Informe Proyecto 2-3
Ciencia de datos

Asignatura:

Introducción a la ciencia de los datos

Programa:

Ingeniería de Sistemas

Estudiantes:

Juan Esteban López Aránzazu - 202313026
Daniel Meléndez Ramirez - 202313024
Joan Sebastián Saavedra Perafán - 202313025

Semestre:

Séptimo Semestre

Docente:

Héctor Fabio Ocampo Arbeláez

Universidad del Valle – Sede Tuluá

Tabla de contenido

Informe del proyecto

I. Estructura del proyecto

El proyecto tiene la siguiente estructura de carpetas/archivos

1. data: directorio para guardar los archivos csv
2. notebooks: directorio con los archivos jupyter notebook para analizar los datos del archivo csv
    a. Análisis descriptivo
    b. Limpieza y normalización de datos
    c. Modelos predictivos
3. README.md: descripcion del proyecto
4. .gitignore: para ignorar archivos

5. Informe: informe del proyecto de la explicación del código

Dependencias utilizadas

1. pandas: para analizar y manipular los datos de un dataset
2. numpy: para hacer cálculos numéricos y análisis de grandes volúmenes de datos
3. matplotlib: para generar las gráficas
4. seaborn: para generar las gráficas
5. sklearn: para los modelos de predicción

II. Análisis Descriptivo del Dataset

Gráficas implementadas

1. Histogramas
2. Boxplots
3. Gráficas de variables numéricas
4. Gráficas de variables categóricas

Datos relevantes del dataset

La media y mediana de las variables numéricas, son medidas de tendencia central donde, la primera nos da una idea general del valor promedio, e identifica cambios cuando se hacen modificaciones al conjunto de datos, y la segunda al ser sensible a los valores extremos, muestra mejor la posición central real de un conjunto de datos.

Su importancia radica al aplicar ambas, por que si son muy diferentes, indicaría que el dataset está sesgado (o asimétrico), lo que es clave para entender cómo interpretar los datos o si necesitas transformarlos antes de aplicar modelos

```
Media de las características numéricas:
Inches                15.022902
Ram                    8.440784
Weight                 2.040525
Price_euros         1134.969059
ScreenW             1900.043922
ScreenH             1073.904314
CPU_freq               2.302980
PrimaryStorage       444.517647
SecondaryStorage     176.069020
dtype: float64

Mediana de las características numéricas:
Inches                15.60
Ram                    8.00
Weight                 2.04
Price_euros          989.00
ScreenW             1920.00
ScreenH             1080.00
CPU_freq               2.50
PrimaryStorage       256.00
SecondaryStorage       0.00
dtype: float64
```

Varianza y desviación estándar de las variables numéricas

Indican qué tanto se dispersan los datos alrededor de la media.

La varianza mide cuánto se alejan los datos de la media en promedio. Existen dos escenarios, si la varianza es baja, los datos están concentrados cerca de la media, y por el contrario, si es alta, los datos están muy dispersos.

La desviación estándar es la raíz cuadrada de la varianza. Por eso: permite entender cuánto se espera que varíen los datos respecto al promedio.

```
Varianza de las características numéricas:
Inches                2.043384
Ram                  25.987660
Weight                0.447823
Price_euros      491054.072107
ScreenW          243390.459608
ScreenH           80590.091308
CPU_freq              0.253860
PrimaryStorage   133617.829162
SecondaryStorage 173023.266818
dtype: float64

Desviación estándar de las características numéricas:
Inches                1.429470
Ram                   5.097809
Weight                0.669196
Price_euros         700.752504
ScreenW             493.346186
ScreenH             283.883940
CPU_freq              0.503846
PrimaryStorage      365.537726
SecondaryStorage    415.960655
dtype: float64
```

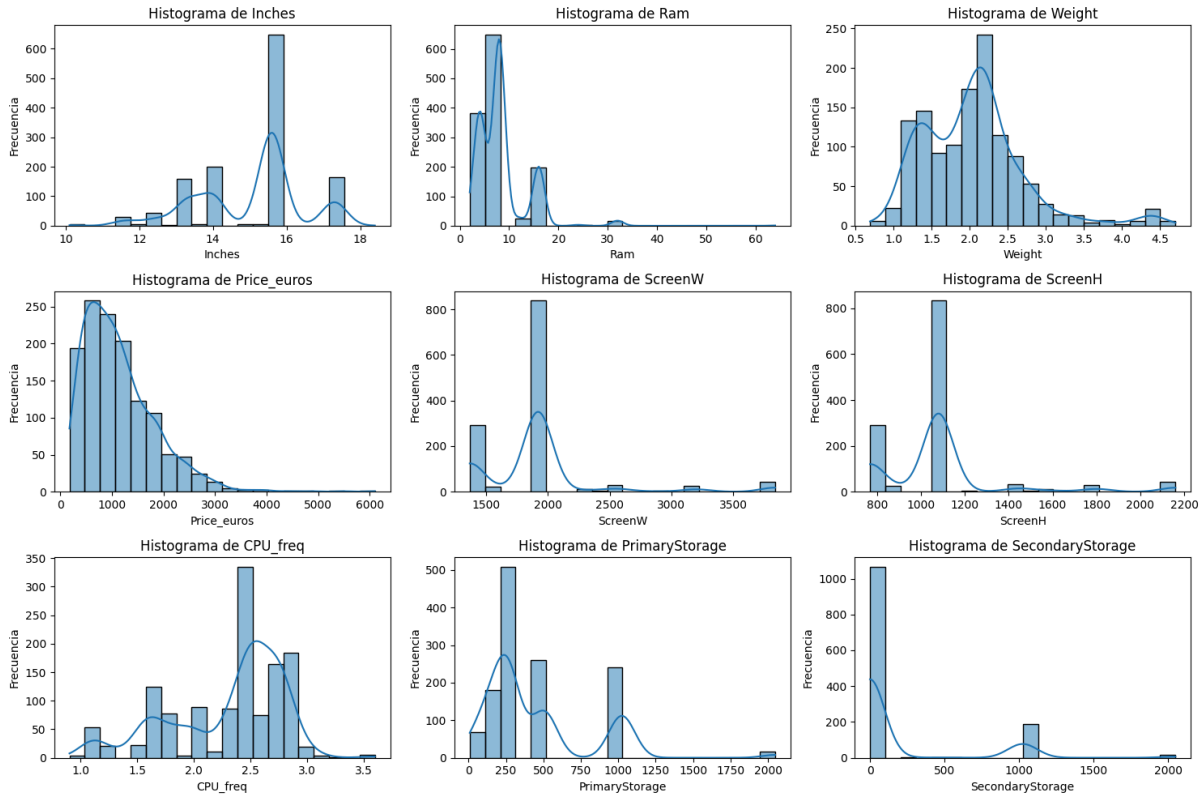
Histograma para las variables numéricas:

Visualizar la distribución de los datos: Permiten observar la forma (asimetría, simetría, sesgo) de cada variable numérica.

Detectar valores atípicos: Al mostrar la frecuencia de los datos agrupados en intervalos, facilitan la identificación de outliers.

Comparar variables: Si se generan varios histogramas, se puede comparar cómo se comportan diferentes variables numéricas entre sí.

Identificar concentraciones de datos: Revelan en qué intervalos se concentra la mayoría de los valores.



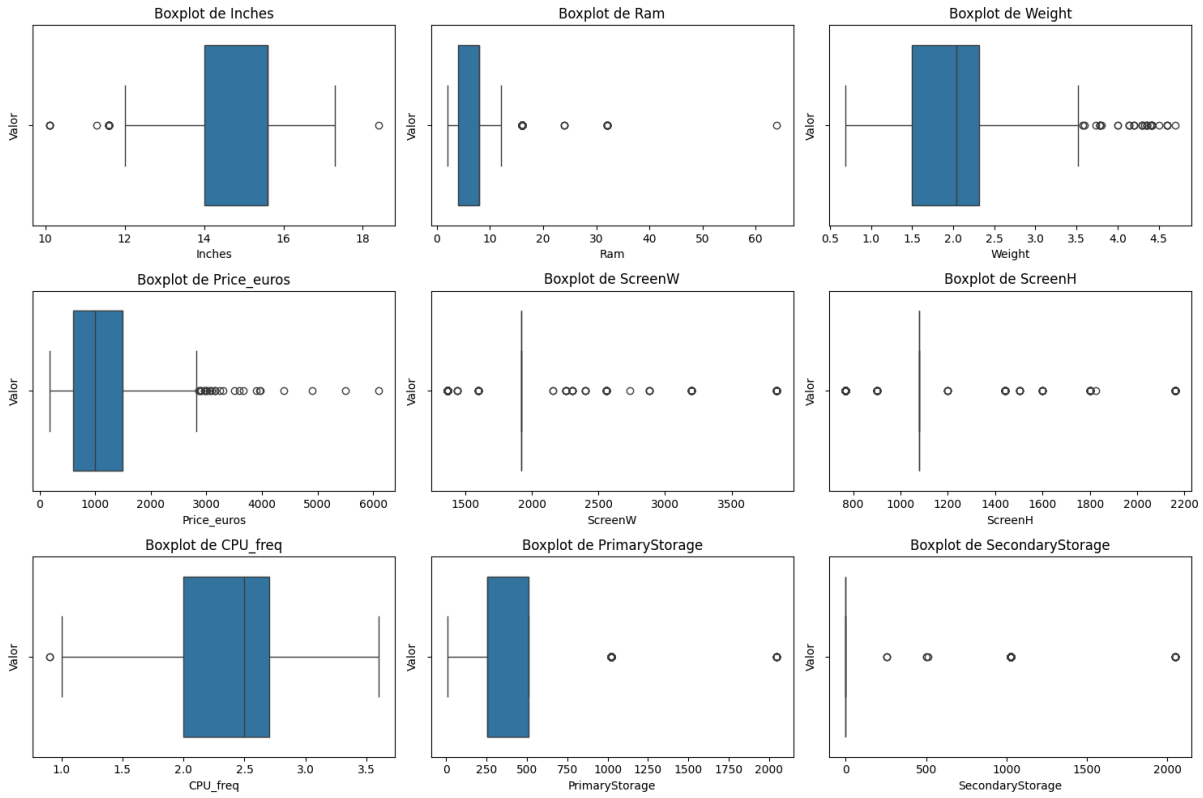
Boxplots para las variables numéricas

Es una herramienta útil que permite identificar:

- la distribución de las variables
- La presencia de valores atípicos (outliers)
- La asimetría de los datos
- Y las diferencias entre categorías si se comparan varios grupos

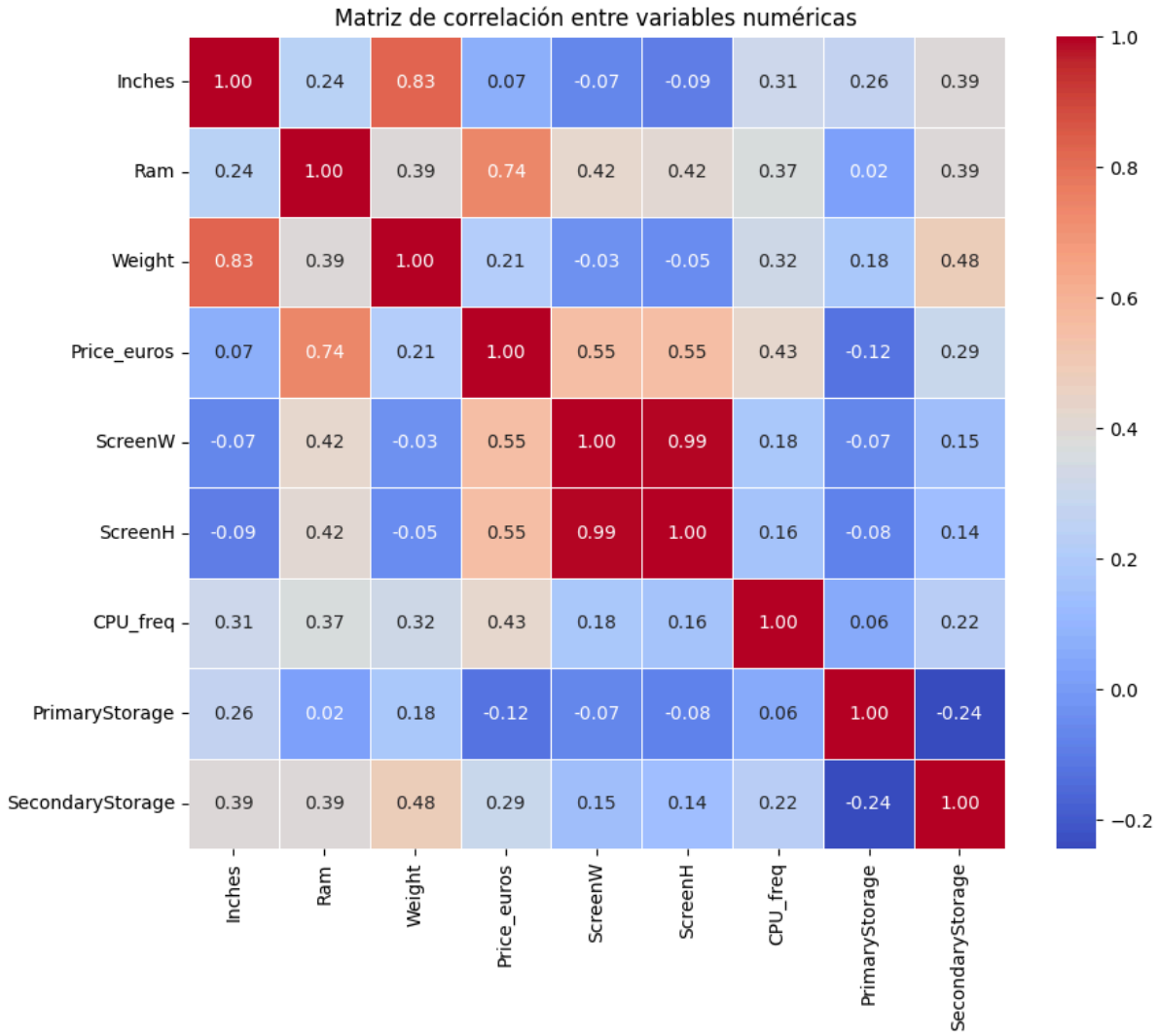
En particular, permiten una visualización rápida de la dispersión y la tendencia central, además de facilitar la comparación entre diferentes variables numéricas medidas.

Esto contribuye significativamente a comprender la estructura de los datos y a detectar posibles problemas que podrían afectar los análisis posteriores, como la regresión o clasificación .



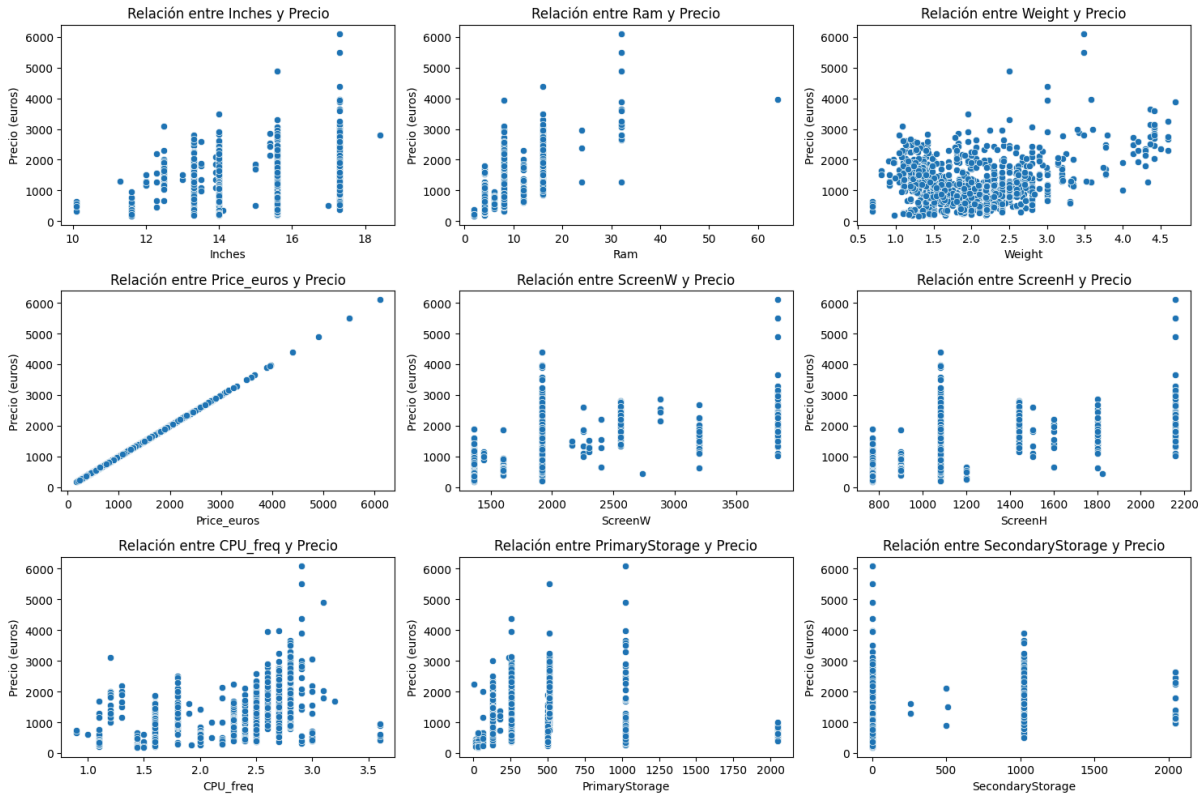
Matriz de correlación para las variables numéricas

En este caso actúa como un filtro exploratorio para guiar decisiones analíticas más fundamentadas en el estudio. Si quieres, puedo ayudarte a interpretar valores específicos de esa matriz o sugerir formas de visualizarla.



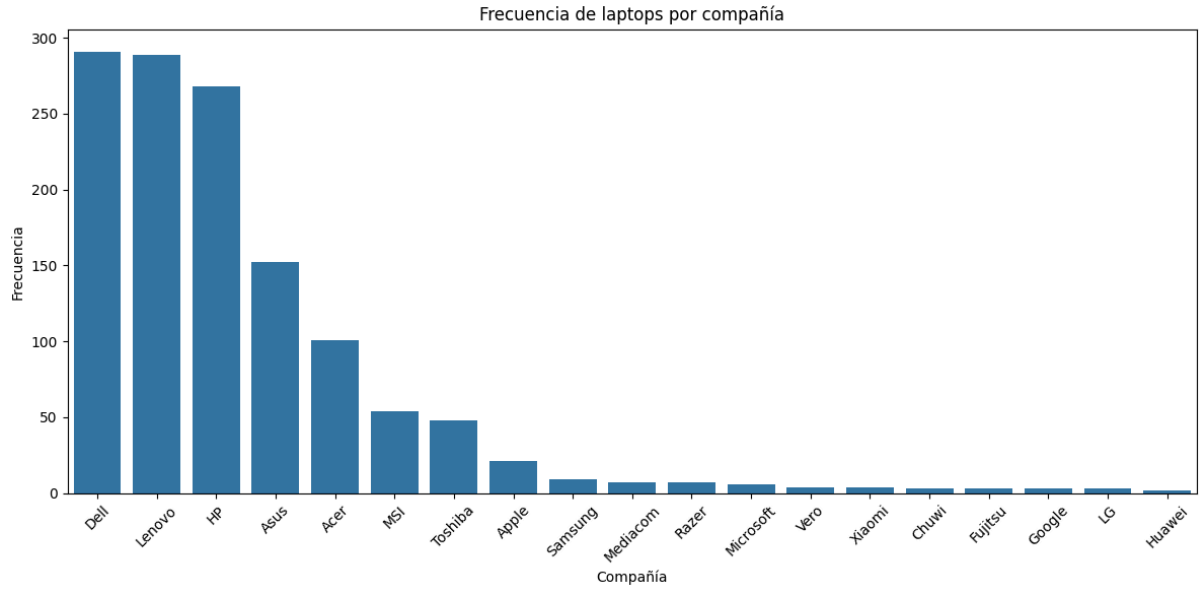
Relación entre el precio y las variables numéricas:

Se concluye que precio está positivamente correlacionado con superficie total, superficie cubierta y ambientes, aunque con dispersión significativa que sugiere la necesidad de modelos predictivos más complejos para explicar el precio completamente.



