Aprendizaje Supervisado

Abraham Zamudio

VI Programa de Especialización en Machine Learning con Python

2020

CTIC-UNI
BUSINESS SCHOOL

En el ejemplo que vamos a utilizar, vamos a imaginarnos que una organización sin fines de lucro soporta su operación mediante la organización periódica de una campaña para recaudar fondos por correo. Esta organización ha creado una base de datos con más de 40 mil personas que por lo menos una vez en el pasado ha sido donante. La campaña de recaudación de fondos se realiza mediante el envío a una lista de correo (o un subconjunto de ella) de un regalo simbólico y la solicitud de una donación. Una vez que se planifica la campaña, el costo total de la misma se conoce de forma automática:

[número de potenciales donantes a contactar] x ([costo de regalo] + [costo de correo])

Sin embargo, el resultado de la recaudación de fondos depende tanto del número de donantes que responde a la campaña, como del importe medio de dinero que es donado.

La idea es que, utilizando las técnicas de Machine Learning sobre la base de datos de esta organización, podamos ayudarla a maximizar los beneficios de la campaña de recaudación, esto es, lograr el máximo importe posible de dinero recaudado, minimizando lo más que se pueda el costo total de la campaña. Debemos tener en cuenta que un miembro de la organización le enviará el correo a un potencial donante, siempre que el rendimiento esperado del pedido excede el costo del correo con la solicitud de donación.

Para nuestro ejemplo, el costo por donante de la campaña va a ser igual al **[costo de regalo] + [costo de correo]**, y esto va a ser igual a $ 0.75 por correo enviado. Los ingresos netos de la campaña se calculan como la suma (**importe de donación real - $ 0.75**) sobre todos los donantes a los que se ha enviado el correo. Nuestro objetivo es ayudar a esta organización sin fines de lucro a seleccionar de su lista de correo los donantes a los que debe abordar a los efectos de maximizar los beneficios de la campaña de recaudación.

Los Datos

El dataset que vamos a utilizar, consiste en la base de datos de la organización sin fines de lucro con la lista de correo de los donantes de sus campañas anteriores.

▸ Data1marzo

El contenido debe ser :

1. `Entrenamiento.csv`
2. `Validacion.csv`
3. `DescripcionData.txt`

Los Datos

Algunos otros datos a tener en cuenta, son los siguientes:

1. El dataset de aprendizaje contiene 47720 registros y 481 columnas. La primera fila / cabecera del mismo contiene los nombres de cada campo.
2. El dataset de validación contiene 47692 registros y 479 columnas. Al igual que en el caso anterior, la primera fila contiene los nombres de cada campo.
3. Los registros del dataset de validación son idénticos a los registros del dataset de aprendizaje, excepto que los valores para nuestros campos objetivo que necesitamos para el aprendizaje, no existen(es decir, las columnas `DONOR_FLAG` y `DONOR_AMOUNT` no están incluidas en el dataset de validación).

Los Datos

4. Los espacios en blanco en los campos de tipo texto y los puntos en los campos de tipo numérico corresponden a valores faltantes o perdidos.
5. Cada registro tiene un identificador único de registro o índice (campo `IDX`). Para cada registro, hay dos variables objetivo (campos `DONOR_FLAG` y `DONOR_AMOUNT`). `DONOR_FLAG` es una variable binaria que indica si ese registro fue donante o no; mientras que `DONOR_AMOUNT` contiene el importe de la donación para los casos que fueron donantes.

Los Datos

6. Algunos de los valores en el dataset pueden contener errores de formato o de ingreso. Por lo que se deberían corregir o limpiar.
7. Una descripción detallada del significado de cada columna del dataset, la pueden encontrar en el archivo `DescripcionData.txt`.

El primer paso que deberíamos emprender, es realizar un pequeño análisis exploratorio de nuestro dataset; es decir, valernos de algunos herramientas de la estadística, junto con algunas visualizaciones para entender un poco más los datos de los que disponemos.

```
1  # Importando las librerias que vamos a utilizar
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  from sklearn.preprocessing import LabelEncoder
7
8  # graficos incrustados
9  %matplotlib inline
10
11 # parametros esteticos de seaborn
12 sns.set_palette("deep", desat=.6)
13 sns.set_context(rc={"figure.figsize": (8, 4)})
```

Preprocesamiento de Datos en Python

Preprocesamiento y exploración

```
1  # importando el dataset a un Dataframe de Pandas
2  ONG_data = pd.read_csv('data/Entrenamiento.csv', header=0)
3
4  # Examinando las primeras 5 filas y 5 columnas del dataset
5  ONG_data.iloc[:5, :5]
6
7  # Contamos  la cantidad de registros
8  ONG_data['DONOR_AMOUNT'].count()
```

Comenzemos a explorar un poco más en detalle este dataset!.En primer lugar, lo que deberíamos hacer es controlar si existen valores faltantes o nulos; esto lo podemos realizar utilizando el método `isnull()` del siguiente modo:

▸ pandas.isnull

```
# Controlando valores nulos
ONG_data.isnull().any().any()
```

Como podemos ver, el método nos devuelve el valor "True", lo que indica que existen valores nulos en nuestro dataset. Estos valores pueden tener una influencia significativa en nuestro modelo predictivo, por lo que siempre es una decisión importante determinar la forma en que los vamos a manejar.

