

Faculté des Sciences et Ingénierie - Sorbonne université

Master Informatique parcours - DAC



PLDAC : Projet Logiciel DAC

Rapport de Projet

XAI : Explications Contrastives Bi-factuelles

Réalisé par :

Faten Racha Said

Ahmed Abdelaziz Mokeddem

Yacine B. D. Chettab

Supervisé par :

Isabelle Bloch

Marie-Jeanne Lesot

Mai 2024

Table des matières

Introduction	1
1 Contexte	3
1.1 Emergence de l'XAI	3
1.2 Contributions de Tim Miller	4
1.3 Travaux connexes	4
2 Causes et explications contrastives bi-factuelles	6
2.1 Graphes causaux	6
2.2 Syntaxe et formalisme	7
2.3 Exemple : Une plateforme de candidature au master DAC	7
2.4 Causes et les explications	10
2.4.1 Causes non-contrastives	11
2.4.2 Causes contrastives bi-factuelles	12
2.4.3 Explications contrastives bi-factuelles	13
3 Approches algorithmiques	15
3.1 Structures de données	15
3.2 Approche naïve	15
3.3 Améliorations générales	17
3.3.1 Optimisation de la fonction AC2	17
3.3.2 Optimisation de la fonction BC1 et BC2	18
3.3.3 Optimisation de la fonction BC4	19
3.3.4 Optimisation de la fonction FindCauses	19
3.4 Amélioration spécifique à notre exemple : amélioration de la fonction AC2_3 . .	20
4 Expérimentations et Résultats	21
4.1 Résultats de la fonction FindCauses	21
4.2 Comparaison des temps d'exécution	22
4.2.1 AC2/AC3 vs AC2_3	22
4.2.2 BC1 vs BC1_2_beta	24
4.2.3 BC3/BC4 vs BC3_4	27

5 CausaLytics : le modèle généralisé	28
5.1 Description du modèle	28
5.2 CausaLytics : une application graphique	29
Conclusion	33
Bibliographie	34

Introduction

L'avènement de l'intelligence artificielle (IA) a révolutionné de nombreux domaines tels que la santé, le marketing et la finance. Cependant, avec cette révolution technologique, vient une complexité croissante des modèles d'IA qui sont, parfois, considérés comme des «boîtes noires», dont les décisions peuvent être difficiles à comprendre pour les utilisateurs.

Ce besoin croissant d'interprétabilité a mené à l'émergence de l'XAI, acronyme pour "eXplainable Artificial Intelligence" ou IA explicable en français, qui œuvre principalement à vérifier la cohérence interne des systèmes en fournissant une explication aux utilisateurs quant au processus de prise de décision du modèle.

De très nombreux systèmes ont été développés [1] [2], dont certains qui s'appuient sur le raisonnement par abduction [3]; que nous introduisons dans le premier chapitre. D'autres systèmes, en revanche, sont basés sur une approche symbolique qui s'appuient sur des logiques, graphe de causalité, règles et arbres de décision, qui les rend plus facilement explicables. C'est sur cette deuxième famille d'approches que repose notre projet. En effet, ce dernier est principalement basé sur l'étude [4] de Tim Miller et vise à construire un système capable de fournir des explications contrastives bi-factuelles pour une situation donnée.

Les explications contrastives bi-factuelles, qui étant donné la description d'un contexte, d'une décision P et d'une décision Q, apportent une réponse aux questions de la forme «Pourquoi la décision est-elle P dans un certain contexte et Q dans un autre?». Pour y répondre, Tim Miller introduit un certain nombre de propriétés à vérifier par le système afin de considérer une explication comme bi-factuelle. Ces propriétés reposent sur des règles de logique et de causalité, dont l'objectif est d'identifier les changements de valeurs de variables qui mènent à un changement de décision.

A travers ce projet, nous apportons une dimension pratique à la théorie énoncé par Tim Miller [4] en proposant un logiciel, qui prend en entrée la représentation informatique d'un contexte, la description d'une décision P et d'une décision Q, retourne les causes bi-factuelles. Nous présentons dans un premier temps une implémentations des propositions formelles, en reprenant fidèlement les instructions de l'article, puis nous analysons la complexité de cette solution. Nous proposons par la suite plusieurs optimisations en argumentant leur utilité à l'aide d'explications théoriques ainsi que des expérimentations pratiques. Ces étapes nous ont permis de concevoir et de développer un logiciel, enrichi d'une première interface graphique, capable de s'adapter à différents types de problèmes donnés pour fournir un ensemble de causes bi-factuelles.

Ce document décrit le travail réalisé pour la mise en œuvre de ce projet et est organisé comme suit :

- **Chapitre 1** : Présentation du contexte et des concepts connexes à notre étude.
- **Chapitre 2** : Description du formalisme de Tim Miller et proposition d'un exemple illustratif réaliste.
- **Chapitre 3** : Description des différentes solutions proposées, partant de l'approche naïve, à des approches plus optimisées/améliorées.
- **Chapitre 4** : Description du protocole expérimental et des résultats obtenus.
- **Chapitre 5** : Présentation de notre application et d'une manière de généraliser notre solution.

Nous concluons notre rapport par un résumé de notre contribution et quelques perspectives de développements futurs.

1.1 Emergence de l'XAI

L'eXplainable Artificial Intelligence (XAI) est un domaine d'application qui remonte à plus de trois décennies [4] [1] [2] ; on peut citer à titre d'exemple les travaux [5], [6] et [7]. Son apparition est notamment motivée par le manque de confiance des utilisateurs vis-à-vis des décisions prises par les modèles d'Intelligence Artificielle, parfois qualifiés de systèmes opaques ; comme le cas des réseaux de neurones profonds.

Nous retrouvons son utilisation dans différents sous-domaines de l'IA, par exemple pour justifier le comportement d'agents autonomes [8], le débogage des modèles d'apprentissage automatique [9], ou l'interprétation des décisions médicales [10].

Le domaine, bien qu'il soit tendance ces dernières décennies, est basé sur des principes qui existent depuis l'antiquité avec l'avènement du raisonnement par abduction prôné par certains philosophes de cette époque. Cette famille d'approche [3] vise à trouver la meilleure explication possible à partir d'un ensemble d'observations, et même de formuler ces hypothèses explicatives en présence de données incomplètes ou incertaines. Toute hypothèse peut être révisée ou rejetée en fonction de nouvelles informations.

Dans d'autres contextes, plusieurs travaux de recherche ont été proposés, certains largement répandus à l'instar de LIME [11] et SHAP [12]. La première approche propose d'expliquer les prédictions de modèles en ajustant localement un modèle interprétable autour du point d'intérêt dans l'espace de données, alors que la seconde est basée sur la théorie des jeux et fournit des explications additives pour les prédictions des modèles.

Bien que toutes ses théories soient intéressantes et pour certaines, notamment le raisonnement par abduction, particulièrement bien adaptée pour répondre au type de problème que nous cherchons à résoudre, ne sont pas au coeur de notre projet. En effet, nous nous sommes concentrés sur une famille d'approches qui est basée sur de la symbolique, qui de part son formalisme, intègre des modèles qui sont plus facilement explicables ; à l'instar des arbres de décisions.

Nous nous sommes plus précisément basé sur des principes de logique et de graphes causaux, comme défini par la théorie de Tim Miller ; que nous détaillons tout au long des prochains chapitres.

1.2 Contributions de Tim Miller

Tim Miller [13] reproche aux travaux en XAI de trop souvent reposer sur l'intuition des chercheurs de ce qui constitue une bonne explication, ce qui peut entraîner dans certains cas un manque de fiabilité dans les interprétations retournées par le système. Ainsi, il suggère de construire des modèles d'XAI en se basant sur des travaux déjà existants qui traitent du processus cognitif de prise de décision tels que des articles en philosophie et en psychologie cognitive, scientifique et sociale.

Il souligne quatre caractéristiques [13] :

- Les questions de type «Pourquoi» sont contrastives, c'est-à-dire elles opposent deux cas de figures différents.
- Les explications sont choisies de façon biaisées, c'est-à-dire que plusieurs facteurs peuvent influencer l'interprétation que nous donnons à certains événements.
- Les explications relèvent d'une dimension sociale, puisqu'elles servent à répondre à un besoin défini par l'être humain.
- Identifier des relations de cause à effet est plus important qu'émettre des hypothèses basées sur des probabilités.

Dans un article ultérieur [4], Tim Miller propose un modèle général pour traiter les explications contrastives, soit fournir des réponses à des questions qui mettent en contraste deux options. On peut distinguer deux familles de questions contrastives de type «Pourquoi?», qui pour un contexte donné et une description d'une décision P et d'une décision Q, on a :

- Pourquoi la décision P et pas Q?
- Pourquoi la décision est-elle P dans un certain contexte et Q dans un autre?

La première consiste à fournir une explication *contrefactuelle* et la seconde une explication *bi-factuelle*.

Ce modèle basé sur une structure causale, détaillé dans le chapitre 2, peut servir de cadre/référence pour des modèles d'Intelligence Artificielle tels que le apprentissage automatique, planning, apprentissage par renforcement, raisonnement à partir de cas, BDI agents, etc.

1.3 Travaux connexes

Il existe de nombreux articles sur les explications contrefactuelles, comme [14], [15] [16]. Le cas bi-factuelles est plus rarement traité. En effet, en reprenant les dires de Tim Miller, il n'existait pas de travaux antérieurs apportant une solution concrète à cette problématique avant la parution de l'article [4].

En revanche, après sa publication, on peut retrouver [17] qui reprennent entre autres cette notion d'explications bi-factuelles et montrent que plusieurs méthodes en XAI peuvent être

associées à des concepts populaires de comportement. Ce travail, bien qu'il souligne les modes de défaillance qui empêchent les méthodes actuelles de fournir de *bonne* explications, ne propose aucune implémentation des propositions formelles énoncées par Tim Miller.

Causes et explications contrastives bi-factuelles

Tim Miller définit les causes et les explications de manière formelle dans son article [4]. Nous allons reprendre les parties importantes dans le cadre de notre projet, centré sur la recherche des explications contrastives bi-factuelles, et les illustrer à travers un exemple simple mais suffisamment complexe de notre invention.

2.1 Graphes causaux

Un graphe causal est un diagramme utilisé pour modéliser les relations de cause à effet entre différentes variables ou événements d'un système. Ce concept trouve une application significative dans le domaine de l'intelligence artificielle, notamment dans les méthodes d'explication des résultats des algorithmes. Dans la figure 2.1, la variable C produit A et B, indépendamment l'une de l'autre, dans le scénario (a) et A produit C qui elle-même produit B dans le scénario (b).

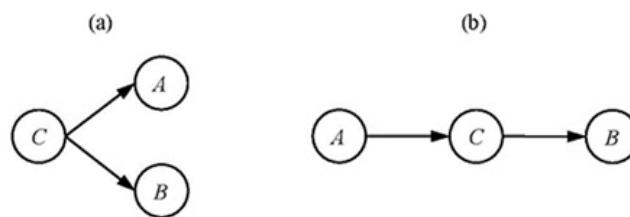


FIGURE 2.1 – Exemple de deux graphes causaux

On distingue dans un modèle causal deux types de variables : les variables exogènes et les variables endogènes. Les variables exogènes sont celles dont les valeurs sont déterminées par des facteurs externes au modèle lui-même. Elles correspondent, par exemple, à des données d'entrée ou des conditions environnementales. En revanche, les valeurs des variables endogènes sont déterminées par les relations avec d'autres variables du modèle, qu'elles soient exogènes ou endogènes.

