DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
Inteligencia de negocios 202220 – Laboratorio 4
PROFESORA: Haydemar Nuñez

Nombres	Apellidos	Código	Login
María Sofía	Álvarez López	201729031	ms.alvarezl
Brenda Catalina	Barahona Pinilla	201812721	bc.barahona
Alvaro Daniel	Plata Márquez	201820098	ad.plata

Informe de laboratorio #4

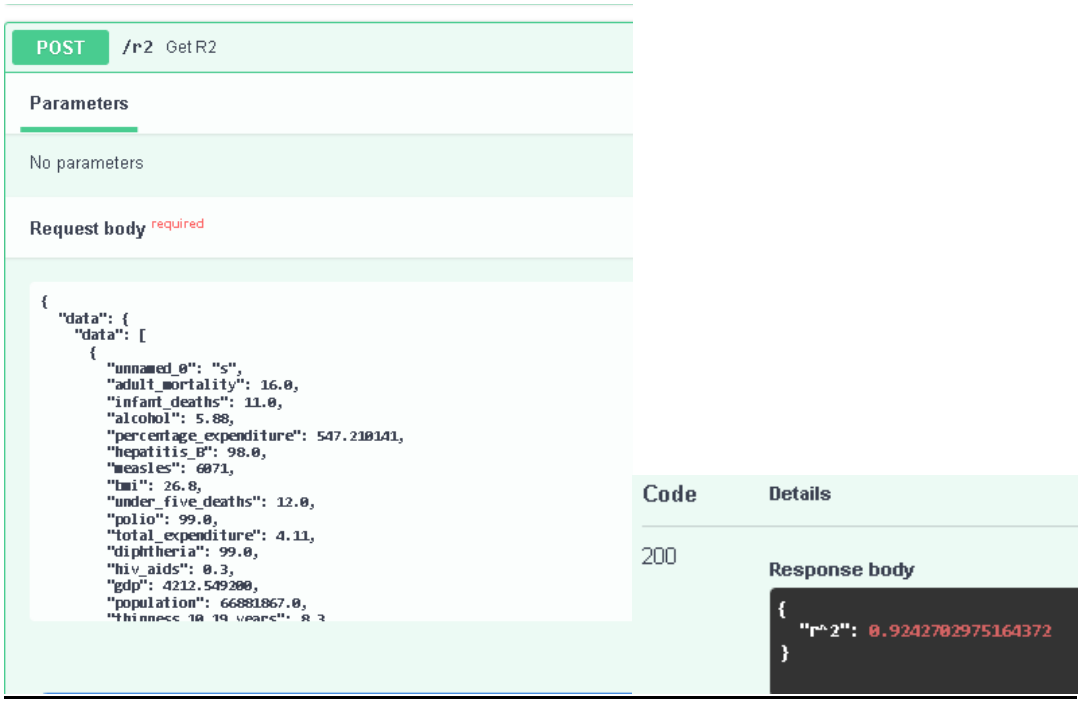
El objetivo de este laboratorio es reforzar el conocimiento adquirido en la construcción de Pipelines (y, preferiblemente -como se realizó-, implementar transformaciones personalizadas), exportar un modelo machine learning utilizando pickle y construir un API para montar el modelo en producción y realizar predicciones mediante peticiones HTTP. Esto último fue logrado tanto local como remotamente, en una instancia EC2 de AWS, cuya IP estática es http://3.228.160.169/ .

Escenarios de prueba para el API y análisis:

Para probar los escenarios, puede encontrar cada uno de los Json en la carpeta "Test". En los escenarios se pondrán los pantallazos de la prueba y el resultado que dio. Además de puntualizar el nombre de la prueba (del archivo).

En este proyecto, se plantearon 6 escenarios. Se buscó que fuese un espectro tal que arrojara predicciones coherentes, incoherentes y erróneas. Los escenarios previamente descritos se muestran a continuación:

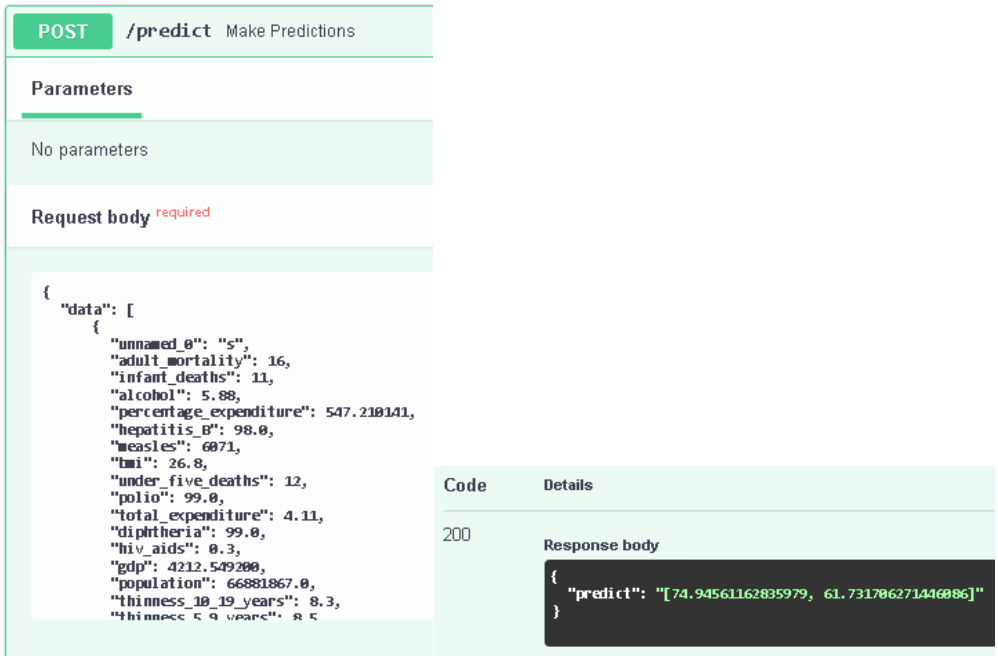
- **R^2 coherente:**
  Nombre del archivo: R2Coherente1



Para este escenario calculamos en Jupyter el r^2 y este es el mismo que el r^2 que obtuvimos con la API. Esta fue la petición realizada en el módulo de FASTAPI. Los resultados de la petición en Postman se presentan a continuación:

La predicción tiene sentido puesto que los datos utilizados fueron seleccionados cuidadosamente tal que siguieran las mismas tendencias que algunos de los suministrados en el archivo de entrenamiento con el que se entrenó el modelo.

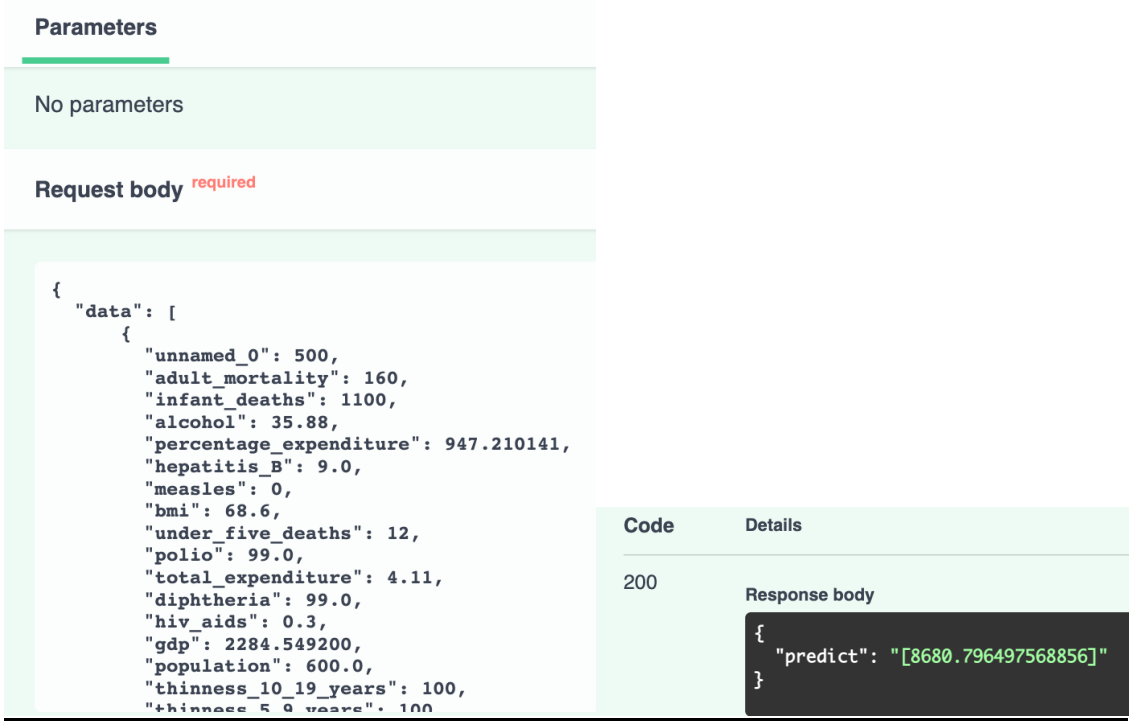
Nombre del archivo: PrediccionCoherente1



