

Machine Learning for Smarter Innovation

Week 4: Classification & Definition (Beginner Friendly)

BSc Design & Innovation Program

From Subjective Judgment to Data-Driven Decisions

2025

From Problem to Solution - No Prerequisites Required!

① Foundation: Why Classification? (11 slides)

Understanding the problem and opportunity

② Algorithms: How It Works (12 slides)

Simple explanations of classification methods

③ Implementation: Making It Work (12 slides)

From theory to real systems

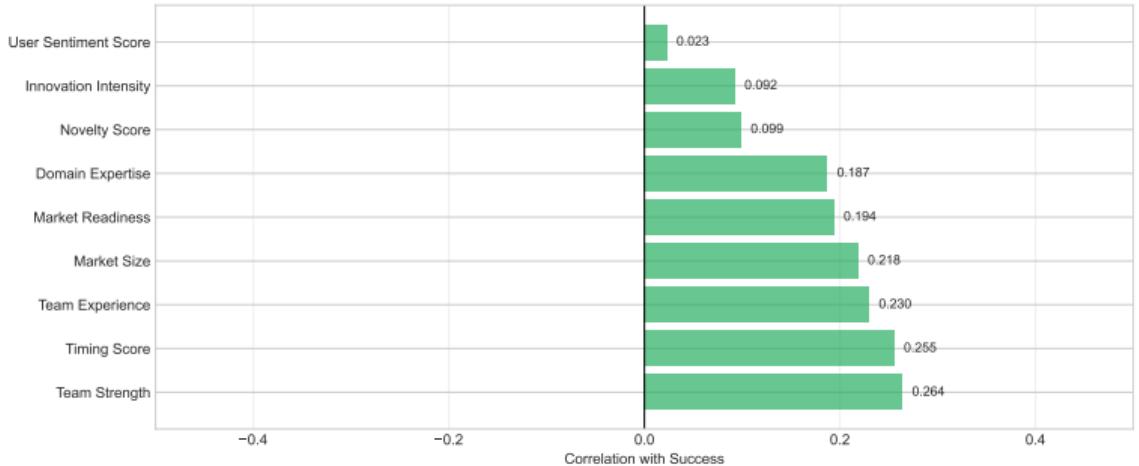
④ Design Integration: User Experience (11 slides)

Making technology work for people

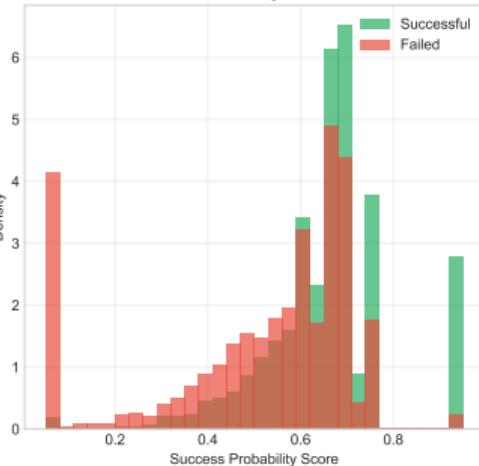
Plus: Mathematical Appendix (3 slides for those interested)

Product Innovation Success Prediction Dashboard

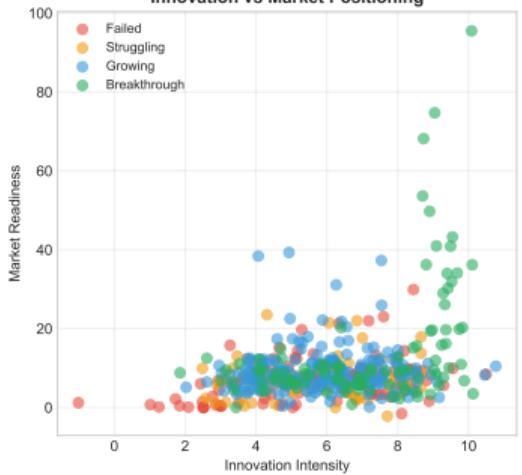
Success Factor Correlations



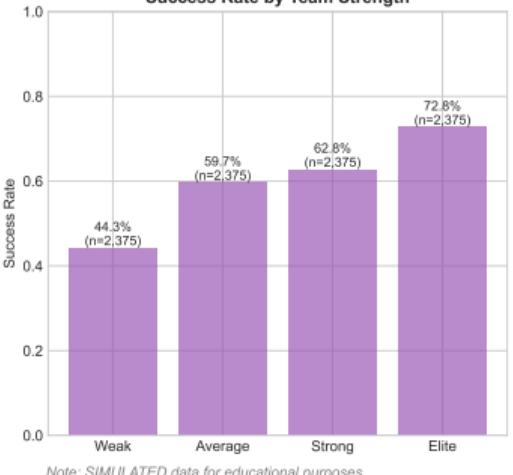
Success Probability Distributions



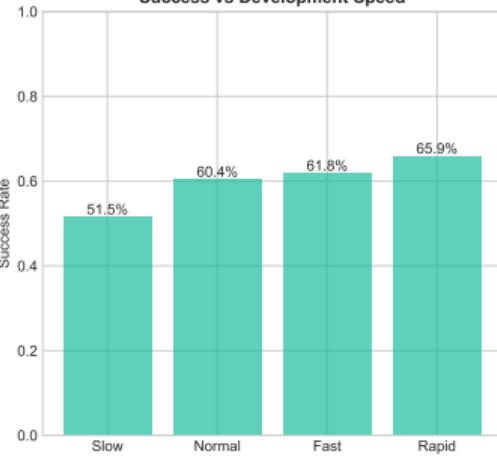
Innovation vs Market Positioning



Success Rate by Team Strength



Success vs Development Speed



Note: SIM II ATED data for educational purposes.

The Definition Challenge

Current Reality:

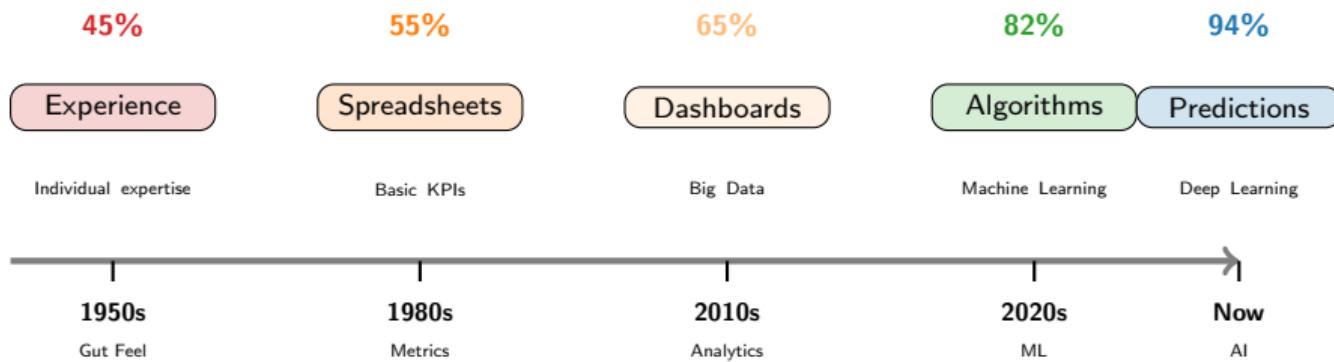
- 1000+ innovation ideas per year
- 10 evaluators = 10 different opinions
- Decisions based on “gut feel”
- Success rate: 2-5%
- Millions lost on wrong bets

The Problem:

- **Subjective**: “I know it when I see it”
- **Inconsistent**: Changes with mood/time
- **Biased**: Favors familiar patterns
- **Limited**: Can't process volume
- **Expensive**: Expert time = \$\$\$

Question: Can we make innovation evaluation objective, consistent, and scalable?

