Trabajando con WEKA

Daniel Marín López

Índice

1. Aunque WEKA soporta conjuntos de datos en diferentes tipos de archivo, es recomendable trabajar con el formato ARFF, el cuál podemos construir fácilmente desde un CSV.

Primero, abrimos la herramienta Explorer de WEKA, una interfaz intuitiva diseñada para la manipulación y análisis de datos. Una vez dentro, nos dirigimos a la sección **Open File**, donde seleccionaremos el archivo CSV que contiene nuestros datos. Este archivo debe estar previamente preparado con las columnas y filas que representan las variables y los casos que vamos a analizar. A continuación, hacemos clic en la opción **"Invoke options dialog"**, lo que nos permite acceder a un cuadro de diálogo donde podemos ajustar configuraciones específicas para la correcta importación de nuestro archivo. En este paso, es crucial prestar atención a los detalles del formato del CSV, como el delimitador utilizado. Cambiamos el separador predeterminado por **";"** para asegurarnos de que WEKA lea correctamente las columnas de datos. Es importante verificar que otras configuraciones, como el tipo de codificación o la presencia de encabezados, también sean compatibles con nuestro archivo. Una vez realizados estos ajustes, procedemos a cargar los datos para su análisis.

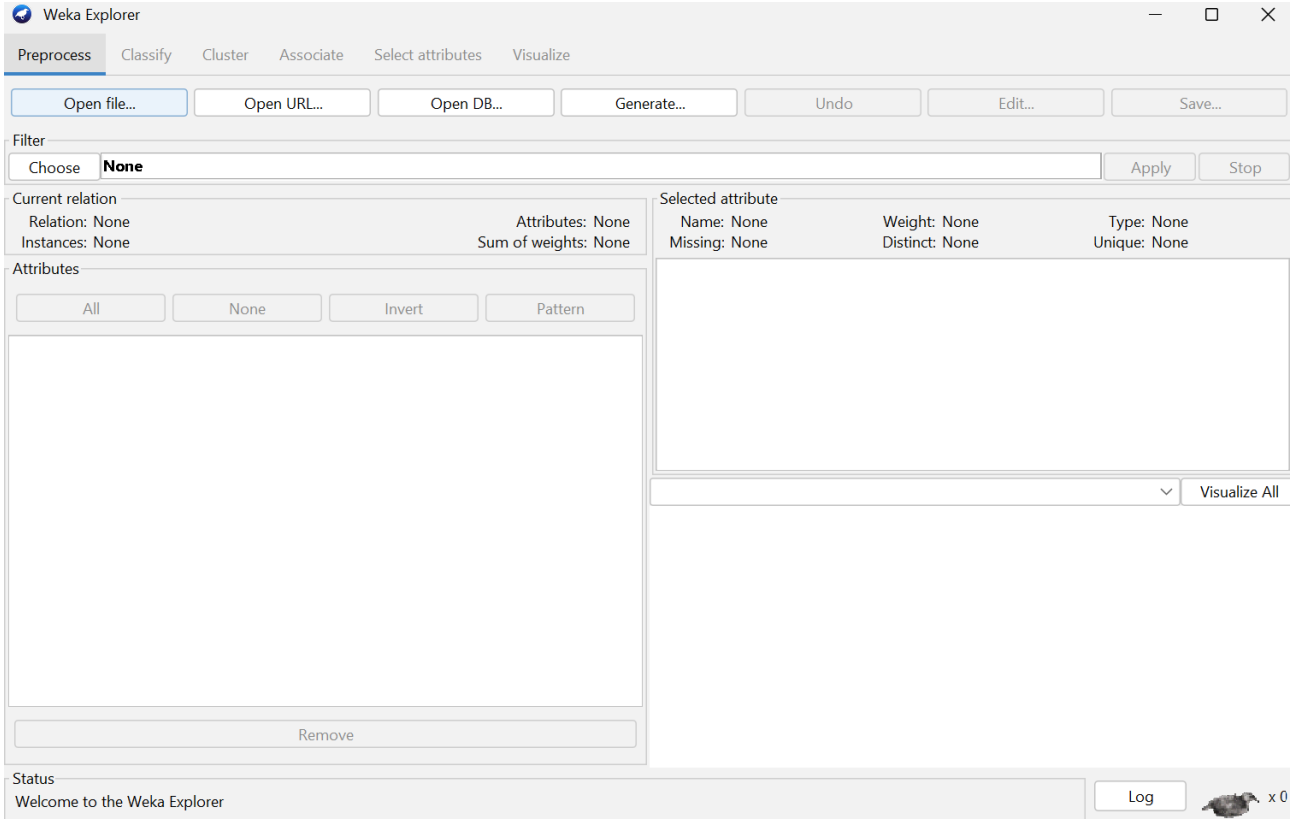

Figura 1: Herramienta Explorer de WEKA

Experimentando con nuestros datos en WEKA

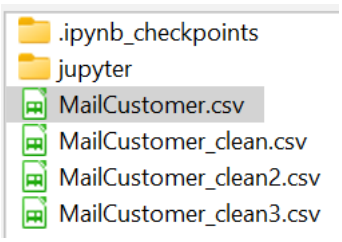

Figura 3: Importando csv

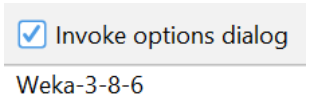

Figura 2: Opción a la derecha del explorer

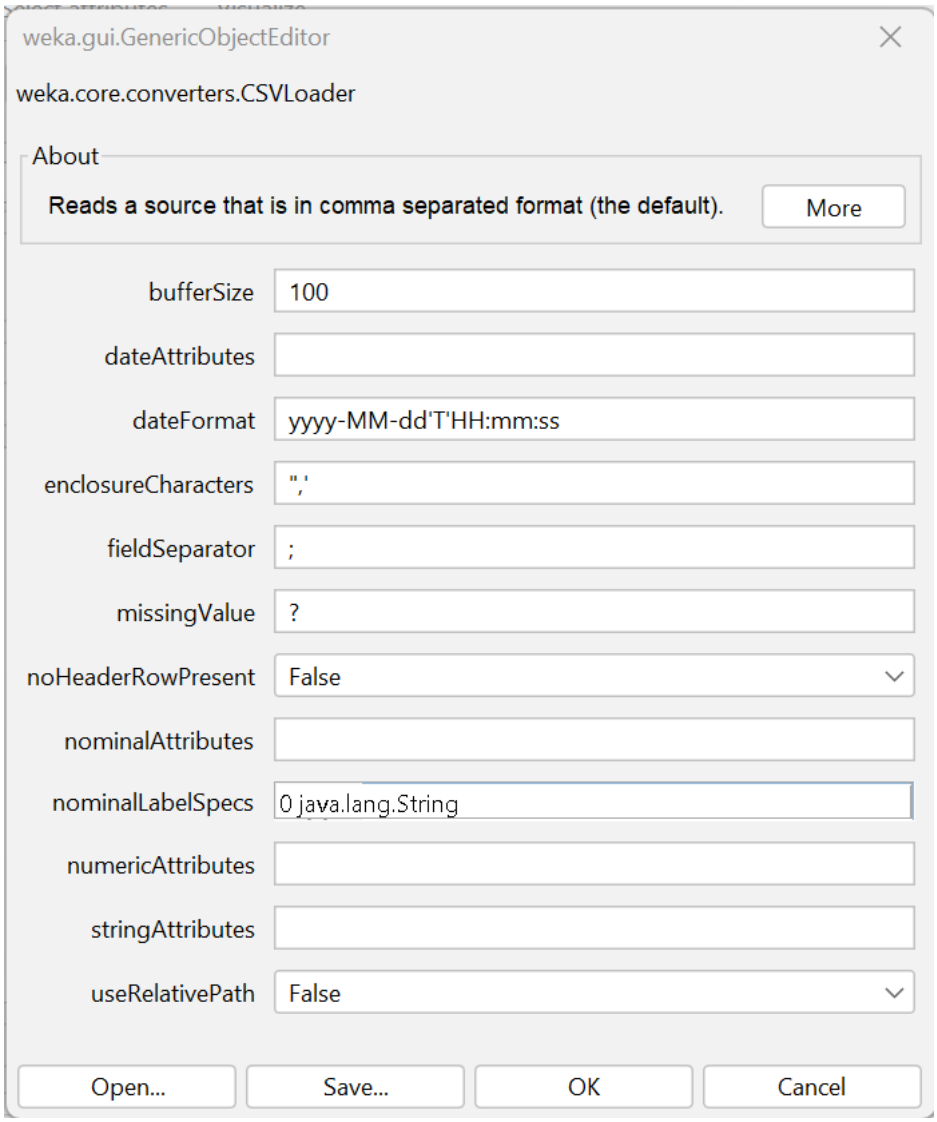

Figura 4: Opciones del CSV

Experimentando con nuestros datos en WEKA

El resultado se muestra de la siguiente forma:

