PROYECTO TEAM 20

ANÁLISIS EXPLORATORIO Y LIMPIEZA DE DATOS DE PACIENTES CON COVID-19: UN ENFOQUE EN CONDICIONES PREEXISTENTES Y RESULTADOS CLÍNICOS

- DE LA CRUZ MUNGUIA ARELY
- PÉREZ MENDOZA LEISLY
- VALLE NÚÑEZ GABRIELA

RESUMEN DE CONTENIDOS

INTRODUCCIÓN

OBJETIVOS

PLANTEAMIENTO DEL PROBLEMA

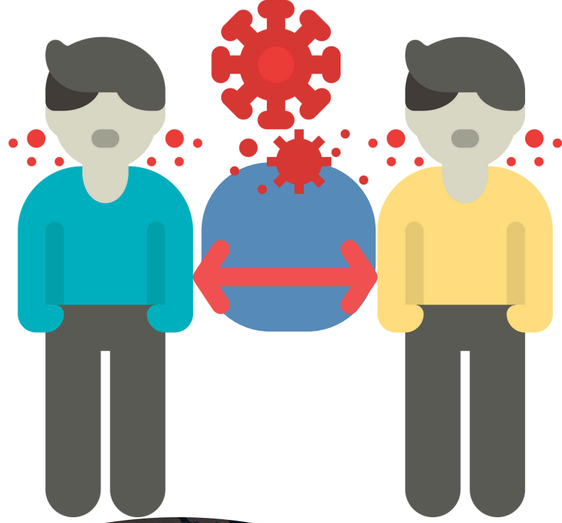
DATASET

ANÁLISIS DE DATOS

CONCLUSIÓN

El COVID-19, originado por el SARS-CoV-2, causó una crisis global que desbordó sistemas de salud y aumentó desigualdades. Este proyecto analiza datos de más de un millón de pacientes en México diagnosticados en 2020, para identificar cómo factores demográficos y clínicos influyen en la gravedad de la enfermedad.

A través de la limpieza de datos y análisis exploratorios, se busca sentar las bases para investigaciones que ayuden a predecir riesgos y desenlaces clínicos.



INTRODUCCIÓN

OBJETIVOS

OBJETIVO GENERAL

El objetivo general de este proyecto es investigar cómo las condiciones de salud preexistentes influyen en la probabilidad de mortalidad y contagio en pacientes con COVID-19 y qué factores de riesgo están asociados a la hospitalización. Esto permitirá identificar patrones y factores críticos que pueden guiar las políticas de salud pública y las estrategias de intervención.

OBJETIVOS ESPECÍFICOS

Evaluar la relación entre las comorbilidades y el desenlace del paciente (alta o defunción) para determinar si ciertas condiciones preexistentes influyen en los resultados de los pacientes con COVID-19.

Limpiar y organizar las columnas seleccionadas para garantizar que los datos estén listos para su análisis.

Investigar diferencias en la tasa de mortalidad entre diferentes grupos de edad y género.

PLANTEAMIENTO DEL PROBLEMA

El problema es la falta de comprensión sobre cómo comorbilidades como diabetes e hipertensión, además del sexo y la edad afectan el riesgo de complicaciones graves en pacientes con COVID-19 en México. Se requiere analizar estos datos para clasificar a los pacientes en grupos de riesgo y mejorar la atención médica y las estrategias preventivas.

DATASET

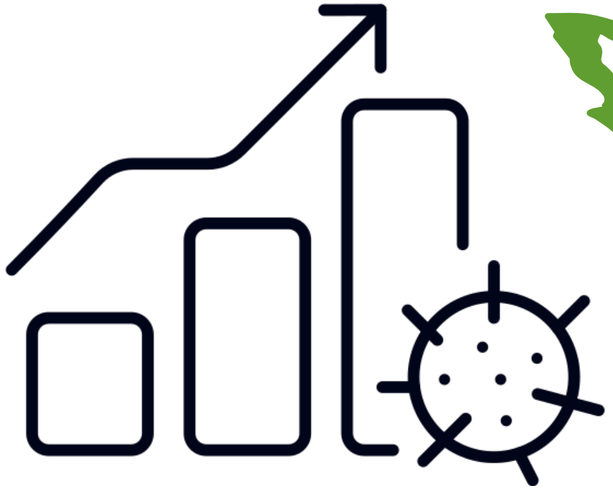
- En las características Booleanas:
  - 1 significa "sí".
  - 2 significa "no".
  - Los valores 97, 98 y 99 representan datos faltantes.

