



Machine Learning Systems Design

-

LolFFate

Milestone 2

Context

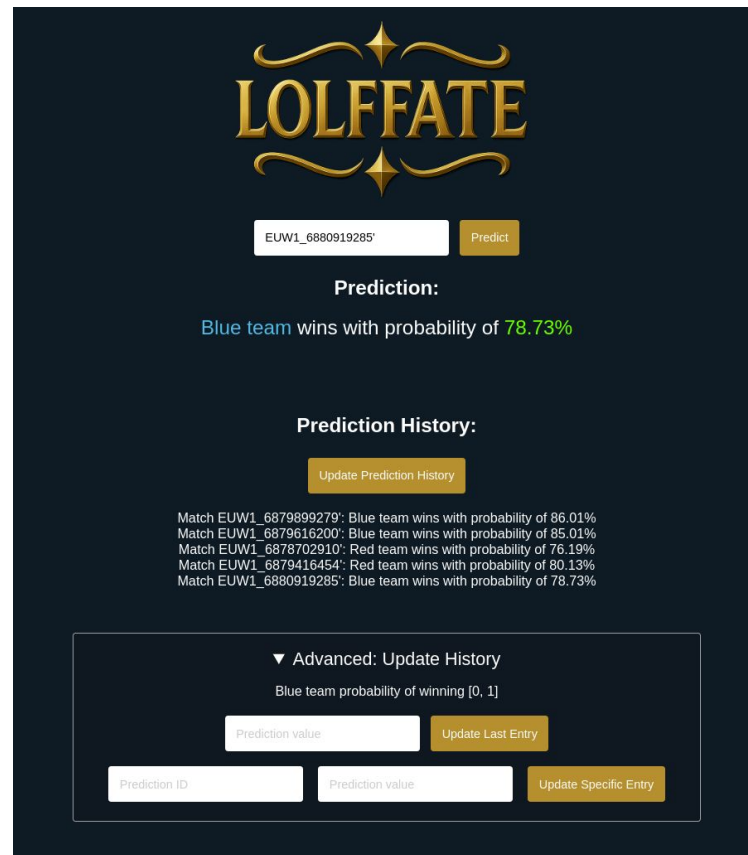
- League of Legends (LoL) is one of the most popular video game in the world.
- It is:
 - a 5v5 multiplayer game;
 - which mixes **strategy** and **skills**;
 - and is known for causing lots of **frustration**.
- Players can **ForFeit (FF)** after **15 min** of gameplay



source: <https://wiki.leagueoflegends.com/en-us/>

Our solution: LolFFate

- **FFate** is a platform allowing LoL players to **predict their probability of winning or losing** their current game.
- The predictions are made thanks to an **ML model** and are served through a **Flask application**.
- As of now, our app has **2 versions** :
 - The **online** version on **Google Cloud**;
 - The **local** version for the **player's machine**.



The screenshot displays the LolFFate web application interface. At the top, the 'LOLFFATE' logo is centered. Below it, a text input field contains 'EUW1_6880919285', followed by a yellow 'Predict' button. The 'Prediction:' section shows 'Blue team wins with probability of 78.73%'. The 'Prediction History:' section includes a yellow 'Update Prediction History' button and a list of match results with their respective probabilities. At the bottom, an 'Advanced: Update History' section contains a 'Blue team probability of winning [0, 1]' label, a 'Prediction value' input field with a yellow 'Update Last Entry' button, and a row with a 'Prediction ID' input field, another 'Prediction value' input field, and a yellow 'Update Specific Entry' button.

LOLFFATE

EUW1_6880919285 Predict

Prediction:

Blue team wins with probability of 78.73%

Prediction History:

Update Prediction History

Match EUW1_6879899279: Blue team wins with probability of 86.01%
Match EUW1_6879616200: Blue team wins with probability of 85.01%
Match EUW1_6878702910: Red team wins with probability of 76.19%
Match EUW1_6879416454: Red team wins with probability of 80.13%
Match EUW1_6880919285: Blue team wins with probability of 78.73%

▼ Advanced: Update History

Blue team probability of winning [0, 1]

