



Trinity College Dublin  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

Parsons Building, Trinity College, Dublin 2, DUBLIN

RAPPORT D'ÉLÈVE INGÉNIEUR  
STAGE DE 2<sup>ÈME</sup> ANNÉE  
FILIÈRE : MODÉLISATION MATHÉMATIQUE ET SCIENCE DES DONNÉES

---

## Comparative Triple Judgement

---

*Présenté par :*  
Romain PERRIER

*Réferent académique :* Jonas KOKO  
*Tutrice ISIMA :* Susan ARBON-LEAHY  
*Tuteur entreprise :* Kevin KELLY

*Date de la Soutenance :*  
9 Mars 2024  
*Durée du stage :* 4 mois

ISIMA, Campus universitaire des Cézeaux, 1 rue de la Chebarde, CS 60026,  
63178 Aubière CEDEX

# Remerciements

J'aimerais remercier mon tuteur d'entreprise, Kevin KELLY, qui m'a permis d'effectuer un stage au Trinity College à Dublin. De surcroît, son accompagnement et ses conseils tout au long du projet m'ont été utiles pour comprendre les concepts encadrant son modèle. Je le remercie également de m'avoir permis de découvrir le domaine mathématique de la théorie de l'information.

Je tiens également à exprimer ma gratitude à Susan ARBON-LEAHY, professeur d'anglais à l'ISIMA et ma tutrice, dont les cours de l'année, sa disponibilité tout au long du stage, ainsi que la mise en contact avec le Trinity College, ont été essentiels pour l'obtention de ce stage.

Je souhaite également remercier Anneliese WALSH et Patrick LYNCH, qui ont été présents avec Kevin KELLY lors des entrevues bihebdomadaires pour discuter des avancées des projets effectués au sein du RAIL.

J'aimerais adresser des remerciements spéciaux à Lucas PICHON ainsi qu'à Beatrice WHARTON, qui m'ont permis de tester la portabilité de ma bibliothèque sur d'autres systèmes d'exploitation.

Je ne saurais oublier de mentionner mes camarades et collègues de l'ISIMA, ainsi que les doctorants du Trinity College et les stagiaires irlandais, avec lesquels j'ai pu échanger sur nos projets respectifs. Leur collaboration et leur soutien ont créé une ambiance propice au travail et à la productivité.

Merci à tous.

# Table des matières

<b>Introduction</b>	<b>8</b>
<b>1 Présentation du stage</b>	<b>9</b>
1.1 Présentation du TCD . . . . .	9
1.1.1 Organisation . . . . .	9
1.1.2 Laboratoire de Robotique et d'Innovation (RAIL) . . . . .	11
1.2 Objectif du stage . . . . .	12
1.2.1 Méthode absolue . . . . .	12
1.2.2 Méthodes comparatives . . . . .	13
<b>2 Travail effectué</b>	<b>16</b>
2.1 Organisation . . . . .	16
2.1.1 Méthode semi-agile . . . . .	16
2.1.2 Planning du stage . . . . .	17
2.2 Etude du Problème . . . . .	18
2.2.1 Sélection des éléments . . . . .	18
2.2.2 Introduction du biais . . . . .	24
2.3 Outils . . . . .	27
2.3.1 Bibliothèques utilisées . . . . .	27
2.3.2 Création de Bibliothèque . . . . .	28
<b>3 Résultat</b>	<b>31</b>
3.1 Evaluation automatique . . . . .	31
3.2 Evaluation manuelle . . . . .	34
3.2.1 Test des modèles . . . . .	37
3.2.2 Conditions réelles . . . . .	37
3.3 Première analyse des modèles . . . . .	37
3.3.1 Nombre d'erreurs . . . . .	37
3.3.2 Seuil de perception . . . . .	39
3.4 A faire . . . . .	40
<b>Conclusion</b>	<b>41</b>

# Table des figures

1.1	Organigramme du TCD . . . . .	10
1.2	Organigramme du RAIL . . . . .	11
1.3	Schéma fonctionnel de la bibliothèque . . . . .	12
1.4	Jugement absolu . . . . .	13
1.5	Itération de l'ACJ . . . . .	13
1.6	Fonctionnement de l'ACJ . . . . .	14
1.7	Itération du CTJ . . . . .	15
1.8	Fonctionnement du CTJ . . . . .	15
2.1	Exemple du Trello . . . . .	16
2.2	Gantt prévisionnel . . . . .	17
2.3	Gantt réel . . . . .	17
2.4	Information au sens de Politt . . . . .	19
2.5	Sélection roulette . . . . .	20
2.6	Entropie de Shannon (quantité d'informations) . . . . .	21
2.7	Sélection roulette pour ACJ . . . . .	22
2.8	Sélection roulette pour CTJ . . . . .	23
2.9	Fonction sigmoïde pour $\lambda = -\frac{\log(\frac{1}{9})}{50}$ . . . . .	24
2.10	Biais pour l'ACJ . . . . .	25
2.11	Arbre de biais . . . . .	25
2.12	Erreur d'échelles pour le CTJ . . . . .	26
2.13	Biais pour le jugement absolu . . . . .	26
3.1	Interface de l'ACJ pour la visualisation d'images . . . . .	35
3.2	Interface du CTJ pour la visualisation d'images . . . . .	35
3.3	Interface du jugement absolu pour la visualisation d'images . . . . .	36
3.4	Interface du CTJ pour la visualisation d'un pdf . . . . .	36
3.5	Nombre d'itérations des modèles en fonction du nombre d'erreurs . . . . .	38
3.6	Précision des modèles en fonction du nombre d'erreurs . . . . .	38
3.7	Nombre d'itérations des modèles en fonction du seuil de perception d'un juge . . . . .	39
3.8	Précision des modèles en fonction du seuil de perception d'un juge . . . . .	40

## Résumé

Lors de l'évaluation ou du jugement d'un élément, il est fréquent de lui attribuer une valeur. Cependant, évaluer des objets liés à des concepts abstraits peut rendre cette attribution de valeur numérique difficile. Pour pallier ce problème, les **jugements comparatifs** offrent une alternative en se concentrant non pas sur la valeur intrinsèque des éléments, mais sur leur comparaison entre eux. Cette méthode permet d'obtenir une vision d'ensemble plus précise et d'effectuer des tests plus simples et plus justes.

Pour comparer les différentes méthodes d'évaluation, j'ai développé une bibliothèque **Python** implémentant une méthode de jugement absolu et deux méthodes de jugement comparatif : le jugement adaptatif comparatif (**ACJ**) et le triple jugement comparatif (**CTJ**). Ces méthodes optimisent la sélection des éléments à comparer en utilisant la **théorie de l'information**, afin d'améliorer la pertinence des comparaisons effectuées par l'utilisateur.

**Mots-clés :** jugement comparatif, **ACJ**, **CTJ**, théorie de l'information, **Python**

## Abstract

When evaluating or judging an item, it is common to assign it a value. However, evaluating objects related to abstract concepts can make this numerical value assignment challenging. To address this issue, **comparative judgments** offer an alternative by focusing not on the intrinsic value of the items, but on their comparison with each other. This method allows for a more accurate overall view and facilitates simpler and fairer tests.

To compare different evaluation methods, I developed a **Python** package implementing one direct judgment method and two comparative judgment methods: adaptive comparative judgment (**ACJ**) and triple comparative judgment (**CTJ**). These methods optimize the selection of items to compare by using **information theory**, enhancing the relevance of the comparisons made by the user.

**Keywords:** **comparative judgment**, **ACJ**, **CTJ**, **information theory**, **Python**

# Lexique

**ACJ** Jugement adaptatif comparatif ou Adaptive Comparative Judgement en anglais. C'est un jugement comparatif entre deux éléments où les utilisateurs doivent sélectionner le meilleur élément. Il sélectionne les éléments à présenter aux utilisateurs en fonction de l'information qu'ils peuvent apporter. 4, 5, 12–14, 18, 19, 21, 22, 24, 25, 28, 29, 32, 34, 35, 37–41

**application web** Du code qui tourne sur un serveur et qui est accessible via une URL. . 12, 17.

**bibliothèque** Un regroupement de code source donnant accès aux utilisateurs à des modules. 8, 12, 17, 24, 27–31, 34, 40, 41

**CTJ** Triple jugement comparatif ou Comparative Triple Judgement, c'est un jugement comparatif pour trois éléments. Il faut les trier puis sélectionner la distance de l'élément du milieu aux autres. Le modèle a été pensé par le Dr KELLY. 4, 8, 12, 14, 15, 17–19, 22, 23, 25, 26, 29, 33, 35–41

**entropie de Shannon** Correspond à la quantité d'informations qu'un élément peut apporter. 4, 20–22, 32

**entropie jointe** Sous-entendu entropie jointe de Shannon, elle correspond à la quantité d'information d'un couple d'éléments. 21, 22

**information d'interaction** Correspond à l'information apportée par un n-uplet, ici un trio. 22

**information mutuelle** Correspond à la dépendance au sens probabilistique des deux éléments d'un couple. 21

**jugement absolu** Le fait de noter une suite d'éléments en tenant compte uniquement de critères prédéfinis, sans prendre en compte les éléments dans leur ensemble. 4, 8, 12, 13, 17, 18, 26, 41 . 12.

**jugement comparatif** Le fait de comparer les éléments entre eux pour établir un ordre et attribuer des valeurs aux éléments en fonction de cet ordre et d'un algorithme déterminé. 8, 12, 13 . 12, 13.

**licence** Document posant des contraintes pour l'utilisation d'un projet ou de tout autre bien, matériel ou non. 27, 29, 40, 41

**n-uplet** Un tuple de n éléments. 13, 19, 22 Un couple est un n-uplet de deux éléments. 13, 19, 21, 22, 26, 32. Un trio est un n-uplet de trois éléments. 14, 15, 22, 23, 26, 35.

**open source** Le fait de développer un projet qui sera libre d'accès, partageable, utilisable et améliorable librement par les utilisateurs sous certaines conditions. [8](#), [27](#), [40](#), [41](#)

**PIP** Un site permettant de publier et tester des bibliothèques. [29](#), [30](#)

**Python** Un langage de programmation orienté objet, l'un des plus utilisés. [8](#), [12](#), [27–29](#), [41](#)

**RAIL** Le Laboratoire de Robotique et d'Innovation, où j'ai effectué mon stage. [4](#), [11](#)

**sélection roulette** Une méthode de sélection d'éléments permettant de garder une part d'aléatoire tout en tenant compte de l'impact d'une ou plusieurs caractéristiques. [4](#), [19–23](#), [27](#)

**TCD** Trinity College Dublin, l'université principale de Dublin, où j'ai effectué mon stage. [4](#), [8–10](#)

# Introduction

Ce stage m'a été confié par le Dr. KELLY, ingénieur en génie mécanique, manufacturier et biomédical, ainsi que professeur associé au Trinity College de Dublin ([TCD](#)).

Les méthodes d'évaluation ont pour objectif, comme leur nom l'indique, d'évaluer un ensemble d'éléments afin de leur associer une valeur. Généralement, cette valeur est comprise dans un intervalle préalablement connu par l'évaluateur. Il existe deux grandes catégories de méthodes d'évaluation : le [jugement absolu](#), qui consiste à évaluer un élément à partir de critères spécifiques et à lui attribuer une note indépendante des autres éléments, et le [jugement comparatif](#), dans lequel l'évaluateur compare les éléments de l'ensemble entre eux, puis un algorithme détermine la note de chaque élément à partir des jugements de l'évaluateur.

Mon stage a eu pour but de développer une [bibliothèque open source](#) en Python facilitant la recherche sur le triple jugement comparatif ([CTJ](#)), un modèle créé par mon maître de stage, le Dr. KELLY.

Pour ce faire, la [bibliothèque](#) doit être portable afin de faciliter son exécution sous différents systèmes d'exploitation. Elle doit implémenter plusieurs méthodes d'évaluation afin de pouvoir les comparer au [CTJ](#). De plus, chacune de ces méthodes doit pouvoir être utilisée dans plusieurs contextes : le premier étant un jugement automatique sans intervention extérieure, avec la possibilité d'introduire des biais ; le second étant un jugement effectué par l'utilisateur pour vérifier les résultats des modèles en connaissance des valeurs des éléments ; et le dernier correspondant à un test en conditions réelles, c'est-à-dire sans connaissance préalable des éléments à évaluer. Une interface graphique doit donc être intégrée pour les deux derniers cas.

Il a donc été nécessaire de réfléchir à la structure de la [bibliothèque](#) et aux fonctions accessibles à l'utilisateur. De plus, le [CTJ](#) étant un modèle récent, il a également fallu réfléchir à la manière dont les éléments à comparer sont sélectionnés afin d'optimiser l'information fournie au modèle par l'évaluation.

Pour présenter ce projet, j'aborderai en premier lieu le stage et son organisation. Par la suite, je présenterai les méthodes d'évaluation implémentées, ainsi que les [bibliothèques](#) utilisées pour l'implémentation. Enfin, je présenterai la [bibliothèque](#) développée, son fonctionnement détaillé ainsi que les premiers résultats d'analyse sur un jeu de données basiques.

# 1 Présentation du stage

## 1.1 Présentation du TCD

Le Trinity College Dublin (**TCD**) est la plus ancienne université d'Irlande ; en effet, elle a été fondée en 1592 par la reine d'Angleterre, ELIZABETH I. Son statut historique spécifique en fait un lieu où étudiants, doctorants et professeurs côtoient touristes et passionnés d'histoire. Ce contraste est d'autant plus présent que le **TCD** comprend en son sein une attraction touristique majeure de Dublin, le "Book of Kells Experience". C'est un musée permettant l'accès à l'ancienne bibliothèque du **TCD**, qui contient plusieurs milliers d'anciens manuscrits.

Le **TCD** est composé de trois facultés distinctes. La faculté des Arts, Humanités et Sciences Sociales regroupe entre autres des cours de littérature, de cinéma, de droit, de philosophie et de psychologie ; c'est le pôle littéraire du **TCD**. Le second pôle est le pôle scientifique, la faculté d'ingénierie, de mathématiques et de sciences. Les principaux cours enseignés sont les mathématiques, les sciences de l'ingénieur, les sciences naturelles et la physique. Enfin, la dernière faculté est celle de la santé, comprenant notamment des cours de pharmacie, de médecine et de dentisterie. En plus de ces trois facultés, une école de commerce interne au **TCD** a été ouverte en 2019.

Cet établissement est également renommé pour avoir formé de grands esprits littéraires et scientifiques tel que, William Rowan Hamilton. Il est surtout connu pour avoir introduit les quaternions, le premier exemple de nombres hypercomplexes, qui sont une extension des nombres complexes en algèbre. On peut également citer Samuel Beckett, lauréat du prix Nobel de littérature, principalement connu pour sa pièce absurde, "En attendant Godot".

### 1.1.1 Organisation

Le **TCD** est administré par un conseil d'administration comprenant le provost, équivalent du directeur, élu pour un mandat de 10 ans et validé par le gouvernement irlandais, le vice-provost, le chef des études, le registraire et l'intendant. Ce conseil inclut également six fellows, ce sont des membres du personnel qui ont eu un impact important dans leurs domaines d'enseignement, cinq autres membres du personnel académique, dont au moins trois maîtres de conférences, deux professeurs, trois membres du personnel non universitaire, quatre étudiants dont au moins un de troisième cycle, un membre externe choisi par un comité du conseil parmi des nominations d'organisations représentatives d'intérêts commerciaux ou professionnels, et un membre nommé par le ministre de l'Éducation.

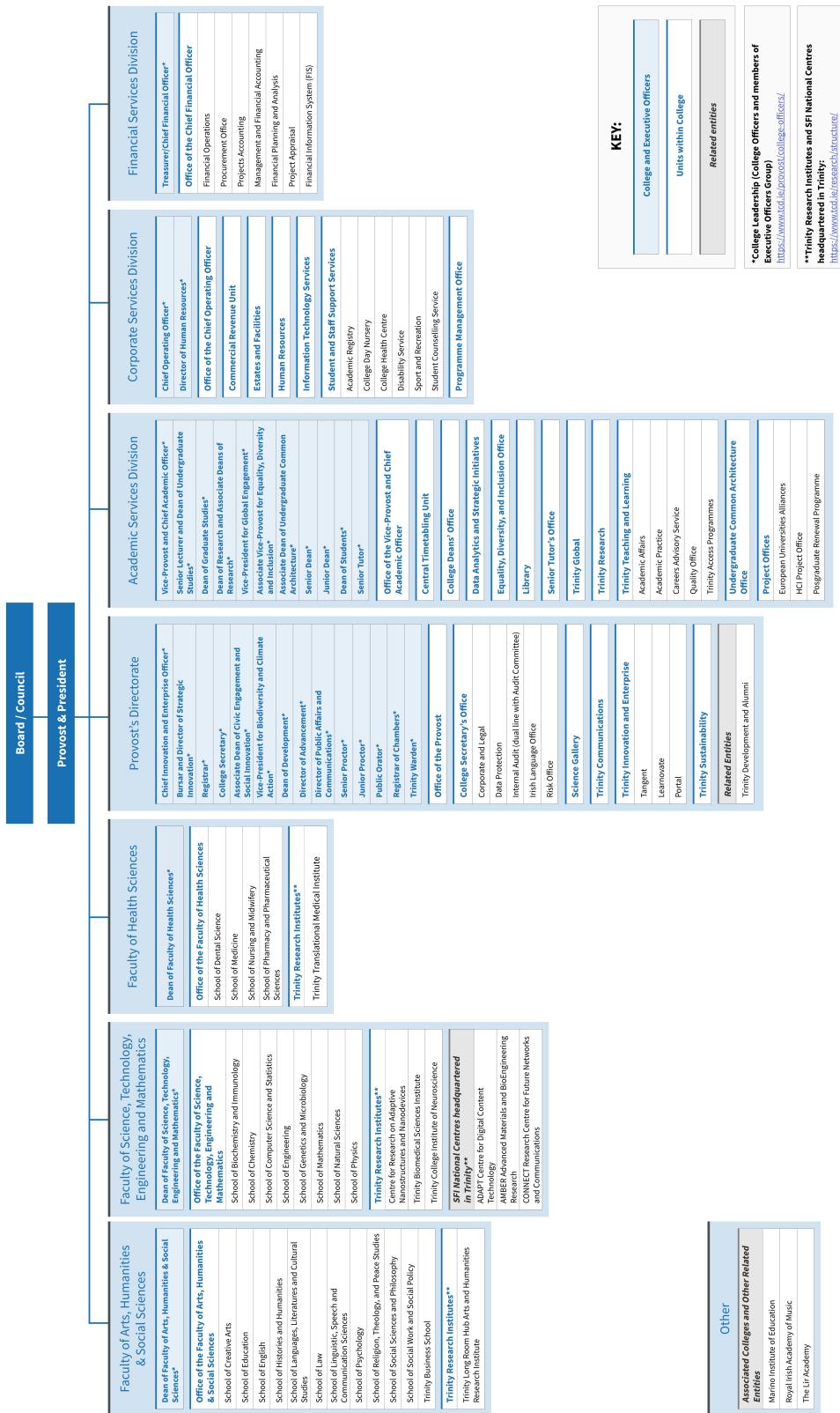


FIGURE 1.1 – Organigramme du TCD

### 1.1.2 Laboratoire de Robotique et d'Innovation (RAIL)

Le **RAIL** est un laboratoire du département de Génie Mécanique, de Fabrication et Biomédical, fondé en 1980. Les membres principaux de ce laboratoire sont le Dr. KELLY et le Dr. LYNCH, qui ont été les tuteurs des étudiants de l'ISIMA, ainsi que des étudiants du cycle Ingénierie et Management. Pour obtenir leur diplôme de fin d'année, les étudiants de ce cycle doivent réaliser en groupe un projet original répondant à une problématique d'ordre public. Le laboratoire est également fréquenté par des doctorants et d'autres stagiaires qui effectuent des travaux d'été pour valider leur cycle universitaire.

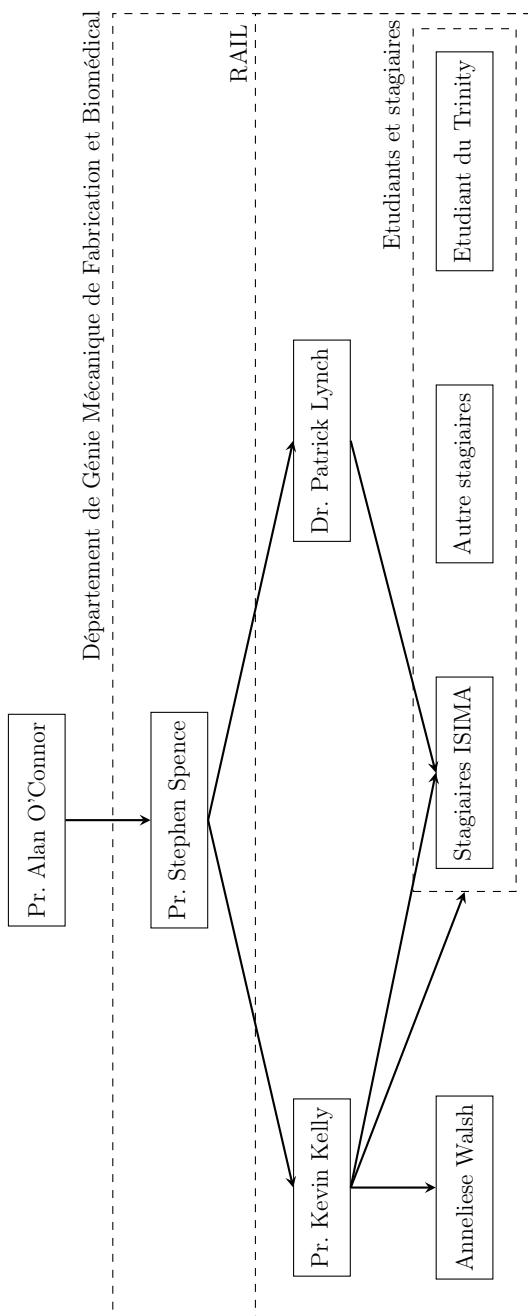


FIGURE 1.2 – Organigramme du **RAIL**

## 1.2 Objectif du stage

L'évaluation de travaux, d'individus ou de concepts se fait actuellement principalement à l'aide de **jugements absous**. L'émergence des **jugements comparatifs** semble indiquer que ces derniers soient plus précis et efficaces lorsque l'on souhaite avoir une notation représentative du groupe évalué. Le Dr. KELLY propose donc une nouvelle méthode, le **CTJ**, qui est une méthode basée sur les **jugements comparatifs**, mais qui aurait des performances supérieures à celles pré-existantes.

Après des tests, principalement réalisés au moyen d'**applications web** [1], il a été décidé de créer une **bibliothèque Python** afin de poursuivre les tests sur cette méthode. En fin de compte, l'objectif est de la distribuer pour permettre à un plus grand nombre de personnes de contribuer aux tests et de vérifier si elle est effectivement plus performante que les méthodes préexistantes.

Pour ce faire, la bibliothèque devra fournir le moyen d'effectuer un **jugement absolu**, ainsi qu'un **jugement comparatif** classique, utilisant une méthode dont l'efficacité a été prouvée, en l'occurrence le choix s'est porté sur le jugement adaptatif comparatif (**ACJ**) [2]. Enfin, elle devra intégrer la méthode que l'on cherche à tester, à savoir le triple jugement comparatif (**CTJ**) [1].

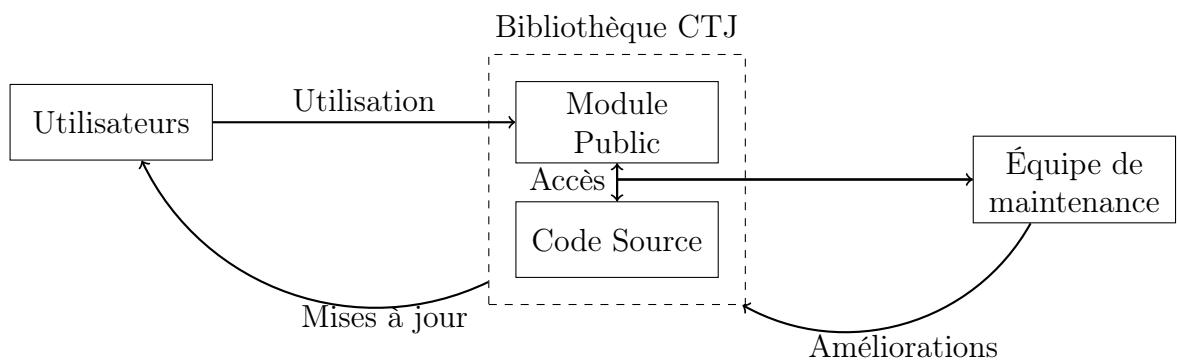


FIGURE 1.3 – Schéma fonctionnel de la bibliothèque

### 1.2.1 Méthode absolue

La méthode absolue consiste en la réalisation d'un **jugement absolu** pour chaque élément de l'ensemble. Elle est la plus répandue lorsqu'il s'agit d'évaluation et correspond à l'attribution d'une valeur à un élément en fonction d'un barème prédéfini, sans avoir de connaissance globale sur l'ensemble des éléments composant le jeu de données à évaluer.

La problématique de cette méthode réside dans plusieurs aspects. Tout d'abord, le classement qui en découle est représentatif d'un élément pris individuellement, mais il n'est pas nécessairement représentatif du groupe dans son ensemble. De plus, l'attribution d'une valeur brute peut devenir complexe si les concepts évalués s'avèrent eux-mêmes complexes. En outre, les évaluateurs peuvent être biaisés par les éléments évalués juste avant ou après, ce qui peut altérer la pertinence de leurs jugements.

Soit un ensemble de 5 éléments  $\{A, B, C, D, E\}$ , on attribue à chaque élément une note.

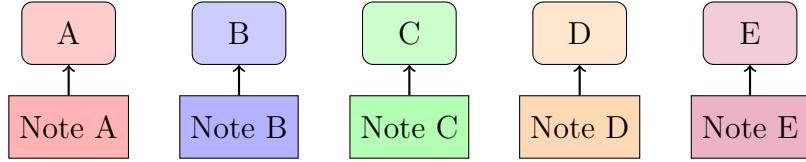


FIGURE 1.4 – Jugement absolu

Ainsi, la nécessité d'utiliser des **jugements comparatifs** semble plus appropriée pour obtenir un classement et une évaluation plus représentatifs de l'ensemble à évaluer, tout en supprimant certains biais et incertitudes.

