Universidad Adolfo Ibañez
Master of Science in Data Science
Nombre: Alfonso Tobar

Reporte 3: Self Organizing Maps

Introducción

El siguiente reporte contiene el análisis realizado para determinar un número de clusters óptimos en un dataset de Tarjetas de Créditos. En este caso queremos segmentar clientes de tal manera que sea posible segmentar clientes que poseen transacciones asociadas a tarjeta de Crédito. El dataset contiene 8950 filas y 18 features. Como parte de la solución se utilizarán Self Organizing Maps, un tipo de red neuronal útil para poder generar clustering. Además se compararán los resultados con Algoritmos de Clustering clásicos como K-Means y una versión más avanzada llamada Fuzzy C-Means.

1. Self Organizing Maps

Los self organizing maps son un algoritmo no supervisado de Machine Learning introducido por el Profesor Treuvo Kohonen en 1980 y es utilizado para producir una representación de baja dimensión, la cual normalmente es de dos dimensiones. EL algoritmo busca representar datos de dimensión más alta pero tratando de preservar la estructura topoógica de éstos. Normalmente el SOM es una red neuronal entrenada utilizando competitive learning.

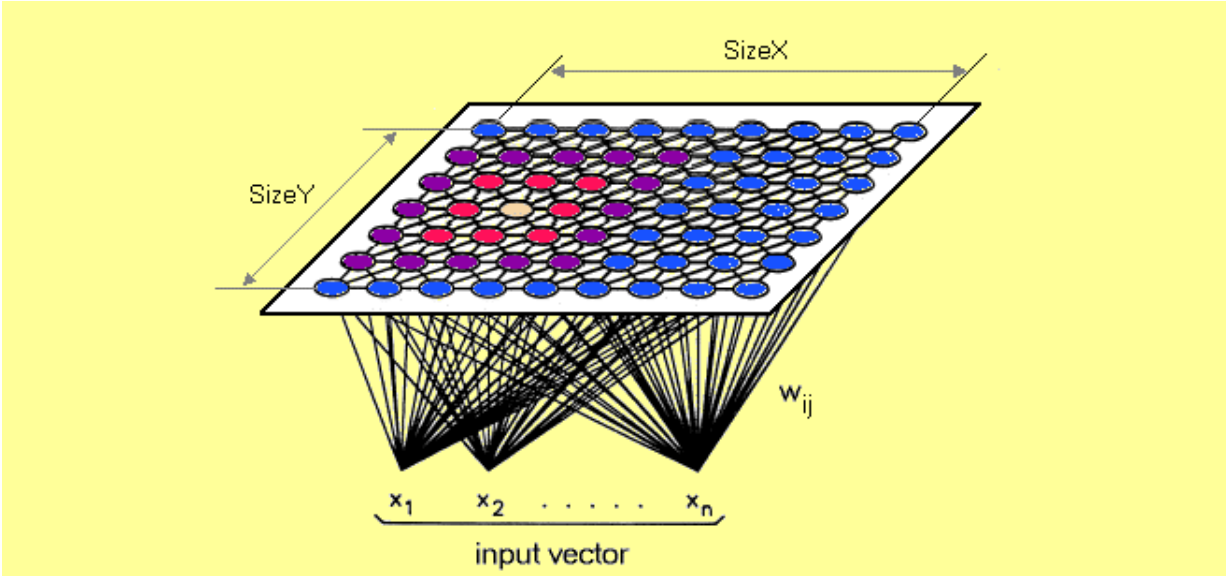

Figura 1: Representación de un Self Organizing Maps.

Normalmente la manera más usual de utilizar el algoritmo es determinar a priori una grilla de neuronas. Estas neuronas irán organizando datos considerando la similaridad de éstos (Ver Figura 1). Para poder determinar los clusters existen al menos dos formas de realizarlo:

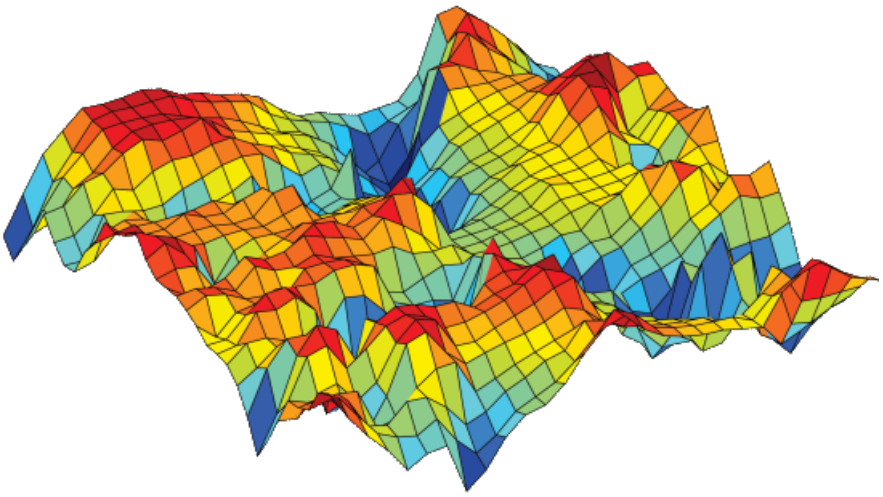

Figura 2: Representación Tridimensional del Self Organizing Map.

