Comparaison CLIPS / PYTHON

Prédiction de plusieurs exemples en Python :

```python
print(model.predict([[35,3566167161116111,35,3566167161116111]])) # 3
print(model.predict([[9,"356616111611161116111",35,"966161116111611116111"]])) # 4
print(model.predict([[35,"666111111",9,"356611161111"]])) # 1
print(model.predict([[17,"1766171171111",17,"1766171171111"]])) # 3
print(model.predict([[17,"966161116111611116111",9,"176616111611161116111"]])) # 2
print(model.predict([[17,"1661788161116111",1,"17661788161116111"]])) # 2
```

```
[3]
[4]
[1]
[2]
[2]
[2]
```

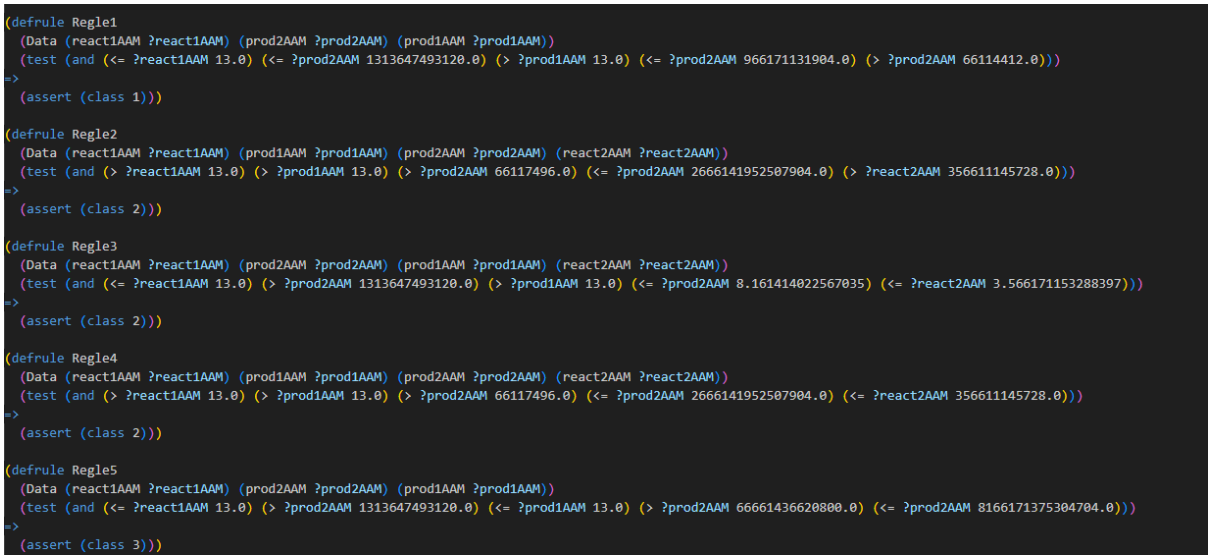
Prédiction de plusieurs exemples en CLIPS:

```
(assert (Data (prod1AAM 35) (prod2AAM 666111111) (react1AAM 9) (react2AAM 356611161111) (category 1))) ; Classe predite : 1 Regle utilisé : 1
(assert (Data (prod1AAM 17) (prod2AAM 1766171171111) (react1AAM 17) (react2AAM 1766171171111) (category 3))) ; Classe predite : 2 Regle utilisé : 4
(assert (Data (prod1AAM 35) (prod2AAM 9661676711) (react1AAM 9) (react2AAM 35661676711) (category 1))) ; Classe predite : 1 Regle utilisé : 1
(assert (Data (prod1AAM 17) (prod2AAM 3566116711) (react1AAM 35) (react2AAM 1766116711) (category 2))) ; Classe predite : 2 Regle utilisé : 4
(assert (Data (prod1AAM 17) (prod2AAM 66117111) (react1AAM 17) (react2AAM 17661711111) (category 3)))  ; Classe predite : 3 Regle utilisé : 20
(assert (Data (prod1AAM 17) (prod2AAM 66111711) (react1AAM 17) (react2AAM 17661171111) (category 4))) ; Classe predite : 4 Regle utilisé : 24
```

```
f-1     (Data (prod1AAM 35) (prod2AAM 666111111) (react1AAM 9) (react2AAM 356611161111) (category 1))
f-2     (class 1)
f-3     (Data (prod1AAM 17) (prod2AAM 1766171171111) (react1AAM 17) (react2AAM 1766171171111) (category 3))
f-4     (class 2)
f-5     (Data (prod1AAM 35) (prod2AAM 9661676711) (react1AAM 9) (react2AAM 35661676711) (category 1))
f-6     (Data (prod1AAM 17) (prod2AAM 3566116711) (react1AAM 35) (react2AAM 1766116711) (category 2))
f-7     (Data (prod1AAM 17) (prod2AAM 66117111) (react1AAM 17) (react2AAM 17661711111) (category 3))
f-8     (class 3)
f-9     (Data (prod1AAM 17) (prod2AAM 66111711) (react1AAM 17) (react2AAM 17661171111) (category 4))
f-10    (class 4)
```

