ÉCOLE POLYTECHNIQUE DE MONTRÉAL

LOG8371- Ingénierie de la qualité en logiciel

TP1 : Plan et test

RÉALISÉ PAR :

HIBA BAGANE - 1679837
SALOUA DJEBELI - 1719832
SAMIA JALIL - 1720168

REMIS À :
Saif-eddine Sajid
Seddik BENYAHIA

21 février 2019

Table des matières

Introduction

Dans le cadre de ce travail, nous allons préparer un plan d'assurance qualité pour le logiciel Weka. Weka (*Waikato Environment for Knowledge Analysis*) est un logiciel libre développé par un groupe de recherche de l'université Waikato et issue sous la licence **GNU**. Il s'agit d'une collection d'algorithmes d'apprentissage machine - ***Machine Learning*** pour la fouille de données - ***Data Mining***. Le logiciel contient des outils pour la préparation, la classification, la régression, le regroupement - ***Clustering***, l'association et la visualisation des données. De plus, il est possible d'utiliser Weka pour traiter des mégadonnées et mettre en pratique l'apprentissage approfondi - ***Deep Learning***.

Le code source utilisé dans ce travail est développé avec le langage de programmation Java. Les algorithmes sont accompagnés par des tests dans l'entrepôt Github du projet.

Weka possède trois composants principaux et chacun de ces derniers possède un rôle spécifique :

- ***Explorer :*** contient tous les paquetages importants de Weka tel que les algorithmes d'apprentissage, les associations, la sélection des attributs et la visualisation etc..
- ***Experimenter :*** permet de créer et d'exécuter différents algorithmes d'apprentissage sur plusieurs données en mode batch et d'analyser les résultats.
- ***Knowledge Flow :*** ce module représente graphiquement les même fonctionnalités d'Explorer.

1. Exigences fonctionnelle

Les fonctionnalités disponibles dans Weka sont les suivantes :

- Apprentissage machine;
- Fouille de données;
- Prétraitement;
- Classification;
- Régression linéaire;
- Clustering;
- Règles d'association;
- Sélection d'attributs;
- Expériences et modèles;
- Flot de travail;
- Visualisation de modèles.

2. Critères et sous-critères de qualités cibles

Les tests de logiciel représentent un facteur critique pour la validation et la vérification d'un système d'apprentissage comme Weka. Dans cette section, nous allons présenter 3 critères de qualité et expliquer leur importance dans le projet.

Tableau 1 : Sous-critères de qualité pour le critère *Functional suitability*

Critère de Qualité : Functional suitability			
Sous-critères	**Description**	**Justifications**	**Objectifs**
Complétude	Ce type de test permet de mesurer le degré pour lequel l'ensemble de fonctions couvre toutes les tâches et objectifs de l'utilisateur.	Il est primordial de vérifier que les analyses faites par les algorithmes de prédiction de Weka sont complètes, car cela va affecter les décisions prises par les utilisateurs.	La complétude des tâches des différents algorithmes de Weka doit être atteindre au moins 95%.
Exactitude	Les tests d'exactitude ont pour but de vérifier que les réponses proposées par le logiciel sont bien celles attendues et avec le niveau de précision attendu. Ces informations sont généralement fournies par des spécifications fonctionnelles.	Les algorithmes de prédiction de Weka touchent plusieurs domaine à savoir : la santé, l'économie etc. Donc, les résultats doivent répondre aux besoins de leurs utilisateurs.	L'exactitude des analyses faites par les algorithmes de Weka doit être supérieure ou égale à 90%.

