
Dario Jauregui Lankenau - A00827837
Cristofer Becerra Sánchez - A01638659
Francisco Leonid Galvez Flores - A01174385
Josué Jemuel Flores Nestor - A01367182
02/Dic/22

Detección de humo, fuego, y predicción de zonas en riesgo de incendio forestal

Inteligencia artificial avanzada para la Ciencia de Datos

TC3007C - Grupo 101

Equipo 5



Metodología General

Para ambos proyectos se utilizó la metodología CRISP-DM (Cross-industry process for data mining), a grandes rasgos CRISP-DM funciona como un ciclo continuo que consta de 6 pasos.

Business Understanding

Durante esta etapa se busca definir el problema y al mismo tiempo tener con claridad los objetivos que se buscarán cumplir, esto incluye conocer las limitaciones que existen, los riesgos y las métricas para saber si el proyecto ha tenido éxito.

Data understanding

En este momento se tienen que revisar los datos que se tienen disponibles y analizarlos para conocer la calidad de la información y saber si los datos serán útiles para el proyecto.

Data preparation

Una vez que se revisan los datos, es momento de que se preparen para el modelo que se va a implementar, esto incluye, seleccionar los datos, limpiar los datos y crear nuevos atributos si es necesario.

Modeling

Se selecciona el tipo de modelo a usar, la arquitectura y una propuesta de modelo, después de esto se puede construir el modelo y dejar todo listo para que se implementen cambios en una futura iteración.

Evaluation

Se deben de revisar las métricas resultantes y lo que sucedió en el proceso, para que, con base en eso, se definan los siguientes pasos para mejorar el proyecto.

Deployment

Se tiene que crear un plan de implementación y al mismo tiempo definir cómo se va a mantener el proyecto una vez implementado.

Computer Vision

Metodología

Business Understanding

En esta etapa fue muy importante la comunicación con Green Tech Innovation, para tener claridad de lo que se buscaba lograr y poder obtener los requerimientos del proyecto, que se vieron reflejados en el documento de requerimientos.

Data understanding

Cuando se tenía claro lo que se quería lograr fue momento de revisar la información disponible, esto representó analizar la información que proporcionó el INEGI, datasets de Kaggle y más fuentes de información disponibles.

Data preparation

Al final se encontró información muy valiosa en Kaggle que se pudo utilizar para el proyecto, pero en una segunda iteración se optó por usar otra fuente de información para las imágenes del modelo.

Modeling

Cuando los requerimientos y el dataset se encontraban definidos se pudo tomar la decisión de la selección del modelo para cumplir con lo que solicitaba Green Tech Innovation.

Evaluation

Después de tener el modelo se revisaron las métricas de precisión del modelo, con ayudas visuales como gráficas para que se pudieran comparar con las siguientes iteraciones.

Deployment

Para la implementación se tomó la decisión de hacerla lo más cercana posible a lo que Green Tech Innovation quiere lograr, que es llevar este modelo a drones, es por eso que la implementación se hizo encaminando hacia ese objetivo.

Datos

En un acercamiento inicial al modelo de Computer Vision, se utilizó un conjunto de 3306 imágenes divididas en “smoke”, “fire” y “forest” obtenido en Kaggle. Se ilustran varios ejemplos del conjunto en la figura 1.



Figura 1: Ejemplos de imágenes pertenecientes al dataset de Kaggle con ejemplos de las tres categorías.

El problema de esta primera instancia y los datos que se utilizaron es que además de que eran un número muy grande para la limpieza, manipulación y evaluación de métricas, existían múltiples imágenes que requerían respetar licencias. Por esta razón, se decidió replantear la implementación.

Entonces se creó un segundo dataset más pequeño, que consta de 781 imágenes de las mismas tres clases: *smoke*, *fire* y *forest*. Dichas imágenes se extrajeron de Google Search a través de búsquedas como “wildfire”, “forest fire”, “forest”, “forest drone view”, “forest smoke” y similares. De cada búsqueda se descargaron todas las imágenes de la página con la extensión Download All Images. Posterior a la descarga, se inspeccionó el conjunto de datos para limpiar aquellas imágenes irrelevantes, repetidas y de traslape entre “smoke” y “fire”; además, se quitaron las imágenes con marcas de agua y obstáculos importantes que pudieran sesgar los pesos de los kernels de la red. La figura 2 muestra algunos ejemplos del nuevo conjunto de imágenes.

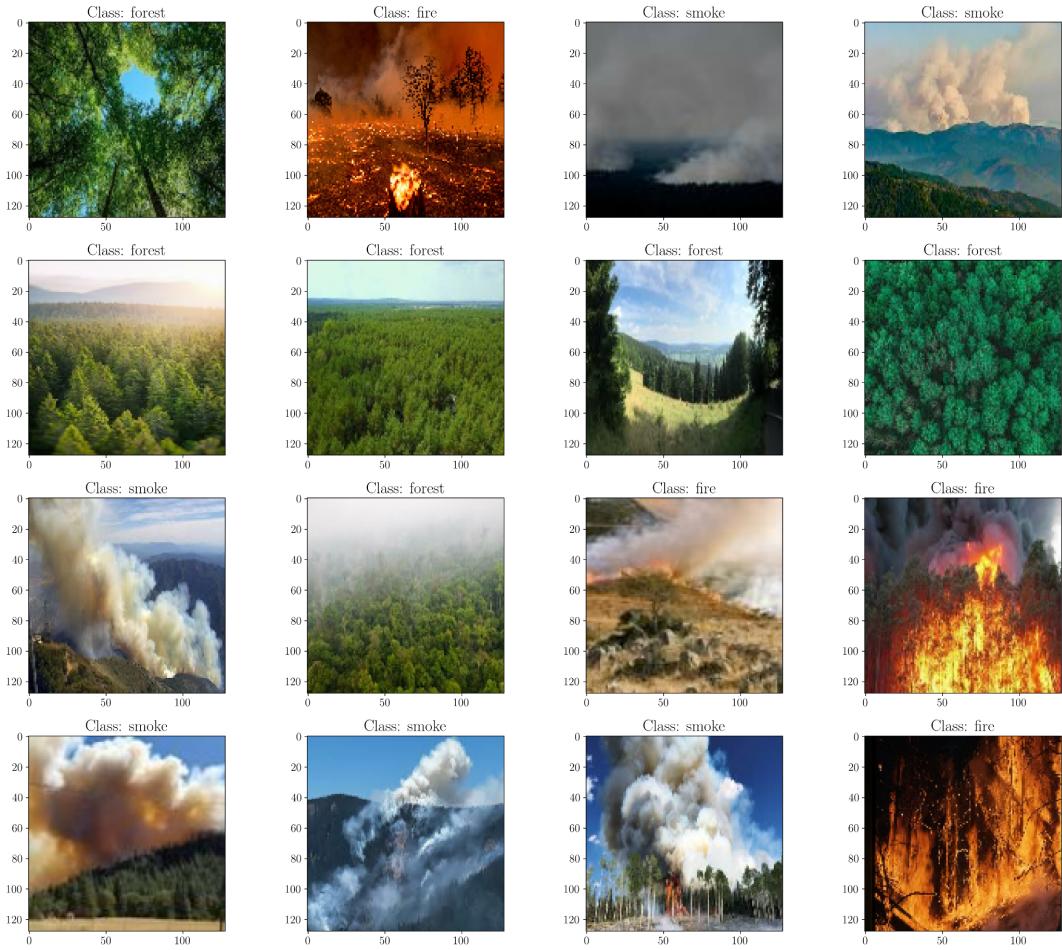


Figura 2: Ejemplos de imágenes del segundo conjunto de datos obtenidos. Se resalta que el nuevo conjunto de imágenes es menos homogéneo que el primero, pues el origen de las imágenes es de naturaleza distinta; quizá el primer conjunto se creó manualmente con una implementación específica en mente (cámaras de seguridad).

