```
CARATULA
```

Introducción

El mercado de teléfonos móviles nos ofrece muchas variedades y formatos según nuestras necesidades o intereses.El público puede acceder múltiples celulares que ofrecen más o menos rendimiento en función de su presupuesto. Algunos se inclinan por marcas que están más asentadas en el mercado, pero, otros se inclinan por las características de los dispositivos. Este criterio va desde la potencia de la batería, tamaño del dispositivo, memoria ram, etc. Por lo tanto, existiendo tantas posibilidades deberíamos acercarnos al mejor dispositivo calidad-precio que esté a nuestro alcance. Una empresa incipiente de telefonía móvil quiere encontrar cuáles son las características que inciden en el precio de estos dispositivos. Para resolver esto, se recopilaron datos de ventas de teléfonos móviles de varias empresas.

Las variables que contiene el dataset son las siguientes:

- battery_power (Potencia)
- m_dep
- sc_h (Altura pantalla)
- blue (bluetooth)
- mobile_wt (peso)
- sc_w (Ancho pantalla)
- clock_speed (Velocidad)
- n_cores (N° cores)
- talk_time (Tiempo llamada)
- dual_sim (Doble SIM)
- pc
- three_g (3G)
- fc (frec)
- px_height (Altura en pix)
- touch_screen (Pant. táctil)
- four_g (4G)
- px_width (Ancho en pix)
- wifi
- int_memory (Memoria Int)
- ram (Memoria RAM)
- price_range (Rango Precio)

Objetivos

- Determinar qué características inciden directamente en el valor de un teléfono celular.

Herramientas

En este trabajo utilizaremos el entorno de Júpiter para poder realizar el análisis del dataset, asimismo trabajaremos con el lenguaje de programación python3, el tratamiento de la base de datos y su limpieza la realizaremos con la librería pandas, mientras que la visualización de gráficas se hará a través de matplotlib y seaborn.

Importamos todas las librerias necesarias

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Leemos el archivo que contiene los datos a utilizar y lo mostramos, para ver como esta conformado

In [17]:
```
df = pd.read_csv("celulares.xlsx - train.csv")
df.iloc[:, :13]
```

Out[17]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES	
0	842.0	0.0	22.0	0.0	1.0	0.0	7.0	6.0	188	2.0	2
1	1021.0	1.0	5.0	1.0	0.0	1.0	53.0	7.0	136	3.0	6
2	563.0	1.0	5.0	1.0	2.0	1.0	41.0	9.0	145	5.0	6
3	615.0	1.0	25.0	0.0	0.0	0.0	10.0	8.0	131	6.0	9
4	1821.0	1.0	12.0	0.0	13.0	1.0	44.0	6.0	141	2.0	14
...	...	...	...	...	...	...	...	...	...	...	
1995	794.0	1.0	5.0	1.0	0.0	1.0	2.0	8.0	106	6.0	14
1996	1965.0	1.0	26.0	1.0	0.0	0.0	39.0	2.0	187	4.0	3
1997	1911.0	0.0	9.0	1.0	1.0	1.0	36.0	7.0	108	8.0	3
1998	1512.0	0.0	9.0	0.0	NaN	1.0	46.0	1.0	145	5.0	5
1999	510.0	1.0	20.0	1.0	5.0	1.0	45.0	9.0	168	NaN	16

2000 rows × 13 columns

Vemos la cantidad de filas y columnas en el Dataset

In [18]:
```
df.shape
```

Out[18]:
```
(2000, 21)
```

Una vez que sabemos que la base de datos contiene informacion de 2000 celulares, vemos si hay columnas que tengan datos nulos o faltantes

In [19]:
```
resultado = df.isnull().any()
print(resultado.to_string())
```

```
battery Power      True
Blue               True
clock  SPEED       True
Dual_Sim           True
fc                 True
fourg 4G           True
internal memory    True
m_dep              True
mobile_weight      True
number CORES       True
pc                 True
PIXEL height       True
PIXEL width        True
RAM                True
Screen height      True
SCREEN width       True
talk Time          True
threeg3G           True
touch_screen       True
WI-FI              True
PRICE range        True
```

Como se puede observar todas en las columnas faltan datos, para elegir alguna tecnica para la manipulacion de datos faltantes vamos a contar la cantidad de datos que faltan en el dataset

