Predictive Machine Learning System for Computer Price Estimation & Explainability

A PREDICTIVE ML SYSTEM FOR ESTIMATING THE PRICE OF COMPUTER CONFIGURATIONS

Group 6

Team & Workflows

Data Preparation

Model Exploration

Addtional models &Validation

Final model & Explainability

UI Development

NLP Integration &Interface

Documentation requirements

Dataset Overview & Key Challenges

8064×135 Spanish-language marketplace dataset

Numerical values embedded in text (e.g., '2,5 GHz', '1 TB')

Missing-Not-At-Random patterns: desktops lack batteries, budget laptops lack benchmarks

Redundant representations of same concept (screen size, CPU frequency)

Data raw to clean transformation

Raw Field	Raw Example	Cleaned Value
15,6 pulgadas	15,6 pulgadas	15.6
1.024,00 € - 2.499,00 €	1.024,00 €	1024.00
512GB SSD NVMe	512GB SSD	512
Turbo máx. 4,3 GHz	4,3 GHz	4.3

The dataset required not just cleaning, but conceptual understanding of hardware

Data Cleaning Strategy

Built extraction functions to normalise frequencies, storage sizes, screen dimensions

Harmonised units (TB→GB, commas→decimal points, cm→inches)

Reduced from ~139 messy raw columns to ~44 structured hardware fields

Subsystem-based cleaning:

CPU
→ GPU
→ RAM
→ Storage
→ Display
→ Physical
→ Market fields

Stage	Columns	Notes
Raw dataset	135	Untidy text, duplicates
After cleaning	44	All numeric or categorical
After encoding	151	Model-ready numerical matrix
After feature selection	17	Only informative features

The core of the project's accuracy comes from engineering high-quality, interpretable features which result in clear impact on the dataset as shown in the table

Data Cleaning & Missingness Patterns

```
=== Missing % for ssd_gb by category ===

Grouped by Tipo de producto:
Tipo de producto
Kit ampliación PC       1.000000
Barebone                0.793651
Netbook                 0.428571
Chromebook              0.381579
Ultrabook               0.375000
PC completo             0.333333
Portátil 3D             0.333333
Mini PC                 0.270270
Portátil convertible    0.224719
Portátil profesional    0.222772
Portátil gaming         0.199229
Portátil multimedia     0.161841
PC de oficina           0.130856
PC multimedia           0.125714
Workstation             0.123188
PC gaming               0.044109
Thin Client             0.040000
Name: ssd_gb, dtype: float64

Grouped by Tipo:
...
Tipo
Desktop    0.741071
Laptop     0.018105
Name: weight_kg, dtype: float64
```

- Missing values follow structural patterns, not randomness
- Desktops show 74% missing weight — this is because sellers rarely include physical specs
- Barebones kits have 100% missing SSD capacity — storage is optional
- Laptops have very low weight missingness — buyers expect this info

Structural missingness (value doesn't exist by design):

- hdd_gb: Missing = no HDD → fill with 0, add has_hdd flag
- battery_wh: Desktops have no battery → fill with 0, add has_battery flag

Unknown values (data not recorded):

- height, weight_kg: Median imputation by product type + missingness flags
- cpu_mark, gpu_mark: Model-based imputation using RandomForest
- Drop very sparse columns

Missing Value Strategy

Structural imputation: desktops have no battery → set to 0

Median imputation by device type for physical attributes

Model-based imputation using RandomForest to infer CPU/GPU marks where missing

Added missingness indicators (“battery_missing”, “height_missing”)

Supports prediction when the user leaves fields blank — required by the assignment

- CPU/GPU benchmarks are missing mostly in low-end devices → imputing them prevents price underestimation
- Battery_wh missing always implies desktop, which we converted to 0
- Missing dimensions (height/weight) were filled using median per Tipo (Laptop/Desktop) to preserve category structure\

Missingness carries semantic meaning; the model learns from it.

