**Proyecto I – Detección de Frailejones, Deep Learning**

Daniel Felipe Rodríguez Cuellara,c, Diana Katherin Ibáñez Contreras a,c, Erika Andrea Cárdenas Rodríguez a,c, Paola Andrea Castro Cardozoa,c y Oscar Eduardo Correcha Guzmán b,c

Sergio Alberto Mora Pardo b,c

aEstudiante de Maestría en Analítica para la Inteligencia de Negocios

bEstudiante de Maestría en Inteligencia Artificial

bProfesor, Facultad de Ingeniería

cPontificia Universidad Javeriana, Bogotá, Colombia

# Entendimiento del Negocio

El frailejón (Espeletia) es una planta nativa de los páramos andinos y su presencia es de vital importancia para manetener la biodiversidad, regular el agua y preservar el suelo. Actuán como esponjas naturales que capturan y almacenan el agua, asegurando el suministro constante para las comunidades locales, incluso durante épocas de sequía. Algunas medidas para su preservación se enfocan en evitar la deforestación, la expansión agrícola, los incendios forestales e invertir en proyectos para replobar y restaurar áreas degradadas. También, educando y sensibilizando a la población sobre su importancia y cuidado. Sin embargo, las proyectos de restauración requieren esfuerzos prolongados y sostenidos, ya que el crecimiento del frailejón por las condiciones adversas de su hábitat, es muy lento y se requieren muchos años para que alcance su tamaño adulto. En contraprestación, son plantas muy longevas.

Los páramos de Chingaza y Cruz Verde tienen una extensión aproximada de 93.000 héctareas (en muchos casos de difícil acceso), por lo que se plantea el presente caso de estudio de analítica de datos, en aras de facilitar y enriquecer el trabajo ambiental requerido.

## Objetivos de Negócio

1. Identificar y calcular automáticamente frailejones en fotografías aéreas de los páramos de Chingaza y Cruz Verde, para apoyar el monitoreo y estudio permanente de estas plantas.
2. Calcular la densidad poblacional de los frailejones en las fotografías, como apoyo en la toma de decisiones y definición de estrategias para su preservación.

### Criterios de éxito de los objetivos de negocio

Prueba cualitativa mediante comparación de la fotografía proporcionada para este estudio y la gráfica que simula la "latitud y longitud" de ubicación de cada frailejón.

## Objetivos de Minería de Datos

1. Construir diferentes modelos de redes neuronales para clasificación de imágenes (*Simple, Multicapa, Convolucional, convolucional con transferencia VGG-16, convolucional con transferencia RestNet50*) para la detección de frailejones en la fotografía proporcionada.
2. Explorar el modelo de red neuronal *convolucional con transferencia VGG-16* aplicando el marco de trabajo para la optimización de hiperparámetros *Optuna*, buscando maximizar el *AUC* resultante y comparar resultados con el mismo modelo sin optimizar.
3. Aplicar la técnica estadística *Gaussian Mixture Model (GMM)* en el análisis de densidad poblacional de los frailejones.

### Criterios de éxito de los objetivos de minería de datos

Seleccionar el modelo que mejor rendimiento demuestre para distinguir entre dos clases: *frailejón* y *no frailejón*, según la métrica de *AUC*, sin generar sobreajuste (*Overfitting)* o subajuste (*Underffiting)*; por lo cual, se escogerá aquel en el que los *AUCs* tanto en entrenamiento como en prueba sean cercanos a uno y no distantes entre sí. *AUC* objetivo del 0,99%.

# Entendimiento de los Datos

Se ha dispuesto una fotografía aérea a color en el archivo *IMG\_3451.jpg,* en la url de *GitHub*: <https://github.com/sergiomora03/AdvancedTopicsAnalytics/blob/main/notebooks/img/IMG_3451.JPG>

El archivo tiene un tamaño de 18,57 Mb. La imagen tiene dimensiones de 4000x3000 píxeles y está en escala de color *RGB*. A simple vista se evidencian los frailejones junto a otras plantas e incluso a dos personas y otros objetos.



Imagen . Fotografía aérea del páramo

# Preparación y Exploración de los Datos

También, se ha proporcionado el archivo *ImportImagenes.py*, en la url de *GitHub*:

<https://raw.githubusercontent.com/sergiomora03/AdvancedTopicsAnalytics/main/notebooks/img/ImportImagenesURL.py>. Este archivo contiene código *Phyton* que carga y retorna los datos preprocesados del archivo binario *data\_F,* de esta manera se obtiene el conjunto de datos X y Y, donde X es la imagen recortada para procesar de la fotografía y Y es la etiqueta, una variable binaria que indica si la imagen es *frailejón* (1) o *no frailejón* (0).

Devuelve X con las dimensiones iguales a (14700, 250), donde 14700 es el número de variables o número de píxeles de la imagen recortada y 250 es el número de muestras o imágenes recortadas obtenidas. Verificando el código del archivo y graficando algunas de las imágenes recortadas que retorna, todas ellas tienen una dimensión de 70x70 píxeles (lo que coincide con los 14700 píxeles que se tienen, al aplanar la imagen en un vector unidimensional), por lo que no es necesario transformar las imágenes para estandarizar sus dimensiones para el modelado.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza mediaImagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen . Imágenes recortadas de 70x70

Y tiene las dimensiones iguales a (1, 250), donde 1 es la columna con las etiquetas de *frailejón* o *no frailejón* en la imagen recortada y 250 es el número de muestras o imágenes recortadas obtenidas.

Explorando los primeros elementos, en X se evidencian los 14700 números que representan los píxeles por imagen en un arreglo y, en Y se evidencian las 250 etiquetas de las imágenes recortadas, apiladas en una matriz vertical.

Imagen que contiene Tabla

Descripción generada automáticamente

Imagen . Primeros 5 elementos en X y Y

También, en el código del archivo ya se han normalizado los datos dividiendo por 255. De este modo, este archivo retorna los arreglos de X con las imágenes normalizadas y aplanadas y el arreglo de Y con las etiquetas.

Los elementos en Y permiten evidenciar que se tienen 105 etiquetas para *frailejón* y 145 para *no frailejón*. Esto corresponde con 42% de muestras para la clase que se quiere predecir *frailejón* y 58% de muestras para la clase *no frailejón*. Con esta diferencia entre clases del 16%, se concluye que no se requiere balancear, puesto que para problemas de clasificación se considera aceptable un desbalanceo inferior al 20%.

Se realiza la división del conjunto de datos, dejando el 70% como conjunto de entrenamiento (CE) y el 30% como conjunto de validación (CV). Para lo anterior, se hace necesario transponer X y Y para asegurar que las dimensiones de los datos estén de acuerdo con lo requerido, de la siguiente manera: X - características (n\_samples = 250, n\_features =14700) y Y - etiquetas (n\_samples = 250). Las dimensiones de los conjuntos resultantes son las siguientes:

Texto, Carta

Descripción generada automáticamente

Imagen . Dimensiones de conjuntos de datos de entrenamiento y validación

# Modelamiento y Evaluación

En la Tabla 1, se detallan los modelos de redes neuronales para clasificación de imágenes seleccionados para el modelamiento, junto con la mejor configuración de parámetros que se ha logrado y, los resultados obtenidos en la métrica de rendimiento escogida: *AUC* tanto en el conjunto de entrenamiento como en el conjunto de validación.