Las alternativas que tenemos son:

1. Dejarlos como están, lo que a la larga nos va a traer bastantes dolores de cabeza ya que en general los algoritmos no los suelen procesar correctamente y provocan errores.
2. Eliminarlos, lo que es una alternativa viable aunque, dependiendo la cantidad de valores nulos, puede afectar significativamente el resultado final de nuestro modelo predictivo.
3. Inferir su valor. En este caso, lo que podemos hacer es tratar de inferir el valor faltante y reemplazarlo por el valor inferido. Esta suele ser generalmente la mejor alternativa a seguir.

Vamos a inferir los valores faltantes utilizando la media aritmética para los datos cuantitativos y la moda para los datos categóricos. Como vamos a utilizar dos métodos distintos para reemplazar a los valores faltantes, dependiendo de si son numéricos o categóricos, el primer paso que debemos realizar es tratar de identificar que columnas de nuestro dataset corresponde a cada tipo de datos; para realizar esto vamos a utilizar el atributo `dtypes` del Dataframe de Pandas.

```
1  # Agrupando columnas por tipo de datos
2  tipos = ONG_data.columns.to_series()
3  tipos = tipos.groupby(ONG_data.dtypes).groups
4
5  # Armando lista de columnas categoricas
6  ctext = tipos[np.dtype('object')]
7  len(ctext)  # cantidad de columnas con datos categoricos.
```

```
# Armando lista de columnas numericas
columnas = ONG_data.columns  # lista de todas las columnas
cnum = list(set(columnas) - set(ctext))
len(cnum)
```

Ahora ya logramos separar a las 481 columnas que tiene nuestro dataset. 68 columnas contienen datos categóricos y 413 contienen datos cuantitativos.

```
# Completando valores faltantas : datos cuantititavos
for c in cnum:
        mean = ONG_data[c].mean()
        ONG_data[c] = ONG_data[c].fillna(mean)

# Completando valores faltantas : datos categoricos
for c in ctext:
        mode = ONG_data[c].mode()[0]
        ONG_data[c] = ONG_data[c].fillna(mode)

# Verificando que no hayan valores faltantes
ONG_data.isnull().any().any()
```

Con este criterio ya realizado procedemos a guardar nuestro dataframe en disco duro.

```
# Guardando el dataset preprocesado
# Con las transformaciones ya realizadas
ONG_data.to_csv("data/Entrenamiento_procesado.csv",
                index=False)
```

Perfecto! Ahora tenemos un dataset limpio de valores faltantes.

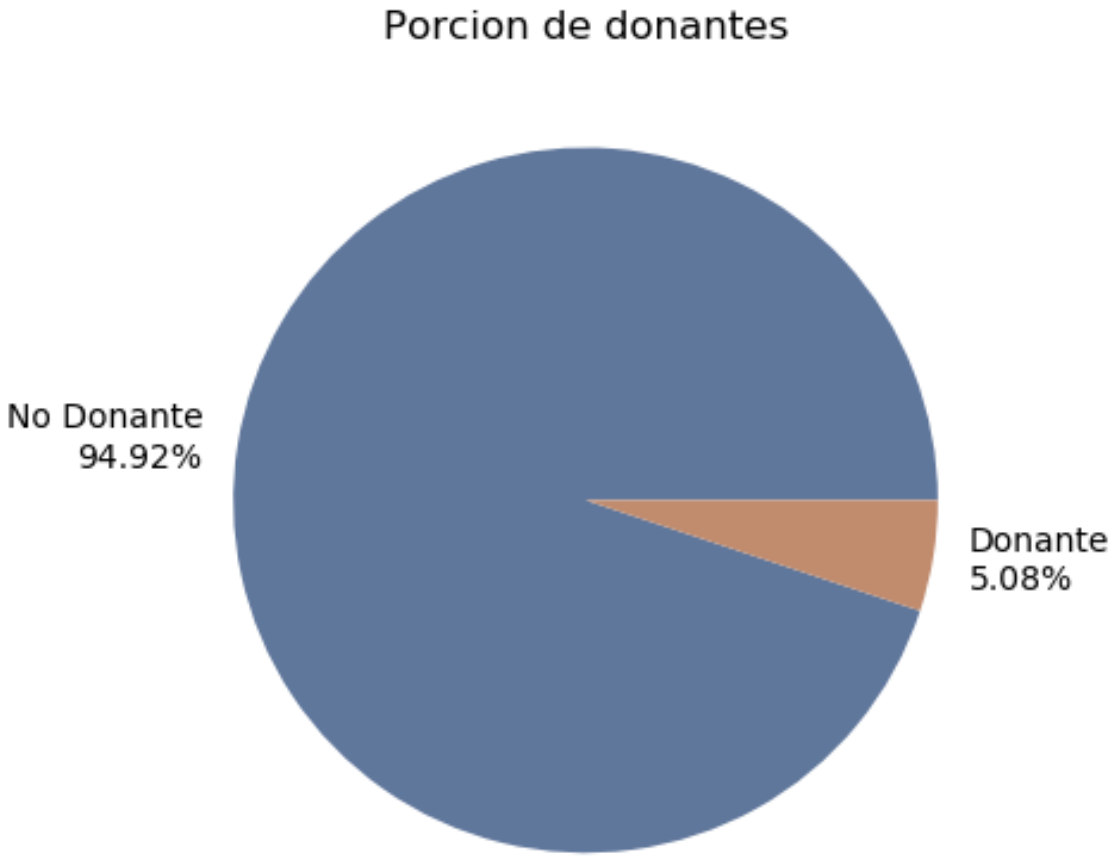
Ya estamos listos para comenzar a explorar los datos, comencemos por determinar el porcentaje de personas que alguna vez fue donante de la ONG y están incluidos en la base de datos con la que estamos trabajando.

```
1  # Calculando el porcentaje de donantes sobre
2  # toda la base de datos
3  porcent_donantes = (ONG_data[ONG_data.DONOR_AMOUNT> 0]
4                                 ['DONOR_AMOUNT'].count() *
5                                 1.0/ ONG_data['DONOR_AMOUNT']
6                                 .count()) * 100.0
7  print("El procentaje de donantes de la base de datos es
8        {0:.2f}%".format(porcent_donantes))
```

