**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| Programa de formación | Algoritmo de agrupamiento no supervisado K-means con Python |

|  |  |  |  |
| --- | --- | --- | --- |
| Competencia | 220501114- Sistematizar datos masivos de acuerdo con métodos de analítica y herramientas tecnológicas | Resultados de aprendizaje | 220501114-02 - Agrupar los datos aplicando un algoritmo no supervisado  220501114-03 - Validar el agrupamiento de datos de acuerdo con los objetivos definidos para el análisis. |

|  |  |
| --- | --- |
| Número del componente formativo | 02 |
| Nombre del componente formativo | Aprendizaje no supervisado K-means |
| Breve descripción | Un tipo de *machine learning* es el aprendizaje no supervisado usando el algoritmo K-means. Este componente formativo se orienta a conocer cómo funciona el algoritmo a través de un ejercicio planteado e identificar el *clúster* de un *dataset* seleccionado, usando Python como lenguaje para realizar algoritmos de inteligencia artificial por su potencia en cuanto a exploración estadística, sus librerías gráficas y librerías de *machine learning.* |
| Palabras clave | Aprendizaje no supervisado, *dataset,* inteligencia artificial, K-means, máquina de aprendizaje. |

|  |  |
| --- | --- |
| Área ocupacional | 2 - Ocupaciones en Ciencias Naturales, aplicadas y relacionadas |
| Idioma | Español |

**A. TABLA DE CONTENIDO**

**1. Aprendizaje no supervisado K-means**

1.1. Selección del algoritmo

1.1.1. Selección del conjunto de datos.

1.2. Extracción y selección de características

1.3. Refinamiento del algoritmo de agrupación

1.4. Segmentación de conjuntos de datos por atributos compartidos

**2. Herramientas tecnológicas para el agrupamiento de datos**

**3. Validación del resultado del análisis**

**B. DESARROLLO DE CONTENIDO**

**Introducción**

Se da la bienvenida al componente formativo denominado “Aprendizaje no supervisado K-Means”, el cual hace parte del programa de formación complementaria “Algoritmo de agrupamiento no supervisado  K-means con Python”. En el siguiente video se amplía la información que se desarrollará alrededor de esta temática.

**CF02\_Introduccion\_Video**

# **1. Aprendizaje no supervisado K-means**

Python es un lenguaje bastante útil para realizar implementaciones de algoritmos de *machine learning*, cuenta con una sintaxis clara y fácil de aprender;  También, posee  tipos de datos de alto nivel, permite procesar y manipular el texto  para procesar datos no numéricos, cuenta con librerías  como SciPy y Numpy, entre otras,  para realizar operaciones de vectores y matrices.

A continuación se conocerán las librerías más usadas en el ámbito de Python:

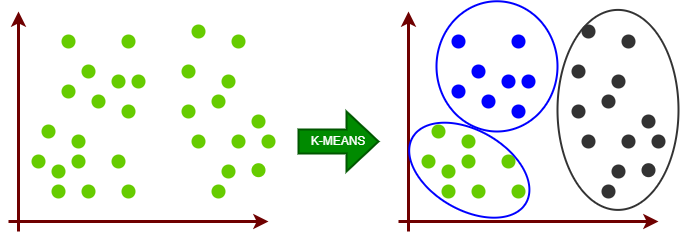
|  |
| --- |
| **CF02\_1\_Pestañas verticales\_Librerías\_Kmeans** |

Con la información anterior es importante tener en cuenta que Python es usado por los científicos de datos, sobrepasando el uso de R, su gama de paquetes y librerías hacen que el estudio de la ciencia de datos sea una ciencia.

Con el aprendizaje no supervisado se trata de aprender las relaciones y estructuras que existen en los datos sin etiquetarlos como variables independientes o variables dependientes; realmente un algoritmo de aprendizaje no supervisado no requiere datos de entrenamiento a diferencia de los algoritmos de aprendizaje supervisado. El algoritmo de aprendizaje no supervisado lo que hace es interpretar y agrupar datos, sin la necesidad de contener datos de entrenamiento, de modo que intentará averiguar a qué grupo o *clúster* pertenecen los nuevos datos, comparando sus características con las de los *clúster,* tal como se representa a continuación.

**Figura 1**

*Algoritmo de aprendizaje*



|  |
| --- |
| En general, lo que se busca con el aprendizaje no supervisado es el conocimiento o patrones existentes entre un montón de datos, de los cuales no se conocen relaciones entre sus variables, no hay datos de referencia; el algoritmo *K-means* es un algoritmo de aprendizaje no supervisado. Los modelos predictivos se aprenden de manera supervisada mientras que los modelos descriptivos son producidos por técnicas de aprendizaje no supervisado. |

## **1.1. Selección del algoritmo**

La agrupación con *K-means*  es una de las técnicas más usadas para particionar datos, si tiene (**n)** observaciones y (**k)** *clúster* conocidos, entonces, cada observación pertenece al *clúster* con la media más cercana. Como medida se usa la distancia euclidiana; entre muchas ventajas que tiene este algoritmo las principales son la **velocidad y la sencillez** al implementarlo, por tanto, se puede usar con millones de observaciones.

