Model Validation and Hyperparameter Tuning for House Price Prediction

Task 3 – AI & Machine Learning Internship

Name: Aryan Marota
Internship Organization: Maincraft Technologies
Task Title: Model Validation and Hyperparameter Tuning
Dataset: California Housing Dataset (Scikit-learn)

---

1. Introduction

In real-world machine learning applications, achieving high accuracy on training data is not sufficient. A reliable model must generalize well to unseen data and provide stable performance across different data samples. Overfitting, where a model learns noise instead of patterns, is a common issue that affects model reliability.

The objective of **Task 3** is to move beyond basic model building and focus on **model validation, overfitting detection, cross-validation, and hyperparameter tuning**. Using the California Housing dataset, this task evaluates model reliability and demonstrates how systematic tuning and validation techniques improve generalization performance.

This task emphasizes **scientific model selection**, rather than choosing a model solely based on a single accuracy score.

---

2. Dataset Description

The California Housing dataset is obtained from the Scikit-learn library and is based on data collected during the 1990 California census. It represents housing information aggregated at the district level.

Dataset Characteristics

- Total records: **20,640**
- Number of features: **8**
- Target variable: **HousePrice**
- Data type: Fully numerical
- Missing values: **None**

Input Features

- Median Income
- House Age
- Average Rooms
- Average Bedrooms

- Population
- Average Occupancy
- Latitude
- Longitude

The dataset was inspected using head() and found to be clean and suitable for regression modeling without additional preprocessing.

---

3. Data Preparation

3.1 Feature Scaling

All input features were scaled using **StandardScaler** to ensure that variables on different numeric ranges do not dominate the learning process. Feature scaling is essential for stable model training, especially for linear and distance-sensitive models.

3.2 Train–Test Split

The dataset was divided into:

- **80% training data**
- **20% testing data**

This split allows evaluation on unseen data and helps detect overfitting.

---

4. Overfitting Analysis

4.1 Baseline Decision Tree Model

A Decision Tree Regressor was trained without restricting model complexity. The performance of the model was evaluated on both training and testing datasets.

Observed Results (Code Block [6]):

- Training RMSE ≈ **0.00**
- Test RMSE ≈ **0.70**

Interpretation

The extremely low training error combined with a much higher test error clearly indicates **overfitting**. The model memorizes the training data instead of learning general patterns. This behavior is typical of tree-based models when depth and split constraints are not applied.

This analysis highlights the importance of controlling model complexity before deploying models in real-world scenarios.

---

5. Cross-Validation for Reliable Evaluation

To obtain a more stable and unbiased estimate of model performance, **5-fold cross-validation** was applied to the Decision Tree model.

Cross-Validation Result (Code Block [7]):

- Mean CV RMSE ≈ **0.8957**

Why Cross-Validation Matters

- Reduces dependency on a single train-test split
- Provides a more realistic estimate of generalization error
- Helps detect variance in model performance

The cross-validation score confirms that the baseline Decision Tree model is unstable and requires tuning to improve generalization.

---

6. Hyperparameter Tuning

6.1 GridSearchCV Approach

To reduce overfitting, **GridSearchCV** was used to systematically search for optimal hyperparameters. The following parameters were tuned:

- max_depth
- min_samples_split

This approach evaluates multiple parameter combinations using cross-validation and selects the configuration that minimizes RMSE.

Best Parameters Identified (Code Block [8]):

max_depth = 10

min_samples_split = 10

These constraints limit tree complexity and prevent the model from memorizing the training data.

---

7. Evaluation of Tuned Model

The tuned Decision Tree model was evaluated on the test dataset.

Tuned Model Performance (Code Block [9]):

- RMSE = **0.6454**
- R^2 Score = **0.6821**

Improvement Analysis

Compared to the untuned model:

- Test RMSE decreased significantly

- Generalization performance improved
- Overfitting was reduced

This demonstrates the effectiveness of hyperparameter tuning in improving model reliability.

---

8. Baseline Model Comparison

To justify final model selection, the tuned Decision Tree was compared against baseline linear models.

Baseline Results:

- **Linear Regression** (Code Block [10])
    - RMSE = **0.7456**
    - R^2 = **0.5758**
- **Ridge Regression** (Code Block [10])
    - RMSE = **0.7456**
    - R^2 = **0.5758**

---

9. Model Comparison Summary

The final comparison of models is summarized below (Code Block [11]):

Model	RMSE	R^2 Score
Linear Regression	0.7456	0.5758
Ridge Regression	0.7456	0.5758
Tuned Decision Tree	**0.6454**	**0.6821**

Observation

The tuned Decision Tree model clearly outperforms the baseline linear models in terms of both RMSE and R^2 score.

---

10. Final Model Selection Justification

The **Tuned Decision Tree Regressor** was selected as the final model based on the following reasons:

- Overfitting was detected and addressed through hyperparameter tuning
- Cross-validation confirmed more stable performance
- Lower RMSE indicates improved prediction accuracy
- Higher R^2 score shows better explanatory power

- Model complexity was controlled without sacrificing performance

This selection balances **performance, generalization, and reliability**, which aligns with professional machine learning practices.

---

11. Conclusion

Task 3 demonstrated the importance of validating machine learning models beyond basic accuracy metrics. Through overfitting analysis, cross-validation, and systematic hyperparameter tuning, the reliability of the Decision Tree model was significantly improved.

This task highlights that proper validation techniques are essential for deploying trustworthy machine learning models in real-world applications.

---

12. Future Scope

- Apply ensemble methods such as Random Forest
- Perform deeper hyperparameter optimization
- Compare validation results with additional regression models