In [20]:
```
var = df.columns.tolist()
print(var)
```

```
['battery Power', 'Blue', 'clock  SPEED', 'Dual_Sim', 'fc', 'fourg 4G', 'inte
rnal memory', 'm_dep', 'mobile_weight', 'number CORES', 'pc', 'PIXEL height',
'PIXEL width', 'RAM', 'Screen height', 'SCREEN width', 'talk Time', 'threeg3
G', 'touch_screen', 'WI-FI', 'PRICE range']
```

In [21]:
```
cant_nulos = 0
for elem in var:
    cont=df[elem].isnull().sum()
    print(f"la cantidad de valores nulos en {elem} es: ",cont)
    cant_nulos += cont
```

```
la cantidad de valores nulos en battery Power es:  1
la cantidad de valores nulos en Blue es:  2
la cantidad de valores nulos en clock  SPEED es:  5
la cantidad de valores nulos en Dual_Sim es:  11
la cantidad de valores nulos en fc es:  133
la cantidad de valores nulos en fourg 4G es:  5
la cantidad de valores nulos en internal memory es:  9
la cantidad de valores nulos en m_dep es:  6
la cantidad de valores nulos en mobile_weight es:  10
la cantidad de valores nulos en number CORES es:  7
la cantidad de valores nulos en pc es:  119
la cantidad de valores nulos en PIXEL height es:  10
la cantidad de valores nulos en PIXEL width es:  10
la cantidad de valores nulos en RAM es:  10
la cantidad de valores nulos en Screen height es:  8
la cantidad de valores nulos en SCREEN width es:  8
la cantidad de valores nulos en talk Time es:  9
la cantidad de valores nulos en threeg3G es:  5
la cantidad de valores nulos en touch_screen es:  10
la cantidad de valores nulos en WI-FI es:  9
la cantidad de valores nulos en PRICE range es:  7
```

La cantidad de datos que faltan en todo el dataset es

In [24]:
```
cant_nulos
```

Out[24]:
```
394
```

Los datos faltantes son 394, que equivaldría a más de un 10% de la base de datos. Podríamos realizar el proceso de limpieza para eliminarlos, pero estaríamos ignorando una parte importante, la pérdida significativa de información. Esta pérdida modifica la distribución de nuestro dataset provocando errores en nuestro análisis. Por lo tanto, vamos a imputar los datos usando la media de cada columna al momento de utilizarla, de modo de evitar alteraciones en la distribución.

Para poder llevar a cabo el proceso descripto anteriormente debemos reemplazar todos los valores nulos y faltantes por 0, para luego realizar la imputacion de los mismos.

In [26]:
```
df.fillna(0, inplace=True)
df.iloc[:, :13]
```

Out[26]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES	I
0	842.0	0.0	22.0	0.0	1.0	0.0	7.0	6.0	188	2.0	2
1	1021.0	1.0	5.0	1.0	0.0	1.0	53.0	7.0	136	3.0	6
2	563.0	1.0	5.0	1.0	2.0	1.0	41.0	9.0	145	5.0	6
3	615.0	1.0	25.0	0.0	0.0	0.0	10.0	8.0	131	6.0	9
4	1821.0	1.0	12.0	0.0	13.0	1.0	44.0	6.0	141	2.0	14
...	...	...	...	...	...	...	...	...	...	...	
1995	794.0	1.0	5.0	1.0	0.0	1.0	2.0	8.0	106	6.0	14
1996	1965.0	1.0	26.0	1.0	0.0	0.0	39.0	2.0	187	4.0	3
1997	1911.0	0.0	9.0	1.0	1.0	1.0	36.0	7.0	108	8.0	3
1998	1512.0	0.0	9.0	0.0	0.0	1.0	46.0	1.0	145	5.0	5
1999	510.0	1.0	20.0	1.0	5.0	1.0	45.0	9.0	168	0.0	16

2000 rows × 13 columns

Al intentar ordenar la columna `mobile_weight` nos encontramos con un valor `'-'` que aparece en esta columna, entonces procedemos a reemplazarlo por 0, para luego poder reemlazo por el promedio de la columna

In [27]:
```
for x in df.index:
    if df.loc[x,'mobile_weight'] == '-':
        df.loc[x, 'mobile_weight'] = '0'
```

In [29]:
```
df = df.astype(int)
df = df.sort_values('mobile_weight',ascending=True)
df.iloc[:, :13]
```