Debido a que el dataset a utilizar es mucho más pequeño y por tanto bastante ligero (160MB aprox), no es pertinente implementar un enfoque orientado a Big Data.

Para el pre-procesamiento de imágenes se utilizó el concepto de data augmentation, el cual incluyó un Random Zoom y un Random Flip de manera horizontal, logrando conseguir más características de cada una de las imágenes y más datos para el entrenamiento del modelo. Otro aspecto importante fue el ajuste de las dimensiones de las imágenes, pues el conjunto no era homogéneo en sus medidas; este proceso fue facilitado por la función `tf.keras.utils.image_dataset_from_directory()` del API de Tensorflow que crea un Dataset con las dimensiones especificadas por el parámetro `image_size`. Las dimensiones finales de las imágenes fueron de (128, 128, 3).

Los datos se dividieron en 3 secciones; el 70% se utilizó para el entrenamiento, el 20% como validación, y finalmente el 10% restante para la parte de prueba. Esta división de imágenes es tomada de forma aleatoria para mejorar la legitimidad de las pruebas.

En términos de ética, las imágenes no contienen datos sensibles o personales que puedan atentar contra la privacidad de las personas, por lo tanto, no es necesario anonimizar los datos del modelo. Adicionalmente, el procedimiento de acceso a los datos son los protocolos de seguridad de Google Drive en Unidades compartidas, y las barreras de ciberseguridad de una máquina para el almacenamiento local. Finalmente, puesto que el uso de los datos es para entrenamiento de modelos de predicción, es decir la extracción de patrones de los datos, está dentro de las excepciones establecidas por el Copyright, Designs and Patents Act de 1988 bajo la categoría de Text and Data Mining.

Modelo

Selección de modelo

Se inició el proceso de selección considerando la naturaleza del problema; se identificaron las dos componentes principales de esta parte del proyecto, a saber, que se trata de imágenes (fotogramas del video de un dron) y se debe diferenciar o reconocer los diferentes estados de una zona forestal: natural, con humo y con fuego; por tanto, se trata de un problema de clasificación de imágenes. Así, se optó por una red neuronal convolucional (CNN), ya que estas disponen de un enfoque más escalable en la detección de objetos en imágenes, esto mediante el uso de álgebra lineal debido a que se usa multiplicación de matrices para la identificación de ciertos patrones en el set de entrenamiento; en la figura 3 se ilustran algunas situaciones en donde implementar una CNN es una excelente opción.



Figura 3: Ejemplos de aplicación de una red neuronal convolucional.

Arquitectura

En un acercamiento inicial, se utilizó un modelo CNN de 8 capas totales: 3 capas convolucionales y 3 capas de max pool, seguidas de una capa densa de 512 neuronas precedida de una capa densa de 3 neuronas de salida: las clases de las imágenes a clasificar. Como se mencionó con antelación, este modelo resultó problemático por diversas razones; una razón pertinente es que el data augmentation resultaba en un conjunto de aproximadamente 13,000 imágenes, por lo cual se volvió

extremadamente ineficiente evaluar el modelo; otra es que las dimensiones de entrada de las imágenes era de (150, 150, 3), y resultaba en un modelo de 37 millones de parámetros. Por estos inconvenientes técnicos se decidió replantear la implementación.

La segunda CNN implementada dispone de un total de 10 capas. Las primeras dos constituyen las capas de Data Augmentation; se tiene una capa de Random Zoom con una proporción de 0.1, y un Random Flip en dirección horizontal para expandir la cantidad de datos de entrenamiento. Las siguientes 8 capas son idénticas al modelo anterior; 3 capas convolucionales con 16, 32 y 64 filtros de 3x3 por capa; 3 capas de max pooling con una matriz de 2x2; dos capas densas de neuronas: una de 512 y otra de 3 neuronas como salida del modelo. Las capas convolucionales y la capa densa de 512 neuronas utilizan una activación ReLu, mientras que la capa de salida de 3 clases utiliza una activación Softmax. Finalmente, se utiliza el optimizador Adaptive Moment Estimation (Adam) y una función de costo SparseCategoricalCrossEntropy. Finalmente, se implementaron dos Callbacks para registro del modelo: un callback de TensorBoard y otro callback llamado ModelCheckpoint configurado para guardar el mejor modelo con el criterio de precisión sobre el conjunto de validación..

Evaluación y Refinamiento

Primer modelo

Para la primera instancia del modelo, el cual utilizaba un dataset de 3306 imágenes –el primero mencionado en la sección de datos–, se logró una precisión del **99.22%** y un costo de **0.0899** para el conjunto de *entrenamiento*. En cuanto al conjunto de *validación*, el modelo obtuvo una precisión de **96.88%** y un costo de **0.0748**. Estos valores pueden apreciarse en la figura 3.

```
Epoch 29/40
16/16 - 18s - loss: 0.1376 - accuracy: 0.9531 - val_loss: 0.0689 - val_accuracy: 1.0000 - 18s/epoch - 1s/step
Epoch 30/40
16/16 - 20s - loss: 0.1142 - accuracy: 0.9453 - val_loss: 0.1293 - val_accuracy: 0.9688 - 20s/epoch - 1s/step
Epoch 31/40
16/16 - 18s - loss: 0.0899 - accuracy: 0.9766 - val_loss: 0.0655 - val_accuracy: 0.9688 - 18s/epoch - 1s/step
Epoch 32/40

Reached 98% accuracy so cancelling training
16/16 - 18s - loss: 0.0410 - accuracy: 0.9922 - val_loss: 0.0748 - val_accuracy: 0.9688 - 18s/epoch - 1s/step
```

Figura 3: Costo y exactitud en el proceso de entrenamiento de la primera red neuronal convolucional.

Adicionalmente, se puede visualizar la evolución tanto de la exactitud como de la función de costo en la figura 4.

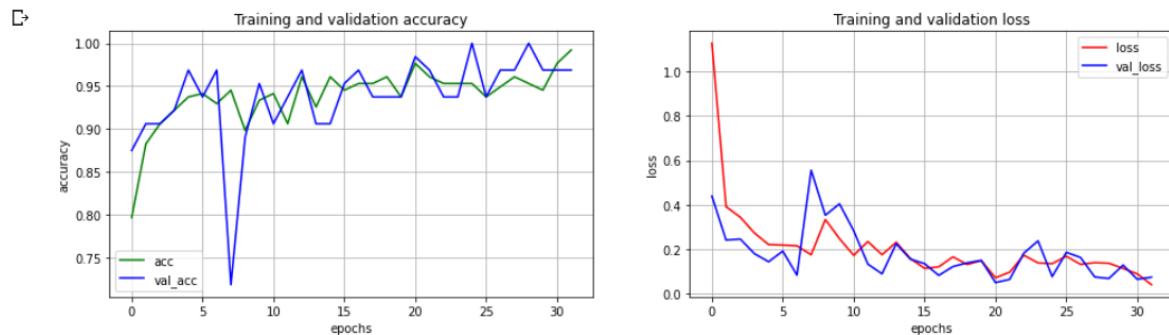


Figura 4: Evolución del costo y la exactitud del primer modelo (entrenamiento y validación) en función de las épocas computadas.

