PRA2: Limpieza y análisis de datos

M2.851 - Tipología y ciclo de vida de los datos

Víctor Blanes Martín
Carlos Allo Latorre

06/06/2021

Tabla de contenido

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

Hoy en día, la compra de dispositivos electrónicos y en especial de portátiles, está a la orden del día y es más frecuente que nunca. A raíz del desarrollo de nuevas tecnologías, cada vez es más frecuente disponer de una oferta de productos más amplia, excesiva en ocasiones, que dificulta la toma de decisiones en cuanto a la compra de dichos productos.

En función del perfil del comprador, el desconocimiento de la gama de productos y sus características puede que los lleve a tomar una decisión de compra poco adecuada o ajustada en precio. Por este motivo, sería interesante conocer qué variables o características son más influyentes en el precio a la hora de compra de un ordenador, para que de esta manera, el comprador:

- Pueda hacerse una idea del presupuesto aproximado que tendrá el ordenador que desea en base a las características técnicas deseadas.
- Pueda conocer si es verdad que algunas marcas, como Apple, tienen un precio algo superior al resto de las marcas.
- Pueda conocer qué características son las más influyentes en el precio para en base a estas, poder centrarse en lo que realmente necesita para incrementar o decrementar su presupuesto.

Para ello, se podría usar el dataset de ordenadores portátiles construido en la práctica 1, en donde, entre otros objetivos a responder, el recién presentado era uno de ellos. Sin embargo, tras un comienzo con este dataset, surgió el problema de que no poseían suficientes datos ni características como para realizar buenos análisis ni modelos, por lo que se decidió buscar bajo el mismo objetivo a tratar, otro dataset. Tras la búsqueda se seleccionó el conjunto de datos "Laptop Prices" de Kaggle (https://www.kaggle.com/ionaskel/laptop-prices), que posee 1303 registros de ordenadores con sus correspondientes características y precio.

En el apartado 4, una vez los datos han sido preparados y limpiados, se mostrará visualmente y de una manera gráfica el dataset para entender de qué variables se parte para poder hacer un buen análisis posterior.

2. Integración y selección de los datos de interés a analizar.

La integración de los datos será mínima ya que todos los datos a usar provienen de un mismo dataset. Como se usará el lenguaje de programación Python, en términos de programación la única importación que se realizará será la de carga del dataset al entorno, que se realizará mediante pandas apuntando al archivo comentado, que ha sido importado al GitHub del proyecto.

```
# Se carga el archivo con referencia al gitHub donde se encuentra el mismo
laptops_file = 'https://raw.githubusercontent.com/carlosalloUOC/PRA2-Limpieza-Analisis/main/csv/laptops_inicial.csv'

# Se lee el fichero anterior, para transformarlo en un dataframe indicándole que en la primera línea se encuentran las cabecera
laptops_initial = pandas.read_csv(laptops_file, header=0, encoding='latin-1')

# Comprobamos que tiene las dimensiones correctas
print(laptops_initial.shape)
```

```
(1303, 13)
```

Cabe resaltar, que si se deseara usar dos datasets diferentes con las mismas características, se debería de realizar una integración de estos, donde habría que tener en cuenta aspectos como posibles repeticiones de objetos, que las propiedades se presenten en las mismas unidades... Este

proceso se realizó en la práctica 1, en el momento en el que se integraban dos datasets diferentes (uno de cada web en donde se realizó WebScraping), en uno sólo.

Respecto a la selección de los datos, nos quedaremos con todas las filas, ya que cada una de ellas corresponde a un ordenador diferente y aporta información al estudio que se está realizando. Además, nos encontramos ante un número de ordenadores (aproximadamente 1300) no tan grande como para tener que hacer reducción de la cantidad. Sin embargo, para el caso de tener que aplicar dicha técnica, consideramos que las dos mejores formas de hacerlo serían el método de muestra aleatoria simple sin sustitución (para no tener repeticiones de ordenadores en el dataset resultante), o muestra de clústeres, donde cada clúster podría estar correspondido por la marca o por intervalos de precio para asegurar que tenemos muestras de todos los precios.

Sobre las columnas podemos obtener un vistazo rápido de las mismas imprimiendo las 5 primeras columnas:

```
laptops_initial.head(5)
```

	Unnamed: 0	Company	Product	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	1	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69
1	2	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94
2	3	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00
3	4	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45
4	5	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60

Aquí encontramos que la primera de ellas que no nos da ninguna información útil para el estudio y ser un simple id ascendente, con lo cual la eliminaremos. Haremos lo mismo con la columna que se refiere al modelo del producto, pues el objetivo en todo momento es realizar una comparación en base a características o marcas, pero no en base al modelo del portátil directamente.

```
# Eliminamos la columna Unnamed: 0
del laptops_initial["Unnamed: 0"]

# Eliminamos la columna Product
del laptops_initial["Product"]

#Imprimimos los datos para comprobar la correcta eliminación
laptops_initial.head(5)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram	Memory	Gpu	OpSys	Weight	Price_euros
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37kg	1339.69
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34kg	898.94
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	1.86kg	575.00
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	1.83kg	2537.45
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37kg	1803.60

Por tanto, tras esta limpieza, las columnas que resultarán del dataset junto con su significado según se proporciona en el repositorio original serán:

- Company --> Company Name
- TypeName --> Laptop Type
- Inches --> Screen Inches
- ScreenResolution --> Screen Resolution
- Cpu --> CPU Model
- Ram --> RAM Characteristics
- Memory --> Memory

- Gpu --> GPU Characteristics
- OpSys --> Operating System
- Weight --> Laptop's Weight
- Price_euros --> Laptop's Price

3. Limpieza de los datos.

3.1 ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?
Tras la carga inicial, se aprecia como no se encuentra ningún valor vacío o cero. Sin embargo, tras el proceso de limpieza que se comentará posteriormente, se encuentran valores nulos en la variable "ScreenResolution_Type", que será creada a partir de la columna ScreenResolution, al no presentarse el tipo explícitamente en la variable inicial.

```
laptops_initial.isnull().sum()
```

```
Company                   0
TypeName                  0
Inches                    0
Ram(GB)                   0
OpSys                     0
Weight(Kg)                0
Price_euros               0
MemorySSD(GB)             0
MemoryHDD(GB)             0
MemoryFlash(GB)           0
MemoryHybrid(GB)          0
CPU_Company               0
CPU_Version               0
CPU_Speed(GHz)            0
GPU_Company               0
GPU_Version               0
ScreenResolution_Type   314
ScreenResolution_Width    0
ScreenResolution_High     0
dtype: int64
```

Para tratar estos nulos, se sustituirán los valores perdidos por una misma constante o etiqueta, en este caso, "Unknown". Se realiza de tal forma ya que falta un gran número de registros (un 33%), y el uso de otras técnicas como la sustitución por la moda o mediana harían que tuviéramos muchísimos datos 'no reales'.

```
# Cambiamos los valores nulos por el literal Unknow
laptops_initial["ScreenResolution_Type"].fillna("Unknown", inplace = True)

# Verificamos que no queda ningún valor nulo
laptops_initial.ScreenResolution_Type.isnull().sum()
```

```
0
```

Asimismo, tampoco tiene sentido aplicar técnicas de sustitución basadas en modelos de predicción, ya que la resolución de la pantalla no es una característica que dependa del resto de variables de la muestra.

Con respecto al resto de variables, tras realizar un pequeño estudio, se aprecia que no hay datos perdidos o que indiquen la pérdida de valor. Si bien es verdad que para sistema operativo encontramos 'No OS', que puede dar lugar a confusión.

```
laptops_initial['OpSys'].unique()
```

```
array(['macOS', 'No OS', 'Windows 10', 'Mac OS X', 'Linux', 'Android',
       'Windows 10 S', 'Chrome OS', 'Windows 7'], dtype=object)
```

Sin embargo, este valor es válido, ya que algunos ordenadores pueden no tener sistema operativo preinstalado y, posiblemente, esto sea algo que abarate el coste del mismo.

3.2 Preparación de datos.

Nos hemos sentido libres de añadir este apartado, por el hecho de que antes de pasar al apartado 4 en donde se analizan más profundamente los datos y se elaboran modelos o incluso del apartado 3.3 consistente en la búsqueda de valores extremos, se ha considera que es necesario un proceso de preprocesado o data cleaning de los mismos.

Para ello, se han centrado los esfuerzos en primer lugar en variables como Ram o Weight, que son variables que se han de tratar como numéricas pero que al tener la unidad de medida en cada uno de sus valores, dificulta su tratamiento. Para ello, se ha comprobado en primer lugar que todos los valores son numéricos y que estaban indicados en las mismas unidades, GB y Kg respectivamente. Para el caso de la RAM, tendríamos el siguiente fragmento de código:

```
#Se realiza un splitado en base a 'GB'
split_ram_value = laptops_initial['Ram'].str.split('GB', 0, expand=True)
#Se comprueba que todos los valores son numericos (no lo serían si hubiera otra medida)
split_ram_value[0].str.isnumeric().unique()
```

```
array([ True])
```

Mientras que, para el caso del peso, es algo más complejo al tener decimales, pero se persigue la misma idea:

```
# Se realiza un splitado en base a 'kg'
split_weight_value = laptops_initial['Weight'].str.split('kg', 0, expand=True)

# Se comprueba que todos los valores son enteros como en el caso anterior o float. Para ello, se define la función is_valid_decimal que
# indicará si un numero es dicimal (True) o no lo es:
def is_valid_decimal(string):
    try:
        float(string)
        return True
    except ValueError:
        return False

# Se crea un objeto serie para almacenar los resultados
is_weight_decimal = pandas.Series([])

# Se recorre la serie con los posibles números resultantes del splitado, y se van añadiendo a la serie creada
for index, value in split_weight_value[0].items():
    is_weight_decimal[index] = is_valid_decimal(value)

# Vemos los diferentes valores que contiene le vector (si todos son true, todos serán valores float por lo que todos serán kg)
is_weight_decimal.unique()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: DeprecationWarning: The default dtype for empty Series will be 'object'

array([ True])
```

Una vez comprobados que todos los valores son numéricos y contienen la misma medida, se ha dejado únicamente el valor numérico como contenido del campo. Además, se ha modificado el nombre de las columnas para indicar las unidades Ram(GB) y Weight(Kg).