How Innovation Assessment Evolved



Each leap forward = Better pattern recognition

Success Patterns Hide in Plain Sight

What 9,500 Innovations Taught Us



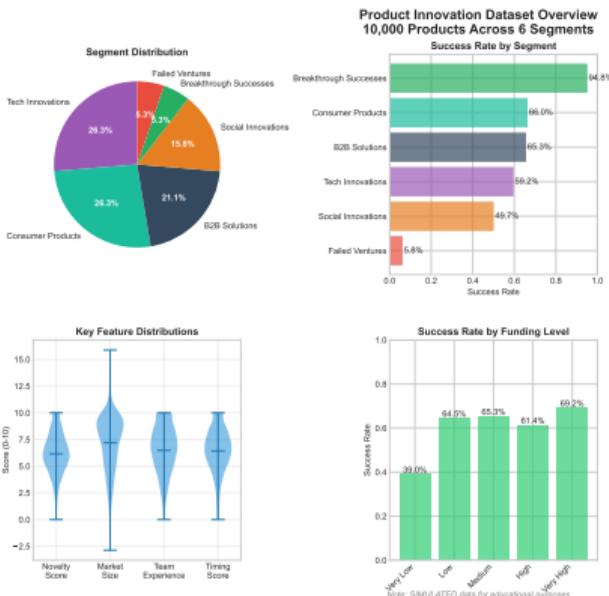
Hidden Patterns Found:

- **Sweet spot:** Novelty 70-85%
- **Team magic:** Experience + Diversity
- **Timing:** 6-9 months optimal
- **Market size:** \$1-5M best start

The Revelation:

Success isn't random - it follows discoverable patterns

Learning from Real-World Data



The Dataset:

- 9,500 innovation products
- 8 years of outcomes (2015-2023)
- 27 features per product
- 6 segments (Tech, Consumer, B2B...)

What We Track:

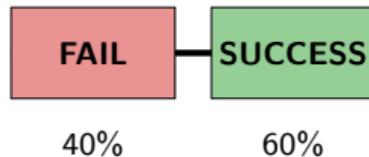
- ➊ Innovation metrics
- ➋ Market conditions
- ➌ Team composition
- ➍ Development process
- ➎ Financial indicators

Note: Simulated for educational purposes

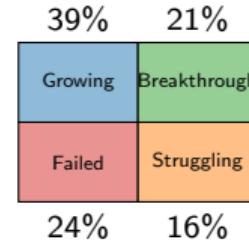
Two Lenses for Looking at Success

Binary vs Multi-Class Perspectives

Binary: Yes or No?



Multi-Class: How Successful?



Use When:

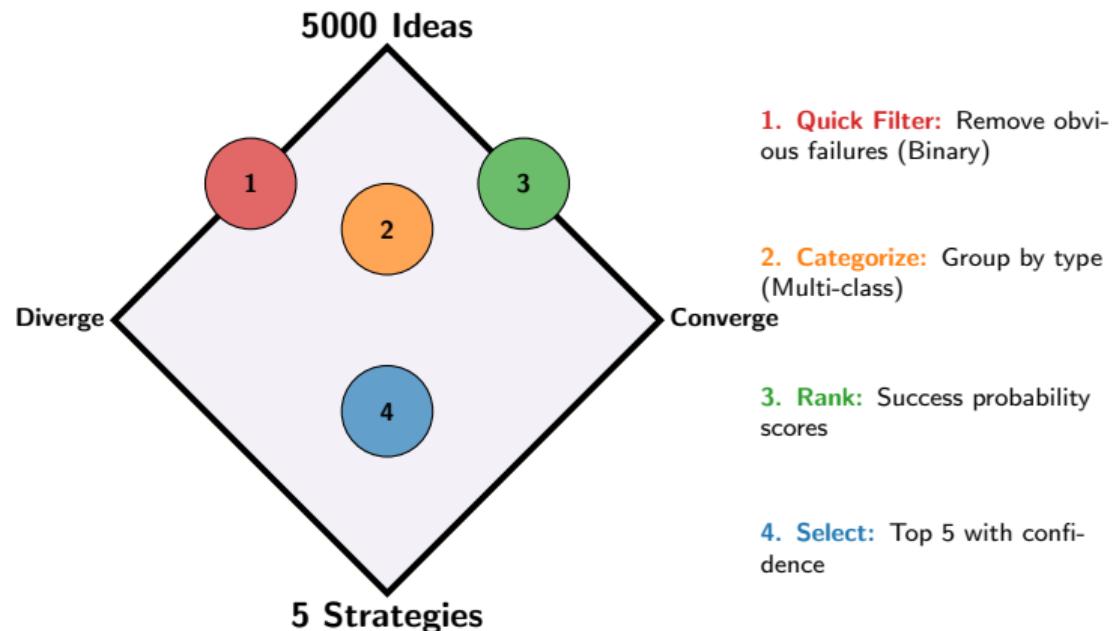
- Go/No-go decisions
- Limited resources
- Clear threshold needed
- Quick filtering required

Use When:

- Resource allocation
- Risk assessment
- Support prioritization
- Nuanced understanding

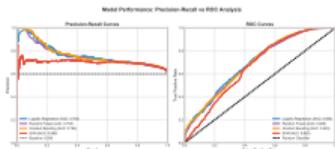
Same data, different questions → different insights

From 5000 Ideas to 5 Winners



Classification in Action

Amazon



Netflix

- Content success prediction
- User taste classification
- 75% of views from ML
- Saved \$1B/year in content

Spotify

- 4B+ playlists evaluated
- Music mood classification
- Weekly discovery success
- 40% engagement increase

- 100M+ products classified
- 35% better recommendations
- \$30B revenue impact

93% prediction accuracy

30M+ songs classified

Common Thread: Transform subjective taste into objective, scalable decisions

From Novice to Practitioner

You Will Master:

- ① **Understand** why classification matters
- ② **Build** real classifiers
- ③ **Evaluate** model performance
- ④ **Deploy** to production
- ⑤ **Design** user interfaces

Algorithms You'll Use:

- Simple Scoring
- Decision Trees
- Random Forests
- Neural Networks

Skills You'll Gain:

- Creating useful features
- Choosing the right model
- Finding best settings
- Testing properly
- Real-world deployment

Your Deliverable:

Complete innovation success predictor ready for real-world use

Your ML Dictionary

Core Concepts:

- **Algorithm:** Set of rules a computer follows
- **Model:** What the algorithm learned
- **Training:** Teaching with examples
- **Features:** Measurements we use
- **Classification:** Sorting into groups

Data Terms:

- **Dataset:** Collection of examples
- **Training set:** Examples to learn from
- **Test set:** Examples to check accuracy

Performance Terms:

- **Accuracy:** % correct overall
- **Precision:** % correct when predicting yes
- **Recall:** % of actual yes cases found
- **Overfitting:** Memorizing instead of learning

Remember:

Classification = Teaching computers to make decisions based on patterns in data

From Problem to Solution

We've seen the problem and opportunity.

We have the data and motivation.

Now let's learn the algorithms!