El problema de esta primera instancia y los datos que se utilizaron es que además de que eran un número muy grande para la limpieza, manipulación y evaluación de métricas, existían múltiples imágenes que requerían respetar licencias. Para un modelo válido y legítimo, consideramos que lo mejor sería buscar otras fuentes de datos, y así generar más métricas que validen la precisión y el uso de nuestro modelo.

Segundo Modelo: elección de métricas

Debido a que esta componente del proyecto se trata de un problema de clasificación, se escogieron diversas métricas relevantes para evaluar el desempeño del modelo. Primero, dada la naturaleza de la implementación, es obligatorio incluir la función de **costo** como un indicador del desempeño, pues es el valor que se busca minimizar para la red neuronal convolucional. Además, se incluye la métrica de **exactitud**, ya que indica la proporción de predicciones correctas de las 3 categorías; la **precisión**, porque indica la habilidad del modelo para identificar verdaderos positivos de cada clase; y la **exhaustividad**, pues sugiere qué tan bueno es el modelo para categorizar cada clase.

Exactitud y Costo: entrenamiento y validación

Se almacenaron las métricas de costo y exactitud de la red neuronal durante el proceso de entrenamiento; este proceso se realizó a lo largo de **40 epochs** con una **tasa de aprendizaje de 0.001**, que eran los mismos parámetros que en el primer

modelo. Se computaron dichas métricas con relación al conjunto de entrenamiento y el conjunto de validación. La evolución de estos valores se ilustra en la figura 5.

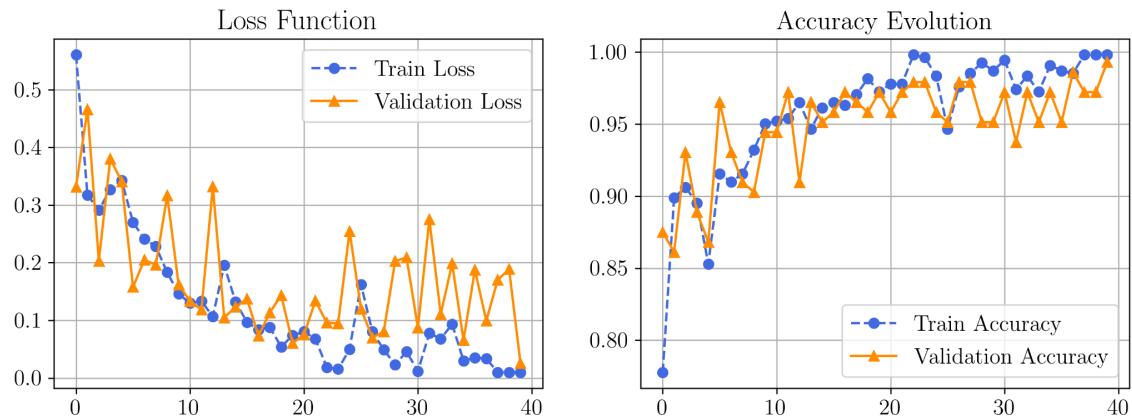


Figura 5: Evolución del costo y la exactitud del segundo modelo. Se observan muchas fluctuaciones en las curvas del conjunto de validación, en particular la función de costo.

Se logró una exactitud final del **99.82%** y un costo de **0.098** para el conjunto de *entrenamiento*. En cuanto al conjunto de *validación*, el modelo alcanzó una exactitud final de **99.31%** y un costo de **0.0256**. Estos valores indican un desempeño impecable del modelo; de esto se espera que el modelo demuestre un desempeño excelente en el conjunto de datos nunca antes vistos: el conjunto de prueba.

Evaluación del conjunto de prueba

El modelo logró una exactitud del **97%** en este conjunto; un valor ligeramente menor a la exactitud casi perfecta en los primeros dos; sin embargo, aún se considera como una exactitud muy buena. Complementando a la precisión, se obtuvo la matriz de confusión ilustrada en la figura 6.

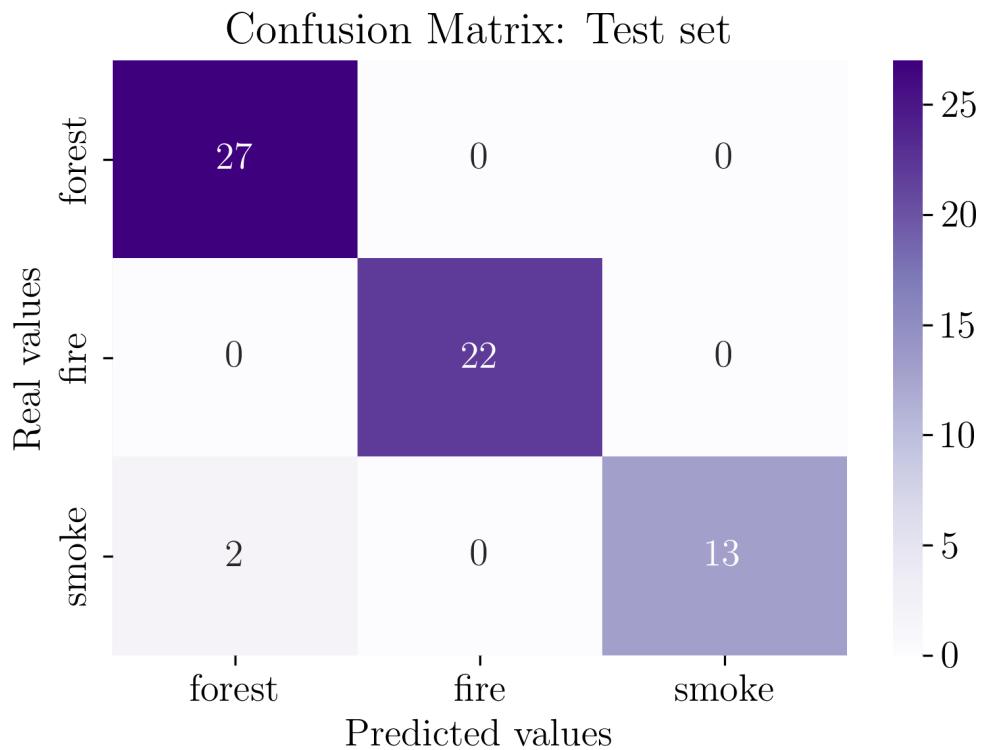


Figura 6: Matriz de confusión de las predicciones en el conjunto de prueba.

Resalta inmediatamente que el modelo tuvo un desempeño perfecto con la categoría *fire*, ya que todas las predicciones fueron acertadas; en contraste, el modelo indicó dos falsos positivos para la categoría *forest*, ya que el modelo predijo dos instancias de *forest* cuando en realidad era *smoke*; de manera complementaria, el modelo tuvo un menor desempeño para la clase *smoke*, pues en dos ocasiones el modelo predijo que la imagen era una instancia de *forest*.

