# Big Data in Finance

## Week 4: Classification Methods

Dr Daniele Bianchi

Spring 2026

# Today's Agenda

From Regression to Classification

Logistic Regression

Classification Trees

The LendingClub Application

Model Evaluation

Threshold Optimization

From Models to Business Decisions

Summary and Next Steps

# From Regression to Classification

# Recap: What We've Done So Far

**Week 2–3:** Predicting **continuous** outcomes

- **The question:** What will next month's market return be?
- **Linear methods:** Ridge, Lasso, Elastic Net
- **Tree-based methods:** Decision Trees, Random Forests, Gradient Boosting
- **Output:** A number (e.g., predicted return of $+2.3\%$)

## This Lecture: A Different Kind of Question

Instead of "how much?" we ask "**which category**?"

# The Classification Problem

**Classification:** Predict which **category** an observation belongs to

**Examples in finance:**

- Will this borrower **default** or **not default**?
- Is this transaction **fraudulent** or **legitimate**?
- Will this company go **bankrupt** or **survive**?
- Will the market go **up** or **down** tomorrow?

**Key difference from regression:**

- Output is a **category** (default/no default), not a number
- Often we also want the **probability** of each category

## Good News

Many concepts from regression carry over — we'll build on what you know!

# Our Running Example: Credit Risk

**The business problem:** Should we approve this loan application?

**What we observe about the borrower:**

- Annual income: \$65,000
- Employment: 5 years at current job
- FICO credit score: 680
- Debt-to-income ratio: 28%
- Requesting: \$15,000 loan

**What we want to predict:**

- Will he/she default?
- **How likely** is default?

**The economic decision:**

- Approve or deny?
- What interest rate?

**Data:** We will use LendingClub peer-to-peer loans as a real example

# What Makes Classification Different?

| | **Regression** | **Classification** |
|---|---|---|
| Output | Continuous number | Category (class label) |
| Example | Return $= +2.3\%$ | Default = Yes/No |
| Often also want | Point estimate | **Probability** of each class |
| Error measure | Squared error | Misclassification rate |

**Why probabilities matter in credit risk:**

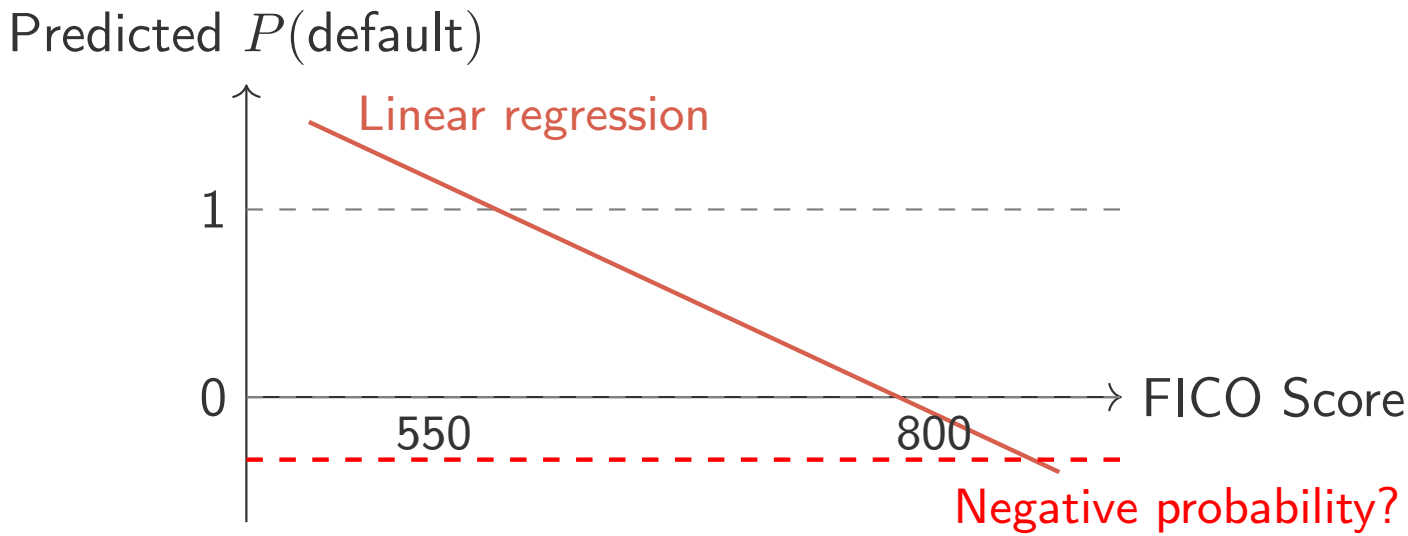
- Not just "approve" or "deny" — need to **price the risk**
- Higher default probability $\Rightarrow$ higher interest rate
- Need accurate probabilities for regulatory capital calculations

**Today's goal:** Learn methods that output valid probabilities between 0 and 1

# Why Can't We Just Use Regression?

**Naive idea:** Code default as 1, no default as 0, run linear regression

**Problem:** Predictions can be nonsensical!




**The issue:** Linear regression can predict values outside $[0, 1]$

- A probability of $-0.2$ or $1.3$ does not make sense!
- We need methods designed for classification

# Key Challenges in Credit Risk Prediction

**Before we dive into methods, let's understand the challenges:**

1. **Class imbalance:** Defaults are relatively rare
   - Typical portfolio: 10–20% default rate
   - A model that always predicts "no default" is 80–90% accurate!
   - But it's completely useless for identifying risky borrowers
2. **We need well-calibrated probabilities:**
   - If model says "15% chance of default," roughly 15% should actually default
   - Critical for pricing and capital calculations
3. **Interpretability and fairness:**
   - Regulators require explanations: "Why was this loan denied?"
   - Cannot discriminate based on protected characteristics

*We'll address all of these on Wednesday's lecture.*

# Logistic Regression

# The Core Idea: Predicting Probabilities

**What we want:** A model that predicts
$P(\text{default} \mid \text{borrower characteristics})$

**Requirements:**

1. Output must be between 0 and 1 (it's a probability!)
2. Higher risk factors should increase the probability
3. Should be interpretable (what drives the prediction?)

**The solution: Logistic regression**

- Most widely used classification method
- Foundation of credit scoring for decades
- Interpretable: Can explain why a loan was approved/denied

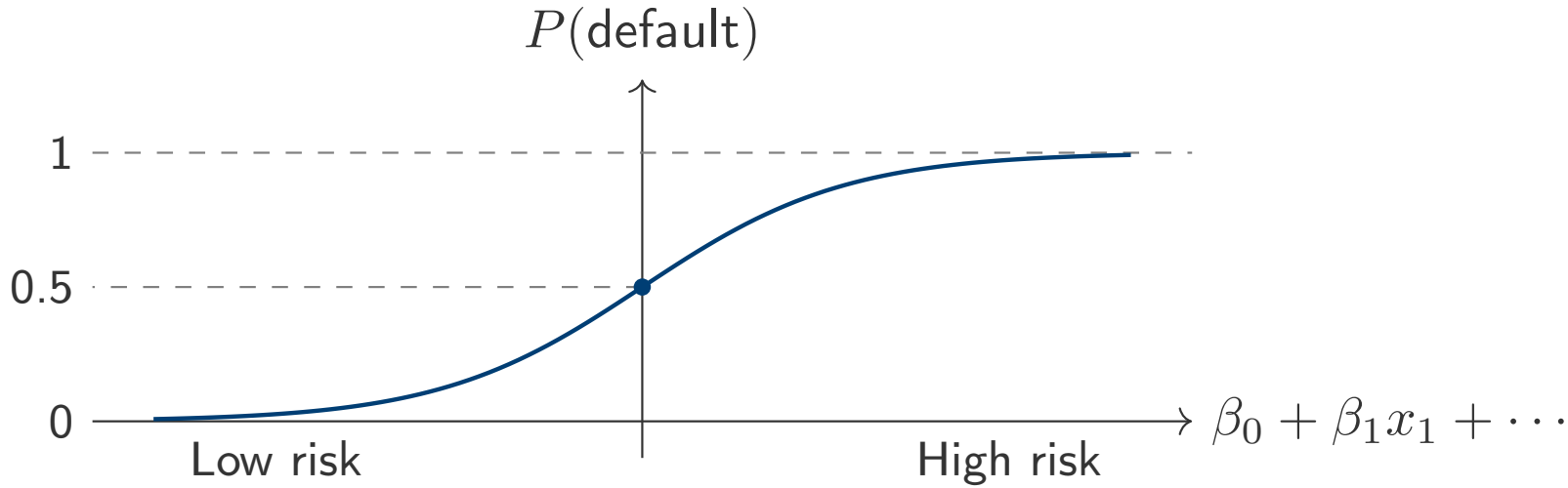
## Key Insight

Logistic regression is like linear regression, but we transform the output to guarantee it's a valid probability

# From Linear to Logistic: The Sigmoid Function

**Linear regression:** $\widehat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$ (can be any number)

**Logistic regression:** Pass the linear combination through a **sigmoid function**




The sigmoid "squashes" any input into the range (0, 1):

$$P(\text{default}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots)}}$$

# A Concrete Example

**Suppose our model learns these coefficients:**

$$\beta_0 = 8 \quad \text{(intercept)}$$
$$\beta_{\text{DTI}} = 0.05 \quad \text{(debt-to-income ratio, in \%)}$$
$$\beta_{\text{FICO}} = -0.015 \quad \text{(credit score)}$$

**Low-risk borrower:** DTI = 25% and FICO = 750

$$\text{Linear part} = 8 + 0.05 \times 25 + (-0.015) \times 750 = 8 + 1.25 - 11.25 = -2.0$$

$$P(\text{default}) = \frac{1}{1 + e^{-(-2.0)}} = \frac{1}{1 + e^{2}} \approx \mathbf{0.12} \quad (12\%)$$

**High-risk borrower:** DTI = 45% and FICO = 620

$$\text{Linear part} = 8 + 0.05 \times 45 + (-0.015) \times 620 = 8 + 2.25 - 9.30 = 0.95$$

$$P(\text{default}) = \frac{1}{1 + e^{-0.95}} \approx \mathbf{0.72} \quad (72\%)$$

# Interpreting the Coefficients: Odds Ratios

**The coefficients tell us how each variable affects default risk**

**Interpretation via odds ratios:**

- For every 1-unit increase in $x_j$, the **odds of default** multiply by $e^{\beta_j}$

**Example:** If $\beta_{\text{DTI}} = 0.05$ (coefficient on debt-to-income ratio)

- $e^{0.05} \approx 1.05$
- Each 1% point increase in DTI increases odds of default by $\sim$5%

**This interpretability is valuable:**

- Regulators can understand what drives decisions
- Loan officers can explain denials to applicants
- Risk managers can identify key risk factors

# How Does Logistic Regression Learn? (Intuition)

**Linear regression:** We minimize squared errors (RSS)

**Logistic regression:** We maximize the **likelihood** of observing our data

**Intuition:**

- For borrowers who *did* default: model should predict high probability
- For borrowers who *didn't* default: model should predict low probability
- Find coefficients that make observed outcomes most "likely"

**Technical detail (for the curious):**

- No closed-form solution (unlike OLS)
- Computer finds optimal coefficients iteratively
- sklearn handles this automatically, you don't need to worry about it!

# Regularization: Same Ideas as Linear Regression

**Problem:** With many predictors, logistic regression can overfit too!

**Solution:** Add penalties, just like Ridge/Lasso

| Penalty Type | Effect |
|---|---|
| L2 (Ridge) | Shrinks all coefficients toward zero |
| L1 (Lasso) | Some coefficients become exactly zero (variable selection) |
| Elastic Net | Combination of both |

**Same intuition as Lecture 2:**

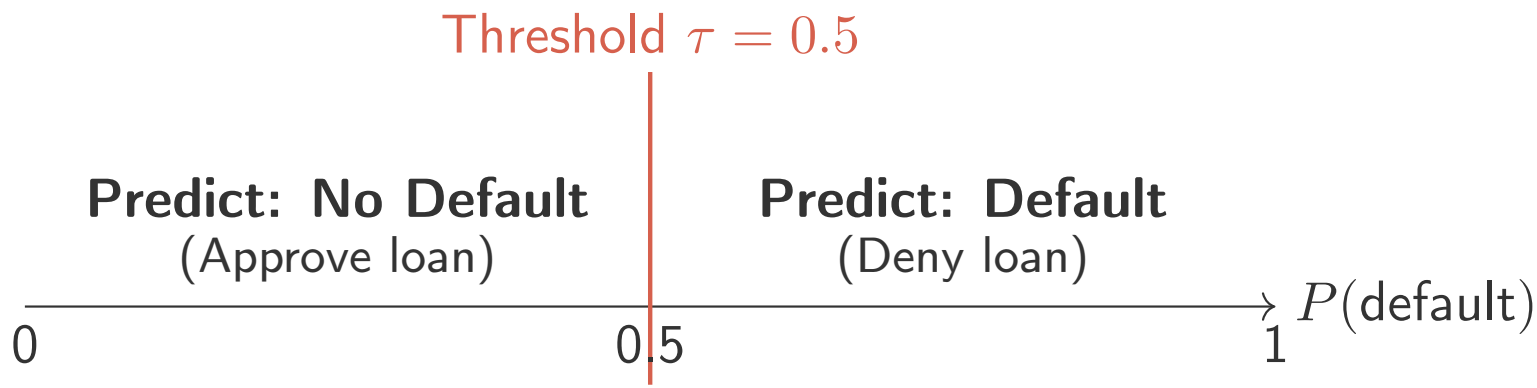
- Accept some bias to reduce variance
- Prevent model from fitting noise in the training data
- Tune the penalty strength using cross-validation

# Making Decisions: The Classification Threshold

Logistic regression gives us probabilities of default.

- How do we decide if a loan is predicted to default?

**Simple rule:** Predict "default" if $P(\text{default}) > \tau$

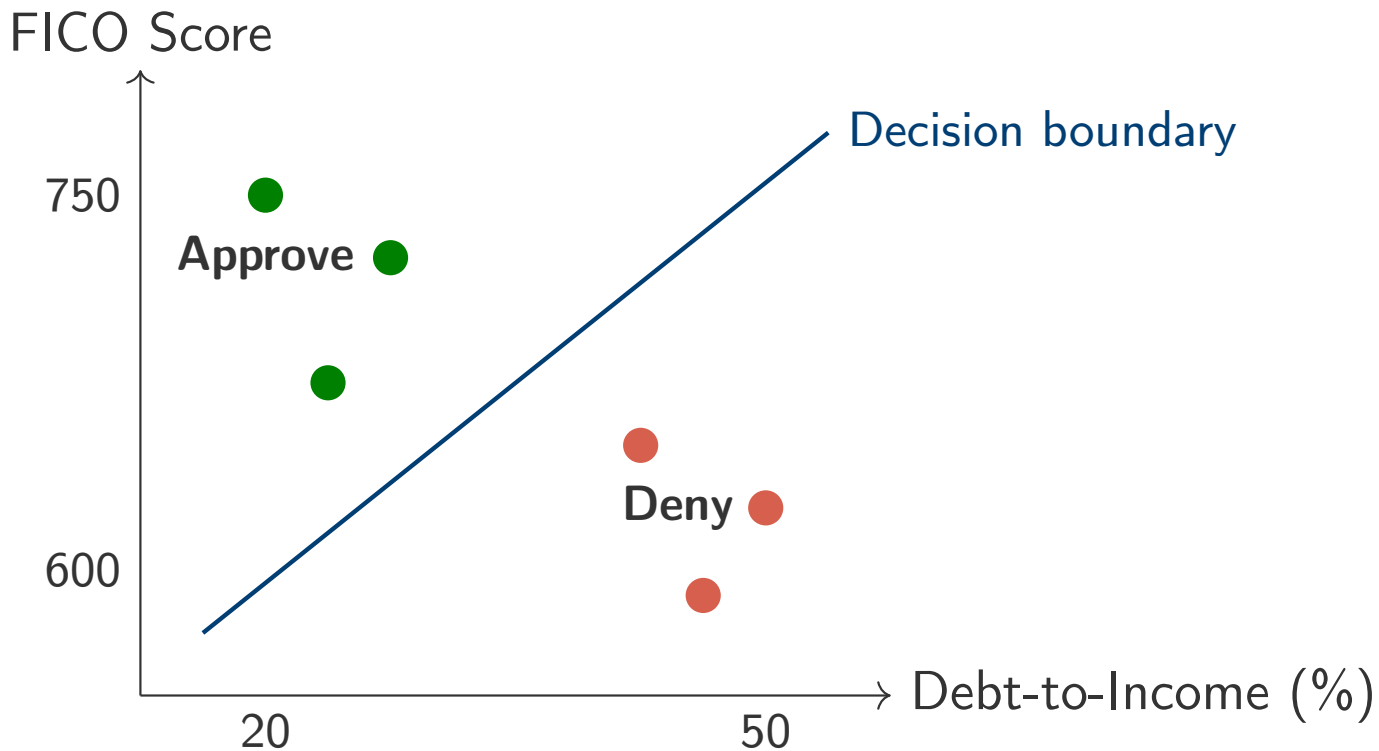



**But $\tau = 0.5$ is not always the right choice!**

- If defaults are very costly, use lower threshold (be more cautious)
- If rejecting good borrowers is costly, use higher threshold
- We'll discuss threshold optimization on Wednesday

# The Decision Boundary

**The threshold creates a decision boundary in feature space**




**Key point:** Logistic regression produces a **linear** decision boundary

- Simple and interpretable
- But may not capture complex patterns in the data

# Logistic Regression: Summary

**What logistic regression does:**

1. Takes borrower characteristics as inputs
2. Computes a weighted sum (like linear regression)
3. Passes through sigmoid to get probability in $(0, 1)$
4. Use threshold to convert probability to decision

**Advantages:**

- Interpretable coefficients (odds ratios)
- Widely understood by regulators and practitioners
- Can add regularization to prevent overfitting

**Limitation:**

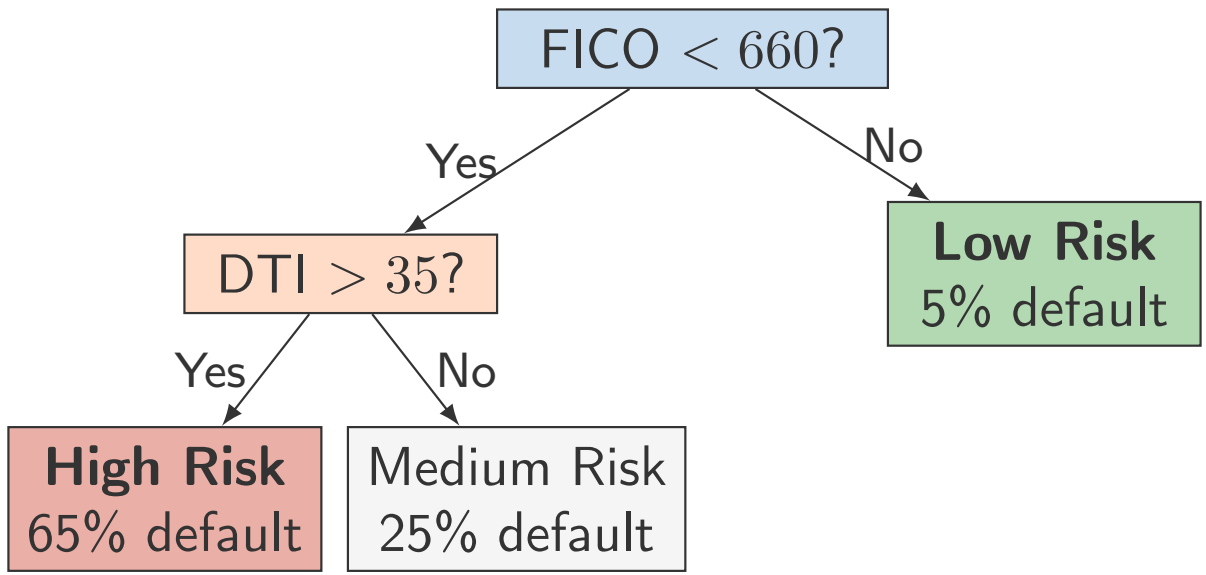
- Linear decision boundary — may miss complex patterns
- Cannot automatically capture interactions between variables

# Classification Trees

# Trees for Classification: The Intuition

**Recall from Lecture 3:** Trees partition the data into regions

**For regression:** Predict the average outcome in each region
**For classification:** Predict the most common class (or class proportions)




**The tree discovers that:** Low FICO + high DTI = highest risk combination

# How Trees Make Splits: Measuring "Purity"

**Goal:** Each split should create more "pure" groups

**Pure node:** All observations belong to same class (all default or all no-default)

**How do we measure impurity?** The **Gini index**

**Intuition:** If we randomly picked two loans from a region, how often would they have different outcomes?

- 50% defaults, 50% non-defaults: **High impurity** (Gini = 0.50)
- 90% defaults, 10% non-defaults: **Low impurity** (Gini = 0.18)
- 100% defaults: **Pure** (Gini = 0)

**The algorithm:** Find the split that reduces impurity the most

*Technical note: "Entropy" is another common measure — same idea, slightly different formula*

# Classification Trees: Probability Outputs

**Trees can give us probabilities, not just class labels**

**How?** Look at the proportion of each class in the leaf node

**Leaf Node**

Contains 100 loans:

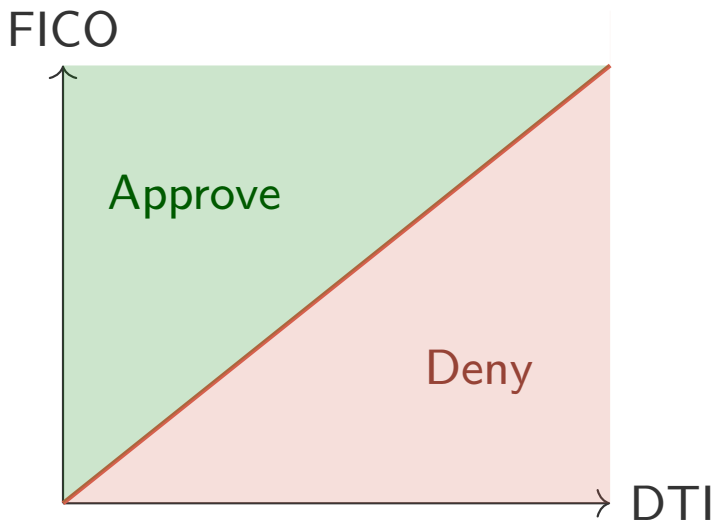
25 defaults, 75 non-defaults

$$\longrightarrow P(\text{default}) = \frac{25}{100} = 0.25$$

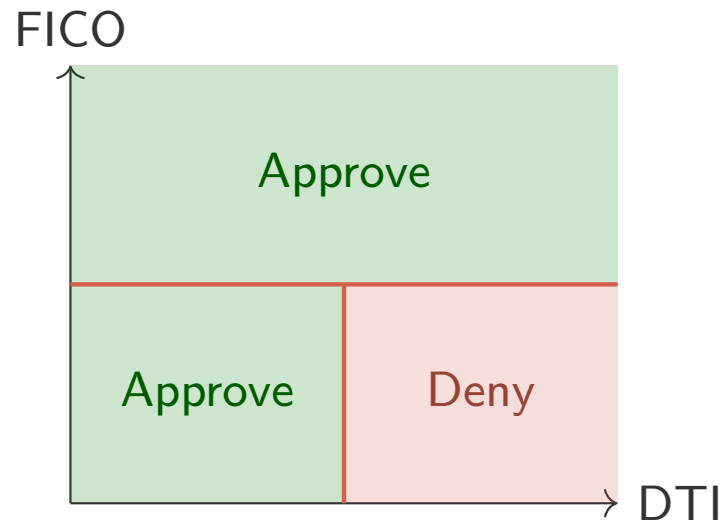
**Limitation:** Single trees produce "coarse" probabilities

- Only as many unique probability values as there are leaf nodes
- A tree with 10 leaves can only output 10 different probabilities
- Ensemble methods (next!) give smoother probability estimates

# Trees vs. Logistic Regression: What's Different?




**Logistic Regression**




**Classification Tree**

**Logistic Regression:**

- Single straight-line boundary
- Same trade-off everywhere: higher FICO always compensates for higher DTI

**Classification Tree:**

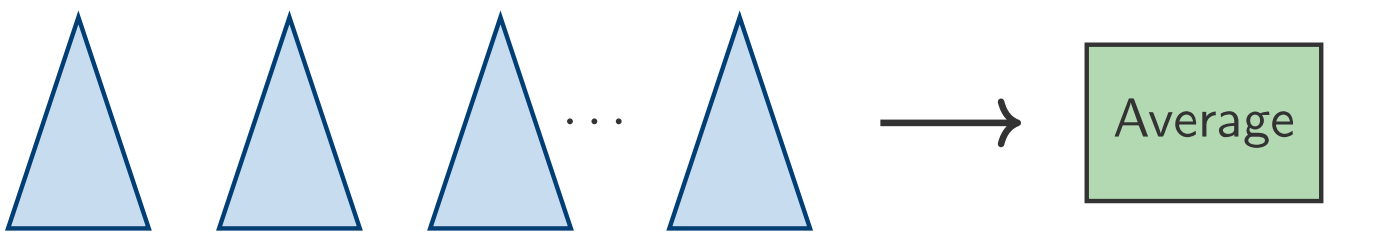
- Axis-aligned splits (rectangles)
- Can capture **interactions**: high DTI only matters for default when FICO is low

# The Problem with Single Trees

**Recall from Lecture 3:** Single trees are **unstable**

- Small changes in data can produce completely different trees
- High variance in predictions
- Coarse probability estimates (limited unique values)

**The solution:** Build many trees and combine their predictions




Many diverse trees Stable prediction

**Same ensemble ideas from Lecture 3 apply to classification!**

# Random Forests for Classification

**Algorithm** (same as regression, different output):

1. Create $B$ bootstrap samples (random samples with replacement)
2. Grow a tree on each sample, using random subset of features at each split
3. Combine predictions from all trees

**How to combine for classification:**

**For class prediction:** Majority vote

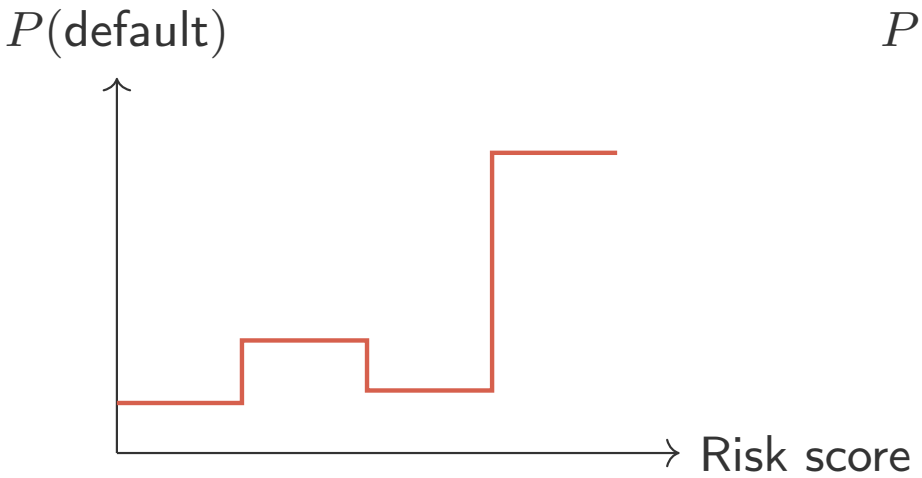
- If 300 trees say "default" and 200 say "no default" $\Rightarrow$ predict "default"

**For probability:** Average the probabilities

$$P(\text{default}) = \frac{1}{B}\sum_{b=1}^{B} P_b(\text{default})$$

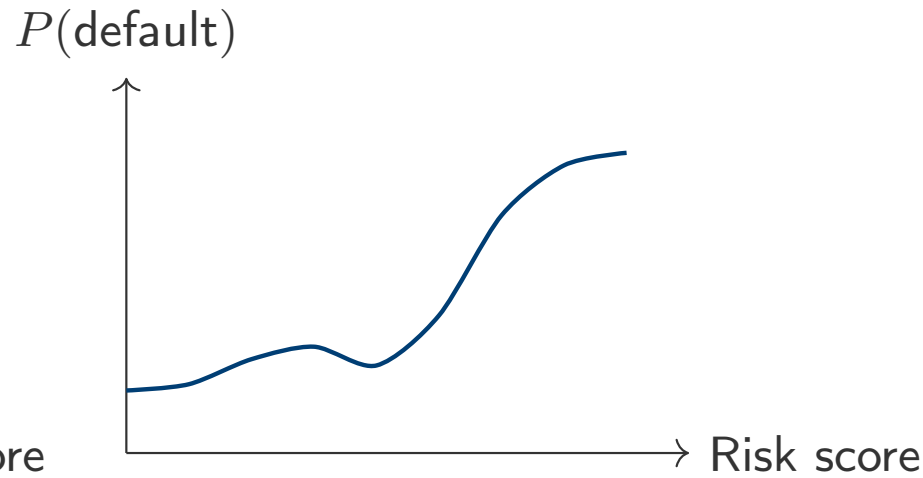
**Key benefit:** Averaging produces **smoother** probability estimates

# Why Averaging Helps: Smoother Probabilities




**Single Tree**

Only 4 probability values
(one for each final leaf)




**Random Forest**

Continuous probabilities
(due to averaging)

**Why this matters for credit risk:**

- Better differentiation between borrowers
- More accurate risk-based pricing
- Smoother ranking of applicants by risk level

# Gradient Boosting for Classification

**Same sequential idea as Lecture 3:**

1. Start with a simple prediction
2. Build trees that focus on **mistakes** from previous trees
3. Combine with shrinkage (learning slowly)

**For classification:** Trees try to improve probability estimates

- Focus on loans where current probability is most wrong
- Gradually refine the predictions

**Key difference from Random Forest:**

- RF: Trees are independent (parallel)
- Boosting: Each tree depends on previous ones (sequential)
- Boosting often achieves better accuracy, but needs more tuning

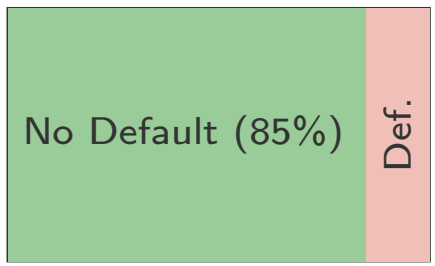
# Handling Class Imbalance

**The problem:** If only 15% of loans default...

- Model can achieve 85% accuracy by always predicting "no default"
- But this is useless for identifying risky borrowers!

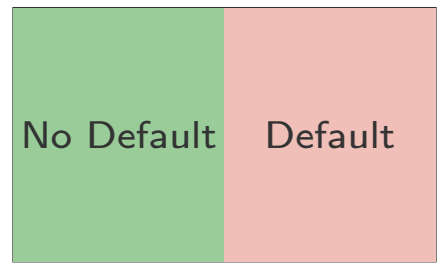
**The intuition:** Make errors on defaults more "costly" during training

**Unweighted**




Equal importance

**Weighted**




Defaults upweighted

**In sklearn:**

- `class_weight='balanced'` — adjusts weights inversely to class frequency.
- **Example:** 85% No Default, 15% Default.
- Weights: No Default $\approx 0.59$, Default $\approx 3.33$.
- Each default loan counts ~5.6x more in the loss.

# The LendingClub Application

# Recap: What We Learned in Part I

**Classification methods for credit risk:**

- **Logistic regression:** Interpretable, linear decision boundary
- **Classification trees:** Automatic interactions, rectangular regions
- **Random Forests:** Variance reduction, smoother probabilities
- **Gradient Boosting:** Often best accuracy, needs tuning

**All methods give us:**

1. Class predictions (default / no default)
2. Probability estimates: $\widehat{P}(\text{default} \mid X)$

## Today's Questions

We've trained our models. Now what? How do we **evaluate** them and turn predictions into **business decisions**?

# The LendingClub Dataset

**Our application:** Peer-to-peer lending platform

**Features available:**

- Loan amount, interest rate
- Annual income
- Employment length
- Debt-to-income ratio
- Credit history length
- Number of open accounts
- Revolving balance & utilization
- Past delinquencies
- Home ownership, loan purpose

**Target:**

- Default (1) vs. No Default (0)

**Sample:**

- $\sim$1.2mln loans
- Default rate: $\sim$20%

**Business goal:**

- Decide: approve or deny?
- Set appropriate interest rate

# We've Trained Three Models

**Setup:**

- 70% training / 15% validation / 15% test split
- Used `class_weight='balanced'` to handle imbalance
- Tuned hyperparameters on validation set

**Models:**

1. **Logistic Regression** (L2 regularized) — our interpretable baseline
2. **Random Forest** (500 trees, max_depth=10)
3. **XGBoost** (with early stopping)

**Now we need to answer:**

- Which model is "best"?
- How do we measure "best"?
- What threshold should we use for decisions?

# The Business Context

**Why do we care about these predictions?**

**Decisions that depend on default probabilities:**

1. **Approve or deny** the loan application
2. **Set the interest rate** (higher risk $\Rightarrow$ higher rate)
3. **Determine loan limits**
4. **Calculate expected losses** for the portfolio
5. **Regulatory capital** requirements

**This means we need:**

- A model that **ranks** borrowers correctly (separates good from bad)
- A way to choose the right **decision threshold**
- **Well-calibrated** probabilities (if using for pricing)

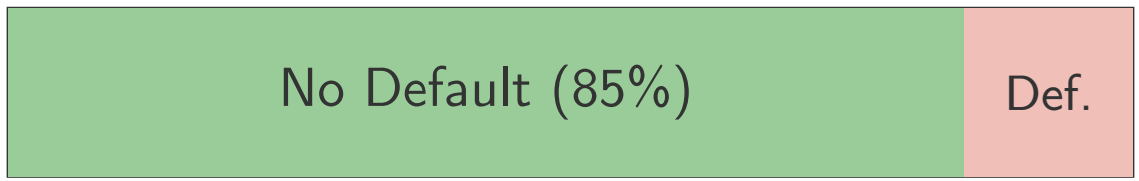
# Model Evaluation

# Why Accuracy Is Misleading

**Accuracy:** Fraction of correct predictions

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total predictions}}$$

**The problem with imbalanced data:**




**"Naive" model: Always predict No Default**

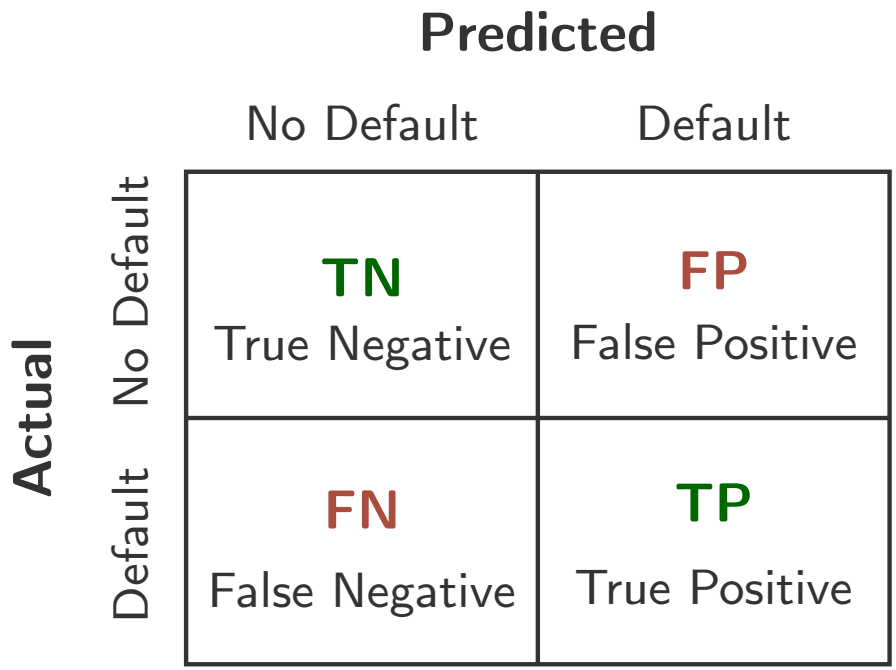
Accuracy = 85% — looks great!

But catches 0% of defaults — completely useless!

**Lesson:** With imbalanced classes, we need better metrics

# The Confusion Matrix

**Foundation:** Count predictions by actual vs. predicted class




**Credit risk interpretation:**

- **FP** (False Positive): Denied a good borrower → lost business
- **FN** (False Negative): Approved a defaulter → lost money

# The Confusion Matrix: Our LendingClub Results

**Logistic regression predictions** (+188k test loans, threshold = 0.5):

| | | **Predicted** | |
|---|---|---|---|
| | | No Default | Default |
| **Actual** | No Default | **95,499** True Negative | **55,264** False Positives |
| | Default | **12,481** False Negative | **25,388** True Positive |

**Reading this:**

- **FP = 55,264**: Good borrowers we would have wrongly denied (lost business)
- **FN = 12,481**: Defaulters we would have approved (lost money!)

# Precision: How Reliable Are Our "Deny" Decisions?

**Precision:** Of those we predict as default, how many actually default?

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Correctly predicted defaults}}{\text{All predicted defaults}}$$

**From our Logistic regression confusion matrix:**

$$\text{Precision} = \frac{25,388}{25,388 + 55,264} = \frac{25,388}{80,652} = 31.5\%$$

**Interpretation:**

- We flagged 80,652 loans as high-risk
- Only 31.5% of them actually defaulted
- The other 68.5% were **good borrowers we would have turned away**

**When precision matters:** When denying good borrowers is costly (lost profit)

# Recall: How Many Defaults Did We Catch?

**Recall:** Of all actual defaults, what fraction did we identify?

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Correctly predicted defaults}}{\text{All actual defaults}}$$

**From our Logistic regression confusion matrix:**

$$\text{Recall} = \frac{25,388}{25,388 + 12,481} = \frac{25,388}{37,869} = 67.1\%$$

**Interpretation:**

- There were 37,869 loans that actually defaulted
- We correctly identified 67.1% of them
- But **32.9% of defaults slipped through** — we approved them!

**When recall matters:** When approving defaulters is costly (lost principal)

# The Precision-Recall Trade-off

**You can't maximize both simultaneously!**

| **Lower threshold** | | **Higher threshold** |
|---|---|---|
| Predict "default" more often | | Predict "default" rarely |
| High Recall | $\longleftrightarrow$ | Low Recall |
| Low Precision | | High Precision |

**The core tension:**

- Lower threshold $\Rightarrow$ catch more defaults, but also deny more good borrowers
- Higher threshold $\Rightarrow$ fewer false alarms, but miss more defaults

**Which matters more?** Depends on the business costs!

# F1 Score: Balancing Precision and Recall

**Problem:** How do we compare models when one has higher precision but lower recall?

**F1 Score:** The harmonic mean of precision and recall

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

**Our Logistic regression model:**

$$F_1 = 2 \times \frac{0.315 \times 0.671}{0.315 + 0.671} = 2 \times \frac{0.211}{0.985} = 0.428$$

**Why harmonic mean?**

- Penalizes extreme imbalances
- If either precision or recall is very low, F1 will be low
- Requires *both* to be good to score well

# Comparing Our Models: Precision, Recall, F1

**Results at threshold = 0.5:**

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| Logistic Regression | 0.315 | 0.670 | 0.428 | 0.641 |
| Random Forest | 0.307 | 0.686 | 0.424 | 0.623 |
| XGBoost | 0.323 | 0.676 | 0.437 | 0.651 |

**Observations:**

- XGBoost has the best F1 score (0.437)
- All models catch only 67–68% of defaults (recall)
- About 30% of "deny" decisions would be wrong (precision $\approx 0.30$)

**Question:** Can we compare models *across all thresholds*?

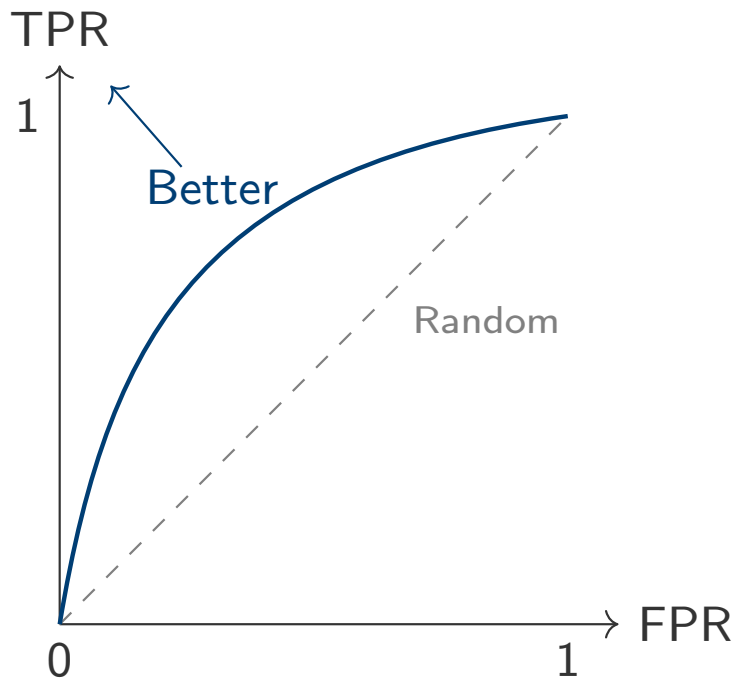
# ROC Curve: Evaluating Across All Thresholds

**Idea:** Plot the trade-off as we vary the threshold from 0 to 1

**Definitions:**

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$



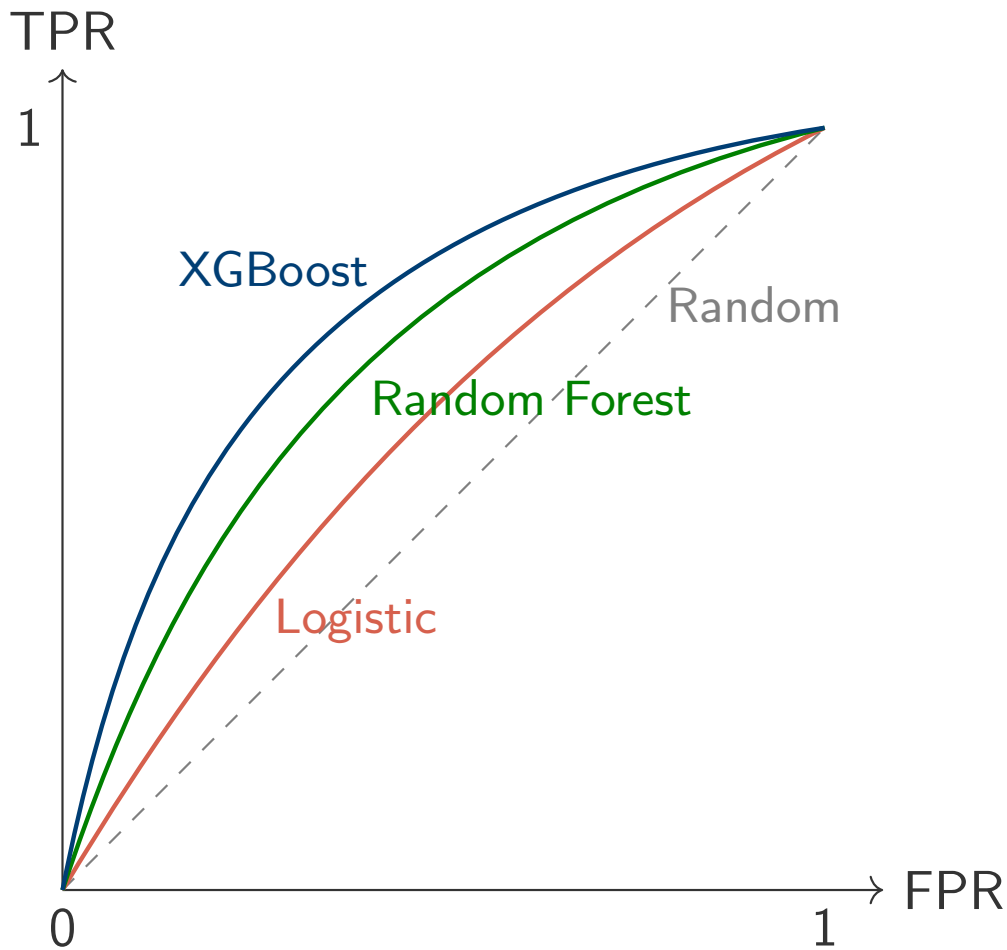

**Interpretation:**

- TPR: Fraction of defaults we catch
- FPR: Fraction of good loans we wrongly flag

**What makes a good ROC curve?**

- Curves toward the **top-left corner** (high TPR, low FPR)
- Diagonal line = random guessing (no predictive power)

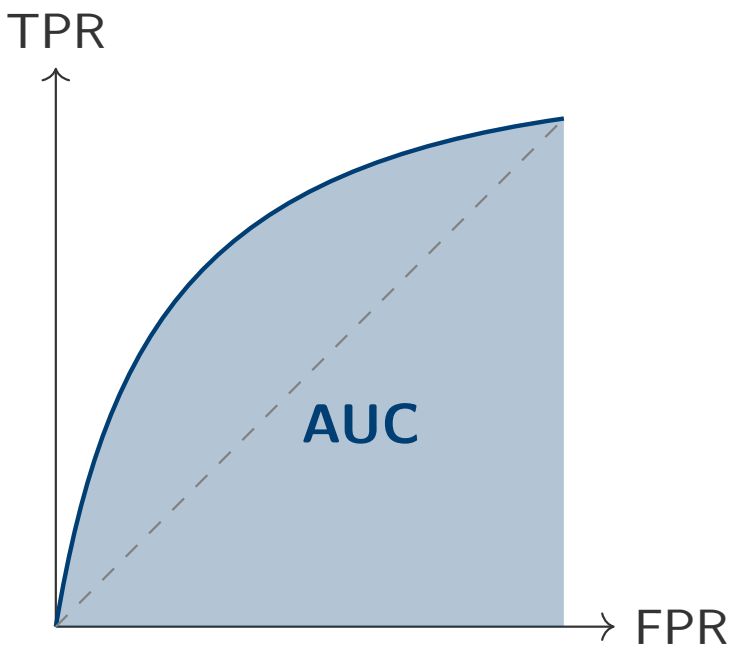
# Our Models: ROC Curves




**Reading this:** XGBoost dominates at almost every threshold

# AUC: A Single Number Summary

**AUC** = Area Under the ROC Curve




**Interpretation:**

- AUC = 0.5: Random guessing
- AUC = 1.0: Perfect
- AUC = 0.73: Good (our XGBoost)

**Intuitive meaning:**
If we randomly pick one defaulter and one non-defaulter from our test set, there's a 73% chance the model correctly ranks the defaulter as riskier.

**LendingClub results:**

| | |
|---|---|
| Logistic Regression | AUC = 0.709 |
| Random Forest | AUC = 0.707 |
| XGBoost | AUC = 0.722 |

# Which Metric Should We Use?

| Situation | Focus on |
|---|---|
| Comparing models (threshold not yet chosen) | **AUC** |
| Limited capital, can't afford bad loans | **Precision** |
| Must identify risky borrowers (regulatory) | **Recall** |
| Need balance between both errors | **F1 Score** |

## Key Insight

There's no single "best" metric — it depends on the **costs of different errors**

# Threshold Optimization

# Why 0.5 Is Usually Wrong

**Default threshold:** Predict "default" if $P(\text{default}) > 0.5$

**This assumes:**

- Cost of denying a good borrower = Cost of approving a defaulter

**But in reality:**

| Error Type | Typical Cost |
|---|---|
| False Positive (deny good borrower) | Lost profit: £500 |
| False Negative (approve defaulter) | Lost principal: £5,000 |

**Defaults are 10× more costly!**

**Implication:** We should use a **lower threshold** to be more cautious

# Cost-Sensitive Decision Making

**Framework:** Choose threshold to minimize expected cost

**Define:**

- $C_{FP} =$ £500 (cost of denying a good borrower)
- $C_{FN} =$ £5,000 (cost of approving a defaulter)

**Total expected cost:**

$$\text{Total Cost} = C_{FP} \times \#FP + C_{FN} \times \#FN$$

**Trade-off:**

- Lower threshold $\Rightarrow$ more FP (deny good borrowers), fewer FN (catch more defaults)
- Higher threshold $\Rightarrow$ fewer FP, more FN

**Goal:** Find the threshold that minimizes total cost

# Finding the Optimal Threshold

**Our XGBoost model at different thresholds:**

| Threshold | FP | FN | Total Cost | |
|---|---|---|---|---|
| 0.50 | 53,588 | 12,263 | £72.0m | |
| 0.40 | 81,325 | 6,367 | £48.1m | |
| 0.30 | 108,799 | 2,677 | £35.1m | |
| 0.20 | 131,401 | 735 | £30.0m | |
| 0.13 | 142,720 | 193 | £29.5m | ← **Optimal** |
| 0.10 | 146,488 | 84 | £29.7m | |

**Key insight:** Optimal threshold is **0.13**, not 0.5!

At $\tau = 0.5$ we miss 12,263 defaults; at $\tau = 0.13$ we miss only 193. The extra denied good borrowers (FP) are worth it.

# Metrics at Optimal Threshold

**XGBoost at threshold = 0.13:**

| **Metric** | **At $\tau = 0.5$** | **At $\tau = 0.13$** |
|---|---|---|
| Recall | 0.68 | 0.99 |
| False Negatives | 12,263 | 193 |
| False Positives | 53,588 | 142,720 |
| Total Cost | £72.0m | £29.5m |

**Interpretation:**

- At $\tau = 0.13$: We catch **99%** of defaults (vs. 68% at 0.5)
- Trade-off: Many more good borrowers denied (142k vs. 54k)
- But total cost drops by **60%** because we avoid expensive defaults

**The business outcome is dramatically different.**

# From Models to Business Decisions

# From Model to Business Decisions

**We now have:**

- A trained model (XGBoost) with predicted probabilities
- An optimal threshold ($\tau = 0.13$) based on costs

1. **Approval decisions:**

| **Predicted Probability** | **Decision** |
|---|---|
| $P(\text{default}) < 0.08$ | Approve at standard rate |
| $0.08 \leq P(\text{default}) < 0.13$ | Approve at higher rate |
| $P(\text{default}) \geq 0.13$ | Deny (or require collateral) |

2. **Risk-based pricing:** Higher $P(\text{default}) \Rightarrow$ higher interest rate
3. **Portfolio expected loss:**

$$\text{Expected Loss} = \sum_i P(\text{default}_i) \times \text{Loan Amount}_i \times \text{LGD}$$

(LGD = Loss Given Default, typically 40–60% of loan value)

# Practical Considerations

**Model choice involves trade-offs:**

| | Logistic Regression | XGBoost |
|---|---|---|
| Predictive accuracy | Lower | Higher |
| Interpretability | High | Low |
| Regulatory acceptance | Easy | Harder |
| Calibration | Usually good | Needs checking |

**Other considerations:**

- **Regulatory:** Models must be explainable (SR 11-7 in US)
- **Fairness:** Cannot use protected characteristics
- **Monitoring:** Performance degrades over time — retrain periodically
- **Reject inference:** We only observe outcomes for approved loans

# Summary and Next Steps

# Key Takeaways from Today

1. Evaluation:
   - Accuracy is misleading with imbalanced data
   - Use precision, recall, F1, and AUC depending on business needs
   - ROC/AUC measures ranking ability across all thresholds
2. Threshold optimization:
   - Default threshold of 0.5 is rarely optimal
   - Choose threshold based on costs of different errors
3. Application:
   - XGBoost often wins on AUC, but logistic is more interpretable
   - Important: predictions are instrumental for pricing, approval, and portfolio management

# Readings and Next Steps

**Readings for Today's Lecture:**

- James et al. (2023), Chapter 4.4–4.5 *[Recomm]*
- Khandani, Kim & Lo (2010) *[Recomm]*
- Hand & Henley (1997), "Statistical Classification Methods in Consumer Credit Scoring" *[Supp.]*

**Coming up in Week 5:**

- Unsupervised learning
  - Principal Component Analysis (PCA)
  - Clustering methods
- Applications on market timing and factor investing.

Questions?