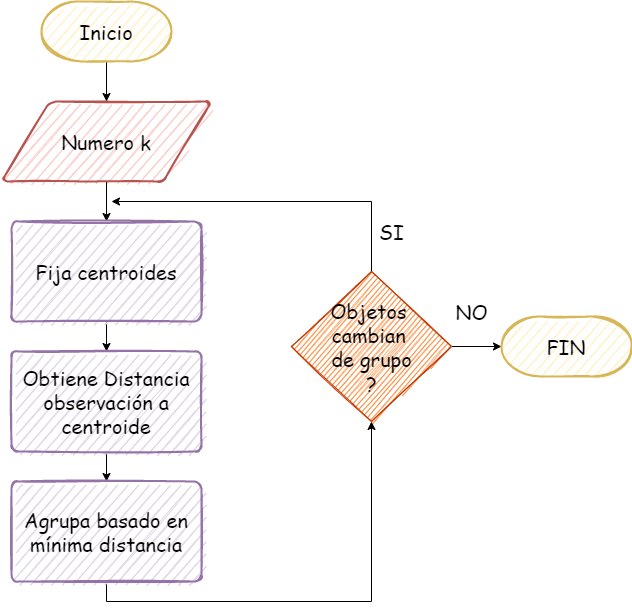
El algoritmo se resume como se muestra a continuación:

|  |
| --- |
| **CF02\_1\_1\_Pasos\_selección algortimo** |

Explicado de otra forma, se puede visualizar por medio del siguiente diagrama.

**Figura 2**

*Proceso de selección de algoritmo*

**

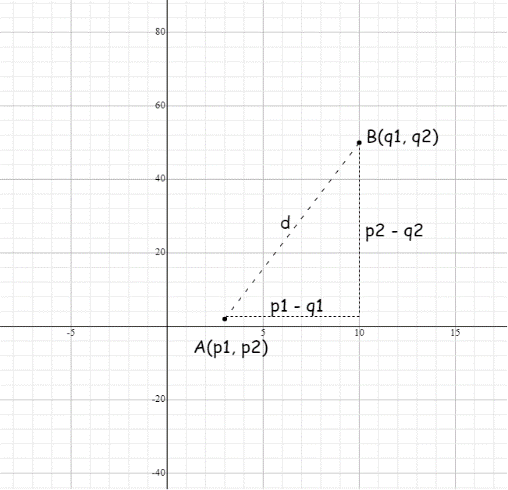
Como ejemplo de aplicación del algoritmo se agruparán diez puntos ubicados en una dimensión para demostrar la sencillez del algoritmo:

|  |
| --- |
| **CF02\_1\_1\_Video animado\_Aplicación del algortimo** |

Ahora bien, en otro ejemplo como el que presenta la siguiente figura, la distancia es observada fácilmente, puesto que es la medida de una línea recta; en dos dimensiones, se debe trabajar principalmente con el concepto de la distancia euclídea.

**Figura 3**

*Distancia euclídea en el plano*



Aplicando el teorema de Pitágoras la distancia euclídea en dos dimensiones es:

### ***Selección del conjunto de datos***

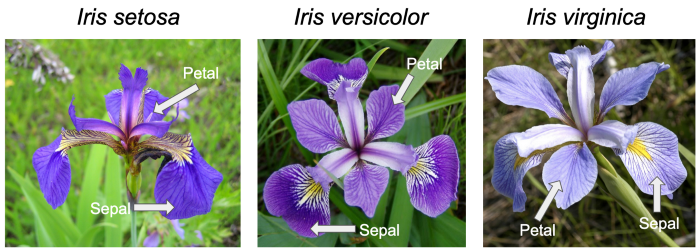
|  |  |
| --- | --- |
| Un conjunto de datos o *dataset* es un componente de la *bigdata* que representa un conjunto de información en una matriz o tabla conformada por filas y columnas, cada fila es una instancia del *dataset*, representa la observación o un registro y cada columna representa una variable o una característica de datos, cada valor puede ser numérico como enteros o decimales y cadenas. | **Figura 4**  *Diagrama de un dataset*    Nota. Tomado de Feeding the machine (2019) |

Antes de realizar cualquier estudio de aprendizaje automático es prioritario conocer muy bien el problema que se quiere resolver y por ende, el *dataset* involucrado; los datos tienen más sentido si se conocen las historias detrás de cada dato. Para comprender mejor la información anterior se presenta el siguiente ejemplo, que será el ejercicio base de todo este componente formativo y del cual se desprenderá la información temática relacionada en el contenido:

**Problema:**

“**Clasificar** las flores tipo iris, con el objetivo de **predecir** a qué tipo de especie pertenece, de la cual se conocen las longitudes de sus pétalos y sépalos”.

A continuación se muestra una variedad de flores llamada **iris**, esta flor tiene tres especies distintas: iris setosa, iris virginica e iris versicolor, que se diferencian por los anchos y altos de los pétalos y de los sépalos.



