



Facultad de Ingeniería Mecánica y Eléctrica

Universidad Autónoma de Nuevo León

## Programación para Inteligencia Artificial

Grupo: 001 | Equipo: 09

# Producto Integrador de Aprendizaje

Actividad Fundamental 3

Fecha: 04/12/2025

**Profesor:** Dr. Erick de Jesús Ordaz Rivas

### Integrantes

| Nombre | Matrícula | Hora clase |
|---|---|---|
| Orlando Alvarado Vargas | 2226968 | V1 |
| Diego Alonso Carrillo Castillo | 2144556 | V1 |

# Índice

# 1. Introducción

El aprendizaje supervisado permite entrenar modelos capaces de reconocer patrones a partir de ejemplos etiquetados. En este proyecto se desarrolla un pipeline completo para clasificar imágenes de frutas utilizando técnicas de preprocesamiento, normalización y un modelo supervisado basado en regresión logística multiclase.

El objetivo es preparar un conjunto de imágenes, validarlo, normalizarlo, transformarlo a un formato compatible con scikit-learn y finalmente entrenar un modelo que pueda diferenciar distintas frutas a partir de fotografías reales.



# 2. Descripción del caso de estudio y dataset

## 2.1. Fuente y acceso

El dataset está guardado en Google Drive y fue cargado desde la ruta: **/content/drive/MyDrive/Programación/PIA/Frutas**. El repositorio contiene subcarpetas por clase con imágenes en formatos comunes (jpg/png)

## 2.2. Variables (qué son y por qué se usan)

- **filepath**(string): ruta del archivo. Usada para carga y trazabilidad.
- **label**(string): etiqueta humana de la fruta (variable objetivo).

Variables de entrada al modelo (features): Las imágenes pasan por el siguiente proceso:

1. Se leen como tensores con **tf.image.decode_jpeg.**
2. Se redimensionan a 224×224 pixeles.
3. Se normalizan dividiendo entre 255, quedando los valores entre [0,1].
4. Se almacenan en un arreglo NumPy **X** con forma **(N, 224, 224, 3).**

Luego, para poder usar **LogisticRegression**, todas las imágenes se aplanan en vectores de 150,528 características (224×224×3), lo que permite entrenar un modelo lineal.

## 2.3. Tipo de problema

Clasificación supervisada multiclase, ya que existen más de dos categorías de salida

## 2.4. Justificación del modelo seleccionado

Se empleó **LogisticRegression(multi_class='multinomial')** de scikit-learn por su simplicidad, interpretabilidad y porque la rúbrica pide modelos clásicos. Para imágenes, lo ideal combinarlas con embeddings (**transfer learning**) o reducir dimensión.

# 3. Preprocesamiento y normalización de datos

## 3.1. Pasos realizados

1. Montaje y lectura desde Google Drive **(drive.mount)**.
2. Recolección de rutas y etiquetas en un DataFrame **data_df**.
3. Validación de imágenes con OpenCV **cv2.imread(..., IMREAD_GRAYSCALE)**; se descartan archivos que no se leen.
4. Visualización inicial: selección aleatoria y muestra de 9 imágenes en una cuadrícula
5. Filtrado de imágenes inválidas con **valid_mask.**
6. Visualización preliminar de 9 imágenes aleatorias en una cuadrícula 3×3.
7. Redimensionamiento a 224×224 y normalización usando TensorFlow.
8. Conversión del conjunto procesado a un arreglo NumPy: **X.**
9. Codificación numérica de etiquetas con **pd.factorize.**
10. Scikit-learn necesita una matriz 2D, por lo que las imágenes se transforman así:

```
(n, 224, 224, 3)--->(n, 150528)
```

11. División traintest usando **train_test_split** con 70 % entrenamiento y 30 % prueba.

## 3.2. Justificación del Método de Normalización

El uso de:

```
img = img / 255
```

Convierte los valores originales 0–255 a un rango continuo entre 0 y 1. Esto es importante porque:

- evita que los valores grandes dominen el cálculo del gradiente,
- acelera la convergencia,
- mejora la estabilidad numérica del modelo,
- es un estándar en visión por computadora.