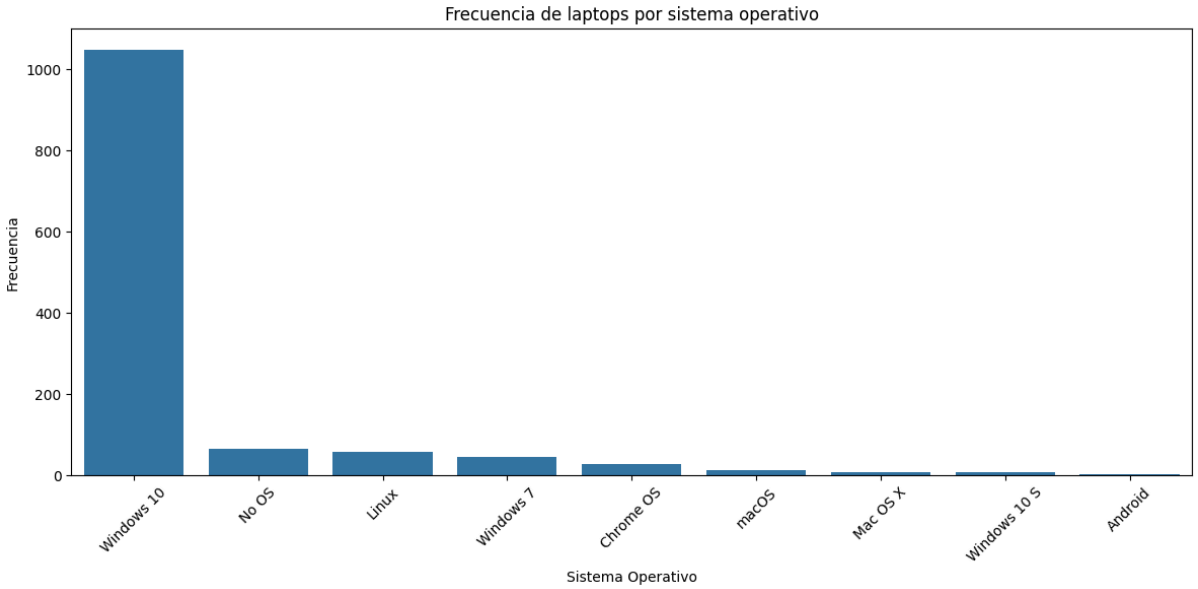
Gráficas variables categóricas

Frecuencia por compañías



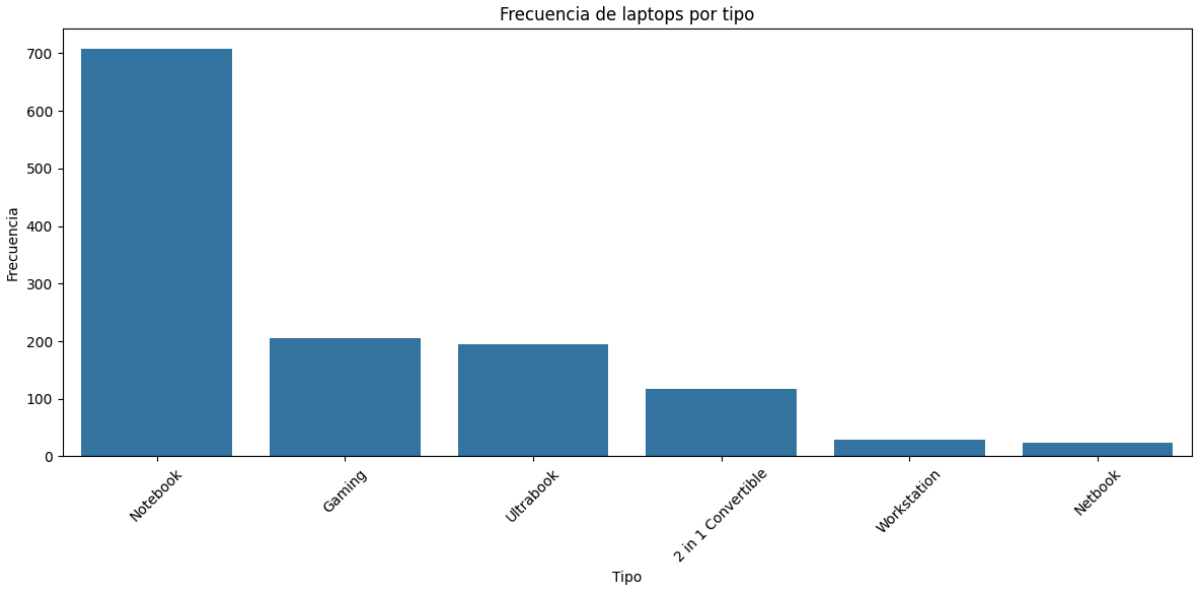
Se puede ver que las compañías con mayor frecuencia son Dell, Lenovo y HP

Frecuencia por SO



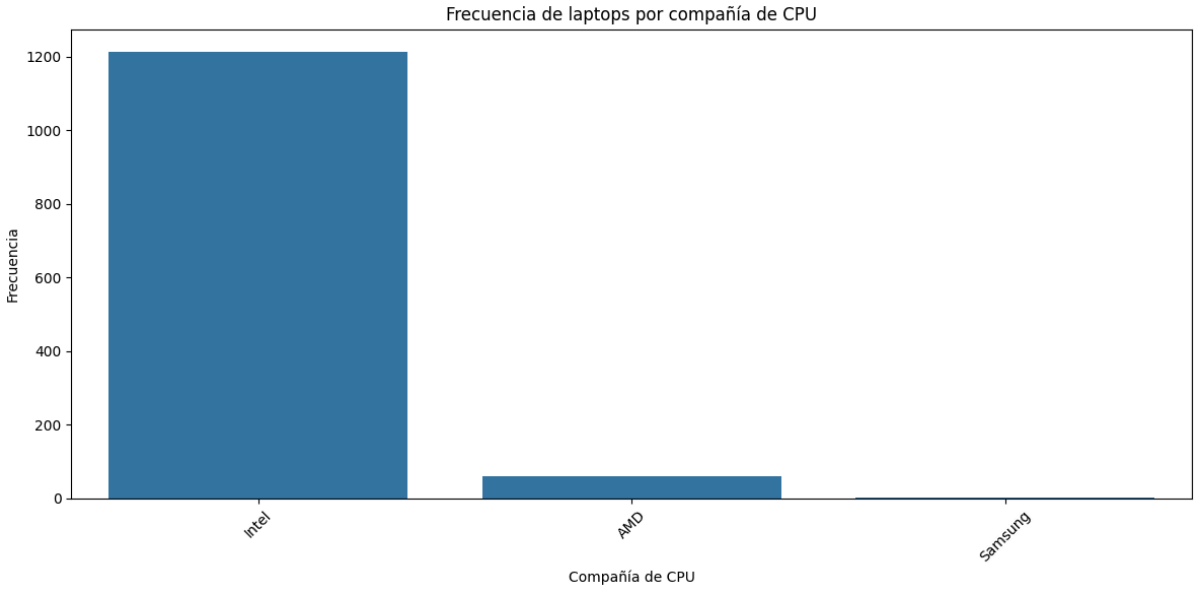
El SO con la mayor frecuencia es Windows 10

Frecuencia por tipo



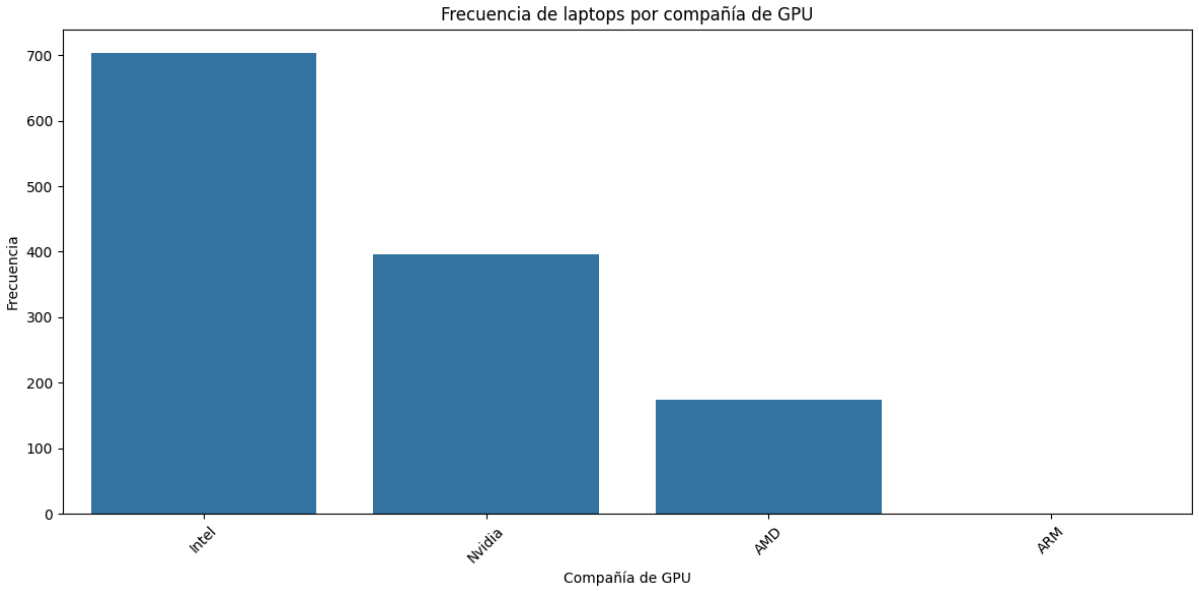
El tipo con mayor frecuencia son los Notebook, seguido de Gaming y Ultrabook