Out[29]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES	pc
362	1976	1	7	1	0	0	32	8	-180	2	7
472	1077	0	25	1	0	1	45	10	-174	3	4
465	1583	1	16	0	5	1	42	8	-118	3	10
1169	763	1	5	0	9	1	51	3	0	0	0
280	1733	1	28	0	0	0	36	10	0	6	0
...	...	...	...	...	...	...	...	...	...	...	...
888	1578	0	29	0	8	0	3	9	200	1	20
846	1489	0	24	1	8	0	32	6	200	1	16
29	851	0	5	0	3	0	21	4	200	5	7
485	1277	0	30	0	0	1	41	3	200	4	3
1690	1606	1	5	1	0	1	33	9	200	5	0

2000 rows × 13 columns

Una vez ordenada la columna `mobile_weight` vamos a reemplazar los valores 0 y los negativos por el promedio de la misma

In [31]:
```
# Calcular el promedio de los valores mayores a 0 en la columna 'mobile_weight
promedio = df['mobile_weight'][df['mobile_weight'] > 0].mean()


promedio
```

Out[31]:
```
140.27139979859012
```

Vamos a reemlazar los datos de `mobile_weight` por `140`

In [45]:
```
for x in df.index:
    if df.loc[x,'mobile_weight'] <= 0:
        df.loc[x, 'mobile_weight'] = round(promedio)

df.iloc[9:20, :13]
```

Out[45]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES	pc
1945	1007	1	0	0	0	0	0	0	140	0	0
1994	858	0	0	0	0	0	0	0	140	0	0
1185	1136	0	5	0	0	1	9	0	140	0	0
1940	1600	0	0	0	0	0	0	0	140	0	0
20	772	0	11	1	0	0	39	8	140	7	14
121	772	1	24	1	1	1	10	5	80	4	2
1426	1454	0	14	1	8	0	37	8	80	8	20
1431	1283	1	7	1	0	1	27	2	80	4	6
298	928	1	5	1	11	0	56	7	80	8	13
940	1456	0	16	1	9	1	39	1	80	8	10
311	1707	0	14	0	0	0	41	8	80	7	2

Creamos una nueva columna con el total de pixeles de la pantalla multiplicando el alto y ancho

In [48]:
```
df = df.assign(total_pixeles=df['PIXEL height'] * df['PIXEL width'])
df.iloc[5:9, 9:22]
```

Out[48]:

	number CORES	pc	PIXEL height	PIXEL width	RAM	Screen height	SCREEN width	talk Time	threeg3G	touch_screen	WI-FI	F
1112	2	5	1226	1389	3646	15	4	11	1	1	1	
1987	4	15	591	724	1424	15	12	7	1	1	0	
82	5	3	708	1752	3484	9	6	11	0	1	1	
384	1	20	431	550	3801	10	6	2	1	0	0	

In [51]:
```
# Calcular el promedio de los valores mayores a 0 en la columna 'RAM'
prom = df['RAM'][df['RAM'] > 0].mean()

for x in df.index:
    if df.loc[x,'RAM'] <= 0:
        df.loc[x, 'RAM'] = round(prom)

print('promedio',round(prom),'Mb')
df.iloc[10:15, 5:18]
```

```
promedio 2124 Mb
```

Out[51]:

	fourg 4G	internal memory	m_dep	mobile_weight	number CORES	pc	PIXEL height	PIXEL width	RAM	Screen height	SCREEN widtl
1994	0	0	0	140	0	0	0	0	2124	0	
1185	1	9	0	140	0	0	0	0	2124	0	
1940	0	0	0	140	0	0	0	0	2124	0	
20	0	39	8	140	7	14	1314	1854	2819	17	15
121	1	10	5	80	4	2	1242	1712	3242	18	1

Reemplazamos los valores de bateria por baja y alta energia

In [52]:
```
datos_limpios = df.copy()
for x in df.index:
    if df.loc[x,'battery Power'] < 1000:
        df.loc[x, 'battery Power'] = 'Baja_Energia'
    else: df.loc[x, 'battery Power'] = 'Alta_Energia'
df.iloc[:, :13]
```

Out[52]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES
362	Alta_Energia	1	7	1	0	0	32	8	140	2
472	Alta_Energia	0	25	1	0	1	45	10	140	3
465	Alta_Energia	1	16	0	5	1	42	8	140	3
1169	Baja_Energia	1	5	0	9	1	51	3	140	0
280	Alta_Energia	1	28	0	0	0	36	10	140	6
...	...	...	...	...	...	...	...	...	...	...
888	Alta_Energia	0	29	0	8	0	3	9	200	1
846	Alta_Energia	0	24	1	8	0	32	6	200	1
29	Baja_Energia	0	5	0	3	0	21	4	200	5
485	Alta_Energia	0	30	0	0	1	41	3	200	4
1690	Alta_Energia	1	5	1	0	1	33	9	200	5

2000 rows × 13 columns

In [68]:
```
bateria = df['battery Power'].value_counts()
bateria = bateria.to_string()
print(bateria)
```

```
Alta_Energia    1305
Baja_Energia     695
```

Para la variable ‘battery power’ modificamos a una variable cualitativa ordinal utilizando el criterio de: `Baja Energía` < 1000 W y `Alta Energía` > 1000 W. Lo que observamos es que en la base de datos hay más dispositivos de alta energía (1305) que de baja energía (695).

Reemlazamos los valores de peso por bajo (140), medio hasta(170) y alto

In [36]:
```
for x in df.index:
    if df.loc[x,'mobile_weight'] < 140:
        df.loc[x, 'mobile_weight'] = 'Bajo'
    elif df.loc[x, 'mobile_weight'] > 140 and df.loc[x, 'mobile_weight'] < 170:
        df.loc[x, 'mobile_weight'] = 'Medio'
    else: df.loc[x, 'mobile_weight'] = 'Alto'
df
```

Out[36]:

	battery Power	Blue	clock SPEED	Dual_Sim	fc	fourg 4G	internal memory	m_dep	mobile_weight	number CORES
362	Alta_Energia	1	7	1	0	0	32	8	Alto	2
472	Alta_Energia	0	25	1	0	1	45	10	Alto	3
465	Alta_Energia	1	16	0	5	1	42	8	Alto	3
1940	Alta_Energia	0	0	0	0	0	0	0	Alto	0
1185	Alta_Energia	0	5	0	0	1	9	0	Alto	0
...	...	...	...	...	...	...	...	...	...	...
1368	Baja_Energia	1	23	0	4	0	51	4	Alto	8
282	Alta_Energia	1	12	0	9	1	54	5	Alto	7
1830	Baja_Energia	1	16	0	9	1	60	4	Alto	1
1588	Alta_Energia	1	20	0	11	0	35	1	Alto	4
1690	Alta_Energia	1	5	1	0	1	33	9	Alto	5

2000 rows × 22 columns

Guardamos los datos limpios en un archivo csv

In [37]:
```
#df.to_csv('datos_limpios.csv', index=False)
```

buscamos las frecuencias para 3 y 4 G

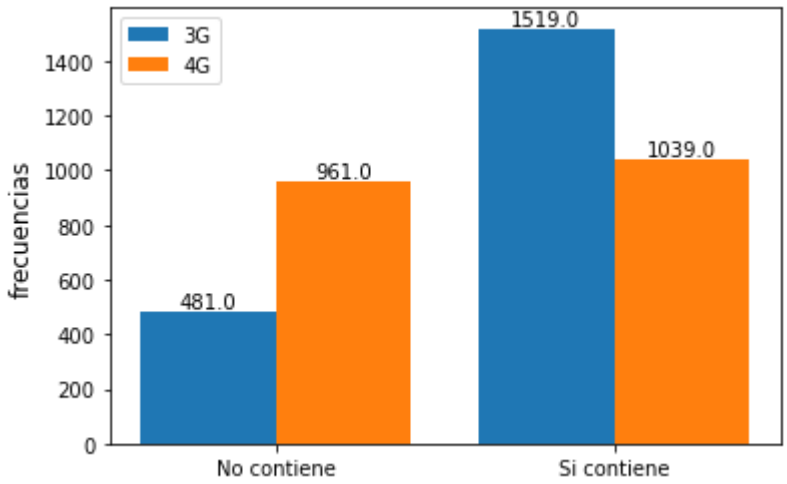
In [38]:
```
#NO ANDA BIEN

# Obtener las tablas de frecuencia
#freq_3g = df['threeg3G'].value_counts()
#freq_4g = df['fourg 4G'].value_counts()

# Combinar las tablas de frecuencia
#tabla_freq = pd.concat([freq_3g, freq_4g], axis=1)
#tabla_freq.columns = ['3G', '4G']
#tabla_freq.index.name = 'Valor'

# Crear un gráfico de barras
#tabla_freq.plot(kind='bar')
#plt.xlabel('Valor')
#plt.ylabel('Frecuencia')
#plt.title('Tabla de Frecuencias')

# Guardar la imagen
#plt.savefig('tabla_frecuencias.png')

# Mostrar la tabla de frecuencia
#print(tabla_freq)
```

