VAMSHI JANDHYALA

When to use decision models

How to choose use cases, set thresholds and evaluate results

Executive summary

- **Many high-volume AI workflows are bounded decisions.** Route this ticket, check this claim, match these records: every acceptable answer can be listed before the input arrives.
- **A decision model returns a probability for each listed answer.** It never writes text, so it never returns an answer you did not offer, and every answer carries a number saying how much to trust it.
- **Three decision shapes cover many operational decisions:** categorical (pick one), binary (true or false) and ordinal (a position on a scale).
- **The product behaviour is a policy you choose.** Thresholds come from the cost of an error against the cost of a review, and uncertain cases need somewhere safe to go.
- **Do not automate until performance has been measured on representative cases,** with the accuracy of the automated share, calibration, important slices and drift all checked, and checked again after every change.

How to read this. Parts 1 to 4 build the model, Part 5 works three business cases in full, Part 6 compares alternatives, Part 7 covers evaluation, and Part 8 is a pilot playbook. Jev, from TypeSafe, is the example model; its product specifics are in the dated appendix. Numbers marked *recorded* are real model outputs; numbers marked *illustrative* are invented to show a method.

PART 1

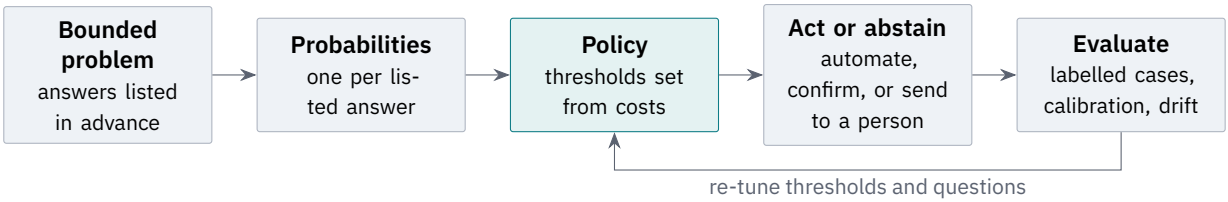
Many high-volume AI workflows are bounded decisions, and a model built only to decide can make them cheaper, faster and measurable

A customer writes to an online shoe shop: *"Shoes arrived two weeks late and in the wrong size. Also I see two charges of $120 on my card. What are you going to do about this?"* The shop's software needs to know which team owns the message, what went wrong, what the customer wants, and how annoyed they are. Every acceptable answer could have been written down before the message arrived. The same holds for routing tickets, flagging transactions, checking citations, matching records and choosing which tool an agent calls next.

The common approach asks a model that writes text to reply in a fixed format, then parses the reply. That carries four costs that grow with volume. The reply can be held to your list of labels, but it rarely comes with a calibrated probability for every option on that list. Uncertainty is hidden: "billing" reads the same whether the model was sure or guessing. Writing is slow and costly next to scoring a fixed list. And free text must be interpreted before it can be checked against the right answer.

A *decision model* can reduce all four. It reads the evidence, scores every answer you listed, and returns a probability for each. Code then applies a policy: act, confirm, or send the case to a person.

Exhibit 1
The decision-model workflow at a glance



So what: The model supplies judgement; everything with consequences, including when to abstain, is a policy in your code that evaluation keeps honest.

PART 2

Decide first whether the workflow is a bounded decision at all

Exhibit 2
Should this be a decision-model workflow?

Use it when all of these hold	Do not use it when any of these holds
1. All acceptable answers can be listed in advance	The answer must be created, not selected
2. The evidence needed for the decision is available as text	The option set changes unpredictably
3. Correctness can be labelled consistently	Reviewers cannot agree on the right answer
4. Uncertain cases have a safe destination	Errors cannot be detected or contained
5. Volume, latency or reliability justify automation	Deterministic code can do the task
6. The workflow can be monitored after launch	

So what: If any condition on the left fails, fix that first or choose another approach (Part 6); the model will not compensate for it.

Bounded decisions show up across the business: routing and escalation in customer operations; coverage and documentation checks in claims; alert triage in security; pay, hold or dispute in finance operations; required clauses and supported citations in compliance; record matching and classification in data quality; re-ranking in search; and, inside AI systems, checking another model's output or choosing which tool to call.

