Etapa de Diseño: Entrada y Preparación de Datos

La etapa de diseño define los fundamentos sobre los cuales se construye la solución, priorizando la calidad de los datos, la seguridad y la preparación adecuada antes de alimentar el modelo de predicción.

Restricciones y características de los datos

Dado que se trata de datos médicos, se deben tener en cuenta las siguientes limitaciones:

- **Datos desbalanceados:** la mayoría de los registros estarán asociados a enfermedades comunes, mientras que las enfermedades huérfanas estarán representadas en cantidades mínimas. Esto puede sesgar el aprendizaje del modelo.

- **Datos sensibles:** los datos personales y clínicos deben ser tratados con confidencialidad, por lo cual se aplicarán estrategias de anonimización y se garantizará la transmisión mediante protocolos seguros como HTTPS.

- **Presencia de datos nulos o mal etiquetados:** es común encontrar síntomas faltantes o inconsistentes en los registros. Esto requiere un proceso de validación riguroso, complementado con imputación estadística o revisión de expertos.

Tipos de datos a considerar

- **Estructurados:** campos como edad, peso y síntomas codificados binariamente (presencia/ausencia).

- **No estructurados (en etapas futuras):** comentarios clínicos o notas médicas en texto libre, que pueden enriquecer el análisis mediante NLP.

Acciones de preprocesamiento y definición de estructura

1. **Validación de datos nulos:** se establecen reglas clínicas para definir si una entrada incompleta puede ser completada (mediante imputación por moda o media) o descartada. La intervención de expertos es clave en casos críticos.

2. **Tratamiento y transformación de datos:**

    - **Codificación:** los síntomas se representan con One-Hot Encoding para mantener independencia semántica.

    - **Imputación:** síntomas ausentes o mal registrados se corrigen con estrategias estadísticas o modelos imputadores.

- **Embeddings (opcional):** si se incorporan características categóricas complejas o textuales, se usarán embeddings para representar semánticamente los conceptos médicos.

3. **Tratamiento del desbalance de clases:**

   - Se aplicará **SMOTE** o **ADASYN** para oversampling en enfermedades poco frecuentes.

   - Si existen datasets externos más completos (por ejemplo, con enfermedades huérfanas), se considerará el uso de **transfer learning** para mejorar la capacidad de generalización del modelo.

4. **Definición de la estructura de datos final:**

   - Se estandariza el esquema de entrada en un formato tabular validado mediante `pydantic` o `jsonschema`, el cual sirve como contrato de entrada tanto para pruebas como para la API de producción.

Infraestructura de almacenamiento

- Los datos brutos se almacenarán en un bucket de **Amazon S3**, aprovechando su escalabilidad y disponibilidad.

- Los datos procesados y transformados se almacenarán en **Amazon Redshift**, lo que permite un análisis SQL enriquecido.

- El versionado de datasets se realizará con **DVC**, permitiendo reproducibilidad exacta del entrenamiento y seguimiento de cambios.

Justificación de tecnologías

- **pandas:** herramienta base para manipulación y análisis exploratorio de datos estructurados.

- **FastAPI + pydantic:** ofrece validación eficiente y esquema de entrada automático para la API.

- **imblearn:** librería robusta para aplicar técnicas de balanceo como SMOTE.

- **DVC:** permite versionar los datos como si fueran código, facilitando auditoría y trazabilidad.

- **S3 + Redshift:** garantizan almacenamiento confiable, escalable y fácilmente integrable con herramientas de análisis y modelado.

Etapa de Desarrollo: Ingesta, Modelado y Evaluación

Esta fase aborda la integración de múltiples fuentes de datos, la construcción iterativa de modelos, y la validación rigurosa del rendimiento del sistema antes de su paso a producción.

3.1 Ingesta de datos y almacenamiento

Los datos utilizados en el sistema provienen de dos fuentes:

- **Fuentes internas:** registros clínicos anonimizados y estructurados provenientes de sistemas hospitalarios o bases médicas institucionales.

- **Fuentes externas:** repositorios abiertos con datos de enfermedades raras (como Orphanet o NIH), fundamentales para enriquecer el entrenamiento de clases poco representadas.

Ambos tipos de datos se almacenan en un entorno de nube (por ejemplo, **Amazon S3**) para facilitar su escalabilidad y disponibilidad. Una vez transformados, se almacenan en bases analíticas como **Amazon Redshift**, aprovechando su integración con S3 y su capacidad de consultas SQL para análisis exploratorio y validación de integridad.

Para mantener la trazabilidad, se utiliza **DVC (Data Version Control)**, lo cual permite versionar datasets como si fueran parte del código, asegurando reproducibilidad en diferentes iteraciones del modelo.

3.2 Análisis exploratorio y Feature Engineering

Con los datos integrados y almacenados, se procede a realizar un análisis exploratorio utilizando **pandas** y **seaborn**. Este análisis permite detectar:

- Valores extremos y atípicos.

- Ausencia o inconsistencias en campos clínicos.

- Correlaciones entre variables (por ejemplo, síntomas y gravedad).

Posteriormente, se realiza el **feature engineering**, que incluye:

- Transformación de variables.

- Interacciones entre síntomas.

- Técnicas de reducción de dimensionalidad (como PCA si es necesario).

Este proceso se desarrolla de forma interactiva en **Jupyter Notebooks**, lo que permite iterar rápidamente sobre diferentes versiones del conjunto de datos y visualizar métricas relevantes en tiempo real.

3.3 Entrenamiento del modelo

Dado que el problema es una clasificación multiclase, se opta inicialmente por modelos de árboles de decisión, como:

- **Random Forest**, por su robustez ante ruido y capacidad para manejar variables categóricas sin requerir transformación previa.
- **XGBoost**, por su eficiencia computacional, regularización integrada y excelente rendimiento en datasets tabulares.

El entrenamiento se realiza con diferentes subconjuntos de datos y configuraciones de hiperparámetros. Se utiliza **k-fold cross-validation** (usualmente con k=5) para asegurar que el modelo generalice bien y no esté sobreajustado.

Se emplean herramientas de tuning como `GridSearchCV` o `Optuna` para encontrar la mejor combinación de hiperparámetros y mejorar la métrica objetivo.

3.4 Evaluación y comparación

Los modelos son evaluados usando:

- **ROC AUC** (área bajo la curva) para verificar discriminación entre clases.
- **F1-score** para equilibrio entre precisión y recall en clases desbalanceadas.
- **Matriz de confusión** para evaluar errores específicos por clase.
- **Tiempo de inferencia** y uso de memoria, para garantizar viabilidad en ejecución local.

El modelo con mejor equilibrio entre rendimiento y eficiencia se selecciona como candidato a producción. Sus resultados se revisan manualmente por expertos médicos para validar que las predicciones sean clínicamente coherentes.

3.5 Visualización y documentación

Los resultados se documentan y visualizan en **Streamlit** o **matplotlib**, generando reportes interactivos con métricas, curvas ROC y ejemplos de predicciones. Estas herramientas permiten que los stakeholders (como médicos o equipos de validación) puedan evaluar el modelo desde una interfaz amigable, sin requerir conocimientos técnicos.

Opcionalmente, herramientas como **MLflow** o **Tableau** podrían integrarse si se requiere seguimiento continuo de experimentos o colaboración multidisciplinaria.

3.5 Validación técnica y ciclo iterativo

Antes de dar por finalizada esta etapa, el modelo pasa por una **revisión técnica** adicional que implica la validación por parte de expertos clínicos (validación funcional) y validación automática vía pruebas unitarias.

- Si el modelo **no cumple** con los criterios definidos (desempeño inferior, errores sistemáticos, inconsistencia clínica), se **devuelve al ciclo de entrenamiento** para una nueva iteración. Esto puede implicar ajustar hiperparámetros, redefinir variables o incorporar nuevos datos.

- Si el modelo **cumple con las expectativas**, se **marca como listo para su paso a producción**.

Este ciclo asegura robustez, confiabilidad y adecuación clínica del modelo antes de su implementación definitiva.

4. Etapa de Producción

Una vez el modelo ha sido validado técnica y estadísticamente, se procede a su paso a producción. Esta etapa incluye el despliegue del modelo, su monitoreo continuo, la gestión de las predicciones generadas diariamente y la lógica de reentrenamiento con base en el desempeño observado.

4.1 Despliegue del modelo

El modelo se empaqueta dentro de un contenedor Docker y se expone mediante una API REST desarrollada con **FastAPI**, por su alto rendimiento, facilidad de documentación automática (con Swagger/OpenAPI) y su asincronía nativa basada en `async/await`, lo cual permite mejorar la escalabilidad. Esta API contiene los endpoints `/predict` y `/reporte`.

El entorno de ejecución puede funcionar localmente (usando `Uvicorn` como servidor ASGI) o en la nube mediante servicios como **Render**, **AWS EC2**, o **Railway**, dependiendo de los recursos disponibles y la estrategia de operación (uso local por médicos o acceso remoto a través de internet). Para ambientes con múltiples instancias o que requieran alta disponibilidad, se recomienda orquestación con **Docker Compose** o **Kubernetes**.

4.2 Generación de predicciones

Cada vez que un usuario envía datos a través del formulario o API, el modelo realiza una predicción y la almacena en un archivo persistente (`data/predicciones.json` o `.txt`). Estos datos permiten auditar el comportamiento del modelo y sirven como base para su evaluación posterior.

Adicionalmente, se habilita un endpoint `/reporte` que permite acceder a:

- Conteo de predicciones por categoría.
- Las últimas 5 predicciones realizadas.
- Fecha de la predicción más reciente.

Este reporte puede ser útil para los profesionales médicos como una herramienta de trazabilidad.

4.3 Monitoreo del sistema

Se implementa monitoreo en dos niveles:

- **Infraestructura**: se utilizan herramientas como **Prometheus** y **Grafana** para registrar y visualizar métricas como tiempos de respuesta de la API, disponibilidad del servicio y uso de recursos (CPU, RAM).
- **Modelo**: diariamente se evalúan métricas clave del modelo como `F1-score`, `precision`, `recall` y `ROC AUC`, comparándolas con los umbrales definidos durante el desarrollo. Si alguna métrica cae por debajo del umbral, se activa una alerta que desencadena el proceso de reentrenamiento.

La evaluación también revisa la distribución de los datos de entrada y si han cambiado significativamente respecto a los datos con los que se entrenó el modelo originalmente (data drift).

4.4 Reentrenamiento automático

El sistema permite la ingesta continua de nuevos datos (por ejemplo, síntomas y diagnósticos confirmados posteriormente) que se almacenan en un bucket S3. Un proceso orquestado por **Apache Airflow** puede revisar periódicamente los datos y, si las condiciones lo ameritan (por ejemplo, degradación de métricas), dispara un reentrenamiento automático.

Este reentrenamiento sigue una lógica similar al entrenamiento original:

- Preprocesamiento de nuevos datos.
- Validación y entrenamiento del modelo.
- Evaluación comparativa con el modelo anterior.
- Promoción del nuevo modelo a producción si supera los umbrales definidos.

Esto cierra el ciclo de mejora continua del sistema y garantiza que el modelo permanezca actualizado frente a la evolución de los síntomas o enfermedades.

4.5 Tecnologías empleadas en esta etapa

- **FastAPI**: API rápida y eficiente, ideal para producción.
- **Docker**: empaquetado del modelo y sus dependencias.
- **Docker Compose / Kubernetes**: orquestación del entorno.
- **Prometheus + Grafana**: monitoreo técnico y funcional.
- **Apache Airflow**: orquestación del reentrenamiento automatizado.
- **AWS S3 / Redshift**: almacenamiento y recuperación eficiente de datos.

5. CHANGELOG

Cambio	Propuesta Original	Propuesta Actual
Almacenamiento de datos	No especificado	S3 para datasets, Redshift para estructurados
Tecnologías por etapa	Mencionadas superficialmente	Justificadas y detalladas
Métricas de validación	No definidas	ROC AUC, F1-score, matriz de confusión
Escenarios de ejecución	Solo nube	Nube o local según recursos del médico
Monitoreo del modelo	No presente	Monitoreo activo con Prometheus + reentrenamiento
Validación de calidad	Implícita	Umbrales definidos + pruebas unitarias

Iteración de modelos	No contemplada	Flujo iterativo documentado
Ingesta de nuevos datos	Manual o ausente	Automatizada con Airflow