

# Régression Logistique Multinomiale pour R

RANDRIAMIARIJAONA Nancy

GABRYSCH Alexis

KOCAB Cyril

1<sup>er</sup> décembre 2024

Voici un lien vers le dépôt GitHub : [RegLogMulti](#).

## Table des matières

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introduction</b>                             | <b>2</b> |
| <b>2</b> | <b>Structure du Projet</b>                      | <b>2</b> |
| <b>3</b> | <b>Modèle et Classes de Prétraitement</b>       | <b>2</b> |
| 3.1      | Théorie du Modèle . . . . .                     | 2        |
| 3.1.1    | Softmax . . . . .                               | 2        |
| 3.1.2    | Fonction de coût . . . . .                      | 3        |
| 3.1.3    | Optimisation par descente de gradient . . . . . | 3        |
| 3.2      | Adam Optimizer . . . . .                        | 3        |
| 3.3      | Learning Rate . . . . .                         | 3        |
| 3.4      | Impact sur l'Entraînement . . . . .             | 3        |
| 3.5      | Implémentation dans le Package . . . . .        | 4        |
| 3.6      | Évaluation du Modèle . . . . .                  | 4        |
| <b>4</b> | <b>Classes de Prétraitement</b>                 | <b>4</b> |
| <b>5</b> | <b>Visualisation avec Shiny</b>                 | <b>4</b> |
| 5.1      | Data . . . . .                                  | 5        |
| 5.2      | Preprocessing . . . . .                         | 5        |
| 5.3      | Training . . . . .                              | 6        |
| <b>6</b> | <b>Comparaison avec sklearn</b>                 | <b>7</b> |
| <b>7</b> | <b>Conclusion</b>                               | <b>8</b> |

# 1 Introduction

Le projet **RegLogMulti** vise à fournir une implémentation efficace de la régression logistique multinomiale en R. En utilisant une technique de descente de gradient, ce package permet de gérer à la fois des variables quantitatives et qualitatives, pour la prédiction d'une variable multi-classes. De plus, une interface utilisateur interactive est fournie via une application Shiny, permettant aux utilisateurs de configurer et d'entraîner le modèle avec les hyper-paramètres choisis.

## 2 Structure du Projet

Le package est organisé de la manière suivante :

- **app.R** : Application Shiny, intégrant les différents modules pour la gestion des données, le prétraitement et l'entraînement du modèle.
- **data/** : Contient les fichiers de données utilisés par le projet, tels que `donnees_qualitatives_corrigees.csv` et `iris_extended.csv`.
- **docs/** : Dossier dédié à la documentation, incluant des fichiers LaTeX.
- **R/** : Regroupe les scripts R organisés en différentes sous-catégories :
  - **preprocessing/** : Contient les classes et fonctions pour le prétraitement des données, telles que `OneHotEncoder.R`, `StandardScaler.R`, etc.
  - **modules/** : Inclut les modules Shiny pour la gestion des données (`module_data.R`), le prétraitement (`module_preprocessing.R`) et l'entraînement (`module_training.R`).
  - **model.R** : Définit la classe `MultinomialLogisticRegression` utilisée pour l'entraînement et l'évaluation du modèle.
- **README.md** : Fournit une vue d'ensemble du projet, incluant les instructions d'installation et d'utilisation.

## 3 Modèle et Classes de Prétraitement

### 3.1 Théorie du Modèle

La régression logistique multinomiale est utilisée lorsque la variable cible possède plus de deux classes. Elle repose sur le principe du *softmax*, qui convertit les sorties d'un modèle en probabilités associées à chaque classe.

#### 3.1.1 Softmax

Pour un ensemble de  $K$  classes, la probabilité  $P(y_i = k \mid x_i, W)$  d'appartenir à la classe  $k$  est donnée par :

$$P(y_i = k \mid x_i, W) = \frac{\exp(w_k^\top x_i)}{\sum_{j=1}^K \exp(w_j^\top x_i)},$$

où :

- $x_i$  est le vecteur des caractéristiques de l'observation  $i$ ,
- $W = [w_1, \dots, w_K]$  représente les coefficients du modèle pour chaque classe.