|  |  |  |  |
| --- | --- | --- | --- |
| MODELO | CONFIGURACIÓN | AUC Train | AUC Test |
| Simple | * 128 neuronas * Función de activación en capa de entrada: relu * Regularización en capa de entrada L2 de 0,01 * Dropout del 20% * Función de activación en capa de salida: sigmoid * Optimizador Adam con learning rate de 0.001 * 100 épocas * No se evidencia mejora con el kernel de inicialización HeNormal. | 0,942 | 0,883 |
| Simple (con Optuna) | * 286 neuronas * Función de activación en capa de entrada: relu * Regularización en capa de entrada L2 de 0,01 * Dropout del 20% * Función de activación en capa de salida: sigmoid * Optimizador Lion con learning rate de 0.00005 * 86 épocas   No se evidencia mejora con el kernel de inicialización HeNormal. | 0,998 | 0,978 |
| Multicapa | * Capa de entrada con 800 neuronas y función de activación relu * Dropout del 20% solo después de la capa de entrada * 2 capas ocultas con 500 y 200 neuronas y, función de activación relu * Función de activación en capa de salida: sigmoid * Optimizador Adam con learning rate de 0.001 * Cross validation con 5 splits * 100 épocas * No se evidencia mejora con el kernel de inicialización HeNormal, ni con regularización L2 | 0,970 | 0,967 |
| Multicapa (con Optuna) | * Capa de entrada con 670 neuronas y función de activación relu * Dropout del 20% solo después de la capa de entrada * 2 capas ocultas con 417 y 215 neuronas y, función de activación relu y tanh * Función de activación en capa de salida: sigmoid * Optimizador Adam con learning rate de 0.00005 * 27 épocas * No se evidencia mejora con el kernel de inicialización HeNormal, ni con regularización L2 | 0,926 | 0,861 |
| Convolucional | * La capa de entrada tiene 64 filtros, kernel = 7. * El modelo tiene 5 capas convolucionales intermedias, con kernel =3, las dos primeras con 128 filtros, y las últimas dos con 256 filtros * Se aplica padding=same. * Como iniciador de Kernel se utiliza “he\_normal” * 3 BatchNormalization(estabilizar) y 3 Max Pooling: (reducir las dimensiones espaciales y extraer características relevantes) * Global Average Pooling: antes de las capas densas para reducir las características espaciales globalmente. * Dropout con una ratio de 0.5 para evitar el sobreajuste. * Capa de salida: **"sigmoid"** para la clasificación binaria. * **100** épocas | 1 | 0.958 |
| Convolucional con transferencia VGG-16 | * Modelo Base: VGG-16 pre-entrenado en ImageNet * Incluye 13 capas convolucionales en 5 bloques * Después de cada bloque convolucional, aplica una capa de MaxPooling2D. * Eliminación (included\_top = False) de cabeza para excluir las capas densas finales del modelo original. * Capas de Salida:   + GlobalAveragePooling2D: Reduce la dimensión de la salida convolucional a un vector.   + Dense (1024): Capa densa intermedia con 1024 unidades y activación ‘relu’.   + Dense (1): Capa de salida con activación ‘sigmoide’ para clasificación binaria. * Optimizador Adam, learning rate: 0.001 * Función pérdida bynary\_crossentropy * Épocas: 100 * Batch size: 32 | 0,980 | 0,976 |
| Convolucional con transferencia VGG-16 (con Optuna) | * Inputlayer: Imágenes tamaño (224,224,3) * VGG-16 Base model   + Bloques capas Conv2D   + MaxPooling2D   + Include\_top = false * Global Average Pooling 2D * Dropout Optuna: 0.2 - 0.6 * Dense Layer Optuna: 128, 256, 512 * Best Model (Trial 1 de 5):   + Learning\_rate: 0.0022578633   + Dropout\_rate: 0.27271536514   + Dense Units: 512 | 0,933 | **0,979** |
| Convolucional con transferencia RestNet50 | * Modelo Base: ResNet50 pre-entrenado en ImageNet * Múltiples bloques residuales, cada uno con menos capas convolucionales. * Bloques convolucionales de diferentes tamaños. * Después de cada bloque convolucional, aplica una capa de MaxPooling2D. * Eliminación (included\_top = False) de cabeza para excluir las capas densas finales del modelo original. * Capas de Salida:   + GlobalAveragePooling2D: Reduce la dimensión de la salida convolucional a un vector.   + Dense (1024): Capa densa intermedia con 1024 unidades y activación ‘relu’.   + Dense (1): Capa de salida con activación ‘sigmoide’ para clasificación binaria. * Optimizador Adam * Función pérdida bynary\_crossentropy * Épocas: 15   + Batch size: 32 | 0.974 | 0.861 |
| Convolucional con transferencia RestNet50 (con Optuna) | * Inputlayer: Imágenes tamaño (224,224,3) * ResNet50 Base model   + Bloques capas Conv2D bloques residuales   + MaxPooling2D   + Include\_top = false * Best Model (Trial 1 de 5):   + Dense Units: 281   + Activación ‘relu’   + Optimizador Adam   + Learning\_rate: 0.04   + Batch size: 32 | 0.784 | 0.764 |

Tabla . Evaluación de resultados de los modelos implementados

Se implementaron modelos con redes pre-entrenadas VGG-16 y RestNet50. El aprendizaje por transferencia utilizando modelos pre-entrenados, es una técnica que se usa especialmente para tareas de visión por computadora. Estas redes fueron entrenadas en el conjunto de datos ImageNet, que contiene muchas imágenes clasificadas en miles de categorías. Durante este entrenamiento el modelo aprendió características como lo son bordes, texturas, formas y patrones, aplicables a una gran variedad de tareas. Con este tipo de redes, se aprovecha este conocimiento previo, evitando la necesidad de aprender desde cero. Adicionalmente se reduce el tiempo de entrenamiento, lo que representa eficiencia en los recursos computacionales.

En este proyecto de predicción de frailejones, al utilizar el modelo pre-entrenado VGG-16, reutilizando las capas convolucionales y demás características que incorpora, se logra un gran aporte a la tarea específica de contar y clasificar frailejones en la imagen proporcionada, al aprovechar patrones visuales previamente aprendidos. Al adaptar las capas superiores de VGG-16 para la nueva tarea, el modelo puede enfocarse en detalles específicos de los frailejones, facilitando así una predicción más precisa y rápida sin requerir un entrenamiento extenso desde cero.

Algunos de los modelos, también han sido probados con Optuna, un marco de trabajo para la optimización de hiperparámetros. Lo anterior, buscando maximizar el *AUC* resultante y comparar resultados con el mismo modelo sin optimizar.

Optuna se utiliza para optimizar los hiperpárametros buscando ajustar las variables críticas en el modelo, como lo son la tasa de aprendizaje, la tasa de abandono y el número de unidades en la capa densa para mejorar el rendimiento del modelo. Mediante un proceso de prueba y error, Optuna explora diversas combinaciones de estos hiperpárametros y evalúa el desempeño del modelo utilizando el área bajo la curva (AUC) como métrica principal. Esto permite encontrar la configuración más efectiva del modelo, maximizando su capacidad para clasificar correctamente las imágenes de frailejones y mejorando la precisión y robustez del sistema de predicción.

En el proyecto, se utilizan Optuna para optimizar tres hiperparámetros clave del modelo: la tasa de aprendizaje, la tasa de abandono y el número de unidades en la capa densa. Optuna ajusta la tasa de aprendizaje explorando valores en un rango logarítmico entre 0.000001 y 0.01, lo cual ayuda a encontrar la velocidad óptima para actualizar los pesos del modelo. La tasa de abandono (dropout) se ajusta dentro de un rango entre 0.2 y 0.6 para prevenir el sobreajuste, eliminando aleatoriamente ciertas neuronas durante el entrenamiento. Por último, el número de unidades en la capa densa se selecciona de entre 128, 256 o 512, permitiendo que Optuna determine la mejor configuración que logre equilibrar la capacidad de aprendizaje y la complejidad del modelo. Estos ajustes combinados buscan maximizar el rendimiento del modelo en la predicción de frailejones.

