Tutorial para colaborar en la app Model4ChemAI

Índice

1. Introducción y objetivos

Esta aplicación tiene como objetivo principal facilitar el análisis de moléculas y sus actividades biológicas mediante modelos de IA.

La app está organizada en varias pestañas que permiten:

- **Carga de datos:** Los usuarios pueden subir archivos CSV con moléculas (representadas por SMILES) y etiquetas de actividad (activo/inactivo). Además, existe la opción de obtener datos directamente de bases externas como ChEMBL y PubMed (estas funciones son opcionales y en desarrollo).
- **Análisis de datos:** Visualización de la distribución de datos cargados, incluyendo cantidad total, activos e inactivos, y la distribución de similitud molecular.
- **Visualización de moléculas activas e inactivas:** Con gráficos de propiedades fisicoquímicas y similitudes, separados para cada clase.
- **Resumen y recomendación:** Basado en los datos analizados, la app sugiere qué tipo de modelo de IA es más adecuado para el conjunto de datos (por ejemplo, CNN para grandes volúmenes, Random Forest para datos pequeños, etc.).

Este proyecto se desarrollará colaborativamente, con tareas individuales asignadas a los miembros del equipo.

---

2. Tareas y asignaciones

Para organizar el desarrollo, las siguientes tareas están divididas para que cada integrante pueda enfocarse en un módulo específico y debe generar una sub-rama dentro de la rama de deploy para desarollar la tarea

Tarea 1: Desarrollo de la pestaña de carga de datos (Prioritaria)

- Implementar la funcionalidad para subir un archivo CSV con moléculas y etiquetas.

- Validar el formato del CSV, manejo de errores y estandarización de nombres de columnas para que sean compatibles con el resto de la app.
- Integrar la opción de elegir entre carga manual o descarga de datos desde ChEMBL y PubMed (esta parte es opcional y se desarrollará si hay tiempo).

Tarea 2: Integración con ChEMBL (Secundaria)

- Desarrollar el código para consultar la API de ChEMBL.
- Extraer los datos de bioactividad de un target proteico o bioassay específico.
- Obtener los SMILES y la etiqueta de actividad para incorporarlos a la app.

Tarea 3: Integración con PubMed Assays (Secundaria)

- Implementar la extracción de datos de PubMed assay.
- Obtener los SMILES y etiquetas de actividad.
- Esta tarea es opcional y se llevará a cabo si el tiempo lo permite.

Tarea 4: Desarrollo de la pestaña de análisis y visualización de datos (Prioritaria)

- Crear visualizaciones para mostrar la distribución total de datos, cantidad de activos e inactivos.
- Mostrar la distribución de similitud molecular diferenciando entre activos e inactivos.

Tarea 5: Visualización separada de activos e inactivos (Secundaria)

- Desarrollar subsecciones para activos e inactivos.
- Mostrar gráficos de propiedades fisicoquímicas y similitudes para cada grupo por separado.

Tarea 6: Resumen y recomendaciones de modelos (Prioritaria)

- Implementar la pestaña que sintetice toda la información.
- Basándose en las características del dataset (cantidad, distribución, etc.), generar recomendaciones automáticas de modelos IA.
- Incorporar el archivo con las reglas y criterios para las recomendaciones.

---

3. Requisitos previos

Antes de comenzar a trabajar en la app, asegurate de tener instalado y configurado lo siguiente:

- **Git:** Para controlar versiones y colaborar con el repositorio.
- **Python 3.8+** (recomendado usar Anaconda/Miniconda para manejar ambientes).
- **Conda:** Para crear y manejar entornos aislados.
- **Acceso al repositorio de GitHub:** Envíale al lider del proyecto tu usuario de GitHub y cloná el repositorio donde está el código.
- **Visual studio code** Para desarrollar la app

---

4. Configuración del entorno local

Clonar el repositorio

```
git clone https://github.com/tu_usuario/Model4ChemAI.git
cd Model4ChemAI
```

Crear y activar el entorno Conda

```
conda create -n model4chemai_env python=3.8 -y
conda activate model4chemai_env
```

Instalar las dependencias

```
pip install -r requirements.txt
```

---

5. Estructura del proyecto

A continuación una descripción rápida de las carpetas y archivos más importantes:

```
Model4ChemAI/
├── app.py                  # Archivo principal de la aplicación Streamlit
├── data/                   # Carpeta para archivos de datos
├── modules/                # Código modular (carga, análisis, visualización)
│   ├── data_loader.py      # Funciones para cargar y validar datos
│   ├── chembl_api.py       # Código para consulta a ChEMBL
│   └── pubmed_api.py       # Código para extracción de datos PubMed
├── requirements.txt        # Lista de dependencias Python
├── README.md               # Documentación principal del proyecto
└── ...
```

---

6. Cómo trabajar con ramas

Para mantener un desarrollo ordenado y evitar romper la app estable, seguimos esta estrategia:

- `main`: Rama para la versión estable y deploy.
- `develop`: Rama para integración de nuevas funcionalidades en desarrollo.
- Ramas de feature: ramas específicas para cada tarea o mejora, creadas a partir de `develop`.

Crear una rama para una feature o bugfix

```
git checkout develop
git pull origin develop
git checkout -b feature/nombre-de-tu-feature
```

Hacer commit y push

```
git add .
git commit -m "Descripción clara de los cambios"
git push origin feature/nombre-de-tu-feature
```

---

7. Editar y probar la app localmente

Ejecutar la app localmente

Desde la carpeta raíz del proyecto, con el entorno activado:

```
streamlit run app.py
```

Esto abrirá la app en tu navegador local para que puedas probar los cambios.

---

8. Hacer un Pull Request (PR)

Para integrar tus cambios a la rama `develop`:

1. Subí tu rama feature al repositorio remoto (como en el paso anterior).
2. En GitHub, crea un Pull Request desde tu rama feature hacia `develop`.
3. Avisá al lider del proyecto para esperar revisiones y aprobaciones.
4. Una vez aprobado, el lider del proyecto hará el merge a `develop`.

---

9. Deploy en Streamlit Cloud

- El deploy es automático cuando se hacen push a la rama configurada (en este caso a `main`).
- En este caso, estaremos trabajando en la rama de develop. Haremos el merge a mail cuando las tareas estén completadas.

---

10. Buenas prácticas

- Usar mensajes claros y descriptivos en los commits.
- Comentar el código para facilitar la colaboración.
- Mantener el entorno local actualizado con los últimos cambios del repo.
- Probar la app localmente antes de subir cambios.

---

11. Preguntas frecuentes y soporte

Problemas comunes

- **Error al instalar dependencias:** Verificá la versión de Python y Conda.
- **App no corre localmente:** Revisá que tengas las últimas dependencias y que el entorno esté activado.
- **Conflictos en Git:** Consultá al equipo o revisá documentación Git para resolver merges.

Dónde pedir ayuda

- Preguntá en el canal de Discord del equipo.
- Contactá al responsable del proyecto.

Material de consulta

- Tutorial de Streamlit
- Uso de RDKit
- Tutorial de GitHub
- Tutorial de Python
- Guía de uso con ejemplos de acceso a PubChem y ChEMBL
- Guía de procesamientos de datos químicos
- Guía de uso de Caracterización y de moléculas
- Guía de cálculos de similitud de moléculas

¡Gracias por colaborar en Model4ChemAI! Con este tutorial queremos que el desarrollo sea fluido, ordenado y colaborativo.

Información adicional

La aplicación ya está desplegada y funcionando en línea en:
https://model4chemai.streamlit.app/

Podés visitarla para ver la última versión estable y probar su funcionalidad.
