INTELIGENCIA ARTIFICIAL II

Implementación de un som

Matías Sanz – Diego Valencia – Ana schleissner- Sara peirotén

2023

[1 Práctica 1: Clasificar colores 3](#_Toc127818874)

[1.1 Implementación 3](#_Toc127818875)

[1.1.1 Implementación del código para obtener el Dataset. 3](#_Toc127818876)

[1.1.2 SOM Setup. 5](#_Toc127818877)

[1.1.3 SOM Entrenamiento 8](#_Toc127818878)

[1.1.4 SOM Clasificación 9](#_Toc127818879)

[1.2 Cuestiones 11](#_Toc127818881)

[1.2.1 ¿Cuáles son los valores más adecuados para lado\_mapa, periodo y eta? 11](#_Toc127818882)

[1.2.2 Datos de la mejor clasificación 17](#_Toc127818883)

[1.2.2.1 Gráfico RGB de pesos iniciales 17](#_Toc127818884)

[1.2.2.2 Gráfico RGB de los pesos entrenados 17](#_Toc127818885)

[1.2.2.3 Número de clases, error topológico y error de cuantificación 18](#_Toc127818886)

[1.2.2.4 Mapa de clasificación 18](#_Toc127818887)

[1.2.2.5 Mapa de activaciones 19](#_Toc127818888)

[1.2.2.6 Mapa de distancias 19](#_Toc127818889)

[1.2.3 Dataset de prueba 20](#_Toc127818890)

[2 Práctica 2: Clasificar países por nivel de desarrollo 23](#_Toc127818891)

[2.1 Implementación 23](#_Toc127818892)

[2.1.1 Obtención del Dataset 23](#_Toc127818893)

[2.1.2 Entrenamiento 24](#_Toc127818894)

[2.1.3 Modificación del mapa de clasificación 24](#_Toc127818895)

[2.2 Cuestiones 25](#_Toc127818896)

[2.2.1 Valores de Lado\_Mapa, Periodo y Eta más adecuados. 25](#_Toc127818897)

[2.2.2 Resultados elegidos 25](#_Toc127818898)

[2.2.3.1 ¿Qué grupos se pueden establecer? 29](#_Toc127818899)

[2.2.3.2 Análisis de los vectores de pesos de Venezuela, Noruega y Etiopía. 29](#_Toc127818900)

[2.2.3.3 Mapa de clasificación. 29](#_Toc127818901)

[2.2.3.4 Ubicaciones de los países subdesarrollados. 29](#_Toc127818902)

# 1 Práctica 1: Clasificar colores

Esta primera práctica consiste en el desarrollo de una herramienta que permita la generación de mapas auto asociativos que pueda crear y entrenar redes SOM a partir de un dataset de entrada y unos parámetros introducidos en tiempo de ejecución.

En nuestro caso concreto, se entrenará con un grupo de hasta cien colores. Una vez entrenada, será capaz de distinguir los diferentes colores presentados en formato R.G.B.

Tras el entrenamiento, se clasificarán los colores para ver los mapas de clasificación, distancias y activaciones. Para encontrar las mejores variables que se ajusten a nuestra red, hemos sometido a pruebas con variables diferentes para poder observar el comportamiento y poder elegir lo que más nos interese.

# 1.1 Implementación

Se divide en cuatro partes, así como el enunciado indica, para el mejor desarrollo de esta memoria, seguimos el mismo esquema relatando en cada una de las partes lo que creemos conveniente.

# 1.1.1 Implementación del código para obtener el Dataset.

El objetivo es crear una matriz de 100 colores al azar, cada uno descrito por su codificación RGB, al que llamaremos *datos*.

Para definir un color cualquiera dentro de este modelo, hemos de conocer que, los colores RGB se definen como 3 valores, cada uno de ellos comprendidos entre 0 y 255, la posición de cada uno marca la intensidad de Rojo, Verde y Azul que tendrá el color obtenido.

De los parámetros que la función del enunciado necesita, destacamos dos, valor\_max y valor\_max, en el caso de los valores RGB, sabemos que comprenden entre 0 y 255, luego asignamos cada uno a los límites inferior y superior, respectivamente.

