Tarea N°1:

Clustering y Reducción de Dimensionalidad

Luis Miranda De la Guarda
Ingeniería Civil Computación, Escuela de Ingeniería, Universidad de O'Higgins
13 de 04 del 2023, Rancagua, Chile
Email: luis.miranda@pregrado.uoh.cl

***Abstract*—Este informe tiene como objetivo utilizar diferentes algoritmos de clustering y evaluar su rendimiento en la base de datos Mice Protein Expression, que incluye niveles de expresión de proteínas medidas en células cerebrales de ratones. Se utilizarán los algoritmos de K-Means, DBSCAN y Clustering Aglomerativo, y se realizarán pruebas con diferentes parámetros. Además, se aplicará PCA para reducir la dimensionalidad de los datos y se comparará su efecto sobre las métricas de rendimiento. Se analizarán cuatro métricas (completeness, homogeneidad, v-medida y silhouette) y se discutirá la capacidad de los algoritmos para manejar outliers y determinar el número de clusters.**

I. Introducción

El clustering es una técnica de aprendizaje no supervisado que se utiliza para agrupar datos en función de su similitud. Los algoritmos de clustering encuentran patrones en los datos y los dividen en grupos, o clusters, que comparten características comunes. En este informe, se utilizarán tres algoritmos de clustering:

1) **K-Means**.
2) **DBSCAN**.
3) **Clustering Aglomerativo**.

El objetivo es comparar su rendimiento en la base de datos Mice Protein Expression, que contiene información sobre la expresión de proteínas en ratones con síndrome de Down y ratones de control. Se analizarán cuatro métricas para evaluar el rendimiento de los algoritmos: completeness, homogeneidad, v-medida y silhouette. Además, se aplicará PCA para reducir la dimensionalidad de los datos y se evaluará su efecto sobre las métricas de rendimiento.

II. Marco Teórico

A. Base de datos Mice Protein Expression Data Set:

La base de datos Mice Protein Expression Data Set es una colección de datos de expresión de proteínas tomados del UC Irvine Machine Learning Repository. Contiene niveles de expresión de 77 proteínas medidas en fracciones nucleares de células en la corteza cerebral. Hay 38 ratones de control y 34 ratones trisómicos (síndrome de Down). Dado que algunos ratones fueron estimulados para aprender y/o inyectados con memantina, hay 8 grupos en total.

B. Clustering:

El Clustering es una técnica de aprendizaje no supervisado que busca agrupar elementos similares en conjuntos llamados "clusters" o "grupos". El objetivo es encontrar una estructura subyacente en los datos sin conocer de antemano las etiquetas de clase o categorías a las que pertenecen los datos.

Existen varios algoritmos de clustering, como K-Means, DBSCAN y Clustering Aglomerativo, cada uno con sus propias fortalezas y debilidades. Estos algoritmos se utilizan para dividir un conjunto de datos en subconjuntos homogéneos en función de una métrica de similitud, como la distancia euclidiana o la similitud del coseno.

C. K-Means:

K-Means es uno de los algoritmos de clustering más comunes. El algoritmo comienza seleccionando aleatoriamente k centroides, uno para cada cluster. Luego, cada punto de datos se asigna al cluster cuyo centroide está más cerca. Después de que todos los puntos han sido asignados, los centroides se recalculan y el proceso se repite hasta que los centroides ya no cambian o se alcanza un número máximo de iteraciones.

K-Means tiene la ventaja de ser rápido y escalable a grandes conjuntos de datos. Sin embargo, puede ser sensible a la elección inicial de centroides y puede producir resultados diferentes según la inicialización.

D. DBSCAN:

DBSCAN es otro algoritmo de clustering común que es útil para encontrar grupos con formas complejas. En lugar de asignar cada punto a un cluster, DBSCAN identifica puntos "núcleo" y agrupa puntos cercanos a estos núcleos en clusters.

DBSCAN tiene la ventaja de no requerir un número predefinido de clusters y es robusto a la presencia de ruido. Sin embargo, puede ser sensible a la elección de parámetros como el radio y el número mínimo de puntos en un cluster.

E. Clustering Aglomerativo:

El Clustering Aglomerativo es un método jerárquico de clustering que comienza considerando cada punto como su propio cluster y luego une los clusters más cercanos hasta que todos los puntos están en el mismo cluster. Este proceso genera un dendrograma que muestra la jerarquía de clusters.

