Shanup.pdf
PDF

Week 4.pdf
PDF

Week 5.pdf
PDF

Notes from course .pdf
PDF

PROMPT START
I learn best when information is delivered in a structured, layered way:
Start simple → build up → reach expert nuance
Use ADHD-friendly formatting: short bullets, punchy phrases, clear headers
Use metaphors, flowcharts, if-then logic, and examples when helpful
Avoid long paragraphs unless I explicitly ask
For any new topic I am studying, follow this exact sequence:

SECTION 1 — Ultra-Short Overview (10 seconds)
Give me the "what it is" and "why it matters" in ONE sentence each.

SECTION 2 — Beginner Layer
Explain the fundamentals in:
5–10 bullets
extremely clear and simple
using tiny examples with numbers if appropriate
include quick "do / don't" rules

SECTION 3 — Intermediate Layer
Now expand slightly:
How this fits into the larger pipeline
Common traps, misconceptions, and best practices
When to use the technique and when NOT to
Where it sits relative to alternatives (small comparison table)

SECTION 4 — Expert Layer
Give me:
edge cases
nuanced decision-making
model / data-type compatibility
performance implications
what professionals look for that beginners miss
Make this as practical as possible.

SECTION 5 — Visual Thinking
Provide:
an ASCII flowchart
an IF-THEN decision map
a compact equation cheat-sheet if relevant
Keep these simple enough to memorise.

SECTION 6 — Apply to a Concrete Example
Take a small synthetic dataset or an example similar to the coursework and:
show the steps
explain what each step achieves
interpret outputs like a tutor sitting next to me

SECTION 7 — Reflection Prompts
Give me:
3–5 questions I should ask myself
what a good answer vs. weak answer looks like
This helps me link the topic to assessment criteria.

SECTION 8 — What to Do Next
Suggest:
what to practice
what to revise
what pitfalls to check for in my own work
Keep this short and actionable.

Important style notes
Be concise.
Don't assume prior steps are obvious — spell out sequencing clearly.
If vocabulary is required, explain it with one friendly example.
Always relate the topic back to data pipelines, feature engineering, dimensionality reduction, EDA, and model performance, because that's my conceptual anchor.
PROMPT END

Got it.
I'll follow that structure **exactly**.

What topic do you want me to explain first?
(e.g. *one-hot encoding*, *PCA*, *multicollinearity*, *z-score anomaly detection*, *t-SNE vs PCA*, *feature selection*, *linear regression assumptions*, etc.)

Name the topic and I'll start immediately at **Section 1 — Ultra-Short Overview**.

Model_solution_CAM_DS_C101_Activit...
PDF

I'm looking at activity 418, and I want to ask some questions about that. I'm going to upload a PDF for you as well, so you have that.

Got it. I've got the **Activity 4.1.8 model solution PDF** loaded and readable Model_solution_CAM_DS_C101_Acti... .

You can go ahead and ask your questions now.

To make this efficient (and not waffle):

- Ask **one question at a time**, or a **clearly numbered list**
- Say **which step** you're confused about (e.g. *dropping CarName*, *ordinal vs one-hot*, *VIF vs MI*, *why remove citympg before enginesize*, *RFE at the end*, etc.)
- If you want, I can answer **using your strict 8-section structure** or **just laser-target the confusion** and explain it plainly

Fire away.

Sources

So what would I get out of this? Is you've got whatever you get any kind of data you're gonna just load it and you're gonna split the target immediately right and then you're gonna split it into a categories and numerics and then you're gonna drop the ID one and then for categories account the values which is a nice test that there's another test we could do and known foso the number of unique values which was useful we dropped car name dropped car ID and then the cylinder mapping is interesting because those all numbers and one bigger than the other door mapping makes sense and then just one haunted code all the other stuff but it has dropped first so I just wanted to make sure about that one heart and coding, if you do on on fuel type and there are four fuel types on fuel types got two types you could drop one because one's true, yeah cause of one's true than the other ones fault you can get that through logic I don't get I guess that would make them correlate so that would improve it would decrease the colonial that messes with your data. Then you do info again. Got a bunch of bulls now and then you've got scaling that's where I'm stuck the standard scale that we didn't even look at this stuff. They didn't even look at our lies. Nothing we just scaled them so it should be not. I've looked at his Instagram scatter graphs and box and whiskers for outliers for possible histograms and for like a log

