République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
École nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Informatique

Option : Systèmes Informatiques et Logiciels

Détection et réparation des anomalies dans les données et les règles de qualité

Réalisé par :
MEFLAH Wided

Encadré par :
Dr. ABDELLAOUI Sabrina

Promotion : 2022/2023

Dédicace

“

À ma merveilleuse mère, Tout ce que je réalise dans la vie, je le dois à vous. Chacun de mes succès est une délicate écho de votre amour inconditionnel et de votre dévouement. Je suis éternellement reconnaissante pour tout ce que vous avez fait pour moi, Votre amour et votre soutien m'ont façonnée, et je vous aime infiniment pour cela. Votre présence est le plus grand trésor de ma vie.

À ma sœur adorée Amira, ton rire est mon rayon de soleil quotidien.Je t'aime énormément.

je dédie ce rapport en signe de reconnaissance pour tout ce que vous avez fait pour moi.

Merci.

”

- Wided

Remerciements

Je tiens tout d'abord à remercier ALLAH le tout puissant pour m'avoir donné le courage et la patience pour bien mener ce travail.

Je tiens à remercier tout particulièrement mon encadrante, Mme Abdellaoui Sabrina, pour l'aide compétente qu'elle m'a apportée, pour sa patience et son encouragement.

Je remercie tous mes enseignants de l'ESI pour leurs efforts continus et pour m'avoir donné les moyens et les connaissances nécessaires pour évoluer dans le monde informatique, où j'ai pu exploiter toutes ces connaissances.

Que les membres du jury trouvent, ici, l'expression de mes sincères remerciements pour l'honneur qu'ils me font en prenant le temps de lire et d'évaluer ce travail. Merci à vous tous.

Merci à vous tous.

Résumé

La mauvaise qualité des données est un problème sérieux et coûteux. Les règles de qualité ont été initialement conçues pour assurer la qualité des données. Tout sous ensemble qui n'est pas conforme aux règles est considéré comme étant une violation.

La majorité des approches de réparation modifient les données afin de les faire correspondre aux règles de qualité. [Geerts et al. 2013, Fan et al. 2014, W. Chu et al. 2013, Lucie Berti-Equille et al. 2019].Cependant, les règles de qualité peuvent évoluer avec le temps à mesure que les règles d'application ou commerciales changent, que les données sont intégrées avec de nouvelles sources de données, ou que la sémantique sous-jacente des données évolue.

Dans des situations similaires, lorsqu'il y a des inconsistances, il devient difficile de déterminer clairement s'il faut réparer les données, ajuster les contraintes, ou prendre des mesures pour les deux à la fois.

Dans la littérature, il existe peu d'études qui suggèrent des solutions pour réparer les anomalies dans les règles de qualité [Golab et al. 2008].De plus, ces études partent du présupposé que les données sont initialement propres.

Moins d'approches considèrent que la source d'une violation peut être aussi bien dans les données que dans les règles de qualité. [Volkovs et al. 2014, Beskales et al. 2013].

Nous présentons une solution visant premièrement à détecter si l'anomalie est due aux erreurs dans les données, dans les règles de qualité, ou aux deux simultanément et deuxièmement à réparer les anomalies identifiées.

L'identification de la source de l'anomalie repose sur l'utilisation d'un ensemble de classifieurs supervisés entraînés sur des données générées par un GAN, en se basant sur des statistiques calculées à partir des caractéristiques des violations détectées.

La réparation des règles de qualité vise à générer à partir d'une règle erronée des règles dérivées de celle-ci en effectuant des ajouts, suppressions ou remplacements ciblés sur certains de ses attributs. Ces opérations sont réalisées en utilisant une mesure basée sur l'information mutuelle qui évalue les liens de dépendances entre les attributs de l'ensemble de données. Ensuite, seule la règle valide et présentant une transformation à changement minimal par rapport à la règle erronée est considérée comme une réparation.

Nous proposant également un processus de réparation hybride qui vise tout d'abord à éliminer les données erronées identifiées, afin d'obtenir un ensemble de données propre nous permettant d'appliquer notre processus de réparation des règles de qualité ,ensuite l'ensemble de données initiales contenant les données erronées est pris en considération

pour appliquer une réparation des données. La réparation des données est en dehors de la portée de ce mémoire.

Pour évaluer l'efficacité de notre solution, nous avons conduit une série d'expérimentations pour mettre en évidence la qualité de notre solution et son temps d'exécution.

Mots clés : Qualité de données, Règle de qualité, Détection des anomalies, Réparation des anomalies.

Abstract

Poor data quality is a serious and costly issue. Quality rules were initially designed to ensure data quality. Any subset that does not comply with the rules is considered a violation.

The majority of repair approaches modify the data to make them conform to the quality rules [Geerts et al. 2013, Fan et al. 2014, W. Chu et al. 2013, Lucie Berti-Equille et al. 2019]. However, quality rules can evolve over time as application or business rules change, as data is integrated with new data sources, or as the underlying data semantics evolve.

In similar situations, when there are inconsistencies, it becomes challenging to determine whether to repair the data, adjust the constraints, or take measures for both simultaneously.

In the literature, there are few studies suggesting solutions for repairing anomalies in quality rules [Golab et al. 2008]. Moreover, these studies assume that the data are initially clean.

Fewer approaches consider that the source of a violation can be in both the data and the quality rules [Volkovs et al. 2014, Beskales et al. 2013].

We present a solution aimed at firstly detecting whether the anomaly is due to errors in the data, in the quality rules, or both simultaneously, and secondly repairing the identified anomalies.

The identification of the source of the anomaly relies on the use of a set of supervised classifiers trained on data generated by a GAN, based on statistics calculated from the features of the detected violations.

Quality rule repair aims to generate derived rules from an erroneous rule by making targeted additions, deletions, or replacements of certain attributes. These operations are performed using a measure based on mutual information that evaluates dependencies between attributes in the dataset. Then, only the valid rule with minimal changes compared to the erroneous rule is considered a repair.

We also propose a hybrid repair process that first aims to eliminate identified erroneous data to obtain a clean dataset for applying our quality rule repair process. Then, the initial dataset containing the erroneous data is considered for data repair. Data repair is outside the scope of this thesis.

To evaluate the effectiveness of our solution, we conducted a series of experiments to highlight the quality of our solution and its execution time.

Keywords : Data Quality, Quality Rule, Anomaly Detection, Anomaly Repair.

ملخص

جودة البيانات الضعيفة هي مشكلة خطيرة ومكلفة. تم تصميم قواعد الجودة في البداية لضمان جودة البيانات. أي مجموعة فرعية لا تتوافق مع القواعد تعتبر مخالفة.

معظم الطرق المستخدمة لإصلاح البيانات تعدل البيانات لجعلها تتوافق مع قواعد الجودة ومع ذلك، يمكن أن تتطور قواعد الجودة مع مرور الوقت بمعدل تغير قواعد التطبيق أو التجارية، أو مع دمج البيانات مع مصادر بيانات جديدة، أو تغير الدلالة الأساسية للبيانات.

في حالات مماثلة، عندما تكون هناك تناقضات، يصبح من الصعب تحديد ما إذا كان يجب إصلاح البيانات، أو تعديل القيود، أو اتخاذ إجراءات للقيام بكليهما معًا.

هناك عدد قليل من الدراسات التي تقترح حلاً لإصلاح التناقضات في قواعد الجودة لكن تفترض هذه الدراسات أن البيانات نظيفة بشكل كلي.

قليلة هي الطرق التي تنظر إلى أن مصدر المخالفة يمكن أن يكون في .البيانات وكذلك في قواعد الجودة

نقدم حلا يهدف في المقام الأول إلى اكتشاف ما إذا كانت الشذوذات ناتجة عن أخطاء في البيانات، أو في قواعد الجودة، أو من الاثنين معًا، وثانيًا إلى إصلاح الشذوذات المحددة.

يعتمد تحديد مصدر الشذوذ على تقنية التصنيف التي يعتمد على الإحصائيات المحسوبة من ميزات الشذوذات المكتشفة.

إصلاح قواعد الجودة يهدف إلى توليد قواعد مشتقة من قاعدة خاطئة عن طريق إجراء إضافات مستهدفة أو حذف أو استبدال لبعض سماتها. تتم هذه العمليات باستخدام قياس يعتمد على المعلومات المتبادلة المستمدة من الارتباطات بين سمات البيانات. ثم، نعتبر فقط القاعدة الصالحة والتي تظهر تغييرًا دنيًا بالمقارنة مع القاعدة الخاطئة إصلاحًا.

نقدم أيضًا عملية إصلاح مختلطة تهدف في البداية إلى القضاء على البيانات الخاطئة المُحددة للحصول على مجموعة بيانات نظيفة تمكننا من تطبيق عملية إصلاح قواعد الجودة، ثم تسترد المجموعة الأصلية التي تحتوي على البيانات الخاطئة لإصلاحها.

لتقييم فعالية حلنا، أجرينا سلسلة من التجارب لتسليط الضوء على جودة الحل وزمن تنفيذه.

الكلمات الرئيسية: جودة البيانات، قاعدة الجودة، اكتشاف الشذوذ، إصلاح الشذوذ.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des sigles et acronymes

DQ	*Data quality/qualité de données*
FD	*Functional Dependency/dépendance fonctionnelle*
RQ	*Quality Rule/régle de qualité*

Introduction générale

La mauvaise qualité des données est un fait incontournable pour la plupart des organisations et peut avoir des implications sérieuses sur leur efficacité. Un exemple de domaine d'application critique est la santé, où des informations incorrectes peuvent provoquer de graves problèmes médicaux. Afin de garantir une haute qualité des données, il est essentiel de disposer d'un processus pour détecter et réparer les anomalies.

La détection des anomalies repose généralement sur des Règles de Qualité qui sont employées comme un moyen déclaratif pour capturer les inconsistance de données. Tout sous-ensemble de données qui n'est pas conforme aux règles définies est considéré comme étant une violation [Bohannon et al. 2005].

La réparation de données vise à réparer les anomalies détectées en minimisant la distance entre la base de données originale et la base de données modifiée selon une certaine fonction de coût. [Laure Berti-Equille et al. 2019].

La majorité des approches de réparation modifient les données afin de les faire correspondre aux règles de qualité. Cependant, les règles de qualité peuvent évoluer avec le temps à mesure que les règles d'application ou commerciales changent, que les données sont intégrées à de nouvelles sources de données, ou que les sémantiques sous-jacentes des données évoluent. [Chiang et al. 2011]. Dans de tels contextes, lorsque survient une inconsistance, il n'est plus clair s'il y a une erreur dans les données, ou si les contraintes ont évolué, ou les deux.

En outre, très peu d'études proposent des réparations pour les anomalies dans les règles de qualité par le biais d'ajouts, de suppressions et de raffinements.[Golab et al. 2008].Cependant, ces études émettent l'hypothèse que les données sont propres.

Dans ce document, nous présenterons notre propre solution pour la détection et la réparation des anomalies, en tenant compte des éventuelles erreurs présentes dans les données, dans les règles de qualité, ainsi que les erreurs présentes simultanément dans les données et dans les règles de qualité. Nous considérons les règles de qualité de type dépendances fonctionnelles. La réparation des données est en dehors de la portée de ce mémoire.

Pour l'identification de la source d'anomalie, nous extrayons les violations des contraintes et leurs caractéristiques qui servent de propriétés pour l'entraînement d'un ensemble de classifieurs supervisés sur des données générées par un GAN, permettant ainsi de prédire la source de l'anomalie.

Pour la réparation des contraintes, nous exploitons les liens de dépendances dans les attributs du jeu de données grâce à une mesure basée sur l'information mutuelle. Ces liens

de dépendances orientent l'ajout, la suppression et le remplacement d'attributs afin de garantir la validité des contraintes

Les objectifs de ce mémoire de fin d'études peuvent être résumés comme suit :

- Effectuer une étude bibliographique approfondie sur les approches d'identification et de réparation des anomalies dans les données et dans les règles de qualité.
- Concevoir des classifieurs supervisés entraînés sur des données générées par un GAN, en se basant sur des statistiques calculées à partir des caractéristiques des violations détectées, dans le but de prédire la source de ces anomalies en sortie.
- Concevoir un processus capable de proposer des transformations visant à réparer une contrainte erronée par ajout, supression ou remplacement.
- Mettre en place un processus de validation des contraintes visant à tester la validité des contraintes générées par transformation.
- Proposer des méthodes permettant de sélectionner la réparation présentant le cout le pus minimal de transformation parmi les règles détéctées valides.
- Concevoir un processus de réparation hybride reposant, pour la réparation des données, sur l'une des approches proposées dans la littérature, telle que [Abdellaoui et al. 2017], et s'appuyant sur notre propre processus proposé pour la réparation des règles, tout en identifiant un ordre correct pour l'application des deux.
- Évaluer notre solution en variant le taux d'erreurs dans les données ainsi que le type d'erreurs dans les règles.

Ce mémoire peut se résumer en 5 chapitres principaux :

1. **État de l'art** : Dans cette section, nous établissons les bases conceptuelles liées aux données et aux règles de qualité. Nous examinons en profondeur les fondements essentiels relatifs à la qualité des données. Par la suite, nous passerons en revue une série d'études tirées de la littérature, présentant des approches de détection des anomalies et réparation des données, des approches de détection des anomalies et réparation des règles de qualité.et notamment des approches hybrides.
2. **Conception** : Dans cette section, nous explorons en détail la conception de notre solution. Nous présentons l'approche globale que nous avons adoptée, en détaillant son fonctionnement, ainsi que les contributions existantes que nous avons adoptées et celles que nous apportons de manière novatrice.
3. **Réalisation** : Cette partie est dédiée à la présentation des aspects techniques de notre solution. Nous définissons les outils et langages que nous avons utilisés pour atteindre notre objectif. Par la suite, nous fournissons des explications détaillées sur la mise en œuvre de chaque étape composant notre solution.
4. **Tests et validation** : Dans cette dernière section, nous évaluons l'efficacité de notre solution en la confrontant à des ensembles de jeux de données réels et synthétiques.

Chapitre 1

État de l'art

1.1 Introduction

Les données sont le fondement sur lequel repose toute analyse et toute prise de décision éclairée. Dans cette perspective, nous allons tout d'abord présenter les notions fondamentales liées aux données, notamment leur qualité et les règles de qualité qui les guident.

La gestion de la qualité des données revêt une importance cruciale pour garantir la consistance des données. Dans cette optique, nous allons étudier les différentes approches existantes pour détecter les anomalies dans les données, identifier leur source, qu'il s'agisse des données elles-mêmes, des règles qui les gouvernent, ou d'une combinaison des deux et les réparer en fonction de leur source.

1.2 Généralités sur les données

1.2.1 Définition de données

En général, Data ou les données sont une séquence de symboles sur lesquels des opérations sont effectuées par un ordinateur. Il existe une infinité de définitions pour Data. Nous avons choisi la définition de [Sharda et al. 2014] :

Les données désignent un ensemble de faits généralement obtenus à la suite d'expériences, d'observations ou d'expérimentations. Les données peuvent consister en des chiffres, des lettres, des mots, des images, des enregistrements vocaux, et plus encore, en tant que mesures d'un ensemble de variables. Les données sont souvent considérées comme le niveau d'abstraction le plus bas. à partir duquel l'information, puis la connaissance, sont dérivées.

1.2.2 Classification des données

Au plus haut niveau d'abstraction, on peut classer les données en données structurées et non structurées (ou semi-structurées). La figure 1.1 présente une taxonomie simple des données [Sharda et al. 2014].

- **Les données catégorielles :** représentent les étiquettes ou les catégories de classes multiples utilisées pour diviser une variable en groupes spécifiques. Elles peuvent être réparties en deux catégories :
  1. **Les données nominales :** Représentent des codes simples attribués à des objets en tant qu'étiquettes.
  2. **Les données ordinales** représentent des codes attribués à des objets ou à des événements qui représentent également leur ordre de classement.
- **Les données numériques :** représentent les valeurs numériques de variables spécifiques, qui peuvent être entières ou réelles. Les données numériques peuvent également être appelées données continues, ce qui implique que la variable contient des mesures continues sur une échelle spécifique.

1. **Les données d'intervalle :** Les données d'intervalle sont des variables qui peuvent être mesurées sur des échelles d'intervalle
2. **les données de Ratio :** Elles comprennent des mesures physiques telles que la masse, la longueur et le temps qui sont mesurées sur des échelles de rapport et ont une valeur zéro non-arbitraire.

D'autres types de données, y compris les données textuelles, spatiales, les images et la voix, doivent être convertis en une forme de représentation catégorielle ou numérique avant de pouvoir être traités par les algorithmes de data mining.

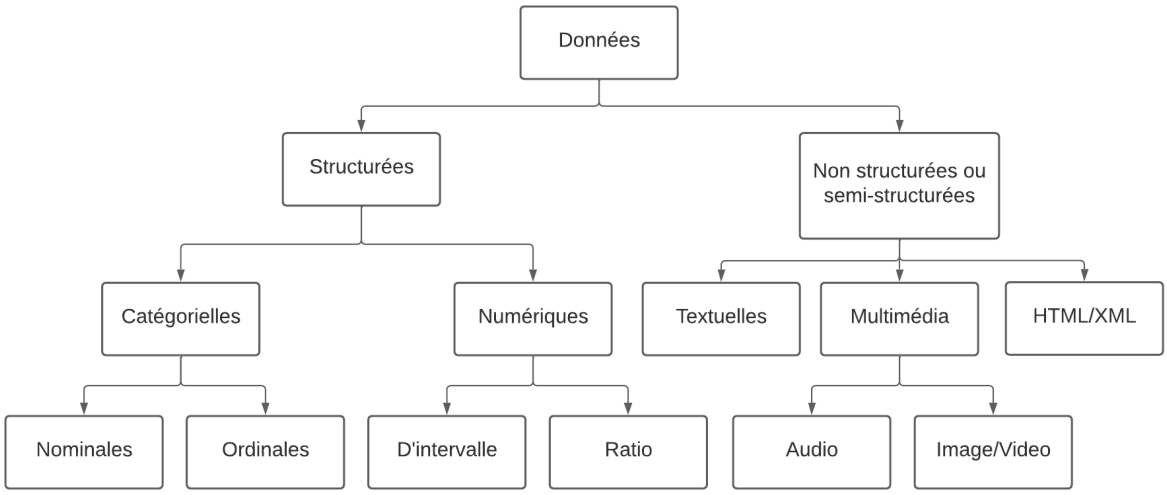


Fig. 1.1 : Une taxonomie simple des données[Sharda et al. 2014]

1.2.3 Définition de métadonnées

En général, les métadonnées constituent un ensemble structuré d'informations contextuelles et descriptives qui accompagnent et définissent les données primaires. Il existe une variété infinie de définitions pour les métadonnées. Dans ce contexte, nous avons adopté la définition de [Leroux et al. 2005]

"Les métadonnées sont des données qui décrivent d'autres données. Elles fournissent des informations sur la nature, l'origine, la structure, la présentation et l'utilisation des données. Les métadonnées peuvent être utilisées pour améliorer la recherche et la récupération d'information, ainsi que pour assurer la cohérence et la fiabilité des données."

1.2.4 Cycle de vie des données

Le cycle de vie des données est un processus continu et itératif qui assure la gestion complète des données, de leur acquisition à leur suppression, tout en respectant les réglementations et en maximisant leur valeur pour l'organisation. Ce cycle se compose de plusieurs étapes interconnectées qui garantissent que les données sont traitées de manière efficace et sécurisée tout au long de leur existence. [Wing 2019] a expliqué ces étapes comme suit :

1. **Génération des données** : Les données sont générées par les individus, les capteurs et les instruments scientifiques. Cela inclut les actions telles que les requêtes de recherche en ligne, les clics sur des liens, la surveillance par des capteurs de l'état de l'infrastructure physique, etc.

2. **Collecte des données** : Toutes les données générées ne sont pas collectées. La décision de ce qui doit être collecté est influencée par des facteurs tels que la nécessité et la capacité de traitement.

3. **Traitement des données** : Cette phase englobe des activités telles que le nettoyage, le formatage, la compression et la sécurité des données pour un stockage efficace.

4. **Stockage des données** : Les données sont stockées dans la mémoire, que ce soit sur des supports tels que des disques durs ou des dispositifs de stockage optiques.

5. **Gestion des données** : La gestion des données vise à optimiser les schémas d'accès prévus et à fournir une généralité maximale. Elle prend en compte la variété des données, leur type et leur vitesse d'arrivée.

6. **Analyse des données** : Cette phase englobe l'utilisation d'algorithmes et de techniques statistiques pour analyser les données en vue d'obtenir des connaissances, des informations, des prédictions ou des décisions.

7. **Visualisation des données** : Les résultats de l'analyse sont présentés de manière visuelle pour que les utilisateurs humains puissent les comprendre plus facilement.

8. **Interprétation des données** : En plus de la visualisation, l'interprétation consiste à expliquer le contexte, le sens, les implications et les conséquences des résultats obtenus à partir des données.

9. **Utilisation par l'humain** : Les utilisateurs humains, qu'ils soient des scientifiques, des décideurs, des médecins, des financiers, des juristes ou des professionnels du secteur, utilisent les informations issues de l'analyse des données pour prendre des décisions et réaliser des actions.

1.2.5 L'importance des Données dans la Prise de Décision

Les liens entre les données et la prise de décision sont profonds et deviennent de plus en plus cruciaux à mesure que la technologie et l'analyse des données progressent. Les données jouent un rôle fondamental dans le processus de prise de décision. L'article de [Davenport et al. 2017] souligne ce lien en déclarant :

"Les données sont devenues essentielles à la prise de décision dans tous les secteurs, allant des entreprises aux gouvernements en passant par les hôpitaux. Elles peuvent fournir des informations précieuses sur les tendances du marché, les besoins des clients et les performances de l'entreprise. Les données sont en mesure d'aider à identifier les opportunités d'amélioration, à réduire les risques et à accroître la satisfaction des clients."

1.3 Généralités sur la qualité de données

1.3.1 Définition de la qualité des données

Une qualité de données élevée est essentielle pour garantir que les informations tirées des données sont correctes et significatives. [Institute 2000] définit la qualité des données comme suit : "La qualité des données est un jugement ou une évaluation de la mesure dans laquelle les données sont adaptées à leur utilisation prévue dans un contexte particulier". De même, selon [ISO 2019] "la qualité des données est la propriété d'une donnée qui la rend appropriée à son utilisation prévue".

1.3.2 Les dimensions de la qualité des données

Dans les travaux de la littérature existante, de nombreuses définitions de dimensions de qualité ont été considérés par différents auteurs. Selon [Sidi et al. 2012], les auteurs ont examiné comment évaluer ou estimer la qualité des données. Ils ont déclaré qu'une partie des évaluations de la qualité des données pouvait être indépendante de la tâche, et donc non liée au contexte de l'application. En revanche, d'autres sont dépendantes de la tâche. Le tableau suivant illustre les principales dimensions qu'ils estiment utiles pour le débat :

Dimensions	Définitions
Accessibilité	Le degré de disponibilité des données, ainsi que la facilité et la rapidité de leur récupération.
Consistance	Mesure le degré de conformité aux règles sémantiques définies pour l'ensemble de données.
Volume de données	Mesure la taille des données pour évaluer si elle est adéquate pour accomplir la tâche.
Crédibilité	Mesure la véracité et la source fiable des données, ainsi que la crédibilité et la convaincance de leur valeur.
Complétude	Mesure l'absence de données manquantes ou insuffisantes.
Représentation Cohérente	Mesure la cohérence dans la manière dont les données sont affichées ou fournies.
Facilité de manipulation	La facilité avec laquelle les données peuvent être traitées et utilisées pour différentes tâches.
Sans erreur	Degré de fiabilité, d'exactitude et d'intégrité des données.
Intelligibilité	Mesure la clarté des langues, des symboles, des unités et des définitions utilisés dans les données.
Objectivité	La mesure dans laquelle les définitions des données sont objectives et ne contiennent pas de langage ou de syntaxe inappropriés.
Pertinence	Mesure la capacité des données à être utiles pour résoudre un problème donné.
Réputation	Degré de confiance accordé aux données.
Sécurité	Le niveau de contrôle d'accès fourni pour garantir la sécurité des données.
Actualité	Le degré de mise à jour des données.
Compréhensibilité	La facilité avec laquelle les données peuvent être comprises par l'utilisateur.
Valeur ajoutée	La valeur intrinsèque des données pour l'utilisateur.

