REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

BETTAHER Freiha

Sur le thème

Transformation des modèles par l'exemple

Soutenu publiquement le 26 / 10/ 2020 à Tiaret devant le jury composé de :

M. KHARROUBI Sahraoui	Grade	Université de Tiaret	Président
M. SI ABDELHADI Ahmed	Grade	Université Tiaret	Encadreur
M.HATTAB Noureddine	Grade	Université Tiaret	Examinateur
M.OUARED Abdelkader	Grade	Université Tiaret	Co-Encadreur

Résumé

Avec l'avènement de l'ingénierie dirigée par les modèles (IDM) la problématique de la transformation de modèles est née. De manière simplifée, un modèle peut être défni comme un ensemble de données représentant un système et écrits dans un langage bien défni. La transformation des modèles (TM) est une étape primordiale pour l'IDM, elle consiste à transformer d'un modèle source (MS) vers un modèle cible (MC) en se basant sur des règles de transformation , connues appropriées. Nous pouvons citer par exemple le passage de formalise UML vers le modèle relationnel . Ces transformations peuvent être proposées par des experts et des professionnels dans le domaine de transformation des modèles. Dans quelques situations, nous constatons l'absence des règles de transformation pour effectuer un certain nombre d'actions. Notre objectif consiste à déduire des règles de transformation en se basant sur l'historique des transformations effectuées sur les modèles. Pour concevoir un modèle de transformation basé sur les algorithmes de machine learning (ML) , nous avons implémenté quatre types d'algorithmes (decision trees , Random forest , Naive Bayes, Logistic regression). Les résulats obtenus motivent notre initiative d'adopter une solution :d'apprentissage machine dans le domaine des transformations des modèles.

Mots-clés : Transformation des Modèles,Ingénierie Dirigée par les Modèles, Modèle Source , Modèle Cible, Machine Learning .

مُلَخَص :

مع ظهور الهندسة القائمة على النموذج ، نشأت مشكلة تحولات نموذج البرمجة بطريقة مبسطة ،يمكن تعريف النموذج على أنه مجموعة من البيانات التي تمثل نظاماً مكتوبا بلغة محددة جيداً .يعد تحويل النماذج خطوة أساسية ، فهي ترتكز على تحويل نموذج المصدر إلى نموذج مستهدف بناءاً على قواعد التحويل المعروفة المناسبة . يمكننا أن نذكر على سبيل المثال الانتقال من التشكيل إلى نموذج كيان الارتباط. يمكن اقتراح هذه القواعد من قبل الخبراء و المهنيين في مجال تحويل النموذج. في حالات قليلة ، نرى عدم وجود قواعد تحويل لأداء عدد من الإجراءات . هدفنا هو استنتاج قواعد التحويل بناءً على تاريخ التحولات التي أجريت على النماذج لتصميم نموذج تحويل يعتمد على خوارزميات التعلم الآلي ،قمنا بتنفيذ أربعة أنواع من الخوارزميات (أشجار القرار،الغابة العشوائية،الانحدار،اللوجستي) تحفز النتائج التي تم الحصول عليها مبادرتنا لاعتماد حل: التعلم الآلي في مجال تحويل النموذج.

الكلمات المفتاحية: (نموذج التحول، الهندسة النموذجية، نموذج المصدر، النموذج المستهدف، التعلم الآلي).

Abstract

With the advent of model-driven engineering (MDE) the issue of model transformation was born. In a simplified way, a model can be defined as a set of data representing a system and written in a well-defined language. The transformation of models (TM) is an essential step for the IDM, it consists in transforming from a source model (SM) to a target model (TM) based on transformation rules, known appropriate. We can cite for example the transition from UML formalization to the relational model. These transformations can be proposed by experts and professionals in the field of model transformation. In a few situations, we see the absence of transformation rules to perform a number of actions. Our goal is to deduce transformation rules based on the history of transformations performed on models. To design a transformation model based on machine learning (ML) algorithms, we implemented four types of algorithms (decision trees, Random forest, Naive Bayes, Logistic regression). The results obtained motivate our initiative to adopt a solution : machine learning in the domain of model transformations.

Keywords : Model Transformation, Model Driven Engineering, Source Modele, Target Modele, Machine Learning .

Table des matières

Liste des figures

Liste des tableaux

Glossaire

AA : Apprentissage Automatique (en anglais Machine Learning)
AG : Algorithme Génétique
ATL : ATLAS Transformation Language
FC : Fragment Cible
FS : Fragment Source
IDM : Ingénierie Dirigée par les Modèles
MC : Modèle Cible
MDA:Model Driven Architecture
MS : Modèle Source
OCL : Object Constraint Langage
OEP :Optimisation par Essains Particulaires
PG : Programmation Génétique
PLI: Programmation Logique Inductive
POO: Paradigme Orienté Objet
RS : Recuit Simulé
QVT : Qualité de Vie au Travail
SysML : Systems Modeling Language
TM : Transformation de Modèles
TMPE : Transformation de Modèles Par l'Exemple
UML : Unified Modeling Language
XMI : XML Metadata Interchange

Introduction Générale

Introduction Générale

"Nothing is particularly hard if you divide it into small jobs".
- Henry Ford (1863-1947)

Introduction Générale

1.1 Problématique

Avec l'avènement de l'ingénierie dirigée par les modèles est née la problématique de la programmation de transformations de modèles. De manière simplifiée, un modèle peut être défini comme un ensemble de données représentant un système et écrits dans un langage bien défini particulier que l'on nomme méta-modèle. On peut par exemple citer les méta-modèles UML, qui définit un ensemble de diagrammes permettant de modéliser les différents aspects d'un programme, et Entité-Relation, qui permet la modélisation d'une base de données. Une transformation de modèles peut alors être vue comme un programme modifiant un modèle pour en changer les données et/ou le réécrire dans un autre langage. Les précédents exemples cités font l'objet d'une transformation souvent utilisée comme référence dans la littérature et qui consiste à transformer un diagramme de classes UML en un modèle Relationnel.

Pour certains problèmes, nous ne connaissons pas de solution exacte pour aller d'un modèle source vers un modèle cible. Souvent le concepteur/developpeur utilise son intuition et son expérience pour réaliser une telle transformation. Cette absence de règles motive la communauté de la trasformation des modèles d'ouvrire des perspective vers la transformation des modèle à base des exemples (*Model Transformation By-Example*) [34].

La question de recherche dans ce projet de fin d'étude : **QR** : *Comment proposer une transformation du modèle dans une situation d'absence de règles ?.*

1.2 Contexte et Motivation

L'ingénierie des modèles vis-à-vis des différentes activités du cycle de développement du logiciel, et de manière plus large, pour le génie logiciel. Un modèle pour editer des vues, mais aussi pour générer du code, générer de la documentation et analyser le modèle édité. Beaucoup de points de vue lors du développement ; Beaucoup de modèles, Formaliser des liens explicites entre les divers modèles et

Automatiser les transformations. Une transformation est une opération qui prend en entrée des modèles (source) et fournit en sortie des modèles (cibles). Généralement un seul modèle source et un seul modèle cible. Afin de realiser cette trasformation, il existe énormement des formalismes (par ex.UML, XMI [1], QVT, SysML [2], EMF [3]) [62], et outils (par ex. ATL (ATLAS Transformation Language) [4], Kermeta [5], QVT [6]) [33] proposés dans le cadre de L'ingénierie des modèles.

1.3 Objectif

Dans ce mémoire, nous nous somme interessés a proposer des techniques d'apprentissage automatique permettant de fournir un modèle de transformation en se basant sur les traces effecuers par l'experiences des ingénieurs métiers. Afin de répondre à notre question de recherche, nous nous sommes fixées un ensemble d'actions qui sont :

- La formalisation du problème de la transformation du modèles à base d'exemple.
- l'identification des dimensions de la base d'exemple (Dataset).
- Collect de la connaissance des métiers d'ingénieurs dans la transformation des modèles sous formes des traces de transformations.
- Préparation de notre base d'exemples (transformation, encondage de données etc.)
- Implémentation des algorithmes d'apprentissage machine.

1.3.1 Organisation de mémoire

Notre mémoire est organisé en quatre chapitre :

1. Chapitre 1 : Ingénierie Dirigée par les Modèles
2. Chapitre 2 : Intégration des Méthodes Formelles à l'IDM
3. Chapitre 3 : Présentation de notre approche
4. Chapitre 4 : L'implémentation et la mise en œuvre de notre application
5. Enfin, la dernière partie présente une synthèse de nos contributions, des limites et des perspectives de notre proposition.

1. XML Metadata Interchange (XMI) est un standard créé par l'Object Management Group (OMG) pour l'échange d'informations de métadonnées UML basé sur XML.
2. Systems Modeling Language (SysML) est un langage de modélisation spécifique au domaine de l'ingénierie système.
3. Eclipse Modeling Framework (EMF) est un framework de modélisation, une infrastructure de génération de code et des applications basées sur des modèles de données structurées.
4. ATLAS Transformation Language (ATL) est un langage de transformation de modèles plus ou moins inspiré par le standard QVT de l'Object Management Group.
5. Kermeta est un langage de métamodélisation exécutable dont le format et le code source est ouvert. Il dispose d'un environnement de développement de métamodèles basé sur EMOF dans un environnement Eclipse.
6. La QVT, une dénomination simple pour regrouper les sujets de la vie au travail

Chapitre 1:
Synthèse Bibliographique

Chapitre

1

Synthèse Bibliographique

2.1 Introduction

Au cours des dernières décennies, le développement de grands projets d'ingénierie, toujours plus complexes, a mis en évidence la nécessité de disposer d'outils, de méthodes et de processus permettant d'en assurer la maîtrise tout au long de leur cycle de vie.
L'ingénierie dirigée par les modèles (IDM), d'abord utilisée principalement dans le domaine des systèmes logiciel, a permis plusieurs améliorations significatives dans le processus de développement de systèmes complexes en se concentrant sur des préoccupations plus abstraites autour des modèles utilisés que sur la programmation classique (le code). Il s'agit donc d'une forme d'ingénierie générative dans laquelle tout ou partie d'une application est engendrée à partir de modèles. Un modèle est une abstraction, une simplification d'un système qui est suffisante, non seulement pour comprendre le système modélisé, mais également pour garantir son bon fonctionnement.
Dans la suite de ce chapitre, nous présentons les concepts essentiels ainsi que la terminologie que nous utiliserons tout au long de ce manuscrit. Nous abordons les principes clés de l'ingénierie IDM et les différentes variantes d'ingénierie centrées sur les modèles. Nous verrons que la maîtrise de la sûreté de fonctionnement joue une part prépondérante dans la conception et le développement des systèmes complexes.

2.2 Ingénierie Système : Notions et Concepts

2.2.1 Système Complexe

Suivant la définition de l'AFIS [45] , un Système Complexe est décrit comme un ensemble organisé d'éléments en interaction permanente entre eux. Cet ensemble forme un tout cohérent et intégré pour assurer une ou plusieurs fonctions correspondant à la finalité du système. Un système est caractérisé par les propriétés qui résultent de l'interaction de ses éléments. Ces propriétés peuvent être qualifiées d'émergentes [28] car ce sont des propriétés nouvelles obtenues au niveau du système du fait des synergies existants entre ses constituants. De telles propriétés émergentes peuvent être souhaitées ou indésirables. L'objectif de l'activité de conception est d'obtenir le comportement global émergent recherché du système en maintenant les comportements émergents non intentionnels dans des limites considérées comme acceptables. Dans ce contexte, l'étude du système d'un point de vue comportemental est donc fondamentale. Il existe plusieurs types de systèmes. Dans leur classification, D. Harel et A. Pnueli [25]distinguent notamment les systèmes transformationnels, les systèmes interactifs et les systèmes réactifs. Les systèmes dits transformationnels sont des systèmes qui acquièrent des données, les traitent,

produisent des sorties puis terminent. Les systèmes interactifs sont des systèmes qui interagissent avec leur environnement, à une vitesse qui leur est propre. Quant aux systèmes dits réactifs, ce sont des systèmes qui interagissent en permanence avec leur environnement, mais à une vitesse imposée par l'environnement. Ce couplage avec l'environnement, qui est par essence non totalement maîtrisable et non totalement prévisible, rend les systèmes réactifs particulièrement difficiles à concevoir. La difficulté à concevoir un système est liée notamment à sa complexité. Celle-ci tient à au moins trois facteurs :

- **Le nombre et la nature de ses éléments.**Le nombre d'éléments peut être élevé. Il peut même être éventuellement variable au cours du temps et la nature de ces éléments peut être variée.
- **La nature de son organisation interne,**qui est liée aux relations qui existent entre ses éléments. Ceux-ci peuvent former des réseaux, des hiérarchies, etc … . L'organisation du système peut également varier au cours du temps.
- **Le couplage avec l'environnement.**Plus le système est en interaction forte avec l'environnement plus il est exposé à l'incertitude et à l'imprévisibilité de celui-ci.

2.2.2 Ingénierie Système

L'Ingénierie Système [2, 45] est une approche méthodologique pluridisciplinaire qui intègre l'ensemble des activités centrées autour du cycle de vie d'un système complexes, depuis sa définition jusqu'à son retrait de service en passant par sa conception, sa validation ou sa maintenance. L'objectif de ce processus est de contrôler la conception de systèmes dont la complexité ne permet pas le pilotage simple .

2.2.3 Modélisation des Systèmes Complexes

Dans le contexte de l'ingénierie système, le rôle principal des modèles est de permettre d'appréhender des systèmes dont la complexité est importante et d'en étudier des aspects particuliers avant même la mise en oeuvre concrète du système [46]. Les modèles permettent alors par exemple d'identifier des incertitudes quant à la spécification du système ou à l'adéquation d'une solution pour sa réalisation lorsque celles-ci sont trop complexes pour permettre un raisonnement immédiat. Par ailleurs, les modèles constituent également des supports pour communiquer des idées au sujet de la conception du système entre les différentes parties impliquées dans le projet. Enfin, les modèles servent également de guides lors des phases d'implémentation du système. Dans ce contexte, les modèles utilisés permettent donc d'anticiper sur le comportement qu'aura le système final. Ils sont appelés modèles prédictifs par opposition aux modèles dits explicatifs dont le rôle est de donner une représentation la plus fidèle possible du monde réel. Les modèles prédictifs peuvent être des modèles physiques, structurels ou comportementaux. Les modèles physiques donnent des informations sur les caractéristiques physiques du futur système (largeur, hauteur, matériaux, etc.). Les modèles structurels sont utilisés pour définir l'organisation des différents éléments du futur système. Enfin, les modèles comportementaux ont pour objectif de spécifier le comportement dynamique du futur système.
Dans la suite de ce manuscrit, nous appelons modèle une représentation prédictive du système qui met en valeur des propriétés que l'on considère intéressantes par rapport à un objectif de conception donné. En ce sens, l'activité de modélisation correspond à la construction d'un modèle en tant que représentation d'un aspect du système pour un objectif de conception donné (étude d'une partie du système, analyse de propriétés particulières, simulation du comportement, etc.). Nous nous intéressons tout particulièrement dans ce manuscrit aux modèles comportementaux.

2.3 Ingénierie Dirigée par les Modèles (IDM)

L' IDM est une approche de développement mettant à disposition des outils, des concepts et des langages afin de simplifier et de mieux maitriser le processus de développement de systèmes qui ne cessent de croitre en complexité. D'autre part, elle permet aussi d'augmenter la productivité, la qualité, la réutilisabilité et l'évolution de ces systèmes.

2.3.1 Une approche autour des modèles

L'IDM est un domaine de recherche en pleine expansion aussi bien dans le monde académique que dans le monde industriel qui considère les modèles comme les éléments de base tout au long du processus de développement [11]. L'IDM raisonne entièrement à un haut niveau d'abstraction et non plus à celui des langages de programmation classiques. Une application sera alors générée en tout ou en partie, automatiquement ou semi-automatiquement à partir de modèles, en utilisant notamment des transformations successives de ces modèles. Il s'agit donc d'une forme d'ingénierie générative, le code source de l'application n'est plus considéré comme l'élément central d'un logiciel, mais comme un élément dérivé d'éléments de modélisation.

Le processus de conception est vu comme un ensemble de transformation de modèles. Cette approche adopte le principe : "tout est modèle" en analogie avec "tout est objet" dans la vision de l'approche orienté objet[47]. Dans cette optique, des outils permettant de construire et d'exploiter ces modèles ont été développés. Ces outils sont construits autour du concept de Méta-modèle qui permet de définir un langage de modélisation particulier à un domaine ou d'intégrer plus facilement plusieurs types de modèles. Ils intègrent aussi le concept de transformation de modèles pour pouvoir, par exemple, relier un modèle à une plate forme technologique spécifique ou générer du code. Le terme IDM [35] recouvre l'ensemble de ces disciplines, dans lesquelles les modèles jouent un rôle primordial.

2.3.2 Modèle, Langage de modélisation et Méta-modèle

Un modèle est une abstraction et une simplification d'un système qu'il représente. Il offre donc une vision schématique d'un certains nombre d'éléments que l'on décrit sous la forme d'un ensemble de faits construits dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions que l'on se pose sur lui.
Il faut noter qu'il reste toutefois difficile de répondre à la question "Qu'est ce qu'un bon modèle ?". Néanmoins un modèle doit être suffisant et nécessaire pour permettre de répondre à certaines questions du système qu'il représente, exactement de la même façon que le système aurait répondu lui-même. Le modèle doit se substituer au système pour permettre d'analyser de manière plus abstraite certaines de ses propriétés [47].

La notion de modèle dans l'IDM fait explicitement référence à la notion de formalisme ou de langagede modélisation bien défini. Un langage de modélisation est défini par une syntaxe abstraite, une syntaxe concrète et une sémantique. La syntaxe abstraite définit les concepts de base du langage. La syntaxe concrète définit le type de notation qui sera utilisé pour chaque concept abstrait qui peut être graphique,

textuelle ou mixte. Enfin, la sémantique définit comment les concepts du langage doivent être interprétés par les concepteurs mais surtout par les machines.

En effet, pour qu'un modèle soit productif, il doit pouvoir être manipulé par une machine. Le langage dans lequel ce modèle est exprimé doit donc être clairement défini. De manière naturelle, la définition d'un langage de modélisation a pris la forme d'un modèle, appelé Métamodèle.

Un méta-modèle est un modèle qui définit le langage d'expression d'un modèle. Autrement dit, le méta-modèle représente (modélise) les entités d'un langage, leurs relations ainsi que leurs contraintes, c'est-à-dire une spécification de la syntaxe du langage.
Le Méta-modèle à son tour est exprimé dans un langage de méta-modélisation spécifié par le Méta-Méta-modèle. Le langage utilisé au niveau du méta-méta-modèle doit être suffisamment puissant pour spécifier sa propre syntaxe abstraite et ce niveau d'abstraction demeure largement suffisant (méta-circulaire). Chaque élément du modèle est une instance d'un élément du métamodèle.

