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 2
2. Integración y selección de los datos de interés a analizar. 2
3. Limpieza de los datos. 3
3.1 ¿Los datos contienen ceros o elementos vacíos? 3
3.2 Preparación de datos. 4
3.3 Identificación y tratamiento de valores extremos. 4
4. Análisis de los datos. 5
4.1 Selección de los grupos de datos que se quieren analizar/comparar 5
4.2 Comprobación de la normalidad y homogeneidad de la varianza. 6
4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos. 6
Contraste de hipótesis 6
Estudio de variables numéricas 8
Estudio de variables categóricas 9
Creación y evaluación del modelo 13
5. Representación de los resultados a partir de tablas y gráficas. 14
6. Resolución del problema. 14
7. Código 14
8. Contribuciones 15

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.

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. 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 1000) 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, 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.

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.

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'.

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. 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 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 que todos los valores estaban indicados en las mismas unidades, GB y Kg respectivamente, y 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).

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). Para tratar este hecho, se han 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).

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. Por tanto, realizando este una separación de estos 3 campos, resultan las columnas CPU_Company, CPU_Version, CPU_Speed(GHz).

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.

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. Tal y como se ha realizado anteriormente, se han separado estos valores en 3 columnas: ScreenResolution_Type, ScreenResolution_Width, ScreenResolution_High.

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 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.

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. 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.

Tras una rápida consulta de precios en el mercado de ordenadores Gaming, 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.

4.1 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:

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

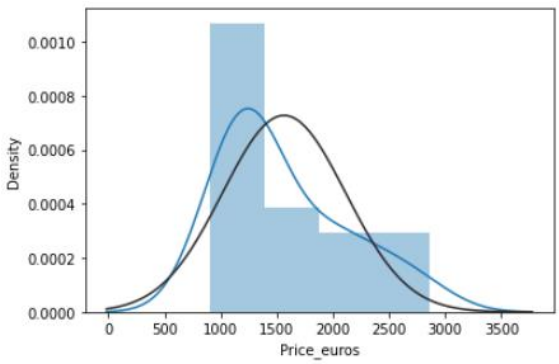
$$H_1: \mu_{Apple} > \mu_{otros}$$

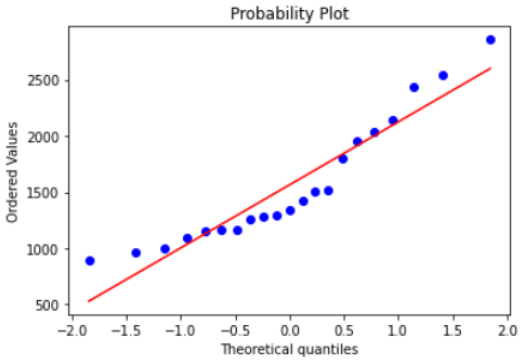
Para llevar a cabo el contraste de muestras, separaremos en dos el dataset de tal forma que contengan respectivamente los precios de los productos de Apple y del resto de marcas.

```
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)
```



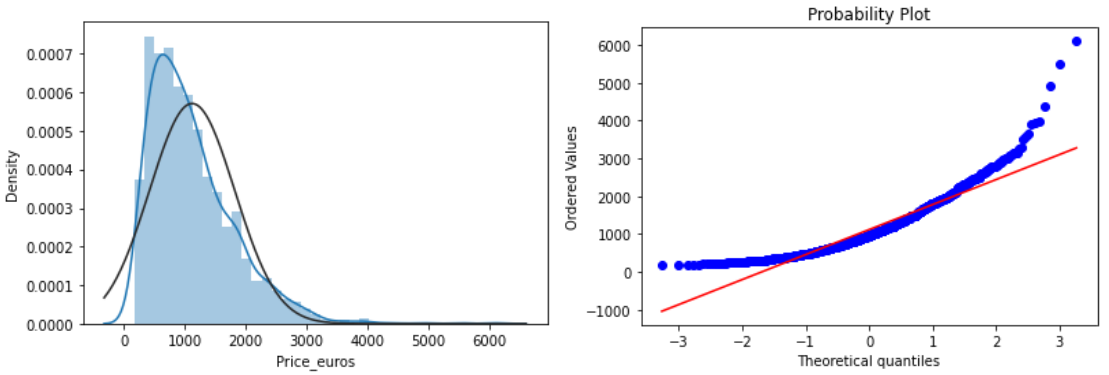



```
# Prueba de Shapiro-Wilk
stat, p = shapiro(apple_prices)
print(f"Stat: {round(stat,3)}")
print(f"p-value: {round(p,3)}")
```

```
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)
```

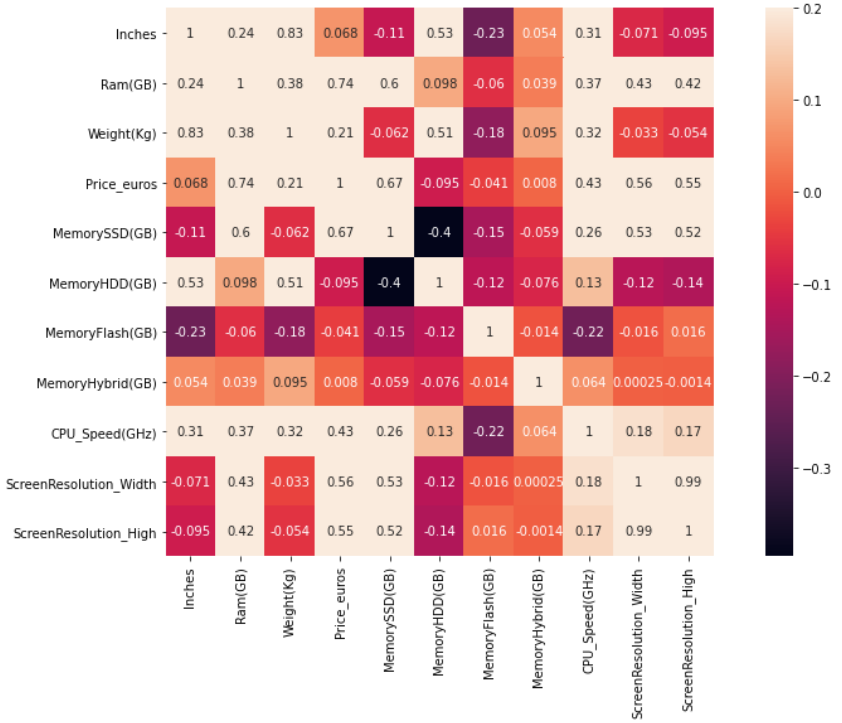
A partir de los resultados obtenidos del test de Mann-Whitney (p-value << 0.05), podemos rechazar la hipótesis nula en favor de la hipótesis alternativa que, en este caso, correspondía con que el precio de los productos de Apple es superior de media que para el resto de marcas.

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.

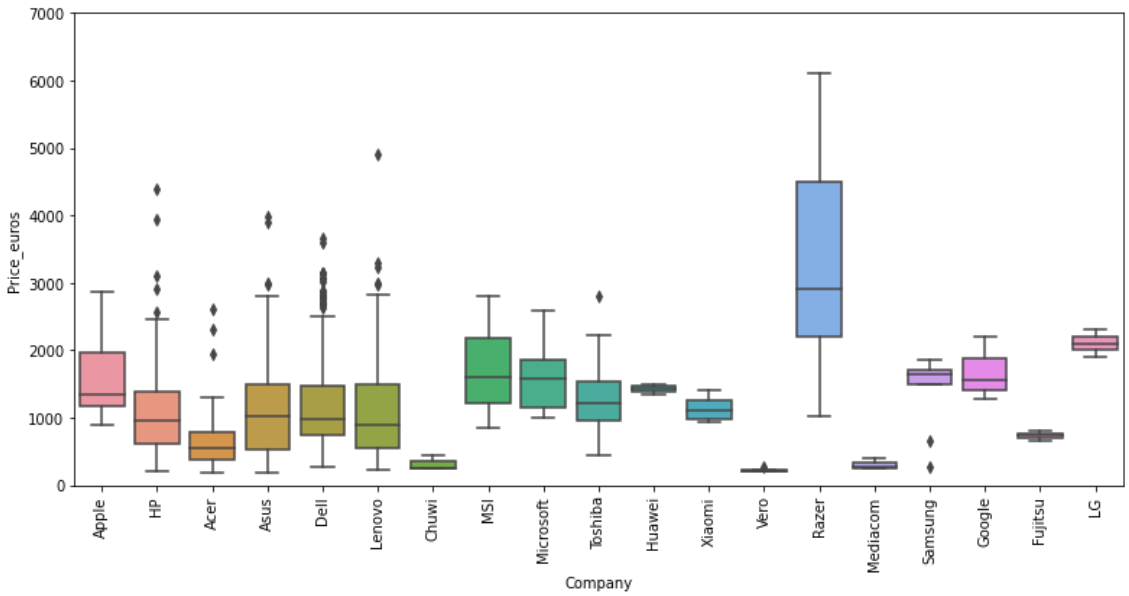
Dado que se identifica que las resoluciones de pantalla a lo ancho y alto están completamente correlacionadas entre sí, se elimina la columna ScreenResolution_High para evitar redundancias.

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

Estudio de variables categóricas

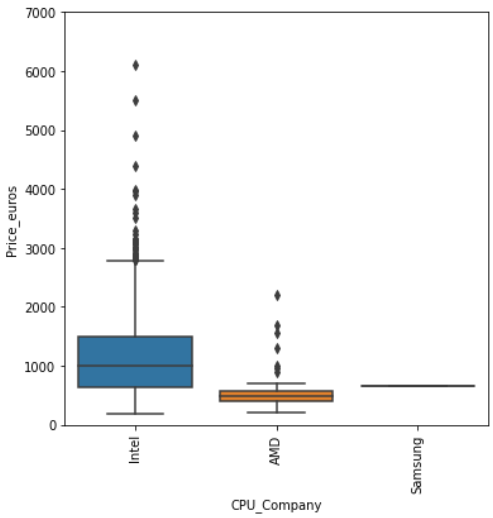
A continuación, vamos a estudiar con más detalle algunas de las variables categóricas que pensamos pueden afectar más al rendimiento del modelo, para identificar visualmente si vemos conveniente que también formen parte del mismo:

```
# 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);
```



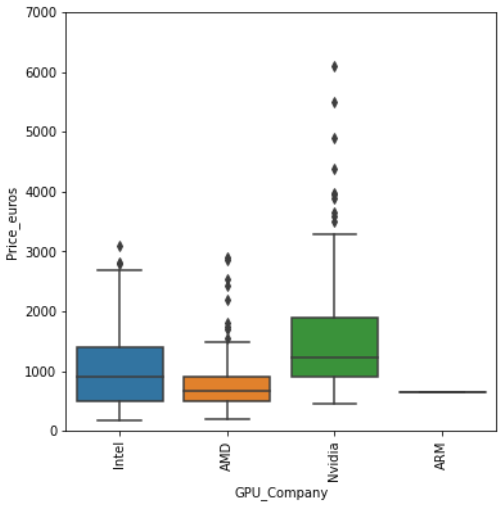
Respecto a las marcas, si bien hay algunas con rangos claramente superiores o inferiores, la mayoría de ellas mantienen una media de precios estable entorno a los 1000-2000€, rango que por otra parte se ha comprobado anteriormente como la media del dataset. En este sentido, no haremos uso de esta variable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



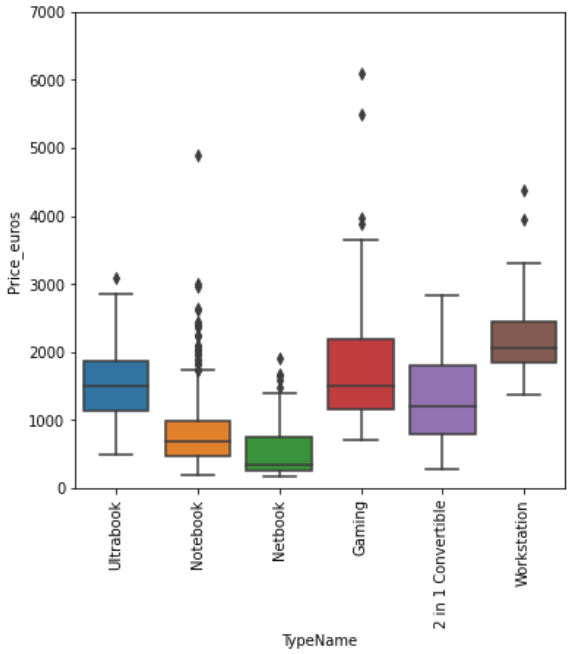
En este caso, se ve una clara diferencia de medias entre las CPU de Intel y las de AMD, que son las dos que abarcan la mayoría de dispositivos. Vemos conveniente por tanto disponer de dicha información en el modelo.

```
# 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

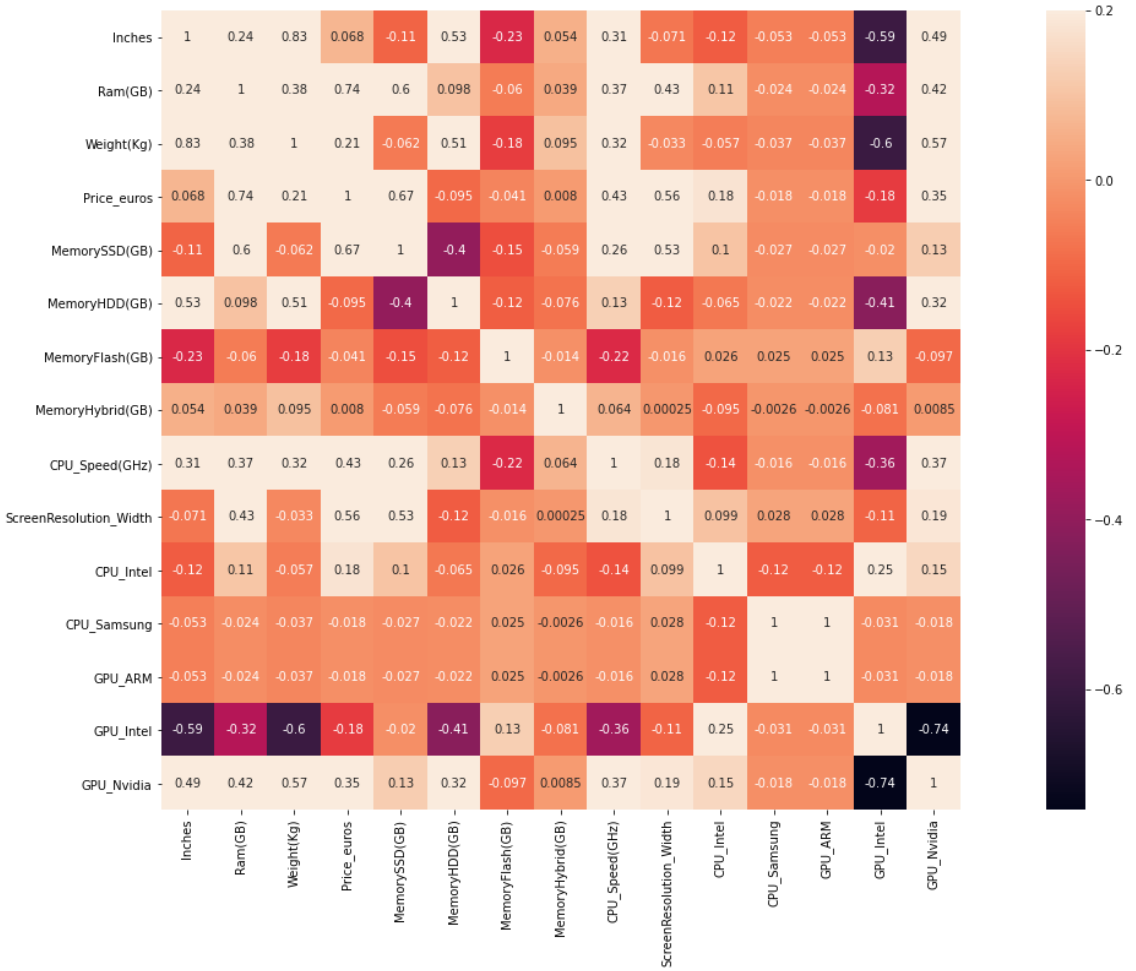
```
# One-Hot de GPU_Company
dummies = pandas.get_dummies(laptops_initial['GPU_Company'], drop_first = True)
dummies = dummies.rename(columns={"ARM": "GPU_ARM", "Intel": "GPU_Intel", "Nvidia": "GPU_Nvidia"})
laptops_initial = laptops_initial.join(dummies)
laptops_initial
```

	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.
- 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.
- 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 ha podido comprobar gracias al análisis de outliers que la categoría de portátiles más caros es la de Gaming, así como la que tiene componentes de mejor prestación pero que, a su vez, tiene los portátiles más pesados.

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 desarrollado en el siguiente enlace de GitHub, 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.

https://github.com/carlosalloUOC/PRA2-Limpieza-Analisis

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