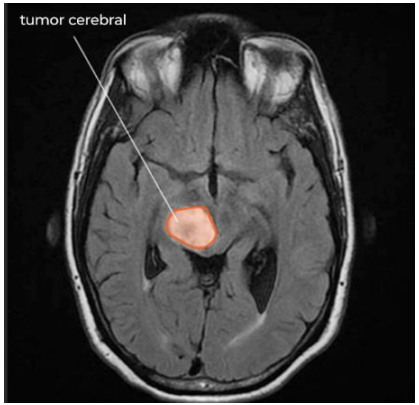
Clasificación y detección de tumores cerebrales con CNN y API en Flask a través de Imágenes de Resonancia Magnética.

Fernando Cabrera Barranzuela

¿QUE ES UN TUMOR CEREBRAL

Un tumor cerebral consiste en un crecimiento anormal de células en el cerebro. El cráneo que encierra el cerebro es muy rígido e ahí el problema que cualquier crecimiento dentro de un espacio tan restringido puede causar problemas.



Fuente: NIH :Instituto Nacional del Cáncer
Análisis de sangre puede detectar cambios genéticos en tumores cerebrales (cancer.gov)

CLASIFICACIÓN DE TUMORES CEREBRALES

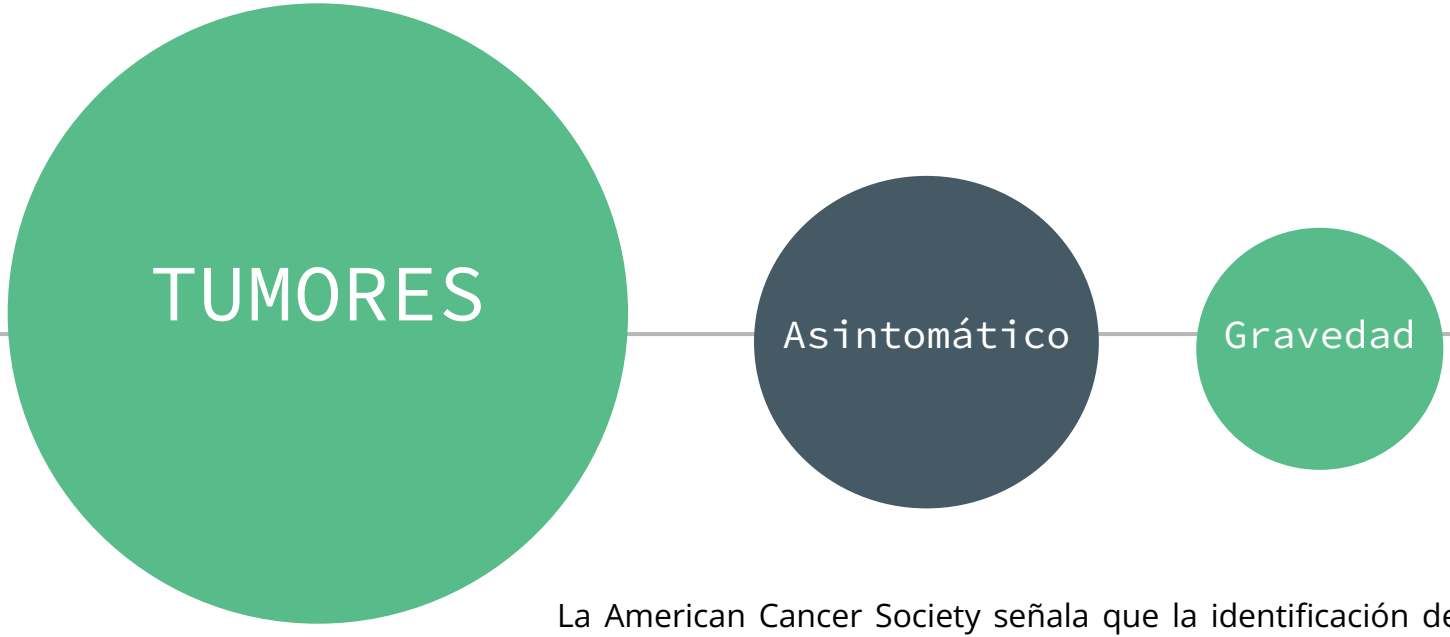
Glioma: Grupo de tumores que se originan en las células gliales del cerebro. Las células gliales son las que proporcionan soporte y protección a las neuronas. Estos tumores pueden ser agresivo y potencialmente mortales. Dependiendo de su tipo y grado. Los gliomas de alto grado son más peligrosos y pueden crecer rápidamente, mientras que los de bajo grado suelen ser más lentos y pueden ser tratados más efectivamente.

Meningioma: Tumor en las meninges, las membranas que rodean el cerebro y la médula espinal. Generalmente benigno, aunque puede causar problemas neurológico que incluyen convulsiones, problemas visuales y dolores de cabeza

Pituitaria: Tumor que se origina en la glándula pituitaria, glándula que regula funciones hormonales, con posibles tumores llamados adenomas, que son en su mayoría benignos. Pueden causar complicaciones graves si crecen y presionan estructuras cerebrales adyacentes interfiriendo en el equilibrio hormonal y causar diversos problemas de salud como crecimiento (enanismo o gigantismo), menstruales u otros desbalances hormonales.

Sin tumor: Ausencia de tumoreS

IMPORTANCIA



La American Cancer Society señala que la identificación de tumores en etapas iniciales permite intervenciones más efectivas, aumentando las probabilidades de un tratamiento exitoso de esta forma salvando vidas.

OBJETIVO

Crear una herramienta médica que agilice esta identificación de sujetos en riesgo.

SOLUCIÓN

Se detectara la presencia de tumores y se clasificara:

Glioma, Meningioma, Pituitaria y no presencia de tumor

ENFOQUE DE EVALUACIÓN

Optimizar RECALL. Es un modelo de clasificación multiclase conformado por 4 clases.

Se busca reducir el "FALSO NEGATIVO".

Recordar que esta métrica mientras más se acerca a la unidad, mejor.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Falso Negativo

Decirle a un paciente que está sano cuando no lo está, esto se refleja a predecir que no existe tumor cuando sí existe. Es un error bastante grave en el área médica.

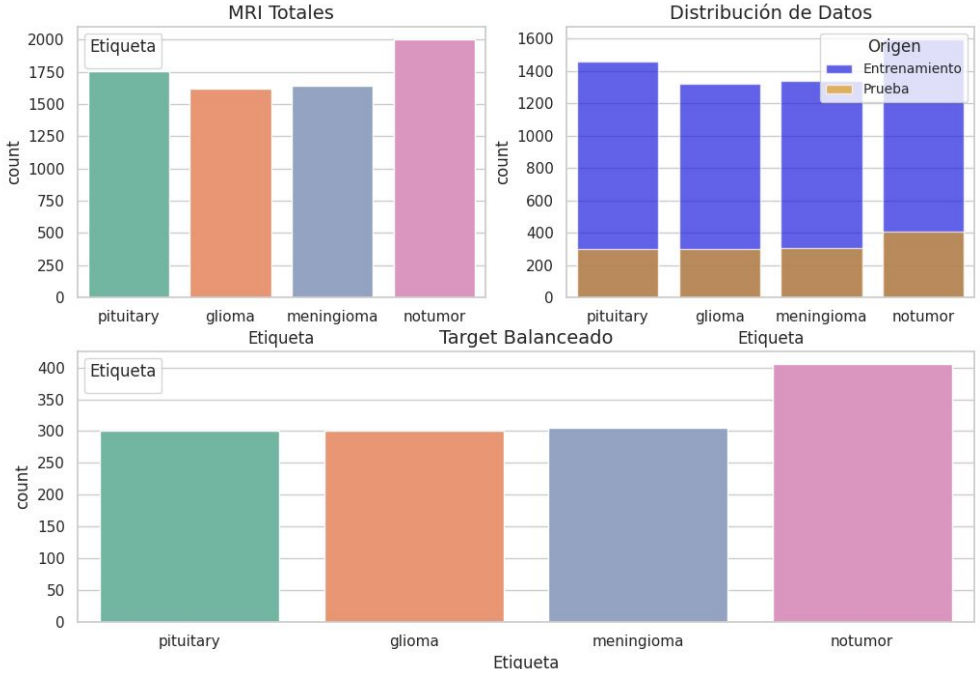
METODOLOGÍA

1. CARGA DE DATOS A TRAVÉS DE API DE KAGGLE

Este conjunto de datos contiene 7023 imágenes de resonancia magnética del cerebro humano que se clasifican en 4 clases: glioma - meningioma - sin tumor y hipófisis.

