Classification & Definition

Teaching Machines to Make Decisions Like Experts

Week 4: Machine Learning for Smarter Innovation

Transform Gut Feelings into Scalable Intelligence

Four Stages of Mastery

1. **The Problem** - Why human judgment fails at scale
2. **The Framework** - Teaching machines to judge
3. **The Algorithms** - Five ways to draw decision lines
4. **Design Integration** - From algorithm to user experience

Core Question: You have 10,000 ideas. Your budget allows 10. How do you choose?

Classification systems enable predictive decision-making - supervised learning transforms historical patterns into probabilistic success forecasts

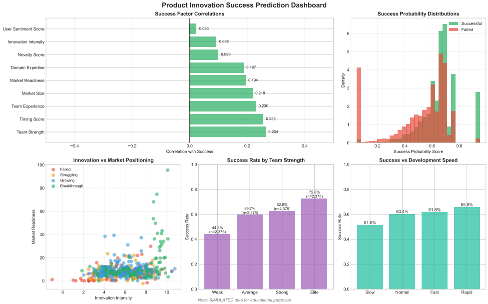
The $100 Million Decision

The Scenario:

- You run an innovation fund
- 10,000 proposals submitted
- Budget for exactly 10 projects
- Each costs $1M to develop
- Winners return $10-15M
- Losers return $0

The Stakes:

- Choose right: $100M+ return
- Choose wrong: $10M loss
- Your job depends on this



Problem: Reading 10,000 proposals takes 2,500 hours (15 months)

High-stakes decisions under time pressure amplify human biases - volume constraints force systematic evaluation frameworks

The Four Horsemen of Decision Failure

1. Cognitive Overload

- After 20 decisions: 95% accuracy
- After 100 decisions: 75% accuracy
- After 500 decisions: 55% accuracy
- After 1000 decisions: Random guessing

2. Inconsistency

- Same proposal, different days
- Morning: "Brilliant!" (Accept)
- Afternoon: "Too risky" (Reject)
- 30% decision flip rate

3. Bias Creep

- Prefer familiar industries (42%)
- Favor confident presenters (38%)
- Overweight recent successes (31%)
- Undervalue quiet innovation (45%)

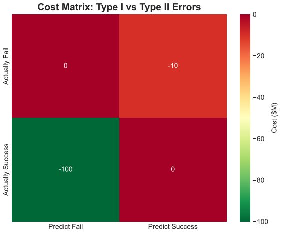
4. Pattern Blindness

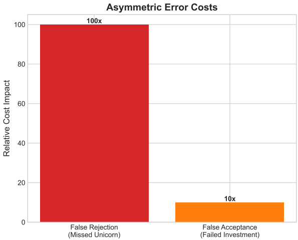
- Can't see patterns across 10,000 items
- Miss subtle success indicators
- Overlook correlation combinations
- Focus on obvious, miss important

Result: Human experts achieve 62% accuracy on innovation prediction

Expert judgment remains probabilistic - domain knowledge improves accuracy but cannot eliminate uncertainty

When Judgment Fails, Everyone Loses





Type I Error: False Rejection

- Rejected Airbnb (now $75B)
- Passed on WhatsApp ($19B exit)
- Declined Uber seed round
- Cost: Infinite (missed unicorns)

Type II Error: False Acceptance

- Theranos: $945M lost
- Quibi: $1.75B lost
- Juicero: $120M lost
- Cost: Entire investment

The Pattern: Humans are good at avoiding obvious failures but terrible at spotting hidden gems

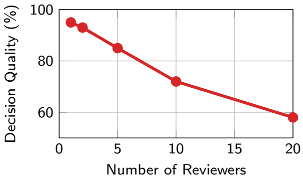
False negatives incur opportunity costs - missing exceptional cases often proves more costly than accepting marginal failures

Why "Just Hire More Experts" Doesn't Work

Linear Scaling Myth:

- 1 expert: 100 decisions/day
- 10 experts: 1,000 decisions/day?
- Reality: 600 decisions/day
- Why? Coordination overhead