Determinando secciones de el espacio en la que varias neuronas representan un cluster. Esto consiste en usar la topología del Clustering para separar clusters (Ver Figura 2). Se espera que montes o valles puedan ser las fronteras de distintas clusters. Mencionaremos que este approach no terminó entregando buenos resultados ya que se hacía muy dificil encontrar separaciones entre clusters. Una segunda forma es estimar una grilla en la cada neurona será un cluster. Este método terminando dando mejores resultados, que de hecho, fueran comparables con los algoritmos más clásicos.

2. Los Datos

Los datos a utilizar corresponden a un dataset extraído desde Kaggle que contiene el comportamiento agregado de clientes que poseen tarjetas de crédito. El dataset intenta resumir el comportamiento a lo largo de la vida del cliente considerando las siguientes features, las cuales serán utilizadas para realizar el proceso de modelamiento.

- **CUSTID** : Identificador del cliente poseedor de la tarjeta de crédito.
- **BALANCE** : Saldo disponible en la cuenta para realizar compras.
- **BALANCEFREQUENCY**: Frecuencia de actualización del saldo en forma de puntaje. 0 representa no actualizado frecuentemente mientras que 1 representa frecuentemente actualizado.
- **PURCHASES** : Cantidad de compras hechas desde la cuenta.
- **ONEOFFPURCHASES** : Monto máximo asociado a una sóla compra.
- **INSTALLMENTSPURCHASES** : Monto de compras hechas en cuotas.
- **CASHADVANCE** : Avances en efectivo realizados por el usuario.
- **PURCHASESFREQUENCY** : Qué tan frecuentemente se realizan compras en forma de puntaje. Valores cercanos a 0 implica que las compras no se realizan de manera frecuente mientras que 1 implica compras frecuentes.
- **CASHADVANCEFREQUENCY**: Frecuencia en la que se paga el avance en efectivo.

- **CASHADVANCETRX**: Número de Transacciones realizadas con el Avance en efectivo.
- **PURCHASESTRX**: Número de compras realizadas.
- **CREDITLIMIT**: Límite de Crédito para el usuario.
- **PAYMENTS**: Monto de pago realizado por el usuario.
- **MINIMUM_PAYMENTS**: Pago mínimo realizado por el usuario.
- **PRCFULLPAYMENT**: Porcentaje de pagos completos realizados por el usuario.
- **TENURE**: Permanencia del cliente con la tarjeta.

3. Análisis Exploratorio

Inicialmente se comenzó revisando los valores nulos del dataset. En la Figura 3 se puede ver que en general está correctamente poblado, donde quizás el Pago Mínimo es la única que tiene valores ausentes que son significativos con un poco más de 300 observaciones.

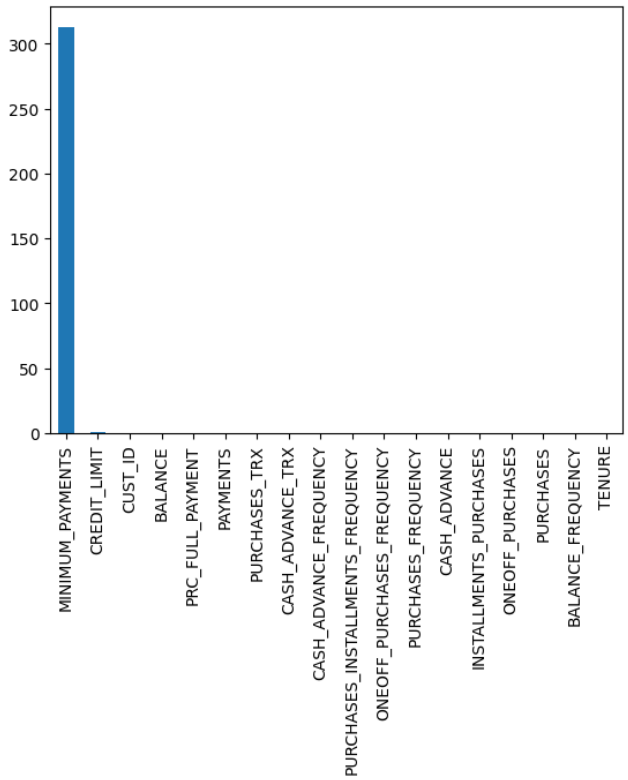

Figura 3: Distribución de valores Nulos.

