Predicción de supervivencia ante la Insuficiencia Cardiaca
Prediction of Survival in Heart Failure

Choque Forra Daniela Patricia
Mamani Mamani Jessyca Liset
Universidad La Salle
dpcfjcrc@gmail.com
jessycamamani20@gmail.com

Resumen

El presente proyecto tiene como objetivo predecir la supervivencia ante la insuficiencia cardiaca, contando con un dataset que contiene registros clínicos de pacientes con insuficiencia cardiaca. Se utilizara diferentes algoritmos de Aprendizaje Automático de los cuales se seleccionara el mejor modelo para luego utilizarlo en una interfaz gráfica. También se realizara el análisis de datos para observar la influencia que tienen respecto a la Insuficiencia Cardiaca.

Abstract

The objective of this project is to predict survival from heart failure, using a data set that contains clinical records of patients with heart failure. Different Machine Learning algorithms will be used from which the best model will be selected and then used in a graphical interface. Data analysis will also be carried out to observe the influence they have regarding Heart Failure.

Introducción

El proyecto trata sobre la supervivencia ante la insuficiencia cardiaca y la aplicación de modelos de aprendizaje automático para predecir la supervivencia ante la Insuficiencia Cardiaca. Ante ello conoceremos que es la Insuficiencia Cardiaca y que es Aprendizaje Automático

¿Que es la Insuficiencia Cardiaca?

La insuficiencia cardíaca (IC) es un síndrome de disfunción ventricular. La insuficiencia ventricular izquierda causa disnea y fatiga, mientras que la insuficiencia ventricular derecha promueve la acumulación de líquido en los tejidos periféricos y el abdomen. Los ventrículos pueden verse involucrados en forma conjunta o por separado. El diagnóstico inicial se basa en la evaluación clínica y se confirma con radiografías de tórax, ecocardiografía y medición de las concentraciones plasmáticas de péptidos natriuréticos. El tratamiento consiste en la educación del paciente, diuréticos, inhibidores de la enzima convertidora de la angiotensina (ECA), bloqueantes de los receptores de angiotensina II, beta-bloqueantes, antagonistas de la aldosterona, inhibidores de la neprilisina, marcapasos/desfibriladores implantables y otros dispositivos especializados y la corrección de la causa o las causas del síndrome de IC.

Causas

La insuficiencia cardíaca casi siempre es una afección prolongada (crónica), pero se puede presentar repentinamente. Puede ser causada por muchos problemas diferentes del corazón.
La enfermedad puede afectar únicamente el lado derecho o el lado izquierdo del corazón. Más frecuentemente, ambos lados del corazón resultan comprometidos.
La insuficiencia cardíaca ocurre cuando:

- Su miocardio no puede bombear (expulsar) la sangre del corazón muy bien. Esto se denomina insuficiencia cardíaca sistólica o insuficiencia cardíaca con una fracción de eyección reducida (HFrEF, por sus siglas en inglés).
- El miocardio está rígido y no se llena de sangre fácilmente. Esto se denomina insuficiencia cardíaca diastólica o insuficiencia cardíaca con una eyección preservada (HFpEF, por sus siglas en inglés).

¿Qué es Aprendizaje Automático?

El proceso de aprendizaje automático es similar al de la minería de datos. Ambos sistemas buscan entre los datos para encontrar patrones. Sin embargo, en lugar de extraer los datos para la comprensión humana –como es el caso de las aplicaciones de minería de datos– el aprendizaje automático utiliza esos datos para detectar patrones en los datos y ajustar las acciones del programa en consecuencia. Los algoritmos del aprendizaje automático se clasifican a menudo como supervisados o no supervisados. Los algoritmos supervisados pueden aplicar lo que se ha aprendido en el pasado a nuevos datos. Los algoritmos no supervisados pueden extraer inferencias de conjuntos de datos.

Conociendo ambos conceptos se puede empezar con el objetivo del proyecto.

Descripción de datos

El presente proyecto para la "Predicción de Supervivencia ante la Insuficiencia Cardiaca" utilizara un dataset que contiene registros clínicos de pacientes con insuficiencia cardiaca. Este dataset cuenta con los siguientes datos:

- **Age:** La edad del paciente
- **Anaemia:** Si el paciente tiene anemia
- **Creatinine_phosphokinase:** El número de Creatinina fosfoquinasa que tiene el paciente
- **Diabetes:** Si el paciente tiene Diabetes
- **Ejection_fraction:** La fracción de eyección del corazón del paciente
- **High_blood_pressure:** Si el paciente tiene hipertensión
- **Platelets:** El número de plaquetas del paciente
- **Serum_creatinine:** El nivel de Suero de creatinina en el paciente
- **Serum_sodium:** El nivel de Sodio sérico en el paciente
- **Sex:** El género del paciente
- **Smoking:** Si el paciente fuma
- **Time:** Días de tratamiento del paciente
- **DEATH_EVENT:** Si el paciente sobrevive o no

Análisis de datos

Importación del Dataset

Se importa el Dataset que utilizaremos en este proyecto el cual es "Registros clínicos de insuficiencia cardíaca"

Especificación de resultados booleanos del dataset

- Género: Masculino = 1, Femenino =0
- Diabetes: No = 0, Si= 1
- Anemia: No = 0, Si= 1
- Alta presión en la sangre: No = 0, Si= 1
- Resultado si fuma el paciente: No = 0, Si= 1
- Sobrevive el paciente: No = 1, Si= 0

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataset = pd.read_csv('heart_failure_clinical_records_dataset.csv')
dataset
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
294	62.0	0	61	1	38	1	155000.00	1.1	143	1	1	270	0
295	55.0	0	1820	0	38	0	270000.00	1.2	139	0	0	271	0
296	45.0	0	2060	1	60	0	742000.00	0.8	138	0	0	278	0
297	45.0	0	2413	0	38	0	140000.00	1.4	140	1	1	280	0
298	50.0	0	196	0	45	0	395000.00	1.6	136	1	1	285	0

299 rows × 13 columns

Como se puede observar nuestro dataset cuenta con 299 registros clínicos con los datos ya mencionados

División aleatoria del Dataset al 90%

Se realiza una separación de datos donde se tomara aleatoriamente el 90% de los datos para el análisis y entrenamiento, el 10% será utilizado para prueba.

```
p_train = 0.90 # Porcentaje de train.
```

```
# Datos Aleatorios
dataset['is_train'] = np.random.uniform(0, 1, len(dataset)) <= p_train
train, test = dataset[dataset['is_train']==True], dataset[dataset['is_train']==False]
dataset = dataset.drop('is_train', 1)

print("Ejemplos usados para entrenamiento: ", len(train))
print("Ejemplos usados para prueba: ", len(test))
```

```
Ejemplos usados para entrenamiento:  270
Ejemplos usados para prueba:  29
```

Comienzo del Análisis de datos

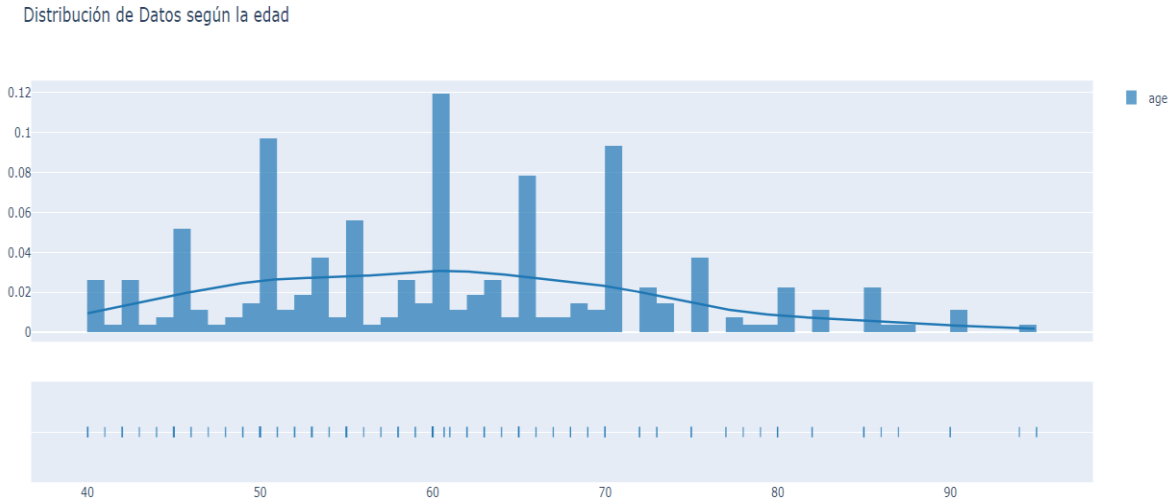
Empezaremos a analizar el dataset tomando algunos factores para observar su influencia en la predicción de la supervivencia ante la Insuficiencia cardiaca. Se acompañara con gráficos

¿Son la edad y el género un factor importante para predecir la muerte ante IC?

En este primer caso se está tomando la edad y genero del paciente para observar si es un factor importante para predecir la muerte ante IC.

```
import plotly.figure_factory as ff
import plotly
edad_datos =[train["age"].values]
edad_grupo = ['age']  #Nombre del Conjunto de Datos
figura = ff.create_distplot(edad_datos, edad_grupo)
figura.update_layout(title_text='Distribución de Datos según la edad')

figura.show()
```

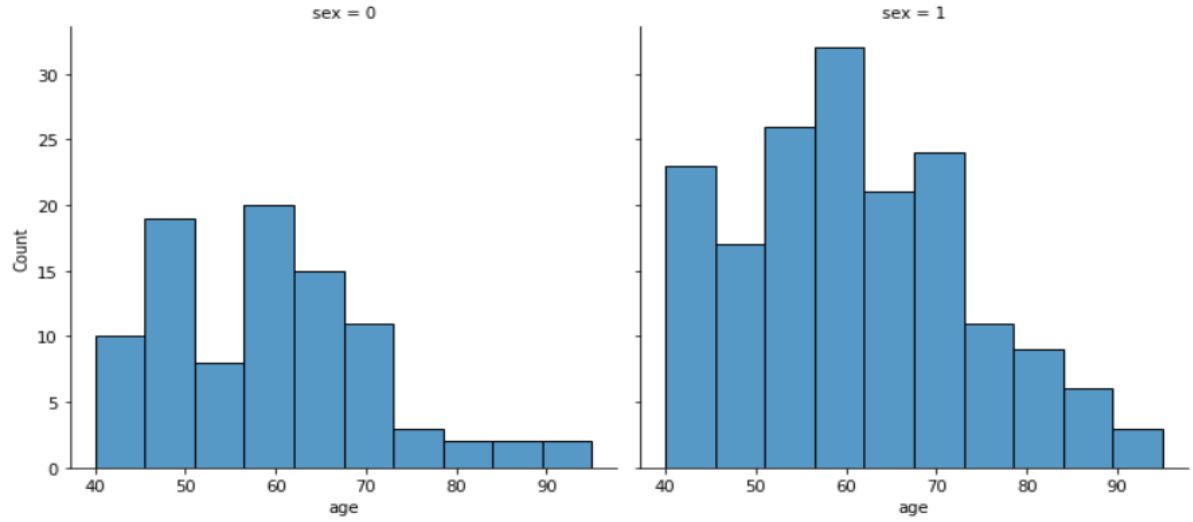


Conclusión: Según el grafico se puede observar que se tienen más pacientes entre 50 a 72 años

```
import plotly.express as px
import seaborn as sns

sns.displot(train, x="age", col="sex", multiple="dodge")
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd4270e2240>
```