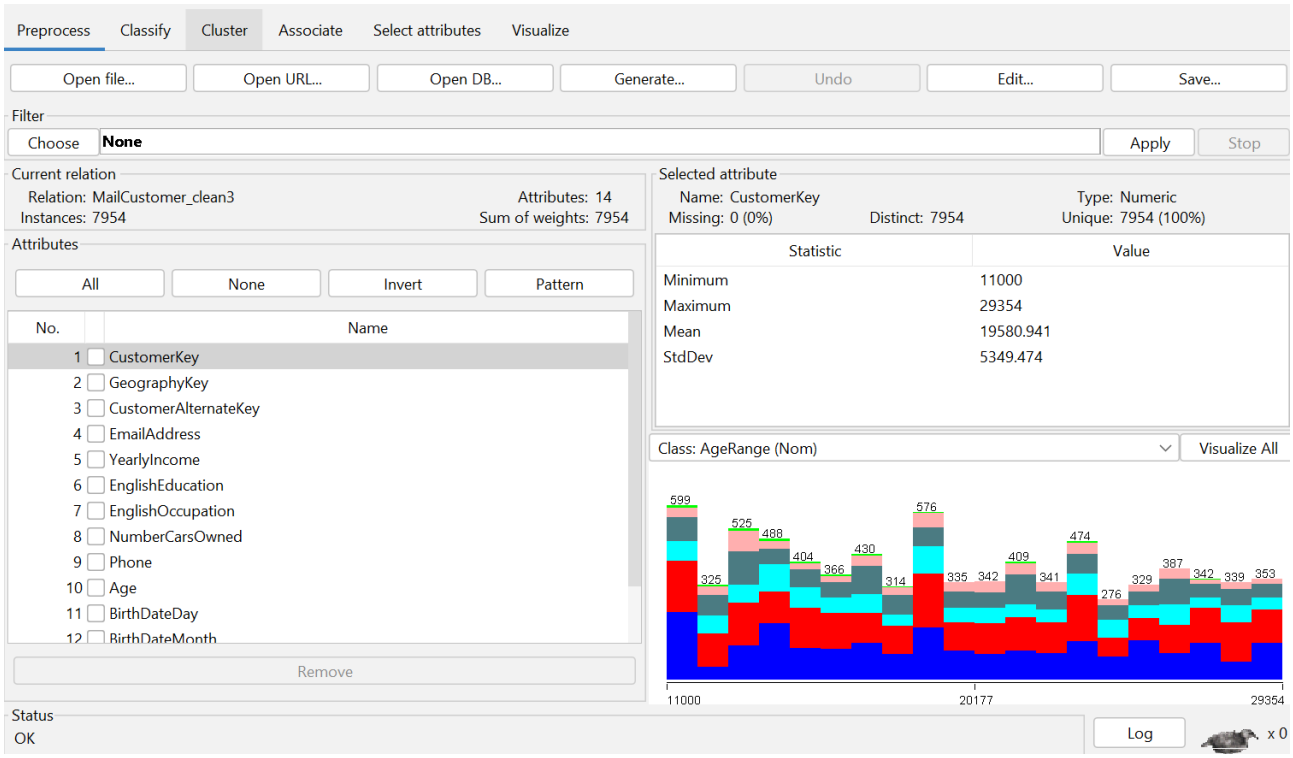

Figura 5: CSV importado con WEKA

Podemos ver a la izquierda el n.º de características que tiene nuestro dataset y a la derecha los datos agrupados en base a una característica concreta. Normalmente es la primera del dataset cuando se cargan los datos

Para guardarlo en ARFF le damos a la opción ***Save***.

Figura 6: Guardando el archivo ARFF

2. Exploraremos nuestros datos con las opciones de visualización.

Para visualizar los datos, seleccionamos la opción **Visualizer**, que nos permite explorar y analizar la información de manera más comprensible. Dentro de esta herramienta, podemos representar los datos gráficamente en función de las diferentes características o variables que los componen, lo cual facilita identificar patrones, tendencias y relaciones significativas. Además, esta visualización puede adaptarse a diferentes formatos, como gráficos de dispersión, histogramas o diagramas de líneas, dependiendo del tipo de datos y los objetivos del análisis. De este modo, **Visualizer** se convierte en una herramienta clave para interpretar la información de manera intuitiva y efectiva, mejorando nuestra capacidad de tomar decisiones basadas en los datos.

Figura 7: Opción Visualizer

Experimentando con nuestros datos en WEKA

Si deseamos ver una gráfica concreta hacemos doble click en una

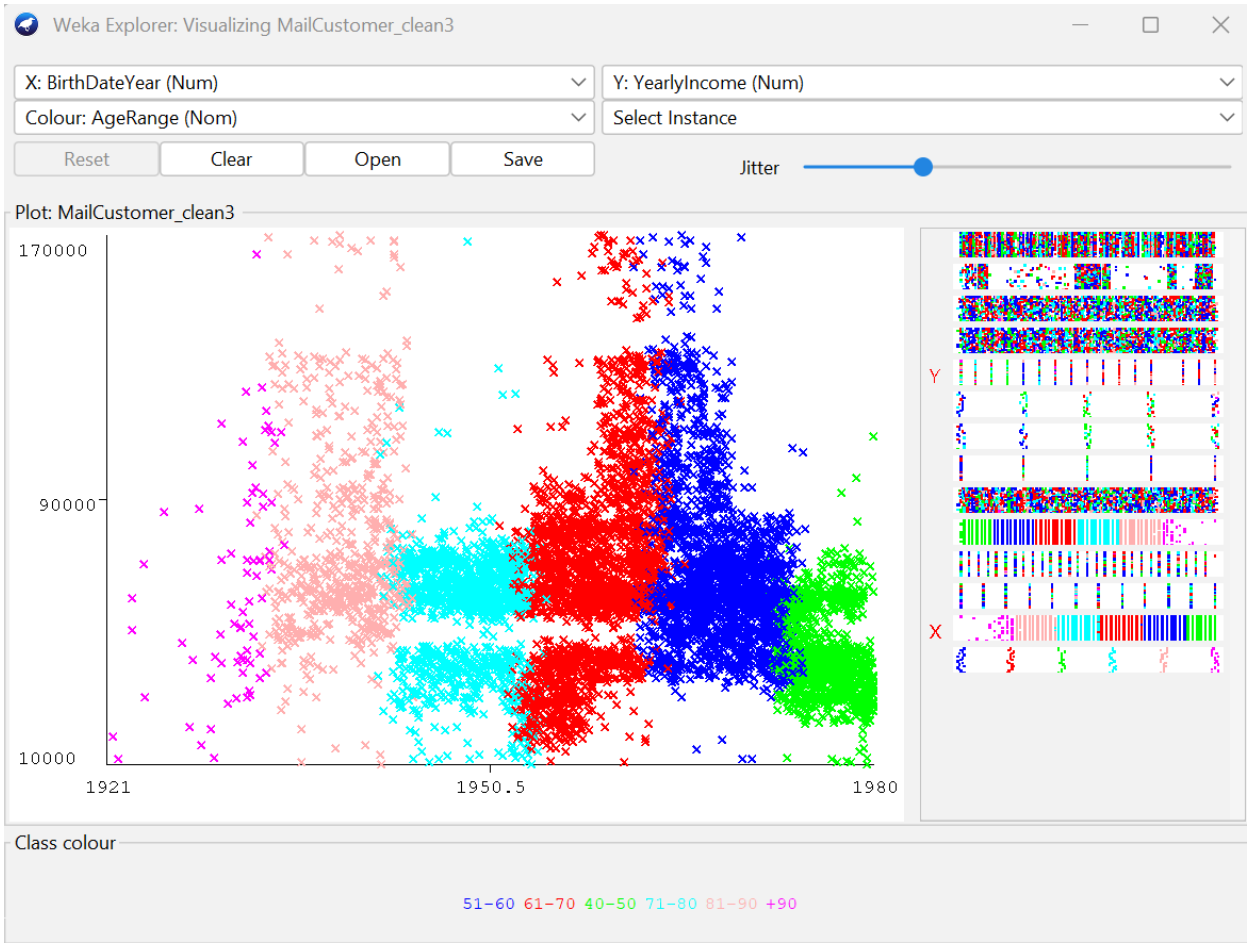

Figura 8: La gráfica de X:BirthDateYear Y:YearlyIncome

3. Realizaremos una selección de características (ranking) y despreciaremos los atributos con valoraciones muy bajas.

Para realizar un ranking de las columnas podemos ir a la sección ***"Select Atributes"***. Aquí tenemos dos campos, el ***evaluador de atributos*** y el ***método de búsqueda***. Cómo método de búsqueda podemos escoger tres: BestFirst, GreedyStepwise y Ranker, dependiendo del método WEKA escogerá un evaluador para este y viceversa. Yo he usado ***Ranker***. Y la columna usada es ***AgeRange***.

