

“Success in creating AI would be the biggest event in human history. Unfortunately, it might also be the last, unless we learn how to avoid the risks.”

Stephen Hawking

The Independent 01/05/2014

Table des figures

1.1	Import of weka libraries	2
1.2	Import of weka libraries	3
2.1	Dataset Labor	4
2.2	Rechercher des valeurs vides	5
2.3	Première instance du Labor Dataset	6
2.4	méthode de remplacement des valeurs manquantes	6
2.5	Instances après remplacement avant normalisation	7
2.6	Instances après remplacement après normalisation	7
3.1	Détails statistiques de l'attribut	8
3.2	Représentation du dataset à l'aide d'un histogramme	9
3.3	Représentation du dataset à l'aide de boîtes à moustaches	10

Chapitre 1

Le framework Weka

1.1 Qu'est ce que Weka

Weka est une collection d'algorithmes d'apprentissage automatique pour les tâches d'exploration de données. Il contient des outils pour la préparation des données, la classification, la régression, le clustering, l'exploration de règles d'association et la visualisation. Weka propose à ses utilisateurs une interface simple d'utilisation et lui permettant au mieux de visualiser, d'analyser et de traiter les données du dataset que celui-ci choisit. Weka est un logiciel open source distribué sous licence GNU General Public License.

1.2 Weka Wrapper avec python

Pour le developpement de cette application nous avons choisi d'utiliser le langage **Python** pour la flexibilité ainsi que la documentation riche qu'offre ce langage. Fort heureusement un Wrapper du framework Weka est disponible pour ce langage et est simple à télécharger et à mettre en place. Pour utiliser ce dernier il suffira d'importer Weka ainsi que certaines fonctions comme suit :

```
import weka.core.jvm as jvm
from weka.core.converters import Loader, Saver
```

FIGURE 1.1 – Import of weka libraries

Le **Weka-Wrapper** pour python offre des méthodes similaires à celles présentes sur l'API Java car ce wrapper est justement converti à partir de l'API Java, nous trouverons donc des méthodes telles que :

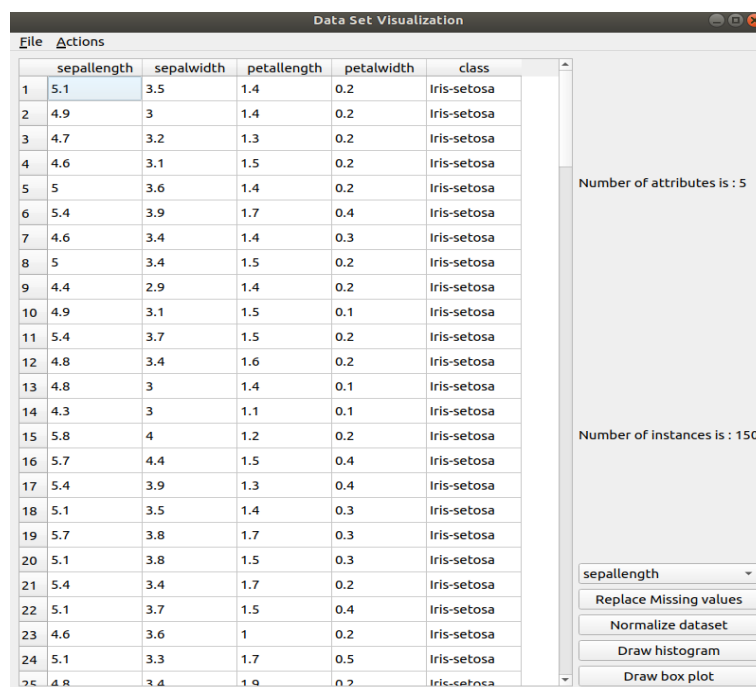
- Loaders and Savers
- Filters
- Classifiers
- Clusterers
- Attribute selection
- Associators
- Experiments

1.3 Interface Utilisateur

Pour la IHM notre choix s'est posé sur **PyQt5**.

Qt est un ensemble de bibliothèques C ++ multi-plates-formes qui implémentent des API de haut niveau pour accéder à de nombreux aspects **Desktop** modernes. PyQt5 est un ensemble complet de liaisons Python pour **Qt v5**. Il est implémenté de telle sorte à permettre à Python d'être utilisé comme langage de développement d'application alternatif au C ++ sur toutes les plates-formes prises en charge, y compris iOS et Android. PyQt5 peut également être intégré à des applications basées sur C ++ pour permettre aux utilisateurs de ces applications de configurer ou d'améliorer les fonctionnalités de ces applications.

