PRÁCTICA 2: LIMPIEZA Y VALIDACIÓN DE LOS DATOS

Autor: Jordi Lago López

1. Descripción del Dataset

Este dataset nos muestra cada una de las medidas y dimensiones que puedes afectar a la hora de padecer o no un ataque cardiaco. El dataset ha sido proporcionado por asignatura y lo podemos encontrar en el siguiente enlace:

Heart Attack Analysis & Prediction Dataset | Kaggle

Dicho esto, vamos a ver que datos tenemos disponibles y su descripción:

- **Age**: Edad del paciente.
- **Sex:** Género del paciente.
- **Cp (Tipo de dolor en el pecho):** 0 = Angina Típica, 1 = Angina Atípica, 2 = Dolor no anginoso, 3 = Asintomático.
- **trtbps** : Presión arterial en reposo (in mm Hg)
- **Chol:** Colesterol mg/dl
- **Fbs**: (nivel de azucar en sangre > 120 mg/dl) ~ 1 = True, 0 = False
- **Restecg**: Resultados electrocardiograma ~ 0 = Normal, 1 = ST-T normalidad en la onda, 2 = Hipertrofia ventricular.
- **Thalachh:** Frecuencia cardíaca máxima alcanzada
- **Oldpeak**: pico anterior
- **Slp**: Pendiente Gráfico Electrocardiograma
- **Caa:** Número de arterias coronarias principales.
- **Thall:** Resultado test de Estrés de Thallium (0,3)
- **Exng**: Angina inducida por el ejercicio → 1 = Si, 0 = No
- **Output:** Variable Target

En esta práctica unificaremos los datos, los limpiaremos y trataremos de estimar un modelo que a partir de los datos nos pueda predecir el sí un paciente va a padecer un ataque cardiaco o no. También realizaremos pruebas estadísticas para verificar la relación entre las variables que hemos escogido.

2. Integración y Selección de Datos.

Primero lo que hemos hecho es cargar el dataset y ver que datos teníamos disponibles:

```
# Leemos el csv Heart
df=pd.read_csv("heart.csv")

# Buscamos las 5 primeras filas y vemos que datos tenemos
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Una vez cargado el dataset, vemos de cuantas filas y columnas se compone.

```
# Contabilizamos el número de filas y columnas que tenemos

print('Número de filas =',df.shape[0], 'y número de columnas =',df.shape[1])
```

```
Número de filas = 303 y número de columnas = 14
```

Vemos que tenemos 303 filas y 14 columnas, de las cuales seleccionamos una serie de campos para no hacer un análisis demasiado amplio.

```
# Vamos a eliminar una serie de columnas que no nos interesan para nuestro análisis futuro.

df.drop(columns=['caa','restecg','slp','cp','oldpeak'],inplace = True)
```

Quedando el siguiente dataframe:

```
df
```

	age	sex	trtbps	chol	fbs	thalachh	exng	thall	output
0	63	1	145	233	1	150	0	1	1
1	37	1	130	250	0	187	0	2	1
2	41	0	130	204	0	172	0	2	1
3	56	1	120	236	0	178	0	2	1
4	57	0	120	354	0	163	1	2	1
...	...	...	...	...	...	...	...	...	...
298	57	0	140	241	0	123	1	3	0
299	45	1	110	264	0	132	0	3	0
300	68	1	144	193	1	141	0	3	0
301	57	1	130	131	0	115	1	3	0
302	57	0	130	236	0	174	0	2	0

303 rows × 9 columns

Una vez obtenido el Dataframe final, vamos a ver la información que tenemos de las variables resultantes:

```
# Ahora vamos a ver que información tenemos de nuestros datos

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   trtbps    303 non-null    int64
 3   chol      303 non-null    int64
 4   fbs       303 non-null    int64
 5   thalachh  303 non-null    int64
 6   exng      303 non-null    int64
 7   thall     303 non-null    int64
 8   output    303 non-null    int64
dtypes: int64(9)
```

Como hemos visto, tenemos todo variables int, por lo que tanto nuestras variables categóricas como continuas son valores numéricos enteros.

A continuación, vamos a analizar más en profundidad nuestros datos y ver si obtenemos valores nulos, duplicados, etc.

3. Limpieza de Datos

3.1. ¿Los datos contienen ceros o elementos vacíos? Gestiona cada uno de estos casos.

Vamos a ver si tenemos valores nulos o ceros en nuestros datos. Cabe decir que en la imagen anterior ya hemos podido ver que los datos no contienen nulos, pero vamos a asegurarnos.