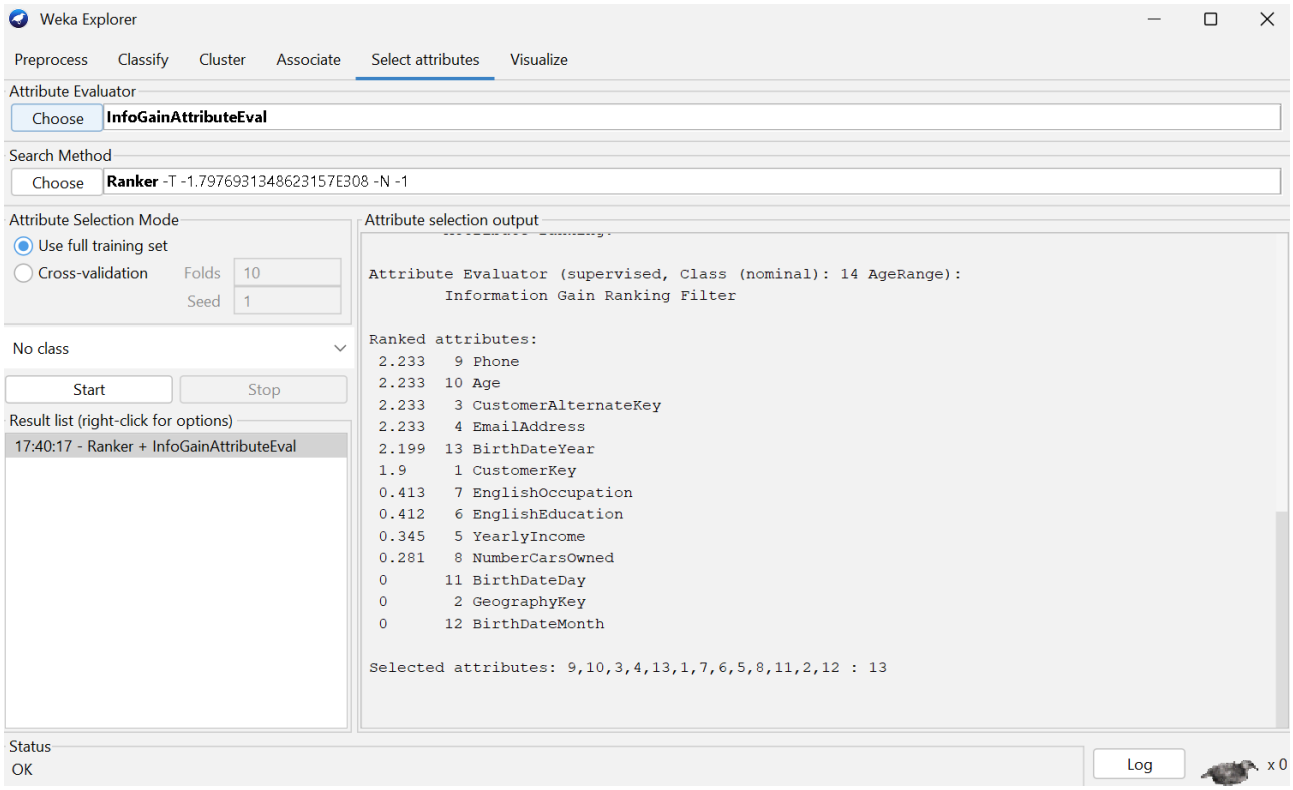

Figura 9: Ranking hecho con Ranker

Las columnas con baja o nula relevancia que se despreciarán son las siguientes:

- EnglishOccupation
- EnglishEducation
- YearlyIncome
- NumberCarsOwned
- BirthDateDay
- GeographyKey
- BirthDateMonth

4. Una vez tenemos el dataset listo, experimentaremos con las siguientes técnicas. Para cada técnica realiza un pequeño informe que detalle los resultados obtenidos y la información descubierta.

Con nuestro dataset preparado tras eliminar las columnas con baja o nula relevancia, realizaremos unas pruebas con distintas técnicas que nos ofrece WEKA.

- K-means

Para realizar el análisis de **K-means**, primero nos dirigiremos a la opción **“Cluster”** en el menú principal de la herramienta. Una vez allí, seleccionaremos el método denominado **“SimpleKMeans”**, que es uno de los algoritmos más utilizados para el agrupamiento no supervisado. Este método permite dividir un conjunto de datos en grupos homogéneos basados en sus características. Al elegir **“SimpleKMeans”**, tendremos la posibilidad de configurar diversos parámetros, como el número de clústeres deseados, el criterio de inicialización, y la distancia que se utilizará para medir la similitud entre los datos. Además, es importante verificar la correcta selección de las variables que participarán en el proceso de agrupamiento para garantizar resultados óptimos y representativos. Una vez configurado todo, procederemos a ejecutar el algoritmo y a analizar los resultados obtenidos, como la asignación de puntos a los clústeres y las estadísticas descriptivas de cada grupo.

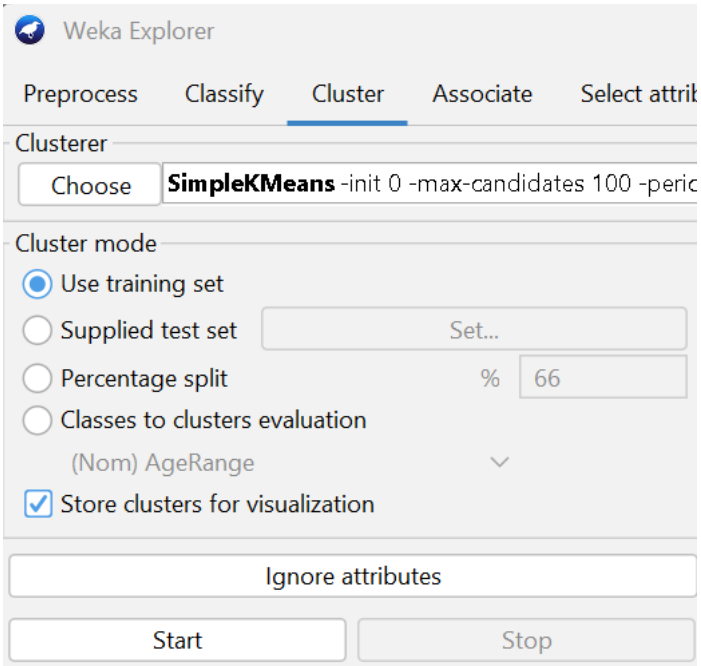

Figura 10: Opción para usar el K-means

También podemos modificar las opciones que tiene el algoritmo si hacemos click en él.

Algunas opciones que podemos encontrar son:

- ***numClusters (k)***: Aquí se puede definir el número de clústeres que quieres generar. Por ejemplo, poniendo 3, el algoritmo intentará agrupar los datos en 3 clústeres.
- ***maxIterations***: El número máximo de iteraciones que se utilizarán en el algoritmo.
- ***distanceFunction***: El tipo de función de distancia (por defecto es la distancia euclidiana).
- ***seed***: Es un valor que controla la inicialización aleatoria de los centroides de los clústeres.
- ***Otras opciones***: Se pueden ajustar opciones adicionales, como el tipo de inicialización, el número máximo de iteraciones, entre otras.

