FIT5202 Data Processing for Big Data

FACULTY OF INFORMATION TECHNOLOGY · MONASH UNIVERSITY · SEMESTER 2, 2026

Full guide + ask Sia about FIT5202

SIDE 1/2 WHAT MUST BE IN YOUR HEAD · Volume → Complexity → Velocity **CLOSED-BOOK MEMORY SHEET** 45% final · Q1 MCQ + Q2–Q6 written

0 · Exam Map

COURSE.JSON · BRIEF

A1 15 | A2A 15 | A2B 15 | Q | L5 | FINAL 45

Assignment 1 — Analysing Road Safety Data	15%
Assignment 2 Part A — Building ML Models	15%
Assignment 2 Part B — Streaming Application	15%
Weekly Quizzes (Weeks 2–11)	5%
Lab Attendance (Sessions 2–11)	5%
Final Examination	**45%**

"Type: Closed Book" — FIT5202 Week 12 Recap 2026 slides, exam format (PDF_FIT5202_12_Recap_2026).

★ Final: **e-assessment · 2h10m · scientific calculator · Weeks 1–12 · Q1 MCQ + Q2–Q6 written.** Recite the rule, write the formula, show the path, interpret the output.

1 · Performance + Partitioning

CH 1–2

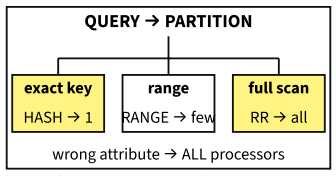
SAME TASK, MORE PROCESSORS

Speed-up $S_p = T_1/T_p$ · ideal linear $S_p = p$

Amdahl: $S_p = 1 / [s + (1-s)/p]$

BIGGER TASK + PROPORTIONALLY MORE RESOURCES

Scale-up = Tsmall / Tlarge · ideal = 1



Source redraw: partitioning × query determines processor activation.

METHOD	WINS	LOSES
Round-robin	even load	no locality; all nodes
Hash	exact match → 1	range → all; skew
Range	ordered/range → selected	popular range skews

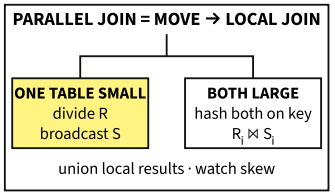
Elapsed time is set by the slowest processor. Start-up, consolidation, interference, communication and skew make speed-up sub-linear. Zipf θ: 0 = no skew; 1 = highly skewed. Ordered partition → binary search; unordered → linear search.

WHY PARALLEL AT ALL?

The five V's are **Volume** (size), **Variety** (structured, semi-structured, unstructured), **Velocity** (batch to continuous streams), **Veracity** (trust) and **Value** (useful insight). FIT5202's three themes are Volume (Weeks 1–4), Complexity/ML (Weeks 5–8) and Velocity/streaming (Weeks 9–11).

2 · Parallel Join

CH 3 · WRITTEN



Source redraw: broadcast the small side; co-partition two large sides.

COST MODEL — WHERE IT BITES: CHOOSE PLAN

Ctotal = CCPU + CI/O + Cnetwork

network/shuffle usually dominates

LOCAL JOIN	RULE	USE
Nested loop	compare pairs	costly fallback
Sort-merge	sort both; merge	already sorted / large
Hash	build smaller; probe larger	equijoin workhorse

Divide-and-broadcast: split R into R_i; replicate all S to each processor; $R_i \bowtie S$ is complete because every R row meets every possible S match. Per processor, an evenly split S sends and receives $|S_i|(N-1)$ records. If both sides are large, hash-partition both on the same join key instead.

SAY IT *"I choose __ because it minimises network traffic; then each processor performs __ locally." Named outer-join methods: ROJA, DOJA, DER; OJSO targets skew.*

3 · External Sort

CH 4

PASS COUNTER — CALCULATION

$N_0 = \lceil F/B \rceil$ sorted runs

$N_{j+1} = \lceil N_j/(B-1) \rceil$ until 1

Total passes = 1 sort pass + merge passes

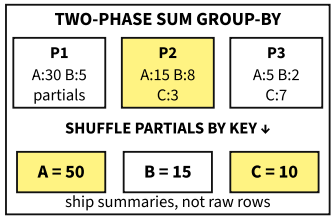
PARALLEL SORT	BOTTLENECK / MOVE
Merge-all	one final K-way merge bottleneck
Binary-merge	pairwise pipelined merge
Redistribution variants	spread merge load; shuffle
Partitioned	range-redistribute → local sort; no merge; skew risk

Reserve one output buffer: fan-in is B−1, never B. Parallel partitioned sort: round-robin load for balance → range redistribute → independent local sort.