Good question. You're basically asking "**what is the actual workflow logic here, and why does scaling suddenly happen without EDA or outlier checks?**"

Let's pin it down cleanly, using *Activity 4.1.8* as the anchor Model_solution_CAM_DS_C101_Acti... .

SECTION 1 — Ultra-Short Overview (10 seconds)

What it is: This activity is teaching a **repeatable feature-engineering → feature-selection pipeline**, not exploratory storytelling.

Why it matters: Feature selection methods like **VIF, MI, and linear regression** behave badly if inputs aren't numerically comparable.

SECTION 2 — Beginner Layer

What you actually "get" from this activity

Think of it as **a checklist you can reuse on any tabular regression dataset**:

- Load data
- Split **target (y)** immediately → avoids leakage
- Split **numeric vs categorical**
- Drop **ID-like junk** (no signal)
- Encode categories properly
- **Scale before model-based selection**
- Diagnose **multicollinearity**
- Select features **without killing performance**

Key beginner rules (do / don't)

DO

- Drop IDs early (`car_ID`)
- Map *ordinal text* → numbers (`two → 2` , `four → 4`)

- One-hot encode *true nominal* variables
- Scale before **VIF / regression**

DON'T

- Treat IDs or names as signal
- Ordinal-encode things like fuel type
- Run VIF on unscaled mixed-unit data
- Assume EDA is mandatory for every task

SECTION 3 — Intermediate Layer

Why `drop_first=True` in one-hot encoding is correct

Fuel type example:

```nginx
fueltype_gas
1 → gas
0 → diesel
```

Copy code

If you kept both:

```nginx
fueltype_gas + fueltype_diesel = 1   (always)
```

Copy code

That causes **perfect multicollinearity**.

So yes — your logic is right:

> "If one is true, the other is false, so we can infer it."

That's *exactly* why one column is dropped.

Why scaling happens "suddenly" (this is the key confusion)

They are **not doing EDA here**.
They are doing **model-centric feature selection**.

Scaling is required because:

Method	What breaks if you don't scale
Linear regression	Coefficients distorted
VIF	Inflated by unit size
MI (sometimes)	Distance-based bias
RFE	Weight magnitude bias

Example:

```nginx
engine_size ≈ 300
mpg ≈ 20
```

Copy code

Without scaling:

- The model "thinks" engine size matters more
- Not because it does
- But because it's numerically larger

That's **wrong signal**.

When to use EDA vs when to skip it

EDA is for:

- Understanding distributions
- Detecting data quality issues
- Communicating insights

This activity is for:

- Mechanical feature selection
- Model performance comparison
- Reproducibility

So skipping histograms and boxplots here is **intentional**, not sloppy.

SECTION 4 — Expert Layer

Why they scale *before* VIF (this matters)

VIF formula depends on **R^2 between predictors**.

If variables are on wildly different scales:

- Numerical instability increases
- Correlations are exaggerated
- VIF ranking becomes noisy

Scaling:

- Does **not** change correlation structure
- **Does** stabilize computation

That's why boolean columns are converted to `0/1` *before* VIF.

Why no outlier handling here

Outliers are **a modelling choice**, not a default rule.

In this dataset:

- Cars with extreme prices are *real*
- Removing them would distort the business question
- Linear regression can tolerate some leverage if interpreted carefully

Professionals remove outliers **only when justified**, not by habit.

