# MACHINE LEARNING AND REAL-WORLD DATA

## Complete Tripos Revision Guide — Paper 3

Computer Science Part IA | University of Cambridge

Based on past papers 2022–2025 | All gaps filled

### Topics covered in this guide

1. Naive Bayes Classification — full worked examples, smoothing, bias detection
2. Hidden Markov Models — parameter estimation, Viterbi (all question variants)
3. Networks & Graph Theory — degree, clustering coeff., betweenness, Newman-Girvan
4. Unsupervised Clustering — K-Means, K-Means++, evaluation metrics
5. Evaluation, Cross-validation, Sign Test — all worked numerically
6. Zipf's Law & Heaps' Law — formulae, log-log estimation, connection to smoothing
7. Inter-Annotator Agreement — Cohen's & Fleiss' Kappa, full worked examples
8. Ethics, Bias, Real-World Failures — case studies, Asimov, AGI
9. Past Paper Pattern Library — 2022–2025, model answer templates

# 1 Naive Bayes Classification

**Core idea**

Find the class with highest posterior probability P(c|document).
The 'naive' assumption: all features (words) are independent given the class.
This lets us decompose the joint probability into a product of individual likelihoods.

## 1.1 The Classification Rule

Starting from Bayes' Theorem:

```
P(c|d)  =  P(c) * P(d|c)  /  P(d)

Since P(d) is constant for all classes, we drop it and get:

   c* = argmax  P(c) * PRODUCT P(wi | c)
         c in C          i=1..n

In log-space to prevent floating-point underflow:

   c* = argmax  [ log P(c)  +  SUM  log P(wi | c) ]
         c in C              i=1..n
```

**Underflow**

Multiplying many small probabilities together quickly reaches floating-point zero.
Always sum log-probabilities in practice. The argmax result is identical.

## 1.2 Parameter Estimation

### Prior probability

```
P(c)  =  N_c  /  N_total

N_c     = number of training documents of class c
N_total = total training documents
```

### Likelihood — Maximum Likelihood Estimate (MLE, no smoothing)

```
P(w|c)  =  count(w, c)  /  SUM count(w', c)
                             w' in V

Problem: if word w never appears with class c in training,
then P(w|c) = 0, making the entire product = 0.
Even one unseen word kills the classifier.
```

### Laplace (Add-One) Smoothing — use this unless told otherwise

```
P(w|c)  =  [count(w, c) + 1]  /  [SUM count(w', c)  +  |V|]
                                   w' in V
```

```
|V| = total number of unique word types across ALL training data

Effect: shifts a small amount of probability mass from seen to unseen
words.
Every word type, even unseen ones, now has P > 0.
```

**Exam rule**

Apply smoothing whenever the question doesn't say 'without smoothing'.
Always state |V| explicitly — it's the denominator correction.

### General smoothing parameter omega

```
P(w|c)  =  [count(w, c) + omega]  /  [SUM count(w', c)  +  omega * |V|]

Laplace smoothing is the special case omega = 1.
omega can be tuned on the development set.
```

## 1.3 Full Worked Example — Football Academy (2024 Q9)

Training data: 6 recruits with features Goals (Many/Few/None), Position, Gender, and label Success (Y/N).
Step 1 — Compute priors:

```
P(Y) = 3/6 = 0.5        P(N) = 3/6 = 0.5
```

Step 2 — Compute likelihoods (no smoothing, as given):

```
P(Goals=Many | Y) = 1/3      P(Goals=Many | N) = 0/3 = 0    <- PROBLEM
P(Goals=Few  | Y) = 1/3      P(Goals=Few  | N) = 2/3
P(Goals=None | Y) = 1/3      P(Goals=None | N) = 1/3

P(Position=Attack    | Y) = 2/3      P(Position=Attack    | N) = 0
P(Position=Defender  | Y) = 1/3      P(Position=Defender  | N) = 2/3
P(Position=Goalkeeper| Y) = 0        P(Position=Goalkeeper| N) = 1/3

P(Gender=M | Y) = 3/3 = 1            P(Gender=M | N) = 2/3
P(Gender=F | Y) = 0/3 = 0            P(Gender=F | N) = 1/3
```

**Bias demonstration (exam question type)**

Test instance: Goals=None, Position=Goalkeeper, Gender=M
P(Y | instance) proportional to 0.5 * (1/3) * 0 * 1 = 0  (Goalkeeper never in Y)
Classifier can NEVER predict Y for a Goalkeeper — regardless of other features.

Test instance: Goals=Many, Position=Attack, Gender=F
P(Y | instance) proportional to 0.5 * (1/3) * (2/3) * 0 = 0  (Female never in Y)
Classifier can NEVER predict Y for a Female player.

These two groups (Goalkeepers, Female players) are systematically excluded.

## 1.4 Detecting Bias — The Method

To mathematically demonstrate that a classifier is biased against a group:

1. Identify a protected attribute (e.g., Gender, Ethnicity).

2. Find a value of that attribute that ONLY appears in one class in training data.
3. Construct a test instance where all OTHER features are neutral (appear in both classes), but the protected attribute takes the biased value.
4. Show that the classifier predicts the same class regardless of other features, because P(protected_value | other_class) = 0.

The useful property: Naive Bayes is INTERPRETABLE. You can inspect every parameter and identify the bias directly from the probability table. This is impossible with black-box models. The problematic property: zero probabilities propagate through the product and make the classifier completely insensitive to all other evidence — one zero kills everything.

