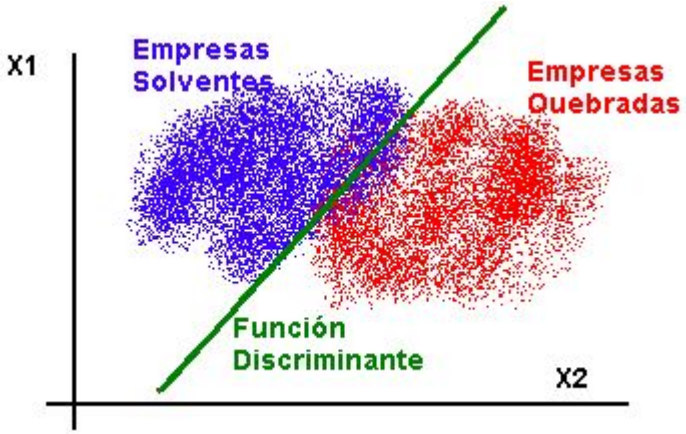
Practicando con el modelo: Análisis discriminante

Sistemas de aprendizaje automático

Profesor:
David Campoy Miñarro

¿Qué es el análisis discriminante?

El análisis discriminante lineal, *linear dicriminant analysis* (LDA), es una técnica de análisis multivariante de clasificación supervisada utilizada para predecir a qué grupo se va a incorporar un elemento nuevo, cuya pertenencia se desconoce, a partir de otros conjuntos previamente conocidos.



Calcula unas funciones lineales a partir de los atributos de su perfil, donde la función que alcanza mayor valor define el grupo al que pertenece el nuevo elemento de forma más probable.

