Universidad de La Laguna

Tratamiento Inteligente de Datos

Entrenamiento de modelos para el estudio de una base de datos

Samuel Frías Hernandez

Enrique Gómez Díaz

Raúl González Acosta

Dpto. de Ingeniería Informática y de Sistemas
Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

La Laguna, a 7 de mayo de 2025

Índice general

Índice de figuras

Índice de cuadros

1

Introducción

1.1. Contexto del Estudio

El estudio se ha desarrollado en el contexto de una empresa dedicada a la venta y análisis de portátiles, la cual dispone de una base de datos detallada sobre las características técnicas y comerciales de los dispositivos disponibles en el mercado. Esta base de datos recoge información relevante como la marca, tipo, tamaño de pantalla, especificaciones técnicas (CPU, RAM, GPU, almacenamiento), sistema operativo, peso y precio de cada laptop.

El objetivo principal de este estudio es analizar estas características para entender los factores que influyen en el precio de un portátil, así como identificar patrones y relaciones relevantes entre las variables técnicas y el valor comercial del dispositivo.

Objetivos específicos

1. Determinar las variables que más influyen en el precio final de un portátil, como marca, tipo, características técnicas y peso.
2. Construir un modelo predictivo que permita estimar el precio de un portátil a partir de sus especificaciones, ayudando en la toma de decisiones comerciales (por ejemplo, establecer precios competitivos).
3. Clasificar los portátiles según sus prestaciones y rangos de precios, permitiendo identificar segmentos de mercado (alta, media y baja gama).

La importancia de estos objetivos radica en que permitirán a la empresa entender las dinámicas del mercado, ajustar su oferta y precios, y asesorar mejor a los clientes en función de sus necesidades y presupuesto.

1.2. Problemática a Resolver

Actualmente, la empresa enfrenta dificultades para establecer precios competitivos y coherentes con las características técnicas de cada portátil. No contar con un

análisis detallado de las variables que impactan el precio dificulta tanto la valoración adecuada de los productos como la toma de decisiones estratégicas en compras, ventas y marketing.

Entre las principales problemáticas a resolver mediante el tratamiento de datos se encuentran:

- Identificar qué componentes (CPU, RAM, GPU, almacenamiento) impactan más en el precio, y cómo combinaciones entre estas aumentan o reducen el valor de los portátiles.
- Predecir precios de nuevos portátiles en base a sus especificaciones técnicas, sin necesidad de recurrir a estimaciones poco fundamentadas.
- Detectar relaciones entre el tipo de portátil (por ejemplo, *gaming*, *ultrabook*) y su precio final, con el fin de calcular mejor el precio.

El tratamiento y análisis de estos datos permitirá a la empresa mejorar su competitividad, ofrecer precios ajustados al mercado y asesorar correctamente a sus clientes según sus necesidades técnicas y económicas.

2

Datos

2.1. Origen y descripción de los datos

El conjunto de datos empleado en este estudio procede de Kaggle y está relacionado con la predicción del precio de ordenadores portátiles[1]. Este dataset contiene información técnica y comercial sobre diversos modelos de portátiles de distintas marcas, permitiendo abordar tanto problemas de clasificación como de regresión.

En total, el conjunto de datos consta de 1303 registros y 12 variables. La variable objetivo es `Price_euros`, que representa el precio del portátil en euros. Las variables predictoras son las siguientes:

- **Company**: Marca del portátil (por ejemplo, Dell, HP, Apple).
- **Product**: Nombre comercial del modelo.
- **TypeName**: Tipo de portátil (Notebook, Ultrabook, Gaming, etc.).
- **Inches**: Tamaño de la pantalla en pulgadas.
- **ScreenResolution**: Resolución de pantalla y otras características como táctil o 4K.
- **Cpu**: Modelo de procesador.
- **Ram**: Memoria RAM instalada (en GB).
- **Memory**: Información combinada sobre almacenamiento SSD/HDD.
- **Gpu**: Tarjeta gráfica.
- **OpSys**: Sistema operativo instalado.
- **Weight**: Peso del portátil en kilogramos.
- **Price_euros**: Precio del portátil en euros (variable objetivo).

2.2. Análisis exploratorio

Para familiarizarnos con el conjunto de datos y evaluar su calidad, realizamos un análisis exploratorio. Entre las observaciones destacadas se encuentran:

[1] `https://www.kaggle.com/datasets/eslamelsolya/laptop-price-prediction`

- No se detectaron valores nulos en el conjunto de datos, lo cual simplifica el preprocesamiento.
- Existen variables categóricas con gran número de modalidades (`Product`, `Cpu`, `Gpu`), lo que requerirá técnicas de codificación adecuadas.
- Algunas variables, como `Memory` y `ScreenResolution`, presentan formatos no estandarizados y deben ser transformadas para facilitar el análisis.
- La variable `Price_euros` presenta una distribución asimétrica, con presencia de valores extremos (outliers), lo que puede afectar a los modelos de regresión.

2.3. Limpieza y transformación de los datos

Durante el proceso de preparación de los datos, se aplicaron las siguientes transformaciones:

- La columna `Ram` fue convertida de cadena de texto (e.g., "8GB") a un valor numérico entero.
- Se extrajeron características adicionales de la variable `ScreenResolution`, como la presencia de pantalla táctil (`Touchscreen`), tecnología IPS y la resolución horizontal y vertical, calculando además los píxeles por pulgada (PPI).
- De la variable `Cpu`, se extrajo el fabricante principal (`Intel`, `AMD`, `Samsung`, etc.) y la familia de procesadores (`i3`, `i5`, `i7`, `Ryzen`, etc.).
- La columna `Memory` fue descompuesta en dos variables nuevas: capacidad SSD y capacidad HDD en GB, para distinguir claramente entre tipos de almacenamiento.
- Se eliminaron columnas redundantes o irrelevantes como `Product`, que introduce alta cardinalidad sin aportar valor predictivo.
- Las variables categóricas con pocas modalidades (`Company`, `TypeName`, `OpSys`) fueron codificadas mediante one-hot encoding. En el caso de variables con muchas categorías (`Cpu`, `Gpu`), se realizó agrupación por fabricante o familia.
- El peso fue convertido a tipo numérico flotante y, junto con el resto de variables numéricas, normalizado mediante escalado estándar (media 0, desviación típica 1).