```
# Se renombran las columnas, añadiendo los GB a la RAM y los KG al peso
laptops_initial = laptops_initial.rename(columns={'Ram': 'Ram(GB)', 'Weight': 'Weight(Kg)'})

# Se hace el camnio en estas columnas por los números extraídos
laptops_initial["Ram(GB)"] = split_ram_value[0]
laptops_initial["Weight(Kg)"] = split_weight_value[0]

# Se imprime la cabecera
laptops_initial.head(5)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram(GB)	Memory	Gpu	OpSys	Weight(Kg)	Price_euros
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	1.37	1339.69
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	1.34	898.94
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	1.86	575.00
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	1.83	2537.45
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	1.37	1803.60

Finalmente en este proceso, se han dejado las columnas en formato numérico para que de esta manera, puedan ser tratados correctamente en los posteriores análisis que se harán:

```
# Transformamos esta columnas al tipo numérico
laptops_initial["Ram(GB)"] = pandas.to_numeric(laptops_initial["Ram(GB)"])
laptops_initial["Weight(Kg)"] = pandas.to_numeric(laptops_initial["Weight(Kg)"])
laptops_initial["Inches"] = pandas.to_numeric(laptops_initial["Inches"])
laptops_initial["Price_euros"] = pandas.to_numeric(laptops_initial["Price_euros"])
```

Siguiendo en la misma línea, se ha realizado un análisis del contenido de la variable memoria RAM. Se aprecian memorias individuales y también compuestas (híbridas) por varios tamaños (GB y TB) y tipos de tecnologías (SSD, Flash Storage, HDD e Hybrid):

```
# Se realiza un splitado en base tan solo al primer espacio , ya que es lo aparentemente separa la cantidad de memoria y el tipo
split_memory_value = laptops_initial['Memory'].str.split(' ', 1, expand=True)

# Imprimimos las segundas partes del splitado
split_memory_value[1].unique()
```

```
array(['SSD', 'Flash Storage', 'HDD', 'SSD +  1TB HDD',
       'SSD +  256GB SSD', 'SSD +  2TB HDD', 'Hybrid', 'SSD +  500GB HDD',
       'SSD +  512GB SSD', 'Flash Storage +  1TB HDD', 'HDD +  1TB HDD',
       'SSD +  1.0TB Hybrid'], dtype=object)
```

Para tratar este hecho, se ha hecho una función bastante larga (que no se adjunta imagen al ser demasiado larga pero se puede consultar en el apartado código si se desea), transformado todas las medidas a GB y se han creado cuatro nuevas columnas cuyo contenido será numérico: MemorySSD(GB), MemoryHDD(GB), MemoryFlash(GB), MemoryHybrid(GB). Se consigue por tanto, que el dataSet en este punto presente la siguiente estructura:

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram(GB)	Gpu	OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	128.0	0.0	0.0	0.0
1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	Intel HD Graphics 6000	macOS	1.34	898.94	0.0	0.0	128.0	0.0
2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	Intel HD Graphics 620	No OS	1.86	575.00	256.0	0.0	0.0	0.0
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	AMD Radeon Pro 455	macOS	1.83	2537.45	512.0	0.0	0.0	0.0
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	Intel Iris Plus Graphics 650	macOS	1.37	1803.60	256.0	0.0	0.0	0.0

Para el caso de la CPU, se ha identificado que el patrón que sigue esta columna es el siguiente: en primer lugar se da el fabricante, en segundo lugar la versión que se proporciona y por último, la velocidad de reloj de la misma. El proceso seguido para conseguir su separación, ha sido en primer lugar, separar el string en base al primer espacio para conseguir la marca:

```
# Se realiza un splitado en base al primer espacio, dejando en la primera parte la supuesta marca, mientras que en la segunda el resto
split_cpu_company_value = laptops_initial['Cpu'].str.split(' ', 1, expand=True)

# Almacenamos las marcas de cpu
company_cpu = split_cpu_company_value[0]

# Comprobamos que efectivamente todas ellas corresponden a marcas
company_cpu.unique()
```

```
array(['Intel', 'AMD', 'Samsung'], dtype=object)
```

En segundo lugar, con la segunda parte de la separación anterior, se separará la versión y la velocidad realizando en este caso el splitado en el último espacio. Así pues, el proceso para obtener la versión será:

```
# Se realiza un splitado de la segunda parte, en donde el mismo se realizará en base al último espacio, al ser la estructura VERSION + ' ' + VELOCIDAD
split_cpu_version_and_speed_value = split_cpu_company_value[1].str.rsplit(' ', 1, expand=True)

# Sacamos las versiones, que estarán en la primera parte del splitado
version_cpu=split_cpu_version_and_speed_value[0]

# Comprobamos que todas ellas son versiones
version_cpu.unique()
```

```
array(['Core i5', 'Core i5 7200U', 'Core i7', 'A9-Series 9420',
       'Core i7 8550U', 'Core i5 8250U', 'Core i3 6006U', 'Core M m3',
       'Core i7 7500U', 'Core i3 7100U', 'Atom x5-Z8350',
       'Core i5 7300HQ', 'E-Series E2-9000e', 'Core i7 8650U',
       'Atom x5-Z8300', 'E-Series E2-6110', 'A6-Series 9220',
       'Celeron Dual Core N3350', 'Core i3 7130U', 'Core i7 7700HQ',
       'Ryzen 1700', 'Pentium Quad Core N4200', 'Atom x5-Z8550',
       'Celeron Dual Core N3060', 'FX 9830P', 'Core i7 7560U',
       'E-Series 6110', 'Core i5 6200U', 'Core M 6Y75', 'Core i5 7500U',
       'Core i7 6920HQ', 'Core i5 7Y54', 'Core i7 7820HK',
       'Xeon E3-1505M V6', 'Core i7 6500U', 'E-Series 9000e',
       'A10-Series A10-9620P', 'A6-Series A6-9220', 'Core i7 6600U',
       'Celeron Dual Core 3205U', 'Core i7 7820HQ', 'A10-Series 9600P',
       'Core i7 7600U', 'A8-Series 7410', 'Celeron Dual Core 3855U',
       'Pentium Quad Core N3710', 'A12-Series 9720P', 'Core i5 7300U',
       'Celeron Quad Core N3450', 'Core i5 6440HQ', 'Core i7 6820HQ',
       'Ryzen 1600', 'Core i7 7Y75', 'Core i5 7440HQ', 'Core i7 7660U',
       'Core M m3-7Y30', 'Core i5 7Y57', 'Core i7 6700HQ',
       'Core i3 6100U', 'A10-Series 9620P', 'E-Series 7110',
       'A9-Series A9-9420', 'Core i7 6820HK', 'Core M 7Y30',
       'Xeon E3-1535M v6', 'Celeron Quad Core N3160', 'Core i5 6300U',
       'E-Series E2-9000', 'Celeron Dual Core N3050', 'Core M M3-6Y30',
       'Core i5 6300HQ', 'A6-Series 7310', 'Atom Z8350',
       'Xeon E3-1535M v5', 'Core i5 6260U', 'Pentium Dual Core N4200',
       'Celeron Quad Core N3710', 'Core M', 'A12-Series 9700P',
       'Pentium Dual Core 4405U', 'A4-Series 7210', 'Core i7 6560U',
       'Core M m7-6Y75', 'FX 8800P', 'Core M M7-6Y75', 'Atom X5-Z8350',
       'Pentium Dual Core 4405Y', 'Pentium Quad Core N3700',
       'Core M 6Y54', 'Cortex A72&A53', 'E-Series 9000', 'Core M 6Y30',
       'A9-Series 9410'], dtype=object)
```

Y de la parte resultante corresponderá a la velocidad:

```
# Sacamos las velocidades, que estarán en la seguna parte del splitado anterior
speed_cpu=split_cpu_version_and_speed_value[1]
speed_cpu.unique()
```

```
array(['2.3GHz', '1.8GHz', '2.5GHz', '2.7GHz', '3.1GHz', '3GHz', '2.2GHz',
       '1.6GHz', '2GHz', '2.8GHz', '1.2GHz', '2.9GHz', '2.4GHz',
       '1.44GHz', '1.5GHz', '1.9GHz', '1.1GHz', '2.0GHz', '1.3GHz',
       '2.6GHz', '3.6GHz', '1.60GHz', '3.2GHz', '1.0GHz', '2.1GHz',
       '0.9GHz', '1.92GHz', '2.50GHz', '2.70GHz'], dtype=object)
```

Que como se aprecia, presenta todas las unidades de medida en GHz, por lo que se realizará la separación de la medida como en el apartado anterior. Por tanto, realizando este una separación de estos 3 campos, resultan las columnas CPU_Company, CPU_Version, CPU_Speed(GHz).

```
# Introducimos las nuevas columnas
laptops_initial["CPU_Company"] = company_cpu
laptops_initial["CPU_Version"] = version_cpu
laptops_initial["CPU_Speed(GHz)"] = speed_cpu_GHz

# La transformamos a numéricas
laptops_initial["CPU_Speed(GHz)"] = pandas.to_numeric(laptops_initial["CPU_Speed(GHz)"])

# Eliminamos la columna inicial que tenía los datos de la CPU sin limpiar
del laptops_initial["Cpu"]

# Verificamos la correcta creación
laptops_initial.head(5)
```

	Company	TypeName	Inches	ScreenResolution	Ram(GB)	Gpu	OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Company	CPU_Version	CPU_Speed(GHz)
0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	8	Intel Iris Plus Graphics 640	macOS	1.37	1339.69	128.0	0.0	0.0	0.0	Intel	Core i5	2.3
1	Apple	Ultrabook	13.3	1440x900	8	Intel HD Graphics 6000	macOS	1.34	898.94	0.0	0.0	128.0	0.0	Intel	Core i5	1.8
2	HP	Notebook	15.6	Full HD 1920x1080	8	Intel HD Graphics 620	No OS	1.86	575.00	256.0	0.0	0.0	0.0	Intel	Core i5 7200U	2.5
3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	16	AMD Radeon Pro 455	macOS	1.83	2537.45	512.0	0.0	0.0	0.0	Intel	Core i7	2.7
4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	8	Intel Iris Plus Graphics 650	macOS	1.37	1803.60	256.0	0.0	0.0	0.0	Intel	Core i5	3.1

En la misma línea con la GPU, resulta claro ver cómo esta presenta la estructura de fabricante de la GPU seguido por la versión, por lo que da lugar a las nuevas columnas GPU_Company, GPU_Version. Extrapolando el método anterior, y para no sobrecargar el informe de código parecido, se muestran las columnas añadidas.

