Universidad Tecnológica de Querétaro

Reporte de Modelos de Aprendizaje Automático

Análisis Supervisado y No Supervisado

Alumno: Tristan Jacob Valencia Ramírez

Docente: Filiberto Ruiz Hernández

Fecha: 5 de diciembre de 2025

Departamento de Tecnologías de la Información

Ingeniería en Desarrollo y Gestión de Software

Índice general

Introducción

El presente documento reúne cuatro análisis completos aplicando técnicas de aprendizaje automático supervisado y no supervisado. Cada capítulo corresponde a un modelo desarrollado con un conjunto de datos específico:

- **Regresión:** Predicción del número de veces que se comparte un artículo en línea usando el dataset `articulos_ml.csv`, mediante un modelo de *Ridge Regression*.
- **Clasificación:** Identificación de presencia de diabetes con el dataset `diabetes.csv` usando un *Árbol de Decisión*.
- **Agrupamiento (Clustering):** Segmentación de clientes con el dataset `cliente_tienda.csv` empleando el algoritmo *K-Means*.
- **Reducción de Dimensionalidad:** Aplicación de *PCA* al dataset `iris.csv` para representar los datos en menos dimensiones conservando la mayor parte de la varianza.

Cada reporte incluye:

- Justificación del algoritmo utilizado.
- Procedimiento detallado del diseño del modelo.
- Evaluación y optimización (en los modelos que lo requieren).
- Gráficas generadas en Python y su interpretación.

1. Modelo de Regresión con el Dataset "articulos_ml.csv"

1.1. Justificación del Algoritmo

Para la predicción de la variable `# Shares` (número de veces que se comparte un artículo), se empleó el algoritmo **Ridge Regression**, que es una variante de la regresión lineal con regularización L2.

Las razones principales de esta elección son:

- La relación entre las características del artículo y el número de compartidos puede aproximarse como una combinación lineal, pero con posible colinealidad entre variables (*Word count*, enlaces, comentarios, etc.), lo que hace útil la regularización.
- Ridge penaliza la magnitud de los coeficientes, ayudando a reducir el sobreajuste y a estabilizar el modelo cuando hay muchas variables numéricas.
- Es un modelo relativamente simple e interpretable: los coeficientes indican la dirección e intensidad de la relación entre cada característica y la variable objetivo.

1.2. Diseño del Modelo

Los pasos implementados en el notebook fueron:

1. Carga del archivo `articulos_ml.csv` y exploración inicial del dataset.
2. Selección únicamente de columnas numéricas y eliminación de filas con valores nulos en la variable objetivo `# Shares`.
3. Relleno de valores faltantes en las variables predictoras con la mediana de cada columna.
4. Definición de la variable objetivo (`# Shares`) y del conjunto de *features*.

5. División del dataset en conjuntos de entrenamiento y prueba (80 % y 20 % respectivamente).

6. Escalado de las variables numéricas usando `StandardScaler`.

7. Entrenamiento de un modelo base `Ridge` sin optimización.

8. Definición de una rejilla de valores de `alpha` y búsqueda de hiperparámetros con `GridSearchCV`.

9. Entrenamiento del modelo final con el mejor valor de `alpha` encontrado.

1.3. Evaluación y Optimización

La evaluación del modelo se realizó con las métricas:

- **MAE** (Mean Absolute Error): error promedio absoluto.
- **RMSE** (Root Mean Squared Error): raíz del error cuadrático medio.
- R^2: proporción de varianza explicada por el modelo.

Primero se entrenó el modelo base de Ridge; posteriormente, mediante **GridSearchCV** se evaluaron distintos valores de `alpha` para encontrar un equilibrio entre ajuste y regularización. El modelo optimizado mostró mejores métricas en el conjunto de prueba, indicando una reducción del error de predicción y un mejor ajuste respecto al modelo base.