Pour illustrer notre propos voici un aperçu de notre interface :



	sepal.length	sepal.width	petal.length	petal.width	class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa
25	4.8	3.4	1.9	0.2	Iris-setosa

FIGURE 1.2 – Import of weka libraries

Chapitre 2

Preprocessing du dataset

2.1 Étude du contenu du dataset

Pour la suite de ce document nous prendrons comme exemple le dataset **Labor**. Nous avons choisis ce dataset est complet :

- Contient des valeurs numeriques non normalises.
- Contient des valeurs nominales.
- contient un nombre important de valeurs manquantes.

Notre application va nous permettre en premier lieu d'avoir un aperçu sur le dataset, ses attributs et leur nombre et ses instances et leur nombre comme suit :

The screenshot shows a web application titled "Data set visualization". It features a menu bar with "File" and "Actions". The main area displays a table with 17 columns and 32 rows of data. The columns are: duration, increase-first, increase-second, increase-third, living-adjust, working-hours, pension, standby-pay, shift-different, vacation-allowance, statutory-holiday, vacation, disability-allowance, pension-to-dent, retirement-assistance, and pension-to-dent. The rows contain numerical and categorical data, with many missing values represented by "?".

On the right side, there is a sidebar with the following information:

- Number of attributes is : 17
- Number of instances is : 57
- A dropdown menu for "duration" with a value of "3".
- Buttons: "Replace Missing values", "Normalize dataset", "Draw histogram", and "Draw box plot".

	duration	increase-first	increase-second	increase-third	living-adjust	working-hours	pension	standby-pay	shift-different	vacation-allowance	statutory-holiday	vacation	disability-allowance	pension-to-dent	retirement-assistance	pension-to-dent
1	1	5	?	?	?	40	?	?	2	?	11	average	?	?	yes	?
2	2	4.5	5.8	?	?	35	ret_allw	?	?	yes	11	below_aver...	?	full	?	full
3	?	?	?	?	?	38	empl_contr	?	5	?	11	generous	yes	half	yes	half
4	3	3.7	4	5	tc	?	?	?	?	yes	?	?	?	?	yes	?
5	3	4.5	4.5	5	?	40	?	?	?	?	12	average	?	half	yes	half
6	2	2	2.5	?	?	35	?	?	6	yes	12	average	?	?	?	?
7	3	4	5	5	tc	?	empl_contr	?	?	?	12	generous	yes	none	yes	half
8	3	6.9	4.8	2.3	?	40	?	?	3	?	12	below_aver...	?	?	?	?
9	2	3	7	?	?	38	?	12	25	yes	11	below_aver...	yes	half	yes	?
10	1	5.7	?	?	none	40	empl_contr	?	4	?	11	generous	yes	full	?	?
11	3	3.5	4	4.6	none	36	?	?	3	?	13	generous	?	?	yes	full
12	2	6.4	6.4	?	?	38	?	?	4	?	15	?	?	full	?	?
13	2	3.5	4	?	none	40	?	?	2	no	10	below_aver...	no	half	?	half
14	3	3.5	4	5.1	tc	37	?	?	4	?	13	generous	?	full	yes	full
15	1	3	?	?	none	36	?	?	10	no	11	generous	?	?	?	?
16	2	4.5	4	?	none	37	empl_contr	?	?	?	11	average	?	full	yes	?
17	1	2.8	?	?	?	35	?	?	2	?	12	below_aver...	?	?	?	?
18	1	2.1	?	?	tc	40	ret_allw	2	3	no	9	below_aver...	yes	half	?	none
19	1	2	?	?	none	38	none	?	?	yes	11	average	no	none	no	none
20	2	4	5	?	tc	35	?	13	5	?	15	generous	?	?	?	?
21	2	4.3	4.4	?	?	38	?	?	4	?	12	generous	?	full	?	full
22	2	2.5	3	?	?	40	none	?	?	?	11	below_aver...	?	?	?	?
23	3	3.5	4	4.6	tc	27	?	?	?	?	?	?	?	?	?	?
24	2	4.5	4	?	?	40	?	?	4	?	10	generous	?	half	?	full
25	1	6	?	?	?	38	?	8	3	?	9	generous	?	?	?	?
26	3	2	2	2	none	40	none	?	?	?	10	below_aver...	?	half	yes	full
27	2	4.5	4.5	?	tc	?	?	?	?	yes	10	below_aver...	yes	none	?	half
28	2	3	3	?	none	33	?	?	?	yes	12	generous	?	?	yes	full
29	2	5	4	?	none	37	?	?	5	no	11	below_aver...	yes	full	yes	full
30	3	2	2.5	?	?	35	none	?	?	?	10	average	?	?	yes	full
31	3	4.5	4.5	5	none	40	?	?	?	no	11	average	?	half	?	?
32	3	3	?	?	tc	40	none	?	5	no	10	below_aver...	yes	half	yes	full

