Machine Learning Systems Design

-

LolFFate

Milestone 3

Context

- **League of Legends** (LoL) is one of the **most popular** video games 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

- **Problem**:
  - Players waste time in unwinnable games, with no early-game performance insight. **They wonder if their match is worth continuing, or how to improve it.**

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

Idea

- **A web dashboard** which could **predict the probability of winning after 15 minutes.**

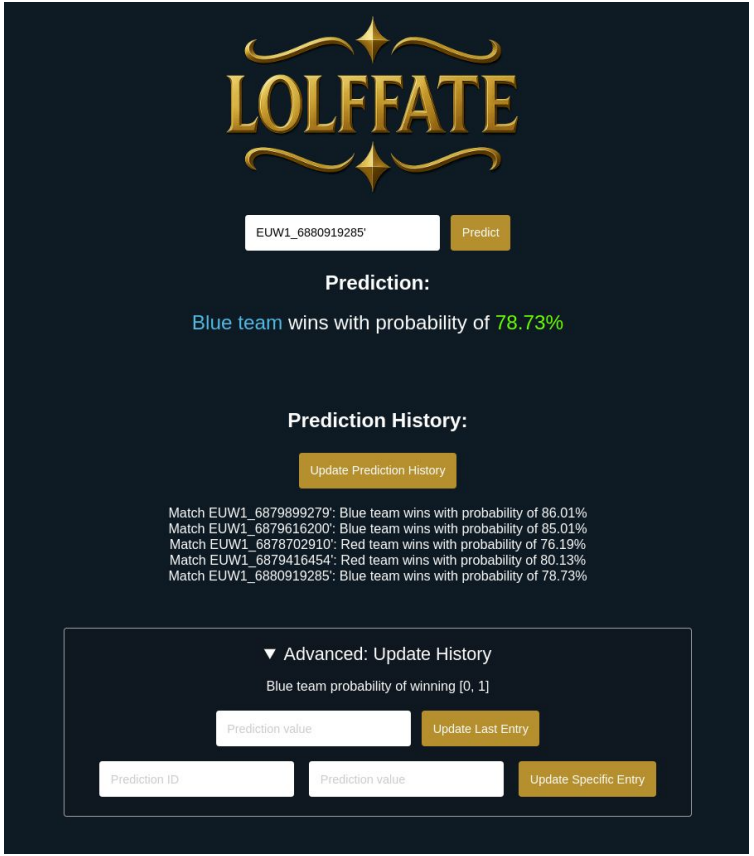
- **Improves decision-making for the players:**
  - They can have a **clearer view** by seeing which factors matter most (like gold, or objectives);
  - They can quickly spot their **strengths and weaknesses**.

- Potentially **reduces frustration**.

Our solution: LolFFate

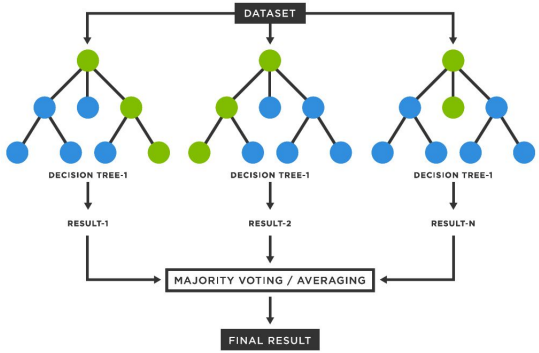
- **FFate** is a platform allowing LoL players to **predict** the **probability of winning or losing** their current game.

- The predictions are made thanks to an **ML model** and are served through **Flask and Streamlit applications** .
- As of now, our app has **2 versions** :
  - The **online** version on **Google Cloud**;
  - The **local** version for the **player's machine**.

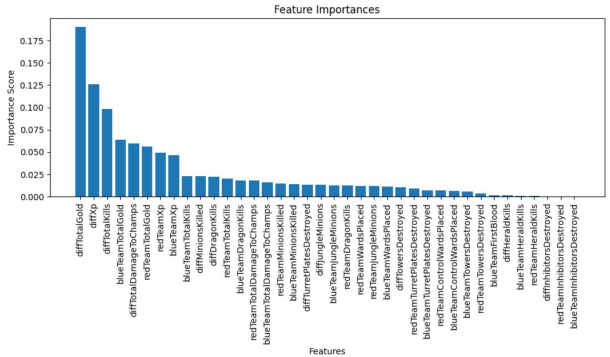


The Model

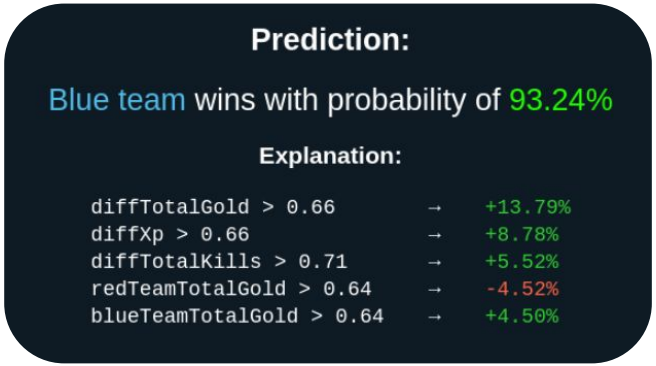
- ML model used: **Random Forest Classifier.**
- It is trained using **Log-Loss** as test metric.



