Nháp

Regression ?

Eval ML + AutoML

<https://www.kaggle.com/code/mpwolke/beautiful-game-eval-ml#Running-the-Auto-ML-to-select-best-Algorithm>

<https://www.kaggle.com/code/abh8017/why-evalml-is-one-of-the-best-automl-library#Why-EvalML-is-one-of-the-best-AutoML-library-you-can-get-your-hands-on>

FAML + AutoML: <https://www.kaggle.com/code/masterofdeception/nqk-flaml-automl>

FAML time budget: <https://github.com/microsoft/FLAML/issues/155>  
https://www.kaggle.com/code/mpwolke/the-joy-of-soccer-flaml-automl/notebook

XG Bootst

Deploy flask

**Football Match Probability Prediction**

**Table of Contents**

[Task 1: Please train a model to predict the results of matches in test.csv. The output format is the sample in sample\_submission.csv 2](#_Toc139849466)

[1. Overview and my approach 2](#_Toc139849467)

[2. Implement 3](#_Toc139849468)

[2.1. FAML + XGBoost 3](#_Toc139849469)

[2.2. RNN - LTSM 9](#_Toc139849470)

[Task 2: Based on the historical data, write code to include results of last 5 matches between two teams. 9](#_Toc139849471)

[Task 3: We need to predict the probability of scores. For example, given a match data, output the probability 10](#_Toc139849472)

[Testing 13](#_Toc139849473)

[Improvement Approaches 13](#_Toc139849474)

**Candicate: Nguyen Quoc Khanh**

**Email:** [**nqkdeveloper@gmail.com**](mailto:nqkdeveloper@gmail.com)

**Note:** I'm a perfectionist so I always want people to understand what I'm saying and writing. So my report has a theory behind everything I say. So whether you're an AI expert or not, if you're uncomfortable with it, just ignore it or give me a signal next time.

# Task 1: Please train a model to predict the results of matches in test.csv. The output format is the sample in sample\_submission.csv

# 1. Overview and my approach

In this problem, we have to predict the probabilty of the home team winning, probability of the away team winning and the probability of a draw so this is **regression** problem.

There are many types of regression models and it is difficult to test them all with large amounts of data. So I will use AutoML (Automated Machine Learning) which is a method of using tools and techniques that automate the process of building and deploying machine learning models. AutoML's goal is to reduce complexity and optimize the workflow of data analysts and developers, while enhancing the efficiency and accuracy of machine learning models.

**AutoML** automates tasks such as:

* Data preprocessing: Includes missing data processing, variable encoding, data normalization, and other data related handling.
* Model selection: Automatically search and select suitable machine learning models for the prediction problem.
* Auto-tuning of hyperparameters: Automatically search and adjust model's hyper-parameters to optimize performance.
* Automatic evaluation and comparison: Automatically perform the evaluation and comparison process between different models to choose the best model.

Another approach to this problem is to use the Recurrent Neural Network (RNN) family of models **(I do not mention in this solution because of the time limit and the length of the report)**. RNN is a model class in Machine Learning designed for sequential and predictive data processing in situations involving order or sequence. Specific models have been applied and topped the leaderboard in the competition such as LTSM, Bidirectional LSTM, GRU.

Especially the LSTM (Long Short-Term Memory). LSTM is used for regression problems (continuous value prediction) in Machine Learning because it has the ability to process sequential data and capture long-term dependencies in the data..

With its ability to process sequential data, LSTM can understand and learn correlation and dependency patterns between data points in time series. Data features such as order and time can be modeled using LSTMs, which capture complex and time-varying relationships in the data..

In addition, LSTM is also capable of long-term data processing and solves the vanishing or exploding gradient problem that traditional RNN models often encounter. With gate mechanisms in LSTM such as input gate, forget gate and output gate, it is possible to regulate the flow of information and decide which data should be stored and which should be forgotten. This helps the LSTM to maintain and use important information from the past to predict future value.

# 2. Implement

## 2.1. FAML + XGBoost

FLAML is a lightweight Python library that finds accurate machine learning models automatically, efficiently and economically. It frees users from selecting learners and hyperparameters for each learner.

This if first my approach for AutoML since for common machine learning tasks like classification and regression, it quickly finds quality models for user-provided data with low computational resources.

Full source code for FAML in **FAML\_AutoML.ipynb**

After handle with missing values and nan values, training with FAML

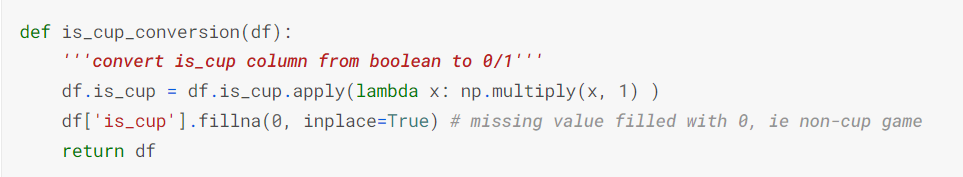


We can see the best ML leaner is XGBoost with its configurations.

I actually switched over to EvalML which is the best rated platform in the library with AutoML but i stuck with some unheard errors and also had previous research results and same results with FAML ( XGBoost) so I'll go straight to the model implementation.

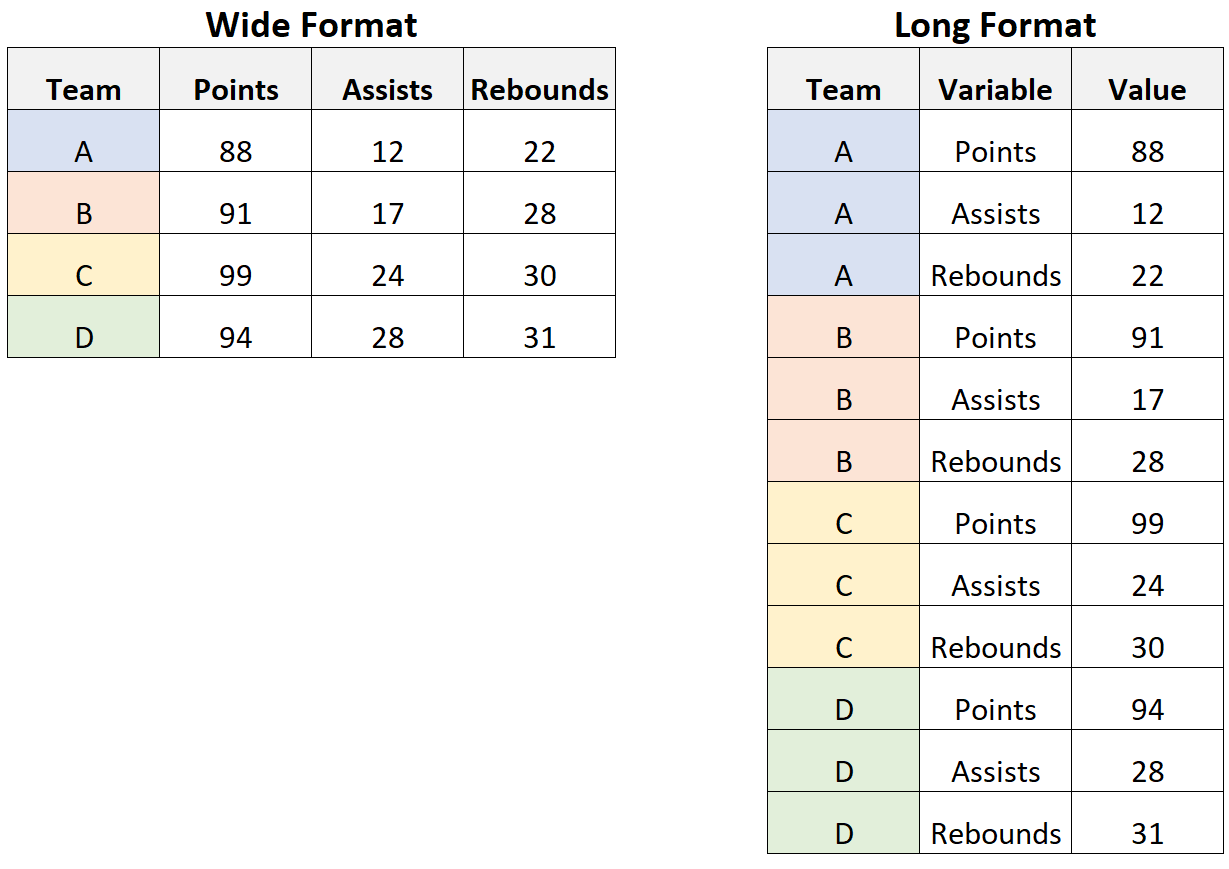
My approach is to construct sets of features to feed into the XGBoost algorithm for training, and see how far it can go.

