Champions League Knockout Stage Predictor Challenge

AI Club : DC Hackathon 25-26

The Challenge

Predict how the UEFA Champions League knockout stages will unfold from Round of 16 to the final champion using historical match data, team performance metrics, and advanced analytics.

1 Problem Statement

Given the European football matches dataset, which consists of match data from various leagues across Europe, develop a predictive pipeline to forecast Champions League knockout match outcomes. You will:

1. Use data up to the start of knockout rounds (training data cut-off: 31-4-2017)
2. Predict the entire tournament bracket progression (testing: 2017/18 to 2024/25 seasons)
3. Compare your predictions against actual outcomes

2 Data Provided

- **European football matches dataset** : Consists of data of all league matches in European leagues with results and scores.
- **Champions League knockout stage games** : These contain dates and scores for Champions League knockout fixtures, which is what you are trying to predict. The train set consists of all the results

from 2004/05 to 2016/17 seasons, while the test set has the initial bracket with filler info to allow you to understand how the bracket will progress.

Here is a visual of how a bracket looks like, telling us how the path of teams is already decided from the Round of 16 onwards itself.

Beyond this, you are free to use any data available online. Scraping data and integrating it into your pipeline will be appreciated. Understand the problem properly and look out for good starter notebooks, repos and articles that may be helpful.

3 Expected Pipeline Components and Resources

This is a pipeline that covers our minimum implementation expectations. Make sure to go through all the resources and understand how each part functions. Test different methods with the data to figure out what works and what doesn't. Remember, when it comes to datathons, there is no 'one-size-fits-all'!

3.1 Data Preparation

- Clean, normalize, and structure the raw dataset
- Create temporal splits respecting chronological constraints
- Identify Champions League matches vs. domestic competitions
- Extract team performance trajectories over time
- **Here's a starter resource for data preparation**
- Go through **this blog** for data preprocessing techniques

3.2 Feature Engineering

- **Team Performance Metrics:**
  - Recent form (e.g., win percentage over last 10 matches)
  - Goals scored/conceded rates
  - Home/away performance differentials

- **Champions League Specific Features:**
  - Historical knockout stage performance
  - Group stage results from current season
  - Experience in late tournament stages
- **Matchup Features:**
  - Head-to-head history
  - Style compatibility metrics
  - Relative team strength indicators
- **Integrating existing metrics**
  - Research about powerful existing metrics like xG
  - Go through **this link**. You may find some interesting metrics
  - Scraping data and integrating it into your data is a huge plus
- **Resources for EDA - Exploratory Data Analysis**
  - **Basic starter to EDA**
  - **Introduction to EDA notebook**.

3.3 Model Development

- Train models using historical knockout matches (model training data cutoff is 31-4-2017)
- Keep a val/test set to validate on held-out seasons
- Optimize feature selection and hyperparameters
- Go through the following blogs for better understanding on **model selection** [**Blog 1** and **Blog 2**]
- Resources for hyperparameter tuning:
  - Know more about hyperparameter tuning **here.**
  - Common tools used for hyperparameter tuning **optuna** and **gridsearchCV**

3.4 Tournament Simulation

- For each test season:
  - Start with actual Round of 16 matchups from the test set
  - Predict each match outcome
  - Advance winners through bracket
  - Update predictions with latest data before each round
  - You may perform inference using the latest data, but the cut-off date for such data will be as per the dates for each round given in the tournament bracket data.

4 Evaluation

The rubric for evaluation is defined as :

$$0.6 * Points + 0.2 * Interpretability + 0.2 * Ingenuity$$

- **Points:** Your predictions for each season in the test set will be scored by the following pattern :
  - Round of 16 correct predictions: 1 point each (max 8 points)
  - Quarterfinal correct predictions: 3 points each (max 12 points)
  - Semifinal correct predictions: 8 points each (max 16 points)
  - Final correct prediction: 20 points
  - Maximum possible score per season: 56 points

  Example: If you correctly predict 6/8 Round of 16 winners, 3/4 quarterfinal winners, 1/2 semifinal winner and the wrong final winner, you would score: $6 + 9 + 8 + 0 = 23$ points. This will be converted into a percentile wrt the top submission.

- **Interpretability:**
  - Clearly explain how your pipeline works
  - List out your experiments with the data and back your decisions with results
  - Maintain a clean codebase

- **Ingenuity:**
  - We don't want you to just ape the pipeline we provided above
  - Make novel choices for your pipeline and explain your intuition and back it up with results

5 Deliverables

1. **Final Predictions on Kaggle**
   - Complete bracket predictions for 2017/18 to 2023/24 seasons
   - .csv file with structured prediction outputs as given in the sample_submission.csv
2. **Codebase**
   - Well commented and neat codebase
   - A .py file which can be directly run by us to get your .csv outputs
3. **Report :**
   - Pipeline explanation
   - Key insights and feature importance analysis
   - Discuss all experiments, whether they pass or fail
   - Performance metrics and validation results