Un modèle est dit conforme à un méta-modèle et constitue une représentation d'un système existant ou imaginaire. La relation entre un méta-méta-modèle et un méta-modèle est analogue à la relation entre un méta-modèle et un modèle. Cette relation est illustrée dans la Figure 2.1 :

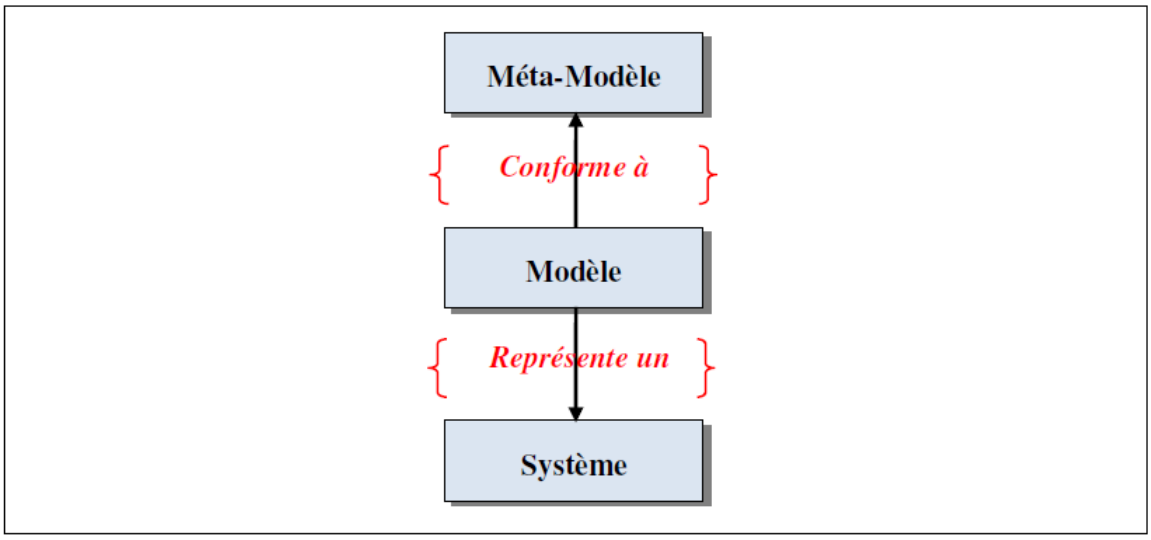


FIGURE 2.1 – Relation entre système, modèle et méta-modèle

2.3.3 Transformation de Modèles

La notion de transformation de modèles constitue l'élément central de la démarche IDM. En effet, cette notion porte sur l'automatisation de l'opération de transformation pendant le cycle de développement qui peut avoir des sémantiques différentes en fonction des utilisations : raffinement, optimisation, génération de code, etc.
La transformation de modèles est une opération qui consiste à générer un ou plusieurs modèles cibles conformément à leur méta-modèle à partir d'un ou de plusieurs modèles sources conformément à leur

méta-modèle. Elle est qualifiée d'endogène si les modèles sources et cibles sont conformes au même méta-modèle (source et cible sont dans le même espace technologique), sinon elle est dite exogène et elle se fait entre deux méta-modèles différents (source et cible sont dans deux espaces technologiques différents).

2.3.3.1 Les caractéristiques de TM

Compte tenu du rôle capital que jouent les transformations de modèles dans l'IDM, beaucoup d'efforts ont été consacrés à la mise en avant d'approches, de langages et d'outils de transformations. Afin de comparer et de catégoriser les différentes approches de TM, Czarnecki et Helsen [11] proposent un modèle de caractéristiques (feature model) dont le niveau le plus haut est illustré dans la Figure 2.2. Nous décrivons brièvement dans ce qui suit, les éléments de ce premier niveau .

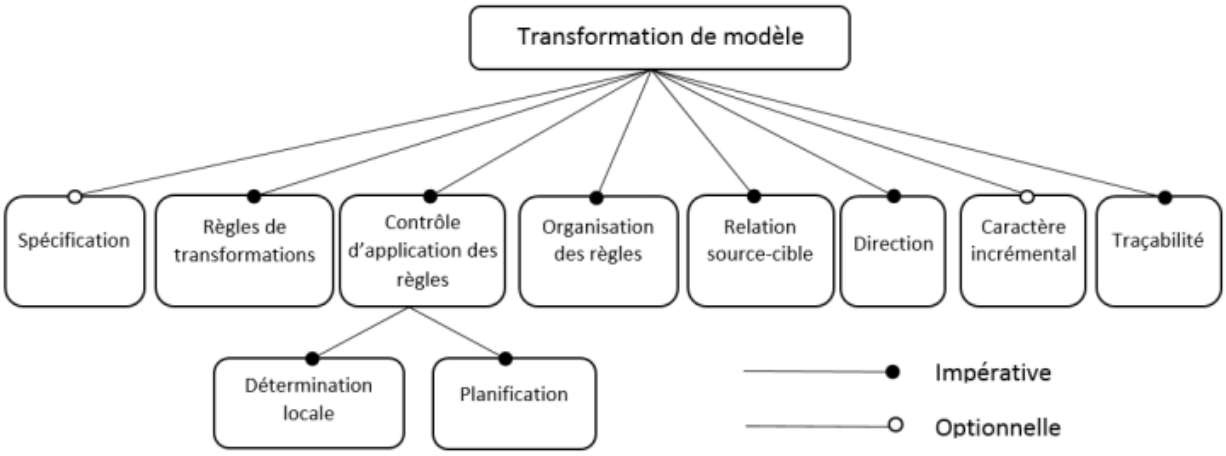


FIGURE 2.2 – Diagramme de caractéristiques des TM au niveau le plus haut

- **Spécification :** certaines approches de transformation permettent d'ajouter par exemple des pré et post-conditions aux règles de la transformation développée. Ces spécifications sont généralement exprimées en utilisant OCL [7].
- **Règles de transformation :** en fonction du langage de transformation, il est possible que les règles de transformation soient explicitement exprimées sous une forme déclarative, où qu'elles prennent la forme de fonctions ou de procédures. Les gabarits (templates) sont également considérés comme une forme de règles dégénérées.
- **Contrôle d'application des règles :** cette caractéristique englobe deux éléments : la détermination locale et la planification. La première consiste à définir, lorsqu'une règle est applicable dans plusieurs endroits dans le modèle, l'ordre des fragments sur lesquelles la règle est appliquée. La planification permet de déterminer l'ordre dans lequel sont exécutées plusieurs règles dont les conditions se trouvent satisfaites simultanément .
- **Organisation des règles :** elle permet de distinguer les mécanismes d'organisation et de réuti-

7. OCL (Object Constraint Language) est un langage informatique d'expression des contraintes utilisé par UML.

lisation mis en œuvre par l'approche tels que les modules, les paquetages, la composition et l'héritage de règles, etc.

- **Relation source-cible :** cet aspect permet de prendre en considération certains élé- ments de la taxonomie , notamment, le caractère endogène/exogène de la transformation ainsi que, dans le cas endogène, l'aspect in-place ou out-place.
- **Caractère incrémental :** il fait référence à la capacité de l'approche à propager les changements ayant lieu dans le modèle source au modèle cible. La préservation des modifications apportées au niveau du modèle cible lors de cette propagation est également un aspect particulièrement important dans certaines activités, telle que la synchronisation de modèles.
- **Direction :** certaines transformations sont unidirectionnelles, c'est-à-dire qu'elles peuvent s'exécuter dans une seule direction seulement. Les transformations multidirectionnelles peuvent, quant à elles, s'exécuter dans plusieurs directions. Dans ce dernier cas, les modèles sources peuvent devenir les modèles cibles et vice versa.
- **Traçabilité :** concerne la capacité de l'approche à enregistrer les détails de l'exécution de la transformation, en maintenant des liens entre les éléments cibles et sources. Cette fonctionnalité est utilisée, en particulier, durant les opérations d'analyse d'impact et de débogage de transformations. Dans la littérature, on peut distinguer trois types de transformations :
  1. **Les transformations verticales :** la source et la cible d'une transformation verticale sont définies à différents niveaux d'abstraction. Une transformation qui baisse le niveau d'abstraction est appelée un raffinement. Une transformation qui élève le niveau d'abstraction est appelée une abstraction.
  2. **Les transformations horizontales :** une transformation horizontale modifie la représentation source tout en conservant le même niveau d'abstraction. La modification peut être l'ajout, la modification, la suppression ou la restructuration d'informations.
  3. **les transformations obliques :** une transformation oblique combine une transformation horizontale et une verticale. Ce type de transformation est notamment utilisé par les compilateurs, qui effectuent des optimisations du code source avant de générer le code exécutable.

De manière orthogonale à cette catégorisation, selon Czarnek [11], il existe deux grandes classes de transformation de modèles : les transformations de type Modèle vers code qui sont aujourd'hui relativement matures et les transformations de type modèle vers modèle qui sont moins maîtrisées. La deuxième classe de transformation fait l'objet de plusieurs recherches au cours de ces dernières années et plus particulièrement depuis l'apparition du MDA (Model Driven Architecture).
La transformation se fait par l'intermédiaire d'un ensemble de règles de transformations décrivant la correspondance entre les entités du modèle source et celles du modèle cible. La façon d'exprimer les règles de transformation peut être déclarative, impérative ou hybride. Il est à la charge de l'utilisateur de définir le langage de transformation qui répond le mieux à ses besoins et à ses compétences. Dans la spécification déclarative, les règles décrivent ce qu'on devrait avoir à l'issue d'un certains nombre d'éléments de modèle source. Par opposition à la spécification déclarative, la spécification impérative permet de décrire comment le résultat devrait être obtenu en imposant une suite d'actions que la machine doit effectuer. Enfin, la spécification hybride regroupe à la fois la spécification déclarative et la spécification impérative.
En réalité, la transformation se situe entre les méta-modèles source et cible. La transformation des entités

du modèle source se fait en deux étapes :

— La première étape permet d'identifier les correspondances entre les concepts des modèles source et cible au niveau de leurs méta-modèles, ce qui induit l'existence d'une fonction de transformation applicable à toutes les instances du méta-modèle source.

— La seconde étape consiste à appliquer la transformation du modèle source afin de générer automatiquement le modèle cible par un programme appelé moteur de transformation ou d'exécution.

Une présentation générale des principaux concepts impliqués dans la transformation de modèles est illustrée dans la Figure 2.3.

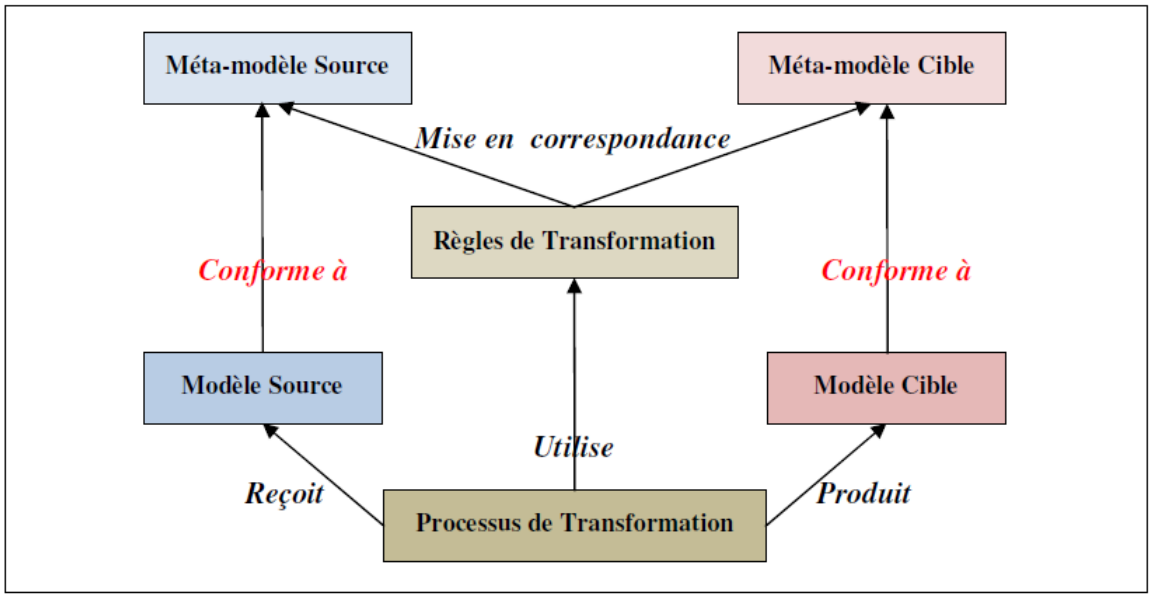


FIGURE 2.3 – Concepts de base de la transformation de modèles

2.3.4 Techniques de transformation

On peut distinguer cinq techniques de transformation de modèles :

1. **Manipulations directes :** Ces approches se basent sur la représentation interne des modèles source et cible et surune collection d'APIs pour les manipuler. L'implémentation des règles de transformation et leur ordonnancement restent à la charge du développeur en langage de programmation standard comme *Java* par exemple.
2. **Approches relationnelles :** Ces approches utilisent une logique déclarative reposant sur des relations d'ordre mathématique pour spécifier les relations entre les éléments des modèles source et cible par le biais de contraintes. L'utilisation de la programmation logique est particulièrement adaptée à ce type d'approche.
3. **Approches guidées par la structure :** Dans ces approches, la transformation se fait en deux

étapes. La première consiste à créer la structure hiérarchique du modèle cible, alors que la deuxième consiste à ajuster les attributs et les références dans le modèle cible.

4. **Transformations de graphes :** Les transformations de graphes, qui sont basées sur les grammaires de graphes [41], sont des techniques et des formalismes directement applicables à la transformation de modèles. Elles sont similaires aux approches relationnelles dans le sens où elles permettent l'expression des transformations sous une forme déclarative. Dans cette approche les modèles source et cible sont représentés sous forme de graphes. Cette notation visuelle permet aussi d'exprimer les règles de transformation sous forme graphique. Cette approche trouve son utilité dans le cas où les formalismes manipulés possèdent des syntaxes concrètes visuelles. Elle est formelle et bien fondée sur des bases mathématiques (comme la théorie des graphes et des grammaires formelles) ce qui permet de vérifier certaines propriétés de la transformation. Cette approche vise à considérer l'opération de transformation comme un autre modèle conforme à son propre méta-modèle (lui-même défini à l'aide d'un langage de méta-modélisation). Une grammaire de graphe permet ainsi en tant que formalisme de modéliser une transformation [69]. Une grande partie des outils récents adoptent cette approche avec certaines différences tels que : VIATRA, AToM3, AGG et GReAT [66].
5. **Approches hybrides :** Les approches hybrides sont une combinaison des différentes techniques. On peut notamment retrouver des approches utilisant à la fois des règles déclarative et impérative. ATL (ATLAS Transformation Language) [32] est un exemple de cette approche.
6. **Qualité des modèles :** Les qualités attendues d'un modèle dans le cadre de l'IDM sont nombreuses. Dans [46], B.Selic liste quatre caractéristiques considérées comme essentielles. Selon lui, un modèle doit être :
   - **Abstrait :**le modèle doit permettre d'omettre ou de cacher les détails que l'on ne souhaite pas considérer lors de l'étude d'une question, que ce soit pour des raisons de complexité ou de pertinence.
   - **Compréhensible :** le modèle doit être compris par les personnes qui l'utilisent. Ce constat a des implications sur le formalisme à utiliser : d'une part, celui-ci doit être en adéquation avec l'objectif du modèle (notamment en termes de niveau d'abstraction, mais également en termes de type de représentation : graphique, textuelle, etc.) et, d'autre part, la sémantique du formalisme doit être communément admise.
   - **Fidèle et précis :**le modèle doit représenter fidèlement les propriétés et les caractéristiques du système lorsqu'il est vu dans une optique spécifique.
   - **Prédictif :** le modèle doit fournir les informations nécessaires et suffisantes pour permettre de faire des prédictions justes au sujet des propriétés du système.

2.3.5 Activités liées à l'IDM

Les activités liées à la manipulation des modèles dans le cadre de l'IDM sont nombreuses et apportent chacune un ensemble de problématiques spécifiques. Nous présentons ici une liste non exhaustive de

ces activités et donnons un aperçu des problématiques qui leur sont liées.

2.3.5.1 Réalisation de modèles

La réalisation des modèles nécessite non seulement l'expertise technique pour comprendre ou concevoir la partie du système concerné mais aussi une bonne connaissance du langage de modélisation utilisé. Dans le cas de systèmes complexes, les modèles deviennent également complexes et surtout de taille importante. La qualité de l'outillage devient alors essentielle car ce sont les outils qui permettent de mieux visualiser le modèle, de s'affranchir de certains détails ou encore de vérifier automatiquement la syntaxe. Les problématiques liées à l'outillage sont variées : elles concernent la visualisation des modèles, les méthodes d'assistance, le support des langages de modélisation, etc.

2.3.5.2 Stockage de modèles

L'informatisation des modèles pose le problème de la gestion de leur persistance puis de leur accès par les utilisateurs. Les problématiques liées à cette activité concernent, par exemple, les formats de stockage, l'organisation du stockage ainsi que la gestion des méta-données concernant les modèles.

2.3.5.3 L'exécution de modèles

L'exécution de modèles comprend un éventail de tâches différentes allant de la simulation à l'exécution en temps réel en passant par l'exécution symbolique ou la génération de code. Dans ce cadre, les problèmes majeurs concernent l'exécutabilité de la sémantique des langages de modélisation utilisés. En effet, cette propriété d'exécutabilité conditionne la possibilité de pouvoir calculer, à partir du modèle, un comportement du système, ou même tous les comportements possibles de ce système.

2.3.5.4 Vérification de modèles

La vérification d'un modèle consiste à vérifier les propriétés propres de ce modèle par rapport à ce que l'on attend de lui (correction syntaxique, sémantique, etc.). La vérification de modèle recouvre différents aspects allant de la vérification de la syntaxe à la vérification de la sémantique. Les problématiques les plus complexes concernent bien sûr la vérification de la sémantique. Différentes techniques de vérification existent, avec leurs problématiques particulières : la preuve, le test ou encore le model-checking. Les techniques de preuve s'appuient sur l'utilisation de représentations formelles (à base de logique, d'automates, de Réseaux de Petri par exemple) du système. Dans ce contexte, on cherche à prouver des propriétés comme la consistance ou la complétude d'un modèle. Dans le cas de systèmes complexes, ces tâches deviennent impossibles à réaliser sur un modèle préexistant et un axe important de recherche vise à obtenir ces propriétés par construction de plusieurs modèles. Les techniques de model-checking visent à analyser le comportement spécifié par le modèle de manière à vérifier des propriétés comme la sûreté, l'atteignabilité ou la vivacité. Les problématiques dans ce contexte sont liées notamment à l'identication avec exhaustivité des états possibles du système (explosion des espaces d'état). Enn, le test est utilisé en complément du model-checking, notamment dans le cas de systèmes pour lesquels le model-checking est particulièrement inecace (lorsque les systèmes sont trop complexes par exemple). L'évaluation de la pertinence des tests est une des dicultés principales dans ce domaine,

2.3.5.5 Validation

La validation permet de vérifier que le système implémenté répond aux besoins initiaux qui ont amené à sa conception. Certaines techniques comme le test peuvent être utilisées à la fois pour la vérification et pour la validation. Dans ce cadre, les modèles permettent notamment de générer des scénarios et des vecteurs de test de façon automatique [7]. Par ailleurs, afin de minimiser les risques d'erreur de conception le plus en amont possible, les modèles peuvent être validés les uns par rapport aux autres au cours du cycle de développement. Il s'agit alors, par exemple, de vérifier que certaines propriétés sont préservées d'un modèle à un autre. Un ensemble de problématiques de ce domaine est lié au mécanisme de raffinement du modèle, par lequel on obtient un modèle plus détaillé à partir d'un autre nécessitant souvent un apport d'information.

