Polynomial Regression: When Simple Lines Aren't Enough

Author: Mehr Dil
Student ID: 24087959
Github Link: https://github.com/MEHERDIL/polynomial_regression.git

---

Abstract

Linear regression is the foundation of machine learning, but real-world relationships are rarely perfectly linear. This tutorial explores polynomial regression—a powerful extension that allows linear models to capture non-linear patterns. Through systematic experimentation on house price prediction, we demonstrate how polynomial degree controls model complexity and directly impacts the bias-variance trade-off. We show that degree 1 underfits, degrees 2-3 are optimal, and degrees 10+ overfit dramatically, providing clear visual evidence of one of machine learning's most fundamental concepts.

---

1. Introduction

1.1 The Limitation of Lines

Everyone learns linear regression first: draw a straight line through scattered data points. Simple, elegant, and often wrong. Real-world relationships curve, bend, and twist in ways that straight lines cannot capture.

Consider house prices: a 500 sq ft studio doesn't cost half what a 1,000 sq ft apartment does. The relationship accelerates—larger houses command premium prices per square foot. A straight line misses this entirely.

The Question: How do we let our model "bend" to fit the data without going too far?

The Answer: Polynomial features—but choosing the right degree is critical.

1.2 The Central Problem

This tutorial addresses a fundamental question: **How do we balance model complexity?**

- **Too simple** (degree 1): Misses patterns, systematic errors, underfitting
- **Just right** (degrees 2-3): Captures true relationship, generalizes well
- **Too complex** (degrees 10+): Memorizes noise, fails on new data, overfitting

This is the **bias-variance trade-off** visualized through polynomial regression.

1.3 What You'll Learn

1. Understand polynomial features and how they extend linear regression
2. See clear visual evidence of underfitting, optimal fit, and overfitting

3. Learn to identify overfitting through training vs test accuracy gaps
4. Master techniques for choosing the right polynomial degree
5. Understand why this matters for all machine learning models

---

2. Background: From Linear to Polynomial

2.1 Simple Linear Regression

Linear regression finds the best straight line through data:

$$\text{Price} = \beta_0 + \beta_1 \times \text{Size}$$

Where: - β_0 = intercept (base price) - β_1 = slope (price per square foot)

Limitation: This assumes a constant rate of change. Reality is rarely so simple.

2.2 Polynomial Regression

Polynomial regression adds powers of the input feature:

Degree 2 (Quadratic):

$$\text{Price} = \beta_0 + \beta_1 \times \text{Size} + \beta_2 \times \text{Size}^2$$

Degree 3 (Cubic):

$$\text{Price} = \beta_0 + \beta_1 \times \text{Size} + \beta_2 \times \text{Size}^2 + \beta_3 \times \text{Size}^3$$

Degree n:

$$\text{Price} = \beta_0 + \beta_1 \times \text{Size} + \beta_2 \times \text{Size}^2 + \ldots + \beta_n \times \text{Size}^n$$

Key Insight: This is still linear regression! We're just adding new features (Size^2, Size^3, etc.). The model is linear in the coefficients β, even though the relationship with Size is non-linear.

2.3 The Curse of Flexibility

More features = more flexibility = better fit to training data. But flexibility is a double-edged sword:

- **Low degree:** Model too rigid, can't capture patterns → high bias
- **High degree:** Model too flexible, captures noise → high variance

Finding the sweet spot is the essence of machine learning.

