UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS (VALLE DE SULA)

Catedrático:

Juan Enrique Alvarenga Rodas

Clase:

Inteligencia Artificial

Sección:

1800

Integrantes:

Andrea Estefanía Andrade Alvarenga
20142003006

Carlos Eduardo Argueta Rivera
20172001120

Rafael Ernesto Morales Díaz
20202001873

Tema:

Modelo Entrenado para Predecir el Estado Vital de Pacientes con COVID-19

Fecha:

11-8-2025

Índice

Introducción

El avance de la inteligencia artificial ha permitido el desarrollo de herramientas capaces de asistir en tareas críticas como el diagnóstico médico y la predicción de resultados clínicos. En el contexto de la pandemia de COVID-19, no solo ha sido fundamental identificar la infección, sino también estimar la probabilidad de desenlaces graves como la muerte. Este proyecto tiene como finalidad implementar un modelo de aprendizaje automático que sea capaz de predecir si un paciente diagnosticado con COVID-19 se encuentra vivo o ha fallecido, en función de sus datos clínicos y síntomas registrados.

El modelo se entrena con base en datos confiables que incluyen información clínica de pacientes confirmados con COVID-19, tales como edad, género, síntomas y condiciones preexistentes. El entrenamiento y la evaluación del modelo se lleva a cabo mediante Google Colab, utilizando algoritmos de clasificación como árboles de decisión y máquinas de soporte vectorial (SVM). Se busca alcanzar una precisión superior al 90%, con el fin de evaluar la viabilidad del modelo como herramienta de análisis de riesgo en contextos clínicos y académicos.

Este proyecto es desarrollado de forma colaborativa por tres estudiantes de la clase de Inteligencia Artificial, bajo una visión crítica, académica y orientada al análisis exploratorio de datos clínicos para apoyar la toma de decisiones en salud.

Objetivos

Objetivo general

- Desarrollar un modelo de inteligencia artificial capaz de predecir con alta precisión (mayor al 90%) si un paciente diagnosticado con COVID-19 se encuentra vivo o ha fallecido, usando técnicas de Machine Learning en Google Colab.

Objetivos específicos

- Realizar una limpieza y preprocesamiento exhaustivo del conjunto de datos clínicos, asegurando la eliminación de valores faltantes, atípicos o irrelevantes para la predicción del estado vital.
- Aplicar técnicas de análisis exploratorio de datos (EDA) para identificar relaciones estadísticas y patrones relevantes entre las variables clínicas y la mortalidad por COVID-19.
- Abordar el desbalance de clases mediante técnicas de sobremuestreo como SMOTE, con el fin de mejorar el desempeño del modelo ante datos clínicos desbalanceados.
- Entrenar y optimizar modelos de clasificación supervisada, como Árboles de Decisión, utilizando métricas de validación cruzada para evaluar su desempeño.
- Evaluar rigurosamente el rendimiento del modelo utilizando métricas como accuracy, precision, recall, F1-score y matriz de confusión, enfocándose en la correcta detección de pacientes fallecidos.
- Documentar e interpretar los resultados obtenidos, proporcionando visualizaciones claras y explicaciones accesibles sobre el comportamiento del modelo y su posible utilidad en escenarios clínicos reales.

Metodología

La metodología incluirá la recopilación de datos, preprocesamiento, análisis exploratorio, balanceo de clases con SMOTE, entrenamiento del modelo (Árboles de Decisión y SVM) y validación mediante métricas como accuracy, precision, recall, F1-score y matriz de confusión.

---

1. Concepto y origen

La pandemia de la enfermedad por coronavirus 2019 (COVID-19) representa uno de los mayores retos sanitarios globales del siglo XXI. El virus SARS-CoV-2 fue identificado por primera vez en Wuhan, provincia de Hubei, China, en diciembre de 2019. La rápida propagación a nivel mundial llevó a la Organización Mundial de la Salud (OMS) a declarar la COVID-19 como una Emergencia de Salud Pública de Importancia Internacional (ESPII) el 30 de enero de 2020, y como pandemia el 11 de marzo del mismo año (WHO, 2020). Los coronavirus son virus envueltos con ARN de cadena positiva que infectan tanto a humanos como a animales, y previamente habían provocado brotes significativos como el SARS-CoV (2002-2003) y el MERS-CoV (2012).

2. Etiología y características del SARS-CoV-2

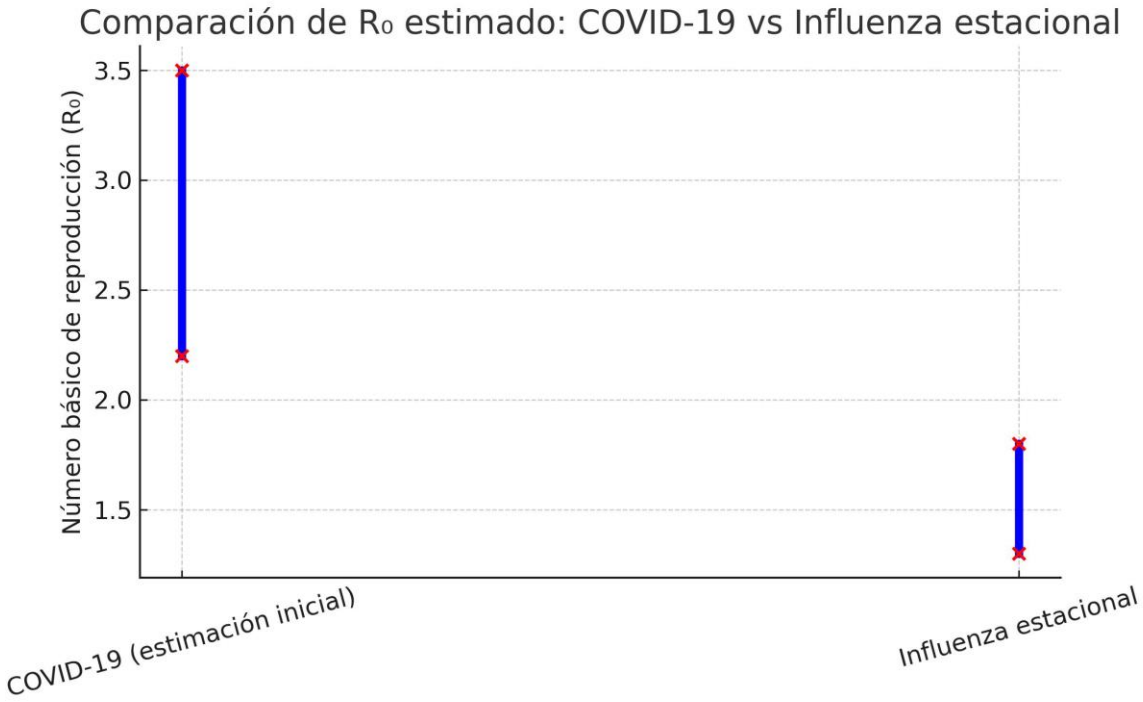
El SARS-CoV-2 pertenece al género Betacoronavirus y utiliza el receptor enzima convertidora de angiotensina 2 (ACE2) para entrar en las células humanas (Hoffmann et al., 2020). Su estructura incluye la proteína S (spike), responsable de la unión al receptor, la proteína N (nucleocápside), la proteína M (de membrana) y la proteína E (de envoltura). La alta afinidad de la proteína S por el receptor ACE2 explica la facilidad de transmisión y la afectación multisistémica, dado que ACE2 se expresa en células del epitelio respiratorio, endotelio vascular, riñones, intestinos y corazón.

3. Transmisión y dinámica epidemiológica

La transmisión principal ocurre por gotículas (pequeñas gotas de saliva a través de las cuales se propaga un virus) respiratorias expulsadas al toser, estornudar o hablar, así como por contacto con superficies contaminadas. También se ha documentado la transmisión aérea en entornos cerrados y poco ventilados. La tasa básica de reproducción (R_o) estimada en las primeras fases fue de entre 2 y 3, lo que refleja un potencial de propagación significativo (Liu et al., 2020). El virus se transmite principalmente por:

- Gotículas respiratorias (>5 µm) emitidas al hablar, toser o estornudar.
- Aerosoles (partículas <5 µm) en ambientes cerrados y mal ventilados.
- Contacto indirecto con superficies contaminadas.