## 1.5 Feature Engineering — Improving Generalisation

| Technique | What it does | Why it helps |
|---|---|---|
| Lowercasing | 'Running' and 'running' become the same type | Reduces vocabulary size, less sparsity |
| Stemming / Lemmatisation | 'argued','arguing','argues' -> 'argu' | Groups morphological variants together |
| Stop-word removal | Remove 'the', 'and', 'is', etc. | Removes high-frequency noise with no discriminative power |
| Hapax removal | Drop words occurring only once (hapax legomena) | Removes noise; rare words rarely generalise |
| Lexicon integration | 'awful' -> <NEGATIVE_WORD> token | Handles unseen synonyms; improves generalisation across language change |
| Binning continuous features | Age 18 -> [17-20] bracket | Reduces sparsity for numerical features |

## 1.6 Naive Bayes vs. Lexicon-Based Approach

| | Naive Bayes wins | Lexicon wins |
|---|---|---|
| Data | Sufficient labelled training data exists | Very little labelled data available |
| Domain | Domain language differs from standard lexicons | Domain well-covered by existing lexicon |
| Interpret. | When statistical patterns capture subtle discrimination | When you need explicit, inspectable rules |
| Best of both | Use lexicon tokens as NB features: replace words with <POS>/<NEG> tokens before training | — |

# 2 Statistical Laws of Language — Zipf & Heaps

> **Why this matters for NB**
> Zipf's and Heaps' Laws explain WHY the unsmoothed classifier fails and why smoothing helps.
> They tell us vocabulary is effectively infinite — we will ALWAYS encounter unseen words.

## 2.1 Zipf's Law

Word frequency is inversely proportional to frequency rank. A small number of words are very frequent; a huge number are rare.

```
Basic Zipf:     f_w  ~  k / r_w

General form:   f_w  ~  k / (r_w + beta)^alpha

f_w    = frequency of word w in corpus
r_w    = frequency rank of word w (rank 1 = most frequent)
alpha  = exponent (~1.0 for English, ~1.3 for German)
k, beta = language-specific constants
```

### Estimating alpha and k

Take log of both sides of the basic form:

```
log(f_w)  =  log(k)  -  alpha * log(r_w)

Plot log(frequency) vs log(rank) -> straight line.
Slope of the line = -alpha
Y-intercept = log(k)  ->  k = exp(y-intercept)

Fit using least-squares regression on the log-log plot.
```

Consequences for NB: the 'long tail' contains many rare words. Even a large training corpus misses many of them. The MLE probability for unseen words is 0 — this is exactly what Laplace smoothing corrects.

## 2.2 Heaps' Law

As corpus size grows, vocabulary keeps growing — but with diminishing returns.

```
Heaps' Law:   |V|  =  k * N^beta

|V|  = vocabulary size (number of unique types)
N    = total number of tokens in the corpus
beta ~ 0.5 (typically between 0.4 and 0.6)
k    ~ 30 to 100 (language-dependent)

Log-log form:  log|V|  =  log(k)  +  beta * log(N)
Plot log(vocab) vs log(tokens) -> straight line with slope beta.
```

Consequence: no matter how much training data you collect, you will always encounter new word types in new text. Smoothing is not optional — it is a mathematical necessity.

# 3 Hidden Markov Models

**Core idea**

Model sequences where we observe outputs (emissions) but not the underlying process (hidden states).
Example: we see weather observations (Snow/Rain/Dry) but not air temperature (Freezing/NotFreezing).
Unlike Naive Bayes, HMMs model dependencies between consecutive items in a sequence.

## 3.1 Formal Definition: mu = (S, V, A, B, pi)

| Symbol | Name | Description |
|---|---|---|
| S | Hidden States | The unobservable states, e.g. {Freezing, NotFreezing} or {L, M, H} |
| V | Observations | The observable emissions, e.g. {Snow, Rain, Dry} |
| A | Transition Matrix | a_ij = P(q_t = s_j \| q_{t-1} = s_i) — moving between states |
| B | Emission Matrix | b_i(v_k) = P(o_t = v_k \| q_t = s_i) — emitting an observation from a state |
| pi | Initial Distribution | pi_j = P(q_1 = s_j) — probability of starting in each state |

## 3.2 Three Implicit Assumptions — always state all three

**Exam pattern**

This sub-question appears every single year. You must state all three and evaluate each in context.

1. First-Order Markov: P(q_t | q_1...q_{t-1}) = P(q_t | q_{t-1}). Only the immediately preceding state matters. FAILS when trends depend on longer history (e.g., multi-day weather patterns).
2. Output Independence: P(o_t | q_1..q_t, o_1..o_{t-1}) = P(o_t | q_t). The observation depends only on the current state, not previous states or observations. FAILS when consecutive observations are correlated.
3. Stationarity: Transition and emission probabilities do not change over time. FAILS when behaviour is seasonal (summer vs. winter temperatures), structural (economic policy changes), or evolutionary.

## 3.3 Parameter Estimation

### Without smoothing (use only when question says so)

```
a_ij  =  count(s_i -> s_j)  /  count(s_i)
b_i(v_k)  =  count(s_i emits v_k)  /  count(s_i)

Problem: any transition or emission with count=0 gives probability 0.
In Viterbi, a zero probability anywhere in a path kills that path
entirely.
```

### With Laplace (Add-One) Smoothing — use by default

```
Transitions:  a_ij  =  [count(s_i -> s_j) + 1]  /  [count(s_i) + |S|]
Emissions:    b_i(v_k)  =  [count(s_i emits v_k) + 1]  /  [count(s_i) +
|V|]

CRITICAL: Denominators are DIFFERENT.
  Transitions denominator: add |S|  (number of hidden states)
  Emissions   denominator: add |V|  (number of observable symbols)
```

**Most common exam error**

Students add |V| to the transition denominator. This is WRONG.
Transitions: +|S|. Emissions: +|V|. They are different quantities.

## 3.4 Full Worked Example — 2022 COVID HMM

Training data (timesteps 1-7): hidden states = infection level L/M/H; observations = positivity +/++/+++.

