UNR Universidad Nacional de Rosario

Trabajo práctico 3 Minería de Datos

Tecnicatura Universitaria en Inteligencia Artificial

Docentes

- Dr. Ing. Flavio E. Spetale
- Ing. Facundo Vasquez
- Ing. Dolores Biondo
- Ing. Sofía Errecarte

Alumnos

- Peroni Antonio
- Raffaeli Taiel

FCEIA
FACULTAD DE CIENCIAS EXACTAS INGENIERIA Y AGRIMENSURA

Año 2023

Consignas

Objetivo

El objetivo de este trabajo práctico es integrar los conocimientos adquiridos en las unidades 5 y 6 en un problema real asociado a la determinación del color de los granos de café mediante la medición de atributos característicos.

Actividades

1. Descargar el conjunto de CoffeeRatings.csv1, para realizar el trabajo práctico. Analizar los atributos del conjunto de datos (distribuciones, valores, outliers, tipos de datos, etc.)

2. Realizar la predicción del atributo Color utilizando máquinas de vectores con kernel lineal analizando el parámetro costo. Mostrar los resultados sobre los conjuntos de test (Precisión, Exhaustividad y Exactitud) utilizando validación cruzada con k =5.

3. Realizar la predicción del atributo Color utilizando máquinas de vectores con kernel gaussiano analizando los parámetros costo y gama. Mostrar los resultados sobre los conjuntos de test (Precisión, Exhaustividad y Exactitud) utilizando validación cruzada con k =5.

4. Realizar la predicción del atributo Color utilizando Random Forest analizando los parámetros cantidad de estimadores y la máxima profundidad de los árboles. Mostrar los resultados sobre los conjuntos de test (Precisión, Exhaustividad y Exactitud) utilizando validación cruzada con k =5.

Resolución de actividades

Dataset

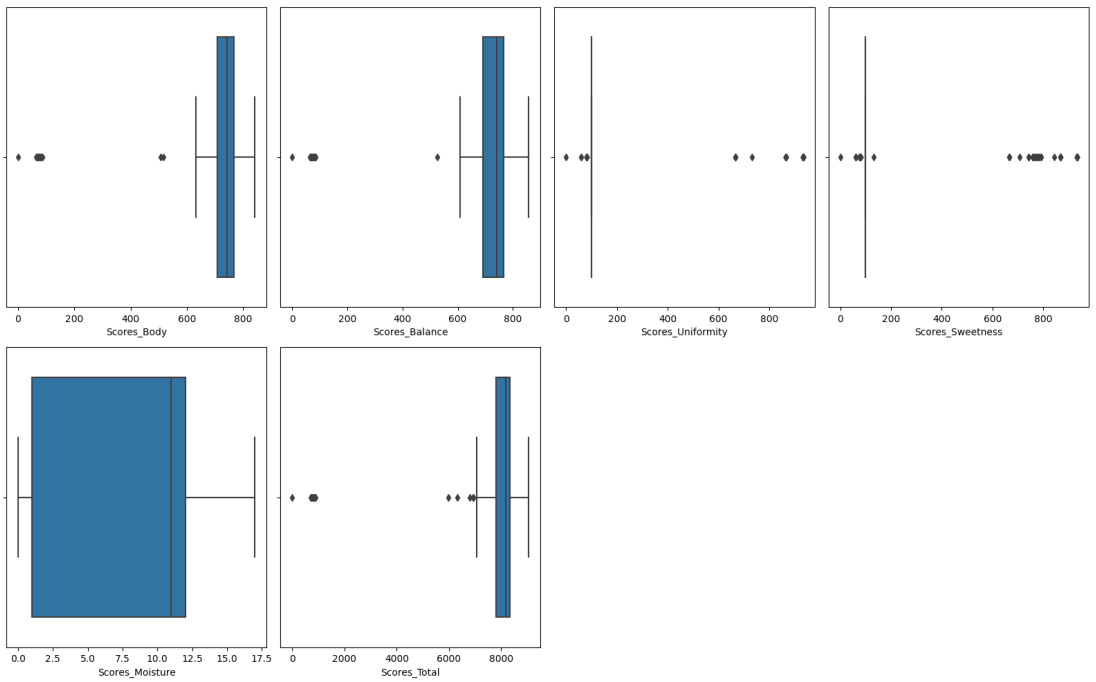
El Dataset proporcionado por el equipo docente es un dataset que contiene información sobre la determinación del color de los granos de café mediante la medición de atributos característicos.