FIGURE 2.1 – Dataset Labor

Ce dataset est constitué de :

- 12 attributs en incluant l'attribut **class**, et 57 instances. Nous savons cela grâce aux deux attributs de Weka : **num ttributs** et **num instances**.
- Contient des valeurs manquantes. Nous pouvons le savoir grâce à une méthode que nous avons écrit et qui détecte s'il existe des ? ou quelques fois des NaN dans notre dataset.
- Contient deux classes **good** et **bad**. Ces dernières ne contiennent pas de valeurs manquantes.

Et comme tout dataset nous devons par une étape de pre-processing qui vont permettre d'avoir une bonne visualisation et une bonne compréhension de ce dernier. Dans ce projet nous nous attardons sur deux étapes de pre-processing.

2.2 Remplissage des valeurs manquantes

Nous allons présenter les étapes suivies pour remplacer ces valeurs manquantes ainsi que les méthodes les plus importantes utilisées à cet effet :

- Verifier si le dataset contient des valeurs manquantes et si c'est le cas, remplacer les ? par des NaN car nous pouvons tomber sur des dataset où de nouvelles instances sont ajoutées et les valeurs manquantes différemment représentées, la norme sera donc Nan.

```
def contains_missing_valeus(self, attribute):  
    if not self.df[attribute].isnull().empty:  
        return True  
    else:  
        return False
```

FIGURE 2.2 – Rechercher des valeurs vides

- Pour chaque attribut, nous devons savoir si ce dernier est **Numerique** ou **Nominal**. Cette information est importante car dans le cas d'une valeur numérique nous remplaçons cette dernière par la **moyenne** de l'attribut et dans le cas d'une valeur nominale, celle ci sera remplacée par le **Mode** de l'attribut.

- Le remplacement des valeurs se fait dépendamment de la classe de l'instance, prenons dans notre dataset la première instance. Dans ce cas nous voyons bien que la classe de l'instance est **good**, donc ici pour remplacer une valeur numérique nous utilisons la moyenne des valeurs de l'attribut ayant pour classe **good** et idem pour les valeurs nominales nous utiliserons le Mode des valeurs ayant pour **good**

	base-first	increase-second	increase-third	if-living-adjust	working-hours	pension	standby-pay	lift-differenti	cation-allowa	atutory-holida	vacation	n-disability-as	ution-to-dent	avement-assist	ution-to-heal	class
1	?	?	?	40	?	?	2	?	11	average	?	?	yes	?	good	

FIGURE 2.3 – Première instance du Labor Dataset

Voici la méthode principale qui effectue ce remplacement. Evidemment il y a beaucoup d'autres méthodes derrière que nous ne détaillerons pas dans ce document :

```
def fill_missing_values_class_dependant(self):
    attributes = self.get_attributes()
    classes = self.get_classes()
    for att in attributes:
        if self.contains_missing_values(att) is True:
            for class_item in classes:
                if self.is_nominal(att):
                    mode_value = self.mode_calcul_missing(att, class_item)
                    self.df.loc[self.df["class"] == class_item, att] = self.df.loc[self.df["class"] == class_item, att].fillna(mode_value)
                else:
                    avg_value = self.mean_calcul_missing(att, class_item)
                    self.df.loc[self.df["class"] == class_item, att] = self.df.loc[self.df["class"] == class_item, att].fillna(avg_value)
```

FIGURE 2.4 – méthode de remplacement des valeurs manquantes

Après avoir rajouté les valeurs manquantes nous passons à présent à la normalisation des valeurs.