```
# Verificamos que no tengamos valores nulos.

df.isnull().sum()
```

```
age         0
sex         0
trtbps      0
chol        0
fbs         0
thalachh    0
exng        0
thall       0
output      0
dtype: int64
```

Podemos concluir que tenemos un 0% de valores nulos en nuestros datos, por lo que a priori no deberemos utilizar ninguna fórmula de sustitución de nulos.

¿Pero tenemos valores duplicados?

```
# Una vez analizados los valores nulos, vamos a ver si tenemos valores duplicados en nuestro dataset.

df.duplicated().sum()
```

```
1
```

Vemos que, si tenemos un valor duplicado, por lo que podemos eliminarlo de nuestro dataset y a ver cómo queda nuestra tabla final en cuanto a filas y columnas.

```
# Eliminamos el valor duplicado obtenido y imprimimos por pantalla el número de filas resultantes.

df.drop_duplicates(inplace=True)
print('Number of rows are =',df.shape[0], ',and number of columns are =',df.shape[1])
```

```
Number of rows are = 302 ,and number of columns are = 9
```

3.2. Identifica y gestiona los valores extremos.

Para el caso de los valores extremos o outliers, vamos a realizar un boxplot de nuestras variables continuas.

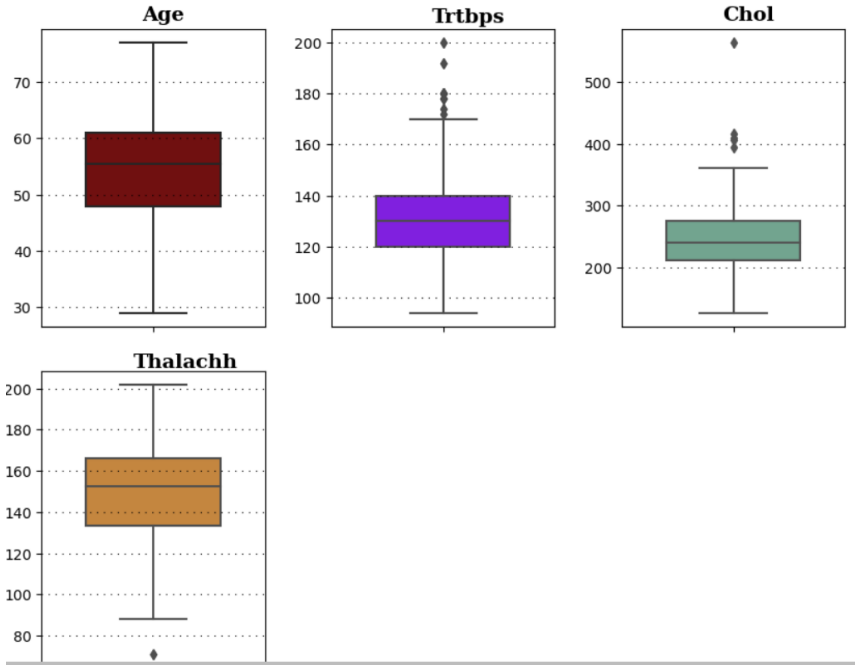
Para ello, primero crearemos dos variables que nos separen las variables categóricas de las continuas y después realizaremos el boxplot para identificar los outliers.