Regarding explanation of the results:



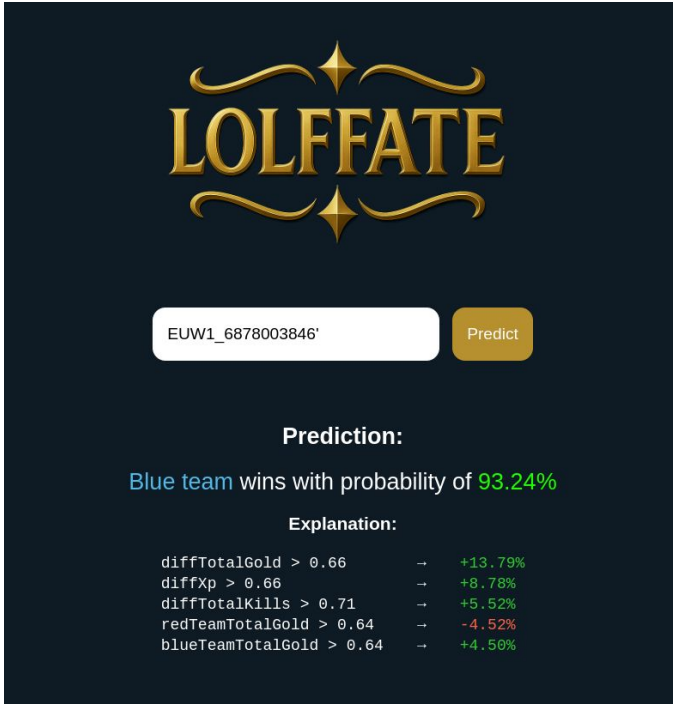
Random Forest feature importances



LIME explainer

Architectures for Model Serving

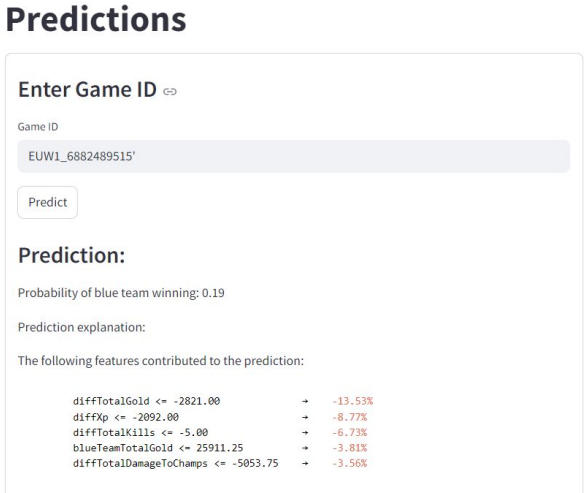
- A **Flask application** serves our **model**.
- The **Flask web platform** provides different **functionalities** to the users:
  - Get the **prediction probability** of a game by inputting a **game ID** ;
  - See the **explanations** on the **features relevance** in the prediction for a better understanding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

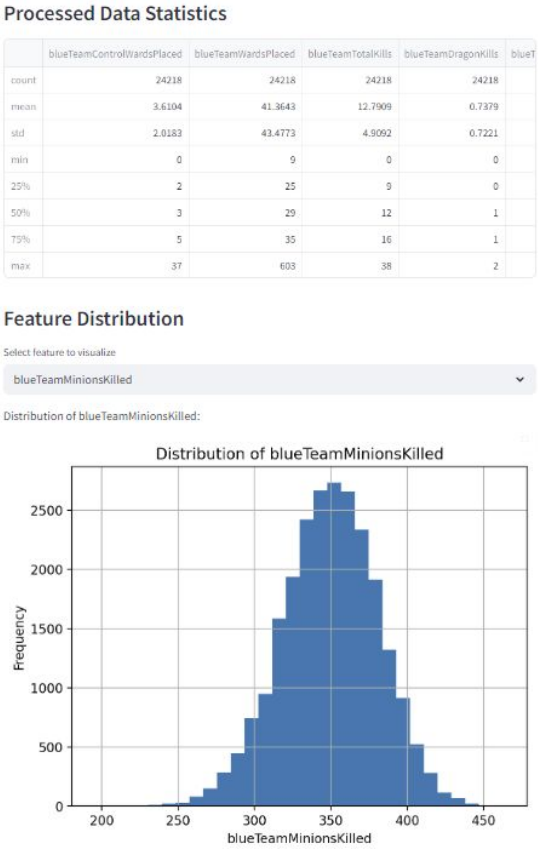


Scan to see it!

