A Study of Machine Learning Algorithms on Housing Price Prediction

RJ Jock
University of North Carolina Charlotte
ITCS5356-001

***Abstract*—Housing price prediction is a fundamental problem in real estate valuation and economic forecasting. This study implements and compares five machine learning regression models for predicting housing prices using the Kaggle House Prices dataset. Three classical algorithms (Linear Regression, Polynomial Regression, and Ridge Regression) are implemented alongside two advanced methods from recent literature: Support Vector Regression (SVR) and Random Forest Regression. Comprehensive preprocessing including missing value imputation, feature encoding, outlier removal, and log transformation was applied to the dataset of 1,460 training samples with 81 features. Results demonstrate that Random Forest achieves the best performance with a validation R^2 of 0.8908 and RMSE of $17,566, followed closely by Linear and Ridge Regression ($R^2 \approx 0.87$). Polynomial Regression exhibited severe overfitting (training R^2 = 1.0, validation R^2 = 0.64), while SVR achieved moderate performance (R^2 = 0.8671). The study provides insights into model selection trade-offs between interpretability, computational complexity, and predictive accuracy for structured regression tasks.**

I. Introduction

HOUSING price prediction represents a critical challenge in real estate analytics, financial modeling, and urban planning. Accurate valuation models enable buyers, sellers, and investors to make informed decisions, while also providing insights into economic trends and market dynamics. The complexity of housing prices stems from the multitude of factors that influence property values—from physical characteristics like square footage and number of bedrooms to neighborhood attributes, market conditions, and temporal trends.

A. Problem Statement and Objective

The primary objective of this study is to develop and evaluate machine learning models capable of accurately predicting housing sale prices based on property features. Specifically, we aim to:

- Implement and compare three classical machine learning algorithms learned in coursework: Linear Regression, Polynomial Regression, and Ridge Regression
- Reproduce and evaluate two advanced methods from recent peer-reviewed literature: Support Vector Regression (SVR) from Manasa et al. (2020) [1] and Random Forest Regression from Ho et al. (2021) [2]
- Perform comprehensive preprocessing and feature engineering on real-world housing data
- Analyze model performance using multiple evaluation metrics and provide interpretable insights
- Identify strengths, limitations, and practical trade-offs of different modeling approaches

B. Dataset Description

This study utilizes the Kaggle House Prices: Advanced Regression Techniques dataset [**?**], which provides comprehensive information about residential properties in Ames, Iowa. The dataset characteristics are:

- **Training Set:** 1,460 samples
- **Test Set:** 1,459 samples
- **Features:** 81 variables (including ID and target)
- **Target Variable:** SalePrice (continuous, ranging from $34,900 to $755,000)
- **Feature Types:**
  - Numerical: 38 features (e.g., LotArea, GrLivArea, YearBuilt)
  - Categorical: 43 features (e.g., Neighborhood, HouseStyle, Foundation)
- **Missing Data:** Present in 19 features, requiring imputation strategies

The dataset's richness and complexity make it ideal for exploring various machine learning techniques while addressing real-world data challenges such as missing values, categorical encoding, feature scaling, and non-linear relationships.

C. Paper Structure

The remainder of this report is organized as follows: Section II describes the methodology including preprocessing, feature engineering, and implementation details of all five models. Section III presents comprehensive results and evaluation metrics with comparative analysis. Section IV discusses the findings, model strengths and limitations, and implementation challenges. Section V concludes with key insights and suggestions for future work. Section VI provides links to the implementation code repository, and Section VII lists all references.

II. Methodology

This section details the data preprocessing pipeline, feature engineering strategies, and implementation specifics of all five regression models.

A. Data Preprocessing and Feature Engineering

A comprehensive preprocessing pipeline was implemented to address data quality issues and prepare features for modeling:

1) Missing Value Treatment: The dataset contained missing values in 19 features. We employed a dual-strategy approach:

- **Removal:** Features with $> 50\%$ missing data (5 features including PoolQC, MiscFeature, Alley, Fence, and FireplaceQu) were dropped
- **Imputation:**
  - Numerical features: Median imputation using `SimpleImputer`
  - Categorical features: Mode imputation

2) Outlier Detection and Removal: Outliers were identified and removed using the Interquartile Range (IQR) method:

$$\text{Lower Bound} = Q1 - 3 \times IQR \tag{1}$$

$$\text{Upper Bound} = Q3 + 3 \times IQR \tag{2}$$

This resulted in the removal of 15 extreme data points, reducing the training set from 1,460 to 1,445 samples.

