



UNIVERSIDAD  
**Panamericana**

## **Análisis de datos y clasificador de ataques cardíacos**

---

Minería de Datos

*Eduardo Solano Jaime*

0213663

*Especialidad en Ciencia de Datos*

*Universidad Panamericana campus Guadalajara*

## Objetivo

El objetivo de este reporte es utilizar el dataset de [Heart Attack Analysis & Prediction Dataset](#) para entrenar un modelo de algoritmo de clasificación de machine learning que prediga si una persona está en riesgo de tener un ataque cardíaco o no.

El dataset contiene información sobre 303 pacientes, incluyendo su edad, sexo, tipo de dolor de pecho, presión arterial en reposo, colesterol, frecuencia cardíaca, índice de masa corporal, glucosa en sangre, tabaquismo, consumo de alcohol, actividad física, entre otros. La variable objetivo es si el paciente testa en riesgo de un paro cardiaco.

Para entrenar el modelo, se utilizó regresión logística, no sin su previo análisis y procesamiento de las variables. Se sigue el proceso de visualización y explicación de las variables, usando métodos de transformación y métodos estadísticos para modificar las distribuciones. Finalmente se califica el modelo con métodos de validación.

Los resultados del modelo se analizarán para determinar su utilidad para la predicción de ataques cardíacos.

El cuaderno de Jupyter completo dónde se realizaron todos los cálculos (y las imágenes completas en tamaño original) se encuentra en mi [repositorio de GitHub](#) para mayor comprensión.

## Metodología

Los pasos a seguir para completar este reporte son los siguientes:

- Exploración del dataset para entender el significado de información de las bases de datos.
- Visualización de los datos mediante varias representaciones
- Preparación de los datos para el entrenamiento del modelo, incluyendo la transformación de las variables.
- Entrenamiento del modelo.
- Evaluación del modelo usando métodos de validación (técnicas de remuestreo).
- Análisis de los resultados para determinar su utilidad para la predicción de ataques cardíacos.

## Análisis de variables

### Exploración del dataset

Al importar el dataset se utilizó la función `DataFrame.head()`, la cual se utiliza para devolver las primeras n filas de un objeto DataFrame. El dataset se importó previamente y se declaró con el nombre 'df', de tipo `DataFrame`.

```
df.head()
```

	<b>age</b>	<b>sex</b>	<b>cp</b>	<b>trtbps</b>	<b>chol</b>	<b>fb</b>	<b>restecg</b>	<b>thalachh</b>	<b>exng</b>	<b>oldpeak</b>	<b>slp</b>	<b>caa</b>	<b>thall</b>	<b>output</b>
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

Antes de continuar con la actividad, se checa la existencia de valores nulos o no-numéricos en el dataset. Por lo que se usó la función `Dataframe.info()` para poder comparar la cantidad total de entradas por variable que no sean nulas o no-numéricas contra la cantidad total de entradas. En este caso los número coinciden, lo que comprueba la no-existencia de estas variables. De igual manera se usa la función para evaluar el tipo de dato que se presenta en el dataset. Así mismo, la función provee el dato del tamaño del dataset, que en este caso es de **(303, 14)**.

Con dicha información y la documentación se puede realizar una explicación profunda de las características de los atributos para su futuro procesamiento y comprensión.

<b>Atributo</b>	<b>Descripción</b>	<b>Tipo</b>
age	Edad en años	Contínua / int64
sex	Sexo (1 = masculino; 0 = femenino)	Categórica / int64
cp	Tipo de dolor torácico (1 = angina típica; 2 = angina atípica; 3 = dolor no anginoso; 0 = asintomático)	Categórica / int64
trestbps	Presión arterial en reposo (en mm Hg al ingresar al hospital)	Contínua / int64
chol	Colesterol sérico en mg/dl	Continua / int64
fbs	Azúcar en sangre en ayunas > 120 mg/dl (1 = verdadero; 0 = falso)	Categórica / int64
restecg	Resultados del electrocardiograma en reposo (1 = normal; 2 = con anormalidad de la onda ST-T; 0 = hipertrofia)	Categórica / int64
thalach	Frecuencia cardíaca máxima alcanzada	Contínua / int64
exang	Angina inducida por el ejercicio (1 = sí; 0 = no)	Categórica / int64
oldpeak	Depresión del ST inducida por el ejercicio en relación con el reposo	Contínua / float
slope	Pendiente del segmento ST del ejercicio máximo (2 = ascendente; 1 = plana; 0 = descendente)	Categórica / int64
ca	Número de vasos principales (0-3) coloreados por fluroscopia	Categórica / int64
thal	Talio (2 = normal; 1 = defecto fijo; 3 = defecto reversible)	Categórica / int64
num	Atributo predicho	Categórica / int64

Finalmente usamos la función `DataFrame.describe()` para poder ver los descriptores estadísticos de los datos.

	<b>age</b>	<b>sex</b>	<b>cp</b>	<b>trtbps</b>	<b>chol</b>	<b>fbs</b>	<b>restecg</b>	<b>thalachh</b>	<b>exng</b>	<b>oldpeak</b>	<b>sip</b>	<b>caa</b>	<b>thall</b>	<b>output</b>
count	303	303	303	303	303	303	303	303	303	303	303	303	303	303
mean	54.36	0.683	0.966	131.62	246.26	0.1485	0.528	149.647	0.326	1.0396	1.39	0.729	2.313	0.5445
std	9.082	0.466	1.03	17.53	51.830	0.356	0.525	22.905	0.469	1.161	0.6162	1.022	0.612	0.498
min	29	0	0	94	126	0	0	71	0	0	0	0	0	0
25%	47.5	0	0	120	211	0	0	133.5	0	0	1	0	2	0
50%	55	1	1	130	240	0	1	153	0	0.8	1	0	2	1
75%	61	1	2	140	274.5	0	1	166	1	1.6	2	1	3	1
max	77	1	3	200	564	1	2	202	1	6.2	2	4	3	1

