Reglas de Juego del Curso "DESARROLLO DE PROYECTOS DE INTELIGENCIA ARTIFICIAL"

CANALES DE COMUNICACIÓN

Para facilitar la comunicación efectiva y constante a lo largo del curso, utilizaremos los siguientes canales:

- **WhatsApp:** Para comunicaciones rápidas e informales.
- Grupo de WhatsApp: Medio principal para anuncios, envío de material adicional como artículos, videos, noticias y ejemplos de código relacionados con el curso [Enlace del grupo].
- **Correo Institucional:** Se utilizará para comunicaciones formales o envíos que requieran un documento adjunto de mayor tamaño o confidencialidad. Japolanco@uao.edu.co
- **Plataforma UAO Virtual:** Lugar de acceso a recursos del curso, material complementario y espacios para entregar tareas.
- **Foro de dudas o inquietudes:** Espacio donde pueden exponer sus dudas sobre el contenido del curso, actividades o aspectos técnicos.

CRONOGRAMA DE CLASES

Duración del Curso

El curso se desarrollará en aproximadamente 3 meses, iniciando el 09 de Agosto y finalizando el 2 de Octubre.

CONTENIDO DEL CURSO

El curso se divide en cuatro módulos principales:

Módulo 1: Introducción al desarrollo de software (25%)

En este módulo se abordan las distintas metodologías usadas para el desarrollo de software y cómo estas influyen en el desarrollo de un proyecto ML-IA. Durante esta

actividad de aprendizaje podrás comprender el flujo de trabajo necesario para desarrollar un proyecto de ML-IA y las tecnologías que apoyan estos desarrollos a través de la definición de una metodología que responda al contexto del problema y de respuesta a este. El proyecto es planteado por la Empresa Talos IA.

Módulo 2: Desarrollo de Software: Etapas y Metodologías (35%)

Este módulo se centrará en elegir las tecnologías para el despliegue, integración y funcionamientos de módulos más convenientes para el desarrollo de un proyecto ML-IA.

Módulo 3: Proyectos de IA-ML (25%)

Este módulo aborda la parte técnica necesaria para desarrollar software, abordando conceptos de diseño y tecnologías de integración entre distintos módulos de un mismo sistema, por ejemplo, GRPC. El objetivo es entonces diseñar e implementar un software que tenga componentes de ML-IA.

Módulo 4: Operación de Proyectos de IA-ML (15%)

En este módulo se exploran herramientas para monitorear el entrenamiento de modelos, así como herramientas para estudiar el desempeño del modelo en producción. Al final, lograremos implementar un Pipeline que monitoree el entrenamiento de un modelo de ML-IA.

Herramientas Adicionales

Se utilizarán recursos de prototipado y herramientas de interfaz como Streamlit, Gradio, tkinter o Mesop para la creación de aplicaciones prácticas y de demostración de los proyectos.

Plataforma de Trabajo

El curso se realizará preferiblemente en Local Usando Visual Studio Code. Usando todas las herramientas de *pair programing* disponibles (GPT 5, Claude Opus 4.1, Deep Seek R1, Gemini 2.5 PRO, etc)

ACTIVIDADES DEL CURSO

Módulo 1: Proyecto Neumonía

- Formato Grupal (Máximo 4 personas no se aceptan envíos individuales)
- Se tiene que clonar la información del repositorio de Neumonía disponible [siguiente enlace]
- El archivo conv_MLP_84.h5 es el que usará para cargar el modelo.
- Se tiene que hacer un patrón de diseño de software
- El sistema tiene que estar bien desacoplado
- El sistema tiene que tener buena cohesión
- Dentro de los entregables debe estar el Docker.
- En el archivo README.md se encuentra la información a realizar, de todas formas, se lista acá las actividades a realizar.
- Se tiene que clonar un repositorio de GitHub con la información de Neumonía donde van a mejorar el proyecto.
- Tienen que crear un virtual para ejecutar el código.
- Se tiene que agregar las versiones a las librerías utilizadas en el archivo requirements.txt
- Se tiene que mejorar el README a nivel de contenido y de información, agregar imágenes y demás información relevante. Agregar el tipo de versión de Python que está utilizando actualmente. Agregar información para poder ejecutar el proyecto utilizando bloques código ya sea de bash, Python, Docker, etc. Agregar un posible tipo de licencia [ejemplo aquí]
- El modelo .h5 se tiene que agregar al .gitignore.
- Si ya tienen la imagen de Docker pero no les funciona pueden validar el proceso agregando la variable de entorno DISPLAY, en este link aparece como es la configuración [Enlace disponible]
- **Implementar integrator.py:** Es un módulo que integra los demás scripts y retorna solamente lo necesario para ser visualizado en la interfaz gráfica.