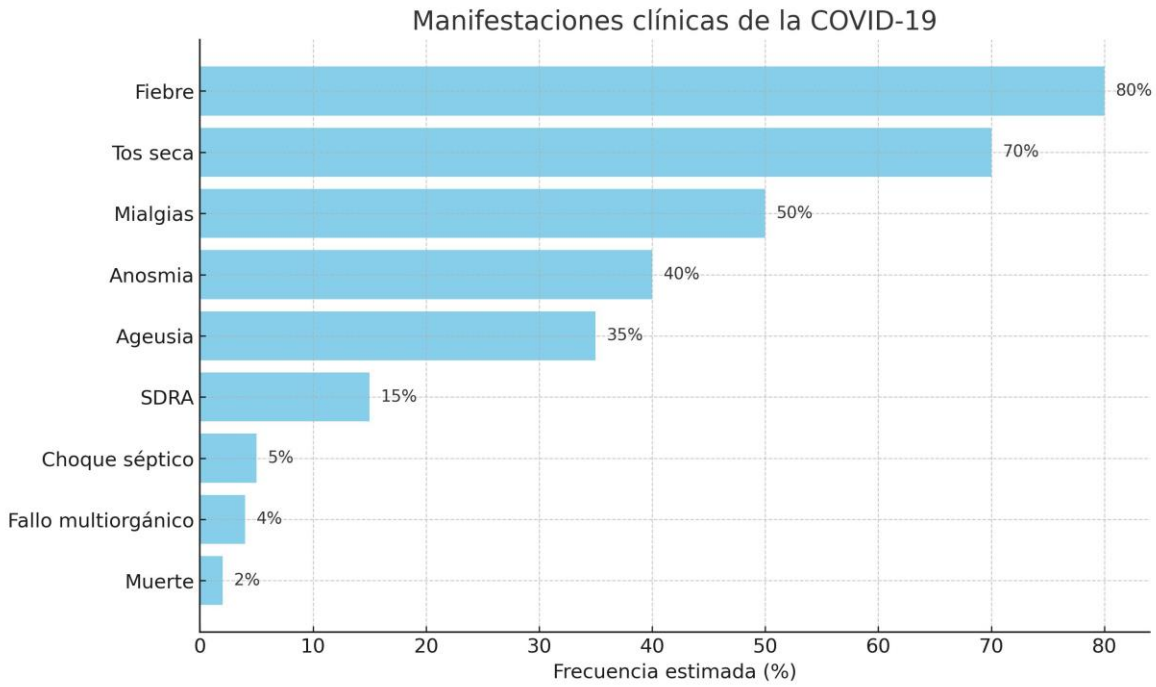
Estudios iniciales estimaron un número básico de reproducción (R_0) entre 2.2 y 3.5, superior al de la influenza estacional (Liu et al., 2020). La epidemiología ha mostrado un patrón con olas de contagio influenciadas por variantes virales, medidas de control y estacionalidad.



4. Manifestaciones clínicas y diagnóstico

La COVID-19 presenta un espectro clínico que va desde infecciones asintomáticas hasta enfermedad grave. Los síntomas más frecuentes incluyen fiebre, tos seca, mialgias, anosmia y ageusia (Guan et al., 2020). En casos severos puede evolucionar a síndrome de dificultad respiratoria aguda (SDRA), choque séptico, fallo multiorgánico y muerte.
El diagnóstico se basa en:

- Pruebas moleculares (RT-PCR) como estándar de oro.
- Pruebas rápidas de antígeno para diagnóstico inicial.
- Serología para estudios epidemiológicos y seguimiento.



5. Medidas de prevención y control

Entre las medidas no farmacológicas adoptadas globalmente destacan:

- Uso universal de mascarilla en espacios públicos.
- Distanciamiento físico de al menos 1-2 metros.
- Higiene frecuente de manos.
- Ventilación adecuada de espacios cerrados.
- Restricciones de movilidad y confinamientos (Flaxman et al., 2020).
- Estas medidas han demostrado eficacia para reducir la transmisión, especialmente antes de la disponibilidad de vacunas.

7. Impacto social y económico

Más allá del ámbito sanitario, la pandemia incrementó los problemas de salud mental como ansiedad, depresión y estrés postraumático. El cierre de actividades afectó a millones de empleos y acentuó desigualdades sociales, especialmente en países en desarrollo (OECD, 2020).

La pandemia ha tenido consecuencias profundas:

- **Económicas:** Caída del PIB global, cierre de negocios, pérdida de empleos, interrupción de cadenas de suministro.

- **Educativas:** cierre de instituciones y transición acelerada a educación en línea.

- **Psicológicas:** aumento de ansiedad, depresión y estrés (OECD, 2020)

- **Desigualdad social:** mayor afectación en poblaciones vulnerables con acceso limitado a servicios de salud.

8. Perspectivas futuras

Aunque la vacunación masiva ha reducido la mortalidad, el virus continúa circulando. Se prevé que el SARS-CoV-2 se torne endémico, con brotes estacionales y necesidad de refuerzos vacunales, especialmente para grupos de riesgo.

El aprendizaje obtenido ha impulsado avances en vigilancia genómica, preparación ante pandemias y desarrollo rápido de vacunas.

Explicación del Dataset

Primero se va a analizar el dataset que contiene una gran cantidad de información anónima relacionada con el paciente, incluidas las condiciones previas.

El conjunto de datos fue proporcionado por el gobierno mexicano (enlace). Este conjunto de datos contiene una enorme cantidad de información anónima relacionada con el paciente, incluidas las condiciones previas. El conjunto de datos sin procesar consta de 21 características únicas y 1.048.576 pacientes únicos. **En las características booleanas, 1 significa "sí" y 2 significa "no". A los valores 97 y 99 les faltan datos**.

- **sexo:** 1 para mujer y 2 para hombre.
- **edad:** Del paciente.
- **Clasificación:** Hallazgos de la prueba de COVID. Los valores 1-3 significan que el paciente fue diagnosticado con COVID en diferentes grados. 4 o superior significa que el paciente no es portador de COVID o que la prueba no es concluyente.
- **tipo de paciente:** Tipo de atención que recibió el paciente en la unidad. 1 para regreso a casa y 2 para hospitalización.
- **neumonía:** Si el paciente ya tiene inflamación de los sacos aéreos o no.
- **embarazo:** Si la paciente está embarazada o no.
- **diabetes:** Si el paciente tiene diabetes o no.
- **EPOC:** Indica si el paciente tiene enfermedad pulmonar obstructiva crónica o no.
- **asma:** Si el paciente tiene asma o no.
- **inmsupr:** Si el paciente está inmunodeprimido o no.
- **hipertensión:** Si el paciente tiene hipertensión o no.
- **cardiovascular:** Si el paciente tiene una enfermedad relacionada con el corazón o los vasos sanguíneos.
- **crónica renal:** Si el paciente tiene enfermedad renal crónica o no.
- **otra enfermedad:** Si el paciente tiene otra enfermedad o no.
- **obesidad:** Si el paciente es obeso o no.

- **tabaco:** Si el paciente es consumidor de tabaco.
- **usmr:** Indica si el paciente atendió unidades médicas de primer, segundo o tercer nivel.
- **unidad médica:** Tipo de institución del Sistema Nacional de Salud que brindó la atención.
- **intubado:** Si el paciente estaba conectado al ventilador.
- **UCI:** Indica si el paciente había sido ingresado en una Unidad de Cuidados Intensivos.
- **fecha de fallecimiento:** Si el paciente falleció indique la fecha de fallecimiento, y 9999-99-99 en caso contrario.

El proceso de entrenamiento del modelo fue realizado utilizando un enfoque supervisado, en el cual el modelo aprende a partir de datos etiquetados. En este caso, la etiqueta corresponde a si un paciente con COVID-19 se encuentra vivo (1) o muerto (0), representada en la columna alive_patients, derivada de la columna DATE_DIED.

Explicación del preprocesamiento de datos

Se crea la columna que se va predecir, la cual es “alive_patients” con respecto a la columna DATA_DIED, se van analizar las columnas que tiene relación con DATA_DIED y así lograr una predicción con un porcentaje alto, se implementará el modelo de Árboles de decisiones para este caso, ya hacer un dataset casi todo booleano y por medio de limpieza del mismo lograremos normalizar el dataset a booleanos, lo que es perfecto para el Árbol de Decisiones.

```
df["alive_patients"] = df["DATE_DIED"].apply(lambda x: 0 if x != "9999-99-99" else 1)
pacientes_vivos = df["alive_patients"].value_counts()[1]
```

“Si el paciente falleció indique la fecha de fallecimiento, y 9999-99-99 en caso contrario.”

