Prof David Miguel Avila Cruz
22/04/2025
Mario Solano

Tarea Semana 1-2: Pipeline de MLOps + Imagen de Docker

Descripción de un Pipeline de MLOps

Dados los avances tecnológicos, en el campo de la medicina la cantidad de información que existe de los pacientes es muy abundante. Sin embargo, para algunas enfermedades no tan comunes, llamadas huérfanas, los datos que existen escasean. Se requiere construir un modelo que sea capaz de predecir, dados los datos de síntomas de un paciente, si es posible o no que este sufra de alguna enfermedad. Esto se requiere tanto para enfermedades comunes (muchos datos) como para enfermedades huérfanas (pocos datos).

Diseño del Pipeline de Machine Learning

El objetivo es generar un modelo que pueda predecir la probabilidad de múltiples enfermedades, comunes y huérfanas, dado un conjunto de síntomas. Debe de ser capaz de generar una respuesta inclusive en condiciones donde no se tenga toda la información.

Data Ingestion

Para la ingesta de los datos debemos de considerar las siguientes restricciones con respecto a la calidad y estructura de los datos:

- Desbalance de datos entre las enfermedades comunes y huérfanas. Esto afectará directamente el entrenamiento y la evaluación del modelo.
- La calidad y heterogeneidad de los datos variarán ya que vamos a considerar diferentes fuentes de datos con formatos y niveles de detalle faltantes así como la existencia de datos faltantes o inconsistencias.
- Al trabajar con información sensible de cada cliente se debe de considerar la privacidad de los mismos utilizando algún tipo de anonimización.

Los datos que esperamos obtener para la detección de enfermedades pueden ser:

- Datos estructurados o semiestructurados que representen los síntomas presentes, ausentes y la severidad de los mismos
- Datos estructurados con respecto a las características de los mismos como edad, sexo, etc.
- Notas médicas con las anotaciones hechas por el doctor encargado.
- Las probabilidades asociadas a cada enfermedad a considerar o la enfermedad diagnosticada.

Las fuentes usadas propuestas para este modelo serán:

- Historias Clínicas
- Registros especializados de enfermedades raras o huérfanas
- Bases de datos de investigación médica

Inicialmente, los archivos fuente se gestionarán en un repositorio centralizado, preferiblemente un bucket de Amazon S3, con el fin de facilitar su procesamiento para la subsiguiente etapa de limpieza. Dentro de este entorno de almacenamiento, se implementarán conjuntos de scripts (en Python, por ejemplo) diseñados para la extracción, transformación y homogeneización de datos provenientes de diversos formatos de archivo. Este proceso culminará con la carga de los datos estructurados en un data warehouse.

Adicionalmente, esta fase inicial incorporará mecanismos robustos de anonimización para garantizar la protección de la privacidad de los pacientes, adhiriéndonos a las regulaciones pertinentes (e.g., HIPAA si aplica). Asimismo, se integrarán procesos de ingeniería de características, que incluirán la conversión de síntomas primarios a variables categóricas mediante técnicas de codificación (e.g., one-hot encoding) o la discretización de variables numéricas continuas, como la temperatura, en intervalos definidos.

Data Wrangling

En la siguiente fase, se abordará la transformación de los datos brutos para su preparación integral de cara al entrenamiento del modelo predictivo. Este proceso se implementará mediante el uso de Jupyter Notebooks y el lenguaje de programación Python, facilitando la exploración y la aplicación de las transformaciones requeridas. Las transformaciones principales serán principales comprenderán:

- **Homogeneización de nomenclatura y codificación de entidades clínicas:** Establecimiento de un vocabulario controlado y la unificación de códigos (e.g., CIE-10) para enfermedades, síntomas y diagnósticos, resolviendo las inconsistencias terminológicas entre diferentes fuentes de datos. Se aplicarán técnicas de mapeo y estandarización léxica.
- **Codificación de variables categóricas:** Conversión de variables cualitativas, como síntomas, nombres de enfermedades o categorías diagnósticas, a representaciones numéricas mediante métodos de codificación. Esto incluirá la aplicación de técnicas como *one-hot encoding*, *label encoding* o *target encoding*, según la naturaleza y cardinalidad de la variable.
- **Generación de *features* mediante discretización:** Creación de nuevas variables categóricas a partir de variables numéricas continuas, como la edad, mediante la definición de intervalos o *bins* para la formación de grupos etarios específicos.
- **Estandarización y normalización de variables numéricas:** Unificación de las unidades de medida para variables cuantitativas (e.g., conversión entre grados Celsius y Fahrenheit, o entre milímetros y pulgadas). Adicionalmente, se aplicarán técnicas de escalado como la normalización (min-max scaling) o la estandarización (z-score) para asegurar la comparabilidad y optimizar el rendimiento de los algoritmos de modelado.

Model Training

Tras la fase de preparación de datos, la siguiente etapa crítica consiste en la realización de un Análisis Exploratorio de Datos (EDA). Este proceso iterativo se llevará a cabo utilizando bibliotecas especializadas de Python, como Pandas para la manipulación y el análisis tabular, y Matplotlib (o Seaborn) para la visualización gráfica. La ejecución se realizará en entornos Jupyter Notebook, permitiendo una exploración interactiva de las propiedades estadísticas y distribucionales del *dataset*.

El EDA facilitará una comprensión profunda de las características de los datos, incluyendo la identificación de patrones, valores atípicos (*outliers*), distribuciones de variables, correlaciones y posibles problemas de calidad de los datos (e.g., valores faltantes). Posteriormente, se procederá a la ingeniería de características (*feature engineering*), que abarcará la creación de nuevas variables a través de agregaciones, la reducción de la dimensionalidad del espacio de *features* mediante técnicas como el Análisis de Componentes Principales (PCA) o la selección de subconjuntos óptimos de características relevantes para el modelado predictivo.

Una vez concluido el proceso de EDA y la optimización del conjunto de *features*, se iniciará el entrenamiento de modelos de clasificación. Dada la naturaleza del problema, se explorarán algoritmos como los árboles de decisión, implementados a través de la biblioteca scikit-learn (sklearn) de Python. Se evaluarán diversas configuraciones de hiperparámetros y se considerarán otros modelos de clasificación robustos, como Random Forests o Gradient Boosting Machines, para comparar su rendimiento y seleccionar el modelo óptimo basado en métricas de evaluación apropiadas (e.g., precisión, *recall*, *F1-score*, AUC).

Model Selection and Evaluation

En la fase de selección de modelos, se implementará un sistema de seguimiento y registro de experimentos, para lo cual se podrán integrar herramientas de *MLOps* como MLflow o TensorBoard. Estas plataformas facilitarán la gestión de los diferentes modelos entrenados, sus configuraciones de hiperparámetros y las métricas de rendimiento asociadas.

Dada la naturaleza del problema, se abordará como una clasificación multi-etiqueta, considerando la potencial coexistencia de múltiples enfermedades en un mismo paciente. Se prestará especial atención al desafío del desbalance de clases, particularmente en la representación de enfermedades raras. Para mitigar este problema, se emplearán técnicas de remuestreo, como el sobremuestreo de la clase minoritaria (e.g., SMOTE), y la ponderación de clases durante el entrenamiento, asignando mayores pesos a las instancias de las enfermedades huérfanas.

El protocolo de evaluación del modelo se basará en una estrategia de *hold-out* con división estratificada del *dataset*. Esta estratificación se realizará considerando la prevalencia de las enfermedades raras, asegurando una representación proporcional de estas clases en los conjuntos de entrenamiento, validación y prueba. El rendimiento del modelo se cuantificará utilizando métricas de clasificación multi-etiqueta como la precisión

(*accuracy*), el *recall* (sensibilidad) y el *F1-score*, con un análisis detallado del desempeño específico en las enfermedades huérfanas. Para garantizar la robustez de las métricas de evaluación, se implementará un esquema de validación cruzada estratificada (*stratified k-fold cross-validation*) durante la etapa de entrenamiento.