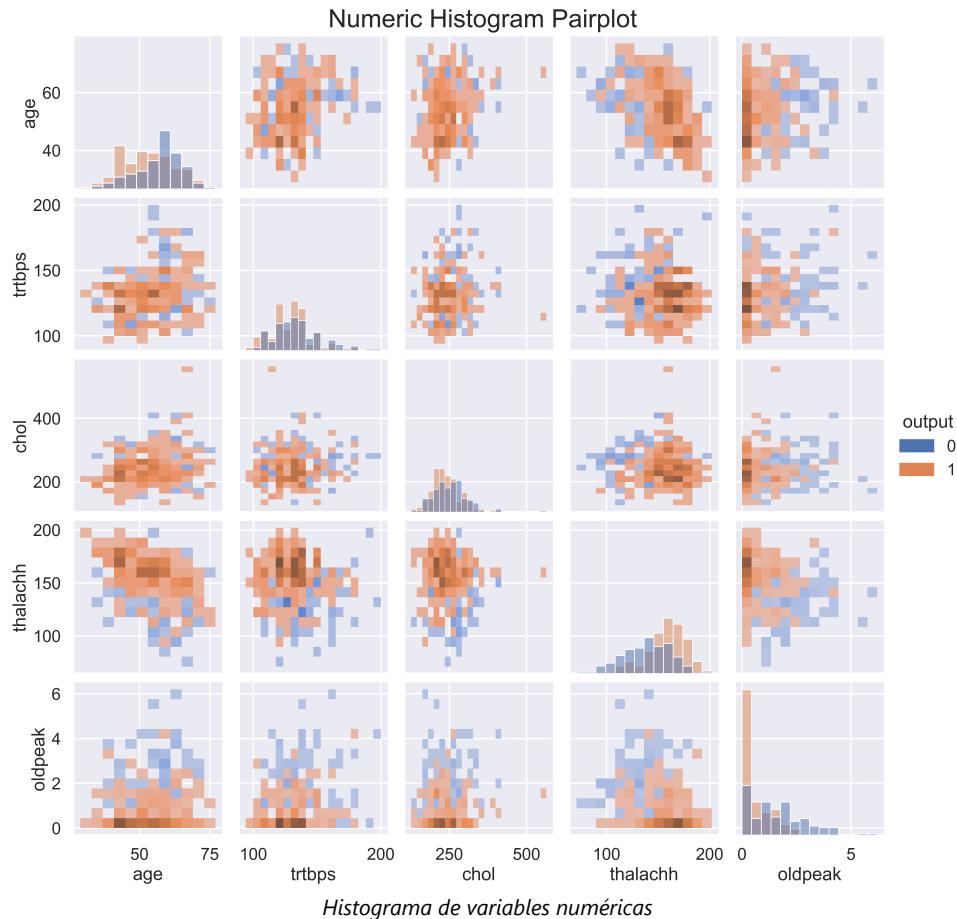
La tabla proporciona los valores de **conteo**, **media**, **desviación estándar**, **mínimo**, **percentil 25**, **percentil 50**, **percentil 75** y **máximo** para cada columna. A continuación se presenta una interpretación estadística de cada columna:

- **Edad:** La edad promedio de los pacientes en el conjunto de datos es de 54.4 años. La edad más joven registrada es de 29 años y la edad más avanzada es de 77 años.
- **Sexo:** El 68.3% de los pacientes son hombres y el 31.7% son mujeres.
- **Cp:** El tipo de dolor en el pecho que experimentan los pacientes se divide en 47.5% de los pacientes con tipo 0, el 40.3% tipo 1, el 9.6% tipo 2 y el 2.6% tipo 3.
- **Trestbps:** La presión arterial en reposo (en mm Hg) de los pacientes tiene una media de 131.6 y una desviación estándar de 17.5. La más baja registrada es de 94 y la más alta es de 200.
- **Chol:** El colesterol sérico (en mg/dl) tiene una media de 246.3 y una desviación estándar de 51.8. El nivel de colesterol más bajo registrado es de 126 y el más alto es de 564.
- **Fbs:** El nivel de azúcar en sangre en ayunas (en mg/dl) tiene una media de 120.9 y una desviación estándar de 30.1. El nivel más bajo registrado es de 78 y el más alto es de 275.
- **Restecg:** El resultado electrocardiográfico en reposo de los pacientes se divide en 50.8% de los pacientes normal, el 47.5% tiene una anormalidad de ST-T y el 1.7% tiene una hipertrofia ventricular izquierda probable o definitiva.
- **Thalach:** La frecuencia cardíaca máxima alcanzada por los pacientes durante el ejercicio tiene una media de 149.6 y una desviación estándar de 22.9. La frecuencia cardíaca más baja registrada es de 71 y la más alta es de 202.
- **Exang:** El 32.7% de los pacientes experimenta angina inducida por ejercicio.
- **Oldpeak:** La depresión del segmento ST inducida por el ejercicio tiene una media de 1.0 y una desviación estándar de 1.2. El valor más bajo es de 0 y el más alto es de 6.2.
- **Slope:** Para pendiente del segmento pico del ejercicio el 46.8% tiene una pendiente ascendente, el 46.8% una pendiente plana y el 6.4% una pendiente descendente.
- **Ca:** El número de vasos principales coloreados por flourosopía tiene una media de 0.7 y una desviación estándar de 1.0. El valor más bajo registrado es de 0 y el más alto es de 4.
- **Thal:** El 54.5% de los pacientes tiene un resultado normal, el 38.1% tiene un resultado defectuoso y el 7.4%

## Visualización de variables

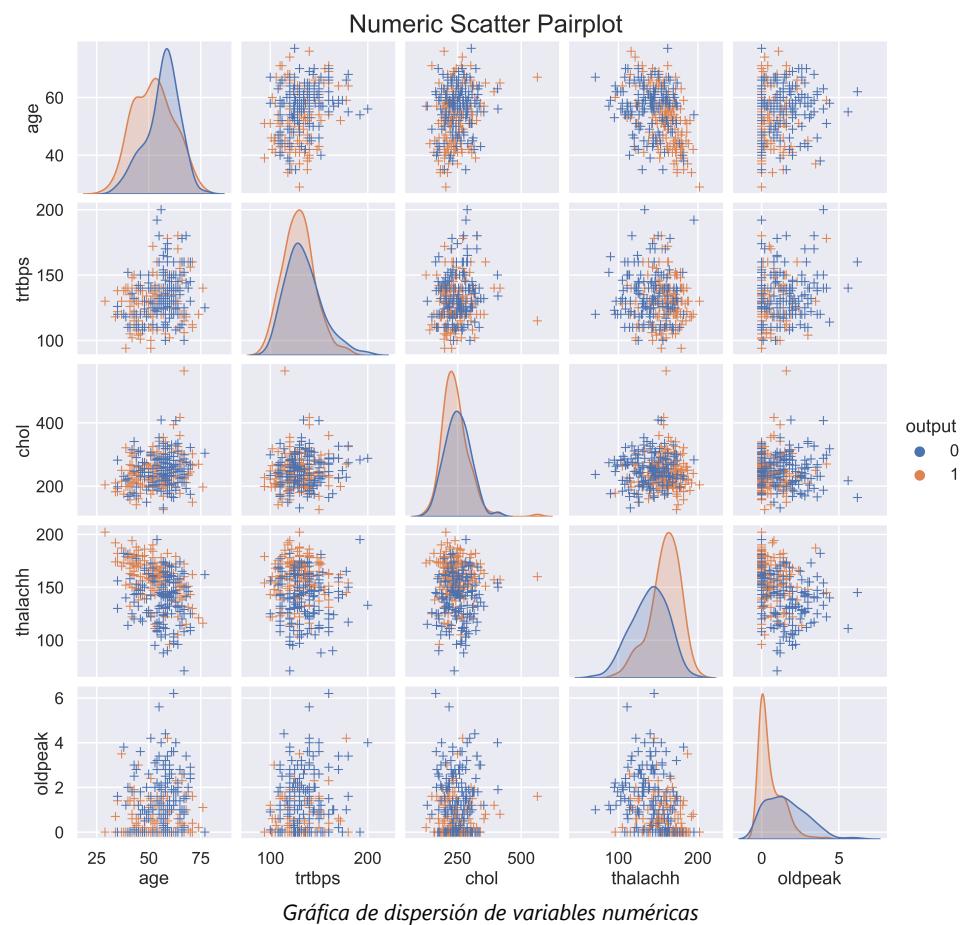
### Variables numéricas

Haciendo uso de la función `seaborn.pairplot(df[numericos], hue='stroke', kind='hist')` se puede representar las variables numéricas comparadas entre sí con respecto a su frecuencia de la siguiente manera (aquellos con riesgo cardíaco representados en naranja y aquellos sin riesgo cardíaco representados en azul):