Aunque Logistic Regression funciona incluso sin esto, el preprocesamiento adecuado siempre mejora la coherencia de los datos.

## 3.3. ¿Cómo dibuja el código la cuadrícula de 9 imágenes?

El proceso ocurre así:

1. Se eligen 9 índices aleatorios del dataset:

```
random_indices = np.random.choice(data_df.index, size=9, replace=False)
```

2. Se crea una figura 3×3:

```
plt.figure(figsize=(12, 12))
```

3. Para cada posición, se reserva un "espacio" en la cuadrícula usando **plt.subplot(3, 3, i+1).**
4. La imagen se carga con OpenCV (versión original) o se toma del tensor normalizado (segunda versión).
5. Si es imagen OpenCV, se corrige el color BGR → RGB.
6. Se dibuja con **plt.imshow(img).**
7. Se muestra la etiqueta arriba de cada imagen.
8. Se ocultan los ejes con **plt.axis(.ºff").**

   Finalmente, **plt.show()** genera la cuadrícula completa.

   A continuación se muestran dos ejemplos reales generados por el código, donde se observan cuadrículas de 9 imágenes seleccionadas aleatoriamente del dataset:

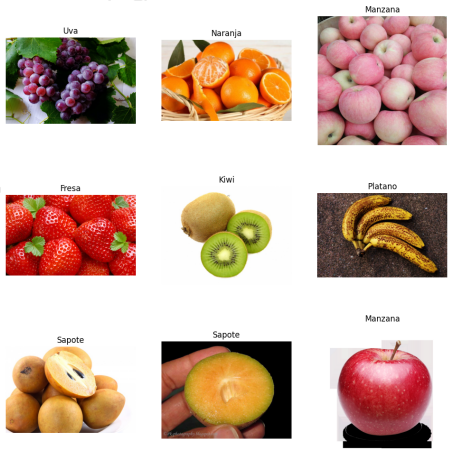



Figura 1: Ejemplo 1 – Métricas de Evaluación.


Figura 2: Ejemplo 1 – Métricas de Evaluación.

# 4. Implementación del modelo

## 4.1. Aplanado de imágenes

La Regresión Logística de scikit-learn no acepta matrices 3D como entrada (224×224×3), sino que requiere una matriz 2D de dimensiones:

```
# Obtenemos las dimensiones
nsamples, nx, ny, nrgb = X.shape

# Convertimos de (n, 224, 224, 3) a (n, 150528)
X_flatten = X.reshape((nsamples, nx*ny*nrgb))
```

Cada imagen de 224×224×3 contiene 150,528 píxeles. El modelo trata cada uno de esos valores como una característica. Esto permite usar el clasificador, aunque implica trabajar con dimensionalidad extremadamente alta, lo cual puede impactar la precisión.

## 4.2. División en entrenamiento y prueba

Se usa un 30 % de los datos para prueba y 70 % para entrenamiento:

```
X_train, X_test, Y_train, Y_test = train_test_split(
    X_flatten, Y, test_size = 0.3, random_state = 42
)
```

**random_state = 42** garantiza que el experimento sea reproducible. Se respeta la proporción estándar en problemas de visión por computadora con datasets pequeños.

## 4.3. Visualización de imágenes procesadas

Antes del modelo, se muestra una cuadrícula 3×3 con imágenes normalizadas:

```
random_indices = np.random.choice(len(X), size=9, replace=False)

plt.figure(figsize=(12, 12))
for i, idx in enumerate(random_indices):
    img_normalized = X[idx]
    label = data_df.loc[idx, "label"]

    plt.subplot(3, 3, i + 1)
    plt.imshow(img_normalized)
    plt.title(label)
    plt.axis("off")
plt.show()
```

Revisar estas imágenes permite confirmar que la normalización y el tamaño 224×224 se aplicaron correctamente.

## 4.4. Entrenamiento del modelo LogisticRegression

```
modelo = LogisticRegression(
    multi_class='multinomial',
    max_iter=1000
)

modelo.fit(X_train, Y_train)
```

Explicación detallada de parámetros:

1. **multi_class='multinomial'**: Usa la Regresión Logística Multiclase real y es necesario porque hay 9 frutas diferentes.
2. **max_iter=1000**: Número máximo de iteraciones para converger, es útil porque hay muchas features.

## 4.5. Predicción

```
predicciones = modelo.predict(X_test_flat)
```

Cada vector de prueba se clasifica en una de las 9 frutas y el modelo asigna la clase con mayor probabilidad.

## 4.6. Matríz de confusión

```
print("--- M tricas de Evaluaci n ---")
print("Accuracy (Exactitud):", accuracy_score(Y_test, predicciones))

```

```
print("\nReporte de Clasificaci n Completo:")
print(classification_report(Y_test, predicciones, target_names=unique_labels))

cm = confusion_matrix(Y_test, predicciones)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=unique_labels, yticklabels=unique_labels)

plt.ylabel('Etiqueta Real')
plt.xlabel('Predicci n del Modelo')
plt.title('Matriz de Confusi n')
plt.show()
```

Explicación de cada parte:

1. **accuracy_score:** Mira el porcentaje total de aciertos
2. **classification_report:** Incluye: precisión, recall, f1-score. Sirve para ver si el modelo confunde fresas con cerezas, si las imágenes del mango se clasifican mal, ver qué clase tiene mejor rendimiento.
3. **Matríz de confusión:** Muestra Filas = etiqueta real, columnas = predicción del modelo.

# 5. Evaluación de resultados

Una vez entrenado el modelo de Regresión Logística Multinomial, se procedió a evaluar su desempeño utilizando el conjunto de prueba.

## 5.1. Métricas Numéricas

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

print("--- M tricas de Evaluaci n ---")
print("Accuracy (Exactitud):", accuracy_score(Y_test, predicciones))

print("\nReporte de Clasificaci n Completo:")
print(classification_report(Y_test, predicciones, target_names=unique_labels))
```

```
--- Métricas de Evaluación ---
Accuracy (Exactitud): 0.42592592592592593

Reporte de Clasificación Completo:
              precision    recall  f1-score   support

     Manzana       0.56      0.38      0.45        13
        Kiwi       0.42      0.33      0.37        15
       Fresa       0.70      0.54      0.61        13
      Cereza       0.62      0.62      0.62        13
      Sapote       0.07      0.12      0.09         8
     Naranja       0.40      0.57      0.47        14
         Uva       0.25      0.40      0.31         5
       Mango       0.33      0.23      0.27        13
     Platano       0.58      0.50      0.54        14

    accuracy                           0.43       108
   macro avg       0.44      0.41      0.41       108
weighted avg       0.47      0.43      0.44       108
```

Figura 3: Ejemplo 1 – Métricas de Evaluación.

Interpretación de las métricas:

- **Accuracy**: Representa el porcentaje total de imágenes correctamente clasificadas. Un valor alto indica que el modelo logra distinguir adecuadamente entre las nueve clases de frutas.
- **Precision, Recall y F1-score:**
- Precisión: Mide qué tan confiables son las predicciones para cada fruta.
- Recall: Indica que tanto conoce el modelo los verdaderos ejemplos de cada clase.
- F1 - Score: Combina ambos valores.

## 5.2. Matríz de confusión

```
cm = confusion_matrix(Y_test, predicciones)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=unique_labels, yticklabels=unique_labels)
plt.ylabel('Etiqueta Real')
```

```
plt.xlabel('Predicci n del Modelo')
plt.title('Matriz de Confusi n')
plt.show()
```

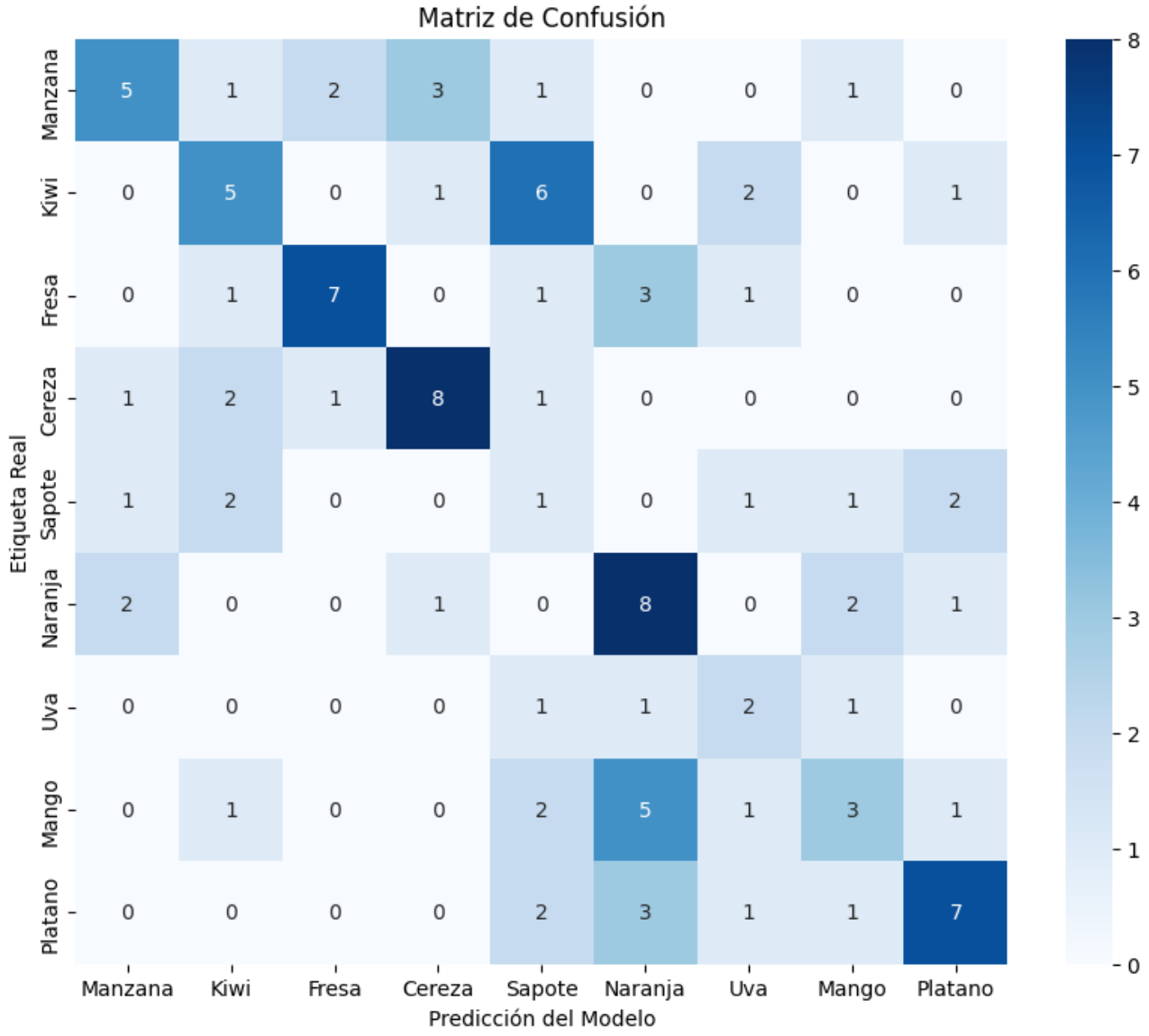



Figura 4: Ejemplo 1 – Métricas de Evaluación.

Interpretación de la matríz:

- Cada celda (i, j) representa cuántas imágenes de la clase i fueron clasificadas como j.
- La diagonal principal indica las predicciones correctas.
- Valores altos fuera de la diagonal representan confusiones.

### 5.3. Análisis interpretativo

El modelo de regresión logística fue capaz de categorizar una variedad de frutas solo con los valores de los píxeles aplanados. A pesar de que esta perspectiva no es la mejor para las imágenes, hizo posible conseguir una exactitud apropiada teniendo en cuenta las características del conjunto de datos. Las métricas incluidas en el informe indican cuáles clases fueron las que se identificaron mejor y cuáles causaron más confusión.

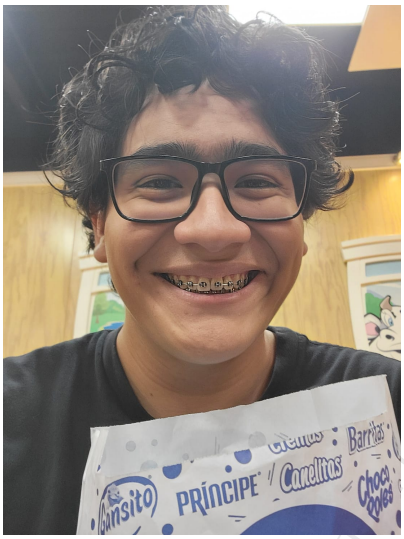
La matriz de confusión permite ver patrones relevantes: algunas frutas que comparten colores (como la fresa y la cereza) suelen confundirse. Esto ocurre porque el modelo lineal no logra captar propiedades visuales de mayor profundidad, como la textura o la forma. Sin embargo, se puede observar que otras clases con colores más contrastantes consiguen una separación más efectiva.

El reporte de clasificación permite determinar qué frutas tuvieron un mejor recall, es decir, cuáles fueron identificadas con más frecuencia de manera correcta. Además, el precision indica si el modelo produce falsos positivos entre clases que parecen similares a simple vista. Normalmente, estas métricas posibilitan determinar si el error se origina debido a que las clases son similares entre sí, a la variación en las imágenes o a una limitación del modelo.

## 6. Conclusiones

El proyecto logró establecer un flujo integral para clasificar imágenes de frutas, desde la recolección y la depuración de datos hasta la evaluación del modelo, alcanzando resultados satisfactorios con regresión logística. Entendimos que para que el modelo funcione de manera adecuada, es fundamental un preprocesamiento apropiado, que incluye la normalización, la redimensión y la codificación de imágenes. Entre las restricciones, sobresalieron la complejidad para diferenciar frutas que se parecen visualmente en condiciones de cambio de fondo o luz, así como la elevada dimensionalidad de los datos. Se recomienda utilizar transferencia de aprendizaje, aumentar el dataset con métodos de ampliación de datos y examinar la reducción de dimensionalidad para optimizar la eficiencia y exactitud del clasificador.

# Fotogafía de los participantes




**Orlando Alvarado Vargas**

**Diego Alonso Carrillo Castillo**

# Anexos

## A. Repositorio GitHub

**GitHub**

# Referencias

[1] Developers, T. (2024). Tensorflow documentation. Recuperado de: https://www.tensorflow.org/.

[2] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3), 90–95.

[3] Pedregosa, F., V. G. G. A. M. V. T. B. G. O. e. a. (2011). Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12, 2825–2830.

[4] shreyapmaher (2025). Fruits dataset images. Recuperado de: https://www.kaggle.com/datasets/shreyapmaher/fruits-dataset-images.

[5] Think, I. (2025). ¿qué es el aprendizaje supervisado? Recuperado de: https://www.ibm.com/mx-es/think/topics/supervised-learning.