# V. Réseaux de neurones

## A. Présentation du procédé

Le réseau de neurones artificiels a été inspiré du mécanisme du système nerveux humain, à travers l’histoire on a cherché à imiter son fonctionnement en commençant par le neurone formel apparu en 1943 avec une modélisation mathématique du neurone biologique, puis par les premières règles d’apprentissage de Hebb en 1949 et de l’algorithme d’apprentissage de rétropropagation en 1986 par Rumelhart.

Cet algorithme est utilisé dans le cadre des problématiques de classification, de reconnaissance de formes, d’association, d’extractions de caractéristiques, d’identification…

En assurance, l’avantage de cet algorithme est de pou voir capter les dépendances non linéaires entre les variables explicatives pour pouvoir affiner le tarif et appréhender le risque.

Pour les parties qui suivent, le paramètre le plus important qui serait à déterminer est le nombre de neurones sur la couche cachée parallèlement aux conditions d’apprentissage que nous citerons par la suite.

Pour le calibrage des deux modèles Fréquence et Coût Moyen, nous avons utilisé plusieurs fonctions intégrées sous R:

* **Tune.nnet** du package e1070 permet par validation croisée de calibrer et d'optimiser simultanément le nombre de neurones sur la couche cachée **(size)** et le paramètre de régularisation **(decay)** (ou le paramètre de décomposition) qui permet d’éviter le sur-ajustement du modèle, le temps d'exécution de cette fonction est très long nous avons donc réduit notre échantillon d'apprentissage en prenant 6 839 polices, nous avons validé cet échantillon par le test sur la conservation de la moyenne et de la variance par rapport à l’échantillon d’apprentissage.
* **Nnet** du package Neuralnet permet d'entraîner un modèle de perceptron multicouche en ajustant notre réseau de neurones avec les paramètres optimaux précédemment trouvés avec la fonction Tune.nnet.

## Modèle de coût

Dans cette partie, nous allons calibrer un modèle coût moyen, pour commencer nous allons par le biais du package Caret procéder par une régression sur notre échantillon réduit représenté par les 6 839 polices afin de déterminer nos paramètres optimaux (le Size et le Decay).

Grâce à cette fonction, nous obtenons nos deux paramètres optimaux: **Size optimal =4** et **Decay optimal= 2**, cette méthode se base sur le RMSE comme critère pour sélectionner ces deux paramètres optimaux en utilisant la plus petite valeur. *(Annexe 1 : RMSE Cross Validation)*

L’optimisation des paramètres a nécessité encore le passage par la validation croisée, la fonction tune.nnet est adaptée à cette démarche, cette méthode nous a suggérée d’utiliser un **Size optimal =7** et **Decay optimal= 3** mais nous avons retenu les deux premiers paramètres car nous avons obtenu de meilleurs résultats*. (Annexe 2 Schéma représentatif coût)*

Nous avons pensé à faire varier le nombre total d’itérations mais cela nous a pris beaucoup de temps surtout que chaque exécution donnait des résultats différents, en plus du temps d’exécutions assez long de la fonction nnet.

Nous nous sommes limités à un nombre d’itérations égales à 500 et nous avons réalisé la calibration du modèle coût moyen sur la totalité de l’échantillon d’apprentissage, nous avons effectué par la suite notre prévision sur l’échantillon test assortie des erreurs estimées MSE et RMSE :

|  |  |
| --- | --- |
| Indicateurs d’écart | Value |
| MSE | 3 161 619 |
| RMSE | 1 778.094 |

|  |  |  |  |
| --- | --- | --- | --- |
| Echantillons | Echantillon Apprentissage | Echantillon Test | Prédiction |
| Coût moyen | 1 434 | 1 452 | 1 434.6 |

Nous pouvons remarquer que le coût moyen prédit est proche de celui ajusté, nous comparerons par la suite avec les résultats obtenus des autres modèles pour choisir celui qui minimise les deux critères : MSE et RMSE.

## B. Modèle Fréquence

Dans cette partie nous avons choisi de modéliser le nombre de sinistres, et nous avons utilisé l’exposition comme pondération tout en considérant les polices avec une exposition inférieure ou égale à 1.

Nous obtenons nos paramètres optimaux grâce à la fonction Caret appliquée à notre échantillon réduit : **Size optimal = 2** et **Decay optimal= 1.** Etant donné que la validation croisée est très chronophage et comme les paramètres optimaux déjà obtenus sont satisfaisants nous nous sommes contentés des paramètres optimaux que nous avons eus grâce à la fonction caret, nous avons essayé de tracer la performance of nnet sur un tout petit échantillon et en réduisant le nombre d’itérations à 100. *(Annexe 3 : RMSE Cross Validation Fréquence, Annexe 5 : Performance of nnet Frequency)*

Nous construisons par la suite notre modèle fréquence intégrant toutes les 8 variables explicatives avec les 2 neurones dans la couche cachée du réseau. *(Schéma représentatif fréquence Annexe 4)*

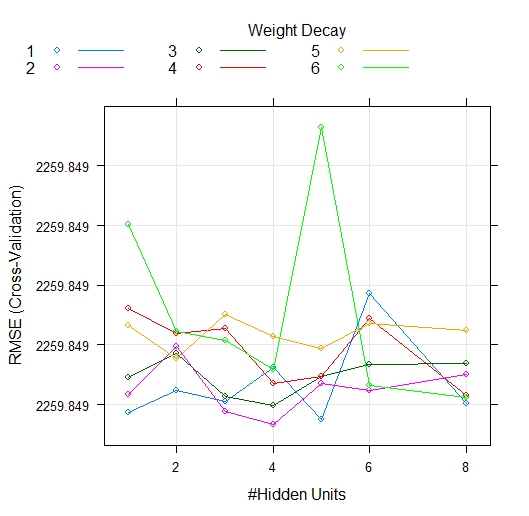
La calibration de ce modèle a été réalisée sur la totalité de l’échantillon apprentissage, nous présentons ci- dessous nos résultats de prévision avec les erreurs estimées.

