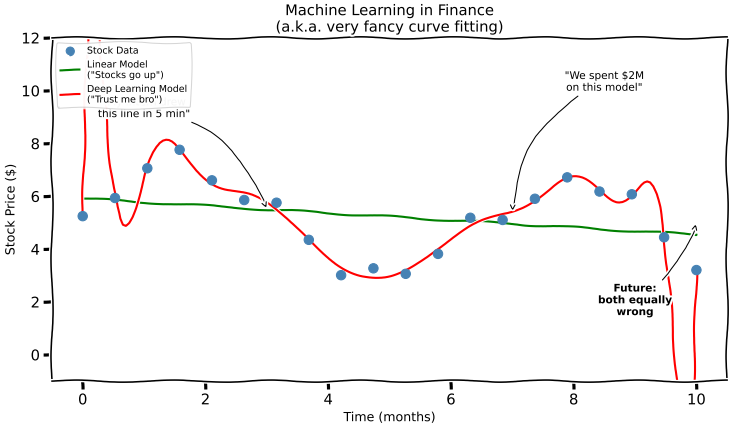
Lesson 5.1: Machine Learning Foundations for Finance

Module 5: Automation and Intelligence

Digital Finance Course

June 2, 2026

ML in Finance: A Candid Look



Every financial institution uses Machine Learning (ML)—but most practitioners lack intuition for when it works, when it fails, and why.

Learning Objectives

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

1. Distinguish classification from regression and identify appropriate financial use cases for each [Understand]
2. Explain the bias-variance tradeoff and why overfitting is the primary risk in financial ML [Understand]
3. Compare tree-based methods (decision tree, random forest, gradient boosting) for financial prediction [Analyze]
4. Interpret a confusion matrix and calculate precision, recall, and F1-score [Apply]
5. Design a feature engineering pipeline for financial data [Create]
6. Evaluate when anomaly detection (Isolation Forest) is more appropriate than supervised classification [Evaluate]

Bloom's levels: Understand (1,2), Apply (4), Analyze (3), Evaluate (6), Create (5). Core ML toolkit used in production finance today.

Bridge: From Financial Systems to Intelligent Automation

Where we have been (teaching order):

- M1 Cost and M2 Access: why payments are expensive and who gets served
- M6 Infrastructure: the rails that clear and settle money
- M3 Trust: cryptography, consensus, smart contracts, DeFi
- M4 Risk: measuring and controlling potential losses

Where we are going:

- M5 Automation: **can machines run these systems?**
- This lesson: the ML foundations every finance professional needs
- Not "how to code ML" but "how to *think* about ML"

Teaching sequence: M4 Risk → M5 Automation (here) → M7 Compliance next.
We built and measured financial systems; now: can machines run them?

M5 synthesizes everything before it: ML is the engine that automates the financial systems we have studied; M7 Compliance follows.

What is Supervised Learning?

Definition:

- Learning from labeled examples (input $\rightarrow$ known output)
- The model finds patterns in historical data to predict future outcomes
- "Supervised" because we provide correct answers during training

Two fundamental tasks:

1. **Classification:** predict a category (yes/no, A/B/C)
2. **Regression:** predict a continuous number

Financial Classification Examples:

- Fraud detection: transaction is fraud or legitimate
- Credit scoring: approve or deny a loan
- Churn prediction: will this customer leave?

Financial Regression Examples:

- Property valuation: predict house price
- Revenue forecasting: predict next quarter's revenue
- Loss Given Default (LGD): predict recovery amount

Key insight: Classification answers "what kind?" while regression answers "how much?" The choice determines your entire modeling pipeline.

What is Overfitting? — The Problem

Definition:

- Model memorizes training data instead of learning general patterns
- Performs brilliantly on historical data but fails on new data
- **The single biggest risk in financial ML**

Why finance is especially vulnerable:

- Markets are non-stationary (patterns change over time)
- Data is noisy (random fluctuations look like patterns)
- Small sample sizes for rare events (fraud, defaults, crises)
- Survivorship bias in historical datasets

Wall Street proverb: "Past performance is no guarantee of future results." Overfitting is the mathematical version of this warning.

Overfitting — Detection and Remedies

Signs of overfitting:

- Training accuracy $\gg$ test accuracy
- Model complexity far exceeds data complexity
- Performance degrades on new time periods

Remedies:

- Cross-validation (see later slide)
- Regularization (penalize complexity)
- Simpler models first (Occam's razor)
- More data (if available)
- Feature selection (fewer inputs)

Detecting overfitting early saves months of wasted effort. Always compare training vs. test performance—if the gap is large, you are overfitting.

What is the Bias-Variance Tradeoff?

Definition:

- Every model's error decomposes into three parts:

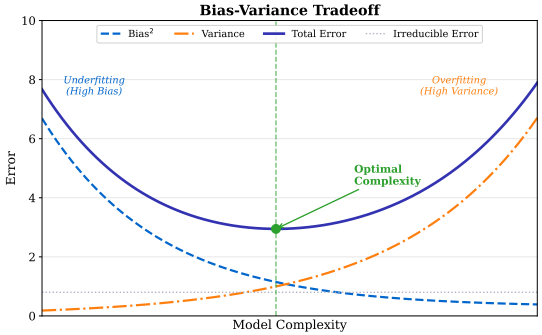
$$\mathbb{E}\left[(y - \hat{f}(x))^2\right] = \underbrace{\left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2}_{\text{Bias}^2} + \underbrace{\text{Var}(\hat{f}(x))}_{\text{Variance}} + \underbrace{\sigma_\varepsilon^2}_{\text{Irreducible}}$$

Bias = error from wrong assumptions

- Linear model on nonlinear data
- Underfitting: model is too simple

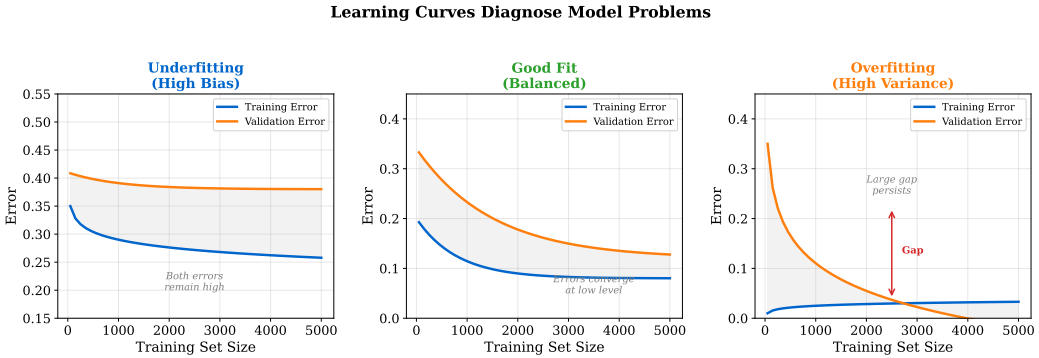
Variance = error from sensitivity to training data

- Different training samples $\Rightarrow$ very different models
- Overfitting: model is too complex



The sweet spot: enough complexity to capture real patterns but not so much that you model noise. This tradeoff governs every ML decision.

Learning Curves Diagnose Model Problems



- **What you see:** Train error (blue) starts low and rises; validation error (orange) starts high and falls as data grows
- **Key pattern:** Converging curves at low error indicate good fit; persistent gap signals overfitting
- **Takeaway:** More data helps overfitting (gap narrows) but not underfitting (both curves stay high)

Practical diagnostic: plot train vs. validation error as data grows. If both are high → underfitting. If gap persists → overfitting. If converging at low level → good fit.

What is Cross-Validation?

Definition:

- Technique to estimate how well a model generalizes to unseen data
- Splits data into k folds; trains on $k-1$, tests on the held-out fold
- Repeats k times, averages performance

k-Fold Cross-Validation (typical: $k = 5$ or $k = 10$):

1. Divide data into k equal parts
2. For each fold i: train on all data except fold i, test on fold i
3. Report mean and standard deviation of scores

Why it matters in finance:

- A single train/test split can be misleading
- Reduces variance of performance estimate
- Detects overfitting: training score $\gg$ CV score

Time-series caution:

- Standard k-fold shuffles data randomly
- Financial data has temporal order
- **Use time-series split**: train on past, test on future
- Never train on tomorrow to predict yesterday

Critical rule: in finance, always use time-aware validation. Standard k-fold leaks future information into training ("look-ahead bias").

What is a Decision Tree? — Structure and Strengths

Definition:

- A flowchart-like model that makes decisions by splitting data on feature thresholds
- Each internal node = a question ("Is income $>$ \$50,000?")
- Each leaf node = a prediction (approve/deny)

Strengths:

- Highly interpretable ("show me why you denied this loan")
- Handles mixed data types (numeric + categorical)
- No feature scaling required
- Captures nonlinear relationships

Decision trees are the building block for all ensemble methods. Their simplicity makes them the default choice when explainability is a regulatory requirement.

Decision Trees — Weaknesses and Applications

Weaknesses:

- **Prone to overfitting** (deep trees memorize noise)
- Unstable: small data change $\Rightarrow$ very different tree
- Greedy splitting (locally optimal, not globally)

Key hyperparameters:

- `max_depth`: limits tree depth
- `min_samples_leaf`: minimum data per leaf

Financial use cases:

- Credit decisioning (regulatory requirement for explainability)
- Insurance underwriting rules
- Customer segmentation

Overfitting is the primary weakness of decision trees. Ensemble methods (Random Forest, XGBoost) solve this by combining many trees.

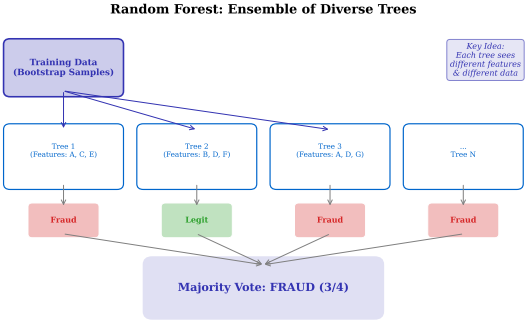
What is a Random Forest?

Definition:

- Ensemble of many decision trees trained on different random subsets
- Each tree sees a bootstrap sample (random rows) and random features
- Final prediction = majority vote (classification) or average (regression)

Why it works ("wisdom of crowds"):

- Individual trees overfit in different ways
- Averaging cancels out individual errors
- Reduces variance without increasing bias



- **What you see:** Multiple diverse trees vote on the same input to produce a robust consensus
- **Key pattern:** Individual trees disagree, but majority vote is more accurate than any single tree
- **Takeaway:** Ensemble diversity reduces overfitting—100 weak learners beat one strong learner

Random forests are the "Swiss army knife" of ML: robust, accurate, and require minimal tuning. Default choice for most tabular finance problems.

What is Gradient Boosting? — The Algorithm

Definition:

- Builds trees *sequentially*, each correcting the errors of the previous one
- Unlike random forest (parallel), boosting trees are *additive*
- Each new tree focuses on the hardest-to-predict examples

Key idea:

1. Fit a weak tree to the data
2. Compute residuals (errors)
3. Fit next tree to the residuals
4. Repeat; final prediction = sum of all trees

Gradient boosting corrects errors iteratively—each tree learns from the mistakes of the previous one.

XGBoost and Boosting vs. Random Forest

What is XGBoost?

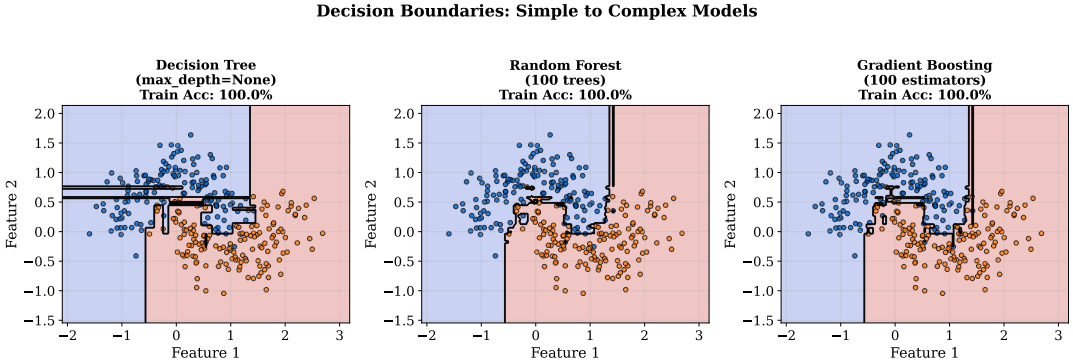
- eXtreme Gradient Boosting
- Optimized implementation with regularization
- Handles missing values natively
- Dominates Kaggle competitions and production finance models

Gradient Boosting vs. Random Forest:

- Boosting: often more accurate but sensitive to hyperparameters
- Random forest: more robust, harder to overfit
- Boosting: slower to train (sequential)
- Random forest: easily parallelizable

XGBoost is the most-used ML algorithm in financial services. It powers credit scoring, fraud detection, and risk models at most major banks.

Decision Boundaries: Simple to Complex Models



- **What you see:** Logistic regression creates a linear boundary; single tree creates jagged regions; ensembles smooth the boundary
- **Key pattern:** Single tree memorizes training noise (overfits); ensembles generalize better by averaging many boundaries
- **Takeaway:** Visual proof that ensemble methods reduce variance—smoother boundaries indicate better generalization

Visualizing the tradeoff: a single tree creates jagged boundaries (overfits). Random forest and gradient boosting smooth the boundary. More complex is not always better.

What is Feature Engineering? — Definition and Techniques

Definition:

- The process of creating, selecting, and transforming input variables to improve model performance
- Often the single most impactful step in ML
- "Garbage in, garbage out" applies forcefully

Common techniques for financial data:

- **Ratios:** debt-to-income, loan-to-value
- **Rolling windows:** 30-day average transaction amount
- **Lagged features:** last month's balance
- **Time since event:** days since last late payment
- **Aggregations:** max, min, std of transaction amounts

Industry saying: "Applied ML is 80% feature engineering and 20% modeling." The best algorithm cannot compensate for poor features.

Feature Engineering — Pipeline and Domain Expertise

Feature engineering pipeline:

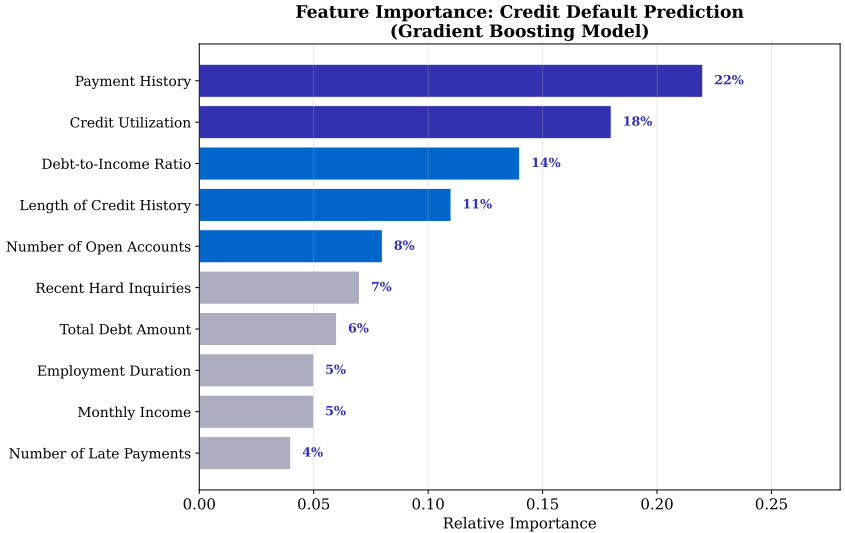
1. **Raw data** → clean missing values, outliers
2. **Domain features** → ratios, interactions
3. **Temporal features** → rolling stats, lags
4. **Encoding** → one-hot for categories
5. **Scaling** → normalize if needed (not for trees)
6. **Selection** → drop low-importance features

Why domain expertise matters:

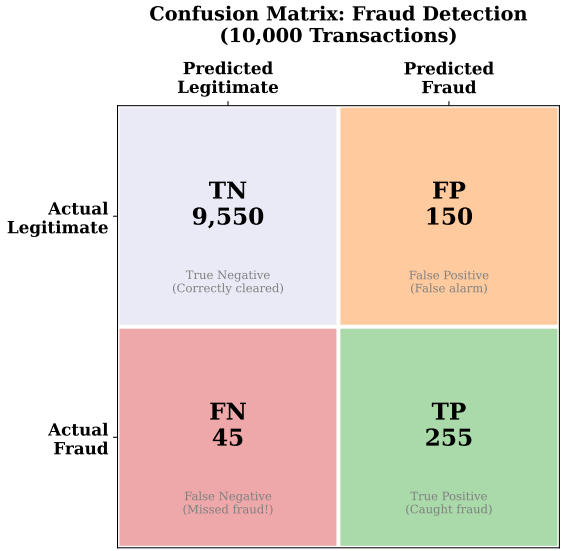
- A finance expert knows "payment velocity" matters
- An ML engineer might miss it
- Best models combine both skills

The best feature engineering combines domain knowledge (what matters in finance) with ML techniques (how to encode it). Neither alone is sufficient.

Feature Importance: What Drives the Model?



What is a Confusion Matrix?




```
Precision = 63.0%
Recall = 85.0%
F1-Score = 0.723
Accuracy = 98.0%
```

What are Precision, Recall, and F1-Score?

Precision ("Of those I flagged, how many were actually fraud?"):

$$\text{Precision} = \frac{TP}{TP + FP}$$

- High precision = few false alarms
- Critical when investigation is expensive

Example: If TP = 85 and FP = 15, Precision $= 85/(85 + 15) = 85\%$. Of 100 flagged transactions, 85 were actually fraud.

F1-Score (harmonic mean of precision and recall):

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Example:
$F_1 = 2 \times (0.85 \times 0.895)/(0.85 + 0.895) = 0.872$. This balances precision and recall into a single metric.

Recall ("Of all actual fraud, how many did I catch?"):

$$\text{Recall} = \frac{TP}{TP + FN}$$

- High recall = catch most fraud
- Critical when missing cases is dangerous

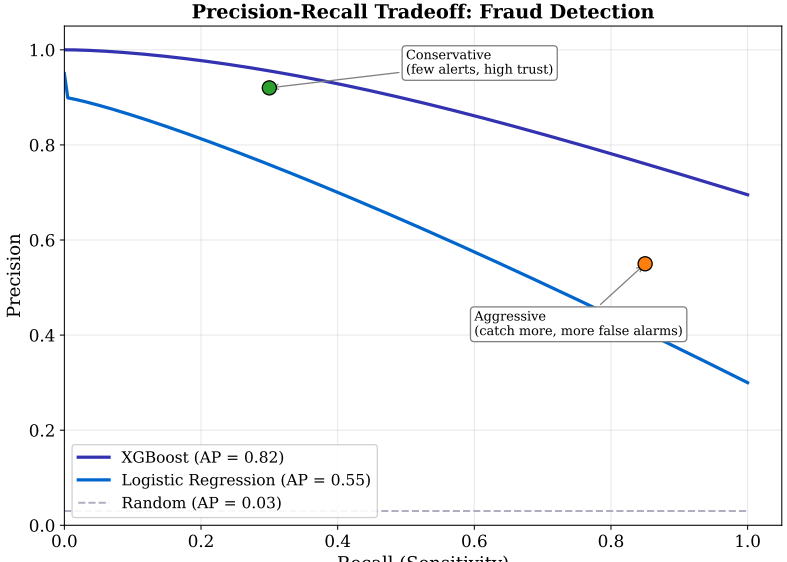
Example: If TP = 85 and FN = 10, Recall $= 85/(85 + 10) = 89.5\%$. The model caught 85 of 95 actual fraud cases.

Why not just use accuracy?

- With 99% legitimate transactions, a model predicting "always legitimate" gets 99% accuracy
- But catches 0% of fraud
- Accuracy is misleading for imbalanced data

In fraud detection, regulators care about recall ("did you catch the fraud?"). In trading signals, precision matters ("when you said buy, were you right?").

Precision-Recall Tradeoff



What is the ROC Curve and AUC?

ROC = Receiver Operating Characteristic:

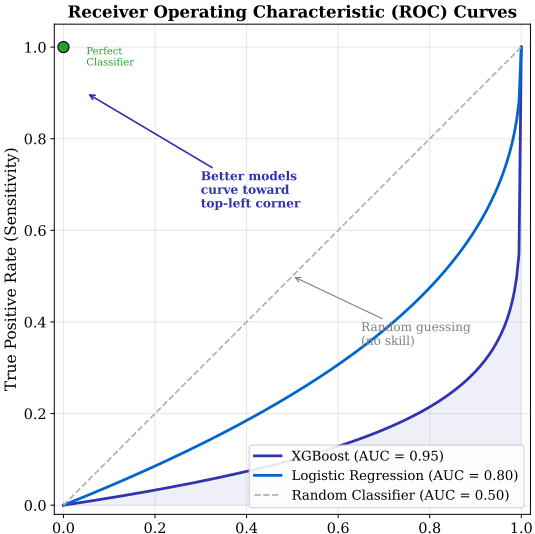
- Plots True Positive Rate vs. False Positive Rate at every threshold
- Originated in World War II radar signal detection

AUC = Area Under the ROC Curve:

- Single number summarizing classifier performance
- AUC = 0.5: random guessing (no skill)
- AUC = 1.0: perfect classifier
- AUC $\geq$ 0.8: generally considered good

Interpretation:

- AUC = probability that the model ranks a random positive higher than a random negative



When to Use Which Metric?

Business Problem	Primary Metric	Why	Example
Fraud detection	Recall	Missing fraud is costly	Catch $\geq 95\%$ of fraud
Trading signal	Precision	False signals lose money	"Buy" should mean buy
Credit scoring	AUC	Ranking quality matters	Rank applicants by risk
Spam filtering	F1-Score	Balance both errors	Don't miss real emails
Medical screening	Recall	Missing disease is fatal	Catch all positive cases
Customer churn	F1-Score	Both types of error matter	Balanced intervention

There is no universally "best" metric. The right metric depends on the business cost of each type of error. Always ask: "What is the cost of being wrong?"

What is Anomaly Detection? — Definition and When to Use It

Definition:

- Identifying data points that deviate significantly from the majority
- **Unsupervised**: does not require labeled examples of anomalies
- Critical when labeled fraud data is scarce or fraud patterns evolve

Supervised vs. unsupervised—when to use which:

- **Use supervised (classification)** when you have abundant labeled examples of both classes
- **Use anomaly detection** when:
  - Labels are scarce or unreliable
  - Fraud patterns change rapidly
  - You need to detect *unknown* fraud types

Anomaly detection does not need labeled fraud examples. It finds "unusual" transactions by learning what "normal" looks like.

Isolation Forest and Financial Applications

What is Isolation Forest?

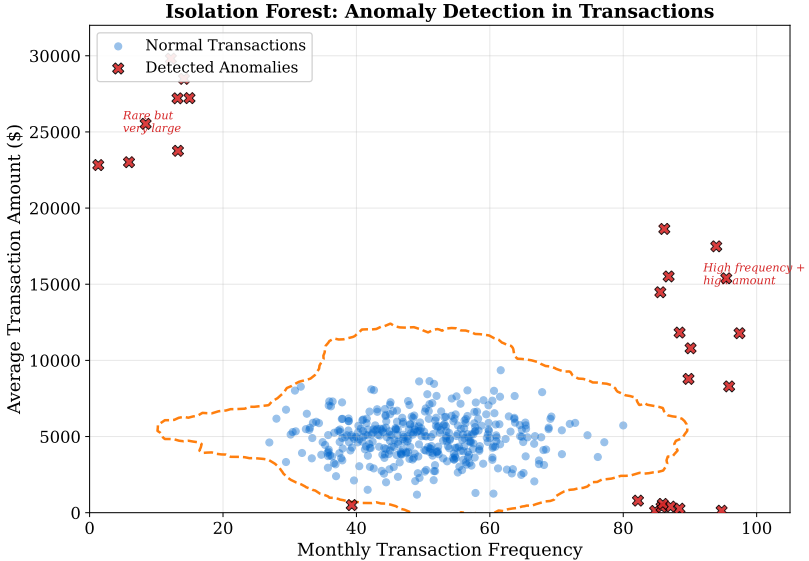
- Isolates anomalies by random recursive partitioning
- Key insight: anomalies are *few* and *different*
- Anomalies need fewer splits to isolate $\Rightarrow$ shorter path length
- Normal points need many splits $\Rightarrow$ longer path length

Financial applications:

- Anti-Money Laundering (AML) transaction monitoring
- Unusual trading pattern detection
- Insider trading surveillance
- Network intrusion detection

Anomaly detection complements classification. In practice, banks use both: supervised models for known fraud patterns, anomaly detection for novel threats.

Anomaly Detection in Action



What is a Neural Network? — Structure and Training

Definition:

- A computational model inspired by biological neurons
- Layers of interconnected nodes that learn weighted combinations of inputs
- Universal function approximator (given enough neurons)

Architecture:

- **Input layer:** raw features (income, credit score, ...)
- **Hidden layers:** learned representations
- **Output layer:** prediction (probability of default)

What is Backpropagation?

- Algorithm to train neural networks
- Computes how much each weight contributed to the error
- Adjusts weights using gradient descent
- "Propagates" error backward through layers

Neural networks learn by adjusting connection weights. Backpropagation computes how much each weight contributed to the error—then nudges it in the right direction.

Neural Networks — When to Use (and When Not To)

When to use neural networks in finance:

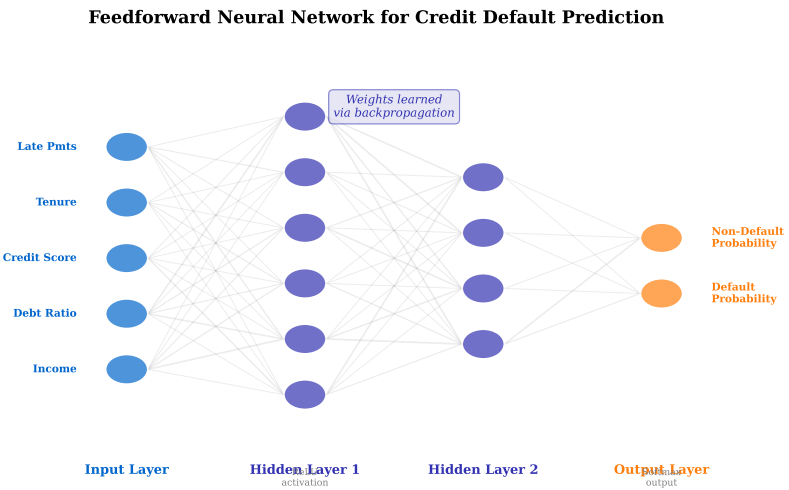
- Large datasets ($>$ 100,000 samples)
- Complex nonlinear patterns
- Unstructured data (text, images)

When NOT to use:

- Small datasets (overfit quickly)
- Explainability is required
- Tabular data (tree-based methods often win)

For tabular financial data, XGBoost typically outperforms neural networks. Neural networks shine with text (NLP for sentiment) and images (check scanning).

Neural Network Architecture



- **What you see:** Input layer (features), two hidden layers (learned representations), output layer (probability of default)
- **Key pattern:** Each connection has a weight; backpropagation adjusts weights to minimize prediction error

The ML Workflow for Financial Applications

End-to-end pipeline for a production ML model:

1. **Problem framing:** Is this classification or regression? What metric matters?
2. **Data collection:** Historical transactions, customer data, market data
3. **Feature engineering:** Create domain-specific features (ratios, rolling stats, lags)
4. **Train/validation split: Time-based split** (never shuffle financial time series!)
5. **Model selection:** Start simple (logistic regression), then try tree-based (XGBoost)
6. **Hyperparameter tuning:** Use cross-validation (time-series aware)
7. **Evaluation:** Confusion matrix, precision/recall, AUC on held-out test set
8. **Model validation:** Independent review (regulatory requirement for banks)
9. **Deployment:** Monitor for drift, retrain periodically

This pipeline is not a one-time exercise. Financial models must be continuously monitored and retrained as market conditions change.

Model Selection Guide for Finance

Model	Best For	Explainable?	Financial Use
Logistic Regression	Simple classification	Yes (coefficients)	Credit scoring baseline
Decision Tree	Rule-based decisions	Yes (path)	Underwriting rules
Random Forest	General-purpose	Partial (importance)	Default prediction
XGBoost	Tabular data, competitions	Partial (SHAP)	Fraud, credit, trading
Neural Network	Unstructured data	No (black box)	NLP, image, large data
Isolation Forest	Anomaly detection	Limited	AML, surveillance

Start with the simplest model that meets your performance and explainability requirements. In regulated finance, explainability often trumps accuracy.

Common Pitfalls — Data and Modeling

Data Pitfalls:

- **Look-ahead bias:** using future information in training features
- **Survivorship bias:** only analyzing companies that survived
- **Label leakage:** target variable encoded in a feature
- **Non-stationarity:** market regimes change; past $\neq$ future

Modeling Pitfalls:

- Overfitting to in-sample data
- Using accuracy on imbalanced datasets
- Random k-fold on time-series data

Data pitfalls are the most dangerous because they produce models that look good in backtesting but fail in production. Look-ahead bias is the single most common error in financial ML.

Common Pitfalls — Deployment and Regulatory

Deployment Pitfalls:

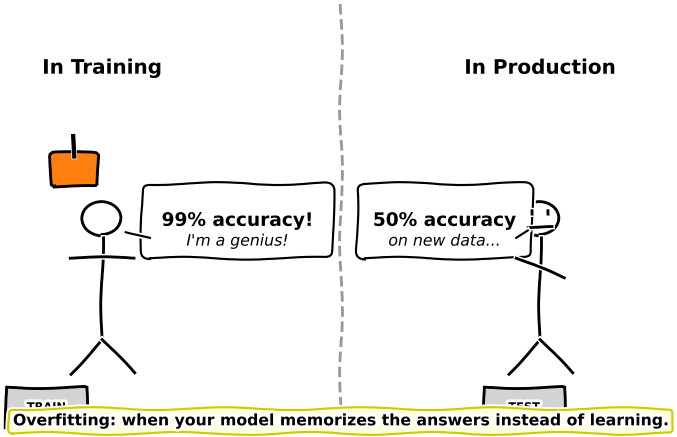
- **Model drift:** performance degrades as data distribution changes
- **Concept drift:** the definition of "fraud" evolves
- Not monitoring model performance in production

Regulatory Pitfalls:

- Using protected attributes (race, gender) as features
- Inability to explain model decisions (EU AI Act, SR 11-7)
- No model validation process
- Failure to document model limitations

Knowing what can go wrong is as important as knowing how to build the model. Every pitfall listed here has caused real financial losses or regulatory fines.

One Last Thought...



Sometimes the best way to remember a concept is to laugh about it.

Case: Knight Capital (2012)

What happened

- 1 August 2012: a NYSE software deployment left an old test routine (SMARS) active in production on 7 of 8 servers
- Within 45 minutes Knight had taken on a 3.5bn-share long position it did not intend
- Realized loss: USD 440M (*SEC Administrative Proceeding 3-15570 16 Oct*, 2013)
- Knight's equity capital was wiped out and the firm was acquired (Getco) in December 2012
- SEC enforcement: **$**12M civil penalty; first action under Reg SCI (proposed shortly after)

Why it matters here

- Algorithmic / ML systems amplify deployment defects beyond human-loop control window
- Direct motivator for the SEC's Regulation Systems Compliance and Integrity (Reg SCI, 2014)
- Worked example for M5L4 MLOps: deployment review + circuit breakers are not optional for trading systems

---

Full writeup: `v4/cases/case_M5_knight_capital.md`. **Host: M5 L1 ML Foundations.**

Case: Knight Capital – Lessons

- Eight-server deploy with one server stale = entire system wrong, not 87.5 percent right
- Without a kill-switch the loss was bounded by capital, not by detection time
- Code archeology: the dormant SMARS routine had been unused for eight years before re-activation
- Mean-time-to-detect, not mean-time-to-failure, is the operational risk metric that matters for ML

See also: M5 L1 ML Foundations; full writeup at `v4/cases/case_M5_knight_capital.md`.

Key Takeaways

1. **Classification vs. regression** is the first modeling decision—it determines your metrics, loss function, and evaluation strategy
2. **Overfitting is the primary risk** in financial ML because markets are noisy, non-stationary, and sample sizes for rare events are small
3. **The bias-variance tradeoff** governs model selection: too simple (high bias) vs. too complex (high variance)
4. **Tree-based ensembles** (random forest, XGBoost) dominate tabular financial data; start simple, add complexity only when justified
5. **Feature engineering** often matters more than model choice—domain expertise is your competitive advantage
6. **Evaluation metrics must match business costs:** accuracy is misleading for imbalanced problems; use precision, recall, F1, or AUC
7. **Anomaly detection** (Isolation Forest) complements supervised learning when labels are scarce or fraud patterns evolve
8. **Cross-validation must be time-aware** in finance—never leak future information into training

---

ML in finance is not about the fanciest algorithm. It is about the right problem framing, the right features, the right evaluation, and rigorous validation.

Summary: Lesson 5.1

Concepts Covered:

- Supervised learning: classification vs. regression
- Bias-variance tradeoff and overfitting
- Cross-validation (time-series aware)
- Decision trees, random forests, gradient boosting (XGBoost)
- Feature engineering for financial data
- Confusion matrix, precision, recall, F1-score
- ROC curve and AUC
- Anomaly detection (Isolation Forest)
- Neural networks and backpropagation

What comes next:

- Lesson 5.2: ML in production—deployment, monitoring, model drift
- Lesson 5.3: Natural Language Processing (NLP) in finance
- Lesson 5.4: Explainability and fairness in financial AI

Practical advice:

- Start with logistic regression or random forest
- Invest time in feature engineering
- Always use time-aware validation
- Monitor models after deployment
- Explain before you deploy

ML foundations are prerequisite knowledge for every subsequent lesson in Module 5. Master these concepts before moving to advanced topics.

Common Misconceptions About ML in Finance

Misconception	Reality
"Higher training accuracy = better model"	A model with 100% training accuracy has memorised noise. Out-of-sample accuracy is the only honest metric. See bias-variance slide.
"Random Forests can't overfit"	They reduce overfitting vs single trees, but can still memorise with too many deep trees. Tree depth and min-samples-per-leaf are critical regularisers.
"More features = better predictions"	Curse of dimensionality: each feature adds noise and interaction complexity. Models with fewer well-chosen features often win out-of-sample.
"Cross-validation proves the model works"	Standard k-fold CV leaks information in time-series. Use purged/embargoed time-series CV (de Prado 2018) for financial data. See M5L3.
"Accuracy is the right metric"	For imbalanced classes (fraud: 0.1%), predicting "not fraud" always gives 99.9% accuracy. Use precision/recall/F1 or PR-AUC.
"ML replaces statistics"	ML is largely applied statistics with bigger datasets and higher compute. Knowing OLS, bias, variance, and significance is still necessary.

The biggest risk in financial ML is not bad code — it is a confident model trained on lookahead-leaking data.

Self-Check: ML Foundations

Attempt these before turning the page.

1. [Understand] Explain the bias-variance tradeoff. Which side of the tradeoff is typically worse for production financial ML, and why?
2. [Apply] Fraud detector confusion matrix: TP = 80, FP = 20, TN = 9800, FN = 100. Compute precision, recall, F1, and accuracy. Which metric is misleading, given that the base rate of fraud is 1.8%?
3. [Evaluate] A client proposes a Random Forest trained on 5 years of daily returns to predict next-day S&P 500 direction. Name two likely failure modes and the single correction that matters most.

Solutions hidden unless `\solutionstrue` is set before compiling.