2.3 Normalisation

La normalisation permet d'ajuster des valeurs (représentant typiquement un ensemble de mesures) suivant une fonction de transformation pour les rendre comparables avec quelques points de référence spécifiques (par exemple, une unité de longueur ou une somme). :

Dans notre cas la normalisation est importante car nous avons des valeurs dans des intervalles drastiquement différents (C'est particulièrement le cas pour l'attribut **working-hours** qui a des valeurs entre 35 et 40 ce qui est conséquent par rapport aux autres attributs numériques). Pour cela nous normalisons toutes les valeurs en utilisant la norme **MinMax** qui va mettre toutes les valeurs entre 0 et 1. Prenons un exemple sur les 10 premières instances du dataset avant et après normalisation :

	duration	>increase-first	increase-second	increase-third	if-living-adjust	working-hours	pension	standby-pay	hi-ft-differenti	vacation-allowa	statutory-holida	vacation	n-disability-as	nuton-to-dent	avement-assist	nuton-tc
1	1.0	5.0	4.4580645...	4.5545454...	'none'	40.0	'empl_contr'	11.4	2.0	'yes'		11.0	'average'	'yes'	'full'	'full'
2	2.0	4.5	5.8	4.5545454...	'none'	35.0	'ret_allw'	11.4	5.8636363...	'yes'		11.0	'below_ave...	'yes'	'full'	'yes'
3	1.25	4.4194444...	4.4580645...	4.5545454...	'none'	38.0	'empl_contr'	11.4	5.0	'yes'		11.0	'generous'	'yes'	'half'	'yes'
4	3.0	3.7	4.0	5.0	'tc'	37.40625	'empl_contr'	11.4	5.8636363...	'yes'		11.515151...	'generous'	'yes'	'full'	'yes'
5	3.0	4.5	4.5	5.0	'none'	40.0	'empl_contr'	11.4	5.8636363...	'yes'		12.0	'average'	'yes'	'half'	'yes'
6	2.0	2.0	2.5	4.5545454...	'none'	35.0	'empl_contr'	11.4	6.0	'yes'		12.0	'average'	'yes'	'full'	'yes'
7	3.0	4.0	5.0	5.0	'tc'	37.40625	'empl_contr'	11.4	5.8636363...	'yes'		12.0	'generous'	'yes'	'none'	'yes'
8	3.0	6.9	4.8	2.3	'none'	40.0	'empl_contr'	11.4	3.0	'yes'		12.0	'below_ave...	'yes'	'full'	'yes'
9	2.0	3.0	7.0	4.5545454...	'none'	38.0	'empl_contr'	12.0	25.0	'yes'		11.0	'below_ave...	'yes'	'half'	'yes'
10	1.0	5.7	4.4580645...	4.5545454...	'none'	40.0	'empl_contr'	11.4	4.0	'yes'		11.0	'generous'	'yes'	'full'	'yes'

FIGURE 2.5 – Instances après remplacement avant normalisation

	duration	>increase-first	increase-second	increase-third	if-living-adjust	working-hours	pension	standby-pay	hi-ft-differenti	vacation-allowa	statutory-holida	vacation	n-disability-as	nuton-to-dent	avement-assist	nuton-tc
1	0.0	0.6	0.4916129...	0.8240469...	'none'	1.0	'empl_contr'	0.7833333...	0.08	'yes'		0.3333333...	'average'	'yes'	'full'	'yes'
2	0.5	0.5	0.7599999...	0.8240469...	'none'	0.6153846...	'ret_allw'	0.7833333...	0.2345454...	'yes'		0.3333333...	'below_ave...	'yes'	'full'	'yes'
3	0.625	0.4838888...	0.4916129...	0.8240469...	'none'	0.8461538...	'empl_contr'	0.7833333...	0.2	'yes'		0.3333333...	'generous'	'yes'	'half'	'yes'
4	1.0	0.3400000...	0.4	0.9677419...	'tc'	0.8004807...	'empl_contr'	0.7833333...	0.2345454...	'yes'		0.4191919...	'generous'	'yes'	'full'	'yes'
5	1.0	0.5	0.5	0.9677419...	'none'	1.0	'empl_contr'	0.7833333...	0.2345454...	'yes'		0.5	'average'	'yes'	'half'	'yes'
6	0.5	0.0	0.0999999...	0.8240469...	'none'	0.6153846...	'empl_contr'	0.7833333...	0.24	'yes'		0.5	'average'	'yes'	'full'	'yes'
7	1.0	0.4	0.6	0.9677419...	'tc'	0.8004807...	'empl_contr'	0.7833333...	0.2345454...	'yes'		0.5	'generous'	'yes'	'none'	'yes'
8	1.0	0.9800000...	0.5599999...	0.0967741...	'none'	1.0	'empl_contr'	0.7833333...	0.12	'yes'		0.5	'below_ave...	'yes'	'full'	'yes'
9	0.5	0.2000000...	1.0	0.8240469...	'none'	0.8461538...	'empl_contr'	0.8333333...	1.0	'yes'		0.3333333...	'below_ave...	'yes'	'half'	'yes'
10	0.0	0.7400000...	0.4916129...	0.8240469...	'none'	1.0	'empl_contr'	0.7833333...	0.16	'yes'		0.3333333...	'generous'	'yes'	'full'	'yes'

FIGURE 2.6 – Instances après remplacement après normalisation

Maintenant que nous avons pré-traité notre dataset nous pouvons passer à l'analyse de ce dernier.

Chapitre 3

Analyse du dataset

3.1 Informations statistiques sur les attributs

Les informations importantes que nous présentons ici sont :

- Le maximum
- Le minimum
- La Moyenne
- Le Mode
- La Mediane
- Le Q1
- Le Q3
- Ainsi qu'est ce que le dataset est symétrique.

Par exemple pour l'attribut **Wage-increase-second-year** nous avons les informations suivantes :

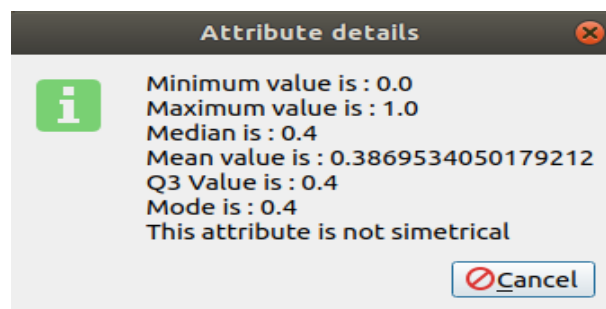


FIGURE 3.1 – Détails statistiques de l'attribut

Ces derniers sont caclulés à l'aide de méthodes que nous offre Weka.

3.2 Histogrammes

Voici à présent une représentation de nos attributs à l'aide d'un **Histogramme** qui met en avant la distribution des attributs ainsi que leur fréquences :

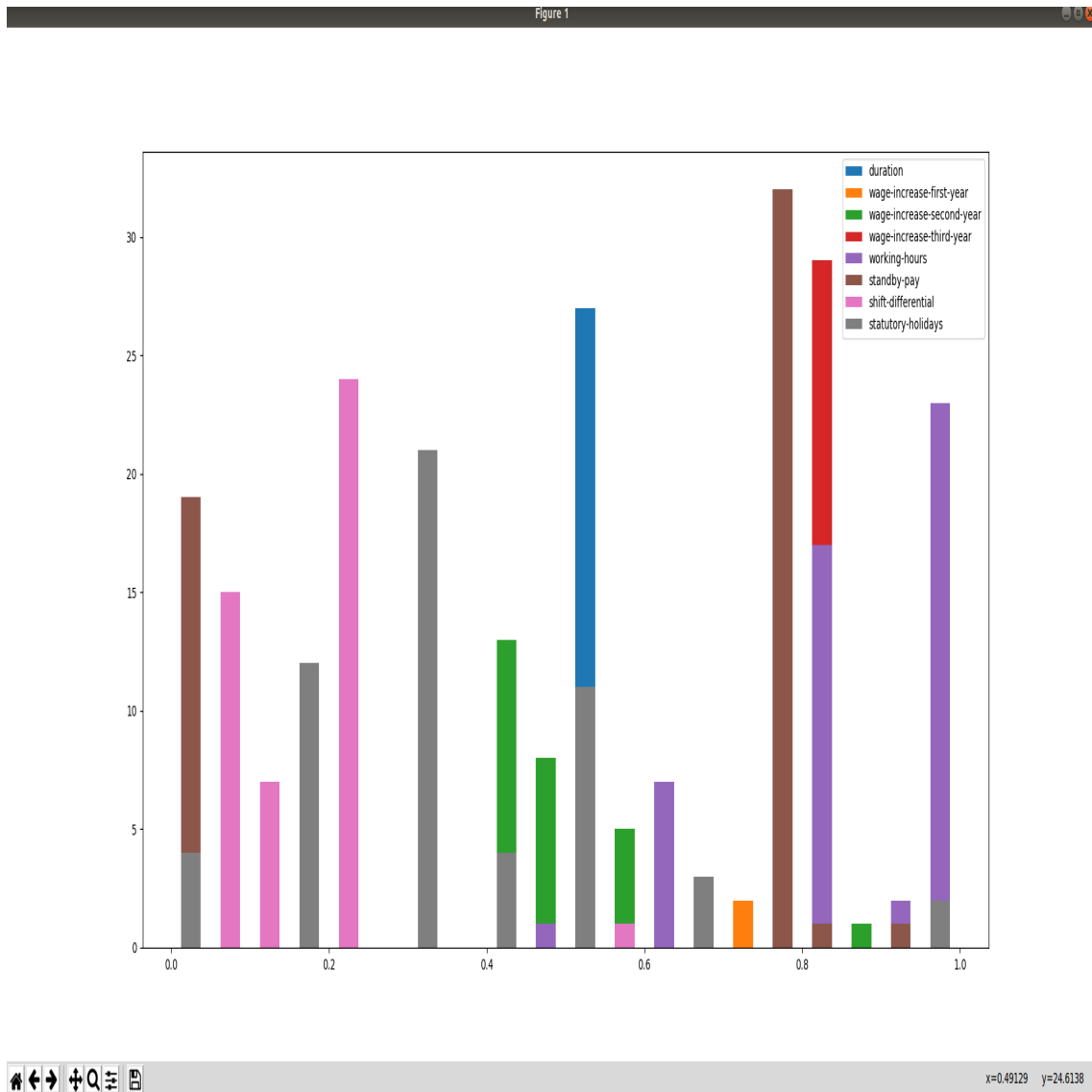


FIGURE 3.2 – Représentation du dataset à l'aide d'un histogramme

Bien sûr l'histogramme prend son sens lorsque les valeurs des attributs sont normalisées.

3.3 Boîtes à moustache

Nous passons à présent à la représentation à l'aide des boîtes à Moustaches ou boxplot en anglais :

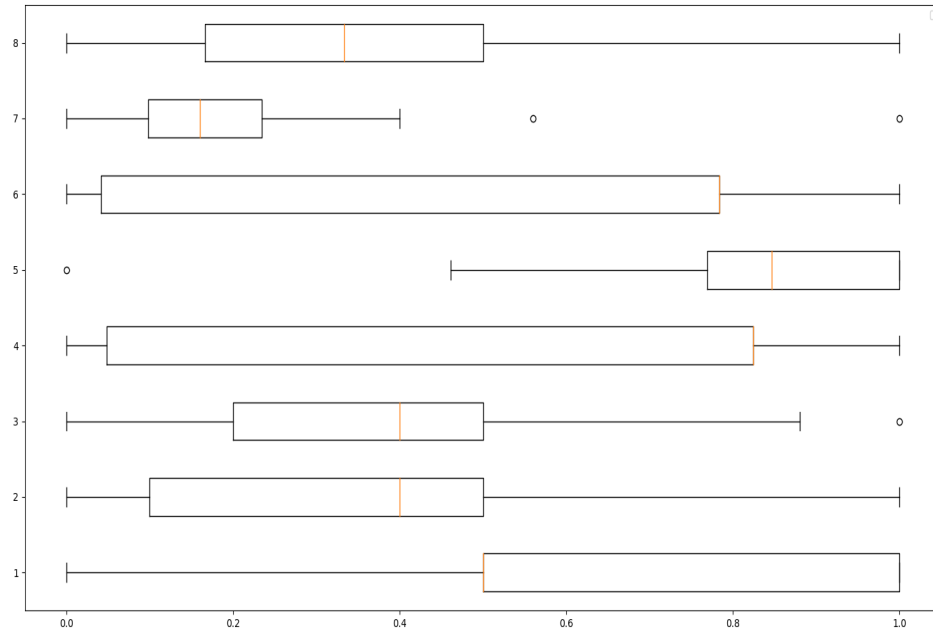


FIGURE 3.3 – Représentation du dataset à l'aide de boîtes à moustaches

3.4 Conclusion

Au cours de ce projet nous avons appris à manipuler des dataset à l'aide du framework Weka, de pouvoir visualiser ce dataset, savoir de quoi il est constitué en termes d'attributs et d'instances, étudier ces dernières, remplacer des valeurs manquantes, les normaliser, les visualiser de manière statistique mais aussi de manière graphique.

Cette première étape est nécessaire car elle nous permettra par la suite de travailler d'une autre manière sur ces dataset afin d'extraire la connaissance de ce dernier, nous parlons ici de méthodes statistiques et/ou méthodes d'apprentissage automatique que nous verrons par la suite.