Model Deployment

Para la fase de *deployment*, la solución se encapsulará mediante la tecnología de contenedores Docker. El modelo de *machine learning* entrenado será serializado utilizando una biblioteca eficiente para la persistencia de objetos Python, como `pickle`. Aunque en el contexto de este ejercicio se omita su encapsulación como una función explícita dentro del código principal.

Model Usage

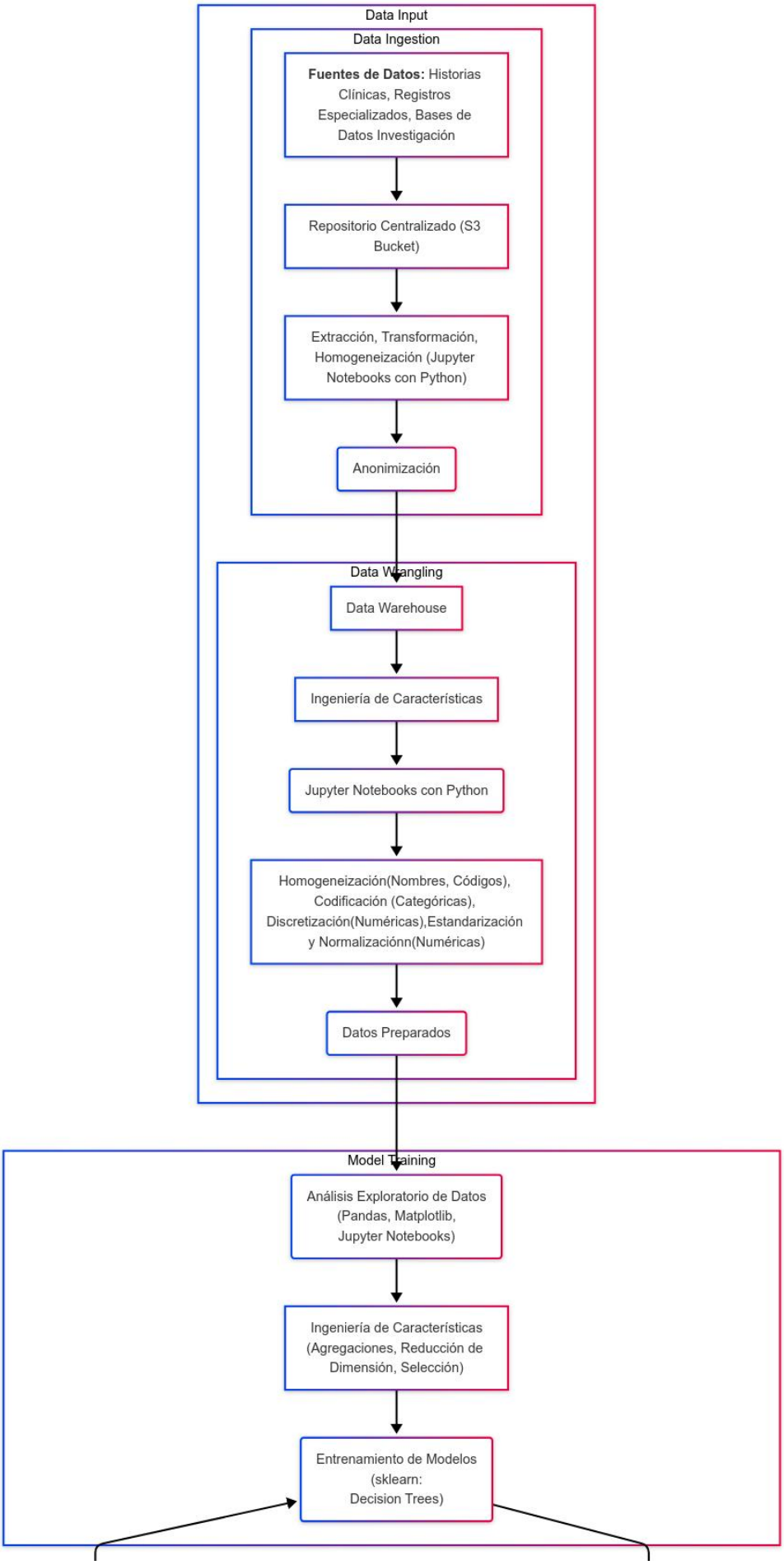
La implementación del modelo predictivo se realizará a través de una interfaz web desarrollada con el *microframework* Flask de Python. Esta interfaz expondrá un formulario que solicitará al menos tres *inputs* relevantes para la predicción. La interacción con el modelo se gestionará mediante peticiones HTTP GET, las cuales invocarán la función de predicción implementada en el *backend* de la aplicación Flask. Al recibir los datos ingresados por el usuario, esta función procesa la información y devolverá la predicción generada por el modelo serializado.