### 1.2.2 Méthodes comparatives

Les méthodes comparatives, ou **jugements comparatifs**, consistent à considérer les éléments comme faisant partie intégrante de l'ensemble. Ainsi, les évaluations ne se concentrent plus sur un unique élément à la fois, mais sur la comparaison des éléments entre eux. Ainsi, à chaque évaluation, un **n-uplet** d'éléments est évalué, et l'évaluateur est chargé de trier les éléments présentés des meilleurs aux pires. Ensuite, un algorithme détermine une valeur associée à chaque élément. [3]

L'enjeu majeur des méthodes comparatives est de sélectionner les éléments présentés afin qu'ils apportent le plus d'informations possibles au système. L'objectif étant de réduire à un nombre raisonnable le nombre d'évaluations à effectuer.

#### ACJ

L'**ACJ** améliore la méthode standard de **Jugement comparatif** pour une évaluation d'un **couple** d'éléments.

Soit un **couple** d'éléments  $A, B$ , les éléments sont présentés lors du jugement comme suit.



Si B est meilleur que A, l'utilisateur doit les trier dans l'ordre suivant, en supposant un classement du meilleur au moins bon de gauche à droite.



FIGURE 1.5 – Itération de l'**ACJ**

Cette amélioration consiste, à chaque évaluation, à présenter un **couple** pouvant apporter plus d'informations au système.

Bien que l'**ACJ** soit plus performant que le **Jugement absolu** et le **Jugement comparatif**

par défaut, un problème persiste. L'**ACJ** retourne un classement des éléments, et l'ajout de la valeur se fait par la suite, ce qui peut entraîner une perte d'informations sur la valeur exacte des éléments, bien que l'ordre soit conservé.

Supposons que toutes les évaluations soient les suivantes :

-Evaluation 1 :



-Evaluation 2 :



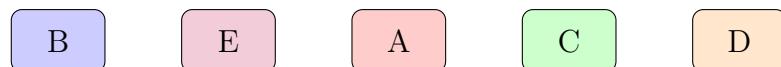
-Evaluation 3 :



-Evaluation 4 :



On en déduit alors l'ordre suivant :



Puis en fonction de cet ordre un algorithme détermine les valeurs associées à chaque élément :

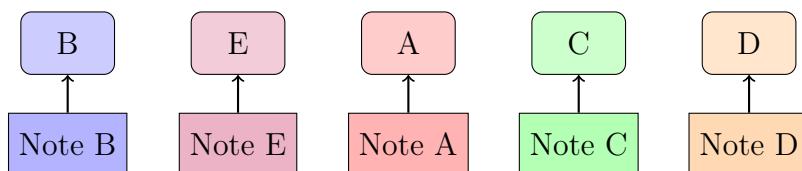


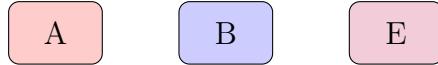
FIGURE 1.6 – Fonctionnement de l'**ACJ**

## CTJ

Le **CTJ** est une méthode dont l'objectif est de créer un classement tout en déterminant simultanément les valeurs associées à chaque élément de l'ensemble. Comme son nom l'indique, il s'agit ici de comparer les éléments trois par trois, et ainsi, à chaque évaluation, un **trio** est évalué.

L'évaluateur doit alors trier le **trio** du pire au meilleur et placer l'élément du milieu à une distance  $d_{best}$  du meilleur élément et  $d_{worst}$  du pire.

Soit un **trio** d'éléments  $A, B, E$  présenté comme suit :



Si le **trio** ordonné du meilleur au moins bon est  $B, E, A$ , alors le jugement de l'utilisateur doit correspondre à :

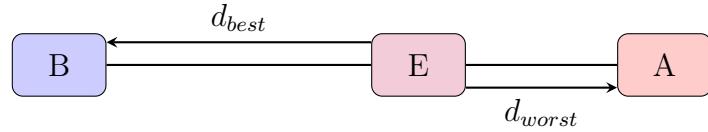
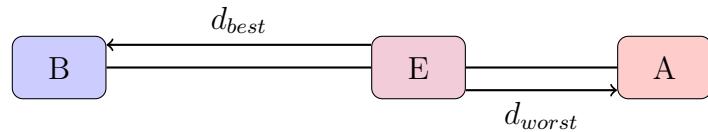


FIGURE 1.7 – Itération du **CTJ**

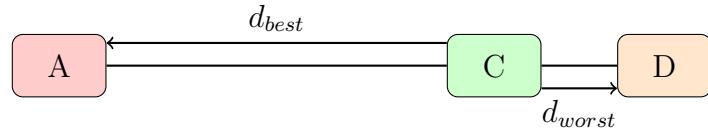
Et ce sont ces évaluations successives qui vont permettre de trier les éléments et d'estimer leurs valeurs grâce à un simple système linéaire.

Supposons que les évaluations soient les suivantes :

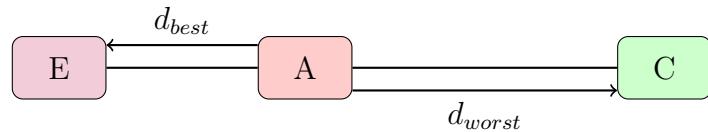
- Evaluation 1 :



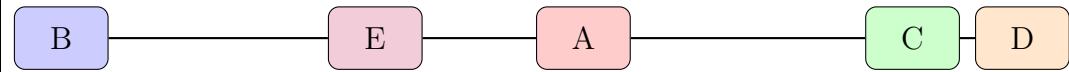
- Evaluation 2 :



- Evaluation 3 :



On en déduit alors l'échelle suivante :



Ensuite, la connaissance des éléments extrêmes permet de déduire les valeurs des autres éléments. Dans le cas où l'on connaît pas les valeurs des éléments extrêmes, on introduit alors deux éléments théoriques qui correspondent aux bornes de ce que l'on souhaite évaluer.

FIGURE 1.8 – Fonctionnement du **CTJ**

La problématique majeure de ce modèle est de savoir comment sélectionner les **trios** apportant le plus d'informations au système.

# 2 Travail effectué

## 2.1 Organisation

### 2.1.1 Méthode semi-agile

Pendant le stage, nous suivions une méthode semi-agile. Chaque mercredi et vendredi, nous consacrons, avec nos tuteurs, une heure pour discuter de nos avancées, de nos prévisions et des problèmes rencontrés avec l'ensemble des stagiaires, qu'ils soient de l'ISIMA ou non. Nous échangions également sur chaque projet afin de réfléchir à des solutions pour surmonter les obstacles rencontrés.

Pour ce faire, nous utilisions l'application Trello, qui permet de suivre les avancées de plusieurs projets simultanément et de créer des plannings. Nous classions nos informations en trois catégories : À faire, En cours et Complété. Ce Trello était mis à jour dynamiquement à chaque début ou fin de tâche.

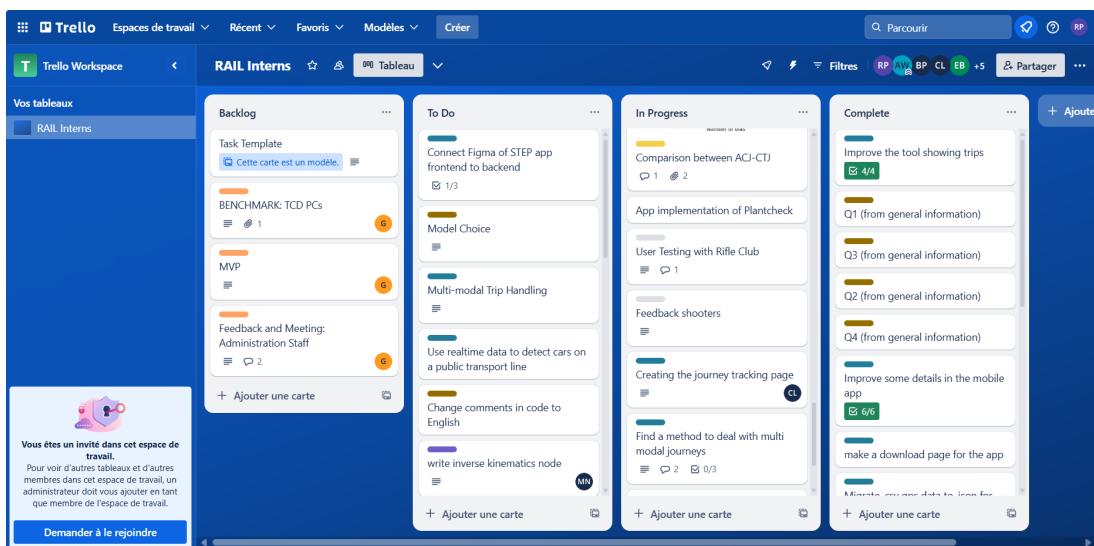


FIGURE 2.1 – Exemple du Trello

## 2.1.2 Planning du stage

Pendant le suivi de l'évolution de mon stage, j'ai créé deux diagrammes de Gantt. À l'origine, je prévoyais de consacrer davantage de temps à l'implémentation des modèles et je n'avais pas envisagé d'ajouter des interfaces graphiques ni d'étudier les informations des éléments. De plus, afin d'optimiser mon temps durant le stage, j'ai finalement rédigé mon rapport en parallèle du projet, sans lui allouer une semaine spécifique.

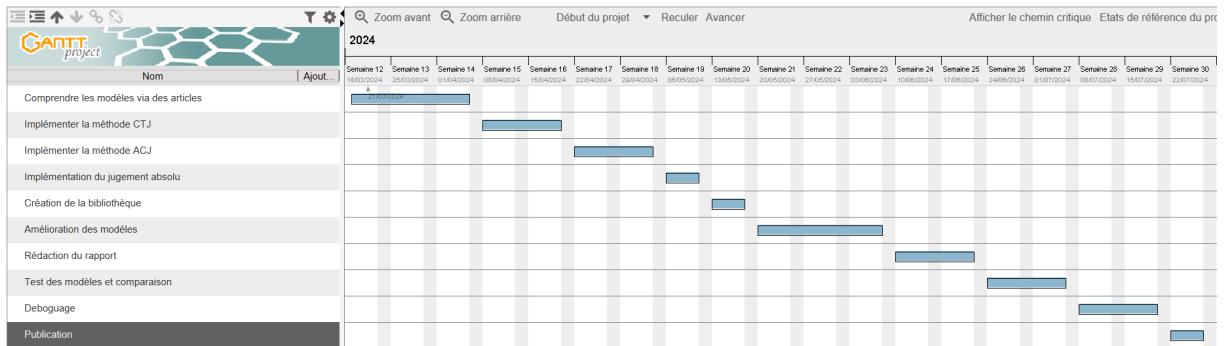


FIGURE 2.2 – Gantt prévisionnel

Initialement, je prévoyais de commencer directement l'implémentation de la **bibliothèque**, puis d'améliorer les modèles ainsi codés. Mais un des projets de l'an passé n'avait pas été totalement finalisé, donc mon tuteur m'a proposé de finir un site web en y implémentant le modèle **CTJ**, puis de me pencher sur le développement de la **bibliothèque**. Ainsi, avant de commencer pleinement mon projet, j'ai dû acquérir des compétences en développement d'**applications web**, en effet je n'en avais jamais développé auparavant. Cependant, ayant des bases en HTML, j'ai pu acquérir les compétences nécessaires en une semaine, puis finir le site web et enfin développer la **bibliothèque**.

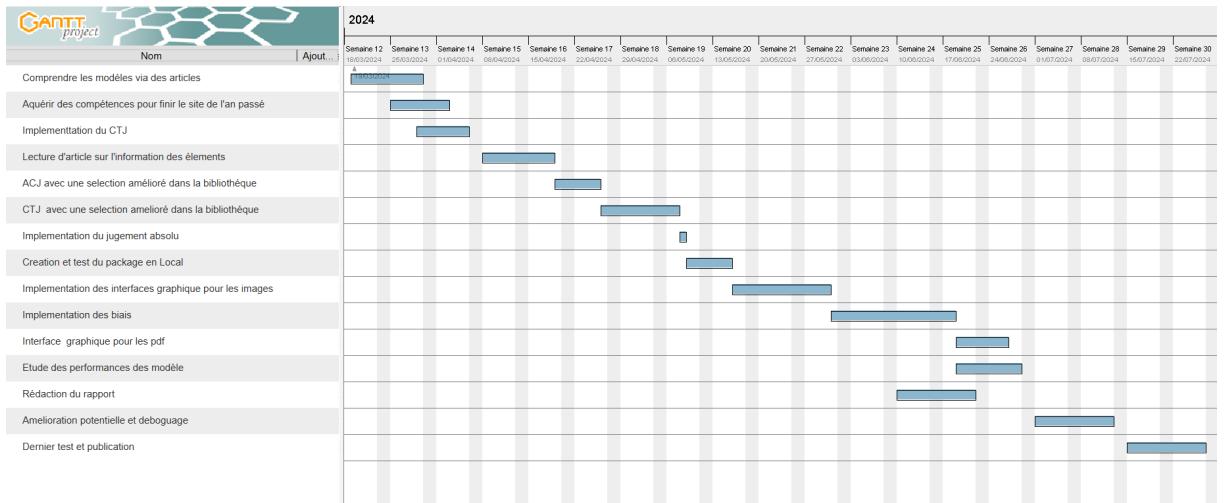


FIGURE 2.3 – Gantt réel

Certaines tâches se sont sous-divisées en plusieurs tâches. Certaines ont pris plus de temps, comme la recherche d'informations sur la manière de comparer des éléments. À l'inverse, d'autres ont été plus courtes, par exemple l'implémentation du **jugement absolu**. J'ai également utilisé le temps qui me restait pour développer deux interfaces graphiques distinctes, ce qui n'était pas prévu initialement : une permettant de visualiser des images et l'autre pour visualiser des fichiers au format pdf.

