Supervised Learning and Model Evaluation: A Case Study with the Boston Housing Dataset

This report provides a comprehensive overview of machine learning concept, with a focus on supervised learning, the concepts of training and test sets, and the theoretical foundations of Decision Tree, Random Forest and Ridge Regression methods. Using the Boston Housing dataset, we illustrate how supervised learning is applied to predict house prices. The report explains the distinctions between supervised, unsupervised, reinforcement, and generative AI, the purpose of training and test sets, and the rationale for choosing the selected methods.

## 1 Introduction

Machine learning (ML) enables systems to learn from data, with supervised learning being a key paradigm for predictive tasks. This report uses the Boston Housing dataset (506 observations, 14 variables, including 13 features like crime rate and number of rooms, and a target variable, median house price, MEDV) to illustrate supervised learning. We cover the three supervised learning tasks—classification, regression, and ranking—focusing on regression for predicting numerical house prices. We also detail the theory of Decision Tree and Random Forest regression models, explain training and test sets, and justify the choice of methods

## 2 Machine Learning Paradigms

Machine learning is divided into four main techniques:

* **Supervised Learning**: Uses labelled data, where inputs X (e.g., CRIM, RM) are paired with a target variable y (e.g., MEDV), to learn a mapping f(X) ≈ y. It is suitable for the Boston Housing dataset, where MEDV is provided.
* **Unsupervised Learning**: Analyzes unlabelled data to find patterns, such as clustering houses by similar features without a target.
* **Reinforcement Learning**: Involves an agent learning through trial and error to maximize rewards (e.g., optimizing a pricing strategy).
* **Generative AI**: Generates new data resembling the training set, such as synthetic house price data or images.

The Boston Housing analysis uses supervised learning because the dataset includes a labelled target (MEDV), enabling prediction of house prices.

## 3 Supervised Learning and Its Tasks

Supervised learning involves training a model on labelled data, where each input X (features) is associated with a known output y (target). The goal is to learn a function f(X) ≈ y that generalizes to new data. Supervised learning encompasses three main tasks:

* **Classification**: Predicts a categorical target variable. For example, classifying houses as” expensive” or” affordable” based on a price threshold.
* **Regression**: Predicts a continuous numerical target variable. In the Boston Housing dataset, the target MEDV (median house price in $1000s) is numerical, making regression the appropriate task. The code uses regression to predict house prices based on features like RM (number of rooms) and LSTAT (lower status population percentage).
* **Ranking**: Orders items based on relevance or preference, such as ranking houses by predicted value for a buyer. This task is not used in the provided code.

Since the Boston Housing dataset’s target variable (MEDV) is continuous (e.g., 22.5, 50.0), the code employs regression to predict house prices, using models like Decision Tree and Random Forest to estimate numerical values.

## 4 Training and Test Sets

To ensure a model generalizes to unseen data, the dataset is split into:

* **Training Set:** A subset (80% in the code, via test\_size=0.2) used to train the model. The algorithm learns patterns, such as how RM and LSTAT affect MEDV, by optimizing parameters.
* **Test Set**: A separate subset (20%) used to evaluate performance on unseen data, providing an unbiased measure of generalization.

**Why Use Training and Test Sets?**

* **Generalization**: The test set mimics real-world scenarios, ensuring the model predicts accurately on new data.
* **Overfitting** Detection: Poor test set performance despite good training performance indicates overfitting, where the model memorizes training data.
* **Unbiased** **Evaluation**: Separating the test set prevents data leakage, ensuring the model doesn’t learn from test data during training.

In the code, train\_test\_split with random\_state=42 ensures reproducible splits. Features are standardized using StandardScaler, fitted on the training set to avoid leakage, ensuring fair evaluation

5 Theory of the Methods

The code uses three regression models: Decision Tree, Random Forest, and RidgeRegression. Below, we detail their theoretical foundations and rationale for use.  
5.1 Decision Tree Regression

Decision Tree Regression is a non-linear model that partitions the feature space into regions based on feature thresholds and assigns a constant value (e.g., average MEDV) to each region.

The detailed theory includes:

* **Tree Structure**: The model builds a binary tree, where each node represents a decision based on a feature threshold (e.g., RM > 6.5). Leaf nodes contain the average target value of training samples in that region.
* **Splitting Criteria**: At each node, the algorithm selects the feature and threshold that minimize a loss function, typically mean squared error (MSE), defined as:

where is the actual value, is the predicted value, and n is the number of samples in the node.

* **Recursive Partitioning**: The algorithm recursively splits nodes until a stopping criterion is met, such as:
* Maximum depth (max\_depth=3 in the code), limiting tree complexity.
* Minimum samples per split (min\_samples\_split=15), ensuring robust splits.
* **Prediction**: For a new observation, the model traverses the tree based on feature values, reaching a leaf node and returning its average target value.
* **Pruning**: Post-training pruning (not implemented in the code) or pre-pruning (via parameters like max\_depth) reduces overfitting by limiting tree size.

**Why Use Decision Trees?**

* **Interpretability**: The tree structure visually explains decisions (e.g., how RM > 6.5 leads to higher MEDV).
* **Non-linearity**: Captures complex, non-linear relationships in the Boston Housing data, such as the interaction between LSTAT and MEDV.
* **Feature** **Importance**: Quantifies feature contributions (e.g., RM, LSTAT), aiding analysis.

In the code, a Decision Tree Regressor is trained with tuned parameters to balance complexity and generalization, achieving an *R*2≈ 0.77. Its simplicity makes it a good baseline, but it may struggle with complex patterns.

### 5.2 Random Forest Regression

Random Forest Regression is an ensemble method that combines multiple decision trees to enhance accuracy and robustness. The detailed theory includes:

* **Bagging** **(Bootstrap** **Aggregating)**: Each tree is trained on a random subset of the training data, sampled with replacement (bootstrap samples). This reduces variance by averaging predictions across diverse trees.
* **Feature** **Randomness**: At each node, a random subset of features is considered for splitting, decorrelating trees and preventing reliance on dominant features (e.g., RM).
* **Aggregation**: For regression, predictions from all trees are averaged:

where is the prediction from tree t, and T is the number of trees (n\_estimators=100 in the code).

* **Variance** **Reduction**: By combining trees, Random Forest reduces the overfitting common in single decision trees, improving generalization.
* **Out-of-Bag (OOB) Error**: Each tree is trained on 63% of the data (due to bootstrap sampling), leaving 37% as out-of-bag samples for internal validation (not used in the code but part of the theory).

**Why Use Random Forest?**

* **High Accuracy**: By averaging multiple trees, it captures complex patterns, achieving a higher *R*2 ≈ 0.89 in the code.
* **Robustness**: Less sensitive to outliers (e.g., high CRIM values) and noise, as seen in tighter residual distributions.
* **Feature** **Importance**: Provides reliable importance scores, identifying RM and LSTAT as key predictors of MEDV.
* **Versatility**: Handles non-linear relationships and interactions in the Boston Housing data effectively.

In the code, a Random Forest with 100 trees outperforms the Decision Tree, with lower MSE and higher *R*2, due to its ensemble approach.

### 6.3 Ridge Regression Ridge

Regression is a linear model with L2 regularization to prevent overfitting. The theory includes:

* **Linear Model:** Assumes a linear relationship:

where w is the weight vector, b is the bias, and X is the feature matrix.

* **L2 Regularization**: Adds a penalty to the loss function to shrink weights:

where α (alpha=1.0 in the code) controls regularization strength, and p is the number of features.

* **Closed-Form Solution**: Solves for weights using:

w =

where I is the identity matrix (bias term unregularized, as in the code).

* **Prediction**: Computes for new inputs.
* **Regularization** **Benefit**: Reduces overfitting by constraining large weights, improving generalization.

**Why Use Ridge Regression?**

* Stability: Handles multicollinearity in features (e.g., correlated variables like TAX and RAD).
* Simplicity: Assumes linear relationships, complementing non-linear models like Random Forest.
* Regularization: Prevents overfitting, suitable for the Boston Housing dataset’s complex feature interactions.

In the code, Ridge Regression is implemented from scratch, achieving competitive performance (e.g., R2 ≈ 0.74), balancing simplicity and robustness.

## 6 Model Evaluation

The code evaluates models using:

* **Mean Squared Error (MSE):** Measures average squared differences, emphasizing larger errors.
* **Root Mean Squared Error (RMSE):** Interpretable in $1000s (e.g., RMSE of 2.82 for Random Forest vs. 4.01 for Decision Tree vs 4.9 for Ridge).
* **Mean Absolute Error (MAE):** Average absolute differences, robust to outliers.
* ***R*2Score**: Proportion of variance explained (e.g., 0.89 for Random Forest vs. 0.77 for Decision Tree vs 0.67 for Ridge).

Residual plots show Random Forest’s residuals are more centred around zero, indicating better accuracy. Feature importance rankings highlight RM and LSTAT as dominant predictors.

## 7 Practical Insights

The Boston Housing analysis demonstrates:

* **Regression** **Task**: Chosen because MEDV is continuous, making regression ideal for predicting numerical house prices.
* **Data** **Preprocessing**: Log-transforming skewed features (e.g., CRIM, LSTAT) and standardizing improve model performance.
* **Model** **Choice**: Random Forest outperforms Decision Tree due to its ensemble nature and Ridge Regression for linear relationships and multicollinearity, suitable for the dataset’s complexity.
* **Evaluation**: Metrics like RMSE and *R*2confirm Random Forest’s superiority.

Future improvements could include hyperparameter tuning (e.g., grid search for n\_estimators), exploring Gradient Boosting, or handling outliers in CRIM.

## 8 Conclusion

Supervised learning, with its tasks of classification, regression, and ranking, is ideal for problems with labelled data, such as predicting MEDV in the Boston Housing dataset. Regression is used here due to the numerical nature of house prices. Training and test sets ensure robust evaluation, while Decision Tree, Random Forest and Ridge Regression offer complementary approaches. Random Forest’s ensemble method excels, achieving higher accuracy. This case study highlights the importance of preprocessing, model selection, and evaluation in supervised learning.

## References

1. Scikit-learn: Machine Learning in Python, <https://scikit-learn.org/stable/>.
2. Boston Housing Dataset, <http://lib.stat.cmu.edu/datasets/boston>.