

## Análisis de imágenes multiespectrales para detección de patrones en Agricultura

Arcia Hernández Jesús, Fajardo Becerra Daian, Salazar Escobar Carlos, & Sepúlveda Jimenez Hernán.

Universidad EAFIT.

Maestría en ciencia de los datos y analítica.

Estadística Multivariada

2021.

## Pregunta de Investigación

¿Qué algoritmos y metodologías de aprendizaje de máquina, son más efectivas para reducir el error de generalización y obtener una precisión mayor al 60%, para la clasificación de imágenes con características en cultivos principalmente de soya y maíz?

## Objetivo General

Identificar y clasificar imágenes de diferentes cultivos de acuerdo con sus características con una precisión mayor al 60%.

## Objetivos Específicos

1. Investigar diferentes métodos para clasificación de imágenes, con diferentes niveles de complejidad.
2. Seleccionar tres métodos de clasificación con diferentes grados de complejidad (baja, media y alta).
3. Evaluar cada técnica de clasificación con diferentes métricas para clasificación con el fin de seleccionar la más eficaz para el set de datos de imágenes seleccionado, buscando mejorar el modelo básico.
4. Analizar como los diferentes clasificadores afectan la estimación del error de generalización.
5. Evaluar los diferentes clasificadores.

## Revisión de la literatura, estado del arte

*La regresión logística* se utiliza para problemas de clasificación de dos clases, en el cual se asignan una etiqueta 1 si el valor es correcto y una etiqueta 0 en caso de ser negativo. Sin embargo, se conoce dos formas de atacar problemas multinomiales:

1. One-vs-rest: Permite el uso de la regresión logística en problemas multiclase, aunque requieren que el problema de clasificación se transforme primero en múltiples problemas de clasificación binaria.  
Dentro de este enfoque se intenta dividir el problema multiclase en varios problemas binarios, ajustando la regresión a cada uno de esos problemas
2. La regresión logística multinomial es una extensión de la regresión logística tradicional, en el cual se cambia su función de perdida por una función de perdida con entropía cruzada y distribución de probabilidad multinomial.

Lo que se busca con este tipo de algoritmos es utilizar theta en cada iteración para poder establecer una relación entre las características de entrada y la etiqueta salida por medio de la función sigmoide. Esto puede lograrse por medio de la función de costo por el cual se indica la distancia que se tiene de la predicción al valor real; y el descenso del gradiente.

Después de la primera iteración, se busca realizar diferentes estrategias con el fin de poder mejorar las métricas de evaluación de los modelos, como lo son: Feature Scaling o normalización; Revisar el desequilibrio de las clases; Ajustar hiperparametros; y Agregar más métricas.

Un *árbol de decisión* es una estructura arbórea similar a un diagrama de flujo en la que un nodo interno representa una característica (o atributo), la rama representa una regla de decisión y cada nodo de hoja representa el resultado. El nodo superior de un árbol de decisión se conoce como nodo raíz. Aprende a dividir en función del valor del atributo. La partición del árbol se denomina partición recursiva. Esta estructura similar a un diagrama de flujo le ayuda en la toma de decisiones. Se visualiza como un diagrama de flujo que imita fácilmente el pensamiento a nivel humano. Por eso los árboles de decisión son fáciles de entender e interpretar.

*Random Forest* es un tipo de Ensamble en Machine Learning que combina diversos árboles, la salida de cada uno es un voto y la opción más votada será la respuesta del bosque aleatorio.

El algoritmo funciona de la siguiente manera:

- Selecciona  $k$  características de las  $n$  totales con  $k < m$  y se crea un árbol de decisión con esas  $k$  características.
- Se crean  $n$  arboles variando siempre la cantidad de características.
- Se toma cada uno de los  $n$  árboles y cada uno hace la misma clasificación y se guardan las  $n$  respuestas.
- Por último, se calculan los votos obtenidos para cada clase y se toma la más votada como la clasificación final.

Este tipo de algoritmo es utilizado en varias aplicaciones, sin embargo, las más populares son aprobación de créditos, diagnósticos médicos y marketing.

En 2004 Yan y Goebel reconocen que el rendimiento de los random forest puede ser limitado, Para potenciar y mejorar el rendimiento, se ha utilizado la clasificación por conjuntos, que se basa en el aprendizaje por conjuntos. El aprendizaje por conjuntos utiliza múltiples modelos para obtener un rendimiento predictivo mejor que el que podría obtenerse de cualquiera de los modelos constitutivos. En 2010 Rokach empieza a emplear múltiples clasificadores y luego los utiliza colectivamente para identificar las instancias no etiquetadas.

*Las redes neuronales convolucionales* se volvieron populares gracias a AlexNet ganando el desafío de ImageNet; el principal objetivo de las redes neuronales ha sido hacer redes más profundas y complicadas para conseguir una mayor precisión, pero esto ha presentado diferentes desafíos como la eficiencia de los modelos respecto a su velocidad.

*ResNet50* Es una red usada como elemento básico para tareas de visión por computadora; este tipo de red se inspira en el hecho que algunas neuronas se conectan con neuronas en capas no necesariamente contiguas, por lo que se saltan capas intermedias. Lo que hace impactante en ResNet50 es que permite por primera vez entrena redes muy profundas (de más de 100 capas) comprobando el problema de desvanecimiento de gradiente.

En 2012 AlexNet fue la ganadora de ImageNet, ya que ayudo al aprendizaje profundo con solo 8 capas convolucionales, la red VGG tenia 19 capas, GoogleNet 22 y ResNet 152. Esto es un gran avance ya que las redes profundas son difíciles de entrar debido al notorio problema del gradiente haciendo que este sea extremadamente pequeño, por lo que el rendimiento del algoritmo se satura y empieza a degradarse rápidamente.

Lo que se realiza en ResNet es no conectar todas las capas necesariamente, y esto se realiza por dos razones:

1. Mitigan el problema del desvanecimiento de gradiente, ya que permite que exista un camino alternativo para que el gradiente fluya
2. Permite que el modelo aprenda la función de identidad que garantice que la capa superior funcione al menos tan bien como la capa inferior.

Este tipo de arquitectura como FCN y la red U, en las cuales se buscan pasar la información de una capa de muestreo descendiente a las capas de muestreo ascendente.

*MobileNet* Se basa en la arquitectura que es utilizada en convoluciones separables en profundidad para construir redes neuronales profundas de poco peso, con dos hiperparámetros globales ayudan a mejorar notoriamente la latencia y precisión del modelo. Dentro de la construcción de MobileNet se debe reducir el cálculo de las primeras capas, también es necesario aplanar las capas con el fin de poder construir una red a partir de convoluciones totalmente factorizadas.

MobileNet utiliza convoluciones separables en profundidad (Esta convolución tiene su origen en la idea de que la profundidad y la dimensión espacial de un filtro pueden separarse, de ahí el nombre de separable). Reduce significativamente el número de parámetros en comparación con la red con convoluciones regulares con la misma profundidad en las redes. Esto da lugar a redes neuronales profundas y ligeras. Una convolución separable en profundidad se realiza a partir de dos operaciones.

1. Convolución en profundidad.
2. Convolución puntual.

## Metodología de Investigación

La metodología utilizada para este proyecto es Team Data Science Process (TDSP), que es una metodología ágil e iterativa de ciencia de datos para entregar soluciones de análisis predictivo y aplicaciones inteligentes de manera eficiente. TDSP tiene 4 componentes principales:

Componentes	Subcomponentes	Donde encontrarlo
<b>Data Science LifeCycle</b>	Bussiness Understanding	Pregunta de investigación y objetivos del proyecto
	Data Acquisition and understanding	Análisis de los datos
	Modelling and Deployment	Uso de metodologías de aprendizaje estadístico.
<b>Standardized Project Structure</b>		Entregables
<b>Infraestructure Resources</b>		Uso de metodologías de aprendizaje estadístico; Entregables
<b>Tools and Utilities</b>		Uso de metodologías de aprendizaje estadístico; Entregables

## Análisis de los datos

Gracias al procesamiento que fue realizado con TensorFlow y la lectura de las imágenes por medio de TFRecords se pudo encontrar que las etiquetas con las cuales se deben trabajar las clasificaciones de imágenes son:

	Class	label
0	storm_damage	0
1	drydown	1
2	endrow	2
3	water	3
4	nutrient_deficiency	4
5	double_plant	5
6	waterway	6
7	weed_cluster	7
8	planter_skip	8

Table 1: Etiquetas para Clasificación

Gracias a la carga y procesamiento que se realizó con Tensor Flow, se puede realizar una visualización de las etiquetas presentadas en la tabla 1:

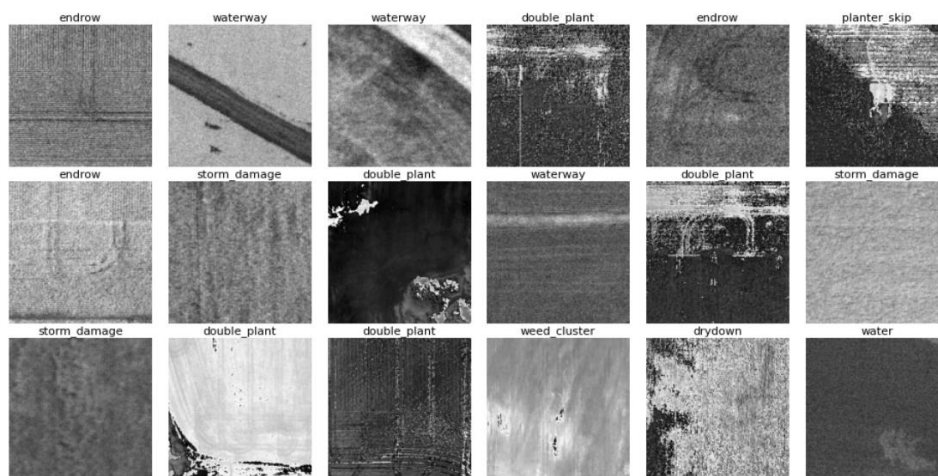


Figure 1: Visualización de imágenes de acuerdo a sus etiquetas







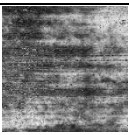


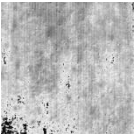

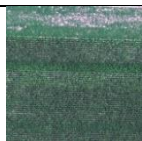
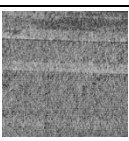

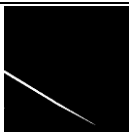

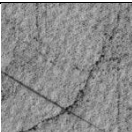
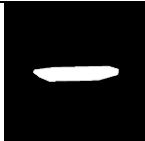

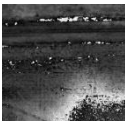

Para cada sección del campo agrícola fotografiado se presentan cinco variantes de la misma imagen distribuidas en los siguientes grupos:



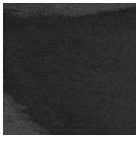


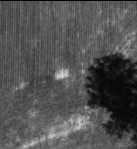

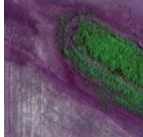
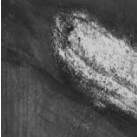

1. **RGB:** Clasificada en el grupo basado en la adicción de colores lumínicos primarios (rojo, verde y azul).
2. **NIR:** Near infrared, correspondiente a la imagen del espectro infrarrojo.
3. **Field labels:** Contiene cuadriláteros demarcando el área de la fotografía que presenta una afectación característica en particular.
4. **Field bounds:** Contiene la imagen bicromática que discrimina que porción del área de la fotografía corresponde a cultivo y cual a algo diferente. (i.e. Una Carretera)

Análisis de imágenes multiespectrales para detección de patrones en Agricultura

5. *Field masks*: Contiene imagen bicromática que determina si hay pixeles inválidos en la imagen.

Algunos ejemplos son:

Característica: Doble plantación				
field_labels/double_plant/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Desección				
field_labels/drydown/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Surco				
field_labels/endrow/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Falta de nutrientes				
nutrient_deficiency/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Salto de siembra				
field_labels/planter_skip/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Daño por tormenta				
field_labels/storm_damage/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				

Característica: Agua estancada				
field_labels/water/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Malezas				
field_labels/weed_cluster/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				
Característica: Camino del agua				
field_labels/waterway/	field_images/rgb/	field_images/nir/	field_bounds/	field_masks/
				

## Uso de la metodología y herramientas de aprendizaje estadístico

Para el desarrollo de este proyecto, se seleccionaron cuatro clasificadores con diferente nivel de complejidad, con el fin de conocer su desempeño con la clasificación de imágenes de cultivos por medio de sus características

### Logistic Regression:

Este algoritmo fue seleccionado como uno de los algoritmos mas básicos para la clasificación de imágenes, y para su desarrollo se realizó el siguiente proceso:

- 1) Cargar el Dataset: Se hizo una carga del Dataset, creando así la matriz de características y su variable respuesta. Cada "característica" es representada por los píxeles de la imagen, y cada píxel puede tomar cualquier valor entero de 0 a 255. Un valor grande para un píxel significa que hay escritura en esa parte de la imagen. Por lo que para este Dataset donde sus imágenes son de 512\*512 se tendrán 262144 características.
- 2) Preparación de la data: Se divide el conjunto en conjunto de prueba y de entrenamiento utilizando train test Split de sklearn.
  - a) En la primera iteración no se hizo escalamiento de los datos.
  - b) En la segunda iteración se hizo escalamiento dividiendo cada característica en 255, así tener un valor entre 0 y 1 y el accuracy mejora en un 8%

- 3) Entrenamiento: Se utilizo LogisticRegression de Sklearn. A diferencia de la regresión lineal, no hay una solución de forma cerrada para la estimación del parámetro de mínimos cuadrados en la regresión logística. Por lo tanto, necesitamos utilizar un "solucionador" que encuentre una solución numérica.
  - a) Se utiliza el parámetro multi\_class, para poder realizar la clasificación de las 9 características
  - b) Se utiliza el solucionador que es Saga ya que este:
    - i) Funciona bien cuando hay un gran número de muestras,
    - ii) Soporta la regresión logística sin penalización de regularización, penalización L1, penalización L2, o ElasticNet
      - (1) L1: Lasso: Es la media del valor absoluto de los coeficientes del modelo. ayuda cuando sospechemos que varios de los atributos de entrada (features) sean irrelevantes.
      - (2) L2: Ridge: Es la media del cuadrado de los coeficientes del modelo. ayuda cuando sospechemos que varios de los atributos de entrada (features) estén correlacionados entre ellos
      - (3) ElasticNet: combinación de L1 y L2: cuando tengamos un gran número de atributos
      - (4) La Regularización se aplica por defecto.
    - iii) Soporta la regresión multinomial con múltiples clases, utilizando la función softmax.
- 4) Interpretación de coeficientes: Podemos utilizar los valores de los coeficientes para entender qué características (es decir, qué píxeles) son importantes para determinar a qué clase pertenece una muestra.
  - a) Si una muestra tiene valores grandes en los píxeles mostrados en azul, la probabilidad de que esa muestra sea la característica buscada aumenta.
  - b) Si la muestra tiene valores grandes en los píxeles del centro de la imagen, la probabilidad de que la muestra sea la característica buscada disminuye
  - c) Muchos píxeles tienen coeficientes cuya magnitud es muy pequeña. Estos se muestran en blanco, y no son muy importantes para esta tarea de clasificación.

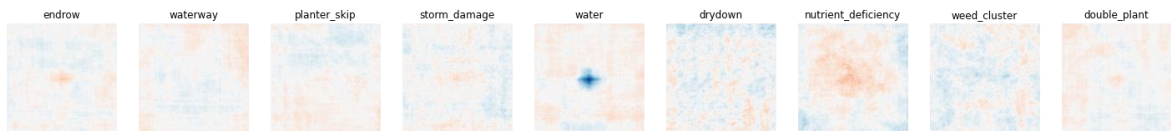


Figure 2: Interpretación de coeficientes

- 5) Fit el Modelo
  - a) Se toma una imagen aleatoria para revisar su probabilidad condicional
  - b) Para obtener la etiqueta predicha, podemos encontrar la clase con la mayor probabilidad.
  - c) La implementación de LogisticRegression en sklearn incluye funciones para calcular tanto la probabilidad por clase como la etiqueta más probable.



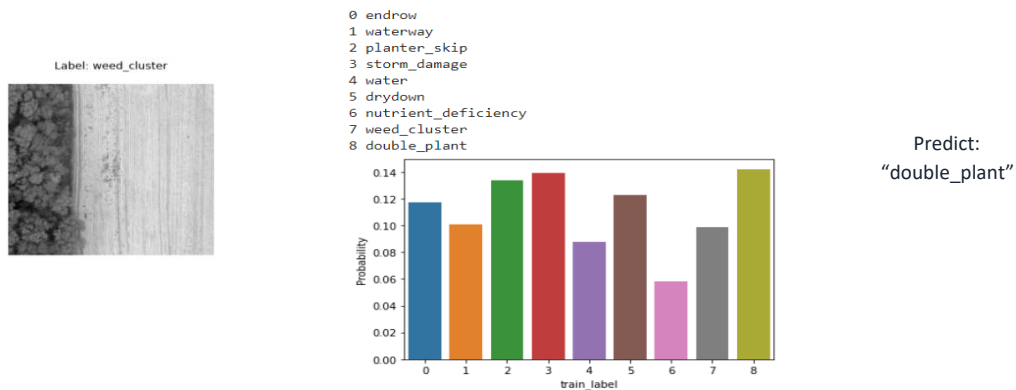


Figure 3: Fit de Regresión logística

- d) Evaluación del clasificador: Para un problema multiclase, podemos ampliar la matriz de confusión para que tenga más filas y columnas. La diagonal de la matriz de confusión multiclase muestra el número de clasificaciones correctas para cada clase, y las otras entradas muestran los casos en los que a una muestra de una clase se le asignó erróneamente una etiqueta de clase diferente.

	precision	recall	f1-score
double_plant	0.12	0.04	0.06
drydown	0.09	0.31	0.14
endrow	0.29	0.16	0.21
nutrient_deficiency	0.14	0.08	0.11
planter_skip	0.38	0.31	0.34
storm_damage	0.11	0.04	0.06
water	0.21	0.33	0.25
waterway	0.19	0.58	0.29
weed_cluster	1.00	0.03	0.06
accuracy			0.19
macro avg	0.28	0.21	0.17
weighted avg	0.32	0.19	0.16

Table 2: Metricas Regresión Logística Training

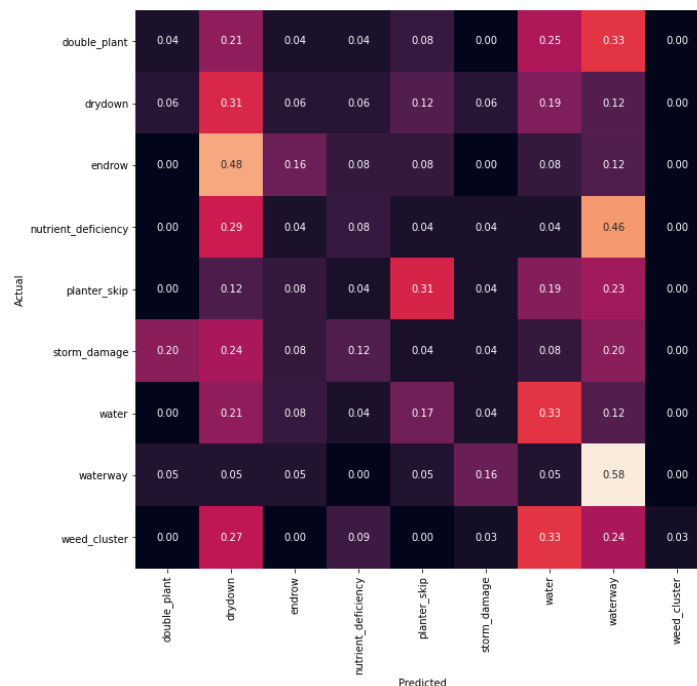


Figure 4: Matriz de Confusión Training

Como se puede observar, waterway y wáter son los más fáciles de clasificar para la regresión logística mientras que weed\_cluster, storm\_damage, nutrient\_efficiency, double\_plant son más difíciles ya que su precisión de la clasificación fue menor para estas etiquetas.

### Random Forest:

La estrategia para clasificar imágenes usando el algoritmo de bosques aleatorios es construir un dataset con las características de las imágenes RGB, para esto se leerá la imagen como una matriz de 512x512x3 dónde cada los tres canales corresponden al Verde, Azul y Rojo.

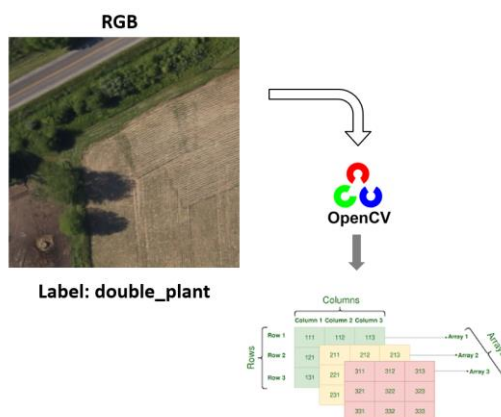


Figure 5: Lectura de la Imagen como matriz

Para extraer las características de la imagen se utilizarán tres métodos de extracción:

Análisis de imágenes multispectrales para detección de patrones en Agricultura

- *Hu Moments*: o momentos, son una medida ponderada de las intensidades de los píxeles de la imagen, son Valores entre 0 y 1, estos *moments* capturan alguna noción de forma; el tipo de momento más sencillo se define como:

$$M = \sum_x \sum_y I(x, y)$$

- *Haralick Texture*: Esta característica describe la textura de una imagen usando la Matriz de Co-ocurrencia a niveles de grises, esta matriz registra que tan seguido aparecen pares de píxeles adyacentes con determinados valores (ejemplo, cuántas veces aparece el par [1, 2], [2,3], [2, 1], etcétera).

El descriptor *Haralick* obtiene 4 matrices correspondiente a las cuatro direcciones (de izquierda a derecha, de arriba abajo, de la esquina superior izquierda a la esquina inferior derecha, de la esquina superior derecha a la esquina inferior izquierda). El resultado son 4 vectores de dimensión 13 o 14, para obtener el vector final, se promedian los cuatro vectores [1].

- *Color Histogram*: Esta característica mide el valor de la saturación del tono con rango entre 0 y 255.

Extraídas las características se almacena en formato tabular donde cada fila es una imagen y las columnas son las características anteriormente descritas.

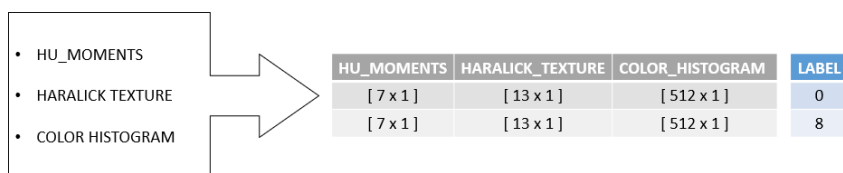


Figure 6: Formato tabular de características.

*Nuevas estrategias*: Se adicionó como característica la Imagen NIR y se adiciona como característica al dataset utilizando la misma estrategia de obtención de características.

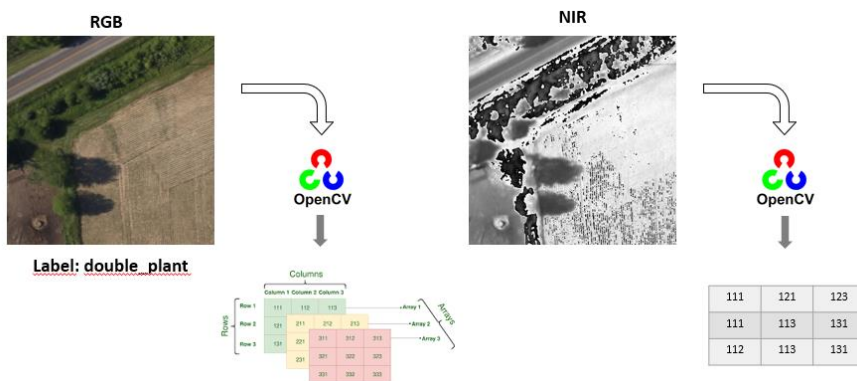


Figure 7: Estrategia RGB + NIR

También se utilizó la imagen “bounds” que nos indica que parte de la imagen es un cultivo y cual no esto ayudaría a eliminar el ruido de la imagen.

En la siguiente tabla se evidencia la precisión de las iteraciones realizadas con las estrategias realizadas:

ITERACION	# DE ARBOLES	MAX CARACTERISTICAS	MAX PROF	ACCURACY	TEST	OBS
1	50	NONE	NONE	100%	39% RGB	
2	25	NONE	NONE	100%	43% RGB	
3	20	NONE	NONE	100%	39% RGB	
4	10	NONE	NONE	99%	38% RGB	
5	5	NONE	NONE	97%	36% RGB	
6	1	NONE	NONE	81%	28% RGB	
7	25	NONE	5	79%	38% RGB	
8	25	NONE	10	97%	40% RGB	
9	25	NONE	11	99%	37% RGB	
10	25	NONE	NONE	100%	41% RGB + NIR	
11	25	NONE	4	70%	38% RGB + NIR	
12	25	NONE	5	81%	42% RGB + NIR	
13	25	NONE	10	100%	41% RGB + NIR	
14	25	NONE	NONE	100%	44% RGB + BOUNDS	
15	25	NONE	5	76%	31% RGB + BOUNDS	
16	25	NONE	6	83%	45% RGB + BOUNDS	
17	25	NONE	11	99%	43% RGB + BOUNDS	
18	25	NONE	NONE	100%	35% RGB + NIR + BOUNDS	
19	25	NONE	5	77%	27% RGB + NIR + BOUNDS	
20	25	NONE	7	91%	29% RGB + NIR + BOUNDS	
21	25	NONE	10	98%	28% RGB + NIR + BOUNDS	

Table 3: Resultados Random Forest

### **Modelo CNN:**

La primera arquitectura utilizada para el clasificador de redes neuronales convolucionales se elaboró con un modelo de 20 capas convolucionales y finalizando con dos capas densas.

Las redes neuronales en general tienden a ser lentas y son propensas a sobreajuste, por lo que requieren de mecanismos de regularización y, por tanto, se incluyeron en el modelo varias capas de Dropout, mecanismo mediante el cual se esconde un porcentaje de las neuronas de algunas capas, y de BatchNormalization, que cumple el mismo propósito de regularización, pero a través de simplificar los enlaces entre neuronas de dos capas contiguas.

Para las funciones de activación de las capas iniciales se utilizó relu y para la capa de salida se utilizó la función softmax. También se incluyeron algunas capas de MaxPoolin2D con strides(2,2) y el tamaño del kernel fue de 3 x 3.

Finalmente se utilizó una función de pérdida Sparse Categorical Crossentropy, ya que las variables salidas se serializaron mediante el método OrdinalEncoder de sklearn, y para la métrica se utilizó Sparse Categorical Accuracy.

Como librerías se utilizaron básicamente las utilidades de TensorFlow y Keras.

A continuación, se muestra el modelo base.

```
# Creamos el modelo 2
K.clear_session()
img_rows, img_cols = 128, 128
n_classes=9
input_shape = (img_rows, img_cols, 5)

model = Sequential()

model.add(Conv2D(filters = 16, kernel_size = (3, 3), activation='relu',
                 input_shape = input_shape))
model.add(BatchNormalization())
model.add(Conv2D(filters = 16, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(strides=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(strides=(2,2)))
model.add(Dropout(0.25))

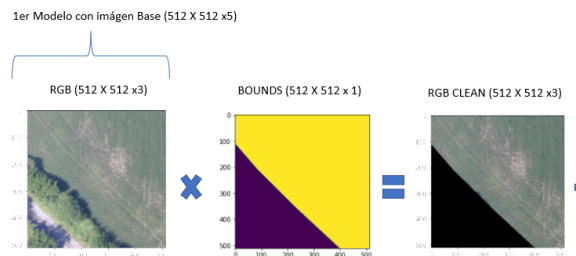
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.25))

model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(n_classes, activation='softmax'))
```

Figure 8: Modelo Base

Para el uso del modelo se intentó crear una línea base con la información correspondiente a las imágenes RGB de tres canales. Un segundo experimento se llevó a cabo incrementando la información suministrada al modelo con la introducción de más características y combinación de las mismas.

En específico, las imágenes RGB (250x250x3) y NIR (250x250x1) fueron corregidas con las respectivas imágenes de bounds (250x250x1) que incluyen la discriminación de las partes de las fotografías que no corresponden al cultivo de estudio, vg, carreteras, lagos otros predios etc. Estas dos fotografías se juntaron adicionando a la primera un cuarto canal con la información de NIR y, a el resultado, se le agregó un canal más con la información de las fotografías de Labels que incluían las áreas de afectación en los predios marcadas por los agrónomos. Un esquema de esta consolidación es el siguiente



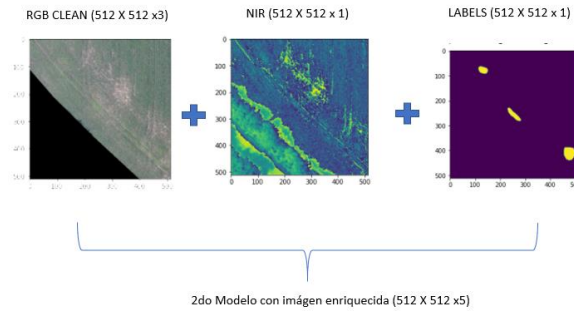


Figure 9: Modificando los canales de color

Finalmente se hizo un corrimiento de ambos modelos, el de información básica y el del array con más información y canales. Ese par de experimento corrió con las mismas condiciones de modelo y se ejecutó para ambos, primero con 20,545 fotografías y luego con 50,000 fotografías, teniendo en cuenta un Split de 80-20 % para entrenamiento y validación.

En el corrimiento con 20,545 fotografías, se utilizaron 6 épocas y se encontró, contrario a lo que inicialmente se esperaba, que el modelo con la información más básica de sólo RGB dio un accuracy de 0,46 en validación frente a 0.40 en la correspondiente prueba con más información. Es posible que la eliminación de parte de las fotografías que no correspondan al predio le incluya información adicional difícil de procesar con pocas fotografías y que con más épocas o más instancias pueda mejorar. Para ambos corrimientos el accuracy de entrenamiento estuvo en rangos de valores que no indicaban sobreajuste, cercanos a los 0.4.

```
# Fitteo del modelo
batch = 30
val_batch = 30
ep = 6
steps_train = math.ceil(len(train_filenames)/batch)
steps_val = math.ceil(len(val_filenames)/batch)

AgriviewModel = model.fit(s3_image_generator(train_filenames), epochs=6, batch_size= 30, steps_per_epoch=steps_train, validation_steps=steps_val)
```

Epoch 1/6  
514/514 [=====] - 146s 253ms/step - loss: 4.7678 - sparse\_categorical\_accuracy: 0.3346 - val\_loss: 4.0343 - val\_sparse\_categorical\_accuracy: 0.2006  
Epoch 2/6  
514/514 [=====] - 114s 221ms/step - loss: 1.7500 - sparse\_categorical\_accuracy: 0.3979 - val\_loss: 1.6593 - val\_sparse\_categorical\_accuracy: 0.4277  
Epoch 3/6  
514/514 [=====] - 109s 212ms/step - loss: 1.7523 - sparse\_categorical\_accuracy: 0.3635 - val\_loss: 1.8443 - val\_sparse\_categorical\_accuracy: 0.3450  
Epoch 4/6  
514/514 [=====] - 107s 208ms/step - loss: 1.6999 - sparse\_categorical\_accuracy: 0.4080 - val\_loss: 3.3190 - val\_sparse\_categorical\_accuracy: 0.3603  
Epoch 5/6  
514/514 [=====] - 106s 206ms/step - loss: 1.6033 - sparse\_categorical\_accuracy: 0.4391 - val\_loss: 1.7427 - val\_sparse\_categorical\_accuracy: 0.4337  
Epoch 6/6  
514/514 [=====] - 105s 205ms/step - loss: 1.5071 - sparse\_categorical\_accuracy: 0.4550 - val\_loss: 1.5535 - val\_sparse\_categorical\_accuracy: 0.4634

Figure 10: Modelo con 20454 fotografías con sólo la imagen RGB

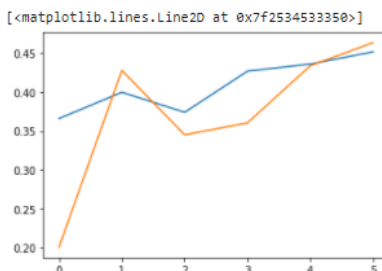


Figure 11: Acuracy Entrenamiento (Azul) vs Validación(naranja) imagen RGB

```
[79] # Fitteo del modelo
batch = 30
val_batch = 30
ep = 6
steps_train = math.ceil(len(train_filenames)/batch)
steps_val = math.ceil(len(val_filenames)/batch)

AgriviewModel = model.fit(s3_image_generator(train_filenames), epochs=6, batch_size= 30, steps_per_epoch=steps_train, validation_steps=steps_val)

Epoch 1/6
514/514 [=====] - 543s 15/step - loss: 4.4309 - sparse_categorical_accuracy: 0.3245 - val_loss: 2.5090 - val_sparse_categorical_accuracy: 0.3498
Epoch 2/6
514/514 [=====] - 527s 15/step - loss: 1.7520 - sparse_categorical_accuracy: 0.3429 - val_loss: 2.0526 - val_sparse_categorical_accuracy: 0.3911
Epoch 3/6
514/514 [=====] - 524s 15/step - loss: 1.7664 - sparse_categorical_accuracy: 0.3246 - val_loss: 2.2984 - val_sparse_categorical_accuracy: 0.3612
Epoch 4/6
514/514 [=====] - 524s 15/step - loss: 1.7678 - sparse_categorical_accuracy: 0.3356 - val_loss: 1.7148 - val_sparse_categorical_accuracy: 0.4295
Epoch 5/6
514/514 [=====] - 526s 15/step - loss: 1.7657 - sparse_categorical_accuracy: 0.3310 - val_loss: 1.6835 - val_sparse_categorical_accuracy: 0.4273
Epoch 6/6
514/514 [=====] - 524s 15/step - loss: 1.7125 - sparse_categorical_accuracy: 0.3677 - val_loss: 1.7213 - val_sparse_categorical_accuracy: 0.4082
```

Figure 12: Modelo para 20,545 fotografías con información consolidada de varias características

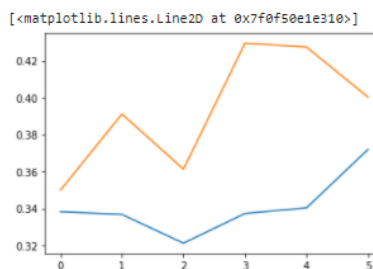


Figure 13: Modelo para 20,545 fotografías con información consolidada de varias características

Desafortunadamente, el corrimiento del modelo con más fotografías redujo el accuracy de validación a niveles de 0,3, como se muestra a continuación, por lo que se optó por incluir otras arquitecturas más potentes que se explican enseguida.

```
# Fitteo del modelo
batch = 30
val_batch = 30
ep = 6
steps_train = math.ceil(len(train_filenames)/batch)
steps_val = math.ceil(len(val_filenames)/batch)

AgriviewModel = model.fit(s3_image_generator(train_filenames), epochs=6, batch_size= 30, steps_per_epoch=steps_train, validation_steps=steps_val)

Epoch 1/6
514/514 [=====] - 146s 253ms/step - loss: 4.7678 - sparse_categorical_accuracy: 0.3346 - val_loss: 4.0343 - val_sparse_categorical_accuracy: 0.2006
Epoch 2/6
514/514 [=====] - 114s 221ms/step - loss: 1.7500 - sparse_categorical_accuracy: 0.3979 - val_loss: 1.6593 - val_sparse_categorical_accuracy: 0.4277
Epoch 3/6
514/514 [=====] - 109s 212ms/step - loss: 1.7523 - sparse_categorical_accuracy: 0.3635 - val_loss: 1.8443 - val_sparse_categorical_accuracy: 0.3450
Epoch 4/6
514/514 [=====] - 107s 208ms/step - loss: 1.6999 - sparse_categorical_accuracy: 0.4080 - val_loss: 3.3190 - val_sparse_categorical_accuracy: 0.3603
Epoch 5/6
514/514 [=====] - 106s 206ms/step - loss: 1.6033 - sparse_categorical_accuracy: 0.4391 - val_loss: 1.7427 - val_sparse_categorical_accuracy: 0.4337
Epoch 6/6
514/514 [=====] - 105s 205ms/step - loss: 1.5071 - sparse_categorical_accuracy: 0.4550 - val_loss: 1.5535 - val_sparse_categorical_accuracy: 0.4634
```

Figure 14: Modelo con 50,000 fotografías con sólo la imagen RGB

```
[79] # Fiteo del modelo
batch = 30
val_batch = 30
ep = 6
steps_train = math.ceil(len(train_filenames)/batch)
steps_val = math.ceil(len(val_filenames)/batch)

AgriviewModel = model.fit(s3_image_generator(train_filenames), epochs=6, batch_size= 30, steps_per_epoch=steps_train, validati

Epoch 1/6
514/514 [=====] - 543s 1s/step - loss: 4.4309 - sparse_categorical_accuracy: 0.3245 - val_loss: 2.5090 - val_sparse_categorical_accuracy: 0.3498
Epoch 2/6
514/514 [=====] - 527s 1s/step - loss: 1.7520 - sparse_categorical_accuracy: 0.3429 - val_loss: 2.0526 - val_sparse_categorical_accuracy: 0.3911
Epoch 3/6
514/514 [=====] - 524s 1s/step - loss: 1.7664 - sparse_categorical_accuracy: 0.3246 - val_loss: 2.2984 - val_sparse_categorical_accuracy: 0.3612
Epoch 4/6
514/514 [=====] - 524s 1s/step - loss: 1.7678 - sparse_categorical_accuracy: 0.3356 - val_loss: 1.7148 - val_sparse_categorical_accuracy: 0.4295
Epoch 5/6
514/514 [=====] - 526s 1s/step - loss: 1.7657 - sparse_categorical_accuracy: 0.3310 - val_loss: 1.6835 - val_sparse_categorical_accuracy: 0.4273
Epoch 6/6
514/514 [=====] - 524s 1s/step - loss: 1.7125 - sparse_categorical_accuracy: 0.3677 - val_loss: 1.7213 - val_sparse_categorical_accuracy: 0.4002
```

Figure 15: Modelo para 50,000 fotografías con información consolidada de varias características

### **Transfer learning:**

En la construcción de modelos de Deep learning con buen desempeño estos modelos deben tener dos características importantes una estructura robusta y el dataset que utilicemos sea representativo, en la práctica estas dos situaciones no se pueden conseguir fácilmente. Cuando esto ocurre una de las técnicas que se puede utilizar para mitigar este efecto es usar lo que llamamos transfer learning, esto se refiere a tomar estructuras de modelos de Deep learning creados y entrenados y usarlos para aplicaciones o reconocimientos diferentes.

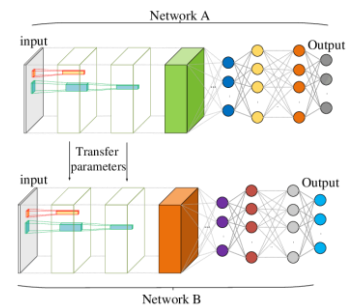


Figure 16: Transfer learning

En el presente proyecto se implementó la red resnet50 donde se aprovechamos el conocimiento adquirido previo a su entrenamiento. ResNet es una red usada como elemento básico en muchas tareas de visión por computadora, fue la red ganadora de la competencia (challenge) ImageNet en 2015, donde, su mayor impacto se debe a que el paradigma ResNet permitió por primera vez entrenar redes muy profundas (de más de 100 capas).

En la implementación de resnet50 se modificó la capa de entrada y la de salida ya que el tamaño de las imágenes eran de 512 x 512 pixeles y las clases a clasificar eran 9, posteriormente, cargamos el modelo resnet50 indicándole que excluyera la capa top y cargara los pesos de entrenamiento del dataset de imagenet, así mismo, utilizamos dos técnicas de regularización dropout y batch standardization.

En la primera y segunda iteración le indicamos al modelo que no utilizara los pesos de entrenamiento del dataset de imagenet en las últimas 10 capas de la arquitectura, variando de 50 y 25 épocas de entrenamiento del modelo.

El resultado no fue el esperado ya que se presentó un sobre ajuste en el modelo y un accuracy muy bajo en su validación.





Figure 17: Iteraciones ResNet50

En la tercera y cuarta iteración le indicamos al modelo que utilizara todos los pesos de entrenamiento donde se evidencia que mejoró el sobre ajuste del modelo, pero el accuracy de validación solo logramos llegar a un 40 % en la predicción.

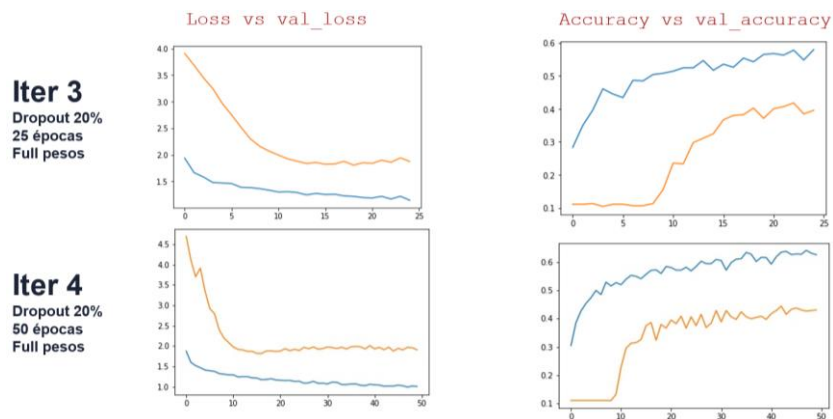


Figure 18: Iteraciones ResNet50

A continuación, mostramos una de las predicciones del modelo correspondiente a la 4 iteración.

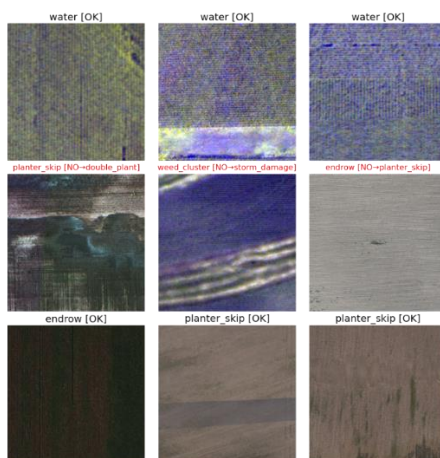


Figure 19: Resultados Predicción

También utilizamos la red Mobilenet donde no realizamos transfer learning ya que esta se basa en una arquitectura racionalizada que utiliza convoluciones separables en profundidad para construir redes neuronales profundas de poco peso.

Al entrenar el modelo con la arquitectura de Mobilenet nos dimos cuenta de que los datos se ajustaron más al modelo evitando sobre ajuste y accuracy por encima del 60 % de predicción, esto se debe al tipo de imágenes con características especiales utilizadas en el entrenamiento del modelo.

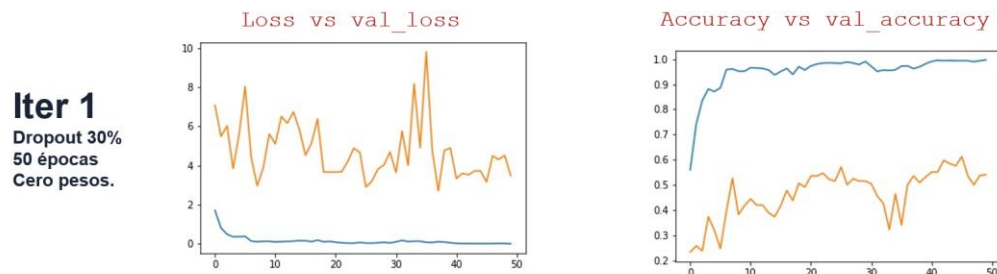


Figure 20: Iteración MobileNet

A continuación, mostramos una de las predicciones del modelo de Mobilenet correspondiente a la 1 iteración.

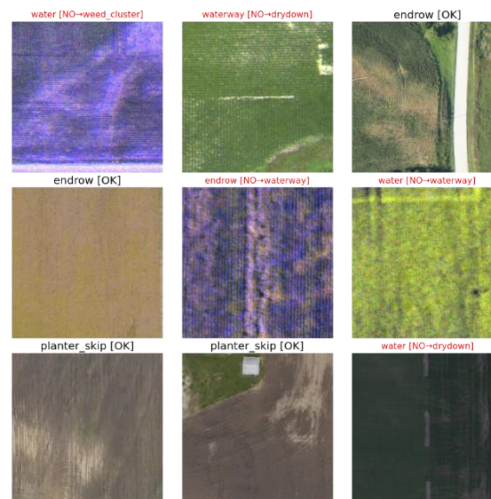


Figure 21: Predicción MobileNet

## Entregables y su descripción

Nuestro proyecto puede ser consultado en el siguiente repositorio de GitHub:

<https://github.com/chucho111/Estadistica-Multivariada.git>

## Conclusiones

1. Dentro de la regresión logística se puede ver que el escalamiento de las imágenes ayuda a aumentare el accuracy en un 8%.
2. Modelo de Random Forest aprende muy bien los datos de entrenamiento, generando overfitting por lo cual se necesitan implementar diferentes estrategias para evitarlo.
3. Logistic Regression y Random Forest no es buen modelo para clasificar este tipo de imágenes.
4. Machine Learning puede dar una mayor precisión que los humanos para imágenes que son bastantes complejas y con características que no son fácilmente diferenciables a simple vista.
5. En nuestro caso fue mejor la red MobileNet sin conocimiento adquirido que la ResNet50 con conocimiento, esto es debido al tipo de imágenes con características especiales utilizadas en el entrenamiento del modelo.
6. El clasificador de regresión logística definitivamente no alcanzó métricas adecuadas de predicción, pero los clasificadores de SVM y CNN con la arquitectura de complejidad intermedia si estuvieron similares en accuracy de validación. Sin embargo, arquitecturas mas robustas como las utilizadas de ResNet50 y MobilNet si definitivamente superaron por mucho a los otros dos clasificadores.
7. La inclusión de más información añadiendo canales a los arreglos de la fotografía base RGB, no generó mejoras en las métricas. Se debe dejar de lado esta estrategia y mejor incluirla haciendo combinación de modelos; es decir, al modelo base agregar unos brazos (branches) con modelos incluyendo la información de los límites (bounds), la información de NIR y la de segmentos afectados (labels), y combinarlos.

## Trabajo futuro

1. Un paso para seguir es incluir toda la información combinando modelos. Es decir, agregar un modelo con una característica (por ejemplo, el nir, o los límites) y combinarlos en el final de las capas.
2. Por tiempo de cómputo y recursos limitados faltó ejecutar más modelamientos con diferentes alternativas, vg, quedó pendiente ejecutar k-fold para tratar de disminuir la varianza del modelo y hacer una mejor medición del error de generalización.
3. Sería importante determinar si el uso de imágenes de satélite podría dar resultados similares o la disparidad de resoluciones disminuye mucho el poder de predicción.
4. También sería importante mejora el proceso de aprendizaje del modelo con más imágenes.