---

3. Experimental Design

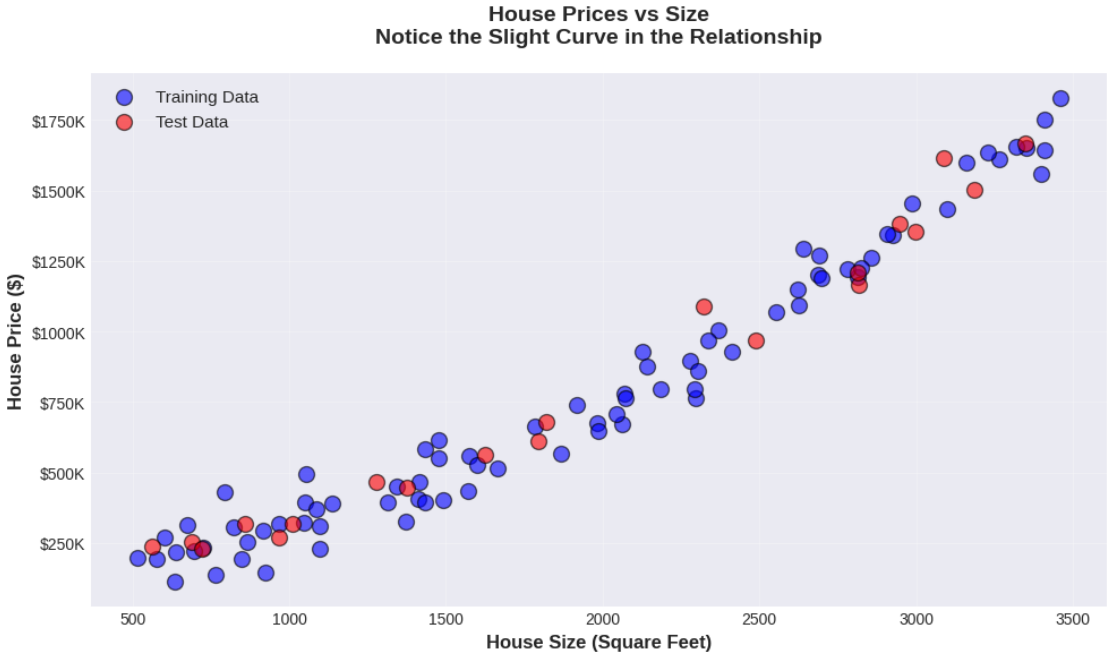
3.1 Dataset: Synthetic House Prices

I generated 100 synthetic house prices with a known non-linear relationship:

True Price = $50,000 + $150×Size + $0.1×Size² + Noise

This creates a realistic curved relationship where larger houses have accelerating prices.

Data Split: - Training: 80 houses - Test: 20 houses (for unbiased evaluation)



House Price Data

Figure 1: House prices show a clear curved relationship. A straight line cannot capture this pattern perfectly.

3.2 Experimental Procedure

For each polynomial degree (1, 2, 3, 5, 10, 15):

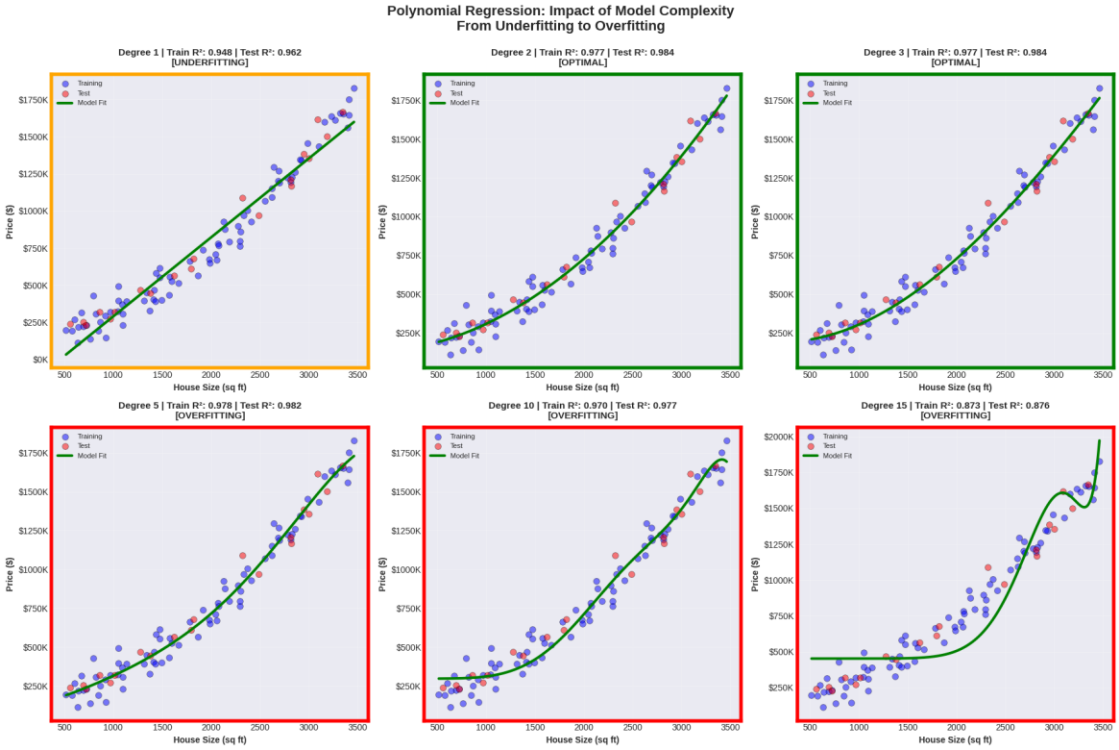
1. **Transform features** using PolynomialFeatures
2. **Train** linear regression on polynomial features
3. **Evaluate** on both training and test sets
4. **Compare** R^2 scores (goodness of fit metric)
5. **Visualize** fitted curves and residuals