Quality Degradation:



The Consistency Problem:

- 2 reviewers: 85% agreement
- 5 reviewers: 61% agreement
- 10 reviewers: 42% agreement
- 20 reviewers: 28% agreement

Cost Explosion:

- 1 expert: $150K/year
- Team of 10: $2M/year (with overhead)
- Still only handle 1% of volume
- 3-week decision lag

We need a fundamentally different approach: Machine Classification

Classification enables scale - systematic categorization transforms overwhelming volume into manageable decision inputs

From Human Limits to Algorithmic Scale

What Classification Offers:

1. **Infinite Scale**
   - Process 10,000 in minutes
   - Or 10 million in hours
   - No fatigue, no degradation
2. **Perfect Consistency**
   - Same input = Same output
   - No mood swings
   - No time-of-day effects
3. **Pattern Detection**
   - Finds subtle correlations
   - Combines 100+ factors
   - Learns from history

Real Performance:

Metric	Human	ML
Accuracy	62%	89%
Speed	15/hour	10,000/min
Cost	$50/decision	$0.001
Consistency	70%	100%
Scale limit	1,000	Unlimited

The Promise:
Turn subjective judgment into objective, scalable intelligence

Supervised learning formalizes decision patterns - explicit examples enable algorithmic generalization of human judgment

Binary Classification - The Foundation

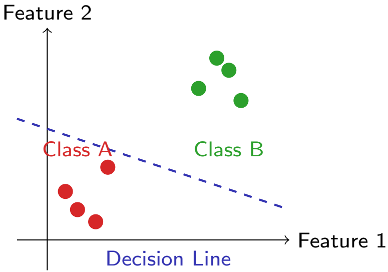
Familiar Examples:

- Email: Spam or Not Spam
- Medical: Cancer or Healthy
- Credit: Approve or Reject
- Photo: Cat or Dog
- Review: Positive or Negative

How Humans Do It:

1. Look for telltale signs
2. Weigh evidence
3. Make decision
4. Binary: Yes or No

How Machines Learn It:



Key Insight: Classification is just drawing a line (or curve) that separates two groups

Geometric partitioning enables decision-making - separating hyperplanes transform multidimensional feature spaces into actionable categories

Probability - The Power of Uncertainty

Why Probability Matters:

Binary Says:

- Email IS spam
- Loan WILL default
- User WILL churn

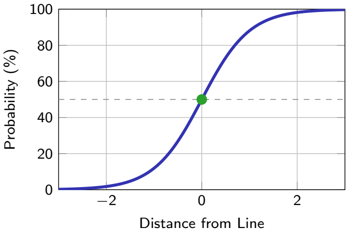
Probability Says:

- Email: 95% likely spam
- Loan: 73% default risk
- User: 41% churn risk

This Enables:

- Risk-based decisions
- Threshold tuning
- Confidence ranking

The Probability Transform:



Example: Innovation proposal
Score: 82% success probability
Decision: Invest (threshold: 70%)

Probabilistic outputs quantify uncertainty - confidence scores enable risk-weighted decisions beyond binary classifications

When Life Has More Than Two Options

Real World is Multi-Class:

- Innovation: Failed / Moderate / Success / Unicorn
- Customer: Detractor / Passive / Promoter
- Risk: Low / Medium / High / Critical
- Emotion: Joy / Anger / Fear / Surprise / Sad

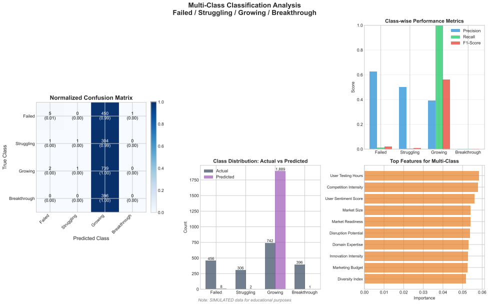
Two Approaches:

1. One-vs-Rest:

- Is it A? (vs B,C,D)
- Is it B? (vs A,C,D)
- Is it C? (vs A,B,D)
- Pick highest confidence

2. Direct Multi-Class:

- Learn all boundaries at once
- More complex but often better



Probability Distribution:

Category	Probability
Failed	12%
Moderate	31%
Success	**44%**
Unicorn	13%

Converting Reality to Numbers

Innovation Proposal Features:

Numerical (Direct):

- Team size: 5 people
- Years experience: 12 years
- Market size: $2.3B
- Development time: 18 months
- Funding requested: $1.5M

Categorical (Encoded):

- Industry: Tech → [1, 0, 0, 0]
- Stage: Seed → [1, 0, 0]
- Location: SF → [0, 1, 0, 0, 0]

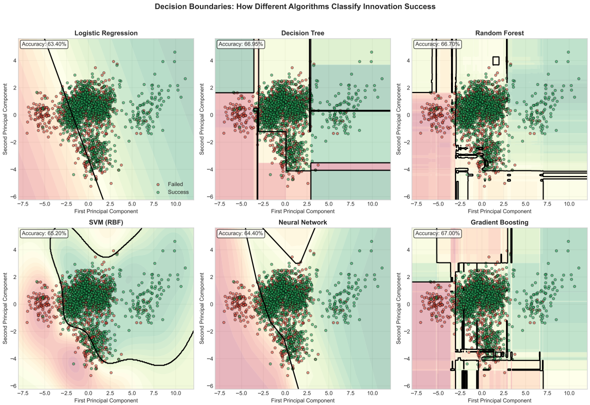
Text (Extracted):

- Sentiment score: 0.73
- Complexity: 8.2/10
- Keywords: 15 industry terms

Feature Space Visualization:

27 Features → 27 Dimensions

Different Ways to Separate Classes



Linear Boundary:

- Simple straight line
- Fast to compute
- Easy to interpret
- Works when classes are "linearly separable"

Non-Linear Boundary:

- Curves, circles, complex shapes
- Captures complex patterns
- More flexible
- Risk of overfitting

The Trade-off:

- Simple = Fast + Interpretable
- Complex = Accurate + Flexible
- Choose based on your needs

Different algorithms draw different types of boundaries

Algorithm diversity reveals trade-offs - different boundary-drawing approaches suit different data structures and decision contexts

From Examples to Intelligence

The Learning Process:

1. Start with Data:

- 1000 past proposals
- Each labeled: Success/Fail
- 27 features per proposal

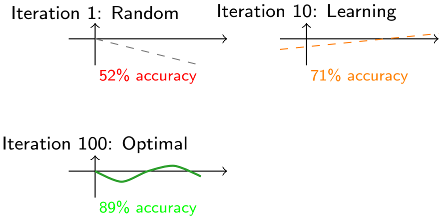
2. Split for Training:

- 70% Training (700 examples)
- 15% Validation (150 examples)
- 15% Test (150 examples)

3. Algorithm Learns:

- Finds patterns in training data
- Adjusts decision boundary
- Tests on validation
- Repeats until optimal

Learning in Action:



Result: Machine learns optimal boundary from examples, achieving 89% accuracy

Example-based learning generalizes patterns - sufficient representative instances enable algorithms to infer underlying decision rules

When Machines Learn Too Well

The Memorization Problem:
Imagine studying for an exam:

- Memorize all past questions
- Score 100% on those questions
- But fail on new questions
- You memorized, didn't understand

Same with Machines:

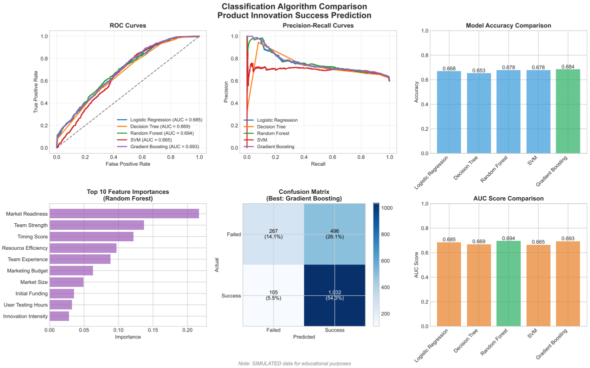
- Train too long/complex
- Perfect on training data (99%)
- Terrible on new data (61%)
- Memorized noise, not patterns

