
RAPPORT FINAL TER HAI823I

Memory Lane : Robots That Negotiate by Learning from Past Interactions

Daniel Vargas Vila 22006745

Franck Réveille 22108009

Ramy Massal 22113060

Groupe RDF

Encadrante : Mme Madaline Croitoru

Master 1 Informatique - IASD et GL

Mai 2025



Université de Montpellier

Faculté des Sciences

Departement Informatique



Sommaire

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 1.1 | Objectif du jeu + design | 3 |
| 1.1.1 | Construction des scénarios | 4 |
| 1.2 | Scénarios choisis pour les différentes dimensions | 4 |
| 1.2.1 | Scénarios Risque + Effort | 5 |
| 1.2.2 | Scénarios Risque + Récompense | 6 |
| 1.2.3 | Scénarios Récompense + Effort | 8 |
| 2 | Technologies utilisées | 9 |
| 2.0.1 | Architecture du front end | 9 |
| 2.1 | Frontend | 10 |
| 2.1.1 | Architecture du front end | 10 |
| 2.1.2 | Mise en place | 11 |
| 2.2 | Backend | 15 |
| 2.2.1 | Choix des technologies | 15 |
| 2.2.2 | Architecture du backend | 15 |
| 2.2.3 | Développement des routes | 17 |
| 2.2.4 | Routes principales | 17 |
| 2.3 | Déploiement de l'application | 18 |
| 3 | Collecte de données | 18 |
| 3.1 | Première collecte | 18 |
| 3.1.1 | Problèmes dans la collecte | 19 |
| 3.2 | Deuxième collecte | 19 |
| 3.2.1 | Retour des personnes | 20 |
| 4 | Une première étude des données | 20 |
| 4.1 | Prédiction du modèle sur une seule dimension | 20 |
| 4.2 | Analyse du jeu de données | 20 |
| 4.3 | Méthodes utilisées | 22 |
| 4.3.1 | Tâche de régression | 22 |
| 4.3.2 | Résultats obtenus | 23 |
| 4.3.3 | Conclusion intermédiaire | 24 |
| 4.4 | Tâche de classification binaire | 25 |
| 4.4.1 | Résultats obtenus | 26 |
| 4.4.2 | Conclusion pour cette première approche | 26 |
| 5 | L'apprentissage sur des données incomplètes | 27 |
| 5.1 | Réorganisation du jeu de données | 27 |
| 5.2 | Tâche de régression avec différents modèles et jeux de données | 28 |
| 5.3 | Imputation de données avec deux approches | 28 |
| 5.3.1 | Imputation par KNN | 28 |
| 5.3.2 | Génération de données synthétiques avec ChatGPT | 30 |
| 5.4 | Un deuxième essai avec la tâche de classification binaire | 34 |

| | |
|--|-----------|
| 6 Test de Turing pour tester le modèle obtenu | 37 |
| 6.1 Recherche des profils | 37 |
| 6.2 Résultats obtenus | 37 |
| 7 Lien avec notre sujet original | 38 |
| 8 Conclusion | 39 |
| 9 Lien du Github et de l'application | 39 |

1 Introduction

Dans nos interactions quotidiennes, nous sommes continuellement confrontés à la nécessité de suivre des règles sociales implicites qui régissent la répartition des ressources, le partage des efforts et la gestion des risques. Ces comportements, influencés par des facteurs démographiques tels que l'âge, le genre ou la force perçue, jouent un rôle central dans l'acceptation sociale des individus et, par extension, des agents artificiels. Comprendre comment les humains équilibrivent risque, récompense et danger perçu dans des situations de négociation constitue donc un enjeu fondamental pour l'intelligence artificielle et l'interaction homme-machine.

Ce projet s'inscrit dans une démarche visant à concevoir et implémenter un jeu à objectif sérieux (GWAP – Game With A Purpose) permettant d'étudier la prise de décision humaine dans des scénarios de négociation de ressources. Le jeu, conçu pour collecter des données sur les choix stratégiques des joueurs, servira de support à l'analyse des tendances, biais et stratégies de compromis, en tenant compte de profils démographiques variés. À partir de ces données, nous développeront un modèle d'intelligence artificielle capable de simuler la prise de décision humaine dans des contextes de négociation, en particulier les arbitrages entre risque et récompense. Ce modèle sera ensuite intégré à un robot humanoïde (QT ou Pepper) afin d'illustrer la capacité des agents artificiels à adopter des comportements socialement acceptables.

Dans un premier temps, l'équipe a déjà développé un agent artificiel capable de répartir séparément la récompense, l'effort et le risque entre des dyades d'individus ou de robots, grâce à un jeu dédié à la collecte de données. L'objectif de cette nouvelle étude est d'aller plus loin : il s'agit désormais de concevoir un agent capable de gérer simultanément des combinaisons de ces dimensions (risque et effort, risque et récompense, ou récompense et effort), afin d'approfondir notre compréhension des mécanismes sous-jacents aux décisions sociales complexes.

L'objectif du projet est donc double :

- Collecter des données sur les stratégies de négociation des joueurs dans un jeu GWAP, en intégrant différentes combinaisons de risque, récompense et effort.
- Développer et intégrer un modèle d'intelligence artificielle capable de prédire et simuler les choix humains dans ces contextes, en vue d'une implémentation sur un robot humanoïde.

Cette démarche vise à offrir une compréhension computationnelle fine des interactions sociales, essentielle à l'acceptation et à l'intégration des agents artificiels dans la société.

1.1 Objectif du jeu + design

La conception de ce jeu a nécessité une réflexion approfondie, car il était impératif qu'il soit accessible et compréhensible aussi bien par des enfants dès 6 ans que par des adultes. Ce souci d'accessibilité n'a pas été anodin : il a considérablement rallongé la phase de développement, chaque choix de design étant guidé par la volonté de garantir la clarté des instructions et la simplicité des scénarios. L'objectif principal était d'éviter toute complexité inutile qui aurait pu perturber la compréhension ou biaiser les réponses des participants, ce qui aurait compromis la qualité des données collectées.

Pour éviter d'influencer inconsciemment les décisions des joueurs, le jeu a été conçu de

façon à limiter au maximum les biais potentiels. Les scénarios proposés sont volontairement simples, ancrés dans des situations du quotidien, faciles à comprendre et à visualiser, même pour de jeunes enfants. Les personnages sont également présentés de manière neutre (ex : “personne âgée”, “enfant”, “robot”, “homme/femme petite/grande taille”), sans stéréotypes, afin de ne pas orienter les choix des utilisateurs.

1.1.1 Construction des scénarios

La première étape du design a consisté à imaginer des scénarios d’allocation suffisamment clairs et universels. Nous avons retenu 9 scénarios, répartis selon trois combinaisons d’enjeux sociaux :

- 3 scénarios Risque + Effort
- 3 scénarios Risque + Récompense
- 3 scénarios Récompense + Effort

Chaque scénario met en scène deux personnages choisis aléatoirement parmi l’ensemble suivant : personne âgée, enfant, robot, homme petite taille, femme petite taille, homme grande taille, femme grande taille. Pour chaque situation, les joueurs doivent répartir une ressource (effort, risque ou récompense) sur une échelle de 1 à 10 à l’aide de curseurs (sliders), comme illustré dans la capture d’écran ci-dessous. Cette interface intuitive permet une prise en main immédiate, quel que soit l’âge ou le niveau de familiarité avec les jeux numériques.

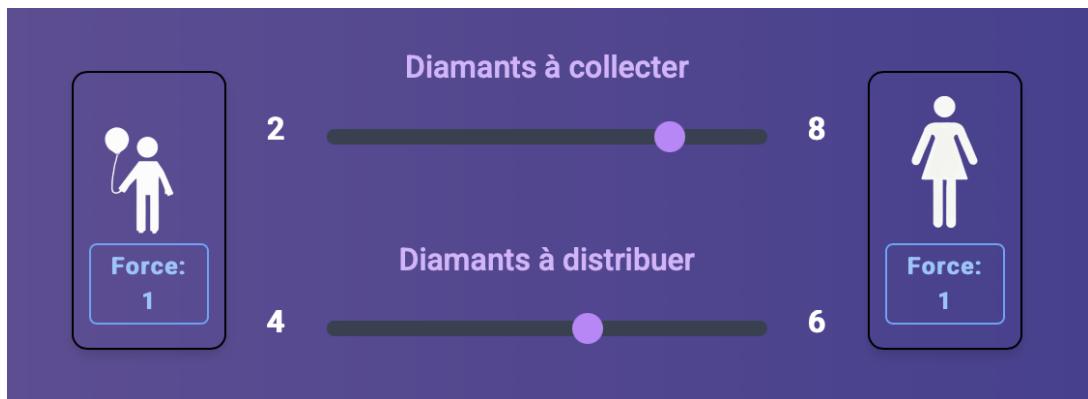


FIGURE 1 – Sliders

Les choix graphiques (couleurs, pictogrammes, textes explicatifs) ont été réalisés pour maximiser la lisibilité et l’engagement, tout en restant sobres et ludiques. L’ensemble du parcours utilisateur a été testé et ajusté pour garantir que chaque étape soit intuitive, sans ambiguïté, et ne suggère aucune “bonne” ou “mauvaise” réponse.

En résumé, la conception du jeu a été guidée par une double exigence : offrir une expérience claire et accessible à tous, et garantir la neutralité des situations afin de collecter des données authentiques, non biaisées par le design ou la formulation des scénarios.

1.2 Scénarios choisis pour les différentes dimensions

Dans cette section, nous présentons les scénarios développés pour chaque combinaison de dimensions (risque + effort, risque + récompense, récompense + effort), ainsi que les textes associés. Ces scénarios ont été conçus pour être simples, anodins et facilement compréhensibles,

tout en mettant en scène des situations du quotidien afin de minimiser les biais et favoriser l'immersion des participants.

De plus, les images associées aux scénarios ont été générées avec ChatGPT.

1.2.1 Scénarios Risque + Effort

Scénario 1 : Éboulement dans une mine

Deux personnages sont bloqués dans une mine à cause d'un éboulement. Pour sortir, ils doivent enlever 10 gros rochers. Mais attention ! Chaque rocher est lourd et les rend fatigués. Certains sont instables et pourraient provoquer un autre éboulement dangereux ! Comment vont-ils s'organiser pour déplacer les rochers et sortir de la mine en sécurité ?

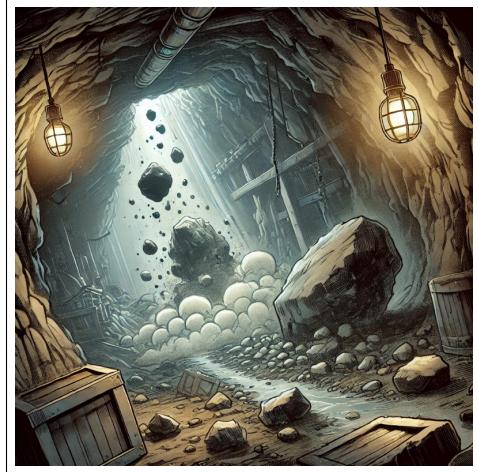


FIGURE 2 – Eboulement dans une mine

Scénario 2 : Eteindre le feu

Deux personnages sont devant un feu de forêt ! Pour l'éteindre, ils doivent apporter de l'eau et la jeter sur les flammes. Apporter l'eau est très fatigant, mais la jeter peut être dangereux à cause de la chaleur. Il faut 10 seaux d'eau en tout. Qui va porter combien de seaux ? Et qui va en jeter combien sur le feu ?



FIGURE 3 – Feu dans une forêt

Scénario 3 : Les vases

Nos deux personnages doivent fabriquer 10 vases et les transporter en marchant sur une étendue d'eau gelée et glissante. Les vases risquent de se casser en les transportant. Combien de vase chaque personnage va fabriquer puis transporter ?

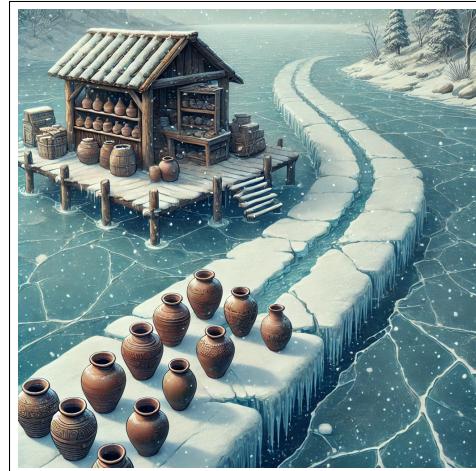


FIGURE 4 – Des vases à construire et transporter

1.2.2 Scénarios Risque + Récompense



FIGURE 5 – Des perles dans l'eau

Scénario 4 : Les perles dans l'eau

Deux personnages plongent sous l'eau pour ramasser 10 perles. Ramasser une perle prend 10 secondes et chacun ne peut plonger qu'une seule fois (2 perles prennent 20 secondes, 3 prennent 30 secondes...). Combien de perles vont-ils ramasser chacun ? Et comment les partager ?

Scénario 5 : Pont des corocodiles

Deux personnages sont de chaque côté d'un vieux pont cassé. Sur le pont il y a des trous et il y a 10 diamants brillants, mais en dessous des trous, des crocodiles attendent ! Plus on avance dans le pont, plus les trous sont grands et plus on risque de se faire manger par les crocodiles. Combien de diamants vont-ils récupérer chacun ? Comment allez-vous partager le trésor ?



FIGURE 6 – Pont des crocodiles



Scénario 6 : Serrures dans l'eau

Deux personnages plongent dans l'eau et trouvent 10 cadenas, du moins profond (1) au plus profond (10). Chacun a une clé et peut ouvrir un seul cadenas. Pour que le coffre plein d'or s'ouvre, les deux personnages doivent choisir deux cadenas de telle sorte que la somme des deux numéros inscrits dans ceux-ci doit soit égale à 10 ! Sinon, le trésor restera caché. Qui ouvre quel cadenas ? Et comment partagez-vous l'or ?

FIGURE 7 – Les serrures dans l'eau

1.2.3 Scénarios Récompense + Effort

Scénario 7 : Pont a 10 000€

Deux personnages doivent construire un pont pour un village. Le projet dure 10 heures et, une fois terminé, les villageois leur verseront une somme de 10 000 €. Comment allez-vous répartir le travail et la récompense entre les deux personnages ?



FIGURE 8 – Pont à 10 000 €

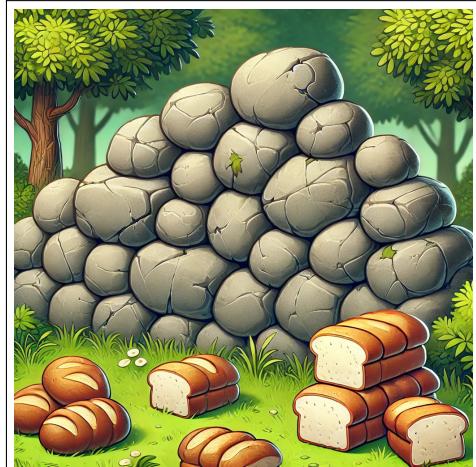


FIGURE 9 – Pains et rocher a deplacer

Scénario 8 : Pains et rochers à déplacer

Des gros rochers bloquent un endroit où il y a de la nourriture. Deux personnages doivent pousser ces rochers pour atteindre les pains cachés derrière. Il y a 10 rochers. Qui va déplacer combien de rochers ? Et comment allez-vous partager les pains entre ces deux personnes ?

Scénario 9 : Coffre de la plage

Il y a 10 marques sur le sol, elles montrent où sont enterrés des coffres. Deux personnages doivent creuser pour les sortir et découvrir ce qu'il y a dedans. Combien de coffres chacun va devoir déterrer ? Et comment partagez-vous le trésor entre ces deux personnes ?



FIGURE 10 – Coffre de la plage

2 Technologies utilisées

Nous avons choisi assez tôt dans la phase de mise en place du projet de nous appuyer sur des outils que nous connaissons déjà. En effet, lors de projets réalisés les années précédentes, nous avons utilisé la stack MERN (Mongo, Express, React, Node.js). Cependant, ce projet représentait une excellente opportunité pour expérimenter et apprendre de nouvelles technologies. C'est pourquoi nous avons décidé de légèrement dévier de notre stack habituelle et d'opter pour la stack MEAN (MongoDB, Express, Angular et Node.js), afin de nous initier à l'utilisation d'Angular. Nous aurions pu utiliser une variante de celle-ci se basant sur Nest.js mais l'apprentissage d'Angular et le fonctionnement de Node.js plus que suffisant pour notre projet nous a fait garder ce dernier.

2.0.1 Architecture du front end

Voici l'architecture de notre front-end qui respecte l'architecture type d'un projet Angular :

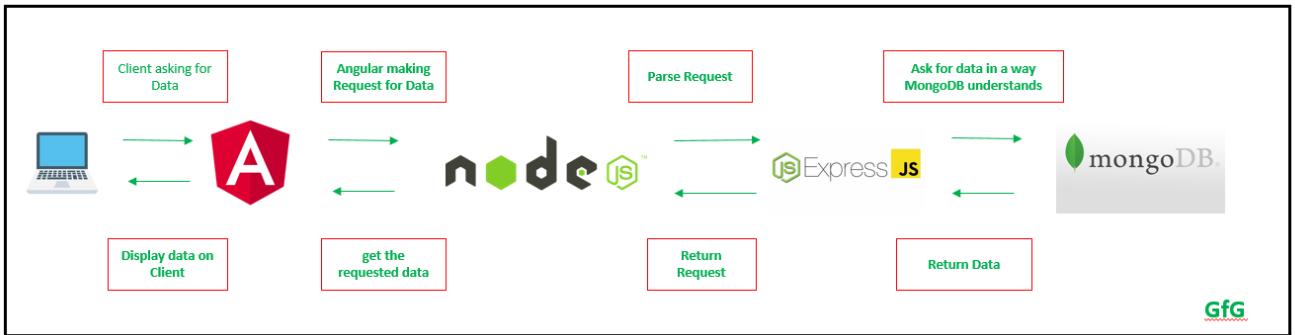


FIGURE 11 – Stack MEAN

2.1 Frontend

2.1.1 Architecture du front end

Voici l'architecture de notre front-end qui respecte l'architecture type d'un projet Angular :

```

frontend/
└── frontend-app/
    ├── .angular/
    ├── dist/
    ├── node_modules/
    └── public/
    └── src/
        └── app/
            ├── component/
            │   ├── contexte/
            │   ├── final-page/
            │   ├── forces-page/
            │   ├── formulaire/
            │   ├── home-page/
            │   ├── img-text/
            │   ├── slider/
            │   ├── stickman1/
            │   └── submit/
            ├── services/
            │   └── data.service.ts
            ├── app.component.css
            ├── app.component.html
            ├── app.component.spec.ts
            ├── app.component.ts
            ├── app.config.ts
            └── app.routes.ts
        └── assets/

```

FIGURE 12 – Architecture Front

Premièrement, le dossier assets contient exclusivement les fichiers image nécessaires à

l'application, qu'il s'agisse des illustrations des personnages ou des images associées aux scénarios.

Ensuite le fichier app.routes.ts définit la table de routage de l'application. Les routes définies ici servent au routage dynamique de composants via la balise spéciale d'angular : <router-outlet>

Nous avons également une unique classe de service Angular, fournissant une interface avec notre back-end via un lot de méthodes. Cette interface permet de séparer la logique web de la logique pur de nos composants.

Enfin, nous avons les composants. Les composants forment le cœur d'une application Angular. Ils structurent l'interface utilisateur en affichant dynamiquement les données, les formulaires et les scénarios selon l'état de l'application. Ce sont eux qui permettent la nature multi-pages d'Angular, permettant des changements d'UI fluide sans recharge de la page.

Voici un diagramme montrant "l'assemblage" des composants. Le composant "img-text" inclus statiquement ses enfants.

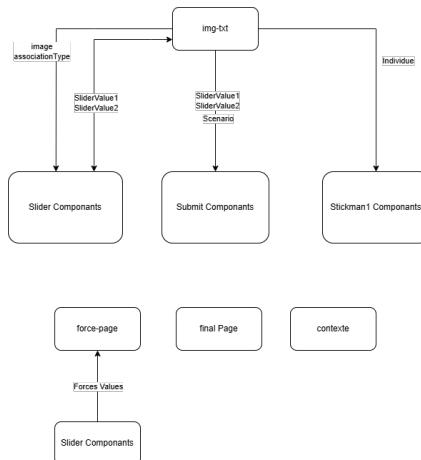


FIGURE 13 – Architecture Front

Nous avons aussi ajouté des composants material déjà existant durant notre projets.

2.1.2 Mise en place

2.1.2.1 Page principale

La majorité du jeu se passe dans le composant "img-text". Celui-ci orchestre la logique de chaque manche et est donc le premier que nous avons développés.

Sa méthode `ngOnInit()` va s'occuper de récupérer les données du scénario correspondant à la manche courante (scénario, personnages mis en opposition ainsi que leurs forces), incrémenter le compteur de tour mais surtout de vérifier que le recharge de la page ne fasse pas passer à un nouveau scénario. Les données du scénario sont conservés dans un attribut de classe qui est un objet TypeScript contenant de nombreuses informations sur ce dernier.

Le composant est séparés en trois parties logiques, dont certaines implémentées en incluant statiquement le composant correspondant (directement via leur sélecteur, plutôt que par un routage dynamique).

La partie "scénario", contenant l'image et le texte du scénario de la manche courante :

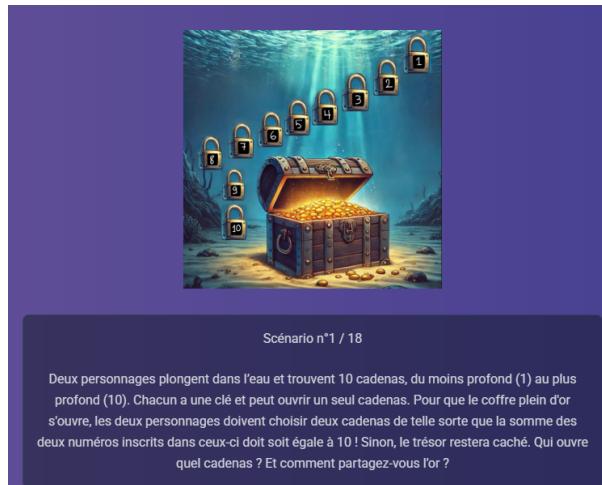


FIGURE 14 – Scénario

Le composant `img-txt` possède parmis ses attributs, un de type `string` correspondant à l'URL de l'image associé au scénario. Le texte du scénario est obtenu via le champ `text` de l'objet `scénario`. Ces données permettent d'afficher dynamiquement le contenu du scénario sur lequel l'utilisateur va réfléchir à l'écran.

Vient ensuite la section consacrée à la réponse de l'utilisateur, elle-même structurée en deux sous-parties.

1. Les personnages :

Le composant `img-txt` récupère les deux personnages désignés aléatoirement par le serveur via les champs `individuA` et `individuB` de l'objet `scénario`. Le composant `stick-man1` est ensuite inclus statiquement deux fois, en passant à chaque composant le bon individu via le passage de propriétés de parents à enfants d'Angular.

2. Le composant `slider` :

Ce composant a deux utilisations principales dans notre application. Dans ce contexte précis, il est utilisé pour permettre à l'utilisateur de répartir sur une échelle de 0 à 10 la distribution des dimensions à chaque personnage. Par exemple, si on est dans une situation **risk-reward**, il permet d'attribuer un taux de risque, puis un taux de récompense à chaque personnage. Son comportement est défini via une directive Angular, qui permet via une conditionnelle directement dans le code HTML du composant de modifier son apparence.

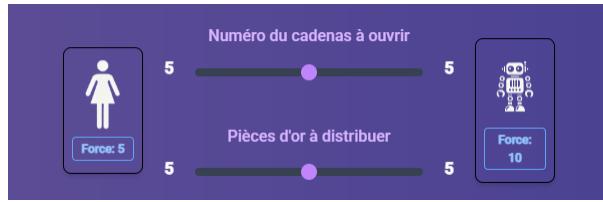


FIGURE 15 – Slider

Le bouton submit :

Enfin, nous avons le bouton `submit`, qui joue un rôle essentiel dans le déroulement du scénario. Il récupère les différentes valeurs nécessaires à l'enregistrement de la réponse : les types des personnages, leurs forces respectives, ainsi que les valeurs issues des deux `sliders` pour chacun d'eux.

Ces données sont ensuite envoyées au *backend* à chaque validation de question, permettant ainsi une collecte structurée et répétée des réponses tout au long de la session.



FIGURE 16 – Boutton Submit

2.1.2.2 Page d'accueil



FIGURE 17 – Boutton Submit

La page d'accueil est une page de présentation simple, affichée à l'arrivée sur le site. Elle permet aux participants de comprendre rapidement le contexte et le but du jeu ou de l'expérience proposée.

L'interaction est volontairement minimale : un seul bouton est disponible, permettant de passer à la page suivante et ainsi démarrer la session.

2.1.2.3 Formulaire utilisateur

Vient ensuite le formulaire, qui doit être rempli par l'utilisateur avant de commencer l'expérience. Celui-ci nous permet de recueillir certaines informations sociodémographiques

(genre, âge, nationalité, niveau d'études), dans le but de regrouper et classer les données lors de l'analyse finale.

Les données sont entièrement anonymisées. De plus, dans le cadre de ce projet universitaire, nous veillons à respecter les principes du RGPD en ne collectant aucune donnée personnelle identifiable, et en utilisant les informations uniquement à des fins de recherche pédagogique.

The screenshot shows a blue-themed form titled "Informations du Joueur". It contains four dropdown menus: "Quel est votre genre?", "Quel est votre âge?", "Quelle est votre nationalité?", and "Quel est votre niveau d'études?". Each dropdown has a pre-selected value: "Homme", "24", "Française (France)", and "Bac+4" respectively. Below the dropdowns is a large orange button labeled "Valider".

FIGURE 18 – Boutton Submit

2.1.2.4 Page des forces

Avant de commencer la session, une page intermédiaire s'affiche pour expliquer aux participants la nécessité d'attribuer une force à chaque personnage.

Cette étape est essentielle au bon déroulement du jeu, car ces forces seront utilisées dans les différents scénarios pour évaluer et comparer les décisions prises par les participants.