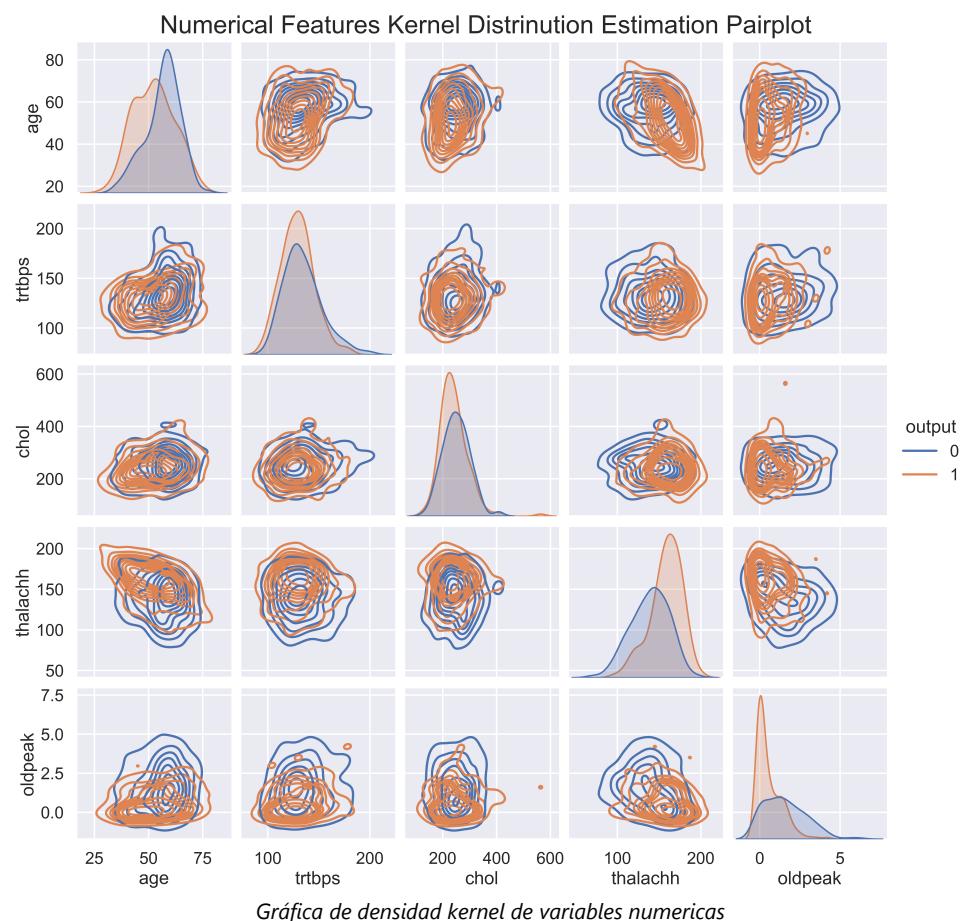


1. **Edad (age):** La distribución de la edad parece ser ligeramente diferente entre los dos grupos. Los individuos con riesgo cardíaco tienden a ser mayores que aquellos sin riesgo cardíaco.
2. **Presión arterial en reposo (trtbps):** Ambos grupos muestran una distribución similar en la presión arterial en reposo. Sin embargo, parece que hay una ligera tendencia hacia valores más altos en el grupo con riesgo cardíaco.
3. **Colesterol sérico en mg/dl (chol):** No parece haber una diferencia significativa en los niveles de colesterol entre los individuos con y sin riesgo cardíaco.
4. **Frecuencia cardíaca máxima alcanzada (thalach):** Los individuos con riesgo cardíaco parecen alcanzar una frecuencia cardíaca máxima más alta en comparación con aquellos sin riesgo cardíaco.
5. **Depresión del ST inducida por el ejercicio en relación con el reposo (oldpeak):** Los individuos con riesgo cardíaco muestran valores más altos de oldpeak en comparación con aquellos sin riesgo cardíaco, lo que indica una mayor depresión del ST durante el ejercicio.

De igual manera, y con las mismas conclusiones, se analizó la distribución de las variables categóricas mediante el uso de una gráfica de dispersión `seaborn.pairplot(df[numericos], hue='stroke', kind='scatter')` donde la diagonal de la cuadrícula muestra la distribución de cada variable con un gráfico de densidad superpuesto y los gráficos fuera de la diagonal muestran la relación entre cada par de variables. Los puntos están coloreados por la variable objetivo.

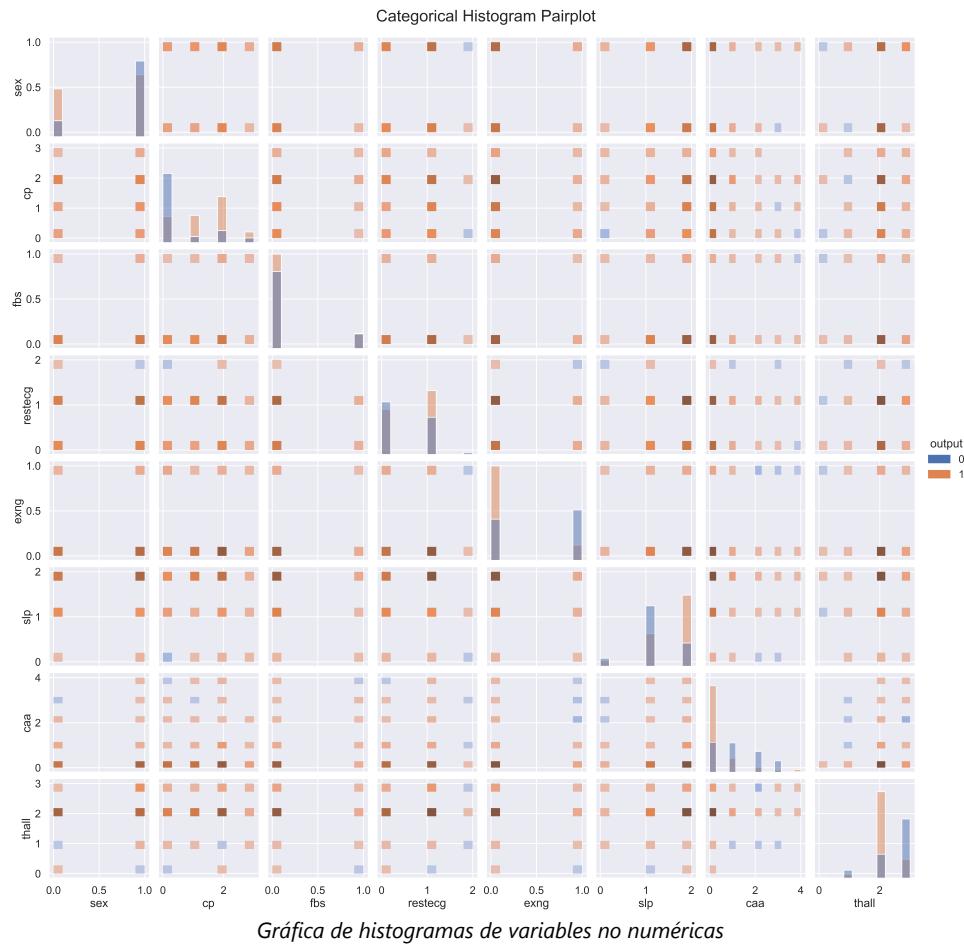


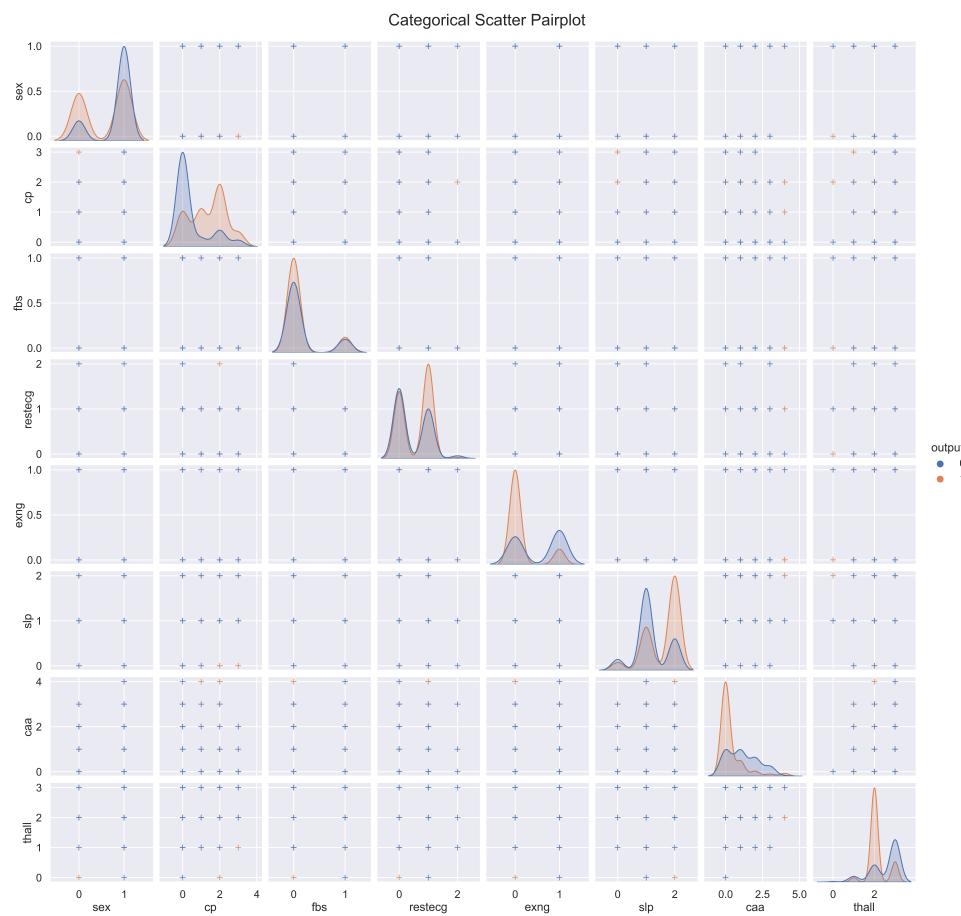
Finalmente se usa la estimación de densidad kernel (KDE) `seaborn.pairplot(df[numericos], hue='stroke', kind='kde')` que a diferencia de los histogramas, proporcionan una estimación suave, continua, y precisa de la densidad de probabilidad. Esto se logra utilizando una función de kernel, que es una función no negativa que se coloca en cada observación de la muestra.



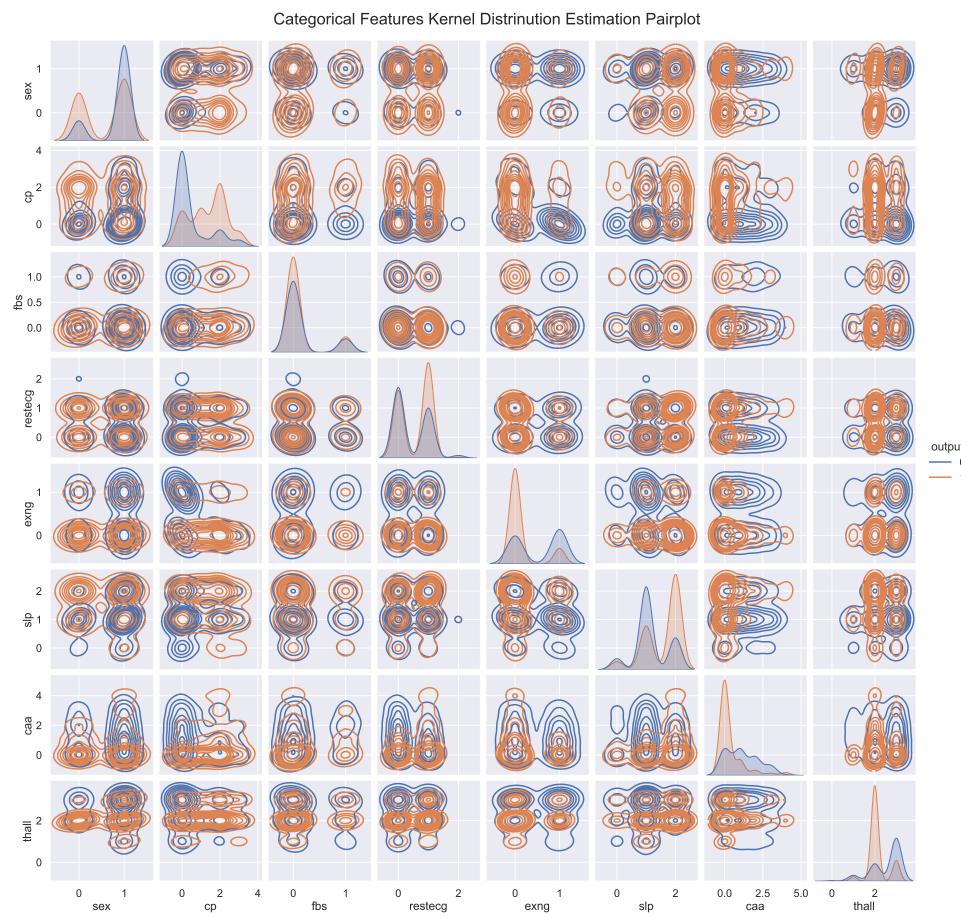
De igual manera se visualizaron las representaciones de variables no numéricas pero no se pueden obtener las mismas conclusiones debido a la falta de significado en las gráficas pues, a diferencia de las variables numéricas continuas, las variables categóricas no muestran una relación directa o patrones claros en los gráficos de dispersión, histogramas y kde. La naturaleza discreta y la ausencia de continuidad en los datos categóricos dificultan la representación visual significativa de tendencias o correlaciones a través de estas formas de visualización.

Se agregan las gráficas de igual manera para completar el reporte.





Gráfica de dispersión de gráficas no numéricas



Gráfica de densidad de kernel de variables no numéricas

## Transformacion de datos

Se utilizó `ColumnTransformer` de Scikit-learn para aplicar transformaciones específicas a diferentes columnas de tu conjunto de datos. Específicamente se implementó `OneHotEncoder` en los datos categóricos; esta técnica convierte variables categóricas en representaciones numéricas binarias, lo que es crucial para que el algoritmo pueda interpretar las variables adecuadamente. Se aplicó en las variables `onehot = ['cp', 'restecg', 'slp', 'caa', 'thall']`

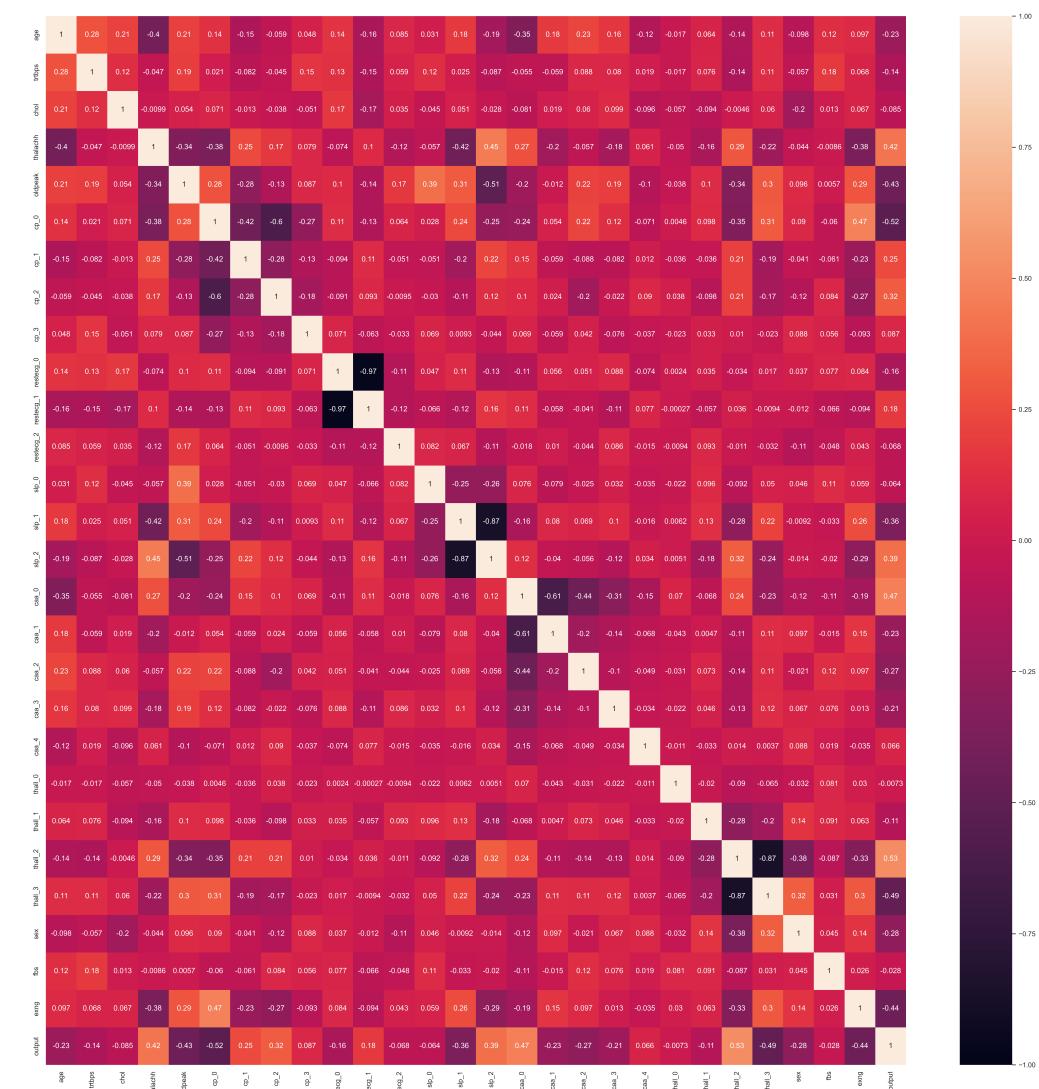
En cuanto al `StandardScaler`, se utilizó para estandarizar las variables numéricas, es decir, para llevarlas a una escala común, lo que ayuda al modelo a interpretar todas las variables con la misma importancia y a evitar que alguna variable domine simplemente debido a su escala original. Se aplicó con las variables `numeric = ['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']`

```
def Transform(data):
    ct = ColumnTransformer(transformers=
        [('numeric', StandardScaler(), numeric),
         ('work', OneHotEncoder(), onehot)],
        remainder='passthrough')

    data_t = pd.DataFrame(ct.fit_transform(data))
    data_t.columns = [s[s.find('_')+2:] for s in list(ct.get_feature_names_out())]
    return data_t
```

## Matriz de correlación

Antes de seguir el procesamiento y comenzar a entrenar se realizó una matriz de correlación para ver cuales podrían ser eliminadas debido a su alta correlación y poder simplificar el modelo. Desafortunadamente para ese objetivo, las variables no resultaron mutuamente correlacionadas; sin embargo, se puede utilizar el valor de relevancia que tienen las variables con la variable objetivo para descartar las que no son relevantes.



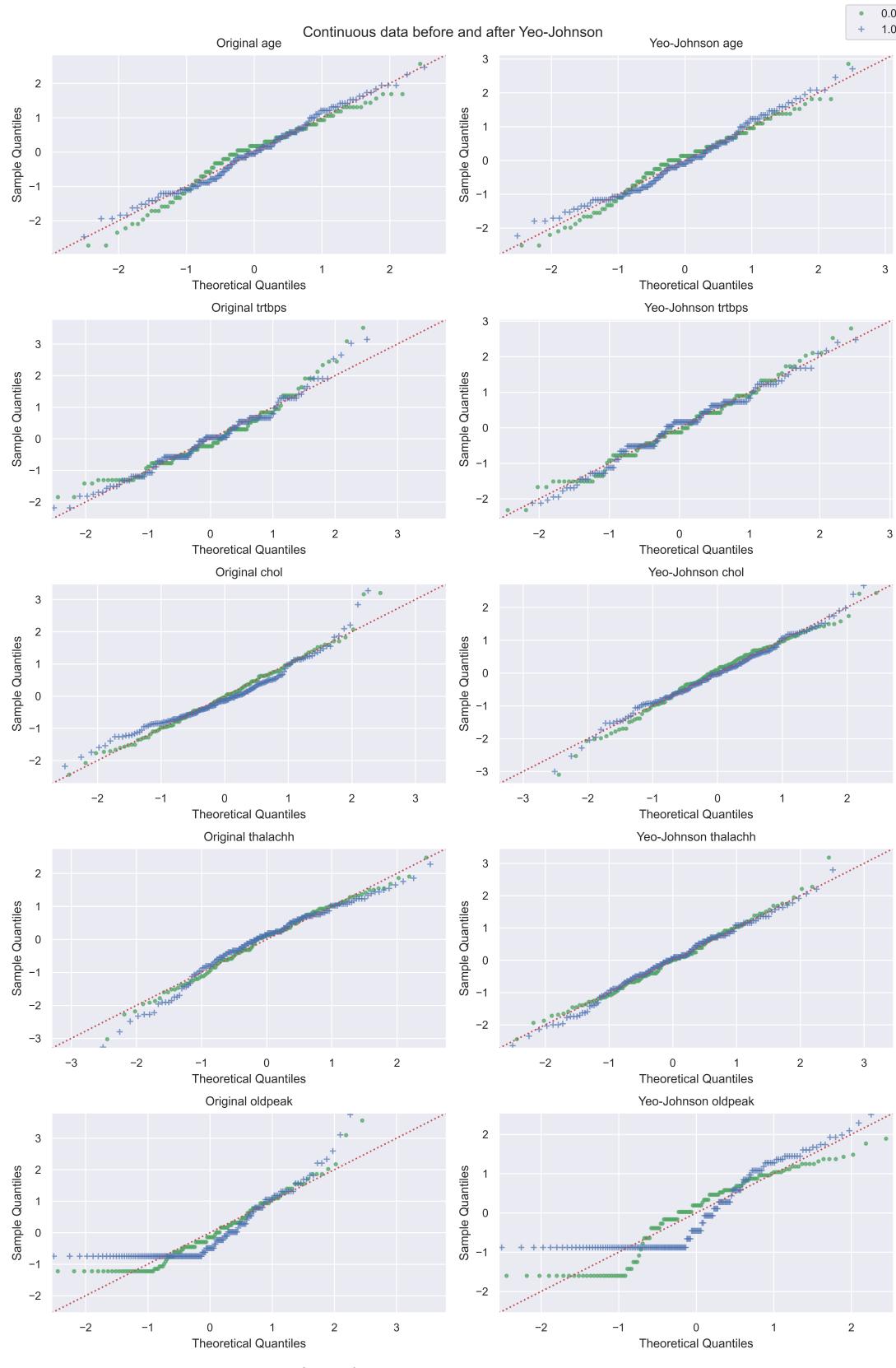
### *Matriz de correlación de las variables*

Se establece las variables no relevantes aquellas que su índice de correlación con respecto a la variable objetivo es menor a **0.05**, descartando las variables `[ 'thalachh', 'fbs' ]` del *DataFrame*

### Yeo-Johnson

Finalmente, el último paso antes de entrenar, es corregir errores de normalidad y asimetría. Para lo cual se usó la transformación Yeo-Johnson para ajustar mejor al próximo modelo estadísticos de aprendizaje automático, mejorando así la validez de los análisis y la precisión de dicho modelo y que aproveche las cualidades gaussianas.

Esta transformación solo se usó en los datos numéricos continuos `numeric = [ 'age', 'trtbps', 'chol', 'thalachh', 'oldpeak' ]`.



Transformación de datos continuos con Yeo-Johnson

## Métodos de validación

Se procedió a la construcción del modelo utilizando la técnica de `LogisticRegression`. Para garantizar una evaluación precisa de su rendimiento, se dividió la información en conjuntos de entrenamiento y prueba. Esta división permitió entrenar el modelo utilizando únicamente los datos de entrenamiento, lo que proporciona al modelo la capacidad de aprender patrones y características subyacentes.

```
# Asumiendo que ya se aplicaron las transformaciones al dataset
X = df.drop(target, axis=1)
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=5338)

lr = LogisticRegression()
lr.fit(X_train, y_train)
```

### Validación por porcentaje

Mediante el empleo de `score(X_test, y_test)`, se logró calcular un puntaje de precisión de **0.85246**. Este puntaje representa la capacidad del modelo para predecir con exactitud los valores en el conjunto de datos de prueba. Un valor de 0.85246 indica que el modelo fue capaz de predecir correctamente aproximadamente el 85.25% de las instancias en el conjunto de prueba, subrayando así su capacidad para generalizar y realizar predicciones precisas sobre datos no vistos.

### Validación por porcentaje con repeticiones

Aplicando la validación cruzada con repeticiones, se obtuvo una precisión promedio del **84.52%**, con una desviación estándar de **0.0444**. Este resultado, derivado de múltiples iteraciones de validación cruzada, muestra que el modelo mantiene un nivel constante de precisión alrededor del **84.52%**, evidenciando su capacidad para realizar predicciones consistentes en diferentes conjuntos de datos. La pequeña desviación estándar indica una consistencia en el rendimiento del modelo durante estas iteraciones, lo que respalda su estabilidad y confiabilidad en la generalización de predicciones. Para este método se uso `ShuffleSplit(n_splits=1000, test_size=0.2)`.

### Validación cruzada y validación cruzada con repeticiones

Al emplear la validación cruzada `KFold(n_splits=10, shuffle=True)`, se logró obtener un puntaje de precisión del 84.84%, con una desviación estándar de **0.0872**, mientras que al aplicar la validación cruzada con repeticiones `ShuffleSplit(n_splits=1000, test_size=1/3)`, se obtuvo un puntaje de precisión del 84.12% con una desviación estándar de **0.0313**. Estos resultados resaltan la consistencia del modelo, mostrando que, en promedio, logra una precisión cercana al \*88.4%\*. Sin embargo, la validación cruzada estándar muestra una desviación estándar ligeramente mayor, una precisión más variada a lo largo de las iteraciones. Y la validación estándar con repeticiones tiene una mayor estabilidad en el rendimiento del modelo en diferentes subconjuntos de datos.

## Métricas para evaluación de algoritmos

### Matriz de confusión

Usando `confusion_matrix` de `sklearn.metrics` se obtiene:

- Verdaderos positivos (TP): 148
- Verdaderos negativos (TN): 112
- Falsos positivos (FP): 26
- Falsos negativos (FN): 17

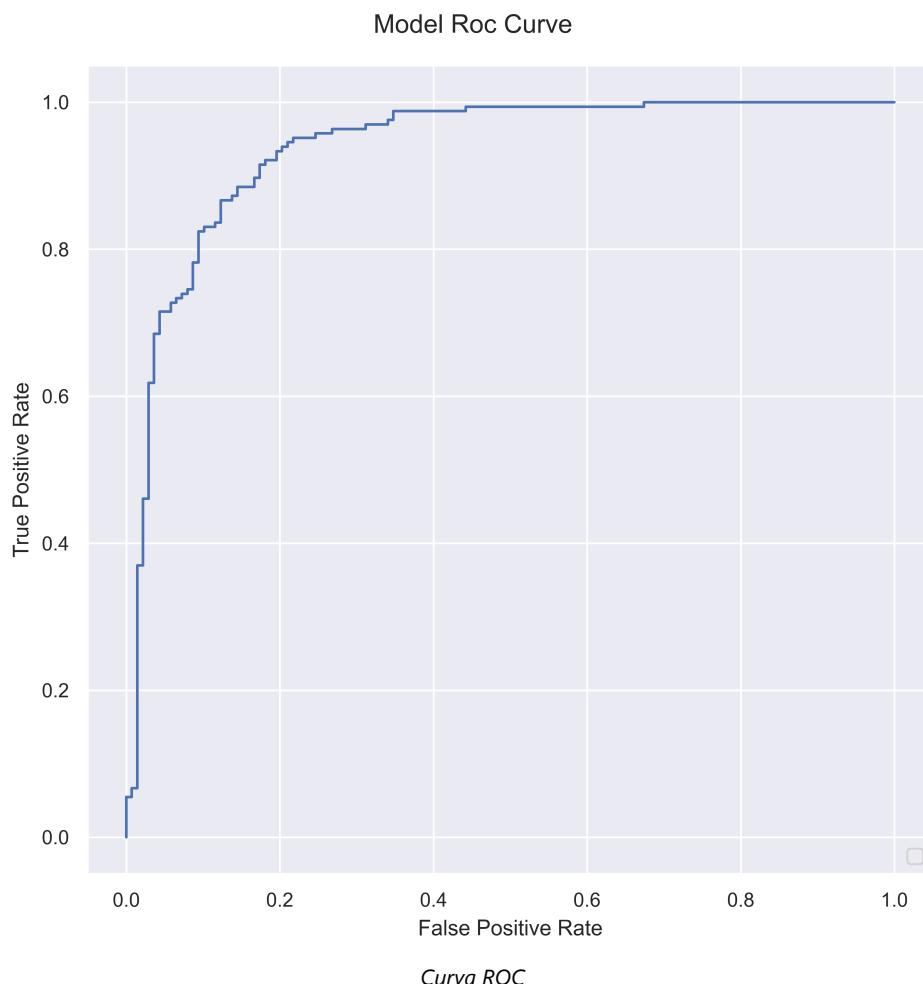
Esto significa que el modelo acertó en clasificar correctamente 148 instancias como positivas y 112 como negativas. Sin embargo, se equivocó al clasificar 26 instancias como positivas cuando en realidad eran negativas (falsos positivos) y 17 como negativas cuando en realidad eran positivas (falsos negativos).

### Cohen Score

Al usar `cohen_kappa_score` obtenemos **0.7124**. Esto indica que el modelo ha logrado un grado notable de precisión más allá de lo que podría explicarse por el azar ya que el puntaje de Cohen's Kappa es una medida de la fiabilidad de la clasificación del modelo, considerando la posibilidad de aciertos simplemente por casualidad, y en este caso, el score muestra una consistencia significativa en las predicciones del modelo.

### Curva ROC

Finalmente se usa 'roc\_curve' para medir el desempeño del algoritmo. La gráfica ROC muestra que el modelo de clasificación binaria tiene un buen desempeño, ya que la curva ROC está muy por encima de la línea diagonal y tiene un AUC cercano a 1. Esto indica que el modelo puede separar las clases con una alta tasa de verdaderos positivos y una baja tasa de falsos positivos.



## Conclusiones

Basado en los análisis realizados, se destaca la solidez del modelo de regresión logística empleado en este estudio. Las evaluaciones, tanto mediante la separación tradicional de datos en conjuntos de entrenamiento y prueba, como a través de técnicas más robustas como la validación cruzada estándar y con repeticiones, han revelado un rendimiento consistente en torno al **84%** de precisión. Este resultado demuestra la capacidad del modelo para generalizar y realizar predicciones precisas en diferentes escenarios, lo que respalda su aplicabilidad en entornos del mundo real.

No obstante, la matriz de confusión revela cierta discrepancia entre los verdaderos positivos y negativos, y los falsos positivos y negativos. Aunque el modelo muestra una capacidad equitativa para clasificar correctamente tanto instancias positivas como negativas, la presencia significativa de falsos positivos y falsos negativos sugiere áreas para la mejora del modelo, especialmente en la reducción de estos errores.

Si bien el modelo creado a partir de los datos trabajados y transformados demostró ser consistente y preciso, se sugiere una optimización adicional para reducir los falsos positivos y falsos negativos, lo que contribuiría a mejorar aún más su capacidad de predicción y su aplicabilidad en situaciones del mundo real.

## Referencias

1. Rahman, R., & Pritom, R. (n.d.). Heart Attack Analysis & Prediction Dataset [Data set]. Kaggle. Retrieved from [Kaggle](#)
2. Scikit-learn: Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830. [Enlace a la documentación de Scikit-learn](#).
3. Seaborn: Waskom, M., 2022. seaborn: statistical data visualization. [Enlace a la documentación de Seaborn](#).