In [39]:
```
print('3g:\n',df['threeg3G'].value_counts(),'\n 4g\n',df['fourg 4G'].value_cour
```

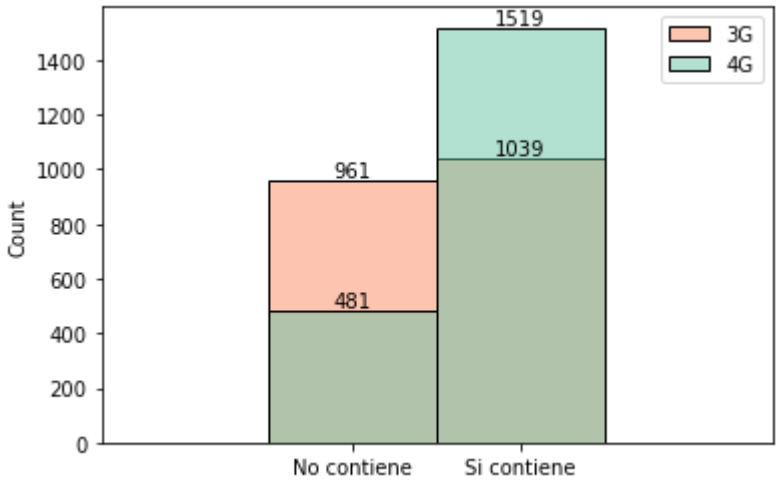
```
3g:
 1    1519
0     481
Name: threeg3G, dtype: int64
 4g
 1    1039
0     961
Name: fourg 4G, dtype: int64
```

Grafico un histograma con estos datos:

In [40]:
```
# Crear una figura y ejes de subtrama
fig, ax = plt.subplots()

# Crear el histograma y obtener las coordenadas de las barras
counts, bins, patches = ax.hist([df['threeg3G'], df['fourg 4G']], histtype='bar

# Agregar los valores numéricos encima de las barras
for p in ax.patches:
    ax.annotate(str(round(p.get_height())), (p.get_x() + p.get_width() / 2, p.g

# Cambiar los valores del eje x
plt.xticks([0.25, 0.75], ['No contiene', 'Si contiene'])

# Centrar los valores del eje x
#plt.xlim(-0.5, 1.5)
#ax.set_xlim([-0.5, 1.5])

# Etiqueta del eje y
plt.ylabel('frecuencias', size=12)

# Agregar una leyenda
plt.legend(('3G', '4G'),loc='upper left')#, bbox_to_anchor=(1.0, 0.5))

# Mostrar el histograma
plt.show()
```



In [41]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Crear el histograma
fig, ax = plt.subplots()
sns.histplot([df['threeg3G'], df['fourg 4G']], ax=ax, bins=2, color=['red', 'bl

# Agregar los valores numéricos encima de las barras
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() + p.get_width() / 2, p.get_heig

# Cambiar los valores del eje x
plt.xticks([0.25, 0.75], ['No contiene', 'Si contiene'])

# Centrar los valores del eje x
plt.xlim(-0.5, 1.5)
ax.set_xlim([-0.5, 1.5])

# Agregar una leyenda
plt.legend(('3G', '4G'))

# Mostrar el histograma
plt.show()
```



NO SE QUE HISTOGRAMA DEJAR

Hago una correlacion de todas las variables contra el precio

In [42]:
```
#datos_limpios['PRICE range']
precio_01 = datos_limpios[datos_limpios['PRICE range'].isin([0, 1])]
precio_23 = datos_limpios[datos_limpios['PRICE range'].isin([2, 3])]

# Crea una tabla con dos columnas y tres filas
data = {'Peso': [round(precio_01['mobile_weight'].mean())],
        'Ram': [round(precio_01['RAM'].mean())],
        'Bateria': [round(precio_01['battery Power'].mean())]}

d_f = pd.DataFrame(data)

# Crea una tabla con dos columnas y tres filas con los valores medios de precio
data_23 = {'Peso': [round(precio_23['mobile_weight'].mean())],
           'Ram': [round(precio_23['RAM'].mean())],
           'Bateria': [round(precio_23['battery Power'].mean())]}
df_23 = pd.DataFrame(data_23)

# Concatena el DataFrame df_23 con el DataFrame original d_f
d_f = pd.concat([d_f, df_23], ignore_index=True)

# Renombra las filas
d_f = d_f.rename(index={0: 'Low Price (0 y 1)', 1: 'High Price (2 y 3)'})

# Muestra la tabla en el notebook
display(d_f)
```

	Peso	Ram	Bateria
Low Price (0 y 1)	141	1242	1173
High Price (2 y 3)	140	3014	1304

In [43]:
```
#datos_limpios['PRICE range']
precio_0 = datos_limpios[datos_limpios['PRICE range'].isin([0])]
precio_3 = datos_limpios[datos_limpios['PRICE range'].isin([3])]

# Crea una tabla con dos columnas y tres filas
data_0 = {'Peso': [round(precio_0['mobile_weight'].mean())],
        'Ram': [round(precio_0['RAM'].mean())],
        'Bateria': [round(precio_0['battery Power'].mean())]}

d_f0 = pd.DataFrame(data_0)

# Crea una tabla con dos columnas y tres filas con los valores medios de precio
data_3 = {'Peso': [round(precio_3['mobile_weight'].mean())],
           'Ram': [round(precio_3['RAM'].mean())],
           'Bateria': [round(precio_3['battery Power'].mean())]}
df_3 = pd.DataFrame(data_3)

# Concatena el DataFrame df_3 con el DataFrame original d_f0
datos = pd.concat([d_f0, df_3], ignore_index=True)

# Renombra las filas
datos = datos.rename(index={0: 'Low Price (0)', 1: 'High Price (3)'})

# Muestra la tabla en el notebook
display(datos)
```

	Peso	Ram	Bateria
Low Price (0)	141	807	1118
High Price (3)	136	3445	1377

In [44]:
```
corr = df.corr()
corr[['PRICE range']].sort_values(by = 'PRICE range',ascending = False).style.b
```

Out[44]:

	PRICE range
PRICE range	1.000000
RAM	0.911680
total_pixeles	0.180237
PIXEL width	0.168762
PIXEL height	0.153423
internal memory	0.041591
SCREEN width	0.033858
talk Time	0.029181
pc	0.026778
threeg3G	0.024128
WI-FI	0.021478
Dual_Sim	0.020604
Blue	0.020496
Screen height	0.020066
fourg 4G	0.012239
number CORES	0.007198
m_dep	-0.001701
fc	-0.004437
clock SPEED	-0.005946
touch_screen	-0.024553

In [45]:
```
# Calcular el promedio de los valores mayores a 0 en la columna 'RAM'
#prom = df['RAM'][df['RAM'] > 0].mean()

#for x in df.index:
#    if df.loc[x,'RAM'] <= 0:
#        df.loc[x, 'RAM'] = round(prom)

#print('promedio',round(prom),'Mb')
#df
```

In [ ]:
```

```

In [ ]:
```
# Crear una lista para almacenar los valores de la suma de los cuadrados de las
#ssd = []

# Convertir la variable RAM en un arreglo 2D con una columna
#ram = df['RAM']
#X = np.array(ram).reshape(-1, 1)

#X = np.array(df['RAM']).reshape(-1, 1)

# Calcular la suma de los cuadrados de las distancias para diferentes valores d
#for k in range(1, 11):
#    kmeans = KMeans(n_clusters=k, random_state=0)
#    kmeans.fit(X)
#    ssd.append(kmeans.inertia_)

# Graficar la suma de los cuadrados de las distancias para diferentes valores d
#plt.plot(range(1, 11), ssd, marker='o')
#plt.xlabel('Número de Clusters')
#plt.ylabel('Suma de los Cuadrados de las Distancias')
#plt.title('Método del Codo')
#plt.show()
```

In [ ]:

In [ ]:
```
#Prompt como haces esto : Indicar el rango y la media de la variable “ram”. Eli
#Perplexiti
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# Seleccionar la variable RAM
ram = df['RAM']

# Calcular la media y el rango de la variable RAM
media = np.mean(ram)
rango = np.ptp(ram)

# Imprimir la media y el rango de la variable RAM
print("Media de la variable RAM:", media)
print("Rango de la variable RAM:", rango)

# Graficar la variable RAM utilizando un histograma
plt.hist(ram, bins=10)
plt.xlabel('RAM')
plt.ylabel('Frecuencia')
plt.title('Histograma de la variable RAM')
plt.show()