sex	int64
patient_type	int64
date_died	object
intubed	int64
pneumonia	int64
age	int64
pregnant	int64
diabetes	int64
copd	int64
asthma	int64

inmsupr	int64
hipertension	int64
other_disease	int64
cardiovascular	int64
obesity	int64
renal_chronic	int64
tobacco	int64
clasiffication_final	int64
icu	int64

- El dataset tiene por nombre “COVID-19 Dataset”, este fue recuperado de la plataforma de kaggle en el siguiente link: https://www.kaggle.com/datasets/meirnizri/covid19-dataset

- El dataset fue proporcionado por el gobierno mexicano. Contiene una gran cantidad de información anonimizada sobre pacientes, incluyendo condiciones preexistentes. El dataset crudo consta de 21 características únicas y 1,048,575 pacientes únicos.



ANÁLISIS DE DATOS

Para la limpieza, resolución y análisis exploratorio de los datos decidimos hacer uso del archivo csv para trabajarlo dentro de Jupyter Notebook. Tal como vimos previamente el archivo cuenta con 21 columnas y 1,048,575. La lectura y carga de datos se hizo a través de un bloque Try-Exception para cubrir posibles errores.

```
try:
    # Cargamos el archivo, para su principal lectura y procesamiento de datos.
    df = pd.read_csv(dataset + 'Covid Data.csv')
    display(df)
except Exception as e:
    print("Algo ocurrio con la lectura del archivo verifica la ruta")
    print(f'Error: {e}')
else:
    print("Lectura finalizada")
```

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTENSION	OTHER_DISEASE	CARDIOVASCULAR	OBESITY	RENAL_CHRONIC	TOBACCO	CLASIFFICATION_FINAL	ICU
0	2	1	1	1	03/05/2020	97	1	65	2	2	...	2	2	1	2	2	2	2	2	3	97
1	2	1	2	1	03/06/2020	97	1	72	97	2	...	2	2	1	2	2	1	1	2	5	97
2	2	1	2	2	09/06/2020	1	2	55	97	1	...	2	2	2	2	2	2	2	2	3	2
3	2	1	1	1	12/06/2020	97	2	53	2	2	...	2	2	2	2	2	2	2	2	7	97
4	2	1	2	1	21/06/2020	97	2	68	97	1	...	2	2	1	2	2	2	2	2	3	97
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048570	2	13	2	1	9999-99-99	97	2	40	97	2	...	2	2	2	2	2	2	2	2	7	97
1048571	1	13	2	2	9999-99-99	2	2	51	97	2	...	2	2	1	2	2	2	2	2	7	2
1048572	2	13	2	1	9999-99-99	97	2	55	97	2	...	2	2	2	2	2	2	2	2	7	97
1048573	2	13	2	1	9999-99-99	97	2	28	97	2	...	2	2	2	2	2	2	2	2	7	97
1048574	2	13	2	1	9999-99-99	97	2	52	97	2	...	2	2	2	2	2	2	2	2	7	97

