

# Comparaison d'algorithmes de classification supervisée

---

Antoine Cornuéjols

AgroParisTech – INRAE Paris-Saclay

[antoine.cornuejols@agroparistech.fr](mailto:antoine.cornuejols@agroparistech.fr)

# Plan

---

1. Introduction
2. Comparaison sur les mêmes jeux de données
  1. T-test couplé
  2. Test de McNemar
  3.  $5 \times 2$  cv tests couplés
3. Comparaison sur des jeux de données différents
  1. Test de rang signé de Wilcoxon
4. Conclusions

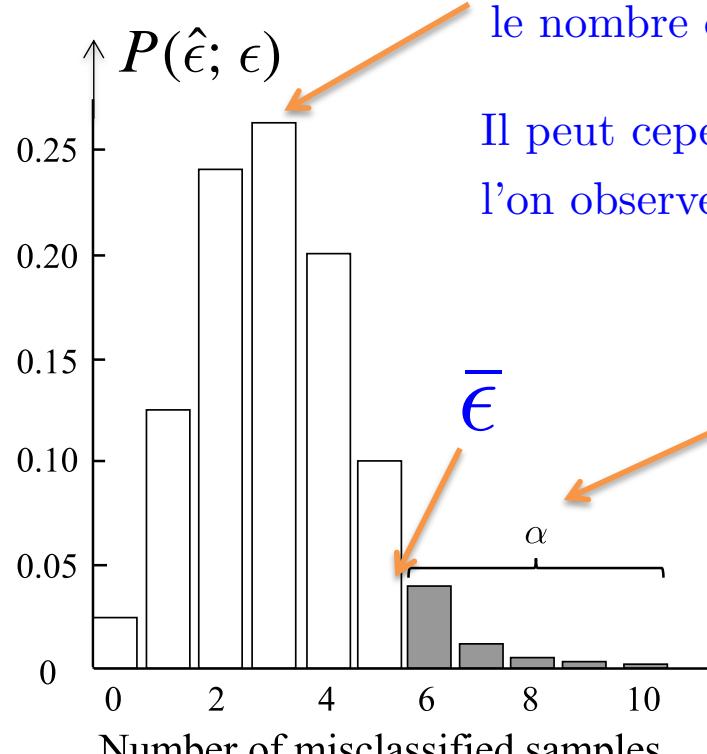
# Rappels : un algorithme de classification binaire

---

- Soit un classifieur d'erreur  $\epsilon$ 
  - La probabilité qu'il classe mal  $m'$  exemples parmi  $m$  est :
$$\binom{m}{m'} \epsilon^{m'} (1 - \epsilon)^{m-m'}$$
  - D'où la probabilité qu'il classe mal  $\hat{\epsilon} \times m$  exemples est :
$$P(\hat{\epsilon}; \epsilon) = \binom{m}{\hat{\epsilon} \times m} \epsilon^{\hat{\epsilon} \times m} (1 - \epsilon)^{m - \hat{\epsilon} \times m}$$
  - Ce qui est la probabilité que son **erreur apparente** sur un ensemble de test de taille  $m$  soit  $\hat{\epsilon}$

## Rappels : un algorithme de classification binaire (2)

- On voudrait tester si  $\epsilon \leq \epsilon_0$  (e.g.  $\epsilon \leq 0.3$ )



Distribution binomiale avec  $m = 10$  et  $\epsilon = 0.3$

Sous l'hypothèse que  $\epsilon = 0.3$   
le nombre d'erreurs le plus probable est 3 pour 10 exemples  
Il peut cependant arriver que  
l'on observe un nombre d'erreurs supérieur à  $\epsilon = 0.3$

$$\sum_{i=6}^{10} P(\hat{\epsilon} = \frac{i}{10} | \epsilon = 0.3) \leq \alpha = 0.10$$

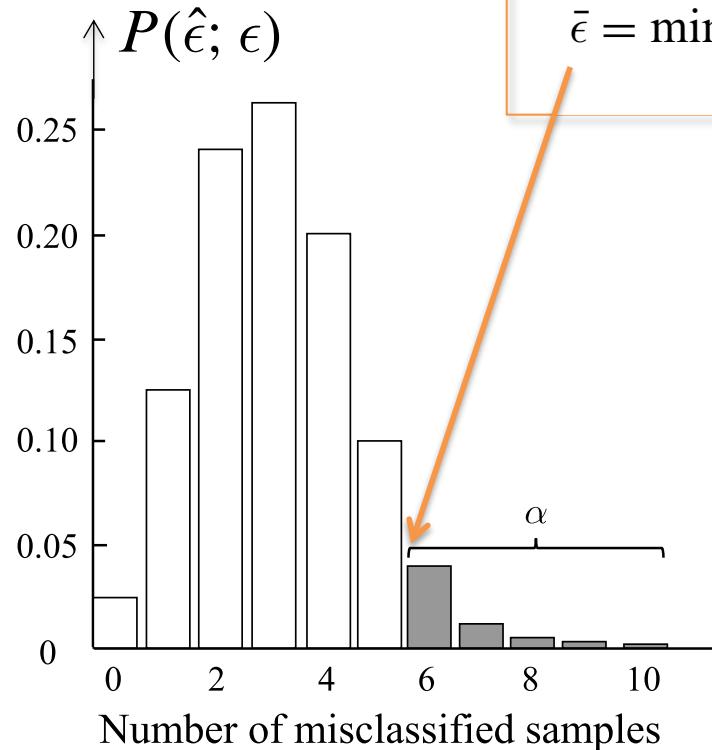
Probabilité que cela arrive pour un nb d'erreurs  $\geq 0.6$

Pour  $\alpha = 0.1$ , le plus petit taux d'erreur  $\bar{\epsilon}$  pour que cela arrive est 0.6

Si  $\epsilon$  plus petit, alors  $\bar{\epsilon}$  plus petit aussi

## Rappels : un algorithme de classification binaire (2)

- On voudrait tester si  $\epsilon \leq \epsilon_0$  (e.g.  $\epsilon \leq 0.3$ )



$$\bar{\epsilon} = \min \epsilon \text{ s.t. } \sum_{i=\epsilon \times m + 1}^m \binom{m}{i} \epsilon_0^i (1 - \epsilon_0)^{m-i} < \alpha$$

et  $P(\hat{\epsilon} > \bar{\epsilon}; \epsilon_0) < \alpha$

et encore plus si  $\epsilon < \epsilon_0$

Donc le taux d'erreur  $\epsilon$  de l'algorithme  
n'est pas  $> \epsilon_0$  avec niveau de confiance  $1 - \alpha$

Distribution binomiale avec  $m = 10$  et  $\epsilon = 0.3$

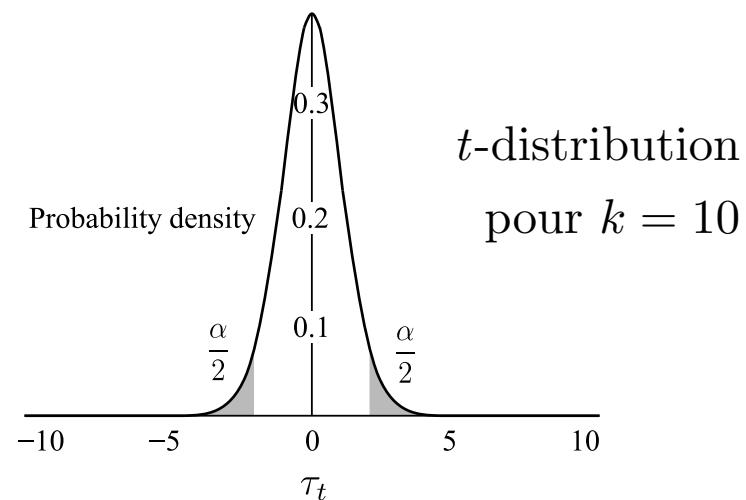
## Un algorithme et plusieurs tests (e.g. 10-CV)

- Soient  $\hat{\epsilon}_1, \hat{\epsilon}_2, \dots, \hat{\epsilon}_k$  les  $k$  taux d'erreur mesurés
- Alors on a la moyenne et la variance :

$$\mu = \frac{1}{k} \sum_{i=1}^k \hat{\epsilon}_i \quad \sigma^2 = \frac{1}{k-1} \sum_{i=1}^k (\hat{\epsilon}_i - \mu)^2$$

- En considérant que les  $k$  taux d'erreur sont des tirages i.i.d. du taux d'erreur en généralisation, alors la variable  $\tau_t$  suit une **distribution  $t$**  à  $k-1$  degrés de liberté.

$$\tau_t = \frac{\sqrt{k}(\mu - \epsilon_0)}{\sigma}$$



## Un algorithme et plusieurs tests (e.g. 10-CV) (2)

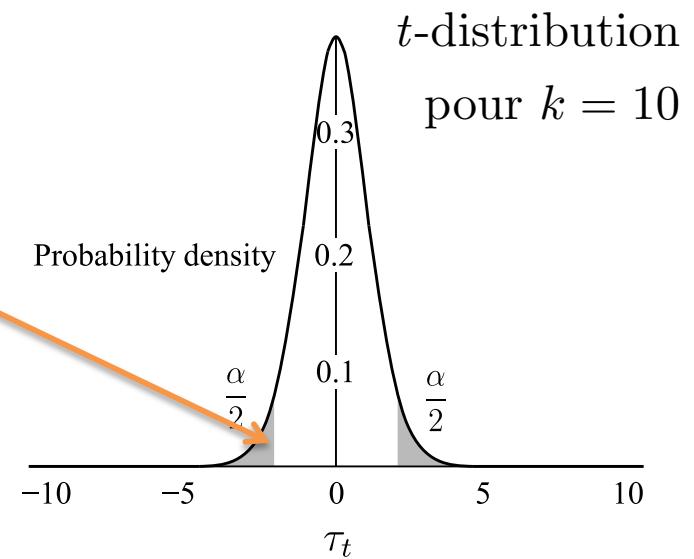
- Pour l'hypothèse  $\mu = \epsilon_0$  et le niveau de confiance  $\alpha$ , on peut calculer l'erreur observable maximale avec une probabilité  $1 - \alpha$  (ici avec une hypothèse à deux côtés (« two-sided »))
- Si  $\tau_t \in [t_{\alpha/2}, t_{\alpha/2}]$
- Alors l'hypothèse que  $\mu$  (taux d'erreur apparent) est significativement différent de  $\epsilon_0$  peut être rejetée