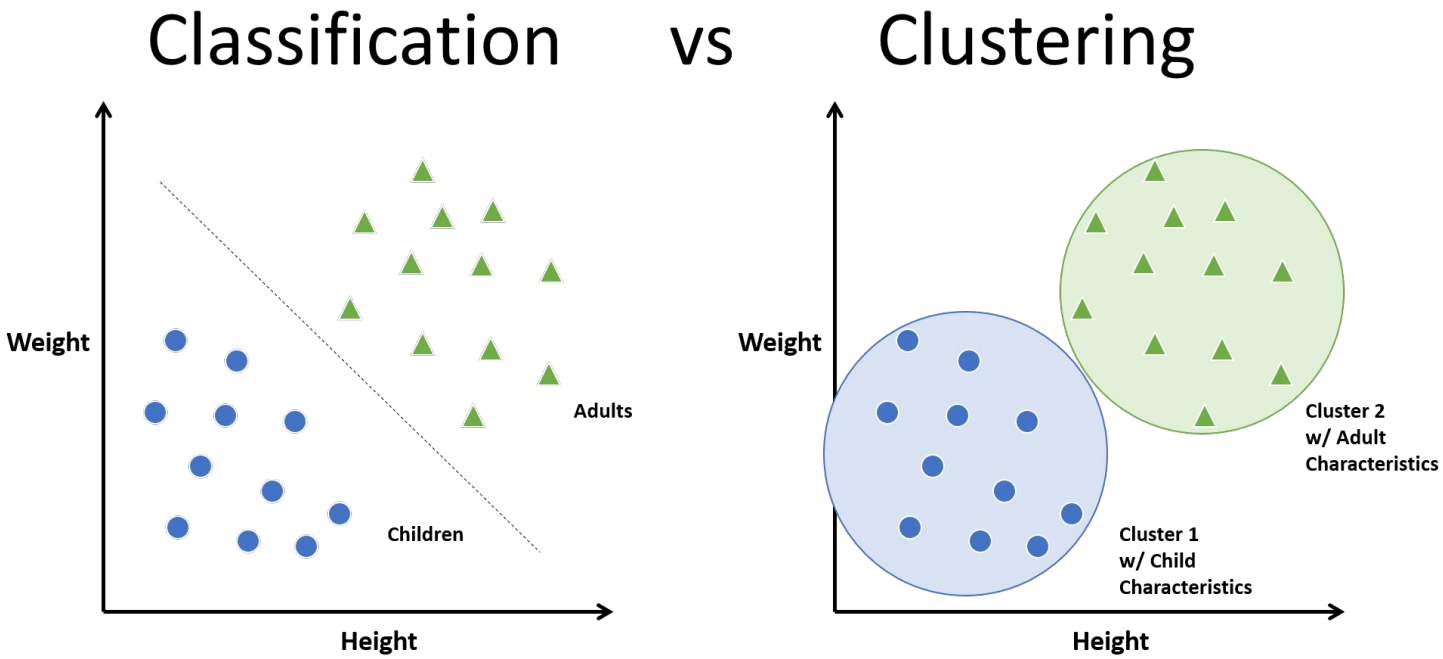
Diferencias entre LDA y clustering

Análisis discriminante: Busca encontrar las características que mejor discriminan entre clases predefinidas.

Clustering: Agrupa datos similares sin tener información previa sobre las clases o etiquetas.

Análisis discriminante: Es un método supervisado, utiliza información de las clases conocidas para encontrar la discriminación óptima.

Clustering: Es un método no supervisado, no utiliza etiquetas o clases previas para agrupar los datos.



Análisis discriminante: Se usa principalmente en problemas de clasificación para reducir la dimensionalidad y mejorar la separación entre clases.

Clustering: Es útil para descubrir patrones, segmentar datos y encontrar estructuras ocultas en conjuntos de datos.

¿Y para qué sirve el análisis discriminante?

Clasificación de múltiples clases:

Cuando tienes un problema de clasificación con más de dos clases y buscas un método que pueda manejar la separación entre múltiples clases de manera efectiva.

Reducción de dimensionalidad con discriminación:

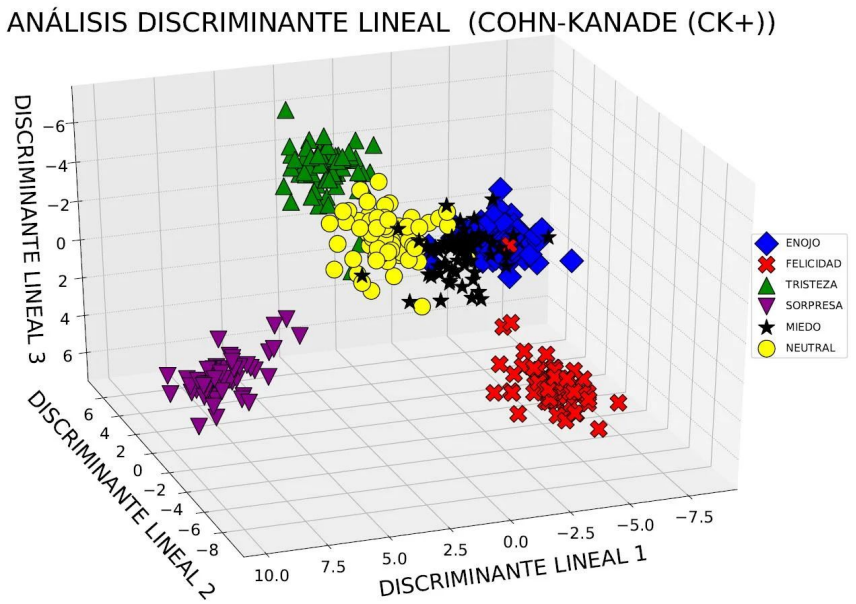
Cuando buscas reducir la dimensionalidad de tus datos manteniendo la capacidad de discriminar entre clases.

Preparación de datos para otros modelos:

Cuando deseas utilizar AD como un paso de preprocesamiento para otros algoritmos de aprendizaje automático.

Reconocimiento facial o biométrico:

En escenarios de reconocimiento facial o biométrico, donde se busca reducir la dimensionalidad y clasificar patrones.



¿Qué es la reducción de dimensionalidad?

La "Reducción de dimensionalidad" es la técnica disminuir el número de variables en un conjunto de datos, **manteniendo al mismo tiempo la mayor cantidad posible de información relevante**.

