Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3084 - Data Science - Ing. Lynette Garcia Perez

Proyecto 2

CGIAR Eyes on the Ground Challenge

Cristian Fernando Laynez 201281
Juan Angel Carrera 20593
Sara Maria Paguaga 20634
Guillermo Santos Barrios 191517

Guatemala,17 de noviembre de 2023

Índice

Introducción

En ciertas regiones de África, donde predomina el tiempo seco y la falta de precipitaciones, los agricultores corren el riesgo de sufrir grandes pérdidas de cosechas debido a las sequías recurrentes. Para reducir estos riesgos, ACRE África ayuda a los agricultores a gestionar el riesgo agrícola y realiza evaluaciones de pérdidas basadas en imágenes de los cultivos asegurados que presentan los agricultores para evaluar la reclamación del seguro y determinar las indemnizaciones adecuadas. Sin embargo, evaluar miles de imágenes de agricultores asegurados lleva mucho tiempo y ralentiza el proceso, por lo que es necesario automatizar la liquidación de siniestros para que el proceso sea más eficiente y ágil.

Para abordar esta necesidad, proponemos un proyecto para desarrollar un sistema automatizado de clasificación de imágenes para evaluar los daños en los cultivos a partir de las fotos enviadas. El sistema utilizará redes neuronales convolucionales, un tipo de algoritmo de aprendizaje profundo, para analizar las características de las imágenes y clasificar el nivel de daño en diferentes clases, como sano, daño leve, daño moderado o daño grave. La red neuronal se entrenará con un gran conjunto de datos etiquetados de imágenes de cultivos con evaluaciones de daños proporcionadas por agrónomos. Una vez que sea lo bastante preciso, el algoritmo podrá clasificar rápidamente las nuevas imágenes enviadas por los agricultores y proporcionar estimaciones de daños para agilizar el proceso de reclamación.

La automatización de la evaluación de los daños en los cultivos mediante la clasificación de imágenes puede beneficiar enormemente a los pequeños agricultores que dependen de los pagos de los seguros de cosechas tras las catástrofes meteorológicas. Al acelerar la tramitación de las reclamaciones, los pagos pueden llegar antes a los agricultores afectados, de modo que dispongan de recursos para replantar y seguir cultivando la próxima temporada. Este proyecto pretende aprovechar el poder del aprendizaje automático para crear una solución eficiente y escalable que ayude a mejorar la seguridad alimentaria y reducir los riesgos financieros de los agricultores de regiones propensas a la sequía. La capacidad de procesar rápidamente miles de reclamaciones podría hacer del seguro de cosechas una herramienta de gestión de riesgos más accesible y eficaz para las comunidades agrícolas rurales.

Objetivos

Objetivo general

- Generar un modelo preciso capaz de predecir y analizar daños en los cultivos causados por la sequía en base a imágenes de cultivos.

Objetivos específicos

- Implementar algoritmos de aprendizaje de máquina para el análisis y clasificación de imágenes de cultivos.
- Evaluar la precisión de los modelos generados y seleccionar al modelo más eficiente.
- Comparar la precisión del modelo más eficiente generado con soluciones existentes para la clasificación de daños por sequía.

Marco Teórico

Información sobre procesamiento de imágenes

El procesamiento de imágenes es un paso importante a la hora de entrenar modelos de aprendizaje automático con datos de imágenes. Existen varias técnicas habituales para preparar las imágenes para el entrenamiento de modelos:

Preprocesamiento: consiste en transformar las imágenes sin procesar en un formato adecuado para el modelo. Entre los pasos de preprocesamiento más comunes se incluyen el cambio de tamaño de las imágenes a un tamaño consistente, la conversión a escala de grises o RGB y la normalización de los valores de los píxeles. Esto ayuda al modelo a aprender más eficazmente de los datos de la imagen.

Aumento: Para aumentar el tamaño y la diversidad del conjunto de datos de entrenamiento, es habitual aumentar las imágenes utilizando técnicas como el volteo horizontal, las rotaciones, los recortes, los cambios de color, etc. Esto ayuda a exponer el modelo a representaciones variadas de los mismos patrones subyacentes durante el entrenamiento.

Formateo: las imágenes deben formatearse en tensores antes de introducirlas en un modelo de red neuronal. Esto implica convertir las imágenes en matrices multidimensionales y, potencialmente, codificar las etiquetas en un solo paso. La arquitectura del modelo determina exactamente cómo deben formatearse los tensores de la imagen.

El tratamiento adecuado de las imágenes es crucial para entrenar modelos de visión por ordenador precisos y robustos. Las técnicas utilizadas dependen de factores como el tipo de modelo, los datos de entrenamiento y los objetivos finales. Bibliotecas comunes como OpenCV, PIL y scikit-image proporcionan herramientas para implementar flujos de trabajo de procesamiento de imágenes estándar.

Información teórica sobre análisis de las imágenes desde el punto de vista de un agrónomo

Como agrónomo con más de 15 años de experiencia trabajando con pequeños agricultores del corredor seco, creo que este sistema de clasificación de imágenes podría ser transformador para las comunidades agrícolas vulnerables a la sequía. Acelerar el proceso de reclamación de los seguros de cosechas permite desembolsar rápidamente los pagos a los agricultores afectados para que puedan comprar insumos y cubrir los costes de plantación para la próxima temporada.

Con la evaluación manual tradicional de las imágenes, las liquidaciones pueden tardar semanas o meses, lo que obliga a las familias de agricultores en apuros a endeudarse mientras esperan la indemnización. Pero con la evaluación automatizada, las reclamaciones pueden verificarse y pagarse en cuestión de días. Este tiempo de respuesta más rápido es absolutamente crítico para los agricultores que viven temporada tras temporada.

Además, la evaluación automatizada de los daños en los cultivos puede ser más precisa y coherente que los métodos manuales. Se eliminan factores como la fatiga y la subjetividad, lo que mejora la precisión. Con los algoritmos de aprendizaje automático, la precisión de las clasificaciones de daños mejorará continuamente a medida que se recopilen más datos con el paso del tiempo.

Al permitir una evaluación más rápida y fiable de las pérdidas de cosechas, este sistema de clasificación de imágenes tiene el potencial de aumentar la confianza y la participación en los programas de seguros de cosechas. Los agricultores se sentirán más seguros al invertir en semillas, fertilizantes, herramientas y otros insumos de calidad, sabiendo que sus solicitudes se tramitarán rápidamente en caso de catástrofe. Esto puede contribuir a romper los ciclos de pobreza e inseguridad alimentaria en las comunidades agrícolas rurales. Apoyo incondicionalmente el desarrollo de esta solución basada en la IA como una innovación impactante que podría cambiar vidas en todo el continente.

Modelos Investigados

SVM (*Support Vector Machine*):

Las Máquinas de vector soporte son modelos supervisados que se destacan al manejar múltiples variables continuas o categóricas (Sanghvi, 2022). La idea es generar hiperplanos que puedan separar las distintas clases en las que se desean clasificar los datos. Es importante seleccionar una función de kernel correctamente para poder obtener los resultados deseados. Entre las funciones más utilizadas están: kernel lineal, kernel gaussiano y kernel polinomial.

K-Nearest Neighbor

Este algoritmo es de los más simples. Puede utilizarse para clasificar imágenes poco complejas. Si el conjunto de datos está lo suficientemente balanceado, también podrá clasificar conjuntos más complejos. El objetivo es obtener la clase para cada uno de los ejemplos deseados. Esto lo hace evaluando las distancias a las demás clases (distancia

Euclidiana o distancia Manhattan). Se selecciona la clase que tiene más vecinos cercanos (Sanghvi, 2022).

CNN (Convolutional Neuronal Networks):

Son redes diseñadas especialmente para identificar imágenes. Capas importantes de este tipo de redes son las capas de convolución y las capas de pooling. Las capas de convolución permiten identificar bordes en una imagen por medio de filtros y las capas de pooling reducen el tamaño de las matrices con los valores más representativos de las imágenes, de esta forma abstraen información y se reduce el costo computacional.

(Mishra, 2020)

Arquitecturas de CNNs para detección de imágenes:

- ResNet:
  Es común integrar más capas en una red neuronal profunda, para disminuir la tasa de error. Sin embargo, añadir más capas no siempre garantiza mejores resultados y puede presentar el problema de desvanecimiento y explosión de gradientes.

  La arquitectura ResNet permite que los gradientes fluyan de una mejor forma a través de la red al momento de hacer backpropagation, permitiendo entrenar redes mucho más profundas. Esta arquitectura propone una técnica llamada skip connections, la cual consiste en conectar las activaciones de una capa con otras capas saltándose algunas capas intermedias. Forma un bloque residual y las redes se forman apilando los bloques residuales.

  (“Residual Networks ResNet Deep Learning,” 2020)
- LeNet-5:

  LeNet-5 es una de las primeras redes neuronales convolucionales diseñadas específicamente para reconocimiento de dígitos escritos a mano. Fue introducida por Yann LeCun en 1998 y es reconocida por establecer la base para las redes convolucionales que la sucedieron.

  Arquitectura:
  La arquitectura LeNet-5 consta de 7 capas en total, entre las cuales hay capas de convolución, capas de pooling (subsampling) y capas completamente conectadas.

  Función de Activación:
  LeNet-5 utiliza la función sigmoide tangente hiperbólica como función de activación. A pesar de que la arquitectura LeNet-5 es bastante simple en comparación con las redes modernas, fue pionera y demostró la eficacia de las redes neuronales convolucionales para tareas de visión por computadora. A lo largo de los años, las CNNs han evolucionado y se han vuelto más complejas y profundas, pero LeNet-5 sigue siendo una referencia importante en la historia de las redes neuronales.

  (Gradient-based learning applied to document recognition, 1998)

RNN (*Recurrent Neuronal Networks*):

Es un tipo de red neuronal que utiliza datos secuenciales, es utilizada para casos con datos ordinales, es decir, que pueden ser ordenados según una escala o para problemas temporales. Este tipo de red neuronal se caracteriza por su capacidad de tomar

inputs previos para influenciar *inputs* y *outputs* actuales y tener una “memoria”, lo cual permite dar predicciones más precisas de información reciente. (*IBM*, 2023)

La estructura de este tipo de red es simple y tiene pocos parámetros lo cual representa una ventaja en términos de memoria y costo computacional así como un efecto positivo en tiempo de entrenamiento. Sin embargo, con este tipo de red se presenta el problema de explosión y desvanecimiento del gradiente cuando se entrena con secuencias muy largas lo cual genera inestabilidad numérica. (“How Do You Choose between RNN and LSTM for Natural Language Processing Tasks?,” 2023)

LSTM (Long Short Term Memory):

Es un tipo de red neuronal recurrente, este modelo permite retener información por un largo periodo de tiempo. Además, se usa para predecir y clasificar en base a datos de una serie de tiempo. Está diseñada precisamente para manejar datos secuenciales y es capaz de aprender de dependencias a largo plazo. (“Deep Learning Introduction to Long Short Term Memory,” 2019)

A comparación de la red neuronal recurrente este tipo de red evita el problema de explosión y desvanecimiento del gradiente y soporta entrenar con secuencias más largas. Por otra parte es menos propensa a sobre ajustarse en comparación a una RNN. No obstante, este tipo de red puede tener un mayor tiempo computacional. (“How Do You Choose between RNN and LSTM for Natural Language Processing Tasks?,” 2023)

Tanto una Red Neuronal Recurrente (RNN) como una red de tipo LSTM pueden ser de ayuda para la predicción de la extensión del área infectada y establecer la extensión de daño en un cultivo, dado que son datos secuenciales en una serie de tiempo.

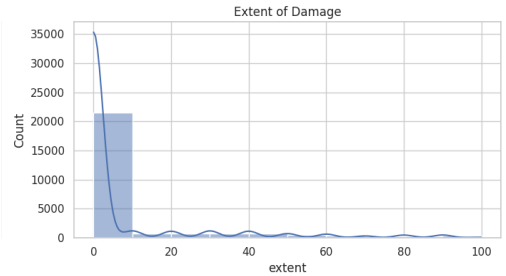
Metodología

Pasos que siguió el grupo para resolver el problema

1. Análisis del problema planteado por la competencia
    - Se realizó un análisis del problema que se pretendía resolver en la competencia de Zindi, para poder entender tanto los datos como las reglas proveídas.
2. Investigación del contexto del problema planteado
    - Para poder entender mejor el problema se leyó e investigó más del tema para tener una mejor idea de lo que se quería resolver.
3. Selección de algoritmos
    - Se evaluaron los algoritmos que podían ser aptos para generar una solución y en base a las necesidades se seleccionaron los algoritmos a plantear.
4. Preprocesamiento de datos
    - Se preprocesaron los datos para tener un mejor rendimiento de los algoritmos, tal como recorte de imágenes y data augmentation.
5. Implementación de algoritmos
6. Análisis de resultados
    - En base a los resultados obtenidos se realizó un análisis de los modelos, su precisión y generalización. En base a ello se eligieron los mejores modelos y que se pudieran adaptar a las necesidades del problema.

Explicación de cómo seleccionó el grupo los conjuntos de entrenamiento y prueba.

Teniendo en cuenta que teníamos un dataset desbalanceado como se puede observar en la siguiente imagen:



Utilizamos dos métodos de data para el desbalanceo de los datos, los cuales fueron el Oversampling y el Undersampling. Para que el sampleo tuviera sentido primero hicimos una división del 80% para el training y luego 20% para el testing, con esto lograremos que en el testing no existan datos del training.

El Oversampling se refiere a las técnicas que aumentan el número de ejemplos de la clase minoritaria para equilibrar la distribución de clases. Esto se consigue replicando o sintetizando nuevos ejemplos de la clase minoritaria. Entre los métodos de sobremuestreo más comunes se incluyen

- Oversampling aleatorio - Duplicar aleatoriamente las imágenes de la clase minoritaria. Es sencillo, pero puede dar lugar a sobre ajustes.
- SMOTE (Synthetic Minority Oversampling Technique) - Genera imágenes sintéticas combinando características de imágenes de clases minoritarias existentes. Reduce el riesgo de sobreajuste.
- Aumento de datos - Amplía el conjunto de datos aplicando transformaciones como giros, rotaciones y cambios de color a las imágenes de clases minoritarias existentes. Eficaz para datos de imágenes.

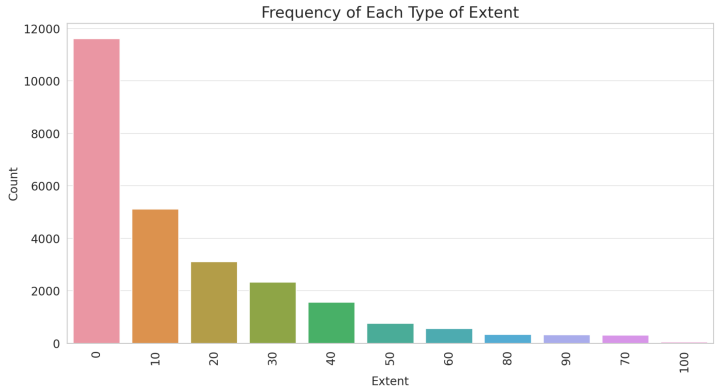
El Undersampling se refiere a las técnicas que reducen el número de ejemplos de la clase mayoritaria para equilibrar la distribución de clases. Para ello, se eliminan las imágenes de la clase mayoritaria. Los métodos de submuestreo más comunes son:

- Undersampling aleatorio: descarte aleatorio de las imágenes de la clase mayoritaria. Es rápido y sencillo, pero pierde datos potencialmente útiles.
- Undersampling informado: se eliminan estratégicamente las imágenes de clase mayoritaria redundantes o no críticas. Conserva más información útil.
- Agrupación de datos - Agrupar la clase mayoritaria en subgrupos y submuestrear cada grupo por separado. Preserva la diversidad de los datos.

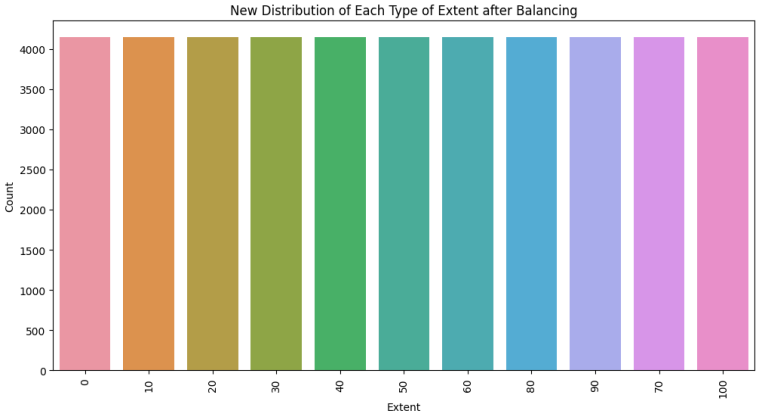
En nuestro hicimos una combinación de ambos teniendo que se hizo un aumento de datos para todos los datos de la extensiones mayor iguales a 20, luego se hizo un undersampling

para los datos que tuviera una extensión 0 para luego reducirlo a la cantidad de datos que tienen 10 de extensión de daño

Pasamos de este dataset:



Al siguiente, manteniendo la cantidad de datos igual a la segunda clase con mayor representación la cual en este caso fue la de la extensión de 10.



Explicación de la selección de los algoritmos y las razones por las cuales los escogieron.

CNN:

Elegir una red neuronal convolucional (CNN) para tareas de regresión basadas en imágenes puede ser ventajoso por varias razones:

Extracción de características: Las CNN son excelentes en la extracción automática y eficiente de características de las imágenes. Esto es crucial para las tareas de regresión, ya que permite a la red aprender las características más relevantes sin ingeniería manual de características.

Jerarquía espacial: Las CNN pueden captar la jerarquía espacial de las imágenes. Pueden reconocer varios aspectos de una imagen, desde bordes y texturas en las capas iniciales hasta patrones y objetos más complejos en capas más profundas. Este aprendizaje jerárquico de características es beneficioso para tareas de regresión que dependen de la comprensión del contexto espacial en las imágenes.

Robustez a las variaciones: Las CNN son relativamente robustas a las variaciones en la entrada, como cambios de escala, traslación e incluso pequeñas deformaciones. Esto las hace idóneas para situaciones reales en las que las imágenes no siempre están perfectamente alineadas o son coherentes.

Eficacia: Una vez entrenadas, las CNN pueden procesar rápidamente nuevas imágenes, lo que las hace adecuadas para aplicaciones que requieren predicciones en tiempo real o casi real.

Aprendizaje de extremo a extremo: Las CNN permiten un aprendizaje de extremo a extremo, en el que la red aprende a asignar directamente los valores brutos de los píxeles a los resultados. Esto elimina la necesidad de etapas separadas de extracción de características, selección de características y regresión, lo que simplifica el proceso de modelado.

Todos estos atributos de una CNN nos hicieron enfocarnos en este modelo de NN para este proyecto ya que con modelos más simples como una regresión lineal o un random forest no se obtuvieron resultados buenos por lo que una técnica de aprendizaje profundo fue necesaria para tener unos mejores resultados a la hora de hacer una regresión sobre el daño que tenían las plantas basado en la imagen.

Explicación de selección de las herramientas utilizadas

Tensorflow

Tensor Flow es una librería de código abierto desarrollada por el equipo de Google Brain. Se utiliza principalmente para tareas de aprendizaje automático e inteligencia artificial, específicamente redes neuronales.

Características principales

- Flexibilidad: Admite una amplia gama de modelos de ML y aprendizaje profundo.
- Escalabilidad: Trabaja eficientemente con sistemas y conjuntos de datos a gran escala.
- Visualización: Integrado con TensorBoard para visualizar gráficos de modelos, métricas y más.
- Comunidad y Soporte: Amplio soporte de la comunidad y actualizaciones periódicas.

Pytorch

PyTorch es una biblioteca de aprendizaje automático de código abierto desarrollada por el laboratorio AI Research de Facebook. Es conocida por su flexibilidad e interfaz fácil de usar, especialmente en el ámbito del aprendizaje profundo y las redes neuronales.

Características principales

- Gráfico computacional dinámico: Conocido como Autograd, esta función proporciona flexibilidad a la hora de construir y modificar gráficos sobre la marcha.
- Naturaleza Pitónica: Se integra perfectamente con el ecosistema Python y es intuitivo para los usuarios de Python.
- Rico ecosistema: Una amplia gama de herramientas y bibliotecas para funcionalidades extendidas como TorchVision para tareas de visión por ordenador.

Google Colab

Google Colab es un servicio basado en la nube que permite a los usuarios escribir y ejecutar código Python a través del navegador. Es ampliamente utilizado para el aprendizaje automático, análisis de datos, y la educación.

Características principales

- Cero Configuración: No requiere configuración y el código se ejecuta en la nube.
- Acceso gratuito a GPUs: Proporciona acceso gratuito a unidades de procesamiento gráfico (GPU) y unidades de procesamiento tensorial (TPU).
- Colaboración: Facilidad para compartir y colaborar en proyectos, similar a Google Docs.
- Integración: Se integra a la perfección con Google Drive y otros servicios de Google.

Paperspace

Paperspace es una plataforma de computación en la nube que ofrece máquinas virtuales aceleradas por GPU. Se utiliza para una variedad de propósitos, incluyendo el aprendizaje automático, juegos y diseño 3D.

Características principales

- Potentes GPUs: Ofrece acceso a GPUs de alto rendimiento, lo que es ideal para tareas computacionales intensivas.
- Versatilidad: Admite una amplia gama de aplicaciones, desde el aprendizaje automático hasta las de uso intensivo de gráficos.
- Interfaz fácil de usar: Ofrece una interfaz intuitiva y una configuración sencilla.
- Precios flexibles: Ofrece varios planes de precios, incluidas opciones de pago por uso, lo que lo hace accesible para diferentes tipos de usuarios.

Recursos de cómputo.

Google Colab PRO

Dentro de colab pro se utilizó el entorno de ejecución T4 GPU para poder ser capaces de entrenar el primer modelo, evaluar su desempeño y generalización de los datos, aprovechando de esta forma el uso eficiente de recursos.

PaperSpace

Dentro de Paperspace utilizamos pytorch para entrenar el segundo modelo. Para aprovechar al máximo las características de computación utilizamos una tarjeta NVIDIA A-6000 de 48 GiB de GPU, 8 núcleos de CPU con 45 GiB de RAM.

Lenguajes de programación, bibliotecas y/o paquetes utilizados

Python

Python es un lenguaje de programación interpretado de alto nivel conocido por su sencillez y legibilidad. Su naturaleza versátil lo hace adecuado para diversas aplicaciones, desde el desarrollo web hasta la informática científica. La amplia biblioteca estándar de Python, combinada con su comunidad de apoyo, ha dado lugar a un rico ecosistema de bibliotecas de terceros, lo que lo convierte en una opción popular entre desarrolladores, investigadores y científicos de datos. Todas nuestras bibliotecas son de python, además de que todos los modelos y la aplicación fueron creadas en python con el uso de las siguientes librerías.