## 2.2 Etude du Problème

Les modèles étant présents dans la littérature ou ayant un contact avec le créateur de l'un des modèles, leur implémentation a été facile. Cependant, une question s'est rapidement posée : comment comparer deux modèles si leurs optimisations et biais ne sont pas équivalents ? Il a donc été nécessaire, dans un premier temps, de réfléchir à la manière d'optimiser les modèles pouvant l'être.

### 2.2.1 Sélection des éléments

En l'occurrence, cette optimisation a été réalisée sur les modèles **ACJ** et **CTJ**. En effet, le **jugement absolu** n'a pas besoin d'optimisation, car il a un nombre d'itérations fini, correspondant au nombre d'éléments à évaluer. Après tout, chacune de ses itérations consiste à évaluer un élément différent. Le problème s'est donc posé uniquement pour les deux autres modèles. Cette optimisation consiste en la sélection des éléments à comparer afin de permettre aux évaluateurs de fournir des jugements faciles et de qualité.

Ainsi, deux versions de l'**ACJ** seront implémentées : la version classique où l'information est calculée avec une formule spécifique et les éléments sont sélectionnés spécifiquement, et une variante, utilisant le même procédé que le **CTJ**.

#### Pour l'**ACJ** classique

Pour utiliser la méthode **ACJ** classique, on va utiliser la probabilité qu'un élément,  $e_1$ , soit préféré à un élément  $e_2$ , sachant leur valeur, respectivement  $v_1$  et  $v_2$  [4].

$$P(e_1 \text{ meilleur que } e_2 | v_1, v_2) = \frac{\exp(v_1 - v_2)}{1 + \exp(v_1 - v_2)} \quad (2.1)$$

(Probabilité qu'un élément soit meilleur qu'un autre)

Politt (2011) [5] définit l'information apportée par un jugement comme étant le produit de la probabilité définie précédemment 2.1 et celle de l'évènement inverse.

$$I = p(1 - p) \quad (2.2)$$

(Information d'un jugement définie par Politt)

On remarque que le maximum de l'information de Politt est atteint pour une probabilité de 0.5.

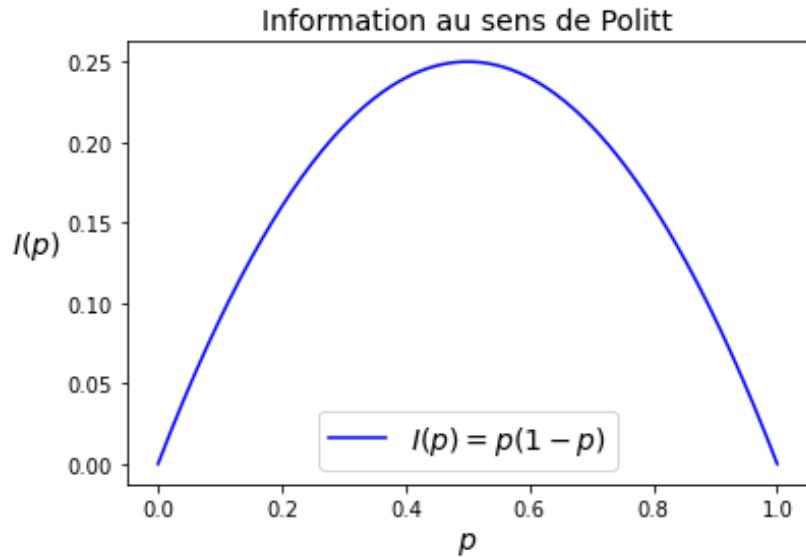


FIGURE 2.4 – Information au sens de Politt

Ainsi, pour l'optimisation de l'**ACJ**, il faut sélectionner les **couples** apportant le plus de cette information au système.

On remarque que la valeur de probabilité qui induit un apport d'informations maximum correspond au fait que la valeur des deux éléments soit égale. En effet, si  $v_1 = v_2$ , on a l'équation 2.1 qui se réécrit comme suit :

$$P(e_1 \text{ meilleur que } e_2 | v_1, v_2) = \frac{1}{2} \quad (2.3)$$

(Probabilité qu'un élément soit meilleur qu'un autre si  $v_1 = v_2$ )

On en déduit que plus deux éléments sont proches, plus l'information de Politt apportée est importante. Le nouveau **couple** sélectionné sera donc celui où la probabilité 2.1 est la plus proche de 0.5.

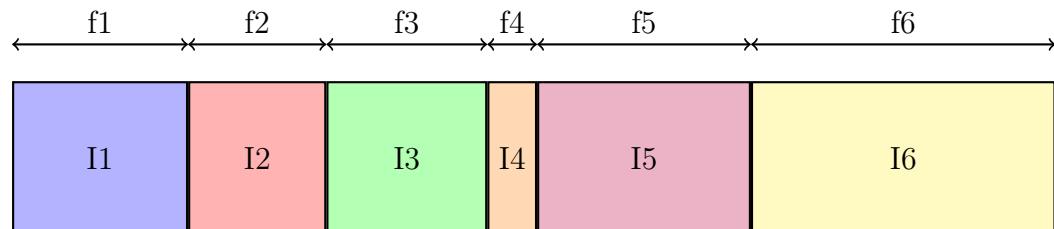
### Sélection roulette et entropie de Shannon

Cette sélection d'éléments, pour la variante de l'**ACJ** et pour le **CTJ**, se fait grâce à une **sélection roulette** sur les **n-uplets** à comparer, afin d'introduire de l'aléatoire. En effet, les jugements apportant le plus d'informations ne sont pas toujours les meilleurs à effectuer.

Une **sélection roulette** consiste, comme son nom l'indique, à faire "tourner" une roulette virtuelle ayant des cases de tailles différentes. Il faut donc réfléchir au calcul de la longueur de ces cases. [6]

Individu	Fitness
I1	f1
I2	f2
I3	f3
I4	f4
I5	f5
I6	f6

Soit 6 individus  $I_1, I_2, I_3, I_4, I_5, I_6$ . On définit la fitness de chaque individu avec une fonction éponyme. Elle correspond à l'"épaisseur" de chaque individu.



On fait ensuite tourner une roulette virtuelle pour sélectionner un individu :

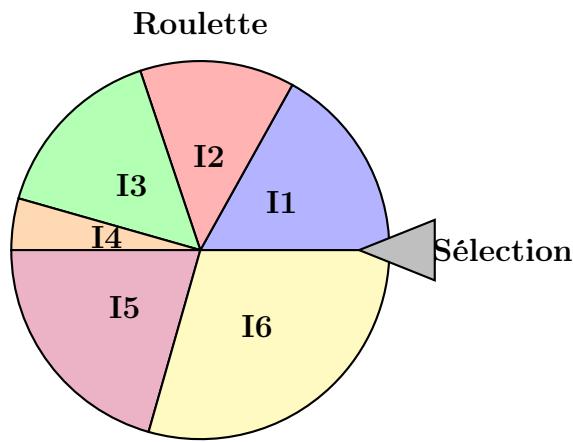


FIGURE 2.5 – Sélection roulette

Pour ce faire il faut comprendre ce qu'est la quantité d'informations d'un élément pour une suite de jugements.

On utilise l'[entropie de Shannon](#) pour expliciter cette quantité d'informations appropriée par un élément. On définit  $p_i$  qui représente la probabilité de tirer l'élément  $i$  parmi tous les éléments déjà comparés. On note  $X_i$  l'évènement associé à ce tirage.

Dans notre cas, nous sommes face à un évènement binaire : soit on tire cet élément, soit on ne le tire pas. Ainsi, on peut écrire l'[entropie de Shannon](#) comme suit. [7]

$$H(X_i) = -p_i \log p_i + (1 - p_i) \log(1 - p_i) \quad (\text{Entropie de Shannon}) \quad (2.4)$$

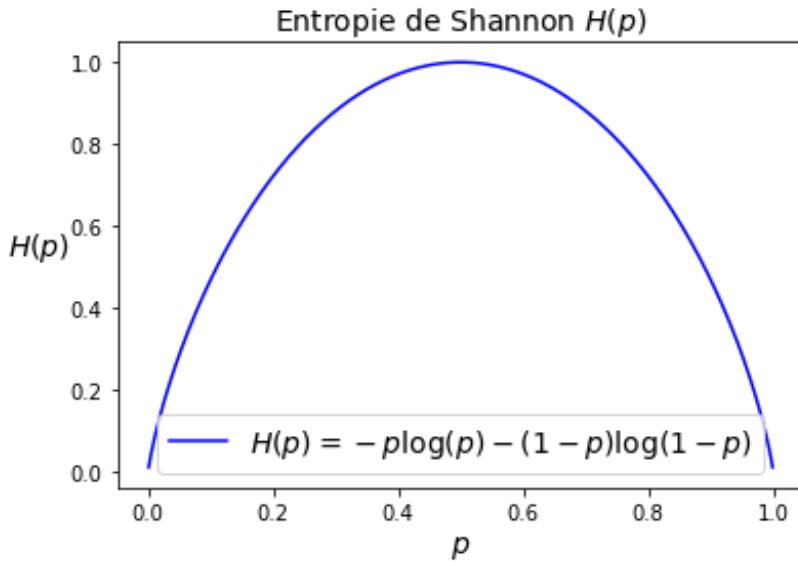


FIGURE 2.6 – Entropie de Shannon (quantité d’informations)

### Pour la variante de l’ACJ

Ainsi, pour la variante de l’ACJ, il est important de connaître la quantité d’informations (au sens de la théorie de l’information) apportée par le couple évalué. Il s’avère qu’il existe une formule permettant de déterminer l’information mutuelle, soit la quantité d’informations apportée par un couple d’éléments au système.

$$MI(X_i; X_j) = H(X_i) + H(X_j) - H((X_i, X_j)) \quad (\text{Information mutuelle}) \quad (2.5)$$

Où  $H(X)$  correspond à l’entropie de Shannon définie précédemment. Et où  $H((X, Y))$  correspond à l’entropie jointe. [7] Ici, l’évènement  $X_i$  correspond au fait de tirer l’élément  $i$  parmi les jugements déjà effectués, de même pour  $X_j$  avec l’élément  $j$ .

$$H((X_i, X_j)) = - \sum_{x_i} \sum_{x_j} p(x_i, x_j) \log p(x_i, x_j) \quad (\text{Entropie jointe}) \quad (2.6)$$

Sachant que des éléments plus proches fournissent davantage d’informations supplémentaires (au sens de Politt), je définis la longueur d’une case (fitness) de la roulette comme suit, chaque case correspondant à un couple. Pour le couple  $(i, j)$ , on a :

$$f(i, j) = MI(X_i, X_j) \cdot \text{Prox}(i, j)^2 \quad (2.7)$$

(Fonction fitness pour la sélection roulette pour l’ACJ)

Avec  $\text{Prox}(i, j)$ , la proximité de deux éléments est définie comme suit :

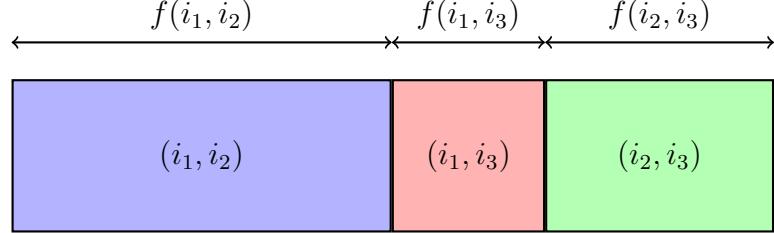
$$\text{Prox}(i, j) = \frac{\text{MAX\_VALUE}}{|\text{estimated\_values}_i - \text{estimated\_values}_j|} \quad (2.8)$$

(Proximité des éléments  $i$  et  $j$ )

Avec  $\text{MAX\_VALUE}$  la valeur maximale théorique associée au jeu de données, et  $\text{estimated\_values}_i$  correspondant à la valeur estimée pour l’élément  $i$  lors de la dernière itération, de même pour  $\text{estimated\_values}_j$  avec l’élément  $j$ .

Ainsi si la roulette tombe sur une case, cela signifiera que le **couple** correspondant a été sélectionné.

Soit un ensemble de 3 éléments  $i_1, i_2, i_3$ . On définit la fitness de chaque **couple** avec la fonction fitness (2.7).



On fait ensuite tourner la roulette pour sélectionner un **couple**.