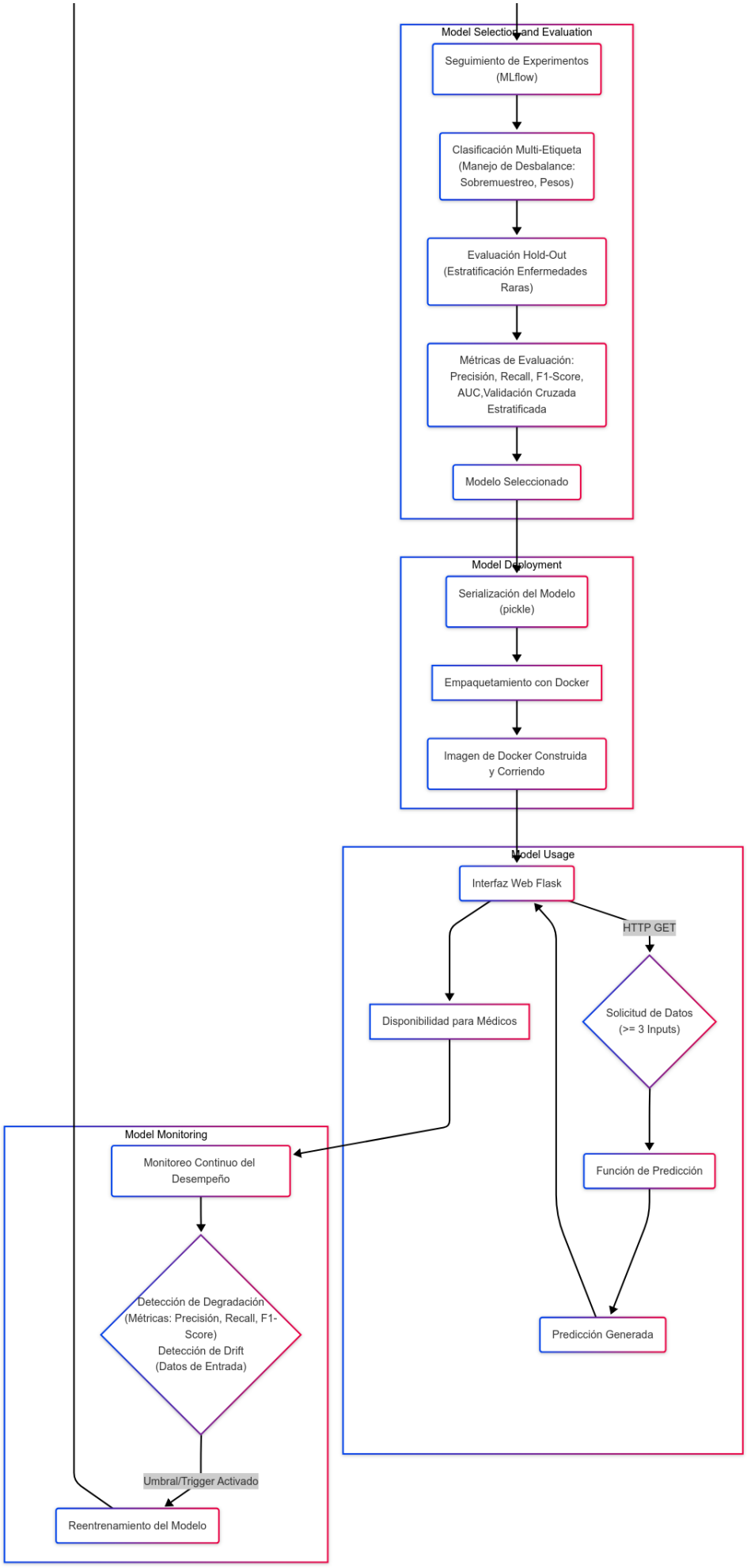
Model Monitoring

Se implementará un sistema de monitoreo continuo del rendimiento del modelo predictivo en producción. Este sistema evaluará métricas clave de rendimiento, como la precisión, el *recall* y el *F1-score*, con el objetivo de detectar cualquier degradación en la capacidad del modelo para identificar correctamente las enfermedades, especialmente ante posibles cambios en la distribución de las variables de entrada o en la manifestación clínica de las patologías.

El sistema de monitoreo incorporará la definición de umbrales o disparadores (*triggers*) basados en la desviación de estas métricas de rendimiento. Al detectarse una disminución significativa en el desempeño del modelo, o cambios sustanciales en las características de los datos de entrada (detectados mediante técnicas de *drift detection*), se activará automáticamente un proceso de reentrenamiento del modelo con los datos más recientes. Este mecanismo de *retraining pipeline* permitirá mantener la precisión y la relevancia del modelo a lo largo del tiempo.

Diagrama del proceso


Data Input
Data Ingestion
Fuentes de Datos: Historias Clínicas, Registros Especializados, Bases de Datos Investigación
Repositorio Centralizado (S3 Bucket)
