Proyecto de aprendizaje no supervisado

Equipo 4: Jesús yair Ramírez Islas A01275404, Luis Gerardo Lagunes Najera A01275215 y Sophia Fernanda Valencia Bautista A01745091

INTRODUCCIÓN AL CONJUNTO DE DATOS SELECCIONADO Y JUSTIFICACIÓN DE SU ELECCIÓN

Nosotros decidimos conservar el mismo dataset "*heart_statlog_cleveland_hungary_final.csv*", el cual utilizamos en el proyecto pasado. Únicamente quitamos la última columna que era la que mencionaba si una persona tenía una enfermedad cardiaca.

Hacemos uso de esta base de datos porque así podemos conocer grupos de personas que tienen similares algunas características y eventualmente llevar a algunas conclusiones médicas, además de que consideramos que su dimensión es adecuada.

MÉTODO: K-MEANS

Preparación de datos

Dentro de esta sección importamos las librerías que ocuparemos para realizar el método, al igual que el documento csv, el cual se guarda en la variable dt, y se visualizan las primeras filas con la función $.head()$. Después se utiliza $.info()$ para ver los tipos de datos que contiene cada columna, la cantidad de datos de cada variable y de datos faltantes.

Debido a que trabajamos con un método de aprendizaje no supervisado, eliminamos la última columna que contiene información sobre si una persona presenta una enfermedad cardiaca, al igual que la columna de ID. Luego, utilizamos la función $.describe()$ para observar las medidas de tendencia central y de dispersión.

Implementación y entrenamiento

Para conocer la cantidad de clusters, se crea una ciclo for con el cual se prueba de 1 hasta 10 clusters utilizando K-Means y después se grafica el arreglo que formó el ciclo. Esta gráfica es del número de clusters contra la suma de cuadrados. Esta técnica se llama Codo de jambú, con el cual pudimos conocer que se requieren 2 clusters para el método. Por lo

anterior, utilizamos $KMeans()$ donde colocamos el número de agrupaciones y las iteraciones máximas. Esto se guarda en la variable $cluster$, a la cual se le aplica la función $.fit()$ para que sea al dataframe que tenemos.

Finalmente, añadimos una columna al dataframe con el valor de cluster para cada fila.

Link al código: K-Means

Evaluación del modelo entrenado

Para la evaluación del modelo, se utilizó el Silhouette coefficient para determinar si el número de clusters representaba el mayor valor de este coeficiente. El cual tenía un coeficiente 0.6936.

Conclusiones del método con la base de patrones utilizada

Fue un método con un coeficiente bastante bueno, cuya implementación y tiempo de ejecución fue bastante corto. En nuestros resultados observamos que de los dos clusters el del valor de 0 es en el que se encuentran clasificados la mayoría de de los datos, resultando así aproximadamente 200 datos para el cluster de valor 1.

MÉTODO: K-MEDIODS

Nombre de la persona encargada del método

Yair

Preparación de datos

Al ser el dataset previamente utilizado únicamente removí la columna target para cumplir con las características antes de aplicar el método no supervisado. Sin embargo incluí el uso del comando .info y describe para observar que se hayan removido la columa targen y id correctamente, también para comprobar si existían valores faltantes y las características de los datos.

Implementación y entrenamiento

Para determinar el número de cluster primero utilice el todo del codo de jambu. El cual utiliza la suma de cuadrados para observar el comportamiento para determinados clusters. Para realizar la gráfica se utiliza un ciclo for que aplicó Kmediods con cluster que aumentan en cada iteración, una lista guarda el parámetro inertia lo cual nos indica cómo se comportan los datos según el cluster. Con matplotlib se hizo la gráfica y podemos observar que el valor de cluster adecuado es 3 porque a partir de este valor, la gráfica deja de tener una caída tan drástica. Para comprobar esta hipótesis y tener un parámetro con el

cual evaluar modelo se calculó el coeficiente de Silhouette, el proceso es parecido al anterior solo que en este caso el valor que guardamos en un lista es el que se obtiene con la función silhouette_score, la cual recibe el data frame y los labels que determina el modelo según el número de clusters. El método de graficación es el mismo y con esto podemos observar que efectivamente 3 clusters son lo más adecuado pues es el punto que tiene le mayor score.

Para aplicar el modelo simplemente se indican los parámetros que seguirá el modelo es decir el número de cluster y con .fit se evalúa en el data set. Para finalizar se añaden los valores que representan en que cluster está clasificado cada dato (labels) al dataset original.

Link al código: K-Medoids

Evaluación del modelo entrenado

Se utilizó el coeficiente de Silhouette obteniendo un .4368 algo bastante bajo considerando el modelo anterior.

Conclusiones del método con la base de patrones utilizada

Fue un modelo bastante fácil de aplicar y parece haber imitado un poco los resultados que tenía anteriormente la variable target. Esto tal vez nos habla de una posible manera de predecir este valor. Sin embargo hay que considerar que hay valores en un cluster que podría representar que no son concluyentes los datos para determinar si tiene o no una enfermedad cardiaca.

MÉTODO: HIERARCHICAL CLUSTERING

Nombre de la persona encargada del método

Sophia

Preparación de datos

Para esta sección importé las librerías que utilizaré y el dataframe. Este último lo guardé en una variable llamada $data$ y le apliqué la función $.head()$ que permite visualizar las primeras líneas de la base de datos. Los próximos dos pasos son conocer el tamaño del csv con $.shape$ y observar si no existen datos faltantes con ayuda de la función $.isna.sum()$.

Posteriormente, le removí al dataframe la última columna. La cual es la que mencionaba si una persona tenía una enfermedad cardiaca. Esto lo hice igualando la variable que tenía el dataframe ($data$) a otro en el que ya se encontraban filtradas las columnas. Para comprobar que haya sido así, ocupé la función $.head()$.

Implementación y entrenamiento

Creé una variable llamada $caracteristicas$ en la que guardé los valores de las columnas del dataframe. Esto lo hice con $.values$ y después visualicé cómo es el arreglo de los valores. La siguiente línea de código que escribí consiste en una variable $clustering\_jerarquico$ que contiene la función $linkage()$ que es un criterio que determina qué métrica se va a utilizar para la combinación. En este caso, decidí "ward" que es el más ocupado y minimiza, dentro de los grupos, la suma de las diferencias al cuadrado y a su vez la varianza.

Luego, grafique el dendrograma utilizando $.dendrogram()$ para determinar a qué distancia realizaría el corte de número de clusters. Determiné que a una distancia de 2,000, ya que cortaba las ramas principales (las cuales eran 2). Con esto, creé otro dendrograma en el que se indicaba dicho corte.

Guardé en la variable $clusters$ en la que guardó la función $fcluster()$ donde se coloca la distancia a la que determiné el corte de clusters y el criterio, el cual es la distancia. $clusters$ es un arreglo que contiene los clusters y esto se pudo visualizar en una línea donde se coloca esta variable para desplegar el arreglo.

Por último, se añade como columna el arreglo de $clusters$ al dataframe.

Link al código: Clustering jerárquico

Evaluación del modelo entrenado

Utilicé el coeficiente de Silhoutte para determinar si había seleccionado el número de clusters más adecuado de acuerdo con el puntaje de este coeficiente. Esta métrica es para conocer qué tan bien categorizado está un punto, es decir qué tan bien encaja un dato en un cluster. En este caso tuvo un coeficiente de 0.6968, el cual al estar cerca de 1 indica que están más cerca los datos de su cluster a los otros clusters.

Conclusiones del método con la base de patrones utilizada

Una de las particularidades de este método es que no menciona de qué es cada grupo o qué significan. Además, se puede escoger cualquier número de clusters. Lo que ayuda a escoger esa cantidad es el dendrograma, donde se ven las ramas principales.

Tiene un buen comportamiento para separar los clusters debido a que se tuvo un puntaje de Silhoutte de 0.6968.

MÉTODO: GAUSSIAN MIXTURE MODEL

Nombre de la persona encargada del método

Luis

Preparación de datos

Para la preparación de los datos se procedió a importar la base de datos, después de eso se analizó con un *drop* se eliminó la columna de "*target*" y después se hace lo mismo con "*id*" para poder implementar de manera correcta los métodos y con un head() evaluamos que en efecto se hayan borrado estas variables de nuestra base de datos para poder implementar nuestro modelo.

Implementación y entrenamiento

Para la implementación del modelo se importó la librería *GaussianMixture* para que esta sea la que aplique el modelo, como primer punto evaluamos con *silhouette_coefficients* que tan bueno es nuestro modelo y los números de clusters que serían mejor adoptar para que nuestro modelo sea mucho mejor, a partir de esto nos da un coeficiente de .5404 con la aplicación de 2 clusters.

Procedemos a definir nuestra variable de entrada, en este caso X la cuál se define con todas nuestras variables de nuestra base de datos de la siguiente manera: *X = df.iloc[:,0:11].values*, a partir de esto convertimos la base de datos en varias bases de datos y procedemos a graficar todos los datos de nuestras variables y obtenemos una gráfica con varios puntos del mismos color.