1. Análisis de datos del dataset

-Con **df.info()** miramos como está compuesto el dataframe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 21 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   USMER                 1048575 non-null  int64
 1   MEDICAL_UNIT          1048575 non-null  int64
 2   SEX                   1048575 non-null  int64
 3   PATIENT_TYPE          1048575 non-null  int64
 4   DATE_DIED             1048575 non-null  object
 5   INTUBED               1048575 non-null  int64
 6   PNEUMONIA             1048575 non-null  int64
 7   AGE                   1048575 non-null  int64
 8   PREGNANT              1048575 non-null  int64
 9   DIABETES              1048575 non-null  int64
 10  COPD                  1048575 non-null  int64
 11  ASTHMA                1048575 non-null  int64
 12  INMSUPR               1048575 non-null  int64
 13  HIPERTENSION          1048575 non-null  int64
 14  OTHER_DISEASE         1048575 non-null  int64
 15  CARDIOVASCULAR        1048575 non-null  int64
 16  OBESITY               1048575 non-null  int64
 17  RENAL_CHRONIC         1048575 non-null  int64
 18  TOBACCO               1048575 non-null  int64
 19  CLASIFFICATION_FINAL  1048575 non-null  int64
 20  ICU                   1048575 non-null  int64
dtypes: int64(20), object(1)
memory usage: 168.0+ MB
```

- Tenemos 21 Columnas y 1048575 Filas.
- Todas las columnas son enteras excepto DATE_DIED.
- Debemos analizar valores nulos, valores (values) como 97 y 99 para “missing data”.
- En las columnas booleanas 0. bool = 2 -> no | 1. bool = 1 -> yes.
- La columna Target la deduciremos de la columna DATE_DIED, para saber el estado del paciente.

-Con el **df.head(5)** miramos que CLASIFICATION_FINAL Y AGE no son booleanas

NIA	AGE	PREGNANT	DIABETES	COPD	ASTHMA	INMSUPR	HIPERTENSION	OTHER_DISEASE	CARDIOVASCULAR	OBESITY	RENAL_CHRONIC	TOBACCO	CLASIFFICATION_FINAL	ICU	alive_pat
1	65	2	2	2	2	2	1	2	2	2	2	2	3	97	
1	72	97	2	2	2	2	1	2	2	1	1	2	5	97	
2	55	97	1	2	2	2	2	2	2	2	2	2	3	2	
2	53	2	2	2	2	2	2	2	2	2	2	2	7	97	
2	68	97	1	2	2	2	1	2	2	2	2	2	3	97	

Vamos a excluir las columnas CLASIFICATION_FINAL Y AGE para no alterar la información contenida ya que no son booleanas.

```
# Lista de columnas a excluir
columnas_excluir = ['CLASIFFICATION_FINAL','AGE']

# Reemplazar todos los valores 2 por 0 en las columnas excepto las excluidas
df[df.columns.difference(columnas_excluir)] = df[df.columns.difference(columnas_excluir)].replace(2, 0)
```

✓ 0.3s

2. Valores Nulos

```
print("Columnas que continen valores 97 99.")
print()
for columna in df.columns:

    valores_97_99 = df[columna].isin([97, 99])

    cuenta = valores_97_99.sum()

    print(f"Columna: {columna} - Cuenta: {cuenta} Nulos")
```

[10] ✓ 0.1s Python

```
Columnas que continen valores 97 99.

Columna: USMER - Cuenta: 0 Nulos
Columna: MEDICAL_UNIT - Cuenta: 0 Nulos
Columna: SEX - Cuenta: 0 Nulos
Columna: PATIENT_TYPE - Cuenta: 0 Nulos
Columna: DATE_DIED - Cuenta: 0 Nulos
Columna: INTUBED - Cuenta: 855869 Nulos
Columna: PNEUMONIA - Cuenta: 16003 Nulos
Columna: AGE - Cuenta: 221 Nulos
Columna: PREGNANT - Cuenta: 523511 Nulos
Columna: DIABETES - Cuenta: 0 Nulos
Columna: COPD - Cuenta: 0 Nulos
Columna: ASTHMA - Cuenta: 0 Nulos
Columna: INMSUPR - Cuenta: 0 Nulos
Columna: HIPERTENSION - Cuenta: 0 Nulos
Columna: OTHER_DISEASE - Cuenta: 0 Nulos
Columna: CARDIOVASCULAR - Cuenta: 0 Nulos
Columna: OBESITY - Cuenta: 0 Nulos
Columna: RENAL_CHRONIC - Cuenta: 0 Nulos
Columna: TOBACCO - Cuenta: 0 Nulos
Columna: CLASIFFICATION_FINAL - Cuenta: 0 Nulos
Columna: ICU - Cuenta: 856032 Nulos
Columna: alive_patients - Cuenta: 0 Nulos
```

Podemos observar con este ciclo **for** que tenemos varias columnas con registros nulos (**los valores 97 y 99 en registros indican que les faltan datos**) que nos puedan afectar en nuestra predicción, por lo tanto, vamos a considerar eliminar la columna INTUBED, ICU, ya que a nuestro parecer son columnas con un alto índice de nulos y no creo que sean tan relevantes a la hora de entrenar.

