Reducción de dimensiones a través de Análisis de Componentes Principales (PCA), Sammon y Autocodificadores (Autoencoders)

Miguel Alexander Sánchez García[1]
[1]Licenciatura en Ciencia de Datos, ESCOM-IPN, CDMX
Analítica y Visualización de Datos
Grupo: [1]5AV1

Resumen — En este reporte, se utilizó el dataset "Breast Cancer Wisconsin Diagnostic" para explorar y comparar tres técnicas de reducción de dimensionalidad: Análisis de Componentes Principales (PCA), Proyección de Sammon y Autoencoders. Cada técnica fue aplicada para reducir las dimensiones del conjunto de datos, manteniendo la mayor cantidad posible de información relevante. Los resultados permitieron evaluar la eficiencia y precisión de cada método en la representación de los datos, destacando las ventajas y limitaciones en términos de interpretabilidad, complejidad computacional y preservación de la estructura original del dataset. Este análisis ofrece una visión integral sobre cómo estas herramientas pueden ser empleadas en problemas de diagnóstico médico, donde la reducción de dimensionalidad es crucial para mejorar la interpretación y procesamiento de datos complejos.

Palabras Clave *– EDA, PCA, Autoencoders, Sammon, red neuronal, covarianza, correlación, varianza explicada.*

I. Introducción

La reducción de dimensionalidad es una técnica fundamental en el análisis de datos, especialmente cuando se trabaja con conjuntos de datos de alta dimensionalidad, como los provenientes del campo médico. En este contexto, el dataset "Breast Cancer Wisconsin Diagnostic" de la UCI Machine Learning Repository ofrece una oportunidad ideal para aplicar y comparar diferentes métodos de reducción de dimensionalidad. Este artículo explora tres técnicas ampliamente utilizadas: Análisis de Componentes Principales (PCA), Proyección de Sammon y Autoencoders. Cada una de estas herramientas busca simplificar la representación de los datos, reduciendo el número de variables mientras se preserva la mayor cantidad posible de información relevante. A través de esta comparación, se pretende destacar las ventajas y desventajas de cada enfoque, proporcionando una guía para su aplicación en problemas de diagnóstico médico y en otras áreas donde la comprensión y visualización de datos complejos es crucial.

El análisis se llevó a cabo utilizando Python, aprovechando librerías como *pandas*, *numpy*, *matplotlib, ScikitLearn y Tensorflow* para la manipulación de datos, y *seaborn* para la visualización.

II. Metodología

Este trabajo presenta y comparara los resultados de tres distintos tipos de reducción de dimensionalidad al dataset "Breast Cancer Wisconsin Diagnostic".

Primeramente se modificaron los nombres de las dimension del dataset, pues no contenian nombres, siendo un total de 32 dimensiones y 569 registros; dichas dimensiones se muestran en la siguiente tabla:

TABLA 1
DESCRIPCIÓN DEL CONJUNTO DE DATOS

Nombre de la Variable	Tipo	Descripción
Diagnosis	Categórico	Diagnóstico (M = maligno, B = benigno)
radius1	Continua	Radio (media de distancias desde el centro hasta los puntos en el perímetro)
texture1	Continua	Textura (desviación estándar de los valores en escala de grises)
perimeter1	Continua	Perímetro
area1	Continua	Área
smoothness1	Continua	Suavidad (variación local en las longitudes de radio)
compactness1	Continua	Compacidad (perímetro² / área - 1.0)
concavity1	Continua	Concavidad (gravedad de las porciones cóncavas del contorno)
concave_points1	Continua	Puntos cóncavos (número de porciones cóncavas del contorno)
symmetry	Continua	Simetría
fractal dimension	Continua	Dimensión fractal (aproximación de "costas" - 1)

Fuente: Elaboración propia a partir de las fuentes consultadas

Cabe recalcar que las dimensiones que tienen un "1" al final de su nombre es por que a su vez hay una version 2 y 3, de esa forma, haciendo un total de 31 columnas, puesto que tambien se eliminó la columna 'ID'.

