Big Data in Finance

Lecture 4: Classification Methods

Dr Daniele Bianchi

Spring 2026

Contents

Overview 1

1 From Regression to Classification 1
1.1 Why Can't We Just Use Regression? . . . 2

2 Logistic Regression 2
2.1 A Concrete Example . . . 2
2.2 Interpreting Coefficients: Odds Ratios . . . 3
2.3 Estimation and Regularization . . . 3
2.4 The Decision Boundary . . . 3

3 Classification Trees 4
3.1 Splitting Criterion: Gini Index . . . 4
3.2 What Classification Trees Discover . . . 4
3.3 Decision Boundaries . . . 5

4 Ensemble Methods for Classification 5
4.1 Random Forests . . . 5
4.2 Gradient Boosting . . . 5
4.3 Handling Class Imbalance . . . 6

5 The LendingClub Application 6
5.1 Models Trained . . . 6

6 Evaluation Metrics 6
6.1 The Confusion Matrix . . . 7
6.2 Precision: When We Predict Default, Are We Right? . . . 7
6.3 Recall: How Many Defaults Do We Catch? . . . 7
6.4 The Precision-Recall Tradeoff . . . 7
6.5 F1 Score: Balancing Precision and Recall . . . 8
6.6 Comparing Models at Threshold 0.5 . . . 8
6.7 ROC Curves and AUC . . . 8
6.8 LendingClub AUC Results . . . 9
6.9 Which Metric to Use? . . . 9

7 Threshold Optimization 9
7.1 Cost-Sensitive Decision Making . . . 9

8 From Models to Business Decisions 10
8.1 Practical Considerations . . . 10

9 Summary and Key Takeaways 11

Readings **11**

Overview

In Lectures 2 and 3, we studied methods for predicting continuous outcomes: what will next month's market return be? The output was a number—a predicted return of +2.3% or −1.1%. But many important questions in finance ask for something different: not "how much?" but "which category?"

Will this borrower default or repay? Is this transaction fraudulent or legitimate? Will the market go up or down? These are **classification** problems, and they require different tools than regression.

Sections 1 to 4 introduce classification methods, with credit risk as our running application. We begin with logistic regression—the workhorse of credit scoring for decades—which models the probability of default as a function of borrower characteristics. We then extend the tree-based methods from Lecture 3 to classification, showing how Random Forests and Gradient Boosting can capture complex patterns in default risk.

But building a model is only half the story. Sections 5 to 8 address the questions that matter for business decisions: How do we *evaluate* classification models when accuracy is misleading? How do we choose the right threshold for approve/deny decisions? How do we translate predicted probabilities into interest rates and risk assessments?

The LendingClub dataset—over one million peer-to-peer loans with detailed borrower characteristics and default outcomes—provides a concrete setting for these questions. By the end of this lecture, you will understand not just how to build credit risk models, but how to evaluate them and turn their predictions into economically sound decisions.

1 From Regression to Classification

In regression, we predict a continuous outcome $y \in \mathbb{R}$. In classification, we predict a categorical outcome $y \in \{0, 1\}$ (for binary classification) or $y \in \{1, 2, \ldots, K\}$ (for multi-class). The output is a *category*, not a number.

Credit risk is the canonical example. Given a loan applicant's characteristics—income, credit score, debt-to-income ratio, employment history—we want to predict whether they will default or repay. The outcome is binary: default (1) or no default (0).

But in practice, we usually want more than just a binary prediction. We want the **probability** of default: $P(\text{default} \mid \text{borrower characteristics})$. This probability drives critical business decisions:

Pricing: A borrower with 5% default probability should pay a different interest rate than one with 25% probability. Without probability estimates, we cannot do risk-based pricing.

Capital requirements: Regulators require banks to hold capital against expected losses. This requires accurate probability estimates across the portfolio.

Portfolio management: Expected loss on a loan portfolio is $\sum_i P(\text{default}_i) \times \text{Loan Amount}_i \times \text{LGD}$, where LGD is loss given default.

So our goal in classification is typically twofold: predict the class label, and estimate the probability of each class.

1.1 Why Can't We Just Use Regression?

A naive approach might be to encode default as 1 and non-default as 0, then run a linear regression. The predicted value $\widehat{y}$ would then be interpreted as $P(\text{default})$.

This fails for a simple reason: linear regression can predict values outside $[0, 1]$. If a borrower with a FICO score of 800 has a predicted default probability of -0.15, what does that mean? Probabilities must be between 0 and 1.

The solution is to use methods designed specifically for classification—methods that guarantee valid probability outputs.

2 Logistic Regression

Logistic regression elegantly addresses the probability problem. Instead of predicting y directly, it models the **log-odds** of the outcome as a linear function:

$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

The left side is the log-odds ratio. The right side is a familiar linear combination. By solving for $P(y=1)$, we get:

$$P(y=1 \mid x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_p x_p)}} = \frac{e^{\beta_0+\beta' x}}{1+e^{\beta_0+\beta' x}}$$

This is the **sigmoid** (or logistic) function. It takes any real number as input and outputs a value between 0 and 1. The linear combination $\beta_0 + \beta' x$ can range from $-\infty$ to $+\infty$, but the sigmoid squashes it into a valid probability.

2.1 A Concrete Example

Suppose our logistic regression model learns these coefficients for predicting loan default:

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

For a low-risk borrower with DTI = 25% and FICO = 750:

$$\begin{aligned}
\text{Linear combination} &= 8 + 0.05 \times 25 + (-0.015) \times 750 = 8 + 1.25 - 11.25 = -2.0 \\
P(\text{default}) &= \frac{1}{1+e^{-(-2.0)}} = \frac{1}{1+e^{2}} \approx 0.12
\end{aligned}$$

For a high-risk borrower with DTI = 45% and FICO = 620:

$$\begin{aligned}
\text{Linear combination} &= 8 + 0.05 \times 45 + (-0.015) \times 620 = 8 + 2.25 - 9.30 = 0.95 \\
P(\text{default}) &= \frac{1}{1+e^{-0.95}} \approx 0.72
\end{aligned}$$

The model assigns a 12% default probability to the good borrower and 72% to the risky one.

2.2 Interpreting Coefficients: Odds Ratios

Logistic regression coefficients have a natural interpretation in terms of **odds ratios**. The odds of default are:

$$\text{Odds} = \frac{P(\text{default})}{1 - P(\text{default})}$$

For every one-unit increase in predictor x_j, the odds multiply by e^{β_j}. If $\beta_{\text{DTI}} = 0.05$, then $e^{0.05} \approx 1.05$, meaning each 1 percentage point increase in debt-to-income ratio increases the odds of default by about 5%.

This interpretability is crucial in credit risk. Regulators require explanations for loan decisions. A loan officer must be able to tell a denied applicant: "Your application was declined primarily because your debt-to-income ratio exceeds our threshold." Logistic regression makes such explanations possible.

2.3 Estimation and Regularization

Unlike OLS, which minimizes squared errors, logistic regression is estimated by **maximum likelihood**. The idea is intuitive: find the coefficients that maximize the probability of the observed outcomes.

For borrowers who actually defaulted, the model should assign a high probability to default. For those who repaid, it should assign a low probability. Maximum likelihood finds the β values that maximize this joint probability.

There is no closed-form solution. The computer finds optimal coefficients iteratively using numerical optimization (gradient descent or Newton-Raphson). In practice, you call `LogisticRegression()` in sklearn, and the optimization happens automatically.

With many predictors, logistic regression can overfit just like linear regression. The solution is the same: add penalty terms. L2 regularization (Ridge) shrinks coefficients toward zero. L1 regularization (Lasso) sets some coefficients exactly to zero, performing variable selection. Elastic Net combines both.

In sklearn, `LogisticRegression` uses L2 penalty by default, with the regularization strength controlled by the parameter C (where smaller C means stronger regularization, the inverse of λ).

2.4 The Decision Boundary

To convert probabilities to predictions, we need a **threshold**. The simplest rule is: predict "default" if $P(\text{default}) > 0.5$. This creates a **decision boundary** in feature space—the set of points where $P(\text{default}) = 0.5$.

For logistic regression, this boundary is linear: it is a hyperplane in feature space. On one side, we predict default; on the other, no default. The boundary is where the linear combination $\beta_0 + \beta' x = 0$.

This linearity is both a strength (simplicity, interpretability) and a limitation (it cannot capture complex nonlinear patterns). If the true boundary between defaults and non-defaults is curved or irregular, logistic regression will approximate it with a straight line.

3 Classification Trees

In Lecture 3, we built regression trees that partition feature space into regions and predict the *mean* outcome within each region. Classification trees work similarly, but instead of predicting

means, they predict *classes* or *class proportions*.

For a classification tree:

- Each leaf node contains a set of training observations
- The predicted class is the *majority class* in that leaf
- The predicted probability is the *proportion* of each class in the leaf

For example, if a leaf contains 100 loans with 25 defaults and 75 non-defaults, the predicted class is "no default" (majority) and the predicted probability is $P(\text{default}) = 25/100 = 0.25$.

3.1 Splitting Criterion: Gini Index

Regression trees split to minimize RSS (variance within regions). Classification trees split to maximize **purity**—the extent to which each region contains only one class.

The most common measure is the **Gini index**:

$$G = \sum_{k=1}^{K} \widehat{p}_k(1 - \widehat{p}_k)$$

where $\widehat{p}_k$ is the proportion of class k in the node. For binary classification:

$$G = 2\widehat{p}(1 - \widehat{p})$$

The Gini index measures the probability that two randomly chosen observations from the node belong to different classes. A pure node (all one class) has $G = 0$. Maximum impurity (50-50 split) has $G = 0.5$.

At each split, the tree algorithm finds the variable and threshold that most reduce the weighted average Gini index of the children. This is analogous to minimizing RSS in regression trees.

3.2 What Classification Trees Discover

Consider a simple credit risk tree:

- First split: FICO < 660?
  - Yes $\rightarrow$ check DTI
    - DTI $> 35\%$ $\rightarrow$ High Risk (65% default)
    - DTI $\leq 35\%$ $\rightarrow$ Medium Risk (25% default)
  - No $\rightarrow$ Low Risk (5% default)

The tree has discovered an **interaction**: debt-to-income ratio only matters for borrowers with low credit scores. For high-FICO borrowers, DTI is irrelevant—they are low risk regardless. This is exactly the kind of pattern that logistic regression (with its linear decision boundary) cannot capture without manually specifying the interaction term.

3.3 Decision Boundaries

The decision boundaries reveal a fundamental difference:

Logistic regression: Single linear boundary. The tradeoff between FICO and DTI is the same everywhere—higher FICO always compensates for higher DTI at the same rate.

Classification tree: Axis-aligned rectangular boundaries. The tree can say "DTI matters a lot when FICO is low, but not at all when FICO is high"—a nonlinear, context-dependent rule.

4 Ensemble Methods for Classification

4.1 Random Forests

Random Forests address both the instability of single trees and the coarseness of their probability estimates. The algorithm is the same as for regression:

1. Create B bootstrap samples
2. Grow a tree on each sample, using random subset of features at each split
3. Combine predictions across all trees

For classification, combining works as follows:

Class prediction: Majority vote. If 300 trees predict "default" and 200 predict "no default," the ensemble predicts "default."

Probability prediction: Average the predicted probabilities:

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

This averaging produces **smoother** probability estimates than single trees. Even if individual trees output coarse probabilities (0.1, 0.2, 0.3, ...), the average of 500 trees can produce nearly continuous values. This is crucial for risk-based pricing.

4.2 Gradient Boosting

Gradient Boosting extends naturally to classification. Instead of fitting residuals (as in regression), it fits trees to the gradient of the log-loss (cross-entropy). The intuition is similar: each tree focuses on observations where the current model is making mistakes.

For binary classification:

1. Initialize with log-odds of the base rate: $F_0 = \log(p_1/(1-p_1))$ where p_1 is the proportion of class 1
2. For each iteration:
   - Compute pseudo-residuals: how wrong is the current probability estimate?
   - Fit a tree to these residuals
   - Add the tree to the ensemble with shrinkage
3. Convert final log-odds to probability via sigmoid

Gradient Boosting often achieves the best predictive accuracy among classification methods, but requires more careful tuning than Random Forests.

4.3 Handling Class Imbalance

Credit risk data is typically **imbalanced**: defaults are relatively rare. If only 15% of loans default, a model that always predicts "no default" achieves 85% accuracy—but is completely useless for identifying risky borrowers.

The solution is to reweight the classes during training. In sklearn, `class_weight='balanced'` adjusts weights inversely proportional to class frequency. If 85% of loans are non-defaults and

15% are defaults, the weights become approximately 0.59 for non-defaults and 3.33 for defaults. Each default observation counts about 5.6 times as much in the loss function.

This forces the model to take defaults seriously even though they are rare. The tradeoff is that the model will make more false positives (flagging good loans as risky), but it will catch more true defaults.

5 The LendingClub Application

LendingClub was a peer-to-peer lending platform that published detailed data on its loans. Our dataset contains approximately 1.2 million loans with rich borrower characteristics:

- Loan amount and interest rate
- Annual income and employment length
- Debt-to-income ratio
- Credit history length and number of accounts
- Revolving balance and utilization
- Past delinquencies and public records
- Home ownership and loan purpose

The target variable is binary: default (1) vs. no default (0). The default rate is approximately 20%.

5.1 Models Trained

We train three models on a 70% training / 15% validation / 15% test split, using `class_weight='balanced'` to handle imbalance:

1. **Logistic Regression** with L2 regularization—our interpretable baseline
2. **Random Forest** with 500 trees and maximum depth of 10
3. **XGBoost** with early stopping based on validation performance

All models output predicted probabilities $\widehat{P}(\text{default} \mid X)$.

6 Evaluation Metrics

Accuracy is the most intuitive metric: what fraction of predictions are correct?

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

With imbalanced data, accuracy is misleading. If 80% of loans are non-defaults, a model that always predicts "no default" achieves 80% accuracy. This looks impressive but is completely useless—it identifies zero defaults.

We need metrics that separately assess performance on each class.

6.1 The Confusion Matrix

The **confusion matrix** is the foundation of classification evaluation. It cross-tabulates predicted versus actual classes:

	Predicted: No Default	**Predicted: Default**
Actual: No Default	True Negative (TN)	False Positive (FP)
Actual: Default	False Negative (FN)	True Positive (TP)

In credit risk terms:

- **FP (False Positive):** Good borrower flagged as risky → lost business
- **FN (False Negative):** Defaulter approved as safe → lost money

For our Logistic Regression model at threshold 0.5:

	Pred: No Default	**Pred: Default**
Actual: No Default	95,499 (TN)	55,264 (FP)
Actual: Default	12,481 (FN)	25,388 (TP)

6.2 Precision: When We Predict Default, Are We Right?

Precision answers: of those we predict as defaults, what fraction actually default?

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

We flagged 80,652 loans as high-risk, but only 31.5% actually defaulted. The other 68.5% were good borrowers we would have turned away.

When precision matters: When denying good borrowers is costly. High precision means our "deny" decisions are reliable—when we say someone is risky, they usually are.

6.3 Recall: How Many Defaults Do We Catch?

Recall (also called sensitivity or true positive rate) answers: of all actual defaults, what fraction did we identify?

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

Of 37,869 loans that actually defaulted, we correctly identified 67.1%. But 32.9% of defaults slipped through—we approved them.

When recall matters: When approving defaulters is costly. High recall means we catch most of the bad loans, even if we also flag some good ones.

6.4 The Precision-Recall Tradeoff

You cannot maximize both precision and recall simultaneously. Lowering the classification threshold (predict "default" more often) increases recall—you catch more defaults. But it decreases precision—more of your "default" predictions are wrong.

This is a fundamental tradeoff, determined by the threshold τ:

- Lower τ: Higher recall, lower precision (conservative, catches more defaults)
- Higher τ: Lower recall, higher precision (aggressive, fewer false alarms)

The right balance depends on the business costs of each error type.

6.5 F1 Score: Balancing Precision and Recall

The **F1 score** is the harmonic mean of precision and recall:

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

For our logistic regression: $F_1 = 2 \times (0.315 \times 0.671)/(0.315 + 0.671) = 0.428$.

The harmonic mean penalizes extreme imbalances. If either precision or recall is very low, F1 will be low. This makes F1 useful when you need both to be reasonable.

6.6 Comparing Models 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

XGBoost has the best F1 score, but all models catch only about 67–68% of defaults. About 30% of "deny" decisions would be wrong.

But this comparison is at one particular threshold. Can we compare models across all thresholds?

6.7 ROC Curves and AUC

The **ROC curve** (Receiver Operating Characteristic) plots the tradeoff between True Positive Rate (recall) and False Positive Rate as the threshold varies from 0 to 1:

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} = \text{Recall}$$
$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

TPR is the fraction of defaults we catch. FPR is the fraction of good loans we wrongly flag. The ROC curve shows how these two quantities trade off as we adjust the threshold.

A good model has high TPR and low FPR—its ROC curve hugs the top-left corner. A random model (no predictive power) falls on the diagonal: TPR = FPR for any threshold.

The **AUC** (Area Under the ROC Curve) summarizes the ROC curve in a single number:

- AUC = 0.5: Random guessing
- AUC = 1.0: Perfect separation
- AUC = 0.72: Good discrimination (our XGBoost)

The AUC has an intuitive interpretation: if we randomly pick one defaulter and one non-defaulter, the AUC is the probability that the model ranks the defaulter as riskier. An AUC of 0.72 means we rank correctly 72% of the time.

6.8 LendingClub AUC Results

Model	AUC
Logistic Regression	0.709
Random Forest	0.707
XGBoost	0.722

XGBoost has the best ranking ability. At almost every threshold, it achieves higher TPR for the same FPR (or lower FPR for the same TPR).

6.9 Which Metric to Use?

Situation	Use
Comparing models before choosing threshold	AUC
Limited capital, can't afford bad loans	Precision
Must identify risky borrowers (regulatory)	Recall
Need balance between error types	F1 Score

The key insight is that there is no single "best" metric—it depends on the costs of different errors.

7 Threshold Optimization

The default threshold of 0.5 assumes that the cost of denying a good borrower equals the cost of approving a defaulter. This is almost never true.

In credit risk:

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 than lost business! We should use a lower threshold to be more cautious.

7.1 Cost-Sensitive Decision Making

The optimal threshold minimizes total expected cost:

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

With $C_{FP} = £500$ and $C_{FN} = £5000$, we evaluate 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	

The 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 (142,720 vs. 53,588) are worth it because defaults are so much more costly.

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

At the optimal threshold, we catch 99% of defaults (vs. 68% at 0.5). The tradeoff is many more denied good borrowers (142k vs. 54k). But total cost drops by 60%.

The business outcome is dramatically different depending on threshold choice.

8 From Models to Business Decisions

With probability estimates and an optimal threshold, we can create tiered decision rules:

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)

This is more nuanced than a simple approve/deny. Borderline borrowers can get loans, but at rates that compensate for their risk.

Risk-Based Pricing: If probabilities are well-calibrated, we can price loans to cover expected losses:

$$\text{Rate} = \text{Base Rate} + \text{Risk Premium} = r_f + P(\text{default}) \times \text{LGD} + \text{Margin}$$

A borrower with 10% default probability and 50% LGD needs a risk premium of 5% just to break even on expected losses. Add margin for costs and profit.

Portfolio Expected Loss: For portfolio management and regulatory capital:

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

Accurate probability estimates are essential for this calculation. A model that is systematically overconfident (probabilities too low) will underestimate expected loss and undercapitalize.

8.1 Practical Considerations

Model choice involves tradeoffs:

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

Regulatory requirements: Credit models must be explainable. Regulators (e.g., SR 11-7 in the US) require banks to understand and justify their models. Logistic regression is easier to defend than XGBoost.

Fairness: Models cannot discriminate based on protected characteristics (race, gender, etc.). Even if a model does not explicitly use these features, it may learn proxies that create disparate impact.

Monitoring and retraining: Model performance degrades over time as borrower populations change. Regular monitoring and periodic retraining are essential.

Reject inference: We only observe outcomes for approved loans. If the model denies certain borrowers, we never learn if they would have defaulted. This creates selection bias that requires special techniques to address.

9 Summary and Key Takeaways

This lecture has covered classification methods and their application to credit risk. The key points:

Classification predicts categories, not numbers. But we usually want probabilities, not just class labels. Logistic regression and tree ensembles both provide probability estimates.

Logistic regression is the workhorse of credit scoring. The sigmoid function guarantees valid probabilities. Coefficients have interpretable odds ratio interpretations. The decision boundary is linear.

Classification trees capture interactions automatically. Unlike logistic regression, trees can discover that DTI matters only when FICO is low. But single trees have coarse probabilities and are unstable.

Random Forests produce smoother probability estimates. Averaging many trees creates nearly continuous probabilities suitable for risk-based pricing.

Accuracy is misleading with imbalanced data. Use precision, recall, F1, and AUC depending on business needs. The confusion matrix is the foundation.

The default threshold of 0.5 is rarely optimal. Choose threshold based on the costs of different errors. In credit risk, defaults are typically much more costly than lost business, justifying lower thresholds.

Model choice involves tradeoffs. XGBoost often wins on predictive accuracy, but logistic regression is more interpretable and easier to explain to regulators.

In the next lecture, we turn to unsupervised learning—methods that find structure in data without labeled outcomes. Principal Component Analysis and clustering will help us understand the structure of financial data and reduce dimensionality.

Readings

Required:

- James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning with Applications in Python*, Chapter 4.

Supplementary:

- Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). "Consumer Credit-Risk Models via Machine-Learning Algorithms." *Journal of Banking & Finance*, 34(11), 2767–2787. Application of ML to credit risk with discussion of economic value.
- Hand, D. J., & Henley, W. E. (1997). "Statistical Classification Methods in Consumer Credit Scoring: A Review." *Journal of the Royal Statistical Society: Series A*, 160(3), 523–541. Classic review of credit scoring methods.