Se extrajeron los datos del api de Kaggle. El tamaño de las imágenes en este conjunto de datos es diferente, encontramos imágenes en 2D, 3D y 4D

2. EXPLORACION Y ANALISIS DE DATOS



DATOS ORGINALES, total: 7023

	Etiqueta	Conteo	Porcentaje
0	notumor	2000	28.0%
1	pituitary	1757	25.0%
2	meningioma	1645	23.0%
3	glioma	1621	23.0%

DISTRIBUCION DE DATOS ENTRENAMIENTO, total: 5712

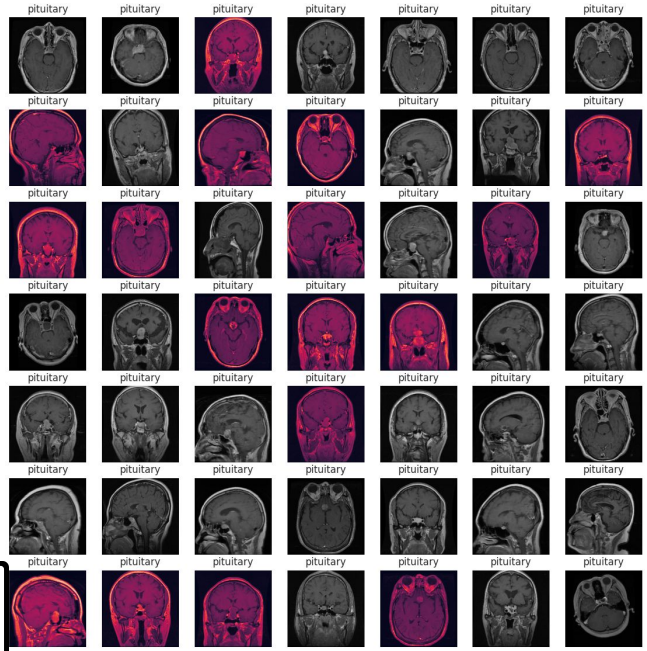
Etiqueta	Etiqueta	Conteo	Porcentaje
Etiqueta			
notumor	notumor	1595	28.0%
pituitary	pituitary	1457	26.0%
meningioma	meningioma	1339	23.0%
glioma	glioma	1321	23.0%

DISTRIBUCION DE DATOS PRUEBA, total: 1311 = 18.67 %

Etiqueta	Etiqueta	Conteo	Porcentaje
Etiqueta			
notumor	notumor	405	31.0%
meningioma	meningioma	306	23.0%
pituitary	pituitary	300	23.0%
glioma	glioma	300	23.0%

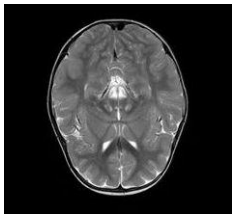
3. PREPROCESAMIENTO DE IMÁGENES

Escalamiento imágenes y etiquetas. Se tuvo problemas con la imagen en 2D, 3D y 4D, se unificó todo al formato RGB

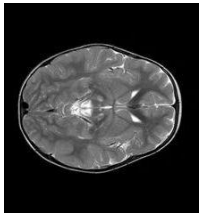


4. INGENIERÍA DE CARACTERÍSTICAS

Se agregó 4 tipos diferentes de grados para rotar la imagen (30, 90, 180 y 270 grados). A la cantidad original de imágenes se le aumentó 4 veces la misma cantidad. Se observa el aumento de imágenes de la cantidad inicial x 5.



90°



DATOS ORGINALES, total: 35115

	Etiqueta	Conteo	Porcentaje
0	notumor	10000	28.0%
1	pituitary	8785	25.0%
2	meningioma	8225	23.0%
3	glioma	8105	23.0%

DISTRIBUCION DE DATOS ENTRENAMIENTO, total: 28560

Etiqueta	Etiqueta	Conteo	Porcentaje
notumor	notumor	7975	28.0%
pituitary	pituitary	7285	26.0%
meningioma	meningioma	6695	23.0%
glioma	glioma	6605	23.0%

DISTRIBUCION DE DATOS PRUEBA, total: 6555 = 19 %

Etiqueta	Etiqueta	Conteo	Porcentaje
notumor	notumor	2025	31.0%
meningioma	meningioma	1530	23.0%
pituitary	pituitary	1500	23.0%
glioma	glioma	1500	23.0%

5. CREAR MODELO CNN

Se fueron modificando parámetros para obtener los mejores resultados.

