Laboratorio 4
Camilo Salinas, Felipe Bedoya, Nicolas Orjuela.

Contribuciones Individuales al Laboratorio:

Felipe Bedoya: Jupyter notebook con la construcción del pipeline y la explicación de cada uno de los pasos realizados
Nicolás Orjuela: Repositorio de GitHub con la API con dos URL habilitadas.
Camilo Salinas: Escenarios de prueba, Readme con instrucciones de instalación, Archivo de requerimientos “Requeriments.txt” y despliegue a Heroku.

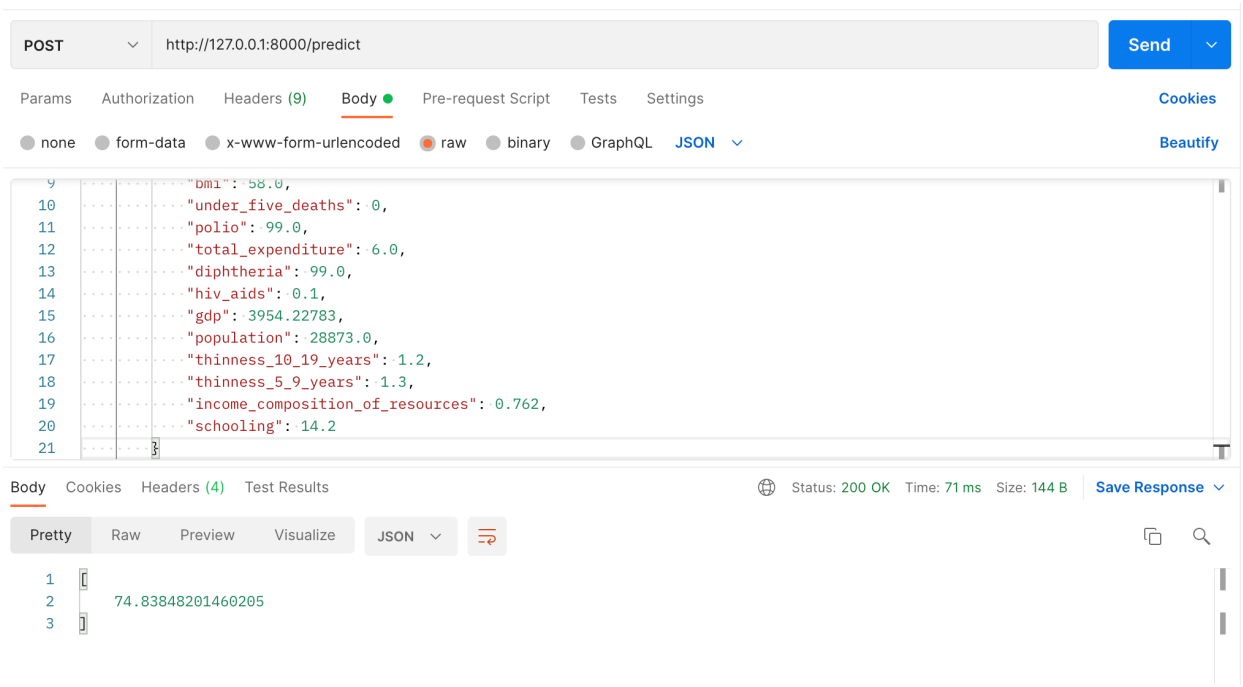
Escenarios de prueba:

Para este ejercicio hemos seleccionado 5 escenarios de prueba donde validaremos el funcionamiento del API. Entre los escenarios incluidos tenemos:

- Escenario normal con una predicción sobre los rangos.
- Escenario con datos por fuera de los rangos esperados, con una predicción mala por los mismos datos erróneos.
- Escenario con datos con campos incorrectos/erróneos, con una respuesta de error del API
- Escenario probando el R^2 con todos los datos de prueba reales, y el resultado de ese R^2
- Escenario probando el R^2 con algunos datos alterados para que se salgan de los rangos comunes y el R^2 se vea alterado.