1048575 rows × 21 columns

```
Lectura finalizada
```

CARACTERISTICAS DE LOS DATOS

Debido a las caracteristicas del dataset, en un principio no encontramos valores nulos ya que recibieron un tratamiento específico rellenandolos con valores especificos, en cuanto al tipo de datos la mayoria se maneja como int y solo tenemos un Object, pero para el analisis adecuado hicimos algunos cambios.

```
Tipos de datos por columna:
USMER                    int64
MEDICAL_UNIT             int64
SEX                      int64
PATIENT_TYPE             int64
DATE_DIED               object
INTUBED                  int64
PNEUMONIA                int64
AGE                      int64
PREGNANT                 int64
DIABETES                 int64
COPD                     int64
ASTHMA                   int64
INMSUPR                  int64
HIPERTENSION             int64
OTHER_DISEASE            int64
CARDIOVASCULAR           int64
OBESITY                  int64
RENAL_CHRONIC            int64
TOBACCO                  int64
CLASIFFICATION_FINAL     int64
ICU                      int64
dtype: object
```

```
Cantidad de datos nulos por columna:
                        Nulos  Porcentaje (%)
USMER                       0             0.0
MEDICAL_UNIT                0             0.0
SEX                         0             0.0
PATIENT_TYPE                0             0.0
DATE_DIED                   0             0.0
INTUBED                     0             0.0
PNEUMONIA                   0             0.0
AGE                         0             0.0
PREGNANT                    0             0.0
DIABETES                    0             0.0
COPD                        0             0.0
ASTHMA                      0             0.0
INMSUPR                     0             0.0
HIPERTENSION                0             0.0
OTHER_DISEASE               0             0.0
CARDIOVASCULAR              0             0.0
OBESITY                     0             0.0
RENAL_CHRONIC               0             0.0
TOBACCO                     0             0.0
CLASIFFICATION_FINAL        0             0.0
ICU                         0             0.0
```

LIMPIEZA Y TRANSFORMACIÓN

Se cambio el nombre de las variables para seguir la nomenclatura snake_case

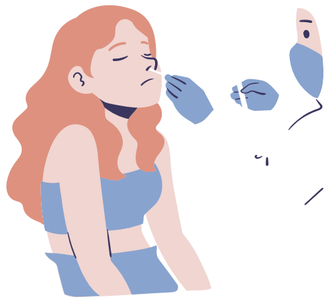
USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTENSION	OTHER_DISEASE	CARDIOVASCULAR	OBESITY	RENAL_CHRONIC	TOBACCO	CLASIFFICATION_FINAL	ICU

```python
# Convertir nombres de columnas a snake_case
df_1.columns = df.columns.str.lower().str.replace(' ', '_')
df_1.head()
```

usmer	medical_unit	sex	patient_type	date_died	intubed	pneumonia	age	pregnant	diabetes	...	asthma	inmsupr	hipertension	other_disease	cardiovascular	obesity	renal_chronic	tobacco	clasiffication_final	icu

```python
# Reemplazar los valores 97,98 y 99 por NaN en todas las columnas
df_1.replace([97, 98, 99], np.nan, inplace=True)

# En el análisis 0 tiene sentido para los datos faltantes
df_1.fillna(0, inplace=True)

# Reemplazar '9999-99-99' con NaT antes de convertir
df_1['date_died'].replace('9999-99-99', np.nan, inplace=True)
#Validar que los datos manejados solo sean de tipo date
df_1['date_died'] = pd.to_datetime(df_1['date_died'], errors='coerce')

datos_nulos={
    'date_died': pd.to_datetime(np.random.uniform(df_1['date_died'].min().value, df_1['date_died'].max().value)).normalize()
}

df_1 = df_1.astype(diccionario_de_conversion)
df_1.fillna(datos_nulos,inplace=True)

#explorando el dataset
resumen_dataframe(df_1)
```

En cuanto a los valores como 97,98 y 99 se transformaron en NaN, para posteriormente transformarlos en 0 para tener un menor sesgo en el analisis posterior, para el caso de las fechas igualmente los valores pro default se pasaron a NaN, se transformaron en tipo date time y finalmente los valores nulos se rellenaron con fechas random.



