18780 - Luis Martin Castillo Uvalle
20852 - Gerardo Valencia Rodríguez

El programa se divide en varias secciones que abarcan desde la carga de datos hasta la evaluación de un modelo de red neuronal utilizando k-fold cross validation.

Carga de datos:
La primera sección se encarga de cargar los datos desde un archivo csv utilizando la función read.csv de R. El archivo contiene información nutricional de diferentes cereales, incluyendo el nombre del cereal, su rating, y su contenido calórico, de proteínas, grasas, sodio y fibra. Estos datos se almacenan en un objeto llamado data.

Selección de datos de entrenamiento y prueba:
En la siguiente sección, se seleccionan aleatoriamente los índices de los datos de entrenamiento y prueba utilizando la función sample. Se utiliza el 60% de los datos para entrenamiento y el 40% restante para pruebas. Estos índices se almacenan en un objeto llamado index, que se utiliza más adelante para dividir los datos en conjuntos de entrenamiento y prueba.

Normalización de datos:
En la siguiente sección, los datos se normalizaron utilizando la función scale. La normalización es importante porque algunos algoritmos de aprendizaje automático requieren que todas las variables tengan la misma escala. En este caso, se utiliza la escala min-max para normalizar los datos. Se almacenan los valores máximos y mínimos de cada columna en objetos llamados max y min, respectivamente. Luego, se utiliza la función scale para normalizar los datos utilizando estos valores y se almacenan en un objeto llamado scaled.

Entrenamiento de una red neuronal:
En la siguiente sección, se entrena un modelo de red neuronal utilizando la función neuralnet de la biblioteca neuralnet. El modelo se entrena utilizando los datos de entrenamiento y la fórmula rating ~ calories + protein + fat + sodium + fiber. El modelo tiene tres capas ocultas y utiliza una salida lineal. El modelo se almacena en un objeto llamado NN.

Predicción con el modelo de red neuronal:
En la siguiente sección, se utiliza el modelo de red neuronal entrenado para hacer predicciones sobre los datos de prueba. Las predicciones se obtienen utilizando la función compute y se almacenan en un objeto llamado predict_testNN. Luego, se convierten de nuevo a la escala original utilizando los valores máximos y mínimos de la columna rating y se almacenan en el mismo objeto predict_testNN.

Evaluación del modelo de red neuronal:

En la siguiente sección, se evalúa la calidad del modelo de red neuronal utilizando el error cuadrático medio (RMSE) sobre los datos de prueba. Se calcula el RMSE utilizando la fórmula (sum((datatest$rating - predict_testNN)^2) / nrow(datatest)) ^ 0.5 y se almacena en un objeto llamado RMSE.NN.

Validación cruzada del modelo de red neuronal:
Finalmente, se calcula la mediana de los RMSE obtenidos en cada una de las 56 iteraciones del modelo de red neuronal con la validación cruzada. Para hacer esto, se utiliza la función colMedians() del paquete matrixStats.

Se instala y carga el paquete matrixStats. Luego, se calcula la mediana de los RMSE obtenidos en cada iteración para cada tamaño de conjunto de entrenamiento. Finalmente, se grafica la mediana del RMSE en función del tamaño del conjunto de entrenamiento, lo que permite visualizar cómo el rendimiento del modelo varía en función del tamaño del conjunto de datos de entrenamiento.

La gráfica resultante muestra que el RMSE disminuye al aumentar el tamaño del conjunto de entrenamiento, pero la tasa de disminución se estabiliza a partir de un tamaño de muestra de alrededor de 45-50. Esto indica que, para este conjunto de datos y este modelo de red neuronal en particular, no es necesario utilizar un tamaño de muestra mayor a 50 para obtener un buen rendimiento del modelo.