GPU_Company	GPU_Version
Intel	Iris Plus Graphics 640
Intel	HD Graphics 6000
Intel	HD Graphics 620
AMD	Radeon Pro 455
Intel	Iris Plus Graphics 650

Finalmente, con la resolución de pantalla se observa el siguiente patrón: en primer lugar, en algunas ocasiones, se describe cualitativamente la resolución de la pantalla (p.ej.: Full-HD), seguido por el tamaño de pantalla en píxeles con el patrón alto x ancho. Por tanto, se procederá a fraccionar esta variable en estos 3 atributos nuevos. Para ello, en primer lugar se extraerá el tamaño de la pantalla mediante el uso de una expresión regular:

```
# Limpieza columna ScreenResolution (se dejan los valores que indican únicamente la resolución en píxeles)
pixels_ScreenResolution = laptops_initial.ScreenResolution.str.extract("(\d+x\d+)", expand=False)

# Se comprueba que no hay nulos (que para todos los valores se ha conseguido obtener los píxeles)
pixels_ScreenResolution.isnull().sum()
```

```
0
```

Para, seguidamente, mediante la función de splitado, dividir el valor en alto y ancho:

```
# Se realiza un splitado en base a la x
split_pixels_value = pixels_ScreenResolution.str.split('x', 0, expand=True)

# Se separan en ancho (width) y alto (high)
split_pixels_width = split_pixels_value[0]
split_pixels_high = split_pixels_value[1]
```

En segundo lugar, se procede a extraer el tipo de resolución. Para ello, se hará un splitado en base al primer espacio. En este caso, como no todos los registros poseen un valor, si en la posición 0 de este splitado contamos con la dimensión de pantalla, significará que no tiene tipo indicado, por lo que en este caso dejaremos el tipo a nulo. Esto, traducido a código será:

```
# Se realiza un splitado en base al último espacio, ya que si tiene tipo, en la primera variable de este splitado estará el mismo
# y si no lo tiene, estarán los píxeles sacados anteriormente
split_pixels_by_last_space = laptops_initial.ScreenResolution.str.rsplit(' ', 1, expand=True)

# Se crea un objeto serie para almacenar los tipos
type_pixels = pandas.Series([])

# Comprobamos uno a uno si son igualas a los píxeles (no tiene tipo) o si no lo son (tiene el tipo)
for index, value in split_pixels_by_last_space[0].items():
  if (split_pixels_by_last_space[0][index] == pixels_ScreenResolution[index]):
    type_pixels[index]=np.NaN
  else:
    type_pixels[index]=split_pixels_by_last_space[0][index]
```

Será en esta variable donde aparezcan, como era de esperar, los valores nulos que se han comentado en el apartado anterior. Una vez tenemos los datos preparados, se crearán las 3 nuevas columnas en el dataset ScreenResolution_Type, ScreenResolution_Width, ScreenResolution_High, y se eliminarán las columnas de las que se han partido.

```
# Introducimos las nuevas columnas
laptops_initial["ScreenResolution_Type"] = type_pixels
laptops_initial["ScreenResolution_Width"] = split_pixels_width
laptops_initial["ScreenResolution_High"] = split_pixels_high

# La transformamos a numéricas las dos últimas
laptops_initial["ScreenResolution_Width"] = pandas.to_numeric(laptops_initial["ScreenResolution_Width"])
laptops_initial["ScreenResolution_High"] = pandas.to_numeric(laptops_initial["ScreenResolution_High"])

# Eliminamos la columna inicial que tenia los datos de ScreenResolution sin limpiar
del laptops_initial["ScreenResolution"]

#Verificamos la correcta creación
laptops_initial.head(5)
```

OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Company	CPU_Version	CPU_Speed(GHz)	GPU_Company	GPU_Version	ScreenResolution_Type	ScreenResolution_Width	ScreenResolution_High
macOS	1.37	1339.69	128.0	0.0	0.0	0.0	Intel	Core i5	2.3	Intel	Iris Plus Graphics 640	IPS Panel Retina Display	2560	1600
macOS	1.34	898.94	0.0	0.0	128.0	0.0	Intel	Core i5	1.8	Intel	HD Graphics 6000	NaN	1440	900
No OS	1.86	575.00	256.0	0.0	0.0	0.0	Intel	Core i5 7200U	2.5	Intel	HD Graphics 620	Full HD	1920	1080
macOS	1.83	2537.45	512.0	0.0	0.0	0.0	Intel	Core i7	2.7	AMD	Radeon Pro 455	IPS Panel Retina Display	2880	1800
macOS	1.37	1803.60	256.0	0.0	0.0	0.0	Intel	Core i5	3.1	Intel	Iris Plus Graphics 650	IPS Panel Retina Display	2560	1600

Una vez se ha llegado a este punto, ya se poseen datos más preparados para poder realizar análisis posteriores, y muchas variables numéricas listas para trabajar con ellas. De forma previa a realizar este procesamiento, muchas de las variables estaban en formato string ya que incluían la unidad de medida o información adicional en el propio valor. Esto nos impedía su uso para hacer buenos análisis o modelos al ser muchas de ellas variables categóricas y no numéricas.

3.3 Identificación y tratamiento de valores extremos.

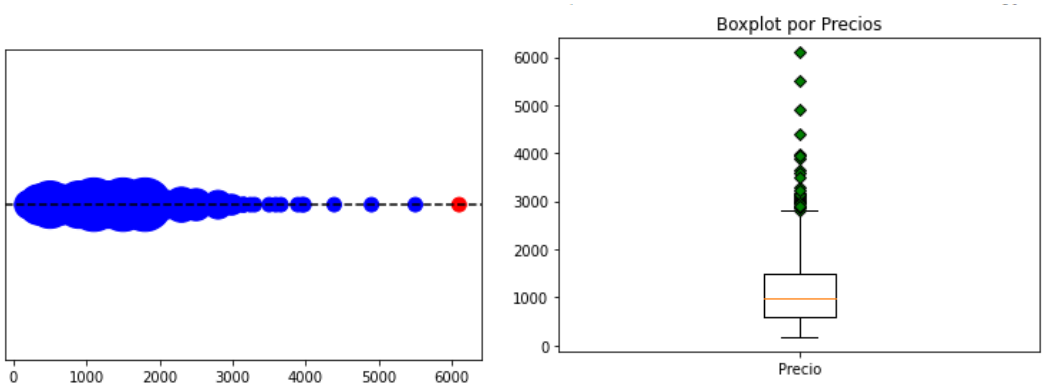
Se analiza en primer lugar la variable precio, variable numérica más importante de nuestro estudio. Se procede en primer lugar a realizar un pequeño resumen estadístico de esta variable:

```
# Percentiles y resumen estadístico de la columna de precios
laptops_initial["Price_euros"].describe()
```

```
count    1303.000000
mean     1123.686992
std       699.009043
min       174.000000
25%       599.000000
50%       977.000000
75%      1487.880000
max      6099.000000
```

Se aprecia como aparentemente, aunque el percentil 50 está en 977 y su 75 en 1487, la media es de 1123, lo que es indicio de que, tras el percentil 75 encontraremos algún valor más elevado, hasta llegar al máximo de 6099.

Gracias a las representaciones realizadas en el Notebook, se aprecia como hay una gran concentración de datos en la zona en torno a 1000 euros, y que va disminuyendo hasta llegar hasta los 3000, donde los precios empiezan a encontrarse más distantes entre sí hasta llegar a los 6000, dato que podríamos plantearnos como punto alejado o outlier. Las dos gráficas más representativas de este hecho, serán las que se presentan a continuación:



Sin embargo, al analizar estos puntos 'alejados' para estudiar si realmente son outliers que hay que eliminar o si son datos posibles, se aprecia que la gran mayoría de portátiles de precios elevados están catalogados como ordenadores Gaming.

```
# Ordenar los ordenadores por precio, e imprimir las columnas más caras
laptops_initial.sort_values('Price_euros').tail(10)
```

	Company	TypeName	Inches	Ram(GB)	OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Company	CPU_Version	CPU_Speed(GHz)	GPU_Company	GPU_Version	S
1231	Razer	Gaming	14.0	16	Windows 10	1.95	3499.0	1024.0	0.0	0.0	0.0	Intel	Core i7 7700HQ	2.8	Nvidia	GeForce GTX 1060	
780	Dell	Gaming	17.3	32	Windows 10	4.42	3588.8	1024.0	1024.0	0.0	0.0	Intel	Core i7 7700HQ	2.8	Nvidia	GeForce GTX 1070M	
723	Dell	Gaming	17.3	32	Windows 10	4.36	3659.4	1024.0	1024.0	0.0	0.0	Intel	Core i7 7700HQ	2.8	Nvidia	GeForce GTX 1070	
238	Asus	Gaming	17.3	32	Windows 10	4.70	3890.0	512.0	1024.0	0.0	0.0	Intel	Core i7 7820HK	2.9	Nvidia	GeForce GTX 1080	
1136	HP	Workstation	17.3	8	Windows 7	3.00	3949.4	256.0	0.0	0.0	0.0	Intel	Core i7 6700HQ	2.6	Nvidia	Quadro M3000M	
1066	Asus	Gaming	17.3	64	Windows 10	3.58	3975.0	1024.0	0.0	0.0	0.0	Intel	Core i7 6820HK	2.7	Nvidia	GeForce GTX 980	
749	HP	Workstation	17.3	16	Windows 7	3.00	4389.0	256.0	0.0	0.0	0.0	Intel	Xeon E3-1535M v5	2.9	Nvidia	Quadro M2000M	
610	Lenovo	Notebook	15.6	32	Windows 10	2.50	4899.0	1024.0	0.0	0.0	0.0	Intel	Xeon E3-1535M v6	3.1	Nvidia	Quadro M2200M	
830	Razer	Gaming	17.3	32	Windows 10	3.49	5499.0	512.0	0.0	0.0	0.0	Intel	Core i7 7820HK	2.9	Nvidia	GeForce GTX 1080	
196	Razer	Gaming	17.3	32	Windows 10	3.49	6099.0	1024.0	0.0	0.0	0.0	Intel	Core i7 7820HK	2.9	Nvidia	GeForce GTX 1080	