La représentation graphique des relations causales entre les variables permet de visualiser de manière claire et concise la structure du système et les dépendances entre ses composants, cette clarté et cette concision étant conditionnées par la complexité du graphe. Cette modélisation graphique facilite l'analyse des effets des changements sur le système et contribue à la compréhension des mécanismes sous-jacents aux phénomènes observés.

2.2 Syntaxe et formalisme

Dans l'article [4], Tim Miller formalise des définitions des explications contrastives en se basant sur les modèles structurels de Halpern et Pearl [15]. Son approche repose sur des explications contrefactuelles, modélisées à l'aide d'équations structurelles.

Un modèle causal est défini sur deux ensembles de variables exogènes \vec{U} et endogènes \vec{V} et une fonction R telle que $R(X)$ est le domaine de toutes les valeurs possibles de la variable X . Étant donné une signature $S = (\vec{U}, \vec{V}, R)$, pour un ensemble de variables endogènes $\vec{X} \subseteq \vec{V}$ et $\vec{x} \in R(\vec{X})$, on définit :

- Un évènement primitif, noté $X = x$, signifie que la valeur x est affectée à la variable X .
- Formule causale de base, notée $[\vec{X} \leftarrow \vec{x}]\varphi$, qui peut être décomposée en : $[X_1 \leftarrow x_1, \dots, X_n \leftarrow x_n]\varphi$. Elle permet de lier φ (une combinaison booléenne d'évènements primitifs) avec un ensemble de variables $\vec{X} \in \vec{V}$ associées à leurs valeurs $\vec{x} \in R(\vec{X})$.
- Formule causale comme une combinaison booléenne de formules causales de base.
- Un modèle $M = (S, F)$ étant donné que F soit un ensemble de fonctions qui servent à déduire les variables endogènes à partir d'autres variables (modélisé par les arêtes du graphe causal associé).
- Un contexte \vec{u} est un vecteur des valeurs attribuées aux variables exogènes.
- La paire M, \vec{u} est appelée une situation.

L'objectif est de voir si φ change en changeant les valeurs de \vec{X} .

On note : $M, u \models [\vec{X} \leftarrow \vec{x}]\varphi$ un modèle M et un contexte u , avec les variables dans \vec{X} mises aux valeurs de \vec{x} , produit la décision φ .

2.3 Exemple : Une plateforme de candidature au master DAC

Nous proposons un exemple illustratif que nous utiliserons pour le reste de notre projet, notamment dans la définition des causes et des explications dans la suite du rapport. Cet exemple n'est ni trop complexe à comprendre, mais il n'est pas non plus trop simplifié au point qu'on trouve plus de cas particuliers intéressants, surtout que le graphe causal correspondant est à 4 niveaux. Le choix de cet exemple est fait de telle sorte que le lecteur suivra avec clarté les définitions, puisqu'il s'agit d'un concept avec lequel tout le monde est à l'aise : candidater à un master.

Le processus de traitement d'un dossier de candidature au master DAC est représenté dans ce graphe causal. Nous supposons que l'admission exige de bons résultats dans les cinq premiers semestres ainsi qu'une note supérieure à un seuil spécifique dans trois matières précises :

Logique, Bases de données et Probabilités et statistiques. Nous faisons l'hypothèse que les semestres où ces trois dernières matières sont enseignées ne sont pas connus, nous les considérons comme indépendantes des moyennes semestrielles.

Cet exemple peut-être formalisé de la façon suivante : le modèle $M = (S, F)$, tel que $S = (\vec{U}, \vec{V}, R)$, défini par :

- $\vec{U} = \{N_candidat\}$, avec $N_candidat$ un entier.
- $\vec{V} = \{S1, S2, S3, S4, S5, Logique, BDD, PS, Moyenne, Eligible, Décision\}$
- $R(v) = \{1, \dots, 20\}$ pour tout $v \in V \setminus \{Eligible, Décision\}$, $R(Eligible) = \{True, False\}$, $R(Décision) = \{A, L, R\}$ représentant respectivement : accepté, en liste d'attente ou rejeté.
- L'ensemble des fonctions F est défini comme $F = \{F_1, F_2, F_3, F_4, F_5, F_{Logique}, F_{BDD}, F_{PS}, F_{Moyenne}, F_{Eligible}, F_{Décision}\}$ où les fonctions F_i ne peuvent pas être explicitées, elles associent un $N_candidat$ à ses moyennes semestrielles. De même, F_{BDD} , F_{PS} , $F_{Logique}$ associent un $N_candidat$ à ses notes de ces matières. Pour les 3 dernières fonctions, nous proposons de considérer :

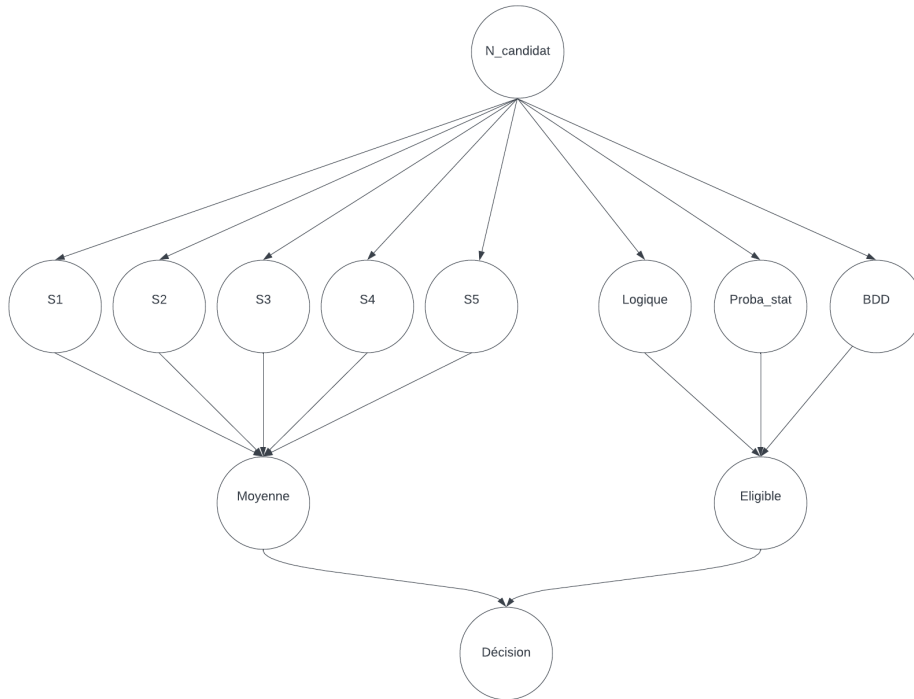


FIGURE 2.2 – Graphe causal de l'exemple proposé de plateforme inspiré de «MonMaster»

$$F_{Eligible} : R(Logique) \times R(BDD) \times R(PS) \rightarrow R(Eligible)$$

$$F_{Eligible}(n_l, n_b, n_p) = (n_l \geq 12) \wedge (n_b \geq 12) \wedge (n_p \geq 12)$$

$$F_{Moyenne} : R(S1) \times R(S2) \times R(S3) \times R(S4) \times R(S5) \rightarrow R(Moyenne)$$

$$F_{Moyenne}(n_{S1}, n_{S2}, n_{S3}, n_{S4}, n_{S5}) = \text{Floor} \left(\frac{\sum_{i=1}^5 n_{Si} \times coef_i}{\sum_{i=1}^5 coef_i} \right) \text{ où les coefficients sont tous mis à } 1$$

pour des raisons de simplicité.

$$F_{D\acute{e}cision} : R(Moyenne) \times R(Eligible) \rightarrow R(D\acute{e}cision)$$

$$F_{D\acute{e}cision}(m, e) = \begin{cases} A & \text{si } e \wedge (m \geq 14) \\ L & \text{si } e \wedge (12 \leq m < 14) \\ R & \text{sinon} \end{cases}$$

Pour chaque variable $i \in \vec{V} \setminus \{Moyenne, Eligible, D\acute{e}cision\}$, $F_i(n)$ indique la note du candidat n dans la mati\ere repr\esent\ee par la variable i . Cette structure S, F peut \^etre illustr\ee graphiquement par le graphe causal pr\esent\ee dans la figure 2.2. Le tableau 2.1 pr\esente cinq exemples d'un ensemble de donn\ees o\`u les coefficients de calcul de moyenne sont \`a 1.

$N_{candidat}$	$S1$	$S2$	$S3$	$S4$	$S5$	$Logique$	BDD	PS	$Moyenne$	D\acute{e}cision
0001	15	13	16	17	13	17	12	14	15	A
0002	11	5	8	12	10	12	14	16	10	R
0003	14	12	11	15	13	12	14	16	13	L
0004	16	15	15	18	15	0	9	15	16	R
0005	11	11	11	11	11	10	10	9	11	R
0006	0	1	0	0	0	13	13	13	1	R
0007	1	0	0	0	0	14	14	14	1	R

TABLE 2.1 – Donn\ees des candidats avec d\acute{e}cisions

Discr\etisation

Pour optimiser les temps de calculs (voir 3.2), nous proc\edons \`a une discr\etisation des variables Logique, Bases de donn\ees (BDD) et Probabilit\es et statistiques (PS), ainsi que de la moyenne, en fonction de leurs seuils respectifs : pour les variables Logique, BDD et PS, la nouvelle valeur est d\et\ermin\ee en la comparant \`a un seuil de 12. Quant \`a la moyenne, elle est discr\etis\ee en trois cat\egories : de 0 \`a 12, de 12 \`a 14 et de 14 \`a 20. Cette d\emarche nous sera b\en\efique ult\erieurement car elle r\eduit la cardinalit\ee du domaine de d\efinition de ces variables, ce qui r\eduit la complexit\ee de nos algorithmes.

$N_{candidat}$	$S1$	$S2$	$S3$	$S4$	$S5$	<i>Logique</i>	<i>BDD</i>	<i>PS</i>	<i>Moyenne</i>	Décision
0001	15	13	16	17	13	1	1	1	2	A
0002	11	5	8	12	10	1	1	1	0	R
0003	14	12	11	15	13	1	1	1	1	L
0004	16	15	15	18	15	0	0	1	2	R
0005	11	11	11	11	11	0	0	0	0	R
0006	0	1	0	0	0	1	1	1	0	R
0007	1	0	0	0	0	1	1	1	0	R

TABLE 2.2 – Données discrétisées des candidats avec décisions

Le but des explications est de guider l'utilisateur vers un justificatif de la décision attribuée par le modèle d'affectation. Cet exemple est utilisé tout au long du rapport pour illustrer des notions et concepts en considérant comme question sous-jacente : «pourquoi telle décision pour chaque étudiant qui n'a pas accès aux détails du modèle causal?»

2.4 Causes et les explications

On a distingué deux types principaux de questions : les non-contrastives et les contrastives. La distinction entre les questions *Pourquoi* non-contrastives et contrastives réside dans leur approche vis-à-vis de l'explication d'un événement. Alors que les questions non-contrastives se contentent d'expliquer un événement sans référence à d'autres alternatives, les questions contrastives mettent ces dernières en lumière. Les questions contrastives comparent l'événement considéré avec une autre solution possible, soit en demandant pourquoi un certain événement a eu lieu plutôt qu'un autre (*foil*), soit en cherchant à expliquer pourquoi un événement s'est produit dans un contexte donné alors qu'un autre événement (*surrogate*) s'est produit dans un autre contexte similaire. Les premières sont appelées "*counterfactual explananda*", tandis que les secondes sont appelées des explications "*bi-factual explananda*". Ces dernières utilisent le *surrogate* comme point de référence pour le contraster avec l'événement.