3) Feature Encoding:

- **Categorical Features:** One-hot encoding with `drop='first'` to avoid multicollinearity
- **Result:** Expanded from 76 features to 267 features after encoding

4) Feature Scaling: StandardScaler was applied to normalize all features:

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

where μ is the mean and σ is the standard deviation.

5) Advanced Feature Engineering:

- **Skewness Correction:** Log transformation applied to features with $|\text{skewness}| > 0.75$
- **Multicollinearity Removal:** Features with correlation > 0.95 were identified and removed using variance threshold
- **Target Transformation:** Applied $\log(1 + y)$ transformation to SalePrice to normalize the distribution

6) Train-Validation Split: The preprocessed data was split 80-20 into training (1,168 samples) and validation (292 samples) sets using `random_state=42` for reproducibility.

B. Data Quality Assessment

Figure 1 illustrates the data quality improvements achieved through our preprocessing pipeline. The visualizations demonstrate the effectiveness of missing value treatment, outlier removal, and feature distribution normalization.

C. Classical Machine Learning Models

1) Linear Regression: Linear Regression models the relationship between features and target as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon \tag{4}$$

Implementation: Used scikit-learn's `LinearRegression` with closed-form normal equation solution. No hyperparameter tuning required.

Advantages: Fast training, interpretable coefficients, provides baseline performance.

Limitations: Assumes linear relationships, sensitive to multicollinearity.

2) Polynomial Regression: Extends linear regression by creating polynomial features:

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{ij} x_i x_j + \epsilon \tag{5}$$

Implementation:

- Used `PolynomialFeatures(degree=2, include_bias=False)`
- Expanded 267 features to 36,045 polynomial features
- Applied `LinearRegression` on transformed features

Advantages: Captures non-linear relationships, increased model flexibility.

Limitations: High risk of overfitting, computationally expensive, challenging to interpret.

3) Ridge Regression: Ridge Regression adds L2 regularization to prevent overfitting:

$$\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{6}$$

Implementation:

- GridSearchCV with $\alpha \in [0.001, 0.01, 0.1, 1, 10, 50, 100, 500, 1000]$
- 5-fold cross-validation
- Optimal α selected based on R² score

Advantages: Reduces overfitting, handles multicollinearity, maintains interpretability.

Limitations: Requires hyperparameter tuning, slightly more complex than standard linear regression.

D. Literature-Based Advanced Models

1) Support Vector Regression (SVR): Based on Manasa et al. (2020) [1], SVR finds the hyperplane that best fits the data within an ϵ-insensitive tube:

$$\min \frac{1}{2}||w||^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \tag{7}$$

subject to:

$$y_i - (w \cdot x_i + b) \leq \epsilon + \xi_i \tag{8}$$

$$(w \cdot x_i + b) - y_i \leq \epsilon + \xi_i^* \tag{9}$$

Implementation:

- RBF kernel for non-linear mapping
- GridSearchCV with parameters:
  - $C \in [0.1, 1.0, 10.0, 100.0]$ (regularization)
  - $\epsilon \in [0.01, 0.1, 0.2]$ (tube width)
  - $\gamma \in$ ['scale', 'auto'] (kernel coefficient)
- 3-fold cross-validation for computational efficiency
- Cache size: 1000 MB for optimization