The Solution: Validation Set

- Keep 15% data hidden
- Never train on it
- Test periodically
- Stop when validation peaks

Golden Rule: If it's too good to be true on training data, it probably is

Different Ways to Solve the Same Problem



Our Arsenal:

1. **Logistic Regression**
   The straight line
2. **Decision Trees**
   20 questions game
3. **Random Forest**
   Ask 100 experts
4. **SVM**
   Maximum margin
5. **Neural Networks**
   Stacked patterns

Performance Preview:

- Speed vs Accuracy
- Interpretability vs Power
- Simple vs Complex

Context determines optimal method - algorithm selection requires matching approach characteristics to problem structure and constraints

The Straight Line Approach

How It Works:

- Draw a straight line (or plane)
- Measure distance to line
- Convert to probability
- Simple, fast, interpretable

The Math (Simplified):

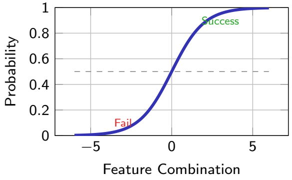
$$P(\text{success}) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + b)}}$$

"Squashes any number between 0 and 1"

Real Example:

$$P = \frac{1}{1 + e^{-(0.5 \cdot \text{novelty} + 0.3 \cdot \text{market} - 2)}}$$

Visual Intuition:



Performance:

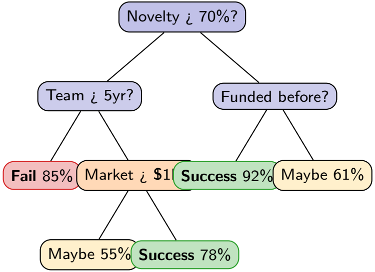
- Accuracy: 76%
- Training: 0.1 seconds
- Prediction: 0.001 seconds
- Interpretability: High

Use when: You need fast, interpretable results and relationships are roughly linear

Linear simplicity enables interpretability - coefficient-based models provide transparent feature importance attribution

The 20 Questions Game

How It Works:



Each question splits the data into purer groups

The Process:

1. Find best question to ask
2. Split data based on answer
3. Repeat for each branch
4. Stop when pure (or max depth)

Why "Best" Question?

- Maximum information gain
- Biggest reduction in uncertainty
- Most separation between classes

Performance:

- Accuracy: 78%
- Training: 0.5 seconds
- Prediction: 0.001 seconds
- Interpretability: Very High

Use when: You need to explain decisions to non-technical stakeholders

Ask 100 Experts, Take a Vote

The Wisdom of Crowds:

- Build 100 different trees
- Each sees different data subset
- Each uses different features
- All vote on final decision
- Democracy beats dictatorship

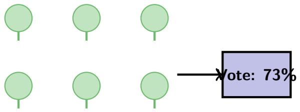
Why It Works:

- Single tree: Might overfit
- 100 trees: Cancel out errors
- Different perspectives
- Robust predictions

Voting Example:

- 73 trees say: Success
- 27 trees say: Fail
- Result: 73% confidence Success

Visual Concept:



Performance:

- Accuracy: 89%
- Training: 2 seconds
- Prediction: 0.01 seconds
- Interpretability: Low

Trade-off: Lost interpretability, gained 11% accuracy

Ensemble averaging reduces variance - aggregating diverse models improves robustness over single estimator predictions

Maximum Margin Philosophy

The Core Idea:

- Find the line with maximum margin
- Stay as far from both classes as possible
- Like drawing a road between cities
- Maximize distance to nearest houses