Añadimos un 3 para marcar que queremos 3 valores entre los intervalos especificados y 100 para el número total de colores que queremos.

Imagen que contiene Texto

Descripción generada automáticamenteUsando la librería *numpy* recogemos estos valores en una matriz que conformará el Dataset deseado.

# 1.1.2 SOM Setup.

Para construir e inicializar el SOM, hemos de inicializar también sus valores y atributos.

Texto

Descripción generada automáticamente

Hemos de definir el *lado del mapa*, esto marca el número de neuronas por lado del mapa de Kohonen. Tras el análisis de las pruebas hemos elegido el número que creemos mejores resultados nos aporta.

El *periodo* define el número de iteraciones, este número marca mucho el tiempo de ejecución, pero hemos de encontrar un valor para el que, a pesar de sumarle muchas iteraciones, el resultado siga siendo de una calidad similar sin aumentar exponencialmente el tiempo de ejecución.

El *learning rate* o coeficiente de aprendizaje es el porcentaje de cambio con el que se actualizan los pesos en cada una de las iteraciones. Un valor muy alto hace que el valor de la salida sea menos preciso, a lo largo del entrenamiento se quiere ir obteniendo una disminución progresiva del valor.

Texto, Aplicación

Descripción generada automáticamenteTodos estos valores se introducen a mano y son resultado de nuestras pruebas y test.

Ahora establecemos las funciones que definirán el SOM.

Primero definimos el número de entradas, en nuestro caso equivale al lado del mapa, seguimos con el número de datos que obtiene el número de parámetros de un patrón.

Seguimos con el cálculo del vecindario inicial, en el caso de Kohonen, este equivale a la mitad del lado de mapa. El vecindario es el conjunto de neuronas adyacentes a la neurona ganadora.

A la hora de normalizar los datos hemos de dividir cada dato por el valor máximo de la matriz, o nuestra variable valor\_max.

Creamos una matriz de pesos con los valores aleatorios entre 0 y 1, esta matriz es tridimensional usando el lado del mapa, dos veces, y el número de datos (num\_datos).

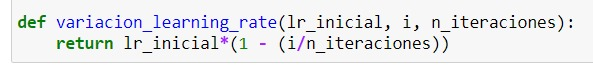
Obtenida la matriz de pesos nos disponemos a crear las funciones de entrenamiento y clasificación.

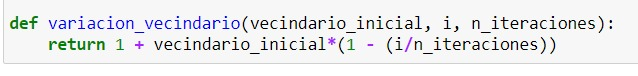
Primero hemos de obtener la BMU o neurona ganadora, de la capa de Kohonen, esta neurona se define por tener el vector de pesos más cercano al patrón de entrada. La función contará con el patrón de entrada, el número de entradas y la matriz de pesos.

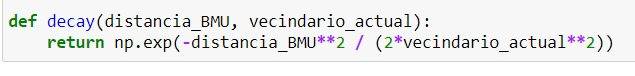
Para seleccionar la BMU recogemos, en una matriz de dimensiones *lado\_mapa* y *lado\_mapa*, las distancias euclídeas entre el patrón y las neuronas, para poder obtener el valor más pequeño, que es el que nos interesa. Guardaremos el vector de pesos de esa neurona en concreto, así como sus coordenadas.

Texto

Descripción generada automáticamente

El tercer punto que debemos abordar es el descenso del coeficiente de aprendizaje o eta, nuestro objetivo sigue siendo, que, a raíz de las numerosas iteraciones, este vaya descendiendo su valor hasta 0.

El vecindario de una neurona también se va reduciendo, respondiendo a la fórmula, en función del número de iteraciones hasta legar a 1.

Para calcular el descenso del coeficiente de aprendizaje en función a la distancia de la BMU hacemos uso de una función ya codificada en el enunciado llamada *decay*, esta funciona variando en menor medida de peso de las neuronas cuanto más alejado se encuentren de la BMU.

Ahora para pintar la matriz de valores como colores RGB hacemos uso también del enunciado con la función *pintar\_mapa*. Se ha modificado la función para mostrar la variación del nº de iteraciones.