SECTION 5 — Visual Thinking

Pipeline flow (memorise this)

```sql
Raw data
   ↓
Drop target
```

```
   ↓
Drop IDs
   ↓
Ordinal map (where justified)
   ↓
One-hot encode (drop_first)
   ↓
Scale numeric features
   ↓
Convert bool → int
   ↓
VIF + MI
   ↓
Iterative feature removal
   ↓
Model check (Adj R²)
```

Copy code

IF–THEN decision map

css

```css
IF feature has order → ordinal encode
IF feature has no order → one-hot encode
IF using regression / VIF → scale
IF VIF high AND MI low → drop
IF VIF high BUT MI high → pause
IF Adj R² drops → feature was important
```

Copy code

Equation cheat-sheet

- StandardScaler

ini

```ini
z = (x − μ) / σ
```

Copy code

- VIF

ini

```ini
VIF_j = 1 / (1 - R²_j)
```

Copy code

SECTION 6 — Concrete Example

Say we skip scaling:

ini

```ini
enginesize = 300
horsepower = 150
citympg = 20
```

Copy code

Regression “sees”:

```
300 >> 20
```

Copy code

After scaling:

```ini
enginesize = 1.2
horsepower = 0.9
citympg = -0.8
```

Copy code

Now:

- Importance reflects **relationship**
- Not **measurement units**

That's the whole point.

SECTION 7 — Reflection Prompts

Ask yourself:

1. **Can this feature be inferred from others?**
   - Good: "Yes → drop one dummy"
   - Weak: "I kept everything"
2. **Does scaling change meaning or just magnitude?**
   - Good: "Magnitude only"
   - Weak: "I'm not sure"
3. **Did model performance guide decisions?**
   - Good: "Adj R^2 checked every time"
   - Weak: "I removed things because VIF was high"

SECTION 8 — What to Do Next

Practice

- Re-run this pipeline **without scaling** and compare VIF chaos
- Try log-transforming `price` and re-running selection

Revise

- Dummy variable trap
- VIF vs MI roles

Check your own work

- Are you scaling before regression?
- Are you removing features without checking model impact?

Bottom line (tell-it-like-it-is)

This activity is **not about EDA**.
It's about learning how to **systematically reduce features without lying to your model**.

If you want, next we can:

- Redo this pipeline *with EDA added*
- Or rewrite it as a **one-page exam-ready checklist**
- Or contrast this with **tree-based models where scaling doesn't matter**

Just say which.

Sources

ChatGPT 5.2 Share ···

tell me about MI and VIF scores and industry benchmarks again

Below is the **clean, exam-safe, industry-real explanation of MI vs VIF**, exactly in the layered way you asked. No fluff. No mysticism. This is how professionals actually use them.

SECTION 1 — Ultra-Short Overview (10 seconds)

MI: measures how much a feature helps predict the target.

VIF: measures how much a feature is *redundant with other features*.

SECTION 2 — Beginner Layer

Mutual Information (MI) — "Does this feature matter?"

- Measures **dependency between X and y**
- Captures **linear + non-linear** relationships
- Higher score = more predictive power
- Unitless, relative score (not absolute)

Tiny example

- `engine_size → price` → high MI
- `car_colour → price` → near-zero MI

DO

- Use MI to rank features by usefulness
- Compare features **relative to each other**

DON'T

- Treat MI as "statistical significance"
- Compare MI values across datasets

Variance Inflation Factor (VIF) — "Is this feature redundant?"