FIGURE 19 – Texte des Forces

Pour permettre à l'utilisateur de sélectionner les forces, nous utilisons la seconde fonctionnalité du composant `slider`. Celui-ci permet d'attribuer une valeur à chaque personnage, valeur qui sera ensuite sauvegardée et utilisée tout au long du jeu.

Ainsi, les forces définies à cette étape seront associées à chaque personnage et prises en compte dans les scénarios suivants pour l'analyse des décisions.



FIGURE 20 – Tableau des forces

2.2 Backend

2.2.1 Choix des technologies

Pour le backend de notre projet, nous avons choisi d'utiliser les technologies suivantes :

- **Node.js + Express** : Cette combinaison populaire basé sur Node.js et son framework Express offre une solution simple et suffisante pour développer notre serveur dans le cadre de notre TER. De plus, tous les membres du groupe connaisse bien cette technologie ce qui a très largement fluidifié le développement.
- **MongoDB** : Base de données NoSQL orientée documents, particulièrement adaptée aux applications web modernes. Elle offre une grande flexibilité dans la modélisation des données, notamment grâce au format *JSON*, ce qui facilite l'intégration avec *Node.js* et *Express*.

2.2.2 Architecture du backend

Notre back-end suit une architecture Monolithique, c'est-à-dire que l'ensemble des fonctionnalités back-end (API, logique métier, interrogation de la base de données) est regroupé dans un seul serveur déployé en un seul bloc. Cette approche fonctionne totalement dans le cadre de notre application, qui n'a pas besoin de gérer énormément de connexions simultanées (une centaine tout au plus) et permet en outre de simplifier significativement le développement et le déploiement.

Voici l'architecture du back-end :

```

backend/
|
|   config/
|   |   └── db_conn.js
|
|   models/
|   |   ├── formModel.js
|   |   ├── responseModel.js
|   |   ├── textModel.js
|   |   └── README.md
|
|   routes/
|   |   ├── Route.js
|   |   └── README.md
|
|   node_modules/
|
|   .env
|   index.js
|   package.json
|   package-lock.json
└── routes.js

```

FIGURE 21 – Architecture du back-end

- **Serveur Express** : Nous avons mis en place un serveur *Express* minimal, configuré pour gérer les requêtes HTTP et communiquer avec la base de données *MongoDB*.
- **Gestion de la configuration** : pour garantir la sécurité et la séparation des configurations, toutes les informations sensibles (comme l’URI de la base de données) sont stockées dans un fichier `.env`. Ces variables sont ensuite utilisées via un fichier dédié (`db-conn.js`), ce qui permet une gestion centralisée et sécurisée des accès.
- **Modèles de données** : nous avons défini trois modèles principaux :
 - **Text** : permet d’enregistrer les scénarios prédéfinis du jeu. Chaque document contient l’URL de l’image associée, le texte du scénario et son type (par exemple : *risk-reward*, *effort*, etc.).
 - **Form** : utilisé pour stocker les informations relatives aux utilisateurs, telles que l’âge, le genre, la nationalité et le niveau d’étude.
 - **Response** : permet d’enregistrer les réponses individuelles des utilisateurs. Pour assurer la cohérence entre les réponses et le profil utilisateur, nous utilisons un identifiant de session (`sessionId`) commun entre les modèles **Form** et **Response**.

2.2.3 Développement des routes

Le développement des routes s'est fait en parallèle du frontend, avec plusieurs itérations pour s'adapter aux nouvelles décisions prises au fil du projet. Nous avons retravaillé et affiné les routes à chaque étape, afin de garantir la cohérence et la robustesse de l'API.

2.2.4 Routes principales

Les routes sont définies le fichier Route.js. Elles sont ajoutées au middleware Router d'express. Cette organisation permet d'encapsuler la logique des routes de notre API proprement sans noyer le code du serveur.

Voici une explication des routes notre API :

1. **Route /init :**

Cette route initialise une session pour un utilisateur. Elle récupère tous les scénarios disponibles, les duplique, les mélange aléatoirement, puis sélectionne un scénario à afficher. Elle associe également deux personnages générés aléatoirement puis ajoute le tout dans la session.

2. **Route /next :**

Permet d'accéder au scénario suivant en extrayant le prochain élément de la liste initialisée via /init. Elle met à jour le scénario courant et incrémente le compteur de tours.

3. **Route /submitForm :**

Cette route enregistre les informations socio-démographiques renseignées par l'utilisateur (âge, genre, nationalité, niveau d'étude), en les associant à un identifiant de session unique.

4. **Route /submit :**

Elle enregistre la réponse de l'utilisateur à un scénario donné. Elle collecte les valeurs issues des sliders, les forces attribuées à chaque personnage et les stocke dans la base de données sous forme de document Response, avec l'identifiant de session associé.

5. **Route /reset-session :**

Réinitialise les données de session tout en conservant éventuellement des informations utilisateur si nécessaire. Cela permet de relancer une session proprement sans perturber la logique du site.

6. **Route /compute-stats :**

Traite les scores reçus pour chaque personnage et chaque catégorie (risk, reward, effort). Elle calcule la moyenne et identifie le personnage le plus performant dans chaque dimension.

7. **Fonction getIndividus() :**

Génère aléatoirement deux personnages différents parmis la liste définie.

8. **Fonction initScore() :**

Initialise la structure des scores pour tous les personnages et toutes les catégories. Chaque entrée contient un score cumulé et un compteur pour le calcul des moyennes.

2.3 Déploiement de l'application

Une fois le jeu finalisé, l'étape suivante consistait à déployer l'application afin de la rendre accessible au public. Pour cela, nous avons utilisé **Vercel**, une plateforme de déploiement en continu très populaire pour les applications front-end, notamment celles développées avec des frameworks comme **Angular**. Vercel permet une mise en ligne rapide et une gestion efficace des différentes versions grâce à l'intégration avec **Git**.

Cependant, cette étape s'est révélée plus complexe que prévu. Bien que l'application fonctionnait parfaitement en local, de nombreux problèmes sont apparus une fois en production. Nous avons notamment rencontré des problèmes de session, des dysfonctionnements dans l'affichage des scénarios, et d'autres bugs imprévus. Il nous a fallu analyser minutieusement les logs fournis par Vercel pour identifier et corriger ces anomalies.

Plusieurs versions du jeu ont ainsi été déployées successivement, en lien avec les ajustements décrits dans les sections précédentes, afin d'améliorer la clarté des scénarios et de la jouabilité.

L'un des bugs les plus importants détectés lors du dernier déploiement concernait un double appel à une route backend. Ce problème entraînait un décalage dans la séquence des scénarios dans notre base de données, faussant ainsi les réponses des joueurs. Ce type de bug, difficilement visible au premier abord, nous a rappelé l'importance de ne pas reporter les "petits" problèmes techniques, car ils peuvent avoir un impact significatif à grande échelle.

Pour faciliter l'accès à l'application, nous avons également généré un QR code menant directement à l'interface du jeu. Cela nous permet, entre autres, de diffuser plus facilement le lien et de suivre l'activité des joueurs, notamment à travers les statistiques d'usage.

Cette phase de déploiement étant terminée, nous avons pu commencer la collecte des données : des sessions de jeu ont été organisées, lors desquelles nous avons invité des participants à jouer, afin de récupérer leurs réponses. Ces données constituent la base essentielle pour la suite de notre projet : l'analyse et la modélisation via des approches de machine learning.

3 Collecte de données

La collecte des données a débuté légèrement plus tard que prévu, principalement en raison de difficultés techniques rencontrées lors du développement du jeu. Ces retards ont été occasionnés par la nécessité d'assurer la stabilité de la plateforme, la clarté des scénarios, ainsi que la neutralité et l'accessibilité de l'interface pour tous les utilisateurs, notamment les enfants. Une fois ces ajustements effectués, la phase de collecte a pu être lancée dans de bonnes conditions.

3.1 Première collecte

Pour cette première étape, près de 300 personnes ont été sollicitées afin de participer à l'expérience. L'échantillon comprend des profils variés : amis, membres de la famille, camarades de classe, ainsi que des personnes issues de différents milieux sociaux et tranches d'âge. Cette diversité vise à garantir la représentativité des données collectées et à permettre une analyse fine des comportements selon différents facteurs démographiques. La participation volontaire

de ce large panel a permis d'obtenir un volume de données significatif, essentiel pour l'analyse des tendances et la validation des hypothèses initiales concernant la prise de décision dans des situations de répartition des ressources. Les retours des participants ont également permis d'identifier d'éventuels points d'amélioration dans la conception du jeu et dans la formulation des scénarios, afin d'optimiser la qualité des données pour les phases ultérieures du projet.

3.1.1 Problèmes dans la collecte

Suite à notre première collecte de données, nous avons constaté une erreur dans la méthode de collecte. En effet, un décalage d'une question par rapport aux réponses a été détecté. Après investigation, nous avons identifié que ce décalage était causé par une double initialisation de la session, ce qui provoquait l'affichage de la première question sans enregistrement correct de la réponse correspondante.

Même après correction du bug, nous avons estimé que les données collectées précédemment étaient invalides. Bien qu'il aurait été possible de realigner les réponses en décalant les indices, cela n'aurait pas résolu le problème principal : les participants ont répondu à des associations erronées.

Nous avons donc pris la décision de retirer ces réponses du jeu de données, afin de garantir la fiabilité de notre analyse.

Nous avons également rencontré un autre problème important. Initialement, nous avions conçu notre système de collecte de données de manière à enregistrer chaque réponse individuellement, sans l'associer explicitement à un utilisateur.

Cependant, nous avons rapidement réalisé que, pour que ces données soient pertinentes sur le plan analytique, il était essentiel de pouvoir les relier aux informations des participants (âge, genre, nationalité, niveau d'étude). Cela permet notamment d'étudier les corrélations entre profils utilisateurs et types de décisions.

À partir de ce constat, nous avons revu la structure de notre base de données, ainsi que le format des réponses enregistrées, afin d'y intégrer un identifiant de session commun entre les réponses et les formulaires utilisateurs. En conséquence, toutes les réponses collectées avant cette refonte ont dû être écartées, car elles n'étaient associées à aucun profil utilisateur et ne permettaient donc aucune analyse exploitable.

3.2 Deuxième collecte

La deuxième phase de collecte de données a été marquée par un volume de participation moins important que prévu, principalement en raison d'un problème technique lié au QR Code utilisé pour accéder au jeu. En effet, le QR Code affiché par la professeure dans la salle de cours présentait une erreur, empêchant certains étudiants de rejoindre la plateforme et de participer à l'expérience. Ce dysfonctionnement a eu pour conséquence de limiter la collecte, nous privant d'environ 100 réponses supplémentaires.

Malgré ce contretemps, nous avons poursuivi la diffusion du jeu auprès de nouvelles personnes, ce qui a permis d'atteindre un total de 129 participants. Parmi eux, 104 ont réellement pris part à l'expérience et ont complété le jeu dans son intégralité.

Il est important de préciser que, afin de garantir la validité des données et d'éviter tout biais, il n'a pas été possible de solliciter à nouveau les personnes ayant déjà participé à la

première collecte. Cette contrainte a limité la rapidité de la collecte, mais elle était nécessaire pour préserver l'intégrité des résultats et assurer la diversité des profils des joueurs.

3.2.1 Retour des personnes

Les participants ayant testé le jeu ont globalement exprimé une appréciation positive de l'expérience ludique proposée. Ils ont notamment salué la simplicité des mécaniques, l'aspect intuitif de l'interface ainsi que la variété des scénarios proposés, qui rendent le jeu accessible et engageant.

Cependant, plusieurs retours ont mis en lumière des difficultés liées à la motivation à poursuivre jusqu'à la fin du parcours. Un certain nombre de joueurs ont confié ressentir de la lassitude ou “avoir la flemme” de compléter l'ensemble des scénarios, notamment à cause du volume de texte à lire à chaque étape. Cette surcharge de lecture a parfois été perçue comme un frein à l'immersion et à la fluidité du jeu, incitant certains participants à abandonner avant d'avoir terminé.

4 Une première étude des données

Une fois la phase de collecte terminée, nous disposions enfin d'un jeu de données complet contenant les réponses des participants. La première étape de notre analyse consistait à tester différents modèles d'apprentissage automatique afin d'évaluer leur capacité à prédire les décisions des joueurs face aux scénarios proposés.

4.1 Prédiction du modèle sur une seule dimension

Comme expliqué dans les sections précédentes, chaque scénario comporte deux dimensions sociales parmi les suivantes : risk-effort, risk-reward ou effort-reward. Cela signifie que pour chaque situation, les participants doivent répartir une quantité donnée sur deux axes de décision.

Pour simplifier le problème, notre encadrante nous a d'abord demandé de construire un modèle qui ne prédit qu'une seule dimension à la fois. Par exemple, si l'on choisit de modéliser la dimension effort, alors le modèle doit uniquement prédire la quantité d'effort attribuée à un individu dans les scénarios contenant la dimension effort (c'est-à-dire les scénarios de type effort-reward ou risk-effort). Les scénarios ne comportant pas cette dimension (par exemple risk-reward) sont ignorés dans cet apprentissage.

Cette première approche nous permettait de réduire la complexité du problème initial, d'isoler les performances du modèle sur une dimension sociale précise et d'identifier d'éventuels biais ou tendances dans les décisions des participants sur cette dimension.

