Universidad de Cundinamarca

Ingeniería de Sistemas y Computación

Clasificación del Dataset Iris usando Regresión Lineal

Análisis completo con visualizaciones

Autores:
Juan Esteban Moya Riaño
Maryuri Alejandra Espinosa

Docente:
Alexander Espinosa

Machine Learning
Septiembre 2025

Objetivo del Proyecto

El objetivo de este proyecto es construir un modelo de clasificación para identificar correctamente las especies de flores en el **dataset Iris**, utilizando **regresión lineal** con la estrategia One-vs-Rest. Se busca demostrar cómo este método puede aplicarse a problemas de clasificación multiclase, analizar sus limitaciones y comparar su rendimiento frente a otras técnicas más avanzadas.

Descripción del Dataset

El dataset Iris es uno de los conjuntos de datos más conocidos en Machine Learning, originalmente publicado por Ronald Fisher en 1936 y disponible en el repositorio de UCI Machine Learning.

- 150 muestras en total.
- 3 clases (50 muestras cada una): *Iris-setosa*, *Iris-versicolor*, *Iris-virginica*.
- 4 características: *sepal length*, *sepal width*, *petal length*, *petal width*.
- No contiene valores faltantes.

Metodología

El procedimiento se desarrolló en varias etapas:

1. **Preprocesamiento:** estandarización de características y codificación de etiquetas.
2. **Entrenamiento:** se aplicó regresión lineal con la estrategia One-vs-Rest, entrenando un modelo independiente por cada clase.
3. **Predicción:** se utilizaron las salidas de cada modelo como puntajes y se aplicó la función softmax para obtener probabilidades finales.
4. **Evaluación:** se emplearon métricas como *accuracy*, matriz de confusión y análisis de sobreajuste.

Análisis Detallado del Código

El programa implementa un clasificador para el **dataset Iris** utilizando **regresión lineal** con la estrategia One-vs-Rest (OvR). A continuación, se describe detalladamente cada componente del código.

Importación de Librerías y Configuración Inicial

El código comienza con la importación de librerías esenciales para el análisis:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import accuracy_score, classification_report
    , confusion_matrix
```

Listing 1: *Librerías utilizadas*

Estas librerías permiten:

- `numpy`, `pandas`: manipulación de datos numéricos y tabulares.
- `matplotlib`, `seaborn`: visualización gráfica.
- `scikit-learn`: preprocesamiento, regresión lineal y métricas.

Definición de la Clase IrisLinearClassifier

La lógica central se encapsula en la clase `IrisLinearClassifier`. Este diseño modular facilita la reutilización y claridad del código.

```python
class IrisLinearClassifier:
    def __init__(self, use_regularization=False, alpha=1.0):
        self.scaler = StandardScaler()
        self.label_encoder = LabelEncoder()
        self.use_regularization = use_regularization
        self.alpha = alpha
        self.class_names = ['Iris-setosa', 'Iris-versicolor', '
            Iris-virginica']
        self.feature_names = ['sepal_length', 'sepal_width', '
            petal_length', 'petal_width']
        self.models = {}
```

Listing 2: *Inicialización del clasificador*

Se destacan dos parámetros importantes:

- `use_regularization`: permite usar regresión Ridge en lugar de lineal estándar.
- `alpha`: controla la intensidad de la regularización.

Carga y Preprocesamiento de Datos

El método `load_data` carga el dataset y separa características (`X`) y etiquetas (`y`). Posteriormente, `preprocess_data` estandariza las características y codifica las etiquetas.

```python
X_scaled = self.scaler.fit_transform(X)
y_encoded = self.label_encoder.fit_transform(y)
```

Listing 3: *Preprocesamiento de datos*

La **estandarización** es crucial porque evita que características con valores más grandes (ej. longitud del sépalo) dominen el modelo.

Entrenamiento con One-vs-Rest

El método `train_models` entrena un modelo independiente por cada clase. Para la clase *setosa*, por ejemplo, el modelo aprende a distinguir entre *setosa* y las otras dos especies.