El objetivo es simplificar la representación de los datos al eliminar características redundantes, ruidosas o irrelevantes, lo que puede facilitar su visualización, comprensión y análisis, así como mejorar el rendimiento de ciertos algoritmos de aprendizaje automático al reducir la complejidad del espacio de características.

Conjunto de datos [editar]

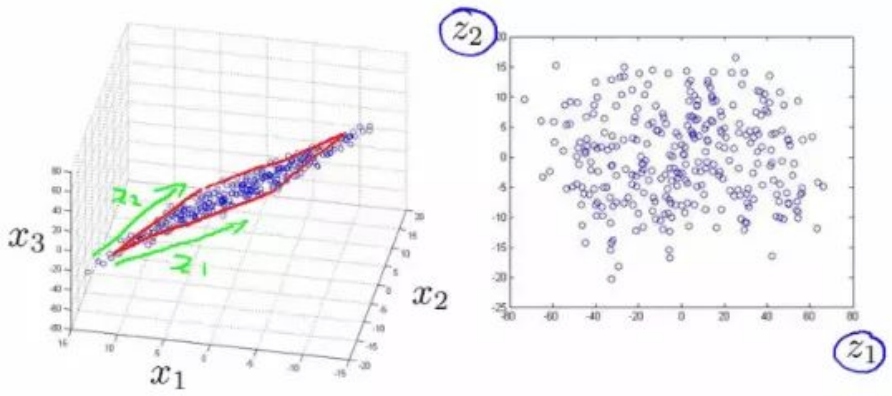
Fisher's *Iris* Data

Largo de sépalo	Ancho de sépalo	Largo de pétalo	Ancho de pétalo	Especies
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*

Por ejemplo:

Recuerda que la tabla de las flores Iris tiene 4 variables (dimensiones) para determinar el tipo de flor. ¿Y si únicamente fueran necesarias 3 variables?

Pues resulta que el análisis discriminante es capaz de averiguar las dimensiones más importantes.

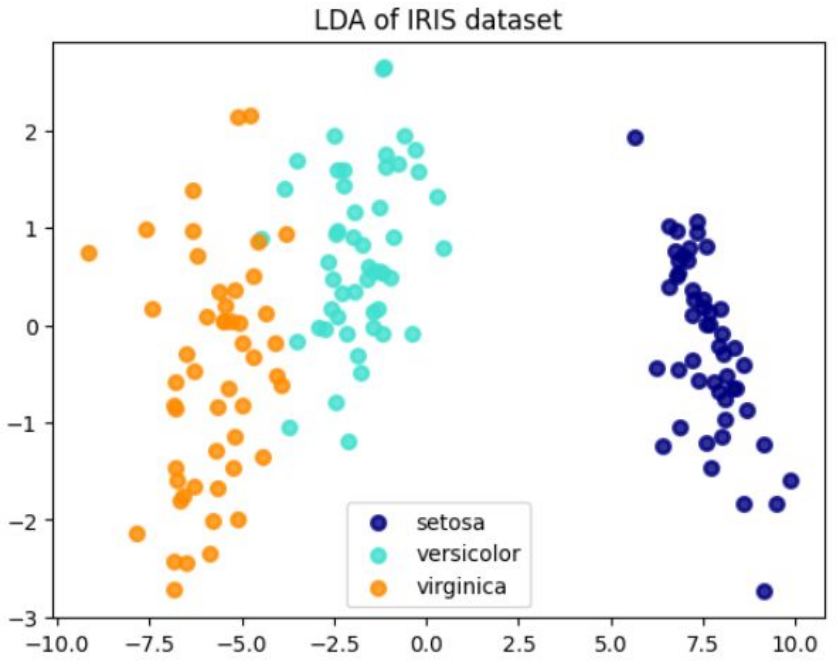


Actividad 1.

```python
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Cargar el conjunto de datos de iris
iris = datasets.load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# Aplicar LDA para reducción de dimensionalidad
lda = LinearDiscriminantAnalysis(n_components=2)
X_r = lda.fit(X, y).transform(X)

# Visualizar los resultados
plt.figure()
colors = ['navy', 'turquoise', 'darkorange']
lw = 2

for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=.8, lw=lw,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA of IRIS dataset')

plt.show()
```

Este programa utiliza LDA para agrupar el dataset de Iris.

Pero para mostrarlo en una gráfica de 2 dimensiones, primero ha tenido que convertir las 4 entradas en 2 entradas.



Actividad 1.

```python
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Cargar el conjunto de datos de iris
iris = datasets.load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# Aplicar LDA para reducción de dimensionalidad
lda = LinearDiscriminantAnalysis(n_components=2)
X_r = lda.fit(X, y).transform(X)

# Crear un DataFrame para mostrar los datos antes y después de LDA
data_before_lda = pd.DataFrame(X, columns=iris.feature_names)
data_after_lda = pd.DataFrame(X_r, columns=['LDA Component 1', 'LDA Component 2'])

# Mostrar la tabla con los datos antes y después de LDA
print("Datos antes de LDA:")
print(data_before_lda.head(4))  # Muestra las primeras 4 filas

print("\nDatos después de LDA:")
print(data_after_lda.head(4))  # Muestra las primeras 4 filas
```

La siguiente tabla muestra la conversión de las 4 dimensiones en 2 dimensiones:

Datos antes de LDA:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

Datos después de LDA:

	LDA Componente 1	LDA Componente 2
0	8.061800	-0.300421
1	7.128688	0.786660
2	7.489828	0.265384
3	6.813201	0.670631
4	8.132309	-0.514463
5	7.701947	-1.461721
6	7.212618	-0.355836
7	7.605294	0.011634
8	6.560552	1.015164
9	7.343060	0.947319

Actividad 1.

¿Y cambiarlo a 3 dimensiones?

```
número máximo de componentes = min(número de características, número de clases -1 )
```

```
ValueError: n_components cannot be larger than min(n_features, n_classes - 1).
```

En el caso del conjunto de datos Iris:

- 4 características (dimensiones, variables de entrada)
- 3 clases (diferentes salidas).

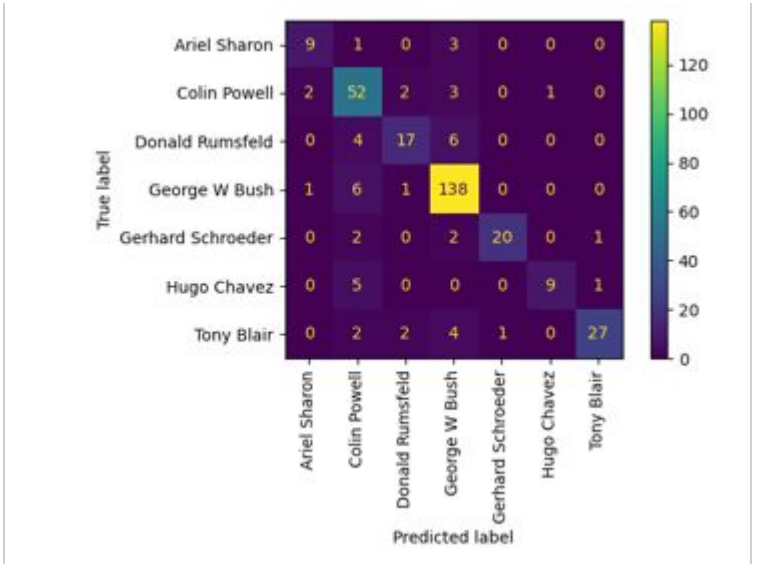
Por lo tanto, el número máximo de componentes que se pueden generar con LDA es 2 (mínimo entre 4 características y 3 clases menos uno).

Solución: PCA (Análisis de Componentes Principales)

Actividad 2. Biometría

Conjunto de datos "*Labeled Faces in the Wild*" (LFW).

Contiene imágenes de caras de personas famosas y es utilizado en tareas de reconocimiento facial. Vamos a realizar un ejemplo simplificado utilizando LDA para reducir la dimensionalidad y visualizar las clases en un espacio bidimensional.