PART 3

Three decision shapes cover many operational decisions

Every request has two parts. The **state** is the evidence: a message, a claim file, a pair of records. The **questions** say what to decide, each with an instruction and a short description of every possible answer. The model sees those descriptions and nothing else about your intent, so they are where your domain knowledge goes.

Exhibit 3
Three decision shapes

Shape	Asks	Returns	In code
Categorical	Which of these?	A probability per option, summing to 1	A branch per option
Binary	Is this true?	The probability of yes	An `if` with a threshold
Ordinal	Where on this scale?	A probability per level, and a position between levels	A number to sort or threshold

So what: Pick the shape by what your code will do next: branch, check, or rank.

Some decision models also return a *confidence* for categorical and ordinal answers: one number from 0 to 1 summarising how concentrated the probabilities are, 0 for an even spread and 1 when all the weight sits on one answer. Appendix B gives a formula consistent with Jev's published outputs and maps its terms to these shapes.

PART 4

Probabilities become product behaviour through a policy, and the policy is a cost calculation

Exhibit 4
Recorded answers for the shoe-shop ticket: five categorical questions in one request

Question	Probabilities	Confidence
Department	returns 0.61, billing 0.35, shipping 0.04	0.42
Return reason	wrong size 1.00, four others 0.00	1.00
Shipping issue	delayed 0.74, other 0.26, three others 0.00	0.67
Resolution wanted	refund 0.40, replacement 0.34, exchange 0.24, information 0.02	0.20
Tone	frustrated 0.84, angry 0.16, calm 0.00	0.76

So what: Two teams are involved and the customer's wish is unclear, and the numbers show both.

Reading it. Returns wins at 0.61, but billing has 0.35: the model has noticed the double charge. The gap between the top two, the *margin*, is 0.61 − 0.35 = 0.26, small enough that the ticket belongs to two teams. Resolution spreads over three answers with none above 0.40, because the customer never says what they want. The shipping question was asked speculatively and is ignored, because the department is not shipping.

Acting on it. Four rules: give the ticket to the top department (returns); copy any other department above 0.25 (billing, since 0.35 > 0.25); use only answers that apply; and if confidence is below 0.5, ask instead of guessing (resolution is 0.20, so ask the customer).

Three habits make numbers like these safe to act on. For a binary check, a value near 0.5 means the model cannot tell, so use a band instead of one cut-off: below 0.30 is no, above 0.70 is yes, and in between goes to a person. For an ordinal decision, the score is a weighted position: probabilities 0, 0.57, 0.43 on levels 0, 1, 2 give 0 × 0 + 1 × 0.57 + 2 × 0.43 = 1.43, a position between levels 1 and 2 and not a measurement of anything. And set every threshold from costs.

Setting a threshold from costs

Act or review. If a wrong automatic action costs E and a review costs R, and p is the probability the answer is right, act alone when $(1 - p)\,E < R$, that is when $p > 1 - R/E$. Example: a wrong refund costs £40, a review £2, so act alone above $1 - 2/40 = 0.95$. At $p = 0.97$ the expected error cost is 0.03 × £40 = £1.20, less than £2, so act; at $p = 0.93$ it is £2.80, so review.

Two kinds of error. When the choice is between two actions, such as flag or pass, and a false alarm costs C_{FP} while a miss costs C_{FN}, flag when $p > C_{FP}/(C_{FP} + C_{FN})$. Example: a needless fraud review costs £5 and a missed fraud £500, so flag above $5/505 = 0.0099$, about 1 per cent.

Caveat. These rules assume the probabilities are calibrated, the costs are reasonably estimated, and an error can be represented by an average cost. Validate all three before automating.

PART 5

Three worked cases show how to design the questions, read the numbers and decide

Each case follows six steps: problem, design, what the model returned, reading, rule, evaluation. Worked examples use Jev; implementation details appear in Appendix B.

Case A Categorical: classifying companies by industry

PROBLEM Assign every company to one of 75 industry groups, for onboarding, risk reporting and sales coverage, from the company's own long description of its business.

DESIGN The state is that description (700 to 2,200 words). One categorical question over the 75 groups, each described by the industries inside it. The groups roll up into 10 broad divisions.

RETURNED

Exhibit 5
Recorded confidences for four companies

Business, as classified	Confidence	Favourite's probability
Chemicals and allied products	1.00	1.00
Life, accident and health insurance	1.00	1.00
Search, detection and navigation equipment	0.22	0.23
Engineering, accounting and research services	0.29	0.30

So what: Two companies are clear-cut; two are close calls between neighbouring groups.

READING The model is only 0.22 confident about the third company. What does that mean?

- Convert confidence back to the favourite's probability (Appendix B): $(74 \times 0.22 + 1) \div 75 = 17.28 \div 75 = 0.23$.
- A blind guess over 75 groups would give each $1 \div 75 = 0.013$, about 1 per cent.
- So the favourite got about 17 times a blind guess. The model has a clear favourite, split between similar groups.

In short: low confidence here means "a close call between neighbours", not "no idea".

RULE Confidence 0.9 or more: use the specific group. Otherwise use its broad division, *manufacturing* instead of *search and navigation equipment*. A broad answer that is right beats a specific one that is wrong.

EVALUATION

Exhibit 6
Accuracy at the operating point, recorded on 60 companies

	Right / cases	Accuracy	Plausible range
All, forced to name a group	39 / 60	65%	52 to 76%
Confident (≥ 0.9), group reported	27 / 30	90%	74 to 97%
Unsure, group reported	12 / 30	40%	25 to 58%
Unsure, division reported	21 / 30	70%	52 to 83%
With the rule	48 / 60	80%	68 to 88%

Plausible range: 95 per cent interval for the true rate given the sample size. The 60 companies were chosen so each description supports its registered industry, so real-world accuracy will be lower.

So what: The rule turns 65 per cent into 80 per cent useful answers from the same responses; with only 30 cases per group, the true figures could differ by 15 points either way.

Case B Binary: checking an insurance claim before payment

PROBLEM An automated note has marked a motor claim "approved, pay full amount". Should it be paid?

DESIGN The claim is deliberately awkward. The policy excludes track driving, and the damage happened at a track-day event but in the car park while stationary; a rental car is claimed without rental cover; a police report is required for collisions over $2,000 and none is attached. Fourteen binary checks go in one request, each phrased so that yes means the checked thing is true: *Is the loss covered? Does an exclusion apply? Is the documentation sufficient? Is the rental eligible? Are there fraud indicators?* The arithmetic stays in code.

Exhibit 7
Calculations done in code before the model's answers are used

Bumper $1,700 + paint $800 + sensors $450 + rental $300	$3,250 claimed
Within the $10,000 limit? 3,250 ≤ 10,000	yes
Police report required? 3,250 > 2,000, none attached	missing
Less rental (not covered) and the $500 deductible: 3,250 – 300 – 500	$2,450 maximum

So what: Even if the loss is covered, the correct payment is at most $2,450, not the $3,250 approved.

RETURNED The same claim and questions were sent fifteen times, with a fresh irrelevant identifier added to each request. The coverage check came back between 0.43 and 0.53, and the exclusion check between 0.53 and 0.62 (recorded).

READING Each answer is the probability of yes: near 1 means sure it is yes, near 0 sure it is no, near 0.5 cannot tell. Coverage sits in the middle, because whether a parked car at a race-track event counts as "track driving" is a genuine judgement call; a claims handler would hesitate too. A single cut-off at 0.5 would give opposite decisions on identical files: 0.49 on one run, 0.51 on the next.

RULE Use the three bands: below 0.30 no, above 0.70 yes, in between a person decides. When could payment happen with no person involved? If a wrong payment loses $2,450 and a review costs $30 (illustrative), the chance that the payment decision is wrong must be below 30 ÷ 2,450 = 0.012, so the decision must be right with probability above 1 – 0.012 = 0.988. That applies to the final decision, not to each check: fourteen independent checks each right 98.8 per cent of the time are all right only $0.988^{14} \approx 0.85$ of the time, and real checks are rarely independent. So evaluate the whole payment policy on labelled claims and automate only if its final decisions clear 0.988. This claim goes to a handler, with the ceiling and the missing report already worked out.

EVALUATION Across the fifteen runs the answers moved by about 0.01 on average, and the two that moved most stayed inside the uncertain band every time, so the decision never changed. Calibration is checked on labelled claims, as in Part 7.

Case C Ordinal: deciding whether two records are the same thing

PROBLEM Two product catalogues overlap, and a first pass found 450 candidate pairs. Each must be merged, left apart or sent to a curator. A wrong merge is expensive to undo; a missed one only leaves a duplicate.

DESIGN The state holds both records: name, maker, style, strength. One ordinal question with three levels written as outcomes: 0 *different products* (leave apart), 1 *related but possibly not the same*, such as a variant (curator), 2 *the same product* (merge). Binary checks on name, maker and style tell the curator which field disagrees; strength is a number, compared in code.

RETURNED

Exhibit 8
Recorded scores for four candidate pairs

Pair	Score	Confidence	Top level's probability	Outcome
Identical product	1.94	0.92	0.95	merge
Two different products	0.03	0.95	0.97	leave
Same name and maker, style worded differently	1.30	0.27	0.51	curator
A product and its flavoured variant	1.10	0.77	0.85	curator

So what: Both curator cases have similar scores. The confidence shows how settled each level is; the companion checks show which attributes caused the ambiguity.

READING The score says where a pair sits on the 0 to 2 scale; the confidence says how sure the model is about the level.

- *Flavoured variant*: score 1.10 with 0.85 on one level, the middle one. The model is confident these are related but not the same.
- *Style worded differently*: score 1.30 but only 0.51 on its top level (Appendix B: (2 × 0.27 + 1) ÷ 3). The model is torn between "related" and "same".

A score of 1.30 does not mean "65 per cent the same". It is an average position on the scale.

RULE Cut points halfway between levels: below 0.5 leave apart, 0.5 to 1.5 curator, above 1.5 merge. Because a wrong merge costs most, raise the upper cut point if merges prove unreliable.

EVALUATION Of 450 pairs, 360 were left apart, 40 merged and 50 sent to the curator, so 400/450 = 89 per cent needed no person. What matters most is merge accuracy on labelled pairs: 39 correct of 40 would be 97.5 per cent (illustrative). Sensitivity matters too: 47 pairs sat within 0.1 of the lower cut point and 9 of the upper, so nudging the lower one changes the curator's workload far more.

PART 6

Decision models are one option among several, and each alternative wins somewhere

Exhibit 9
Alternatives to consider

Approach	Wins when	Watch out for
Deterministic rules	The logic can be written down exactly	Brittle with messy language
Trained classifier	Categories are stable and thousands of labelled examples exist	Retraining whenever categories change
Embeddings and retrieval	The task is finding similar items, not choosing a label	Similarity is not the same as the right answer
Writing model, structured output	Volume is low, or reasoning or new text is needed	Cost, latency, no calibrated probability for each option
Human review	Stakes are high, cases rare, or judgement contested	Cost, speed, inconsistency between reviewers
Decision model	Answers are bounded, labels are scarce, volume or speed matter	Calibration must be verified; weak at arithmetic and multi-step reasoning
Hybrid	Most real systems: rules first, a model for judgement, people for the uncertain rest	More parts to monitor

So what: Most good systems are hybrids: code for what can be computed, a model for bounded judgement, people for what remains uncertain.

PART 7

Measure on your own representative cases before automating, and keep measuring after launch

An *eval* is a repeatable test: fixed inputs, a reference answer for each, and a scoring rule. Vendors' evals often measure agreement with larger models on tasks the vendor wrote; where those models are wrong, agreeing scores as right. The eval that governs a deployment uses cases drawn from the real operating mix, labelled by the people accountable for the decision, with a slice held back while wording and thresholds are tuned.

Exhibit 10
The evaluation scorecard

Measure	How to compute it	What it tells you
Automation rate	Share of cases above the threshold	How much work leaves people
Accuracy at threshold	Right ÷ automated, with the raw counts	Whether the automated share is safe
Error costs	False positives and false negatives, costed separately	Whether the threshold is set right
Calibration	Per band: share right against average probability	Whether the numbers mean what they say
Slices	The above for each important segment, especially rare, costly ones	Where averages hide failures
Consistency	Share of repeat runs giving the same answer	Whether decisions are stable
Baseline and drift	Today's process, and the same measures month by month	What improved, and what is changing

So what: Report every percentage with its raw counts, and never judge on the overall average alone.

How many cases. The number depends on the error rate and the precision you need. To estimate an error rate near 5 per cent to within 2 percentage points with 95 per cent confidence:

$$n = \frac{1.96^2 \times 0.05 \times 0.95}{0.02^2} = \frac{3.84 \times 0.0475}{0.0004} \approx 457.$$

That is per slice you care about. A 75-way classification needs enough cases in each important class, which is why Case A's 60 companies illustrate a method rather than justify a deployment.

Calibration. Group labelled cases by the probability the model gave, and compare each group's hit rate with its average probability.

Exhibit 11
A calibration check

Probability band	Right / cases	Share right	Average probability
0.5 to 0.7	35 / 60	0.58	0.61
0.7 to 0.9	87 / 110	0.79	0.81
0.9 to 1.0	198 / 230	0.86	0.96

Illustrative counts, not a measurement of Jev.

So what: The model is overconfident exactly where automation happens.

In the top band the model said 0.96, so it expected about $230 \times 0.04 = 9$ mistakes, but it made $230 - 198 = 32$: a real error rate of $32 \div 230 = 14$ per cent. The £40 refund rule pays automatically above 0.95, trusting that it is wrong at most 5 per cent of the time. At 14 per cent, each automatic refund risks $0.139 \times £40 = £5.56$, more than the £2 a review costs: £3.56 lost per case, about £820 across these 230. Raise the threshold until the real hit rate in the top band exceeds 95 per cent, or treat 0.96 as 0.86 in the cost calculation.

Consistency and monitoring. Send the same cases several times: 13 identical answers in 15 runs is 13 ÷ 15 = 0.87 agreement. In a recorded test of eight moderation questions run fifteen times, each with a fresh irrelevant identifier, the most common answer repeated 90.8 per cent of the time; treating answers below a 0.60 top probability as uncertain raised agreement to 99.2 per cent while still automating 74.2 per cent. After launch, log every decision's probabilities, threshold and model version, keep sampling automated decisions for review, and re-evaluate whenever the model, the questions, the options, the policy or the mix of inputs changes.

PART 8

Pilot on one narrow, reversible workflow before scaling

Exhibit 12
The pilot playbook

	Step	Done when
1	Choose one narrow, reversible workflow	Mistakes can be caught and undone
2	Define the options and the abstention route	Every answer, including "unsure", has an owner
3	Label representative historical cases	Enough per important slice (Part 7)
4	Record the current process as a baseline	Its accuracy, cost and speed are known
5	Run the model without affecting users	Shadow decisions are logged beside real ones
6	Choose thresholds from error and review costs	Each threshold has a stated cost rationale
7	Launch with human review	Reviewers see the model's answer and probability
8	Monitor accuracy, automation, calibration, slices and drift	A dashboard exists and has an owner
9	Revalidate after any change	Model, options, policy or input mix

So what: Understanding becomes value only through a pilot that measures before it automates.

Much of what products ask of AI is a choice among answers that could have been listed in advance. Treated that way, it becomes testable with a labelled set and the arithmetic in this report, whichever model sits underneath.

APPENDIX A

Formulas

Formula sheet

Margin	first answer's probability − second's
Ordinal score	∑(level × probability)
Act or review	act when $p > 1 - R/E$
Flag or pass	flag when $p > C_{FP}/(C_{FP} + C_{FN})$
Accuracy at threshold	right ÷ automated
Sample size	$n = 1.96^2\, p(1-p)/m^2$ for error rate p within $\pm m$
Agreement	repeats ÷ runs

APPENDIX B

How the framework maps to Jev

Product snapshot as of September 2026, Jev versions 1.12 and 1.13.

This report describes decision models in vendor-neutral terms. Jev is one implementation of the framework. The mapping below reflects the product versions in TypeSafe's published examples and may change over time.

Exhibit 13
Vendor-neutral concepts and their Jev equivalents

General concept	Jev term	Output
Evidence supplied for a decision	State	Text examined by every question
Categorical decision	Choice	Winner, probability per option, confidence
Binary check	Noul (for Bernoulli)	Probability that the statement is true
Ordinal decision	Score	Weighted position, probability per level, confidence
Option definition	Criteria	Meaning, boundaries and examples
Product policy	Application code	Threshold, action, review or fallback

1. Request anatomy

A request carries one *state* and any number of *questions*. Each question has an ID of your choosing, a type (Choice, Noul or Score), an *instruction*, and *criteria*: the options of a Choice, the ordered levels of a Score, or optional definitions of yes and no for a Noul. Instructions and criteria may be plain text or structured, for example stating what an option covers, what it excludes, and examples. TypeSafe calls the model class *System One*, after Daniel Kahneman's term for fast, intuitive judgement.

2. Reading Jev's output

A Choice answer for the shoe-shop ticket's department question looks like this (recorded):

```
"department": {"type": "choice", "choice": "returns", "confidence": 0.42,
               "probabilities": {"returns": 0.61, "billing": 0.35, "shipping": 0.04}}
```

Confidence. The published outputs are consistent with the following formula. For a Choice or Score with n options, where p_{max} is the top probability:

$$\text{confidence} = \frac{n \cdot p_{max} - 1}{n - 1}, \qquad p_{max} = \frac{(n - 1) \cdot \text{confidence} + 1}{n}.$$

For the department answer, $n = 3$ and $p_{max} = 0.61$, so confidence = $(1.83 - 1) \div 2 = 0.415$, reported as 0.42. TypeSafe describes confidence as derived from the spread of probabilities without publishing an exact definition; this formula is inferred, and it reproduces every confidence in this report. On that reading confidence is a rescaled top probability, not a full measure of how concentrated the distribution is: it ignores everything except the winner and the number of options, so it cannot reveal a close runner-up; compute the margin for that. A Noul returns only its probability, which already expresses its uncertainty.

Score. The `score` field is ∑(level × probability), with levels numbered from 0 in the order given, and it is returned with the probability of each level and a confidence.

3. Current product characteristics

- Questions in one request are answered independently and in parallel against the same state. In a recorded test, thirteen questions about one long document took 0.27 seconds as one request and 2.71 seconds as thirteen, with identical answers.
- Up to 255 options per Choice and 2 to 10 levels per Score; text input only, English strongest; about 32,000 tokens for the state plus the longest question.

- Measured round trips in TypeSafe's own tests: 111 milliseconds for the fourteen-check claim, 270 milliseconds for thirteen questions over a long document. These are single benchmarks, not a service guarantee.
- The same model serves every customer, with no fine-tuning; answers are shaped only through the state and the wording of questions and criteria.
- Named aliases move to new versions, so pin the version your thresholds were tuned on and re-evaluate before moving.
- Known weaknesses, as TypeSafe lists them: literal reading of instructions, unreliable counting and date comparison, weaker performance on multi-step questions and large irrelevant input, and susceptibility to text written to steer the answer. Related questions need not give consistent numbers: a check and its negation can sum to more than 1.
- Answers vary slightly between repeated requests, by about 0.01 on average in the claim test.

4. Evidence and sources

Recorded results come from TypeSafe's published worked examples and were not independently re-run. Each source title is a link; TypeSafe documentation was accessed on 24 September 2026.

Result	Sample	Model version	Source
Shoe-shop ticket	1 message, 5 Choice questions	1.13, Sep 2026	Choice
Company classification	60 annual reports, 75 groups, pre-selected to fit their industry	1.12, Aug 2026	Classification using confidence
Insurance claim	1 claim, 14 Noul checks, 15 runs with a fresh irrelevant identifier	1.13, Sep 2026	Self-consistency: nouls
Record matching	450 candidate pairs from two public catalogues	1.12, Aug 2026	Knowledge graph entity alignment
Consistency	8 moderation questions, 15 runs with a fresh irrelevant identifier	1.13, Sep 2026	Self-consistency: choices
One request against many	13 questions over one document, 5 runs	Not stated	Parallel questions
Ordinal score			Score
Confidence, derived from the probabilities			Confidence
Known weaknesses			Jev 1.13 jaggedness
Noul named for Bernoulli, by the founder on 15 September 2026			Hacker News launch thread

Illustrative figures, invented to show a method, are: the £40 refund and £2 review, the £5 and £500 fraud costs, the $30 claim review, the calibration table, the 39 of 40 merges, and the 13 of 15 agreement.