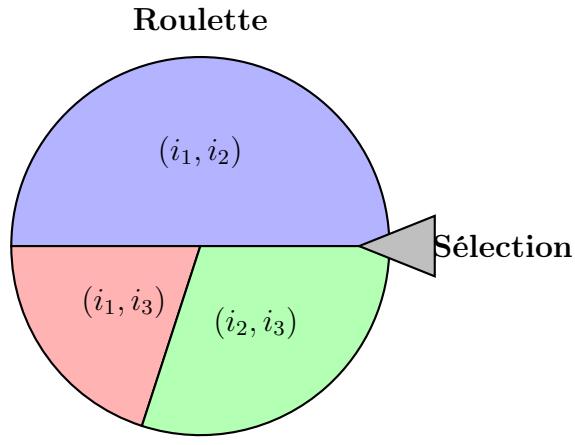


FIGURE 2.7 – Sélection roulette pour ACJ

### Pour le CTJ

Ainsi, pour le **CTJ**, il est important de connaître la quantité d’informations apportée par le **trio** évalué. Pour ce faire, il a fallu utiliser l'**information d’intéraction**. Cette dernière consiste à déterminer la quantité d’informations apportée par un **n-uplet** au système. Dans notre cas,  $n = 3$ , j’utilise la formule suivante adaptée aux **trios**.

$$\begin{aligned} II(X_i, X_j, X_k) &= H(X_i) + H(X_j) + H(X_k) \\ &\quad - (H((X_i, X_j)) + H((X_i, X_k)) + H((X_k, X_j))) \\ &\quad + H((X_i, X_j, X_k)) \end{aligned} \tag{2.9}$$

(Information d’intéraction)

Ou  $H(X)$  correspond à l'**entropie de Shannon**,  $H((X, Y))$  correspond à l'**entropie jointe** définie précédemment. Et ou  $H((X, Y, Z))$  est défini comme suit [8]. Ici, l’évènement  $X_i$  correspond au fait de tirer l’élément  $i$  parmi les jugements déjà effectués, de même pour  $X_j$  avec l’élément  $j$ , et  $X_k$  avec l’élément  $k$ .

$$H((X_i, X_j, X_k)) = - \sum_{x_i} \sum_{x_j} \sum_{x_k} p(x_i, x_j, x_k) \log p(x_i, x_j, x_k) \tag{2.10}$$

La difficulté dans la sélection de l'épaisseur de la case dans la **sélection roulette** réside dans la définition de la distance entre trois éléments. J'ai constaté que la proximité mutuelle des trois éléments ne produit pas des résultats satisfaisants. À travers nos expérimentations, j'ai trouvé que la maximisation de la distance euclidienne pour un point en trois dimensions semble plus appropriée.

$$\begin{aligned} \text{Prox}(i, j, k) = & ((\text{estimated\_values}_i - \text{estimated\_values}_j)^2 \\ & + (\text{estimated\_values}_j - \text{estimated\_values}_k)^2 \\ & + (\text{estimated\_values}_i - \text{estimated\_values}_k)^2)^{\frac{1}{2}} \end{aligned} \quad (2.11)$$

(Distance entre trois éléments)

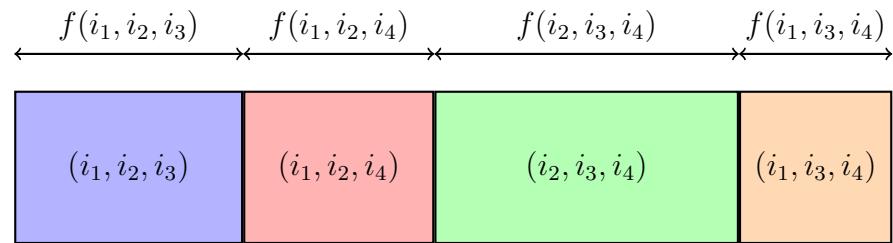
Je définis donc l'épaisseur d'une case comme suit :

$$f(i, j, k) = \Pi(X_i, X_j, X_k)^2 \cdot \text{Prox}(i, j, k) \quad (2.12)$$

(Fonction fitness pour la **sélection roulette** pour **CTJ**)

Ainsi si la roulette tombe sur une case, cela signifiera que le **trio** correspondant a été sélectionné.

Soit un ensemble de 4 éléments  $i_1, i_2, i_3, i_4$ . On définit la fitness de chaque **trio** avec la fonction fitness (2.12).



On fait ensuite tourner la roulette pour sélectionner un **trio**.

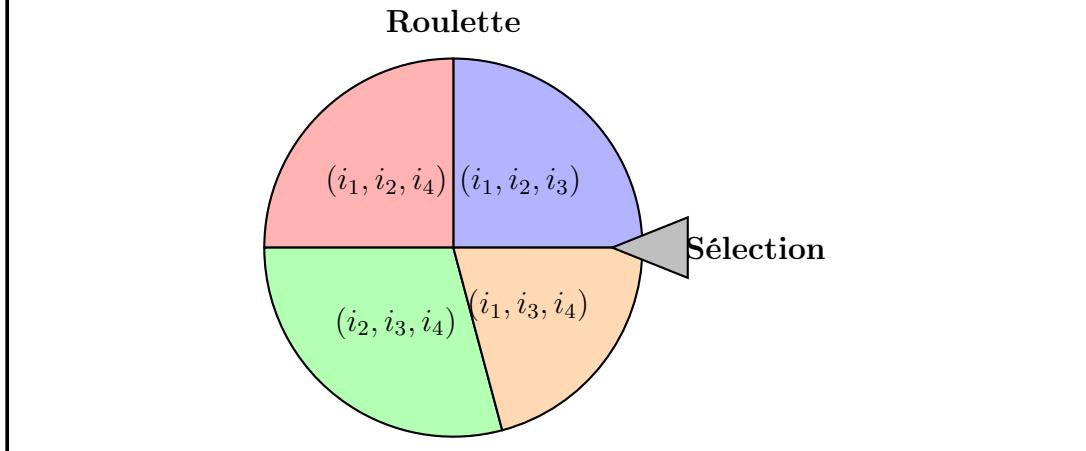


FIGURE 2.8 – **Sélection roulette** pour **CTJ**

## 2.2.2 Introduction du biais

La [bibliothèque](#) développée étant destinée à des tests, il a été important de réfléchir à la manière d'introduire des biais pour comparer les performances des algorithmes face aux erreurs des utilisateurs.

Pour les comparaisons d'éléments, il est nécessaire de considérer un seuil de perception pour les utilisateurs. En dessous de ce seuil, des erreurs sont permises. Afin de refléter le fait que plus les éléments sont proches, plus ils sont susceptibles d'être confondus, j'ai utilisé une fonction sigmoïde ([2.13](#)).

$$\sigma_\lambda(x) = \frac{1}{1 + e^{-\lambda x}} \quad (2.13)$$

Ainsi, par la suite, lorsque je parlerais de seuil de perception, il correspondra à la valeur de l'écart entre deux valeurs qui admettra une probabilité d'inversion de 0.1. En dessous de ce seuil, je considère que l'utilisateur simulé ne confond pas les éléments.

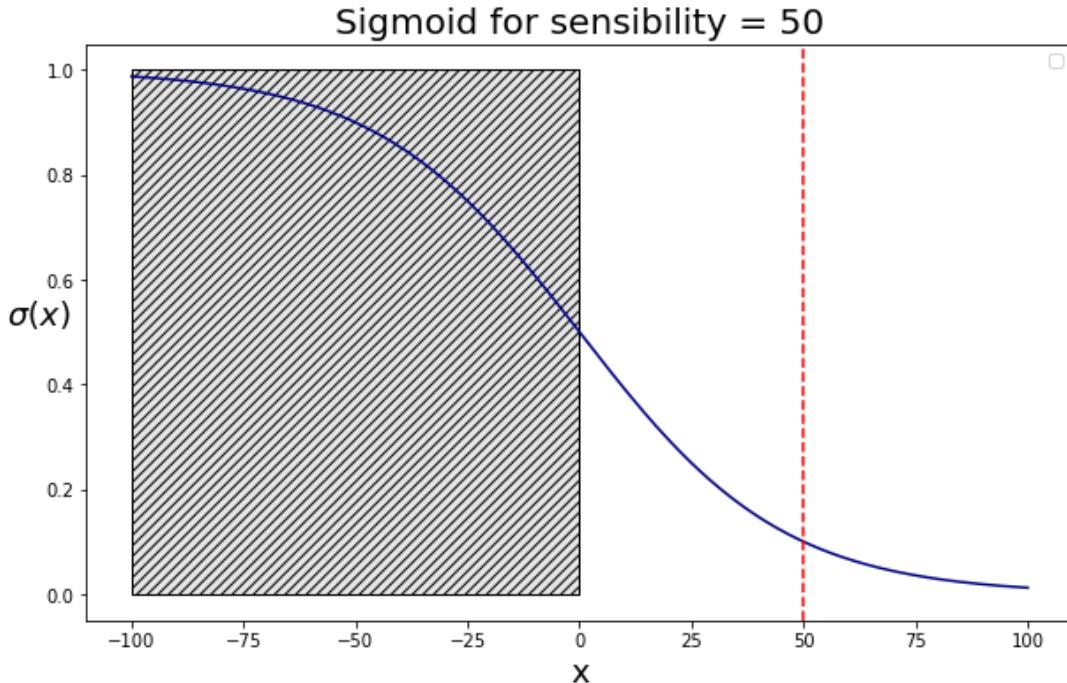
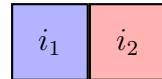


FIGURE 2.9 – Fonction sigmoïde pour  $\lambda = -\frac{\log(\frac{1}{9})}{50}$

### Pour l'ACJ

Dans le cas de l'[ACJ](#), le processus est simple : le biais est introduit uniquement par l'inversion de deux éléments, ce qui correspond nativement à la définition précédente. Ainsi, il suffit de fournir une valeur de seuil de perception à chaque juge simulé, puis je calcule la probabilité que les deux éléments soient inversés.

Soit deux éléments,  $i_1, i_2$  dont le bon ordre est le suivant  $i_1, i_2$ .



Un jugement biaisé correspond alors à :

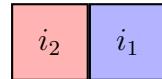


FIGURE 2.10 – Biais pour l'**ACJ**

L'introduction du biais est le même pour la version classique de l'**ACJ** et pour sa variante.

### Pour le CTJ

Cependant, dans le cas du **CTJ**, un problème se pose : en effet, il faut tenir compte que l'on évalue ici trois éléments, et la définition pure du seuil de perception est difficile à satisfaire. Cependant, nous voulons que l'introduction soit aisée, ainsi nous voulons garder une unique valeur à présenter pour le biais d'inversion d'éléments. Ainsi, nous allons procéder de la manière suivante.

Soit trois éléments,  $i_1, i_2, i_3$  dont le bon ordre est le suivant  $i_1, i_2, i_3$ . Notons  $p_{i,j}$  la probabilité d'inverser les éléments  $i$  et  $j$ .

Toutes les possibilités d'inversions sont les suivantes, on étudie cas par cas.

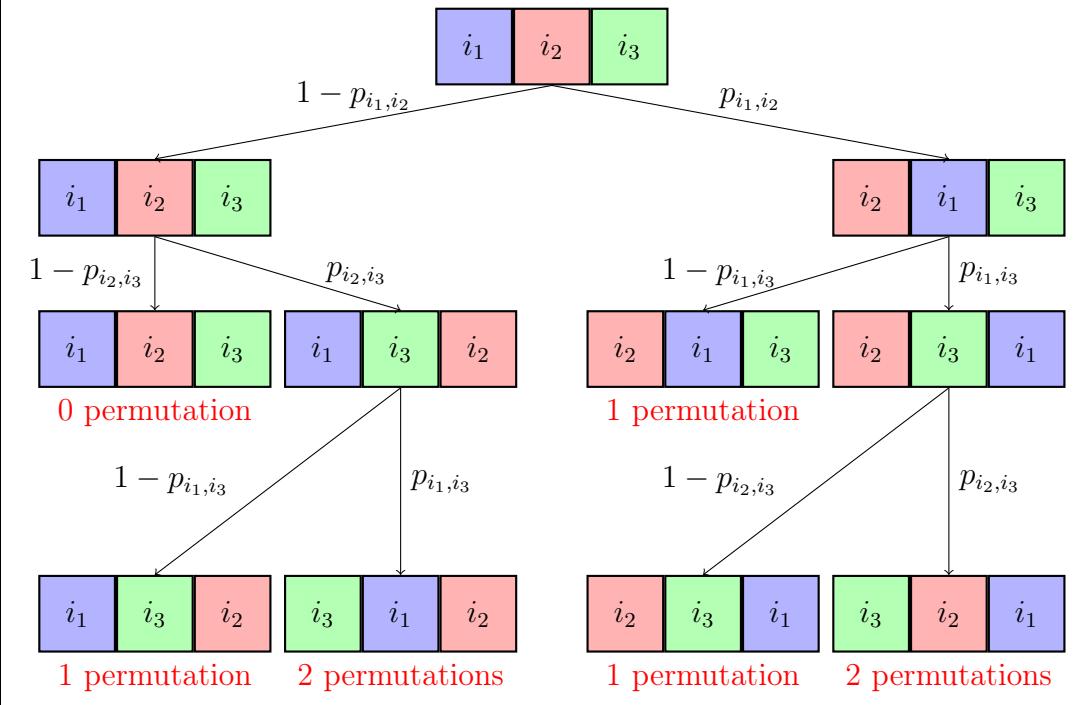


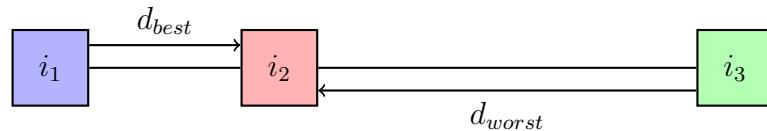
FIGURE 2.11 – Arbre de biais

Sur un seul jugement on peut introduire 0, 1 ou 2 erreur d'inversion. Mais le biais de jugement n'est pas le seul à pouvoir être introduit, en effet il est possible également de faire une erreur sur l'échelle.

Ce biais-ci va s'exprimer avec deux facteurs, le premier est la valeur absolue de l'erreur sur l'échelle et le second la probabilité de faire cette erreur. Ainsi le biais sur **CTJ**, sera défini comme un **trio**, le premier élément étant le seuil de perception, le second la valeur absolue d'erreur sur l'échelle, et le dernier étant la probabilité de l'erreur.