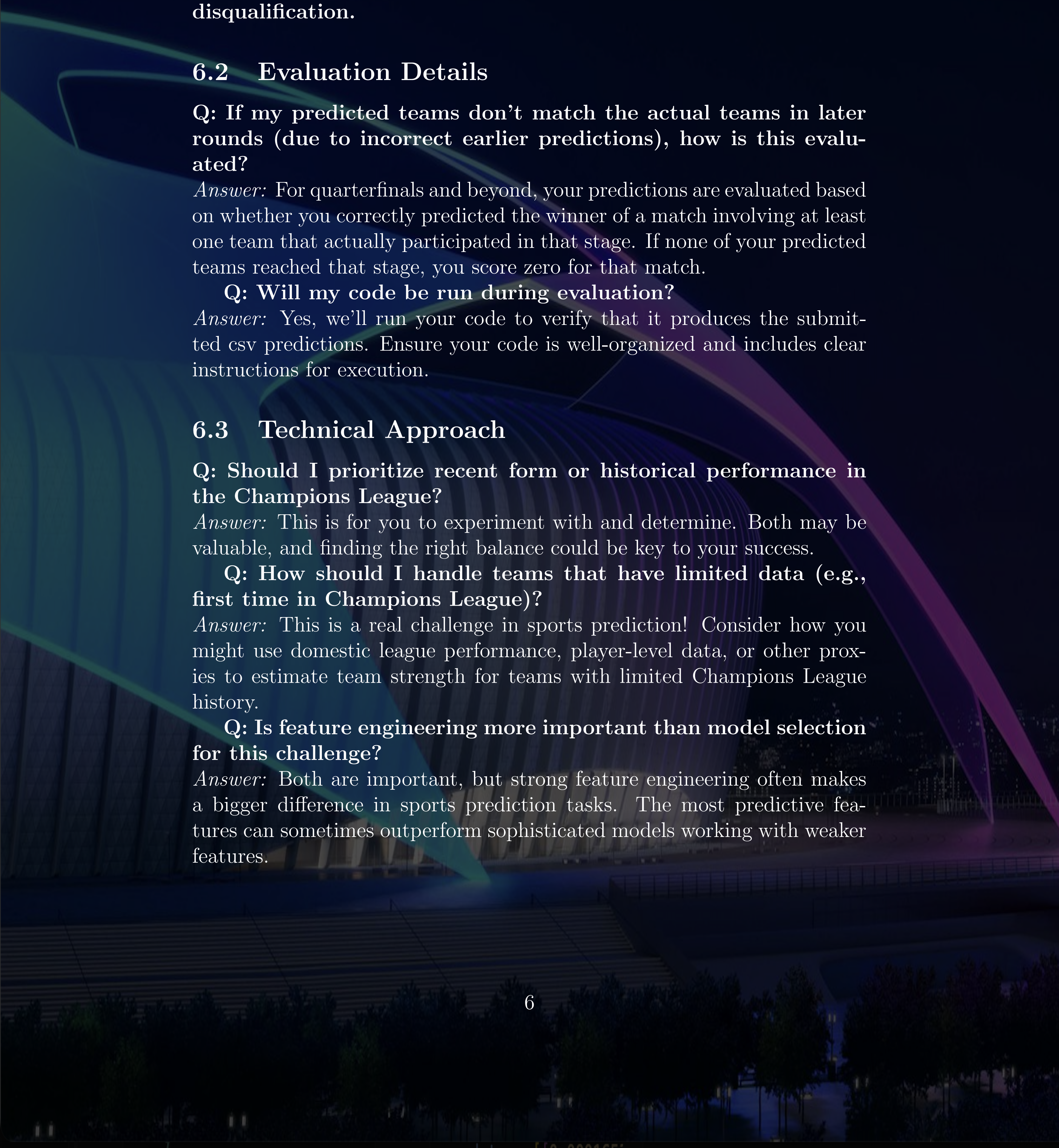
All these files have to be zipped into a file named $< team\_name >$ *.zip* and submitted in this **Google Form**.

6 Frequently Asked Questions

6.1 Data and Access

Q: Can I use external data beyond what is provided?
Answer: Yes, you are encouraged to use any publicly available data. Scraping additional data and creatively integrating it into your pipeline will be viewed positively during evaluation.

Q: What is the exact cutoff date for training data?
Answer: 31-04-2017 is the cutoff for model training data. However, for inference you may use any latest data with the cutoff being the date specified for that round.



Q: What is a data leak?
Answer: Data leak occurs when the answer of the test set itself is fed into the model for training, leading to unrealistically high accuracy. This can happen if you include data post a cutoff date for training/inference. **Any instance of a data leak in your codebase will lead to direct disqualification.**

6.2 Evaluation Details

Q: If my predicted teams don't match the actual teams in later rounds (due to incorrect earlier predictions), how is this evaluated?
Answer: For quarterfinals and beyond, your predictions are evaluated based on whether you correctly predicted the winner of a match involving at least one team that actually participated in that stage. If none of your predicted teams reached that stage, you score zero for that match.

Q: Will my code be run during evaluation?
Answer: Yes, we'll run your code to verify that it produces the submitted csv predictions. Ensure your code is well-organized and includes clear instructions for execution.

6.3 Technical Approach

Q: Should I prioritize recent form or historical performance in the Champions League?
Answer: This is for you to experiment with and determine. Both may be valuable, and finding the right balance could be key to your success.

Q: How should I handle teams that have limited data (e.g., first time in Champions League)?
Answer: This is a real challenge in sports prediction! Consider how you might use domestic league performance, player-level data, or other proxies to estimate team strength for teams with limited Champions League history.

Q: Is feature engineering more important than model selection for this challenge?
Answer: Both are important, but strong feature engineering often makes a bigger difference in sports prediction tasks. The most predictive features can sometimes outperform sophisticated models working with weaker features.