Prediction value Update Last Entry

Prediction ID Prediction value Update Specific Entry

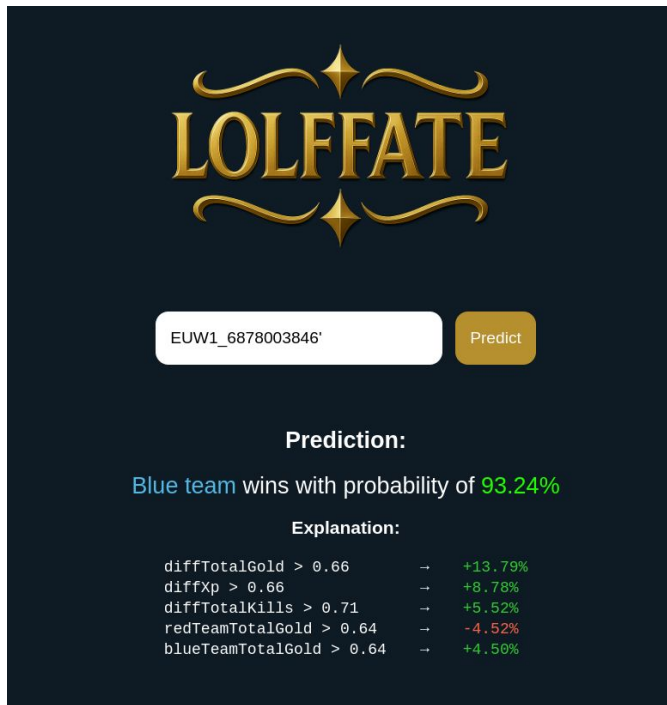
Milestone 1: Feedback consideration



- Available on GitHub in [Documentation/Milestone2](#)
- Experimentation review
 - Further **explanations** for our motivations:
 - Addressing features **relevance & selection**;
 - Addressing our **model evaluation**.
- Use case review
 - Better **definition** of our product value:
 - Prediction of the **winning probability**;
 - Proposition of an **online dashboard**;
 - Allows an **improved decision making**.

Architecture for Model Serving

- A Flask application serves our model.
- The Flask web platform provides different functionalities to the users:
 - Get the **prediction probability** of a game by inputting a **game ID** ;
 - See the **explanations** on the **features relevance** in the prediction for a better understanding of it.



Model Deployment



- The **Flask application** can be packaged in a **Docker** Container and run **locally**.
- However, the application is also available on **Google Cloud**
 - It can therefore be **accessed remotely**
→ <https://flask-app-30182159501.europe-west1.run.app/>
- The **online** version of our application currently makes predictions on **existing matches** from the training dataset.
 - For example, try entering the match ID `EUW1_6880890229` '

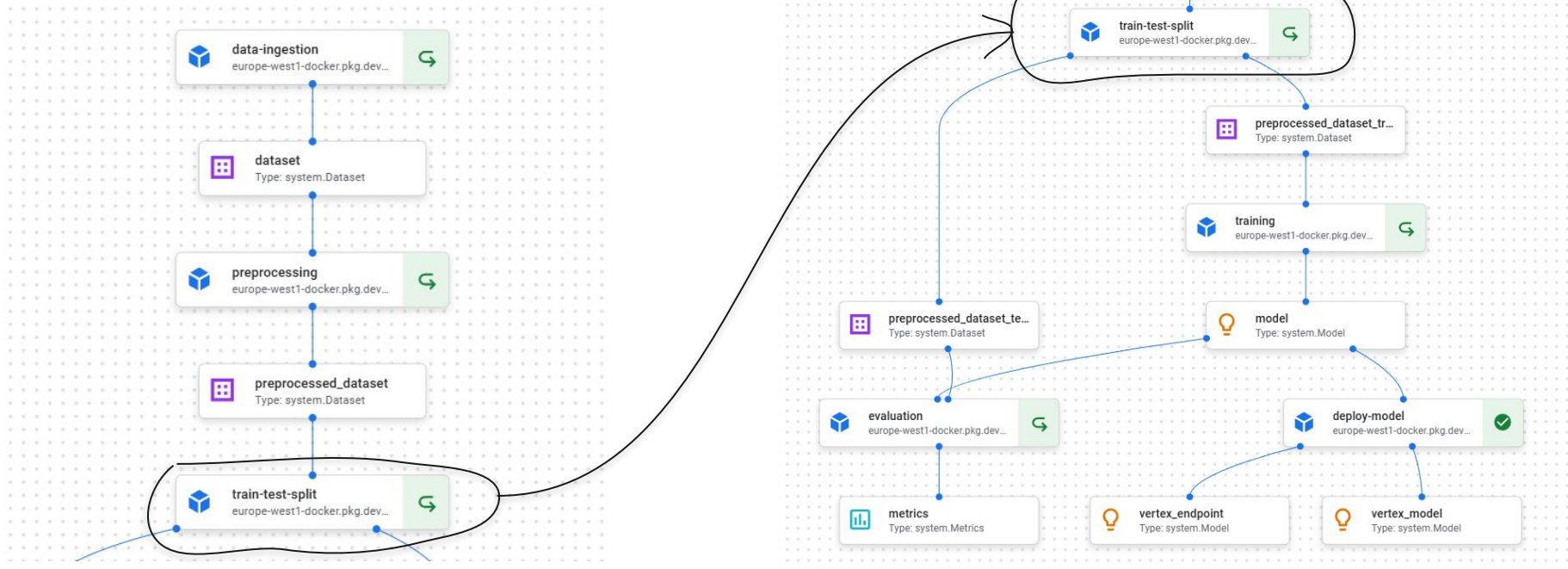
Model Pipeline

- **Vertex AI API pipelines** to build our model **automatically** through different steps:
 - Data **ingestion**
 - Dataset **pre-processing**
 - Dataset **splitting**
 - Model **training**
 - Model **evaluation**
 - Model **deployment**
- **After running**, the pipeline provides us:
 - The model **evaluation metrics**
 - The **deployed model** and its **endpoint**



Model Pipeline

- Our Pipeline Topology:



Riot API & Local Version

- Integration of Riot API(s)
 - Riot Web API (online)
 - Players history, rank, details about past games
 - Client API (local)
 - Information about local live game
- The **local** version of our application uses the Client API to make predictions on the **ongoing matches** that are launched in the **local machine** of the player.
- Further improvement : **combining** the two versions





Machine Learning Systems Design

-

LolFFate

Thank you for listening!