2.3.5.6 Gestion de l'évolution des modèles

Les modèles évoluent au cours du cycle de développement du système. Ils peuvent être modifiés dans le cadre de correction d'erreurs ou d'ajout de fonctionnalités par exemple. Les problématiques dans ce contexte concernent en particulier la répercussion automatique des modifications sur les différents modèles impliqués.

2.4 les approches de l'Ingénierie dirigée par les modèles

L'IDM peut être considérée comme un domaine qui a émergé avec les technologies liées à l'instrumentation des modèles. Il existe différentes approches concrétisant différentes façons d'utiliser les modèles dans leur processus de développement des systèmes. L'approche la plus connue et peut-être la plus développée est l'approche MDA. Nous présentons cette approche dans la sous-section suivante, avant d'évoquer brièvement d'autres approches existantes.

2.4.1 L'Architecture Dirigée par les Modèles (MDA)

L'Architecture Dirigée par les Modèle est une approche de développement proposée et soutenue par **l'OMG** (Object Management Group) [56] . L'idée de base du MDA est de séparer les spécifications fonctionnelles d'un système des spécifications techniques de son implémentation sur une plateforme donnée. Autrement dit, cette approche permet de réaliser le même modèle sur plusieurs plates-formes grâce à des projections. La mise en oeuvre du MDA est entièrement basée les modèles et leurs transformations. Le cycle de développement de l'approche MDA est vu sous la forme d'un Y [19] (voir la Figure 2.4) dont les branches représentent respectivement les spécifications fonctionnelles du système et les spécifications techniques de la plate-forme cible qui, une fois intégrées, mènent à l'implémentation. Par conséquent, le fossé entre le modèle et le système n'existe plus car le modèle est le système, ou au moins le système est sensé être généré directement et automatiquement à partir du modèle.

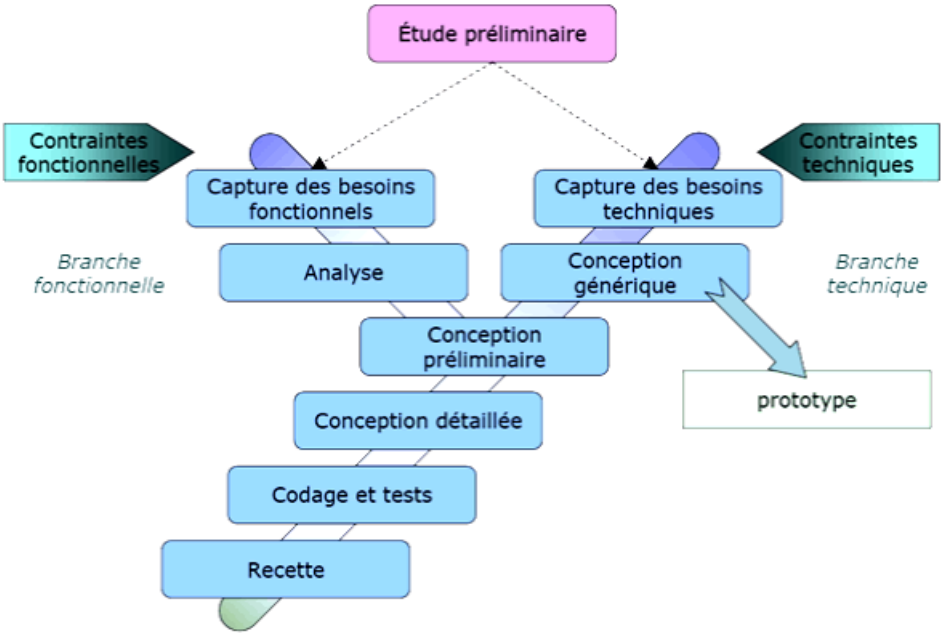


FIGURE 2.4 – Le cycle de développement en Y

2.4.1.1 Standards de l'OMG

L'approche MDA a le mérite d'être élevée au rang de standards de l'OMG (voir la Figure 2.5), dont notamment UML, OCL, MOF, XMI et CWM.
- **UML** (Unified Modeling Language) [23]Un langage visuel permettant de modéliser des systèmes à l'aidede diagrammes et de textes. Il permet aussi de décrire des architectures, des solutions ou des points devue.
- **OCL** Object Constraint Language) [50] Un ajout à UML, lui apportant la capacité de formaliser l'expression des contraintes. Il est désormais intégré à UML.

- **MOF** (Meta-Object Facility) [51] Un ensemble d'interfaces standards permettant de définir et de modifier des méta-modèles et leurs modèles correspondants. Le MOF est un standard de métamodélisation

pour définir la syntaxe et la sémantique d'un langage de modélisation, il a été créé par l'OMG afin de définir la notation UML. **XMI** (XML Metadata Interchange) [52] Un standard d'échange de métadonnées.

CWM (Common Warehouse Metamodel) [54] Une interface servant à faciliter les échanges de métadonnées entre outils, plates-formes et bibliothèques de métadonnées dans un environnement hétérogène. Il est basé sur UML, MOF et XMI.

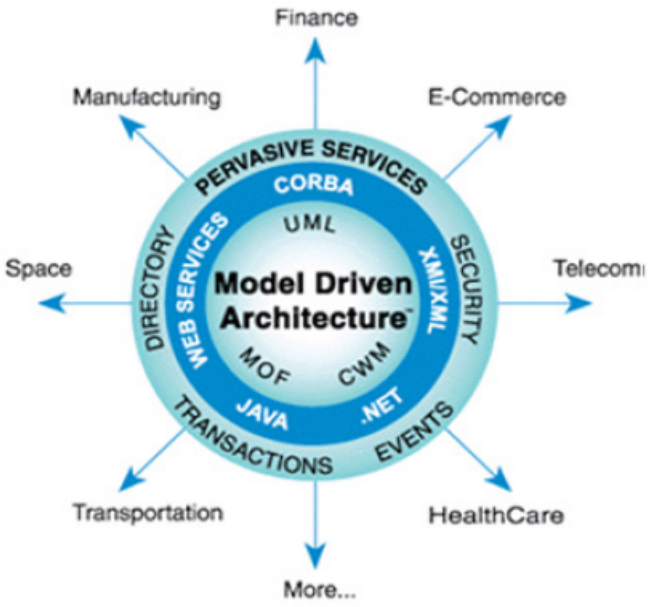


FIGURE 2.5 – Les standards de l'Architecture Dirigée par les Modèles

2.4.1.2 Transformations de modèles dans MDA

Dans l'approche MDA, l'OMG a défini plusieurs modèles qui vont servir dans un premier temps à modéliser l'application puis, par transformations successives, à générer le code de l'application. Les quatre principaux types de modèles définis dans l'approche MDA sont les suivants[36] :

- **CIM** (Computation Independant Model) : Appelé aussi modèle de domaine ou modèle métier, le CIM capture les exigences en termes de besoins et décrit la situation dans laquelle le système sera utilisé. Son but est d'aider à la compréhension du problème mais aussi de fixer un vocabulaire commun pour un domaine particulier. Dans la pratique, l'appellation "CIM" est très peu utilisée.
- **PIM**(Platform Independant Model) : Le PIM décrit le système indépendamment de la plate-forme cible sur laquelle il s'exécutera. Il présente donc une vue fonctionnelle détaillée du système, sans détails techniques. Il peut être raffiné progressivement jusqu'à intégrer des détails d'architecture spécifiques à un type de plate-forme (machine virtuelle, système d'exploitation, etc.) mais il doit rester technologiquement neutre.
- **PDM** (Platform Description Model) : Le PDM est le modèle qui décrit une plate-forme d'exécution. Il fournit un ensemble de concepts techniques représentant les différentes parties de la plate-forme et/ou les services qu'elle fournit.

— **PSM** (Platform Specific Model) : Le PSM est le résultat de la combinaison du PIM et du PDM. Il représente une vue technique détaillée du système. Il peut exister avec différents niveaux de détails. Dans sa forme la plus détaillée, il sert de base à la génération de l'implémentation.

—

La FFigure 2.6 donne une vue générale des transformations possibles entre ces différents types de modèles.

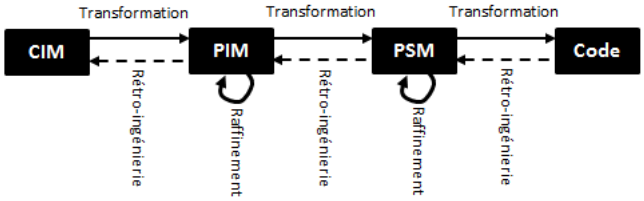


FIGURE 2.6 – Les modèles et les transformations dans l'approche MDA

Transformations PIM -> PIM et PSM -> PSM : Les transformations de type PIM vers PIM ou PSM vers PSM visent à enrichir, filtrer ou spécialiser le modèle. Il s'agit de transformations de modèle à modèle [12].

Transformation PIM PSM : La transformation de PIM vers PSM permet de spécialiser le PIM en fonction de la plate-forme cible choisie. Elle n'est effectuée qu'une fois le PIM suffisamment raffiné. Cette transformation de modèle à modèle est réalisée en s'appuyant sur les informations fournies par le PDM.

Transformation PSM Code : La transformation de PSM vers l'implémentation (le code) est une transformation de type modèle à texte [12]. Le code est parfois assimilé à un PSM exécutable. Dans la pratique, il n'est généralement pas possible d'obtenir la totalité du code à partir du modèle et il est alors nécessaire de le compléter manuellement.

Transformations inverses PSM PIM et code PSM : Ces transformations sont des opérations de rétro-ingénierie (reverse engineering). Ce type de transformations pose de nombreuses difficultés mais il est essentiel pour la réutilisation de l'existant dans le cadre de l'approche MDA.

2.4.1.3 MOF et L'architecture à quatre niveaux

L'OMG, dans le cadre de ses travaux concernant la méta-modélisation, a défini la notion de méta-méta-modèle ainsi que la standardisation d'une architecture générale décrivant les liens entre modèles, méta-modèles et méta-méta-modèles [56]. Cette architecture est hiérarchisée en quatre niveaux comme le montre la Figure 2.7.

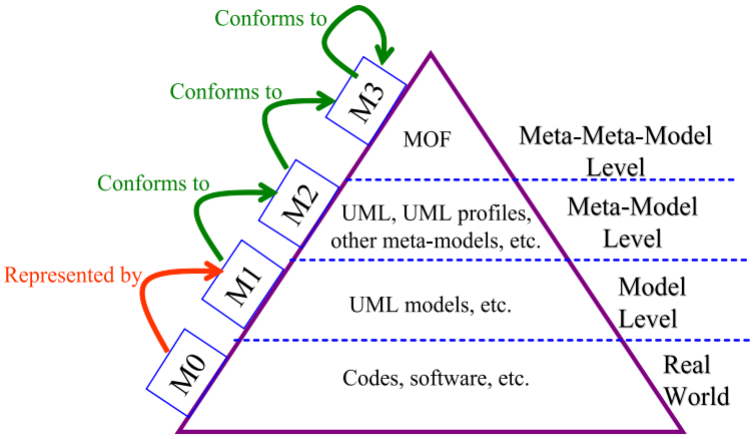


FIGURE 2.7 – Architecture à quatre niveaux

Niveau $M3$ **(MMM) :** Le niveau $M3$(ou méta-méta-modèle) est composé d'une seule entité qui s'appelle le MOF (Meta Object Facility [OMGb]). Le MOF permet de décrire la structure des méta-modèles, d'étendre ou de modifier les méta-modèles existants. Le MOF est réflexif,

Niveau $M2$ **(MM) :** Le niveau $M2$ (ou méta-modèle) définit le langage de modélisation et la grammaire de représentation des modèles $M1$. Le méta-modèle UML, qui définit la structure interne des modèles UML suivant le standard UML, fait partie de ce niveau. Les profils UML, qui étendent le méta-modèle UML, appartiennent aussi à ce niveau. Les concepts définis par un métamodèle sont des instances des concepts du MOF.

Niveau $M1$ **(M) :** Le niveau $M1$ (ou modèle) est composé de modèles d'information. Il décrit les informations de $M0$. Les modèles UML, les PIM et les PSM appartiennent à ce niveau. Les éléments d'un modèle ($M1$) sont des instances des concepts décrits dans un méta-modèle ($M2$).

Niveau $M0$ **:** Le niveau $M0$ (ou instance) correspond au monde réel. Il ne s'agit pas à proprement parler d'un niveau de modélisation. Ce niveau contient les informations réelles de l'utilisateur.
Il est important de noter que la proposition initiale d'OMG dans l'approche MDA était d'utiliser le langage UML comme unique langage de modélisation. Cependant, les limites d'UML ont rapidement été atteintes et il a fallu rapidement ajouter la possibilité d'étendre le langage UML, par exemple en créant des profiles, afin d'exprimer de nouveaux concepts relatifs à des domaines spécifiques. Ces extensions devenant de plus en plus importantes et la communauté MDA a élargi son point de vue en considérant des langages de modélisation spécifiques à un domaine.

2.4.2 Modèle de Calcul Intégré (MIC)

Les travaux autour du Modèle de Calcul Intégré (Model Integrated Computing (MIC)) [65], au milieu des années 90, ont proposé une approche de développement logiciel basée sur des modèles spécifiques au domaine d'application spécifique. La motivation de cette approche était d'apporter une méthodologie de développement ainsi que des outils logiciels comme support.
La méthodologie proposée est décomposée en deux phases. La première phase consiste à analyser le

domaine d'application afin de trouver les paradigmes de modélisation les plus appropriés et de définir avec précision le langage de modélisation qui sera utilisé. Un outil automatique peut alors utiliser ces informations pour générer automatiquement un environnement de modélisation dédié au domaine. Cet environnement est utilisé directement dans la seconde phase qui consiste à modéliser l'application désirée. La plate-forme GME (Generic Modeling Environment) [13] est une implémentation de la méthodologie définie par l'approche MIC

2.4.3 Les usines logicielles (Software Factories)

Les usines logicielles "Software Factories" [22] représentent la vision que Microsoft a proposée pour l'ingénierie dirigée par les modèles. Le terme d'usine logicielle fait référence à une industrialisation du développement logiciel similaire aux lignes d'assemblage dans l'industrie qui se base sur les idées suivantes :

- Les lignes d'assemblage ne fabriquent généralement qu'un seul type de produit avec de faibles points de variation. Ainsi les lignes de production dans l'automobile ne produisent qu'un seul type de voiture, avec des variations possibles pour la couleur ou les combinaisons d'options.

- Les ouvriers sont souvent spécialisés. Si certains ont parfois des activités relativement variées, elles ne couvrent jamais la totalité des activités de la ligne d'assemblage.

- Les outils utilisés de la ligne d'assemblage sont très spécialisés et très automatisés. Ils ne peuvent généralement pas être réutilisés sur d'autres lignes de production que celle pour laquelle ils ont été conçus, ce qui permet d'atteindre des degrés d'automatisation importants.

- Les composants à assembler ou à usiner proviennent souvent de tierces parties (fournisseurs). Par exemple, les lignes de production automobile assemblent généralement des pièces qui sont produites dans d'autres usines.

L'approche de Microsoft propose d'adapter ces idées au développement de systèmes logiciels. Selon les deux premiers points, les produits logiciels ainsi que les développeurs au sein des équipes de développements devraient être hautement spécialisés. Le troisième point correspond également à l'utilisation d'outils de développement spécialisés au domaine d'application. Ce principe inclut les langages de modélisation, les logiciels d'assistance ainsi que les transformations. Le dernier point pousse à la réutilisation de composants déjà développés ailleurs.

En résumé, une "Software Factory" est un environnement de développement configuré pour produire rapidement un type spécifique d'applications. L'environnement de développement intégré "Microsoft Visual Studio.NET" 2008 met en application ces idées et propose une plateforme de développement qui peut être configurée pour cibler un domaine particulier.

2.4.4 Synthèse

Les approches à base de modèles précédentes dans les sections précédentes ont permis l'émergence de l'IDM dont l'objectif est de mettre l'accent sur un certain nombre d'idées clef de chaque approche. Le MDA a mis en évidence l'intérêt de séparer les spécifications fonctionnelles d'une application de son implémentation sur une plate-forme donnée. Le MIC a fait émerger la notion de générateur d'éditeurs de modèles pour un domaine spécifique. Les usines logicielles ont mis en évidence la notion de chaîne de transformation d'artéfacts.

2.5 Processus de Vérification en IDM

Dans le contexte de l'ingénierie dirigée par les modèles, le développement des systèmes complexes fait appel à différentes techniques de modélisation qui dépendent :

- Du domaine du système ou du sous système.
- Des différentes phases du cycle de développement.
- Des différents niveaux d'abstraction aux quels le système est étudié.
- Des différents aspects spécifiques à analyser et à vérifier lors de la conception.

Dans le but d'analyser le comportement modélisé pour assurer le bon fonctionnement du système et détecter d'éventuels problèmes, différentes techniques telles que la simulation, le test ou les méthodes formelles sont alors utilisées dès les premières étapes de sa conception. Dans ces techniques, les modèles peuvent être utilisés comme support pour ces activités de vérification et de validation. Certaines analyses peuvent être conduites directement en utilisant les modèles. Par exemple, des simulations ou des tests sur des modèles comportementaux sont souvent favorisés. En effet, il s'agit d'une méthode simple et relativement peu coûteuse, consistant à générer des cas de tests pertinents pour vérifier les scénarios attendus. En revanche, même pour un système d'une taille raisonnable, l'analyse des comportements possibles ne peut pas être exhaustive et l'on court le risque de ne pas identifier des situations potentiellement dangereuses pour le système.
D'autres analyses nécessitent la transformation des modèles réalisés dans des formalismes formels adaptés à un type de vérification formelle désirée. Les méthodes formelles, telles que la preuve de théorème, les réseaux de Petri ou le Model-Checking, s'appuient sur un cadre mathématique précis et non ambigu permettant de modéliser à la fois le système et les propriétés qu'il doit vérifier. Par exemple, la déduction d'un réseau de Petri global modélisant le comportement du système à partir des modèles comportementales permet de mettre en oeuvre les outils de preuve formelle associés aux réseaux de Petri.

2.6 Conclusion

Dans ce chapitre nous avons défini le contexte dans lequel se place ce mémoire est détaillé les principaux concepts du domaine de l'ingénierie des modèles. Nous avons mis en évidence le rôle central des modèles et de l'outillage sous-jacent dans le processus de conception des systèmes. Nous avons aussi identifié et mis en valeur les problématiques actuelles liées à la manipulation des modèles et au

besoin de s'assurer de leur qualité tout en réduisant les coûts et le délai de production. Dans le cadre de l'approche IDM, nous avons présenté différentes techniques de traitement des modèles parmi lesquelles nous avons détaillé la méta-modélisation ainsi que les techniques de transformation de modèles. Ces techniques sont aujourd'hui des pivots majeurs pour l'innovation dans les domaines de recherche liés à la modélisation des systèmes complexes.