Además, la Figura 4 muestra que la escala de las variables son bastante distintas, teniendo variables con muy poca variación (como las Frecuencias, que normalmente van entre 0 y 1) y otras con mucha variación como los montos o números de transacciones que pueden llegar a la escalada de decenas de mil.

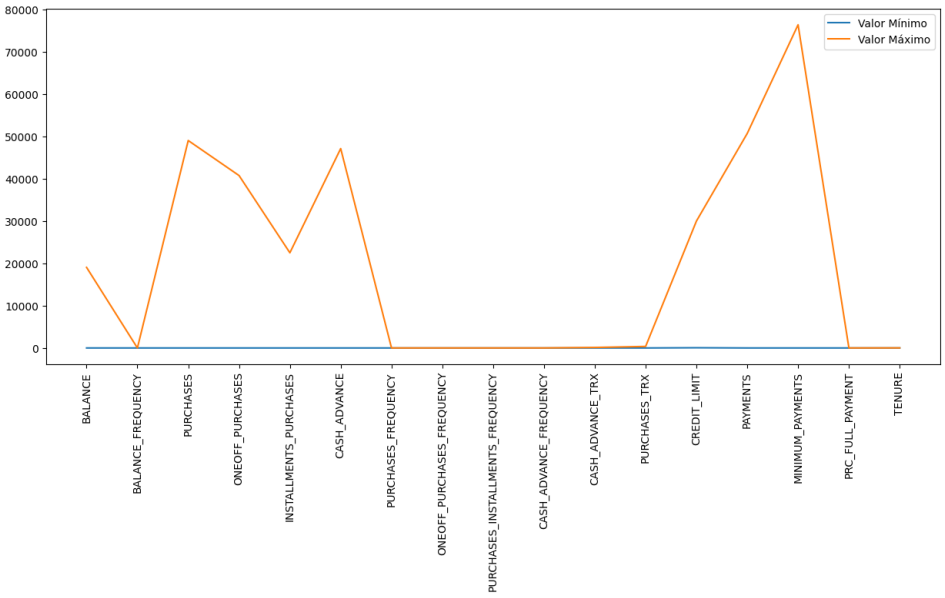

Figura 4: Rango de cada Variable.

Normalmente los algoritmos de distancia dependen fuertemente de distancias, por lo que tener esta alta variabilidad de las escalas puede ser perjudicial. Para mitigar esto se estandarizarán las features para igualar las escalas.

Por otra parte en la Figura 5 se puede observar la distribución de cada variable predictora.

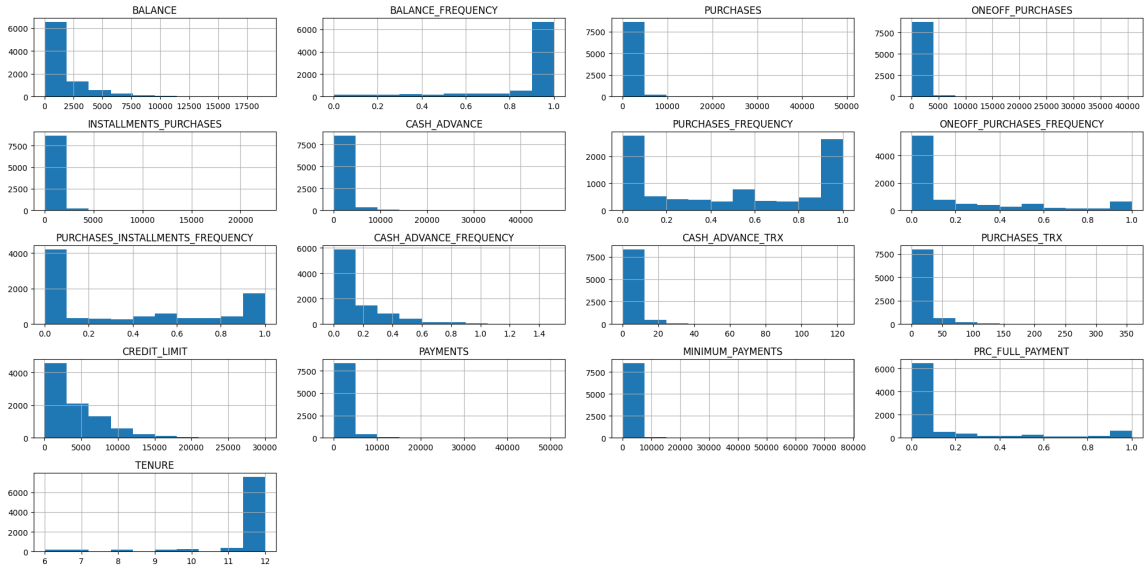

Figura 5: Distribución de Variables Predictoras.