Feature Engeneering & Encoding

```
Data columns (total 49 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Título                   7927 non-null   object
 1   Precio_Rango             7927 non-null   object
 2   Pantalla_Luminosidad     7927 non-null   object
 3   Procesador_Procesador    7927 non-null   object
 4   Gráfica_Tarjeta gráfica  7927 non-null   object
 5   Tipo                     7927 non-null   object
 6   screen_size_inch         7927 non-null   float64
 7   cpu_base_ghz             7927 non-null   float64
 8   cpu_turbo_ghz            7927 non-null   float64
 9   cpu_tdp_w                7927 non-null   float64
 10  ram_gb                   7927 non-null   float64
 11  ssd_gb                   7927 non-null   float64
 12  hdd_gb                   7927 non-null   float64
 13  vram_gb                  7927 non-null   float64
 14  height                   7927 non-null   float64
 15  width                    7927 non-null   float64
 16  depth                    7927 non-null   float64
 17  weight_kg                7927 non-null   float64
 18  chassis_color            7927 non-null   object
 19  battery_wh               7927 non-null   float64
...
 47  cpu_vendor               7927 non-null   object
 48  gpu_vendor               7927 non-null   object
dtypes: float64(19), int64(7), object(23)
```

Feature engineering mattered more than the model family

- **Fuzzy matching with external CPU/GPU benchmark datasets → cpu_mark, gpu_mark**
- **First-label simplification for multi-label categories**
- **Vendor extraction (Intel/AMD/NVIDIA/Apple)**
- **Frequency encoding of high-cardinality columns (Series, Brand)**
- **One-hot encoding for low-cardinality categorical fields**
- **Engineered performance ratios: gpu/cpu ratio, storage ratios**

Feature Engeneering & Encoding

Fuzzy matching with external CPU/GPU benchmar
→ cpu_ma

dinality

First-label
categories

nality

Vendor ex
(Intel/AMD

Scaling

We selected only the continuous hardware-related variables for standardization:

- CPU/GPU benchmark scores, clock speeds, core/thread counts
- RAM size and frequency, SSD and HDD capacities
- Screen size, physical dimensions, weight, brightness, refresh rate
- Battery capacity, numeric counts (offers_num, os_bits)

These features have very different ranges and units, so scaling helps models that are sensitive to feature magnitude.

Feature engineering mattered more than the model family

Feature Selection

- Used **Random Forest** on unscaled data to get ‘feature_importances’
- Kept variables with importance ≥ 5% of max → 17 features.
- Force-kept key specs: RAM, CPU/GPU benchmarks, SSD size, screen size.
- Added interaction features (total_performance, total_storage, perf_per_core, etc.)
- Grouped selected features into UI blocks (Memory, CPU, GPU, Display, Physical, Power) and ranked blocks by total importance

Block	Features
Memory & Storage	ram_gb, ssd_gb, hdd_gb, ram_freq_mhz
Processor	cpu_mark, cpu_cores, cpu_turbo_ghz
Graphics	gpu_mark, vram_gb, is_gaming
Display	screen_size_inch, refresh_rate_hz
Physical	height, width, weight_kg
Power	battery_wh, has_battery

Result: A compact, domain-aware feature set that keeps the strongest price drivers.

Higher importance = more impact on price = should be collected first

Model Validation Strategy

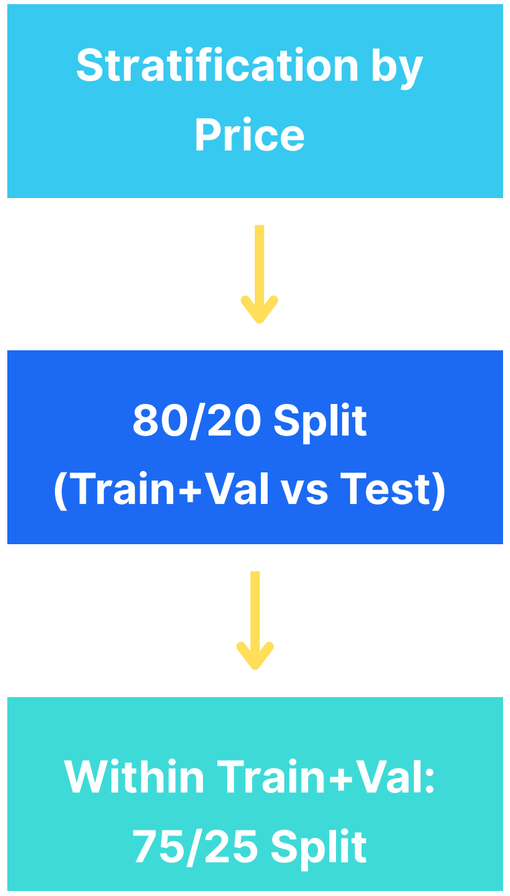
Validation Strategy

The approach ensured balanced representation of different price ranges across all splits, which is crucial given the skewed nature of the target variable. By reserving 20% of the data for final testing and using 60% for training and 20% for validation, we maintained a clear separation between tuning and evaluation phases. Stratified validation allowed for fair and reliable comparison across algorithms and highlights their real-world predictive strength.

price_num
target variable
(numeric price)

A robust validation strategy by **splitting the cleaned dataset into training, validation, and test** sets using stratification based on price quantiles.

