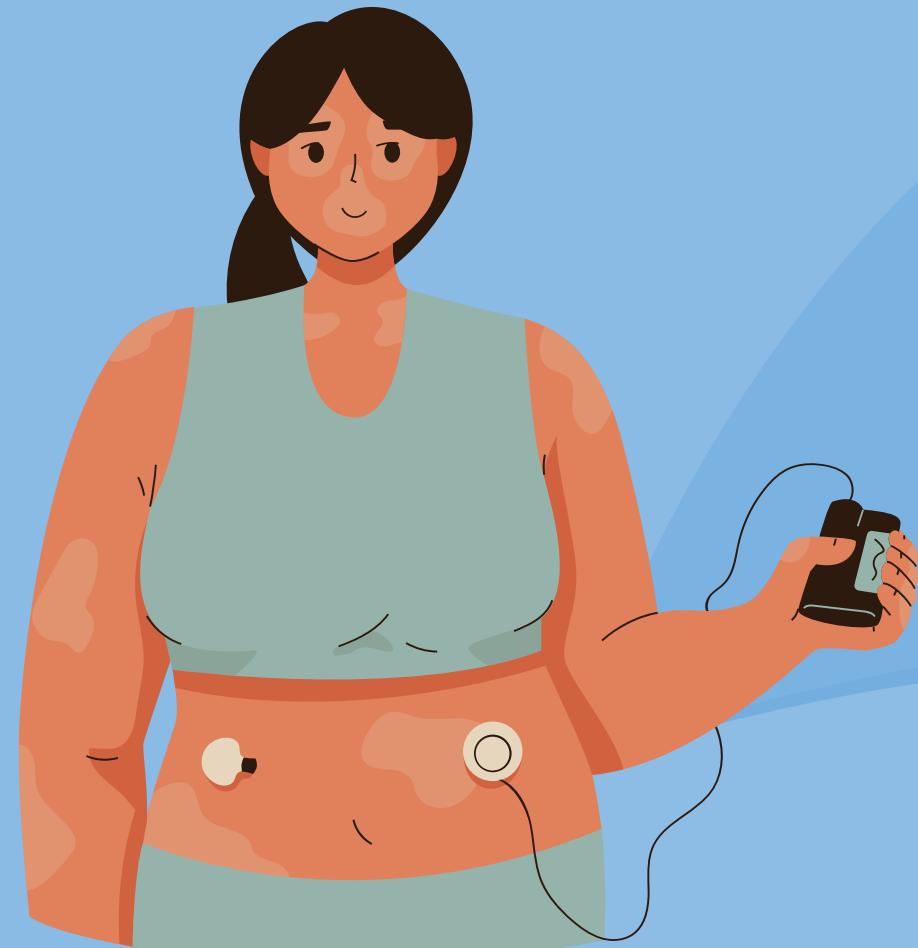
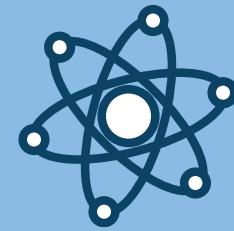


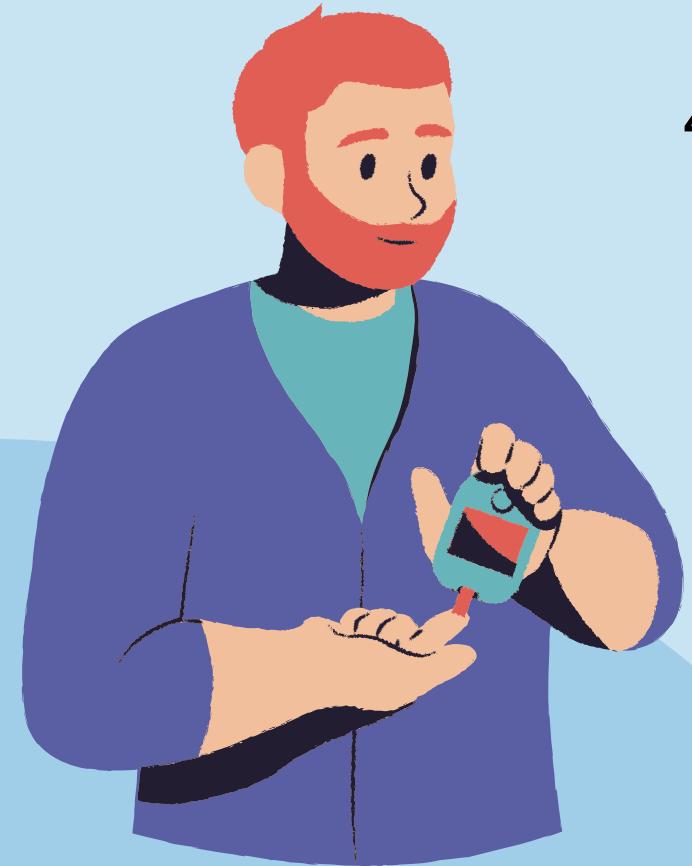
Equipa 13

ANÁLISIS DE DATOS CON PYTHON DIABETES



OBJETIVO

Usaremos herramientas estadísticas y de visualización para explorar los factores de salud asociados con diferentes tipos de diabetes. Aplicando análisis bidimensional , regresión y algoritmos de clasificación.