La distinction entre les explications contrastives contrefactuelles et bi-factuelles se fait principalement sur la base de la nature d'autres solutions considérées. Les explications bi-factuelles mettent en parallèle deux événements réels qui se sont produits dans des contextes différents. Elles répondent à des questions du type «Pourquoi $u \models \phi$ mais $u' \models \psi$?», où u et u' représentent deux contextes distincts alors que les explications contrefactuelles se reposent sur un événement et une alternative dans le même contexte, autrement dit, elles répondent à des questions de type «Pourquoi $u \models \phi$ plutôt que $u \models \psi$?».

Notre objectif maintenant est de comprendre comment les causes non-contrastives sont d'abord définies formellement par Tim Miller, car elles ouvrent la voie à la définition formelle des causes contrastives bi-factuelles que nous mettrons en pratique lors de leur génération.

2.4.1 Causes non-contrastives

Les causes non-contrastives ont une importance quand il s'agit des explications contrastives bi-factuelles car elles figurent dans la définition formelle.

- *Cause* : Une cause est une conjonction d'événements primitifs $[\vec{X} \leftarrow \vec{x}]$.
- *Sufficient Cause* : Une *sufficient cause*, ou cause suffisante, est une conjonction d'événements primitifs telle que le changement de la valeur de certaines variables entraîne la non-occurrence de l'événement.
- *Actual Cause* : Une *actual cause* est une version minimale d'une *sufficient cause* ; c'est-à-dire qu'elle ne contient aucune conjonction inutile à la prise de décision (la production de l'événement ϕ).
- *Partial Cause* : Une *partial cause*, ou cause partielle, est un sous-ensemble (inclus ou égal) de conjonctions d'une *actual cause*.

Tim Miller [4] propose de définir formellement la notion de la *actual cause* de la façon suivante :

La conjonction $[\vec{X} \leftarrow \vec{x}]$ est une *actual cause* de ϕ dans \vec{u} si :

1. **AC1** : $M, \vec{u} \models \phi$ & $\vec{X} \leftarrow \vec{x}$. Le contexte produit la décision et la cause.
2. **AC2** : Il existe un ensemble de variables $\vec{W} \subseteq \vec{V}$ et une configuration \vec{x}' des variables \vec{X} telles que si $M, \vec{u} \models \vec{W} = \vec{w}$, alors $M, \vec{u} \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}] \neg \phi$. Il doit exister un ensemble de variables \vec{W} tel que si les valeurs de ces variables sont maintenues inchangées et \vec{X} prennent d'autres valeurs que \vec{x} , alors l'événement ne se produit pas.
3. **AC3** : \vec{X} est minimal ; aucun sous-ensemble de \vec{X} ne satisfait simultanément AC1 et AC2.

Explication des AC_i : AC1 garantit que la cause et l'événement se sont produits. AC2 impose que, pour que $\vec{X} = \vec{x}$ soit une cause suffisante, le changement des valeurs de toutes ces variables, en gardant d'autres variables \vec{W} inchangées, ne produise plus l'événement. Une convention que nous avons adoptée est de prendre $\vec{W} = \vec{V} \setminus \vec{X}$. AC3 assure que la cause suffisante est bien aussi une *actual cause*.

Exemples : Revenons à notre exemple de la plateforme de candidature de la section 2.3. Observons les cas où $\vec{X} = \vec{x}$ est une *actual cause* :

1. **M, $\mathbf{u}_{0001} \models (\text{Décision} = \mathbf{A})$:**

Examinons le candidat 0001 avec $\vec{X} = \{S4\}, \vec{x} = \{17\}$

AC1 : $\{S4 = 17\}$, Décision=A sont vraies.

AC2 : Avec $\vec{W} = \vec{V} \setminus \vec{X}$, nous obtiendrions une décision différente si nous avons changé la valeur de S4 à 0.

AC3 : $|\vec{X}| = 1$. Il est évident que \vec{X} est de cardinalité minimale.

2. $u_{0004} \models (\text{Décision} = R)$:

Examinons le candidat 0004 avec $\vec{X} = \{\text{Logique}, BDD\}$ et $\vec{x} = \{0, 0\}$

$AC1$: $\{\text{Logique} = 0, BDD = 0\}$, $\text{Décision} = R$ sont vraies.

$AC2$: En prenant $\vec{W} = \vec{V} \setminus \vec{X}$, nous obtiendrions une décision différente si nous avons changé la valeur de Logique et BDD à la fois à 1.

$AC3$: $|\vec{X}| = 2$. Nous remarquons que changer une unique variable ne suffit pas à changer la valeur de la décision, donc \vec{X} est minimal.

Observons des cas où ce n'est pas une actual cause :

1. Du moment que la ligne existe dans le tableau et qu'elle soit cohérente, $AC1$ est vérifiée. Un exemple de cas d'échec : des notes des semestres contradictoires ou manquantes. Une façon comment les notes peuvent être contradictoire est si une variable, comme *Moyenne*, n'est pas égal à la valeur calculée avec $F_{Moyenne}$.
2. $M, u_{00005} \models (\text{Décision} = R)$ et $\vec{X} = \{\text{Logique}, BDD, PS\}$, $\vec{x} = \{0, 0, 0\}$ et $\vec{W} = \vec{V} \setminus \vec{X}$. Le changement de toutes les valeurs de \vec{X} ne suffit pas pour changer la décision car la moyenne reste toujours dans la catégorie 0.
3. $M, u_{0004} \models (\text{Décision} = R)$ et $\vec{X} = \{\text{Logique}, BDD, S1\}$, $\vec{x} = \{0, 0, 16\}$ et $\vec{W} = \vec{V} \setminus \vec{X}$. $AC1$ est satisfaite par \vec{X} , $\text{Décision} = R$. $AC2$ est satisfaite car les nouvelles valeurs données à \vec{X} : $\vec{x}' = \{1, 1, 14\}$ changent la valeur de la décision à «A», mais \vec{X} n'est pas minimal car nous avons pu trouver un ensemble plus petit qui les satisfait au début de l'exemple, $\vec{X} = \{\text{Logique}, BDD\}$.

2.4.2 Causes contrastives bi-factuelles

Une cause contrastive bi-factuelle est une paire d'évènements $\langle \vec{X} = \vec{x}, \vec{X} = \vec{y} \rangle$ qui produisent simultanément les deux évènements distincts, ϕ et ψ , dans leurs contextes u et u' . Elle est formellement définie par les 4 conditions suivantes :

1. **BC1** : $\vec{X} = \vec{x}$ est une cause partielle de ϕ dans le contexte u .
2. **BC2** : $\vec{X} = \vec{y}$ est une cause partielle de ψ dans le contexte u' .
3. **BC3** : $(\vec{X} = \vec{x}) \cap (\vec{X} = \vec{y}) = \emptyset$; il n'y a pas d'évènements communs entre les deux ensembles.
4. **BC4** : \vec{X} est maximal ; il n'existe pas de sur-ensemble de \vec{X} qui satisfait les critères BC1, BC2 et BC3.

Explication des BC_i : BC1 et BC2 imposent que $\vec{X} = \vec{x}$ et $\vec{X} = \vec{y}$ soient respectivement des causes partielles de l'évènement et du *surrogate*. Une cause partielle peut être aussi une *actual cause*. Autrement dit, si l'ensemble n'est pas déjà une *actual cause* et en ajoutant des variables supplémentaires (de manière judicieuse), nous devrions être en mesure d'identifier une *actual*

cause. BC3 exclut les cas où une variable prend la même valeur dans deux contextes différents, car si un événement est commun dans ces deux contextes, il n'aide pas à expliquer le contraste entre les décisions attribuées. De ce fait, nous pouvons éviter tous les cas où une variable de la cause à étudier a la même valeur dans les deux contextes u et u' . BC4 représente une condition de maximalité de la cardinalité de \vec{X} vérifiant les trois premières conditions.

Exemples :

Comparons les deux étudiants 0001 et 0004 : le but est de savoir pourquoi le 1e est accepté alors que le 2d est rejeté. Les définitions BCi ci-dessus conduisent à l'explication $\langle \{Logique = 1\}, \{Logique = 0\} \rangle$ c-à-d que l'étudiant 0001 a été accepté car il avait validé logique contrairement à l'étudiant 0004.

$\mathbf{M}, \mathbf{u}_{0001} \models (\text{Décision} = \mathbf{A})$

$\mathbf{M}, \mathbf{u}_{0004} \models (\text{Décision} = \mathbf{R})$

$\langle \{Logique = 1\}, \{Logique = 0\} \rangle$ est une cause contrastive bi-factuelle associée aux événements : $(\text{Décision} = \mathbf{A})$, $(\text{Décision} = \mathbf{R})$ respectivement.

$BC1 : \langle Logique = 1 \rangle$ est une cause partielle et une *actual cause* de $(\text{Décision} = \mathbf{A})$.

$BC2 : \langle Logique = 0 \rangle$ est une cause partielle de $(\text{Décision} = \mathbf{R})$, mais elle n'est pas une *actual cause*. Le changement de Logique à 1 ne suffit pas à faire changer la décision : il faudrait rajouter BDD pour obtenir une *actual cause*.

$BC3 : \langle Logique = 1 \rangle \cap \langle Logique = 0 \rangle = \emptyset$

$BC4 : L'$ ensemble \vec{X} est maximal. Le choix des variables est fait par la différence entre la ligne de l'étudiant 0001 et celle de l'étudiant 0004. Nous vérifions, par exemple, si \vec{X} valait $\{Logique, BDD\}$, il aurait vérifié BC1, BC2 et BC3 mais $\{Logique = 1, BDD = 1\}$ n'est pas une cause partielle de $(\text{Décision} = \mathbf{A})$ car ce n'est pas un sous-ensemble d'une *actual cause*. L'ensemble \vec{X} des *actual causes* = $\{\{Logique\}, \{BDD\}\}$.

Observons des cas où $\langle \vec{X} = \vec{x}, \vec{X} = \vec{y} \rangle$ n'est pas une cause bi-factuelle :

1. Pour $\vec{X} = \{Logique, PS\} : \langle Logique = 0, PS = 1 \rangle$ n'est pas une cause partielle de $(\text{Décision} = \mathbf{R})$, car \vec{X} est un sur-ensemble de la *actual cause* $Logique = 1$ du premier événement.
2. Pour $\vec{X} = \{Logique, PS, BDD\} : \langle Logique = 1, BDD = 1, PS = 1 \rangle \cap \langle Logique = 0, BDD = 10, PS = 1 \rangle \neq \emptyset$ ce qui contredit la BC3.

2.4.3 Explications contrastives bi-factuelles

Une explication est une justification concise d'un processus de prise de décision, fournissant uniquement les informations essentielles pour établir la causalité. Elle est définie comme étant un ensemble d'informations qui, une fois découvertes, peuvent servir de cause à un événement donné, mais qui sont initialement inconnues. Par conséquent, une explication est relative à

un état épistémique, c'est-à-dire à l'ensemble des connaissances ou croyances d'un agent à un moment donné.