Five forms: merge-all; binary-merge; redistribution binary-merge; redistribution merge-all; partitioned sort. Redistribution spreads merge work but shuffles. Partitioned sort needs no final merge because ranges establish global order; bad boundaries create skew.

4 · Group-By

CH 4 · METHOD CHOICE



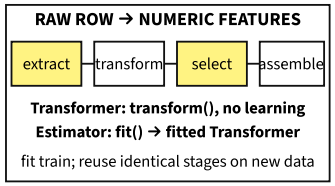
Source redraw: local combine → key shuffle → final combine.

METHOD	FLOW	DECISION
Traditional	local → coordinator	single-node finish
Two-phase	local partial → shuffle partials → final	best when local reduction is large
Redistribution	shuffle raw → local aggregate	few groups / little reduction

AVG trap: never average partial averages. Carry (sum, count); final AVG = Σsum/Σcount. The same network-cost logic used for joins decides which group-by wins.

5 · Featurisation Pipeline

CH 5



Source redraw: ordered Spark ML Pipeline stages.

JOB	TOOLS / OUTPUT
Extract text	CountVectorizer counts; TF-IDF weights; Word2Vec vectors; FeatureHasher
Transform	Tokenizer, stop-word removal, StringIndexer, OneHotEncoder
Assemble	VectorAssembler → one features vector

TF-IDF — FEATURE WEIGHTING

IDF(t,D)=ln[(|D|+1)/(DF(t,D)+1)]

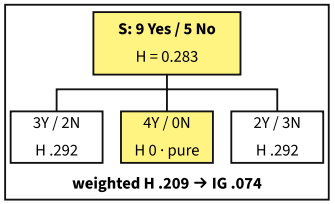
TFIDF = TF × IDF

StringIndexer orders labels by frequency; One-Hot removes the false numerical order. Spark uses an N−1-dimensional vector for N categories. Train/test roles stay distinct: fit on training data; evaluate on held-out data.

Task gate: supervised data have labels—classification predicts a discrete label; regression a continuous value. Unsupervised data have none—clustering finds groups; association finds co-occurrence. Featurisation is Extraction + Transformation + Selection; the pipeline preserves the same order for training and later records.

6 · Decision Trees

CH 6 · CORE CALC



Source redraw: the taught base-10 entropy split.

ID3 — SPLIT SCORING

$H(S) = \sum p_i \log_{10}(1/p_i)$

$H(S,A) = \sum(|S_j|/|S|)H(S_j)$

$IG(S,A) = H(S) - H(S,A)$ → choose largest

1. Compute parent entropy.
2. For every attribute: child entropies → weighted entropy → information gain.
3. Split on maximum gain; recurse until pure or stopped.

Depth judgement: too deep memorises noise (overfit); too shallow underfits. Limit maxDepth or prune. Random forest = bootstrap trees + feature subsets + majority vote; reduces one tree's variance. Data parallelism partitions records; result parallelism grows tree nodes across processors.

7 · Classifier Evaluation

CH 6

	PRED +	PRED −
ACTUAL +	TP	FN
ACTUAL −	FP	TN

Accuracy=(TP+TN)/all

Precision=TP/(TP+FP) — *false alarms*

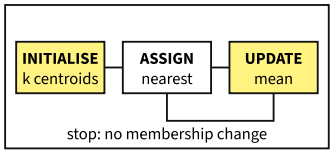
Recall=TP/(TP+FN) — *misses*

F1=2PR/(P+R)

Fix the positive class first. On imbalanced data, accuracy can look excellent while the rare positive class is missed. Report precision, recall and F1, then state the FP/FN trade-off.

8 · k-Means

CH 7 · TRACE



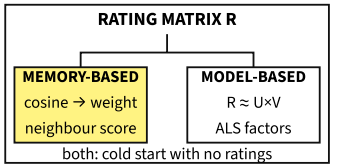
Source redraw: assign → update → convergence.

$d(a,b) = \sqrt{\sum(a_i - b_i)^2}$ · 1-D: |a-b|

centroid = Σ cluster points / count

9 · Collaborative Filtering

CH 8



Source redraw: direct similarity versus learned latent factors.

USER SIMILARITY — CO-RATED ITEMS ONLY

$sim(u,u') = \sum r_{ui} r_{u'i} / [\sqrt{\sum r_{ui}^2}\ \sqrt{\sum r_{u'i}^2}]$

PREDICT UNSEEN RATING

$\hat{r}_{ui} = k \sum sim(u,u') r_{u'i}$

$k = 1/\sum|sim(u,u')|$

1. Restrict each user pair to co-rated items.
2. Compute similarities.
3. Use only neighbours who rated the target item; weight and normalise.
4. Rank predicted unseen ratings for top-N.

CHOICE	MEANING / SCALING
User-based	similar users → borrow item ratings
Item-based	similar item columns → recommend
Matrix factorisation	R(N×M) ≈ U(N×r)V(r×M)
ALS	fix U solve V; fix V solve U; minimise squared error/RMSE

Sanity check: prediction must stay in the rating scale. Forgetting k can push it outside. New user/item → too few ratings: cold start.

10 · Method Chooser

COMPLEXITY

labelled + discrete target	**classification**
labelled + continuous target	**regression**
unlabelled groups	**clustering**
sparse user–item preferences	**CF / ALS**

WRITTEN MOVE *Name the task, define inputs/output, state the selection metric, show the update, give the stop/evaluation rule.*

ALS + COLD-START RECALL

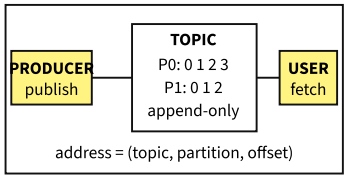
MODEL OBJECTIVE

$\min \sum(r_{ui} - u_u \cdot v_i)^2$ over observed ratings

R is N users × M items; U is N×r and V is r×M, with r latent features. ALS makes the joint problem manageable by fixing U while solving V, then fixing V while solving U, alternating to reduce error. A predicted cell is the user-profile/item-profile dot product. Memory-based CF re-scans neighbours; factorisation learns compact profiles for scale.

11 · Kafka + Windows

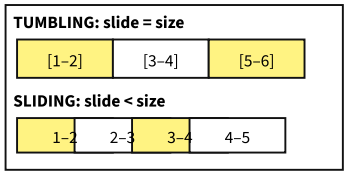
CH 9



Source redraw: Kafka offsets are per-partition sequence IDs.

TERM	RECALL LINE
Topic	named feed
Partition	parallel append-only log
Offset	unique increasing ID within partition
Broker	server storing/replicating partitions
Consumer	tracks its own position; reads do not delete

Event time = produced at source; **processing time** = reaches processor. Delay/bursts separate them. Spark Structured Streaming follows producer → processor → consumer.



Source redraw: overlap controls reuse and output rate.

12 · Stream Join + Reduction

CH 10–11

SHJ ARRIVAL RULE

expire old → probe OTHER table → emit → insert OWN

Two tables preserve partners arriving in either order. A window or watermark bounds otherwise unbounded state; older event-time state can be discarded. M-Join: expire, probe *all* other tables, then hash into own.

SLIDE S VS WINDOW W	CLASS
s = 1 record	overlap, no reduction; moving average
1 < s < w	overlap, with reduction
s = w	tumbling, with reduction

Sensor arrays: same thing → Reduce over time then Merge sensors, or reverse. Different things require Normalize around the merge. Sampling and load shedding trade detail/accuracy for throughput.

State logic: stream–static buffers the streaming side; stream–stream retains both because either partner may arrive later. Window expiry bounds the two tables; a watermark advances an event-time threshold while tolerating some lateness.

Forward this to your tutorial group

SHARED BY _____

Your exam is closed book: nothing printed goes in. Use this sheet to revise and self-test, not to carry.

Monash Semester 2 final assessment period, 2–18 November 2026 (check your personal exam timetable).

FIT5202 Self-Test Drills · Answer Paths + Traps

DATA PROCESSING FOR BIG DATA · MONASH UNIVERSITY · SEMESTER 2, 2026

Full guide + ask Sia about FIT5202

SIDE 2/2 FIVE WORKED PATHS · judgement cards · classic traps · exam-morning plan

COVER THE ANSWERS FIRST

Say → write → calculate → check → interpret

13 · Rapid Volume Checks

CH 1–4 DRILLS

60 min · 10% serial · p=4 · 20% overhead

serial	parallel	overhead	total
6	54/4	+2.7	22.2

speed-up = 60/22.2 ≈ 2.7

sub-linear: 2.7 < 4

Source redraw: separate serial work before dividing.

IF YOU SEE…	DO / SAY
same workload; more nodes	T_1/T_p → speed-up
task + resources grow together	Tsmall/Tlarge → scale-up
exact key on hash key	activate 1 processor
range on hash partition	activate all
query on non-partition attribute	activate all
one slow partition	skew controls elapsed time

Check: 4 processors, hash on `customer_id`. Equality =5051 → 1; range 2000–3000 → all 4; round-robin equality → all 4. Justify with grouping, not guesswork.

14 · Sort + Join Checks

CH 3–4

EXTERNAL SORT: F=150 PAGES, B=8

```
Pass 0: ⌈150/8⌉ = 19 runs
fan-in = B-1 = 7
Pass 1: ⌈19/7⌉ = 3
Pass 2: ⌈3/7⌉ = 1 → 3 passes
```

BROADCAST: N=4, $|S_i|$=1000, |S|=4000

```
send each = 1000(4-1)=3000
receive each = 4000-1000=3000
all transfers = 4×3000=12 000
```

Completeness comes from each R_i meeting all S. Cost grows with N−1, so choose this only when S is small. Two large sides → hash both on the join key.

TRAP SCAN *B-way merge ✗; broadcasting the large side ✗; counting only sends, not receives ✗; claiming CPU dominates a shuffle-heavy join ✗.*

14b · Group-By + Sort Recall

CH 4 TOPIC MAP

PROMPT	FULL-CREDIT SKELETON
two-phase SUM	local partial sums → shuffle (key,sum) → final sum
two-phase AVG	local (sum,count) → shuffle → Σsum/Σcount
partitioned sort	round-robin load → range redistribute → local sort
merge-all weakness	one coordinator performs final K-way merge
redistribution benefit	spreads merge load; pays network cost

15 · Worked 1 – Entropy + IG

CH 6 WORKED

Given: S has 9 Yes, 5 No. Outlook branches: Sunny 3Y/2N, Overcast 4Y/0N, Rainy 2Y/3N. Use $\log_{10}$.

PARENT → CHILDREN → GAIN

$H(S)=(9/14)\log_{10}(14/9)$
$+(5/14)\log_{10}(14/5)=0.283$

$H_{children}=(5/14).292+(4/14)0$
$+(5/14).292=0.209$

IG=0.283−0.209=0.074

Recomputed source values; pure Overcast contributes zero entropy.

1. Parent: (9/14)(0.1924)+(5/14)(0.4472)=0.1237+0.1597=**0.283**.
2. Sunny H=(3/5)$\log_{10}$(5/3)+(2/5)$\log_{10}$(5/2)=**0.292**. Rainy mirrors it; Overcast is pure → 0.
3. Weighted entropy = (10/14)(0.292)=**0.209**.
4. Gain = **0.074**. Compute for every attribute; choose the largest.

TRAP *Base 2 rescales the numbers. In this course's taught example, state and use base 10.*

16 · TF-IDF + Pipeline Recall

CH 5 WORKED

Three documents. "the" appears in all 3: IDF = ln(4/4)=**0**. "spark" appears in 1: IDF = ln(4/2)=**0.693**. In d1, TF=2 → TF-IDF = 2(0.693)=**1.386**. Rare across corpus + frequent in document = useful.

SEE X	DO Y
common term	IDF tends to 0
nominal category	StringIndexer → One-Hot
mixed feature columns	VectorAssembler
learned vocabulary/model	Estimator.fit()
fixed row mapping	Transformer.transform()

16b · Featurisation Decisions

CH 5 TOPIC MAP

RAW FIELD / NEED	TOOL → RESULT
token counts	CountVectorizer → vocabulary-aligned sparse vector
discriminating text	TF-IDF → common corpus terms down-weighted
semantic word relation	Word2Vec → fixed-size word vectors; average for document
unordered label	StringIndexer then One-Hot to remove ordinality
many feature columns	VectorAssembler → one `features` vector

17 · Worked 2 – Classifier Metrics

CH 6 WORKED

Given: spam positive; TP=40, FP=10, FN=5, TN=45; total 100.

ACTUAL \ PREDICTED		
	+ / +	− / +
+ / −	**TP 40**	**FN 5**
− / −	**FP 10**	**TN 45**

P uses FP · R uses FN

Source-derived confusion matrix; label the positive class first.

```
Accuracy=(40+45)/100=0.85
Precision=40/(40+10)=0.80
Recall=40/(40+5)=0.889
F1=2(.80)(.889)/(.80+.889)=0.842
```

Interpret: recall exceeds precision: the model catches most spam but accepts more false alarms. Accuracy alone can hide imbalance.

18 · Judgement Cards

CH 6–7

IF YOU SEE…	DO / STATE
pure decision-tree branch	H=0; leaf
candidate attributes	largest IG wins
tree perfect on train, weak on test	overfit → prune / reduce maxDepth
imbalanced labels	precision, recall, F1; not accuracy alone
unlabelled numeric points	k-Means; choose k by elbow
parallel centroid update	combine sums and counts, not local means
single-tree high variance	random forest: bootstrap trees + feature subsets; majority vote
parallel tree learning	vertical = data parallelism; horizontal = result parallelism by tree nodes

METRIC DENOMINATOR *Precision asks "of predicted positives?" → FP. Recall asks "of actual positives?" → FN.*

18b · Tree + Cluster Oral Check

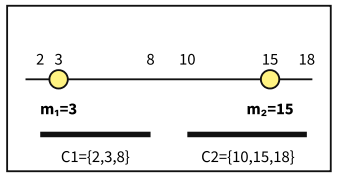
CH 6–7 FAQ

ID3 recital: entropy of current set → for each candidate, branch entropies → size-weighted average → parent minus average → split on maximum gain → recurse. A pure node has H=0. Pruning removes subnodes that do not improve generalisation; a random forest votes across bootstrap-trained trees and feature subsets.

19 · Worked 3 – k-Means

CH 7 WORKED

Given: D={2,3,8,10,15,18}, k=2, m_1=3, m_2=15; distance |x−m|.



Source redraw: nearest-centroid assignment on a 1-D line.

1. Assign: 2,3,8 → C1; 10,15,18 → C2. At 8: 5<7; at 10: 7>5.
2. Update: m_1=(2+3+8)/3=**4.33**; m_2=(10+15+18)/3=**14.33**.
3. Reassign. At 8: 3.67<6.33; at 10: 5.67>4.33. Nobody moves → **converged**.

TRAP *"One update" is not proof of convergence: run the membership check after updating.*

20 · Worked 4 – User CF

CH 8 WORKED

USER	MATRIX	INCEPTION	INTERSTELLAR	DUNE
Alice	5	3	4	?
Bob	4	2	5	5
Carol	5	3	3	4

ALICE–BOB

```
numerator=5·4+3·2+4·5=46
denominator=√50·√45≈47.43
sim≈0.97
```

ALICE–CAROL

```
numerator=46;
denominator=√50·√43≈46.37
sim≈0.99
```

```
k=1/(.97+.99)=0.510
D̂une=.510[(.97·5)+(.99·4)]
=.510(8.81)=4.49≈4.5
```

Both highly similar neighbours rate Dune well → recommend. Use only co-rated items for similarity, then neighbours who rated the target item.

SIMILARITY → NORMALISE → PREDICT

.97, .99	k=.510	4.49

must remain inside rating scale

Answer check: normalisation keeps the prediction in range.

20b · CF Judgement Cards

CH 8 FAQ

IF YOU SEE…	CHOOSE / EXPLAIN
similar rating rows	user-based CF

21 · Worked 5 – Windows + SHJ

CH 9–10 WORKED

Window data: t=1…6 values 10,20,30,40,50,60.

WINDOW	INTERVALS	AVERAGES
Tumbling size 2	[1–2),[3–4),[5–6]	**15,35,55**
Sliding size 2, slide 1	[1–2),[2–3),[3–4),[4–5),[5–6]	**15,25,35,45,55**

Write intervals before calculating. Tumbling has three outputs from six inputs; sliding overlaps and has five.

SHJ: r1, r2, s1, s3, r3

r1 r2 s1 s3 r3

each: probe other → emit → insert own

after s1	after r3
emit key 1	emit key 3

final outputs: orderID 1 and 3

Source redraw: arrival order does not lose a match.

1. r1: probe HTS ∅; insert HTR {1}.
2. r2: probe HTS ∅; insert HTR {1,2}.
3. s1: probe HTR → emit 1; insert HTS {1}.
4. s3: no match; insert HTS {1,3}.
5. r3: probe HTS → emit 3; insert HTR. Outputs: **1,3**.

WHY TWO TABLES? *Whichever partner arrives first waits in its own table for the later tuple.*

23 · Classic Traps

ALL CHAPTERS

WRONG	REPAIR
divide whole time by p	separate serial fraction first
"hash makes ranges local"	hash localises equal keys only
fan-in B	fan-in B−1
average partial AVGs	carry sum + count
entropy in base 2	use taught $\log_{10}$ example
accuracy only	P/R/F1 for imbalance
centroids updated, stop	reassign; check membership
cosine over missing cells	co-rated items only
omit CF normaliser	divide by Σ|sim|
insert then probe SHJ	probe other, then insert own

VELOCITY QUICK-FIRE

Kafka: topics split into partitions; offsets increase only within one partition; address = (topic, partition, offset). Reads do not delete records; consumers track position. Brokers store/replicate partitions.

Windows: time-based windows fix duration while tuple count varies with arrival rate; tuple/count windows fix how many records enter. Tumbling has slide=size and no overlap. Sliding has slide<size, so a record can contribute to several outputs. List the intervals before aggregating.

22 · Reduction Drill

CH 11 WORKED

REDUCE TIME → MERGE SENSORS

A: 20,22,24 mean = 22°C

B: 21,23,25 mean = 23°C

merge = 22.5°C

Source redraw: same-thing sensor array, Reduce then Merge.

Window classification: 6-month rolling average sliding one record → overlap and no granularity reduction; outputs equal inputs. For sensors, Reduce A=(20+22+24)/3=22°C; Reduce B=(21+23+25)/3=23°C; Merge=(22+23)/2=**22.5°C**. For plain aligned averages, Merge-then-Reduce matches; rate, reliability or units can make order matter.

IF YOU SEE…	DO / SAY
same measured thing	Reduce/Merge in either taught order
different things/units	Normalize around merge
arrival exceeds capacity	load shed: accuracy ↔ throughput
late event-time tuples	watermark bounds tolerated lateness/state
unbounded stream join	window expiry bounds memory

24 · Exam-Morning Recital

STUDY STRATEGY

1. **Volume:** speed-up vs scale-up; partitioning × query; join strategy + dominant network cost; sort pass recurrence; two-phase group-by.
2. **Complexity:** pipeline roles; $\log_{10}$ entropy/IG; confusion metrics; k-Means assign/update/stop; cosine + normalised prediction.
3. **Velocity:** Kafka address; event vs processing time; tumble vs slide; SHJ probe-then-insert; watermark; reduction class.
4. Cover every answer here. Recite the method, reproduce the formula, redo the numbers, then explain the result in one sentence.

LAST CHECK *Direction words earn marks: same vs bigger task; exact vs range; small vs both large; partial vs raw shuffle; predicted vs actual; event vs processing time.*

CLOSED-BOOK RETRIEVAL SEQUENCE

For every method, say five things without looking: **input → decision rule → calculation/update → stop condition → interpretation**. For every comparison, name both alternatives and the switching condition: hash/range/round-robin; broadcast/disjoint; two-phase/redistribution; classification/clustering; memory/model CF; tumbling/sliding; same-thing/different-thing sensor arrays.

Forward this to your tutorial group

SHARED BY _____

Your exam is closed book: nothing printed goes in. Use this sheet to revise and self-test, not to carry.

Monash Semester 2 final assessment period, 2–18 November 2026 (check your personal exam timetable).