Three Way Stratified Split



Model Exploration

	Model	MAE	RMSE	R2_val	R2_cv_mean	R2_cv_std
1	GradientBoosting	342.649311	573.485327	0.716848	0.654554	0.033694
0	RandomForest	328.245928	615.404509	0.673941	0.702277	0.041281
3	XGBoost	325.667176	620.890166	0.668102	0.696091	0.064594
2	ExtraTrees	330.822773	654.513398	0.631182	0.667281	0.066521
8	KNN	369.904232	671.098478	0.612253	0.549843	0.047412
6	Lasso	447.919084	712.483101	0.562957	0.497915	0.051630
5	Ridge	447.942217	712.502052	0.562933	0.497910	0.051650
4	LinearRegression	447.966013	712.515468	0.562917	0.497904	0.051644
7	ElasticNet	447.255617	713.186523	0.562093	0.495328	0.051901
9	SVR	442.676372	807.562470	0.438529	0.347372	0.039810

- Compared 10 models on the validation split using the same selected features:
- **Tree ensembles** (RandomForest, ExtraTrees, GradientBoosting, XGBoost*) on unscaled data
- **Linear & distance-based** (Linear, Ridge, Lasso, ElasticNet, kNN, SVR) on scaled data

Used both scaled and unscaled datasets depending on algorithm

Cross-validated on validation split (20%)

Hyperparameter Tuning

Metric	Before	After	Change
R^2	0.640	0.714	+11.5%
RMSE	746	665	-81
MAE	426	357	-69

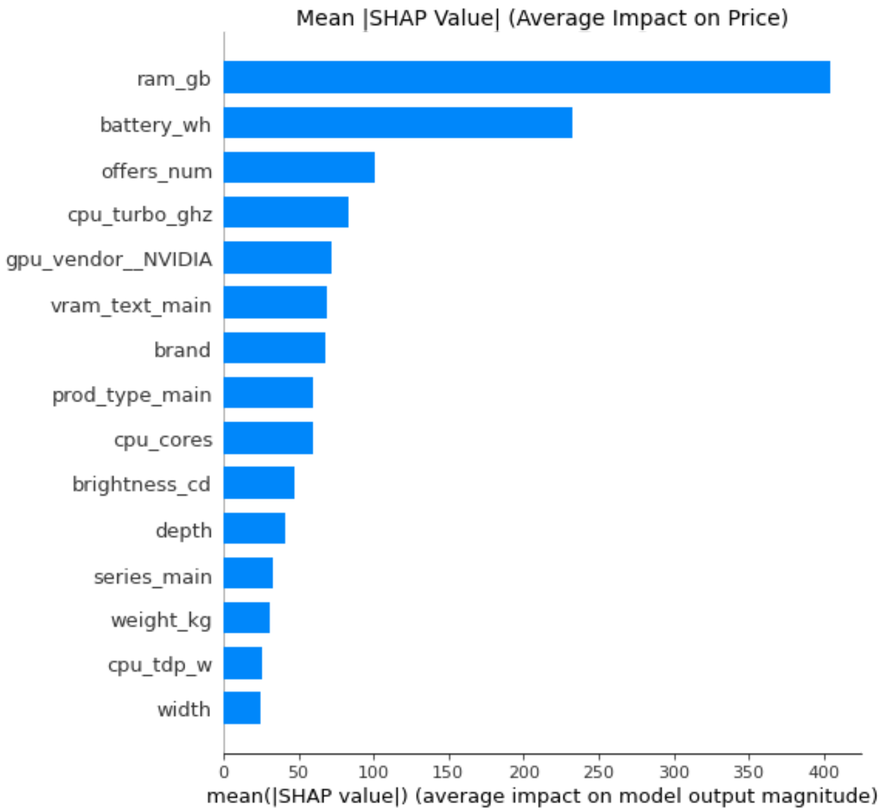
GridSearchCV with 5-Fold Cross-Validation