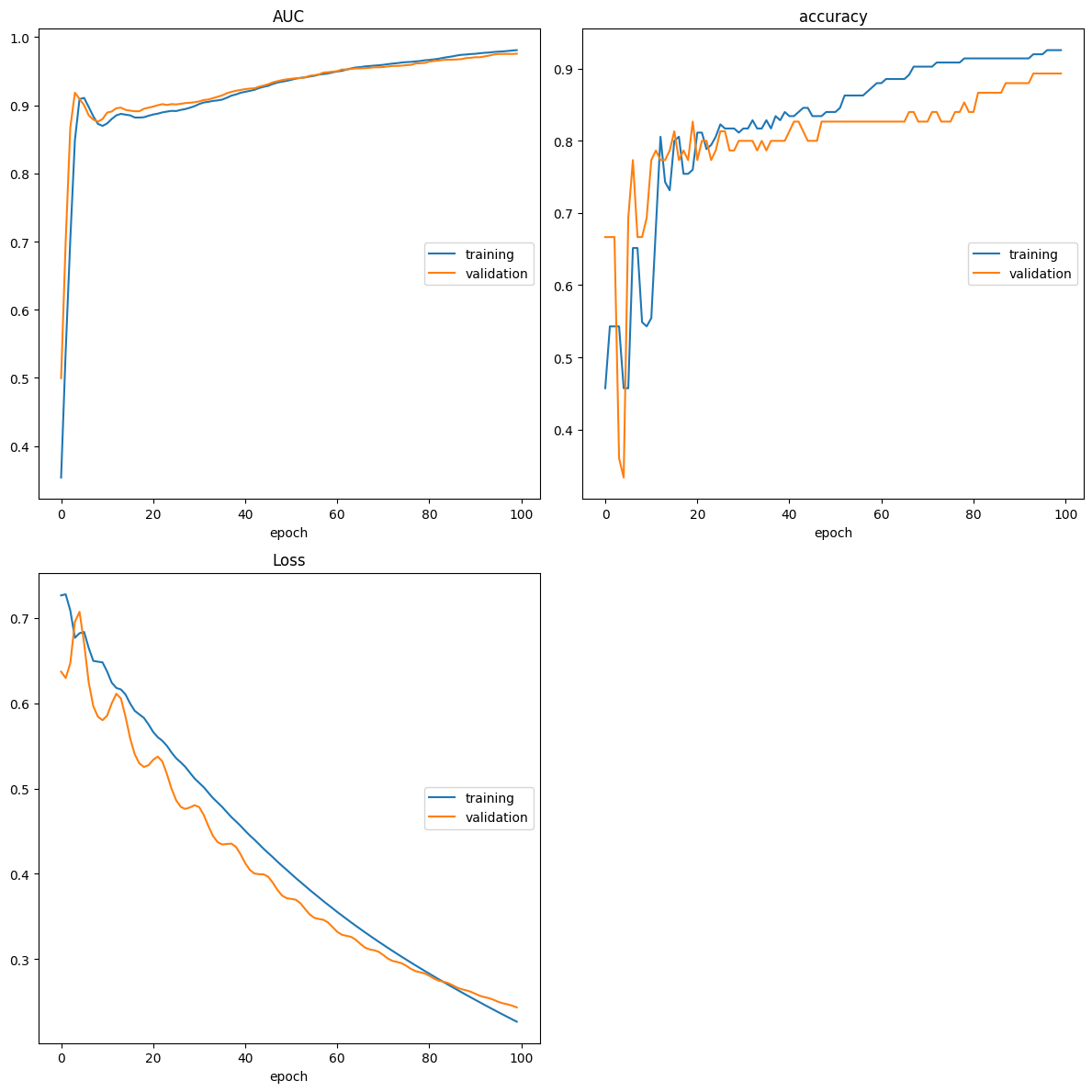


Imagen . AUC de Modelo CNN con transformación VGG-16