Architectures for Model Serving

Scan to see it!

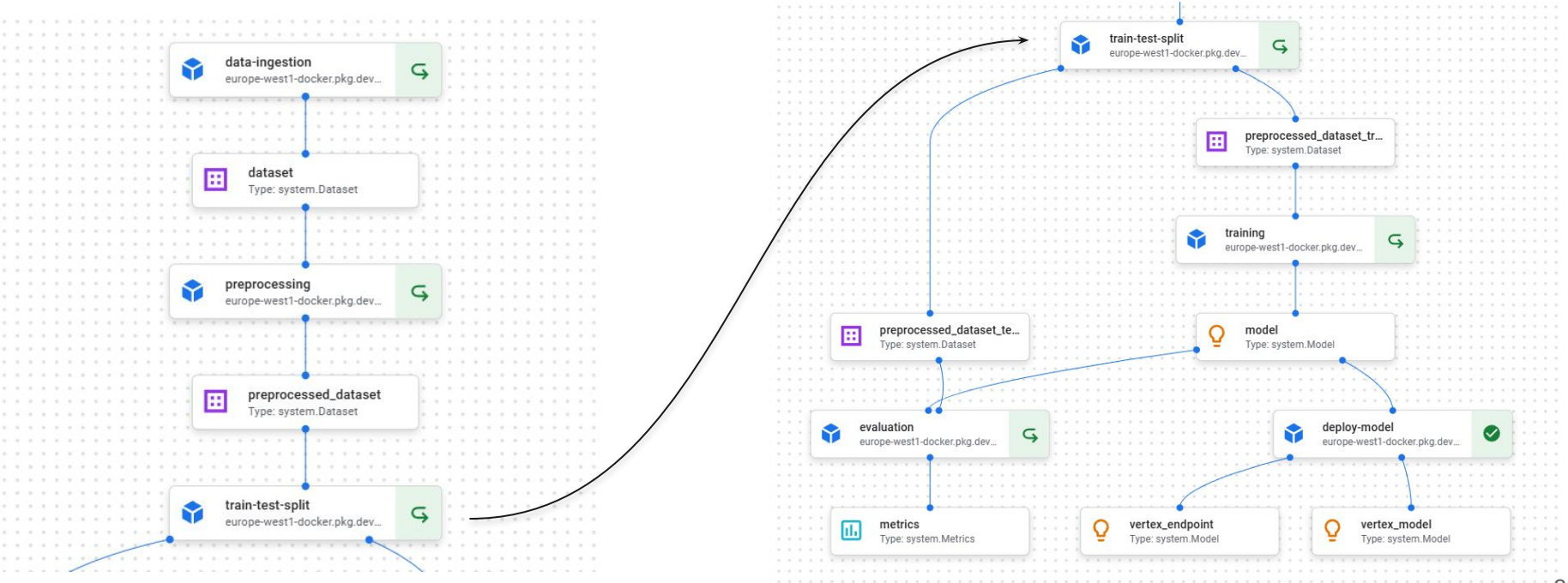




- A **Streamlit app** also serves our **model** !
- **Same functionality** as the flask app for the prediction part.
- … but it also has **data visualization** for the **dataset**.

Model Pipeline

- **Vertex AI pipeline** to build our model :



Model Deployment

- Both the **Flask** and **Streamlit applications** are available on **Google Cloud Run**
  - They can therefore be **accessed remotely**

    Flask App → https://flask-app-30182159501.europe-west1.run.app/

    Streamlit App → https://streamlit-app-30182159501.europe-west1.run.app/

- The **online** version of our app makes predictions on **existing matches** from the training dataset.
  - For example, try entering the match ID `EUW1_6880890229'`

- The **latest Flask application** is **automatically re-deployed on Google Cloud**

  when a pull request on the main branch is merged thanks to the **GitHub CICD**.

Riot Web API

- Riot Games provides **two APIs** for LoL:
  - Web API;
  - Client API (local).
- The **Web API** features :
  - Player info;
  - Match history;
  - Game statistics;
  - Timeline…
- The **API key:**
  - Only lasts 24h;
  - is Rate-limited.
- **Permanent access** to the API requires
  - Production key;
  - Manual validation from Riot Games through a formal application process.

Client API & Local Version

- The **Client API** features:
  - Information about local live game.
- We built a **local version** of our application which:
  - Uses Client API;
  - Makes outcome predictions on the ongoing match launched on the player's machine.

- **Demo**

Machine Learning Systems Design

-

LolFFate

Thank you for listening!