|  |  |  |  |
| --- | --- | --- | --- |
| Echantillons | Echantillon Apprentissage | Echantillon Test | Prédiction |
| Fréquence | 0.05283 | 0.05322 | 0.06564 |

|  |  |
| --- | --- |
| Indicateurs d’écart | Value |
| MSE | 0.05876905 |
| RMSE | 0.2424233 |

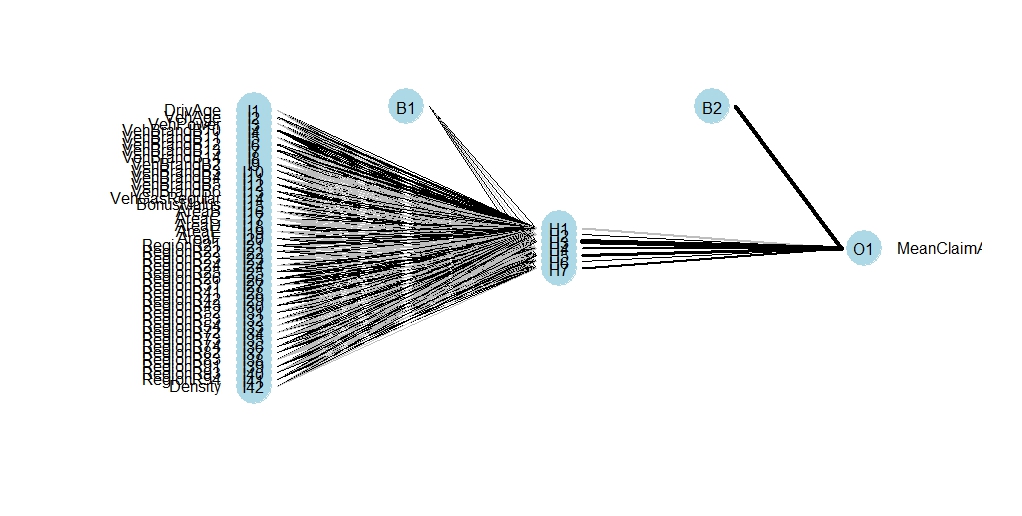
# ANNEXES

### Annexe 1 : RMSE CROSS VALIDATION COUT

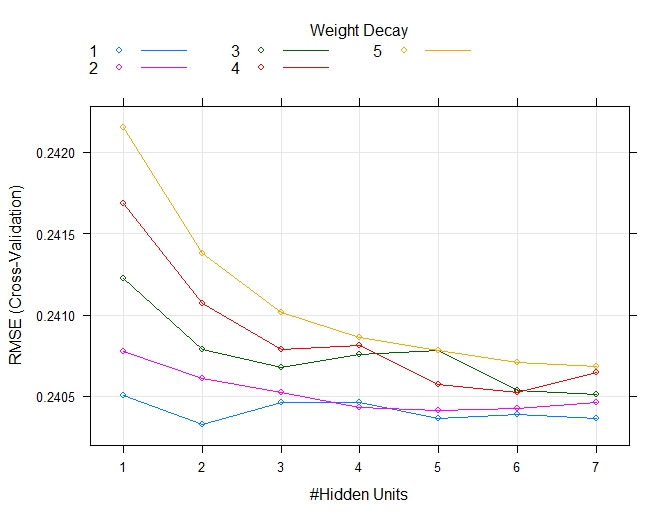


### Annexe 2 : Schéma représentatif du modèle construit pour le coût

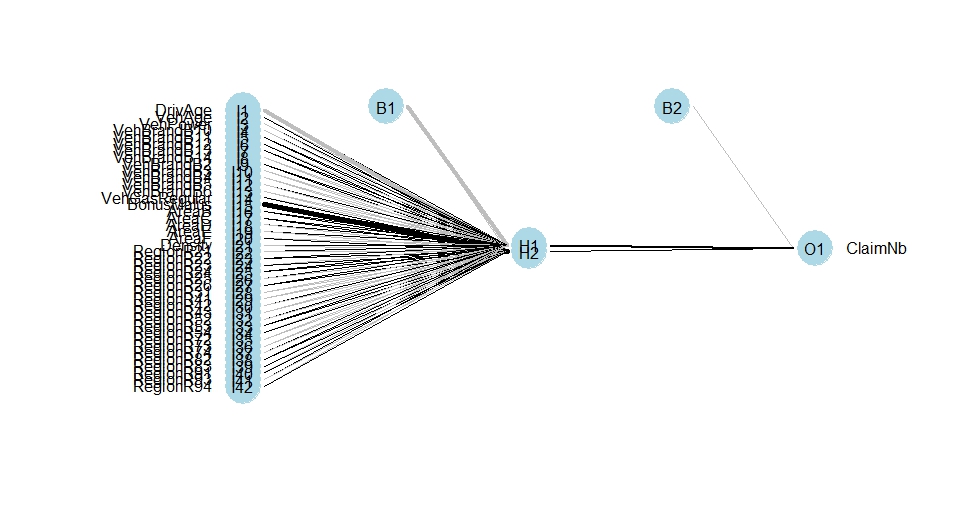
### 



### Annexe 3 : RMSE CROSS VALIDATION FREQUENCE



### Annexe 4 : Schéma représentatif du modèle construit pour la fréquence



### Annexe 5 : Performance of Nnet (Fréquence)