Chapitre 2:
Transformation de modèles par l'exemple

Chapitre

2

Transformation de modèles par l'exemple

3.1 Introduction

L'objectif de ce chapitre est d'explorer les travaux qui traitent l'apprentissage de transformations dans le cadre de l'IDM. Ce chapitre est consacrées à l'apprentissage des transformations de modèles. Nous allons passer en revue les différentes contributions qui s'inscrivent dans le cadre de TMPE. Nous verrons deux familles d'approches d'apprentissage existantes dans la littérature, à savoir, les approches d'apprentissage par l'exemple et celles par démonstration. Enfin, la dernière partie de ce chapitre prend la forme d'une synthèse qui résume les caractéristiques clés et les limites des contributions identifiées.

3.2 Principe de TMPE

L'écriture de transformations de modèles est une tâche difficile qui nécessite parfois beaucoup d'efforts et de temps. Dans ce sens, la transformation de modèle par l'exemple (TMPE) semble être une solution appropriée à cette problématique. De manière analogue à la programmation par l'exemple[24] et à l'interrogation par l'exemple [73], les approches de TMPE ont pour but d'apprendre automatiquement un programme de transformation à partir d'artéfacts fournis en guise d'exemples.
Il existe deux axes de recherche portant sur l'apprentissage de transformations de modèles par l'exemple : les approches par l'exemple proprement dites (TMPE), ainsi que celles par démonstration (TMPD). Nous explorons dans cette section la première catégorie, tandis que nous aborderons la seconde dans la section suivante. Les travaux de TMPE œuvrent à dériver automatiquement un programme de transformation à partir d'un ensemble d'exemples fournis en entrée. Chaque exemple est une paire de modèles constituée d'un modèle source et d'un modèle cible. En plus des exemples, la majorité des approches de TMPE exploitent des traces fines de transformation. Ces traces sont des liens, de plusieurs-à-plusieurs, qui associent un groupe de n éléments sources à un groupe de m éléments cibles. La Figure 3.1 montre trois traces. Dans chaque trace, le fragment cible (en bas de la figure) est associé au fragment source (en haut de la figure). Les traces de transformation sont généralement définies par des experts du domaine.

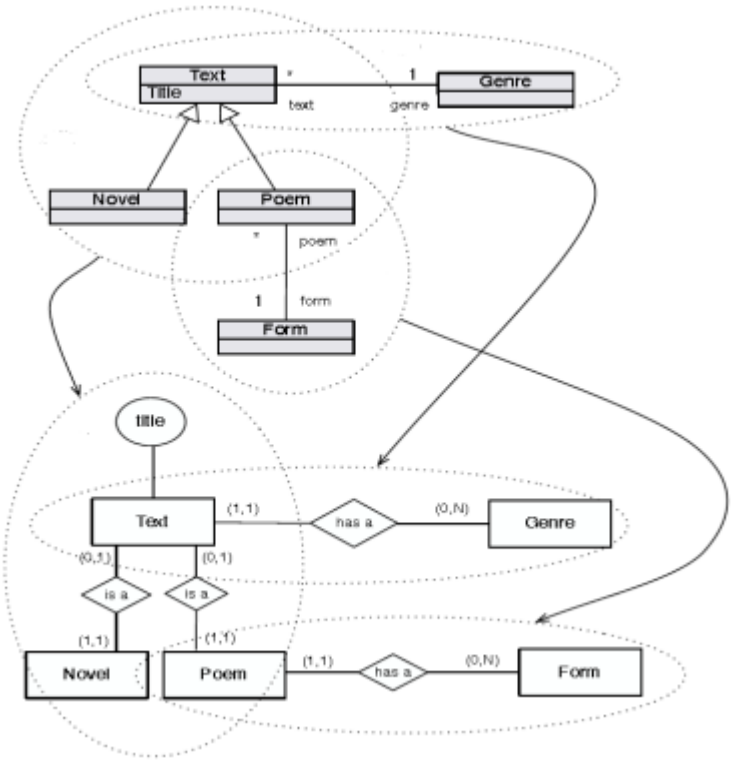


FIGURE 3.1 – Un exemple de traces entre un diagramme de classes et un modèle entité.

3.3 Appproches existantes

3.3.1 Approche de Varró

En 2006, Varró [70] propose une première approche pour l'apprentissage de transformations à partir de paires d'exemples en utilisant un algorithme ad-hoc basé sur les graphes. L'algorithme prend en entrée des paires de modèles interreliés (accompagnés de traces) et dérive un ensemble de règles de transformation de type (1,1). Le processus de dérivation est itératif et interactif, à chaque itération, les règles produites sont raffinées par l'utilisateur. La contribution est ensuite automatisée davantage par Balogh et al. [6] en utilisant la programmation logique inductive (PLI). Cette nouvelle approche, bien que toujours itérative et incrémentale, permet de dériver des règles de transformation plus complexes (de type (n,m)).

3.3.2 Approche de Wimmer

Durant la même période Wimmer et al. [72] propose une approche similaire pour dériver des transformations sous la forme de règles de type (1,1) exprimées en ATL. La contribution en question est également itérative et incrémentale, elle diffère cependant des deux contributions précédentes sur deux aspects. D'abord, les traces de transformation exploitées sont spécifiées dans une syntaxe concrète plutôt qu'abstraite. Ensuite, le processus de dérivation se fait selon une approche orientée objet, contrairement à celui de Varró où des graphes sont utilisés. La contribution de Wimmer est également améliorée

dans des contributions subséquentes, par Strommer et al. [63, 64], où un langage pour la définition de correspondances de type (n,m) est présenté, jumelé à un algorithme de raisonnement pour la dérivation de règles ((nm) également) à partir des correspondances exprimées. L'usage d'opérateurs de chaînes tel que la concaténation est également brièvement mentionné dans l'une des deux contributions.

3.3.3 Approche de Garcia-Magarino

Un autre travail qui s'inscrit dans le contexte des transformations de modèles par l'exemple est celui de Garcia-Magarino et al.[21] qui propose un algorithme permettant de générer des règles de transformation de type (n,m), vraisemblablement dans plusieurs langages de transformation, à partir d'un ensemble de paires de modèles sources et cibles interconnectés (agrémentés par des traces). À l'instar de [70], l'approche utilise des graphes. Afin de pallier le fait que certains langages de transformations ne permettent pas d'écrire des règles de plusieurs à plusieurs, les règles sont d'abord dérivées dans un langage de transformation générique avant d'être transformées dans le langage de transformation souhaité .

3.3.4 Approche de Kessentini

Kessentini et al. [37, 39] utilise des analogies pour effectuer une transformation. Contrairement aux contributions citées plus haut, cette approche ne produit pas de règles de transformation, mais dérive plutôt le modèle cible directement à partir du modèle source en considérant la transformation de modèles comme un problème d'optimisation. Le problème tel que posé est abordé en utilisant l'optimisation par essaims particulaires (OEP) dans la première contribution et une combinaison de OEP et du recuit simulé (RS) dans la deuxième. L'approche d'apprentissage est ensuite améliorée dans [38] où des règles de transformation sont produites à partir de modèles sources et cibles qui ne sont pas accompagnés de traces de transformation. L'approche est validée sur une transformation de modèles comportementaux (diagramme de séquence vers réseau de Petri colorés).

3.3.5 Approche de Dolques

Une autre contribution de TMPE qui ne produisait pas de règles de transformation initialement est celle de Dolques et al. [14]. Ce travail se base sur l'analyse relationnelle de concepts (ARC), une variante de l'analyse formelle de concepts, pour classifier les éléments sources et cibles ainsi que les correspondances fournies en entrée. Les patrons identifiés sont organisés dans des treillis partiellement ordonnés et sont ensuite analysés pour filtrer les plus pertinents. L'approche est également étendue par Saada et al. [59, 60] où des règles exécutables sont produites à partir des patrons filtrés. Dans cette dernière contribution, les règles sont exprimées en Jess. **Jess :**
C'est un environnement de transformation de modèles. Jess est un moteur de règles qui permet de stocker en mémoire des connaissances exprimées sous la forme de faits (facts). Il est possible ensuite de raisonner sur ces connaissances à l'aide de règles déclaratives. Jess applique les règles sur la base de faits en utilisant un filtrage par motifs basé sur l'algorithme Rete [18]. Il est possible d'associer la notion de fait en Jess au concept d'objet dans le paradigme orienté objet (POO). Chaque fait peut contenir plusieurs attributs, appelés « slots ». Jess offre également la possibilité de définir des gabarits de faits (template) qui correspondent à la notion de classe dans le POO. Afin d'utiliser Jess pour la transformation de modèles, nous exprimons chaque modèle sous la forme d'un ensemble de faits. Chaque élément du modèle est un fait dont les slots sont des attributs ou des références vers d'autres éléments. Les méta-modèles sont

exprimés, quant à eux, sous la forme de gabarits de faits. La Liste 3.1 présente un exemple trivial, où sont représentés, un méta-modèle et un modèle comportant un seul élément.

```
1        (deftemplate class (slot name (type STRING)))
2(assert (class (name Personne)))
```

Listing 3.1 – Exemple d'une règle exprimée en Jess

Une transformation de modèles est représentée sous la forme d'un ensemble de règles Jess. À l'instar des transformations basées sur les graphes, chaque règle comporte une partie gauche (G) qui exprime les conditions nécessaires au déclenchement de la règle, ainsi que d'une partie droite (D) qui consiste en l'assertion d'éléments cibles. L'initialisation des attributs cibles par des attributs sources se fait en utilisant le même nom de variable. La Liste 3.2 présente une règle qui transforme des éléments « Classe » en des éléments « Table » du même nom.

```
1        (defrule Class 2Table
2(class (name ?c))
3=>
4(assert (table (name ?c)))
5)
```

Listing 3.2 – Exemple d'une règle exprimée en Jess

Outre les faits et règles, Jess fournit un ensemble de fonctions mathématiques et de manipulation de chaînes. Cet ensemble peut être étendu par des fonctions définies par l'utilisateur. De plus, il est possible de créer en Jess des primitives de navigation qui parcourent les faits à la recherche de relations particulières.

3.3.6 Approche de Faunes

Les travaux les plus récents dans le contexte de la TMPE sont ceux de Faunes et al. [15, 16] dans lesquels la programmation génétique (PG) est utilisée pour faire évoluer une population de transformations sur plusieurs générations jusqu'à produire la transformation attendue. Le processus de dérivation prend en entrée des paires d'exemples de modèles sources et cibles uniquement (sans traces de transformation) et produit en sortie des règles exécutables de type (n,m). L'approche est ensuite améliorée par la contribution de Baki et al.[5] où le programme génétique tente d'apprendre simultanément les règles de transformation ainsi que le contrôle d'exécution qui doit être exercé sur cellesci pour former un programme de transformation correct. Cette seconde version permet également de dériver des règles de transformations plus complexes en incluant la né- gation de conditions, l'usage de primitives de navigation ainsi que la considération des types et des domaines de définition lors de la construction du modèle cible.

3.3.7 Approche de Baki

cette approche prend en entrée un ensemble de pairs d'exemples de modèles sources et cibles accompagnés de traces de transformations capturant la correspondance entre les différents fragments sources et cibles. Le processus d'apprentissage proposé dans cette dernière permet alors de dériver un

Approche	Algorithme	Entrée	Sortie
Varró	Ad-hoc	Exemples et Traces	Règles
Wimmer	Ad-hoc	Exemples et Traces	Règles
Balogh	PLI	Exemples et Traces	Règles
Kassentini	OEP/OEP-RS	Exemples	MC
Deloques	ARC	Exemples et Traces	Règles
Faunes	PG	Exemples	Règles
Baki	PG	Exemples	Règles

Tableau 3.1 – Caractéristiques des approches MTPE actuelles.

programme de transformation en trois étapes principales. Dans un premier temps, les traces fournies sont complétées et analysées afin de construire des pools d'exemples cohérents. Ensuite, un ensemble de règles est dérivé pour chaque pool en utilisant un programme génétique. Durant cette étape, les règles sont également traitées pour éliminer d'éventuelles erreurs ou incohérences. Enfin, la troisième et dernière étape consiste à fusionner les groupes de règles apprises en un programme de transformation qui est affiné en utilisant la méthode du recuit simulé. Le processus d'apprentissage de cette approche est validé sur sept (7) cas de transformations pré- sentant divers caractéristiques et degrés de complexité. Les résultats obtenus indiquent que les transformations les plus communes sont parfaitement apprises. Par ailleurs, l'apprentissage des cas de transformations les plus complexes permet de générer des modèles cibles très similaires à ceux attendus [5].

3.4 Synthèse et Positionnement de notre travail

Nous avons synthétiser les travaux qui traitent le problème de transformation de modèles par l'exemples. Ces travaux sont basés sur des approches heuristiques comme par exemple les algorithmes génétiques [5, 16],des approches basées sur les graphes et des approches basées sur l'apprentissage automatique . Ces travaux traitent la problématique liée à la transformation de modèles comme l'absence des règles , la dérivation des règles , l'analyse de dépendances entre les attributs , le fusionnement ,création ..,etc.

Nous avons comparés ces travaux en terme de trois critères :

1. Le type algorithme utilisé comme (Ad hoc, PLI,PG..,etc)
2. Les entrées de ce problème .
3. La solution générée par l'approche .

A notre connaissance , ces travaux proposent une approche pour la résolution de ce probème(transformation de modèles par l'exemples .Ces travaux ne prennent pas en considération les informations détaillées sur les fragments , et ne sont pas orientées vers des outils supports dédiée par les utilisateurs finaux (Designers,Développeurs,Architectes logiciels, ..,) Dans ce travail nous nous sommes intéresser au problème de transformation de modèles dirigée par les données , nous allons considérer ce problème comme étatnt un problème d'apprentissage supervisé basé sur des traces de transformations qui décrit la spécification détaillée de modèle d'entrée avec les différentes actions qui peuvent être effectuer sur les fragments . Les modèles proposés sera baser sur des algorithmes orientée vers d'apprentissage supervisé comme par exemple (Decision Tree , Ranom Forest ,..,etc).

3.5 Conclusion

Dans ce chapitre nous avons cité les différentes contributions dans le cadre de l'apprentissage de transformation de modèles par l'exemple .Et pour le résume de ces approches , on a classées dans un tableau comparatif qui décrit les approches les plus utilisées dans TMPE et résumer dans ce chapitre . Pour conclure , nous avons proposés une approche basée sur l'apprentissage automatique supervisé , parce que le nombre de classes sont connue pour l'entrainement . on va détailler cette dernière dans le prochain chapitre .

Chapitre 3:
Démarche adoptée

Chapitre

3

Démarche adoptée

4.1 Introduction

La transformation de modèle par l'exemple (TMPE) semble être une solution appropriée du problème de transformation des modèles . De manière analogue à la programmation par l'exemple et à l'interrogation par l'exemple , les approches de TMPE ont pour but d'apprendre automatiquement un programme de transformation à partir d'artéfacts fournis en guise d'exemples. La machine peut utliser une série d'exemples avec des traces détaillées, la signature et les composantes de modèle. Dans ce chapitre nous allons formaliser note problème de recherche afin de spécifier les entrées sorties, et nous allons proposer les algorithmes d'apprentissage automatique (machine Learning) afin de trouver le modèle qui transforme une nouvelle instance de modèle source vers le model cible correspond.

4.2 Vue globale de notre approche :

La Figure 4.1 illustre la vue globale de notre approche . Nous avons proposer un système de fragmentation à base d'exemples , constitué de trois services pricipales :

1. **Service 1** : il reçoit en entrée un modèle source globale .Il décompose ce modèle en fragments en se basant sur l'hypothése suivante : pour chaque fragment , il doit contenir maximum une relation .Les fragments générés seront les etnrés de service 2 .

2. **Service 2** : se service sagit par la transfoormation de chaque fragment généré par le premier service vers un fragment qui correspond en se basant sur des algorithmes d'apprentissage automatique(Decision Tree, Random Forrest,..,etc).

3. **Service 3** : une fois que les fragments cibles sont générés en se basant sur le service précédent .Ces fragments doivent être fusionner pour obtenir le modèle cible globale.

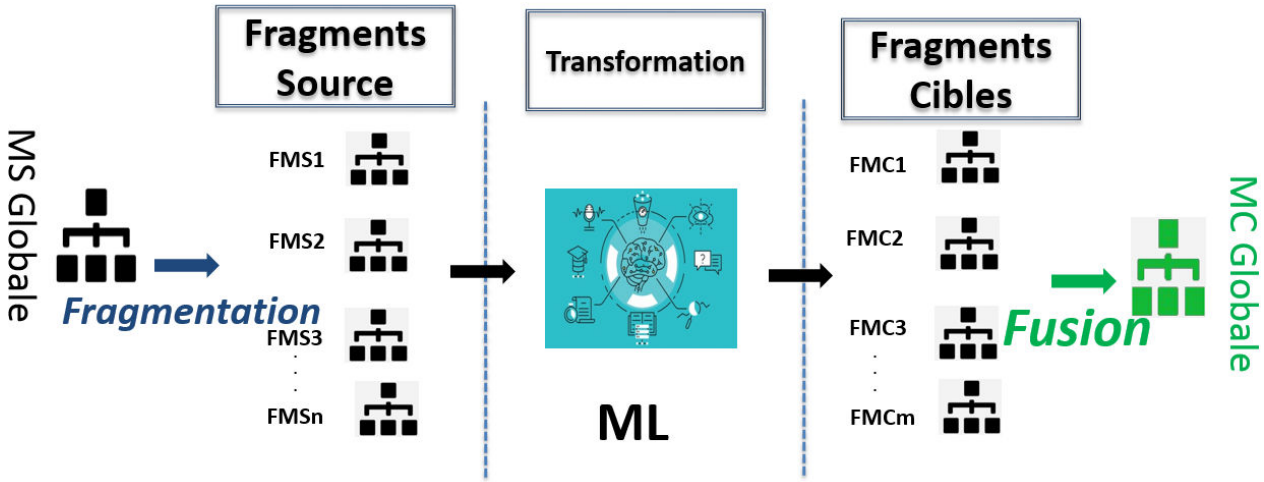


FIGURE 4.1 – Vue globale de notre approche

4.3 Etapes de l'approche

Pour répondre au problème du projet de fin d'étude, nous allons proposer une approche basé sur l'apprentissage automatique , la Figure 4.2 ci-dessus illustre notre approche [57]

1. Définir le problème .
   - Ientifier les objectifs.
   - Identifier les objectifs d'apprentissage automatique.
2. Identifier dataset .
   - Accéder aux données nécessaires.
   - Collecter et comprendre les données.
3. Preparer et prétraiter .
   - Sélectionner les données requises.
   - Nettoyer / formater les données si nécessaire.
4. Citer les modèles de donnés .
   - Sélectionner les algorithmes .
   - Construire des modèles prédictifs.
5. Entrainer et tester le modèle .
   - Entraîner le modèle avec des exemples d'ensembles de données .
   - Tester et itérer le modèle .
6. Le déploiement et la vérification .
   - vérifier le modèle finale.
   - préparer la visualisation et déployer.

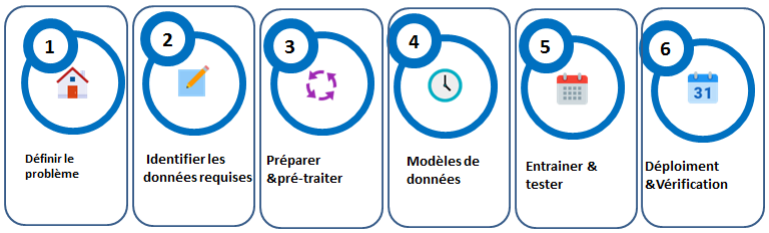


FIGURE 4.2 – Etapes de notre approche

4.3.1 Illustration du problème

Pour que le problème soit machinale, les caractéristiques du modèle source et cible puis être extraire avec les traces des actions effectuées. La Figure 4.3 ci-dessus illustre notre problème.

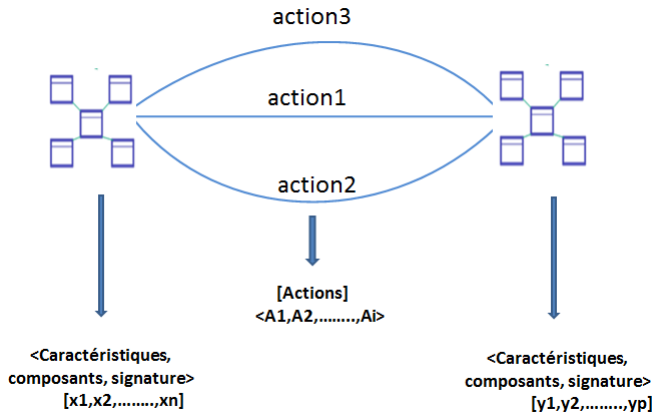


FIGURE 4.3 – Illustration de notre problème

4.3.2 Formalisation du problème

Dans cette section , nous allons formaliser notre problème pour spécifier les entrés-sorties.

Entrées :

— modèle source MS (x_1,x_2,...,x_n)

— modèle cible MC (y_1,y_2,.....,y_p)

Sortie :

— Les actions(Action1 ,Action2,Action3,.., etc) Figure 4.4

4.3.3 Les entrées et sorties du problème

Entrées : Soit une paire d'exemples de modèles source et cible notée (MS, MC), où chaquecorrespondance estune paire de fragments (c a d sous ensemble de MS et MC) source et cible .

— Fi= (FSi ; FCi) , i∈ 1..*m*..

Un fragment source (resp. cible) est un ensemble d'éléments sources (resp. cibles).formellement : (FSi, FCi) = (x1,x2,. . . .,xn), (y1,y2,. . . . . ,yp) .

— FSi : est un vecteur de caractéristique de n entités .

— FCi : est un vecteur de p entités .(Les éléments de notre problème sont présentés dans le Tableau 4.1 :)

Caractéristiques	Description
MS	Modèle source (ex. modèle en classe)
MC	Modèle cible (ex. modèle schèma relationnel)
FSi	Fragment source (maximum deux classes =>une relation)
FCi	Fragment cible
x1,x2,. . . .,xn	Les variables de fragment sources
y1,y2,. . . . . ,yp	Les variables de fragment cible

Tableau 4.1 – La description des entrés du problème

Sortie :

La sortie du problème consiste à identifié les différentes actions effectués sur le modèle source (MS) . Ces actions transformes Ms vers MC(modèle cible).Nous avons suggéré les différentes actions dans la Figure 4.4 .

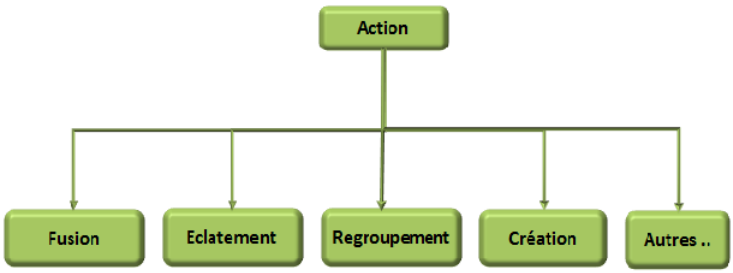


FIGURE 4.4 – Les actions (sortie) de notre problème

Fusion :

cette action ce fait quand on a deux classes du modèle source séparé par une association du cardinalité (1-1) , l'exemple ci-dessus consiste à fusionner les classes reliées à travers une association(1-1) dans une seule table , lorsqu'une des deux classes ne possède pas d'autres liens (c.-à-d., qu'elle n'est pas impliquée dans une autre association ou relation d'héritage) . Dans la Figure 4.5 la classe Adresse ne possède pas de lien autre que celui avec la classe Personne, par conséquence, une seule table est crée [4].

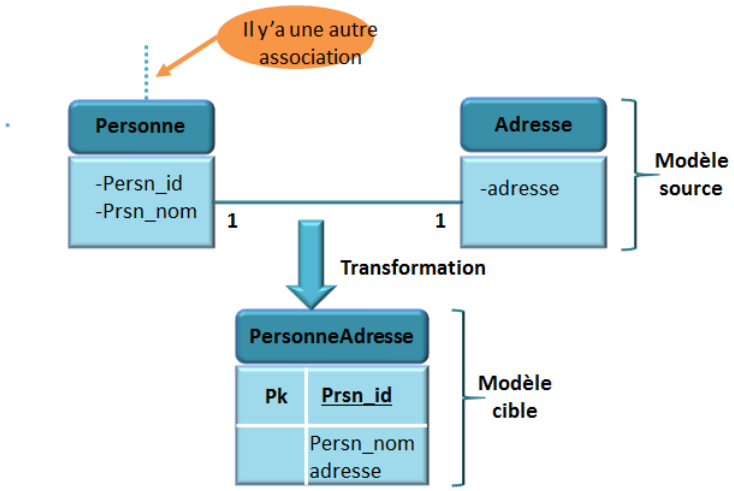


FIGURE 4.5 – Fusion

Création :
Si ce n'est pas le cas de fusion comme nous l'avons détaillés au dessus , une table est créée pour chaque classe .la situation est illustrée par l'exemple de la Figure 4.6 [4].

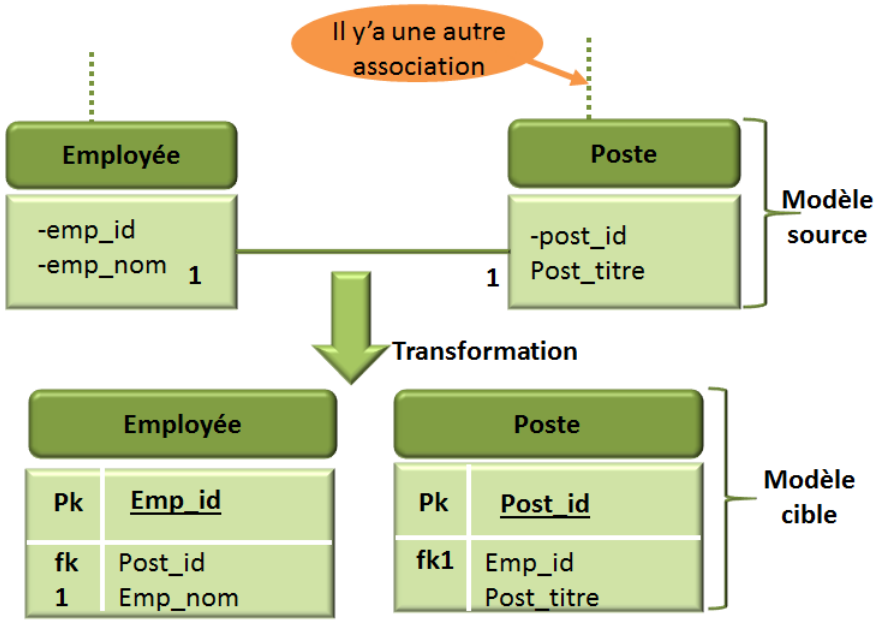


FIGURE 4.6 – Un fragment source transformé en une deux tables

Inversement, dans la Figure 4.5 , les deux classes employée et Poste possèdent au moins unautre lien. Ainsi, deux tables doivent être créées et reliées par des clés étrangères.
Eclatement :cette action est rare , C'est à dire, une seule classe du modèle source qui se transforme en deux tables différentes [4].

4.4 Définition des concepts

4.4.1 Intelligence artificielle

Intelligence artificielle, souvent abrégée en IA, est définie par l'un de ses créateurs, Marvin Lee Minsky, comme : " la construction de programmes informatiques qui donnent des tâchesde façon plus satisfaisante par des êtres humains car elles demandent des processus mentauxde haut niveau" tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critiquée [1].
Elle est utilisée dans diffèrent domaines d'application et des usages potentiels tels que :la compréhension du langage naturel, la reconnaissance visuelle, robotique, un système autonome. [53] L'I.A. est, de ce double point de vue, une science expérimentale : expériencessur ordinateurs qui permettent de tester et d'affiner les modèles exprimés dans les programmessur de nombreux exemples ; observations sur l'homme (en général le chercheur lui-meme) pourdécouvrir ces mdèles et mieux comprendre le fonctionnement de l'intelligence humaine.[42]

4.4.2 Apprentissage Automatique (ML)

Par principe, les machines, les ordinateurs et les programmes ne fonctionnent que de la façon dont vous les avez préalablement configurés : « si le cas A survient, activer B ». Cependant, nos attentes à l'égard des systèmes informatiques modernes sont de plus en plus élevéeset les programmes ne peuvent prévoir tous les cas imaginables et imposer une solution à l'ordinateur. Il est donc nécessaire que le logiciel prenne des décisions de manière indépendanteet réagisse de manière appropriée aux situations inconnues. Mais des algorithmes doivent être disponibles pour permettre aux programmes d'apprendre. Cela signifie que dans un premier temps, qu'il doit être alimenté avec des données et, qu'il puisse dans un second temps faire des associations[29].

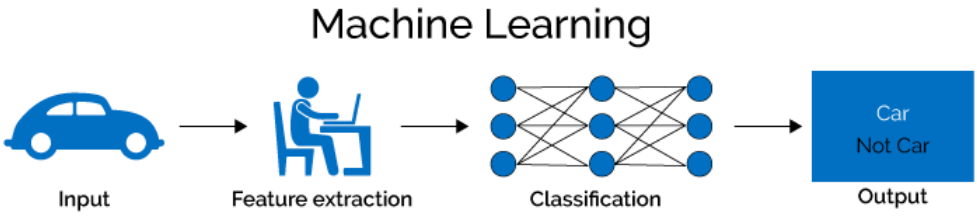


Figure 4.7 – Machine learning

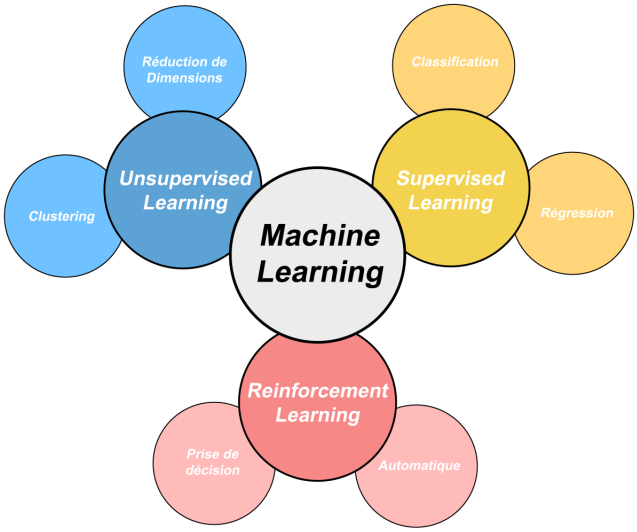


FIGURE 4.8 – Les types de machine learning

L'objectif visé est de rendre la machine ou l'ordinateur capable d'apporter des solutions à des problèmes compliqués, par le traitement d'une quantité abusif d'informations. Cela offre ainsi une possibilité d'analyser et de mettre en évidence les corrélations qui existent entre deux ou plusieurs situations données, et de prédire leurs différentes implications.[29]
L'apprentissage automatique est un sous-domaine de l'intelligence artificielle (IA). En général, l'objectif de l'apprentissage automatique est de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par les tout le monde.[29]

4.4.3 Les types d'apprentissage automatique

Dans cette section , on va citer les différentes types d'apprentissage automatique (ML) .

4.4.3.1 L'apprentissage supervisé

Les données utilisées pour l'apprentissage supervisé, notées $y1$, ...,yn, sont dites « complètes»car elle contiennent à la fois les valeurs $x1$, ..., xn prises par les P variables explicatives etleur appartenance aux k classes $z1$, ..., zn. Les données complètes sont donc l'ensemble descouples observations–labels, i.e.y1, ..., yn=($x1$, $z1$), ..., (xn, zn).
Exemple : en botaniquevous avez effectué des mesures (longueur de la tige, des pétales, ...) sur 100 plantes de 3espèces différentes. Chacune des mesures est étiquetée de l'espèce de la plante. Vous souhaitezconstruire un modèle qui saura automatique dire à quelle espèce une nouvelle plante appartientd'après le même type de mesures [8].

4.4.3.2 L'apprentissage non supervisé

Les données utilisées pour l'apprentissage non supervisé ne sont pas « complètes » car elles ne contiennent que les valeurs $x1$, ..., xn prises par les p variables explicatives [8].

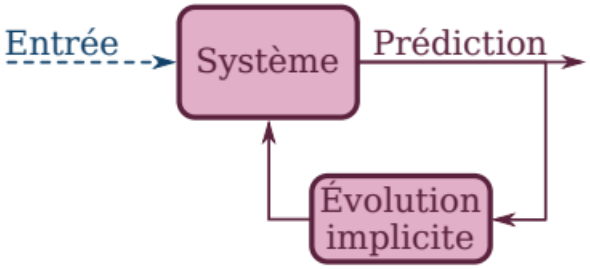


FIGURE 4.9 – Apprentissage non supervisé

On appelle apprentissage non supervisé car, contrairement à l'apprentissage supervisé ci-dessus, il n'y a pas de réponse correcte ni d'enseignant. Les algorithmes sont laissés à leurspropres mécanismes pour découvrir et présenter la structure intéressante des données. L'apprentissage non supervisé comprend deux catégories d'algorithmes : Algorithmes de regrou-pement et d'association[43] . L'apprentissage non supervisé consiste à ne disposer que dedonnées d'entrée (X) et pas de variables de sortie correspondantes. L'objectif de l'apprentissagenon supervisé est de modéliser la structure ou la distribution sous-jacente des données afind'en apprendre davantage sur les données. [43]
On va lister les avantages et les incovénients de chaque algorithme supervisé et non supervisé , comme illustrer dans la Tableau 4.2.

Apprentissage	*Type_Algorithme*	*Algorithme*	*Avantages*	*Incovénients*
Supervisé	*Classification*	***KNN***	1. facile à implémente. 2. efficace. 3. L'algorithme est polyvalent	1. Calculer chaque fois la similarité entre les k. [51] 2. grande capacité de stockage. 3. utilise de nombreuses données de références pour classifier les nouvelles entrées [50]
		SVM	1. Leur capacité à manipuler de grandes quantités de données 2. Le faible nombre d'hyper paramètres. 3. Elles sont bien fondées théoriquement. [52]	1. complexes pour la classification des corpus. 2. demande un temps énorme pendant les phases de test. [43]
		Arbre de décision	1. faciles à comprendre. 2. Ils permettent de sélectionner l'option la plus appropriée parmi plusieurs. 3. Il est facile de les associer à d'autres outils de prise de décision. [53]	1. instables. [53] 2. Certains concepts sont difficiles à exprimer à l'aide d'arbres de décision (comme XOR). [29]
		Naïve Bayes	1. La facilité et la simplicité de leur implémentation. 2. Leur rapidité. 3. Les méthodes Naïve Bayes donnent de bons résultats. [54]	1. faire le même travail de classification. [55] [56]
	Regression	***Linéaire***	1. Simplicité d'interprétation. 2. facilité de calcul [30]	Elle ne traite pas les valeurs manquantes de variables continues sensible aux valeurs hors norme de variables continues [57]
Non Supervisé	***CLUSTERING***	***K-means***	1. Simple 2. Flexible 3. Efficace 4. Complexité temporelle. [48]	1. Ensemble non optimal de clusters 2. Manque de cohérence 3. Limitation des calculs 4. Spécifiez les valeurs k [48]
	REDUCTION DES DI MENSIONS	***PCA***	1. Simplicité mathématique 2. Simplicité des résultats 3. Puissance 4. Flexibilité [58]	1. l'ACP n'a pas réellement 2.s'applique simplement sur des cas précis 3.Perte d'information par l'emploi fréquent de la 1ère composante principale uniquement. [58]

Tableau 4.2 – Comparaison entre les algorithmes du ML supervisé et non supervisé

4.4.3.3 L'apprentissage par renforcement

Il se produit lorsque vous présentez l'algorithme avec des exemples qui manquent d'étiquettes, comme dans l'apprentissage non supervisé. Cependant, vous pouvez accompagner un exemple de rétroaction positive ou négative selon la solution proposée par l'algorithme.[43]

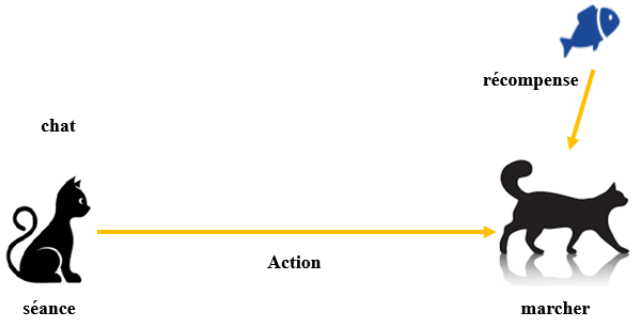


FIGURE 4.10 – L'apprentissage par renforcement

4.4.4 Apprentissage profond (Deep learning)

Le Deep Learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine Learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées Le deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou il détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit. [20]

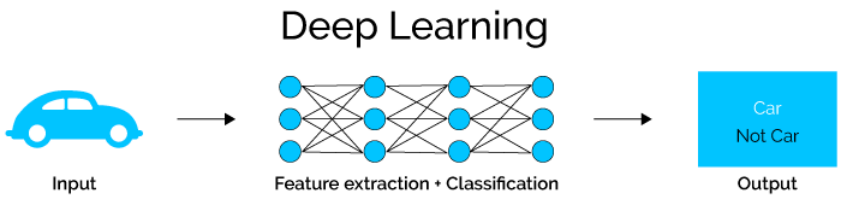


FIGURE 4.11 – Apprentissage profond(Deep learning)

4.5 Création du Dataset

Nous avons utilisés le format *CSV* pour la création du dataset,est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. La Figure 4.12 illustre la structure du dataset et les composantes (variables) :

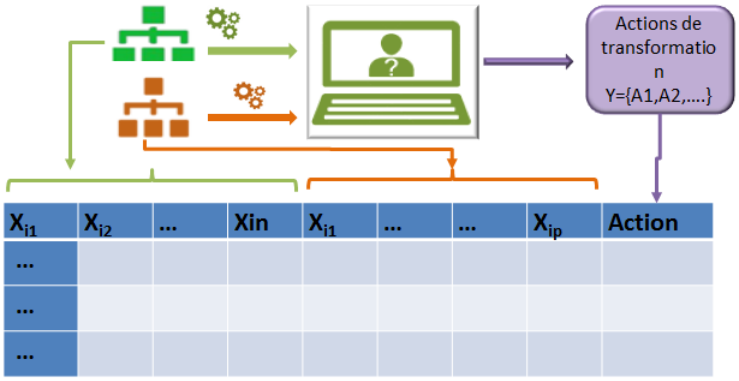


FIGURE 4.12 – Création du dataset

4.6 Réduction des dimensions du dataset

On peut souhaiter réduire de nombre de dimensions d'un jeu de données :

- pour le compresser : diminution le volume d'informations utiles à stocker et par la même occasion la durée d'exécution d'un algorithme d'apprentissage (car l'espace à explorer est plus petit)
- pour n'en conserver que les caractéristiques (features) discriminantes et éviter ainsi le surapprentissage (apprentissage du bruit dans les données).[55]

4.6.1 PCA - L'analyse de la composante principale

Algorithme : L'analyse en composante principale pour des données numériques en n dimensions est un algorithme non supervisé d'identification des dimensions de variance décroissante et de changement de base pour ne conserver que les k dimensions de plus grande variance[55]. Il consiste à :

- Optionnel : Normaliser les données (important si les données n'ont par exemple pas été mesurées aux mêmes échelles)
- Projeter les données initiales dans cet espace de dimension k

4.6.2 L'analyse discriminante linéaire ou quadratique (LDA/QDA)

Plutôt que de maximiser la variance sur des dimensions des données, on va ici chercher à maximiser la variance inter-classes par rapport à celle intra classe. Cette méthode transformera donc l'espace d'origine en un espace plus adapté que l'ACP dans un objectif de classification.[55]

4.7 Sélection et transformation des attributs

Cet opérateur peut être utilisé pour sélectionner un attribut (ou un sous-ensemble d'attributs) en définissant une expression régulière pour le nom de l'attribut et s'applique au sous-ensemble résultant des opérateurs internes. Notez que cet opérateur sera également utiliser des attributs spéciaux, ce qui rend nécessaire pour toutes les étapes de prétraitement à effectuer sur les attributs spéciaux (et

généralement pas effectuée sur les attributs spéciaux). Cet opérateur est très puissant et peut être utilisé pour créer de nouveaux systèmes de prétraitement combiné avec d'autres opérateurs de prétraitement. Cependant, il Ya deux restrictions importantes (parmi d'autres) : d'abord, parce que le résultat interne combinée avec le reste de l'ensemble des exemples d'entrée, le nombre d'exemples (points de données) n'est pas autorisé à modifier dans le prétraitement du sous-ensemble. Deuxièmement, les attributs de l'évolution du rôle ne seront pas libérés à l'extérieur parce que l'intérieur de tous les attributs spéciaux sera changé pour les opérateurs internes réguliers et les changements de rôles ne peuvent être livrés plus tard.[68]

4.8 Recherche des modèles

Dans notre solution , on appliqués cinq algorithme d'apprentissage automatique .

4.8.1 Arbre de décision

L'arbre de décision est une méthode d'apprentissage supervisé (ayant une variable cible prédéfinie) qui est principalement utilisé dans les problèmes de classification, et de prédiction. La construction d'un arbre de décision consiste à sélectionner pour chaque nœud de décision le test d'attribut approprié, et définir la classe relative à chaque feuille de l'arbre Figure 4.13. Cette méthode est utilisée notamment dans l'analyse des risques financiers, le diagnostic médical, et le diagnostic des pannes des équipements.[17]

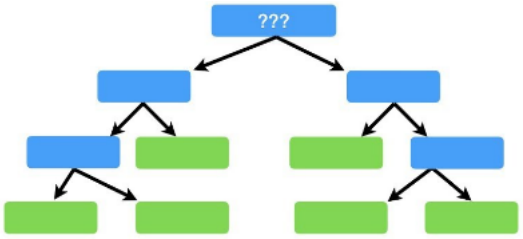


FIGURE 4.13 – Modèle d'arbre de décision

4.8.1.1 Avantages

Les arbres de décision ont un nombre de propriétés qui font d'eux un outil précieux, surtout quand il s'agit de faire l'analyse rapide d'un jeu de données ou d'élaborer un prototype de classifieur [9] :

- Modèle white box : le résultat est facile à conceptualiser, à visualiser et a interpréter.
- Ils nécessitent peu de préparation de données (e.g. normalisation, etc.).
- Le coût d'utilisation des arbres est logarithmique (classification d'une nouvelle donnée très rapide).
- Ils sont capables d'utiliser des données catégorielles et continues.
- Ils sont capables de gérer des problèmes multi-classes.
- Ils ont un bon comportement par rapport aux valeurs extrêmes .
- Ils gèrent bien les données manquantes.

4.8.1.2 Incovénients

— Parfois les arbres générés ne sont pas équilibrés (ce qui implique que le temps de parcours n'est plus logarithmique). Il est donc recommandé d'équilibrer la base de données avant la construction, pour éviter qu'une classe domine (en termes de nombre d'exemples d'apprentissage).
— Sur-apprentissage : parfois les arbres générés sont trop complexes et généralisent mal (solution : élagage, le contrôle de la profondeur de l'arbre et de la taille des feuilles).
— Ils sont instables : des changements légers dans les données produisent des arbres très différents. Les changements des nœuds proches de la racine affectent beaucoup l'arbre résultant. On dit que les arbres produisent des estimateurs de variance élevée.[9]

4.8.2 Foret aléatoire (Random forest)

L'algorithme des « forêts aléatoires » (ou Random Forest parfois aussi traduit par forêt d'arbres décisionnels) est un algorithme de classification qui réduit la variance des prévisions d'un arbre de décision seul, améliorant ainsi leurs performances. Pour cela, il combine de nombreux arbres de décisions dans une approche de type bagging. chaque arbre de la forêt aléatoire est entrainé sur un sous ensemble aléatoire de données selon le principe du bagging, avec un sous ensemble aléatoire de features (caractéristiques variables des données) selon le principe des « projections aléatoires ». Les prédictions sont ensuite moyennées lorsque les données sont quantitatives ou utilisés pour un vote pour des données qualitatives, dans le cas des arbres de classification. L'algorithme des forêts aléatoires est connu pour être un des classifieurs les plus efficaces « out-of-the-box » (c'est-à-dire nécessitant peu de prétraitement des données). La Figure 4.14 illustre la structure de l'algorithme Raandom Forest .[9]

4.8.2.1 Définition du Bagging

le bagging a pour but de réduire la variance de l'estimateur, en d'autres termes de corriger l'instabilité des arbres de décision (le fait que de petites modifications dans l'ensemble d'apprentissage entraînent des arbres très différents).[9]

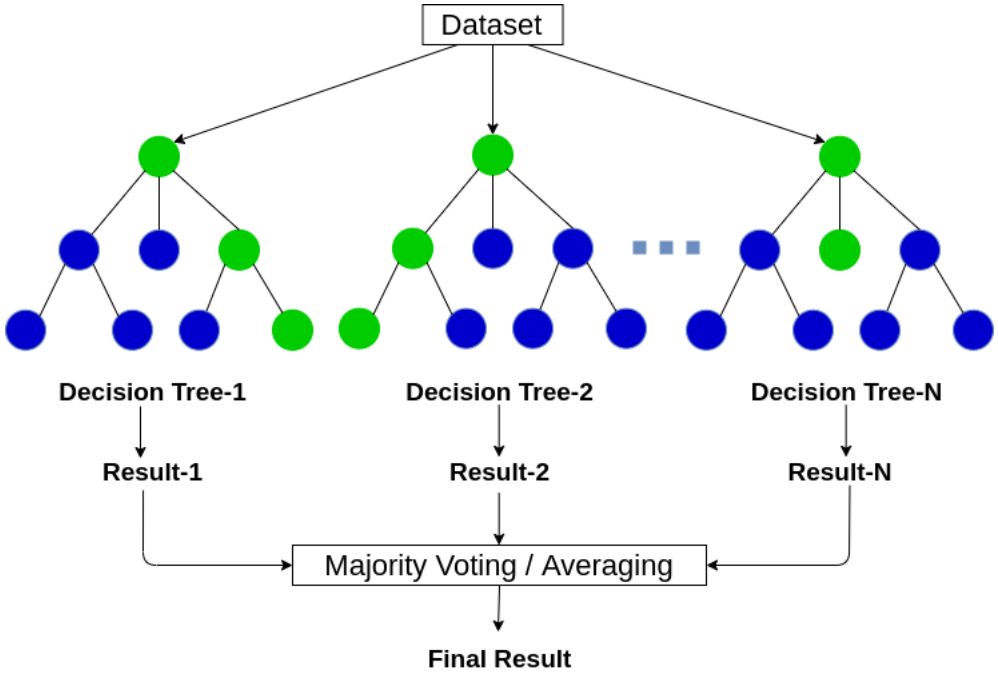


FIGURE 4.14 – l'algorithme de foret aléatoire

4.8.3 Algorithme de Random Forest

On tire au hasard dans la base d'apprentissage B échantillons avec remise zi,i=1,...,Bzi, i=1,...,B (chaque échantillon ayant n points). Pour chaque échantillon i on construit un arbre CART $Gi(x)$ selon un algorithme légèrement modifié : a chaque fois qu'un nœud doit être coupé (étape "split") on tire au hasard une partie des attributs (q parmi les p attributs) et on choisit le meilleur découpage dans ce sous-ensemble.[9]

Régression : agrégation par la moyenne

$$G(x) = \frac{1}{B}\sum_{i=1}^{B} G(x). \tag{4.1}$$

Classement : agrégation par vote G(x)=Votemajoritaire ($G1(x)$,...,$GB(x)$)

4.8.4 Avantages

- Reconnaissance trés rapide [48].
- Multi-classes par nature [48].
- Efficace sur inputs de grande dimension [48].
- Robustesse aux outliers [48].

4.8.5 Inconvénients

- Apprentissage souvent long

— Valeurs extrêmes souvent mal estimées dans cas de régression [48].

4.8.6 La régression logistique

La régression logistique est devenue un outil important dans la discipline de l'apprentissage automatique. Cette approche permet d'utiliser un algorithme dans l'application d'apprentissage automatique pour classer les données entrantes en fonction des données historiques. Plus il y a de données pertinentes en entrée, plus l'algorithme est en mesure de prédire des classifications au sein des jeux de données [58].

4.8.7 K plus proches voisins (KNN)

La méthode des k plus proches voisins (k-nearest neighbor, KNN) est une méthode supervisée. Elle a été utilisée dans l'estimation statistique et la reconnaissance des modèles comme une technique non paramétrique, cela signifie qu'elle ne fait aucune hypothèse sur la distribution des données.[67]
L'algorithme de KNN [67]
Soit : **X :** ensemble d'entrainement.
Y : étiquettes de classe de X.
x : individu inconnu.

Pour $i = 1$ **a** m **faire**

Calculer la distance d (X_i, x)

Fin pour

Construire l'ensemble I contenant des indices pour k plus petite distance d (X_i, x)

Retourner Étiquette majoritaire pour $\{Y_i, ou\ i\ \in I\}$

FIGURE 4.15 – L'algorithme des k plus proches voisins

4.8.7.1 Avantages :

La méthode des k plus proches voisins représente des avantages [67] tels que :

1. L'algorithme KNN est robuste envers des données bruitées
2. La méthode des k plus proches voisins est efficace si les données sont larges et incomplètes.
3. Cette méthode est l'une des plus simples de tous les algorithmes d'apprentissage automatique.

4.8.7.2 Inconvénients :

La méthode des k plus proches voisins comporte des inconvénients tels que :

1. Le temps de prédiction est très long puisqu'on doit calculer la distance de tous les exemples.

4.8.8 Les machines à support de vecteurs (SVM)

Introduit par Vapnik en 1990, les machines à vecteurs de support sont des techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression. Elles reposent sur deux notions principales : la notion de marge maximale et la notion de fonction noyau.[27]

4.8.8.1 Avantages :

Les SVM représentent plusieurs avantages, notamment ceux-ci :

1. Elles ont une base théorique solide[27].
2. Les SVM sont efficaces dans les espaces de grande dimension.[27]
3. Différentes fonctions noyau peuvent être spécifiées.[27]

4.8.8.2 Inconvénients :

Malgré leurs performances, les SVM représentent aussi des faiblesses, notamment celles-ci :

1. Elles utilisent des fonctions mathématiques complexes pour la classification.[27]
2. Les machines à support de vecteurs demandent un temps énorme durant les phases de test[27].

4.8.9 L'algorithme de Naïve Bayes

Un classifieur naïf de Bayes est un classifieur probabiliste basé sur l'application du théorème de Bayes avec l'hypothèse naïve, c'est-à-dire que les variables explicatives (Xi) sont supposées indépendantes conditionnellement à la variable cible (C). Malgré cette hypothèse forte, ce classifieur s'est avéré très efficace sur de nombreuses applications réelles et est souvent utilisé sur les flux de données pour la classification supervisée. Le classifieur naïf de Bayes nécessite simplement en entrée l'estimation des probabilités conditionnelles par variable P (Xi|C) et les probabilités a priori P (C). Pour une utilisation sur les flux de données, cette estimation peut être fournie à l'aide d'un « résumé supervisé en-ligne de quantiles ». L'état de l'art montre que le classifieur naïf de Bayes peut être amélioré en utilisant une méthode de sélection ou de pondération des variables explicatives. La plupart de ces méthodes ne peuvent fonctionner que hors-ligne car elles nécessitent de stocker toutes les données en mémoire et/ou de lire plus d'une fois chaque exemple. Par conséquent, elles ne peuvent être utilisées sur les flux de données. Cet article présente une nouvelle méthode basée sur un modèle graphique qui calcule les poids des variables d'entrée en utilisant une estimation stochastique. La méthode est incrémentale et produit un classifieur Naïf de Bayes Pondéré pour flux de données. Cette méthode est comparée au classique classifieur naïf de Bayes sur les données utilisées lors du challenge « Large Scale Learning » [61].

4.8.9.1 Avantages :

1. La facilité et la simplicité de leur implémentation.
2. Leur rapidité.
3. Les méthodes Naïve Bayes donnent de bons résultats.

		Classe réelle	
		-	+
Classe Prédite	-	vrai négatif	faux négatif
	+	faux positif	vrai positif

Tableau 4.3 – La matrice de confusion

4.8.9.2 Inconvénients :

À cause de l'hypothèse d'indépendance des mots dans ce modèle, on le qualifié souvent de naïf ou de simple. En général, ce type d'algorithmes permet de faire le même travail de classification que les autres algorithmes qui existent déjà, mais ces performances sont limitées quand il s'agit d'une grande quantité de lexiques à traiter. En effet, si le nombre de lexiques augmente, alors le nombre des dépendances entre l' ensemble des mots augmentent, et donc, la vérification de l'hypothèse de Naïve Bayes diminue [26].

4.9 Mesures d'évaluation (performances) du modèle

Evaluation de la prédictivité des hypothèses produites : ensemble d'exemples séparés en ensemble d'apprentissage A et ensemble test T.

- Apprentissage d'une hypothèse H avec A
- Evaluation de la prédiction de H sur T

4.9.1 Matrice de confusion ou de contingence

Une matrice de confusion ou tableau de contingence est un outil permettant de mésurer les performances d'un modèle d'apprentissage automatique, ou c'est un résumé des résultats de prédictions sur un problème de classification. Le tableau ci-dessus illustre une matrice de confusion en vue globale. Les prévisions sont alignées sur la droite et les données réelles en haut. Les prévisions ou les résultats sont présentés dans les lignes, juste en dessous des données réelles. Les résultats peuvent comprendre l'indication correcte d'une valeur positive comme étant un vrai positif ou d'une valeur négative comme étant un vrai négatif, ou encore d'une valeur positive incorrecte comme un faux positif ou d'une valeur négative incorrecte comme un faux négatif. [40]

4.10 Les variables de la matrice de confusion

Vrai positif : est un résultat où le modèle prédit correctement la classe positive. [40]
Vrai négatif : est un résultat où le modèle prédit correctement la classe négative.
Faux positif : est un résultat où le modèle prédit incorrectement la classe positive.
Faux négatif : est un résultat où le modèle prédit incorrectement la classe négative.

qui illustre les quatre résultats possibles : Les vrais négatifs ainsi que les vrais positifs correspondent à un fonctionnement correct de la technique de machine Learning ce qui signifie que la technique de data machine Learning a prédit avec succès respectivement le comportement normal et les attaques. Les faux négatifs sont des attaques incorrectement prédites comme des comportements normaux. Les métriques traditionnelles de classification comprennent le taux d'exactitude, le taux de fausse alerte et le taux d'erreur de la classification, elles sont définies comme suit :

Le taux d'exactitude (TE) : montre à quel point le système est exact, c'est le nombre de type bien classé sur le nombre de type de tout le corpus .

$$\boldsymbol{Exactitude} = \frac{VP + VN}{VP + VN + FP + FN}$$

4.11 Conclusion

Ce chapitre présente les concepts de base de l'apprentissage automatique. Premièrement, sa définition et ses types ont été abordés pour donner une image claire, et deuxièmement, les algorithmes ont été expliqués en détail, pca et donnez des exemples. Après cela, nous discuterons d'une application utilisant la technologie ML.

Chapitre 4 : Implémentation

Chapitre

4 Implémentation

5.1 Introduction

L'objectif de ce chapitre est de présenter les étapes de l'implémentation de l'approche proposée dans le cadre de transformation de modèles par l'exemples . Nous commençons tout d'abord par la présentation des outils technologies utilisés , et de l'environnement de développement que nous avons utilisés et on termine par les étapes de la réalisation et l'évaluation du modèle. Ce chapitre est composé de deux parties, l'implémentation du système et les résultats expérimentaux.

5.2 Environnement de développement et Les outils techniques

5.2.1 Définition du langage Python en informatique :

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages [44]. On a utilisé la version Python 3.4.8 mais la dernière version c'est Python 3.9.0.

5.2.2 Anaconda

Anaconda est une distribution open source pour python et R. Il est utilisé pour la science des données, l'apprentissage automatique, l'apprentissage en profondeur, etc. Avec la disponibilité de plus de 300 bibliothèques pour la science des données, il devient assez optimal pour tout programmeur de travailler sur anaconda pour la science des données [3].

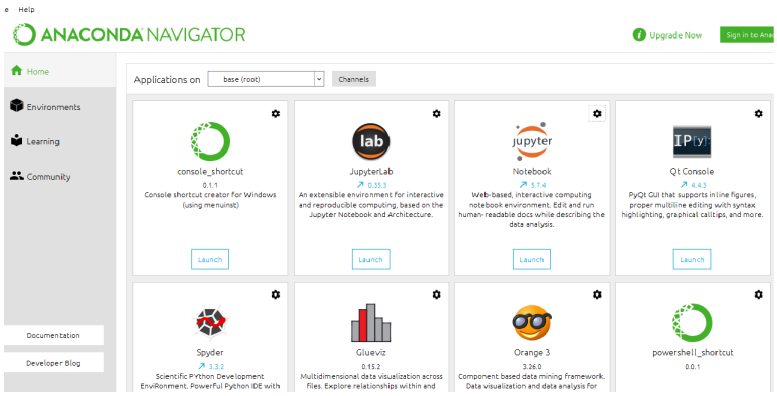


FIGURE 5.1 – L'environement Anaconda

5.2.3 Jupyter :

se présente comme un outil extrêmement simple à mettre en oeuvre qui vous permettra de transformer vos Jupyter Notebooks en applications web ou en Dashboard quasiment automatiquement [71].

5.3 Les bibliothèques utilisées :

On a utilisé plusieurs bibiothèques , on peut citer :

5.3.1 Numpy :

Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées (diffusion), on peut aussi l'intégrer le code C / C ++ et Fortran [49].

5.3.2 Matplotlib :

Est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy . Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter , wxPython , Qt ou GTK + [31].

5.3.3 Pandas :

Pandas pandas est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python . pandas est un projet sponsorisé par NumFOCUS . Cela contribuera

au succès du développement de pandas en tant que projet open source de classe mondiale et permettra de faire un don au projet [10].

5.3.4 Scikit-learn :

Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria3 et Télécom Paris. 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 [30].

Testez le modèle sur l'ensemble de test et évaluez la performance de notre modèle.

Avantages de la séparation train / test :

- Le modèle peut être formé et testé sur des données différentes de celles utilisées pour la formation.
- Les valeurs de réponse sont connues pour l'ensemble de données de test, donc les prédictions peuvent être évaluées.
- La précision des tests est une meilleure estimation que la précision de la formation des performances hors échantillon.

5.4 Apprentissage et Paramétrage des Modèles :

5.4.1 Préparation du Dataset

Les spécificités du processus de préparation des données varient selon le secteur d'activité, l'entreprise et les besoins, mais le cadre de travail/framework demeure essentiellement le même.

1. Collecte de données Le processus de préparation des données commence par la recherche des données les plus utiles. Ces données peuvent provenir d'un catalogue existant ou être ajoutées en mode ad hoc.
2. Découvrir et évaluer les données Lorsque les données ont été collectées, il est important de découvrir les différents datasets. Cette étape permet de mieux connaître les données et de déterminer le traitement à leur appliquer avant qu'elles deviennent exploitables dans un contexte particulier.
3. Nettoyer et valider les données En général, le nettoyage des données est l'étape la plus longue du processus de préparation des données, mais cette opération est cruciale pour éliminer les données erronées et combler d'éventuelles lacunes. Lors du nettoyage, les tâches importantes sont notamment les suivantes :
   - Supprimer les données superflues et les valeurs aberrantes.
   - Ajouter les valeurs manquantes .
   - Adapter les données à une structure standard , les données numériques de l'ensemble de données sont de différentes plages, ce qui pose certains défis au classificateur pendant la

formation pour compenser ces différences. Ainsi, il est important de normaliser les valeurs de chaque attribut, de sorte que la valeur minimale de chaque attribut soit nulle, tandis que le maximum est un. Cela fournit des valeurs plus homogènes au classificateur tout en maintenant la relativité entre les valeurs de chaque attribut. avec la technique StandardScaler () on a normalisé les entités ,l'idée derrière StandardScaler est qu'il transformera les données de telle sorte que leur distribution aura une valeur moyenne de 0 et un écart-type de 1.

- Masquer les données privées ou sensibles Lorsque les données ont été nettoyées, elles doivent être validées, à savoir déterminer si des erreurs se sont produites dans le processus de préparation des données jusqu'à ce point (il peut arriver qu'une erreur apparaisse pendant cette étape, et il est alors nécessaire de la corriger avant de poursuivre).

4. Transformer et enrichir les données « Transformer les données » consiste à mettre à jour les entrées de format ou de valeur de manière à obtenir un résultat clairement défini ou à rendre les données plus faciles à comprendre par un plus grand nombre d'employés. « Enrichir les données » consiste à ajouter des données et à les relier à des données apparentées de manière à dégager des connaissances approfondies.
5. Stocker les données Lorsque la préparation des données est terminée, celles-ci peuvent être stockées ou routées vers une application tierce – par exemple, un outil de Business Intelligence – avant leur traitement et analyse.

5.4.2 Identification des variables de prédiction

Aprés la lecture et le prétraitement de notre dataset , on appliquer la fonction pd.Factorize qui permet d'encoder l'objet en tant que type énumeré ou variable catégorielle . la Liste 5.1 illustre un pseudo code qui permet d'identi$_e rvariabledeprdictionetencoderlesvariablesdetypestringversentier$:

```
# Encoding The data
data['class1_MS'],_ = pd.factorize(data['class1_MS'])
data['class2_MS'],_ = pd.factorize(data['class2_MS'])
data['ordreRelation_MS'],_ = pd.factorize(data['ordreRelation_MS'])
data['nbAtt_touch_MS'],_ = pd.factorize(data['nbAtt_touch_MS'])
```

Listing 5.1 – Identifier la variable cible du dataset

```
[[ 0  0  0 ...  0  0  0]
 [ 1  1  0 ...  1  1  1]
 [ 2  2  0 ...  0  0  0]
 ...
 [-1 -1 -1 ... -1 -1 -1]
 [-1 -1 -1 ... -1 -1 -1]
 [-1 -1 -1 ... -1 -1 -1]] [ 0  1  0  0  1  0  1  0  1  0  0  0  1  1  1  1  1  0  1  0  0 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
```

Figure 5.2 – Conversation des variables de prédiction en entiers

La prochaine étape consiste à diviser les données ensembles d'apprentissage et de test, afin d'atteindre notre objectif, 70 des données de chacun des mélanges sont utilisés pour entrainer le classificateur

,puis 30 des données ont été utilisés pour valider le model.

```
#spiting the data test train split
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(X,y , test_size = 0.3,
    random_state = 40)
```

Listing 5.2 – Diviser les données du dataset

5.4.2.1 Avantages de la séparation train / test :

- Le modèle peut être formé et testé sur des données différentes de celles utilisées pour la formation.
- Les valeurs de réponse sont connues pour l'ensemble de données de test, donc les prédictions peuvent être évaluées.
- La précision des tests est une meilleure estimation que la précision de la formation des performances hors échantillon.

5.5 Réduction des dimensions du dataset

5.5.1 Application de PCA :

Pour exécuter PCA on a utilisé la bibliothèque *Scikit-Learn* de Python. La classe PCA est utilisée à cet effet. PCA dépend uniquement de l'ensemble de fonctionnalités et non des données d'étiquette. La réalisation de PCA à l'aide de Scikit-Learn est un processus en deux étapes :

- Initialisez la classe PCA en passant le nombre de composants au constructeur.
- Appelez l'ajustement, puis transformez les méthodes en transmettant l'ensemble de fonctionnalités à ces méthodes. La méthode de transformation renvoie le nombre spécifié de composants principaux, voici le code suivant :

```

# Standerdizing
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)

# trying PCA
from sklearn.decomposition import PCA
pca = PCA(2)
new_X = pca.fit_transform(X)
ratio = pca.explained_variance_ratio_

# Plotting The data

plt.scatter(new_X[:,0] , new_X[:,1] , c = y )
plt.legend()
plt.show()
```

Listing 5.3 – Code python de l'algorithme PCA

5.5.2 Réduction les dimensions de dataset

Ensuite , on a créer un modèle PCA pour faire choisi le nombre optimale du dataset. Pour exécuter PCA on a utilisé la bibliothèque *Scikit-Learn* de Python. La classe PCA est utilisée à cet effet. PCA dépend uniquement de l'ensemble de fonctionnalités et non des données d'étiquette. La réalisation de PCA à l'aide de Scikit-Learn est un processus en deux étapes :Initialisez la classe PCA en passant le nombre de composants au constructeur. Appelez l'ajustement, puis transformez les méthodes en transmettant l'ensemble de fonctionnalités à ces méthodes. La méthode de transformation renvoie le nombre spécifié de composants principaux, voici le code Liste 5.3 suivant : la Figure 5.3 affiche l'exécution du l'algorithme réduction les dimensions de dataset (PCA).

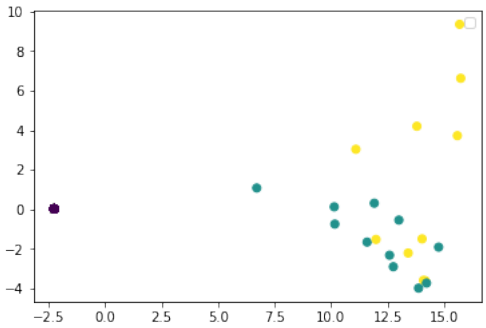


Figure 5.3 – Le modèle PCA

– 5.5.3 Algorithme LDA/QDA

Pseudo-code de l'algorithme LDA/QDA Le code ci-dessus Liste 5.4 montre l'algorithme L'analyse discriminante linéaire ou quadratique

```
def graph_acp2(X_PC2, y):
    plt.figure(figsize=(15,4))
    plt.subplot(1, 3, 1)
    plt.scatter(X_PC2[:, 0], np.ones(X_PC2.shape[0]), c=y)
    plt.subplot(1, 3, 3)
    plt.title("Dimension de 2nde variance :")
    plt.scatter(X_PC2[:, 1], np.ones(X_PC2.shape[0]), c=y)
    plt.show()
graph_acp2(new_X, y)
#trying LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X_iris_LDA = lda.fit(new_X, y).transform(new_X)
print("Rappel des composantes identifi es par le PCA :")
graph_acp2(new_X, y)
print("Composantes identifi es par le LDA (on remarque une meilleure
    s paration des classes sur la 1 re composante) :")
```

```
graph_acp2(X_iris_LDA , y)
```

Listing 5.4 – Code python de l'algorithme LDA

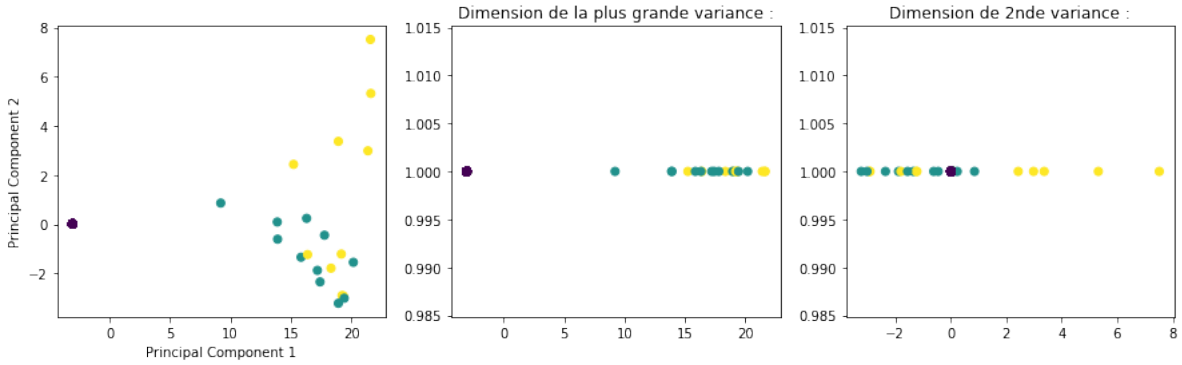


FIGURE 5.4 – L'exécution de l'algorithme LDA

5.6 Apprentissage du modèle

Dans la recherche de modèles , on utiliser des algorithmes d'apprentissage supervisé .

5.6.1 Algorithme de la régression logistique

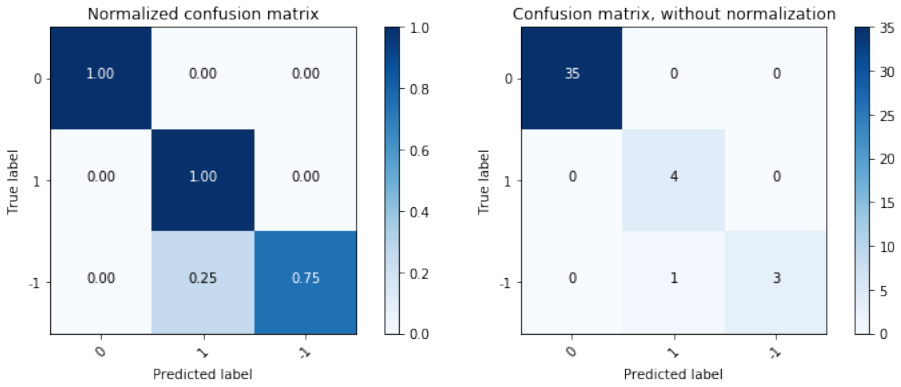
Pseudo-code de l'algorithme régression logistique On a créer un modèle de régression logistique , pour classer dataset , et on a fait la prédiction sur les données de test , et on a la résultat afficher sur la matrice de confusion. la Liste 5.5 illustre pseudo-code de l'algorithme régression logistic :

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train , y_train)
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test , y_pred)
print('Confusion Matrix is :', cm)
```

Listing 5.5 – Identifier la variable cible du dataset

Le Tableau 5.1 explique la performance du modèle . Comme on peut l'observer , une matrice de confusion a en colonnes et en lignes les memes intitulés : beginitemize

1. En ligne : on lit les labels des individus .
2. En colonne :les labels prédits par le modèle Ainsi , on peut conclure par exemple que dans 35 situation Où il appartient à la classe 0 , le modèle a bien prédit les classes 1 et -1 déduite par le modèle contient 8 observatiions bien classés . l'exécution de cet algorithme illustré dans la figure suivante :



(Avec Normalization) (Sans normalization)

Tableau 5.1 – Matrice de confusion de l'algorithme Logistic Regression

5.6.2 Algorithme de KNN(K plus proches voisins)

Pseudo-code de l'algorithme Naïve Bayes

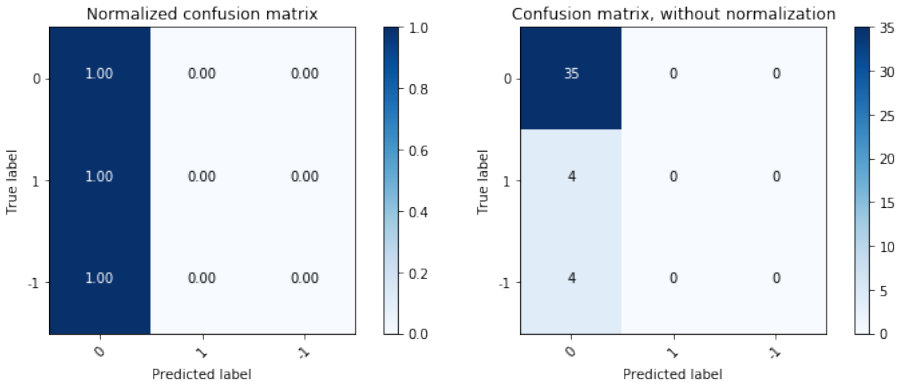
le pseudo-code suivant Liste 5.6 montre l'algorithme de KNN en python : On a créer un classicateur KNN , pour classer dataset , et on a fait la prédiction sur les données de test , et on a la résultat afficher sur la matrice de confusion .

```
#Algorithme KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 56, metric = 'minkowski',p =2)
knn.fit(x_train , y_train)
```

Listing 5.6 – Algorithme de KNN

Le Tableau 5.2 explique la performance du modèle . Comme on peut l'observer , une matrice de confusion a en colonnes et en lignes les memes intitulés :

- En ligne : on lit les labels des individus .
- En colonne :les labels prédits par le modèle Ainsi , on peut conclure par exemple que dans 35 situation Où il appartient à la classe 0 , le modèle a bien prédit les classes 1 et -1 déduite par le modèle contient 8 observatiions mal classés . L'exécution de l'algorithme de KNN illustré dans la figure suivante :



(Avec Normalization) (Sans normalization)

Tableau 5.2 – Matrice de confusion de l'algorithme KNN

5.6.3 Algorithme Arbre de décision

Pseudo-code de l'algorithme Arbre de décision

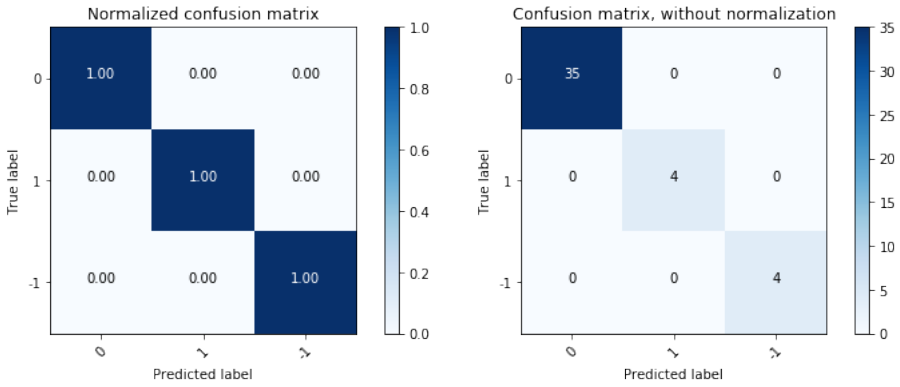
Le pseudo-code ci-decussListe 5.7 montre le modèle d'arbre de décision , pour classer dataset , et on a fait la prédiction sur les données de test , et on a la résultat afficher sur la matrice de confusion .

