# Tarea 2 - Introducción a la Ciencia de Datos

2025

Esta tarea es la continuación de la Tarea 1, por lo que se utilizarán los mismos datos y se puede reutilizar cualquier parte del código de dicha tarea.

En el repositorio intro-cd bajo la carpeta ***Tarea_2***, se encuentra el notebook de referencia sobre el que se deberá trabajar.

La entrega se debe dejar disponible en el mismo repositorio de la Tarea 1. Los archivos a evaluar deben estar en la *branch* principal (*main*). En dicha rama no debe haber *commits* posteriores a la fecha de entrega estipulada. Los archivos que deben estar presentes en el repositorio son:

- Un informe en formato PDF incluyendo todos los resultados relevantes, y este será en general el trabajo a evaluar.
- Se revisará todo el código implementado (al menos un notebook y posibles scripts adicionales), pero solo si existen dudas sobre la implementación. De todas formas, se evaluará la documentación de los notebooks para que otra persona pueda comprenderlos y, por ejemplo, volver a ejecutarlos con los mismos datos.

Agregar un archivo README.md al repositorio, con indicaciones básicas, por ejemplo, indicando cual es el informe de cada tarea, notebooks o scripts utilizados para responder las preguntas y en caso de haber más de uno, indicar para qué se usó cada uno.

Recuerde que el propósito de la tarea es poner en práctica algunos conceptos clave de aprendizaje automático, **no** necesariamente profundizar en técnicas específicas de procesamiento de lenguaje natural.
El foco debe estar en entender el proceso y la interpretación de los resultados a nivel conceptual, y no en mejorar las métricas indefinidamente. Si los resultados no son buenos, la idea es generar hipótesis y buscar posibles razones.

## Parte 1: Dataset y representación numérica de texto

Para esta parte, se utilizará como referencia la sección Extracting features from text files de la documentación oficial de scikit-learn.

1. En el notebook de la tarea, se utilizará un conjunto de datos reducido de los tres candidatos con más discursos. Se espera que utilice su propia versión de la función *clean_text()* de la Tarea 1.

Particione los datos para generar un conjunto de test del 30% del total, utilizando ***muestreo estratificado***.

**Sugerencia**: utilice el parámetro *stratify* de la función *train_test_split* de scikit-learn y fije también el valor de *random_state* para obtener resultados reproducibles.

2. Genere una visualización que permita verificar que el balance de discursos de cada candidato es similar en train y test.
3. Transforme el texto del conjunto de entrenamiento a la representación numérica (*features*) de conteo de palabras o *bag of words*. Explique brevemente cómo funciona esta técnica y muestre un ejemplo. En particular, explique el tamaño de la matriz resultante y la razón por la que es una matriz es *sparse*.

   **Sugerencia:** puede ser útil imaginar qué sucedería con la memoria RAM requerida si no estuviéramos trabajando con un conjunto de datos tan reducido.
4. Explique brevemente qué es un *n-grama*. Obtenga la representación numérica *Term Frequency - Inverse Document Frequency*. Explique brevemente en qué consiste esta transformación adicional.
5. Muestre en un mapa el conjunto de entrenamiento, utilizando las dos primeras componentes **PCA** sobre los vectores de *tf-idf*. Analice los resultados y compare qué sucede si utiliza el filtrado de *stop_words* para idioma inglés, el parámetro *use_idf=True* y *ngram_range=(1,2)*. Opcionalmente, también puede analizar qué sucede si no elimina los signos de puntuación.

   ¿Se pueden separar los candidatos utilizando sólo 2 componentes principales? Haga una visualización que permita entender cómo varía la varianza explicada a medida que se agregan componentes (e.g: hasta 10 componentes).

## Parte 2: Entrenamiento y Evaluación de Modelos

1. Entrene el modelo Multinomial Naive Bayes para clasificar los discursos según a qué candidato o candidata pertenece el texto. A continuación, utilice el modelo para clasificar los discursos del conjunto de test, y reporte el valor de *accuracy* y la matriz de confusión. Reporte el valor de *precision* y *recall* para cada candidato. Explique cómo se relacionan estos valores con la matriz anterior.

   ¿Qué problemas puede tener el hecho de mirar sólamente el valor de *accuracy*? Considere qué sucedería con esta métrica si el desbalance de datos fuera aún mayor entre candidatos.

   **Sugerencia:** utilice el método *from_predictions* de *ConfusionMatrixDisplay* para realizar la matriz.
2. Explique cómo funciona la técnica de validación cruzada o *cross-validation*. Implemente una búsqueda de hiperparámetros usando GridSearchCV. Genere una visualización que permita comparar las métricas (e.g: *accuracy*) de los distintos modelos entrenados, viendo el valor promedio y variabilidad de las mismas en todos los *splits* (e.g: en un gráfico de violín).

3. Elija el mejor modelo (mejores parámetros) y vuelva a entrenar sobre todo el conjunto de entrenamiento disponible (sin quitar datos para validación). Reporte el valor final de las métricas y la matriz de confusión. Discuta las limitaciones de utilizar un modelo basado en *bag-of-words* o *tf-idf* en cuanto al análisis de texto.
4. Evalúe al menos un modelo más (dentro de scikit-learn) aparte de *Multinomial Naive Bayes* para clasificar el texto utilizando las mismas *features* de texto. Explique brevemente cómo funciona y compare los resultados con el anterior.
5. Evalúe el problema cambiando al menos un candidato. En particular, observe el (des)balance de datos y los problemas que pueda generar, así como cualquier indicio que pueda ver en el mapeo previo con PCA. Puede ser útil comentar acerca de técnicas como sobre-muestreo y submuestreo, no es necesario implementarlo.
6. Busque información sobre al menos una técnica alternativa de extraer *features* de texto. Explique brevemente cómo funciona y qué tipo de diferencias esperaría en los resultados. No se espera que implemente nada en esta parte.
7. **(Opcional)** Repetir la clasificación con los tres candidatos con más discursos, pero esta vez clasificando a nivel de párrafos y no de discursos enteros.