



Predicción de precios para laptops

Sthiben Jesus Ruiz Reyes
Ciencia de Datos



Descripción del Proyecto

Este interés sobre la predicción de precios puede ser útil para consumidores, empresas u posibles competidores.

Este proyecto tiene como objetivo desarrollar un modelo de aprendizaje automático para predecir el precio de las laptops en función de sus características. Tome un conjunto de datos en la página Kaggle llamado Laptop Prices la intención de este proyecto es desarrollar un modelo de aprendizaje que pueda predecir los precios de diferentes laptops.

Se exploraron diferentes algoritmos de aprendizaje automático, como regresión lineal y regresión neuronal, para construir el modelo predictivo. Se evaluará el rendimiento de los modelos utilizando métricas como el error cuadrático medio (MSE), la raíz del error cuadrático medio (RMSE) y el coeficiente de determinación (R^2).



Metodología

Se recopiló un conjunto de datos que contiene información sobre las especificaciones técnicas, la marca, el modelo y el precio de las laptops.

Se realizará un análisis y limpieza de datos para eliminar valores faltantes, inconsistencias y errores.

Se explorarán diferentes algoritmos de aprendizaje automático, como regresión lineal y regresión neuronal, para construir el modelo predictivo.

Se utilizará una parte del conjunto de datos para entrenar los modelos y la parte restante para evaluar su rendimiento utilizando métricas como el MSE, el RMSE y el R^2 .

Se seleccionará el modelo con el mejor rendimiento para la predicción del precio de las laptops.

Análisis exploratorio de datos (EDA)



[Análisis Dinámico en Lookerstudio haz click aquí](#)

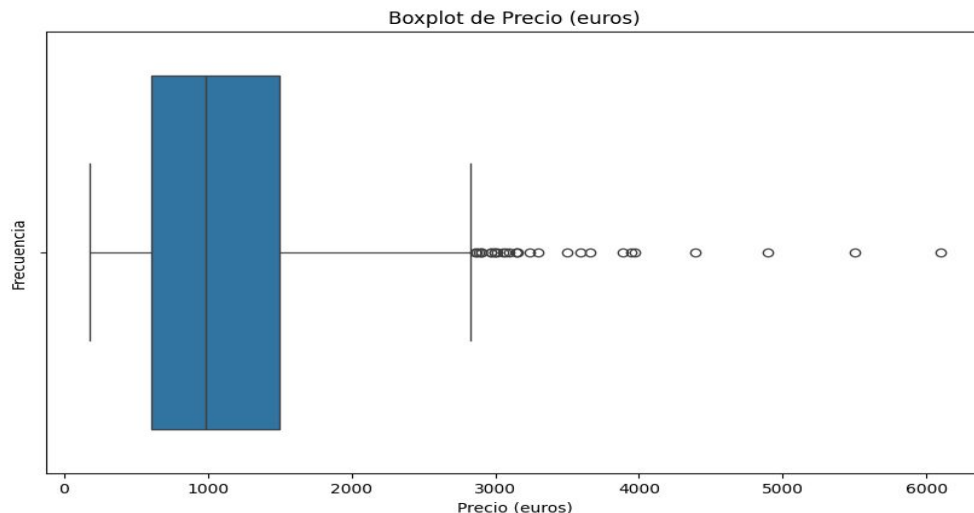


Análisis de datos (EDA)

	Inches	Ram	Weight	Price_euros	ScreenW	ScreenH	CPU_freq	PrimaryStorage	SecondaryStorage
count	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000
mean	15.022902	8.440784	2.040525	1134.969059	1900.043922	1073.904314	2.302980	444.517647	176.069020
std	1.429470	5.097809	0.669196	700.752504	493.346186	283.883940	0.503846	365.537726	415.960655
min	10.100000	2.000000	0.690000	174.000000	1366.000000	768.000000	0.900000	8.000000	0.000000
25%	14.000000	4.000000	1.500000	609.000000	1920.000000	1080.000000	2.000000	256.000000	0.000000
50%	15.600000	8.000000	2.040000	989.000000	1920.000000	1080.000000	2.500000	256.000000	0.000000
75%	15.600000	8.000000	2.310000	1496.500000	1920.000000	1080.000000	2.700000	512.000000	0.000000
max	18.400000	64.000000	4.700000	6099.000000	3840.000000	2160.000000	3.600000	2048.000000	2048.000000

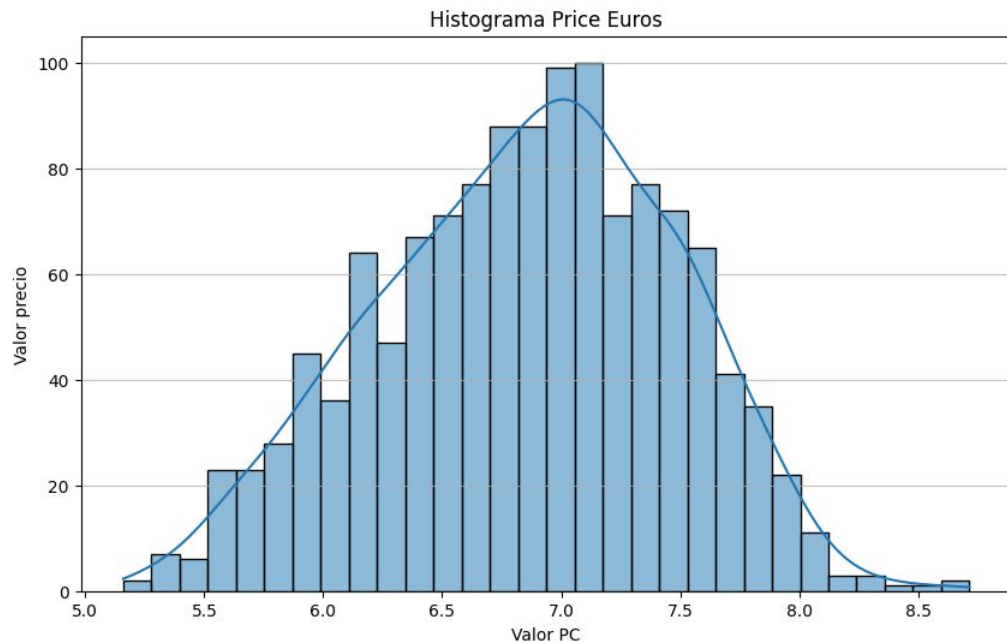
La alta desviación estándar en Price_euros indica que estas variables podrían requerir una transformación

Metodo detección Intercuartílico (IQR)



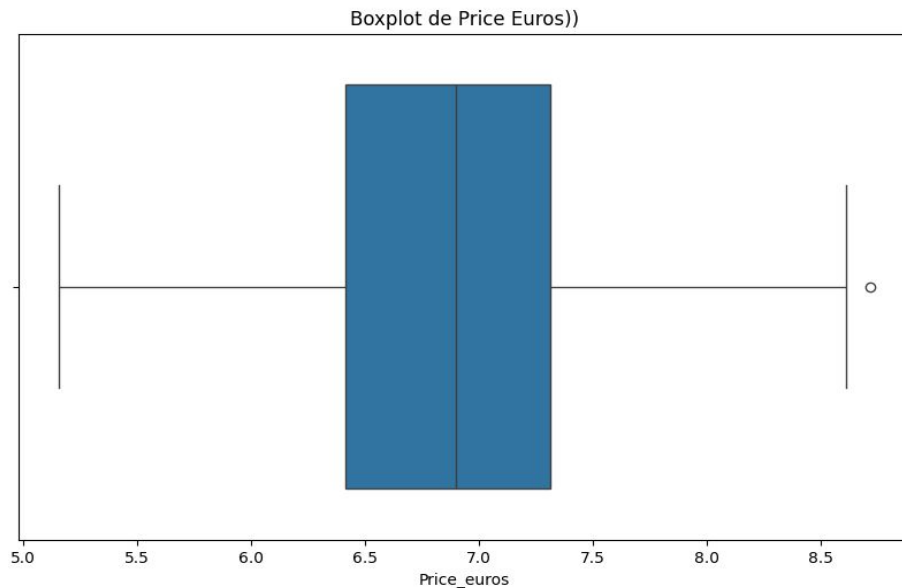
Hay una gran cantidad de valores atípicos por encima del bigote superior, lo que indica la presencia de laptops con precios significativamente más altos que la mayoría. Esto confirma lo que se observó en `df.describe()`

Transformación Logarítmica



El histograma resultante muestra una distribución que se asemeja más a una normal después de la transformación logarítmica

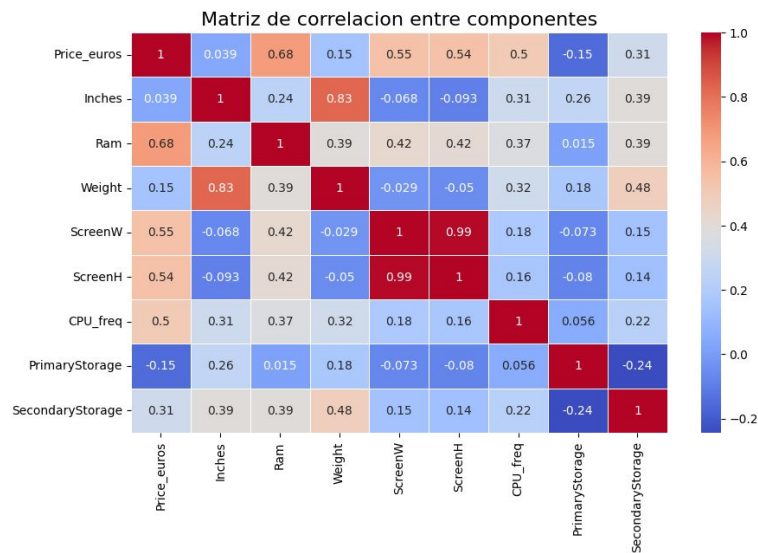
Boxplot Transformación Logarítmica



En este análisis del Boxplot puedo determinar que hay un único valor atípico, este valor representa una laptop con un precio alto, en la escala logarítmica.

La distribución del logaritmo de los precios es relativamente simétrica, a diferencia de la distribución de los precios originales.

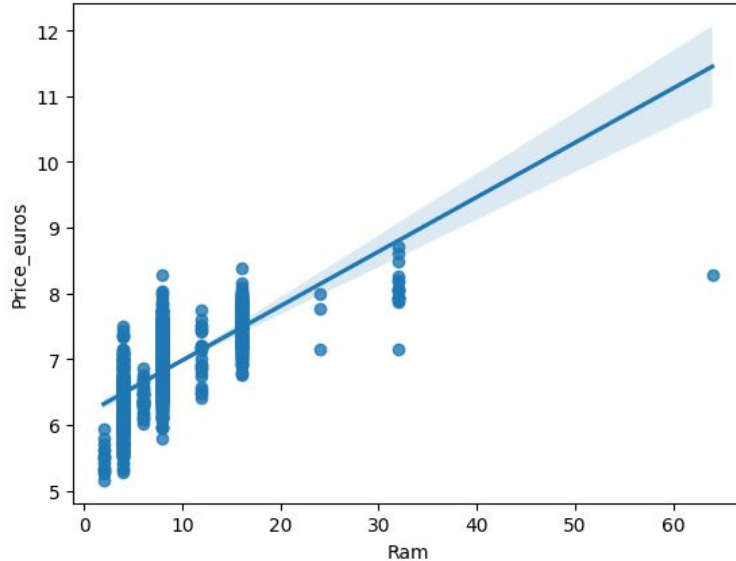
Matriz de Correlación de Pearson



La matriz de correlación es útil para identificar variables que pueden ser redundantes o que pueden estar influyendo mutuamente en un modelo.

En este caso la alta correlación entre ScreenW y ScreenH no es necesario incluir ambas variables en un modelo, ya que aportan información muy similar.

Regresion Lineal Simple



El gráfico confirma la correlación positiva entre 'Ram' y 'Price_euros' observada en la matriz de correlación anterior.

La relación lineal no dice que un modelo de regresión lineal es el adecuado para modelar la relación entre estas variables

Modelado de datos



Creación del pipeline de Entrenamiento

```
[21] from sklearn.preprocessing import OrdinalEncoder

[22] enc = OrdinalEncoder()
df_ordinal_encoder = df.copy()
df_ordinal_encoder[['Company', 'Product', 'TypeName', 'OS', 'Screen',
                    'Touchscreen', 'IPSPanel', 'RetinaDisplay', 'CPU_company',
                    'CPU_model', 'PrimaryStorageType', 'SecondaryStorageType',
                    'GPU_company', 'GPU_model']] = enc.fit_transform(df_ordinal_encoder[['Company', 'Product', 'TypeName', 'OS', 'Screen',
                    'Touchscreen', 'IPSPanel', 'RetinaDisplay', 'CPU_company',
                    'CPU_model', 'PrimaryStorageType', 'SecondaryStorageType',
                    'GPU_company', 'GPU_model']])

df_ordinal_encoder.head(3)
```

	Company	Product	TypeName	Inches	Ram	OS	Weight	Price_euros	Screen	ScreenW	...	RetinaDisplay	CPU_company	CPU_freq	CPU_model	PrimaryStorage	SecondaryStorage	PrimaryStorageType	SecondaryStorageType	GPU_company	GPU_model
0	1.0	300.0	4.0	13.3	8	8.0	1.37	7.200194	3.0	2560	...	1.0	1.0	2.3	40.0	128	0	3.0	2.0	2.0	60.0
1	1.0	301.0	4.0	13.3	8	8.0	1.34	6.801216	3.0	1440	...	0.0	1.0	1.8	40.0	128	0	0.0	2.0	2.0	53.0
2	7.0	50.0	3.0	15.6	8	4.0	1.86	6.354370	1.0	1920	...	0.0	1.0	2.5	46.0	256	0	3.0	2.0	2.0	55.0

3 rows x 23 columns

```
# Creacion del modelo
import statsmodels.api as sm
x_train_2 = sm.add_constant(x_train_2, prepend=True)
modelo = sm.OLS(endog = y_train_2,
                 exog = x_train_2)
modelo = modelo.fit()
print(modelo.summary())
```

OLS Regression Results

Dep. Variable:	y	R-squared:	1.000
Model:	OLS	Adj. R-squared:	1.000
Method:	Least Squares	F-statistic:	5.711e+27
Date:	Sat, 18 Jan 2025	Prob (F-statistic):	0.00
Time:	22:05:12	Log-Likelihood:	29282.
No. Observations:	1020	AIC:	-5.854e+04
Df Residuals:	1009	BIC:	-5.849e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	4.374e-14	4.39e-14	0.997	0.319	-4.24e-14	1.3e-13
Company	1.023e-16	6.58e-16	0.155	0.877	-1.19e-15	1.39e-15
Weight	3.886e-15	5.11e-15	0.760	0.447	-6.15e-15	1.39e-14
Price_euros	1.0000	5.41e-15	1.85e+14	0.000	1.000	1.000
ScreenW	-5.421e-16	5.08e-17	-10.671	0.000	-6.42e-16	-4.42e-16
ScreenH	8.786e-16	8.86e-17	9.912	0.000	7.05e-16	1.05e-15
IPSPanel	1.268e-14	6.21e-15	2.042	0.041	4.93e-16	2.49e-14
PrimaryStorage	4.833e-17	7.93e-18	6.094	0.000	3.28e-17	6.39e-17
SecondaryStorage	1.076e-16	2.26e-17	4.756	0.000	6.32e-17	1.52e-16
SecondaryStorageType	-4.718e-16	1.33e-14	-0.036	0.972	-2.65e-14	2.55e-14
GPU_model	1.878e-16	1.05e-16	1.796	0.073	-1.74e-17	3.93e-16

Omnibus:	74.221	Durbin-Watson:	0.753
Prob(Omnibus):	0.000	Jarque-Bera (JB):	104.511
Skew:	-0.589	Prob(JB):	2.02e-23
Kurtosis:	4.035	Cond. No.	3.98e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified

[2] The condition number is large, 3.98e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Modelo de regresión OLS

En la segunda regresión del modelo OLS, el R-squared 1.000, el coeficiente de determinación R^2 es 1.000, es evidente un sobreajuste (overfitting) o un error en la configuración.

Modelo Predictivo con Red Neuronal Multicapa

```
[41] parametros = {'hidden_layer_sizes': [(10), (10,10), (8, 8)],
                  'alpha': np.logspace(-3,3,3),
                  'learning_rate_init': [0.001, 0.1]}

[42] modelo_regresion = RandomizedSearchCV(
    estimator = MLPRegressor(solver = 'lbfgs', max_iter=1000),
    param_distributions = parametros,
    n_iter= 50,
    scoring = 'neg_mean_squared_error',
    n_jobs = multiprocessing.cpu_count()-1,
    cv = 3,
    verbose = 0,
    random_state = 61212024,
    return_train_score = True,
    error_score='raise'
)

modelo_regresion.fit(X = x_train_3, y = y_train_3)
resultados = pd.DataFrame(modelo_regresion.cv_results_)

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
self.n_iter = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.11/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

RandomizedSearchCV es una técnica eficiente para explorar un espacio de búsqueda de hiperparámetros grande.

Modelo de Regresión Bosque Aleatorio

Modelo de validación

```
[61] from sklearn.metrics import mean_squared_error, r2_score
      pred_y_4 = model.predict(features_validate_x)
      mse = mean_squared_error(validate_y, pred_y_4)
      rmse = mse**0.5 # O usar np.sqrt(mse)
      r2 = r2_score(validate_y, pred_y_4)
      print(f"MSE: {mse}")
      print(f"RMSE: {rmse}")
      print(f"R²: {r2}")
```

```
➡ MSE: 0.08565322008017036
   RMSE: 0.2926657138787705
   R²: 0.7844331439599459
```

Error Cuadrático Medio (MSE): Mide el promedio de los errores al cuadrado. Es sensible a los valores atípicos, si el número es Menor MSE indica un mejor modelo.

Raíz del Error Cuadrático Medio (RMSE): Es la raíz cuadrada del MSE. También es sensible a los valores atípicos. Menor RMSE indica un mejor modelo.

Coefficiente de Determinación (R^2): Mide la proporción de la varianza en la variable dependiente que es explicada por el modelo. Varía entre 0 y 1. Un R^2 cercano a 1 indica un buen ajuste, mientras que un R^2 cercano a 0 indica un mal ajuste.

```

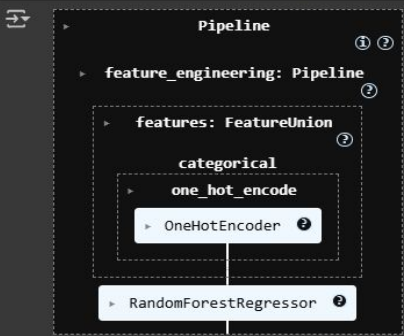
  ▾ Construcción del pipeline final

[62] final_inference_pipeline = Pipeline([
      ("feature_engineering", clone(feature_engineering_pipeline)),
      ("model", RandomForestRegressor(n_estimators=100))
  ])

[63] final_training_dataset = pd.concat([train_x_4, validate_x])
      final_training_response = pd.concat([train_y_4, validate_y])

[64] final_inference_pipeline.fit(final_training_dataset, final_training_response)

```



Pipeline

Los pipelines son una herramienta fundamental en scikit-learn para organizar y automatizar el flujo de trabajo de machine learning.

Evitan errores comunes (como aplicar transformaciones diferentes a los datos de entrenamiento y prueba) y simplifican el despliegue del modelo.

Conclusiones

```
predictions_df.sort_values(by='predicted_price', ascending=False).head(20)
```

	predicted_price	actual_price	Company
424	7.985656	7.937017	Dell
835	7.922047	7.928273	Dell
1231	7.857028	8.160232	Razer
578	7.817501	7.911691	MSI
693	7.793093	7.859413	Lenovo
301	7.790263	7.660114	Asus
811	7.788860	7.789455	MSI
121	7.788860	7.714901	MSI

Tras varios modelos de regresión, que abarcan desde la carga y limpieza de los datos hasta la construcción y evaluación he encontrado un modelo de regresión basado en Random Forest, he obtenido resultados prometedores en la predicción de precios de laptops. El análisis exploratorio de datos (EDA) inicial nos permitió comprender las distribuciones de las variables, identificar posibles valores atípicos y descubrir relaciones entre las características y el precio, la aplicación de una transformación logarítmica a la variable objetivo ('Price_euros') resultó crucial para abordar la asimetría observada en su distribución original, mejorando la adecuación del modelo de regresión lineal a los datos transformados.



Conclusión Final Modelo Random Forest

El modelo de Random Forest frente a los otros modelos fue el que mejor desempeño predictivo tubo, tras ser ajustado y optimizado mediante una búsqueda aleatoria de hiperparámetros (RandomizedSearchCV) y validación cruzada, demostró un rendimiento notable en el conjunto de validación, alcanzando un coeficiente de determinación (R^2) de 0.7844, este valor indica que el modelo predice aproximadamente el 78.4% de los datos

Link repositorio github

https://github.com/sthiben/prediccion_precios_laptops.git