El Clustering Aglomerativo tiene la ventaja de permitir la visualización de la jerarquía de clusters y no requiere una elección predefinida del número de clusters. Sin embargo, puede ser computacionalmente costoso en grandes conjuntos de datos.

F. PCA (Análisis de Componentes Principales):

Es una técnica de reducción de la dimensionalidad que busca transformar los datos en un espacio de menor dimensión mientras se conserva la mayor cantidad posible de información. Esto se logra identificando las direcciones en las que los datos tienen mayor variabilidad y proyectando los datos en esas direcciones. Los componentes principales obtenidos son combinaciones lineales de las variables originales, donde cada componente representa una fracción de la varianza total de los datos. PCA se utiliza comúnmente como una técnica previa al Clustering para reducir la complejidad del dataset.

G. Metricas

Para evaluar la calidad de un algoritmo de agrupamiento, se utilizan varias métricas, en particular las que se usarán en los experimentos son las siguientes:

- **Homogeneity:** Es una métrica que mide la medida en que los clústeres contienen solo puntos de datos que pertenecen a la misma clase. Un resultado de agrupación satisface Homogeneity si todos sus grupos contienen solo puntos de datos que pertenecen a una sola clase.
- **Completeness:** Es una métrica que mide la medida en que todos los puntos de datos que pertenecen a la misma clase se asignan al mismo grupo. Un resultado de agrupación satisface esta métrica si todos los puntos de datos que pertenecen a una clase determinada se asignan al mismo grupo.
- **V-Measure:** Una media armónica de homogeneidad y Completeness que da igual importancia a ambas medidas. Se puede interpretar como un promedio ponderado de las dos métricas.
- **Silhouette Coefficient:** Es una métrica que mide la similitud de un punto de datos con su propio grupo en comparación con otros grupos. Va de -1 a 1, donde un valor más alto indica que el punto de datos coincide bien con su propio clúster y no coincide con los clústeres vecinos.

III. METODOLOGÍA

A. Lectura y preprocesamiento de datos:

Se utilizará la librería Pandas para leer el conjunto de datos, pero la forma de acceder a estos será a través de una descarga directa con el link de descarga de la base de datos, esto se hará con la ayuda de una librería extra que se ha agregado, que se presenta a continuación:

1) urllib.request

Luego con la siguiente función de esta librería, se logra descargar y acceder a la base de datos directamente: **urllib.request.urlretrieve(url, filename)**, donde url es el link directo de descarga y filename es el nombre como se guardará la base de datos en el directorio de trabajo.

Luego de tener la base de datos descargada, como parte del preprocesamiento, la columna 'BCL2 N' se eliminará debido a la gran cantidad de valores NaN que contiene y también se eliminaran todas las filas con valores NaN.

Finalmente, se ha decidido eliminar la columna 'Mouse ID' del tipo 'object', porque no representa información útil para los experimentos, luego las columnas **Genotype, Treatment, Behavior y class** que también son de tipo 'object' se han codificado a valores numéricos, utilizando la función de sklearn **LabelEncoder**, ya que si se decidía eliminar, se podría perder información importante del set de datos.

```
[ ]  1 from sklearn.preprocessing import LabelEncoder
     2 # Creamos una instancia de la clase LabelEncoder y ajustamos a los valores único
     3 le1 = LabelEncoder().fit(df['Genotype'].unique())
     4 le2 = LabelEncoder().fit(df['Treatment'].unique())
     5 le3 = LabelEncoder().fit(df['Behavior'].unique())
     6 le4 = LabelEncoder().fit(df['class'].unique())
     7
     8 # Transformamos los valores de la columna a valores numéricos
     9 df['Genotype'] = le1.transform(df['Genotype'])
    10 df['Treatment'] = le2.transform(df['Treatment'])
    11 df['Behavior'] = le3.transform(df['Behavior'])
    12 df['class'] = le4.transform(df['class'])
```

Fig. 1: Convierte atributos object a int.

B. Análisis Exploratorio de Datos:

Se hará un análisis exploratorio de datos de cada uno de los atributos del set de datos, examinando la distribución de cada columna, a través de histogramas, además se observarán y buscarán datos del tipo outliers, a través de gráficos de caja. Finalmente, se examinará la matriz de covarianza entre las características, para ver que tan relacionadas están cada una de estas entre sí.

```
 1 # Veamos las distribuciones de los datos.
 2 num_cols = df.shape[1]
 3 num_rows = int(np.ceil(num_cols / 6))
 4 fig, axs = plt.subplots(nrows=num_rows, ncols=6, figsize=(20, 4*num_rows))
 5
 6 # Recorrer cada columna del array y graficar su histograma en un subplot diferente
 7 for i, ax in enumerate(axs.flatten()):
 8     if i < num_cols:
 9         ax.hist(df.iloc[:, i])
10         ax.set_title(df.columns[i])
11         ax.set_xlabel("Valor")
12         ax.set_ylabel("Frecuencia")
13
14 # Ajustar los subplots para que se vean bien y mostrar la figura
15 plt.tight_layout()
16 plt.show()
```

Fig. 2: Histogramas de los atributos.