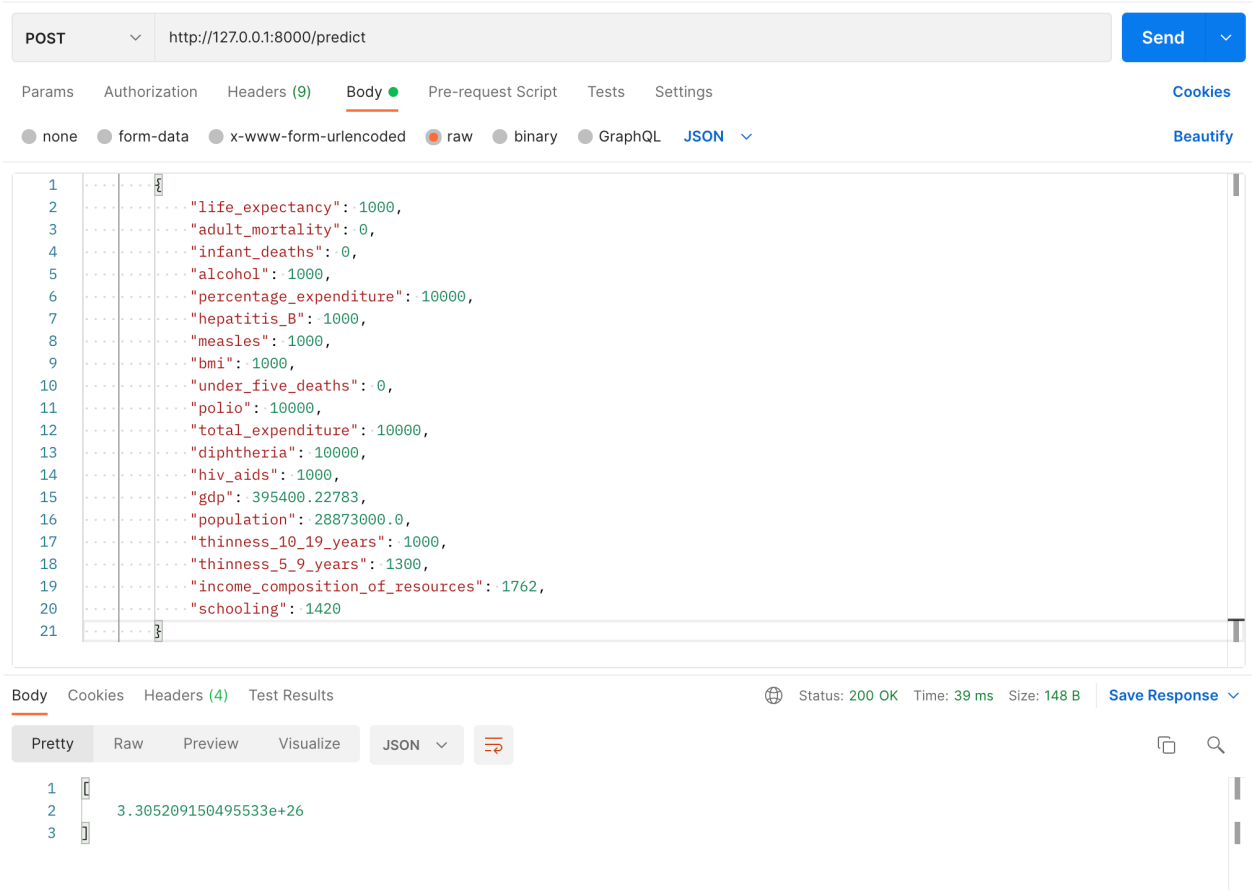
Escenario normal con una predicción sobre los rangos:

Los datos de este escenario se pueden observar en el JSON “PRUEBAdato1.json”, es un objeto de prueba dentro de los rango normales.



Cómo se puede observar en la prueba, el escenario es coherente y funciona de acuerdo a lo esperado.

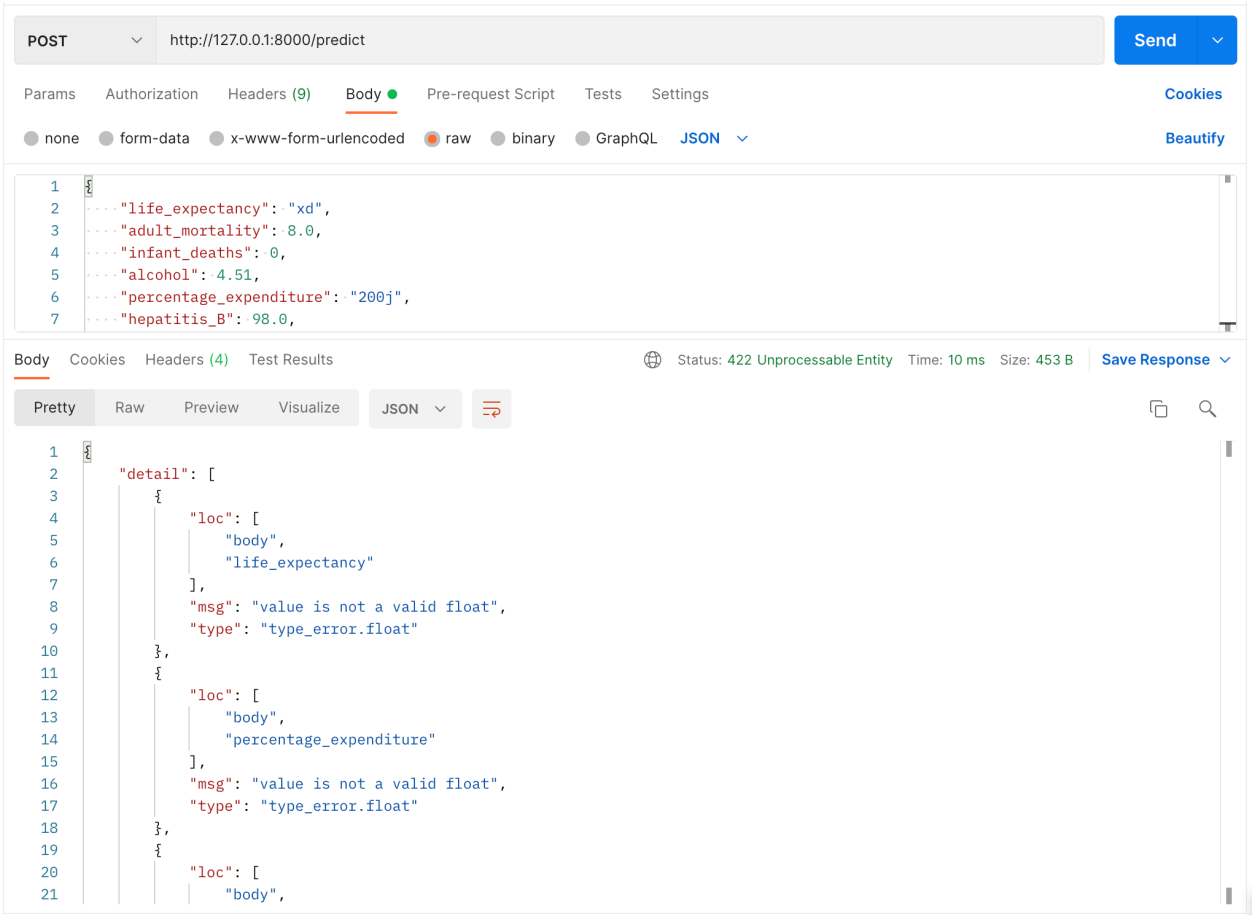
Escenario con datos por fuera de los rangos esperados, con una predicción mala por los mismos datos erróneos:
Los datos de este escenario se pueden observar en el JSON “PRUEBAdatoIrreal.json”, es un objeto de prueba con datos exagerados y totalmente irreales.
Al ejecutar la prueba se observa lo siguiente:



Cómo podemos ver el resultado es irreal dados la poca calidad de los datos otorgados.

Una estrategia de mitigación podría ser validar la entrada de los datos para que efectivamente limite la ejecución de aquellos datos que se salen demasiado de los rangos esperados.

Escenario con datos con campos incorrectos/erróneos, con una respuesta de error del API:
Los datos de este escenario se pueden observar en el JSON “PRUEBAdatoCamposErroneos.json”, es un objeto de prueba con datos erróneos, tales cómo strings.

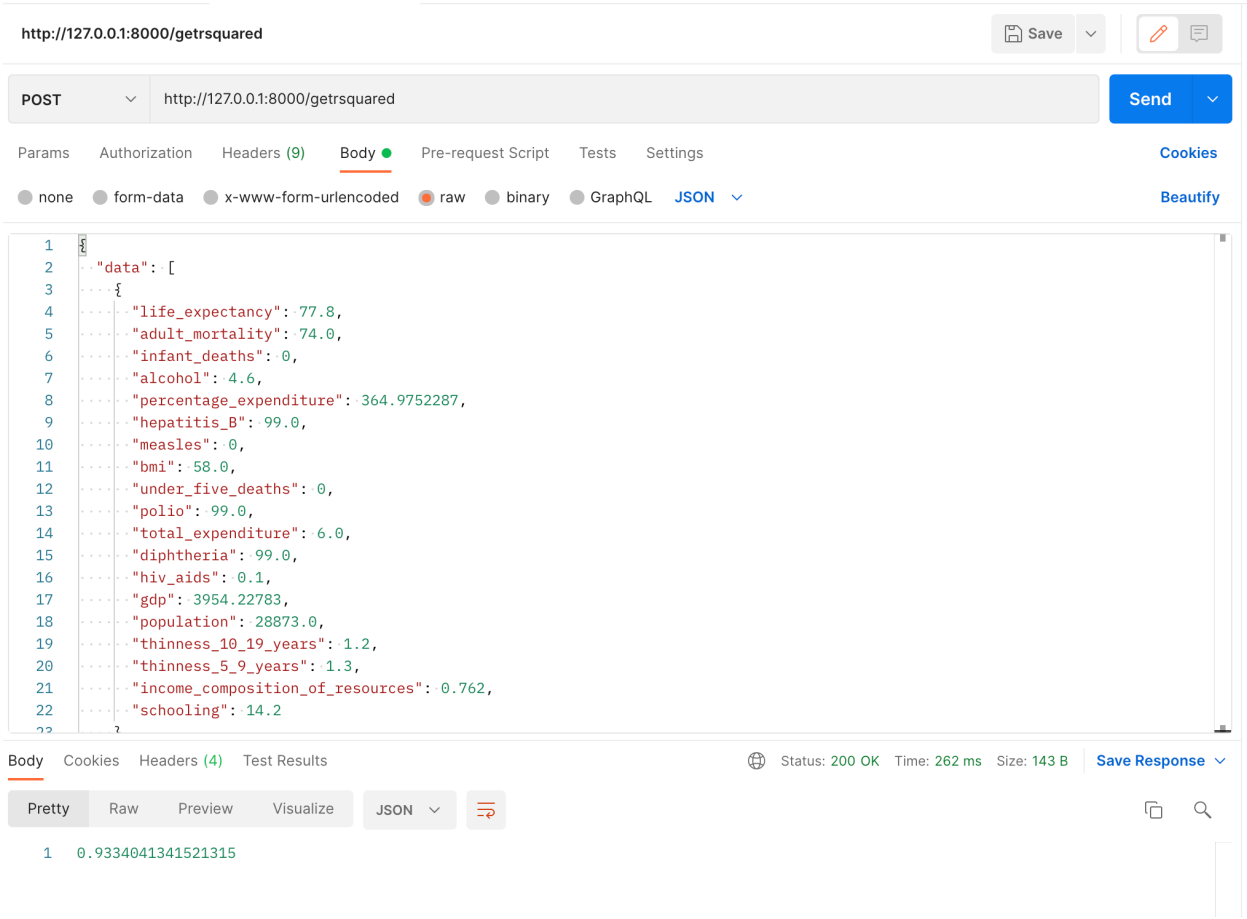


Como podemos ver el API reacciona de manera correcta a y avisa de los errores en los campos que no tenían el formato esperado.

Aunque este es un comportamiento esperado y no deberían pasar datos en formato distinto al numérico. Podría intentarse parsear los strings cómo numero y si es posible ejecutar la predicción. (Por ejemplo “100.000” → 100.000)

Escenario probando el R^2 con todos los datos de prueba reales, y el resultado de ese R^2:

