B\")
    str = Replace(str, "CCR FERMENTÕES \" & chr(34) & "C\" & chr(34), "CCR FERMENTÕES \C\")
    str = Replace(str, "FERMENTÕES \" & chr(34) & "A\" & chr(34), "FERMENTÕES \A\")
    str = Replace(str, "FERMENTÕES \" & chr(34) & "B\" & chr(34), "FERMENTÕES \B\")
    str = Replace(str, "FERMENTÕES \" & chr(34) & "C\" & chr(34), "FERMENTÕES \C\")
    str = Replace(str, "CJ A. GARRETT \" & chr(34) & "A\" & chr(34), "CJ A. GARRETT \A\")
    str = Replace(str, "CJ A. GARRETT \" & chr(34) & "B\" & chr(34), "CJ A. GARRETT \B\")
    str = Replace(str, "NAAL PASSOS MANUEL\" & chr(34) & "A\" & chr(34), "NAAL PASSOS MANUEL\A\")
    str = Replace(str, "NAAL PASSOS MANUEL\" & chr(34) & "B\" & chr(34), "NAAL PASSOS MANUEL\B\")
    str = Replace(str, "SC Espinho \" & chr(34) & "A\" & chr(34), "SC Espinho \A\")
    str = Replace(str, "SC Espinho \" & chr(34) & "B\" & chr(34), "SC Espinho \B\")
    str = Replace(str, "SC ESPINHO \" & chr(34) & "A\" & chr(34), "SC Espinho \A\")
    str = Replace(str, "SC ESPINHO \" & chr(34) & "B\" & chr(34), "SC Espinho \B\")
    str = Replace(str, "SC ESPINHO \" & chr(34) & "C\" & chr(34), "SC Espinho \C\")
    str = Replace(str, "CD Feirense \" & chr(34) & "A\" & chr(34), "CD Feirense \A\")
    str = Replace(str, "CD Feirense \" & chr(34) & "B\" & chr(34), "CD Feirense \B\")
    str = Replace(str, "CD Feirense \" & chr(34) & "C\" & chr(34), "CD Feirense \C\")
    str = Replace(str, "CD FEIRENSE \" & chr(34) & "A\" & chr(34), "CD Feirense \A\")
    str = Replace(str, "CD FEIRENSE \" & chr(34) & "B\" & chr(34), "CD Feirense \B\")
    str = Replace(str, "CD FEIRENSE \" & chr(34) & "C\" & chr(34), "CD Feirense \C\")
    str = Replace(str, "ALAVARIUM / LOVE TILES \" & chr(34) & "A\" & chr(34), "ALAVARIUM / LOVE TILES
\A\")
    str = Replace(str, "Alavarium Love Tiles \" & chr(34) & "B\" & chr(34), "Alavarium Love Tiles \B\")
    str = Replace(str, "Alavarium \" & chr(34) & "A\" & chr(34), "Alavarium \A\")
    str = Replace(str, "Alavarium \" & chr(34) & "B\" & chr(34), "Alavarium \B\")
    str = Replace(str, "ALAVARIUM AC \" & chr(34) & "A\" & chr(34), "Alavarium \A\")
    str = Replace(str, "ALAVARIUM AC \" & chr(34) & "B\" & chr(34), "Alavarium \B\")
    str = Replace(str, "Alavarium AC \" & chr(34) & "A\" & chr(34), "Alavarium \A\")
    str = Replace(str, "Alavarium AC \" & chr(34) & "B\" & chr(34), "Alavarium \B\")
    str = Replace(str, "CDE BOA ÁGUA ANDEBOL \" & chr(34) & "A\" & chr(34), "CDE BOA ÁGUA ANDEBOL
\A\")
    str = Replace(str, "CDE BOA ÁGUA ANDEBOL \" & chr(34) & "B\" & chr(34), "CDE BOA ÁGUA ANDEBOL
\B\")
    str = Replace(str, "ABC ANDEBOL SAD \" & chr(34) & "A\" & chr(34), "ABC ANDEBOL SAD \A\")
    str = Replace(str, "ABC ANDEBOL SAD \" & chr(34) & "B\" & chr(34), "ABC ANDEBOL SAD \B\")