Soit trois éléments,  $i_1, i_2, i_3$  dont le bon ordre est le suivant  $i_1, i_2, i_3$ . Et dont les distances sont  $d_{worst}$  et  $d_{best}$ .

Ainsi la bonne évaluation est la suivante :



Une erreur d'échelle correspond à l'introduction d'un biais  $b$  avec une probabilité  $p_b$  :

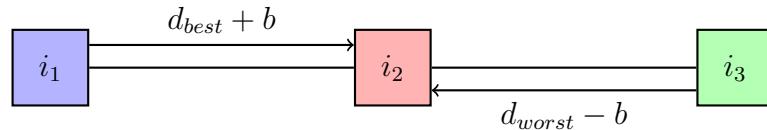
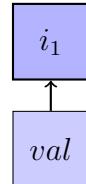


FIGURE 2.12 – Erreur d'échelles pour le **CTJ**

### Pour le jugement absolu

Pour le **jugement absolu**, l'introduction du biais est beaucoup plus simple : il s'agit d'un **couple**, le premier étant l'erreur absolue introduite sur les éléments, et le second la probabilité de faire cette erreur.

Soit un élément,  $i$  dont la valeur est  $val$  :



On introduit un biais  $b$  sur la valeur  $val$  avec une probabilité  $p_b$  :

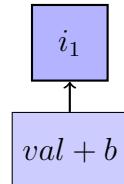


FIGURE 2.13 – Biais pour le **jugement absolu**

## 2.3 Outils

Dans l'objectif d'avoir un projet et une bibliothèque [open source](#), il est nécessaire d'utiliser des bibliothèques qui le sont elles-mêmes et permettent une redistribution correcte.

### 2.3.1 Bibliothèques utilisées

Une [bibliothèque Python](#) est un regroupement de modules accessible par les utilisateurs. Ces derniers peuvent l'importer afin d'utiliser les fonctions qui y sont implémentées.

Le choix des [bibliothèques](#) a été influencé afin que leurs [licences](#) soient en adéquation avec un projet [open source](#).

- **NumPy**

NumPy est une [bibliothèque](#) définissant des opérations mathématiques standard, comme exponentielle, logarithme et racine. Elle est utilisée également lorsque l'on doit manipuler une quantité importante de valeurs. En effet, elle permet de regrouper ces valeurs dans des structures telles que des tableaux ou matrices, et propose également des opérations entre ces structures. Le choix de NumPy est dû à une question de vitesse d'exécution : en effet, les opérations sur ces dernières sont bien plus rapides que sur les listes natives en [Python](#).

- **SciPy**

SciPy est une [bibliothèque open source](#) qui étend les fonctionnalités de NumPy. Elle utilise les fonctions prédéfinies par cette [bibliothèque](#) afin de définir des fonctions mathématiques plus complexes. Je l'utiliserais principalement pour effectuer des calculs statistiques standard.

- **Random**

La [bibliothèque](#) Random est utilisée pour générer des nombres pseudo-aléatoires. Elle offre la possibilité d'utiliser des fonctions pour les opérations aléatoires, telles que le choix d'un élément au hasard dans une liste, la génération de nombres pseudo-aléatoires suivant différentes distributions (uniforme, gaussienne, etc.). Cette [bibliothèque](#) sera utilisée pour effectuer les tirages et implémenter la [sélection roulette](#).

- **Scikit-Learn**

Scikit-Learn est une [bibliothèque open source](#) dédiée à l'apprentissage automatique. Elle fournit des outils simples et efficaces pour la modélisation prédictive, y compris la classification, la régression, le clustering, et la réduction de dimensionnalité. Scikit-Learn est basé sur NumPy et SciPy. Elle sera utilisée pour réarranger des valeurs entre les deux valeurs théoriques extrêmes.

- **Itertools**

La [bibliothèque](#) itertools fournit des outils pour créer et manipuler des itérateurs. Elle inclut des fonctions pour les permutations, les combinaisons, les produits cartésiens, et d'autres opérations itératives. Cette [bibliothèque](#) sera utilisée pour générer des combinaisons d'éléments.

- **Choix**

La **bibliothèque Choix** est un outil pour implémenter des modèles de choix, comme le modèle de Bradley-Terry, utilisé pour le classement des éléments en fonction des préférences. Elle sera utilisée pour déterminer les valeurs estimées lors de l'[ACJ](#).

- **Time**

La **bibliothèque time** permet de manipuler le temps. Elle offre des fonctions pour obtenir l'heure actuelle, mesurer des durées, et créer des pauses dans l'exécution du code. Elle sera utilisée pour chronométrier les performances des utilisateurs lors des jugements.

- **os**

La **bibliothèque os** fournit des fonctions pour interagir avec le système d'exploitation. Elle permet d'effectuer des opérations telles que la gestion de fichiers et de répertoires, l'exécution de commandes système, et l'accès à des variables d'environnement. Elle sera utilisée pour accéder aux représentations graphiques (image ou pdf) des objets à évaluer.

- **Tkinter**

Tkinter est une **bibliothèque** pour créer des interfaces graphiques interactives. Tkinter sera utilisé pour permettre aux utilisateurs de faire les jugements avec les représentations graphiques des éléments importés grâce à **os**.

- **Pillow**

Pillow est une **bibliothèque** pour le traitement des images. Elle permet d'ouvrir, de manipuler et de sauvegarder différents formats d'image. Elle sera utilisée pour afficher des images dans l'interface créée grâce à Tkinter.

### 2.3.2 Crédit de Bibliothèque

Pour pouvoir créer une **bibliothèque**, il faut suivre une architecture particulière.

#### Structure de Répertoire

La première étape pour créer une **bibliothèque Python** est de structurer correctement le répertoire du projet. Voici la structure que j'ai utilisé.

```
CTJ/
|-- CTJ/
|   |-- __init__.py
|   |-- ACJ.py
|   |-- assessment_method.py
|   |-- CTJ.py
|   |-- Rubric.py
|   |-- selection.py
|   |-- util.py
|-- Notebook/
|   |-- ACJ-Tutorial.ipynb
|   |-- black.png
```

```

|   |-- CTJ-Tutorial.ipynb
|   |-- g1.png
|   |-- g2.png
|   |-- g3.png
|   |-- g4.png
|   |-- g5.png
|   |-- Rubric-Tutorial.ipynb
|   |-- white.png
|-- fix_import_error_tkinter.md
|-- LICENSE
|-- MANIFEST.in
|-- README.md
|-- setup.py

```

- CTJ/ est le répertoire principal de la **bibliothèque**.
  - `__init__.py` : Fichier permettant de traiter ce répertoire comme un package **Python**.
  - `ACJ.py`, `assessment_method.py`, `CTJ.py`, `Rubric.py`, `selection.py`, `util.py` : Les différents modules de la **bibliothèque**.
- Notebook/ est le répertoire contenant les exemples et tests pour la **bibliothèque**.
  - `ACJ-Tutorial.ipynb`, `CTJ-Tutorial.ipynb`, `Rubric-Tutorial.ipynb` : Les tests unitaires pour les différents modules.
  - `*.png` : Les PNG correspondent aux images qui seront affichées, ici ce sont des nuances de gris.
- `fix_import_error_tkinter.md` : Un fichier expliquant comment installer `tkinter` s'il n'est pas installé nativement, il est affiché dans une exception générée par la **bibliothèque**.
- `LICENSE` : Fichier de **licence** pour le projet.
- `MANIFEST.in` : Fichier donnant des instructions sur les fichiers à mettre dans la **bibliothèque**.
- `README.md` : Fichier contenant la description du projet.
- `setup.py` : Script de configuration pour l'installation de la **bibliothèque**.

Les fonctions accessibles pour les utilisateurs sont `ACJ`, `Rubric`, `CTJ` définit dans les fichiers python éponymes. Et `ACJ_assessment_image`, `CTJ_assessment_image`, `Rubric_assessment_image`, `ACJ_assessment_pdf`, `CTJ_assessment_pdf`, `Rubric_assessment_pdf` définient dans `assessment_method.py`.

## PIP

PyPI permet un partage simple des bibliothèques, c'est pour ça qu'il a été envisagé de l'utiliser.

Pour installer une **bibliothèque** avec **PIP**, on peut utiliser la commande suivante dans un environnement console :

```
pip install 'nom_de_la_bibliothèque'
```

## Wheel

Le format Wheel est un standard pour la distribution des packages sur [PIP](#). Pour ce faire il faut créer un fichier ‘.whl‘ associé à la [bibliothèque](#). On peut ensuite distribuer ce fichier ou le télécharger sur PyPI pour le rendre disponible via [PIP](#).

## Source Distribution (sdist)

La distribution source (‘sdist’) est un format permettant de distribuer le code source de votre [bibliothèque](#). Voici comment créer une distribution source pour la [bibliothèque](#) : Cette commande créera un fichier ‘.tar.gz‘ dans le répertoire ‘dist/‘. Ce fichier contient tous les fichiers nécessaires pour installer la [bibliothèque](#) à partir du code source. On peut distribuer ce fichier ou le télécharger sur PyPI pour le rendre disponible via [PIP](#).

Dans le cas du local, on peut utiliser ce fichier tar.gz pour installer la bibliothèque localement.

# 3 Résultat

La bibliothèque a fini d'être développée. Dans cette section je vais m'attarder sur son fonctionnement et sur les problématiques futures.

Il y a deux grandes manières de l'utiliser. La première est l'évaluation automatique et la seconde l'évaluation manuelle.

Dans la suite pour illustrer nos résultats nous allons utiliser l'ensemble d'éléments suivants :

```
couleurs = ['g1', 'g2', 'g3', 'g4', 'g5']
```

La liste couleurs représente le nom des éléments que l'on souhaite évaluer ici leurs noms sont  $g_i$  avec  $i$  entre 1 et 5, pour représenter 5 nuances de gris. On définit également valeurs, la valeur associée à chaque nuance en utilisant le niveau de gris pour un octet, ainsi 0 correspond à du noir et 255 à du blanc.

```
valeurs = [160, 106, 209, 80, 135]
```

On définit donc également les deux valeurs théoriques extrêmes, noir avec la plus petite valeur, donc le pire élément et blanc la plus grande donc le meilleur.

```
pire      = [0, 'noir']
best     = [255, 'blanc']
```

## 3.1 Evaluation automatique

Les évaluations automatiques permettent de tester rapidement les modèles tout en pouvant introduire des biais pour comparer le comportement des modèles influencés par les erreurs des juges.

Cependant pour pouvoir effectuer ce genre d'évaluation, la condition sine qua non est la connaissance des valeurs des éléments à comparer. Il faudra donc dans chacune des fonctions définir le paramètre true\_value avec la liste des valeurs de chaque élément.

## Pour la méthode ACJ classique

Par défaut on peut directement l'utiliser comme suit :

```
ACJ(pire, meilleur, couleurs, true_values = valeur)
```

Dans ce cas aucun biais n'est introduit et le jugement est effectué pour un unique juge, le nombre maximal d'itérations est de 30 et la précision du modèle est à 0.95, l'algorithme s'arrête quand l'une de ces deux conditions est remplie. On peut introduire un biais et augmenter le nombre de juges comme suit :

```
ACJ(pire, meilleur, couleurs, true_values = valeur, nb_judge = 3,  
sensibility = [20, 50, 90])
```

La liste de sensibilité correspond à la perception de chaque juge. Pour rappel, la valeur du seuil de perception correspond à la différence de valeur entre deux éléments qu'il peut confondre avec une probabilité de 0.1. Après exécution, l'algorithme retourne dans tous les cas :

- les valeurs estimées de chaque élément
- le nombre d'itérations
- la précision atteinte
- le nombre d'erreurs
- le temps mis pour effectuer les jugements

Et affiche dans la console les résultats principaux, on obtient par exemple avec la dernière définition d'**ACJ** :

```
=====
| Result of ACJ algorithm
| Items : ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'white']
| True values : [0, 160, 106, 209, 80, 135, 255]
| Estimated values : [0, 188, 84, 218, 100, 150, 255]
| Accuracy : 0.9571026766588642
| Number of error : [1. 2. 4.]
| itération : 29
=====
```

## Pour la variante de la méthode ACJ

Pour utiliser la variante de l'**ACJ**, il suffit d'ajouter le paramètre `entropy` comme suit, par défaut il faut `false`, c'est à dire qu'on utilise la version de base, en le mettant à `true`, on utilise la variante utilisant l'[entropie de Shannon](#).

```
ACJ(pire, meilleur, couleurs, true_values = valeur, entropy = True)
```

La seule fonction qui change est celle définissant la manière de sélectionner le nouveau couple d'éléments.

```
ACJ(pire, meilleur, couleurs, true_values = valeur, nb_judge = 3,  
sensibility = [20, 50, 90], entropy = True)
```

```
=====
| Result of ACJ algorithm
| Items : ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'g6', 'white']
| True values : [0, 160, 106, 209, 80, 135, 234, 255]
| Estimated values : [0, 165, 69, 194, 100, 135, 219, 255]
| Accuracy : 0.9607996175572444
| Number of error : [1. 6. 6.]
| Iteration : 26
=====
```

## Pour la méthode CTJ

Par défaut on peut directement l'utiliser comme suit :

```
CTJ(pire, meilleur, couleurs, true_values = valeur)
```