- Measures **collinearity between X variables**
- High VIF = feature can be predicted by other features
- Based on regression between predictors

Tiny example

- `citympg ↔ highwaympg` → high VIF
- `engine_size ↔ horsepower` → high VIF

DO

- Use VIF to diagnose multicollinearity
- Remove features **iteratively**

DON'T

- Use VIF to decide importance
- Run VIF without scaling

SECTION 3 — Intermediate Layer

Where they sit in the pipeline

powershell

Copy code

```
Raw features
   ↓
Encoding + scaling
   ↓
MI → importance vs target
   ↓
VIF → redundancy vs features
   ↓
Model check (Adj R²)
```

They answer **different questions**:

Question	Tool
"Does this feature help prediction?"	MI
"Does this feature duplicate others?"	VIF

Common traps

- ❌ Dropping features **only** because VIF > 5
- ❌ Keeping features **only** because MI is high
- ❌ Treating VIF like a hard rule instead of a warning light

Correct mindset:

> MI decides *what matters*
> VIF decides *what overlaps*

SECTION 4 — Expert Layer (Industry Reality)

Industry benchmark ranges (this is what assessors expect)

VIF thresholds (guidelines, not laws)

VIF	Interpretation	Industry action
< 2	Safe	Keep
2–5	Mild collinearity	Monitor
5–10	High	Investigate
> 10	Severe	Likely remove
> 30	Toxic	Almost always remove

⚠️ **Exception:**
If VIF is high **and** MI is very high → **pause**, test removal impact.

MI benchmarks (relative, not absolute)

There are **no universal cut-offs**, but in practice:

MI rank	Meaning
Top 10–20%	Core drivers
Middle 60%	Contextual

MI rank	Meaning
Bottom 20%	Noise candidates

Professionals look at:

- **Rank stability**
- **Consistency across folds**
- **Impact on model performance**

Why MI and VIF can "disagree" (this is important)

Example:

- `citympg` → high MI
- `highwaympg` → high MI
- Both → high VIF

Interpretation:

- They both matter
- They encode the *same signal*
- Keep one, not both

This is **not a contradiction** — it's the point.

SECTION 5 — Visual Thinking

Mental model (memorise this)

```ini
MI = Signal strength
VIF = Signal duplication
```

IF–THEN decision map

```powershell
IF MI low AND VIF high → drop immediately
IF MI high AND VIF low → keep
IF MI high AND VIF high → test removal
IF Adj R² drops → feature was essential
IF Adj R² stable → redundancy confirmed
```

Equation intuition (no maths panic)

- **MI**: "How surprised am I about y after seeing x?"
- **VIF**: "How well can other X's predict this X?"

SECTION 6 — Concrete Example (from your activity)

From the table you saw:

- `enginesize`
  - MI: very high
  - VIF: very high

They removed it → **Adjusted R^2 collapsed**.

Tutor interpretation:

> "Yes it's collinear, but it carries unique economic meaning. Keep it."

That's professional judgement, not contradiction.

SECTION 7 — Reflection Prompts (assessment-ready)

Ask yourself:

1. **Am I removing features for mathematical or predictive reasons?**
   - Good: "Predictive performance checked"
   - Weak: "VIF said so"
2. **Do remaining features still explain the business story?**
   - Good: "Engine capacity still represented"
   - Weak: "The model is smaller"
3. **Did I validate every removal with the model?**
   - Good: "Adj R^2 tracked"
   - Weak: "Assumed safe"

SECTION 8 — What to Do Next

Practice

- Remove a high-MI, high-VIF feature and *prove* the damage
- Try MI-only selection and watch collinearity explode

Revise

- Dummy variable trap
- Adjusted R^2 vs R^2

Pitfall check

- Are you treating VIF as a rule instead of a diagnostic?
- Are you explaining *why* a feature stayed despite high VIF?

Final blunt truth

- **MI tells you what matters**
- **VIF tells you what overlaps**
- **The model decides who survives**

If you want next, I can:

- Turn this into a **one-page exam justification**
- Compare MI/VIF with **tree-based importance**
- Or show **what changes if this were PCA instead**

Just say which.

OK I've moved on with the activity now. I've gone through the whole thing. Just just give me a step-by-step which are the decisions they made and what they dropped and now that effect affected R squared and is all of this effort worth it. We're going from .9 to .88 is that worth losing seven features?

SECTION 1 — Ultra-Short Overview (10 seconds)

What it is: A step-by-step feature-selection workflow that trades **complexity** for **nearly the same predictive power**. Model_solution_CAM_DS_C101_Acti...

Why it matters: It shows you how to reduce features while keeping model performance stable and making the model easier to explain/use. Model_solution_CAM_DS_C101_Acti...

SECTION 2 — Beginner Layer

The decisions, in plain steps (what they did, in order)

1. **Split target**
   - `y = price`, drop `price` from X. Model_solution_CAM_DS_C101_Acti...
2. **Identify categorical vs numeric**
   - Then inspect unique values to spot "problem" categoricals. Model_solution_CAM_DS_C101_Acti...
3. **Drop high-cardinality category**
   - Dropped `CarName` (147 unique values / 205 rows). Model_solution_CAM_DS_C101_Acti...
4. **Drop obvious non-signal**
   - Dropped `car_ID`. Model_solution_CAM_DS_C101_Acti...
5. **Ordinal-encode true ordinals**
   - `cylindernumber` mapped to 2,3,4,5,6,8,12
   - `doornumber` mapped to 2,4 Model_solution_CAM_DS_C101_Acti...
6. **One-hot encode remaining nominals**
   - `drop_first=True` to avoid dummy-variable trap. Model_solution_CAM_DS_C101_Acti...
7. **Scale numeric/ordinal**
   - StandardScaler on continuous + ordinal features. Model_solution_CAM_DS_C101_Acti...
8. **Convert booleans to 0/1**
   - Because their VIF tool doesn't handle `bool` cleanly. Model_solution_CAM_DS_C101_Acti...
9. **Compute MI + VIF together**
   - MI = usefulness vs target, VIF = redundancy among predictors. Model_solution_CAM_DS_C101_Acti...
10. **Iteratively drop features (guided by VIF but checked by Adjusted R^2)**
11. **Finish with RFE**

- Take df6 and select top 13, and confirm Adj R^2 stays similar. Model_solution_CAM_DS_C101_Acti...

Do / Don't

- **Do** validate every drop with Adjusted R^2
- **Don't** drop purely by VIF if MI is high

SECTION 3 — Intermediate Layer

"What did they drop, and what happened to Adjusted R^2?"

Here's the sequence exactly as the notebook shows it:

Stage	What they dropped	Why	Adjusted R^2
Baseline	nothing	reference	**0.9083** Model_solution_CAM_DS_C101_Acti...
df1	`fueltype_gas`, `fuelsystem_idi`, `compressionratio`	highest VIF offenders	**0.9033** Model_solution_CAM_DS_C101_Acti...
df2	next batch incl. `carbody_sedan`, `enginetype_ohcf`, `carbody_wagon`, `carbody_hatchback`, `enginetype_...`	reduce more multicollinearity (batch)	**0.8889** Model_solution_CAM_DS_C101_Acti...

Stage	What they dropped	Why	Adjusted R^2
df3	`cylindernumber`, `carlength`, `wheelbase`	start trimming "important-ish" ones	**0.8885** Model_solution_CAM_DS_C101_Acti...
df4	`citympg`	high VIF, important-ish → test	**0.8884** Model_solution_CAM_DS_C101_Acti...
df5	`curbweight`	test removal	**0.8868** Model_solution_CAM_DS_C101_Acti...
df6	`horsepower`	test removal	**0.8872** (tiny improvement) Model_solution_CAM_DS_C101_Acti...
df7	`enginesize`	test removal	**0.7529** (collapse → keep enginesize) Model_solution_CAM_DS_C101_Acti...
RFE final	keep 13 features from df6	smaller model	**0.8811** Model_solution_CAM_DS_C101_Acti...

Key interpretation

- Most removals barely hurt performance.
- Removing **enginesize** nuked the model → it's a core driver, even if collinear.
  Model_solution_CAM_DS_C101_Acti...
- RFE trimmed further with only a small drop.

SECTION 4 — Expert Layer

Is the effort "worth it" for 0.90 → 0.88?

Tell-it-like-it-is: **sometimes yes, sometimes no** — depends on the business goal.

When it *is* worth it

- You need **interpretability** (stakeholders want "what drives price?")
- You need **simpler deployment** (fewer inputs to collect/clean)
- You're fighting **unstable coefficients** (multicollinearity makes linear model explanations shaky)
- You want better **generalisation** (less chance of overfitting in small-ish data)

When it's *not* worth it

- Your only KPI is predictive accuracy and you'll use a stronger model later (GBMs etc.)
- Feature collection is cheap and you don't care about explanation
- You're not using coefficients for decisions

The hidden win beginners miss

Adjusted R^2 dropping from **0.9083** → **0.8811** is a performance cost, yes.
But the benefit is:

- fewer features
- less collinearity
- more stable, defensible model story

That's often worth more than 0.027 Adj R^2.

SECTION 5 — Visual Thinking

ASCII flowchart

scss Copy code

```
Encode + scale
   ↓
MI (importance) + VIF (redundancy)
   ↓
Drop high VIF + low MI first
   ↓
Re-fit OLS → check Adj R²
   ↓
If Adj R² stable → keep dropping
If Adj R² drops hard → undo drop
   ↓
RFE to compress further
```

IF–THEN decision map

powershell Copy code

```powershell
IF VIF huge AND MI low → drop now
IF VIF huge BUT MI high → test drop (model decides)
IF Adj R² collapses → keep feature
IF Adj R² stable after dropping → redundancy confirmed
```

Cheat-sheet

- VIF: redundancy warning light
- MI: signal strength
- Adj R^2: "did we actually hurt the model?"

SECTION 6 — Apply to a Concrete Example

Think of `citympg` and `highwaympg`:

- both predictive (MI high-ish)
- also overlap heavily (VIF high)

  Dropping `citympg` barely changed Adj R^2 → it was mostly redundant signal.

  Model_solution_CAM_DS_C101_Acti...

Then `enginesize`:

- predictive AND overlapping

  But dropping it caused Adj R^2 to crash → it carried unique info the rest couldn't replace.

  Model_solution_CAM_DS_C101_Acti...

That's the "tutor next to you" lesson: **not all collinear features are disposable**.

SECTION 7 — Reflection Prompts

1. Am I optimising for **accuracy** or **explainability**?
   - Good: "Explainability matters; stable coefficients matter."
   - Weak: "Higher number always wins."
2. Did feature removal improve **stability** (VIF down) without killing performance?
   - Good: "Adj R^2 barely moved until we hit enginesize."
   - Weak: "I removed random stuff."
3. Would I rather collect/clean **20 inputs** or **13** forever?
   - Good: "13 reduces pipeline friction."
   - Weak: "Doesn't matter."

SECTION 8 — What to Do Next

- **Practice:** Run the same pipeline but evaluate with **cross-validation RMSE**, not just Adj R^2 (more realistic).
- **Revise:** Dummy variable trap, MI vs VIF roles, why Adj R^2 is used.
- **Pitfall:** Don't conclude "feature selection is pointless" from a small R^2 change — the value is often **deployability + interpretability**, not just the metric.

If you tell me what your goal is (prediction-only vs explainability), I'll give you the correct "worth it / not worth it" verdict for *your* use case.

Sources

Copy of CAM_DS_C101_Demo 4.2.2.ipy...
PDF

great now let's go thoguh this one

Perfect — this one is **PCA done properly**, and it answers a *different question* from MI/VIF.
I'll walk it **step-by-step**, exactly what decisions were made, **why**, and **what you should conclude**, anchored to Demo 4.2.2 Copy of CAM_DS_C101_Demo 4.2.2.... .

SECTION 1 — Ultra-Short Overview (10 seconds)

What it is: PCA compresses many correlated features into fewer uncorrelated components.
Why it matters: You trade **interpretability** for **dimensionality reduction, speed, and stability**.