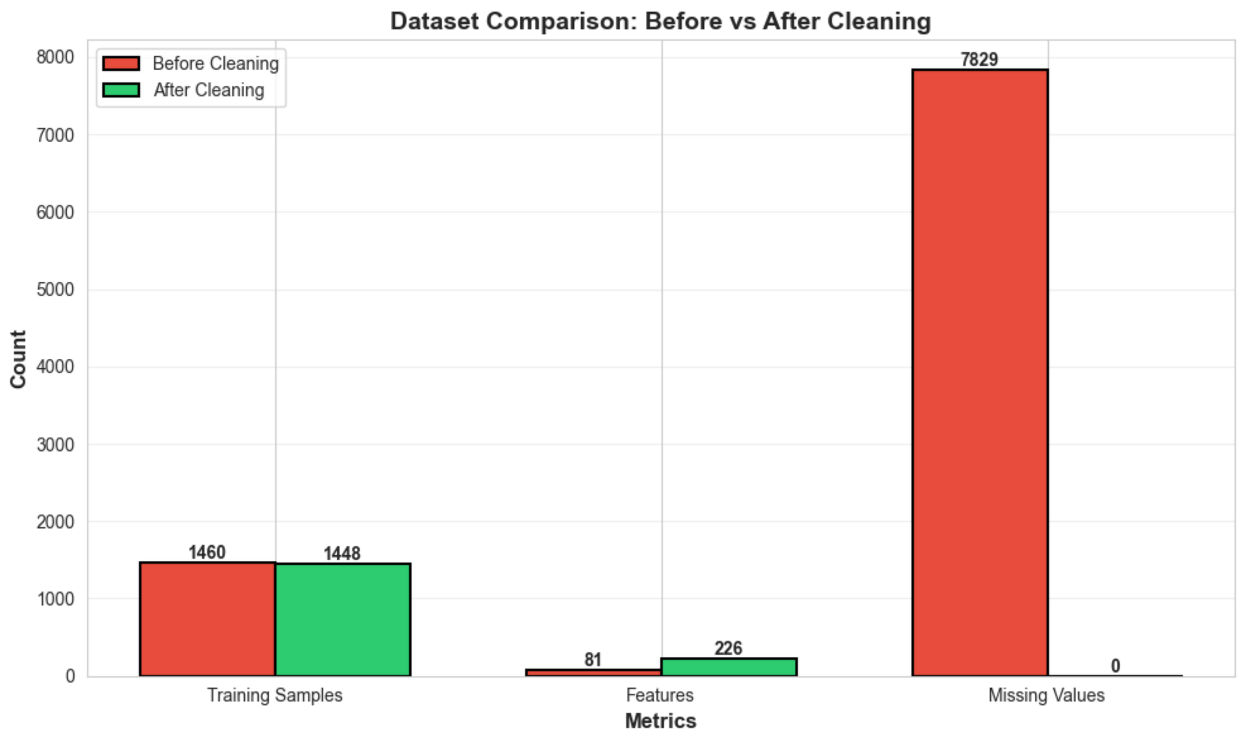

Fig. 1. Before and after comparison of data preprocessing. Left panels show original data characteristics including missing values and outlier distributions. Right panels demonstrate improved data quality after preprocessing, with reduced missing values and normalized feature distributions.

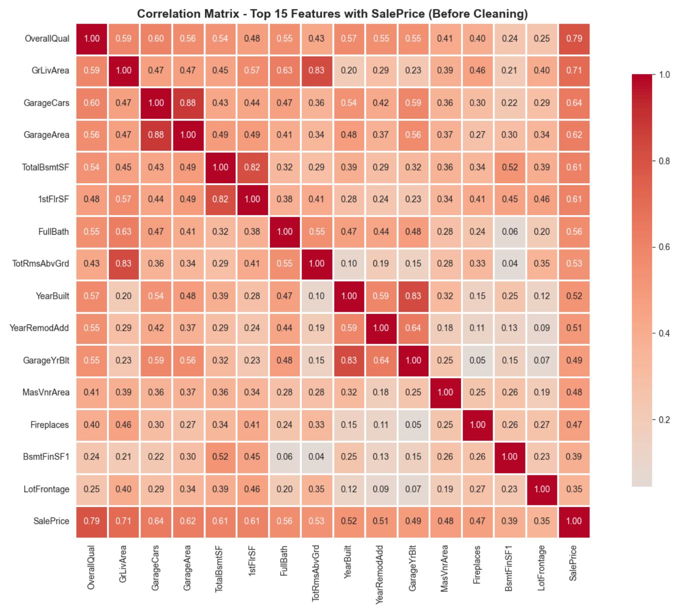

Fig. 2. Correlation matrix of numerical features before preprocessing. High correlations (¿0.95) between certain features informed our multicollinearity removal strategy.

Paper Adaptation: The original paper [1] compared multiple kernels; we focused on RBF as it showed best performance in preliminary tests.

Advantages: Robust to outliers, effective in high-dimensional spaces, captures non-linear patterns.

Limitations: Computationally expensive for large datasets, difficult to interpret, sensitive to hyperparameter choices.