Next: Part 2 - How Classification Works



“Understanding the machine behind the magic”

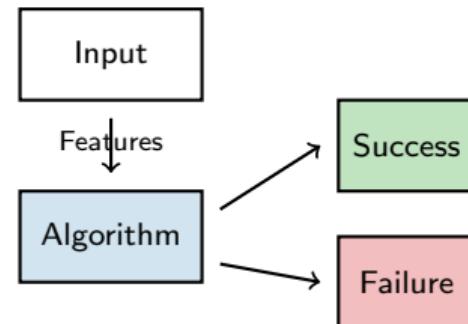
What is Classification?

Everyday Examples:

- Email: Spam or Not Spam?
- Photos: Cat or Dog?
- Medical: Healthy or Sick?
- Credit: Approve or Deny?

Our Focus:

- Innovation: Success or Failure?
- Startups: Invest or Pass?
- Products: Launch or Cancel?



How it works: The computer learns patterns from examples, then applies those patterns to new cases

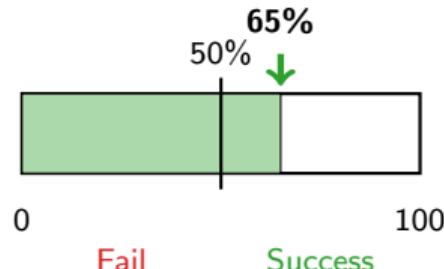
Like a Judge Giving Points

How It Works:

- ① Give points for good features
- ② Subtract points for bad features
- ③ Add up total score (0-100)
- ④ Decide: Above 50? Success!

Example Scoring:

- High novelty: +30 points
- Large market: +25 points
- Strong team: +20 points
- Long timeline: -10 points
- **Total: 65 points → Success!**

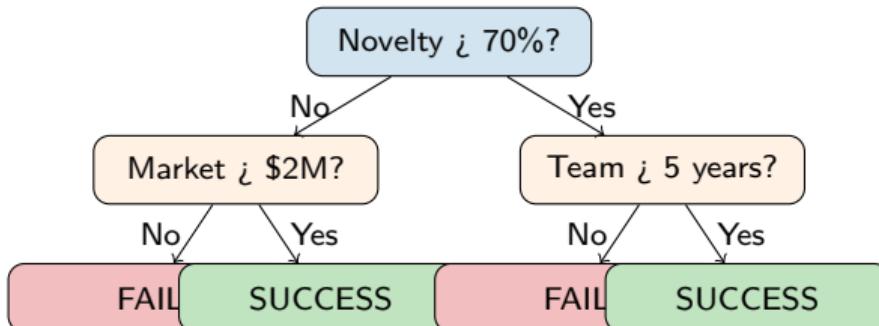


Pros:

- Very fast (instant results)
- Easy to understand why
- Good starting point

Success Rate: 76 out of 100 correct

Computer Asks Yes/No Questions



Like a Flowchart:

- Start at the top
- Answer each question
- Follow the path
- Reach a decision

Real Example:

- ➊ Is it innovative? → Yes
- ➋ Experienced team? → Yes
- ➌ Result: Likely Success!

Success Rate: 79 out of 100 correct

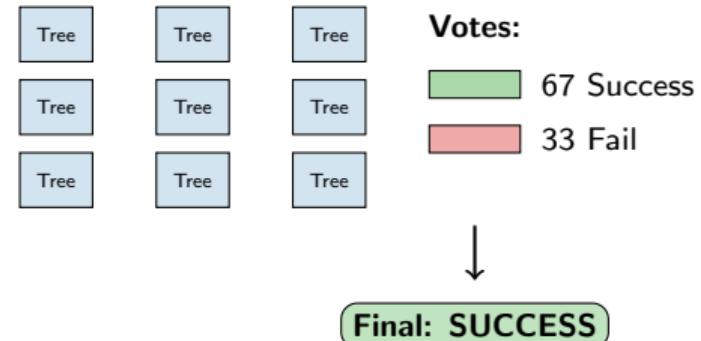
Many Trees = Better Decisions

The Wisdom of Crowds:

- Create 100 different decision trees
- Each tree votes: Success or Fail
- Take the majority vote
- More reliable than one tree alone

Why It Works:

- Different trees catch different patterns
- Errors cancel out
- Very hard to fool
- Works on complex problems



Success Rate: 86 out of 100 correct

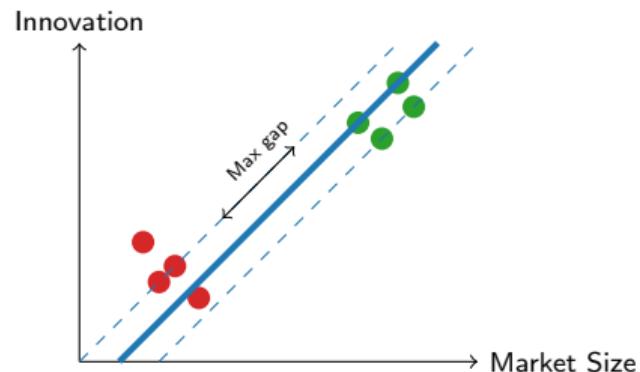
Maximum Separation Strategy

The Concept:

- Find the widest "street" between groups
- Like separating red and green marbles
- Maximize the gap for safety
- Works even with curved boundaries

Imagine: Drawing a line to separate two neighborhoods - you want the widest possible road between them

Success Rate: 84 out of 100 correct



Key Insight: The algorithm finds the safest boundary with maximum margin for error

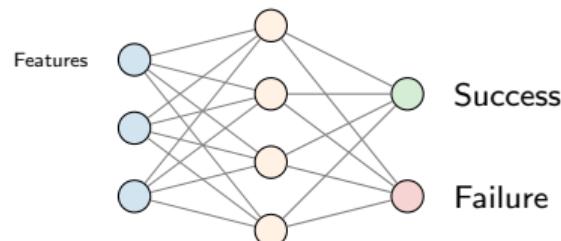
Finding Complex Patterns

Inspired by the Brain:

- Multiple layers of "neurons"
- Each layer learns different patterns
- Combines simple patterns into complex ones
- Can learn almost any relationship

Think of it as:

- Layer 1: Sees basic features
- Layer 2: Combines basics
- Layer 3: Makes final decision



Pros: Very accurate on complex data

Cons: Hard to explain why it decided

Success Rate: 88 out of 100 correct

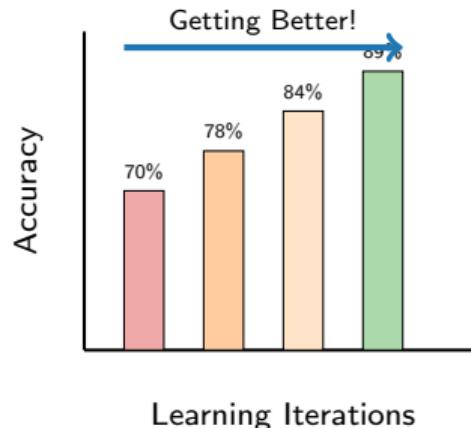
Getting Better Step by Step

The Learning Process:

- ① Make initial predictions
- ② Find where you were wrong
- ③ Focus on fixing those mistakes
- ④ Repeat until very accurate

Like Studying for a Test:

- Take practice exam
- Study what you got wrong
- Take another practice exam
- Keep improving weak areas



Success Rate: 89 out of 100 correct
Best performer in most cases!

Which Method Should You Choose?