LIMPIEZA Y TRANSFORMACIÓN

Pudimos notar que habian datos duplicados por lo que se hizo uso de el borrado de duplicados quedando unicamente 213,912 datos

```
# Detectar valores duplicados
df_1.drop_duplicates(inplace=True)
resumen_dataframe(df_1)
```

```
============================================================
RESUMEN GENERAL DEL DATAFRAME
============================================================
Dimensiones del DataFrame: 213912 filas y 21 columnas
------------------------------------------------------------
```

Se eliminaron las variables usmr y medical_unit ya que dentro de nuestro enfoque no se encontro relevancia de mantener esta información.

```
[112] #Eliminar columnas no necesarias del DataFrame y guardarlo en una nueva variable
      df_1 = df_1.drop(columns=['usmer', 'medical_unit'])
      df_1.head(10)
```

	sex	patient_type	date_died	intubed	pneumonia	age	pregnant	diabetes	copd	asthma	inmsupr	hipertension	other_disease	cardiovascular	obesity	renal_chronic	tobacco	clasiffication_final	icu

La mayoria de las variables describian situaciones que podian ser mapeadas para una mejor comprensión y analisis, en el caso de la clasificación del diagnostico la variable se mapeo con valores como Positivo-leve, moderado y grave o desconocido segun el valor asociado. En cuanto al sexo se mapeo a femenino y masculino y el tipo de paciente a valores como alta y hospitalización.

```
# Mapear la variable clasificación
def map_classification(x):
  if x >= 4:
    return 'Negativo o Inconcluso'
  elif x == 1:
    return 'Positivo - Leve'
  elif x == 2:
    return 'Positivo - Moderado'
  elif x == 3:
    return 'Positivo - Grave'  # No cambia el valor 3
  return x

df_1['clasiffication_final'] = df_1['clasiffication_final'].apply(map_classification)
```

```
def mapear_variables(df):
    # Mapeo para la variable 'sex'
    if df['sex'].dtype == 'int64':  # Verifica si los valores son numéricos
        df['sex'] = df['sex'].map({1: 'Femenino', 2: 'Masculino'})

    # Mapeo para la variable 'patient_type'
    if df['patient_type'].dtype == 'int64':  # Verifica si los valores son numéricos
        df['patient_type'] = df['patient_type'].map({1: 'Alta', 2: 'Hospitalizado'})

    return df

# Aplicar la función al DataFrame solo una vez
df_1 = mapear_variables(df_1)
df_1.head()