En esta predicción se puede ver que se ponen a prueba dos casos, y podemos ver que el life expectancy que resulta es muy cercano al que es en realidad. Para el primer caso, el valor de la variable de interés es de 74.94, cuando en realidad debió ser de 73.7. Para la segunda, la predicción dio 61.73, siendo en realizada 59.2. Esta fue la petición realizada en el módulo de FASTAPI. Los resultados de la petición en Postman se presentan a continuación:

Note que los resultados son iguales a los obtenidos con FastAPI.

- **Predicción incoherente:**
  Nombre del archivo: PrediccionIncoherente1



Podemos ver que varios de estos datos no tienen sentido, entre estos la cantidad de población y la cantidad de muertes de niños no es coherente, de acuerdo con las suposiciones e información del negocio. Adicional a esto la delgadez en niños de 10 a 19 años y 5 a 9 años es alta. Sin embargo, se predice que la esperanza de vida es 8680.8 años, lo cual es imposible para cualquier país del mundo.

La realidad sobre esta predicción tan elevada recae en que el income composition of resources, que de acuerdo con el análisis cualitativo es la de mayor peso, está teniendo un valor de 693 (ver json), cuando debería tener un valor entre 0 y 1. Esto termina entonces generando un valor de expectativa de vida supremamente grande.

Note que los resultados son los mismos realizando la petición en Postman:

POST http://3.228.160.169/predict

Params Authorization Headers (8) Body Pre-request Script Test

none form-data x-www-form-urlencoded raw binary Gr

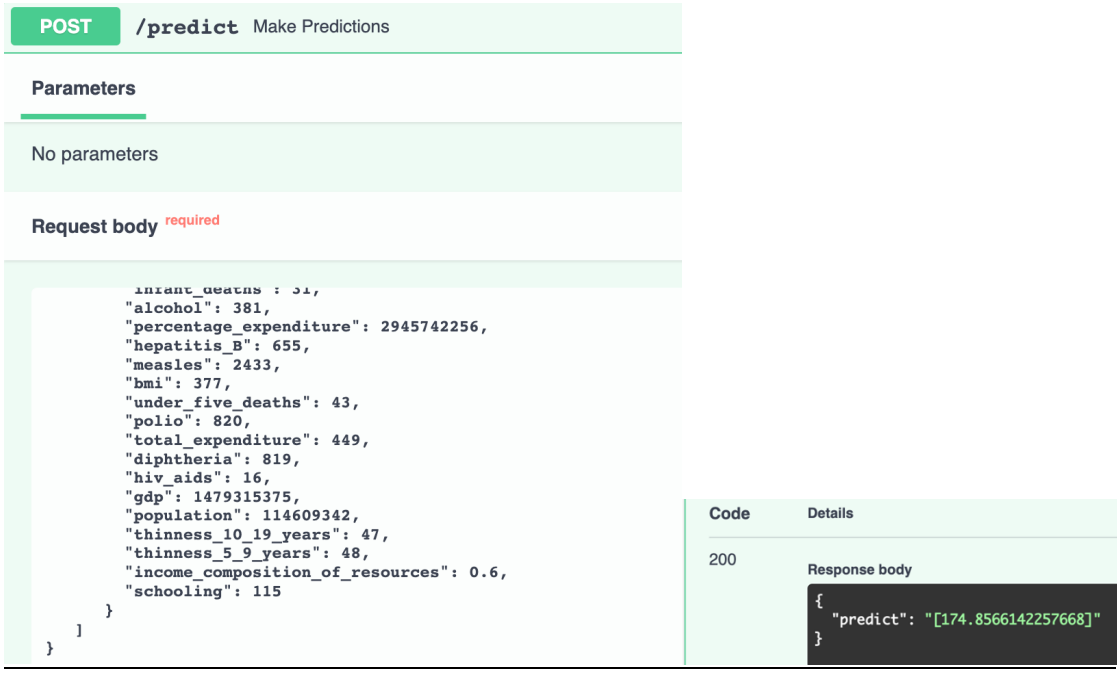

```
{
  "data": [
    {
      "unnamed_0": 500,
      "adult_mortality": 160,
      "infant_deaths": 1100,
      "alcohol": 35.88,
      "percentage_expenditure": 947.210141,
      "hepatitis_B": 9.0,
      "measles": 0,
      "bmi": 68.6,
      "under_five_deaths": 12,
      "polio": 99.0,
      "total_expenditure": 4.11,
      "diphtheria": 99.0,
      "hiv_aids": 0.3,
      "gdp": 2284.549200,
      "population": 600.0,
      "thinness_10_19_years": 100,
      "thinness_5_9_years": 100,
      "income_composition_of_resources": 693,
      "schooling": 14.6
    }
  ]
}
```


Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "predict": "[8680.796497568856]"
}
```

Nombre del archivo: PrediccionIncoherenteMedia2



Decidimos realizar una predicción usando los valores promedio de cada una de las variables del conjunto de entrenamiento. Note que los valores extremos (outliers) pueden llegar a afectar mucho la predicción. Como se sabe, la media es un valor bastante sensible a los outliers: en este caso, es posible que la media de todos los datos no sea un dato coherente para el modelo, pues no solo combina información de diferentes países en distintas franjas de tiempo, sino que además está caracterizado por tener muchos valores fuera de rango.

Con esto, se ve que la media termina representando datos incoherentes para un país. Esto también se ve reflejado en la variable de interés, ya que la esperanza de vida es de 174.85, lo cual no tiene sentido.

Note que los resultados en Postman son iguales:

POST http://3.228.160.169/predict

Params Authorization Headers (8) Body Pre-request Sc

none form-data x-www-form-urlencoded raw bin


```
{
  "data": [
    {
      "unnamed_0": 78,
      "adult_mortality": 1625,
      "infant_deaths": 31,
      "alcohol": 381,
      "percentage_expenditure": 2945742256,
      "hepatitis_B": 655,
      "measles": 2433,
      "bmi": 377,
      "under_five_deaths": 43,
      "polio": 820,
      "total_expenditure": 449,
      "diphtheria": 819,
      "hiv_aids": 16,
      "gdp": 1479315375,
      "population": 114609342,
      "thinness_10_19_years": 47,
      "thinness_5_9_years": 48,
      "income_composition_of_resources": 0.6,
      "schooling": 115
    }
  ]
}
```


Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "predict": "[174.8566142257668]"
}
```

Nombre del archivo: R2Incoherencia1

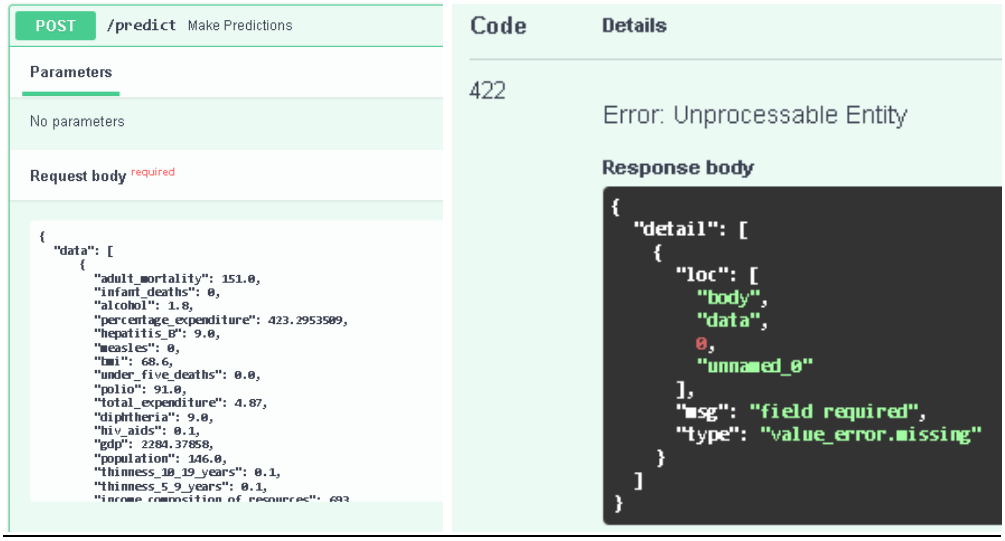
En el último escenario que probamos, quisimos ver qué sucedía con el r^2 si tomábamos datos incoherentes (por ejemplo, los usados para los anteriores dos escenarios). A continuación, el resultado:

Podemos ver que, en este escenario, el r^2 toma un valor negativo que no tiene coherencia, esto se debe a que la predicción hecha y esperada es muy diferente. Este valor no tiene sentido este puede tener un valor máximo de 1 y, para representar el ajuste de los datos al modelo, debe ser positivo.

La razón por la cual esto ocurre es que el valor promedio del income composition of resources para ambos valores enviados es dos órdenes de magnitud mayor que lo esperado (está alrededor de 500, cuando debería ser 1 – como máximo -). Para mejorar este comportamiento, en el pipeline conviene dividir todos aquellos valores mayores a 1 entre 1000. Esto dará un resultado más certero con respecto a la variable de expectativa de vida. Es importante mencionar que esto no se realiza actualmente porque la variable, en entrenamiento, no presentaba ningún problema de rangos. No obstante, ahora, conviene realizar estos ajustes. En particular, de nuevo, es income composition of resources la que más despista al modelo. Al tener una media de 597, cuando realmente debería estar en 1, hace que la predicción sobre la expectativa de vida sea mucho mayor. Esto será implementado en las mejoras al pipeline.

Note que los resultados en postman son los mismos que los obtenidos previamente:

- **Hay fallas**
  Nombre del archivo: FallaUnnamed



Podemos ver que ocurre un error, al realizar una prueba sin la columna "Unnamed". Esto ocurre porque en el modelo teníamos en los datos esta columna, que, en el pipeline eliminamos para hacer el modelo final de predicción. Con lo anterior generamos un problema ya que requerimos de esta columna para que el pipeline y por consiguiente el modelo funcione. Debemos definir entonces, en la clase DataModel.py, un atributo correspondiente a Unnamed_0 (i.e. la columna Unnamed: 0 con la que venía el archivo de entrenamiento), y enviar el JSON con este campo. En realidad, como Unnamed_0 no es un parámetro del modelo, no tiene sentido incluirlo. Por lo tanto, quitar este parámetro sería parte de la ejecución del bono (como se verá más adelante).

Note que los resultados obtenidos en Postman son idénticos:

POST http://3.228.160.169/predict

Params Authorization Headers (8) Body Pre-request Script Test

none form-data x-www-form-urlencoded raw binary Gr


```
{
  "data": [
    {
      "adult_mortality": 151.0,
      "infant_deaths": 0,
      "alcohol": 1.8,
      "percentage_expenditure": 423.2953509,
      "hepatitis_B": 9.0,
      "measles": 0,
      "bmi": 68.6,
      "under_five_deaths": 0.0,
      "polio": 91.0,
      "total_expenditure": 4.87,
      "diphtheria": 9.0,
      "hiv_aids": 0.1,
      "gdp": 2284.37858,
      "population": 146.0,
      "thinness_10_19_years": 0.1,
      "thinness_5_9_years": 0.1,
      "income_composition_of_resources": 693,
      "schooling": 14.6
    },
    {
      "unnamed_0": 22,
      "adult_mortality": 153.0,
```


Body Cookies Headers (5) Test Results

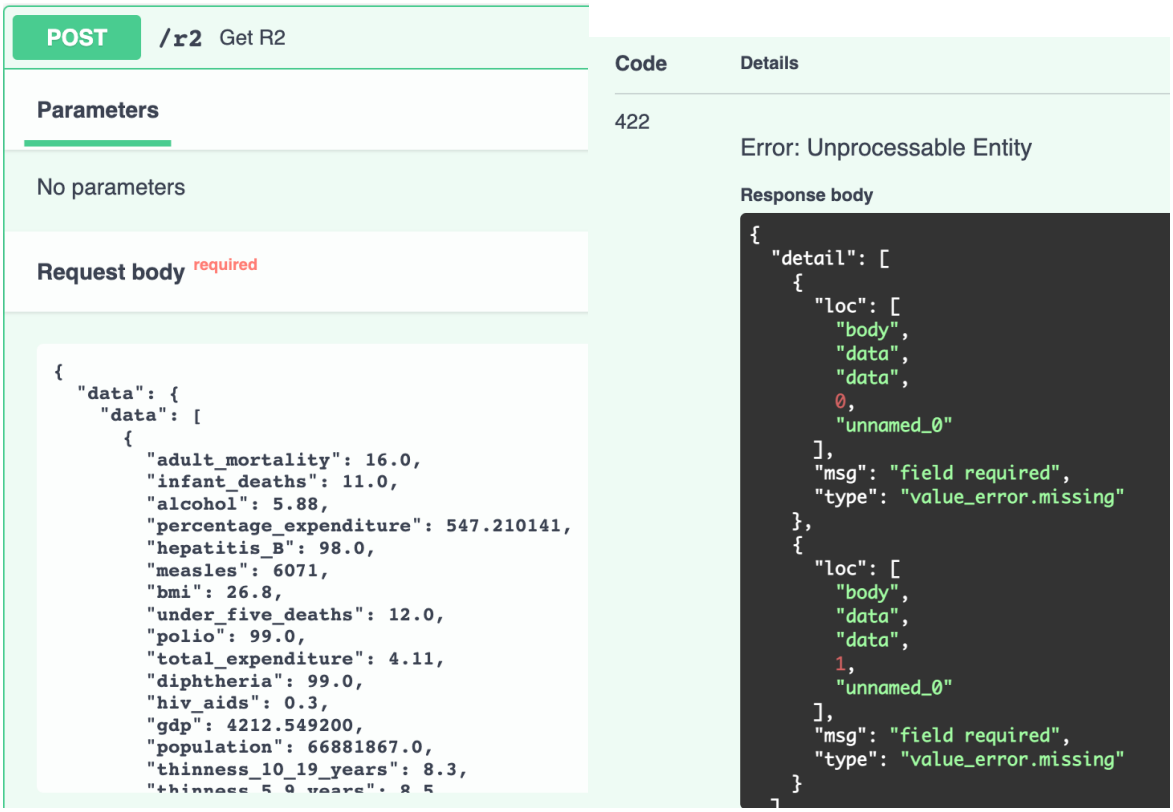
Pretty Raw Preview Visualize JSON


```
{
    "detail": [
        {
            "loc": [
                "body",
                "data",
                0,
                "unnamed_0"
            ],
            "msg": "field required",
            "type": "value_error.missing"
        }
    ]
}
```

Como se puede apreciar, se obtiene un error de tipo: value_error.missing, correspondiente a la carencia del campo unnamed_0 en el primer elemento del json.

Nombre del archivo: FallaUnnamedR2

El mismo error se hace evidente en la petición del r^2, si no se envía el campo con Unnamed: 0, por exactamente las mismas razones que las descritas previamente.



Note que el resultado es exactamente en Postman, como es de esperarse. Es posible notar que el modelo no se ejecuta porque hay un error que está haciendo falta enviar en la petición.

POST http://3.228.160.169/r2

Params Authorization Headers (8) Body Pre-request Scri

none form-data x-www-form-urlencoded raw bina


```
"data": {
  "data": [
    {
      "adult_mortality": 16.0,
      "infant_deaths": 11.0,
      "alcohol": 5.88,
      "percentage_expenditure": 547.210141,
      "hepatitis_B": 98.0,
      "measles": 6071,
      "bmi": 26.8,
      "under_five_deaths": 12.0,
      "polio": 99.0,
      "total_expenditure": 4.11,
      "diphtheria": 99.0,
      "hiv_aids": 0.3,
      "gdp": 4212.549200,
      "population": 66881867.0,
      "thinness_10_19_years": 8.3,
      "thinness_5_9_years": 8.5,
      "income_composition_of_resources": 0.706,
      "schooling": 13
    },
    {
      "adult_mortality": 393.0,
      "infant_deaths": 2,
```


Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
            "body",
            "data",
            "data",
            0,
            "unnamed_0"
        ],
        "msg": "field required",
        "type": "value_error.missing"
    },
```

Estrategia por desarrollar sobre el software para mitigar incoherencias en el resultado y fallas en el sistema:

Una posible estrategia es mejorar el pipeline, en esta forma podríamos tener en cuenta la posibilidad de que no sea agregado la columna "Unnamed: 0", por lo que, el modelo debería correr exitosamente con o sin esta columna. Creemos que podría mejorarse solamente usando un try-catch, o un condicional, al momento de eliminar las columnas.

Por otro lado, podríamos dar un mejor manejo a los valores nulos, nosotros los tratamos con la media, pero se podría probar alternativas como la eliminación de estos datos o la imputación con otro valor significativo que no sea la media. En este caso, proponemos dividir todos aquellos valores que sean mayores que 1, entre 1000, para evitar este error.

El intento de ambas mejoras en la implementación puede encontrarlo en el repositorio https://github.com/sofiaalvarezlopez/BI-Lab4-Mejoras .

Conclusiones:

En este laboratorio, pudo implementarse satisfactoriamente un pipeline con transformaciones personalizadas, que fue desplegada tanto local como remotamente (en un servidor de AWS).

Hay que tener en cuenta que los datos solo funcionan para países vecinos a los Alpes y que sean de años cercanos, para que tengan estadísticas similares, ya que, debido a factores macro y micro, los índices o variables de estos países pudieron haber cambiado drásticamente, por lo que el modelo dejaría de predecir adecuadamente la expectativa de vida. Es por esta razón que los datos incoherentes (como, por ejemplo, el valor de income composition of resources) afectan tanto (y negativamente) la predicción del modelo.

Bono:

1. **Construir transformaciones personalizadas e incluirlas en el pipeline y garantizar que el proceso completo es correcto.**

El pipeline que usamos para la realización de este laboratorio tiene transformaciones personalizadas, estas clases se pueden ver en el documento clases.py, en la carpeta Notebook.

Es importante notar que, al comienzo, las transformaciones personalizadas se implementaban directamente en el notebook. No obstante, por errores de serialización, estas fueron puestas en el archivo clases.py.

2. Desplegar la API en un servidor gratuito como Heroku para que pueda prestar servicio a cualquiera haciendo uso de una URL.

Desplegamos la API en AWS, en este logramos que la IP sea estática. Puede acceder en : http://3.228.160.169/docs

3. Implementar la estrategia para mitigación de errores identificados en los escenarios y documentados por ustedes en el documento de entrega.

Remítase al repositorio: https://github.com/sofiaalvarezlopez/BI-Lab4-Mejoras . Las mejoras se encuentran en los archivos clases.py de notebook y de la carpeta raíz, respectivamente.

Para el primer error (el de la columna: "Unnamed: 0"), fue solamente necesario implementar un try-catch tal que, si no se encontraba dicha columna en los datos de test, no se intentara eliminar.

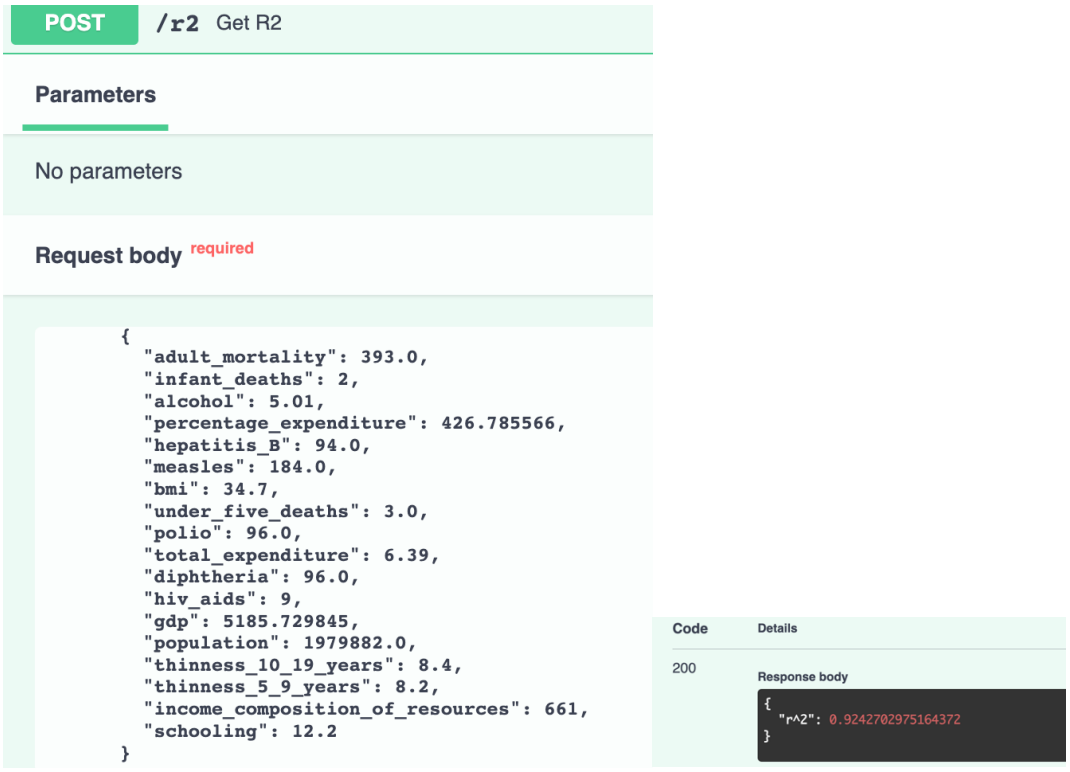
```
def transform(self,X,y=None):
    # Optimizacion 1 implementada: Solamente eliminar Unnamed: 0 cuando exista.
    if not 'Unnamed: 0' in self.columns:
        return X.drop(self.columns,axis=1)
    else:
        return X
```

Para la segunda, como esto el proceso de tratar la variable de income composition of resources (que además es la de mayor influencia en el modelo, de acuerdo con el análisis realizado), se dividieron entre 1000 todos los valores que fuesen mayores que 1:

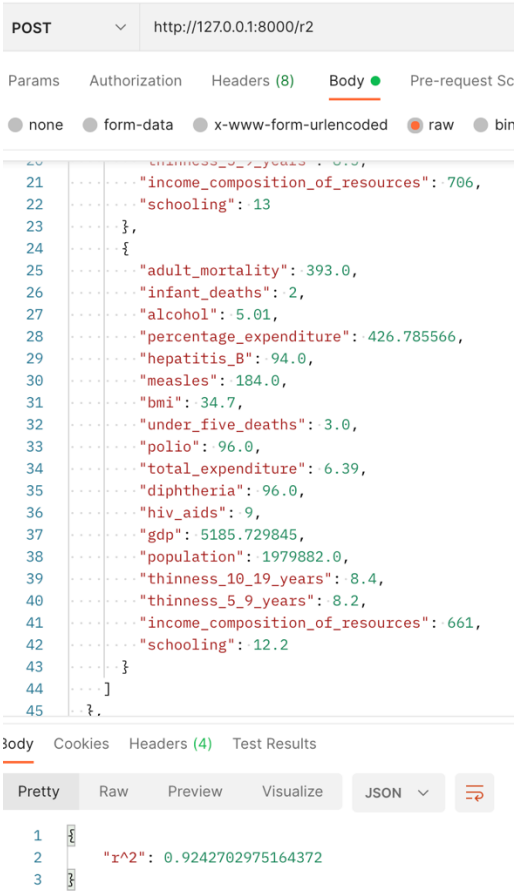
```
# Optimizacion 2 implementada: Sobre los datos de train no se hace nada sobre este atributo porque todos los valores estan entre 0 y 1.
# Remplazamos por el valor dividido mil si es un valor mayor a 1.
X["Income composition of resources"] = X["Income composition of resources"].apply( lambda x : x/1000 if x>1 else x )
X[self.columns] = X[self.columns].apply(lambda x:x.replace({np.nan: (np.sum(x)/len(x))}))
print(X)
```

Note que, ahora, al correr localmente el proyecto y ejecutar los archivos con fallas y resultados incoherentes (Estos últimos sin unnamed), arrojan resultados coherentes.

Por ejemplo, con FallaUnnamedR2.json:



En Postman:



Note que este caso es interesante puesto que no solo no hay unnamed, sino que además el valor de income composition of resources está fuera del rango de la variable original (i.e., usando la optimización # 2). Lo mismo sucede en el caso de FallaUnnamed.json.

En el caso de PrediccionIncoherente1.json, eliminando el campo de unnamed_0 (Esto último no es necesario, pero se eliminó por motivos ilustrativos):

En Postman:

Note que, ahora, la predicción sí es coherente con el rango de valores que el modelo arrojaba en entrenamiento, así como con los valores de la vida real para la expectativa de vida de un país.