Conclusión: Se observa que se tiene más pacientes hombres. Lo cual indica que tienden a padecer IC

Análisis de porcentaje de Supervivencia según el género

Se selecciona el género y sexo. Se le asigna a cada uno el evento de muerte para determinar cuántos hombres y mujeres sobreviven o no sobreviven ante la IC

```
hombre = train[train["sex"]==1]
mujer = train[train["sex"]==0]

supervivencia_H = hombre[train["DEATH_EVENT"]==0]
NoSupervivencia_H = hombre[train["DEATH_EVENT"]==1]
Supervivencia_M = mujer[train["DEATH_EVENT"]==0]
NoSupervivencia_M = mujer[train["DEATH_EVENT"]==1]

etiquetas = 'supervivencia_H', 'NoSupervivencia_H', 'Supervivencia_M', 'NoSupervivencia_M'
porcentaje = [len(hombre[train["DEATH_EVENT"]==0]),len(hombre[train["DEATH_EVENT"]==1]),
         len(mujer[train["DEATH_EVENT"]==0]),len(mujer[train["DEATH_EVENT"]==1])]

fig1, ax1 = plt.subplots()
ax1.pie(porcentaje, labels=etiquetas, autopct='%1.1f%%', shadow=True, startangle=90)
ax1.axis('equal')

plt.title("Supervivencia según el género")
plt.show()
```

Conclusión: Según el grafico se puede observar que:

- El 22.7% de las mujeres sobreviven ante la IC
- El 12.1% de las mujeres no sobreviven ante la IC
- El 44.3% de los hombres sobreviven ante la IC
- El 20.8% de los hombres no sobreviven ante la IC

¿Qué tan probable es que una persona con diabetes o sin diabetes sobreviva ante la IC?

La diabetes está estrechamente asociada a la insuficiencia cardíaca y, de hecho, se estima que cerca del 40% de los pacientes con insuficiencia cardiaca son diabéticos. A su vez, la diabetes acelera la evolución de esta enfermedad cardiaca.

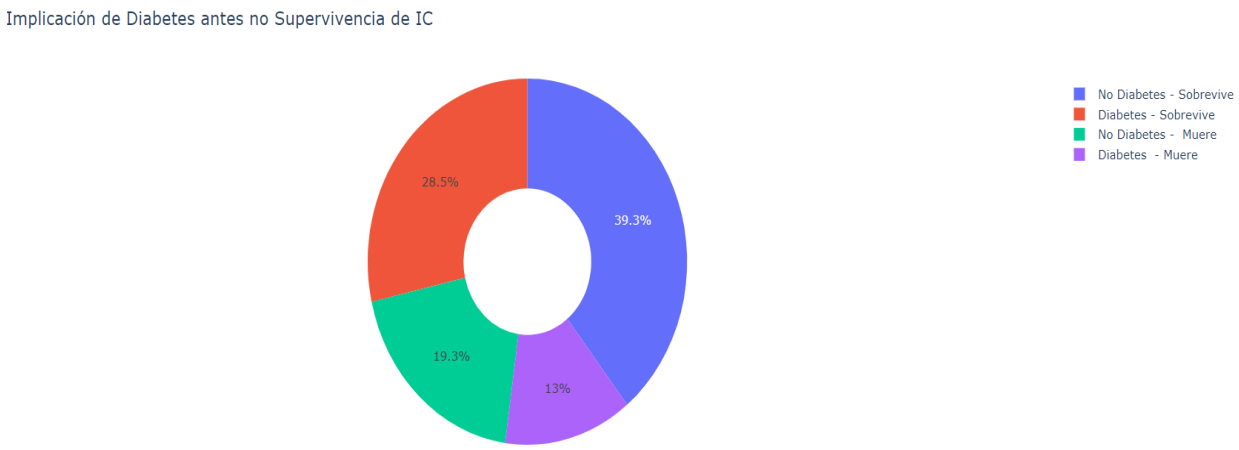
Se selecciona los datos donde una persona con diabetes o sin diabetes sobrevive o no sobrevive ante la IC

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

d1 = train[(train["DEATH_EVENT"]==0) & (train["diabetes"]==0)]
d2 = train[(train["DEATH_EVENT"]==0) & (train["diabetes"]==1)]
d3 = train[(train["DEATH_EVENT"]==1) & (train["diabetes"]==0)]
d4 = train[(train["DEATH_EVENT"]==1) & (train["diabetes"]==1)]


label2 = ['No Diabetes - Sobrevive','Diabetes - Sobrevive', "No Diabetes -  Muere", "Diabetes  - Muere"]
values2 = [len(d1),len(d2),len(d3),len(d4)]

fig = go.Figure(data=[go.Pie(labels=label2, values=values2, hole=.4)])
fig.update_layout(title_text="Implicación de Diabetes antes no Supervivencia de IC")
fig.show()
```



Conclusión: Según el grafico se puede observar que:

- El 39.3% de personas que no tienen diabetes sobreviven ante la IC
- El 28.5% de personas que tienen diabetes sobreviven ante la IC
- El 19.3% de personas que no tienen diabetes no sobreviven ante la IC
- El 20.8% de personas que tienen diabetes no sobreviven ante la IC

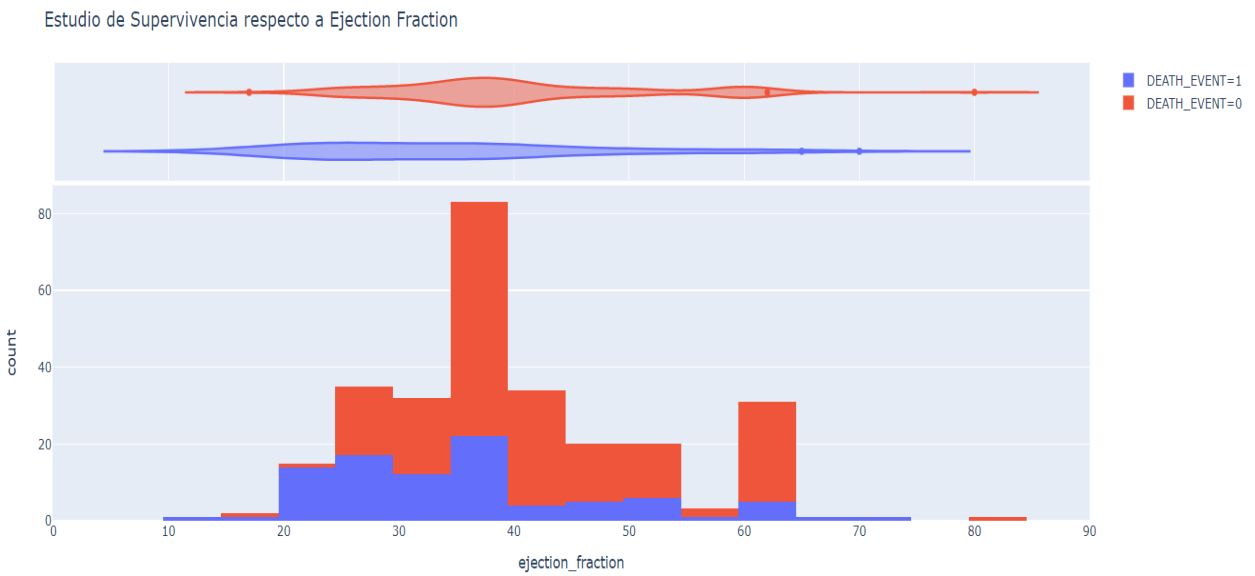
¿Qué tan probable es que una persona con un bajo porcentaje de sangre que sale del corazón en cada contracción no sobreviva ante la IC?

La fracción de eyección es una medida que puede medir la salud del corazón. Un número bajo de fracción de eyección puede ser un indicador de insuficiencia cardíaca
Números de fracción de eyección:

- 50 a 70%: función cardíaca normal
- 40 a 55%: función cardíaca por debajo de lo normal. Puede indicar daño cardíaco previo por ataque cardíaco o miocardiopatía.
- Más del 75%: puede indicar una afección cardíaca como la miocardiopatía hipertrófica, una causa común de paro cardíaco repentino.
- Menos del 40%: puede confirmar el diagnóstico de insuficiencia cardíaca.

Se selecciona el dato de fracción de eyección y se lo relaciona con el dato de evento de muerte

```
porcentajeS = px.histogram(train, x="ejection_fraction", color="DEATH_EVENT", marginal="violin", hover_data=train.columns)
porcentajeS.update_layout(title_text="Estudio de Supervivencia respecto a Ejection Fraction")
porcentajeS.show()
```



Conclusión: Se puede observar que una mayoría se encuentra debajo de lo normal. Y a causa de que un porcentaje que está debajo de lo normal no han sobrevivido ante la IC.

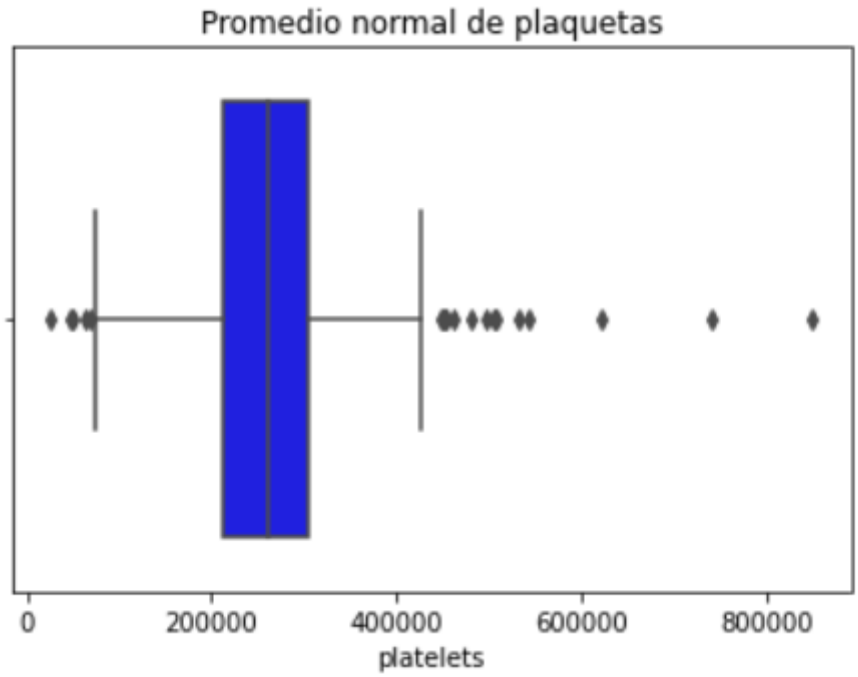
¿Son el desbalance en el promedio normal de plaquetas una razón para determinar la Supervivencia ante IC?

Las plaquetas, son células sanguíneas producidas en la médula ósea y que son responsables por el proceso de coagulación sanguínea.

La cantidad normal de plaquetas en la sangre es de 150,000 a 450,000 por microlitro (mcL) o 150 a 400 × 109/L. Los rangos de los valores normales pueden variar ligeramente.

Se selecciona el dato de número de plaquetas en la sangre para observar si los pacientes se encuentran o no dentro de lo normal.

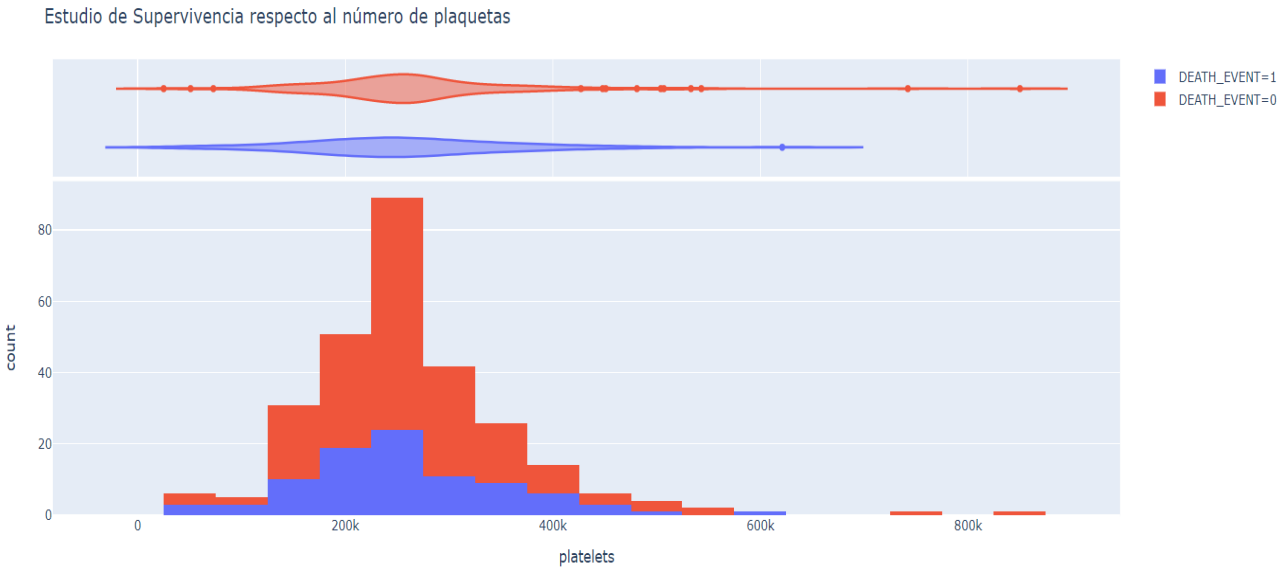
```
sns.boxplot(x = train.platelets, color = 'blue')
plt.title("Promedio normal de plaquetas")
plt.show()
```



Conclusión: Observamos que una mayoría se encuentra dentro de lo normal en el número de plaquetas en la sangre y se encuentran pocos valores atípicos.

Ahora seleccionamos el dato que contiene el número de plaquetas en la sangre relacionándolo con el dato de evento de muerte.

```
porcentajeP = px.histogram(train, x="platelets", color="DEATH_EVENT", marginal="violin", hover_data=train.columns)
porcentajeP.update_layout(title_text="Estudio de Supervivencia respecto al número de plaquetas")
porcentajeP.show()
```



Conclusión: En la gráfica se puede observar que un porcentaje sobrevive ante la IC ya que se encuentra dentro de lo normal. Pero otro porcentaje que no se encuentra de lo normal tiende a no sobrevivir.

¿Es la hipertensión un factor importante ante la supervivencia de IC?

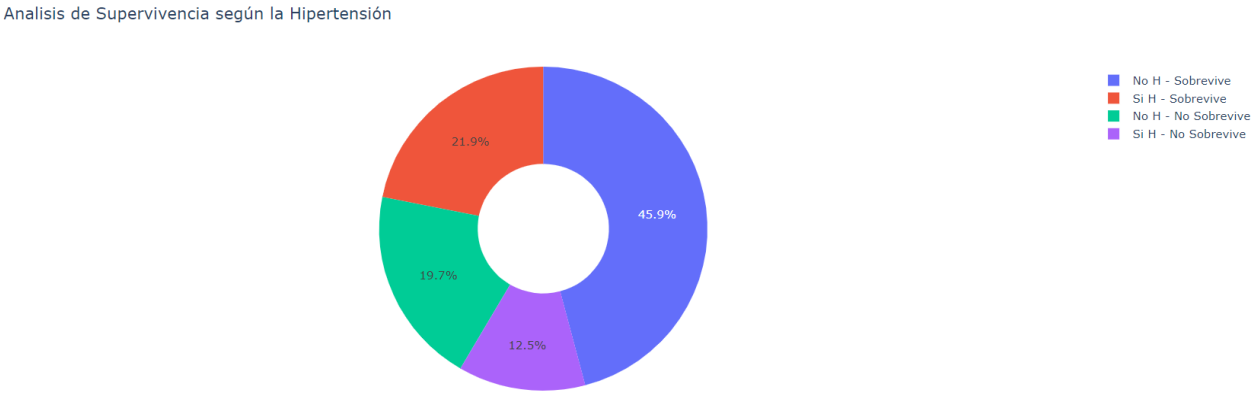
El corazón es el órgano encargado de bombear la sangre oxigenada a través de las arterias hacia todo el organismo. Al avanzar, la sangre ejerce una presión contra las paredes de las arterias, que se mide como presión arterial.
La hipertensión arterial se define por la detección de promedios de la presión arterial sistólica (“máxima”) y/o diastólica (“mínima”) por encima de los límites establecidos como normales. Dicho límite es de 140 mmHg para la sistólica y de 90 mmHg para la diastólica.

Se selecciona los datos donde una persona con hipertensión o sin hipertensión sobrevive o no sobrevive ante la IC

```
SiH = train[train['high_blood_pressure']==1]
NoH = train[train['high_blood_pressure']==0]

SiHSobrevive= SiH[train["DEATH_EVENT"]==0]
SiHNoSobrevive = SiH[train["DEATH_EVENT"]==1]
NoHSobrevive = NoH[train["DEATH_EVENT"]==0]
NoHNoSobrevive = NoH[train["DEATH_EVENT"]==1]

labels = ['Si H - Sobrevive','Si H - No Sobrevive', 'No H - Sobrevive', 'No H - No Sobrevive']
values = [len(SiH[train["DEATH_EVENT"]==0]),len(SiH[train["DEATH_EVENT"]==1]),
         len(NoH[train["DEATH_EVENT"]==0]),len(NoH[train["DEATH_EVENT"]==1])]
Hip = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.4)])
Hip.update_layout(
    title_text="Analisis de Supervivencia según la Hipertensión")
Hip.show()
```



Conclusión: Según el grafico se puede observar que:
- El 45.9% de personas que no tienen hipertensión sobreviven ante la IC
- El 21.9% de personas que tienen hipertensión sobreviven ante la IC
- El 19.7% de personas que no tienen hipertensión no sobreviven ante la IC
- El 12.5% de personas que tienen hipertensión no sobreviven ante la IC

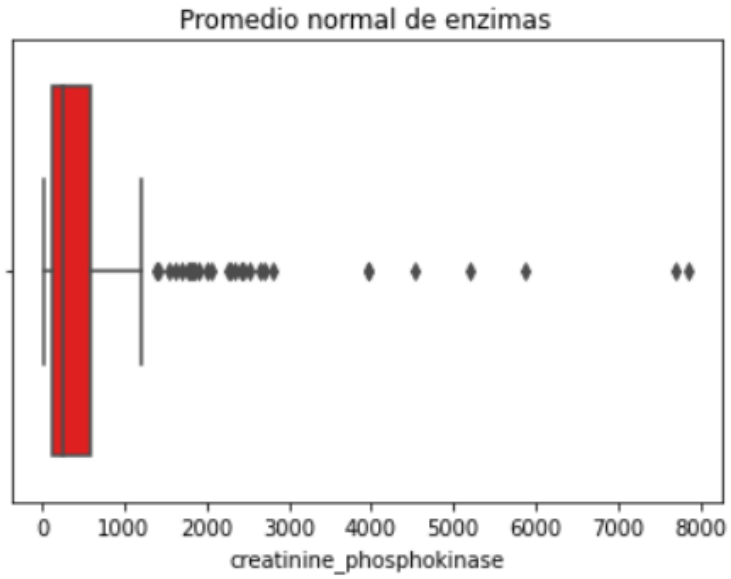
¿Es el alto nivel de enzimas un factor para la No Supervivencia ante la IC?

Las enzimas son proteínas complejas que producen un cambio químico específico en todas las partes del cuerpo. Por ejemplo, pueden ayudar a descomponer los alimentos que consumimos para que el cuerpo los pueda usar. La coagulación de la sangre es otro ejemplo del trabajo de las enzimas.
Normalmente se encuentran niveles bajos de estas proteínas y enzimas en la sangre, pero si el músculo cardíaco está lesionado, como por un ataque cardíaco, las proteínas y enzimas se escapan de las células dañadas del músculo cardíaco y aumentan sus niveles en el torrente sanguíneo.

Valores normales de CPK total: de 10 a 120 microgramos por litro (mcg/L).

Se selecciona el dato del nivel de encimas para observar si los pacientes se encuentran o no dentro de lo normal.

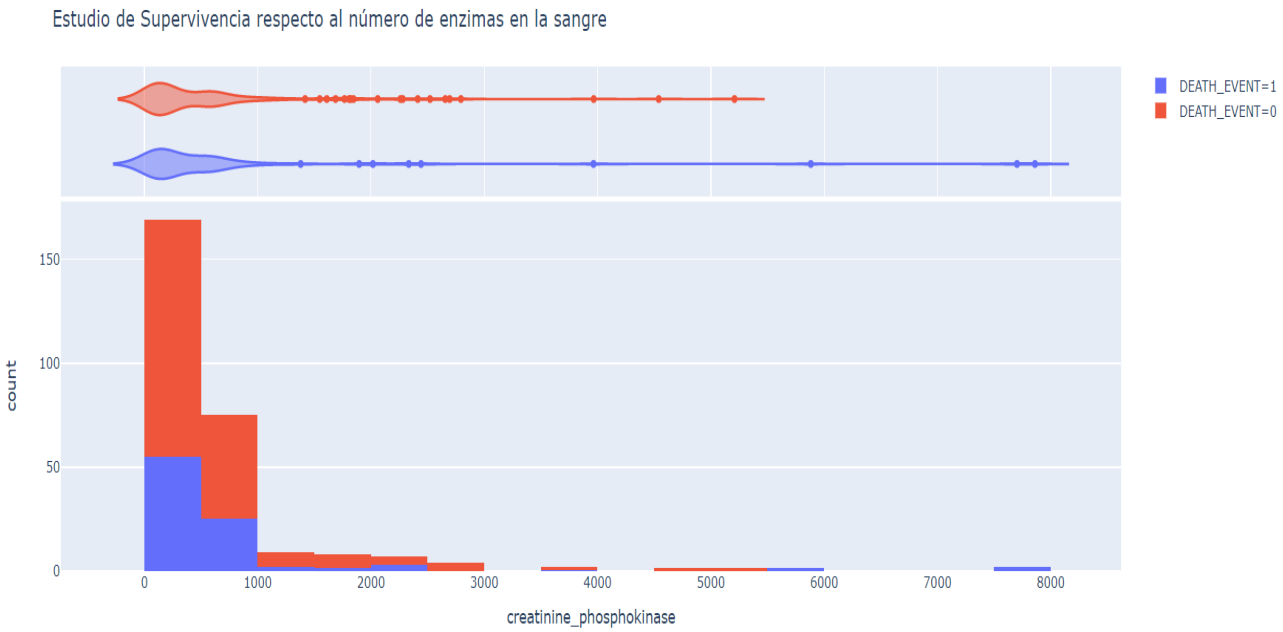
```
sns.boxplot(x = train.creatinine_phosphokinase, color = 'red')
plt.title("Promedio normal de enzimas")
plt.show()
```



Conclusión: Según la gráfica observamos que una mayoría se encuentra dentro de lo normal en el nivel de enzimas y se encuentran pocos valores atípicos.

Ahora seleccionamos el dato que contiene el nivel de enzimas relacionándolo con el dato de evento de muerte.