- 200 trees with depth 5 → captures complex patterns without overfitting
- Learning rate 0.15 → aggressive but balanced by more trees
- Low regularization → data is clean, model benefits from flexibility

Optimal:

- n_estimators=200
- max_depth=5
- learning_rate=0.15

What Drives Computer Prices? (SHAP Analysis)

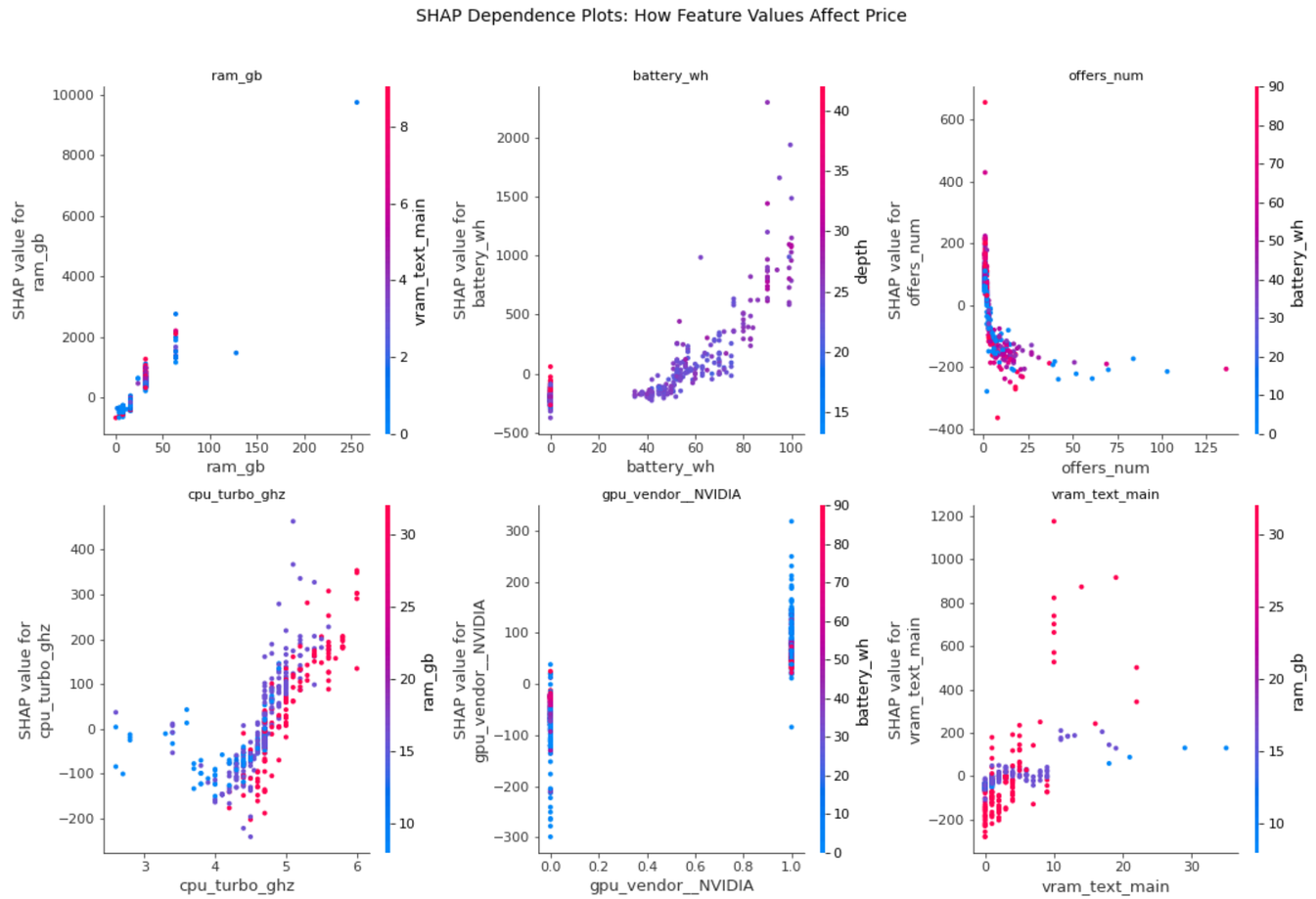


Key Insights

1. **RAM has the highest impact** (~€404 average) - the dominant price driver
2. **Battery capacity** (~€232) - proxy for premium/ultrabook builds
3. **Number of offers** - market competition affects pricing
4. **CPU turbo speed & NVIDIA GPU**: performance specs matter

A small set of features explains most of the price variation - RAM alone has 2x the impact of any other feature

How Features Affect Price (Non-Linear Relationships)



Key Insights

- RAM: Linear with premium jump at high configs
- Battery: Interacts with form factor (thin + powerful = premium)
- Market competition shows diminishing marginal effects

System Architecture & Inference Pipeline

Offline Training

- We select the winning algorithm (Gradient Boosting) which performed best in our Notebook analysis.

train_model.py

- Purpose: Orchestrates the entire ETL pipeline. It cleans data, imputes missing values, and trains the model.