Posterior a ello se hizo una estandarización de todos los datos numéricos usando la librería StandardScaler de ScikitLearn, asimismo, se hizo uso de la librería de PCA de ScikitLearn para realizar el PCA; se obtuvo el código de un repositorio de github para implementar la proyección de Sammon[1] y se uso la librería de Keras de Tensorflow para poder implementar la red neuronal.

Los resultados de las distintas implementaciones son mostradas en la próxima sección.

[1] https://github.com/tompollard/sammon

III. RESULTADOS Y CONCLUSIONES

El la *figura 1* se muestra las 6 dimensiones con mayor varianza (medida de dispersión que nos dice que tan alejados estan los datos con respecto a la media), mostrados con boxplots, notar que los datos aún no estan estandarizados por lo que no se puede interpretar de forma correcta que tan dispersos estan los datos en esas dimensiones.

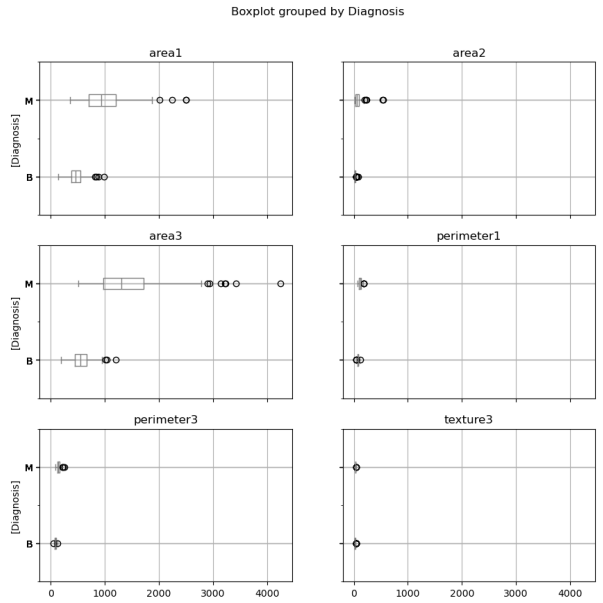

Figura 1. Diagramas de caja de las 6 dimensiones con mayor varianza agrupadas por el diagnosis (M, B) – No estandarizado

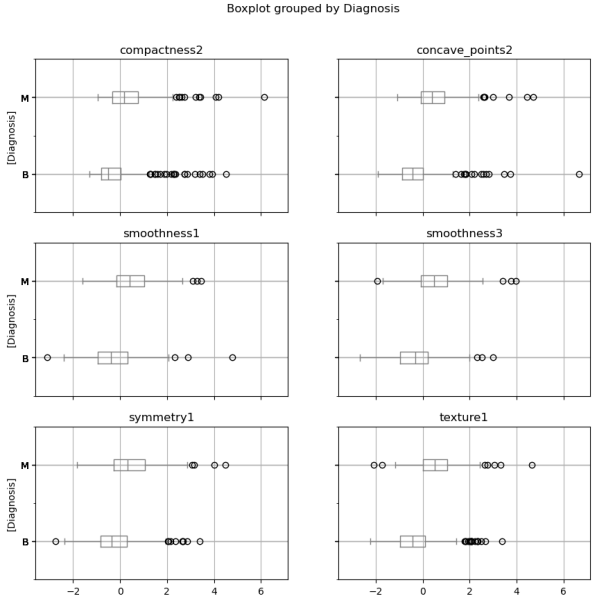

Figura 2. Diagramas de caja de las 6 dimensiones con mayor varianza agrupadas por el diagnosis (M, B) - Estandarizado

En la *figura 2* se vuelven a mostrar las 6 dimensiones con la dimensionalidad más alta pero ya estandarizados. Estos diagramas nos dicen que esos datos son los más dispersos de todo el conjunto de datos, por lo que cuando se haga la reducción de dimensionalidad serán las dimensiones que menos relevancia tengan con respecto a las demás.

En la *figura 3* ya se pueden visualizar los datos en tres dimensiones despues de hacer el PCA, donde los datos en negro son los registros con un diagnistico Maligno y los grises son un diagnostico Benigno. Claramente se puede ver como los datos se separan en dos grandes conjuntos. El entrenamiento del PCA tomo un tiempo de alrededor de 2 seg.

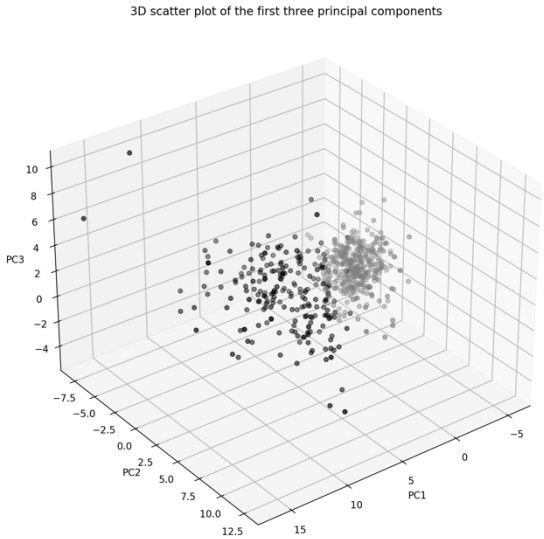

Figura 3. Diagrama de dispersion de las 3 componentes principales del conjuto de datos

Primeramente, la varianza explicada por un componente principal indica qué proporción de la varianza total de los datos originales es capturada por ese componente. Dicho de otra forma la varianza explicada es una medida de cuantá información de los datos originales es capturada por cada componente principal. En este caso la primera componente es la que más informacion captura, pues representa alrededor del 44.3% del conjunto de datos, mostrado en la *tabla 2.*

TABLA 2
VARIANZA EXPLICADA DE LAS 10 COMPONENTES PRINCIPALES

	Principal Component	Explained Variance
0	PC1	0.442720
1	PC2	0.189712
2	PC3	0.093932
3	PC4	0.066021
4	PC5	0.054958
5	PC6	0.040245
6	PC7	0.022507
7	PC8	0.015887
8	PC9	0.013896
9	PC10	0.011690

Fuente: Elaboración propia a partir del código anexo

La *figura 3* nos muestra un gráfico llamado Spree Plot, dicho gráfico nos muestra las primeras 3 componentes principales mostradas en la *tabla 2* en forma de gráficos de barras, y la linea superior es la varianza explicada acumulada, dicho de otra forma, es la suma de la varianza que tiene por debajo más la anterior, cabe

recalcar que el valor máximo al que se debe llegar es a 1, o 100%, y podemos ver que con las primeras 3 componentes principales podemos capturar y mostrar con certeza más del 70% de la información del conjunto de datos.

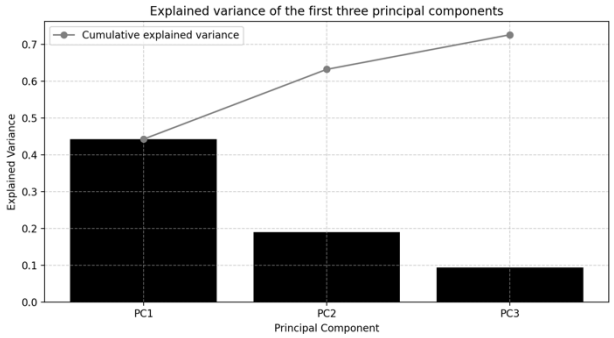

Figura 4. Spree Plot de la varianza explicada de las tres principales componentes

La correlación o el coeficiente de Pearson nos dice que tan relacionadas estan 2 variables, toma valores entre [-1, 1], por lo que entre más acercado al cero hay una menor relación entre las variables. La figura 5 nos muestra una circunferencia de radio uno donde cada vector representa la correlación de cada dimensión original con respecto a las dos primeras principales componentes, entre más cercano a la circunferencia, más correlación. En este caso 'perimeter1', 'perimeter2', 'area1, y 'area2', son las dimensiones que más estan relacionadas con la componente principal 1.

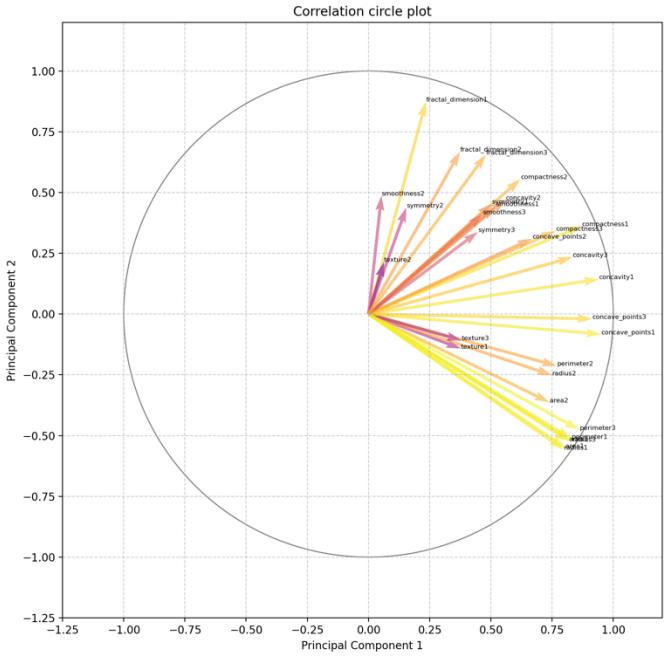

Figura 5. Circle Plot de la correlación de las dimensiones originales con respecto a los dos componentes principales.

La *figura 6* nos muestra algo parecido a la *figura 3*, con la diferencia de que no se toma en cuenta la tercera componente principal, por lo que de cierta forma es como verlo en 2 dimensiones, e igual claramente se pueden ver dos grupos.

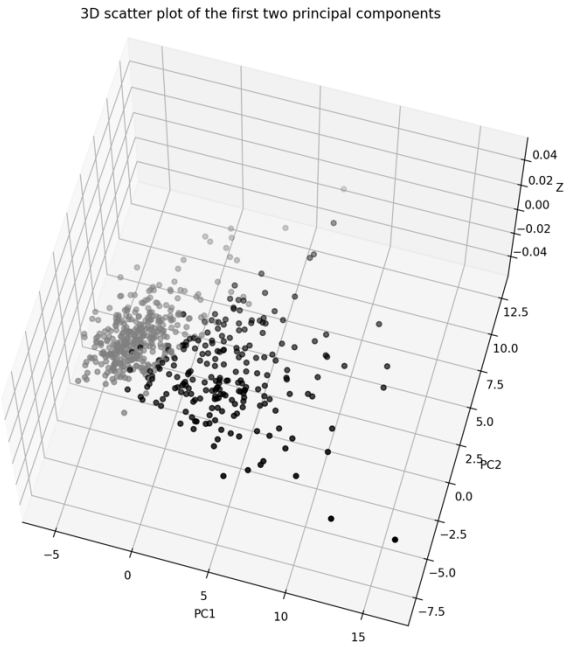

Figura 6. Gráfico de dispersion de las dos componentes principales.

Para la realización de la proyección de Sammon, despues de implementar la función, que tomo un tiempo de alrededor de 2 seg, nos devuelve las tercias ordenadas para observar los datos reducidos en dimensionalidad vistos en la *figura 7.* Podemos ver como tiene una semenjaza alta con la gráfica de la *figura 3,* donde igual podemos ver dos grandes grupos, ademas de que el rango de valores es similar.

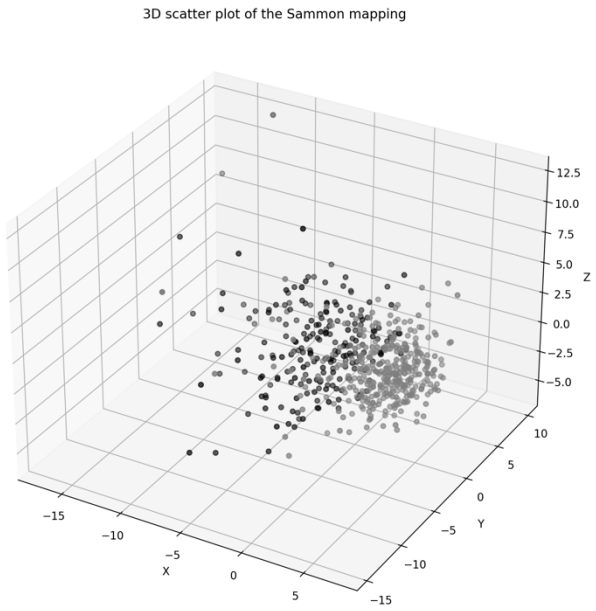

Figura 7. Gráfico de dispersion de la Proyección de Sammon

En la *figura 8*, igual se hizó de la proyección de Sammon, pero esta vez se usaron los datos de las primeras 10 componentes principales calculadas anteriormente y podemos ver que tiene una similitud alta con respecto a la *figura 7.*

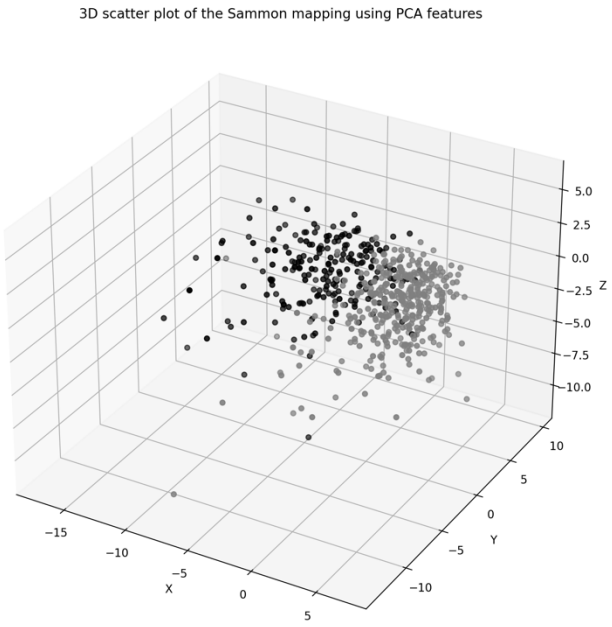

Figura 8. Gráfico de dispersion de la Proyección de Sammon (Usando las primeras 10 compomentes principales)

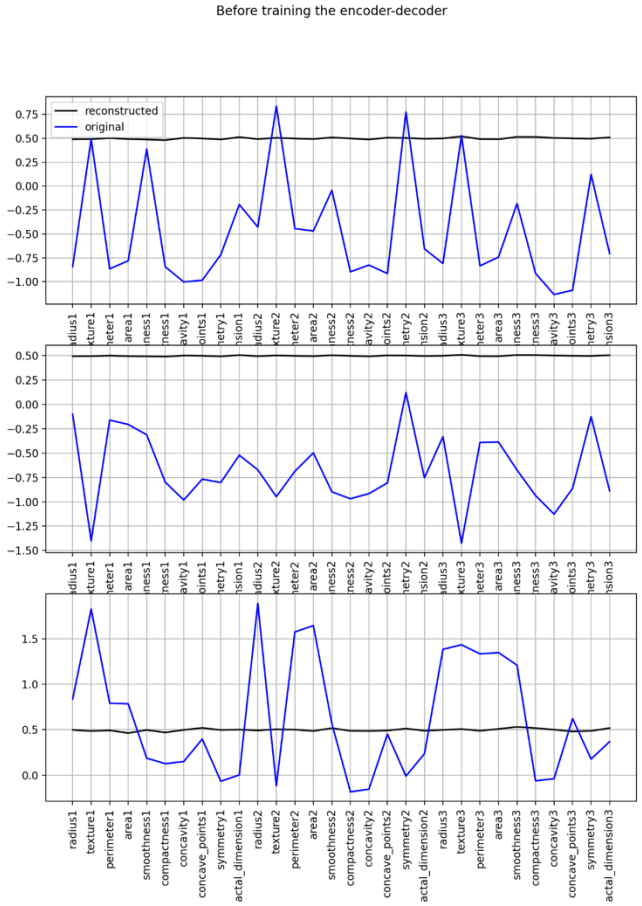

Figura 9. Gráfico de Linea de reconstrucción de datos con respecto a datos originales.

Para el autoencoder, antes a su entrenamiento (mismo que tardó alrededor de 1 min con 4 seg) revisamos como realmente no nos puede reconstruir los datos correctamente puesto que no esta entrenado, esto lo podemos ver en la *figura 9,* donde se toman muestras aleatorias para representar el espectro de los datos originales en 3 dimensiones (linea azul) y lo que construye la red neuronal (linea negra), y podemos obsevar como aún no reconstruye nada, pero es correcto por que se compilo de forma exitosa la red neuronal.

La *figura 10* nos muestra los valores que la función de pérdida va tomando con respecto a las Epoch (épocas) que va teniendo el entrenamiento de la red neuronal, de forma ideal, lá pérdida es menor conforme aumentan las Epoch, situación que se ve como ocurre en la *figura 10,* esto nos quiere decir que tuvo un entrenamiento correcto la red neuronal.

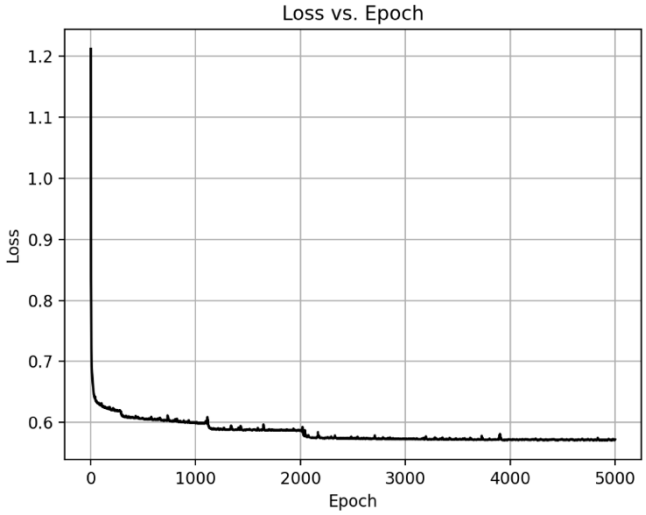

Figura 10. Historigrama de los valores de la función de pérdida con respecto a su epoch.

Despues del entrenamiento ahora si podemos revisar como la red neuronal puede reconstruir de forma correcta los datos originales, mostrada en la *figura 11*, por lo que hacemos la misma gráfica que en la *figura 9.* Podemos ver como la red neuronal reconstruye de forma muy similar a los datos correctos.

Ahora si podemos observar como los datos graficados en 3 dimensiones se asemejan igual a la proyección de Sammon y al PCA, pero si hay diferencias con respecto a los otros dos métodos; datos mostrados en la *figura 12*.

Finalmente, los 3 métodos lográn reducir de forma similar la dimensionalidad del conjunto de datos original, cada una con sus ventajas y desventajas, siendo el PCA el que más ventajas contiene, pues podemos ver el nivel de relevancia que tiene cada componente, cosa que no se puede realizar con los otros dos métodos, y para la red neuronal, especificamente para este problema no obtuvo los mejores resultados, tiene mejores usos para eliminar ruido de un conjunto de datos más dirijidos a imágenes.

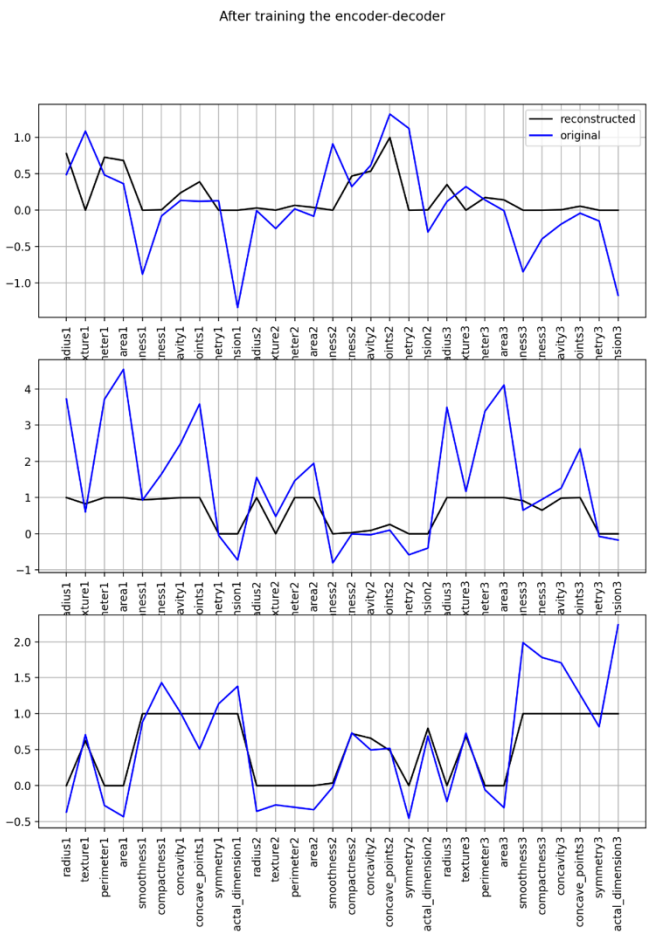

Figura 11. Gráfico de Linea de reconstrucción de datos con respecto a datos originales. (Red Neuronal entrenada)

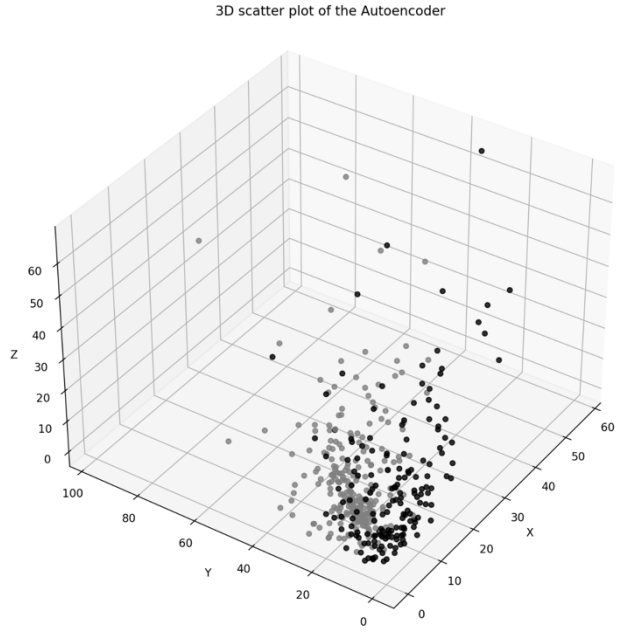

Figura 12. Gráfico de dispersion de las caráctericas despues del entrenamiento del Autoencoder.

IV. REFERENCIAS

[1] Towards Data Science. (n.d.). *A step-by-step introduction to PCA*. Towards Data Science. https://towardsdatascience.com/a-step-by-step-introduction-to-pca-c0d78e26a0dd

[2] Chouinard, J. C. (n.d.). *PCA plot visualization in Python*. J. C. Chouinard. https://www.jcchouinard.com/pca-plot-visualization-python/

[3] Nirpy Research. (n.d.). *PCA correlation circle*. Nirpy Research. https://nirpyresearch.com/pca-correlation-circle/

[4] Pollard, T. (n.d.). *Sammon mapping implementation*. GitHub. https://github.com/tompollard/sammon

[5] Ekamperi, G. (2021, January 21). *Encoder-decoder model*. Ekamperi Blog. https://ekamperi.github.io/machine%20learning/2021/01/21/encoder-decoder-model.html

[6] Saivaran, K. (n.d.). *Dimensionality reduction using Keras auto-encoder*. Kaggle. https://www.kaggle.com/code/saivarunk/dimensionality-reduction-using-keras-auto-encoder

[7] TensorFlow. (n.d.). *Autoencoder*. TensorFlow. https://www.tensorflow.org/tutorials/generative/autoencoder

[8] IBM. (n.d.). *¿Qué es una red neuronal?* IBM. https://www.ibm.com/mx-es/topics/neural-networks#:~:text=Una%20red%20neuronal%20es%20un,opciones%20y%20llegar%20a%20conclusiones