Tab. 1.1 : Table des dimensions de qualité des données [Sidi et al. 2012]

1.3.3 Importance de la qualité des données

La qualité des données est un aspect essentiel de toute stratégie basée sur les données (Data Driven). Aujourd'hui, les organisations génèrent et collectent des quantités énormes de données, et la qualité de ces données peut avoir un impact significatif sur les décisions, les opérations et les résultats de l'entreprise. Une mauvaise qualité des données peut entraîner des analyses incorrectes ou incomplètes, des décisions erronées, un gaspillage de ressources et des opportunités perdues. En revanche, des données de haute qualité peuvent aider les organisations à obtenir des informations pertinentes, à prendre des décisions plus éclairées et à améliorer leurs performances globales. Par conséquent, il est essentiel de garantir la qualité des données pour toute organisation qui souhaite maximiser la valeur de ses données et obtenir un avantage concurrentiel dans son secteur. [Batini et al. 2009]

Le développement des technologies de l'information au cours des dernières décennies a permis aux organisations de collecter et de stocker d'énormes quantités de données. Cependant, à mesure que les volumes de données augmentent, leur gestion devient de plus en plus complexe. Étant donné que des ressources informationnelles plus grandes et plus

complexes sont collectées et gérées par les organisations aujourd'hui, cela signifie que le risque de mauvaise qualité des données augmente. Une mauvaise qualité des données peut entraîner de nombreuses conséquences négatives pour une entreprise. En premier lieu, des données de mauvaise qualité qui ne sont pas identifiées et corrigées peuvent avoir des impacts économiques et sociaux considérables sur une organisation. Cela se traduit par une satisfaction client réduite, des coûts d'exploitation accrus, des processus décisionnels inefficaces et des performances réduites. De plus, une mauvaise qualité des données augmente les coûts opérationnels, car du temps et d'autres ressources sont consacrés à la détection et à la correction des erreurs. Étant donné que les données sont créées et utilisées dans toutes les opérations quotidiennes, elles constituent des éléments essentiels pour presque toutes les décisions et contribuent implicitement à définir les termes communs dans une entreprise. Ainsi, une mauvaise qualité des données peut avoir des effets négatifs sur la culture organisationnelle.

selon [Redman 1998], au moins trois études propriétaires ont donné des estimations de manière informelle, 40 à 60 % des dépenses de l'organisation de services peuvent être consommées en raison de données de mauvaise qualité. Ces chiffres montrent que même de petites inexactitudes dans les données peuvent avoir un impact économique significatif.

il existe un manque apparent d'études approfondies sur la qualité des données dans les revues universitaires, mais de nombreux experts de l'industrie en fournissent. [Marsh 2005] résume les conclusions de ces enquêtes sous forme de points clés suivants :

- 88% de tous les projets d'intégration des données échouent complètement ou dépassent considérablement leur budget.
- 75% des organisations ont identifié des coûts découlant de données incorrectes.
- 33% des organisations ont retardé ou annulé de nouveaux systèmes informatiques en raison de données de mauvaise qualité.
- 611 milliards de dollars par an sont perdus aux États-Unis uniquement en raison de courriers mal ciblés et de frais généraux du personnel.
- Selon Gartner, les données erronées sont la principale cause d'échec des systèmes CRM.
- Moins de 50% des entreprises affirment avoir une grande confiance dans la qualité de leurs données.
- Les projets de Business Intelligence (BI) échouent souvent en raison de données incorrectes, il est donc impératif que les décisions commerciales basées sur la BI reposent sur des données propres.
- Seulement 15% des entreprises ont une grande confiance dans la qualité des données externes qui leur sont fournies.
- Les données clients se détériorent généralement de 2% par mois, soit 25% par an.
- Les organisations surestiment généralement la qualité de leurs données et sous-estiment le coût des erreurs.

- Les processus commerciaux, les attentes des clients, les systèmes sources et les règles de conformité changent constamment. Les systèmes de gestion de la qualité des données doivent en tenir compte.
- Des quantités considérables de temps et d'argent sont dépensées pour la programmation personnalisée et les méthodes traditionnelles - généralement pour lutter contre une crise immédiate plutôt que de traiter le problème à long terme.

1.3.4 Classification des problèmes de qualité des données

La classification des problèmes de qualité des données est une étape essentielle dans la gestion et l'amélioration de la qualité des données. Elle permet de catégoriser les différentes difficultés auxquelles on peut être confronté lors de la manipulation et de l'utilisation des données. Plusieurs travaux de recherche se sont intéressés à cette problématique et ont proposé différentes approches de classification.

Dans notre étude, nous avons choisi une classification multiple des problèmes de qualité des données. [Rahm et al. 2000] propose une classification des problèmes à résoudre lors du nettoyage des données. Ils distinguent entre les problèmes liés à une seule source et ceux liés à plusieurs sources, ainsi qu'entre les problèmes liés au schéma et ceux liés aux instances (voir la figure 1.2). Les problèmes liés à plusieurs sources surviennent lorsqu'il y a plusieurs sources de données à intégrer, avec différentes représentations de données, des données qui se chevauchent ou se contredisent. Les problèmes liés au schéma sont des problèmes de qualité qui peuvent être évités grâce à des contraintes d'intégrité appropriées ou à une meilleure conception de schéma, tandis que les problèmes liés aux instances ne peuvent pas être évités au niveau du schéma (par exemple, les erreurs d'orthographe). Cela explique la classification des types d'erreurs.

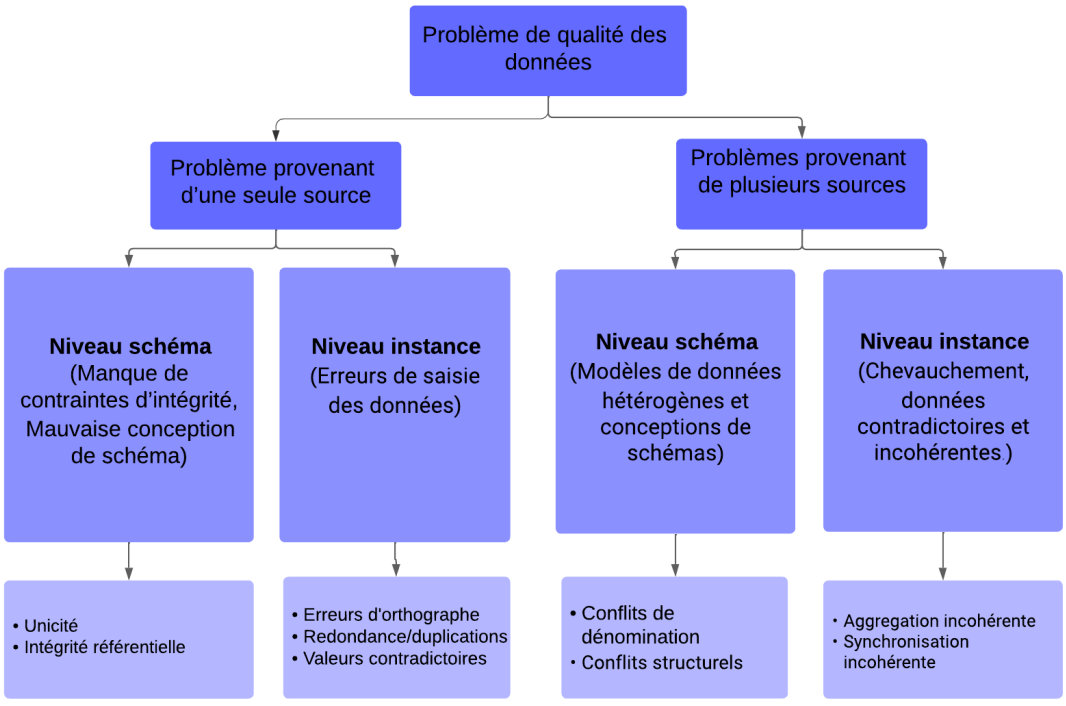


Fig. 1.2 : Classification des problèmes de la qualité des données [Rahm et al. 2000].

[Müller et al. 2005] décrivent une classification plus détaillée des anomalies de données. Ils commencent par différencier les anomalies syntaxiques, les anomalies sémantiques et les anomalies de couverture (valeurs manquantes). Les anomalies syntaxiques comprennent les erreurs lexicales, les erreurs de format de domaine et les irrégularités concernant l'utilisation non uniforme des valeurs (par exemple, l'utilisation de devises différentes). Les anomalies sémantiques incluent les violations de contraintes d'intégrité, les contradictions (par exemple, une différence entre l'âge et la date de naissance), les entrées dupliquées et les tuples invalides. Dans ce contexte, les tuples invalides ne représentent pas des entités valides du monde réel, mais ne violent pas non plus de contraintes d'intégrité. Les anomalies de couverture peuvent être divisées en valeurs manquantes et en tuples manquants.

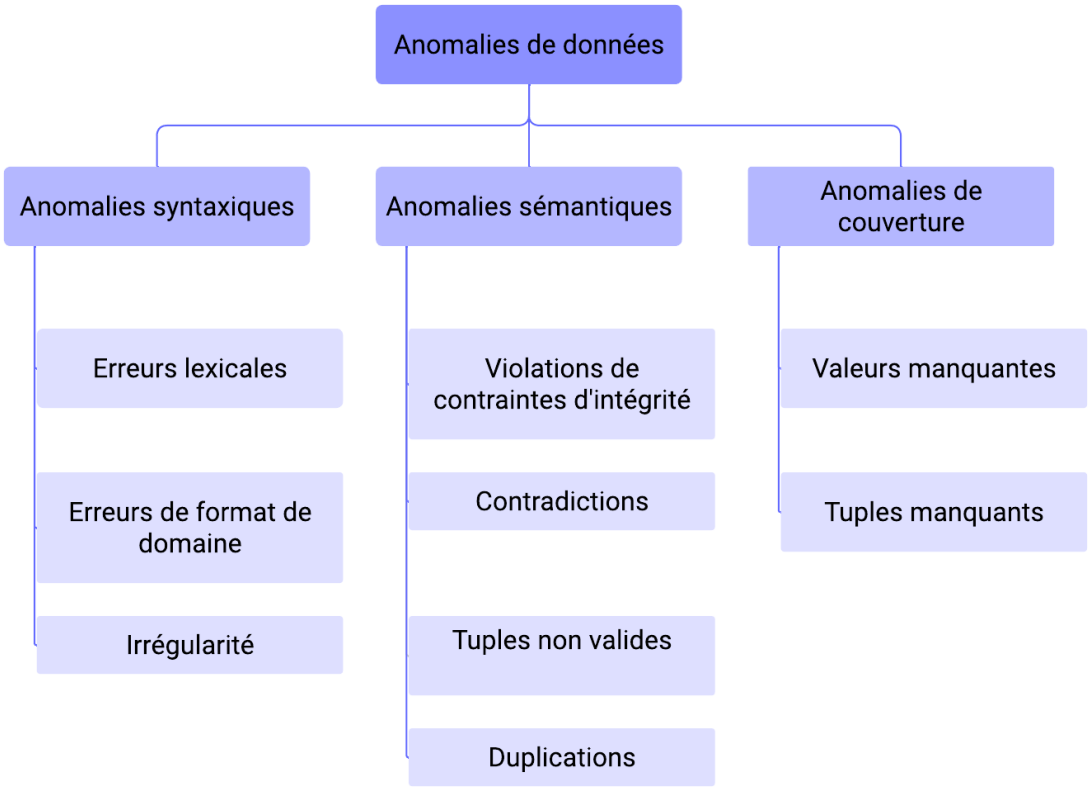


Fig. 1.3 : Classification des problèmes de la qualité des données [Müller et al. 2005].

1.4 Généralités sur les règles de qualité

1.4.1 Définition des règles de qualité

Les règles de qualité de données sont des directives et critères établis pour assurer l'exactitude, la cohérence, la complétude et d'autres caractéristiques essentielles des données dans un système ou une organisation. Elles garantissent que les données sont fiables, pertinentes et adaptées à leur utilisation prévue. selon [Wang 1998] une règle de qualité de données sont définies comme suit : "Une règle de qualité de données est une assertion qui décrit une propriété que les données doivent posséder. Cette propriété peut être une propriété de l'ensemble des données, d'une partie de l'ensemble des données, ou d'une donnée individuelle. Les règles de qualité de données peuvent être utilisées pour spéci-

fier les exigences de qualité pour un ensemble de données particulier, ou pour décrire les propriétés que les données doivent posséder pour être considérées comme de qualité."

1.4.2 Types des règles de qualité

[Ilyas et al. 2015] explique plusieurs types de règles de qualité de données, parmi ces types on cite :

1. **Dépendances fonctionnelles** : Les dépendances fonctionnelles sont des règles qui expriment la relation entre les valeurs d'attributs dans une table de base de données. Une dépendance fonctionnelle indique que les valeurs d'un ensemble d'attributs (appelé attributs de gauche) déterminent de manière unique les valeurs d'un autre ensemble d'attributs (appelé attributs de droite). Formellement, si A et B sont des ensembles d'attributs, alors $A \rightarrow B$ si, pour chaque occurrence de même valeur dans A, les occurrences correspondantes dans B sont également identiques.

2. **Dépendances fonctionnelles conditionnelles** : Les dépendances fonctionnelles conditionnelles sont des extensions des dépendances fonctionnelles standard. Elles définissent des dépendances entre attributs sous certaines conditions. Ainsi, la dépendance entre A et B peut être valable uniquement si une certaine condition C est satisfaite.

3. **Contraintes de déni** : Les contraintes de déni énoncent les conditions sous lesquelles certaines relations entre les données ne sont pas autorisées. Elles peuvent être utilisées pour empêcher des combinaisons de données inappropriées ou incorrectes. Par exemple, une contrainte de déni pourrait spécifier qu'un client ne peut pas avoir plus d'une adresse de livraison active en même temps.

4. **Dépendances d'inclusion** : Les dépendances d'inclusion définissent que si un ensemble d'attributs A appartient à une table, alors un autre ensemble d'attributs B doit également y appartenir. En d'autres termes, si une condition est remplie pour A, elle doit également être remplie pour B.

5. **Dépendances de correspondance** : Les dépendances de correspondance exigent que les valeurs d'attributs dans un ensemble correspondent exactement aux valeurs dans un autre ensemble. Ces règles sont souvent utilisées pour garantir la cohérence entre les données provenant de différentes sources.

6. **Dépendances fonctionnelles métriques** : Les dépendances fonctionnelles métriques sont des règles qui reposent sur des mesures ou des métriques spécifiques associées aux attributs. Par exemple, une telle règle pourrait exiger que le ratio entre deux attributs numériques soit toujours supérieur à une valeur seuil.

7. **Dépendances fonctionnelles numériques** : Les dépendances fonctionnelles numériques s'appliquent spécifiquement aux attributs numériques. Elles définissent les relations entre les valeurs numériques des attributs, garantissant que certaines valeurs numériques déterminent de manière unique d'autres valeurs.

1.4.3 Violation des règles de qualité

La violation des règles de qualité fait référence à la situation où les critères établis pour garantir la qualité des données sont enfreints. Cela peut conduire à l'utilisation de données incorrectes, incohérentes ou non conformes aux critères préétablies. [Zhang et al. 2010] définit la violation des règles de qualité comme suit : "Une violation des règles de qualité de données est une instance où une donnée ne répond pas à une règle ou une contrainte définie." Et selon [Özyurt et al. 2007] sa définition est comme suit :"Une violation des règles de qualité de données est une erreur dans les données qui peut affecter la précision, l'intégrité ou la cohérence des données."

1.5 Gestion de qualité des données

La gestion de la qualité des données est un processus continu qui vise à assurer la consistance des données au sein d'une organisation. C'est un élément essentiel pour la prise de décision et l'analyse des données. [Dallachiesa et al. 2020]

Le processus de gestion de la qualité des données est une série d'étapes et d'activités, il englobe principalement la détection des anomalies et leur réparation.

1.6 Détection des anomalies

Les anomalies dans les données sont très fréquentes et ont eu de graves conséquences pour de nombreuses entreprises, entraînant d'importantes pertes financières. Parmi les anomalies, on trouve celles liées aux violations des règles de qualité. Il existe des méthodes permettant la détection de ce type d'anomalie.

La détection des violations de contraintes est le processus par lequel on identifie les enregistrements ou les tuples dans un ensemble de données qui ne respectent pas les contraintes spécifiées [Shaoxu Song 2016].

1.6.1 Étude bibliographique

Dans cette partie, nous allons détailler les différents travaux qui traitent la détection des anomalies des données.

1.6.1.1 Constraint-Variance Tolerant Data Repairing (Shaoxu Song Han Zhu Jianmin Wang) [Shaoxu Song 2016]

l'article définit la notion de violation comme tous les enregistrements de l'ensemble de données qui violent une contrainte spécifique . Cet ensemble est constitué de listes de tuples, où chaque tuple contient des valeurs qui violent la contrainte . L'ensemble de violations total pour l'ensemble de contraintes est l'union de tous les ensembles de violations individuels pour chaque contrainte. Cela permet de regrouper toutes les violations de contraintes dans un seul ensemble, ce qui simplifie leur gestion.

Pour représenter ces violations de manière efficace, l'article introduit le concept de "cellule" qui correspond à une unité de données dans l'ensemble de données. En d'autres termes, chaque attribut ou colonne de l'ensemble de données est considéré comme une cellule.

Pour chaque contrainte , l'article définit l'ensemble des cellules qui sont impliquées dans les attributs de cette contrainte, c'est-à-dire les cellules qui participent aux conditions de la contrainte. Cela permet de créer une structure de données qui relie les violations de contraintes aux cellules spécifiques qui les ont causées.

L'article utilise ensuite ces informations pour construire un "graphe de conflit". Dans ce graphe, chaque cellule de l'ensemble de données est un nœud, et chaque violation, représentée par une liste de tuples, forme une arrête reliant les cellules qui ont causé la violation.

1.6.1.2 Continuous Data Cleaning (Maksims Volkovs , Fei Chiang, Jaroslaw Szlichta, Renee J. Miller)[Volkovs et al. 2014]

L'article définit une notion appellée motif qui est un ensemble de lignes spécifiques dans les données, obtenu en choisissant certaines colonnes à partir de toutes les lignes d'une relation.

Ensuite, il explique que pour détecter les violations de contraintes, on compare ces motifs. Si on trouve deux motifs distincts qui ont la même sélection de colonnes gauches mais des données différentes dans les colonnes droites, alors on considère que c'est une violation. Cela signifie que la règle qui devrait être respectée n'est pas satisfaite, car les données ne correspondent pas comme elles le devraient.

1.6.2 Synthèse

Les deux articles examinent des techniques liées à la détection des violations de contraintes dans les ensembles de données. Chacun introduit des structures de données distinctes pour représenter et gérer ces violations

Le premier article [Shaoxu Song 2016] a mis l'accent sur la détection des violations de contraintes en regroupant les enregistrements violant les règles en ensembles de tuples et en utilisant un "graphe de conflit" pour visualiser les relations entre les cellules de données et les violations.

Le deuxième article [Volkovs et al. 2014] repose sur la détection de violations en comparant des motifs, en recherchant des lignes où les colonnes de gauche sont identiques mais les données de colonnes de droite diffèrent, signalant ainsi des incohérences par rapport aux contraintes établies.

1.7 Identification de la source d'anomalie

La majorité des approches de réparation des données consiste à utiliser les règles de qualité pour identifier les tuples contenant des erreurs, puis à utiliser ces règles pour dériver

des réparation pour ces tuples. Cependant, dans de nombreuses applications modernes, les contraintes peuvent évoluer avec le temps à mesure que les règles de l'application ou de l'entreprise changent, que les données sont intégrées à de nouvelles sources de données ou que la sémantique sous-jacente des données évolue. Dans de tels contextes, lorsqu'une incohérence se produit, il n'est plus clair s'il y a une erreur dans les données et les données doivent être réparées ou si les contraintes ont évolué et les contraintes doivent être réparées ou les deux simultanément.

1.7.1 Étude bibliographique

Dans cette partie, nous allons détailler les différents travaux qui traitent l'identification de la source d'anomalie.

1.7.1.1 Unified Model for Data and Constraint Repair (Fei Chiang, Renee J. Miller) [Chiang et al. 2011]

Pour déterminer si la source d'une anomalie est due aux règles ou aux données, l'article propose d'abord d'effectuer les deux types de réparation. Celle qui génère un coût minimal est celle qui reflète le type d'anomalie.

Le modèle de coût proposé par l'article repose sur trois aspects principaux :

- **La précision** : Cela mesure la capacité d'une réparation des données à minimiser la différence entre les valeurs existantes et les valeurs cibles, tout en exigeant un niveau minimal de support.
- **La redondance** : Ce critère vise à maximiser la couverture des données tout en maintenant les règles, même en ajoutant des attributs si nécessaire.
- **La concision** : Il vise à obtenir des règles réparées qui expliquent les données de manière concise, sans être trop ajustées ou simplifiées.

Si la réparation des données donne le coût minimal, alors la source de l'anomalie est la donnée. Sinon, c'est la règle.

1.7.1.2 Continuous Data Cleaning (Maksims Volkovs , Fei Chiang, Jaroslaw Szlichta, Renee J. Miller)(Volkovs et al. 2014)

L'article propose une méthode de prédiction de la source d'anomalie qui repose sur un modèle de classification, plus précisément la régression logistique, qui utilise un processus sophistiqué pour prédire la source d'anomalie en fonction des caractéristiques des données et des règles de qualités.

Cette approche commence par la collecte de données d'apprentissage, qui comprend une vaste quantité d'informations représentant les relations entre les données, les dépendances fonctionnelles qui les régissent, ainsi que les actions de réparation qui ont été précédemment mises en œuvre pour résoudre des incohérences. Une fois ces données recueillies, l'accent est mis sur la caractérisation des motifs de violation des dépendances

fonctionnelles. Ces motifs représentent des groupes de données qui ne satisfont pas aux contraintes définies, ce qui implique l'existence d'anomalie.

Pour prédire la source d'anomalie, le modèle de régression logistique utilise un ensemble de 22 statistiques calculées à partir de ces motifs de violation. Ces statistiques comprennent des mesures telles que la fréquence d'apparition des motifs, le nombre de tuples de données en violation, et d'autres métriques.

Enfin, en utilisant ces statistiques, le modèle attribue une classe d'anomalie parmi trois possibles à chaque motif en violation, indiquant ainsi si une anomalie est liée aux dépendances fonctionnelles, aux données, ou les deux simultanément.

1.7.2 Synthèse

Les deux articles abordent le problème de l'identification de la source d'anomalie dans les données. Le premier article [Chiang et al. 2011] propose une approche basée sur la réparation des données et des règles, en sélectionnant la source d'anomalie comme celle ayant entraîné la réparation la moins coûteuse, et il élimine la possibilité qu'une anomalie puisse résulter des deux types d'erreurs simultanément.