C. Benchmarking de algoritmos de clustering:

Se utilizarán las funciones 'bench k means()' y 'bench clustering2()' proporcionadas en el código de U-Campus para obtener un benchmark de K-Means, DBSCAN y Clustering Aglomerativo. Estas funciones no serán modificadas en absoluto.

D. Entrenamiento de algoritmos de clustering:

Se entrenarán los siguientes algoritmos de clustering con los parámetros especificados en el enunciado de la tarea, para esto se ha dividido el set de datos, dejando en una variable aparte la columna **class**, que corresponde a los labels y eliminándola del set de datos completo.

1) K-Means con inicialización aleatoria, utilizando 8 clusters.
2) K-Means++, utilizando 8 clusters.
3) DBSCAN con épsilon por defecto.

```python
# Primero que todo preparamos los datos
# cortando el dataframe de la siguiente forma
data = np.array(df.drop('class',axis=1))
labels = np.array(df['class'])
```

Fig. 3: División del dataset para entrenamiento y pruebas.

4) DBSCAN con épsilon 0.7.
5) DBSCAN con épsilon 0.2.
6) DBSCAN con épsilon por defecto, agregando outliers a un cluster extra.
7) DBSCAN con épsilon 0.7, agregando outliers a un cluster extra.
8) DBSCAN con épsilon 0.2, agregando outliers a un cluster extra.
9) Clustering aglomerativo, utilizando 8 clusters.

```python
print(90 * "_")
print("init\t\tnum\ttime\tinertia\thomo\tcompl\tv-meas\tARI\tAMI\tsilhouette")

kmeans_random = KMeans(n_clusters=8, init='random')
bench_k_means(kmeans_random, "K-Means Random:", data, labels)

kmeans_plusplus = KMeans(n_clusters=8, init='k-means++')
bench_k_means(kmeans_plusplus, "K-Means++", data, labels)

dbscan_default = DBSCAN()
bench_clustering2(dbscan_default, "DBSCAN", data, labels)

dbscan_07 = DBSCAN(eps=0.7)
bench_clustering2(dbscan_07, "DBSCAN e0.7", data, labels)

dbscan_02 = DBSCAN(eps=0.2)
bench_clustering2(dbscan_02, "DBSCAN e0.2", data, labels)

dbscan_outliers = DBSCAN()
bench_clustering2(dbscan_outliers, "DBSCAN out", data, labels, use_outliers=True)

dbscan_outliers_07 = DBSCAN(eps=0.7)
bench_clustering2(dbscan_outliers_07, "DBSCAN e0.7 out", data, labels, use_outliers=True)

dbscan_outliers_02 = DBSCAN(eps=0.2)
bench_clustering2(dbscan_outliers_02, "DBSCAN e0.2 out", data, labels, use_outliers=True)

agglo = AgglomerativeClustering(n_clusters=8)
bench_clustering2(agglo,"Agg 8", data, labels)

print(90 * "_")
```

Fig. 4: Entrenamiento y prueba de los modelos.

E. Evaluación de resultados:

Se compararán los algoritmos de clustering con base en cuatro métricas, para cada conjunto de datos (original y reducido con PCA):

1) Completeness.
2) Homogeneity.
3) v-measure.
4) Silhouette.

Además, se indicará cuál variante es mejor y cuál es peor según cada métrica. También se analizará el número de clusters obtenidos por DBSCAN y su efecto sobre estas, como también el efecto de considerar o no los outliers de DBSCAN como un cluster extra y su efecto sobre el desempeño. Finalmente, se analizará el efecto de utilizar PCA en el conjunto de datos, y como afecta esto al desempeño de los modelos, comparando cada resultado antes de aplicar PCA y después de haber aplicado PCA.

IV. Resultados

A. Resultados Análisis Exploratorio de Datos

Al observar los histogramas de las diferentes columnas, podemos ver distintas distribuciones, en general se reconocen 4 distribuciones que se repiten a traves del dataset, las cuales son:

1) Distribución Normal.
2) Distribución Asimétrica Positiva.
3) Distribución Asimétrica Negativa.
4) Distribución Bimodal.

