🚀 Roadmap de proyecto de Meta-Learning

1️⃣ Preparación y recopilación de datos

Objetivo: Tener todos los datasets, resultados previos y meta-features listos para construir tu meta-dataset.

Issue	Descripción	Prioridad	Notas
`data-collection`	Descargar datasets de OpenML relevantes para clasificación y regresión.	Alta	Usar OpenML100 como punto de partida. Guardar IDs de datasets y tareas.
`pipelines-historicos`	Recopilar pipelines existentes de PIPES (o AutoML: TPOT, AlphaD3M) para cada dataset.	Alta	Guardar como JSON o CSV, con hiperparámetros y resultados.
`meta-feature-extraction`	Extraer meta-features de cada dataset (num. instancias, num. atributos, stats por columna, etc.).	Alta	Implementar funciones reutilizables para meta-learning.
`meta-dataset-construction`	Construir meta-dataset combinando: meta-features, algoritmo, hiperparámetros, métricas de performance.	Alta	Formato: cada fila = configuración + dataset + resultado.

2️⃣ Análisis exploratorio

Objetivo: Conocer la distribución de algoritmos, hiperparámetros y métricas.

Issue	Descripción	Prioridad	Notas
`eda-algorithm-performance`	Analizar qué algoritmos funcionan mejor según tipo de dataset.	Media	Graficar rankings por precisión, tiempo y otras métricas.
`eda-hyperparameter-impact`	Evaluar tunabilidad de hiperparámetros.	Media	Inspirarse en Probst et al. (Tunability).
`eda-meta-feature-correlation`	Estudiar correlación de meta-features con performance de algoritmos.	Media	Útil para seleccionar meta-features más predictivas.

3️⃣ Modelado de meta-learning

Objetivo: Entrenar modelos que recomienden algoritmos/pipelines para nuevas tareas.

Issue	Descripción	Prioridad	Notas
`meta-learner-algorithm`	Implementar modelo que prediga ranking de algoritmos basado en meta-features.	Alta	Puede ser k-NN, Random Forest o LightGBM.
`meta-learner-pipeline`	Extender para predecir pipelines completos (selección de preprocesamiento + algoritmo + hiperparámetros).	Alta	Inspirarse en AlphaD3M y PIPES.
`loss-time-curve`	Implementar evaluación tipo *Loss-Time Curve* para ranking de recomendaciones.	Media	Inspirado en "Fast Algorithm Selection using Learning Curves".
`pairwise-meta-rules`	Opcional: mejorar ranking usando reglas pairwise sobre datasets similares.	Media	Inspirado en *Pairwise meta-rules for better meta-learning*.

4️⃣ Evaluación y validación

Objetivo: Comprobar que el meta-learner genera recomendaciones útiles.

Issue	Descripción	Prioridad	Notas
`cross-validation-meta`	Validar meta-modelo usando *leave-one-dataset-out*.	Alta	Cada dataset nuevo es tratado como una nueva tarea.
`compare-baseline`	Comparar recomendaciones con valores por defecto y ranking aleatorio.	Alta	Medir mejora en performance y tiempo.
`hyperparameter-tuning-effect`	Evaluar cuánto mejora la recomendación ajustando hiperparámetros críticos.	Media	Basarse en análisis de tunabilidad.

5️⃣ Interfaz / Output

Objetivo: Facilitar uso de recomendaciones en nuevos datasets.

Issue	Descripción	Prioridad	Notas
`recommendation-api`	Implementar función que reciba dataset y devuelva ranking de algoritmos/pipelines.	Alta	Input: meta-features; Output: ranking + configuraciones sugeridas.
`visualizations`	Graficar rankings, curvas de Loss-Time, impacto de hiperparámetros.	Media	Útil para análisis y reportes.

Issue	Descripción	Prioridad	Notas
export-results	Guardar recomendaciones y evaluaciones en CSV/JSON para reproducibilidad.	Media	Integrar con pipelines históricos.

6️⃣ Extras / Futuro

Issue	Descripción	Prioridad	Notas
integration-pipes	Integrar PIPES meta-dataset completo como referencia.	Baja	Para enriquecer histórico de pipelines.
reinforcement-learning-pipeline	Explorar MCTS o RL para optimizar pipelines automáticamente.	Baja	Inspirado en AlphaD3M.
hyperparameter-prioritization	Usar tunabilidad para priorizar tuning en pipelines recomendados.	Media	Reduce tiempo de computación.

```
flowchart TD
    A[📥 Datasets OpenML / PIPES] --> B[🔹 Extracción de Meta-features]
    B --> C[📊 Construcción de Meta-dataset]
    C --> D[🔍 Análisis Exploratorio]
    D --> E[🤖 Modelado de Meta-Learner]
    E --> F[📈 Evaluación y Validación]
    F --> G[💡 Recomendaciones de Algoritmos / Pipelines]
    G --> H[📊 Visualización y Exportación de Resultados]

    subgraph "Preparación de datos"
        A --> B
        B --> C
    end

    subgraph "Análisis y Modelado"
        C --> D
        D --> E
    end

    subgraph "Producción y Resultados"
        E --> F
        F --> G
        G --> H
    end

    %% Notas adicionales
    B -.-> B1[Incluye stats, número de instancias, clases, correlaciones,
etc.]
    E -.-> E1[Algoritmos: k-NN, Random Forest, LightGBM, RL para pipelines]
    F -.-> F1[Cross-validation, comparación con baseline y valores por
```

```
defecto]
    G -.-> G1[Ranking, hiperparámetros sugeridos, configuración completa
del pipeline]
```