Para evaluar cuantitativamente el conjunto de prueba, se calcularon las métricas de exactitud, precisión, y F-Score que están implícitas en la matriz de confusión anterior. Los valores de estos resultados se despliegan en la tabla 1.

	Precisión	Exhaustividad	F-Score
Forest	0.93	1.00	0.96
Fire	1.00	1.00	1.00
Smoke	1.00	0.87	0.93

Tabla 1: Métricas de desempeño del modelo para las 3 categorías de imágenes.

Se nota, como en la matriz de confusión, que la clase *fire* tiene una evaluación perfecta. En contraste, las clases *forest* y *smoke* se ven afectadas por las predicciones del modelo; por ejemplo, la precisión de *forest* se ve golpeada ya que el modelo indicó dos falsos positivos entregando un puntaje de 27/29 imágenes. La exhaustividad de *forest* es perfecta ya que el modelo no predice ningún falso negativo. Para la clase *smoke* se nota una precisión perfecta pero, en términos de exhaustividad, el modelo predice dos falsos negativos obteniendo un puntaje del 87% –mismas clasificaciones que afectan la precisión de *forest*–.

Adicionalmente, se calcularon los valores promedios y promedios ponderados de las métricas anteriores, los cuales se muestran en la tabla 2.

	Precisión	Exhaustividad	F-Score
Promedio	0.98	0.96	0.96
Promedio Ponderado	0.97	0.97	0.97

Tabla 2: Métricas de desempeño promedio del modelo.

Observando estos promedios y tomando en cuenta la baja cantidad de elementos que se tuvieron para la clase con el peor puntaje, se consideran estas métricas como excelentes. Sin duda, en un contexto real, las diferencias entre un F-Score promedio del 96% y uno del 98% serían virtualmente indistinguibles.

Predicciones

Se graficaron algunas predicciones del modelo para el conjunto para visualizar algunos ejemplos y, en particular, uno de los dos ejemplos que clasificó mal el modelo. Las imágenes con sus respectivas predicciones se ilustran en la figura 7.

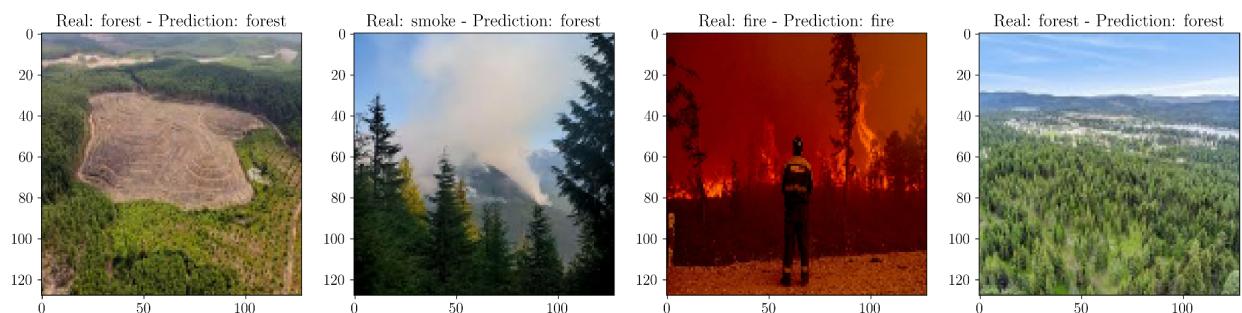


Figura 7: Ejemplos de predicciones del modelo sobre el conjunto de prueba. Estas son imágenes nunca antes vistas por el modelo. Se distingue una predicción errónea para la segunda imagen.

Características extraídas

Se pueden extraer las primeras capas del modelo ya entrenado para observar las características que se están extrayendo de las convoluciones realizadas. Se comienza graficando la imagen cuya predicción fue errónea,

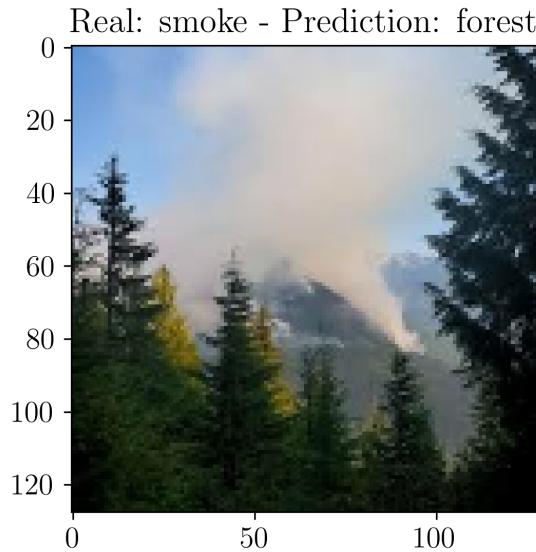


Figura 8: Ejemplo de predicción errónea del conjunto de prueba. El modelo indicó que se trata de una instancia de bosque mientras que la clase verdadera es *smoke*.

y posteriormente alimentarla a las primeras capas del modelo para observar el resultado de los kernels, es decir, los mapas de características. Esto sugiere un poco lo que la red convolucional está “viendo” al procesar cada una de las imágenes.

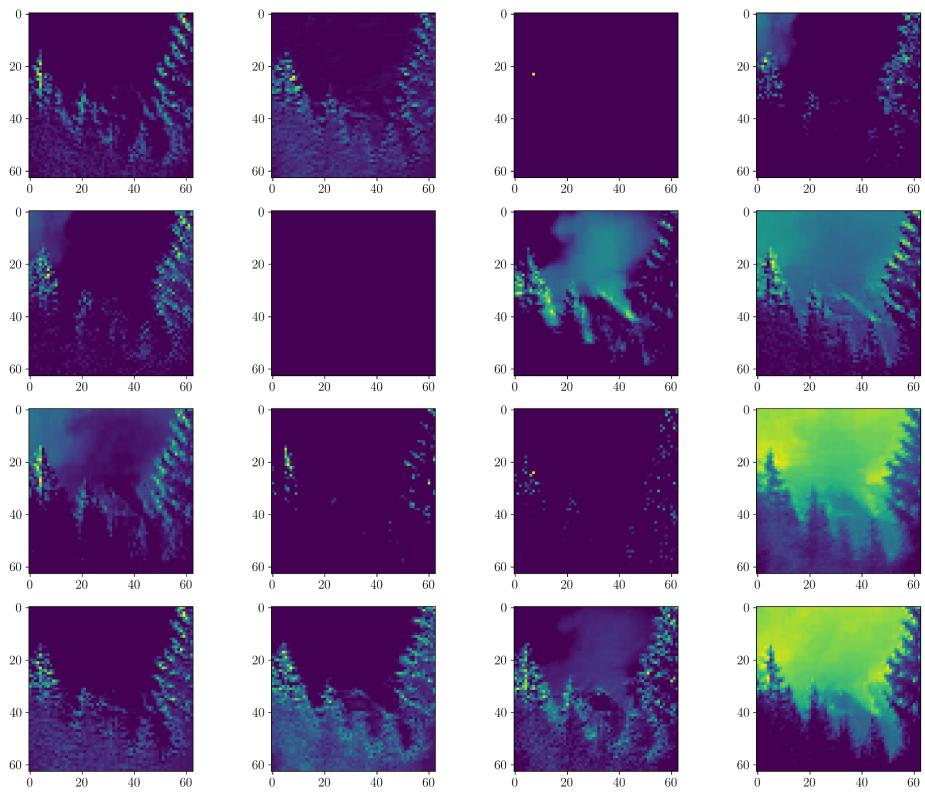


Figura 9: Características obtenidas de una predicción incorrecta.

Por simple inspección resaltan las imágenes (3,4) y (4,4) ya que se observa claramente el color en el mapa que indica el peso que la red le da al cielo, donde se puede encontrar el humo; además, la imagen (2,3) se notan más aislados los píxeles del humo. No obstante, la mayoría de los mapas identifican los bordes de los árboles; sólo 4 mapas no hacen totalmente evidente estas figuras. Para complementar, se ilustra un ejemplo de una predicción correcta del modelo para una imagen que contiene fuego; es decir, de la clase *fire*.

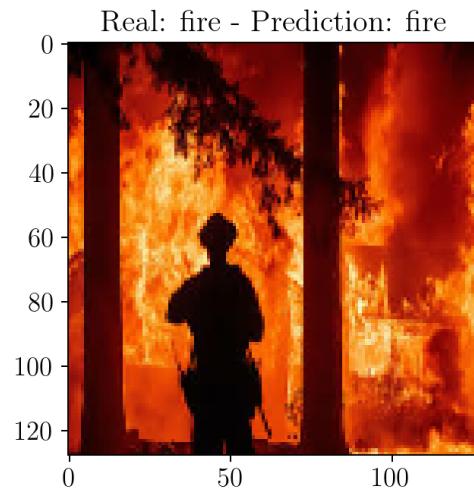


Figura 10: Ejemplo de predicción correcta del conjunto de prueba. Se resalta la diferencia de la categoría con el ejemplo anterior.

Tras observar la imagen de entrada, se vuelve a alimentar a las primeras capas de la red para extraer los mapas de características de la red entrenada. Obsérvese la intensidad del fuego y las siluetas que genera.

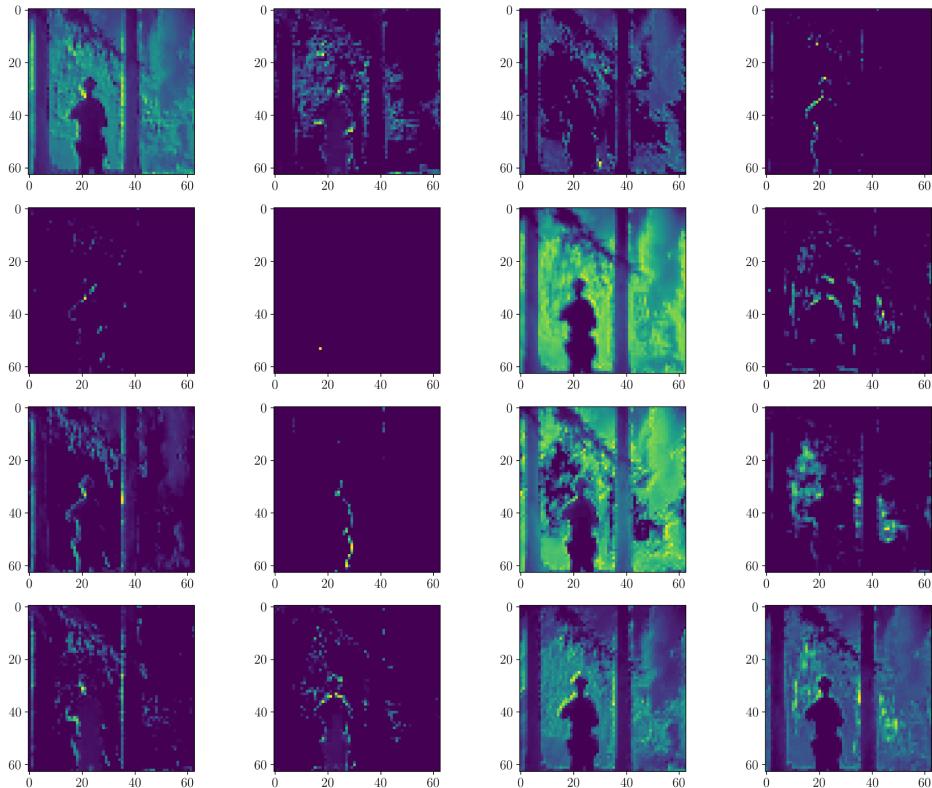


Figura 11: Características obtenidas de una predicción correcta.

Se pueden observar algunas características que son muy claras, como por ejemplo, el fuego. En la predicción correcta, se puede ver en la imagen (3,3) y en la imagen (3,2) que la red puede ver de manera clara el fuego. Si se inspecciona la imagen en la que la predicción no fue correcta es posible ver que no encuentra características de fuego, pero sí encuentra humo, en las imágenes (4,3) y (4,4), esto debería ayudar a la red a dar un resultado correcto, pero no lo hace, la característica que podría estar confundiendo a la red es la característica que se encuentra en la imagen (1,2), que en la imagen del fuego podemos ver que no visualiza información, pero en la imagen de la predicción errónea visualiza el bosque. Tomando en cuenta este análisis sobre las características que ve la red, se puede llegar a entender porque el error ha sido incorrecto.

Implementación

Objetivo

Se busca que el modelo se pueda implementar en los drones que visitarán las zonas de riesgo para poder detectar lo que sucede en ellas, en tiempo real.

Limitaciones

Por el momento no se cuenta con acceso a estos drones por lo que se implementará el modelo para que pueda detectar los tres casos que tenemos (*smoke*, *fire* y *forest*), en la webcam de una computadora en tiempo real.

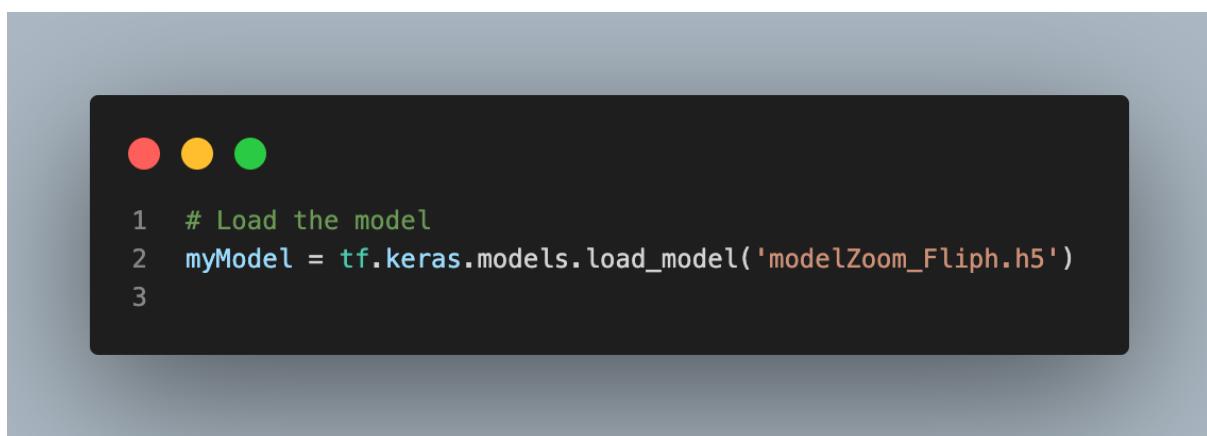
Implementación

El modelo está implementado en un script de Python, lo cual da facilidad para que pueda funcionar en distintos sistemas operativos, pensando en que en un futuro se buscará su implementación en drones.

