L03: P2P Lending & Robo-Advisors

Extended Slides – BSc Digital Finance Course

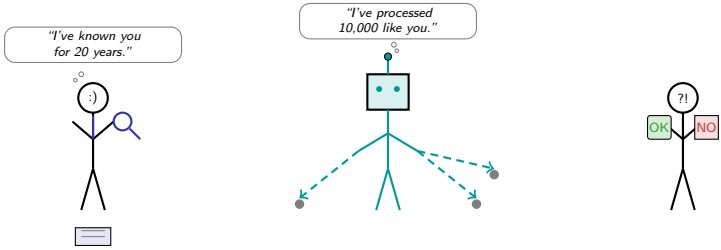
Digital Finance

What Will You Be Able to Do After This Lecture?

1. Formalize the information asymmetry problem in lending markets using principal-agent models
2. Implement a logistic regression credit scoring model in Python and interpret the coefficients
3. Evaluate model performance using ROC curves, AUC, precision, recall, and confusion matrices
4. Construct a mean-variance efficient portfolio using Modern Portfolio Theory
5. Quantify the fairness-accuracy tradeoff using demographic parity and equalized odds
6. Compare robo-advisory portfolio optimization with traditional human advisory

Prerequisites: Linear algebra basics, Python (numpy, pandas, sklearn), probability (Bayes' theorem).

These six objectives span theory (1), implementation (2–3), optimization (4), ethics (5), and application (6).



Same borrower, same data, different verdict. Welcome to data-driven lending.

The transition from relationship banking to algorithmic lending changes WHO decides – but the information problem remains.

How Do Economists Formalize the Lemon Problem?

Adverse Selection (Akerlof 1970)
Let $\theta \in \{L, H\}$ be borrower type (Low risk, High risk). The lender observes a noisy signal s but not θ directly.

Posterior probability:

$$P(\theta = H \mid s) = \frac{P(s \mid \theta = H)\, P(\theta = H)}{P(s)}$$

Market equilibrium: If the market interest rate r^* exceeds the fair rate for low-risk borrowers ($r^* > r_L$), good borrowers exit the market:

$$E[r] = \frac{n_H\, r_H + n_L\, r_L}{n_H + n_L} \xrightarrow{n_L \to 0} r_H$$

Signal quality determines market survival:

Signal Quality	Separating Eq.	Market Outcome
Perfect (s reveals θ)	Yes	Efficient pricing
Moderate (noisy signal)	Partial	Cross-subsidy, some exit
None (no screening)	No	Market unraveling

Intuition

Imagine a used-car market where sellers know the quality but buyers do not. Buyers offer an average price. Owners of good cars refuse to sell at the average price and leave the market. Only "lemons" remain.

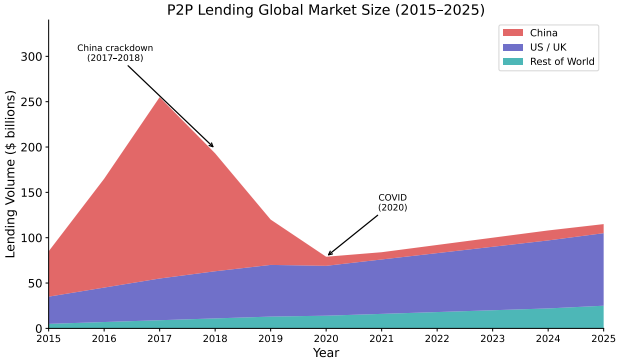
In lending:

- Sellers = borrowers (they know their own risk)
- Buyers = lenders (they observe signals, not truth)
- Lemons = high-risk borrowers who accept high rates
- Good cars = low-risk borrowers who exit when rates rise

P2P platforms attempt to break the cycle by using ML to extract better signals from alternative data – turning a pooling equilibrium into a separating one.

Akerlof's insight: without credible signals, the market converges to the worst-case equilibrium. ML credit scoring is an attempt to provide those signals.

What Is the Business Model of a P2P Lending Platform?



Platform revenue model

- Origination fees: 1–5% of loan amount
- Servicing fees: 0.5–1% annually
- Late payment fees: charged to borrowers

Cost structure

- Technology and ML infrastructure
- Regulatory compliance (licensing, audits)
- Credit losses: provision fund vs. pass-through

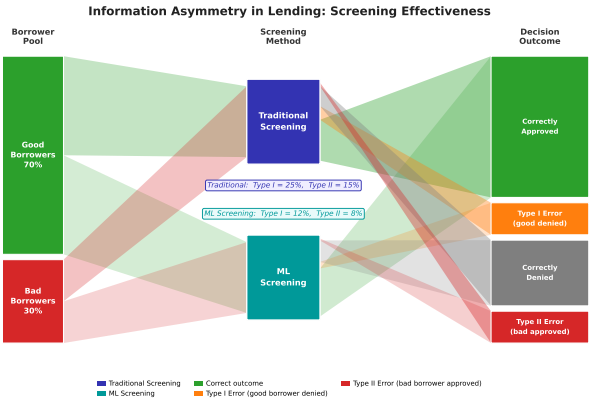
The data flywheel

- More borrowers → better data
- Better data → better models
- Better models → more investors
- More investors → lower rates
- Lower rates → more borrowers

The network effect IS the competitive moat.

P2P platforms are data businesses that happen to process loans – the data flywheel is the moat.

Where Does Information Get Lost Between Borrower and Lender?



Screening effectiveness

Error	Traditional	ML
Type I (false alarm)	15%	8%
Type II (missed bad)	25%	12%

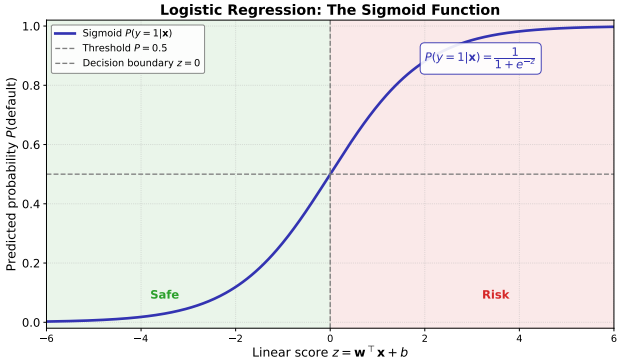
ML screening reduces *total* classification errors but may introduce *systematic* bias through correlated features:

- ZIP code correlates with race
- Job title correlates with gender
- Education correlates with socioeconomic status

Key insight: Fewer errors overall does not mean fewer errors for each group. ML may redistribute mistakes unfairly across demographic categories.

ML reduces total classification errors but may redistribute them unfairly across demographic groups.

How Does Logistic Regression Predict Default Probability?



Illustrative sigmoid function; z combines borrower features weighted by model coefficients

Model

$$P(y{=}1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

Log-odds (logit)

$$\log \frac{P(y{=}1 \mid \mathbf{x})}{1 - P(y{=}1 \mid \mathbf{x})} = \mathbf{w}^T\mathbf{x} + b$$

Each weight w_i: change in log-odds per unit increase in feature x_i.

Decision boundary

$$P = 0.5 \;\Leftrightarrow\; \mathbf{w}^T\mathbf{x} + b = 0$$

Loss (binary cross-entropy)

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1{-}y_i)\log(1{-}\hat{y}_i)\right]$$

Logistic regression is the workhorse of credit scoring because each coefficient has a direct economic interpretation.

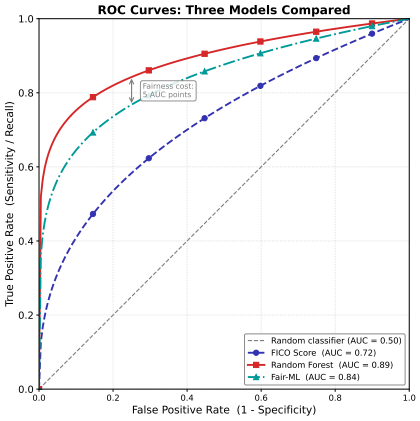
Building a Credit Scoring Model in 20 Lines of Python

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, classification_report

# Load and prepare data
df = pd.read_csv('lending_club_sample.csv')
features = ['annual_inc', 'dti', 'fico_score', 'emp_length',
            'loan_amnt', 'int_rate', 'revol_bal']
X = df[features].fillna(df[features].median())
y = df['default'].astype(int)

# Train/test split (80/20, stratified)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

# Fit logistic regression with L2 regularization
model = LogisticRegression(max_iter=1000, C=1.0)
model.fit(X_train, y_train)

# Evaluate on held-out test set
y_prob = model.predict_proba(X_test)[:, 1]
print(f"AUC: {roc_auc_score(y_test, y_prob):.3f}")
print(classification_report(y_test, model.predict(X_test)))
```

Key choices: `C=1.0` controls regularization strength (higher = less regularization). Stratified split preserves class balance. AUC is the primary metric because class imbalance makes accuracy misleading.

This minimal example fits a real credit scoring model. The next three slides analyze what it learns.

How Do You Choose the Right Threshold for Loan Approval?



Simulated curves based on published performance benchmarks for credit scoring models

What the ROC curve shows
Plots TPR vs. FPR at every threshold.

AUC: probability that a randomly chosen defaulter scores higher than a non-defaulter.

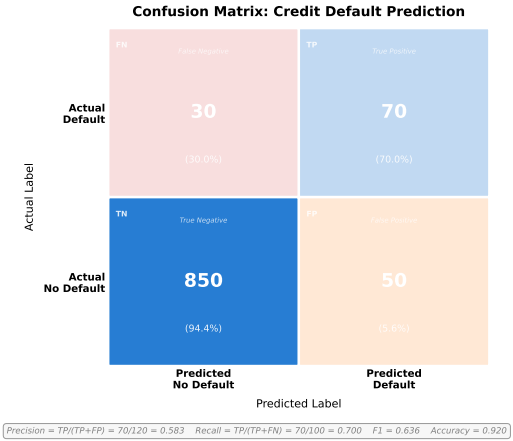
Threshold selection

- Higher threshold: fewer approvals, fewer defaults, more missed good borrowers
- Lower threshold: more approvals, more defaults, better inclusion

Cost-sensitive thresholding: Cost $= c_{FN} \cdot FN + c_{FP} \cdot FP$. Banks set $c_{FN} \gg c_{FP}$ because a missed default costs the entire loan principal.

AUC measures overall discrimination ability. The THRESHOLD is where business judgment meets statistics.

Why Is Missing a Default 10x Worse than Rejecting a Good Borrower?



Cost asymmetry in lending

- False Negative (missed default): lose entire principal (e.g., $10,000)
- False Positive (rejected good borrower): lose interest income (e.g., $1,000)
- Cost ratio: $c_{FN}/c_{FP} \approx 10$

Key metrics

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

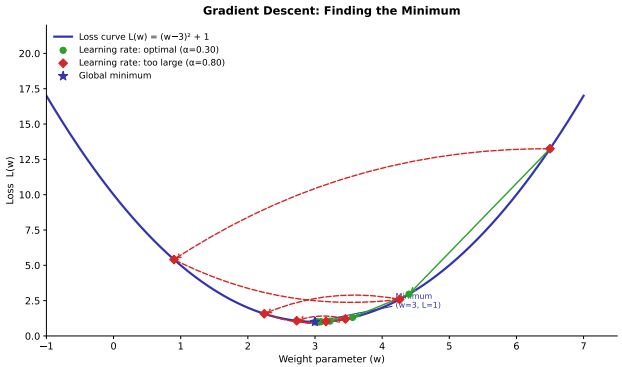
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}$$

In credit scoring, **recall** (catching defaults) matters more than precision.

In credit scoring, recall (catching defaults) matters more than precision – a missed default costs the entire loan principal.

How Does the Model Find the Best Weights?



Gradient descent update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \, \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t)$$

Learning rate η

- Too large: oscillation, divergence
- Too small: slow convergence
- Just right: steady decrease in loss

Stochastic Gradient Descent (SGD)
Use random mini-batches instead of the full dataset for each update – trades precision for speed.

Convergence criterion

$$|\mathcal{L}_t - \mathcal{L}_{t-1}| < \epsilon$$

When the loss stops decreasing meaningfully, the model has found its (local) optimum.

Gradient descent is the optimization engine behind every ML credit model – understanding it demystifies "the algorithm."

What Separates a Raw Variable from a Predictive Feature?

Raw Data	Engineered Feature	Rationale
Monthly income series	Income stability (coefficient of variation)	Captures consistency, not just level
Transaction history	Essential/discretionary spending ratio	Reveals financial discipline
Employment records	Job tenure (months at current employer)	Stability signal
Bank account balance	Minimum balance over 6 months	Worst-case liquidity
Loan payment history	Days-past-due frequency	Direct default predictor

Controversial features (use with caution):

- ZIP code, school name, device type, browsing behavior – all correlate with protected attributes (race, gender, socioeconomic status)
- Feature interactions: `income * zip_code` may encode discrimination even if neither variable alone does
- The EU AI Act classifies credit scoring as "high-risk AI" – every feature must be justified and auditable

Feature engineering is where domain knowledge meets data science – and where discrimination can enter invisibly.

Transforming Raw Data into Credit Features in Python

```python
def engineer_credit_features(df):
    """Transform raw lending data into ML-ready features."""
    features = pd.DataFrame()

    # Income stability (coefficient of variation)
    features['income_cv'] = df.groupby('borrower_id')[
        'monthly_income'].transform(
        lambda x: x.std() / x.mean() if x.mean() > 0 else 1.0)

    # Debt-to-income ratio
    features['dti'] = (df['total_debt']
                       / df['annual_income'].clip(lower=1))

    # Payment discipline: fraction of on-time payments
    features['ontime_ratio'] = (df['payments_ontime']
                                / df['total_payments'].clip(lower=1))

    # Spending behavior: essential / total spending
    features['essential_ratio'] = (df['essential_spend']
                                   / df['total_spend'].clip(lower=1))

    # Credit utilization
    features['utilization'] = (df['revolving_balance']
                               / df['credit_limit'].clip(lower=1))

    return features
```

Note: `.clip(lower=1)` prevents division by zero. The `groupby` + `transform` pattern computes time-series features per borrower while preserving the original DataFrame shape.

Good feature engineering adds more predictive power than a more complex model – always start with features, not algorithms.

How Do You Measure Risk Mathematically?

Expected portfolio return

$$E[R_p] = \sum_{i=1}^{n} w_i \, E[R_i]$$

Portfolio variance

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i \, w_j \, \sigma_{ij} = \mathbf{w}^T \mathbf{\Sigma} \, \mathbf{w}$$

Correlation

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \, \sigma_j}, \qquad -1 \leq \rho_{ij} \leq 1$$

Diversification benefit (equal weights, equal pairwise correlation ρ):

$$\sigma_p^2 = \frac{\sigma^2}{n} + \frac{n-1}{n} \, \rho \, \sigma^2$$

As $n \to \infty$: $\sigma_p^2 \to \rho \, \sigma^2$ (systematic risk cannot be diversified away).

Intuition: the umbrella and the sunscreen

If you own stock in an umbrella company and a sunscreen company, your portfolio is safer than owning either alone. On rainy days the umbrella company profits; on sunny days the sunscreen company profits. Your portfolio earns something every day.

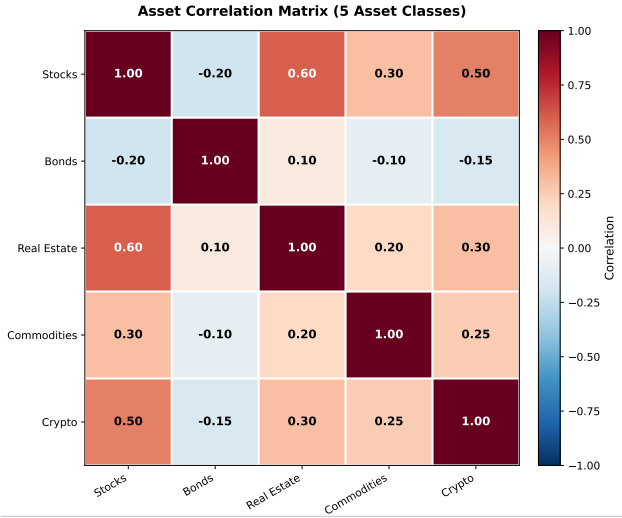
The math says the same thing:

- When $\rho = +1$: no diversification benefit at all (both go up and down together)
- When $\rho = 0$: risk drops by $\sqrt{n}$ (independent assets)
- When $\rho = -1$: perfect hedge – risk can reach zero

Real-world complication: during crises, correlations spike toward $+1$ – diversification fails precisely when you need it most.

The portfolio variance formula is the mathematical foundation of everything robo-advisors do.

Why Does Correlation Between Assets Determine Diversification Benefit?



Five asset classes
Stocks, Bonds, Real Estate, Commodities, Crypto.

Key observations

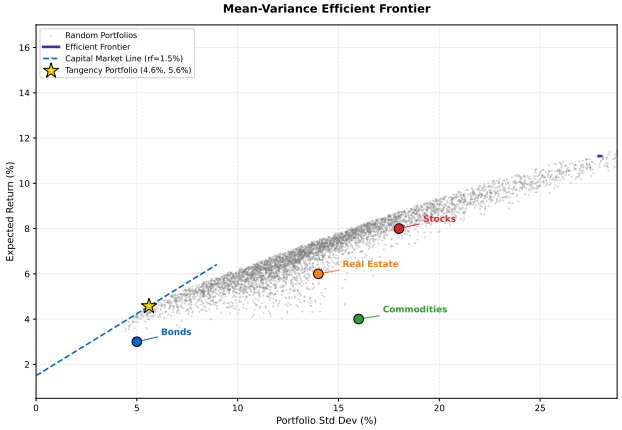
- Bonds–Stocks ($\rho = -0.2$): strongest diversification pair – when stocks fall, bonds tend to rise
- Crypto–Stocks ($\rho = 0.5$): limited diversification benefit – crypto is NOT a safe haven during equity crashes
- Real Estate–Stocks ($\rho = 0.6$): moderate correlation – both driven by economic growth

During crises, correlations spike toward $+1$. The matrix you see here reflects normal times – not the stress scenarios that matter most for risk management.

Garbage in, garbage out: the correlation matrix is the INPUT to portfolio optimization. Estimation errors propagate directly into sub-optimal allocations.

The correlation matrix is the INPUT to Modern Portfolio Theory – garbage in, garbage out.

How Do Robo-Advisors Find the Optimal Portfolio?



Markowitz optimization

$$\min_{\mathbf{w}} \ \mathbf{w}^T \boldsymbol{\Sigma}\, \mathbf{w}$$

subject to:

$$\mathbf{w}^T \boldsymbol{\mu} = \mu_{\text{target}}$$
$$\mathbf{w}^T \mathbf{1} = 1$$
$$w_i \geq 0 \ \ \forall\, i$$

Efficient frontier: the set of all optimal portfolios as μ_{target} varies.

Capital Market Line

$$E[R_p] = r_f + \frac{E[R_m] - r_f}{\sigma_m}\, \sigma_p$$

Tangency portfolio: the portfolio with the maximum Sharpe ratio $= \dfrac{E[R_p] - r_f}{\sigma_p}$.

Robo-advisors solve this optimization problem in milliseconds – the same math that earned Markowitz the Nobel Prize.

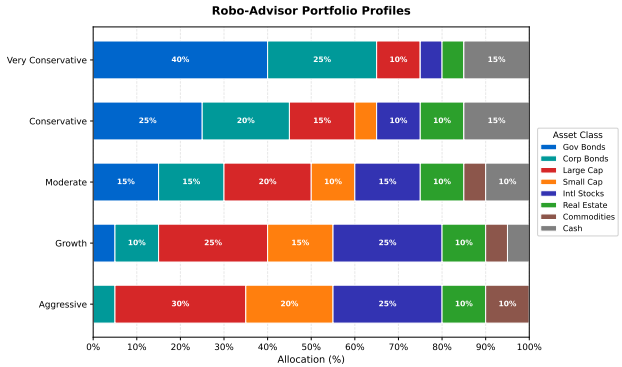
Mean-Variance Portfolio Optimization in Python

```python
import numpy as np
from scipy.optimize import minimize

def efficient_frontier(mu, cov, n_points=50, rf=0.015):
    """Compute the mean-variance efficient frontier."""
    n = len(mu)
    target_returns = np.linspace(mu.min(), mu.max(), n_points)
    frontier = []

    for target in target_returns:
        # Minimize portfolio variance subject to constraints
        result = minimize(
            fun=lambda w: w @ cov @ w,       # Objective
            x0=np.ones(n) / n,               # Equal-weight start
            method='SLSQP',
            bounds=[(0, 1)] * n,             # No short-selling
            constraints=[
                {'type': 'eq', 'fun': lambda w, t=target:
                    w @ mu - t},             # Return target
                {'type': 'eq', 'fun': lambda w:
                    np.sum(w) - 1},          # Weights sum to 1
            ])
        if result.success:
            vol = np.sqrt(result.fun)
            frontier.append((vol, target, result.x))

    return frontier
```

Note: `SLSQP` (Sequential Least Squares Programming) handles equality and inequality constraints. `bounds=[(0,1)]` enforces the no-short-selling constraint. The `t=target` default argument avoids Python's late-binding closure problem.

This 25-line function computes the same efficient frontier that a traditional advisor draws by hand – in under a second.

How Does Your Risk Tolerance Translate to an Asset Mix?



Five risk profiles

- Very Conservative: 80% bonds, capital preservation
- Conservative: 60% bonds, income focus
- Moderate: balanced stocks and bonds
- Growth: 60% stocks, long-term appreciation
- Aggressive: 80% stocks + alternatives

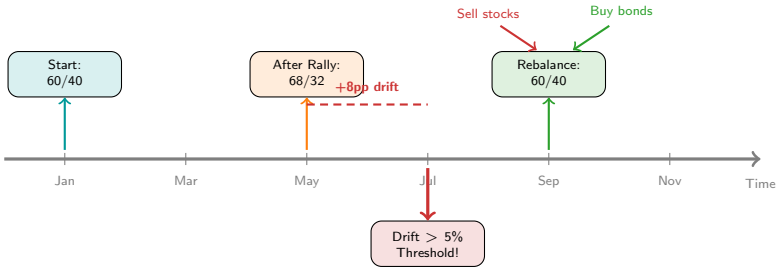
Risk questionnaire mapping

- Time horizon (most important factor)
- Income stability and emergency fund
- Loss tolerance ("How would you react to a 20% portfolio drop?")
- Investment experience

Key insight: The "moderate" profile is the most popular – but moderation in allocation does not mean moderate risk in absolute terms. A balanced portfolio can still lose 20% in a crisis.

Your risk profile is a function of TIME HORIZON, not personality – a 25-year-old can afford more volatility than a 60-year-old.

What Happens When Your Portfolio Drifts from Its Target?

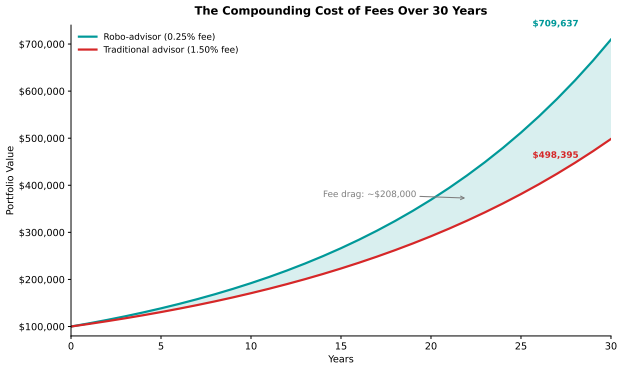


Tax-loss harvesting: Sell losers to realize capital losses, immediately buy correlated (but not "substantially identical") ETF. Same exposure, lower tax bill.

Tax savings = Capital loss × Marginal tax rate

Automated rebalancing and tax-loss harvesting are where robo-advisors create measurable value – estimated 0.5–1.5% annually in after-tax returns.

How Much Does a 1.25% Fee Difference Really Cost Over 30 Years?



Compound wealth formula

$$W_T = W_0 \cdot (1 + r - f)^T$$

Robo-advisor (0.25% fee):

$$W_{30} = 100{,}000 \times (1.0675)^{30} \approx \$706{,}859$$

Traditional advisor (1.50% fee):

$$W_{30} = 100{,}000 \times (1.055)^{30} \approx \$498{,}395$$

Difference: $\sim$\$208,000
That is 29.5% of the robo-advisor's final value – lost purely to higher fees.

The fee difference is invisible year-by-year (about \$1,250 in year one) but devastating over a career because it *compounds against you*.

Fees are the only variable in investing that you can control with certainty – and they compound against you.

What Does 'Fair' Actually Mean – Mathematically?

Let A = protected attribute (e.g., race), Y = true outcome, $\hat{Y}$ = model prediction.

Demographic Parity:

$$P(\hat{Y}{=}1 \mid A{=}0) = P(\hat{Y}{=}1 \mid A{=}1)$$

"Approval rates must be equal across groups."

Equalized Odds:

$$P(\hat{Y}{=}1 \mid Y{=}y,\ A{=}0) = P(\hat{Y}{=}1 \mid Y{=}y,\ A{=}1) \quad \forall\, y \in \{0, 1\}$$

"True positive and false positive rates must be equal across groups."

Individual Fairness:

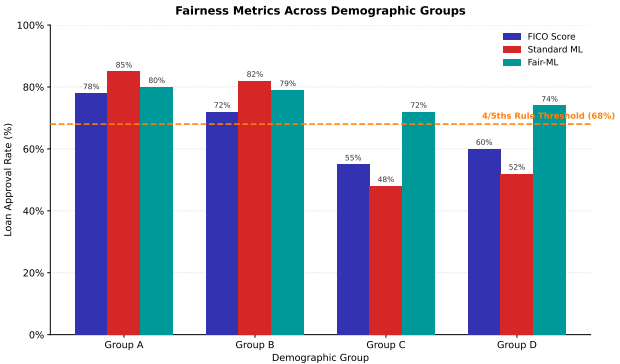
$$d\big(\hat{Y}(\mathbf{x}_1),\ \hat{Y}(\mathbf{x}_2)\big) \leq \epsilon \quad \text{if} \quad d(\mathbf{x}_1, \mathbf{x}_2) \leq \delta$$

"Similar individuals should be treated similarly."

Impossibility Theorem (Chouldechova 2017): Except in trivial cases, a model **cannot** simultaneously satisfy calibration, predictive parity, and equalized odds. **You MUST choose** which notion of fairness to optimize – you cannot have all of them.

The impossibility theorem is the mathematical proof that the fairness-accuracy tradeoff is not an engineering problem – it is a VALUES problem.

How Do Different Models Score on Fairness Across Groups?



Three models, four demographic groups

The 4/5ths rule (US EEOC): if Group B's approval rate $< 80\%$ of Group A's rate, disparate impact is presumed.

Results

- ML Black-Box: highest overall accuracy, but Group C falls below the 4/5ths threshold – presumed disparate impact
- Fair-ML: all groups above threshold, but overall accuracy drops 5–8 AUC points
- FICO: moderate disparity, moderate accuracy

The question is not WHETHER to trade accuracy for fairness, **but HOW MUCH**.

The 4/5ths rule is a legal standard, not a mathematical one – but it translates directly to constraints on the optimization problem.

The 4/5ths rule (US EEOC) is a legal standard, not a mathematical one – but it translates directly to constraints on the optimization problem.

Auditing a Credit Model for Bias in Python

```python
from sklearn.metrics import confusion_matrix
import numpy as np

def fairness_audit(y_true, y_pred, protected_attr):
    """Compute fairness metrics by protected group."""
    groups = np.unique(protected_attr)
    results = {}
    for g in groups:
        mask = (protected_attr == g)
        cm = confusion_matrix(y_true[mask], y_pred[mask])
        tn, fp, fn, tp = cm.ravel()
        # y=1 means DEFAULT. Approved = predicted non-default
        results[g] = {
            'approval_rate': (tn + fn) / len(y_true[mask]),
            'tpr': tp / (tp + fn) if (tp + fn) > 0 else 0,
            'fpr': fp / (fp + tn) if (fp + tn) > 0 else 0,
        }
    # Check demographic parity (4/5ths rule)
    rates = [r['approval_rate'] for r in results.values()]
    max_rate = max(rates)
    for g, r in results.items():
        r['disparate_impact'] = r['approval_rate'] / max_rate
        r['passes_4_5ths'] = r['disparate_impact'] >= 0.8

    return results
```

Usage: Pass true labels, model predictions, and a protected attribute array. The function returns per-group approval rates, TPR, FPR, and a 4/5ths rule check. A production system needs ongoing monitoring, not just a one-time audit.

This audit function is the minimum viable fairness check – a production system needs ongoing monitoring, not just a one-time audit.

What Do Regulators Demand from Algorithmic Lending?

Regulation	Jurisdiction	Key Requirement	Impact on ML Models
EU AI Act	EU	High-risk classification for credit scoring; explainability, human oversight, bias testing	Must audit models, provide explanations for rejections
ECOA	US	Prohibits discrimination based on race, religion, sex, age	Models cannot use protected attributes (but proxies still enter)
FCRA	US	Right to know what is in your credit file; dispute inaccurate data	Alternative data sources must be disputable
GDPR Art. 22	EU	Right not to be subject to purely automated decisions	Human-in-the-loop required for credit decisions
FINMA 2008/21	CH	Model risk management for credit models	Validation, back-testing, documentation

The regulatory trajectory is clear: **more oversight, more explainability, more fairness auditing**. Models that cannot explain their decisions will not survive the next wave of regulation.

Regulation is converging globally toward explainability and fairness – black-box models face an increasingly hostile regulatory environment.

When Does an Algorithm Actually Outperform a Human Advisor?

Dimension	Robo	Human	Edge
Portfolio construction	MPT-optimal	Often sub-optimal (home bias)	Robo
Behavioral coaching	Nudges, auto-rebalance	Talk off the ledge	Human
Tax optimization	Systematic TLH	Ad hoc, advisor-dependent	Robo
Estate planning	None	Comprehensive	Human
Conflict of interest	None (algorithm)	Commission risk	Robo
Black swan response	Follows rules	Can exercise judgment	Human
Trust / relationship	Low attachment	Decades of trust	Human

Synthesis

Robo-advisors win on **mechanical tasks**: rebalancing, tax-loss harvesting, fee minimization. These are rule-based, repeatable, and benefit from automation.

Humans win on **judgment tasks**: estate planning, crisis response, behavioral coaching. These require empathy, context, and the ability to deviate from the model.

The hybrid model wins overall.

The future is not robo OR human – it is robo for the mechanics and human for the judgment. Most major platforms (Vanguard Personal Advisor, Schwab Intelligent Portfolios Premium) now offer hybrid tiers.

The future is not robo OR human – it is robo for the mechanics and human for the judgment.

Automated Rebalancing: The Code Behind the Robo-Advisor

```python
def rebalance_portfolio(current_weights, target_weights,
                        threshold=0.05):
    """Determine trades needed to rebalance a portfolio.

    Args:
        current_weights: dict {asset: current_weight}
        target_weights: dict {asset: target_weight}
        threshold: max allowed drift before rebalancing

    Returns:
        trades: dict {asset: amount} (positive=buy, negative=sell)
    """
    trades = {}
    needs_rebalance = False

    for asset in target_weights:
        drift = abs(current_weights.get(asset, 0)
                    - target_weights[asset])
        if drift > threshold:
            needs_rebalance = True
            break

    if needs_rebalance:
        for asset in target_weights:
            trades[asset] = (target_weights[asset]
                             - current_weights.get(asset, 0))

    return trades if needs_rebalance else {}
```

Example: Current {stocks: 0.68, bonds: 0.32}, target {stocks: 0.60, bonds: 0.40}, threshold 0.05. Drift $= 0.08 > 0.05$, so trades $=$ {stocks: -0.08, bonds: $+0.08$}.

This simplified rebalancer captures the core logic. Production systems add tax-lot selection, transaction cost minimization, and wash-sale avoidance.

When Alternative Data Crosses the Line: Three Real Cases

Case 1: Apple Card (2019)

- **What happened:** Goldman Sachs' algorithm gave women lower credit limits than their husbands – even with higher credit scores
- **Why it matters:** The model did not use gender as a feature, but correlated features produced gendered outcomes
- **Outcome:** Investigated by NYDFS; Goldman committed to algorithmic auditing

Case 2: Upstart (2020)

- **What happened:** Used education and employment data for credit scoring, creating proxy discrimination against minority borrowers
- **Why it matters:** "Alternative data" often encodes the same biases as traditional data through different channels
- **Outcome:** DOJ settlement requiring ongoing disparate impact monitoring

Case 3: Ant Group (2020)

- **What happened:** Used Sesame Credit social scoring to influence lending decisions at massive scale
- **Why it matters:** Social scoring blurs the line between creditworthiness and social conformity
- **Outcome:** Chinese regulators forced restructuring; IPO halted

Synthesis: In every case, the algorithm produced accurate predictions. In every case, the predictions were also discriminatory.

These are not hypothetical scenarios – they are regulatory actions taken in the last five years against real companies.

Is Data Replacing Banks, or Just Creating New Middlemen?

Problem	Bank Solution	Platform Solution	New Risk
Adverse selection	Relationship, collateral	ML scoring, alt data	Algorithmic bias
Moral hazard	Monitoring, covenants	Automated tracking	Privacy invasion
Principal-agent	Fiduciary duty	Algorithm transparency	No recourse
Diversification	Advisor expertise	MPT automation	Model risk
Access	Branch network	Smartphone app	Digital divide

The key insight

P2P lending and robo-advisory did not eliminate intermediation. They replaced **human** intermediaries with **algorithmic** ones.

The information problems are the same:

- Borrowers still know more than lenders
- Agents still have misaligned incentives
- Diversification still requires expertise

The solutions are different – faster, cheaper, more scalable. But the failure modes are also different: model risk, algorithmic bias, and the illusion of objectivity.

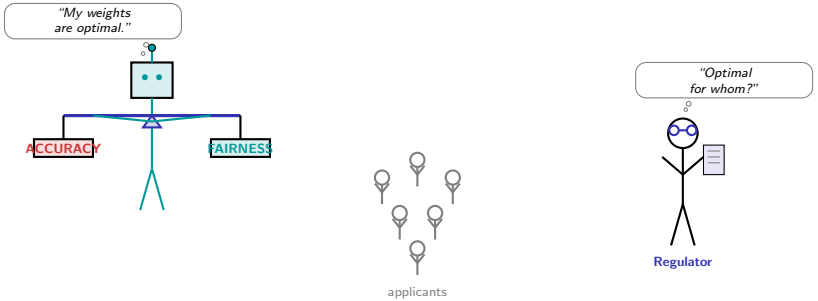
Data-driven finance solves information problems more cheaply but introduces new ones.

Data-driven finance solves information problems more cheaply but introduces new ones: model risk, algorithmic bias, and the illusion of objectivity.

Key Takeaways

1. **Information asymmetry** (adverse selection, moral hazard, principal-agent) is the fundamental problem in lending and advisory – technology changes the solution, not the problem
2. **Logistic regression** remains the industry workhorse for credit scoring because of interpretability, but ML models (random forest, gradient boosting) achieve higher AUC at the cost of explainability
3. **Model evaluation** requires BOTH accuracy metrics (AUC, precision, recall) AND fairness metrics (demographic parity, equalized odds) – optimizing one without the other is irresponsible
4. **Mean-variance optimization** (Markowitz) is the mathematical engine inside every robo-advisor – understanding it demystifies "the algorithm"
5. **The fairness-accuracy tradeoff is not solvable** – it is a value judgment. The impossibility theorem (Chouldechova 2017) proves you cannot satisfy all fairness criteria simultaneously
6. **Regulation** (EU AI Act, ECOA, GDPR Art. 22) is converging toward explainability and fairness auditing – black-box models face an increasingly hostile environment

Next: Lesson 04 – RegTech and Compliance. How technology automates the regulatory response to these information problems.



The algorithm sees everything. The question is what it chooses to see.

Every model makes choices about what to optimize. The regulator's job is to ask: optimal for whom?