Nous avons donc testé différents modèles de classification ou de régression selon les cas, afin de mesurer leur performance sur cette tâche ciblée.

4.2 Analyse du jeu de données

Comme vous le savez, notre jeu de données est issu de notre jeu interactif composé de 18 scénarios présentés à chaque joueur. Chaque scénario demande au joueur de répartir des

valeurs (effort, risque, récompense) entre deux individus, en fonction de leur profil et du contexte.

Au total, **129 joueurs** se sont enregistrés pour participer. Parmi eux :

- **104 joueurs** ont effectivement joué au moins un scénario,
- **29 joueurs** n'ont pas terminé le jeu et ont répondu à moins de 18 scénarios,
- **64 joueurs** ont complété exactement les 18 scénarios attendus,
- **11 joueurs** ont joué plus de 18 scénarios (ils ont recommencé ou poursuivi au-delà d'une session complète).

Cette distribution met en évidence un déséquilibre important entre joueurs complets et incomplets, ce qui représente un défi pour l'entraînement de modèles cohérents. Le traitement des sessions incomplètes, que nous aborderons via des techniques d'imputation plus tard, est donc une composante essentielle de notre analyse.

Par ailleurs, chaque ligne du jeu de données correspond à une réponse d'un joueur à un scénario spécifique. Cela signifie que pour un joueur ayant complété les 18 scénarios, 18 lignes sont présentes. Les colonnes contiennent à la fois :

- des informations sur le joueur (âge, genre, nationalité, niveau d'études),
- les deux personnages du scénario (type A et type B),
- les forces physiques attribuées à chaque personnage par le joueur (de 1 à 10),
- et les valeurs de répartition choisies (effort, risque ou récompense).

Cette structuration nous permet d'analyser à la fois les comportements individuels et les tendances globales à travers différents types de scénarios.

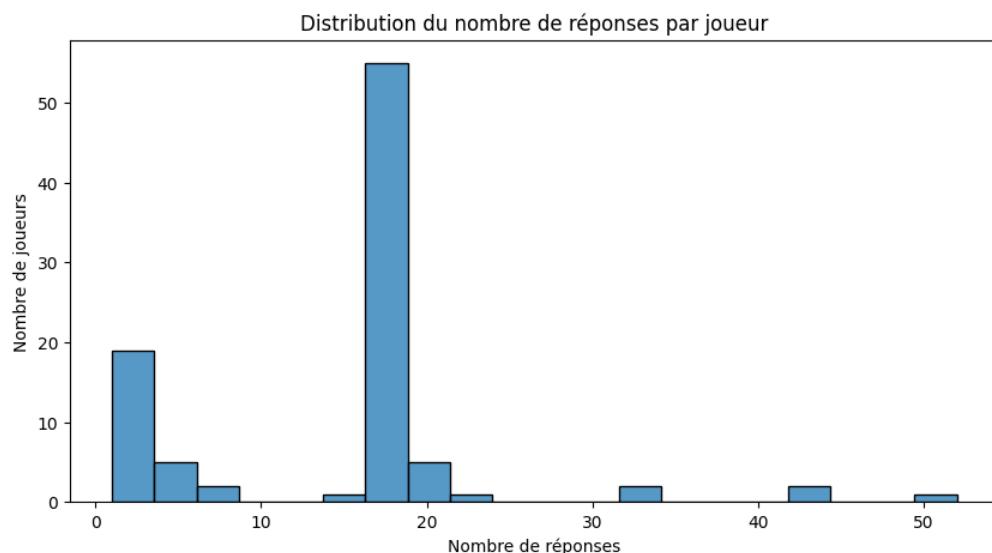


FIGURE 22 – Diagramme de barres représentant le nombre de réponses par joueur

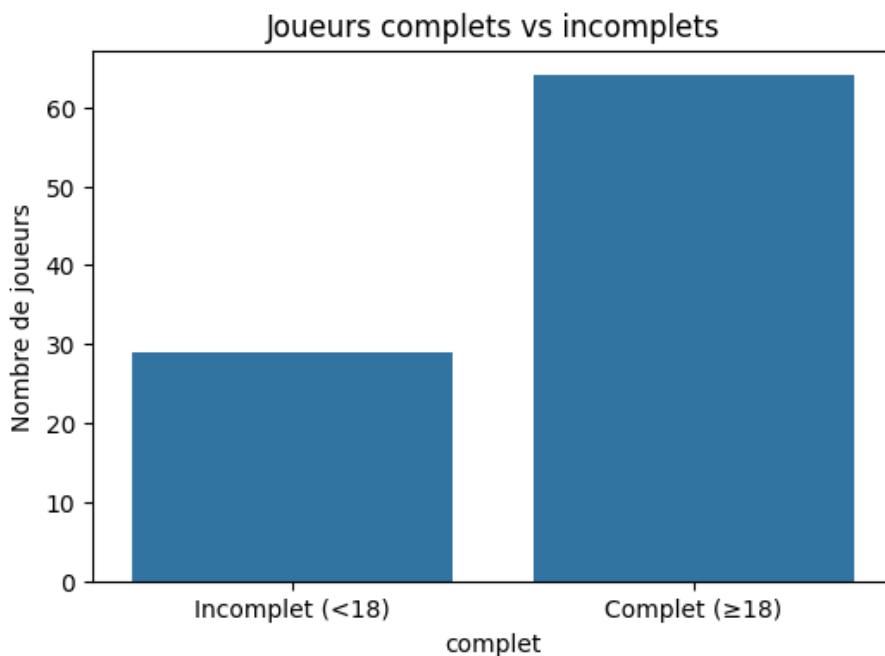


FIGURE 23 – Diagramme de barres - Joueurs complets vs incomplets

4.3 Méthodes utilisées

4.3.1 Tâche de régression

Dans une première approche, nous avons formulé le problème comme une tâche de régression multivariée. L'objectif est de prédire, pour chaque scénario, les valeurs numériques attribuées par le joueur à deux personnages (individus A et B), selon une dimension spécifique (effort, risque ou récompense). Ces valeurs vont de 0 à 10 et représentent des répartitions réalisées par le joueur en fonction du contexte.

Afin d'évaluer les performances des modèles de régression, nous avons utilisé trois métriques standards :

- **MAE (Mean Absolute Error)** : moyenne des valeurs absolues des écarts entre les prédictions et les vraies valeurs. Plus cette valeur est faible, meilleure est la performance du modèle. Un MAE de 0 signifie que le modèle prédit parfaitement toutes les valeurs.
- **RMSE (Root Mean Squared Error)** : racine carrée de la moyenne des erreurs au carré. Cette métrique pénalise davantage les grandes erreurs. Comme pour la MAE, plus c'est bas, mieux c'est.
- **R² (coefficients de détermination)** : proportion de la variance des données expliquée par le modèle. Un R² proche de 1 indique que le modèle explique bien les variations des réponses. Une valeur proche de 0 signifie que le modèle n'apporte aucune amélioration par rapport à une simple moyenne, et une valeur négative indique un modèle moins bon qu'un prédicteur naïf.

Pour illustrer ces métriques, supposons que les vraies valeurs soient [5, 6, 7] et que le modèle prédit [4, 6, 8].

- $\text{MAE} = \frac{|4-5|+|6-6|+|8-7|}{3} = \frac{2}{3} \approx 0.67$
- $\text{RMSE} = \sqrt{\frac{(4-5)^2+(6-6)^2+(8-7)^2}{3}} = \sqrt{\frac{2}{3}} \approx 0.82$
- R^2 dépend de la variance de [5, 6, 7] et de l'erreur résiduelle, mais dans cet exemple, il serait relativement élevé car les prédictions suivent globalement les vraies tendances.

Ces trois indicateurs complémentaires permettent d'avoir une vision globale de la qualité des modèles de régression entraînés.

4.3.2 Résultats obtenus

| Résultats pour la dimension Risk : | | | | | |
|------------------------------------|-------------------|---------------|---------------|----------------|--|
| | Model | MAE | RMSE | R2 | |
| 0 | Random Forest | 1.899 ± 0.138 | 2.632 ± 0.200 | 0.108 ± 0.139 | |
| 1 | Gradient Boosting | 1.985 ± 0.129 | 2.637 ± 0.149 | 0.107 ± 0.103 | |
| 2 | Linear Regression | 2.009 ± 0.099 | 2.591 ± 0.140 | 0.139 ± 0.087 | |
| 3 | XGBoost | 2.060 ± 0.165 | 2.912 ± 0.208 | -0.090 ± 0.157 | |
| 4 | LightGBM | 1.978 ± 0.094 | 2.676 ± 0.129 | 0.081 ± 0.095 | |

| Résultats pour la dimension Effort : | | | | | |
|--------------------------------------|-------------------|---------------|---------------|---------------|--|
| | Model | MAE | RMSE | R2 | |
| 0 | Random Forest | 1.637 ± 0.073 | 2.265 ± 0.096 | 0.355 ± 0.054 | |
| 1 | Gradient Boosting | 1.669 ± 0.056 | 2.213 ± 0.099 | 0.384 ± 0.061 | |
| 2 | Linear Regression | 1.764 ± 0.090 | 2.339 ± 0.103 | 0.314 ± 0.035 | |
| 3 | XGBoost | 1.737 ± 0.088 | 2.438 ± 0.124 | 0.253 ± 0.072 | |
| 4 | LightGBM | 1.707 ± 0.065 | 2.253 ± 0.108 | 0.361 ± 0.065 | |

| Résultats pour la dimension Reward : | | | | | |
|--------------------------------------|-------------------|---------------|---------------|---------------|--|
| | Model | MAE | RMSE | R2 | |
| 0 | Random Forest | 1.452 ± 0.100 | 2.032 ± 0.117 | 0.231 ± 0.046 | |
| 1 | Gradient Boosting | 1.530 ± 0.089 | 2.059 ± 0.108 | 0.210 ± 0.043 | |
| 2 | Linear Regression | 1.634 ± 0.114 | 2.267 ± 0.113 | 0.043 ± 0.036 | |
| 3 | XGBoost | 1.588 ± 0.068 | 2.275 ± 0.066 | 0.034 ± 0.051 | |
| 4 | LightGBM | 1.532 ± 0.088 | 2.097 ± 0.107 | 0.181 ± 0.032 | |

FIGURE 24 – Résultats pour chaque dimension séparément

Et voici un diagramme de barres pour voir quel modèle se débrouille le mieux en moyenne :

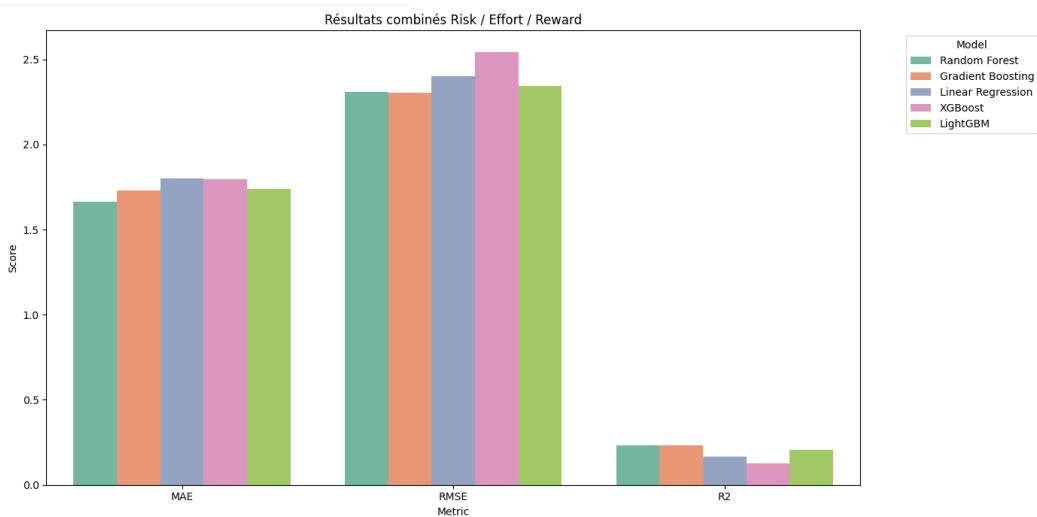


FIGURE 25 – Résultats de chaque modèles pour les 3 dimensions combinées

Les modèles ont été entraînés pour prédire les scores attribués à deux individus (A et B) sur trois dimensions : **effort**, **risque** et **récompense**. L'évaluation repose sur trois métriques standards : **MAE**, **RMSE** et **R²**.

- **MAE (Mean Absolute Error)** : Les valeurs observées sont comprises entre **1,8 et 2,1**, ce qui signifie que, en moyenne, les modèles se trompent d'environ **2 points**. Ce niveau d'erreur est modéré mais montre que les prédictions ne sont pas encore pleinement satisfaisantes.
- **RMSE (Root Mean Squared Error)** : Les valeurs varient entre **2,5 et 2,8**, systématiquement supérieures au MAE. Cela indique que certaines prédictions sont particulièrement éloignées de la vérité, le RMSE étant sensible aux erreurs importantes.
- **R² (coefficients de détermination)** : Les scores se situent autour de **0,2 à 0,3**, ce qui signifie que les modèles expliquent environ **20 à 30 %** de la variance des données. Ce niveau est relativement faible et montre que les modèles ne capturent qu'une partie limitée des dynamiques sous-jacentes.