# 1.1.3 SOM Entrenamiento

Para el entrenamiento se generará un patrón aleatorio obtenido de nuestro conjunto de datos de entrada. Esto se hará tantas veces como *periodo* se haya especificado.

Por cada iteración calcularemos la neurona ganadora (BMU), el vecindario y la variación de eta (indica como desciende el coeficiente de aprendizaje). Cada vez que calculamos cada uno de estos parámetros, se ajustan los pesos de la BMU y de las neuronas pertenecientes al vecindario.

Utilizamos sumatorio para calcular la norma euclidiana del vector de pesos actualizado. Primero se actualizan los pesos con la fórmula correspondiente y después se calcula la norma euclidiana del vector resultante utilizando la fórmula del sumatorio de los cuadrados de cada componente del vector, que se almacenan en la variable sumatorio. Al final se divide el vector actualizado por la norma euclidiana calculada para normalizarlo.

Texto

Descripción generada automáticamente

# 1.1.4 SOM Clasificación

La clasificación tiene como tarea asignar cada uno de los patrones de entrada a una de clases existentes:

* Patrones existentes.
* Patrones nuevos.

Para hacer una buena clasificación se deben de calcular antes una serie de valores entre los que están:

**El error de cuantificación:** es la distancia promedio entre cada vector de datos y su BMU.



**El error topológico:** es una proporción de todos los vectores de datos para los que la primera y segunda BMU son adyacentes.

Texto

Descripción generada automáticamentePor último, crearemos las tres matrices que se nos piden para mostrar los errores topológicos y de cuantificación. Además, también se calcula el número de clases que se crean.

Texto, Carta

Descripción generada automáticamente

# 1.2 Cuestiones

# 1.2.1 ¿Cuáles son los valores más adecuados para lado\_mapa, periodo y eta?

A la hora de elegir los valores más adecuados para nuestra red, hemos sometido a una serie de pruebas el código que tenemos hasta el momento. Estas pruebas sirven para la elección de los valores de las variables lado\_mapa, periodo y learning\_rate, que mejor nos convengan.

Para la elección de lado\_mapa, hemos probado a iterar con diferentes valores comprendidos entre 5 y 100, los valores que queremos observar son los errores topológico y cuantitativo, así como el número de clases.

Podemos ver en el gráfico de abajo el lado del mapa más optimo.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Tras analizar los diferentes valores teniendo siempre un periodo de 10000 y un eta de 0.4, lo que queremos buscar es el menor error topológico y cuantitativo, mientras que el valor de las clases es estable. Podemos observar que las clases se estabilizan desde que el lado del para tiene valor 30, hasta que tiene valor 100.

Fijándonos en los errores de los lados comprendidos entre 30 y 100, el que menos error topológico tiene es el 35 ya que es cercano a 0 y el error cuantitativo es considerablemente bajo también, por lo que hemos decidido coger ese valor.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lado mapa | Periodo | eta | Error topológico | Error cuantitativo | Clases |
| 5 | **10000** | **0.4** | **0.0** | **0.2780** | **25** |
| 10 | **10000** | **0.4** | **0.01** | **0.2255** | **68** |
| 15 | **10000** | **0.4** | **0.01** | **0.2256** | **87** |
| 20 | **10000** | **0.4** | **0.03** | **0.2331** | **94** |
| 25 | **10000** | **0.4** | **0.01** | **0.2518** | **99** |
| 30 | **10000** | **0.4** | **0.01** | **0.2288** | **97** |
| 35 | **10000** | **0.4** | **0.05** | **0.2020** | **100** |
| 40 | **10000** | **0.4** | **0.03** | **0.2136** | **100** |
| 45 | **10000** | **0.4** | **0.04** | **0.2030** | **100** |
| 50 | **10000** | **0.4** | **0.08** | **0.1994** | **100** |
| 55 | **10000** | **0.4** | **0.06** | **0.2413** | **100** |
| 60 | **10000** | **0.4** | **0.08** | **0.2165** | **100** |

La grafica tiene 3 variables:

La línea roja; el número de clases, la línea morada; el error de cuantificación y la verde; el error topológico.