Tableau 2 : Sous-critères de qualité pour le critère Fiabilité

Critère de Qualité : Fiabilité			
Sous-critères	**Description**	**Justifications**	**Objectifs**
Récupérabilité	Ce type de test permet de vérifier si le système peut récupérer les données affectées directement en cas d'interruption ou de défaillance,et rétablir l'état souhaité du système	WEKA travaille sur des logiciels permettant de prédire des données et des etats pour dans différents secteurs. Il faut que leur programme soit efficace, c'est-à-dire que les fautes sont récupérables	Le temps de récupération doit être au maximum 3 minutes en cas de défaillance.
Maturité	Ce type de test permet de mesurer le degré pour lequel un système répond aux besoins de fiabilité en mode	Les fichiers à analyser par les algorithme de WEKA peuvent contenir des données inutiles (nom du fichier, date	Weka doit doit être en mesure de s'adapter et s'ajuster avec les besoins des utilisateurs dans plus de 80%

	de fonctionnement normal et d'assurer que les requis des utilisateurs sont bien couverts.	…) , les algorithmes de WEKA doivent être en mesure de filtrer ces données et de les ignorer.	des cas.
Tolérance aux fautes--- Robustesse	Ce type de test permet de mesurer le degré pour lequel un système, un produit ou un composant fonctionne comme prévu malgré la présence des défauts matériels ou logiciels.	Weka cible des domaines très importants et parfois les donnés qu'elle traite sont très sensibles. Les algorithmes développés par Weka doivent être plus ou moins capables de gérer les défauts matériels ou logiciels pour que ces derniers n'impactent pas les analyses fournis par ces algorithmes.	Le système de WEKA doit être capable de traiter les fautes et les erreurs dans 80% des cas.

Tableau 3 : Sous-critères de qualité pour le critère Maintenabilité

Critère de Qualité : Maintainability			
Sous-critères	**Description**	**Justifications**	**Objectifs**
Réutilisabilité	Ce type de test permet de mesurer le degré pour lequel les algorithmes développés peuvent être réutilisés sur d'autres logiciels.	Les algorithmes développés par Weka sont utilisés pour expérimenter des différentes données dans n'importe quel domaine. Avec ce type de test, on s'assure que ces algorithmes peuvent être réutilisés sur n'importe quel type de données.	Les algorithmes de Weka doivent être réutilisés sur au moins 80% des différents types de données
Modifiabilité	Ce type de test permet de mesurer le degré pour lequel un système peut être efficacement modifié sans introduire de défauts ou une dégradation de la qualité des fonctionnalités existantes.	Vu les dépendances entre les différents modules de Weka, ce test est nécessaire pour s'assurer que les changements faits sur un module n'affectent pas les autres modules.	Le fonctionnement de Weka doit avoir toujours les même comportements attendus suite à l'ajout ou la modification de nouvelles fonctionnalités.

3. Plan de vérification et validation des objectifs

3.1. Plan de test

Portée des tests :

Nous avons choisi 5 algorithmes à tester :

1. **Association.**Apriori**:** cet algorithme permet de générer les règles d'association sur les différents attributs contenus dans les fichiers d'extension .arff.

2. **Association.**FPGrowth**:** cet algorithme permet de trouver des ensembles d'articles volumineux sans génération de candidats. Il permet de réduire itérativement le support minimum jusqu'à ce qu'il trouve le nombre requis de règles avec la métrique minimale donnée.

3. **Clusters.**SimpleKMeans**:** cet algorithme permet de gérer automatiquements les jeux de données selon des critères spécifiques lors des calculs des distances. L'algorithme WEKA SimpleKMeans utilise une mesure de distance euclidienne pour calculer les distances entre les instances et les clusters.

4. **Classifiers.Functions.**LinearRegression**:** cet algorithme utilise la régression linéaire pour la sélection d'un modèle selon le critère AKAIKE et il est capable de gérer des instances pondérées.

5. **Classifiers.Functions.**Logistic**:** cet algorithme permet de construire des modèles de logistique linéaire. LogitBoost avec des fonctions de régression simples en tant que apprenants de base et est utilisé pour adapter les modèles logistiques. Le nombre optimal d'itérations LogitBoost à effectuer est validé de manière croisée, ce qui conduit à la sélection automatique d'attributs.