Ahora aplicamos y entrenamos nuestro modelo con *gmm = GaussianMixture(n_components = 2)*, aplicamos n_components como dos ya que eso fue lo recomendado en el *silhouette_coefficients,* con la función *gmm.fit(d)* hacemos una distribución de los dataframe obtenidos con nuestros 2 clusters, entrenamos el modelo con *gmm.predict(d)* y la igualamos con la variable *labels* y finalmente calculamos nuestros resultados y los procedemos a graficar.

Link al código: Gaussian Mixture Model

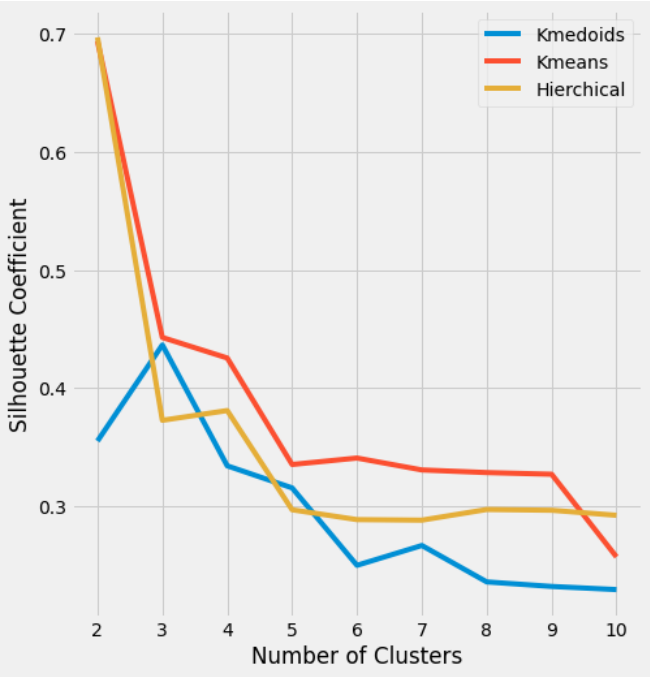
Evaluación del modelo entrenado

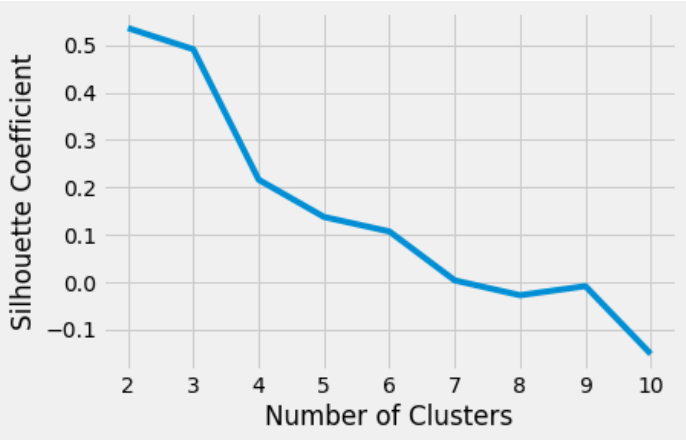
Se utilizó el coeficiente de Silhouette obteniendo un .5404 que se puede considerar bajo pero con un resultado un poco eficaz para su aplicación, ya que este coeficiente nos dice que existe una exactitud entre los puntos y los clusters.

Conclusiones del método con la base de patrones utilizada

El método de Gaussian Mixture Model en general funciona bien si hay una dimensión n grande para poder crear los clusters, es por ello que es muy útil en los métodos de aprendizaje no supervisado y nos puede dar una idea de cómo se están comportando los datos de alguna base de datos de n variables.

COMPARACIÓN DE TODOS LOS MÉTODOS IMPLEMENTADOS





En esta gráfica se comparan los distintos métodos a través del coeficiente de Silhouette, que menciona el mayor valor de cada uno y así se puede observar cuál es el mejor y peor método para clasificar.

Se tiene que el orden de métodos de mayor a menor coeficiente es Hierarchical Clustering (0.6968), K-Means (0.6936), Gaussian Mixture Model (0.5404) y K-Medoids (0.4368).

CONCLUSIONES

Posibles mejoras de cada método y en general

El mejor método es Hierarchical Clustering con un Silhouette coefficient de 0.6968.

K-Medoids: Se podría "jugar" con los parámetros que recibe el algoritmo como la tolerancia relativa con respecto a la norma de Frobenius. Para ver si también se podría implementar el método de PCA para reducir las dimensiones de nuestros datos y poder visualizar cómo es que se segmentan nuestros datos para no tener que hacerlo de forma manual mediante la observación del dataset resultante.