Frecuencia por CPU



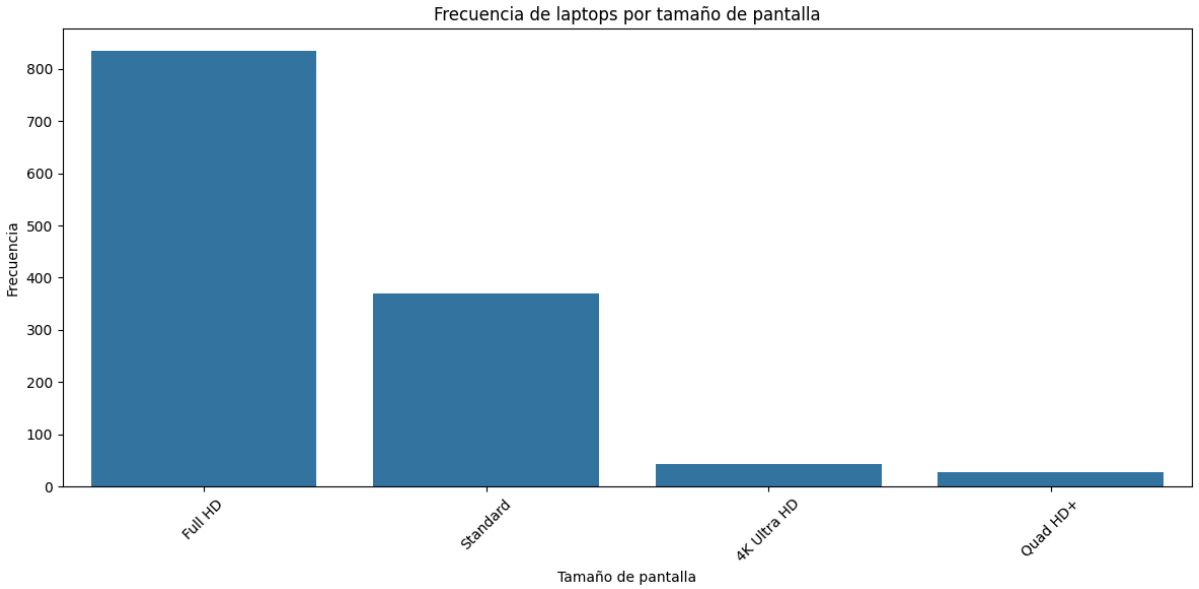
La compañía de CPU con mayor frecuencia es Intel

Frecuencia por GPU



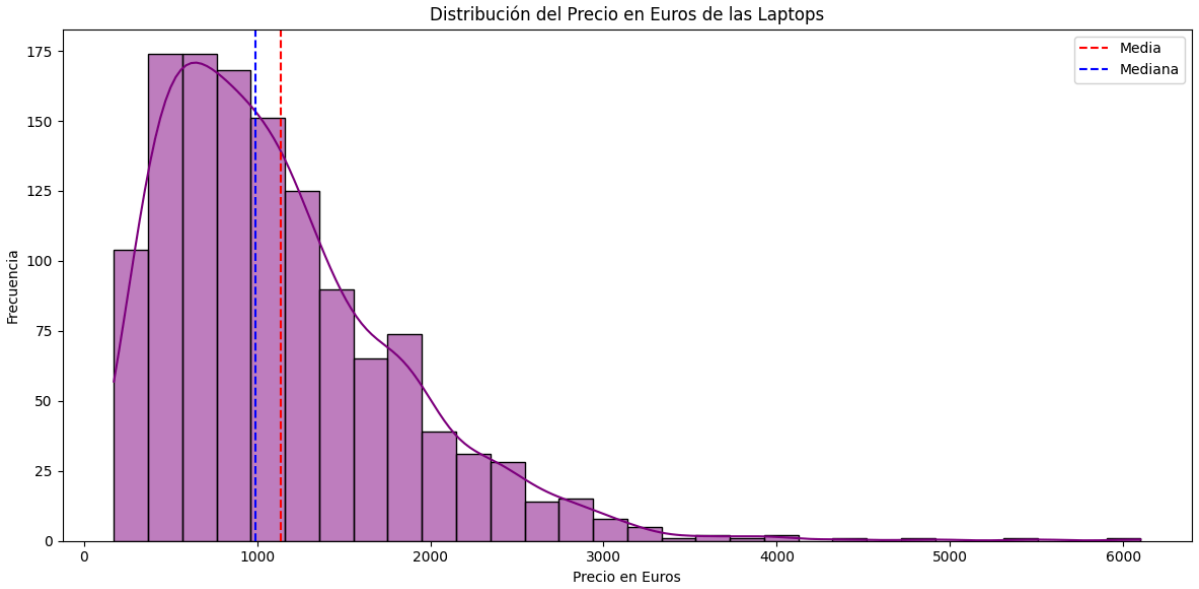
La compañía de GPU con mayor frecuencia es Intel, seguido de Nvidia y AMD

Frecuencia por Tamaño de pantalla



El tamaño de pantalla con mayores frecuencias fueron Full HD y Standard

Distribución del precio en Euros



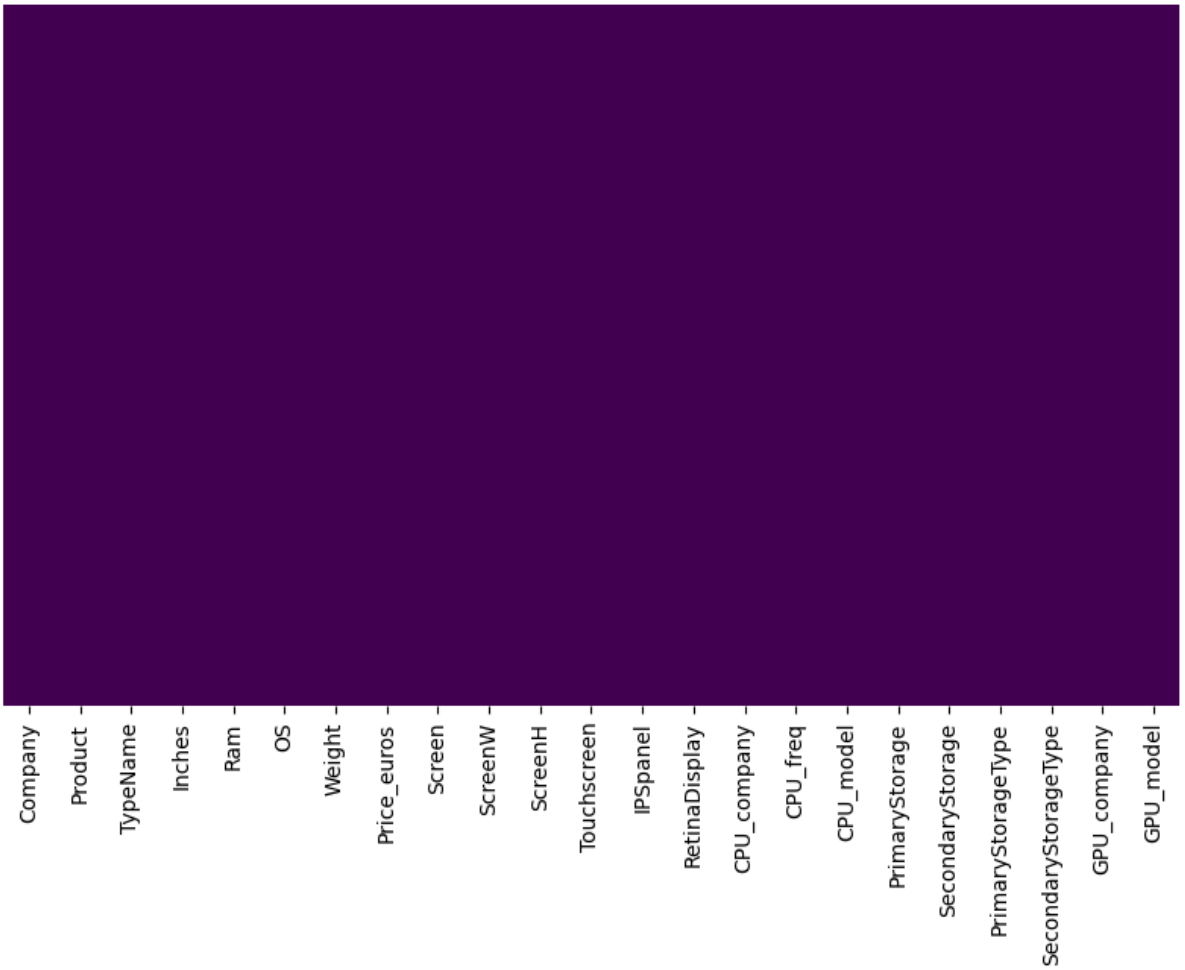
La media está en 1134.969059 y la mediana está 989.00

III. Limpieza y Normalización de Datos

Técnicas utilizadas

1. **Detectar valores atípicos:** Buscar datos que se alejan significativamente del resto de los valores.
2. **Tratar valores atípicos:** Decidir qué hacer con los outliers una vez detectados.
3. **Detectar valores faltantes:** Buscar campos vacíos que no tienen información (`NaN`, `null`, etc.).
4. **SimpleImputer para tratar los valores faltantes:** Una clase de Scikit-learn (`sklearn.impute.SimpleImputer`) que rellena automáticamente los valores faltantes.
5. **Eliminar variables irrelevantes para los modelos:** Eliminar columnas que no aportan valor al análisis o pueden causar ruido.
6. **Variables dummies:** Convertir variables categóricas en columnas numéricas binarias (0 o 1), una por cada categoría posible.