Se puede ver que las distribuciones muestran un alto grado de asimetría presentando colas largas que pueden no ser tan buenas para el modelo. Se considera que aplicando un procesamiento llevando a escala logarítmica puede ser de beneficio. Las variables transformadas se pueden ver en la Figura 6:

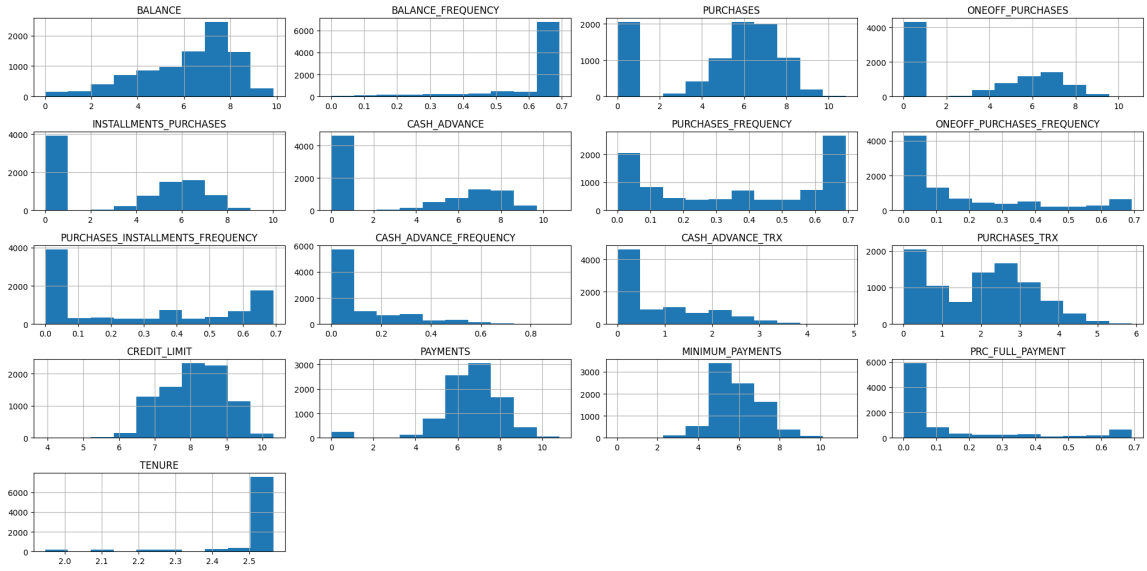

Figura 6: Distribución de Variables Predictoras en Logaritmo.

Luego del preprocesamiento se puede ver que algunas variables tales como Balance, Purchases y Payments tienen una distribución más similar a una Distribución Normal.

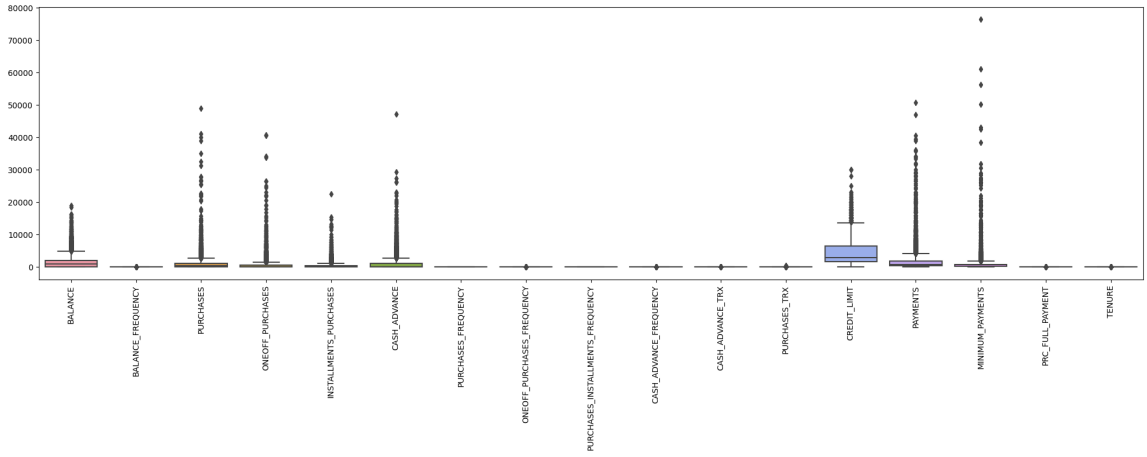

Figura 7: Boxplot de Variables Predictoras.