Modelo CNN (red neuronal convoluciona)
N°epocas = 55
Regularizado L1 ; reduce overfitting
Optimizador Adam
Funcion de perdida:categorical_crossentropy
Metrica evaluada:accuracy
early_stopping = 10

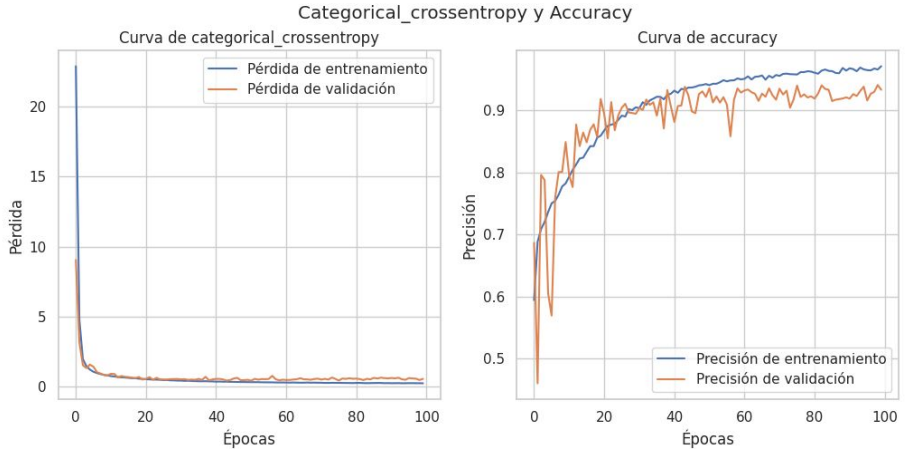
#Capa inicial
Conv2d(filters = 16)
Dropout(0.3)
Conv2d(filters = 16)
#Capa oculta
Conv2d(filters = 32)
Conv2d(filters = 32)
#Capa oculta
Conv2d(filters = 64)
Conv2d(filters = 64)
#Capas agrupamiento
MaxPooling2D
#Capa de aplanamiento
Flatten
#Capa de salida
Dense(128)
Dropout(0.3)
Dense(256)
Dense(4, activation='sofmax')

```
205/205 [==============================] - 6s 31ms/step
METRICAS DE EVALUACION MODELO RNN:
----------------------------------

--Reporte de Clasificacion:--
              precision    recall  f1-score   support

           0       0.74      0.91      0.81      1500
           1       0.83      0.66      0.73      1530
           2       0.95      0.89      0.92      2025
           3       0.91      0.96      0.94      1500

    accuracy                           0.86      6555
   macro avg       0.86      0.86      0.85      6555
weighted avg       0.87      0.86      0.86      6555
```

- Glioma = 0
- Meningioma = 1
- Notumor = 2
- Pituitary = 3

```
Modelo CNN (red neuronal convoluciona)
Épocas = 100
Regularizado L1 ; reduce overfitting
Optimizador Adam
Función de perdida : categorical_crossentropy
Métrica evaluada : accuracy

#Capa inicial
Conv2d(filtros = 16)
Conv2d(filtros = 16)
#Capa oculta
Conv2d(filtros = 32)
Conv2d(filtros = 32)
#Capa oculta
Conv2d(filtros = 64)
Conv2d(filtros = 64)
#Capas agrupamiento
MaxPooling2D
#Capa de aplanamiento
Flatten
#Capa de salida
Dense(128)
Dropout(0.3)
Dense(256)
Dense(4, activation='sofmax')
```

6. EVALUACIÓN DE VALIDACIÓN



Conclusion del sobreajuste

ACCURACY. Los resultados sugiere que el modelo está generalizando de manera razonable a datos que no ha visto antes, sin una caída drástica en el rendimiento. Existe un minimo rango de sobreajuste, pero es aceptable.