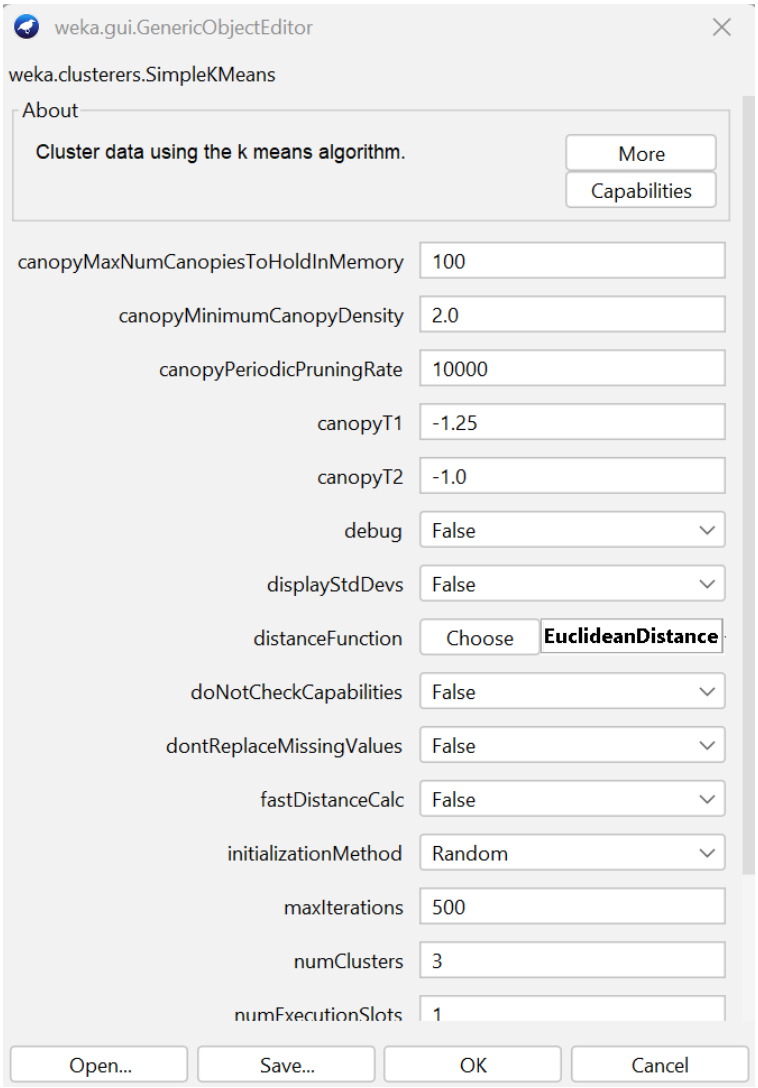

Figura 11: Opciones de K-means

Tras la ejecución, obtenemos la siguiente salida:

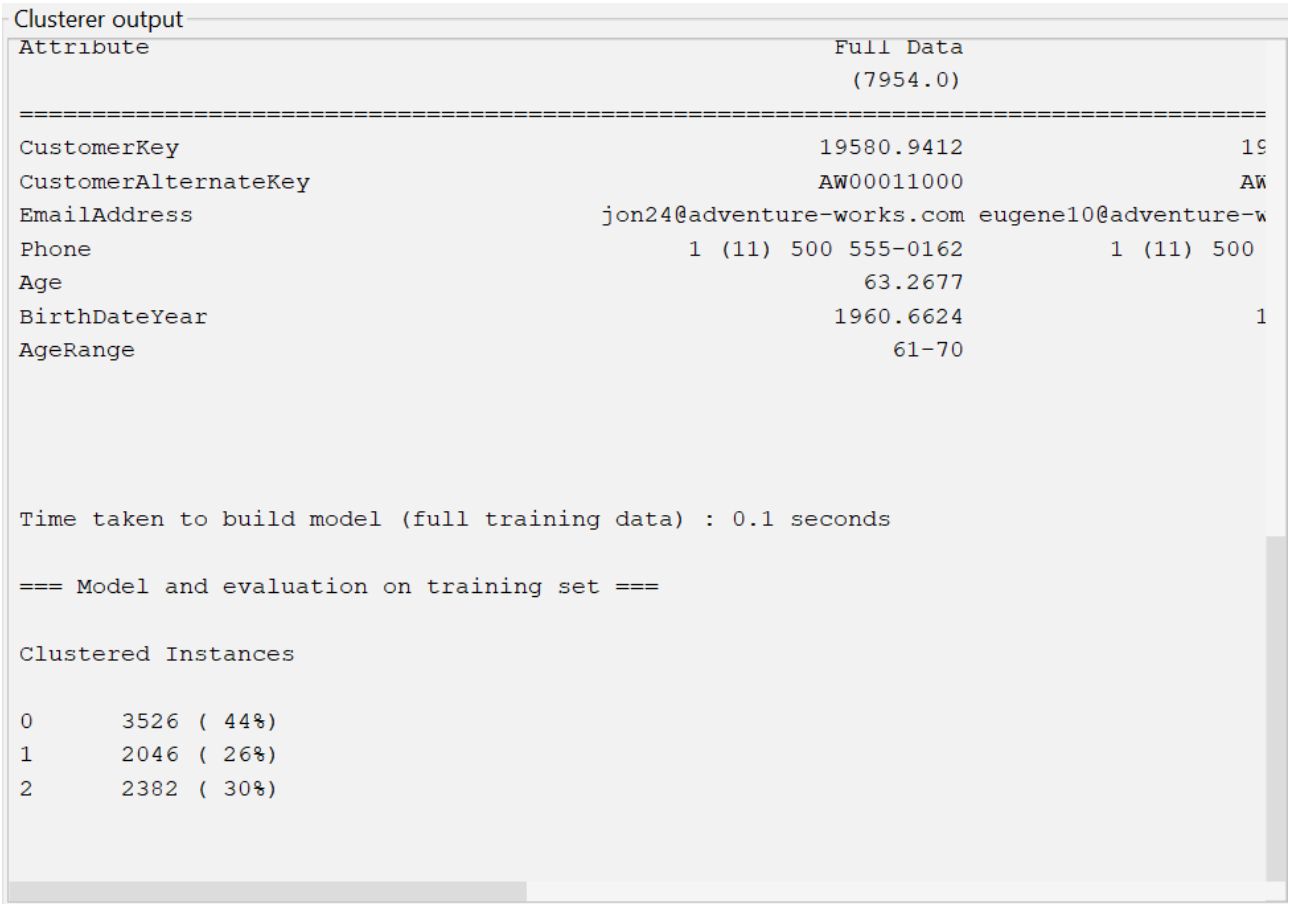

Figura 12: Resultado de K-means

El resultado obtenido muestra que los datos se han agrupado en los 3 clústeres que hemos configurado previamente. El primer campo es el ***nombre/ID*** del clúster, el segundo campo son los ***datos que se han agrupado*** en cada clúster y el último un ***porcentaje*** del total.

Las conclusiones que se pueden obtener son que la mayoría de los datos (44%) se agrupan en el clúster 0, luego el 30% en el clúster 2 y el resto en el 26%.

Si eliminamos las columnas hasta quedarnos con ***Age*** y ***YearlyIncome*** obtenemos este resultado:

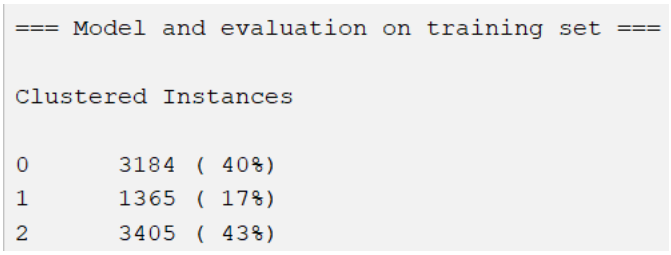

Figura 13: Segundo resultado

Vemos que el clúster 2 ahora tiene un 43%, el clúster 0 tiene un 40% y el 1 un 17%.

Esto implica que, con solo ***Age*** y ***YearlyIncome***, las diferencias entre los clústeres pueden estar más relacionadas con factores demográficos básicos (como ingresos por rango de edad) en lugar de características más complejas. El hecho de que el **Clúster 1** disminuya drásticamente podría indicar que era un grupo más específico, dependiente de características eliminadas, y menos distinguible cuando solo se analizan dos variables.

