Tarea 1: Visualizació de la Información

Fragoso Alvarado Daniel | 316049054

De acuerdo con la Organización Mundial de la Salud (OMS) las enfermedades cardiovasculares son la principal causa de defunción en el mundo y, según estimaciones, se cobran 17,9 millones de vidas cada año. Estas enfermedades constituyen un grupo de trastornos del corazón y los vasos sanguíneos que incluyen cardiopatías coronarias, enfermedades cerebrovasculares y cardiopatías reumáticas. Más de cuatro de cada cinco defunciones por enfermedades cardiovasculares se deben a cardiopatías coronarias y accidentes cerebrovasculares, y una tercera parte de esas defunciones ocurren prematuramente en personas menores de 70 años.

In [ ]:
```
# importemos las bibliotecas

import pandas as pd
import cufflinks as cf
from IPython.display import display,HTML
```

In [ ]:
```
# Configuremos el tema con el que vamos a trabajar

cf.getThemes()
```

Out[ ]:
```
['ggplot', 'pearl', 'solar', 'space', 'white', 'polar', 'henanigans']
```

In [ ]:
```
# Seleccionemos el tema

cf.set_config_file(sharing='public', theme='white', offline=True)
```

In [ ]:
```
# Vamos a importar por medio de Pandas el conjunto de datos

heart = pd.read_csv('heart.csv')

# Visualicemos
heart.head(5)
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

En el archivo heart.csv se presenta un conjunto de datos reales que incluye las características importantes de pacientes relacionados a enfermedades cardiovasculares. Hay trece características y un objetivo como se muestra a continuación:

- age: la edad de la persona en años
- sex: el sexo de la persona ($1 =$ masculino, $0 =$ femenino)
- cp: El dolor toráxico experimentado (Valor 1: angina típica, Valor 2: angina atípica, Valor 3: dolor no anginoso, Valor 4: asintomático)
- trestbps: la presión arterial en reposo de la persona (mm Hg al ingreso en el hospital)
- chol: la medida de colesterol de la persona en mg/dl
- fbs: azúcar en sangre en ayunas de la persona ($> 120\text{mg}/\text{dl}, 1 =$ verdadero; $0 =$ falso)
- resecg: medición electrocardiográfica en reposo ($0 =$ normal, $1 =$ con anormalidad de onda ST-T, $2 =$ mostrando hipertrofia ventricular izquierda probable o definitiva según los criterios de Estes)
- thalach: la frecuencia cardíaca máxima alcanzada por la persona
- exang: angina inducida por el ejercicio ($1 =$ sí; $0 =$ no)
- old peak: depresión del ST inducida por el ejercicio en relación con el reposo
- slope: la pendiente del segmento ST del ejercicio máximo (Valor 1: pendiente ascendente, Valor 2: plano, Valor 3: pendiente descendente)
- ca: El número de vasos principales (0-3)
- thal: un trastorno de la sangre llamado Talasemia ($3 =$ normal; $6 =$ defecto fijo; $7 =$ defecto reversible)
- target: enfermedad cardíaca ($0 =$ no, $1 =$ sí)

Utilizaremos el archivo `heart.csv` para determinar el porcentaje de personas que han sido diagnosticados con una enfermedad cardíaca mediante la columna `target`

In [ ]:
```
# Primero vamos a eliminar los elemntos faltantes

heart = heart.dropna()
```

In [ ]:
```
# Ahora vamos a obtener un subcojunto que solo posea la columna target y un contador:
heart['cont'] = 1

target = heart[['target', 'cont']]

target.head(5)
```

Out[ ]:

	target	cont
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1

In [ ]:
```
# Y ahora hagamos un conteo mediante la función groupby y el método count

conteo = target.groupby('target').count()

conteo
```

Out[ ]:

	cont
target	
0	138
1	165

In [ ]:
```
# Determinar el procentaje de peronas con enfermedad cardiaca

total = conteo.cont[0] + conteo.cont[1]
car = conteo.cont[1] / total * 100

print(f'De un total de {total} pacientes, se le han diagnosticado enfermedades cárdiacas a un {car:.2f}%')
```

```
De un total de 303 pacientes, se le han diagnosticado enfermedades cárdiacas a un 54.46%
```

Elabore un histograma o grafica de barras que permita visualizar la edad (age) comparada con el porcentaje de personas si diagnosticadas con una enfermedad cardiaca y las que no han sido diagnosticadas (target, donde $0 =$ no, $1 =$ sí).

In [ ]:
```
# Hay dos formas de hacerlo, primero contabilizaremos mediante funciones con el DataFrame:

# Vamos a separar las edades en 10: 0-10, 11-20, 21-30, 31-40, 41-50 ...
# Y vamos a contar la cantidad de peronas, y desde luego las peronas que tienen diagnosticada
# una enfermedad cardiovascular
edades = heart[['age', 'target']]

edad_20 = edades.loc[(edades['age'] <= 29)  & (edades['age'] >= 20)]
edad_30 = edades.loc[(edades['age'] <= 39)  & (edades['age'] >= 30)]
edad_40 = edades.loc[(edades['age'] <= 49)  & (edades['age'] >= 40)]
edad_50 = edades.loc[(edades['age'] <= 59)  & (edades['age'] >= 50)]
edad_60 = edades.loc[(edades['age'] <= 69)  & (edades['age'] >= 60)]
edad_70 = edades.loc[(edades['age'] <= 79)  & (edades['age'] >= 70)]
edad_80 = edades.loc[(edades['age'] <= 89)  & (edades['age'] >= 80)]
```

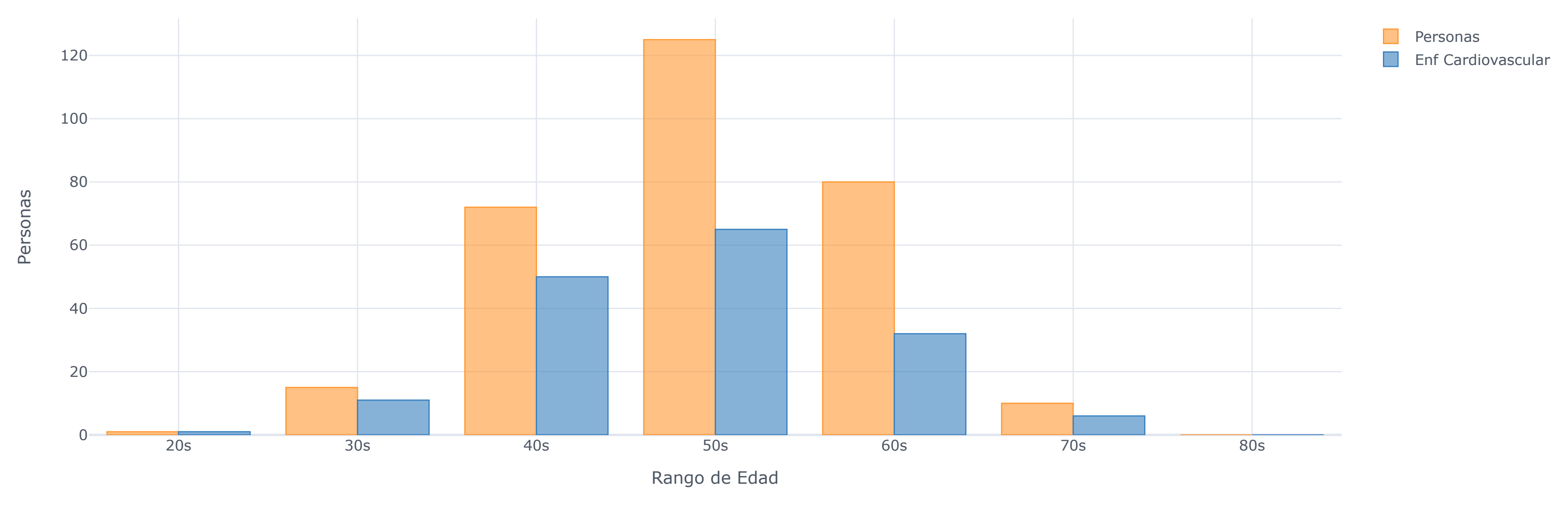
In [ ]:
```
# Vamos a hacer un nuevo Data Frame:

cont = pd.DataFrame()
# creamos las columnas
cont['Personas'] = None
cont['Enf Cardiovascular'] = None

# añadimos filas por su nombre de fila
cont.loc['20s'] = [edad_20.age.count(), edad_20.target.sum()]
cont.loc['30s'] = [edad_30.age.count(), edad_30.target.sum()]
cont.loc['40s'] = [edad_40.age.count(), edad_40.target.sum()]
cont.loc['50s'] = [edad_50.age.count(), edad_50.target.sum()]
cont.loc['60s'] = [edad_60.age.count(), edad_60.target.sum()]
cont.loc['70s'] = [edad_70.age.count(), edad_70.target.sum()]
cont.loc['80s'] = [edad_80.age.count(), edad_80.target.sum()]
```

In [ ]:
```
cont
```

Out[ ]:

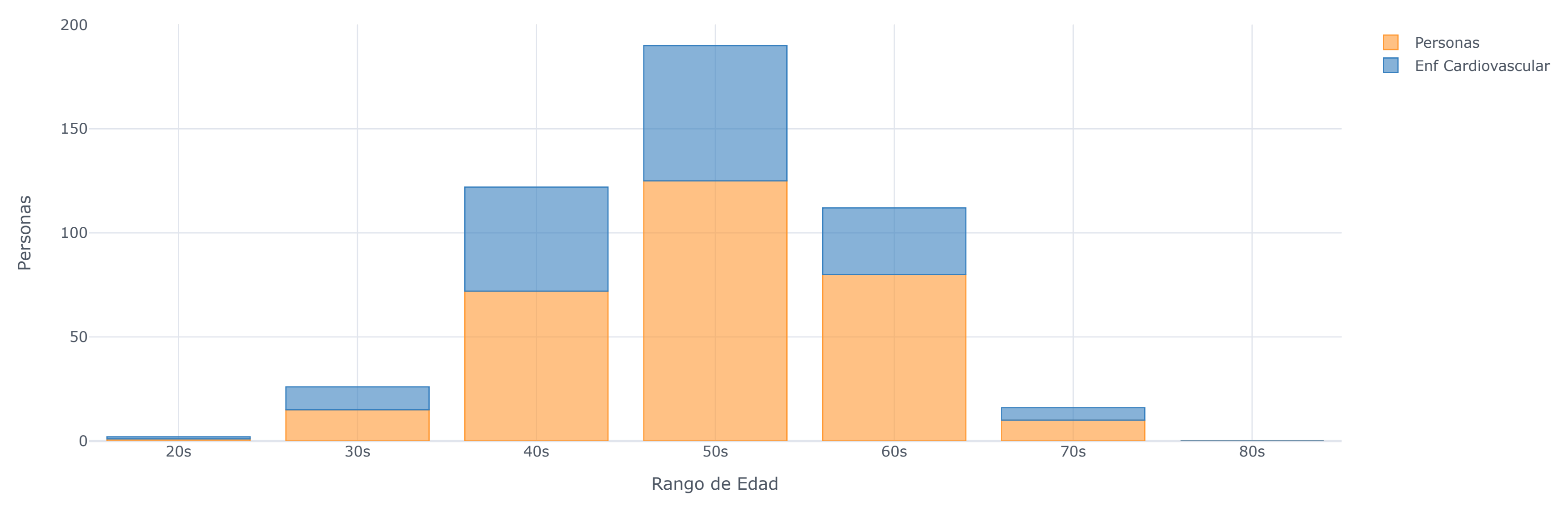
	Personas	Enf Cardiovascular
20s	1	1
30s	15	11
40s	72	50
50s	125	65
60s	80	32
70s	10	6
80s	0	0

In [ ]:
```
# plotting
cont.iplot(kind='bar', xTitle='Rango de Edad',
                       yTitle='Personas')
```



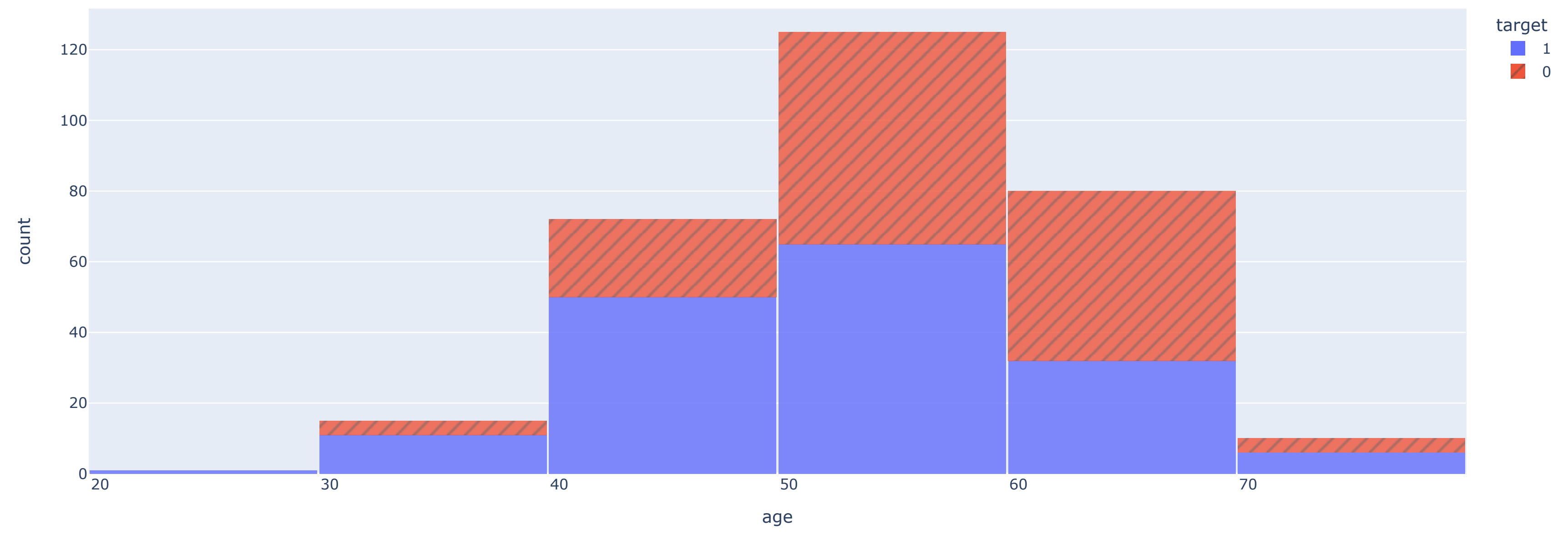
Export to plot.ly »

In [ ]:
```
# Podemos tambien mantenerlos sobre la misma columna:

cont.iplot(kind='bar', xTitle='Rango de Edad',
                       yTitle='Personas', barmode = 'stack')
```



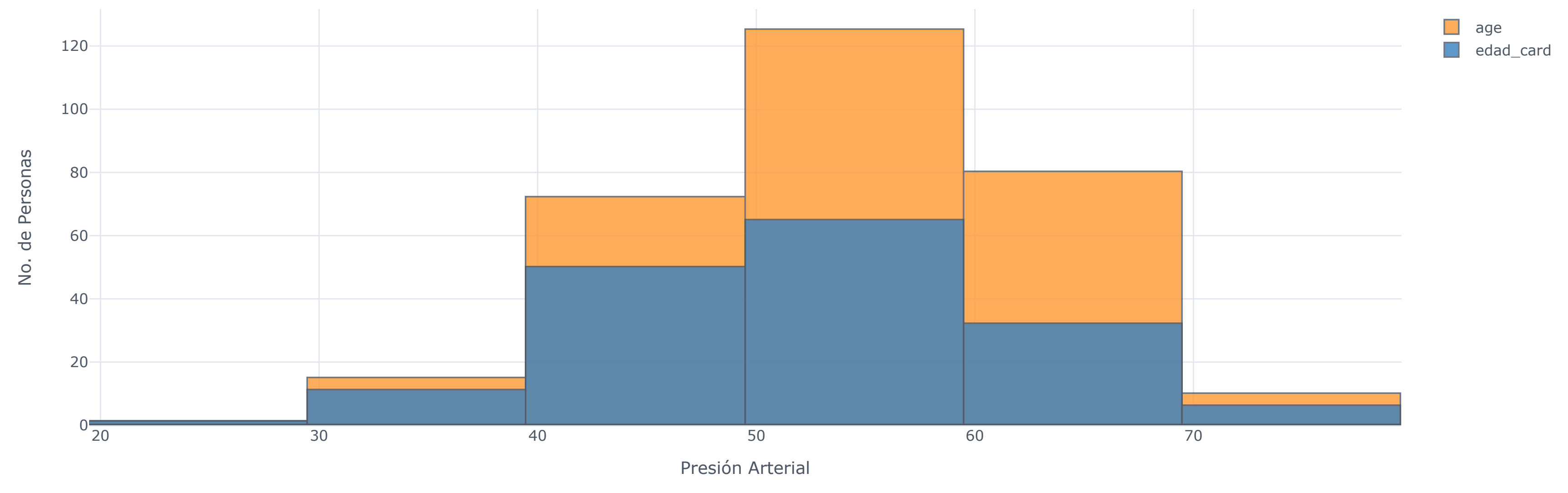
Export to plot.ly »

In [ ]:
```
# Otra manera (más sencilla) de hacerlo es utilizando plotly mendiate un histograma:

import plotly.express as px

edades = heart[['age', 'target']]
#df = px.edades.tips()
fig = px.histogram(edades, x="age", color="target", pattern_shape="target", nbins = 6,
                opacity=0.8)
fig.update_layout(bargap=0.01)
fig.show()
```



Hay que mencionar que este método de contar no es muy óptimo por lo tardado que es, así que lo que una forma más útil de hacerlo es sobre poniendo histogramas a partir de la división del conjunto de datos en las cátegorias que deseamos:

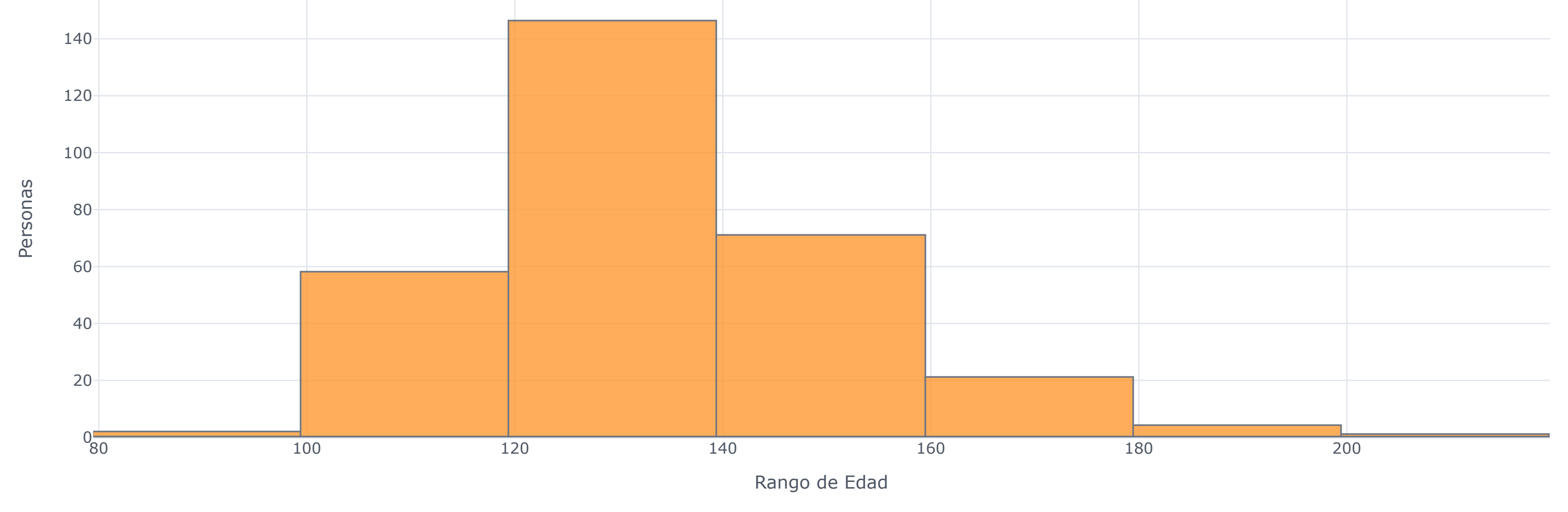
In [ ]:
```
# Agregamos al conjunto de datos una nueva columna, con solo aquellas personas con efermedad cardiaca

heart['edad_card'] = heart[heart['target'] == 1]['age'] # Las edades de quienes tienen una enf. Cardiovascular

heart[["age","edad_card"]].iplot(kind="histogram", bins=6,
         xTitle='Presión Arterial', yTitle='No. de Personas')
```



Export to plot.ly »

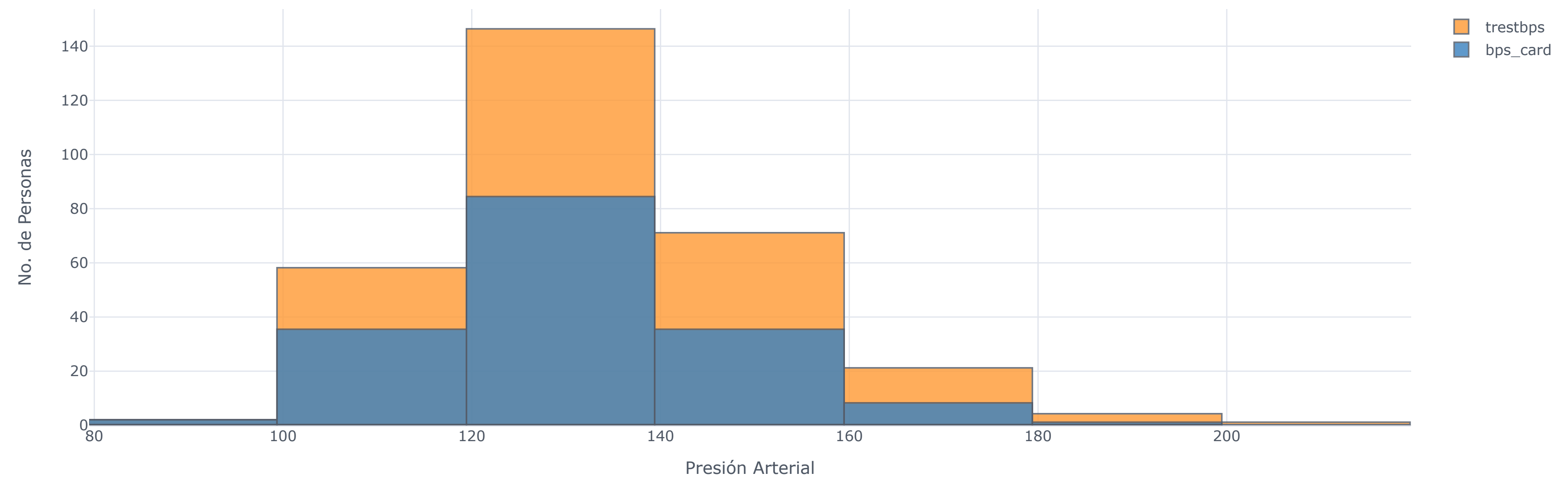
Ahora elaboraremos un histograma que permita visualizar la presión arterial (`trestbps`) comparada con el porcentaje de personas si diagnosticadas con una enfermedad cardiaca y las que no han sido diagnosticadas.

In [ ]:
```
# El histograma de la distribución de la presión
heart['trestbps'].iplot(kind='histogram', xTitle='Rango de Edad',
                        yTitle='Personas', bins = 6)
```



Export to plot.ly »

In [ ]:
```
# Vamos a repetir el proceso para obtener los conteos, sin embargo ahora dado que queremos hacerlo
# a partir de histogramas, solo vamos a dividir el conjunto de datos en dos:

# Agregamos al conjunto de datos una nueva columna, con solo aquellas personas con efermedad cardiaca

heart['bps_card'] = heart[heart['target'] == 1]['trestbps']

heart[["trestbps","bps_card"]].iplot(kind="histogram", bins=6,
         xTitle='Presión Arterial', yTitle='No. de Personas')
```



Export to plot.ly »

In [ ]:
```
# Otra manera (más sencilla) de hacerlo es utilizando plotly mendiate un histograma:

pcard = heart[['trestbps', 'target']]
#df = px.edades.tips()
fig = px.histogram(pcard, x="trestbps", color="target", pattern_shape="target", nbins = 6,
                opacity=0.8)
fig.update_layout(bargap=0.01)
fig.show()
```