Hay similitudes en morfología entre esas especies, aunque todas tienen el mismo nombre hay alguna diferencia entre las tres especies. Se observa que hay una diferencia notable en la semillas de esas tres especies, además de la diferencia en los tamaños de sus pétalos y sépalos.

|  |  |
| --- | --- |
|  | La flor de iris puede identificarse atendiendo a los anchos y altos de sus pétalos y sépalos.  Si no se conoce qué es el pétalo y qué es el sépalo observe la figura, los sépalos protegen la flor, brindan protección, ya que los pétalos rodean las unidades reproductoras de las flores. Los sépalos en conjunto forman el cáliz y los pétalos en conjunto forman la corola. |

Para resolver el problema con Python se usará una *dataset* en la nube llamadoiris.csv (Kagle.com, 2022), cuyos datos contienen características (longitud y anchura de sépalos y pétalos) de 50 muestras de cada una de las tres especies de iris (setosa, virginica y versicolor) para un total de 150 observaciones.

|  |  |
| --- | --- |
| El *dataset* se puede obtener en la siguiente ruta <https://www.kaggle.com/datasets/saurabh00007/iriscsv> o directamente se puede cargar de la ruta https://raw.githubusercontent.com/toneloy/data/master/iris.csv como se muestra en el código fuente. También como anexo en el presente componente formativo se encuentra el archivo en el que se muestran los datos de *dataset*: **Anexo1-iris.csv**  De igual forma, el código también se encontrará en el anexo denominado: **Anexo2-ejecutable-iris.ipynb** | **Ver anexo 01** |
| **Ver anexo 02** |

### 

### **Procedimiento de resolución del problema**

### **Introducción a los *dataframes***

En Python un *dataframe* es la estructura  de datos fundamental de la librería denominada **Pandas** (Python Data Analysis Library) y para iniciar  la exploración de este *dataset* en Python se debe primero importar la librería; Pandas permite importar archivos csv, excel, json, html o sql, el código fuente de ejemplo se encuentra en el anexo indicado previamente: **Anexo1-iris** Observe el ejemplo a continuación:

|  |
| --- |
| import pandas as pd # importa pandas como pd , es un alias para referirse a panda iris\_df = pd.read\_csv (<https://raw.githubusercontent.com/toneloy/data/master/iris.csv>) iris\_df.head(8) # este comando permite visualizar los primeros 8 registros. |

Los *dataset* pueden estar almacenados en repositorios de  archivos o en discos duros, memorias etc., y los *dataframes* son estructuras que se obtienen a través  de los *datasets* y se guardan en la memoria RAM, de este modo se pueden hacer operaciones sobre sus datos.

A continuación conocerá cómo inicia el procedimiento con la introducción de datos en el *dataframe:*

|  |
| --- |
| **CF02\_1\_1\_1\_pasos horizontales\_Introducción Datos** |

## **1.2. Extracción y selección de características**

### El análisis exploratorio de datos es usado para ganar un mejor entendimiento de los datos y conocimiento de la información en profundidad del problema propuesto, como características principales, variables y relaciones ocultas entre ellas.

### Existe un sinnúmero de métodos para realizar la exploración de datos como:

|  |
| --- |
| **CF02\_1\_2\_tarjetas\_Métodos** |

### Para comprender los datos se debe **iniciar con un análisis exploratorio**, de esta forma se conoce la información en profundidad del problema propuesto, en Python se puede realizar fácilmente.

### Lo primero que se debe hacer es importar las librerías necesarias como muestra el código fuente:

|  |
| --- |
| import pandas as pd # librería para obtener datos y usar la ciencia de datos. import matplotlib.pyplot as plt # librería para realizar gráficas. import seaborn as sns #librería para hacer gráficas, más elaboradas que la anterior. import numpy as np # librería que tiene métodos numéricos para Python. from scipy.stats import norm  from sklearn.preprocessing import StandardScaler from scipy import stats  %matplotlib inline |

* **Uso de boxplot**

Para realizar la exploración de los datos se usarán gráficos como: el diagrama de dispersión y el diagrama de pares o **boxplot** (el diagrama de cajas y bigotes); con este último, se puede observar la variable categórica especies (***species***) y su variación de longitudes y anchos de los sépalos (***petal\_length, petal\_width***) y pétalos (***sepal\_length, sepal\_width***) para cada especie. Con este diagrama se puede comparar la variación de datos entre las diferentes especies, tal como se evidencia en el siguiente recurso.

|  |
| --- |
| **CF02\_1\_2\_acordeon con numeral\_boxplot** |

Observando los cuatro gráficos boxplotse puede obtener algunas conclusiones preliminares, por ejemplo:

La longitud de los sépalos de la especie setosa son menores que las otras especies; pero su ancho es de longitud mayor que las otras especies, se observa que la longitud y ancho del sépalo se diferencia de las otras dos especies; pero hay una diferenciación menor entre las otras dos especies.

La variables de los pétalos de las tres especies tienen variaciones importantes tanto en longitud del pétalo como el ancho del pétalo respecto a las variaciones de los anchos y longitudes de los sépalos, lo cual hace que las variables de longitud y ancho de los pétalos contribuyan más a la hora de definir a qué especie pertenece una flor.

* **Uso de Scatter Plot o diagrama de dispersión**

El Scatter Plot o diagrama de dispersión es un tipo de despliegue de datos que muestra la relación entre dos variables numéricas en un *dataframe*; el diagrama muestra  qué tanto se afectan entre sí dichas variables o que grado de independencia hay entre ellas; pero se deben entender que la  dispersión se define como la medida de distancia entre los valores de un *dataset*a su punto medio.

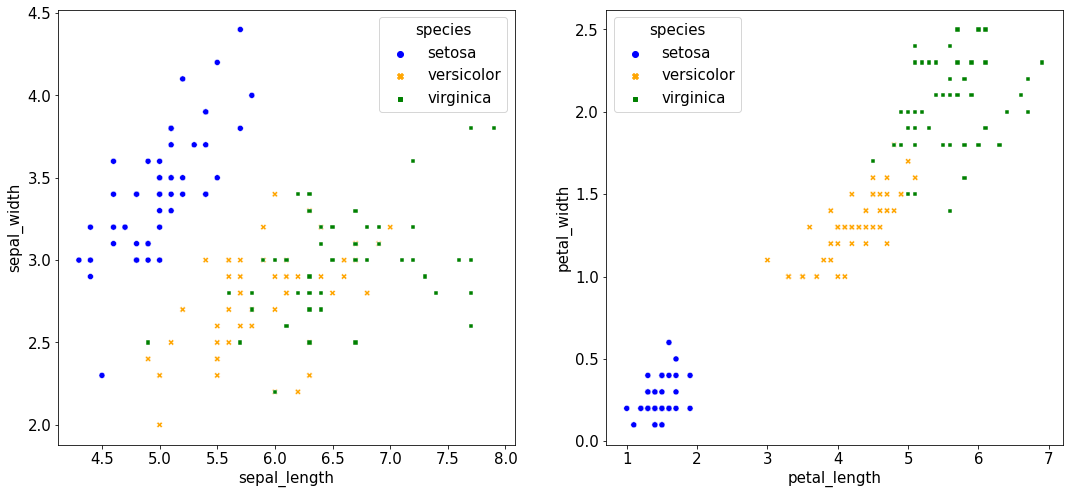
En el caso del *dataset* iris.csv (Kagle.com, 2022) se va a revisar la relación entre longitudes y anchos del sépalo y entre longitudes y anchos del pétalo, para lo cual se usará el siguiente código:

|  |
| --- |
| f, ax = plt.subplots(figsize=(18,8)) #área en donde voy a graficar plt.subplot (1,2,1) sns.scatterplot (x = iris\_df.sepal\_length, y = iris\_df.sepal\_width, hue = iris\_df.species, style = iris\_df.species, palette =['blue', 'orange', 'green'])  plt.subplot (1,2,2) sns.scatterplot (x = iris\_df.petal\_length, y = iris\_df.petal\_width, hue = iris\_df.species, style = iris\_df.species, palette =['blue', 'orange', 'green']) |

Se obtienen los siguientes gráficos:

**Figura 5**

*Distribución de longitud y ancho de sépalo y pétalo*



El gráfico confirma las observaciones realizadas con los boxplot, que las longitudes de las especies setosa son más pequeñas que la de los otras dos especies y con el ancho del sépalo más grande que las otras dos especies, mostrando el patrón de la figura; las otras dos especies no se agrupan de una manera diferenciadora, si se desea clasificar solo la especie setosa.

En cambio, en el gráfico de dispersión de longitudes y en el ancho del pétalo sí se muestra un patrón que permite clasificar una flor en cada especie, así se concluye que las variables longitud y ancho de los pétalos contribuyen enormemente a la hora de clasificar el tipo de especie a la que pertenece una flor.

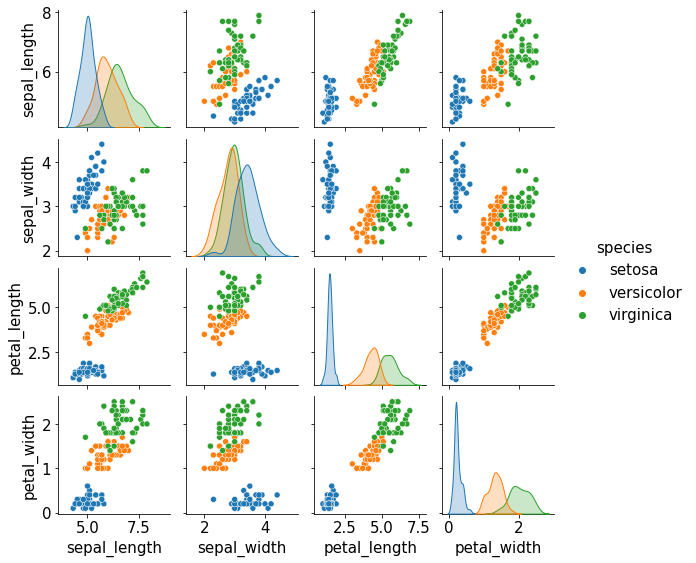
* **Uso de Pairplot o gráfica de pares**

Con esta gráfica se puede observar a simple vista todas las relaciones de las columnas o valores entre sí, en una cuadrícula con ejes **X** y **Y.** Las variables pueden ser continuas o categóricas, se usa para entender mejor el conjunto de característica que ayuda a interpretar la relación entre dos variables o para formar los *clúster* más separados, también ayuda a formar modelos de clasificación simples observando y separando los datos.

Se usa el mismo ejemplo para entender la gráfica de pares, aplicando:

sns.pairplot(iris\_df, hue='species', height=2)

**Figura 6**

*Gráfico de pares o pairplot*

Esta gráfica permite observar algunos patrones tales como la longitud del sépalo y el ancho del sépalo que presenta mucho traslape entre las especies, a diferencia de la longitud y el ancho de los pétalos, cuyo traslape es poco, de esta forma se puede escoger la longitud del pétalo (*petal\_length*) y el ancho del pétalo (*petal\_width*) como características para la clasificación.

## **1.3. Refinamiento del algoritmo de agrupación**

La agrupación de datos es un método de clasificación no supervisada, cuyo principal objetivo es buscar  patrones ocultos en los datos y  con estos crear grupos específicos llamados clúster, estos métodos son usados en la detección de anomalías y en muchas áreas del conocimiento tales como la medicina, las matemáticas, la biología, la astronomía y la industria. A continuación se explica un poco más en qué consiste el refinamiento del algoritmo de agrupación:

|  |
| --- |
| **CF02\_1\_3\_pestañas\_refinamiento** |

* **Continuación del procedimiento**

Antes de realizar el procesamiento de los datos se separan unas filas que pueden ser los datos de muestra, las cuales posteriormente se usan para verificar a qué *clúster* pertenecen, pues los datos se eligen al azar.

**Nota: en este ejercicio se obtendrá una muestra de 3 registros de los 150 registros originales.**

Se elige aleatoriamente con cualquier mecanismo el registro 30, el registro 51 y el registro 120, usando la función ***loc*** del *dataframe,* tal como se presenta a continuación:

|  |
| --- |
| indiceMuestra =[30,51,120] muestras = pd.DataFrame (iris\_df.loc[indiceMuestra], columns = iris\_df.keys()).reset\_index(drop =True) |

Dando como resultado

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 4.8 | 3.1 | 1.6 | 0.2 | setosa |
| **1** | 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| **2** | 6.9 | 3.2 | 5.7 | 2.3 | virginica |

Una vez escogidas las muestras, estas se eliminan de la *data* original **iris\_df,**  de los datos de muestra, para que estos no se integren con el resto de la información y para ello se usa el siguiente código.

|  |
| --- |
| iris\_df= iris\_df.drop(indiceMuestra, axis = 0) #axis =0 significa que se eliminan filas |

En una variable **X** se guardan las columnas con valores del *dataset* original, con los cuales se va a entrenar el algoritmo; para esto se usa la función **iloc** sobre el *dataset* original **iris\_df.** Estos datos corresponden a las **variables de longitudes y anchos de los sépalos y los pétalos únicamente**.

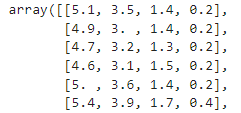
El conjunto de datos no cuenta con una columna de variable dependiente o **Y** en aprendizaje no supervisado, todos son variables dependientes, en este caso es **X.**

|  |
| --- |
| X = iris\_df.iloc[:, [0, 1, 2,3]].values |

Al imprimir **X** una muestra de los resultados es:

**Figura 7**

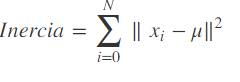
*Variable X consistente en resultados a entrenar con el algoritmo k-means*



Todos los datos son medidas en centímetros de los pétalos y los sépalos, por lo cual no es necesario realizar algún trabajo de transformación en las variables.

Para hallar el valor óptimo de K necesario en K-means se aplicará el **método del codo**, calculando el algoritmo de agrupación para diferentes valores de (**K).**

**El método del codo o método Elbow** se construye calculando la inercia después de aplicar el método K-means a 1 , 2 , 3, …, N *clúster*, siendo la inercia la suma de las distancias al cuadrado de cada objeto del *clúster* a su centroide.



El punto en el que se observa un cambio brusco en los datos de las inercias se tomará como el valor de **K**, la línea forma algo similar a la de un brazo y su codo, el código fuente en K-means se muestra a continuación:

|  |
| --- |
| inercia =[] # suma de cuadrados for i in range (1,20):   algoritmo = KMeans (n\_clústers = i, max\_iter =300).fit(X)  inercia.append(algoritmo.inertia\_) |

Una vez se obtiene la inercia se grafica el vector de inercia con el siguiente código:

|  |
| --- |
| plt.figure(figsize =[10,6]) plt.title('Método del codo') plt.xlabel('Número de clústers') plt.ylabel('Inercia') plt.plot ( list(range(1,20)), inercia, marker='o') plt.show |

Generando la siguiente figura.

Forma

Descripción generada automáticamente

**Figura 8**

*Diagrama de codo*

Se observa que el codo se encuentra entre los valores de 2.5 y 5, se usará por tanto **K=3**.

Definido el valor de **K** ya se podrá aplicar el algoritmo de agrupamiento con el siguiente código fuente:

|  |
| --- |
| algoritmo = KMeans( n\_clústers=3, init ='k-means++', max\_iter=300, n\_init =10) |

**n *clúster*:** se refiere a los *clúster* que se averiguó anteriormente.

**max\_iter:**  número máximo de iteraciones en una misma ejecución.

**n\_init**: número de veces que se ejecutará con diferentes centroides.

Una vez definido el algoritmo y sus parámetros se entrena junto con los datos de la variable **X ,** para realizar esto se usa el comando **fit.**

|  |
| --- |
| algoritmo.fit(X) |

Una vez entrenado el algoritmo se debe revisar los datos de los centroides y las etiquetas obtenidas, estas etiquetas no son más que la identificación del clúster en donde queda la flor.

|  |
| --- |
| centroides, etiquetas = algoritmo.clúster\_centers\_, algoritmo.labels\_ |

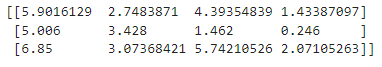
El algoritmo genera centroides  en la variable  algoritmo.cluster\_centers\_ y las etiquetas en la variable algoritmo.labels*\_*

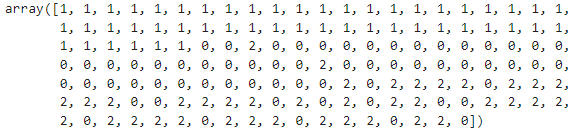
En el caso del ejercicio del ejemplo, el algoritmo arroja  los siguientes centroides y etiquetas.

Los centroides y etiquetas se muestran a continuación.

**Figura 8**

*Centroides y etiquetas*





Indicando los centroides son las coordenadas y las etiquetas representan el *clúster* en el que queda cada una de las observaciones.

## **1.4. Segmentación de conjuntos de datos por atributos compartidos**

Una de las principales aplicaciones del aprendizaje no supervisado es la segmentación de *dataset,* cada *clúster* es mutuamente excluyente y se conoce como segmentos, la técnica de agrupar se conoce como segmentación. En las empresas, la segmentación permite identificar diferentes preferencias de los clientes, el resultado es importante para realizar el mercadeo efectivo de los productos y los servicios.

Entre aplicaciones de segmentación o *clustering* están:

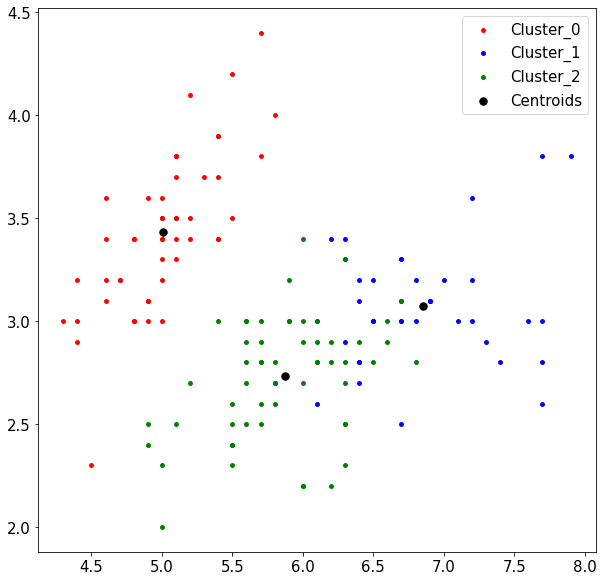
* Identificación de patrones de compra, recomendaciones de películas  o series a los usuarios de *streaming.*
* Detección de fraudes bancarios.
* Detección de atípicos.
* Riesgos de seguros.
* Comportamiento de los pacientes.
* Clasificación de documentos.
* Identificación de síntomas.

En el caso del ejemplo propuesto es conveniente graficar los datos obtenidos en *clúster*, para lo cual se usa el siguiente código de las longitudes y ancho de los sépalos:

|  |
| --- |
| #Aplicando K-means al dataset / creación del clasificador con k-means plt.scatter(X[etiquetas == 0,0], X[etiquetas==0,1], s = 15, c= 'red', label = 'Clúster\_0') plt.scatter(X[etiquetas== 1,0], X[etiquetas==1,1], s = 15, c= 'blue', label = 'Clúster\_1') plt.scatter(X[etiquetas == 2,0], X[etiquetas==2,1], s = 15, c= 'green', label = 'Clúster\_2') plt.scatter(centroides[:,0], centroides[:,1], s = 55, c = 'black', label = 'Centroids')  plt.legend() plt.show() |

**Figura 9**

*Centroides y clúster*



En este caso se han graficado los *clúster* obtenidos, teniendo en cuenta la longitud y ancho del sépalo que son las columnas 0 y 1 del *dataset*, se observa que el *clúster* 0 de color rojo podría ser los datos de la especie setosa, el *clúster* 1 de color azul equivale a la especie virgínica y el *clúster* 2 de color verde corresponde al versicolor.

# **2. Herramientas tecnológicas para el agrupamiento de datos**

Todos los algoritmos de agrupación de observaciones buscan el mismo objetivo, identificar el *clúster*  basado en patrones ocultos en los datos, los  datos en el mismo *clúster* tienen características  parecidas, los puntos  con características  no similares deben estar ubicados en *clúster* diferentes, algunos algoritmos más utilizados son los siguientes:

|  |
| --- |
| **CF02\_2\_tarjetas verticales unidas\_herramientas** |

Continuando con el caso presentado, y como bien se ha visto, en Python se ha usado la librería scikit-learn para obtener los *clúster* **K-means**.

|  |
| --- |
| from sklearn.clúster import KMeans  km = KMeans(  n\_clústers=3, init='random',  n\_init=10, max\_iter=300,  tol=1e-04, random\_state=0  )  y\_km = km.fit\_predict(X) |

En el código fuente la función tiene los siguientes parámetros de entrada:

**n\_*clúster*:** número de *clúster* deseados y obtenidos mediante algún método como el del codo.

**n\_init:** dice cuántas veces va a correr el algoritmo independientemente, con diferentes centroides aleatorios para escoger un modelo final con el más bajo error posible.

**max\_iter:** especifica el máximo número de iteraciones para cada ejecución única.

El algoritmo detiene la ejecución al momento en que los centroides ya no cambian, a pesar de que no se haya alcanzado el máximo número de iteraciones.

Existen otras técnicas para realizar la clusterización, tales como:

|  |
| --- |
| **CF02\_2\_infografía\_técnicas** |

Una librería importante también es **TensorFlow**, que fue liberada por Google, esta contiene cálculos numéricos e inteligencia artificial y fue desarrollada por miembros del equipo de Google y corre en una variedad de plataformas como Python.

# **3. Validación del resultado del análisis**

Lo más importante al usar *machine learning* de algoritmos supervisados o no supervisados es la interpretación de los resultados.  En el caso de la clusterización con k-means  se deben revisar  si los *clúster* encontrados tienen sentido lógico y cumplen con los objetivos, si un *clúster* tiene muy pocos  o muchos datos lo más responsable es volver a ejecutar el análisis con otros centroides por ejemplo.

Una variable importante es la distancia euclídea  definitiva  entre cada punto y su centroide, sus promedios deben tener una variabilidad moderada, si esto no se cumple, lo mejor es repetir el proceso, también es importante medir qué tan separado está un *clúster* de otros.

Muchas veces la validación de los *clúster* es lo más difícil en el análisis de los *clúster* encontrados; pero es necesario hacerlo usando medidas internas y externas.

Existen dos parámetros importantes:

|  |
| --- |
| **CF02\_3\_pestañas horizontales\_parámetros** |

Para ir dando finalización al proceso ya iniciado se deben usar las muestras separadas inicialmente y observar a qué *clúster* pertenecen, además de forma preliminar observar los resultados que predice el algoritmo. Se retoma entonces la prueba original consistente en tres observaciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| **0** | 4.8 | 3.1 | 1.6 | 0.2 | setosa |
| **1** | 6.4 | 3.2 | 4.5 | 1.5 | versicolor |
| **2** | 6.9 | 3.2 | 5.7 | 2.3 | virginica |

|  |
| --- |
| Xmuestras = muestras.iloc[:, [0, 1, 2,3]].values  YMuestraPrediccion = algoritmo.predict (Xmuestras )  print (YMuestraPrediccion) |

[0 2 1] es el resultado de cada una de las muestras, las cuales están ubicadas en el *clúster* 0, 2 y 1 respectivamente y de acuerdo con los resultados anteriores, el *clúster* 0 se refiere a la especie setosa, el *clúster* 1 es virginica y el *clúster* 2 es versicolor.

Con el siguiente código se ubica esta muestra tomada y al graficarla se observa en los círculos más grandes los *clúster* en donde fueron ubicadas las tres muestras.

|  |
| --- |
| figure(figsize =[10,10])  plt.scatter(X[etiquetas == 0,2], X[etiquetas==0,3], s = 15, c= 'red', label = 'Clúster\_0')  plt.scatter(X[etiquetas== 1,2], X[etiquetas==1,3], s = 15, c= 'blue', label = 'Clúster\_1')  plt.scatter(X[etiquetas == 2,2], X[etiquetas==2,3], s = 15, c= 'green', label = 'Clúster\_2')  plt.scatter(centroides[:,2], centroides[:,3], s = 55, c = 'black', label = 'Centroids')  # Se dibujan las muestras iniciales para prueba y son pintadas en el gráfico  plt.scatter(Xmuestras[0,2], Xmuestras[0,3], s = 90, c= 'red')  plt.scatter(Xmuestras[1,2], Xmuestras[1,3], s = 90, c= 'blue')  plt.scatter(Xmuestras[2,2], Xmuestras[2,3], s = 90, c= 'green')  plt.legend()  plt.show() |

Esto conduce a que el resultado de este código es **array([2])**, lo cual significa que queda ubicada en el *clúster* 2, lo cual quiere decir que probablemente esta flor se encuentra clasificada como especie **versicolor.**

También se puede predecir en qué *clúster* quedaría identificada una flor iris, con datos diferentes a los obtenidos en las muestras anteriores; el proceso que se debe hacer es tal como muestra el código, por ejemplo, se tiene una flor con longitud sépalo = 7, ancho sépalo = 3, longitud de pétalo = 5, ancho de pétalo = 2.

|  |
| --- |
| datos\_flor =np.array([1,3,5,2])  datos\_flor\_fm =datos\_flor.reshape(1, -1) #con este comando se le dice a python que cambie el array a una matriz de una sola fila  datos\_flor\_fm  algoritmo.predict(datos\_flor\_rs) |

En resumen K-means es un método de agrupamiento, con el cual no se debe evaluar la precisión, porque se entrena el modelo con datos de etiquetas de clase y por tanto, puede haber diferencia entre las etiquetas verdaderas y las etiquetas pronosticadas.

Es conveniente comparar los gráficos de dispersión originales y con el resultado de K-means para evaluar el rendimiento de agrupación.

**Síntesis**

Como se ha visto en el desarrollo de este ejercicio es necesario que antes de aplicar cualquier algoritmo de ***machine learning*** se realice la exploración de los datos y detectar patrones que luego permitirán identificar cuál es el algoritmo de aprendizaje que se puede usar, el entendimiento de los datos es lo que más debería consumir tiempo.

El algoritmo de aprendizaje no supervisado, específicamente K-means está basado en la identificación de patrones como *clúster* y los datos toman una gran importancia en el aprendizaje automático, conceptualizando la importancia de la ciencia de datos, *bigdata*, tipos de variables y códigos fuente en Python.

Con el algoritmo K-means se pueden revisar patrones en diferentes exploraciones o en una muestra determinada obtenida aleatoriamente y en nuevas muestras diferentes al *daset* inicial. Con Python se hace la exploración de datos usando anaconda y júpiter *lab*, donde se toman las estadísticas descriptivas del *dataset y* se pueden realizar gráficos como **boxplots**, gráficos de dispersión y gráficos de pares o **paiplot.**

Así pues, un resumen de lo visto en el presente componente podrá ser visualizado en el siguiente mapa conceptual:

|  |
| --- |
| **CF02\_3\_pestañas horizontales\_parámetros** |

**C. ACTIVIDADES DIDÁCTICAS**

|  |  |
| --- | --- |
| Descripción de la actividad didáctica | |
| Nombre de la actividad | Aprendizaje no supervisado K-Means |
| Objetivo de la actividad | Afianzar los conocimientos sobre los aspectos básicos del aprendizaje no supervisado K-Means. |
| Tipo de actividad sugerida | Falso-Verdadero |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Carpeta Formatos DI: CF02\_ActividadDidáctica |

**D. MATERIAL COMPLEMENTARIO**

| Tema | Referencia APA del material | Tipo de material  (video, capítulo de libro, artículo, otro) | Enlace del recurso o  archivo del documento o material |
| --- | --- | --- | --- |
| * 1. Selección del algoritmo | Anaconda.documentation. (2022). *Installing on Windows*. <https://docs.anaconda.com/anaconda/install/windows/> | Documento web | <https://docs.anaconda.com/anaconda/install/windows/> |
| * 1. Selección del algoritmo | Kaggle. (2022). *Iris.csv.* Kaggle*.* <https://www.kaggle.com/datasets/saurabh00007/iriscsv> | Página web | <https://www.kaggle.com/datasets/saurabh00007/iriscsv> |
| 1.1.1. Selección del conjunto de datos | Cui, Y. (2020). *The iris dataset - a little bit of History and Biology.* Tds*.* <https://towardsdatascience.com/the-iris-dataset-a-little-bit-of-history-and-biology-fb4812f5a7b5> | Artículo web | <https://towardsdatascience.com/the-iris-dataset-a-little-bit-of-history-and-biology-fb4812f5a7b5> |
| 1.3. Refinamiento del algoritmo de agrupación | González, F. (2019). [*Machine learning: construí tu primer algoritmo inteligente*](https://somospnt.com/blog/58-hello-world-en-machine-learning). Somospnt. <https://somospnt.com/blog/58-hello-world-en-machine-learning> | Artículo web | <https://somospnt.com/blog/58-hello-world-en-machine-learning> |

**E. GLOSARIO**

|  |  |
| --- | --- |
| Término | Significado |
| Aprendizaje automático | rama de la inteligencia artificial, cuyo objetivo es implementar técnicas que permitan a los computadores aprender mediante un proceso de inducción del conocimiento. |
| Aprendizaje automático supervisado | el algoritmo recibe datos de entrenamiento consistente en datos etiquetados. |
| Aprendizaje automático no supervisado | el algoritmo identifica patrones y saca conclusiones de los datos que se le proporcionan. |
| *Clúster* | conjunto de objetos o registros que son similares entre sí. |
| *Clustering* | proceso de dividir un conjunto de objetos o registros en subconjuntos llamados *clúster* que tienen semejanzas. |
| Distancia euclídea | es la longitud de segmento entre dos puntos que definen las observaciones más cercanas para asignarlas a un *clúster*. |
| Inteligencia artificial | sistemas informáticos que pueden aprender como aprende un ser humano. |
| *Machine* *learning* | aprendizaje automático. |
| K-means | lenguaje de alto nivel usado para construir todo tipo de aplicaciones y muy usado en la ciencia de datos. |
| Python | proceso criptográfico que proporciona comunicaciones seguras a través de las redes, haciendo que la información entre extremos se transporte en forma segura mediante el uso de criptografía. |
| Método del codo | método consistente en ejecutar K-means para un *clúster* hasta n *clúster* y graficar la inercia por cada uno, que es la sumatoria de la distancia al cuadrado desde cada observación hasta el centroide, el valor k se toma de la gráfica. |

**F. REFERENCIAS BIBLIOGRÁFICAS**

Fedding the machine (2019). Schematic Diagram of Dataset in Dataverse 4.0. [Imagen]. https://www.feedingthemachine.ai/wp-content/uploads/2019/03/DatasetDiagram.png

Github.com. (s.f.). iris.csv. https://raw.githubusercontent.com/toneloy/data/master/iris.csv

Kaggle. (2022). *Iris.csv.* Kaggle*.* <https://www.kaggle.com/datasets/saurabh00007/iriscsv>

# 

G. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia  *(Para el SENA indicar Regional y Centro de Formación)* | Fecha |
| Autor (es) | Héctor Henry Jurado Soto | Experto temático - contratista | Regional Cauca – Centro de Teleinformática y Producción Industrial | Mayo de 2022 |
| Caterine Bedoya Mejía | Diseñadora instruccional | Regional Distrito Capital – Centro de Gestión Industrial | Mayo de 2022 |
| Carolina Coca Salazar | Asesora metodológica | Regional Distrito Capital – Centro | Mayo de 2022 |
| Julia Isabel Roberto | Correctora de estilo | Regional Distrito Capital – Centro de Diseño y Metrología | Junio de 2022 |
| Miroslava González Hernández | Diseñadora instruccional | Regional Santander - Centro Industrial del Diseño y la Manufactura | Julio 2023 |
| Rafael Neftalí Lizcano Reyes | Responsable Equipo de desarrollo curricular | Regional Santander - Centro Industrial del Diseño y la Manufactura | Julio 2023 |

**H. CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| Autor (es) |  |  |  |  |  |