4.3.3 Conclusion intermédiaire

Ces résultats traduisent une **performance limitée** des modèles actuels, probablement liée à plusieurs facteurs :

- la **complexité de la tâche**, qui demande de prédire plusieurs valeurs continues par observation,
- la **présence de données manquantes** ou incomplètes,
- la **formulation du problème** en régression, qui peut être sous-optimale pour ce type de choix préférentiel.

Une autre approche consisterait de reformuler le problème en une **tâche de classification binaire**, en se concentrant non plus sur la valeur exacte donnée à chaque individu, mais sur le **choix relatif entre A et B**. Cela permettrait de simplifier le cadre du problème et potentiellement d'augmenter la stabilité et la précision des prédictions.

4.4 Tâche de classification binaire

En parallèle de la régression, nous avons formulé une tâche de classification binaire visant à prédire, pour chaque scénario, lequel des deux individus (A ou B) reçoit une part plus importante d'une dimension donnée (effort, risque ou récompense). L'idée est de simplifier la tâche d'apprentissage en transformant une prédition numérique en un choix binaire : "A reçoit plus que B" (1) ou "B reçoit plus ou égal à A" (0).

Cela permet de capter le comportement du joueur de manière plus qualitative : qui, selon lui, mérite ou doit assumer davantage dans la situation donnée.

Pour évaluer les modèles de classification, nous utilisons quatre métriques classiques :

- **Accuracy (exactitude)** : proportion de bonnes prédictions parmi l'ensemble. C'est une mesure globale, mais elle peut être trompeuse en cas de classes déséquilibrées.
- **Precision (précision)** : proportion de vraies prédictions positives parmi toutes les prédictions positives. Elle indique dans quelle mesure le modèle évite de faire de fausses alertes.
- **Recall (rappel)** : proportion de vraies prédictions positives parmi toutes les vraies valeurs positives. Elle mesure la capacité du modèle à bien identifier tous les cas positifs.
- **F1-score** : moyenne harmonique entre précision et rappel. Cette métrique est particulièrement utile lorsque les classes sont déséquilibrées, car elle pénalise les écarts entre précision et rappel.

Soit une cible binaire indiquant si l'individu A a reçu plus que B pour l'effort :

- Si la vérité est : [1 , 0 , 1 , 1 , 0] (A a eu plus dans 3 cas sur 5)
- et que le modèle prédit : [1 , 1 , 1 , 0 , 0]

Alors :

- **Accuracy** = $3/5 = 0.60$
- **Precision** = $2/3 \approx 0.67$ (2 vrais positifs sur 3 prédits)
- **Recall** = $2/3 \approx 0.67$ (2 vrais positifs sur 3 attendus)
- **F1-score** ≈ 0.67

Ces indicateurs permettent de mesurer la capacité du modèle à reproduire les choix humains de manière qualitative.

4.4.1 Résultats obtenus

Résultats pour la dimension Effort

 Classification binaire who_more_oneA
Accuracy : 0.769 ± 0.015
Precision : 0.708 ± 0.035
Recall : 0.582 ± 0.045
F1 : 0.637 ± 0.021

 Classification binaire who_more_twoA
Accuracy : 0.779 ± 0.011
Precision : 0.653 ± 0.021
Recall : 0.499 ± 0.071
F1 : 0.562 ± 0.042

 Classification binaire who_more_oneB
Accuracy : 0.731 ± 0.015
Precision : 0.660 ± 0.090
Recall : 0.530 ± 0.039
F1 : 0.580 ± 0.055

 Classification binaire who_more_twoB
Accuracy : 0.742 ± 0.025
Precision : 0.650 ± 0.048
Recall : 0.488 ± 0.062
F1 : 0.556 ± 0.055

 Classification binaire who_equal_oneB
Accuracy : 0.774 ± 0.025
Precision : 0.653 ± 0.057
Recall : 0.451 ± 0.064
F1 : 0.531 ± 0.055

 Classification binaire who_equal_twoA
Accuracy : 0.712 ± 0.023
Precision : 0.649 ± 0.046
Recall : 0.532 ± 0.057
F1 : 0.581 ± 0.027

Résultats pour la dimension Risk

 Classification binaire who_more_oneA
Accuracy : 0.751 ± 0.030
Precision : 0.672 ± 0.070
Recall : 0.491 ± 0.045
F1 : 0.567 ± 0.052

 Classification binaire who_more_twoA
Accuracy : 0.799 ± 0.025
Precision : 0.647 ± 0.079
Recall : 0.447 ± 0.055
F1 : 0.528 ± 0.060

 Classification binaire who_more_oneB
Accuracy : 0.758 ± 0.031
Precision : 0.694 ± 0.081
Recall : 0.573 ± 0.055
F1 : 0.625 ± 0.053

 Classification binaire who_more_twoB
Accuracy : 0.746 ± 0.030
Precision : 0.667 ± 0.058
Recall : 0.492 ± 0.051
F1 : 0.566 ± 0.054

 Classification binaire who_equal_oneB
Accuracy : 0.778 ± 0.018
Precision : 0.690 ± 0.054
Recall : 0.520 ± 0.028
F1 : 0.593 ± 0.038

 Classification binaire who_equal_twoA
Accuracy : 0.743 ± 0.030
Precision : 0.708 ± 0.037
Recall : 0.636 ± 0.040
F1 : 0.670 ± 0.034

Résultats pour la dimension Reward

 Classification binaire who_more_oneA
Accuracy : 0.811 ± 0.023
Precision : 0.743 ± 0.029
Recall : 0.706 ± 0.068
F1 : 0.722 ± 0.041

 Classification binaire who_more_twoA
Accuracy : 0.795 ± 0.030
Precision : 0.636 ± 0.070
Recall : 0.382 ± 0.054
F1 : 0.474 ± 0.048

 Classification binaire who_more_oneB
Accuracy : 0.826 ± 0.015
Precision : 0.768 ± 0.024
Recall : 0.689 ± 0.038
F1 : 0.725 ± 0.016

 Classification binaire who_more_twoB
Accuracy : 0.739 ± 0.022
Precision : 0.549 ± 0.076
Recall : 0.366 ± 0.065
F1 : 0.435 ± 0.058

 Classification binaire who_equal_oneB
Accuracy : 0.740 ± 0.023
Precision : 0.616 ± 0.072
Recall : 0.457 ± 0.082
F1 : 0.522 ± 0.073

 Classification binaire who_equal_twoA
Accuracy : 0.670 ± 0.049
Precision : 0.671 ± 0.089
Recall : 0.632 ± 0.093
F1 : 0.644 ± 0.058

FIGURE 26 – Résultats dimension Effort

FIGURE 27 – Résultats dimension Risque

FIGURE 28 – Résultats dimension Récompense

Les résultats sont globalement **meilleurs que ceux obtenus en régression**, en particulier avec les forêts aléatoires et le gradient boosting. Cela suggère que la tâche de comparaison relative est plus facile à modéliser que la prédiction de valeurs continues.

Cette reformulation du problème en classification binaire a donc permis une **simplification efficace**, conduisant à des prédictions plus stables, plus robustes et mieux adaptées à la nature du problème étudié.

4.4.2 Conclusion pour cette première approche

Les résultats obtenus pour la tâche de régression se sont révélés globalement peu satisfaisants. Cela peut s'expliquer par plusieurs facteurs. Tout d'abord, notre jeu de données est de taille relativement réduite, ce qui limite la capacité des modèles à généraliser. De plus, la tâche de prédiction que nous cherchons à modéliser est particulièrement complexe : il s'agit de capturer des comportements humains dans des scénarios à dimension morale, influencés par des caractéristiques personnelles et des préférences subjectives. De plus, certaines de nos données comprennent des valeurs pas très "cohérentes", c'est-à-dire que par exemple un joueur n'a pas fini de bien comprendre le jeu est saisi des valeurs qui ne sont pas celles qu'il souhaite : une force de 1 pour un homme de grande taille par exemple, ou des valeurs 5 et 5 pour le premier scénario, etc.

À cela s'ajoute la problématique des données incomplètes. Sur les 104 joueurs recensés, seuls 75 ont répondu à l'ensemble des 18 questions prévues, tandis que 29 joueurs ont des sessions partielles. Cette hétérogénéité impacte directement l'apprentissage supervisé.

En revanche, la tâche de classification binaire a montré de meilleurs résultats, avec des F1-scores souvent supérieurs à 0.60, voire atteignant 0.72 pour certaines dimensions comme la récompense. Cela suggère que le modèle parvient à capter des tendances qualitatives sur “qui reçoit plus”, même si l'écart-type reste important — signe d'une variance élevée probablement due au manque de données.

5 L'apprentissage sur des données incomplètes

Le développement du jeu, ainsi que la phase de collecte des données, ont été réalisés en collaboration avec le groupe NFE. Ensemble, nous avons conçu les scénarios, amélioré les formulations et diffusé le jeu afin de recueillir un maximum de réponses provenant de profils variés. Cette collaboration a permis d'établir une base solide pour la suite du projet.

Notre groupe s'est ensuite focalisé sur une problématique spécifique : l'analyse des données collectées, et plus particulièrement l'impact des données incomplètes sur l'entraînement des modèles prédictifs. Cette étape vise à exploiter les sessions de jeu incomplètes, à identifier les profils partiellement renseignés, et à concevoir des modèles capables d'imiter les décisions humaines dans des situations de partage d'effort, de risque ou de récompense.

La question centrale à laquelle nous tentons de répondre est la suivante : **comment entraîner des modèles fiables en présence de données incomplètes, tout en capturant des comportements humains complexes ?**

5.1 Réorganisation du jeu de données

Pour pouvoir travailler correctement avec les modèles d'apprentissage supervisé, il a été nécessaire de réorganiser la structure du jeu de données.

Chaque joueur devait initialement répondre à 18 scénarios. Cependant, dans la pratique, certains joueurs n'ont répondu qu'à une partie de ces scénarios, tandis que d'autres ont joué plusieurs fois et répondu à plus de 18 questions. Pour uniformiser le nombre de lignes par session, nous avons adopté la stratégie suivante :

- Si une session contient **moins de 18 lignes**, nous avons complété les données avec des lignes artificielles contenant des NaN, jusqu'à atteindre 18 lignes.
- Si une session contient **entre 19 et 36 lignes**, nous avons complété jusqu'à 36 lignes (soit 18 + 18).
- Si une session dépasse **36 lignes**, nous avons complété jusqu'à 54 lignes (soit 36 + 18).

L'objectif est de garantir qu'un joueur ayant joué plusieurs fois ait toujours un multiple de 18 lignes, ce qui correspond à un ou plusieurs blocs complets de jeu. Cette structure facilite ensuite les étapes de traitement, d'imputation et d'analyse.

Nous avons ensuite exploré plusieurs approches pour gérer les valeurs manquantes. L'une d'elles consiste à utiliser un KNN Imputer afin d'estimer les valeurs manquantes (les NaN) en s'appuyant sur les sessions les plus similaires. Cela permet d'exploiter les sessions incomplètes sans les écarter, tout en préservant une certaine cohérence statistique dans le comportement simulé.

Les méthodes utilisées pour ce traitement sont détaillées dans les sections suivantes.

5.2 Tâche de régression avec différents modèles et jeux de données

Nous avons testé la tâche de régression sur plusieurs variantes du jeu de données que nous avons construites au cours de l'analyse.

Dans un premier temps, nous avons utilisé uniquement les sessions contenant **exactement 18 réponses**, correspondant aux joueurs ayant complété une session complète du jeu. Ce jeu de données propre permettait d'avoir un cadre stable pour tester les modèles de régression multivariée, comme Random Forest, XGBoost ou Linear Regression.

Par la suite, nous avons construit une seconde version, dans laquelle nous avons décidé de **supprimer les sessions contenant plus de 36 réponses**. L'objectif ici était d'écartier les sessions suspectes, issues de joueurs ayant joué plus de deux fois, souvent avec des réponses moins cohérentes ou aléatoires dans les blocs supplémentaires. Cette version n'intégrait pas encore d'imputation : nous filtrions simplement les données selon les critères de complétude par sessionId, sans introduire de valeurs manquantes.

Les performances observées avec ces deux jeux de données ont été décevantes : les scores de MAE et de R² sont restés faibles, montrant une difficulté notable à capturer la complexité des répartitions réalisées par les joueurs. Ces résultats étaient même moins bons que ceux obtenus dans la tâche initiale commune (voir section précédente).

Ces constats nous ont poussés à adopter une nouvelle stratégie : **conserver l'ensemble des joueurs**, y compris ceux ayant répondu à moins de 18 scénarios, mais en complétant les lignes manquantes par des NaN. L'idée est de tester l'impact de l'imputation via KNN Imputer, en conservant un maximum d'information tout en homogénéisant la structure des sessions. Cette approche permet d'évaluer si l'on peut entraîner un modèle capable de généraliser à partir de données partiellement reconstruites.

5.3 Imputation de données avec deux approches

Une partie centrale de notre problématique concerne la gestion des données incomplètes. En effet, une proportion significative de joueurs n'a pas répondu aux 18 scénarios prévus, ce qui rend difficile l'apprentissage de modèles prédictifs cohérents. Afin de pouvoir exploiter un maximum de sessions tout en conservant une structure homogène (18 réponses par session), nous avons exploré deux approches distinctes d'imputation.