### Counting transitions from training data

```
State sequence: L  M  H  H  H  M  M

L -> M: 1           M -> H: 1     M -> M: 2    H -> H: 2    H -> M: 1
count(L) = 1        count(M) = 3  count(H) = 3

Without smoothing:   a(L->M) = 1/1 = 1.0    a(M->H) = 1/3   a(M->M) = 2/3
                     a(H->H) = 2/3           a(H->M) = 1/3
                     a(L->L) = 0/1 = 0  [problematic without smoothing]
```

### Counting emissions

```
Observations: +  ++  ++  ++  +++  ++  ++

L emits +:   1/1 = 1.0       M emits ++:  3/3 = 1.0       H emits ++:
2/3
                              M emits +:   0   <- zero      H emits +++:
1/3
                              M emits +++: 0   <- zero
```

### Viterbi for timesteps 8-10: observations +++, ++, ++

Initialisation: use the state distribution at t=7. Most likely at t=7 is M (we are IN that state). So delta_M(7) = 1, delta_L(7) = 0, delta_H(7) = 0.

```
t=8, observation = +++
  delta_L(8) = max[delta_i(7) * a(i->L) * b_L(+++)]:  b_L(+++) = 0 ->
delta_L(8) = 0
  delta_M(8) = max[delta_M(7)*a(M->M)*b_M(+++)] = 1 * 2/3 * 0 = 0  <-
zero emission
              [delta_H(7)*a(H->M)*b_M(+++)] = 0 (delta_H=0) -> 0
  delta_H(8) = max[delta_M(7)*a(M->H)*b_H(+++)] = 1 * 1/3 * 1/3 = 1/9
             backpointer: came from M
  -> Best state at t=8: H  (delta_H=1/9 > all others)

t=9, observation = ++
```

```
  delta_H(9) = delta_H(8)*a(H->H)*b_H(++) = 1/9 * 2/3 * 2/3 = 4/81
  delta_M(9) = delta_H(8)*a(H->M)*b_M(++) = 1/9 * 1/3 * 1 = 1/27 = 3/81
  -> Best at t=9: H  (4/81 > 3/81)

t=10, observation = ++
  delta_H(10)= delta_H(9)*a(H->H)*b_H(++) = 4/81 * 2/3 * 2/3 = 16/729
  delta_M(10)= delta_H(9)*a(H->M)*b_M(++) = 4/81 * 1/3 * 1 = 4/243 =
12/729
  -> Best at t=10: H  (16/729 > 12/729)

Optimal path: t=8 -> H,  t=9 -> H,  t=10 -> H
Backtrack: t10 came from H(t9), t9 came from H(t8), t8 came from M(t7).
```

**Exam layout tip**

Draw a table: rows = states, columns = timesteps. Show every multiplication.
Circle the max in each column and draw an arrow (backpointer) back to its source.
At the end, state the optimal path and its probability explicitly.

## 3.5 Common HMM Shortcomings (2-4 marks each year)

- Stationarity violated — transition probs change with seasons, policy shifts, etc.
- First-order Markov too short — real-world trends persist over longer windows
- Small training set — unreliable parameter estimates (e.g., 7 data points for COVID HMM)
- Discrete hidden states too coarse — continuous reality squeezed into 3 levels
- Output independence violated — consecutive emissions often correlated
- Supervised training requires labelled dual-tape data (both states AND observations known)

## 3.6 Building a Time-Aware HMM (Compound States — 2024 Q8d)

Problem: stationarity fails when transitions differ by season. Solution: create compound states.

```
Original:  S = { Freezing (F), NotFreezing (NF) }
           A = 2x2 matrix

New:       S = { F_Summer, F_Winter, NF_Summer, NF_Winter }
           A = 4x4 matrix

Data transformation:
  Label each month with its season: Oct-Mar = Winter, Apr-Sep = Summer.
  Relabel each state: Oct NF -> NF_Winter, Dec F -> F_Winter, etc.

Re-estimate A and B from the relabelled data.

Result: P(NF_Summer -> F_Summer) << P(NF_Winter -> F_Winter)
The model now captures that freezing is far less likely in summer.
```

# 4 Networks and Graph Theory

## 4.1 Core Measures with Formulae

| Measure | Formula | Intuition |
|---|---|---|
| Degree k_i | Count edges incident to node i | How many direct neighbours |
| Clustering Coeff. C_i | C_i = 2*e_i / [ k_i*(k_i-1) ]  e_i = edges between i's neighbours | Fraction of i's neighbours that are also connected to each other |
| Betweenness CB(v) | CB(v) = SUM_{s!=v!=t} sigma_st(v)/sigma_st  sigma_st = number of shortest paths from s to t sigma_st(v) = those passing through v | How often v lies on shortest paths between other pairs; measures 'bottleneck-ness' |
| Diameter | max_{s,t} d(s,t) | Longest of all shortest paths; size of network |

> **Undirected graphs**
> HALVE all betweenness centrality scores at the end.
> Each path s->t is also counted as t->s, so every path is double-counted.

## 4.2 Worked Example — 2025 Q8 Graph

Graph: A-B, A-E, B-C, B-F, C-D, D-F, D-H, F-G, G-H, C-F (plus E connected only to A)

### Degree

```
k_A = 2  (connected to B, E)
k_C = 3  (connected to B, D, F)
k_H = 2  (connected to D, G)
```

### Clustering Coefficient

```
C_A: k_A=2, e_A = edges between {B,E} = 0 (B and E not connected)
     C_A = 2*0 / (2*1) = 0

C_C: k_C=3, neighbours = {B, D, F}
     Edges among {B,D,F}: B-F (yes), B-D (no), D-F (yes) -> e_C = 2
     C_C = 2*2 / (3*2) = 4/6 = 0.667

C_H: k_H=2, neighbours = {D, G}
     Edge D-G? No. -> e_H = 0
     C_H = 0
```

### Betweenness Centrality (qualitative approach for exam)

For each node, count how many (source, target) pairs have their shortest path passing through it. For undirected, halve at the end.

```
C_A: lies on paths from E to any other node. E has only one neighbour
(A),
     so ALL paths from E go through A.
```

```
     Paths from E to {B,C,D,F,G,H}: 6 paths through A (out of 6 total).
     Rough CB(A) = 6 (before halving) = 3 after halving.

C_H: lies on paths between G-D (via H or via other routes).
     H is not on many shortest paths in the main cluster.
     CB(H) is relatively low.

C_C: lies on paths between B-cluster and D-H cluster.
     Many paths from {A,B,E} to {D,H,G} pass through C or F.
     CB(C) is moderate to high.
```