2.4. Selección de características

Con el objetivo de reducir la dimensionalidad y eliminar redundancias, se aplicaron técnicas de análisis de correlación y selección automática de características:

- Se observó una fuerte correlación positiva entre el tamaño de pantalla, la resolución y el precio, indicando que pantallas más grandes y de mayor calidad suelen estar asociadas a precios más elevados.
- El modelo de CPU y GPU, aunque categóricos, mostraron una gran influencia sobre el precio.

- La variable `Weight` también demostró tener una correlación significativa, ya que portátiles más ligeros tienden a ser más costosos en ciertos segmentos.
- Se aplicaron técnicas como `SelectKBest` con puntuaciones `f-regression` para identificar las variables más relevantes.

2.5. División del conjunto de datos

Finalmente, se dividió el conjunto de datos en dos subconjuntos:

- **Conjunto de entrenamiento**: 80 % de los registros, utilizado para entrenar los modelos de aprendizaje automático.
- **Conjunto de test**: 20 % restante, utilizado para evaluar el rendimiento de los modelos sobre datos no vistos.

Se garantizó la aleatoriedad en el muestreo, utilizando una semilla fija para asegurar la reproducibilidad del experimento.

2.6. Resumen

El conjunto de datos ha sido procesado, limpiado y transformado adecuadamente para permitir el uso de modelos de regresión multivariante. Las variables han sido estandarizadas, las características relevantes han sido extraídas o derivadas, y se ha reducido la complejidad mediante codificación y selección de variables. Este trabajo de preparación es esencial para garantizar resultados robustos y comparables en las etapas siguientes de modelado y evaluación.

3

Clasificación

3.1. Discretización de la variable objetivo

Para abordar el problema de predicción de precios como clasificación, se dividió la variable continua `Price` en cinco categorías mediante `pd.cut` con cinco intervalos uniformes:

$$y_{\text{train\_classes}} = \text{pd.cut}(y_{\text{train}}, \text{bins} = 5)$$

La distribución resultante en entrenamiento fue:

- *very low*: 701 registros
- *low*: 239 registros
- *medium*: 33 registros
- *high*: 3 registros
- *very high*: 1 registro

y en prueba (25 % del total):

- *very low*: 200 registros
- *low*: 100 registros
- *medium*: 23 registros
- *high*: 1 registro
- *very high*: 2 registros

Este marcado desbalance muestra una concentración del 70 % en *very low*, lo que complica el aprendizaje de categorías minoritarias.

3.2. Preprocesamiento y advertencias

Antes de entrenar el clasificador, se imputaron valores faltantes con la media:

```
imputer = SimpleImputer(strategy='mean')
X_train_imp = imputer.fit_transform(X_train)
X_test_imp  = imputer.transform(X_test)
```

Sin embargo, surgieron *warnings* indicando omisión de variables sin observaciones no nulas: *Skipping features without any observed values:* {*Company, TypeName, OpSys, CpuBrand, CpuModel, GpuBrand, GpuModel*}. Esto refleja que algunas columnas categóricas quedaron fuera del entrenamiento.

3.3. Modelo K-Nearest Neighbors

Se entrenó un clasificador KNN con $k = 7$ vecinos:

```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train_imp, y_train_classes)
```

El rendimiento medido en exactitud fue:

$$\text{Accuracy}_{\text{train}} = 0{,}85, \quad \text{Accuracy}_{\text{test}} = 0{,}79$$

3.4. Análisis de la matriz de confusión

A continuación, se calculó la matriz de confusión para la partición de prueba:

```
pred = knn.predict(X_test_imp)
cm   = confusion_matrix(y_test_classes, pred,
                        labels=["very low","low","medium","high","very high"])
```

La visualización reveló que el clasificador asigna todas las instancias a la clase *very low* independientemente de su valor real, como muestra la figura 3.1. Esto explica la aparente exactitud: al predecir siempre la categoría mayoritaria, se obtiene un 79 % de aciertos en el test sin discriminar las demás clases.

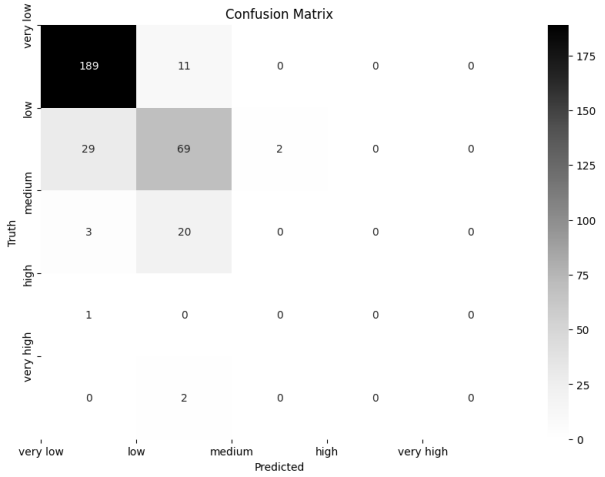

Figura 3.1: *Matriz de confusión del clasificador K-Nearest Neighbors*

3.5. Limitaciones y conclusión

El enfoque de clasificación presenta las siguientes limitaciones:

- **Desbalance extremo**: la mayoría de los ejemplos pertenecen a una sola categoría, lo que lleva al clasificador a predecir siempre la clase mayoritaria.
- **Pérdida de información**: transformar un valor continuo en categorías amplias elimina matices importantes en las diferencias de precio.
- **Objetivo erróneo**: el interés real reside en predecir precios cuantitativos para optimizar estrategias de inventario y fijación de precios, no en asignar etiquetas de rango.