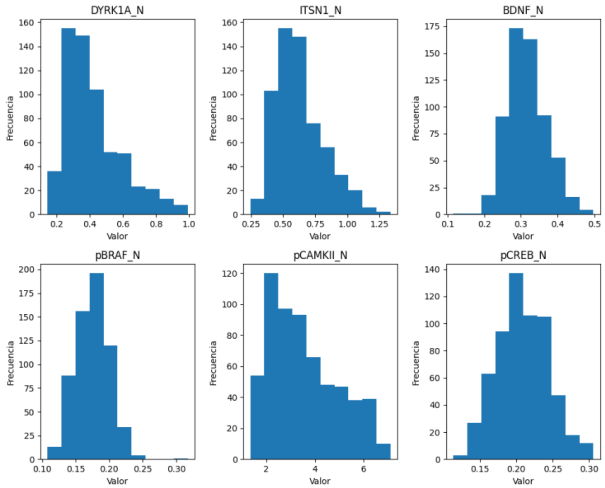


Fig. 5: Muestra de algunos histogramas del dataset.

Además, el estudio de los gráficos de caja arrojo que existen 65 columnas que tienen valores outliers en la medición de los datos, esto se puede generar debido a distintas circunstancias como errores de medición, errores en la entrada de datos, eventos raros o simplemente por la variabilidad natural de los datos. Es importante considerar los outliers al analizar los datos, ya que pueden afectar la precisión de las estadísticas y de los modelos. Por lo tanto, es necesario determinar si los outliers son verdaderos valores extremos o simplemente errores de medición, y tomar medidas apropiadas según el caso.

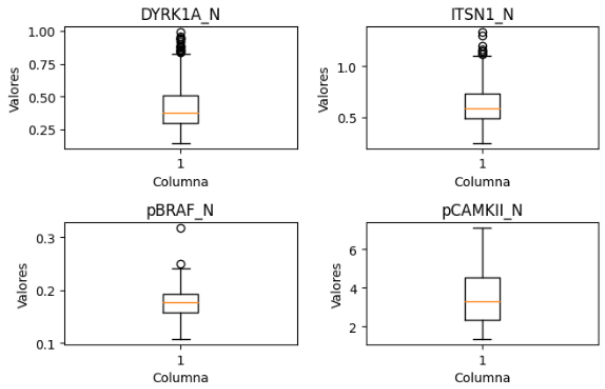
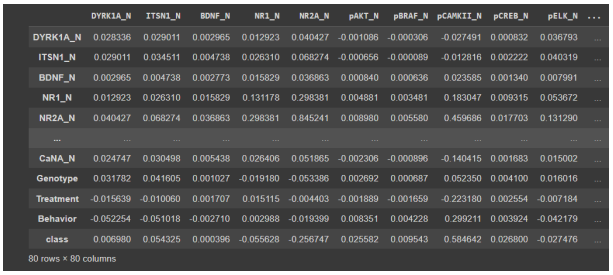


Fig. 6: Muestra de algunos gráficos de caja del dataset.

Por último, se puede destacar en la matriz de covarianza valores absolutos bastante bajos, por lo que se puede deducir que las características no están muy relacionadas entre sí, como también se encuentran covarianzas negativas, donde en esos casos los atributos están relacionados inversamente, como se muestra en la figura 7.



	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pCREB_N	pELK_N	...
DYRK1A_N	0.028336	0.029011	0.002965	0.012923	0.040427	-0.001086	-0.000306	-0.027491	0.000832	0.036793	...
ITSN1_N	0.029011	0.034511	0.004738	0.026310	0.068274	-0.000656	-0.000089	-0.012816	0.002222	0.040319	...
BDNF_N	0.002965	0.004738	0.002773	0.015829	0.036863	0.000840	0.000636	0.023585	0.001340	0.007991	...
NR1_N	0.012923	0.026310	0.015829	0.131178	0.298381	0.004881	0.003481	0.183047	0.009315	0.053672	...
NR2A_N	0.040427	0.068274	0.036863	0.298381	0.845241	0.008980	0.005580	0.459686	0.017703	0.131290	...
...	...	...	...	...	...	...	...	...	...	...	...
CaNA_N	0.024747	0.030498	0.005438	0.026406	0.051865	-0.002306	-0.000896	-0.140415	0.001683	0.015002	...
Genotype	0.031782	0.041605	0.001027	-0.019180	-0.053386	0.002692	0.000687	0.052350	0.004100	0.016016	...
Treatment	-0.015639	-0.010060	0.001707	0.015115	-0.004403	-0.001889	-0.001659	-0.223180	0.002554	-0.007184	...
Behavior	-0.052254	-0.051018	-0.002710	0.002988	-0.019399	0.008351	0.004228	0.299211	0.003924	-0.042179	...
class	0.006980	0.054325	0.000396	-0.055628	-0.256747	0.025582	0.009543	0.584642	0.026800	-0.027476	...