Dans ce cas aucun biais n'est introduit et le jugement est effectué avec une échelle égale à 10, c'est à dire qu'il y a 10 positions possible pour placer l'élément central entre les deux éléments extrêmes. le nombre maximal d'itérations est de 30 et la précision du modèle est à 0.95, l'algorithme s'arrête quand l'une de ces deux conditions est remplie. On peut introduire un biais et changer l'échelle comme suit :

```
CTJ(pire, meilleur, couleurs, true_values = valeur, scale = 30,
sensitivity = (60, 2, 0.3))
```

La sensibilité est un tuple de trois éléments, le premier élément correspond au seuil de perception du juge, le second est la valeur absolue de l'erreur potentielle sur l'échelle et le troisième est la probabilité de faire une erreur sur l'échelle. Dans chaque cas, l'algorithme retourne :

- les valeurs estimées de chaque élément
- le nombre d'itérations
- la précision atteinte
- le nombre d'erreurs sous forme d'un tuple, le premier élément correspond au nombre d'inversion et le second au nombre d'erreurs d'échelle
- le temps mis pour effectuer les jugements

Et affiche dans la console les résultats principaux, on obtient par exemple avec la dernière définition de la fonction **CTJ** :

```
=====
| Result of CTJ algorithm
| Items : ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'g6', 'white']
| True values : [0, 160, 106, 209, 80, 135, 255]
| Estimated values : [0, 147, 63, 212, 82, 134, 255]
| Accuracy : 0.956117726908126
| Number of inversion : 1
| Number of scale error : 1
| itération : 9
=====
```

## Pour le Jugement absolu

Par défaut on peut directement l'utiliser comme suit :

```
Rubric(pire, meilleur, couleurs, true_values = valeur)
```

Dans ce cas aucun biais n'est introduit et le résultat correspondra à l'association des véritables valeurs à chaque élément. On peut introduire un biais comme suit :

```
Rubric(pire, meilleur, couleurs, true_values = valeur,  
sensitivity = (10, 0.5))
```

La sensibilité est un tuple de deux éléments, le premier élément correspond à la valeur absolue de l'erreur potentielle sur la valeur réelle et le second à la probabilité de faire cette erreur. Dans chaque cas, l'algorithme retourne :

- les valeurs estimées de chaque élément
- le nombre d'itérations
- la précision atteinte
- le nombre d'erreurs
- le temps mis pour effectuer les jugements

Et affiche dans la console les résultats principaux, on obtient par exemple avec la dernière définition de Rubric :

```
=====
| Result of Rubric algorithm
| Items : ['black', 'g1', 'g2', 'g3', 'g4', 'g5', 'white']
| True values : [0, 160, 106, 209, 80, 135, 255]
| Estimated values : [0, 166, 110, 227, 93, 130, 255]
| Accuracy : 0.9871270390253974
| Number of error : 4
| itération : 7
=====
```

## 3.2 Evaluation manuelle

Pour permettre aux utilisateurs d'effectuer eux-mêmes les évaluations, j'ai implémenté dans la [bibliothèque](#) des interfaces graphiques. Pour ce faire il doit y avoir dans le répertoire courant des fichiers png ayant le même nom que la liste d'éléments passée en argument. En l'occurrence pour la liste couleur, il doit y avoir dans le répertoire courant des fichiers tel que g2.png.

### Pour la méthode ACJ

Dans le cas de l'[ACJ](#) les deux éléments vont apparaître côté à côté, et il suffira de cliquer sur l'élément que l'on suppose meilleur, ici pour les couleurs, la valeur correspond au niveau de gris, plus elle est importante plus la nuance de gris est claire, il faut donc sélectionner l'élément le plus clair.

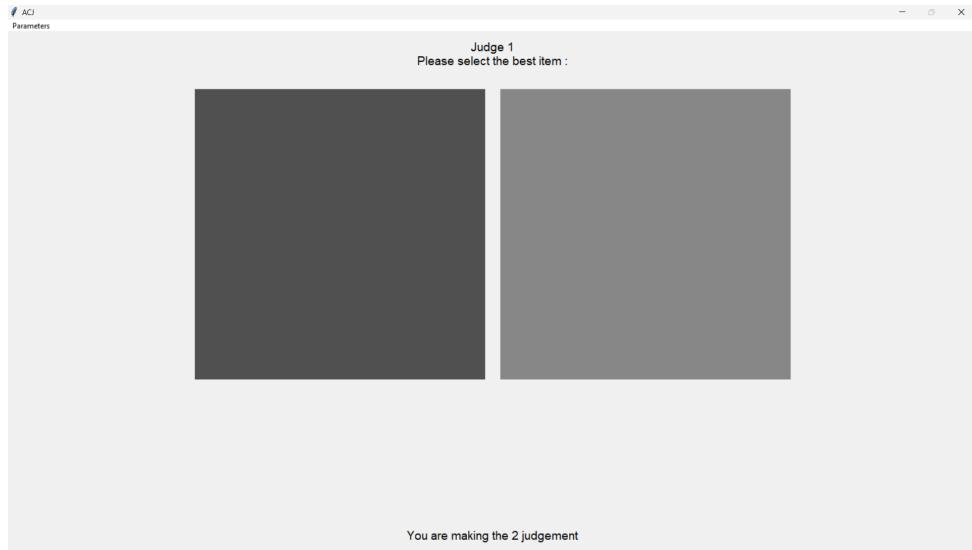


FIGURE 3.1 – Interface de l'**ACJ** pour la visualisation d’images

### Pour la méthode CTJ

Dans le cas du **CTJ** le **trio** va apparaître, sous ce dernier il y a l’échelle pour sélectionner la distance de l’élément central aux deux autres éléments. Pour trier les éléments, il suffit de cliquer sur les deux éléments dont on souhaite inverser les positions et répéter le processus jusqu’à ce que l’on pense que les éléments sont triés. Encore une fois, au vu de notre choix, on trie ici du plus clair au plus sombre. Il suffit ensuite d’utiliser le slider pour placer l’élément central à une distance de l’élément de gauche. Pour passer au jugement suivant il suffit de cliquer sur le bouton **Next**.

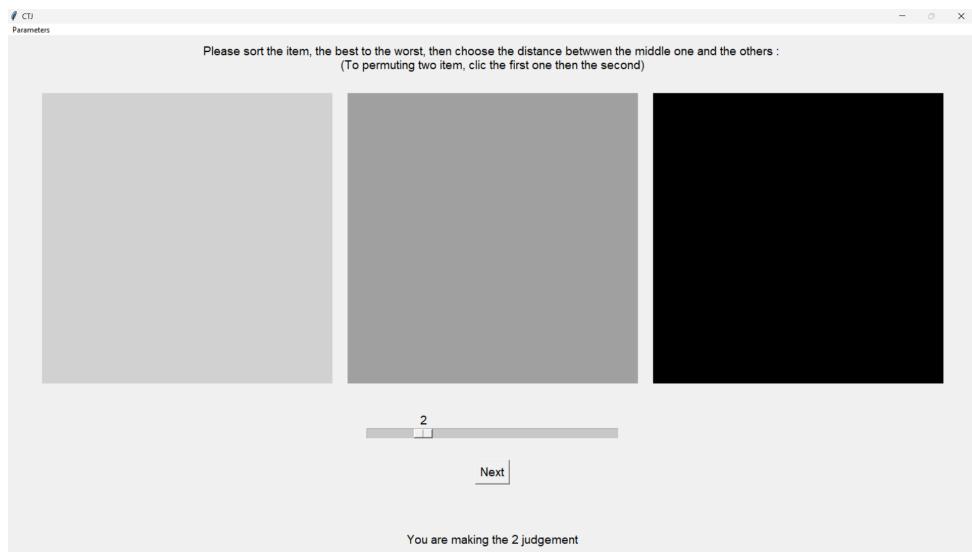


FIGURE 3.2 – Interface du **CTJ** pour la visualisation d’images

### Pour le Jugement absolu

Dans le cas du jugement absolu la représentation graphique de l’élément apparaît, sous laquelle il y a une boîte de saisie où l’utilisateur peut entrer la valeur supposée de cet élément. Pour passer au jugement suivant, il suffit de cliquer sur le bouton **Next**.

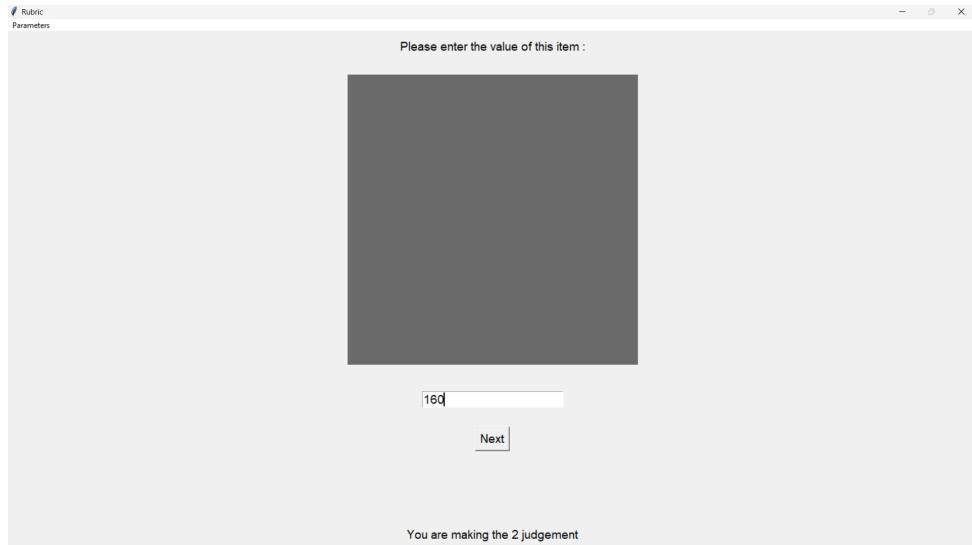


FIGURE 3.3 – Interface du jugement absolu pour la visualisation d’images

### Pour des fichiers pdf

J’ai également implémenté des méthodes d’évaluation pour visualiser des pdf. L’interface est similaire, mais il y a l’ajout d’un bouton sous chaque image afin d’ouvrir le pdf. L’image affichée dans le jugement est la première page du pdf. Voilà l’exemple pour le CTJ.

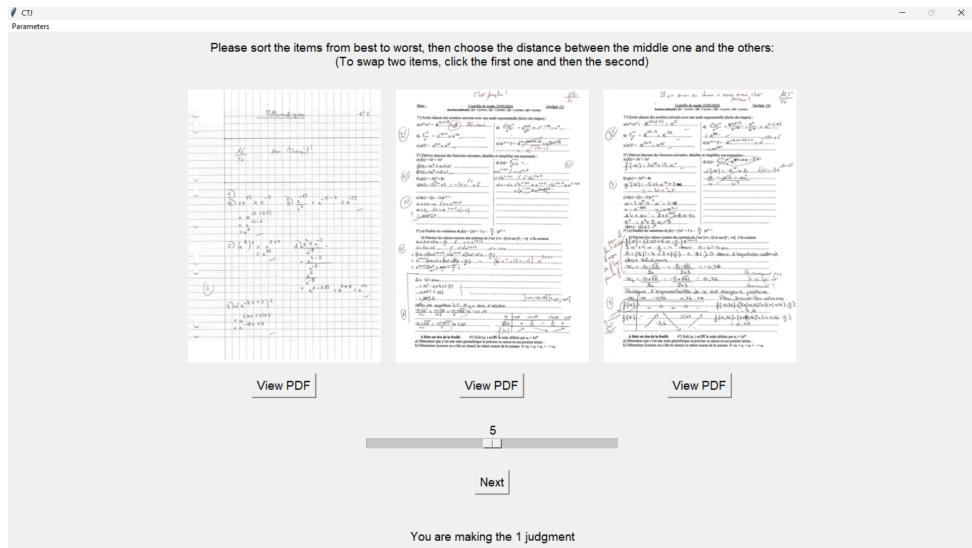


FIGURE 3.4 – Interface du CTJ pour la visualisation d’un pdf

Il y a deux manières d’utiliser les méthodes avec les interfaces graphiques. La première a pour but de tester l’efficacité des modèles lors d’un jugement, et comparer le temps d’exécution des jugements, pour regarder quelle méthode est la plus rapide. La seconde est l’utilisation en conditions réelles des modèles.

### 3.2.1 Test des modèles

Pour le test des modèles, il faut ajouter en plus des éléments nécessaires, les valeurs des éléments, et il faut également ajouter la méthode utilisée pour effectuer le jugement.

Voilà un exemple pour le **CTJ** :

```
CTJ(pire, meilleur, couleurs, true_values = valeur,  
assessment_method = CTJ_assessment_image)
```

### 3.2.2 Conditions réelles

Pour les conditions réelles, il faut ajouter seulement la méthode utilisée pour effectuer le jugement, les valeurs réelles ne sont plus nécessaires, et on calcule la condition d'arrêt en fonction de la convergence des résultats, pour ce faire il est conseillé d'augmenter la précision du modèle.

Voilà un exemple pour le **CTJ** :

```
CTJ(pire, meilleur, couleurs, true_values = valeur,  
assessment_method = CTJ_assessment_image, max_precision = 0.99)
```

## 3.3 Première analyse des modèles

### 3.3.1 Nombre d'erreurs

Dans un premier temps pour analyser les comportements des modèles **ACJ** et **CTJ**, nous allons fixer des erreurs similaires, en l'occurrence, nous fixerons un seuil d'inversion de 60, et sans modifier la valeur d'échelle.