```

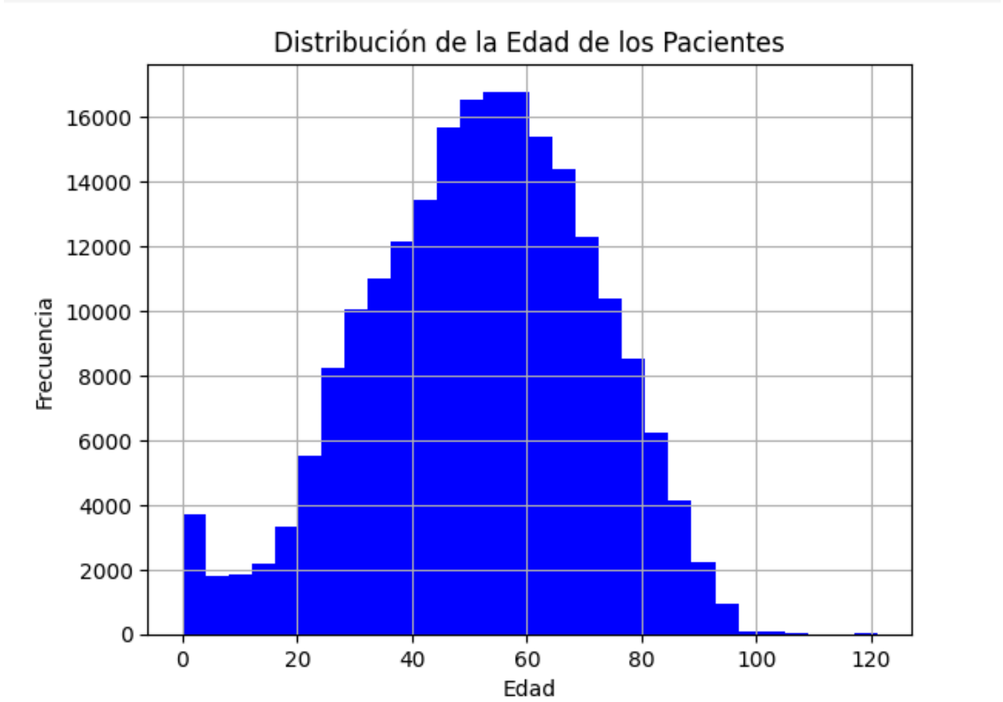
LIMPIEZA Y TRANSFORMACIÓN

Finalmente las variables que solo representaban si padecia o no dicha condición se mapeo a si, no y desconocido si tenia un valor 0. De esta forma podemos interpretar y comprender mejor la información sin tener que consultar continuamente cadauna de las características y entender que representa cada número.


```python
def mapeo_int(df):
# Mapeo para las variables binarias (1 = Si, 2 = No)
  binarias = ['intubed','pneumonia','pregnant','diabetes','copd','asthma','inmsupr','hipertension',
              'other_disease','cardiovascular','obesity','renal_chronic','tobacco']

  for col in binarias:
      df[col] = pd.to_numeric(df[col], errors='coerce')  # Convierte a numérico, reemplaza errores con NaN
      df[col] = df[col].fillna(0).astype(int)  # Reemplaza NaN con 0 y convierte a int
      df[col] = df[col].map({0: 'Desconocido', 1: 'Sí', 2: 'No'})

  return df
  # Aplicar la función al DataFrame
df_1 = mapeo_int(df_1)
df_1.head()
```

OBTENIENDO RESPUESTAS

Para poder entender los datos que limpiamos hicimos algunas preguntas que respondimos através de diversas exploraciones. Un factor interesante era poder comprender las edades de los pacientes que se encontraban en el dataset obtenido, pudimos notar que habia pacientes desde 0 años (niñ@s de meses) hasta una persona de 121 años.




```
[119]  1 #funcion que calcula las estadisticas de las columnas.
       2 def calcular_estadisticas(df, columna):
       3     print(f"Estadísticas de la columna {columna}:")
       4     print(f"Media: {df[columna].mean()}")
       5     print(f"Desviación estándar: {df[columna].std()}")
       6     print(f"Mínimo: {df[columna].min()}")
       7     print(f"Máximo: {df[columna].max()}")
       8
       9 calcular_estadisticas(df_1, 'age')
      10
```

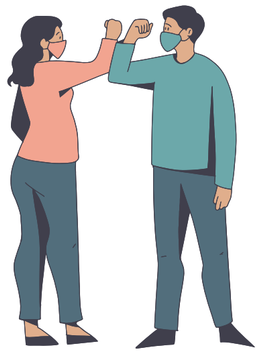
```
Estadísticas de la columna age:
Media: 51.78696847301694
Desviación estándar: 19.63103931349849
Mínimo: 0
Máximo: 121
```

La media se concentro en los 51-52 años por lo que a partir de esto pudimos notar que este podria ser la parte de la población que podria sufrir más riesgo de contagio.

OBTENIENDO RESPUESTAS

Otro factor interesante era saber cuál era el porcentaje del sexo femenino y masculino que habia presente en los datos, para esto agrupamos los datos dependiendo el sexo y pudimos notar que el mayor porcentaje era masculino

	Sexo	Conteo	Proporción (%)
0	Femenino	99765	46.638337
1	Masculino	114147	53.361663



Dentro del datasdet habia diferentes grados de covid que iban desde leve hasta grave y descubrimos que la mayoria de pacientes tenia covid de forma grave

	Diagnóstico	Cantidad de Pacientes
0	Positivo - Leve	4672
1	Positivo - Moderado	1787
2	Positivo - Grave	89483

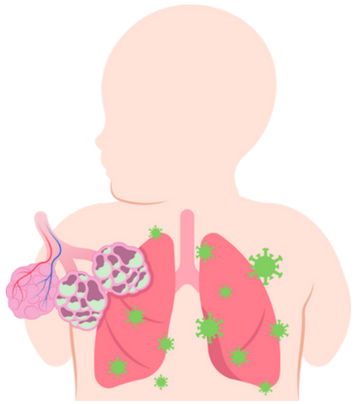
	Tipo de Atención	Clasificación de COVID	Cantidad de Pacientes
0	Alta	Negativo o Inconcluso	62470
1	Alta	Positivo - Grave	32112
2	Alta	Positivo - Leve	2789
3	Alta	Positivo - Moderado	166
4	Hospitalizado	Negativo o Inconcluso	55500
5	Hospitalizado	Positivo - Grave	57371
6	Hospitalizado	Positivo - Leve	1883
7	Hospitalizado	Positivo - Moderado	1621

