# AI Participation Prediction — Case Study v1

## 1. Product context

This platform was designed to support the organization of board game events between players. Beyond core operational needs, its main goal is to provide reliable participation data to support product and operational decision-making. The main problem of our users is to help end-users to host and play games that are likely to succeed, ie the games won't be canceled nor too few players will participate to these games.

## 2. Business problem

The risk of such a problem is the low user adoption of our app, and for internal user not too react quickly enough. Before having a prediction signal, the assessment of the games probability to succeed was guessed by intuition.

## 3. Solution overview

The architecture of the platform is relatively simple : the end-user app produces business data history, that is consumed by the ai service to train the prediction model, then the end-user app backend consumes the ai signal, by providing data about a planned game, and getting the participation prediction result (indicator of game success).

The signal is a predictive ratio of the game's participation, enriched by a risk level (High/Medium/Low). Each time a game is updated by a player or an host, the signal is recalculated and returned to the product backend.

The signal rendering never blocks nor disturbs the end-user experience : the calls to the signal are time restricted (3 seconds max), and the potential signal service dysfunctions are protected by a fallback process.

## 4. Observed results

The prediction seems to be reliable on "extreme scenarios" such as hosts reliability or delay for game creation. When the model has not enough historic data about the host, it interprets the game as "Low risk", which is optimistic. The model lacks of subtlity with the delays of game creation, for instance a 15 days delay is considered highly risky, which doesn't seems to be right intuitively. The data must be enriched with broader and better quality data.

## 5. Limitations & next steps

### Current limitations

This first version of the prediction is intentionally simple and exploratory.

- The dataset is limited in size and mainly based on historical operational data. Some contextual factors (user motivation, external constraints, last-minute changes) are not captured.
- The prediction does not automate any decision. Final actions always remain in the hands of product and operations teams. These limitations are assumed and aligned with the objective of learning before scaling.

### Potential next steps

Several improvements could be considered in future iterations:

- Add monitoring to compare predicted participation with actual outcomes over time.
- Introduce feedback loops from internal teams to refine thresholds and interpretations.
- Recalibrate or retrain the model as more data becomes available.
- Extend the signal to additional use cases once data maturity increases.

## 6. Product trade-offs

### Simple model instead of complex

We made the choice to put in place a simple model for our prediction, with a simple set of features, because we wanted to avoid putting our efforts in a wrong direction. That choice helped us deliver a quicker solution to the customer's needs, letting us decide later how to improve this model, with the help of user's feedback. In addition, this simple model was good enough regarding the current data quality. We had to compromise the perfect accuracy of the prediction, which can be improved in future releases.

### Simple API exposition for users instead of fancy dashboards

We decided to simply expose the signal through the use of an API endpoint, without providing any dashboards to the users. This decision helped us reducing the cost of misleading about the user's needs. That way, the users will be able to discover the prediction signal in a neutral way at first, and later suggest us what kind of dashboards they would need. The delivery timing of the signal was also positively affected. We may have sacrificied the product adoption for certain users.

### Simple API calls instead of automations

We deliberately decided to implement simple calls to the API (and traces of the signal) in the end-user application, without automating any action in the application. The goal was to avoid any side-effect of the potential inaccuracy of the signal for the end-user, and deliver quickly a first version of the signal. In our opinion, we thought it was better to trade off the potential benefits for the end-user of the automations. In the future we could think about implementing functionnalities such as suggestions, notifications for the end-users.

## 7. Risks

### Data risks

The model was trained on simulated data, as a consequence, the AI service can provide inaccurate results on certain scenarios. The user using the service could make wrong assumptions on certain games that doesn't reflect reality. We can mitigate this risk by returning a custom answer "Unsufficient data" for exceptionnal scenarios.

### Product risks

The signal could be interpreted as a certainty by users. The users could make major decisions that impact negatively the product. We can mitigate this risk by documenting the signal interpretation for the teams of users.

### Technical risks

the AI Service can be unavailable, the users aren't able to make product decisions anymore. Users should be aware that the product decisions are still on the human side, the AI signal is just an hint, not an universal truth.

## 8. Roadmap & next iterations

### Now (0-1 month)

- Monitor and consolidate the AI calls by the API (add an "Insufficient data" response)
- Observe score usage in the teams decisions
- Document signal interpretation guidelines

### Next (1-3 months)

- Compare prediction to the reality
- Adjust risks thresholds (High/Medium/Low)
- Collect internal team's feedback

### Later (3-6 months)

- Add new features to the signal
- Improve model

We don't add new feature to the AI service before having a real usage.