Este tipo de ordenadores destacan por necesitar de una potencia gráfica y de procesamiento muy superior a la media, hecho que también hace incrementar su precio. Este hecho es fácilmente visible si realizamos un estudio estadístico de los ordenadores del tipo Gaming:

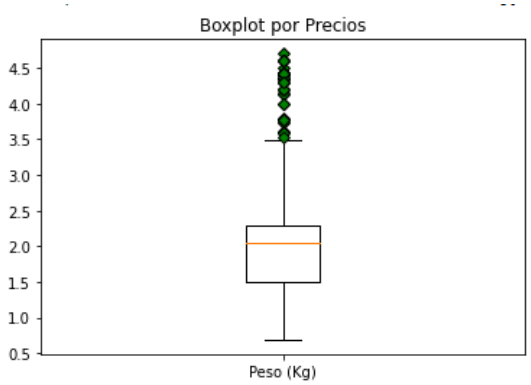
```
laptops_initial[laptops_initial['TypeName']=='Gaming'].describe()
```

	Inches	Ram(GB)	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Speed(GHz)	ScreenResolution_Width	ScreenResolution_High
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0	205.000000	205.000000	205.000000	205.000000
mean	16.345854	14.048780	2.949761	1731.380634	236.019512	879.141463	0.0	19.980488	2.722439	2048.000000	1152.000000
std	0.927174	7.234013	0.759205	814.174430	189.474608	423.528276	0.0	141.982875	0.142057	471.666216	265.312247
min	14.000000	4.000000	1.600000	699.000000	0.000000	0.000000	0.0	0.000000	2.100000	1920.000000	1080.000000
25%	15.600000	8.000000	2.430000	1169.000000	128.000000	1024.000000	0.0	0.000000	2.600000	1920.000000	1080.000000
50%	15.600000	16.000000	2.700000	1492.800000	256.000000	1024.000000	0.0	0.000000	2.800000	1920.000000	1080.000000
75%	17.300000	16.000000	3.350000	2199.000000	256.000000	1024.000000	0.0	0.000000	2.800000	1920.000000	1080.000000
max	18.400000	64.000000	4.700000	6099.000000	1024.000000	2048.000000	0.0	1024.000000	3.200000	3840.000000	2160.000000

Tras una rápida consulta de precios en el mercado de ordenadores Gaming en las webs de los principales vendedores de elementos electrónicos, confirmamos que no es descabellado que se

den dichos precios en este sector específico, con lo que hemos considerado que han de mantenerse en el dataset.

Además, realizando el mismo estudio para el caso del peso se aprecia que ocurre algo similar, donde algunos ordenadores se alejan bastante de la mayoría por la parte superior.



Si analizamos la tipología de son estos ordenadores, corresponden también a ordenadores Gaming, conocidos por unas pantallas más grandes por lo general y que requieren de un cuerpo mayor para disponer de una mayor capacidad de disipación del calor, hecho que también hace aumentar su peso. En base a estos análisis, no hemos considerado necesario considerar ningún valor del peso como outlier.

4. Análisis de los datos.

4.0 Resumen del dataSet

En un primer lugar, se realizará un estudio de las variables numéricas. Para ello, mostraremos sus valores medios, desviaciones, medias al igual que sus percentiles.

```
laptops_initial.describe()
```

	Inches	Ram(GB)	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Speed(GHz)	ScreenResolution_Width	ScreenResolution_High
count	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000	1303.000000
mean	15.017191	8.382195	2.038734	1123.686992	184.027629	422.477360	4.555641	9.034536	2.298772	1894.784344	1070.830391
std	1.426304	5.084665	0.665475	699.009043	188.268689	528.006298	30.274090	94.738890	0.506340	494.641028	284.519410
min	10.100000	2.000000	0.690000	174.000000	0.000000	0.000000	0.000000	0.000000	0.900000	1366.000000	768.000000
25%	14.000000	4.000000	1.500000	599.000000	0.000000	0.000000	0.000000	0.000000	2.000000	1600.000000	900.000000
50%	15.600000	8.000000	2.040000	977.000000	256.000000	0.000000	0.000000	0.000000	2.500000	1920.000000	1080.000000
75%	15.600000	8.000000	2.300000	1487.880000	256.000000	1024.000000	0.000000	0.000000	2.700000	1920.000000	1080.000000
max	18.400000	64.000000	4.700000	6099.000000	1024.000000	2048.000000	512.000000	1024.000000	3.600000	3840.000000	2160.000000

Con lo que se ha visto hasta el momento, de esta tabla resumen lo primero que llama la atención es que se puede confirmar la afirmación mostrada anteriormente. Todos los valores presentan medias y percentiles menores que los que se han dado en los ordenadores gaming, menos en la memoria Flash, ya que los ordenadores de estas características priorizan los otros tipos de memoria, en especial la SSD. Por tanto, se aprecia como lo descrito anteriormente es cierto.

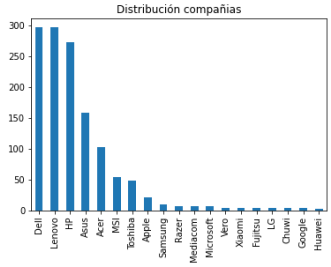
Por otro lado, si ignoramos las variables de memoria, no se producen desviaciones estándar muy elevadas, excepto en el precio (que tiene sentido al tener ordenadores de diversas marcas, características) y en la resolución de pantalla, que tiene sentido también ya que son números más elevados y es lógico que de un notebook a un ordenador gamer se encuentre esta diferencia en la pantalla.

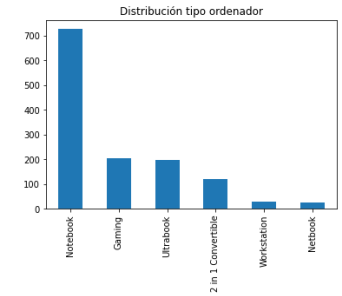
En las memorias vemos como las desviaciones son mayores (teniendo en cuenta su unidad de medida, medias y límites), y esto es así por el hecho de que algunos ordenadores poseen un tipo de memoria y otros dos. Este hecho, por la decisión que se ha tomado antes, hace que si no se dispone de un tipo de memoria, se ponga el valor de 0 para ese caso, por lo que en una columna

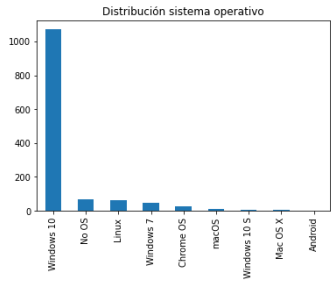
se pueden apreciar datos algo diversos por el hecho de que un ordenador tenga una memoria de un tipo pero de otra no. Debido a este hecho, y a que estas columnas se podrían llegar a considerar discretas (las memorias pueden ser de unos valores en concreto), se harán una representación gráfica de las mismas como con las categóricas.

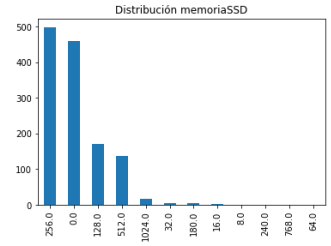
Para las variables categóricas, siempre que no nos encontremos con una cantidad de posibles valores desorbitada, se realizarán gráficos de barra que permiten ver de una forma más visual el tipo de datos con los que se trabaja. En caso de que se encuentren muchas más categorías, se realizará una tabla de frecuencia relativa a esa variable, hecho que ocurre tanto para las versiones de GPU como de CPU como para los tipo de resolución .

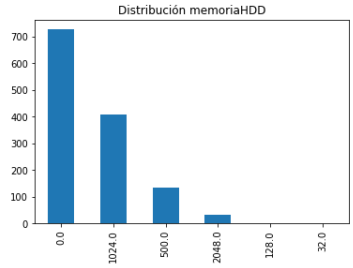
Por tanto, el conjunto de gráficas resultantes serán:

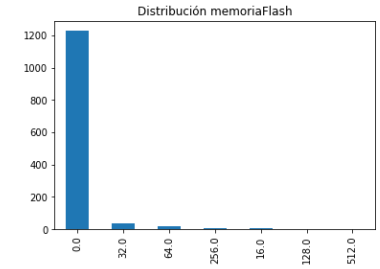


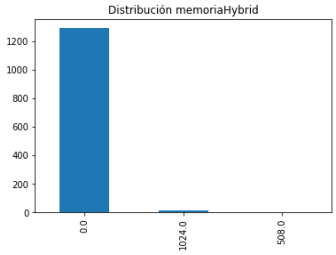


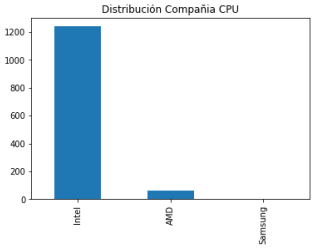


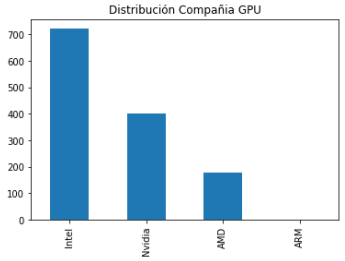


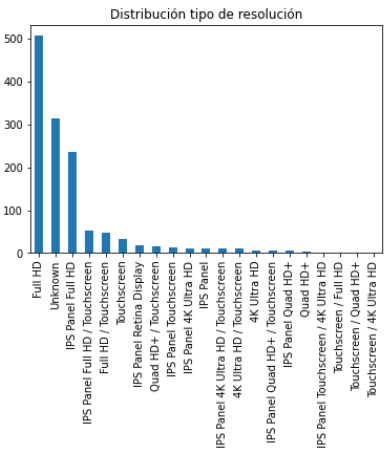












De estas se pueden extraer de primera mano varias conclusiones, como son:

- El dataset cuenta con gran cantidad de marcas de ordenadores diferentes, pero se observa una predominancia de Dell, Lenovo y HP. Posiblemente sea porque son los que más modelos tengan.

- Los ordenadores de tipo Notebok (al ser los más comunes en el día a día), son los que más porción ocupan del dataSet.

- El sistema operativo por excelencia es Windows 10, seguido muy de lejos por ordenadores libres y a continuación Linux.

- En el caso de contar memoria SSD (que más de la mitad cuentan con este tipo de memoria), la misma es de 256 GB. En el caso de tener HDD, es 1024GB, aunque predominan los ordenadores sin este tipo de memoria (habitualmente es porque los ordenadores, como se ha podido ver en la limpieza, es más habitualmente que posean memoria SSD que HDD al ser esta más rápida). Para los casos de flash e Hybrid, no la gran mayoría no disponen de esta memoria.

- Se puede afirmar, que Intel es la marca líder en el mercado en "darle vida al ordanador". Esta marca es la líder en tanto en GPU como en CPU. En GPU, es seguida por Nvidia, sistema operativo presente en la gran mayoría de ordenadores Gaming como se puede apreciar en la tabla del apartado anterior.

Para las variables que no se han mostrado gráficamente, se muestran las tablas de frecuencia para las mismas (en algunas al tener tantos valores solo se mostrarán los máximos y mínimos).

```
Core i5 7200U     14.811972
Core i7 7700HQ    11.281658
Core i7 7500U     10.360706
Core i3 6006U      6.216424
Core i7 8550U      5.602456
                     ...
Atom x5-Z8300      0.076746
A4-Series 7210     0.076746
Core i5 7500U      0.076746
E-Series 9000      0.076746
Core i7 6920HQ     0.076746
Name: CPU_Version, Length: 93, dtype: float64
```

```
HD Graphics 620     21.565618
HD Graphics 520     14.198005
UHD Graphics 620     5.218726
GeForce GTX 1050     5.065234
GeForce GTX 1060     3.683807
                       ...
Quadro 3000M         0.076746
Radeon Pro 555       0.076746
GeForce GTX1060      0.076746
HD Graphics 540      0.076746
GTX 980 SLI          0.076746
Name: GPU_Version, Length: 110, dtype: float64
```

```
Full HD                                 38.910207
Unknown                                 24.098235
IPS Panel Full HD                       18.035303
IPS Panel Full HD / Touchscreen          4.067536
Full HD / Touchscreen                    3.607061
Touchscreen                              2.455871
IPS Panel Retina Display                 1.304682
Quad HD+ / Touchscreen                   1.151190
IPS Panel Touchscreen                    0.997698
IPS Panel 4K Ultra HD                    0.920952
IPS Panel                                0.844206
IPS Panel 4K Ultra HD / Touchscreen      0.844206
4K Ultra HD / Touchscreen                0.767460
4K Ultra HD                              0.537222
IPS Panel Quad HD+ / Touchscreen         0.460476
IPS Panel Quad HD+                       0.383730
Quad HD+                                 0.230238
IPS Panel Touchscreen / 4K Ultra HD      0.153492
Touchscreen / Full HD                    0.076746
Touchscreen / Quad HD+                   0.076746
Touchscreen / 4K Ultra HD                0.076746
Name: ScreenResolution_Type, dtype: float64
```

Respecto a estas tablas de frecuencias, se aprecia como ha gran cantidad de versiones tanto de CPU como de GPU. Estos son principalmente el core i5 7200U, que se lleva casi el 15% de ordenadores (este hecho puede ser debido a que es una versión de CPU de calidad baja válida para trabajos diarios), seguidos por CPU de calidad algo mayores al hablar de core i7. Para el cado de las GPU, predominan los HD Graphics 620 y 520, llevándose estas el 35% de los ordenadores. Finalmente, con respecto al tipo de resolución predomina la HD en su mayor parte, aunque en un 25% de las ocasiones, no se ha indicado el tipo de la misma como se ha analizado en el apartado de limpieza.

4.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

A lo largo de este apartado y los siguientes, se han realizado análisis tanto cuantitativos como cualitativos que permiten ver qué variables son las mejores candidatas para formar parte de un

modelo o estudio que permita responder a las dudas/objetivos que habíamos planteado al inicio, es decir, que el comprador:

- Pueda conocer si es verdad que algunas marcas, como Apple, tienen un precio algo superior al resto de las marcas.
- Pueda conocer qué características son las más influyentes en el precio para en base a estas, poder centrarse en lo que realmente necesita para incrementar o decrementar su presupuesto.
- Pueda hacerse una idea del presupuesto aproximado que tendrá el ordenador que desea en base a las características técnicas deseadas.

Para responder a la primera de las cuestiones, se ha planteado un contraste de medias que nos indicará si los productos de Apple son significativamente más caros, de media, que el resto de los productos de otras marcas.

Para el segundo objetivo, se ha evaluado la correlación de las variables numéricas respecto a la variable dependiente del precio, para identificar cuáles de ellas tienen más influencia en la variabilidad del precio.

Para el tercero, se han generado modelos de regresión lineal que predicen el precio esperado a partir de un conjunto de características seleccionadas a lo largo del objetivo anterior.

4.2 Comprobación de la normalidad y homogeneidad de la varianza.
El estudio de la normalidad y homogeneidad de la varianza se incluye en el apartado "Contraste de muestras", correspondiente a la pregunta 4.3.

4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

Contraste de hipótesis

Se plantea si los precios de los portátiles de Apple tienen un precio superior, de media, al resto de productos de otras compañías. Para poder verificar estadísticamente dicha afirmación, se plantea un contraste de hipótesis como el siguiente:

$$H_0: \mu_{Apple} = \mu_{otros}$$

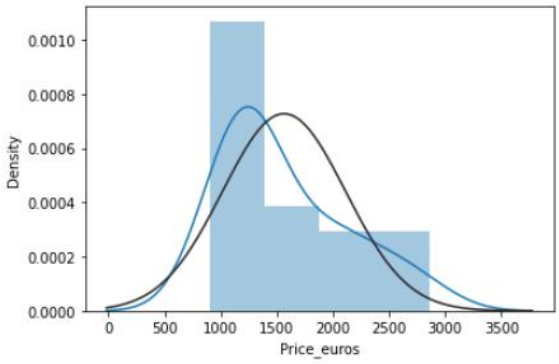
$$H_1: \mu_{Apple} > \mu_{otros}$$

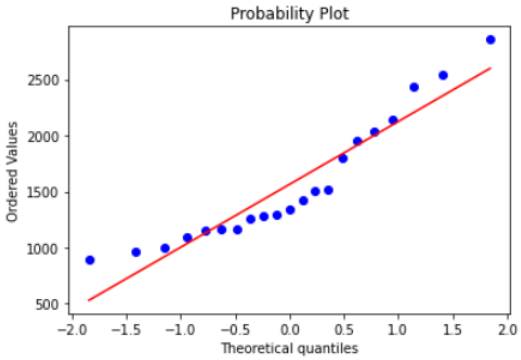
Para llevar a cabo el contraste de muestras, separaremos en dos el dataset de tal forma que contengan respectivamente los precios de los productos de Apple y del resto de marcas.

```
apple_prices = laptops_initial[laptops_initial["Company"] == "Apple"]["Price_euros"]
other_prices = laptops_initial[laptops_initial["Company"] != "Apple"]["Price_euros"]
```

Una vez separadas las muestras, estudiamos la normalidad de cada una de ellas:

```
# Histograma y gráfico de probabilidad normal de los precios de Apple:
sns.distplot(apple_prices, fit = norm);
fig = plt.figure()
res = stats.probplot(apple_prices, plot = plt)
```



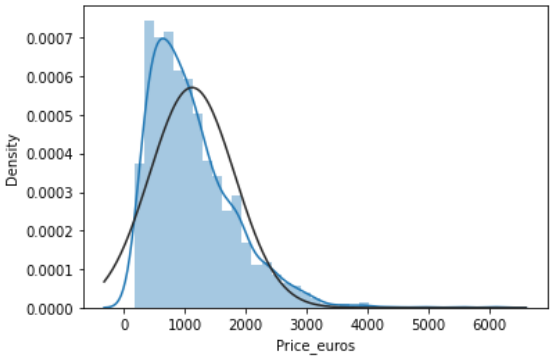


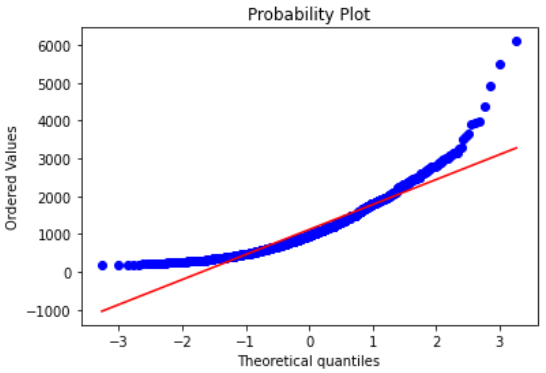

```
# Prueba de Shapiro-Wilk
stat, p = shapiro(apple_prices)
print(f"Stat: {round(stat,3)}")
print(f"p-value: {round(p,3)}")
```

```
Stat: 0.893
p-value: 0.026
```

En base a los resultados, dado que p-value es menor que el valor de significancia 0.05, podemos descartar la hipótesis nula de normalidad, por lo que no podemos considerar que la distribución que siguen los precios de apple se ajuste a una normal. Asimismo, en el caso del subset de Apple, no sería posible tampoco aplicar el teorema del límite central al no superar las 30 muestras.

```
# Histograma y gráfico de probabilidad normal de los precios del resto de marcas:
sns.distplot(other_prices, fit = norm);
fig = plt.figure()
res = stats.probplot(other_prices, plot = plt)
```






```
# Prueba de Shapiro-Wilk
stat, p = shapiro(other_prices)
print(f"Stat: {round(stat,3)}")
print(f"p-value: {round(p,3)}")
```

```
Stat: 0.891
p-value: 0.0
```

En base a los resultados, dado que p-value es menor que el valor de significancia 0.05, podemos descartar la hipótesis nula de normalidad, por lo que no podemos considerar que la distribución que siguen los precios de productos distintos a apple se ajuste a una normal.

