Machine Learning I

Taller N°3: PCA en Flujos de Trabajo de Machine Learning

Alejandro Gil Gallego (274361), Juan David Sánchez González (272906)
Universidad La Sabana, Chia, Cundinamarca
alejandrogiga@unisabana.edu.co, juansangon@unisabana.edu.co
Maestría en Análitica Aplicada
Profesor: Hugo Franco, Ph.D.
10 de octubre de 2025

Resumen

Este documento trata sobre la utilización de PCA para la reducción de dimensionalidad, comparando su rendimiento comparado con otros modelos como lo son kNN, en este caso se aplicó sobre un dataset sobre vinos. Durante la realización del taller se realizaron comparaciones entre distintos modelos y distintas formas de tomar los datos y como el modelo de PCA arroja diferentes resultados.

Index Terms

PCA, workflow / flujo de trabajo, performance / rendimiento, clasificación

I. Introducción

El taller fue propuesto como parte del curso de machine learning I, este tenía como objetivo la utilización de PCA (Análisis de componentes principales), modelo que reduce la dimensionalidad de un dataset reteniendo la mayor parte de la información, transformando variables correlacionadas por componentes [1], dentro de un flujo de trabajo, dicho flujo de trabajo manejaba el dataset de *winequality-red.csv*. Una lista de tareas a realizar en este taller se nos fue suministrada.

Las instrucciones para la realización del taller fueron las siguientes:

1. Añadir y organizar el ejemplo de acuerdo al flujo de trabajo de ciencia de datos.
2. Aplicar el flujo de trabajo al dataset *wine*.
3. Completar los flujos de trabajo de aprendizaje supervisado de acuerdo a los requerimientos y preparación que requiere cada modelo teniendo en cuenta las recomendaciones dadas durante el curso.
4. Comparar el rendimiento de la clasificación usando las características originales y usando únicamente dos características transformadas por PCA.
5. Modificar el ejemplo para realizar una clasificación binaria y comparar los resultados con el multiclase.

II. Metodología

Para el desarrollo de este taller se utilizó el dataset Wine Quality - Red Wine del repositorio UCI Machine Learning. Este dataset contiene diferentes mediciones de propiedades fisicoquímicas de vinos tintos de una región de Portugal. El conjunto de datos está compuesto por 1,599 muestras de vino, cada una definida por 11 características predictorias que representan las propiedades fisicoquímicas, y una variable objetivo denominada quality que corresponde a la puntuación de calidad del vino en una escala de 0 a 10.

La metodología se dividió en cuatro etapas principales que permitieron desarrollar el trabajo de una manera correcta y bien estructurada, siguiendo el flujo de trabajo de ciencia de datos aplicado para problemas de aprendizaje supervisado visto en clase.

- **Carga y exploración de datos:** En este primer paso, se importó el dataset Wine Quality y se analizó la distribución de las variables y la variable objetivo quality.
- **Transformación de la variable objetivo:** Se crearon dos tipos de clasificación según las instrucciones del taller. En primer lugar, se realizó una clasificación multiclase con tres categorías que corresponden a poor, fair y good. Y por otro lado, una clasificación binaria con dos categorías que son poor y good.
- **Entrenamiento de modelos:** Para cada tipo de clasificación se probaron dos versiones del algoritmo Knn. La primera versión usó las 11 características originales y la otra aplicó una reducción de dimensiones haciendo uso del PCA, en la cual permitió quedarse con solo 2 componentes.
- **Evaluación y comparación:** Se evaluó el desempeño de los modelos con ayuda de las diferentes métricas de evaluación.

II-A. Carga y exploración de datos

El primer paso consistió en cargar el dataset desde el repositorio en línea de Machine Learning. Luego, se realizó el análisis exploratorio para entender mejor el conjunto de datos y sus variables. Adicionalmente, se visualizó la distribución de la variable objetivo quality mediante un gráfico de barras, para identificar el desbalanceo de clases presentes. Esta exploración de los datos permitió observar que las puntuaciones de calidad son mayores en los valores de 5 y 6.

```
Algoritmo 1. Carga y exploración de datos

Inicio
   url <- "https://.../winequality-red.csv"
   df <- leer_csv(url, separador=';')
   Mostrar_info(df)
   Graficar_conteo(df, x='quality')
Fin
```

La implementación detallada del algoritmo 1 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

II-B. Trasformación de la variable objetivo

Como ya se mencionó anteriormente, se procedió a transformar la variable quality en dos tipos de de clasificación diferente. El primero consiste en una clasificación multiclase con tres categorías donde poor representa los vinos con puntuación menor o igual a 5, fair son aquellos con puntuación igual a 6 y good los que tienen una puntuación mayor a 6. Esta transformación se hizo con el algoritmo 2 que se observa a continuación, que clasifico en categorías y se analizó su distribución con un conteo y un gráfico de barras para observar el número de muestras en cada clase.

```
Algoritmo 2. Tranformación multiclase

Inicio
   Si calidad ≤5 →clase = 'poor'
   Sino si calidad ≤6 →clase = 'fair'
   Sino →clase = 'good'
   Crear nueva columna 'quality_class' aplicando la función anterior
   Mostrar conteo de clases
   Graficar distribución de 'quality_class'
Fin
```

La implementación detallada del algoritmo 2 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

Luego se implementó la segunda clasificación, que fue de tipo binaria. En este caso, poor correspondía a los vinos con una puntuación menor o igual a 6, y good a los vinos con una puntuación mayor a 6. Esta transformación se realizó ajustando el código para generar únicamente dos clases y así cumplir correctamente con los requerimientos del taller.

```
Algoritmo 3. Transformación binaria

Inicio
   Si calidad ≤6 →clase = 'poor'
   Sino →clase = 'good'
   Crear columna 'quality_class' aplicando la función anterior
   Mostrar conteo de clases binarias
   Graficar distribución de 'quality_class'
Fin
```

La implementación detallada del algoritmo 2 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

II-C. Entrenamiento de modelos

II-C1. Clasificación multiclase: En esta fase, se prepararon los datos para el entrenamiento de los modelos de clasificación. Se seleccionaron las 11 variables predictorias y se eliminaron las columnas quality y quality_class, definiendo así la variable objetivo con las categorías ya transformadas. Después, el conjunto de datos se dividió en conjuntos de entrenamiento y prueba con una proporción de 80-20, usando muestreo estratificado.

```
Algoritmo 4. Preparación datos para modelado

Inicio
   Eliminar columnas 'quality' y 'quality_class' de df y guardar en X
   Asignar 'quality_class' a y
   Dividir X, Y en conjuntos de entrenamiento y prueba (80/20) con estratificación
Fin
```

La implementación detallada del algoritmo 4 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

El primer modelo implementado fue el algoritmo Knn, en donde se usó las 11 características originales sin ninguna transformación. Este modelo base sirvió para comparar los resultados después de aplicar PCA, siendo entrenado con los datos de entrenamiento y registrando el tiempo de ejecución. Luego se hicieron predicciones sobre el conjunto de prueba y se evaluó el desempeño con ayuda de métricas de desempeño, lo cual permitió identificar evaluar el modelo y las clases.

```
Algoritmo 5. Modelado Knn clasificación multiclase

Inicio
   Crear modelo k-NN con 3 vecinos
   Entrenar modelo con X_train y y_train
   Predecir valores de X_test y guardar en baseline_pred
   Calcular tiempo de entrenamiento

   Mostrar medidas de desempeño del modelo
Fin
```

La implementación detallada del algoritmo 5 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

En el segundo modelo, se aplicó una reducción de dimensiones mediante PCA dentro de un pipeline. El cuál incluyó tres pasos principales, en primer lugar se estandarizaron las variables, luego se redujeron las dimensiones a 2 componentes principales con PCA y se aplicó el clasificador Knn. Al igual que el modelo base, se registró el tiempo de entrenamiento y se evaluaron las mismas métricas para comparar su desempeño y eficiencia.

```
Algoritmo 6. Modelado con PCA clasificación multiclas

Inicio
   Crear pipeline con tres pasos: estandarización, PCA con 2 componentes y k-NN con 3 vecinos
   Entrenar pipeline con X_train y y_train
   Predecir valores de X_test y guardar en pipeline_pred
   Calcular tiempo de entrenamiento

    Mostrar medidas de desempeño del modelo
Fin
```

La implementación detallada del algoritmo 6 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

Finalmente, se aplicó el método PCA a los datos ya estandarizados. Se creó un gráfico para ver cómo aumenta la información que explica el modelo al agregar más componentes. Y luego, los datos se redujeron a dos componentes principales que se mostraron en un gráfico de dos dimensiones.

```
Algoritmo 7. Análisis de varianza explicada con PCA

Inicio
   Estandarizar X y guardar en X_scaled
   Aplicar PCA sobre X_scaled
   Graficar varianza acumulada

   Aplicar PCA con 2 componentes y transformar X_scaled en 2d
   Graficar en espacio bidimensional

   Mostrar varianza explicada
```

```
Fin
```

La implementación detallada del algoritmo 7 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

II-C2. Clasificación binaria: Como se mencionó, también se repitió el mismo proceso utilizando la clasificación binaria. Se prepararon nuevamente las variables de entrada y la variable objetivo, se dividieron los datos en conjuntos de entrenamiento y prueba con una proporción 80-20, manteniendo la misma distribución de clases mediante muestreo estratificado.

```
Algoritmo 8. Preparación datos para modelado

Inicio
   Eliminar columnas 'quality' y 'quality_class' de df1 y guardar en X
   Asignar 'quality_class' a y
   Dividir X, Y en conjuntos de entrenamiento y prueba (80/20) con estratificación
Fin
```

La implementación detallada del algoritmo 8 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

El modelo base implementado para la clasificación binaria utilizó el mismo Knn con k=3 vecinos y las 11 características originales. Esto permitió comparar su desempeño con el del modelo multiclase, manteniendo la misma estructura de implementación.

```
Algoritmo 9. Modelado Knn clasificación binaria

Inicio
   Crear modelo k-NN con 3 vecinos
   Entrenar modelo con X_train y y_train
   Predecir valores de X_test y guardar en baseline_pred
   Calcular tiempo de entrenamiento

   Mostrar medidas de desempeño del modelo
Fin
```

La implementación detallada del algoritmo 9 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

De manera similar al caso multiclase, se desarrolló un pipeline que integró estandarización, el PCA con dos componentes y Knn para el problema de clasificación binaria. Esto permitió evaluar si al reducir las dimensiones del número de clases se tiene un impacto en el desempeño del modelo.

```
Algoritmo 10. Modelado con PCA clasificación binaria

Inicio
   Crear pipeline con estandarización, PCA con 2 componentes y k-NN con 3 vecinos
   Entrenar pipeline con X_train y y_train
   Predecir valores de X_test y guardar en pipeline_pred
   Calcular tiempo de entrenamiento

   Mostrar medidas de desempeño del modelo
Fin
```

La implementación detallada del algoritmo 10 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [2].

Finalmente, también se realizó el análisis de varianza, generando el gráfico de varianza acumulada y la visualización de muestras en el espacio de dos dimensiones. Este análisis permitió ver si la variación de los datos se comporta de forma parecida en ambos tipos de clasificación y si las clases se separan mejor al usar solo dos categorías.

III. Resultados

Como continuación del proceso metodológico de trabajo se codificó de acuerdo a los requerimientos iniciales. De esta manera se logró la creación de un notebook *PCA-challenge3.ipynb* así como dos documentos *.py* que cumplen con la misma funcionalidad que el notebook.

El comportamiento del dataset, como lo muestra la figura N°1, tiene una serie de características mucho más predominante que las demás, con este gráfico se puede concluir que para el dataset es necesario utilizar PCA para obtener mejores resultados.

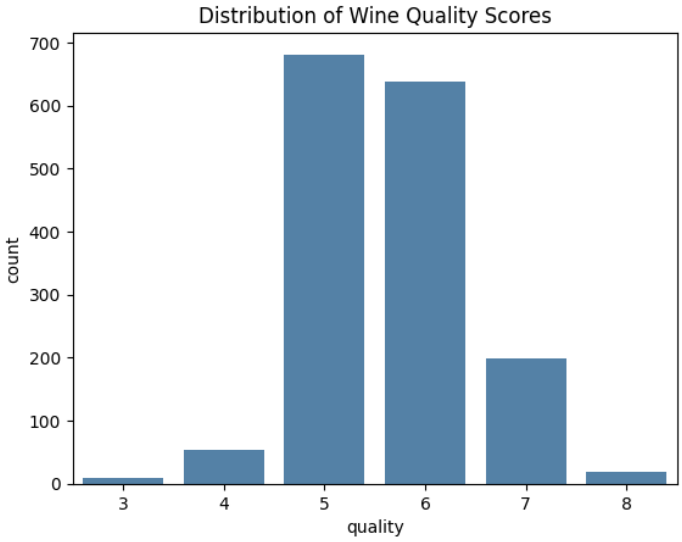

Figura 1. Distribución de las calificaciones de calidad de los vinos

Al realizar las clasificaciones, tanto para el modelo multiclase (Figura N°2) y el modelo binario (Figura N°3) se consiguieron las siguientes distribuciones de las clases nuevas creadas.

Figura 2. Distribución multiclase

Figura 3. Distribución binaria

Posterior a esto se realizó el entrenamiento de los modelos, como base se realizó un entrenamiento con el modelo de kNN el cual para el caso multiclase se consiguió una precisión del 0.5594 y la Figura N°4 muestra la matriz de confusión para este caso. Por otro lado para el caso binario se consiguió una precisión del 0.7188 y la Figura N°5 muestra la matriz de confusión para este caso.

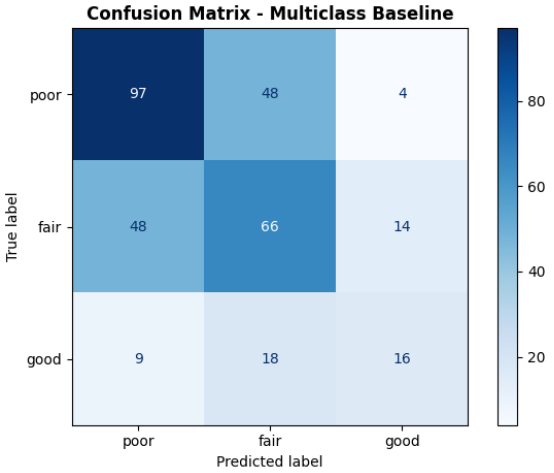

Figura 4. Matriz de confusión kNN multiclase

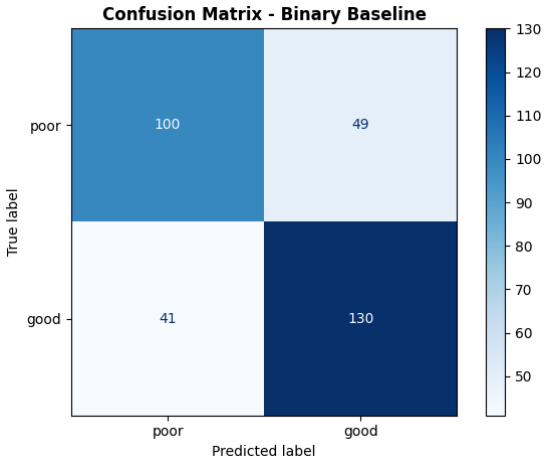

Figura 5. Matriz de confusión kNN binaria

Luego se hizo el entrenamiento para los dos casos con un pipeline con PCA incluido, para el caso multiclase se obtuvo una precisión de 0.5719, mejorando en comparación con solo kNN, y la matriz de confusión mostrada en la Figura N°6. Mientras tanto para el caso binario se obtuvo una precisión del 0.6250, empeorando comparado con solo kNN, y la matriz de confusión mostrada en la Figura N°7.

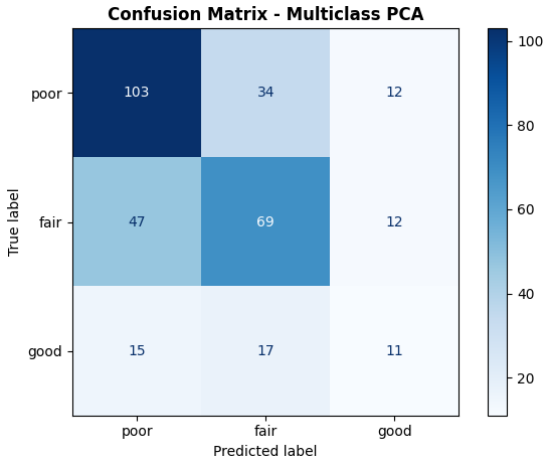

Figura 6. Matriz de confusión PCA multiclase

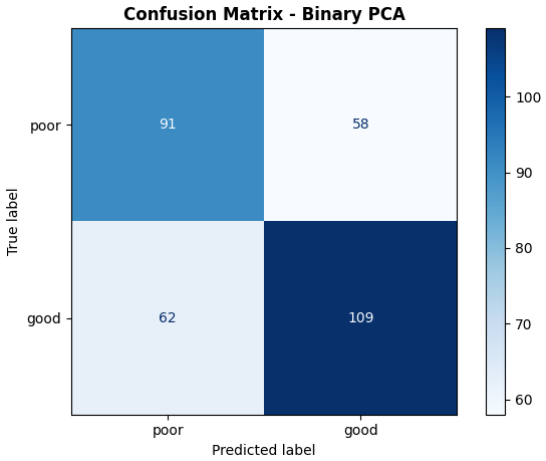

Figura 7. Matriz de confusión PCA binaria

Despues se graficó la varianza acumulada según el número de componentes, para los dos casos el gráfico resultante en la Figura N°8 fue el mismo.

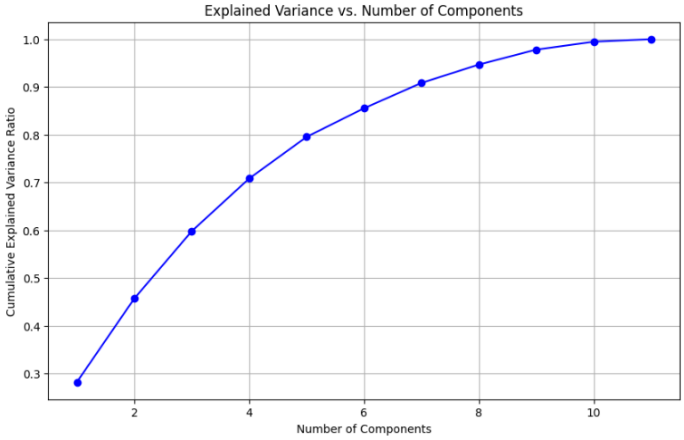

Figura 8. Varianza vs. Numero de componentes

A partir de esta Figura N°8 se puede evidenciar que para tener una buena explicación de la varianza, alrededor de un 80-90 % se puede trabajar con solo de 5 a 7 componentes, disminuyendo la dimensionalidad y manteniendo la mayor parte de la información.

Por último se realizaron scatter plots para mostrar la repartición de los componentes principales en el data set. la Figura N°9 muestra el resultado del caso multiclase y la Figura N°10 muestra el resultado del caso binario.

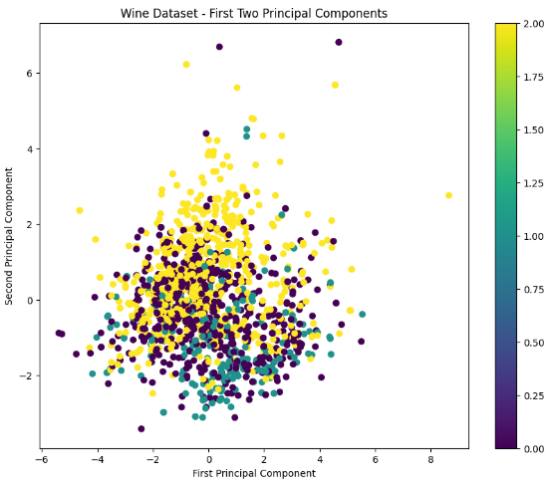

Figura 9. First two Principal Components Multiclass

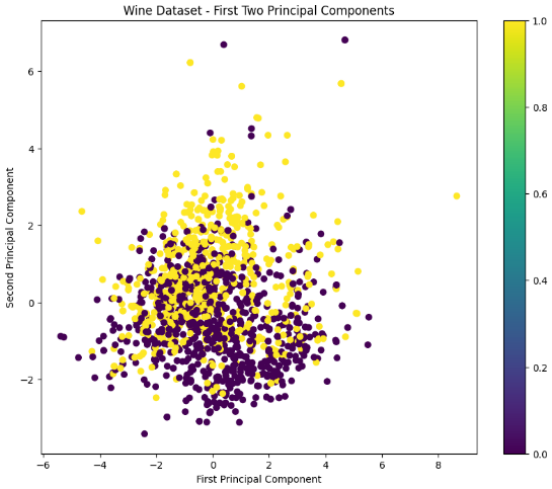

Figura 10. First two Principal Components Binary

Tambien se obtuvo la varianza total que representaban los dos primeros componentes creados por PCA en el dataset la Figura N°9 como maneja tres clases los datos son coloreados con tres tonalidades, mientras que la Figura N°10 lo colera con dos, para ambos casos el ratio de la varianza que explican estos dos componentes dan en total un 0.4568. De estas Figuras 9 y 10 se puede concluir que las clases estan unas sobres las otras por lo que es practicamente imposible encontrar patrones.

Tomando en cuenta las instrucciones suministradas y los procesos realizados para la construcción de un código funcional, se llegó a la siguiente conclusión.

- Al realizar la comparación del desempeño tanto del caso binario y el caso multiclase se llego a diferentes resultados en la utilización de PCA. Para el caso multiclase se logró evidenciar una mejora en la utilización de PCA comparado con kNN, mientras que en el caso binario se evidencia un empeoramiento de PCA comparado con kNN. Esto se debe a que PCA es mucho más util para manejar conjuntos multiclase, donde maneje más variedad de datos y de esta manera sea eficaz creando una representación de los componentes principales. Mientras que no es recomendado utilizar en casos binarios, ya que son más simples y el modelo PCA al ser mucho más complejo termina empeorando el rendimiento.

IV. Discusión

La utilidad de este taller recae en el entendimiento más profundamente, a partir de la prueba, del modelo PCA. De igual manera que el taller anterior el objetivo final es darnos una idea de un caso encapsulado y probado para entender el funcionamiento del modelo y entenderlo para a futuro saber en que momento es correcto utilizar el modelo PCA para obtener buenos resultados.

Referencias

[1] IBM, "What is Principal Component Analysis (PCA)?,ÏBM Think, https://www.ibm.com/think/topics/principal-component-analysis(accessedOct.9,2025)..

[2] Alejandro Gil, Juan David Sánchez, *Taller3_ML1*, GitHub, 2025. [En línea]. Disponible en: https://github.com/AGGE82/Taller-3-ML1