Le modèle prédit la classe avec la probabilité la plus élevée.

### 3.1.2 Fonction de coût

La fonction de coût utilisée est l'entropie croisée catégorielle, exprimée par :

$$J(W) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{ik} \log P(y_i = k \mid x_i, W),$$

où  $y_{ik}$  est une variable indicatrice valant 1 si  $y_i = k$  et 0 sinon.

### 3.1.3 Optimisation par descente de gradient

La descente de gradient ajuste les poids  $W$  pour minimiser  $J(W)$ . À chaque itération :

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W},$$

où  $\eta$  est le taux d'apprentissage.

## 3.2 Adam Optimizer

L'optimiseur Adam (*Adaptive Moment Estimation*) est utilisé pour accélérer la convergence. Il combine les avantages des méthodes de gradient à taux d'apprentissage adaptatif et de momentum. Les paramètres  $m_t$  (moment) et  $v_t$  (variance) sont mis à jour comme suit :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$

où  $g_t$  est le gradient actuel, et  $\beta_1, \beta_2$  sont des hyperparamètres.

Les poids sont mis à jour en utilisant :

$$W \leftarrow W - \frac{\eta m_t}{\sqrt{v_t} + \epsilon},$$

avec  $\epsilon$  comme valeur de lissage pour éviter les divisions par zéro.

## 3.3 Learning Rate

Nous avons instauré la possibilité de choisir le taux d'apprentissage mais aussi son évolution dans le temps. En effet,

**Diminution avec décroissance par étape :**

$$\eta_t = \eta_0 \cdot \frac{1}{1 + \lambda t},$$

où  $\lambda$  est un hyperparamètre déterminant la vitesse de diminution.

## 3.4 Impact sur l'Entraînement

Le choix de  $\eta$ , du *LR decay* et du *warm-up* impacte directement :

- **La vitesse de convergence** : un bon choix de stratégie peut accélérer l'apprentissage.
- **La stabilité** : éviter les oscillations ou la divergence.
- **La généralisation** : un taux mal configuré peut entraîner un surapprentissage ou un sous-apprentissage.

### 3.5 Implémentation dans le Package

1. **Initialisation des poids** : Les poids sont initialisés aléatoirement pour toutes les classes.
2. **Prédictions avec Softmax** : Le calcul des probabilités pour chaque classe est effectué et la classe est attribué à la plus grande.
3. **Calcul de la perte** : La fonction de coût est évaluée pour ajuster les poids.
4. **Mise à jour des poids** : Les gradients sont calculés et les poids sont mis à jour à chaque itération et/ou chaque batch.

### 3.6 Évaluation du Modèle

Les performances du modèle sont mesurées à l'aide de métriques telles que :

- **La précision** : proportion de prédictions correctes.
- **La matrice de confusion** : évalue les erreurs entre classes.
- **Les courbes de perte et de taux d'apprentissage** : montrent la convergence du modèle.

## 4 Classes de Prétraitement

Plusieurs classes R6 sont implémentées pour le prétraitement des données, facilitant la préparation des données avant l'entraînement du modèle :

- **OneHotEncoder** : Encode les variables qualitatives en utilisant l'encodage One-Hot.
- **StandardScaler** : Normalise les variables quantitatives en leur appliquant une standardisation.
- **MinMaxScaler** : Met à l'échelle les variables en les ramenant dans une plage spécifiée.
- **RobustScaler** : Applique une mise à l'échelle robuste en utilisant des statistiques résilientes aux valeurs aberrantes.
- **SimpleQualiImputer** et **SimpleQuantiImputer** : Imputent les valeurs manquantes dans les données qualitatives et quantitatives respectivement.

Ces classes sont organisées dans le dossier `R/preprocessing/` et sont intégrées dans le flux de travail via les modules Shiny correspondants.