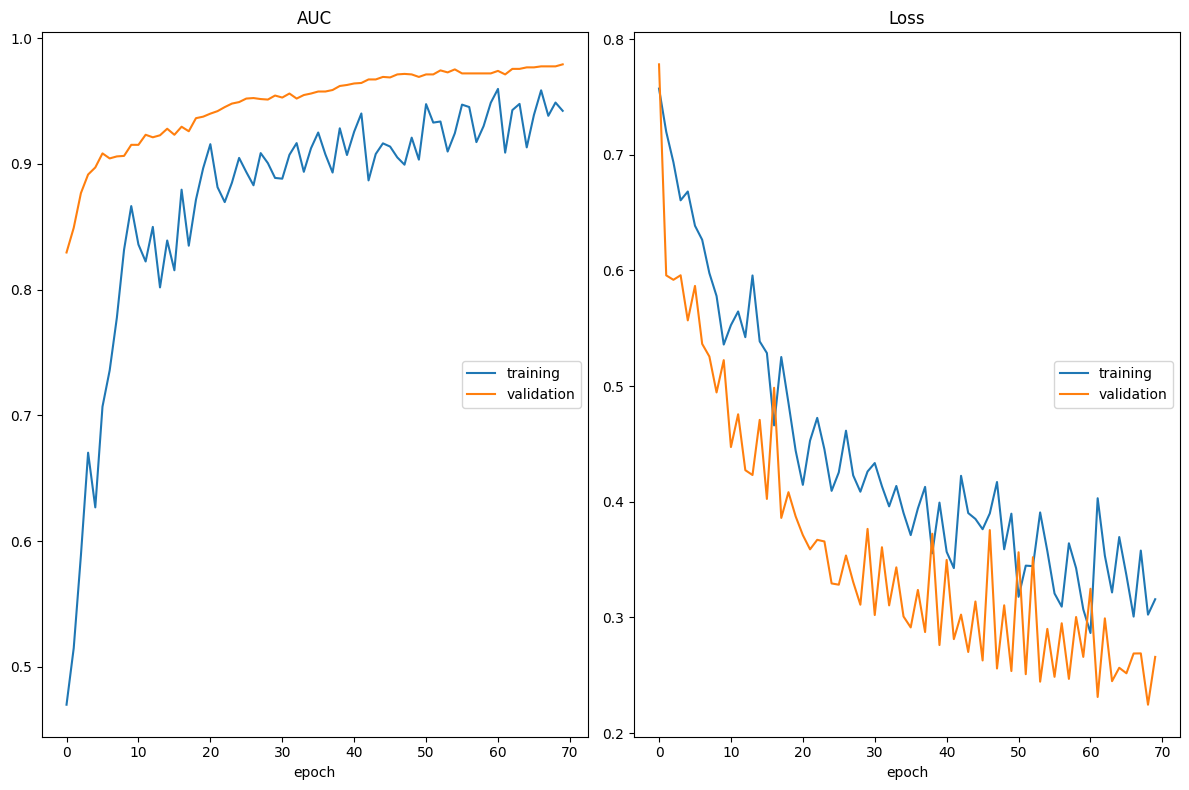
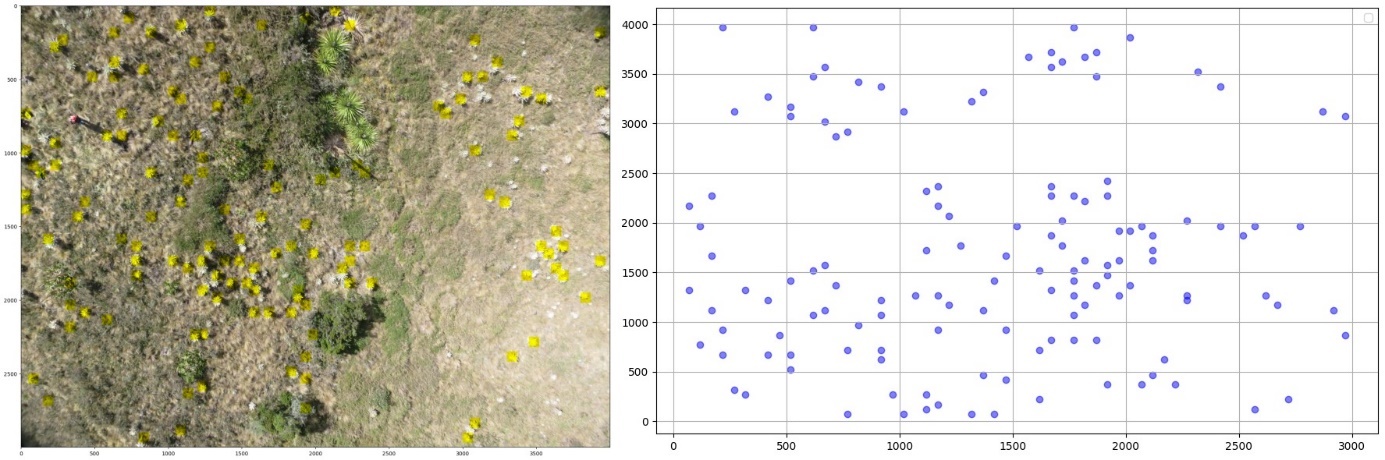


