Elements of Machine Learning and Data Science

Part I: Data Science — Exam Notes (Living Document)

Emir Pisirici

January 29, 2026

Exam likelihood: High (overall Data Science part)

This document is structured to match the lecture topics exactly and is designed for adding **exam-style notes**, **common traps**, and **visual summaries**.

Contents

1 Introduction to Data Science

1.1 Introduction

1.2 Tabular Data

1.3 Data Science Process

Exam likelihood: High

Framework questions are easy to grade and strongly test “big picture” understanding.

Examiner favorite (what they love to ask)

Typical asks: **ETL vs ELT**, **CRISP-DM phases**, and mapping a scenario to the correct phase. Also: where data leakage/bias lives (data understanding + evaluation).

1.3.1 ETL vs ELT (Definitions + Differences)

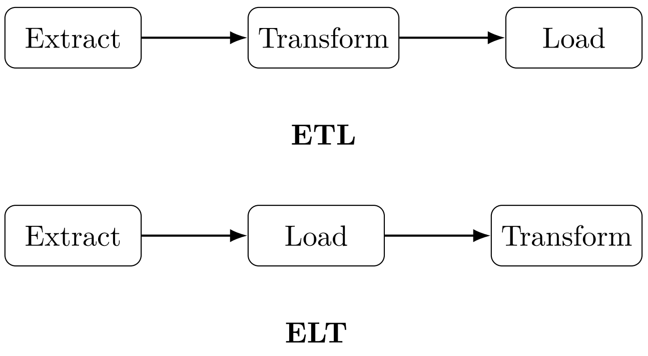
Cheat sheet / must-memorize

ETL: Extract → Transform → Load (transform before target).
ELT: Extract → Load → Transform (transform inside target platform).
Key contrast: where transformations happen; governance vs flexibility; raw history availability.

Common pitfall

People confuse “ELT = no cleaning”. Wrong. It means cleaning happens *after loading*, often in warehouse/lakehouse layers (staging → curated).

Visual

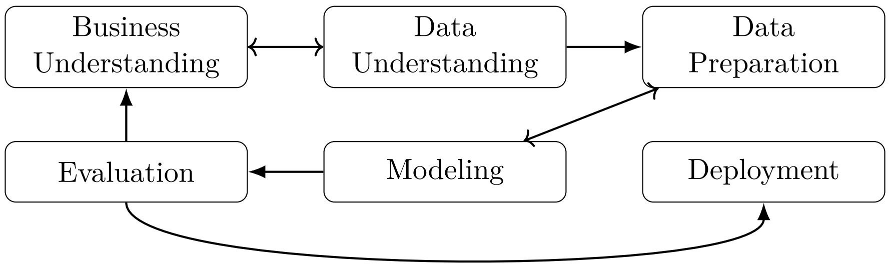


1.3.2 CRISP-DM

Cheat sheet / must-memorize

CRISP-DM: Business Understanding → Data Understanding → Data Preparation → Modeling → Evaluation → Deployment (iterative loops).

Visual



1.3.3 PDCA

Cheat sheet / must-memorize

PDCA: Plan → Do → Check → Act (continuous improvement loop).

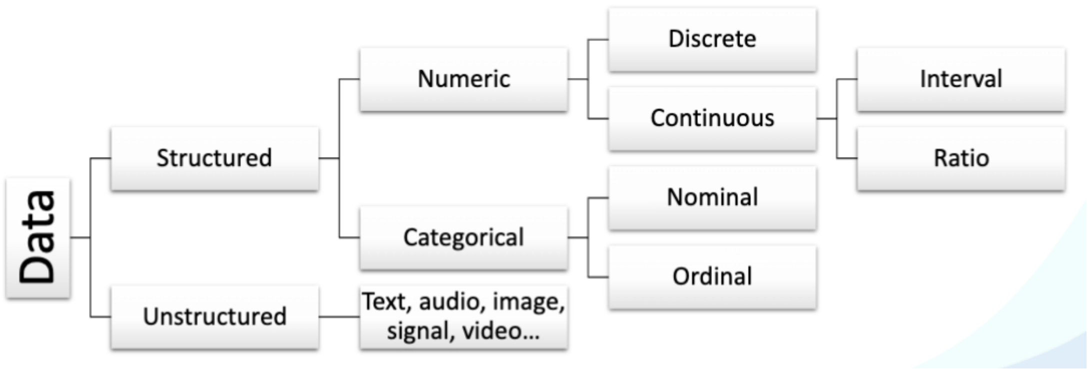
1.3.4 DMAIC

Cheat sheet / must-memorize

DMAIC: Define → Measure → Analyze → Improve → Control. Often used for process/quality improvement + monitoring and part of the Six Sigma methodology.

