Travaux Pratiques
Apprentissage Automatique I

TP #2 – Régression Logistique

Professeur

Abdelouahab Moussaoui

Partie 1 — Régression Logistique en action

Dans cette partie nous allons apprendre à réaliser avec R et Python des modèles de *régression logistique*.

Exercice 1 — Application de la Régression Logistique en R

On veut prédire si une masse tissulaire donnée est *bénigne* ou *maligne*. Voyons maintenant comment implémenter la régression logistique à l'aide de l'ensemble de données *BreastCancer* dans le package "mlbench". Vous devrez installer le package "mlbench".
L'objectif ici est de modéliser et de prédire si un spécimen donné (ligne dans l'ensemble de données) est bénin ou malin, sur la base de 9 autres caractéristiques cellulaires. Alors, chargeons les données et ne gardons que les cas complets.

L'ensemble de données comporte 699 observations et 11 colonnes. La colonne Classe est la variable de réponse (dépendante) et elle indique si un tissu donné est malin ou bénin.

Application avec R

Exécuter et commenter le script suivant dans R.

```
# Load data
install.packages('mlbench')

data(BreastCancer, package="mlbench")
bc <- BreastCancer[complete.cases(BreastCancer), ]  # keep complete rows

# remove id column
bc <- bc[,-1]

# convert to numeric
for(i in 1:9) {
  bc[, i] <- as.numeric(as.character(bc[, i]))
}

# Change Y values to 1's and 0's
bc$Class <- ifelse(bc$Class == "malignant", 1, 0)
bc$Class <- factor(bc$Class, levels = c(0, 1))

# Prep Training and Test data.
library(caret)
'%ni%' <- Negate('%in%')  # define 'not in' func
options(scipen = 999)  # prevents printing scientific notations.
set.seed(100)
trainDataIndex <- createDataPartition(bc$Class, p = 0.7, list = F)
trainData <- bc[trainDataIndex, ]
testData <- bc[-trainDataIndex, ]

# Class distribution of train data
table(trainData$Class)


# Down Sample
set.seed(100)
down_train <- downSample(x = trainData[, colnames(trainData) %ni% "Class"],
                         y = trainData$Class)

table(down_train$Class)
```

```r
# Up Sample (optional)
set.seed(100)
up_train <- upSample(x = trainData[, colnames(trainData) %ni% "Class"],
                     y = trainData$Class)

table(up_train$Class)

# Build Logistic Model
logitmod <- glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family = "binomial",
data = down_train)

summary(logitmod)

pred <- predict(logitmod, newdata = testData, type = "response")
pred

# Recode factors
y_pred_num <- ifelse(pred > 0.5, 1, 0)
y_pred <- factor(y_pred_num, levels = c(0, 1))
y_act <- testData$Class

# Accuracy
mean(y_pred == y_act)  # 94%
```

Exercice 2 — Application de la Régression Logistique en Python

Cas python de régression logistique, *k-Fold Cross Validation* et déploiement de la matrice de confusion

- *Introduction*

Dans cette lecture, nous allons parcourir un exemple d'application du modèle de régression logistique de la bibliothèque python *Scikit-learn* pour prédire si une tumeur est maligne ou bénigne en fonction des informations fournies (alias prédicteurs ou caractéristiques).

- *Informations sur l'ensemble de données*

Notre ensemble de données provient du référentiel d'apprentissage automatique de l'UCI (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) et contient les informations d'attribut suivantes :

1. *Exemple de numéro de code : numéro d'identification*
2. *Épaisseur des touffes : 1–10*
3. *Uniformité de la taille des cellules : 1 à 10*
4. *Uniformité de la forme cellulaire : 1–10*
5. *Adhésion marginale : 1–10*
6. *Taille de cellule épithéliale unique : 1 à 10*
7. *Noyaux nus : 1 à 10*
8. *Chromatine fade : 1–10*
9. *Nucléoles normaux : 1–10*
10. *Mitoses : 1 à 10*
11. *Classe : (2 pour bénigne, 4 pour maligne)*

Les données ont été fournies par la base de données originale sur le cancer du sein du Wisconsin

Application avec Python

Exécuter le script suivant en Python.

 # Importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

 # Importation du dataset

Nous divisons le jeu de données en :

- x: variables indépendantes ou prédicteurs
- y: variable dépendante ou prédiction

```
dataset = pd.read_csv('breast_cancer.csv')

x = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

 # Vérification des données manquantes

```
np.any(np.isnan(x)) # is there any missing value in x?
## False

np.any(np.isnan(y)) # is there any missing value in y?
## False
```

 # Fractionnement de l'ensemble de données en ensemble d'apprentissage et en ensemble de test

Nous divisons les données en un ensemble d'apprentissage (pour ajuster notre modèle) et un ensemble de test (pour #tester les prédictions de notre modèle ajusté)

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,
                                                    random_state = 0)
```

 # Entraînement du modèle de régression logistique sur l'ensemble d'apprentissage

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)

LogisticRegression(C = 1.0, class_weight = None, dual = False, fit_intercept = True,
                   intercept_scaling = 1, l1_ratio = None, max_iter = 100,
                   multi_class = 'auto', n_jobs = None, penalty = 'l2',
                   random_state = 0, solver = 'lbfgs', tol = 0.0001, verbose = 0,
                   warm_start = False)
```

 # Prédire les résultats de l'ensemble de tests

```
y_pred = classifier.predict(x_test)
y_pred[0:10]
```

N'oubliez pas que :

- *2 = bénin*
- *4 = malin*

```
comparison = np.concatenate((y_test.reshape(-1,1), y_pred.reshape(-1,1)), axis = 1)
#comparing the real values vs predictions

comparison[0:10,:]
```

Construire la matrice de confusion

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print(cm)
```

Ci-dessous nous pouvons observer, de manière plus conviviale, la matrice de confusion de nos prédictions à l'aide de ce modèle :

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, x_test, y_test)
plt.xlabel("Predicted tumor class")
plt.ylabel("True tumor class")
plt.show()
```

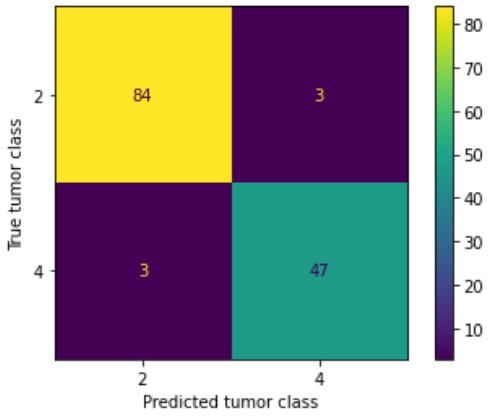


Figure 1. Confusion Matrix (2 = benign tumor, 4 = malignant tumor)

À partir de la matrice de confusion de la figure 1, nous pouvons voir que **84** tumeurs bénignes et **47** tumeurs malignes ont été prédites avec précision, les deux catégories présentant **3** prédictions erronées.

Calcul de la précision avec k-Fold Cross Validation

Enfin, nous calculons la *précision* et *l'écart type* de notre modèle pour voir comment il s'est bien fonctionné pour faire les bonnes prédictions tumorales.

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv=10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

## Accuracy: 96.70 %
## Standard Deviation: 1.97 %
```

Partie II — Utilisation de R

1- Exercice 1 — Modèle Logistique Simple

On a observé p variables sur n individus. On dit qu'il s'agit d'un protocole multivarié.
On a relevé l'âge et la présence(1) ou l'absence (0) d'une maladie cardiovasculaire chez 100 individus. Les données sont stockées dans le fichier **"MCV.txt"**: sur une ligne donnée, la variable **AGE** fournit l'âge d'un individu tandis que la variable **CHD** prend la valeur 1 en cas de présence d'une maladie cardiovasculaire chez cet individu et la valeur 0 sinon. Les variables **ID** et **AGRP** donnent respectivement le numéro d'un individu et sa classe d'âge.