Le deuxième article [Volkovs et al. 2014] propose une méthode de classification basée sur la régression logistique, utilisant 22 statistiques pour classifier les motifs de violation des dépendances fonctionnelles en trois catégories : anomalie dans les données, anomalie dans la règle et anomalie des deux.

1.8 Réparation des données

L'automatisation de la gestion de la qualité des données a toujours été un objectif majeur, et la réparation des données se positionne au cœur de ce processus de gestion. La réparation de données consiste à trouver une modification aux données de manière à éliminer toutes les violations par rapport aux contraintes. En faisant référence au principe du changement minimum, on s'attend à ce que la réparation des données soit aussi proche que possible de la version originale des données [Shaoxu Song 2016].

1.8.1 Étude bibliographique

Dans cette partie, nous allons détailler les différents travaux qui traitent la réparation des données

1.8.1.1 Continuous Data Cleaning (Maksims Volkovs , Fei Chiang, Jaroslaw Szlichta, Renee J. Miller)[Volkovs et al. 2014]

L'approche présentée dans cet article se concentre sur la réparation d'une violation d'une dépendance fonctionnelle $F : X \rightarrow A$. Lorsque deux tuples, t1 et t2, enfreignent cette dépendance en ayant des valeurs différentes pour A tout en ayant des valeurs identiques pour X, l'algorithme propose deux types de réparations :

- Réparation de données à droite : Cette approche consiste à changer les valeurs de l'attribut A pour qu'elles soient identiques entre t1 et t2. L'attribut A est modifié de manière à ce que A(t1) soit égal à A(t2).
- Réparation de données à gauche : Dans ce cas, les valeurs de l'attribut X sont modifiées pour qu'elles soient différentes entre t1 et t2. L'algorithme cherche un autre tuple, t3, qui partage les mêmes valeurs pour l'attribut A avec t1 (A(t1) = A(t3)), mais qui a des valeurs différentes pour l'attribut X (X(t1) X(t3)). Les valeurs de X(t1) sont ajustées pour correspondre à X(t3).

la réparation choisie pour t1 est généralement appliquée à tous les tuples partageant les mêmes valeurs d'attributs X et A que t1.

1.8.1.2 Unified Model for Data and Constraint Repair (Fei Chiang, Renee J. Miller) [Chiang et al. 2011]

L'article présente un algorithme de réparation de données visant à réparer les données qui ne respectent pas certaines contraintes spécifiées tout en minimisant le coût global de la réparation.

Pour chaque dépendance fonctionnelle F : X → Y et des tuples t1 et t2 qui violent cette dépendance, l'algorithme propose deux types de réparations : soit réparer les valeurs Y pour les rendre identiques, soit réparer les valeurs X pour les rendre différentes. Dans les deux cas, l'algorithme exige que les données soutiennent la modification, c'est-à-dire qu'il utilise des preuves dans les données pour effectuer la réparation.

L'idée générale de cette approche est d'optimiser la réparation des données en identifiant des motifs de tuples fréquents (motifs de base) et en les remplaçant par des signatures pour réduire le coût global de la représentation des données tout en maintenant la similarité des motifs de déviants avec les motifs de base au-dessus d'un seuil défini. Cette méthode permet une réparation efficace des données tout en minimisant la perte d'information.

Plus en détail, l'algorithme commence par considérer les motifs de tuples de base comme signatures, tandis que les motifs de déviants sont des candidats pour la réparation. Il utilise des fonctions de similarité sur les attributs pour évaluer le coût de différentes réparations de données. De plus, il examine l'impact potentiel de la réparation des données sur d'autres contraintes pour garantir la cohérence globale des données.

1.8.1.3 Constraint-Variance Tolerant Data Repairing (Shaoxu Song Han Zhu Jianmin Wang) [Shaoxu Song 2016]

Les auteurs dans cet article abordent le problème de la réparation des données. L'approche présentée vise à résoudre les violations des contraintes existantes tout en minimisant le coût de réparation des données. Contrairement à l'approche de réparation holistique des données, qui élimine uniquement les violations actuellement observées mais peut introduire de nouvelles violations, cette méthode garantit une réparation en une seule étape sans introduire de nouvelles violations.

L'approche prend en compte non seulement les violations existantes mais aussi les "suspects", c'est-à-dire les données qui pourraient potentiellement introduire de nouvelles violations après la réparation. Pour chaque suspect, un contexte de réparation est défini pour garantir que les modifications apportées aux données en cours de réparation respectent les prédicats des contraintes.

En cas d'impossibilité de satisfaction de tous les contextes de réparation avec les valeurs existantes, des variables fraîches sont introduites pour représenter des valeurs inconnues. Le processus de réparation est décomposé en composants pour permettre le partage de la résolution entre différentes parties du problème, ce qui améliore l'efficacité globale.

De plus, les contextes de réparation peuvent être raffinés pour devenir plus stricts, facilitant ainsi la résolution du problème. Enfin, les solutions optimales trouvées pour des problèmes similaires peuvent être réutilisées pour accélérer la résolution de problèmes ultérieurs.

1.8.2 Synthèse

Les trois articles examinés abordent la réparation de données, mais chacun adopte une approche distincte pour résoudre ce problème.

Le premier article [Volkovs et al. 2014] propose deux types de réparations : la réparation des données à droite, où les valeurs de l'attribut droite sont modifiées pour être identiques entre les tuples de violations et la réparation des données à gauche, où les valeurs de l'attribut à gauche sont modifiées pour être différentes entre les tuples de violations. De plus, la réparation choisie pour un tuple est généralement appliquée à tous les tuples partageant les mêmes valeurs d'attributs constituant la DF violée.

Le deuxième article [Chiang et al. 2011] propose les mêmes type de réparation des données à gauche et à droite mais il utilise de plus des preuves dans les données pour justifier la réparation en identifiant des motifs de tuples fréquents et en remplaçant ces motifs par des signatures pour réduire le coût global tout en maintenant la similarité des motifs.

Le troisième article [Shaoxu Song 2016] aborde le problème de la réparation des données en prenant en compte non seulement les violations de contraintes existantes, mais aussi les "suspects" qui pourraient introduire de nouvelles violations après réparation. Cette méthode garantit une réparation en une seule étape sans introduire de nouvelles violations en définissant des contextes de réparation pour chaque suspect, et en cas d'impossibilité de satisfaction de tous les contextes de réparation avec les valeurs existantes, des variables fraîches sont introduites pour représenter des valeurs inconnues.

1.9 Réparation des règles de qualité

Les règles de qualité sont un outil important pour garantir un aspect de la qualité des données, à savoir la cohérence des données.

Les règles de qualité ont été initialement conçues pour garantir la qualité des données en empêchant les données de devenir incohérentes. Cependant, les règles de qualité de données peuvent être inexactes en raison du bruit inhérent à leur découverte, ou bien elles peuvent devenir obsolètes du fait que la logique métier n'est pas statique et peut évoluer avec le temps.

Ce problème a été relativement peu exploré dans la littérature. Il existe un nombre limité d'articles qui abordent spécifiquement cette problématique.

1.9.1 Étude bibliographique

Dans cette partie, nous allons détailler les différents travaux qui traitent la réparation des RQs.

1.9.1.1 Unified Model for Data and Constraint Repair (Fei Chiang, Renee J. Miller) [Chiang et al. 2011]

Dans cet article, les auteurs abordent la réparation des dépendances fonctionnelles violées, qui sont des règles définissant une relation entre un ensemble d'attributs à gauche et un autre ensemble d'attributs à droite.

La proposition de réparation des contraintes consiste à ajouter un attribut à l'ensemble des attributs à gauche de la dépendance fonctionnelle. Cet attribut supplémentaire fournit un contexte supplémentaire pour déterminer quand la règle est effectivement respectée. Cela peut résoudre les incohérences lorsque certaines tuples violent la contrainte mais sont proches d'autres valeurs bien supportées dans la relation.

L'algorithme évalue les attributs qui ne sont pas présents dans la contrainte en calculant leur score de pertinence. Il sélectionne l'attribut ayant le score le plus bas comme l'attribut gagnant pour la réparation de la contrainte.

Les mesures proposées pour calculer ce score sont les suivantes :

- **Le partitionnement des attributs** : il consiste à regrouper les tuples ayant les mêmes valeurs pour un attribut X. Ensuite, des raffinements de ce partitionnement sont créés en formant des classes pour chaque ensemble de tuples partageant les mêmes valeurs pour X et Y. Ce partitionnement permet de comparer les valeurs de l'attribut candidat avec les valeurs de l'attribut en conflit.
- **La mesure de la variance de l'information** : elle est dérivée des techniques de regroupement et évalue la similarité entre la distribution des valeurs de l'attribut candidat et les tuples nécessitant une réparation. En minimisant cette mesure, l'attribut qui présente la distribution de valeurs la plus similaire à celle des tuples nécessitant une réparation est identifié.
- **L'évaluation de l'homogénéité et de l'exhaustivité** : L'homogénéité mesure si chaque classe d'attribut ne contient que des valeurs provenant d'un seul ensemble de tuples en conflit, tandis que l'exhaustivité mesure si tous les tuples d'un ensemble en conflit sont assignés à une seule classe d'attribut. En sélectionnant l'attribut

avec le score d'homogénéité le plus bas, celui qui peut encoder le plus grand nombre de valeurs nécessitant une réparation est choisi. à celle des tuples nécessitant une réparation est identifié.

1.9.1.2 Continuous Data Cleaning (Maksims Volkovs , Fei Chiang, Jaroslaw Szlichta, Renee J. Miller)[Volkovs et al. 2014]

Dans cet article, l'auteur propose une méthode visant à remédier aux contraintes en ajoutant des attributs supplémentaires à gauche des dépendances fonctionnelles existantes, afin de résoudre les violations. Toutefois, il souligne l'importance d'ajouter ces attributs de manière réfléchie. Il est crucial de choisir des attributs permettant une séparation claire des tuples en conflit, tout en préservant autant que possible la cohérence des données existantes par rapport aux dépendances fonctionnelles. De plus, l'auteur mentionne que, dans certains cas, il peut être nécessaire de solliciter l'avis de l'utilisateur du domaine pour déterminer un seuil qui fixe la limite du nombre d'attributs à ajouter, étant donné qu'il peut y avoir des informations supplémentaires ou des préférences spécifiques.

1.9.1.3 On the Relative Trust between Inconsistent Data and Inaccurate Constraints (George Beskales ,Ihab F. Ilyas , Lukasz Golab , Artur Galiullin)[Beskales et al. 2013]

Selon l'article, il est possible de réparer les FDs existantes en ajoutant des attributs à la partie gauche (LHS) de ces FDs. Cependant, cela doit être fait en respectant certaines conditions comme l'interdiction d'ajouter les attributs de la partie droite ou l'ensemble des attributs déjà présents dans la partie gauche de la FD à la partie gauche afin d'éviter de produire des FDs triviales.

1.9.1.4 Constraint-Variance Tolerant Data Repairing (Shaoxu Song Han Zhu Jianmin Wang) [Shaoxu Song 2016]

L'article aborde la question de la suppression des prédicats dans le cadre de la réparation des contraintes. Les auteurs reconnaissent que trop de prédicats peuvent entraîner une contrainte surajustée aux données, tandis qu'une suppression excessive de prédicats peut conduire à une contrainte excessivement simplifiée. Par conséquent, les auteurs suggèrent qu'une suppression modérée des prédicats est préférable pour une réparation à faible coût. Ils soulignent que plus on supprime de prédicats, plus la contrainte devient simplifiée.Mais sans donner plus de détails sur la manière dont les attributs à supprimer sont choisis.

1.9.2 Synthèse

Les articles étudiés ont apporté des idées importantes pour la réparation des contraintes en incluant l'ajout ou la suppression d'un ou plusieurs attributs à gauche de la contrainte existante. Cependant, le processus de sélection des attributs n'est pas détaillé de manière

exhaustive dans ces articles, ce qui constitue une limitation. Bien qu'un des articles propose trois techniques différentes pour effectuer ce choix [Chiang et al. 2011]cependant cet approche est limitée à l'ajout d'un seul attribut.

Une autre limite à prendre en compte est que la réparation des contraintes peut également se faire en remplaçant des attributs existants.

Il faut également prendre en compte que plusieurs propositions de contraintes peuvent être générées en utilisant l'ajout ou la suppression mais cela n'implique pas nécessairement qu'elles soient valides.

En résumé, les articles fournissent des contributions significatives en matière de réparation des contraintes en utilisant des techniques d'ajout et de suppression d'attributs. Cependant, des améliorations peuvent être apportées en détaillant davantage le processus de sélection des attributs, en explorant la possibilité d'effectuer la suppression et le remplacement des attributs des deux côtés de la contrainte, et en proposant des méthodes pour choisir entre différents ensembles de contraintes résultants. Ces développements futurs permettraient de renforcer l'efficacité des approches de réparation des contraintes basées sur l'ajout et la suppression d'attributs.

1.10 Réparation hybride

Contrairement aux méthodes traditionnelles qui se concentrent uniquement sur la réparation des erreurs dans les données ou sur la réparation des règles de qualité, des études récentes mettent également en évidence la présence simultanée d'erreurs dans les données et dans les règles de qualité, et elles travaillent à la réparation à la fois des règles et des données simultanément.

1.10.1 Étude bibliographique

Dans cette partie, nous allons détailler les différents travaux qui traitent la réparation hybride.

1.10.1.1 On the Relative Trust between Inconsistent Data and Inaccurate Constraints (George Beskales ,Ihab F. Ilyas , Lukasz Golab , Artur Galiullin) [Beskales et al. 2013]

L'article propose un algorithme de réparation des données et des règles (DFs) basé sur le principe de confiance relative exprimée sous forme d'un seuil. Ce seuil limite le nombre maximal de modifications autorisées dans les données, ce qui devient crucial compte tenu de l'incertitude quant à la fiabilité des données par rapport aux contraintes.

L'approche algorithmique présentée dans l'article génère plusieurs suggestions de modifications des données et/ou des contraintes afin d'atteindre la cohérence. Pour cela, l'article introduit le concept de réparations minimales et utilise le seuil de confiance relatif pour déterminer le niveau de confiance accordé aux données par rapport aux contraintes.

De plus, l'article explore l'espace des réparations possibles en considérant différentes combinaisons de modifications des données et des contraintes. Ces réparations sont générées en optimisant la minimisation des changements tout en respectant le seuil de confiance spécifié.

Ainsi, l'article fournit plusieurs suggestions de réparations correspondant à différents niveaux de confiance relative. Cela permet aux utilisateurs de choisir la meilleure approche pour résoudre les inconsistances dans leurs données, en fonction de leur niveau de confiance souhaité.

1.10.1.2 Constraint-Variance Tolerant Data Repairing (Shaoxu Song Han Zhu Jianmin Wang) [Shaoxu Song 2016]

Cet algorithme propose une méthode de réparation en générant différentes variations pour chaque contrainte et en choisissant celle qui minimise les réparations des données nécessaires.

Pour commencer, l'article explique comment l'algorithme génère plusieurs variations pour chaque contrainte. L'algorithme évalue chaque contrainte en générant différentes variantes qui respectent les contraintes de base mais présentent des différences mineures.

Ensuite, l'article présente un graphe de conflit pour représenter les violations de contraintes. Chaque violation est représentée par une hyperarête dans le graphe, où chaque arête correspond à une liste de tuples qui violent une contrainte. Les coûts de réparation des données sont calculés en utilisant une couverture minimale des sommets du graphe de conflit. Cela signifie que l'algorithme identifie les violations les plus importantes et cherche à les réparer de manière optimale en minimisant les coûts associés.

L'algorithme examine ensuite un ensemble de variantes de contraintes. Pour chaque variante, il calcule la borne inférieure du coût de réparation. Si cette borne inférieure est inférieure à la borne supérieure actuellement connue, la réparation des données pour la variante est effectuée. L'algorithme met à jour la borne supérieure lorsqu'il trouve un résultat amélioré, ce qui lui permet de converger vers la meilleure solution possible.

En résumé, cet algorithme génère des variations maximales de contraintes et utilise un graphe de conflit pour évaluer les violations et calculer les coûts de réparation des données. Il choisit ensuite la variante de contrainte qui permet d'obtenir une réparation de données minimale, en minimisant les coûts associés à ces réparations.

1.10.2 Synthèse

Les deux articles présentent des approches pour la réparation des données et des règles dans un modèle de données, en mettant l'accent sur la minimisation des coûts et la maximisation de l'efficacité.

ces articles traitent de la modification simultanée des données et des règles. Une idée est d'utiliser des seuils de confiance pour limiter le coût des modifications permettant d'obtenir un bon compromis entre des règles trop simplifiées et des règles trop raffinées. Cependant, cette approche présente un inconvénient, car la performance dépend forte-

ment du seuil fixé et de l'intervention humaine pour le fixer. L'autre approche qui peut être plus pertinente repose sur la génération de variations maximales de contraintes, qui peuvent être considérées comme des réparations déjà générées en utilisant une réparation des contraintes seules, et faire la réparation des données nécessaire pour chacune des contraintes générées. En choisissant la contrainte générée qui minimise la réparation des données.

1.11 Étude comparative entre les approches de détection et réparation des anomalies

Travaux	Détection des Anomalies	Identification de la Source d'Anomalie	Réparation des Données	Réparation des Règles de Qualité	Réparation Hybride	Intervention Humaine
(Chiang et al. 2011)	-	Après la réparation en calculant le coût minimal	A gauche ou à droite et en remplaçant les motifs par des signatures.	Ajout d'un attribut à gauche de la DF	-	Vérification des réparations appliquées
(Volkovs et al. 2014)	Construction de patterns en regroupant les motifs de violation	Avant la réparation par un classifieur de prédiction de la source d'anomalie	A gauche ou à droite en effectuant plusieurs étapes.	Ajout d'un ou plusieurs attributs à gauche de la DF	-	Validation des réparations proposées
(Shaoxu Song 2016)	Regroupement de violation et représentation par graphe de conflit	-	A gauche ou à droite en une seule étape	Ajout ou suppression d'un attribut à gauche de la DF	Génération de variations maximales des contraintes et ajustement des données	Indication du seuil de différence entre la contrainte erronée et sa réparation.
(Beskales et al. 2013)	-	-	A gauche ou à droite en limitant le nombre de réparations	Ajout d'un ou plusieurs attributs à gauche de la DF	utilisation d'un seuil de confiance pour contrôler les réparations	Sélection de réparation et indication du seuil de confiance

Tab. 1.2 : Table de comparaison entre les approches de détection et réparation des anomalies des données [Meflah,2023].

1.11.1 positionnement

Après l'étude comparative effectuée, nous préconisons de suivre l'approche de détection des anomalies telle qu'elle est présentée dans tous les articles, car elle offre une base solide pour identifier les données problématiques en utilisant les règles de qualité.

En ce qui concerne l'identification de la source de l'anomalie, nous suggérons de la traiter comme indiqué dans l'article [Volkovs et al. 2014] en tant que tâche de classification, en ajoutant d'autres classifieurs et techniques avancées comme le GAN pour améliorer la précision de cette classification.

Pour la réparation des règles de qualité, nous envisageons de développer une méthode pour le choix des attributs à manipuler, vue qu'elle n'est pas spécifiée explicitement dans les articles étudiés. Nous envisageons d'utiliser d'autres transformations en plus de celles citées, notamment le remplacement et la suppression des deux côtés. De plus, nous souhaitons consolider l'aspect de validité des règles, c'est pourquoi nous ajoutons des tests de validité pour les transformations générées, avec l'objectif de rendre cette démarche entièrement automatique, contrairement aux articles étudiés.

Enfin, nous suggérons d'exploiter la réparation des contraintes et des données pour mettre en œuvre notre propre solution pour la réparation hybride en adoptant l'approche de réparation des données à droite décrite dans les articles.

1.12 Conclusion

Assurer la qualité des données est une tâche difficile mais essentielle dans le domaine des données. La capacité à détecter les anomalies et à les réparer devient de plus en plus précieuse à mesure que le volume de données augmente.

Dans ce chapitre, nous avons exploré les concepts liés aux données, discuté de la qualité des données et exposé le processus de gestion de la qualité des données en passant en revue un ensemble d'approches spécialisées dans l'identification et la réparation des anomalies. Cette étude nous a permis de comprendre les avantages et les limites propres à chaque approche, ainsi que de les comparer entre elles.

Chapitre 2

Conception

2.1 Introduction

Dans ce chapitre, nous allons proposer notre propre solution pour l'identification de différents types d'anomalies dans les données et les règles et par la suite leur réparation. Nous allons concevoir une solution pour l'identification de la source de l'anomalie, capable d'extraire les différentes violations et de prédire la source de l'anomalie détectée. S'il s'agit d'une anomalie de contrainte, nous indiquerons la contrainte violée. S'il s'agit d'une anomalie de données, nous indiquerons les lignes de données violant la contrainte en question. Ainsi que les lignes de données erronées et la contrainte erronée en en cas d'anomalie simultanée dans les deux.

Nous allons proposer une approche pour la réparation des contraintes, qui permettra d'effectuer différentes transformations sur la contrainte erronée tout en choisissant les attributs à manipuler intelligemment. Ainsi, cette approche doit garantir la validité des transformations proposées et choisir la transformation qui minimise le coût de changement.

Nous allons également concevoir une approche de réparation hybride pour réparer les erreurs dans les données et les contraintes simultanément en trouvant une bonne combinaison entre notre approche de réparation des contraintes et l'approche de réparation des données choisie. Nous tenons à rappeller que la réparation des données est en dehors de la portée de notre travail.

Nous avons divisé notre solution en trois parties détaillées ci-dessous.

2.2 Identification de la source d'anomalie

2.2.1 Approche générale de la solution

Avant de procéder à la réparation des anomalies il est impératif de détecter leurs sources afin d'apporter les réparations adéquates à chaque source d'anomalie.

Notre processus d'identification de la source d'anomalie prend comme entrée l'ensemble de données et l'ensemble des règles de qualité existantes. et Il se divise en quatres étapes :

La première étape de notre processus vise à identifier les anomalies au sein de l'ensemble de données. Ces anomalies sont définies comme des violations des règles de qualité préalablement établies. Toute donnée qui ne respecte pas une règle de qualité est considérée comme une anomalie. À cette étape, chaque ensemble de données anormales est associé à la règle de qualité qu'il viole.

Après avoir identifié les anomalies, la deuxième étape consiste à extraire les caractéristiques spécifiques de chaque anomalie détectée. Ces caractéristiques sont calculées en analysant les données qui violent chaque règle de qualité.

Dans la troisième étape, nous créons des classifieurs supervisés pour catégoriser les anomalies. Les caractéristiques extraites à l'étape précédente servent d'entrées pour ces classifieurs. De manière innovante, nous intégrons également un réseau GAN (Generative Adversarial Network) dans cette étape. Le GAN génère des données artificielles ressem-

blant à des anomalies potentielles, qui sont ensuite combinées avec les données réelles pour l'entraînement des classifieurs.

Dans la quatrième étape, les classifieurs supervisés formés dans l'étape précédente sont utilisés pour prédire les classes d'anomalies pour de nouvelles données en utilisant les caractéristiques extraites. Ces classes comprennent les erreurs dans les données, les erreurs dans les règles, et les erreurs combinant les deux.Nous optons ensuite pour le modèle de classifieur dont les performances semblaient être les meilleures par les évaluations à effectuer.