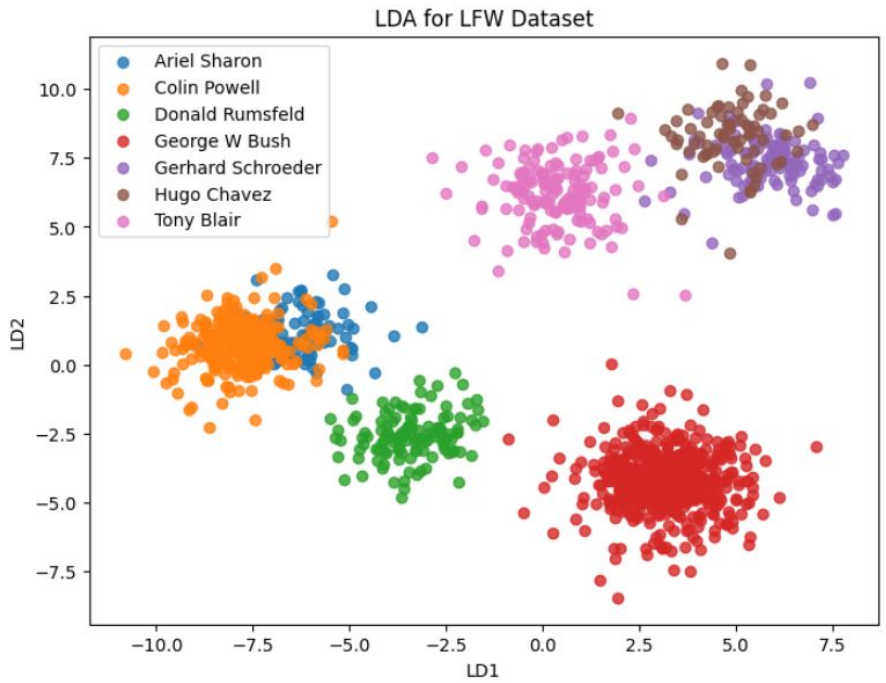


https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html

Actividad 2. Biometría

```python
from sklearn.datasets import fetch_lfw_people
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import matplotlib.pyplot as plt

# Cargar el conjunto de datos LFW
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

# Obtener características e etiquetas
X = lfw_people.data
y = lfw_people.target
print("Número de características originales:", X.shape[1])

# Aplicar LDA para reducir la dimensionalidad a 2 componentes
lda = LDA(n_components=2)
X_r = lda.fit(X, y).transform(X)

# Visualizar los resultados
plt.figure(figsize=(8, 6))

for i in range(len(lfw_people.target_names)):
    plt.scatter(X_r[y == i, 0], X_r[y == i, 1], alpha=.8, label=lfw_people.target_names[i])

plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA for LFW Dataset')
plt.xlabel('LD1')
plt.ylabel('LD2')

plt.show()
```

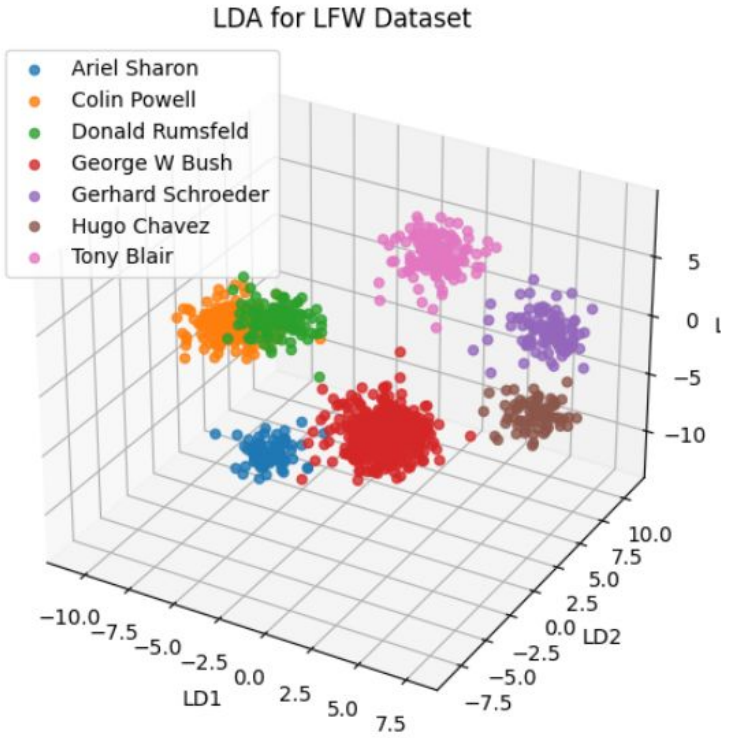
```
Número de características originales: 1850
```



Hemos reducido la dimensionalidad a 2, pero ¿cuáles son las dimensiones posibles?

Actividad 2. Biometría

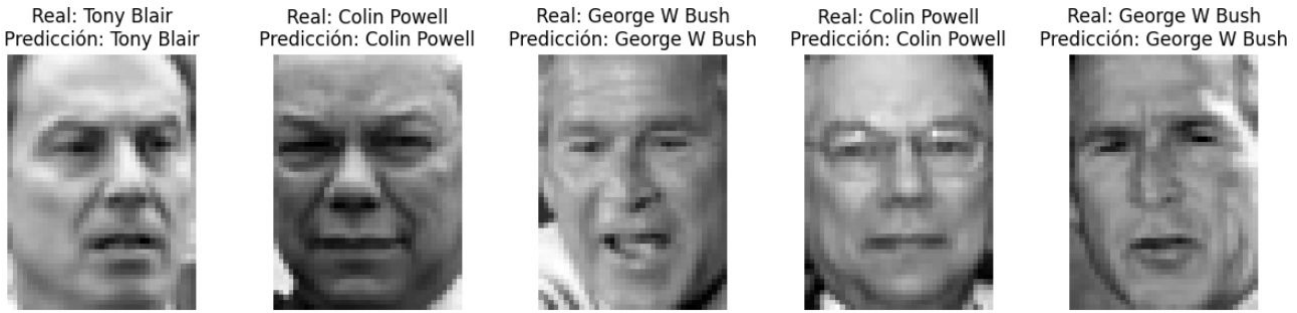
```python
from sklearn.datasets import fetch_lfw_people
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import matplotlib.pyplot as plt

# Cargar el conjunto de datos LFW
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

# Obtener características e etiquetas
X = lfw_people.data
y = lfw_people.target

# Aplicar LDA para reducir la dimensionalidad a 3 componentes
lda = LDA(n_components=3)
X_r = lda.fit(X, y).transform(X)

# Visualizar los resultados
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

for i in range(len(lfw_people.target_names)):
    ax.scatter(X_r[y == i, 0], X_r[y == i, 1], X_r[y == i, 2], alpha=.8, label=lfw_people.target_names[i])

ax.legend(loc='best', shadow=False, scatterpoints=1)
ax.set_title('LDA for LFW Dataset')
ax.set_xlabel('LD1')
ax.set_ylabel('LD2')
ax.set_zlabel('LD3')

plt.show()
```



LDA para clasificación

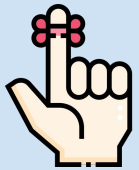
```
from sklearn.datasets import fetch_lfw_people
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
import matplotlib.pyplot as plt
import numpy as np

lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
X = lfw_people.data
y = lfw_people.target

_, X_test, _, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

lda = LDA(n_components=None)
X_lda = lda.fit_transform(X, y)

lda_classifier = LDA()
lda_classifier.fit(X_lda, y)

X_test_lda = lda.transform(X_test)
y_pred = lda_classifier.predict(X_test_lda)
```

```
accuracy = np.mean(y_pred == y_test)

random_indices = np.random.choice(len(X_test), 5, replace=False)
X_samples = X_test[random_indices]
y_samples = y_test[random_indices]

X_samples_lda = lda.transform(X_samples)
y_pred_samples = lda_classifier.predict(X_samples_lda)

plt.figure(figsize=(15, 3))
for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(X_samples[i].reshape(50, 37), cmap='gray')
    plt.title(f"Real:
{lfw_people.target_names[y_samples[i]]}\nPredicción:
{lfw_people.target_names[y_pred_samples[i]]}")
    plt.axis('off')

plt.show()
```



Actividad 3. LDA para preprocesamiento

Preparación de datos para otros modelos:



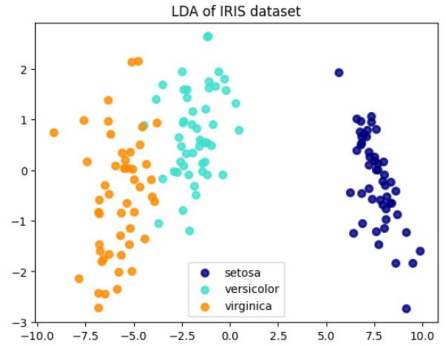
Cuando deseas utilizar LDA como un paso de preprocesamiento para otros algoritmos de aprendizaje automático.

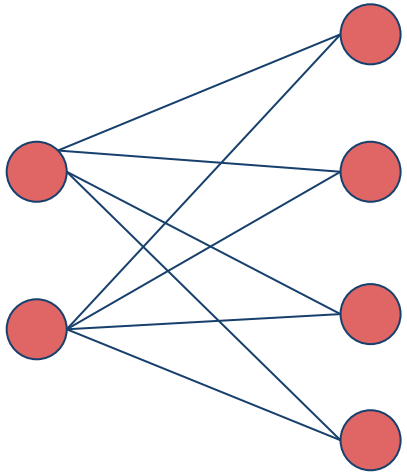
Puede ser útil como paso de preprocesamiento para algoritmos de aprendizaje automático, incluyendo **redes neuronales**.

Fisher's *Iris* Data

Largo de sépalo	Ancho de sépalo	Largo de pétalo	Ancho de pétalo	Especies
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*