L'épistémique, dans ce contexte, se réfère à l'état de connaissances ou de croyances d'un individu. Ainsi, une explication est considérée comme étant épistémiquement pertinente lorsqu'elle fournit des informations qui vont au-delà de l'état de connaissance actuel de l'agent, ce qui lui permet de mieux comprendre la cause de l'événement en question. En d'autres termes, une explication pertinente épistémiquement doit mettre à jour l'état épistémique de l'agent, lui permettant ainsi de comprendre la causalité derrière l'événement.

Les explications contrastives bi-factuelles sont considérées équivalentes aux causes contrastives bi-factuelles selon la théorie présentée par Tim Miller. Pour plus de détails, voir des définitions des EXi et $BE'i$ dans l'article de Tim Miller et le **théorème 7** qui prouve une telle équivalence. [4]. Nous retenons que : $\mathbf{BCi} \equiv \mathbf{BEi} \equiv \mathbf{BE'i}$

Approches algorithmiques

Dans ce chapitre, nous présenterons les algorithmes initiaux ainsi que les stratégies d'optimisation que nous avons mises en œuvre pour les améliorer. Notre objectif final est de créer un programme en Python qui, étant donné (F, S) , deux formules ϕ et ψ , renvoie $\vec{X} = \vec{x}$ et $\vec{X} = \vec{y}$ comme illustré dans le chapitre précédent. Cette démarche découle de notre compréhension de l'équivalence entre les explications contrastives bi-factuelles et les causes contrastives bi-factuelles, ce qui nous offre la possibilité de les mettre en pratique. Ces processus reposent sur des parcours combinatoires, ce qui soulève des considérations de complexité algorithmique que nous aborderons également.

3.1 Structures de données

Dans le cadre de notre projet¹, nous avons défini deux éléments essentiels pour répondre à notre objectif. Tout d'abord, les entités sujettes à étude constituent notre base de données. Elles sont organisées dans une structure tabulaire de type Pandas DataFrame, où chaque ligne représente une entité indépendante identifiée par une clé, telle que le numéro de candidat N dans l'exemple 2.3. Les variables du modèle sont représentées par les colonnes de ce tableau.

Ensuite, les causes à étudier seront représentées sous forme de dictionnaires, où les clés correspondent aux noms des variables \vec{X} , identiques dans les deux éléments de la cause bi-factuelle, et les valeurs correspondent aux valeurs de ces variables \vec{x} et \vec{y} . Dans les algorithmes, nous utiliserons la notation X et V au lieu de \vec{X} et \vec{V} respectivement.

La classe *Data* est responsable de la gestion des données, y compris les opérations d'accès aux valeurs et de manipulation des données. Elle constitue une interface essentielle pour la manipulation des données dans notre programme.

En parallèle, la classe *Modèle* dépasse le simple rôle de gestion des données. Elle contient également les fonctions et les méthodes que nous développerons dans ce chapitre. Cette classe joue un rôle central dans la modélisation des données et dans l'exécution des algorithmes que nous présenterons par la suite.

3.2 Approche naïve

L'approche naïve consiste simplement à vérifier chaque propriété comme décrite par l'article de Tim Miller [4]. Dans cette approche, un objet de la classe *Modèle* a les méthodes suivantes :

1. <https://github.com/chettabyacine/M1-S2-PLDAC>

- **AC1**(*id*, *X*, *y*) - *test de consistance* : vérifie si la ligne de *id* existe, et que les valeurs de variables de *X* et la décision sont correspondent à l'objet *Data* du modèle.
- **AC2**(*id*, *X*, *y*) - *test de la suffisance (sufficient cause)* : essaie toutes les **combinaisons de valeurs des variables** de *X* , pour trouver comment changer la valeur de décision, en changeant toutes les variables de *X* à la fois. Si la décision change pour une combinaison, la fonction renvoie *True*. Si aucune combinaison n'est trouvée, elle renvoie *False*.
- **AC3**(*id*, *X*, *y*) - *test de minimalité de la actual cause* : génère tous les sous-ensembles de \vec{X} et teste si l'un d'autres eux vérifie à la fois **AC1** et **AC2**. Si c'est le cas, elle renvoie *False*.
AC1, **AC2** et **AC3** servent à vérifier si *X* correspond à une *actual cause*.
- **BC12**(*id*, *X1*, *y1*)/**BC12**(*id*, *X2*, *y2*) - *test de la causalité de chacun des deux évènements* : génère dans un premier temps les sur-ensembles de \vec{X} en ajoutant incrémentalement une variable de $\vec{V} \setminus (\vec{X})$ (les variables de *V* sauf celles qui appartiennent à *X* aussi), puis elle teste si ce sur-ensemble est une *actual cause* en faisant appel aux *AC_i*.
- **BC3**((*id1*, *id2*),(*X1*, *X2*), (*y1*, *y2*)) - *test de différence* : vérifie que *X1* et *X2* ont des valeurs toutes différentes.
- **BC4**((*id1*, *id2*),(*X1*, *X2*), (*y1*, *y2*)) - *test de maximalité de la cause contrastive bi-factuelle* : génère les sur-ensembles de \vec{X} jusqu'à \vec{V} et teste si *BC₁*, *BC₂* et *BC₃* sont satisfaites pour au moins un sur-ensemble, et dans ce cas elle renvoie *False*. Sinon aucun contre-exemple n'est trouvé, elle renvoie *True*.
- **BC**((*id1*, *id2*),(*X1*, *X2*), (*y1*, *y2*)) est la conjonction des 4 *BC_i*.
- **FindCause**(*id1*, *id2*) génère les sous-ensembles de *V_{diff}*, l'ensemble des variables pour lesquelles les deux *id* ont des valeurs différentes. Parcourir chaque sous-ensemble pour chacun d'eux, faire appel aux fonctions *BC*, qui vérifient si ce sous-ensemble est une cause bi-factuelle. Dans le cas d'une cause bi-factuelle, elle la sauvegarder.

L'avantage de cette approche est qu'elle est directe et simple à implémenter, une fois que les définitions mathématiques sont bien acquises. L'inconvénient est sa complexité.

Complexité théorique :

1. **AC1** : Elle est de complexité liée à la cardinalité de *X* qui peut aller jusqu'à $|V|$, avec l'hypothèse que la recherche d'une ligne par son *id* et la vérification de la décision sont de complexité $\mathbf{o}(1)$. **Complexité_{AC1}** = $\mathbf{o}(|X|)$
2. **AC2** : Parcourir toutes les combinaisons de valeurs des variables dans *X*. Cela dépend de la taille des *R(v)*, le domaine de définition de chaque variable endogène *v*. La complexité est donc liée à ce produit cartésien des domaines.
Complexité_{AC2} = $\mathbf{o}(\prod_{v \in V} |R(v)|)$.

Cette complexité motive l'amélioration proposée dans l'exemple 2.3, qui consiste à réduire le domaine de recherche par un facteur de $\frac{21^4}{2^3 \times 3} \approx 8103$

3. **AC3** : Tester tous les sous-ensembles des X , donc des combinaisons de cardinalité de $|X|$ à 1.

$$\text{Complexité}_{AC3} = 2^{|X|} \times (\text{Complexité}_{AC1} + \text{Complexité}_{AC2})$$

$$\text{Complexité}_{AC3} = 2^{|X|} \times (o(|X|) + o(\prod_{v \in V} |R(v)|)).$$

4. **BC1/BC2** : Faire augmenter la cardinalité de X pour passer d'une cause partielle à une *actual cause*. Il faut parcourir, dans le pire des cas, les combinaisons de cardinalité de $|X|$ à $|V|$.

$$\text{Complexité}_{BC12} = 2^{|V|-|X|} \times (\text{Complexité}_{AC1} + \text{Complexité}_{AC2} + \text{Complexité}_{AC3}).$$

$$= 2^{|V|-|X|} \times (o(|X|) + o(\prod_{v \in V} |R(v)|) + 2^{|X|} \times (o(\prod_{v \in V} |R(v)|) + o(|X|))).$$

$$= 2^{|V|-|X|} \times (2^{|X|} + 1) \times (o(|X|) + o(\prod_{v \in V} |R(v)|)).$$

$$\text{Complexité}_{BC12} = (2^{|V|-|X|} + 2^{|V|}) \times (o(|X|) + o(\prod_{v \in V} |R(v)|)).$$

5. **BC3** - : Vérifie que aucune valeur est communes entre les deux causes à tester. Son temps d'exécution dépendra de la cardinalité de X , qui peut, dans le pire des cas, être égale à $|V|$.

$$\text{Complexité}_{BC3} = o(|X|)$$

6. **BC4** - : Tester toutes les combinaisons de $|X|$ jusqu'à $|V|$. Dans chaque étape, il faut faire appel à **BC12** deux fois et **BC3**.

$$\text{Complexité}_{BC4} = 2^{|V|-|X|} \times (\text{Complexité}_{BC3} + 2 \times \text{Complexité}_{BC12}).$$

$$= 2^{|V|-|X|} \times (o(|X|) + (2^{|V|-|X|+1} + 2^{|V|+1}) \times (o(|X|) + o(\prod_{v \in V} |R(v)|))).$$

7. **FindCause** : Elle génère les sous-ensembles de V , et voir s'ils vérifient **BC1** et **BC2**. Ainsi, la complexité de la version naïve est :

$$\text{Complexité}_{FindCauses} = 2^{|V|} \times (2 \times \text{Complexité}_{BC12} + \text{Complexité}_{BC3} + \text{Complexité}_{BC4}).$$

3.3 Améliorations générales

Dans le but de réduire le traitement de ces fonctions, nous proposons dans la suite de ce chapitre des approches d'optimisations relatives aux méthodes précédemment présentées et dont la complexité a été détaillée.

3.3.1 Optimisation de la fonction AC2

La première implémentation de la fonction **AC2**, que nous avons proposé, décrite dans la section précédente, reflète fidèlement la définition de l'auteur [4]. Dans cette section, nous proposons une nouvelle fonction basée sur un principe simple : explorer les sous-ensembles de

X dans un ordre croissant de leur cardinalité. L'objectif est de réduire le temps d'exécution en traitant d'abord les cas « faciles », c'est-à-dire ceux qui sont moins complexes en termes de combinaisons. Cette nouvelle fonction satisfait à la fois la condition AC2 et AC3, énoncées par Tim Miller. Sa complexité est dans le pire cas égale à celle de l'approche naïve, car la fonction optimisée permet de déterminer un résultat à *False* plus rapidement en cas d'échec.

Le pseudo-code de la fonction est présenté dans l'annexe.

Notons que, tout au long du projet, nous avons utilisé une fonction prédéfinie de python pour la génération des combinaisons de clés de X , ce qui rend l'exécution de la tâche très rapide même sur un ensemble de cardinalité importante. La génération des valeurs possibles pour chaque combinaison peut, en revanche, prendre un temps conséquent dans certains cas.

3.3.2 Optimisation de la fonction BC1 et BC2

Nous avons vu dans l'approche naïve décrite dans la section 3.2 que la fonction BC1/BC2 génère directement les sur-ensembles de \vec{X} en ajoutant de façon incrémentale une variable de $\vec{V} \setminus \vec{X}$ et teste si ce sur-ensemble est une *actual cause* en faisant appel aux ACi .

Le problème avec cette approche est que rien ne nous garanti que X lui même n'est pas une *actual cause* ou même un *sufficient cause* (sur-ensemble de *actual cause*). De plus, nous ne gardons aucune trace des exécutions passées des ACi , alors que cela peut nous servir étant donné qu'un appel à BC1 peut engendrer plusieurs appels aux ACi .

