Opdracht 2: Overfitting & Underfitting – Ontdek het Verschil

🎯 Doel van deze opdracht

In deze opdracht ga je zelf ontdekken:

- Wat underfitting is
- Wat overfitting is
- Hoe je goede generalisatie herkent
- Waarom testen op nieuwe data cruciaal is

Je doet dit door **regels te maken op basis van trainingsdata**, en daarna te controleren hoe goed je regels werken op **nieuwe, onbekende data**.

📦 Trainingsdataset (15 personen)

Persoon	Leeftijd	Dagen_per_week	Vriend_gaat_ook	Weer	Gaat_naar_sportschool
A	25	3	Ja	Goed	Ja
B	30	2	Nee	Slecht	Nee
C	22	4	Ja	Goed	Ja
D	35	1	Nee	Slecht	Nee
E	28	3	Ja	Goed	Ja
F	40	2	Nee	Goed	Nee
G	20	5	Ja	Goed	Ja
H	32	1	Nee	Slecht	Nee
I	26	4	Ja	Goed	Ja
J	38	2	Nee	Slecht	Nee
K	24	3	Ja	Slecht	Ja

Persoon	Leeftijd	Dagen_per_week	Vriend_gaat_ook	Weer	Gaat_naar_sportschool
L	29	2	Nee	Goed	Nee
M	21	4	Ja	Goed	Ja
N	33	1	Nee	Slecht	Nee
O	27	3	Ja	Goed	Ja

> 💡 Let op: in echte datasets komen uitzonderingen voor. Bijvoorbeeld Persoon K gaat ondanks slecht weer tóch naar de sportschool. Dat is bewust: niet alle data volgt perfect dezelfde logica.

🧪 Testdataset (nieuwe data)

Persoon	Leeftijd	Dagen_per_week	Vriend_gaat_ook	Weer	Gaat_naar_sportschool
P	23	3	Ja	Goed	Ja
Q	31	2	Nee	Slecht	Nee
R	19	4	Ja	Goed	Ja
S	36	1	Nee	Goed	Nee
T	25	2	Ja	Slecht	Ja

📝 Jouw opdracht

Stap 1 — Maak drie regels

🔹 Regel 1 — Underfitting (te simpel)

Gebruik **slechts één factor**.

Voorbeelden (kies er zelf één):

- "Als leeftijd < 25 → Ja" (te specifiek, mist veel gevallen)
- "Als dagen_per_week = 5 → Ja" (te specifiek, alleen voor extreme gevallen)
- "Als weer = Goed → Ja" (negeert andere belangrijke factoren)

Let op: Sommige simpele regels werken verrassend goed op deze dataset. Kies een regel die NIET goed past, zodat je ziet wat underfitting betekent. Het gaat erom dat je begrijpt WAAROM de regel te simpel is.

🔹 Regel 2 — Normale regel (goede generalisatie)

Gebruik **2 factoren** die logisch lijken.

Voorbeeld:

> "Als (dagen_per_week ≥ 3) AND (vriend_gaat_ook = Ja) → Ja"

Deze regel moet **logisch en compact** zijn, niet te specifiek.

🔹 Regel 3 — Overfitting (te complex)

Maak een regel die **alle factoren gebruikt** en **perfect op de trainingsdata past**.

Voorbeelden:

- Een regel met veel AND's en exacte waardes
- Een regel die elke combinatie hard-codeerd
- Alles matcht precies → 15/15 correct op training

Dit mag *niet logisch* zijn. Dit is opzet.

Stap 2 — Test je regels op de trainingsdata

Tel hoeveel rijen goed voorspeld worden.

💻 Optioneel: Gebruik het Python script

Je kunt het Python script `evaluate_regels_sportschool.py` gebruiken om automatisch je regels te testen:

1. Open het bestand `grader/evaluate_regels_sportschool.py`
2. Vervang de `raise NotImplementedError(...)` regels door je eigen code in de functies `rule1()`, `rule2()` en `rule3()`
3. Voer uit: `python evaluate_regels_sportschool.py`
4. Bekijk de resultaten

📊 **Of handmatig:**
Gebruik deze tabel:

Regel	Aantal correct (train)	Percentage (train)
Regel 1	…	…
Regel 2	…	…
Regel 3	…	…

Stap 3 — Test op nieuwe data (testdataset)

Vul dezelfde tabel in:

Regel	Aantal correct (test)	Percentage (test)
Regel 1	…	…
Regel 2	…	…
Regel 3	…	…

Stap 4 — Analyseer en beantwoord

Beantwoord:

1. Welke regel presteerde het beste op de **trainingsdata**?
2. Welke regel presteerde het beste op de **testdata**?
3. Wat valt op bij Regel 3 (overfitting)?
4. Waarom werkt een te complexe regel slecht op nieuwe data?
5. Wat zou volgens jou een betere regel zijn?

🎉 Klaar!

Twee groepen zullen dit presenteren.