```
columns_to_drop = ["DATE_DIED","INTUBED","ICU"]
df = df.drop(columns_to_drop, axis=1)
```

✓ 0.0s Python

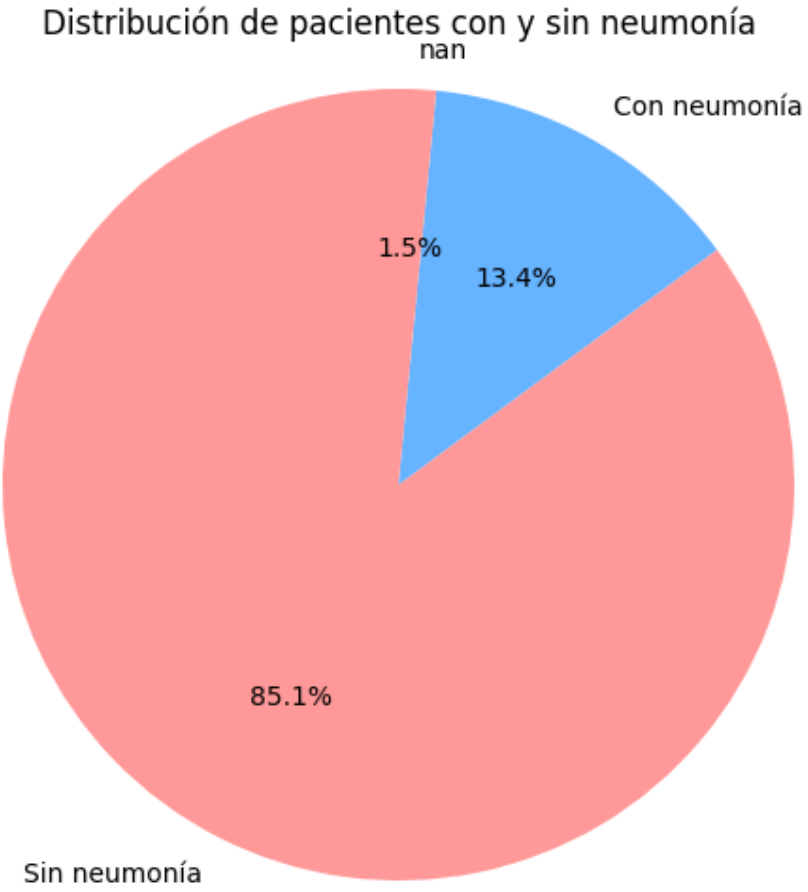
Nos quedan las siguientes columnas con nulos:

PNEUMONIA - Cuenta: 16003 Nulos.

AGE - Cuenta: 221 Nulos.

PREGNANT - Cuenta: 523511 Nulos.

Vamos analizar la columna **PNEUNOMIA**

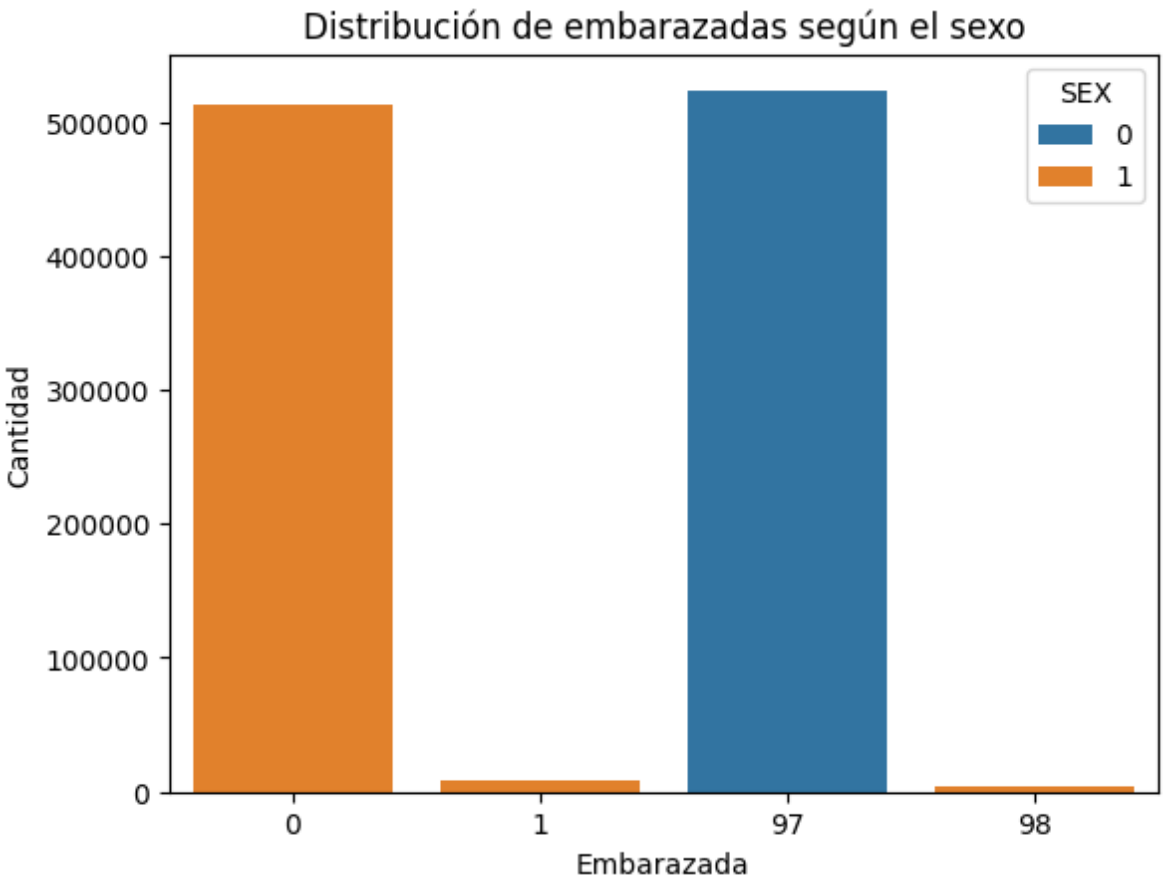


Con este gráfico podemos comprobar que los pacientes que no han rellenado el campo de neumonía son pocos

Podemos ver como esta distribuida la columna PNEUMONIA, los valores nulos suponen un 1,5%, esto sería un 1,5% de datos perdidos si borráramos la columna, visto que la mayoría son sin neumonía, vamos a establecer ese 1,5% a pacientes sin neumonía, ya que no significa mucho.

```
df['PNEUMONIA'] = df['PNEUMONIA'].replace([97, 99], 0)
```
✓ 0.0s

Ahora nos queda **AGE** - Cuenta: 221 Nulos **PREGNANT** - Cuenta: 523511 Nulos



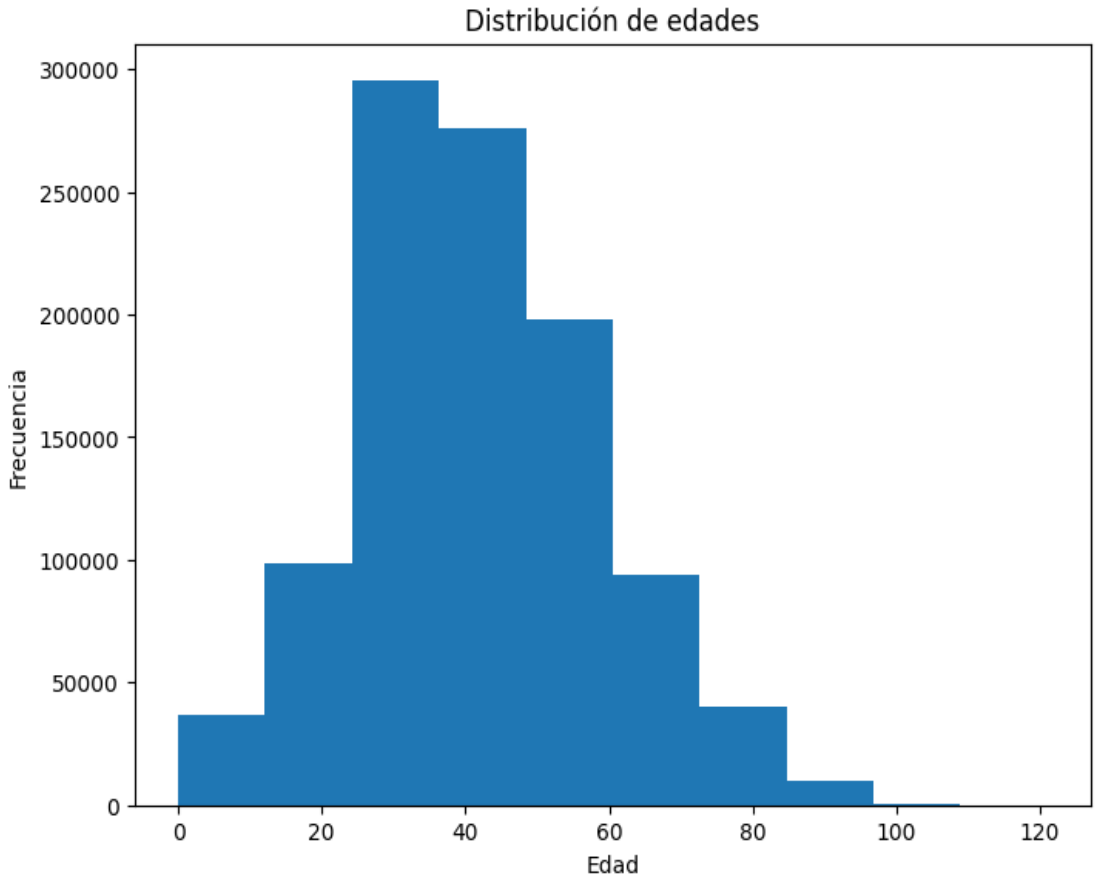
De este gráfico podemos extraer una conclusión y es que ese valor 97 en esta columna corresponde a **male** (hombre). Entonces vamos a sustituir en principio al 97 por 0 para indicarle que es sexo masculino.

Por lo que decidimos mejor eliminar la columna, ya que el valor dominante es de 0 y que 0 = masculino y por lógica no puede estar embarazados.

```
+ Code  + Markdown

df = df.drop('PREGNANT', axis=1)
[9]  ✓ 0.0s
```

Solo queda por tratar la columna **AGE** viendo que los nulos son 221, vamos a rellenar esos datos con la media de la columna.

```
# Calcular la media de la columna "AGE" sin considerar los valores 97 y 99
age_mean = df.loc[(df['AGE'] != 97) & (df['AGE'] != 99), 'AGE'].mean()

# Reemplazar los valores 97 y 99 por la media de la columna "AGE"
df.loc[df['AGE'].isin([97, 99]), 'AGE'] = age_mean
[21]  ✓ 0.0s
```



Y además convertirnos la columna AGE a **int** para así tener todas las columnas con el mismo tipo y tener todos los datos normalizados.