Déroulement des tests :

Nous allons commencer par exécuter les tests unitaires disponibles déjà dans weka. Ensuite, nous allons exécuter les tests d'intégration. Finalement, on exécute les tests de régression.

Environnement de test :

Pour exécuter nos tests, on aura besoin :

- Eclipse (JUnit)
- Un serveur Jenkins (configuré avec JDK8 ant/maven)
- Github

3.1.1. Description des tests

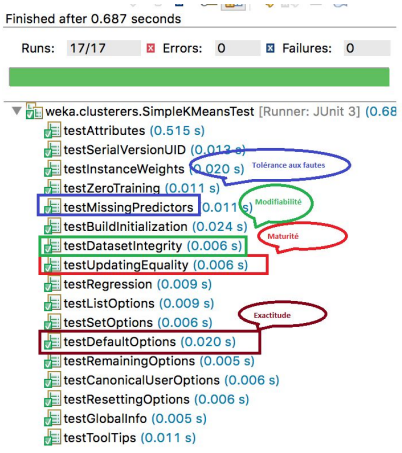
Tableau 4 : Critères de qualité couverts par les tests choisis

Tests unitaires							
Algorithmes	**Critères couverts**						
	Exactitude	**Complétude**	**Modifiabilité**	**Maturité**	**Tolérance aux fautes Robustesse**	**Réutilisabilité**	**Récupérabilité**
FPGrowth	x			x	x		
Apriori			x	x		x	x
SimpleKMeans	x			x	x		
LinearRegression		x	x				x
Logistc	x	x		x		x	

3.1.2. Rapports de tests

3.1.2.1. Résultats

➔ Tous les tests réussissent après leur exécution (voir **Figure 1**).



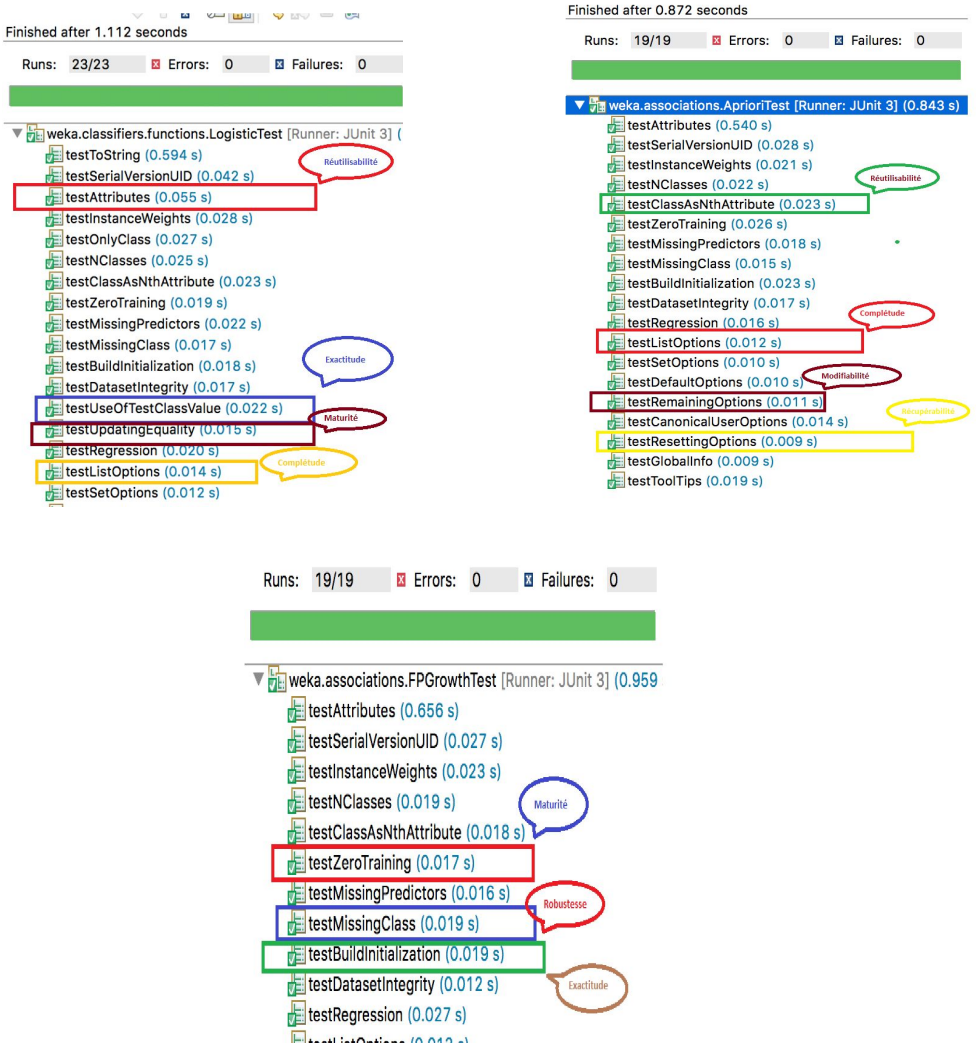


Figure 1 : Résultats de l'exécution des tests sur les 5 algorithmes choisis.

3.2. Stratégies

3.2.1. Intégration continue

Pour cette partie, nous avons choisi de configurer notre entrepôt Github avec l'outil d'intégration continue Jenkins. Nous avons configuré et déployé notre serveur Jenkins, ensuite nous avons ajouté un webhook dans la configuration de notre entrepôt de code. Le projet a été configuré sur Jenkins pour pouvoir effectuer un build automatique du projet Weka avec Maven et JDK 8. Les tests seront exécutés pendant le build et les résultats seront disponibles dans la vue du projet sur Jenkins. Après chaque ajout de code dans l'entrepôt, un build automatique sera lancé par Jenkins. Si tous les tests réussissent, le build sera réussi et donc stable. Dans le cas où des tests échouent, le build sera instable.

Résultats d'intégration obtenus avec Jenkins :

All Failed Tests

Test Name	Duration	Age
weka.filters.unsupervised.attribute.PrincipalComponentsTest.testBuffered	7 ms	1
weka.classifiers.functions.GaussianProcessesTest.testUseOfTestClassValue	12 ms	2
weka.attributeSelection.PrincipalComponentsTest.testRegression	26 ms	5
weka.classifiers.evaluation.EvaluationTest.testRegression	22 ms	5
weka.classifiers.functions.GaussianProcessesTest.testRegression	19 ms	5
weka.classifiers.functions.GaussianProcessesTest.testBuildInitialization	16 ms	5
weka.classifiers.functions.LinearRegressionTest.testRegression	9 ms	5
weka.classifiers.meta.ClassificationViaRegressionTest.testRegression	16 ms	5
weka.classifiers.rules.M5RulesTest.testRegression	16 ms	5
weka.classifiers.trees.M5PTest.testRegression	50 ms	5

On remarque que le build est instable sur Jenkins, car il y a 10 tests (majoritairement des tests de régression) qui échouent. Les tests échouent, car les sorties obtenues ne correspondent pas au sorties attendues par les assertions.

3.2.2. Revue et inspection

Le but des activités de revue et d'inspection est de détecter les défauts d'une manière manuelle. Les revues de code se font par une ou plusieurs personnes afin d'inspecter la qualité de code, son fonctionnement et son efficacité. Pour le projet Weka, ce genre d'activité est très important. Afin de vérifier et de valider la qualité du code produit dans ce projet, nous avons estimé qu'il serait nécessaire d'effectuer une séance de revue de code après la complétion de chaque fonctionnalité, ajout ou modification dans Weka. La revue de code peut se faire sur Github. De plus, pour les activités d'inspection, il faut toujours produire des checklists et les mettre à jour après la complétion de chaque activité.