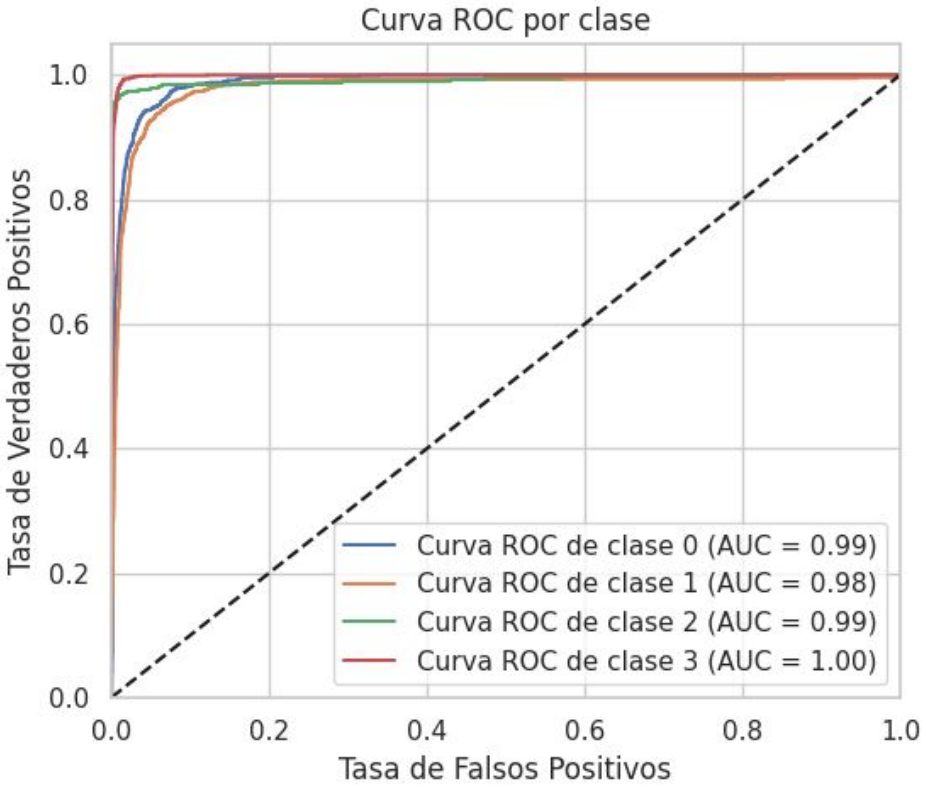
- accuracy prueba = 94%
- accuracy validacion = 93.31%
- accuracy entrenamiento = 97%

LOSS. Se demuestra que el modelo tiene una buena capacidad de predecir en terminos de minimizar error

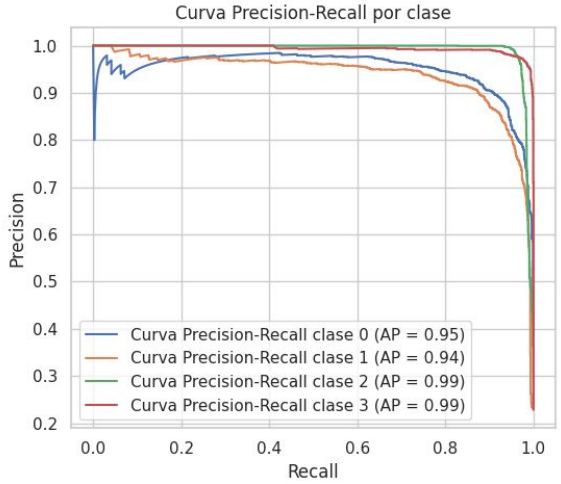
-loss prueba = 0.44
-loss validacion = 0.56
-loss entrenamiento = 0.24

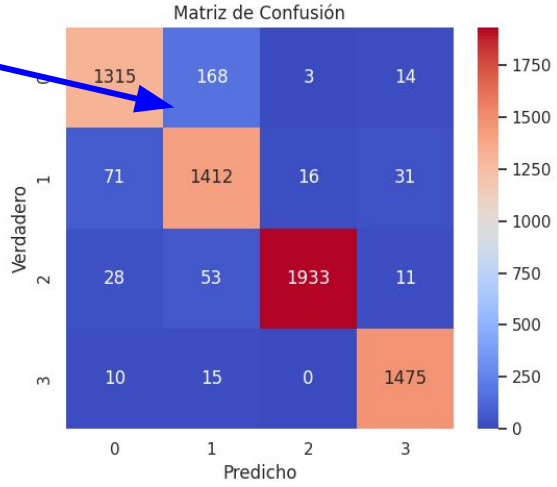
Interpretacion El modelo probablemente ha sufrido un ligero grado de sobreajuste, ya que su rendimiento en el conjunto de validación es un poco más bajo que en el conjunto de entrenamiento en accuracy. El modelo esta generalizando bien, con una precisión consistente en los conjuntos de validación y prueba.

7. MÉTRICAS EVALUACIÓN -PREDICCIÓN



```
205/205 [==============================] - 8s 39ms/step
METRICAS DE EVALUACION MODELO RNN:
----------------------------------

--Reporte de Clasificacion:--
              precision    recall  f1-score   support

           0       0.92      0.88      0.90      1500
           1       0.86      0.92      0.89      1530
           2       0.99      0.95      0.97      2025
           3       0.96      0.98      0.97      1500

    accuracy                           0.94      6555
   macro avg       0.93      0.93      0.93      6555
weighted avg       0.94      0.94      0.94      6555
```





DESPLEGAR API EN GCP

API CON FASK

Se construyó un api para recibir las imágenes, procesarlas y devolver la predicción

app.py

```python
from flask import Flask, request, jsonify, render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os

app = Flask(__name__)
modelo = load_model("modelo_mri_tumor.h5")

def preproceso_imagenes(img_path):
    imagen = image.load_img(img_path, target_size=(224,224))

    # Convertir la imagen a array de numpy
    img_array = image.img_to_array(imagen)/255.0  # Escalar la imagen a valores entre 0 y 1

    # Expandir dimensiones para que coincida con el formato de entrada del modelo (batch size, height, width, channels)
    img_array = np.expand_dims(img_array, axis=0)

    return img_array

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"})

    file = request.files['file']

    if file.filename == '':
        return jsonify({"error": "No selected file"})

    if file:
        # Crear la carpeta 'uploads' si no existe
        if not os.path.exists('uploads'):
            os.makedirs('uploads')

        file_path = os.path.join("uploads", file.filename)
        file.save(file_path)

        # Intentar preprocesar la imagen
        try:
            img_array = preproceso_imagenes(file_path)
            prediction = modelo.predict(img_array)
        except Exception as e:
            return jsonify({"error": str(e)}), 500  # Devolver el error si algo falla

        class_names = ['glioma','meningioma','notumor', 'pituitary']
        predicted_class = class_names[np.argmax(prediction)]

        return jsonify({"prediction": predicted_class})

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

CONTENER LA APLICACIÓN CON DOCKER

Para que sea más fácil desplegar la aplicación, se utiliza Docker para crear un contenedor con todo el entorno necesario.

Dockerfile

```dockerfile
# Usa una imagen base de Python
FROM python:3.8-slim

# Copia los archivos de tu proyecto a la imagen
COPY . /app

# Establece el directorio de trabajo
WORKDIR /app

# Instala las dependencias necesarias
RUN pip install -r requirements.txt

# Expone el puerto en el que correrá Flask
EXPOSE 5000

# Comando para ejecutar la aplicación
CMD ["python", "app.py"]
```

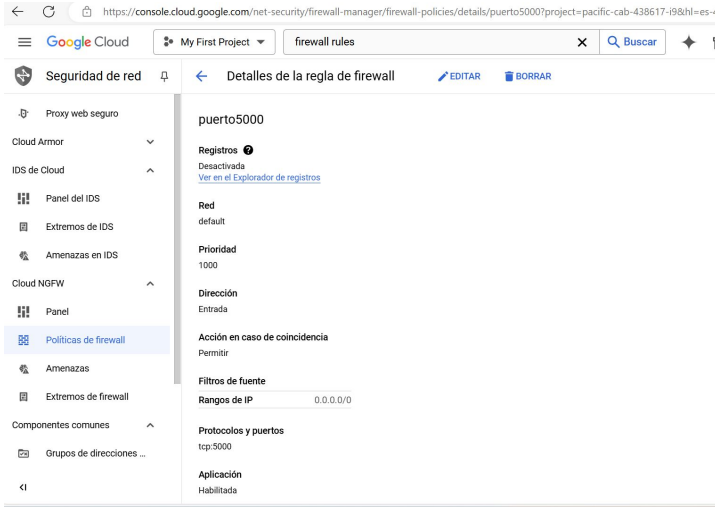
DEPENDENCIAS

requirements.txt