```
df['AGE'] = df['AGE'].astype(int)
```
✓ 0.0s

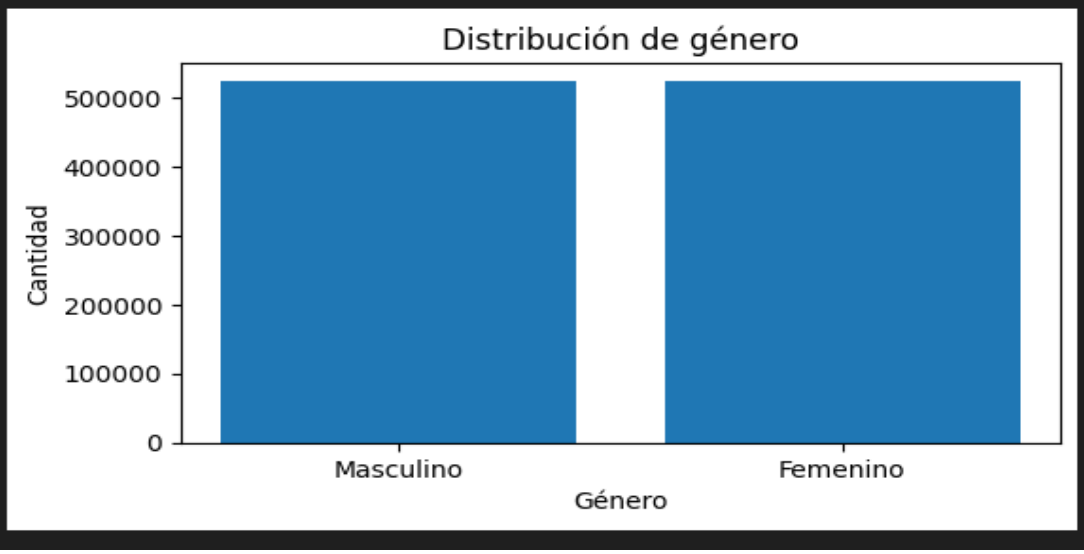
Ya no tenemos nulos ahora podemos proceder a analizar los datos para hacer nuestro entrenamiento de IA.

```
Columnas que continen valores 98.

Columna: USMER - Cuenta: 0 Nulos
Columna: MEDICAL_UNIT - Cuenta: 0 Nulos
Columna: SEX - Cuenta: 0 Nulos
Columna: PATIENT_TYPE - Cuenta: 0 Nulos
Columna: PNEUMONIA - Cuenta: 0 Nulos
Columna: AGE - Cuenta: 124 Nulos
Columna: DIABETES - Cuenta: 3338 Nulos
Columna: COPD - Cuenta: 3003 Nulos
Columna: ASTHMA - Cuenta: 2979 Nulos
Columna: INMSUPR - Cuenta: 3404 Nulos
Columna: HIPERTENSION - Cuenta: 3104 Nulos
Columna: OTHER_DISEASE - Cuenta: 5045 Nulos
Columna: CARDIOVASCULAR - Cuenta: 3076 Nulos
Columna: OBESITY - Cuenta: 3032 Nulos
Columna: RENAL_CHRONIC - Cuenta: 3006 Nulos
Columna: TOBACCO - Cuenta: 3220 Nulos
Columna: CLASIFFICATION_FINAL - Cuenta: 0 Nulos
Columna: alive_patients - Cuenta: 0 Nulos
```

Vamos analizar otras columnas que puedan ser importantes y ver si hay irregularidades.

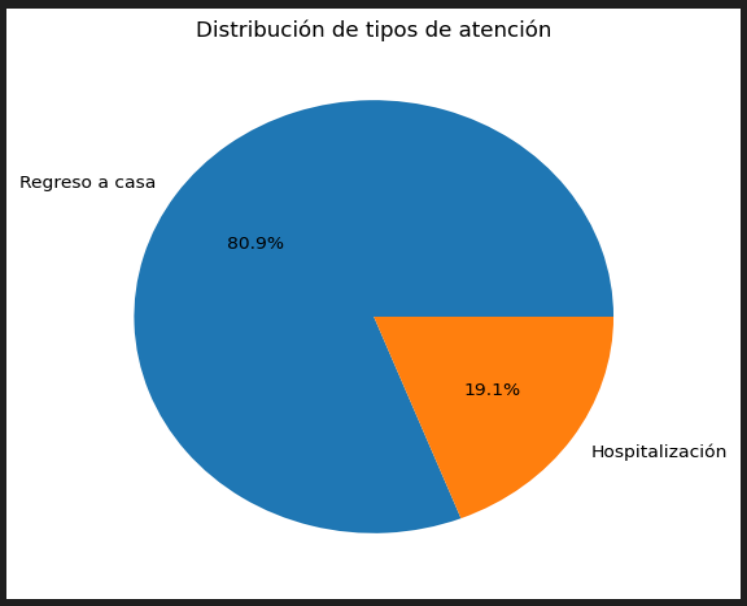
```python
# Gráfico de barras para la columna 'SEX'
sex_counts = df['SEX'].value_counts()
plt.figure(figsize=(6, 3))
plt.bar(sex_counts.index, sex_counts.values)
plt.xlabel('Género')
plt.ylabel('Cantidad')
plt.title('Distribución de género')
plt.xticks(sex_counts.index, ['Femenino', 'Masculino'])
plt.show()
```

✓ 0.1s



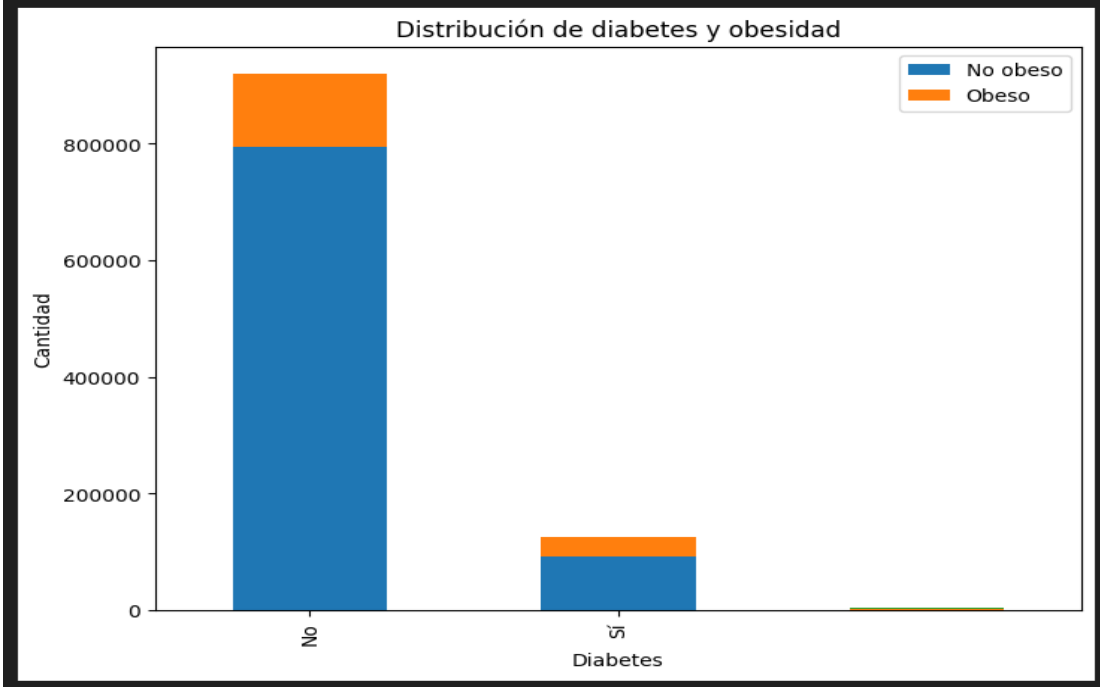

```python
# Gráfico de pastel para la columna 'PATIENT_TYPE'
patient_type_counts = df['PATIENT_TYPE'].value_counts()
plt.figure(figsize=(8, 6))
plt.pie(patient_type_counts.values, labels=['Regreso a casa', 'Hospitalización'], autopct='%1.1f%%')
plt.title('Distribución de tipos de atención')
plt.show()
```

✓ 0.0s



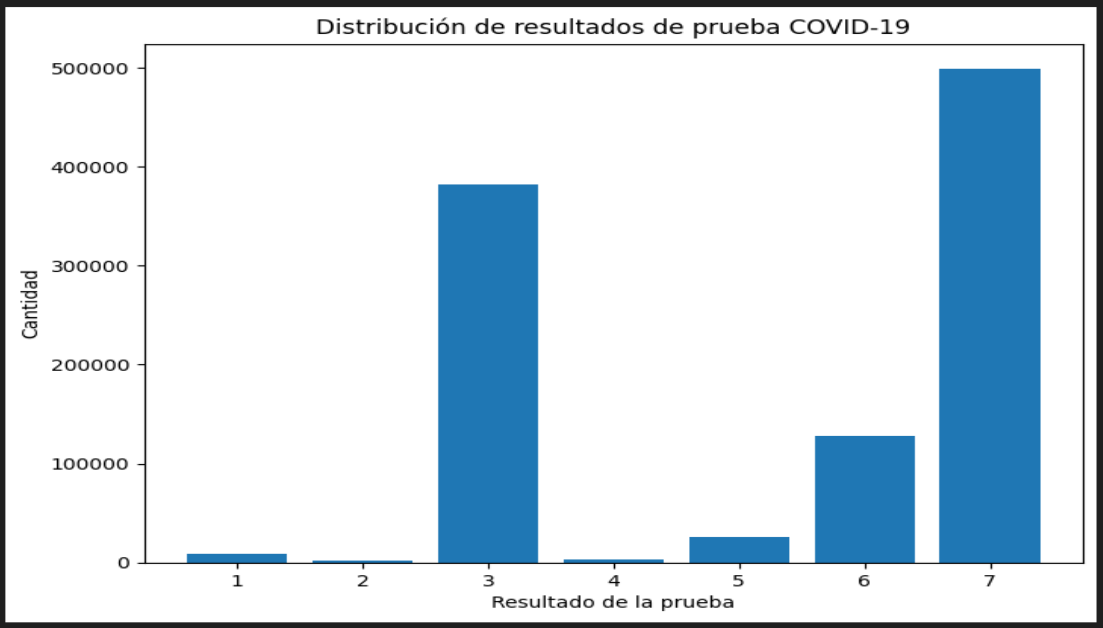
```python
# Gráfico de barras apiladas para las columnas 'DIABETES' y 'OBESITY'
diabetes_obesity_counts = df.groupby(['DIABETES', 'OBESITY']).size().unstack()
diabetes_obesity_counts.plot(kind='bar', stacked=True, figsize=(8, 6))
plt.xlabel('Diabetes')
plt.ylabel('Cantidad')
plt.title('Distribución de diabetes y obesidad')
plt.xticks([0, 1], ['No', 'Sí'])
plt.legend(['No obeso', 'Obeso'])
plt.show()
```

✓ 0.2s



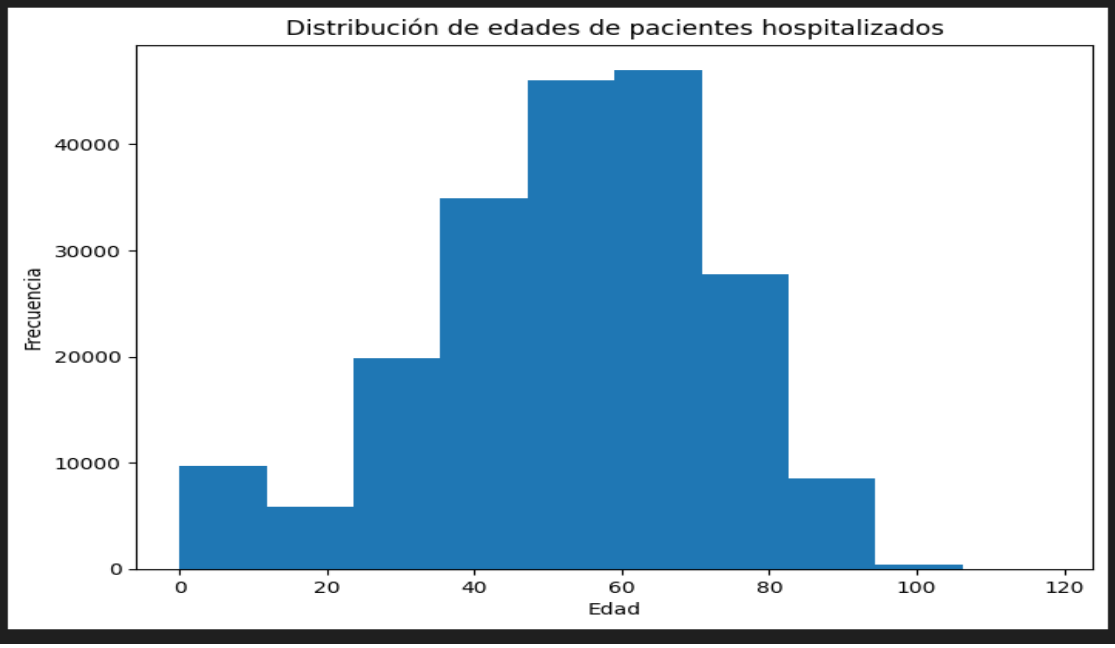

```python
# Gráfico de barras para la columna 'CLASIFFICATION_FINAL'
classification_counts = df['CLASIFFICATION_FINAL'].value_counts()
plt.figure(figsize=(8, 6))
plt.bar(classification_counts.index, classification_counts.values)
plt.xlabel('Resultado de la prueba')
plt.ylabel('Cantidad')
plt.title('Distribución de resultados de prueba COVID-19')
plt.show()
```

✓ 0.1s



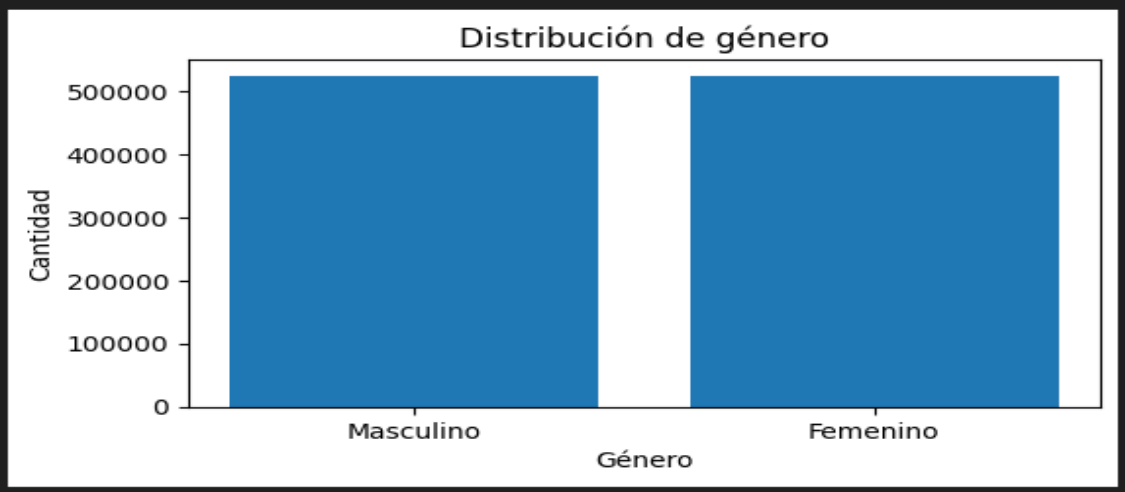
```python
# Filtrar los pacientes hospitalizados
hospitalized_patients = df[df['PATIENT_TYPE'] == 0]

# Gráfico de histograma para las edades de los pacientes hospitalizados
plt.figure(figsize=(8, 6))
plt.hist(hospitalized_patients['AGE'], bins=10)
plt.xlabel('Edad')
plt.ylabel('Frecuencia')
plt.title('Distribución de edades de pacientes hospitalizados')
plt.show()
```

[29] ✓ 0.1s