1.4 Data Types

Visual



1.5 Descriptive Statistics

Exam likelihood: High

Frequent: compute variance/STD/covariance/correlation by hand; read a correlation matrix.

Examiner favorite (what they love to ask)

Explain why covariance depends on units, and why correlation is normalized in $[-1, 1]$.

Why (motivation): Quantify spread and association between variables.
What (definition): Variance/STD measure spread; covariance/correlation measure linear association.
How (procedure/usage): Compute formulas, then interpret sign/magnitude and check the correlation matrix.

Cheat sheet / must-memorize

- **Variance (sample):** $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
- **Std dev:** $s = \sqrt{s^2}$
- **Covariance (sample):** $\mathrm{cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$
- **Correlation:** $r = \frac{\mathrm{cov}(X,Y)}{s_X s_Y}$ (unitless, -1 to 1)
- **Correlation matrix:** table of pairwise correlations; symmetric with 1s on the diagonal.

Common pitfall

Correlation $\neq$ causation; a strong correlation can be driven by a confounder or Simpson's paradox.

Visual

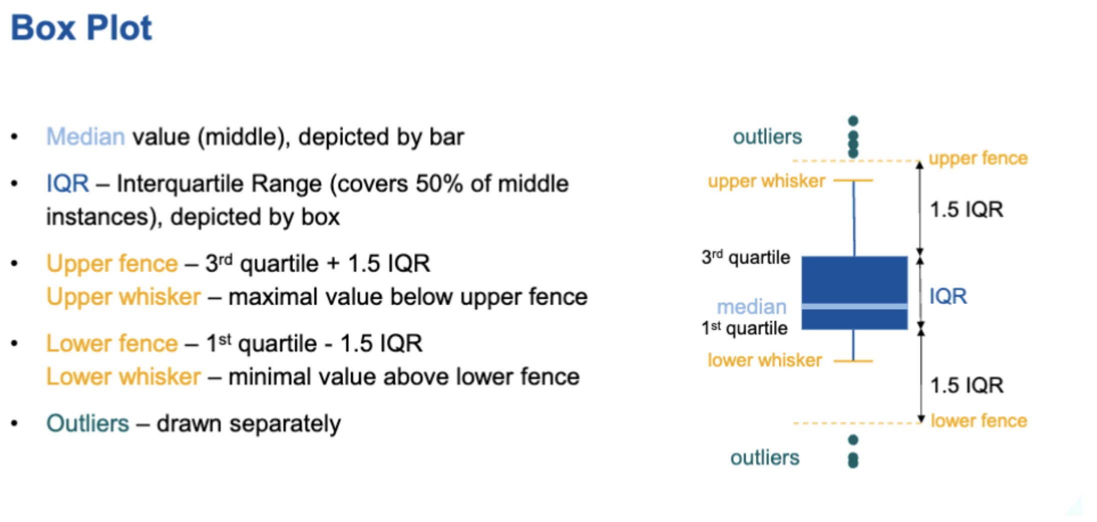
1	r_{12}	r_{13}
r_{21}	1	r_{23}
r_{31}	r_{32}	1

Correlation matrix

Key takeaways: Know formulas + interpretations; correlation matrix is symmetric with 1s on the diagonal.

1.6 Basic Visualizations

Visual



Box Plot

- Median value (middle), depicted by bar
- IQR – Interquartile Range (covers 50% of middle instances), depicted by box
- Upper fence – 3rd quartile + 1.5 IQR
  Upper whisker – maximal value below upper fence
- Lower fence – 1st quartile - 1.5 IQR
  Lower whisker – minimal value above lower fence
- Outliers – drawn separately

1.7 Feature Transformations

Exam likelihood: High

Typical: pick the right transform (scale, log, encode) and explain why.

Examiner favorite (what they love to ask)

Identify data leakage in preprocessing; name the correct order for train/test transformations.

Why (motivation): Turn raw categorical/continuous variables into model-ready features.
What (definition): Encoding or discretizing features without changing the target meaning.
How (procedure/usage): Choose encoding by category type; choose binning by distribution.