## Ejecución del plan

Dentro del anteproyecto inicial se planeó una ejecución del plan el cual se ejecutó de la siguiente manera:

Plan Propuesto	Plan Real - Comentarios	Lecciones Aprendidas
Planeamiento del problema	Esta fase se realizo en el tiempo establecido, sin embargo, se reviso de una manera superficial el Dataset con el fin de conocer cuál es la problemática inicial.	
Exploración del Dataset	Esta fase tuvo un retraso de dos semanas debido a que se debió investigar sobre diferentes	Tener o contactar un SME con el fin de investigar las herramientas

	herramientas que nos ayudaran con la visualización de imágenes.	necesarias para la visualización de imágenes para machine learning
Investigación de clasificadores de imágenes	Esta fase fue la que tomo tiempo, ya que tuvimos que realizar muchas investigaciones de cómo utilizar los clasificadores con imágenes. También se evidencio la complejidad de redes neuronales, lo que nos llevó a buscar diferentes redes como ResNet50 y MobileNet. Adicional se tuvo que investigar como se debe utilizar Keras para el procesamiento de imágenes y para la clasificación con Deep learning.	Es importante contar con una persona que conozco sobre Deep Learning, ya que debido a la complejidad que tiene esta metodología no pudimos finalizar el plan inicial el cual contaba con una segmentación semántica.
Entrenamiento de Modelos	Dentro de esta fase existieron diferentes inconvenientes, como el hecho de implementar un algoritmo y que este no funcionara para el tipo de Dataset que se adquirió, por lo que se debía replantear los clasificadores que se usarían en el proyecto. Esto llevo a que volviéramos a la fase de investigación con el fin de poder traer un algoritmo mas óptimo para el problema.	El tener un conocimiento sobre el procesamiento de imágenes hubiera ayudado a la selección de mejores clasificadores para el problema.
Evaluación de modelos	Dentro de la evaluación de los modelos se pudo evidenciar que se requerían implementar diferentes estrategias que ayudaran a la mejora de las métricas de evaluación de los algoritmos.	Es importante contar dentro del cronograma con tiempos de posibles iteraciones futuras con el fin de poder tener un manejo del tiempo óptimo.

## Implicaciones éticas

La aplicación de tecnologías en el ambiente de la agricultura puede dar riesgos socio éticos como es la dependencia de la tecnología, monocultivo, aumento en la brecha digital, posible concentración y manipulación de datos, incluido la dependencia por parte de los agricultores de insumos externos suministrados por proveedores de tecnologías, amenaza contra la sostenibilidad de los pequeños agricultores locales, control y prácticas desleales.

Entre los principales riesgos éticos asociados en la inclusión de tecnologías en el ámbito de la agricultura esta la seguridad alimentaria y las consideraciones ecológicas, desarrollo sostenible, la necesidad de tener en cuenta las necesidades de las generaciones futuras, cambios sociales que se puedan desencadenar por la adopción nuevas tecnologías, la brecha social entre pequeños y grandes agricultores, tanto dentro del país como fuera sobre todo en los países en vía de desarrollo dado el costo de incluir estas tecnologías, el equilibrio entre el aumento de la productividad y la eficiencia para la agricultura tradicional y la sostenibilidad del medio ambiente.

## Aspectos legales y comerciales

Lo primero a tener en cuenta para poder realizar este proyecto son los términos de uso especificados para el uso del Dataset Agriculture-vision, en el cual se especifica que el conjunto de datos puede ser descargado y utilizado siempre y cuando no se utilice con fines comerciales. Tampoco se puede vender, transferir o licenciar el Dataset a terceros (excepto a colegas de investigación), ni crear ningún trabajo derivado del Conjunto de Datos que no sea el desarrollo de aplicaciones de investigación no comerciales

Gracias a los drones en la agricultura, se puede verificar las características de las plantaciones a través del análisis de imágenes multiespectrales, donde se representan niveles de reflectancia de luz obteniendo mapas de colores que indican las diferentes características de los cultivos y que permiten:

1. Explorar los campos en menos tiempo.
2. Tomar decisiones por medio de los mapas de color generados por los drones.
3. Conocer si el campo a mejorado o desmejorado desde su última revisión.

Se conoce que el sector agropecuario ha tenido un gran crecimiento en Colombia donde contribuye con el 6.8% del PIB, por lo que el uso de drones para detectar anomalías en las características fundamentales que puedan afectar el producto final es crucial en los mercados de hoy en día, permitiendo así mejorar las prácticas de manejo del cultivo y dando la posibilidad de tener reacciones rápidas que eviten pérdidas y mejoren productividad.

## Bibliografía

1. Brownie J (2020). *How to Identify Overfitting Machine Learning Models in Scikit-Learn* [Article]. Recuperado de: <https://machinelearningmastery.com/overfitting-machine-learning-models/>
2. Asaithambi S (2017). *Why, How and When to Scale your Features* [Article]. Recuperado de: <https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e>
3. Brownie J (2021). *Multinomial Logistic Regression With Python* [Article]. Recuperado de: <https://machinelearningmastery.com/multinomial-logistic-regression-with-python/>
4. Nasrin R (2020). *Multiclass Classification Using Logistic Regression* [Article]. Recuperado de: <https://towardsdatascience.com/multiclass-classification-algorithm-from-scratch-with-a-project-in-python-step-by-step-guide-485a83c79992>
5. Lin Y, Xiangzheng D & Xing L (2014). *Comparison of multinomial logistic regression and logistic regression: which is more efficient in allocating land use?* [ Paper]
6. Bogotobogo (2020). *Logistic Regression, Overfitting and Regularización* [Article]. Recuperado de: [https://www.bogotobogo.com/python/scikit-learn/scikit-learn\\_logistic\\_regression.php](https://www.bogotobogo.com/python/scikit-learn/scikit-learn_logistic_regression.php)
7. Rahman F (2020). *Logistic Regression for Image Classification* [ Article]. Recuperado de: <https://medium.com/swlh/logistic-regression-for-image-classification-e15d0ae59ce9>
8. Haralick R. M., Shanmugam K. & Dinstein. (1973) "Textural Features for Image Classification" *IEEE Transaction on Systems, Man, And Cybernetics*. U.S.A.
9. Rivera M (2019). *La Red Residual (Residual Network ResNet)* [Artículo]. Recuperado de: [http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje\\_profundo/resnet/resnet.html#:~:text=\(2015\),tareas%20de%20Visi%C3%B3n%20por%20Computadora.&text=Su%20mayor%20impacto%20se%20debe,de%20Gradiente%20\(Vanishing%20Gradient\).](http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/resnet/resnet.html#:~:text=(2015),tareas%20de%20Visi%C3%B3n%20por%20Computadora.&text=Su%20mayor%20impacto%20se%20debe,de%20Gradiente%20(Vanishing%20Gradient).)
10. Dwivedi P (2019). *Understanding and coding a ResNet in Keras* [Artículo]. Recuperado de: <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>
11. Keras (2021). *ResNet and ResNetV2* [ Documentación] . Recuperado de: <https://keras.io/api/applications/resnet/>
12. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Application*
13. Abhijeet Pujara (2020). *Imagen Classifier with MobileNet*[Article]. Recuperado de: <https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470>
14. Fawagreh K (2014). *Random forests: from early developments to recent advancements* [Article]. Recuperado de: <https://www.tandfonline.com/doi/full/10.1080/21642583.2014.956265>