- KNN

Para implementar el algoritmo K-Nearest Neighbors (***KNN***), seguimos los pasos indicados dentro de la herramienta. Nos dirigimos a la pestaña o sección denominada "***Classify***," donde se encuentran los métodos de clasificación disponibles. Dentro de esta sección, seleccionamos la opción llamada "***IBk***," que corresponde a la implementación del algoritmo KNN. Esta herramienta permite configurar parámetros específicos, como el número de vecinos a considerar (k) y el método de distancia utilizado, lo que nos ofrece flexibilidad para ajustar el modelo a las características del conjunto de datos. Al elegir "IBk," se abre una ventana con opciones adicionales para personalizar el proceso de clasificación, asegurando que los resultados sean adecuados para nuestras necesidades específicas.

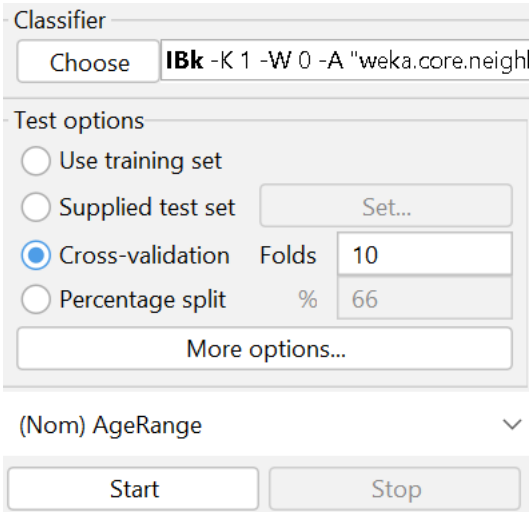

Figura 14: Selección de KNN

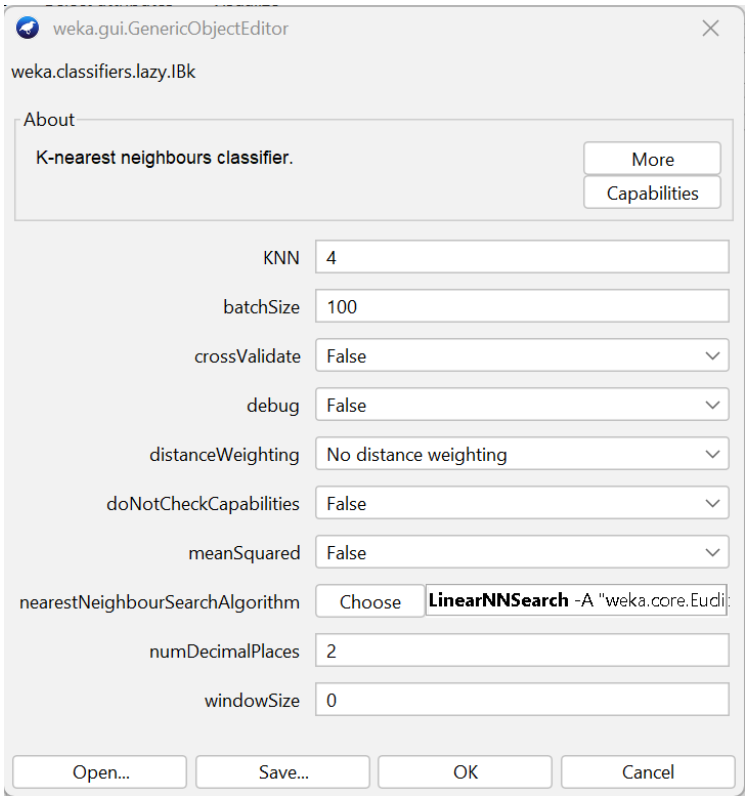

Figura 15: Opciones de KNN

Además, contamos con una variedad de hiperparámetros que desempeñan un papel crucial en la configuración y rendimiento del modelo. Entre ellos se encuentra el número de vecinos, que determina cuántos puntos de datos se considerarán para realizar las predicciones. También podemos optar por utilizar la media al cuadrado como medida para evaluar las distancias, lo que puede influir significativamente en la precisión del modelo dependiendo de la naturaleza de los datos. La validación cruzada es otro elemento fundamental, ya que nos permite evaluar la generalización del modelo dividiendo los datos en subconjuntos y probándolo de manera iterativa. Por último, el peso por distancia es un hiperparámetro que asigna mayor relevancia a los puntos más cercanos al realizar las predicciones, lo que puede ser útil para capturar patrones locales en los datos. La elección y ajuste de estos hiperparámetros son esenciales para optimizar el rendimiento del modelo y adaptarlo a las características específicas del problema que se desea resolver.

Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        7893               99.2331 %
Incorrectly Classified Instances        61                0.7669 %
Kappa statistic                          0.99
Mean absolute error                      0.0032
Root mean squared error                  0.0422
Relative absolute error                  1.2668 %
Root relative squared error             11.8206 %
Total Number of Instances             7954

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,994    0,002    0,994      0,994    0,994      0,992    0,999     0,997     51-60
                 0,995    0,003    0,992      0,995    0,994      0,991    0,999     0,998     61-70
                 0,992    0,000    1,000      0,992    0,996      0,996    0,998     0,998     40-50
                 0,993    0,002    0,992      0,993    0,992      0,991    0,999     0,998     71-80
                 0,989    0,002    0,976      0,989    0,982      0,981    1,000     0,997     81-90
                 0,877    0,001    0,941      0,877    0,908      0,908    1,000     0,982     +90
Weighted Avg.    0,992    0,002    0,992      0,992    0,992      0,990    0,999     0,997

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2333   14    0    0    0    0 |    a = 51-60
    4 2370    0    8    0    0 |    b = 61-70
    9    0 1171    0    0    0 |    c = 40-50
    0    4    0 1339    6    0 |    d = 71-80
    0    0    0    3  616    4 |    e = 81-90
    0    0    0    0    9   64 |    f = +90
```

Figura 16: Resultado del KNN

Aquí podemos ver un resumen por distintas métricas y luego una versión más detallada donde usa el rango de edad para distinguir los tipos de vecinos.

De el 100% de los datos, el 99.23% han sido clasificados correctamente mientras que tan solo un 0.8% han sido clasificados incorrectamente. En la matriz de confusión se puede ver que los datos han sido agrupados correctamente salvo por unos errores que reflejan el 0.8% mencionado antes.

Decir que entre las ***Test options*** y los botones de ***Start y Stop*** se encuentra el ***objetivo y*** que se va utilizar, pues el resto de características serían la ***matriz de características X***.

Haremos otra prueba en donde usemos 10 vecinos y con validación cruzada.

Estos son los resultados:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        6972               87.654  %
Incorrectly Classified Instances       982               12.346  %
Kappa statistic                          0.8388
Mean absolute error                      0.0612
Root mean squared error                  0.1785
Relative absolute error                 23.9689 %
Root relative squared error             49.9438 %
Total Number of Instances             7954
```

Figura 17: Resumen del segundo resultado

```
=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2037  234   76    0    0    0 |    a = 51-60
  187 2070    0  125    0    0 |    b = 61-70
   59    0 1121    0    0    0 |    c = 40-50
    0  141    0 1161   47    0 |    d = 71-80
    0    1    0   56  564    2 |    e = 81-90
    0    0    0    3   51   19 |    f = +90
```

Figura 18: Matriz de confusión del segundo resultado

Vemos que los resultados del segundo modelo indican que el modelo solo ha acertado el 87,654%. Lo que implica que este modelo con 10 vecinos con validación cruzada no es tan bueno como el anterior que tenía 4 vecinos y sin validación cruzada.

Existe una posibilidad de que el modelo con 4 vecinos esté sobreentrenado al conjunto de datos de entrenamiento, y que su desempeño sea inferior en datos no vistos. Elegir entre los dos modelos dependerá del contexto. Si la validación cruzada es importante (por ejemplo, en aplicaciones prácticas donde se prioriza la generalización), el modelo con 10 vecinos podría ser más adecuado. Si el objetivo es maximizar la precisión en este conjunto de datos específico, el modelo con 4 vecinos es mejor.

- RN (Perceptrón)

Para realizar el algoritmo de Redes Neuronales, primero debemos dirigirnos a la sección ***Classify*** dentro del programa o plataforma que estemos utilizando. Una vez allí, seleccionamos la opción ***MultilayerPerceptron***, que corresponde a un modelo de redes neuronales de tipo perceptrón multicapa. Este modelo es ampliamente utilizado en tareas de clasificación y predicción debido a su capacidad para capturar relaciones no lineales complejas en los datos. Al seleccionarlo, tendremos la posibilidad de configurar parámetros clave, como el número de capas ocultas, las funciones de activación y el número de iteraciones, que son esenciales para ajustar el modelo y optimizar su desempeño en el conjunto de datos en cuestión. Finalmente, procederemos con el entrenamiento del modelo, evaluando su desempeño para garantizar que cumple con los objetivos establecidos.