    str = Replace(str, "ABC ANDEBOL SAD \" & chr(34) & "C\" & chr(34), "ABC ANDEBOL SAD \C\")
    str = Replace(str, "CE Levada \" & chr(34) & "A\" & chr(34), "CE Levada \A\")
    str = Replace(str, "CE Levada \" & chr(34) & "B\" & chr(34), "CE Levada \B\")
    str = Replace(str, "AC Sismaria \" & chr(34) & "A\" & chr(34), "AC Sismaria \A\")
    str = Replace(str, "AC Sismaria \" & chr(34) & "B\" & chr(34), "AC Sismaria \B\")
    str = Replace(str, "CD S Bernardo \" & chr(34) & "B\" & chr(34), "CD S Bernardo \B\")
    str = Replace(str, "CD S Bernardo \" & chr(34) & "A\" & chr(34), "CD S Bernardo \A\")
    str = Replace(str, "CD S.Bernardo \" & chr(34) & "A\" & chr(34), "CD S Bernardo \A\")
    str = Replace(str, "CD S.Bernardo \" & chr(34) & "B\" & chr(34), "CD S Bernardo \B\")
    str = Replace(str, "CD S. Bernardo \" & chr(34) & "A\" & chr(34), "CD S Bernardo \A\")
    str = Replace(str, "CD S. Bernardo \" & chr(34) & "B\" & chr(34), "CD S Bernardo \B\")
    str = Replace(str, "Escola Formaçao Espinho - Os Tigres \" & chr(34) & "A\" & chr(34), "Escola Formaçao Espinho -
Os Tigres \A\")
    str = Replace(str, "Escola Formaçao Espinho - Os Tigres \" & chr(34) & "B\" & chr(34), "Escola Formaçao Espinho -
Os Tigres \B\")
    str = Replace(str, "A Ac. Espinho \" & chr(34) & "A\" & chr(34), "A Ac. Espinho \A\")
    str = Replace(str, "A Ac. Espinho \" & chr(34) & "B\" & chr(34), "A Ac. Espinho \B\")
    str = Replace(str, "ADA Canelas \" & chr(34) & "B\" & chr(34), "ADA Canelas \B\")
    str = Replace(str, "ADA Canelas \" & chr(34) & "A\" & chr(34), "ADA Canelas \A\")
    str = Replace(str, "ADA Canelas \" & chr(34) & "C\" & chr(34), "ADA Canelas \C\")
    str = Replace(str, "ADA canelas \" & chr(34) & "A\" & chr(34), "ADA Canelas \A\")
    str = Replace(str, "ADA canelas \" & chr(34) & "B\" & chr(34), "ADA Canelas \B\")
    str = Replace(str, "ADA canelas \" & chr(34) & "C\" & chr(34), "ADA Canelas \C\")
    str = Replace(str, "AC FAFE \" & chr(34) & "A\" & chr(34), "AC FAFE \A\")
    str = Replace(str, "AC FAFE \" & chr(34) & "B\" & chr(34), "AC FAFE \B\")
    str = Replace(str, "ACD Monte \" & chr(34) & "A\" & chr(34), "ACD Monte \A\")
    str = Replace(str, "ACD Monte \" & chr(34) & "B\" & chr(34), "ACD Monte \B\")
    str = Replace(str, "CF OS BELENENSES\" & chr(34) & "A\" & chr(34), "CF OS BELENENSES \A\")
    str = Replace(str, "CF OS BELENENSES\" & chr(34) & "B\" & chr(34), "CF OS BELENENSES \B\")
    str = Replace(str, "CF OS BELENENSES \" & chr(34) & "A\" & chr(34), "CF OS BELENENSES \A\")
    str = Replace(str, "CF OS BELENENSES \" & chr(34) & "B\" & chr(34), "CF OS BELENENSES \B\")
    str = Replace(str, "CDC S.Paio de Oleiros \" & chr(34) & "A\" & chr(34), "CDC S.Paio de Oleiros \A\")
