AAGE - Práctica 2: Aprendizaje federado y continuo

Grupo 2: Andrés Lires Saborido, Ángel Vilariño García

Curso 2025-2026

Parte I: Aprendizaje Federado con Flower

Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo. Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo. Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo. Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo.

Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo. Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo. Esto es un ejemplo de texto. Solo para rellenar espacio y ver cómo queda el formato. Repite esto varias veces para simular un documento más largo.

Parte II: Aprendizaje Continuo con River

En esta parte de la práctica, se trabajará con el dataset de *Electricity*, el cual tiene 45312 instancias y 8 atributos. El objetivo es predecir si el precio de la electricidad subirá o bajará en función de los atributos disponibles. A continuación, se describen las distintas experimentaciones realizadas con modelos de aprendizaje para *data streaming*.

De *batch* a *streaming*

En primer lugar, se entrenará el modelo *Gaussian Naive Bayes* utilizando la librería `scikit-learn` en un enfoque de *batch learning*. Pero antes, fue necesario convertir el dataset a un *array* de NumPy y dividir en conjunto de entrenamiento (70%) y de test (30%). La evaluación del modelo tuvo como resultado una precisión del 75.39%.

A continuación, se implementó el mismo modelo utilizando la librería `River`, que está diseñada para el aprendizaje continuo. El modelo fue entrenado de manera incremental, procesando una instancia a la vez y evaluando su precisión después de cada instancia. El modelo alcanzó una precisión final del 72.87%, ligeramente inferior al enfoque de *batch learning*, posiblemente debido a la naturaleza secuencial del aprendizaje continuo, como se puede observar en la Figura 1.

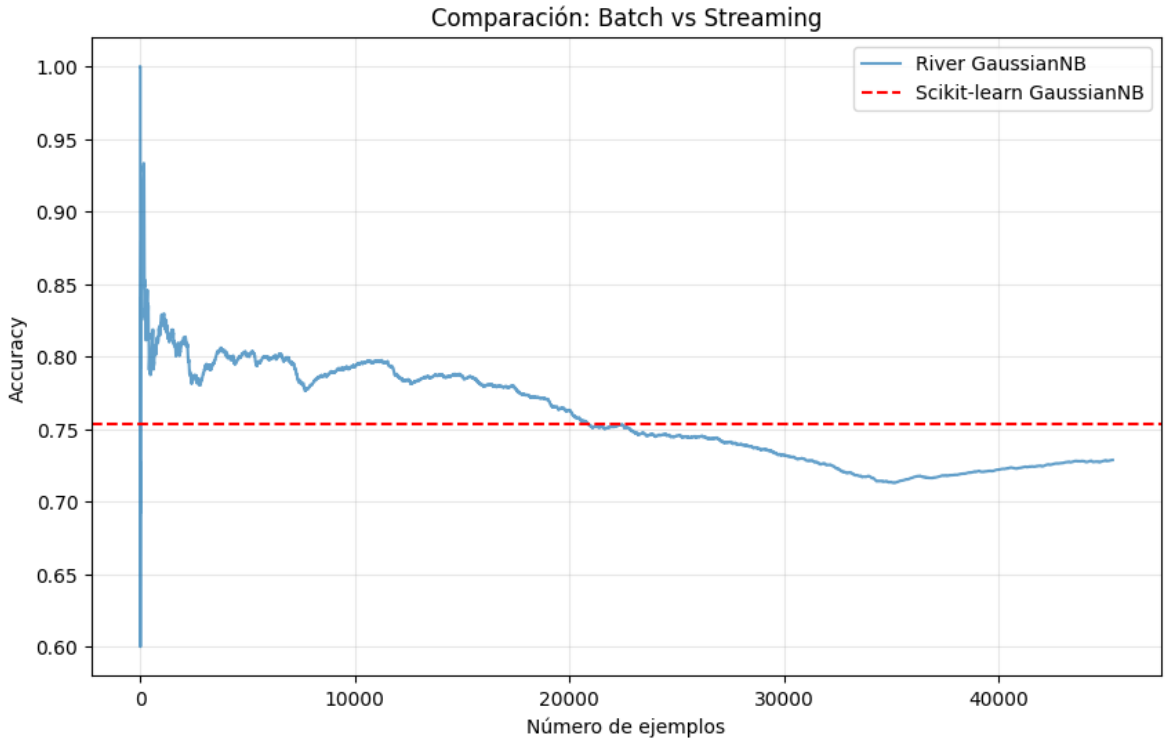

Figura 1: Comparación de precisión entre aprendizaje batch y streaming

Manejo de *concept drift*

Para abordar el *concept drift*, se creó un detector de cambios utilizando el método ADWIN (Adaptive Windowing). Este detector monitorea el rendimiento del modelo y ajusta su ventana de datos cuando detecta un cambio significativo en la distribución de los datos. A medida que van llegando los datos, se evalúa el rendimiento del modelo y si la predicción del dato actual es incorrecta, se actualiza el detector ADWIN con un valor de 1; si es correcta, se actualiza con un valor de 0. Cuando ADWIN detecta un cambio, se reinicia el modelo para adaptarse a la nueva distribución de datos.

En la Figura 2 se puede observar cómo el modelo maneja el *concept drift*, detectando 54 cambios a lo largo del flujo de datos y estabilizándose a lo largo del flujo, al contrario que el modelo anterior que presentaba más fluctuaciones en su precisión. El modelo con manejo de *concept drift* alcanzó una precisión final del 80.37%.

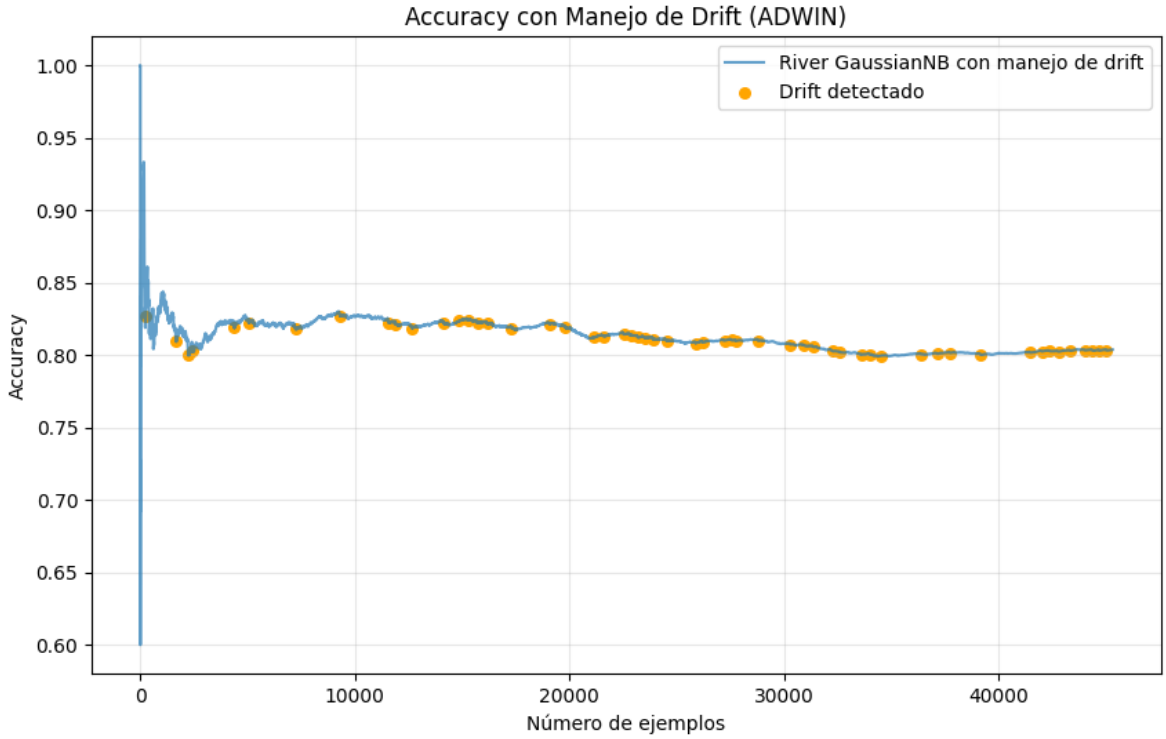

Figura 2: Accuracy con manejo de drift (ADWIN)

Modelos adaptativos

Finalmente, se implementaron dos modelos adaptativos: un *Hoeffding Adaptive Tree* (HAT) y un *Adaptive Random Forest* (ARF). Ambos modelos están diseñados para adaptarse automáticamente a los cambios en la distribución de los datos sin necesidad de reiniciar el modelo manualmente. Los dos modelos fueron entrenados y evaluados de la misma manera que el modelo *Gaussian Naive Bayes* de *streaming*.

El modelo HAT obtuvo una precisión final del 81.50%, mientras que el modelo ARF alcanzó una precisión del 89.43%.