**Sub ultils function for for convenience and better speed of manipulation**

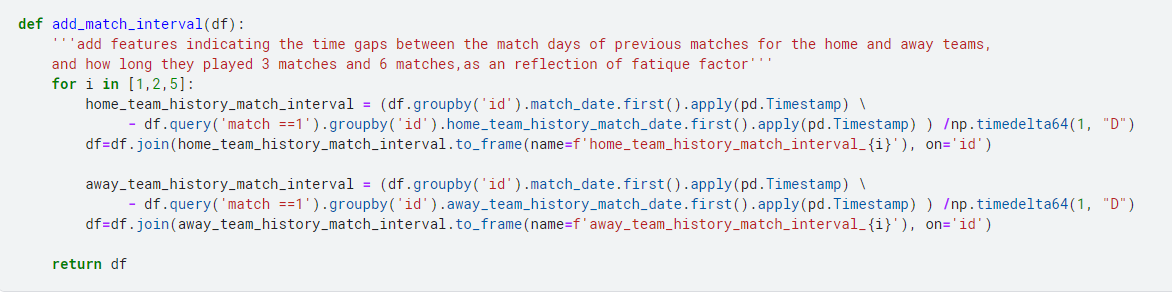


Converting the history columns into **long form**, for convenience and better speed of manipulation. The long-form data structure makes it easier for us to manipulate and analyze the data during the processing phase, such as calculating averages, sorting, and filtering data based on conditions.



  
*Diff. Between wide format and long format*

Add features indicating the time gaps between the match days of previous matches for the home and away teams, and how long they played 3 matches and 6 matches.



These time gaps reflect the interval between consecutive matches and can provide insights into the fatigue factor of the teams.

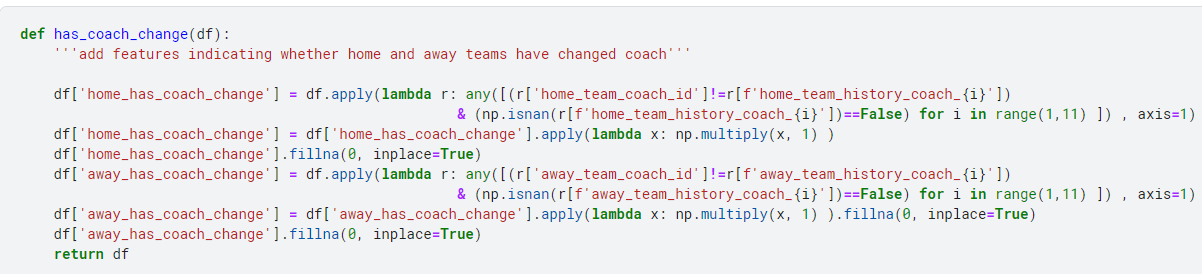
**Calculate goal difference and the point derived from it (3 points for win, 1 point for draw and 0 point for lose) of each past match**



Next **adding form features for home and away teams**, including rating difference and goal difference for each previous match, and the average points got, average goal scored, average goal conceded and average goal difference up to that match as previous n-match form. (See code in my notebook, too long to snap here)

**Add features about whether the teams have recent coach change ?**

In the context of the analysis, when a team undergoes a coach change, it implies that there has been a shift in leadership and management strategies. This change can sometimes bring about a positive impact on the team's dynamics, motivation, and gameplay. The phrase "rebound of form from the bottom" refers to the idea that a team that was previously struggling or performing poorly might experience a resurgence or improvement in their performance after a coach change.



And no matter how I kept adding features, the most salient ones remain the medium rating differences of past 10 matches of the away and home team. Although the host of this competition published an article and notebook on Elo ratings, examination of rating changes between matches for a few teams shows that the ratings are not Elo ratings for the team, at least not directly used. And the host confirmed they are calculated pairwise for the home and away teams, and combining the attack strength of one side and defense strength of the other side. In other notebook of other user, they have shown the ratings are most likely the expected goals scored by the team of the match.

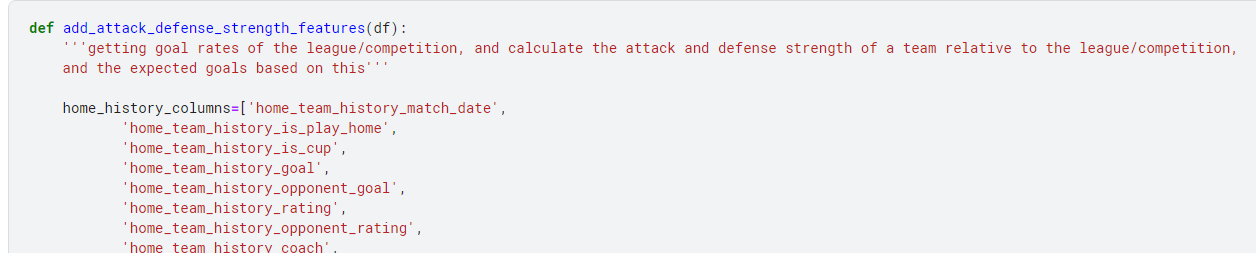
The rating differences of the past matches are informative as a rough indicator of the general strength of a team, but cannot be directly carried to the match in interest. The host does not provide the ratings for the matches to be predicted for the obvious reason. Though the host said the ratings can be calculated based on past ratings, without knowing what is the model used, it is a hard code to crack. The series of ratings in each row are combining the offensive strength of the team and the defensive strength of different opposing teams, so doing an autocorrelation on the series of ratings does not look hopeful. Statisticians of football results modeling usually fit their models with seasons of data of one or more leagues, it is not certain whether it is the appropriate approach trying to be learnt and applied for the case here.

So I decided to construct a simple model of expected goal as a feature. A simple idea used by David Spiegelhalter, a Cambridge statistician, and some pundits goes like this: first you get

**The average goals scored by an average team in a league or competition, then the offensive strength of a team is the the average goals scored by that team divided by the league average, and the defensive strength of the team is its average goals conceded divided by the league average.** With the average goals scored by a home team and the away team in the league or competition, for every match we have:

* Expected goals scored by the home team = Average goals scored by a home team in the league or competition \* Offensive strength of the home team \* Defensive strength of the away team
* Expected goals scored by the away team = Average goals scored by an away team in the league or competition \* Offensive strength of the away team \* Defensive strength of the home team

Pundits would feed these expected goals data into some Poisson distribution model to get the probabilities of every possible score line and the prediction of match result, but I leave it for XGBoost to figure out modeling.