1. Importar Librerías Necesarias
pandas para manipulación de datos.
2. Cargar Datos desde un Archivo CSV
 - Muestra las primeras filas del DataFrame para verificar la carga correcta.
3. Análisis Exploratorio de Datos
 - Se imprimen los valores nulos en cada columna usando df.isnull().sum().
 - Se eliminan las filas duplicadas y con valores nulos para asegurar datos limpios.
4. Renombrar columnas
 - Para estandarizar el DataFrame, las columnas son renombradas. Esto asegura claridad y uniformidad al acceder a los datos.

```
# %% Importar librerías necesarias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from IPython.display import display

# Configuración de estilo para gráficos
sns.set_theme(style="whitegrid")

# 1. Cargar datos desde un archivo CSV
ruta_dataset = "./diabetes.csv"
df = pd.read_csv(ruta_dataset, encoding='latin-1', index_col=0)
display(df.head()) # Mostrar las primeras filas del DataFrame

# 2. Análisis exploratorio de datos
# Verificar valores faltantes
print("Valores faltantes por columna:")
print(df.isnull().sum())

# Eliminar filas y columnas con valores nulos y duplicados
df = df.dropna() # Elimina filas con valores nulos
df = df.drop_duplicates() # Elimina filas duplicadas
print("-" * 50)
display(df.columns)

# Renombrar columnas para uniformidad y claridad
df = df.rename(columns={
    'Marcadores genéticos': 'marcadores_geneticos',
    'Autoanticuerpos': 'autoanticuerpos',
    'Antecedentes familiares': 'antecedentes_familiares',
    'Factores ambientales': 'factores_ambientales',
    'Niveles de insulina': 'niveles_insulina',
    'Edad': 'edad',
    'IMC': 'imc',
    'Actividad física': 'actividad_fisica',
    'Abitos dietéticos': 'habitos_dieteticos',
    'Presión sanguínea': 'presion_sanguinea',
    'Niveles de colesterol': 'niveles_colesterol',
    'Origen étnico': 'origen_etnico',
    'Talla': 'talla',
    'Niveles de glucosa': 'niveles_glucosa',
    'Factores socioeconómicos': 'factores_socioeconomicos',
    'Tabaquismo': 'tabaquismo',
    'Consumo de alcohol': 'consumo_alcohol',
    'Tolerancia a la glucosa': 'tolerancia_glucosa',
    'Síndrome de ovario poliquístico': 'sindrome_ovario_poliquistico',
    'Diabetes gestacional': 'diabetes_gestacional',
    'Tipo de embarazo': 'tipo_embarazo',
    'Aumento de peso en el embarazo': 'aumento_peso_embarazo',
    'Salud pancreática': 'salud_pancreatica',
    'Niveles de encimas digestiva': 'niveles_de_encimas_digestiva',
    'Análisis de orina': 'analisis_orina',
    'Peso al nacer': 'peso_nacimiento',
    'Síntomas de inicio temprano': 'sintomas_de_inicio_temprano'
})
```

```

# Convertir la columna de gramos a kilogramos
df['peso_nacimiento'] = df['peso_nacimiento'] / 1000

# Verificar los primeros valores para asegurar que la conversión fue correcta
print(df['peso_nacimiento'].head())

# 3. Comprobamos el cambio y hacemos una copia para trabajar con el DataFrame
df.reset_index(inplace=True)
df.to_csv('db_diabetes.csv', index=False) # Para que la primera columna se unique
display(df)

# 4. Cambiar el índice a columna y arreglar una columna
df = df.rename(columns={'Tipo': 'tipo_diabetes'})
print(df.columns)
display(df)

# 5. Seleccionar solo columnas numéricas
datos_numericos = df.select_dtypes(include=['int64', 'float64'])

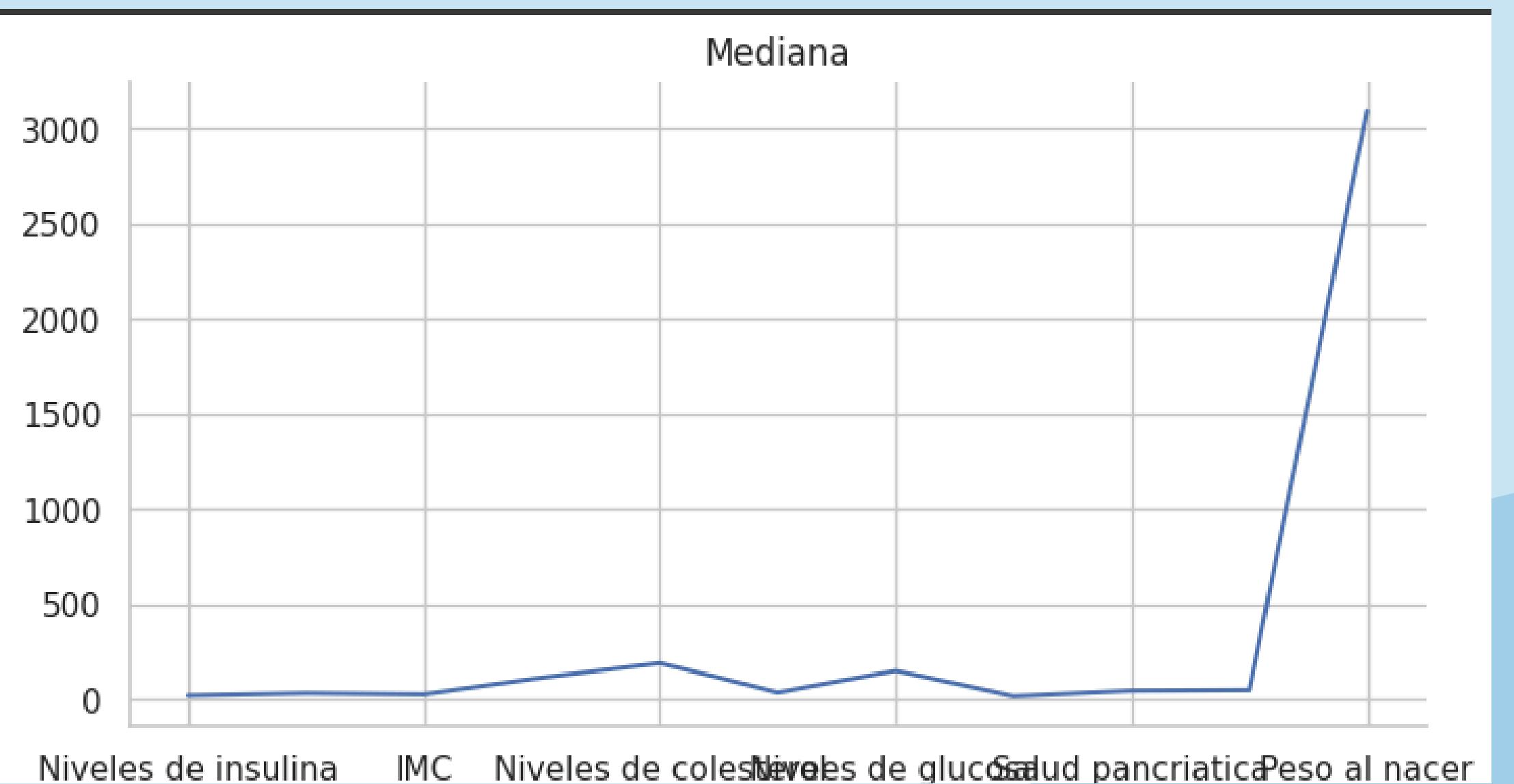
# 6. Calcular medidas de localización y variabilidad
valores_media = datos_numericos.mean()
valores_mediana = datos_numericos.median()
desviacion_estandar = datos_numericos.std()
iqr = datos_numericos.quantile(0.75) - datos_numericos.quantile(0.25)

# Crear un DataFrame con estadísticas descriptivas
estadisticas_descriptivas = pd.DataFrame({
    'Media': valores_media,
    'Mediana': valores_mediana,
    'Desviación Estándar': desviacion_estandar,
    'IQR': iqr
})
display("Estadísticas Descriptivas:")
display(estadisticas_descriptivas)