Vemos que el resultado es: `El procentaje de donantes de la base de datos es 5.08%`

Proporción de Donantes

```
# Grafico de totas del porcentaje de donantes
# Agrupando por DONOR_FLAG
donantes = ONG_data.groupby('DONOR_FLAG').IDX.count()
# Creando las leyendas del grafico.
labels = [ 'Donante\n' + str(round(x * 1.0/donantes.sum()*
                                   100.0, 2)) + '%'
                                   for x in donantes ]
labels[0] = 'No ' + labels[0]
plt.pie(donantes, labels=labels)
plt.title('Porcion de donantes')
plt.savefig("data/Pie_Donantes.png")
```


Porcion de donantes
No Donante
94.92%
Donante
5.08%

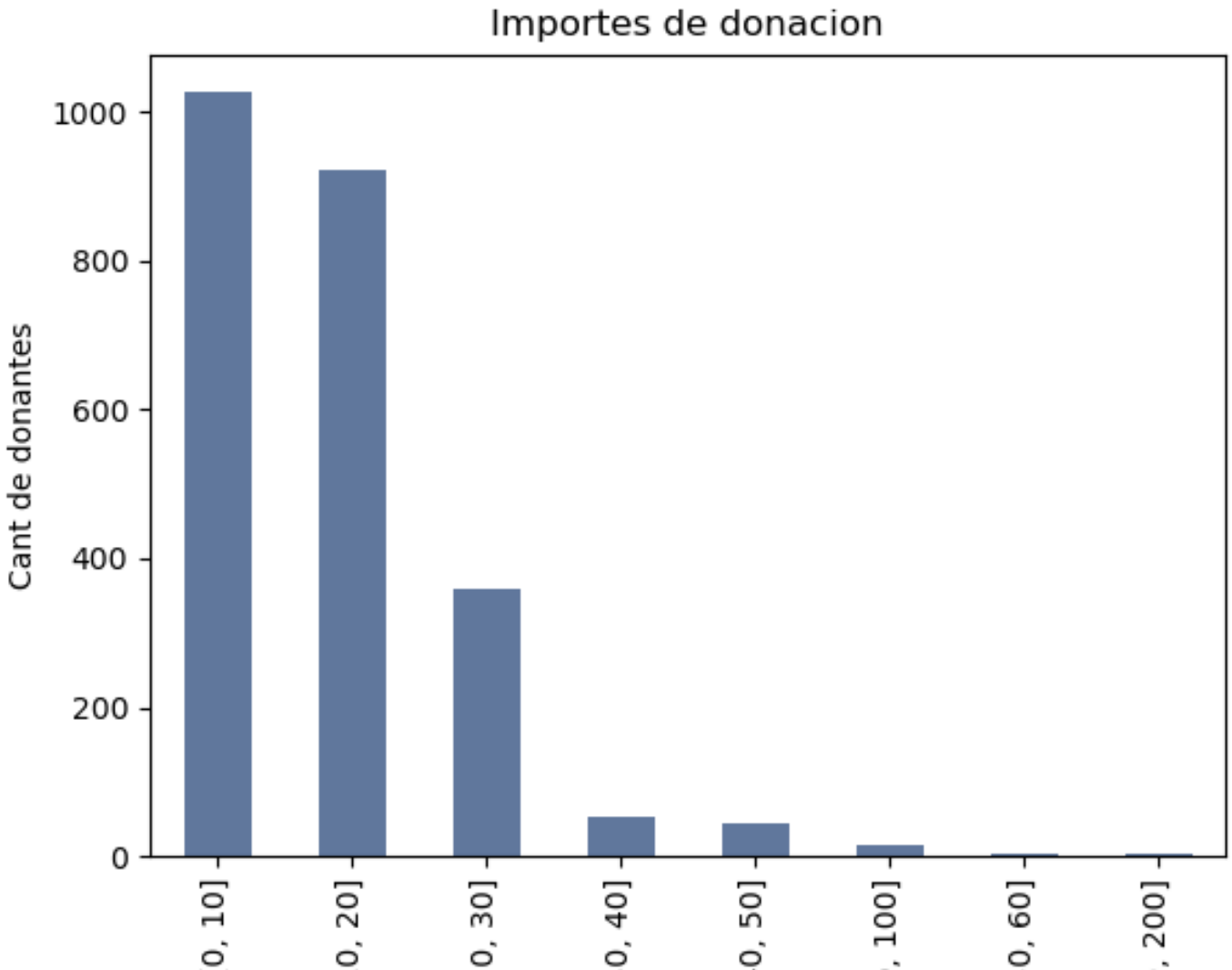
```
# Creando subset con solo los donates
ONG_donantes = ONG_data[ONG_data.DONOR_AMOUNT > 0]

# cantidad de donantes
len(ONG_donantes)
```

Aquí podemos ver que el porcentaje de personas que fueron donantes en el pasado es realmente muy bajo, solo un 5% del total de la base de datos (**2423** personas). Este es un dato importante a tener en cuenta ya que al existir tanta diferencia entre las clases a clasificar, esto puede afectar considerablemente a nuestro algoritmo de aprendizaje.