Imagen . AUC de modelo VGG-19 con Optuna

# Análisis cualitativo

Adicional a las métricas del mejor modelo, se realiza el siguiente análisis cualitativo entre la fotografía inicial y la gráfica generada que simula la "latitud y longitud" de ubicación de cada frailejón:



La imagen original y la predicción contienen tres canales correspondientes a los colores RGB, con dimensiones de 3000 píxeles de alto y 4000 píxeles de ancho. Los colores permiten identificar características importantes como texturas y niveles de profundidad en el área analizada.

Al comparar la imagen generada por el modelo con la original, se puede observar cómo el modelo resalta la presencia de frailejones, representados en color amarillo. Además, el modelo ha sido efectivo al excluir de la detección elementos, como personas, rocas o plantas diferentes al objetivo. También se destaca que la precisión del modelo es mayor en áreas donde predominan píxeles verdes, lo que sugiere que el modelo identifica con mayor acierto los frailejones en zonas vegetales.

En cuanto a la gráfica de dispersión, se observa una concentración notable de frailejones en las coordenadas X: 1000-2000 y Y: 500-2500, lo cual coincide con la distribución mostrada en la imagen original. Sin embargo, la gráfica no detecta frailejones en las zonas cercanas a los bordes, lo que indica una limitación en la precisión en esas áreas.

**Conclusión**

El análisis cualitativo de la distribución de frailejones en el terreno del caso de estudio muestra que esta no es homogénea. Esto sugiere que podrían existir condiciones ambientales específicas, como la altura o el tipo de suelo, que favorecen la presencia de estas plantas en determinadas áreas.

En las zonas rocosas no se observa una concentración de frailejones, lo cual puede representar un reto para la siembra en dichas áreas y el aprovechamiento del terreno. Finalmente, la comparación entre la gráfica y la imagen ha permitido identificar zonas donde no hay frailejones, lo que podría indicar áreas potenciales para futuras siembras y sostenimiento ambiental.

# Cálculo de densidad poblacional

Para el cálculo de la densidad poblacional de los frailejones en la fotografía, se ha aplicado la técnica estadística *Gaussian Mixture Model (GMM).*

Respecto a la técnica estadística consiste en considerar que existen varias relaciones de subpoblaciones diferentes las cuales se modelan con distribuciones normales o gaussianas, lo cual permite considerar que la subpoblación corresponda a una densidad o agrupamiento especifico. Matemáticamente contiene 3 características; como lo son la media, la varianza y el peso (probabilidad).