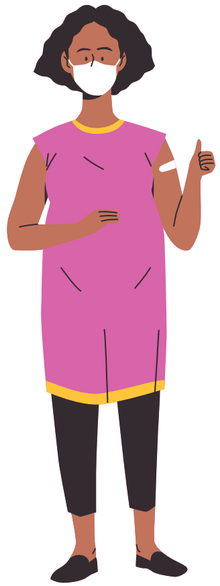
De esas personas con diferentes diagnosticos fueron o dadas de alta o hospitalizados segun su diagnóstico

	Tipo de Atención	Cantidad de Pacientes
0	Hospitalizado	116375
1	Alta	97537

OBTENIENDO RESPUESTAS

Un 50% de los contagiados ya tenian como padecimiento neumonia y solo el 1.45% de las mujeres estaba embarazada.





```python
# Filtrar los pacientes diagnosticados con COVID (Positivo - Leve, Moderado, Grave)
covid_diagnosticados = df_1[df_1['clasiffication_final'].isin(['Positivo - Leve', 'Positivo - Moderado', 'Positivo - Grave'])]

# Calcular la cantidad de pacientes con neumonía dentro del grupo de diagnosticados con COVID
pacientes_con_neumonia = covid_diagnosticados[covid_diagnosticados['pneumonia'] == 'Sí'].shape[0]

# Calcular el total de pacientes diagnosticados con COVID
total_covid_diagnosticados = covid_diagnosticados.shape[0]

# Calcular el porcentaje
porcentaje_con_neumonia = (pacientes_con_neumonia / total_covid_diagnosticados) * 100

# Mostrar el resultado
print(f"Porcentaje de pacientes diagnosticados con COVID que ya tenían neumonía: {porcentaje_con_neumonia:.2f}%")

```

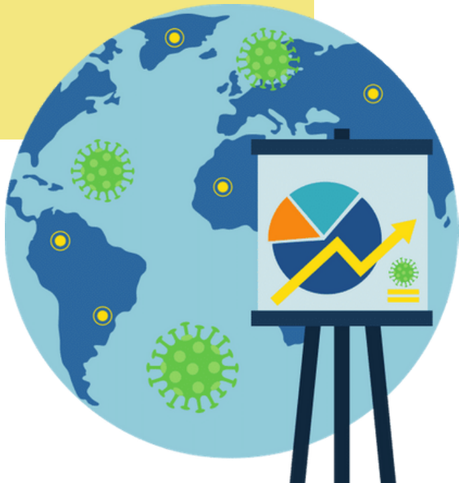
```
Porcentaje de pacientes diagnosticados con COVID que ya tenían neumonía: 50.43%
```

```python
# Calcular el número total de pacientes en el dataset
total_pacientes = df_1.shape[0]

# Calcular el número de pacientes embarazadas (asumiendo que 'Sí' indica embarazo)
pacientes_embarazadas = df_1[df_1['pregnant'] == 'Sí'].shape[0]

# Calcular el porcentaje de pacientes embarazadas
porcentaje_embarazadas = (pacientes_embarazadas / total_pacientes) * 100

# Mostrar el resultado utilizando display() para una mejor visualización
print(f"Porcentaje de pacientes embarazadas en el dataset: {porcentaje_embarazadas:.2f}%")

```

```
Porcentaje de pacientes embarazadas en el dataset: 1.41%
```

CONCLUSIÓN

Analizar cómo las condiciones preexistentes influyen en la mortalidad y hospitalización por COVID-19 aportará información clave para mejorar la respuesta a la pandemia, proteger a las poblaciones vulnerables y optimizar la gestión de recursos.




¡MUCHAS
GRACIAS!