Predicting Apartment Prices in Morocco

An End-to-End Machine Learning Project
Inspired by *Hands-On Machine Learning* — Aurélien Géron

1,058	7	3	Mubawab
clean listings	Moroccan cities	models benchmarked	data source

Mustapha Aya
AI Engineer | AYAutomate | ayautomate.com
March 2026

1. Motivation

After reading Chapter 2 of Aurélien Géron's *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, I was struck by how clearly a full ML pipeline could be assembled — from raw data all the way to a tuned model — using nothing but Python and scikit-learn. Géron walks through every step on the California Housing dataset: data exploration, stratified splitting, feature engineering, pipeline construction, model comparison, and hyperparameter tuning.

My immediate reaction was a question: *what would this look like on Moroccan data?* Morocco's real estate market is active, geographically diverse, and largely untouched by public ML research. Apartment prices vary dramatically between a coastal city like Agadir and a financial hub like Casablanca, even for properties of comparable size. That variability felt like a natural test for a regression pipeline.

This project is the direct result of that curiosity. It is not a production system — it is a learning exercise, deliberately structured to mirror Géron's workflow step by step, applied to a real Moroccan dataset I scraped myself. The goal: see which parts transfer directly, where the Moroccan data introduces new challenges, and what the models can realistically learn from a few hundred listings.

2. Data Source & Collection

The data was sourced from **Mubawab.ma** — Morocco's leading real estate listings platform. To collect it, I built a custom **Apify actor**, a cloud-based web scraper that navigates Mubawab's listing pages, extracts structured property data, and stores results in an Apify Dataset called `mubawab-housing`. Each scrape run appends new listings to the same cloud dataset — by design, since the same pipeline will later support online/incremental learning as new listings arrive.

For this initial project, the scrape was run once in March 2026, producing **1,194 raw rows**. After deduplication, filtering to sale listings only, and removing rows with missing target values, the final working dataset contained **1,058 clean listings** across 7 cities.

Dataset Overview

Feature	Type	Missing	Notes
city	Categorical	0%	7 cities: Casablanca, Marrakech, Rabat, Tanger, Agadir, Meknès, Oujda
surface_m2	Numeric	<1%	Apartment area in square metres
num_rooms	Numeric	7%	Total number of rooms
num_bathrooms	Numeric	3%	Number of bathrooms
floor	Numeric	45%	Floor level — heavily missing in scraped listings
state	Categorical	1%	Construction state: new build, good condition, to renovate…
standing	Categorical	78%	High-end vs. mid-range — mostly missing
price_dh (target)	Numeric	10%	Listed price in Moroccan Dirham (MAD)

Table 1 — Feature summary. High missing rates in floor (45%) and standing (78%) are typical of scraped listing data where sellers frequently omit these fields.

3. Exploratory Data Analysis

Before building any model, I followed Géron's approach of visualising the data thoroughly — understanding its distribution, geographic spread, and correlations. Three patterns stood out: the dataset is dominated by a handful of cities; prices are heavily right-skewed with a long tail of luxury properties; and surface area is the strongest single predictor of price.

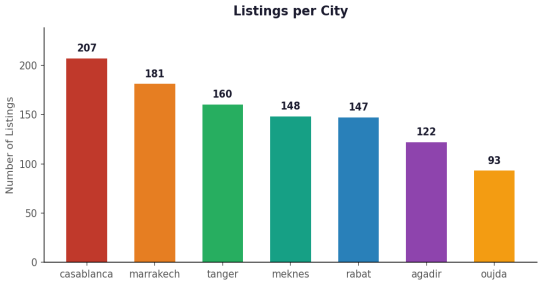

Figure 1 — Listing counts by city.

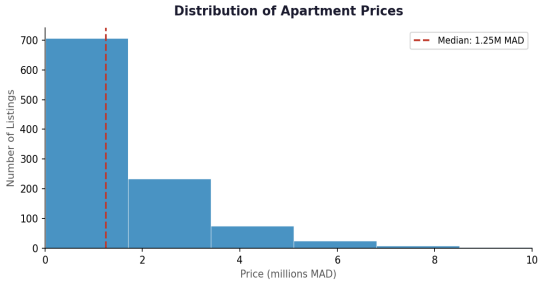

Figure 2 — Price distribution. Median: 1.25M MAD. Long right tail visible.

Casablanca and Tanger lead in listing volume. The price distribution confirms what practitioners expect from scraped listings: most properties cluster between 500K and 3M MAD, but a small number of luxury properties push the mean above the median. This right skew motivates a **log-transform of the target** in the online learning phase.

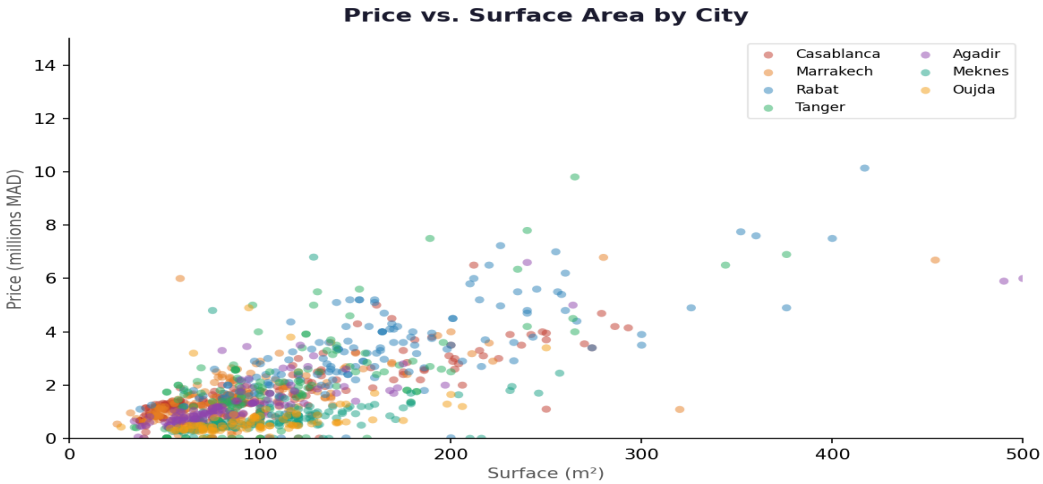

Figure 3 — Price vs. Surface by city. Casablanca (red) commands the highest prices per m². Oujda and Meknès cluster at lower price points for equivalent surfaces.

The scatter plot reveals a clear positive relationship between surface area and price, but with substantial variance. The city effect is pronounced: Casablanca listings are systematically more expensive per square metre than equivalent-sized listings in Meknès or Oujda — a dynamic the model must capture through categorical encoding of the city feature.

4. Pipeline Architecture

Following Géron's Chapter 2 methodology, I built a full scikit-learn pipeline that chains feature engineering, imputation, encoding, and scaling into a single reusable object. This ensures the exact same transformations are applied to both training and test data with zero data leakage.

Feature Engineering

Inspired by Géron's *CombinedAttributesAdder*, I created a custom sklearn transformer — `MoroccanFeatureAdder` — that derives three ratio features before any scaling or encoding takes place:

Feature	Formula	Intuition
`surface_per_room`	surface_m2 ÷ num_rooms	Space quality per room
`rooms_per_bathroom`	num_rooms ÷ num_bathrooms	Layout quality proxy
`surface_per_bathroom`	surface_m2 ÷ num_bathrooms	Combined space and plumbing signal

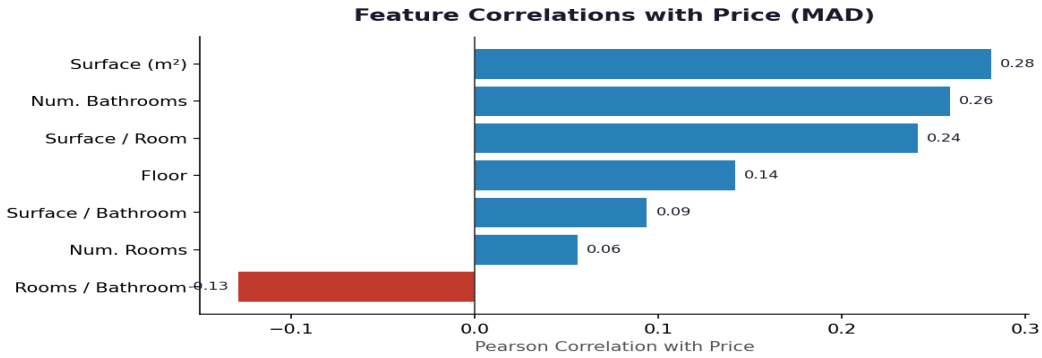

Figure 4 — Pearson correlations with price. Surface area and its ratios dominate. Floor level and room counts show weaker but positive signals.

Preprocessing

The full preprocessing uses scikit-learn's `ColumnTransformer` with two parallel sub-pipelines. Numeric features receive median imputation followed by standard scaling. Categorical features (`city`, `state`, `standing`) receive constant imputation followed by one-hot encoding. The `neighborhood` column — with 149 unique values — was dropped; target encoding is planned for a future iteration.

Train / Test Split

Stratified sampling on the `city` column preserves the geographic distribution in both train (80%) and test (20%) sets — exactly as Géron uses income categories for California. This is critical because city is the strongest categorical predictor.

5. Results

Three models were evaluated using 5-fold cross-validation on the training set, measuring Root Mean Squared Error (RMSE) in Moroccan Dirham. The Random Forest outperformed both the linear baseline and the decision tree.

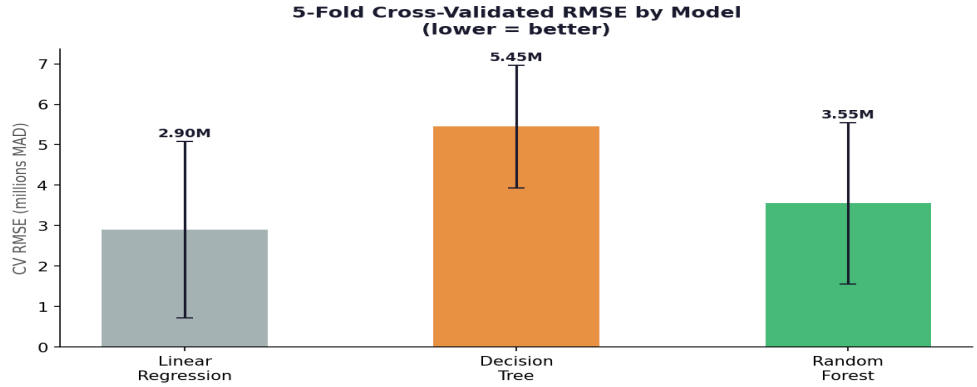

Figure 5 — 5-fold CV RMSE by model. Error bars show standard deviation across folds. Lower is better.

Model	CV RMSE (mean)	CV RMSE (std)	Notes
Linear Regression	2.90M MAD	±2.18M	Reasonable mean, high variance across folds
Decision Tree	5.45M MAD	±1.51M	Overfits — highest error overall
Random Forest ✓	3.55M MAD	±1.99M	Best mean RMSE, most stable

Table 2 — Cross-validation results. RMSE in millions of Moroccan Dirham. Median listing price: 1.25M MAD.

6. Discussion & Limitations

The Random Forest's CV RMSE of ~3.6M MAD may appear high relative to the 1.25M MAD median. Several factors explain this. The dataset contains extreme outliers — listings above 10M MAD that skew RMSE. The absence of latitude/longitude means the model cannot learn neighbourhood-level price gradients the way California's census-block data allows; city is a coarse signal. And the *standing* column — likely the strongest quality indicator — is missing for 78% of listings. Despite these constraints, the pipeline validates cleanly: the Random Forest correctly identifies Casablanca as a premium market and new-build listings as commanding a price premium over existing properties.

7. What Comes Next

This validation notebook is the first step in a larger project. The Apify scraper continues to run on demand, appending new listings to the cloud dataset. The next phase replaces the batch Random Forest with an **online learning pipeline** using `SGDRegressor.partial_fit()`: each time the scraper runs, the model updates incrementally on new listings without retraining from scratch. A built-in benchmark automatically compares the online model against a fresh Random Forest, tracking whether the online model is converging or drifting over time.

Neighbourhood target encoding: Replace OHE on the 149-value neighborhood column for a stronger geographic signal.

Log-transform the target: Apply log1p to price_dh to reduce the influence of luxury outliers on RMSE.

Latitude / longitude enrichment: Geocode each neighbourhood to enable spatial features — the equivalent of California's ocean proximity.

XGBoost / LightGBM: Benchmark gradient boosting, which typically outperforms Random Forests on tabular data.

All code is available on request. The Apify scraper, validation notebook, and online learning pipeline were built entirely in Python using scikit-learn, pandas, matplotlib, and ReportLab. If you are working on a similar project in MENA real estate, I would love to connect.

Mustapha Aya · AI Engineer · AYAutomate · ayautomate.com · linkedin.com/in/liaichi-mustapha