$$\tau_t = \frac{\sqrt{k}(\mu - \epsilon_0)}{\sigma}$$

Valeurs usuelles pour le t-test à deux côtés

$\alpha$	$k$				
	2	5	10	20	30
0.05	12.706	2.776	2.262	2.093	2.045
0.10	6.314	2.132	1.833	1.729	1.699

$$-t_{\alpha/2} \quad \text{and} \quad t_{\alpha/2}$$



Comparaison de **2** algorithmes  
sur **un** même jeu de données

# Comparaison d'algorithmes

---

- Principe général
  - Caractériser la **distribution de probabilité** sur les différences possibles de performances sous **hypothèse nulle  $H_0$**  : pas de différence en fait entre les algorithmes
  - Décider qu'il y a une différence si la **probabilité des différences observées est suffisamment petite sous  $H_0$** .  
(donc, c'est peu probable sous  $H_0$ )

# Le two-match-sample t-test

---

- Teste si la **différence** entre deux échantillons (de performances) est **significative**
  - Fonction de la **différence de moyennes**
  - Fonction de la **différence de variance**
- On fait souvent l'hypothèse suivante sur la **distribution de la différence des moyennes**
  - Tend vers une **loi normale**

## 2 algorithmes : t-test en validation croisée

---

- Soient deux algorithmes A et B avec les erreurs respectives :

$$\epsilon_1^A, \epsilon_2^A, \dots, \epsilon_k^A \quad \text{et} \quad \epsilon_1^B, \epsilon_2^B, \dots, \epsilon_k^B$$

- On calcule les différences **pour chaque paire** de résultats

$$\Delta_i = \epsilon_i^A - \epsilon_i^B$$

- Si les deux algorithmes sont de performances équivalentes, la **moyenne** des différences **devrait être proche de 0.**
- On réalise **donc un t-test** sur les valeurs  $\Delta_1, \Delta_2, \dots, \Delta_k$  avec l'hypothèse  $H_0$  : A et B sont de même performance

## 2 algorithmes : t-test en validation croisée (2)

- Soient deux algorithmes A et B
- Avec les différences de performances  $\Delta_i = \epsilon_i^A - \epsilon_i^B$
- On calcule la moyenne  $\mu$  et la variance  $\sigma^2$  des  $\Delta_i$

Et si

$$\tau_t = \left| \frac{\sqrt{k} \mu}{\sigma} \right| < t_{\alpha/2, k-1}$$

au niveau de signification  $\alpha$

**Alors**  $H_0$  (même performance) ne peut être rejetée

- Rq : Les  $\Delta_i$  ne sont pas i.i.d. d'où la suggestion de 5x2 validation croisée de Tom Dietterich (1998)

# Illustration : Comparaison de performances sur 10-CV

<i>Fold</i>	<i>Naive Bayes</i>	<i>Decision tree</i>	<i>Nearest neighbour</i>
1	0.6809	0.7524	0.7164
2	0.7017	0.8964	0.8883
3	0.7012	0.6803	0.8410
4	0.6913	0.9102	0.6825
5	0.6333	0.7758	0.7599
6	0.6415	0.8154	0.8479
7	0.7216	0.6224	0.7012
8	0.7214	0.7585	0.4959
9	0.6578	0.9380	0.9279
10	0.7865	0.7524	0.7455
avg	0.6937	0.7902	0.7606
stdev	0.0448	0.1014	0.1248

From [Peter Flach (2012) Machine Learning. *The art and science of algorithms that make sense of data*. Cambridge University Press]

# Illustration : Comparaison de performances sur 10-CV

Fold	Naive Bayes	Decision tree	Nearest neighbour
1	0.6809	0.7524	0.7164
2	0.7017	0.8964	0.8883
3	0.7012	0.6803	0.8410
4	0.6913	0.9102	0.6825
5	0.6333	0.7758	0.7599
6	0.6415	0.8154	0.8479
7	0.7216	0.6224	0.7012
8	0.7214	0.7585	0.4959
9	0.6578	0.9380	0.9279
10	0.7865	0.7524	0.7455
avg	0.6937	0.7902	0.7606
stdev	0.0448	0.1014	0.1248

Fold	NB-DT	NB-NN	DT-NN
1	-0.0715	-0.0355	0.0361
2	-0.1947	-0.1866	0.0081
3	0.0209	-0.1398	-0.1607
4	-0.2189	0.0088	0.2277
5	-0.1424	-0.1265	0.0159
6	-0.1739	-0.2065	-0.0325
7	0.0992	0.0204	-0.0788
8	-0.0371	0.2255	0.2626
9	-0.2802	-0.2700	0.0102
10	0.0341	0.0410	0.0069
avg	-0.0965	-0.0669	0.0295
stdev	0.1246	0.1473	0.1278
p-value	<b>0.0369</b>	<b>0.1848</b>	<b>0.4833</b>

La p-valeur dans la dernière ligne est calculée en utilisant la  $t$ -distribution avec  $k - 1 = 9$  degrés de liberté. Et seule la différence entre Naïve Bayes et les Arbres de Décision est jugée significative au niveau  $\alpha = 0.05$ . ( $0.0369 < 0.05$ )

