Aprendizaje automático. Clustering.

Datos del proyecto:

- Asignatura: Programación Declarativa.
- Titulación: Doble Grado en Ingeniería Informática y Matemáticas.
  Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla.
- Profesores: Miguel Ángel Martínez del Amor.
  David Solís Martín.
- Lenguaje de programación utilizado: Haskell.
- Contenido: Aprendizaje automático no supervisado. Librería de algoritmos de clustering, tratamiento de datasets y representación gráfica.

Autoría del proyecto:

- Pablo Reina Jiménez — Datos de contacto: pabreijim1, pabreijim1 ARROBA alum.us.es
- María Lourdes Linares Barrera — Datos de contacto: marlinbar, marlinbar ARROBA alum.us.es

ÍNDICE DE CONTENIDOS

TEMÁTICA ELEGIDA

La temática elegida para el proyecto es el análisis de datos mediante técnicas de aprendizaje automático no supervisado y la posterior representación de los resultados obtenidos.

Librería de algoritmos de clustering

Hemos decidido implementar una librería que proporcione una amplia cobertura de los principales algoritmos de clustering: **clustering de partición estricta** (conocido como k-medias) y **clustering jerárquico aglomerativo**. En este segundo caso, el modelado del algoritmo se ha realizado desde dos enfoques diferentes. La librería también permite al usuario seleccionar la **función de distancia** que desea utilizar según la naturaleza de las conclusiones que desee extraer.

Técnicas de representación gráfica

Finalmente, se han implementado distintas formas de visualización de los resultados. Para el algoritmo de k-medias, el usuario puede decidir si quiere visualizar únicamente los centros de los clusters o la composición completa de los mismos. Además, en el caso de que el dataset cuente con datos 2-dimensionales, hemos implementado la funcionalidad que permite visualizar gráficamente los clusters (en colores diferenciados) y los centros de los mismos, utilizando el **api de codeworld**.

Por su parte, la visualización de los resultados de la aplicación de técnicas de clustering jerárquico aglomerativo van desde una lista de evolución de los clusters, una representación jerárquica en forma de árbol normal (tal y como hemos visto en las lecciones de la asignatura) a una visualización más precisa de esta jerarquía, mediante un árbol escalonado (utilizando la **librería Data.Tree)**.

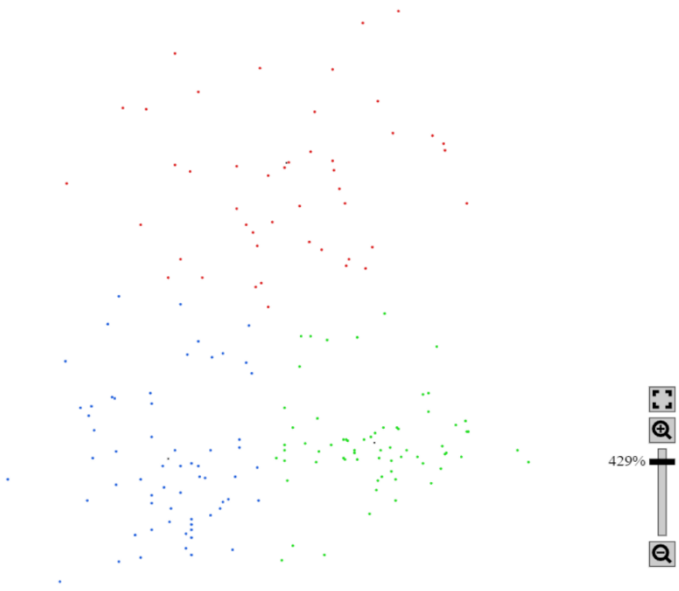

Ilustración 1. Aplicación de k-means sobre wine_clustering.csv

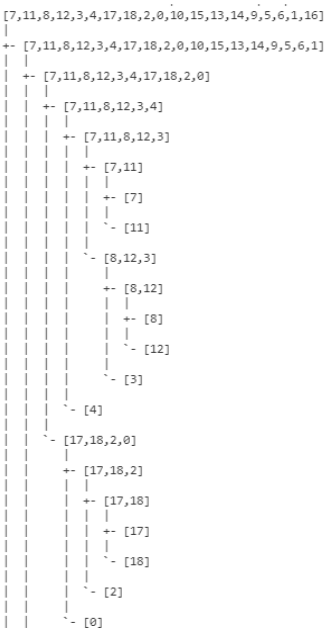

Ilustración 2. Clustering aglomerativo sobre user_knowledge.csv (vista parcial)