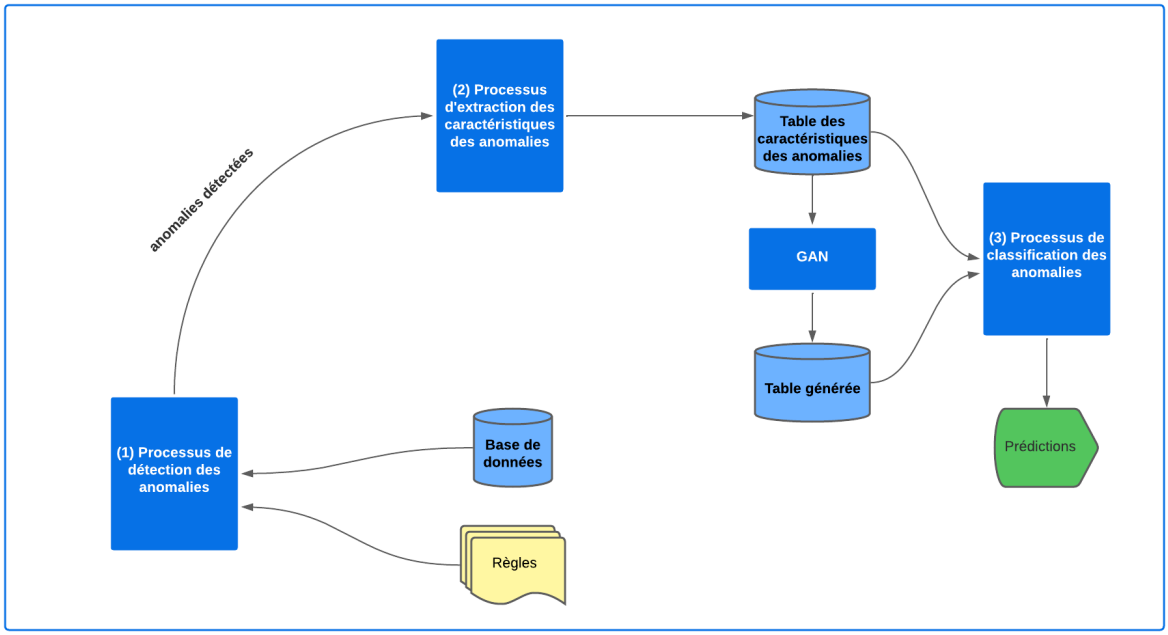


FIG. 2.1 : Schéma général de l'identification de la source d'anomalie

2.2.2 Etape 1: Détection des anomalies

La détection des violations de contraintes est le processus par lequel on identifie les enregistrements ou les tuples dans un ensemble de données qui ne respectent pas les contraintes spécifiées.[SHAOXU SONG 2016].

[VOLKOVS et al. 2014] définit une notion appellée motif qui est un ensemble de lignes spécifiques dans les données, obtenu en choisissant certaines colonnes à partir de toutes les lignes d'une relation.

Ensuite, il explique que pour détecter les violations de contraintes, on compare ces motifs. Si on trouve deux motifs distincts qui ont la même sélection de colonnes gauches mais des données différentes dans les colonnes droites, alors on considère que c'est une violation. Cela signifie que la règle qui devrait être respectée n'est pas satisfaite, car les données ne correspondent pas comme elles le devraient.

Pour notre processus, nous avons opté pour la même méthode utilisée par [VOLKOVS et al. 2014] afin de détecter les anomalies sous forme de motifs partageant la même règle violée.Nous sélectionnons les colonnes des motifs de manière à ce qu'elles correspondent à chaque règle de qualité violée.

2.2.3 Etape 2: Extraction des caractéristiques des anomalies

Afin de mieux comprendre la nature des anomalies identifiées, il est essentiel d'extraire des informations telles que les valeurs en question, les relations entre les données concernées et les règles violées, sous forme de caractéristiques décrivant chaque anomalie.

[Chiang et al. 2011] a définis un ensemble de mesures quantitatives qui permettent de caractériser les écarts entre les distributions statistiques des données source et cible.

Il a présenté les caractéristiques sous forme de statistiques suivantes :

- **Statistiques sur les FD F**
  - Nombre de tuples violés dans F par rapport au nombre total de tuples dans I : g1 = NF / N.
- **Statistiques sur un motif p d'une FD F**
  - Fréquence de p par rapport au nombre total de tuples : g2 = freq(p) / N.
  - Fréquence de p par rapport au nombre total de tuples violés pour F. g3 = freq(p) / NF
  - **Statistiques sur le motif p de F à travers un ensemble de FDs** Pour une FD F0 F, le recouvrement de p est défini comme la fraction de tuples qui violent F' et contiennent le motif p dans leurs attributs XA. Formellement, overlap(F', p) = |t | t I F', ΠXA(t) p| / freq(p). Nous calculons des statistiques descriptives pour toutes les FDs de F.
    - g4 = moyenne(overlap(F', p) F' F).
    - g5 = écart-type(overlap(F', p) F' F).
    - g6 = maximum(overlap(F', p) F' F).
    - g7 = minimum(overlap(F', p)F' F).

Nous avons proposé de calculer ces statistiques et de considérer les résultats par la suite comme des attributs supplémentaires pour l'ensemble de données.

2.2.4 Etape 3: Entrainement de classifieurs supervisés avec l'utilisation d'un GAN

Afin de pouvoir classifier la source des anomalies détectées en prenant en compte les statistiques calculées dans l'étape précédente, nous introduisons maintenant un ensemble de modèles de classifieurs supervisés dans le but de prédire la classe de source d'une anomalie, à savoir : les erreurs dans les données, les erreurs dans la règle de qualité, ou les erreurs dans les données et la règle simultanément.

[Kumar et al. 2017] a défini un ensemble de modèles de classification supervisée comme suit :

- **SVM (Machine à Vecteurs de Support) :** Les SVM sont des modèles d'apprentissage supervisé utilisés pour la classification et la régression. Ils cherchent à trouver des hyperplans optimeaux qui sépare les données en classes de manière à maximiser la marge entre les différentes classes. Les données d'entraînement les plus proches de cet hyperplan sont appelées vecteurs de support. Les SVM peuvent également utiliser des noyaux pour gérer des données non linéaires en projetant les données dans un espace de dimension supérieure.

- **Arbre de Décision :**Un arbre de décision est un modèle d'apprentissage supervisé qui divise récursivement les données d'entrée en sous-groupes en fonction des caractéristiques, en choisissant à chaque étape la caractéristique la plus discriminante. Cela crée une structure arborescente où chaque feuille de l'arbre représente une classe ou une valeur de sortie. L'arbre est utilisé pour prendre des décisions en suivant les branches de haut en bas en fonction des caractéristiques d'entrée.

- **Forêt Aléatoire :** Une forêt aléatoire est un modèle d'ensemble d'arbres de décision. Elle combine plusieurs arbres de décision pour réduire le surapprentissage et améliorer la généralisation. Chaque arbre est formé sur un sous-ensemble aléatoire des données d'entraînement et des caractéristiques. Lors de la prédiction, chaque arbre donne un vote pour la classe cible, et la classe majoritaire est généralement choisie comme prédiction finale.

- **Régression Logistique :** La régression logistique est un modèle d'apprentissage supervisé utilisé pour la classification. Il est basé sur la fonction logistique et est principalement utilisé pour estimer la probabilité qu'une instance appartienne à une classe particulière. Le modèle ajuste des poids aux caractéristiques d'entrée et utilise une fonction logistique pour calculer la probabilité de classe, généralement avec une frontière de décision seuillée à 0,5.

Nous entraînons ensuite chacun de ces classifieurs sur notre ensemble de données ayant comme propriétés les statistiques déjà calculées. Afin d'améliorer les performances des modèles de classification que nous avons envisagé d'utiliser, nous proposons de combiner l'ensemble de données réel avec des données synthétiques. Cette approche permet aux modèles de s'entraîner sur un ensemble de données plus diversifié, améliorant ainsi leur capacité à généraliser et à classifier correctement de nouvelles données. À cette fin, nous utilisons un GAN (Generative Adversarial Network).

[Creswell et al. 2017] a décrit le fonctionnemnt d'un GAN comme suit : Un GAN se compose de deux parties, un générateur et un discriminateur.

Le générateur prend du bruit aléatoire en entrée et génère des données synthétiques, tandis que le discriminateur prend des données réelles et synthétiques en entrée et essaie de les distinguer. Les deux réseaux sont formés de manière antagoniste, où le générateur essaie de tromper le discriminateur en générant des données de plus en plus réalistes, tandis que le discriminateur essaie de devenir de plus en plus précis pour distinguer les données réelles des données générées.

Une fois que le GAN est correctement formé, le générateur peut être utilisé pour produire des données synthétiques qui ressemblent à celles du jeu de données d'entraînement.

Ces données synthétiques peuvent être utilisées pour augmenter la taille du jeu de données d'entraînement.

À l'issue de cette étape, nous obtenons le résultat de nos quatre classifieurs entraînés sur un ensemble de données contenant à la fois les données réelles et les données générées par le GAN.

2.2.5 Etape 4: Prédiction des classes d'anomalies

À cette étape, nous procédons à la détection de la source des anomalies en utilisant les classifieurs que nous avons entraînés sur l'ensemble de données combinant à la fois les données réelles et celles générées par le GAN. Nous testons ces classifieurs sur un ensemble de données contenant différents types d'anomalies, notamment des erreurs de données, des violations de règles et des combinaisons des deux. Ensuite, nous sélectionnons le classifieur qui présente les meilleures performances afin de l'adopter pour notre processus d'identification de la source des anomalies.

2.3 Réparation des règles de qualité

2.3.1 Approche générale de la solution

Après avoir identifié la source de l'anomalie détectée, s'il s'agit d'erreurs dans la règle de qualité, il devient impératif d'aborder la question de la réparation de cette règle de qualité.

Notre processus de réparation des règles de qualité prend comme entrée l'ensemble de données, l'ensemble des règles de qualité existantes, et la règle détéctée erronée par le processus d'identification de la source d'anomalie. Il se divise en trois étapes :

La première étape consiste à générer plusieurs contraintes dérivées de la contrainte erronée en effectuant certaines transformations sur celle-ci. Les transformations proposées sont :

- l'ajout d'un ou plusieurs attributs à gauche.
- la suppression d'un ou plusieurs attributs à gauche ou à droite.
- le remplacement d'un attribut à gauche ou à droite.

Ces transformations ne sont pas réalisées de manière aléatoire, mais plutôt en étudiant les différentes dépendances entre les attributs de l'ensemble de données en utilisant une mesure basée sur l'information mutuelle.

La deuxième étape consiste à filtrer les contraintes générées afin de ne conserver que celles qui n'existent pas déjà parmi les contraintes données, tout en veillant à ce que les contraintes restantes soient valides en testant leurs satisfactions par l'ensemble des données.

Une troisième étape alternative se présente lorsque plusieurs contraintes ont été évaluées comme valides. Dans ce cas, notre approche consiste à sélectionner la réparation qui engendre des changements minimales par rapport à la contrainte erronée. Pour cela, nous proposons trois altérnatives pour effectuer ce choix :

La première altérnative consiste à calculer le coefficient de Jaccard entre chaque contrainte candidate et la contrainte erronée, afin de sélectionner celle qui présente le coefficient de Jaccard le plus élevé, indiquant ainsi la plus grande similarité avec la contrainte erronée.

la deuxième altérnative consiste à calculer la distance de Levenshtein entre chaque contrainte candidate et la contrainte erronée, afin de sélectionner celle qui présente la distance de Levenshtein la plus minimale, indiquant ainsi le plus petit nombre d'opérations pour la transformation d'une contrainte en une autre.

La troisième altérnative intervient en cas de possibilité d'intégrer l'utilisateur pour qu'il choisisse la contrainte qu'il juge la plus pertinente selon son expérience. Le schéma suivant résume le processus général de notre solution de réparation des contraintes.

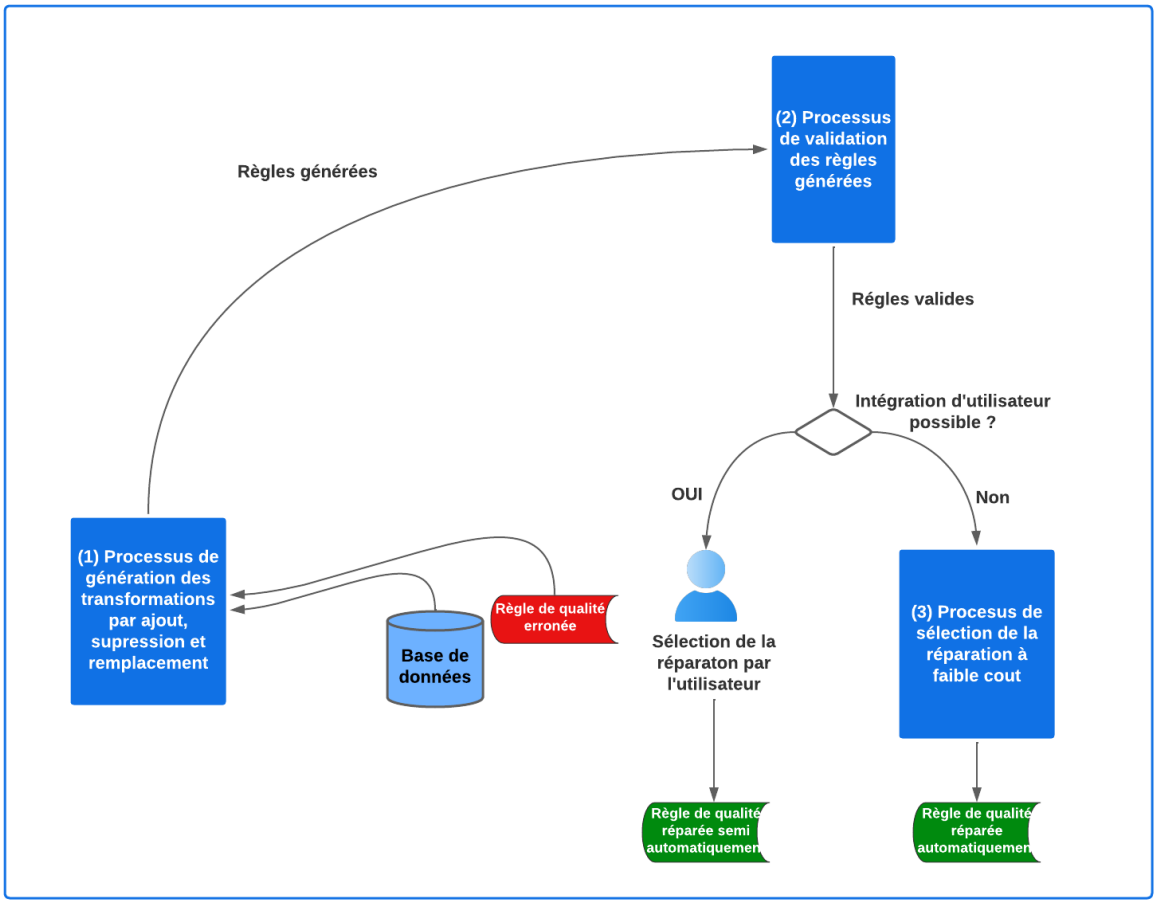


Fig. 2.2 : Schéma général de la réparation des règles de qualité

2.3.2 Etape 1: Génération des différentes contraintes candidates

Les contraintes peuvent être réparées si elles sont trop simplifiées en ne prennat pas en compte toutes les relations réelles entre les données. Dans ce cas, la réparation vise à modifier ces contraintes en insérant davantage d'attributs supplémentaires issus de l'espace des attributs de l'ensemble de données aux cotés gauches des contraintes. Elles peuvent également être réparées si elles sont sur entraînées sur les données. Dans ce cas, la réparation consiste à modifier ces contraintes en supprimant les attributs inutiles des deux cotés. [Shaoxu Song 2016].

Pour effectuer ces modifications, le choix des attributs à manipuler est crucial, afin de comprendre ce choix nous avons étudié comment les attributs sont choisis dans le processus de la découverte automatique des dépendances fonctionnelles.

Les dépendances fonctionnelles sont construites en étudiant les liens de dépendances entre différents attributs au sein d'une base de données. Cela se fait en observant comment certaines valeurs d'attributs dépendent d'autres valeurs d'attributs. Si ces dépendances se produisent de manière cohérente, on peut conclure qu'il existe une relation de dépendance entre ces attributs. [Liu et al. 2012].

[Rakotomalala 2020] étudie les liens de dépendances entre les attributs au sein d'une base de données. Il présente une mesure basée sur l'information mutuelle. Il définit l'information mutuelle comme une mesure de la quantité d'information qu'une variable contient sur une autre. Il détaille le processus du calcul de la mesure d'information mutuelle entre deux attributs comme suit :

- **Calcul des distributions de probabilité :** Pour des données qui peuvent être représentées sous forme de tableau ou de matrice, où chaque ligne correspond à une instance et chaque colonne représente un attribut, il faut calculer les distributions de probabilité des attributs concernés. Cela implique de déterminer la fréquence de chaque valeur d'attribut dans l'ensemble de données. Par exemple, si nous avons un attribut A avec les valeurs A1, A2, A3, nous devons calculer la probabilité de chaque valeur P(A1), P(A2), et P(A3).
- **Calcul de l'entropie :** L'entropie est une mesure de l'incertitude dans une distribution de probabilité. Pour calculer l'entropie d'un attribut X, on utilise la formule de l'entropie de Shannon : H(X) = - P(X=x) * log2(P(X=x)) où P(X=x) est la probabilité d'observer une valeur spécifique x de l'attribut X.
- **Calcul de l'entropie conditionnelle :** Pour calculer l'entropie conditionnelle H(Y|X), qui représente la quantité d'information que l'attribut X apporte sur l'attribut Y, on utilise la formule suivante : H(Y|X) = P(X=x) * H(Y|X=x) où P(X=x) est la probabilité d'observer la valeur x de l'attribut X, et H(Y|X=x) est l'entropie conditionnelle de l'attribut Y sachant la valeur x de l'attribut X.
- **Calcul de l'information mutuelle :** L'information mutuelle entre les attributs X et Y est définie comme la différence entre l'entropie de Y et l'entropie conditionnelle de Y sachant que X : I(X ;Y) = H(Y) - H(Y|X) L'information mutuelle mesure la réduction d'incertitude dans Y en connaissant X. Plus l'information mutuelle est élevée, plus les deux attributs sont dépendants.

- **Calcul de l'information mutuelle normalisée :** Pour obtenir des valeurs d'informations mutuelles comprises entre 0 et 1, la formule est la suivante :

$$\text{Information Mutuelle Normalisée} = \frac{I(X;Y)}{\sqrt{H(X) \cdot H(Y)}}$$

Dans notre solution, nous utilisons la mesure d'information mutuelle normalisée afin de sélectionner parmi les attributs de l'espace des attributs de l'ensemble de données, ceux qui sont étroitement liés aux attributs droits de contraintes. Cela nous permet de les ajouter à la partie gauche de la contrainte. Ainsi, nous utilisons cette mesure pour sélectionner parmi les attributs du côté droit de la contrainte , ceux qui ont une faible corrélation avec les attributs du côté gauche de la contrainte. Cela nous permet de les exclure de la partie droite de la contrainte.

De même, en supprimant parmi les attributs du côté gauche de la contrainte ceux qui présentent une faible corrélation avec les attributs du côté droit. De plus, nous combinons l'ajout et la suppression afin de permettre en outre le remplacement des attributs par d'autres.

Nous détaillons ces modifications sur une contrainte erronée comme suit :

2.3.2.1 Ajout d'attributs à la partie gauche :

Selon [Beskales et al. 2013] , il n'est pas pertinent d'ajouter à la partie gauche d'une contrainte des attributs présents dans sa partie droite, afin de ne pas produire des contraintes triviales. Ainsi, la première étape dans notre processus d'ajout d'attributs à la partie gauche consiste à réduire l'espace de recherche des attributs à ajouter en éliminant ceux déjà présents dans la contrainte.

Ensuite, notre processus de sélection d'un attribut à ajouter à la partie gauche d'une contrainte consiste à calculer l'information mutuelle pour chaque attribut de l'espace de recherche avec chacun des attributs de la partie droite de la contrainte. Par la suite, afin de pouvoir comparer les résultats obtenus pour chaque attribut candidat, nous calculons la moyenne des résultats des calculs d'information mutuelle obtenus pour chacun d'entre eux. Les attributs candidats sont classés en fonction de leurs moyennes d'information mutuelle calculée, et l'attribut ayant la moyenne la plus élevée est sélectionné pour l'ajouter à la partie gauche de la contrainte . En cas d'égalité de moyennes, toutes les attributs ayant cette valeur de moyenne sont choisis. Des contraintes générées sont construites en ajoutant à chacune un de ces attributs sélectionnés.

[Volkovs et al. 2014] souligne que pour couvrir plus d'anomalies dans les contraintes, il est plus pertinent de pouvoir ajouter non pas un seul attribut à la partie gauche, mais un ensemble d'attributs. Afin d'étendre notre processus de modification de contrainte par l'ajout à gauche avec cette possibilité, nous choisissons, parmi les attributs candidats, un ensemble d'attributs ayant les moyennes des informations mutuelles les plus élevées. La taille de cet ensemble varie de deux jusqu'au nombre limite des attributs à ajouter.

[Volkovs et al. 2014] permet à l'utilisateur de définir ce nombre limite. Cependant, dans notre processus et pour le maintenir entièrement automatique, nous proposons le nombre limite d'attributs à ajouter à la partie gauche comme étant le nombre total d'attributs de l'espace, en excluant le nombre d'attributs déjà présents dans la contrainte. Ceci

vise à maximiser le nombre d'attributs à ajouter tout en éliminant les attributs inutiles.

Voici l'algorithme détaillant la modification par ajout des attributs à la partie gauche de la contrainte qu'on a proposé :

Algorithm 1 : ajouter_attributs_gauche(df, contrainte_erronee, n)

```
   Data : df : dataframe, contrainte_erronee : contrainte, n : entier
   Result : nouvelles_contraintes : liste
 1 if n est None then
 2 |   n ← taille de attributs_restants
 3 end
 4 foreach attribut_r dans attributs_restants do
 5 |   info_mutuelle_attr_r ← Liste vide
 6 |   foreach attribut_d dans attributs_droite do
 7 |   |   data_d ← données correspondant à attribut_d dans df
 8 |   |   data_r ← données correspondant à attribut_r dans df
 9 |   |   info_mutuelle ← CalculerInformationMutuelle (data_d, data_r)
10 |   |   ajouter info_mutuelle à info_mutuelle_attr_r
11 |   end
12 |   ajouter info_mutuelle_attr_r à info_mutuelles
13 end
14 foreach info_mutuelle_attr_r dans info_mutuelles do
15 |   moyenne_info_mutuelle ← Moyenne de info_mutuelle_attr_r
16 |   ajouter moyenne_info_mutuelle à moyennes_info_mutuelles
17 end
18 foreach attribut_r dans attributs_restants do
19 |   ajouter attribut_r à resultats_tries en utilisant moyennes_info_mutuelles
   |    pour trier par ordre décroissant
20 end
21 attributs_gauche_liste ← Convertir attributs_gauche en liste
22 foreach attribut dans resultats_tries do
23 |   if moyennes_info_mutuelles[attributs_gauche_liste.index(attr)] est égal à
   |    moyennes_info_mutuelles[0] then
24 |   |   nouvelle_contrainte ← Construire la nouvelle contrainte en ajoutant
   |   |    attribut au côté gauche de contrainte_erronee
25 |   |   ajouter nouvelle_contrainte à nouvelles_contraintes
26 |   end
27 end
28 if n > 1 then
29 |   for j de 2 à n do
30 |   |   attributs_sup_jeme_moy ← Liste des attributs ayant une moyenne
   |   |    d'information mutuelle supérieure ou égale à
   |   |    moyennes_info_mutuelles[j-1]
31 |   |   combinaisons_attributs ← GenererCombinaisons
   |   |    (attributs_sup_jeme_moy, j)
32 |   |   foreach combinaison dans combinaisons_attributs do
33 |   |   |   nouvelle_contrainte ← Construire la nouvelle contrainte en ajoutant
   |   |   |    la combinaison au côté gauche de contrainte_erronee
34 |   |   |   ajouter nouvelle_contrainte à nouvelles_contraintes
35 |   |   end
36 |   end
37 end
38 return nouvelles_contraintes
```

2.3.2.2 Suppression d'attributs de la partie droite : Pour cette modification, nous utilisons la même logique qu'on a proposé pour le processus de modification des contraintes par l'ajout à gauche, basée sur le choix des attributs en fonction du calcul de l'information mutuelle.