En este sentido, para poder confirmar si podemos asumir homocedasticidad (igualdad de varianzas entre muestras) debemos aplicar el test de Fligner-Killeen (no paramétrico) al no poder suponer normalidad:

```
# Fligner-Killeen test
fligner_test = stats.fligner(apple_prices, other_prices, center='median')
fligner_test
```

```
FlignerResult(statistic=0.9821213321141975, pvalue=0.32167564402079707)
```

A raíz de los resultados (p value >> 0.05) no podemos descartar la hipótesis nula, por lo que se confirma la homocedasticidad. No obstante, debido a que no hemos podido afirmar que sigan distribuciones normales, no podremos aplicar un contraste de muestras paramétrico (t-Student), si no que tendremos que aplicar uno no paramétrico (Mann-Whitney):

```
# Mann-Whitney test
mannwhitneyu_test = stats.mannwhitneyu(apple_prices, other_prices, alternative="greater")
mannwhitneyu_test
```

```
MannwhitneyuResult(statistic=19689.0, pvalue=0.00013581790526573893)
```

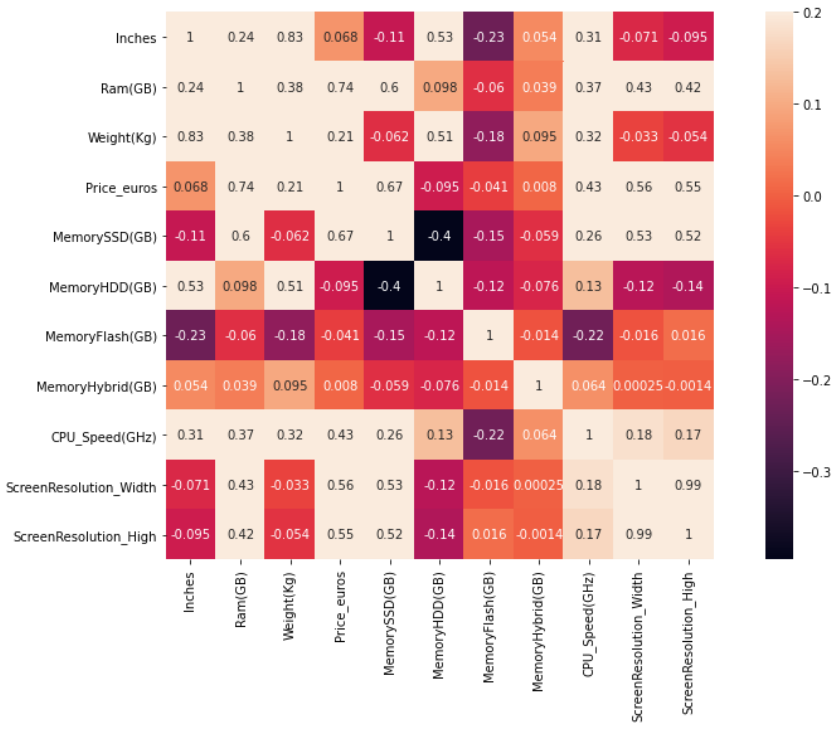
A partir de los resultados obtenidos del test de Mann-Whitney (p-value << 0.05), podemos rechazar la hipótesis nula en favor de la hipótesis alternativa que, en este caso, correspondía con que el precio de los productos de Apple es superior de media que para el resto de marcas.

Estudio de variables numéricas

En este apartado se pretende estudiar qué variables numéricas son más influyentes a la hora de determinar el valor que toma la variable dependiente Precio_euros.

Para ello, realizamos una matriz de correlación, que nos podrá indicar con qué variables numéricas tiene más relación el valor del precio:

```
# Matriz de correlación:
corrmat = laptops_initial.corr()
f, ax = plt.subplots(figsize=(15, 8))
sns.heatmap(corrmat, annot=True, vmax=.2, square=True);
```



Tal y como se puede apreciar en el mapa de calor, la variable numérica con la que tiene más correlación el precio es con la cantidad de RAM que tenga el dispositivo, seguido de la cantidad de memoria SSD, la resolución de pantalla y la CPU que monte el dispositivo. Todas ellas, serán candidatas para formar parte del modelo de predicción del precio de un dispositivo.

Al contrario, las variables que menos influyen en la variabilidad de precio son el resto de tipos de almacenamiento (HDD, Flash e Híbrido), así como las pulgadas del portátil. El peso, pese a que influye en la valoración del portátil, lo hace de forma muy débil.

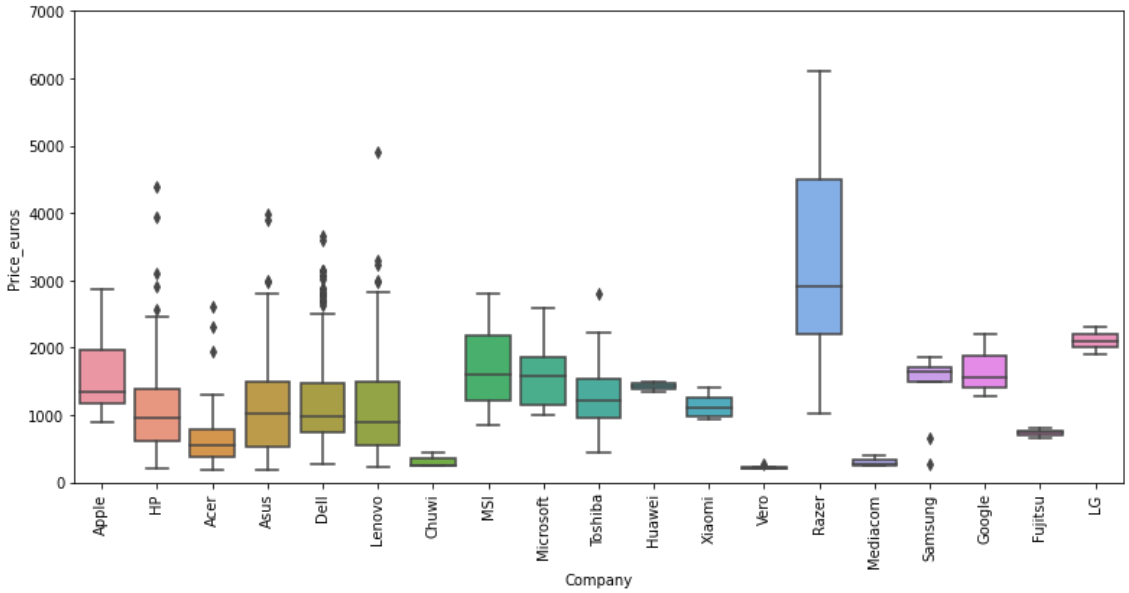
Dado que se identifica que las resoluciones de pantalla a lo ancho y alto están completamente correlacionadas entre sí, se elimina la columna ScreenResolution_High para evitar redundancias.

```
# Eliminación de la columna redundante ScreenResolution_High
del laptops_initial['ScreenResolution_High']
```

Estudio de variables categóricas

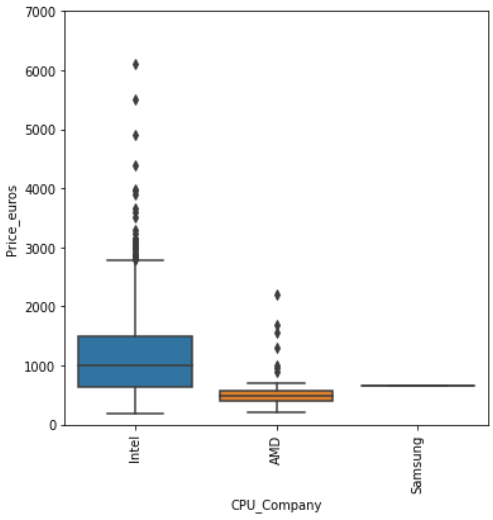
A continuación, vamos a estudiar con más detalle algunas de las variables categóricas que pensamos pueden afectar más al rendimiento del modelo, para identificar visualmente si vemos conveniente que también formen parte del mismo:

```
# Boxplot Company/Price_euros:
var = 'Company'
data = pandas.concat([laptops_initial['Price_euros'], laptops_initial[var]], axis=1)
f, ax = plt.subplots(figsize=(13, 6))
fig = sns.boxplot(x=var, y="Price_euros", data = data)
fig.axis(ymin=0, ymax=7000);
plt.xticks(rotation=90);
```



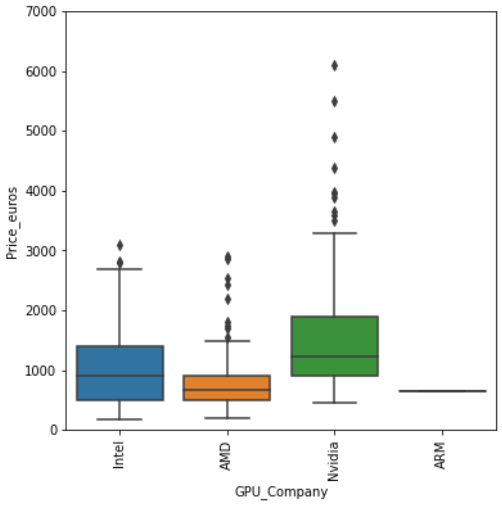
Respecto a las marcas, si bien hay algunas con rangos claramente superiores o inferiores, la mayoría de ellas mantienen una media de precios estable entorno a los 1000-2000€, rango que por otra parte se ha comprobado anteriormente como la media del dataset. En este sentido, no haremos uso de esta variable para el modelo.

```
# Boxplot CPU_Company/Price_euros:
var = 'CPU_Company'
data = pandas.concat([laptops_initial['Price_euros'], laptops_initial[var]], axis=1)
f, ax = plt.subplots(figsize=(6, 6))
fig = sns.boxplot(x=var, y="Price_euros", data = data)
fig.axis(ymin=0, ymax=7000);
plt.xticks(rotation=90);
```