Mirando a la gráfica, podemos ver que no podemos coger valores por debajo de 30 ya que el número de clases no está estabilizado. Mirando al error topológico, podemos ver que teniendo 35 lados del mapa es 0.00 y en ese mismo valor, podemos ver que el error cuantitativo también es bastante bajo. Por lo cual, hemos elegido entrenar al algoritmo con un mapa de Kohonen de 35 neuronas por lado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Periodo | Lado mapa | eta | Error topológico | Error cuantitativo | Clases |
| 1000 | **35** | **0.3** | **0.04** | **0.2606** | **94** |
| 2000 | **35** | **0.3** | **0.04** | **0.2093** | **97** |
| 3000 | **35** | **0.3** | **0.03** | **0.2216** | **99** |
| 4000 | **35** | **0.3** | **0.04** | **0.2241** | **99** |
| 5000 | **35** | **0.3** | **0.06** | **0.2306** | **99** |
| 6000 | **35** | **0.3** | **0.01** | **0.2250** | **98** |
| 7000 | **35** | **0.3** | **0.0** | **0.2601** | **98** |
| 8000 | **35** | **0.3** | **0.0** | **0.2474** | **100** |
| 9000 | **35** | **0.3** | **0.01** | **0.2651** | **100** |
| 10000 | **35** | **0.3** | **0.01** | **0.2144** | **99** |
| 11000 | **35** | **0.3** | **0.0** | **0.2276** | **99** |
| 12000 | **35** | **0.3** | **0.05** | **0.2129** | **100** |
| 13000 | **35** | **0.3** | **0.01** | **0.2669** | **100** |
| 14000 | **35** | **0.3** | **0.0** | **0.2538** | **99** |
| 15000 | **35** | **0.3** | **0.05** | **0.2008** | **98** |

Gráfico, Gráfico de líneas

Descripción generada automáticamenteA continuación, para calcular el periodo hemos realizado una serie de pruebas en las que el *lado del mapa* se mantiene constante (35) puesto que anteriormente hemos decidido que es el valor óptimo para esta variable. *Eta lo dejaremos a 0.3 e iremos variando el periodo:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| eta | Lado mapa | Periodo | Error topológico | Error cuantitativo | Clases |
| 0.06 | **35** | **8000** | **0.03** | **0.2417** | **98** |
| 0.12 | **35** | **8000** | **0.0** | **0.2186** | **97** |
| 0.18 | **35** | **8000** | **0.02** | **0.2372** | **100** |
| 0.24 | **35** | **8000** | **0.02** | **0.2122** | **99** |
| 0.30 | **35** | **8000** | **0.02** | **0.2199** | **100** |
| 0.36 | **35** | **8000** | **0.03** | **0.2199** | **99** |
| 0.42 | **35** | **8000** | **0.05** | **0.2143** | **100** |
| 0.48 | **35** | **8000** | **0.02** | **0.2138** | **100** |
| 0.54 | **35** | **8000** | **0.04** | **0.2500** | **98** |
| 0.60 | **35** | **8000** | **0.03** | **0.2197** | **99** |
| 0.66 | **35** | **8000** | **0.05** | **0.2166** | **100** |
| 0.72 | **35** | **8000** | **0.03** | **0.2358** | **99** |
| 0.78 | **35** | **8000** | **0.06** | **0.2069** | **100** |
| 0.84 | **35** | **8000** | **0.05** | **0.2476** | **98** |
| 0.90 | **35** | **8000** | **0.03** | **0.2292** | **100** |

Buscamos minimizar el error topológico y el error de cuantificación lo máximo posible y comprobar el número de clases obtenidas mediante una gráfica:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Teniendo ya elegidos dos de los tres valores, lado\_mapa como 35 y periodo como 8.000, ahora queda ir iterando hasta obtener el eta que mejor se ajuste.

Hemos elegido un valor de 0.4 para el eta, primero elegimos de entre los que tienen un alto número de clases, observamos varios valores entre 98y 100 clases, descartamos los anómalos rápidamente.

En función de los errores, ambos valores (error topológico y de cuantificación), son de los más bajos obtenidos lo que es exactamente lo que buscamos al realizar estas pruebas.