Cheat sheet / must-memorize

- **One-hot encoding:** create a 0/1 column per category (nominal).
- **Binary encoding:** represent categories as binary digits (compact one-hot).
- **Ordinal encoding:** map ordered categories to ranks (only if order is real).
- **Binning:** convert continuous to categories.
- **Equal-width binning:** fixed interval sizes across the range.
- **Equal-frequency binning:** same number of samples per bin.

Common pitfall

Fitting transforms on the full dataset (leakage). Always fit on training data, then apply to validation/test.

Key takeaways: Use one-hot for nominal, ordinal for ordered labels, and binning for simplification.

1.8 "How to lie with statistics"

2 Decision Trees

2.1 Introduction to Decision Trees

Exam likelihood: High

Intro questions often ask you to explain how trees split data and what leaves represent.

Examiner favorite (what they love to ask)

Draw a small tree from a toy dataset or explain interpretability vs overfitting.

Why (motivation): Learn a function from labeled training instances to make predictions.
What (definition): A tree partitions the feature space by sequential if-then splits; leaves output a class or value.
How (procedure/usage): Choose splits to improve class purity or reduce prediction error.

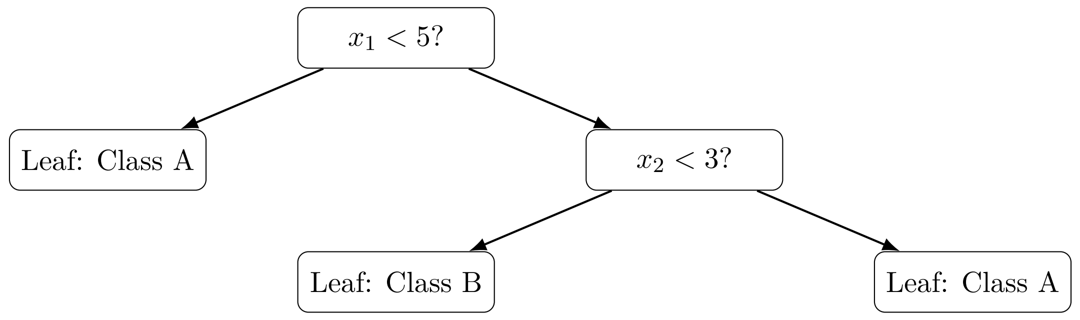
Cheat sheet / must-memorize

- **Goal:** learn a function $f(X)$ from labeled data to predict labels/values.
- **Tree structure:** root node, internal (non-leaf) nodes, leaf nodes.
- **Split rule:** one feature + threshold; paths are if-then rules.
- **Leaf meaning:** prediction (class/value) for that region of the space.

Common pitfall

Overly deep trees memorize training data; control with max depth, min samples, or pruning.

Visual



Key takeaways: Trees learn f from labeled data using splits; nodes/leaf roles are core.

2.2 Entropy and Information Gain

Exam likelihood: Very High

Almost guaranteed: compute entropy and information gain for candidate splits.

Examiner favorite (what they love to ask)

Given a small labeled dataset, compare splits and pick the one with highest information gain.

Why (motivation): Choose splits that make child nodes as pure as possible.
What (definition): Entropy measures impurity; information gain is impurity reduction.
How (procedure/usage): Compute parent entropy, child entropies, then IG = parent - weighted children.

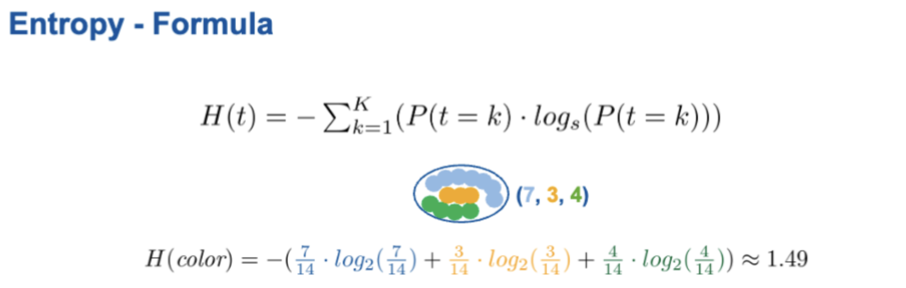
Cheat sheet / must-memorize

- **Entropy:** $H(S) = -\sum_c p_c \log_2 p_c$ (define $0 \log 0 = 0$).
- **Weighted child entropy:** $\sum_k \frac{|S_k|}{|S|} H(S_k)$.
- **Information gain:** $\mathrm{IG}(S, \text{split}) = H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k)$.
- **Goal:** choose split with highest IG (most impurity reduction).

Common pitfall

Forgetting to weight child entropies by subset size; using raw entropy sums gives wrong IG.

Visual

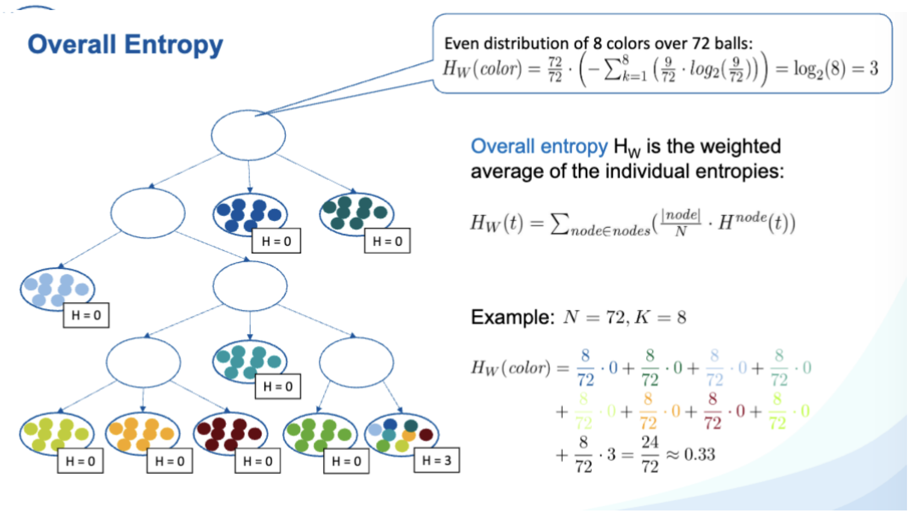


Entropy - Formula

$$H(t) = -\sum_{k=1}^{K} (P(t=k) \cdot log_s(P(t=k)))$$

(7, 3, 4)

$$H(color) = -(\tfrac{7}{14} \cdot log_2(\tfrac{7}{14}) + \tfrac{3}{14} \cdot log_2(\tfrac{3}{14}) + \tfrac{4}{14} \cdot log_2(\tfrac{4}{14})) \approx 1.49$$



Overall Entropy

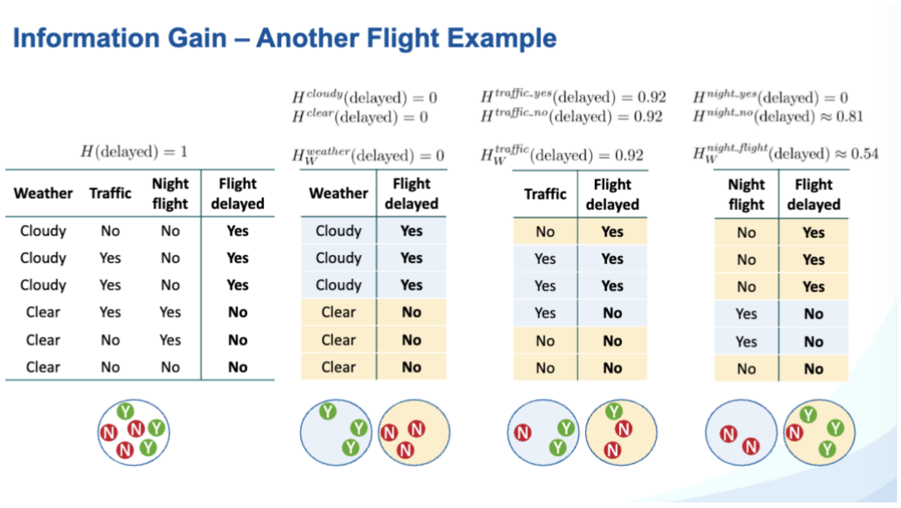
Even distribution of 8 colors over 72 balls:
$H_W(color) = \frac{72}{72} \cdot \left(-\sum_{k=1}^{8} \left(\frac{9}{72} \cdot log_2(\frac{9}{72})\right)\right) = \log_2(8) = 3$