5.3.1 Imputation par KNN

Dans un premier temps, nous avons adopté une stratégie classique d'imputation à l'aide du KNN Imputer. Nous avons identifié les sessions incomplètes (moins de 18 réponses), et avons complété les lignes manquantes avec des NaN, que nous avons ensuite estimés à partir des sessions les plus similaires selon les autres variables du joueur et des personnages.

L'objectif était de préserver le maximum d'information réelle tout en comblant les trous structurels dans les sessions. Cependant, les résultats obtenus se sont révélés **décevants**.

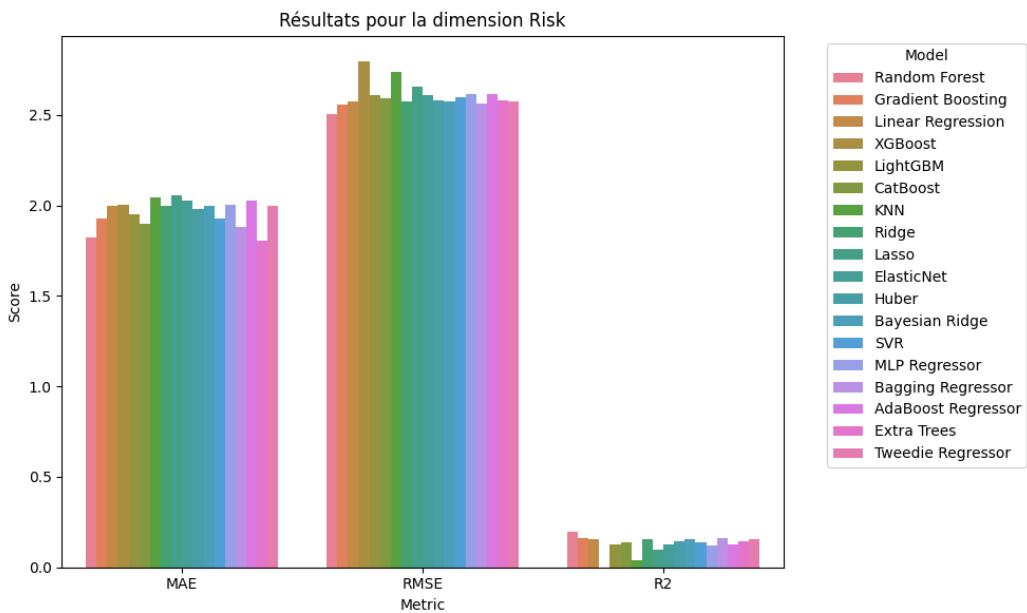


FIGURE 29 – Diagramme de barre Risk avec KNN

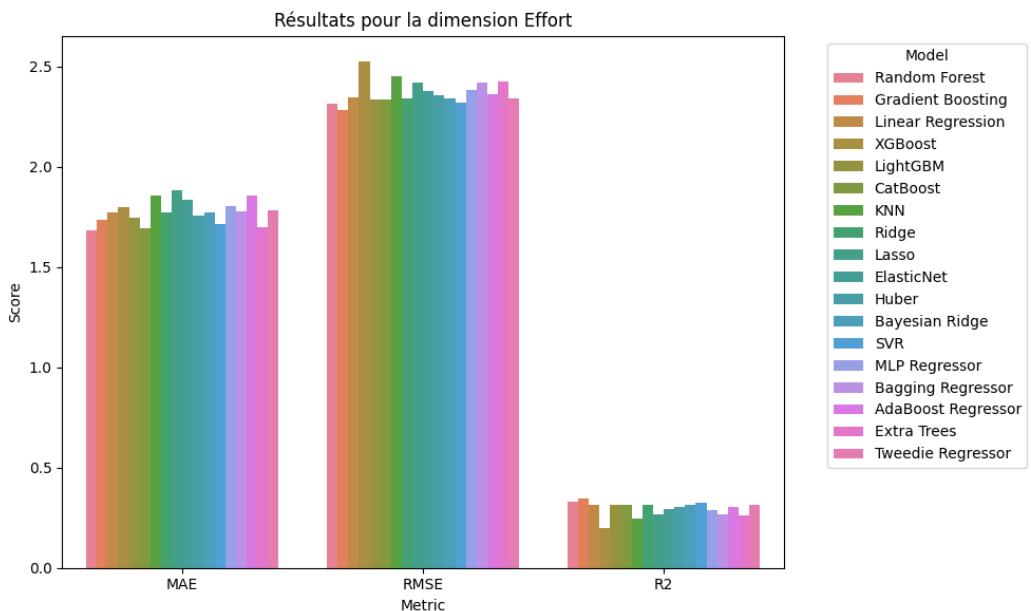


FIGURE 30 – Diagramme de barre Effort avec KNN

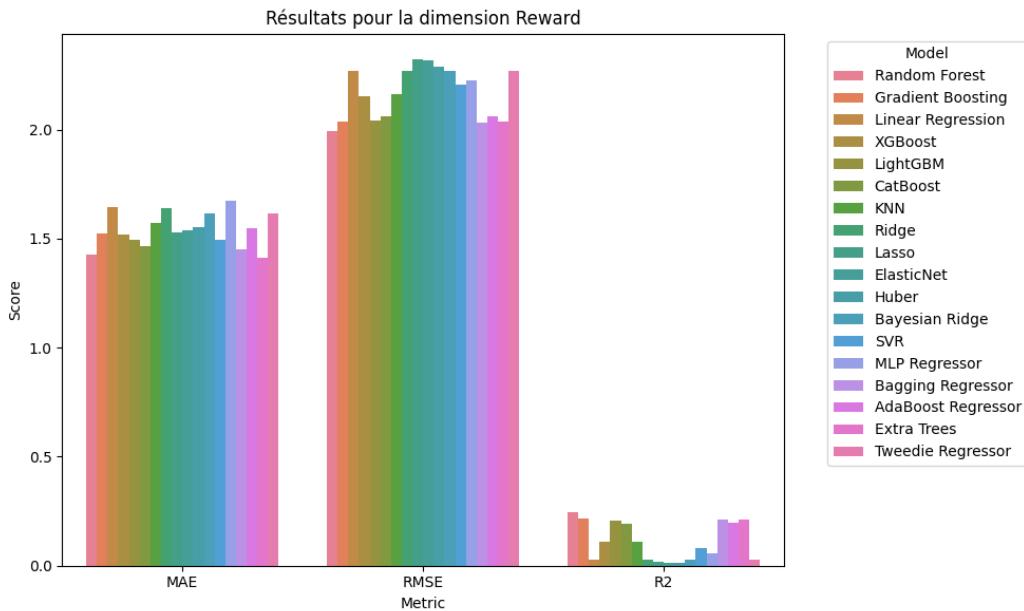


FIGURE 31 – Diagramme de barre Reward avec KNN

Les performances des modèles de régression ont chuté par rapport aux tests réalisés sur les données sans imputation. Cela s’explique notamment par la nature très contextuelle et subjective de la tâche : l’imputation de valeurs numériques dans un comportement humain de répartition (effort, risque, récompense) s’avère artificielle. Les décisions d’un joueur dépendent fortement de ses préférences implicites, de sa logique propre, ou encore de biais culturels — des éléments très difficiles à approximer par simple similarité statistique.

5.3.2 Génération de données synthétiques avec ChatGPT

Face à ces limites, nous avons tenté une approche plus exploratoire : la génération de données manquantes à l’aide d’un modèle de langage, en l’occurrence ChatGPT. L’idée était de simuler des réponses humaines plausibles, en fournissant en entrée le scénario, le profil des personnages (type A et B, avec leur force respective), ainsi que des éléments du profil joueur.

Cette méthode permet de générer des répartitions plus cohérentes d’un point de vue narratif et psychologique, car elle intègre implicitement des raisonnements et des stéréotypes humains. Elle reste bien sûr perfectible, mais ouvre des perspectives intéressantes : la génération assistée peut, dans certains cas, compléter utilement des sessions partiellement renseignées sans dénaturer la logique du jeu.

Pour cela, nous avons fourni à ChatGPT notre fichier CSV contenant les données existantes, et nous lui avons demandé de générer les lignes manquantes selon un ensemble de contraintes garantissant la cohérence interne du jeu de données. Les règles imposées étaient les suivantes :

- Cohérence des forces déjà attribuées** : pour chaque sessionID, les valeurs de forceA et forceB devaient rester constantes si les types de personnages personAType et personBType avaient déjà été rencontrés dans cette même session. Par exemple, si dans une session donnée, le joueur avait attribué une force de 2 au type “Enfant”, alors

toute nouvelle ligne générée dans cette session avec un `personAType` = “Enfant” devait automatiquement reprendre cette valeur de force (2). La même logique s’applique pour `personBType` et `forceB`.

2. **Respect des répartitions entre 0 et 10** : les valeurs de `valueOneA`, `valueOneB`, `valueTwoA` et `valueTwoB` devaient être comprises entre 0 et 10, avec la contrainte que chaque paire devait avoir une somme égale à 10. Par exemple, si `valueOneA` = 2, alors `valueOneB` devait être 8, et inversement. Cela permet de maintenir l’équilibre des répartitions dans les scénarios, conformément à la logique du jeu.
3. **Cohérence du type de scénario** : l’attribution de la variable `associationType` devait être en accord avec les valeurs attribuées. Par exemple, si une ligne présentait des répartitions pour le risque (`valueOne`) et la récompense (`valueTwo`), alors `associationType` devait obligatoirement être “risk-reward”. De même pour “risk-effort” ou “effort-reward”, selon la combinaison des deux dimensions renseignées.

Grâce à cette approche, nous avons pu générer des sessions complètes de 18 lignes pour les joueurs incomplets, sans compromettre la logique ou la structure du jeu. Bien que cette méthode repose sur une génération automatique, elle intègre des éléments narratifs et de bon sens plus proches du raisonnement humain que des interpolations purement statistiques.

Cependant, malgré l’intuition prometteuse de cette méthode, les résultats obtenus avec les données générées par ChatGPT se sont révélés encore moins satisfaisants que ceux obtenus avec l’imputation par KNN. En particulier, les performances des modèles de régression ont diminué, ce qui suggère que la cohérence apparente des données générées ne suffit pas nécessairement à améliorer l’apprentissage.

Voici les résultats obtenus :

Résultats pour la dimension Risk :

| | Model | MAE | RMSE | R2 |
|-----------|--------------------|-------------------|-------------------|--------------------|
| 0 | Random Forest | 2.023 ± 0.080 | 2.720 ± 0.071 | 0.024 ± 0.102 |
| 1 | Gradient Boosting | 2.069 ± 0.069 | 2.674 ± 0.067 | 0.060 ± 0.059 |
| 2 | Linear Regression | 2.081 ± 0.104 | 2.655 ± 0.079 | 0.074 ± 0.043 |
| 3 | XGBoost | 2.189 ± 0.086 | 2.964 ± 0.047 | -0.158 ± 0.100 |
| 4 | LightGBM | 2.096 ± 0.074 | 2.750 ± 0.089 | 0.002 ± 0.109 |
| 5 | CatBoost | 2.070 ± 0.072 | 2.768 ± 0.058 | -0.010 ± 0.089 |
| 6 | KNN | 2.176 ± 0.102 | 2.842 ± 0.132 | -0.067 ± 0.135 |
| 7 | Ridge | 2.081 ± 0.104 | 2.655 ± 0.079 | 0.074 ± 0.043 |
| 8 | Lasso | 2.096 ± 0.122 | 2.698 ± 0.098 | 0.045 ± 0.003 |
| 9 | ElasticNet | 2.078 ± 0.113 | 2.658 ± 0.081 | 0.073 ± 0.019 |
| 10 | Huber | 2.078 ± 0.104 | 2.667 ± 0.085 | 0.065 ± 0.058 |
| 11 | Bayesian Ridge | 2.078 ± 0.101 | 2.649 ± 0.074 | 0.079 ± 0.042 |
| 12 | SVR | 2.026 ± 0.117 | 2.671 ± 0.103 | 0.062 ± 0.072 |
| 13 | MLP Regressor | 2.108 ± 0.066 | 2.723 ± 0.058 | 0.023 ± 0.085 |
| 14 | Bagging Regressor | 2.131 ± 0.074 | 2.867 ± 0.068 | -0.084 ± 0.104 |
| 15 | AdaBoost Regressor | 2.086 ± 0.103 | 2.671 ± 0.073 | 0.064 ± 0.023 |
| 16 | Extra Trees | 2.039 ± 0.084 | 2.843 ± 0.029 | -0.065 ± 0.085 |
| 17 | Tweedie Regressor | 2.078 ± 0.100 | 2.648 ± 0.073 | 0.079 ± 0.042 |

