

Feature Selection & Model Selection – Part 2

Under/Overfitting, Bias-Variance,
Cross-Validation, Train/Val/Test
Split, Learning/Validation Curves

Underfitting

- Definition: Model is too simple to learn patterns in data.
- Example: Straight line on curve data → high training and test errors.
- Real-Life: Predict house price using only number of rooms.

Overfitting

- Definition: Model is too complex, memorizes noise.
- Example: 10-degree polynomial for 10 points behaves wildly.
- Real-Life: Predict house price using irrelevant features like zip code.
- Tip: High training accuracy but low validation accuracy = Overfitting.

Bias-Variance Tradeoff

- Bias: Error due to simple model (underfitting).
- Variance: Error due to sensitivity to data (overfitting).
- Goal: Balance both errors for best performance.
- Analogy: Guitar tuning – bias = tight string, variance = loose string.

Cross-Validation

- Definition: Evaluate model performance on multiple data splits.
- K-Fold CV: Split data into k parts, train on k-1, test on 1.
- Repeat for each fold, average performance.
- Tip: Use `sklearn.model_selection.cross_val_score()`.

Train/Validation/Test Splits

- Training: Train the model.
- Validation: Tune hyperparameters.
- Test: Final unbiased evaluation.
- Example: 1000 students → 600 train, 200 val, 200 test.

Learning Curve

- Shows performance vs. training set size.
- Helps detect underfitting and overfitting.
- High training error → underfitting.
- High training accuracy but low val accuracy → overfitting.

Validation Curve

- Plots performance vs. model complexity (e.g., depth, degree).
- Helps identify optimal hyperparameter value.
- x-axis: hyperparameter value, y-axis: accuracy.

Student Marks Prediction Project

- Inputs: Study hours, Attendance, Mid-term marks, Assignments.
- Models: Linear Regression, Decision Tree, Random Forest.
- Steps:
 1. Split dataset into train/val/test.
 2. Use cross-validation.
 3. Plot learning curve.
 4. Use validation curve to tune.
 5. Avoid overfitting with pruning.
 6. Test on unseen data only.

Summary Table

Concept	What it Solves	Tools in Python
Under/Overfitting	Model too simple/too complex	Learning Curves, <code>train_test_split</code>
Bias-Variance Tradeoff	Balancing generalization	Choose simpler/complex models
Cross-Validation	Better evaluation	<code>cross_val_score</code>
Train/Val/Test Split	Reliable testing and tuning	<code>train_test_split</code>
Learning/Val Curves	Diagnose & tune models	<code>validation_curve</code> , <code>learning_curve</code>

Project Description

- - Predict final exam scores based on:
- - • Study hours
- - • Attendance
- - • Midterm marks
- - • Assignments
- - Model: Linear Regression
- - Evaluation: Under/Overfitting, Bias-Variance, CV, Learning/Validation Curves

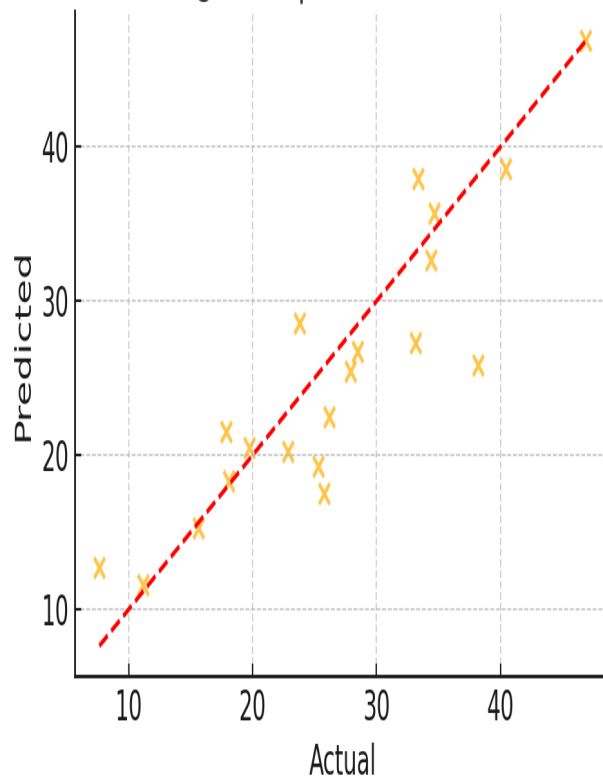
Student Marks Prediction Project

Feature Selection & Model Selection
Concepts with Graphs and
Explanations

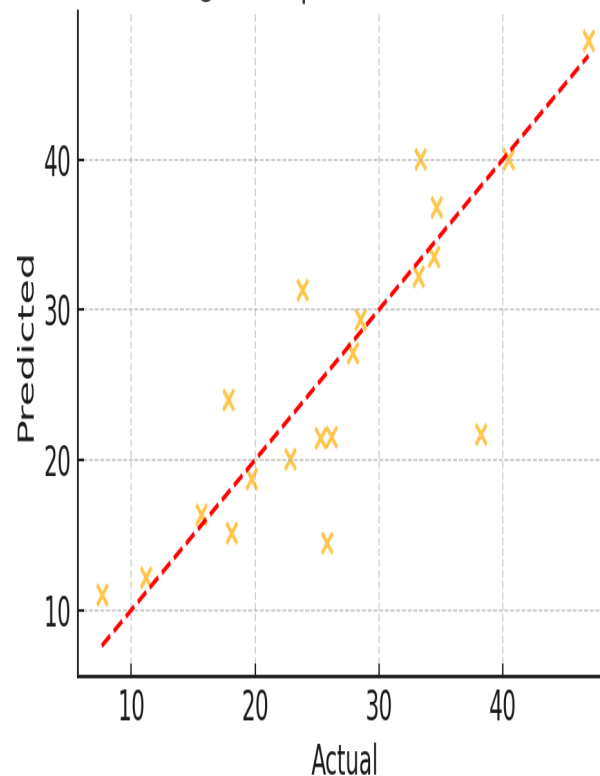
Underfitting vs Overfitting

Underfitting vs Overfitting

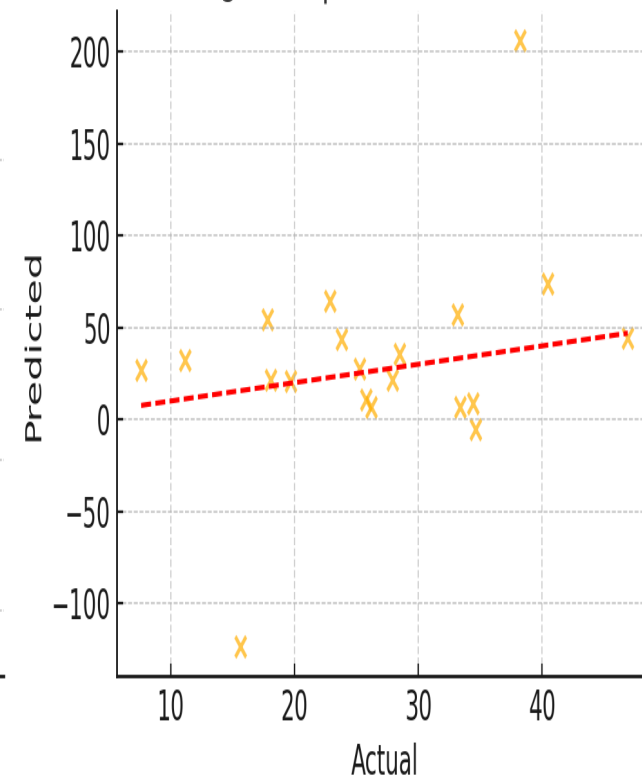
Degree 1 | Val MSE: 20.72



Degree 2 | Val MSE: 30.75

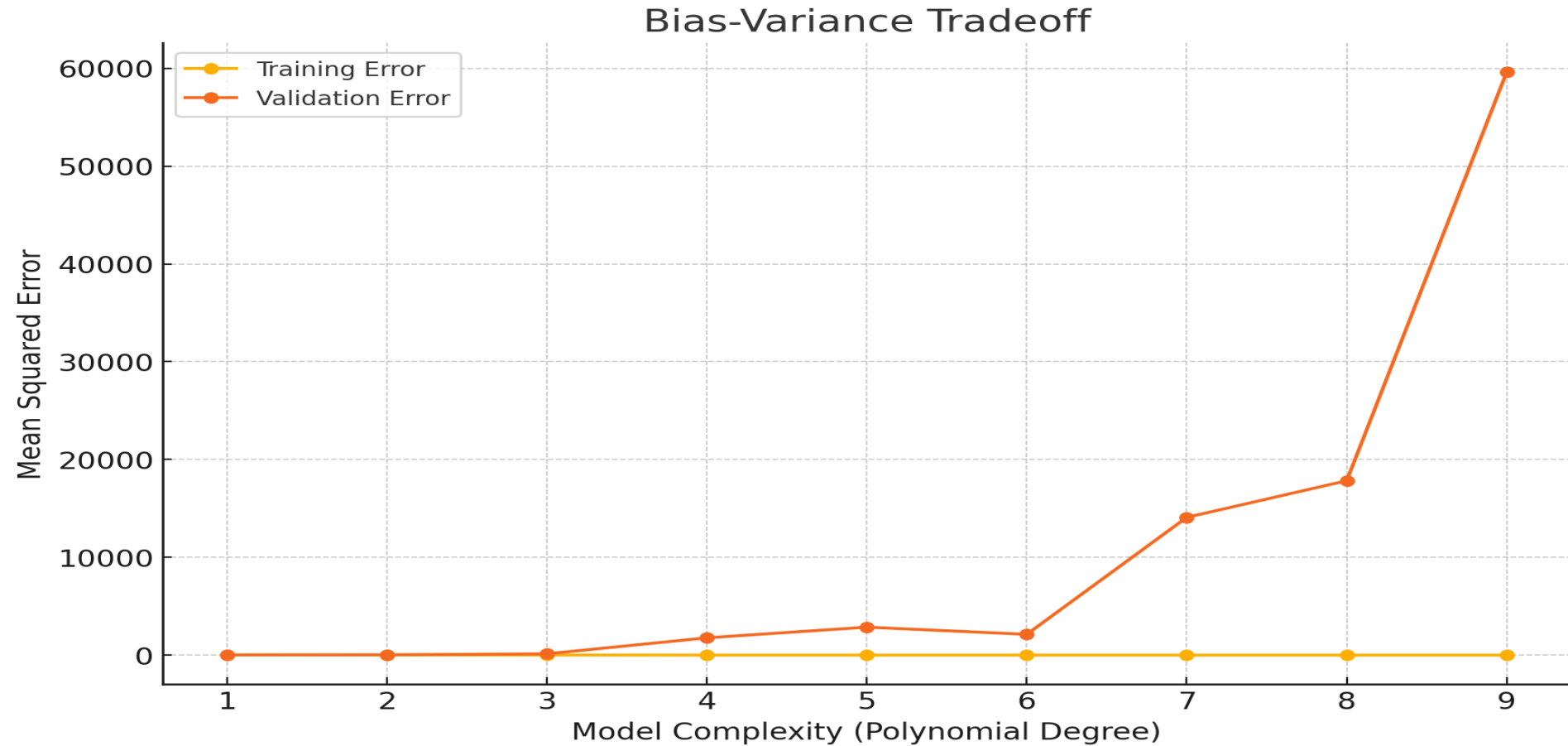


Degree 5 | Val MSE: 2857.88



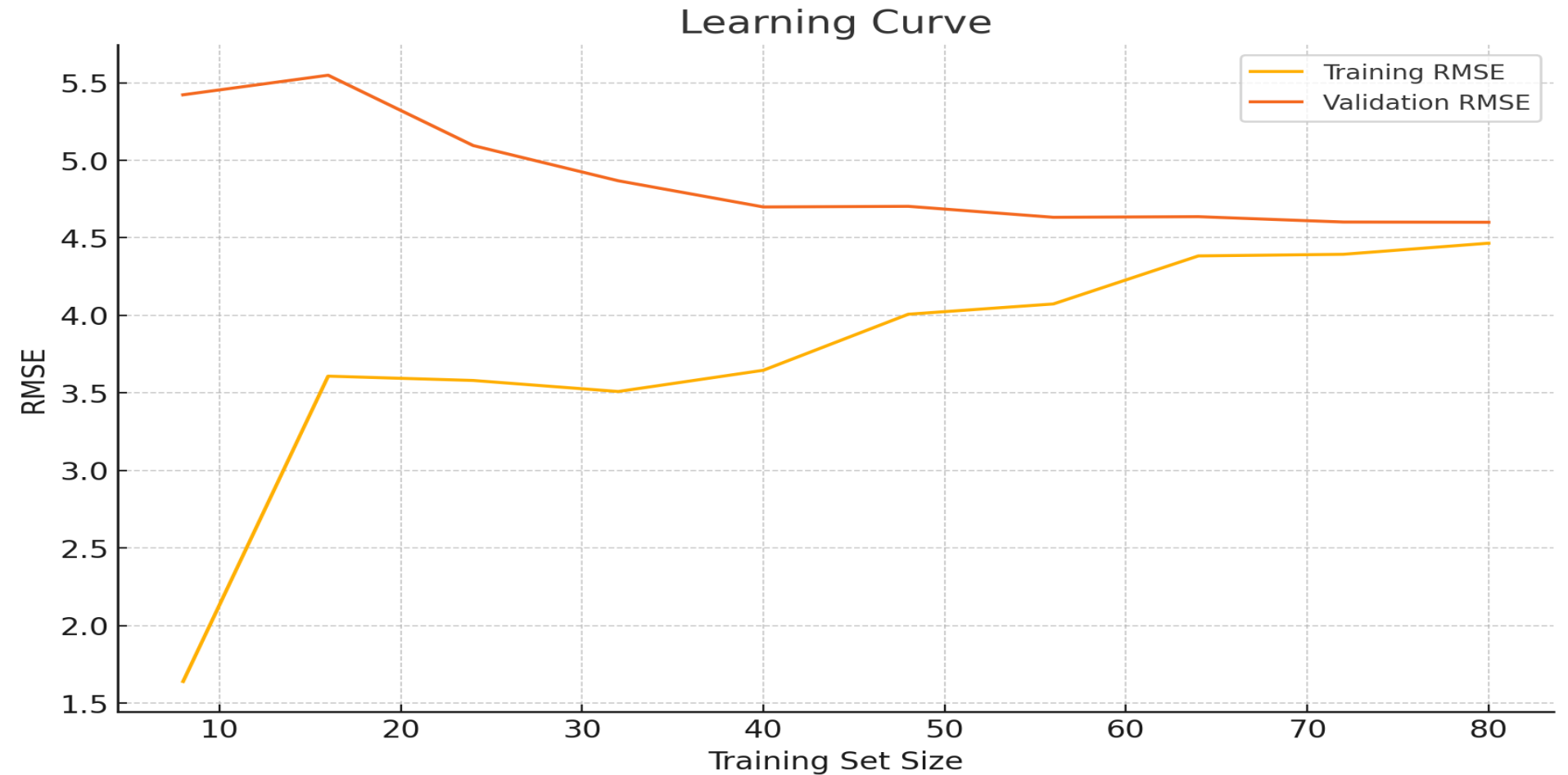
Degree 1: Too simple (underfit), Degree 2: Balanced fit, Degree 5: Too complex (overfit).

Bias-Variance Tradeoff



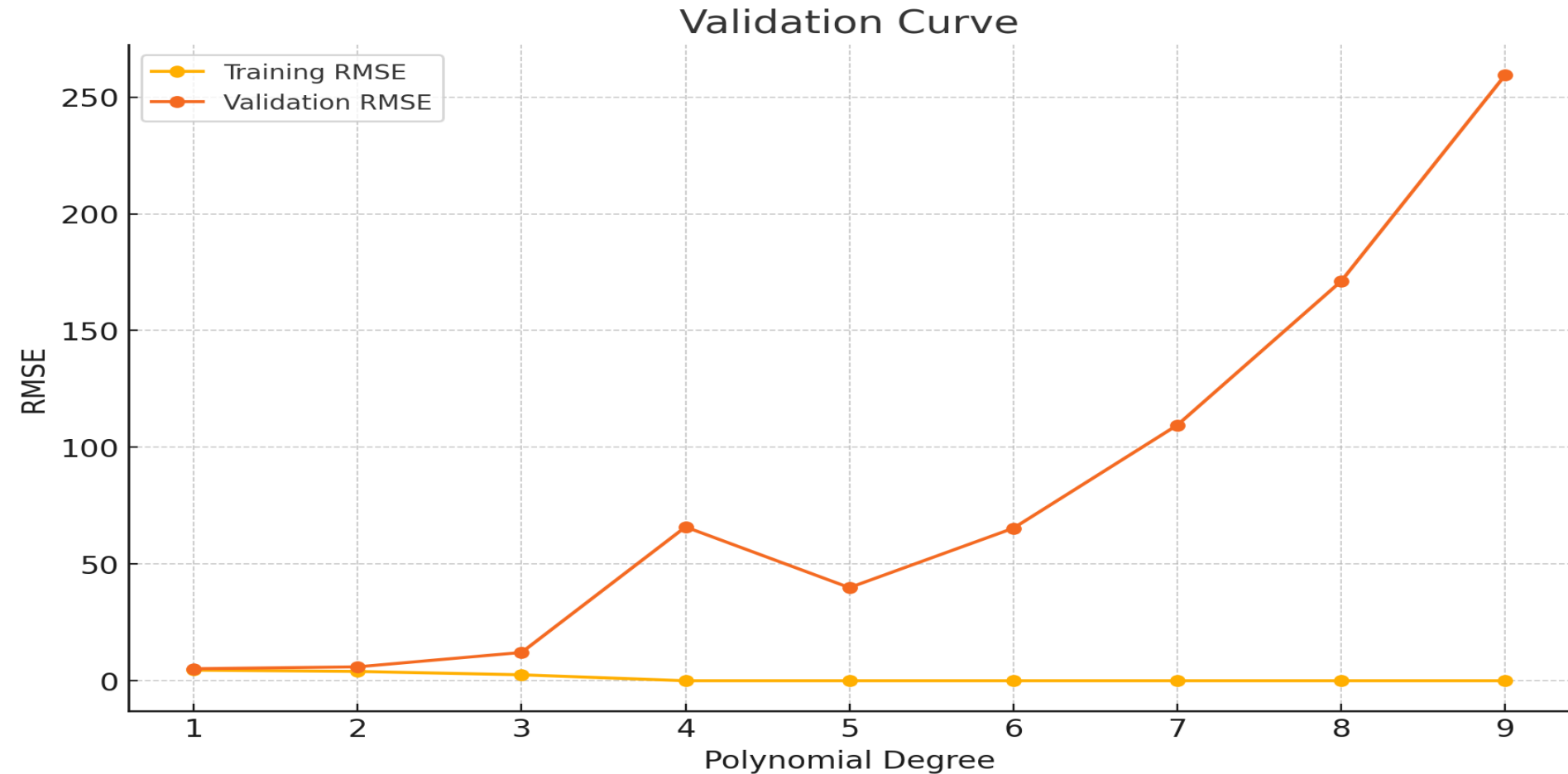
Low-degree models underfit (high bias); high-degree models overfit (high variance).

Learning Curve



Shows model improvement with more data. Validation RMSE decreases with more training samples.

Validation Curve



Validation RMSE vs polynomial degree. Helps find optimal model complexity.