Informe para auditoria: Big Eye BA, Sprint 3

Objetivos Iniciales.

- ✅ Regenerar el dataset con datos del Catastro filtrados.
- ✅ Mejorar el preprocesado de los datos de Copérnico.
- ❌ Obtener datos reales de nubes de puntos.
- ❌ Optimizar los hiperparámetros del modelo de segmentación.
- ⚠️ Optimizar los hiperparámetros del modelo de TCN.
- ❌ Optimizar los hiperparámetros del modelo de clasificación.
- ✅ Investigación sobre el despliegue de modelos de genéricos.
- ✅ Investigación sobre el despliegue de modelos de pytorch.
- ❌ Investigación sobre el despliegue de modelos de tensorflow.
- ⚠️ Desplegar los modelos en un entorno productivo.
- ✅ Creación de una herramienta de visualización de mapas interactiva.
- ⚠️ Integrar el modelo de segmentación en la interfaz gráfica.
- ✅ Integrar el modelo de TCN en la interfaz gráfica.
- ⚠️ Integrar el modelo de clasificación en la interfaz gráfica.
- ❌ Utilizar los resultados del equipo de Big Eye BI.

✅ Completado con éxito. ⚠️ Surgieron problemas. ❌ No se completó.

Problemas que han surgido.

- El refinamiento de los datos obtenidos del Catastro resultó ser más complejo de lo esperado. En un principio solo iba a ser necesario:
    1. Concatenar los diferentes archivos shapefile de cada uno de los municipios para poder generar correctamente las máscaras de zonas que incluyeran la vista aérea de más de un municipio.
    2. Aumentar la resolución de las imágenes solicitadas al PNOA.
    3. Filtrar irregularidades notables como la base aérea o el aeropuerto, que introducían información al conjunto sobre falsos edificios de gran extensión.

    No obstante, resultó que había muchas más irregularidades. Las piscinas, campos de futbol, aparcamientos o sótanos, entre muchos otros, estaban catalogados como edificaciones.

    - Solución: Después de varias aproximaciones se optó por hacer un preprocesado de los shapefile utilizando la propiedad “CONSTRU” que tenía codificada en forma de números romanos y otras siglas información sobre la altura de cada parcela (en número de plantas).
- La investigación sobre el despliegue de modelos de PyTorch reveló que el proceso sería más complejo de lo estimado inicialmente. La librería de PyTorch es principalmente usada en entornos académicos y de investigación, por lo que las herramientas de despliegue son algo básicas. Como aclaración, estamos obligados

a utilizar PyTorch porque el modelo original de SAM está diseñado usando esta librería.

Se ha probado a crear tanto una API de inferencia genérica (utilizando las funcionalidades de MLFlow), como una dedicada a modelos de PyTorch (utilizando el módulo de torch serve). Cada una tiene sus ventajas e inconvenientes.

  - El servicio de MLFlow permitiría desplegar todos los modelos por igual, aprovechando directamente los resultados guardados en los artefactos de MLFlow. Sin embargo, no contempla el preprocesado de las entradas o el postprocesado de las salidas; por lo que sería una tarea que realizar en nuestro backend, lo que reduce la escalabilidad. Por otro, y principal motivo de no haber elegido esta tecnología, es que no permite la entrada de datos complejos no serializables a json. Es decir, se pueden hacer peticiones a su API de inferencia mediante todo tipo de información en un json, mediante un array de numpy o mediante un tensor de pytorch. Lo que permite el correcto funcionamiento de múltiples modelos. Sin embargo, el modelo de SAM requiere de un tensor para la imagen y un "prompt" para hacer predicciones. Esta combinación no se puede llevar a cabo, y no se puede crear una serialización propia del tensor de la imagen porque la herramienta no permite preprocesado para descodificar el tensor.
  - Por otro lado, el servicio de PyTorch si permite el preprocesado y postprocesado de las entradas y salidas. Pero al ser exclusivo de esta librería obligaría a crear otros servicios para el resto de los modelos usando otras tecnologías. Sin embargo, la cantidad de información es escasa, la herramienta para debug está aún en desarrollo y la mayoría de los errores se reflejan después de haberse producido en la imposibilidad de acceder a archivos temporales porque los procesos afectados por los fallos no han llegado a cerrarlos. Todo esto retrasó en gran medida el desarrollo usando esta tecnología.
    - Solución: El despliegue de los modelos se realizará temporalmente como un servicio provisto localmente desde nuestro backend.
- Diversas dificultades, ajenas a la asignatura de proyectos, imposibilitaron la investigación sobre el despliegue de modelos de TensorFlow en un entorno productivo. Dichas dificultades personales tienen un carácter puntual y no afectarán significativamente el avance del proyecto hasta la presentación con el cliente.
    - Solución: Se ha usado la misma estrategia de despliegue que para el modelo de segmentación.
- La librería específica usada para la creación del modelo de TCN no tiene un módulo propio para el despliegue en producción. Desde la fase de investigación de predicción de series temporales hasta ahora ha salido una nueva versión de PyTorch que permitiría, posiblemente, hacer el mismo desarrollo usando esta librería en lugar de Dart. Esto facilitaría el despliegue del modelo de TCN. No se ha podido llevar a cabo debido a la falta de tiempo ocasionada por la falta de previsión sobre la carga de trabajo asociada al desarrollo frontend y backend.
    - Solución: Se ha usado la misma estrategia de despliegue que para el modelo de segmentación.