Extracción, Transformación, Homogeneización (Jupyter Notebooks con Python)
Anonimización
Data Wrangling
Data Warehouse
Ingeniería de Características
Jupyter Notebooks con Python
Homogeneización(Nombres, Códigos), Codificación (Categóricas), Discretización(Numéricas),Estandarización y Normalizaciónn(Numéricas)
Datos Preparados
Model Training
Análisis Exploratorio de Datos (Pandas, Matplotlib, Jupyter Notebooks)
Ingeniería de Características (Agregaciones, Reducción de Dimensión, Selección)
Entrenamiento de Modelos (sklearn: Decision Trees)


Model Selection and Evaluation
Seguimiento de Experimentos (MLflow)
Clasificación Multi-Etiqueta (Manejo de Desbalance: Sobremuestreo, Pesos)
Evaluación Hold-Out (Estratificación Enfermedades Raras)
Métricas de Evaluación: Precisión, Recall, F1-Score, AUC,Validación Cruzada Estratificada
Modelo Seleccionado
Model Deployment
Serialización del Modelo (pickle)
Empaquetamiento con Docker
Imagen de Docker Construida y Corriendo
Model Usage
Interfaz Web Flask
HTTP GET
Disponibilidad para Médicos
Solicitud de Datos (>= 3 Inputs)
Función de Predicción
Predicción Generada
Model Monitoring
Monitoreo Continuo del Desempeño
Detección de Degradación (Métricas: Precisión, Recall, F1-Score)
Detección de Drift (Datos de Entrada)
Umbral/Trigger Activado
Reentrenamiento del Modelo