```

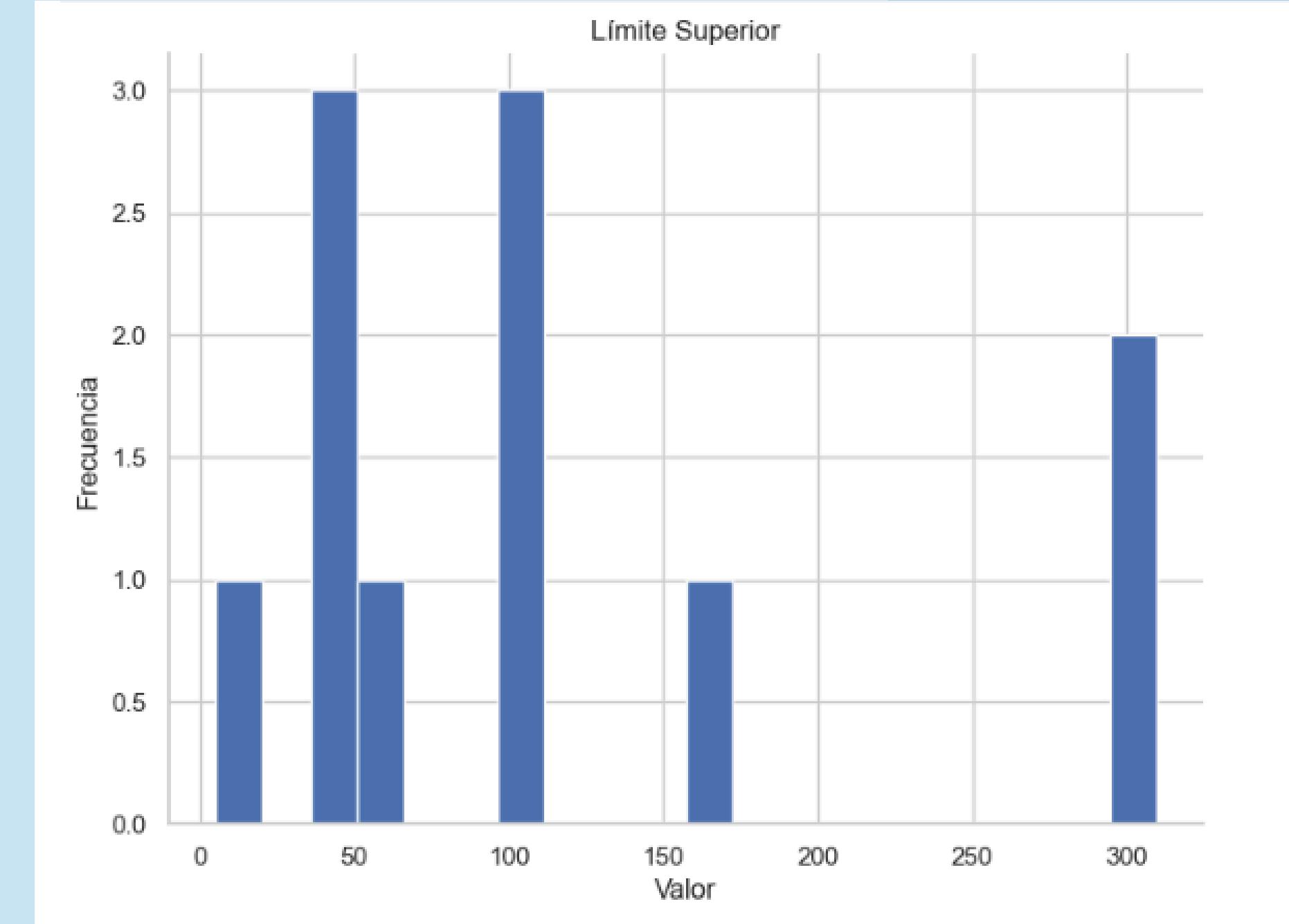


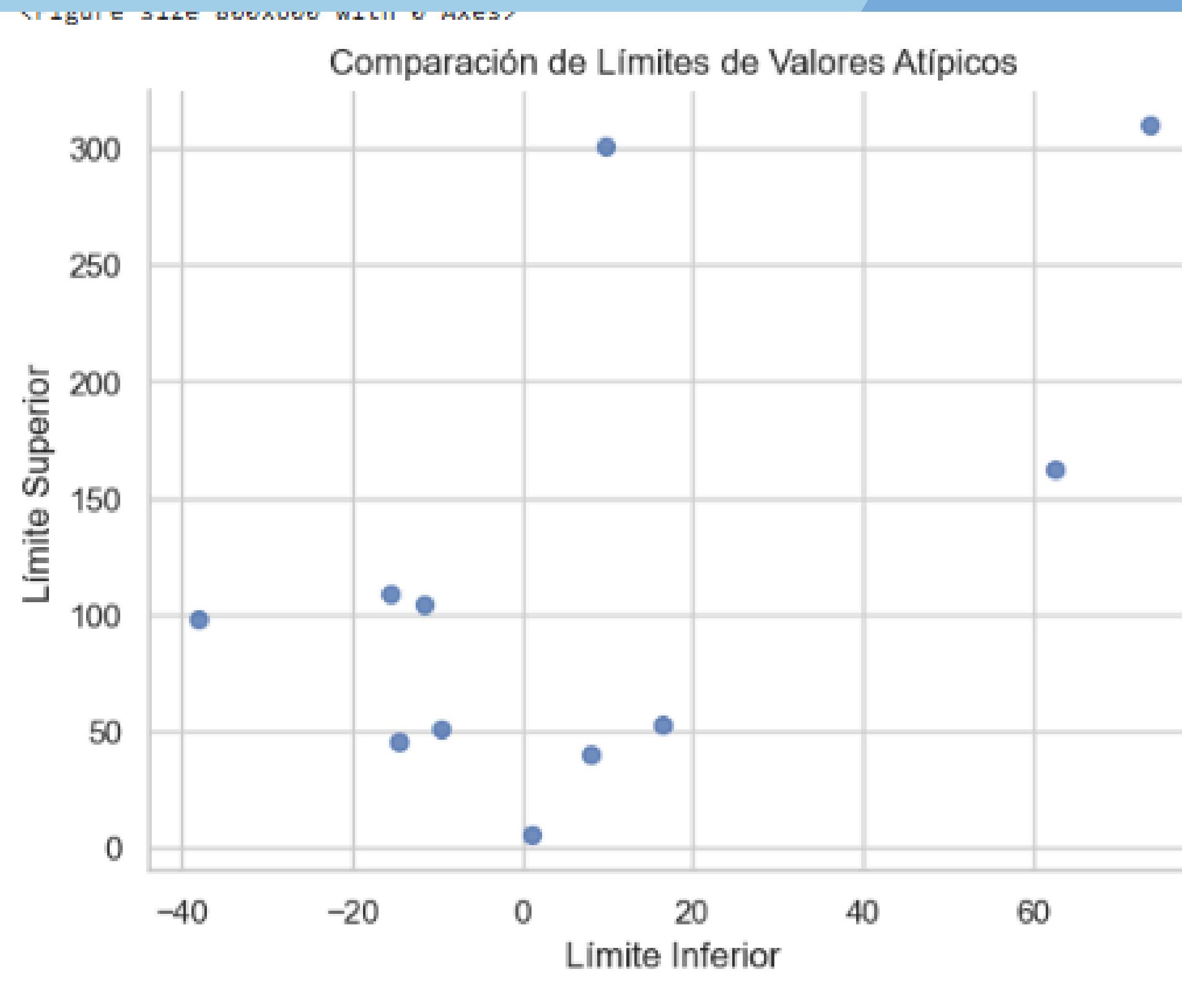
- Convertir Gramos a Kilogramos en la Columna peso_nacimiento
- Restablecer el índice y guardar el DataFrame
- Se calculan estadísticas descriptivas como la media , mediana, desviación estándar e IQR (rango intercuartílico).
- Estas estadísticas se muestran en un DataFrame.