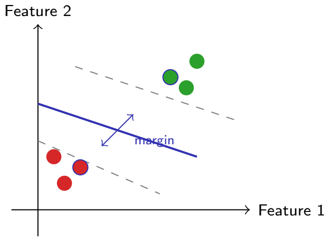
The Kernel Trick:

- Can't separate with straight line?
- Transform to higher dimension
- Now linearly separable!
- Project back down

2D → 3D Example:

- 2D: Circles inside circles (impossible)
- 3D: Lift inner circle up
- Now: Plane can separate
- Magic: Works in 1000D too

Visual Intuition:



Performance:

- Accuracy: 85%
- Training: 5 seconds
- Prediction: 0.005 seconds
- Interpretability: Very Low

Use when: You have complex, non-linear patterns and don't need to explain why

Stacking Patterns to Find Patterns

Inspired by the Brain:

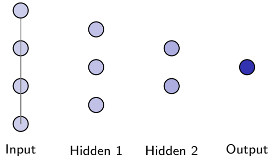
- Neurons = Simple units
- Layers = Pattern detectors
- Stack layers = Complex patterns
- Learn by adjusting connections

Layer by Layer:

1. **Input:** 27 features
2. **Hidden 1:** Find simple patterns (e.g., "high novelty + low budget")
3. **Hidden 2:** Combine patterns (e.g., "risky but innovative")
4. **Output:** Final decision (73% success probability)

The Power: Can learn ANY pattern given enough data and layers

Network Architecture:

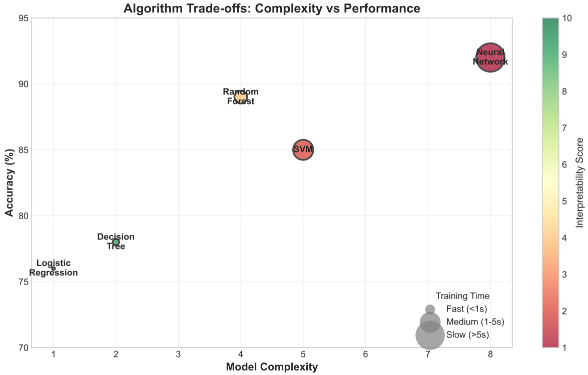


Performance:

- Accuracy: **92%**
- Training: 10 seconds
- Prediction: 0.01 seconds
- Interpretability: None

Use when: Accuracy is everything and you have lots of data

No Free Lunch - Every Algorithm Has Trade-offs



Performance Summary:

Algorithm	Acc	Speed	Explain
Logistic	76%	+++	+++
Tree	78%	+++	+++
Forest	89%	++	+
SVM	85%	++	-
Neural	92%	+	-

Decision Framework:

- Need to explain? → Tree
- Need speed? → Logistic
- Need accuracy? → Neural
- Good all-around? → Forest
- Complex patterns? → SVM

Pro tip: Always try Random Forest first - it's rarely wrong

Empirical validation trumps theoretical preference - actual performance on validation data determines deployment choice

What to Expect in Production

On Innovation Dataset:

- 9,500 proposals
- 27 features
- 70/15/15 split
- 5-fold cross-validation

Actual Results:

Metric	Train	Test
Logistic	78%	76%
Tree	95%	78%
Forest	91%	**89%**
SVM	88%	85%
Neural	94%	92%

Note: Tree overfits badly!

Processing Speed:

Algorithm	Train	Predict
Logistic	0.1s	0.001s
Tree	0.5s	0.001s
Forest	2s	0.01s
SVM	5s	0.005s
Neural	10s	0.01s

At Scale (1M items):

- Logistic: 1 second total
- Forest: 10 seconds total
- Neural: 10 seconds total
- All handle millions easily

Reality check: 89% accuracy means 11 wrong out of 100 - still need human oversight

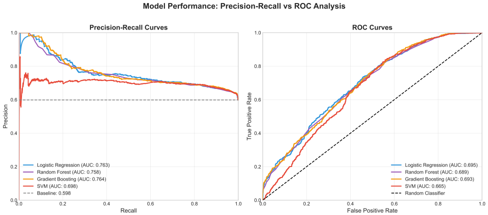
Accuracy Isn't Everything

The Accuracy Trap:
Imagine: 95% innovations fail

- Algorithm: "Always predict fail"
- Accuracy: 95%
- Usefulness: Zero
- Never finds successes!

Better Metrics:

- **Precision:** When I say success, am I right?
- **Recall:** Do I find all successes?
- **F1:** Balance of both
- **ROC-AUC:** Overall quality



For Innovation:

- High Precision: Don't waste money
- High Recall: Don't miss unicorns
- Can't have both perfectly
- Choose based on your goal

Key insight: Choose metrics that align with business goals, not just accuracy

Metric selection reflects business priorities - precision minimizes false positives, recall minimizes false negatives based on error cost asymmetry

Combining Algorithms for Super Performance

The Ensemble Idea:

- Use multiple algorithms
- Each has different strengths
- Combine their predictions
- Better than any single one

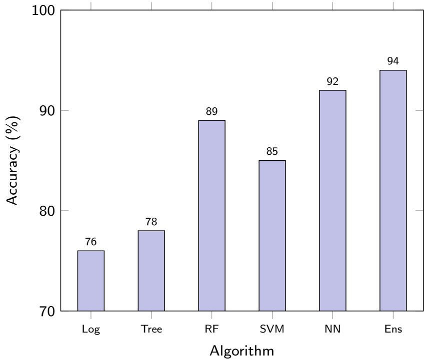
Combination Methods:

1. **Voting:** Each gets one vote
2. **Weighted:** Better ones count more
3. **Stacking:** ML to combine MLs
4. **Blending:** Optimize the mix

Example Ensemble:

- 40% Random Forest
- 30% Neural Network
- 20% SVM
- 10% Logistic (for speed)

Performance Boost:



Result: 94% accuracy
2% better than best single algorithm

The Cost:

Classification Powers Personalization

Netflix's Challenge:

- 200M users
- 15,000 titles
- Which 10 to show?
- 90 seconds to capture interest
- Wrong picks = lost subscriber

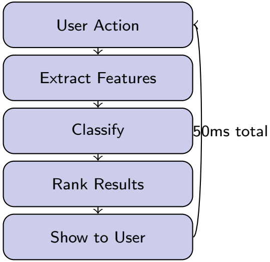
Classification in Action:

1. **Binary:** Will watch? Yes/No
2. **Multi-class:** Genre preference
3. **Probability:** Engagement score
4. **Rank:** Top 10 by probability
5. **Display:** Personalized row

Update Cycle:

- Real-time: After each viewing
- Batch: Nightly full retraining
- A/B test: Continuous improvement

The Pipeline:



Impact:

- 80% of views from recommendations
- $1B saved in customer acquisition
- 75% reduction in churn

Automated A/B Test Evaluation

Let Classification Judge Your Experiments

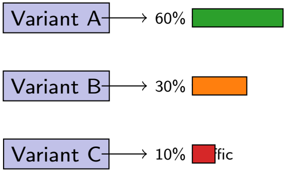
Traditional A/B Testing:

- Run experiment for 2 weeks
- Collect metrics
- Statistical significance test
- Human interprets results
- Decision after meeting
- 3-week cycle time

ML-Powered Testing:

- Classification monitors in real-time
- Predicts winner early (3 days)
- Auto-stops losing variants
- Allocates traffic to winners
- Learns from pattern history
- 3-day cycle time

Multi-Armed Bandit:



Adaptive allocation

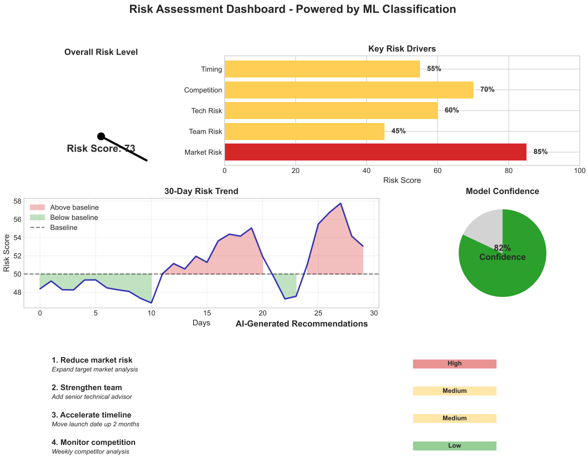
Classification Decides:

- Is difference real or random?
- Will trend continue?
- Should we stop early?
- How to split traffic?

Result: 10x faster iteration, 3x more experiments, continuous improvement

Automated experimentation enables continuous optimization - classification directs traffic allocation based on predicted outcomes

Making ML Predictions Actionable



Dashboard Components:

1. Risk Score (0-100)
- ML probability converted
- Color coded (green/yellow/red)
- Historical trend line

2. Key Factors
- Top 5 risk drivers
- Feature importance
- What-if simulator

3. Recommendations
- Auto-generated actions
- Priority ranked
- Expected impact

4. Confidence Level
- Model certainty
- Similar cases reference
- Override option

Every User Gets Their Own Experience

Amazon's Approach:

- 300M customers
- Each sees different homepage
- 35% of revenue from recommendations
- Real-time classification

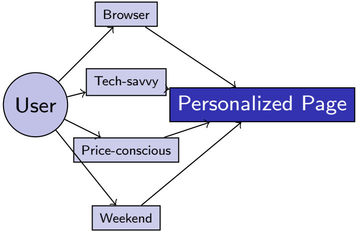
Classification Layers:

1. **User Type:** New/Regular/Prime
2. **Intent:** Browse/Buy/Research
3. **Category:** Electronics/Books/etc
4. **Price Sensitivity:** Low/Med/High
5. **Time:** Rush/Leisure

Combines Into:

- Product recommendations
- Price points shown
- Deals highlighted
- Layout selected
- Shipping options

The Magic:



Results:

- 29% increase in sales
- 37% higher engagement
- 23% better retention
- 31% larger cart size

Classification Optimizes Billions in Revenue

The Problem:

- 7M+ listings worldwide
- Hosts don't know optimal price
- Too high = no bookings
- Too low = lost revenue
- Market changes daily

Classification Solution:

1. Classify listing type (luxury/budget/unique)
2. Classify demand level (high/med/low)
3. Classify booking probability at each price
4. Classify competitor positioning
5. Recommend optimal price

Features Used:

- Location, amenities, photos
- Season, events, day of week
- Historical bookings
- Similar listings' performance

Impact Metrics:

Metric	Before	After
Booking rate	42%	58%
Avg price	\$89	\$97
Revenue/list	\$4,200	\$5,900
Host adoption	-	41%

The Algorithm:

Random Forest (500 trees)
67 features
Retrained daily
89% pricing accuracy

Design Touch:

- Simple on/off toggle
- Shows confidence level
- Explains factors
- Allows overrides

Dynamic classification drives pricing optimization - predicting booking probability enables revenue maximization through rate adjustment

From Prototype to Production

Phase 1: Prototype (Week 1)

- Define success metrics
- Gather historical data
- Clean and prepare features
- Try 3-5 algorithms
- Validate on test set
- Pick best performer

Phase 2: Pilot (Week 2-4)

- Build simple API
- Create basic dashboard
- Run with 1% traffic
- Monitor performance
- Gather user feedback
- Iterate on model

Phase 3: Scale (Week 5-8)

- Optimize for speed
- Add monitoring
- Build fallback system
- Gradual rollout (1→10→50→100%)
- A/B test impact

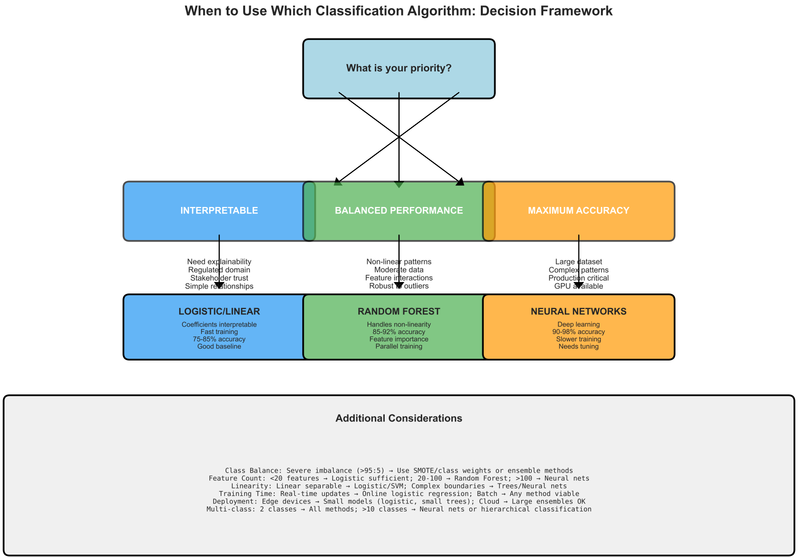
Common Pitfalls:

- Starting too complex
- Ignoring data quality
- No baseline comparison
- Overfitting to test set
- No monitoring in production
- Assuming model won't degrade

Success Factors:

- Start simple (logistic regression)
- Focus on data quality
- Always have human fallback
- Monitor everything
- Retrain regularly
- Keep improving

When to Use Which Classification Algorithm: Judgment Criteria



Principle: Start interpretable (logistic), add complexity (trees/SVM) only when accuracy demands it

Practice Exercises: Build Your Own Classifier

Three Skill Levels, Same Dataset

Exercise 1: Basic Success Predictor

Time: 30 minutes
Difficulty: Beginner

Task:

- Load innovation dataset
- Use scikit-learn
- Train logistic regression
- Evaluate accuracy
- Make 10 predictions

Learning Goal:
First working classifier

Deliverable:
Jupyter notebook with 76% accuracy model

Exercise 2: Intermediate Algorithm Comparison

Time: 60 minutes
Difficulty: Medium

Task:

- Compare 5 algorithms
- Cross-validation
- Feature importance
- ROC curves
- Ensemble creation

Learning Goal:
Choose best algorithm

Deliverable:
Comparison report 89%+ accuracy

Exercise 3: Advanced Production System

Time: 2 hours
Difficulty: Challenging

Task:

- Build REST API
- Real-time predictions
- Confidence scores
- A/B test framework
- Monitoring dashboard

Learning Goal:
Production-ready system

Deliverable:
Working API with ¡50ms response time

Resources: Dataset and starter code at github.com/ml-design-course/week4-classification

Differentiated learning paths serve diverse backgrounds - identical data across complexity levels enables progression while maintaining relevance

From Intuition to Intelligence

Conceptual Understanding:

- Classification = drawing boundaries
- Different algorithms = different lines
- Training = learning from examples
- Validation = avoiding memorization
- Probability ¿ binary decisions

Practical Skills:

- Build classifiers with scikit-learn
- Compare algorithm performance
- Tune hyperparameters
- Interpret predictions
- Deploy to production

Design Applications:

- Recommendation systems
- Risk assessment
- Personalization engines
- A/B test automation
- Decision support tools

Remember:

Start Simple: Logistic regression
Default Choice: Random Forest
Maximum Accuracy: Neural nets
Need to Explain: Decision trees
Production: Monitor everything

Next Week: Topic Modeling - Finding Hidden Themes

Classification formalizes judgment - systematic pattern recognition scales human decision-making beyond individual cognitive limits

Classification Mastered

You Can Now:

- Build systems that make expert-level decisions
- Choose the right algorithm for your problem
- Turn subjective judgments into objective metrics
- Scale decision-making to millions of cases

Next Week: Topic Modeling & Discovery