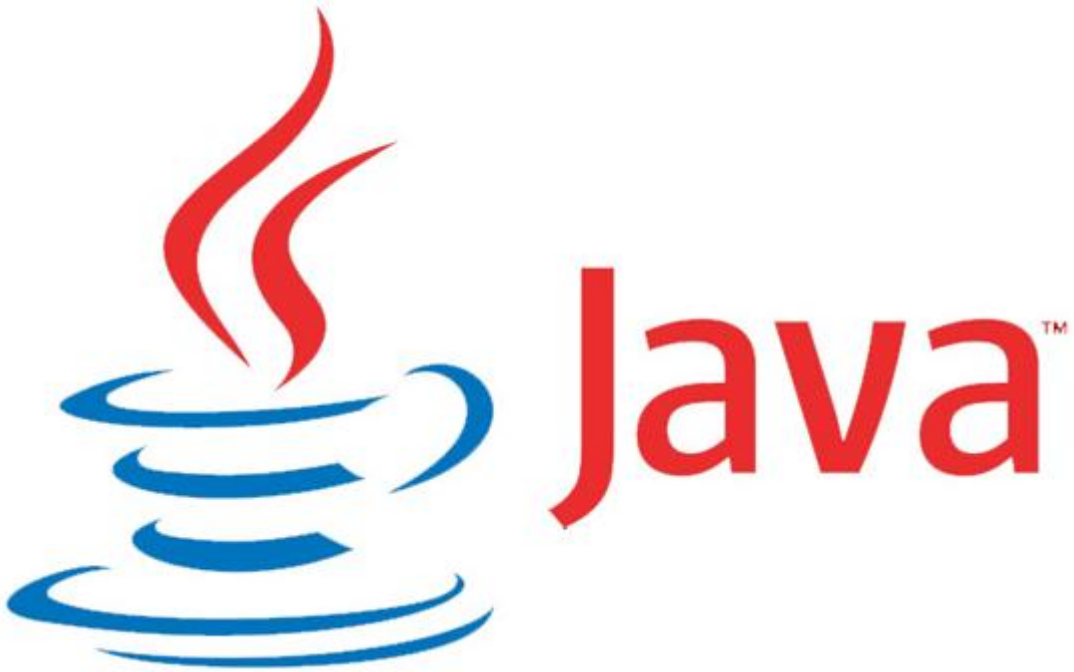
UNIVERSITE DE BRETAGNE SUD
Master 1 Ingénierie Statistique et Décisionnelle

PROJET JAVA



Réalisé par Steven HELLEC

Année universitaire 2011-2012

SOMMAIRE

1. Fonctionnement de l'application

Vous pouvez utiliser l'application en mode graphique sans passer par une invite de commande. Dans l'archive vous trouverez un fichier qui se nomme « Projet_JAVA.jar », c'est un exécutable de mon application.

Dans le répertoire « vidéos » vous trouverez 5 vidéos présentant le fonctionnement de l'application et la comparaison des résultats entre Weka et mon application.

Dans le répertoire « datasets », vous retrouverez plusieurs jeux de données, que vous pouvez utiliser pour tester l'application. Vous pouvez utiliser d'autres fichiers de données que ceux présents dans le dossier. Par contre, ils doivent respecter au moins les 2 règles suivantes :

- Les fichiers XML doivent être conventionnelles à la norme XML de Microsoft Excel ou OpenOffice Calc.

- Pour tous les fichiers de données la première ligne ne peut pas être vide.

Mon application est capable d'importer des tables d'une base de données sur un serveur MySQL en local et à distance. La procédure pour importer des tables est décrite dans les vidéos : « video_apiSH_apiWeka.avi » et « connexion_bdd_distante.avi ».

2. Algorithmes de régression

1.1. Régression avec mon API

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} x_1' \\ x_2' \\ \vdots \\ x_n' \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1K} \\ 1 & x_{21} & \cdots & x_{2K} \\ \vdots & \ddots & \vdots & \\ 1 & x_{n1} & \cdots & x_{nK} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_K \end{pmatrix}$$

1.1.1. Calcul des coefficients de régressions

$$\hat{\beta} = (X'X)^{-1}X'y$$

1.1.2. Calcul des valeurs prédites

$$\hat{y}_i = x_i\hat{\beta}$$

1.1.3. Calcul des erreurs de prédictions

$$\hat{u}_i = y_i - \hat{y}_i$$

1.1.4. Calcul de l'écart-type des erreurs de prédictions

$$\hat{\sigma}_\varepsilon^2 = \frac{1}{n-p-1}\sum_{i=1}^{N}\varepsilon_i^2$$

1.1.5. Calcul de la variance et covariance des estimateurs

Var-Cov= $\hat{\sigma}_\varepsilon^2(X'X)^{-1}$

Ce calcul renvoie une matrice carrée d'ordre p (=le nombre de paramètres).

1.1.6. Calcul de l'écart-type des estimateurs

C'est la racine carré des coefficients de la diagonale de la matrice de variance-covariance.

1.1.7. Calcul des valeurs de Student des coefficients

C'est la division des paramètres de la régression par l'écart-type calculé dans la partie 1.1.6.

1.1.8. Calcul de la signification des estimateurs

C'est le calcul de la p-value des paramètres estimés de la régression. On fait appel à une fonction StudentDist.cdf(), qui se trouve dans l'un des packages de l'université de Montréal. Cette fonction retourne une probabilité.

Formule = 2*(1- StudentDist.cdf(n-p,valeur de student))

n=Nombre d'individus
p=Nombre de paramètres

1.1.9. Calcul de la moyenne de la variable à expliquer

Formule = 1/n*Σ(Y)

1.1.10. Calcul de la somme des carrés des écarts au modèle (SCM)

$$SCE = \sum_i (\hat{y}_i - \bar{y})^2$$

1.1.11. Calcul de la somme des carrés des écarts résiduels (SCR)

$$SCR = \sum_i (y_i - \hat{y}_i)^2$$

1.1.12. Calcul de la somme des carrés des écarts totaux (SCT)

$$SCT = \sum_i (y_i - \bar{y})^2$$

1.1.13. Calcul du carré moyen au modèle (CME)

$$CME = \frac{SCE}{p-1}$$

1.1.14. Calcul du carré moyen au modèle (CMR)

$$CMR = \frac{SCR}{n-p}$$

1.1.15. Calcul de la valeur de Fisher de la régression

$$\mathbf{F} = \frac{CME}{CMR}$$

1.1.16. Calcul de la p-value de la valeur de Fisher

Utilisation d'une fonction contenu dans le package de l'université de montréal pour calculer la valeur de Fisher FisherFDist.cdf().

Formule = 1- FisherFDist.cdf (p-1, n-p, valeur de Ficher)

n=Nombre d'individus
p=Nombre de paramètres

1.1.17. Calcul du R2

$$R^2 = \frac{SCE}{SCT}$$

1.1.18. Calcul du R2 ajusté

$$\bar{R}^2 = 1 - \frac{n-1}{n-p}(1 - R^2)$$

1.1.19. Calcul de la log- vraisemblance du modèle

L(v) = 0.5* (- n * (ln(2 * π) + 1 - log(n) +ln(Σ(res^2))))

n : le nombre d'observations
res : sont les résidus de la régression

1.1.20. Calcul de l'AIC (Akaïke Information Criterion)

AIC= -2* L(v) +2*(p+1)

p : le nombre de pramètre constante comprise

1.1.21. Calcul de l'BIC (Baysian Information Criterion)

BIC = -2*log(L(v))+log(n)*(p+1)

p : le nombre de paramètres, constante comprise
n : le nombre d'observations

1.1.22. Calcul de la null déviance

Null deviance = sum(wt*((Y - moyenne(Y))^2))

wt : vecteur de pondération, qui vaut 1 dans notre cas

1.1.23. Calcul de la déviance résiduelle

Deviance résiduelle = sum(wt*((Y – pred(Y))^2))

wt : vecteur de pondération, qui vaut 1 dans notre cas
pred(Y) : prédictions de Y.

1.1.24. Analyse du type I

Cela permet de récupérer la valeur de la SCEM de chaque variable quand elle rentre dans le modèle et avec l'ordre que l'utilisateur a choisi.

Exemple : Y=X1+X2

On fait la régression de X1 sur Y puis on retire l'effet de X1 sur Y (Y-^Y) --> on récupère la SCEM de la variable X1 à ce moment
On fait la régression de X1 sur X2 puis on retire l'effet de X1 sur X2 (X2-^X1)
On fait la régression de X2 sur (Y-^Y) --> on récupère la SCEM de la variable X2.

1.1.25. Analyse du type III

Cela permet de récupérer la valeur de la SCEM de chaque variable quand toutes les autres variables choisies par l'utilisateur sont déjà entrées.

Exemple : Y=X1+X2

On fait la régression de X1 sur Y puis on retire l'effet de X1 sur Y (Y-^Y)
On fait la régression de X1 sur X2 puis on retire l'effet de X1 sur X2 (X2-^X1)
On fait la régression de X2 sur (Y-^Y) --> on récupère la SCEM de la variable X2 à ce moment

On recommence l'algorithme en changeant l'ordre des variables dans le modèle Y=X2+X1 --> on récupère la SCEM de la variable X1 quand elle rentre en dernière position.

1.1.26. La sélection Forward (en avant)

A chaque pas de construction du modèle on ajoute une nouvelle variable. On sélectionne la variable qui explique le mieux la variable à expliquer au sens de l'AIC. Puis on ajoute une seconde variable explicative à la première qui explique la variable expliquée. Ainsi de suite jusqu'à épuisement des variables sélectionnées.

Exemple :

Y = X1+X2+X3+X4

Pas 1 :

Y=X3

Pas 2 :

Y=X3+X1

Pas 3 :

Y=X3+X1+X4

Pas 4 :

Y=X3+X1+X4+X2

1.1.27. La sélection Backward (en arrière)

On commence l'algorithme avec le modèle complet (toutes les variables sélectionnées par l'utilisateur). A chaque pas de construction du modèle on retire la variable qui fait augmenter le plus l'AIC. On teste tous les sous modèles possibles avec une variable en moins et on garde celui qui explique le mieux la variable à expliquer. Une variable exclue ne peut pas revenir plus tard dans la construction de sous modèle. Ainsi de suite jusqu'à n'avoir plus qu'une seule variable.

Exemple :

Y = X1+X2+X3+X4

Pas 1 :

Y= X1+X2+X3+X4

Pas 2 :

Y=X3+X1+X4

Pas 3 :

Y=X3+X1

Pas 4 :

Y=X3

1.1.28. La sélection Stepwise (pas à pas)

La sélection Stepwise est un mélange des 2 méthodes de sélection précédentes, c'est-à-dire que l'on peut enlever et/ou retirer des variables à chaque pas de construction. De plus, une variable qui a été retiré peut revenir plus tard dans la construction du modèle.

Pour réaliser cette sélection j'ai créé 2 algorithmes.

Le premier construit tous les combinaisons possibles de variables sans remise (algorithme récursive). Donc, pour k variables cela fait (2^k)-1 possibilités.

Le deuxième sélectionne le meilleur modèle à une variable, puis deux, …, jusqu'à p variables.

Exemple :

Y = X1+X2+X3

Tous les modèles possibles :

Y = X1
Y = X2
Y = X3
Y = X1 + X2
Y = X1 + X3
Y = X2 + X3
Y = X1 + X2 + X3

Le meilleur modèle à 1 variable :

Y=X2

Le meilleur modèle à 2 variables :

Y=X3+X1

Le meilleur modèle à 3 variables :

Y= X2+X3+X1

1.2. Régression avec l'API Weka

Pour faire de la régression avec l'API Weka, il a fallu que j'importe la librairie weka.jar dans mon application.

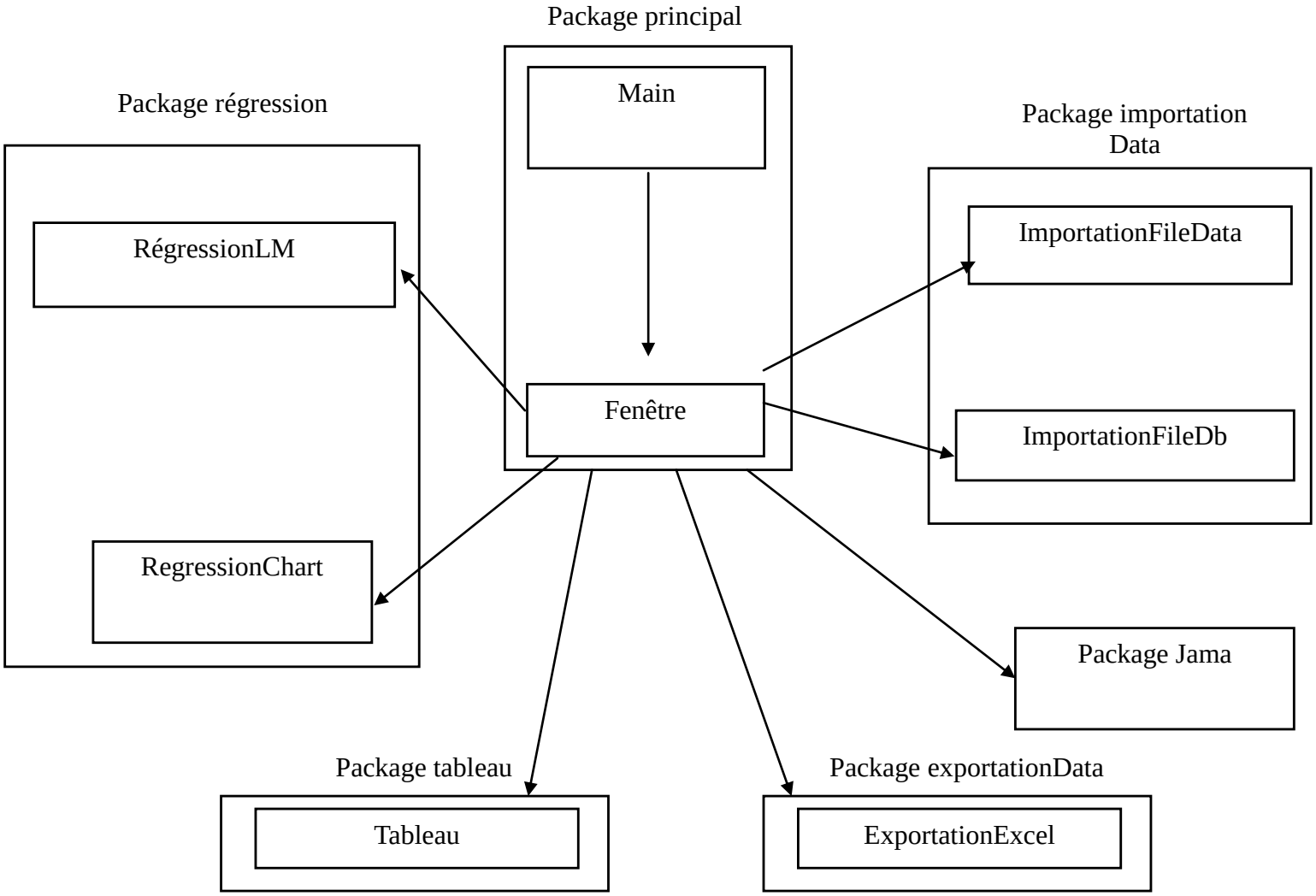
Il est assez simple de faire de la régression avec l'API Weka une fois que les données sont importées. Son inconvénient, c'est qu'elle ne permet pas de sélectionné les variables explicatives que l'on souhaite, elle prend par défaut toutes les variables qui sont contenues dans le jeu de données.

Pour résoudre ce problème, j'ai créé un algorithme qui permet de sélectionner les variables que l'utilisateur souhaite avoir dans sa régression. Weka possède une méthode pour supprimer des variables dans le jeu de données importé. Du coup, pour ne garder que les variables que l'utilisateur souhaite, l'application fait une copie du jeu de données et supprime les variables qui n'ont pas été sélectionnées. Puis, l'API Weka fait la régression sur la copie modifiée du jeu de données.

3. Bibliothèques utilisés

Blass.jar
Colt.jar
Event-1.6.5.jar
Gnujaxp.jar
Hamcrest-core-1.1.jar
Interpreter-1.6.8
iText-2.1.5.jar
jcommon-1.0.16.jar
jfreechart.jar
junit.jar
jxl.jar
language-1.6.7.jar
logger-1.6.4.jar
mysql-connector-java-3.1.8-bin.jar
optimization.jar
servlet.jar
ssj.jar
swt.jar
swtgraphics2d.jar
tcode.jar
weka.jar

4. Architecture de l'application



RégressionLM : Cette classe possède tous les algorithmes pour faire de la régression

RegressionChart : Cette classe permet de créer un graphique (un nuage de points + une droite de régression).

ImportationFileData : C'est la classe qui se charge d'importer tous les fichiers de données (xml, txt, csv, xls) pour faire des graphiques et de la régression par mon API.

ImportationFileDb : C'est la classe qui se charge d'importer des tables dans une base de données de données (seulement MySQL) pour faire des graphiques et de la régression par mon API.

Tableau : Cette classe instancie un tableau de caractère. Elle a été créée pour pallier aux problèmes des erreurs manquantes dans les fichiers de données. Cette classe permet de vérifier s'il y a des valeurs manquantes ou non-numériques et d'extraire les enregistrements qui sont uniquement constitué de valeurs numériques pour faire de la régression.

Fenêtre : contient toutes les données concernant l'IHM de l'application et les procédures événementielles qui font appellent aux autres classes.

Main : contient le main de l'application. C'est la classe qui instancie lance la fenêtre de l'application.

Jama : C'est le package qui contient toutes les classes pour faire du calcul matriciel.

5. Conclusion

Pendant l'élaboration de mon application j'ai rencontré de nombreuses difficultés sur la compréhension du langage avec la notion de classe, de package, d'API, de JRE, etc. La deuxième difficulté était d'écrire les algorithmes de la régression notamment les algorithmes de sélection et la programmation de graphique. Pour remédier à ces difficultés, j'ai fait beaucoup de recherche sur internet.

Ce projet m'a permis de me familiariser avec le langage JAVA. C'est un langage que je ne connaissais pas avant d'arriver en master. Je trouve que c'est un langage très intéressant car il est d'une part exportable sur tous les systèmes d'exploitation et qu'il permet de programmer différent types d'applications : applications consoles, des applications graphiques, des applets, des applications pour téléphones portables, etc.