Cette fois-ci , notre processus de sélection d'un attribut à supprimer de la partie droite d'une contrainte consiste à calculer l'information mutuelle pour chaque attribut de la partie droite avec chacun des attributs de la partie gauche de la contrainte. Par la suite, afin de pouvoir comparer les résultats obtenus pour chaque attribut de la partie droite, nous calculons la moyenne des résultats des calculs d'information mutuelle obtenus pour chacun d'entre eux. Les attributs de la partie droite sont classés en fonction de leurs moyennes d'information mutuelle calculées, et l'attribut ayant la moyenne la plus faible est sélectionné pour être supprimé de la partie droite de la contrainte. En cas d'égalité des moyennes, plusieurs attributs de la partie droite peuvent être choisis. Des contraintes générées sont construites en supprimant un de ces attributs sélectionnés dans chacune d'elles.

Afin de permettre la suppression de plusieurs attributs de la partie droite lorsque le nombre d'attributs de cette partie dépasse deux, nous sélectionnons, parmi les attributs de la partie droite, un ensemble d'attributs ayant les moyennes d'informations mutuelles les plus faibles. La taille de cet ensemble varie de deux jusqu'au nombre limite d'attributs à supprimer. Afin de ne pas supprimer tous les attributs de la partie droite, ce qui rendrait la contrainte mal définie, nous définissons le nombre maximal d'attributs à supprimer comme étant le nombre d'attributs de la partie droite moins un.

Voici l'algorithme détaillant la modification par suppression de la partie droite de la contrainte qu'on a proposé :

Algorithm 2 : supprimer_attributs_droite(df, contrainte_erronee, n)

```
   Data : df : dataframe, contrainte_erronee : contrainte, n : entier
   Result : nouvelles_contraintes : liste
 1 if n est None then
 2 |  n ← taille de attributs_droite
 3 end
 4 else
 5 |  foreach attribut_r dans attributs_droite do
 6 |  |  info_mutuelle_attr_r ← Liste vide
 7 |  |  foreach attribut_d dans attributs_gauche do
 8 |  |  |  data_d ← données correspondant à attribut_d dans df
 9 |  |  |  data_r ← données correspondant à attribut_r dans df
10 |  |  |  info_mutuelle ← CalculerInformationMutuelle (data_d, data_r)
11 |  |  |  ajouter info_mutuelle à info_mutuelle_attr_r
12 |  |  end
13 |  |  ajouter info_mutuelle_attr_r à info_mutuelles
14 |  end
15 |  foreach info_mutuelle_attr_r dans info_mutuelles do
16 |  |  moyenne_info_mutuelle ← Moyenne de info_mutuelle_attr_r
17 |  |  ajouter moyenne_info_mutuelle à moyennes_info_mutuelles
18 |  end
19 |  foreach attribut_r dans attributs_droite do
20 |  |  ajouter attribut_r à resultats_tries en utilisant moyennes_info_mutuelles
   |  |   pour trier par ordre croissant
21 |  end
22 |  foreach attribut dans resultats_tries do
23 |  |  if moyennes_info_mutuelles[0] est égal à
   |  |   moyennes_info_mutuelles[attributs_droite_liste.index(attr)] then
24 |  |  |  attributs_droite_liste_copy ← Copie de attributs_droite_liste
25 |  |  |  Supprimer attribut de attributs_droite_liste_copy
26 |  |  |  nouvelle_contrainte ← Construire la nouvelle contrainte en utilisant
   |  |  |   attributs_droite_liste_copy
27 |  |  |  ajouter nouvelle_contrainte à nouvelles_contraintes
28 |  |  end
29 |  end
30 |  if n > 1 then
31 |  |  for j de 2 à n do
32 |  |  |  attributsInfMoy← Liste des attributs ayant une moyenne
   |  |  |   d'information mutuelle inférieure ou égale à
   |  |  |   moyennes_info_mutuelles[j-1]
33 |  |  |  combinaisons_attributs ← GenererCombinaisons (attributsInfMoy, j)
34 |  |  |  foreach combinaison dans combinaisons_attributs do
35 |  |  |  |  nouvelle_contrainte ← Construire la nouvelle contrainte en
   |  |  |  |   enlevant les attributs de la combinaison du côté droit
36 |  |  |  |  ajouter nouvelle_contrainte à nouvelles_contraintes
37 |  |  |  end
38 |  |  end
39 |  end
40 |  return nouvelles_contraintes
41 end
```

2.3.2.3 Suppression d'attributs de la partie gauche : En suivant le même processus de suppression d'attributs de la partie droite de la contrainte, cette fois-ci, le processus supprime les attributs à gauche ayant les informations mutuelles les plus faibles.

2.3.2.4 Remplacement d'un attribut de la partie gauche : Pour effectuer cette modification, nous combinons le processus de suppression d'attributs de la partie gauche avec le processus d'ajout d'attributs à la partie gauche de la contrainte. Tout d'abord, nous spécifions pour chacun de ces deux processus le nombre maximal d'attributs à ajouter à supprimer à un seul attribut. Par la suite, l'attribut du côté gauche ayant la moyenne des informations mutuelles la plus faible sera supprimé et remplacé par l'attribut de l'espace de données ayant la plus grande valeur de moyenne des informations mutuelles. Voici l'algorithme détaillant les étapes de cette modification :

Algorithm 3 : remplacer_attribut_gauche(df, contrainte_erronee)

Data : df : dataframe, contrainte_erronee : contrainte
Result : nouvelles_contraintes : liste
1 nouvelles_contraintes_supprimees ← `supprimer_attributs_gauche` (df, contrainte_erronee, 1)
2 nouvelles_contraintes ← Liste vide
3 **foreach** *nouvelle_contrainte_supprimee dans nouvelles_contraintes_supprimees* **do**
4 | nouvelles_contraintes_ajoutees ← `ajouter_attributs_gauche` (df, nouvelle_contrainte_supprimee, 1)
5 | ajouter toutes les nouvelles_contraintes_ajoutees à nouvelles_contraintes
6 **end**
7 **return** nouvelles_contraintes

2.3.2.5 Remplacement d'un attribut de la partie droite : En suivant le même processus de remplacement d'un attribut de la partie gauche de la contrainte, cette fois-ci, le processus supprime l'attribut à droite ayant la moyenne des informations mutuelles la plus faible et le remplace par l'attribut de l'espace de données ayant la plus grande valeur de moyenne des informations mutuelles. Afin de permettre ce remplacement, on ajoute un processus pour l'ajout d'un attribut à la partie droite de la contrainte, qui a une logique similaire à l'ajout d'attributs à gauche de la contrainte, tout en mesurant cette fois-ci l'information mutuelle par rapport aux attributs de gauche de la contrainte et en limitant le nombre d'attributs à ajouter à la partie droite à un seul.

En combinant ces différentes modifications, on peut générer une liste exhaustive de toutes les contraintes potentielles qui pourraient réparer la contrainte erronée. Chaque modification est appliquée séquentiellement à la contrainte erronée, ce qui permet d'obtenir une liste de nouvelles contraintes candidates.

L'algorithme suivant décrit notre processus complet de génération des différentes contraintes candidates :

Algorithm 4 : GenererModificationsContraintes(dataframe, contrainte_erronee)

Data : dataframe : dataframe, contrainte_erronee : contrainte
Result : contraintes_generees : liste

```
1  contraintes_generees ← Liste vide
2  ajouter_attributs_gauche_resultat ← ajouter_attributs_gauche (dataframe,
     contrainte_erronee, None)
3  ajouter toutes les contraintes de ajouter_attributs_gauche_resultat à
     contraintes_generees
4  supprimer_attributs_droite_resultat ← supprimer_attributs_droite
     (dataframe, contrainte_erronee, None)
5  ajouter toutes les contraintes de supprimer_attributs_droite_resultat à
     contraintes_generees
6  supprimer_attributs_gauche_resultat ← supprimer_attributs_gauche
     (dataframe, contrainte_erronee, None)
7  ajouter toutes les contraintes de supprimer_attributs_gauche_resultat à
     contraintes_generees
8  remplacer_attribut_gauche_resultat ← remplacer_attribut_gauche
     (dataframe, contrainte_erronee)
9  ajouter toutes les contraintes de remplacer_attribut_gauche_resultat à
     contraintes_generees
10 remplacer_attribut_droite_resultat ← remplacer_attribut_droite
     (dataframe, contrainte_erronee)
11 ajouter toutes les contraintes de remplacer_attribut_droite_resultat à
     contraintes_generees
12 return contraintes_generees
```

2.3.3 Étape 2: Filtrage des contraintes générées

A.Élimination des contraintes existantes

Notre solution permet de générer plusieurs contraintes en sortie, parmi lesquelles il est possible de trouver des contraintes déjà existantes dans notre système. Cela n'apporte aucune valeur ajoutée par rapport au processus de réparation. C'est pourquoi un filtrage est nécessaire pour ne conserver que les nouvelles contraintes générées. Notre processus du filtrage des contraintes prend en entrée deux listes : les contraintes existantes et les contraintes générées. Son objectif est d'éliminer les doublons des contraintes générées en les comparant avec les contraintes existantes.

Pour ce faire, nous proposons de considérer chaque côté d'une contrainte comme un ensemble et d'effectuer la comparaison côté par côté. On définit la légalité entre deux contraintes si l'ensemble représentant le côté gauche d'une contrainte est égal à l'ensemble représentant le côté gauche de l'autre contrainte, et de même si l'ensemble représentant le côté droit d'une contrainte est égal à l'ensemble représentant le côté droit de l'autre contrainte.

L'égalité entre deux ensembles d'éléments est définie par extensionnalité, deux ensembles sont égaux quand ils ont les mêmes éléments, c'est-à-dire que :

$A = B$ si et seulement si $A \subset B$ et $B \subset A$. [Vaught 2001]

B.Élimination des contraintes non valides

Un choix de réparation parmi l'ensemble de contraintes candidates produites par la variation d'une contrainte erronée doit être valide. Par conséquent, il doit minimiser le nombre de violations par l'ensemble de données.[Shaoxu Song 2016]. Une violation d'une dépendance fonctionnelle est considérée comme l'ensemble des tuples avec des valeurs d'attributs du côté droit qui sont distinctes mais produites par les mêmes valeurs du côté gauche de la FD. [Volkovs et al. 2014]

Afin d'assurer la validité des réparations, on propose de filtrer toutes les contraintes générées dans l'étape précédente, en ne conservant que celles qui sont satisfaites par l'ensemble des données.Par conséquent pour chacune des contraintes candidates, on considère comme contraintes valides celles qui unifient les valeurs de la partie droite pour le même ensemble de valeurs de la partie gauche.

Nous détaillons le processus que nous avons proposé pour la sélection des contraintes valides parmi l'ensemble des contraintes générées, comme suit :

Algorithm 5 : valider_contraintes(df, constraints)

```
   Data : df : dataframe, constraints : liste
   Result : valid_constraints : liste
 1 valid_constraints ← Liste vide
 2 foreach constraint DANS constraints do
 3 |   lhs, rhs ← Séparer les parties gauche et droite de la contrainte
 4 |   lhs ← Convertir lhs en liste de colonnes
 5 |   rhs ← Convertir rhs en liste de colonnes
 6 |   foreach col DANS lhs + rhs do
 7 |   |   if col N'EST PAS DANS les colonnes du DataFrame then
 8 |   |   |   Lever une erreur avec le message "La colonne 'col' n'existe pas dans le
   |   |   |    DataFrame."
 9 |   |   end
10 |   end
11 |   grouped_data ← Agréger les données du DataFrame en fonction des colonnes
   |    lhs et compter les valeurs distinctes dans les colonnes rhs
12 |   if toutes les valeurs du DataFrame grouped_data sont égales à 1 then
13 |   |   Ajouter constraint à valid_constraints
14 |   end
15 end
16 return valid_constraints
```

2.3.4 Etape 3 :Sélection de la contrainte réparation

Après le test de validité effectué à l'étape précédente, plusieurs contraintes résultantes peuvent être trouvées, pas seulement une, d'où la nécessité de choisir la contrainte à considérer comme réparation.

[Beskales et al. 2013] souligne qu'une solution pertinente pour la réparation des dépendances fonctionnelles vise à modifier la FD de manière quasi minimale. Par conséquent, nous proposons de mesurer la similarité entre chaque paire constituée d'une contrainte candidate et de la contrainte erronée afin de déterminer celle qui se rapproche le plus de la contrainte erronée et donc présentant la réparation ayant le coût le plus minimal par conséquent et afin de choisir la contrainte présentant la réparation ayant le coût le plus minimal. Nous proposant trois altérnatives permettant d'effectuer ce choix .

La première altérnative consiste à considérer la contrainte à faible coût comme celle qui est la plus similaire en termes d'attributs à la contrainte erronée. Pour cela, on considère une contrainte comme un ensemble d'attributs, et on utilise ainsi une mesure de similarité entre les ensembles.

[Choi et al. 2010] étudie la similarité entre les ensembles plusieurs mesures permettant d'évaluer la similarité entre deux ensembles. Il présente une mesure appelée la distance de jaccard.Il définit la distance de jaccard comme mesure évalue le rapport entre l'intersection et l'union des ensembles des éléments constituants les deux ensembles. Il détaillé le processus du calcul du coefficient de Jaccard comme suit :

1. Calcul de l'intersection : Trouver l'intersection des deux ensembles , c'est-à-dire les éléments communs aux deux ensembles.

2. Calcul de l'union : Trouver l'union des deux ensembles , c'est-à-dire tous les éléments uniques présents dans les deux ensembles combinés.

3. Calcul du coefficient de Jaccard : Diviser le nombre d'éléments dans l'intersection par le nombre d'éléments dans l'union. Le résultat est le coefficient de Jaccard, qui est une valeur comprise entre 0 et 1. Plus le coefficient est proche de 1, plus les ensembles sont similaires. La formule pour calculer le coefficient de Jaccard est la suivante :

   $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, où

   $J(A, B)$ représente le coefficient de Jaccard entre les ensembles A et B,

   $|A \cap B|$ représente le nombre d'éléments dans l'intersection des ensembles A et B

   $|A \cup B|$ représente le nombre d'éléments dans l'union des ensembles A et B.

Notre premier processus de sélection de la réparation à faible coût consiste à calculer les coefficients de Jaccard pour toutes les paires d'ensembles d'attributs représentant la contrainte erronée et un autre ensemble d'attributs représentant une contrainte parmi celles à tester. En comparant les valeurs des coefficients de Jaccard pour toutes les paires d'ensembles d'attributs, on sélectionne la paire qui présente le coefficient de Jaccard le plus élevé. la contrainte représentée par l'ensemble d'attributs qui fait partie de cette paire est choisie.

l'algorithme détaillant le processus de cette première altérnative du choix de réparation à faible coût est le suivant :

Algorithm 6 : Jaccard_selectionner_reparation(contrainteErronee, ensembleContraintes)

Data : contrainteErronee : chaîne de caractères, ensembleContraintes : liste
Result : contraintePlusSimilaire : chaîne de caractères

```
1  maxJaccard ← 0
2  contraintePlusSimilaire ← Aucune
3  foreach contrainte DANS ensembleContraintes do
4  |   coefficientJaccard ← jaccard_coefficient (contrainteErronee, contrainte)
5  |   if coefficientJaccard > maxJaccard then
6  |   |   maxJaccard ← coefficientJaccard
7  |   |   contraintePlusSimilaire ← contrainte
8  |   end
9  end
10 return contraintePlusSimilaire
```

La deuxième altérnative consiste à considérer la contrainte à faible coût comme celle qui minimise les opérations de transformation de la contrainte erronée en celle-ci.

[Aouragh et al. 2019] étudie la correction des chaînes de caractères erronées. Il utilise une mesure appelée la distance de Levenshtein pour choisir entre les différentes corrections proposées. Il définit la distance de Levenshtein comme une mesure évaluant la similarité entre deux chaînes de caractères en calculant une distance d'édition. Elle est définie comme le nombre minimum d'opérations d'édition de base nécessaires pour transformer une chaîne incorrecte en une chaîne correcte.

Il a détaillé le processus du calcul de la distance de Levenshtein comme suit :

1. Initialisez une matrice D de dimensions $(m+1)(n+1)$ comme suit :

$$D[i][j] = \begin{cases} 0 & \text{si } i = 0 \text{ et } j = 0 \\ i & \text{si } j = 0 \\ j & \text{si } i = 0 \end{cases}$$

2. Parcourez la matrice D de la manière suivante :

$$D[i][j] = \min \begin{cases} D[i-1][j] + 1 \\ D[i][j-1] + 1 \\ D[i-1][j-1] + (s_i \neq t_j) \end{cases}$$

où :

$D[i][j]$ est la distance de Levenshtein entre les préfixes de longueurs i et j des chaînes s et t, respectivement.

3. La distance de Levenshtein totale entre les chaînes s et t est donnée par $D[m][n]$.

Notre deuxième processus de sélection pour la réparation à faible coût consiste à considérer les contraintes comme des chaînes de caractères et calculer les distances de Levenshtein entre toutes les paires de chaînes de caractères représentant la contrainte erronée et une autre contrainte à tester. En comparant les valeurs des distances de Levenshtein pour toutes les paires de chaînes de caractères, nous sélectionnons la paire qui présente la distance de Levenshtein la plus minimale. La contrainte représentée par la chaîne de caractères faisant partie de cette paire est choisie. L'algorithme détaillant le processus de cette deuxième altérnative de choix de réparation à faible coût est le suivant :

Algorithm 7 : levenshtein_selectionner_reparation(contrainteErronee, ensembleContraintes)

Data : contrainteErronee : chaîne de caractères, ensembleContraintes : liste
Result : contraintePlusSimilaire : chaîne de caractères

```
1  min_distance ← ∞
2  most_similar ← Aucune
3  foreach contrainte DANS ensembleContraintes do
4  |  distance ← levenshtein_Distance (contrainteErronee, contrainte)
5  |  if distance < min_distance then
6  |  |  min_distance ← distance
7  |  |  most_similar ← contrainte
8  |  end
9  end
10 return contraintePlusSimilaire
```

La revue de la littérature nous a permis de constater que l'inclusion de l'utilisateur dans le processus de réparation est crucial pour parvenir à des réparations de haute qualité. Notre troisième altérnative consiste à impliquer l'utilisateur dans cette dernière étape afin qu'il choisisse la contrainte qui, selon lui, présente la réparation la plus pertinente. Ceci vise à conserver ladite contrainte en tant que réparation de la contrainte erronée.

Enfin, pour réparer plusieurs contraintes erronées, le processus complet de réparation d'une contrainte doit être répété pour chaque contrainte individuellement. Voici l'algorithme détaillant les étapes de notre processus de réparation des contraintes :

Algorithm 8 : Reparer_contraintes(df, liste_contraintes_erronees, contraintes_systemes)

Data : df : dataframe, liste_contraintes_erronees : liste, contraintes_systemes : liste
Result : contraintes_reparees

```
1  contraintes_reparees ← Liste vide
2  foreach contrainte_erronee DANS liste_contraintes_erronees do
3  |  contraintes_generees ← generer_contrainte (df, contrainte_erronee,
   |   contraintes_systemes)
4  |  contraintes_valides ← (df, contraintes_generees)
5  |  if longueur de contraintes_valides > 1 then
6  |  |  contrainte_finale ← (contrainte_erronee, contraintes_valides)
7  |  end
8  |  else
9  |  |  contrainte_finale ← Première contrainte de contraintes_valides
10 |  end
11 |  Ajouter (contrainte_erronee, contrainte_finale) à contraintes_reparees
12 end
13 return contraintes_reparees
```

2.4 Réparation hybride

Lorsque à la fois les contraintes et les données sont identifiées comme erronées, il devient impératif de réparer simultanément les contraintes et les données. [Volkovs et al. 2014] souligne que la réparation de données est effectuée en supposant que l'ensemble donné de contraintes est correct, et en recherchant des réparations pour les données qui satisfont les contraintes. Donc, nous proposons un processus de réparation hybride qui commence par réparer les contraintes afin de satisfaire l'hypothèse de cohérence des contraintes avant d'effectuer la réparation des données.

Le processus que nous avons proposé pour la réparation des contraintes utilise principalement la mesure de l'information mutuelle, appliquée sur des données présentant des liens cohérents entre les attributs. Des valeurs erronées des attributs peuvent affecter les valeurs de l'information mutuelle calculée et donc influencer la cohérence des réparations des contraintes proposées. Afin de pouvoir appliquer correctement notre processus de réparation des contraintes, nous commençons par éliminer les données détectées comme erronées de l'ensemble de données. Cela nous permet de conserver uniquement les liens entre les attributs qui sont corrects. Ensuite, on répare les contraintes erronées en appliquant le processus de réparation des contraintes.

Après cette première réparation et en obtenant des contraintes correctes permettant la réparation des données, nous reconsidérons l'ensemble des données initiales, y compris les données détectées comme erronées, et appliquons un processus de réparation des données. La réparation des données est en dehors de la portée de ce mémoire. On peut appliquer n'importe quelle approche proposée dans la littérarure telle que [Abdellaoui et al. 2017]

Voici l'algorithme détaillant le processus qu'on a proposé pour la réparation hybride :

Algorithm 9 : reparer_hybride(df, contrainte_erronee, liste_lignes_erronees)

Data : df : dataframe, contrainte_erronee : contrainte, liste_lignes_erronees : liste

Result : donnees_reparees, contrainte_reparee

1 df_propre ← `construire_df_propre` (df, liste_lignes_erronees)
2 contrainte_reparee ← `Reparer_Contraintes` (df_propre, contrainte_erronee, contraintes_systemes)
3 donnees_reparees ← `reparer_donnees` (df, contrainte_reparee)
4 **return** donnees_reparees, contrainte_reparee

2.5 Conclusion

Dans ce chapitre nous avons présenté notre propre solution pour l'identification de différents types d'anomalies dans les données et les règles ainsi que leurs réparations.

Nous avons conçu une solution pour l'identification de la source d'anomalie en extrayant les caractéristiques des violations détectées. Par la suite, nous calculons des statistiques basées sur ces caractéristiques, notamment le nombre de lignes violées. Ces statistiques sont utilisées comme propriétés pour entraîner des classifieurs supervisés sur des données générées par un GAN. Nos classifieurs sont capables de prédire la source d'anomalie, qu'il s'agisse d'une anomalie de contrainte en indiquant la contrainte violée, d'une anomalie de donnée en indiquant les lignes de données violant la contrainte, ou encore des lignes de données erronées et de la contrainte erronée en cas d'anomalie simultanée dans les deux.

Nous avons proposé une approche pour la réparation des contraintes. Cette approche consiste à exploiter les liens de dépendances dans les attributs du jeu de données en utilisant une mesure basée sur l'information mutuelle. Ces liens de dépendances sont calculés pour permettre de choisir les attributs à ajouter au côté gauche de la contrainte, les attributs à supprimer de la partie droite ou gauche de la contrainte. De plus, ces transformations d'ajout et de suppression sont fusionnées pour donner la possibilité de remplacer un attribut à gauche ou un attribut à droite.

