Le présent document porte sur l'utilisation du paquetage `keras` pour la création d'un réseau de neurones à propagation directe de trois couches cachées. On construit le réseau sur les données `freMTPLfreq` du paquetage `CASdatasets`.

1 Présentation

Le paquetage `R`, `keras`, est une interface au paquetage `Keras` développé en `Python`. Cela permet d'utiliser toutes les fonctionnalités de `Keras` tout en utilisant `R`.

`Keras` est une interface de programmation applicative (« API, application programming interface ») qui permet d'utiliser plusieurs bibliotèques d'apprentissage machine comme `TensorFlow`, `Microsoft Cognitive Toolkit`, `Theano` ou `PlaidML`. Cette interface permet d'implémenter rapidement des réseaux de neurones profonds avec des architectures de toutes sortes. Elle permet aussi de traiter le même code sur CPU ou sur GPU.

L'avantage d'utiliser `keras` avec `R` est d'avoir la possibilité de créer toutes sortes de modèles d'apprentissage profond avec une relative facilité pour débuter. L'inconvénient est de devoir télécharcher une distribution `Python`.

2 Installation

Par défaut, `keras` utilise la bibliotèque `TensorFlow`. La fonction `install_keras()` permet de configurer `Keras` et `TensorFlow` en même temps pour une utilisation sur CPU. Il est possible de les configurer pour une utilisation sur GPU en changeant les paramètres de la fonction `install_keras()`.

```
install.packages("keras")

library(keras)
install_keras()
```

3 Pré-traitement des données

```

library(CASdatasets)
library(tidyverse)
data(freMTPLfreq)

dat <- freMTPLfreq %>%
```

```
  as_tibble() %>%
    mutate_at(vars(Gas, Brand, Region), factor) %>%
    mutate_at(vars(Power),as.integer) %>%

    mutate(Exposure = if_else(Exposure > 1, 1, Exposure))%>%
    mutate(DriverAge= ifelse(DriverAge > 85,85,DriverAge)) %>%
  mutate(CarAge = ifelse(CarAge > 20,20,CarAge))

# Création échantillons d'entrainement, de test et de
    validatiion

set.seed(100)
ll <- sample(which(dat$ClaimNb==0), round(0.8*length(which(dat
    $ClaimNb==0))), replace = FALSE)
ll <- c(ll,sample(which(dat$ClaimNb==1), round(0.8*length(
    which(dat$ClaimNb==1))), replace = FALSE))
ll <- c(ll,sample(which(dat$ClaimNb==2), round(0.8*length(
    which(dat$ClaimNb==2))), replace = FALSE))
ll <- c(ll,sample(which(dat$ClaimNb==3), round(0.8*length(
    which(dat$ClaimNb==3))), replace = FALSE))
ll <- c(ll,sample(which(dat$ClaimNb==4), round(0.8*length(
    which(dat$ClaimNb==4))), replace = FALSE))


learn <- dat[ll,]
testNN <- dat[-ll,]

set.seed(200)
ll2 <- sample(which(learn$ClaimNb==0), round(0.75*length(which
    (learn$ClaimNb==0))), replace = FALSE)
ll2 <- c(ll2,sample(which(learn$ClaimNb==1), round(0.75*length
    (which(learn$ClaimNb==1))), replace = FALSE))
ll2 <- c(ll2,sample(which(learn$ClaimNb==2), round(0.75*length
    (which(learn$ClaimNb==2))), replace = FALSE))
ll2 <- c(ll2,sample(which(learn$ClaimNb==3), round(0.75*length
    (which(learn$ClaimNb==3))), replace = FALSE))
ll2 <- c(ll2,sample(which(learn$ClaimNb==4), round(0.75*length
    (which(learn$ClaimNb==4))), replace = FALSE))

learnNN <- learn[ll2,]
valNN <- learn[-ll2,]
```