1.4. Gráficas e Interpretación

Dispersión Word Count vs # Shares

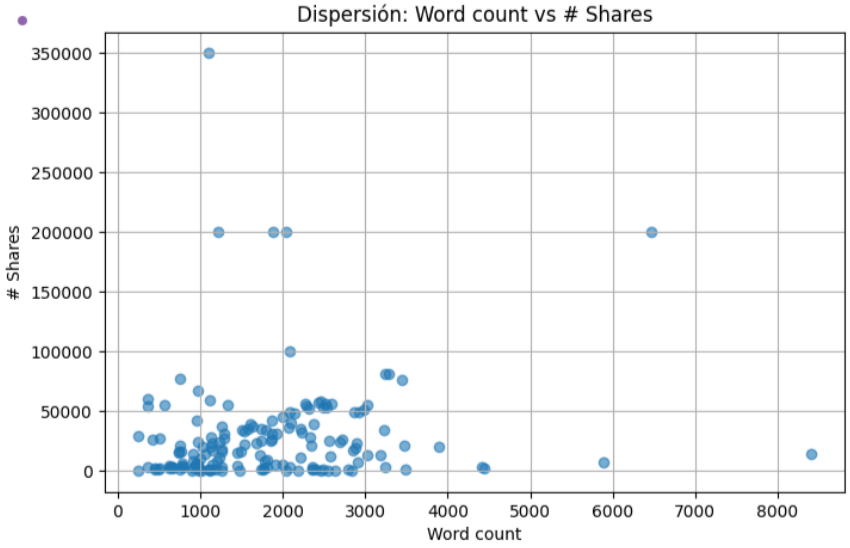

Figura 1.1: Dispersión de *Word count* vs *# Shares*.

Permite observar si existe alguna tendencia entre la longitud del artículo y la cantidad de compartidos.

Valores Reales vs Predichos

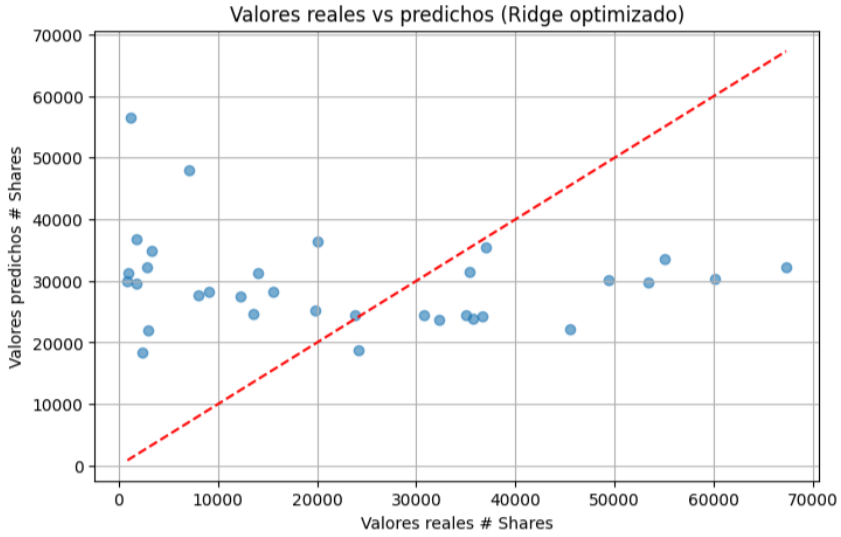

Figura 1.2: Comparación entre valores reales y valores predichos de *# Shares*.

Los puntos cercanos a la diagonal indican buenas predicciones del modelo; una gran dispersión respecto a esa línea sugiere errores más altos.

Coeficientes del Modelo Ridge

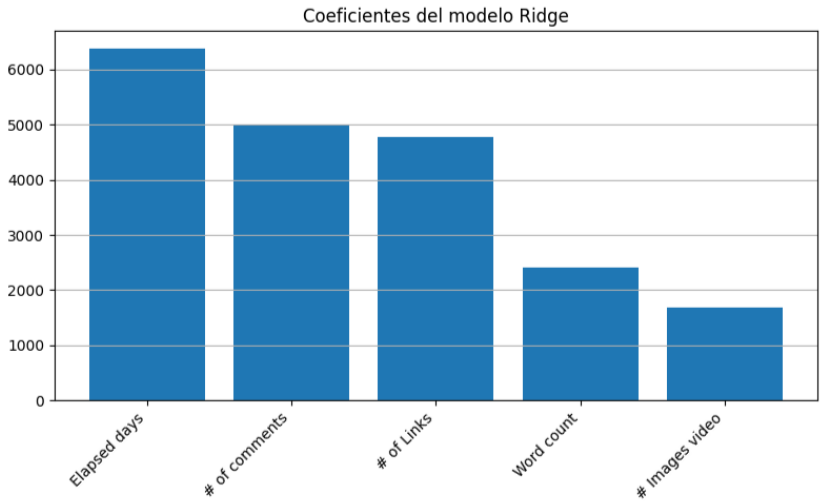

Figura 1.3: Coeficientes del modelo Ridge para cada característica.

Los coeficientes muestran qué variables influyen positivamente o negativamente en la predicción de `# Shares` y con qué intensidad relativa.

1.5. Modelo Guardado

El modelo entrenado (incluyendo el escalador y las columnas usadas) se guardó como:

`modelo_ridge_articulos.pkl`

2. Modelo de Clasificación con el Dataset "diabetes.csv"

2.1. Justificación del Algoritmo

Para clasificar a los pacientes entre **con diabetes** y **sin diabetes**, se eligió un **Árbol de Decisión (DecisionTreeClassifier)**. Este modelo ofrece:

- Alta interpretabilidad: se pueden extraer reglas claras a partir del árbol.
- Capacidad de manejar relaciones no lineales entre características clínicas.
- Compatibilidad con datos numéricos sin necesidad de escalado.

2.2. Diseño del Modelo

1. Carga y exploración del dataset `diabetes.csv`.
2. Separación de variables de entrada `X` y la variable objetivo `Outcome`.
3. División del dataset en entrenamiento y prueba, manteniendo la proporción de clases (`stratify`).
4. Entrenamiento de un árbol de decisión base.
5. Evaluación inicial con **accuracy** y **classification report**.
6. Optimización de hiperparámetros con `GridSearchCV`, variando:
   - `max_depth`
   - `min_samples_split`
   - `min_samples_leaf`
   - `criterion`
7. Entrenamiento del mejor modelo y análisis de métricas finales.

2.3. Evaluación y Optimización

Se emplearon métricas de clasificación como:

- **Exactitud (Accuracy)**.
- **Precision**, **Recall** y **F1-score**.
- **Matriz de confusión**.
- **Curva ROC y AUC**.

Estas métricas permiten evaluar no solo la cantidad de aciertos totales, sino también la calidad de las predicciones para cada clase, lo cual es importante en problemas médicos.

2.4. Gráficas e Interpretación

Matriz de Confusión

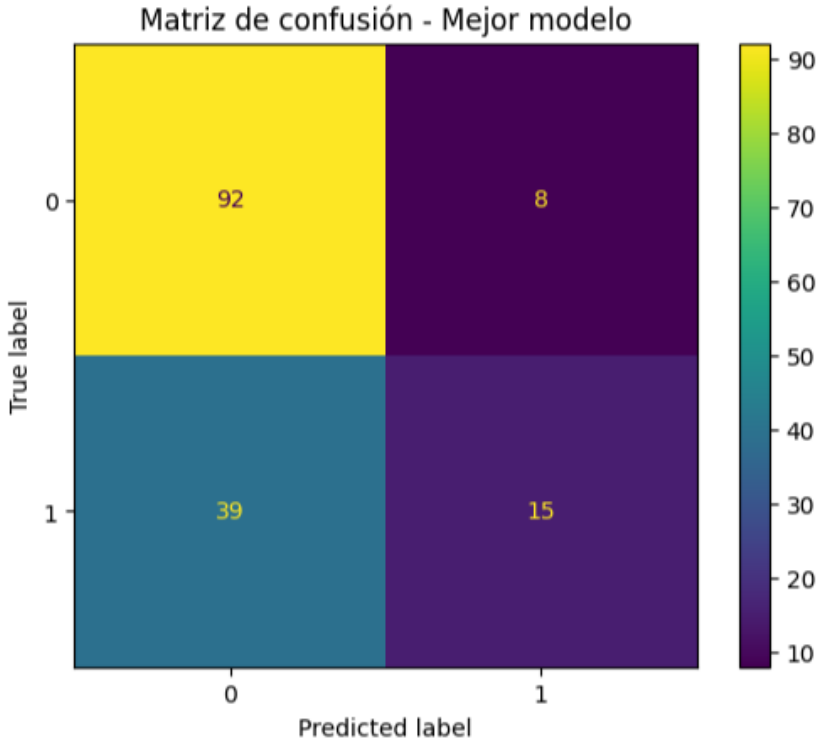

Figura 2.1: Matriz de confusión del modelo optimizado.

Muestra la proporción de pacientes correctamente clasificados y los errores tipo falso positivo y falso negativo.

Curva ROC

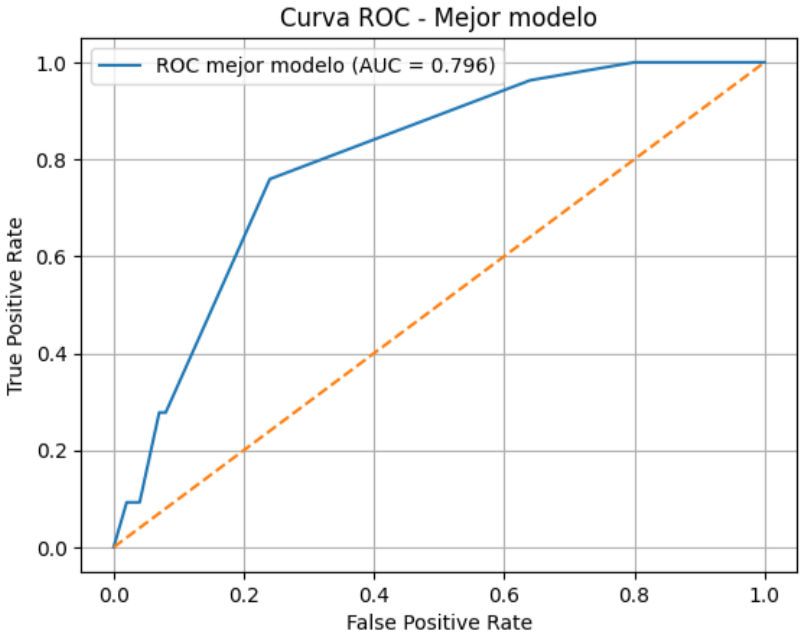

Figura 2.2: Curva ROC del clasificador de diabetes.

La curva ROC y el área bajo la curva permiten medir la capacidad del modelo para distinguir entre pacientes con y sin diabetes a diferentes umbrales de decisión.

Árbol de Decisión

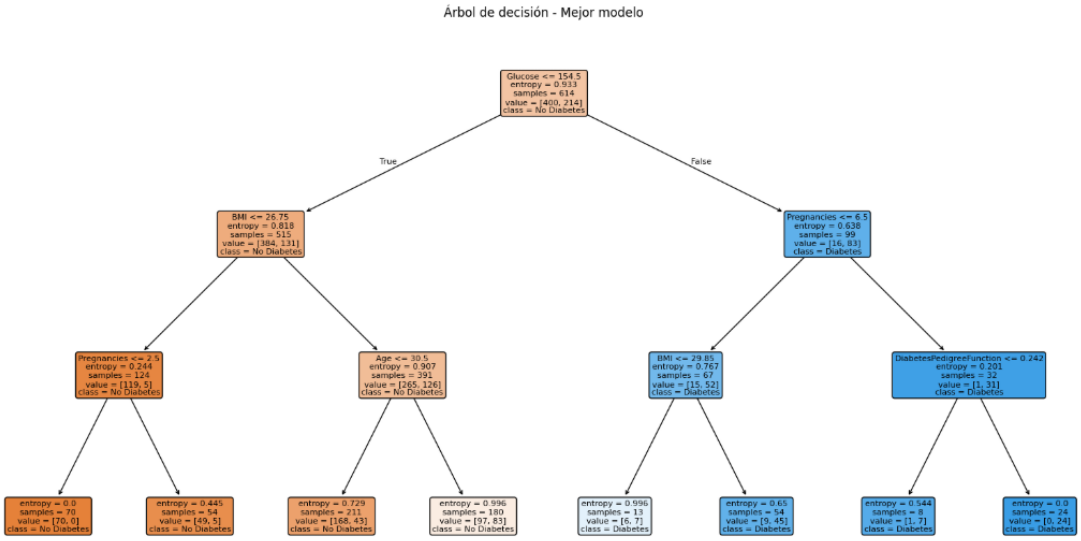

Figura 2.3: Árbol de decisión del modelo final.

La estructura del árbol muestra qué variables se usan en cada división y cómo se llega a la decisión final para cada hoja.

2.5. Modelo Guardado

`modelo_arbol_decision_diabetes.pkl`

3. Modelo de Agrupación con el Dataset "cliente_tienda.csv"

3.1. Justificación del Algoritmo

Se empleó el algoritmo **K-Means** para realizar la segmentación de clientes debido a que:

- Es uno de los métodos de clustering más utilizados y eficientes para datos numéricos.
- Busca particionar los datos en grupos compactos y bien separados, lo cual es adecuado para este dataset.
- Funciona muy bien en combinación con técnicas de reducción de dimensionalidad como PCA.

3.2. Diseño del Modelo

El procedimiento seguido consistió en:

1. Carga y exploración del dataset `cliente_tienda.csv`.
2. Selección de las características numéricas relevantes.
3. Normalización de los datos mediante **StandardScaler** para asegurar igual peso entre variables.
4. Aplicación de PCA para reducción de dimensionalidad con el objetivo de conservar al menos el 95 % de la varianza del dataset. Este proceso resultó en un PCA compuesto por 5 componentes principales.
5. Entrenamiento de K-Means sobre el espacio reducido de PCA.
6. Búsqueda del número óptimo de clusters probando valores de k entre 2 y 10.
7. Evaluación de cada modelo mediante:

- **Inertia** (Método del codo).
- **Silhouette Score**.

8. Selección del valor de k más adecuado considerando ambas métricas.
9. Reducción posterior a 2 componentes con PCA únicamente para fines de visualización gráfica.

3.3. ¿Por qué se usaron 2 componentes para la visualización?

Aunque el análisis PCA completo requirió 5 componentes para conservar más del 95 % de la varianza del dataset, se utilizaron únicamente **2 componentes principales** para la visualización de los clusters. Esto se debe a que:

- Las gráficas sólo pueden representarse en un plano bidimensional, por lo que es necesario reducir los datos a dos ejes.
- Las dos primeras componentes principales concentran la mayor cantidad posible de información dentro de ese espacio 2D.
- El objetivo del PCA en esta etapa no es mejorar el rendimiento del modelo, sino **permitir una interpretación visual clara** de los grupos formados por K-Means.

En resumen, los 5 componentes del PCA se utilizan para entrenar el modelo, mientras que las 2 componentes se emplean únicamente para la representación gráfica de los clusters.

3.4. Optimización del Modelo

La optimización del modelo se enfocó en elegir el número ideal de clusters. Para ello se aplicaron dos métodos:

- **Método del codo (Inertia):** se identifica el punto donde la disminución de la inercia deja de ser significativa.
- **Silhouette Score:** se selecciona el valor de k que maximiza la separación entre grupos.

Ambos indicadores permitieron seleccionar un número de clusters que proporcionó buena cohesión interna y separación entre grupos.

3.5. Gráficas e Interpretación

Método del Codo

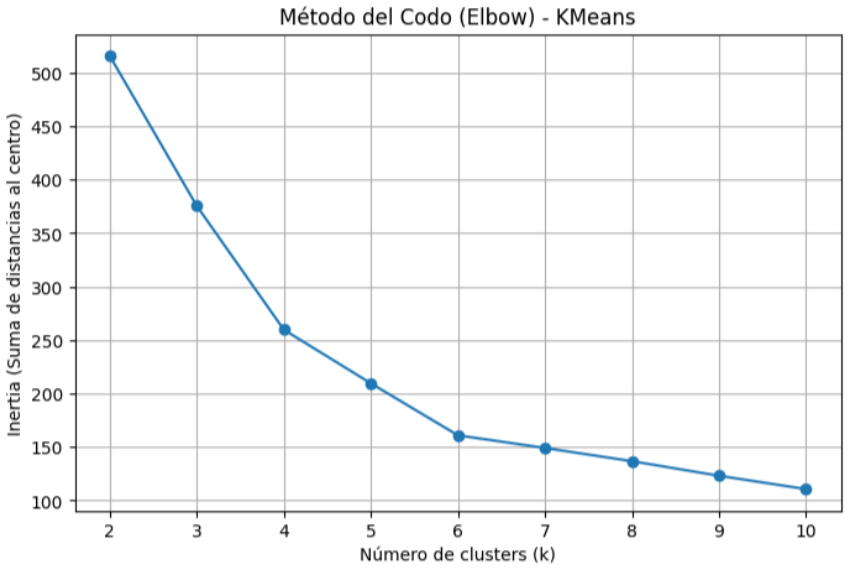

Figura 3.1: Método del codo: Inertia vs número de clusters.

Silhouette Score

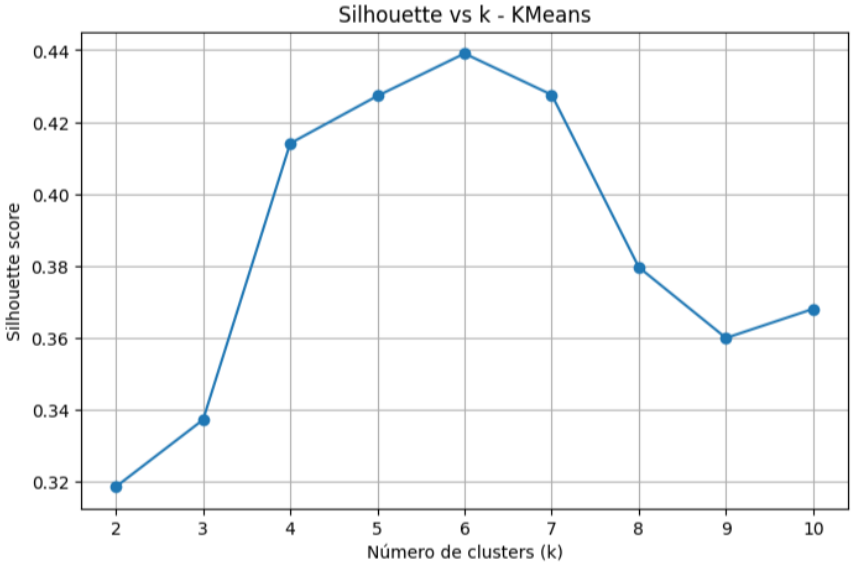

Figura 3.2: Silhouette Score para diferentes valores de k.

Clusters visualizados con PCA 2D

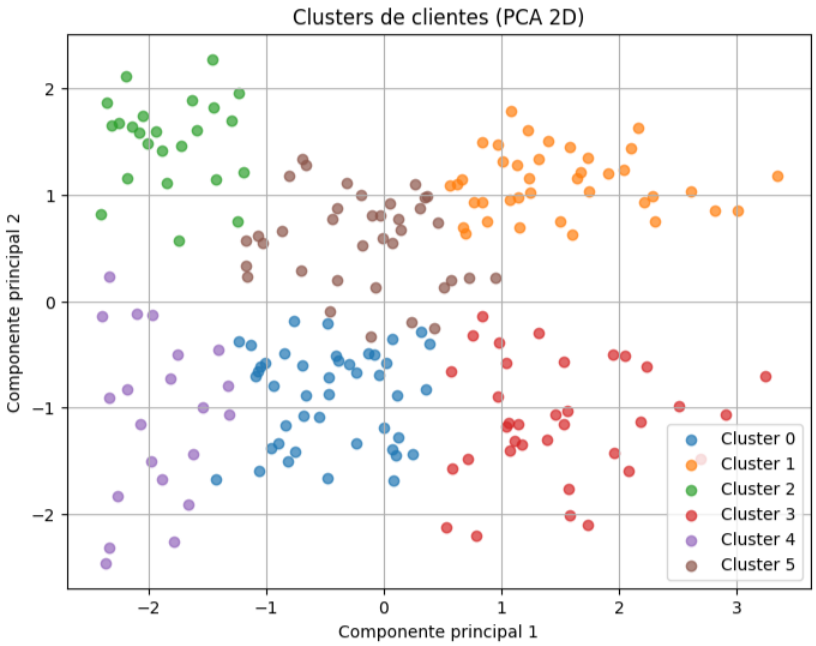

Figura 3.3: Clusters proyectados en 2D mediante PCA.

Esta gráfica facilita interpretar cómo se distribuyen los clientes en el espacio reducido y cómo los separa el modelo.

3.6. Modelo Guardado

El modelo final entrenado se exportó como:

`modelo_kmeans_cliente_tienda.pkl`

4. Reducción de Dimensionalidad con PCA sobre "iris.csv"

4.1. Justificación del Algoritmo

Se aplicó **Análisis de Componentes Principales (PCA)** al dataset `iris.csv`. PCA es adecuado porque:

- Reduce la dimensionalidad del conjunto de datos manteniendo la mayor parte de la varianza.
- Permite visualizar datos multivariados en un plano bidimensional.
- Facilita la identificación de separaciones naturales entre las especies de iris.

4.2. Diseño del Modelo

1. Carga del dataset y revisión de las variables disponibles.
2. Separación de las características numéricas y de la etiqueta de especie.
3. Escalado de las variables con `StandardScaler`.
4. Aplicación de PCA sin fijar el número de componentes para calcular la varianza explicada por cada uno.
5. Cálculo de la varianza acumulada y decisión del número de componentes a utilizar.
6. Selección de **2 componentes principales**, suficientes para explicar la mayor parte de la varianza.
7. Reaplicación de PCA con `n_components = 2`.
8. Generación de gráficas: varianza acumulada, proyección 2D y biplot.

4.3. Reducción de Dimensionalidad y Justificación

El análisis de la varianza explicada acumulada mostró que:

- Las dos primeras componentes concentran más del 95 % de la variabilidad total del dataset.
- El uso de dos componentes permite representar los datos en 2D de forma clara.

Por tanto, se eligió $n\_components = 2$ como valor adecuado para la reducción de dimensionalidad.

4.4. Gráficas e Interpretación

Varianza Explicada

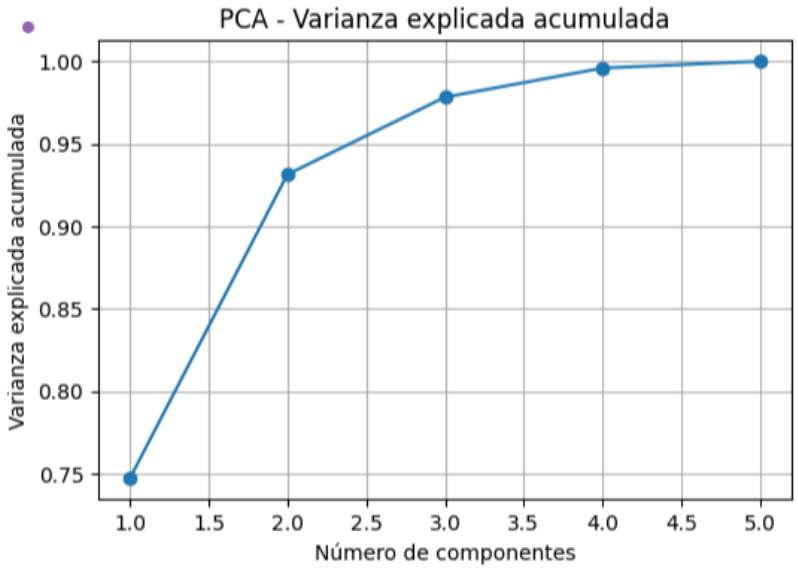

Figura 4.1: Varianza explicada acumulada por número de componentes.

La gráfica confirma que pocas componentes son suficientes para capturar la mayor parte de la información.

Proyección PCA 2D

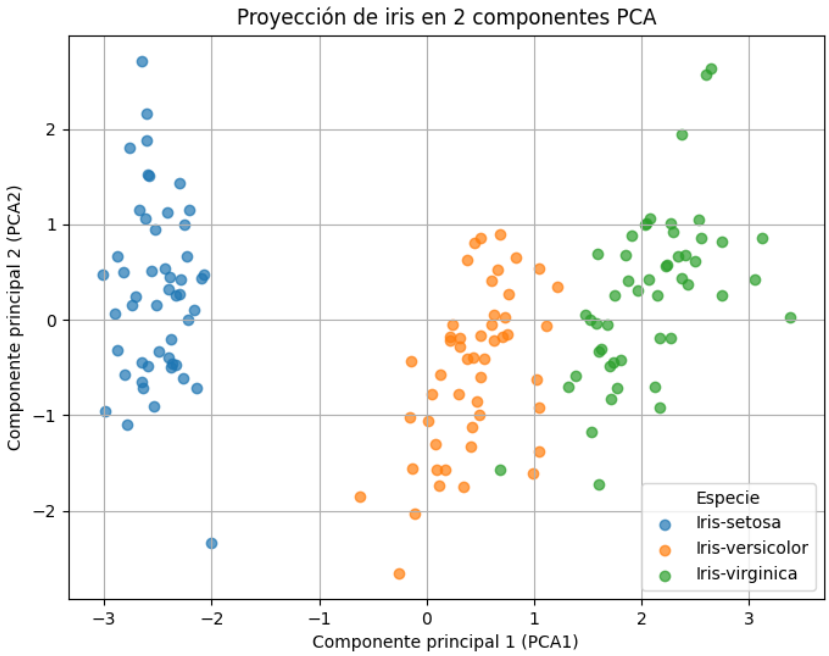

Figura 4.2: Proyección del dataset `iris` en los dos primeros componentes principales.

Se observa cómo las distintas especies de iris tienden a ocupar regiones separadas en este plano reducido.

Biplot

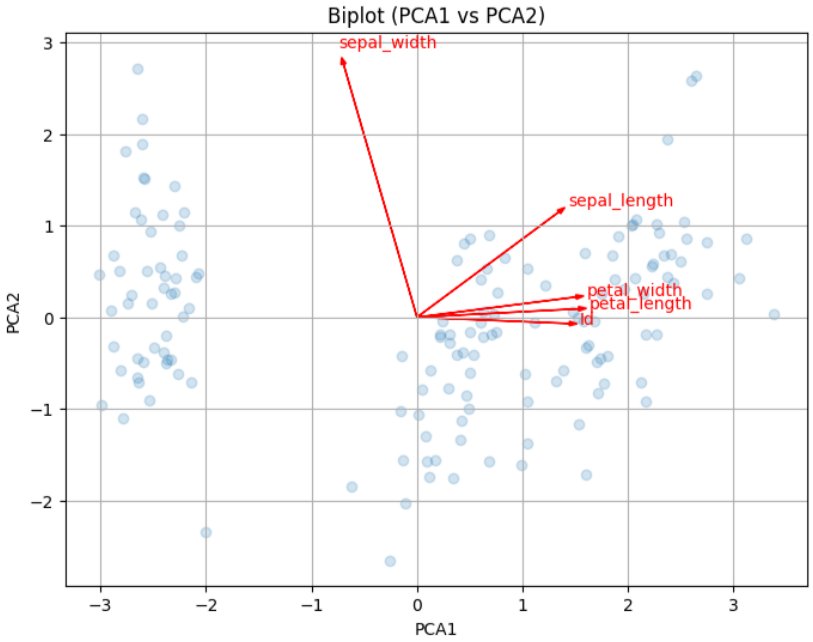

Figura 4.3: Biplot con las cargas de las variables originales en el espacio de PCA.

El biplot muestra qué variables (longitud/ancho de pétalo y sépalo) contribuyen más a cada componente y cómo se relacionan entre sí.

4.5. Modelo Guardado

`modelo_pca_iris.pkl`

Repositorio del Proyecto

Todos los notebooks, modelos entrenados (`.pkl`), datasets utilizados y las imágenes generadas en este proyecto se encuentran disponibles en el siguiente repositorio de GitHub:

Repositorio oficial del proyecto en GitHub

El repositorio contiene:

- Los notebooks en formato `.ipynb` para cada ejercicio.
- Los modelos entrenados (`modelo_ridge_articulos.pkl`, `modelo_arbol_decision_diabetes.pkl`, `modelo_kmeans_cliente_tienda.pkl`, `modelo_pca_iris.pkl`).
- Los scripts y recursos adicionales necesarios para reproducir los resultados.