Afin de garantir la validité des transformations générées, nous avons proposé un algorithme pour tester la satisfaction des données aux contraintes générées. Seules ceux qui sont valides sont conservées afin de procéder par la suite au choix final de la réparation à considérer.

Pour effectuer ce choix, nous avons proposé trois altérnatives visant à sélectionner la réparation à adopter en fonction du coût minimal de changement apporté à la contrainte erronée. La première altérnative utilise le coefficient de Jaccard afin de sélectionner la contrainte la plus similaire en termes d'attributs à la contrainte erronée. La deuxième altérnative utilise la distance de Levenshtein afin de sélectionner la contrainte qui minimise le nombre de transformations de la contrainte erronée pour y parvenir. La troisième, en cas de possibilité d'intégrer l'utilisateur à ce niveau, le choix de la réparation finale est effectué par lui.

Nous avons également conçu une approche de réparation hybride pour corriger les erreurs dans les données et les contraintes simultanément. Cette approche consiste à éliminer les erreurs de données et à appliquer processus de réparation des contraintes à un ensemble de données propre pour garantir la consistances des réparations de contraintes proposées. Ensuite à restaurer l'ensemble de données initial et appliquer l'approche de réparation des données que nous avons choisie pour que les données satisfassent les nouvelles réparations des contraintes proposées. La prochaine étape de notre travail sera dédiée à l'implémentation technique de notre solution.

Chapitre 3

Réalisation

3.1 Introduction

Après avoir exposé en détail la conception de notre solution, nous aborderons dans ce chapitre la phase de mise en œuvre. Pour commencer, nous présenterons l'environnement de développement, le langage de programmation et les bibliothèques que nous avons utilisés. Ensuite, nous explorerons en détail les fonctions développées.

3.2 Outils utilisés

3.2.1 Langages de programmation

Python.

En termes de langages de programmation, une panoplie de choix s'offre à nous, mais on a jugé convenable d'implémenter notre solution en Python.

Le langage de programmation Python est un langage à usage général, interprété et de haut niveau qui offre principalement une bonne lisibilité du code. Il a été créé par Guido van Rossum et publié en 1991. Non seulement sa bibliothèque standard est riche en fonctionnalités, mais il offre aussi un très grand nombre de bibliothèques crées par la communauté pour répondre a plusieurs besoins dans de nombreux domaines, y compris le développement web, l'automatisation, l'analyse de données, et l'apprentissage automatique.

Fig. 3.1 : Logo Python

3.2.2 Bibliothèques

Python est un langage qui offre une vaste gamme de bibliothèques externes. Pour répondre aux exigences de notre solution en termes de traitement des flux de données, représentation des données, manipulation de tableaux et matrices multidimensionnels, ainsi que pour effectuer des calculs de probabilité, d'information mutuelle et des opérations mathématiques et combinatoires sur les données, nous avons fait usage d'un ensemble de bibliothèques présentées ci-dessous.

1. **NumPy**

   NumPy est une bibliothèque Python qui ajoute le support pour les tableaux et les opérations mathématiques de haut niveau. Elle fournit des structures de données de

tableau multidimensionnel efficaces, ainsi que des fonctions pour effectuer des calculs numériques avancés. Numpy est largement utilisé pour le traitement des données, la manipulation des tableaux et les opérations scientifiques et mathématiques.

FIG. 3.2 : Logo NumPy

2. **SciPy**

Scipy (Scientific Python) est une bibliothèque Python destinée aux calculs scientifiques et techniques. Elle fournit des fonctionnalités avancées pour l'optimisation, l'algèbre linéaire, le traitement du signal, l'intégration numérique, les statistiques et bien plus encore. SciPy est souvent utilisé en complément de NumPy pour effectuer des analyses numériques et résoudre des problèmes mathématiques complexes.

FIG. 3.3 : Logo SciPy

3. **Scikit-learn**

Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment NumPy et SciPy.

Fig. 3.4 : Logo Scikit-learn

4. **TensorFlow**

   TensorFlow est une bibliothèque open-source pour l'apprentissage automatique et l'intelligence artificielle. Elle est principalement utilisée pour la création et l'entraînement de réseaux de neurones profonds. TensorFlow propose une structure de données appelée "Tenseurs" qui permet de manipuler des opérations mathématiques sur des tableaux multidimensionnels.

Fig. 3.5 : Logo TensorFlow

3.2.3 Environnement de développement

Google Colab.

Google Colab (Colaboratory) est un environnement de développement basé sur le cloud, fourni par Google, qui permet d'exécuter du code Python dans des blocs appelés "cellules". Il est principalement utilisé pour le développement en langage Python, l'apprentissage automatique, l'analyse de données et d'autres tâches liées à la science des données.

Fig. 3.6 : Logo Google Colab

3.3 Lecture des flux de données

Pour lire un flux de données en utilisant Python, on a adopté la bibliothèque pandas. pandas est une bibliothèque très populaire pour la manipulation et l'analyse de données

en Python. Elle est principalement utilisée pour travailler avec des données tabulaires bidimensionnelles, appelées DataFrames. Ces DataFrames peuvent être de taille variable et potentiellement hétérogènes. Ils contiennent des axes étiquetés (lignes et colonnes), ce qui permet de réaliser des opérations arithmétiques en alignant les étiquettes des lignes et des colonnes.

pandas offre diverses méthodes pour lire des données à partir de différentes sources. Voici quelques-unes des méthodes de lecture de données prises en charge :

Lecture de fichiers :

- CSV : pandas permet de lire des fichiers CSV (comma-separated values), qui sont des fichiers texte délimités utilisant une virgule pour séparer les valeurs.
- Excel : pandas peut lire des fichiers Excel (.xls, .xlsx, etc.) et créer des DataFrames à partir des feuilles de calcul qu'ils contiennent.
- ARFF : pandas peut lire des fichiers ARFF (Attribute-Relation File Format), qui sont des fichiers texte ASCII décrivant une liste d'instances partageant un ensemble d'attributs.
- JSON : pandas prend en charge la lecture de données au format JSON (JavaScript Object Notation), qui est couramment utilisé pour le transfert de données entre applications web.
- Parquet : pandas peut lire des fichiers au format Parquet, un format de stockage de données optimisé pour les analyses en Big Data.

Lecture de données depuis le Web :

- HTML : pandas peut extraire des tables de données à partir de pages web HTML et les convertir en DataFrames.
- XML : pandas peut lire des données au format XML (eXtensible Markup Language), un format de données structurées souvent utilisé pour échanger des données entre applications.

Lecture de données à partir d'une base de données :

SQL : pandas prend en charge la lecture de données à partir de bases de données relationnelles à l'aide du langage SQL.

Lecture de données depuis des APIs :

pandas peut être utilisé pour interagir avec diverses APIs et lire des données à partir de sources telles que les services web, les API REST, etc.

3.4 Identification de la source d'anomalie

L'approche pour mettre en place l'identification de la source d'anomalie repose essentiellement sur l'utilisation des fonctionnalités de machine learning disponibles dans les bibliothèques Python. Pour parvenir à cette identification, nous avons pris soin de diviser méticuleusement chaque étape du processus en sous-tâches distinctes et indépendantes.Dans ce qui suit, nous allons présenter les attributs et méthodes de chaque étape.

3.4.1 Détection des anomalies

Méthode	Paramètres	Explication
verifie_dependances	dataframe, liste de contraintes	Identifier les lignes qui violent chaque contrainte.

Tab. 3.1 : Description des méthodes de détection des anomalies

3.4.2 Extraction des caractéristiques des anomalies

Méthode	Paramètres	Explication
statistique1	dataframe, Table contenant les contraintes avec leurs lignes de violations	Calculer, pour chaque contrainte, le nombre de tuples de violation par rapport au nombre total de tuples.
statistique2	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer, pour chaque contrainte, la moyenne des violations pour chaque valeur de la partie gauche par rapport aux tuples erronés.
statistique3	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer, pour chaque contrainte, la moyenne des violations pour chaque valeur de la partie gauche par rapport au nombre total de tuples.
overlap	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Mesurer à quelle fréquence chaque tuple de violation est présent dans les tuples qui violent une contrainte donnée, en proportion de la fréquence totale d'apparition de ces mêmes motifs de violation dans les données.
statistique4	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer la moyenne des overlap calculées pour chaque contrainte.
statistique5	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer l'écart type des overlap calculées pour chaque contrainte.
statistique6	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer le maximum des overlap calculées pour chaque contrainte.
statistique7	dataframe, Table contenant les contraintes avec leurs lignes de violations, liste de contraintes	Calculer le minimum des overlap calculées pour chaque contrainte.

Tab. 3.2 : Description des méthodes d'extraction des caractéristiques des anomalies

3.4.3 Entraînement des classifieurs et prédiction des classes des anomalies

Méthode	Paramètres	Explication
créer_modèles_classification	-	Créer les 4 classifieurs supervisés (LR, RF, DT, SVM).
creer_gan	-	Créer un réseau génératif antagoniste.
entrainer_gan	gan, dataframe	Entraîner le GAN sur l'ensemble de données.
entraîner_prédire	dataframe, liste de contraintes	Entraîner chaque modèle, effectuer des prédictions, calculer et afficher les métriques de performance pour chaque modèle.
choisir_le_meilleur_classifieur	Résultats des mesures de performance	Choisir le meilleur modèle de classification.

Tab. 3.3 : Description des méthodes d'entraînement des classifieurs et prédiction des classes des anomalies

3.5 Réparation des règles de qualité

L'implémentation de la réparation des règles de qualité a été réalisée de manière méthodique et modulaire, en capitalisant sur les algorithmes conçus lors de la section préliminaire. Pour atteindre cet objectif, chaque étape du processus de réparation a été minutieusement décomposée en sous-tâches distinctes et autonomes. Cette approche a permis de créer un code bien organisé et facilement maintenable, tout en évitant les redondances et en favorisant la réutilisation des fonctions.

Chaque méthode a été conçue pour accomplir une tâche spécifique dans le processus de réparation des contraintes, ce qui améliore la lisibilité du code et facilite sa maintenance. Une fois les différentes méthodes mises en place, nous avons créé une fonction finale de réparation des contraintes qui orchestre l'exécution séquentielle des étapes de cette réparation. Cette fonction utilise les appels aux méthodes déjà implémentées, assurant ainsi une cohérence globale du processus. Dans ce qui suit, nous allons présenter les attributs et méthodes de chaque étape

3.5.1 Préparation des données

Avant de commencer, il est nécessaire de représenter les données textuelles en convertissant les attributs catégoriels en valeurs numériques. Cela prépare les données pour l'utilisation de l'information mutuelle dans les étapes suivantes.

3.5.2 Génération des contraintes candidates

Méthode	Paramètres	Explication
ajouter_attributs_gauche	dataframe, contrainte_erronée, n	Ajoute les attributs appropriés à la partie gauche de la contrainte erronée pour la corriger.
ajouter_attributs_droite	dataframe, contrainte_erronée, n	Ajoute les attributs appropriés à la partie droite de la contrainte erronée pour la corriger.
supprimer_attributs_droite	dataframe, contrainte_erronée, n	Supprime les attributs inappropriés de la partie droite de la contrainte erronée pour la corriger.
supprimer_attributs_gauche	dataframe, contrainte_erronée, n	Supprime les attributs inappropriés de la partie gauche de la contrainte erronée pour la corriger.
remplacer_attribut_gauche	dataframe, contrainte_erronée	Remplace l'attribut le plus inapproprié dans la partie gauche de la contrainte par l'attribut le plus approprié.
remplacer_attribut_droite	dataframe, contrainte_erronée	Remplace l'attribut le plus inapproprié dans la partie droite de la contrainte par l'attribut le plus approprié.
generer_combinaisons	liste_attributs_candidats, j	Génère des combinaisons d'attributs de taille j à partir de la liste d'attributs candidats pour l'ajout ou la suppression d'attributs.
generer_modifications_contraintes	dataframe, contrainte_erronée	Génère toutes les contraintes candidates en appelant les méthodes d'ajout, de suppression et de remplacement.

Tab. 3.4 : Description des méthodes de la génération des transformations pour la contrainte erronée.

3.5.3 Filtrage des contraintes

Méthodes	Paramètres	Explication
nettoyer_contraintes	contraintes_générées	Mettre les contraintes générées au bon format et supprimer celles qui ne respectent pas le format des dépendances fonctionnelles.
supprimer_contraintes_système	contraintes_nettoyées, contraintes_système	Supprimer parmi toutes les contraintes nettoyées celles qui existent déjà dans le système.
valider_contraintes	dataframe, contraintes_conservées	Choisir parmi toutes les contraintes conservées celles qui sont valides et qui satisfont toutes les lignes des données.

Tab. 3.5 : Description des méthodes du filtrage des contraintes générées

3.5.4 Choix final de la réparation

Méthodes	Paramètres	Explication
Jaccard_coefficient	contrainte1, contrainte2	Calculer le coefficient de Jaccard entre deux contraintes.
trouver_contrainte_plus_similaire	contrainte_cerronée, liste de contraintes	Choisir parmi la liste de contraintes la contrainte la plus similaire à la contrainte erronée.
Levenshtein_distance	contrainte1, contrainte2	Calculer la distance de Levenshtein entre deuc contraintes
trouver_contrainte_longueur_proche	contrainte_erronée, liste de contraintes	Choisir parmi la liste de contraintes la contrainte qui a la longueur la plus proche a la longueur de contrainte erronée.

Tab. 3.6 : Description des méthodes pour le choix final de la réparation

3.5.5 Réparation d'une liste des contraintes erronées

Méthodes	Paramètres	Explication
réparer_contraintes	dataframe, contraintes_système, contraintes_erronées	Réparer les contraintes erronées en boucle, en suivant les étapes déjà citées pour chacune d'entre elles.

Tab. 3.7 : Description des méthodes de la réparation d'une liste des contraintes erronées

3.6 Réparation hybride

La mise en œuvre de la réparation hybride repose sur une approche qui combine de manière cohérente nos techniques de correction des règles de qualité avec celles de la réparation des données. Cette approche synergique vise à optimiser la fiabilité des données corrigées en tirant parti des avantages des deux méthodologies.

Une fonction globale a été implémentée, prenant en entrée les résultats du processus d'identification d'anomalie hybride q comprenant les identifiants des lignes erronées ainsi que la contrainte erronée.Cette fonction englobe une nouvelle fonction préliminaire . Cette fonction revêt une importance cruciale, car elle permet la construction d'une nouvelle table de données propre en excluant les lignes de données identifiées comme étant erronées. Cela a pour effet de garantir la fiabilité du processus de réparation des règles de qualité.

Cette fonction est suivie par un appel de fonction de réparation des règles de qualité prenant en entrée cette nouvelle table propre, ainsi que la règle identifiée comme erronée, ce qui produit une réparation pour la règle.

Un dernier appel est effectué pour la fonction de réparation des données, qui prend en entrée la table d'origine avec toutes les lignes de données, ainsi que la contrainte réparée, afin de produire en sortie une table rectifiée.

3.7 Conclusion

Dans cette partie, nous avons abordé la réalisation de notre solution en définissant les différents outils et bibliothèques que nous avons utilisés pour parvenir à cette réalisation. Ensuite, nous avons décrit l'ensemble des fonctions développées ainsi que leurs différents détails techniques.

Dans le chapitre suivant, nous entamerons la phase de test et de validation de notre solution.

Chapitre 4

Tests et validation

4.1 Introduction

Dans ce chapitre, nous allons tester notre solution sur différents jeux de données réels afin d'étudier l'efficacité de nos modules implémentés.

Pour l'évaluation de l'approche d'identification de la source d'anomalie, nous allons modifier l'ensemble des données en y introduisant des erreurs. De plus, nous allons également introduire des erreurs dans certaines des règles de qualité. Par la suite, nous utiliserons la validation croisée pour tester la performance de notre approche. En outre, nous allons tester le temps d'exécution en variant la taille des données.

En ce qui concerne l'approche de réparation de contraintes, nous allons tester son efficacité en variant les types d'erreurs dans les contraintes, notamment l'ajout, la suppression et le remplacement, en utilisant à chaque fois une méthode de choix final de réparation que nous avons proposée. Notamment celle qui est basée sur le coefficient de Jaccard, l'autre qui est basée sur la distance de Levenshtein, et celle qui implique l'utilisateur pour effectuer ce choix, afin de pouvoir par la suite choisir la méthode que nous adoptons pour notre solution. Nous allons également tester le temps d'exécution en variant le nombre de contraintes à réparer ainsi que la taille du jeu de données.

Pour l'évaluation de l'approche hybride, nous allons adopter la meilleure version de l'approche de réparation de contraintes obtenue aprés l'evaluation cette approche, et nous allons varier les types d'erreurs dans les contraintes et les taux d'erreur dans les données afin de tester l'efficacité de notre approche. Nous commencerons par la définition des différents jeux de données que nous allons utiliser pour cette évaluation.

4.2 Jeux de données

Nous évaluons notre solution sur 2 jeux de données bien connus issus de la littérature existante.

4.2.1 Hospital

les données constituant ce jeu de données ont été obtenues auprès du Département de la Santé et des Services Sociaux des États-Unis.Elles comprennent 100 000 enregistrements avec 9 attributs utilisés dans les règles de qualité des données : Provider Number,zip, city, state, phoneNumber, Measure Code, Measure Name , condition, et stateAvg.

Les dépendances fonctionnelles valides au sein de ce jeu de données sont :

- $\text{MeasureCode}, \text{State} \rightarrow \text{StateAvg}$
- $\text{ZIPCode} \rightarrow \text{State}$
- $\text{PhoneNumber} \rightarrow \text{ZIPCode}$
- $\text{MeasureCode} \rightarrow \text{MeasureName}$
- $\text{MeasureCode} \rightarrow \text{Condition}$

- PhoneNumber → City
- PhoneNumber → State
- State, MeasureName → StateAvg

4.2.2 Tax

Représente les informations fiscales personnelles aux États-Unis, pour ce jeu de données nous avons extrait 50000 enregistrements avec 3 attributs utilisés dans les règles de qualité des données : zip, city et state.

Les dépendances fonctionnelles valides au sein de ce jeu de données sont :

- Zip → State
- Zip → City

Pour les deux jeux de données, afin de pouvoir tester notre solution contre les anomalies des données, nous avons introduit des erreurs en ajoutant du texte aléatoire aux attributs à des taux de : 2%, 4%, 6%, 8% et 10%.

4.3 Mesures d'évaluation

L'évaluation objective des performances et de l'efficacité du système que nous avons élaboré est impérative pour déterminer dans quelle mesure le système répond aux normes et aux exigences préétablies. C'est à ce stade que les mesures d'évaluation entrent en jeu, nous offrant la possibilité de quantifier et d'analyser divers aspects de la performance du système. Ces aspects incluent la rapidité d'exécution, la capacité à identifier les anomalies avec précision et à effectuer des réparations pertinentes, tout en maintenant la qualité des règles établies.

1. Temps d'exécution : Le temps d'exécution mesure la durée nécessaire à notre système pour effectuer le traitement requis. Il est essentiel d'évaluer cette mesure pour déterminer si notre système est suffisamment rapide pour être utilisé dans des scénarios en temps réel ou s'il nécessite des améliorations afin de réduire le délai de traitement.

2. Précision : La précision est une mesure qui évalue la capacité de notre système à fournir des résultats exacts et pertinents. Elle quantifie la proportion des résultats pertinents parmi l'ensemble des résultats fournis par notre système.

3. Rappel : Le rappel est une mesure qui évalue la capacité de notre système à identifier et à récupérer la totalité des éléments pertinents présents. Il mesure la proportion des éléments pertinents correctement identifiés parmi tous les éléments pertinents existants.

4. La F-mesure : La F-mesure est une mesure composite qui combine à la fois la précision et le rappel en un seul score. Elle permet de prendre en compte à la fois la qualité des résultats et la capacité de notre système à en identifier un maximum.

4.4 Identification de la source d'anomalie

4.4.1 Mesures d'évaluation

On redéfinit les mesures de performance pour cette partie d'évaluation comme suit :

- **La précision** : évalue la proportion des motifs positifs correctement classés parmi tous les motifs classés comme positifs. On peut définir la formule comme suit :

$$\text{Précision} = \frac{VraisPositifs}{VraisPositifs + FauxPositifs}$$

- **Le rappel** : évalue la proportion des motifs positifs correctement classés parmi tous les motifs réellement positifs. On peut définir la formule comme suit :

$$\text{Rappel} = \frac{VraisPositifs}{VraisPositifs + FauxNgatifs}$$

- **F-mesure** : combine les deux mesures précision et rappel pour fournir une vue globale de la performance de l'identification de la source d'anomalie. On peut définir la formule comme suit :

$$F1 = \frac{2 \cdot (\text{Précision} \cdot \text{Rappel})}{\text{Précision} + \text{Rappel}}$$

4.4.2 Evaluation sur le jeu de données Hospital

On exécute le processus d'identification de la source d'anomalie sur le jeu de données *hospital* avec les paramètres suivants :

- Taille des données : 100k lignes
- Taux d'erreur sur les données : 4%

Résultats d'efficacité :

Nous avons utilisé trois FDs valides et introduit des FDs erronées de manière aléatoire dans ce jeu de données. En altérant les dépendances correctes par la suppression, l'ajout ou le remplacement d'attributs, nous avons créé diverses contraintes erronées, à savoir :

- MeasureCode $\rightarrow$ City
- PhoneNumber $\rightarrow$ City, Condition

- MeasureCode → Condition, City

- MeasureCode → MeasureName, City

- PhoneNumber → ZIPCode, Condition

On applique notre approche d'identification de la source d'anomalie avec la technique de validation croisée en prenant 30 % de l'ensemble de données présentant les anomalies et les données générées comme ensemble de test, et le reste comme ensemble d'entraînement.

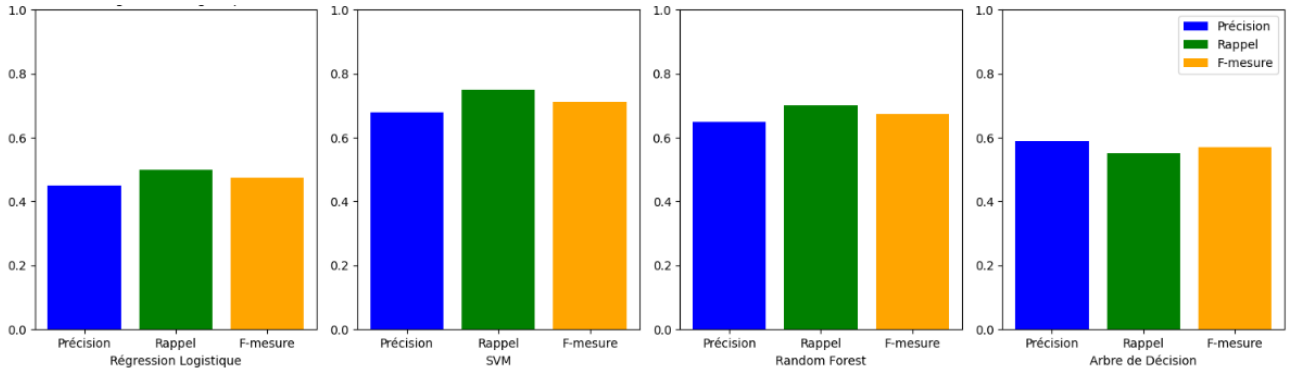


FIG. 4.1 : Evaluation de l'efficacité de l'identification de la source d'anomalie pour le jeu de données Hospital

Les résultats des quatre figures montrent que notre approche d'identification de la source d'anomalie a donné de meilleurs résultats en utilisant le classifieur SVM. Cela est justifié par sa capacité à maximiser la marge de séparation entre les classes, à traiter les anomalies comme des points cruciaux aux marges de classe ou des erreurs de classification. En utilisant des noyaux, il peut également gérer des données non linéaires, ce qui est essentiel pour détecter le type des anomalies hybrides.

Résultats du temps d'exécution :

Nous avons mesuré le temps d'exécution du processus d'identification de la source d'anomalie en utilisant le classifieur svm qui a donné des bons results lors de l'évaluation précédente et en variant la taille du jeu de données.

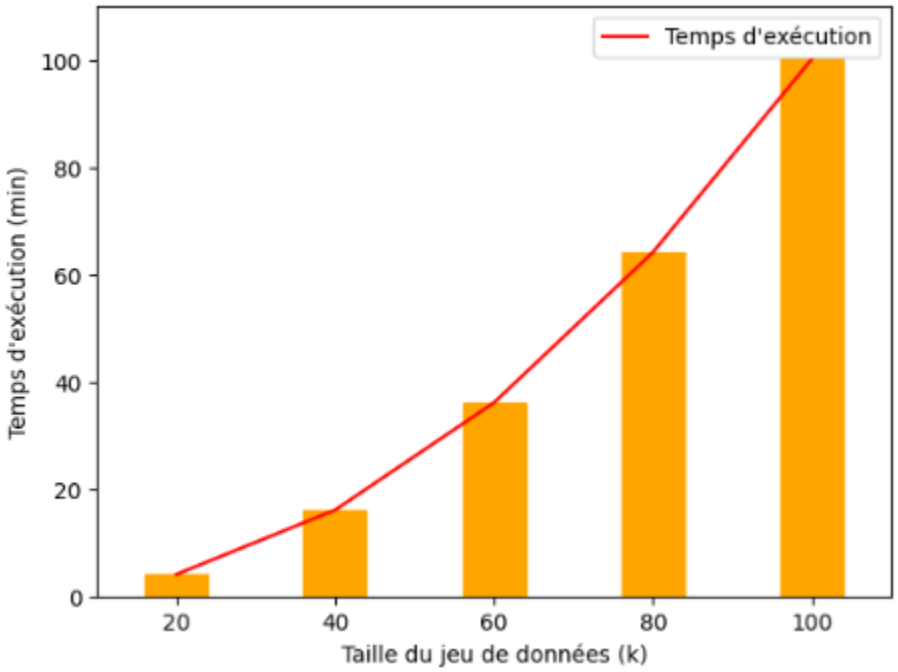


FIG. 4.2 : Evaluation du temps d'exécution de l'identification de la source d'anomalie pour le jeu de données Hospital

Les résultats de la figure montrent l'évolution du temps d'exécution de notre processus d'identification de la source d'anomalie en variant la taille du jeu de données. Les résultats indiquent que notre processus augmente de manière croissante avec l'augmentation de la taille du jeu de données, mais de manière non linéaire.

4.4.3 Evaluation sur le jeu de données Tax

Afin de confirmer l'efficacité de notre approche d'identification de la source d'anomalies et de consolider les conclusions tirées lors de la première évaluation avec le jeu de données "Hospital", nous avons choisi de mettre à l'épreuve notre approche sur un deuxième jeu de données. On exécute le processus d'identification de la source d'anomalie sur le jeu de données "tax" avec les paramètres suivants :

- Taille des données : 50k lignes
- Taux d'erreur dans les données : 6%

Résultats d'efficacité :

De plus des FDs valides nous avons introduit aléatoirement des dépendances fonctionnelles erronées dans ce jeu de données. Cela a été réalisé en altérant les dépendances correctes, soit en ajoutant de nouveaux attributs, soit en remplaçant certains attributs existants. Nous n'avons pas effectué de suppressions puisque les deux FD sont construites à partir de exactement deux attributs. Ainsi,nous avons créé les contraintes erronées suivantes :

- State → City

- City → State

On applique notre approche d'identification de la source d'anomalie avec la technique de validation croisée en prenant 30 % de l'ensemble de données présentant les anomalies et les données générées comme ensemble de test, et le reste comme ensemble d'entraînement.

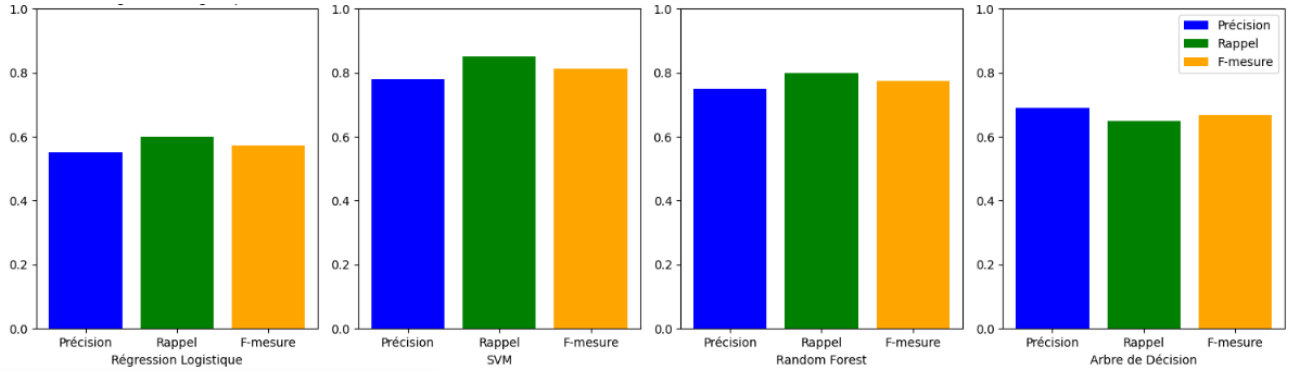


FIG. 4.3 : Evaluation de l'efficacité de l'identification de la source d'anomalie pour le jeu de données Tax

Les résultats présentés dans la figure confirment les conclusions tirées de l'évaluation du jeu de données hospital, démontrant que notre processus d'identification de la source d'anomalie présente de meilleurs résultats avec le classifieur SVM.

Résultats du temps d'exécution :

Nous avons mesuré le temps d'exécution du processus d'identification de la source d'anomalie en variant la taille du jeu de données.

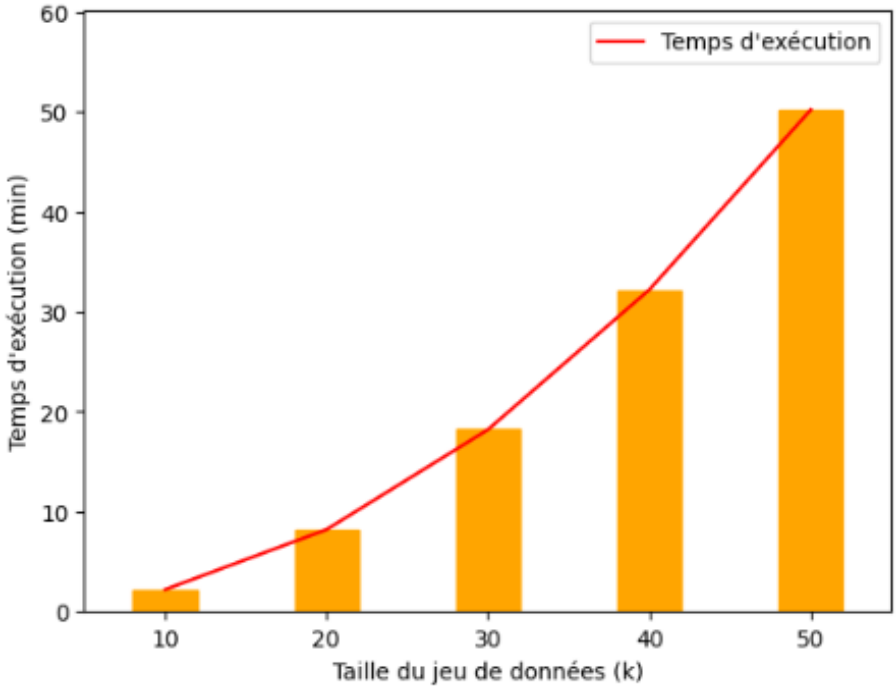


FIG. 4.4 : Evaluation du temps d'exécution de l'identification de la source d'anomalie pour le jeu de données Tax

Les résultats dans la figure reflètent également les résultats obtenus lors de l'évaluation du jeu de données "Hospital", confirmant que le temps d'exécution de notre processus

d'identification de la source d'anomalie est corrélé à la taille du jeu de données, mais n'augmente pas de manière linéaire.

4.5 Réparation des règles de qualité

4.5.1 Mesures d'évaluation

On redéfinit les mesures de performance pour cette partie d'évaluation comme suit :

- **La précision** : évalue la proportion des réparations correctes parmi celles proposées par l'algorithme. On peut définir la formule comme suit :

$$\text{FD-Précision} = \frac{CA + CS}{TA + TS}$$

  Où :

  CA représente le nombre d'attributs correctement ajoutés.

  CS représente le nombre d'attributs correctement supprimés.

  TA représente le nombre total d'attributs ajoutés.

  TS représente le nombre total d'attributs supprimés.

- **Le rappel** : quantifie la proportion des réparations correctes identifiées parmi toutes les réparations nécessaires. On peut définir la formule comme suit :

$$\text{FD-Rappel} = \frac{CA + CS}{NA + NS}$$

  Où :

  CA représente le nombre d'attributs correctement ajoutés.

  CS représente le nombre d'attributs correctement supprimés.

  NA représente le nombre d'attributs nécessaires à ajouter.

  NS représente le nombre d'attributs nécessaires à supprimer.

- **F-mesure** : combine les deux mesures précision et rappel pour fournir une vue globale de la performance des réparations proposées. On peut définir la formule comme suit :

$$FD - F1 = \frac{2 \cdot (\text{FD-Précision} \cdot \text{FD-Rappel})}{\text{FD-Précision} + \text{FD-Rappel}}$$

4.5.2 Evaluation sur le jeu de données Hospital

On exécute l'algorithme de réparation des contraintes sur le jeu de données *hospital* avec les paramètres suivants :

- Taille des données : 100k lignes

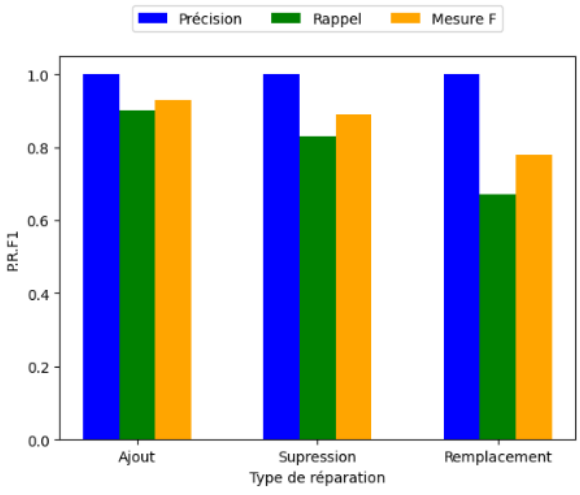
- Taux d'erreur sur les données : 0%

Résultats d'efficacité :

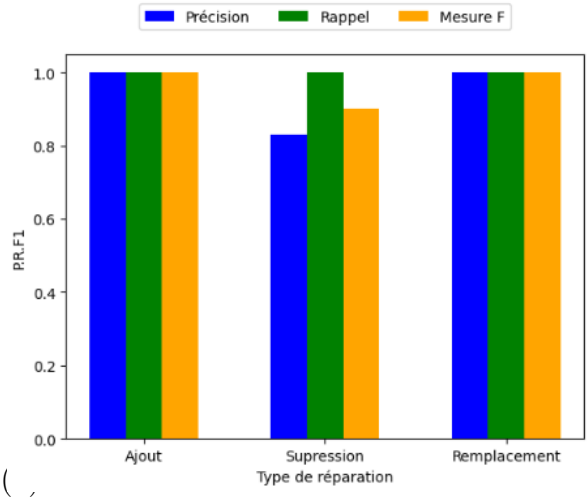
Nous avons introduit des FDs erronées aléatoires dans ce jeu de données. En altérant les dépendances correctes par la suppression, l'ajout ou le remplacement d'attributs, nous avons créé divers contraintes erronées qui sont :

- MeasureCode → City
- PhoneNumber → City, Condition
- MeasureCode → Condition, City
- MeasureCode → MeasureName, City
- PhoneNumber → ZIPCode, Condition
- MeasureCode, State → StateAvg, ZIPCode
- PhoneNumber → MeasureName
- State → StateAvg
- MeasureCode → StateAvg
- State, Condition → StateAvg

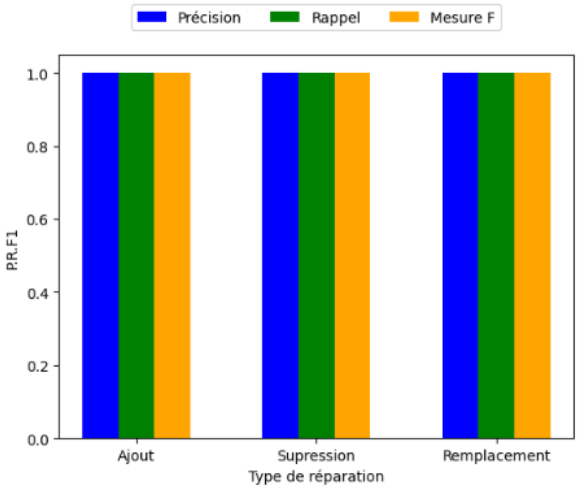
On applique notre approche de réparation des contraintes aux contraintes erronées. À chaque fois, nous utilisons l'une des trois méthodes que nous avons définies pour sélectionner la réparation finale. Ensuite, nous étudions l'impact des variations du type d'erreur pour chaque méthode.



(a) Sélection de réparation via la similarité de Jaccard



(b) Sélection de réparation via la distance de Levenshtein



(c)
Sélection de réparation par l'utilisateur

FIG.4.5 : Évaluation de l'efficacité de la Réparation des Contraintes en Variant les Types d'erreurs pour le Jeu de données "Hospital".

Les résultats illustrés dans la figure (a) démontrent que notre algorithme de réparation des dépendances fonctionnelles en utilisant la méthode de choix de la réparation finale basée sur la similarité de Jaccard a obtenu des scores élevés dans le cas où les FD devraient être réparées par l'ajout, et il a montré des résultats moins élevés pour les FD nécessitant des réparations par suppression ou remplacement. Nous expliquons le score inférieur pour les réparations par suppression ou remplacement par le fait que le coefficient de Jaccard utilisé pour le choix final de la réparation privilégie les contraintes avec le plus grand nombre d'attributs en commun avec la contrainte erronée, ce qui diminue sa valeur pour les contraintes générées en supprimant ou remplaçant des attributs.

Les résultats illustrés dans la figure (b) montrent que notre algorithme de réparation des FD en utilisant la méthode de choix de la réparation finale basée sur la distance de Levenshtein a obtenu des scores plus élevés dans tous les cas d'erreurs avec une F-mesure qui dépasse 90%, bien que les résultats pour les FD nécessitant des réparations par supression de plusieurs attributs ne soient pas à 100% exactes. Nous expliquons cela par le fait que la distance de Levenshtein privilégie de minimiser le nombre d'opérations de transformation.

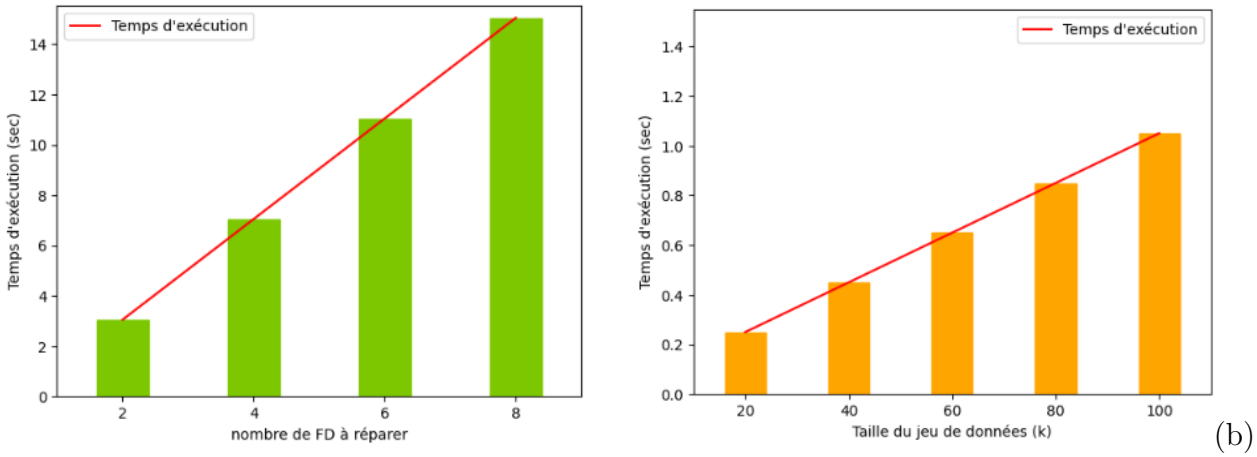
Les résultats présentés dans la figure (c) démontrent que notre algorithme de réparation des dépendances fonctionnelles , qui implique l'intervention de l'utilisateur dans le choix final de la réparation, a obtenu des résultats parfaits dans tous les cas. Cela met en évidence l'efficacité de notre algorithme de réparation des FD avec une f mesure de 100% dans tous les cas.cela montre que notre processus dès sa première étape, qui consiste à appliquer des transformations à la contrainte erronée, il est capable de générer la réparation souhaitée. Cette réparation est ensuite validée dans la deuxième étape de test de validité. L'utilisateur peut alors prendre des décisions éclairées pour choisir la contrainte à adopter en tant que réparation finale parmi les contraintes suggérées par notre processus.

Ces résultats montrent que notre approche de réparation des contraintes propose des réparations parfaites dans le cas où l'interaction humaine est possible lors de la dernière étape du choix de réparation. Dans le cas contraire, il présente de bons résultats pour la méthode de choix de réparation basée sur la distance de Levenshtein, mais des résultats

moins convaincants pour celle basée sur le coefficient de Jaccard.

Résultats du temps d'exécution :

Nous avons mesuré le temps d'exécution du processus de réparation des FD en utilisant la version de la distance de Levenshtein qui a donné des bons results lors de l'évaluation précédente et en variant le nombre de FDs à réparer et en variant aussi la taille du jeu de données.



(a) Variation du nombre de FD à réparer (b) Variation de la taille du jeu de données

FIG. 4.5 : Évaluation du temps d'exécution pour le jeu de données "Hospital"

Dans la Figure (a), nous montrons le résultat du temps d'exécution de notre processus de réparation des FD en fixant la taille du jeu de donnée à 100k et en variant le nombre de DF à réparer. Le temps d'exécution augmente linéairement à mesure que le nombre de FD à réparer augmente.

Dans la Figure (b),nous montrons le résultat du temps d'exécution de notre processus de réparation des FD en fixant le nombre de FD à réparer à une seule et en variant la taille du jeu de données.Le résultat montre que notre processus se met bien à l'échelle et répare les FDs en temps linéaire.

Ces résultats mettent en évidence l'efficacité de notre approche pour réparer les FDs en quelques secondes.

4.5.3 Evaluation sur le Jeu de données Tax

Afin de confirmer l'efficacité de notre approche de réparation des contraintes et de consolider les conclusions tirées lors de la première évaluation avec le jeu de données "Hospital", nous avons choisi de mettre à l'épreuve notre approche sur un deuxième jeu de données. On exécute l'algorithme de réparation des contraintes sur le jeu de données "tax" avec les paramètres suivants :

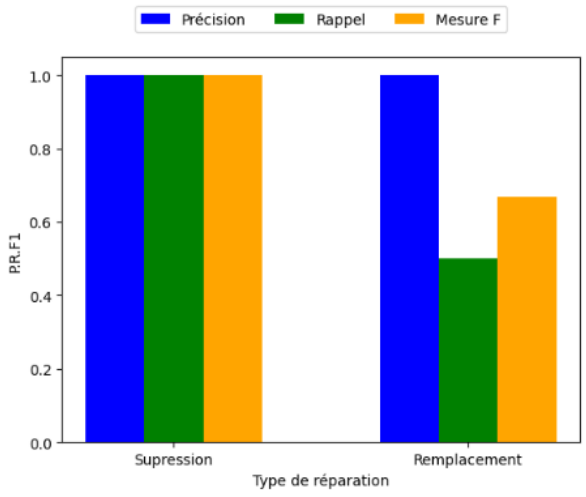
- Taille des données : 50k lignes
- Taux d'erreur dans les données : 0%

Résultats d'efficacité :

Nous avons introduit aléatoirement des dépendances fonctionnelles erronées dans ce jeu de données. Cela a été réalisé en altérant les dépendances correctes, soit en ajoutant de nouveaux attributs, soit en remplaçant certains attributs existants. Nous n'avons pas effectué de suppressions puisque les deux FD sont construites à partir de exactement deux attributs. Ainsi,nous avons créé divers contraintes erronées qui sont :

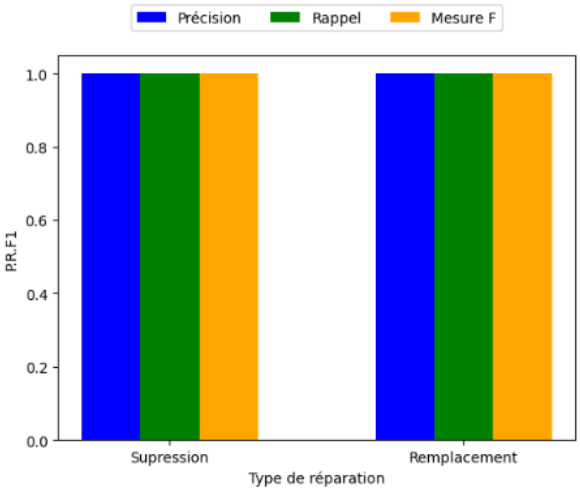
- State → City
- City → State
- Zip, State → City
- Zip, City → State

On applique notre approche de réparation des contraintes aux contraintes erronées. À chaque fois, nous utilisons l'une des trois méthodes que nous avons définies pour sélectionner la réparation finale. Ensuite, nous étudions l'impact des variations du type d'erreur pour chaque méthode.