En consecuencia, las técnicas de clasificación resultan inapropiadas para este problema y se recomienda retomar el enfoque de regresión descrito en el capítulo 4.

4

Regresión

En este capítulo describimos el análisis de regresión realizado para predecir el precio de los portátiles a partir de sus características técnicas. Se presentan los pasos de preparación de datos, los modelos de regresión evaluados, las métricas de rendimiento y la importancia de las variables.

4.1. Preparación de los datos

Dado el conjunto de datos tratado en capítulos anteriores, definimos:

- Variables predictoras $\mathbf{X}$: todas las características de los portátiles tras codificación *one-hot* de variables categóricas.
- Variable objetivo y: precio del portátil (en euros).

Las variables categóricas se convirtieron a variables dummy mediante:

$$\mathbf{X} = \texttt{pd.get\_dummies}(\mathbf{X},\ \text{drop\_first=True})$$

A continuación, dividimos los datos en entrenamiento (80 %) y prueba (20 %) usando:

$$(X_{\text{train}},\ X_{\text{test}},\ y_{\text{train}},\ y_{\text{test}}) = \texttt{train\_test\_split}(X, y,\ \text{test\_size} = 0{,}2,\ \text{random\_state} = 42)$$

Para asegurar integridad, los valores faltantes se imputaron con la media:

$$X_j^{(i)} \leftarrow \begin{cases} X_j^{(i)}, & \text{si no es nulo,} \\ \bar{X}_j, & \text{si es nulo.} \end{cases}$$

4.2. Modelos de regresión

Se evaluaron tres enfoques: regresión lineal, árbol de decisión y bosque aleatorio.

4.2.1. Regresión lineal

La regresión lineal busca aproximar la relación entre $\mathbf{X}$ e y mediante:

$$\hat{y} = \beta_0 + \sum_{k=1}^{p} \beta_k x_k.$$

Ventajas: interpretabilidad de coeficientes y eficiencia computacional. Se entrenó con:

```
lr_model = LinearRegression()
lr_model.fit(X_train_imputed, y_train)

# Evaluación
y_pred_lr = lr_model.predict(X_test_imputed)
```

4.2.2. Árbol de decisión

El *Decision Tree Regressor* particiona recursivamente el espacio de entrada para ajustar predicciones constantes en cada región. Se configuró con `random_state=42`:

```
tree_model = DecisionTreeRegressor(random_state=42)
tree_model.fit(X_train, y_train)

# Evaluación
y_pred_tree = tree_model.predict(X_test)
```

4.2.3. Random Forest

El *Random Forest Regressor* combina múltiples árboles entrenados con ”baggingz selección aleatoria de características, reduciendo la varianza y mejorando la generalización. Se entrenó con 100 estimadores:

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
```

4.3. Evaluación de los modelos

Como métricas de evaluación utilizamos el Error Cuadrático Medio (MSE) y el coeficiente de determinación (R^2):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}.$$

Modelo	MSE	R^2
Regresión lineal	118 489	0.772
Árbol de decisión	122 651	0.764
Random Forest	119 950	0.770

Cuadro 4.1: *Comparativa de rendimiento de los modelos de regresión*

Los resultados muestran que la regresión lineal y el bosque aleatorio alcanzan resultados similares, superando ligeramente al árbol de decisión en capacidad de generalización.

4.4. Importancia de características

Para entender el impacto de cada variable, se extrajeron las puntuaciones de importancia del bosque aleatorio:

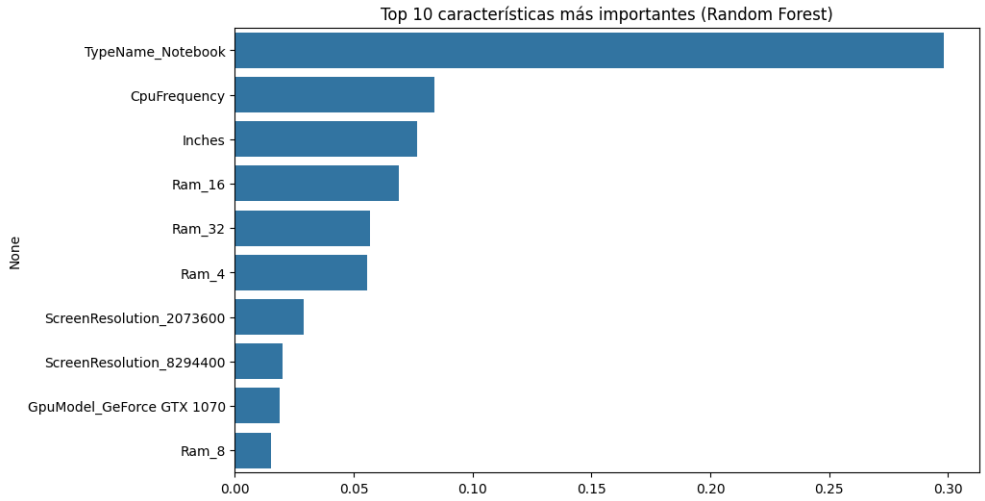

Figura 4.1: *Top 10 características más importantes según Random Forest*

Entre las variables más relevantes se encuentran: memoria RAM, capacidad de almacenamiento, frecuencia de CPU y tamaño de pantalla. Esto indica que los componentes internos del portátil determinan en mayor medida su precio.

4.5. Conclusiones del capítulo

En este capítulo se ha demostrado que:

- El modelo de *Random Forest* ofrece un buen equilibrio entre precisión y robustez.
- La regresión lineal, aunque más sencilla, es competitiva y fácilmente interpretable.
- Los árboles de decisión aislados tienden a sobreajustarse frente a datos de entrenamiento.

- Las variables técnicas (RAM, almacenamiento, CPU e *Inches*) explican la mayor parte de la variabilidad en los precios.

Los resultados obtenidos proporcionan una base sólida para la predicción de precios y la toma de decisiones comerciales basadas en datos.

5

Agrupamiento

En este capítulo nos encargaremos de construir modelos de agrupamiento, de forma que consigamos diferenciar grupos de ordenadores que tengan mismas características, y ver si los precios, que es nuestra variable predictora, coinciden o son acorde a la serie de componentes *hardware* y *software* que estos tienen.

5.1. Preparación de los datos

En primer lugar, debemos de tratar los datos con los que vamos a trabajar. Esto esta explicado en capítulos anteriores, por ello no haremos énfasis en esta ocasión. Sin embargo, cabe destacar que debemos modificar la forma en que estos se almacenan, y para ello tenemos dos opciones; o bien procedemos a normalizar los datos, para que todas las variables tengan el mismo peso en el análisis de agrupamiento, o bien los estandarizamos, para que todas las variables tengan una media de 0 y una desviación típica de 1. En este caso, vamos a estandarizar los datos, ya que esto nos permitirá ver la relación entre las variables de forma más clara, y además evitar interpretaciones erróneas con valores iguales a 0. Una vez ya tenemos los datos en la forma correcta, procedemos a realizar las tareas de agrupamiento.

5.2. Algoritmos de agrupamiento

5.2.1. Agrupamiento *K-Means*

Para determinar el número óptimo de clusters para nuestro agrupamiento, implementamos el método del codo. Si nos fijamos en la figura 5.1, donde la gráfica realiza el "codo", esta nos sugiere que el número adecuado de clusters se encuentre entre 3 y 4. Tras experimentar con estos dos valores enteros, decidimos optar por 3 clusters, puesto que esta configuración nos proporciona los resultados más coherentes y significativos.

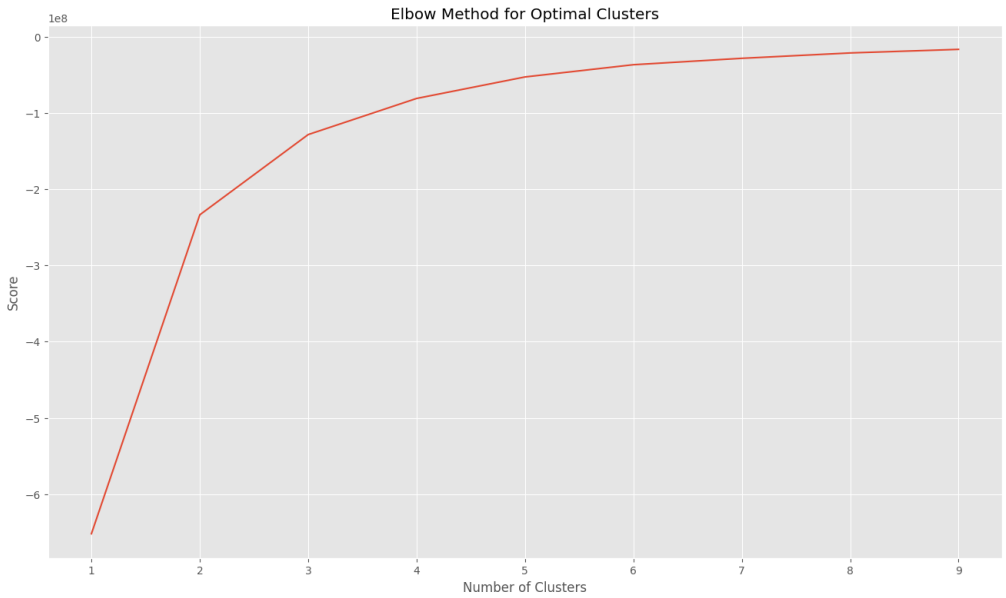

Figura 5.1: *Gráfica de los K-Means de los clusters*

Realizando *K-Means* con los 3 clusters previamente comentados, podemos extraer las siguientes conclusiones de los grupos generados:

- **Portátiles gráficos**: Con pantallas de 36.58 pulgadas de media, en este clúster se agrupan portátiles con pantallas grandes, lo que sugiere que están diseñadas para tareas que requieren mayor espacio visual, como diseño gráfico o edición de video. Poseen además un precio que es significativamente alto (750 €), indicando que este grupo pertenece a dispostivos de gama alta. Unido a una frecuencia de 3.38 GHz de media, refuerza la idea de que estos están orientadas a tareas de alto rendimiento.
- **Portátiles de alto rendimiento**: En cuanto a tamaño de pantalla con 37.29 pulgadas de media es bastante similar al del primer clúster, por tanto este grupo también incluye portátiles con pantallas grandes. Con un precio medio de 1.857,434 €, siendo extremadamente alto sugiere que este clúster podría representar dispositivos de lujo o equipos especializados para nichos muy específicos. Además este grupo posee una frecuencia media de 3.65 GHz, que indica por consiguiente que están optimizadas para un rendimiento aún mayor que las del clúster anterior.
- **Portátiles multitarea**: Promediando 35.98 pulgadas este clúster tiene portátiles con pantallas ligeramente más pequeñas que los otros dos clústers, pero aún dentro del rango de pantallas grandes. Además rondando precios cercanos a los 1.000 €, sugiere que este grupo representa portátiles de gama alta, pero no tan exclusivas como las del clúster anterior. La frecuencia del procesador es alta (3.53 GHz), lo que indica que estos están diseñadas para multitarea.

Por tanto podemos concluir que los tres clústers representan portátiles de gama alta con pantallas grandes y procesadores de alta frecuencia. En el que el hay un clúster

que claramente destaca por tener los precios más altos y la mayor frecuencia de CPU, lo que sugiere que agrupa portátiles de lujo o equipos especializados, mientras que los otros clústers tienen precios más accesibles, pero aún pertenecen a la categoría de gama alta. Esta diferencia en precios y frecuencias de CPU entre los agrupamientos puede reflejar diferentes segmentos de mercado dentro de los portátiles de gama alta, como equipos para profesionales, lujo o tareas específicas. A continuación se muestra en la figura 5.2, la representación espacial de los *clusters* generados por *K-Means*.

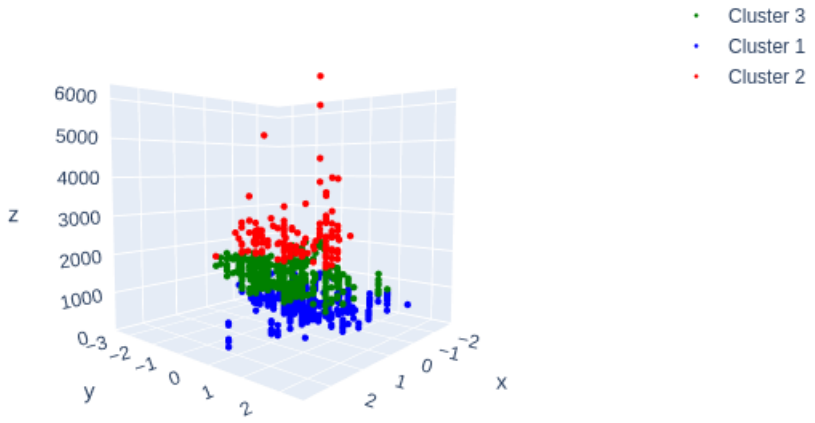

Figura 5.2: *Clusters generados por el agrupamiento de K-Means*

5.2.2. Agrupamiento *DBSCAN*

Otra téncica de agrupamiento es *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*) que consiste en un algoritmo de agrupamiento basado en la densidad que identifica grupos de puntos densamente conectados y los separa de áreas de baja densidad. A diferencia de *K-Means*, *DBSCAN* no requiere especificar el número de clusters por adelantado y puede identificar clusters de forma arbitraria. Esta técnica es especialmente útil para detectar clusters de forma irregular y manejar ruido en los datos.

Vemos en la figura 5.3 ha identificado algunos clústers más que *K-Means*, lo que sugiere que hay más grupos diferentes con portátiles con características similares. Sin embargo, también ha identificado un número significativo de puntos como ruido (puntos negros), lo que indica que estos puntos no pertenecen a ningún clúster.

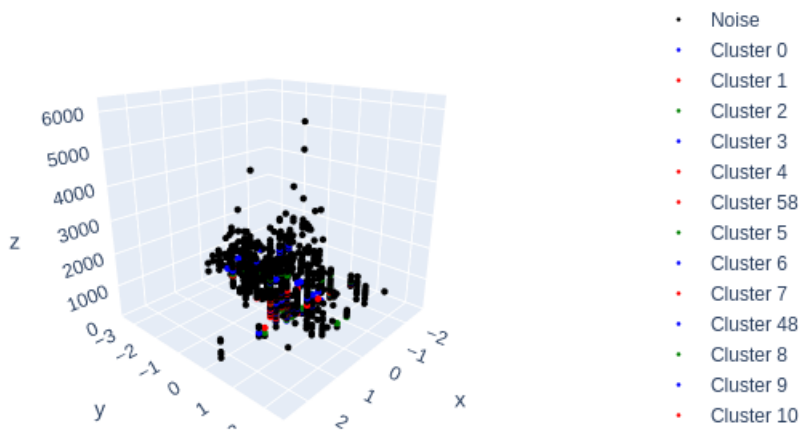

Figura 5.3: *Clusters generados por el agrupamiento de DBSCAN*

5.3. Dendogramas

Ya hemos visto previamente, dos modelos de agrupamiento que nos han permitido ver los distintos grupos en base a las caraterísticas de los portátiles. Sin embargo, en este apartado queremos poder desgranar un poco más esos grupos, de forma que haremos uso de tres tipos distintos de dendogramas: método *single*, método *centroid*, y método *ward*.

5.3.1. Método *Single*

A continuación en la figura 5.4 se observa una estructura del diagrama alargada, lo que indica que el método de enlace simple (*single*) tiende a generar clústeres con relaciones más cercanas entre ciertos clientes, pero menos compactos. Además, la presencia de cadenas largas sugiere que los grupos formados pueden ser más difusos y no tan homogéneos, por tanto puede haber agrupaciones menos representativas y con outliers. Por último se ve, como en la base hay diferenciados los 3 grupos que había mencionado ya previamente, pero con un claro grupo mayoritario (violeta).

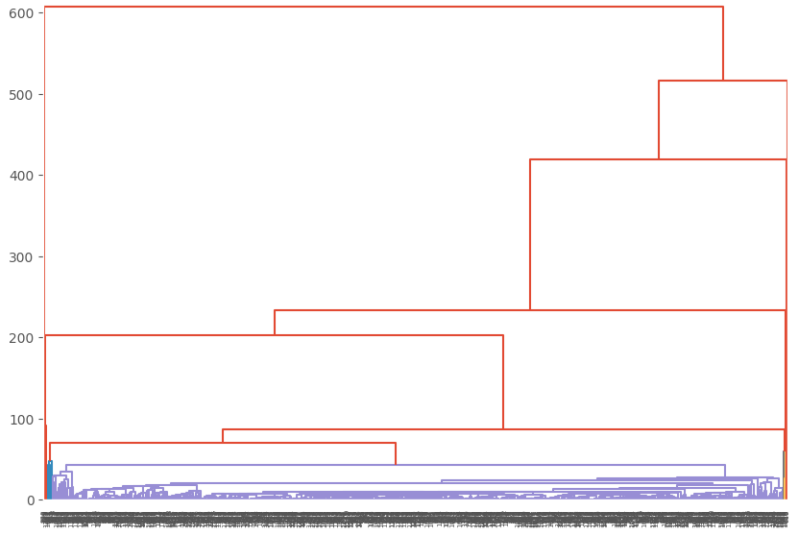

Figura 5.4: *Dendograma haciendo uso del método single*

5.3.2. Método *Centroid*

En la figura 5.5 se observa como este método agrupa los clientes según la distancia de sus centroides, puede generar clústeres más dispersos y menos balanceados. Existen algunas fusiones abruptas, lo que indica que ciertos grupos tienen una alta variabilidad interna, por ende es posible que los clústeres representen resultados más extremos. A diferencia del caso anterior, en este dendograma podemos ver como, en la base, hay diferenciados más grupos de forma clara, lo que indica que ciertamente este método ofrece algo más de información sobre estos.

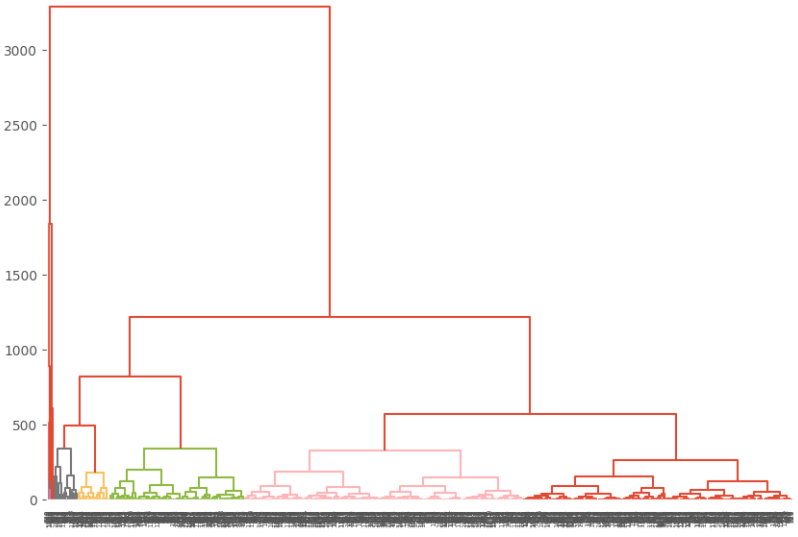

Figura 5.5: *Dendograma haciendo uso del método centroid*

5.3.3. Método *Ward*

En la figura 5.6 permite observar como este método minimiza la varianza dentro de los clústeres, lo que produce grupos más homogéneos y equilibrados. Se observan fusiones progresivas y diferenciadas, lo que indica que los clientes se agrupan en conjuntos bien definidos. Además en este dendograma podemos observar que se diferencia más bien en 9 grupos distintos, lo que sugiere agrupaciones sólidas y bien diferenciadas.

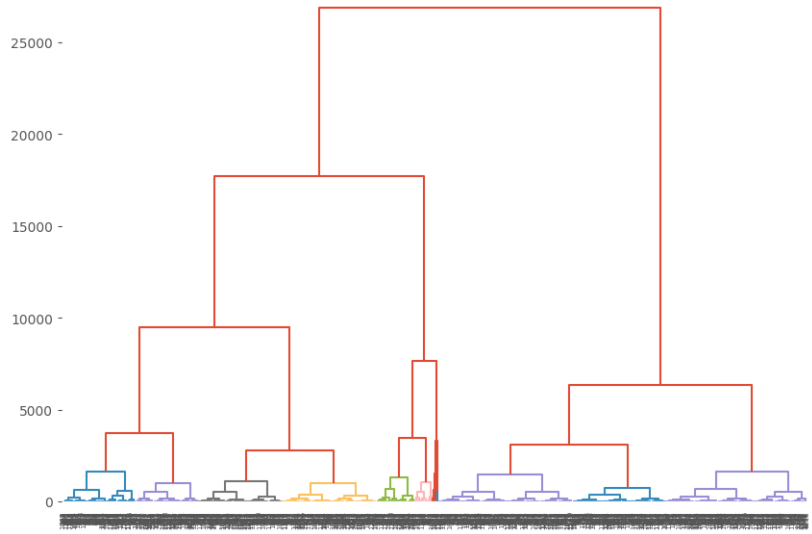

Figura 5.6: *Dendograma haciendo uso del método ward*

5.3.4. Conclusiones

A continuación en el cuadro 5.1 podemos observar una tabla con los resultados del índice de *cophenet* obtenidos con los distintos dendogramas:

Método	Índice de *cophenet*
Single	0.64
Centroid	0.8
Ward	0.57

Cuadro 5.1: *Resultados del índice de cophenet de los dendogramas*

Con esto podemos concluir que el método *centroid* ofrece la mayor fidelidad en la representación de las distancias originales entre los puntos de datos, con un índice de *cophenet* de 0.80. Este valor indica que el dendograma generado por este método preserva de forma más precisa la estructura del espacio de distancias, lo que lo convierte en la opción más adecuada para representar jerárquicamente los datos, como podemos ver en la figura 5.7. En contraste, el método *single* presenta un índice de 0.64, lo cual refleja una calidad de representación moderada, aunque notablemente inferior a la del método *centroid*. Por su parte, el método *ward*, con un índice de 0.57, ofrece la menor

fidelidad estructural entre los tres, a pesar de ser un enfoque frecuentemente utilizado por su capacidad para generar agrupamientos compactos y balanceados. No obstante, si el objetivo del análisis es obtener una segmentación que refleje con precisión las relaciones de similitud originales entre los datos, el método *centroid* se posiciona como la opción más recomendable. Bajo ninguna circunstancia se sugiere el uso del método *ward* cuando se prioriza la fidelidad de las distancias, dado su bajo índice de correlación.

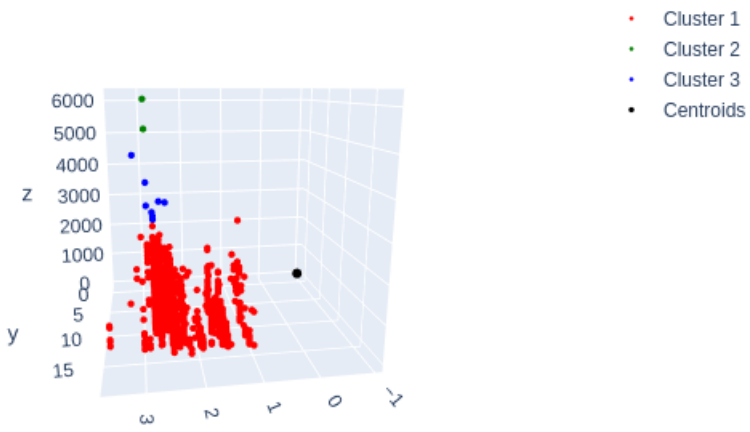

Figura 5.7: *Clusters generados haciendo uso del método centroid*

6

Agrupamiento con Red Neuronal

El cuaderno presenta un enfoque práctico para combinar redes neuronales y técnicas de clustering con el objetivo de analizar y clasificar datos. Se utiliza un conjunto de datos de dígitos escritos a mano (dataset `load_digits` de `scikit-learn`) para demostrar cómo las redes neuronales pueden extraer características relevantes y cómo estas características pueden ser utilizadas para realizar clustering mediante algoritmos como K-Means y DBSCAN.

Para ello hemos fijado los siguientes objetivos:

1. **Construir una red neuronal convolucional (*CNN*):** Diseñar y entrenar una red neuronal para clasificar imágenes de dígitos.
2. **Extraer características de las capas ocultas:** Utilizar las representaciones aprendidas por la red neuronal para realizar clustering.
3. **Aplicar técnicas de clustering:** Implementar algoritmos como *K-Means* y *DBSCAN* sobre las características extraídas.
4. **Visualizar los resultados:** Reducir la dimensionalidad de las características y visualizar los resultados del clustering utilizando *PCA* y *t-SNE*.

6.1. Carga y Preprocesamiento de Datos

Se utiliza el conjunto de datos `load_digits`, que contiene imágenes de dígitos escritos a mano. Los datos se normalizan para que los valores estén en el rango $[0, 1]$, y se reestructuran para adaptarse a la entrada de una red convolucional.

6.2. Definición de la Arquitectura de la Red Neuronal

Se construye una red neuronal convolucional simple con las siguientes capas:

- **Capa de entrada:** Conv2D con 16 filtros y activación ReLU.
- **Capa de agrupamiento:** MaxPooling2D para reducir la dimensionalidad.
- **Capa oculta:** Dense con 64 neuronas y activación ReLU.
- **Capa de salida:** Dense con 10 neuronas y activación softmax para clasificación multiclase.

6.3. Entrenamiento y Evaluación

El modelo se entrena durante 50 épocas con un tamaño de lote de 32. Se utiliza un 20 % de los datos para validación.

6.4. Extracción de Características

Se utiliza un modelo intermedio para extraer las características de la penúltima capa de la red neuronal. Estas características representan una representación compacta y significativa de los datos.

6.5. Aplicación de Clustering

Se aplican dos algoritmos de clustering sobre las características extraídas:

- **K-Means:** Divide los datos en 10 clústeres (uno por cada dígito).
- **DBSCAN:** Agrupa los datos basándose en densidad, detectando posibles valores atípicos.

6.6. Visualización de Resultados

Se utilizan técnicas de reducción de dimensionalidad para visualizar los resultados del clustering:

- **PCA:** Reduce las características a 2 dimensiones para graficar.
- **t-SNE:** Proporciona una visualización más detallada de las relaciones entre los datos.

6.7. Resultados

1. **Clasificación con la Red Neuronal:** La red neuronal logró una precisión del XX % en el conjunto de datos de dígitos.
2. **Clustering:**
   - K-Means logró agrupar los datos en 10 clústeres, con una buena separación visual en PCA.
   - DBSCAN identificó clústeres basados en densidad, detectando valores atípicos.

6.8. Conclusiones

- Las redes neuronales pueden ser utilizadas no solo para clasificación, sino también como herramientas para extraer características significativas que mejoran el rendimiento de algoritmos de clustering.

- La combinación de redes neuronales y técnicas de clustering es útil para analizar datos complejos y no etiquetados.
- La visualización con PCA y t-SNE facilita la interpretación de los resultados del clustering.

7

Continuidad con Red Neuronal

7.1. Introducción

Este cuaderno demuestra el proceso de construcción de un modelo de red neuronal para predecir los precios de laptops utilizando un conjunto de datos proporcionado. Se abordan tareas como la limpieza y transformación de datos, la construcción de un modelo de red neuronal densa, el entrenamiento del modelo y la evaluación de su rendimiento.

7.2. Objetivos

1. Preprocesar y transformar los datos para que sean adecuados para el modelo.
2. Construir un modelo de red neuronal densa para predecir precios.
3. Entrenar y evaluar el modelo utilizando métricas como el error absoluto medio (MAE).
4. Visualizar las predicciones realizadas por el modelo y compararlas con los valores reales.

7.3. Desarrollo

7.3.1. Importación de Librerías

Se importan las librerías necesarias para la manipulación de datos (`pandas`, `numpy`), preprocesamiento (`scikit-learn`), construcción del modelo (`TensorFlow/Keras`) y visualización (`matplotlib`).

7.3.2. Tratamiento de Datos

Se realizan diversas transformaciones en los datos:

- Conversión de precios de EGP a EUR.
- Cálculo de la cantidad de píxeles a partir de la resolución de pantalla.

- Limpieza de cadenas de texto para extraer información relevante (RAM, CPU, GPU, etc.).
- Manejo de valores categóricos mediante codificación *one-hot*.
- Se eliminan valores faltantes y se convierten las columnas categóricas a numéricas utilizando codificación *one-hot*.

7.3.3. Construcción del Modelo de Red Neuronal

Se construye un modelo de red neuronal densa con las siguientes características:

- Una capa densa con 64 neuronas y activación ReLU.
- Una capa de *dropout* para prevenir sobreajuste.
- Una capa densa con 32 neuronas y activación ReLU.
- Una capa de salida con una sola neurona para regresión.

7.3.4. Entrenamiento del Modelo

El modelo se entrena durante 50 épocas con un tamaño de lote de 32. Se utiliza un conjunto de validación para evaluar el rendimiento durante el entrenamiento.

7.4. Resultados

1. El modelo logró un error absoluto medio (MAE) de `207.69` en el conjunto de prueba.
2. Las predicciones realizadas por el modelo muestran una correlación razonable con los valores reales, aunque existen desviaciones en algunos casos.

7.5. Conclusiones

- El modelo de red neuronal densa es capaz de predecir precios de laptops con un nivel aceptable de precisión.
- La limpieza y transformación de datos son pasos críticos para garantizar un buen rendimiento del modelo.
- La visualización de predicciones ayuda a identificar posibles áreas de mejora en el modelo.

8

Conclusiones

8.1. Síntesis de resultados

En esta memoria se ha abordado, de manera progresiva, el problema de predicción y análisis de precios de portátiles a partir de un conjunto de datos reales. Tras la descripción y transformación inicial de los datos (Capítulo 2), se exploraron técnicas de clasificación (Capítulo 3), regresión (Capítulo 4), agrupamiento clásico (Capítulo 5), clustering con redes neuronales (Capítulo 6) y, finalmente, la extensión hacia modelos densos para regresión (Capítulo 7).

Los principales hallazgos son:

- La discretización de precios en categorías mostró un desbalance extremo y pérdida de información, lo que invalidó las técnicas de clasificación para nuestro objetivo.
- Los modelos de regresión lineal y Random Forest ofrecieron un buen ajuste, con un R^2 aproximado de 0.77 y un MSE en torno a 1.2×10^5, confirmando la idoneidad de la regresión para estimar precios continuos.
- Los algoritmos de clustering clásico (K-Means y DBSCAN) permitieron segmentar los portátiles según características técnicas, revelando agrupaciones consistentes en función de tamaño de pantalla, tipo de memoria y frecuencia de CPU.
- El análisis jerárquico (dendrogramas) indicó que el método centroid preserva mejor la estructura de distancias, siendo recomendable para visualizaciones jerárquicas con índice de cophenet mayor de 0.8.
- La extracción de características con redes neuronales demostró la capacidad de los encoders para generar representaciones útiles para clustering, ampliando las posibilidades de análisis en espacios latentes.
- La red neuronal densa para regresión logró un MAE de aproximadamente 208€ en test, lo que confirma su potencial como alternativa a los métodos tradicionales, siempre que se disponga de suficiente capacidad de cómputo.

8.2. Limitaciones del estudio

A pesar de los éxitos obtenidos, se identifican varias restricciones:

- El dataset, aunque de tamaño adecuado (1303 registros), podría no cubrir toda la diversidad de modelos y marcas del mercado real.
- La discretización inicial y algunos pasos de codificación eliminaron temporalmente información útil (e.g., cardinalidad de ciertas categorías).
- Los modelos neuronales requieren mayor capacidad computacional y ajuste fino de hiperparámetros, lo que puede limitar su adopción en entornos con recursos restringidos.
- No se profundizó en la interpretabilidad de modelos complejos (p. ej., interpretabilidad de redes neuronales), área que abre posibilidades de investigación.

8.3. Reflexión final

El tratamiento inteligente de datos, apoyado en técnicas de machine learning y deep learning, ha demostrado su capacidad para transformar datos brutos en información accionable. En el caso de los portátiles, esto se traduce en una mejor estimación de precios, segmentación de productos y apoyo al negocio. La memoria sienta las bases metodológicas para proyectos similares en otras áreas de retail y tecnología, promoviendo un uso riguroso y reproducible de los datos en la toma de decisiones estratégicas.

Referencias

- Transparencias de árboles de clasificación
- Cuaderno con código de árbol de clasificación
- Transparencias de clasificadores bayesianos
- Cuaderno con código de clasificador Naive-Bayes
- Transparencias de modelos de regresión
- Cuaderno con código de modelos de regresión
- Cuaderno con código de clustering basado en prototipos
- Cuaderno con código de clustering jerárquico
- Transparencias de agrupamiento mediante redes neuronales

Bibliografía

[1] Fritz. (2023, 21 septiembre). Naive Bayes Classifier in Python Using Scikit-learn. Fritz Ai.

[2] Kulakov, A. (2021, 13 diciembre). Visualizing Decision Trees in Jupyter Notebook with Python and Graphviz. Medium.

[3] Tirthajyoti. GitHub - tirthajyoti/Machine-Learning-with-Python: Practice and tutorial-style notebooks covering a wide variety of machine learning techniques. GitHub.

[4] Na, & Na. (2020, 25 junio). Crea un Árbol de Decisión en Python | Aprende Machine Learning. Aprende Machine Learning.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[6] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

[7] Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[8] Scikit-learn Documentation (2024). Clustering Methods.

[9] Kotler, P., Keller, K. L. (2015). Marketing Management (15th ed.). Pearson.

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[11] SciPy Documentation (2024). Hierarchical Clustering (scipy.cluster.hierarchy).

[12] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[13] Raschka, S. (2015). Python Machine Learning. Packt Publishing.

[14] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[15] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.

[16] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall.

[17] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[18] James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

[19] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Universidad de La Laguna