Detección de Valores Atípicos

- Se calculan los límites superior e inferior para cada columna usando el IQR. Esto permite identificar valores atípicos.
- Los límites se almacenan y se muestran en un DataFrame.



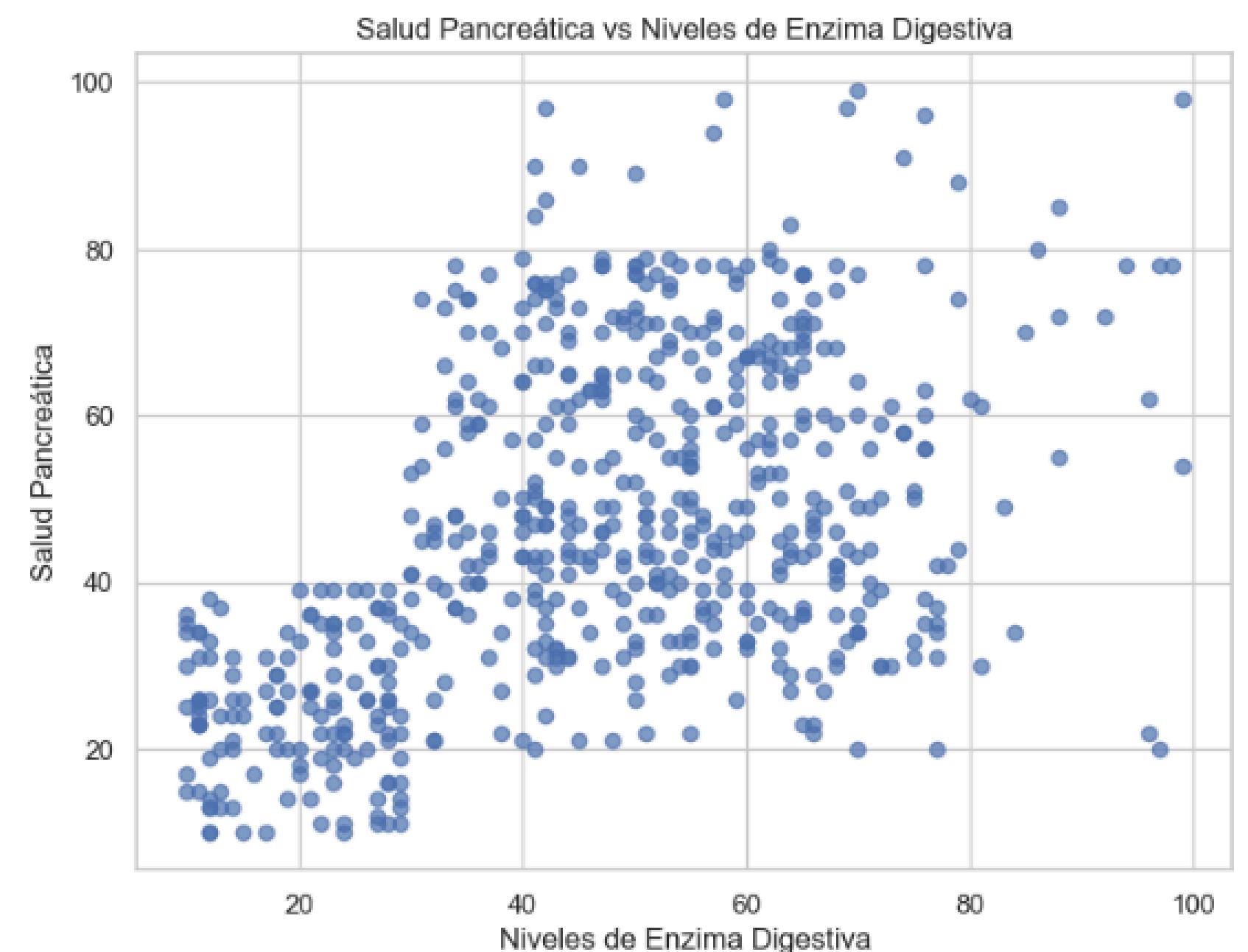


Valores Atípicos

- Cada punto corresponde a una columna numérica en nuestro conjunto de datos, mostrando los valores de los límites para identificar posibles valores atípicos.



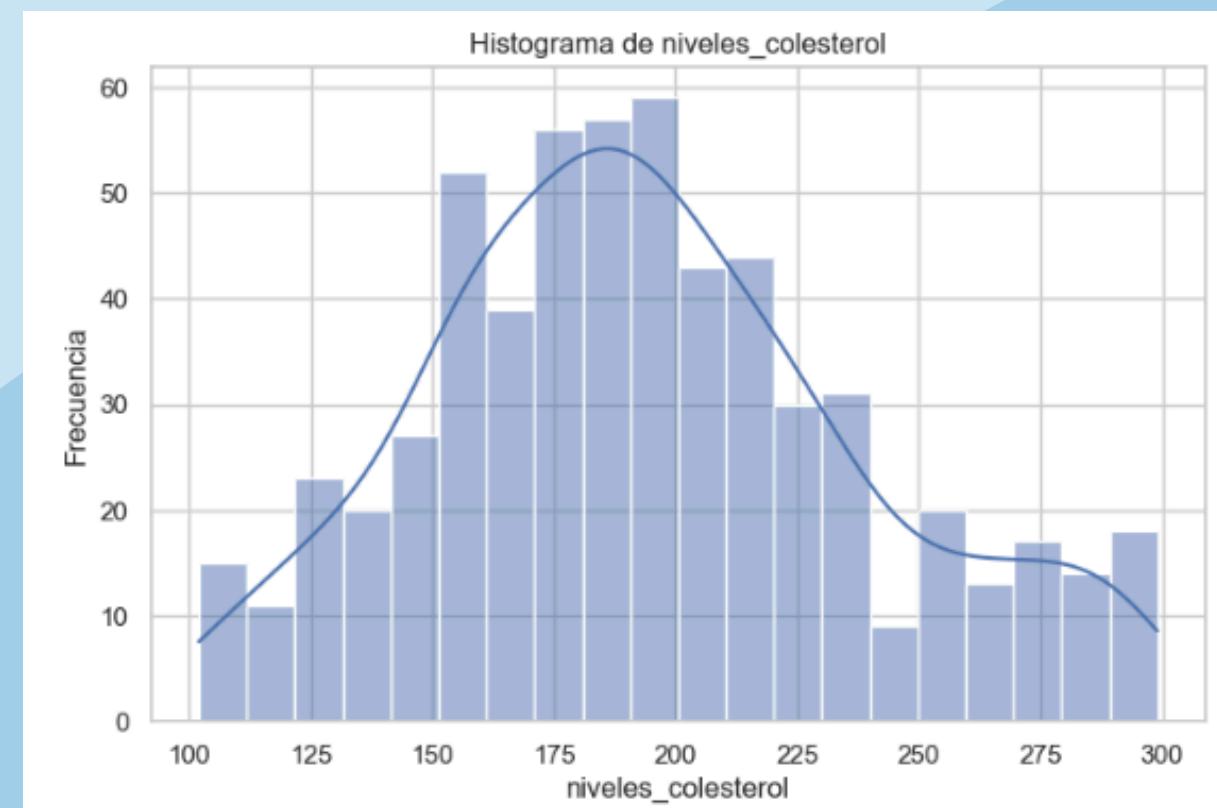
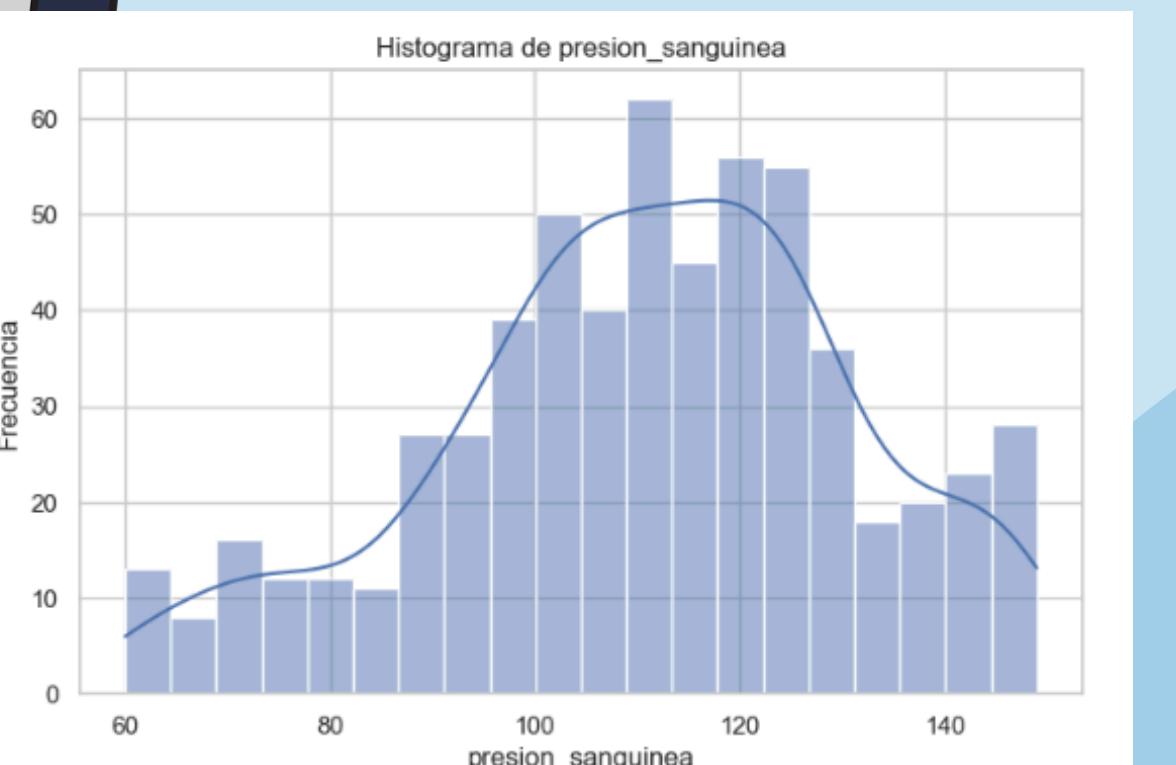
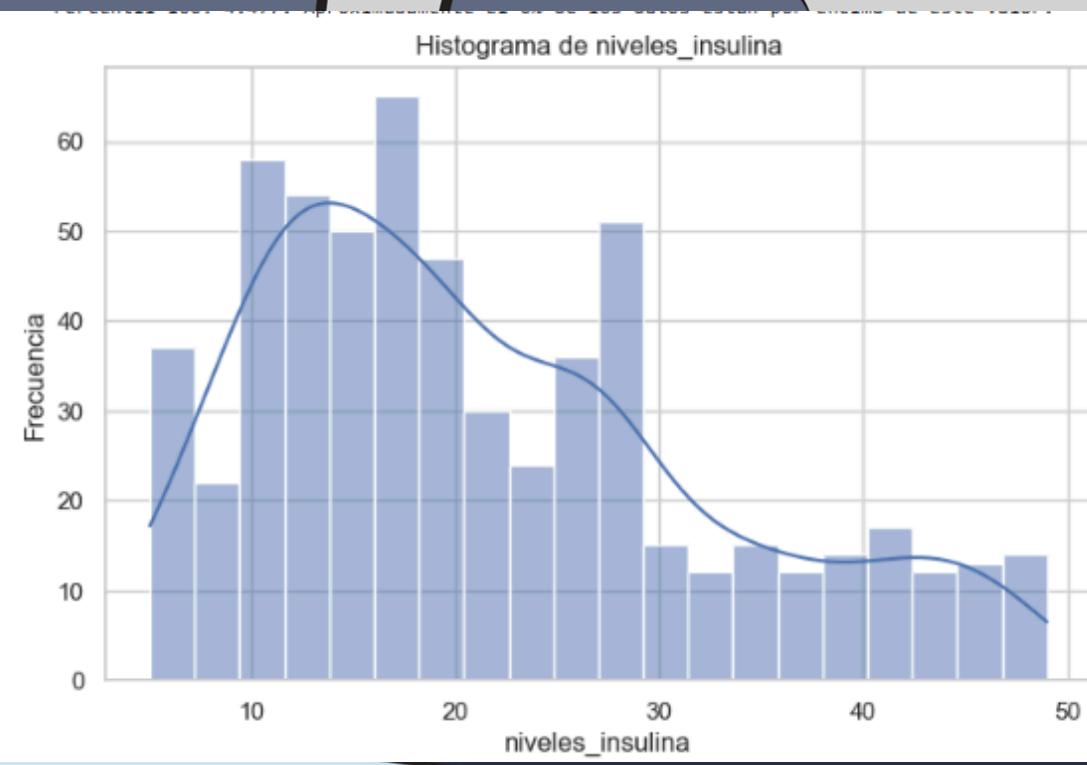
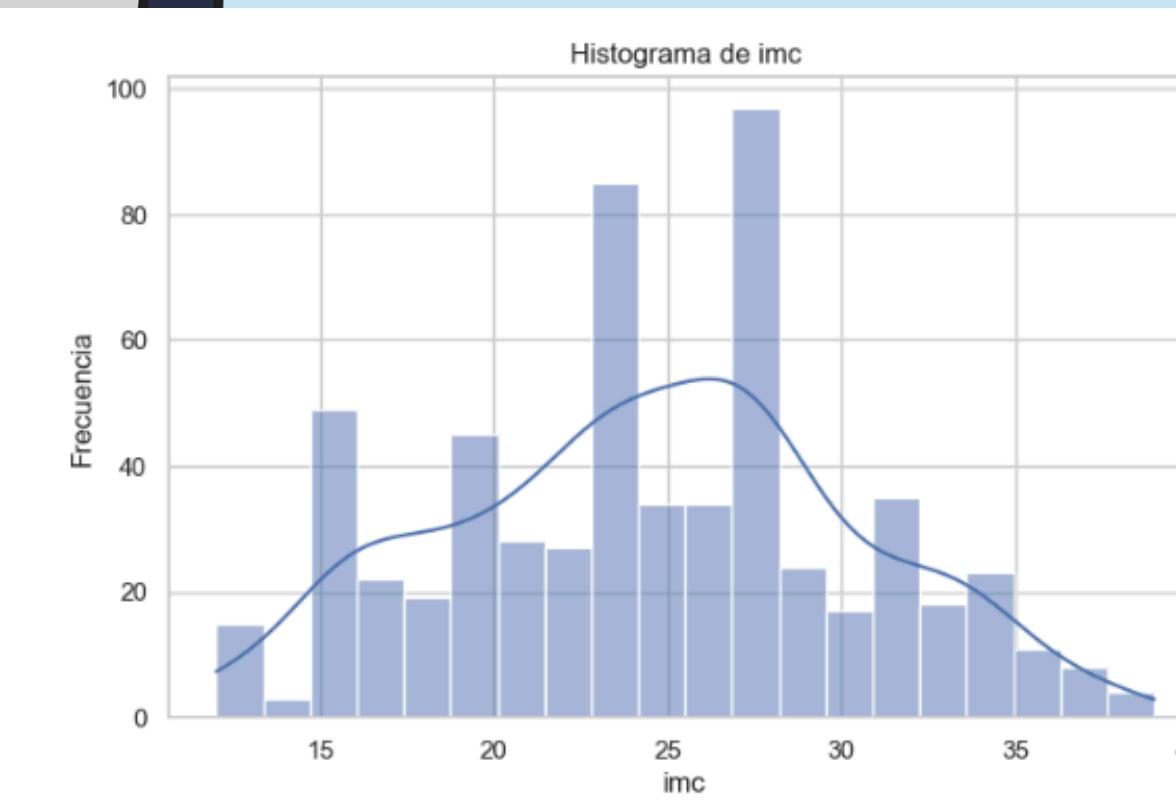
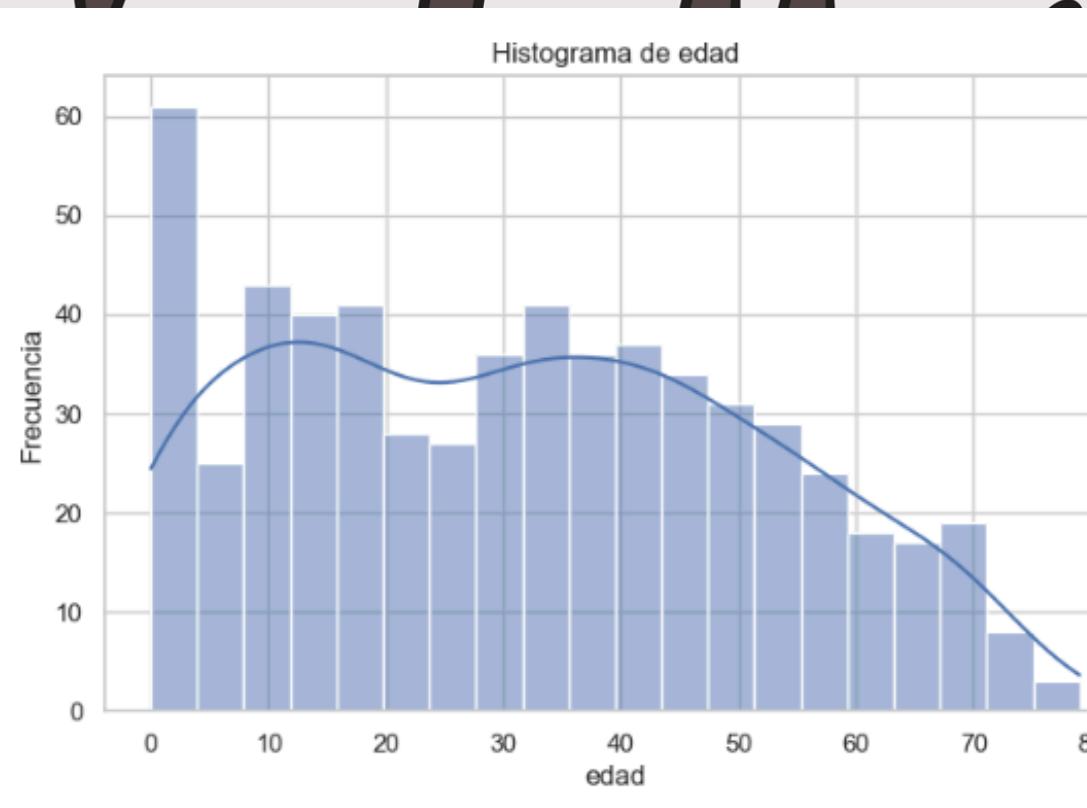
Este gráfico de dispersión permite observar la relación entre dos variables, `salud_pancreatica` y `niveles_de_encimas_digestiva`, para analizar su evaluación.



```
# 12. Gráfico de Salud Pancreática vs Niveles de Enzima Digestiva
plt.figure(figsize=(8, 6))
plt.scatter(df['niveles_de_encimas_digestiva'], df['salud_pancreatica'], alpha=0.7)
plt.xlabel('Niveles de Enzima Digestiva')
plt.ylabel('Salud Pancreática')
plt.title('Salud Pancreática vs Niveles de Enzima Digestiva')
```