LDA

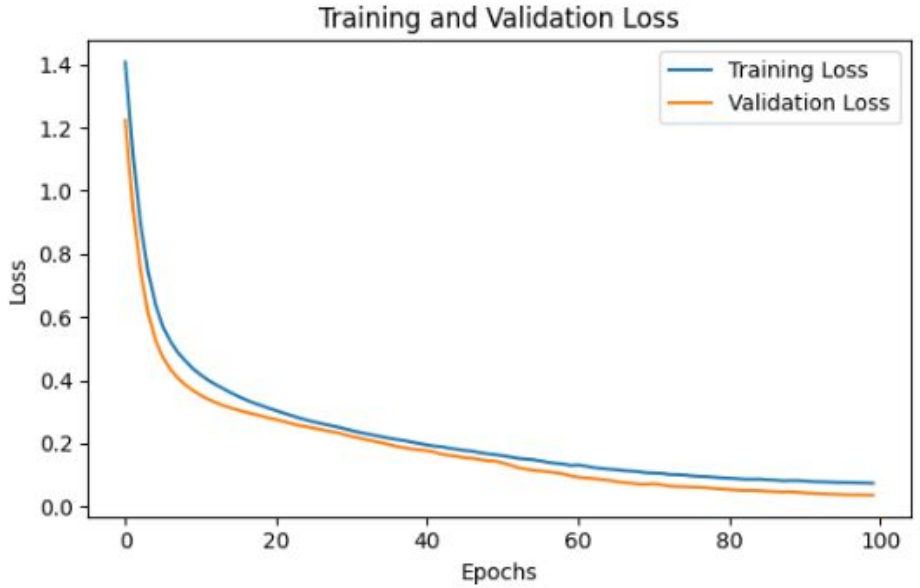


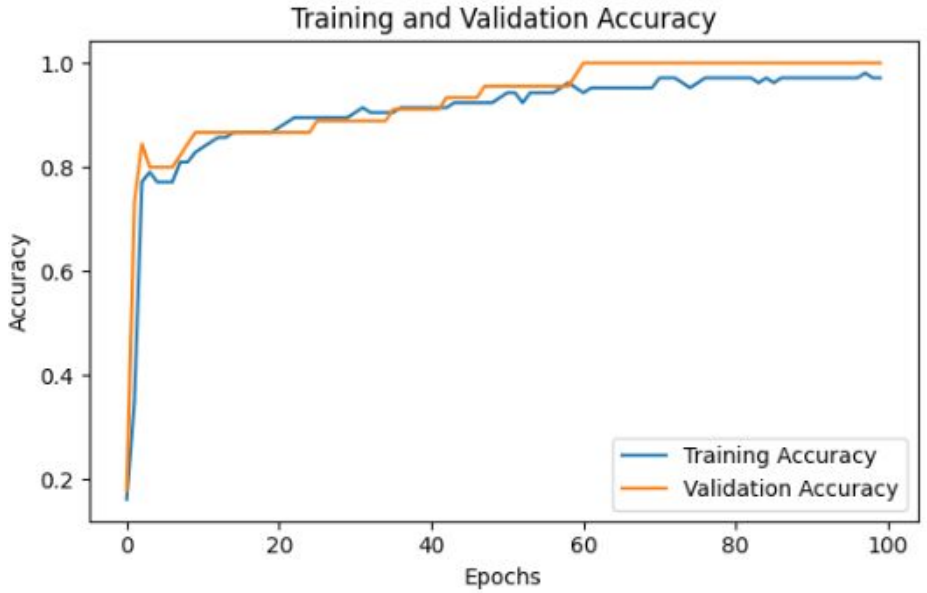


Tenemos 4 variables de entrada

Lo hemos reducido a 2

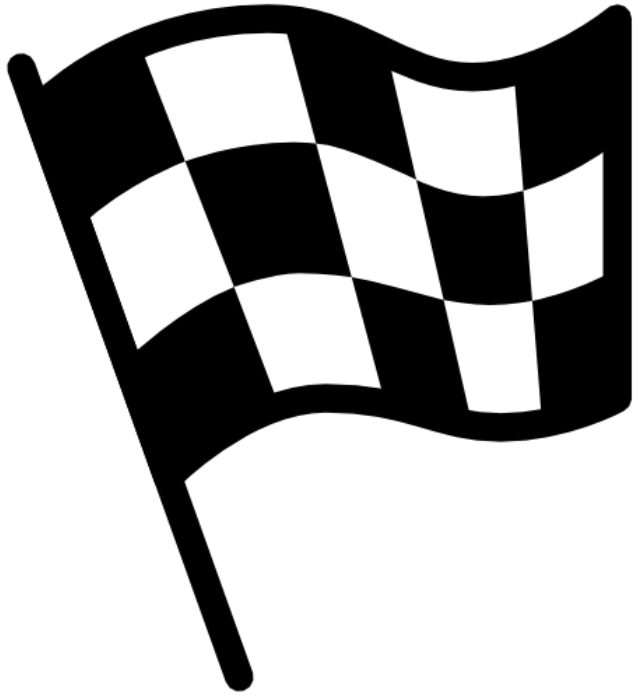
```python
import numpy as np
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.3,
random_state=42)
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
lda = LDA(n_components= 2)
X_train_lda = lda.fit_transform(X_train_scaled, y_train)
X_test_lda = lda.transform(X_test_scaled)

model = tf.keras.Sequential([
    tf.keras.layers.Dense( 50, activation='relu', input_shape=( 2,)),
    tf.keras.layers.Dense( 50, activation='relu'),
    tf.keras.layers.Dense( 3, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy' ,
metrics=['accuracy'])
model.fit(X_train_lda, y_train, epochs= 100, batch_size=32, verbose=0)
test_loss, test_accuracy = model.evaluate(X_test_lda, y_test, verbose= 0)
print(f"Precisión de la red neuronal después de LDA: {test_accuracy:.2f}")
```





¿Qué has aprendido?

- Utilizar el modelo Análisis discriminante.
- Cambiar la dimensionalidad (LDA)
- Diferenciar clasificación y clusterización
- Utilizar LDA para clasificar.
- Utilizar LDA para preprocesammiento.
- Utilizar LDA para biometría.
- ¿te parece poco?



"Inteligencia artificial, aprendizaje profundo, aprendizaje automático… te dediques a lo que te dediques, si no lo comprendes tienes que ponerte con ello y aprender qué es. Porque de lo contrario serás un dinosaurio dentro de 3 años"

Mark Cuban, empresario estadounidense, inversionista, y dueño de los Mavericks de Dallas de la NBA