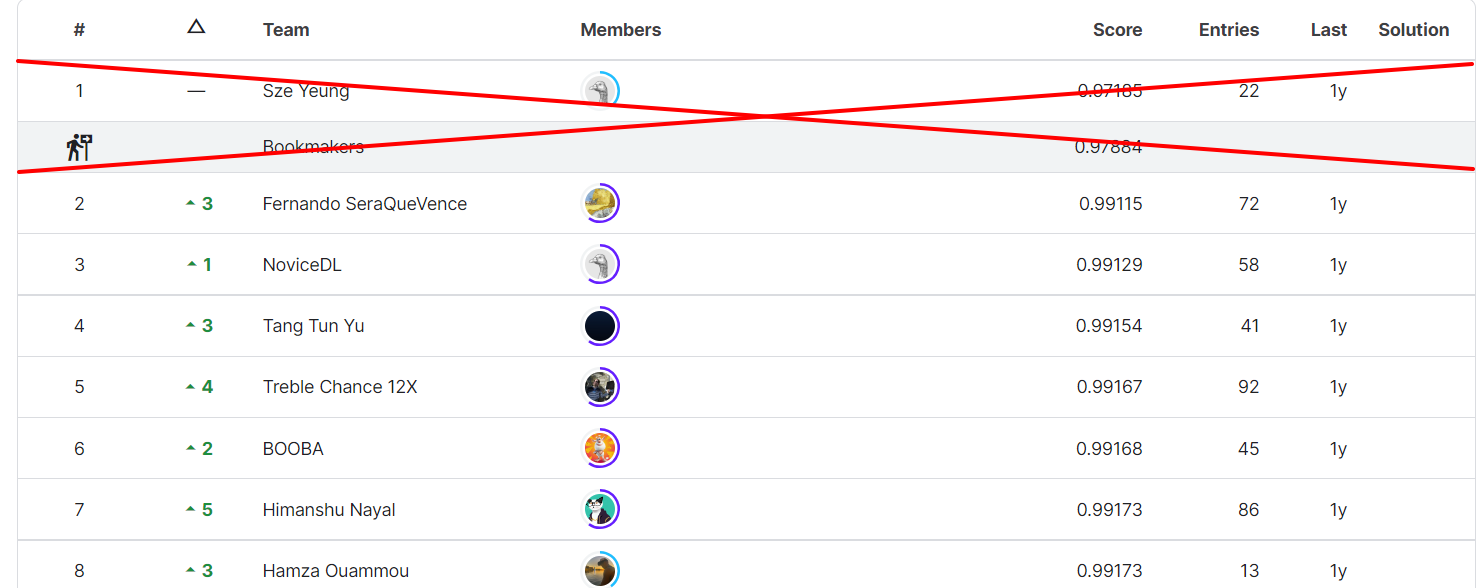
**Full code implement in add\_attack\_defense\_strength\_features() function**

The average goals data of every league in this dataset can be found by pooling the history feature columns data and groupby league id.

As for the team goals data, it should be aligned with the league id of the match to be predicted, as different tournaments have different competition levels. But I thought within the 10 previous matches data, cup matches normally only feature a couple of times to be reliable, so in the first trial I used data of all ten matches without regard to the league id.

To collect more league goals data of a team beyond the 10 past matches, normally we should use the team identifiers. But as the team names are masked in the test set following concern about data leak, and no team ids are provided, the coach id becomes an imperfect proxy, as coach records can be coming from different teams for coaches who have changed teams. So after setting up this set of features by groupby coach id and league id and starting the XGBoost training process, I did not have much expectation and did internet surfing elsewhere.

**Hyperparameter tuning on an XGBoost classifier**

**This final version has a public score of 0.99853 and private score of 0.99302**.

**LeaderBoard for Problem (1 year ago)**

Result for **XGBoost**:

* Private Score: 0.99302
* Public Score: 0.99853

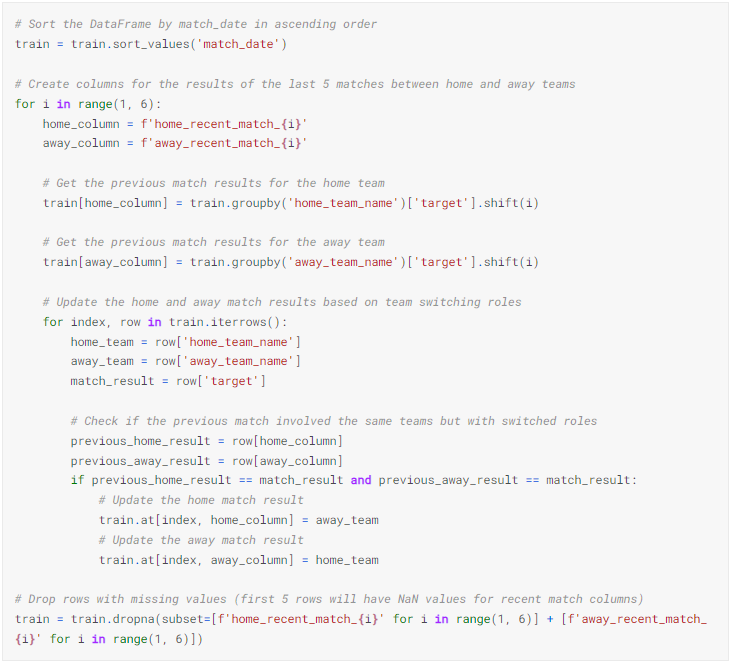
According to the results of the XGBoost model, the results are still not optimized compared to the rankings of a year ago, we need to improve further or go a different direction.

## 2.2. RNN - LTSM

This competition is to predict the results of football matches using primarily the information of past ten matches, so sequential learning may be more suitable to this job.

So I move to RNN approach. In this approach, I will more deep in handling data

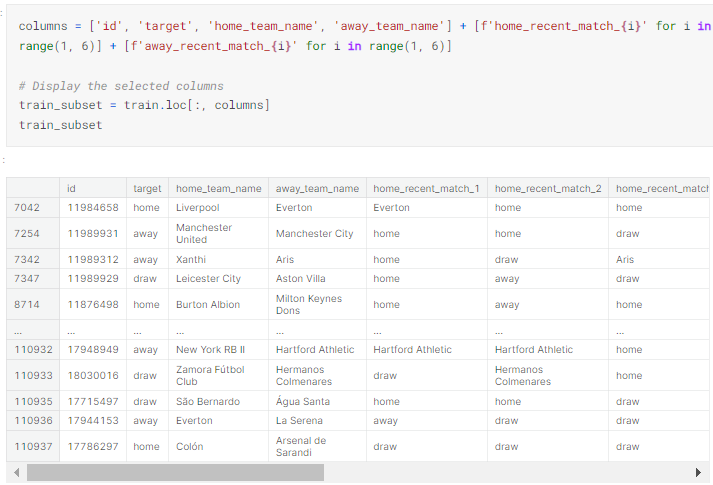
# Task 2: Based on the historical data, write code to include results of last 5 matches between two teams.



Here is a summary of the alogorithm:

1. The DataFrame **train** is sorted by the **match\_date** column in ascending order to ensure the matches are ordered chronologically.
2. For each of the last 5 matches (indexed as **i** from 1 to 5), new columns named **home\_recent\_match\_i** and **away\_recent\_match\_i** are created.
3. The code then iterates over the rows of the DataFrame and assigns the previous match results for the home and away teams in the respective columns.
4. Additionally, it checks if the previous match involved the same teams but with switched roles (i.e., the home team became the away team and vice versa). If this condition is met, it updates the match result columns accordingly.
5. Finally, the rows with missing values in the recent match columns (the first 5 rows) are dropped.

The resulting DataFrame will contain the original columns (**id**, **target**, **home\_team\_name**, **away\_team\_name**) along with the updated columns representing the results of the last 5 matches between the teams.



# Task 3: We need to predict the probability of scores. For example, given a match data, output the probability

As I calculated earlier, I already have the parameters

* home\_team\_coach\_attack\_strength
* home\_team\_coach\_defense\_strength
* away\_team\_coach\_attack\_strength
* away\_team\_coach\_defense\_strength
* home\_team\_coach\_expected\_goal
* away\_team\_coach\_expected\_goal

Với các tham số trên, ta đã sẵn sàng để dự đoán tỉ số của trận đấu sử dụng **The Poisson Distribution formula,**

The formula is as follows:

* P(x; μ): Probability of x events occurring
* e: Euler's number, approximately 2.71828
* μ: Average rate of occurrence
* x: Number of events