Texto

Descripción generada automáticamente

*.*

# 1.2.2 Datos de la mejor clasificación

# 1.2.2.1 Gráfico RGB de pesos iniciales

Gráfico

Descripción generada automáticamente

Este gráfico es la muestra del mapa antes de ser entrenado, cada color es una de las neuronas con su vector de pesos (creado aleatoriamente) sin ajustar.

# 1.2.2.2 Gráfico RGB de los pesos entrenados

Gráfico

Descripción generada automáticamente

En esta imagen se puede apreciar como el mapa de Kohonen queda ordenado por colores tras haber sido entrenado 8000 veces, utilizando patrones de entrada generados de manera aleatoria.

# 1.2.2.3 Número de clases, error topológico y error de cuantificación

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Nos quedamos con esta opción, de entre varias muy similares, pues posee un número de clases óptimo, así como una relación de errores lo suficientemente baja como para ser un gran candidato.

# 1.2.2.4 Mapa de clasificación

Gráfico

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente

# 1.2.2.5 Mapa de activaciones

Podemos observar como las 100 clases han sido activas en sus diferentes coordenadas de BMU

Gráfico, Histograma

Descripción generada automáticamente

# 1.2.2.6 Mapa de distancias

Imagen que contiene Gráfico de dispersión

Descripción generada automáticamente

# 1.2.3 Dataset de prueba

Gráfico de dispersión

Descripción generada automáticamente con confianza media

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamenteObservamos que existe una clase menos de las esperadas, nos encontramos con un error con el color negro o [0, 0, 0] en RGB, el problema lo solucionamos sustituyendo por números muy cercanos a 0 como es 0.0001. A la hora de formar las clases nuestra red considera similares al [0, 0, 0] y al [255, 255, 255] por eso se ahorra una.

Imagen que contiene Calendario

Descripción generada automáticamenteImagen que contiene Gráfico de dispersión

Descripción generada automáticamente

Imagen que contiene Tabla

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

# 2 Práctica 2: Clasificar países por nivel de desarrollo

En esta segunda práctica, usaremos el programa SOM desarrollado en la Práctica 1 para entrenar una red que clasificará 17 características de 171 países, resultado de mediciones del año 2010.

# 2.1 Implementación

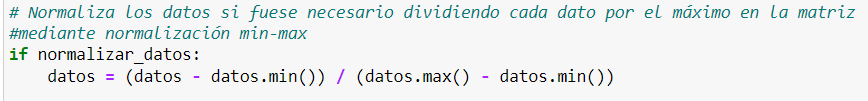
# 2.1.1 Obtención del Dataset

A la hora de observar el dataset, vamos a prescindir de la columna que guarda el nombre de cada país, el resto contiene información relevante para el estudio comparativo que queremos llevar a cabo.

Imagen que contiene Texto

Descripción generada automáticamente

Usando el método *drop* prescindimos de una columna especificando el nombre y con el método *tail* mostramos el final del dataset, dónde encontramos el nombre de Venezuela, RB, que nos dará problemas en el futuro pues no sabíamos como estaba escrito en el dataset dado.



Hemos usado una normalización diferente a la de la primera práctica 1, al tener valores tan dispares hemos optado por usar la función *min-max*.

# 2.1.2 Entrenamiento

Hemos usado el mismo entrenamiento, como pide el enunciado.

# 2.1.3 Modificación del mapa de clasificación

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

Imagen que contiene Texto

Descripción generada automáticamente

Para la clasificación hemos llevado a cabo el mismo proceso, salvo que, hemos creado una matriz en la que insertamos las coordenadas de la BMU transformadas mediante un vector que contiene todos los nombres de los países en la matriz *mat\_name*.

Por otro lado, hemos creado una matriz de colores en la cual se mostraron las coordenadas de la BMU convertidas en colores, de hexadecimal a RGB (los colores asignados a cada país son aleatorios).

Texto

Descripción generada automáticamente con confianza media

Con el cambio de países por colores, así como la obtención de la letra inicial, no podemos sacarle mucho uso ya que los colores son asignados de manera aleatoria y las iniciales de los países pueden coincidir, la virtud de este cambio sale a relucir en una de las cuestiones, dónde, al ser sólo 3 países, distinguimos de manera muy sencilla por su inicial, sacando así uso a este sistema.

Otro de los cambios realizados por sucesivos errores ha sido el cambio de como buscamos la bmu:

Texto

Descripción generada automáticamente

Con np.argwhere() como usábamos en la clasificación anterior, nos daba los mismos patrones de coordenadas de BMU o problemas con los tipos de variables, por tanto, buscamos otro método para lograr lo mismo.

# 2.2 Cuestiones

# 2.2.1 Valores de Lado\_Mapa, Periodo y Eta más adecuados.

Estas son las gráficas que hemos realizado para saber que lado de mapa, periodo y learning rate debíamos ejecutar la práctica.

Texto

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Con estas tres gráficas hemos concluido que la mejor configuración es:

Texto

Descripción generada automáticamente

# 2.2.2 Resultados elegidos

Texto

Descripción generada automáticamente

Nos quedamos con esta opción, de entre varias muy similares, pues posee un número de clases óptimo, así como una relación de errores lo suficientemente baja como para ser un gran candidato.

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza baja

Gráfico, Histograma

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

# 2.2.3.1 ¿Qué grupos se pueden establecer?

Como hemos realizado el entrenamiento y clasificación con los 17 parámetros de cada uno de los países observamos que todos están muy próximos. Esto se podría solucionar utilizando los parámetros más característicos del dataset con algún método como k-means.

Además cuanto mayor sea el lado del mapa, mejor se observarían estos grupos, pero viendo el rendimiento y tiempo de ejecución de la práctica era inviable y no nos daba tiempo para realizar dicha comprobación.

# 2.2.3.2 Análisis de los vectores de pesos de Venezuela, Noruega y Etiopía.

Para los 3 países la clasificación ha dado los siguientes resultados:  
Imagen que contiene Logotipo

Descripción generada automáticamente

Gráfico, Gráfico radial

Descripción generada automáticamente

Contestación junto con el siguiente apartado

# 2.2.3.3 Mapa de clasificación.

Realizamos la clasificación con unos datos de entrada que corresponden a el *GPD per capita* y el *mortality rate neonatal*; Normalizamos usando el mismo método anterior (min-max) y seguimos con la clasificación.

La diferencia frente a la de colores es que en la clasificación hemos creado un mapa de colores y uno de nombres con el fin de hacer representaciones visuales más intuitivas.

Para este caso hemos utilizado únicamente los países del enunciado ya que, al usar los 171, como hemos dicho anteriormente, la asignación aleatoria de colores no nos ayudaría, así como la confusión resultante de usar solamente la letra inicial.

Además, hemos realizado este apartado conjuntamente con el anterior; por tanto hemos mostrado la relación que tienen Venezuela, Noruega y Etiopia en base a su *GPD per capita* y el *mortality rate neonatal.* Este seria el mapa de nombres que mostraría la primera inicial del país en la matriz y el mapa de colores mostrando cada uno de los países.

Observamos que Venezuela es el que más alejado se encuentra, no existen similitudes entre los países, cada uno se encuentra en un lugar del mapa, esto se puede deber a la relación de natalidad y el número de habitantes que pensamos que son dos de los factores que más influyen en este estudio.

Hemos comprobado que el color salmon corresponde a Venezuela, el azul a Etiopia y el azul clarito a Noruega.

Gráfico

Descripción generada automáticamente

Calendario

Descripción generada automáticamente

# 2.2.3.4 Ubicaciones de los países subdesarrollados.

Para comprobar este experimento, hemos usado una lista de países que ya sabemos si son, o no, considerados subdesarrollados (Haiti, Etiopia, Yemen, Bangladesh y Nepal), así como un grupo de países considerados desarrollados (Estados Unidos, Japan, Alemania, Inglaterra y Canada).

Como podemos observar en la clasificación se dividen en dos grupos los países dándonos a entender que por un lado están los subdesarrollados y en su opuesto los desarrollados.

Hemos comprobado debugueadon que los países están siendo colocados correctamente en función de sus colores. A continuación, mostraremos la clasificación para los datos de entrada de países descritos en el apartado:

Texto

Descripción generada automáticamente con confianza baja

Gráfico de dispersión

Descripción generada automáticamente con confianza mediaGráfico

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja