Informe para auditoria: Big Eye BA, Sprint 2

Objetivos Iniciales.

- ✅ Creación de un dataset con datos del Catastro.
- ✅ Investigación sobre modelos de segmentación.
- ⚠️ Crear o hacer fine tuning de un modelo básico de segmentación.
- ❌ Visualizar la segmentación de edificios.
- ⚠️ Despliegue de infraestructura MLOps (con MLFlow y Prefect).
- ✅ Refinamiento del modelo TCN inicial.
- ❌ Encontrar un enfoque para el modelo TCN.
- ⚠️ Investigar fuentes de datos de nubes de puntos de edificios.
- ❌ Obtención automatizada de nubes de puntos.
- ✅ Investigación sobre modelos de clasificación de nubes de puntos.
- ✅ Implementación de un primer modelo de nubes de puntos.
- ✅ Visualización de los resultados de nubes de puntos.

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

Problemas que han surgido.

- Dificultades en el entrenamiento del modelo de segmentación de instancias: las limitaciones de velocidad de procesamiento de los dispositivos personales ralentizan las operaciones de preprocesamiento (rasterizado de los archivos shapefile de catastro para obtener máscaras de edificios) y obliga a utilizar el entorno de Google colab para el entrenamiento del modelo. Esto dificulta la integración con la arquitectura MLOps.
- Dificultades en el despliegue de Prefect como herramienta de control del ciclo de vida del aprendizaje automático: Prefect presenta algunas limitaciones que habían pasado inadvertidas en el primer sprint. La primera es que cada cuenta puede tener un máximo de un colaborador que acceda a sus workpools. La segunda es la incapacidad de gestionar workers de diferentes sistemas operativos (y coincide que los tres dispositivos personales en el equipo utilizan tres sistemas operativos distintos).
    - Solución: Al primer problema se ha decido crear una cuenta común he iniciar todos sesión desde ella para acceder a la API. Al segundo problema se ha renunciado a crear un “cluster” gestionado por Prefect. De esta manera cada worker trabaja exclusivamente en el deployment creado con su mismo sistema operativo.
- Dificultades en la obtención de nubes de puntos de edificios en España: de las opciones estudiadas la mayoría eran incompatibles con los requisitos. Algunas eran difícilmente automatizables para obtener grandes cantidades de datos (obtención manual desde el software QGis), tenían poca resolución o cantidad de puntos por metro cuadrado, o, presentaban ambos defectos. La única fuente de datos que cumpliera con nuestros requisitos era una api de Google de pago.

- Solución: Se ha optado por utilizar un dataset de nubes de puntos ya generado (buildingnet). Esto ha permite paralelizar las tareas de creación de un primer modelo con la continuación de la búsqueda de fuentes de datos españoles con los que entrenar el modelo o refinar el creado con estos datos.

Detalle de las tareas finales realizadas.

- **Creación de un dataset con datos del Catastro:** Se automatiza la creación de un dataset para el modelo de segmentación de instancias. Para ello se ha utilizado el proyecto PNOA (Plan Nacional de Ortografía Aérea) como fuente de datos de mapas base en formato GeoTiff y la información vectorial sobre la construcción de edificios obtenida directamente del Catastro para la Comunidad de Madrid en formato shapefile. La información vectorial ha sido rasterizada para obtener las máscaras asociadas a cada mapa base. Aún se están solucionando problemas con el preprocesamiento como municipios limítrofes que aparecen en más de un mapa base pero no en sus versiones vectoriales (puesto que cada una se limita a un solo municipio).
- **Investigación de los modelos de segmentación:** Se encontró un proyecto de segmentación prometedor, SAMGeo, que implementaba todas las fases de proceso: obtención de datos, segmentación de edificios, vegetación agua e incluso coches, y, visualización interactiva de los datos. Usar este proyecto se hubiera basado en optimizar sus escasos hiperparámetros, lo que resultaba incompatible con los objetivos básicos de aprendizaje de la asignatura. No obstante, estudiar en profundidad esta herramienta y el código que usaba por detrás facilitó algunas tareas como el preprocesado de los datos o la visualización de los resultados.
- **Creación de un modelo de segmentación:** Finalmente se decidió hacer fine tuning del modelo de segmentación preentrenado de SAM (Segment Anything Model). El proceso fue más largo del previsto debido a falta de capacidad de computación. Para comprobar los resultados se ha utilizado la métrica del coeficiente de “dice” o Dice coefficient en inglés. Los resultados obtenidos son, de momento, inconsistentes.
- **Visualización de los resultados de segmentación:** Pese a que los resultados no son concluyentes, se han utilizado los datos de entrada para hacer las visualizaciones utilizando la librería de leafmap. Debido al retraso ocasionado por el entrenamiento del modelo las representaciones son funcionales, pero aún, no son visualmente atractivas.
- **Replicar Prefect y MLFlow en todos los ordenadores:** Replicamos la infraestructura elegida en el resto de los ordenadores y modelos que vamos a utilizar. Nos encontramos con algunos obstáculos con respecto al límite de usuarios y tipo de deployments que se pueden tener sin pagar. Finalmente, conseguimos encontrar una opción donde todos podemos estar conectados en el mismo proyecto sin tener que pagar y donde podemos aplicar MLFlow a todos nuestros modelos. Aun así, nuestra estructura tiene ciertas limitaciones que no podemos cambiar, pero sí trabajar con los programas a pesar de ellas.
- **Revisar los datos de Copérnico:** Analizando el aspecto de la serie temporal obtenida, vimos que las anomalías se debían a los datos obtenidos de días donde había muchas nubes en la zona, lo que alteraba el valor del índice de ese día.

Después de investigar cómo funcionaba el parámetro max_cloud_cover, llegamos a la conclusión de que tenemos que usarlo con valor 1 para que nos incluya la menor cantidad de nubes posible. El problema de hacer esto, es que disminuye la cantidad de datos obtenidos en ese periodo de tiempo porque elimina aquellos que no sirven. Por ello, decidimos utilizar como zona a analizar el Parque Natural Sierra Alhamilla en Almería ya que es un lugar con vegetación y donde hay pocas nubes en general. Finalmente, hemos sacado 142 datos entre las fechas 2018-05-01 y 2022-05-30.

- **Refinar el modelo actual:** Una vez hemos obtenido buenos datos, hemos ejecutado el modelo con las características que habíamos obtenido originalmente. Observando la gráfica resultante, vimos que el aspecto de los datos era mucho más coherente sin tantos picos como antes, por lo que era mucho más fácil para el modelo realizar una buena predicción.
  Hemos adaptado los diferentes hiper parámetros del modelo para obtener un mejor resultado y sobre todo, hemos utilizado métricas para poder interpretar de la mejor manera posible los resultados obtenidos. Nos hemos decantado por utilizar un MAE, MAPE y DTW para analizar los resultados del modelo.
- **Dar un enfoque al modelo TCN:** Una vez hemos concluido cómo funciona el modelo y qué datos cogemos para entrenarlo, tenemos que pensar el enfoque que le vamos a dar. Nuestra gran duda era si utilizar metadatos o si analizar la estacionalidad. Finalmente, vamos a entrenar el modelo de manera que no sirva para predecir cómo va a evolucionar ese índice en zonas de climas parecidos a las que vamos a utilizar para entrenar. Este punto aún se encuentra en desarrollo porque tenemos que averiguar cómo implementarlo exactamente.
- **Investigar fuentes de datos de nubes de puntos:** Esta tarea ha sido la que más problemas ha generado ya que nos ha sido imposible conseguir nubes de puntos del gobierno de España. Por lo que tuvimos que buscar alguna alternativa al menos temporalmente para que no bloqueara el resto de las tareas.
- **Obtener datos de nubes de puntos:** Al final conseguimos un dataset que tiene miles de nubes de puntos de edificios y que en su nombre viene una categoria a la que pertenecen por lo que puede ser usado como label para entrenar nuestro modelo.
- **Investigar modelos y arquitecturas de redes:** Comparamos las principales arquitecturas de redes neuronales que esten especializadas en datos 3D y elegimos Pointnet porque ya habia sido usada antes, ya que cumplia con los requisitos para este proposito.
- **Visualizar nubes de puntos:** Mediante la librería open3d podemos hacer visualización de las nubes de puntos conseguidas y podemos ver que tienen gran resolución, pero por limitaciones de hardware tendremos que reducirlas mucho.
- **Implementar modelo de clasificación de edificios por nube de puntos:** Usando la arquitectura de PointNet y el dataset mencionado anteriormente hemos conseguido llegar a tener un modelo usable para clasificar edificios por categorias mediante nubes de puntos. Esta es la primera versión de nuestro modelo y aún tiene poco accuracy.