Para la implementación del modelo se usa la librería OpenCv que permite capturar en tiempo real los frames de una cámara conectada al sistema operativo.

Funcionamiento del programa

Primero se debe cargar el modelo ya entrenado para que se pueda utilizar.



```
● ● ●
1 # Load the model
2 myModel = tf.keras.models.load_model('modelZoom_Flip.h5')
3
```

Figura 12: Código de la implementación del modelo; se carga el modelo utilizando la biblioteca de TensorFlow.

Después se crea la función de preprocesamiento para cada frame que se va a analizar.

```

1 # Preprocessing frame
2 def modelResult(frame):
3     #OpenCv uses BGR and the model is trained on RGB
4
5     # Resizing, adding a dimension and changing BGR to RGB
6     frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
7     frame = cv2.resize(frame,(128,128),3)
8     #frame = cv2.resize(frame,(150,150),3)
9     frame = np.expand_dims(frame, axis=0)
10
11    #Here we get the result
12    result = myModel.predict(frame).argmax(axis=1).astype(int)
13    classes = ['forest', 'fire', 'smoke']
14    return classes[result[0]]

```

Figura 13: Código de la implementación del modelo; función para preprocesar de cada fotograma de video.

Finalmente se crea el bucle while para hacer predicciones de los fotogramas de la cámara en tiempo real, desplegando el texto de la predicción en la pantalla.

```

1 #Open the video capture with a 0 to use the webcam
2 cap = cv2.VideoCapture(0)
3
4 while True:
5     ret, frame = cap.read()
6     mensaje = modelResult(frame)
7     #Creating the frame with the result as a text
8     image = cv2.putText(frame,mensaje, (200,200), cv2.FONT_HERSHEY_SIMPLEX, 3, (0,0,0),10)
9     #Displaying the new frame with the text
10    cv2.imshow("Modelo de detección de incendios",image)
11    if cv2.waitKey(1)==ord('q'):
12        break

```

Figura 14: Código de la implementación del modelo; bucle donde se abre la cámara, se preprocesa el fotograma y se realiza la predicción.

Manual de usuario

Para poder utilizar el proyecto se requiere lo siguiente:

- Python 3.7 en adelante
- Librería OpenCv
- Librería Numpy
- Librería Tensor Flow
- Cámara

Se debe de tener el modelo guardado en formato h5, una vez cumpliendo los requerimientos anteriores, se puede usar el programa.

El programa solo requiere iniciar y en automático mostrará la imagen que está viendo en ese momento con el resultado que está obteniendo después de pasar por el modelo.

Zonas de riesgo

Metodología

Business Understanding

En esta etapa fue muy importante la comunicación con Green Tech Innovation, para tener claridad de lo que se buscaba lograr y poder obtener los requerimientos del proyecto, que se vieron reflejados en el documento de requerimientos.

Data Understanding

Para lograr los requerimientos necesarios fue fundamental entender la mejor manera de utilizar los datos que teníamos disponibles, como los datos geoespaciales y satelitales proporcionados principalmente por la NASA.

Data Preparation

Para la preparación de los datos se realizaron diferentes procesos para cada caso, por ejemplo, para aplicar el Random Forest de manera correcta, se generaron puntos de calor aleatorios por toda la zona de Nuevo León y así comparar con los valores reales de puntos de calor obtenidos por el sistema de monitoreo de incendios forestales CONABIO.

Modeling

Cuando los requerimientos y el dataset se encontraban definidos, se pudo tomar la decisión de la selección del modelo para cumplir con lo que solicitaba Green Tech Innovation. En este caso se utilizó un Random Forest.

Evaluation

Después de tener el modelo se revisaron las métricas de precisión del modelo, con ayudas visuales como gráficas.

Deployment

Para la implementación se tomó la decisión de hacerla lo más cercana posible a lo que Green Tech Innovation quiere lograr. Una manera fácil de interactuar con el mapa, en este caso una interfaz web.

Datos

El modelo desarrollado para el actual proyecto se alimenta de distintas fuentes de datos. En particular, se trata con bases de datos geoespaciales, principalmente en formato shapefile (.shp) de potenciales variables que pueden ayudar a detectar zonas de riesgo de incendio forestal. Se determinó que, variables como temperatura, emisividad y reflectancia de la superficie de la Tierra, índices de vegetación, redes viales y datos demográficos servirían principalmente para predecir los puntos de calor (otro conjunto de datos a conseguir) que tiende a representar incendios forestales. Dichas variables se consiguieron mediante las siguientes fuentes de datos:

- **Temperatura y emisividad de la superficie terrestre, 8 días Global 1km (MODIS/061/MOD11A2)**
- **Mapeo Global Satelital de Precipitaciones (GSMap)**
- **Datos de Elevación Digital SRTM 30m**
- **Reflectancia de superficie de la Tierra 8 días Global 500m (MODIS/061/MOD09A1)**
- **Índices de vegetación de Terra 16 días Global 500 m (MODIS/061/MOD13A1)**
- **Sistema de pronóstico global 384 horas de datos de atmósfera pronosticados (GFS)**
- **Sistema de información de incendios para el manejo de recursos (FIRMS)**
- **Puntos de Calor registrados en Nuevo León en un rango de fecha (CONAFOR y CONABIO)**
- **Puntos de Calor aleatorios en la región de Nuevo León**

En cuanto a las herramientas empleadas en el proyecto, la investigación previa permitió determinar dos principales: primero, el software QGIS, aplicación que permite visualizar, editar, imprimir y analizar datos geoespaciales; segundo, la plataforma Google Earth Engine.

Herramienta	Uso	Razón

	<p>Análisis y manipulación de datos no satelitales, como los puntos de calor y los datos demográficos, para aplicar algoritmos como distancia euclídea y generar nuevas capas de datos.</p>	<p>Su funcionamiento fue explicado en un tutorial por parte del INEGI, lo cual nos facilita usar sus datos en el mismo software.</p>
 Google Earth Engine	<p>Análisis y manipulación de datos satelitales, para aplicar algoritmos como RandomForest y generar capas nuevas que permitan predecir zonas de riesgo de incendios.</p>	<p>Su capacidad de extraer, visualizar y manipular datos satelitales es muy rápida y eficaz debido al uso de Google Cloud.</p>

* Debido a que todos los datos son procesados con tecnologías Cloud (GEE y Colab) y se actualizan cada cierto tiempo, no es necesario o recomendable el almacenamiento local.

Tabla 3: Herramientas utilizadas para el modelo

Modelo

El modelo para la predicción de zonas de riesgo de incendio se basó en un Random Forest, la cual fue implementada con la función de `ee.Classifier.smileRandomForest` por parte de Google Earth Engine.