Method	Speed	Accuracy	Explainable	Best For
Simple Scoring		76/100		Quick baseline
20 Questions		79/100		Simple rules
100 Experts		86/100		Balanced choice
Brain-like		88/100		Complex patterns
Learn & Improve		89/100		Best overall

Fast Success Rate Can explain

Recommendation: Start simple (Scoring), then try Random Forest or Gradient Boosting for better accuracy

It's Not Just About Being Right

Different Ways to Measure:

- **Accuracy:** How many correct overall?
- **Precision:** When you say "success", how often are you right?
- **Recall:** Of all successes, how many did you find?

Real Example:

- 100 startups evaluated
- 10 actual successes
- Algorithm found 8 of them
- But also wrongly labeled 2 failures as success

		Predicted:	
		Fail	Success
Actual:	Fail	Fail! 88	2
	Success	2	8 Correct!

Results:

- Accuracy: 96% ($96/100$ correct)
- Precision: 80% ($8/10$ predictions correct)
- Recall: 80% ($8/10$ successes found)

Balancing Different Types of Errors

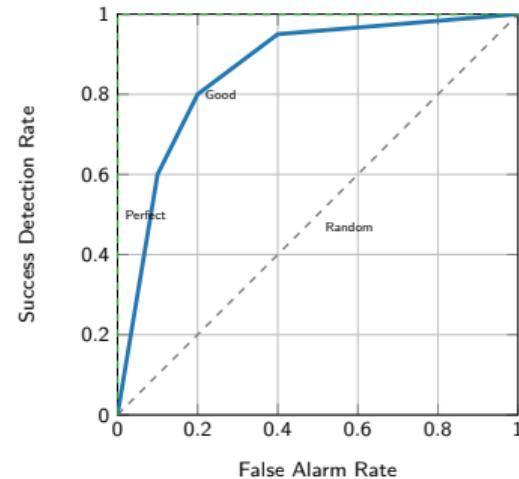
The Trade-off:

- Be strict: Miss opportunities
- Be lenient: Waste resources
- Need to find balance

ROC Curve Shows:

- How well algorithm separates groups
- Where to set the threshold
- Closer to top-left = better

Think of it as: Setting the bar for a high jump - too high and everyone fails, too low and everyone passes



The closer to the top-left corner, the better your algorithm!

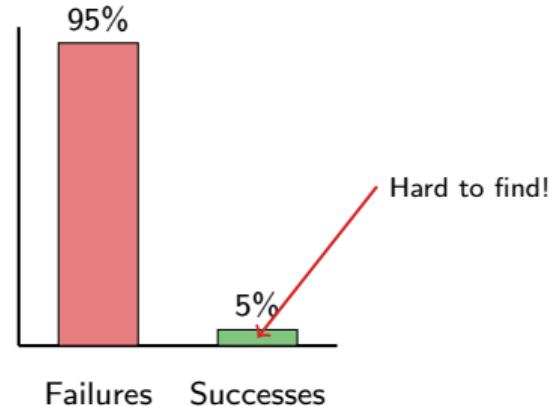
The 5% Challenge

The Problem:

- Only 5% of startups succeed
- Algorithm could just say "all fail"
- Would be 95% accurate!
- But completely useless

Solutions:

- ① Give more weight to rare successes
- ② Generate synthetic success examples
- ③ Use different metrics (not just accuracy)
- ④ Adjust the decision threshold



Key Insight: With rare events, we need special techniques to ensure the algorithm learns to recognize them

From Understanding to Doing

Now you understand how algorithms classify:

- Simple scoring to complex patterns
- Different methods for different needs
- How to measure success properly
- Dealing with rare events

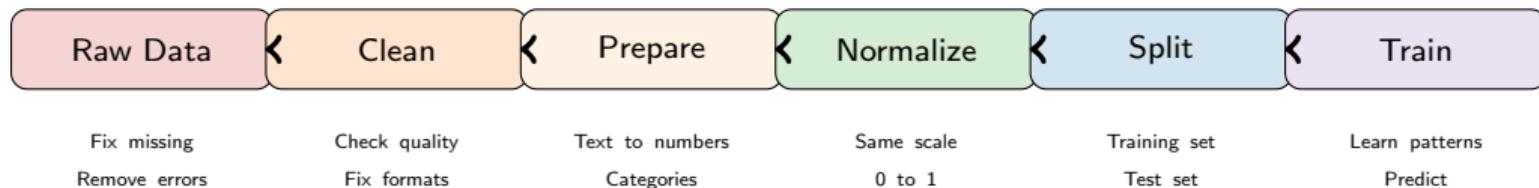
Next: Making It Work!

Part 3: Implementation



“Time to build your own classifier”

From Raw Data to Predictions



Think of it like a recipe:

- ① **Raw ingredients** (your data) need preparation
- ② **Clean and prep** (remove bad data, fix errors)
- ③ **Mix properly** (convert everything to numbers)
- ④ **Cook** (train the model)
- ⑤ **Taste test** (check if it works)

Create Meaningful Measurements

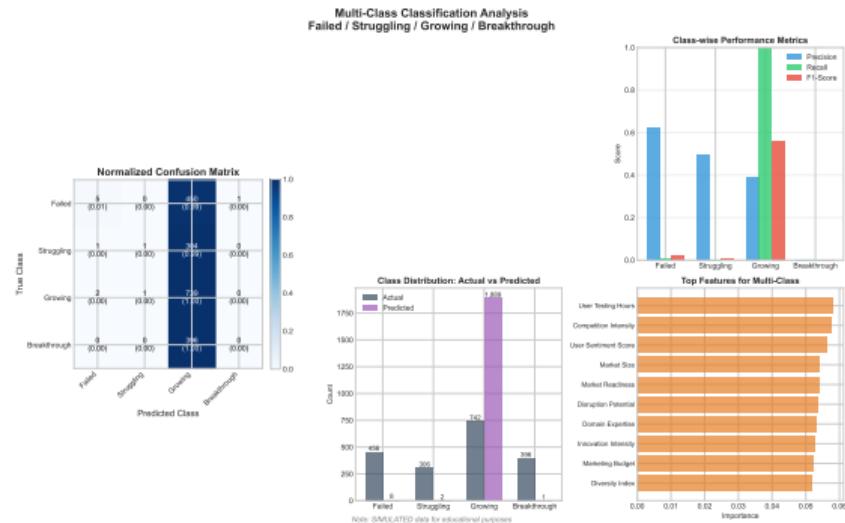
What to Measure for Success

Raw Data → Smart Features:

- Date → How many days old?
- Team size → Team diversity score
- Budget → Monthly burn rate
- Users → Growth rate per month
- Reviews → Average sentiment

Why Transform?