No hay valores nulos en el dataset:



Detectar valores atípicos de las variables numéricas del dataset:

- Calcular los cuartiles (Q1 y Q3) y el Rango Intercuartílico (IQR).

- Establecer los límites inferior y superior utilizando la fórmula:
  Límite inferior = Q1 - 1.5 × IQR
  Límite superior = Q3 + 1.5 × IQR

- Filtrar los datos para identificar los registros que superan los límites establecidos, considerados como valores atípicos

- Comprobación de valores atípicos

```python
# Detectar valores atípicos en el las variables numéricas
for col in numerical_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

    print(f"Cantidad de valores atípicos en {col}: {len(outliers)}")
```

Cantidad de valores atípicos en:

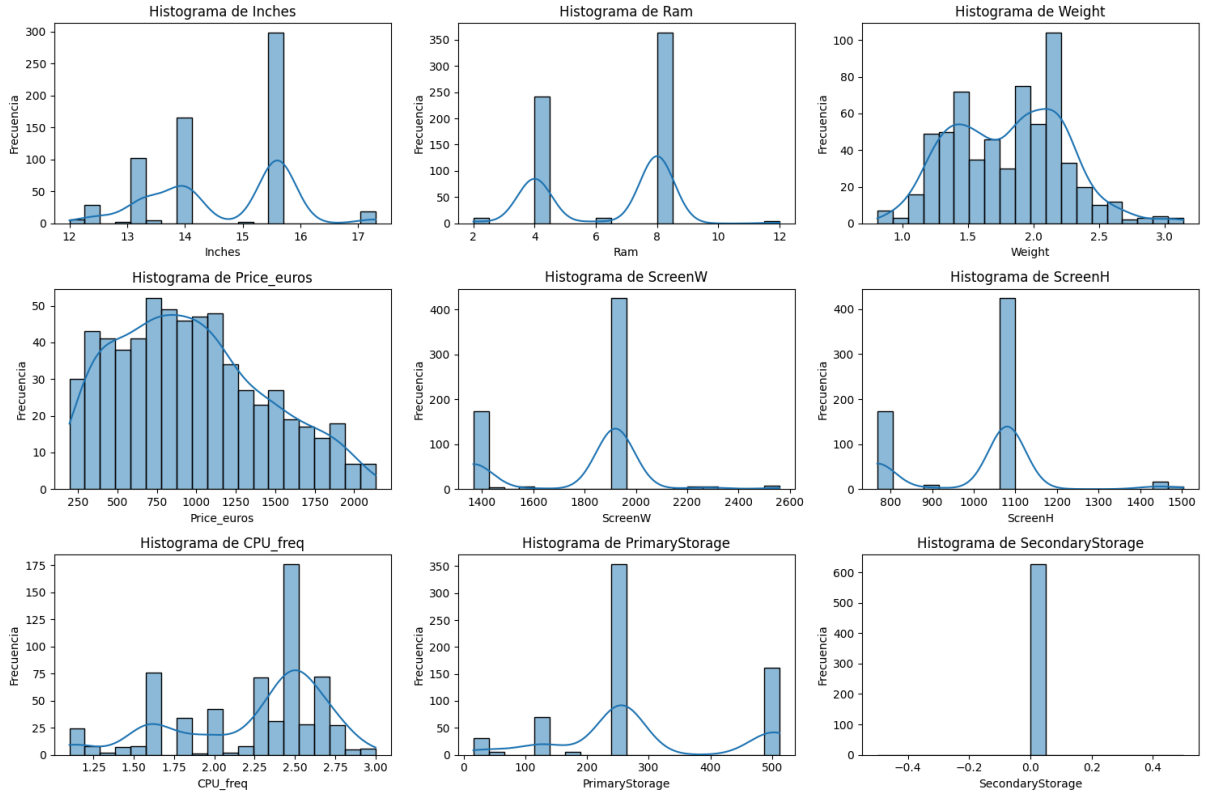
Inches	37
RAM	219
Weight	45
Price_euros	28
ScreenW	437
ScreenH	442
CPU_freq	2
PrimaryStorare	256
SecondaryStorage	208

Tratamiento de los valores atípicos:

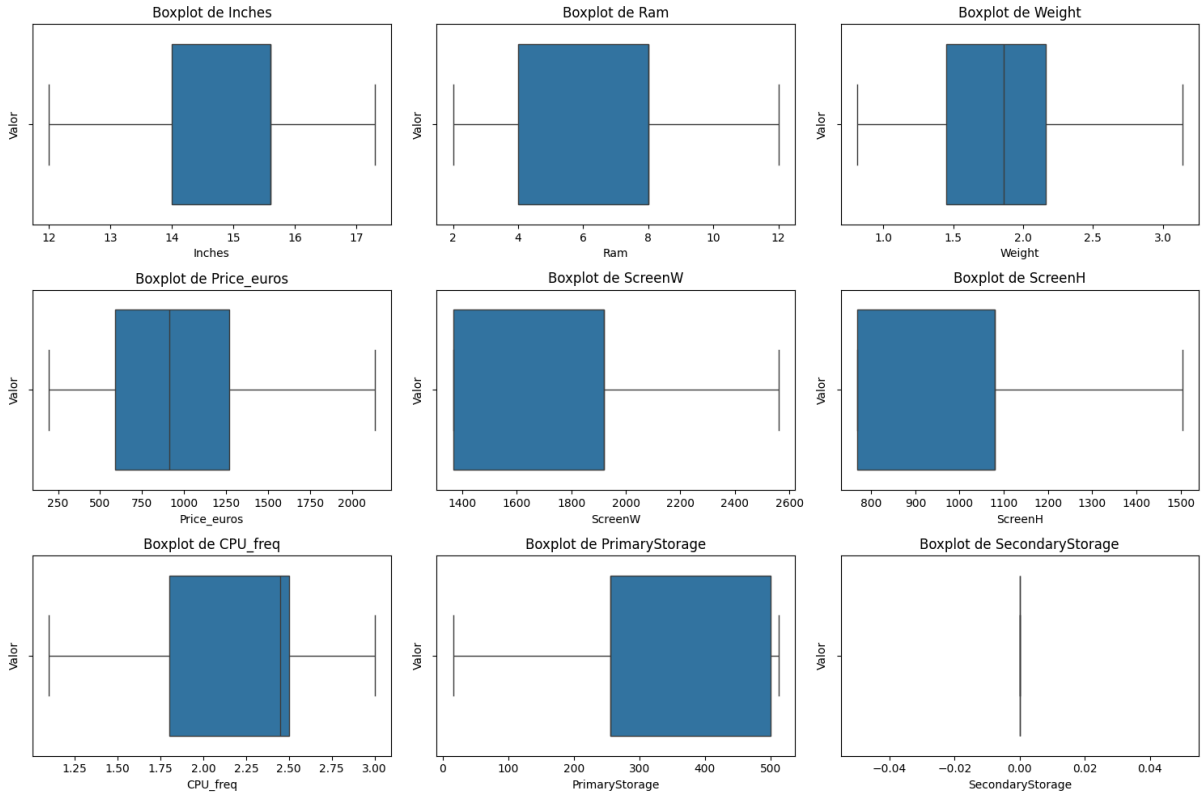
```python
# Tratar valores atípicos en las variables numéricas
df_cleaned = df.copy()

# Eliminar valores atípicos usando el método IQR
for col in numerical_columns:
    Q1 = df_cleaned[col].quantile(0.25)
    Q3 = df_cleaned[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_cleaned = df_cleaned[(df_cleaned[col] >= lower_bound) & (df_cleaned[col] <= upper_bound)]

print(f"Filas originales: {len(df)}")
print(f"Filas después de eliminar outliers: {len(df_cleaned)}")
```

Filas originales: 1275
Filas después de eliminar outliers: 628

Histograma de variables numéricas después de eliminar los valores atípicos:



Boxplots para las variables numéricas después de eliminar los valores atípicos:



La imputación sirve para rellenar los valores faltantes (NaN) en el conjunto de datos, en este caso para las variables numéricas se usa la media y para las variables categóricas se usa la moda, además se eliminan algunas variables que causan ruido o son poco relevantes para los modelos predictivos, por último se transformar las variables categóricas a dummies

```
# Imputación de valores faltantes

# Imputar valores faltantes en columnas numéricas con la media
num_imputer = SimpleImputer(strategy='mean')
df_cleaned[numerical_columns] = num_imputer.fit_transform(df_cleaned[numerical_columns])

# Imputar valores faltantes en columnas categóricas con la moda
cat_imputer = SimpleImputer(strategy='most_frequent')
df_cleaned[categorical_columns] = cat_imputer.fit_transform(df_cleaned[categorical_columns])

# Eliminar columnas irrelevantes para el modelo
df_cleaned.drop(columns=['Product', 'CPU_model', 'GPU_model'], inplace=True)

# Transformar variables categoricas en variables dummy
df_cleaned = pd.get_dummies(df_cleaned, drop_first=True)

df_cleaned.head()
```

Guardar los datos en un nuevo CSV:

```
# Guardar los datos preprocesados
df_cleaned.to_csv("../data/laptop_prices_preprocessed.csv", index=False)
print("Datos preprocesados guardados en 'laptop_prices_preprocessed.csv'")
```

IV. Implementación de Modelos Predictivos

Modelos implementados

1. Regresión Lineal
2. Random Forest
3. MLP Redes Neuronales

Preparación de los datos para los modelos con un 80% de entrenamiento y 20% de prueba, donde la variable objetivo (target) es Price_euros y las variables predictoras son las demás del dataset procesado

```
# Preparación de los datos

# Separar variables predictoras y variable objetivo
X = df.drop('Price_euros', axis=1)
y = df['Price_euros']

# Conjunto de entrenamiento y prueba (80% - 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Escalar las variables predictoras
scaler = StandardScaler()

numeric_cols = ['Inches', 'Ram', 'Weight', 'ScreenW', 'ScreenH', 'CPU_freq', 'PrimaryStorage', 'SecondaryStorage']

X_train[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test[numeric_cols] = scaler.transform(X_test[numeric_cols])

# Mostrar las formas de los conjuntos resultantes
print(f"Datos de entrenamiento: {X_train.shape}, {y_train.shape}")
print(f"Datos de prueba: {X_test.shape}, {y_test.shape}")
```