De hecho se puede apreciar que antes de la transformación Logarítmica hay muchos outliers (ver Figura 7) mientras que post Logaritmo hay una candidad mucho menor (ver Figura 8):

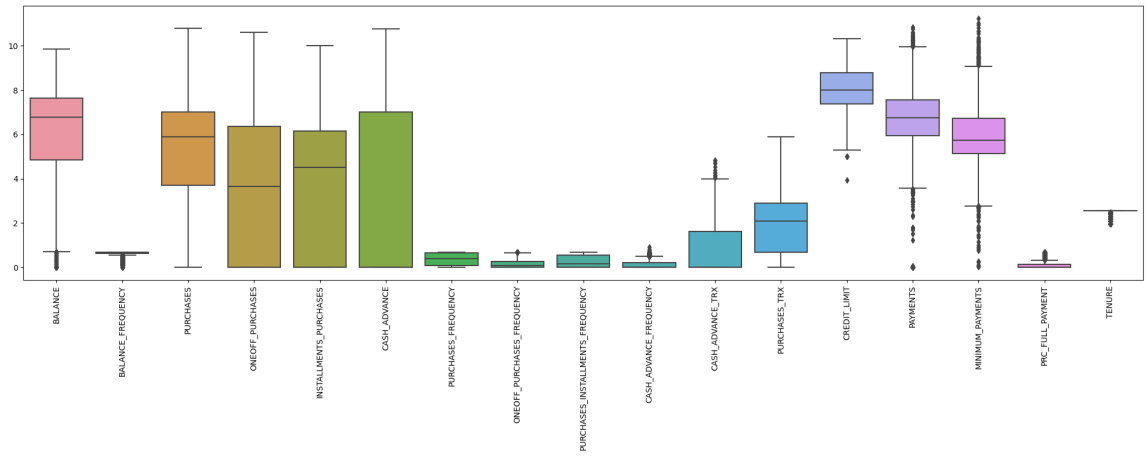

Figura 8: Boxplot de Variables Predictoras en Logaritmo.

Finalmente se determina que para poder experimentar se aplicarán algoritmos de clustering con todos los predictores con y sin transformación logarítmica para poder determinar el efecto de este preprocesamiento.

En este caso no hace sentido eliminar variables. Debido a que queremos considerar el efecto de todas ellas al momento de separar clientes. Esto tiene particular sentido cuando se quiere interpretar el contenido de cada uno de los segmentos resultantes del proceso. En este caso, queremos comparar la capacidad de separar elementos de los distintos modelos por lo que la interpretación de los clusters quedarán fuera del alcance de este reporte.

4. Modelamiento

Inicialmente se comenzó aplicando directamente el Self Organizing Map. Una de las principales ventajas de utilizar SOM es que no es necesario entregar a priori un número de Clusteres. Dada esa ventaja se decidió entrenar un SOM con las siguientes características:

- Malla de 30 x 30.
- Distancias Euclideanas o Mahalanobis.
- learning_rate: 0.05

Luego de entrenar se utlizó la u_matrix para poder determinar posibles separaciones dentro de los datos obteniendo lo siguiente:

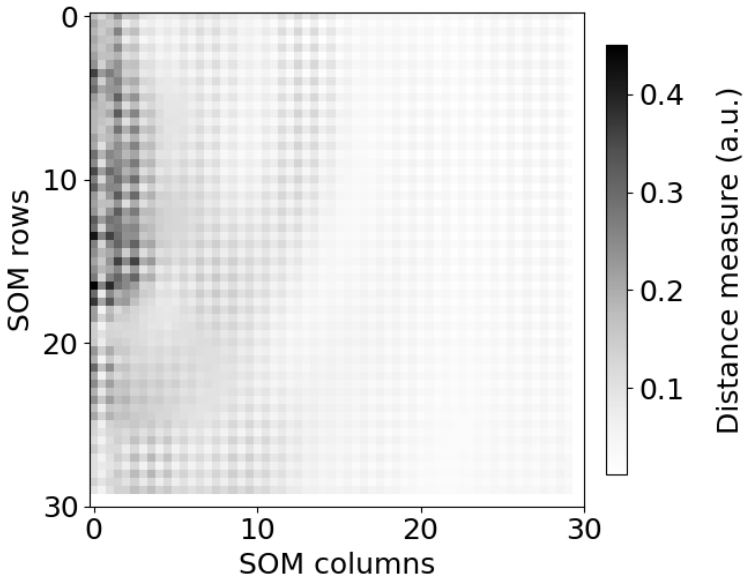

Figura 9: U_Matrix para el entrenamiento de una SOM de 30x30.

Sin importar los distintos Hiperparámetros aplicados al momento de entrenar la Red, los resultados presentaban patrones similares a la Figura 9. Se aprecian pequeñas zonas en negro representando mayor separación de las neuronas, pero no son zonas que permiten claramente delimitar clusters.

Dado los pobres resutados encontradas se decidió utilizar como approach que el número de clusters fuera igual al número de neuronas. De esta manera cada conjunto de puntos asociado a una neurona pasaría a ser un cluster. Para esto se requerirá definir los clusters a priori. Para ello se decidió utilizar una técnica clásica para encontrar el número de clústers que maximiza el Coeficiente de Silhouette. Dado que esta técnica es ineherentemente utilizada con KMeans se realizó con este algoritmo. Es por esto es que el reporte buscará chequear si los Self Organizing Maps pueden separar los datos de una mejor manera que los algoritmos de K-Means y Fuzzy C-Means.

La Figura 10 muestra una curva que representa el Coeficiente de Silhouette para distintos números de clusters.

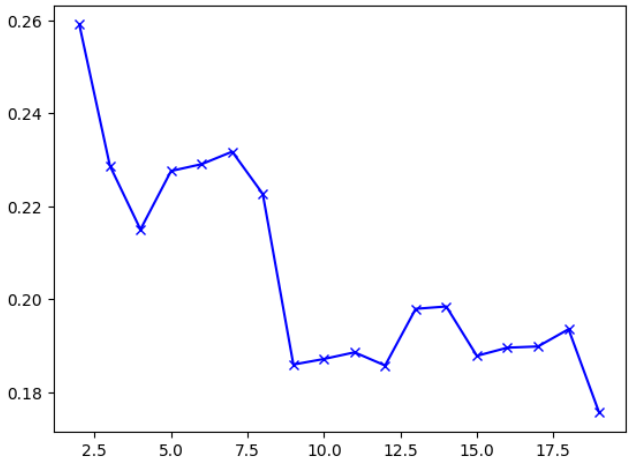

Figura 10: Búsqueda de Clúster Óptimo.

Al chequear los resultados se puede ver que el número de clusters que entrega los mejores resultados son 2 y 7. Obviamente el alto valor dado en 2 clusters es por la simplicidad de la separación. Por lo tanto,

se considerará sólo el estudio de 7 clusters. Esta configuración se entrenará utilizando los 3 algoritmos descritos anteriormente y comparando los predictores sin alterar y con transformación logarítmica.

4.1. Evaluación

La evaluación se realizará mediante el Coeficiente de Silhouette además de una inspección visual para chequear la correcta separación de los clusters. Para mostrar los resultados se mostrará el cosine similarity pero reducida a dos dimensiones mediante PCA.

4.2. Resultados del Modelo

Los resultados del modelo se ven a continuación. Cabe destacar que los modelos se entrenaron utilizando las siguiente configuración:

- 7 Clusters para K-Means.
- Grilla de 7 por 1 para el SOM.
- 7 Clusters para Fuzzy C-Means.

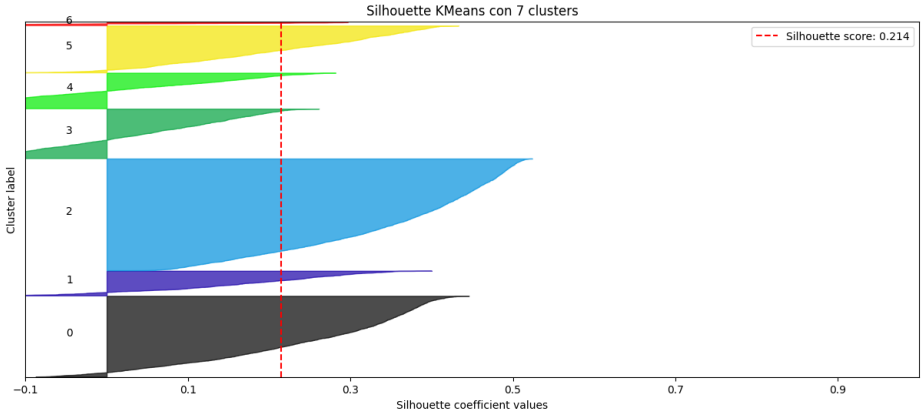

Figura 11: Silhouette K-Means.

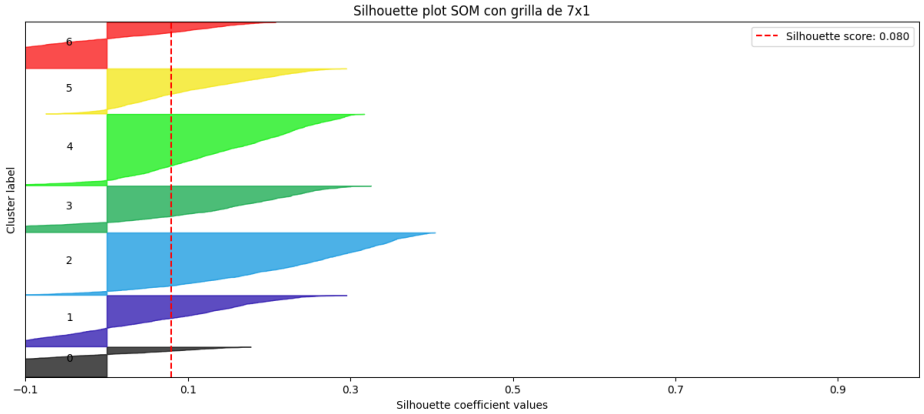

Figura 12: Silhouette SOM.

