Predicting Neighborhood Affordability Tiers in Toronto Using Static Urban Features

Laura S. Gomez Villalba
Western University
lgomezvi@uwo.ca

Thomson Lam
Western University
zlam5@uwo.ca

Oreoluwa Ogundipe
Western University
oogundi2@uwo.ca

Kevin Liu
Western University
kliu469@uwo.ca

Besma Serrai
Western University
bserrai@uwo.ca

***Abstract*—We present a machine learning system for predicting which affordability tier (Budget, Moderate, Expensive, or Premium) a Toronto neighborhood will occupy two years ahead, using only non-rent urban characteristics. Our dataset spans 15 years (2010–2024) across 158 neighborhoods, integrating rental, crime, transit, and green space data. We conducted a randomized hyperparameter search across 60 configurations of three model families evaluated under both stratified and temporal cross-validation. The best model (Gradient Boosting) achieves a cross-validated macro F1 of 0.740 and a held-out test F1 of 0.604 (95% CI: 0.560–0.644), a 2.4× improvement over random baseline. The convergence of all three model families to similar performance provides evidence that the ceiling is data-driven. We complement the classifier with K-Means clustering and deploy the system as an interactive web application. Code: github.com/Western-Artificial-Intelligence/condo-cost-predictor.**

I. Introduction

A. Motivation

Toronto's rental market has experienced sustained price growth, with average one-bedroom rents increasing from approximately $1,100 in 2010 to over $2,200 in 2024. The Canada Mortgage and Housing Corporation reports that Toronto's rental vacancy rate fell below 1.5% in 2023 [1]. For students and young professionals, understanding which neighborhoods are likely to remain affordable is a practical concern with limited tooling support.

With the growing availability of municipal open data [2], there is an opportunity to combine longitudinal rental data with neighborhood characteristics—transit, safety, green space—to reason about future affordability. Prior work has focused on listing-level features [3], [4], leaving neighborhood-level forecasting underexplored. We aim to predict which affordability tier a neighborhood will occupy two years from now using only non-rent urban features.

Our contributions are: (1) a longitudinal dataset of 158 neighborhoods across 15 years from 4 municipal sources; (2) a systematic model selection pipeline with cross-validated search and bootstrapped confidence intervals; (3) an analysis of the predictive ceiling imposed by static features, framed as a feature limitation within tree-based models; and (4) an interactive web application for affordability exploration.

B. Related Works

Rental price prediction has been studied extensively at the listing level, where features like square footage, number of bedrooms, and building age directly describe the unit being priced. Pow et al. [3] applied gradient boosting and random forests to Montreal listings, achieving strong predictive accuracy. Ho et al. [4] compared machine learning methods for property price prediction in Hong Kong, finding that ensemble tree methods consistently outperform linear models when non-linear feature interactions are present.

At the neighborhood level, prediction is less studied. Chaphalkar and Sandbhor [5] used hedonic pricing models with neighborhood amenities but relied on cross-sectional data, limiting temporal generalization. Li et al. [6] applied gradient boosting to predict neighborhood-level rent changes in New York using time-varying features (construction permits, subway ridership changes), achieving higher accuracy than static-feature approaches.

Our work differs in two ways: (1) we predict ordinal affordability tiers rather than continuous dollar amounts, which is more robust to the autocorrelation problem in rent time series, and (2) we explicitly characterize the performance ceiling imposed by static features rather than treating it as a limitation to be minimized.

C. Problem Definition

Given $\mathbf{x}_i \in \mathbb{R}^{21}$ describing neighborhood i at year t, we predict $y_i \in \{1, 2, 3, 4\}$ (the rent quartile at $t+2$). We maximize macro F1:

$$\text{Macro-F1} = \frac{1}{4}\sum_{c=1}^{4}\frac{2P_cR_c}{P_c + R_c}, \tag{1}$$

which weights all tiers equally. Critically, $\mathbf{x}_i$ excludes current rent (`avg_rent_1br`), as its inclusion causes the model to learn the trivial mapping "current tier $\approx$ future tier." By design, the model must predict affordability from *structural* neighborhood characteristics—density, transit access, green space, safety—rather than from rent itself. This means the model learns which physical attributes are associated with each tier, not how rent changes over time within a neighborhood.

