Quantitative Social Science in the Age of Big Data and AI

Lab 7: Machine Learning to Prediction(I)

Zhaopeng Qu
Hopkins-Nanjing Center
June 03 2025

Review last three lectures

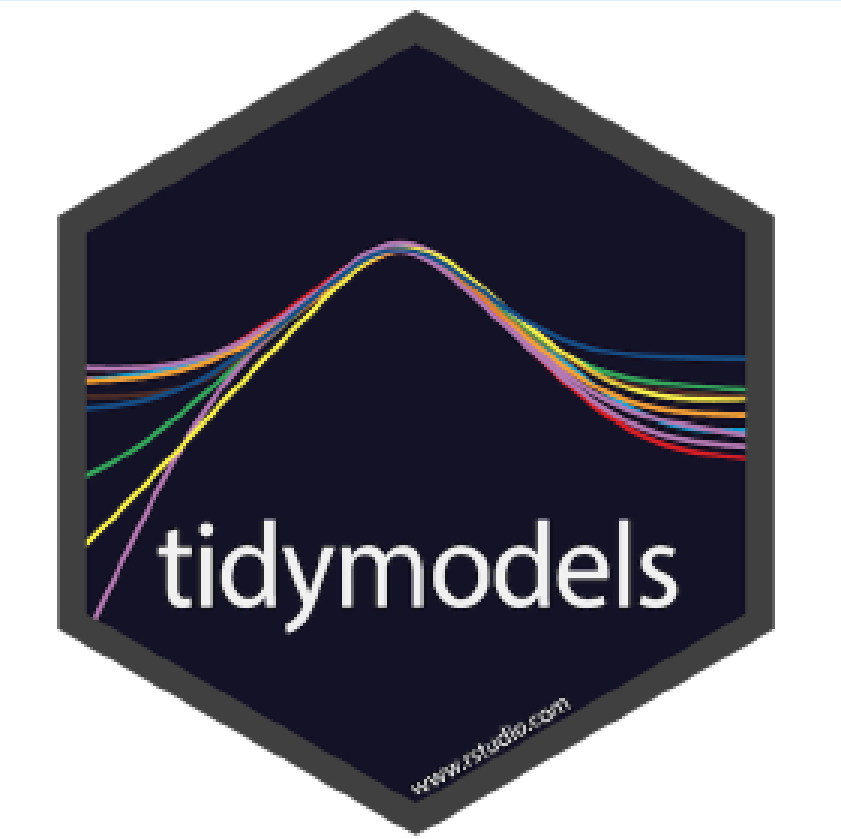
Machine Learning Basics

- Machine Learning is all about building models to predict outcomes.
- Supervised Learning: Predict outcomes based on features.
  - Regression: Predict continuous outcomes.
  - Classification: Predict discrete outcomes.
- Specific Algorithms:
  - Linear Regression: Predict continuous outcomes based on linear relationships between features and outcomes.
  - Ridge Regression: Regularize linear regression by adding a penalty term to the loss function.
  - Lasso Regression: Regularize linear regression by adding a penalty term to the loss function.
  - Logistic Regression: Predict discrete outcomes based on linear relationships between features and outcomes.

The workflow of machine learning

1. Define the Prediction Task
2. Explore the Data
3. Set Model and Tuning Parameters
4. Perform Cross-Validation
5. Evaluate the Models and Select the Best One or Ensemble Methods

The `tidymodels` package in R



> "`tidymodels` is a "meta-package" for modeling and statistical analysis that share the underlying design philosophy, grammar, and data structures of the tidyverse."

GET STARTED

Define the Prediction Task

Predict housing prices in Boston

- Dataset: 506 census tracts from the 1970 Boston census (Harrison & Rubinfeld, 1978)
- It includes 13 features that might influence housing prices, such as crime rate, property tax rate, and accessibility to employment centers.
- The target variable is the median home value (`medv`).

Source: https://www.bostonusa.com/

Variables in the `BostonHousing` Dataset

- `crim` – Per capita crime rate by town
- `zn` – Proportion of residential land zoned for lots over 25,000 sq. ft.
- `indus` – Proportion of non-retail business acres per town
- `chas` – Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox` – Nitric oxides concentration (parts per 10 million)
- `rm` – Average number of rooms per dwelling
- `age` – Proportion of owner-occupied units built prior to 1940
- `dis` – Weighted distances to five Boston employment centers
- `rad` – Index of accessibility to radial highways
- `tax` – Full-value property tax rate per $10,000
- `ptratio` – Pupil-teacher ratio by town
- `b` – 1000(Bk - 0.63)^2, where Bk is the proportion of Black residents by town
- `lstat` – % lower status of the population which is a proxy for the socioeconomic status of the population.
- `medv` – Median value of owner-occupied homes in $1000s

Load and Examine Data

First 6 rows of Boston Housing Dataset

ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
1	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
3	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
6	0.02985	0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

Data Quality Assessment

- Missing values

```
#> Missing values per column:

#>      ID    crim      zn   indus    chas     nox      rm     age     dis     rad
#>       0       0       0       0       0       0       0       0       0       0
#>     tax ptratio       b   lstat    medv
#>       0       0       0       0       0
```

Exploratory Data Analysis

Summary Statistics

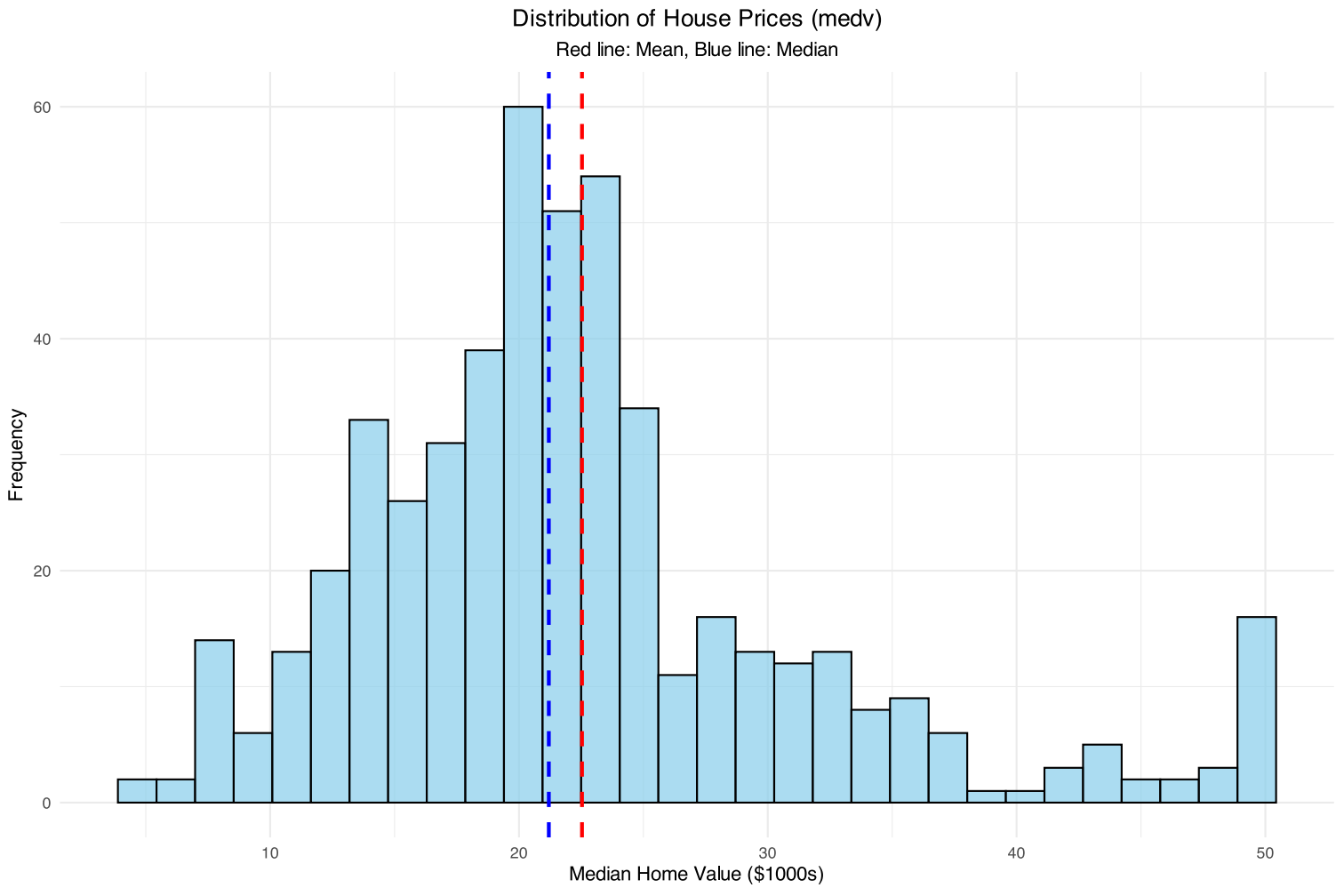
Variables 1-7

Variable	Mean	SD
crim	3.614	8.602
zn	11.364	23.322
indus	11.137	6.860
chas	0.069	0.254
nox	0.555	0.116
rm	6.285	0.703
age	68.575	28.149

Variables 8-13

Variable	Mean	SD
dis	3.795	2.106
rad	9.549	8.707
tax	408.237	168.537
ptratio	18.456	2.165
b	356.674	91.295
lstat	12.653	7.141
medv	22.533	9.197

Target Variable Distribution

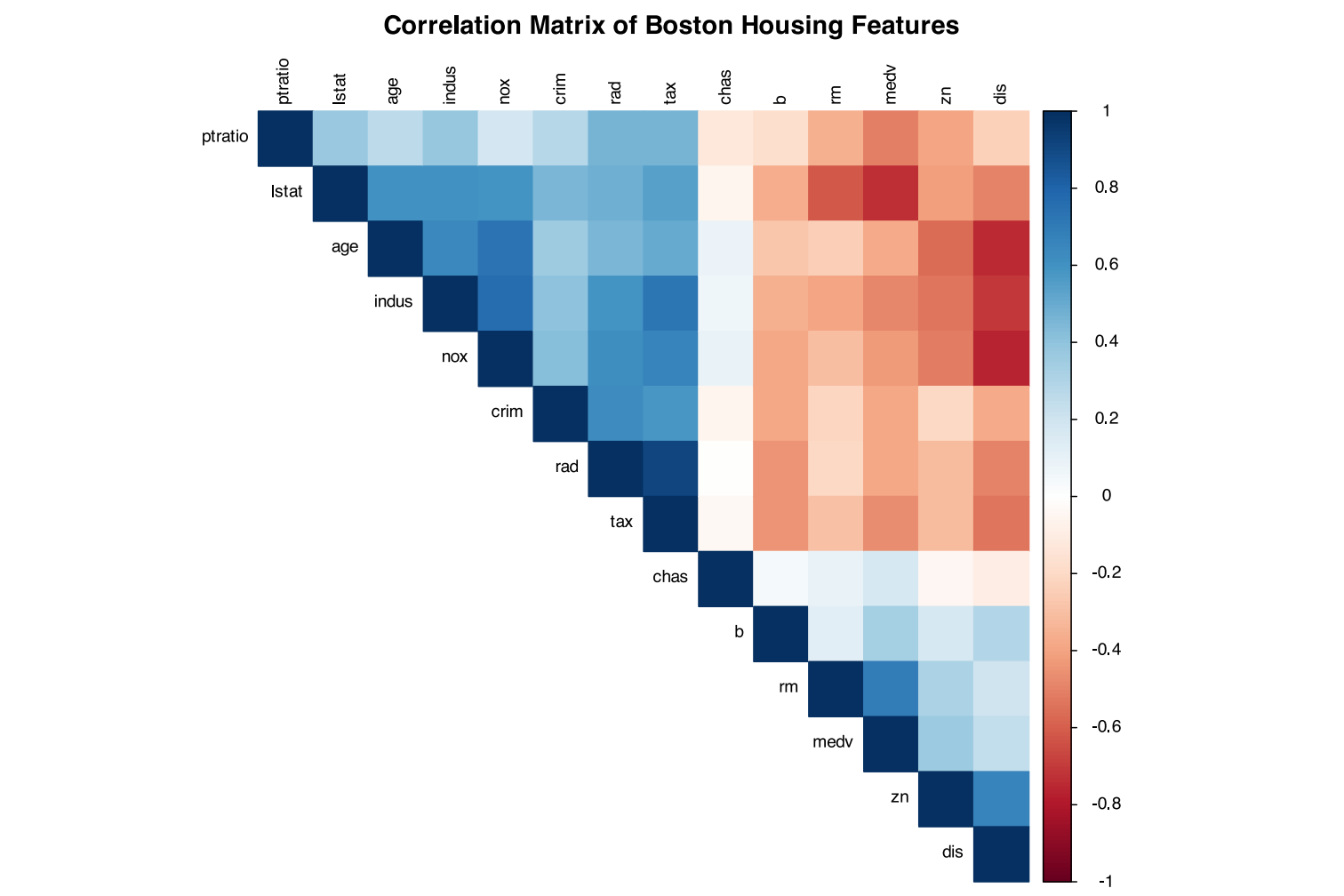


- Distribution of house prices shows a roughly normal distribution with some right skewness.

Basic statistics for target variable

```
#> Target Variable Statistics:

#> Mean: 22.53

#> Median: 21.2

#> Standard Deviation: 9.2
```

Correlation Analysis

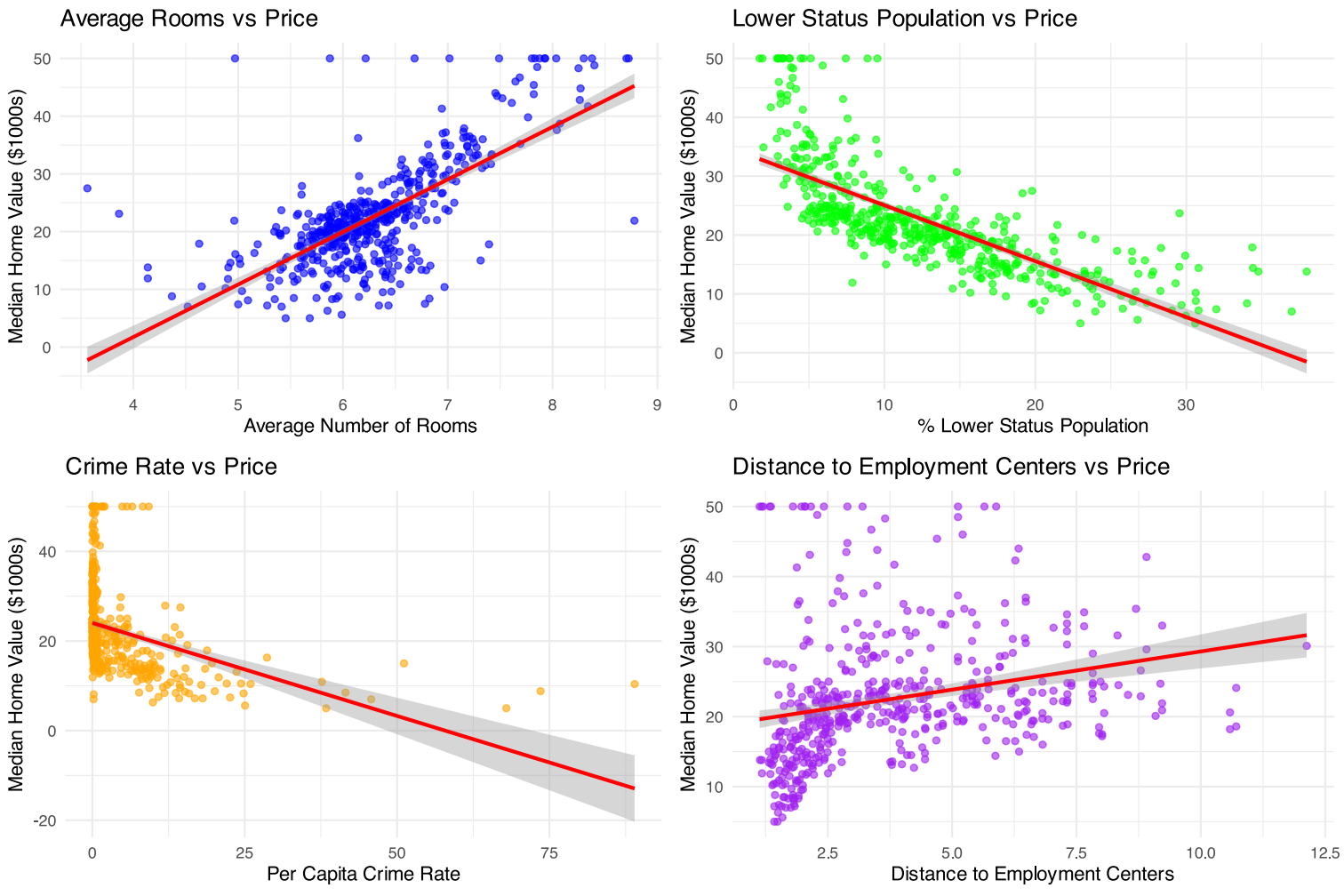


Display strongest correlations with target variable

```
#> Strongest correlations with house prices:

#> lstat : -0.738
#> rm : 0.695
#> ptratio : -0.508
#> indus : -0.484
#> tax : -0.469
#> nox : -0.427
#> crim : -0.388
#> rad : -0.382
#> age : -0.377
#> zn : 0.36
#> b : 0.333
#> dis : 0.25
#> chas : 0.175
```

- `lstat` – % lower status of the population which is a proxy for the socioeconomic status of the population.
- `rm` – Average number of rooms per dwelling
- `dis` – Weighted distances to five Boston employment centers
- `rad` – Index of accessibility to radial highways
- `tax` – Full-value property tax rate per $10,000
- `ptratio` – Pupil-teacher ratio by town
- `b` – 1000(Bk - 0.63)^2, where Bk is the proportion of Black residents by town
- `chas` – Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

Scatter Plots of Key Features vs Housing Prices



Set Model and Tuning Parameters

Data Splitting and Scaling

Data Splitting

```
#> Data split summary:

#> Training set size: 407 samples

#> Test set size: 99 samples

#> Training set proportion: 0.804
```

Feature Scaling

```
#> Feature scaling completed.

#> Training features mean (should be ~0): 0

#> Training features std (should be ~1): 1
```

Machine Learning Models

Model Specifications

1. OLS Regression: `lm`
2. Ridge Regression: `glmnet` (with CV tuning)
3. Lasso Regression: `glmnet` (with CV tuning)
4. K-Nearest Neighbors: `knn` (with CV tuning)
5. Decision Tree: `rpart` (with CV tuning)
6. Random Forest: `rf` (with CV tuning)

Model Evaluation

- MAE: The mean absolute error,thus $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- MSE: The mean squared error,thus $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- R^2 Score: The coefficient of determination, which measures the proportion of variance in the dependent variable that is explained by the independent variables.

- Here we use MAE to evaluate the model performance.

Linear Models: OLS Regression

- OLS Regression is a linear regression model that minimizes the sum of squared errors between the predicted and actual values.
- The loss function is:

$$\text{Loss} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- The result of OLS Regression is:

```
#> === LINEAR REGRESSION ===

#>
#>  Linear Regression  Results:
#> R² Score: 0.7299
#> MAE: 3.2817
```

Linear Models: 2. Ridge Regression

- Ridge Regression is a regularization method that adds a penalty term to the loss function to prevent overfitting.
- The loss function is:

$$\text{Loss} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

- The regularization parameter λ is used to control the strength of the regularization.
- The result of Ridge Regression is:

```
#> === RIDGE REGRESSION ===

#>
#>  Ridge Regression  Results:
 #> R² Score: 0.7268
   #> MAE: 3.2172
  #> Best alpha: 0
 #> Best lambda: 0.4522
```

Linear Models: 3. Lasso Regression

- Lasso Regression is a regularization method that adds a penalty term to the loss function to prevent overfitting.
- The loss function is:

$$\text{Loss} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- The regularization parameter λ is used to control the strength of the regularization.
- The result of Lasso Regression is:

```
#> === LASSO REGRESSION ===

#>
#>  Lasso Regression  Results:
#> R² Score: 0.7289
#> MAE: 3.2741
#> Best alpha: 1
#> Best lambda: 0.01
```

Non-Linear Models: 1. K-Nearest Neighbors

- K-Nearest Neighbors (KNN) is a non-parametric model that predicts the target variable by averaging the values of the k nearest neighbors.
- For regression, the prediction formula is:

$$\hat{y} = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

- The distance metric used is typically Euclidean distance or other distance metrics.
- The parameter k controls the model complexity: smaller k leads to more complex models with higher variance, while larger k leads to simpler models with higher bias.
- Optimal k selection: The best k value should be chosen through cross-validation to balance bias and variance.

Non-Linear Models: 1. K-Nearest Neighbors

- The result of K-Nearest Neighbors is:

```
#> === K-NEAREST NEIGHBORS ===

#>
#>  KNN  Results:
#> R² Score: 0.7343
#> MAE: 2.837
#> Best k: 10
#> Cross-validation used for k selection
```

Non-Linear Models: 2. Decision Tree

- Decision Trees make predictions by learning simple decision rules inferred from the data features through recursive binary splitting.

- The algorithm minimizes the sum of squared errors at each split:

$$\text{SSE} = \sum_{i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \hat{y}_{R_2})^2$$

where R_1 and R_2 are the two regions created by the split.

- The optimal split is chosen by:

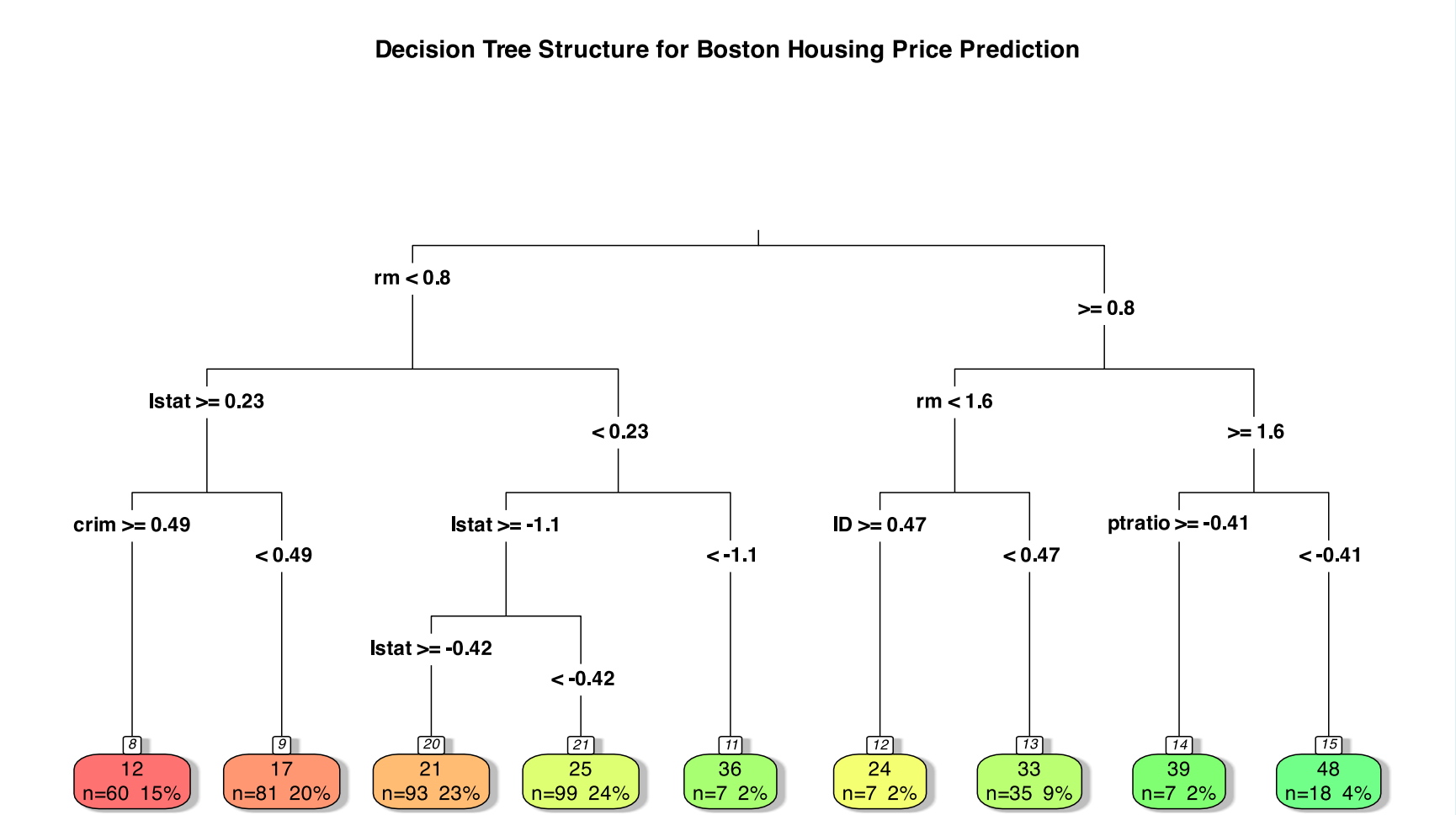
$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

where j is the variable and s is the split point.

- The model complexity is controlled by parameters like maximum depth, minimum samples per split, and minimum samples per leaf.

Decision Tree Structure Visualization

```
#> === DECISION TREE STRUCTURE ===
```



Decision Tree Performance

```
#> === DECISION TREE PERFORMANCE ===

#>
#>  Decision Tree  Results:
#> R² Score: 0.7193
#> MAE: 3.0648
#> Best cp (complexity parameter): 0.001
#> Cross-validation used for parameter selection
```

Non-Linear Models: 3. Random Forest

- Random Forest is an ensemble method that combines multiple decision trees using bootstrap aggregating (bagging) and random feature selection.The prediction is the average of all tree predictions:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

where B is the number of trees and $T_b(x)$ is the prediction from the b-th tree.

- Each tree is trained on a bootstrap sample of the data, and at each split, only a random subset of features (m_{try}) is considered:

$$m_{try} = \sqrt{p}$$

  where p is the total number of features.

- Key hyperparameters include number of trees (n_{tree}), features per split (m_{try}), and minimum node size.

Non-Linear Models: 3. Random Forest

- The result of Random Forest is:

```
#> === RANDOM FOREST ===

#>
#>  Random Forest  Results:
#> R² Score: 0.9146
#> MAE: 1.8222
#> Best mtry (variables per split): 6
#> Number of trees: 100
#> Cross-validation used for mtry selection
```

Model Comparison and Results

Initial Results Summary

Model Performance Comparison
(Sorted by MAE - Lower is Better)

	Name	MAE	R2_Score
6	Random Forest	1.8222	0.9146
4	KNN	2.8370	0.7343
5	Decision Tree	3.0648	0.7193
2	Ridge Regression	3.2172	0.7268
3	Lasso Regression	3.2741	0.7289
1	Linear Regression	3.2817	0.7299

- Random Forest is the best model with the lowest MAE of 1.8222.

Ensemble Methods

Ensemble Methods

What are Ensemble Methods?

Ensemble methods are machine learning strategies that combine multiple models to achieve better predictive performance than any individual model. The core idea is "wisdom of crowds beats individual expertise."

🎯 Core Principles of Ensemble Methods

1. Diversity: Different models have different strengths and weaknesses
2. Error Compensation: Errors from one model can be corrected by others
3. Variance Reduction: Averaging multiple models reduces overfitting
4. Robustness: Enhanced resistance to outliers and noise

Main Ensemble Strategies

1. Voting (Averaging)

- Principle: Multiple models "vote" on the final prediction
- Regression: Calculate weighted average of predictions
- Classification: Use majority voting or probability averaging

2. Stacking (Meta-Learning)

- Principle: Train a "meta-learner" to optimally combine base model predictions
- Two-Layer Structure: Base models + Meta model
- Intelligence: Meta model learns optimal combination strategy

Voting Regressor

📝 How It Works

Voting regressor combines predictions through weighted averaging:

$$\hat{y}_{voting} = \frac{w_1 \cdot \hat{y}_1 + w_2 \cdot \hat{y}_2 + \ldots + w_n \cdot \hat{y}_n}{w_1 + w_2 + \ldots + w_n}$$

where w_i is the weight for model i, and $\hat{y}_i$ is its prediction.

💡 Our Weighting Strategy

- Random Forest: Weight = 3 (highest weight, typically best performer)
- Ridge/Lasso/Linear: Weight = 2 (stable linear methods)
- KNN/Decision Tree: Weight = 1 (supplementary models)

Voting Regressor: Results

```
#> === VOTING REGRESSOR ===

#> Voting Regressor R²: 0.8336

#> Voting Regressor MAE: 2.409
```

Stacking Flow

🔄 Stacking Process Explanation

🎯 Layer 1: Base Models

- Input: Original Boston Housing data
- Process: 6 diverse models train independently
- Output: 6 different predictions for each house

🧠 Layer 2: Meta Learning

- Input: Predictions from all base models
- Process: Meta model learns optimal combination
- Output: Single improved prediction

💡 Key Advantages

🔄 Adaptive Combination

- Not fixed weights like voting
- Learns when to trust which model

🎯 Error Compensation

- One model's mistakes corrected by others
- Leverages model complementarity

📈 Superior Performance

- Typically outperforms individual models
- Reduces prediction variance

Mathematical Representation

Base Model Predictions:

$$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$$

- eg. use every model to predict the housing price, then we have 6 predictions for each house.

Meta Feature Construction:

$$\mathbf{X}_{meta} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n]$$

- eg. we have 6 predictions for each house, then we can construct a meta feature matrix.

Mathematical Representation

Meta Model Training:

$$\hat{y}_{stacking} = f_{meta}(\mathbf{X}_{meta}) = f_{meta}([\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n])$$

- eg. we can use a linear regression to predict the housing price based on the meta feature matrix.

Final Prediction:

$$\hat{y}_{stacking} = w_1\hat{y}_{LR} + w_2\hat{y}_{Ridge} + w_3\hat{y}_{Lasso} + w_4\hat{y}_{KNN} + w_5\hat{y}_{DT} + w_6\hat{y}_{RF} + b$$

- eg. we can use a linear regression to predict the housing price based on the meta feature matrix.
- Essentially, stacking is a meta model that learns the optimal combination of base models.

Voting vs Stacking Comparison

Feature	Voting Regressor	Stacking Regressor
Complexity	Simple	Complex
Combination	Fixed weighted average	Learned optimal combination
Training Time	Fast	Slow
Adaptability	Low	High
Overfitting Risk	Low	Medium
Performance	Good	Usually Superior

Why Stacking is Often More Effective

1. Adaptive Weights: Meta model automatically learns optimal combinations
2. Non-linear Combinations: Can capture complex relationships between models
3. Conditional Strategies: Uses different strategies under different conditions
4. Feature Interactions: Leverages interactions between different model predictions

Stacking Regressor: Results

```
#> === STACKING REGRESSOR ===

#> Stacking Regressor R²: 0.9375

#> Stacking Regressor MAE: 1.654

#>
#> Meta-model coefficients:

#>                Estimate Std. Error    t value     Pr(>|t|)
#> (Intercept) -1.0323900 1.05280004 -0.9806135 3.293551e-01
#> lr           7.2894436 4.96412837  1.4684237 1.454001e-01
#> ridge       -0.3519687 0.83429428 -0.4218760 6.740997e-01
#> lasso       -6.9587685 5.58520101 -1.2459298 2.159537e-01
#> knn         -0.2082519 0.11878077 -1.7532460 8.289061e-02
#> dt          -0.2585475 0.08413162 -3.0731310 2.787056e-03
#> rf           1.5303228 0.11155947 13.7175513 5.699395e-24
```

Stacking Result Interpretation

Above meta-model coefficients tell us the importance of each base model in the final prediction:

- Positive coefficients: This model's prediction contributes positively to the final result
- Negative coefficients: This model may act as a "correction" in certain situations
- Coefficient magnitude: Reflects the importance of this model in the ensemble

Ensemble Method Practical Advantages

1. Error Type Complementarity

- Linear models: Good at capturing linear relationships, fast computation
- Tree models: Good at handling non-linear and feature interactions
- KNN: Good at local pattern sensitivity
- Ensemble: Integrates various advantages, reduces single model limitations

2. Prediction Stability

- Single model may perform poorly on some samples
- Ensemble method reduces prediction variability through "averaging effect"
- Improves generalization ability to new data

3. Business Application Value

- House Price Prediction: More accurate valuation aids investment decisions
- Risk Control: Reduces single model failure risk
- Market Analysis: Multi-angle understanding of housing price factors

Feature Importance Analysis

Understanding Feature Importance

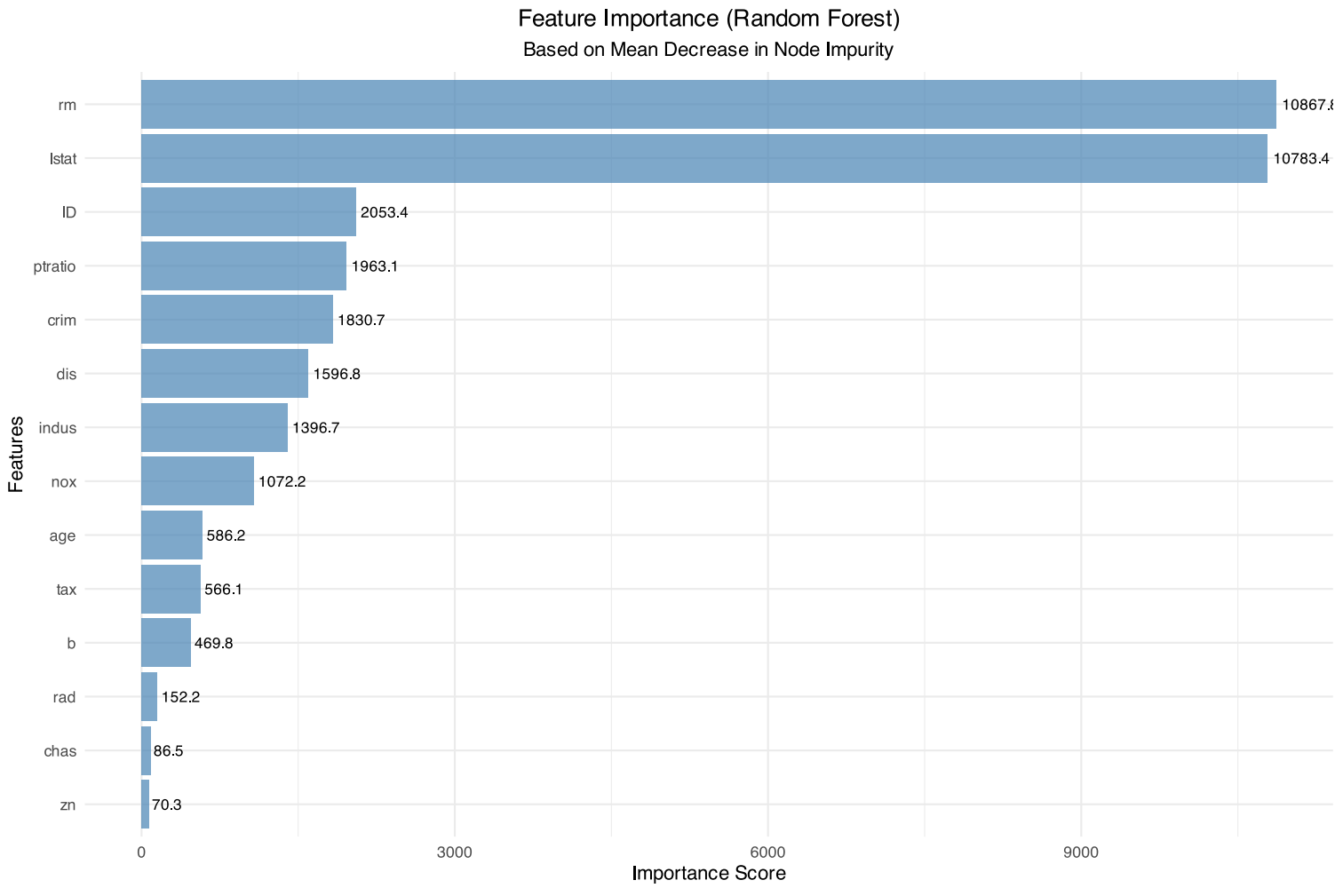
- Feature importance analysis helps us identify which variables are most influential for predicting house prices.
    - In decision trees, we use Gini Index to measure feature importance.
    - In random forests, we use Mean Decrease in Node Impurity to measure feature importance.
    - In linear models, we use Coefficient Magnitude to measure feature importance.
    - In KNN, we use Distance to Nearest Neighbors to measure feature importance.

🎯 Interpretation Principles

- Higher Importance Score → This feature is more important for prediction
- Relative Comparison: Importance is relative value, for feature comparison
- Practical Meaning: High importance feature is a major driver of house prices

Feature Importance Results

```
#> === FEATURE IMPORTANCE ANALYSIS ===
```



Final Results and Model Selection

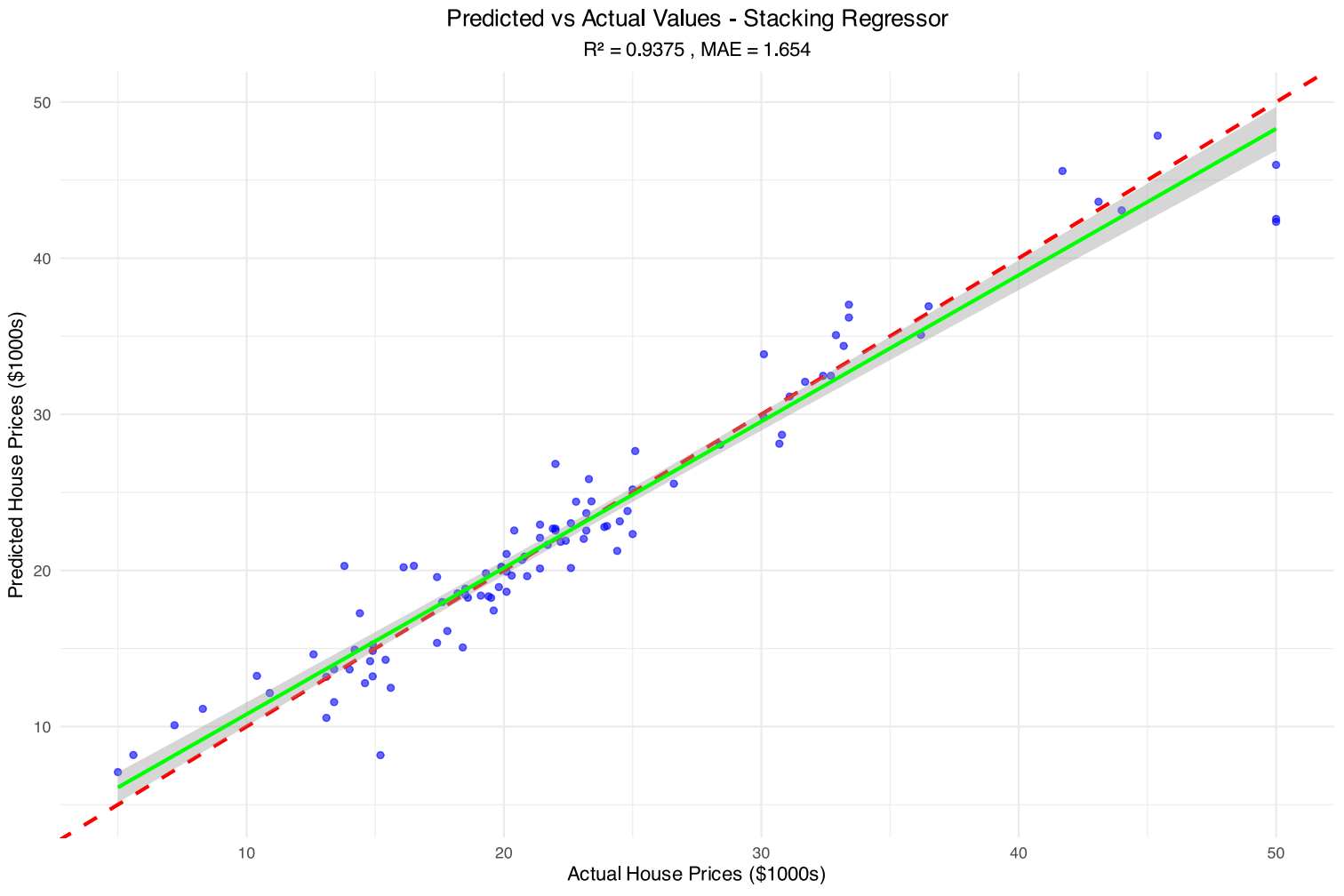
```
#> === FINAL RESULTS SUMMARY ===
```

Final Model Performance Ranking
(Sorted by MAE - Lower is Better)

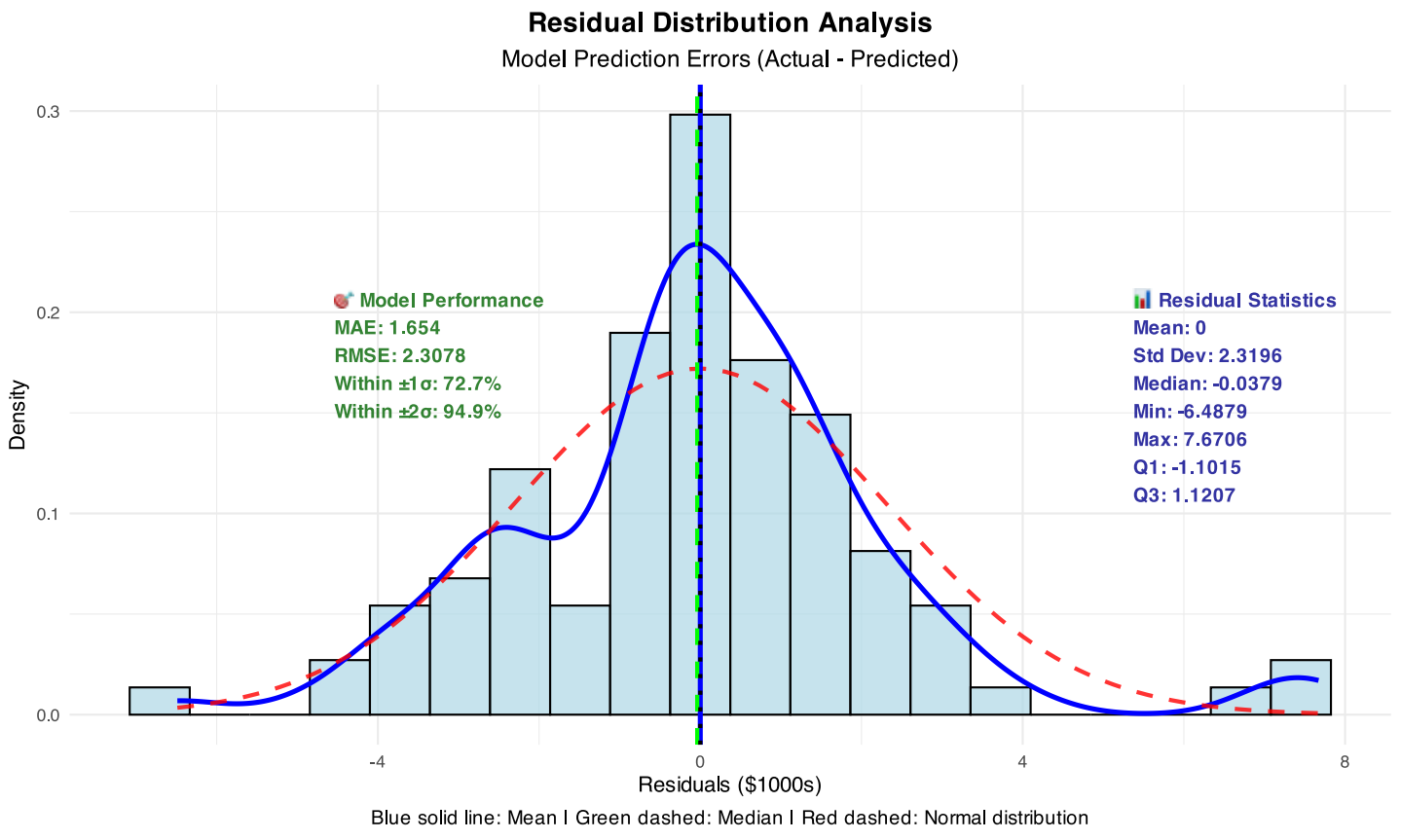
	Model	R2_Score	MAE
8	Stacking Regressor	0.9375	1.6540
6	Random Forest	0.9146	1.8222
7	Voting Regressor	0.8336	2.4090
4	KNN	0.7343	2.8370
5	Decision Tree	0.7193	3.0648
2	Ridge Regression	0.7268	3.2172
3	Lasso Regression	0.7289	3.2741
1	Linear Regression	0.7299	3.2817

```
#>
#> 🏆 BEST PERFORMING MODEL (Lowest MAE): Stacking Re

#> MAE: 1.654

#> R² Score: 0.9375
```

Prediction vs Actual Visualization



Calculate and display residual statistics



Detailed Residual Analysis

```
#>
#> ================================================
#> 🔍 DETAILED RESIDUAL ANALYSIS
#> ================================================

#> 📊 Basic Statistics:

#>      Mean Std_Dev     Min    Max  Median      Q1     Q3
#> 25%     0  2.3196 -6.4879 7.6706 -0.0379 -1.1015 1.1207

#>
#> 🎯 Model Performance Metrics:

#> MAE (Mean Absolute Error): 1.654

#> RMSE (Root Mean Square Error): 2.3078

#> MAPE (Mean Absolute Percentage Error): 8.94 %

#>
#> 📍 Outlier Analysis:

#> Houses with large prediction errors (>2σ): 5 out o

#> Outlier indices: 39 44 80 81 99
#> Outlier residuals: 7.67 -4.82 7.48 -6.49 7.03

#>
#> ✅ Model Diagnosis:

#> ✓ No systematic bias (mean ≈ 0)

#> ✓ Approximately symmetric distribution
```

Conclusions and Key Insights

Summary of Findings

1. Best Performing Model: The Stacking Regressor achieved the best performance with the lowest MAE of 1.654 and R^2 score of 0.9375.

2. Key Predictive Features:

   - `rm` (average number of rooms): Strong positive correlation with house prices
   - `lstat` (% lower status population): Strong negative correlation with house prices
   - `dis` (distance to employment centers): Moderate influence on pricing

3. Model Performance Hierarchy (Based on MAE):

   - Models are ranked by Mean Absolute Error (lower is better)
   - Ensemble Methods and Tree-based models typically perform well
   - Linear models provide good baseline performance

4. Data Quality: The dataset had no missing values and showed good feature diversity for prediction.

Recommendations

1. **For Production Use**: Deploy the Stacking Regressor as it provides the most accurate predictions

2. **For Interpretability**: Use Random Forest or Decision Tree for feature importance insights

3. **For Speed**: Linear regression provides fast predictions with reasonable accuracy

4. **For Robustness**: The voting regressor offers good performance with lower complexity