K-Means: Al ser un modelo tan parecido a K-Medoids se podrían hacer las mismas mejoras.

Diferencia entre K-Means y K-Medoids: Se podría decir que K-Medoids es una variación de K-Means por su forma de agrupar los datos. Sin embargo la principal diferencia es que mientras K-Means utiliza como centroides con valores aleatorios que representan el centro del cluster, K-Medoids hace algo similar pero utilizando los valores de una fila del dataset

este se elige tomando en cuenta la distancia promedio de este elemento y los demás. De hecho el principal objetivo de K-Medoids es minimizar esta distancia a comparación de K-Means que busca minimizar el error cuadrático total.

Hierarchical clustering: Una posible mejora podría ser cambiar el método que se ocupa para agrupar los clusters y escoger el número de clusters más adecuado para cada diferente método.

Gaussian Mixture Model: Una de las mejoras que se podrían implementar es el entrenamiento del algoritmo para que logre ser mucho más preciso al momento de dividir los clusters y no ocurran algunos errores de etiquetado y organización en los datos.

CONCLUSIONES INDIVIDUALES

LUIS

El método que implementé, es bastante bueno para entender cómo es que funcionan los clusters en los datos y la manera en que se diferencian, aunque es un modelo muy sensible es de muy buena utilidad al querer observar el comportamiento de los datos, aunque puede tener algunas desventajas, su aplicación es sencilla y fácil de entender.

A mi parecer este proyecto nos enseña muchas cosas a mí en lo particular me hizo razonar y entender un poco más acerca de cómo funciona el aprendizaje no supervisado y en qué casos podemos aplicarlo y también qué métodos podrían funcionar de mejor manera.

Lo que menos me gusto pudo haber sido el tiempo que duró el desarrollo del curso ya que me hubiera gustado profundizar más en este para poder desarrollar aún más este proyecto y obtener resultados muchos mejores en los métodos empleados.

YAIR

El método que implementé, K-Medoids me ayudó a entender mejor cómo funciona K-means pues al tener que describir las diferencias tuve que investigar a detalle cada uno. Lo considero útil pues es un modelo del cual he escuchado mucho. Y parece tener mejores resultados.

Me gustó mucho aplicar el aprendizaje no supervisado y poder observar sus ventajas y desventajas como lo difícil que puede ser visualizar los datos en ciertos casos.

Creo que no menos me gusto fue lo difícil de encontrar una manera de determinar cuál era el mejor pues no es algo tan evidente como el caso del aprendizaje supervisado.

SOPHIA

El método que implementé ayudó a notar o encontrar que existen tres grupos con características en común, además de cuánta es la distancia entre los distintos clusters.

Lo que aprendí del proyecto es entender un poco más el aprendizaje no supervisado, ya que pensaba que cada método mencionaba qué significaba cada cluster. En este caso, con el dataset que seleccionamos, imaginaba que determinaba los grupos de personas con y sin enfermedad cardiaca. Ahora aprendí que depende mucho del algoritmo y en lo que se base para generar los clusters.

Lo que más me gustó fue aprender más sobre dos métodos de clustering, al igual que comprender bien lo que es el aprendizaje no supervisado. Por otro lado, lo que menos me gustó fue que no se tiene una predicción con un significado exacto como en el aprendizaje supervisado.

REFERENCIAS

Algoritmo estándar de K-medias. (s. f.). math.le.ac.uk.

http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html#:~:text=K%2Dmeans%20attempts%20to%20minimize,centers%20(%20medoids%20or%20exemplars)

Amat, J. (diciembre del 2020). *Clustering con Python*. Ciencia de Datos.

https://www.cienciadedatos.net/documentos/py20-clustering-con-python.html

Chavez, R. [Rocio Chavez Ciencia de Datos]. (2021). *Clustering Jerárquico en Python*.

[Archivo de video]. Youtube. https://www.youtube.com/watch?v=iT4xYghl7Rg

Chavez, R. [Rocio Chavez Ciencia de Datos]. (2021). *Clustering Método K-Means en Python*.

[Archivo de video]. Youtube
https://www.youtube.com/watch?v=s6PSSzeUMFk&t=12s&ab_channel=RocioChavezCienciadeDatos

ScikitLearn. (S.F.). 2.3. *Agrupamiento*.

https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

Usman, M. (26 de abril del 2022). *Agrupación jerárquica con Python y Scikit-Learn*.

StackAbuse.
https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/

S.A (18 de noviembre del 2021). *Modelo de mezcla gaussiana*.

https://www.geeksforgeeks.org/gaussian-mixture-model/