Análisis de datos: datasets.

La fuente de la que hemos obtenido los datos sobre los que trabajar ha sido la página de Kaggle. La comunidad ofrece un amplio repositorio de datasets de temáticos y de contenido muy diverso.

ESTRUCTURACIÓN DEL PROYECTO. REQUISITOS.

El proyecto se estructura como se muestra a la derecha. A continuación, se pasa a describir el contenido de cada uno de los módulos y carpetas del proyecto, las funciones o ficheros que los integran y los requerimientos que satisfacen.

Observación: Al final de los módulos se adjuntan ejemplos que permiten hacer pequeñas pruebas unitarias si el corrector lo estimara oportuno. Como complemento a las pruebas sobre los datasets.

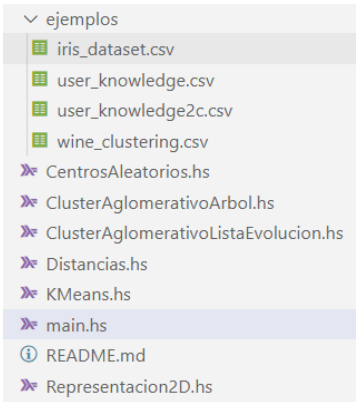

Ilustración 3. Estructura del proyecto

Módulo para distancias

El módulo del proyecto asociado es **Distancias.hs**. Este módulo integra funcionalidades que resultan de utilidad en el resto del proyecto: tipos de datos comunes y funciones distancia (entre las que el usuario podrá elegir cual utilizar para el clustering).

Módulos y librerías utilizadas (l. 24-27)

Se ha utilizado la librería **Data.Array**.

Definición de tipos (l. 29-41)

```
type Vector = Array Int Double                  -- Tipo para los datos del dataset
type Cluster = [Vector]                         -- Tipo para los clusters
type Distancia = Vector -> Vector -> Double     -- Tipo para funciones de distancia
```

Lista de funciones del módulo (l. 44-87)

```
listaVector :: [Double] -> Vector     -- Transforma una lista de valores en un vector.
-- Funciones distancia :: Vector -> Vector -> Double. Distancia entre dos vectores.
distEuclidea :: Distancia             -- Distancia Euclídea
distManhattan :: Distancia            -- Distancia Manhattan
distHamming :: Distancia              -- Distancia Hamming
```

Módulos para k-medias

En esta sección se recogen los módulos integran el algoritmo de kMeans y permite la representación de los resultados por consola o de forma gráfica. Los módulos asociados son:

- **CentrosAleatorios**: Módulo que permite generar los centros aleatorios que inicializan el algoritmo.
- **KMeans:** Módulo que implementa el algoritmo.
- **Representacion2D:** Módulo que implementa la representación gráfica de los clusters en casos 2-dimensionales.

Generación de centros aleatorios

Módulos y librerías utilizadas (l. 17-23)

Se han utilizado varias librerías, entre las destacan: Distancias(+Data.Array), **System.Random** y Data.List.