**Effect of removing D-H and adding C-E (part e)**

```
Remove D-H:
  H now only connects via G -> F -> cluster.
  ALL paths to/from H now go through G, then F or D.
  CB(G) increases (more paths through it).
  CB(H) changes: H becomes a leaf — no paths pass THROUGH it -> CB(H)
decreases.

Add C-E:
  E now has TWO neighbours: A and C.
  Paths from E to some nodes may now go via C directly.
  CB(A) decreases (some E-paths no longer need A).
  CB(C) increases (now also an intermediary for E-paths).
  CB(E) may increase slightly (lies on its own new paths).
```

## 4.3 Diameter Calculation

Find the shortest path between every pair of nodes (BFS from each source). The diameter is the largest of these shortest-path lengths.
For the 2025 graph: the longest shortest path is from E to H. E -> A -> B -> C -> D -> H = 5 steps. Diameter = 5.

## 4.4 Strongly Connected Directed Graphs (2025 Q8f)

```
A directed graph G is strongly connected if:
  for every pair of nodes (u, v), there exists a directed path from u to
v
  AND a directed path from v to u.

Is it possible to orient the 2025 graph to make it strongly connected?
Answer: YES, if and only if every edge belongs to at least one cycle.
  The original undirected graph is connected and has many cycles.
  Orient edges so that every cycle has consistent direction.
  E is a leaf (degree 1, only connected to A).
  Leaf nodes CANNOT be in a strongly connected component if their
  single edge is directed away from them (no return path).
  -> The original graph with the leaf E CANNOT be made strongly
connected.
```

## 4.5 Newman-Girvan Community Detection

> **Critical rule**
> Recalculate edge betweenness after EVERY single edge removal. Never remove multiple edges before recalculating.

1. Compute edge betweenness centrality for ALL edges.
2. Remove the edge with the HIGHEST betweenness (it is the bridge).
3. Recalculate edge betweenness for ALL remaining edges.
4. Repeat until you have the desired number of connected components.

## 4.6 Network Taxonomy

| Type | Properties | Exam context |
|---|---|---|
| Social | Undirected. High clustering coefficients. Dense cliques linked by weak ties (high edge betweenness). Triadic closure. | Facebook, Twitter, friend networks |
| Information | Directed. Bow-tie structure. Massive in-degree disparities. Power-law degree distribution. | The Web, citation networks |
| Technological | Constrained by geography. Max degree capped. Vulnerable to targeted hub attack. | Internet routers, power grids |
| Biological | Directed or undirected. Highly modular. Resilient to random failure. | Protein interaction networks |

## 4.7 Key Phenomena

### Small-World

```
Average path length: L ~ log(N)    (grows slowly with network size)
Clustering coefficient: C >> C_random  (much higher than random graph)

Both conditions must hold simultaneously for small-world classification.
Explanation: a few random long-range 'shortcut' edges collapse average
path length
while dense local clustering is preserved.
```

### Power-Law Degree Distribution (Scale-Free)

```
P(k) ~ k^(-alpha)    where alpha typically 2-3

Implies: few hubs with very high degree; most nodes have very low degree.
Origin: preferential attachment ('rich get richer').
Consequence: robust to random failure (most nodes have low degree);
             fragile to targeted attack on hubs.
```

# 5 Evaluation, Data Splits & Significance Testing

## 5.1 Confusion Matrix and Metrics

Given a 2x2 confusion matrix:

| | System says: POSITIVE | System says: NEGATIVE |
|---|---|---|
| TRUTH: POSITIVE | TP — True Positive (correct positive prediction) | FN — False Negative (missed positive) |
| TRUTH: NEGATIVE | FP — False Positive (wrong alarm) | TN — True Negative (correct negative prediction) |

```
Accuracy  = (TP + TN) / (TP + FP + FN + TN)   <- misleading if classes
imbalanced
Precision = TP / (TP + FP)    <- 'of what I claimed +, how many truly
were +?'
Recall    = TP / (TP + FN)    <- 'of all actual +, how many did I find?'
F1        = 2 * P * R / (P + R)  <- harmonic mean of Precision and Recall
```

| Goal | Prioritise | Example |
|---|---|---|
| Find all positives, tolerate false alarms | High Recall | Disease screening, spam filter, brand attack detection |
| Only flag when very confident | High Precision | Medical treatment decisions, legal evidence flagging |
| Balance both | High F1 | Most NLP classification tasks |

## 5.2 Multi-Class Averaging

```
Macro-averaging:
  Compute metric (e.g. F1) for EACH class independently.
  Average the per-class scores.
  Treats all classes equally regardless of size.
  Best for imbalanced data where minority classes matter.

Micro-averaging:
  Pool ALL TP, FP, FN globally across classes.
  Compute metric on the pooled counts.
  Weighted by class size — large classes dominate.
```

## 5.3 Data Splits

| Split strategy | How it works | When to use |
|---|---|---|
| Train / Dev / Test | Train fits params. Dev tunes features. Test used ONCE only. | Always the baseline approach |
| N-fold Cross-validation | Divide into N folds; train on N-1, test on 1; rotate. Report mean and variance. | Limited data; maximise training |

| Split strategy | How it works | When to use |
|---|---|---|
| Stratified CV | Each fold preserves class distribution. | Imbalanced datasets |
| Leave-one-out | N = dataset size. Train on all-but-one. | Very small datasets; computationally expensive |
| Dependency-sensitive CV | Fold by domain (author, genre, source). | Test generalisation to truly new domains |

**Multiple Comparisons Problem**

If you evaluate on the test set repeatedly during development, you are implicitly tuning to it. Statistical significance tests become invalid — the test set is no longer 'unseen'.
The test set should be touched ONCE: to report final results.

