Housing Market Analysis

Authors: Christian Ahamba, Solange Ruiz, Damian Trent Wilson
Department of Information Systems, California State University Los Angeles
CIS 5560 Introduction to Big Data Science
cahamba@calstatela.edu sruiz85@calstatela.edu dwilso33@calstatela.edu

Abstract: The U.S. housing market significantly influences national economic activity and individual financial decisions. This project aims to analyze and predict housing market trends using an 8.5GB dataset from Redfin, enriched with real-time and historical data spanning over a decade. Leveraging Apache Spark ML and scalable machine learning pipelines, we forecast the national average median sale price using regression models and classify buyer behavior patterns. Unique aspects of this project include the integration of public and proprietary datasets, a dual focus on price forecasting and consumer behavior modeling, and the combination of long-term historical data with near real-time housing metrics. The insights generated are designed to support informed decision-making for homebuyers, investors, and policymakers.

1. Introduction

The U.S. housing market plays a vital role in economic stability and individual financial decisions. This project analyzes over 8.5GB of Redfin data, combining 10+ years of historical trends with near real-time housing information to forecast the national average median sale price. Using Apache Spark and Spark ML, we build scalable regression and classification models to predict pricing trends and classify buyer behavior. By integrating Redfin with public datasets, we aim to uncover patterns in price volatility and consumer dynamics. The insights generated are intended to support data-driven decisions for homebuyers, investors, and policymakers.

2. Related Work

Study 1

In the study *"Predicting Housing Prices and Analyzing Real Estate Market in the Chicago Suburbs Using Machine Learning"* by Kevin Xu and Hieu Nguyen (2023), the authors explore the use of various machine learning algorithms—namely linear regression, support vector regression, decision tree regression, random forest regression, and XGBoost regression—to predict housing prices in the Chicago suburbs. Utilizing Redfin sales data from 2018 through the summer of 2022, the research analyzes both model accuracy and market trends during a period marked by post-pandemic volatility. The results highlight XGBoost as the most effective model for price prediction. Furthermore, the study applies SHAP (Shapley Additive Explanations) values to interpret feature importance, offering insights into the key factors influencing housing prices in the region.

Study 2

Based on the study *"An Optimal House Price Prediction Algorithm: XGBoost"* by Hemlata Sharma, Hitesh Harsora, and Bayode Ogunleye (2024), this research approaches house price prediction as a regression task, evaluating the performance of several machine learning models including Support Vector Regressor, Random Forest Regressor, XGBoost, Multilayer Perceptron, and Multiple Linear Regression. Utilizing the Ames City housing dataset from Iowa, the study identifies significant features impacting housing prices. The findings indicate that XGBoost outperforms the other models, demonstrating superior accuracy and its ability to effectively model complex patterns in the housing data.

Study 3

In the 2020 study *Housing Market Prediction Problem Using Different Machine Learning Algorithms: A Case Study*, Shashi Bhushan Jha and colleagues investigate the effectiveness of several machine learning models—namely XGBoost, CatBoost, Random Forest, Lasso, and Voting Regressor—in predicting housing prices. Using a comprehensive dataset of 62,723 housing records from Volusia County, Florida, collected between January 2015 and November 2019, the researchers evaluate each model's performance based on metrics such as R^2, mean squared error (MSE), mean absolute error (MAE), and computational time. Their analysis reveals that the XGBoost model delivers the most accurate predictions, outperforming the other algorithms in both accuracy and efficiency.

Comparison with our Apache Spark-Based Approach

Our housing market analysis project presented differs notably from the other studies, through its integration of big data technologies and cloud computing. While the referenced studies primarily focus on evaluating the performance of machine learning algorithms such as linear regression, random forest, and XGBoost on relatively small, structured datasets using traditional computing environments, this project utilizes a significantly larger dataset—over 8.5GB of Redfin housing market data spanning more than a decade. The analysis is conducted using Apache Spark and Hive within a cloud-based environment (Databricks), enabling distributed data processing and scalable machine learning. Unlike the referenced studies, which do not leverage cloud

infrastructure, this project addresses real-world challenges related to data volume, system limitations, and model scalability. Additionally, the scope extends beyond price prediction to include consumer behavior modeling, offering more comprehensive insights for stakeholders such as homebuyers, investors, and policymakers. The use of advanced techniques like log transformations, TrainValidationSplit, Cross Validation, and performance benchmarking reflects a practical and infrastructure-aware approach to housing market prediction that sets this project apart from more traditional, algorithm-focused studies.

