Propuesta de guión

Lo más importante de todo es:
ATENEOS A CUMPLIR CON LOS REQUISITOS FUNDAMENTALES DE LA PRÁCTICA

Todo aquello indicado como *Opcional* en este guión es eso, opcional. Ignoradlo hasta que os sobre el tiempo. Dividir el proyecto en apartados:

1. Carga y preprocesado de los datos

Para cargar nuestro Google Drive como si fuera un sistema de archivos local (en otras palabras, como si fuera nuestro propio ordenador), no tenemos más que ejecutar:

```
from Google.colab import drive
drive.mount('/content/drive')
```

- Eliminar columnas irrelevantes
- Convertir variables categóricas (labels, features) en variables numéricas
- División train/val/test
- Estudio de posibles valores missing
- Normalización de los datos
- Confirmar, cerciorar y asegurar que las particiones mantienen el orden entre datos tabulares, imágenes, y sus correspondientes etiquetas. Recordad, para poder comparar dos modelos, estos tienen que haber sido entrenados y testeados sobre los mismos datos. De cualquier otra manera la comparativa no es formalmente correcta y puede llevar a conclusiones incorrectas.

2. Hito 1

Aplicación de lo aprendido en las sesiones 1-3. Asignar prioridad a obtener un modelo funcional. Visualizar aprendizaje. Potencial búsqueda de hiperparámetros.

Opcional: Modelo 1D basado en los píxeles de las imágenes, o una representación de las mismas (por ejemplo, tras primero aplicar un PCA).

3. Hito 2

Aplicación de lo aprendido en las sesión 4-5. Prioridades:

- Empleo de un modelo preentrenado para obtener embeddings al final de la extracción de características (transfer-learning offline).
- Entrenamiento tan sólo de un clasificador sobre esos embeddings.
- Visualizar aprendizaje.

Opcional: Definición de un modelo propio / Transfer-learning online / Re-entrenamiento (fine-tuning). Notad que ello conlleva calcular el gradiente en más (o en todas) las capas. Incluir técnicas de interpretabilidad (sesión 5).

4. Hitos 3 & 4

Aplicación de lo aprendido en la sesión 6 (o 7). Consideración acerca de la longitud de las representaciones aprendidas por cada modalidad. Si una es mucho mayor que otra, dominará en cualquier aprendizaje posterior. Recordad, la división hecha entre train/val/test debe seguir vigente aquí.

Importante: Emplear los modelos entrenados anteriormente para extraer predicciones/embeddings. Optimizar tiempo de cómputo en modelo de clasificación final.

5. Presentación de los resultados

Ejecución de los diferentes modelos sobre el conjunto de test reservado.

Opcional: ¿Funciona igual un modelo para todas las clases? ¿Y para todas las edades? ¿Y para ambos sexos?

6. Discusión de resultados

Breve discusión donde se dé una interpretación razonada y justificada en base a todo lo anterior acerca del buen (o mal) funcionamiento de los diferentes modelos. Previsión de trabajo a futuro.

Notas adicionales

Desarrollar la práctica (el código) empleando CPU es algo recomendado. Al desarrollar código, limitar a 1 época el entrenamiento de modelos, sólo para confirmar que el código es operativo. Conectar a GPU y ejecutar con todas las prestaciones cuando podamos simplemente centrarnos en experimentar con hiperparámetros y configuraciones.

Si la función de pérdidas es NaN, hay un problema con los datos.

Normalizar las entradas y las salidas es buena costumbre.

Minimizar uso de recursos, ser inteligente en el procesado de los datos. Es mejor pensar con un cuaderno y lápiz 20 minutos y desarrollar en 5, que estar 40 minutos programando sin tener un plan.

Si encontramos problemas con Google Colab, o no entendemos algo, contactar con el profesor