```
for i, class_name in enumerate(self.class_names):
    y_binary = (y == i).astype(int)
    model = LinearRegression()
    model.fit(X, y_binary)
    self.models[class_name] = model
```

Listing 4: *Entrenamiento One-vs-Rest*

Cada modelo devuelve un puntaje que luego se combina con **softmax** para producir probabilidades normalizadas.

Predicción y Evaluación

El método `predict` aplica los modelos entrenados y usa softmax para obtener probabilidades:

```
exp_scores = np.exp(class_scores - np.max(class_scores))
probs = exp_scores / np.sum(exp_scores)
predicted_class = np.argmax(probs)
```

Listing 5: *Predicción con softmax*

El método `evaluate` calcula métricas estándar:

- **Accuracy**: proporción de aciertos globales.
- **Precision, Recall, F1-score**: métricas detalladas por clase.
- **Matriz de confusión**: errores y aciertos por categoría.

Visualizaciones

El código incluye múltiples funciones gráficas:

- `plot_data_distribution`: pairplot y correlación entre características.
- `plot_confusion_matrix`: matriz de confusión.
- `plot_feature_importance`: coeficientes de los modelos.
- `plot_decision_boundaries_2d`: fronteras de decisión usando sólo los sépalos.

Estas gráficas permiten entender las limitaciones del modelo, especialmente en la separación entre *versicolor* y *virginica*.

Función Principal `main`

La función `main()` integra todo el flujo:

1. Carga y preprocesamiento del dataset.
2. División en entrenamiento (70%) y prueba (30%).
3. Entrenamiento de los tres modelos lineales.
4. Evaluación y visualización de resultados.