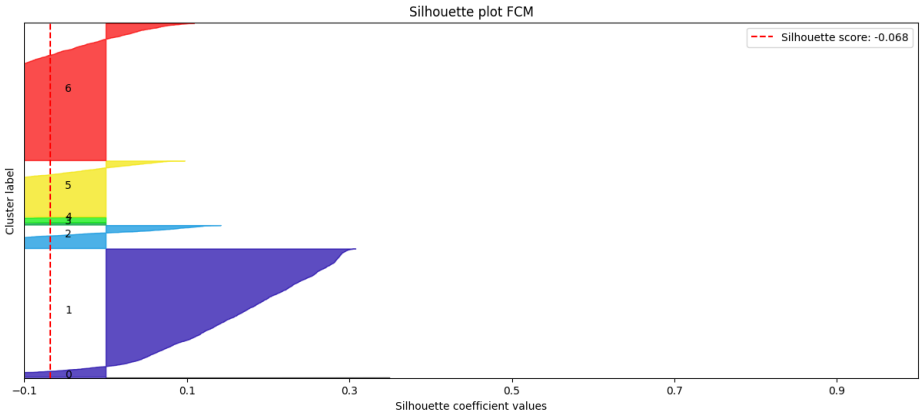

Figura 13: Silhouette Fuzzy C-Means.

Los resultados de las curvas de Silhouette se pueden ver en la Figura anterior. Es posible ver que el Algoritmo de K-Means es el que entrega menos discordancias obteniendo un Coeficiente de Silhouette de 0.214. Por otro lado, los algoritmos de SOM y Fuzzy C-Means entregan resultados bastante pobres de 0.080 y -0.068. Ahora estos resultados deben tomarse con cautela debido a que la manera de generar clusters de SOM y Fuzzy C-Means no tiene por qué ser esférico como si lo es K-Means.

Ahora al chequear la separación de los datos se obtiene lo siguiente:

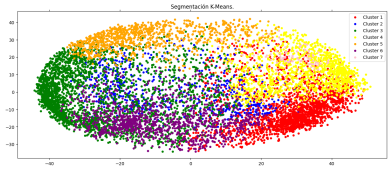

Figura 14: K-Means

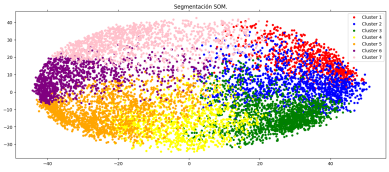

Figura 15: SOM

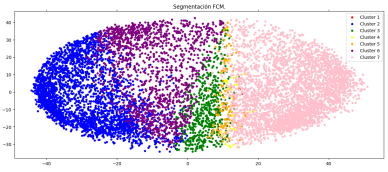

Figura 16: Fuzzy C-Means

Se puede ver que a pesar de los resultados en Silhouoette difieren bastante, la separación de K-Means y SOM se parecen bastante. Por otro lado el Fuzzy C-Means tiende a separar elementos a la largo de la componente principal, lo cual probablemente no es tan óptimo en este caso.

Los resultados utilizando la transformación Logarítmica se pueden ver a continuación:

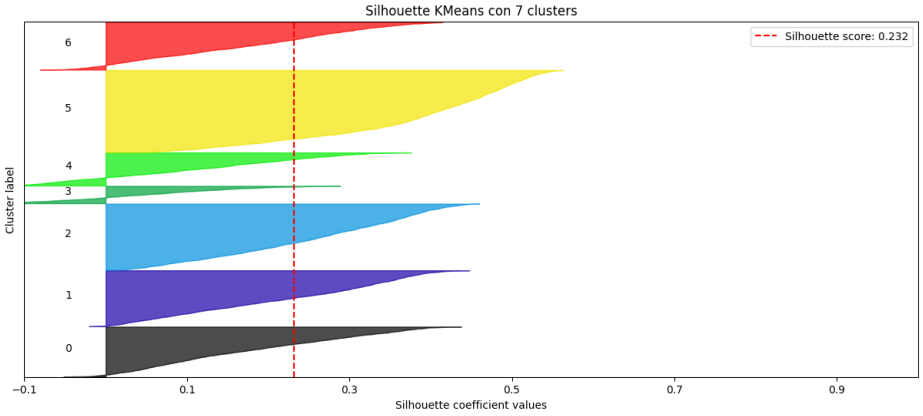

Figura 17: Silhouette K-Means.