## 5 Visualisation avec Shiny

L'application Shiny offre une interface utilisateur interactive permettant de gérer l'ensemble du processus de modélisation, de la sélection des données au prétraitement, puis à l'entraînement et à l'évaluation du modèle. Les principales fonctionnalités incluent :

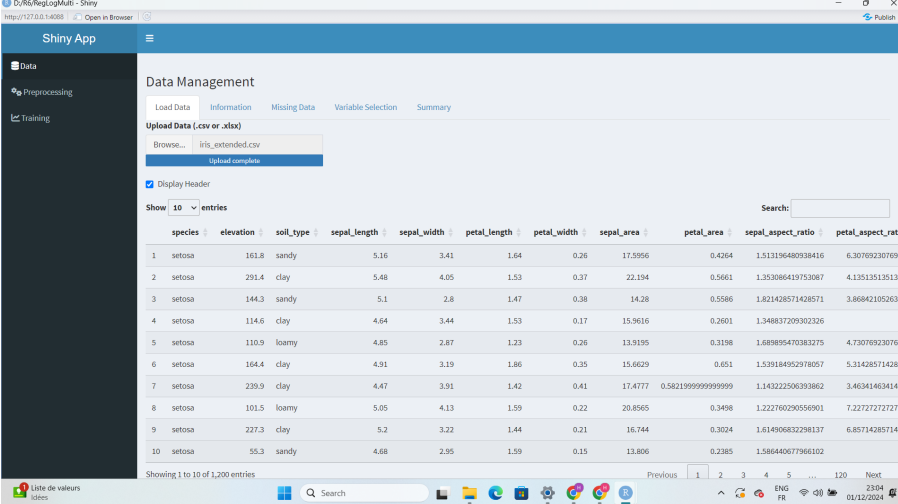
- **Gestion des Données** : Importation et visualisation des jeux de données, sélection des variables cibles et explicatives.
- **Prétraitement** : Application des techniques de prétraitement via des boutons d'action, visualisation des données transformées.
- **Entraînement du Modèle** : Configuration des paramètres du modèle, lancement de l'entraînement avec une barre de progression, affichage des résultats tels que la précision et la matrice de confusion.

- **Visualisation des Performances** : Graphiques illustrant la courbe de perte et l'évolution du taux d'apprentissage au cours de l'entraînement.

L'intégration de Shiny permet aux utilisateurs de configurer et d'affiner facilement les modèles sans nécessiter de compétences avancées en programmation.

## 5.1 Data

Cette section est dédiée à la gestion des données et propose plusieurs fonctionnalités clés :



The screenshot shows the 'Data Management' section of a Shiny application. It includes tabs for 'Load Data', 'Information', 'Missing Data', 'Variable Selection', and 'Summary'. The 'Load Data' tab is active, showing an 'Upload Data (.csv or .xlsx)' section with a 'Browse...' button and an 'Upload complete' button. Below this, there is a 'Display Header' checkbox and a 'Show 10 entries' dropdown. The main area displays a table with 10 rows of data. The table has columns for species, elevation, soil\_type, sepal\_length, sepal\_width, petal\_length, petal\_width, sepal\_area, petal\_area, sepal\_aspect\_ratio, and petal\_aspect\_ratio. The data is as follows:

|    | species | elevation | soil_type | sepal_length | sepal_width | petal_length | petal_width | sepal_area | petal_area         | sepal_aspect_ratio | petal_aspect_ratio |
|----|---------|-----------|-----------|--------------|-------------|--------------|-------------|------------|--------------------|--------------------|--------------------|
| 1  | setosa  | 161.8     | sandy     | 5.16         | 3.41        | 1.64         | 0.26        | 17.5956    | 0.4264             | 1.513196480938416  | 6.30709230709      |
| 2  | setosa  | 291.4     | clay      | 5.48         | 4.05        | 1.53         | 0.37        | 22.194     | 0.5661             | 1.353086419753087  | 4.13513513513      |
| 3  | setosa  | 144.3     | sandy     | 5.1          | 2.8         | 1.47         | 0.38        | 14.28      | 0.5586             | 1.821428571428571  | 3.66842105263      |
| 4  | setosa  | 114.6     | clay      | 4.64         | 3.44        | 1.53         | 0.17        | 15.9616    | 0.2601             | 1.348837209302326  |                    |
| 5  | setosa  | 110.9     | loamy     | 4.85         | 2.87        | 1.23         | 0.26        | 13.9195    | 0.3198             | 1.689895470383275  | 4.73070923070      |
| 6  | setosa  | 164.4     | clay      | 4.91         | 3.19        | 1.86         | 0.35        | 15.6629    | 0.651              | 1.539184952978057  | 5.31428571428      |
| 7  | setosa  | 239.9     | clay      | 4.47         | 3.91        | 1.42         | 0.41        | 17.4777    | 0.5821999999999999 | 1.143222506393862  | 3.46341463414      |
| 8  | setosa  | 101.5     | loamy     | 5.05         | 4.13        | 1.59         | 0.22        | 20.8565    | 0.3498             | 1.222760290556901  | 7.22727272727      |
| 9  | setosa  | 227.3     | clay      | 5.2          | 3.22        | 1.44         | 0.21        | 16.744     | 0.3024             | 1.61490683228137   | 6.85714285714      |
| 10 | setosa  | 55.3      | sandy     | 4.68         | 2.95        | 1.59         | 0.15        | 13.806     | 0.2385             | 1.588440077966102  |                    |

FIGURE 1 – Data management

- **Importation des données** : Permet de charger des fichiers au format .csv ou .xlsx dans l'application. Une fois le fichier chargé, les données sont affichées sous forme de tableau interactif.
- **Informations** : Fournit des détails sur les données, y compris les types de variables (qualitatives ou quantitatives), le nombre total d'observations, et le total des valeurs manquantes dans l'ensemble des données.
- **Missing Data** : Offre une interface pour gérer les valeurs manquantes. Les utilisateurs peuvent ajouter ou imputer des données manquantes, que ce soit pour des variables qualitatives ou quantitatives.
- **Summary** : Présente un résumé statistique des données, notamment des mesures comme la moyenne, l'écart-type, les minimums et maximums, et les quantiles.

## 5.2 Preprocessing

Cette section est consacrée à la préparation des données avant leur utilisation pour l'entraînement des modèles. Les options disponibles sont :

- **One-Hot Encoding** : Applique une transformation pour convertir les variables qualitatives en une représentation numérique, facilitant leur utilisation dans les modèles.
- **Scalers pour variables quantitatives** : Trois types de normalisations/scalings sont proposés :
  - **Standard Scaler** : Standardisation des données pour qu'elles aient une moyenne de 0 et un écart-type de 1.

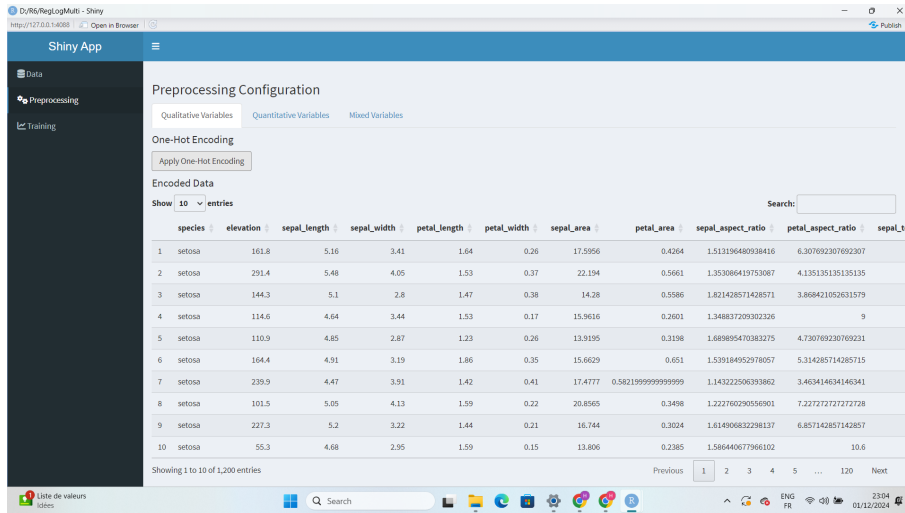


FIGURE 2 – Preprocessing

- **Min-Max Scaler** : Mise à l'échelle des données dans une plage spécifiée (souvent  $[0, 1]$ ).
- **Robust Scaler** : Scaling robuste aux valeurs aberrantes en utilisant les médianes et les quartiles.
- **FAMD** : Utilisé pour gérer les variables mixtes (qualitatives et quantitatives) en les intégrant dans une même représentation.

### 5.3 Training

Cette section permet de configurer et de suivre l'entraînement du modèle d'apprentissage automatique :

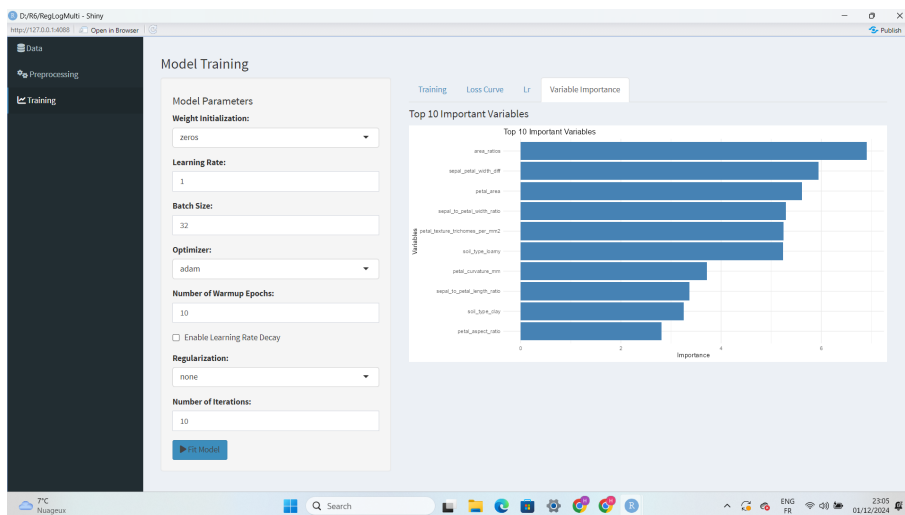


FIGURE 3 – Training

- **Configuration des hyperparamètres** : Permet de paramétrer le modèle, notamment la méthode d'initialisation des poids, le taux d'apprentissage, la taille des lots (*batch size*), le nombre d'itérations, et le choix de l'optimiseur (ex. : Adam).

- **Résumé du modèle et progression de l'entraînement** : Affiche un récapitulatif des paramètres du modèle ainsi que la progression de l'entraînement en temps réel.
- **Loss Curve** : Graphique montrant l'évolution de la fonction de perte au fil des itérations, permettant de vérifier la convergence du modèle.
- **Learning Rate Visualization** : Permet de visualiser l'évolution du taux d'apprentissage au cours de l'entraînement.
- **Importance des variables** : Met en évidence l'importance relative des différentes variables dans le modèle final, offrant des indications sur leur impact dans les prédictions.

Cette interface offre une gestion complète du pipeline de données, allant de leur importation à l'entraînement des modèles, tout en assurant un suivi visuel pour faciliter l'interprétation des résultats.

## 6 Comparaison avec sklearn

Cette partie compare l'implémentation d'un modèle de régression logistique multinomiale en R avec Scikit-learn, un framework pour le Machine Learning sous Python. Les aspects évalués incluent : *performances, scalabilité, efficacité, et prétraitement*. Le modèle R est basé sur une classe R6, tandis que Scikit-learn utilise un moteur optimisé pour les calculs matriciels.

