Módulo 4. Aprendizaje no supervisado

Video de inmersión

Unidad 1. *K-Means*

Tema 1. ¿Qué es un modelo no supervisado?

Con los 2 módulos anteriores de nuestro curso, hemos aprendido acerca de modelos de aprendizaje supervisado para diversas aplicaciones de machine learning y su uso en diversos ámbitos de trabajo. Sin embargo, no podemos dejar de lado el aprendizaje no supervisado, el cual también tiene un potencial en distintos campos de aplicación dentro de diversas industrias.

Recordemos algunos puntos que anteriormente tratamos acerca del aprendizaje no supervisado.

Figura 1. Frutas y verduras

Fuente: Hogarmania, s.f., http://bit.ly/3fsfAp3

Cuando éramos niños, comíamos arvejas, pepino y otras verduras de color verde y era probable que no nos gustasen. No es que nuestros padres nos decían que no iban a gustarnos los vegetales verdes en su totalidad, esto dependía de nuestros aprendizajes

y gustos. Es decir, si nuestro propio paladar nos decía: “Todas aquellas verduras que tengan color verde no te van a gustar”, era nuestra etiqueta.

De esta manera, los algoritmos o modelos intentan darles sentido a los datos por sí mismos mediante la búsqueda de patrones y características que sean semejantes a la observación de los datos.

Ahora, veamos una definición más formal acerca del aprendizaje no supervisado.

Aprendizaje no supervisado

Dentro del *machine learning*, nos enfrentaremos a problemas en los que los datos disponibles consisten en ejemplos **sin etiquetar**. Lo que significa que cada punto de datos contiene características únicamente, sin una etiqueta asociada o, en otras palabras, sin una variable de salida definida. El objetivo de los algoritmos de aprendizaje no supervisado es aprender patrones útiles o propiedades estructurales de los datos.

A diferencia con el aprendizaje supervisado donde los datos son etiquetados por una persona, por ejemplo, la muerte de una persona del Titanic o si el animal tratado en una veterinaria es “perro” o “gato”, los métodos no supervisados exhiben patrones de conocimiento donde se generan agrupaciones según sus comportamientos.

Algunos ejemplos de tareas de aprendizaje no supervisadas podrían ser el agrupamiento, la reducción de dimensiones y la estimación de densidad.

En la vida cotidiana, tanto *Data Scientists* y/o analistas de datos pueden realizan agrupaciones de aprendizaje no supervisado para encontrar respuestas a problemas de áreas como:

- ***Marketing***

Encontrar grupos de clientes con un comportamiento similar dada una gran base de datos de clientes que contienen sus propiedades y registros de compras anteriores.

- **Biología**

Clasificación de plantas y animales dadas sus características.

- **Estudios de terremotos**

Aglomeración de epicentros de terremotos observados para identificar zonas peligrosas. (Aprende IA, s.f.a, https://aprendeia.com/todo-sobre-aprendizaje-no-supervisado-en-machine-learning/)

En este sentido, ¿cuáles son las ventajas del aprendizaje no supervisado?

Ventajas del aprendizaje no supervisado

El uso de algoritmos de aprendizaje no supervisado en sus datos tiene muchos beneficios. Estas son algunas de las razones más comunes por las que las personas recurren al aprendizaje no supervisado:

- Puede manejar grandes cantidades de datos sin etiquetar y sin estructurar.
- Hace que sea más fácil y rápido analizar datos complejos.
- Es capaz de identificar patrones previamente no detectados.
- Aprende acerca de los datos para que puedan enseñarle lo que no sabe.

El aprendizaje no supervisado ayuda a los humanos a tomar mejores decisiones, por lo que las empresas recurren a este tipo de aprendizaje para obtener información muy específica sobre sus conjuntos de datos más complejos. La plataforma no supervisada actúa como un maestro, evalúa cómo las personas aprenden y entienden los datos para brindar mejores conocimientos.

Desventajas del aprendizaje no supervisado

- No se puede obtener información precisa con respecto a la clasificación de datos y la salida como datos utilizados en el aprendizaje no supervisado está etiquetada y no se conoce.
- La menor precisión de los resultados se debe a que los datos de entrada no son conocidos y no están etiquetados por la gente de antemano. Esto significa que la máquina requiere hacer esto por sí misma.
- Las clases espectrales no siempre corresponden a las clases informativas.
- El usuario necesita dedicar tiempo a interpretar y etiquetar las clases que siguen esa clasificación.
- Las propiedades espectrales de las clases también pueden cambiar con el tiempo, por lo que no puede tener la misma información de clase mientras

se mueve de una imagen a otra.

El secreto para obtener una ventaja competitiva en el mercado específico está en el uso eficaz de los datos. Los algoritmos de aprendizaje no supervisado te ayudan a segmentar los datos para estudiar las preferencias de tu público objetivo o ver cómo reacciona un virus específico a un antibiótico específico. (Aprende IA, s.f.b, https://aprendeia.com/aprendizaje-no-supervisado-machine-learning/)

Con esta información ya podemos entender de qué manera se clasifican las bases del *machine learning* y así comprender la gran utilidad que nos da su existencia.

Actividades

Según lo aprendido sobre aprendizaje no supervisado, determina cuál de las siguientes alternativas correspondería a un modelo que permita predecir información con estas características:

Clasificación de gustos alimenticios para frutas color rojo.

Detección de fraudes bancarios.

Predicción de salvación de una persona en el Titanic.

Deserción estudiantil.

Justificación

Tema 2. Modelos de aprendizaje no supervisado

Los modelos de aprendizaje no supervisados se utilizan para dos tareas principales: agrupar datos y asociar datos. A continuación, definiremos cada método de aprendizaje y destacaremos algoritmos y enfoques comunes para llevarlos a cabo de manera efectiva.

- **Agrupación de datos:** es un método para agrupar los objetos en grupos, de modo que los objetos con la mayoría de las similitudes permanezcan en un grupo y tengan menos o ninguna similitud con los objetos de otro grupo. El análisis de

conglomerados encuentra los puntos en común entre los objetos de datos y los clasifica según la presencia o ausencia de esos puntos en común.

- **Asociación:** una regla de asociación es un método de aprendizaje no supervisado que se utiliza para encontrar las relaciones entre las variables en la gran base de datos. Determina el conjunto de elementos que aparecen juntos en el conjunto de datos. La regla de asociación hace que la estrategia de *marketing* sea más efectiva. Por ejemplo, las personas que compran el artículo X (supongamos que un pan) también tienden a comprar el artículo Y (mantequilla/mermelada). Un ejemplo típico de la regla de asociación es el análisis de la cesta de la compra.

Para poder comprender un poco más sobre esto, presentaremos una lista de algoritmos que utilizaremos para nuestros modelos básicos de *machine learning*. Es preciso considerar que existen muchos modelos de aprendizaje no supervisado, de los cuales prácticamente nacen día a día nuevos, como a su vez los ya existentes se actualizan y generan nuevos modelos derivados, por lo cual la recomendación es siempre mantenerse actualizado respecto a los nuevos hallazgos de este tipo de aprendizaje.

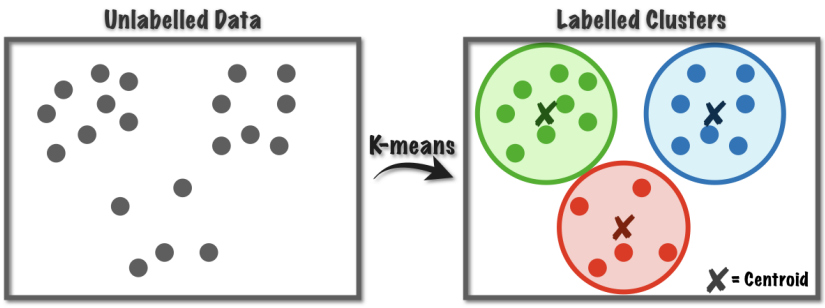
Algoritmos de aprendizaje no supervisado

Agrupamiento de *k-medias*

K-medias o *k-means* se refiere a una colección de datos agregados debido a ciertas similitudes. Para poder aplicarlo se debe definir un número objetivo, “k”, que se refiere a la cantidad de centroides que se necesita en el conjunto de datos. Un centroide es la ubicación imaginaria o real que representa el centro del grupo.

Dicho de otro modo, lo que buscar este algoritmo es poder agrupar datos en grupos que tendrán como centro del grupo un centroide.

Figura 2. Ejemplo *k-means*



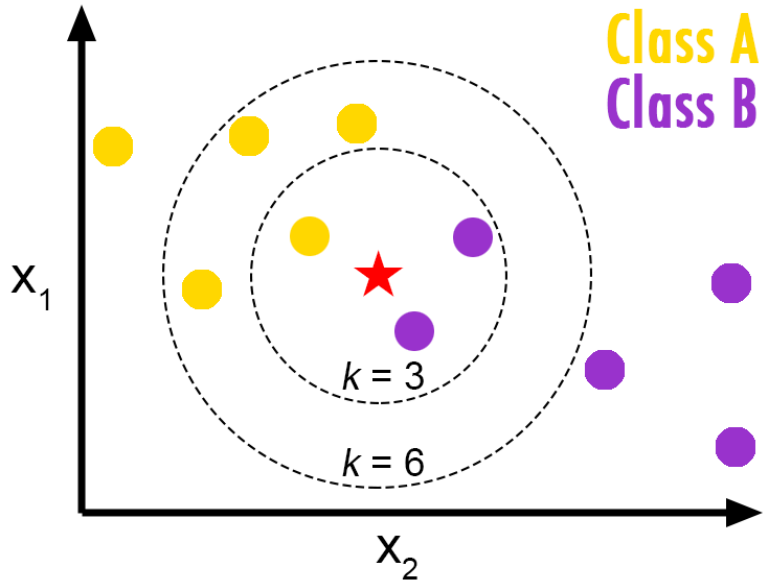
Fuente: indicar, no se encontró en Internet.

KNN (k-vecinos más cercanos)

Este algoritmo tiene como finalidad, asignar un conjunto de datos a cuantos de ellos corresponderán los “k” números de los cuales se va a componer una agrupación. Por ejemplo, si K= 3, quiere decir que sobre el centro de los datos se van a asignar los 3 puntos o vecinos de datos más cercanos. Y si K = 6, se amplía el rango del clúster para asignar a cada agrupación.

En el siguiente gráfico podemos expresar esta situación.

Figura 3. KNN



Fuente: Italo, 2018, https://bit.ly/3ha7IZw

Agrupación jerárquica

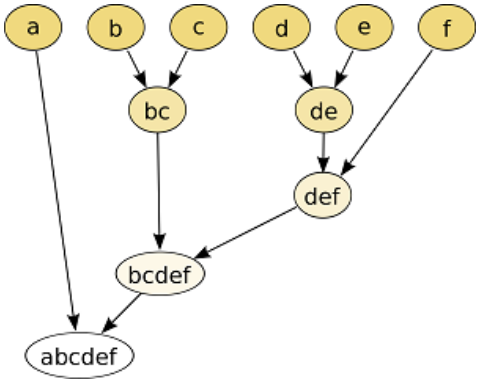
El algoritmo de clúster jerárquico agrupa los datos basándose en la distancia entre cada uno y buscando que los datos que están dentro de un clúster sean los más similares entre sí.

En una representación gráfica los elementos quedan anidados en jerarquías con forma de árbol.

Lo mejor para explicarlo es una imagen. Así que para ilustrar mejor este tema de agrupación en categorías voy a retomar un ejemplo gráfico muy difundido – y a la vez es el más descriptivo que he encontrado – que es el que exponen en la Wikipedia. (Duk2, s.f., https://estrategiastrading.com/clustering-

jerarquico/#:~:text=El%20algortimo%20de%20cl%C3%BAster%20jer%C3%A1rquico,para%20explicarlo%20es%20una%20imagen.)

Figura 4. Agrupación jerárquica



Fuente: Duk2, s.f., https://bit.ly/3FEIc9a

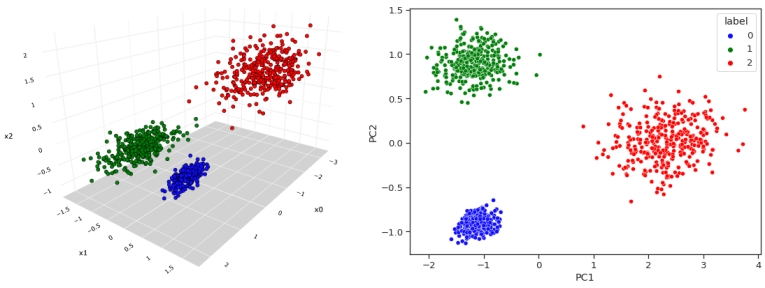
Análisis de componentes principales

El análisis de componentes principales es una técnica popular de aprendizaje no supervisado para reducir la dimensionalidad de los datos. Aumenta la interpretabilidad y, al mismo tiempo, minimiza la pérdida de información.

Su finalidad es ayudar a encontrar las características más significativas en un conjunto de datos y facilita el trazado de los datos en 2D y 3D. PCA ayuda a encontrar una secuencia de combinaciones lineales de variables.

Un ejemplo de visualización de PCA es el que puede verse en la siguiente imagen.

Figura 5. Reducción de dimensionalidad de 3D a 2D



Fuente: elaboración propia.

Actividades

Un cientista de datos está en búsqueda de poder agrupar las características de una planta conocida como “la flor del iris”, para lo cual, le gustaría juntar información de datos que consideren la unión de 7 valores más cercanos a un centro para realizar su agrupación. En este sentido, cual sería uno de los algoritmos más apropiados para poder realizar este análisis:

K-means.

Clasificación jerárquica.

KNN (K-vecinos más cercanos).

PCA.

IPNA.

Justificación

Tema 3. Evaluación de modelo con *k-means*

Con lo ya visto, hemos aprendido algunos fundamentos básicos de algoritmos no supervisados. Ahora, ha llegado el momento de ponerlo en práctica.

Para ello utilizaremos uno de los conjuntos de datos más populares dentro del aprendizaje de la ciencia de datos. Este *dataset* es muy popular, por lo cual Sklearn lo tiene incluido dentro de su librería, lo que nos hará mucho más fácil trabajar la información.

Este *dataset* se llama iris, el cual contiene cincuenta muestras de cada especie de flores con sus características. Posiblemente, ya hayamos trabajado con este set de datos anteriormente, este tiene tres componentes de la flor como son: virginica, versicolor y cetosa, analizados en base al ancho y largo del sépalo y pétalo, es decir, de las características de la flor.

Comencemos a importar esta información para poder trabajarla.

Figura 6. Paso 1 - Importar librerías a utilizar

```
from sklearn.cluster import KMeans
from sklearn import datasets
import pandas as pd

import matplotlib.pyplot as plt

```

Fuente: elaboración propia.

Lo siguiente que vamos a hacer es cargar nuestra información de datos que tú ya tienes disponibles al cargar Sklearn, bajo el código que se visualiza en la Figura 7.

Figura 7. Paso 2 - Cargar los datos

```
from sklearn import datasets
iris = datasets.load_iris() #Cargando el dataset de la flor de iris

tempDF=pd.DataFrame(iris.data,columns=iris.feature_names)

```

```
iris.data
iris.keys()
iris['feature_names']
print(iris['DESCR']) #Caracteristicas del data set
```

Fuente: elaboración propia.

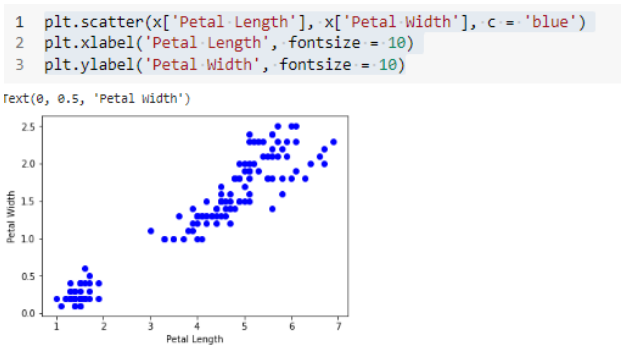
Figura 8. Paso 2 - Cargar los datos

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
```

Fuente: elaboración propia.

En el paso 3, graficaremos algunos datos para poder tener una idea de cómo se comportan, por ejemplo, el largo del petalo vs. el ancho del petalo de la flor.

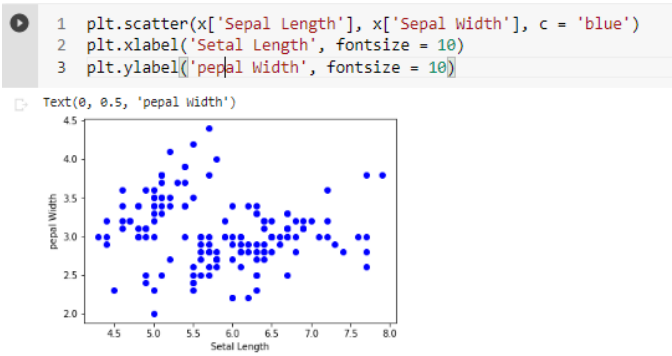
Figura 9. Paso 3 - largo del petalo vs. el ancho del petalo de la flor



Fuente: elaboración propia.

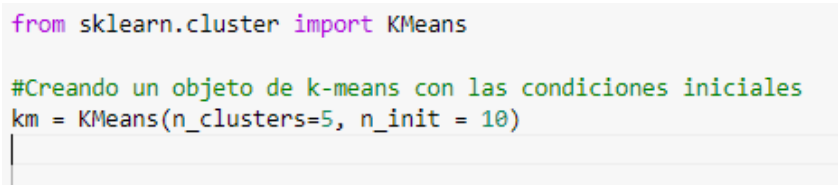
También podríamos graficar otros datos con el largo del sépalo vs. ancho de sépalo.

Figura 10. Paso 3 - largo del sépalo vs. ancho de sépalo



Fuente: elaboración propia.

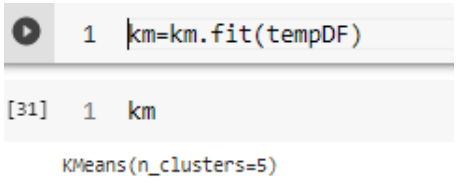
Figura 11. Paso 4 - Crear objeto *k-means*



Fuente: elaboración propia.

En este caso n_clusters nos dice la cantidad de agrupaciones en los cuales queremos separar nuestra muestra (para el ejemplo estos son 5), mientras que “n_init”, corresponde al número de semillas con las cuales se hará funcionar la búsqueda de centroides.

Figura 12. Paso 5 - Entrenar el modelo de datos



Fuente: elaboración propia.

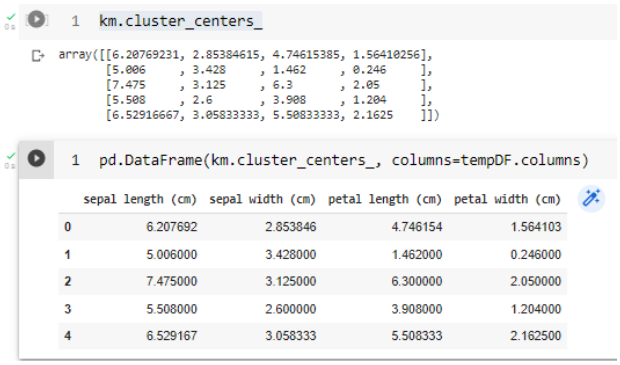
El **paso 6** consiste en revisar las características del modelo entrenado.

Una vez entrenado el modelo, existen nuevas características que podemos observar a nivel de atributos:

- cluster_centers_: Las coordenadas de los centroides. Si el algoritmo no converge estos no serán consistentes con los *labels* (etiquetas).
- Inertia_: la suma total de la distancia que existe entre clústeres.
- labels: Las etiquetas de cada punto (clústeres al cual pertenece cada dato).
- n_iter_: número de iteraciones del algoritmo.

En nuestro caso, se presentan los 5 centroides y sus puntos centrales del entrenamiento del algoritmo.

Figura 13. Paso 6



Fuente: elaboración propia.

Respecto a la inercia que nos muestra la distancia entre los distintos clústeres, podemos encontrar que es la que se visualiza en la siguiente figura.

Figura 14. Distancia entre los clústeres

```
[34]  1  km.inertia_
```

```
46.446182051282065
```

```
1  km.labels_ #Etiquetas para saber a que cluster pertenece cada dato
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 3, 0, 0, 0, 3, 0, 3, 3, 0, 3, 0, 3, 0,
       0, 3, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 0, 3, 0, 0, 0,
       3, 3, 3, 0, 3, 3, 3, 3, 3, 0, 3, 3, 4, 0, 2, 4, 4, 2, 3, 2, 4, 2,
       4, 4, 4, 0, 4, 4, 4, 2, 2, 0, 4, 0, 2, 0, 4, 2, 0, 0, 4, 2, 2, 2,
       4, 0, 0, 2, 4, 4, 0, 4, 4, 4, 0, 4, 4, 4, 0, 4, 4, 0], dtype=int32)
```

Fuente: elaboración propia.

Ahora podemos crear una columna más a nuestro *data frame* para saber dentro de la división de nuestros 5 clústeres, a cuál de ellos pertenece cada dato.

Figura 15. Creación de columna clúster

```
1  tempDF['cluster'] = km.labels_ #se crea columna cluster para ver a cual corresponde cada una de las etiquetas
2  tempDF
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	cluster
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	4
146	6.3	2.5	5.0	1.9	0
147	6.5	3.0	5.2	2.0	4
148	6.2	3.4	5.4	2.3	4
149	5.9	3.0	5.1	1.8	0

150 rows × 5 columns

Fuente: elaboración propia.

Actividades

Para poder realizar un entrenamiento de *k-means*, se debe definir, primero, el número de clústeres a utilizar. Para ellos se hace uso del código:

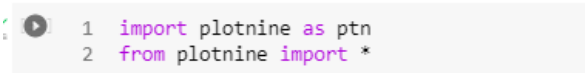
- n_intertia
- clúster_centers_
- KNN
- Random Forest

Justificación

Tema 4. Gráficas de un modelo *k-means*

Inicialmente, utilizaremos una nueva librería para poder hacer uso de gráficos que son un poco más difíciles de realizar. En este caso, la librería Plotnine nos ayudará para la creación de gráficos complejos a partir de los datos en un *data frame.* Esta utiliza configuraciones por defecto que ayudan a crear gráficos de calidad con unos pocos ajustes

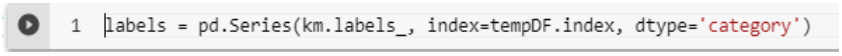
Figura 16. Librería Plotnine



Fuente: elaboración propia.

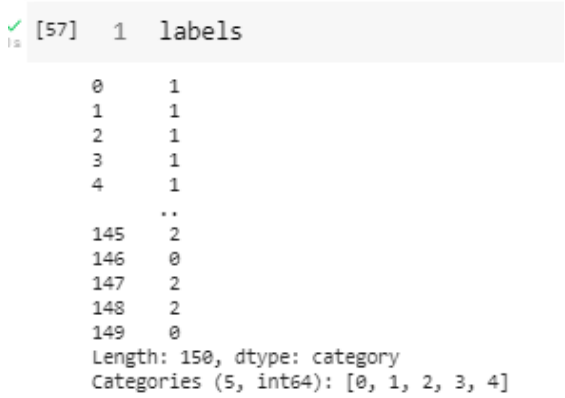
Después, definiremos la variable *labels* para poder marcar aquellas etiquetas con las cuales fueron categorizados los clústeres anteriormente.

Figura 17. Variable *labels*



Fuente: elaboración propia.

Figura 18. Variable *labels*

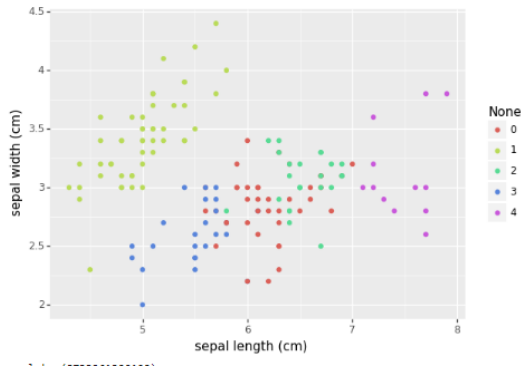


Fuente: elaboración propia.

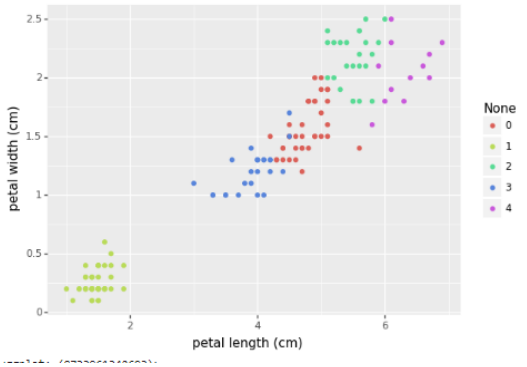
Luego, graficamos los valores anteriormente realizados, pero ahora agregamos color a cada una de las etiquetas marcadas anteriormente.

Figuras 19 y 20. Agregar color a las etiquetas

```
1 ggplot(tempDF, aes(x="sepal length (cm)", y="sepal width (cm)", color=labels)) + geom_point()
```




```
1 ggplot(tempDF, aes(x="petal length (cm)", y="petal width (cm)", color=labels)) + geom_point()
```



Fuente: elaboración propia.

Con esto ya podemos tener una idea de qué manera se componen cada uno de los clústeres del set de datos en sus agrupaciones buscadas.

Posteriormente, conoceremos el “método del codo” con el cual podremos determinar si realmente la cantidad de datos con los cuales estamos agrupando nuestro set de datos es el correcto o varía según su utilidad.

Actividades

Para poder realizar un entrenamiento de K-means, mediante el uso de que función podemos analizar la etiqueta de los clusters utilizados:

- n_intertia
- clúster_centers_
- KNN
- Random Forest

Unidad 2. Visualización de *clúster* y comparativa de modelos

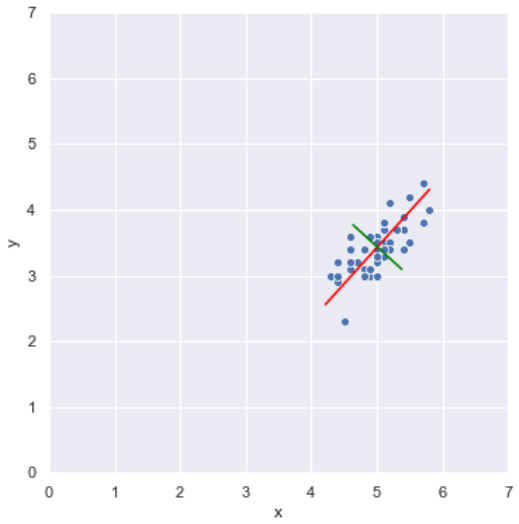
Tema 1. Visualización de clúster vía PCA

El método de Principal Component Analysis (PCA, o Análisis de Componentes Principales) gira los datos de forma que, desde un punto de vista estadístico, no exista una correlación entre las características rotadas pero que conserven la mayor cantidad posible de la varianza de los datos originales. Tras realizar la transformación suele realizarse una selección de las nuevas características.

Explicado de forma simple, PCA reduce la dimensionalidad de un conjunto de datos proyectándolos sobre un subespacio de menor dimensionalidad. Por ejemplo, datos con dos características (dispuestos en un plano) pueden ser proyectados sobre una línea tal y como vimos en el ejemplo de la introducción. O un conjunto de datos de tres características (dispuestos en un espacio de tres dimensiones) pueden ser proyectados en un plano (de dos dimensiones). En incluso aquí, los datos resultantes en el plano podrían ser reducidos a una única línea, pasando de las tres dimensiones originales a una sola.

En general, un conjunto de datos de n dimensiones puede ser reducido "proyectándolo" sobre un subespacio de m dimensiones, siendo m menor que n. Por ejemplo, consideremos el siguiente conjunto de muestras pertenecientes a un conjunto de datos de dos dimensiones:

Figura 21. Conjunto de datos de dos dimensiones



Fuente: Interactive Chaos, s.f.a, http://bit.ly/3NxQNfG

En la imagen anterior se muestra en rojo la recta correspondiente al primer componente principal, recta sobre la que se proyectarían los puntos en el caso de querer reducir la dimensionalidad del conjunto de datos. En verde, perpendicular a la anterior, se muestra la recta asociada al segundo componente principal.

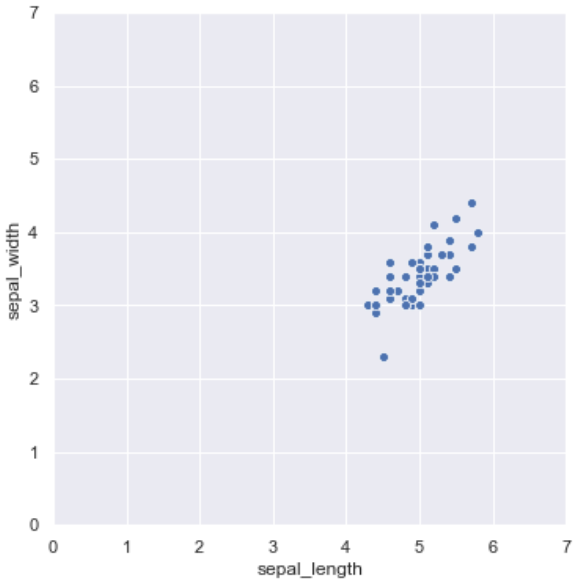
Estos datos tienen la máxima variación a lo largo de la recta correspondiente al primer componente principal, por lo que proyectarlos sobre dicha recta supone mantener la varianza a lo largo de dicho eje aun cuando se esté perdiendo la información de la varianza correspondiente al eje perpendicular (definido por la línea verde). En todo caso, con esta proyección se estaría reduciendo la dimensionalidad manteniendo la mayor información posible sobre la varianza del conjunto de datos original. (Interactive Chaos, s.f.a, https://interactivechaos.com/es/manual/tutorial-de-machine-learning/principal-component-analysis)

Figura 22 y 23. Ejemplos *dataset* IRIS con PCA

```
iris = sns.load_dataset("iris")
iris = iris.loc[iris.species == "setosa"][["sepal_length", "sepal_width"]]
iris.head(2)
```

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0

```
fig = plt.figure(figsize = (6, 6))
ax = plt.axes(aspect = "equal", xlim = (0, 7), ylim = (0, 7))
sns.scatterplot("sepal_length", "sepal_width", data = iris);
```



Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

Ahora importamos la clase PCA e instanciamos el aprendiz especificando que el número de componentes a extraer es igual a 1.

Figura 24. Instanciar el aprendiz

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 1, random_state = 0)
```

Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

Figura 25. Entrenamos y transformamos el *dataset*

```
iris_transformed = pca.fit_transform(iris)
```

Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

En la variable iris_transformed tenemos los datos transformados que, en este caso, son datos en una estructura de una dimensión.

Figura 26. Variable iris_transformed

```
iris_transformed[:5, :]
```

```
array([[ 0.1164805 ],
       [-0.38825857],
       [-0.37445282],
       [-0.51570561],
       [ 0.12338337]])
```

Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

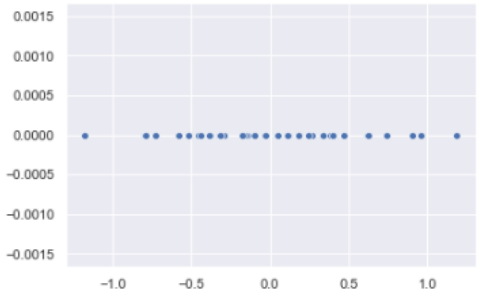
Si quisiéramos, podríamos mostrar estos puntos en el eje x de un espacio bidimensional.

Figura 27. Espacio bidimensional

```
sns.scatterplot(iris_transformed.flatten(), [0] * len(iris_transformed));
```

Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

Figura 28. Espacio bidimensional



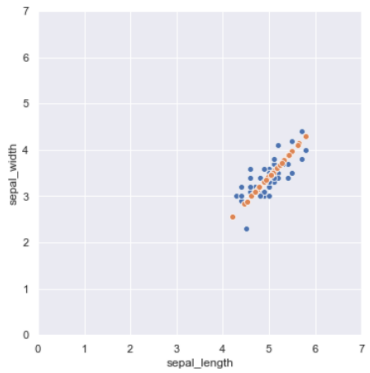
Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

Sin embargo, lo más interesante es mostrarlos directamente sobre la recta del primer componente principal. Para ello podemos usar el método inverse_transform de PCA que transforma los datos al espacio original.

Figura 29. Método inverse_transform de PCA

```
iris_o_transformed = pca.inverse_transform(iris_transformed)
```

```
fig = plt.figure(figsize = (6, 6))
ax = plt.axes(aspect = "equal", xlim = (0, 7), ylim = (0, 7))
sns.scatterplot("sepal_length", "sepal_width", data = iris);
sns.scatterplot(iris_o_transformed[:, 0], iris_o_transformed[:, 1]);
```



Fuente: Interactive Chaos, s.f.b, http://bit.ly/3zJ1I0l

Actividades

A cuál de los siguientes conceptos corresponde la siguiente aseveración: “método capaz de reducir la dimensionalidad de un conjunto de datos proyectándolos sobre un subespacio de menor dimensionalidad”.

PCA.

Random forest.

Sklearn.

K-means.

Anaconda.

Justificación

Tema 2. Determinación de número de clústeres

Muchas veces podríamos definir de manera “aleatoria” el número de clústeres que nos permitan agrupar de mejor forma cada uno de los datos de *k-means*, sin embargo, es necesario seleccionar un valor k que sea óptimo para el entendimiento del número de grupos en los que se agrupan los datos.

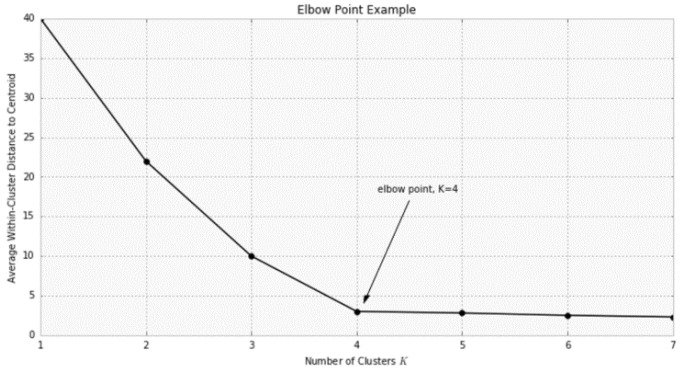
En general, no hay una forma exacta de determinar el número de grupos, pero se pueden usar ciertas reglas o estadísticos que nos ayudan a estimar el número de grupos, para ello utilizaremos el conocido método del codo.

El método del codo

La idea básica de los algoritmos de clustering es que cada observación se encuentre muy cerca a las de su mismo grupo y los grupos lo más lejos posible entre ellos.

El método del codo utiliza la distancia media de las observaciones a su centroide. Es decir, se fija en las distancias intra-clúster. Cuanto más grande es el número de clústeres k, la varianza intra-clúster tiende a disminuir. Cuanto menor es la distancia intra-clúster, esto es mucho mejor, ya que significa que los clústeres son más compactos. El método del codo busca el valor k que satisfaga que un incremento de k, no mejore sustancialmente la distancia media intra-clúster. (Alvaro, 2019, https://machinelearningparatodos.com/segmentacion-utilizando-k-means-en-python/)

Figura 30. Gráfico



Fuente: Alvaro, 2019, http://bit.ly/3h3hOvj

En este caso la interpretación del método del codo será buscar el número de clústeres (eje x) en el que el quiebre de la curva sea totalmente significativo en comparación a si miráramos la gráfica en su totalidad.

En este caso, podemos darnos cuenta de que esto ocurre cuando en número de clústeres son 4. Para ello, debemos considerar que cada caso a analizar es distinto, por lo que no existe un número de clústeres únicos que caracterice la mejor forma de analizar un algoritmo de *k-means*.

Llevemos esto a la práctica y retomemos nuestro ejercicio de la flor de iris. En relación con los centroides ya creados, buscaremos el valor de K de la siguiente forma:

Figura 31. Código

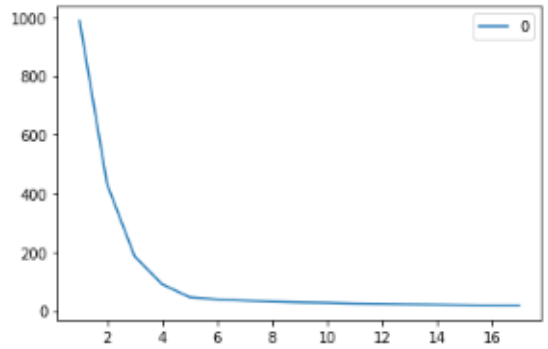
```
sse = [] #Variable para guardar el within cluster distance
numK = 18
for k in range(1, numK): #número de cluster a probar
    #Creando el modelo, entrenandolo y obteniendo el valor del within cluster distance
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(tempDF)
    sse.append(kmeans.inertia_)
```

Fuente: elaboración propia.

Figura 32. Código

```
1  pd.DataFrame(sse, index=np.arange(1,numK)).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f188a8e9790>
```



Fuente: elaboración propia.

De esta forma, la interpretación es a elección del analista de datos, al parecer el quiebre más significativo de los datos ocurre cuando el número de clústeres es 5. Justamente, con los datos que comenzamos a trabajar al comienzo de nuestro ejercicio.

Actividades

A qué concepto hace referencia la siguiente frase: "corresponde al método a través del cual se determina el posible número de clústeres clave para poder realizar nuestros agrupamientos no supervisados":

- Método del codo.
- Método de árboles de decisión.
- Método de entrenamiento y validación.
- Método de validación de datos.
- Método intra-clúster.

Justificación

Tema 3. Análisis de modelos supervisados y no supervisados

A lo largo de los módulos, hemos podido revisar modelos de aprendizaje supervisado y no supervisado, dos tipos diferentes de enfoque de modelo de aprendizaje automático.

Estos difieren en la forma en que se entrenan los modelos y la condición de los datos de entrenamiento que se requieren. Cada enfoque tiene diferentes puntos fuertes, por lo que la tarea o el problema al que se enfrenta un modelo de aprendizaje supervisado o no supervisado suele ser diferente.

Es por ello que es muy importante comprender las diferencias fundamentales entre el aprendizaje supervisado y el no supervisado. Si una organización busca implementar un modelo de aprendizaje automático, la elección se hará al comprender los datos disponibles y el problema que debe resolverse.

Aprendizaje supervisado vs. no supervisado comparado (Aprende IA, s.f.b)

La principal diferencia entre el aprendizaje supervisado y el no supervisado es la necesidad de datos de entrenamiento etiquetados. El aprendizaje automático supervisado se basa en datos de entrenamiento de entrada y salida etiquetados, mientras que el aprendizaje no supervisado procesa datos sin etiquetar o sin procesar.

En el aprendizaje automático supervisado, el modelo aprende la relación entre los datos de entrada y salida etiquetados. Los modelos se ajustan hasta que puedan predecir con precisión los resultados de los datos no vistos. Sin embargo, la creación de datos de entrenamiento etiquetados a menudo requerirá muchos recursos. El aprendizaje automático no supervisado, por otro lado, aprende de datos de entrenamiento sin etiquetar. Un modelo no supervisado aprenderá relaciones y patrones dentro de este conjunto de datos sin etiquetar, por lo que a menudo se usa para descubrir tendencias inherentes en un conjunto de datos determinado.

Entonces, en general, el aprendizaje supervisado y no supervisado son diferentes en el enfoque del entrenamiento y los datos de los que aprende el modelo. Pero como resultado, también difieren en su aplicación final y fortalezas específicas.

Los modelos de aprendizaje automático supervisado, generalmente, se usan para predecir resultados para datos no vistos. Esto podría ser predecir fluctuaciones en los precios de la vivienda o comprender el sentimiento de un mensaje.

Los modelos, también, se utilizan para clasificar datos no vistos contra patrones aprendidos. Por otra parte, las técnicas de aprendizaje automático no supervisado se usan, por lo general, para comprender patrones y tendencias dentro de datos no etiquetados. Esto podría ser agrupar datos debido a similitudes o diferencias, o identificar patrones subyacentes dentro de conjuntos de datos. El aprendizaje automático no supervisado se puede utilizar para agrupar datos de clientes en campañas de *marketing* o para detectar anomalías y valores atípicos.

Las principales diferencias entre el aprendizaje supervisado y el no supervisado incluyen:

- La necesidad de datos etiquetados en el aprendizaje automático supervisado.
- El problema que se despliega para resolver el modelo. El aprendizaje automático supervisado, generalmente, se usa para clasificar datos o hacer predicciones, mientras que el aprendizaje no supervisado se usa, por lo general, para comprender las relaciones dentro de los conjuntos de datos.
- El aprendizaje automático supervisado requiere muchos más recursos debido a la necesidad de datos etiquetados.
- En el aprendizaje automático no supervisado puede ser más difícil alcanzar niveles adecuados de explicabilidad debido a la menor supervisión humana.

Actividades

Según lo aprendido sobre aprendizaje no supervisado, determina cuál de las siguientes alternativas correspondería a un modelo que permita predecir información de este tipo:

Clasificación de plantas y animales dadas sus características.

Detección de fraudes bancarios.

Detección de spam.

Deserción estudiantil.

Justificación

Tema 4. Siguientes pasos en *machine learning*

Dentro de este curso, hemos podido revisar, a modo general, los distintos caminos introductorios al *machine learning*. Si bien es cierto que es muy útil para la clasificación y solución de problemas que requieren el análisis de datos complejos, existe un siguiente paso dentro de esta rama de aprendizaje: el aprendizaje profundo o también conocido como *deep learning*.

¿Qué es el aprendizaje profundo?

El aprendizaje profundo es un sector de la inteligencia artificial (IA) relacionado con la creación de estructuras informáticas que imitan las redes neuronales altamente complejas del cerebro humano. Debido a esto, a veces también, se lo denomina aprendizaje neuronal profundo o redes neuronales profundas (DNN).

Como un subconjunto del aprendizaje automático, las redes neuronales artificiales utilizadas en el aprendizaje profundo son capaces de clasificar mucha más información de grandes conjuntos de datos para aprender y, en consecuencia, usarla para tomar decisiones. Estas grandes cantidades de información que las DNN rastrean en busca de patrones a veces se denominan macrodatos.

El aprendizaje automático tradicional requiere una programación basada en reglas y una gran cantidad de preprocesamiento de datos sin procesar por parte de científicos y analistas de datos. Esto es propenso al sesgo humano y está limitado por lo que podemos observar y calcular mentalmente antes de entregar los datos a la máquina. El aprendizaje supervisado son formas en que las computadoras se familiarizan con los datos y aprenden qué hacer con ellos.

Las redes usan capa tras capa de neuronas para que puedan procesar una gran cantidad de datos rápidamente. Como resultado, tienen el "poder mental" para comenzar a notar otros patrones y crear sus propios algoritmos basados en lo que están "viendo". Este es un aprendizaje no supervisado y conduce a avances tecnológicos que a los

humanos les llevaría mucho más tiempo lograr. El modelado generativo es un ejemplo de aprendizaje no supervisado.

Ejemplos del mundo real de aprendizaje profundo

Las aplicaciones de aprendizaje profundo se utilizan (y se desarrollan) cada vez que realiza una búsqueda en Google. También se utilizan en escenarios más complicados, como en automóviles autónomos y en el diagnóstico de cáncer. En estos escenarios, la máquina casi siempre busca irregularidades. Las decisiones que toma la máquina se basan en la probabilidad para predecir el resultado más probable. Obviamente, en el caso de la conducción automatizada o las pruebas médicas, la precisión es más crucial, por lo que las computadoras se prueban rigurosamente con datos de entrenamiento y técnicas de aprendizaje.

Los ejemplos cotidianos de aprendizaje profundo se complementan con la visión artificial para el reconocimiento de objetos y el procesamiento del lenguaje natural para cosas como la activación por voz. El reconocimiento de voz es una función con la que estamos familiarizados mediante el uso de asistentes activados por voz, como Siri o Alexa, pero la capacidad de una máquina para reconocer el lenguaje natural puede ayudar de maneras sorprendentes.

¿Cómo funciona el aprendizaje profundo?

Los algoritmos de aprendizaje profundo utilizan conjuntos de datos etiquetados muy grandes, como imágenes, texto, audio y video, para generar conocimiento. En su cálculo del contenido, escaneando y familiarizándose con él, la máquina comienza a reconocer y saber qué buscar. Al igual que el cerebro humano, cada neurona de la computadora tiene un papel en el procesamiento de datos, proporciona una salida al aplicar el algoritmo a los datos de entrada proporcionados. Las capas ocultas contienen grupos de neuronas.

En el corazón de los algoritmos de aprendizaje automático se encuentra la optimización automatizada. El objetivo es lograr el resultado más preciso, por lo que necesitamos la velocidad de las máquinas para evaluar de manera eficiente toda la información que tienen y comenzar a detectar patrones que podemos haber pasado por alto. Esto también es fundamental para el aprendizaje profundo y cómo se entrenan las redes

neuronales artificiales.

Las redes neuronales recurrentes (RNN) se consideran de última generación porque son las primeras de su tipo en utilizar un algoritmo que les permite recordar su entrada. Debido a esto, los RNN se utilizan en el reconocimiento de voz y el procesamiento del lenguaje natural en aplicaciones como Google Translate.

Como vemos, el mundo de la ciencia de datos cada día puede sorprendernos más, por lo cual nos propongo seguir actualizándonos en él para que a futuro podamos realizar modelos más complejos y que generen alto valor en la organización en la cual nos encontremos.

Actividades

Determina cuál es el indicador que permite medir la calidad del modelo de machine learning en tareas de clasificación.

- Precisión.
- Accuracy.
- Recall.
- F1-Score.

Justificación

Video de habilidades

Glosario

Referencias

Alvaro. (2019). Segmentación utilizando *k-means* en Python. https://machinelearningparatodos.com/segmentacion-utilizando-k-means-en-python/

Aprende IA. (s.f.a). Todo sobre aprendizaje no supervisado. https://aprendeia.com/todo-sobre-aprendizaje-no-supervisado-en-machine-learning/

Aprende IA. (s.f.b). Aprendizaje no supervisado. https://aprendeia.com/aprendizaje-no-supervisado-machine-learning/

Chauhan, N. S. (2020). Métricas de evaluación de modelos en el aprendizaje automático. https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico

Duk2. (s.f.). Algoritmos de Data Mining para agrupar datos. *Clustering Jerárquico*. *Estrategias de trading* [blog]. https://estrategiastrading.com/clustering-jerarquico/#:~:text=El%20algortimo%20de%20cl%C3%BAster%20jer%C3%A1rquico,para%20explicarlo%20es%20una%20imagen.

Hogarmania. (s.f.). https://www.hogarmania.com/salud/bienestar/dieta-sana/calendario-frutas-verduras-temporada.html

Interactive Chaos. (s.f.a). Principal Component Analysis. https://interactivechaos.com/es/manual/tutorial-de-machine-learning/principal-component-analysis

Interactive Chaos. (s.f.b). Ejemplos con PCA. https://interactivechaos.com/es/manual/tutorial-de-machine-

learning/ejemplo-con-pca

Italo, J. (2018). KNN (K-Nearest Neighbors) #1. https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d