Torch & Torchvision

Torch: Un marco de computación científica con amplio soporte para algoritmos de aprendizaje automático, Torch se basa principalmente en Lua, pero tiene una contraparte PyTorch en Python. Ofrece gran flexibilidad y velocidad, sobre todo en aplicaciones de aprendizaje profundo y redes neuronales.
Torchvision: Es un paquete de la librería PyTorch, específicamente diseñado para tareas de visión por computador. Proporciona acceso a conjuntos de datos populares, arquitecturas de modelos y transformaciones de imágenes comunes para la visión por ordenador.
Con estas herramientas se entrenó el segundo modelo en la plataforma de paperspace usando Clases como NN para la creación de las capas Convolucionales y neuronales haciendo uso de las funciones de activación como relu y las librerías para carga de dataset.

PIL (Image)

PIL, o Python Imaging Library, ahora conocida como "Pillow" en su versión mantenida, es una biblioteca en Python que permite abrir, manipular y guardar muchos formatos diferentes de archivos de imagen. Se utiliza ampliamente para tareas de procesamiento de imágenes como lectura y escritura de imágenes, transformaciones de imágenes y filtrado. Con esto hacemos la carga manual de las imágenes al modelo.

Pandas

Pandas es una potente y flexible biblioteca de Python utilizada principalmente para la manipulación y el análisis de datos. Ofrece estructuras de datos como DataFrame y Series, ideales para manejar datos estructurados. Pandas simplifica tareas como la limpieza, transformación y agregación de datos, lo que la hace indispensable en los flujos de trabajo de la ciencia de datos. Nos ayudó con la transformación de los datos desde la normalización de la extensión hasta el oversampling y undersampling de los datos.

MatplotLib

Matplotlib es una biblioteca de gráficos en Python para crear visualizaciones estáticas, interactivas y animadas. Proporciona una amplia variedad de diagramas y gráficos, que son altamente personalizables. Matplotlib es especialmente útil en la visualización de datos para comprender tendencias, patrones y distribuciones en los datos. Nos ayudó a visualizar los datos dentro de python como la distribución y el loss de los modelos.

sklearn

Scikit-learn, a menudo conocida como sklearn, es una biblioteca de Python ampliamente utilizada para el aprendizaje automático. Proporciona herramientas sencillas y eficaces para la minería y el análisis de datos. Sklearn es conocida por su completa colección de algoritmos de clasificación, regresión, agrupación y reducción dimensional, junto con herramientas de selección de modelos y preprocesamiento. Nos ayudó con la división de los dataset de training y testing para entrenar los modelos.

Tensor Flow

Tensor Flow, actualmente es una de las librerías más utilizadas y más importantes para el desarrollo de la inteligencia artificial y el aprendizaje automático. Fue desarrollado por

Google para satisfacer las necesidades a partir de redes neuronales artificiales, esta librería permite construir y entrenar redes neuronales para detectar patrones y razonamientos. Gracias a esto todo esto Tensor flow permite la creación e implementación de modelos de aprendizaje automático, esto nos permite automatizar ciertos procesos de manera intuitiva, nos ayudó en la creación de varios modelos y redes neuronales para entrenar y analizar las imágenes a realizar.

Streamlit

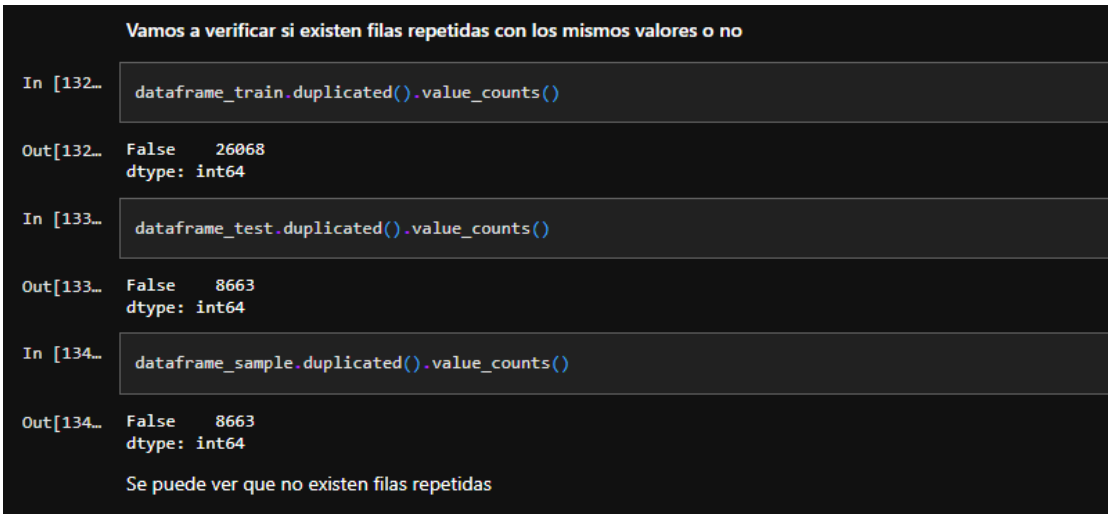
Streamlit es un framework de python que permite crear de forma sencilla e integrada, desarrollar aplicaciones interactivas gracias a la interacción con otras librerías, esto se aplica principalmente en la ciencia de datos. Ofrecen diversas plantillas para observar e interactuar de diferentes formas los datos analizados. Gracias a este framework se pudo crear una bonita aplicación interactiva para ver los resultados de los datos analizados.

Resultados y Análisis de Resultados

Tareas de limpieza y de preprocesamiento

En este modelo se hizo un custom loader para cada imagen para poder hacer esto se hizo un resize de la imagen a 128 x 128 para luego hacer una normalización del tensor para poder entrenarla en la arquitectura descrita anteriormente.

Se evaluó si en los datos descargados había presencia de datos inválidos o duplicados. La revisión confirmó que la data está limpia, sin duplicaciones ni datos erróneos, reafirmando la calidad de la información para el análisis. Además se evaluó que no existieran imágenes que no estaban dentro del dataset, es decir en todo el zip si estuvieran todos los datos, obtuvimos que hicieron falta alrededor de 2000 imágenes entonces esos datos fueron descartados. Luego hicimos un re-shape a las imágenes para ajustarlas al modelo



Vamos a verificar si hay data duplicada en cada una de las filas

```
In [135…  dataframe_train['filename'].value_counts().sort_values(ascending=False)

Out[135…  L427F01330C01S03961Rp02052.jpg     1
          L415F01160C39S14146Rp41363.jpg     1
          L341F00167C01S00324Rp14178.jpg     1
          L1084F02394C39S13931Ip.jpg         1
          L361F02347C01S10018Rp27925.jpg     1
                                            ..
          L406F02369C01S06908Rp32150.jpg     1
          L371F04405C20S11053Rp27732.jpg     1
          L1150F01299C39S12449Rp33134.jpg    1
          L342F01261C01S00366Rp01836.jpg     1
          L406F00362C01S00614Rp06760.jpg     1
          Name: filename, Length: 26068, dtype: int64
```

Análisis Exploratorio

A, B) Descripción de Variables y Observaciones Disponibles

	ID	filename	growth_stage	damage	extent	season
0	ID_1S8OOWQYCB	L427F01330C01S03961Rp02052.jpg	S	WD	0	SR2020
1	ID_0MD959MIZ0	L1083F00930C39S12674Ip.jpg	V	G	0	SR2021
2	ID_JRJCI4Q11V	24_initial_1_1463_1463.JPG	V	G	0	LR2020
3	ID_DBO3ZGI1GM	L341F00167C01S00324Rp14178.jpg	M	DR	60	SR2020
4	ID_ORZLWTEUUS	L1084F02394C39S13931Ip.jpg	V	G	0	SR2021
...	...	...	...	...	...	...
26063	ID_3II1SXC0ZO	L1084F03259C39S12149Rp41671.jpg	M	DR	30	SR2021
26064	ID_OE7OU9ZF4U	L406F04369C01S07190Rp22847.jpg	V	G	0	LR2021
26065	ID_20M531UIZZ	L134F00766C01S09784Rp26034.jpg	M	G	0	LR2021
26066	ID_BZBV2FH0KL	L1153F02464C01S00194Rp01561.jpg	F	G	0	SR2020
26067	ID_ZVBF61EA0I	L406F00362C01S00614Rp06760.jpg	V	G	0	SR2020