Overall entropy H_W is the weighted average of the individual entropies:

$$H_W(t) = \sum_{node \in nodes} (\frac{|node|}{N} \cdot H^{node}(t))$$

Example: $N = 72, K = 8$

$$H_W(color) = \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 0 + \frac{8}{72} \cdot 3 = \frac{24}{72} \approx 0.33$$



Information Gain – Another Flight Example

$H(\text{delayed}) = 1$

Weather	Traffic	Night flight	Flight delayed
Cloudy	No	No	Yes
Cloudy	Yes	No	Yes
Cloudy	Yes	No	Yes
Clear	Yes	Yes	No
Clear	No	Yes	No
Clear	No	No	No

$H^{cloudy}(\text{delayed}) = 0$
$H^{clear}(\text{delayed}) = 0$

$H_W^{weather}(\text{delayed}) = 0$

Weather	Flight delayed
Cloudy	Yes
Cloudy	Yes
Cloudy	Yes
Clear	No
Clear	No
Clear	No

$H^{traffic\_yes}(\text{delayed}) = 0.92$
$H^{traffic\_no}(\text{delayed}) = 0.92$

$H_W^{traffic}(\text{delayed}) = 0.92$

Traffic	Flight delayed
No	Yes
Yes	Yes
Yes	Yes
Yes	No
No	No
No	No

$H^{night\_yes}(\text{delayed}) = 0$
$H^{night\_no}(\text{delayed}) \approx 0.81$

$H_W^{night\_flight}(\text{delayed}) \approx 0.54$

Night flight	Flight delayed
No	Yes
No	Yes
No	Yes
Yes	No
Yes	No
No	No

Key takeaways: Compute entropy, weight children, pick split with highest IG.

Exam likelihood: Very High

Almost guaranteed: compute entropy / information gain on a small dataset.

2.3 ID3 Algorithm

Exam likelihood: Very High

Common: list the ID3 steps or run one iteration to choose the best split.

Examiner favorite (what they love to ask)

Explain stopping conditions and why ID3 prefers high information gain.

Why (motivation): Build a decision tree that best separates labeled data.
What (definition): ID3 is a greedy, top-down tree induction algorithm using information gain.
How (procedure/usage): Compute IG for each attribute, split on the best, and recurse.

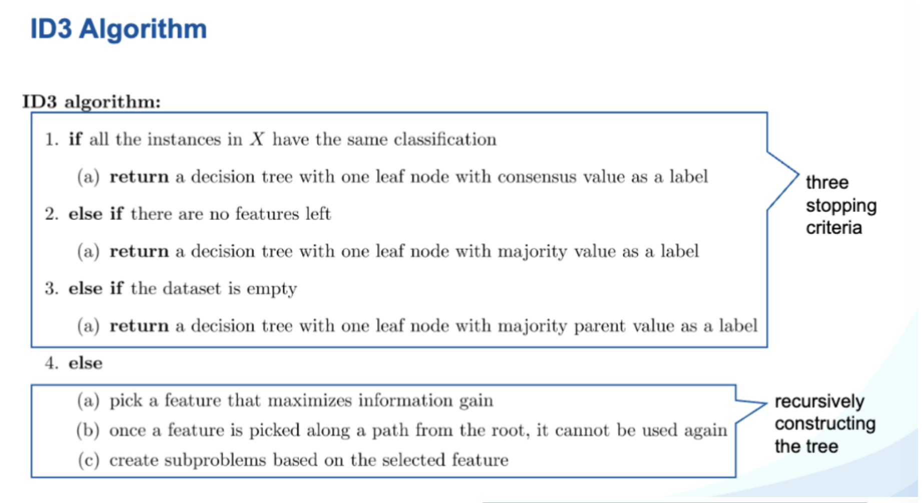
Cheat sheet / must-memorize

- **Input:** labeled dataset S with categorical attributes (ID3 original).
- **Step 1:** if all labels same $\rightarrow$ make a leaf.
- **Step 2:** if no attributes left $\rightarrow$ leaf with majority class.
- **Step 3:** choose attribute with highest IG.
- **Step 4:** split S by attribute values and recurse.
- **Output:** a decision tree; leaves store class label.

Common pitfall

ID3 favors attributes with many values; without corrections (e.g., gain ratio) it can overfit.

Visual



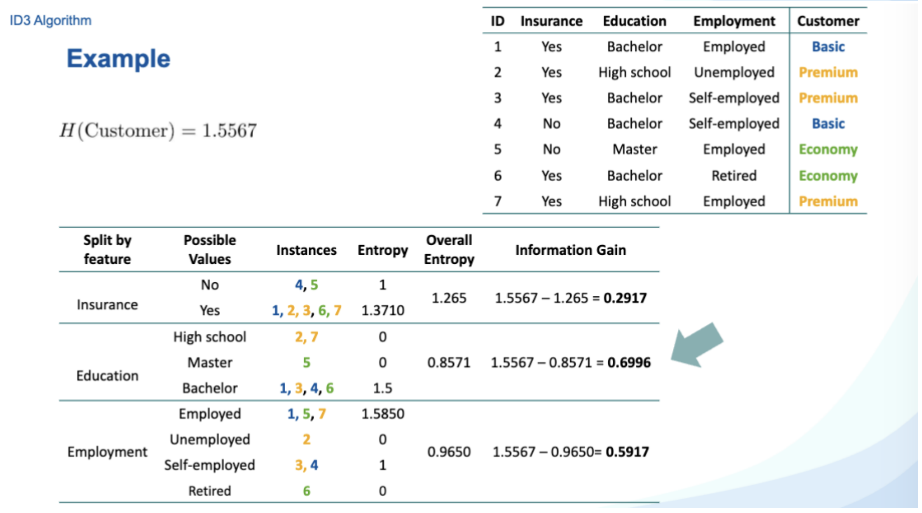
ID3 Algorithm

ID3 algorithm:

```
1. if all the instances in $X$ have the same classification
    (a) return a decision tree with one leaf node with consensus value as a label
2. else if there are no features left
    (a) return a decision tree with one leaf node with majority value as a label
3. else if the dataset is empty
    (a) return a decision tree with one leaf node with majority parent value as a label
4. else
    (a) pick a feature that maximizes information gain
    (b) once a feature is picked along a path from the root, it cannot be used again
    (c) create subproblems based on the selected feature
```

three stopping criteria

recursively constructing the tree



ID3 Algorithm

Example

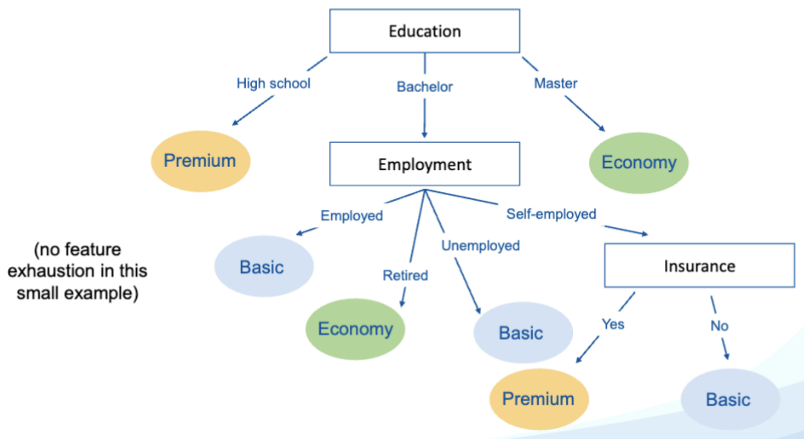
$H(\text{Customer}) = 1.5567$

ID	Insurance	Education	Employment	Customer
1	Yes	Bachelor	Employed	Basic
2	Yes	High school	Unemployed	Premium
3	Yes	Bachelor	Self-employed	Premium
4	No	Bachelor	Self-employed	Basic
5	No	Master	Employed	Economy
6	Yes	Bachelor	Retired	Economy
7	Yes	High school	Employed	Premium

Split by feature	Possible Values	Instances	Entropy	Overall Entropy	Information Gain
Insurance	No	4, 5	1	1.265	1.5567 – 1.265 = **0.2917**
	Yes	1, 2, 3, 6, 7	1.3710		
Education	High school	2, 7	0	0.8571	1.5567 – 0.8571 = **0.6996**
	Master	5	0		
	Bachelor	1, 3, 4, 6	1.5		
Employment	Employed	1, 5, 7	1.5850	0.9650	1.5567 – 0.9650= **0.5917**
	Unemployed	2	0		
	Self-employed	3, 4	1		
	Retired	6	0		

ID3 Algorithm

Example



Key takeaways: ID3 is greedy; compute IG, split, recurse, stop with pure/majority leaves.

2.4 Quantifying Information Gain

Exam likelihood: Very High

Often compute IG for a specific split and compare candidate attributes.

Examiner favorite (what they love to ask)

Show all intermediate steps: parent entropy, each child entropy, weighted sum, IG.

Why (motivation): Convert "best split" into a concrete, comparable number.
What (definition): IG = parent entropy minus weighted child entropies; Split Info measures how evenly the split divides data; Gain Ratio normalizes IG.
How (procedure/usage): Compute IG, then divide by split info to get gain ratio.

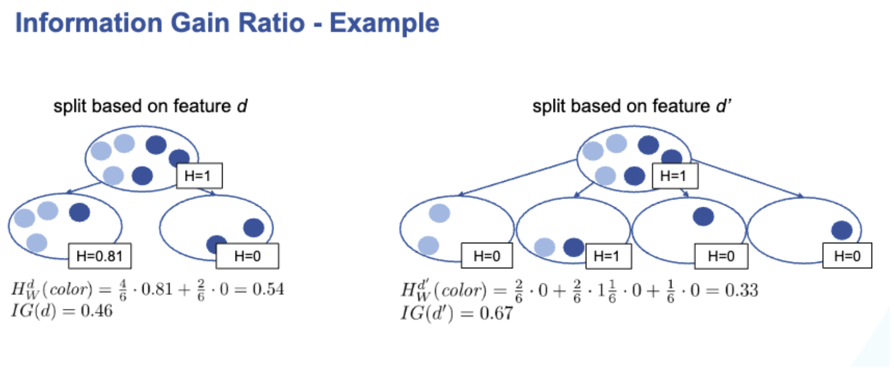
Cheat sheet / must-memorize

- **Step 1:** compute parent entropy $H(S)$.
- **Step 2:** split by attribute values.
- **Step 3:** compute each child entropy $H(S_k)$.
- **Step 4:** compute weighted sum $\sum_k \frac{|S_k|}{|S|} H(S_k)$.
- **Step 5:** IG $= H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k)$.
- **Split info (lecture: $H(d)$):** entropy of split proportions (how evenly data is partitioned). $H(d) = SI = -\sum_k \frac{|S_k|}{|S|} \log_2 \frac{|S_k|}{|S|}$.
- **Gain ratio:** $GR = \frac{IG}{H(d)}$ (same as IG/SI; penalizes many-valued attributes).

Common pitfall

Information gain is biased toward attributes with many values; use gain ratio to correct. Also: weight by subset size and keep log base consistent.

Visual



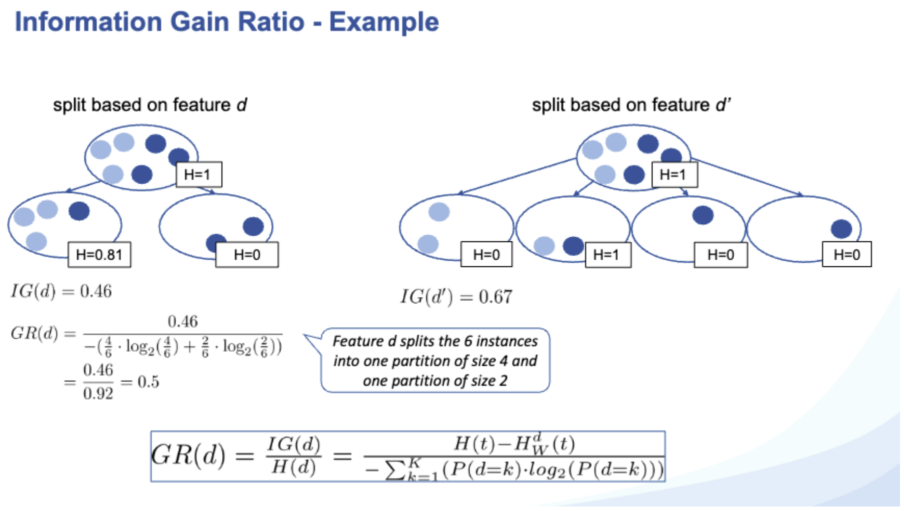
Information Gain Ratio - Example

split based on feature *d*

split based on feature *d'*

$H_W^d(color) = \frac{4}{6} \cdot 0.81 + \frac{2}{6} \cdot 0 = 0.54$
$IG(d) = 0.46$