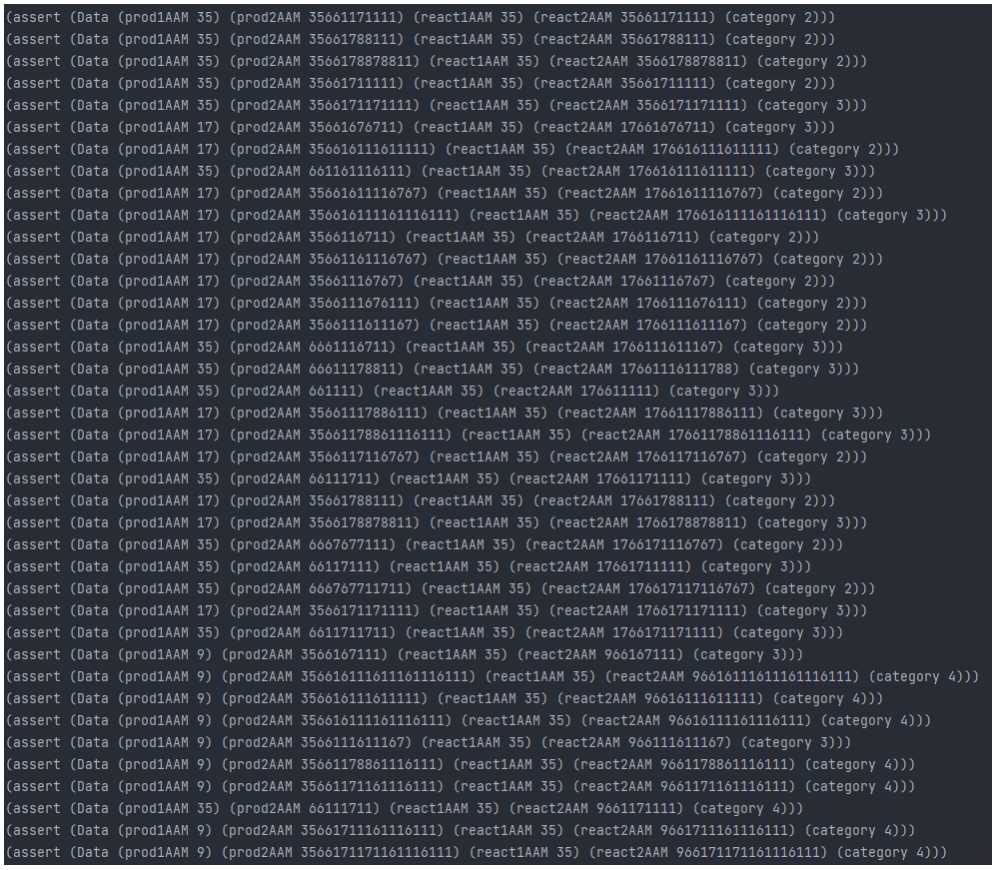
On peut voir que les deux modèles sont presque aussi précis même si le modele python avait plus tendance a se tromper car a chaque fois que l'on tente une prediction il se reparamètre, mais si on reste sur un parametrage maximal alors il sera plus precis que CLIPS.

Voici un exemple des règles générées qui ont été traduite en CLIPS :



```
(defrule Regle1
  (Data (react1AAM ?react1AAM) (prod2AAM ?prod2AAM) (prod1AAM ?prod1AAM))
  (test (and (<= ?react1AAM 13.0) (<= ?prod2AAM 1313647493120.0) (> ?prod1AAM 13.0) (<= ?prod2AAM 966171131904.0) (> ?prod2AAM 66114412.0)))
=>
  (assert (class 1)))

(defrule Regle2
  (Data (react1AAM ?react1AAM) (prod1AAM ?prod1AAM) (prod2AAM ?prod2AAM) (react2AAM ?react2AAM))
  (test (and (> ?react1AAM 13.0) (> ?prod1AAM 13.0) (> ?prod2AAM 66117496.0) (<= ?prod2AAM 2666141952507904.0) (> ?react2AAM 356611145728.0)))
=>
  (assert (class 2)))

(defrule Regle3
  (Data (react1AAM ?react1AAM) (prod2AAM ?prod2AAM) (prod1AAM ?prod1AAM) (react2AAM ?react2AAM))
  (test (and (<= ?react1AAM 13.0) (> ?prod2AAM 1313647493120.0) (> ?prod1AAM 13.0) (<= ?prod2AAM 8.161414022567035) (<= ?react2AAM 3.566171153288397)))
=>
  (assert (class 2)))

(defrule Regle4
  (Data (react1AAM ?react1AAM) (prod1AAM ?prod1AAM) (prod2AAM ?prod2AAM) (react2AAM ?react2AAM))
  (test (and (> ?react1AAM 13.0) (> ?prod1AAM 13.0) (> ?prod2AAM 66117496.0) (<= ?prod2AAM 2666141952507904.0) (<= ?react2AAM 356611145728.0)))
=>
  (assert (class 2)))

(defrule Regle5
  (Data (react1AAM ?react1AAM) (prod2AAM ?prod2AAM) (prod1AAM ?prod1AAM))
  (test (and (<= ?react1AAM 13.0) (> ?prod2AAM 1313647493120.0) (<= ?prod1AAM 13.0) (> ?prod2AAM 66661436620800.0) (<= ?prod2AAM 8166171375304704.0)))
=>
  (assert (class 3)))
```

On voit directement avec cet exemple que le modelé CLIPS n'est clairement pas adapte pour ce type d'IA, les décisions des règles sont trop aléatoires il est donc impossible de les constituer à partir de l'expertise d'un expert, de plus il est très peu pratique d'utiliser un dataset comme nous le faisons en python car nous sommes forces à devoir faire un assert pour chaque sample de notre dataset :



```
(assert (Data (prod1AAM 35) (prod2AAM 35661171111) (react1AAM 35) (react2AAM 35661171111) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 35661788111) (react1AAM 35) (react2AAM 35661788111) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 3566178878811) (react1AAM 35) (react2AAM 3566178878811) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 35661711111) (react1AAM 35) (react2AAM 35661711111) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 3566171171111) (react1AAM 35) (react2AAM 3566171171111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 35661676711) (react1AAM 35) (react2AAM 17661676711) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 356616111611111) (react1AAM 35) (react2AAM 176616111611111) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 661161116111) (react1AAM 35) (react2AAM 176616111611111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 35661611116767) (react1AAM 35) (react2AAM 17661611116767) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 356616111161116111) (react1AAM 35) (react2AAM 1766161111161116111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 3566116711) (react1AAM 35) (react2AAM 1766116711) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 35661161116767) (react1AAM 35) (react2AAM 17661161116767) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 35661116767) (react1AAM 35) (react2AAM 17661116767) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 3566111676111) (react1AAM 35) (react2AAM 1766111676111) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 3566111611167) (react1AAM 35) (react2AAM 1766111611167) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 6661116711) (react1AAM 35) (react2AAM 1766111611167) (category 3)))
(assert (Data (prod1AAM 35) (prod2AAM 66611178811) (react1AAM 35) (react2AAM 17661116111788) (category 3)))
(assert (Data (prod1AAM 35) (prod2AAM 661111) (react1AAM 35) (react2AAM 176611111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 35661117886111) (react1AAM 35) (react2AAM 17661117886111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 35661178861116111) (react1AAM 35) (react2AAM 17661178861116111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 3566117116767) (react1AAM 35) (react2AAM 1766117116767) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 66111711) (react1AAM 35) (react2AAM 17661171111) (category 3)))
(assert (Data (prod1AAM 17) (prod2AAM 35661788111) (react1AAM 35) (react2AAM 17661788111) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 3566178878811) (react1AAM 35) (react2AAM 1766178878811) (category 3)))
(assert (Data (prod1AAM 35) (prod2AAM 6667677111) (react1AAM 35) (react2AAM 1766171116767) (category 2)))
(assert (Data (prod1AAM 35) (prod2AAM 66117111) (react1AAM 35) (react2AAM 17661711111) (category 3)))
(assert (Data (prod1AAM 35) (prod2AAM 666767711711) (react1AAM 35) (react2AAM 176617117116767) (category 2)))
(assert (Data (prod1AAM 17) (prod2AAM 3566171171111) (react1AAM 35) (react2AAM 1766171171111) (category 3)))
(assert (Data (prod1AAM 35) (prod2AAM 6611711711) (react1AAM 35) (react2AAM 1766171171111) (category 3)))
(assert (Data (prod1AAM 9) (prod2AAM 3566167111) (react1AAM 35) (react2AAM 966167111) (category 3)))
(assert (Data (prod1AAM 9) (prod2AAM 356616111611161116111) (react1AAM 35) (react2AAM 9661611161116116111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 356616116111111) (react1AAM 35) (react2AAM 96616116111111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 356616111161116111) (react1AAM 35) (react2AAM 96616111161116111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 3566111611167) (react1AAM 35) (react2AAM 966111611167) (category 3)))
(assert (Data (prod1AAM 9) (prod2AAM 35661178861116111) (react1AAM 35) (react2AAM 9661178861116111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 35661171161116111) (react1AAM 35) (react2AAM 9661171161116111) (category 4)))
(assert (Data (prod1AAM 35) (prod2AAM 66111711) (react1AAM 35) (react2AAM 9661171111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 35661711161116111) (react1AAM 35) (react2AAM 9661711161116111) (category 4)))
(assert (Data (prod1AAM 9) (prod2AAM 356617117161116111) (react1AAM 35) (react2AAM 96617117161116111) (category 4)))
```

Bien que les deux IA soient basées sur un arbre de décision on voit très bien le contraste entre une IA symbolique et une IA numérique dans ce cas précis, en effet il est très peu pratique d'utiliser de très gros dataset et surtout dans des conditions ou l'expertise ne peut pas se faire sur ce modèle.

De plus, CLIPS ne sait pas lire les données si les chiffres sont trop grands.