```
# Dividimos por el tipo de columnas que tenemos.

cat_cols = ['sex','exng','fbs','thall']
con_cols = ["age","trtbps","chol","thalachh"]
target_col = ["output"]
print("The categorial cols are : ", cat_cols)
print("The continuous cols are : ", con_cols)
print("The target variable is :  ", target_col)
```

```
The categorial cols are :  ['sex', 'exng', 'fbs', 'thall']
The continuous cols are :  ['age', 'trtbps', 'chol', 'thalachh']
The target variable is :   ['output']
```

Una vez definidas las variables, pasamos a dibujar nuestros gráficos.

```
fig = plt.figure(figsize=(10,8))
gs = fig.add_gridspec(2,3)
gs.update(wspace=0.3, hspace=0.15)
ax0 = fig.add_subplot(gs[0,0])
ax1 = fig.add_subplot(gs[0,1])
ax2 = fig.add_subplot(gs[0,2])
ax3 = fig.add_subplot(gs[1,0])

# Age
ax0.text(-0.05, 81, 'Age', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax0.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxplot(ax=ax0,y=df['age'],palette=["#800000"],width=0.6)
ax0.set_xlabel("")
ax0.set_ylabel("")

# Trtbps
ax1.text(-0.05, 208, 'Trtbps', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxplot(ax=ax1,y=df['trtbps'],palette=["#8000ff"],width=0.6)
ax1.set_xlabel("")
ax1.set_ylabel("")

# Chol
ax2.text(-0.05, 600, 'Chol', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax2.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxplot(ax=ax2,y=df['chol'],palette=["#6aac90"],width=0.6)
ax2.set_xlabel("")
ax2.set_ylabel("")

# Thalachh
ax3.text(-0.09, 210, 'Thalachh', fontsize=14, fontweight='bold', fontfamily='serif', color="#000000")
ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxplot(ax=ax3,y=df['thalachh'],palette=["#da8829"],width=0.6)
ax3.set_xlabel("")
ax3.set_ylabel("")
```



Vemos lo siguiente:

- **Age:** La variable age no dispone de valores outliers, por lo que no hay que tratarla.
- **Trbps:** Vemos como esta variable si tiene valores extremos, a partir del 170, pero no debemos eliminarlos, ya que esos valores puedes ser indicios de problemas cardiovasculares.
- **Chol:** En cuanto al colesterol, también vemos valores extremos, pero al igual que el Trbps, nos indican problemas de corazón y pueden ser un indicio a un ataque cardiaco.
- **Thalachh:** Este último valor solo tiene un outlier por debajo de 80, pero entiendo que tampoco debemos eliminarlo, ya que posteriormente el algoritmo debe aprender que tipo de paciente

- puede padecer un ataque o no, y es posible que estos valores de frecuencias cardiacas sean un indicativo.

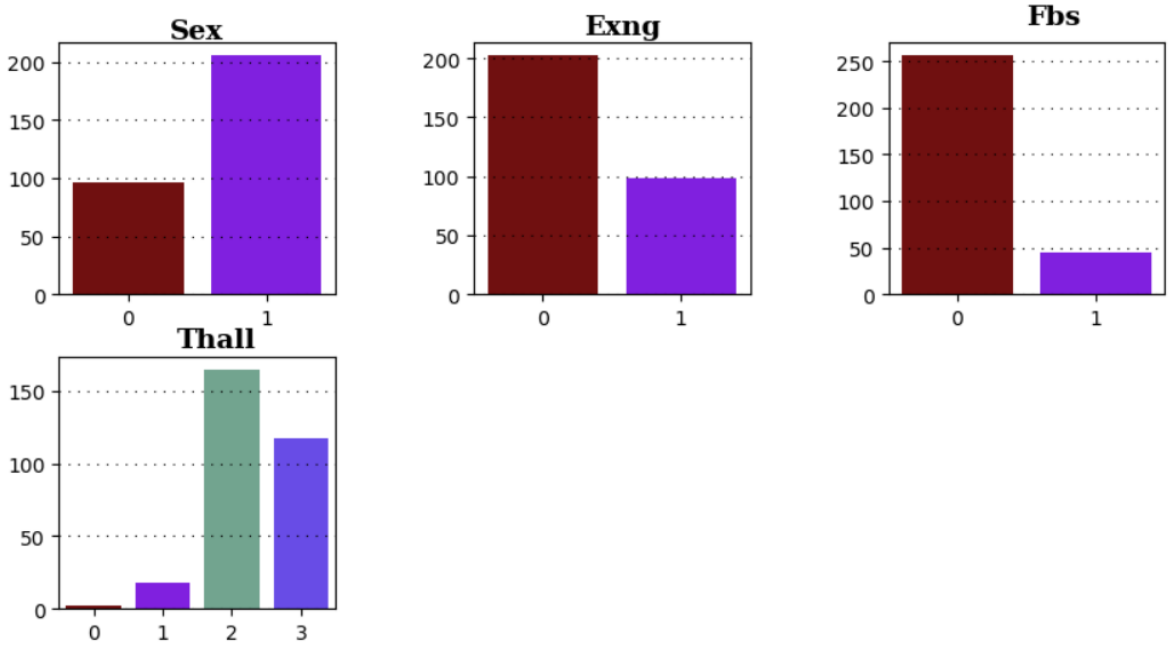
Dicho esto, y una vez tratados los nulos, duplicados y outliers, Vamos a ver como sería nuestra tabla y como quedan nuestras variables.

```
# Hecho esto, vamos a ver como nos queda el análisis descriptivo de nuestros datos resultantes.

df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	302.0	54.420530	9.047970	29.0	48.00	55.5	61.00	77.0
sex	302.0	0.682119	0.466426	0.0	0.00	1.0	1.00	1.0
trtbps	302.0	131.602649	17.563394	94.0	120.00	130.0	140.00	200.0
chol	302.0	246.500000	51.753489	126.0	211.00	240.5	274.75	564.0
fbs	302.0	0.149007	0.356686	0.0	0.00	0.0	0.00	1.0
thalachh	302.0	149.569536	22.903527	71.0	133.25	152.5	166.00	202.0
exng	302.0	0.327815	0.470196	0.0	0.00	0.0	1.00	1.0
thall	302.0	2.314570	0.613026	0.0	2.00	2.0	3.00	3.0
output	302.0	0.543046	0.498970	0.0	0.00	1.0	1.00	1.0

- La presión arterial promedio de un individuo es 131.6 mientras que el valor máximo sube a 200.
- La frecuencia cardíaca media del grupo es de 149.5, con un máximo de 202. También vemos que el 75% de los datos tienen valores inferiores a 166.
- La edad del grupo varía de 29 a 77 años y la media de edad es de 54.42.
- También podemos observar cómo tenemos más sex = 1 que igual a 0. Esto lo veamos porque el promedio es superior a 0.5, lo que nos indica que hay más valores igual a 1.

Podemos ver también como se distribuyen nuestras variables categóricas.



Y como se correlacionan cada uno de nuestros datos.

```
# Ahora vemos la correlación de núestrass variables continuas

df_corr = df.corr().transpose()
df_corr
```

	age	sex	trtbps	chol	fbs	thalachh	exng	thall	output
age	1.000000	-0.094962	0.283121	0.207216	0.119492	-0.395235	0.093216	0.065317	-0.221476
sex	-0.094962	1.000000	-0.057647	-0.195571	0.046022	-0.046439	0.143460	0.211452	-0.283609
trtbps	0.283121	-0.057647	1.000000	0.125256	0.178125	-0.048023	0.068526	0.062870	-0.146269
chol	0.207216	-0.195571	0.125256	1.000000	0.011428	-0.005308	0.064099	0.096810	-0.081437
fbs	0.119492	0.046022	0.178125	0.011428	1.000000	-0.007169	0.024729	-0.032752	-0.026826
thalachh	-0.395235	-0.046439	-0.048023	-0.005308	-0.007169	1.000000	-0.377411	-0.094910	0.419955
exng	0.093216	0.143460	0.068526	0.064099	0.024729	-0.377411	1.000000	0.205826	-0.435601
thall	0.065317	0.211452	0.062870	0.096810	-0.032752	-0.094910	0.205826	1.000000	-0.343101
output	-0.221476	-0.283609	-0.146269	-0.081437	-0.026826	0.419955	-0.435601	-0.343101	1.000000

Aquí vemos como la variable objetivo “output” tiene una correlación más fuerte con thalachh de manera positiva, y una correlación negativa con exng.

4. Análisis de Datos

4.1. Selección de los grupos de datos que se quieren analizar/comparar (p. ej., si se van a comparar grupos de datos, ¿cuáles son estos grupos y qué tipo de análisis se van a aplicar?)

Podríamos hacer diferentes selecciones de datos para realizar el análisis mediante contrastes de hipótesis, regresiones, correlaciones, etc. Estos son algunos de los grupos que podríamos utilizar:

Como se trata de un problema de clasificación binaria, necesitaremos variables que nos permitan comparar la proporción de éxito entre dos grupos. Algunos ejemplos podrían incluir:

- Comparar la proporción de éxito entre hombres y mujeres: H0: p1 = p2 (la proporción de tener un output = 1 entre hombres y mujeres es igual) vs H1: p1 ≠ p2 (la proporción de tener un output = 1 entre hombres y mujeres es diferente).
- Comparar la proporción de output = 1 entre pacientes con niveles altos y bajos de colesterol: H0: p1 = p2 (la proporción de output = 1 entre pacientes con niveles altos y bajos de colesterol es igual) vs H1: p1 ≠ p2 (la proporción de output = 1 entre pacientes con niveles altos y bajos de colesterol es diferente).
- Ver que variables están más correlacionadas con la posibilidad de obtener un output = 1. Esto lo podríamos hacer con una regresión y ver los coeficientes de cada variable respecto a la variable objetivo.

En cualquiera de estos casos, se debe elegir un nivel de significancia apropiado y utilizar una prueba estadística apropiada (como la prueba Z o la prueba t) para determinar si existe una diferencia estadísticamente significativa entre los grupos.

4.2. Aplicación de pruebas estadísticas para comparar los grupos de datos.

1. Por ejemplo, para el primer caso, podríamos hacer lo siguiente:

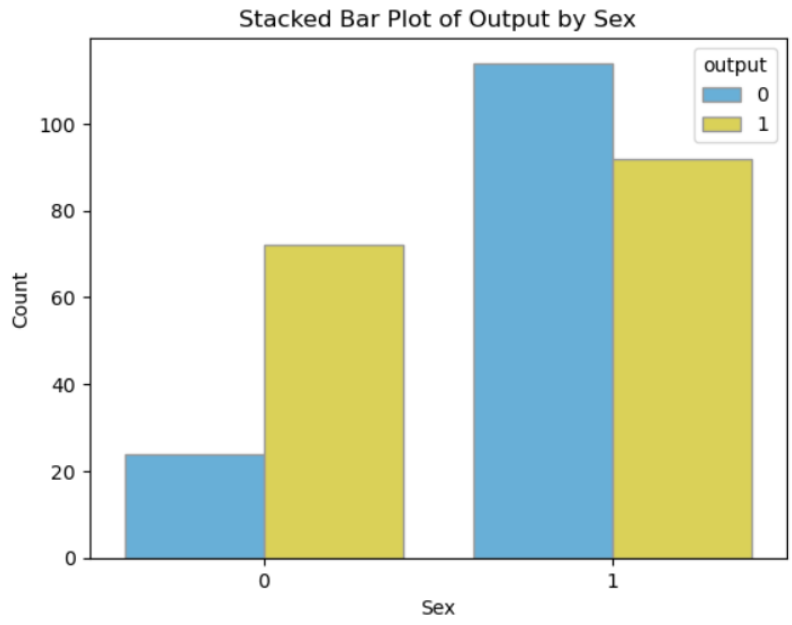
```
from scipy.stats import chi2_contingency

# Creamos una tabla de contingencia para sex y output
ct = pd.crosstab(df["sex"], df["output"])

# Realizamos el chi-cuadrado
chi2, p, dof, expected = chi2_contingency(ct)

# Print
print(f"Chi-square test statistic: {chi2}")
print(f"p-value: {p}")

# Indicamos nuestro nivel de significancia
alpha = 0.05
if p < alpha:
    print("Existe relación entre el sexo y tener o no un infarto")
else:
    print("No existe relación entre el sexo y tener o no un infarto")
```

```
Chi-square test statistic: 23.083879459669042
p-value: 1.5508552054949547e-06
Existe relación entre el sexo y tener o no un infarto
```