- La falta de tiempo ha impedido realizar la comunicación de nuestros servicios con los del equipo de Big Eye BI en ninguno de los aspectos previstos. Tampoco ha sido posible crear una herramienta capaz de obtener información sobre la geometría de edificios en España debido a este mismo motivo. Aunque es cierto que el enfoque que se tenía previsto (reconstruir las nubes de puntos con la información de la altura de las parcelas) tenía muchas posibilidades de incurrir en la pérdida de información relevante para el desempeño del modelo, por lo que no se priorizó esta tarea.

Detalle de las tareas finales realizadas.

- **Filtrado de los datos del Catastro:** Como se ha mencionado, ha sido necesario regenerar el dataset para el modelo de segmentación de edificios debido a datos incoherentes que causaban ruido. Para ello se ha probado a eliminar los edificios "rurales" y dejar solo los "urbanos". Esto no fue suficiente e hizo falta eliminar edificios basándose en la cantidad de plantas de cada una de las parecelas. Este proceso comenzó a ser más complejo y por recomendación del profesor, se ha planteado la posibilidad de estandarizar el proceso y permitir que se ejecute usando los flows de Prefect. Aún así, los resultados siguen siendo nefastos.
- **Investigación sobre la puesta en producción de los modelos:** Como se ha comentado, se han probado las tecnologías de MLFlow y torch serve para el despliegue de los modelos. No se ha conseguido un producto que se adapte a las necesidades del proyecto. No se ha llegado a investigar la forma de desplegar un modelo de TensorFlow (más allá de lo comentado sobre el despliegue genérico usando MLFlow).
- **Creación de un servicio de inferencia para el modelo de segmentación de edificios:** Para solucionar temporalmente el problema de las puestas en producción se ha abstraído un método de predicción que reciba los inputs desde el frontend, solicite la información necesaria al Catastro y devuelva las predicciones postprocesadas aptas para su visualización en la web. El objetivo final es sustituir la llamada a este método temporal por la llamada a la API de inferencia.
- **Mejorar el preprocesado de los datos de Copérnico:** Se mejoró el dataset de datos de Copérnico para intentar tratar los outliers provocados por los días nublados. Para ellos de aplicó una interpolación cúbica a cada serie temporal. Después se aplicó un filtro de medias móviles para suavizar esos valores atípicos. Teniendo limpiadas varias series temporales, se entrenó el modelo TCN con todas ellas.
- **Entrenamiento de TCN**: se han refinado algo los hiperparámetros para intentar que haga la mejor predicción posible. Los resultados aún son mejorables, por lo que se va a seguir entrenando el modelo para mejorar sus resultados todo lo posible.
- **El despliegue de los modelos se ha realizado utilizando Flask:** para manejar temporalmente las peticiones del cliente se han desplegado los modelos en el propio backend. Para cada modelo, hemos preparado una petición POST que envía a la API las coordenadas sobre las que se quiere predecir y una petición GET para recibir de la API la predicción realizada por el modelo. Estos modelos se han cargado en Python y el preprocesamiento y la predicción se han realizado en el backend. Hemos conseguido conectar el modelo TCN y el modelo de segmentación y ahora estamos en el proceso de conectar el modelo de

clasificación. Además, vamos a mejorar la interfaz del usuario para que sea más intuitiva.

- **Visualización en web de nubes de puntos:** Se ha creado una visualización interactiva usando React para las nubes de puntos, que ya pueden cargarse desde el backend.
- **Mejora en el modelo de clasificación de nubes de puntos:** se han hecho cambios en el dataset y su preprocesado.
- **Creación de una User Interface que integre las funcionalidades de los tres modelos**: El equipo ha creado una interfaz de usuario utilizando React JS sobre un mapa de la librería LeafLet. En ella, encontramos una leyenda en la parte superior derecha que permite activar y desactivar las distintas visualizaciones que ofrece nuestro proyecto, como puede ser el NDVI. En el caso del modelo de segmentación y clasificación, el usuario tiene la posibilidad de crear un polígono en el mapa para delimitar la región en la que quiere que el modelo trabaje. El front envía una petición al modelo y, tras recibir la resolución de este, obtiene en el mapa una nueva capa en forma de ráster para visualizar el resultado. Tras ello, el usuario puede pinchar sobre un edificio para dirigirse a una visualización de nube de puntos del mismo, además de visualizar el resultado obtenida por el modelo de clasificación.

Próximas tareas

- **Continuar con el despliegue en producción:** Es necesario dedicar más tiempo a la investigación de otras tecnologías y/o solucionar los problemas que han surgido sobre las tecnologías empleadas. También se ha de tomar aún la decisión de cómo desplegar el modelo de TCN, si reconstruirlo usando PyTorch o si es mejor buscar la forma de poner en producción el modelo actual de Dart. Para el modelo de clasificación de nubes de puntos será además necesario crear los procedimientos para hacer peticiones de predicción desde la UI.
- **Refinamiento de los modelos:** Es necesario mejorar el rendimiento de todos los modelos creados hasta ahora. Especialmente el de segmentación de edificios.
- **Mejora de la UX:** Retocar algunos aspectos de la interfaz gráfica para que sea más amigable.
- **Puesta en producción del preprocesado de Catastro:** El preproceso de los datos catastrales ha sido local y rudimentario, por lo que se van a crear los procesos necesarios para que sea un proceso replicable y más eficiente.