Ainsi, pour cette optimisation deux idées clés sont à retenir. La première part du principe qu'avant de générer les sur-ensembles de X et voir s'ils vérifient les propriétés des *actual causes*, il faut d'abord s'assurer que X n'est lui-même pas un sur-ensemble d'une *actual cause*. La seconde consiste à introduire une structure de données, *var_notAC*, afin de conserver les tests (les ensembles de variables) non concluants ; c'est-à-dire déjà évalués à *False* par les fonctions ACi . Cela a pour objectif d'éviter de perdre du temps à les ré-évaluer lors des prochains appels de cette fonction.

Notons que l'ordre des appels des fonctions permettant de vérifier les propriétés des ACi est important. Tester, par exemple, si $AC1$ est vérifiée avant de passer à $AC2$ peut constituer un gain considérable en temps d'exécution du programme. Nous avons veillé à optimiser ces appels tout au long de notre implémentation.

Le pseudo-code de la fonction est donné dans l'annexe.

En matière de complexité, notre contribution se révèle particulièrement utile lorsque l'utilisateur introduit un X déjà trop étendu. Elle permet d'éviter de générer des sur-ensembles inutiles, lesquels sont d'autant plus coûteux en temps de calcul, étant donné que la combinatoire croît de façon exponentielle par rapport à la cardinalité de l'ensemble, ce qui entraîne un temps d'exécution plus conséquent.

Aussi, nous verrons aussi l'avantage d'avoir introduit à ce niveau une telle structure de données pour mémoriser les cas d'échecs, lors de l'optimisation de la fonction BC4 décrite dans

la section 3.3.3 .

3.3.3 Optimisation de la fonction BC4

L'approche naïve de BC4 décrite dans la section 3.2 consistait à tester toutes les combinaisons de $|X|$ jusqu'à $|V|$ et renvoie *False* s'il existe un sur-ensemble de X qui vérifie toutes les propriétés définies par BC1/BC2 et BC3.

Le problème avec cette approche est qu'elle exécute tout de manière séquentielle et indépendante. En effet, le système doit vérifier une à une toutes les combinaisons des variables jusqu'à $|V|$ deux fois (pour vérifier à la fois P et Q) sans prendre en compte le fait que la BC3 devrait à priori être vérifiée; ceci peut engendrer un nombre de combinaisons très important.

Cette nouvelle approche inclut à la fois la vérification des deux propriétés BC3 et BC4. En effet, nous sommes partis du principe il n'était pas nécessaire de générer tous les sur-ensembles de X car cela peut entraîner des combinaisons trop importantes, mais qu'il suffisait de générer tous les sur-ensembles de X inclus dans V_{max} , l'ensemble de variables produites par l'intersection de $X1$ et $X2$.

De plus, comme BC4 cherche à trouver un sur-ensemble de X de taille maximale vérifiant les BCi, notre programme traite les combinaisons dans l'ordre décroissant de leur cardinalité.

Le pseudo-code de la fonction est donné dans l'annexe.

Il est important de souligner que la conception de notre solution fait que le traitement en premier des sur-ensembles ne pose pas de problème, puisque *BC1_2_beta* optimise particulièrement bien les cas dont la valeur de retour est *False*. En effet, comme nous l'avons expliqué dans la section 3.3.2, cette fonction commence par vérifier si X n'est pas déjà trop étendu avant de générer et parcourir ses sous-ensembles. Elle introduit de plus la structure de données *var_notAC*, présentée dans le titre précédent, pour éviter de ré-évaluer un même ensemble de variables plusieurs fois.

3.3.4 Optimisation de la fonction FindCauses

Cette optimisation est inspirée de la fonction *BC3_4*. En effet, pour trouver les causes bi-factuelles pour deux entités représentées par deux lignes du tableau de données, nous prenons l'ensemble des variables pour lesquelles ces deux lignes ont des valeurs différentes V_{diff} , duquel nous générons l'ensemble des parties que nous parcourrons dans l'ordre décroissant de leurs cardinalités. Ainsi, parcourir uniquement les sous-ensembles de V_{diff} garantit que BC3 est bien vérifiée. Il est clair que toutes les variables d'une cause bi-factuelle sont toujours incluses dans V_{diff} pour que BC3 soit vérifiée. De ce fait, si la cause bi-factuelle existe, elle est explorée dans la version optimisée. De plus, le parcours dans l'ordre décroissant garantit que BC4 est vérifiée. En effet, si un sous-ensemble de V_{diff} est une cause bi-factuelle, alors tout sous-ensemble de cette cause ne peut l'être et est donc éliminé du parcours. Pour les propriétés BC1 et BC2,

nous exploitons les versions optimisées décrites dans la section 3.3.2.

Le pseudo-code de la fonction est donné dans l'annexe.

La complexité de cette implementation est exponentielle en nombres de combinaisons possibles. La complexité de génération de l'ensemble des parties de \mathbf{V}_{diff} étant négligeable et dans la pratique exécutée instantanément par une machine, la complexité de FindCause consiste en la complexité de BC12 (car nous faisons appel à BC1 et BC2 dans la boucle interne de parcours des éléments de `all_combinaisons_var` et cet appel est effectué $|\text{all_combinaisons_var}|$ fois. Nous savons que le cardinal de l'ensemble des parties d'un ensemble fini est de $2^{|\mathbf{V}_{\text{diff}}|}$. Ainsi, la complexité de FindCauses est $\text{Complexité}_{\text{FindCauses}} = 2^{|\mathbf{V}_{\text{diff}}|} \times (2 \times \text{Complexité}_{\text{BC12}})$.

3.4 Amélioration spécifique à notre exemple : amélioration de la fonction AC2_3

Dans un premier temps, il convient de noter que cette fonctionnalité est propre à l'exemple de «MonMaster» que nous avons proposé dans la section 2.3.

Considérons par exemple un étudiant avec une décision 'R' et $S1 = 8$, nous pouvons être sûrs que l'algorithme ne trouvera pas une valeur $v \in [0, 8[$ de $S1$ qui mènera à un changement de décision, il est donc inefficace pour l'algorithme de faire sa recherche sur cette plage de valeurs.

Nous exploitons dans cette partie les caractéristiques de ce cas d'application pour proposer une version améliorée de la fonction AC2_3 dans la section 3.3.1 à l'utilisateur, l'idée étant de renvoyer une solution qui garantit un changement minimal de valeurs de variables ; il est important de distinguer la différence entre un changement minimal du nombre de variables (déjà vérifié par AC3) et un changement minimal des valeurs de variable.

En reprenant l'exemple de «MonMaster», nous intégrons le principe suivant, qui exploite une relation d'ordre/monotonie entre les décisions, à l'algorithme présenté dans la section 3.3.1 :

- Évaluer la décision prise par le modèle : $\{\text{'A'}, \text{'L'}, \text{'R'}\}$
- Si la décision est 'A', alors diminuer progressivement la note de l'étudiant jusqu'à trouver la valeur qui change la décision, ou jusqu'à ce que la note atteigne 0 (ou le seuil fixé).
- Si la décision est 'R', alors augmenter progressivement la note de l'étudiant jusqu'à trouver la valeur qui change la décision, ou jusqu'à ce que la note atteigne 20 (ou le seuil fixé).
- Si la décision est 'L', alors augmenter la note de l'étudiant jusqu'à trouver la valeur qui change la décision. Si celle-ci reste inchangée et que la note a atteint 20, alors diminuer la note jusqu'à trouver la valeur qui change la décision, ou jusqu'à ce que la note atteigne 0 (ou le seuil fixé).

Notons que cette amélioration concerne uniquement les notes des semestres (variables S_i) et participe à l'optimisation de la fonction AC2_3 puisqu'elle évite à l'algorithme de rechercher des combinaisons inutiles de valeurs de variables.

Expérimentations et Résultats

Dans ce chapitre nous affichons dans un premier temps les résultats renvoyés par notre système sur un certain nombre d'exemples choisis. Dans un second temps, nous effectuons une étude expérimentale empirique qui consiste à comparer le temps d'exécution des fonctions décrites dans le chapitre précédent avant et après leur optimisation.

4.1 Résultats de la fonction FindCauses

Nous avons testé la fonction FindCauses sur les exemples décrits dans le tableau 4.1 et obtenu les résultats et temps d'exécution correspondants comme indiqué dans le tableau 4.1.

Exemples	N_candidats	Résultat
1	(0006, 0007)	[{'S1', 'S2'}]
2	(0001, 0002)	[{'S5'}, {'S4'}, {'S3'}, {'S2'}, {'S1'}]
3	(0002, 0004)	[{'S5'}, {'S4'}, {'S3'}, {'S2'}, {'S1'}, {'BDD'}, {'Logique'}]

TABLE 4.1 – Résultats de la fonction FindCauses

Exemples	Temps (ms)
1	177
2	10^3
3	$1014 \cdot 10^3$

TABLE 4.2 – Temps d'executions de la fonction FindCauses

Le résultat est une liste contenant l'ensemble des variables qui constituent une cause contrastive bi-factuelle pour les paires de candidats. Ainsi, pour l'exemple 1, les deux étudiants 6 et 7, les deux ayant été refusés, la réponse à la question Pourquoi les deux étudiants ont été refusés ? est la cause bifactuelle formée par les variables S1 et S2. En effet, il s'agit d'un cas extrême, certes, mais qui illustre le cas où la cause bifactuelle est le V_diff . C'est pourquoi, nous remarquons que le temps d'exécution est très petit puisque la vérification des causes bifactuelles débute par V_diff qui, en l'occurrence, est une cause bifactuelle.

En ce qui concerne l'exemple 2, les deux étudiants 1 et 2, le premier étant Accepté, et le deuxième étant refusé, les réponses à la question Pourquoi l'étudiant 1 a été accepté et l'étudiant 2 a été refusé ? sont les cinq causes bifactuelles. Pour arriver aux singletons, il a fallu parcourir toutes les combinaisons de cardinalité supérieure avec un V_diff de longueur 5 (les variables

Si). Enfin, l'exemple 3 illustre un cas similaire au précédant mais avec un `V_diff` incluant les variables catégorielles Logique et BDD. Le temps d'exécution est alors le même que le précédent auquel on rajoute le coût des combinaisons supplémentaires explorées.

4.2 Comparaison des temps d'exécution

Nous définissons la mesure *Accélération* comme suit : $\frac{\text{Temps_exécution_version_optimisée}}{\text{Temps_exécution_version_naïve}}$.

Cette métrique permet d'évaluer l'amélioration obtenue grâce à l'optimisation apportée à notre implémentation.

4.2.1 AC2/AC3 vs AC2_3

En reprenant le cas d'application défini dans la section 2.3, nous exécutons les fonctions AC2/AC3 et AC2_3 sur les exemples donnés dans le tableau 4.3 et la moyenne de leurs temps d'exécution est montrée dans le tableau 4.10.

Nous avons choisi ces exemples en particuliers pour illustrer et expliquer différents cas de figures auxquels nous pouvons être confrontés lors de tests sur un plus grand jeu de données.

Exemple	N_candidat	Ensemble des variables X	Décision
1	0001	{ 'Logique' : '1', 'BDD' : '1' }	'A'
2	0004	{ 'Logique' : 0, 'BDD' : 0, 'S1' : 16, 'S2' : 15 }	'R'
3	0001	{ 'S1' : 15 }	'A'
4	0002	{ 'S1' : 11, 'S5' : 10 }	'R'
5	0004	{ 'S5' : 11, 'BDD' : 0 }	'R'
6	0006	{ 'S1' : '0', 'S2' : '1', 'S3' : '0' }	'R'

TABLE 4.3 – Exemples choisis pour la comparaison entre AC2/AC3 et AC2_3

Notons que l'ordre des exemple n'a pas d'importance car dans ces expérimentations les exemples sont exécutés de façon indépendante.