FIGURE 32 – Résultats Risk avec plusieurs modèles

Résultats pour la dimension Effort :

| | Model | MAE | RMSE | R2 |
|-----------|--------------------|-------------------|-------------------|--------------------|
| 0 | Random Forest | 2.256 ± 0.044 | 2.926 ± 0.070 | 0.073 ± 0.059 |
| 1 | Gradient Boosting | 2.205 ± 0.041 | 2.798 ± 0.062 | 0.153 ± 0.043 |
| 2 | Linear Regression | 2.307 ± 0.060 | 2.855 ± 0.051 | 0.118 ± 0.025 |
| 3 | XGBoost | 2.411 ± 0.045 | 3.155 ± 0.074 | -0.078 ± 0.080 |
| 4 | LightGBM | 2.291 ± 0.074 | 2.929 ± 0.080 | 0.072 ± 0.055 |
| 5 | CatBoost | 2.261 ± 0.047 | 2.915 ± 0.072 | 0.080 ± 0.061 |
| 6 | KNN | 2.336 ± 0.060 | 2.987 ± 0.093 | 0.034 ± 0.071 |
| 7 | Ridge | 2.307 ± 0.060 | 2.855 ± 0.051 | 0.118 ± 0.025 |
| 8 | Lasso | 2.397 ± 0.061 | 2.909 ± 0.049 | 0.085 ± 0.009 |
| 9 | ElasticNet | 2.359 ± 0.054 | 2.875 ± 0.043 | 0.106 ± 0.016 |
| 10 | Huber | 2.281 ± 0.060 | 2.878 ± 0.056 | 0.104 ± 0.035 |
| 11 | Bayesian Ridge | 2.310 ± 0.059 | 2.852 ± 0.047 | 0.120 ± 0.024 |
| 12 | SVR | 2.204 ± 0.049 | 2.837 ± 0.055 | 0.129 ± 0.048 |
| 13 | MLP Regressor | 2.255 ± 0.073 | 2.869 ± 0.117 | 0.108 ± 0.086 |
| 14 | Bagging Regressor | 2.338 ± 0.039 | 3.019 ± 0.063 | 0.013 ± 0.049 |
| 15 | AdaBoost Regressor | 2.313 ± 0.053 | 2.834 ± 0.048 | 0.132 ± 0.017 |
| 16 | Extra Trees | 2.327 ± 0.045 | 3.091 ± 0.064 | -0.035 ± 0.060 |
| 17 | Tweedie Regressor | 2.313 ± 0.059 | 2.852 ± 0.046 | 0.120 ± 0.024 |

FIGURE 33 – Résultats Effort avec plusieurs modèles

Résultats pour la dimension Reward :

| | Model | MAE | RMSE | R2 |
|----|--------------------|-------------------|-------------------|--------------------|
| 0 | Random Forest | 1.856 ± 0.073 | 2.509 ± 0.113 | 0.012 ± 0.055 |
| 1 | Gradient Boosting | 1.842 ± 0.108 | 2.427 ± 0.127 | 0.078 ± 0.021 |
| 2 | Linear Regression | 1.862 ± 0.132 | 2.517 ± 0.145 | 0.008 ± 0.017 |
| 3 | XGBoost | 1.989 ± 0.077 | 2.705 ± 0.141 | -0.149 ± 0.082 |
| 4 | LightGBM | 1.894 ± 0.089 | 2.502 ± 0.128 | 0.019 ± 0.034 |
| 5 | CatBoost | 1.904 ± 0.076 | 2.547 ± 0.118 | -0.018 ± 0.045 |
| 6 | KNN | 2.000 ± 0.092 | 2.610 ± 0.120 | -0.068 ± 0.043 |
| 7 | Ridge | 1.862 ± 0.132 | 2.517 ± 0.145 | 0.008 ± 0.017 |
| 8 | Lasso | 1.808 ± 0.149 | 2.530 ± 0.151 | -0.001 ± 0.002 |
| 9 | ElasticNet | 1.805 ± 0.152 | 2.528 ± 0.154 | 0.001 ± 0.004 |
| 10 | Huber | 1.845 ± 0.131 | 2.520 ± 0.145 | 0.006 ± 0.014 |
| 11 | Bayesian Ridge | 1.836 ± 0.135 | 2.513 ± 0.148 | 0.012 ± 0.012 |
| 12 | SVR | 1.813 ± 0.140 | 2.479 ± 0.151 | 0.038 ± 0.019 |
| 13 | MLP Regressor | 1.949 ± 0.112 | 2.538 ± 0.140 | -0.009 ± 0.024 |
| 14 | Bagging Regressor | 1.912 ± 0.112 | 2.584 ± 0.146 | -0.046 ± 0.051 |
| 15 | AdaBoost Regressor | 1.837 ± 0.141 | 2.444 ± 0.153 | 0.066 ± 0.016 |
| 16 | Extra Trees | 1.890 ± 0.100 | 2.634 ± 0.141 | -0.090 ± 0.088 |
| 17 | Tweedie Regressor | 1.849 ± 0.132 | 2.513 ± 0.146 | 0.012 ± 0.014 |

FIGURE 34 – Résultats Reward avec plusieurs modèles

Une explication possible réside dans le phénomène d'**hallucination des modèles de langage**. ChatGPT peut produire des valeurs qui semblent logiques sur le plan syntaxique ou narratif, mais qui ne correspondent pas aux schémas latents réels présents dans les données. Par exemple, il peut générer des répartitions ou associer des types de personnages de manière biaisée ou incohérente à grande échelle, même si cela semble plausible à l’œil humain.

Ainsi, bien que la génération assistée par LLM soit une piste intéressante, son usage dans des tâches de modélisation quantitative reste à manier avec précaution. Elle nécessite un encadrement strict des règles de génération, une vérification manuelle des résultats et, idéalement, des comparaisons rigoureuses avec des modèles entraînés uniquement sur des données observées.

5.4 Un deuxième essai avec la tâche de classification binaire

Après avoir exploré la tâche de régression, nous avons décidé de revisiter la tâche de classification binaire en nous appuyant cette fois sur le **nouveau jeu de données restructuré**, dans lequel les lignes manquantes ont été gérées et de nouvelles variables comme les ratios de force ont été intégrées. Cette version enrichie du dataset nous a permis de tester à nouveau la classification avec une base plus propre et mieux préparée.

Comme vous l'avez dit dans la section précédente, nous avons introduit des cibles dérivées à partir des répartitions numériques entre les deux personnages A et B. Ces cibles, notées `whoMoreOneA` et `whoMoreTwoA` par exemple, permettent de transformer une prédiction de valeurs continues (régression) en une décision binaire : qui a reçu davantage de la première ou de la deuxième dimension du scénario.

- `whoMoreOneA = 1` signifie que `valueOneA > valueOneB` (le personnage A a reçu plus dans la première dimension : risque ou effort).
- `whoMoreTwoA = 1` signifie que `valueTwoA > valueTwoB` (le personnage A a reçu plus dans la deuxième dimension : effort ou récompense).
- Et ainsi de suite

Cette formulation nous permet d'interpréter les résultats sous forme de choix : qui a reçu la plus grande part du risque ? de l'effort ? de la récompense ?

Les résultats se sont **nettement améliorés** par rapport aux premiers essais de classification. Grâce à une meilleure représentation des données (via l'encodage OneHotEncoder et l'ajout de `forceA_ratio` et `forceB_ratio`), les modèles ont pu mieux distinguer les cas où un personnage A ou B recevait une part plus importante de l'effort, du risque ou de la récompense.

Nous avons cette fois évalué plusieurs modèles de classification, tels que : **Random Forest Classifier**, **Logistic Regression**, **Support Vector Classifier (SVC)** ou **Gradient Boosting Classifier**, etc.

Voici quelques exemples concrets extraits de nos tests :

| Résultats pour la dimension : Risk | | | | | |
|--|--|---------------|---------------|---------------|---------------|
| | Classification binaire : who_more_oneA | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.734 ± 0.025 | 0.633 ± 0.048 | 0.487 ± 0.040 | 0.549 ± 0.036 |
| 1 | Logistic Regression | 0.677 ± 0.027 | 0.513 ± 0.024 | 0.659 ± 0.054 | 0.575 ± 0.023 |
| 2 | SVM | 0.729 ± 0.009 | 0.660 ± 0.066 | 0.376 ± 0.042 | 0.478 ± 0.048 |
| 3 | SVM Poly | 0.710 ± 0.019 | 0.698 ± 0.150 | 0.237 ± 0.034 | 0.352 ± 0.054 |
| 4 | SVM RBF | 0.729 ± 0.009 | 0.660 ± 0.066 | 0.376 ± 0.042 | 0.478 ± 0.048 |
| 5 | SVM Sigmoid | 0.650 ± 0.063 | 0.474 ± 0.082 | 0.403 ± 0.112 | 0.430 ± 0.090 |
| 6 | KNN | 0.708 ± 0.020 | 0.575 ± 0.051 | 0.481 ± 0.022 | 0.523 ± 0.032 |
| 7 | Gradient Boosting | 0.738 ± 0.009 | 0.628 ± 0.027 | 0.511 ± 0.049 | 0.562 ± 0.033 |
| 8 | AdaBoost | 0.711 ± 0.025 | 0.611 ± 0.073 | 0.367 ± 0.042 | 0.458 ± 0.051 |
| 9 | Naive Bayes | 0.681 ± 0.027 | 0.569 ± 0.085 | 0.154 ± 0.049 | 0.241 ± 0.067 |
| Meilleur modèle (Logistic Regression) pour Risk - who_more_oneA sauvegardé dans : best_model_Risk_who_more_oneA_Logistic_Regression.joblib | | | | | |
| | Classification binaire : who_more_twoA | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.812 ± 0.030 | 0.715 ± 0.062 | 0.423 ± 0.073 | 0.530 ± 0.072 |
| 1 | Logistic Regression | 0.685 ± 0.025 | 0.424 ± 0.057 | 0.708 ± 0.072 | 0.530 ± 0.063 |
| 2 | SVM | 0.751 ± 0.031 | 0.160 ± 0.320 | 0.019 ± 0.038 | 0.034 ± 0.068 |
| 3 | SVM Poly | 0.747 ± 0.028 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000 |
| 4 | SVM RBF | 0.751 ± 0.031 | 0.160 ± 0.320 | 0.019 ± 0.038 | 0.034 ± 0.068 |
| 5 | SVM Sigmoid | 0.650 ± 0.038 | 0.194 ± 0.109 | 0.170 ± 0.097 | 0.180 ± 0.100 |
| 6 | KNN | 0.752 ± 0.015 | 0.515 ± 0.053 | 0.370 ± 0.041 | 0.427 ± 0.025 |
| 7 | Gradient Boosting | 0.777 ± 0.044 | 0.592 ± 0.095 | 0.378 ± 0.081 | 0.460 ± 0.088 |
| 8 | AdaBoost | 0.749 ± 0.028 | 0.516 ± 0.089 | 0.238 ± 0.031 | 0.324 ± 0.041 |
| 9 | Naive Bayes | 0.535 ± 0.039 | 0.335 ± 0.012 | 0.857 ± 0.042 | 0.481 ± 0.017 |
| Meilleur modèle (Random Forest) pour Risk - who_more_twoA sauvegardé dans : best_model_Risk_who_more_twoA_Random_Forest.joblib | | | | | |

FIGURE 35 – Résultats Risk pour les cibles `whoMoreOneA` et `whoMoreTwoA`

| Résultats pour la dimension : Effort | | | | | |
|--|---------------------|---------------|---------------|---------------|---------------|
| Classification binaire : who_more_oneA | | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.767 ± 0.027 | 0.717 ± 0.037 | 0.546 ± 0.068 | 0.619 ± 0.054 |
| 1 | Logistic Regression | 0.716 ± 0.035 | 0.577 ± 0.065 | 0.679 ± 0.056 | 0.624 ± 0.061 |
| 2 | SVM | 0.753 ± 0.017 | 0.712 ± 0.050 | 0.494 ± 0.031 | 0.582 ± 0.029 |
| 3 | SVM Poly | 0.729 ± 0.028 | 0.712 ± 0.015 | 0.373 ± 0.048 | 0.488 ± 0.045 |
| 4 | SVM RBF | 0.753 ± 0.017 | 0.712 ± 0.050 | 0.494 ± 0.031 | 0.582 ± 0.029 |
| 5 | SVM Sigmoid | 0.634 ± 0.028 | 0.476 ± 0.045 | 0.506 ± 0.048 | 0.490 ± 0.042 |
| 6 | KNN | 0.741 ± 0.019 | 0.647 ± 0.041 | 0.561 ± 0.030 | 0.601 ± 0.033 |
| 7 | Gradient Boosting | 0.765 ± 0.031 | 0.709 ± 0.064 | 0.560 ± 0.053 | 0.624 ± 0.047 |
| 8 | AdaBoost | 0.746 ± 0.028 | 0.680 ± 0.067 | 0.501 ± 0.063 | 0.577 ± 0.065 |
| 9 | Naive Bayes | 0.658 ± 0.091 | 0.556 ± 0.094 | 0.511 ± 0.198 | 0.497 ± 0.105 |

| Meilleur modèle (Logistic Regression) pour Effort - who_more_oneA sauvegardé dans : best_model_Effort_who_more_oneA_Logistic_Regression.joblib | | | | | |
|--|---------------------|---------------|---------------|---------------|---------------|
| Classification binaire : who_more_twoA | | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.775 ± 0.021 | 0.639 ± 0.035 | 0.502 ± 0.057 | 0.560 ± 0.041 |
| 1 | Logistic Regression | 0.678 ± 0.031 | 0.461 ± 0.039 | 0.561 ± 0.040 | 0.541 ± 0.028 |
| 2 | SVM | 0.741 ± 0.026 | 0.608 ± 0.093 | 0.280 ± 0.084 | 0.377 ± 0.086 |
| 3 | SVM Poly | 0.714 ± 0.010 | 0.502 ± 0.318 | 0.056 ± 0.039 | 0.057 ± 0.066 |
| 4 | SVM RBF | 0.741 ± 0.026 | 0.608 ± 0.093 | 0.280 ± 0.084 | 0.377 ± 0.086 |
| 5 | SVM Sigmoid | 0.574 ± 0.037 | 0.213 ± 0.066 | 0.178 ± 0.054 | 0.194 ± 0.059 |
| 6 | KNN | 0.741 ± 0.028 | 0.561 ± 0.073 | 0.464 ± 0.061 | 0.506 ± 0.056 |
| 7 | Gradient Boosting | 0.753 ± 0.027 | 0.597 ± 0.058 | 0.420 ± 0.085 | 0.491 ± 0.073 |
| 8 | AdaBoost | 0.754 ± 0.025 | 0.609 ± 0.062 | 0.402 ± 0.089 | 0.409 ± 0.072 |
| 9 | Naive Bayes | 0.444 ± 0.061 | 0.332 ± 0.029 | 0.904 ± 0.045 | 0.484 ± 0.027 |

| |
|--|
| Meilleur modèle (Random Forest) pour Effort - who_more_twoA sauvegardé dans : best_model_Effort_who_more_twoA_Random_Forest.joblib |
|--|