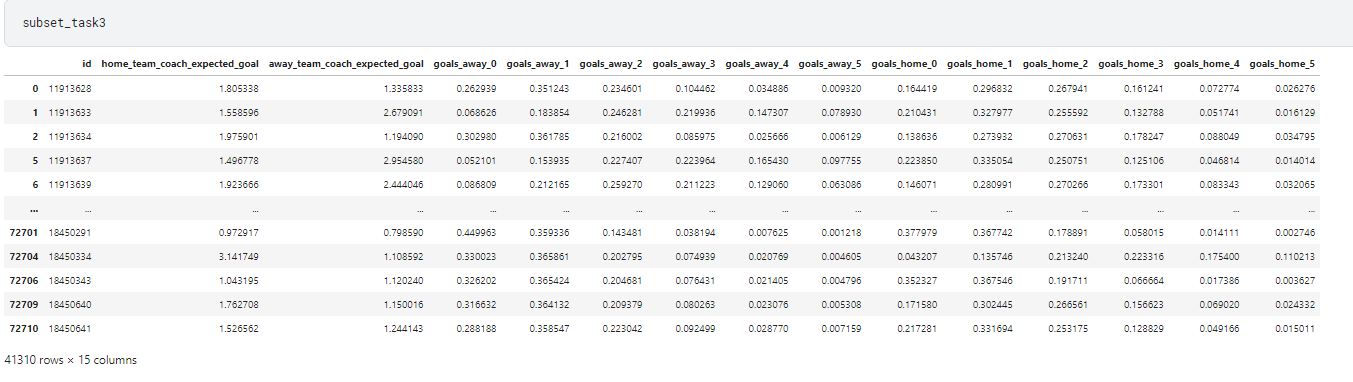
Assume that Team A’s scoring probability is 1.623 and Team B’s - 0.824. Looking at the matchup between two, we are interested in 0-5 goals for each one. Using the aforementioned tools, we are going to get the following results.

**Team A vs. Team B Poisson Distribution**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Goals** | **0** | **1** | **2** | **3** | **4** | **5** |
| **Team A** | 19.73% | 32.02% | 25.99% | 14.06% | 5.07% | 1.85% |
| **Team B** | 43.86% | 36.14% | 14.89% | 4.09% | 0.84% | 0.14% |

Code for implement:





Multiply the corresponding probabilities for each team to get the joint probability of a specific score combination. In this case, example I want to calculate the probability of a 3-1 score, which corresponds to Team A scoring 3 goals and Team B scoring 1 goal.

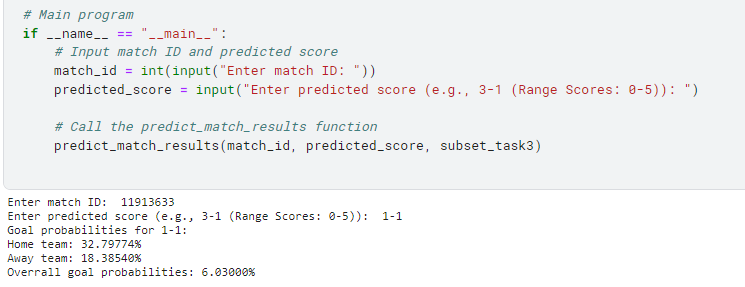
* Probability of Team A scoring 3 goals: 14.06%
* Probability of Team B scoring 1 goal: 36.14%

Multiply the probabilities together to get the overall probability of the specific score combination:

* Probability of A:B - 3:1 = 14.06% \* 36.14%

So, the probability of the match result being A:B - 3:1 is the product of these two probabilities.

## Testing



**Theory Reference:**

* <https://www.bettingwell.com/sports-betting-guide/football-bettors-guide/how-predict-score-football-betting>

# Improvement Approaches

* Data Augmentation: Generate synthetic data points or augment the existing data to increase the diversity and quantity of the training set. Some additional factors that could be taken into account are:
  + Angle, height and direction of a shot.
  + Player’s and goalkeeper’s skill.
  + Type of assist (cross, pass, rebound, etc).
  + Number of touches before the shot.

References for this: <https://xgscore.io/xg-statistics/explained>

* **Time Series Analysis:** Consider the temporal aspect of the data by applying time series analysis techniques. Explore seasonality patterns, trends, and other time-dependent factors that can impact match outcomes. Incorporate time series models like ARIMA, SARIMA, GRU or LSTM (BI-LSTM) (Long Short-Term Memory) to capture and utilize this temporal information.
* **Transfer Learning**: Explore the potential of transfer learning by leveraging pre-trained models or embeddings from related domains, such as player performance in other sports, to extract useful representations and features for the Football Match Probability Prediction problem.