Lista de funciones del módulo (l. 25-100)

```
listaMinMax :: [Vector] -> [(Double, Double)]
-- Dada una lista de puntos, devuelve una lista de pares (mínimo, máximo)
      minMaxCoord              -- Devuelve el par (mínimo, máximo) de una coordenada
      extraeValoresCoord       -- Extrae los valores que toma una coordenada
generaCentros :: Int -> [Vector] -> IO [Vector] -- Genera una lista de vectores
aleatorios
      listaVectorIO            -- Transforma una lista en un tipo Vector
      generaUnNumero           -- Genera una lista de números aleatorios (un vector)
```

Algoritmo de kMeans

Módulos y librerías utilizadas (l. 22-29)

Se han utilizado muchas librerías, entre las destacan: Distancias (+Data.Array), **CentrosAleatorios**, **System.Random** y Data.List.

Lista de funciones del módulo (l. 31-174)

```
kMeans :: Int -> [Vector] -> Distancia -> IO [Vector]
-- Genera los centros finales del algoritmo
kMeansCompleto :: Int -> [Vector] -> Distancia -> IO [(Vector, Vector)]
-- Genera la asociación de vectores iniciales y centros finales
kMeansAux :: [Vector] -> [Vector] -> Distancia -> [Vector]
-- Función recursiva del algoritmo. Para cuando los centros de 2 iteraciones coinciden
getNewM :: [(Vector, Vector)] -> [Vector] -> [Vector]
-- Genera los nuevos centros
calculaMediaM :: [(Vector, Vector)] -> Vector -> Vector
-- Calcula los nuevos centros en función de los puntos asociados a cada cluster
      calculaMediaMAux         -- Función auxiliar de calculaMediaM
asocXM :: [Vector] -> [Vector] -> Distancia -> [(Vector,  Vector)]
-- Asocia a cada punto su centro más cercano
      asocXMAux                -- Función auxiliar de asocXM
      getMinDist               -- Calcula el centro más cercano a un punto
      fstTuple                 -- Comparador por el primer elemento de la tupla
```

Representación 2D

Módulos y librerías utilizadas (l. 17-22)

Se han utilizado muchas librerías, entre las destacan: Distancias (+Data.Array), **Codeworld** (ocultando el tipo Vector para evitar conflicto con el tipo definido).

Lista de funciones del módulo (l. 24-97)

```
dibuja :: [Vector] -> [(Vector, Vector)] -> IO ()
-- Representa los puntos y los centros de los clusters
representaCentros :: [([Double], Color)] -> Picture
-- Representa los centros de los clusters
      dibujaCentro             -- Representa un centro
representasClusters :: [(Vector, Vector)] -> [([Double], Color)] -> Picture
-- Representa los puntos asociados a un cluster
      dibujaPunto              -- Representa un punto en concreto
asociaColorACluster :: [Vector] -> [Color] -> [([Double], Color)]
-- Asocia un color a los puntos de un cluster
      buscaColor               -- Dado un punto, obtiene su color asociado
      listaColores :: [Color]  -- Lista de colores (tipo Color de CodeWorld)
```

Módulos para clustering aglomerativo

El algoritmo comienza con un cluster por cada punto. En las sucesivas iteraciones, se toman los dos clusters más próximos y se fusionan. El algoritmo finaliza cuando todos los puntos pertenecen al mismo cluster. El modelado de este algoritmo se ha materializado en dos implementaciones (módulos):

- **ClusterAglomerativoArbol**: Módulo que implementa el algoritmo modelándolo mediante un dendrograma (bosque de árboles que refleja cómo se van fusionando los clusters).
- **ClusterAglomerativoListaEvolucion**: modelado mediante una lista de evolución.

Clustering aglomerativo (modelo árbol)

Módulos y librerías utilizadas (l. 26-31)

Se han utilizado varias librerías, entre las destacan: Distancias(+Data.Array), **Data.Maybe** y Data.List.

Definición de tipos (l. 34-57)

```
type IdCluster = [Int]          -- Identificador de un cluster en el árbol
                                -- (Índices de los subclusters que contiene)
data Arbol = H IdCluster Cluster | N IdCluster Cluster Arbol Arbol
    deriving Eq                 -- Tipo "árbol" (como se van uniendo los clusters)
instance Show Arbol where ...
type Dendrogram = [ Arbol ]     -- Tipo "bosque".
```

El concepto de bosque es una generalización del concepto de árbol. En el algoritmo, partimos que vectores que forman hojas sin emparejar. Conforme se van fusionando los clusters se van formando árboles en paralelo (bosque). Hasta el final no se obtiene un único árbol.

