

# Práctica 3

Competición Kaggle: Clasificación de Hojas de Tomate

Inteligencia de Negocio

Jesús J. Cantero

Grupo A

[jesusjcl@correo.ugr.es](mailto:jesusjcl@correo.ugr.es)

Curso 2025–2026

Grado en Ingeniería Informática  
Universidad de Granada (Ceuta)

## [IN] Práctica 3 - 25/26

[Submit Prediction](#)

...

[Overview](#) [Data](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Submissions](#)

[Search leaderboard](#)

This leaderboard is calculated with all of the test data.

#	Team	Members	Score	Entries	Last
1	JesusCantero_UGR_IN		0.84782	3	10m
2	YasmineMohamed_UGR_IN		0.00000	1	2h

Captura del Leaderboard de Kaggle – Fecha: 23/12/2025

### Resumen de participación:

**Usuario:** JesusCantero\_UGR\_IN  
**Posición actual:** 1 de 2  
**Mejor Score:** 0.84782  
**Número de entries:** 3

# Índice

<b>1. Introducción</b>	<b>4</b>
1.1. Contexto y Motivación . . . . .	4
1.2. Descripción del Dataset . . . . .	4
1.3. Herramientas y Tecnologías . . . . .	4
1.4. Métrica de Evaluación . . . . .	5
<b>2. Análisis Exploratorio de Datos (EDA)</b>	<b>6</b>
2.1. Distribución de Clases . . . . .	6
2.2. Variables de Fluorescencia . . . . .	6
2.3. Variables Espectrales . . . . .	6
2.4. Análisis de Valores Faltantes y Outliers . . . . .	7
2.5. Reducción de Dimensionalidad (PCA) . . . . .	7
<b>3. Preprocesamiento de Datos</b>	<b>8</b>
3.1. Limpieza de Datos . . . . .	8
3.2. Preparación de Features . . . . .	8
3.3. Arquitectura del Código . . . . .	8
<b>4. Modelado y Experimentación</b>	<b>9</b>
4.1. Metodología . . . . .	9
4.2. Experimento 01: Modelos Baseline . . . . .	9
4.3. Configuración del Mejor Modelo . . . . .	9
<b>5. Registro de Experimentos</b>	<b>10</b>
5.1. Detalle del Experimento 01 . . . . .	10
<b>6. Conclusiones y Trabajo Futuro</b>	<b>10</b>
6.1. Conclusiones del EDA . . . . .	10

6.2. Trabajo Futuro . . . . .	11
<b>7. Estructura de la Entrega</b>	<b>12</b>
<b>8. Bibliografía</b>	<b>12</b>

## 1. Introducción

### 1.1. Contexto y Motivación

Esta práctica consiste en una competición Kaggle de clasificación binaria de hojas de tomate. El objetivo es distinguir entre hojas sanas (*control*) y hojas infectadas por el hongo *Botrytis cinerea* (*botrytis*) utilizando datos de fluorescencia multicolor e imágenes hiperespectrales.

La detección temprana de enfermedades en cultivos es fundamental para la agricultura de precisión y la gestión eficiente de recursos. Las técnicas de aprendizaje automático permiten automatizar el diagnóstico a partir de mediciones no destructivas, reduciendo pérdidas y optimizando el uso de tratamientos fitosanitarios.

### 1.2. Descripción del Dataset

El dataset proporcionado contiene mediciones de hojas de tomate:

- **Conjunto de entrenamiento:** 336 muestras con etiquetas
- **Conjunto de test:** 143 muestras sin etiquetas
- **Total de variables:** 309 columnas

Tipo	Columnas	Descripción
Metadatos (NO USAR)	exp, dpi, leaf, spot	Información experimental
Fluorescencia	F440, F520, F680, F740	4 valores de fluorescencia
Hiperespectral	w388.13 a w1028.28	300 variables espetrales
Target	class	<b>control</b> (0) o <b>botrytis</b> (1)

Cuadro 1: Descripción de las variables del dataset.