```
porcentajeE = px.histogram(train, x="creatinine_phosphokinase", color="DEATH_EVENT", marginal="violin", hover_data=train.columns)
porcentajeE.update_layout(title_text="Estudio de Supervivencia respecto al número de enzimas en la sangre")
porcentajeE.show()
```



Conclusión: En la gráfica se puede observar que un gran porcentaje sobrevive ante la IC ya que se encuentra dentro de lo normal en el nivel de enzimas

¿Es el alto nivel de creatinina sérica un factor para la No Supervivencia ante la IC?

La creatinina es un producto de desecho generado por los músculos como parte de la actividad diaria. Normalmente, los riñones filtran la creatinina de la sangre y la expulsan del cuerpo por la orina.

Un resultado normal de nivel de creatinina sérica es de 0.7 a 1.3 mg/dL (de 61.9 a 114.9 µmol/L) para los hombres y de 0.6 a 1.1 mg/dL (de 53 a 97.2 µmol/L) para las mujeres. Las mujeres con frecuencia tienen niveles de creatinina más bajos que los hombres. Esto se debe a que ellas frecuentemente tienen menor masa muscular.

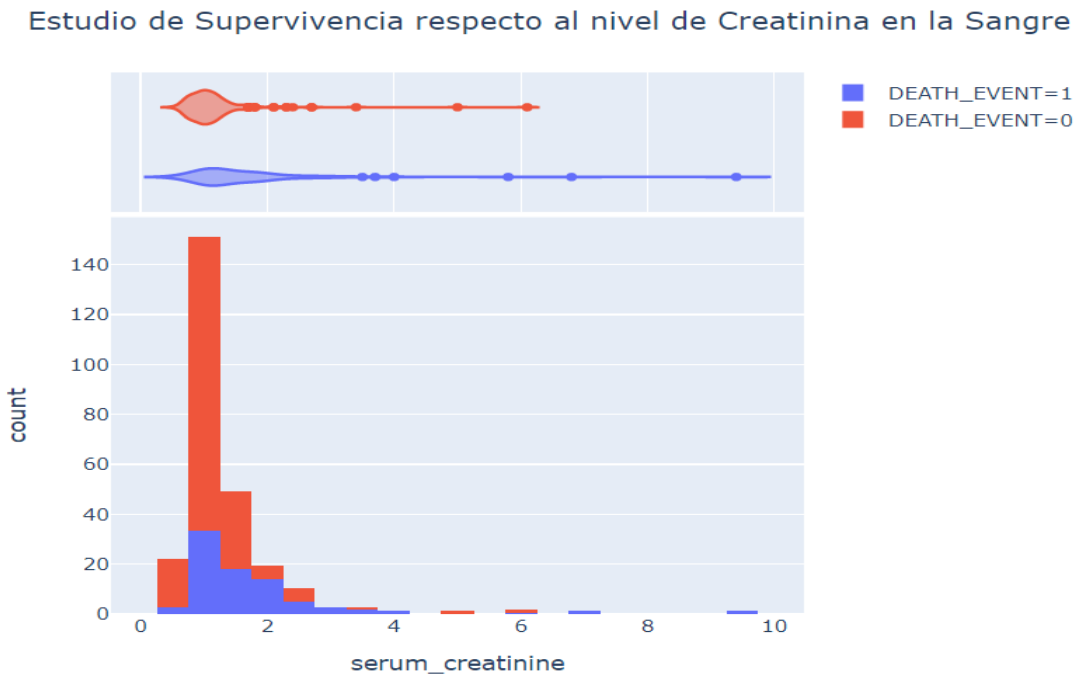
Se selecciona el dato del nivel de creatinina sérica para observar si los pacientes se encuentran o no dentro de lo normal.

```
sns.boxplot(x = train.serum_creatinine, color = 'purple')
plt.title("Promedio normal de Creatina Séricia")
plt.show()
```

Conclusión: Según la gráfica observamos que una mayoría se encuentra dentro de lo normal en el nivel de creatinina sérica y se encuentran pocos valores atípicos.

Ahora seleccionamos el dato que contiene el nivel de creatinina sérica relacionándolo con el dato de evento de muerte.

```
porcentajeC = px.histogram(train, x="serum_creatinine", color="DEATH_EVENT", marginal="violin", hover_data=train.columns)
porcentajeC.update_layout(title_text="Estudio de Supervivencia respecto al nivel de Creatina en la Sangre")
porcentajeC.show()
```



Conclusión: En la gráfica se puede observar que un gran porcentaje sobrevive ante la IC ya que se encuentra dentro de lo normal en el nivel de creatinina sérica

Algoritmo de Predicción

En el presente proyecto se utilizara 5 algoritmos de predicción y se identificara cual es más preciso para la predicción ante la IC.

Entrenamiento de Datos para definir la Supervivencia de un Paciente con IC

Primeramente se importara las librerías que utilizaremos como:

- **Pandas:** Es una biblioteca de software escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.
- **Numpy:** Es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- **Matplotlib:** Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar a la de MATLAB.

También se importara el dataset que contiene los registros clínicos de personas que tiene IC

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

Datos seleccionados para el entrenamiento

Se seleccionaran los siguientes datos para realizar nuestro entrenamiento ya que sé que se consideran importantes.

- Age: La edad, ya que se tienen más pacientes de la tercera edad.
- Sex: El sexo ya que se observó según el análisis de datos que hay diferencia de sobrevivir ante la IC entre hombres y mujeres.
- Creatinine_phosphokinase: El nivel de enzimas más alto de lo normal, podría significar que tiene inflamación del músculo cardíaco (miocardio) o que está teniendo o ha tenido recientemente un ataque al corazón.
- Ejection_fraction: Una medición de la fracción de eyección inferior al 40% puede ser evidencia de insuficiencia cardíaca o miocardiopatía.
- High_blood_pressure: La hipertensión supone una mayor resistencia para el corazón, que responde aumentando su masa muscular (hipertrofia ventricular izquierda). A la larga, está hipertrofia patológica (al contrario de la hipertrofia que se produce con el ejercicio) acaba siendo perjudicial, pudiendo producir angina, arritmias e insuficiencia cardíaca.
- Platelets: Ya que tener un numero fuera de lo normal de plaquetas en la sangre puede producir un ataque cardiaco
- Serum_creatinine: Un nivel alto de creatinina puede ser a causa de que un paciente tiene hipertensión
- Time: Los días de tratamiento son importantes ya que a menores días de tratamiento es muy probable que el paciente no sobreviva, mientras que a mayores días de tratamiento es más probable que el paciente si sobreviva.

Importación de los módulos y división aleatoria del Dataset al 80%

Se importa los diferentes módulos que utilizaremos de la librería sklearn

Scikit-learn es la librería más útil para Machine Learning en Python, proporciona algoritmos de aprendizaje supervisados y no supervisados.

Aquí tomaremos el conjunto del dataset dividido al 80% de los datos donde el 20% serán los datos de prueba, es decir, son aquellos datos que no vio el algoritmo de predicción.

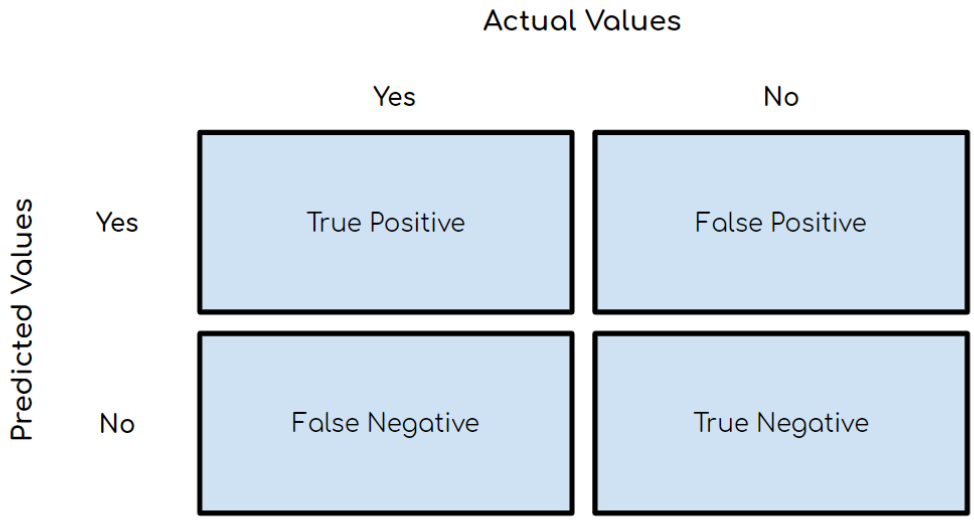
Se seleccionan los datos que tomaremos para el entrenamiento del dataset
Se utiliza el train_test_split. La función train_test_split nos permite dividir un dataset en dos bloques, típicamente bloques destinados al entrenamiento y validación del modelo (llamemos a estos bloques "bloque de entrenamiento " y "bloque de pruebas").

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix

Datos = ['age','sex','creatinine_phosphokinase','ejection_fraction','high_blood_pressure','platelets','serum_creatinine','time']
x = dataset[Datos]
y = dataset['DEATH_EVENT']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=2)
resultados=[]
```