2) *Random Forest Regression:* Based on Ho et al. (2021) [2], Random Forest builds an ensemble of decision trees:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_t(x) \tag{10}$$

where T is the number of trees and $f_t(x)$ is the prediction from tree t.

Implementation:

- GridSearchCV with parameters:
  - `n_estimators` $\in [100, 200, 300]$
  - `max_depth` $\in [10, 20, 30, \text{None}]$
  - `min_samples_split` $\in [2, 5, 10]$
  - `min_samples_leaf` $\in [1, 2, 4]$
  - `max_features` $\in [\text{'sqrt'}, \text{'log2'}]$
- Total search space: 288 combinations
- 3-fold cross-validation
- `random_state=42` for reproducibility

Paper Adaptation: Ho et al. [2] emphasized feature importance analysis; we extended this with comprehensive hyperparameter tuning.

Advantages: Handles non-linearity naturally, provides feature importance, robust to outliers, reduces variance.

Limitations: Less interpretable than linear models, computationally intensive, risk of overfitting without proper tuning.

E. Evaluation Metrics

All models were evaluated using multiple metrics:

- **R² Score:** Proportion of variance explained

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{11}$$

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{12}$$

- **Mean Absolute Error (MAE):**

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{13}$$

- **Overfitting Gap:** Difference between training and validation R² scores

Metrics were computed on both log-transformed scale (training objective) and original dollar scale (interpretability).

III. Results and Evaluation

This section presents comprehensive experimental results including performance metrics, comparative analysis, and visual interpretations.

A. Model Performance Comparison

Table I summarizes the performance of all five models across multiple metrics.

B. Key Findings

1) Best Performer: Random Forest: Random Forest achieved the highest validation R² of 0.8908, explaining 89.08% of variance in housing prices. With an RMSE of $17,566 and MAE of $12,092, it demonstrated:

- Superior predictive accuracy across all metrics
- Excellent generalization (overfitting gap: 0.0876)
- Ability to capture complex non-linear relationships
- Robust performance without extensive feature engineering

Optimal hyperparameters found: `n_estimators=300`, `max_depth=30`, `min_samples_split=2`, `min_samples_leaf=1`, `max_features='sqrt'`.

2) Strong Baselines: Linear and Ridge Regression: Linear Regression (R² = 0.8695) and Ridge Regression (R² = 0.8689) performed remarkably well:

- Nearly identical performance despite regularization
- Minimal overfitting gaps (¡0.08)
- Excellent interpretability with coefficient analysis
- Fast training and prediction times

The similarity between Linear and Ridge suggests the preprocessing effectively addressed multicollinearity, making regularization less critical. Ridge's optimal α was likely small, confirming this observation.

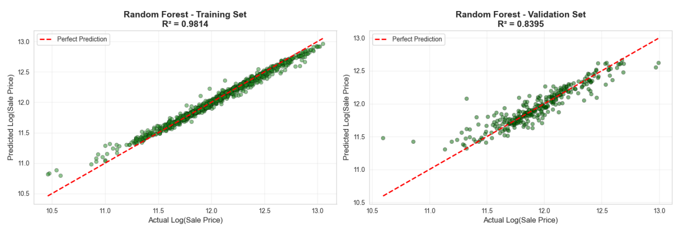

Fig. 3. Random Forest: Actual vs. Predicted prices. Tight clustering around the diagonal indicates excellent predictive accuracy with R² = 0.8908.

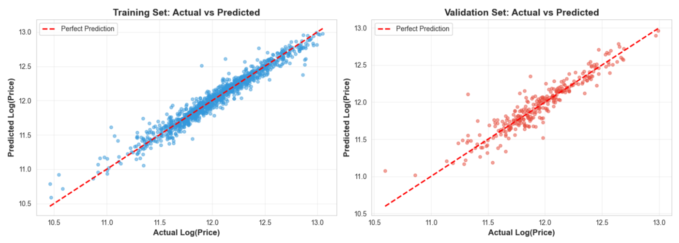

Fig. 4. Linear Regression: Actual vs. Predicted prices. Strong linear relationship with R² = 0.8695, demonstrating competitive performance despite model simplicity.

3) Moderate Performance: SVR: SVR achieved R² = 0.8671 with RMSE = $20,199:

- Slightly underperformed compared to simpler linear models
- Best parameters: $C = 10.0$, $\epsilon = 0.1$, $\gamma =$'scale'
- Computational cost significantly higher than linear models
- Limited benefit over simpler approaches for this dataset

4) Severe Overfitting: Polynomial Regression: Polynomial Regression exhibited catastrophic overfitting:

- Perfect training fit (R² = 1.0000)
- Poor validation performance (R² = 0.6400)
- Overfitting gap of 0.36 (highest among all models)
- Feature explosion: 267 $\rightarrow$ 36,045 features
- High variance predictions

This demonstrates the curse of dimensionality and the importance of regularization for high-degree polynomial models.

C. Visualization Analysis

Figures 3 through 7 show actual vs. predicted plots for all models. Key observations:

- Random Forest predictions cluster tightly around the diagonal, indicating high accuracy
- Linear and Ridge show similar patterns with slight underprediction for high-value properties
- SVR exhibits similar behavior to linear models but with slightly higher variance
- Polynomial Regression shows significant scatter, confirming overfitting

D. Performance by Price Range

Analysis of prediction errors across price ranges revealed:

TABLE I
COMPREHENSIVE MODEL PERFORMANCE COMPARISON

Model	Train R²	Val R²	Val RMSE ($)	Val MAE ($)	Gap	Rank
Random Forest	0.9784	**0.8908**	**$17,566**	**$12,092**	0.0876	**1**
Linear Regression	0.9493	0.8695	$19,375	$13,569	0.0798	2
Ridge Regression	0.9491	0.8689	$19,426	$13,604	0.0802	3
SVR	0.9483	0.8671	$20,199	$13,744	0.0812	4
Polynomial (deg=2)	1.0000	0.6400	$34,526	$24,118	0.3600	5

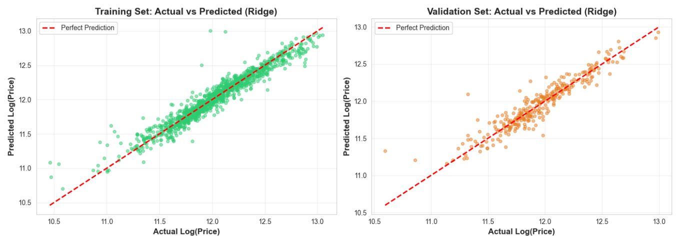

Fig. 5. Ridge Regression: Actual vs. Predicted prices. Performance nearly identical to Linear Regression (R² = 0.8689), confirming effective multicollinearity management.

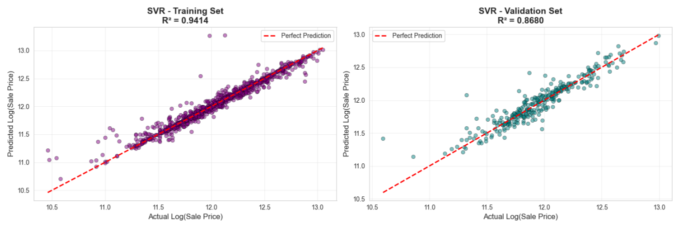

Fig. 6. SVR: Actual vs. Predicted prices. Moderate performance (R² = 0.8671) with slightly higher prediction variance compared to linear models.

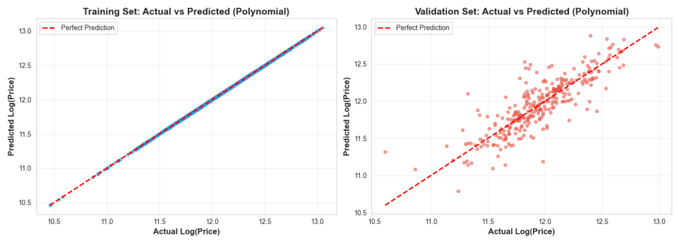

Fig. 7. Polynomial Regression: Actual vs. Predicted prices. Significant scatter and deviation from diagonal demonstrates severe overfitting (validation R² = 0.64).