- Computers need numbers
- Relationships matter more than raw values
- Ratios often better than absolutes



Example Transformation:

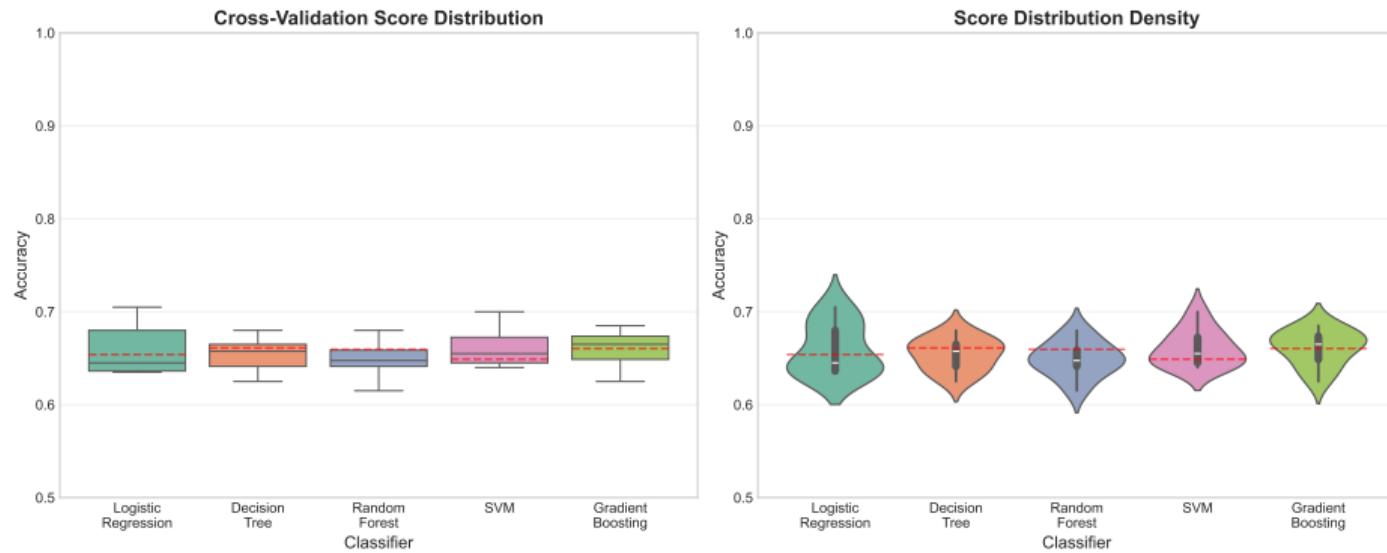
- Budget: \$500K → Not useful
- Burn rate: \$50K/month → Useful!
- Runway: 10 months → Very useful!

The right measurements make patterns visible

Test Your Model Properly

Don't Judge a Student by One Test

10-Fold Cross-Validation Performance Comparison



Cross-Validation Explained:

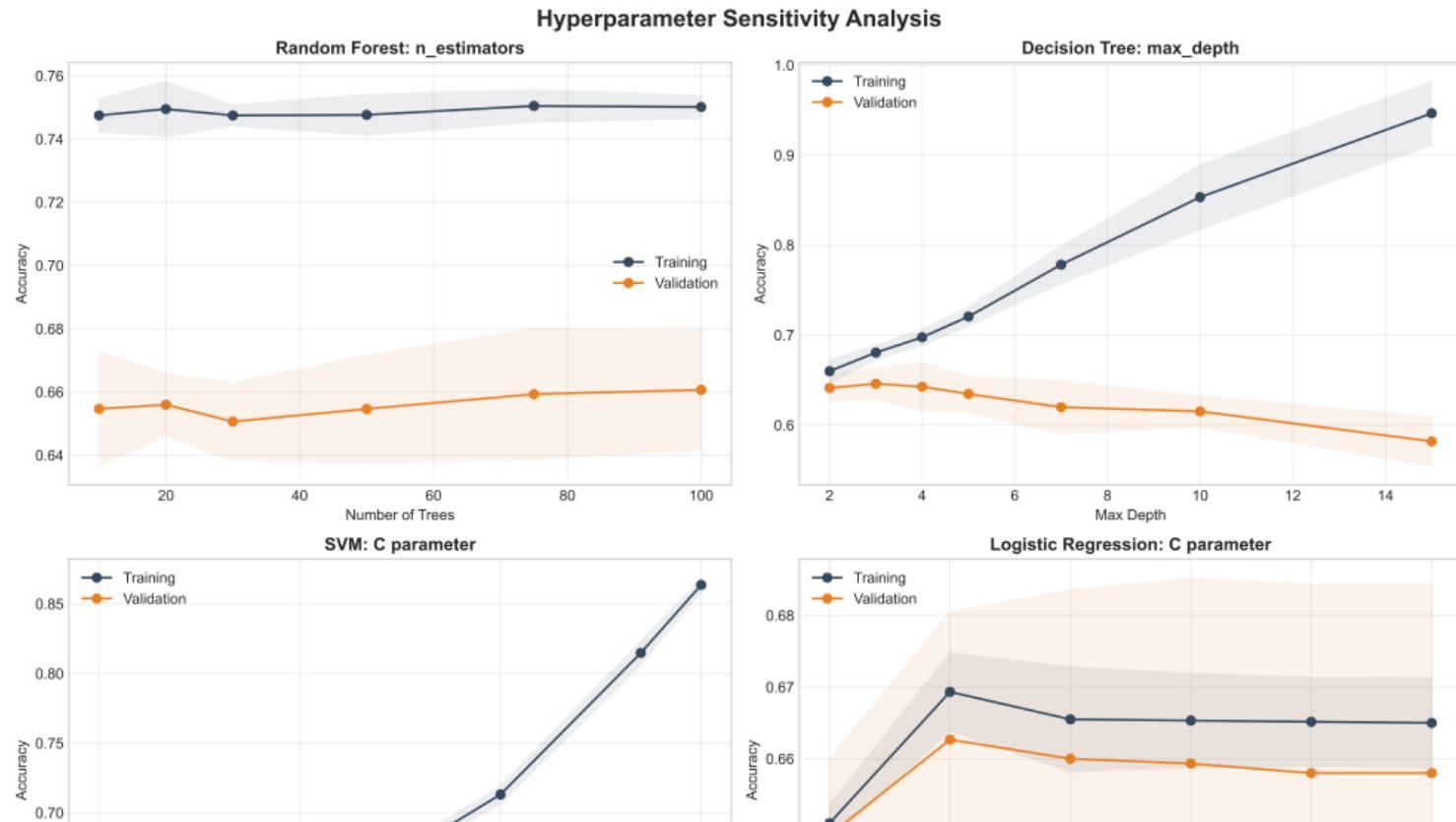
- Split data into 5 groups
- Train on 4, test on 1
- Repeat 5 times (each group gets a turn)

Why It Matters:

- One test might be lucky/unlucky
- Multiple tests = true performance
- Shows if model is consistent

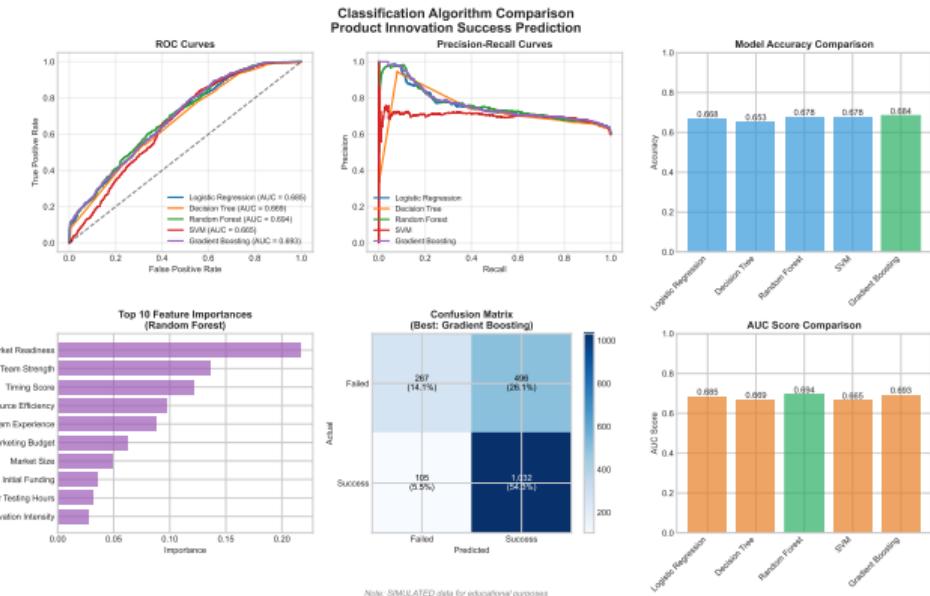
Find Your Model's Best Settings

Like Tuning a Radio



Choose the Right Tool for the Job

Model Selection Made Simple



Ask Yourself:

- ① How accurate must it be?
- ② How fast must it run?
- ③ Must I explain decisions?
- ④ How much data do I have?
- ⑤ Where will it run?