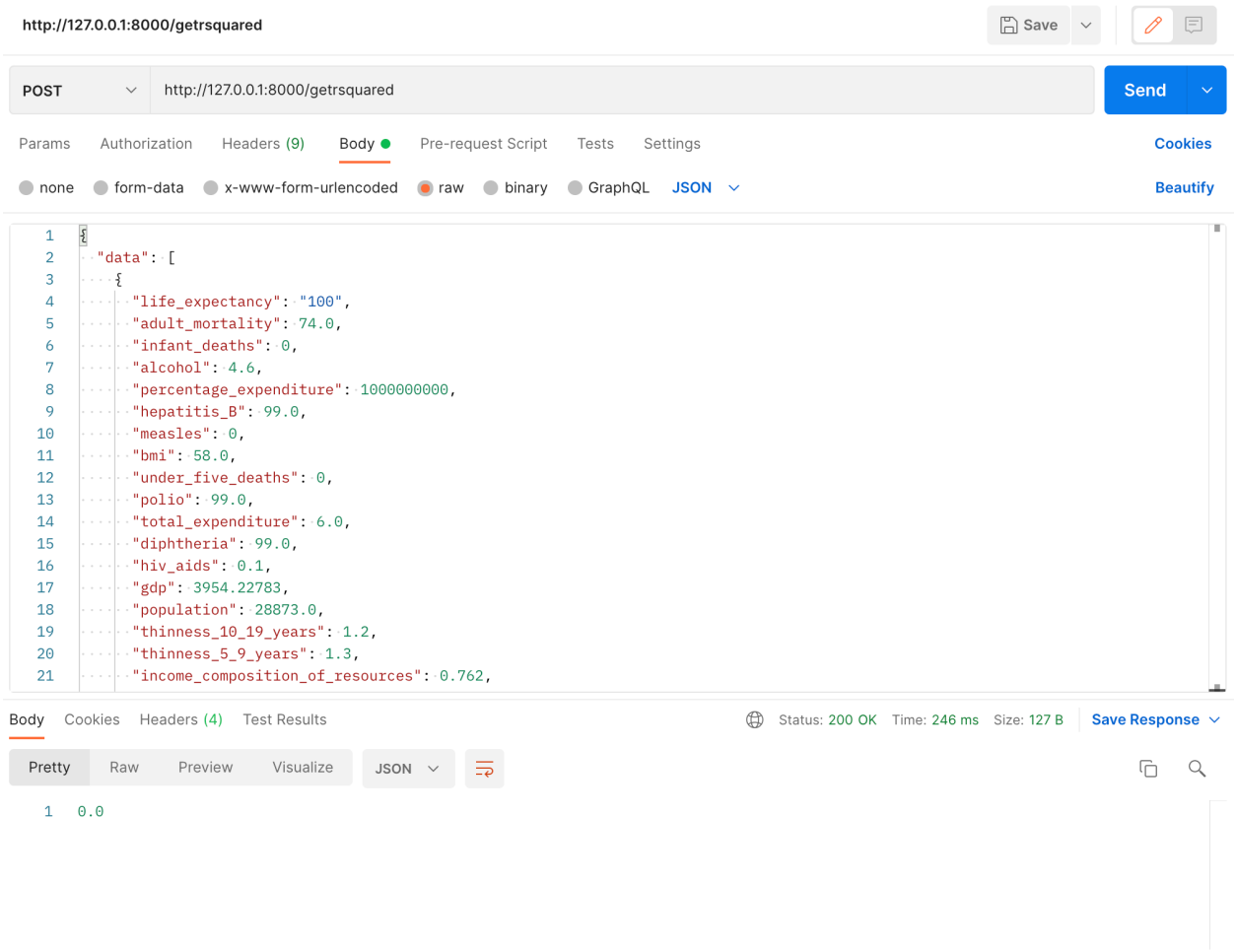
Los datos de este escenario se pueden observar en el JSON “PRUEBAgetSquared.json”, es un objeto que tiene en la llave “data” un arreglo con todos los elementos de prueba otorgados en “Datos recientes” para los cuales vamos a calcular el R^2, estos datos no han sido modificados ni alterados de ninguna manera más que en su representación de CSV a JSON.



Cómo se puede observar el R^2 tiene sentido.

Escenario probando el R^2 con algunos datos alterados para que se salgan de los rangos comunes y el R^2 se vea alterado:

Los datos de este escenario se pueden observar en el JSON “PRUEBAgetSquared.json”, es un objeto que tiene en la llave “data” un arreglo con todos los elementos de prueba otorgados en “Datos recientes” para los cuales vamos a calcular el R^2. Estos datos **han** sido modificados. Hemos remplazado algunos campos por datos irreales y otros por datos erróneos.



Como podemos observar el R^2 nos da 0.0 lo cual no tiene sentido.

Una estrategia de mitigación para este escenario es verificar la calidad de los datos y tal y como se sugirió más arriba, intentar parsear los strings a números para permitir más información. Quizá también se podría evaluar si hay campos erróneos en algunos objetos y descartarlos para hallar el R^2.