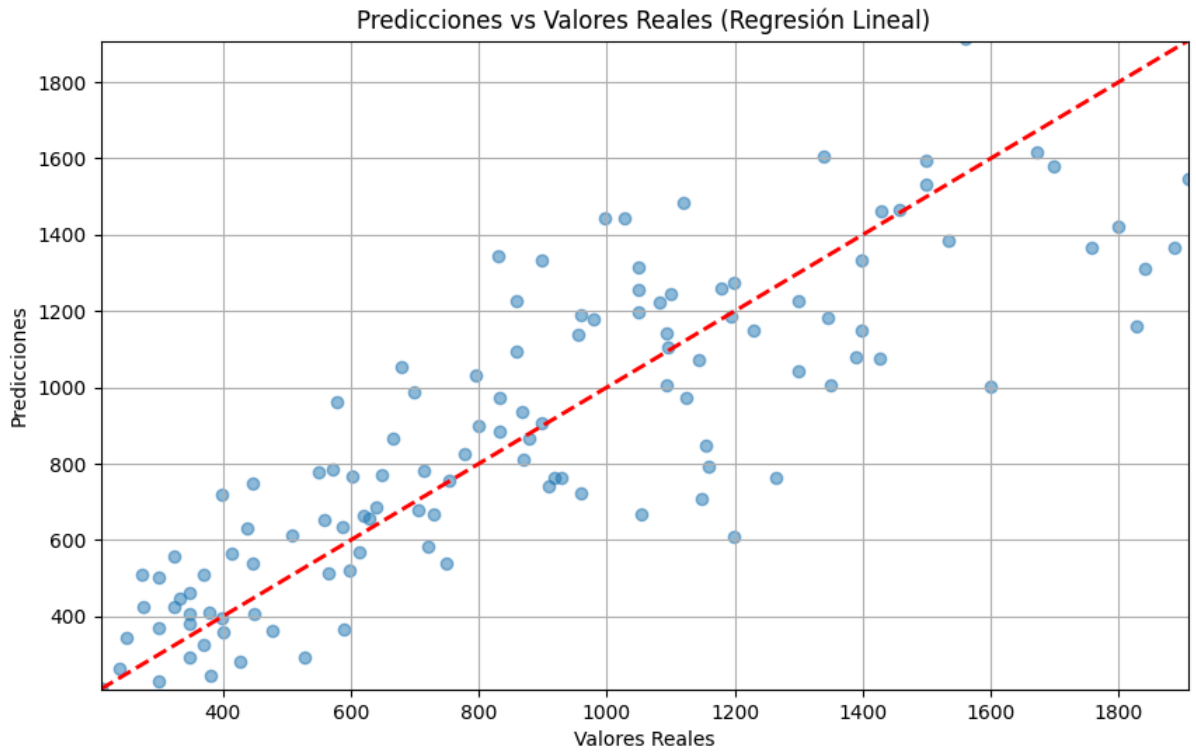
Regresión Lineal:

```
# Regresión lineal
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Evaluación del modelo
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# Imprimir resultados
print(f"Error Cuadrático Medio: {mean_squared_error(y_test, y_pred_lr)}")
print(f"R²: {r2_score(y_test, y_pred_lr)}")
```

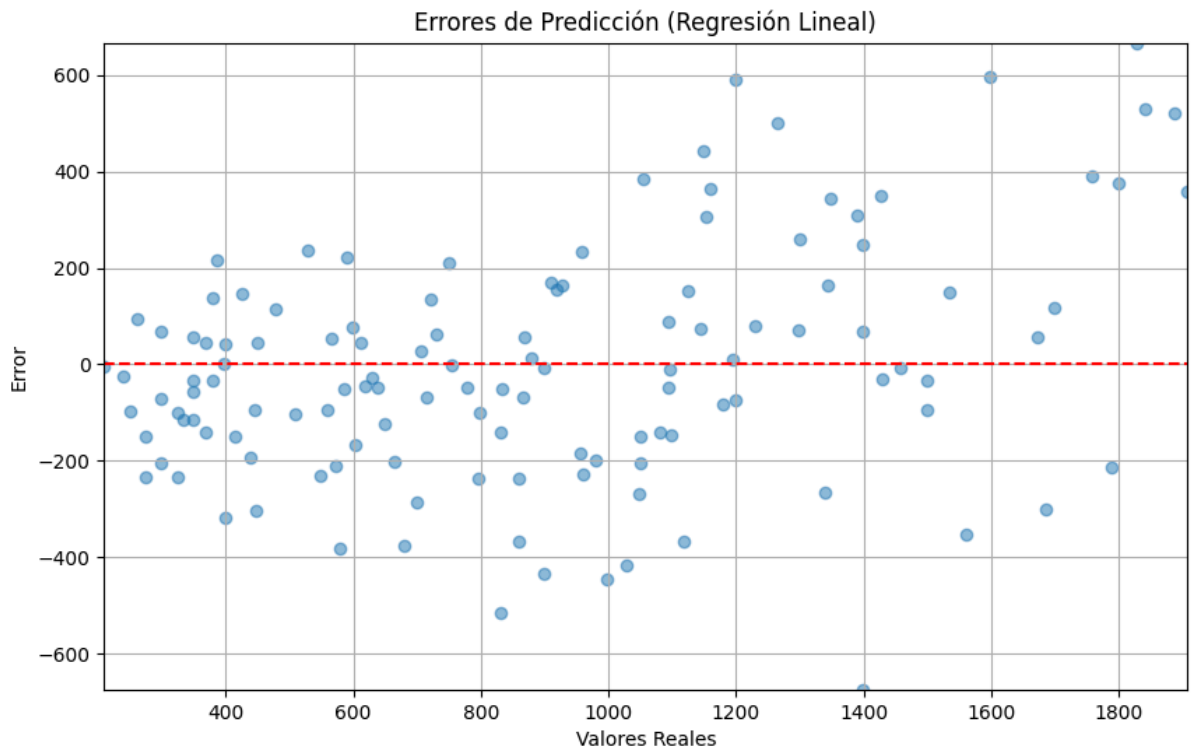
Resultados obtenidos con MSE y R2

Error Cuadrático Medio: 59001.80363525369
R^2: 0.7068927570217696

Predicciones VS Valores Reales del modelo de Regresión Lineal:



Errores de predicción del modelo de Regresión Lineal:

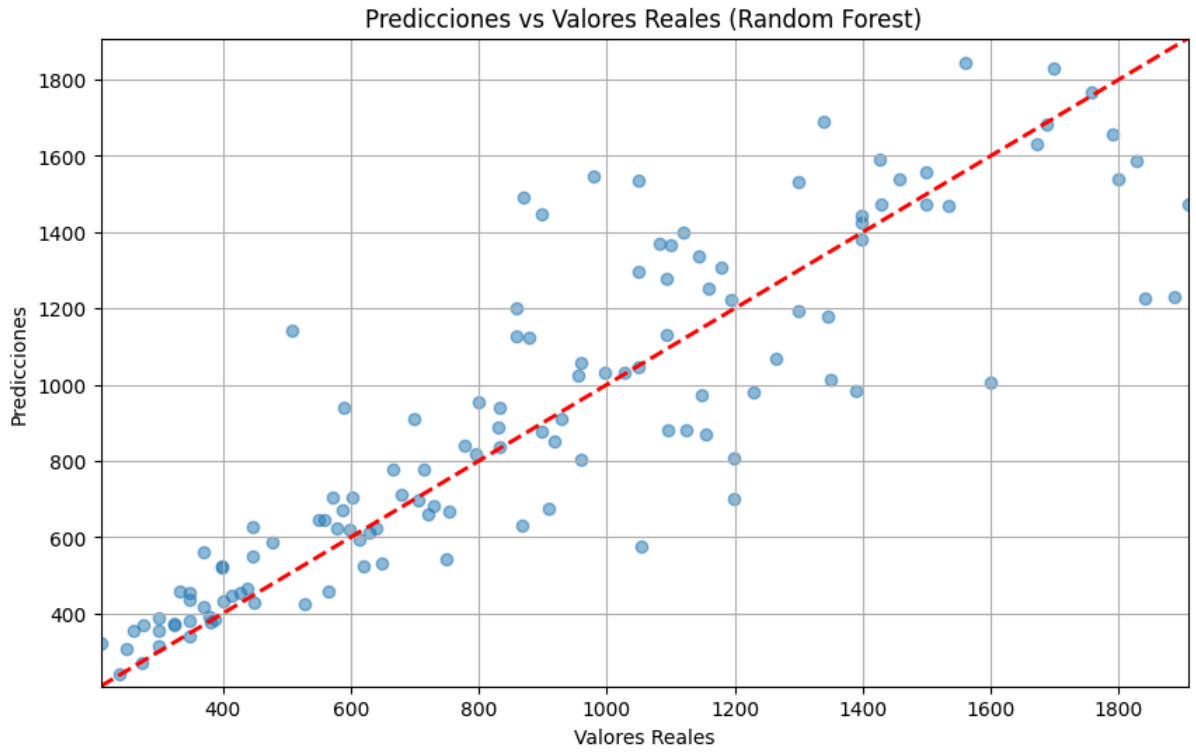


Random Forest:

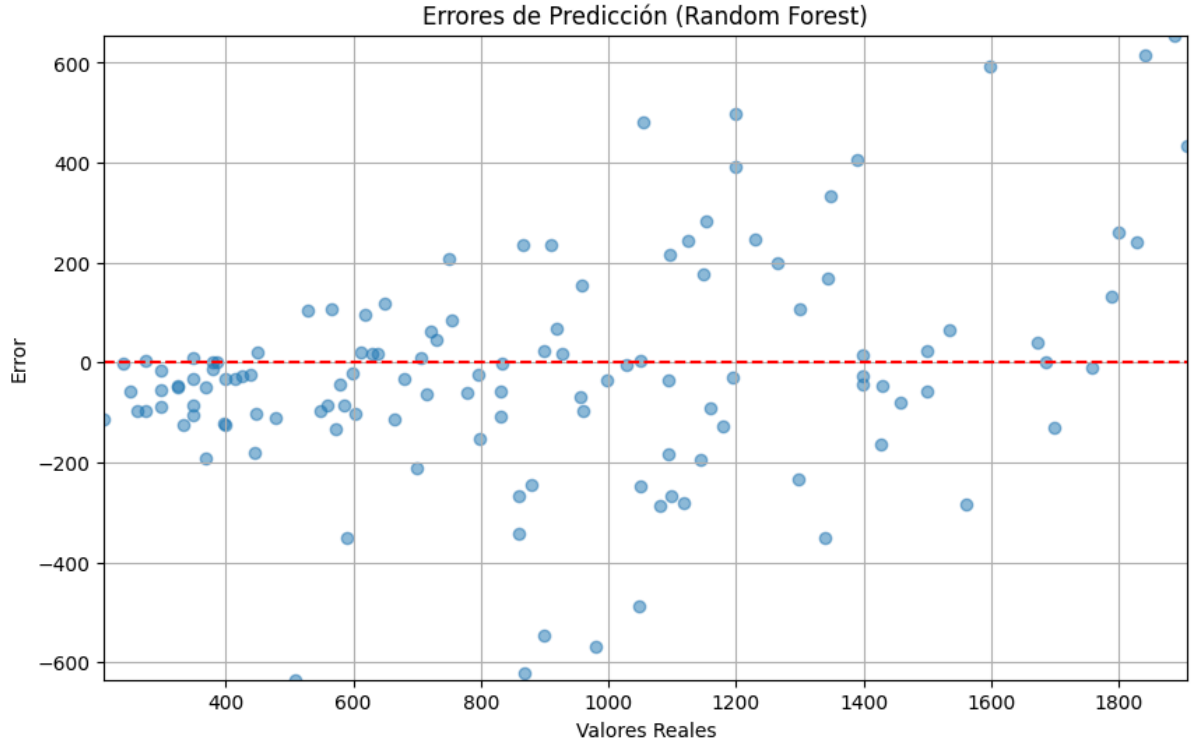
```python
# Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Evaluación del modelo
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# Imprimir resultados
print(f"Error Cuadrático Medio: {mean_squared_error(y_test, y_pred_rf)}")
print(f"R²: {r2_score(y_test, y_pred_rf)}")
```

Resultados obtenidos de MSE y R2
Error Cuadrático Medio: 48655.207093574674
R^2: 0.758292243133879

Predicciones VS Valores Reales del modelo Random Forest:



Errores de predicción del modelo Random Forest:

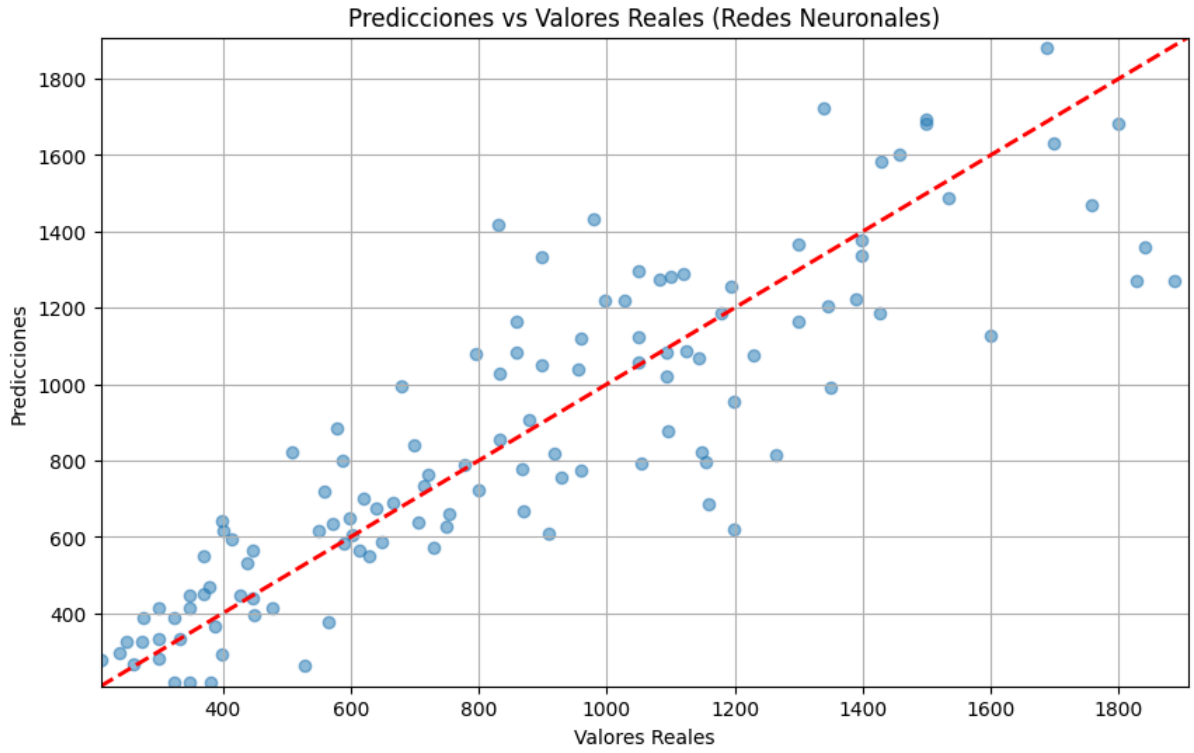


MLP Redes Neuronales:

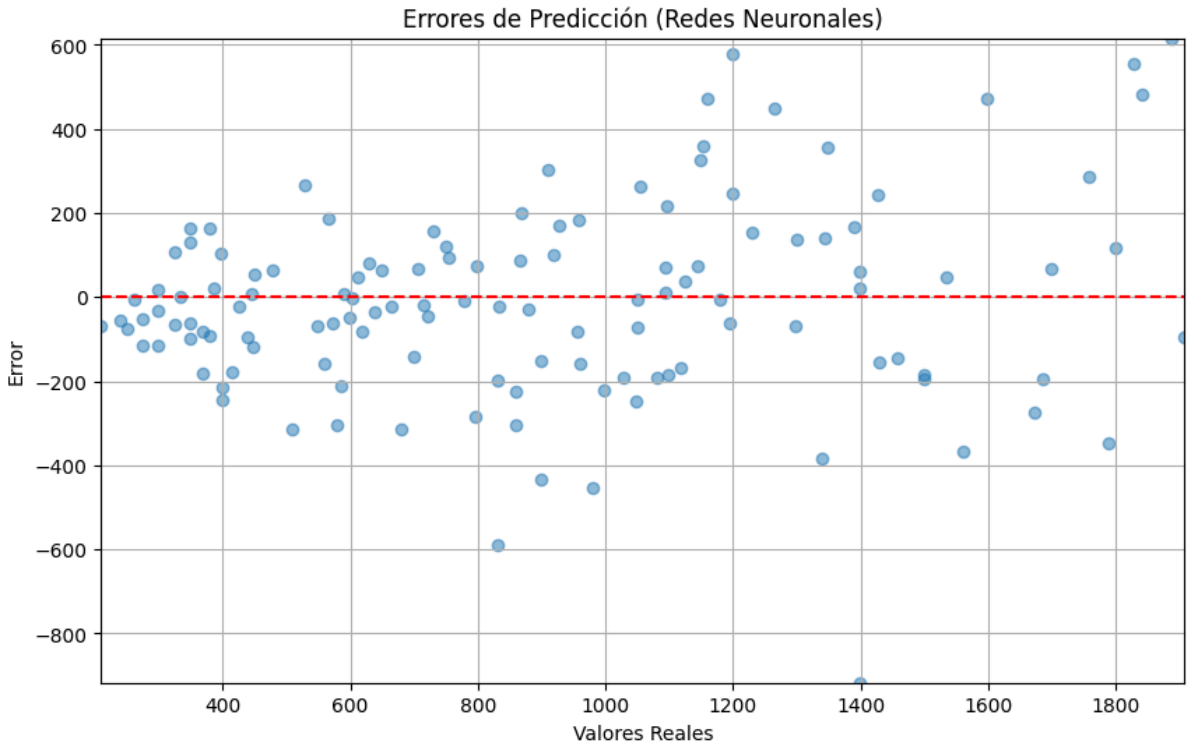
```python
# Redes Neuronales
mlp = MLPRegressor(hidden_layer_sizes=(100,), activation='relu', solver='adam', learning_rate_init=0.01, max_iter=1000)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)

# Evaluación del modelo
mse_mlp = mean_squared_error(y_test, y_pred_mlp)
r2_mlp = r2_score(y_test, y_pred_mlp)

# Imprimir resultados
print(f"Error Cuadrático Medio: {mean_squared_error(y_test, y_pred_mlp)}")
print(f"R²: {r2_score(y_test, y_pred_mlp)}")
```

Resultados obtenidos de MSE Y R2
Error Cuadrático Medio: 52997.0191749102
R²: 0.7367231301529084

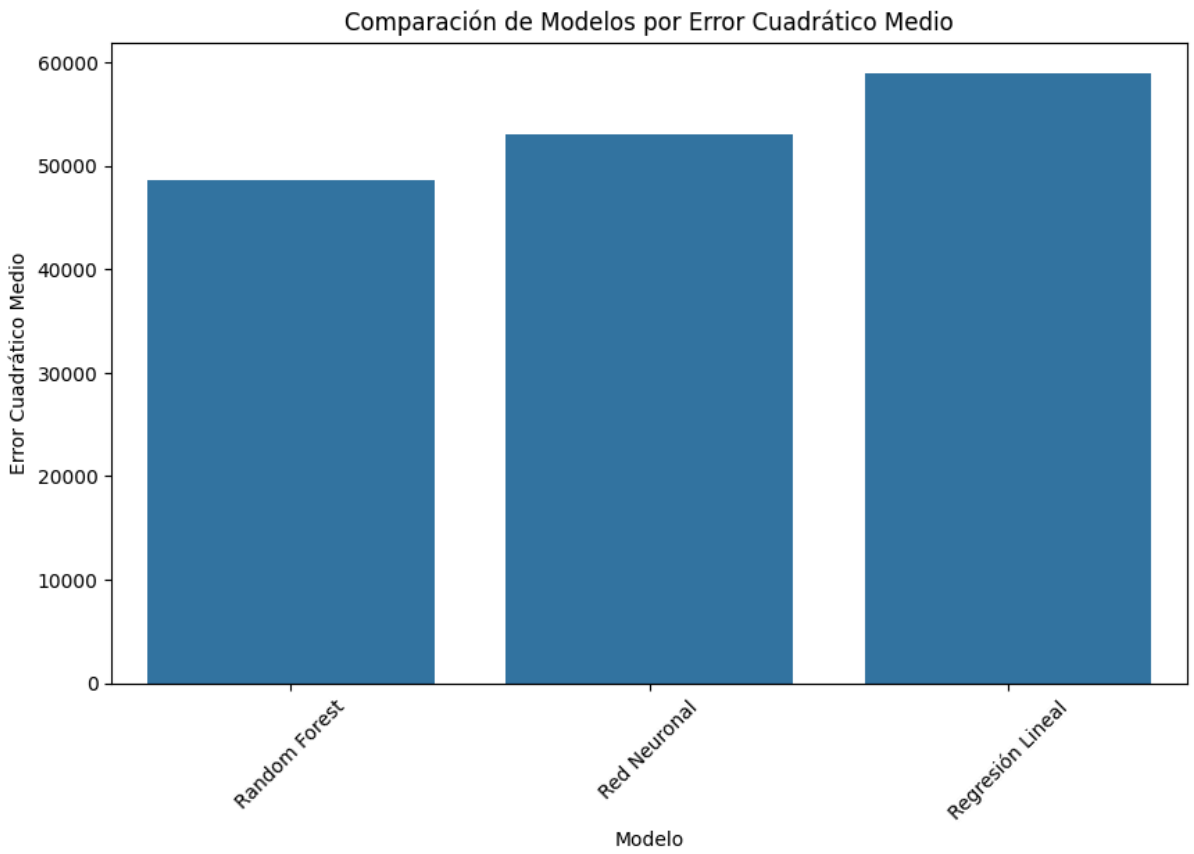
Predicciones VS Valores Reales del modelo MLP Redes Neuronales:



Errores de predicción del modelo MLP Redes Neuronales:



Comparación MSE (Error Cuadrático Medio) de los tres modelos:



En la gráfica se presentan los resultados del error cuadrático medio MSE para tres modelos de regresión utilizados para predecir una variable objetivo a partir de características numéricas:

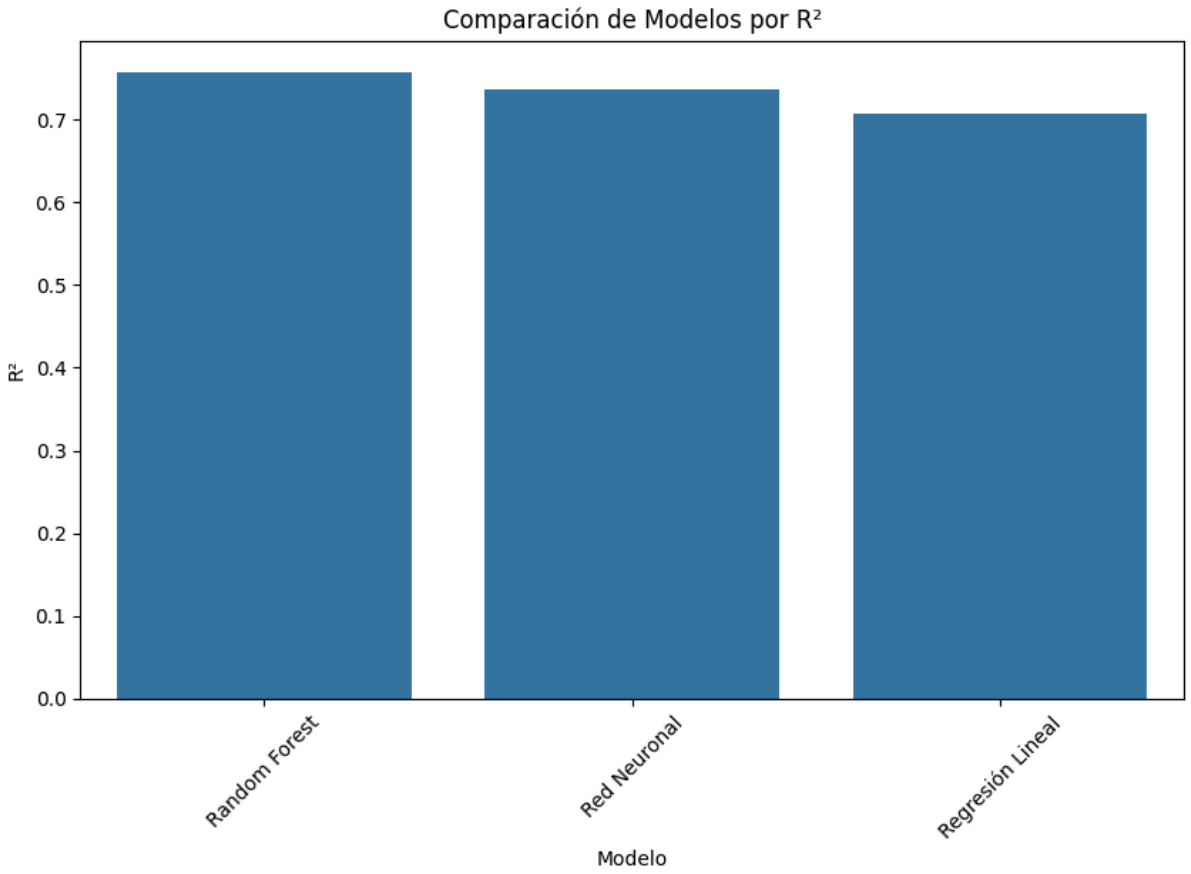
- **Random Forest**
- **Red Neuronal**
- **Regresión Lineal**

Interpretación

- MSE bajo: El modelo comete pocos errores y pequeños, lo que indica mejor rendimiento.
- MSE alto: El modelo tiene errores grandes, es decir, predice mal.

La comparación de los tres modelos muestra que Random Forest obtuvo el valor más bajo de MSE, el cual indica un mejor rendimiento que los demás modelos

Comparación R2 (Coeficiente de determinación):



En la gráfica se presentan los resultados del coeficiente de determinación R2 para tres modelos de regresión utilizados para predecir una variable objetivo a partir de características numéricas:

- **Random Forest**
- **Red Neuronal**
- **Regresión Lineal**

Interpretación

El coeficiente de determinación R2 mide la proporción de la varianza de la variable dependiente que puede ser explicada por las variables independientes del modelo. Su valor varía entre 0 y 1, donde:

- Un valor cercano a **1** indica que el modelo explica bien los datos.
- Un valor cercano a **0** indica que el modelo apenas mejora la predicción con respecto a una media constante.

La comparación muestra que los modelos no lineales (Random Forest y Red Neuronal) ofrecen un mejor ajuste a los datos que la regresión lineal. En particular, Random Forest es el modelo más efectivo en este caso, y sería el candidato más fuerte para ser implementado en un sistema de predicción, a menos que se prioricen la interpretabilidad o la eficiencia computacional, donde la regresión lineal podría seguir siendo útil

Sugerencias de Mejora para los Modelos

Regresión Lineal: Incluir variables polinómicas y eliminar valores atípicos para captar relaciones no lineales y reducir errores.

Random Forest: Ajustar hiperparámetros (número de árboles, profundidad) y seleccionar solo las variables más relevantes para mejorar la precisión.

MLP (Red Neuronal): Probar diferentes arquitecturas ocultas, aumentar el número de épocas, aplicar regularización y usar early stopping para evitar el sobreajuste.

General: Normalizar los datos, eliminar valores atípicos y hacer una selección adecuada de características mejora significativamente todos los modelos.

V. Conclusiones

Este proyecto tuvo como objetivo principal desarrollar un modelo predictivo capaz de estimar el precio de dispositivos electrónicos, utilizando un conjunto de datos con múltiples características técnicas. Para lograr este propósito, se siguió una metodología estructurada que incluyó análisis exploratorio, limpieza de datos, normalización de variables y la implementación de modelos de regresión supervisada.

En la fase inicial, se realizó un análisis descriptivo con el fin de comprender la estructura del dataset, detectar valores atípicos y estudiar la distribución de las variables. Esto permitió identificar relaciones relevantes entre las características técnicas del hardware (como pulgadas de pantalla, memoria RAM, peso, frecuencia del CPU y tipo de almacenamiento) y el precio del dispositivo. Además, se eliminaron valores nulos y se unificaron unidades de medida, lo cual fue esencial para asegurar la calidad de los datos.

Posteriormente, se aplicó una normalización mediante StandardScaler, lo cual fue clave para garantizar que todas las variables tuvieran el mismo peso al momento de entrenar los modelos.

Por último, se entrenaron y compararon tres modelos: Regresión Lineal, Red Neuronal y Random Forest. El modelo de Random Forest demostró ser el más eficaz, obteniendo el mayor coeficiente de determinación R2 **0.758292243133879** y el menor error cuadrático medio de **48655.207093574674**, lo que indica una alta capacidad para explicar la variabilidad del precio y una baja tasa de error en sus predicciones. En contraste, la

Regresión Lineal, aunque simple e interpretable, no logró capturar la complejidad de las relaciones presentes en los datos, obteniendo métricas notablemente inferiores.