```
if __name__ == "__main__":
    classifier, train_results, test_results = main()
```

Listing 6: *Ejecución principal*

Conclusiones del Análisis

El código está bien estructurado y documentado. Sus principales aportes son:

- Uso claro de la estrategia One-vs-Rest.
- Integración de visualizaciones interpretativas.
- Flujo reproducible de carga, entrenamiento, evaluación y análisis.

Sin embargo, la **regresión lineal** tiene limitaciones para problemas de clasificación, ya que no garantiza fronteras óptimas entre clases no linealmente separables. Por ello, se recomienda comparar este enfoque con regresión logística o SVM.

Resultados y Análisis Gráfico

Distribución de Clases y Estadísticas

La Figura 2 Se observa que las tres especies de iris (*setosa*, *versicolor* y *virginica*) están perfectamente balanceadas, cada una con 50 muestras. Este equilibrio es una ventaja para los algoritmos de clasificación, ya que evita sesgos hacia una clase mayoritaria y permite evaluar de forma justa el rendimiento del modelo en todas las categorías.

Estadísticas Descriptivas

El cuadro resume las medidas principales de cada característica:

- **Longitud del sépalo (sepal_length)**: presenta una media de 5.84 cm y una desviación estándar de 0.83 cm. Su rango varía entre 4.30 y 7.90 cm, lo cual indica variabilidad moderada entre especies.
- **Ancho del sépalo (sepal_width)**: con una media de 3.05 cm y menor dispersión (desviación estándar de 0.43 cm). Sus valores se concentran entre 2.00 y 4.40 cm.

- **Longitud del pétalo (petal_length)**: es una de las variables más discriminativas, con una media de 3.76 cm pero alta dispersión (1.76 cm). El rango amplio (1.00–6.90 cm) refleja que especies como *setosa* tienen pétalos mucho más cortos en comparación con las otras dos.
- **Ancho del pétalo (petal_width)**: muestra la mayor variabilidad relativa, con una media de 1.20 cm y desviación estándar de 0.76 cm. El rango (0.10–2.50 cm) también evidencia la clara separación de *setosa* respecto a *versicolor* y *virginica*.

Interpretación