    str = Replace(str, "CDC S.Paio de Oleiros \" & chr(34) & "B\" & chr(34), "CDC S.Paio de Oleiros \B\")
    str = Replace(str, "ALMADA AC\" & chr(34) & "A\" & chr(34), "ALMADA AC \A\")
    str = Replace(str, "ALMADA AC \" & chr(34) & "B\" & chr(34), "ALMADA AC \B\")
    str = Replace(str, "CP Valongo do Vouga \" & chr(34) & "A\" & chr(34), "CP Valongo do Vouga \A\")
    str = Replace(str, "CP Valongo do Vouga \" & chr(34) & "B\" & chr(34), "CP Valongo do Vouga \B\")
    str = Replace(str, "CD MAFRA / HELIA CORREIA\" & chr(34) & "A\" & chr(34), "CD MAFRA / HELIA
CORREIA\A\")
    str = Replace(str, "CD MAFRA / HELIA CORREIA\" & chr(34) & "B\" & chr(34), "CD MAFRA / HELIA
CORREIA\B\")
    str = Replace(str, "CD MAFRA\" & chr(34) & "A\" & chr(34), "CD MAFRA\A\")
    str = Replace(str, "CD MAFRA\" & chr(34) & "B\" & chr(34), "CD MAFRA\B\")
    str = Replace(str, "CCD S.Paio Oleiros \" & chr(34) & "A\" & chr(34), "CCD S.Paio Oleiros \A\")
    str = Replace(str, "CCD S.Paio Oleiros \" & chr(34) & "B\" & chr(34), "CCD S.Paio Oleiros \B\")
    str = Replace(str, "BOA HORA FC /ROFF\" & chr(34) & "A\" & chr(34), "BOA HORA FC /ROFF\A\")
    str = Replace(str, "BOA HORA FC /ROFF\" & chr(34) & "B\" & chr(34), "BOA HORA FC /ROFF\B\")
    str = Replace(str, "BOA HORA FC / ROFF\" & chr(34) & "A\" & chr(34), "BOA HORA FC /ROFF\A\")
    str = Replace(str, "BOA HORA FC / ROFF\" & chr(34) & "B\" & chr(34), "BOA HORA FC /ROFF\B\")
    str = Replace(str, "CD XICO ANDEBOL \" & chr(34) & "A\" & chr(34), "CD XICO ANDEBOL \A\")
    str = Replace(str, "CD XICO ANDEBOL \" & chr(34) & "B\" & chr(34), "CD XICO ANDEBOL \B\")
    str = Replace(str, "CD XICO ANDEBOL \" & chr(34) & "C\" & chr(34), "CD XICO ANDEBOL \C\")
    str = Replace(str, "AC CACEM\" & chr(34) & "A\" & chr(34), "AC CACEM\A\")
    str = Replace(str, "AC CACEM\" & chr(34) & "B\" & chr(34), "AC CACEM\B\")
    str = Replace(str, "LOS PRIMOS\" & chr(34) & "A\" & chr(34), "LOS PRIMOS\A\")
    str = Replace(str, "LOS PRIMOS\" & chr(34) & "B\" & chr(34), "LOS PRIMOS\B\")
    str = Replace(str, "ND SANTA JOANA - MAIA - \" & chr(34) & "A\" & chr(34), "ND SANTA JOANA - MAIA -
\A\")
    str = Replace(str, "ND SANTA JOANA - MAIA - \" & chr(34) & "B\" & chr(34), "ND SANTA JOANA - MAIA -
\B\")
    str = Replace(str, "CR B º JANEIRO\" & chr(34) & "A\" & chr(34), "CR B º JANEIRO\A\")
    str = Replace(str, "CR B º JANEIRO\" & chr(34) & "B\" & chr(34), "CR B º JANEIRO\B\")
    str = Replace(str, "ADA MAIA / ISMAI - 5M \" & chr(34) & "A\" & chr(34), "ADA MAIA / ISMAI - 5M \A\")
    str = Replace(str, "ADA MAIA / ISMAI - 5M \" & chr(34) & "B\" & chr(34), "ADA MAIA / ISMAI - 5M \B\")
    str = Replace(str, "CCR ALTO MOINHO \" & chr(34) & "A\" & chr(34), "CCR ALTO MOINHO \A\")
    str = Replace(str, "CCR ALTO MOINHO \" & chr(34) & "B\" & chr(34), "CCR ALTO MOINHO \B\")
    str = Replace(str, "CCR ALTO MOINHO\" & chr(34) & "A\" & chr(34), "CCR ALTO MOINHO \A\")
    str = Replace(str, "CCR ALTO MOINHO\" & chr(34) & "B\" & chr(34), "CCR ALTO MOINHO \B\")
    str = Replace(str, "CD Rio Tinto Minis 5 \" & chr(34) & "A\" & chr(34), "CD Rio Tinto Minis 5 \A\")
    str = Replace(str, "CD Rio Tinto Minis 5 \" & chr(34) & "B\" & chr(34), "CD Rio Tinto Minis 5 \B\")
    str = Replace(str, "UAS - Beach Handball \" & chr(34) & "A\" & chr(34), "UAS - Beach Handball \A\")
    str = Replace(str, "UAS - Beach Handball \" & chr(34) & "B\" & chr(34), "UAS - Beach Handball \B\")
    str = Replace(str, "Ilhavo AC \" & chr(34) & "A\" & chr(34), "Ilhavo AC \A\")
    str = Replace(str, "Ilhavo AC \" & chr(34) & "B\" & chr(34), "Ilhavo AC \B\")
    str = Replace(str, "ACADÉMICO FC (minis 5) \" & chr(34) & "A\" & chr(34), "ACADÉMICO FC (minis 5) \A\")
    str = Replace(str, "ACADÉMICO FC (minis 5) \" & chr(34) & "B\" & chr(34), "ACADÉMICO FC (minis 5) \B\")
    str = Replace(str, "A. A. AVANCA \" & chr(34) & "A\" & chr(34), "A. A. AVANCA \A\")
    str = Replace(str, "A. A. AVANCA \" & chr(34) & "B\" & chr(34), "A. A. AVANCA \B\")
    str = Replace(str, "CD Rio Tinto Minis 5 \" & chr(34) & "A\" & chr(34), "CD Rio Tinto Minis 5 \A\")
    str = Replace(str, "CD Rio Tinto Minis 5 \" & chr(34) & "B\" & chr(34), "CD Rio Tinto Minis 5 \B\")
    str = Replace(str, "FC INFESTA \" & chr(34) & "A\" & chr(34), "FC INFESTA \A\")
    str = Replace(str, "FC INFESTA \" & chr(34) & "B\" & chr(34), "FC INFESTA \B\")
    str = Replace(str, "FC INFESTA \" & chr(34) & "C\" & chr(34), "FC INFESTA \C\")
    str = Replace(str, "AC OLIVEIRA FRADES \" & chr(34) & "A\" & chr(34), "AC OLIVEIRA FRADES \A\")
    str = Replace(str, "AC OLIVEIRA FRADES \" & chr(34) & "B\" & chr(34), "AC OLIVEIRA FRADES \B\")
    str = Replace(str, "AC OLIVEIRA FRADES \" & chr(34) & "JUNIORES\" & chr(34), "AC OLIVEIRA FRADES
\JUNIORES\")
    str = Replace(str, "C.D.Feirense \" & chr(34) & "A\" & chr(34), "C.D.Feirense \A\")
    str = Replace(str, "C.D.Feirense \" & chr(34) & "B\" & chr(34), "C.D.Feirense \B\")
    str = Replace(str, "ACADEMICO VISEU FC \" & chr(34) & "A\" & chr(34), "ACADEMICO VISEU FC \A\")
    str = Replace(str, "ACADEMICO VISEU FC \" & chr(34) & "B\" & chr(34), "ACADEMICO VISEU FC \B\")
    str = Replace(str, "Académico do Funchal \" & chr(34) & "B\" & chr(34), "Académico do Funchal \B\")
    str = Replace(str, "Académico do Funchal \" & chr(34) & "A\" & chr(34), "Académico do Funchal \A\")
    str = Replace(str, "CD B. Perestrelo \" & chr(34) & "B\" & chr(34), "CD B. Perestrelo \B\")
    str = Replace(str, "CD B. Perestrelo \" & chr(34) & "A\" & chr(34), "CD B. Perestrelo \A\")


    'Loop each JSON columns to save to Cols collection
    For chrIndex = 3 To Len(str)
        chrJson = Mid(str, chrIndex, 1)

        'chr(34) is quotation mark (‘’)
        If chrJson = chr(34) Then
            countAspas = countAspas + 1

            'Condition to save string
            If countAspas = 3 Or (countAspas - 3) Mod 4 = 0 Then
                saveChr = True

            'Condition to stop saving string and add to Cols colletion (refering to data)
            ElseIf countAspas = 4 Or (countAspas - 4) Mod 4 = 0 Then
                saveChr = False
                'save string into the collection
                Cols.Add colName
                'clean variable responsible to insert string into collection
                colName = ""

            End If
        'To build the word from all characters of data string
        ElseIf saveChr Then
            colName = colName & chrJson

        End If
    Next

End Function
```

VBA code function ‘SaveColumnsToCollection’:

```
Function SaveColumnsToCollection(str As String, ByRef Cols As Collection)
    Dim chrIndex&, countAspas As Long
    Dim chrJson$, colName As String
    Dim saveChr As Boolean

    'Loop each JSON columns to save to Cols collection
    For chrIndex = 1 To Len(str)
        chrJson = Mid(str, chrIndex, 1)

        If chrJson = chr(34) Then
            countAspas = countAspas + 1

            'Condition to save string
            If countAspas = 1 Or (countAspas - 1) Mod 4 = 0 Then
                saveChr = True

            'Condition to stop saving string and add to Cols colletion
            ElseIf countAspas = 2 Or (countAspas - 2) Mod 4 = 0 Then
                saveChr = False
                If colName = "_id" Then
                    Cols.Add colName
                    Exit Function
                End If
                If Functions.InCollection(Cols, colName) Then Exit Function
                Cols.Add colName
                colName = ""

            End If
        ElseIf saveChr Then
            colName = colName & chrJson

        End If
    Next

End Function
```

VBA code program ‘ExtractJSON()’:

```
Sub ExtractJSON()
    Dim shtJson$, shtHistory$, shtRegistrations$, id$, strHistory$, _
    strRegistrations$, colStrHistory$, colStrRegistrations As String
    Dim lastRow&, row&, nRows&, lastRowWithData&, rowAdd&, count As Long
    Dim colHistory%, col%, colRegistrations%, historyCols%, registrationsCols As Integer 'dimensioning variables
    Dim ColsHistory As New Collection, ColsRegistrations As New Collection
    Dim DataHistory As New Collection, DataRegistrations As New Collection
    Dim cel As Range, dataRange As Range

    'variables definition
    shtJson = "Folha 1 - players"
    shtHistory = "Folha History"
    shtRegistrations = "Folha Registrations"
    'first line with an array in order to retrieve the structure (columns)
    row = 32

    With ThisWorkbook
     'save the columns inside of collection
       With .Sheets(shtJson)
          colHistory = .Range("R" & 1).Column
          colRegistrations = .Range("V" & 1).Column
          'passing data to collection
          Call SaveJSONtoCollection.SaveColumnsToCollection(.Cells(row, colHistory).Value, ColsHistory)
```

```
        Call SaveJSONtoCollection.SaveColumnsToCollection(.Cells(row, colRegistrations).Value, ColsRegistrations)
        historyCols = ColsHistory.count
        'inside of with the prupose is to count the number of columns inside the array
        registrationsCols = ColsRegistrations.count
    End With

    'create sheet if not existing
    If Not Functions.WorksheetExists(shtHistory, ThisWorkbook) Then .Sheets.Add(After:=Sheets(shtJson)).Name = shtHistory
    If Not Functions.WorksheetExists(shtRegistrations, ThisWorkbook) Then .Sheets.Add(After:=Sheets(shtHistory)).Name =
shtRegistrations

    'inserting the columns inside sheet in line 1
    With .Sheets(shtHistory)
        If Len(.Cells(1, 1).Value) < 1 Then
            .Cells(1, 1).Value = "ID"
            For col = 1 To historyCols
                .Cells(1, col + 1).Value = ColsHistory(col)
            Next col
        End If
    End With

    With .Sheets(shtRegistrations)
        If Len(.Cells(1, 1).Value) < 1 Then
            .Cells(1, 1).Value = "ID"
            For col = 1 To registrationsCols
                .Cells(1, col + 1).Value = ColsRegistrations(col)
            Next col
        End If
    End With

    'obtain the last line with data
    'column 2 was chosen because doesn't have null values
    lastRow = .Sheets(shtJson).Cells(Rows.count, 2).End(xlUp).row

    For row = 3 To lastRow
    'save the data inside of collection
        With .Sheets(shtJson)
            id = .Cells(row, 2).Value
            strHistory = .Cells(row, colHistory).Value
            strRegistrations = .Cells(row, colRegistrations).Value
            Call SaveJSONtoCollection.SaveDataToCollection(.Cells(row, colHistory).Value, DataHistory)
            Call SaveJSONtoCollection.SaveDataToCollection(.Cells(row, colRegistrations).Value, DataRegistrations)

        End With

        'for string history with data
        If Len(strHistory) > 2 Then
            With .Sheets(shtHistory)
             'count of "gameNumber"
                nRows = (Len(strHistory) - Len(Replace$(strHistory, "gameNumber", ""))) / Len("gameNumber")
                lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row

                'add ID in the rows that will be fullfilled
                For rowAdd = lastRowWithData + 1 To lastRowWithData + nRows
                    .Cells(rowAdd, 1).Value = id
                Next rowAdd

                'transform last column number into letter
                colStrHistory = Split(Cells(1, historyCols + 1).Address, "$")(1)
                'select the place where we will copy our data
                Set dataRange = .Range("B" & lastRowWithData + 1 & ":" & colStrHistory & lastRowWithData + nRows)

                'to cover all cells to datarange
                count = 0
                For Each cel In dataRange
                    count = count + 1
                    cel.Value = DataHistory(count)
                Next cel
            End With
        End If

        If Len(strRegistrations) > 2 Then
```

```
            With .Sheets(shtRegistrations)
                nRows = (Len(strRegistrations) - Len(Replace$(strRegistrations, "clubId", ""))) / Len("clubId")
                lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row

                For rowAdd = lastRowWithData + 1 To lastRowWithData + nRows
                    .Cells(rowAdd, 1).Value = id
                Next rowAdd

                colStrRegistrations = Split(Cells(1, registrationsCols + 1).Address, "$")(1)
                Set dataRange = .Range("B" & lastRowWithData + 1 & ":" & colStrRegistrations & lastRowWithData + nRows)

                count = 0
                For Each cel In dataRange
                    count = count + 1
                    cel.Value = DataRegistrations(count)
                Next cel
            End With
        End If

        Set DataHistory = Nothing
        Set DataRegistrations = Nothing
    Next row

   'creation of Yellowcard? column
   With .Sheets(shtHistory)
        .Range("P1").Value = "Yellowcard?"
        lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
        .Range("P2:P" & lastRowWithData).FormulaR1C1 = "=IF(ISBLANK(RC[-7])=TRUE,""No"",""Yes"")"

   'structure cells of table History
          With .Range("A1:P" & lastRowWithData)
             .HorizontalAlignment = xlGeneral
             .VerticalAlignment = xlTop
             .WrapText = False
             .Orientation = 0
             .AddIndent = False
             .IndentLevel = 0
             .ShrinkToFit = False
             .ReadingOrder = xlContext
             .MergeCells = False
             .EntireColumn.AutoFit
         End With

    End With

    'creation of Step and District column
    With .Sheets(shtRegistrations)
         .Range("S1").Value = "Step"
         .Range("T1").Value = "District"
         lastRowWithData = .Cells(Rows.count, 1).End(xlUp).row
         .Range("S2:S" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-10],StepVlookup!R2C1:R50C2,2,FALSE)"
         .Range("T2:T" & lastRowWithData).FormulaR1C1 = "=VLOOKUP(RC[-13],DistrictVlookup!R2C1:R42C2,2,FALSE)"

  'structure cells of table Registrations
          With .Range("A1:T" & lastRowWithData)
             .HorizontalAlignment = xlGeneral
             .VerticalAlignment = xlTop
             .WrapText = False
             .Orientation = 0
             .AddIndent = False
             .IndentLevel = 0
             .ShrinkToFit = False
             .ReadingOrder = xlContext
             .MergeCells = False
             .EntireColumn.AutoFit
         End With
   End With
 End With
End Sub
```