Lista de funciones "herramienta" del módulo (l. 59-111)

```
listaClustersActuales(2) :: Dendrogram -> [(IdCluster, Cluster)] ([Cluster])
-- Obtiene el ultimo estado de los clusters de un dendrograma.
arbolAsociadoACluster :: Dendrogram -> Cluster -> Maybe Árbol
-- Devuelve el árbol del dendrograma asociado a un cluster.
      datosClusterFromArbol          -- Devuelve el cluster asociado a un árbol
```

Lista de funciones "algorítmicas" del módulo (l. 114-275)

```
inicializaClusteringAglomerativoA :: [Vector] -> Dendrogram
-- Obtiene el primer dendrograma a partir de los datos (cada vector es una hoja).
clusteringAglomerativoA :: Distancia -> Dendrogram -> Arbol
-- Función base del algoritmo de clustering: obtiene un árbol que refleja cómo se han
ido fusionando los clusters. Aplica iteraciones del algoritmo.
calculaSiguienteNivel :: Distancia -> Dendrogram -> Dendrogram
-- Toma un dendrograma y fusiona los dos clusters más cercanos.
      eliminaCluster                 -- Elimina un cluster del dendrograma
clustersDistanciaMinima :: Distancia -> [Arbol] -> ((Arbol, Arbol), Double)
-- Devuelve el par (2 clusters más cercanos, distancia entre ellos)
      sndTuple                       -- Comparador por el segundo elemento de la tupla
calculaMatrixProximidad :: Distancia -> [Arbol] -> [((Arbol, Arbol), Double)]
-- Matriz simetrica que devuelve la distancia entre dos clusters cualesquiera
      calculaDistanciasAUnCluster    -- Distancia de todos los clusters a uno
      distanciaEntreClusters         -- Distancia entre dos clusters
      calculaMedia                   -- Calcular el punto medio de un cluster
      calculaMediaAux                -- Función auxiliar de calculaMedia
```

Clustering aglomerativo (modelo lista de evolución)

Módulos y librerías utilizadas (l. 24-30)

Se han utilizado varias librerías, entre las destacan: Distancias(+Data.Array), **Data.Matrix** y Data.List.

Definición de tipos (l. 32-48)

```
type Nivel = (Int, [Cluster], Int)     -- (numIter, listClusters, numClusters)
                                       -- Tipo para los clusters en una iteracion
type EvolucionClusters = [Nivel]       -- Tipo evolución de los clusters
```

El modelado mediante lista consiste en que cada elemento de la lista (Nivel) recoge el resultado de una interacción del algoritmo, es decir, los clusters formados en la interacción i-ésima.

Lista de funciones "algorítmicas" del módulo (l. 51-247)

```
inicializaClusteringAglomerativoLE :: [Vector] -> EvolucionClusters
-- Obtiene el primer nivel a partir de los datos (cada vector forma un cluster).
clusteringAglomerativoLE :: Distancia -> EvolucionClusters -> EvolucionClusters
-- Función base del algoritmo de clustering: obtiene la evolución de
-- la lista de clusters. Aplica iteraciones y va actualizando la lista de evolución.
calculaSiguienteNivel :: Distancia -> Nivel -> Nivel
-- Toma un nivel, fusiona los clusters más cercanos y devuelve el siguiente nivel.
      eliminaCluster     -- Elimina un cluster
clustersDistanciaMinima :: Distancia -> [Cluster] -> ((Cluster, Cluster), Double)
-- Devuelve el par (2 clusters más cercanos, distancia entre ellos)
    sndTuple                          -- Comparador por el segundo elemento de la tupla
transformaMatriz :: Matrix Double -> [Cluster] -> [((Cluster, Cluster), Double)]
-- Transforma la matriz de distancias en una lista de tuplas con la distancia entre
clusters
calculaMatrixProximidad :: Distancia -> [Cluster] -> Matrix Double
-- Matriz simetrica que devuelve la distancia entre dos clusters cualesquiera
    calculaMatrixProximidadAux       -- Función auxiliar de calculaMatrixProximidad
    recalculaMatriz                  -- Actualiza la matriz con distancias obtenidas
    calculaDistanciasAUnCluster      -- Distancia de todo los clusters a uno
    distanciaEntreClusters           -- Distancia entre dos clusters
    calculaMedia                     -- Calcular el punto medio de cada cluster
    calculaMediaAux                  -- Función auxiliar de calculaMedia
```