80 rows × 80 columns

Fig. 7: Matriz de Covarianza de las primeras 10 columnas.

B. Resultados benchmarks sin PCA:

Los resultados obtenidos para cada métrica, al entrenar y probar los modelos, con el set de datos, sin haber aplicado reducción de características, se presentan a continuación:

init	num	time	inertia	homo	compl	v-meas	ARI	AMI	silhouette
K-Means Random:	8	0.054s	1433	0.395	0.395	0.395	0.224	0.383	0.206
K-Means++	8	0.418s	1415	0.425	0.425	0.425	0.243	0.414	0.218
DBSCAN	12	0.009s	-1	1.000	0.705	0.827	0.539	0.771	0.717
DBSCAN e0.7	42	0.011s	-1	1.000	0.551	0.711	0.278	0.652	0.434
DBSCAN e0.2	0	0.007s	-1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
DBSCAN out	13	0.009s	-1	0.107	0.365	0.165	0.002	0.113	-0.319
DBSCAN e0.7 out	43	0.008s	-1	0.575	0.433	0.494	0.069	0.427	0.031
DBSCAN e0.2 out	1	0.008s	-1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Agg 8	8	0.053s	-1	0.518	0.521	0.519	0.310	0.509	0.180

Fig. 8: Resultados sin PCA.

C. Resultados benchmarks con PCA:

Los resultados obtenidos para cada métrica, al entrenar y probar los modelos, con el set de datos, después de haber aplicado reducción de características, se presentan a continuación:

init	num	time	inertia	homo	compl	v-meas	ARI	AMI	silhouette
K-Means Random:	8	0.030s	351	0.248	0.251	0.249	0.116	0.234	0.371
K-Means++	8	0.051s	351	0.241	0.244	0.243	0.113	0.227	0.371
DBSCAN	1	0.007s	-1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
DBSCAN e0.7	1	0.007s	-1	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
DBSCAN e0.2	21	0.005s	-1	0.386	0.288	0.330	0.113	0.265	0.320
DBSCAN out	2	0.006s	-1	0.005	0.093	0.010	0.000	0.004	0.367
DBSCAN e0.7 out	2	0.007s	-1	0.005	0.335	0.010	0.000	0.005	0.498
DBSCAN e0.2 out	22	0.006s	-1	0.252	0.222	0.236	0.044	0.185	-0.088
Agg 8	8	0.013s	-1	0.247	0.251	0.249	0.117	0.234	0.332

Fig. 9: Resultados con PCA.

V. Análisis

En esta sección se hará análisis de cada una de las preguntas de investigación del enunciado, utilizando la metodología antes descrita, por lo tanto, se procede a mostrar al mejor y peor modelo por métrica en el set de datos sin PCA y luego en el set de datos con PCA:

A. Análisis de Resultados sin PCA:

1) Completeness:

- **Mejor Modelo:** DBSCAN (default) **0.705**
- **Peor Modelo:** DBSCAN out **0.365**

2) Homogeneity:

- **Mejor Modelo:** DBSCAN (default) y DBSCAN e0.7 **1.000**
- **Peor Modelo:** DBSCAN out **0.107**

3) V-Measure:

- **Mejor Modelo:** DBSCAN (default) **0.827**
- **Peor Modelo:** DBSCAN ou **0.165**

4) Silhouette:

- **Mejor Modelo:** DBSCAN (default) **0.717**
- **Peor Modelo:** DBSCAN e0.7 out **0.031**

Los casos donde no fue posible calcular las métricas de DBSCAN son:

- DBSCAN e0.2
- DBSCAN e.02 out.

Esto, pues DBSCAN se basa en la densidad de los puntos para agruparlos, y para que un punto sea considerado como parte de un clúster, debe haber suficientes puntos cercanos a él.

Si el valor de épsilon es demasiado bajo, entonces el algoritmo no puede encontrar ningún grupo de puntos cercanos que cumpla con el criterio de densidad, y como resultado, no se formará ningún clúster.