## 5.4 Sign Test — Full Worked Method

Goal: test whether System A is significantly better than System B.

```
Setup:
  H_0 (Null Hypothesis): A and B are identical; differences are due to
chance.
  Compare systems item by item (each document is one 'event').

Counting:
  Let k = number of items where A beats B
  Let d = number of TIES (A and B give same result)
  MLRD tie rule: add d/2 to k and d/2 to (N-k). Round up k at end.
  N = total items (do NOT discard ties).

Example: 200 documents. A wins 110, B wins 70, ties = 20.
  k = 110 + (20/2) = 120  (rounded up from 120.0)
  N = 200 (unchanged)

Test statistic: X ~ Binomial(N=200, p=0.5) under H_0
p-value (one-tailed) = P(X >= 120 | N=200, p=0.5)
p-value (two-tailed) = 2 * P(X >= 120 | N=200, p=0.5)

If p-value <= 0.05: reject H_0. System A is significantly better.
If p-value >  0.05: inconclusive. Cannot claim A is better.
```

**MLRD tie rule**

Standard textbooks say to DISCARD ties. For classification tasks, this is wrong.
Discarding ties reduces N, which inflates the test statistic and causes a TYPE 1 ERROR.
ALWAYS distribute ties as 0.5 to each side. This is the MLRD-specific rule.

### Error types and test power

| Error type | What happens | Consequence |
|---|---|---|
| Type 1 Error (alpha) | Reject H_0 when it is true | Falsely claim one system is better |
| Type 2 Error (beta) | Fail to reject H_0 when it is false | Miss a real improvement |
| Power (1 - beta) | Correctly reject a false H_0 | Higher power = more sensitive test |

| Error type | What happens | Consequence |
|---|---|---|
| Specificity (1 - alpha) | Correctly NOT reject a true H_0 | Protects against false discoveries |

## 5.5 Permutation / Randomisation Test

A more powerful alternative to the sign test. Does not need to handle ties specially.

```
Concept: under H_0, A and B are interchangeable.
Randomly swapping their predictions for any item should not
systematically
change the overall metric difference.

Procedure:
  1. Compute observed metric difference: delta_obs = metric(A) -
metric(B).
  2. Enumerate all 2^N ways to swap predictions between A and B.
  3. For each permutation, compute metric difference delta_perm.
  4. p-value = proportion of permutations where delta_perm >= delta_obs.

If p-value <= 0.05: reject H_0.
More powerful than sign test; naturally handles ties.
```

# 6 Inter-Annotator Agreement

**Why we need it**

Before trusting labelled training data, we must verify that humans agree on the labels.
Raw agreement is inflated by chance — especially with few classes.
Chance-corrected metrics (Kappa) tell us how much better than chance the agreement is.

## 6.1 Cohen's Kappa — Two Annotators

```
kappa = [ P(A) - P(E) ] / [ 1 - P(E) ]

P(A) = observed agreement = proportion of items where both annotators
agree

P(E) = expected chance agreement
     = SUM over classes c of:  P(Ann_1 chooses c) * P(Ann_2 chooses c)

Interpretation:
  kappa = 1.0 : perfect agreement
  kappa = 0.0 : no better than chance
  kappa < 0   : worse than chance (annotators actively disagree)
  kappa > 0.8 : generally considered strong agreement
  kappa > 0.6 : generally considered acceptable
```

**Worked example (2023 Q7, Annotators A and B only)**

From the 2023 data table (8 items, 4 classes: I, II, III, IV):

```
A's assignments: III, IV, II, I, II, I, IV, II
B's assignments: III,  I, II, IV, IV, I, IV, I

Items where A=B: item1 (III=III), item3 (II=II), item6 (I=I), item7
(IV=IV) -> 4 agreements
P(A) = 4/8 = 0.5

Marginal frequencies for Annotator A:
  P_A(I) = 2/8 = 0.25    P_A(II) = 3/8 = 0.375
  P_A(III) = 1/8 = 0.125  P_A(IV) = 2/8 = 0.25

Marginal frequencies for Annotator B:
  P_B(I) = 3/8 = 0.375   P_B(II) = 1/8 = 0.125
  P_B(III) = 1/8 = 0.125  P_B(IV) = 3/8 = 0.375

P(E) = (0.25)(0.375) + (0.375)(0.125) + (0.125)(0.125) + (0.25)(0.375)
     = 0.09375 + 0.046875 + 0.015625 + 0.09375
     = 0.25

kappa = (0.5 - 0.25) / (1 - 0.25) = 0.25 / 0.75 = 0.333

Interpretation: moderate agreement, well above chance but not strong.
```

## 6.2 Fleiss' Kappa — More Than Two Annotators

Used when multiple annotators each label all items (or a crowdsourcing setting).

```
P_a (observed) = (1/N) * SUM_{i=1..N} [ SUM_{j=1..C} n_ij*(n_ij-1) /
(k*(k-1)) ]

where:
  N    = number of items
  k    = number of annotators per item
  C    = number of categories
  n_ij = number of annotators who assigned category j to item i

Note: n_ij*(n_ij-1)/2 is the number of AGREEING PAIRS for class j on item
i.
      k*(k-1)/2 is the total number of pairs. The formula uses k*(k-1)
not /2
      because both numerator and denominator cancel the factor of 2.

P_e (chance) = SUM_{j=1..C} p_j^2
  where p_j = (total annotations in class j) / (N * k)
  i.e. the overall proportion of all annotations assigned to class j

kappa = (P_a - P_e) / (1 - P_e)
```

**Worked mini-example (3 annotators, 3 items, 2 classes: POS/NEG)**

```
Item 1: A=POS, B=POS, C=NEG  -> n_POS=2, n_NEG=1
Item 2: A=POS, B=NEG, C=NEG  -> n_POS=1, n_NEG=2
Item 3: A=POS, B=POS, C=POS  -> n_POS=3, n_NEG=0

P_a per item:
  Item 1: [2*(2-1) + 1*(1-1)] / [3*(3-1)] = [2+0]/6 = 0.333
  Item 2: [1*(1-1) + 2*(2-1)] / 6 = [0+2]/6 = 0.333
  Item 3: [3*(3-1) + 0*(0-1)] / 6 = [6+0]/6 = 1.0
P_a = (0.333 + 0.333 + 1.0) / 3 = 0.556

Overall proportions: total annotations = 3*3 = 9
  POS: (2+1+3)=6 annotations -> p_POS = 6/9 = 0.667
  NEG: (1+2+0)=3 annotations -> p_NEG = 3/9 = 0.333
P_e = 0.667^2 + 0.333^2 = 0.444 + 0.111 = 0.556

kappa = (0.556 - 0.556) / (1 - 0.556) = 0 / 0.444 = 0.0
Interpretation: agreement is no better than chance.
```