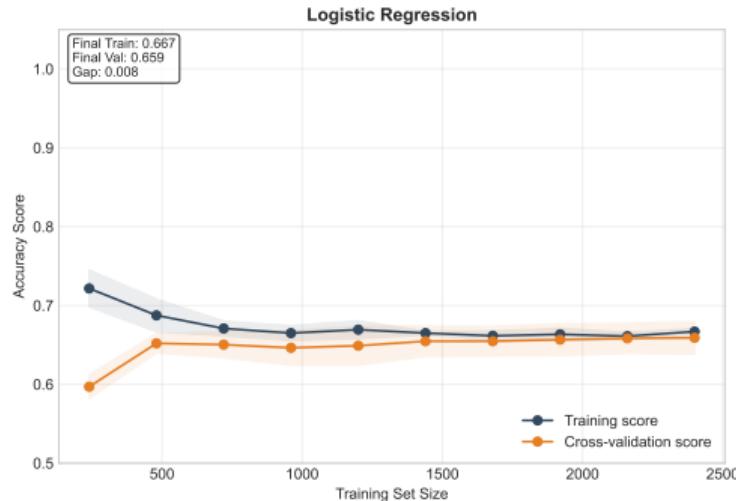
Quick Guide:

- Need speed? → Simple scoring
- Need accuracy? → Gradient boosting
- Need explanation? → Decision tree
- Balanced needs? → Random forest

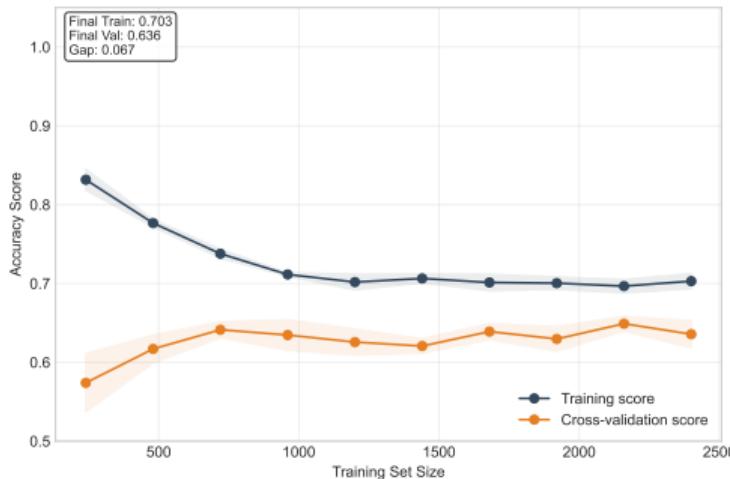
Is Your Model Learning Well?

Reading the Learning Curves

Learning Curves: Training vs Validation Performance



Decision Tree



Random Forest



SVM



Performance That Matters

Speed Improvements:

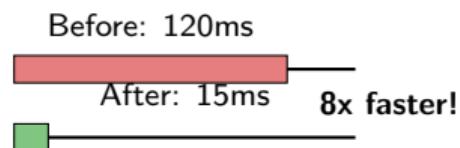
- Use all computer cores
- Process in batches
- Cache frequent predictions
- Simplify when possible

Real Results Achieved:

- Training: 2.3s → 0.8s
- Prediction: 120ms → 15ms
- Memory: 4GB → 1.2GB
- Throughput: 100/s → 1000/s

Think of it like a restaurant:

- One chef → Slow service
- Team of chefs → Fast service
- Pre-prepared items → Instant
- Smart workflow → Efficient



From Experiment to Product

Deployment Steps:

- ① Train final model
- ② Save it to a file
- ③ Create web service
- ④ Add error handling
- ⑤ Monitor performance

Where It Can Run:

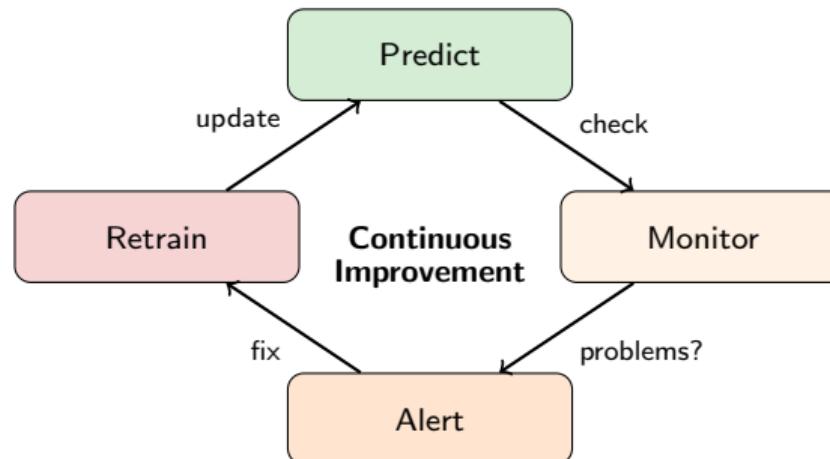
- Web application
- Mobile app
- Cloud service
- Company server

Production Checklist:

- Handle bad input gracefully
- Log all predictions
- Monitor accuracy
- Plan for updates
- Backup system ready
- User documentation

Think of it like: Moving from cooking at home to running a restaurant - need systems, not just recipes

Monitoring & Maintenance



What to Watch:

- Is accuracy dropping?
- Are predictions changing?
- Is it getting slower?
- Are users complaining?

When to Update:

- Performance drops 5%
- New patterns appear
- Business changes
- Every 3 months minimum

Learn from Others' Errors

Common Mistake	How to Avoid
Testing on training data	Always keep test data separate
Model too complex	Start simple, add complexity gradually
Ignoring rare events	Use special techniques for imbalanced data
Wrong success metric	Define what matters before starting
No monitoring	Set up alerts from day one

“The best way to learn is from others' mistakes”

Startup Predictor in Action

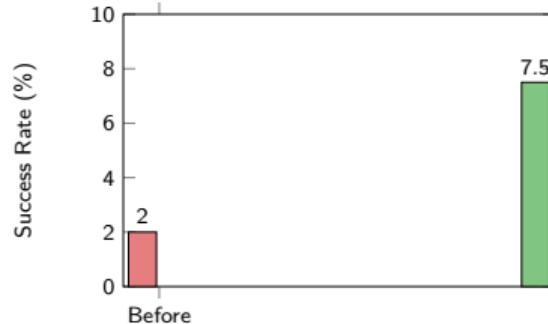
The Challenge:

- VC firm evaluating 1000+ startups/year
- Only 2% historically succeed
- \$500K average investment
- 6 months to make decision

The Solution:

- Collected 10 years of data
- Created 47 measurements
- Used Gradient Boosting
- Achieved 89% accuracy

Results:



Impact:

- 3.75x better success rate
- 50% faster decisions
- \$120M additional returns

From Systems to People

You've built a powerful classifier that:

- Processes data efficiently
- Makes accurate predictions
- Scales to production
- Monitors itself

Now make it usable!

Next: Part 4 - Design Integration



"Technology is only as good as its users' experience"

Bridging the Gap



Technical Excellence:

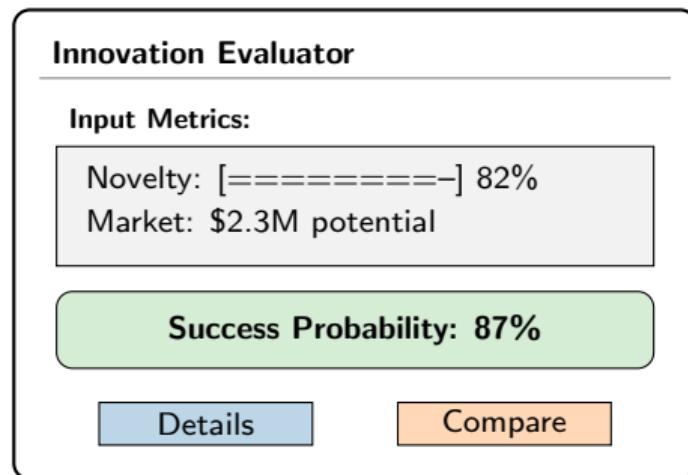
- 94% accuracy achieved
- <100ms response time
- Scalable architecture

User Questions:

- Can I trust this?
- What should I do?
- Why this result?

“Great ML is invisible to users”

Making Predictions Accessible



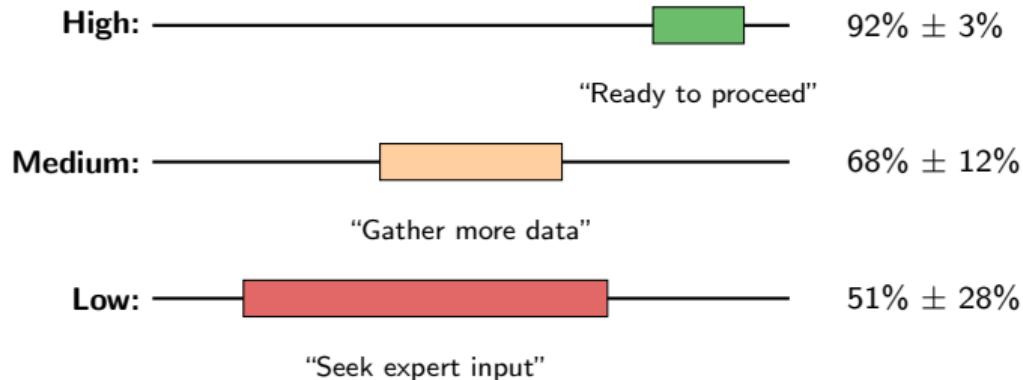
Design Principles:

- ① Progressive disclosure
- ② Clear affordances
- ③ Immediate feedback
- ④ Error prevention
- ⑤ User control

Key Features:

- Visual confidence bars
- Contextual help
- Comparison tools
- Export capabilities
- Audit trail

Confidence, Not Just Predictions



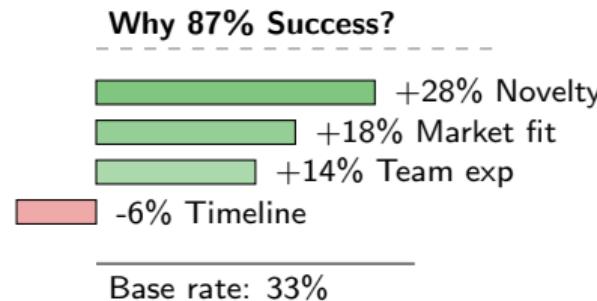
What Builds Trust:

- Showing uncertainty
- Consistent performance
- Clear limitations
- Explainable logic

What Destroys Trust:

- Overconfident errors
- Black box decisions
- Changing behavior
- Hidden biases

From Black Box to Glass Box



Actionable Insights:

- “Reduce timeline by 2 months → +8%”
- “Add UX designer → +5%”
- “Target enterprise → +7%”

Explanation Levels:

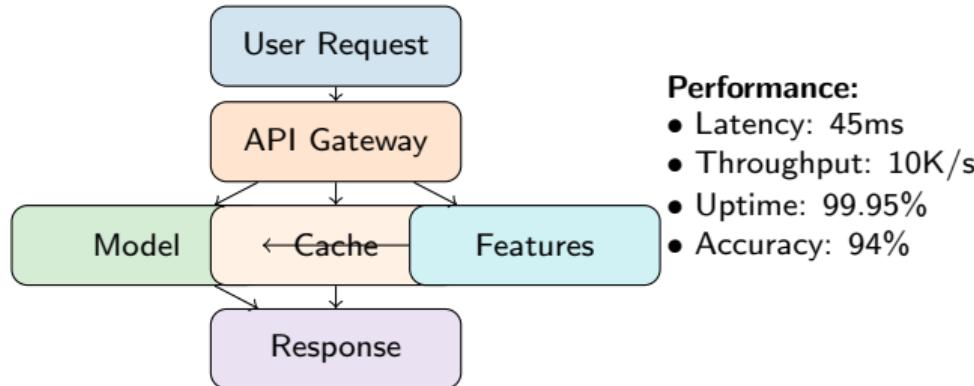
- ➊ **Summary:** High success likely
- ➋ **Factors:** Top 3 drivers shown
- ➌ **Details:** All features ranked
- ➍ **Counterfactual:** What-if analysis

User Benefits:

- Understand reasoning
- Identify improvements
- Validate with domain knowledge
- Learn patterns

Explanations increase adoption by 3x

Instant Intelligence at Scale



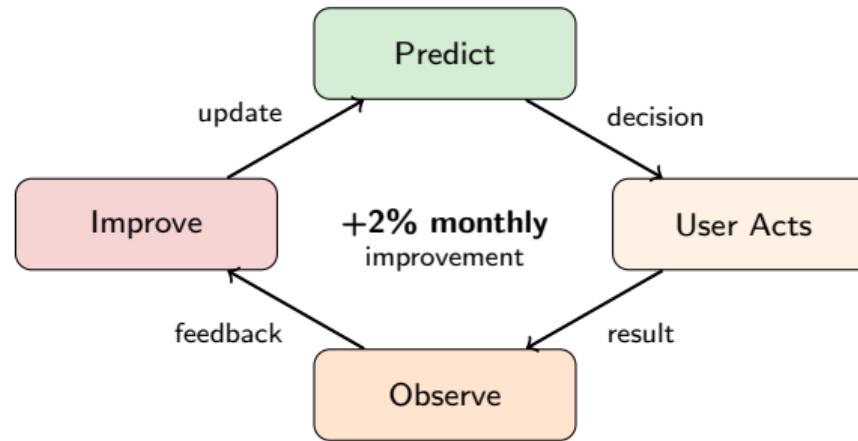
Critical Design Choices:

- Graceful degradation
- Fallback predictions
- Queue management
- Result caching

User Experience:

- Instant feedback
- Progress indicators
- Partial results
- Offline mode

The Feedback Loop



Implicit Feedback:

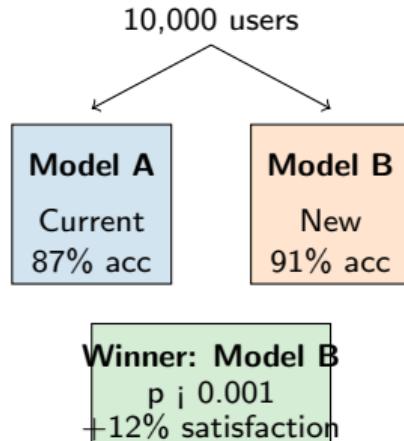
- Time on page
- Actions taken
- Selections made
- Features used

Explicit Feedback:

- Thumbs up/down
- Corrections
- Comments
- Ratings

"Every user interaction teaches the system"

A/B Testing with Intelligence



Test Metrics:

- Accuracy improvement
- User satisfaction
- Task completion
- Time to decision
- Error reduction

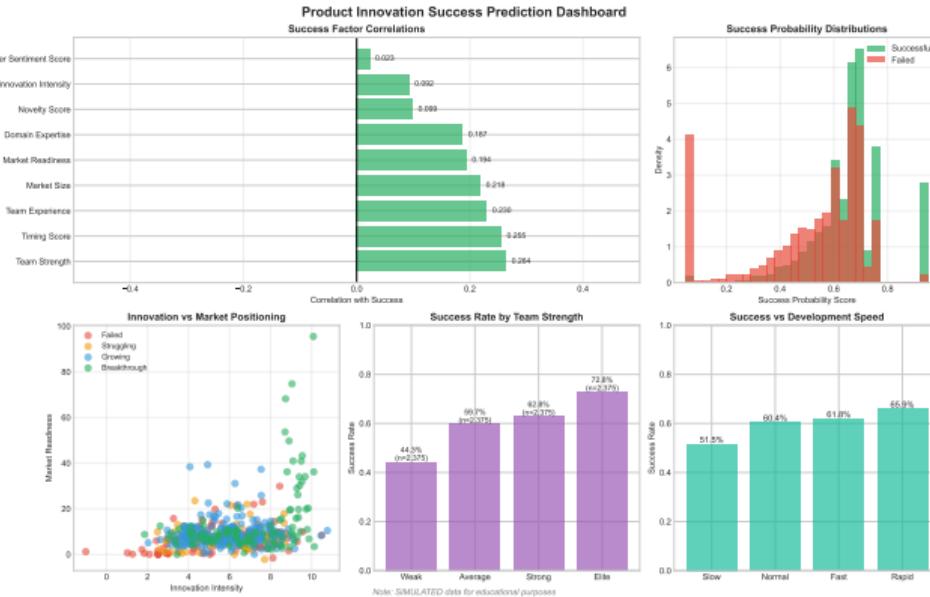
Best Practices:

- Random assignment
- Sufficient sample size
- Multiple metrics
- Guard rails
- Rollback plan

15% average improvement through systematic testing

Optimize Your Innovation Portfolio

Strategic Decision Support



Portfolio Intelligence:

- Risk distribution
- Success probabilities
- Resource allocation
- Timeline optimization
- Synergy detection

Recommended Mix:

- 20% Moonshots
- 50% Core innovation
- 30% Quick wins

ML-optimized portfolios show 40% better returns

Responsible Classification

Fairness Checks:

- Demographic parity
- Equal opportunity
- Individual fairness
- Counterfactual fairness

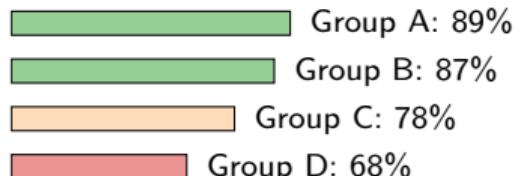
Bias Sources:

- Historical data
- Sampling bias
- Label bias
- Feedback loops

Mitigation Strategies:

- Diverse training data
- Fairness constraints
- Regular audits
- Human oversight

Fairness Dashboard

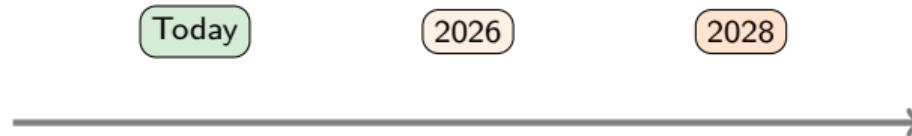


Alert: Disparity detected

Ethical Guidelines:

- Transparency first
- User consent
- Right to explanation
- Human appeal process
- Regular fairness audits

Where We're Heading



Rule-based
decisions

Adaptive
systems

Autonomous
innovation

Emerging Capabilities:

- Self-improving models
- Causal inference
- Few-shot learning
- Multimodal classification
- Quantum ML

Design Opportunities:

- Conversational AI interfaces
- Augmented decision making
- Predictive user needs
- Collaborative human-AI teams
- Ethical AI by design

"The best interface is no interface - just intelligence"

From Code to Human Impact

You've learned to bridge ML and UX:

Technical Mastery:

- Build accurate classifiers
- Deploy at scale
- Monitor performance
- Iterate based on data

Design Excellence:

- Create intuitive interfaces
- Build user trust
- Provide explanations
- Enable user control

Key Principles:

- ① Users don't care about algorithms
- ② Trust beats accuracy
- ③ Explanations drive adoption
- ④ Feedback improves everything
- ⑤ Ethics are non-negotiable

Remember:

Great ML empowers humans,
it doesn't replace them

Now: Let's practice!

Gradient Descent Optimization

Log-Likelihood Function:

$$\ell(\beta) = \sum_{i=1}^n [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

where $p_i = \frac{1}{1+e^{-\beta^T x_i}}$

Gradient:

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^n (y_i - p_i)x_{ij}$$

Update Rule:

$$\beta^{(t+1)} = \beta^{(t)} + \alpha \sum_{i=1}^n (y_i - p_i^{(t)})x_i$$

Convergence: When $\|\nabla \ell\| < \epsilon$ or maximum iterations reached

Regularization: Add penalty term $-\lambda \|\beta\|^2$ to prevent overfitting

Entropy and Information Gain

Entropy (Impurity Measure):

$$H(S) = - \sum_{c \in C} p_c \log_2(p_c)$$

where p_c is the proportion of samples in class c

Information Gain:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$$

Gini Impurity (Alternative):

$$Gini(S) = 1 - \sum_{c \in C} p_c^2$$

Example Calculation:

Parent node: 60 success, 40 fail

$$H(parent) = -0.6 \log_2(0.6) - 0.4 \log_2(0.4)$$

$$H(parent) = 0.971$$

After split:

Left: 50 success, 10 fail

Right: 10 success, 30 fail

$$IG = 0.971 - 0.811 = 0.160$$

Appendix: SVM and the Kernel Trick

Maximum Margin Optimization

Primal Optimization Problem:

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1$$

Dual Form (Using Lagrange Multipliers):

$$\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Kernel Trick: Replace $x_i^T x_j$ with kernel function $K(x_i, x_j)$

Common Kernels:

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (x_i^T x_j + r)^d$
- RBF (Gaussian): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
- Sigmoid: $K(x_i, x_j) = \tanh(\kappa x_i^T x_j + c)$

Decision Function:

$$f(x) = \text{sign} \left(\sum_{i \in SV} \alpha_i y_i K(x_i, x) + b \right)$$

Thank You!

Questions?

Remember: Everyone starts as a beginner!

innovation-ml-course@university.edu