### 1.3. Herramientas y Tecnologías

- **pandas y numpy:** manipulación de datos
- **scikit-learn:** modelos de clasificación, escalado y validación cruzada
- **matplotlib y seaborn:** visualización
- **Jupyter Notebook:** análisis exploratorio interactivo

#### 1.4. Métrica de Evaluación

La métrica utilizada en la competición es el **F1-score**, definido como:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Esta métrica es apropiada para problemas de clasificación binaria con posible desbalance de clases, ya que considera tanto los falsos positivos como los falsos negativos.

## 2. Análisis Exploratorio de Datos (EDA)

### 2.1. Distribución de Clases

El análisis de la distribución de clases revela un desbalance moderado:

Clase	Muestras	Porcentaje
botrytis	196	58.3 %
control	140	41.7 %

Cuadro 2: Distribución de clases en el conjunto de entrenamiento.

El ratio de desbalance es de 1.40:1, lo cual es moderado y no requiere técnicas agresivas de balanceo.

### 2.2. Variables de Fluorescencia

Las 4 variables de fluorescencia presentan las siguientes características:

Variable	Media	Std	Min	Max
F440	235.57	9.19	209.95	257.33
F520	431.88	31.66	327.26	517.98
F680	773.10	120.98	484.27	1270.81
F740	1403.55	227.24	795.12	2125.33

Cuadro 3: Estadísticas descriptivas de las variables de fluorescencia.

Se observan dos grupos de variables correlacionadas:

- **Grupo 1:** F440 y F520 (correlación positiva: 0.89)
- **Grupo 2:** F680 y F740 (correlación positiva: 0.96)

Existe correlación negativa entre ambos grupos, sugiriendo características complementarias.

### 2.3. Variables Espectrales

El dataset incluye 300 variables espectrales en el rango de 388.13 nm a 1028.28 nm. El análisis de espectros promedio por clase muestra diferencias sutiles pero consistentes entre hojas sanas e infectadas.

## 2.4. Análisis de Valores Faltantes y Outliers

- **Valores faltantes:** No se detectaron valores faltantes en ninguno de los conjuntos.
- **Outliers** (método IQR en fluorescencia):
  - F440: 4 outliers (1.2 %)
  - F520: 10 outliers (3.0 %)
  - F680: 6 outliers (1.8 %)
  - F740: 4 outliers (1.2 %)

## 2.5. Reducción de Dimensionalidad (PCA)

El análisis de componentes principales revela alta reducibilidad dimensional:

- **3 componentes** explican el 95 % de la varianza
- **7 componentes** explican el 99 % de la varianza

Esto indica que, a pesar de tener 304 features, la información relevante se concentra en pocas dimensiones.

### 3. Preprocesamiento de Datos

#### 3.1. Limpieza de Datos

Durante la carga de datos se detectó un problema de formato: algunos valores numéricicos contenían espacios (ej: '232 .25'). Se implementó una función de limpieza automática:

```
def clean_numeric_columns(df):
    for col in df.columns:
        if df[col].dtype == 'object' and col not in METADATA_COLS + [TARGET_COL]:
            df[col] = df[col].astype(str).str.replace(' ', '').astype(float)
    return df
```

#### 3.2. Preparación de Features

El pipeline de preprocesamiento incluye:

1. **Exclusión de metadatos:** Se eliminan las columnas `exp`, `dpi`, `leaf`, `spot`
2. **Codificación del target:** `control` → 0, `botrytis` → 1
3. **Escalado:** StandardScaler para normalizar las features

#### 3.3. Arquitectura del Código

El código se organiza en módulos reutilizables:

- `src/preprocessing.py`: Funciones de carga, limpieza y transformación
- `src/models.py`: Definición y evaluación de modelos
- `src/utils.py`: Utilidades para submissions y logging

## 4. Modelado y Experimentación

### 4.1. Metodología

Se utilizó validación cruzada estratificada con 5 folds para evaluar los modelos, asegurando que cada fold mantiene la proporción de clases del dataset original.

### 4.2. Experimento 01: Modelos Baseline

Se compararon 5 algoritmos de clasificación con parámetros por defecto:

Modelo	F1-Score (CV)	Desv. Std
Logistic Regression	<b>0.9388</b>	0.0254
SVM (RBF)	0.9326	–
Random Forest	0.9266	–
Gradient Boosting	0.9203	–
KNN (k=5)	0.8943	–

Cuadro 4: Comparación de modelos baseline con StandardScaler.

**Resultado:** Logistic Regression obtuvo el mejor F1-score en validación cruzada (0.9388), siendo seleccionado para la primera submission.

### 4.3. Configuración del Mejor Modelo

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# Preprocesamiento
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Modelo
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_scaled, y_train)

# Predicciones
predictions = model.predict(X_test_scaled)
```

## 5. Registro de Experimentos

La siguiente tabla contiene el registro obligatorio de todas las submissions realizadas a Kaggle, incluyendo fecha/hora, posición en el momento de la subida, scores de entrenamiento y test, preprocesado aplicado, algoritmo utilizado y configuración de parámetros.

Nº	Fecha/Hora	Pos.	F1 Train	F1 Kaggle	Preprocesado	Algoritmo	Parámetros
01	23/12/2025 20:30	1	0.8278	0.8478	StandardScaler	LogisticRegression	max_iter=1000, random_state=42

Cuadro 5: Registro obligatorio de submissions a Kaggle.

### 5.1. Detalle del Experimento 01

- **Fecha y hora de subida:** 23/12/2025 20:30
- **Posición en el momento:** 1 de 2
- **Score en entrenamiento (CV 5-fold):** 0.8278 ( $\pm 0.0232$ )
- **Score en Kaggle (test):** 0.8478
- **Preprocesado:**
  - Limpieza de valores numéricos con espacios
  - Exclusión de columnas de metadatos (exp, dpi, leaf, spot)
  - Normalización con StandardScaler
  - Sin reducción de dimensionalidad (304 features)
- **Algoritmo:** Logistic Regression
- **Parámetros:** `max_iter=1000, random_state=42`
- **Observaciones:** Modelo baseline. El score en Kaggle (0.8478) supera ligeramente al score de validación cruzada (0.8278), indicando buena generalización.

## 6. Conclusiones y Trabajo Futuro

### 6.1. Conclusiones del EDA

1. El dataset presenta un desbalance moderado (1.40:1) que no requiere técnicas especiales de balanceo.
2. Las variables de fluorescencia forman dos grupos correlacionados con información complementaria.
3. Alta reducibilidad dimensional: 3 componentes PCA capturan el 95 % de la varianza.
4. No hay valores faltantes y los outliers son escasos (~3 %).

## 6.2. Trabajo Futuro

- Experimentar con PCA (3-7 componentes) para reducir dimensionalidad
- Probar XGBoost y LightGBM
- Optimización de hiperparámetros con Optuna
- Técnicas de ensemble (Voting, Stacking)
- Selección de features con SelectKBest

## 7. Estructura de la Entrega

```
P3/
+-- data/                      # Datos originales
|   +-- train.csv
|   +-- test.csv
|   +-- sample_submission.csv
+-- notebooks/                 # Jupyter notebooks
|   +-- 01_EDA.ipynb            # Analisis exploratorio
+-- src/                       # Código fuente
|   +-- preprocessing.py       # Preprocesado
|   +-- models.py              # Modelos
|   +-- utils.py               # Utilidades
+-- scripts/                   # Scripts de experimentos
|   +-- exp_01_baseline.py     # Experimento baseline
+-- submissions/               # Archivos CSV para Kaggle
+-- docs/                      # Documentacion
|   +-- latex/                 # Fuentes LaTeX
+-- PLAN_TRABAJO.md            # Plan de trabajo
+-- REGISTRO_EXPERIMENTOS.md   # Registro de experimentos
+-- requirements.txt            # Dependencias
```

## 8. Bibliografía

- Pedregosa, F. et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR, 12, 2825–2830.
- Documentación oficial de scikit-learn: <https://scikit-learn.org/stable/>
- Documentación oficial de pandas: <https://pandas.pydata.org/>