Main

El archivo asociado es **main.hs**. Ejecutando este código se nos mostrará por línea de comandos un menú en el que podremos ir navegando por los distintos algoritmos que ofrece el proyecto.

Módulos y librerías utilizadas (l. 13-34)

Entre las librerías utilizadas destacan: Distancias, KMeans, Representacion2D, ClusterAglomerativoListaEvolucion, ClusterAglomerativoArbol, **Data.Maybe**, **Data.Tree,** Data.List, Data.Char, **System.IO, Text.CSV, System.Directory,** Debug.Trace y Data.Typeable.

Definición de tipos (l. 36-48)

```
data Dataset4Clustering = Dataset4Clustering {nombre :: String, cabecera ::
[String], datos :: [Vector] } deriving (Show, Eq)
-- Tipo dataset, procesamiento de los datos leídos.
```

Lista de funciones del módulo (l. 50-457)

```
main :: IO ()
-- Encargada de solicitar el nombre del dataset y de su lectura
```

```
      parseadorCSV              -- Transforma el dataset al tipo Dateset4Clustering
      fila2Array                -- Transforma las filas del dataset al tipo Vector
      seleccionAlgoritmo        -- Solicita el algoritmo que se desea usar
      seleccionaDistancia       -- Solicita la distancia que se desea usar
algKMeans :: [Vector] -> Distancia -> Int -> IO ()
-- Ejecuta el algoritmo de kMeans y representa los resultados
      algKMeansDistancia        -- Solicita la distancia para el algoritmo kMeans
      algKMeansCentros          -- Solicita el número de centros para el algoritmo
      obtieneCentros            -- Obtiene los centros de una lista de asociados
      compruebaEntero           -- Comprueba si un String puede ser parseado como Int
      representaKmeansCompleto -- Representa los puntos asociados a cada centro
      representaAsociadosAM     -- Representa los puntos asociados a un centro
      representa                -- Representa los puntos y centros de clusters
clustAglomerativo :: Cluster -> Distancia -> IO ()
-- Ejecuta el algoritmo de cluster aglomerativo y representa los resultados
      clustAglomerativoDistancia -- Solicita la distancia para el algoritmo
representaClusterAglomerativoLE :: [(Int, [Cluster], Int)] -> IO ()
-- Representa el cluster aglomerativo mediante una lista de evolución
      representaUncluster       -- Representa un nivel concreto
      fst' (a,_,_) = a          -- Funciones usadas para extraer los elementos
      snd' (_,a,_) = a          -- de tuplas de tres elementos
      thr' (_,_,a) = a
clustAglomerativoArbol :: Distancia -> Cluster -> IO ()
-- Representa el cluster aglomerativo mediante un dendrograma
      representaArbol           -- Representa los resultados en forma de árbol
      toDataTreeId              -- Transforman nuestra estructura de datos
      toDataTreeCl              -- a una del tipo Data.Tree
```

Ejemplos

En la carpeta ejemplos se encuentran los datasets recomendados para visualizar el funcionamiento de los distintos algoritmos.

Dataset	Información
iris_dataset.csv	Contiene información acerca de características del sépalo y pétalo de las flores Iris. El objetivo es aplicar clustering para distinguir las distintas especies de Iris.
user_knowledge.csv	Recoge distintas métricas respecto al estudio de los alumnos y sus resultados académicos. El objetivo es distinguir distintos grupos en el alumnado.
user_knwoledge2c.csv	Versión 2-dimensional del dataset anterior.
wine_clustering.csv	Este dataset 2-dimensional recoge los valores de distintos tipos de vinos. El objetivo es aplicar clustering para distinguir distintos grupos entre los vinos dados.

Observaciones:

1) ***Para el testeo del algoritmo de k-medias con representación gráfica se recomienda utilizar los datasets 2-dimensionales (i.e. user_knowledge2c.csv o wine_clustering.csv).***
2) ***Para la correcta visualización del árbol se recomienda utilizar datasets de tamaño más reducido (i.e. user_knowledge.csv).***

3) ***En cualquier otro caso (clutering aglomerativo por lista de evolución, k-medias con resultados por consola…), los resultados son igualmente interpretables utilizando cualquier dataset.***

Ejemplo 1 (Clustering Aglomerativo versión árbol):

```
Introduce el nombre del fichero: ejemplos/user_knowledge.csv
Seleccione el algoritmo a usar: kMeans (KM), clusterAglomerativo (CA): CA
Indique el tipo de distancia a utilizar: Euclidea (DE), Manhattan (DM) o Hamming (DH):
DE
Seleccion el tipo de estructura de datos: listaEvolucion (LE), Arbol (A): A
Seleccione la forma de representacion por pantalla: arbol de id (AI), arbol de clusters
(AC), normal (N): AI
```

Ejemplo 2 (KMeans):

```
Introduce el nombre del fichero: ejemplos/wine_clustering.csv
Seleccione el algoritmo a usar: kMeans (KM), clusterAglomerativo (CA): KM
Indique el tipo de distancia a utilizar: Euclidea (DE), Manhattan (DM) o Hamming (DH): DE
Indique el numero de centros para el algoritmo (menor que 10 si desea una representacion grafica):
3
Indique que datos desea extraer: unicamente los centros de los clusters (M), centros y datos
asociados a cada uno (CM): M
¿Quiere una representacion grafica de los puntos: SI (S), NO (N)? S (abrir link)
```

Las salidas gráficas de estos dos tests pueden observarse en la primera página de la memoria (temática). Más ejemplos en README.md.

Cumplimiento de requisitos

Debido a la extensión y complejidad del trabajo desarrollado, los requisitos mínimos se satisfacen, así como el número mínimo de uso de determinados conceptos es sobrepasado. Nos limitamos por tanto a continuación a mencionar algunas funciones o situaciones en los que se han utilizado, pudiendo observarse muchos más usos de todos ellos en el código fuente del proyecto.

Conceptos y requisitos	**Algunos ejemplos de uso**
Funciones básicas del Prelude	- Uso de sqrt en distEuclidea (Distancias.hs) - Uso de abs en distHamming (Distancias.hs) - Uso de fromIntegral en calculaMediaMAux (KMeans.hs) - Uso de fst, snd en calculaMediaMAux (KMeans.hs) …
Funciones de Data.List	- Uso de nub en obtieneCentros (main.hs) - Uso de sortBy en clustersDistanciaMinima (clusterAglomerativoArbol.hs) - Uso de tail en asociaColorACluster (Representacion2D.hs) - Uso de minimum (maximum) en minMaxCoord (CentrosAleatorios.hs) - Uso de zip en fila2Array (main.hs) …
Declaración de tipos	- Todas las funciones con su signatura.
Guardas	Muy utilizado a lo largo de todo el proyecto. - listaMinMax, generaCentros y generaUnNumero (CentrosAleatorios.hs) - representaArbol (main.hs) …
Patrones	Muy utilizado a lo largo de todo el proyecto.

	- listaClustersActuales, arbolAsociadoACluster (ClusterAglomerativoArbol.hs) - datosClusterFromArbol (ClusterAglomerativoArbol.hs) …
Case of	- parseadorCSV (main.hs) - seleccionaDistancia (main.hs) …
If then else	- Utilización en numerosas funciones de main.hs - calculaMediaMAux (KMeans.hs) …
Recursividad	- kmeansAux (KMeans.hs) - representaClusters (representacion2D.hs) …
Listas por comprensión	- calculaMediaAux (ClusterAglomerativoArbols.hs) - inicializaClusteringAglomerativoA (ClusterAglomerativoArbol) - inicializaClusteringAglomerativoLE (ClusterAglomerativoListaEvolucion.hs) …
Funciones de orden superior	- Uso de foldr en listaMinMax (CentrosAleatorios.hs) - Uso de takeWhile en eliminaCluster (ClusterAglomerativoArbol.hs) - Uso de map en extraeValoresCoord (CentrosAleatorios.hs) - Uso de filter en parseadorCSV (main.hs) …
Creación de módulos	Creación de 6 módulos: - Distancias - CentrosAleatorios, KMeans, Representacion2D - ClusterAglomerativoArbol, ClusterAglomerativoListaEvolucion …
Tipos de datos algebraicos y árboles	- tipos Vector, Cluster, Distancia (Distancias.hs) - tipos Árbol, Dendrograma (bosque), IdCluster (ClusterAglomerativoArbol.hs) - tipos Nivel, EvolucionClusters (ClusterAglomerativoListaEvolucion.hs) - sintaxis de registro tipo Dataset4Clustering (main.hs) …
Tipos de datos abstractos	- Uso de Array (tipo Vector). Ejemplo: dibujaPunto (Representacion2D.hs) - Matrix en calculaMatrixProximidad (ClusterAglomerativoListaEvolucion.hs) …
Data.Maybe	- Usando en arbolAsociadoACluster (ClusterAglomerativoArbol.hs) - Usado en seleccionaDistancia (main.hs) …
Entrada-Salida	- Todas las funciones de main.hs
Representación gráfica	- Usando Codeworld (main.hs, Representacion2D.hs) - Usando Data.Tree (main.hs) …
Evaluación perezosa	- calculaMatrixProximidad (ClusterAglomerativoLE.hs) - compruebaEntero (main.hs) …
Gestión de errores	- representaArbol (main.hs) Por si el modo introducido no es válido - buscaColor (representacion2D.hs) Por si la lista de colores es vacía …

Observación: La descripción que se ha realizado de los módulos y requisitos es somera y esquemática debido a la limitación de espacio. Si se desea conocer más detalles sobre las funciones descritas y la utilización de los conceptos, puede recurrir al código donde está comentado en detalle.

LANZAMIENTO DEL PROYECTO

Disponemos de dos opciones para el lanzamiento del proyecto. La primera, es la **opción interpretada**. Debemos abrir el fichero main.hs y hacer Load GHCi (ghci "ruta del archivo"). Posteriormente escribimos por línea de comandos main y el programa comenzará a ejecutarse.

```
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Pablo\OneDrive - UNIVERSIDAD DE SEVILLA\Universidad\4º\Cuatrimestre1\PD\Trabajo PD\Trabajo-Haskell>ghci "c:\Users\Pablo\OneDrive - UNIVERSIDAD DE SEVILLA\Universidad\4º\Cuatrimestre1\PD\Traba
jo PD\Trabajo-Haskell\main.hs"
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 7] Compiling Distancias       ( Distancias.hs, interpreted ) [Data.Array changed]
[2 of 7] Compiling ClusterAglomerativoListaEvolucion ( ClusterAglomerativoListaEvolucion.hs, interpreted ) [Data.Matrix changed]
[3 of 7] Compiling ClusterAglomerativoArbol ( ClusterAglomerativoArbol.hs, interpreted ) [Data.Array changed]
[4 of 7] Compiling CentrosAleatorios ( CentrosAleatorios.hs, interpreted ) [System.Random changed]
[5 of 7] Compiling KMeans           ( KMeans.hs, interpreted ) [System.Random changed]
[6 of 7] Compiling Representacion2D ( Representacion2D.hs, interpreted ) [CodeWorld changed]
[7 of 7] Compiling Main             ( C:\\Users\Pablo\OneDrive - UNIVERSIDAD DE SEVILLA\Universidad\4º\Cuatrimestre1\PD\Trabajo PD\Trabajo-Haskell\main.hs, interpreted ) [flags changed]
Ok, 7 modules loaded.
*Main> main

--------------------------------------
       ELECCION DEL DATASET
--------------------------------------

Introduce el nombre del fichero:
```

Ilustración 4. Lanzamiento con intérprete

En segundo lugar, tenemos la **opción compilada** (es necesario usar Linux). Para esta opción debemos escribir lo siguiente por línea de comandos ghc main.hs -o main. Luego simplemente debemos llamar a este nuevo ejecutable main y comenzará la ejecución del programa.

```
preinaj@DESKTOP-9H2F9P3:/mnt/c/Users/Pablo/OneDrive - UNIVERSIDAD DE SEVILLA/Universidad/4º/Cuatrimestre1/PD/Trabajo PD/Trabajo-Haskell$ ghc main.hs -o main
[1 of 7] Compiling Distancias       ( Distancias.hs, Distancias.o )
[2 of 7] Compiling ClusterAglomerativoListaEvolucion ( ClusterAglomerativoListaEvolucion.hs, ClusterAglomerativoListaEvolucion.o )
[3 of 7] Compiling ClusterAglomerativoArbol ( ClusterAglomerativoArbol.hs, ClusterAglomerativoArbol.o )
[4 of 7] Compiling CentrosAleatorios ( CentrosAleatorios.hs, CentrosAleatorios.o )
[5 of 7] Compiling KMeans           ( KMeans.hs, KMeans.o )
[6 of 7] Compiling Representacion2D ( Representacion2D.hs, Representacion2D.o )
[7 of 7] Compiling Main             ( main.hs, main.o )
Linking main ...
preinaj@DESKTOP-9H2F9P3:/mnt/c/Users/Pablo/OneDrive - UNIVERSIDAD DE SEVILLA/Universidad/4º/Cuatrimestre1/PD/Trabajo PD/Trabajo-Haskell$ ./main

--------------------------------------
       ELECCION DEL DATASET
--------------------------------------

Introduce el nombre del fichero:
```

Ilustración 5. Compilación

Las **librerías utilizadas** se han ido detallando a lo largo de la memoria. Entre ellas, puede ser necesario que instale csv, matrix, array, random y codeworld-api.

Observación: Se recomienda consultar README.md donde se explica cómo llevar a cabo las instalaciones y se recogen varios ejemplos de uso y una guía de como navegar por el menú de opciones que ofrece el programa.

BIBLIOGRAFÍA

Gutiérrez Naranjo, M.A., Martín Mateos, F.J, Ruiz Reina, J.L. (2020). *Tema 6: Introducción al aprendizaje automático*. Dpto. Ciencias de la Computación e Inteligencia Artificial

Martínez del Amor, M.A., Solís Martín, D. (2021). *Temario asignatura Programación Declarativa.* Dpto. Ciencias de la Computación e Inteligencia Artificial. Disponible en: https://www.cs.us.es/cursos/pd/Contents.html#temas

Google LLC. *Kaggle Datasets*. user_knowledge.csv, iris_dataset.csv, wine_clustering.csv. Disponible en: https://www.kaggle.com/datasets

Haskell Library Database & Documentation. *Hackage*. Disponible en: https://hackage.haskell.org/

Alonso Jiménez, J.A. (2018). *Manual de la librería de matrices Data.Matrix*. Disponible en: https://www.cs.us.es/~jalonso/cursos/i1m-17/doc/manual-Data.Matrix.html