Laboratorio 4
Inteligencia de negocios
Sergio Andrés Molano Valencia - 201814278
María Paula Téllez - 201823028
Santiago Hernández - 201922432

Construcción de pipeline

Para la construcción de pipelines, se utilizaron las funciones ya creadas en sklearn debido a las complicaciones que trae el crear transformadores propios al momento de serializarlos y con el manejo de los pipelines. Para la transformación de los datos, se realizó una selección de únicamente las columnas 'Income composition of resources','Schooling','Adult Mortality', adicionalmente se realizó una estandarización de los datos ya que es necesario que todos los datos estén dentro de un rango considerable para tener una buena regresión lineal, se rellenaron todos los nulos de las variables a utilizar ya que consideramos que no era buena estrategia dejarlos en nulo o eliminarlos.

Para poder realizar cada una de estas transformaciones, se pusieron como pasos a seguir antes de crear la regresión lineal, al momento de crear el Pipeline:

```
# Declara el pipeline
pipeline = Pipeline(steps=[
    ('data_preprocessing', ColumnTransformer([
        ("selector", "passthrough", selected_cols)
    ], remainder="drop")),
    ('data_scaling', StandardScaler()),
    ('num_imputer', SimpleImputer(missing_values=np.nan, strategy='mean')),
    ('model', LinearRegression()),
    ])
```

Finalmente se entrena el pipeline y se guarda para su uso posterior.

Para la realización del API, se utilizó fastapi y pydantic para la creación de los modelos y el manejo de los JSON, en este caso todo se realizó en el archivo DataModel.py. Se crearon 4 modelos, 1 : DataModelapp, tiene todas las variables menos la objetivo, 2: DataPrediccion que tiene todas las variables incluyendo la objetivo, finalmente el modelo 3 y 4 son simplemente listas que contienen varios datos del tipo 1 y 2.

Respecto a los enlaces, se tienen 3.

El primero es /Data/predict , este enlace recibe únicamente una filade todas las variables y realiza la predicción en base a sus valores y haciendo uso del modelo (pipeline) anteriormente guardado.

El segundo enlace es /Data/predict/list este enlace recibe una lista de peticiones del tipo del primer enlace, básicamente realiza lo mismo que el primer enlace, pero este enlace es capaz de recibir varios predictores X y le calcula una predicción a cada uno.

Finalmente, el tercer enlace es /Data/predict/Rcuadrado/list recibe una lista de predictores X y de valores esperados de Y y lo que hace es separar los valores Y y realiza una predicción sobre los predictores X, para finalmente realizar una comparación frente a las predicciones realizadas y los

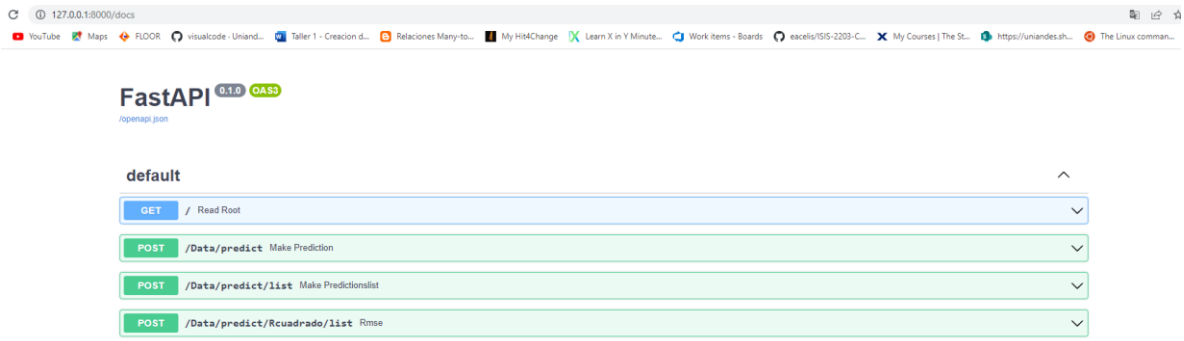
valores esperados de Y, calculando así una métrica de desempeño llamada Rcuadrado. Para este apartado no se realizó el recibir únicamente un registro ya que le Rcuadrado es una métrica que necesita al menos 2 registros para funcionar de manera correcta.

Escenarios de prueba

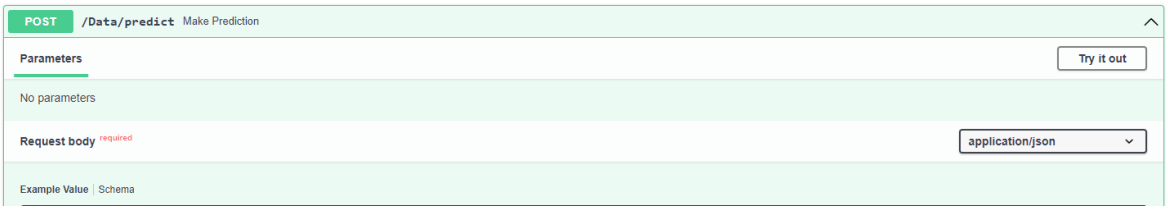
Una vez ubicados en

http://127.0.0.1:8000/docs

podremos ver los 3 links:



Dando click en el primero y en el botón Try it out



Podemos ver un ejemplo de un json, para las pruebas usamos el siguiente:

{

"Adult_mortality": 263.0,

"infant_deaths": 62,

"alcohol": 0.01,

"percentage_expenditure": 71.279624,

"hepatitis_B": 65.0,

"measles": 1154,

"bmi": 19.1,

"under_five_deaths": 83,

"polio": 6.0,

"total_expenditure": 8.16,

"diphtheria": 65.0,

"hiv_aids": 0.1,

"gdp": 584.259210,

"population": 33736494.0,

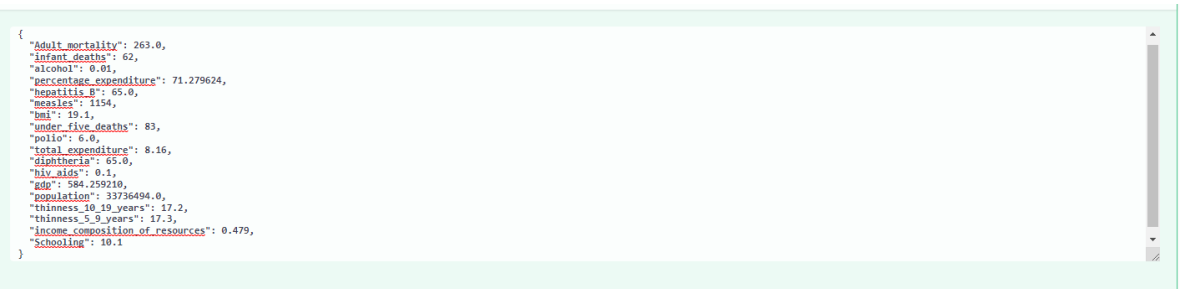
"thinness_10_19_years": 17.2,

"thinness_5_9_years": 17.3,

"income_composition_of_resources": 0.479,

"Schooling": 10.1

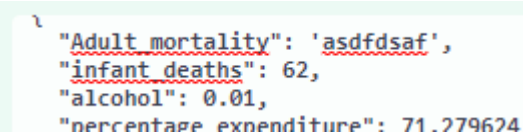
}



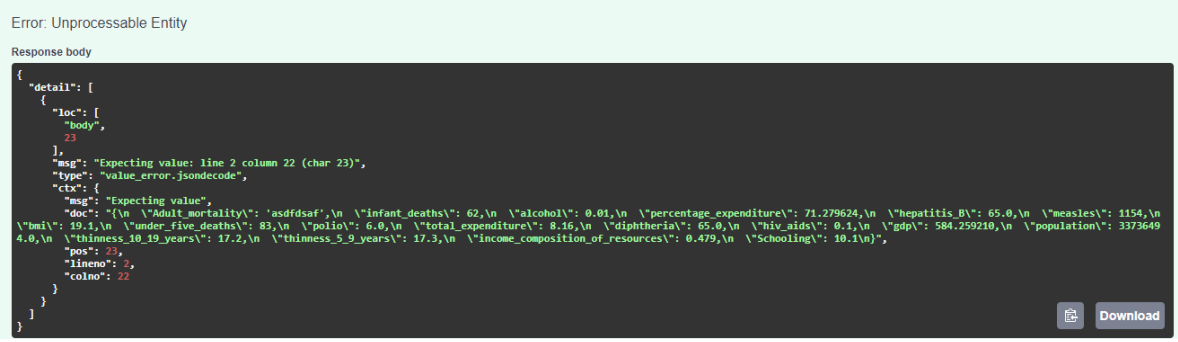
se corre y se obtiene el siguiente resultado:

Viendo que el modelo lo dio un valor de 61.7 a la variable objetivo, en este caso, la variable tiene un valor de 65, por lo que el modelo estuvo levemente desfazado, pero da una aproximación precisa.

Un ejemplo donde falla es si por ejemplo, se cambia algún valor por un string:



Y se obtiene el siguiente error:

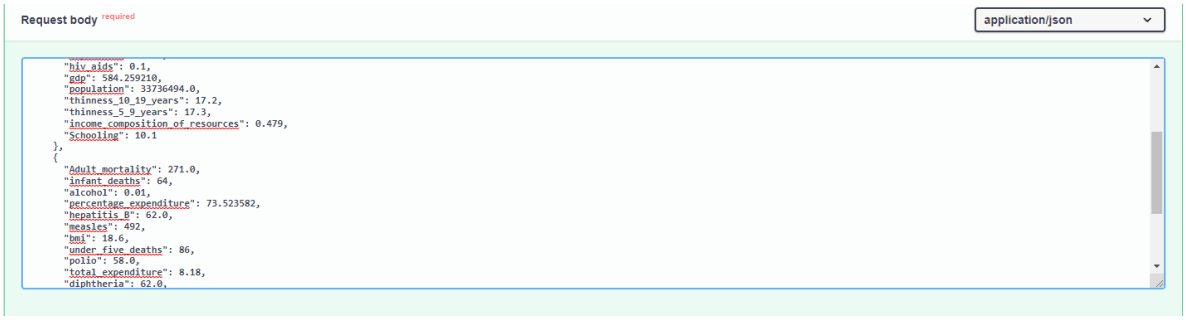


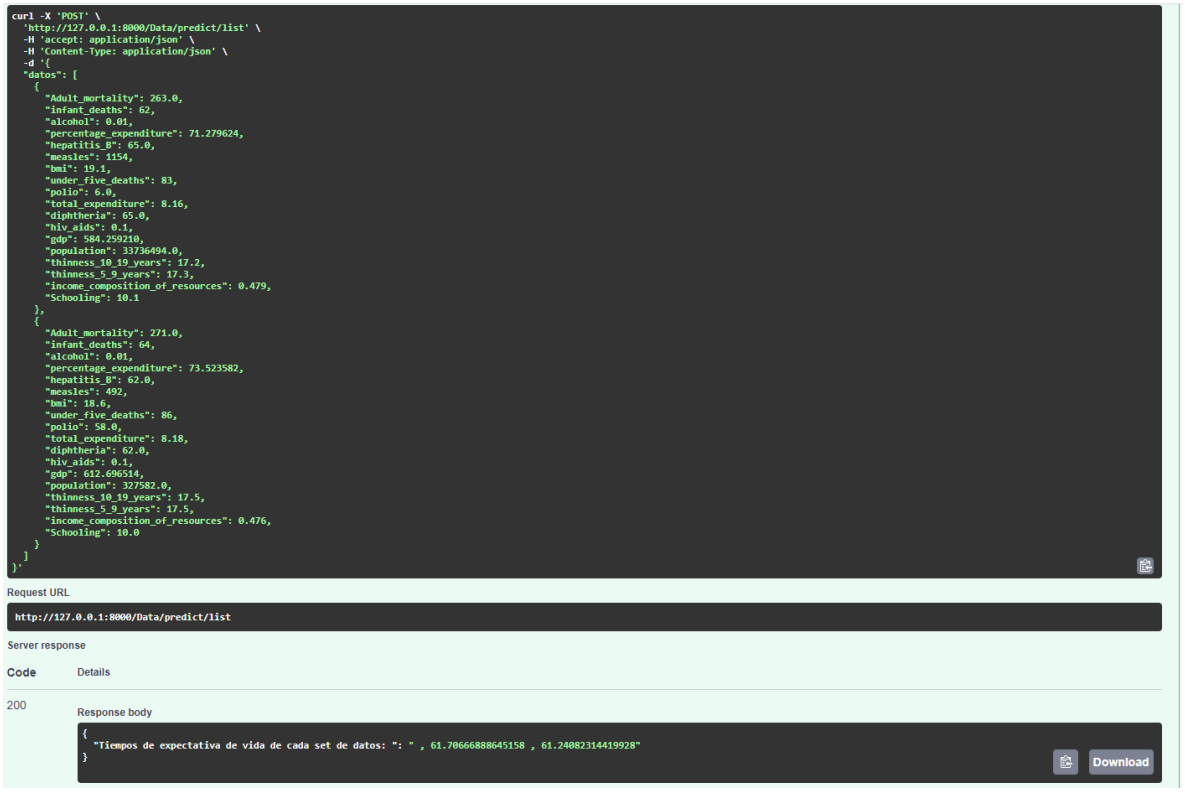
Segundo enlace:

Para este enlace se realiza el mismo procedimiento, pero en el cuerpo, se debe pasar una lista de registros, json de ejemplo:


```
{
  "datos": [
    {
      "Adult_mortality": 263.0,
      "infant_deaths": 62,
      "alcohol": 0.01,
      "percentage_expenditure": 71.279624,
      "hepatitis_B": 65.0,
      "measles": 1154,
      "bmi": 19.1,
      "under_five_deaths": 83,
      "polio": 6.0,
      "total_expenditure": 8.16,
      "diphtheria": 65.0,
      "hiv_aids": 0.1,
      "gdp": 584.259210,
      "population": 33736494.0,
      "thinness_10_19_years": 17.2,
      "thinness_5_9_years": 17.3,
      "income_composition_of_resources": 0.479,
      "Schooling": 10.1
    },
    {
      "Adult_mortality": 271.0,
      "infant_deaths": 64,
      "alcohol": 0.01,
      "percentage_expenditure": 73.523582,
      "hepatitis_B": 62.0,
      "measles": 492,
      "bmi": 18.6,
      "under_five_deaths": 86,
      "polio": 58.0,
```

```
    "total_expenditure": 8.18,
    "diphtheria": 62.0,
    "hiv_aids": 0.1,
    "gdp": 612.696514,
    "population": 327582.0,
    "thinness_10_19_years": 17.5,
    "thinness_5_9_years": 17.5,
    "income_composition_of_resources": 0.476,
    "Schooling": 10.0
  }
 ]
}
```

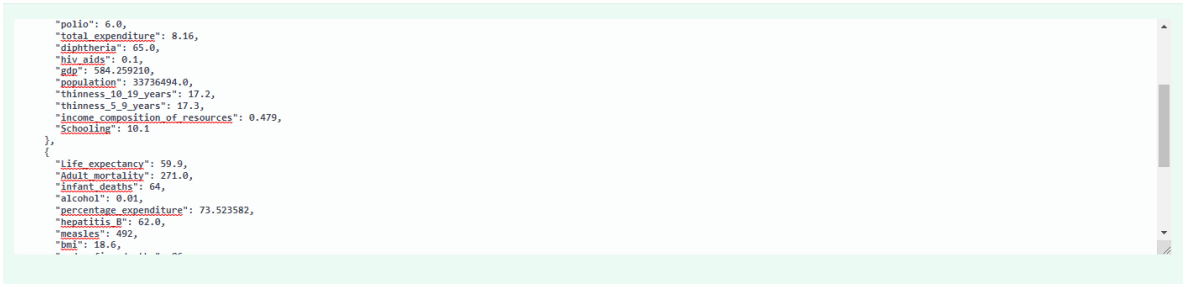


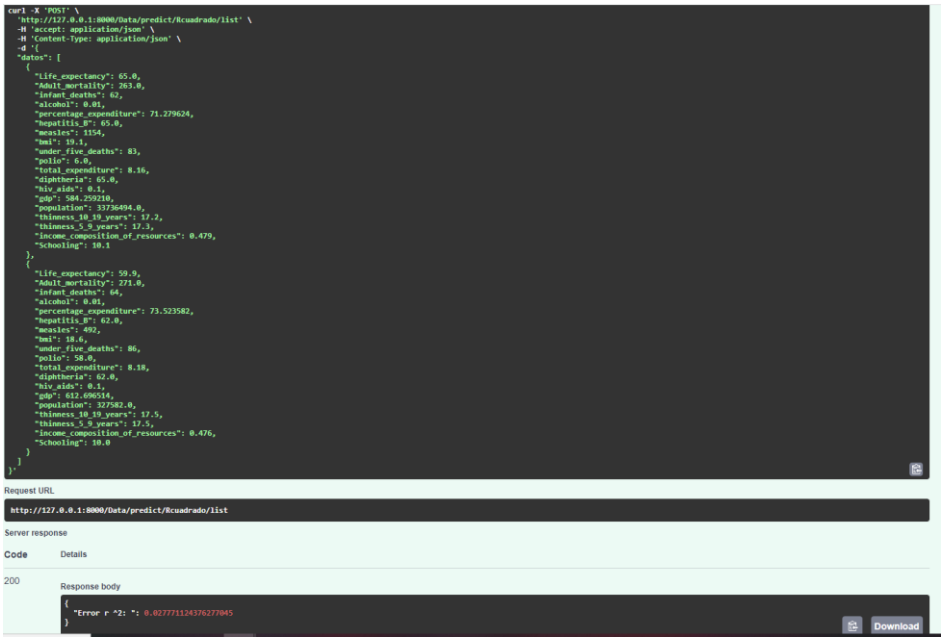
Como se enviaron 2 registros, se obtienen 2 predicciones en el cuerpo de la respuesta. Estos resultados tienen sentido ya que el valor esperado del segundo registro es de 59.9 y el modelo dio un valor de 61.2, bastante cercano al valor esperado.

Finalmente, para el tercer enlace, se añade la variable objetivo:


```
{
  "datos": [
    {
      "Life_expectancy": 65.0,
      "Adult_mortality": 263.0,
      "infant_deaths": 62,
      "alcohol": 0.01,
      "percentage_expenditure": 71.279624,
      "hepatitis_B": 65.0,
      "measles": 1154,
      "bmi": 19.1,
      "under_five_deaths": 83,
      "polio": 6.0,
      "total_expenditure": 8.16,
      "diphtheria": 65.0,
      "hiv_aids": 0.1,
      "gdp": 584.259210,
      "population": 33736494.0,
      "thinness_10_19_years": 17.2,
      "thinness_5_9_years": 17.3,
      "income_composition_of_resources": 0.479,
      "Schooling": 10.1
    },
    {
      "Life_expectancy": 59.9,
      "Adult_mortality": 271.0,
      "infant_deaths": 64,
      "alcohol": 0.01,
      "percentage_expenditure": 73.523582,
      "hepatitis_B": 62.0,
      "measles": 492,
```

```
    "bmi": 18.6,
    "under_five_deaths": 86,
    "polio": 58.0,
    "total_expenditure": 8.18,
    "diphtheria": 62.0,
    "hiv_aids": 0.1,
    "gdp": 612.696514,
    "population": 327582.0,
    "thinness_10_19_years": 17.5,
    "thinness_5_9_years": 17.5,
    "income_composition_of_resources": 0.476,
    "Schooling": 10.0
  }
 ]
}
```

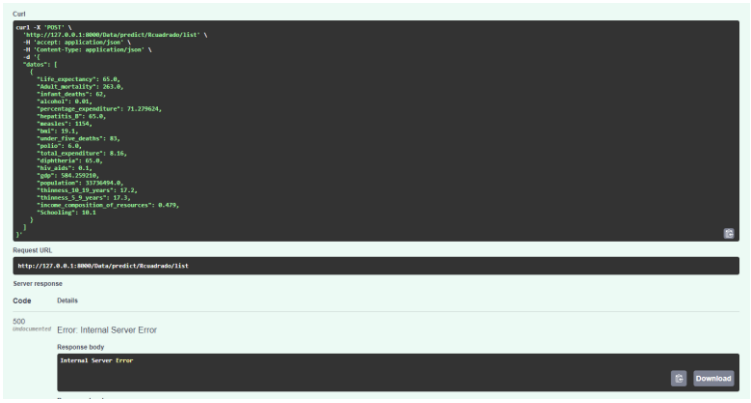


Como se puede ver en la respuesta de la petición, se registra el valor de la métrica r^2, se logra obtener un valor de 0.03, lo cual es bastante regular, pero consideramos que es debido a los pocos registros que se utilizan para su cálculo, únicamente 2, pero al usar un número considerable esta métrica tenderá a subir.

Ejemplo de fallo:


```
{
  "datos": [
    {
      "Life_expectancy": 65.0,
      "Adult_mortality": 263.0,
      "infant_deaths": 62,
      "alcohol": 0.01,
      "percentage_expenditure": 71.279624,
      "hepatitis_B": 65.0,
      "measles": 1154,
      "bmi": 19.1,
      "under_five_deaths": 83,
      "polio": 6.0,
      "total_expenditure": 8.16,
      "diphtheria": 65.0,
      "hiv_aids": 0.1,
      "gdp": 584.259210,
      "population": 33736494.0,
      "thinness_10_19_years": 17.2,
      "thinness_5_9_years": 17.3,
      "income_composition_of_resources": 0.479,
      "Schooling": 10.1
    }
  ]
}
```


Debido a que el R^2 es una métrica que necesita al menos 2 registros para su funcionamiento, si en el JSON, solo se envía un registro, ocurre un error:



Con esto se pueden ver 5 ejemplos del correcto e incorrecto funcionamiento del API.

IMPORTANTE

Debido a que en formato Word puede que el JSON no quede bien, dentro de la carpeta, LAB4API, hay un archivo llamado ejemplosjson.txt en donde se puede copiar y pegar directamente cada uno de los JSON usados, están separados por '---' para cada enlace del API.

Para mitigar los fallos creemos que es necesario, realizar una buena transformación de los datos, el preprocesamiento de los datos es algo esencial para tener un buen rendimiento de los modelos de regresión, por eso creemos que una buena estrategia es dejar toda esa parte muy bien hecha desde anaconda para que al momento de exportar el modelo con joblib, quede de la mejor forma para ser usada por el api.