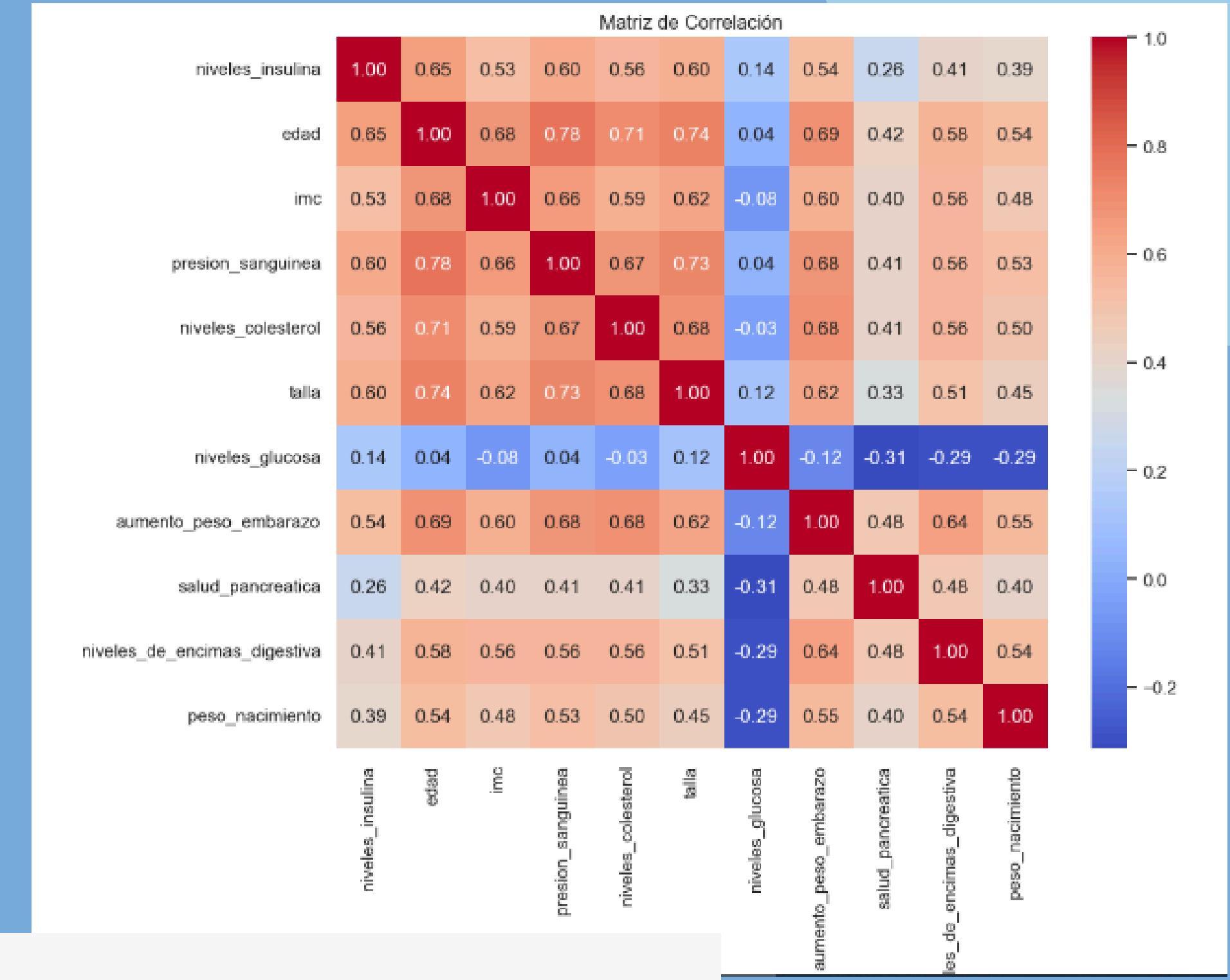


Histogramas para cada columna numérica, mostrando la distribución y permitiendo observar posibles sesgos



Mapa de calor que muestra la comparación entre las variables numéricas.

Se muestra una matriz de evaluación de las variables numéricas. Esto permite visualizar cómo cada variable se correlaciona con las demás.



```
# 16. Graficar matriz de correlación
numeric_df = df.select_dtypes(include=['int64', 'float64']) # Seleccionar solo las columnas numéricas
correlation_matrix = numeric_df.corr()

# Graficar el mapa de calor de La matriz de correlación
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True, linewidths=0.5)
plt.title('Matriz de Correlación')
plt.show()
```

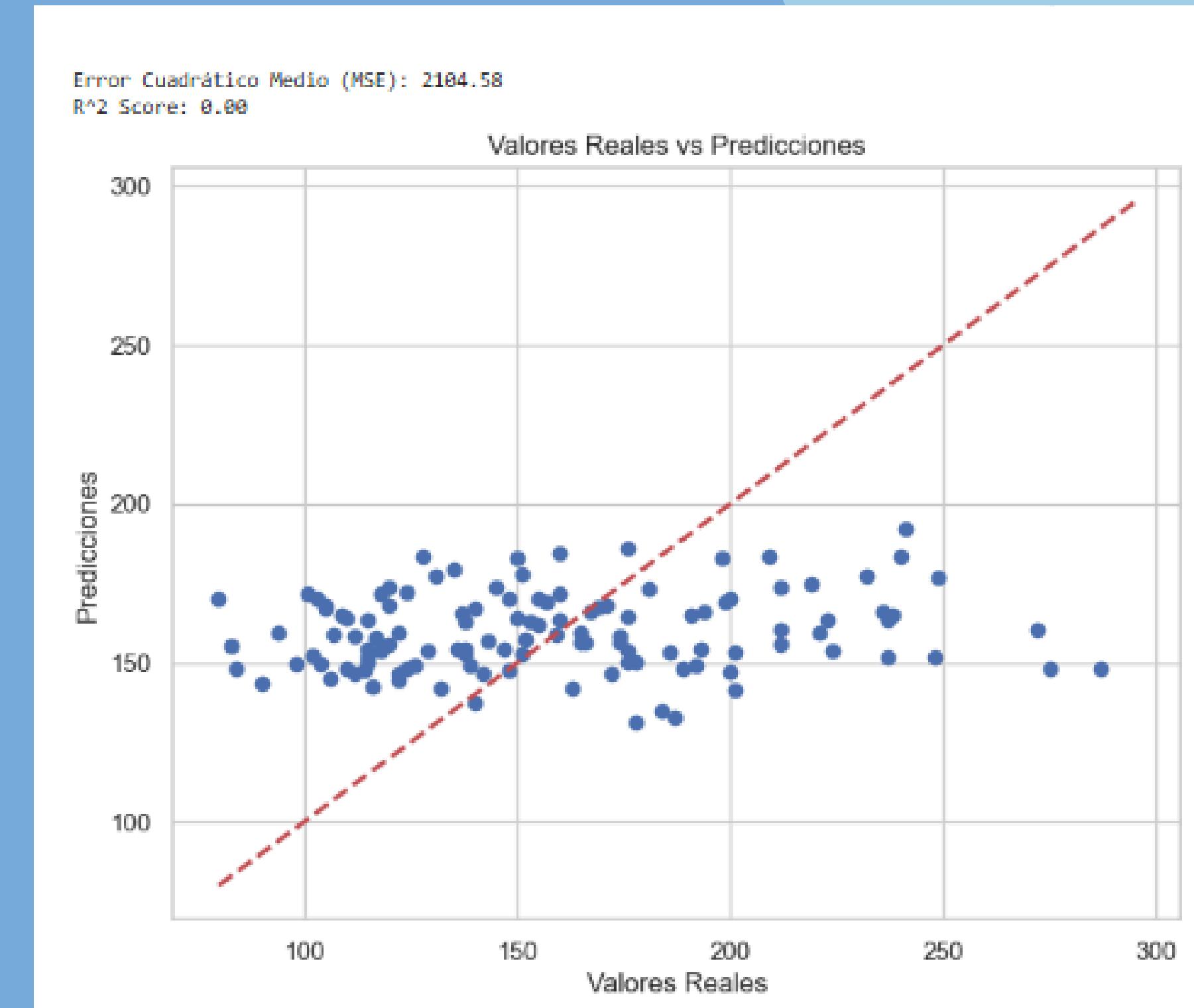


Gráfico de dispersión para comparar las predicciones del modelo con los valores reales.

Puntos azules : Representan los datos reales.

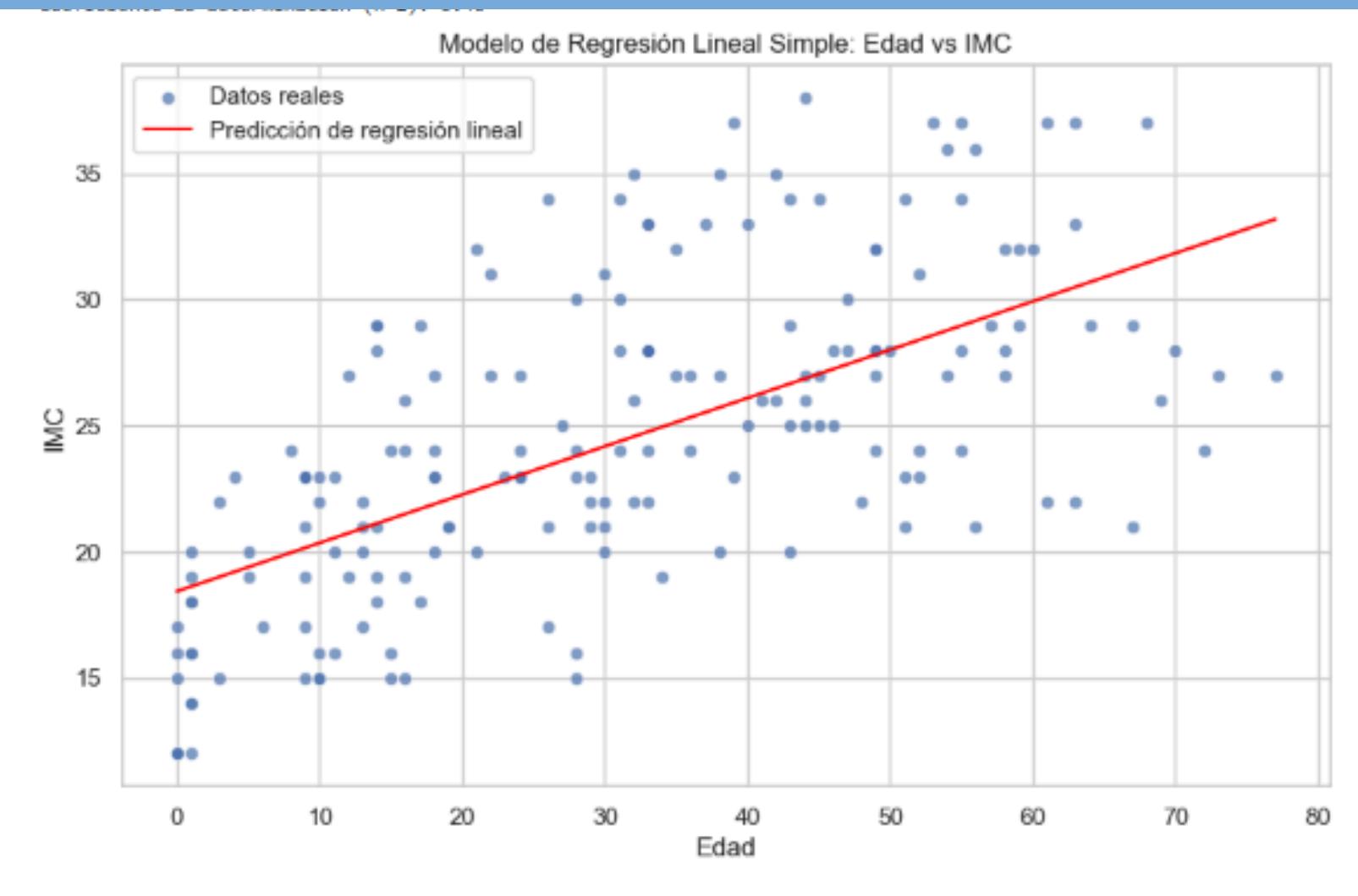
Línea roja : Muestra la predicción de un modelo de regresión lineal.

Los puntos permiten visualizar la distribución y tendencia de los datos, mientras que la línea de regresión muestra la relación lineal



Modelo de Regresión Lineal

- Se separan los datos en conjuntos de entrenamiento y prueba (80-20).
- Se entrena un modelo de regresión lineal para predecir niveles_glucosaen función de imc, niveles_insulina, y edad.
- Evaluación del modelo



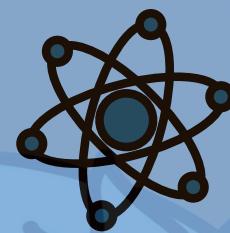
```
# 3. Visualizar la regresión Lineal
plt.figure(figsize=(10, 6))
# Gráfico de dispersión para los datos reales
sns.scatterplot(x=X_test['edad'], y=y_test, label="Datos reales", alpha=0.7)
# Línea de regresión para las predicciones
sns.lineplot(x=X_test['edad'], y=y_pred, color="red", label="Predicción de regresión lineal")
plt.title("Modelo de Regresión Lineal Simple: Edad vs IMC")
plt.xlabel("Edad")
plt.ylabel("IMC")
plt.legend() # Mostrar la Leyenda
plt.grid(True) # Añadir una cuadrícula
plt.show()
```



CONCLUSIÓN

- Proporciona una comprensión valiosa sobre las variables que influyen en la diabetes.
- Mediante el uso de técnicas estadísticas y de visualización, se lograrán identificar patrones significativos, así como valores atípicos y relaciones clave entre variables.
- Este proceso permite construir modelos predictivos efectivos y puede sentar las bases para futuras investigaciones y aplicaciones prácticas en el ámbito de la salud, facilitando una toma de decisiones más informada y personalizada en el manejo de la diabetes y sus factores de riesgo.





Equipa 13

MUCHAS
GRACIAS