```python
# Gráfico de barras para la columna 'SEX'
sex_counts = df['SEX'].value_counts()
plt.figure(figsize=(6, 3))
plt.bar(sex_counts.index, sex_counts.values)
plt.xlabel('Género')
plt.ylabel('Cantidad')
plt.title('Distribución de género')
plt.xticks(sex_counts.index, ['Femenino', 'Masculino'])
plt.show()
```

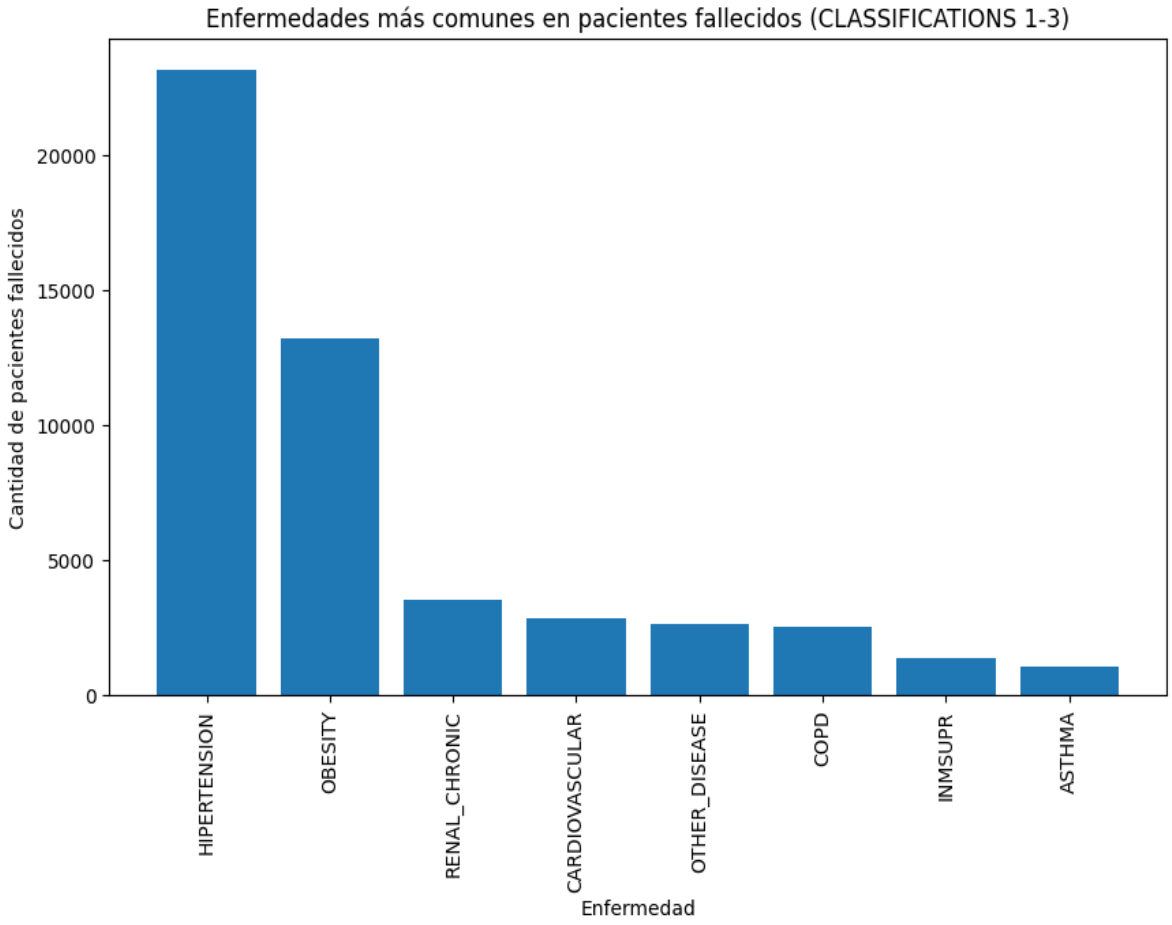
✓ 0.1s



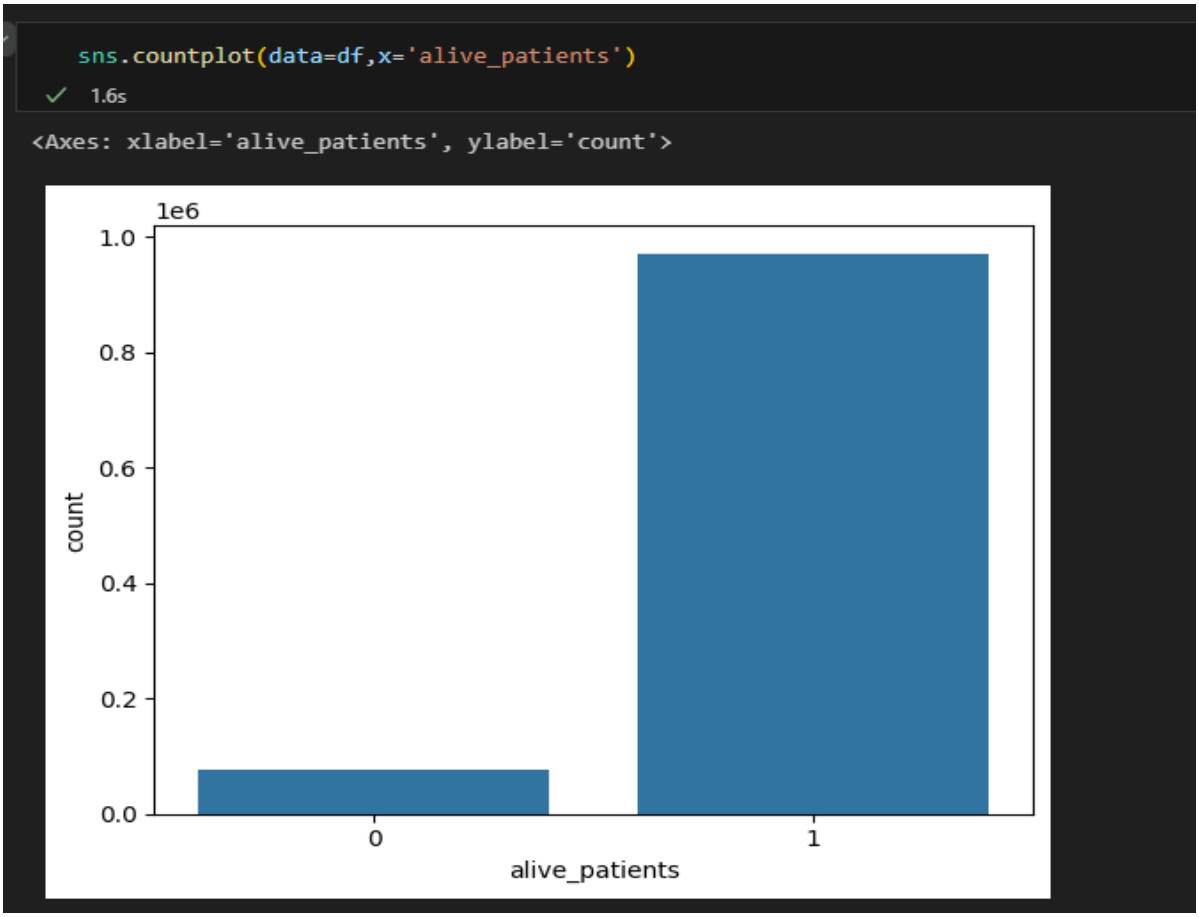
3. Análisis de esas Columnas

- Sacamos la conclusión de que hay más complicaciones con niños que adolescentes y los adultos. Cuanto mayor es la edad más frecuente la hospitalización.
- Hay una gran cantidad de los pacientes que tenían diabetes, también tenían obesidad.
- Pudimos comprobar la compensación en del género.
- La mayoría de los pacientes están entre los 25 y los 60.
- Logramos ver la cantidad de pacientes que se han regresado a casa y cuantos están hospitalizados.
- La mayoría de pacientes no tiene un test de COVID o no tiene COVID.

Ahora vamos analizar las demás columnas que tiene correlaciones con alive_patients.



Aquí podemos ver las relaciones entre todas esas columnas, por lo se decide incluir a todas las demás columnas del dataset, ya que no cuenta con nulos, solo con valores booleanos y que tienen importancia o relación con la predicción que queremos hacer.



Con este gráfico entendemos que hay un claro desbalance. Pero como esto es un proyecto con poco tiempo para realizar, se usará **SMOTE,** una técnica no vista en clase, pero que se estudió previamente. Vamos a hacer un equilibrio con todas las columnas sin excluir ninguna, aunque ya logramos que todas las columnas sean importantes en la predicción.