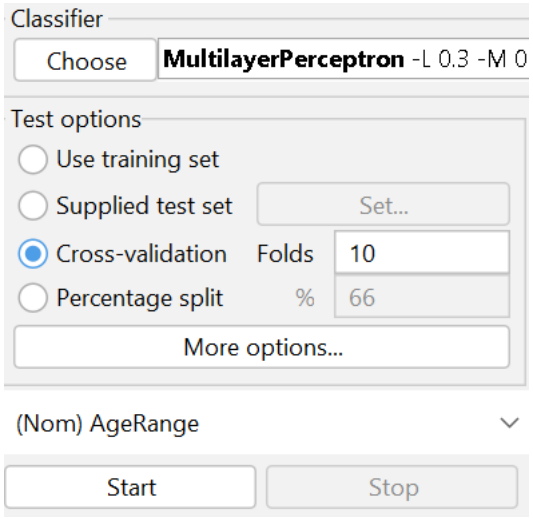

Figura 19: Selección de RN

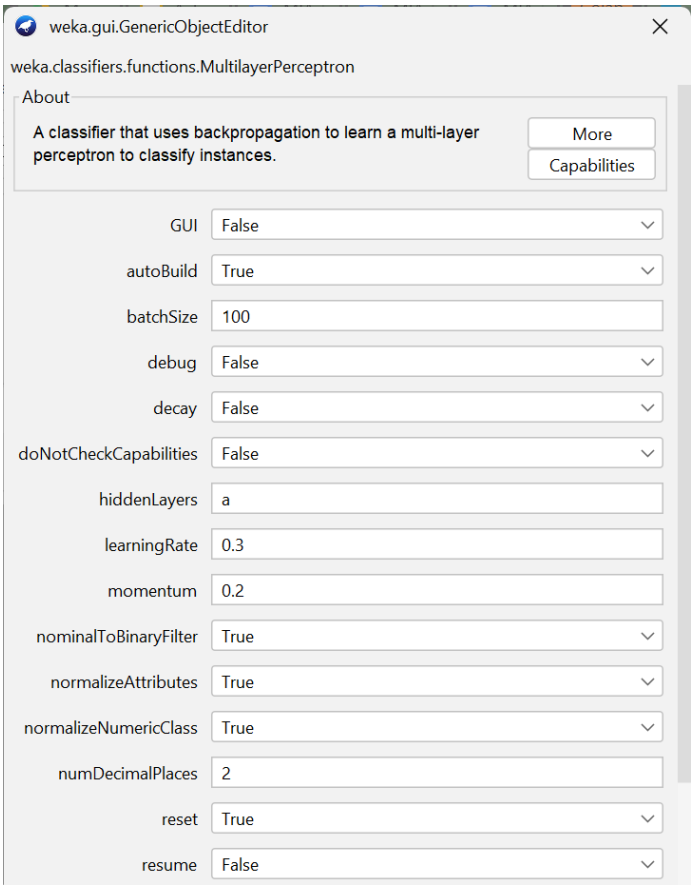

Figura 20: Opciones del RN

Para este caso, he usado las siguientes columnas para que pudiera ejecutarse el algoritmos sin problemas:

- Age
- BirthDateYear
- YearlyIncome
- EnglishEducation
- NumberCarsOwned
- AgeRange

La red neuronal resultante es la siguiente:

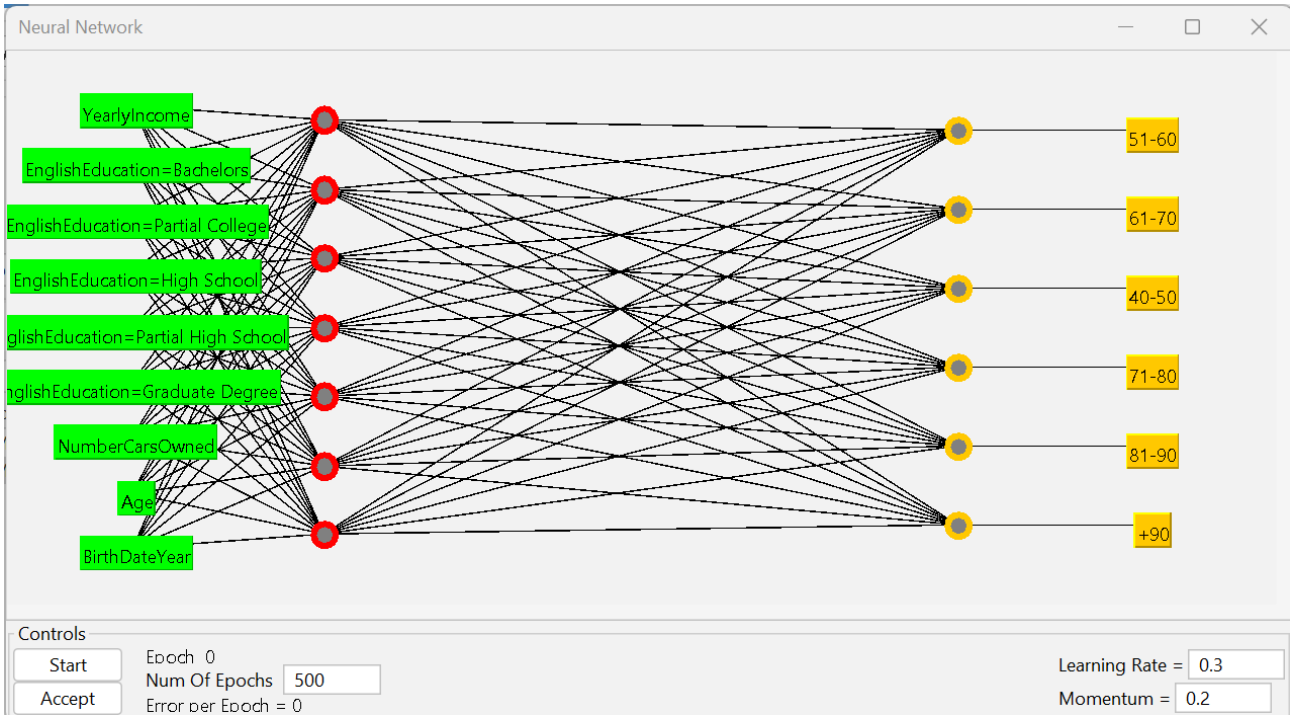

Figura 21: Red neuronal generada

Con la GUI activada, procederemos seleccionando la opción ***Start***, lo que permitirá iniciar el proceso de entrenamiento del modelo utilizando el número de ***Epochs*** previamente especificado en la configuración. Este paso marca el inicio del aprendizaje del modelo, en el que ajustará sus parámetros a través de iteraciones consecutivas sobre el conjunto de datos de entrenamiento. Una vez que el entrenamiento haya comenzado, observaremos el progreso en tiempo real, incluyendo métricas relevantes que se actualizarán tras cada época, lo que nos proporcionará una visión más clara del desempeño del modelo. Al finalizar esta etapa, daremos clic en ***Aceptar*** para proceder con las fases subsiguientes, permitiendo que el

sistema complete todas las validaciones cruzadas planificadas. Este proceso asegurará una evaluación exhaustiva del modelo mediante la partición y prueba en diferentes subconjuntos del conjunto de datos, lo que garantiza una mayor robustez y precisión en los resultados obtenidos.