```
Flask==3.0.3
tensorflow
numpy
pillow
```

CONFIGURAR SERVIDOR GCP

Crear una nueva reglas de firewall de GCP donde abro el puerto 5000 para poder acceder a la aplicación desde el navegador.



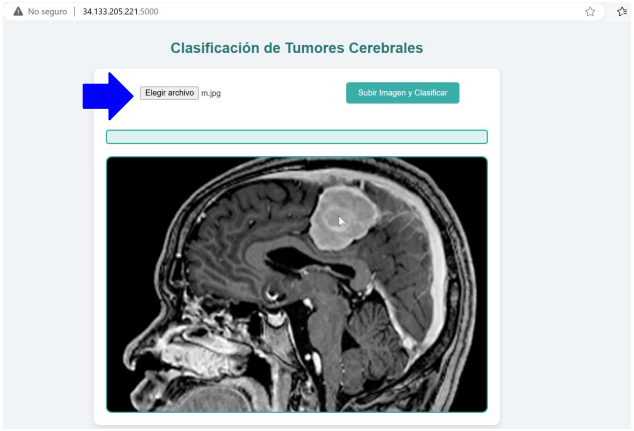
CONECTAR API CON HTML

Crear un archivo html, que contenga todas las interacciones que se mostrará en la web. Conectarlo a mi api, ejecutar y hacer las pruebas

RESULTADOS:

1. Api pública:puerto 5000

2. Elegir archivo(radiografía)



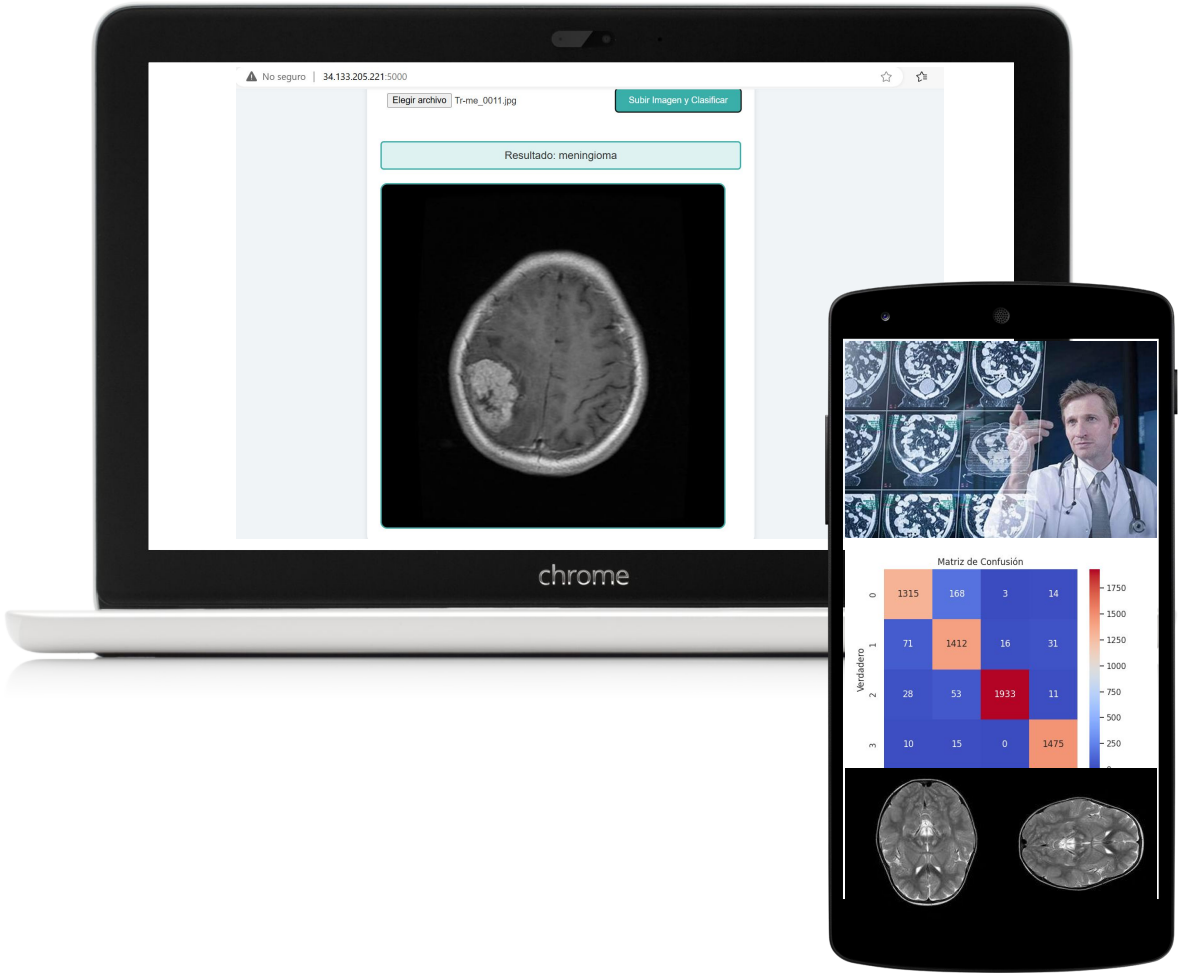
3. Subir imagen y clasifica

RESULTADO:

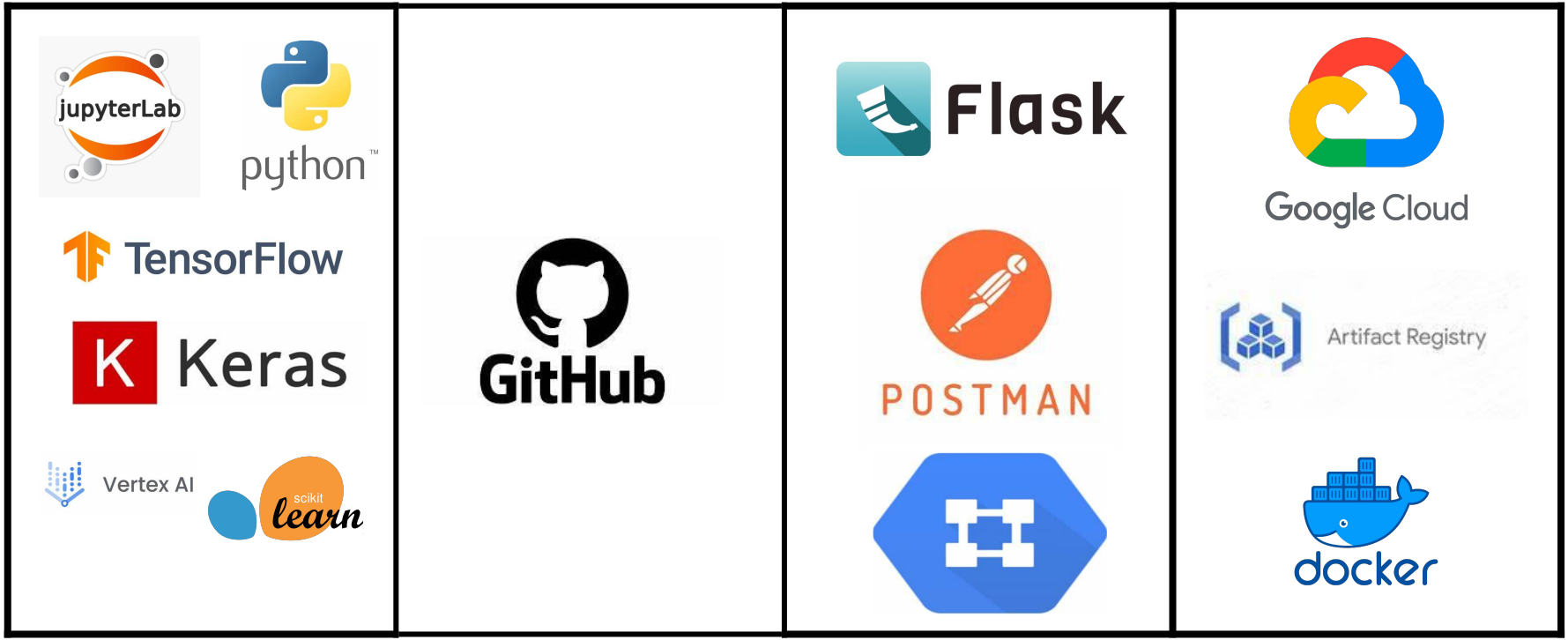
El modelo funciona, no presenta sobreajuste, tiene buen recall. Optimizar la clase 1.

MEJORAS

- Lo más real sería una data desbalanceada
- Aumentar grados a la clasificación, para determinar si es mortal o no
- Aumentar a 30% la cantidad de prueba
- Una resonancia magnética está conformada por SLICES de imágenes que nos dan una visión general del cerebro. Mejorar el dataset con exclusivamente slices centrales .



TECNOLOGÍAS USADAS



EDA, Investigación y construcción del modelo

Crear repositorio

Pruebas

Despliegue