Fonction	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5	Exemple 6
AC2/AC3	0.78	9.28	0.2	2.98	1.56	51.24
AC2_3	0.52	1.14	0.26	2.02	1.02	51.76

TABLE 4.4 – Comparaison des temps d'exécution (ms) moyens entre AC2/AC3 et AC2_3

Fonction	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5	Exemple 6
AC2/AC3	2.0	3.79	4.86	3.19	2.58	11.81
AC2_3	1.34	2.14	1.6	2.44	1.19	12.43

TABLE 4.5 – Écarts types des temps d'exécution entre AC2/AC3 et AC2_3

Mesure	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5	Exemple 6
Accélération	0.67	0.12	1.3	0.68	0.65	1.01

TABLE 4.6 – Comparaison des score d'accélération entre AC2/AC3 et AC2_3

Nous présentons ci-dessous le graphe associé aux tableaux 4.10 et 4.11.

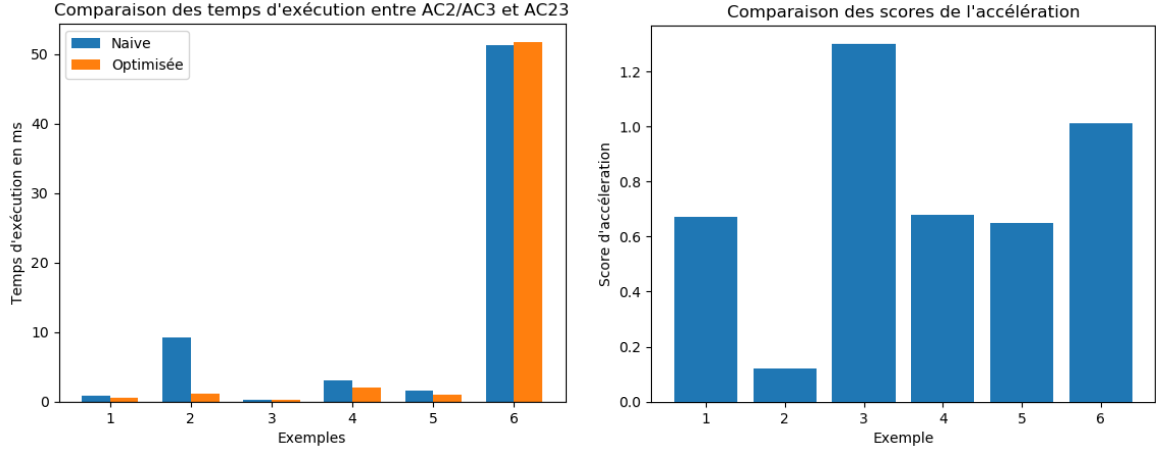


FIGURE 4.1 – Comparaison des temps d'exécution (ms) moyens et score d'accélération entre AC2/AC3 et AC2_3

Nous expliquons les résultats obtenus pour chaque exemple par ce qui suit :

- **Exemple 1** : le résultat renvoyé est *False*. L'impact de la version optimisée apparaît car le changement d'une seule variable suffit à changer de décision. Bien que la différence ne soit pas flagrante sur le graphique, elle apparaît dans le tableau 4.10 comme étant deux fois plus petite que la version naïve.
- **Exemple 2** : le résultat renvoyé est *False* car la *actual cause* dans ce cas est $\{\text{'Logique'} : 0, \text{'BDD'} : 0\}$. Nous avons ajouté S1 et S2 à X afin de montrer la différence entre AC2/AC3 qui commence par modifier une à une toutes les variables de X et AC2_3 qui procède de façon incrémentale.
- **Exemple 3** : le résultat renvoyé est *True*. X est déjà de longueur minimale, de ce fait, nous ne pouvons pas observer l'effet de l'optimisation étant donné que X n'a pas de sous-ensembles à part lui même. Les deux fonctions sont équivalentes.
- **Exemple 4** : le résultat renvoyé est *True*. X est une *actual cause*, ce qui signifie que les deux fonctions comparées exécutent le même nombre d'étapes, mais dans un ordre différent. De ce fait, les deux fonctions sont théoriquement équivalentes. Pour ce qui est des temps d'exécutions, nous supposons que la différence est négligeable.
- **Exemple 5** : le résultat renvoyé est *False*. Le changement de toutes les variables ne mène pas à un changement de décision. Ainsi, les deux fonctions sont théoriquement équivalentes. Pour ce qui est des temps d'exécutions, nous supposons que la différence est négligeable.

- **Exemple 6** : le résultat renvoyé est *True*. X est une *actual cause*, ce qui signifie que les deux fonctions comparées exécuteront le même nombre d'étapes, mais dans un ordre différent. De ce fait, les deux fonctions sont équivalentes. Notons que le temps d'exécution pour cet exemple est particulièrement élevé car le nombre de variables est plus élevé que celui des exemples précédents, de plus, l'intervalle de test des valeurs de chaque variable est maximal (selon la définition de notre cas d'application).

Il faut souligner que, pour l'exemple 5, le temps d'exécution n'est pas aussi élevé que pour l'exemple 4, mais pas aussi bas que pour l'exemple 1, malgré un nombre identique de variables testées. Cela est dû à l'intervalle de valeurs que prend chaque variable. En effet, le modèle teste toutes les valeurs possibles pour chaque combinaison de variable (en excluant la combinaison initiale de valeurs) ; soit $19 + 19 + 19 * 19 = 399$ valeurs dans l'exemple 4, $19 + 1 + (20 * 2 - 1) = 59$ valeurs dans l'exemple 5, $1 + 1 + (2 * 2 - 1) = 5$ valeurs dans l'exemple 1. Pour rappel, les S_i sont des entiers définis sur une plage de valeurs de 0 à 20, et les variables "Logique" et "BDD" sont des variables binaires. Ainsi, pour illustrer le processus sur l'exemple 5, l'algorithme teste toutes les valeurs possibles sur $\{S_5\}$ en excluant la valeur 11, puis teste le complémentaire sur $\{BDD\}$, et enfin teste toutes les combinaisons de valeurs sur $\{S_5, BDD\}$ en excluant la combinaison (11, 0).

Notons que pour certains exemples, bien que la version optimisée et naïve soient théoriquement équivalentes, nous avons observé une différence quant au temps d'exécution. Cela peut être justifié par la forte variance entre les durées enregistrées sur l'ensemble des exécutions. Pour rappel, les tests ont été répétés plusieurs fois afin de garder une valeur moyenne des temps d'exécution.

Nous pouvons conclure que l'optimisation proposée n'est jamais pire que la version classique. On peut observer l'impact de notre optimisation sur les exemples 1 et 2 : plus la cardinalité de X est grande, plus l'optimisation est efficace. La mesure d'accélération pour ces deux exemples prend respectivement pour valeur 0.47 et 0.07.

4.2.2 BC1 vs BC1_2_beta

En reprenant le cas d'application défini dans la section 2.3, nous exécutons la fonction BC avec les deux variantes de la fonction BC1 (une version naïve et une optimisée). Les exemples choisis sont donnés dans le tableau 4.7, et la moyenne de leurs temps d'exécution est montrée dans le tableau 4.9.

Nous avons choisi ces exemples en particuliers pour illustrer et expliquer différents cas de figures auxquels nous pouvons être confrontés lors de tests sur un plus grand jeu de données.

Exemple	N_candidats	X du 1e candidat	X du 2d candidat
1	(0001, 0003)	{'S1' : '15', 'S2' : '13'}	{'S1' : '14', 'S2' : '12'}
2	(0004, 0005)	{'Logique' : '0', 'BDD' : '0'}	{'Logique' : '0', 'BDD' : '0'}
3	(0001, 0002)	{'S2' : '13'}	{'S2' : '5'}
4	(0001, 0003)	{'S5' : '13'}	{'S5' : '13'}
5	(0003, 0004)	{'Logique' : '1', 'BDD' : '1', 'S2' : '12'}	{'Logique' : '0', 'BDD' : '0', 'S2' : '15'}

TABLE 4.7 – Exemples choisis pour la comparaison entre BC1 et BC1_2_beta

Exemples	Décisions
1	('A', 'L')
2	('R', 'R')
3	('A', 'R')
4	('A', 'L')
5	('L', 'R')

TABLE 4.8 – Décisions des exemples choisis pour la comparaison entre BC1 et BC1_2_beta

Notons que l'ordre des exemple n'a pas d'importance car dans ces expérimentations les exemples sont exécutés de façon indépendante, la structure de données est réinitialisée entre deux exécutions d'exemple.

Fonctions	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
BC1	15.5	100.0	1.0	<1	6.0
BC1_2_beta	<1	<1	1.0	<1	6.0

TABLE 4.9 – Comparaison des temps d'exécution (s) entre BC1 et BC1_2_beta

Fonctions	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
BC1	0.0	1.5	0.0	0.0	0.0
BC1_2_beta	0.0	0.0	0.5	0.0	0.0

TABLE 4.10 – Écarts types des temps d'exécution entre BC1 et BC1_2_beta

Mesure	Exemple 1	Exemple 2	Exemple 3	Exemple 4	Exemple 5
Accélération	0.01	0.0	1.0	1.0	1.0

TABLE 4.11 – Comparaison des score d'accélération entre BC1 et BC1_2_beta

Nous présentons ci-dessous le graphe associé au tableau 4.9.

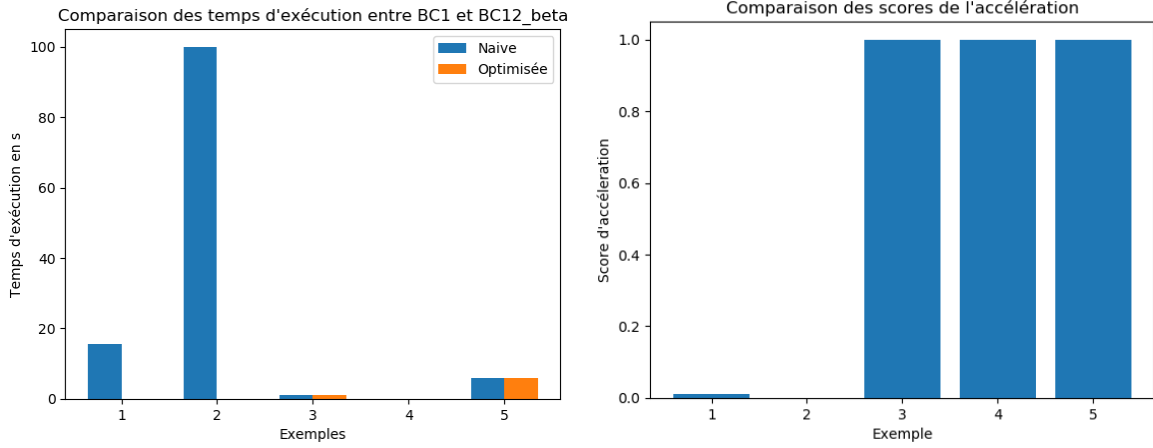


FIGURE 4.2 – Comparaison des temps d'exécution (ms) et score d'accélération entre BC1 et BC1_2_beta

- **Exemple 1** : le résultat renvoyé est *False*. La version optimisée se démarque positivement étant donné que l'ensemble X n'est pas minimal ; prendre $S1$ ou $S2$ aurait suffi à changer de décision.
- **Exemple 2** : le résultat renvoyé est *False* car l'intersection entre les valeurs de variables des deux candidats est vide. Dans cet exemple, la version optimisée permet de détecter que c'est une *actual cause* du candidat 0004 sans avoir à générer tous les sur-ensembles de X , alors que la version naïve génère et teste les sur-ensembles pour les 2 candidats.
- **Exemple 3** : le résultat renvoyé est *True*. Les deux fonctions sont équivalentes car $|X| = 1$ donc X est, de façon triviale, une cause partielle.
- **Exemple 4** : le résultat renvoyé est *False* car l'intersection entre les valeurs de variables des deux candidats n'est pas vide. Les deux fonctions sont équivalentes car $|X| = 1$ donc X est, de façon triviale, une cause partielle.
- **Exemple 5** : le résultat renvoyé est *False* car X n'est pas une cause partielle (le changement de valeur d'une seule variable suffit pour que le candidat 0003 change de décision). Le temps d'exécution des deux versions est égal. Les deux fonctions ne sont pas significativement différentes dans cet exemple car même si la version naïve commence par générer les sous-ensembles de X , ceux-ci passent par AC3 qui les décompose et renvoie *False* à la première erreur trouvée.

