Análisis Comparativo de Técnicas de Reducción de Dimensionalidad y Clasificadores: PCA, Autoencoders, k-NN y Perceptrón Multicapa

Jordi Blasco Lozano[1]

Resumen

Este trabajo aborda la práctica 3 en la asignatura de Fundamentos del Aprendizaje Automático, comparando técnicas de reducción de dimensionalidad (PCA y Autoencoders) combinadas con dos clasificadores (k-NN y Perceptrón Multicapa). Se evalúa el rendimiento sobre cinco datasets mediante validación cruzada 5-fold estratificada, empleando pruebas estadísticas de Wilcoxon para validar diferencias significativas.

1. Introducción

La reducción de dimensionalidad es un componente esencial en el aprendizaje automático moderno, permitiendo mitigar la maldición de la dimensionalidad, reducir el coste computacional y eliminar redundancias en los datos. Esta práctica investiga dos enfoques fundamentales: **PCA** (Principal Component Analysis), una técnica lineal clásica que maximiza la varianza preservada, y **Autoencoders**, arquitecturas neuronales capaces de aprender representaciones no lineales mediante compresión-reconstrucción.

Complementariamente, se evalúan dos clasificadores de naturaleza distinta: **k-NN** (k-Nearest Neighbors), un método no paramétrico basado en proximidad local, y **MLP** (Multi-Layer Perceptron), una red neuronal feedforward que aprende fronteras de decisión complejas mediante retropropagación. La combinación sistemática de estas técnicas sobre los cinco datasets permite analizar cuándo la reducción de dimensionalidad es beneficiosa y qué tipo de clasificador se adapta mejor a cada escenario.

2. Datasets Utilizados

Se han empleado cinco datasets con características diversas, permitiendo evaluar la generalización de las técnicas:

1. Breast Tissue (106 instancias, 9 características, 6 clases): Dataset médico de impedancia eléctrica en tejido mamario, clasificando tipos de tejido (carcinoma, fibroadenoma, mastopía, tejido glandular, conjuntivo, adiposo). Presenta alta dimensionalidad relativa ($d/N \approx 0.085$) y clases con solapamiento conceptual, dificultando la separación lineal.

2. Cinema (1500 instancias, 14 características, 2 clases): Dataset binario sobre ocupación de salas de cine según características temporales, meteorológicas y de tráfico. Con 1500 instancias bien balanceadas, representa un escenario favorable para clasificación con datos suficientes y estructura clara.

3. Spotify (50,000 instancias, 17 características, 10 clases): Dataset masivo de clasificación de géneros musicales basado en características acústicas extraídas por Spotify (danceability, energy, loudness, speechiness, acousticness, etc.). La alta dimensionalidad ($d = 17$) combinada con 10 géneros genera solapamientos complejos entre categorías musicales cercanas (e.g., rock-metal, jazz-blues).

4. Wine Quality (1600 instancias, 11 características, 6 clases): Dataset sobre calidad del vino basado en pruebas fisicoquímicas (acidez, azúcar, pH, alcohol, etc.). Originalmente un problema de regresión (puntuación 3-8), se trata aquí como clasificación. Es un dataset difícil donde las fronteras entre calidades adyacentes son difusas.

5. Zoo (101 instancias, 16 características, 7 clases): Dataset clásico de clasificación de animales en 7 tipos (mamíferos, aves, reptiles, peces, anfibios, invertebrados, insectos) mediante 16 atributos binarios (pelo, plumas, huevos, vuela, etc.). Con solo 101 instancias, representa un escenario de datos escasos donde la reducción puede causar pérdida crítica de información.

3. Metodología

3.1. Preprocesamiento y Corrección de Datos

Durante la implementación se identificaron y resolvieron dos problemas que afectaban la integridad de los datasets:

Problema 1: Inconsistencia en columnas de clase. En breast_tissue, la columna `class` aparecía como primera columna en lugar de última, violando el formato estándar. En *cinema*, existían dos columnas redundantes (`occupancy_class` y `class`) con información idéntica. Esto causaba errores al procesar automáticamente los archivos CSV. Por ello, normalicé los CSV unificando el nombre de la clase a `class`, eliminando columnas redundantes en *cinema* y reubicando la clase al final para garantizar un formato uniforme.

Problema 2: Colapso de salidas en Autoencoder. El uso inicial de activación ReLU provocó representaciones latentes nulas ("neuronas muertas"), especialmente en *breast_tissue*. Para solucionarlo, cambié la activación a `tanh`, que preserva mejor la información al mapear valores en $(-1, 1)$, y aumenté `max_iter` a 2000. Esta configuración se aplicó uniformemente a todos los datasets para garantizar la coherencia experimental.

3.2. Técnicas de Reducción de Dimensionalidad

PCA (Principal Component Analysis): Se aplica PCA estándar preservando **60% de varianza acumulada**. Esta es una reducción bastante fuerte que probablemente cause una pérdida de rendimiento en los datasets que tienen pocas dimensiones, ya que cada característica original suele ser importante. Al quedarnos solo con el 60% de la varianza, el algoritmo selecciona las componentes principales más relevantes y descarta el resto. Esto simplifica los datos, pero conlleva el riesgo de perder detalles necesarios para una buena clasificación.

Autoencoder Neuronal: Hemos utilizado la función `MLPRegressor` de la librería `scikit-learn` para configurar una red neuronal que aprende a comprimir los datos reduciendo sus dimensiones a la mitad. La idea es forzar a la red a pasar la información por un "cuello de botella" (una capa con menos neuronas), obligándola a quedarse solo con lo más importante para poder reconstruir el dato original después. Para asegurar que funcione bien y no pierda información valiosa (evitando neuronas "muertas"), hemos ajustado la configuración interna usando funciones suaves (`tanh`) y permitiendo un entrenamiento más largo. Una vez entrenada la red, nos quedamos con la información comprimida de la capa central.

3.3. Clasificadores

k-Nearest Neighbors (k-NN): Clasificador no paramétrico que asigna la clase mayoritaria entre los k vecinos más cercanos según distancia Euclidiana. Hiperparámetro k optimizado mediante búsqueda en $k \in \{1, 3, 5, 7, 9, 11\}$ usando validación cruzada anidada. Con $k = 1$, el método se vuelve determinista (asigna clase del vecino más próximo); valores mayores suavizan fronteras de decisión pero pueden diluir información local.

Multi-Layer Perceptron (MLP): Es una red neuronal artificial clásica que aprende relaciones complejas y no lineales en los datos. Hemos utilizado una arquitectura con una capa oculta, probando diferentes tamaños (50, 100 o 200 neuronas) y funciones de activación (ReLU o tangente hiperbólica) para encontrar la mejor configuración. El entrenamiento se realizó mediante el optimizador Adam, ajustando los pesos de la red para minimizar el error de clasificación, permitiendo hasta 2000 iteraciones para asegurar que el modelo converja adecuadamente.

4. Resultados

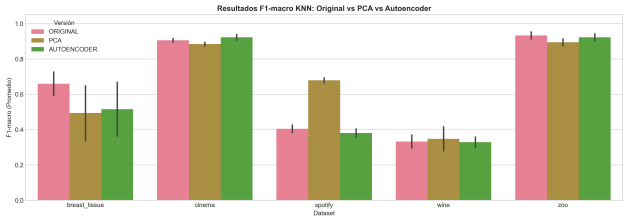

Figure 1. Comparación de resultados F1-macro para k-NN en los 5 datasets.

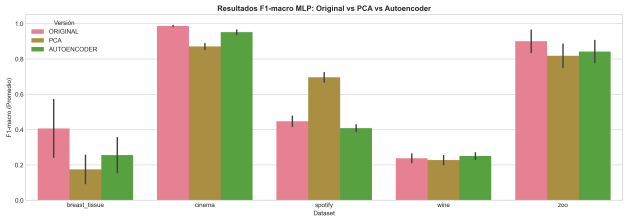

Figure 2. Comparación de resultados F1-macro para MLP en los 5 datasets.

4.1. Análisis por Dataset

Breast Tissue: El mejor resultado se obtiene con los datos originales usando k-NN (F1=0.659). La reducción de dimensionalidad afecta negativamente, especialmente a MLP con PCA, que cae drásticamente a F1=0.175. El Autoencoder mejora ligeramente respecto a PCA (F1=0.516 con k-NN), pero no logra superar al original, indicando que la compresión elimina información relevante en un dataset con pocas instancias.

Cinema: MLP con datos originales alcanza un rendimiento casi perfecto (F1=0.987). La reducción con PCA disminuye el rendimiento (F1=0.871 con MLP), mientras que el Autoencoder recupera parte de la información (F1=0.951), aunque sin igualar al original. Esto sugiere que las características originales son necesarias para una separación óptima.

Spotify: PCA mejora notablemente el rendimiento, subiendo de F1=0.405 (k-NN original) a F1=0.679. MLP con PCA obtiene el mejor resultado global (F1=0.696). Esto confirma la presencia de ruido y redundancia en los datos originales que PCA logra filtrar eficazmente. El Autoencoder, sin embargo, no logra capturar esta ventaja (F1=0.381 con k-NN).

Wine Quality: El rendimiento general es bajo. PCA mejora levemente a k-NN (F1=0.348 vs 0.333 original), su-

giriendo cierta redundancia. MLP tiene dificultades en todas las configuraciones (F1 $\approx$ 0.23-0.25), lo que indica una complejidad en los datos que los modelos actuales no logran modelar adecuadamente.

Zoo: Los datos originales dominan con k-NN (F1=0.932). PCA reduce ligeramente el rendimiento (F1=0.895), mientras que el Autoencoder se mantiene muy competitivo (F1=0.922), demostrando capacidad para comprimir características binarias sin gran pérdida de información, a pesar del tamaño reducido del dataset.

4.2. Análisis Estadístico: Test de Wilcoxon

Para validar si las diferencias observadas son estadísticamente significativas, se aplica el **test de Wilcoxon signed-rank** con $\alpha = 0.05$, comparando rendimientos F1 sobre los 25 folds totales (5 folds $\times$ 5 datasets). La Table 1 muestra los resultados.

Table 1. Resultados del test de Wilcoxon (nivel $\alpha = 0.05$)

COMPARACIÓN	N	P-VALOR	SIG.
K-NN ORIG. VS K-NN PCA	25	0.5249	NO
K-NN ORIG. VS K-NN AE	25	0.0626	NO
K-NN PCA VS K-NN AE	25	0.8532	NO
MLP ORIG. VS MLP PCA	25	0.2411	NO
MLP ORIG. VS MLP AE	25	**0.0096**	SÍ
MLP PCA VS MLP AE	25	0.5424	NO

La única diferencia estadísticamente significativa se encuentra entre **MLP Original y MLP Autoencoder** (p=0.0096), donde el modelo original supera consistentemente a la versión con Autoencoder. Esto confirma que, para redes neuronales, la pérdida de información al comprimir los datos (incluso con un Autoencoder no lineal) perjudica el rendimiento global más que el beneficio que podría aportar la reducción de ruido o dimensionalidad. Las comparaciones con k-NN no muestran diferencias significativas globales, lo que indica que la efectividad de la reducción depende totalmente del dataset específico (mejorando en Spotify/Wine, empeorando en otros), cancelándose los efectos al promediar.

5. Discusión y Conclusiones

Este trabajo revela patrones críticos sobre cuándo aplicar reducción de dimensionalidad y qué clasificador emplear según las características del dataset:

1. La reducción NO siempre mejora: Contraintuitivamente, en 3 de 5 datasets (breast_tissue, cinema, zoo) los datos originales superan las versiones reducidas. Esto desmitifica la noción de que "menos dimensiones = mejor rendimiento". La reducción solo aporta valor cuando: (a) existe redundancia significativa (caso Spotify), (b) el ruido en features originales corrompe señal, o (c) el dataset es tan masivo que el coste computacional de dimensiones altas es prohibitivo. En datasets pequeños y cuidadosamente diseñados (zoo, breast_tissue), cada feature contiene información única que PCA o Autoencoders descartan al comprimir.

2. PCA destaca en datos ruidosos con alta redundancia: El caso Spotify es paradigmático: 17 características acústicas (danceability, energy, valence, etc.) presentan correlaciones evidentes (e.g., energy-loudness, acousticness-valence), y PCA al proyectar sobre direcciones de máxima varianza elimina ruido efectivamente. Mejora del 67% en F1 (0.405$\rightarrow$0.679) demuestra que PCA es una técnica simple pero poderosa cuando las asunciones lineales son válidas. Sin embargo, falla en espacios donde la estructura es intrínsecamente no lineal o donde pocas dimensiones contienen toda la información (breast_tissue).

3. Autoencoders requieren datos abundantes: La arquitectura neuronal del Autoencoder necesita miles de muestras para aprender representaciones útiles. Con 101 instancias (zoo), el Autoencoder entrena pero no generaliza óptimamente, quedando ligeramente por debajo del original. Con 50,000 instancias (Spotify), sin embargo, el Autoencoder falla porque la reducción a la mitad ($17 \rightarrow 8$ dimensiones) es demasiado agresiva para un problema de 10 clases. Esto sugiere que el *bottleneck* debe ajustarse dinámicamente: mantener más dimensiones en problemas complejos ($k \approx 2d/3$ en lugar de $d/2$).

4. k-NN vs MLP: complementariedad: El test de Wilcoxon revela que MLP sufre significativamente más que k-NN al usar Autoencoders (p=0.0096). Esto es lógico: MLP ya tiene capacidad de aprender representaciones internas en sus capas ocultas; forzar una compresión externa previa con otro MLP (el Autoencoder) puede interferir con su aprendizaje. k-NN, al ser un método basado en distancias, es más agnóstico al origen de las características, pero se beneficia enormemente de la limpieza de ruido que hace PCA en datasets como Spotify.

Conclusión final: Este estudio demuestra que no existe una solución universal en reducción de dimensionalidad. La efectividad de PCA, Autoencoders, k-NN y MLP depende críticamente de la estructura intrínseca de los datos: redundancia, ruido, tamaño muestral y complejidad de clases. El análisis riguroso mediante validación cruzada y tests estadísticos permite tomar decisiones informadas, evitando aplicar técnicas sin justificación empírica.

[1]Jordi Blasco Lozano – DNI: 74527208D – jordiblloz@gmail.com