Se realizó un breve análisis descriptivo para visualizar la distribución de datos, outliers y densidad de las clases. Al no tener datos faltantes y estar completo no se requirió hacer un análisis profundo.

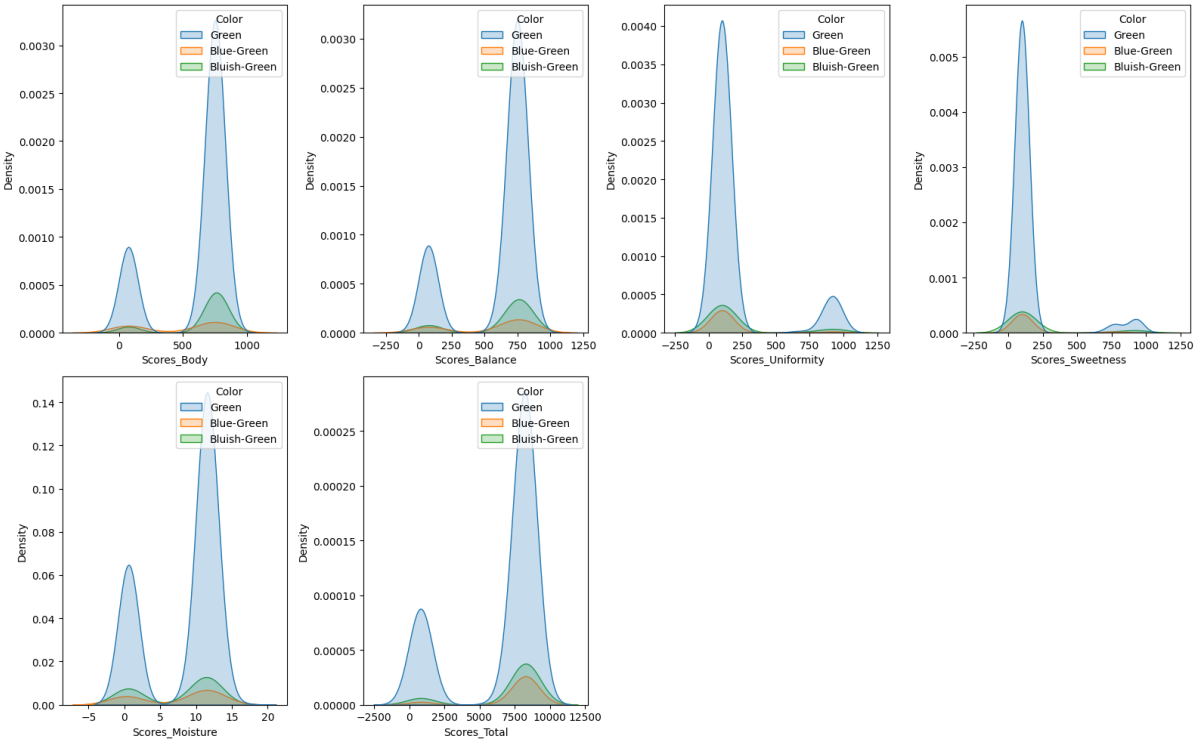
Conclusión de gráficos y datasets
La mayoría de los datos tienen distribuciones binomiales y outliers de valores grandes dentro del conjunto. La densidad de los datos en su distribución de clases está desbalanceada considerablemente.