Nous pouvons remarquer que l'optimisation a un impact différent sur les exemples 4 et 2, pourtant le système renvoie *False* dans les deux cas et pour la même raison (l'intersection entre les valeurs des variables des deux candidats n'est pas vide). Nous expliquons cela par le fait que l'exemple 4 est un cas particulier où $|X| = 1$ ce qui fait que X est, de façon triviale, une cause partielle.

En conclusion, les deux exemples 1 et 2 montrent l'impact de notre optimisation. La métrique d'accélération pour ces deux exemples est respectivement de 0.05 et 0.009. Ainsi, la version

optimisée n'est jamais pire que la version classique.

4.2.3 BC3/BC4 vs BC3_4

L'approche optimisée que nous proposons améliore grandement les performances de notre solution étant donné que la version naïve de BC4 sur notre cas d'application ne peut pas être exécutée par notre ordinateur. En effet, lorsque nous testons cette fonction sur le couple de candidats (0001, 0004), celle-ci teste les conditions des BCi sur tous les sous-ensembles de X , mais échoue (mémoire insuffisante) lorsqu'elle atteint la cardinalité maximale, c'est-à-dire $|X| = |V|$; le nombre des combinaisons devient trop élevé. La fonction BC3_4, quant à elle, s'exécute en un temps raisonnable, comme nous avons pu le voir sur les exemples présentés dans la section 4.2.2.

CausaLytics : le modèle généralisé

Nous avons réalisé des expérimentations sur le modèle lié à notre exemple 2.3. Cependant, la majeure partie du code est conçue pour être généralisable à n'importe quel nombre de variables. Nous présentons un modèle simple à trois niveaux de causalité, illustré par la figure 5.1, qui peut être utilisé au lieu d'un autre modèle d'un graphe de causalité plus complexe et plus profond. Nous l'avons nommé Causalytics.

5.1 Description du modèle

Le modèle Causalytics est pour but d'être utilisable dans le plus nombre possible de scénarios. Il conçu pour être abstrait, personnalisable et extensible, correspondant à un graphe causal à trois niveaux : un premier niveau pour les variables exogènes, un deuxième niveau composé de d variables endogènes x_i , et enfin une dernière variable endogène y représentant la décision ou la classe dans le troisième niveau. Les couches intermédiaires ne sont pas visibles pour l'utilisateur, ce qui signifie qu'il n'a pas accès aux configurations telles que la moyenne, l'éligibilité, les seuils et les coefficients, comme illustré dans notre exemple décrit dans la section 2.3. Cette conception s'inspire de la transitivité des relations de cause à effet ; en d'autres termes, dans un graphe causal, si A produit B et B produit C , alors A produit également C . Dans une extension future, nous pourrions enchaîner ces modèles simples en reliant la sortie du premier à l'entrée du second.

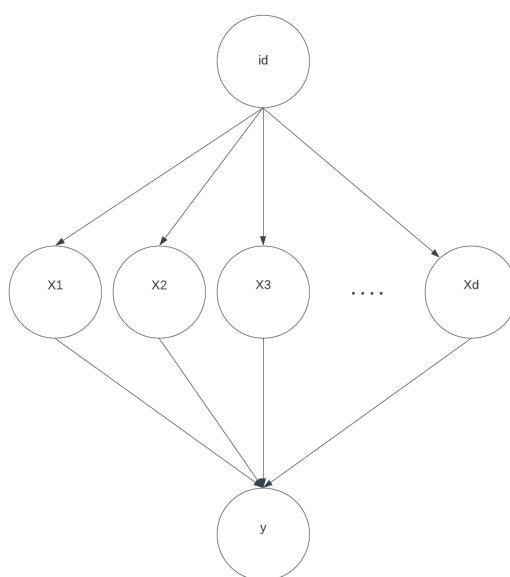


FIGURE 5.1 – Le graphe causal du modèle général

Le modèle est conçu pour des classifications complexes et peut accepter des valeurs réelles, booléennes ou de tout autre type. La fonction de classification peut être explicite ou un classifieur probabiliste, et elle peut être transmise en tant qu'hyper-paramètre.

Le problème de recherche de causes avec ce modèle revient à déterminer la ou les combinaisons de variables x_i qui satisfont la définition des causes bi-factuelles 2.4.2.

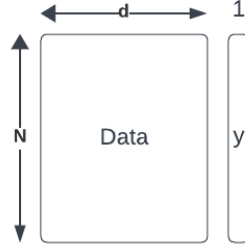


FIGURE 5.2 – La représentation matricielle du modèle généralisé

Une façon d'implémenter ce modèle est d'organiser les données sous forme matricielle, soit une matrice X où chaque ligne représente une entité (comme un candidat dans notre exemple 2.3) et chaque colonne représentant une variable exogène (à l'exception de la variable de décision). y est un vecteur colonne qui représente la classe de chaque entité, ayant autant de lignes que X comme expliqué dans la figure 5.2.

Le modèle génère les explications contrastives bi-factuelles à partir de deux identifiants (deux numéros de lignes de la matrice X) en faisant appel aux fonctions détaillées dans le chapitre 4.

5.2 CausaLytics : une application graphique

Notre application, décrite par le prototype illustré dans les figures , propose à l'utilisateur de configurer son modèle généralisé en définissant les variables (avec leurs types et valeurs par défaut), le nombre de données à générer et la méthode de classification. Une fois les données prêtes, l'utilisateur est invité à entrer deux identifiants, puis l'application fait appel aux fonctions développées dans le chapitre 4 pour générer les explications contrastives bi-factuelles. L'utilisateur a également la possibilité d'importer un fichier de données (de format *.csv*) et il aura la possibilité de modifier le type et le domaine de définition des variables. Une fois qu'on a toutes les données, qu'on a configuré les variables et classifieur, en entrant les deux identifiants, le résultat affiché est une liste de toutes les causes contrastives bi-factuelles comme illustré dans la figure 5.4 .

Le développement de l'application repose sur l'utilisation de la bibliothèque *Thinker*. Nous structurons nos fonctions de recherche de causes dans une classe appelée *Modèle*. Cette classe sera instanciée au moment de la configuration du modèle, incluant la définition des variables,

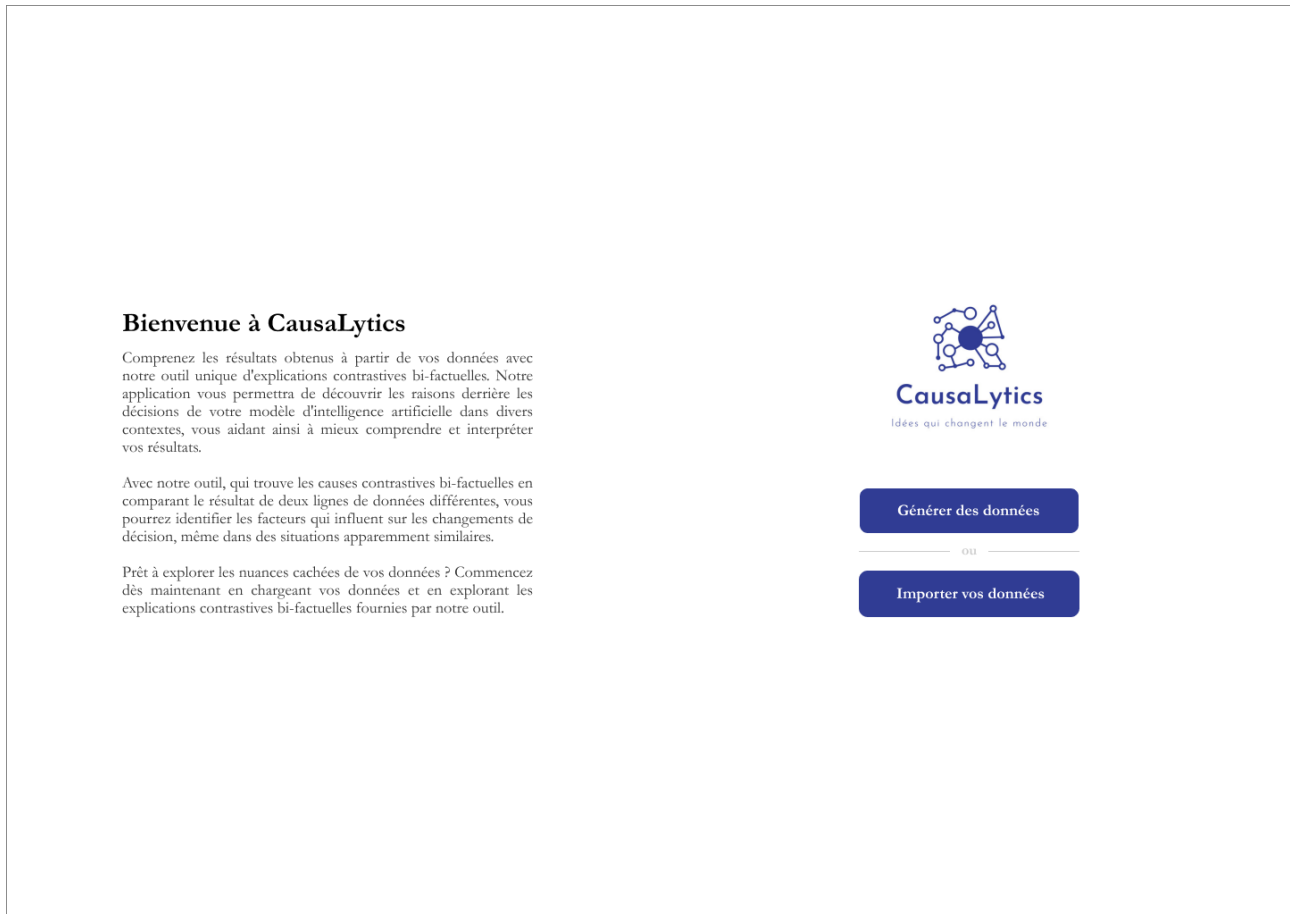



FIGURE 5.3 – La page principale

des données, ainsi que la fonction de classification. Cette approche permet une gestion efficace des opérations de génération d'explications contrastives bi-factuelles.