Vemos como obtenemos un p-value inferior al valor alfa, lo que nos dice que debemos rechazar la hipotesis nula ¡, y decir que si que existe una relación entre el sexo y tener o no un ataque cardiaco.

Vamos a ver un grafico de barras, para ver como relaciona estos valores.

```
# Creamos un gráfico de barras por sexo.
sns.countplot(x="sex", hue="output", data=df,palette=["#56B4E9","#F0E442"],edgecolor=".6")

# Añadimos labels y título
plt.xlabel("Sex")
plt.ylabel("Count")
plt.title("Stacked Bar Plot of Output by Sex")

plt.show()
```



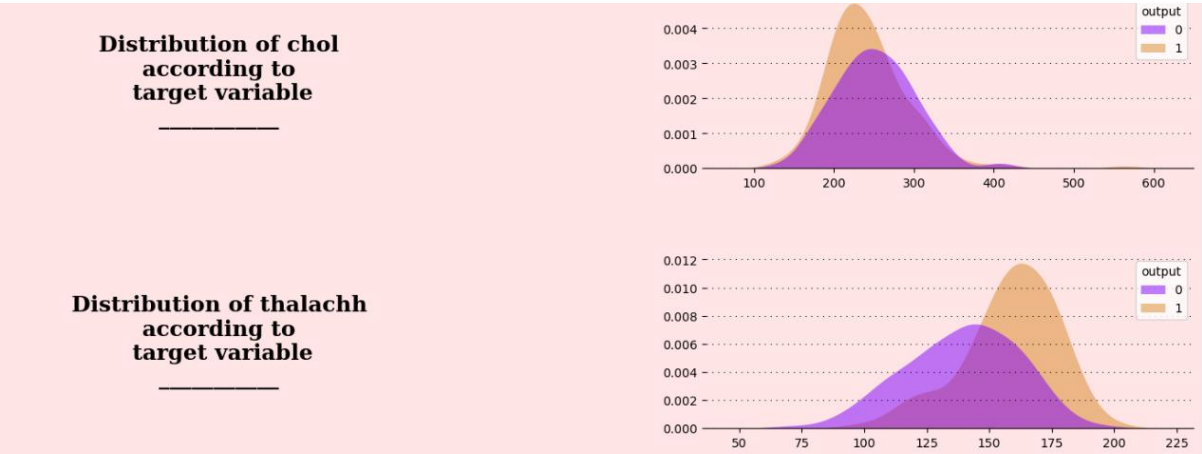
2. Para el segundo caso, podemos calcular p mediante la z de contraste.

```
# Hacemos algo parecido pero para ver si hay diferencias entre valores altos y bajos de chol y thalachh

from scipy.stats import norm

success_rate_low = df[(df["chol"] <= 200) & (df["thalachh"] <= 150)]["output"].mean()

success_rate_high = df[(df["chol"] > 200) & (df["thalachh"] > 150)]["output"].mean()

# Calculamos nuestro valor z de contraste
z =(success_rate_high - success_rate_low)/np.sqrt((success_rate_low * (1 - success_rate_low)) /
                                                  len(df[(df["chol"] > 200) & (df["thalachh"] > 150)]) +
                                                  (success_rate_high * (1 - success_rate_high)) /
                                                  len(df[(df["chol"] > 200) & (df["thalachh"] > 150)]))

# Calculamos el p-value gracias a nuestra z.
p = 2 * (1 - norm.cdf(abs(z)))

# Añadimo el valor de significancia.
alpha = 0.05
if p < alpha:
    print(p, "Existe diferencia entre clientes con menos de 200 de colesterol y pacientes con más de 200 en colesterol")
else:
    print(p, "No existe diferencia entre clientes con menos de 200 de colesterol y pacientes con más de 200 en colesterol")
```

```
4.96589436238537e-11 Existe diferencia entre clientes con menos de 200 de colesterol y pacientes con más de 200 en colesterol
```

En este caso el valor ptambién es inferior a alfa, por lo que rechazamos la hipotesis nula. Esto quiere decir que pacientes con valores altos de estas dos variables combinadas, tienen más indicios de sufrir problemas cardiacos.

Vemos un gráfico de estas variables continuas.



En el gráfico también podemos ver como Valore superiores a 200 en Colesterol, te dan indicios de problema cardiaco. Pero si además lo combinas con Thalachh, que claramente a mayor frecuencia máxima alcanzada mayor riesgo, entonces es muy probable sufrir alguna deficiencia y obtener un output = 1.

3. Realizamos una regresión logística de nuestro modelo, que nos servirá para predecir outputs en función de nuevos valores de nuestras variables.

Para ello, primero debemos realizar un encoding de las variables categóricas y posteriormente escalar todas nuestras variables continuas para que estén al mismo nivel.

```
# Creamos una copia de df
df1 = df

# DEfinimos las columnas para realizar el encoding y el escalado
cat_cols = ['sex','exng','fbs','thall']
con_cols = ["age","trtbps","chol","thalachh"]

# Realizamos un encoding de las variables categóricas
df1 = pd.get_dummies(df1, columns = cat_cols, drop_first = True)

# Defiimos la variable X e Y
X = df1.drop(['output'],axis=1)
y = df1[['output']]

# Escalamos nuestros datos
scaler = RobustScaler()

X[con_cols] = scaler.fit_transform(X[con_cols])
print("The first 5 rows of X are")
X.head()
```

Una vez realizado esto, obtenemos lo siguiente:

```
The first 5 rows of X are
```

	age	trtbps	chol	thalachh	sex_1	exng_1	fbs_1	thall_1	thall_2	thall_3
0	0.576923	0.75	-0.117647	-0.076336	1	0	1	1	0	0
1	-1.423077	0.00	0.149020	1.053435	1	0	0	0	1	0
2	-1.115385	0.00	-0.572549	0.595420	0	0	0	0	1	0
3	0.038462	-0.50	-0.070588	0.778626	1	0	0	0	1	0
4	0.115385	-0.50	1.780392	0.320611	0	1	0	0	1	0

Una vez tenemos esto, dividimos nuestro dataset en un conjunto de train y test:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42)
print("El tamaño de X_train es  ", X_train.shape)
print("El tamaño de X_test es   ",X_test.shape)
print("El tamaño de y_train es  ",y_train.shape)
print("El tamaño de y_test es   ",y_test.shape)
```

```
El tamaño de X_train es   (241, 10)
El tamaño de X_test es    (61, 10)
El tamaño de y_train es   (241, 1)
El tamaño de y_test es    (61, 1)
```

Y ya podemos iniciar nuestro modelo y entrenar los datos.

```
# Instanciamos el modelo
model = LogisticRegression()

# Hacemos un fit sobre los datos de entrenamiento
model.fit(X_train, y_train)

# Realizamos el predict sobre los datos de test.
y_pred = model.predict(X_test)

# Vemos la precisión del modelo
print("La accuracy es ", accuracy_score(y_test, y_pred))
```

```
La accuracy es  0.819672131147541
```

Vemos como una vez realizada la predicción, obtenemos una precisión del 82%. Por lo que el modelo consigue predecir correctamente en el 82% de los casos.

Vemos que obtenemos en los valores de intercept y coeficientes:

```
model.intercept_
```

```
array([1.33231298])
```

```
model.coef_
```

```
array([[-0.23403231, -0.38330727, -0.19594377,  0.88928466, -1.07073171,
        -1.21505946,  0.43143275, -0.14902272,  0.74539303, -0.98051027]])
```

En nuestro caso el intercept tiene un valor de 1.33231298 en tu caso, lo que significa que cuando todas las variables independientes son cero, la probabilidad de ocurrencia del evento es 1 / (1 + e^(-1.33231298)) = 0.27 . Este valor es solo un punto de partida para entender cómo afecta cada variable independiente a la probabilidad de ocurrencia del evento.

En cuanto al resultado de los coeficientes, cuando tenemos un coeficiente negativo, éstos indican que hay una relación negativa entre la primera variable independiente y la probabilidad de ocurrencia del evento. Esto significa que a medida que aumenta el valor de la primera variable independiente, la probabilidad de ocurrencia del evento disminuye.

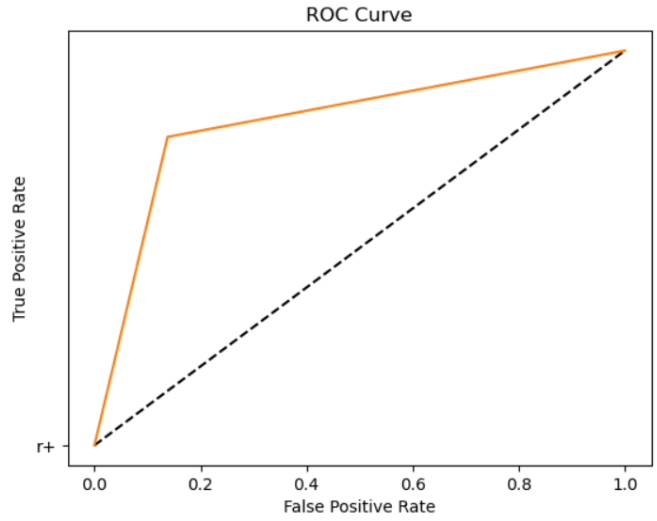
Para este caso tendríamos a las variables **age, trbps, chol, sex, exng y thall_3.**

Mientras que, si tenemos coeficientes positivos, tendremos que existe una relación positiva entre la primera variable independiente y la probabilidad de ocurrencia del evento. Esto significa que a medida que aumenta el valor de la primera variable independiente, la probabilidad de ocurrencia del evento aumenta.

Y en este caso tendríamos las variables **thalachh, fbs, thall_1 y thall_2.**

Vemos el resultado de la curva Roc:

```
# Realizamos la curva roc.
fpr,tpr,threshols=roc_curve(y_test,y_pred)

plt.plot([0,1],[0,1],"k--",'r+')
plt.plot(fpr,tpr,label='Logistic Regression')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()
```



5. Conclusiones

1. Hay ciertos valores atípicos en todas las características continuas.

2. Los datos consisten en más del doble del número de personas con sexo = 1 que sexo = 0.

3. No existe una correlación lineal aparente entre las variables continuas.

4. Es intuitivo que las personas mayores pueden tener mayores posibilidades de sufrir un ataque al corazón, pero de acuerdo con el gráfico de distribución de la salida de la edad, es evidente que este no es el caso.

5. De acuerdo con el diagrama de distribución de la producción de thalachh, las personas con una frecuencia cardíaca máxima más alta alcanzada tienen mayores posibilidades de sufrir un ataque cardíaco.

6. Las personas con sexo = 1 tienen mayor probabilidad de sufrir un infarto.

7. Las personas con thall = 2 tienen muchas más posibilidades de sufrir un ataque al corazón.

8. Las personas sin angina inducida por el ejercicio (exng), tienen una mayor probabilidad de sufrir un ataque al corazón.

6. Enlace Video Google Drive.

Adjunto también en el documento el enlace al video de Google Drive.

https://drive.google.com/file/d/1fwzlBvcWTv-XApvP8LT4LKsn43BoRXtz/view?usp=share_link