Para analizar el efecto de la cantidad de clusters en los resultados de DBSCAN sin aplicar PCA, podemos observar la métrica silhouette para cada valor de épsilon. Esta métrica mide la cohesión y separación de los clusters, podemos observar que con una cantidad de 12 clusters, el algoritmo DBSCAN default logró un alto desempeño, en cambio, al aumentar el valor de épsilon a 0.7 y agregar un cluster extra con outlier, el valor de esta métrica disminuye considerablemente.

Además, podemos ver que para DBSCAN con épsilon 0.2 se obtuvo un valor de -1.0 en todas las métricas, lo que indica que todos los puntos se etiquetaron como ruido. Esto puede deberse a que el valor de épsilon es demasiado pequeño para los datos, lo que hace que todos los puntos estén aislados y no se formen clusters.

B. Análisis de Resultados con PCA:

1) Completeness:

- **Mejor Modelo:** DBSCAN e0.7 out.
- **Peor Modelo:** DBSCAN out.

2) Homogeneity:

- **Mejor Modelo:** DBSCAN e0.2.
- **Peor Modelo:** DBSCAN e0.7 out.

3) V-Measure:

- **Mejor Modelo:** DBSCAN e0.2.
- **Peor Modelo:** DBSCAN e0.7 out.

4) Silhouette:

- **Mejor Modelo:** DBSCAN e0.7 out.
- **Peor Modelo:** DBSCAN e0.2.

Los casos donde no fue posible calcular las métricas de DBSCAN son:

- DBSCAN (default).
- DBSCAN e.07.

Esto puede deberse a que el parámetro épsilon utilizado no es adecuado para la nueva dimensionalidad de los datos. En este caso, podría intentarse ajustar el valor de épsilon o probar con otro algoritmo de clustering que sea más adecuado para los datos reducidos con PCA.

C. Análisis al agregar Outliers.

En el caso de los resultados sin PCA, podemos observar que cuando se consideran los outliers como un cluster extra en DBSCAN, se obtiene un mayor número de clusters en comparación con el DBSCAN sin outliers. Esto puede ser beneficioso para la homogeneidad y completitud, ya que se consideran más grupos en los resultados. Sin embargo, la puntuación de la silueta es mucho más baja para DBSCAN con outliers que para DBSCAN sin outliers. En general, parece que DBSCAN sin outliers es la mejor opción en términos de equilibrio entre las métricas.

En el caso de los resultados con PCA, podemos observar que para DBSCAN sin outliers se obtiene un número significativamente mayor de clusters que para DBSCAN con outliers. Esto puede ser beneficioso para la homogeneidad y completitud, ya que se consideran más grupos en los resultados. Sin embargo, la puntuación de la silueta es mucho más baja para DBSCAN sin outliers que para DBSCAN con outliers. En general, parece que DBSCAN con outliers es la mejor opción en términos de equilibrio entre las métricas para los resultados con PCA.

D. Análisis de aplicar PCA

Se puede observar que, en general, los valores y desempeños de las métricas disminuyeron al aplicar esta técnica, por lo que para esta base de datos, no se recomienda usar reducción de características, ya que se ha perdido información importante respecto a los datos, generando que los algoritmos disminuyan su desempeño.

VI. Conclusiones

En general, los resultados muestran que el uso de PCA disminuye eficiencia de los algoritmos de clustering y no reduce el efecto negativo de los datos ruidosos en los resultados.

En cuanto a los algoritmos de clustering, se observó que DBSCAN y sus variantes tienen un rendimiento bastante alto en comparación con los algoritmos de centroides como K-Means. Además, el uso de un valor de épsilon demasiado grande puede resultar en la identificación de todos los puntos como ruido, mientras que un valor demasiado pequeño puede resultar en una gran cantidad de clusters pequeños y poco útiles.

En cuanto a la inclusión o exclusión de los outliers como un cluster extra, se encontró que la eliminación de los outliers en algunos casos mejoró las métricas, mientras que al agregarlos como cluster extra, los desempeños disminuyeron.

En general, se recomienda el uso de PCA para reducir la dimensionalidad y mejorar la eficiencia de los algoritmos de clustering, pero en esta base de datos no se logró ese objetivo, ya que los desempeños de los diferentes algoritmos se vio disminuido.

Bibliografía

[1] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2011.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, *Data clustering: 50 years beyond K-means*. CRC Press, 2010.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.