3. Methodology

3.1 Cluster Set Up

Table I

Components	Configuration
Total Nodes	5
Master Nodes	2 (High Availability enabled)
Worker Nodes	3
Memory/Node	32 GB RAM
CPU/Node	~2.3 GHz
Network Access	SSH via 144.24.13.0 subnet

3.2 Dataset Overview

Source: Kaggle (Redfin Housing Market Data)
Format: TSV
Size: ~8.5GB
Time Span: 10 years

Table II

Raw Data (Uncleaned)	
File Name	**Size**
county_market_tracker.csv	363 MB
Neighborhood_market_tracker.csv	4.8 GB
State_market_tracker.csv	20 MB
Us_national_market_tracker.csv	983KB
Zip_code_market_tracker.csv	2.8GB

3.3 Data Cleaning

Due to file size constraints in Databricks Community Edition, we cleaned and sampled the two largest Redfin datasets **(zip_code_market_tracker and neighborhood_market_tracker)** using Kaggle. Cleaning steps included duplicate removal, median/mode imputation for missing values, and dropping non-essential columns. We then sampled the cleaned data (approx. 15%) and exported it as CSV. These samples were uploaded to HDFS and used for model training and evaluation in Spark. Final model results were produced using spark-submit for full reproducibility and performance tracking.

Table III

Cleaned Data	
File Name	**Size**
county_market_tracker.csv	371 MB
Neighborhood_market_tracker.csv	5.0 GB
State_market_tracker.csv	20 MB
Us_national_market_tracker.csv	983KB
Zip_code_market_tracker.csv	3.0GB

3.4 Workflow Diagram

The explanation for the **Data Processing Workflow** section is now expanded with clear descriptions for each step in the pipeline.

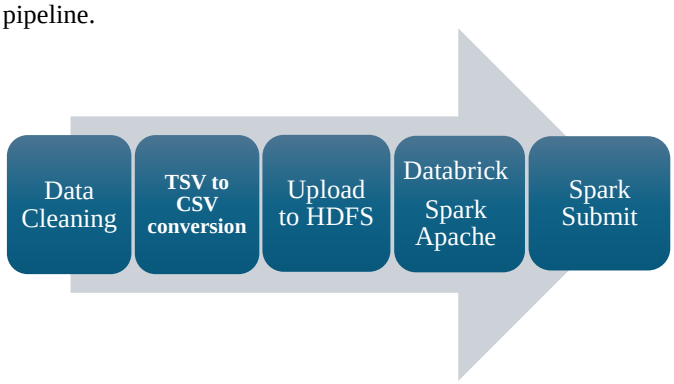

Fig. 1. Data processing workflow for housing market analysis.

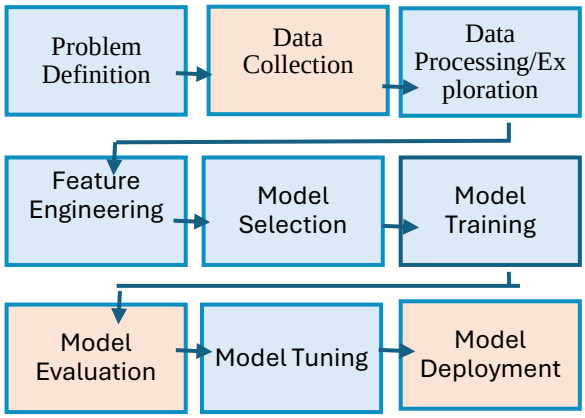

Fig. 2. End-to-end architecture for model training and evaluation.

4. Modeling and Results

4.0 Strategy Overview

We apply three strategies Baseline, TrainValidationSplit (TVS), and CrossValidation (CV) to each dataset/model combination. Dataset-model pairings were selected based on file size, feature richness, and spatial resolution.

4.1 Random Forest Regression
County Market Tracker (2.8MB)

To improve **median list price** prediction, we trained a Random Forest Regressor on county level housing data using a wide range of market, price, and supply features. Give the skewed prices distribution, we applied a log transformation to the target **(median_list_price)** to reduce the impact of outliers and stabilize variance, a common in housing models. Training was run via spark-submit, completing in **~1 hour 32 minutes** across all strategy and target combinations (raw/log x Baseline, TSV, CV). As expected, Cross Validation was the most computed intensive, training 18 models total. Top Features (Figure) included **median_list_ppsf**, **avg_sale_to_list**, and **median_ppsf**. In the **log-transformed model**, **median_list_ppsf** became even more dominant, suggesting better alignment with the transformed target and improved model focus.