$H_W^{d'}(color) = \frac{2}{6} \cdot 0 + \frac{2}{6} \cdot 1 \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 0 = 0.33$
$IG(d') = 0.67$



Information Gain Ratio - Example

split based on feature *d*

split based on feature *d'*

$IG(d) = 0.46$

$IG(d') = 0.67$

$$GR(d) = \frac{0.46}{-(\frac{4}{6} \cdot \log_2(\frac{4}{6}) + \frac{2}{6} \cdot \log_2(\frac{2}{6}))} = \frac{0.46}{0.92} = 0.5$$

Feature d splits the 6 instances into one partition of size 4 and one partition of size 2

$$\boxed{GR(d) = \frac{IG(d)}{H(d)} = \frac{H(t) - H_W^d(t)}{-\sum_{k=1}^{K}(P(d=k) \cdot log_2(P(d=k)))}}$$

Key takeaways: IG is a weighted impurity reduction; higher is better.

2.5 Pruning

Exam likelihood: High

Often: explain why pruning reduces overfitting and name pre- vs post-pruning.

Examiner favorite (what they love to ask)

Given a tree, identify which branches to prune using validation error or complexity.

Why (motivation): Reduce overfitting by simplifying a deep tree.
What (definition): Pruning removes splits/branches that do not improve generalization.
How (procedure/usage): Stop early (pre-pruning) or cut back after training (post-pruning).

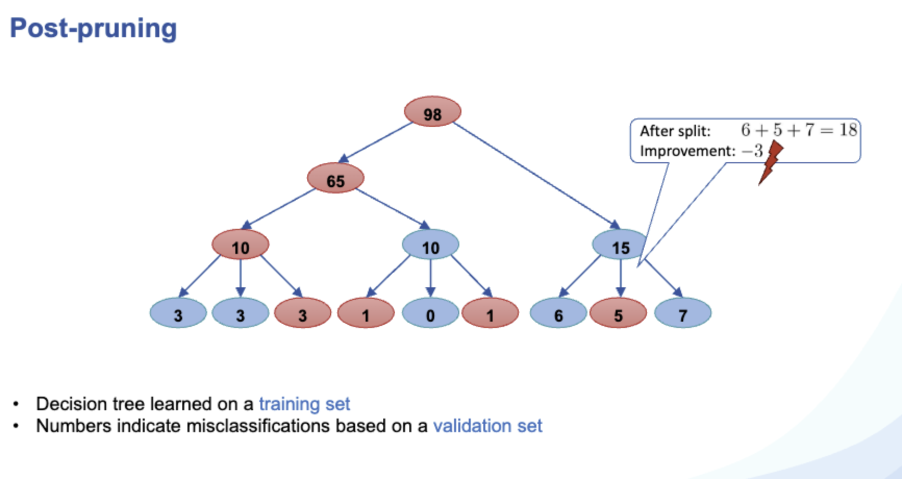
Cheat sheet / must-memorize

- **Pre-pruning:** stop splitting early using rules like: max depth, min samples per node, min impurity decrease, min samples per leaf.
- **Post-pruning:** grow full tree, then prune using validation error or cost-complexity.
- **Goal:** simpler tree with similar or better validation performance.

Common pitfall

Pruning too aggressively can underfit; always tune on validation data, not test data.

Visual



Post-pruning

- Decision tree learned on a training set
- Numbers indicate misclassifications based on a validation set

Key takeaways: Pruning trades depth for generalization; use validation to choose.

2.6 Continuous Data (Threshold splits)

2.7 Ensembles (Bagging/Random Forest/Boosting)

3 Clustering

3.1 Introduction to Unsupervised Learning

3.2 Introduction to Clustering

3.3 Similarity and Dissimilarity

3.4 K-means and K-medoids

3.5 Agglomerative Clustering

3.6 DBSCAN

3.7 Closing

4 Frequent Itemsets

4.1 Introduction

4.2 Properties of Frequent Itemsets

4.3 Apriori Algorithm

4.4 FP-Growth Algorithm

5 Association Rules

5.1 Introduction

5.2 Generating Association Rules

5.3 Evaluation (support, confidence, lift, conviction)

5.4 Applications

5.5 Simpson's Paradox

6 Time Series

6.1 Temporal Data

6.2 Introduction to Time Series

6.3 Analysis

6.4 Forecasting