Exploremos también un poco más en detalle a este grupo pequeño de personas que fueron donantes; veamos por ejemplo como se dividen de acuerdo a la cantidad de dinero donado.

```
# Analizando el importe de donanciones
# Creando un segmentos de importes
imp_segm = pd.cut(ONG_donantes['DONOR_AMOUNT'],
                  [0, 10,  20, 30, 40, 50, 60, 100, 200])
# Creando el grafico de barras desde pandas
plot = pd.value_counts(imp_segm).plot(kind='bar',
                                      title='Importes[donacion]')
plot.set_ylabel('Cant de donantes')
plot.set_xlabel('Rango de importes')
plt.savefig("data/Histograma_Donantes.png")
```

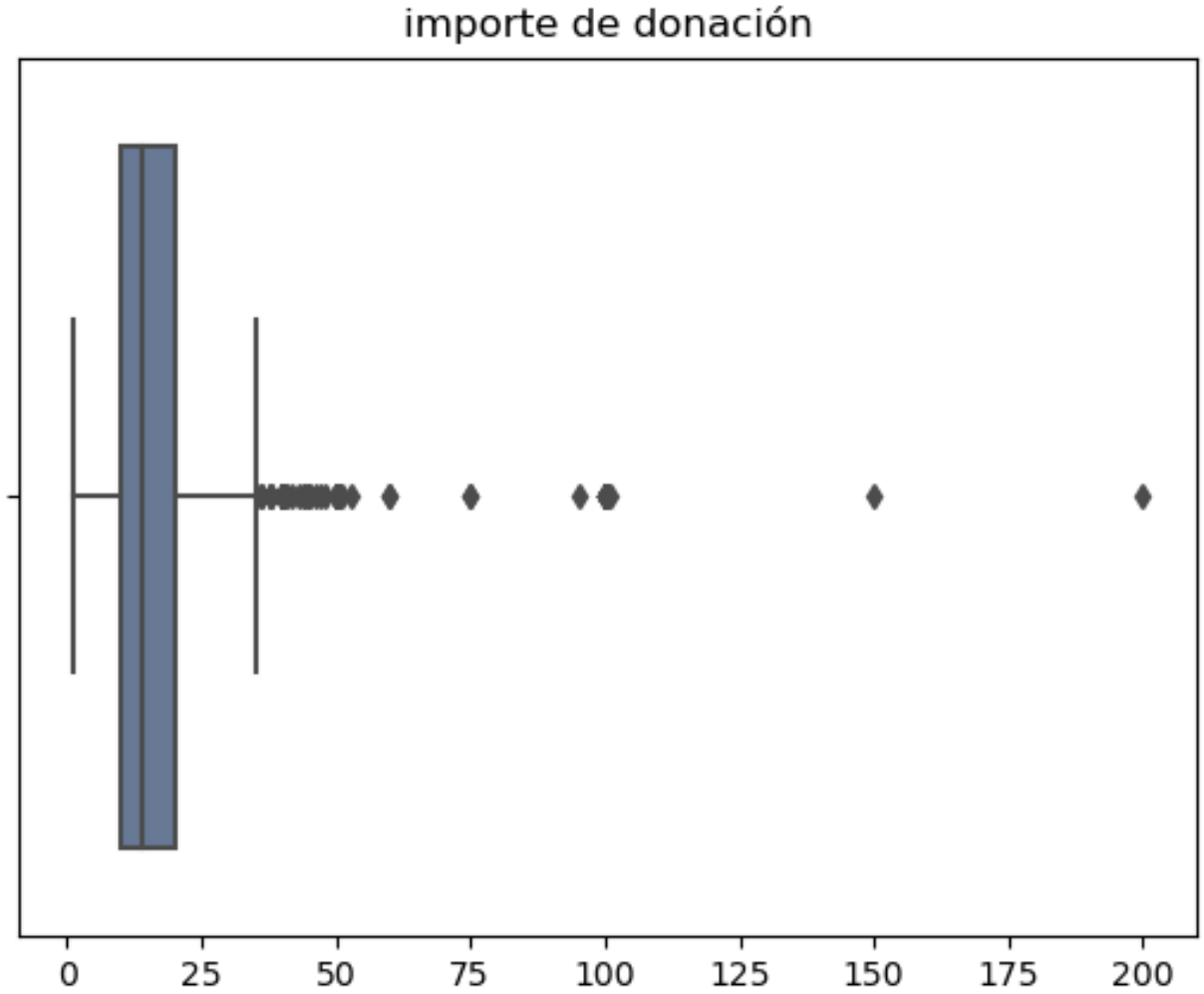

Importes de donacion
Cant de donantes
1000
800
600
400
200
0
0, 10]
0, 20]
0, 30]
0, 40]
0, 50]
, 100]
0, 60]
, 200]

```
# Agrupacion por segmento segun importe donado.
pd.value_counts(imp_segm)
```

```
(0, 10]        1026
(10, 20]        921
(20, 30]        358
(30, 40]         53
(40, 50]         43
(60, 100]        15
(50, 60]          4
(100, 200]        3
Name: DONOR_AMOUNT, dtype: int64
```