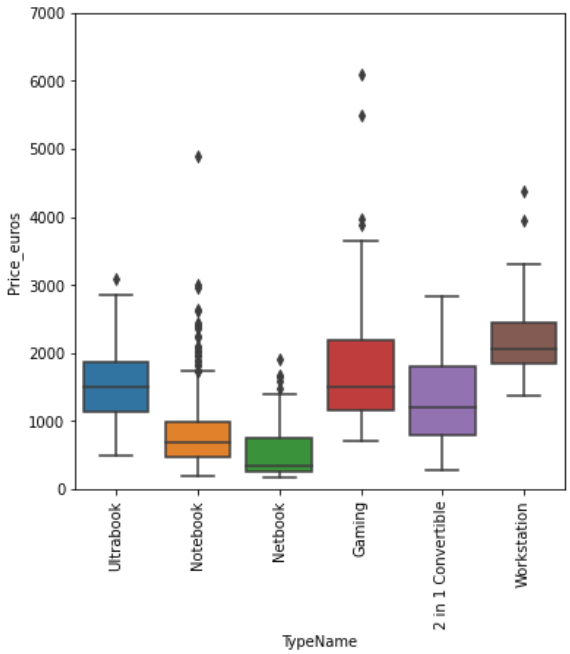
En este caso, se ve una clara diferencia de medias entre las CPU de Intel y las de AMD, que son las dos que abarcan la mayoría de dispositivos. Vemos conveniente por tanto disponer de dicha información en el modelo.

```
# Boxplot GPU_Company/Price_euros:
var = 'GPU_Company'
data = pandas.concat([laptops_initial['Price_euros'], laptops_initial[var]], axis=1)
f, ax = plt.subplots(figsize=(6, 6))
fig = sns.boxplot(x=var, y="Price_euros", data = data)
fig.axis(ymin=0, ymax=7000);
plt.xticks(rotation=90);
```



En este caso, aunque no de forma tan clara como en el de las CPU, se pueden apreciar lo que serían 3 gamas de GPU. De más barata a más cara estarían: AMD, Intel y Nvidia. Por lo tanto, consideramos relevante añadirlo al modelo.

```
# Boxplot TypeName/Price_euros:
var = 'TypeName'
data = pandas.concat([laptops_initial['Price_euros'], laptops_initial[var]], axis=1)
f, ax = plt.subplots(figsize=(6, 6))
fig = sns.boxplot(x=var, y="Price_euros", data = data)
fig.axis(ymin=0, ymax=7000);
plt.xticks(rotation=90);
```



En tipos de dispositivos, si bien hay alguna categoría que destaca (como Gaming), el resto de los dispositivos parece que abarcan una franja amplia de precios, con lo que no se ve recomendable incorporarla al modelo.

A continuación procedemos a codificar las variables categóricas como numéricas mediante el método de one-hot para su inclusión en el modelo que generaremos a posteriori. Dado que desgranamos una variable categórica en tantas variables como categorias tenga (k), siempre podremos eliminar una de estas variables resultantes para mejorar el rendimiento del modelo y evitar redundancias.

```
# One-Hot de CPU_Company
dummies = pandas.get_dummies(laptops_initial['CPU_Company'], drop_first = True)
dummies = dummies.rename(columns={"Intel": "CPU_Intel", "Samsung": "CPU_Samsung"})
laptops_initial = laptops_initial.join(dummies)
laptops_initial
```

	Company	TypeName	Inches	Ram(GB)	OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Company	CPU_Version	CPU_Speed(GHz)	GPU_Company	GPU_Version	ScreenResolution_Type	ScreenResolution_Width	CPU_Intel	CPU_Samsung
0	Apple	Ultrabook	13.3	8	macOS	1.37	1339.69	128.0	0.0	0.0	0.0	Intel	Core i5	2.3	Intel	Iris Plus Graphics 640	IPS Panel Retina Display	2560	1	0
1	Apple	Ultrabook	13.3	8	macOS	1.34	898.94	0.0	0.0	128.0	0.0	Intel	Core i5	1.8	Intel	HD Graphics 6000	Unknown	1440	1	0
2	HP	Notebook	15.6	8	No OS	1.86	575.00	256.0	0.0	0.0	0.0	Intel	Core i5 7200U	2.5	Intel	HD Graphics 620	Full HD	1920	1	0
3	Apple	Ultrabook	15.4	16	macOS	1.83	2537.45	512.0	0.0	0.0	0.0	Intel	Core i7	2.7	AMD	Radeon Pro 455	IPS Panel Retina Display	2880	1	0
4	Apple	Ultrabook	13.3	8	macOS	1.37	1803.60	256.0	0.0	0.0	0.0	Intel	Core i5	3.1	Intel	Iris Plus Graphics 650	IPS Panel Retina Display	2560	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	14.0	4	Windows 10	1.80	638.00	128.0	0.0	0.0	0.0	Intel	Core i7 6500U	2.5	Intel	HD Graphics 520	IPS Panel Full HD / Touchscreen	1920	1	0
1299	Lenovo	2 in 1 Convertible	13.3	16	Windows 10	1.30	1499.00	512.0	0.0	0.0	0.0	Intel	Core i7 6500U	2.5	Intel	HD Graphics 520	IPS Panel Quad HD+ / Touchscreen	3200	1	0
1300	Lenovo	Notebook	14.0	2	Windows 10	1.50	229.00	0.0	0.0	64.0	0.0	Intel	Celeron Dual Core N3050	1.6	Intel	HD Graphics	Unknown	1366	1	0
1301	HP	Notebook	15.6	6	Windows 10	2.19	764.00	0.0	1024.0	0.0	0.0	Intel	Core i7 6500U	2.5	AMD	Radeon R5 M330	Unknown	1366	1	0
1302	Asus	Notebook	15.6	4	Windows 10	2.20	369.00	0.0	500.0	0.0	0.0	Intel	Celeron Dual Core N3050	1.6	Intel	HD Graphics	Unknown	1366	1	0

1303 rows × 20 columns

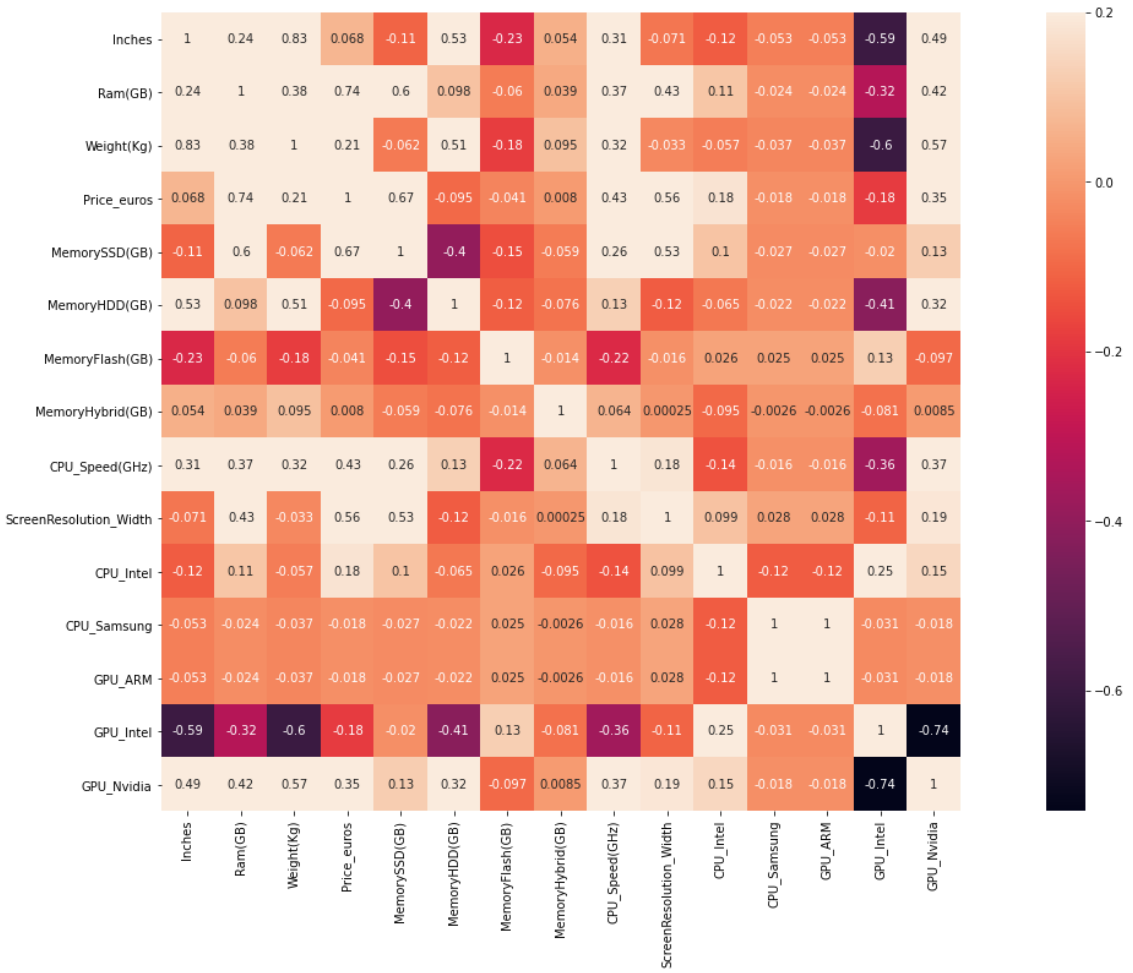
```
# One-Hot de GPU_Company
dummies = pandas.get_dummies(laptops_initial['GPU_Company'], drop_first = True)
dummies = dummies.rename(columns={"ARM": "GPU_ARM", "Intel": "GPU_Intel", "Nvidia": "GPU_Nvidia"})
laptops_initial = laptops_initial.join(dummies)
laptops_initial
```