Questions :

1. Charger et afficher les données regroupées en classe d'âge.
2. On souhaite étudier la relation entre CHD et la variable explicative **AGE**. Afficher les données à l'aide d'un nuage de points. Commenter la Figure.
3. Calculons la proportion de malades observée selon les classes d'âge définies par la variable **AGRP**. Définir un vecteur **centre** qui donne les centres de chaque classe puis représenter le nuage de points de **p** versus **centre**. Y a-t-il une liaison entre **CHD** et **AGE** ? Quelle est la forme de ce graphique ? Quel est son intérêt comparativement au graphique précédant ? Quel modèle suggérez-vous d'utiliser ?
4. Commençons pour ajuster une régression logistique de **CHD** en fonction de **AGE**. Commenter les résultats (tests de significativité, nombre de degrés de liberté).
5. Afin de mieux discerner les relations entre les différentes classes, il est demandé de représenter sur un même graphique les proportions selon la classe d'âge et la courbe logistique ajustée.
6. Ajuster de même le modèle **"probit"** puis comparer les deux modèles. Commenter les résultats.
7. Estimer, dans chacun des deux modèles la cote d'un individu âgés de 30 ans. Commenter. Estimer le rapport de cotes correspondant à la variable **AGE**.

Exercice 2 — Modèle Logistique Multiple

Nous traitons un problème de défaut bancaire. Nous cherchons à déterminer quels clients seront en défaut sur leur dette de carte de crédit (ici **defaut** = 1 si le client fait défaut sur sa dette). La variable **defaut** est la variable réponse. La base de données **Defaut** est accessible à partir du package **ISLR** que vous devez installer au préalable.
La base **Defaut** dispose d'un échantillon de tailles 10000 et 3 variables explicatives. Les variables explicatives sont les suivantes :

- **student:** variable à 2 niveaux {0,1} (**student** = 1 si le client est un étudiant).
- **balance:** montant moyen mensuel d'utilisation de la carte de crédit.
- **income:** revenu du client.

Questions :

8. Charger et afficher les données.
9. Commenter les données en utilisant la fonction **summary**.
10. Afin de faciliter le traitement, vous devez transformation de la variable **default** à 0 si Non et 1 si Yes.

11. Construire un modèle de régression logistique avec la variable balance comme variable explicative qualitative. Commenter.
12. Une fois que les coefficients ont été estimés, il est simple de calculer la probabilité de défaut étant donné balance (solde moyen de carte de crédit donné). En utilisant les estimations des coefficients indiqués dans le tableau précédant, prédire la probabilité de **defaut** pour un client qui a une balance de **1000**, **1500**, **2000** et **3000** dollars respectivement. Commenter.
13. Donner le tableau de contingence des variables **default** et **student** puis estimer (avec un calcule à la main) les coefficients du modèle logistique.
14. Estimer P(default = Yes|student = Yes) et P(default = Yes|student = Non).
15. Construire un modèle de régression logistique multiple avec les 2 variables explicatives **student** et **balance**.
16. Construire un modèle de régression logistique multiple avec les 2 variables explicatives **student** et **balance**.
17. Construire un modèle de régression logistique multiple avec les 3 variables explicatives **student** et **balance** et **income**.

Exercice 3 : Modèle linéaire généralisé

Une régression logistique est habituellement utilisée lorsqu'il y a une variable de résultat dichotomique (telle que gagner ou perdre) et une variable prédictive continue qui est liée à la probabilité ou à la probabilité de la variable de résultat. Il peut également être utilisé avec des prédicteurs catégoriques et avec de prédicteurs multiples.
Supposons que nous partons d'une partie du jeu de données **" mtcars"** intégré dans R. Les données ont été extraites du magazine 1974 de Motor Trend US et comprennent la consommation de carburant et 10 aspects de la conception et de la performance automobile pour 32 automobiles (modèles 1973-74). Nous utiliserons **"vs"** comme variable de résultat, **"mpg"** comme prédicteur continu, et **"am"** comme prédicteur catégorique (dichotomique ou binaire).

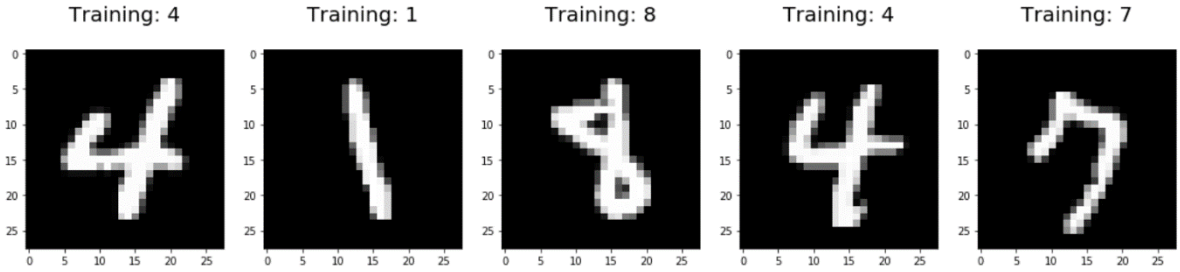
Questions :

1. Charger et afficher les données.
2. Créer un modèle logistique où vous considéré "**mpg"** est la variable prédictive continue et **"vs"** est la variable de résultat qualitative binaire (dichotomique).
3. Interpréter les résultats.
4. Tracer avec la fonction **plot** le graphe des données et du modèle régression logistique.
5. Refaire la même chose avec la fonction **ggplot2**.
6. Refaire les questions précédentes mais avec cette fois ci "**am"** comme variable prédictive continue et **"vs"** comme variable de résultat qualitative binaire (dichotomique).
7. Construire le modèle de régression avec "**mpg"** comme variable prédictive continue, "**am"** comme variable prédictive dichotomique et **"vs"** comme variable de résultat qualitative binaire (dichotomique).
8. Comparer les résultats avec le modèle "**probit"**.
9. Considérer maintenant que le modèle est linéaire. Refaire les mêmes questions et Conclure.

Partie III : Utilisation de Python

Exercice 1 — Régression logistique des chiffres (utiliser la fonction load_digit() de la bibliothèque scilit-learn)

Nous voulons utiliser la régression logistique pour prédire les étiquettes de chiffres en fonction des images. L'image ci-dessus montre un ensemble de chiffres d'apprentissage (observations) issus du jeu de données MNIST dont l'appartenance à une catégorie est connue (étiquettes 0 à 9). Après avoir appris un modèle avec régression logistique, il peut être utilisé pour prédire une étiquette d'image (étiquettes 0–9) à partir d'une image.



Visualiser les images et les étiquettes dans le jeu de données MNIST

Questions :

1. Chargez les données (jeu de données numériques).
2. Affichez les images et les étiquettes (jeu de données numériques).
3. Fractionnez les données en ensembles d'entrainement et de test (jeu de données de chiffres).
4. Créez un Modèle d'apprentissage (jeu de données de chiffres) :
   a. Étape 1. Importez le modèle que vous souhaitez utiliser.
   b. Étape 2. Créez une instance du modèle.
   c. Étape 3. Formation du modèle sur les données, stockage des informations apprises à partir des données.
   d. Étape 4. Prédisez les étiquettes pour les nouvelles données (nouvelles images).
5. Mesurez la performance du modèle (jeu de données numériques).
6. Donner matrice de confusion (jeu de données numériques). Expliquez.

Exercice 2 : Régression logistique (MNIST)

Questions :

1. Téléchargez les données (MNIST). (http://yann.lecun.com/exdb/mnist/)
2. Divisez les données en ensembles d'apprentissage et de test (MNIST).
3. Affichez les images et les étiquettes (MNIST).
4. Créez un Modèle d'apprentissage (MNIST) :
5. Étape 1. Importez le modèle que vous souhaitez utiliser.
6. Étape 2. Créez une instance du modèle.

7. Étape 3. Formation du modèle sur les données, stockage des informations apprises à partir des données.
8. Étape 4. Prédisez les étiquettes pour les nouvelles données (nouvelles images).
9. Mesurez la performance du modèle (MNIST).
10. Affichez les images mal classées avec des étiquettes prédites (MNIST).

