Model Validation, Overfitting Control & Hyperparameter Tuning

---

1. Aim / Objective of the Task

The main objective of Task-3 is to understand and implement model validation, overfitting control, and hyperparameter tuning techniques in machine learning. This task demonstrates that high training accuracy alone is not sufficient; a model must also perform well on unseen data.

---

2. Dataset Description

The California Housing Dataset was used for this task.

Target Variable: Median House Value

Input Features include:

- Median Income
- House Age
- Average Rooms
- Population
- Location-based features

The dataset was preprocessed and Standard Scaling was applied to ensure fair and stable learning.

---

3. Tools and Technologies Used

The following tools and technologies were used:

- Python
- Jupyter Notebook
- NumPy
- Pandas
- Matplotlib
- Scikit-learn

---

4. Overfitting Analysis

Overfitting occurs when a model performs extremely well on training data but poorly on new or unseen data.

A Decision Tree Regressor was trained without applying constraints, and the training and testing RMSE values were compared.

Observation:

- Training RMSE was very low
- Testing RMSE was relatively higher

This large difference indicates that the model was overfitting. Decision Trees without restrictions can become overly complex and memorize the training data.

---

5. Cross-Validation Technique

To ensure reliable performance evaluation, 5-Fold Cross-Validation was applied instead of relying on a single train-test split.

In this method, the dataset was divided into five equal parts. The model was trained on four parts and validated on the remaining part. This process was repeated five times so that each fold was used as validation once.

This approach reduces random bias and provides a more accurate estimate of the model's generalization capability.

---

6. Hyperparameter Tuning Approach

The default Decision Tree model showed signs of overfitting.

To control model complexity, hyperparameters such as:

- max_depth
- min_samples_split

were tuned using GridSearchCV.

By limiting tree depth and increasing minimum samples required for splitting, model complexity was reduced and overfitting was controlled.

---

7. Model Evaluation Metrics

Model performance was evaluated using:

- RMSE (Root Mean Squared Error)
- R^2 Score

Interpretation:

- Lower RMSE indicates better prediction accuracy
- Higher R^2 indicates better explanatory power

The tuned model showed improvement in both metrics.

---

8. Model Comparison

The following models were compared:

1. Linear Regression
2. Ridge Regression
3. Tuned Decision Tree Regressor

Based on RMSE and R^2 scores, the Tuned Decision Tree captured complex patterns more effectively and delivered the best overall performance.

---

9. Final Model Selection and Justification

The Tuned Decision Tree Regressor was selected as the final model.

Reasons for selection:

1. Overfitting was successfully reduced
2. Cross-validation ensured reliable evaluation
3. Lower RMSE and higher R^2 score
4. Ability to handle non-linear relationships

Although slightly more complex, the model is suitable for practical deployment after tuning.

---

10. Conclusion

This task highlights the importance of validation and tuning in real-world machine learning projects. Cross-validation and GridSearchCV represent professional ML workflows. Tuned models provide more reliable and generalizable predictions compared to untuned models.

---

11. Final Outcome

- Overfitting analysis was successfully performed
- Hyperparameter tuning was applied
- The best model was scientifically selected
- Industry-aligned machine learning practices were followed