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

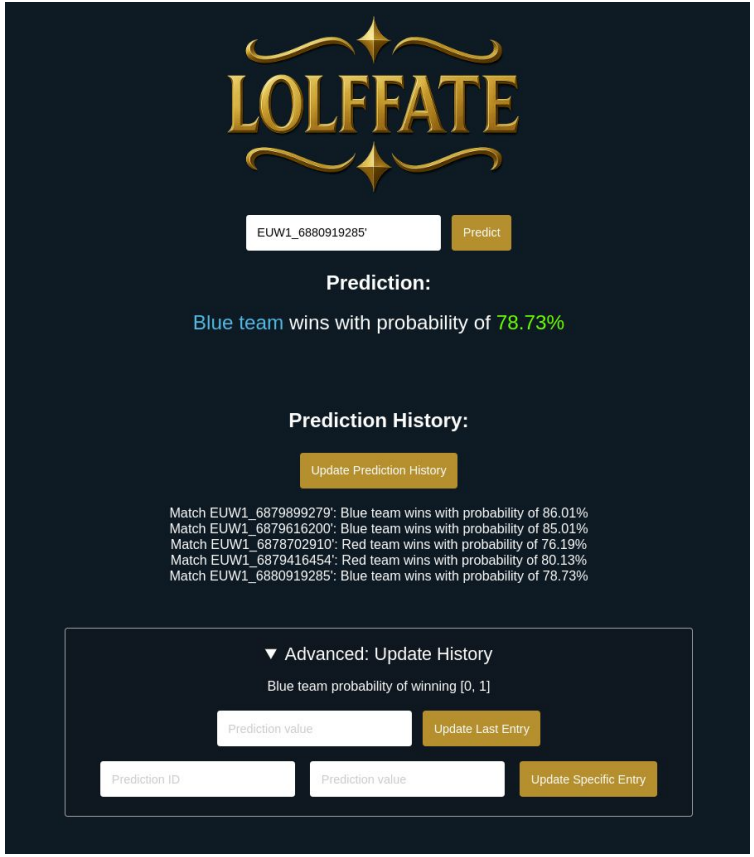
source: https://wiki.leagueoflegends.com/en-us/

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**.

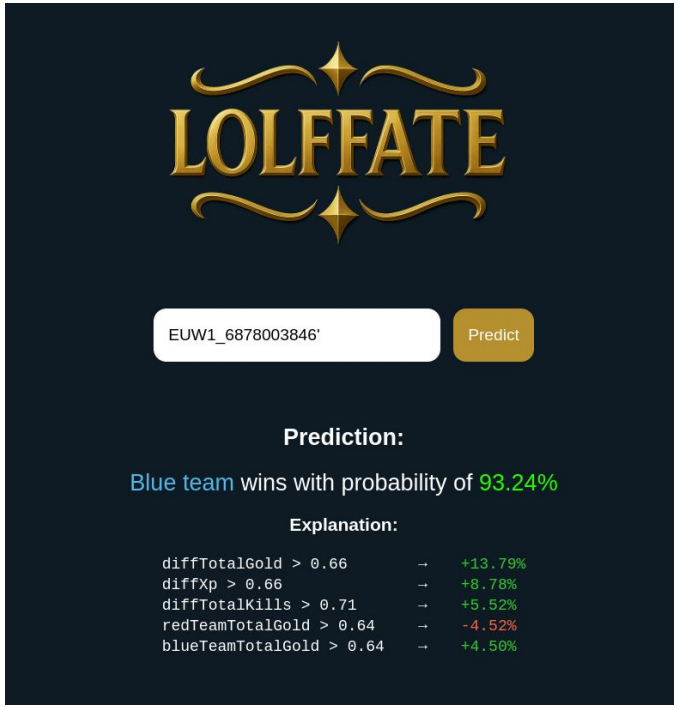


Milestone 1: Feedback consideration

- **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'`

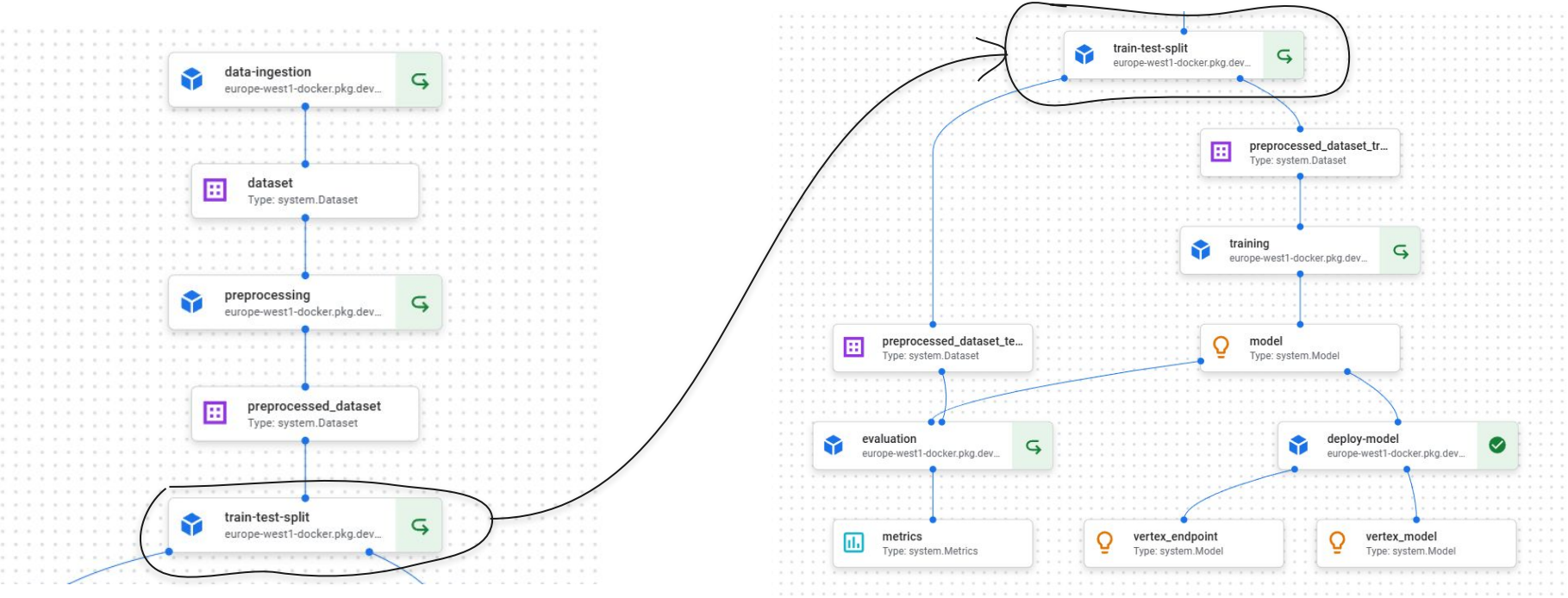
Model Pipeline

- **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!