Para cada una de las capas que se extraen de datos satelitales, se necesitó extraer solamente las “bandas” o propiedades importantes en nuestro modelo, como el NDVI, el cual se extrae la media de valores para el índice de NDVI en todo el mes, además de enmascarar por nubes debido a que las nubes en las imágenes satelitales pueden obstruir los valores necesarios para el modelo.

```

var maskClouds= function (image){
  var quality= image.select("SummaryQA");
  var mask= quality.eq(3).not();
  return image.updateMask(mask);
};
var NDVI_mask=modisNDVI.map(maskClouds)
  .select("NDVI")
  .min();
  
```

Figura 15: Manipulación de NDVI

Para nuestra función se definió un valor de 50 árboles aleatorios, siendo entrenado por la importante fusión de diferentes capas de datos satelitales que tenemos, además de los puntos de calor históricos en la zona de Nuevo León.

```
var Mosaico_capas = DEM.addBands(Pendiente)
    .addBands(Media_T)    .addBands(NDVI_mask)
    .addBands(sumaP)
    .addBands(Vial)
    .addBands(Rmedia);
//2. NOMBRAMIENTO Y SEPARACIÓN DE BANDAS
var Nombre_bandas = Mosaico_capas.bandNames();
```

Figura 16: Fusión de variables

```
var clasificador = ee.Classifier.smileRandomForest(50)
    .train(regiones_entrenamiento, 'Real', Nombre_bandas)
    .setOutputMode('PROBABILITY');
```

Figura 17: Creación de Random Forest

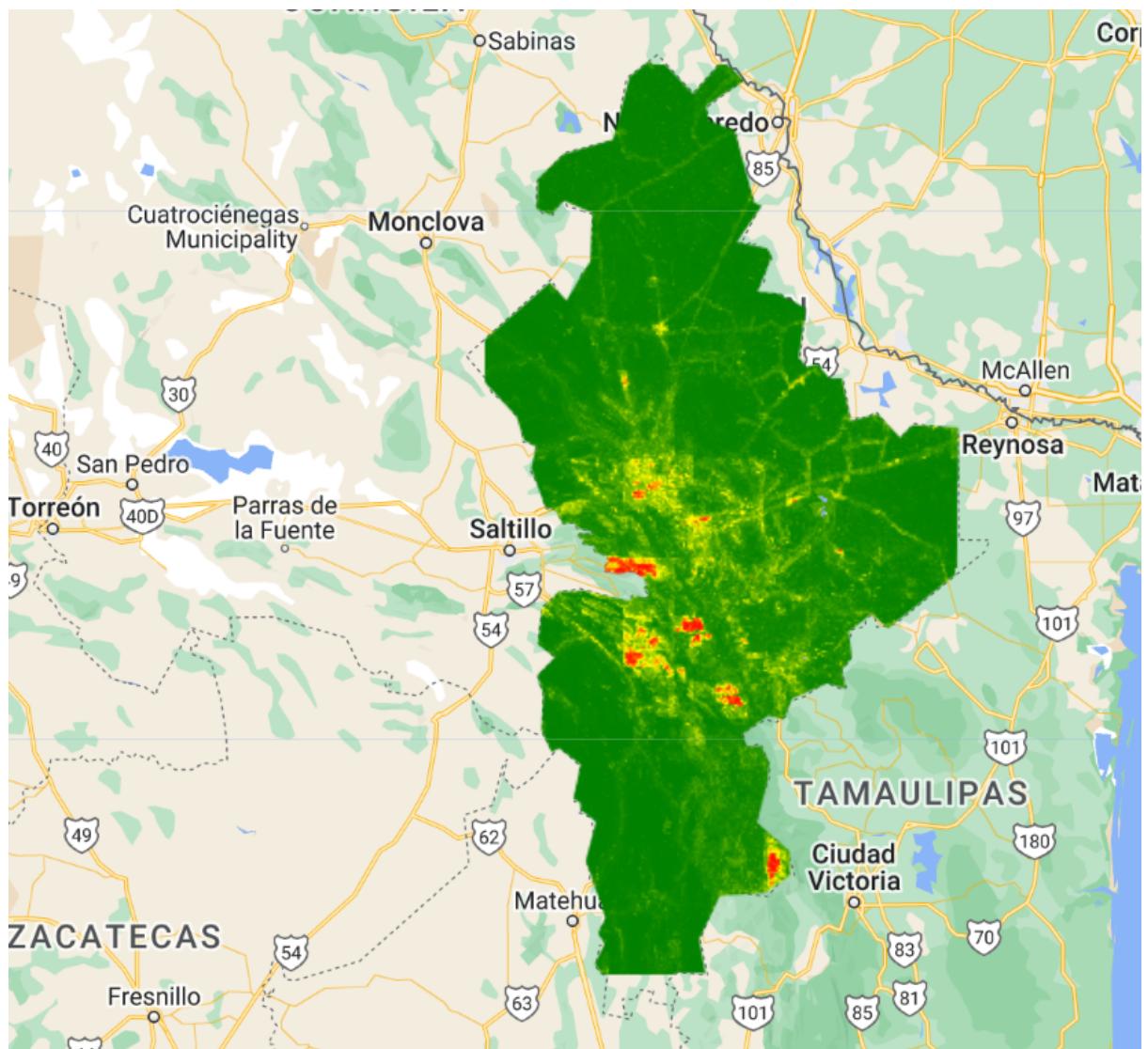


Figura 18: Mapa de riesgo de incendio para el mes de Marzo 2021.



Google Earth Engine

Datos de entrada

- 🌡 Temperatura y emisividad (MODIS/061/MOD11A2)
- 🌧 Precipitación (GSMaP)
- 🏔 Datos de Elevación Digital SRTM
- 🌐 Reflectancia de superficie de la Tierra (MODIS/061/MOD09A1)
- 🌿 Índices de vegetación (MODIS/061/MOD13A1)
- 🌪 Datos de atmósfera pronosticados (GFS)
- 🔥 Sistema de información de incendios para el manejo de recursos (FIRMS)
- 🔥 Puntos de Calor registrados en Nuevo León en un rango de fecha (CONAFOR y CONABIO)
- 📊 Datos demográficos y de redes viales (INEGI)

Procesamiento

Importación de datos en la plataforma de Google Earth Engine

Limpieza de variables

Fusión de variables

Random Forest Probabilístico

Algoritmo de distancia euclíadiana en QGIS

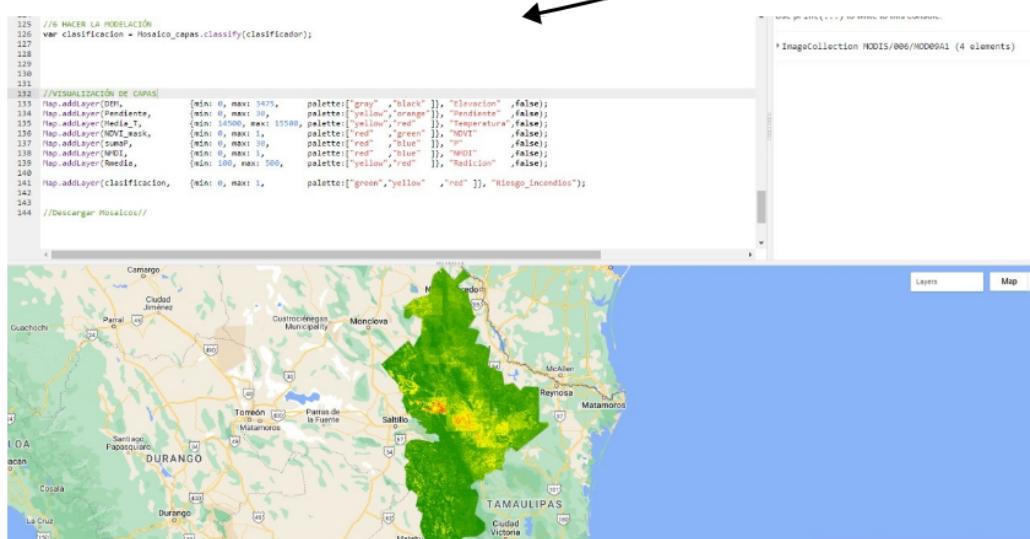


Figura 19: Diagrama de flujo del modelo.

Evaluación

El segundo modelo a implementar, para conocer la predicción de zonas de riesgo con datos geoespaciales a partir de un algoritmo Random Forest, también entrega algunas métricas de evaluación, aunque en este caso es muy difícil obtener un número considerable de métricas debido a las limitaciones de Google Earth Engine. En particular, pueden resaltar la importancia de las variables del modelo, el puntaje y el error OOB.

OOB Score y OOB Error

OOB Score es una métrica de validación muy potente que se utiliza especialmente para el algoritmo Random Forest en el cual está basado nuestro modelo para obtener resultados de mínima varianza y, por lo tanto, hace un modelo predictivo mucho mejor que un modelo que usa otras técnicas de validación. El error OOB es el número de clasificación incorrecta de la muestra OOB. En nuestro caso el error es muy bajo, lo cual indica un buen desempeño del modelo.

```
numberOfTrees: 50  
outOfBagErrorEstimate: 0.06349206349206349
```

Figura 20: Valores de error en el modelo

Importancia de Variables

La disminución media del coeficiente de Gini es una medida de cómo cada variable contribuye a la homogeneidad de los nodos y hojas en el bosque aleatorio resultante. Cuanto mayor sea el valor de la precisión de disminución media o la puntuación de Gini de disminución media, mayor será la importancia de la variable en el modelo.

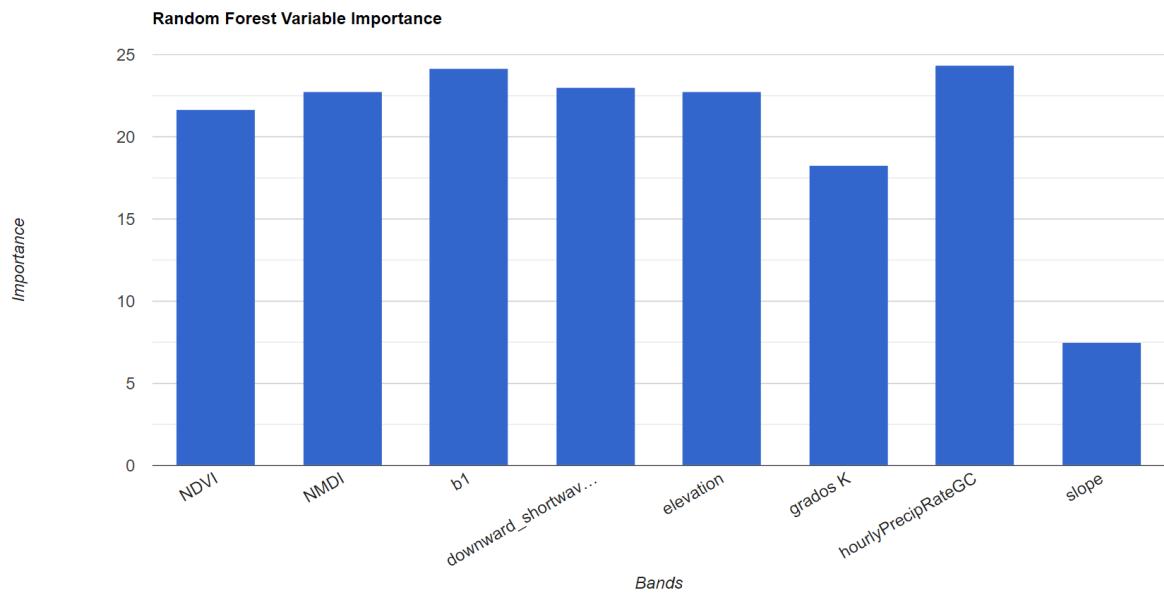


Figura 21: Importancia de variables en el modelo de Random Forest para el mes de Marzo 2021.

Implementación

Objetivo

Se busca que el modelo se pueda visualizar de manera sencilla y rápida para cualquier usuario de Green Tech Innovation.

Limitaciones

En un futuro se buscará continuar en el desarrollo de una plataforma web más completa, ya que por el momento existen varias limitaciones que se traducen en menos elementos de interacción en la plataforma.

Implementación

El modelo está implementado en un script de Google Earth Engine, lo cual hace muy fácil la publicación de la aplicación en la plataforma de Earth Engine Apps.

<https://fire-fighter.users.earthengine.app/view/prediccion-de-incendios-forestales-en-nuevos-lados>

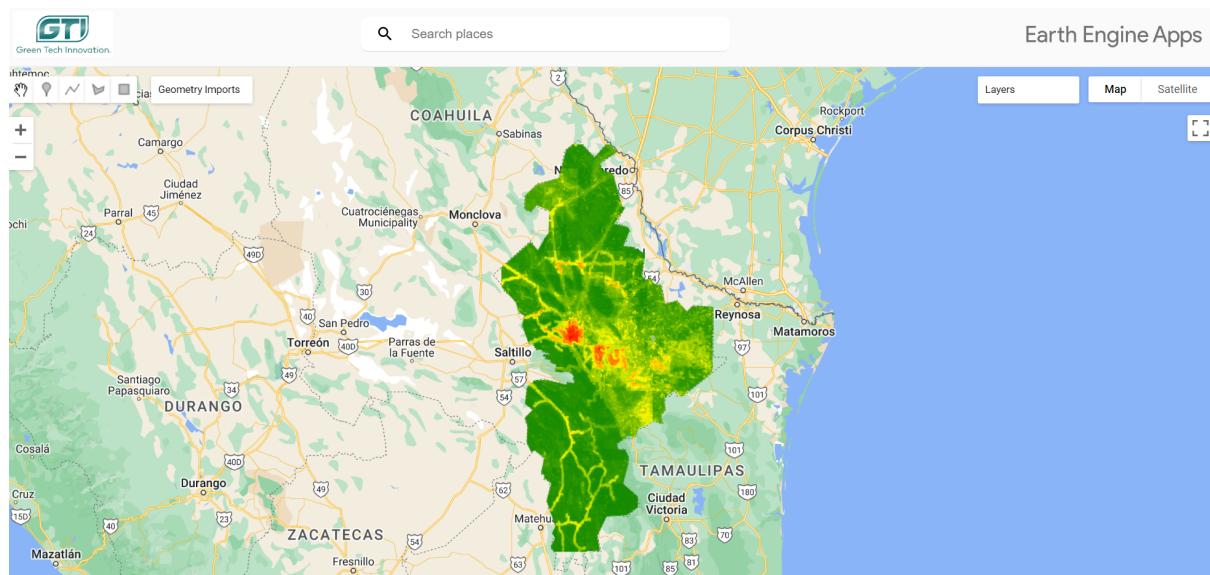


Figura 22: Página web que muestra el mapa de zonas de riesgo de incendios.

Presentación final

https://youtu.be/Fq_DmQHMi54