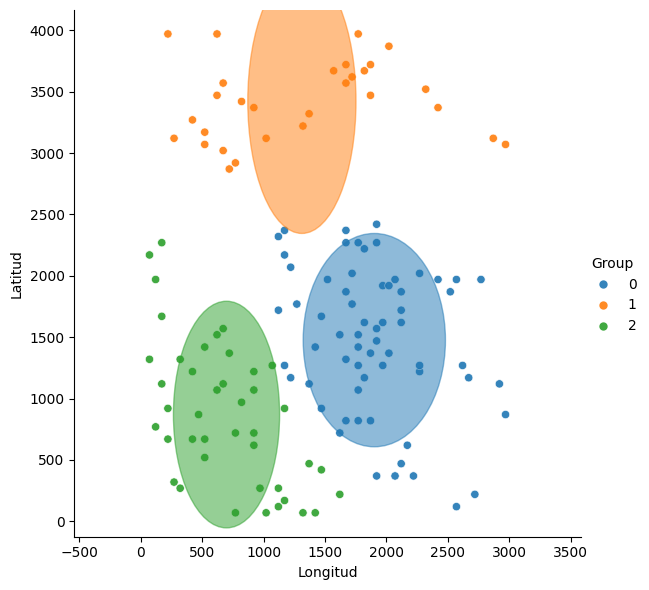
El método asigna puntos de datos en los cuales se crean calculando las probabilidades de que un punto pertenezca a una clase gaussiana y asigne el punto al que tenga mayor probabilidad.

En el caso de estudio encontramos los siguientes resultados:

|  |  |
| --- | --- |
| Grupo | Unidades |
| 0 | 64 |
| 1 | 29 |
| 2 | 42 |

Gráfico, Gráfico de líneas

Descripción generada automáticamente



Para identificar agrupaciones geográficas en nuestros datos, empleamos el algoritmo de **Mezclas Gaussianas (GMM)**. Utilizando la longitud y latitud de cada punto, y aplicando el método del codo para determinar el número óptimo de clusters, identificamos **tres grupos principales**. Estos grupos contienen, respectivamente, 69, 29 y 42 elementos.

A diferencia de K-means, que asume clusters esféricos, **GMM permite modelar clusters con formas más elípticas**, al considerar la varianza en cada dimensión. Esto se traduce en una mayor flexibilidad para capturar patrones complejos en los datos. En la visualización de los resultados, observamos que los clusters no son círculos perfectos, sino elipses de diferentes tamaños y orientaciones, lo que refleja la variabilidad inherente a los datos geográficos.

**La ventaja de utilizar GMM en este caso es que nos permite:**

* **Capturar patrones más sutiles:** Al considerar la covarianza entre las variables, GMM puede identificar agrupamientos que no serían evidentes con K-means.
* **Obtener una medida de incertidumbre:** La probabilidad de pertenencia de un punto a un cluster proporciona información sobre la confianza en la asignación.
* **Modelar datos con distribuciones no esféricas:** La flexibilidad de GMM permite adaptarse a una amplia variedad de distribuciones de datos.

**En conclusión,** los resultados obtenidos con GMM sugieren que existen tres agrupaciones geográficas distintas en nuestros datos, cada una con características específicas. La forma elíptica de los clusters indica que los puntos dentro de cada grupo presentan cierta dispersión en ambas dimensiones (longitud y latitud), lo que podría estar relacionado con factores geográficos.

# Anexos

Como soporte y complemento de los resultados del caso de analítica desarrollado y documentado en el presente documento, se anexa el notebook *Proyecto\_Frailejones.ipynb.*

# Bibliografía

El Tiempo. (20 de Agosto de 2009). *En riesgo 386.000 hectáreas de páramo en Cundinamarca, por minería, ganadería y agricultura*. Obtenido de https://www.eltiempo.com/archivo/documento/CMS-5902889

*Gaussian Mixture Models*. (s.f.). Obtenido de https://nbviewer.org/github/sergiomora03/Gaussian\_Mixture\_Models/blob/master/intro\_to\_gmm\_%26\_em.ipynb

Rodrigo, J. A. (mayo de 2021). *Ciencia de datos, Redes neuronales con Phyton.* Obtenido de https://cienciadedatos.net/documentos/py35-redes-neuronales-python

Takuya Akiba, S. S. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework.* Obtenido de https://optuna.org/

Torres, L. (s.f.). *The Machine Learners. Optuna: ¿El futuro de la búsqueda de hiperparámetros?* Obtenido de https://www.themachinelearners.com/optuna-busqueda-hiperparametros/