FIGURE 36 – Résultats Effort pour les cibles whoMoreOneA et whoMoreTwoA

| Résultats pour la dimension : Reward | | | | | |
|--|---------------------|---------------|---------------|---------------|---------------|
| Classification binaire : who_more_oneA | | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.805 ± 0.017 | 0.757 ± 0.036 | 0.655 ± 0.038 | 0.701 ± 0.029 |
| 1 | Logistic Regression | 0.784 ± 0.035 | 0.669 ± 0.040 | 0.778 ± 0.030 | 0.719 ± 0.026 |
| 2 | SVM | 0.782 ± 0.025 | 0.728 ± 0.034 | 0.607 ± 0.046 | 0.661 ± 0.038 |
| 3 | SVM Poly | 0.776 ± 0.031 | 0.729 ± 0.043 | 0.578 ± 0.054 | 0.644 ± 0.043 |
| 4 | SVM RBF | 0.782 ± 0.028 | 0.728 ± 0.034 | 0.607 ± 0.046 | 0.661 ± 0.038 |
| 5 | SVM Sigmoid | 0.642 ± 0.059 | 0.490 ± 0.069 | 0.476 ± 0.103 | 0.480 ± 0.081 |
| 6 | KNN | 0.779 ± 0.027 | 0.693 ± 0.035 | 0.668 ± 0.036 | 0.680 ± 0.031 |
| 7 | Gradient Boosting | 0.817 ± 0.022 | 0.759 ± 0.032 | 0.707 ± 0.027 | 0.732 ± 0.023 |
| 8 | AdaBoost | 0.787 ± 0.030 | 0.724 ± 0.034 | 0.631 ± 0.074 | 0.673 ± 0.055 |
| 9 | Naive Bayes | 0.747 ± 0.045 | 0.635 ± 0.030 | 0.643 ± 0.169 | 0.628 ± 0.112 |

| Meilleur modèle (Gradient Boosting) pour Reward - who_more_oneA sauvegardé dans : best_model_Reward_who_more_oneA_Gradient_Boosting.joblib | | | | | |
|--|---------------------|---------------|---------------|---------------|---------------|
| Classification binaire : who_more_twoA | | | | | |
| | Model | accuracy | precision | recall | f1 |
| 0 | Random Forest | 0.793 ± 0.025 | 0.608 ± 0.031 | 0.429 ± 0.096 | 0.496 ± 0.067 |
| 1 | Logistic Regression | 0.704 ± 0.017 | 0.431 ± 0.047 | 0.701 ± 0.073 | 0.533 ± 0.050 |
| 2 | SVM | 0.757 ± 0.028 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000 |
| 3 | SVM Poly | 0.757 ± 0.028 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000 |
| 4 | SVM RBF | 0.757 ± 0.028 | 0.000 ± 0.000 | 0.000 ± 0.000 | 0.000 ± 0.000 |
| 5 | SVM Sigmoid | 0.662 ± 0.045 | 0.211 ± 0.053 | 0.140 ± 0.046 | 0.165 ± 0.041 |
| 6 | KNN | 0.752 ± 0.021 | 0.496 ± 0.104 | 0.318 ± 0.070 | 0.381 ± 0.074 |
| 7 | Gradient Boosting | 0.796 ± 0.018 | 0.638 ± 0.097 | 0.367 ± 0.056 | 0.464 ± 0.062 |
| 8 | AdaBoost | 0.755 ± 0.028 | 0.483 ± 0.185 | 0.148 ± 0.068 | 0.223 ± 0.095 |
| 9 | Naive Bayes | 0.295 ± 0.054 | 0.254 ± 0.034 | 0.972 ± 0.029 | 0.401 ± 0.040 |

FIGURE 37 – Résultats Reward pour les cibles whoMoreOneA et whoMoreTwoA

Les métriques mesurées (accuracy, precision, recall, F1-score) ont toutes montré une amélioration significative, en particulier pour les scénarios de type **effort - reward** où le modèle a pu capturer des tendances claires dans les décisions des joueurs.

Cependant, pour le test de Turing prévu dans le cadre du projet, nous avons choisi de **conserver la version basée sur la régression**. En effet, bien que la classification binaire présente des performances intéressantes, la régression permet une prédiction plus fine et continue des répartitions, ce qui la rend plus adaptée à notre objectif principal : simuler les nuances du comportement humain face à des dilemmes sociaux.

Cette décision reflète notre volonté de privilégier la précision des réponses sur la simplicité de leur évaluation.

6 Test de Turing pour tester le modèle obtenu

Afin d'évaluer les performances de notre modèle de prédiction, nous avons mené une forme simplifiée de **Test de Turing**.

Le Test de Turing, proposé par le mathématicien **Alan Turing en 1950**, est une méthode classique pour évaluer l'intelligence d'une machine. Le principe est simple : si un humain interagissant avec une machine ne peut pas distinguer si les réponses proviennent d'un humain ou d'un algorithme, alors la machine est considérée comme capable d'imiter l'intelligence humaine.

Dans notre projet, nous avons adapté ce test de manière simplifiée pour évaluer notre modèle de prédiction. L'idée était de comparer les réponses générées par notre modèle à celles données par de vrais participants, et de mesurer à quel point elles se rapprochaient.

6.1 Recherche des profils

Pour cela, nous avons sollicité environ une trentaine de personnes issues de notre entourage (famille, amis, passants dans la rue), n'ayant jamais joué au jeu auparavant, afin d'obtenir des réponses "neutres" et non biaisées par une expérience antérieure.

Nous avons cherché à sélectionner des profils aussi proches que possible de ceux présents dans notre base de données, en termes d'âge, de genre, de niveau d'études ou encore de nationalité. Cela nous permettait de comparer plus rigoureusement les réponses de ces nouveaux participants à celles générées par notre modèle.

Chaque personne a joué au jeu sans connaître les objectifs de notre test. Ensuite, pour chaque participant, nous avons générée les prédictions du modèle correspondant à son profil (ou similaire), et comparé les deux ensembles de réponses.

6.2 Résultats obtenus

Les résultats ont montré que le modèle n'était pas toujours précis dans ses prédictions : les réponses générées ne correspondaient que partiellement à celles des participants humains. Cela ne nous a pas véritablement surpris, étant donné la complexité de la tâche. En effet, prévoir les décisions d'un individu face à un dilemme moral ou à une situation sociale implique de nombreux facteurs contextuels, personnels et culturels que notre modèle, basé sur un ensemble limité de données, ne peut entièrement capturer.

Néanmoins, certains résultats se sont révélés encourageants. À titre d'illustration, nous avons soumis au modèle deux exemples concrets de situations :

- **Exemple 1** : scénario de type *risk-reward*, avec une femme de 23 ans (niveau bac+4, nationalité DZA), confrontée à une situation où les personnages sont une "Femme grande taille" (force = 7) et une "Personne âgée" (force = 3).
- **Exemple 2** : scénario de type *risk-effort*, avec une femme de 22 ans (niveau bac+3, nationalité FRA), dans lequel les personnages sont un "Homme grande taille" (force = 8) et une "Personne âgée" (force = 3).

Le modèle a produit les prédictions suivantes :

Exemple 1 (risk-reward) : [5.64, 4.35]
Exemple 2 (risk-effort) : [6.12, 3.87]

Bien que ces prédictions ne correspondent pas exactement aux décisions que pourrait prendre un être humain, elles restent **plausibles et raisonnablement équilibrées**. Elles témoignent d'une certaine capacité du modèle à intégrer les rapports de force et le type de scénario pour produire une répartition logique.

Ces résultats, même imparfaits, montrent que le modèle est capable d'approximer un raisonnement de bon sens dans un contexte donné. Toutefois, une amélioration significative nécessiterait probablement davantage de données, ainsi que l'introduction de variables plus fines sur les préférences des joueurs ou le contexte émotionnel des choix.

Cette expérience a été très positive et enrichissante. Beaucoup de participants ont exprimé un intérêt particulier pour le jeu, notamment après qu'on leur ait expliqué les normes sociales utilisées dans le jeu (risque, effort, récompense) qui influencent les choix dans les scénarios. Plusieurs ont souligné qu'ils ne s'étaient jamais interrogés sur l'influence de leur culture, de leur éducation ou de leur âge sur ce type de décisions, et ils ont apprécié de réfléchir à celles-ci.

Cette prise de conscience constitue déjà, en soi, une réussite pédagogique du projet, au-delà des performances purement quantitatives du modèle. Ainsi, même si notre modèle reste perfectible, cette phase de test a confirmé la pertinence sociale du jeu.

7 Lien avec notre sujet original

Notre sujet original avait pour ambition finale d'intégrer le modèle développé dans un robot humanoïde. Cette intégration aurait permis de maximiser le réalisme de la décision prise par le modèle. Cependant, cette ambition reposait sur l'idée que nous aurions un modèle de prédiction plus que satisfaisant, ce qui n'est pas le cas aux vus des résultats obtenus.

De plus, le sujet initial implique une amélioration continue du modèle. En effet, une étape logique de l'évolution du projet consisterait à intégrer de l'apprentissage par renforcement. Toutefois, nous avons estimé que le modèle n'était pas encore suffisamment performant pour cette phase. Par ailleurs, cette approche nécessiterait un plus grand nombre de participants. Il serait donc nécessaire de lancer une campagne de collecte de données plus conséquente en amont.

8 Conclusion

En résumé, ce projet avait pour ambition de modéliser la prise de décision humaine lors de négociations impliquant le risque, la récompense et l'effort. Nous avons d'abord conçu un jeu sérieux pour collecter des données authentiques sur les choix des participants, en veillant à la diversité et à la neutralité des scénarios. La collecte de données a constitué une étape essentielle, mais aussi un défi majeur, car la quantité limitée d'exemples a fortement conditionné la suite du travail.

L'étape qui a suivi a consisté à développer un modèle prédictif à une seule dimension, permettant d'analyser séparément la manière dont les participants répartissent les ressources selon chacun des trois axes principaux : risque, récompense et effort. Cette phase, menée en collaboration avec le groupe NFE, a permis de poser les bases solides du projet et de collecter un ensemble de données variées, essentielles pour garantir la robustesse des analyses ultérieures.

Par la suite, notre groupe s'est concentré sur l'élaboration d'un modèle à une dimension, capable de prédire simultanément les choix des joueurs selon un critères, comme le risque, la récompense et l'effort, en prenant en compte l'absence de réponses des joueurs.

Cependant, le travail autour du machine learning s'est avéré particulièrement difficile, principalement en raison de la faible quantité et de la nature limitée des données collectées. Malgré nos efforts pour diversifier les profils des participants et concevoir des scénarios variés, la taille réduite de notre jeu de données a fortement limité la capacité du modèle à généraliser et à produire des résultats probants. Les performances obtenues restent donc modestes, mais elles mettent en lumière la difficulté de travailler sur des modèles prédictifs dans un contexte où les données sont rares et parfois peu informatives. Cette expérience souligne l'importance cruciale de la collecte de données massives et de qualité dans le domaine du machine learning appliqué aux sciences sociales.

L'objectif ultime de ce projet était de contribuer à la compréhension des mécanismes sous-jacents aux négociations humaines, en mettant l'accent sur les trois dimensions centrales : risque, récompense et effort. En neurosciences, ces dimensions sont au cœur de l'analyse des comportements sociaux et des processus de décision, car elles déterminent la façon dont les individus arbitrent entre différentes options et interagissent avec leur environnement.

9 Lien du Github et de l'application

- **Lien de l'application web :**
<https://choices-eta.vercel.app>
- **Lien du dépôt GitHub pour le jeu :**
https://github.com/NinaShine/TER_RiskReward
- **Lien du dépôt GitHub pour la partie Machine Learning :**
https://github.com/NinaShine/ML_risk_reward