El análisis preliminar sugiere que las características relacionadas con los pétalos (*petal_length* y *petal_width*) tienen mayor poder de discriminación entre especies, especialmente para separar *setosa* del resto. En contraste, las medidas del sépalo presentan más solapamiento, lo que anticipa dificultades para separar *versicolor* y *virginica* únicamente con esas variables.

Este diagnóstico inicial resulta fundamental antes de aplicar modelos predictivos, ya que orienta sobre qué características pueden aportar mayor valor al proceso de clasificación.

Característica	Media	Desv. Est.	Mín	Máx
sepal_length	5.84	0.83	4.30	7.90
sepal_width	3.05	0.43	2.00	4.40
petal_length	3.76	1.76	1.00	6.90
petal_width	1.20	0.76	0.10	2.50

Figure 1: Resumen del dataset Iris: distribución de clases, estadísticas

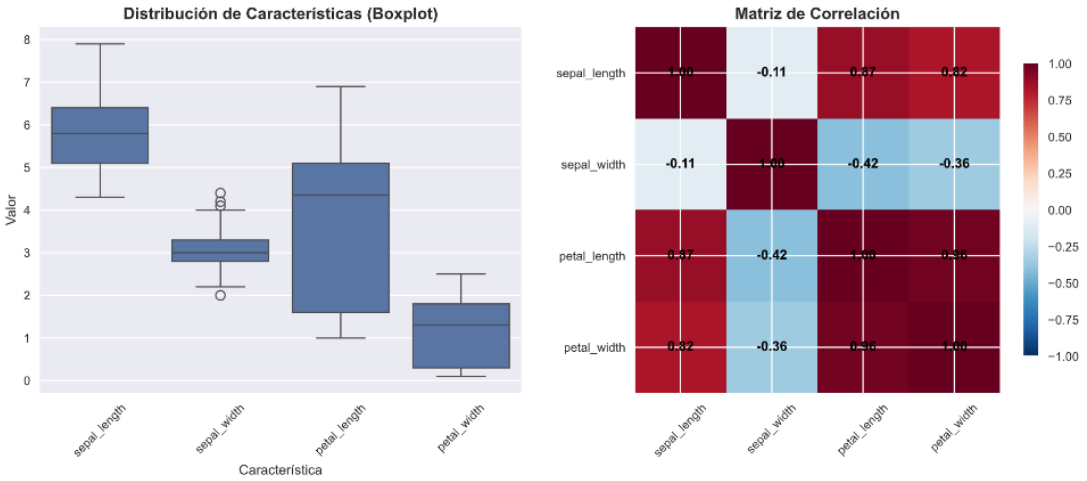

Figure 2: Resumen del dataset Iris: boxplots y matriz de correlación.

Distribución de Características (Boxplot)

Los diagramas fig 2 de caja permiten observar la dispersión y la presencia de posibles valores atípicos:

- **Longitud del sépalo (sepal_length)**: presenta un rango de aproximadamente 4.3 a 7.9 cm, con mediana cercana a 5.8 cm. Se observa una distribución relativamente simétrica sin valores extremos significativos.
- **Ancho del sépalo (sepal_width)**: es la variable con menor rango (2.0–4.4 cm). Se identifican algunos valores atípicos hacia la parte inferior, lo que indica variabilidad en ciertos ejemplares.
- **Longitud del pétalo (petal_length)**: muestra alta variabilidad, con una mediana de 4.3 cm y un rango muy amplio (1.0–6.9 cm). Esto confirma su capacidad discriminativa entre especies.
- **Ancho del pétalo (petal_width)**: aunque presenta mayor concentración, su rango de 0.1 a 2.5 cm también la convierte en una característica relevante para separar clases.

Matriz de Correlación

fig 2 La matriz de correlación revela las relaciones lineales entre características:

- Existe una fuerte correlación positiva entre **longitud y ancho del pétalo** (0.96), lo cual sugiere que ambas variables están estrechamente relacionadas en el proceso de diferenciación de especies.
- La **longitud del sépalo** también se relaciona de forma moderada con la longitud del pétalo (0.87), lo que indica cierta dependencia morfológica.
