# Task 1 : MLP

In this task, you are supposed to work with a football dataset! This dataset contains the match result and game status of road to Qatar 2022 FIFA World Cup Qualifiers along with the actual matches of Qatar 2022 and you are going to train a Multi Layer Perceptron to predict the result of games. Finally you are going to use your model to predict the entire tournament of Qatar 2022.

## 1-Dataset Loading

First, let's explain the dataset. This dataset is consisted of multiple attributes which are to be explained:

1. home_team: The national team designated as the home team for the match.
2. home_goals: The number of goals scored by the home team in the match.
3. away_goals: The number of goals scored by the away team in the match.
4. away_team: The national team designated as the away team for the match.
5. wcm: Stands for "World Cup Match" — this is a binary flag (1 or 0) indicating whether this match is part of the actual World Cup tournament or just a qualifier match.
6. dif_inter_match: Difference in the total number of international matches played by the home team minus the away team.
7. dif_inter_match_won: Difference in the number of international matches won (home - away).
8. dif_inter_match_lost: Difference in the number of international matches lost (home - away).
9. dif_inter_match_tie: Difference in the number of international matches tied (home - away).
10. dif_inter_goals_sco: Difference in goals scored in international matches (home - away).
11. dif_inter_goals_con: Difference in goals conceded in international matches (home - away).
12. dif_wc_match: Difference in the total number of World Cup matches played (home - away).
13. dif_wc_match_won: Difference in the number of World Cup matches won (home - away).
14. dif_wc_match_lost: Difference in the number of World Cup matches lost (home - away).
15. dif_wc_match_tied: Difference in the number of World Cup matches tied (home - away).
16. dif_wc_goals_sco: Difference in goals scored in World Cup matches (home - away).

17. dif_wc_goals_con: Difference in goals conceded in World Cup matches (home - away).
18. status: The result of the match, typically encoded as:
    a. 1: home won
    b. 2: tie
    c. 3: home loss

Now let's dive into the steps of this task one by one.

## 3- Exploratory Data Analysis

In the Exploratory Data Analysis (EDA) phase, start by examining the dataset to understand its overall structure, including the number of rows and columns, data types, missing values, and the distribution of features. Analyze the distribution of match outcomes to identify whether the classes are balanced or imbalanced. Investigate numerical features, such as differences in the number of matches played or goals scored, to uncover potential relationships with match results. Utilize visualizations like histograms, box plots, and correlation matrices to gain deeper insights. Analyze team performance in home and away matches to validate logical patterns in the dataset. As suggested, ensure missing values are handled and exclude features that directly determine the outcome (e.g., goals scored) to avoid data leakage. Finally, check whether numerical features require standardization or normalization to ensure the data is properly prepared for training the model.

## 2-Data Preparation

First you need to prepare this data to feed it to a model. Load the dataset as a pandas dataframe. The process of training the model is on games whose wcm attribute is 0 because we want to have a final prediction on the winner of the World Cup, we need the world cup matches' data to remain unseen for our model.

Then we will need to select X and y. X is our set of features and y is the label which is to be predicted. In the context of this dataset, X will be all columns except the following columns:

1. home_team
2. home_goals
3. away_goals
4. away_team
5. status

and y will be the status column. The reason for which we omitted the above columns is obvious for columns home_goals and away_goals because the result of the match is specified if we know the number of goals scored by home and away sides. On the other hand, we omit name of teams from our features because we want to force our model to learn the patterns of the match according to its status; otherwise it will be biased to name of

teams; For example if Germany has more wins than Spain, the prediction of the model will be Germany without considering the other criterias.
Then we need to encode labels. You can use sklearn's label encoder for this phase.
Then split your data into train and test sets using a test portion of 30%.
The next thing you should do is standardize features. You can use StandardScalar or sklearn for this phase.

- Warning! One important thing to note here is to avoid data leakage from test set to train set. The StandardScalar should be fit only to train data and fitting the scalar to train and test set jointly will cause leakage.

Then you should convert the data to pytorch tensor. At this phase, your data is ready to be fed to a neural network model.

## 3- Model Definition

Now you should write your model class. Note that you are only allowed to use pytorch or tensorflow library (but we suggest pytorch) to define and train your model. Your model class should inherit from torch.nn. Module and should implement two methods __init__ and forward. The arguments of init depends on how you define the model. But note that the forward method should always have input X as an argument.
The way you define your MLP is arbitrary and you can use as many layers as you want.

## 4- Model Training

Let's have a brief description over training a pytorch model for classification. We need to define a set of tools for training a pytorch model.
First you need to specify the criterion by which you want to train your model. For multi class classification, torch.nn.CrossEntropyLoss is a proper choice.
Then you will need an optimizer. Optimizer is instantiated with two important arguments; model.parameters() and learning rate.
Now let's talk about the training loop; you should repeat the training loop for a fixed number of epochs.
In the training loop, you need to clear the gradient cache of the optimizer to avoid summing the gradient of the previous loop with the current loop; this can be achieved by the zero_grad() method of the optimizer.
Then you need to pass the batch of training data to the model and calculate the output; having calculated the output, you can calculate the loss, using the criterion. Then using the backward() method of loss tensor, you will have the gradients calculated at the computation tree of model parameters. Then by calling the step() method of the optimizer, one step of optimization will be taken and repeating this loop, the model will be trained.

## 5- Evaluating Model

After training the model, you need to evaluate its performance on the test set. Accuracies higher than 50 will achieve full marks. We really don't expect our model to achieve higher than 90% accuracies like other datasets because football is so unpredictable!

## 6- Run FIFA World CUP!

In this part, you are going to use your model to predict the tournament including group stage and knockout stage. As you know, the first two teams of each group are qualified for the knockout stage. Don't forget to use the real groups of Qatar 2022 as your initial configuration: