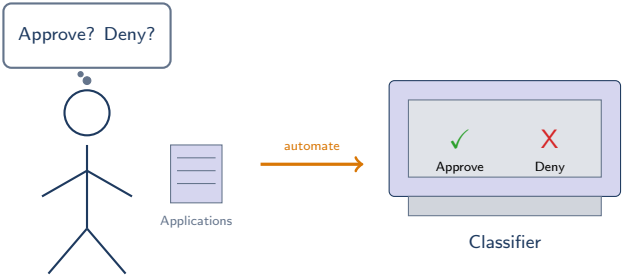
Classification in Supervised Learning

Data Science with Python – BSc Course

90 Minutes

The Classification Challenge



"What if machines could learn the pattern?"

Every loan decision, fraud flag, and diagnosis is a classification problem.

Learning Objectives

After this lecture you will be able to:

1. **Explain** how the sigmoid function converts scores into probabilities
2. **Compare** decision boundaries of logistic regression vs. decision trees
3. **Evaluate** classifiers using confusion matrix, precision, recall, and AUC
4. **Apply** SMOTE and threshold tuning to handle imbalanced data
5. **Justify** metric selection based on business cost

Bloom's taxonomy: Remember → Understand → Apply → Analyze → Evaluate

---

These five objectives map to Bloom's levels 2–5.

Bridge: From Regression to Classification

What you already know

- Linear regression predicts continuous values
- Train/test split prevents overfitting
- MSE and R^2 measure regression quality

What this lecture adds

- Output is a probability, not a number
- Decision boundaries separate classes
- Imbalanced data needs special care

Sigmoid | Boundary | Trees | Metrics | Imbalance | Business | Limits

Seven sections, one lecture: from probability to business decisions.

Can we automate the boundary between yes and no?

Three sub-questions drive today's lecture:

1. How does a model turn features into a probability?
2. How do we measure whether the boundary is in the right place?
3. What happens when one class vastly outnumbers the other?

Every slide answers at least one of these three questions.

Why Classification Matters

When classification fails, billions are lost.

- **2008 mortgage crisis**: Models approved borrowers who could not repay — massive false negatives on default risk
- **London Whale (2012)**: Risk models failed to flag anomalous positions — \$6.2 B loss at JPMorgan
- **Wirecard (2020)**: Auditors classified fraudulent accounts as legitimate

Definition. *Classification* assigns an observation to one of a finite set of categories (e.g., default/no-default, fraud/legitimate).

Classification errors carry asymmetric costs — the topic of Section 4.

The Cost of Being Wrong

	Actually Good	**Actually Bad**
Predict Good	True Negative — correct	False Negative — missed risk
Predict Bad	False Positive — lost revenue	True Positive — caught risk

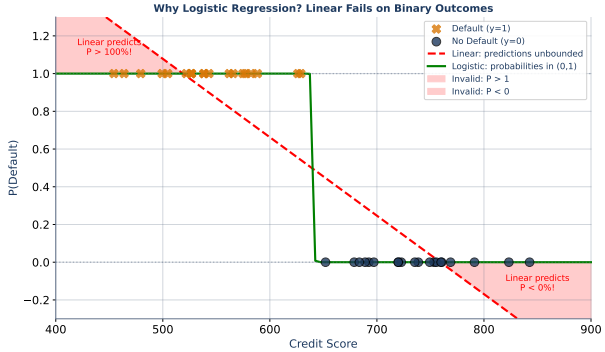
- **FP** (approve bad loan): bank absorbs the loss
- **FN** (reject good customer): bank loses lifetime revenue

The optimal threshold depends on which error costs more.

This 2×2 table is the confusion matrix — the foundation of all classification metrics.

Regression vs. Classification

What you see: regression predicts unbounded values; classification predicts probabilities between 0 and 1.



The S-curve "squashes" any input into the probability range — no negative probabilities, no values above 1.

Classification in Three Sentences

1. **Score**: combine features into a single number $z = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$
2. **Transform**: pass through the sigmoid $p = \frac{1}{1+e^{-z}}$
3. **Decide**: compare p to a threshold t — if $p \geq t$, predict positive

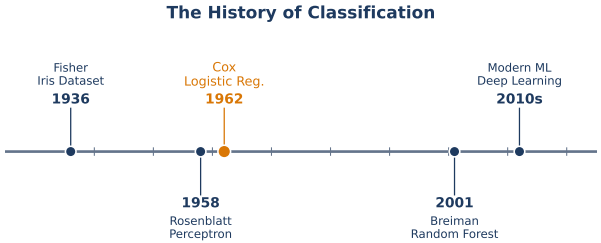
Worked example. Credit score 720, debt-to-income 0.25, age 35.

- Model outputs $z = -3.5$, so $p = \frac{1}{1+e^{3.5}} \approx 0.03$
- With threshold $t = 0.05$: predict **no default** (approve loan)

Three steps: score → probability → decision. The threshold is the lever you control.

A Brief History of Classification

What you see: milestones from Fisher's 1936 discriminant analysis to modern ensemble classifiers.



80+ years of research, same core idea: find the boundary that separates classes.

Why Not If-Else Rules?

A first instinct: write manual rules.

- **Don't scale**: 50 features $\times$ interactions $=$ thousands of rules
- **Miss interactions**: "high income $+$ high debt" is risky, but neither alone triggers a rule
- **Can't adapt**: when data shifts, every rule must be rewritten manually

Machine learning classifiers *learn* boundaries from data and update automatically when retrained.

Rules are brittle; learned boundaries are flexible.

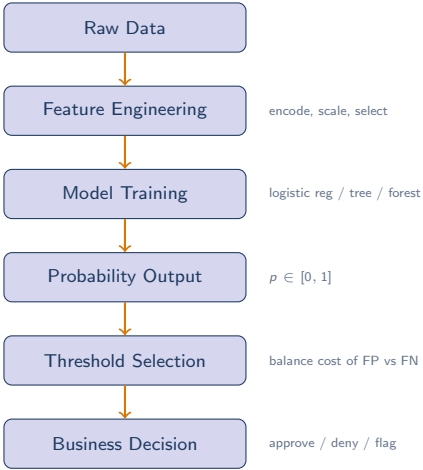
The Classification Toolbox

Model	Boundary Shape	Best For
Logistic Regression	Linear (hyperplane)	Interpretable baseline
Decision Tree	Axis-aligned rectangles	Explainability, regulations
Random Forest	Averaged rectangles	General-purpose, robust
SVM	Flexible (kernel-dependent)	High-dimensional data
Neural Network	Arbitrary nonlinear	Large data, complex patterns

Today: logistic regression and decision trees. Both transparent enough for regulated industries.

No single model wins everywhere — model selection depends on data size, interpretability, and cost.

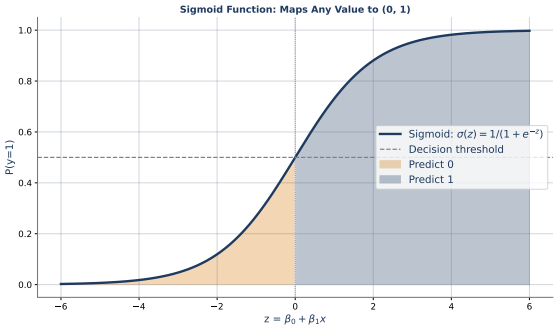
The Classification Pipeline



Six steps from raw data to action. The threshold is where business logic meets statistics.

The Sigmoid Function

What you see: the S-curve maps any real-valued input to a probability between 0 and 1.

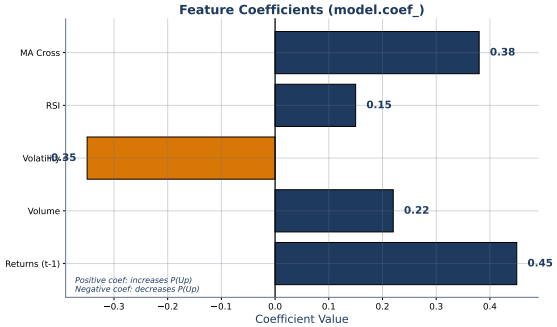


Worked example: $z = 0 \Rightarrow p = 0.50$; $z = 2 \Rightarrow p = 0.88$; $z = -3 \Rightarrow p = 0.05$

The sigmoid is the bridge between a linear score and a probability.

Which Features Drive the Prediction?

What you see: bars showing each feature's coefficient — positive pushes toward default, negative toward safety.

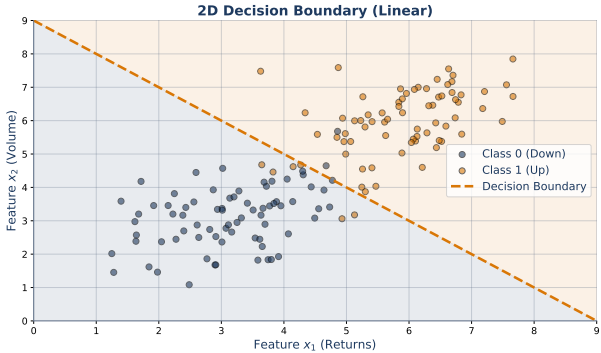


Coefficients are log-odds: e^{β} gives the odds ratio per unit change.

Interpretability is a key advantage of logistic regression over black-box models.

Logistic Regression: A Linear Boundary

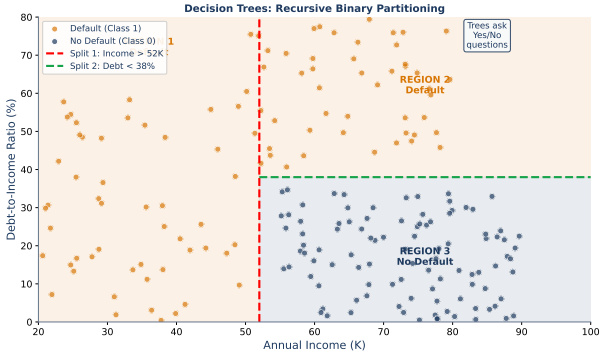
What you see: the decision boundary is a straight line — logistic regression is fundamentally linear.



Linear boundaries work well when classes are roughly separable by a hyperplane.

Decision Trees: A Different Approach

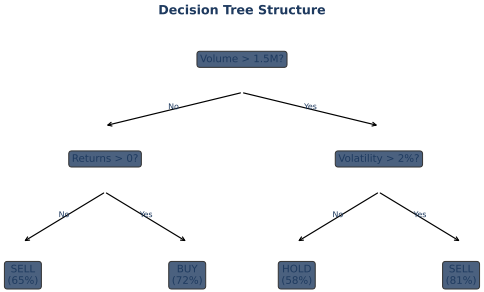
What you see: a tree splits the feature space with axis-aligned cuts — each split asks one yes/no question.



Trees are non-parametric: no assumption about the shape of the boundary.

Reading a Decision Tree

What you see: each internal node asks a yes/no question; follow the path to a leaf for the prediction.

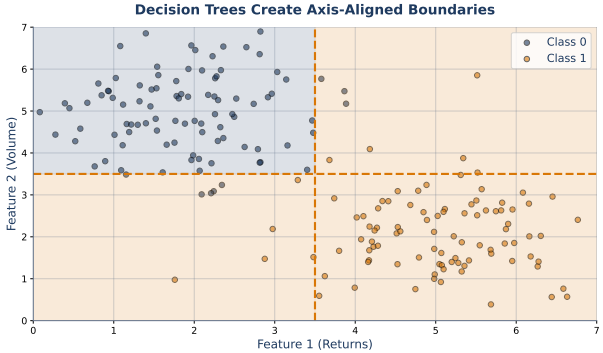


- Depth = number of questions asked
- Leaves = final predictions (class label or probability)

A tree is a flowchart that a regulator can read line by line.

Tree Boundaries Are Rectangular

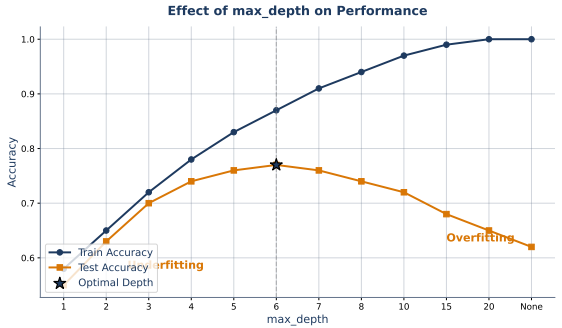
What you see: axis-aligned splits produce rectangular decision regions.



Compare to the diagonal line of logistic regression — trees capture interactions but only along axes.

Overfitting: When Trees Go Too Deep

What you see: as max depth increases, training accuracy rises but test accuracy eventually drops.



- Shallow tree = underfitting (high bias)
- Deep tree = overfitting (high variance)
- Sweet spot found via cross-validation

The bias-variance tradeoff is universal — it applies to every classifier.

Model Says Fraud — Is It Right?

Scenario. 10 000 transactions per day. 50 are actual fraud. Your model flags 100.

- Of the 100 flagged: 40 are real fraud (TP), 60 are legitimate (FP)
- Of the 9 900 not flagged: 10 are missed fraud (FN), 9 890 correct (TN)

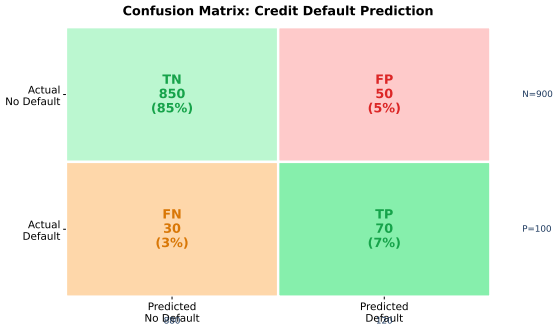
Questions:

1. What fraction of flags are correct? *(Precision = 40/100 = 40%)*
2. What fraction of fraud is caught? *(Recall = 40/50 = 80%)*
3. Overall accuracy? *(9 930/10 000 = 99.3% — misleading!)*

Accuracy hides the thing you care about: how many frauds did you actually catch?

The Confusion Matrix

What you see: a 2×2 grid of TP, FP, FN, TN — every metric is a ratio of these four cells.

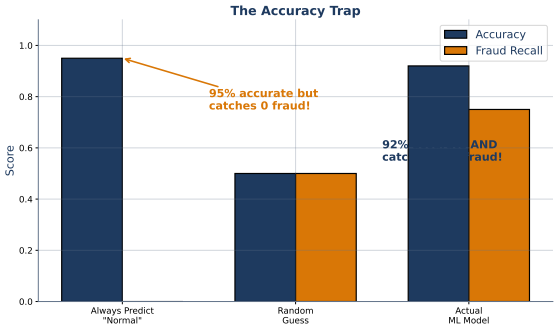


$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}, \quad \text{Accuracy} = \frac{\text{TP+TN}}{\text{All}}$$

Memorize the matrix; every metric is just a different slice of these four numbers.

The Accuracy Trap

What you see: with 99% non-fraud, predicting "all legitimate" gives 99% accuracy but catches zero fraud.

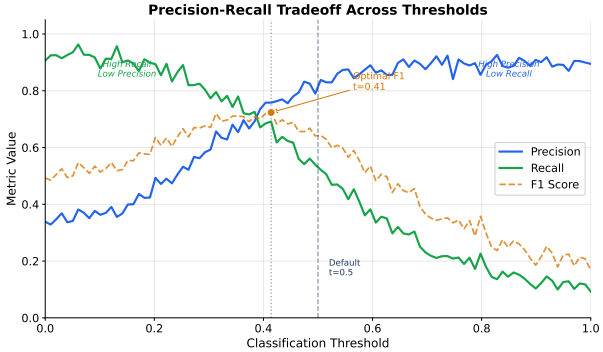


- A model that does *nothing* scores 99%
- Accuracy is useless under class imbalance

Never report accuracy alone on imbalanced data. Always include precision, recall, or F1.

Precision vs. Recall Tradeoff

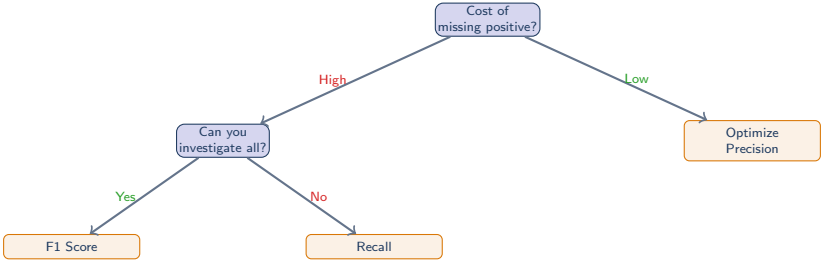
What you see: lowering the threshold catches more positives (higher recall) but also more false alarms (lower precision).



You cannot maximize both simultaneously — the threshold sets the balance.

Think-Pair-Share: Which Metric?

Scenario. Your fraud team can investigate 50 cases per day. The model flags 200.

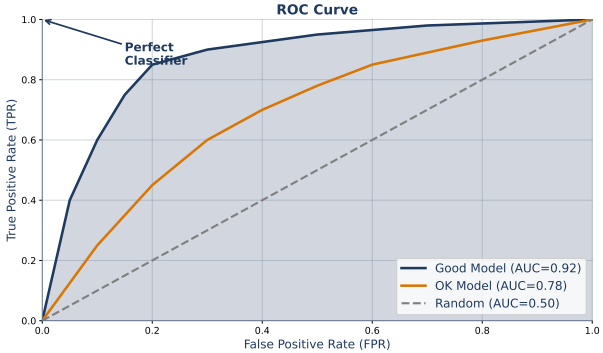


Discuss with your neighbour (2 min): should the fraud team optimize precision or recall?

Limited investigation capacity → precision matters. Unlimited → recall dominates.

ROC Curve: All Thresholds at Once

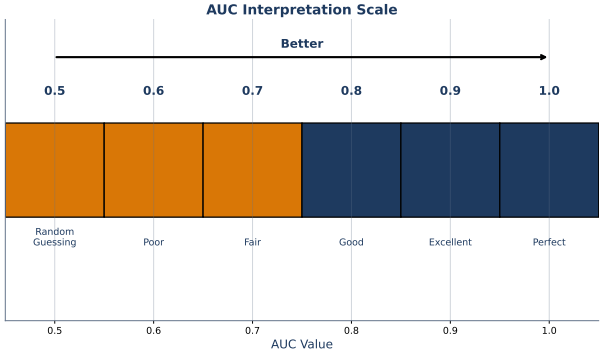
What you see: the ROC curve plots true positive rate vs. false positive rate at every possible threshold.



Closer to the top-left corner = better. The diagonal is random guessing.

AUC: One Number to Summarize

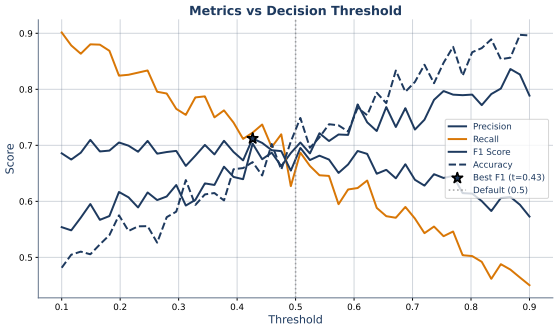
What you see: AUC 0.5 = random coin flip, 0.8 = good, 0.9+ = excellent discrimination.



AUC is threshold-independent — useful for comparing models before choosing a threshold.

Every Metric Shifts with the Threshold

What you see: precision, recall, F1, and accuracy all change as the classification threshold moves from 0 to 1.



- There is no "correct" threshold — only a business-appropriate one
- Report metrics at the threshold you will deploy

Always specify the threshold when reporting classification performance.

The Imbalanced-Data Reality

In production, the interesting class is almost always rare.

- Credit card fraud: $\sim$0.1% of transactions
- Loan default: $\sim$3% of borrowers
- Customer churn: $\sim$5% per month

A naive model learns to always predict the majority class — and achieves high accuracy while being useless.

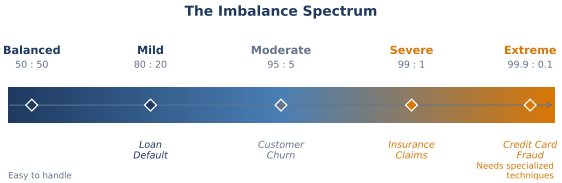
Three categories of solutions:

1. Resample the data (oversample minority / undersample majority)
2. Adjust the algorithm (class weights, cost-sensitive learning)
3. Adjust the threshold (move the decision boundary post-training)

Imbalance is not a bug — it reflects reality. The fix is in how we train and evaluate.

The Imbalance Spectrum

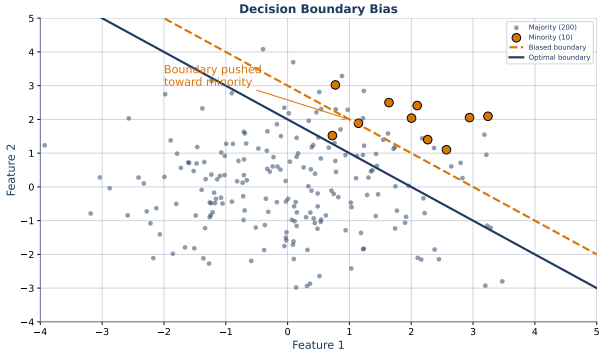
What you see: from mild imbalance (70:30) to extreme imbalance (99.9:0.1) — different ratios need different strategies.



Mild imbalance may need no treatment; extreme imbalance demands resampling + threshold tuning.

How Imbalance Shifts the Boundary

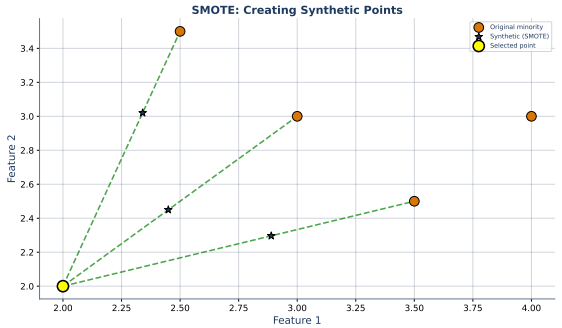
What you see: the decision boundary shifts toward the minority class, misclassifying the group we care about most.



The model "sees" more majority examples and learns to favor them — a data problem, not a model bug.

SMOTE: Creating Synthetic Minorities

What you see: SMOTE generates new minority samples by interpolating between existing neighbours.

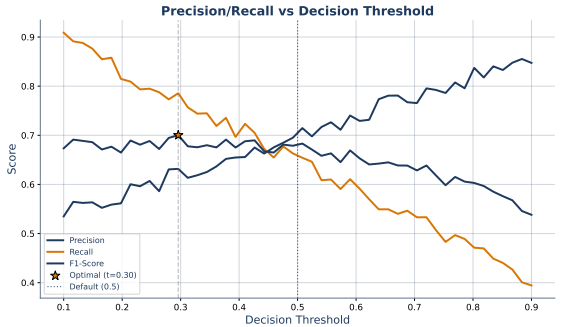


- Pick a minority point, find its k nearest minority neighbours
- Create a new point on the line segment between them
- Repeat until classes are balanced

SMOTE adds information, not noise — unlike simple duplication of existing rows.

Threshold Tuning

What you see: moving the classification threshold trades recall for precision — the model stays the same, only the decision rule changes.

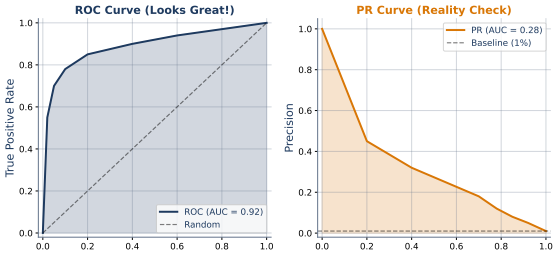


- Lower threshold → more flags → higher recall, lower precision
- Higher threshold → fewer flags → higher precision, lower recall

Threshold tuning is free — no retraining required. Always tune after model selection.

PR Curve vs. ROC Curve Under Imbalance

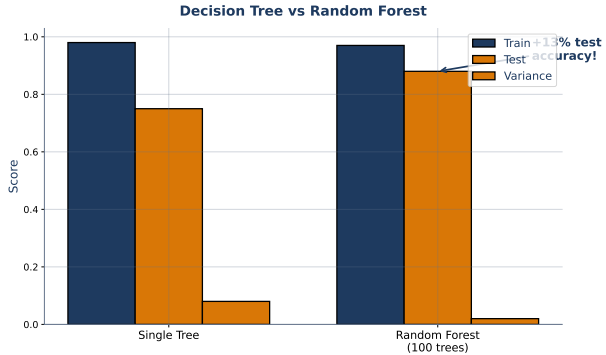
What you see: the PR curve gives an honest picture under imbalance; the ROC curve can look optimistic.



Rule of thumb: if the positive class is $<$10% of data, report the PR curve, not just ROC.

Single Tree vs. Random Forest

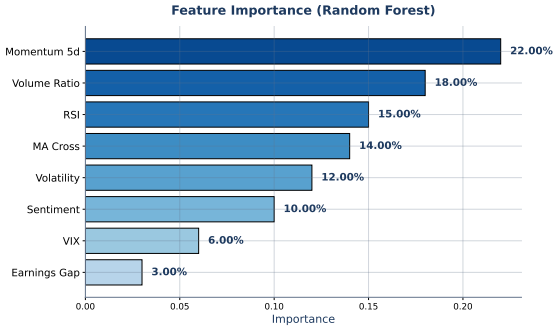
What you see: a single tree has high variance; averaging many trees (a forest) produces stable predictions.



Ensembles reduce variance without increasing bias — the "wisdom of crowds" for classifiers.

Feature Importance and Model Selection

What you see: feature importance bars show which variables the tree relies on most.



Small + Simple Logistic Reg	**Small + Complex** Decision Tree
Large + Simple Logistic Reg / RF	**Large + Complex** Random Forest / NN

Match model complexity to data size and pattern complexity.

Discussion: Designing a Fraud Detection System

Scenario. You are building a fraud detection system for a European bank.

- 500 million transactions per year
- Fraud rate: 0.1% ($\approx$ 500 000 fraudulent transactions)
- Cost of a false positive: €50 (investigation cost)
- Cost of a false negative: €5 000 (average fraud loss)

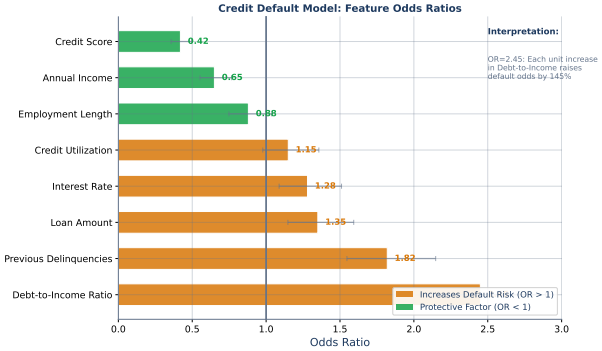
Group task (5 minutes):

1. Should you optimize precision or recall?
2. What threshold would you set?
3. Which model family would you start with, and why?

FN is 100$\times$ more costly than FP — this asymmetry should drive every design choice.

Case Study: Credit Default Prediction

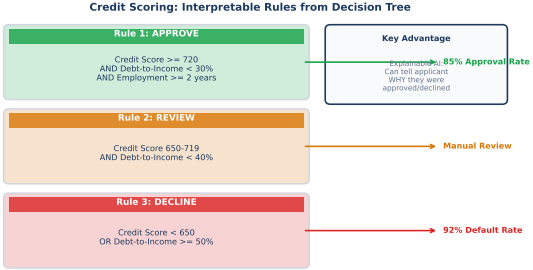
What you see: predicted probability of default for each applicant, with a threshold line separating approve/deny.



Logistic regression remains the industry standard for credit scoring — interpretable and auditable.

Interpretable Rules for Regulators

What you see: a decision tree produces human-readable rules that a compliance officer can verify.

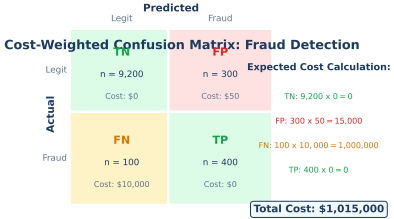


- GDPR Art. 22: right to explanation for automated decisions
- Trees and logistic regression satisfy this requirement directly

In regulated finance, interpretability is not optional — it is a legal requirement.

Asymmetric Costs in Practice

What you see: the cost matrix shows that missing a fraud case is orders of magnitude worse than a false alarm.

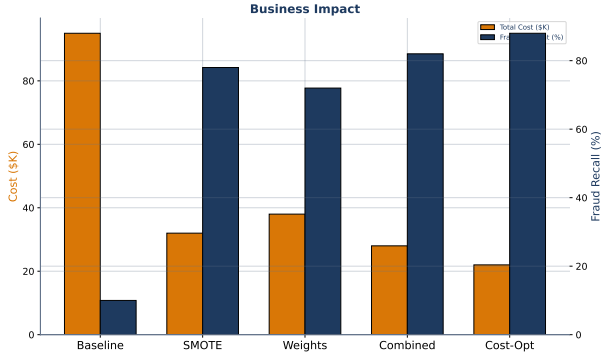


- Cost-sensitive learning weights the loss function by these costs
- Equivalent to adjusting the threshold post-training

When costs are asymmetric, accuracy is the wrong metric. Expected cost is the right one.

From Metrics to Dollars

What you see: translating a 5-point recall improvement into actual dollar savings for the business.



A model is only as good as the business impact it creates. Report metrics and dollars.

Four Limitations to Remember

1. **Bias in, bias out**: if historical data reflects discrimination, the model will too (e.g., biased lending data $\rightarrow$ biased credit scores)
2. **Correlation $\neq$ causation**: the model finds patterns, not causes. A feature correlated with default may not *cause* default.
3. **Distribution shift**: a model trained on 2019 data may fail on 2020 pandemic data. Monitor performance continuously.
4. **Interpretability tradeoff**: more complex models (RF, NN) are harder to explain. Regulation may require simpler models.

Every model is a simplification of reality — know its boundaries.

Covered Today vs. What's Next

Covered today

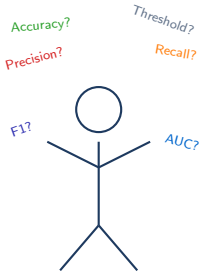
- Sigmoid and probability output
- Linear vs. tree boundaries
- Confusion matrix, precision, recall, AUC
- SMOTE, threshold tuning
- Business cost integration

What's next

- Support Vector Machines (non-linear kernels)
- Neural networks for classification
- Model deployment and monitoring
- Fairness and bias mitigation

Today's foundations — metrics, imbalance, cost — apply to every future classifier.

From Overwhelmed to Organized



- ✓ 1. What is the cost?
- ✓ 2. Imbalanced?
- ✓ 3. Pick metric
- ✓ 4. Tune threshold

"Decisions, not just predictions."

Classification is not about the model — it is about the decision the model enables.

Five Takeaways

1. **Probabilities enable decisions.** The sigmoid maps scores to $[0, 1]$; the threshold maps probabilities to actions.
2. **Accuracy lies under imbalance.** A 99% accuracy model may catch 0% of fraud.
3. **Imbalance is the norm, not the exception.** SMOTE, class weights, and threshold tuning address it.
4. **The right metric = the business cost.** FP and FN have different prices; the metric should reflect that.
5. **Every model has limits.** Bias, shift, and interpretability constrain what classifiers can do.

Classification in supervised learning: from probability to decision, guided by cost.