

Digital Finance 3: Technology in Finance

Lesson 27: Supervised Learning - Regression

FHGR

December 11, 2025

By the end of this lesson, you will be able to:

- Explain the supervised learning paradigm (features, labels, training)
- Understand simple and multiple linear regression
- Interpret regression coefficients in financial contexts
- Evaluate model performance using R-squared and related metrics
- Recognize overfitting and apply regularization techniques
- Identify finance applications of regression models

Core Idea:

- Learn from labeled examples
- **Training data:** $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$
- X = features (inputs, predictors)
- Y = label (output, target)
- Goal: Learn function $f : X \rightarrow Y$

Two Types:

- ① **Regression:** Predict continuous Y (today's lesson)
- ② **Classification:** Predict discrete Y (next lesson)

Finance Example (Regression):

- Features X : Company financials (P/E, ROE, Size)
- Label Y : Next-month stock return
- Training: Historical data (2000-2020)
- Test: Predict 2021 returns

Key Steps:

- ① Collect labeled data
- ② Split: Train (70%), Validation (15%), Test (15%)
- ③ Train model on training set
- ④ Tune on validation set
- ⑤ Evaluate on test set (never seen before)

Golden Rule: Never use test data until final evaluation (avoid overfitting).

Simple Linear Regression: The Basics

Model:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- Y : Dependent variable (target)
- X : Independent variable (feature)
- β_0 : Intercept (value when $X = 0$)
- β_1 : Slope (change in Y per unit X)
- ϵ : Error term (residual)

Goal: Find β_0, β_1 that minimize errors.

Method: Ordinary Least Squares (OLS)

$$\min_{\beta_0, \beta_1} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

where $\hat{Y}_i = \beta_0 + \beta_1 X_i$

Finance Example:

Predict stock return (Y) from P/E ratio (X).

Suppose OLS gives:

$$\text{Return} = 0.05 - 0.002 \times \text{P/E}$$

Interpretation:

- Intercept (0.05): Expected 5% return for P/E = 0 (extrapolation, not meaningful)
- Slope (-0.002): Each 1-point increase in P/E decreases expected return by 0.2%
- Negative relationship: Higher P/E (expensive) → lower return (value effect)

Limitations:

- Assumes linear relationship
- Single predictor (oversimplified)

Model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

- Multiple features: X_1, X_2, \dots, X_p
- Each β_j : Partial effect (holding others constant)
- OLS still minimizes squared errors

Matrix Form:

$$\begin{aligned} Y &= X\beta + \epsilon \\ \hat{\beta} &= (X^T X)^{-1} X^T Y \end{aligned}$$

Finance Example:

Predict stock return from:

- X_1 : P/E ratio
- X_2 : Debt/Equity
- X_3 : Market cap (log)
- X_4 : Past 12-month return (momentum)

Estimated model:

$$\begin{aligned} \text{Return} &= 0.03 - 0.001 \times \text{P/E} \\ &\quad - 0.015 \times \text{D/E} \\ &\quad + 0.002 \times \log(\text{Size}) \\ &\quad + 0.12 \times \text{Mom} \end{aligned}$$

Interpretation:

- Momentum (0.12): Strongest predictor
- Debt (-0.015): Financial risk reduces returns
- Size (+0.002): Weak positive effect

Five Key Assumptions:

- ① **Linearity:** Relationship is linear
- ② **Independence:** Observations are independent
- ③ **Homoscedasticity:** Constant error variance
- ④ **Normality:** Errors normally distributed
- ⑤ **No multicollinearity:** Features not perfectly correlated

Diagnostics:

- Residual plots (linearity, homoscedasticity)
- QQ plots (normality)
- Variance Inflation Factor (VIF) for multicollinearity

Bottom Line: Regression is robust, but severe violations reduce reliability.

Violations in Finance:

- **Non-linearity:** Returns vs. ratios often non-linear
- **Heteroscedasticity:** Volatility clustering (GARCH effects)
- **Autocorrelation:** Time series dependence
- **Multicollinearity:** Related accounting ratios

Remedies:

- Transformations (log, square root)
- Robust standard errors (White, Newey-West)
- Feature selection (remove correlated vars)
- Non-linear models (polynomial, GAM)

R-squared (R^2):

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum(Y_i - \hat{Y}_i)^2}{\sum(Y_i - \bar{Y})^2}$$

- Proportion of variance explained
- Range: $[0, 1]$ (higher is better)
- $R^2 = 0$: Model no better than mean
- $R^2 = 1$: Perfect fit (suspicious!)

Interpretation:

- $R^2 = 0.25$: Model explains 25% of variance
- Remaining 75%: Unexplained (noise, other factors)

Adjusted R-squared:

$$R_{adj}^2 = 1 - (1 - R^2) \frac{n - 1}{n - p - 1}$$

- Penalizes adding features
- Use when comparing models with different p

Typical R^2 in Finance:

- Stock return prediction: 0.02-0.10 (very noisy)
- Bond yield modeling: 0.70-0.95 (more predictable)
- Credit spreads: 0.40-0.60

Warning:

- High R^2 doesn't mean good out-of-sample performance
- Can overfit to training data

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$$

- Average absolute prediction error
- Same units as Y (interpretable)
- Robust to outliers

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}$$

- Penalizes large errors more (squared)
- Same units as Y
- Most common in ML

Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right|$$

- Percentage error (scale-free)
- Problematic if Y_i near zero

Which to Use?

- RMSE: Standard choice (differentiable, penalizes outliers)
- MAE: If outliers less important
- MAPE: Comparing models across different scales
- R^2 : Variance explanation (interpretability)

Key: Always evaluate on held-out test set.

What is Overfitting?

- Model learns training data too well
- Captures noise, not signal
- Poor generalization to new data

Symptoms:

- High training R^2 (0.95), low test R^2 (0.20)
- Complex model (many features)
- Unstable coefficients

Causes:

- Too many features relative to observations ($p \approx n$)
- Features without predictive power
- Overly flexible models
- Lack of regularization

Bias-Variance Tradeoff:

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- **Bias:** Error from wrong assumptions (underfitting)
- **Variance:** Error from sensitivity to training data (overfitting)
- Simple models: High bias, low variance
- Complex models: Low bias, high variance

Goal: Find sweet spot (minimize total error).

Detection:

- Plot training vs. validation error
- Cross-validation
- Out-of-sample testing

Idea: Penalize large coefficients to reduce overfitting.

Ridge Regression (L2):

$$\min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

- Penalty: Sum of squared coefficients
- Shrinks coefficients toward zero
- All features retained (no feature selection)
- λ : Regularization strength (tune via CV)

Effect:

- Reduces variance (less overfitting)
- Increases bias (slightly)
- Handles multicollinearity well

Lasso Regression (L1):

$$\min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

- Penalty: Sum of absolute coefficients
- Sets some β_j exactly to zero (feature selection)
- Sparse solutions (interpretable)

Elastic Net:

$$\text{Penalty} = \lambda_1 \sum |\beta_j| + \lambda_2 \sum \beta_j^2$$

- Combines L1 and L2
- Best of both worlds

Choosing λ :

- Cross-validation (grid search)
- Higher $\lambda \rightarrow$ more regularization

Why Cross-Validation?

- Estimate out-of-sample performance
- Tune hyperparameters (e.g., λ)
- Maximize use of limited data

K-Fold Cross-Validation:

- ① Split data into K folds (typically 5 or 10)
- ② For each fold k :
 - Train on $K - 1$ folds
 - Validate on fold k
- ③ Average performance across folds

Advantages:

- All data used for training and validation
- Reduces variance of performance estimate

Leave-One-Out CV (LOOCV):

- $K = n$ (extreme case)
- Train on $n - 1$, test on 1
- Computationally expensive
- Low bias, high variance

Time Series CV:

- Cannot randomly shuffle (temporal order)
- Use expanding or rolling windows
- Example: Train on 2000-2010, test on 2011; train on 2000-2011, test on 2012; etc.

Best Practice:

- Use CV for model selection
- Reserve separate test set for final evaluation
- Never use test set for tuning

Transformations:

- **Log:** $\log(Y)$ or $\log(X)$ (skewed distributions)
- **Polynomial:** X, X^2, X^3 (capture non-linearity)
- **Interactions:** $X_1 \times X_2$ (joint effects)
- **Binning:** Convert continuous to categorical

Finance-Specific:

- Ratios: P/E, P/B, ROE, Debt/Equity
- Momentum: Past returns (1-month, 12-month)
- Volatility: Rolling standard deviation
- Technical indicators: MA, RSI, MACD

Normalization:

- **Standardization:** $(X - \mu)/\sigma$ (mean 0, std 1)
- **Min-Max:** $(X - X_{min})/(X_{max} - X_{min})$ (range [0,1])
- Important for regularization (features on same scale)

Lag Variables (Time Series):

- Y_{t-1}, Y_{t-2}, \dots (autoregressive)
- Moving averages
- Seasonal indicators

Avoid:

- Leakage (using future info)
- Perfectly correlated features
- Too many features (curse of dimensionality)

Problem Setup:

- Target: Next-month stock return
- Features: Fundamentals, technical, macro
- Data: Monthly, 1990-2020
- Universe: S&P 500 stocks

Feature Categories:

- ① **Value:** P/E, P/B, dividend yield
- ② **Momentum:** Past 12-month return
- ③ **Quality:** ROE, profit margin, accruals
- ④ **Size:** Market cap (log)
- ⑤ **Volatility:** 60-day std dev

Model Comparison:

- OLS: $R^2 = 0.04$ (test)
- Ridge ($\lambda = 10$): $R^2 = 0.06$
- Lasso ($\lambda = 0.1$): $R^2 = 0.07$ (selected 12/30 features)

Key Findings:

- Momentum strongest predictor ($\beta = 0.15$)
- P/B negative ($\beta = -0.03$) - value effect
- Low overall R^2 (markets are noisy)
- Lasso improves via feature selection

Reality Check:

- Transaction costs erode small edges
- Out-of-sample performance lower
- Regime shifts (models break in crises)

Problem:

- Predict bond yields at various maturities
- Features: Macro variables, term structure factors
- More predictable than stocks

Nelson-Siegel Model (Parametric):

$$Y(m) = \beta_1 + \beta_2 e^{-m/\tau} + \beta_3 \frac{m}{\tau} e^{-m/\tau}$$

- β_1 : Long-term level
- β_2 : Short-term component
- β_3 : Curvature
- m : Maturity, τ : Decay parameter

ML Approach (Non-Parametric):

- Features: GDP growth, inflation, Fed Funds rate, VIX, yield spreads
- Target: 10-year Treasury yield
- Ridge regression: $R^2 = 0.82$ (test)

Key Predictors:

- Fed Funds rate ($\beta = 0.65$)
- Inflation expectations ($\beta = 0.42$)
- 2-year yield ($\beta = 0.58$)

Use Cases:

- Portfolio allocation
- Hedging interest rate risk
- Trading strategies (carry, curve)

Problem:

- Predict property sale price
- Features: Size, location, age, amenities
- Traditional: Hedonic pricing models

Features:

- Square footage
- Number of bedrooms, bathrooms
- Lot size
- Age of property
- Zip code (location proxy)
- School quality, crime rates
- Distance to CBD, transit

Model:

$$\log(\text{Price}) = \beta_0 + \beta_1 \log(\text{SqFt}) + \beta_2 \text{Beds} + \dots$$

- Log-log form: Elasticities
- Ridge to handle multicollinearity (correlated location vars)

Results:

- $R^2 = 0.75$ (typical)
- Square footage: $\beta = 0.6$ (10% increase in size → 6% price increase)
- Extra bedroom: +\$20k

Applications:

- Automated valuation models (AVMs) - Zillow Zestimate
- Mortgage underwriting
- Investment property analysis

Limitations of Linear Regression

When Regression Fails:

- Non-linear relationships (interactions, thresholds)
- High-dimensional data ($p \gg n$)
- Complex feature interactions
- Heavy-tailed distributions (outliers)
- Non-stationary data (regime changes)

Example:

Stock returns vs. market cap:

- Small-cap premium (non-linear)
- January effect (seasonality)
- Crisis periods (structural breaks)

Linear regression assumes constant relationship, which breaks.

Alternatives:

- **Polynomial regression:** Add X^2, X^3 terms
- **Generalized Additive Models (GAM):** Smooth non-linear functions
- **Tree-based methods:** Random Forests, Gradient Boosting (later lessons)
- **Neural networks:** Deep learning (later lessons)

Trade-offs:

- Linear: Simple, interpretable, fast
- Non-linear: Flexible, accurate, complex

Best Practice:

- Start with linear (baseline)
- If poor fit, try non-linear
- Balance accuracy vs. interpretability

Data Preparation:

- Winsorize outliers (1st-99th percentile)
- Check for multicollinearity ($VIF \geq 10$)
- Normalize features (especially for regularization)
- Handle missing data carefully

Model Selection:

- Start simple (OLS)
- Add regularization (Ridge/Lasso)
- Use cross-validation for λ
- Check residual diagnostics

Avoid Common Mistakes:

- Look-ahead bias (using future data)
- Overfitting (too many features)
- Ignoring transaction costs
- Extrapolation beyond data range

Interpretation:

- Report coefficients with confidence intervals
- Economic significance & statistical significance
- Explain magnitude in practical terms

Validation:

- Out-of-sample testing (time series: walk-forward)
- Robustness checks (different time periods, subsamples)
- Benchmark against simple models (mean, random walk)

Communication:

- Visualize predictions vs. actuals
- Report multiple metrics (R^2 , RMSE, MAE)
- Acknowledge limitations

Core Concepts:

- Supervised learning: Learn from labeled data
- Linear regression: $Y = X\beta + \epsilon$
- OLS minimizes squared errors
- Multiple regression: Multiple predictors

Evaluation:

- R^2 : Variance explained
- RMSE, MAE: Prediction error
- Always test out-of-sample

Overfitting:

- Central problem in ML
- Regularization (Ridge, Lasso) helps
- Cross-validation for tuning

Finance Applications:

- Stock returns (low R^2 , noisy)
- Bond yields (higher R^2 , predictable)
- Real estate (moderate R^2)
- Limitations: Non-linearity, regime changes

Lesson 28: Supervised Learning - Classification

Topics to be covered:

- Logistic regression (binary classification)
- Decision boundaries and probabilities
- Confusion matrix (TP, FP, TN, FN)
- Accuracy, precision, recall, F1-score
- ROC curves and AUC
- Applications: Credit default, fraud detection

Preparation:

- Review probability basics (odds, log-odds)
- Think: What financial problems involve yes/no predictions?