Persistence Layer

We serialize the fitted model object (internal parameters) into a portable file

price_predictor.joblib

Purpose: A binary file created by the Joblib library. It does not save the price; it saves the Trained Model Object (the 100 decision trees and their weights).

feature_info.json

Purpose: The Schema Contract. Stores specific metadata like median_ram=16GB and Apple=4.5 needed for preprocessing.

UI

The Streamlit App loads this saved model to execute inference on demand.

app.py

Purpose: The Streamlit frontend. Captures raw user input (sliders/dropdowns).

app_utils.py

Purpose: The logic engine. Loads the artifacts and runs the calculation.

Live Market Analytics

Browse Computers

Smart Recommendations
K-Nearest Neighbors

- Inside app_utils.py, we run find_similar_products().
- Maps input to vector space, calculates Euclidean Distance.
- Returns 5 mathematical “twins” from 8,000+ database

- Why: This is purely mathematical search, requiring no trained model, just the raw data in memory.

Market Segmentation (Analyst View)

Market Segmentation
K-Means Clustering

- The Logic: We run this Live (perform_clustering) on the DataFrame.
- Automatically detects "Families" of computers (e.g., Budget, Gaming, Workstation) based on Specs & Price.
- Because it runs live, we can visualize the market structure dynamically
- Offers business insights beyond simple price prediction.

The Agentic Chatbot

The Brain: Gemini 2.5 Flash with Function Calling (Tools)

- We integrated Google's Gemini 2.5 Flash model to provide a conversational interface.
- But purely generative models hallucinate prices. To fix this, we gave it "Hands".

Tool 1: Price Prediction

get_price_prediction_for_agent:

LLM Action: "User asked for price." -> Tool Call: Runs our specific XGBoost .joblib model. -> Result: Returns precise € calculation.

Tool 2: Recommendations

recommend_laptops_for_agent:

LLM Action: "User wants recommendations." -> Tool Call: Runs our KNN function. -> Result: Returns actual inventory.

Tool 3: Data Query

get_brand_count:

LLM Action: "How many Dells?" -> Tool Call: Runs a Pandas count query. -> Result: 100% factual answer.

A Chatbot that calculates, not just chats

Advancements, Improvements & Future Work

Model

- Target R^2 above 0.80 with hyperparameter tuning
- Ensemble stacking: combine GradientBoosting + XGBoost + RandomForest

Data

- Expand dataset with more products and retailers
- Extract text features from descriptions using NLP

Predictions

- Validate input combinations (realistic specs only)
- Recommend similar computers based on prediction

UI

- Comparison mode: evaluate configurations side-by-side

Advancements, Improvements & Future Work

Uncertainty estimation & Prediction intervals

- Use Quantile Regression Forests or bootstrapped ensembles.
- • Improve user trust by presenting ranges instead of point estimates.

Device-Type Routing Models

- Separate models for desktops vs laptops vs ultra-portables.
- • Each has distinct price dynamics → boosts accuracy noticeably.

Hardware-Aware Neural Models

- Build a hybrid model combining engineered features with embeddings of hardware names (CPU/GPU text embeddings).

Dynamic Market Adaptation

- Integrate periodic retraining schedules.
- Track distribution drift → alert when model becomes outdated.

Similarity & Recommendation Engine

- kNN + cosine similarity for “best value for money”
- Bridge predictive → prescriptive functionality

Improved Benchmark Matching via NLP models

- Replace regex matching with transformer embeddings.
- • Increase CPU/GPU benchmark coverage significantly.

Dockerised Deployment + CI Pipeline

- Full reproducibility.
- Auto-builds, versioning, and model registry.

Limitations & Risks

- Dataset reflects only one marketplace → potential bias
- No temporal component → cannot capture price seasonality

Thanks for Listening!

Group 6