# Graficar la variable RAM utilizando un diagrama de caja
plt.boxplot(ram)
plt.ylabel('RAM')
plt.title('Diagrama de caja de la variable RAM')
plt.show()
```

In [ ]:
```
```

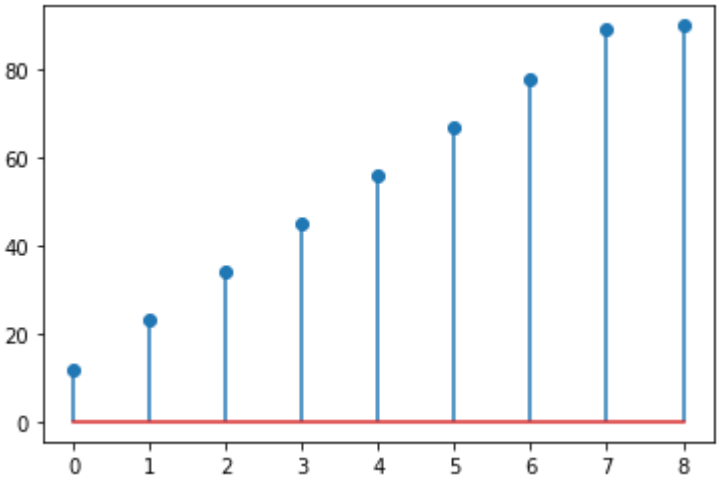
In [ ]:
```
ram = df['RAM']
media = ram.mean()
rango = ram.max() - ram.min()
print("Media de la variable ram:", media)
print("Rango de la variable ram:", rango)
plt.hist(ram, bins=20, color='blue', alpha=0.5)
plt.axvline(media, color='red', linestyle='dashed', linewidth=1)
plt.xlabel('Ram', size=12)
plt.ylabel('Frecuencia', size=12)
plt.title('Distribución de valores de ram en 2000 teléfonos')
plt.show()
```

In [ ]:
```
```

In [ ]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression

X = datos_limpios[['PRICE range']]
y = datos_limpios['RAM']

model = LinearRegression()
model.fit(X, y)

print(model.intercept_)
print(model.coef_)
```

In [ ]:
```
data_=datos_limpios[["PRICE range","RAM","number CORES", "mobile_weight", "batt
sns.pairplot(data_)
plt.show()
```

In [1]:
```
import matplotlib.pyplot as plt

datos = [12, 23, 34, 45, 56, 67, 78, 89, 90]

plt.stem(datos)
plt.show()
```



In [9]:
```
print(df.head().to_string())#max_columns=None))
```

```
   battery Power  Blue  clock  SPEED  Dual_Sim    fc  fourg 4G  internal memo
ry  m_dep mobile_weight  number CORES    pc  PIXEL height  PIXEL width     RA
M  Screen height  SCREEN width  talk Time  threeg3G  touch_screen  WI-FI  PRI
CE range
0          842.0   0.0          22.0       0.0   1.0       0.0
7.0    6.0           188           2.0   2.0          20.0        756.0  254
9.0           9.0          7.0       19.0       0.0           0.0    1.0
1.0
1         1021.0   1.0           5.0       1.0   0.0       1.0            5
3.0    7.0           136           3.0   6.0         905.0       1988.0  263
1.0          17.0          3.0        7.0       1.0           1.0    0.0
2.0
2          563.0   1.0           5.0       1.0   2.0       1.0            4
1.0    9.0           145           5.0   6.0        1263.0       1716.0  260
3.0          11.0          2.0        9.0       1.0           1.0    0.0
2.0
3          615.0   1.0          25.0       0.0   0.0       0.0            1
0.0    8.0           131           6.0   9.0        1216.0       1786.0  276
9.0          16.0          8.0       11.0       1.0           0.0    0.0
2.0
4         1821.0   1.0          12.0       0.0  13.0       1.0            4
4.0    6.0           141           2.0  14.0        1208.0       1212.0  141
1.0           8.0          2.0       15.0       1.0           1.0    0.0
1.0
```

In [14]:
```
corr = df.corr()
corr_filtrado = corr.loc[(corr['PRICE range'] > 0.6) | (corr['PRICE range'] < -
corr_filtrado
```

Out[14]:

	PRICE range
RAM	0.917108
PRICE range	1.000000

In [ ]:
```
```