El resultado es el siguiente:

Classifier output

```
=== Summary ===

Correctly Classified Instances        7900               99.3211 %
Incorrectly Classified Instances        54                0.6789 %
Kappa statistic                          0.9911
Mean absolute error                      0.0048
Root mean squared error                  0.0439
Relative absolute error                  1.8669 %
Root relative squared error             12.2779 %
Total Number of Instances             7954

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,999    0,000    1,000      0,999    0,999      0,999    1,000     1,000     51-60
                 0,999    0,000    1,000      0,999    0,999      0,999    1,000     1,000     61-70
                 1,000    0,000    0,997      1,000    0,999      0,999    1,000     0,997     40-50
                 0,999    0,000    0,999      0,999    0,999      0,998    1,000     1,000     71-80
                 0,982    0,005    0,943      0,982    0,962      0,959    0,999     0,990     81-90
                 0,521    0,001    0,776      0,521    0,623      0,633    0,995     0,672     +90
Weighted Avg.    0,993    0,001    0,993      0,993    0,993      0,992    1,000     0,996

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2344    0    3    0    0    0 |    a = 51-60
    1 2379    0    2    0    0 |    b = 61-70
    0    0 1180    0    0    0 |    c = 40-50
    0    0    0 1347    2    0 |    d = 71-80
    0    0    0    0  612   11 |    e = 81-90
    0    0    0    0   35   38 |    f = +90
```

Figura 22: Resultado de la red neuronal

El modelo ha arrojado unos resultados bastante satisfactorios, ya que tan solo 54 datos han sido clasificados incorrectamente, lo que representa un bajo porcentaje en comparación con el total de datos procesados. Este desempeño indica que la precisión del modelo es alta y su capacidad de generalización es adecuada para este conjunto de datos. Además, podemos afirmar que los nodos ***Age*** y ***BirthDateYear*** juegan un papel fundamental en la clasificación, ya que proporcionan información clave que permite inferir de manera precisa el rango de edad.

Esto sugiere que estas variables poseen un alto poder predictivo y podrían ser consideradas como las características más relevantes dentro del modelo. Por tanto, su impacto en los resultados refuerza la importancia de seleccionar y utilizar atributos significativos durante el proceso de diseño y entrenamiento del modelo.

Probemos a quitar columnas y usar solamente ***YearlyIncome*** y ***NumberCarsOwned***.

Los resultados son los siguientes:

```
=== Summary ===

Correctly Classified Instances        3673               46.178  %
Incorrectly Classified Instances      4281               53.822  %
Kappa statistic                          0.2889
Mean absolute error                      0.2143
Root mean squared error                  0.3284
Relative absolute error                 83.8982 %
Root relative squared error             91.8975 %
Total Number of Instances             7954
```

Figura 23: Resultados del segundo modelo

```
=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1255  771   96  202   23    0 |    a = 51-60
  603 1151  319  289   20    0 |    b = 61-70
   60  240  685  187    8    0 |    c = 40-50
  169  310  324  546    0    0 |    d = 71-80
   12  264   91  220   36    0 |    e = 81-90
   15   33   13   11    1    0 |    f = +90
```

Figura 24: Matriz de confusión del segundo modelo

Vemos que este modelo es mucho peor que el anterior, de hecho falla hasta un 53,82%. Lo que sugiere que ***YearlyIncome*** y ***NumberCarsOwned*** no tienen ninguna relación con ***AgeRange***, por lo que las columnas que más pistas pueden dar son efectivamente ***Age*** y ***BirthDateYear***.

- Árbol de decisión (J48)

Para ejecutar el algoritmo J48, lo cambiaríamos en la sección de *trees* dentro del programa o del entorno de desarrollo utilizado. El algoritmo J48 es una implementación del árbol de decisión C4.5 y se utiliza para clasificar datos en función de una serie de características. Para ello, primero debemos asegurarnos de que los datos estén preparados y que el entorno de ejecución, como WEKA, tenga la librería o clase que contiene J48. Una vez que se haya configurado el entorno, se selecciona el modelo J48 desde el menú o desde el código, dependiendo de si se está trabajando en una interfaz gráfica o en un entorno de programación. A continuación, se puede ajustar una serie de parámetros, como el tamaño mínimo de las hojas del árbol o el umbral de ganancia para realizar particiones en los datos. Con estos ajustes, el algoritmo J48 podrá generar el árbol de decisión, el cual se entrenará utilizando el conjunto de datos proporcionado. Este árbol se puede usar luego para realizar predicciones sobre nuevos datos y analizar su rendimiento con métricas como la precisión o la matriz de confusión.

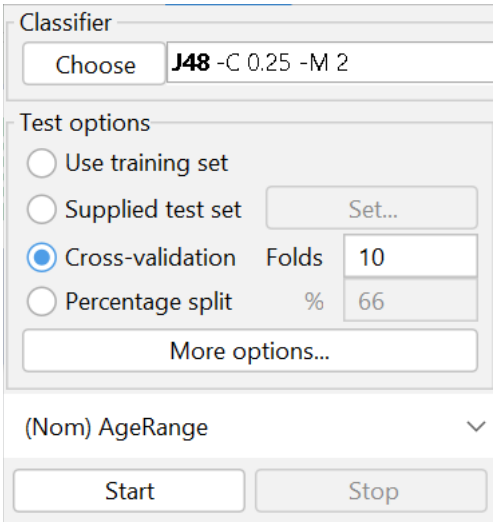

Figura 25: Selección del algoritmo J48

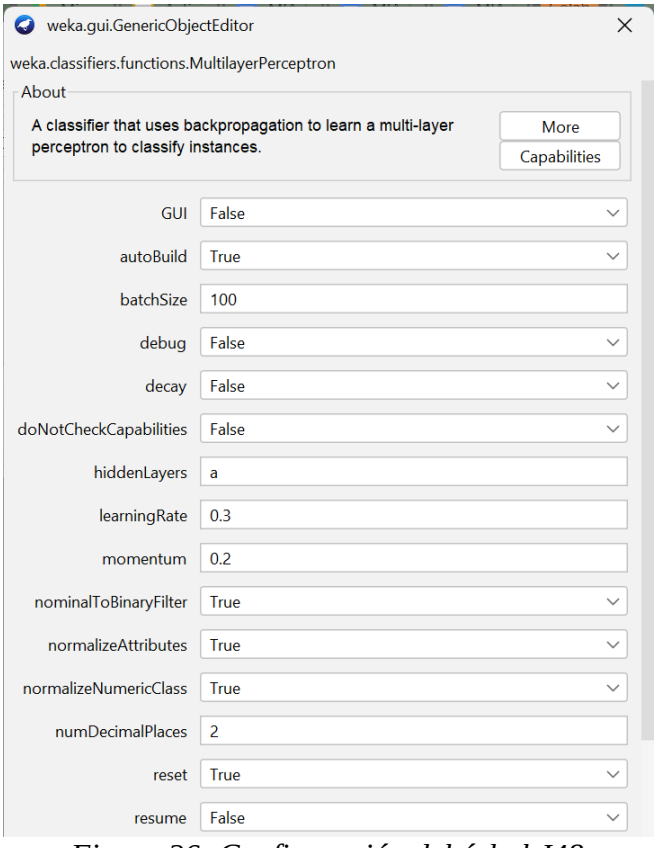

Figura 26: Configuración del árbol J48

Experimentando con nuestros datos en WEKA

El árbol generado es el siguiente:

```
J48 pruned tree
------------------

BirthDateYear <= 1963
|   BirthDateYear <= 1953
|   |   BirthDateYear <= 1943
|   |   |   BirthDateYear <= 1933: +90 (74.0/1.0)
|   |   |   BirthDateYear > 1933: 81-90 (629.0/7.0)
|   |   BirthDateYear > 1943: 71-80 (1353.0/11.0)
|   BirthDateYear > 1953: 61-70 (2391.0/20.0)
BirthDateYear > 1963
|   BirthDateYear <= 1973: 51-60 (2338.0/11.0)
|   BirthDateYear > 1973: 40-50 (1169.0)

Number of Leaves  :     6

Size of the tree :      11


Time taken to build model: 0.09 seconds
```

Figura 27: Árbol J48

El resultado arrojado también es bastante bueno:

Classifier output

```
=== Summary ===

Correctly Classified Instances        7904               99.3714 %
Incorrectly Classified Instances        50                0.6286 %
Kappa statistic                          0.9918
Mean absolute error                      0.0042
Root mean squared error                  0.0456
Relative absolute error                  1.6282 %
Root relative squared error             12.7702 %
Total Number of Instances             7954

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,991    0,002    0,995      0,991    0,993      0,991    0,996     0,989     51-60
                 0,995    0,004    0,992      0,995    0,994      0,991    0,997     0,986     61-70
                 0,991    0,000    1,000      0,991    0,995      0,995    0,998     0,994     40-50
                 0,995    0,002    0,992      0,995    0,993      0,992    0,998     0,984     71-80
                 0,998    0,001    0,989      0,998    0,994      0,993    0,998     0,980     81-90
                 1,000    0,000    0,986      1,000    0,993      0,993    1,000     0,972     +90
Weighted Avg.    0,994    0,002    0,994      0,994    0,994      0,992    0,997     0,987
```

Figura 28: Resultados del árbol

Solamente ha fallado 50 datos, lo cierto es que el árbol de decisión era bastante sencillo y es normal que fallara muy poco, ya que con una estructura simple y pocos nodos, el modelo puede ajustarse con facilidad a los datos disponibles. Además, al contar con un solo atributo para usar, en este caso la edad, es mucho más sencillo identificar patrones y rangos específicos. La variable de edad es generalmente un buen predictor en problemas donde se busca clasificar o segmentar, ya que tiene una relación directa con varios factores de comportamiento o características demográficas. En este caso, la tarea de determinar los rangos de edad se vuelve aún más accesible debido a la simplicidad del modelo, lo que permite que el árbol realice predicciones precisas con un mínimo de error.