En estos modelos se utilizara la función Accuracy_score y una matriz de confusión

- **Accuracy score:** Es la exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado.
- **Matriz de confusión:** La matriz de confusión es una herramienta muy útil para valorar cómo de bueno es un modelo clasificación basado en aprendizaje automático. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra, lo cual nos, permite trabajar de forma separada con distintos tipos de error.



Modelo de Regresión Logística

La regresión logística es una técnica de aprendizaje supervisado para clasificación. Es muy usada en muchas industrias debido a su escalabilidad y explicabilidad.

¿Cómo se está utilizando el modelo de Regresión Logística?
Se crea una instancia de la Regresión Logística:

- reg_log = LogisticRegression()

Datos de entrenamiento.

- x_train serán los datos de entrada
- y_train los de salida

Se entrena la regresión logística con los datos de entrenamiento

- reg_log.fit(x_train,y_train)

Se usa el modelo entrenado para obtener las predicciones con datos nuevos

- reg_log_pred = reg_log.predict(x_test)

Se almacena el porcentaje de exactitud

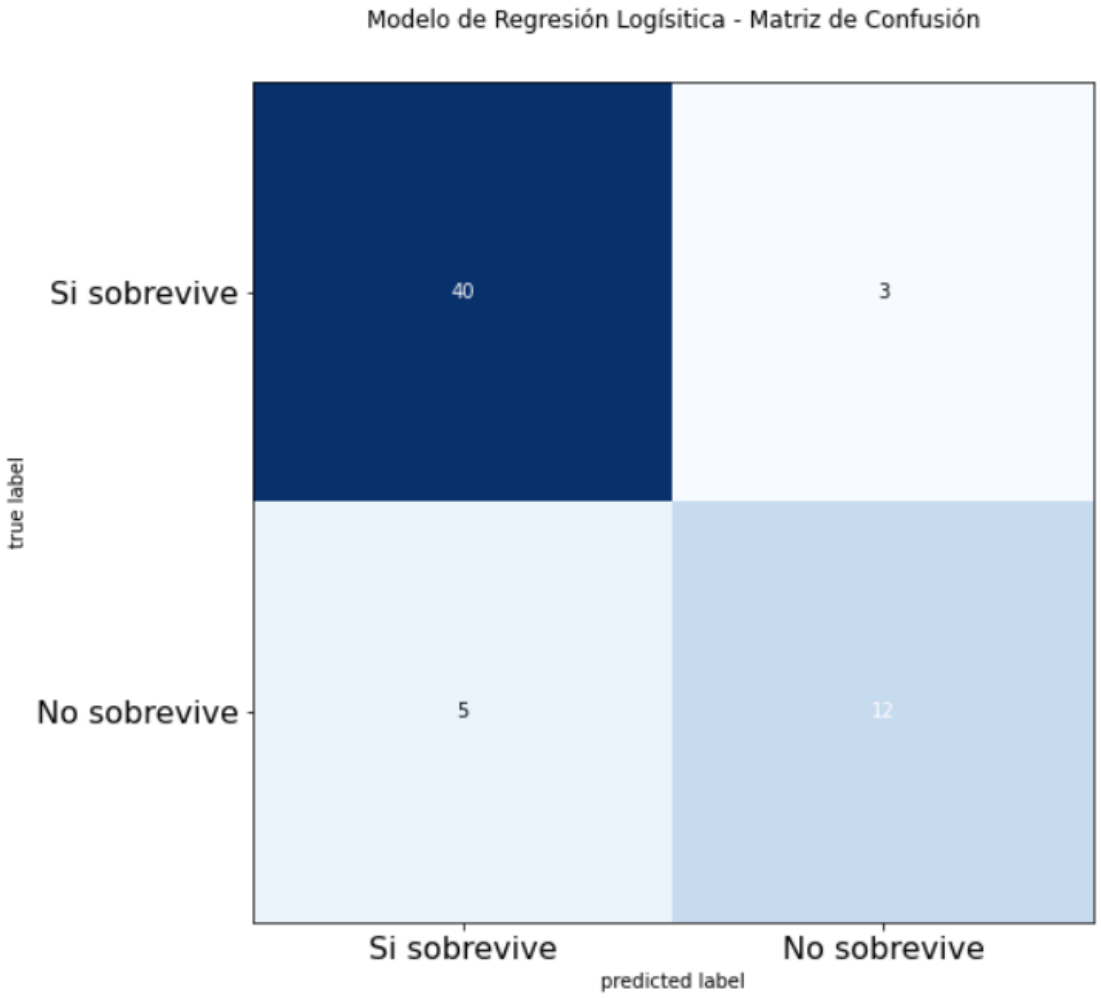
- accuracy_score(y_test, reg_log_pred)

Se importa el modulo para graficar la matriz de confusión. Y se pasa el porcentaje de exactitud del modelo y la predicción.

- cm = confusión_matrix(y_test, reg_log_pred)

```
reg_log = LogisticRegression()
reg_log.fit(x_train,y_train)
reg_log_pred = reg_log.predict(x_test)
reg_log_score = accuracy_score(y_test, reg_log_pred)
resultados.append(100*reg_log_score)
```

```
from mlxtend.plotting import plot_confusion_matrix
cm = confusion_matrix(y_test, reg_log_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Modelo de Regresión Logísitica - Matriz de Confusión")
plt.xticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.yticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.show()
```



Se observa que en la matriz se tiene los siguientes datos

- 40 verdaderos positivos
- 3 falsos positivos
- 5 falsos negativos
- 12 verdaderos negativos

Modelo de KNN

K-Nearest-Neighbor es un algoritmo basado en instancia de tipo supervisado de Machine Learning. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Al ser un método sencillo, es ideal para introducirse en el mundo del Aprendizaje Automático. Sirve esencialmente para clasificar valores buscando los puntos de datos "más similares" (por cercanía) aprendidos en la etapa de entrenamiento (ver 7 pasos para crear tu ML) y haciendo conjeturas de nuevos puntos basado en esa clasificación.

¿Cómo funciona kNN?

Calcular la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento.
Seleccionar los "k" elementos más cercanos (con menor distancia, según la función que se use)
Realizar una "votación de mayoría" entre los k puntos: los de una clase/etiqueta que <<dominen>> decidirán su clasificación final.
En nuestro caso el n_neighbors está de forma predeterminada.

¿Cómo se está utilizando el modelo de KNN?

Se crea una instancia de KNN:

- clas_KNN = KNeighborsClassifier()

Datos de entrenamiento.

- x_train serán los datos de entrada
- y_train los de salida

Se entrena KNN con los datos de entrenamiento

- class_KNN.fit(x_train,y_train)

Se usa el modelo entrenado para obtener las predicciones con datos nuevos

- knn_pred = clas_KNN.predict(x_test)

Se almacena el porcentaje de exactitud

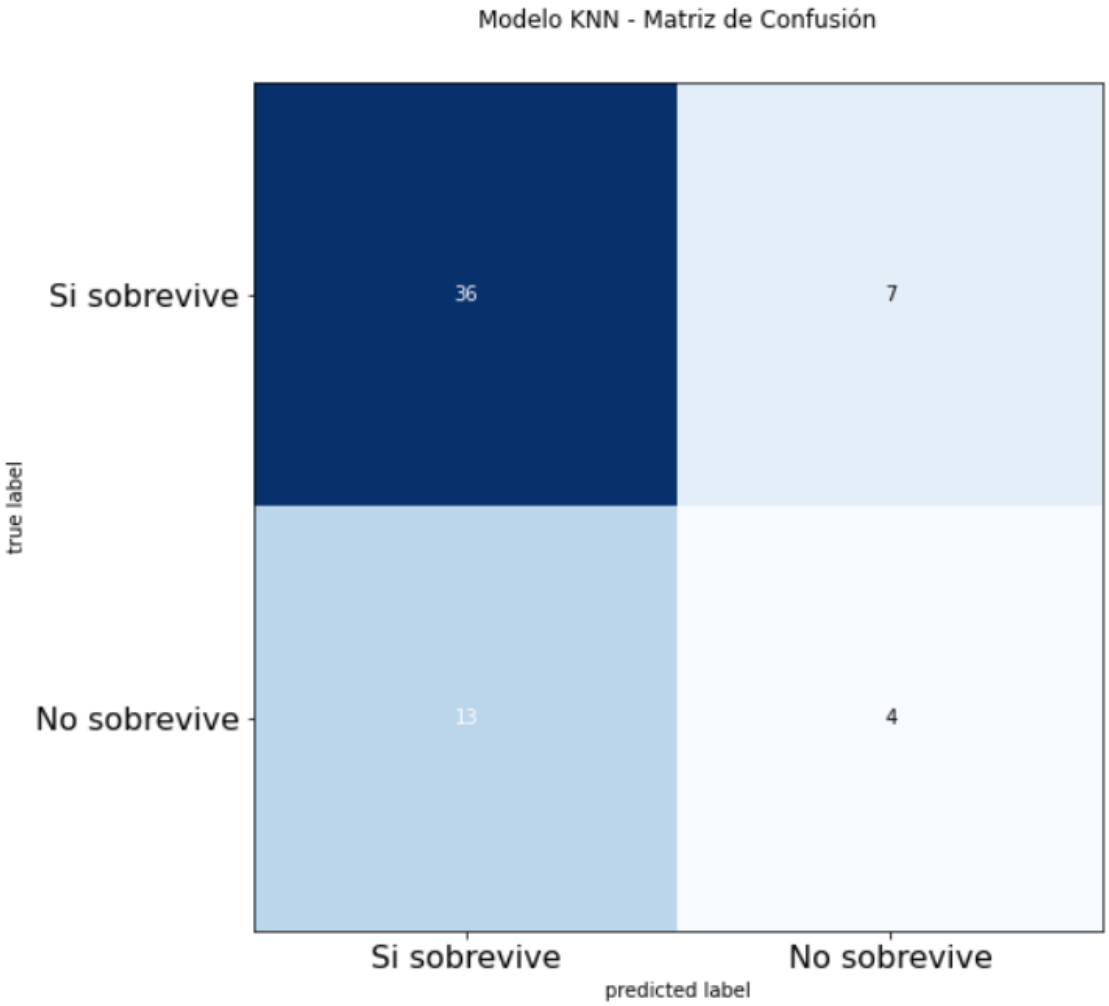
- knn_score = accuracy_score(y_test, knn_pred)

Se importa el modulo para graficar la matriz de confusión. Y se pasa el porcentaje de exactitud del modelo y la predicción.

- cm = confusión_matrix(y_test, knn_pred)

```
clas_KNN = KNeighborsClassifier()
clas_KNN.fit(x_train, y_train)
knn_pred = clas_KNN.predict(x_test)
knn_score = accuracy_score(y_test, knn_pred)
resultados.append(100*knn_score)
```

```
cm = confusion_matrix(y_test, knn_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Modelo KNN - Matriz de Confusión")
plt.xticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.yticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.show()
```



Se observa que en la matriz se tiene los siguientes datos

- 36 verdaderos positivos
- 7 falsos positivos
- 13 falsos negativos
- 4 verdaderos negativos

Modelo de Arboles de Decisión

Son un tipo de algoritmos de aprendizaje supervisado (i.e., existe una variable objetivo predefinida).
Principalmente usados en problemas de clasificación.
Las variables de entrada y salida pueden ser categóricas o continuas.
Divide el espacio de predictores (variables independientes) en regiones distintas y no sobrepuestas.

Parámetros de Árbol de decisión:

- **max_leaf_nodes:** Los mejores nodos se definen como una reducción relativa de la impureza.
- **random_state:** Controla la aleatoriedad del estimador. Las características siempre se permutan aleatoriamente en cada división
- **criterion:** La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y "entropía" para la ganancia de información. En nuestro caso utilizaremos el entropia

¿Cómo se está utilizando el modelo de Arboles de Decisión?
Se crea una instancia de Arboles de Decisión:

- arbol_des = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0, criterion='entropy')

Datos de entrenamiento.

- x_train serán los datos de entrada
- y_train los de salida

Se entrena el Árbol de Decisión con los datos de entrenamiento

- arbol_des.fit(x_train, y_train)

Se usa el modelo entrenado para obtener las predicciones con datos nuevos

- arbol_pred = arbol_des.predict(x_test)

Se almacena el porcentaje de exactitud

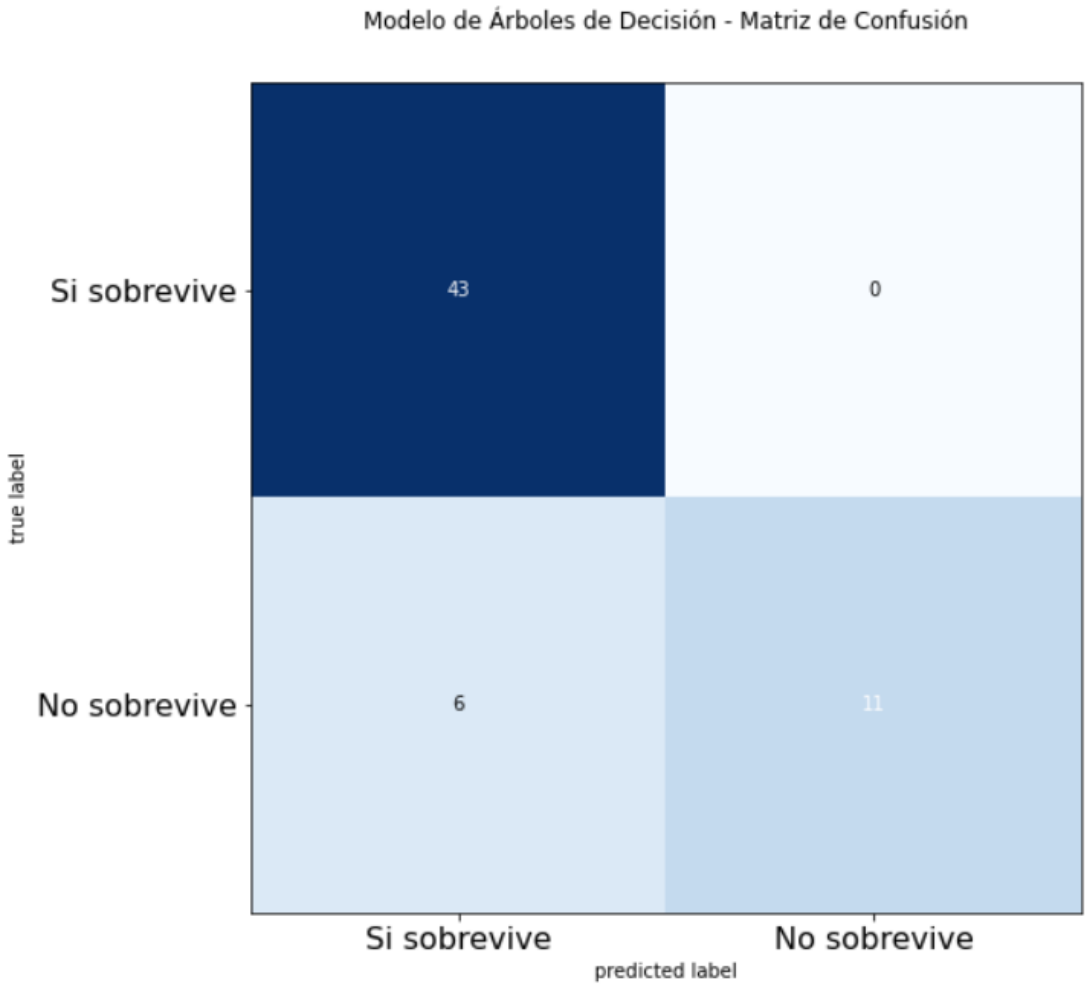
- arbol_score = accuracy_score(y_test, arbol_pred)

Se importa el modulo para graficar la matriz de confusión. Y se pasa el porcentaje de exactitud del modelo y la predicción.

- cm = confusion_matrix(y_test, arbol_pred)

```
arbol_des = DecisionTreeClassifier(max_leaf_nodes=3, random_state=0, criterion='entropy')
arbol_des.fit(x_train, y_train)
arbol_pred = arbol_des.predict(x_test)
arbol_score = accuracy_score(y_test, arbol_pred)
resultados.append(100*arbol_score)
```

```
cm = confusion_matrix(y_test, arbol_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Modelo de Árboles de Decisión - Matriz de Confusión")
plt.xticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.yticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.show()
```



Se observa que en la matriz se tiene los siguientes datos

- 43 verdaderos positivos
- 0 falsos positivos
- 6 falsos negativos
- 11 verdaderos negativos

Soporte de Maquina Vectorial SVM

En el aprendizaje automático, las máquinas de vectores de soporte (SVM, también redes de vectores de soporte) son modelos de aprendizaje supervisado con algoritmos de aprendizaje asociados que analizan los datos utilizados para el análisis de clasificación y regresión.
Las SVM se pueden utilizar para resolver varios problemas del mundo real:

- Las SVM son útiles en la categorización de texto e hipertexto, ya que su aplicación puede reducir significativamente la necesidad de instancias de entrenamiento etiquetadas tanto en la configuración estándar inductiva como transductiva.
- La clasificación de imágenes también se puede realizar utilizando SVM.
- Clasificación de datos satelitales como datos SAR utilizando SVM supervisado.
- Los caracteres escritos a mano se pueden reconocer utilizando SVM.
- El algoritmo SVM se ha aplicado ampliamente en las ciencias biológicas y otras.

¿Cómo se está utilizando el modelo de SVM?

Se crea una instancia de SVC:

- svm_des = SVC()

Datos de entrenamiento.

- x_train serán los datos de entrada
- y_train los de salida

Se entrena el SVM con los datos de entrenamiento

- svm_des.fit(x_train, y_train)

Se usa el modelo entrenado para obtener las predicciones con datos nuevos

- svm_pred = svm_des.predict(x_test)

Se almacena el porcentaje de exactitud

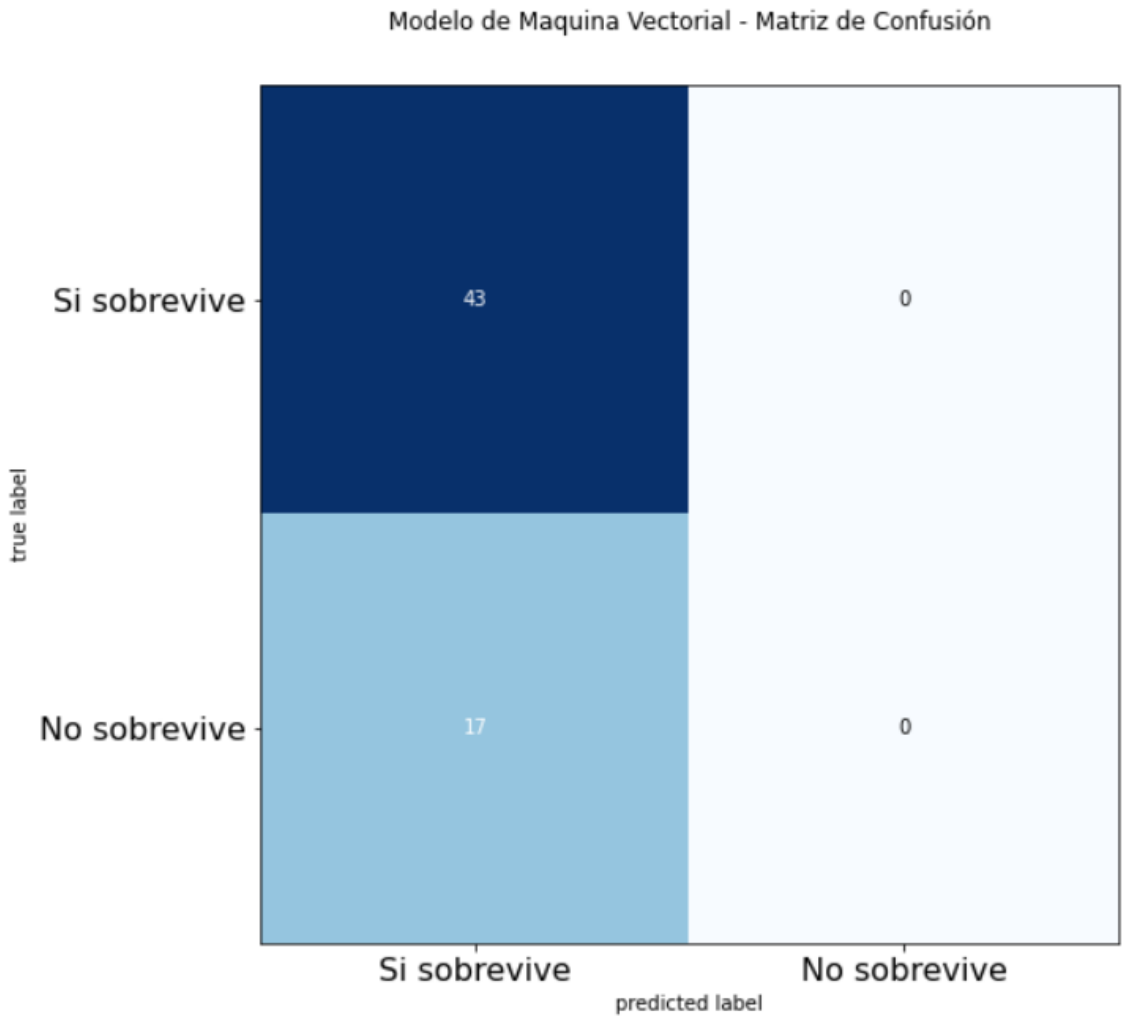
- svm_score = accuracy_score(y_test, svm_pred)

Se importa el modulo para graficar la matriz de confusión. Y se pasa el porcentaje de exactitud del modelo y la predicción.

- cm = confusion_matrix(y_test, svm_pred)

```
svm_des = SVC()
svm_des.fit(x_train, y_train)
svm_pred = svm_des.predict(x_test)
svm_score = accuracy_score(y_test, svm_pred)
resultados.append(100* svm_score)
```

```
cm = confusion_matrix(y_test, svm_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Modelo de Maquina Vectorial - Matriz de Confusión")
plt.xticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.yticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.show()
```



Se observa que en la matriz se tiene los siguientes datos

- 43 verdaderos positivos
- 0 falsos positivos
- 17 falsos negativos
- 0 verdaderos negativos

Random Forest

Un modelo Random Forest está formado por un conjunto de árboles de decisión individuales, cada uno entrenado con una muestra ligeramente distinta de los datos de entrenamiento generada mediante bootstrapping). La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo.

Parámetros de Random Forest:

- **max_features:** La cantidad de características que se deben considerar al buscar la mejor división:
  - Si es int, considere las max_featurescaracterísticas en cada división.
  - Si es flotante, entonces max_featureses una fracción y las características se consideran en cada división.int (max_features * n_features)
  - Si es "automático", entonces max_features=sqrt(n_features).
  - Si es "sqrt", entonces max_features=sqrt(n_features)(igual que "auto").
  - Si es "log2", entonces max_features=log2(n_features).
  - Si es Ninguno, entonces max_features=n_features.

- **max_depth:** La profundidad máxima del árbol. Si es None, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos de min_samples_split muestras.
- **random_state:** Controla tanto la aleatoriedad

¿Cómo se está utilizando el modelo Random Forest?

Se crea una instancia de Random Forest:
- randForest = RandomForestClassifier(max_features=0.5, max_depth=15, random_state=1)

Datos de entrenamiento.
- x_train serán los datos de entrada
- y_train los de salida

Se entrena el Random Forest con los datos de entrenamiento
- randForest.fit(x_train, y_train)

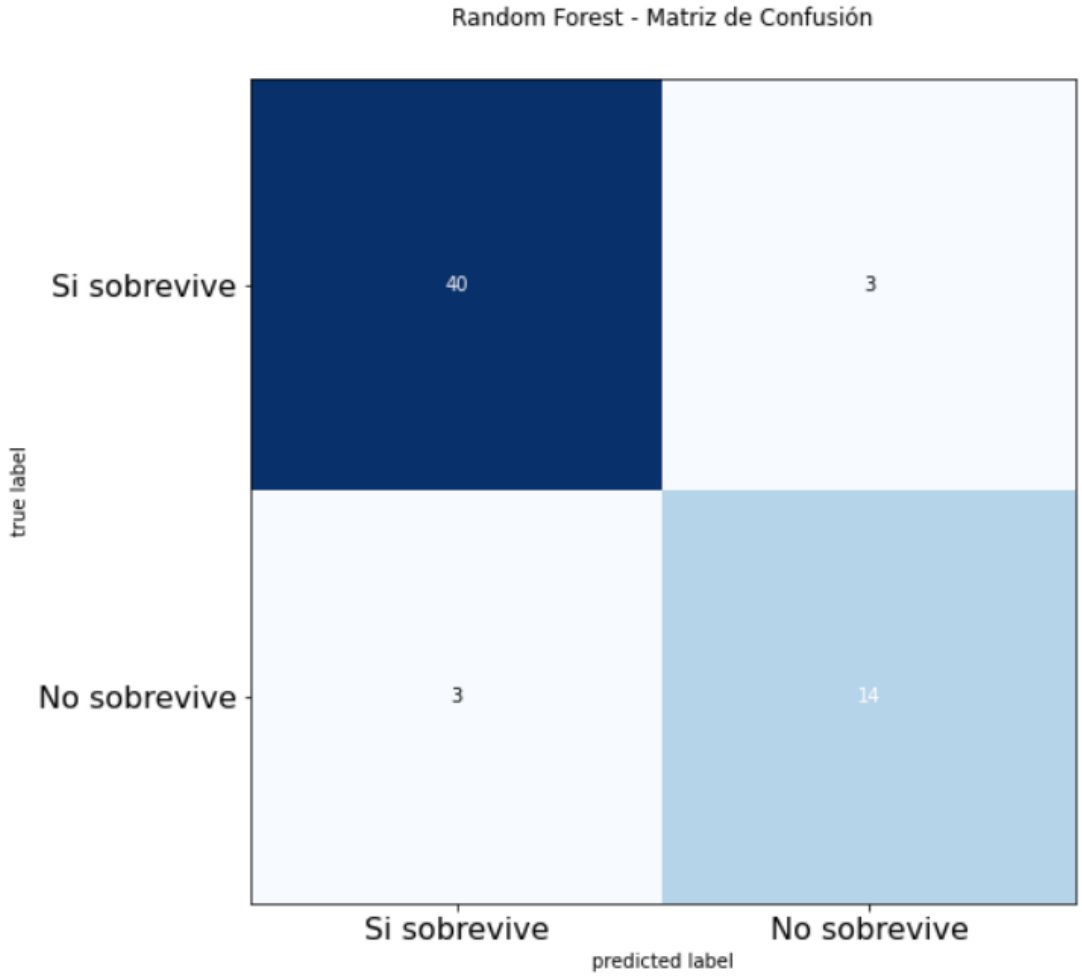
Se usa el modelo entrenado para obtener las predicciones con datos nuevos
- randForest_pred = randForest.predict(x_test)

Se almacena el porcentaje de exactitud
- randForest_score = accuracy_score(y_test, randForest_pred)

Se importa el modulo para graficar la matriz de confusión. Y se pasa el porcentaje de exactitud del modelo y la predicción.
- cm = confusion_matrix(y_test, randForest_pred)

```
randForest = RandomForestClassifier(max_features=0.5, max_depth=15, random_state=1)
randForest.fit(x_train, y_train)
randForest_pred = randForest.predict(x_test)
randForest_score = accuracy_score(y_test, randForest_pred)
resultados.append(100*randForest_score)
```

```
cm = confusion_matrix(y_test, randForest_pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Random Forest - Matriz de Confusión")
plt.xticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.yticks(range(2), ["Si sobrevive","No sobrevive"], fontsize=16)
plt.show()
```



Se observa que en la matriz se tiene los siguientes datos

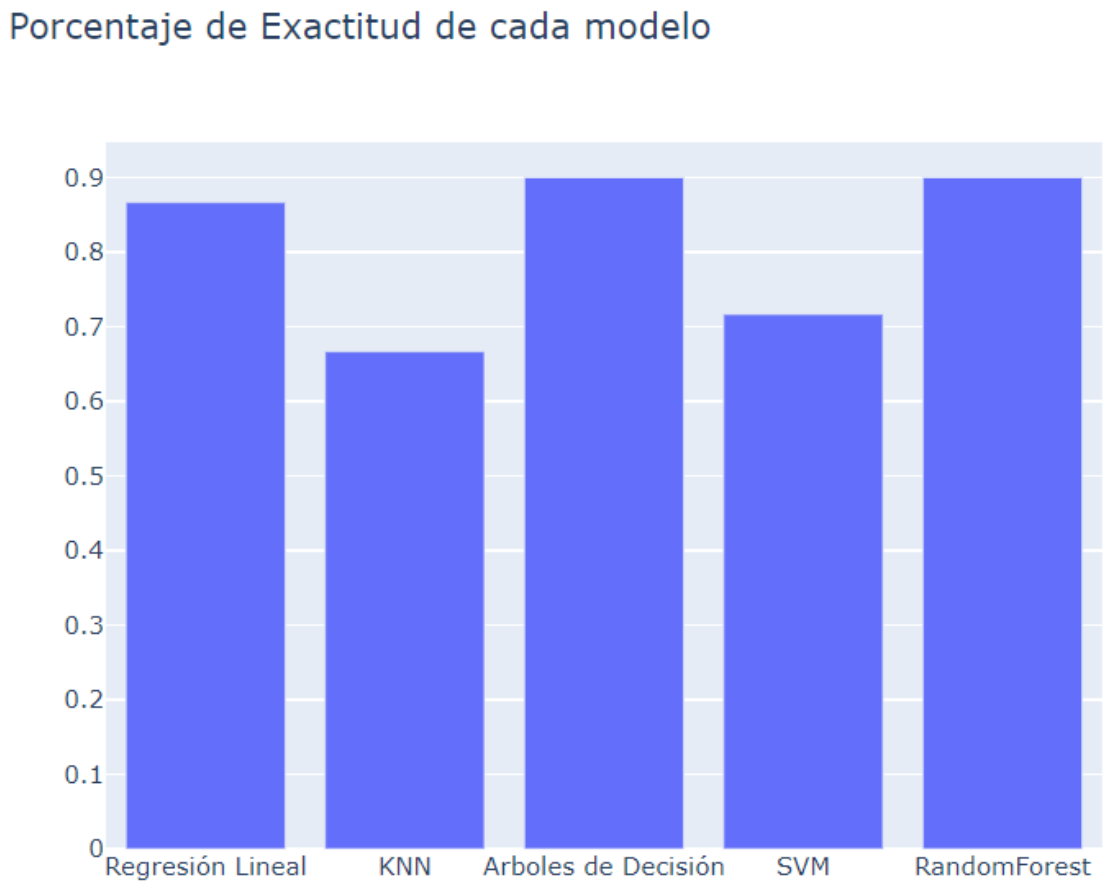
- 40 verdaderos positivos
- 3 falsos positivos
- 3 falsos negativos
- 14 verdaderos negativos

Evaluación de exactitud de los modelos

Se realiza un gráfico de barras y se observa que modelo tiene más precisión según su porcentaje.

```
import plotly.graph_objects as go
modelos= ['Regresión Lineal', 'KNN','Arboles de Decisión', 'SVM', 'RandomForest']

fig = go.Figure([go.Bar(x=modelos, y=[reg_log_score,knn_score,arbol_score,svm_score
fig.update_layout(title_text="Porcentaje de Exactitud de cada modelo")
fig.show()
```



- **Regresión Lineal:** 87% de precisión
- **KNN:** 67% de precisión
- **Arboles de Decisión:** 90% de precisión
- **SVM:** 72% de precisión
- **Random Forest:** 90% de precisión

Se puede observar que el modelo de Arboles de Decisión y el modelo Random Forest tienen el mismo porcentaje de precisión pero gracias a la matriz de confusión de cada uno podemos identificar cual es el mejor
En la matriz de confusión del modelo de Arboles de decisión se tiene 0 falsos positivos mientras que en la matriz de confusión de Random Forest se tiene 3 falsos positivos, entonces podemos determinar que el mejor modelo es el de Arboles de decisión. Ya que tener falsos positivos nos trae peligros.

Almacenamiento del mejor modelo

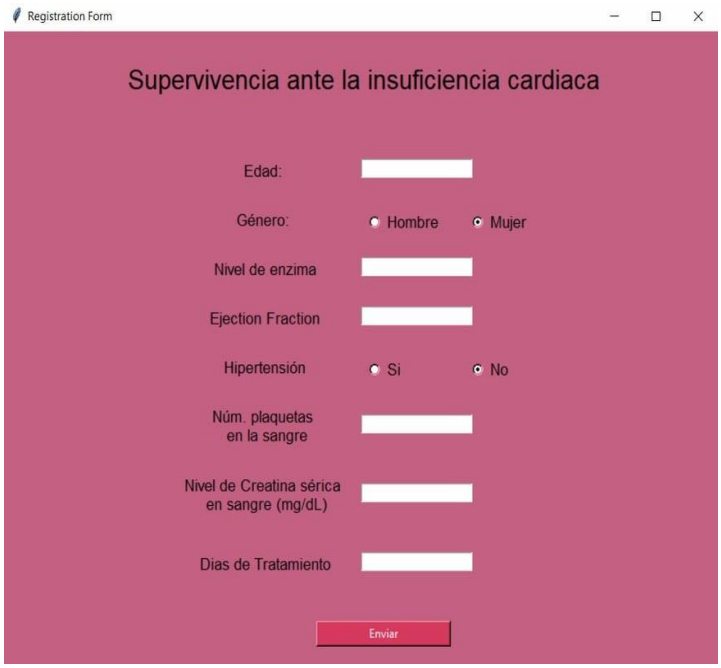
Ya que se ha obtenido el mejor modelo de predicción para nuestro caso se lo almacena, para utilizarlo en una interfaz gráfica.

```
import pickle
# save the model to disk
filename = 'modeloIC.sav'    .
pickle.dump(arbol_des, open(filename, 'wb'))
```

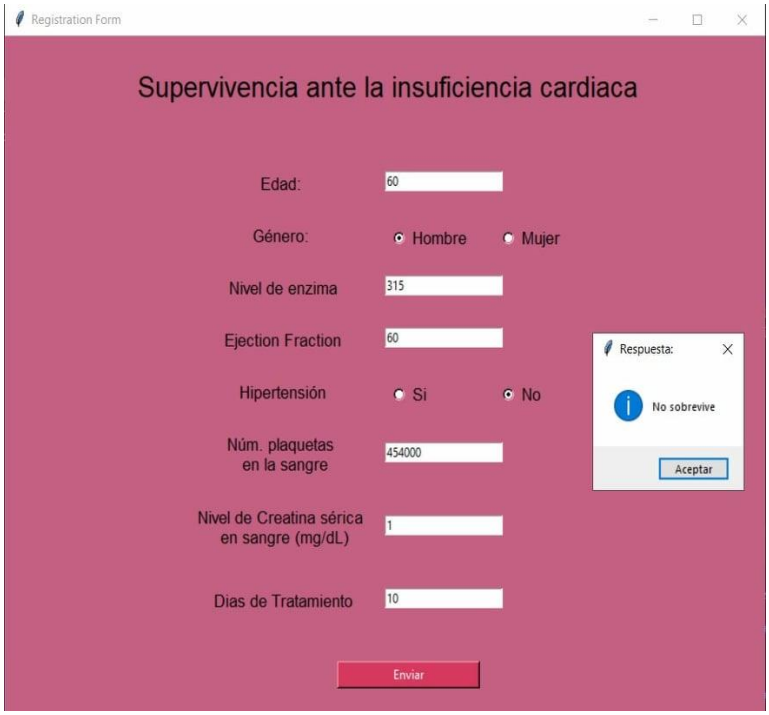
Interfaz Grafica

Se cuenta con una interfaz gráfica para ingresar los datos de prueba y este nos indique el evento de muerte.

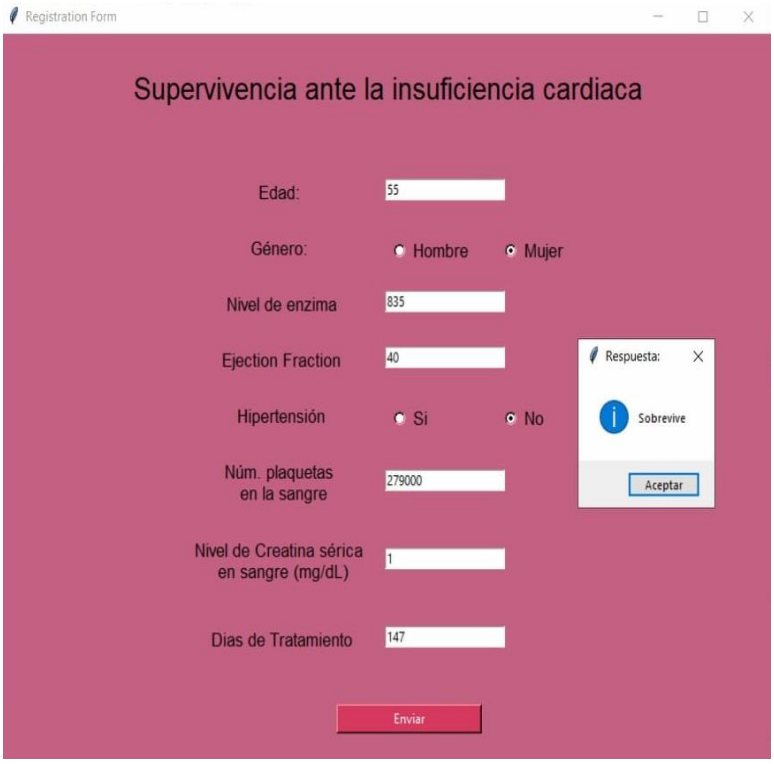
Interfaz



La interfaz con los datos ya mencionados



Ejemplo: Datos ingresados de un paciente que no sobrevivió a la IC



Ejemplo: Datos ingresados de un paciente que si sobrevivió a la IC

Conclusión

Como conclusión se obtiene que gracias a los algoritmos de Machine Learning se puede llegar a predecir si una persona con insuficiencia cardiaca puede o no sobrevivir. Y esto puede se puede aplicar en el mundo real lo cual sería de gran ayuda para hospitales que traten estas enfermedades.
Utilizar Machine Learning de manera adecuada puede llegar a ser muy beneficiosa para diferentes áreas del mundo real.

Referencias

https://www.pennmedicine.org/updates/blogs/heart-and-vascular-blog/2014/october/ejection-fraction-what-the-numbers-mean

http://www.wikicardio.org.ar/wiki/Hipertensi%C3%B3n?gclid=CjwKCAiA_Kz-BRAJEiwAhJNY70_vyMwESso0A5MXH31G2Tm6sUpRg1aawM7i17YxAk9qm146IgdzxRoCr8cQAvD_BwE

https://www.niddk.nih.gov/health-information/informacion-de-la-salud/enfermedades-rinones/presion-arterial-insuficiencia-renal

https://www.goredforwomen.org/es/health-topics/heart-failure/diagnosing-heart-failure/ejection-fraction-heart-failure-measurement

https://www.hcmarbella.com/es/hipertension-que-es-y-como-nos-afecta/

http://www.wikicardio.org.ar/wiki/Que_es_la_talasemia?gclid=CjwKCAiA_Kz-BRAJEiwAhJNY79a0xNDzUGG3K_oqMDPcP9L1o_fUiS_q6i5iQT1B9exbyziKdz2ZFhoCPlsQAvD_BwE

https://www.niddk.nih.gov/health-information/informacion-de-la-salud/enfermedades-rinones/presion-arterial-insuficiencia-renal

https://www.interactivechaos.com/python/function/traintestsplit

https://aprendeia.com/libreria-scikit-learn-de-python/

https://www.iartificial.net/como-usar-regresion-logistica-en-python/

https://empresas.blogthinkbig.com/como-interpretar-la-matriz-de-confusion-ejemplo-practico/

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html

https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://en.wikipedia.org/wiki/Support_vector_machine#Applications

https://www.cienciadedatos.net/documentos/py08_random_forest_python.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://medlineplus.gov/spanish/ency/article/000158.htm#:~:text=La%20insuficiencia%20card%C3%ADaca%20es%20una,s%C3%ADntomas%20en%20todo%20el%20cuerpo.

https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-automatico-machine-learning
https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/#:~:text=Un%20%C3%A1rbol%20de%20decisi%C3%B3n%20en,nodo%20hoja%20representa%20el%20resultado.&text=Esta%20estructura%20tipo%20diagrama%20de%20flujo%20lo%20ayuda%20a%20tomar%20decisiones