Supervised Learning: The Complete Picture

Data Science with Python – BSc Course

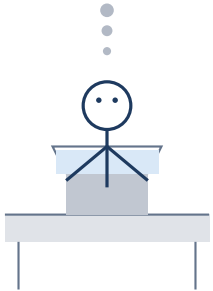
90–120 Minutes

Every morning, the same two questions...

Door 1

How much?

(Regression)



Door 2

Which one?

(Classification)

Learning Objectives

After this session you will be able to:

1. **Explain** supervised learning and how it differs from unsupervised learning
2. **Compare** regression and classification tasks with finance examples
3. **Evaluate** model quality using appropriate metrics (MSE, R^2, F1, AUC)
4. **Apply** regularization techniques (Ridge, Lasso) to prevent overfitting
5. **Distinguish** precision from recall and choose the right trade-off

---

Bloom's taxonomy: Explain → Compare → Evaluate → Apply → Distinguish

The Big Picture: Supervised Learning

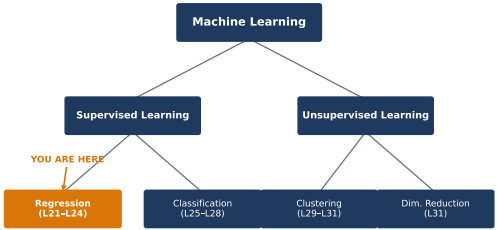
Core idea: Learn a mapping from inputs to outputs using labelled data.

$$\hat{y} = f(X) + \epsilon$$

- X = features (price, volume, fundamentals)
- $\hat{y}$ = prediction
- f = learned model

Example: For MSFT with $X = [\text{price}, \text{volume}, \text{PE}, \text{debt/equity}, \text{sector}]$ predict $\hat{y}$ = next-day return. Feed 5 years of daily rows so f learns the pattern.

Read the chart: The tree splits ML into Supervised (labels available) and Unsupervised (no labels). Each branch lists two common task types. Only the supervised branch trains on (X, y) pairs.



Supervised = we have the "answer key" (labels) during training — e.g., past stock returns paired with next-day outcomes

Two Flavors of Prediction

Regression – continuous target

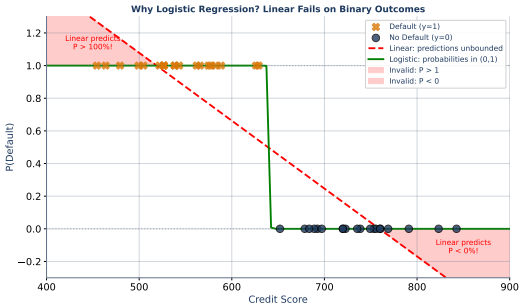
- Stock return tomorrow
- House price estimate
- Portfolio risk forecast

Classification – categorical target

- Buy / Sell / Hold
- Default vs. No Default
- Fraud vs. Legitimate

Example: Same AAPL features (return-to-date, volume, RSI). Regression output $\hat{y} = 2.3\%$ (expected return). Classification output $\hat{y} =$ "Buy" (label).

Read the chart: Blue straight line = linear regression output (any real number). Amber S-curve = logistic regression output (bounded in $[0, 1]$). The same input x produces two different shapes of prediction.



Same pipeline, different loss functions — a bank predicts default probability (classification) and loss amount (regression)

Linear Regression: The Starting Point

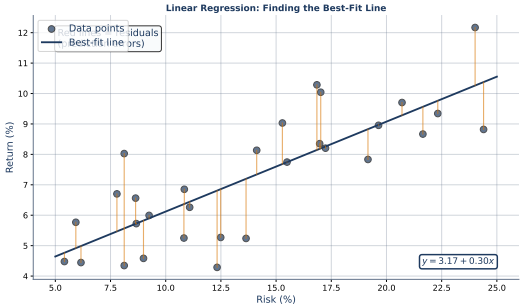
Ordinary Least Squares (OLS) fits a straight line by minimising squared errors:

$$\hat{y} = \beta_0 + \beta_1 x$$

- β_0 = intercept, β_1 = slope
- Slope = change in $\hat{y}$ per unit change in x

Example: Fitted on AAPL returns vs SPY returns: $\hat{y} = 0.02 + 0.80 \cdot x$. If SPY rises 10%, predicted AAPL return is $0.02 + 0.80 \cdot 10 = 8.02\%$.

Read the chart: Each navy dot is one observation. The navy line minimises the total *squared* vertical distance from every dot. Dots far from the line contribute more (because the distance is squared) than dots close to the line.



OLS: the most interpretable model in finance — regress stock returns on market returns to estimate beta

The OLS Formula

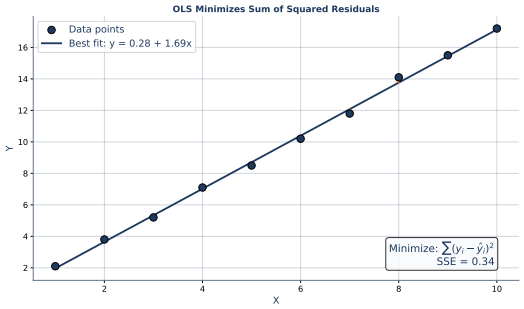
Closed-form solution – no iterative optimisation needed.

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{\text{Cov}(x, y)}{\text{Var}(x)}$$

- Numerator: how x and y move together
- Denominator: how spread out x is

Example: On AAPL vs SPY daily returns: $\text{Cov}(\text{AAPL}, \text{SPY}) = 0.00024$, $\text{Var}(\text{SPY}) = 0.00018$. Then $\beta_1 = 0.00024/0.00018 = 1.33$.

Read the chart: The navy dots are observations; the navy line is the OLS fit. The short vertical segment at each dot is its residual $(y_i - \hat{y}_i)$ – the quantity the formula squares and sums before minimising.



$\beta_1 = \text{Cov}(x, y)/\text{Var}(x)$ **— this is how you compute the sensitivity of an asset to any factor**

CAPM Beta: Regression in Action

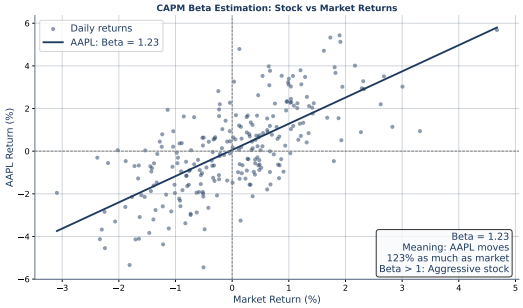
The **Capital Asset Pricing Model** is a single-factor linear regression:

$$R_i - R_f = \alpha + \beta(R_m - R_f) + \epsilon$$

- $\beta > 1$: amplifies market moves (aggressive)
- $\beta < 1$: dampens market moves (defensive)
- α: excess return beyond market exposure

Example: Tesla ($\beta \approx 1.5$): if market excess return is $+4\%$, Tesla's expected excess return is $1.5 \times 4\% = +6\%$. Utility ($\beta \approx 0.5$): $0.5 \times 4\% = +2\%$.

Read the chart: X-axis = market excess return; Y-axis = stock excess return. Each navy dot is one trading day. The navy line is the OLS fit; its slope equals β.



Every portfolio manager uses beta — Tesla ($\beta \approx 1.5$) moves 50% more than the market; utilities ($\beta \approx 0.5$) move half as much

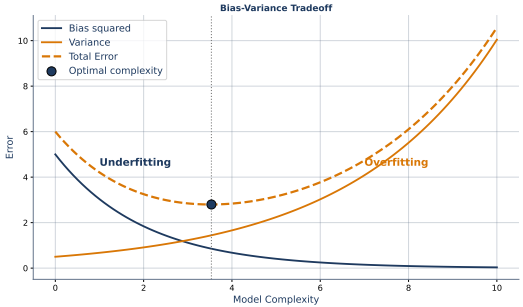
The Bias–Variance Tradeoff

Underfitting (high bias): model too simple, misses patterns.
Overfitting (high variance): model memorises noise.

- Sweet spot: enough complexity to capture signal, not noise
- A 20-factor model on 30 data points will overfit

Example: On 500 days of returns, a 1-factor model (just market) has high bias ($R^2 = 0.4$). A 50-factor model overfits: $R^2_{\text{train}} = 0.95$ but $R^2_{\text{test}} = 0.10$.

Read the chart: X-axis = model complexity; Y-axis = prediction error. The U-shaped navy curve (total error) splits into a falling *bias* part (left) and a rising *variance* part (right). The vertical dashed line marks the sweet-spot complexity.



Regularization is the tool that navigates this tradeoff — e.g., shrinking 50 macro factors to the few that matter

Ridge Regression: Turn Down the Volume

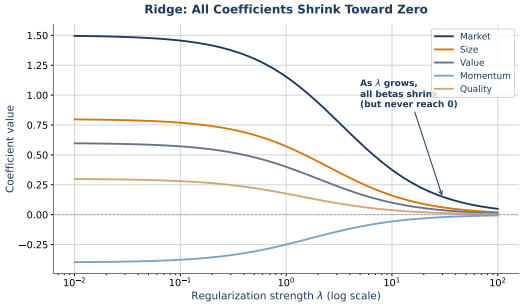
L2 penalty shrinks all coefficients toward zero:

$$\mathcal{L} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$$

- λ controls shrinkage strength
- Keeps all features but reduces their influence
- Best when many features contribute small amounts

Example: OLS estimates (Market=1.5, Size=0.8). With $\lambda = 1$: Ridge shrinks to (1.2, 0.6). With $\lambda = 10$: further to (0.6, 0.3). Neither becomes exactly zero.

Read the chart: X-axis = λ on a log scale. Each coloured line is one feature's coefficient as λ grows. All five paths *approach* zero but none actually touch the horizontal axis – the L2 signature.



Ridge = "turn down the volume" on all coefficients equally — useful when 30 macro factors each explain a small part of returns

Lasso Regression: Mute the Noise

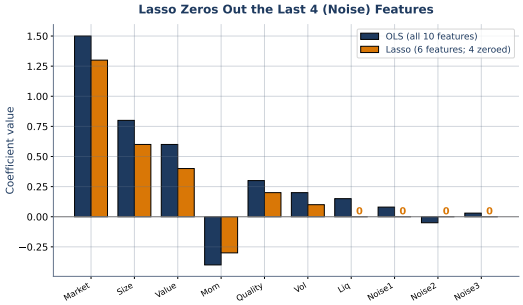
L1 penalty drives some coefficients to exactly zero:

$$\mathcal{L} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$$

- Automatic **feature selection**: only important factors survive
- From 50 macro factors, maybe 5 get nonzero coefficients

Example: Hedge fund starts with 10 signals. OLS keeps all 10. Lasso ($\lambda = 1$) zeros out the 4 noise signals, leaving 6 non-zero coefficients. The model now has built-in feature selection.

Read the chart: Side-by-side bar pairs: navy bars are OLS coefficients (all 10 are non-zero), amber bars are Lasso coefficients. The last 4 amber bars are labelled "0" at the baseline – Lasso dropped them entirely.



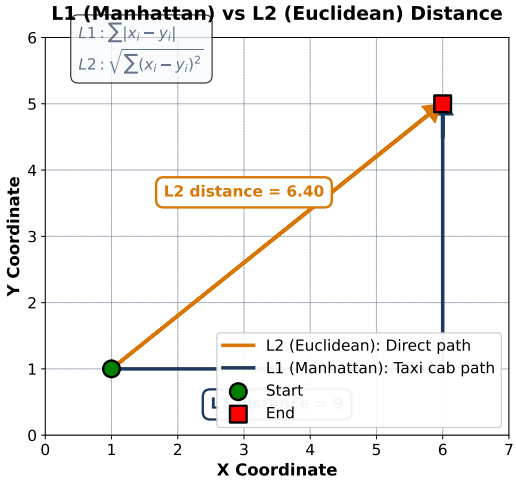
Lasso = built-in variable selection — a hedge fund screening 200 signals uses Lasso to find the 10 that matter

Why Lasso Zeros Out Coefficients

- **L1 (Lasso):** diamond constraint – corners touch axes, so the OLS solution hits a corner $\Rightarrow$ exact zeros
- **L2 (Ridge):** circle constraint – no corners, so the solution lands on the curve $\Rightarrow$ small but nonzero

Example: With 2 features, OLS optimum sits at $(3, 0.5)$. The L1 diamond of radius 2 is first touched at the corner $(2, 0)$ – Lasso zeros out β_2. The L2 circle of radius 2 is touched at roughly $(1.97, 0.33)$ – Ridge keeps both.

Read the chart: Dashed grey ellipses are OLS contour lines around the unconstrained minimum. The navy circle (L2) and amber diamond (L1) are constraint regions. The solution is where a contour first touches the constraint – for L1 that first touch point is a diamond corner (on the axis).



Diamond corners create sparsity — this geometric insight is the key difference between Ridge and Lasso

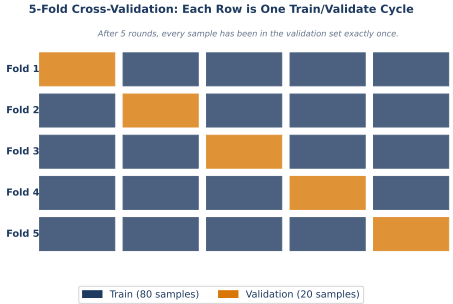
Cross-Validation: Choosing λ

K-fold cross-validation prevents data leakage when tuning λ:

- Split data into k folds (typically 5 or 10)
- Train on $k-1$ folds, validate on held-out fold
- Repeat k times, average the error

Example: 1000 days of returns, 5-fold CV: each fold trains on 800 days, validates on 200. Repeat 5 times so every day is validated once. Average RMSE across folds is the CV score.

Read the chart: Five horizontal rows, one per fold. In each row, one amber block = that round's validation set and the navy blocks = the training set. After all 5 rows, every column position has been amber exactly once.



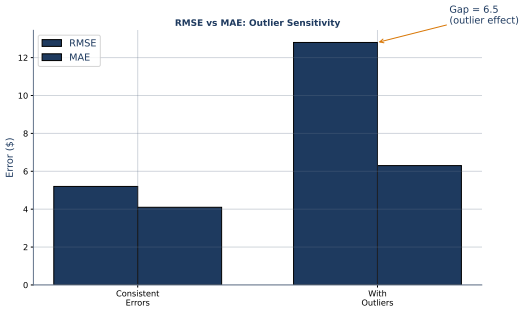
Never tune hyperparameters on the test set — 5-fold CV on 10 years of monthly returns uses 8 years for training per fold

Regression Metrics: How Good Is Good?

- **MSE** $= \frac{1}{n}\sum(y_i - \hat{y}_i)^2$ – penalises large errors heavily
- **RMSE** $= \sqrt{\text{MSE}}$ – same units as y
- **MAE** $= \frac{1}{n}\sum|y_i - \hat{y}_i|$ – robust to outliers

Example: Errors $[0.5, 0.3, 0.4, 0.2, 3.0]$ (last = outlier). MSE $= 1.91$, RMSE $= 1.38\%$, MAE $= 0.88\%$. RMSE blows up due to the 3.0; MAE barely moves.

Read the chart: Three grouped bars per error size. As the residual magnitude grows from left to right, the MSE bar (navy) shoots up quadratically, RMSE (amber) rises more moderately, and MAE (slate) stays roughly proportional.



If MAE = 0.8% on daily returns, the average prediction misses by 0.8 percentage points — useful for setting trading thresholds

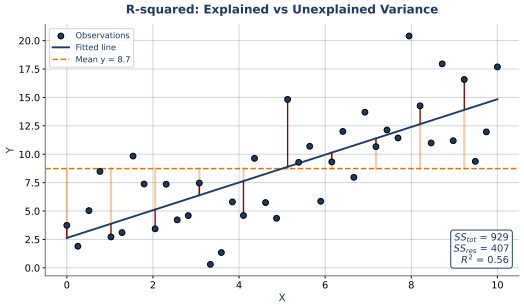
R^2: Proportion of Variance Explained

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

- $R^2 = 1$: perfect fit
- $R^2 = 0$: no better than predicting the mean
- Even $R^2 = 0.02$ can be profitable at scale

Example: $SS_{\text{tot}} = 400$ (total variance), $SS_{\text{res}} = 100$ (unexplained by the model). Then $R^2 = 1 - 100/400 = 0.75$ – the model captures 75% of the variance in y.

Read the chart: Navy dots are observations; the navy line is the OLS fit; the dashed amber line is $\bar{y}$. Amber vertical segments (from mean to fit) = explained variance; dark-red segments (from fit to point) = residual. The ratio of their squared lengths gives R^2.



A low R^2 does not mean a useless model — Renaissance Technologies reportedly profits from $R^2 < 0.05$ on daily returns

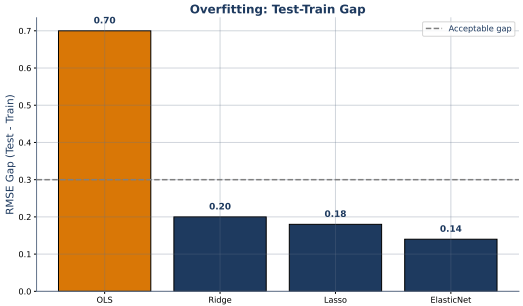
The Overfitting Gap

Training error always decreases with complexity.
Test error eventually rises – the gap signals overfitting.

- Monitor both curves as you add features
- Stop when the gap starts widening

Example: At 5 features: train $R^2 = 0.62$, test $R^2 = 0.58$ (gap $= 0.04$, healthy). At 50 features: train $R^2 = 0.95$, test $R^2 = 0.45$ (gap $= 0.50$, overfit).

Read the chart: Navy curve = training error (falls monotonically). Amber curve = test error (dips then rises). At any x-value, the vertical gap between them is the overfitting gap; the red-shaded region marks where test error starts deteriorating.



The train-test gap is your overfitting detector — backtested returns that diverge from live returns signal the same problem

Factor Models: Multi-Factor Regression

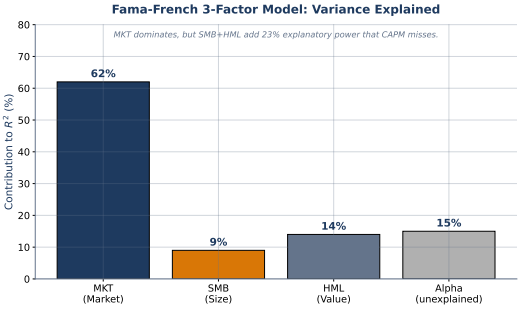
The **Fama–French** model extends CAPM with size and value factors:

$$R_i - R_f = \alpha + \beta_1(\text{MKT}) + \beta_2(\text{SMB}) + \beta_3(\text{HML}) + \epsilon$$

- SMB: Small Minus Big (size factor)
- HML: High Minus Low (value factor)
- Each coefficient is an OLS regression slope

Example: Apple (large-cap growth) typical coefficients: $\beta_{\text{MKT}} = 1.2$, $\beta_{\text{SMB}} = -0.3$ (behaves like big-cap), $\beta_{\text{HML}} = -0.5$ (behaves like growth, not value).

Read the chart: Four vertical bars showing the share of variance each factor explains. MKT (navy, 62%) dominates. SMB (amber, 9%) and HML (slate, 14%) add 23% combined. The final grey bar (Alpha, 15%) is what the model still cannot explain.



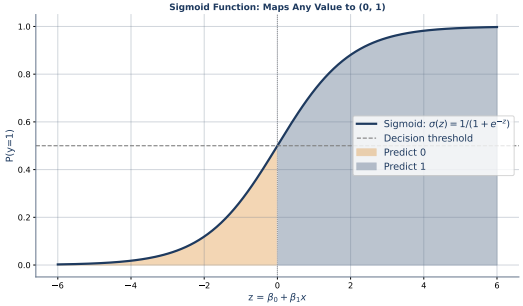
Factor models = multiple regression for asset pricing — Fama and French won the Nobel Prize for this framework

Logistic Regression: Probabilities, Not Lines

The **sigmoid** maps any real number to $(0, 1)$:

$$P(y{=}1 \mid x) = \frac{1}{1 + e^{-z}}$$

- Threshold at $P = 0.5$; output is a probability
- Despite the name, this is a classification algorithm
- This *probability* already implies a **decision boundary** at $P = 0.5$



Example: Borrower with $z = \beta_0 + \beta_1 \cdot \text{debt} + \beta_2 \cdot \text{income} = 2.3$. Then $P = 1/(1 + e^{-2.3}) = 0.91$ – 91% default probability, above 0.5 threshold $\Rightarrow$ predict default.

Read the chart: X-axis = linear score z; Y-axis = probability. The navy S-curve crosses 0.5 exactly at $z = 0$. Far left ($z < -4$) gives $P \approx 0$; far right ($z > 4$) gives $P \approx 1$. In 2D feature space, the horizontal line $P = 0.5$ becomes a decision boundary.

Banks use logistic regression to estimate probability of default — the sigmoid output IS the PD estimate

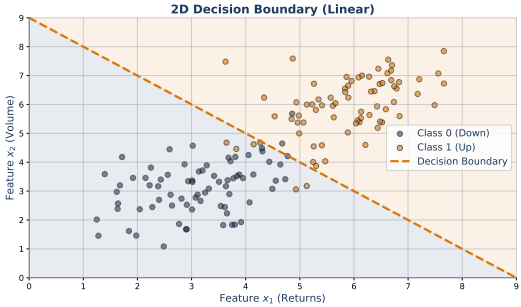
Decision Boundaries in 2D

With two features the sigmoid creates a **linear boundary** separating classes:

- Points on one side: predicted class 1
- Points on the other: predicted class 0
- Boundary shifts as coefficients change
- **This thread returns:** every classifier we meet next has its own boundary shape

Example: Credit model $z = -5 + 0.06 \cdot \text{debt}/\$\text{k} - 0.04 \cdot \text{income}/\$\text{k}$. Boundary $P = 0.5$ occurs where $z = 0$, i.e. debt $\approx 0.67 \cdot$ income. A borrower with \$30k income and \$25k debt just crosses onto the default side.

Read the chart: X- and Y-axes are the two credit features. Navy dots = no-default; amber dots = default. The straight line is the $P = 0.5$ decision boundary – points on one side are predicted non-defaulters, on the other, defaulters. Dots close to the line are the most uncertain predictions.



Logistic regression draws a straight line between classes — separating defaulters from non-defaulters based on income and debt ratio

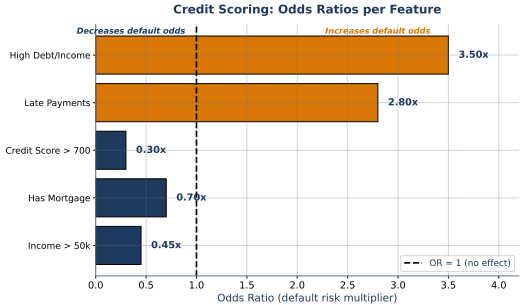
Credit Scoring: Odds Ratios

Odds ratio $= e^{\beta_j}$: how a one-unit increase in x_j changes the odds.

- $e^{\beta} > 1$: increases default risk
- $e^{\beta} < 1$: decreases default risk
- Regulators require interpretable models

Example: Coefficient on "Late Payments" is $\beta = 1.03$. Then OR $= e^{1.03} = 2.80$: each additional late payment *multiplies* the odds of default by 2.8.

Read the chart: Horizontal bars, one per feature, with an "OR=1" dashed reference line. Navy bars to the left of 1 (Income > 50k: 0.45, Credit Score > 700: 0.30) *decrease* default odds. Amber bars to the right (Late Payments: 2.80, High Debt/Income: 3.50) *increase* default odds.



Odds ratios make logistic regression the go-to model for regulated industries — "each 10k in debt multiplies default odds by 1.3x"

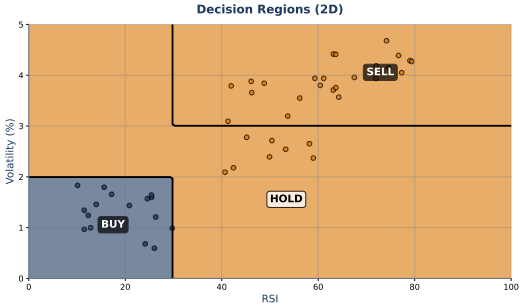
Decision Trees: If–Then Rules

A tree splits data with **yes/no questions** at each node:

- Root: most informative split
- Leaves: final predictions
- Fully interpretable – you can read the rules
- **Decision boundary:** axis-aligned rectangles, unlike logistic regression's straight line

Example: Rule: "Is debt/income $> 40\%$?" Yes $\rightarrow$ "Is credit score < 650?" Yes $\rightarrow$ predict *high risk*. No $\rightarrow$ predict *low risk*.

Read the chart: X-axis = RSI, Y-axis = Volatility. The tree carves feature space into three axis-aligned rectangles – BUY (low RSI, low vol), HOLD (middle), SELL (high RSI, high vol). Each black line is one tree split; stacked splits build the full boundary.



Trees are the most interpretable non-linear model — "Is debt-to-equity > 2?" $\rightarrow$ high credit risk

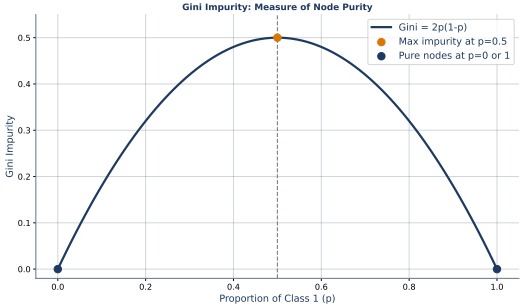
Gini Impurity: Measuring Split Quality

$$\text{Gini}(S) = 1 - \sum_{i=1}^{C} p_i^2$$

- p_i = proportion of class i in the node
- Gini = 0: perfectly pure (one class only)
- Gini = 0.5: maximum impurity (50/50 split)

Example: A node with 90 defaults and 10 non-defaults: $p_1 = 0.9, p_2 = 0.1$ so Gini $= 1 - (0.81 + 0.01) = 0.18$ – nearly pure. Split gives two children Gini 0.05 and 0.08: use this split.

Read the chart: X-axis = proportion of the positive class; Y-axis = Gini value. The navy curve peaks at $p = 0.5$ with Gini = 0.5 and drops to 0 at both extremes ($p = 0$ and $p = 1$). Trees pick the split that lowers (weighted) Gini the most.



Trees choose the split that reduces Gini impurity the most — a node with 90% defaults has Gini = 0.18 (nearly pure)

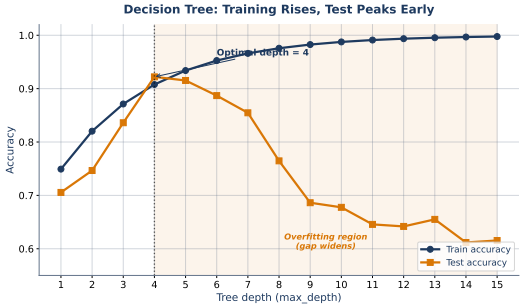
Tree Overfitting: When Depth Hurts

Deep tree: memorises training data.
Shallow tree: generalises but may underfit.

- Control depth via `max_depth`, `min_samples_leaf`
- Over-complex trees fail on new market regimes

Example: On the same 150-point dataset – depth 1: train 65%, test 64% (underfit). Depth 4: train 92%, test 88% (sweet spot). Depth None: train 100%, test 72% (overfit).

Read the chart: X-axis = max tree depth; Y-axis = accuracy. The navy training curve rises monotonically toward 100%. The amber test curve peaks around depth 5 then decays – that peak marks the "optimal depth". The amber-shaded region flags the overfitting zone.



Pruning a tree is regularization for non-linear models — a tree trained on 2019 data may not survive the 2020 regime shift

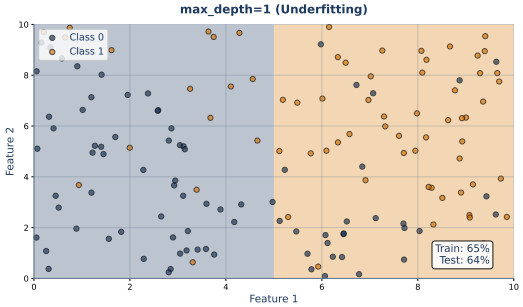
Shallow Tree Boundary (Underfit)

Depth = 1 tree splits once: one vertical line, two rectangles.

- Single feature gate (e.g. Feature 1 > 5)
- Ignores the true diagonal pattern
- Train 65%, Test 64% – high bias

Example: Underfit boundary predicts "Class 1" for every point with Feature 1 > 5, even when Feature 2 is low. Misclassifies the bottom-right corner entirely.

Read the chart: One straight vertical cut separates navy (Class 0) and amber (Class 1) regions. Many dots sit on the "wrong" side – the boundary is clearly too coarse.



Underfit boundary: one cut, zero diagonal awareness — the model sees only half the signal

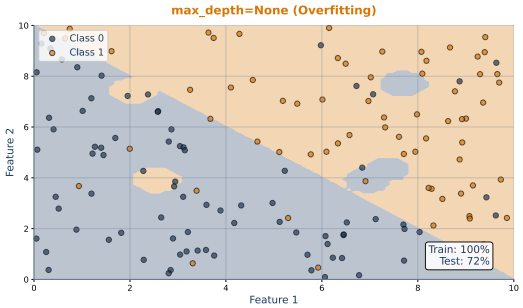
Deep Tree Boundary (Overfit)

Depth = None tree keeps splitting until every leaf is pure.

- Fragmented, irregular regions
- Chases individual noise points
- Train 100%, Test 72% – high variance

Example: The tree creates a tiny "island" region around every mislabelled training point. Great on training data, terrible on any new sample that falls in those islands.

Read the chart: The boundary is now a jigsaw of rectangles and carved-out pockets. Compare with the previous slide: same data, far more complex boundary – a classic variance explosion.



Overfit boundary: islands around every noise point — 100% train accuracy is a red flag, not a trophy

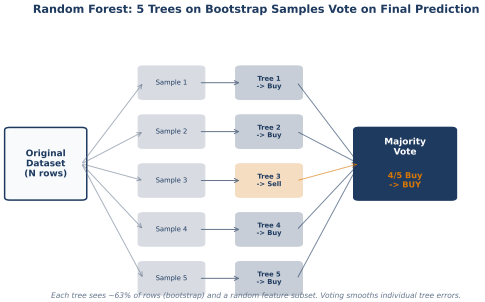
Random Forests: Wisdom of Crowds

Bagging + random feature subsets = Random Forest:

- Train many trees on bootstrap samples
- Each tree sees a random subset of features
- Aggregate by majority vote (or averaging)

Example: 100 trees vote on a trade. 64 vote "Buy", 36 vote "Sell" ⇒ ensemble predicts "Buy". A single tree misclassifies 25% of samples; the forest misclassifies only 12%.

Read the chart: Left box = original dataset. Middle column = 5 bootstrap resamples each feeding its own tree. Arrow colours show each tree's vote: 4 navy (Buy) + 1 amber (Sell). Right box aggregates: 4/5 Buy ⇒ final "BUY".



Random Forests reduce variance by averaging uncorrelated trees — widely used in credit scoring where stability matters

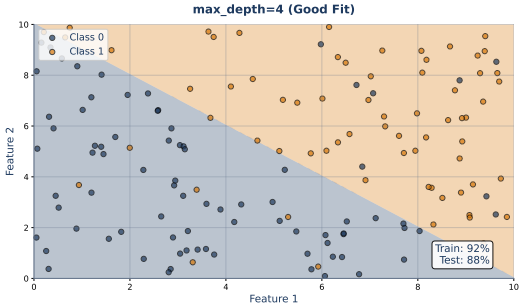
Random Forest Boundary (Smoothed)

Averaging many jagged tree boundaries yields a **smooth** ensemble boundary.

- Each tree has its own errors
- Uncorrelated errors cancel on average
- Result: lower variance without raising bias

Example: 100 trees each have ragged axis-aligned boundaries. Averaging their predictions yields a boundary that roughly follows the true diagonal – no islands, no staircase artefacts.

Read the chart: Compare with the jagged deep-tree boundary two slides back. This "good-fit" boundary (depth 4 single tree, used here as an RF-like reference) is smoother, follows the diagonal, and generalises to test data (88% accuracy).



Ensemble boundary = voting averages out individual tree noise — the hallmark of Random Forests

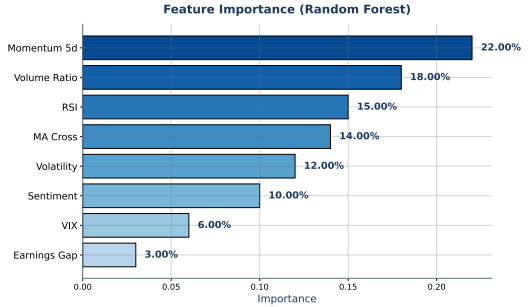
Feature Importance: What Drives Predictions?

Permutation importance: shuffle one feature, measure accuracy drop.

- Large drop $\Rightarrow$ important feature
- No change $\Rightarrow$ redundant feature

Example: Credit model accuracy = 88%. Shuffle "Debt/Income": accuracy drops to 60% (importance 0.28). Shuffle "Zip Code": accuracy stays 88% (importance 0.00). Top 3 features often sum to 75%+ of total importance.

Read the chart: Horizontal bars sorted from top (most important) to bottom. Bar length = accuracy drop when that feature is shuffled. The top 3 bars dominate the total length – Pareto-style concentration.



Feature importance answers "which macro variables matter for credit risk?" — often the top 3 features carry 80% of predictive power

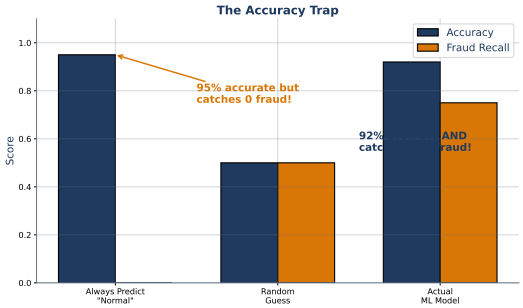
The Accuracy Trap

99% accuracy sounds great – until you realise 99% of transactions are legitimate.

- A model predicting "no fraud always" gets 99% accuracy
- It catches zero fraud – completely useless
- Accuracy is misleading with imbalanced classes

Example: 1000 transactions, 10 fraud, 990 legitimate. The "always legit" rule: 990 correct out of 1000 $\Rightarrow$ 99% accuracy, but 0/10 fraud caught $\Rightarrow$ recall = 0%. Worthless.

Read the chart: Left bar: naive majority-class model (99% accuracy, 0% recall). Right bar: real fraud model (96% accuracy, 70% recall). Accuracy alone cannot distinguish them; recall on the minority class does.



When classes are imbalanced, accuracy is the wrong metric — a fraud detection model with 99% accuracy and 0% recall is worthless

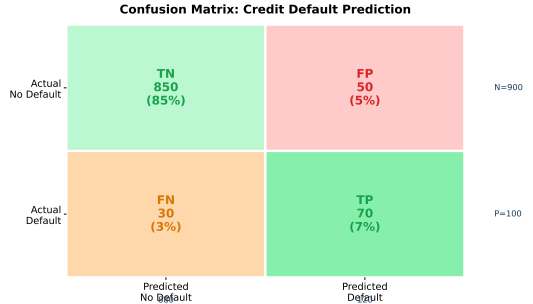
The Confusion Matrix

A **confusion matrix** is a 2×2 table counting every prediction outcome:

- **TP**: correctly predicted positive (caught the fraud)
- **FP**: false alarm (flagged legitimate transaction)
- **FN**: missed positive (fraud slipped through)
- **TN**: correctly predicted negative

Example: 1000 transactions, 100 fraud. Model: TP = 85, FP = 15, FN = 15, TN = 885. Read as: "caught 85 fraud, raised 15 false alarms, missed 15, correctly cleared 885".

Read the chart: A 2×2 grid with rows = actual class, columns = predicted class. The diagonal cells (top-left TN and bottom-right TP) hold correct predictions; off-diagonal cells (TP/FP top-right and FN/TN bottom-left per layout) hold the two error types.



The confusion matrix is the foundation of all classification metrics — every metric is a ratio of these four cells

Precision, Recall, and F1

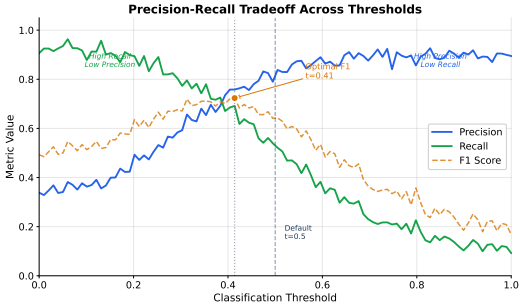
$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

- **Precision**: of those flagged, how many are truly positive?
- **Recall**: of all actual positives, how many did we catch?
- **F1**: harmonic mean balancing both

Example: From the previous matrix (TP=85, FP=15, FN=15). Precision $= 85/100 = 0.85$. Recall $= 85/100 = 0.85$. $F_1 = 2 \cdot 0.85 \cdot 0.85/1.70 = 0.85$.

Read the chart: X-axis = classification threshold from 0 to 1. Two curves: precision (rises) and recall (falls) as threshold grows. Where they cross (around threshold 0.5) is the balanced operating point; F1 peaks near there.



There is always a precision–recall tradeoff — a bank may prefer high recall (catch all defaults) even at the cost of more false alarms

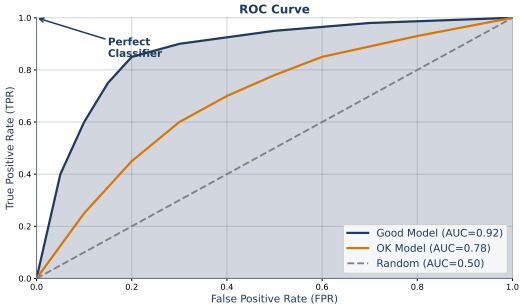
ROC Curve and AUC

The **ROC curve** plots True Positive Rate vs. False Positive Rate across all thresholds.

- **AUC = 1.0**: perfect classifier
- **AUC = 0.5**: random guessing (diagonal)
- Threshold-independent model comparison

Example: AUC = 0.85 means: pick a random fraud case and a random legit case; the model assigns a *higher* score to the fraud case 85% of the time. AUC = 0.5 would be pure coin flip.

Read the chart: X-axis = False Positive Rate, Y-axis = True Positive Rate. The navy curve bows toward the top-left corner; the dashed diagonal line is random guessing. Area under the navy curve (shaded) is the AUC score.



AUC compares models without committing to a threshold — an AUC of 0.85 means an 85% chance a random positive ranks higher than a random negative

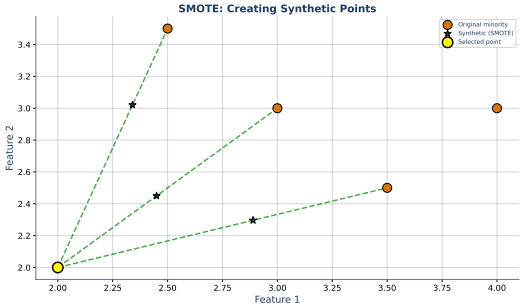
SMOTE: Synthesising Minority Samples

SMOTE creates synthetic samples by interpolating between existing minority points:

- Pick a minority sample and one of its k nearest neighbours
- Create a new point along the line between them
- Balances the dataset without simple duplication

Example: 10 fraud samples out of 1000. SMOTE with $k = 5$ generates new points along segments between fraud neighbours. After SMOTE: 500 minority (original 10 + 490 synthetic) = 50/50 balance.

Read the chart: Amber dots = original minority samples; navy dots = synthetic SMOTE samples. Thin connecting lines show the interpolation segments – new points sit between a minority point and its nearest neighbour. No duplicates, just coverage.



SMOTE generates diverse synthetic samples — useful when fraud cases are 0.1% of transactions and you need more examples to train on

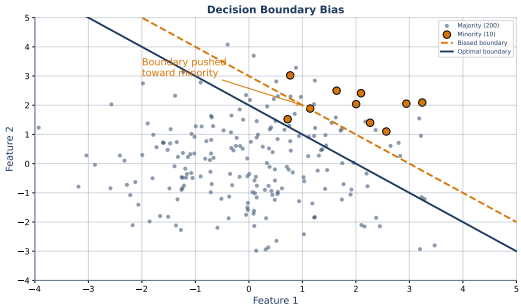
Imbalance Bias in the Decision Boundary

On imbalanced data, the default boundary is **pushed toward the minority** – the model prefers to misclassify rare events.

- Majority dominates the loss
- Minority points are treated as noise
- Rebalancing (SMOTE, weights) corrects this

Example: 200 majority, 10 minority. Unweighted logistic puts the line too far toward the minority cluster. Catches $< 30\%$ of minority cases. Rebalancing shifts the line and raises recall.

Read the chart: Navy dots = majority (200), amber dots = minority (10). The dashed amber line is the biased (unweighted) boundary; the solid navy line is the corrected "optimal" boundary. The amber arrow shows how far the biased line has been pushed.



Imbalanced data bends the boundary away from the minority — SMOTE and class weights pull it back

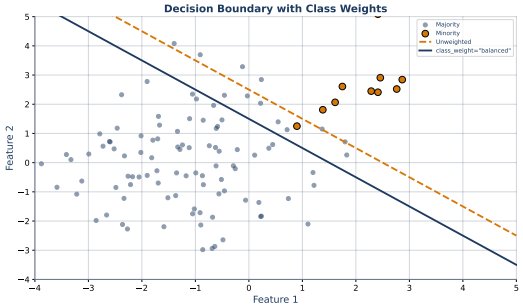
Handling Class Imbalance: Strategy Comparison

Multiple approaches – choose based on dataset size:

- **Oversampling** (SMOTE): generate more minority examples
- **Undersampling**: discard majority examples
- **Class weights**: penalise misclassifying minority

Example: 0.1% fraud in 100k rows (100 fraud). SMOTE: add 49,900 synthetic fraud (50/50). Undersampling: drop 99,800 legit (keep 200). Class weights: use `class_weight='balanced'` = weight 999 for fraud, 1 for legit.

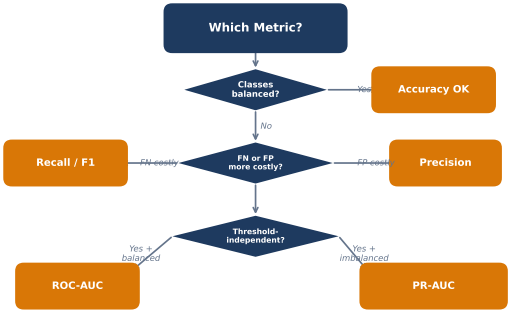
Read the chart: Same data; two decision boundaries. The dashed amber line is the *unweighted* classifier boundary (pushed toward the minority). The solid navy line is the `class_weight="balanced"` boundary – closer to the true separator, catching the amber minority cluster.



In finance, the rare event (default, fraud) is the important one — class weights are simplest, SMOTE works best with small minority counts

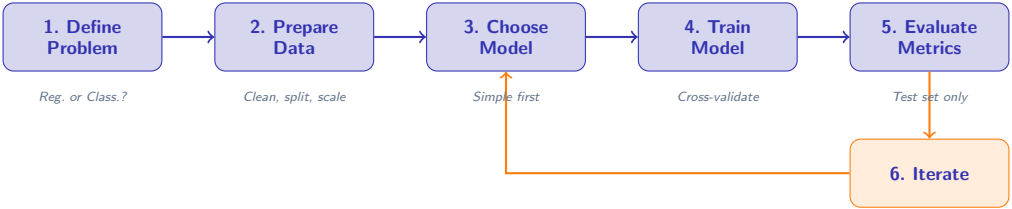
Regression vs. Classification: Summary

Aspect	Regression	Classification
Target	Continuous	Categorical
Loss	MSE / MAE	Cross-entropy
Metrics	R^2, RMSE	F1, AUC
Example	Stock return	Default flag
Baseline	Mean	Majority class



Match your metric to your problem type — never use accuracy for imbalanced data, never use R^2 alone for financial returns

The 6-Step Supervised Learning Pipeline



- Always start simple (linear / logistic regression)
- Iterate between model choice and evaluation
- Step 2 in finance includes look-ahead bias checks

This pipeline applies to every supervised learning project — from predicting stock returns to detecting fraudulent claims

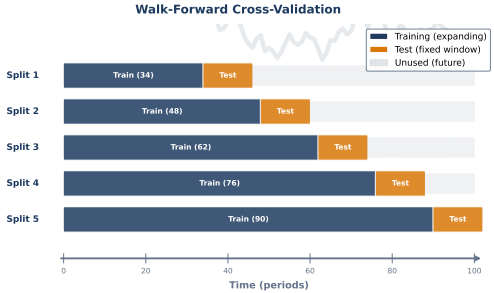
Walk-Forward Validation for Time Series

Standard CV **shuffles data** – dangerous for time series (future leaks into past).

- **Walk-forward**: train on past, test on next period, slide forward
- Respects the arrow of time
- The only valid backtesting approach

Example: Train 2018–2020 → test 2021. Next window: train 2018–2021 → test 2022. Window slides forward; test data never appears in training.

Read the chart: Each row shows one validation window. Navy (train) always precedes amber (test) on the time axis. As you move down the rows, the window slides right – never left.



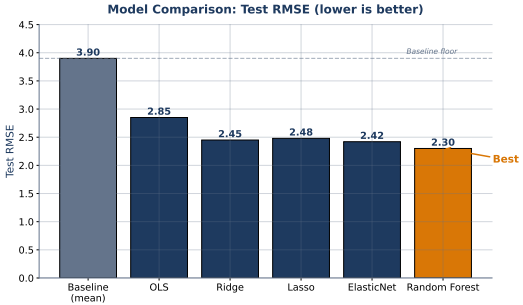
In finance, always use walk-forward — random k-fold on daily returns lets 2023 data "predict" 2022, inflating results

Model Comparison: Start Simple

- **Baseline**: mean (regression) or majority class (classification)
- **Linear / logistic**: interpretable, fast
- **Regularized**: Ridge / Lasso when features > observations
- **Trees / forests**: when non-linearity matters

Example: Same daily-return dataset – Baseline RMSE = 3.90. OLS = 2.85 (overfit). Ridge = 2.45. Lasso = 2.48. ElasticNet = 2.42. RF = 2.30. ElasticNet wins on interpretability; RF wins on score.

Read the chart: Six vertical bars, lowest test RMSE wins. The grey baseline bar sets the floor; navy bars are linear variants; amber bar is Random Forest (best). The dashed line marks the baseline RMSE for visual comparison.



The simplest model that meets your threshold is the best model — a bank regulator will not approve a black-box when logistic regression suffices

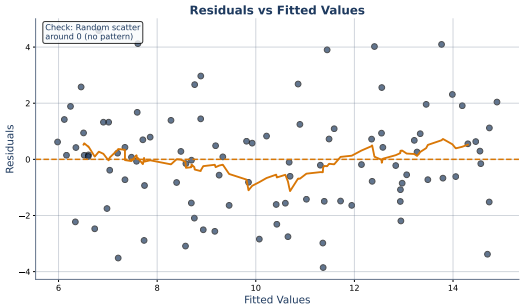
Residual Analysis: Diagnosing Your Model

Residuals $= y_i - \hat{y}_i$ reveal what the model missed:

- Random scatter: model is well-specified
- Patterns (curves, fans): missing nonlinearity or heteroskedasticity

Example: Predicted return $\hat{y} = 0.01$, actual $y = 0.04$. Residual $= 0.03$ (3% error). Plot all 500 residuals; if they form a fan shape (wider at high $\hat{y}$), variance grows with prediction level.

Read the chart: X-axis = fitted value; Y-axis = residual. Navy dots scattered above and below the zero line with no visible pattern $\Rightarrow$ model is well-specified. A fanning/curving pattern would reveal a missing structural term.



Always plot residuals — clustered residuals in finance suggest regime changes (e.g., pre- vs post-COVID volatility)

Regression Formula Sheet

OLS Prediction

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Slope

$$\beta_1 = \frac{\text{Cov}(x, y)}{\text{Var}(x)}$$

Intercept

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

CAPM

$$R_i - R_f = \alpha + \beta(R_m - R_f) + \epsilon$$

MSE

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

R-Squared

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Ridge

$$\mathcal{L} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \beta_j^2$$

Lasso

$$\mathcal{L} = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$$

Eight core regression formulas — the OLS slope and CAPM beta use the same formula applied to different data

Classification Formula Sheet

Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Logit (Log-Odds)

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Odds Ratio

$$\text{OR}_j = e^{\beta_j}$$

Gini Impurity

$$\text{Gini}(S) = 1 - \sum_{i=1}^{C} p_i^2$$

Entropy

$$H(S) = -\sum_{i=1}^{C} p_i \log_2 p_i$$

Precision

$$P = \frac{TP}{TP + FP}$$

Recall

$$R = \frac{TP}{TP + FN}$$

F1 Score

$$F_1 = \frac{2PR}{P + R}$$

Eight classification formulas — precision and recall definitions appear on every exam, so write them from the confusion matrix

Common Misconceptions (Part 1)

Myth 1: "High R^2 means a good model"

Reality: Overfitted models have high R^2 on training data. Always check R^2 on the test set. In finance, $R^2 = 0.02$ out-of-sample can be very profitable.

Myth 2: "99% accuracy means a good classifier"

Reality: With 1% fraud rate, a model predicting "no fraud always" gets 99% accuracy and catches nothing. Use precision, recall, and AUC instead.

Myth 3: "More features always help"

Reality: More features increases the risk of overfitting, especially with small datasets. Regularization (Lasso) or feature selection is essential.

These three myths account for most beginner mistakes — a hedge fund with 500 features and 200 observations is almost guaranteed to overfit

Common Misconceptions (Part 2)

Myth 4: "Trees are always better than logistic regression"

Reality: Logistic regression often wins on small, clean, linear datasets. Trees shine with non-linear boundaries and interactions. Regulated industries require interpretability.

Myth 5: "$\beta = 1.5$ means the stock WILL move 1.5x"

Reality: Beta is an average relationship estimated from historical data. It describes tendency, not certainty. Standard errors and confidence intervals matter.

Myth 6: "Lasso is always better than Ridge"

Reality: Lasso is better when few features matter (sparse signal). Ridge is better when many features contribute small amounts. Elastic Net combines both.

There is no universally best model — a credit scorecard (logistic regression) beats a random forest when the regulator demands explainability

Self-Assessment (Part 1)

Question 1: A stock has $\beta = 0.8$ in a CAPM regression. The market rises 10% tomorrow. What is your *expected* excess return for the stock?

Self-Assessment (Part 1)

Question 1: A stock has $\beta = 0.8$ in a CAPM regression. The market rises 10% tomorrow. What is your *expected* excess return for the stock?

Answer: $0.8 \times 10\% = 8\%$ expected excess return. Not guaranteed – just the OLS prediction.

Question 2: Your fraud model has Precision $= 0.90$ and Recall $= 0.30$. What does this mean in plain language?

Self-Assessment (Part 1)

Question 1: A stock has $\beta = 0.8$ in a CAPM regression. The market rises 10% tomorrow. What is your *expected* excess return for the stock?

Answer: $0.8 \times 10\% = 8\%$ expected excess return. Not guaranteed – just the OLS prediction.

Question 2: Your fraud model has Precision $= 0.90$ and Recall $= 0.30$. What does this mean in plain language?

Answer: 90% of flagged transactions are real fraud (few false alarms), but you only catch 30% of all fraud (many cases slip through). You need to lower the threshold to increase recall.

Test yourself: can you explain these to a non-technical manager at a bank?

Self-Assessment (Part 2)

Question 3: You have 500 features and 200 observations. Should you use Ridge or Lasso?

Self-Assessment (Part 2)

Question 3: You have 500 features and 200 observations. Should you use Ridge or Lasso?

Answer: Lasso – it will zero out irrelevant features, effectively reducing dimensionality. With $p > n$, you need sparsity. (Elastic Net is also a good choice.)

Question 4: A medical test has 99% accuracy on a disease affecting 0.1% of the population. You test positive. What is the approximate probability you actually have the disease?

Self-Assessment (Part 2)

Question 3: You have 500 features and 200 observations. Should you use Ridge or Lasso?

Answer: Lasso – it will zero out irrelevant features, effectively reducing dimensionality. With $p > n$, you need sparsity. (Elastic Net is also a good choice.)

Question 4: A medical test has 99% accuracy on a disease affecting 0.1% of the population. You test positive. What is the approximate probability you actually have the disease?

Answer: Only about 9% (Bayes' theorem). The base rate is so low that most positives are false positives. This is the accuracy trap applied to medicine.

If you got all four right, you have solid supervised learning intuition — the base-rate problem applies to fraud detection too

What Comes Next?

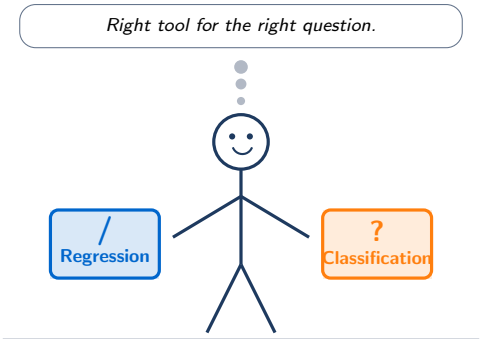
Deep-dive companions (one per topic):

- L21: Linear Regression & OLS
- L22: Ridge, Lasso, Elastic Net
- L23: Regression Metrics (R^2, MSE, cross-validation)
- L24: Factor Models (CAPM, Fama–French)
- L25: Logistic Regression & Credit Scoring
- L26: Decision Trees & Random Forests
- L27: Classification Metrics (F1, AUC, ROC)
- L28: Class Imbalance & SMOTE

Looking ahead:

- L29–L32: Unsupervised Learning (clustering, PCA)
- L33–L36: Deep Learning (neural networks, CNNs)

This overview is your map — the lesson decks are the detailed terrain covering 8 weeks of supervised learning


Right tool for the right question.
/
Regression
?
Classification