## 6.3 Partial Annotation — Handling Annotators D and E (2023 Q7c)

**Problems with merging D and E as one fake annotator**

1. Decisions from D (items 3-8) and E (items 1-8) were made at different times under different conditions.
2. Randomly discarding one annotation throws away real signal and introduces arbitrary bias.
3. The 'merged' annotator has an inconsistent coverage pattern, making comparison unfair.
4. Inflates N if D covered more items than E, creating asymmetric weighting.

Adaptation for partial annotation:

1. Calculate P_a only over OVERLAPPING items: items annotated by at least 2 annotators from the full set {A,B,C,D,E}.
2. Calculate P_e from ALL available annotations (marginal proportions across the full dataset, regardless of overlap).
3. The denominator k*(k-1) for each item uses the number of annotators who actually labelled THAT item (not a global k).

## 6.4 Crowdsourcing Problems

- Very small batches (e.g., 2 items each) create high variance in agreement estimates — unreliable
- Random assignment means many annotator pairs never share an item — pairwise kappa impossible
- Quality varies widely across crowd workers — some may be spamming randomly
- No inter-section consistency check if annotators do not share items across sections

# 7 Unsupervised Clustering — K-Means

**Use case**

Data has features but NO labels. We want to discover natural groupings.
Unlike Naive Bayes or HMMs, no labelled training data is needed.

## 7.1 K-Means Algorithm

1. Initialise: choose K centroids. Standard: random selection from data points.
2. Assign: assign each data point to the cluster of its nearest centroid (Euclidean distance).
3. Update: recompute each centroid as the mean of all points currently assigned to it.
4. Repeat steps 2-3 until assignments stop changing (convergence).

**K-Means++ initialisation**

Problem: random initialisation can lead to poor local optima.
Solution: choose first centroid randomly, then each subsequent centroid with probability proportional to d(x)^2 — points far from existing centroids are more likely to be chosen. Results in better convergence.

## 7.2 Hard vs. Soft Clustering

| | Hard (standard K-Means) | Soft Clustering |
|---|---|---|
| Assignment | Each point belongs to exactly ONE cluster | Each point has a probability distribution over clusters |
| Use case | Clear, non-overlapping natural groupings | Overlapping groups (e.g., people in multiple social circles, topics) |
| Output | Cluster label per point | Membership probabilities per cluster per point |

## 7.3 Evaluation Metrics

### Intrinsic — no ground truth labels needed

```
WCSS (Within-Cluster Sum of Squares):
  WCSS = SUM_{n=1..N} distance(x_n, mu_{z_n})^2
  mu_{z_n} = centroid of the cluster x_n is assigned to
  Lower = tighter, more compact clusters.
  Use elbow plot: plot WCSS vs K; choose K at the 'elbow'.

Silhouette Score:
  a = mean distance to other points in the SAME cluster (cohesion)
  b = mean distance to points in the NEAREST other cluster (separation)
  S = (b - a) / max(a, b)
  Range: -1 (incorrectly clustered) to +1 (perfectly clustered).
  S near 0: point is on a cluster boundary.
```

### Extrinsic — requires ground truth labels

| Metric | How it works | Range |
|---|---|---|
| Purity | Assign majority class label to each cluster; compute accuracy | 0 to 1 (higher=better) |
| Adjusted Rand Index (ARI) | Pairwise cluster matching corrected for chance agreement (like Kappa) | −1 to 1 (1=perfect) |
| Normalised Mutual Information (NMI) | Information-theoretic: how much does cluster assignment tell us about class? | 0 to 1 (1=perfect) |

## 7.4 Application: School Class Grouping (2024 Q7e)

Task: group students of similar ability into roughly equal-sized classes WITHOUT using level labels.

1. Use available quantitative features: total score, individual question responses.
2. Represent each student as a feature vector (e.g., binary vector of questions answered correctly).
3. Run K-Means with K = desired number of classes.
4. Add a size constraint or post-processing step to balance cluster sizes.
5. Evaluate using silhouette score (intrinsic) or, if labels later become available, purity/ARI.

# 8 Ethics, Bias & Real-World Failures

**Core principle**

Algorithms automate, obfuscate, and scale human biases encoded in training data.
They do not possess objective fairness.
'We are just reflecting what is in the data' is NOT an acceptable response.

## 8.1 Case Studies — Required for Exam

### London Medical School Admissions (1970s)

A program was built to replicate human admissions decisions as closely as possible. It succeeded — including replicating gender and ethnic discrimination. The historical training data encoded prejudice; the algorithm learned and automated it. Eventually led to a case by the Commission for Racial Equality.

**Exam lesson**

Fitting human decisions does not produce fair decisions if humans themselves were unfair.
Historical data encodes historical injustice.

### Pneumonia-Asthma Case (Caruana et al.)

A neural network trained to predict pneumonia complications learned that having asthma LOWERED mortality risk. The real reason: asthma patients were fast-tracked to ICU, so survived at higher rates in the data. The true causal relationship is the opposite.
A transparent rule-based system immediately revealed this deadly logical flaw. The neural network — being uninterpretable — concealed it. Despite high accuracy, the neural net was dangerously unsuitable for clinical triage.

**Exam lesson**

Interpretability is a safety requirement in high-stakes domains, not a convenience.
High test-set accuracy does not imply the model has learned the right thing.

## 8.2 Proving Bias Mathematically

To demonstrate that a classifier is biased against a group:

1. Identify a protected attribute (e.g., Gender, Ethnicity).
2. Find a value of that attribute that only appears in one class in training data.
3. Construct two identical test instances, differing ONLY in the protected attribute.
4. Show the classifier gives different predictions for the two instances.
5. OR: show that P(class X | protected_value) = 0, so the class can never be predicted for that group.

## 8.3 AI Hype and Reporting Problems

| Problem | Description | MLRD term |
|---|---|---|
| Cherry picking | Only reporting evaluations on easy data slices where the system performs well | — |
| Anthropomorphisation | Ascribing human consciousness, intent, or feelings to statistical models | — |

| Problem | Description | MLRD term |
|---|---|---|
| No significance test | Claiming 'better' based on raw number differences alone | Methodological unsoundness |
| Significance misuse | Using 'significant' to mean 'large effect size' rather than statistically significant | Scientific illiteracy |
| Reporting without test | Making 'better/outperforms' claims with no test at all | Scientific fraud |

## 8.4 Asimov's Laws of Robotics

```
0th Law (added later):
  A robot may not harm humanity, or through inaction allow humanity to
come to harm.

1st Law:
  A robot may not injure a human being, or through inaction allow a human
to come to harm.

2nd Law:
  A robot must obey orders given by humans, unless these conflict with
the 1st Law.

3rd Law:
  A robot must protect its own existence, unless this conflicts with Laws
1 or 2.
```

These laws illustrate the DIFFICULTY of formally encoding ethics. Each law has known loopholes — the Zeroth Law paradox enables utilitarian harm to individual humans 'for the greater good'. They are a philosophical framework, not a technical solution.

## 8.5 AGI and Existential Risk

The concern that superintelligent algorithms could pose an existential threat to humanity. Studied by the Cambridge Centre for the Study of Existential Risk (CSER) and the Leverhulme Centre for the Future of Intelligence. The MLRD lectures present this as a serious academic research topic. Current risks exist even without AGI: autonomous stock trading, load balancing, autonomous vehicles — real-world decisions already made without human intervention.

# 9 Past Paper Pattern Library — 2022 to 2025

| Year | Q | Topic | Pattern and key marks |
|---|---|---|---|
| 2022 | 8 | HMM | (a) Define+estimate HMM without smoothing [4]. (b) Assumptions + appropriateness [4]. (c) Viterbi for 3 timesteps, show all equations+calcs [8]. (d) Two shortcomings [4]. |
| 2022 | 9 | NB | (a) NB equations [2]. (b) Data split justification [2]. (c)(i) Features for attacked/safe, generalisation [6]. (c)(ii) Two feature improvements [4]. (d) Real-time modifications [3]. (e) Post-level advantages/disadvantages [3]. |
| 2023 | 7 | IAA | (a) Pairwise agreement, calculate [4]. (b)(i) Why chance correction matters [2]. (b)(ii) Kappa chance formula + calculate [2]. (b)(iii) Kappa formula + calculate [2]. (c)(i) Problems with merging D+E [4]. (c)(ii) Adapted Kappa formula [4]. (c)(iii) Problem in crowdsourcing [2]. |
| 2023 | 8 | NB | (a) NB setup + formulae [3]. (b) Two evaluation methods [2]. (c)(i-iv) Handle area removed/new/split/merged [2 each]. (d) Rank situations [2]. (e) Area similarity + citation network + visualisation [5]. |
| 2023 | 9 | HMM | (a) Define+estimate HMM [4]. (b) Viterbi to find inflation sequence for obs high-med-low [6]. (c)(i) Is low->high possible? (c)(ii) Low stays low vs rises? [4]. (d) Three shortcomings [6]. |
| 2024 | 7 | NB | (a) NB setup + smoothing [4]. (b) Compute all feature probabilities [6]. (c) Classify new student [2]. (d) Feature relevance methods [4]. (e) Clustering for class grouping [4]. |
| 2024 | 8 | HMM | (a) Define+estimate HMM [4]. (b)(i) Most likely next month [4]. (b)(ii) Most likely in 3 months [included]. (c) Viterbi for July-Oct sequence [6]. (d) Time-aware compound-state model [6]. |
| 2024 | 9 | NB+Ethics | (a) NB parameter estimation [4]. (b) Identify 2 biased groups + demonstrate [4]. (c) Naive property: useful + problematic [2]. (d) Incorporate pre-2020 data [4]. (e) Trajectory model (HMM solution) [6]. |
| 2025 | 7 | NB+Lexicon | (a)(i) Steps to build lexicon [2]. (a)(ii) How detector works [3]. (a)(iii) Evaluation metrics + data splits [4]. (b) Human rating reliability (Kappa) [3]. (c)(i) NB setup + smoothing [4]. (c)(ii) NB vs lexicon + combining [4]. |
| 2025 | 8 | Networks | (a) Degree/CC/BC for A,C,H [5]. (b) Measure clustering in a network [2]. (c) Connector measure [2]. (d) Diameter [2]. (e) Edge deletion+addition effect on BC [4]. (f) Strongly connected + can we make it so? [3]. (g) Community detection algorithm [2]. |
| 2025 | 9 | HMM | (a) Transition matrix A [4]. (b) Emission probabilities for a and l [2]. (c) Most likely next letter from state a [2]. (d) Most likely letter 2 steps ahead from e [4]. (e) Probability of sequence b-e-a-t [2]. (f) Smoothing + new probability [6]. |

## 9.1 Model Answer Templates for High-Mark Questions

### Template: 'Define and estimate HMM components without smoothing' [4 marks]

```
1. State the five components: S (hidden states), V (observations),
   A (transition matrix), B (emission matrix), pi (initial distribution).

2. Write the general formula for each:
```

```
   a_ij = P(q_t = s_j | q_{t-1} = s_i)
   b_i(v_k) = P(o_t = v_k | q_t = s_i)

3. Extract state sequence and observation sequence from training data.

4. Count each transition and emission. Divide by source-state counts.
   a_ij = count(s_i -> s_j) / count(s_i)
   b_i(v_k) = count(s_i emits v_k) / count(s_i)

5. Write out the full A matrix and B matrix with all numerical values.

6. Note any zero probabilities and state this is a problem without
smoothing.
```