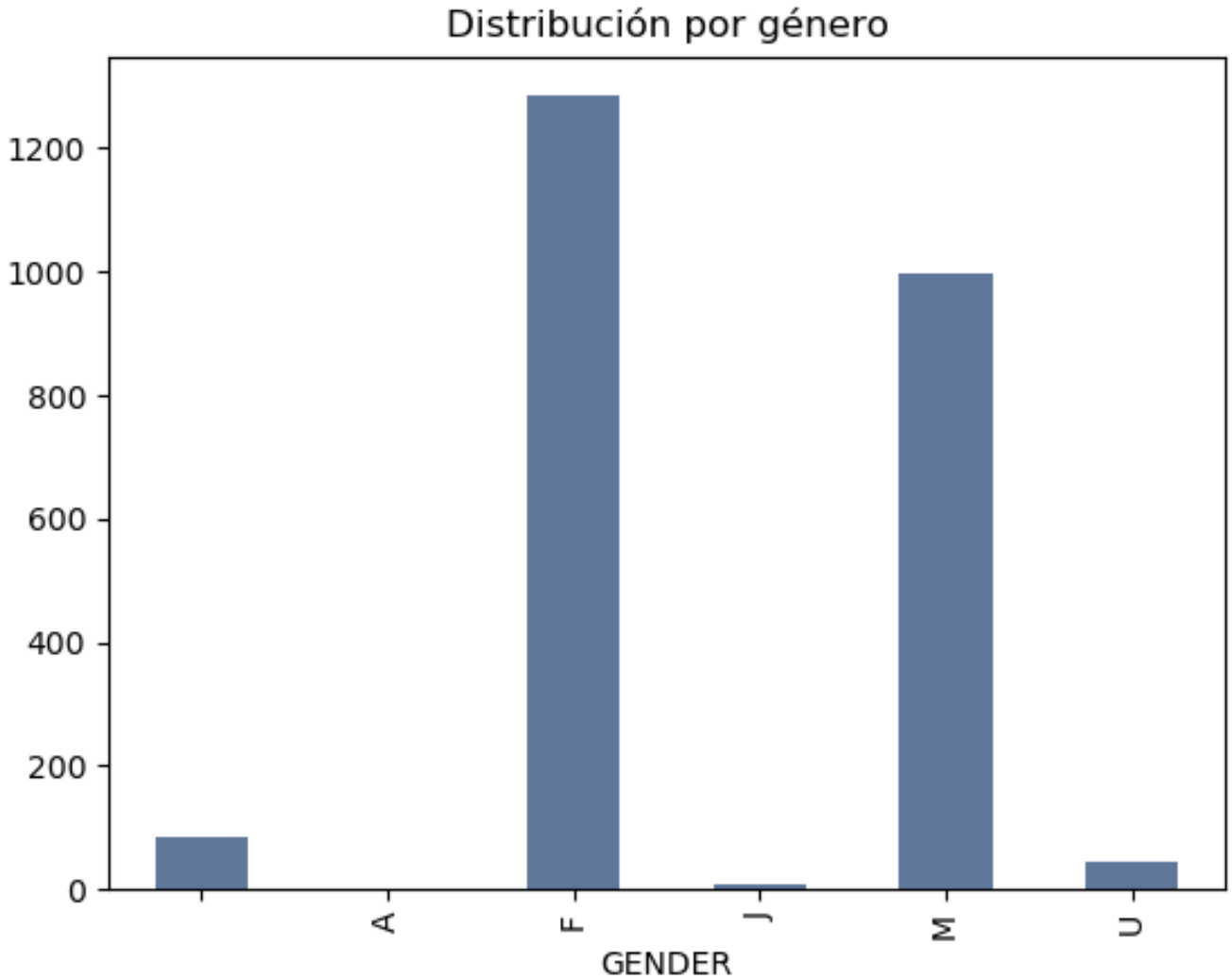
```
# importe de donacion promedio
ONG_donantes['DONOR_AMOUNT'].mean()


# Grafico de cajas del importe de donacion
sns.boxplot(list(ONG_donantes['DONOR_AMOUNT']))
plt.title('importe de donacion')
plt.savefig("data/Boxplot_Donacion.png")
```