```
#Algorithme Decision Tree
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion ='entropy')
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Decision Trees: ' , r2_score(y_test , y_pred))
print('The accuracy For Decision Trees is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test, y_pred)
```

Listing 5.7 – le code d'algorithme arbre de décision

Le Tableau 5.3 explique la performance du modèle . Comme on peut l'observer , une matrice de confusion a en colonnes et en lignes les memes intitulés :

- En ligne : on lit les labels des individus .
- En colonne :les labels prédits par le modèle Ainsi , on peut conclure par eemple que dans 35 situation Où il appartient à la classe 0 , le modèle a bien prédit les classes 1 et -1 déduite par le modèle contient 8 observatiions bien classés . la figure suivante illustre l'affichage de l'exécution du code précédent



(Avec Normalization) (Sans normalization)

Tableau 5.3 – Matrice de confusion de l'algorithme Decision Trees

5.6.4 Algorithme de Naïve Bayes

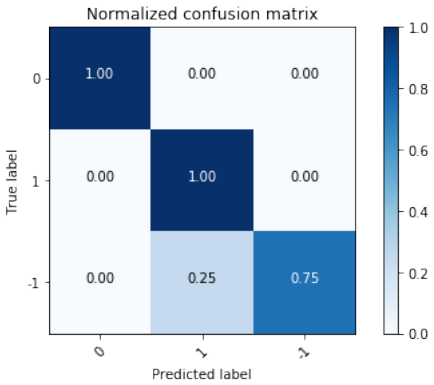
Pseudo-code de l'algorithme Naïve Bayes le code suivant Liste 5.8 montre l'algorithme de Naïve Bayes en python : On a créer un modèle de Naïve Bayes , pour classer dataset , et on a fait la prédiction sur les données de test , et on a la résultat afficher sur la matrice de confusion

```
#Algorithme Na ve Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Naive Bayes: ', r2_score(y_test , y_pred))
print('The accuracy for NB is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
```

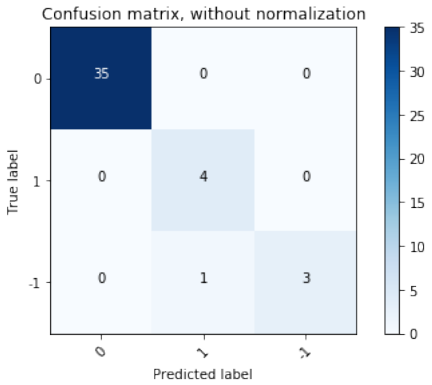
Listing 5.8 – Algorithme de Naïve Bayes

Le Tableau 5.4 explique la performance du modèle . Comme on peut l'observer , une matrice de confusion a en colonnes et en lignes les memes intitulés :

- En ligne : on lit les labels des individus .
- En colonne :les labels prédits par le modèle Ainsi , on peut conclure par eemple que dans 35 situation Où il appartient à la classe 0 , le modèle a bien prédit les classes 1 et -1 déduite par le modèle contient 8 observatiions bien classés . La figure sivante illustre la matrice de confusion de l'algorithme Naïve Bayes :



(Avec Normalization)



(Sans normalization)

Tableau 5.4 – Matrice de confusion de l'algorithme Naïve Bayes

5.6.5 Algorithme de Random forest

Pseudo-code de l'algorithme de construction d'un arbre de décision Pseudo-code de l'algorithme de construction d'un arbre de décision La Liste 5.9illustre le pseudo-code de l'algorithme Random forest . On a créer un modèle de Random Forest , pour classer dataset , et on a fait la prédiction sur les données de test , et on a la résultat afficher sur la matrice de confusion.

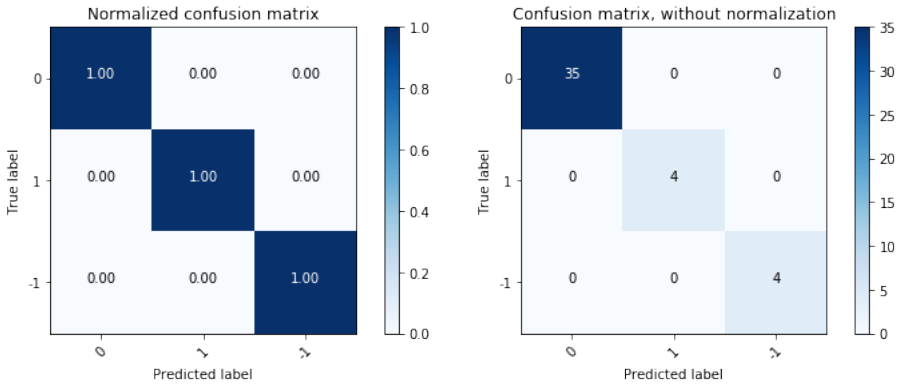
La Liste 5.9illustre le pseudo-code de l'algorithme Rndom forest.

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier( n_estimators = 50 , criterion ='entropy')
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
print('The score For Random Forest: ' , r2_score(y_test , y_pred))
print('The accuracy For Random Forest is ', (y_pred == y_test).mean())
cm = confusion_matrix(y_test,y_pred)
print('Confusion Matrix is :', cm)
```

Listing 5.9 – Algorithme de Random forest

Le tableau 5.5 explique la performance du modèle . Comme on peut l'observer , une matrice de confusion a en colonnes et en lignes les memes intitulés :

- En ligne : on lit les labels des individus .
- En colonne :les labels prédits par le modèle Ainsi , on peut conclure par eemple que dans 35 situation Où il appartient à la classe 0 , le modèle a bien prédit les classes 1 et -1 déduite par le modèle contient 8 observatiions bien classés . la figure suivante illustre l'exécution du code de l'algorithme Random forest.



(Avec Normalization) (Sans normalization)

Tableau 5.5 – Matrice de confusion de l'algorithme Random Forest

5.7 Résultats expérimentaux

5.7.1 Rapport de classification (Classification report) :

Le rapport de classification est utilisé pour mesurer la qualité des prévisions de notre classifieur. Combien de prédictions sont vraies et combien sont fausses. Plus précisément, les vrais positifs, les faux positifs, les vrais négatifs et les faux négatifs sont utilisés pour prédire les mesures d'un rapport de classification, comme indiqué ci-dessous Figure 5.5 :

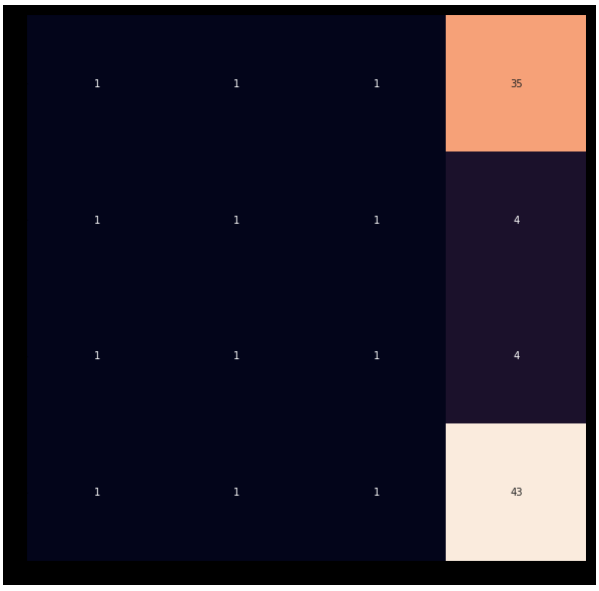


FIGURE 5.5 – rapport de classification.

Le rapport montre la précision des mesures de classification, le rappel et le score f1 principaux par classe. Les métriques sont calculées en utilisant des vrais et des faux positifs, des vrais et des faux

négatifs.
On a la précision, le rappel, le score f1 et le support pour chaque classe que nous avons essayé de trouver.

- Le rappel signifie "combien de cette classe vous trouvez sur le nombre entier d'éléments de cette classe".
- La précision sera "combien sont correctement classés dans cette classe"
- Le score f1 est la moyenne harmonique entre précision et rappel
- Le support est le nombre d'occurrences de la classe donnée dans votre ensemble de données.

5.7.2 Tableau comparatif de l'implémentation de certains modèles :

Algorithme	Précision	Score
Decision Trees	0.7	0.8
Random Forest	0.6	0.5
KNN	0.8	0.4
Naive Bayes	0.6	0.6
Regression Logistic	0.8	0.7

Tableau 5.6 – Tableau comparatif de l'implémentation de certains modèles

Nous constatons que le résultat obtenu avec le modèle : l'algorithme decision trees est le meilleur.

5.8 Conclusion

Dans ce chapitre nous avons, en premier lieu, présenté les différents outils et langages que nous avons utilisés pour implémenter notre model. Ainsi une approche basée sur les différents algorithmes d'apprentissage suervisé (Decision Trees, Random Forest,..,etc). On a aussi citer les différentes étape de l'approche proposée dans le cadre de transformation de modèles , et les résultats obtenues sont créer dans un tableau.

Conclusion et Perspectives

Conclusion

"To accomplish great things, we must not only act, but also dream ;
not only plan, but also believe".
Anatole France (1844-1924)

Conclusion et Perspectives

L'intérêt majeur de ce mémoire est d'assisté le concepteur/developpeur pour effectuer la transformation des modèles. Nous avons effectué une recherche d'informations pour se familiariser avec le sujet, ce qui nous a permis de produire une synthèse bibliographique sur le cœur de ce travail. Nous avons présenté une nouvelle approche basée sur l'apprentissage machine afin de faciliter et d'aider le concepteur dans les tâches de transformation du modèle. Notre solution permet d'effectuer plusieurs actions sur le modèle source : fusion, décomposition, éclatement, etc. Le travail présenté de ce rapport s'articule dans cinq points essentiels :

- La formalisation du problème de la transformation du modèles à base d'exemple.
- l'identification des dimensions de la base d'exemple (Dataset).
- Collect de la connaissance des métiers d'ingénieurs dans la transformation des modèles sous formes des traces de transformations.
- Préparation de notre base d'exemples (transformation, encodage de données ,etc.)
- Implémentation des algorithmes d'apprentissage machine.

Notre travail ouvre plusieurs perspectives, nous pouvons citer :

- Appliquer notre algorithme sur une grande base d'exemple.
- Implémenter d'autre Algorithme d'apprentissage automatique.
- Développer un outil support à la base de cette approche proposée.
- Appliquer l'approche proposée sur d'autre formalise comme DEVES et le formalisme B.

.

Bibliographie

[1] G. D. 1s1. Définition de l'Intelligence Artificielle. Centre de recherche en informatique de Montréal. http://tpe-intelligence-artificielle-2013.e-monsite.com/pages/definition-de-l-intelligence-artificielle.html. Page consultée le 16 janvier 2020. 2013 .

[2] *About Systems Engineering*. AFIS – 32 Rue Jean Rostand. https://www.incose.org/about-systems-engineering. Page consultée le 16 janvier 2020. 2019.

[3] *Anaconda Individual Edition* https://www.anaconda.com/products/individual.

[4] I. Baki. « Intelligence Artificielle, Machine Learning, Deep Learning : une histoire de poupées russes ». In : (2015).

[5] I. Baki et al. « Learning implicit and explicit control in model transformations by example ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2014, p. 636–652 .

[6] Z. Balogh et D. Varró. « Model transformation by example using inductive logic programming ». In : *Software & Systems Modeling* 8.3 (2009), p. 347–364.

[7] G. Bernot, M. C. Gaudel et B. Marre. « Software testing based on formal specifications : a theory and a tool ». In : *Software Engineering Journal* 6.6 (1991), p. 387–405.

[8] C. Bouveyron. « Modélisation et classification des données de grande dimension : application à l'analyse d'images. » Thèse de doct. Université Joseph-Fourier-Grenoble I, 2006.

[9] Kontonatsios, G., Korkontzelos, Y., Tsujii, J. I., & Ananiadou, S. (2014, April). Using a random forest classifier to compile bilingual dictionaries of technical terms from comparable corpora. In 14th Conference of the European Chapter of the Association for Computational Linguistics.

[10] B. Cayla. *Python Pandas – Tuto* https://pandas.pydata.org/ *(Partie N°1).*

[11] K. Czarnecki et S. Helsen. « OOPSLA ' 03 Workshop on Generative Techniques in the Context of Model-Driven Architecture 1 Classification of Model Transformation Approaches ». In : 2003.

[12] K. Czarnecki et S. Helsen. « Feature-based survey of model transformation approaches ». In : *IBM systems journal* 45.3 (2006), p. 621–645.

[13] J. Davis. « GME : the generic modeling environment ». In : jan. 2003, p. 82–83. doi : 10.1145/949344.949360 .

[14] X. Dolques et al. « Learning transformation rules from transformation examples : An approach based on relational concept analysis ». In : *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*. IEEE. 2010, p. 27–32.

[15] M. Faunes, H. Sahraoui et M. Boukadoum. « Generating model transformation rules from examples using an evolutionary algorithm ». In : *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE. 2012, p. 250–253 .

[16] M. Faunes, H. A. Sahraoui et M. Boukadoum. « Genetic-Programming Approach to Learn Model Transformation Rules from Examples ». In : *Proceedings of the 6th International Conference on Theory and Practice of Model Transformations, ICMT 2013*. Sous la dir. de K. Duddy et G. Kappel. T. 7909. Lecture Notes in Computer Science. Budapest, Hungary : Springer, 2013, p. 17–32. doi : doi:10.1007/978-3-642-38883-5_2

[17] M. Feriel. « Conception et réalisation dun système de détection des fraudes dans le secteur des assurances automobiles. » Thèse. Ecole national supérieure d'informatique, 2015.

[18] « Rete : A fast algorithm for the many pattern/many object pattern match problem ». In : *Artificial Intelligence*. Sous la dir. de C. Forgy. T. 19. Integrated series in information systems. Springer, 1982. Chap. 1, p. 17–37.

[19] P. R. Franck Vallée. *UML en action : De l'analyse des besoins à la conception en Java*. 1ère édition. Pearson Education, 2000.

[20] Futura. *-automatique/*.

[21] I. García-Magariño, J. J. Gómez-Sanz et R. Fuentes-Fernández. « Model transformation by-example : An algorithm for generating many-to-many transformation rules in several model transformation languages ». In : *International Conference on Theory and Practice of Model Transformations*. Springer. 2009, p. 52–66

[22] J. Greenfield et K. Short. « Software factories : Assembling applications with patterns, frameworks, models and tools ». In : (2004).

[23] O. S. S. Group. *Object Management Group : OMG Unified Modeling Language specification*. Object Management Group :. https://www.omg.org/mda/. Page consultée le 6 février 2020. 2003.

[24] D. C. Halbert. « Programming by Example ». Thèse. California, Berkeley : University of California, Berkeley, 1984 .

[25] D. Harel et A. Pnueli. « On the development of reactive systems ». In : *Logics and models of concurrent systems*. 1985, p. 477–498.

[26] H. Hilali. « Application de la classification textuelle pour l'extraction des règles d'association maximales ». Thèse de doct. Université du Québec à Trois-Rivières, 2009.

[27] H. Hilali. « Application de la classification textuelle pour l'extraction des règles d'association maximales ». Thèse. Université du Québec à Trois-Rivières : Mathématiques et informatique appliquées, 2011.

[28] J. H. Holland. *Emergence : From Chaos to Order*. UK : Oxford University Press, 2000 .

[29] D. G. IONOS. *{f}onctionne le machine learnings*.

[30] E. F.M. D. John Hunter Darren Dale. *Lancement de l'initiative scikit-learn, bibliothèque logicielle de référence en machine learning* (cf. p. 57).

[31] E. F.M. D. John Hunter Darren Dale. *Matplotlib*.

[32] F. Jouault et al. « ATL : a QVT-like transformation language ». In : t. 2006. Jan. 2006, p. 719–720. doi : 10.1145/1176617.1176691.

[33] F. Jouault et al. « ATL : a QVT-like transformation language ». In : *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. 2006, p. 719–720 (cf. p. 4).

[34] G. Kappel et al. *Model transformation by-example : a survey of the first wave, Conceptual Modelling and Its Theoretical Foundations : essays dedicated to Bernhard Thalheim on the occasion of his 60th birthday*. 2012

[35] S. Kent. « Model Driven ». In : 2002.

[36] E. HILLALI KERKOUCHE. « Modélisation Multi-Paradigme : Une Approche Basée sur la Transformation de Graphes ». Thèse. UNIVERSITE DEMENTOURI CONSTANTINE . Département d'informatique, 2011

[37] M. KESSENTINI, H. SAHRAOUI et M. BOUKADOUM. « Model transformation as an optimization problem ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2008, p. 159–173

[38] M. KESSENTINI et al. « Generating transformation rules from examples for behavioral models ». In : *Proceedings of the Second International Workshop on Behaviour Modelling : Foundation and Applications*. 2010, p. 1–7.

[39] M. KESSENTINI et al. « Search-based model transformation by example ». In : *Software & Systems Modeling* 11.2 (2012), p. 209–226.

[40] B. L. *Confusion Matrix : l'outil de mesure de performances du Machine Learning*.

[41] C. LARMAN. *Transformation : The Missing Link of MDA*. Anna Gerber,Michael Lawley ,Kerry Raymond,Jim Steel ,Andrew Wood : International Conference on Graph Transformation, 2002.

[42] J.-L. LAURIERE. *Intelligence artificielle : résolution de problèmes par l'homme et la machine*. Eyrolles Paris, 1987.

[43] LE-DATASCIETIST.FR. *Supervisé Vs. Non Supervisé*.

[44] *Le tutoriel Python* .

[45] *L'INGÉNIERIE SYSTÈME*. AFIS – 32 Rue Jean Rostand. https://www.afis.fr/lassociation-francaise-dingenierie-systeme/. Page consultée le 16 janvier 2020. 2020.

[46] G. M. LUCIANO LAVAGNO et B. V. SELIC. « UML for real : design of embedded real-time systems ». In : *design of embedded real-time systems*. Kluwer Academic, 2003.

[47] M. MATTER et MODELS. *Semantic Information Processing*. Proc : International Federation of Information Processing Congress 1965, 1998 .

[48] P. F. MOUTARDE. *Arbres de Décision et Forêts Aléatoires*. Paris : MINES ParisTech,PSL, 2017 (cf. p. 48, 49).

[49] *NumPy.«https://numpy.org/*.

[50] OMG. *Object Constraint Language (OCL)*, Object Management Group. http://www.omg.org/ocl/.. Page consultée le 6 février 2020. 2001.

[51] OMG. *The MetaObject Facility Specification(MOF)*. Object Management Group. https://www.omg.org/mof/.. Page consultée le 6 février 2020. 2003.

[52] OMG. *XML Metadata Interchange*. Object Management Group. https://www.omg.org/spec/XMI/About-XMI/.. Page consultée le 6 février 2020. 2003.

[53] ORACLE2019. *Artificielle, Machine Learning, Deep Learning : une histoire de poupées russes*.

[54] J. POOLE et al. *Common warehouse metamodel*. T. 20. John Wiley & Sons, 2002.

[55] *éduction des dimensions*.

[56] T. O.S.S. G. RICHARD SOLEY. MODEL DRIVEN ARCHITECTURE. White Paper. omg.org/mda/mda_files/model_driven_architecture.htm. Page consultée le 6 janvier 2020. 2004.

[57] C. ROLLAND. *Ingénierie des Besoins : L'Approche L'Ecritoire*. 2003 .

[58] M. ROUSE. *égression logistique*.

[59] H. SAADA et al. « Generation of operational transformation rules from examples of model transformations ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2012, p. 546–56.

[60] H. Saada et al. « Learning Model Transformations from Examples using FCA : One for All or All for One ? » In : *CLA : Concept Lattices and their Applications*. Universidad de Malaga. 2012, p. 45–56 .

[61] C. Salperwyck, V. Lemaire et C. Hue. « Classifieur naïf de Bayes pondéré pour flux de données ». In : *EGC*. 2014 .

[62] B. Schätz. « Formalization and rule-based transformation of EMF Ecore-based models ». In : *International Conference on Software Language Engineering*. Springer. 2008, p. 227–244 .

[63] M. Strommer, M. Murzek et M. Wimmer. « Applying model transformation by-example on business process modeling languages ». In : *International Conference on Conceptual Modeling*. Springer. 2007, p. 116–125.

[64] M. Strommer et M. Wimmer. « A framework for model transformation by-example : Concepts and tool support ». In : *International Conference on Objects, Components, Models and Patterns*. Springer. 2008, p. 372–391 .

[65] J. Sztipanovits et al. « MULTIGRAPH : an architecture for model-integrated computing ». In : *Proceedings of First IEEE International Conference on Engineering of Complex Computer Systems. ICECCS'95*. 1995, p. 361–368.

[66] G. Taentzer et al. « Model transformation by graph transformation : A comparative study ». In : (jan. 2013).

[67] S. Thirumuruganathan. *Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm* .

[68] *des attributs , Réduction des données et le travail de transformation , Septembre sur la sélection du sous-ensemble*

[69] H. Vangheluwe et J. De Lara. « Meta-models are models too ». In : *Proceedings of the Winter Simulation Conference*. T. 1. IEEE. 2002, p. 597–605 .

[70] D. Varró. « Model transformation by example ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2006, p. 410–424 .

[71] *Voilà, des dashboards à partir de vos Jupyter Notebooks* .

[72] M. Wimmer et al. « Towards model transformation generation by-example ». In : *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. IEEE. 2007, 285b–285b .

[73] M. M. Zloof. « Query by example ». In : *Proceedings of the May 19-22, 1975, national computer conference and exposition*. 1975, p. 431–438 .