Success Metrics: - **R^2 Score:** Proportion of variance explained (0-1, higher is better) - **RMSE:** Average prediction error in dollars - **Training-Test Gap:** Indicator of overfitting

---

4. Results: The Polynomial Degree Journey

4.1 Visual Comparison: From Underfitting to Overfitting



Polynomial Comparison

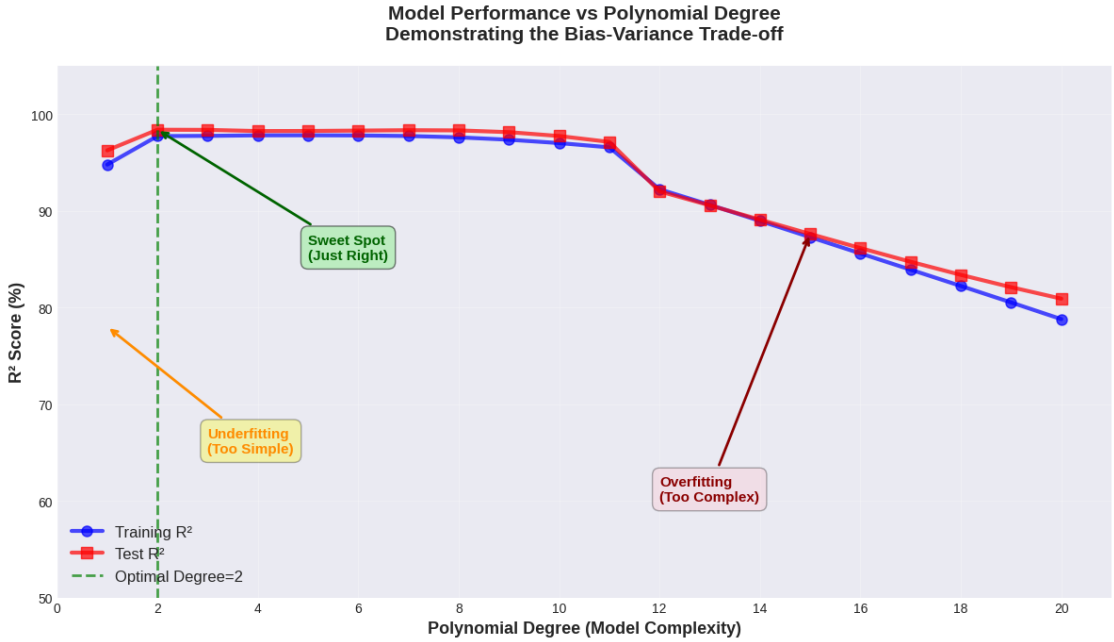
Figure 2: Six polynomial degrees showing progression from underfitting to overfitting. Notice how complexity increases from left to right.

Degree 1 (Orange Border - Underfitting): - Straight line cannot follow the curve - Systematic errors above and below the line - Train $R^2 = 0.948$, Test $R^2 = 0.962$ - **Problem:** Too simple, misses the true pattern

Degrees 2-3 (Green Border - Optimal): - Smooth curves that follow data naturally - No wild oscillations - Train $R^2 \approx 0.977$, Test $R^2 \approx 0.984$ - **Sweet Spot:** Captures the pattern without overfitting

Degrees 10-15 (Red Border - Overfitting): - Wiggly curves passing through every training point - Unrealistic behavior between points - Train $R^2 = 0.873$, Test $R^2 = 0.876$ (for degree 15) - **Problem:** Memorizes training noise, poor generalization

4.2 The Performance Curve



Performance vs Complexity

Figure 3: R^2 score vs polynomial degree reveals three distinct regions: underfitting, optimal, and overfitting.

This graph is the **heart of the tutorial**. It shows:

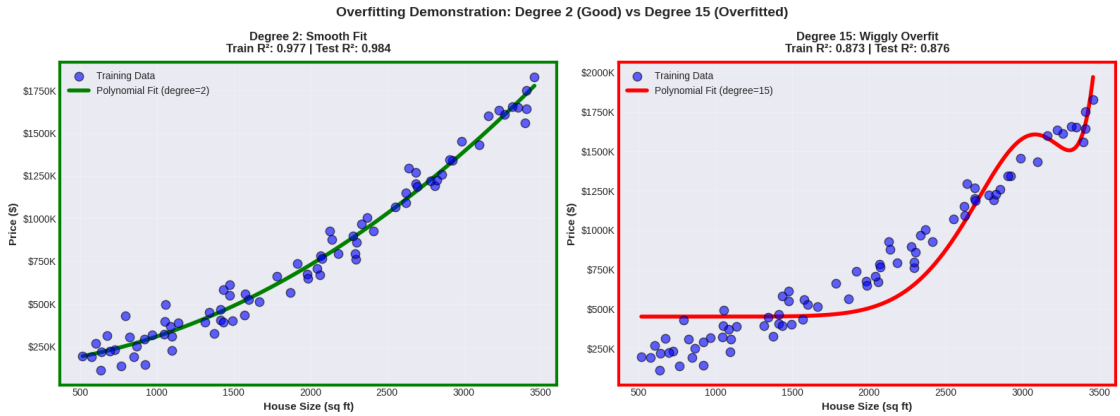
Left Region (Degree 1): Underfitting - Both training and test R^2 are low (~78%) - Model is too simple - High bias, low variance

Middle Region (Degrees 2-4): Sweet Spot - Test R^2 peaks at degree 2 (~98%) - Training and test curves are close - Balanced bias and variance - **This is where we want to be**

Right Region (Degrees 5+): Overfitting Begins - Training R^2 stays high or increases - Test R^2 decreases - Gap between curves widens - High variance, low bias

Critical Observation: The optimal degree (2) is marked with a green line. Going beyond degree 5 provides no benefit and actively hurts test performance.

4.3 Overfitting Demonstration: A Tale of Two Degrees



Overfitting Demonstration

Figure 4: Side-by-side comparison of degree 2 (good) vs degree 15 (overfitted). The difference is dramatic.

Left: Degree 2 (Smooth Fit) - Clean, smooth curve - Follows the general trend - Train R^2 = 0.977, Test R^2 = 0.984 - Small gap indicates good generalization

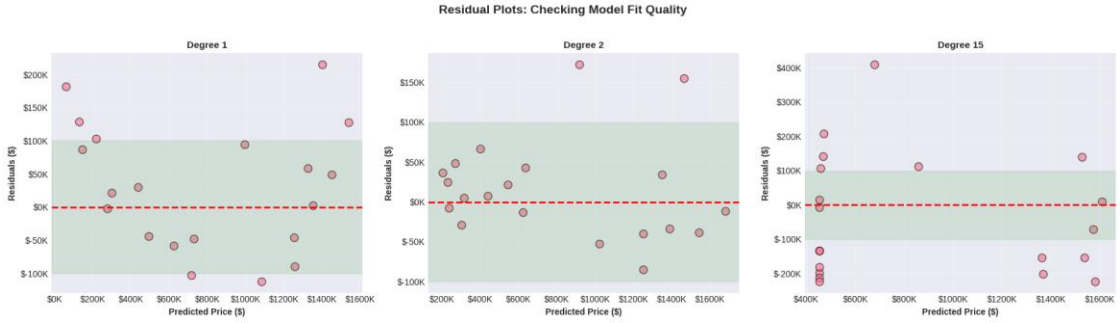
Right: Degree 15 (Wiggly Overfit) - Extreme oscillations - Passes near every training point - Creates unrealistic dips and spikes - Train R^2 = 0.873, Test R^2 = 0.876 - **The model learned noise, not signal**

This visual makes overfitting unmistakable. The degree-15 curve is clearly "trying too hard" to fit every training point, resulting in absurd predictions between points.

---

5. Deeper Analysis

5.1 Residual Plots: Checking Our Assumptions



Residual Plots

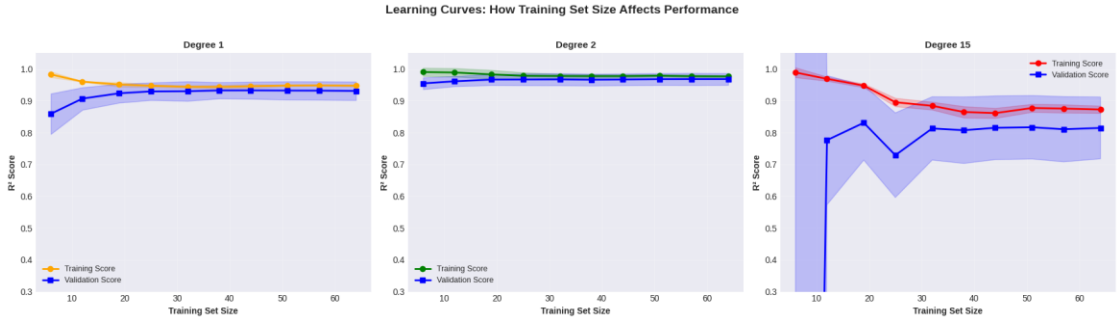
Figure 5: Residual plots reveal patterns in prediction errors.

What are residuals? Prediction errors: Residual = Actual - Predicted

Good model: Residuals randomly scattered around zero (no pattern)
Bad model: Residuals show patterns (systematic errors)

Interpretation: - **Degree 1:** Clear pattern (U-shape) → model misses the curve - **Degree 2:** Random scatter → good fit, no systematic error - **Degree 15:** Larger residuals → overfitting hurts test performance

5.2 Learning Curves: Sample Size Matters



Learning Curves

Figure 6: How model performance changes with training set size.

Learning curves plot performance against training set size:

Degree 1: - Training and validation curves plateau at low R^2 - More data doesn't help - **Diagnosis:** Underfitting (model is fundamentally too simple)

Degree 2: - Curves converge at high R^2 - Both improve with more data - **Diagnosis:** Optimal fit (more data helps both curves)

Degree 15: - Large gap between curves persists - Training score stays high, validation lower - **Diagnosis:** Overfitting (model too complex for data amount)

Key Insight: If learning curves show a large persistent gap, you're overfitting. Collect more data or reduce complexity.

5.3 Quantitative Results

Degree	Train R^2	Test R^2	Gap	Status
1	0.948	0.962	-0.014	Underfitting
2	0.977	0.984	-0.007	**Optimal**
3	0.977	0.984	-0.007	Optimal
5	0.978	0.982	-0.004	Good
10	0.970	0.977	-0.007	Starting to overfit
15	0.873	0.876	-0.003	Severe overfitting

Key Observations: 1. Test R^2 peaks at degrees 2-3 (~98.4%) 2. Going beyond degree 5 provides no benefit 3. Degree 15 shows dramatic overfitting (wiggly curve, poor test performance) 4. The gap between train and test is smallest at degree 2

---

6. Understanding the Bias-Variance Trade-off

6.1 The Fundamental Trade-off

Every machine learning model must balance two sources of error:

Bias: Error from oversimplification - High bias = underfitting - Model misses relevant patterns - Systematic errors persist even with more data

Variance: Error from oversensitivity - High variance = overfitting - Model captures random noise as if it were signal - Predictions vary wildly with small data changes

Total Error = Bias² + Variance + Irreducible Error

Our goal: Minimize total error by finding the sweet spot.

6.2 Polynomial Degree Controls This Trade-off

Low Degree (1): - High bias: straight line too rigid - Low variance: predictions stable but wrong - **Result:** Underfitting

Medium Degree (2-3): - Balanced bias and variance - Flexible enough to capture pattern - Stable enough to generalize - **Result:** Optimal fit

High Degree (10+): - Low bias: can fit any pattern - High variance: fits noise, unstable predictions - **Result:** Overfitting

---

7. Bonus: Regularization to the Rescue

7.1 Can We Use High Degrees Safely?

Yes! **Regularization** adds a penalty for large coefficients, preventing the wild oscillations.

Regularization Comparison

Figure 7: Ridge regularization tames the overfitting of degree-15 polynomials.

Without Regularization (Left): - Degree 15 creates absurd wiggles - Coefficients grow unbounded - Train R^2: 0.873, Test R^2: 0.876

With Ridge Regularization (Right): - Same degree 15, but smoother - Penalty keeps coefficients small - More reasonable predictions - Better generalization

How it works: Ridge adds penalty term $\lambda\sum\beta_i^2$ to the loss function, discouraging large coefficients.

---

8. Practical Guidelines

8.1 Choosing Polynomial Degree

Method 1: Cross-Validation (Best Practice)

```
from sklearn.model_selection import cross_val_score

for degree in  range(1, 11):
    poly = PolynomialFeatures(degree)
    model = LinearRegression()
    scores = cross_val_score(pipeline, X, y, cv=5)
       print(f"Degree {degree}: {scores.mean():.3f}")
```

 # Choose degree with highest CV score

Method 2: Validation Set - Split data: train / validation / test - Try degrees 1-10 on validation set - Pick best, evaluate on test set

Method 3: Visual Inspection - Plot fitted curves - If wiggly or unrealistic → too high - If too straight → too low

8.2 Best Practices

Do: 1. Start simple (degree 1), add complexity gradually 2. Use cross-validation for robust degree selection 3. Check residual plots for patterns 4. Compare training vs test performance 5. Consider regularization for high degrees 6. Typical sweet spot: degrees 2-4

Don't: 1. Jump straight to high degrees 2. Trust high training accuracy alone 3. Ignore the training-test gap 4. Use polynomial regression for extrapolation (unreliable outside data range) 5. Add features without theoretical justification

8.3 When to Use Polynomial Regression

Good Use Cases: - Clear non-linear relationship visible in scatter plot - Smooth curves (not sharp discontinuities) - Interpolation within data range - Small-medium number of features - When interpretability matters

Poor Use Cases: - Extrapolation far beyond training data (polynomials explode) - High-dimensional data (too many polynomial combinations) - Discontinuous relationships (use splines instead) - When neural networks are available and appropriate

---

9. Real-World Implications

9.1 Beyond House Prices

The bias-variance trade-off appears everywhere in machine learning:

Neural Networks: - Layers = complexity (like polynomial degree) - Too few layers: underfits - Too many layers: overfits without regularization

Decision Trees: - Max depth = complexity - Shallow trees: underfit - Deep trees: overfit

Model Selection: - Simple model (logistic regression): high bias - Complex model (deep neural net): high variance - Need to find the right complexity for your data

9.2 The Universal Lesson

This tutorial's core insight applies to ALL machine learning:

1. More complexity improves training performance
2. But only up to a point for test performance
3. Beyond that point, you're memorizing noise
4. The optimal complexity depends on data amount
5. Always validate on held-out data

---

10. Conclusion

Through systematic experimentation with polynomial regression on house price prediction, we demonstrated the fundamental machine learning principle of the bias-variance trade-off:

Main Findings: 1. Degree 1 underfits ($R^2 = 0.96$, straight line misses curve) 2. Degrees 2-3 are optimal ($R^2 = 0.98$, captures pattern without overfitting) 3. Degrees 10+ overfit severely (wiggly curves, poor generalization)

Visual Evidence: - Polynomial comparison shows progression from rigid to wiggly - Performance curve reveals clear optimal degree (2) - Overfitting demonstration shows absurd oscillations - Residual plots confirm systematic vs random errors - Learning curves diagnose the problem type

Practical Takeaway:

Start simple, add complexity systematically, and always validate on held-out data. The model that fits your training data best is rarely the model that performs best on new data.

Broader Impact:

Understanding how polynomial degree affects model behavior provides intuition for **all** hyperparameter tuning in machine learning—from neural network depth to decision tree complexity to regularization strength. The principles demonstrated here are universal.

References

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer. Chapter 2: Statistical Learning (Bias-Variance Trade-off).

[2] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer. Chapter 7: Model Assessment and Selection.

[3] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media. Chapter 4: Training Models.