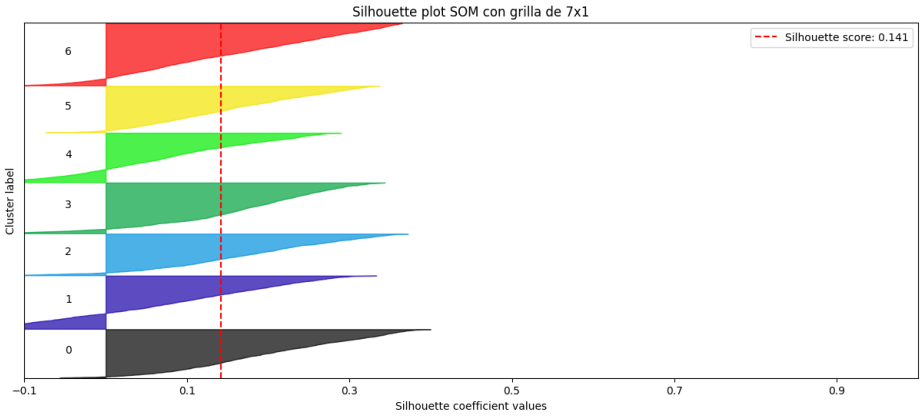

Figura 18: Silhouette SOM.

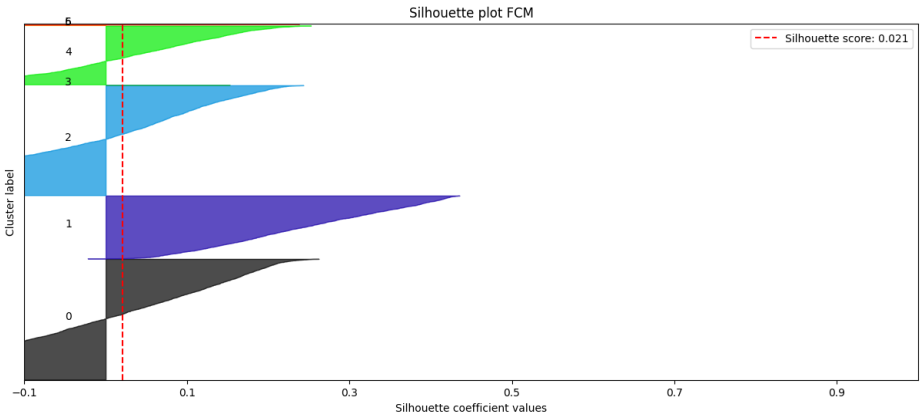

Figura 19: Silhouette Fuzzy C-Means.

El aplicar logaritmo tiene un efecto en el Coeficiente de Silhouette obteniendo 0.232, 0.141 y 0.021 generando una mejora en la cohesión de los clusters.

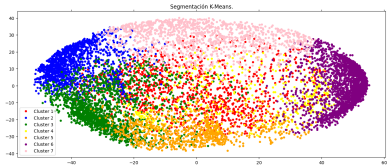

Figura 20: K-Means

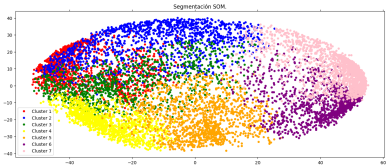

Figura 21: SOM

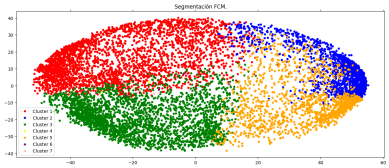

Figura 22: Fuzzy C-Means

Se puede observar una mayor compactación de los clusters. Es posible que esto se deba a que el logaritmo tiende precisamente a compactar los valores que por diferencia de escala y potenciales valores extremos se ven más distanciados.

Es interesante que a pesar de que el coeficiente de Silhouette tiende a entregar mejores resultados para el Algoritmo de K-Means no se ven grandes diferencias con SOM. Si bien los clusters no son para nada idénticos, tienden a hacer una separación similar concentrando clusters en los bordes de la elipse mostrada.

5. Conclusiones

- Se puede ver que el funcionamiento de los Self Organizing Maps son muy similares a los algoritmos de Clustering Convenciones.
- Si bien el hecho de no definir clusters a priori en los Self Organizing Maps parece ser una carácterística poderosa terminó no siendo nada útil ya que normalmente los U Matrix resultantes no mostraban patrones útiles que permitieran determinar el número de clusters de los datos.
- Se puede ver que quizás métricas como el Coeficiente de Silhouette deben ser mirados con cautela. Si bien en todo momento el Coefieciente de Silhouette entregó resutados superiores para K-Means, inspeccionando visualmente se puede ver que K-Means y SOM segmentan los datos de manera muy similar.
- Fuzzy C-Means terminó dando consistentemente los peores resultados. Se puede ver que de alguna forma el Fuzzy C-Means va separando los datos a lo largo de una componente principal. Lo cual no parece ser la manera más apropiada en este caso, y que consistentemente entrega malos resultados en términos del Coeficiente de Silhouette.