### Analyse des performances

#### Temps de calcul et efficacité

| Critère                     | Modèle R  | Scikit-learn   |
|-----------------------------|---|--|
| Optimisation algébrique     | Utilise les bibliothèques <code>Matrix</code> et <code>matrixStats</code> . Bonnes performances sur des données moyennes. | Basé sur <code>numpy</code> et <code>scipy</code> , avec des implémentations optimisées en Cython. |
| Gestion des grandes données | Limité par la RAM, ralentissement notable sur des millions de lignes.   | Plus efficace grâce à des optimisations internes.  |
| Gestion des mini-batches    | Implémenté explicitement dans notre modèle.   | Géré automatiquement par les algorithmes.  |
| Initialisation              | Supporte <code>xavier</code> et <code>zeros</code> .  | Initialisations automatiques optimisées (ex : <code>He initialization</code> ).                    |
| Temps d'entraînement        | Plus lent pour des données volumineuses et des itérations élevées.  | Rapide grâce à des optimisations et parallélisation.   |

#### Temps d'entraînement estimé

| Taille des données             | Modèle R                                       | Scikit-learn                       |
|--------------------------------|--|------------------------------------|
| Petites données (<10k lignes)  | Quelques secondes.                             | Quelques millisecondes à secondes. |
| Moyennes données (100k lignes) | Quelques minutes.                              | Quelques secondes.                 |
| Grandes données (>1M lignes)   | Limité par la RAM, plusieurs minutes à heures. | Quelques minutes (RAM suffisante). |

## Prétraitement des données

L'implémentation inclut un pipeline complet de prétraitement des données (One-Hot Encoding, scaling, AFDM). Scikit-learn propose des pipelines similaires mais optimisés pour les données volumineuses ainsi que pour la parallélisation des calculs.

## Comparaison des prétraitements

| Prétraitement                                | Modèle R   | Scikit-learn  |
|--|--|---|
| One-Hot Encoding                             | Réalisé via une classe dédiée ( <code>One.Hot.Encoder</code> ). Bonnes performances.       | Géré automatiquement avec <code>OneHotEncoder</code> . Plus rapide.                         |
| Normalisation / Standardisation              | Scalers implémentés ( <code>Standard</code> , <code>MinMax</code> , <code>Robust</code> ). | Scalers natifs optimisés (ex : <code>StandardScaler</code> ).                               |
| AFDM (Analyse Factorielle de Données Mixtes) | Implémenté pour les variables mixtes via <code>FactoMineR</code> .                         | Pas directement supporté, nécessite des transformations manuelles ou via d'autres packages. |

## Optimisation et scalabilité

### Modèle R

- Optimisation du SGD via Adam.
- Approche par mini-batches par défaut.
- Limité par la RAM et la vitesse d'exécution des boucles R.

### Scikit-learn

- Intègre des algorithmes optimisés pour des données volumineuses.
- Supporte la parallélisation et l'exécution sur GPU via des bibliothèques tierces.

## 7 Conclusion

L'implémentation en R est robuste et bien adaptée aux jeux de données de taille petite à moyenne. Cependant, Scikit-learn surpasse en termes de performance et d'efficacité



pour des données volumineuses grâce à ses optimisations natives. Toutefois, notre modèle R possède l'atout non négligeable de ne pas être une boîte noire en son cœur notamment en raison de son caractère paramétrable et modulable. De plus, on peut noter la possibilité de faire varier le taux d'apprentissage, un paramètre crucial lorsque l'on fait du Machine Learning. En sommes, il s'agit d'un package permettant de faire de la classification multi-classe par le biais d'une régression logistique multinomiale. Que les variables explicatives soient quantitatives ou qualitatives, de nombreuses fonctions de prétraitements sont applicables afin de préparer les données à la modélisation. La faiblesse de ce package réside donc dans sa difficulté à gérer de grands volumes de données, ce qui pourrait faire l'objet de travaux futurs, mais sa force réside donc dans la diversité des prétraitements possibles ainsi que sa relative autonomie par rapport aux autres