SECTION 2 — Beginner Layer

The concrete decisions they made (in order)

1. **Chose a high-dimensional dataset**
   - MNIST: **784 pixel features per image** Copy of CAM_DS_C101_Demo 4.2.2....
   - Classic case where feature selection is impossible manually
2. **Immediately scaled the data**
   - `StandardScaler()` before PCA Copy of CAM_DS_C101_Demo 4.2.2....
   - PCA is variance-based → scale is **non-negotiable**
3. **Reduced sample size**
   - 70,000 → **5,000 rows** (random subset) Copy of CAM_DS_C101_Demo 4.2.2....
   - Decision = computational practicality, not statistics
4. **Applied PCA with 2 components**
   - Purpose: **visualisation**, not modelling Copy of CAM_DS_C101_Demo 4.2.2....
5. **Plotted the 2D embedding**
   - Saw partial digit clustering (but overlap) Copy of CAM_DS_C101_Demo 4.2.2....
6. **Increased components (20, 100, 784)**
   - Purpose: **variance analysis**, not prediction Copy of CAM_DS_C101_Demo 4.2.2....
7. **Used explained variance + cumulative variance**
   - To decide "how many components are enough" Copy of CAM_DS_C101_Demo 4.2.2....
8. **Computed MI on PCA components**
   - To check which *components* still relate to the label Copy of CAM_DS_C101_Demo 4.2.2....

SECTION 3 — Intermediate Layer

What PCA is doing in the pipeline

csharp Copy code

```
Raw pixels (784)
   ↓
Highly correlated
   ↓
Scale
   ↓
Rotate feature space
   ↓
New axes = principal components
   ↓
Rank by variance captured
```

Key shift from the last activity:

- **MI/VIF removed features**
- **PCA replaces features**

You don't "drop pixel 417" — you **re-express all pixels**.

Explained variance decisions (this is examinable)

From the demo results Copy of CAM_DS_C101_Demo 4.2.2.... :

# Components	Variance Explained	Interpretation
20	~40%	Too lossy
100	~80%	Reasonable compression
784	100%	No reduction

Decision logic:

- Look for the **elbow** in cumulative variance
- Accept loss if task allows it

SECTION 4 — Expert Layer

Why PCA is fundamentally different from MI/VIF

Aspect	MI / VIF	PCA
Uses target?	MI yes, VIF no	❌ No
Preserves meaning?	✅ Yes	❌ No
Removes features?	Yes	No (recombines)
Handles multicollinearity?	Manually	Automatically
Interpretability	High	Low

This is why PCA is **not** used when:

- Stakeholders ask "*why*?"
- Features must remain meaningful
- Coefficients matter

The key professional insight (many miss this)

> **High variance ≠ high predictive power**

That's why they computed **MI on PCA components** Copy of CAM_DS_C101_Demo 4.2.2.... .

Example from MNIST:

- PC3, PC4 had higher MI than PC1
- Meaning: early variance ≠ best label signal

That's subtle, and very exam-worthy.

SECTION 5 — Visual Thinking

PCA mental model (memorise)

```yaml
Original axes: messy, correlated
New axes: clean, orthogonal
Variance = length along axis
```

IF–THEN decision map

```scss
IF features are many + correlated → consider PCA
IF interpretability matters → do NOT use PCA
IF visualisation needed → PCA(2)
IF modelling speed needed → PCA(n where CV ≈ 80–95%)
```

Equation cheat-sheet (conceptual)

- PCA finds directions that maximise:

```css
Var( X · w )
```

subject to:

```nginx
w ⊥ previous components
```

SECTION 6 — Concrete Example (MNIST)

- 784 pixels
- Adjacent pixels are highly correlated
- PCA compresses "stroke patterns"

Result:

- Digits cluster roughly
- But overlap → PCA alone ≠ classifier

Tutor interpretation: