The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document

Project: Predicting Soccer Matches Results

Author: Igor Palmieri

1 Use case

In this project we will develop and test machine learning models to predict outcomes of professional soccer matches. This is a historical relevant subject, but nowadays is being even more explored in the context of abundant sources of information and the emergence of powerful computational models.

1.1 Use case description

Prediction of general results of a soccer match can be considered a challenging problem, since are ultimately conditioned by the unrolling of multiple possible micro events that by themselves are hard to foresee (e.g. fouls committed, cards issued, individual players performance, etc); by the environment in which the match is being played; and also by unexplainable factors that some would consider pure effects of luck. Sometimes the context provides a strong clue for a match, when for example a strong team faces a weak opposition. But even then, unexpected results are possible and do happen occasionally.

To narrow the possibilities that this problem provides, and make models more focused in a single categorical target, we chose to predict the general result of the match i.e. if the home team wins, loses or the match is a draw.

Our reference case is the *Brazilian national soccer championship*, which has plenty of historical databases of matches and its teams. But we will try to rely only on most common and basic match information as input features, which can make the project useful and directly applicable to other leagues that do not have easily accessible historical data with detailed information.

1.2 Data sources

The reference datasource for this project is the “*Campeonato Brasileiro de futebol*” dataset. It contains results and some metrics for **8.405 matches between 2003 and 2023**, a period when the Brazilian national soccer championship maintained a consistent format in terms of rules and games through all the seasons.

This dataset contains 4 main files, in a csv (comma separated values) format.

- **campeonato-brasileiro-cartoes.csv:** 8 columns x 18.857 rows; contains each card (red or yellow) issued during the period. Features include referee, player, color of card, etc.
- **campeonato-brasileiro-estatisticas-full.csv:** 13 columns x 16.810 rows; contains match metrics for each team in each game (thus rows are roughly twice the number of matches in the database). Features include shots on target, passes, fouls, etc.
- **campeonato-brasileiro-full.csv:** 16 columns x 8.405 rows; the main dataset, consisting of metrics and results of each match in the period. Features include team names, goals of each team, referee, coaches, stadiums, etc.
- **campeonato-brasileiro-gols.csv:** 6 columns x 8.932 rows; contains details of every goal scored. Features include player name, team, time of the game, etc.

More information and access to the data can be found on the following links:

Kaggle: https://www.kaggle.com/datasets/adaoduque/campeonato-brasileiro-de-futebol

Github: https://github.com/adaoduque/Brasileirao_Dataset

1.3 Data repository

Since data is provided as *.csv* files, there is no need for a specific data repository. **All the useful files are downloaded directly from the GitHub project and read into dataframes during runtime.**

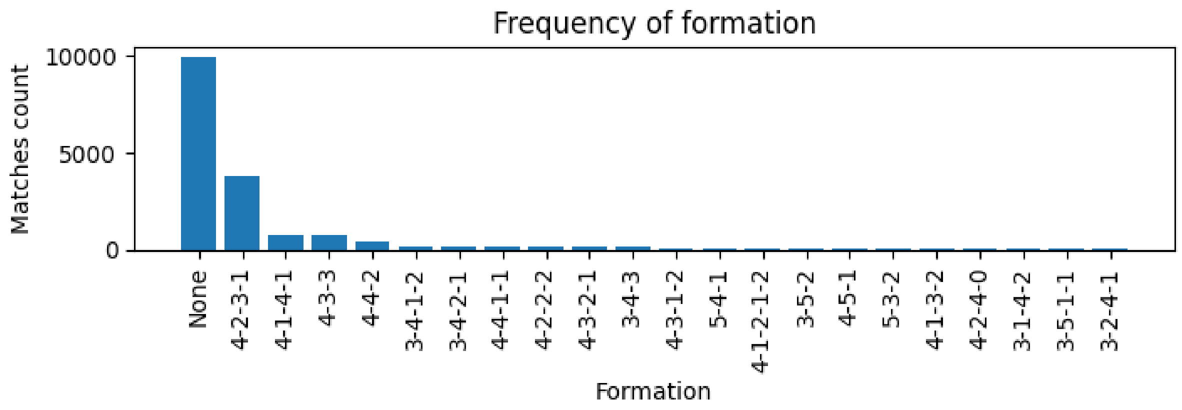
Besides containing a lot of useful information and statistics, this is not a database that requires plenty storage resources, using dozens of megabytes at maximum. Thus, this strategy allows us to simplify the process of data retrieval and the use of cloud infrastructure.

2 Solution and methods

2.1 Data quality assessment

The first step in this project is to explore the content of each dataset, and define which information can be directly used, and which need correction and adaptation before being considered in the next stages. This includes plotting data points and the categories distributions. A first analysis shows that some typos can be easiy corrected, and that some fields

A good example is the '*formation*' field. We can see from the notebook that it contains a lot of *None* (empty) values. Since it is a valuable information in a match, we have two options: either discard rows witch with no values in this field, or fill these cases with some arbitrary value.



Briefly, the data quality assessment process is summarized as follows:

- Plot distributions of classes or values of each column
- Count number of missing values (e.g. Null or N/A)
- Calculate outliers and assess if they are of interest
- Fix typos and errors in strings to consolidate categories

At this stage, we decided to focus on the **campeonato-brasileiro-full** dataset for a first iteration of this project. It contains a good number of features that are useful and with a reasonable level of quality.

Below, a list of fields which were assessed in terms of quality and usefullness for the project. More details of each case can be found in the project's notebook.

- Home and away team names
- Teams coach names
- Teams formation in the match
- Match stadium name and location
- Match scores
- Match final result (win, draw, loss)

This is a basic set of information to kick start the process. More fields, even from other tables, can be added later to the project if it is convenient.

2.2 Feature engineering

The process of feature engineering is contained in Sections 3 (Feature Generation) and 4 (Feature Encoding) of the notebook. The following features are the result of this process:

2.2.1) Series of overall past results: this is a group of multiple features with the common characteristic that each contains a time series of past values of a selected variable. This allow any model to receive as input a broader context that includes short-term performance of each team. The generated features are

scored_mandante: *number of goals scored by the home team*
scored_visitante: *number of goals scored by the away team*
against_mandante: *number of goals conceded by the home team*
against_visitante: *number of goals conceded by the away team*
results_mandante: *match results (win, draw, loss) of the home team*
results_visitante: *match results (win, draw, loss) of the away team*

Each of these variables is included with N_p data points, which means that the will generate N_p new features, with values of the last match till the N_p previous match, of the each team.

We defined $N_p = 5$, a value chosen with a good compromise between complexity and impact on result. Thus, N_p*(6 series) = **30 features are generated by this process.** This parameter can be further explored to enhance results of simplify the model training process.

2.2.2) Series of past results between teams: here the same logic of the overall past results features applies. In this case, however, we include only past values of matches between the home and away team, in any order. This gives a short-term context of previous games between the two teams. The following data is included:

scored_past: *number of goals scored by the home team*
against_past: *number of goals conceded by the home team*
results_past: *match results (win, draw, loss) of the home team*

Note that we don't need to include the respective variables for the away team, because it is already represented by the opposite variable of the home team (e.g. the conceded goals of the away team is the scored goals of the home team).

Here, the length of the time series is represented by the parameter N_v. We define $N_v = 3$, which gives us N_v*3 = ***9 new features***.

In total, we added 30+9=39 new features representing time series of previous data for each match.

Besides that, we also engineer new features by encoding original dataframe columns in a way that better represents the information, and that will allow machine learning models to use it appropriately.

2.2.3) Feature encoding: below, we describe how some features are encoded and the respective resulting features:

- team names: one-hot encoding of each team name. We also tested the use of combining the away and home teams into a single vector by a weighted sum of the two vectors, and by concatenating them. Eventually, the best results were achieved with 2 separated one-hot encoded vectors (home and away).

- other categorical variables: this includes 7 variables (*home team formation, away team formation, home team coach, away team coach, stadium, home team state and away team state). Each one is encoded using a* ***string indexer*** *encoder.*

- target encoding*:* we also encode the result of the match, which is our target variable to be predicted, into 3 possible values: 2 (win), 1 (loss) and 0 (draw). This scheme is also used in any other features that represent past results.

For reference, this is a list of all features that compose the final dataframe and their respective descriptions:

scored_mandante_d1: number of goals scored by the home team in its last match

scored_mandante_d2: number of goals scored by the home team in its 2nd to last match

scored_mandante_d3: number of goals scored by the home team in its 3rd to last match

scored_mandante_d4: number of goals scored by the home team in its 4th to last match

scored_mandante_d5: number of goals scored by the home team in its 5th to last match

scored_visitante_d1: number of goals scored by the away team in its last match

scored_visitante_d2: number of goals scored by the away team in its 2nd to last match

scored_visitante_d3: number of goals scored by the away team in its 3rd to last match

scored_visitante_d4: number of goals scored by the away team in its 4th to last match

scored_visitante_d5: number of goals scored by the away team in its 5th to last match

against_mandante_d1: number of goals conceded by the home team in its last match

against_mandante_d2: number of goals conceded by the home team in its 2nd to last match

against_mandante_d3: number of goals conceded by the home team in its 3rd to last match

against_mandante_d4: number of goals conceded by the home team in its 4th to last match

against_mandante_d5: number of goals conceded by the home team in its 5th to last match

against_visitante_d1: number of goals conceded by the away team in its last match

against_visitante_d2: number of goals conceded by the away team in its 2nd to last match

against_visitante_d3: number of goals conceded by the away team in its 3rd to last match

against_visitante_d4: umber of goals conceded by the away team in its 4th to last match

against_visitante_d5: number of goals conceded by the away team in its 5th to last match

scored_past_d1: number of goals scored by the home team in the last match that both teams played each other

scored_past_d2: number of goals scored by the home team in the 2nd to last match that both teams played each other

scored_past_d3: number of goals scored by the home team in the 3rd to last match that both teams played each other

against_past_d1: number of goals conceded by the home team in the last match that both teams played each other

against_past_d2: umber of goals conceded by the home team in the 2nd to last match that both teams played each other

against_past_d3: umber of goals conceded by the home team in the 3rd to last match that both teams played each other

home_team: one-hot encoded vector representing the home team name

away_team: one-hot encoded vector representing the away team name

formacao_mandante_index: index encoding the home team formation scheme

formacao_visitante_index: index encoding the away team formation scheme

tecnico_mandante_index: index encoding the home team coach name

tecnico_visitante_index: index encoding the away team coach name

arena_index: index encoding the stadium (venue) name

mandante_Estado_index: index encoding the home team state of origin

visitante_Estado_index: index encoding the away team state of origin

results_mandante_d1: result of the last match of the home team

results_mandante_d2: result of the 2nd to last match of the home team

results_mandante_d3: result of the 3rt to last match of the home team

results_mandante_d4: result of the 4th to last match of the home team

results_mandante_d5: result of the 5th to last match of the home team

results_visitante_d1: result of the last match of the away team

results_visitante_d2: result of the 2nd to last match of the away team

results_visitante_d3: result of the 3rt to last match of the away team

results_visitante_d4: result of the 4th to last match of the away team

results_visitante_d5: result of the 5th to last match of the away team

results_past_d1: result of the last match between both teams, in respect to the home team

results_past_d2: result of the 2nd to last match between both teams, in respect to the home team

results_past_d3: result of the 3rd to last match between both teams, in respect to the home team

result: result of the match **(target variable)**

2.3 Algorithms

For the modeling process, we employed 3 different machine learning algorithms. These models were chosen because are very known in the machine learning area, and have ready to use implementations in our chosen framework. **The idea is to try all of them and choose the one with best performance as the solution for our project.**

i) Multilayer Perceptron Classifier (MLP): consisting of a neural network with concatenated layers. The most basic version contains an input layer, a hidden layer, and an output layer. Input and output layers have a number of neurons that are equal to the input and output size, respectively. In our case, we added an extra hidden layer, making a total of **5 layers** of neurons. The input layer contains **136 neurons** (the number of elements of the vectorized input); The output layer contains **3 neurons.** So, the number of neurons of all the layers is **[136, 64, 32, 3]** from input to output.

ii) Gradient Boosting Model (GBT): classic gradient boost trees algorithm. This technique combines multiple weak tree models to create a single, more accurate model iteratively till a minimum error is achieved. We used *maxBins*=256. All the other parameters were set with default values.

iii) Random Forest Classifier (RF): classic random forest algorithm. This algorithm uses multiple decision trees with random parameters each to combine their outputs into a single result. The following parameters were set: *maxBins*=256, *maxDepth*=25, *numTrees*=128.

The last two models (GBT and RF) were trained using a **One-vs-Rest approach**, since our problem is of a multiclass classification (3 possible match outcomes) and these models deal with binary labels. In practice, this is done by using a One-vs-Rest meta model that spawns multiples instances of each binary classification models.

Note that we didn’t conduct an exhaustive parameter optimization process. Since we had to deal with limited resources, we did a minimum experimentation with parameters of each model, leaving the values that achieved best results in a trial-and-error fashion. There are a lot of room for improvement in this aspect.

For example, the *maxBins* parameter were found to be crucial for both GBT and RF models, and values of 32, 64, 128 and 256 were simulated, with 256 achieving a good trade-off in terms of performance and computational complexity.

2.4 Frameworks

This project is mainly centered on the use of **Apache Spark** and its corresponding Python API, called **PySpark.** Besides the fact that Spark is a very robust and powerful distributed data processing solution, this is an expected choice since they are the main technologies used in the previous IBM Advanced Data Science courses, and one of the objectives of this project is to put into practice what has been learned.

Apache Spark™: is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. Link: spark.apache.org

PySpark: is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing. Link: spark.apache.org/docs/latest/api/python/index.html

We focused on the use of the same libraries, data structures and algorithms that were discussed through the courses. Some punctual adaptations were made in specific cases, but overall the project is restricted to the technologies contained in the courses.

2.5 Performance evaluation

There are a multitude of performance metrics for the multiclass classification problem. Our solution is not only a toy problem, but also a real world challenge, and thus we opted for simple yet representative metrics. The main idea here is that these metrics should be able to assess the performance of a system that is really capable of predicting matches despite the probability distribution of the outcomes.

Accuracy: defined technically as

$$Acc = \frac{correct\ classifications}{all\ classfications}$$

In the context of our problem, a high accuracy means that the model correctly predicted the outcomes, and thus even if the results are skewed to some particular output, the model will be useful as a predictor.

That means that you can bet on the model and you will be right following this ratio. It is a simple (maybe the simplest) yet useful metric for a project like this.

Confusion Matrix: it is a table that allows the counting of the different of classification results in comparison with the true labels. Usually, **each row contains the predicted values and the columns the true values**. So, in each cell, we can find the count of a unique predicted and true label.

This matrix allows us to see the correct answers in its diagonal, and the different possibilities of classification errors in the other cells. It is useful because we can assess the model taking into consideration the distribution of the outputs. For instance, in our case usually a home team has advantage and thus a model that skews to predicting cases as a home win will have a natural advantage, and this matrix will allow us to analyze this behavior and quantify it.

It also has the advantage of providing a visual representation of the results. As we can see in the notebook, this matrix can be plotted as a heat map, or a 2 variable probability distribution.

There are of course other metrics like F1-score that are equally useful and comprehensive for our problem. But our two choices can be a base for a proper classification performance assessment.

3 Technical components

This project started as a Jupyter Notebook on **IBM Watson Studio**, following the same format as all the exercises of the courses in the program. Soon, resource limitations impacted the project, since every simple iteration of a model required significant computing time, which was quickly exhausted on the free tier environment offered by this platform.

Then, out of this necessity, we migrated the development of the notebook to the **Google Colaboratory** platform. This allowed to keep using the same PyPark framework and its libraries, with minimal adaptations. Thus, as this project reaches its completion, it is in the form of a notebook that is ready to be run in Google Colaboratory if needed. One should still be able run it on Watson Studio if needed, but that may require some (also minimal) adaptations – and availability of suitable computing resources.

Note that since there is no other infrastructure component that is required, everything that is needed to run all the project analyses and models are contained in the notebook. It can simply be imported and executed, and the same results shoud be achieved.

IBM Watson Studio:IBM's software platform for data science. The platform consists of a workspace that includes multiple collaboration and open-source tools for use in data science.
Link: www.ibm.com/products/watson-studio

Google Colab: Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.
Link: colab.google

4 Final remarks

Author contacts:

e-mail: palmieri.igor@gmail.com

Github: https://github.com/ipalmieri

Project repository:

https://github.com/ipalmieri/advanced-data-science-capstone