Por último, la matriz de confusión (que muestra las clasificaciones erróneas) es la siguiente:

```
=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2327   20    0    0    0    0 |    a = 51-60
    0 2371    0   11    0    0 |    b = 61-70
   11    0 1169    0    0    0 |    c = 40-50
    0    0    0 1342    7    0 |    d = 71-80
    0    0    0    0  622    1 |    e = 81-90
    0    0    0    0    0   73 |    f = +90
```

Figura 29: Matriz de confusión del árbol J48

Si realizamos el árbol de la misma forma que la red neuronal obtenemos el siguiente árbol:

```
J48 pruned tree
------------------

NumberCarsOwned ≤ 1
|   YearlyIncome ≤ 40000
|   |   NumberCarsOwned ≤ 0
|   |   |   YearlyIncome ≤ 30000
|   |   |   |   YearlyIncome ≤ 20000
|   |   |   |   |   YearlyIncome ≤ 10000: 51-60 (6.0/4.0)
|   |   |   |   |   YearlyIncome > 10000: 61-70 (7.0/2.0)
|   |   |   |   YearlyIncome > 20000: 40-50 (14.0/5.0)
|   |   |   YearlyIncome > 30000: 51-60 (132.0/22.0)
|   |   NumberCarsOwned > 0
|   |   |   YearlyIncome ≤ 30000: 40-50 (160.0/87.0)
|   |   |   YearlyIncome > 30000: 61-70 (561.0/291.0)
```

```
|   YearlyIncome > 40000
|   |   YearlyIncome ≤ 50000: 51-60 (330.0/36.0)
|   |   YearlyIncome > 50000
|   |   |   NumberCarsOwned ≤ 0
|   |   |   |   YearlyIncome ≤ 70000: 51-60 (988.0/293.0)
|   |   |   |   YearlyIncome > 70000
|   |   |   |   |   YearlyIncome ≤ 80000: 61-70 (211.0/93.0)
|   |   |   |   |   YearlyIncome > 80000
|   |   |   |   |   |   YearlyIncome ≤ 90000: 61-70 (22.0/7.0)
|   |   |   |   |   |   YearlyIncome > 90000: 51-60 (37.0/18.0)
|   |   |   NumberCarsOwned > 0
|   |   |   |   YearlyIncome ≤ 70000: 61-70 (842.0/445.0)
|   |   |   |   YearlyIncome > 70000
|   |   |   |   |   YearlyIncome ≤ 80000: 51-60 (151.0/51.0)
|   |   |   |   |   YearlyIncome > 80000
|   |   |   |   |   |   YearlyIncome ≤ 90000: 61-70 (108.0/43.0)
|   |   |   |   |   |   YearlyIncome > 90000: 51-60 (51.0/27.0)
NumberCarsOwned > 1
|   YearlyIncome ≤ 70000
|   |   YearlyIncome ≤ 40000
|   |   |   YearlyIncome ≤ 20000
|   |   |   |   YearlyIncome ≤ 10000: 71-80 (21.0/5.0)
|   |   |   |   YearlyIncome > 10000
|   |   |   |   |   NumberCarsOwned ≤ 2: 61-70 (145.0/39.0)
|   |   |   |   |   NumberCarsOwned > 2: 71-80 (5.0)
|   |   |   YearlyIncome > 20000
|   |   |   |   NumberCarsOwned ≤ 2: 40-50 (1066.0/475.0)
|   |   |   |   NumberCarsOwned > 2: 71-80 (82.0/40.0)
|   |   YearlyIncome > 40000
|   |   |   YearlyIncome ≤ 50000
|   |   |   |   NumberCarsOwned ≤ 2: 81-90 (165.0/75.0)
|   |   |   |   NumberCarsOwned > 2: 51-60 (29.0)
|   |   |   YearlyIncome > 50000: 71-80 (1733.0/1088.0)
```

```
|   YearlyIncome > 70000
|   |   YearlyIncome ≤ 80000: 61-70 (247.0/88.0)
|   |   YearlyIncome > 80000
|   |   |   NumberCarsOwned ≤ 3
|   |   |   |   YearlyIncome ≤ 120000
|   |   |   |   |   NumberCarsOwned ≤ 2
|   |   |   |   |   |   YearlyIncome ≤ 90000: 81-90 (91.0/40.0)
|   |   |   |   |   |   YearlyIncome > 90000
|   |   |   |   |   |   |   YearlyIncome ≤ 100000: 51-60 (36.0/13.0)
|   |   |   |   |   |   |   YearlyIncome > 100000
|   |   |   |   |   |   |   |   YearlyIncome ≤ 110000: 61-70 (34.0/23.0)
|   |   |   |   |   |   |   |   YearlyIncome > 110000: 51-60 (7.0/3.0)
|   |   |   |   |   NumberCarsOwned > 2
|   |   |   |   |   |   YearlyIncome ≤ 90000: 51-60 (67.0/21.0)
|   |   |   |   |   |   YearlyIncome > 90000
|   |   |   |   |   |   |   YearlyIncome ≤ 110000: 61-70 (103.0/64.0)
|   |   |   |   |   |   |   YearlyIncome > 110000: 81-90 (45.0/22.0)
|   |   |   |   YearlyIncome > 120000
|   |   |   |   |   NumberCarsOwned ≤ 2
|   |   |   |   |   |   YearlyIncome ≤ 130000: 51-60 (28.0/6.0)
|   |   |   |   |   |   YearlyIncome > 130000: 61-70 (31.0/16.0)
|   |   |   |   |   NumberCarsOwned > 2: 61-70 (102.0/59.0)
|   |   |   NumberCarsOwned > 3
|   |   |   |   YearlyIncome ≤ 90000: 61-70 (13.0)
|   |   |   |   YearlyIncome > 90000
|   |   |   |   |   YearlyIncome ≤ 100000: 51-60 (55.0/26.0)
|   |   |   |   |   YearlyIncome > 100000: 61-70 (229.0/119.0)

Number of Leaves  :     37

Size of the tree :     73
```

Por último, miramos los resultados obtenidos:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4217               53.0173 %
Incorrectly Classified Instances      3737               46.9827 %
Kappa statistic                          0.3857
Mean absolute error                      0.1976
Root mean squared error                  0.3154
Relative absolute error                 77.3635 %
Root relative squared error             88.2639 %
Total Number of Instances             7954
```

Figura 30: Resultados del segundo modelo

```
=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1367  658   16  229   77    0 |    a = 51-60
  402 1297  226  404   53    0 |    b = 61-70
   56  183  673  246   22    0 |    c = 40-50
   70  275  289  712    3    0 |    d = 71-80
   25  137   27  266  168    0 |    e = 81-90
   10   33   11   13    6    0 |    f = +90
```

Figura 31: Matriz de confusión del segundo modelo

Esto vuelve a confirmar que ***AgeRange*** está estrechamente relacionada con ***Age*** y ***BirthDateYear*** y que sin ellas nuestro modelos tienen altas probabilidades de fallar. En este caso, el modelo ha acertado 53.02%, algo mejor que la red neuronal.

Conclusiones

Tras el análisis, hemos llegado a la conclusión de que ***AgeRange*** está sujeta a las características ***Age*** y ***BirthDateYear***. Y que nuestro modelos necesitan de ellas para tener una mejor precisión. Hemos aprendido como hacer un ranking de nuestras características y ver cuales son más relevantes en base a nuestro objetivo. Y también hemos visto como se visualizan los datos en las gráficas de WEKA. Otras pruebas que se podrían hacer serían añadir características ya sea creándolas o que estuvieran en nuestro dataset y fueron borradas anteriormente.