- En contraste, el **ancho del sépalo** presenta correlaciones negativas moderadas con las variables de pétalo, especialmente con la longitud (-0.42), reflejando que flores con sépalos más anchos tienden a tener pétalos más cortos.

Interpretación

La combinación de boxplots y correlaciones permite concluir que:

1. Las variables de pétalo son más informativas para la clasificación, al mostrar mayor dispersión y correlaciones fuertes entre sí.
2. Las variables de sépalo, aunque menos discriminativas, aportan información complementaria y ayudan a refinar las fronteras de decisión del modelo.
3. La presencia de atípicos en el ancho del sépalo indica que algunos individuos pueden ser más difíciles de clasificar correctamente.

Este análisis refuerza la hipótesis de que la clasificación del iris depende principalmente de las dimensiones del pétalo, mientras que las características del sépalo cumplen un rol secundario en la diferenciación de especies.

Pairplot de Características

La Figura 3 permite observar la relación entre pares de características. Se aprecia que **Iris-setosa** es linealmente separable, mientras que **versicolor** y **virginica** presentan solapamiento.

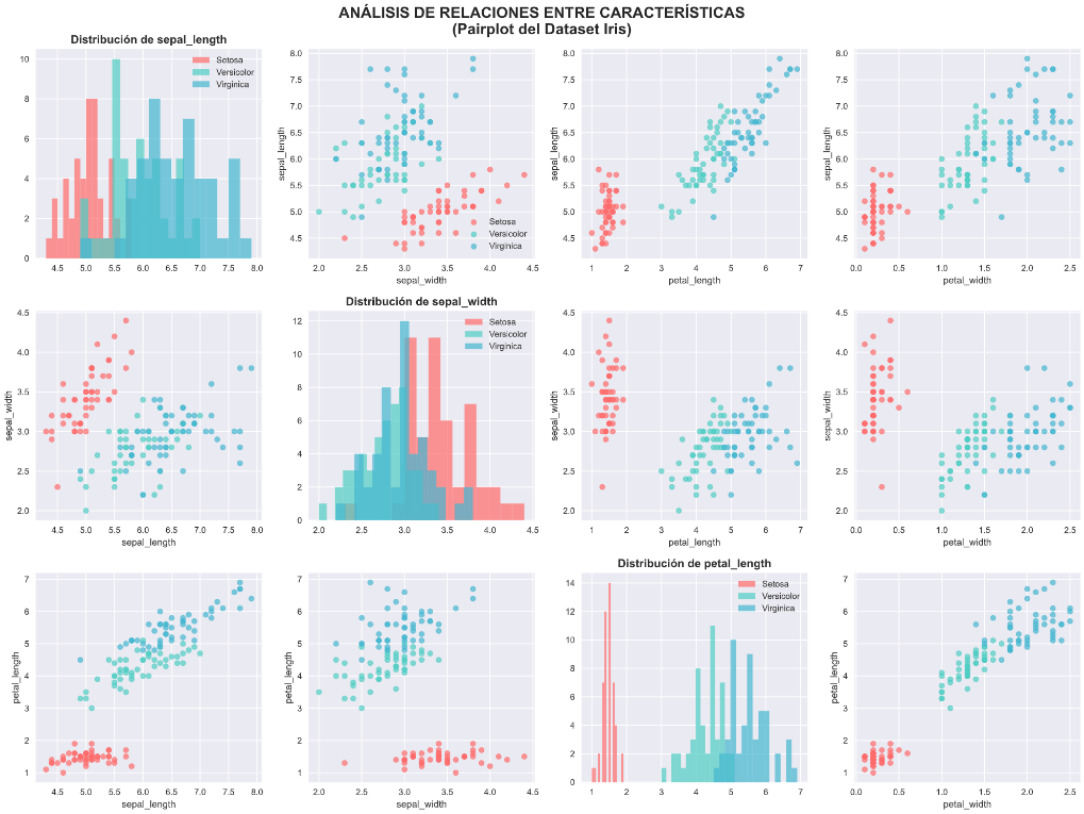

Figure 3: Análisis de relaciones entre características del dataset Iris (Pairplot).

Coeficientes y R^2 Score

La Figura 4 En la Figura se presentan los coeficientes obtenidos en los modelos lineales para cada clase de iris (estrategia One-vs-Rest), así como el valor de R^2 como métrica de ajuste del modelo por especie.

Coeficientes de los Modelos

El gráfico de barras de la izquierda muestra la contribución de cada característica en la clasificación:

- Para **Iris-setosa**, los coeficientes negativos en *petal_length* y *petal_width* indican que esta especie se caracteriza por tener pétalos mucho más pequeños en comparación con las otras clases.
- En **Iris-versicolor**, el coeficiente positivo dominante en *petal_length* sugiere que esta característica es la más relevante para diferenciarla, aunque comparte solapamiento con *virginica*.

- En **Iris-virginica**, el coeficiente más alto aparece en *petal_width*, lo cual refleja que esta especie posee pétalos significativamente más anchos, siendo una variable crítica para su identificación.

R² Score por Modelo

El gráfico de la derecha evidencia el grado de ajuste de cada modelo:

- **Iris-setosa**: obtiene un valor de $R^2 = 0.907$, lo que indica que el modelo captura casi en su totalidad la variabilidad de esta clase. Esto coincide con la alta separabilidad de *setosa*.
- **Iris-versicolor**: presenta un $R^2 = 0.257$, el más bajo de los tres modelos. Esto refleja la dificultad del modelo lineal para distinguir esta especie, debido a su fuerte solapamiento con *virginica*.
- **Iris-virginica**: alcanza un $R^2 = 0.617$, lo que muestra un ajuste intermedio. Si bien el modelo logra capturar cierta diferenciación, aún se observa confusión con *versicolor*.

Interpretación

El análisis conjunto de coeficientes y R^2 sugiere que:

1. La regresión lineal es muy efectiva para separar *Iris-setosa*, dado que sus características de pétalo son claramente distintivas.
2. Existe un reto considerable en diferenciar entre *versicolor* y *virginica*, pues ambas comparten valores cercanos en longitud y ancho de pétalo.
3. El ancho de pétalo (*petal_width*) y la longitud de pétalo (*petal_length*) emergen como las variables más influyentes en la clasificación.

Estos resultados confirman que, aunque el modelo lineal puede capturar patrones importantes, su capacidad es limitada frente a especies con solapamiento, lo que justifica explorar modelos más complejos como la regresión logística o máquinas de soporte vectorial (SVM).

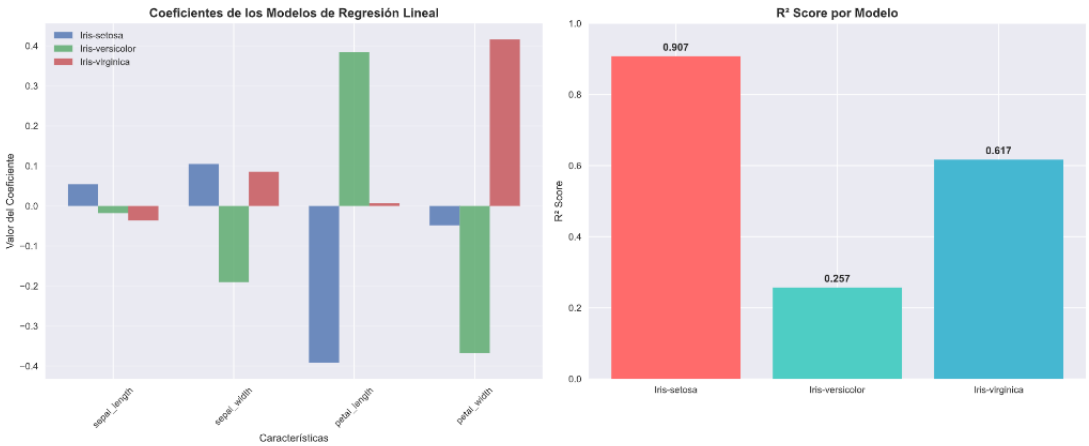

Figure 4: Coeficientes de los modelos y R^2 score por clase.

Matriz de Confusión y Accuracy

La Figura 5 presenta la matriz de confusión sobre el conjunto completo, junto con el análisis de precisión.

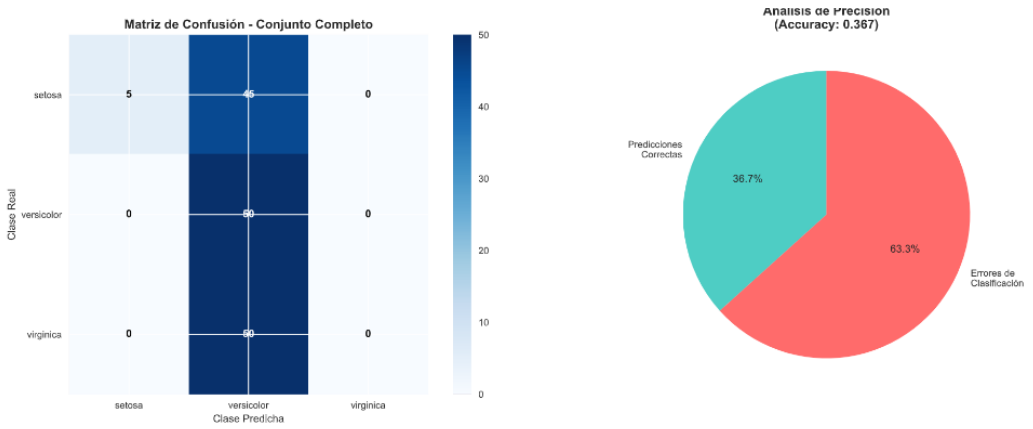

Figure 5: Matriz de confusión y análisis de precisión del modelo.

Fronteras de Decisión

En la Figura 6 se grafican las fronteras de decisión usando sólo dos características (sépalos). Se evidencia la dificultad del modelo para separar versicolor y virginica.

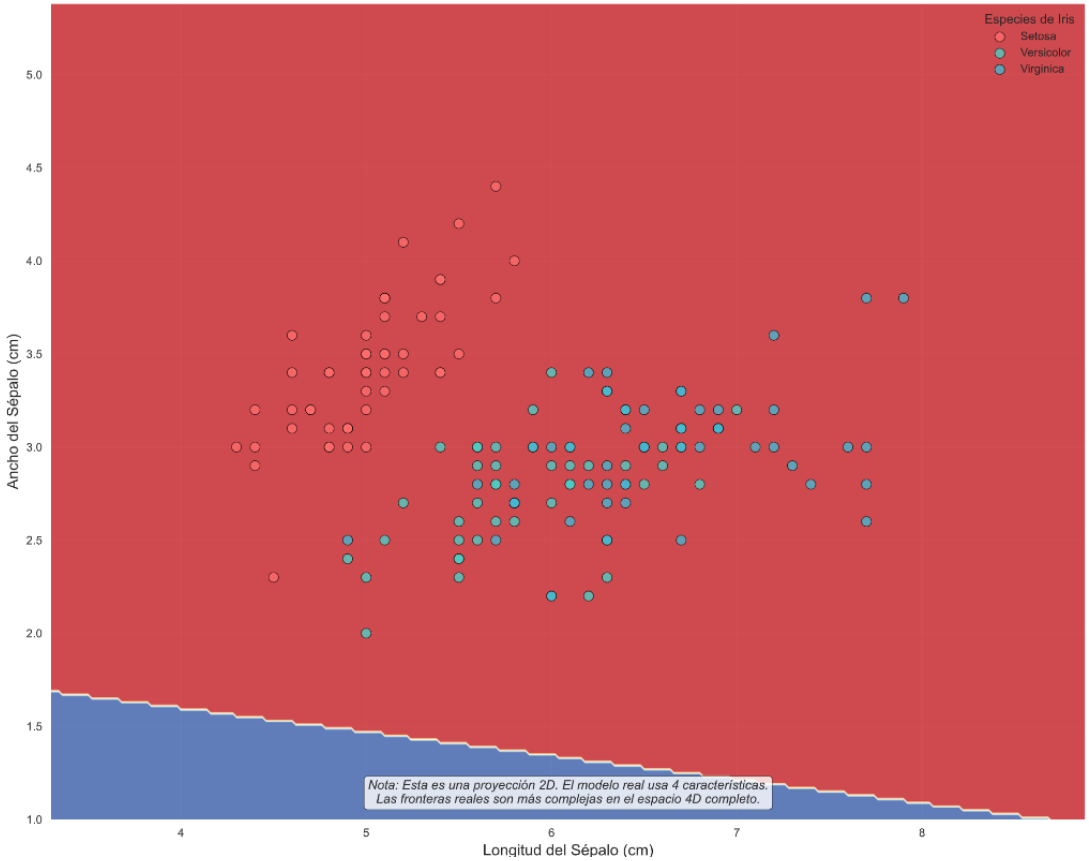

Figure 6: Fronteras de decisión del modelo (proyección 2D).

Conclusiones y Recomendaciones

- El modelo de regresión lineal alcanzó una precisión del **35.2%** en entrenamiento y **33.3%** en prueba.
- La especie **Iris-setosa** es linealmente separable y se clasifica sin errores.
- Las especies **versicolor** y **virginica** presentan mayor dificultad debido a que no son linealmente separables.
- Se recomienda implementar **regresión logística**, **SVM con kernel RBF** o **Random Forest** para mejorar el rendimiento.
- Este proyecto demuestra de forma educativa cómo la regresión lineal puede aplicarse a clasificación multiclase, pero también sus limitaciones frente a datos no linealmente separables.

Repositorio del Proyecto

Con el fin de garantizar la transparencia, reproducibilidad y accesibilidad del trabajo realizado, se ha dispuesto un repositorio en la plataforma GitHub donde se encuentra todo el código fuente y recursos utilizados en este proyecto.

- **Repositorio:** `https://github.com/ju4nesria/iris-project`
- **Contenido principal:**
  1. Código fuente en Python para la implementación del clasificador basado en regresión lineal (`IrisLinearClassifier`).
  2. Scripts para el preprocesamiento de datos, entrenamiento, evaluación y generación de gráficas.
  3. Informes en formato PDF y documentación en LaTeX que describen los experimentos realizados.
  4. Recursos adicionales, como gráficos y visualizaciones, que apoyan el análisis.
- **Licencia:** El repositorio está disponible de forma pública, lo que permite su consulta, descarga y uso educativo.

Este repositorio constituye un recurso complementario que facilita la verificación de resultados y sirve como base para futuras extensiones del trabajo, tales como la incorporación de otros modelos de clasificación (e.g., regresión logística, SVM o redes neuronales).

Referencias

- Fisher, R. A. (1936). *The use of multiple measurements in taxonomic problems.* Annals of Eugenics.
- UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/iris`
- Documentación de Scikit-Learn: `https://scikit-learn.org/`