	Company	TypeName	Inches	Ram(GB)	OpSys	Weight(Kg)	Price_euros	MemorySSD(GB)	MemoryHDD(GB)	MemoryFlash(GB)	MemoryHybrid(GB)	CPU_Company	CPU_Version	CPU_Speed(GHz)	GPU_Company	GPU_Version	ScreenResolution_Type	ScreenResolution_Width	CPU_Intel	CPU_Samsung	GPU_ARM	GPU_Intel	GPU_Nvidia
0	Apple	Ultrabook	13.3	8	macOS	1.37	1339.69	128.0	0.0	0.0	0.0	Intel	Core i5	2.3	Intel	Iris Plus Graphics 640	IPS Panel Retina Display	2560	1	0	0	1	0
1	Apple	Ultrabook	13.3	8	macOS	1.34	898.94	0.0	0.0	128.0	0.0	Intel	Core i5	1.8	Intel	HD Graphics 6000	Unknown	1440	1	0	0	1	0
2	HP	Notebook	15.6	8	No OS	1.86	575.00	256.0	0.0	0.0	0.0	Intel	Core i5 7200U	2.5	Intel	HD Graphics 620	Full HD	1920	1	0	0	1	0
3	Apple	Ultrabook	15.4	16	macOS	1.83	2537.45	512.0	0.0	0.0	0.0	Intel	Core i7	2.7	AMD	Radeon Pro 455	IPS Panel Retina Display	2880	1	0	0	0	0
4	Apple	Ultrabook	13.3	8	macOS	1.37	1803.60	256.0	0.0	0.0	0.0	Intel	Core i5	3.1	Intel	Iris Plus Graphics 650	IPS Panel Retina Display	2560	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1298	Lenovo	2 in 1 Convertible	14.0	4	Windows 10	1.80	638.00	128.0	0.0	0.0	0.0	Intel	Core i7 6500U	2.5	Intel	HD Graphics 520	IPS Panel Full HD / Touchscreen	1920	1	0	0	1	0
1299	Lenovo	2 in 1 Convertible	13.3	16	Windows 10	1.30	1499.00	512.0	0.0	0.0	0.0	Intel	Core i7 6500U	2.5	Intel	HD Graphics 520	IPS Panel Quad HD+ / Touchscreen	3200	1	0	0	1	0
1300	Lenovo	Notebook	14.0	2	Windows 10	1.50	229.00	0.0	0.0	64.0	0.0	Intel	Celeron Dual Core N3050	1.6	Intel	HD Graphics	Unknown	1366	1	0	0	1	0
1301	HP	Notebook	15.6	6	Windows 10	2.19	764.00	0.0	1024.0	0.0	0.0	Intel	Core i7 6500U	2.5	AMD	Radeon R5 M330	Unknown	1366	1	0	0	0	0
1302	Asus	Notebook	15.6	4	Windows 10	2.20	369.00	0.0	500.0	0.0	0.0	Intel	Celeron Dual Core N3050	1.6	Intel	HD Graphics	Unknown	1366	1	0	0	1	0

1303 rows × 23 columns

Para tratar de ver el efecto real de estas variables, ejecutaremos de nuevo la matriz de correlación y veremos si la relación entre las variables que hemos pasado a numéricas y el precio es buena para incluirlas en el modelo.

```
# Matriz de correlación:
corrmat = laptops_initial.corr()
f, ax = plt.subplots(figsize=(25, 12))
sns.heatmap(corrmat, annot=True, vmax=.2, square=True);
```



A partir de los resultados, vemos que la única variable que podría ser relevante para el modelo es la de GPU_Nvidia, al tener una correlación con el precio de 0.35. También podría llegar a ser candidata CPU_Intel, al haber obtenido un 0.18 de correlación.

Creación y evaluación del modelo

A partir de las variables que se ha identificado tienen más correlación e influencia sobre la variable que queremos estimar, Price_euros, generamos un subset de datos que nos servirá para generar el modelo de predicción:

```
# Generación del dataset para cargar en el modelo con las columnas seleccionadas
laptops_finished = laptops_initial[["Ram(GB)", "Weight(Kg)", "MemorySSD(GB)", "CPU_Speed(GHz)", "ScreenResolution_Width", "GPU_Nvidia", "Price_euros"]]
```

Procedemos a normalizar los valores de todas las variables para evitar que las diferentes magnitudes de cada una de ellas afecten al rendimiento del modelo final:

```
# Normalización de los valores del dataset
scaler = MinMaxScaler(feature_range=(0,1))
ind_cols = ["Ram", "Weight", "MemorySSD", "CPU_Speed", "ScreenResolution_Width", "GPU_Nvidia", "Price_euros"]
final_data = laptops_finished.values
laptops_finished = pandas.DataFrame(scaler.fit_transform(final_data), columns=ind_cols)
laptops_finished.head(5)
```

	Ram	Weight	MemorySSD	CPU_Speed	ScreenResolution_Width	GPU_Nvidia	Price_euros
0	0.096774	0.169576	0.125	0.518519	0.482619	0.0	0.196741
1	0.096774	0.162095	0.000	0.333333	0.029911	0.0	0.122353
2	0.096774	0.291771	0.250	0.592593	0.223929	0.0	0.067679
3	0.225806	0.284289	0.500	0.666667	0.611964	0.0	0.398895
4	0.096774	0.169576	0.250	0.814815	0.482619	0.0	0.275038

A continuación, procederemos a dividir el dataset en un subset de entrenamiento y otro de test para evaluar el rendimiento del modelo.

```
# Hacemos uso de la librería sklearn para dividir en conjunto de train y test
train, test = train_test_split(laptops_finished, test_size = 0.30)
```

A continuación generamos el modelo de regresión lineal a partir de los subsets de entrenamiento y test:

```
# Preparación de las variables de train y test separando variables dependientes e independientes
x_train = train.iloc[:, :-1]
y_train = train.iloc[:, -1]
x_test = test.iloc[:, :-1]
y_test = test.iloc[:, -1]

# Creación del modelo de regresión lineal
model = LinearRegression().fit(x_train, y_train)
r_sq = model.score(x_train, y_train)
print('Coefficient of determination:', round(r_sq, 2), "\n")

# Ejecución del modelo para predecir precios sobre el df de test
y_pred = model.predict(x_test)

# Inversión de la normalización para tener los precios en la escala correcta
# Se debe disponer de un dataframe del mismo tamaño que el normalizado
x_test = x_test.reset_index(drop=True)
interm_df = pandas.concat([x_test, pandas.Series(y_pred)], axis=1)

# Aplicación de la inversión de la normalización
y_pred = scaler.inverse_transform(interm_df)[:, -1]
y_test = scaler.inverse_transform(test)[:, -1]

# Generación de un dataframe para presentar los resultados reales y predichos
pred_price = pandas.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(pred_price)
```

```
Coefficient of determination: 0.7

      Actual    Predicted
0    1064.00   924.932780
1     387.00   487.957291
2    2824.00  2767.572720
3    1499.00  1820.829030
4     938.00  1215.283554
..       ...          ...
386   869.01  1077.023886
387  1142.75  1094.477794
388   412.00   594.457539
389  1249.26  2111.131120
390   521.47   370.033210

[391 rows x 2 columns]
```

Los resultados de esta regresión no dan un resultado excesivamente alto respecto a la precisión del modelo, ya que esta se queda en un coeficiente de determinación de 0.7 (siendo 0 el peor valor y 1 el mejor), lo que indica que el 70% de la variabilidad de la variable dependiente está explicada por las variables independientes. No obstante, puede ser considerado suficiente para identificar si un precio de mercado está por encima o por debajo de lo esperado por sus especificaciones.

5. Representación de los resultados a partir de tablas y gráficas.

A lo largo del informe se han ido mostrando las tablas y gráficas necesarias para realizar los análisis propuestos.

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Si nos basamos en los objetivos que nos hemos planteado, para el podemos llegar a las siguientes conclusiones:

- Se ha obtenido un modelo que es capaz de aproximar un precio justo para el tipo de producto que se está tratando. Aún así, hay desviaciones en varias ocasiones respecto al precio esperado, dado que el modelo no está arrojando unos valores de precisión demasiado elevados. Este último hecho tendría fácil solución añadiendo más datos de ordenadores de marcas diferentes para enriquecer el mismo y trabajar con más de 1300 muestras.
- Se ha podido confirmar, en base al contraste de muestras realizado, que los dispositivos de Apple son de media más caros que el resto de los dispositivos de otros fabricantes. Por lo tanto, la afirmación que la gente suele suponer de que la marca "Apple" se paga, según el análisis, es una afirmación correcta.
- Se ha podido verificar cuales de las características de los portátiles eran más influyentes a la hora de subir o bajar el precio del producto. La característica más correlacionada con el precio es la de la memoria RAM.

Asimismo, se han podido extraer otras conclusiones no perseguidas en base al estudio y preparación del dataset. Destacarán, entre otras:

- La categoría de portátiles más caros es la de Gaming, así como la que tiene componentes de mejor prestación pero que, a su vez, hace que los portátiles sean más pesados.
- La compañía que sin lugar a dudas posee el liderazgo en GPU y CPU es Intel.
- Windows es el líder en sistemas operativos, específicamente en la fecha de toma de la muestra, con la versión Windows 10.

7. Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

Se puede encontrar el código mostrado en este informe con todos los pasos detallados con comentarios explicativos, al igual que código adicional de partes que no se han adjuntado por no hacer más largo el informe o evitar repeticiones, en el siguiente enlace de GitHub (https://github.com/carlosalloUOC/PRA2-Limpieza-Analisis), dentro de la carpeta src. Aquí, se encuentra el Notebook desarrollado en formato .ipynb, al igual que un archivo con extensión .html para facilitar su visualización y un extracto del mismo en formato .py. Adicionalmente, por si se tuviera algún problema de visualización, se ha añadido en la misma carpeta un archivo .pdf con el código ejecutado (aunque es preferible por estética ver el .html al ser interactivo y tener estilos que hacen más bonito el estudio y ser este formato interactivo).

8. Contribuciones

Contribuciones	Firma
Investigación previa teórica (lectura material UOC, tutoriales y documentación de limpieza de datos, revisión ejemplos anteriores, etc.)	VB, CA
Investigación y elección dataset (estudio de sus características e idoneidad para llevar a cabo los objetivos).	VB, CA
Desarrollo limpieza de datos	VB, CA
Desarrollo análisis de los datos	VB, CA
Elaboración informe	VB, CA