Table IV

County Market Tracker - Random Forest			
Top 5 Feature Importance			
Raw Data		**Log Data**	
median_list_ppsf	0.2624	median_list_ppsf	0.5749
avg_sale_to_list	0.2107	median_ppsf	0.2858
median_ppsf	0.2070	avg_sale_to_list	0.0661
median_dom	0.1018	price_drops	0.0317
inventory	0.0688	off_market_in_two_weeks	0.0119

Table V

Performance Metrics					
Target Type	**Strategy**	**RMSE**	**R^2**	**Train Time (min)**	**Compute Time**
RAW	Baseline	680.70	0.304	1.19	92 mins
	TSV	707.47	0.248	11.99	
	CV	717.02	0.228	29.66	
LOG	Baseline	0.3057	0.468	1.13	
	TSV	0.2841	0.541	14.39	
	CV	0.2838	0.542	32.20	

4.2 Random Forest Regression
Zip Code Market Tracker (3.0GB)

This model uses Random Forest Regression to predict **median_sale_price** of homes using the **zip_code_market_tracker.csv** dataset. Data preparation involves selecting 13 numerical housing-related features, handling missing values using the Kaggle Notebook UI, while using **VectorAssembler** and **MinMaxScaler** to prepare features for modeling. We trained a baseline **RandomForestRegressor** on the training data and evaluated on the test set using **RMSE** and R^2. **TrainValidationSplit** and **CrossValidator** were used to optimize hyperparameters such as **maxDepth** and **maxBins,** while also evaluating each tuned model on the test set. We compared baseline, TrainValidationSplit and CrossValidator models using RMSE, R^2, and training time, which is shown below. This setup provides both a strong baseline and optimized models, making it a reliable framework for evaluating housing price trends.

Table VI

Zip Code Market Tracker - Random Forest	
Top 5 Feature Importance	
median_list_price	0.1024
median_sale_price_yoy	0.0948
homes_sold	0.0935
median_list_ppsf	0.0891
new_listings	0.0712

TableVII

Performance Metrics			
Strategy	RMSE	R^2	Train Time (s)
Baseline	1002501.14	0.080	0.00
TVS	1015714.02	0.056	589.55
CV	717.02	0.228	191.67

Insights:
The Cross Validator model significantly outperformed the others in both RMSE and R2, indicating a better fit and prediction accuracy.
TrainValidationSplit had the longest training time but did not improve performance over the baseline.
The Baseline model was the fastest but had relatively poor predictive power.

4.3 Linear Regression
State Market Tracker (20MB)

We trained and evaluated a **Linear Regression model** on the state_market_tracker.csv dataset to predict **median_sale_price** based on 13 housing market features. Three modeling strategies were used: Base Linear Regression, TrainValidationSplit and Cross Validation. For the TrainValidationSplit, we tuned over 25 parameter combinations while we used 5 folds in Cross Validation.

Table VIII

State Market Tracker - Linear Regression	
Top 5 Feature Importance	
median_list_price	0.1103
inventory_yoy	0.0979
median_sale_price_yoy	0.0994
median_list_ppsf	0.0904
avg_sale_to_list	0.0771

Table IX

Performance Metrics				
Model	RMSE	R^2	Time (s)	Best Params
Base Linear Regression	65.27	0.773	6.49	nil
TrainValidation Split (TVS)	65.27	0.773	10.50	regParam=0.1 elasticNet=1.0
CrossValidator (CV, 5-fold)	65.28	0.773	18.36	regParam=0.01 elasticNet=0.5

Insights:
Performance (RMSE & R2) is nearly identical across methods, suggesting the dataset is well-behaved and not highly sensitive to hyperparameters. Also, regularization is not impacting the model drastically it's close to standard least squares. Cross Validator took ~3x longer than base training, but it did not yield better results – this is common when the dataset isn't overly complex or overfitted.

4.4 Gradient Boosting Regression
Neighborhood Market Tracker (5GB)

We built and compared **Gradient Boosted Tree (GBT)** regression models to predict housing market trends – specifically the **median_sale_price** – using a dataset of neighborhood-level housing features. The objective was to evaluate and optimize the performance of GBT regression models on real estate data using different tuning strategies. We loaded the data and selected relevant features and prepared a clean data frame with the target column renamed as label. We used **Vector Assembler** to combine input features and applied **MinMaxScaler** to normalize feature vectors. For the model training approaches, we trained directly with default hyperparameters for the base GBT model, then we used **TrainValidationSplit** using a smaller range of **maxDepth** and **maxIter** for the first grid, then tried a larger and more complex search space for the second grid. We also used a 3-fold cross-validation with the first grid. The RMSE measures prediction error while the R^2 measures model fit quality and all models were evaluated on the same test data. Outputs include RMSE, R^2, training time, and best hyperparameters.

Table X

Neighborhood Market Tracker – Gradient Boosting Regression	
Top 5 Feature Importance	
median_list_price	0.1076
median_ppsf	0.0964
Median_sale_price_yoy	0.0961
inventory_yoy	0.0951
new_listings	0.0703

Table XI

Performance Metrics					
Model	RMSE	R^2	Time (s)	Best max Depth	Best max Iter
GBTRegressor	309885.077	0.659	95.39	null	null
TrainValidation (Simple Grid)	287952.75	0.705	171.99	5	10
TrainValidation (Extended Grid	281601.80	0.718	289.0	5	20
CrossValidator (Grid 1)	287952.75	0.705	331.6	5	10

Insights:
The Extended TrainValidationSplit model achieved the best accuracy, with the lowest RMSE and highest R2.
CrossValidator, while offering the same accuracy as the simple TVS, took nearly twice the time, showing the tradeoff between robustness and computational cost.
The Base model performed the worst, validating the importance of hyperparameter tuning.

5. Key Findings

Here is a summary of key findings from all regression models across four housing datasets and modeling techniques:
Log transformation on skewed target data (e.g. housing prices) significantly enhances model stability and accuracy. CrossValidator generally improves model robustness but at a high computational cost – worth it only when the data is complex or noisy. Linear models are efficient and surprisingly strong if data is clean and linear in nature. Gradient Boosted Trees and Random Forests benefit most from careful tuning via TrainValidationSplit.

6. Challenges and Limitations

Table XII

Challenge	Impact
High Compute Cost of CV	Long training times
Diminishing returns from tuning	Little performance gain, high compute cost.
Data size and processing time	Memory bottlenecks, long preprocessing
Model complexity vs. overfitting	Risk of overfitting on smaller datasets

7. Conclusion

This study explored the application of several regression models – including Random Forest, Gradient Boosting, and Linear Regression – across multiple housing datasets of varying granularity: county, zip code, state, and neighborhood levels. Using Apache Spark and distributed processing, we evaluated baseline models alongside tuned variants using TrainValidationSplit and Cross Validator, focusing on metrics like RMSE, R^2, training time, and best hyperparameters.

This comprehensive benchmarking provides a scalable framework for predicting housing trends and highlights the importance of aligning model complexity, resource constraints, and data behavior to achieve optimal results.

References

[1] Xu, K. (2023). Predicting housing prices and analyzing real estate markets in the Chicago suburbs using machine learning. *Journal of Student Research*, *11*(3). https://doi.org/10.47611/jsrhs.v11i3.3459

[2] Sharma, Hemlata, et al. "An Optimal House Price Prediction Algorithm: XGBoost." *Analytics*, vol. 3, no. 1, 1 Mar. 2024, pp. 30–45, www.mdpi.com/2813-2203/3/1/3#:~:text=Among%20them%2C%20XGBoost%20emerges%20as, https://doi.org/10.3390/analytics3010003.

[3] Jha, Shashi Bhushan, et al. "Housing Market Prediction Problem Using Different Machine Learning Algorithms: A Case Study." *ArXiv.org*, 2020, arxiv.org/abs/2006.10092. Accessed 18 May 2025.

[4] V. Gorle, *House prices prediction with regression*, Medium. [Online]. Available: https://medium.com/@venkataramanagorle/house-prices-prediction-with-regression-fe673a6dfa85

[5] Thuynyle, *Redfin housing market data* [Data set], Kaggle. [Online]. Available: https://www.kaggle.com/datasets/thuynyle/redfin-housing-market-data

[6] S. Ruiz, C. Ahamba, and D. Wilson, *Redfin Housing Market Forecasting with Apache Spark* [GitHub Repository]. [Online]. Available: https://github.com/Soligrl/CIS-5560

[7] J. Woo, *CIS 5200 – Big Data Analytics: Lab materials and course lectures*, unpublished instructional materials, California State University, Los Angeles, 2025.