Retorna la clase, la probabilidad y una imagen el mapa de calor generado por Grad-CAM.

- **Implementar read_img.py:** Script que lee la imagen en formato DICOM para visualizarla en la interfaz gráfica. Además, la convierte a arreglo para su preprocesamiento.
- **Implementar preprocess_img.py:** Script que recibe el arreglo proveniente de read_img.py, realiza las siguientes modificaciones: resize a 512x512. Conversión a escala de grises. ecualización del histograma con CLAHEnormalización de la imagen entre 0 y 1. conversión del arreglo de imagen a formato de batch (tensor).
- **Implementar load_model.py:** Script que lee el archivo binario del modelo de red neuronal convolucional previamente entrenado llamado 'WilhemNet86.h5'.
- **Implementar grad_cam.py:** Script que recibe la imagen y la procesa, carga el modelo, obtiene la predicción y la capa convolucional de interés para obtener las características relevantes de la imagen.
- **Generación de pdfs:** Para la generación pueden instalar tkcap, tienen que buscar la información y la documentación para instalar la librería.
- **Código en formato PEP8 (Python Enhancement Proposal 8)** es una guía de estilo para el lenguaje de programación Python. Su objetivo es proporcionar convenciones claras y consistentes sobre cómo escribir código Python legible y fácil de mantener. PEP 8 es ampliamente aceptado por la comunidad Python, y seguir sus directrices ayuda a mejorar la claridad y la coherencia en el código.
    - Nombres de variables, funciones y clases: Usar minúsculas con palabras separadas por guiones bajos (ej. nombre_variable). Para las clases usar la notación de mayúsculas y minúsculas (CamelCase) (ej. NombreClase).
    - Indentación
    - Líneas de código: Limitar el largo de las líneas de código a 79 caracteres (72 para comentarios y docstrings).
    - Espacios en blanco: Evitar el uso innecesario de espacios, especialmente alrededor de operadores y en las declaraciones de función (ej. x = 10 en lugar de x = 10).
    - Comentarios: Los comentarios deben ser claros y útiles, y debe usarse un estilo consistente. Es recomendable usar comentarios en línea cuando sea necesario para explicar código complejo.
    - Todas las funciones **tienen que tene**r Docstrings: Las funciones, clases y módulos deben incluir docstrings (comentarios que documentan su propósito) siguiendo un formato específico. Los docstrings deben estar entre comillas triples (""") y proporcionar una descripción clara de lo que hace el componente.

- Importaciones: Las importaciones deben ir al inicio del archivo y estar agrupadas de la siguiente forma:
  - 1. Importaciones estándar de Python.
  - 2. Importaciones de bibliotecas de terceros.
  - 3. Importaciones locales.
  - - Separar cada grupo con una línea en blanco.
- Se tiene que refactorizar todo el código que ya este obsoleto (deprecated) y con una ejecución sin warnings.
- Agregar 2 pruebas unitarias como mínimo. Las funciones por probar son de libre elección para el estudiante. Para ello el estudiante debe usar pytest.

- Vamos a considerar el siguiente árbol para los proyectos.

📁 data-science-project/

├── 📁 data/

│ ├── 📁 raw/ (Unprocessed datasets)

│ ├── 📁 processed/ (Cleaned and preprocessed datasets)

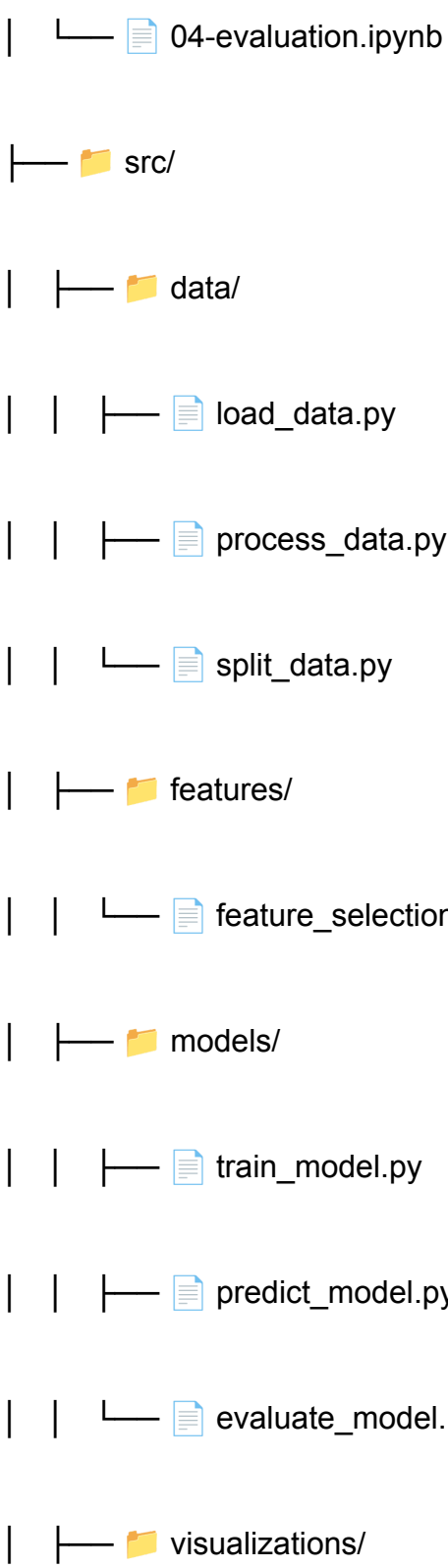
│ └── 📁 external/ (Datasets from third-party sources)

├── 📁 notebooks/

│ ├── 📄 01-data-exploration.ipynb

│ ├── 📄 02-feature-engineering.ipynb

│ ├── 📄 03-model-training.ipynb


04-evaluation.ipynb
src/
data/
load_data.py
process_data.py
split_data.py
features/
feature_selection.py
models/
train_model.py
predict_model.py
evaluate_model.py
visualizations/

```
│   │   └── 📄 plot_results.py
├── 📁 tests/
│   ├── 📄 test_data_processing.py
│   ├── 📄 test_models.py
│   └── 📄 test_visualizations.py
├── 📁 reports/
│   ├── 📁 figures/ (Plots and charts)
│   └── 📄 report.md
├── 📁 docs/
│   └── 📄 README.md
├── 📄 requirements.txt (Python dependencies)
├── 📄 .gitignore
└── 📄 LICENSE
```

Explicación de las carpetas (Directorios)

Una estructura organizada de carpetas es clave para mantener la claridad y escalabilidad en un proyecto de Data Science. A continuación, te presento una guía sobre cómo puedes organizar tu proyecto utilizando directorios específicos:

1. data/
   - **Descripción:** Almacena los datasets en diferentes etapas del proceso de análisis.
   - **Subcarpetas:**
     a. **raw/:** Datos en su forma original, sin procesar.
     b. **processed/:** Datos que ya han sido limpiados y transformados para ser utilizados en el modelo.
     c. **external/:** Datos de fuentes externas que pueden ser usados en el análisis.
2. notebooks/
   - **Descripción:** Carpeta destinada a los notebooks de Jupyter, donde se lleva a cabo la experimentación paso a paso.
   - **Ejemplos de uso:**
     a. Exploración de datos.
     b. Ingeniería de características.
     c. Análisis exploratorio (EDA).
     d. Visualización y pruebas iniciales.
3. src/
   - **Descripción:** Contiene los scripts de Python para las operaciones modulares del proyecto.
   - **Subcarpetas:**
     a. **data/:** Scripts para cargar, limpiar y dividir los datos.
     b. **features/:** Scripts para la ingeniería de características y selección de variables.
     c. **models/:** Scripts para el entrenamiento, predicción y evaluación de modelos.
     d. **visualizations/:** Scripts para generar gráficos y visualizaciones que ayudan a interpretar los resultados.
4. tests/
   - **Descripción:** Contiene las pruebas unitarias para asegurar que los scripts y pipelines funcionen correctamente.
   - **Objetivo:** Validar que todas las funciones y módulos están operando como se espera.
5. reports/

- **Descripción:** Carpeta destinada a los informes finales, figuras y visualizaciones.
- **Contenido:**
    a. Documentos finales del proyecto.
    b. Gráficos y figuras generadas durante el análisis.

6. docs/
    - **Descripción:** Documentación del proyecto y el archivo README.
    - **Contenido:**
        a. Descripción del proyecto, objetivos y metodología.
        b. Guía sobre cómo ejecutar y reproducir el proyecto.
        c. Requisitos y dependencias.

Módulo 2: Esquematización del proyecto

- Mapa mental sobre el desarrollo de software y la metodología ágil.
    - Escoge las ideas centrales de las lecturas sobre los retos del desarrollo de software
    - Escoge las ideas sobre el ciclo de vida de un proyecto de ML-IA
    - Escoge las ideas sobre metodologías ágiles
    - Escoge las ideas sobre tecnologías para el desarrollo ágil.
    - Consulta información adicional para enriquecer tu mapa mental.
    - Agrega ramificaciones que unen las ideas de manera coherente.
    - Agregar imágenes, colores o figuras, que sean de utilidad para complementar las ideas.
    - Se puede utilizar Canva, Miro, Draw.io o el que consideren sea el más adecuado.
    - La actividad se entrega en el grupo planteado al inicio del curso.
    - Grupo máximo de 4 personas, indicar en el mapa mental el nombre y código de los estudiantes.
- Mapa mental sobre la relación del pipeline con las etapas de despliegue del proyecto.
    - Escoge las ideas centrales de las lecturas sobre Integración continua
    - Escoge las ideas centrales de las lecturas sobre despliegue continuo
    - Escoge las ideas centrales de las lecturas sobre concepto de pipeline
    - Escoge las ideas centrales de las lecturas sobre la importancia de un pipeline y su relación con la detección de bugs.
    - Consulta información adicional para enriquecer tu mapa mental.
    - Agrega ramificaciones que unan las ideas de manera coherente.
    - Agrega imágenes, colores o figuras, que sean de utilidad para complementar las ideas.
    - Se puede utilizar Canva, Miro, Draw.io o el que consideren sea el más adecuado.
    - La actividad se entrega en el grupo planteado al inicio del curso.
    - Grupo máximo de 4 personas, indicar en el mapa mental el nombre y código de los estudiantes.
- Entrega propuesta del proyecto de curso
    - Tabla de contenido
    - Contexto del proyecto, link donde esta el modelo, que hace el modelo, quien lo creo, cual es su model card, restricciones de software para su ejecución y demás información relevante que ofrezca más contexto.
    - Objetivos y Alcance
    - Cronogramas y tareas. Se puede hacer en Microsoft Proyect, Hacer cálculo de ruta crítica, Hitos, milestones, Fases y tareas.

    - Se tiene que montar un Tablero de Kanban con las tareas que se van a realizar para el módulo 3, se recomienda utilizar github projects [para más información].
    - Con todos los tickets se tiene la información para hacer el tablero de Kanban, tienen que agregarse todos en el grupo y me pueden agregar a mi para dado el caso estar revisando los tableros o el tablero agregarlo de forma publica y compartir el link
    - Identificar El marco de investigación, cuales modelos se van a considerar (*.pkl, *.h5 o *.keras, etc)
    - Identificar cuales datasets se van a usar.
    - Cuales técnicas se van a usar (ML o DL o CV)
    - Cuales frameworks se van a utilizar (TensorFlow, Keras, Flask, Docker, Streamlit, Pycaret, etc)
    - Cuales librerías en general se van a usar.
    - La información planteada tiene que ser detallada
- Se tiene que crear un documento.
- Se tiene que documentar los objetivos.
- Se tienen que definir el alcance
- Cronograma del proyecto con las tareas a desarrollar, fechas y ruta critica.
- Contexto del proyecto

Módulo 3:

Módulo 4: