

INDUSTRIALIZACIÓN: API DE FLASK

Eva Arribas

OBJETIVO

Programar una API que permita obtener los datos de mi modelo sobre Attrition además de las predicciones.



PASOS PREVIOS



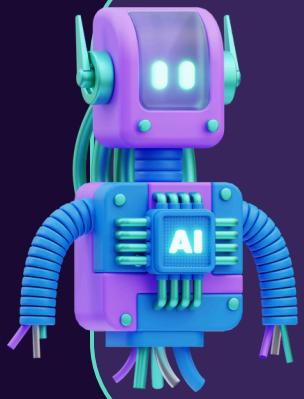
Carga de datos del modelo

Cargamos los datos de nuestro modelo sobre el Attrition.



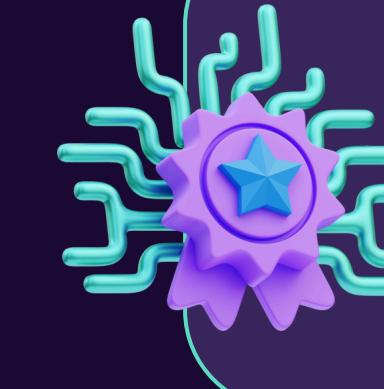
Preprocesamiento de los datos

Unimos los datasets con un merge, eliminamos los valores nulos



Preparación para el modelo

Eliminamos la columna con los datos a predecir y dividimos entre train y test (test_size=0.2, random_state = 42), hacemos el fit y guardamos el modelo en un pickle.

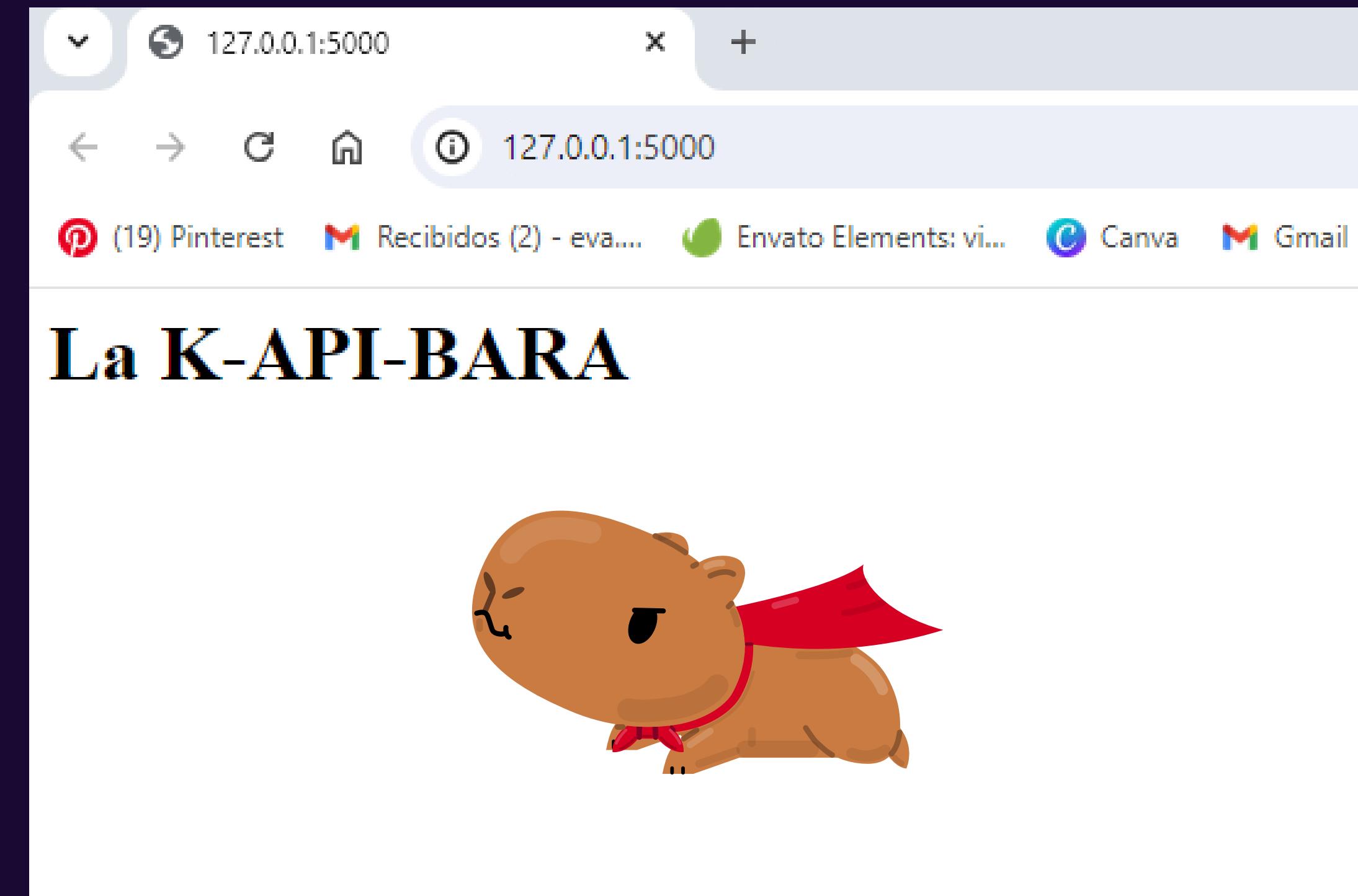
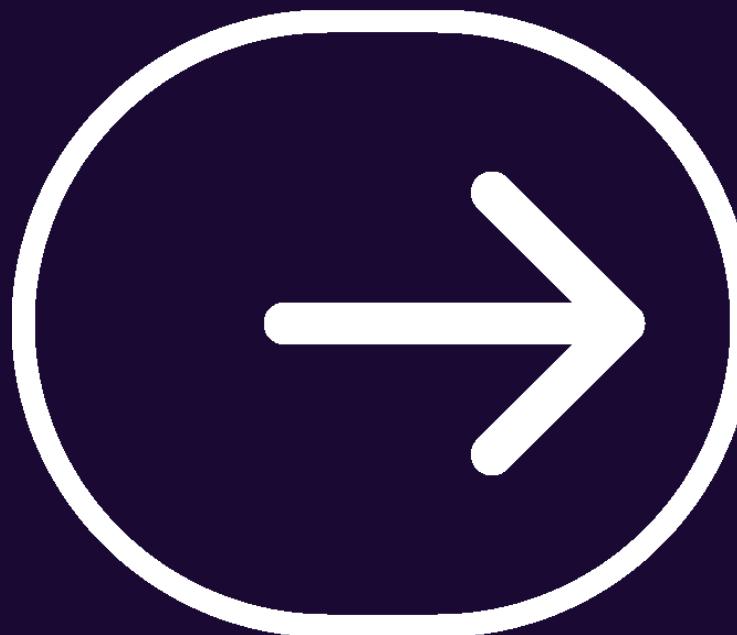


Datos train y test

Guardamos los datos de entrenamiento y de prueba en un json.

API DEFINIDA

```
app.run(port=5000)  
  2.7s  
  
* Serving Flask app '__main__'  
* Debug mode: off  
WARNING: This is a development server.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit
```



ENDPOINT 1: TODOS LOS DATOS



```
@app.route('/data', methods=['GET'])
def get_data():
    return data.to_json(orient='records'), 200
```



A screenshot of a web browser window. The address bar at the top shows the URL "127.0.0.1:5000/data". This URL is circled in orange. Below the address bar, there are several browser navigation icons: back, forward, search, and others. To the right of the address bar, there are links to various social media and services: Pinterest (19), Recibidos (2) - eva..., Envato Elements: vi..., Canva, and Gmail. The main content area of the browser displays a large amount of JSON data. The data consists of an array of objects, each representing a record with fields like Age, DistanceFromHome, Education, EmployeeCount, EmployeeDevelopment, Department_Sales, EducationField_LifeSciences, EducationField_Marketing, EducationField_Medical, JobRole_LaboratoryTechnician, JobRole_Manager, and Scientist. The JSON is very long and repetitive, showing multiple entries with similar values.

ENDPOINT 2: DATOS DEL TRAIN

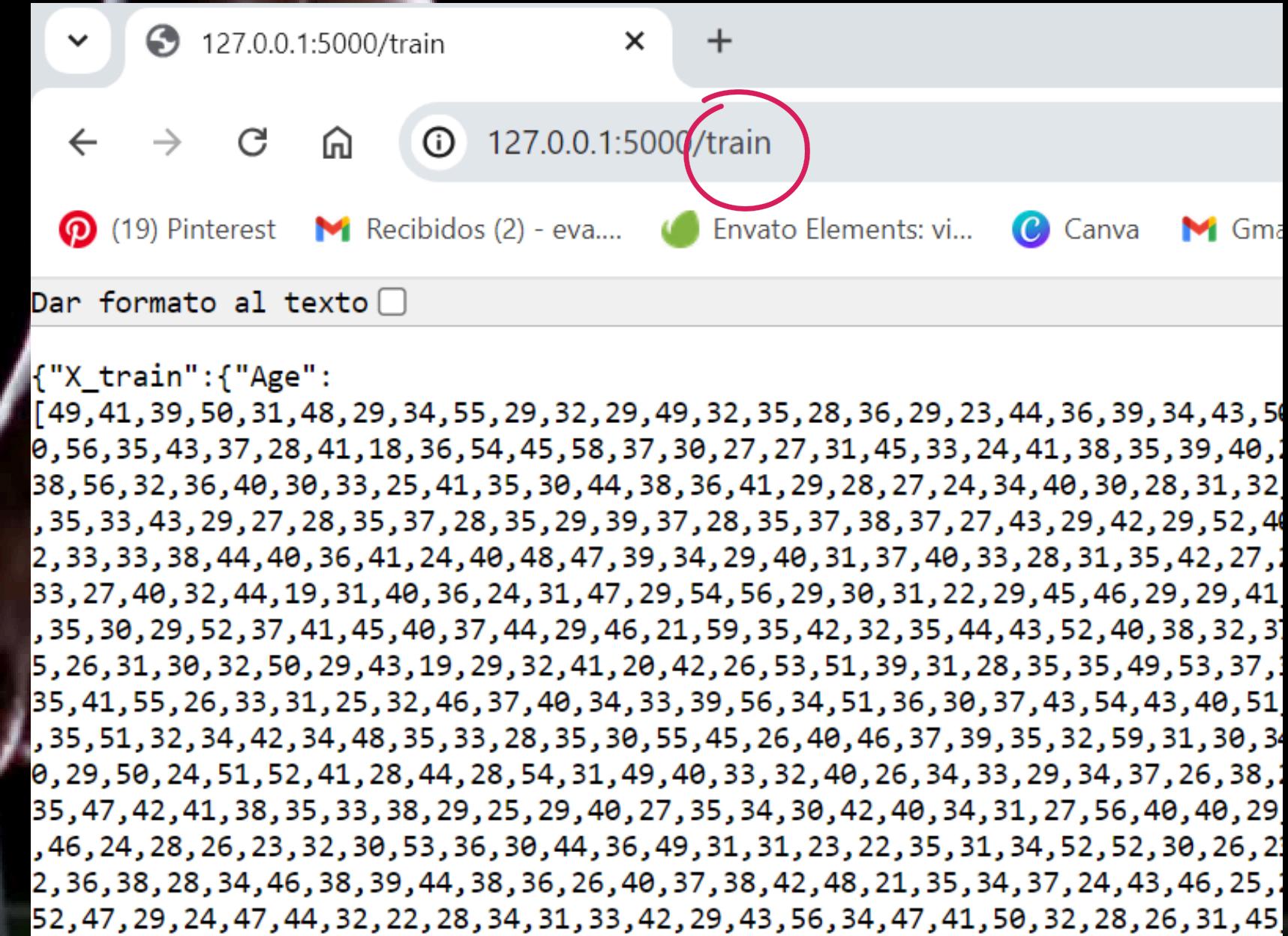


```
# Endpoint para obtener los datos de entrenamiento

def load_train_data():
    global train_data
    with open('train_data.json', 'r') as f:
        train_data = json.load(f)
    ✓ 0.0s

# Endpoint para obtener los datos de entrenamiento

@app.route('/train', methods=['GET'])
def get_train_data():
    global train_data
    if train_data is None:
        return jsonify({"message": "Datos de entrenamiento no disponibles"}), 400
    return jsonify(train_data), 200
    ✓ 0.0s
```



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/train`. A red circle highlights the URL in the address bar. The browser interface includes a back button, forward button, refresh button, and a search bar above the address bar. Below the address bar, there are several browser tabs and icons for Pinterest, Gmail, Envato Elements, Canva, and Google Drive. The main content area of the browser displays a JSON response with the key `"X_train"` and its value being a large list of integers representing ages.

```
{"X_train": {"Age": [49, 41, 39, 50, 31, 48, 29, 34, 55, 29, 32, 29, 49, 32, 35, 28, 36, 29, 23, 44, 36, 39, 34, 43, 50, 0, 56, 35, 43, 37, 28, 41, 18, 36, 54, 45, 58, 37, 30, 27, 27, 31, 45, 33, 24, 41, 38, 35, 39, 40, 1, 38, 56, 32, 36, 40, 30, 33, 25, 41, 35, 30, 44, 38, 36, 41, 29, 28, 27, 24, 34, 40, 30, 28, 31, 32, 2, 35, 33, 43, 29, 27, 28, 35, 37, 28, 35, 29, 39, 37, 28, 35, 37, 38, 37, 27, 43, 29, 42, 29, 52, 40, 2, 33, 33, 38, 44, 40, 36, 41, 24, 40, 48, 47, 39, 34, 29, 40, 31, 37, 40, 33, 28, 31, 35, 42, 27, 1, 33, 27, 40, 32, 44, 19, 31, 40, 36, 24, 31, 47, 29, 54, 56, 29, 30, 31, 22, 29, 45, 46, 29, 29, 41, 1, 35, 30, 29, 52, 37, 41, 45, 40, 37, 44, 29, 46, 21, 59, 35, 42, 32, 35, 44, 43, 52, 40, 38, 32, 31, 5, 26, 31, 30, 32, 50, 29, 43, 19, 29, 32, 41, 20, 42, 26, 53, 51, 39, 31, 28, 35, 35, 49, 53, 37, 1, 35, 41, 55, 26, 33, 31, 25, 32, 46, 37, 40, 34, 33, 39, 56, 34, 51, 36, 30, 37, 43, 54, 43, 40, 51, 1, 35, 51, 32, 34, 42, 34, 48, 35, 33, 28, 35, 30, 55, 45, 26, 40, 46, 37, 39, 35, 32, 59, 31, 30, 34, 0, 29, 50, 24, 51, 52, 41, 28, 44, 28, 54, 31, 49, 40, 33, 32, 40, 26, 34, 33, 29, 34, 37, 26, 38, 1, 35, 47, 42, 41, 38, 35, 33, 38, 29, 25, 29, 40, 27, 35, 34, 30, 42, 40, 34, 31, 27, 56, 40, 40, 29, 1, 46, 24, 28, 26, 23, 32, 30, 53, 36, 30, 44, 36, 49, 31, 31, 23, 22, 35, 31, 34, 52, 52, 30, 26, 21, 2, 36, 38, 28, 34, 46, 38, 39, 44, 38, 36, 26, 40, 37, 38, 42, 48, 21, 35, 34, 37, 24, 43, 46, 25, 1, 52, 47, 29, 24, 47, 44, 32, 22, 28, 34, 31, 33, 42, 29, 43, 56, 34, 47, 41, 50, 32, 28, 26, 31, 45]}
```

ENDPOINT 3: DATOS DEL TEST



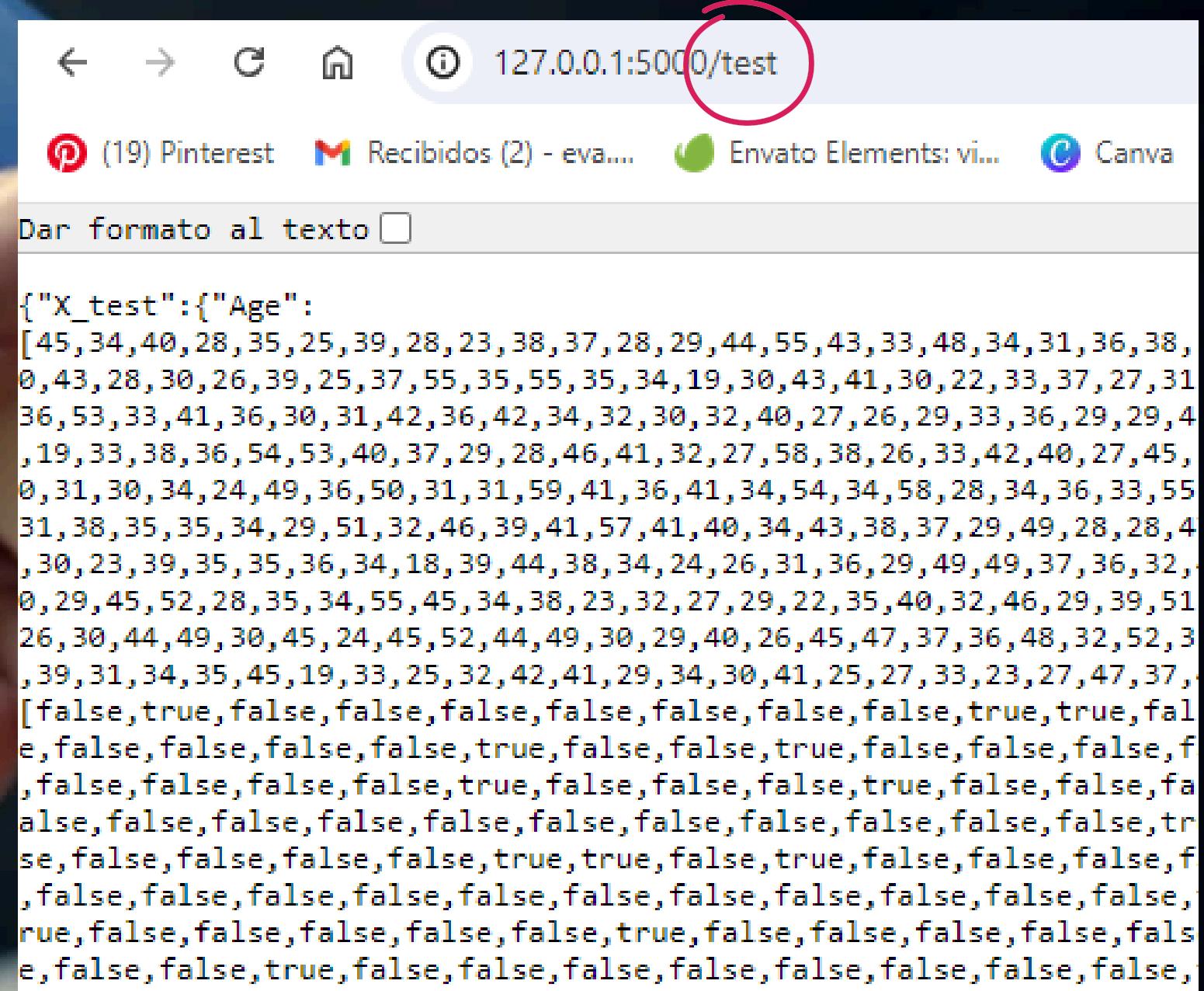
```
def load_test_data():
    global test_data
    with open('test_data.json', 'r') as f:
        test_data = json.load(f)

✓ 0.0s

@app.route('/test', methods=[ 'GET' ])
def get_test_data():
    global test_data
    if test_data is None:
        return jsonify({ "message": "Datos del test no disponibles" }), 400

    return jsonify(test_data), 200

✓ 0.0s
```



ENDPOINT 4: OBSERVACIÓN N DE MI DATASET



```
@app.route('/observacion', methods=['GET'])
def obs():
    return jsonify(data.iloc[34].to_dict())
```

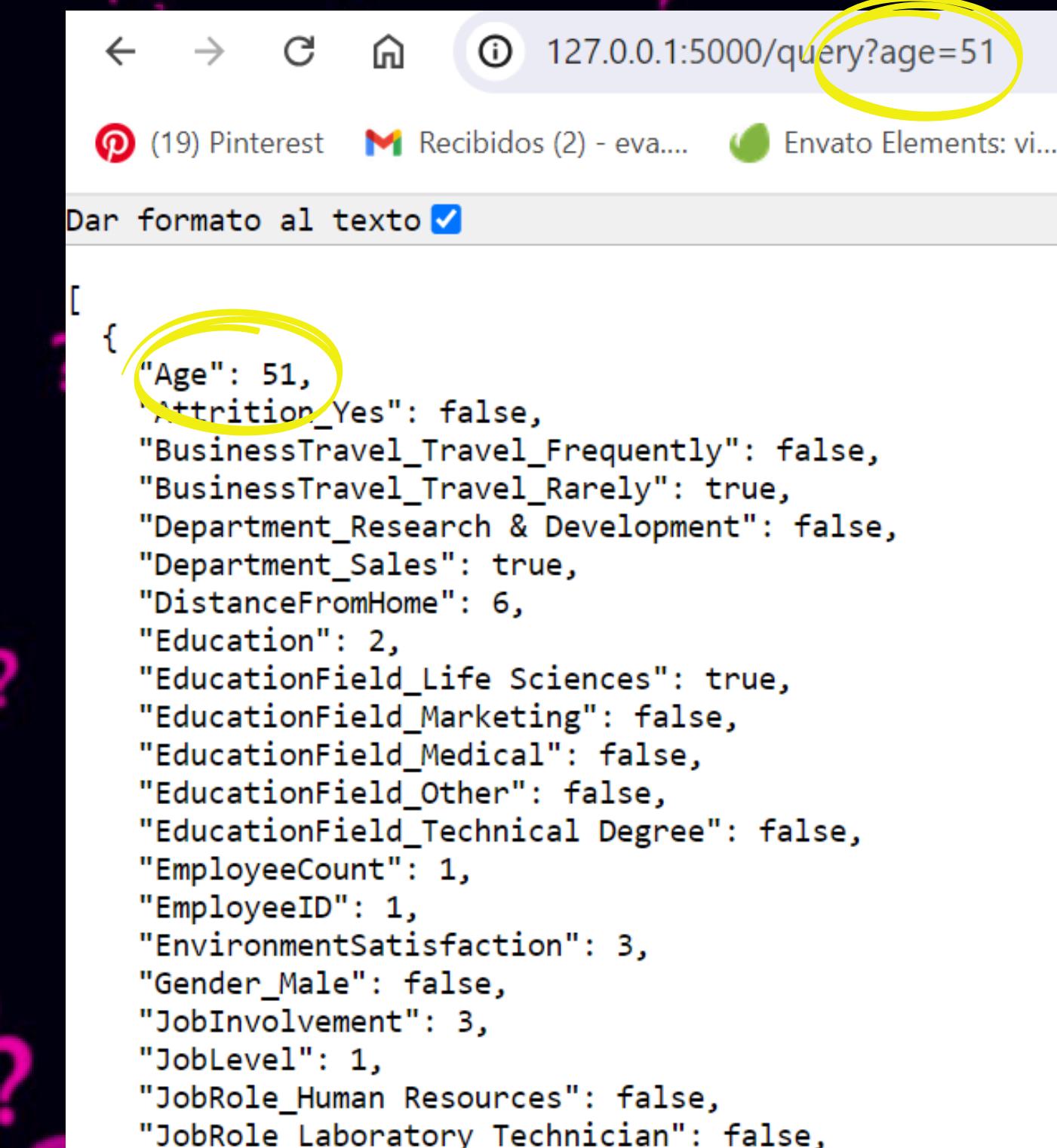
A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/observacion`. Below the address bar, there are social sharing icons for Pinterest, Recibidos, Envato Elements, Canva, and Gmail. A text input field labeled "Dar formato al texto" is present. The main content area displays a JSON object:

```
{"Age":37,"Attrition_Yes":false,"BusinessTravel_Travel_Frequently":true,"BusinessSciences":false,"EducationField_Marketing":true,"EducationField_Medical":false,"Degree":false,"EmployeeCount":1,"EmployeeID":37,"EnvironmentSatisfaction":2.6,"Gender":false,"JobRole_Director":false,"JobRole_Research Director":false,"JobRole_Research Scientist":false,"JobRole_Sales Representative":false,"JobSatisfaction":4.0,"MaritalStatus_Married":true,"ManagersLastYear":3,"WorkLifeBalance":2.0,"YearsAtCompany":4,"YearsSinceLastPromotion":10,"YearsWithCurrentManager":3}
```

ENDPOINT 5: QUERY

```
@app.route('/query', methods=['GET'])
def age():
    if 'age' in request.args:
        age = int(request.args['age'])
    else: "No se encuentra una edad válida"

    edad = data[data['Age']==age]
    edad_dict = edad.to_dict(orient= "records")
    return jsonify(edad_dict)
```

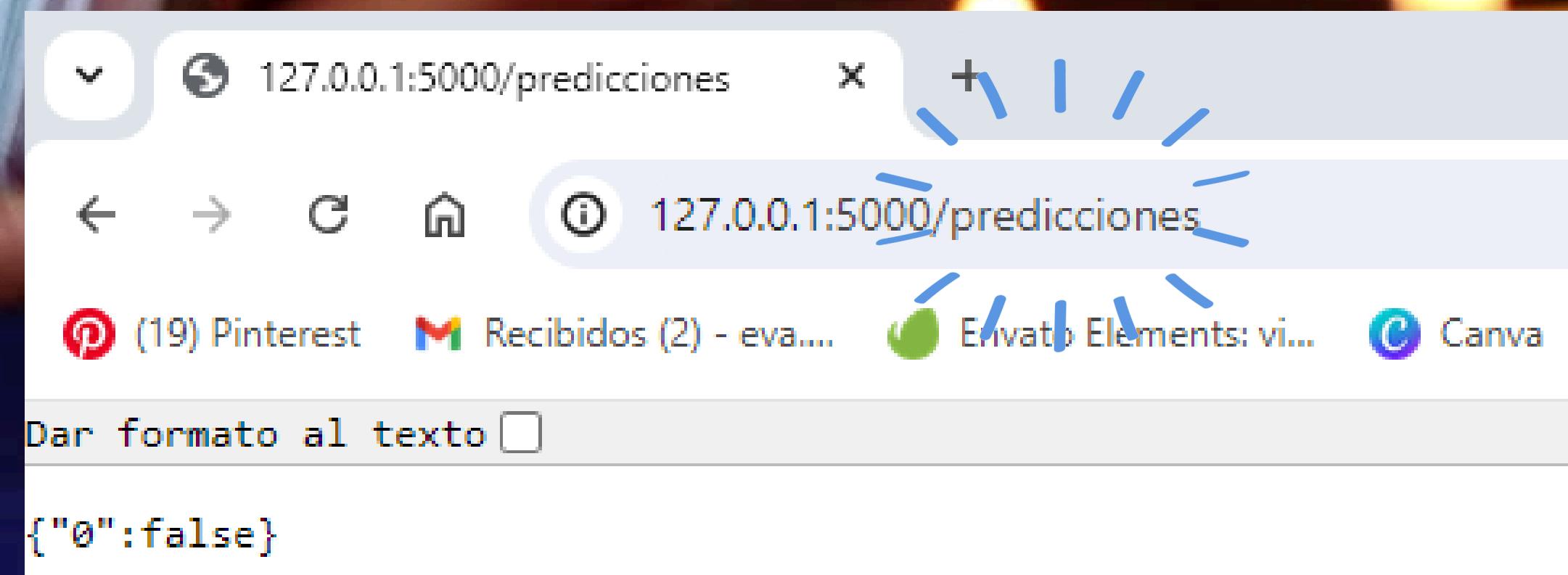


ENDPOINT 6: PREDICCIONES



```
df = pd.DataFrame(y_pred)
```

```
@app.route('/predicciones', methods=['GET'])
def preds():
    return jsonify(df.iloc[2].to_dict())
```





GRACIAS