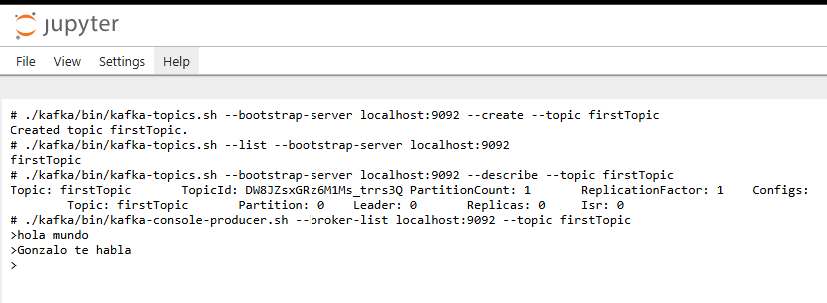
Práctica Optimización para Grandes Volúmenes de Datos: Data streaming con Apache Kafka

Miembros del grupo: Álvaro Fraile, Esteban Martínez, Jaime Álvarez, Alejandro Mendoza

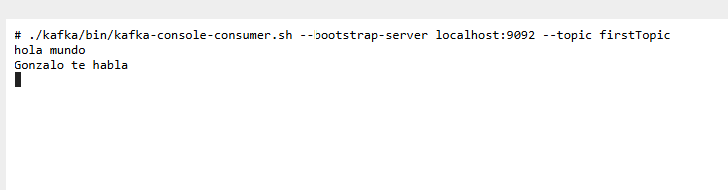
## Parte 1 (Obligatoria): Primeros pasos con Apache Kafka (1 punto)

### 1.1 Sigue las instrucciones de la presentación para crear un publicador que publique a un tópico de Kafka el contenido escrito por la línea de comando y un suscriptor que reciba esos mensajes. Adjunta capturas de ambas pantallas (1 punto).

**Productor:**



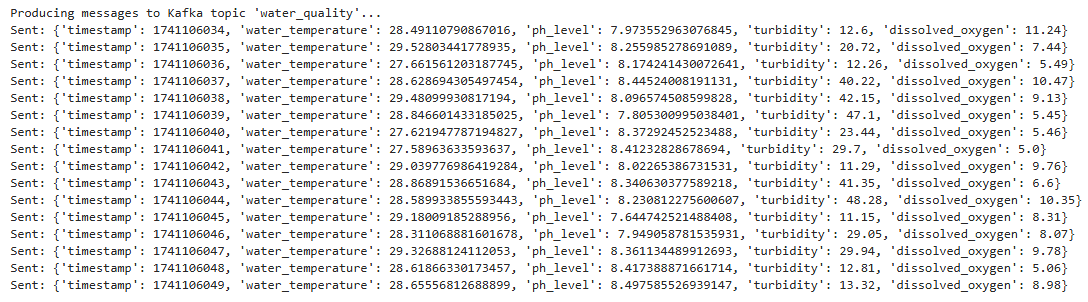
**Consumidor:**



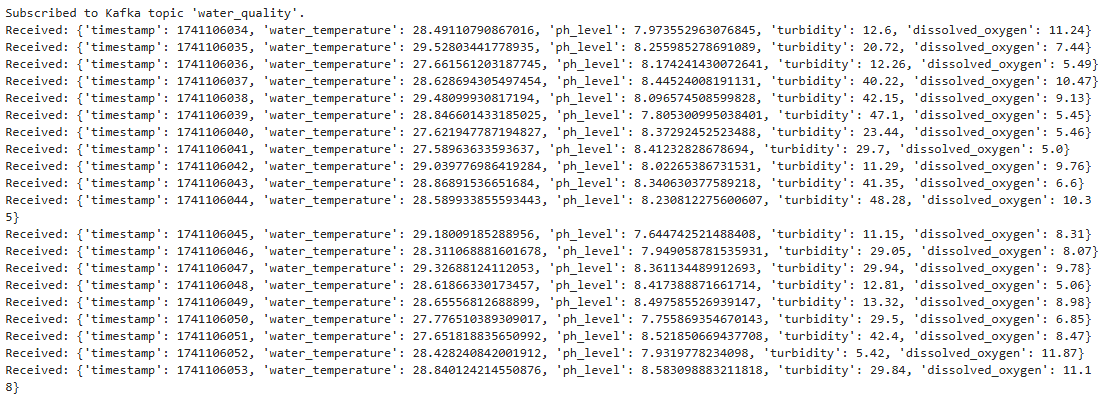
## Parte 2 (Obligatoria): Data streaming usando kafka-python (6 puntos)

### 2.1 Ejecuta las celdas de los archivos water\_data\_producer\_pr\_2.ipynb y water\_data\_consumer\_pr\_2.ipynb y adjunta capturas de los resultados (1 punto).

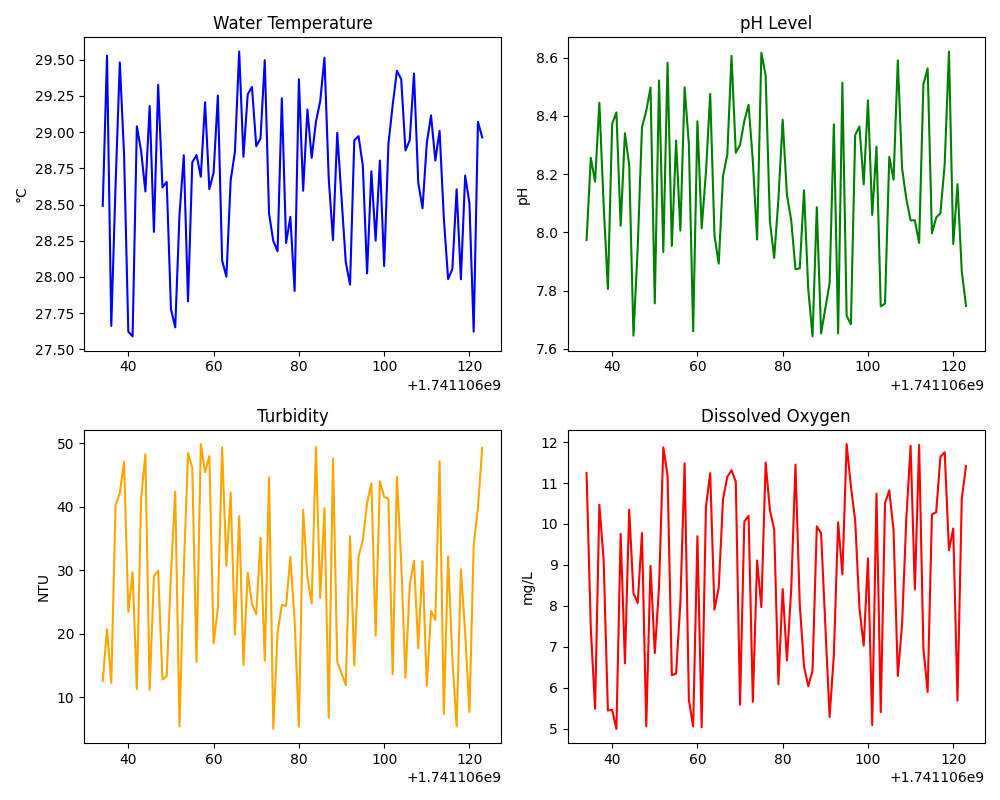
**Productor:**



**Consumidor:**



### 2.2 Adjunta foto del gráfico generado por el consumidor (0.5 puntos)

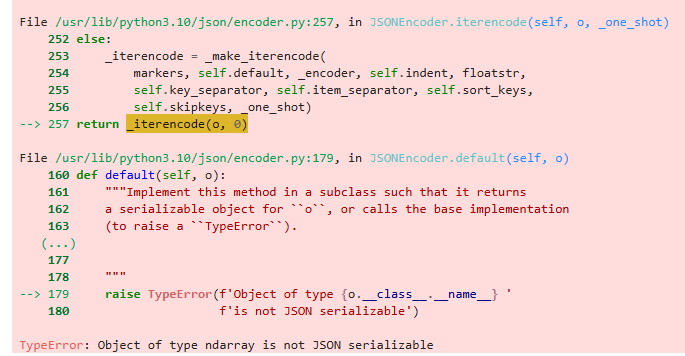


### 2.3 En la configuración del consumidor, ¿qué efecto tiene cambiar la propiedad “auto\_offset\_reset”. Prueba diferentes valores y adjunta tus conclusiones (0.5 puntos).

El parámetro **auto\_offset\_reset** cuando se configura el consumidor de Kafka se define los desplazamientos/offsets para cuando un consumidor se une al grupo o cuando no hay offsets previamente creados. Determinando así desde que punto del histórico del productor puede leer el consumidor dentro del topic de este.

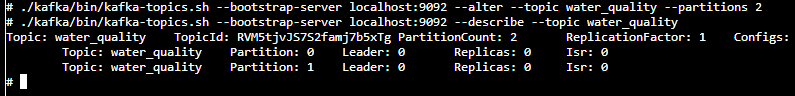
* “earliest”: Cuando un consumidor se suscribe a un topic empezará leyendo el desde el mensaje más antiguo posible. Especialmente útil para casos donde se deba procesar todos los datos relacionados con el topic. Cabe resaltar que el offset puede ser limitado y solo se guardarán los X mensajes más antiguos, pudiendo perderse datos a lo largo del tiempo
* “latest”: Cuando un consumidor se suscribe a un topic empieza procesando el último mensaje publicado y los mensajes que se van publicando a partir de ese punto.
* “none”: Cuando un consumidor se suscribe a un topic y no hay ningún dato almacenado en la partición, el consumidor lanzará un error.

### 2.4 ¿Qué ocurre si intentamos mandar a través del productor un array de numpy (numpy.array)? Adjunta una imagen probándolo y reflexiona sobre los resultados (0.5 puntos).

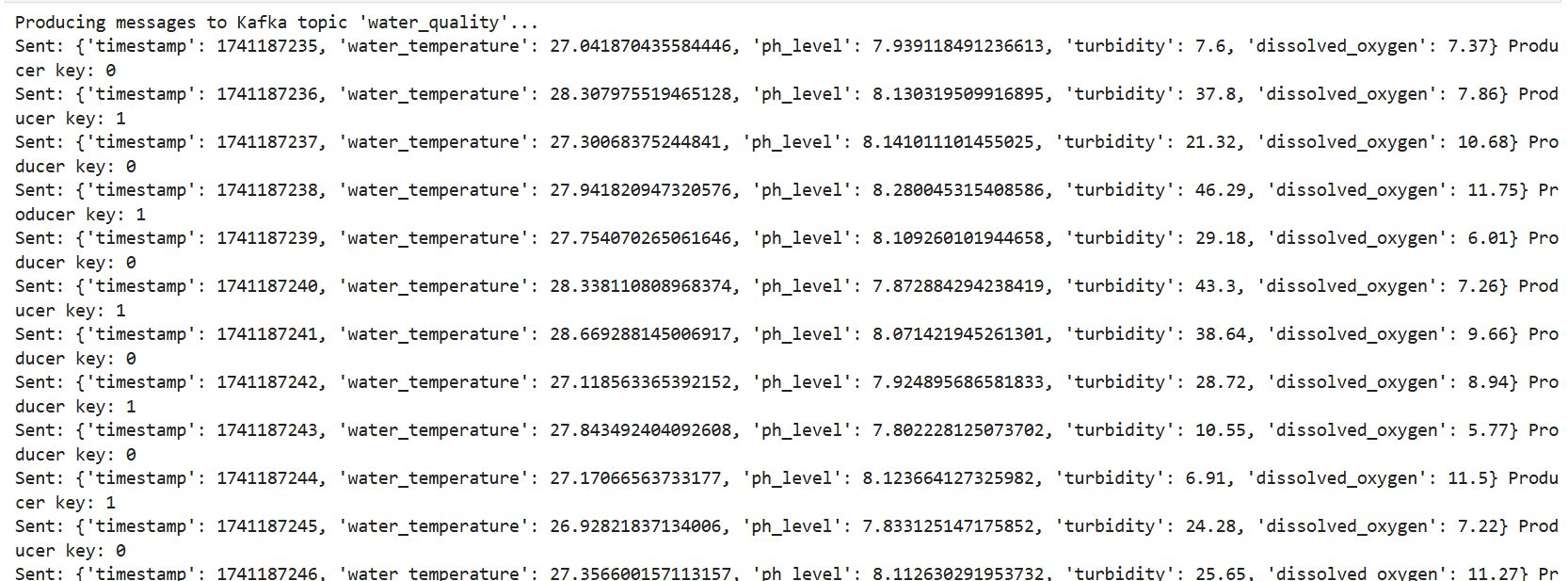


Cuando se intenta mandar un NumpyArray a través del productor de kafka, se genera el error de tipo TyperError, esto indica que este tipo de objetos no puede serializarse en formato JSON como se requiere. Por la documentación se puede concluir que la librería de json de python no reconoce como serializables objetos de tipo numpy.array de forma nativa. Para esta ejecución el error ocurre en la función default de encoder cuya tarea consiste en convertir el objeto, que se pasa por parámetro, a formato JSON. Para solventarlo utilizamos la función tolist en el numpy.array así obtenemos una lista que si es serializable por la librería de json

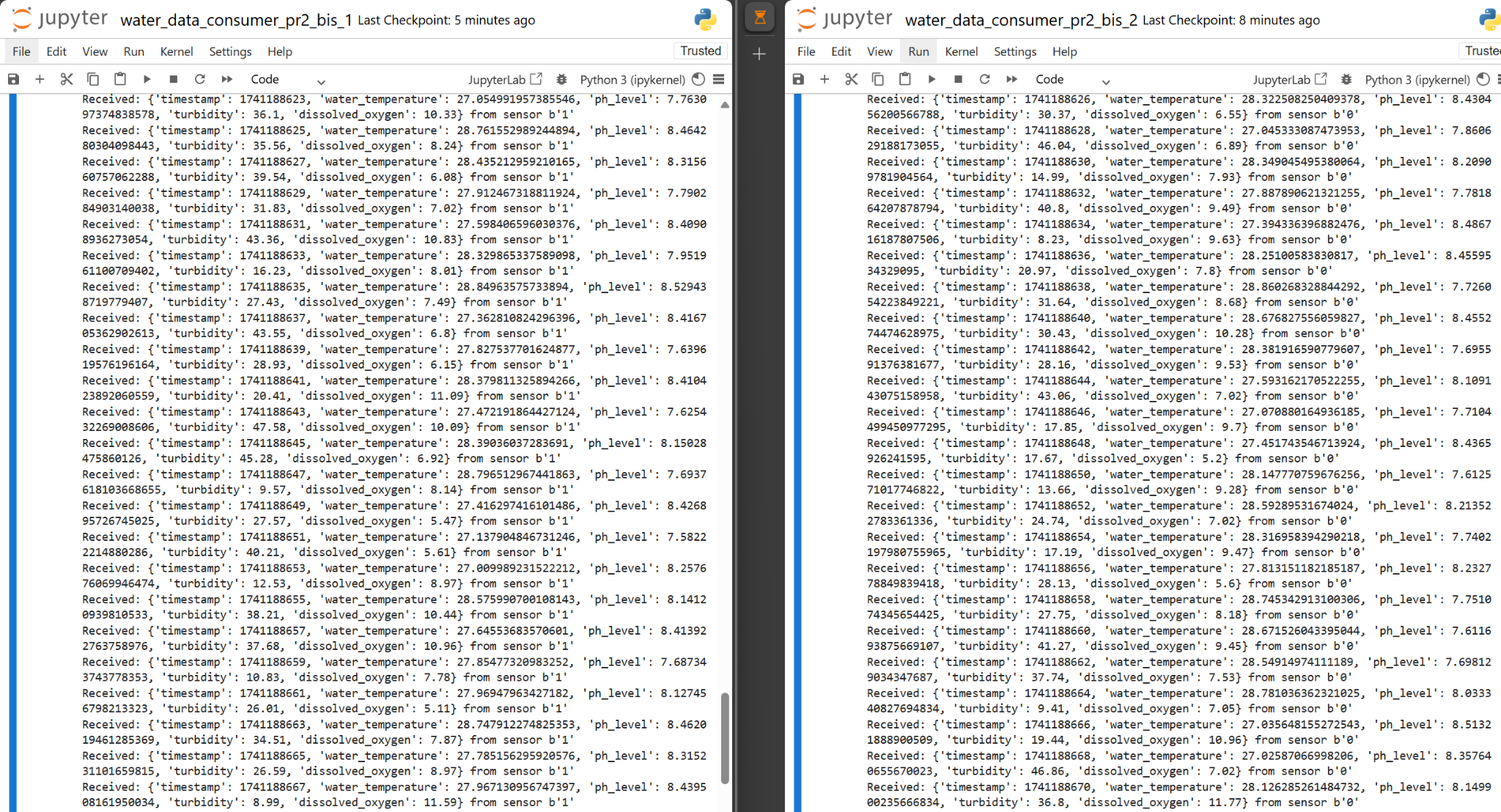
### 2.5 Modifica el tópico para tener 2 particiones. Adjunta captura de la descripción resultante del tópico (0.5 puntos).

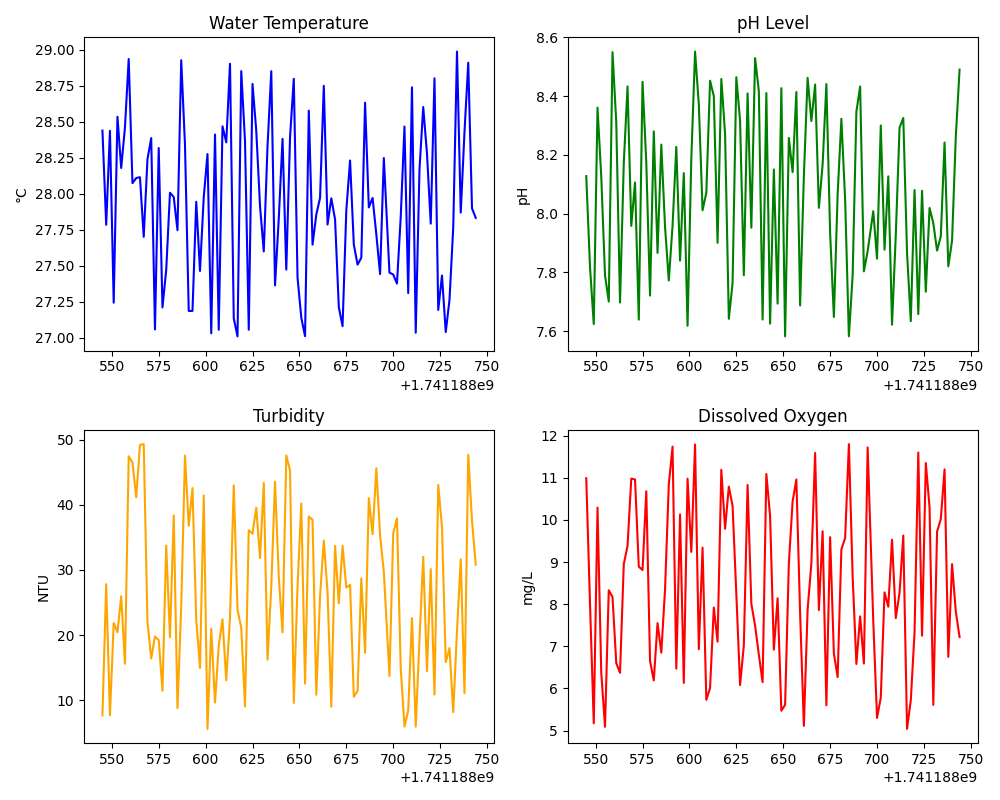


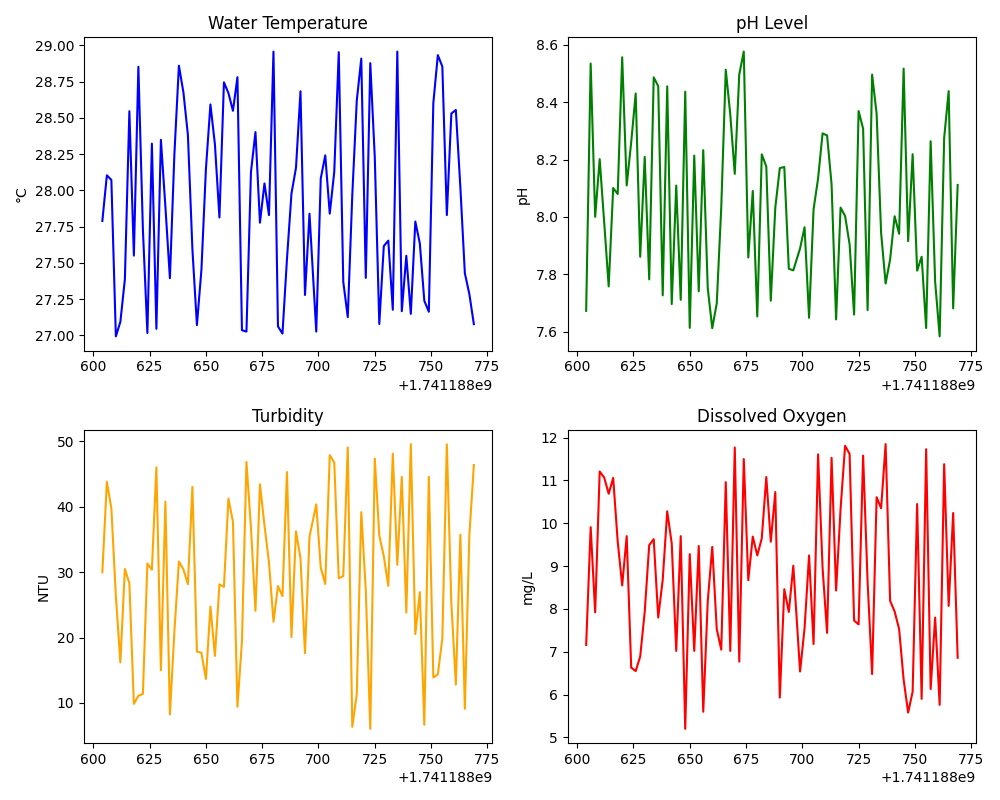
### 2.6 Modifica el archivo water\_data\_producer\_pr\_2.ipynb para que se creen múltiples productores con un id de sensor único que se envíe como key del mensaje.. Adjunta captura en la que se muestre el envío de mensajes de varios productores en el mismo notebook (1.5 puntos).



### 2.7 Modifica el archivo water\_data\_consumer\_pr\_2.ipynb para tener consumidores con un group\_id común. Cada sensor\_id debe producir un plot con los datos producidos por ese sensor\_id. Crea una copia de water\_data\_consumer\_pr\_2.ipynb modificado y adjunta captura de los dos notebooks modificados ejecutándose a la vez y gestionando diferentes sensor\_id y los plots resultantes (1.5 puntos).

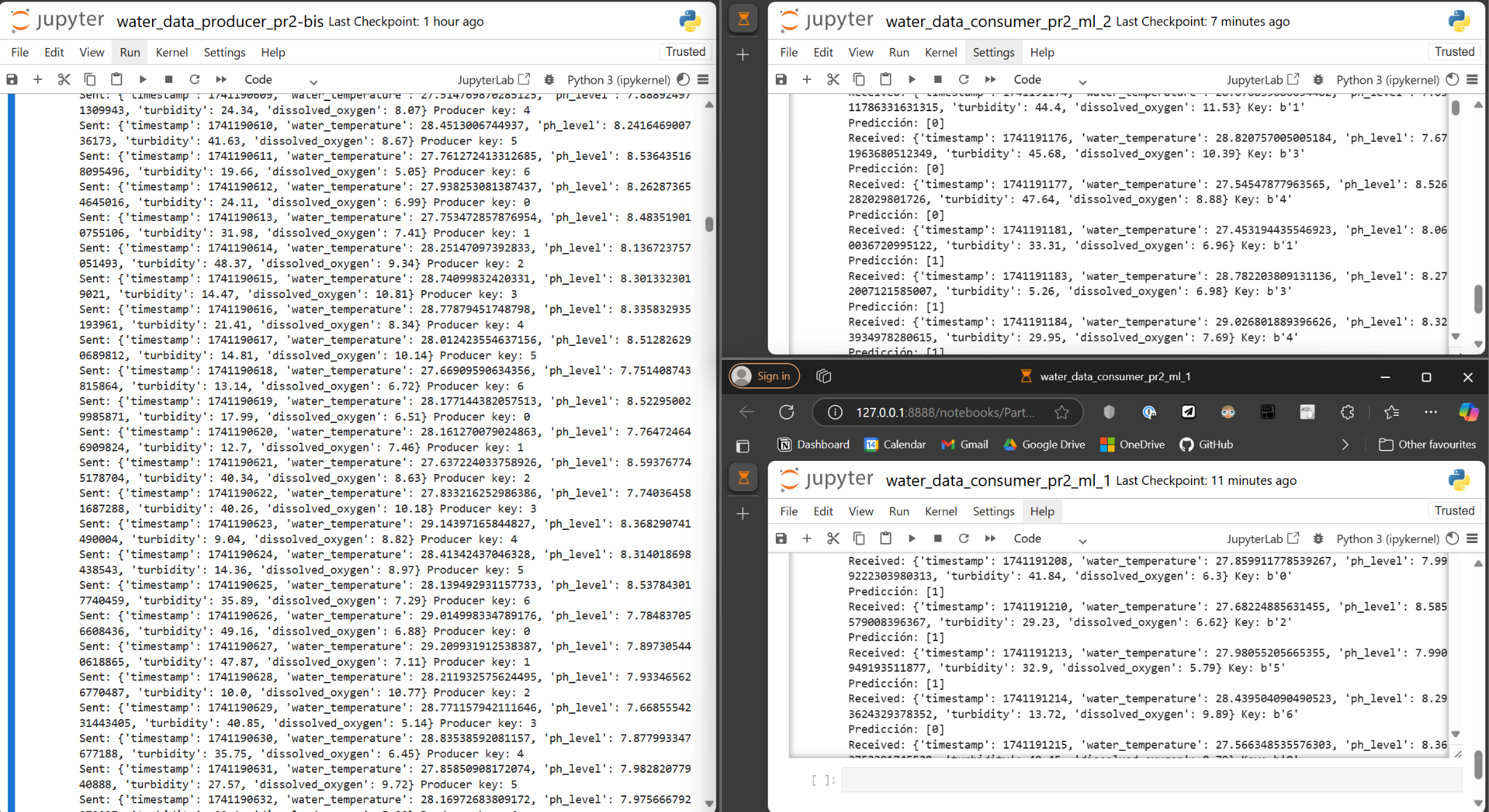


CONSUMER 1:

CONSUMER 0:

## Parte 3 (Opcional): Integrando procesos ML en data streaming (3 puntos)

### 3.1 Descarga el archivo aeration\_classifier.joblib. Modifica el archivo water\_data\_consumer\_pr\_2.ipynb para realizar inferencia con el modelo de clasificación, que es un RandomForest que acepta como input water\_temperature, ph\_level, turbidity and dissolved\_oxygen\_level y da un resultado (0,1) indicando si hay un proceso de aireación presente o no. Añade un productor que publique los resultados de inferencia, publicando la posibilidad de que el ejemplo pertenezca a cada una de las dos clases (Mediante el uso de predict\_proba). Imprime estos resultados de inferencia por pantalla y adjunta una captura de la ejecución con 2 procesadores ML y múltiples productores activos. (1.5 puntos).



### 3.3 Crea otro notebook con un consumidor que se suscriba a los resultados de inferencia publicados por el productor del apartado 3.2 y que los agregue en tres gráficos para cada sensor que contengan:

### 3.3.1 Estado de aireación para cada timestamp.

### 3.3.2 Confianza de la clase de aireación por timestamp cuando es la clase escogida.

### 3.3.3 Confianza de la clase de no aireación por timestamp cuando es la clase escogida.

### Imprime por pantalla los resultados de inferencia recibida por pantalla y adjunta una captura de la ejecución con 2 procesadores ML y múltiples productores activos. Adjunta imágenes de todos los gráficos de paneles de control generados. (1.5 puntos).

