

# Detección de estado de cultivos mediante remote sensing para mejora del despliegue de medios en incendios forestales



Alejandro Sáez Subero  
Data Scientist - Corporación Nacional Forestal



## Contenido

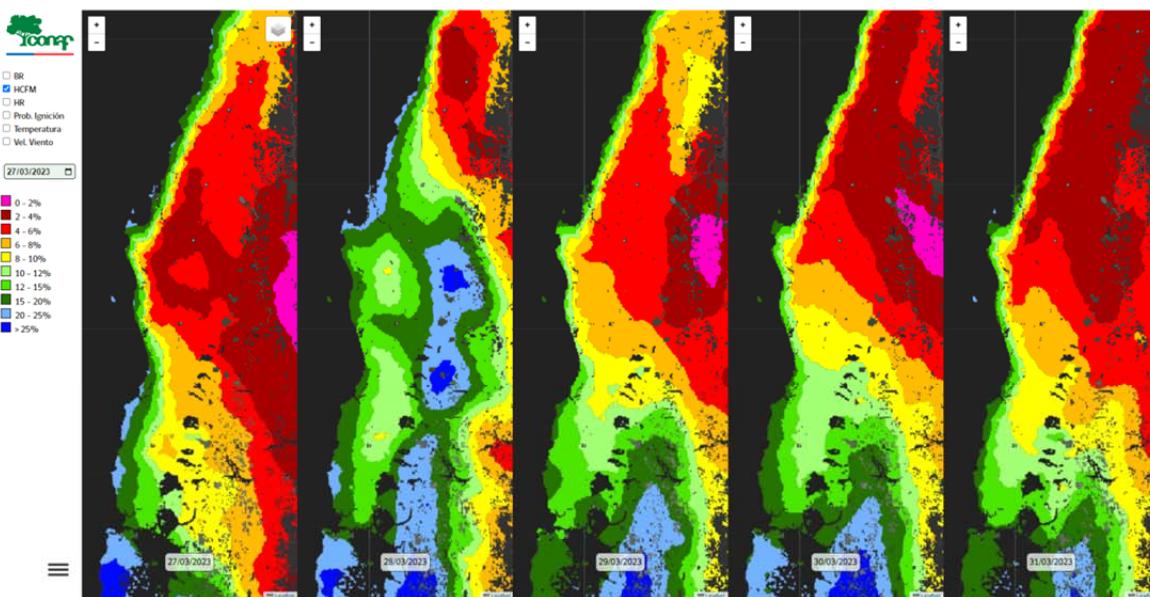
Introducción – Product Vision.....	2
Definición del producto .....	3
Product Backlog .....	3
Definición de los Sprints .....	4
Sprint 1 - Búsqueda y definición de datasets.....	5
Búsqueda de datasets.....	5
Definición de opciones.....	7
Sprint 2 - Análisis y preparación de los datos para el proceso de modelado .....	7
Validación de datos.....	7
Análisis exploratorio EDA .....	8
Pre-procesado .....	12
Sprint 3 - Modelado y evaluación .....	14
Modelo multiclasificación .....	14
Modelo binario de clasificación .....	15
Modelo binario de segmentación.....	18
Conclusiones primera fase de modelado.....	19
Fase de diferenciación entre cultivos verdes y secos .....	20
Sprint 4 - Construcción maqueta visualización.....	22
Sprint 5 - Deploying maqueta Google Cloud .....	23
Sprint 6 - Generación informes y documentación .....	23
Dificultades y desafíos.....	24
Tareas pendientes y posibilidades de mejora .....	25
Conclusiones .....	26

## Introducción – Product Vision

En el Departamento de Investigación y Desarrollo de la Corporación Nacional Forestal de Chile, lugar donde trabajo desde hace un año, y, en particular en la Sección de Estudios y Proyectos, que se encarga principalmente del análisis y gestión de los incendios forestales en el país, se detecta la necesidad de un sistema que sea capaz de diferenciar cultivos verdes de cultivos secos mediante imágenes por satélite.

Esta necesidad, se fundamenta en la velocidad de respuesta y despliegue de forma precisa de medios de combate de incendios, tanto terrestres como aéreos, así como la optimización de recursos en dichos despliegues, durante la temporada de incendios, que en el hemisferio sur comprende los meses de octubre a abril.

La metodología de trabajo que se tiene en nuestro departamento, consiste en general en la utilización de herramientas ya existentes de análisis e imagen de satélite, adaptándose estas, no siempre en las mejores condiciones a las necesidades propias. Se utilizan plataformas como Google Earth Engine, ArcGis o Carto. Algunas de estas herramientas ofrecen modelos de Remote Sensing, pero siempre con las limitaciones de, o bien versiones gratuitas debido a que no se tienen los recursos necesarios o bien, no completamente adaptadas a las necesidades locales.



**Ilustración 1.** Ejemplo de máscara generada en ArcGis y utilizada sobre mapa de Chile que muestra % de Humedad del Combustible Fino Muerto.

En este caso, dichos modelos de detección, son capaces de detectar y diferenciar de forma eficiente en la mayoría de los casos zonas urbanizadas, bosques, cultivos, extensiones de agua, pastizales y otros, pero no existe una herramienta que diferencie cultivos secos o ya recolectados de cultivos verdes.

Este punto es muy importante en el despliegue de medios para el combate de incendios, ya que un campo seco (como por ejemplo de trigo o cebada, amarillo) antes de su recolección, o incluso después, es extremadamente inflamable y extiende rápidamente las llamas, mientras que un campo verde (por ejemplo un maizal), no extenderá las llamas, o, al menos no con la misma intensidad y velocidad.

## Definición del producto

El objetivo es crear un modelo propio, que sea capaz de diferenciar los cultivos en función de si están secos o verdes.

A partir de esta idea central, surgen diferentes posibilidades para una tarea que ya se define como de clasificación. Una opción es la creación de un modelo multiclas, que, al igual que los ya existentes, deberá identificar los distintos tipos de suelo, y además, en una segunda fase, diferenciar qué cultivos están verdes y cuales secos. Otra opción, será la de crear un modelo binario, que sepa en una primera fase diferenciar lo que es cultivo de aquello que no lo es y en una segunda, de todo aquello que sea cultivo, diferenciar si está seco o verde, aplicándolo como una nueva máscara en la visualización en el mapa, por encima de la máscara ya existente.

Se opta por la realización de un modelo binario, con la idea de centrarse en la calidad de la identificación de cultivos, aunque no se descarta realizar pruebas de modelado multiclas.

En ambos casos se deberá entrenar en una primera fase un modelo que diferencie los tipos de suelo y, en una segunda etapa se deberá diferenciar aquellos cultivos secos de los verdes.

Se debe dejar claro, que, debido a la limitación de los tiempos para la primera iteración del proyecto, el objetivo es entregar un prototipo funcional y que, tras ello, quedará mucho trabajo por delante de mejora de los modelos y desarrollo del producto en general.

## Product Backlog

- Búsqueda y definición del dataset
- Validación de datos
- Análisis exploratorio
- Pre procesamiento
- Modelado
- Evaluación de los modelos
- Construcción maqueta para visualización
- Deploying maqueta en Google Cloud

- Generación de informes de resultados
- Evaluación e implementaciones iterativas del modelo para la mejora en base a resultados obtenidos en la maqueta

## Definición de los Sprints

La duración de la primera iteración del proyecto será de 7 semanas, que se distribuyen como se muestra en las siguientes imágenes:

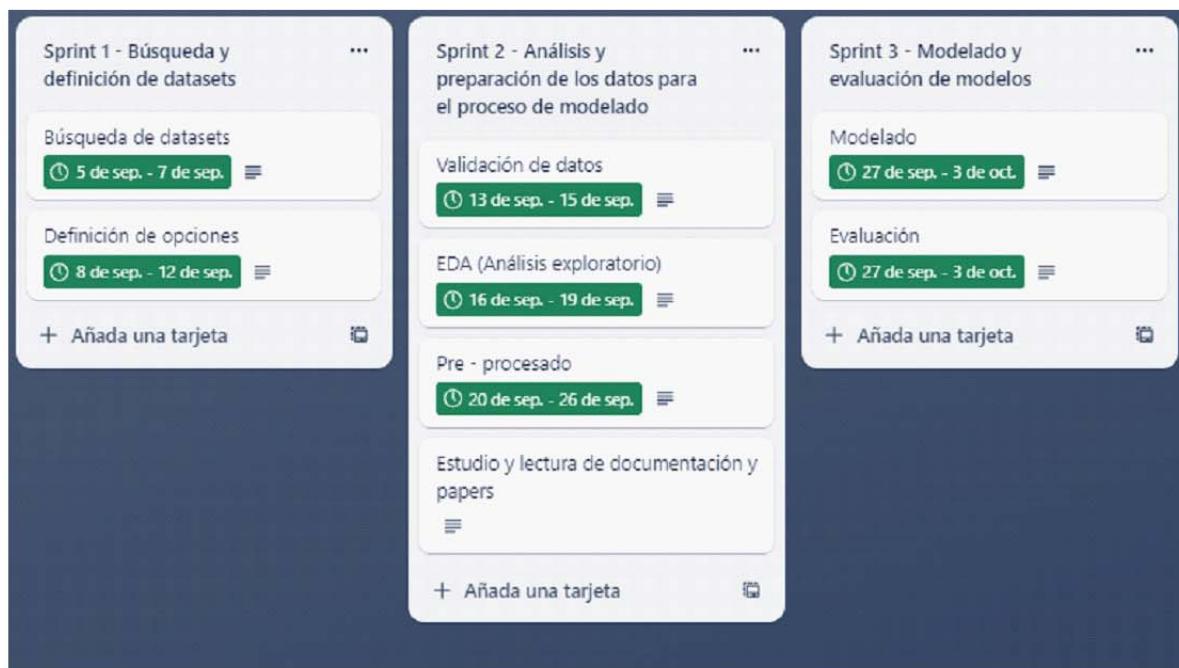


Ilustración 2. Distribución de las fases del proyecto I.

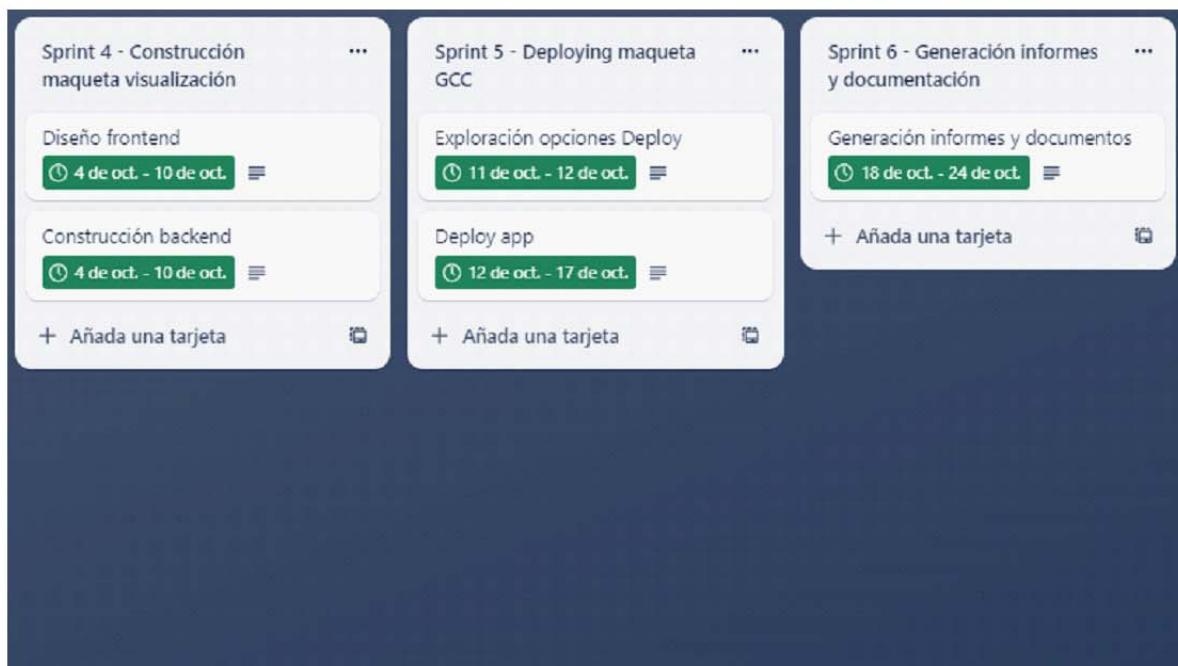


Ilustración 3. Distribución de las fases del proyecto II.

## Sprint 1 - Búsqueda y definición de datasets

### Búsqueda de datasets

La duración de esta fase será de una semana. Durante el proceso de búsqueda de datasets, hay abiertas varias opciones que deben ir concretándose durante este sprint. La primera de ellas es, el tipo de modelo que deseamos diseñar, si será una clasificación de imágenes o, si será segmentación (Image segmentation).

Esta segunda posibilidad se ajusta mejor al propósito y a la necesidad del proyecto, ya que es importante que las máscaras que se generen sobre los mapas sean finas y detalladas, sin embargo, con las limitaciones tanto de tiempo como de recursos, es complejo entrenar un modelo con imágenes a nivel de pixel, ya que la cantidad de recursos necesarios sería muy grande.

Otra opción a definir, será el formato de los datos. Por lo general, los datasets se encuentran en formato jpg y png, pero, al tratarse de imágenes de satélite, estos toman dichas imágenes en diferentes canales. Los satélites Sentinel europeos por ejemplo, toman imágenes en 12 bandas diferentes, llamadas bandas espectrales (por poner algunos ejemplos, B3: longitud de onda 560nm, corresponde a la banda verde; B5: longitud de onda 705nm, pertenece al espectro visible e infrarrojo cercano, VNIR; B9: longitud de onda 940nm, pertenece al espectro de onda corta

infrarroja SWIR, etc.) y cada imagen se compone de esos 12 canales en lugar de los 3 habituales RGB.

Banda	Resolución Espacial	Longitud de onda central	Descripción
B1	60 m	443 nm	Ultra azul (Costa y Aerosol)
B2	10 m	490 nm	Azul
B3	10 m	560 nm	Verde
B4	10 m	665 nm	Rojo
B5	20 m	705 nm	Visible e Infrarrojo Cercano (VNIR)
B6	20 m	740 nm	Visible e Infrarrojo Cercano (VNIR)
B7	20 m	783 nm	Visible e Infrarrojo Cercano (VNIR)
B8	10 m	842 nm	Visible e Infrarrojo Cercano (VNIR)
B8a	20 m	865 nm	Rojo de borde (RedEdge)
B9	60 m	940 nm	Vapor de Agua
B10	60 m	1375 nm	Cirrus
B11	20 m	1610 nm	Onda Corta Infrarroja (SWIR)
B12	20 m	2190 nm	Onda Corta Infrarroja (SWIR)

Ilustración 4. Bandas satélites Sentinel-2.

Entre el material seleccionado, existen tres fuentes de datos que se pueden adecuar a las necesidades del proyecto:

- Sentinelhub (<https://sentinelhub-py.readthedocs.io/en/latest/index.html>): En este caso, se trata de una API propia de Sentinel, con un tiempo limitado de uso gratuito, que permite entre otras funcionalidades obtener imágenes.
- EuroSAT dataset (<https://github.com/phelber/eurosat>): Este dataset, contiene 27.000 imágenes duplicadas, en formato jpg por un lado y también en formato multiespectral con 13 bandas. Cada imagen está clasificada entre un total de 10 clases.
- DeepGlobe dataset (<https://www.kaggle.com/datasets/balrajg8/deepglobe-land-cover-classification-dataset>): Se trata de un dataset pequeño, de 803 imágenes etiquetadas más otras 700 sin etiquetar. Contiene las imágenes en jpg y sus respectivas máscaras en png. La clasificación en este caso es mediante máscaras y a nivel de píxel (segmentación).

### Definición de opciones

Con las opciones indicadas arriba, se decide que para el propósito actual es más conveniente utilizar imágenes en formato RGB por simplicidad, aunque queda pendiente para una próxima iteración el uso de imágenes multibanda, las cuales pueden ser combinadas para utilizar aquellas que sean más indicadas por ejemplo para la detección de humedad o de áreas verdes, y nos puede dar una mayor precisión en un futuro modelo.

Por otro lado, es interesante poder contar con un set de imágenes que dé la posibilidad de trabajar a nivel de pixel y, aunque DeepGlobe es pequeño y solo la mitad de las imágenes está etiquetada, cada una de ellas es gran tamaño (2048x2048), lo que nos abre la puerta a cierto nivel de innovación y nos da más posibilidades en siguientes iteraciones del producto.

Por estas razones, tras realizar un análisis y visualización previa tanto de muestras como de la documentación de las tres opciones destacadas, el dataset a utilizar será DeepGlobe ya que nos aporta simplicidad respecto del formato y mayores posibilidades para implementar iteraciones y evoluciones del producto inicial a futuro.

### Sprint 2 - Análisis y preparación de los datos para el proceso de modelado

La duración de esta fase será de dos semanas. Durante este Sprint, además de las tareas propias de la fase, se va a profundizar en documentación y papers para lograr una tarea de EDA y preprocesado adecuada, además de un buen modelado en la siguiente etapa.

### Validación de datos

En primer lugar, validaremos visualmente algunas imágenes, para conocer más en profundidad su definición, la precisión de sus máscaras etc.

Para ver el proceso de validación: [Validación\\_muestras\\_deepglobe.ipynb](#)

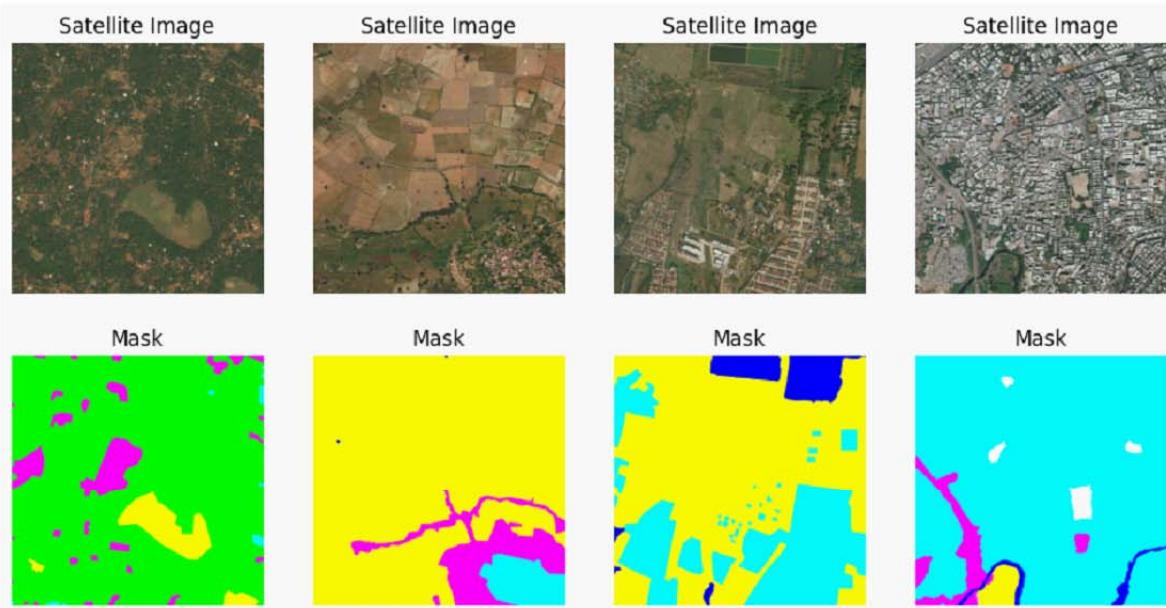


Ilustración 5. Ejemplo imágenes originales y su respectiva máscara. 2048x2048 px.

Durante este proceso se decide que se seguirán dos vías de trabajo, una principal en la que se tratará de desarrollar un modelo de clasificación a partir de estos datos, y otra secundaria, que se desarrollará en futuras iteraciones, en la que se tratará de desarrollar un modelo de segmentación y que requerirá de más recursos y tiempo como ya se comentó anteriormente. En ambos casos será necesario contar con más imágenes y dado que las que disponemos tienen un tamaño de 2048x2048, se generarán a partir de estas imágenes más pequeñas. En este caso trabajaremos con imágenes de 128x128, lo que nos permitirá que nuestros datos tengan un tamaño más manejable y una calidad suficiente para el entrenamiento y, además, solucionar el problema de escasez de muestras. De cada imagen original podremos obtener 256 nuevas imágenes con la nueva dimensión, por lo que ahora contaremos con casi 290.000 imágenes de 128x128 con sus respectivas máscaras, dejando a un lado de momento para este propósito las imágenes sin etiquetar.

Para ver el proceso de generación de tiles: [DeepGlobe\\_TilesGeneration.ipynb](#)

Una vez realizada la generación de los tiles, para seguir por la vía del modelo de clasificación, necesitamos que cada una de las imágenes tenga una sola label, y actualmente son multilabel, por lo que debemos realizar un proceso en el que cada imagen acabe teniendo una sola etiqueta. Para resolver este problema, utilizaremos k-Means para determinar el color dominante de la máscara de cada imagen y con ello, poder etiquetar cada imagen con su clase mayoritaria.

Proceso de etiquetado: [Labeling\\_images\\_DeepGlobe.ipynb](#)

### Análisis exploratorio EDA

El análisis exploratorio se realizará sobre el set de datos de 128x128, que es sobre el que se trabajará a partir de aquí.

Se observan varios aspectos importantes. El primero de ellos es que al hacer esta división de las

imágenes se ha producido un desbalanceo muy importante del número de muestras por clase, como se ilustra a continuación:

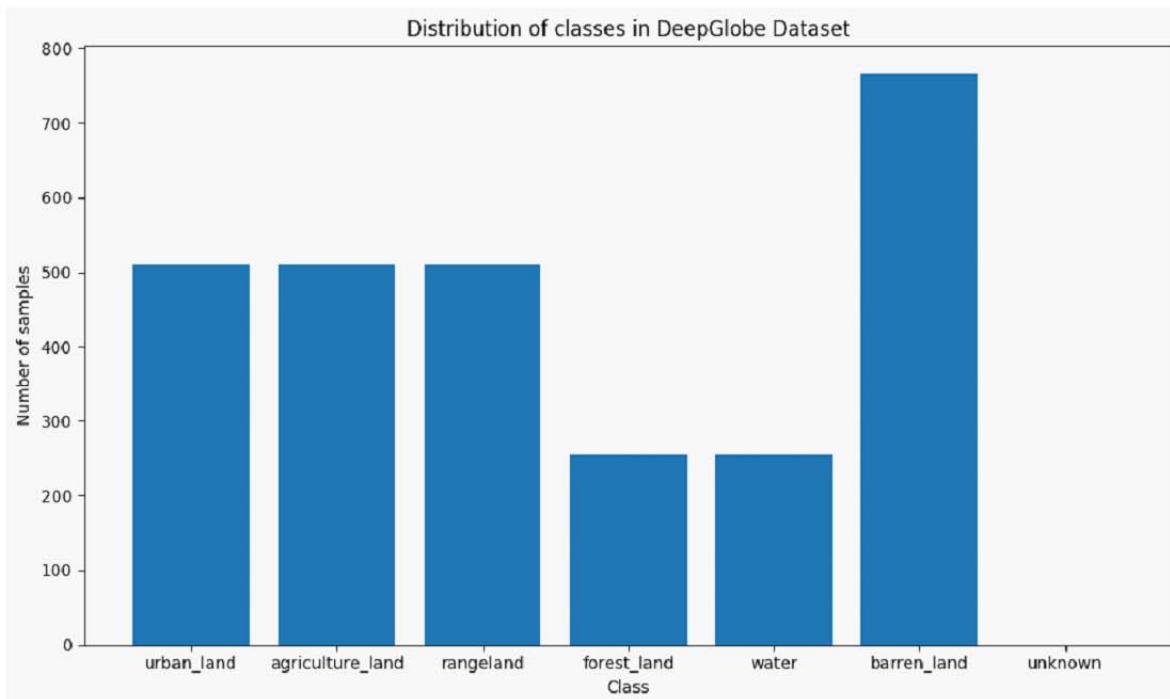


Ilustración 6. Cantidad de muestras por clase en dataset original de 2048x2048 (Tener en cuenta que cada muestra puede pertenecer a más de una clase).

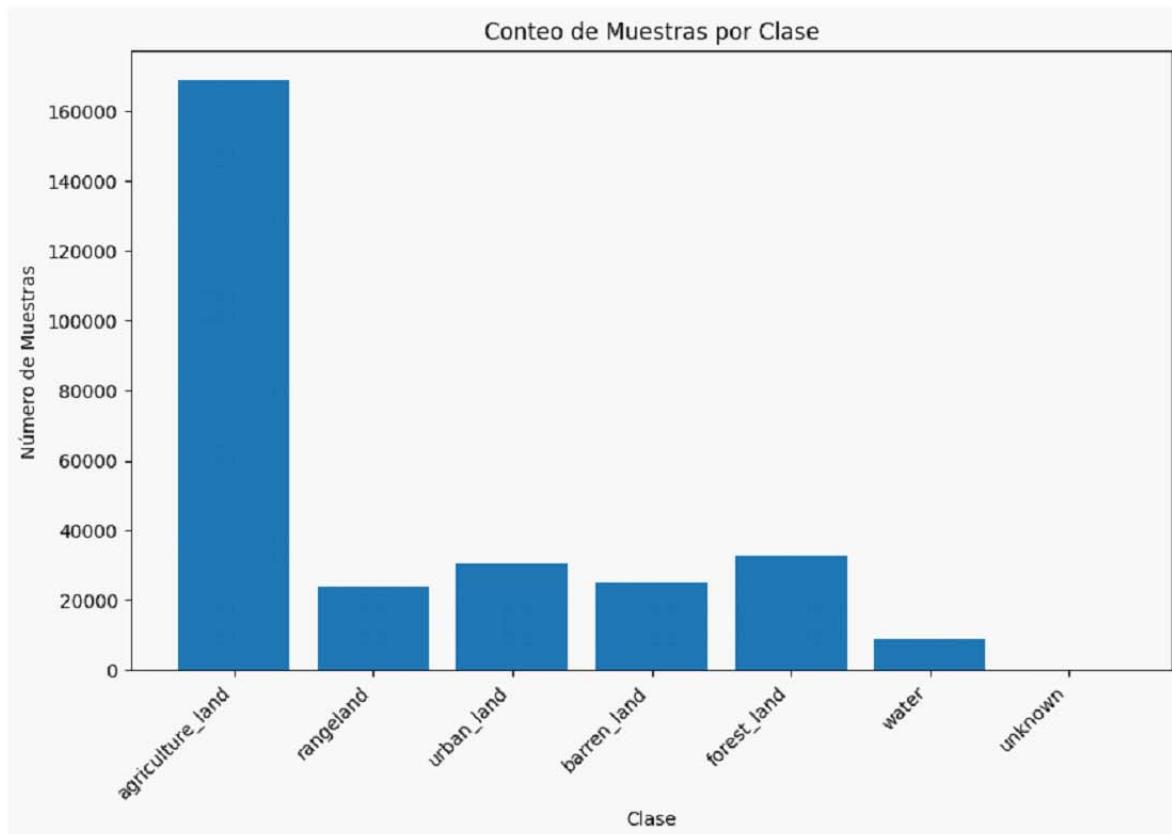


Ilustración 7. Desbalanceo tras dividir imágenes en 128x128 y asignar a cada una su clase mayoritaria (la clase 'unknown' tiene 153 muestras).

Se observa con diferentes muestras que el proceso de Split y etiquetado ha sido realizado de forma correcta:

*Detección de estado de cultivos mediante remote sensing para mejora del despliegue de medios en incendios forestales*

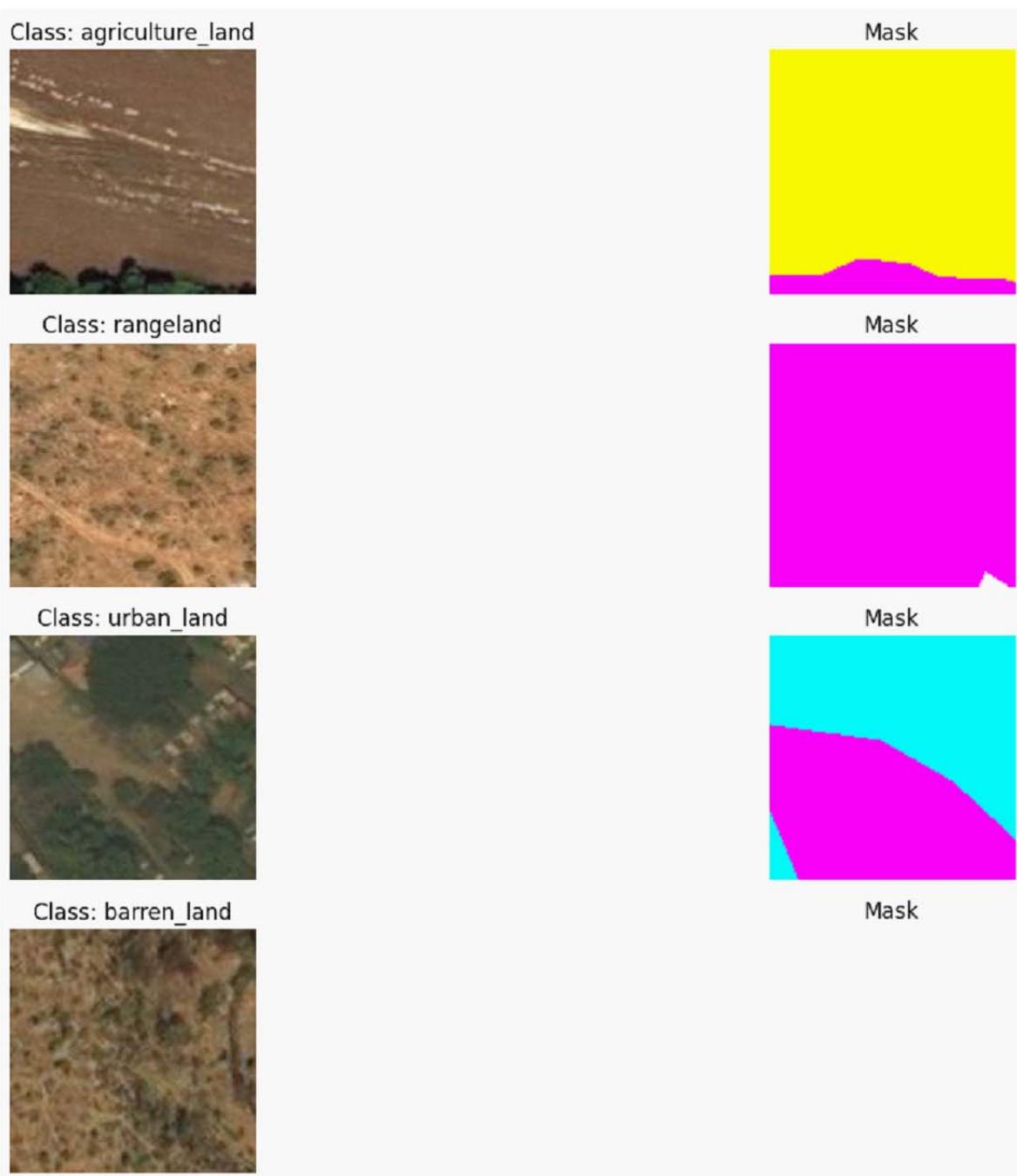


Ilustración 8. Nuevas muestras de 128x128 para cada clase I.

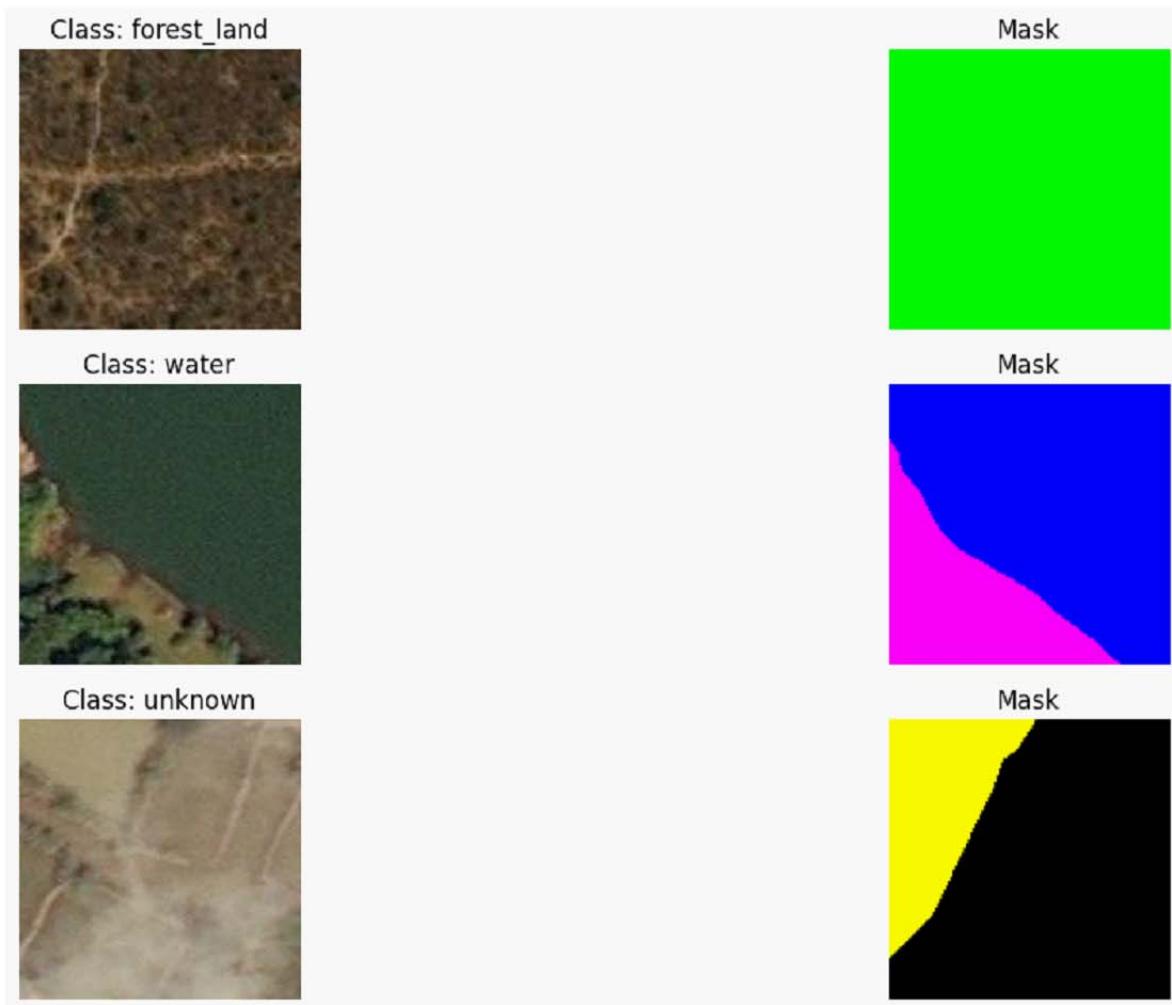


Ilustración 9. Nuevas muestras de 128x128 para cada clase II.

Para ver el proceso de EDA: [EDA\\_DeepGlobe\\_128.ipynb](#)

### Pre-procesado

Tras hacer diferentes pruebas, para el modelo de clasificación se decide trabajar con Transformers, dado que además de unos buenos resultados en entrenamiento, nos van a proporcionar velocidad en el mismo, lo cual, dado los escasos recursos con que se cuenta, es importante tener en cuenta.

Tras hacer algunas pruebas de carga de datos en memoria, y contando con el desbalanceo de clases para el modelo de clasificación, se toma la decisión de limitar el número de muestras en esta primera iteración del producto. Para ello, balancearemos el dataset poniendo como límite el número de imágenes de la clase más pequeña. Como se muestra en el notebook

EDA\_DeepGlobe\_128.ipynb, a priori parece que la clase 'unknown' podría omitirse, ya que apenas cuenta con 200 muestras entre casi 290.000, y que se deben a escenas nubladas, situación que se puede filtrar a la hora de ver imágenes de satélite. En cualquier caso, al modelar, probaremos qué sucede tanto incluyéndola en las muestras como descartando esos datos. Logramos de este modo cerca de 43.000 imágenes, de 128x128, con el mismo número de muestras por clase, elegidas de forma aleatoria, con las que se trabajará para este modelo y para el modelo binario.

Las imágenes son pre procesadas y normalizadas de la forma habitual, y guardadas en vectores, para poder recuperarlas y utilizarlas en cualquier momento.

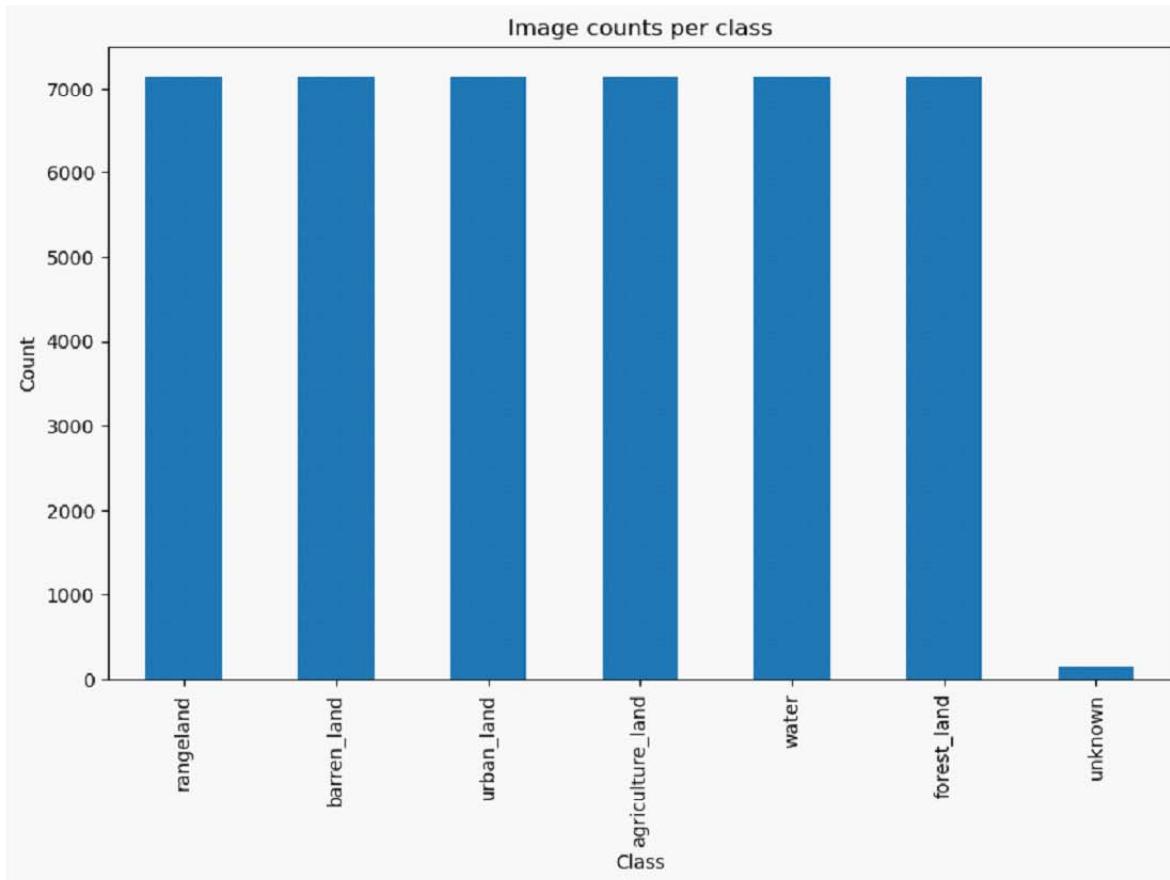


Ilustración 10. Resultado balanceo hacia la clase con menor número de muestras.

Pre-procesado completo: Preprocess\_Transformers\_DeepGlobe\_128.ipynb

## Sprint 3 - Modelado y evaluación

### Modelo multiclase de clasificación

Tras realizar pruebas con diferentes modelos, vamos a trabajar en particular con Vision Transformers (ViT) de Google ([An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#))

Utilizaremos esta arquitectura en la extracción de los embeddings, para después poder hacer un entrenamiento rápido, en principio con un clasificador multiclase sencillo que podremos mejorar en próximas implementaciones. Tras el entrenamiento, obtenemos un accuracy por debajo de 0.7 y analizando la precisión y recall, se puede observar que si bien hay clases que el modelo está aprendiendo a identificar con mucho éxito (F1-Score por encima del 80%), hay otras en las que no es capaz de obtener buenos resultados. Dado que tenemos mucho margen en cuanto a disponibilidad de datos para entrenamiento (para este modelo solo utilizamos 43.000 muestras de 290.000 disponibles), parece que puede haber mucho margen de mejora, por lo que pasamos a trabajar en el modelo binario.

	precision	recall	f1-score	support
0	0.91	0.31	0.46	1065
1	0.90	0.35	0.51	1044
2	0.94	0.61	0.74	1040
3	0.31	0.92	0.46	1093
4	1.00	0.59	0.74	27
5	0.93	0.71	0.80	1089
6	0.98	0.83	0.90	1093
accuracy			0.63	6451
macro avg	0.85	0.62	0.66	6451
weighted avg	0.83	0.63	0.65	6451

Ilustración 11. Resultados en test para el modelo multiclase de clasificación con imágenes de 128x128.

Modelado multiclase de clasificación: [Multiclass\\_Model\\_Transformers\\_DeepGlobe\\_128.ipynb](#)

### Modelo binario de clasificación

Para el modelo binario, será necesario convertir las etiquetas del dataset que no son 'agriculture\_land' a 'No\_agriculture'. También será necesario crear un nuevo subset, como en el caso anterior, limitando el número de muestras para esta primera iteración.

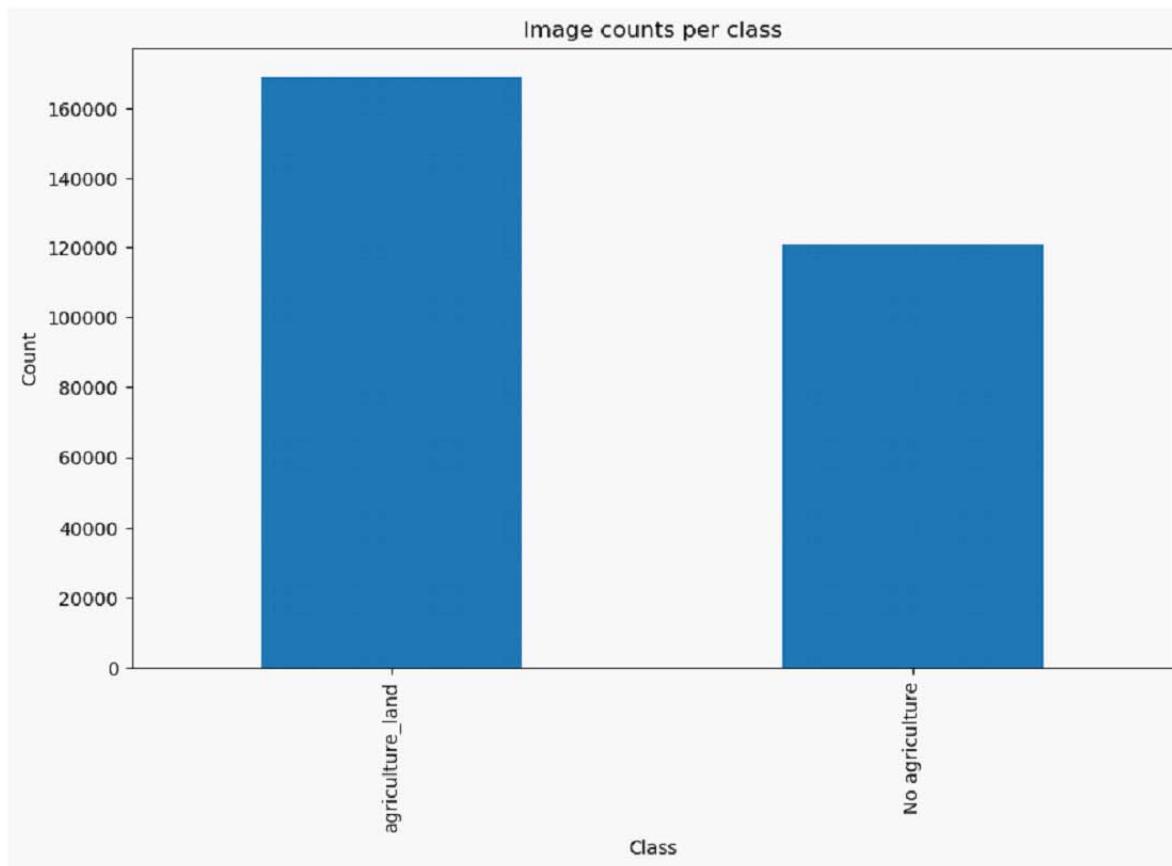


Ilustración 12. Desbalanceo original del dataset binario con todas las muestras.

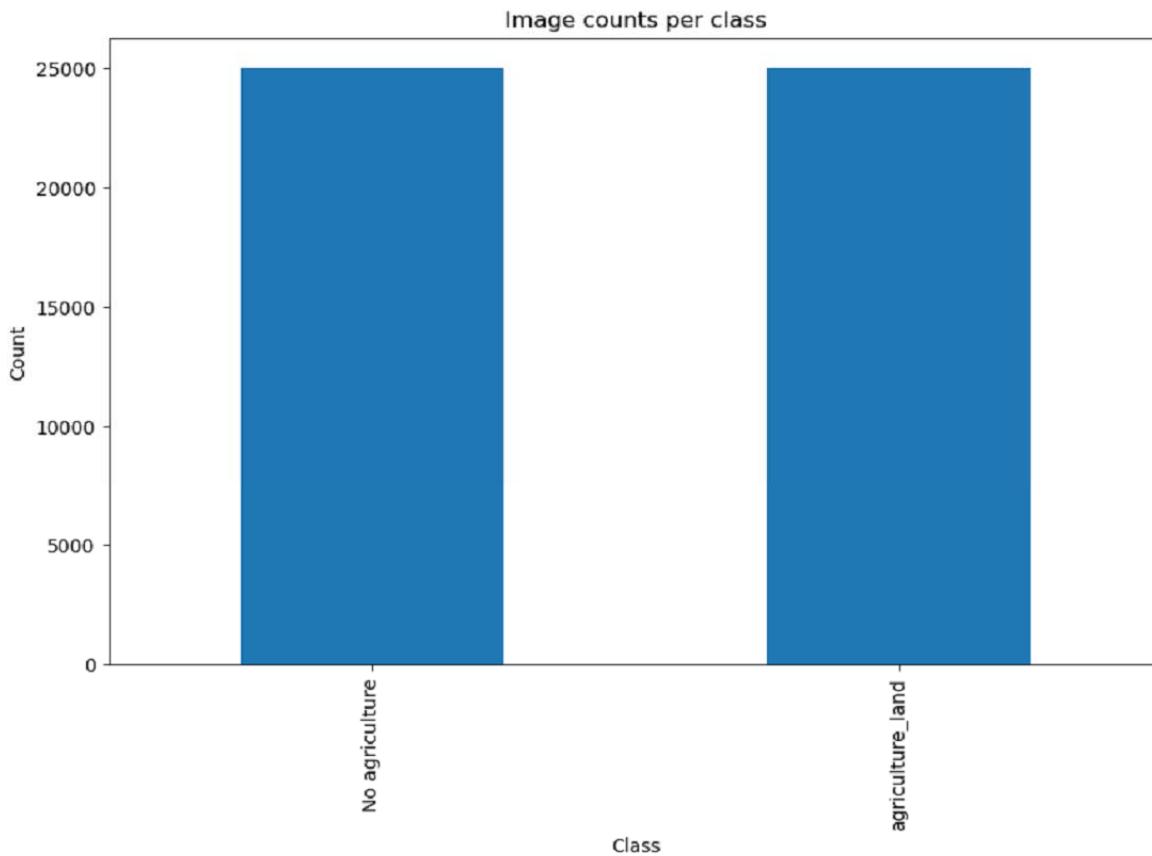


Ilustración 13. Subset balanceado con 50.000 muestras para el modelo binario.

Además, será necesario pre-procesar de nuevo las imágenes ya que el subset que obtendremos será diferente que en el caso anterior. No obstante, reutilizaremos las mismas funciones para ello.

La arquitectura a utilizar para la extracción de embeddings también será la misma (ViT) y en este caso necesitaremos un clasificador binario.

Los resultados obtenidos finalmente en test, son muy superiores, ya que nos dan una accuracy de 0.86 y un F1-score de 0.85 y 0.86 para la clase 'No\_agriculture' y 'agriculture\_land' respectivamente. Además, el ROC-AUC es de 0.95. (Como veremos más tarde, estos buenos resultados no implican que en el entorno de producción los resultados vayan a ser así de buenos, ya que hay muchas otras variables a tener en cuenta que explicaremos en su momento).

	precision	recall	f1-score	support
0	0.83	0.90	0.86	3743
1	0.89	0.81	0.85	3757
accuracy			0.86	7500
macro avg	0.86	0.86	0.86	7500
weighted avg	0.86	0.86	0.86	7500

Ilustración 14. Resultados en test del modelo binario de clasificación para 50.000 muestras.

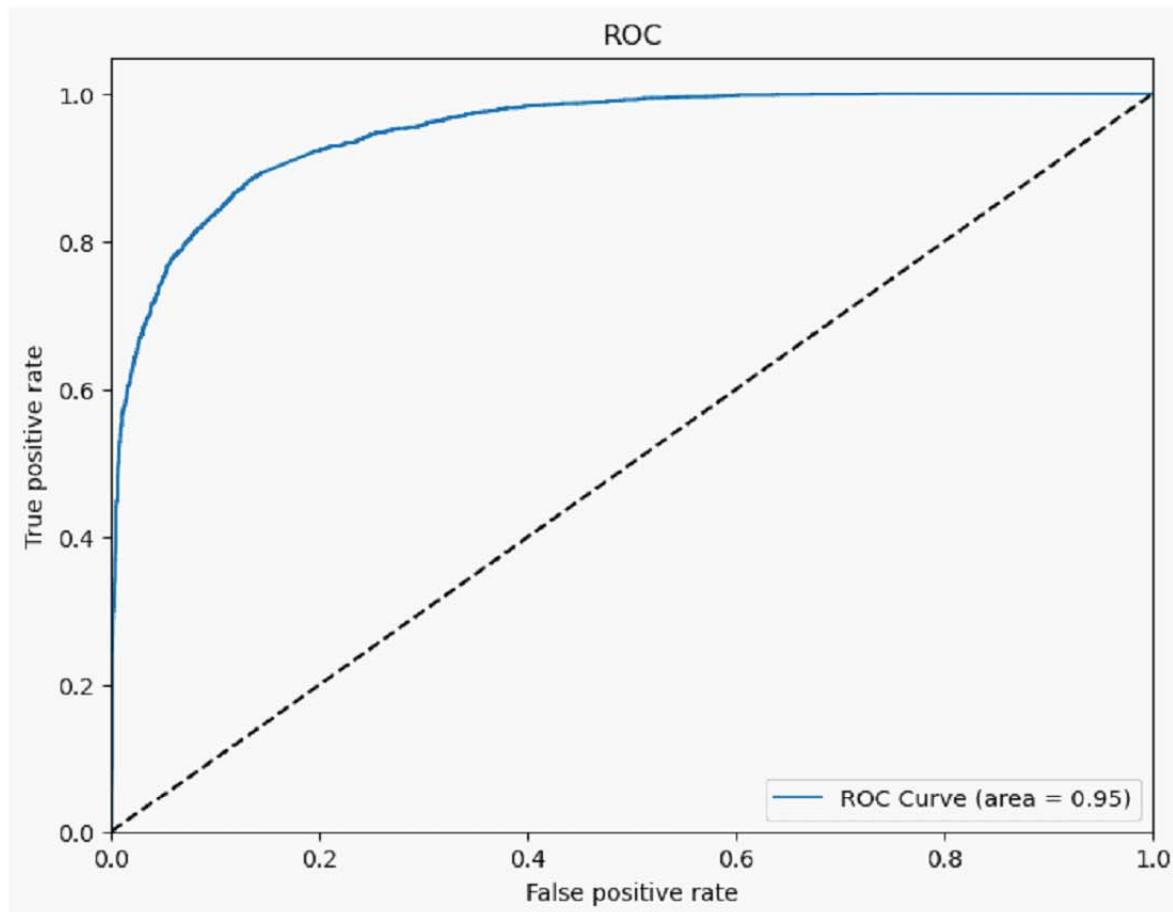


Ilustración 15. ROC-AUC para el modelo binario de clasificación con 50.000 muestras.

Hay algunas diferencias notables que explican esta diferencia con respecto al modelo multiclase. Por un lado, si bien hemos utilizado un máximo de 50.000 muestras, estas se dividían entre todas las clases, mientras que aquí, con esa misma cantidad, contamos con 25.000 ejemplos para cada clase. Por otro lado, también se toma la decisión de eliminar los valores 'unknown' en este modelo. Estas mismas posibilidades, también podrán aplicarse al modelo multiclase y

probablemente los resultados mejoren. Por el contrario, en el caso de las muestras 'No\_agriculture', al haber sido escogidas de forma aleatoria, desconocemos a qué clase original pertenece cada una (podría darse la situación, improbable pero no imposible, de que todas las muestras de 'No\_agriculture' sean de agua, o de bosque) y la capacidad de generalización de este modelo podría ponerse en duda, ya que de forma implícita existe un desbalanceo respecto de las clases originales.

Para ver el proceso de modelado binario de clasificación:

**Binary\_Classification\_Transformers\_DeepGlobe-128.ipynb**

### **Modelo binario de segmentación**

Una vez que se tiene un modelo de clasificación funcional, se decide avanzar con los modelos de segmentación. Este tipo de modelos requieren de mayor capacidad de procesamiento y mayores recursos en general, ya que la clasificación es a nivel de pixel.

Trataremos de utilizar un subset algo más grande que en el caso anterior, (80.000 muestras) y para tratar de obtener algunos resultados comenzaremos con la segmentación binaria.

Además, para tratar de manejar los problemas de recursos, utilizaremos la librería **Dask**, que permite trocear los conjuntos de datos sin necesidad de cargar estos completos en memoria. Si bien el trabajo sigue siendo con el mismo dataset y la carga y pre procesado de los datos es muy similar a los anteriores, en este caso debemos incluir las máscaras, por lo que debemos adaptar la función de pre-procesado a este nuevo escenario. Por lo demás, la tarea es igual que en los puntos anteriores.

Para empezar, entrenaremos una CNN en lugar de utilizar Transformers, y nos decantamos por una arquitectura U-Net ([U-Net: Convolutional Networks for Biomedical Image Segmentation](#)), que, si bien fue diseñada originalmente para segmentación de imágenes en tareas de procesamiento de imágenes médicas, proporciona buenos resultados en tareas generales de segmentación de imágenes.

Debido a que el entrenamiento de una red de estas características requiere de mucho tiempo y con las limitaciones con que contamos actualmente no disponemos de él, realizamos un entrenamiento muy pequeño que no nos da unos resultados útiles, pero que está preparado para poder ampliarlo en el futuro y que será una de las líneas de trabajo principales. En este entrenamiento utilizamos la métrica IoU (Intersection over Union), que representa la relación entre la intersección y la unión de las predicciones del modelo y las máscaras reales en el conjunto en que se evalúa. A pesar de que el resultado no es malo, no se están obteniendo resultados útiles en los tests con diferentes imágenes en este momento, por lo que está pendiente el desarrollo de este modelo.

```
80/80 [=====] - 375s 5s/step
IoU Score (Validation): 0.6860715426499436
```

Ilustración 16. Intersection over Union Score para el modelo binario de segmentación.

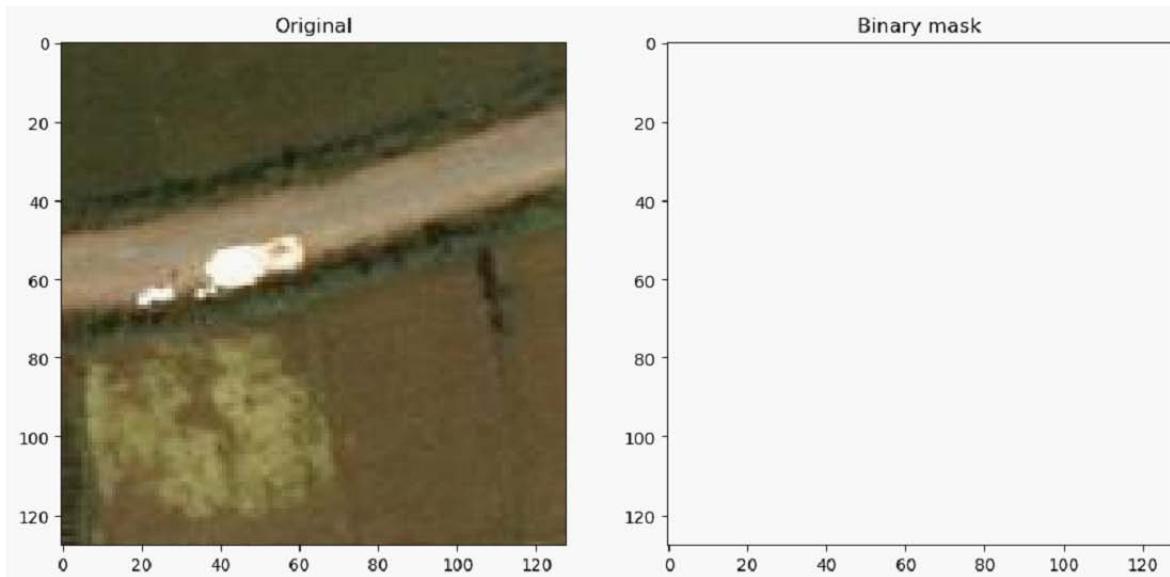


Ilustración 17. El modelo no es capaz de segmentar con un threshold (límite) de 0.5

Finalmente, se trata de repetir el proceso de entrenamiento de un modelo binario de segmentación con Transformers, pero por las mismas razones indicadas en el párrafo anterior, aún no se ha avanzado en ello más que unas líneas y aún no existe un modelo funcional.

Ver: [Binary\\_Segmentation\\_128.ipynb](#)

### Conclusiones primera fase de modelado

Hasta aquí se ha expuesto la primera fase de modelado, pero aún queda una segunda fase de diferenciación entre cultivos verdes y secos. Utilizaremos por simplicidad el modelo con el que mejores resultados hemos obtenido en esta fase: el modelo de clasificación binaria.

### **Fase de diferenciación entre cultivos verdes y secos**

Para la segunda parte de nuestra tarea, la que nos permite diferenciar qué cultivos están verdes y cuales están secos, es necesario analizar diferentes opciones, ya que no disponemos de datos etiquetados que nos digan qué es un cultivo seco y qué es un cultivo verde.

Entre las opciones, se encuentra la utilización de aprendizaje “no supervisado”, pero, en busca de una opción más rápida y eficiente en cuanto a recursos, se decide aplicar una solución alternativa y simple, basada en la fotografía: Dado que las imágenes de las que disponemos y, las que los mapas web generalmente utilizan, están en formato RGB, descomponemos las imágenes en sus tres canales y hallamos aquel que tiene mayor presencia que el resto o mayor *Dominancia*. De este modo, por ejemplo en imágenes de agua tendrá mayor dominancia el canal azul (B) o el verde (G), en imágenes de tierra tendrá mayor dominancia el canal rojo (R) y, en imágenes de cultivos verdes, deberá ser el canal G el que tenga una mayor dominancia.

No obstante, es habitual que en la mayoría de casos, se trate de áreas cultivadas o no, y que no sean de bosque denso o agua, el canal rojo siempre tenga dominancia.

Para solucionar este inconveniente, se crea un coeficiente entre la cantidad de píxeles del canal rojo y la cantidad de pixeles del canal verde y tras hacer pruebas con muestras muy variadas, se determina para nuestro caso de estudio que hasta un 8% más de dominancia del canal rojo sobre el verde, se considerará un cultivo verde y, bajo ese umbral, se considerará un cultivo seco o recolectado. Este umbral siempre será modifiable en función de las necesidades y resultados.

A continuación se ilustra con algunos ejemplos:

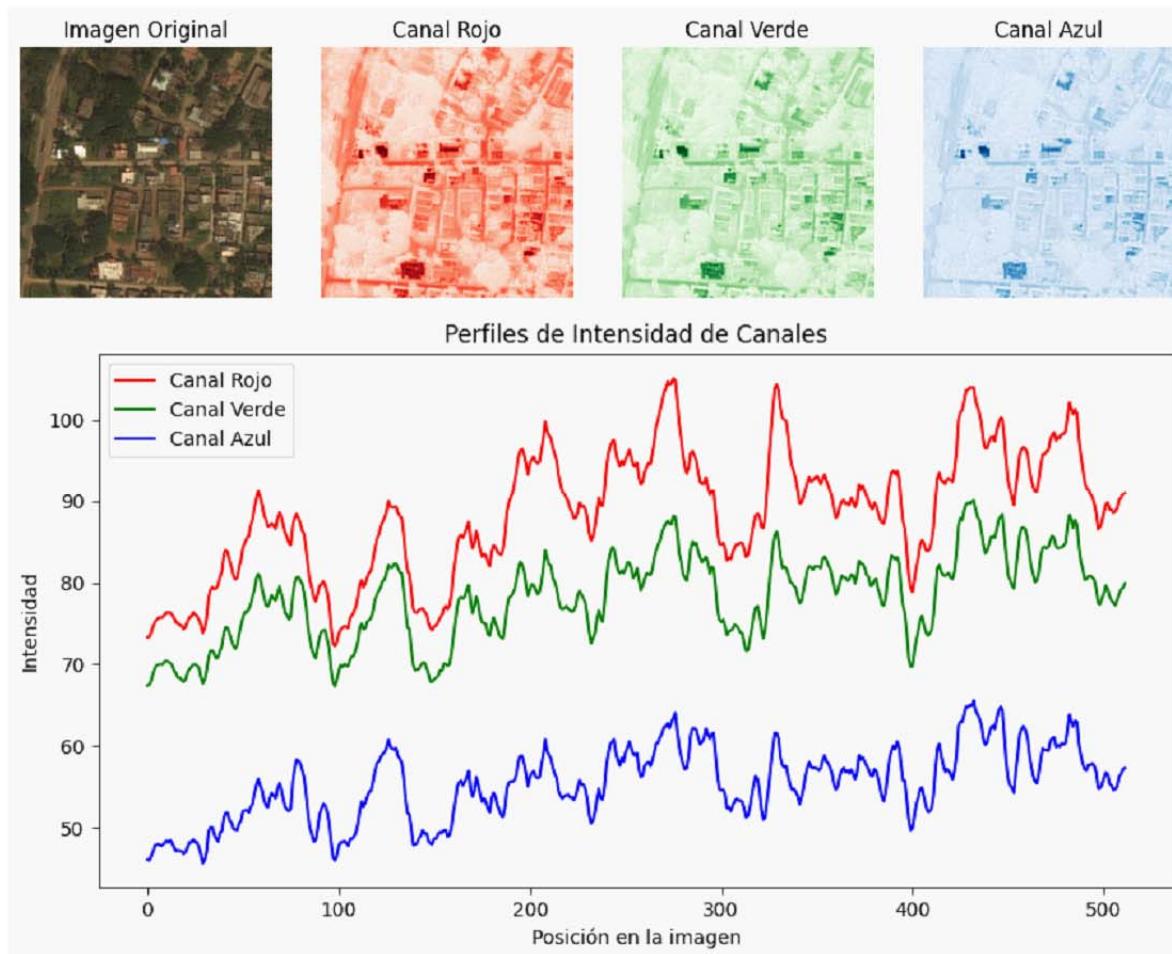


Ilustración 18. Imágenes por canales y gráfico de perfiles de color. Ejemplo 1.

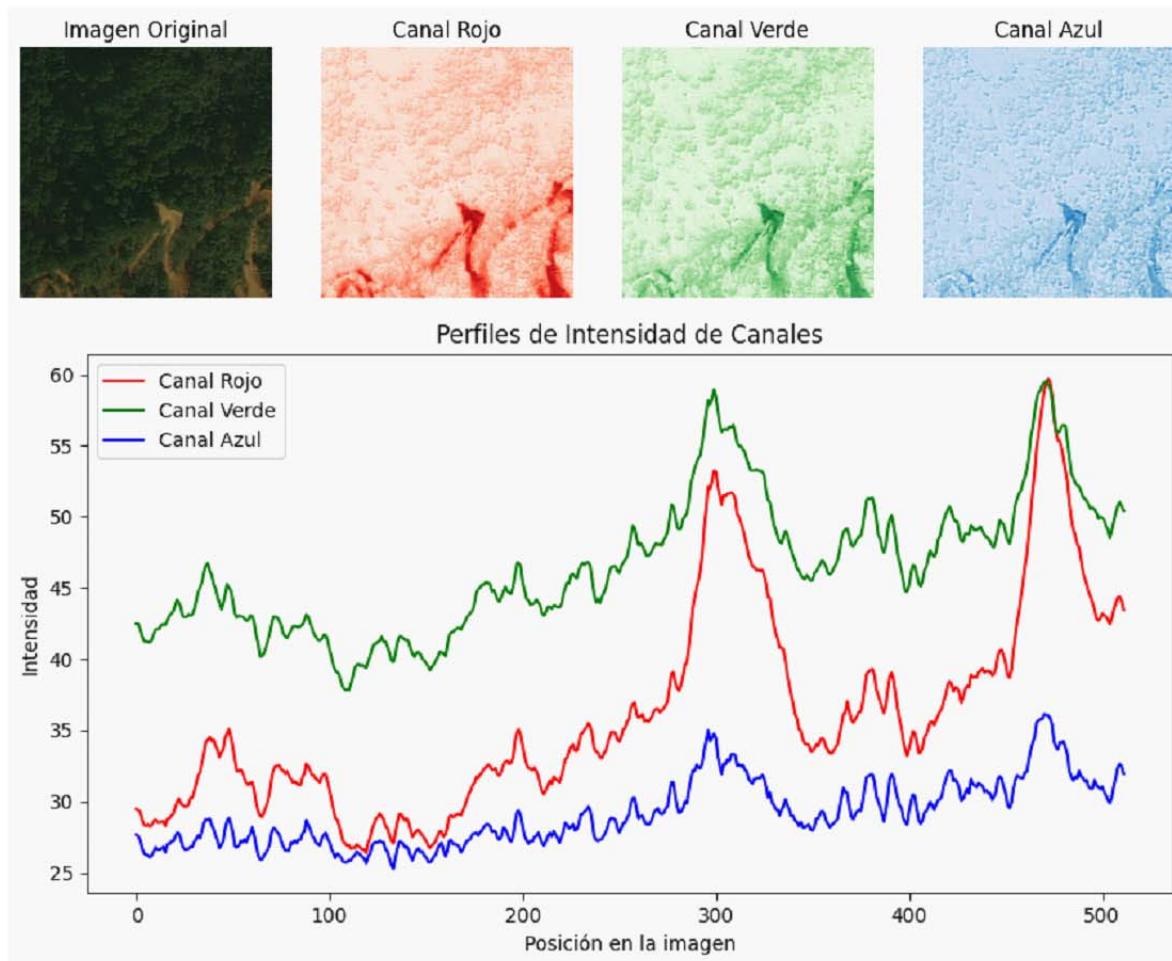


Ilustración 19. Imágenes por canales y gráfico de perfiles de color. Ejemplo 2.

Como se puede observar en los ejemplos anteriores, el canal rojo siempre tiene elevada presencia.

#### Sprint 4 - Construcción maqueta visualización

La duración de esta fase será de una semana. Durante este Sprint, el objetivo es construir una maqueta sencilla del producto, que sirva como muestra del potencial que la herramienta puede llegar a tener con un desarrollo amplio.

Para ello, se diseña una aplicación web que permitirá visualizar un mapa del mundo (nos centraremos en la visualización en una pequeña región, pero no existirán límites en cuanto a la movilidad (al estilo de Google Earth). Para ello, se construirá un visor en HTML y su lógica en Javascript, utilizando la librería Leaflet, ya utilizada por el autor en diversas ocasiones. Todo esto

se conectará con el backend en Python utilizando la librería Flask como servidor local para su desarrollo.

El funcionamiento general de la App será el siguiente:

El frontend se encargará de cargar en pantalla el mapa, desde la API de Leaflet y extraerá las imágenes de éste de forma dinámica (cuando el usuario se mueva por el mapa arrastrando a una nueva región), se seguirán cargando nuevos fragmentos o "tiles" y desde Javascript se extraerán dichos tiles. Estas imágenes se enviarán al backend, donde se pasarán por el modelo que determinará si son cultivo o no y que si es cultivo, calculará, como explicamos en el punto anterior, si está verde o no.

Desde el backend se generará con esta información una máscara para cada imagen que identificará su clasificación. Dichas máscaras serán devueltas al frontend y colocadas delante de su respectivo tile.

No es necesario extenderse más en la explicación de este apartado ya que no es la parte central del proyecto, por lo que para cualquier revisión o consulta acerca del código, se puede acceder a la carpeta donde se encuentra todo lo relativo a la aplicación: **App**.

## Sprint 5 - Deploying maqueta Google Cloud

Este sprint durará una semana. El objetivo de esta fase, aunque no imprescindible en esta etapa del proyecto, es poder mostrar en tiempo real el funcionamiento del prototipo.

Con este objetivo, y debido a una mayor familiaridad con la plataforma, se decide utilizar Google Cloud. Se crea una máquina virtual y se configurarla para que la App web sea accesible desde cualquier navegador.

## Sprint 6 - Generación informes y documentación

Se prevé que este sprint dure una semana. En esta última fase, el objetivo es ultimar la preparación del proyecto para su presentación, además de elaborar todo el material de apoyo necesario para ello. Se incluye también un video con la presentación de la maqueta. También se graba un video con la presentación del proyecto, con el objetivo de que pueda ser revisado en cualquier momento.

Se organiza de la misma manera un repositorio GitHub donde se puede revisar el proyecto al completo, notebooks y su documentación. Solo unos días después de la presentación los videos serán eliminados del repositorio por razones de privacidad.

## Dificultades y desafíos

Durante las diferentes etapas o sprints del proyecto, se encuentran algunos desafíos que en general se han abordado con éxito.

Se identifican dos desafíos principales a lo largo del desarrollo de esta fase del proyecto, que después se pueden ir desgranando. Por un lado, está la **limitación de recursos**, tanto temporales como materiales, principalmente derivados de la necesidad de trabajar siempre con recursos gratuitos. Por otro lado, está la **complejidad de la tarea** en sí, que, como se expone en las siguientes líneas, obliga a tomar decisiones excluyentes en algunos momentos del desarrollo.

Respecto a la **limitación de recursos**, ha habido que enfrentarse a algunas dificultades en varios momentos del proyecto que han afectado a la toma de decisiones.

El trabajo de datos y modelado se ha realizado en notebooks, tanto con Jupyter Notebook como con la versión gratuita de Google Colab. Debido a que el trabajo es con un volumen importante de imágenes, existen diferentes procesos lentos, que requieren de cierta capacidad de cómputo y de tiempo para ejecutarse. Si bien originalmente la opción ideal era Google Colab, ya que proporciona GPU, se dan continuamente problemas de cierres de sesión y de expulsión de las GPU, además de no ser posible dejar una tarea ejecutando durante varias horas sin supervisión, ya que también supone el cierre de la sesión, debiendo muchas veces tener que comenzar de nuevo la ejecución de estas. Dicha situación se da especialmente en las tareas de pre procesado de los datos y en las de extracción de embeddings, así como en las de modelado en el caso de CNNs. Debido a esto, se toma la decisión de trabajar con Jupyter Notebook, lo cual limita la tarea al uso de CPU.

Estas condiciones obligan a trabajar con una cantidad más reducida de datos, para poder tener unos tiempos de ejecución sostenibles y a programar ciertas tareas para que se ejecuten en las noches o en fines de semana completos.

También es esta la causa de que se haya decidido priorizar en esta implementación los modelos de clasificación frente a los de segmentación.

En cuanto a la **complejidad de la tarea**, se encuentran también varios desafíos que abordar. El primero de ellos, consiste en la elección de los modelos, y que, en función de ello, llevará una preparación de los datos diferente. Por ejemplo, en la etapa de análisis, no se tiene en consideración que de las 7 clases existentes, una de ellas 'unknown', pueda ser un problema, debido a su pequeño número de muestras y a su variabilidad, no obstante, la diferencia de usarlos a no usarlos en diferentes modelos es apreciable y en el caso del modelo multiclasé, el resultado podría mejorarse teniendo esto en consideración.

Por otro lado, existen diferencias notables entre el manejo y pre procesado de los datos para los modelos multiclasé y para los binarios: En la decisión de limitar el número de muestras se tiene

en cuenta el balanceo de los datos, y se logra entrenar un modelo con un dataset balanceado en el caso multiclasificación (a excepción de la clase 'unknown'). Pero en el caso del modelo binario, los datos se dividen entre 'agricultura' y 'no agricultura', cabiendo dentro de la etiqueta 'no agricultura' imágenes muy diversas y aleatorias, desconociendo, como se comenta anteriormente, si entre esas imágenes 'no agricultura' puede darse el caso de que la mayoría sean de la misma clase original o, de que apenas existan de alguna determinada clase. Esta última situación podría reducir considerablemente la calidad del modelo obtenido, ya que en el etiquetado original existen clases que aparentemente pueden ser bastante similares, como cultivos y pastizales, y dicha posibilidad no se ha controlado en esta ocasión.

La dificultad más importante y compleja de abordar, que se detecta en la fase de pruebas del proyecto, es el zoom. Esta circunstancia particular de los mapas, y que permite al usuario variar la distancia desde la que ve el mapa, hace que el modelo solo funcione en una determinada "frecuencia", es decir, con un determinado nivel de zoom, que es con el que ha sido entrenado, y conforme se va aumentando o reduciendo este, va perdiendo precisión de forma notable. La solución óptima a este problema consiste en entrenar modelos para diferentes niveles de zoom, y para ello requeriremos datos en varios niveles de zoom y, una vez obtenidos, realizar el trabajo completo de procesado para cada set. Se trata de una tarea larga y no fácil de lograr, especialmente en cuanto a la obtención de datos, que se deberá llevar a cabo en próximas iteraciones.

Por último, otro desafío importante, que aún no se resuelve, es la necesidad de una alta precisión en la detección que se requiere para esta tarea y que es difícil lograr con los modelos de clasificación de imágenes. Con esta línea de trabajo es necesario reducir el tamaño de las imágenes a entrenar, para que el modelo pueda trabajar sobre imágenes más pequeñas y el área de detección también reducida. Además, también es importante el trabajo en los modelos de segmentación, que son capaces de realizar predicciones a nivel de píxel, lo que puede solucionar el problema de la precisión.

## Tareas pendientes y posibilidades de mejora

Existen diferentes líneas de acción definidas para las próximas iteraciones de este proyecto, que supondrán un trabajo más largo que el llevado a cabo hasta aquí.

- 1- En primer lugar, el trabajo con los modelos de segmentación, que se está comenzando a llevar a cabo, para segmentación binaria, y que es necesario desarrollar en profundidad, alimentar con más imágenes y lograr mejores entrenamientos para obtener unos buenos resultados.
- 2- La segunda línea de trabajo, consiste en continuar mejorando los modelos de clasificación, para lo que hay amplio margen, ya que apenas hemos utilizado un 15% de los datos disponibles para estos modelos como ya comentamos anteriormente.
- 3- Una línea que aún no se ha desarrollado y que puede ser más experimental, consiste en estudiar las posibilidades de aplicar el modelo SAM (Segment Anything Model) de Meta,

dado que existen otras necesidades que parten a raíz de este proyecto, como la detección de edificaciones no identificadas en bosques, que también son una necesidad en el combate de incendios forestales.

Respecto de las dos primeras líneas de trabajo, y si el aumento en la cantidad de datos no resulta suficiente para obtener unos resultados óptimos, se prevé la posibilidad de trabajar con las imágenes en el formato de bandas y no RGB, lo que debería ayudar a los modelos a ver y detectar exclusivamente en las bandas relativas a la vegetación.

En los modelos multiclase, se va a trabajar eliminando previamente las muestras de categoría 'unknown', para trabajar solo con 6 clases.

En los modelos binarios, se realizará un balanceo que garantice que hay el mismo número de muestras de cada sub-clase original dentro de la clase 'no\_agriculture', lo que debe mejorar la calidad del entrenamiento.

Por último, desde este proyecto surge una idea de proyecto paralelo muy interesante, debido a la condición de los datos. Como se indica inicialmente, el dataset original se compone de una cantidad de imágenes etiquetadas con sus respectivas máscaras y además, de otras imágenes sin etiquetar. Es un desafío importante el lograr etiquetar esas imágenes mediante aprendizaje semi supervisado.

## Conclusiones

El desarrollo de este proyecto ha conllevado diferentes desafíos y situaciones que han requerido de soluciones y habilidades diversas.

Desde el punto de vista de la aplicación de metodologías, tecnologías y aprendizajes adquiridos en el último año en KeepCoding, supone un auténtico compendio de lo realizado a lo largo de todos estos meses de trabajo. Pero desde el punto de vista de la necesidad de investigación, ampliación de conocimientos y aplicación de estos, ha supuesto un desafío que ha sido siempre motivador y sumamente enriquecedor. Sin los conocimientos adquiridos durante el bootcamp, hubiera sido imposible o como mínimo, hubiera supuesto un esfuerzo mucho más largo y desordenado llegar hasta aquí.

En cuanto al desarrollo de un proyecto de estas dimensiones, supone adquirir la confianza necesaria para enfrentarse a proyectos muy diversos en el futuro, saber que siempre van a existir desafíos que será necesario abordar desde diferentes ángulos y con algo de pensamiento "lateral" y sobretodo, añadir motivación por enfrentarse y hacerse cargo de proyectos y tareas como esta. Supone además un acicate a continuar formándose, a desear seguir buscando conocimientos y a tener en la mira nuevos proyectos que desarrollar y muchos papers pendientes que revisar. Además de no olvidar que cualquier proyecto en nuestro ámbito requiere adquirir nuevos conocimientos y habilidades de algún tipo por su propia naturaleza innovadora.

*Detección de estado de cultivos mediante remote sensing para mejora del despliegue de medios en incendios forestales*

Por último, como ya se indica en el apartado anterior, el proyecto que aquí se presenta, es tan solo un proyecto en su primera etapa, al que le queda por delante mucho más desarrollo que el que ha tenido hasta este punto. Por esto, continúa siendo tan motivante y desafiante, a la vez que se desprenden de él ampliaciones y líneas paralelas de trabajo que podrán formar nuevos proyectos en sí mismas y anima a continuar aprendiendo, investigando y afrontando desafíos.