Nous proposons un ensemble de fonctions de classification usuelles, telles que des opérations logiques pour les variables booléennes, des opérations arithmétiques et statistiques pour les variables réelles et entières. De plus, nous avons développé une fonction de classification spécifique qui implémente la règle considérée pour les dossiers de candidature au master DAC, comme illustré dans la section 2.3.



Comprendre votre classifieur

Idees qui changent le monde

Nombre de données à générer

Classifieur

Nom

Type

Valeur min

Valeur max

	S1	S2	S3	S4	S5	Logique	...	y
0001	15	13	16	17	13	1		A
0002	11	5	8	12	10	1		R
0003	14	12	11	15	13	1		L
0004	16	15	15	18	15	0		R
0005	11	11	11	11	11	0		R


ID1 ID2

Nom	Type	Min	Max
S1	Entier	0	20
S2	Entier	0	20
S3	Entier	0	20
S4	Entier	0	20
S5	Entier	0	20
Logique	Booléen		

Les 5 causes trouvées:

- {S1}
- {S2}
- {S3}
- {S4}
- {S5}

FIGURE 5.4 – Tester son classifieur sur des données artificielles



Comprendre votre base de données

Idees qui changent le monde

Classifieur MonMaster

Nom S1

Type Réel

Valeur min 0

Valeur max 20

Pas 0.1

[Mettre à jour S1](#)

	S1	S2	S3	S4	S5	Logique	...	y
0001	15	13	16	17	13	1		A
0002	11	5	8	12	10	1		R
0003	14	12	11	15	13	1		L
0004	16	15	15	18	15	0		R
0005	11	11	11	11	11	0		R

ID1 0001 **ID2** 0002

[Trouver les causes contrastives bi-factuelles](#)

Nom	Type	Min	Max
S1	Réel	0	20
S2	Entier	0	20
S3	Entier	0	20
S4	Entier	0	20
S5	Entier	0	20
Logique	Booléen		

Les 5 causes trouvées:

- {S1}
- {S2}
- {S3}
- {S4}
- {S5}

FIGURE 5.5 – Importer sa base de données

Conclusion

L'étude sur les explications contrastives bi-factuelles en intelligence artificielle (XAI) a mis fin à l'importance de répondre aux interrogations des utilisateurs concernant les résultats obtenus par les méthodes et algorithmes d'IA.. Les explications contrastives bi-factuelles permettent d'éclaircir des différences de décisions entre deux modèles pour des cas similaires, offrant ainsi une compréhension plus approfondie des mécanismes sous-jacents. Ce projet, mené dans le cadre du PLDAC 2023-2024, s'inscrit dans la continuité des travaux de T. Miller, qui combinent la logique et les graphes de causalité pour identifier les causes contrastives en effectuant des vérifications combinatoires. Nous avons réussi à implémenter des algorithmes et à les optimiser ensuite pour vérifier si un ensemble de variables constitue une cause contrastive bi-factuelle. Nous avons de plus conçu un algorithme pour les lister toutes. En outre, nous avons proposé un modèle généralisé, CausaLytics, qui peut être utilisé dans des différents scénarios. En plus, nous le présentons sous forme d'une interface graphique conviviale.

La généralisation des solutions élaborées et la prise en compte des limites identifiées dans ce projet constitueront des axes essentiels pour favoriser l'adoption et l'impact à long terme de cette approche en XAI. Nous envisageons également le développement d'autres versions de la solution proposée, avec des optimisations plus poussées et une plus grande variété de fonctionnalités pour les utilisateurs, en combinant les graphes de causalité pour obtenir des graphes plus complexes. De plus, nous aspirons à permettre l'interrogation des causes bi-factuelles à n'importe quel niveau d'un graphe complexe, offrant ainsi une analyse plus détaillée et approfondie des relations de causalité.

Annexe

Nous affichons dans cette partie les différents pseudo-codes des fonctions détaillées dans ce document.

Algorithm 1 AC2_3

Entrée : id, X, decision, dataset

Sortie : *True/False*

- Définir la plage de valeurs des variables du dataset.
- Générer tous les sous-ensembles de combinaisons possibles des clés de X (trié dans un ordre croissant des cardinalités).
- Pour chaque sous-ensemble de combinaison :
 - Générer toutes les valeurs possibles pour les clés de ce sous ensemble.
 - S'il existe une combinaison de valeurs qui mène à un changement de décision, renvoyer *False*.
- Pour chaque clé de X :
 - Générer toutes les valeurs possible (en excluant la combinaison identique à X).
 - S'il existe une combinaison de valeurs qui mène à un changement de décision, renvoyer *True*, sinon renvoyer *False*.

Fin

Algorithm 2 BC1_2_beta

Entrée : id, X, decision, dataset, var_notAC

Sortie : *True/False*, var_notAC

— Vérifier que X est cohérente avec le dataset :

si not AC1 **alors renvoyer** (False, var_notAC)

— S'il existe un sous-ensemble de X qui est une *actual cause*, X ne peut pas être une *partial cause*.

si AC2 and not AC3 **alors renvoyer** (False, var_notAC)

— Si X est une *actual cause*, X est aussi une *partial cause*.

si AC2 and AC3 **alors renvoyer** (True, var_notAC)

— Générer les sur-ensembles de X.

— Exclure les sur-ensembles de variables contenus dans var_notAC.

— S'il existe un sur-ensemble de X qui est une *actual cause*, renvoyer (True, var_notAC), sinon ajouter cette nouvelle combinaison de variables à la structure de données var_notAC.

Fin

Algorithm 3 BC3_4

Entrée : id, X, decision, dataset

Sortie : *True/False*

— Générer les sous-ensembles possibles de V_{max} qui vérifient que $X \subseteq V_{max}$.

— Exclure les sous-ensembles inclus dans var_notAC

— Trier les sous-ensembles dans l'ordre décroissant de leur cardinalité.

— Pour chaque sous-ensemble de combinaison :

 — Vérifier si cet ensemble est une *partial cause* en appliquant le processus décrit dans 5.2, auquel cas, renvoyer *True* si sa cardinalité est égale à celle de X.

— Si ni X ni aucun de ses sur-ensembles ne vérifient les *BCi* alors renvoyer *False*.

Fin

Algorithm 4 FindCause

Entrée : id1, id2

Sortie : Liste des causes bi-factuelles (liste de tuples en python)

- Pour colonne dans les colonnes de la base :
 - Si $\text{base}[\text{colonne}][\text{id1}] \neq \text{base}[\text{colonne}][\text{id2}]$:
 - Ajouter colonne à \mathbf{V}_{diff}
- **all_combinaisons_var** <- L'ensemble des parties \mathbf{V}_{diff}
- Pour chaque combinaison "item" dans **all_combinaisons_var** dans un ordre décroissant de leur cardinalité :
 - Si le sous-ensemble courant "item" vérifie la *BC1* pour *id1* et la *BC2* pour *id2*, alors c'est une cause bi-factuelle qu'on rajoutera au résultat final.

Fin

Bibliographie

- [1] Feiyu XU et al., « Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges », *in* : sept. 2019, p. 563-574, ISBN : 978-3-030-32235-9, DOI : 10.1007/978-3-030-32236-6_51.
- [2] Waddah SAEED et Christian OMLIN, « Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities », *in* : *Knowledge-Based Systems* 263 (2023), p. 110273, DOI : <https://doi.org/10.1016/j.knosys.2023.110273>, URL : <https://www.sciencedirect.com/science/article/pii/S0950705123000230>.
- [3] Robert R. HOFFMAN, William J. CLANCEY et Shane T. MUELLER, *Explaining AI as an Exploratory Process: The Peircean Abduction Model*, 2020, arXiv : 2009.14795 [cs.AI].
- [4] Tim MILLER, « Contrastive explanation: a structural-model approach », *in* : *The Knowledge Engineering Review* 36 (2021), ISSN : 1469-8005, DOI : 10.1017/S0269888921000102, URL : <http://dx.doi.org/10.1017/S0269888921000102>.
- [5] Bruce BUCHANAN et Edward SHORTLIFFE, *Rule-based Expert System – The MYCIN Experiments of the Stanford Heuristic Programming Project*, SERBIULA (sistema Librum 2.0), Addison-Wesley, 1984.
- [6] Balakrishnan CHANDRASEKARAN, M.C. TANNER et J.R. JOSEPHSON, « Explaining control strategies in problem solving », *in* : *IEEE Expert* 4.1 (1989), p. 9-15, DOI : 10.1109/64.21896.
- [7] William R. SWARTOUT et Johanna D. MOORE, « Explanation in Second Generation Expert Systems », *in* : *Second Generation Expert Systems*, sous la dir. de Jean-Marc DAVID, Jean-Paul KRIVINE et Reid SIMMONS, Springer (Berlin Heidelberg), 1993, p. 543-585.
- [8] Bradley HAYES et Julie A. SHAH, « Improving Robot Controller Transparency Through Autonomous Policy Explanation », *in* : ACM/IEEE International Conference on Human-Robot Interaction, Association for Computing Machinery, 2017, p. 303-312, DOI : 10.1145/2909824.3020233.
- [9] Todd KULEZA et al., « Principles of Explanatory Debugging to Personalize Interactive Machine Learning », *in* : (2015), Proceedings of the 20th International Conference on Intelligent User Interfaces March, p. 126-137, DOI : 10.1145/2678025.2701399.
- [10] John FOX et al., « Argumentation-Based Inference and Decision Making—A Medical Perspective », *in* : *IEEE Intelligent Systems* 22 (2007), p. 34-41, DOI : 10.1109/MIS.2007.102.

-
- [11] Marco Tulio RIBEIRO, Sameer SINGH et Carlos GUESTRIN, « “Why Should I Trust You?”: Explaining the Predictions of Any Classifier », *in* : *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), p. 9-10, URL : <https://api.semanticscholar.org/CorpusID:13029170>.
 - [12] Scott M. LUNDBERG et Su-In LEE, « A Unified Approach to Interpreting Model Predictions », *in* : *Neural Information Processing Systems*, 2017, p. 8-9, URL : <https://api.semanticscholar.org/CorpusID:21889700>.
 - [13] Tim MILLER, « Explanation in artificial intelligence: Insights from the social sciences », *in* : *Artificial Intelligence* 267 (2019), p. 1-38, DOI : <https://doi.org/10.1016/j.artint.2018.07.007>, URL : <https://www.sciencedirect.com/science/article/pii/S0004370218305988>.
 - [14] Peter LIPTON, « Contrastive Explanation », *in* : *Royal Institute of Philosophy Supplement* 27 (1990), p. 247-266, DOI : 10.1017/S1358246100005130.
 - [15] Joseph Y. HALPERN et Judea PEARL, « Causes and Explanations: A Structural-Model Approach. Part I: Causes », *in* : *British Journal for the Philosophy of Science* 56.4 (2005), p. 843-887, DOI : 10.1093/bjps/axi147.
 - [16] Ramaravind Kommiya MOTHILAL, Amit SHARMA et Chenhao TAN, « Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations », *in* : *CoRR* abs/1905.07697 (2019), arXiv : 1905.07697, URL : <http://arxiv.org/abs/1905.07697>.
 - [17] Alon JACOVI et al., « Diagnosing AI Explanation Methods with Folk Concepts of Behavior », *in* : *CoRR* abs/2201.11239 (2022), arXiv : 2201.11239, URL : <https://arxiv.org/abs/2201.11239>.