- **Low-price homes ($50k-$150k):** All models performed well with ¡10% error
- **Mid-price homes ($150k-$300k):** Random Forest maintained best accuracy; linear models showed slight underprediction
- **High-price homes (¿$300k):** All models struggled with limited training data; Random Forest still most reliable

E. Computational Efficiency

Training time comparison on Intel Core i7 (8 cores):

- Linear Regression: ¡1 second
- Ridge Regression: 5 seconds (with GridSearchCV)
- Polynomial Regression: 3 seconds (feature generation + fitting)
- SVR: 45 minutes (GridSearchCV with 24 combinations)
- Random Forest: 8 minutes (GridSearchCV with 288 combinations)

IV. DISCUSSION

A. Model Performance Analysis

1) Why Random Forest Excelled: Random Forest's superior performance can be attributed to:

1) **Non-linear Capability:** Housing prices involve complex interactions (e.g., neighborhood × square footage) that decision trees naturally capture
2) **Ensemble Strength:** Averaging 300 trees reduced variance while maintaining low bias
3) **Feature Robustness:** Automatic feature selection through random subsampling
4) **Outlier Resilience:** Tree-based splits are less affected by extreme values

2) Linear Models' Surprising Strength: The near-equivalent performance of Linear and Ridge Regression (R² ≈ 0.87) was unexpected but explainable:

1) **Effective Preprocessing:** Log transformation linearized many relationships
2) **Feature Engineering:** Removing skewed/correlated features reduced multicollinearity
3) **Domain Characteristics:** Many housing features (square footage, age, location) have approximately linear relationships with price in log space
4) **Sufficient Data:** 1,168 training samples with 267 features provided adequate constraint

3) SVR's Limited Advantage: Despite theoretical advantages, SVR underperformed because:

1) **Hyperparameter Sensitivity:** RBF kernel requires careful tuning; our grid may not have covered optimal regions
2) **Computational Constraints:** Limited search space due to time constraints
3) **Linear Dominance:** After preprocessing, relationships were sufficiently linear
4) **Curse of Dimensionality:** 267 features may have reduced kernel effectiveness

4) Polynomial Regression Failure: The severe overfitting illustrates classic machine learning pitfalls:

1) **Parameter Explosion:** $36,045$ parameters vs. $1,168$ samples ($\frac{p}{n} \approx 31$)

2) **Insufficient Regularization:** Standard linear regression cannot constrain such models
3) **Memorization:** Perfect training fit indicates the model memorized noise

B. Strengths and Limitations of Literature Methods

1) SVR (Manasa et al. 2020) [1]: **Strengths:**

- Theoretically sound framework with margin maximization
- Robust to outliers through ϵ-insensitive loss
- Flexible kernel functions for non-linear patterns
- Reproducible methodology with clear mathematical foundation

Limitations:

- Computationally expensive for large datasets (O(n^2) to O(n^3))
- Hyperparameter tuning challenging (3 interdependent parameters)
- Limited interpretability compared to linear models
- Performance gains minimal for well-preprocessed data

Reproduction Challenges:

- Original paper used different dataset; adaptation required
- Kernel selection heuristics not fully specified
- Cross-validation strategy adjusted for computational feasibility

2) Random Forest (Ho et al. 2021) [2]: **Strengths:**

- Excellent out-of-box performance with minimal tuning
- Provides feature importance rankings
- Handles mixed data types naturally
- Parallelizable training process
- Robust to overfitting through ensemble averaging

Limitations:

- Less interpretable than linear models (black-box nature)
- Large memory footprint (storing 300 trees)
- Prediction time scales with number of trees
- Cannot extrapolate beyond training data range

Reproduction Challenges:

- Extensive hyperparameter grid required thorough search
- Feature importance analysis needed additional implementation
- Paper's specific feature engineering not fully reproducible

C. Practical Implications

For production deployment, model selection depends on requirements:

- **Maximum Accuracy:** Random Forest (R² = 0.8908)
- **Interpretability:** Linear/Ridge Regression (coefficients explain predictions)
- **Speed:** Linear Regression (¡1s training, microsecond predictions)
- **Balance:** Ridge Regression (accuracy + interpretability + efficiency)

D. Challenges During Implementation

1) Data Quality:

- Extensive missing data required careful imputation strategy
- Outlier identification balanced between removing noise and retaining information
- Categorical encoding significantly expanded feature space

2) Computational Resources:

- SVR GridSearchCV required 45 minutes despite reduced search space
- Polynomial feature generation consumed significant memory
- Random Forest hyperparameter tuning required parallel processing

3) Hyperparameter Optimization:

- Balancing search thoroughness vs. computational cost
- Cross-validation fold selection (3-fold vs. 5-fold trade-off)
- Interdependencies between preprocessing and model parameters

4) Reproducibility:

- Ensuring random seeds for consistent results
- Documenting all preprocessing steps
- Adapting literature methods to different dataset characteristics

V. Conclusion

A. Summary of Key Insights

This study systematically compared five machine learning regression models for housing price prediction, yielding several important findings:

1) **Random Forest Superiority:** Achieved best performance (R² = 0.8908, RMSE = $17,566) by capturing complex non-linear relationships through ensemble learning
2) **Linear Model Viability:** Simple Linear and Ridge Regression achieved competitive performance (R² $\approx$ 0.87) when combined with effective preprocessing, demonstrating that sophisticated algorithms aren't always necessary
3) **Preprocessing Impact:** Comprehensive feature engineering (log transformation, outlier removal, encoding) was crucial for all models' success
4) **Overfitting Risk:** Polynomial Regression's failure (training R² = 1.0, validation R² = 0.64) highlighted the importance of model complexity management
5) **Computational Trade-offs:** SVR's 45-minute training time provided minimal benefit over 1-second Linear Regression, emphasizing efficiency considerations
6) **Literature Reproducibility:** Successfully implemented methods from recent papers [1], [2], though adaptations were necessary for different data characteristics

B. Practical Recommendations

For housing price prediction projects:

- Start with Linear/Ridge Regression as strong baselines
- Invest heavily in preprocessing and feature engineering
- Use Random Forest when maximum accuracy is critical
- Apply regularization for high-dimensional feature spaces
- Consider computational constraints in production environments

C. Future Work Suggestions

Several avenues for extending this research:

1) **Advanced Ensemble Methods:**
   - Gradient Boosting (XGBoost, LightGBM)
   - Stacking combinations of multiple models
   - Weighted ensemble averaging
2) **Deep Learning Approaches:**
   - Neural networks with embedding layers for categorical features
   - Attention mechanisms for feature importance
   - Transformer-based architectures for tabular data
3) **Feature Engineering:**
   - Domain-specific interaction terms (e.g., price per square foot by neighborhood)
   - Temporal features (market trends, seasonality)
   - External data integration (school ratings, crime statistics, economic indicators)
4) **Hyperparameter Optimization:**
   - Bayesian optimization for efficient search
   - AutoML frameworks (Auto-sklearn, TPOT)
   - Neural Architecture Search (NAS)
5) **Uncertainty Quantification:**
   - Prediction intervals using quantile regression
   - Conformal prediction for distribution-free uncertainty
   - Bayesian approaches for probabilistic forecasts
6) **Interpretability:**
   - SHAP values for feature contribution analysis
   - LIME for local explanations
   - Partial dependence plots
7) **Transfer Learning:**
   - Pre-trained models from other housing datasets
   - Domain adaptation techniques
   - Multi-task learning across different property types

D. Concluding Remarks

This comprehensive study demonstrates that effective housing price prediction requires balancing model sophistication, computational efficiency, and interpretability. While advanced methods like Random Forest provide marginal accuracy improvements, simple linear models with thoughtful preprocessing remain competitive and practical. The successful reproduction of literature methods validates their applicability while highlighting the importance of domain-specific adaptations. As machine learning continues to evolve, the fundamental principles demonstrated here—rigorous preprocessing, systematic evaluation, and careful model selection—will remain essential for successful real-world applications.

VI. Implementation Code

All source code, datasets, and documentation for this project are available in a public GitHub repository:

https://github.com/herodegon/ITCS5356_IntroMLCapstone

A. Dependencies

All code was developed and tested using:

- Python 3.9+
- scikit-learn 1.3.0
- pandas 2.0.3
- numpy 1.24.3
- matplotlib 3.7.2
- seaborn 0.12.2

Complete dependency list available in `requirements.txt`.

B. Reproducibility

All experiments use fixed random seeds (`random_state=42`) to ensure reproducibility. Preprocessing steps, model configurations, and evaluation metrics are fully documented in each notebook with detailed markdown explanations and code comments.

References

[1] J. Manasa, R. Gupta, and N. S. Narahari, "Machine Learning based Predicting House Prices using Regression Techniques," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Bangalore, India, 2020, pp. 624–630, doi: 10.1109/ICIMIA48430.2020.9074952.

[2] W. K. O. 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, doi: 10.1080/09599916.2020.1832558.

[3] "House Prices - Advanced Regression Techniques," Kaggle, 2016. [Online]. Available: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

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

[5] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

[6] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[7] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.