```python
from imblearn.over_sampling import SMOTE

# Separar las características y la variable objetivo
X = df.drop('alive_patients', axis=1)
y = df['alive_patients']

# Aplicar la técnica de sobremuestreo SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Crear un nuevo DataFrame con los datos sobremuestreados
oversampled_df = pd.DataFrame(X_resampled, columns=X.columns)
oversampled_df['alive_patients'] = y_resampled
```

✓ 1m 22.4s

```python
value_counts = oversampled_df["alive_patients"].value_counts()
print(value_counts)
```

✓ 0.0s

```
alive_patients
0    971633
1    971633
Name: count, dtype: int64
```

Ya tenemos un balance, con un nuevo dataframe, el cual usaremos.

```python
oversampled_df.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1943266 entries, 0 to 1943265
Data columns (total 18 columns):
 #   Column                Dtype
---  ------                -----
 0   USMER                 int64
 1   MEDICAL_UNIT          int64
 2   SEX                   int64
 3   PATIENT_TYPE          int64
 4   PNEUMONIA             int64
 5   AGE                   int64
 6   DIABETES              int64
 7   COPD                  int64
 8   ASTHMA                int64
 9   INMSUPR               int64
 10  HIPERTENSION          int64
 11  OTHER_DISEASE         int64
 12  CARDIOVASCULAR        int64
 13  OBESITY               int64
 14  RENAL_CHRONIC         int64
 15  TOBACCO               int64
 16  CLASIFFICATION_FINAL  int64
 17  alive_patients        int64
dtypes: int64(18)
memory usage: 266.9 MB
```

Entrenamiento

Vamos usar el Árbol de decisiones y vamos a usar su respectiva librería. Creamos una función llamada **show_metrics** que encapsula el proceso de calcular y visualizar las métricas clave para la evaluación.

También importamos el **classification_report** para proporciona un resumen detallado del rendimiento por clase y la matriz de confusión (confusion_matrix()) como es de costumbre.

Resultados

```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(oversampled_df.drop('alive_patients', axis=1), oversampled_df['alive_patients'], test_size=0.3,random_state=42 )

def show_metrics(clf, y_test, y_pred):
    print(f'Accuracy score: {int(accuracy_score(y_test, y_pred)*100)}%\n')
    print(classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                                  display_labels=clf.classes_)
    disp.plot()
    plt.show()
```

✓ 0.0s

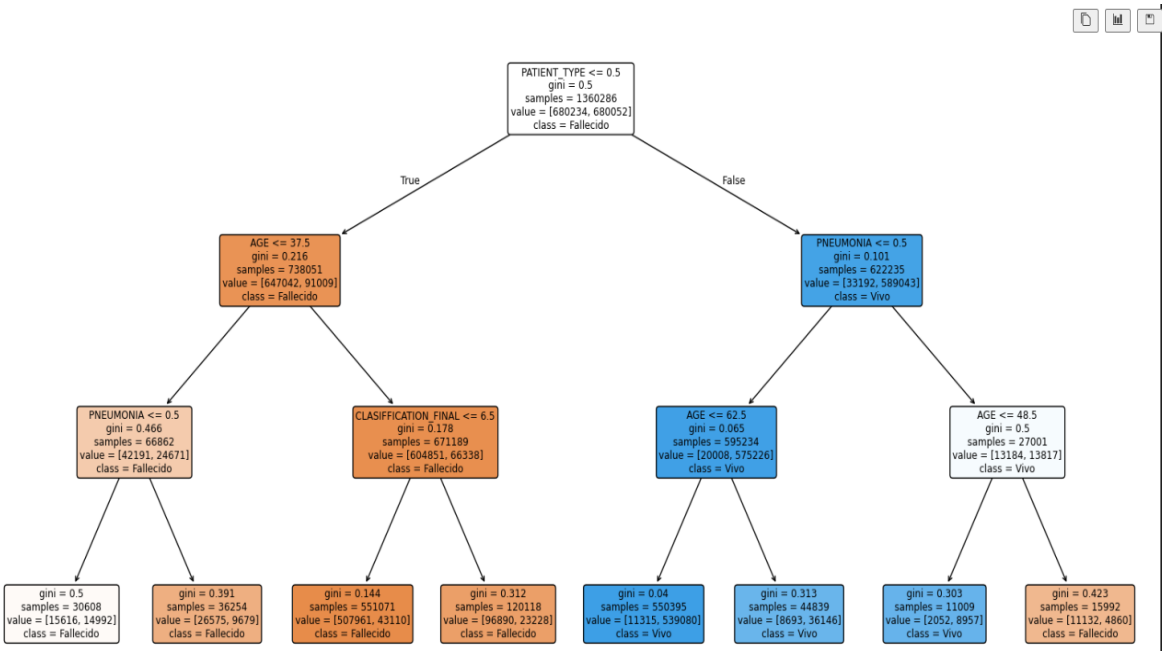
Con este fragmento de código, lo que haremos es ya mostrar tanto una Matriz de Confusión para ver los aciertos y errores que puedan surgir con nuestro código, esperando siempre obtener los resultados esperados.

```
from sklearn.tree import DecisionTreeClassifier

decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_test)
show_metrics(decision_tree,y_test,y_pred)
```

✓ 8.2s

```
Accuracy score: 93%

              precision    recall  f1-score   support

           0       0.91      0.97      0.94    291399
           1       0.97      0.91      0.94    291581

    accuracy                           0.94    582980
   macro avg       0.94      0.94      0.94    582980
```

Nuestra Matriz de Confusión está lista, y nos muestra estos resultados.



Aquí podemos observar el árbol de decisiones del modelo.

Resultados del Entrenamiento

El modelo de Árbol de Decisión es altamente efectivo y robusto para la tarea de predicción. Su alto rendimiento en todas las métricas, incluyendo la precisión, sensibilidad y la matriz de confusión, sugiere que ha aprendido a diferenciar de manera exitosa entre los pacientes que sobrevivirán y los que no. La técnica SMOTE ayudó a mitigar el problema del desbalance de clases, lo que se refleja en la capacidad del modelo para predecir ambas clases con un rendimiento similar. Aunque el número de falsos negativos es significativo (pacientes que se predijeron vivos, pero fallecieron), el modelo ofrece una base sólida y confiable para futuros análisis o aplicaciones.

El modelo logró una precisión del 93%, lo que significa que acertó en 93 de cada 100 predicciones que hizo sobre los datos de prueba.

Conclusión

La implementación de modelos de inteligencia artificial en el ámbito de la salud ofrece una gran oportunidad para apoyar los procesos de análisis y pronóstico clínico. A través de este proyecto, se logró entrenar un modelo que, con base en los datos clínicos de pacientes diagnosticados con COVID-19, es capaz de predecir si el paciente se encuentra vivo o ha fallecido, con un alto nivel de precisión. Este tipo de modelos puede ser útil para priorizar la atención médica y comprender mejor los factores de riesgo asociados a la mortalidad.

Aunque el enfoque es académico y no clínico, los resultados demuestran el potencial de la IA como herramienta de análisis complementario y de apoyo. El trabajo realizado por los tres integrantes del equipo permitió integrar conocimientos de programación, análisis de datos y aprendizaje automático, aplicados a un problema crítico, actual y de gran relevancia social. Se concluye que, con los datos adecuados y una metodología bien definida, es posible obtener y generar modelos predictivos funcionales y valiosos en el área de la salud en corto tiempo y con recursos accesibles.

Bibliografía

Hasell, J., Mathieu, E., Beltekian, D., et al. (2020). A cross-country database of COVID-19 testing. *Scientific Data, 7*(345). https://doi.org/10.1038/s41597-020-00688-8

Johns Hopkins University. (2021). *COVID-19 dashboard by the Center for Systems Science and Engineering (CSSE).* https://coronavirus.jhu.edu/map.html

Mathieu, E., Ritchie, H., Ortiz-Ospina, E., et al. (2021). A global database of COVID-19 vaccinations. *Nature Human Behaviour, 5*, 947–953. https://doi.org/10.1038/s41562-021-01122-8

Our World in Data. (2024). *COVID-19 dataset.* GitHub. https://github.com/owid/COVID-19-data

Takbir Alam. (s.f.). *COVID-19 patients symptoms dataset.* Kaggle. https://www.kaggle.com/datasets/takbiralam/covid19-symptoms-dataset

World Health Organization. (2020). *Coronavirus disease (COVID-19) pandemic.* https://www.who.int/emergencies/diseases/novel-coronavirus-2019