Gráficas del dataset

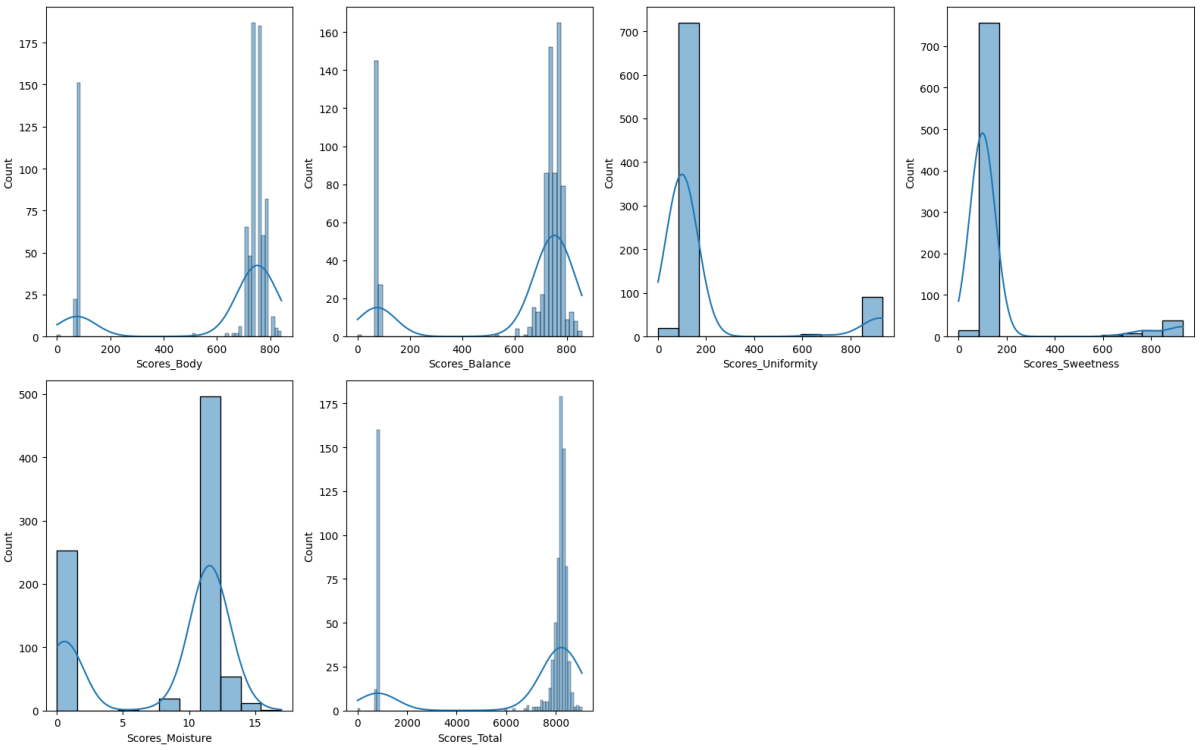
Boxplot



Densidades de clases



Distribuciones



SVM

Descripción de la técnica

Las Máquinas de Vectores de Soporte son un conjunto de algoritmos de aprendizaje supervisado utilizados para clasificación y regresión. La idea principal detrás de SVM es encontrar un hiperplano óptimo que separe de manera efectiva las instancias de diferentes clases en un espacio de características.

La eficacia de SVM radica en su capacidad para manejar conjuntos de datos complejos y no lineales, así como en su capacidad para generalizar bien a partir de un conjunto de datos de entrenamiento limitado.

Funcionamiento

Espacio de características:

1. Cada instancia en el conjunto de datos se representa como un vector en un espacio de características multidimensionales. Estas características son las variables que se utilizan para describir cada instancia.

Selección del hiperplano:

2. SVM busca el hiperplano de decisión que maximiza la separación entre las instancias de diferentes clases. Este hiperplano es el que mejor divide el espacio de características en dos regiones, una para cada clase.

Vectores de soporte:

3. Los vectores de soporte son los puntos de datos que están más cerca del hiperplano de decisión y tienen un papel crucial en la determinación del hiperplano óptimo. Estos puntos influyen en la posición y orientación del hiperplano y, por lo tanto, en la capacidad de generalización del modelo.

Margen:

4. El margen es la distancia entre el hiperplano y los vectores de soporte más cercanos. SVM busca maximizar este margen, lo que ayuda a mejorar la capacidad de generalización del modelo.

Función de pérdida:

5. En el caso de SVM para clasificación, se utiliza una función de pérdida que penaliza los errores de clasificación. El objetivo es encontrar el hiperplano que minimiza esta función de pérdida.

Kernel Trick:

6. SVM puede manejar eficientemente conjuntos de datos no lineales mediante el uso del "Kernel Trick". Esto implica mapear los datos originales a un espacio de características de mayor dimensión, donde los datos pueden ser separados de manera lineal.

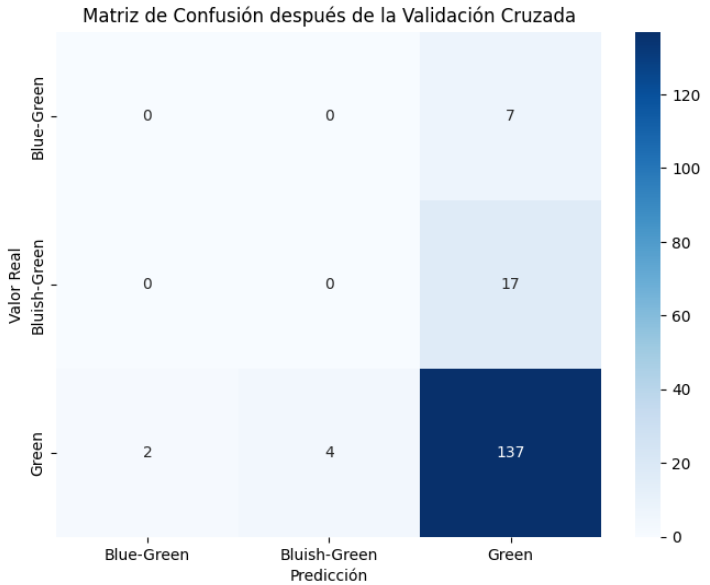
Aplicación SVM kernel lineal

Luego de buscar los mejores hiperparametros en el modelo de SVM con kernel lineal se obtuvo un modelo eficiente encontrando la clase mayoritaria sin embargo clasificar el resto de clases le era muy difícil.

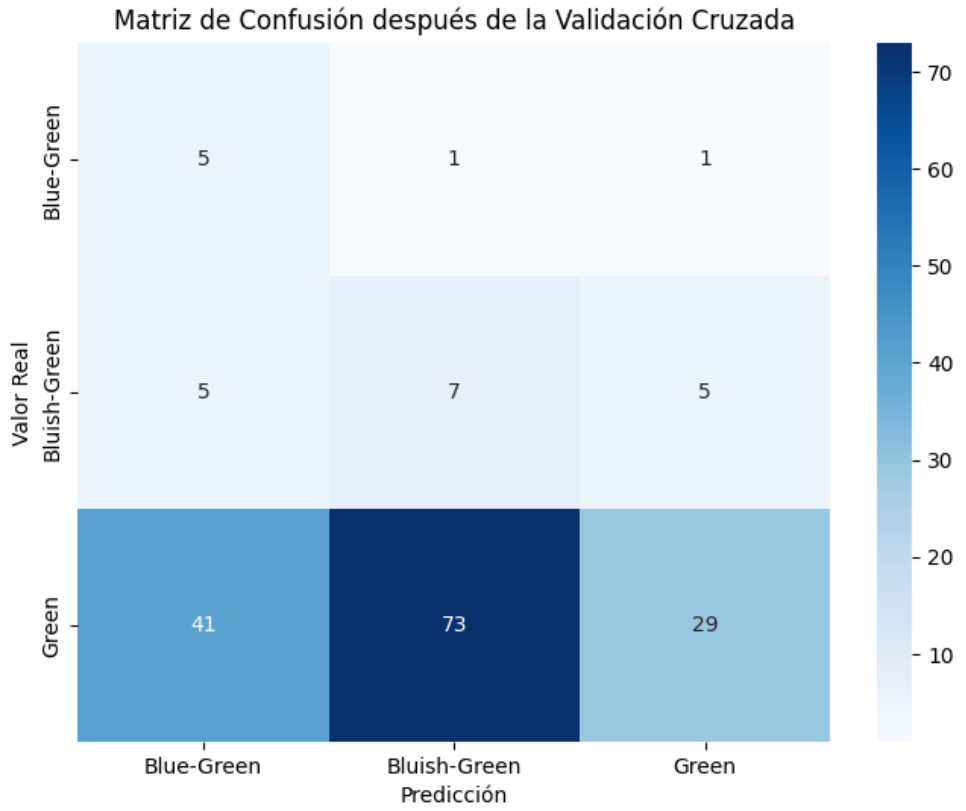
Informe de Clasificación después de la Validación Cruzada:

	precision	recall	f1-score	support
Blue-Green	0.00	0.00	0.00	7
Bluish-Green	0.00	0.00	0.00	17
Green	0.85	0.96	0.90	143
accuracy			0.82	167
macro avg	0.28	0.32	0.30	167
weighted avg	0.73	0.82	0.77	167

Claramente se deben balancear las clases de alguna forma para mejorar el rendimiento del modelo.



Luego de realizar una técnica de balanceo de la biblioteca scikit-learn llamada compute_class_weight y cambiando la métrica a optimizar (a recall_macro) se obtuvo un modelo que era muchísimo más eficiente reconociendo el resto de clases.



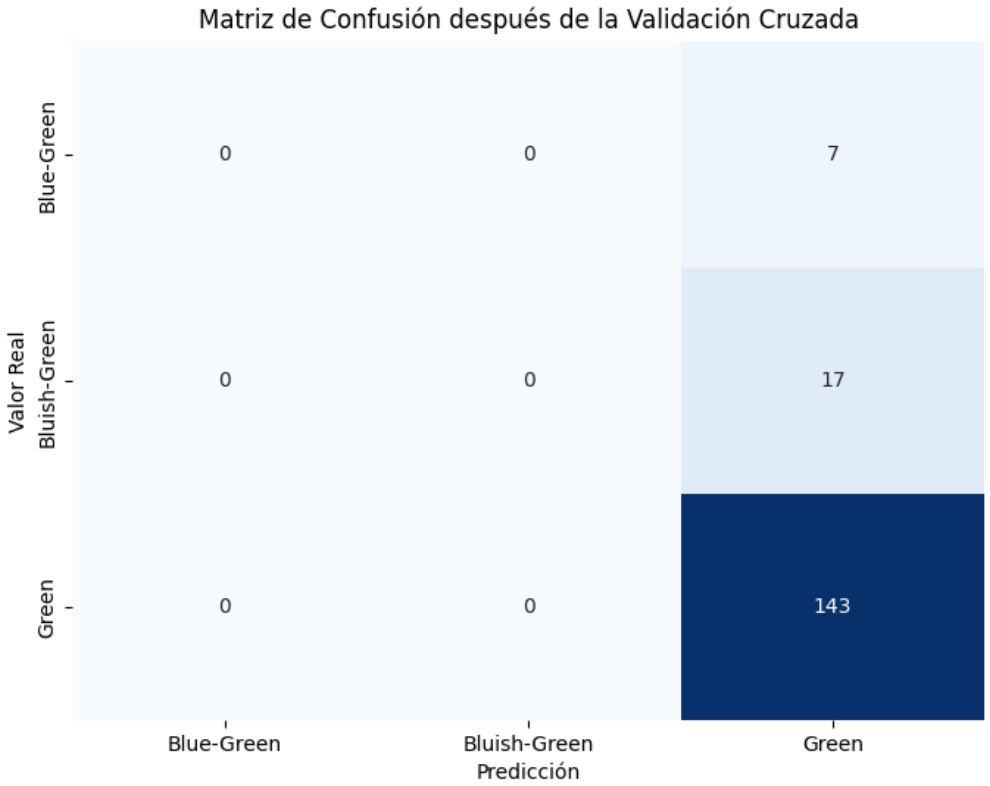
Aplicación SVM kernel gaussiano

En este caso el kernel gaussiano no funcionó como se esperaba, buscando distintos hiperparametros y probando distintas métricas a optimizar los resultados fueron los mismos incluso haciendo un balance de clases como en el kernel lineal.

Informe de Clasificación después de la Validación Cruzada:

	precision	recall	f1-score	support
Blue-Green	0.00	0.00	0.00	7
Bluish-Green	0.00	0.00	0.00	17
Green	0.86	1.00	0.92	143
accuracy			0.86	167
macro avg	0.29	0.33	0.31	167
weighted avg	0.73	0.86	0.79	167

El SVM con kernel gaussiano no logra reconocer las demás clases, solo reconoce la más mayoritaria. Se esperaban mejores resultados respecto al anterior kernel.



Random forest

Random Forest es un algoritmo de aprendizaje supervisado que se utiliza tanto para tareas de clasificación como de regresión. Su capacidad para manejar automáticamente la selección de características y mitigar el sobreajuste lo convierte en un algoritmo popular en machine learning para realizar predicciones más robustas y precisas.

Funcionamiento

Ensamble de árboles de decisión:

1. Random Forest es un tipo de modelo de ensamble, lo que significa que combina múltiples modelos más simples para formar un modelo más robusto y preciso. En el caso de Random Forest, estos modelos más simples son árboles de decisión.

Construcción de árboles de decisión:

2. Se construye un conjunto de árboles de decisión durante el entrenamiento. Cada árbol se construye de manera independiente utilizando un subconjunto aleatorio de características y un subconjunto aleatorio de datos de entrenamiento. Este proceso de selección aleatoria ayuda a diversificar los árboles y evita que el modelo se ajuste demasiado a los detalles específicos del conjunto de datos de entrenamiento.

Votación o promedio:

3. Para la clasificación, cada árbol en el bosque emite una predicción y la clase que obtiene la mayoría de votos se elige como la predicción final del Random Forest. Para la regresión, se promedian las predicciones de todos los árboles para obtener la predicción final.

Bootstrap Aggregating (Bagging):

4. Durante la construcción de cada árbol, se utiliza una técnica llamada Bootstrap Aggregating o Bagging. Esto implica muestrear aleatoriamente con reemplazo del conjunto de datos de entrenamiento. Como resultado, algunos datos pueden estar presentes en el conjunto de entrenamiento de un árbol múltiples veces, mientras que otros pueden no estar presentes en absoluto.

Selección aleatoria de características:

5. En cada nodo de decisión de un árbol, solo se considera un subconjunto aleatorio de características para realizar la división. Esta selección aleatoria ayuda a garantizar que los árboles sean decorrelacionados y mejora la generalización del modelo.

Manejo de overfitting:

6. La diversidad introducida por la construcción de múltiples árboles y la selección aleatoria de características ayuda a mitigar el sobreajuste (overfitting), lo que significa que el modelo es menos propenso a adaptarse demasiado a los detalles específicos del conjunto de datos de entrenamiento y tiene una mejor capacidad de generalización a nuevos datos.

Aplicación de random forest

Se esperaba que el random forest funcionara mejor que SVM, primero se probó el modelo sin balance de clases, lo cual claramente funcionó mal no logrando diferenciar ninguna otra clase que no sea la mayoritaria.

Luego se probó con balance de clases y la métrica recall_macro para optimizar, obteniendo el mejor modelo de todos. El modelo logra reconocer las clases minoritarias de forma eficiente a pesar de tener pocos datos.

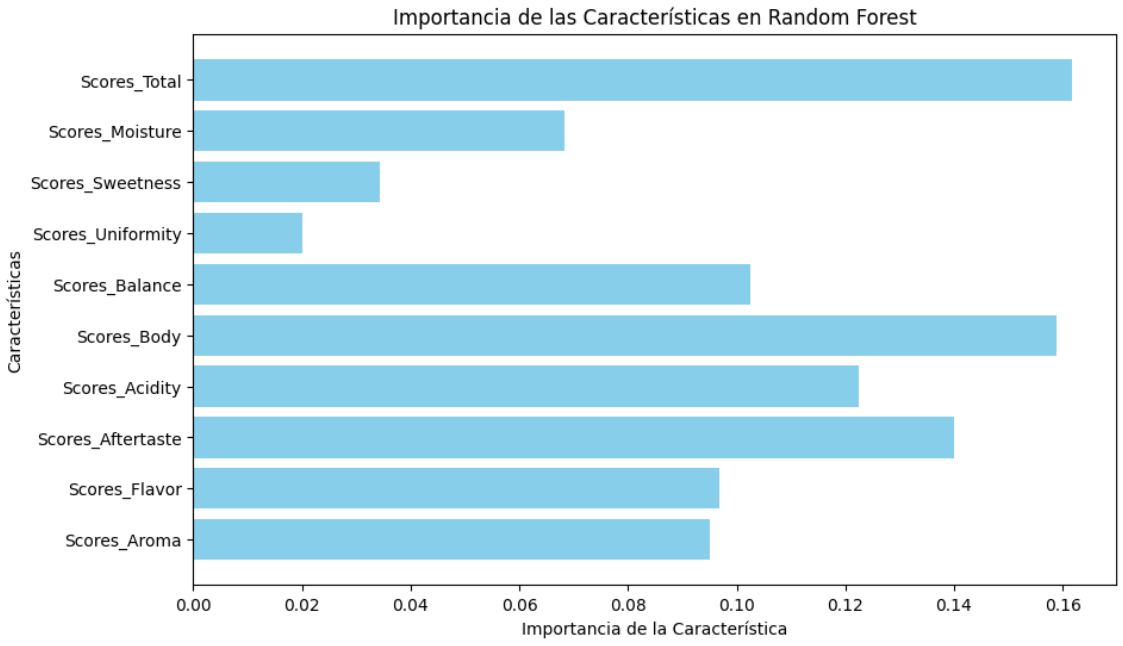
Mejores hiperparámetros:

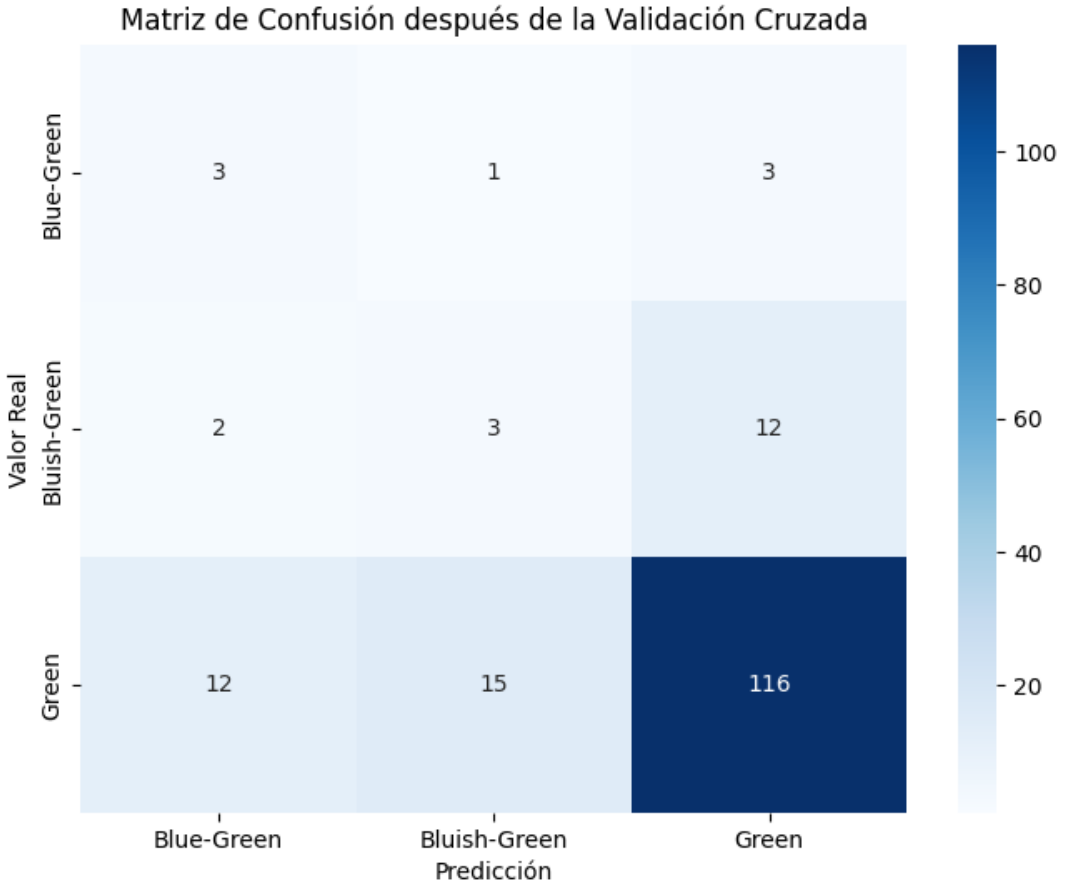
```
{'max_depth': 13, 'n_estimators': 50, 'random_state': 18}
```

Informe de Clasificación después de la Validación Cruzada:

	precision	recall	f1-score	support
Blue-Green	0.18	0.43	0.25	7
Bluish-Green	0.16	0.18	0.17	17
Green	0.89	0.81	0.85	143
accuracy			0.73	167
macro avg	0.41	0.47	0.42	167
weighted avg	0.78	0.73	0.75	167

La característica más importante en la clasificación de clases fue Scores_Total.




Matriz de Confusión después de la Validación Cruzada
Valor Real
Blue-Green
Bluish-Green
Green
3
1
3
2
3
12
12
15
116
100
80
60
40
20
Blue-Green
Bluish-Green
Green
Predicción

Conclusiones

Dataset

Los dataset brindados por la cátedra están completos pero tienen clases desbalanceadas, lo cual dificulta la adaptación correcta de los modelos para clasificar los datos, se realizó una técnica de balanceo de clases y además, se buscaron las mejores métricas a optimizar para contrarrestar esta dificultad.

SVM

Luego de aplicar SVM con dos kernels distintos (con y sin balance de clases), se obtuvieron mejores resultados con el kernel lineal y con el kernel gaussiano no se obtuvo ninguna diferencia con o sin balanceo de clases. Quizás con una técnica de balanceo más robusta o avanzada se logre mejorar los resultados.

Teniendo dataset completamente desbalanceado y con pocos datos, provoca que para los modelos de SVM se les dificulta demasiado el lograr encontrar la división correcta de clases.

Se deben explorar soluciones de balanceo de datos, también buscar otros hiperparametros que mejoren la eficacia de SVM a la hora de clasificar.

El kernel lineal a pesar de funcionar “bien” por reconocer al menos algunos ejemplos de las clases minoritarias clasificaba muy mal la clase mayoritaria, haciendo que la mayoría de los ejemplos sean clasificados incorrectamente. Quizás esto se deba a los pesos asignados a las clases.

Random forest

Se esperaba que random forest obtuviera los mejores resultados. En la primera prueba sin balance de clases, funciona igual que SVM (sin balance) obteniendo un modelo que no logra diferenciar las clases minoritarias de la mayoritaria.

En la segunda prueba con balance de clases y optimizando la métrica recall_macro se obtuvo el modelo más eficiente de todos los testeados. Logrando reconocer varios ejemplos de las clases minoritarias, clasificándolos de forma correcta y sin dejar de clasificar bien la clase mayoritaria, esto es importante ya que se obtuvo ganancia de información sin perderla por otro lado.

A pesar de los pocos datos y el desbalance el modelo funciona bien teniendo un accuracy total de 73%, quizás mejorando otros hiperparametros y con técnicas de

balanceo de clases más robustas se podría incrementar las clasificaciones correctas considerablemente.