### Template: 'HMM assumptions — are they appropriate?' [4 marks]

```
State all THREE assumptions explicitly:

1. First-order Markov: only the previous state matters.
   In this context: [evaluate whether previous state is sufficient,
   e.g., infection levels can persist or reverse over longer windows ->
may fail]

2. Output independence: observation depends only on current state.
   In this context: [evaluate, e.g., consecutive positivity rates are
correlated -> may fail]

3. Stationarity: probabilities do not change over time.
   In this context: [evaluate, e.g., COVID behaviour changes with
variants/seasons -> likely fails]

Always tie EACH assumption back to the SPECIFIC application in the
question.
```

### Template: 'Demonstrate bias with two test instances' [4 marks]

```
Step 1: Inspect the training data. Find a feature value that appears
        ONLY in one class label.

Step 2: Identify the biased group:
   'Feature X = value V never appears with class Y in training.'
   'Therefore P(X=V | Y) = 0.'

Step 3: Construct the test instance:
   'Instance: Feature1=neutral, Feature2=neutral, X=V'
   P(Y | instance) = P(Y) * P(F1=neutral|Y) * P(F2=neutral|Y) * P(X=V|Y)
                   = P(Y) * ... * 0 = 0
   'The classifier can never predict class Y for any instance with X=V.'

Step 4: Repeat for a second protected group / feature value.
```

### Template: 'Rank four situations by damage to classifier' [2 marks] (2023 Q8d)

```
Four situations (from most to least damaging):

1. MOST DAMAGING: New area added.
   Classifier has NO knowledge of this class. Papers in new area are
routed
   to the most 'similar' existing area — systematically wrong. Cannot
recover
   without retraining on new labelled data.

2. SECOND: Existing area split into two.
   Papers are all routed to the original area label — half will be
misrouted.
   Can be partially recovered by splitting the old area's training data.

3. THIRD: Existing area removed.
   Old papers assigned to this area are now mislabelled. Classifier may
still
   predict this area for new papers. Relatively easy to fix: remove the
class.

4. LEAST DAMAGING: Two areas merged.
   Both old classes are now one. Classifier may predict either old class,
   but both are 'correct' for the merged area. Minor post-processing fix
needed.
```

## 9.2 Formula Quick-Reference Card

| Formula | Expression |
|---|---|
| NB classification | c* = argmax_c  P(c) * PROD_{i} P(wi\|c) |
| NB smoothed likelihood | P(w\|c) = [count(w,c)+1] / [SUM count(w',c) + \|V\|] |
| HMM transition (smoothed) | a_ij = [count(si->sj)+1] / [count(si) + \|S\|] |
| HMM emission (smoothed) | b_i(vk) = [count(si emits vk)+1] / [count(si) + \|V\|] |
| Viterbi initialisation | delta_j(1) = pi_j * b_j(o_1) |
| Viterbi recursion | delta_j(t) = max_i [ delta_i(t-1) * a_ij * b_j(o_t) ] |
| Zipf's Law | f_w ~ k / (r_w + beta)^alpha  \|  log-log plot gives straight line |
| Heaps' Law | \|V\| = k * N^beta  \|  log\|V\| = log(k) + beta*log(N) |
| Clustering coefficient | C_i = 2*e_i / [k_i*(k_i-1)] |
| Betweenness centrality | CB(v) = SUM_{s!=v!=t} sigma_st(v)/sigma_st  [halve for undirected] |
| Cohen's Kappa | kappa = [P(A)-P(E)] / [1-P(E)]  \|  P(E) = SUM_c P(c1)*P(c2) |
| Fleiss' Kappa P_a | (1/N)*SUM_i [SUM_j n_ij*(n_ij-1)] / [k*(k-1)] |
| Fleiss' Kappa P_e | SUM_j p_j^2  where p_j = (all annotations in class j)/(N*k) |
| Precision | TP / (TP + FP) |
| Recall | TP / (TP + FN) |
| F1 | 2*P*R / (P+R) |
| Silhouette Score | S = (b-a) / max(a,b)  \|  a=intra-cluster mean dist, b=nearest-cluster mean dist |

| Formula | Expression |
| --- | --- |
| WCSS | SUM_n  distance(x_n, mu_{z_n})^2 |

# 10 Exam Strategy — Choosing Your 2 Questions

**The setup**

You answer 2 out of 3 questions in Paper 3.
Questions 7, 8, and 9 each cover different topics.
Recent pattern: Q7=NB/Classification, Q8=HMM or Networks, Q9=HMM or Ethics/NB.
Knowing your strongest two topics is as important as knowing the content.

| Topic | Appears as | Reliability | Your prep status |
|---|---|---|---|
| Naive Bayes | Q7 or Q9 almost every year | Very predictable — same structure | Review Sections 1-2 |
| HMMs + Viterbi | Q8 or Q9 almost every year | Very predictable — same steps | Review Section 3 |
| Networks | Q8 (2025 was pure networks) | Less frequent but deep | Review Section 4 |
| IAA / Kappa | Often sub-question within Q7/Q9 | Reliable sub-topic | Review Section 6 |
| Ethics | Often sub-question or part of Q | Reliable but short answers | Review Section 8 |
| Zipf/Heaps | Sub-question, contextual | Can be asked without warning | Review Section 2 |
| K-Means Clustering | Sub-question (e.g. 2024 Q7e) | Appears as part of NB question | Review Section 7 |

**Recommended strategy**

Master NB + HMMs first — these appear every year in predictable formats.
They together give you enough to comfortably answer Q7 and one of Q8/Q9.
Networks (Q8 2025) is your 'backup' topic if HMMs appear in Q9 instead.
For each question: always show ALL working. Examiners award method marks even if your final numbers are wrong. A wrong answer with correct methodology can still earn 6/8 marks.