II. Methodology

A. Data and Features

Our dataset integrates four municipal data sources across 158 Toronto neighborhoods:

TABLE I
DATA SOURCES AND TEMPORAL COVERAGE.

Source	Data	Years
TREB	Avg. 1BR rent	2010–2024
Toronto Open Data	Crime rates (4 types)	2014–2024
TTC GTFS	Transit stops, routes	2024
Toronto Open Data	Park boundaries	2024
Statistics Canada	Population	2021

Raw data was collected via per-year ETL notebooks using DuckDB with spatial extensions for point-in-polygon joins (assigning transit stops and parks to neighborhoods). The 15 per-year CSVs were assembled into a master dataset (2,528 rows $\times$ 95 columns), then cleaned to 2,054 rows $\times$ 21 columns by deduplicating erroneous rows, imputing missing crime rates (2010–2013) using training set medians, and dropping rent-derived features to prevent target leakage.

The target (`TARGET_TIER_2YR`) assigns each neighborhood-year to the rent quartile it will occupy two years later:

TABLE II
TIER DEFINITIONS BASED ON WITHIN-YEAR RENT QUARTILES.

Tier	Label	Definition
1	Budget	Bottom 25% of rents
2	Moderate	25th–50th percentile
3	Expensive	50th–75th percentile
4	Premium	Top 25% of rents

Quartile-based tiers handle inflation naturally: a $1,200 rent was "Expensive" in 2010 but "Budget" in 2024.

We use 14 base numeric features, 6 engineered ratio features, and 1 categorical feature (21 total).

TABLE III
BASE NUMERIC FEATURES BY CATEGORY (14 TOTAL).

Category	Features
Geography	area_sq_meters, perimeter_meters
Crime	4 rates: assault, auto theft, robbery, theft over
Transit	stop count, frequency, line length, density, route count (6)
Green space	park_count
Demographics	POPULATION

The engineered features normalize raw counts by area or population:

TABLE IV
ENGINEERED FEATURES AND CORRELATION WITH TARGET.

Feature	Formula	Corr.
`park_density`	parks / area	0.47
`pop_density`	pop. / area	0.38
`transit_per_cap.`	stops / pop.	0.29
`total_crime_rate`	$\sum$ 4 rates	0.07
`compactness`	perim.^{2} / area	varies
`routes_per_stop`	routes / stops	varies

For comparison, raw `park_count` correlates at only 0.01 with the target; `park_density` achieves 0.47—a 47$\times$ improvement from normalizing by area. The single categorical feature, `CLASSIFICATION_CODE` (3 values), is label-encoded.

Critically, `avg_rent_1br` (current rent) is *excluded* from the feature set. Current rent almost perfectly predicts future rent tier due to autocorrelation; including it causes the model to learn the trivial mapping "current tier $\approx$ future tier" and ignore all neighborhood characteristics.

B. Train/Test Split

We use a time-based split: train on 2010–2019 (1,580 rows), test on 2020–2022 (474 rows). No random shuffling—the model should never see future data during training.

TABLE V
TIME-BASED TRAIN/TEST SPLIT.

Split	Years	Rows	Purpose
Train	2010–2019	1,580	CV and model fitting
Test	2020–2022	474	Final evaluation

The training set is approximately balanced: Budget (418), Moderate (403), Expensive (387), Premium (372).

C. Model Families

We evaluated three tree-based ensemble methods:

Random Forest (RF) [7] trains independent decision trees on bootstrap samples of the data, then aggregates predictions by majority vote. It is naturally resistant to overfitting through bagging and feature subsampling.

XGBoost [8] builds trees sequentially, where each tree corrects the residual errors of the ensemble so far. It includes L1 (`reg_alpha`) and L2 (`reg_lambda`) regularization on leaf weights, plus row and column subsampling.

Gradient Boosting (GBM) [9] uses a similar sequential boosting approach but with a different implementation (scikit-learn [10]). It employs stochastic gradient boosting (`subsample` < 1.0) for regularization.

D. Hyperparameter Search

Using `RandomizedSearchCV` from scikit-learn [10], we sampled 10 hyperparameter configurations per model family from predefined distributions and evaluated each under two cross-validation (CV) strategies:

- **StratifiedKFold** (5 splits, shuffled): ensures proportional tier representation per fold. Appropriate because non-rent features are static across years, minimizing temporal leakage risk.
- **TimeSeriesSplit** (5 splits): respects temporal ordering. More conservative but produces unbalanced fold sizes.

Total: 3 models $\times$ 2 CV strategies $\times$ 10 iterations = 60 configurations, 300 model fits. We note that all three families are tree-based; the convergence result (Section III) is therefore suggestive of a feature limitation within this model class, not a universal claim.

E. Model Selection Criterion

The configuration with the highest mean cross-validated macro F1 across all 60 entries was selected. Macro F1 (Equation (1)) was chosen over accuracy because it weights all four tiers equally, preventing the model from ignoring underrepresented classes.

F. Confidence Intervals

After model selection, we estimated 95% confidence intervals (CIs) on the held-out test set using the bootstrap percentile method: resample the 474-row test set with replacement 2,000 times, compute metrics on each resample, and report the 2.5th and 97.5th percentiles as CI bounds.

G. Neighborhood Clustering

Separately, we clustered the 158 neighborhoods in the 2024 snapshot using K-Means on standardized (z-scored) versions of the 20 numeric features. We evaluated $k = 3$ through $k = 12$ using both inertia (elbow method) and silhouette score. The best silhouette ($s = 0.212$) occurs at $k = 3$, but this produces one dominant 96-neighborhood cluster with limited practical utility. We selected $k = 7$ ($s = 0.135$) as a pragmatic trade-off: it yields 7 distinct profiles with no single catch-all bucket, and the smallest cluster (2 neighborhoods) corresponds to a genuinely distinctive transit hub archetype. The clustering serves as an *exploration tool* for the application's "find similar neighborhoods" feature rather than a validated taxonomic finding; the moderate silhouette reflects the high dimensionality (20 features) and the continuous, overlapping nature of urban neighborhood characteristics.

III. Results

A. Cross-Validation Results

TABLE VI
BEST CV MACRO F1 PER MODEL FAMILY AND CV STRATEGY.

Model	CV Strategy	F1	Std
GBM	**StratifiedKFold**	**0.740**	**0.033**
RF	StratifiedKFold	0.734	0.032
XGBoost	StratifiedKFold	0.732	0.025
GBM	TimeSeriesSplit	0.715	—
XGBoost	TimeSeriesSplit	0.712	—
RF	TimeSeriesSplit	0.705	—

All three families converge to a narrow F1 band (0.73–0.74) under StratifiedKFold (Fig. 1). This convergence across bagging (RF), sequential boosting (GBM), and regularized boosting (XGBoost) is suggestive that, within tree-based models, the bottleneck is the feature set rather than model capacity. StratifiedKFold outperforms TimeSeriesSplit by 0.02–0.03 points, as expected given the static features.

B. Held-Out Test Performance

TABLE VII
TEST PERFORMANCE WITH 95% BOOTSTRAPPED CIS.

Tier	P	R	F1	95% CI	n
Budget	.64	.60	.62	[.55,.69]	124
Moderate	.52	.53	.53	[.45,.60]	119
Expensive	.55	.53	.54	[.46,.61]	125
Premium	.70	.76	.73	[.66,.79]	106
Macro	.60	.61	**.60**	**[.56,.64]**	474

Overall accuracy is 60.1% (95% CI: 55.7%–64.3%), a 2.4× improvement over the 25% random baseline. Premium neighborhoods are easiest to predict (F1 = 0.73) due to distinctive structural characteristics (high density, high transit access). Middle tiers are hardest because neighborhoods near the 50th percentile boundary are genuinely ambiguous—a neighborhood at the 48th percentile of rent could plausibly be classified as either Moderate or Expensive. The bootstrap distribution of macro F1 (Fig. 2) confirms the CI is tight relative to the effect size.

C. Confusion Matrix and Error Analysis

The off-diagonal mass concentrates on adjacent tiers: Budget is misclassified as Moderate (29 cases) far more often than as Premium (5). This is consistent with the ordinal nature of the target—neighborhoods near a quartile boundary are genuinely ambiguous.

D. The CV–Test Gap

The CV F1 (0.740) exceeds test F1 (0.604) by 0.136 points. Because the model learns *structural* neighborhood characteristics (density, transit, green space) rather than temporal dynamics, it captures which physical attributes are associated with each tier within the 2010–2019 training distribution. When the 2020–2022 test period introduces pandemic-era disruption (suburban demand increase, downtown vacancy spike), these static structural signals cannot explain the shift. The gap therefore quantifies the cost of relying on time-invariant features in a temporally shifting market. Notably, the 2020–2022 test window likely represents a *lower bound* on performance: test accuracy on a non-pandemic period would plausibly be higher, as the training and test distributions would be more aligned.

E. Ablation: CLASSIFICATION_CODE

SHAP analysis (Fig. 4) ranks `CLASSIFICATION_CODE`—a City of Toronto socioeconomic designation (NIA/EN/none)—as the most-used feature. To test whether the model depends on this government label, we re-ran the full pipeline without it:

TABLE VIII
ABLATION: REMOVING `CLASSIFICATION_CODE`.

Metric	With	Without	Δ
CV F1	0.740	0.739	−0.001
Test F1	0.604	0.591	−0.013
Test F1 CI	[.56,.64]	[.55,.63]	overlap

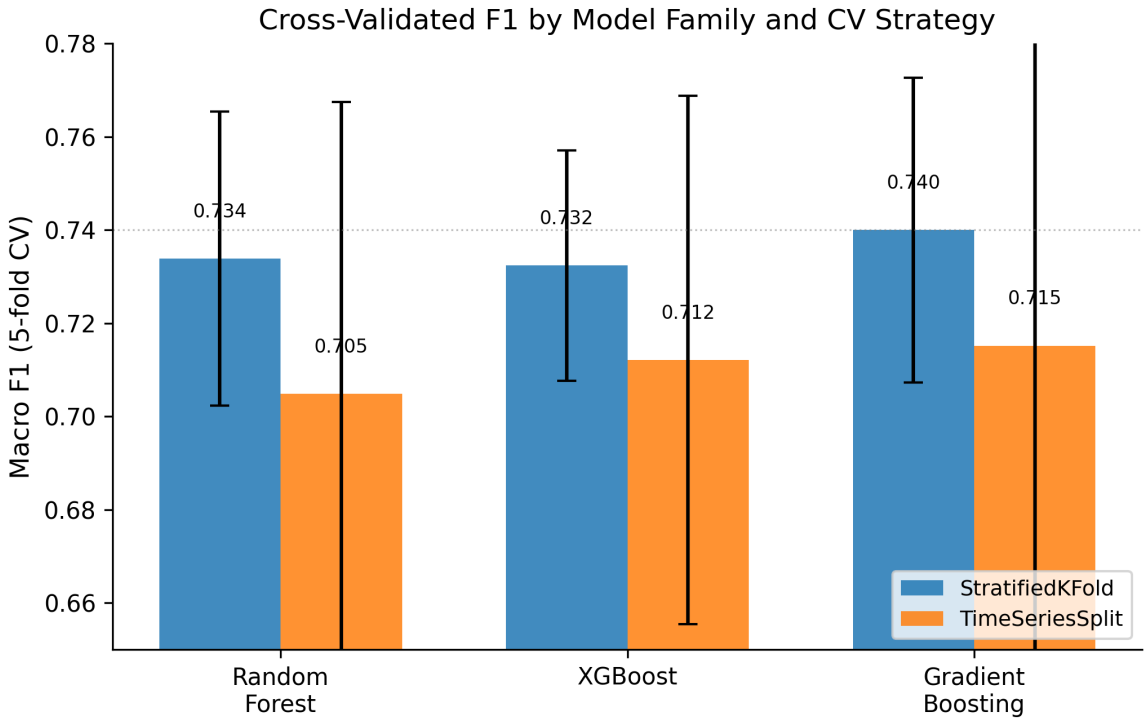

Fig. 1. Cross-validated macro F1 across three tree-based model families and two CV strategies. All StratifiedKFold results converge to 0.73–0.74 with overlapping error bars, suggesting the performance ceiling is feature-limited within this model class.

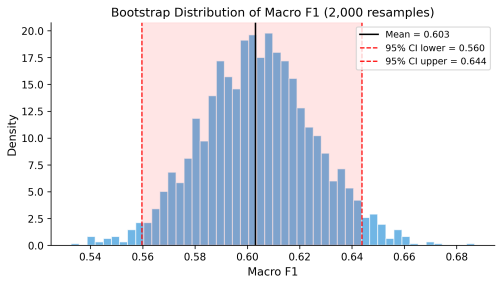

Fig. 2. Bootstrap distribution of macro F1 (2,000 resamples). The 95% CI (shaded) spans 0.560–0.644.

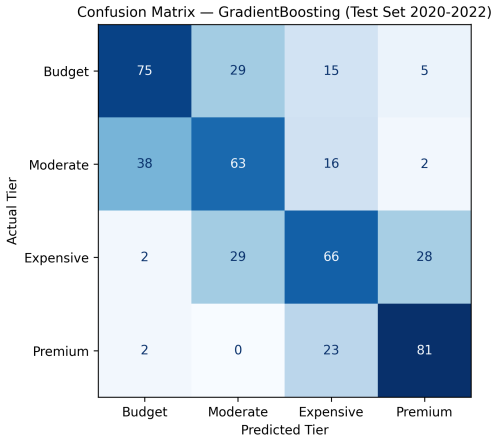

Fig. 3. Confusion matrix (test set). Errors concentrate on adjacent tiers, consistent with the ordinal target structure.

TABLE IX
PER-TIER F1 WITH AND WITHOUT CLASSIFICATION_CODE.

Tier	With	Without	Δ
Budget	0.622	0.602	−0.021
Moderate	0.525	0.521	−0.004
Expensive	0.539	0.522	−0.017
Premium	0.730	0.719	−0.011

Removing the feature costs only 0.013 F1—well within the 95% CI—and essentially zero on cross-validation (−0.001). The same model family (GBM) and CV strategy (StratifiedK-Fold) win in both cases. The model's predictive signal comes from genuine structural characteristics (density, transit, green space), not the government designation.

F. Clustering Results

TABLE X
NEIGHBORHOOD CLUSTERS ($k = 7$).

Cluster	n	Character
Frequent-Service Corridor	22	High-freq. transit
Connected Family	45	Parks + transit
Quiet Low-Density	56	Suburban
Downtown & Entertain.	5	High crime, parks
Major Transit Hub	2	Extreme transit
Transit-Rich Suburban	18	Large + transit
High-Density Urban Core	10	Very high density

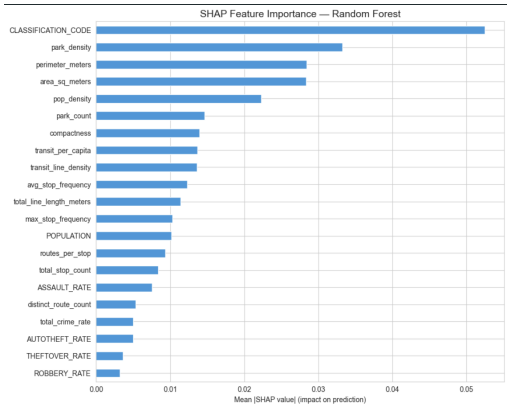

Fig. 4. SHAP feature importance. `CLASSIFICATION_CODE` ranks highest but is redundant (Table V); the true signal comes from density and transit features.

Cluster labels were assigned by inspecting standardized centroid values. The clusters serve the application's "find similar neighborhoods" feature: a user priced out of a High-Density Urban Core neighborhood can discover Transit-Rich Suburban alternatives with similar transit access at lower rent.

IV. Discussion

The model learns a mapping from *structural* neighborhood characteristics to affordability tier. SHAP analysis (Fig. 4) shows `park_density`, `pop_density`, and transit features drive predictions after accounting for the redundant `CLASSIFICATION_CODE` (Table V). This framing is important: the model captures which physical attributes *characterize* each tier, not how neighborhoods *transition* between tiers over time. The CV–test gap (Section III-D) is therefore best understood as a feature limitation—the static features describe neighborhood structure well but cannot capture temporal dynamics like gentrification or new transit openings.

Despite 60% test accuracy, the system has practical value: it is 2.4× better than random on a 4-class problem, and errors concentrate on adjacent tiers (low-cost mistakes). Predicting "Moderate" when the truth is "Budget" is a much smaller error than predicting "Premium" when the truth is "Budget," and the confusion matrix (Fig. 3) confirms the model rarely makes such extreme mistakes. The clustering component works independently of prediction accuracy—"find similar neighborhoods" relies on the same static features that describe neighborhood character well, even when they are limited for temporal prediction.

Limitations. (1) All evaluated models are tree-based; the convergence finding is suggestive within this class but not universal. Neural networks or SVMs may extract different signal. (2) Static features are the primary constraint; time-varying features would likely improve generalization. (3) Census population (2021) is proxied to all years. (4) Crime rates for 2010–2013 are imputed. (5) The 2020–2022 test period includes pandemic disruption; reported test performance is likely a lower bound on what the model would achieve in a normal market.

V. Conclusion

We built a neighborhood-level affordability tier classifier for Toronto achieving 60.1% accuracy (95% CI: 55.7%–64.3%) using only static urban features, a 2.4× improvement over random baseline. A hyperparameter search across 60 configurations of three tree-based families showed convergence to similar CV F1 (0.73–0.74), suggesting the ceiling is feature-limited within this model class. The 0.14-point CV–test gap quantifies the cost of static features in a shifting market, with the pandemic-era test window likely representing a lower bound. An ablation study confirmed the model relies on genuine structural characteristics rather than proxy labels. We deploy the classifier alongside 7-cluster neighborhood groupings in an interactive web application for Toronto rental affordability exploration.

Future work. Incorporating time-varying features (building permits, transit openings), evaluating non-tree architectures, exploring ordinal losses, and extending the prediction horizon analysis would address the primary limitations identified here.

VI. Acknowledgements

We thank the City of Toronto Open Data Portal, the Toronto Regional Real Estate Board, and the Toronto Transit Commission for making the underlying datasets publicly available. This project was developed as part of Western AI.

References

[1] Canada Mortgage and Housing Corporation, "Rental market report: Greater toronto area," CMHC, Tech. Rep., 2023. [Online]. Available: https://www.cmhc-schl.gc.ca/professionals/housing-markets-data-and-research/market-reports/rental-market-reports-major-centres

[2] City of Toronto, "Toronto open data portal," 2024. [Online]. Available: https://open.toronto.ca/

[3] N. Pow, E. Janulewicz, and L. Liu, "Applied machine learning project 4: Prediction of real estate property prices in montreal," in *Course Report, McGill University*, 2017.

[4] W. K. Ho, B.-S. Tang, and S. W. Wong, "Predicting property prices with machine learning algorithms," *Journal of Property Research*, vol. 38, no. 1, pp. 48–70, 2021.

[5] N. Chaphalkar and S. Sandbhor, "Use of hedonic pricing model for assessment of residential property values," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 2277–2282, 2019.

[6] Y. Li *et al.*, "Neighborhood-level rent prediction using gradient boosting with urban dynamics features," in *Proceedings of the ACM SIGKDD Workshop on Urban Computing*, 2022.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[9] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[10] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.