From [Peter Flach (2012) Machine Learning. *The art and science of algorithms that make sense of data*. Cambridge University Press]

# Le test de McNemar

- Au lieu de regarder la différence de taux d'erreur, on peut **comparer les taux de classifications correcte et incorrecte**
- Si les performances de A et B sont équivalentes alors on devrait avoir :  $e_{01} = e_{10}$
- La variable  $|e_{01} - e_{10}|$  suit une distribution gaussienne
- Le **test de McNemar** considère la variable qui suit une loi du chi2
- $H_0$  ne peut être rejetée au niveau  $\alpha$  si

		Algorithm A	
		Correct	Incorrect
Algorithm B	Correct	$e_{00}$	$e_{01}$
	Incorrect	$e_{10}$	$e_{11}$

$$\tau_{\chi^2} = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$$

$$\tau_{\chi^2} = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} < \chi_{\alpha}^2$$

Comparaison de **2** algorithmes  
sur **plusieurs** jeux de données

## Remarque fondamentale

---

- La **performance** des algorithmes **dépend de** leur adéquation au jeu de données traité
- On ne peut plus faire l'hypothèse que les taux d'erreur sur les différents jeux de données sont distribués selon une **loi normale**



Il faut recourir à des tests **non paramétriques**

# Le test de rang signé de Wilcoxon

On calcule la somme des rangs :

$$W_{s1} = \sum_{i=1}^n I(d_i > 0) \text{rank}(d_i) \quad \text{et} \quad W_{s2} = \sum_{i=1}^n I(d_i < 0) \text{rank}(d_i)$$

où  $d_i = \text{Perf}_{\text{moy}}(h_2) - \text{Perf}_{\text{moy}}(h_1)$

Puis :

$$T_{\text{wilcox}} = \min(W_{s1}, W_{s2})$$

La distribution de  $T_{\text{wilcox}}$  peut être approchée par une loi normale quand  $n \geq 25$   
(sinon regarder dans une table)

- On calcule la statistique z

$$z_{\text{wilcox}} = \frac{T_{\text{wilcox}} - \mu_{T_{\text{wilcox}}}}{\sigma_{T_{\text{wilcox}}}}$$

- Avec :

$$\mu_{T_{\text{wilcox}}} = \frac{n(n+1)}{4} \quad \text{et} \quad \sigma_{T_{\text{wilcox}}} = \sqrt{\frac{n(n+1)(2n+1)}{24}}$$

# Le test de rang signé de Wilcoxon

## Application

$$T_{wilcox} = \min(W_{s1}, W_{s2}) = 17$$

Avec  $n-1 = 9$  degrés de liberté

Il faut  $T_{wilcox} < 8$  (one-sided)

ou  $T_{wilcox} < 5$  (two-sided)

(d'après la table pour  $p=0.05$ )

Table 6.4. Classifiers NB and SVM data ran on 10 realistic domains and used in the Wilcoxon test example

Domain no.	NB accuracy	SVM accuracy	NB-SVM	$ NB-SVM $	Ranks ( $ NB-SVM $ )	$\pm$ Ranks ( $ NB-SVM $ )
1	0.9643	0.9944	-0.0301	0.0301	3	-3
2	0.7342	0.8134	-0.0792	0.0792	6	-6
3	0.7230	0.9151	-0.1921	0.1921	8	-8
4	0.7170	0.6616	+0.0554	0.0554	5	+5
5	0.7167	0.7167	0	0	Remove	Remove
6	0.7436	0.7708	-0.0272	0.0272	2	-2
7	0.7063	0.6221	+0.0842	0.0842	7	+7
8	0.8321	0.8063	+0.0258	0.0258	1	+1
9	0.9822	0.9358	+0.0464	0.0464	4	+4
10	0.6962	0.9990	-0.3028	0.3028	9	-9

$$W_{s1} = 28$$

$$W_{s2} = 17$$

Donc on ne peut pas rejeter l'hypothèse nulle d'égalité entre les algorithmes à  $p = 0.05$

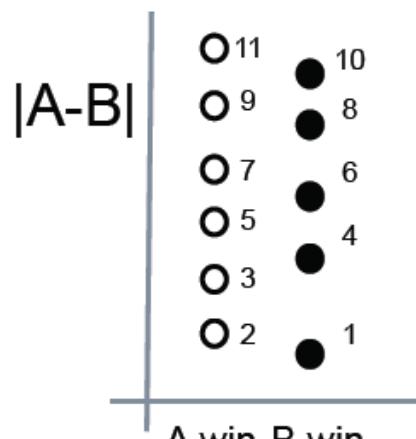
From [Japowicz & Shah (2011) "Evaluating Learning Algorithms. A classification perspective", p.235]

# Le test de rang signé de Wilcoxon

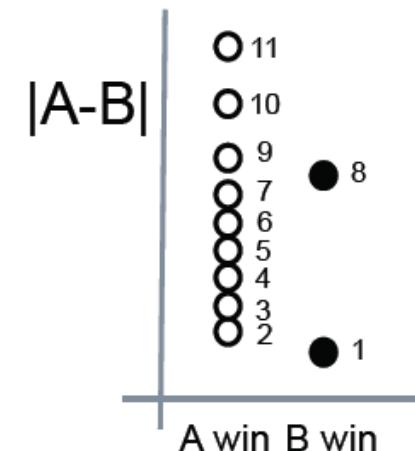
- Wilcoxon signed rank test

For paired data -algorithms A and B:

- Sort result by absolute size of differences
- If no systematic difference then average ranks of A wins and B wins will be roughly the same



$$W_A=37, W_B=29$$



$$W_A=57, W_B=9$$

For 11 datasets, critical value at 95% level is 10 (two-sided test)  
 $W_{\min}$  should be less than 10

# Le test de rang signé de Wilcoxon

<i>Data set</i>	<i>NB-DT</i>	<i>Rank</i>
1	-0.0715	<b>4</b>
2	-0.1947	<b>8</b>
3	0.0209	<b>1</b>
4	-0.2189	<b>9</b>
5	-0.1424	<b>6</b>
6	-0.1739	<b>7</b>
7	0.0992	<b>5</b>
8	-0.0371	<b>3</b>
9	-0.2802	<b>10</b>
10	0.0341	<b>2</b>

Est-ce que l'un des deux algorithmes est significativement meilleur que l'autre ?

From [Peter Flach (2012) Machine Learning. *The art and science of algorithms that make sense of data*. Cambridge University Press]

# Le test de rang signé de Wilcoxon

n	Two-Tailed Test		One-Tailed Test	
	$\alpha = .05$	$\alpha = .01$	$\alpha = .05$	$\alpha = .01$
5	--	--	0	--
6	0	--	2	--
7	2	--	3	0
8	3	0	5	1
9	5	1	8	3
10	8	3	10	5
11	10	5	13	7
12	13	7	17	9
13	17	9	21	12
14	21	12	25	15
15	25	15	30	19
16	29	19	35	23
17	34	23	41	27
18	40	27	47	32
19	46	32	53	37
20	52	37	60	43

[http://www.bios.unc.edu/~mhudgens/bios/662/2008fall/wilcox\\_t.pdf](http://www.bios.unc.edu/~mhudgens/bios/662/2008fall/wilcox_t.pdf)

Comparaison de **N** algorithmes  
sur **plusieurs** jeux de données

# Le test de Friedman

- Chaque algorithme est testé sur chaque jeu de données (par CV ou par jeu de test (« hold-out »))
- Les algorithmes sont alors triés pour chaque jeu de données
- Les algorithmes de performances équivalentes devraient avoir des **rangs moyens proches**
- Soient  $k$  algorithmes et  $N$  jeux de données et  $r_i$  le rang moyen de l'algorithme  $A_i$
- La moyenne et la variance de  $r_i$  sont alors :  $\frac{k+1}{2}$  et  $\frac{k^2 - 1}{12N}$

Data set	Algorithm A	Algorithm B	Algorithm C
$D_1$	1	2	3
$D_2$	1	2.5	2.5
$D_3$	1	2	3
$D_4$	1	2	3
Average rank	1	2.125	2.875

Exemple pour 3 algorithmes et 4 jeux de données

## Le test de Friedman (2)

---

- La variable

$$\begin{aligned}\tau_{\chi^2} &= \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \sum_{i=1}^k \left( r_i - \frac{k+1}{2} \right)^2 \\ &= \frac{12N}{k(k+1)} \left( \sum_{i=1}^k r_i^2 - \frac{k(k+1)^2}{4} \right)\end{aligned}$$

- Suit une loi du chi2 à  $k-1$  degrés de liberté quand  $k$  et  $N$  sont grands
- On utilise souvent la variable suivante qui est valable pour de plus petites valeurs de  $k$  et  $N$
- Et on teste si elle est plus petite qu'une valeur critique donnée par les tables

$$\tau_F = \frac{(N-1)\tau_{\chi}^2}{N(k-1) - \tau_{\chi}^2}$$

## Le test de Friedman (3)

---

- Table de valeurs critiques pour  $\alpha = 0.05$

N	k								
	2	3	4	5	6	7	8	9	10
4	10.128	5.143	3.863	3.259	2.901	2.661	2.488	2.355	2.250
5	7.709	4.459	3.490	3.007	2.711	2.508	2.359	2.244	2.153
8	5.591	3.739	3.072	2.714	2.485	2.324	2.203	2.109	2.032
10	5.117	3.555	2.960	2.634	2.422	2.272	2.159	2.070	1.998
15	4.600	3.340	2.827	2.537	2.346	2.209	2.104	2.022	1.955
20	4.381	3.245	2.766	2.492	2.310	2.179	2.079	2.000	1.935

## Le test de Friedman (4)

---

- Table de valeurs critiques pour  $\alpha = 0.1$

N	$k$								
	2	3	4	5	6	7	8	9	10
4	5.538	3.463	2.813	2.480	2.273	2.130	2.023	1.940	1.874
5	4.545	3.113	2.606	2.333	2.158	2.035	1.943	1.870	1.811
8	3.589	2.726	2.365	2.157	2.019	1.919	1.843	1.782	1.733
10	3.360	2.624	2.299	2.108	1.980	1.886	1.814	1.757	1.710
15	3.102	2.503	2.219	2.048	1.931	1.845	1.779	1.726	1.682
20	2.990	2.448	2.182	2.020	1.909	1.826	1.762	1.711	1.668

# Le test de rang signé de Wilcoxon

- Wilcoxon signed rank test

Table 6.4. *Classifiers NB and SVM data ran on 10 realistic domains and used in the Wilcoxon test example*

Domain no.	NB accuracy	SVM accuracy	NB–SVM	NB–SVM	Ranks ( NB–SVM )	± Ranks ( NB–SVM )
1	0.9643	0.9944	-0.0301	0.0301	3	-3
2	0.7342	0.8134	-0.0792	0.0792	6	-6
3	0.7230	0.9151	-0.1921	0.1921	8	-8
4	0.7170	0.6616	+0.0554	0.0554	5	+5
5	0.7167	0.7167	0	0	Remove	Remove
6	0.7436	0.7708	-0.0272	0.0272	2	-2
7	0.7063	0.6221	+0.0842	0.0842	7	+7
8	0.8321	0.8063	+0.0258	0.0258	1	+1
9	0.9822	0.9358	+0.0464	0.0464	4	+4
10	0.6962	0.9990	-0.3028	0.3028	9	-9

From [Japowicz & Shah (2011) “Evaluating Learning Algorithms. A classification perspective”]

- 
- Situation du **test d'hypothèses multiples**
    - Problème : un algorithme **peut sembler meilleur** par le hasard des tests (cf. lien entre risque empirique et risque réel)
  - **Post hoc** tests
    - **Une fois que** des tests statistiques ont montré une différence entre algorithmes, les **tests post hoc** cherchent quels algorithmes sont effectivement différents
    - Le **test de Nemenyi** est un test post-hoc très employé

# Le test de Nemenyi

---

- Calcule une  **$q$  statistique** sur la différence entre les rangs moyens

- Let  $R_{ij}$  be the rank of classifier  $f_j$  on dataset  $S_i$ .
- We compute the mean rank of classifier  $f_j$  on all datasets:

$$\bar{R}_{\cdot j} = \frac{1}{n} \sum_{i=1}^n R_{ij}.$$

- For any two classifiers  $f_{j1}$  and  $f_{j2}$ , we compute the  $q$  statistic as

$$q = \frac{\bar{R}_{\cdot j_1} - \bar{R}_{\cdot j_2}}{\sqrt{\frac{k(k+1)}{6n}}}.$$

- The null hypothesis is rejected after a comparison of the obtained  $q$  value with the  $q$  value for the desired significance table for critical  $q_\alpha$  values, where  $\alpha$  refers to the significance level.<sup>25</sup> Reject the null hypothesis if the obtained  $q$  value exceeds  $q_\alpha$ .

# Illustration

## Exemple pour trois algorithmes $f_A$ , $f_B$ et $f_C$

Table 6.6. *Sample accuracy results for classifiers  $f_A$ ,  $f_B$ , and  $f_C$  on 10 domains*

Domain	Classifier $f_A$	Classifier $f_B$	Classifier $f_C$
1	85.83	75.86	84.19
2	85.91	73.18	85.90
3	86.12	69.08	83.83
4	85.82	74.05	85.11
5	86.28	74.71	86.38
6	86.42	65.90	81.20
7	85.91	76.25	86.38
8	86.10	75.10	86.75
9	85.95	70.50	88.03
10	86.12	73.95	87.18

Table 6.9. *Rewriting Table 6.6 as ranks*

Domain	Classifier $f_A$	Classifier $f_B$	Classifier $f_C$
1	1	3	2
2	1.5	3	1.5
3	1	3	2
4	1	3	2
5	2	3	1
6	1	3	2
7	2	3	1
8	2	3	1
9	2	3	1
10	2	3	1
Rank Sums ( $R_{.j}$ )	15.5	30	14.5

# Illustration

---

## Exemple pour trois algorithmes $f_A$ , $f_B$ et $f_C$

Table 6.9. Rewriting Table 6.6 as ranks

Domain	Classifier $f_A$	Classifier $f_B$	Classifier $f_C$
1	1	3	2
2	1.5	3	1.5
3	1	3	2
4	1	3	2
5	2	3	1
6	1	3	2
7	2	3	1
8	2	3	1
9	2	3	1
10	2	3	1
Rank Sums ( $R_{.j}$ )	15.5	30	14.5

$$q_{12} = \frac{\bar{R}_{.1} - \bar{R}_{.2}}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{15.5 - 30}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{-14.5}{0.45} = -32.22,$$

$$q_{13} = \frac{\bar{R}_{.1} - \bar{R}_{.3}}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{15.5 - 14.5}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{1}{0.45} = 2.22,$$

$$q_{23} = \frac{\bar{R}_{.2} - \bar{R}_{.3}}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{30 - 14.5}{\sqrt{\frac{3(3+1)}{6 \times 10}}} = \frac{15.5}{0.45} = 34.44.$$

Ici CD = 2,55 (Critical Difference)

Donc  $f_B > f_A$ , et  $f_B > f_C$ , mais pas  $f_A > f_C$

# Illustration

- Reprenons la comparaison :
- Table de valeurs critiques

$\alpha$	$k$								
	2	3	4	5	6	7	8	9	10
0.05	1.960	2.344	2.569	2.728	2.850	2.949	3.031	3.102	3.164
0.10	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

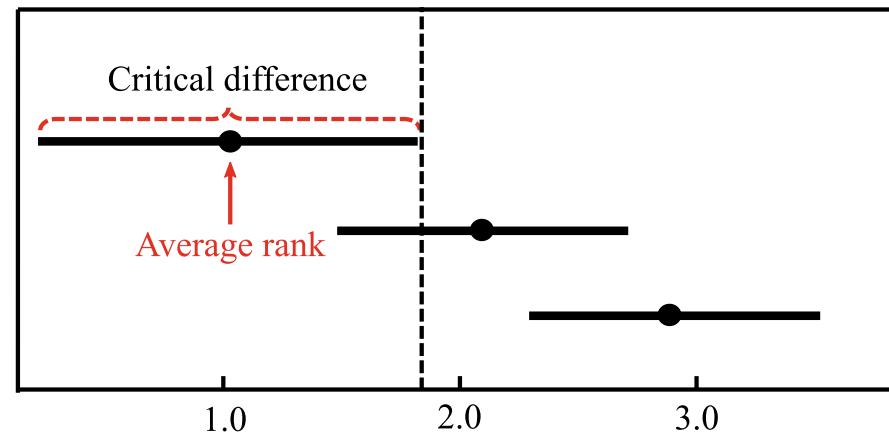
- On a alors :

Ici, A est significativement meilleur que C

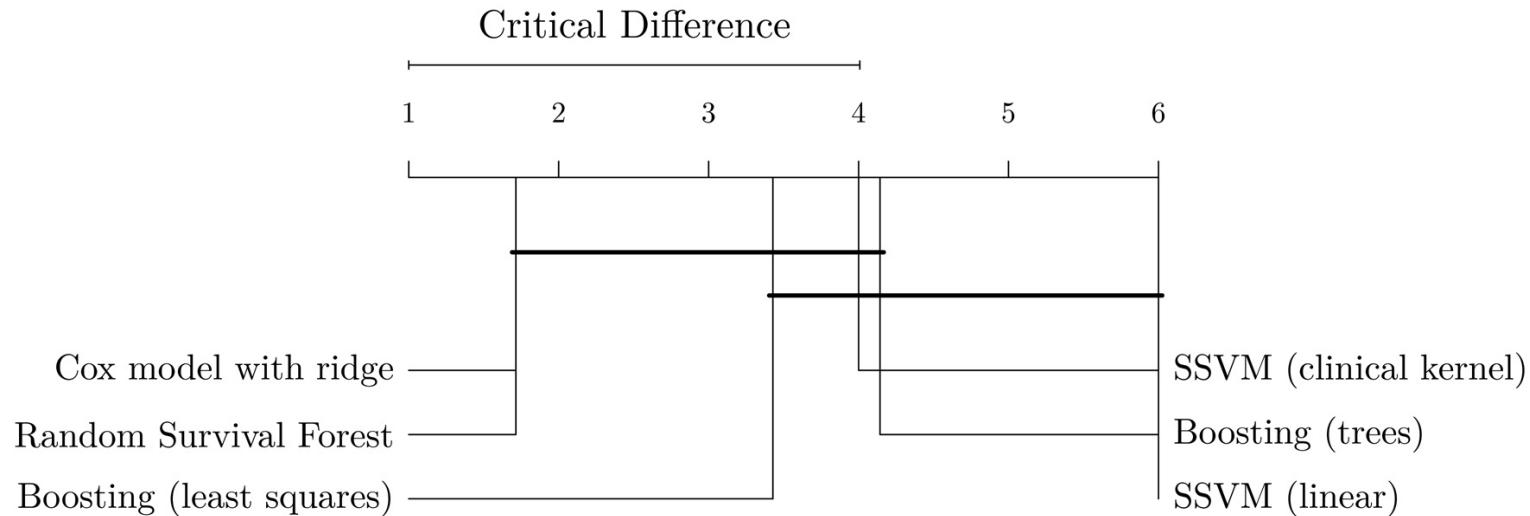
Data set	Algorithm A	Algorithm B	Algorithm C
$D_1$	1	2	3
$D_2$	1	2.5	2.5
$D_3$	1	2	3
$D_4$	1	2	3
Average rank	1	2.125	2.875

Exemple pour 3 algorithmes et 4 jeux de données

Algorithm A  
Algorithm B  
Algorithm C



# Illustration : test de Nemenyi

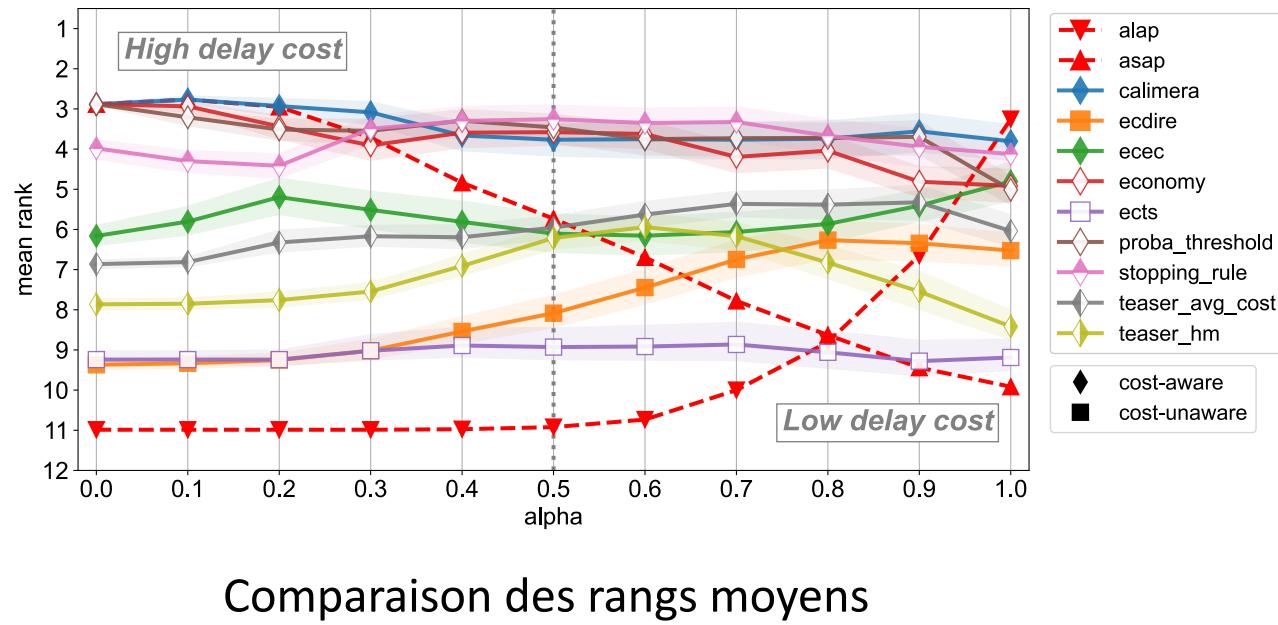


Les méthodes sont **triées par rang moyen** (de la gauche vers la droite) et les groupes de méthodes non significativement différents sont **connectés** ( $p\text{-value} > 0.05$ )

[Sebastian Pölsterl et al. (2017) "Heterogeneous ensembles for predicting survival of metastatic, castrate-resistant prostate cancer patients", F1000Research.]

# Illustration

- Comparaison de 11 algorithmes sur ~30 jeux de données

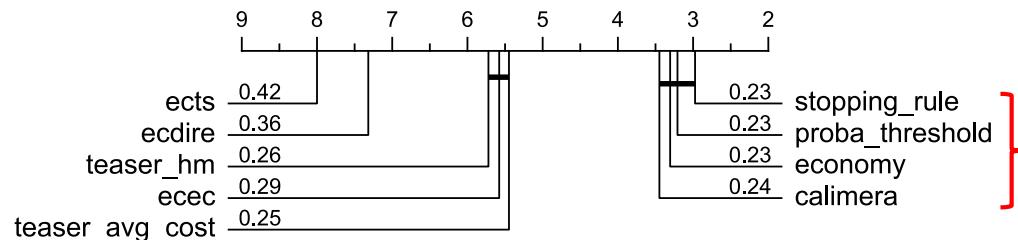
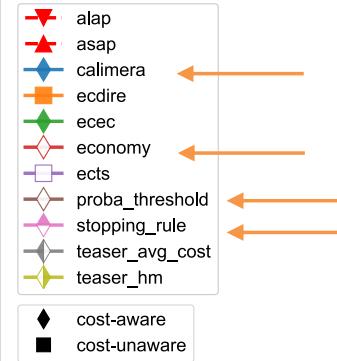
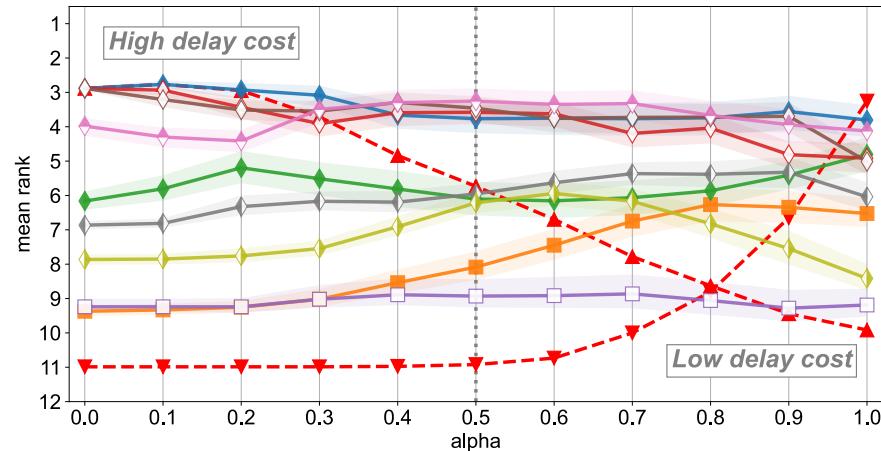


ATTENTION : ne dit rien sur les performances

Peut-être qu'il y a très peu de différences !

# Illustration

- Comparaison de 11 algorithmes sur ~30 jeux de données



Comparaison des rangs moyens

Pour  $\alpha = 0.5$ , on voit 4 algorithmes se détacher

## Références

---

- Nathalie Japkowicz & Mohak Shah (2011) [Evaluating Learning Algorithms. A classification perspective](#). Cambridge University Press.
- Demsar, J. (2006) [Statistical Comparison of Classifiers over Multiple Data Sets](#). *Journal of Machine Learning Research* 7:1-30.