Nous allons observer la précision et le nombre d'itérations en fonction du nombre d'erreurs d'inversion. Nous avons fixé la précision minimale à 0.95 et le nombre d'itérations maximales à 100. Nous prenons une moyenne sur 1000 itérations.

Dans un premier temps nous allons tracer le nombre d'itérations des modèles en fonction du nombre d'erreurs.

De plus, les modèles sont ici étudiés avec les éléments de la liste couleur définis dans la partie 2, il faudra faire de plus amples tests avec divers jeux de données pour pouvoir tirer des conclusions fiables.

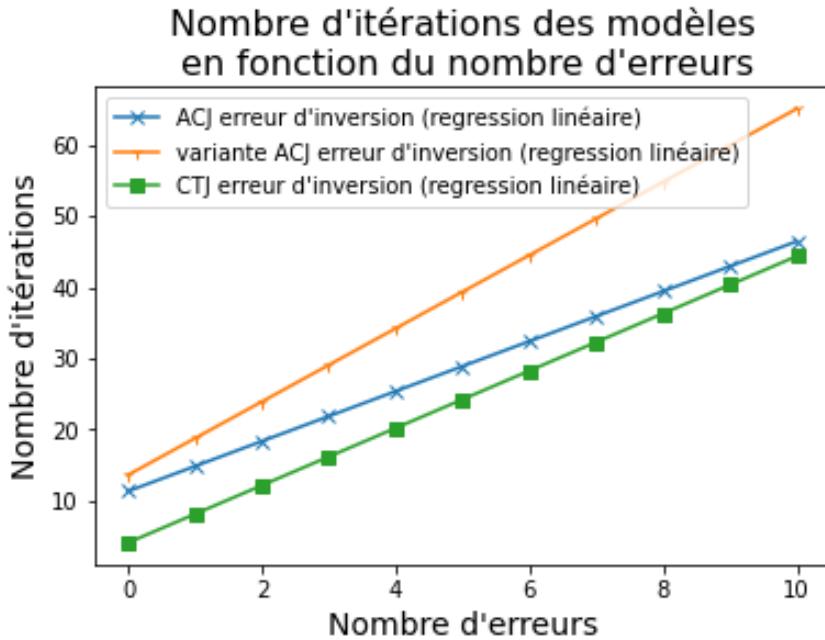


FIGURE 3.5 – Nombre d’itérations des modèles en fonction du nombre d’erreurs

On remarque que le **CTJ** aurait un nombre d’itérations inférieur aux deux versions d’**ACJ** pour un nombre d’erreurs inférieur à 9, puis il semble qu’il fasse plus d’itérations que l’**ACJ** classique, mais moins que sa variante qui est optimisée de la même manière que le **CTJ**. Ici, l’algorithme n’a jamais atteint 100 itérations, on en conclut, que ces résultats sont fiables et correspondent à une précision de plus de 0.95.

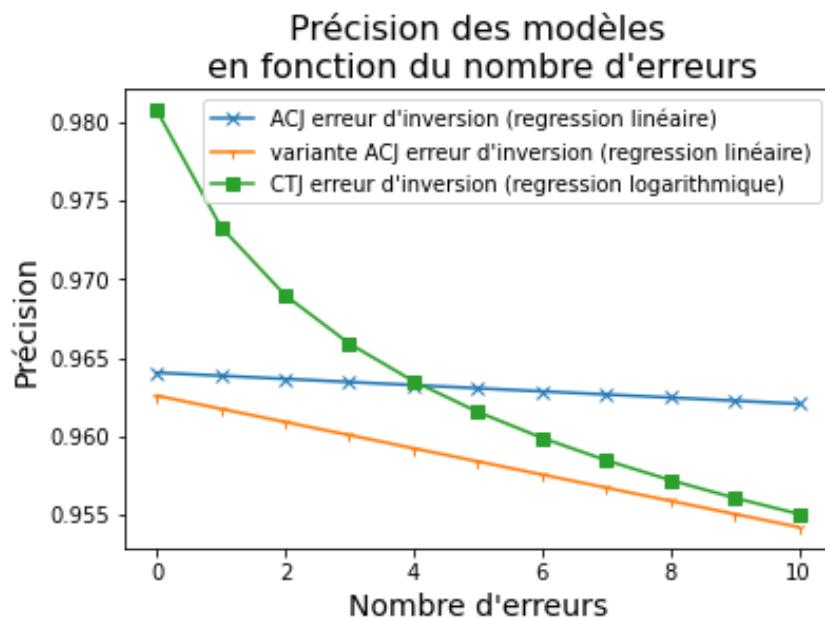


FIGURE 3.6 – Précision des modèles en fonction du nombre d’erreurs

De même on remarque que le **CTJ** a toujours une précision supérieure à celle de la variante de l'**ACJ**. La précision de **CTJ** tend vers celle de la variante de l'**ACJ** quand le nombre d'erreurs augmente. Cependant à partir de 4 erreurs la précision de l'**ACJ** classique est meilleure que celle de la **CTJ**.

### 3.3.2 Seuil de perception

J'ai maintenant fait varier le seuil de perception des juges afin d'évaluer l'efficacité des modèles face à des juges plus ou moins compétents. Le nombre maximal d'itérations est de 200 et la précision minimale requise est de 0,95. Quand l'une des deux conditions est remplie, l'algorithme s'arrête. La sélection des perceptions se fait de 0 à 80 par intervalles de dix.

Les jugements sont toujours effectués avec la liste d'éléments, couleurs, définie dans la partie précédente.

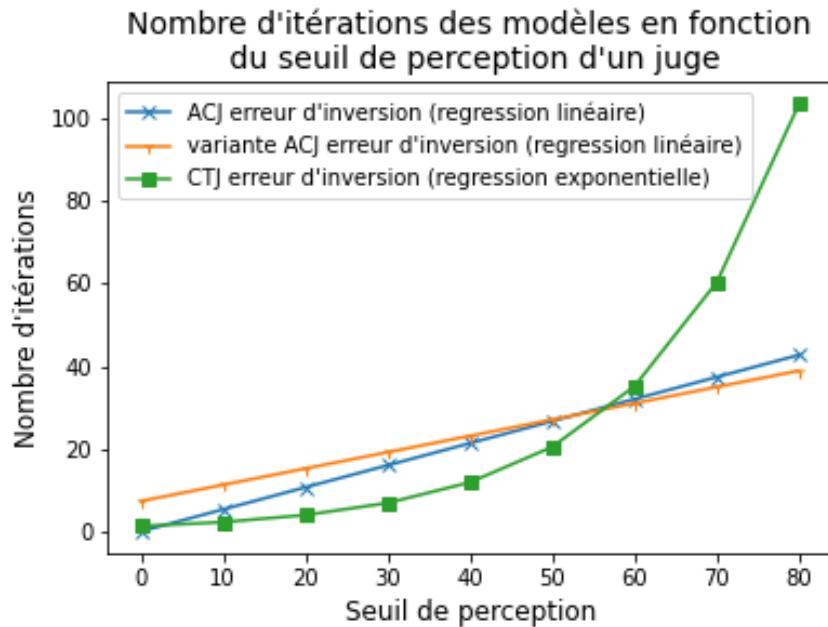


FIGURE 3.7 – Nombre d'itérations des modèles en fonction du seuil de perception d'un juge

On remarque que si le seuil de perception est inférieur à 60, alors le nombre d'itérations nécessaire au **CTJ** pour atteindre au moins 0,95 de précision est moindre que pour toutes les autres méthodes. Cependant, après cette valeur de 60, le nombre d'itérations semble croître fortement, et est supérieur aux deux autres méthodes.

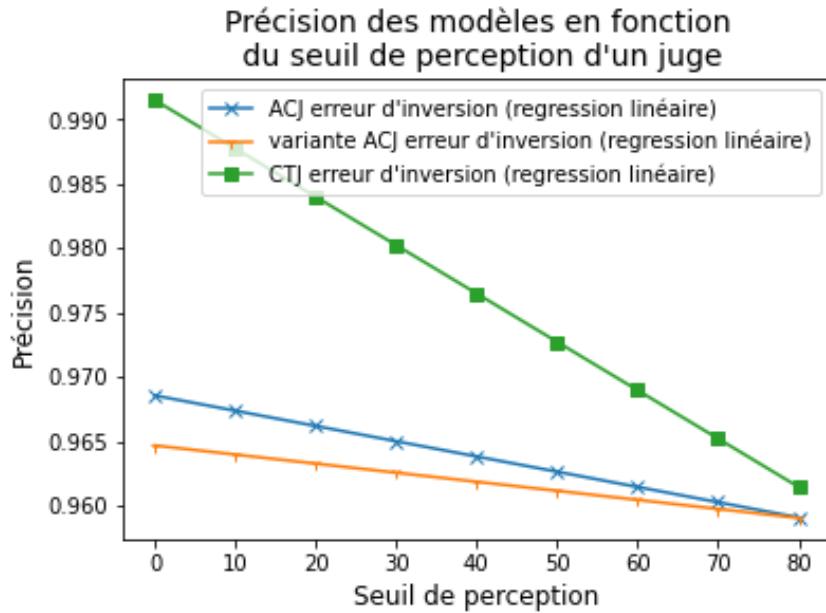


FIGURE 3.8 – Précision des modèles en fonction du seuil de perception d'un juge

Quant à la précision, celle du **CTJ** semble toujours être globalement supérieure à celle des autres méthodes, même si elle tend à se rapprocher de celle de l'**ACJ** lorsque le seuil de précision augmente.

### 3.4 A faire

Le projet était censé être développé comme une **bibliothèque open source**, mais au moment où je finis d'écrire ce rapport, les objectifs ont un peu changé et la **licence** du projet est encore à déterminer, tout comme le moyen de distribution.

En effet, n'ayant aucun papier de recherche écrit au sujet du **CTJ**, la peur que le modèle soit plagié ou utilisé à des fins commerciales n'est pas à exclure. Ainsi la bibliothèque sera probablement publiée ultérieurement une fois qu'un papier de recherche sera publiée à son sujet.

Je vais également devoir continuer d'analyser les modèles et tenter de trouver des liens entre les courbes d'erreurs des **ACJ** et **CTJ**. Tout en utilisant plusieurs jeux de données distincts.

# Conclusion

Je devais créer une [bibliothèque Python](#) permettant l'utilisation et la comparaison de trois méthodes de jugement, le [CTJ](#), l'[ACJ](#) et le [jugement absolu](#). L'objectif était d'avoir une [bibliothèque](#) pouvant tester l'efficacité du [CTJ](#) et de pouvoir l'étudier plus en détail par la suite.

Je l'ai bel et bien implémentée et j'ai également amélioré les sélections des éléments dans les méthodes de comparaison afin que l'optimisation soit similaire pour pouvoir faire une comparaison raisonnable entre les modèles. De plus, j'ai également commencé à faire des analyses des modèles sur des jeux de données. J'ai également implémenté un moyen de rajouter des biais aux jugements automatiques et ai créé des interfaces graphiques permettant l'évaluation d'images et de PDF.

La principale difficulté rencontrée a été l'amélioration de la sélection des éléments. En effet, la notion d'information d'un élément était initialement pour moi assez floue, mais après m'être renseigné, je pense avoir compris ce concept ainsi que les concepts adjacents. Cette difficulté a été un excellent moyen pour moi de me plonger dans un domaine mathématique qui m'était inconnu, la théorie de l'information.

La création de la bibliothèque a également été enrichissante. C'est la première fois que j'ai eu l'occasion de me renseigner sur la manière de publier une [bibliothèque Python](#). Un aspect qui a également été intéressant est la gestion de la [licence](#) du projet. En effet, initialement le projet devait être [open source](#) avec une [licence](#) permissive. Cependant, ce ne sera probablement pas le cas. En effet, à l'heure où je finis de rédiger ce rapport, la question de la [licence](#) est encore débattue.

L'objectif du projet a été globalement atteint. La seule étape restante est la publication et l'analyse en profondeur des méthodes implémentées. La publication est le problème majeur auquel je fais actuellement face. En effet, aucun article de recherche sur le modèle [CTJ](#) n'a été publié, ainsi la publication d'une [bibliothèque](#) l'utilisant pose problème car le modèle pourrait être utilisé sans les crédits adéquats, voire pire, plagié. De plus, la [licence](#) devrait maintenant être une [licence](#) non-commerciale, par volonté de mon tuteur.

# Bibliographie

- [1] I. B. Mullaney, “Comparative triple judgment as an assessment methodology,” bai degree dissertation, Trinity College Dublin, the University of Dublin, Dublin, 2022.
- [2] A. Pollitt, “The method of adaptive comparative judgement,” *Assessment in Education : Principles, Policy & Practice*, vol. 19, no. 3, pp. 281–300, 2012.
- [3] L. L. Thurstone, “A law of comparative judgment.,” *Psychological review*, vol. 101, no. 2, p. 266, 1994.
- [4] D. Andrich, “Relationships between the thurstone and rasch approaches to item scaling,” *Applied Psychological Measurement*, vol. 2, no. 3, pp. 451–462, 1978.
- [5] A. Pollitt, “Comparative judgement for assessment,” *International Journal of Technology and Design Education*, vol. 22, 05 2011.
- [6] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” *Physica A : Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.
- [7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, New Jersey, USA : John Wiley & Sons Inc., 2nd ed., 2005.
- [8] R. W. Yeung, “A new outlook on shannon’s information measures,” *IEEE Transactions on Information Theory*, vol. 37, no. 03, pp. 466–474, 1991.