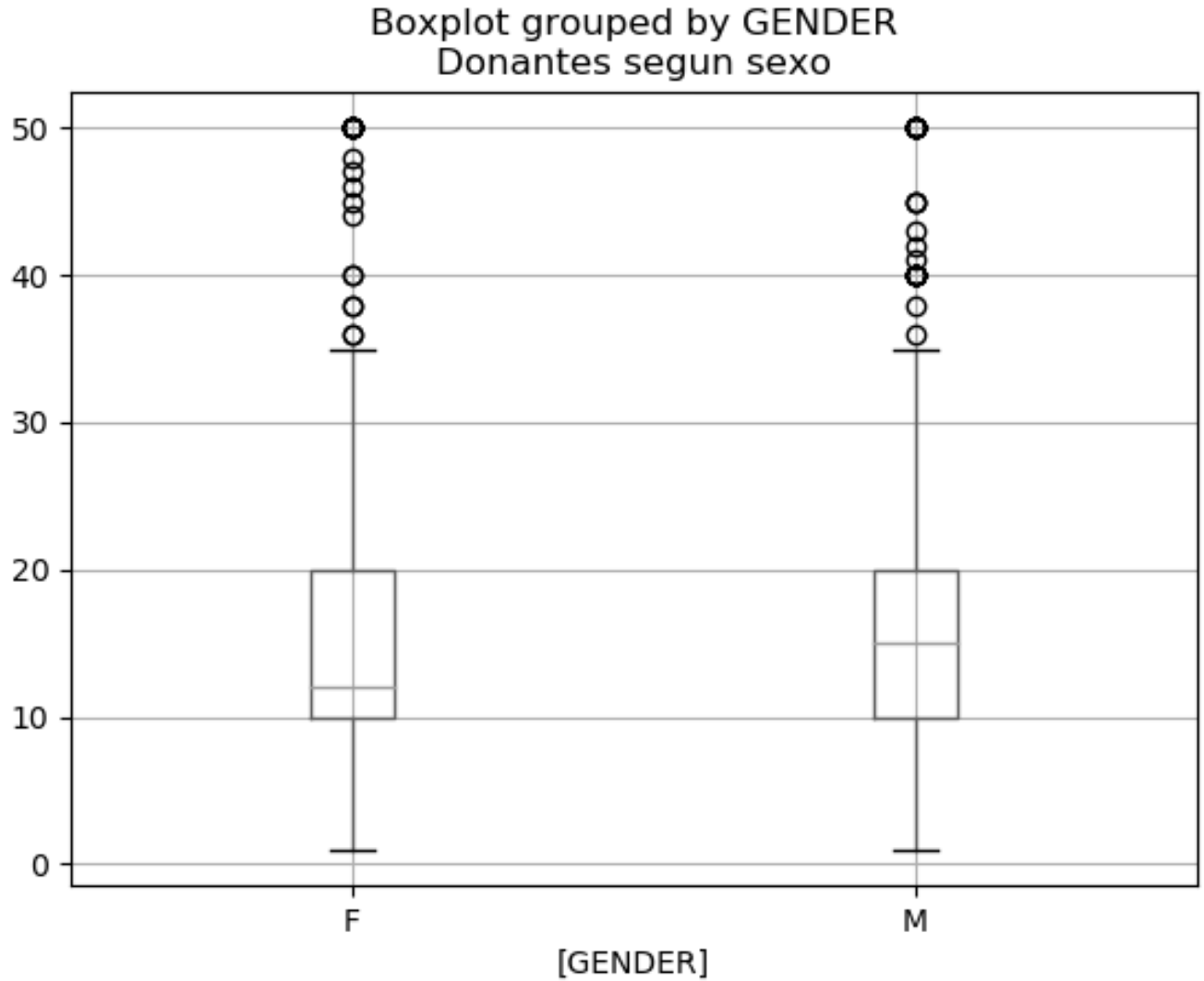

importe de donación
0
25
50
75
100
125
150
175
200

Este análisis nos muestra que la mayor cantidad de donaciones caen en un rango de importes entre 0 y 30, siendo la donación promedio 15.60. También podemos ver que donaciones que superen un importe de 50 son casos realmente poco frecuentes, por lo que constituyen valores atípicos y sería prudente eliminar estos casos al entrenar nuestro modelo para que no distorsionen los resultados.

```
# Grafico del genero de los donantes
ONG_donantes.groupby('GENDER').size().plot(kind='bar')
plt.title('Distribucion por genero')
plt.savefig("data/GrafBarras_Genero.png")
```


Distribución por género
1200
1000
800
600
400
200
0
A
F
J
M
U
GENDER

```
# Donaciones segun el genero
ONG_donantes[(ONG_donantes.DONOR_AMOUNT <= 50)
              & (ONG_donantes.GENDER.isin(['F', 'M'])
                 )][['DONOR_AMOUNT', 'GENDER']]
                 .boxplot(by='GENDER')
plt.title('Donantes segun sexo')
plt.savefig("data/Boxplot_genero_FM-png")
```


Boxplot grouped by GENDER
Donantes segun sexo
50
40
30
20
10
0
F
M
[GENDER]

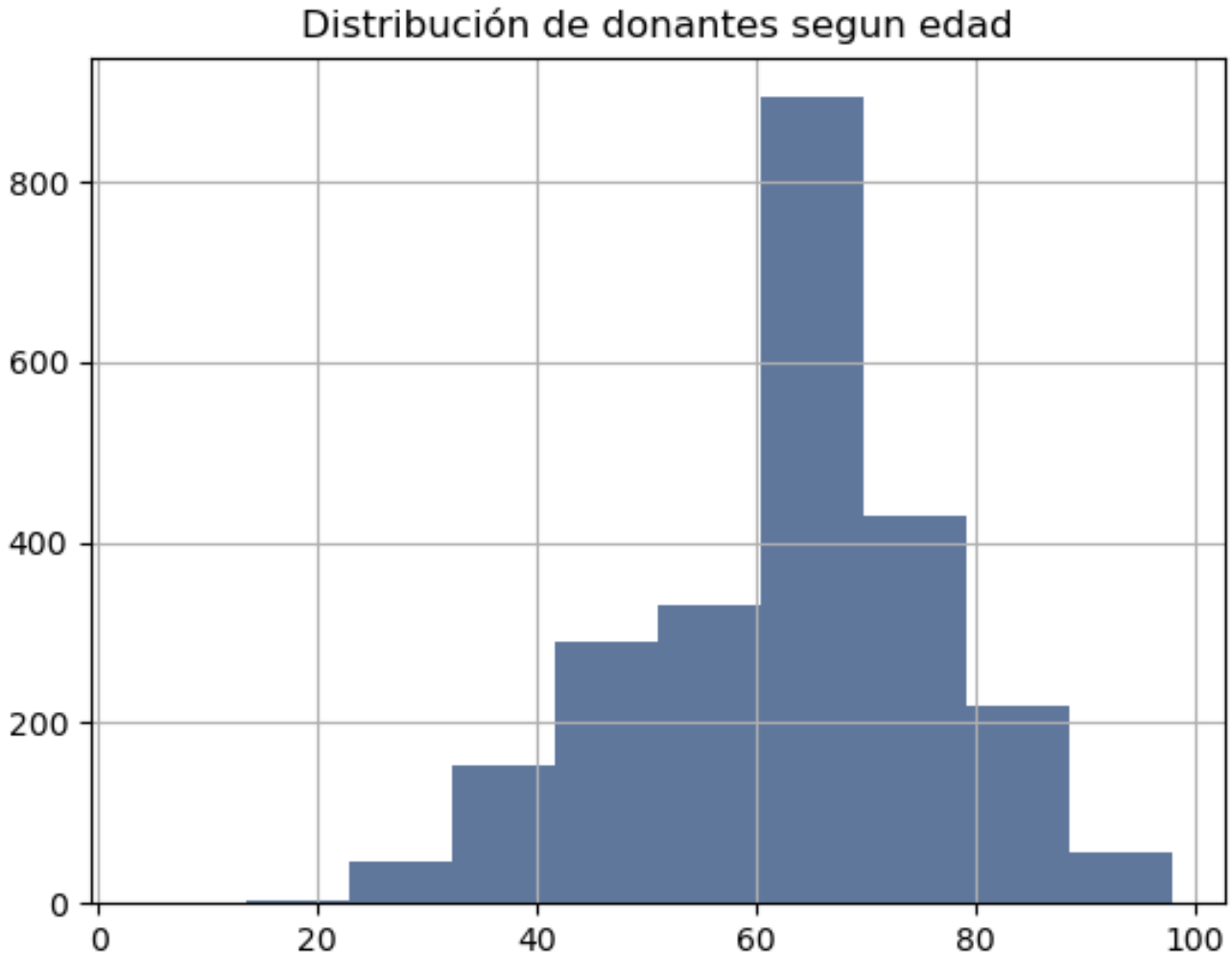
```
# Media de impote donado por mujeres
ONG_donantes[ONG_donantes.GENDER == 'F'][['DONOR_AMOUNT']]
             .mean()


# Media de impote donado por hombres
ONG_donantes[ONG_donantes.GENDER == 'M'][['DONOR_AMOUNT']]
             .mean()
```

Aquí vemos que las mujeres suelen estar más propensas a donar, aunque donan un importe promedio menor (14.61) al que donan los hombres (16.82).

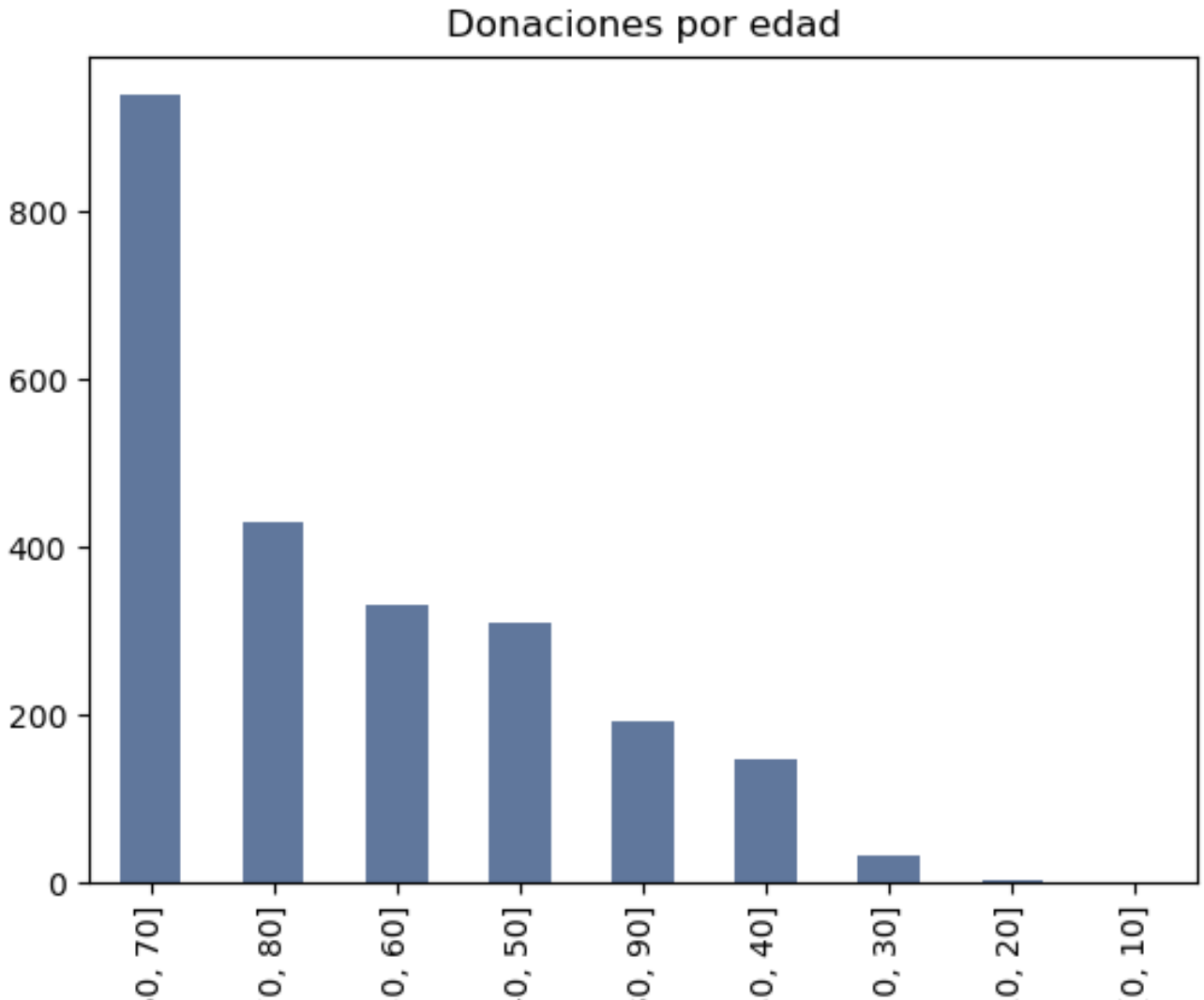
Veamos ahora como se comportan las donaciones respecto a la

```
# Distribucion de la edad de los donantes
ONG_donantes['AGE'].hist()
                   .set_title('Distribucion de
                                donantes segun edad')
plt.savefig("data/hist_edad-png")
```