Número de observaciones (filas): 26,068
Número de variables (columnas): 6

Descripción de cada una de las variables
ID: Identificador único para cada observación (tipo objeto).
filename: Nombre del archivo de la imagen correspondiente a la observación (tipo objeto).

growth_stage: Etapa de crecimiento del cultivo (variable categórica, tipo objeto). Los posibles valores son:

- F: Flowering (Floración)
- M: Maturity (Madurez)
- S: Sowing (Siembra)
- V: Vegetative (Vegetativo)

damage: Tipo de daño observado (variable categórica, tipo objeto). Los posibles valores son:

- DR: Drought (Sequía)
- DS: Disease (Enfermedad)
- FD: Flood (Inundación)
- G: Good (Bueno)
- ND: Nutrient Deficient (Deficiencia de nutrientes)
- PS: Pest (Plaga)
- WD: Weed (Maleza)
- WN: Wind (Viento)

extent: Extensión del daño observado expresado en porcentajes con incrementos del 10% (variable numérica, tipo int64).

season: Temporada de la observación (variable categórica, tipo objeto). Los posibles valores son:

- LR2020
- LR2021
- SR2020
- SR2021

C) Resumen de las variables

Resumen de la variable numérica 'extent'

- Conteo: 26,068
- Media: ~7.10%
- Desviación estándar: ~18.61%
- Mínimo: 0%
- 25% (Primer cuartil): 0%
- 50% (Mediana): 0%
- 75% (Tercer cuartil): 0%
- Máximo: 100%

Tablas de frecuencia para las variables categóricas

Variable 'growth_stage'

- V (Vegetative): 10,015 observaciones
- M (Maturity): 6,664 observaciones
- F (Flowering): 6,164 observaciones
- S (Sowing): 3,225 observaciones

Variable 'damage'

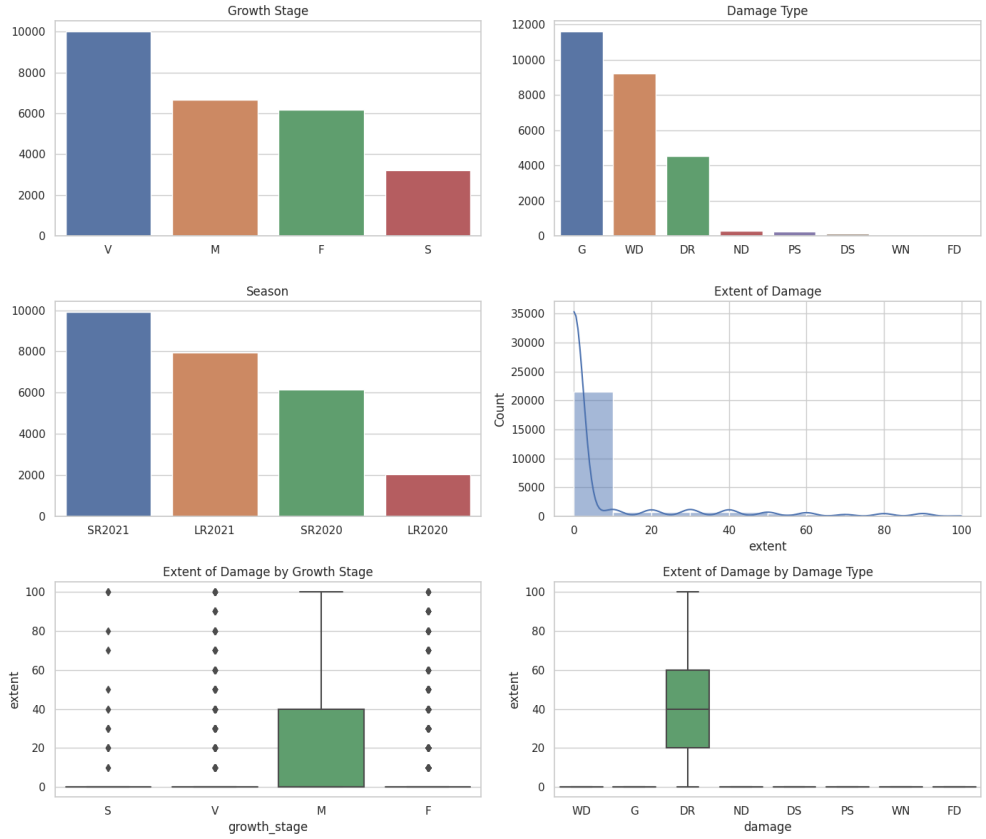
- G (Good): 11,623 observaciones
- WD (Weed): 9,238 observaciones
- DR (Drought): 4,516 observaciones
- ND (Nutrient Deficient): 272 observaciones

- PS (Pest): 254 observaciones
- DS (Disease): 115 observaciones
- WN (Wind): 37 observaciones
- FD (Flood): 13 observaciones

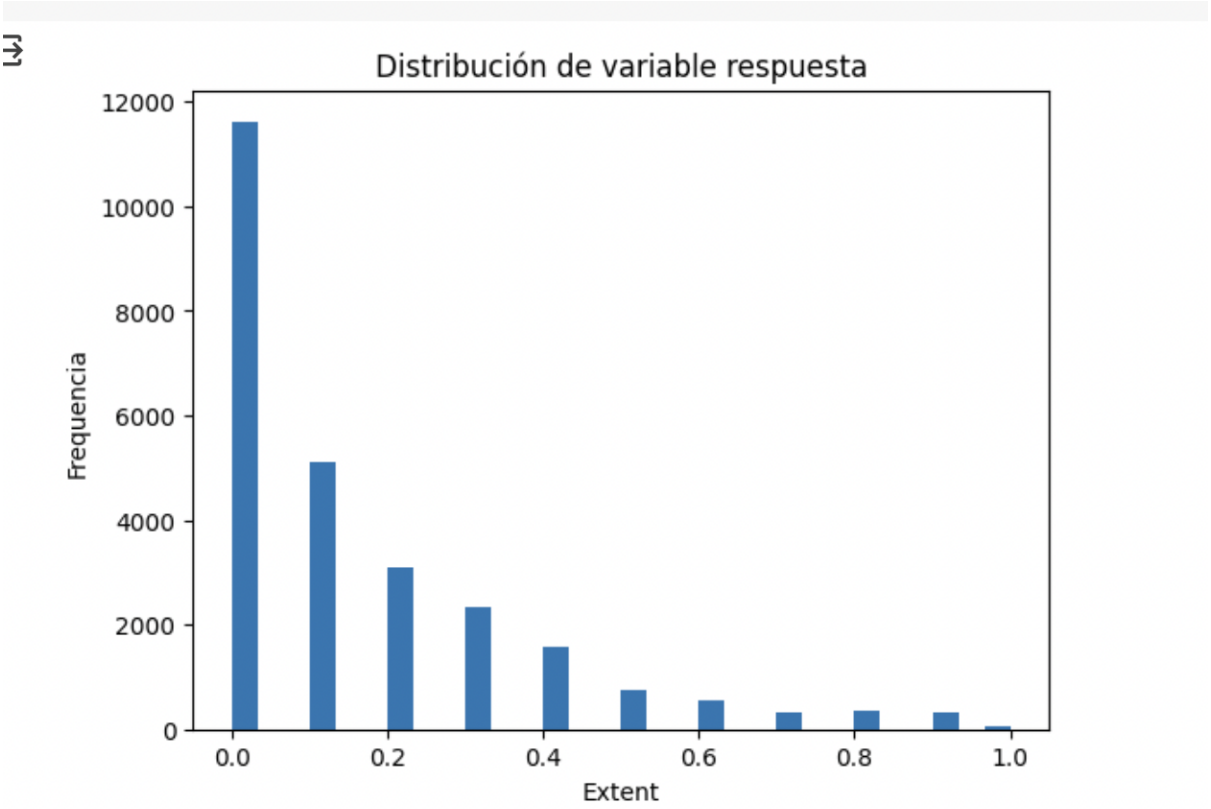
Variable 'season'

- SR2021: 9,927 observaciones
- LR2021: 7,945 observaciones
- SR2020: 6,163 observaciones
- LR2020: 2,033 observaciones

D) Gráficos



Los gráficos anteriores nos proporcionan una visión detallada de cada variable y cómo se relacionan entre sí.



Se ve un claro desbalance entre las categorías de la variable respuesta.

Análisis Individual de Variables

Growth Stage (Etapa de crecimiento)

La mayoría de las observaciones están en la etapa vegetativa (V), seguida por la etapa de madurez (M) y la etapa de floración (F).

Damage Type (Tipo de daño)

La categoría "Good" (sin daño) tiene la mayoría de las observaciones, seguida por "Weed" (presencia de maleza) y "Drought" (sequía).

Season (Temporada)

La temporada con más observaciones es SR2021, seguida de cerca por LR2021.

Extent of Damage (Extensión del daño)

La mayoría de las observaciones tienen un porcentaje de daño del 0%, lo que indica que hay muchas observaciones sin daño. El histograma muestra una distribución muy sesgada hacia la derecha.

Análisis Cruzado de Variables

Extent of Damage by Growth Stage (Extensión del daño por etapa de crecimiento)

Se observa que la mediana del daño en todas las etapas de crecimiento es 0%. Sin embargo, hay una presencia significativa de outliers, especialmente en las etapas de madurez (M) y vegetativa (V).

Extent of Damage by Damage Type (Extensión del daño por tipo de daño)

Aquí podemos ver cómo varía el daño en función del tipo de daño. La categoría "Good" tiene una mediana de 0%, como era de esperar. La categoría "Drought" (DR) muestra una mayor variabilidad en la extensión del daño, con varios casos extremos.

Explica el ajuste de los parámetros

Modelo 1

Arquitectura

- Capa de *data augmentation*
- Capa de normalización
- Capa Convolucional 2D
    - filtros = 32
    - Kernel size = 3
    - Función de activación = relu
    - Input shape = Img height, Img width, 3
- Capa de Max Pooling 2D
- Capa Convolucional 2D
    - filtros = 64
    - Kernel size = 3
    - Función de activación = relu
- Capa de Max Pooling 2D
- Capa Convolucional 2D
    - filtros = 128
    - Kernel size = 3
    - Función de activación = relu
- Capa Flatten
- Capa Dense
    - units = 64
    - Función de activación = sigmoid
- Capa Dense
    - units = 1
    - Función de activación = linear

Explica el ajuste de parámetros

Inicialmente se realizó Data Augmentation en donde se realizaron giros horizontales, así como una rotación y zoom del 10%. Por otra parte, para la capa de normalización se utilizó un parámetro de 1./255 para normalizar los valores de los pixeles y que estuviesen en un rango de 0 y 1. Además, para la capa de convolución se consideraron varios filtros para extraer características de la imagen a distintos niveles de complejidad. También cabe mencionar que en la implementación de *early stopping* se aplicó el parámetro de *patience* = 3 para que se detuviera el entrenamiento si la pérdida de validación no mejoraba después de épocas para evitar el sobreajuste.

Modelo 2:

Arquitectura

- Capa Convolucional
  - Kernel size = 3
  - Stride = 1
  - Padding = 1
  - Capa de 128 x 32
- Capa de Batch Norm
  - num_features = 32
- Relu
  - Como funcion de activacion
- Dropout
  - Probabilidad = 25%

Explica el ajuste de los parámetros

Esto durante 4 veces duplicando el tamaño de la capa cada vez para finalmente hacer dos capas Lineares con un Relu cada una. La capa convolucional y el kernel fueron ajustados a las imágenes que se tienen. También el dropout se fue modificando para ya que al inicio lo teníamos un 35% pero luego nos dimos cuenta que era muy malo en la regresión y lo bajamos a solo 25% pero no hubo mucha mejora. Una recomendación para una siguiente implementación sería eliminar este dropout por completo ya que aun con riesgo de overfitting para una regresión como la que se está haciendo sería mejor tener todas las características posibles sobre cómo es la regresión.

Comparación de la efectividad de los algoritmos

Modelo 1 - CNN (3-C)

Gráfico de pérdida (Modelo entrenado con 10 épocas y Early Stopping)

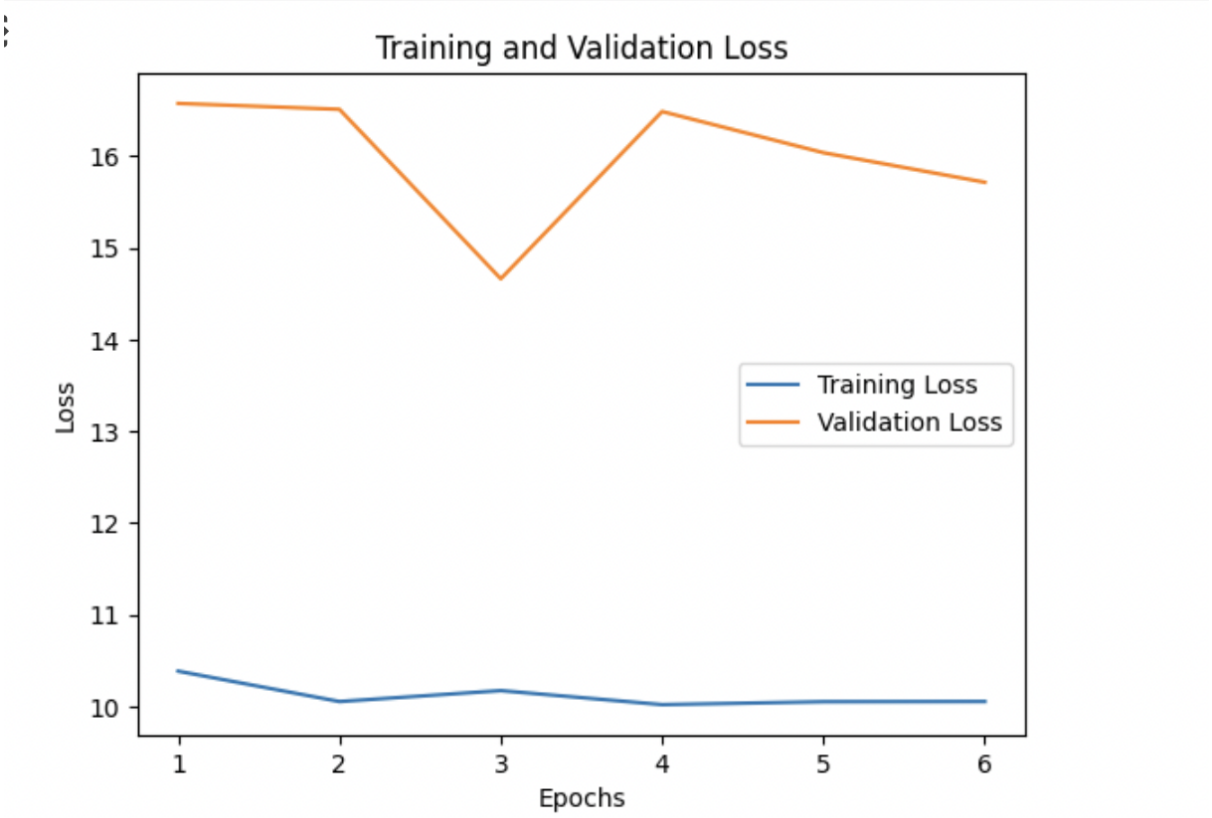


Gráfico MSE (Modelo entrenado con 10 épocas y Early Stopping)

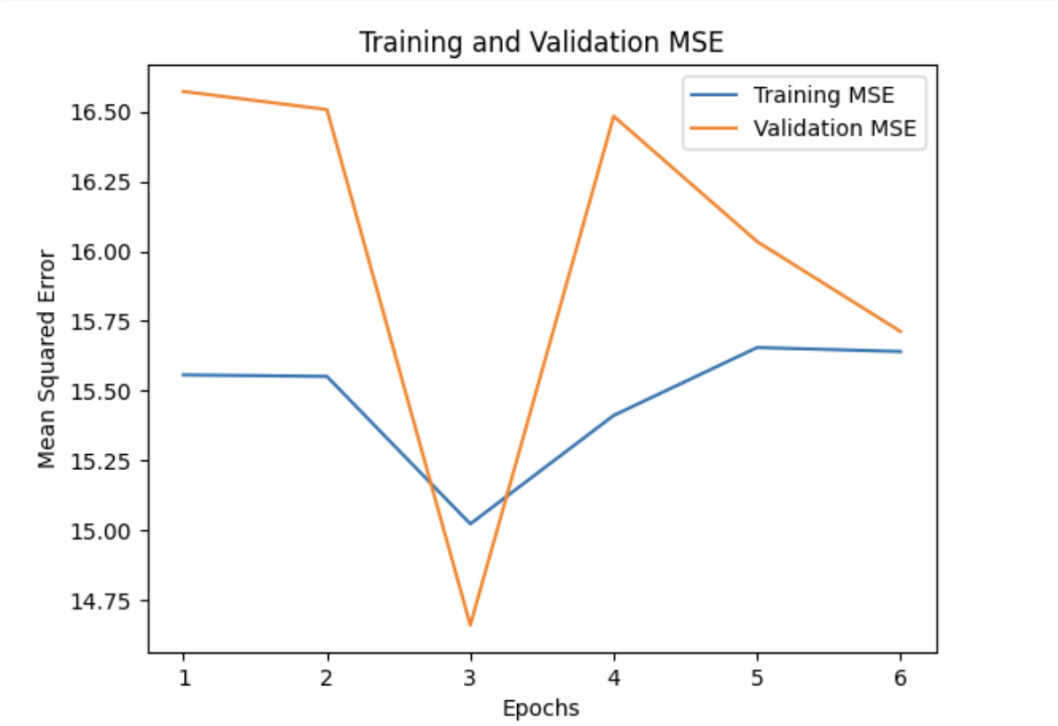
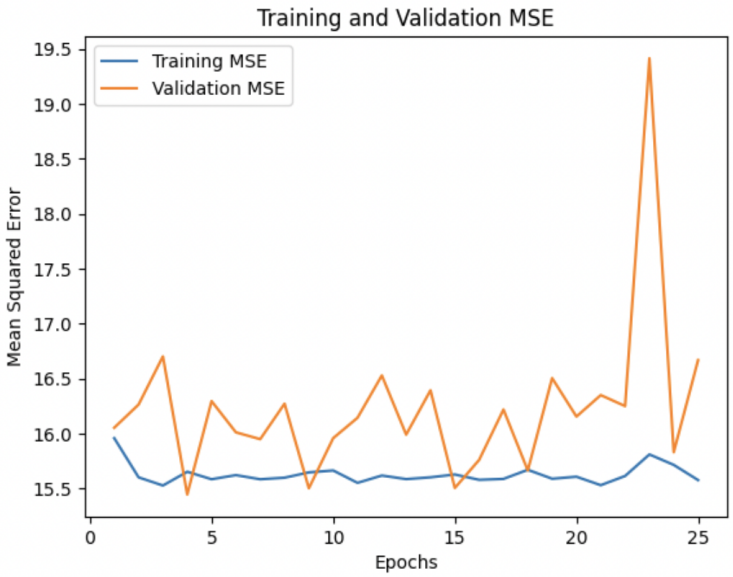


Gráfico de pérdida (Modelo entrenado con 25 épocas)

Gráfico MSE (Modelo entrenado con 25 épocas)



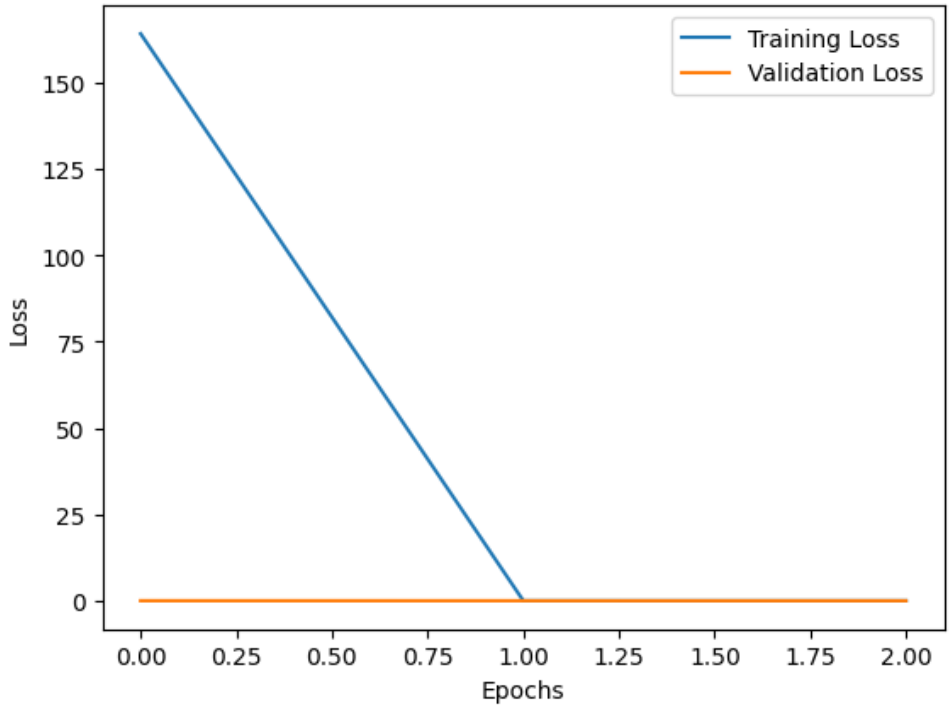

```
img_dir = '11_repeat_2_2416_5427_3808.JPG'
result = img_regression(img_dir)
print(result)
```

```
1/1 [==============================] - 0s 172ms/step
[[5.200368]]
```

Se observa que ambos modelos, el entrenado con 10 épocas y Early Stopping y el de 25 épocas, muestran signos de subajuste. Inicialmente, parecía que el modelo de 10 épocas podría beneficiarse de un mayor número de épocas para superar este problema. Sin embargo, incluso al aumentar a 25 épocas, el patrón de subajuste persistió. A pesar de normalizar los datos y emplear Data Augmentation, el problema de subajuste continuó. Esto sugiere que podría ser necesario disponer de más imágenes para entrenar el modelo y mejorar su capacidad de generalización. Además, se identificó un desbalance en las clases, y aunque se intentó compensar esto asignando pesos a las clases, esta medida no resultó eficaz para resolver el problema de manera clara, es por eso que se intuye que los valores predecidos se encuentraban alrededor de 5.

Modelo 2 - CNN (4-C)

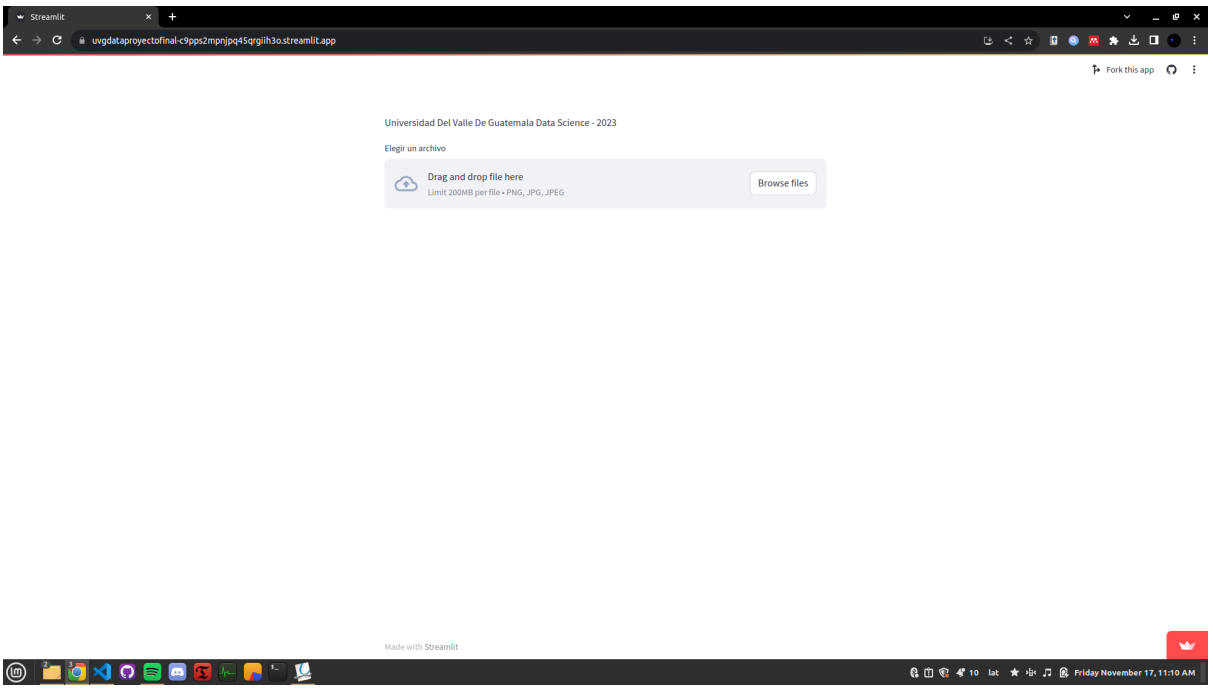
Entrenamos el modelo durante 3 épocas, cada época tardaba en entrenar alrededor de una hora para al final tener que tan solo después de 1 época tuviera un comportamiento tan extraño ya que básicamente colapso en 0 y el loss del validation también siempre estuvo alrededor de 0.



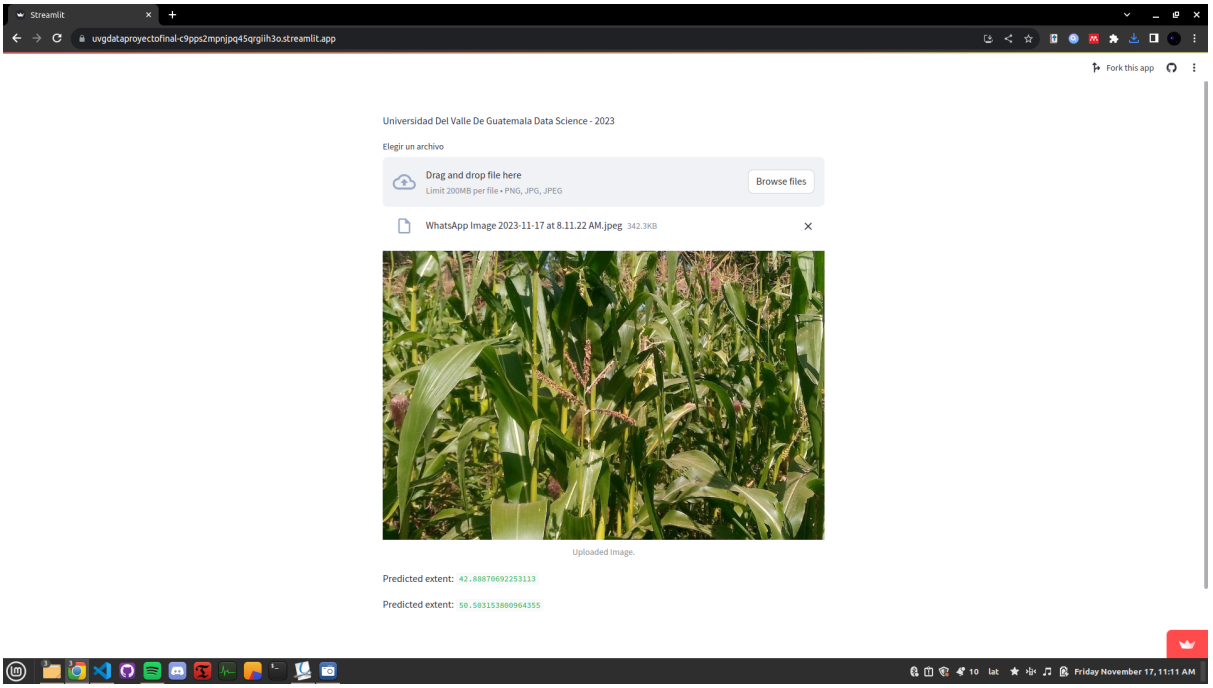
En cuanto al tiempo para procesar una nueva imagen es de alrededor de 2.30 segundos lo cual es bastante lento en comparación al otro modelo que solo tarda 77 ms en procesar una nueva imagen.

Aplicación creada para probar los algoritmos

Como se mencionó anteriormente se creó la aplicación con Streamlit. En la aplicación se permite subir una imagen de una planta tipo png, jpg y jpeg siempre y cuando no exceda los 200 MB.



Luego al subir la imagen a analizar se obtienen los resultados de la predicción.



Conclusiones

- El desbalance de clases influye en el subajuste de un modelo.
- Para llevar a cabo el entrenamiento de los modelos es importante realizar una limpieza de datos y en casos necesarios ajustar las imágenes para un mejor entrenamiento para el modelo a implementar.
- La técnica de dropout influye en la pérdida de características importantes en el modelo.

- El ajuste de hiper parámetros permite analizar e ir comprendiendo cómo mejorar los modelos.

Enlace de github: https://github.com/MGonza20/Proyecto2_DataScience
Enlace de drive: https://docs.google.com/document/d/1wQ1eMSmTLkHKK6peDZ5iynn-pZVl0Q8IGvc7P29r4fs/edit?usp=sharing
Enlace de visualización interactiva: https://uvgdataproyectofinal-c9pps2mpnjpq45qrgiih3o.streamlit.app/
Enlace de canva: https://www.canva.com/design/DAF0b0rl_os/fvBnB1VksJTPA1_mH2dGRw/edit?utm_content=DAF0b0rl_os&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Bibliografía

Referencias:

Mishra, M. (2020, Augosto 26). Convolutional Neural Networks, Explained - Towards Data Science. Recuperado de: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Sanghvi, K. (2023, Abril 21). Image Classification Techniques - Analytics Vidhya - Medium. *Medium*. Recuperado de: https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac#:~:text=The%20algorithms%20include%20linear%20regression,%2C%20and%20k%2Dnearest%20neighbor.

Residual Networks ResNet Deep Learning. (2020, Junio 3). Recuperado de: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

What are Recurrent Neural Networks? | IBM. (2023). Ibm.com. Recuperado de: https://www.ibm.com/topics/recurrent-neural-networks

Deep Learning Introduction to Long Short Term Memory. (2019, Enero 16). Recuperado de: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

How do you choose between RNN and LSTM for natural language processing tasks? (2023). Recuperado de: https://www.linkedin.com/advice/0/how-do-you-choose-between-rnn-lstm-natural-language