SMART PROPERTY PRICING FOR REAL ESTATE AGENCIES

By Pascal Opara

CONTENTS

PROJECT OBJECTIVE

The aim of this project is to develop a predictive pricing model for residential properties based on various housing features and location data. The goal is to support real estate agencies with intelligent pricing strategies that reflect property quality, size, location, and amenities

DATASET SUMMARY

- **Total Entries**: 4820
- **Columns**: 16 original columns including sold_price, sqft, longitude, latitude, year_built, garage, taxes, bathrooms, bedrooms, and more
- **Missing Values**: Only kitchen_features had missing values, which were filled with "Unknown
- **No zero values** were found in critical numeric columns like sold_price, sqft, longitude, latitude

FEATURE ENGINEERING

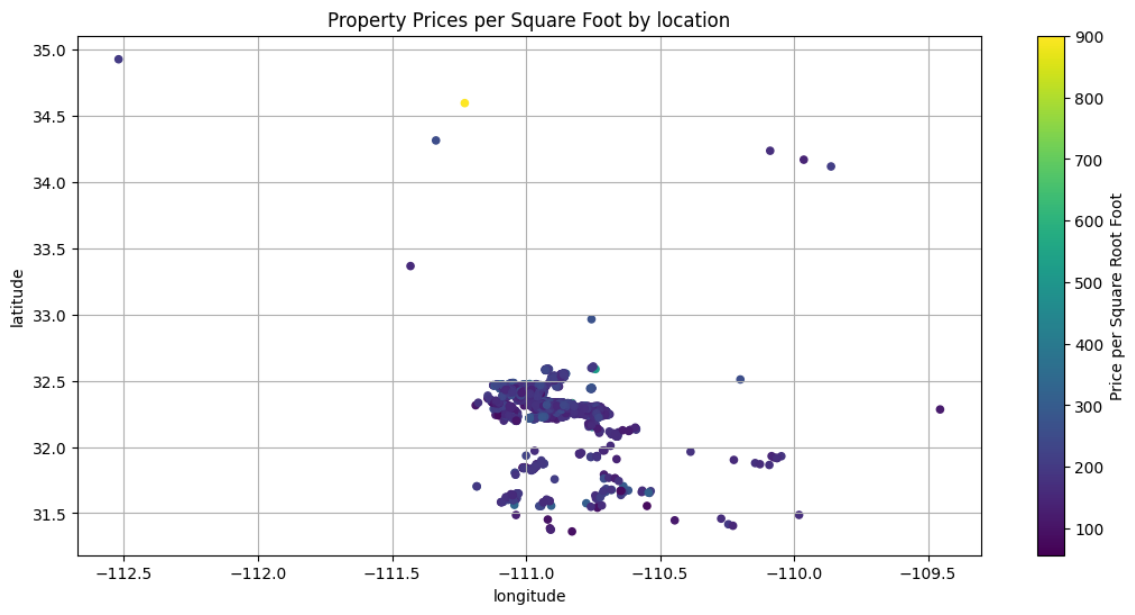
To enhance prediction, these columns were created:

- Price_per_sqft = sold_price~~/sqft~~ was created and visualized to reflect location-based property value
- House_age = 2025 – year_built captured property age.
- total_rooms = bedrooms + bathrooms combined interior capacity.
- Is_luxury = binary classification based on (garage > 2 and fireplaces > 2), indicating high-end homes.
- All of the new columns were later dropped from the final model to reduce multicollinearity and complexity.

EXPLORATORY DATA ANALYSIS

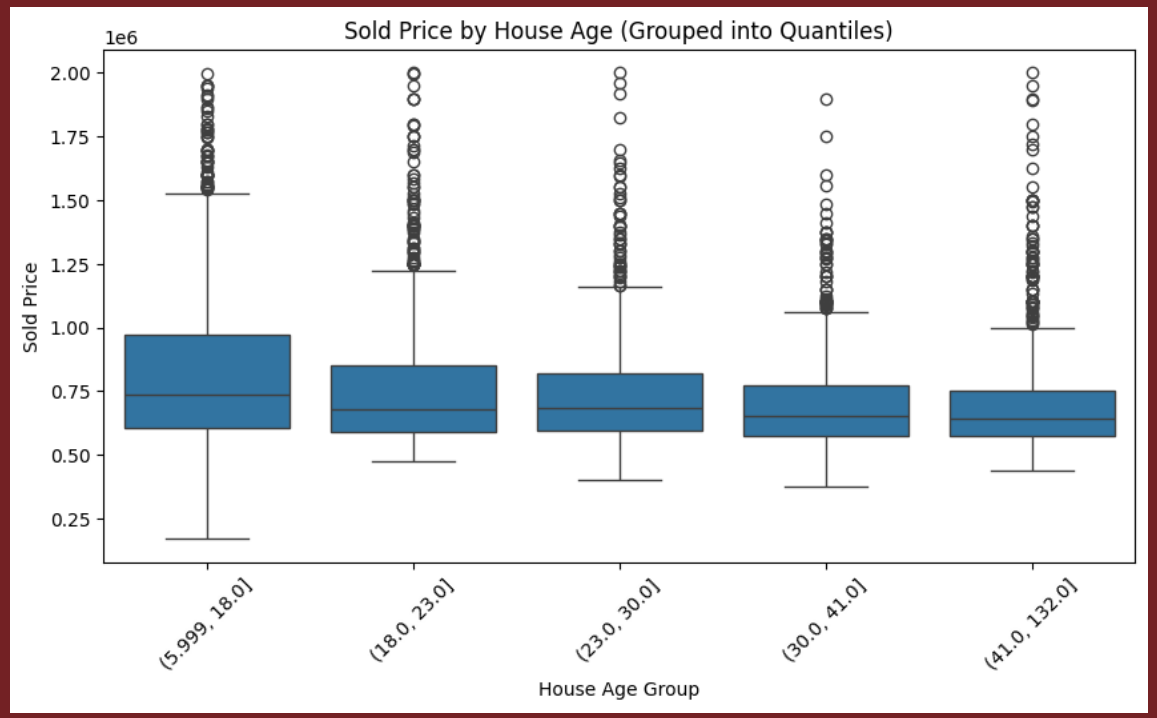
Insights:

- Lower-value properties are mostly concentrated in southern and central zones(lat. 31-32.5)
- High-value properties are visible in northern regions(lat. 34-35)



Insights:

- Prices vary widely with location.
- Younger houses tend to sell at higher prices
- Newer homes also show more outliers – indicating more luxury/high-value properties.



MODELLING APPROACH

K-Nearest Neighbors Regressor

- Custom KNNRegressor class with soft weighting using distance
- 70%/30% training-test data split
- Trained on scaled X values (StandardScaler) and non-scaled X values
- Y value index was initially reset to align with distance-based index referencing during prediction but was later reversed as I updated my .fit() method in my class and also changed .predict() method

Multivariate Linear Regressor using SGD(OLS)

- Custom MVLinearRegression using gradient descent and loss tracking.
- 70%/30% training-test data split
- X values were scaled, Y values were converted to NumPy for matrix operations (not reset)
- Learning rate (eta) and epochs were tuned:
- Eta = 0.01 and epochs = 3000 provided smooth early flattening training curve.

COMPARISION: SCALED VS UNSCALED

Observation:

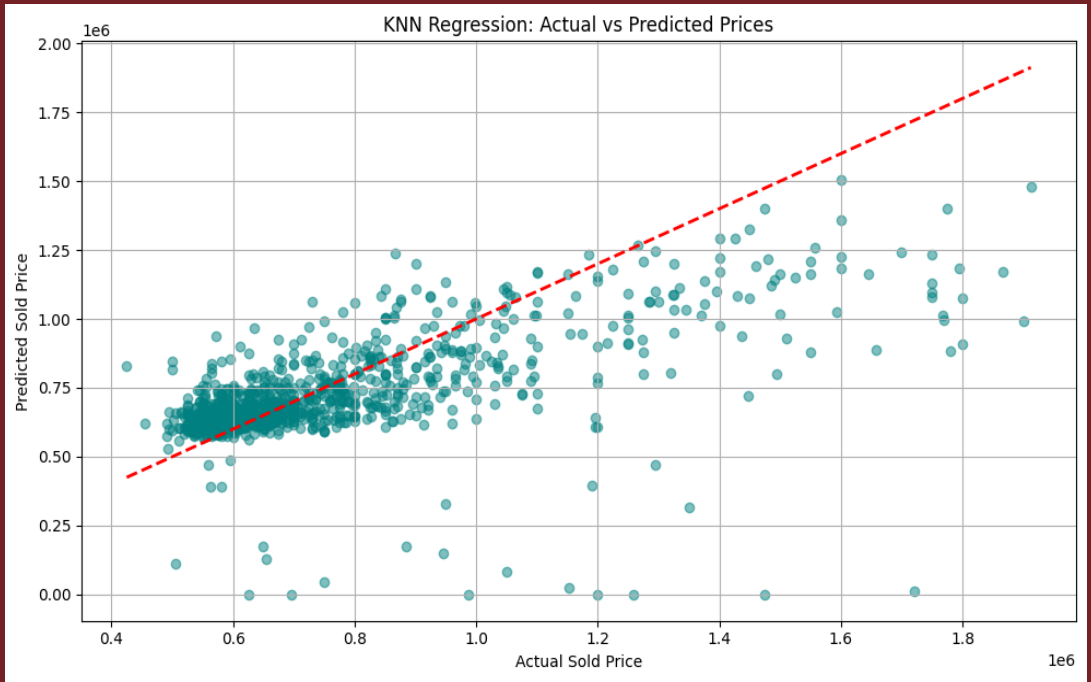
- OLS Regression failed because the model underfit the complex nonlinear relationships present in the housing data
- High residuals and cost despite good training convergence show OLS couldn't generalize
- Model overpriced small-range houses and underpriced luxury home due to location and a limited set of data used to train and KNN is distance based as it picks its closest neighbors.
- Non-scaled KNN Regression did not work because since KNN works based on distance, our dataset features are on different scales. Larger scale features dominated which led to bad predictions

Model	Scaled	MAPE
KNN	Yes	12.91%
KNN	No	98.65%
OLS	Yes	104.33%
OLS	No	Similar

KNN REGRESSION VISUALS

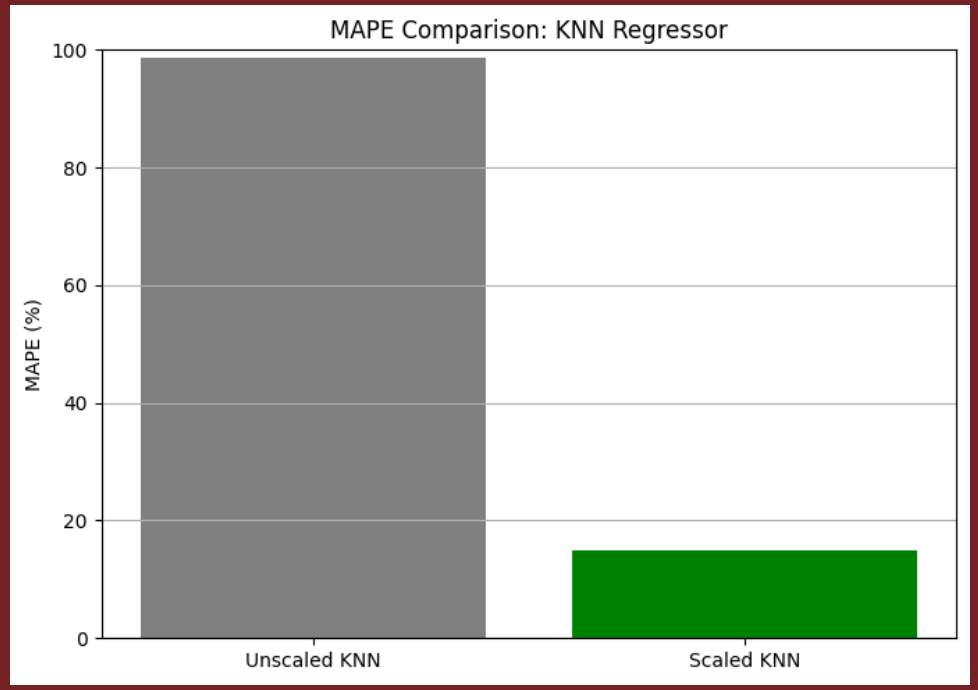
ACTUAL VS PREDICTED PRICES SCATTER PLOT

- Most predictions align well with actual sold prices, with tighter clustering near the line of perfect prediction, showing reliable performance on most data points.



MAPE COMPARISION BAR CHART

- Scaling improves KNN accuracy dramatically by normalizing feature influence. MAPE drops from nearly 100% to 12.91%, showing the importance of preprocessing in distance based models.



FINAL MODEL JUSTIFICATION AND LESSONS LEARNED

KNN Regressor on Scaled X(features) was selected as the final model because:

- Strong generalization (MAPE - 12.91%)
- Natural fit for non-linear relationships in house data
- Clear visual alignment in prediction vs. Actual distribution

The lessons learned were as thus:

- Scaling is critical for KNN, not optional
- I updated my .fit() method to convert y into a NumPy array and change the .predict method by modifying the gamma_k weight calculation as when I entered foreign values my model was overfitting.
- OLS may not capture non-linear patterns in real estate data, even after tuning
- 70/30 training-test split might suggest that test data contained outliers or price patterns not well represented in training data which led me to alter the random state and training-test split.
- Visualization early in the project helped guide model decisions.

RECOMMENDATION & CONCLUSION

Recommended Model: KNN Regressor (with Scaled Features)

Why KNN?

• It performs well on non-linear and location-sensitive data, which reflects real housing markets more accurately.

• It doesn't assume a fixed mathematical relationship — instead, it learns directly from similar past records (neighbors).

• With scaling applied, KNN becomes sensitive to all relevant features equally, improving prediction accuracy.

It needs a wide variety of data to function properly

Practical Implication for Real Estate Agencies:

KNN can power intelligent pricing tools that suggest accurate selling prices based on similar past sales. Agencies can use this for automated valuation, smart recommendations, and dynamic pricing strategies that consider size, location, amenities, and recent trends.

• The KNN regression model with feature scaling outperformed the OLS model and provided strong accuracy and generalization.

• OLS regression failed to adapt to the wide range of values and non-linear relationships in housing prices, even with tuning.

• The final KNN model captured geographic trends, property characteristics, and market behavior far better than linear regression could.

What this means:

For real estate agencies, this model can be integrated into digital platforms to automatically evaluate properties, price listings more competitively, and support smarter negotiations — all while staying grounded in real transaction data.

It fulfills the use case goal of making pricing smarter, faster, and backed by data.

Q & A