Partie IV — Mini-Projet à Rendre

Mini-Projet 1 — Filtrage de Spams

George Forman, chercheur chez Hewlett-Packard, a créé un ensemble de données sur le spam. Il a pris 4601 de ses propres messages électroniques, les a étiquetés et a extrait diverses fonctionnalités. Les caractéristiques comprennent la fréquence de divers mots (par exemple, " money"), des caractères spéciaux (par exemple des signes de dollar) et l'utilisation de majuscules dans le message.
Les données spam peuvent être téléchargées à partir de : ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/

Questions :

Il est demandé de :

1. Créer un modèle de régression logistique à partir des données spams.
2. Décrire et commenter le modèle.
3. Valider le classifieur avec un taux d'erreur acceptable.
4. Le valider par une technique de validation connue (validation croisée par exemple).

Documenter toutes les étapes.

Mini-Projet 2 — CreditCard Fraud Detection

Selon *Infosecurity Magazine*, la fraude a coûté 3,2 billions de livres sterling à l'économie mondiale en 2018 (https://www.infosecurity-magazine.com/news/global-fraud-hits-32-trillion/). Pour de nombreuses entreprises, les pertes liées à la fraude transactionnelle représentent plus de 10 % de leurs dépenses totales. L'inquiétude face à ces pertes massives conduit les entreprises à rechercher constamment de nouvelles solutions pour prévenir, détecter et éliminer la fraude. L'apprentissage automatique est l'une des armes technologiques les plus prometteuses pour lutter contre la fraude financière.

L'objectif de ce projet est de créer un modèle de *régression logistique simple* capable de détecter la fraude dans les opérations de carte de crédit, cherchant ainsi à minimiser le risque et la perte de l'entreprise. Le plus grand défi est de créer un modèle très sensible à la fraude, car la plupart des transactions sont légitimes, ce qui rend la détection difficile.

L'ensemble de données utilisé contient les transactions effectuées par les titulaires de cartes de crédit européennes qui ont eu lieu sur deux jours en septembre 2013.

Il s'agit d'un ensemble de données très déséquilibré, c'est-à-dire qu'il compte 492 transactions frauduleuses, ce qui ne représente que 0,172 % des 284 807 transactions.

Les variables d'entrée sont numériques, résultat d'une transformation PCA. En raison de problèmes de confidentialité, les données originales et d'autres informations complémentaires n'ont pas été mises à disposition.

Les seules variables qui n'ont pas été transformées avec l'ACP sont 'Temps' et 'Valeur'. La variable 'Time' contient les secondes entre chaque transaction et la première transaction

dans l'ensemble de données. La variable 'Montant' fait référence au montant de la transaction.

La variable 'Classe' est la variable réponse (Cible) et vaut "1" en cas de fraude et "0" sinon.

Questions :

Il est demandé de :

1. Charger les données *"creditcard.csv"*
2. Comparaison de la valeur du montant des transactions normales par rapport à la fraude.
3. Créer un modèle de régression logistique à partir des données *"creditcard.csv"*.
4. Décrire et commenter le modèle.
5. Valider le classifieur en utilisant les mesures de performances suivantes : **matrice de confusion** (*confusion matrix*), **précision** (*accuracy*), **Précision**, **Rappel** (*Recall*), **AUC** (*Under of Curve*)

Documenter toutes les étapes.

Mini-Projet 3 — Malicious URLs Detection

le risque d'insécurité des informations réseau augmente rapidement en nombre et en niveau de danger. Les méthodes les plus utilisées par les pirates aujourd'hui consistent à attaquer la technologie de bout en bout et à exploiter les vulnérabilités humaines. Ces techniques incluent l'ingénierie sociale, le phishing, le pharming, etc. L'une des étapes de la conduite de ces attaques consiste à tromper les utilisateurs avec des URL malveillantes. Par conséquent, la détection d'URL malveillantes est d'un grand intérêt de nos jours.

Nous proposons à travers ce projet de développer un algorithme de machine Learning à base de la *régression logistique* pour détecter et classer les URLs malveillantes (Malicious URLs)

Questions :

Il est demandé de :

1. Charger les données *" malicious_phish.csv "*
2. Afficher les statistiques de ces données.
3. Créer un modèle de régression logistique à partir des données *" malicious_phish.csv "*.
4. Valider le modèle.

Documenter toutes les étapes.

Bonne Chance