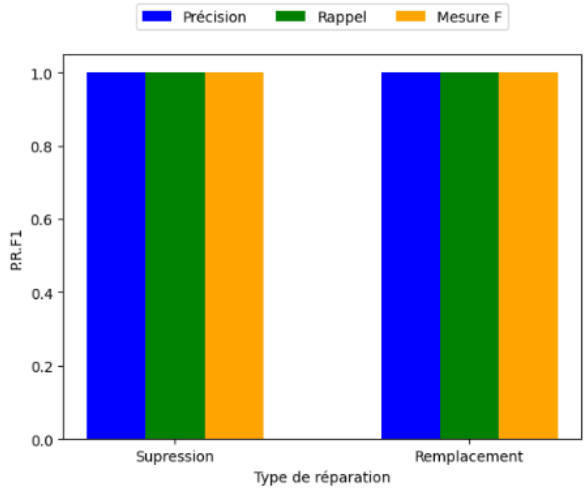
(a)

Sélection de réparation via la similarité de jaccard



(b)

Sélection de réparation via la similitude de longueur



(c)

Sélection de réparation par l'utilisateur

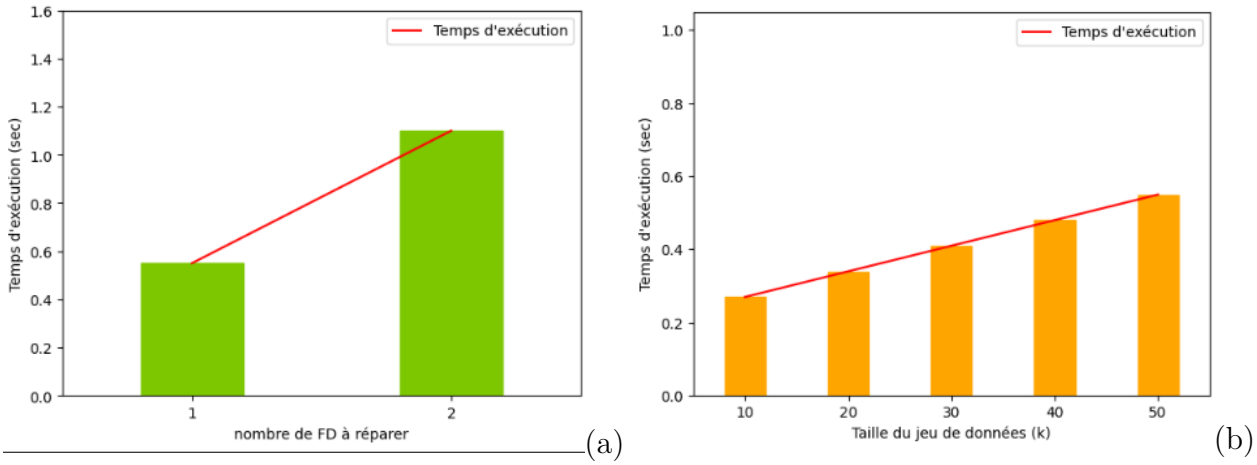
Fig. 4.6 : Évaluation de l'efficacité de la Réparation des Contraintes en Variant les Types d'erreurs pour le Jeu de données"Tax"

Les résultats des trois figures confirment les conclusions tirées de l'évaluation du jeu de données hospital, démontrant que notre algorithme de réparation des contraintes propose des réparations parfaites lorsque l'interaction humaine est permise lors de la dernière étape du choix de réparation. En l'absence de cette interaction, l'algorithme présente des résultats satisfaisants pour la méthode de réparation basée sur la distance de Levenshtein avec une mesure F de 100%, tandis que les résultats sont moins convaincants pour celle basée sur le coefficient de Jaccard.

La F-mesure pour les trois cas est supérieure à celle obtenue lors de l'évaluation du jeu de données hospitalier. Cette amélioration peut s'expliquer par le fait que, dans ce jeu de données, chaque dépendance fonctionnelle à réparer est formée par exactement deux attributs, ce qui réduit le nombre des transformations générées et renforce la précision de la détection de la dépendance fonctionnelle choisie comme réparation finale.

Résultats du temps d'exécution :

Nous avons mesuré le temps d'exécution du processus de réparation des FD en variant le nombre de FDs à réparer et en variant la taille du jeu de données.



(a) Variation du nombre de FD à réparer

(b) Variation de la taille du jeu de données

FIG. 4.7 : Évaluation du temps d'exécution pour le jeu de données "Tax"

Les résultats dans les deux figures simulent également les résultats obtenus lors de l'évaluation du jeu de données "Hospital", confirmant que le temps d'exécution de notre processus de réparation des dépendances fonctionnelles évolue de manière proportionnelle à la taille du jeu de données et au nombre de FD à réparer.

4.6 Réparation hybride

4.6.1 Mesures d'évaluation

Pour l'évaluation de l'efficacité de la réparation des données après l'application de notre approche de réparation hybride,on utilise la même formule pour le calcul de la mesure F utilisé pour l'évaluation de l'approche de réparation des contraintes et on redéfinit la précision et le rappel comme suit :

- **Data-précision** : évalue le nombre de cellules correctement réparées par rapport

au nombre total de cellules réparées.

- **Data-Rappel** : évalue le nombre de cellules correctement réparées par rapport au nombre total de cellules erronées.

4.6.2 Evaluation sur le jeu de données Hospital

On Utilise pour l'évaluation de notre processus de réparation hybride le jeu de données *hospital* avec une taille de 20k enregistrements et en prenant en compte l'ensemble complet de ses règles, en incluant 3 DF erronées, chacune contenant un type d'erreur différent (suppression, ajout, remplacement). On exécute notre processus de réparation hybride en utilisant la version de réparation des contraintes basée sur la distance de Levenshtein qui a donné des bonnes résultés lors de l'évaluation de cette approche, et à chaque fois on injecte un taux d'erreur des données diffèrent.

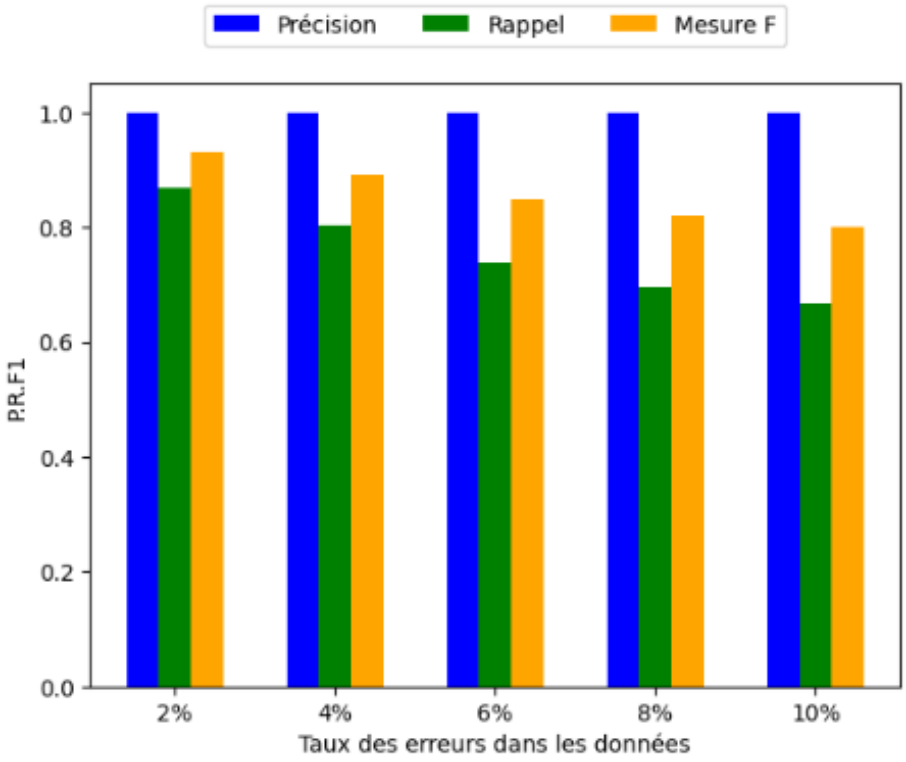


Fig. 4.8 : Évaluation de l'efficacité de l'approche hybride pour le jeu de données Hospital.

Les résultats illustrés dans la figure ci-dessus montrent que notre approche de réparation hybride fonctionne très bien lorsqu'il s'agit de traiter à la fois les erreurs de données et les erreurs de FDs. Ils montrent également que le score F1 est stable et supérieur à 80% avec une précision de 100%.

Cela est dû au fait que les données ne sont modifiées que lorsque nous réparons toutes les FD erronées sachant que toutes les FDs sales utilisés pour ce test ont été correctement réparées dans l'évaluation de l'approche de réparation des règles de qualité.

4.6.3 Evaluation sur le jeu de données Tax

Afin de confirmer l'efficacité de notre approche hybride , nous avons choisi de mettre à l'épreuve notre approche sur un deuxième jeu de données. on utilise cette fois ci le jeu de données *Tax* avec une taille de 5k enregistrements et en injectant des erreurs sur ses deux règles. On exécute notre processus de réparation hybride en utilisant la version de réparation des contraintes basée sur la distance de Levenshtein qui a donné des bonnes résultés lors de l'évaluation de cette approche, et à chaque fois on injecte un taux d'erreur des données diffèrent.

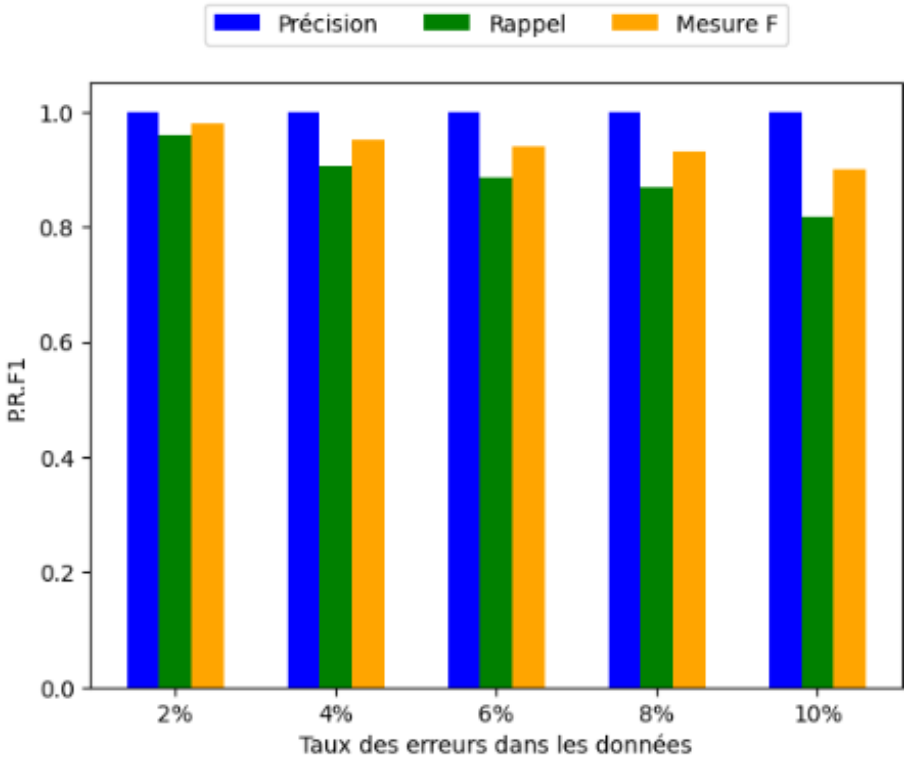


FIG. 4.9 : Évaluation de l'efficacité de l'approche hybride pour le jeu de données Tax.

Les résultats de cette figure confirment l'efficacité de notre approche observé lors de l'évaluation du jeu de données "Hospital", démontrant que notre algorithme de réparation hybride permet de réparer les règles de qualité et les données simultanément et avec des résultats satisfaisants.

4.7 Conclusion

Dans cette partie, nous avons étudié l'efficacité de notre solution sur deux ensembles réels de jeux de données.

L'évaluation de l'approche d'identification de la source d'anomalie en introduisant des erreurs dans les données et les règles de qualité a montré de bons résultats, avec une bonne F-mesure mesurant la performance de notre processus.

En ce qui concerne l'évaluation de notre approche de réparation de contraintes en variant les types d'erreurs dans les contraintes, notamment l'ajout, la suppression et le

remplacement, elle a donné des résultats parfaits lorsque l'intégration de l'utilisateur est possible pour le choix final de la réparation. Dans le cas contraire, elle a obtenu de bons résultats avec une F-mesure élevée en utilisant la distance de Levenshtein pour effectuer ce choix final. Cette approche a également montré un temps d'exécution optimal et linéairement croissant en variant le nombre de contraintes à réparer ainsi que la taille du jeu de données.

L'évaluation de l'approche hybride en variant les types d'erreurs dans les contraintes et les taux d'erreur dans les données a également montré de bons résultats, avec une F-mesure élevée mesurant la performance de notre processus de réparation hybride.

Les résultats obtenus à partir de nos différents tests démontrent que notre solution est capable de détecter et réparer les anomalies efficacement.

Conclusion et perspectives

Le processus de gestion de la qualité des données, englobant principalement la détection et la réparation des anomalies, se positionne comme un élément crucial pour la prise de décisions éclairées. Les anomalies dans les données ne se restreignent pas uniquement aux données elles-mêmes, mais peuvent également inclure des anomalies au niveau des règles de qualité, voire simultanément dans les données et les règles. Au cœur de cette dynamique réside le défi complexe d'identifier la source de l'anomalie et de proposer une réparation appropriée en lien avec cette source.

Dans ce mémoire de fin d'études, nous avons réalisé une analyse approfondie des travaux portant sur cette problématique. Cette démarche nous a ensuite guidé vers l'élaboration de notre propre approche en vue de l'identification et de la réparation des anomalies.

Nous avons conçu une solution pour l'identification de la source d'anomalie en extrayant les anomalies sous forme de données représentant les violations des contraintes ainsi que les contraintes elles-mêmes qui ont été violées. Par la suite, nous avons calculé des statistiques basées sur les caractéristiques de ces violations, notamment le nombre de lignes violées. Ces statistiques sont utilisées comme des propriétés pour entraîner un ensemble de classifieurs supervisés sur des données générées par un GAN. Nos classifieurs sont capables de prédire la source d'anomalie, que ce soit une anomalie de contrainte en indiquant la contrainte violée, une anomalie de données en indiquant les lignes de données violant la contrainte, voire les lignes de données erronées et la contrainte erronée en cas d'anomalie simultanée dans les deux.

Nous avons proposé une approche pour la réparation des contraintes. Cette approche consiste à exploiter les liens de dépendances dans les attributs du jeu de données en utilisant une mesure basée sur l'information mutuelle. Ces liens de dépendances sont calculés pour permettre de choisir les attributs à ajouter au côté gauche de la contrainte, les attributs à supprimer de la partie droite ou gauche de la contrainte.De plus, ces transformations d'ajout et de suppression sont fusionnées pour donner la possibilité de remplacer un attribut à gauche ou un attribut à droite. Afin de garantir la validité des transformations générées, nous avons testé la satisfaction du jeu de données aux contraintes générées. Seules ceux qui sont valides sont conservées afin de procéder par la suite au choix final de la réparation à considérer, qui est choisi en fonction du coût minimal de changement apporté à la contrainte erronée. Pour ce choix, nous avons proposé trois méthodes : l'une basée sur la mesure de la similarité entre les contraintes en termes d'attributs, une autre basée sur le calcul du nombre d'opérations nécessaires pour la transformation d'une contrainte à une autre, et enfin une dernière méthode impliquant l'utilisateur pour effectuer ce choix final.

Nous avons également conçu une réparation hybride pour la réparation des erreurs dans les données et les contraintes à la fois. Elle consiste à éliminer les erreurs de données et à appliquer notre processus de réparation des contraintes sur un ensemble de données propre pour garantir la consistance des réparations proposées. Par la suite, nous avons réexaminé l'ensemble de données initial et avons appliqué le processus de réparation des données que nous avons choisi afin de réparer les données pour qu'elles satisfassent les nouvelles réparations des contraintes proposées.

On a implémenté notre solution en utilisant trois modules distincts : un premier module pour l'identification de la source d'anomalie, un deuxième module pour la réparation des contraintes et un autre module pour la réparation hybride. Ces modules sont développés séparément afin de faciliter leurs maintenances ultérieures, d'améliorer la lisibilité du code et de réduire la complexité de la solution globale. Cette réalisation a été effectuée en utilisant le langage Python et en exploitant ses bibliothèques.

Afin de tester l'efficacité de notre solution, nous avons réalisé une série d'évaluations sur deux ensembles de données réels en les altérant avec différents taux d'erreurs dans les données et en incluant différents types d'erreurs dans certaines de leurs règles. Ensuite, nous avons évalué chaque module implémenté sur chaque variante des jeux de données. En particulier, le module indiquant la réparation des contraintes a été testé avec ses trois altérnatives différentes de choix final de la contrainte de réparation. Notre évaluation a été réalisée en définissant des mesures pour évaluer la précision, le rappel et la F-mesure à la fois pour les données et les contraintes. Enfin, nous avons également évalué le temps d'exécution.

La méthode du choix final de réparation des contraintes basée sur le calcul du nombre d'opérations nécessaires pour la transformation d'une contrainte à une autre a abouti à une F-mesure élevée, tandis que la méthode d'intégration de l'utilisateur a produit des résultats parfaitement ajustés. De plus, une mesure F satisfaisante a été obtenue pour la réparation hybride. et enfin le temps d'exécution a été évalué comme optimal. Dans l'ensemble, notre solution a produit des résultats efficaces pour la problématique de détection et de réparation des anomalies.

Pour conclure, nous présentons ci-dessous un ensemble de perspectives et d'améliorations potentielles à explorer à l'avenir :

- Tester notre processus de réparation hybride avec une autre méthode de réparation des données différente de celle utilisée.L'objectif est de déterminer la méthode de réparation des données qui pourrait conduire à des performances accrues pour notre solution de réparation hybride.

- Tester notre processus de réparation des contraintes sur des contraintes autres que les dépendances fonctionnelles, notamment les dépendances fonctionnelles conditionnelles, les contraintes de denial, etc. permettra d'évaluer la polyvalence et l'efficacité de notre approche dans un contexte plus diversifié.

- Tirer parti d'une base de connaissances fondamentale contenant des contraintes préalablement validées : Avant d'amorcer notre processus d'identification de la source d'anomalie, la base de connaissances est consultée. Si la contrainte violée par les données erronées est répertoriée dans la base de connaissances, la source

de l'anomalie est directement déduite que c'est une anomalie des données. En revanche, si la contrainte violée n'est pas enregistrée dans la base de connaissances, nos classifieurs entrent en action pour prédire la source de l'anomalie.cela permettra d'optimiser notre processus d'identification de la source d'anomalie.

Bibliographie

Abdellaoui, Sabrina, Fahima Nader et Rachid Chalal (2017). "QDflows : A System driven by Knowledge Bases for Designing Quality-aware Data Flows". In : *Journal of Data and Information Quality* 8.3-4, p. 283-310.

Aouragh, Si Lhoussain, Hicham Gueddah et Abdellah Yousfi (2019). "Adapting the Levenshtein distance to contextual spelling correction". In : *Expert Systems with Applications* 133, p. 113-122.

Batini, Carlo et al. (2009). "Methodologies for data quality assessment and improvement". In : *ACM Computing Surveys* 41.3, 16:1-16:35.

Berti-Equille, Laure et al. (2019). "Reinforcement Learning for Data Preparation with Active Reward Learning". In : *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.4, p. 1-24.

Berti-Equille, Lucie, Vincent Lemaire et Pierre Lenca (2019). "A survey on collaborative filtering for recommender systems". In : *arXiv preprint arXiv :1901.07285*.

Beskales, George et al. (2013). "On the relative trust between inconsistent data and inaccurate constraints". In : *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, p. 541-552.

Bohannon, Michael W. et Mark K. Jones (2005). "The Effects of Task Complexity and Task Load on the Performance of Virtual Reality Applications". In : *Virtual Reality* 9.2, p. 83-97.

Chiang, Fei et Renée J Miller (2011). "A unified model for data and constraint repair". In : *2011 IEEE 27th International Conference on Data Engineering*. IEEE, p. 446-457.

Choi, Seung-Seok, Sung-Hyuk Cha et Charles C Tappert (2010). "A survey of binary similarity and distance measures". In : *Journal of Systemics, Cybernetics and Informatics* 8.1, p. 43-48.

Chu, Wei et al. (2013). "A survey of session-based recommendation systems". In : *Knowledge and Data Engineering, IEEE Transactions on* 25.10, p. 2222-2245.

Creswell, Antonia et al. (2017). "Generative Adversarial Networks : An Overview". In : *arXiv preprint arXiv :1710.07035*.

Dallachiesa, Michele et al. (2020). "NADEEF : A Commodity Data Cleaning System". In : *IEEE Transactions on Knowledge and Data Engineering* 32.10, p. 2095-2109.

Davenport, Thomas H et Jeanne G Harris (2017). "The data-driven decision-making imperative". In : *Harvard Business Review* 95.1, p. 77-88.

Fan, Rong-En et al. (2014). "Lightgbm : A highly efficient gradient boosting decision tree". In : *Advances in neural information processing systems* 27, p. 3146-3154.

Geerts, Frank, Dirk Van den Poel et Johan Vanthienen (2013). "A survey of collaborative filtering techniques". In : *ACM Computing Surveys (CSUR)* 45.4, p. 52.

Golab, L. et al. (2008). "DelaRtice : A Probabilistic Approach to Data Cleaning". In : *VLDB*, p. 106-117.

Ilyas, Ihab F, Xu Chu et al. (2015). "Trends in cleaning relational data : Consistency and deduplication". In : *Foundations and Trends® in Databases* 5.4, p. 281-393.

Institute, Data Quality (2000). "Guidelines for Quality Data". In : *White Paper* 1.1, p. 1-10.

ISO (2019). "Management de la qualité - Lignes directrices pour la terminologie". In : *Norme ISO 80000-2:2019* 1.1, p. 1-10.

Kumar, Ashish, Akhil Yadav et Vikas Singh (2017). "Supervised Machine Learning Algorithms : Classification and Comparison". In : *International Journal of Computer Trends and Technology (IJCTT)* 48.3, p. 128-135.

Leroux, Jean-Pierre et Jean-François Lapointe (2005). "Gestion des connaissances et de l'information". In : *Revue internationale de systèmes d'information* 14.2, p. 141-162.

Liu, Wei et Xiaoyan Li (2012). "Discover dependencies from data—a review". In : *IEEE Transactions on Knowledge and Data Engineering* 24.3, p. 495-509.

Marsh, Richard (2005). "Drowning in dirty data ? It's time to sink or swim : A four-stage methodology for total data quality management". In : *Journal of Database Marketing & Customer Strategy Management* 12, p. 105-112.

Müller, Heiko et Johann Christoph Freytag (2005). *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik.

Özyurt, M.T. et M.E. Özkan (2007). "Data quality : A multi-disciplinary perspective". In : sous la dir. d'A. Kandel, J.R. Lee et J. Srivastava, p. 1-21.

Rahm, Erhard, Hong Hai Do et al. (2000). "Data cleaning : Problems and current approaches". In : *IEEE Data Eng. Bull.* 23.4, p. 3-13.

Rakotomalala, Ricco (2020). "Étude des dépendances - Variables qualitatives". Version 2.1. In : *Université Lumière Lyon 2*.

Redman, Thomas C (1998). "The impact of poor data quality on the typical enterprise". In : *Communications of the ACM* 41.2, p. 79-82.

Shaoxu Song Han Zhu, Jianmin Wang (2016). "Constraint-Variance Tolerant Data Repairing". In : *Constraint-Variance Tolerant Data Repairing*. Tsinghua National Laboratory for Information Science et Technology, p. 877-892.

Sharda, Ramesh et al. (2014). "Business intelligence and analytics". In : *System for Decesion Support*.

Sidi, Fatimah et al. (2012). "Data quality : A survey of data quality dimensions". In : *2012 International Conference on Information Retrieval & Knowledge Management*. IEEE, p. 300-304.

Vaught, Robert L. (2001). "Set Theory". In : *Department of Mathematics, University of California* 2.

Volkovs, Maksims et al. (2014). "Continuous data cleaning". In : *2014 IEEE 30th international conference on data engineering*. IEEE, p. 244-255.

Wang, Wei-Ping (1998). "Data quality : A framework for understanding, assessing, and improving data quality". In : *Communications of the ACM* 41.2, p. 86-95.

Wing, Jeannette M (2019). "The data life cycle". In : *Harvard Data Science Review* 1.1, p. 1-14.

Zhang, Peng, Boualem Benatallah et Hans A Reijers (2010). "Data quality issues and challenges : A literature review". In : *Information Management* 47.2, p. 183-196.

Annexes