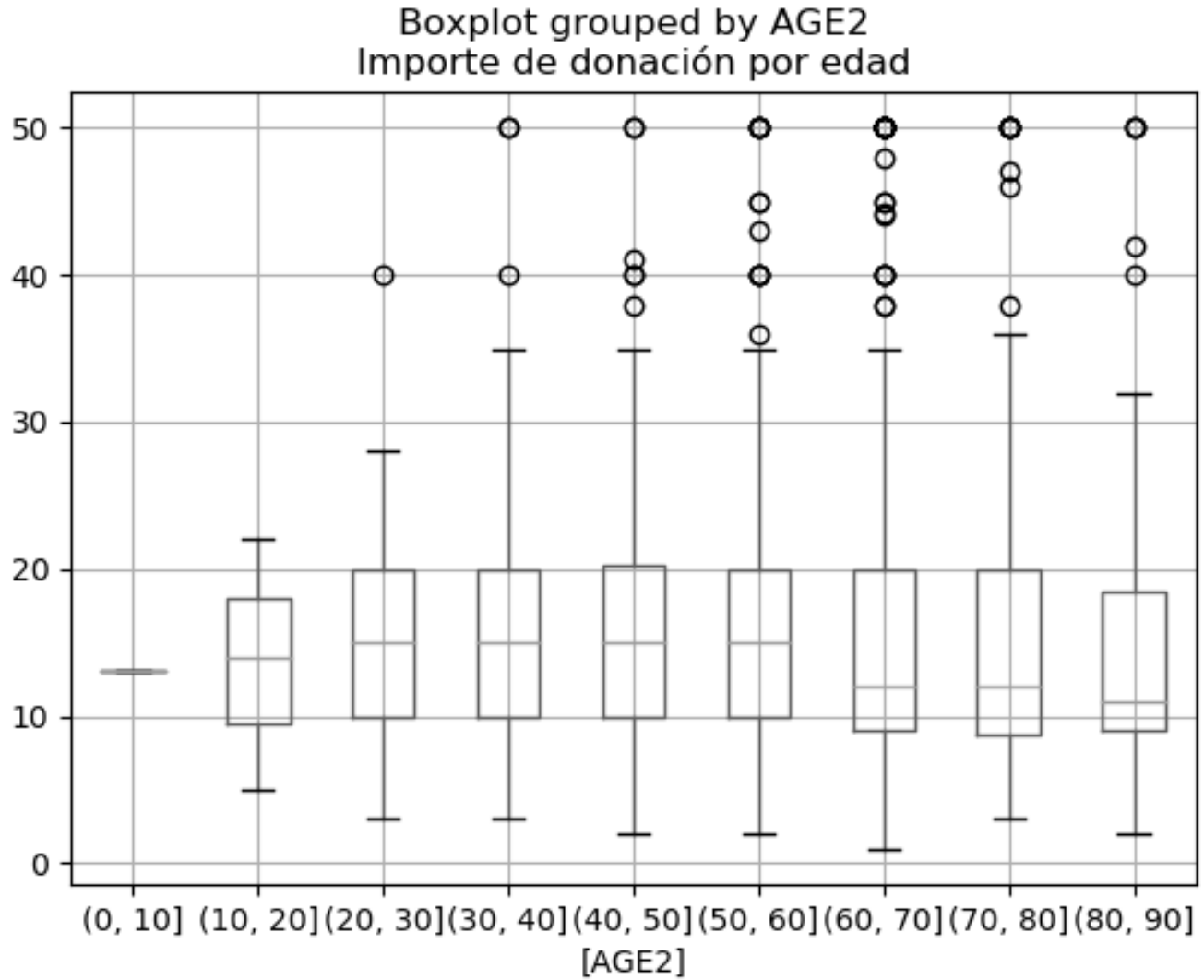

Distribución de donantes segun edad
800
600
400
200
0
0
20
40
60
80
100

```
# Agrupando la edad por rango de a 10
AGE2 = pd.cut(ONG_donantes['AGE'], range(0, 100, 10))
ONG_donantes['AGE2'] = AGE2

# Grafico de barras de donaciones por edad
pd.value_counts(AGE2).plot(kind='bar',
                           title='Donaciones por edad')
plt.savefig("data/barras_edad.png")
```


Donaciones por edad
800
600
400
200
0
0, 70]
0, 80]
0, 60]
0, 50]
0, 90]
0, 40]
0, 30]
0, 20]
0, 10]

```
# Importes de donacion por grango de edad
ONG_donantes[ONG_donantes.DONOR_AMOUNT <= 50][['DONOR_AMOUNT',
              'AGE2']].boxplot(by='AGE2')
plt.title('Importe de donacion por edad')
plt.savefig("data/ImporteEdad.png")
```


Boxplot grouped by AGE2
Importe de donación por edad
50
40
30
20
10
0
(0, 10]
(10, 20]
(20, 30]
(30, 40]
(40, 50]
(50, 60]
(60, 70]
(70, 80]
(80, 90]
[AGE2]

En este último análisis podemos ver que la mayor cantidad de los donantes son personas de entre 60 y 70 años, aunque la media de importe donado más alta la tienen las personas que van desde los 30 a los 60 años.

1 _____

_____

1 _____

_____

1 _____

_____