MLEA - Rapport de TP 2

Florent D'Halluin Hervé Cuche

14 Novembre 2009

Quickstart

Parcours de correction rapide et efficace:

- `make` dans un term libre (Compilation des tests et géneration des images en arrière-plan — peut prendre 2 heures).
- Lecture du rapport.
- *Optionnel: Ctrl-c le make (les images les plus significatives sont générées en premier dans* `classification/images`*).*
- Jeter un oeil aux images générées, tester quelques commandes.
- Noter.

Note: L'architecture des fichiers rendus a été reprise des TPs de MLEA1 (Florent D'Halluin). Elle comprend tout le travail effectué jusqu'ici.

1 ID3

`cd classification`. `make` (génère les images pour toutes les sous-sections).

1.1 Notes sur l'implémentation

Test ID3:

```
make <Dataset>
python id3.py <Dataset>_data [<Discretization>] [<Gain>] [--prune] --test
dot -Tpdf <Dataset>_data_graph.dot -o <Dataset>_data_id3_graph.pdf
display <Dataset>_data_id3_graph.pdf
```

Classification:

```
make <Dataset>
python classification.py <Dataset>_data <Positive label> <Data Size>
```

Note: Étant donné le grand nombre de tests possibles, choisir les tests désirés en modifiant classification.py (fin du fichier).

Datasets:

```
donut
donut_simple
xor
xor_simple
linear
linear_simple
tennis
glass
iris
optdigits_tes
optdigits_tra
krkopt
```

Discretization:

```
--efd
--ewd
```

Gain:

```
--gain
--gainratio
--gini
```

Fichiers:

- `<Dataset>_data`: Dataset.
- `<Dataset>.py`: Extraction du dataset.
- `id3.py`: Implémentation de ID3.
- `kfcv.py`: K-Fold Cross-Validation.
- `classification.py`: Tests et génération des courbes.
- `discretization.py`: Méthodes de discrétisation.

L'implémentation est basée sur les notes de cours. Pour chaque noeud de l'arbre, on conserve les probabilités à priori des éléments qui se retrouvent dans les fils, ce qui permet d'assigner un label par défaut aux éléments pour lesquels on n'arrive pas à descendre jusqu'à une feuille.

Lors de l'apprentissage, la profondeur de l'arbre n'augmente plus lorsque l'entropie atteint 0 pour un noeud donné, ni lorsque tous les éléments sont identiques (même si l'entropie n'est pas nulle). Ces deux limites sont adoptées par tous les algorithmes qu'on a pu voir dans la littérature.

1.1.1 PlayTennis

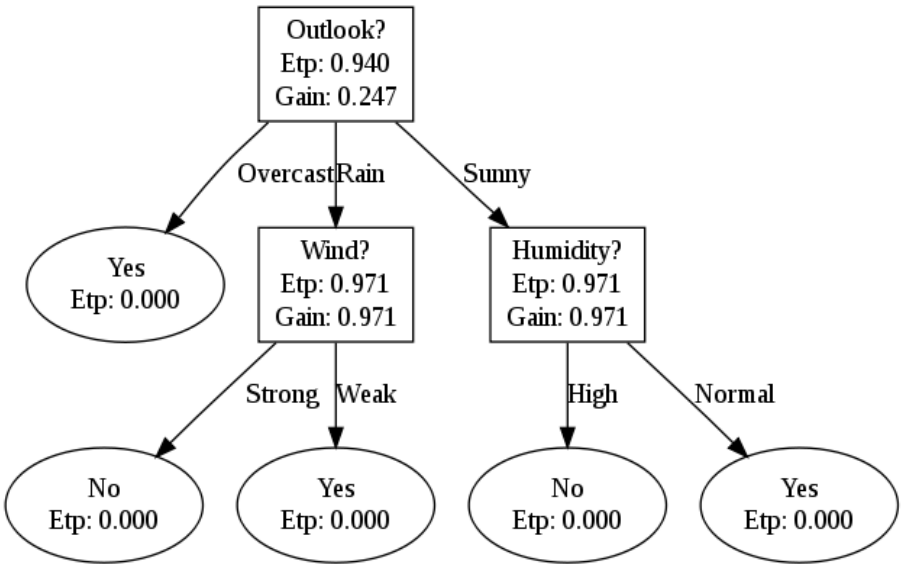

Figure 1: Arbre de décision pour le dataset PlayTennis (ID3).

Figure 1 montre l'arbre obtenu pour le dataset PlayTennis. Le dataset étant simple, l'arbre de décision a peu de noeuds.

1.1.2 Mushroom

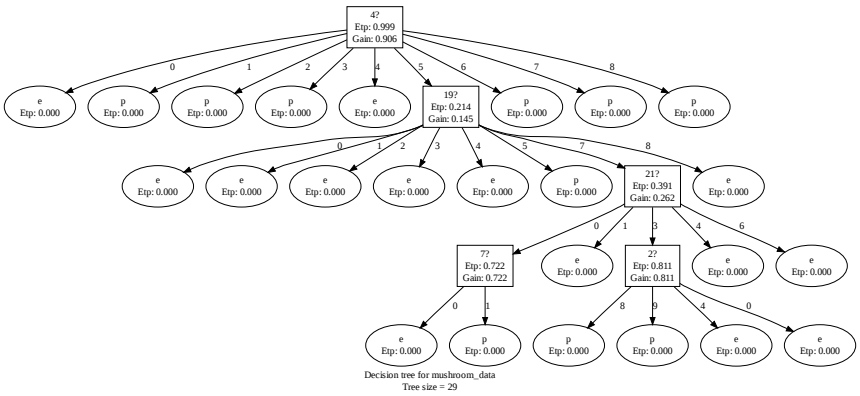

Figure 2: Arbre de décision pour le dataset Mushroom (ID3).

Figure 2 montre l'arbre obtenu pour le dataset Mushroom. L'arbre induit reste simple (29 noeuds), alors que le dataset a une taille importante (8000 points). De plus, toutes les feuilles ont une entropie de 0, ce qui suggère que l'arbre aura de bonnes performances (par exemple sur une K-Fold cross validation). Figure 7 et Figure 8 illustrent ce dernier point: l'arbre de décision obtenu pour id3 a des performances similaires à KNN (en pratique, l'algorithme est également plus rapide).

1.2 Améliorations

1.2.1 Sélection de l'attribut à tester

Les deux expressions alternatives du gain ont été implémentées (GainRatio et impureté de Gini). Figure 3 et Figure 4 sont à comparer à Figure 1 (dataset Tennis). Figure 5 et Figure 6 sont à comparer à Figure 2 (dataset Mushroom). On observe des arbres de tailles similaires pour les méthodes Gain et GainRatio, alors que la méthode Gini produit des arbres avec un nombre de noeuds plus important Table 1. En outre, les taux de reconnaissance sont meilleurs avec les méthodes Gain et GainRatio qu'avec Gini (Figure 7 sur une partie du dataset Mushroom et Figure 8 sur l'ensemble du dataset).

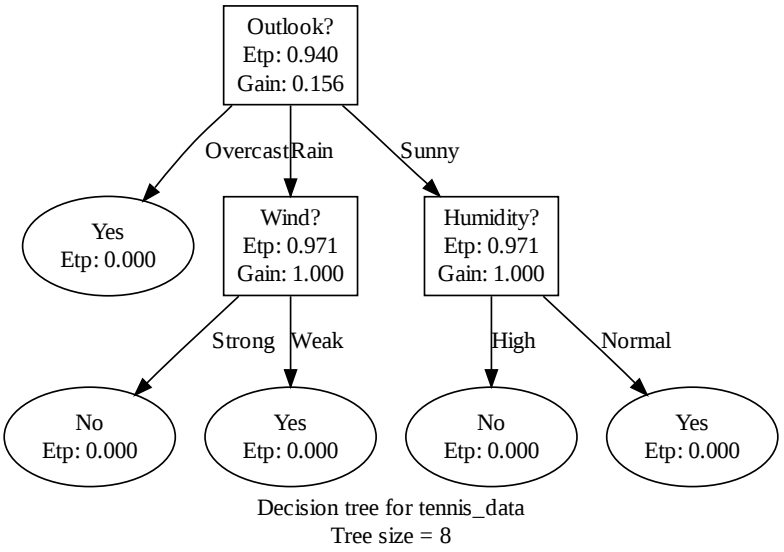

Figure 3: Arbre de décision pour le dataset Tennis (ID3, GainRatio).

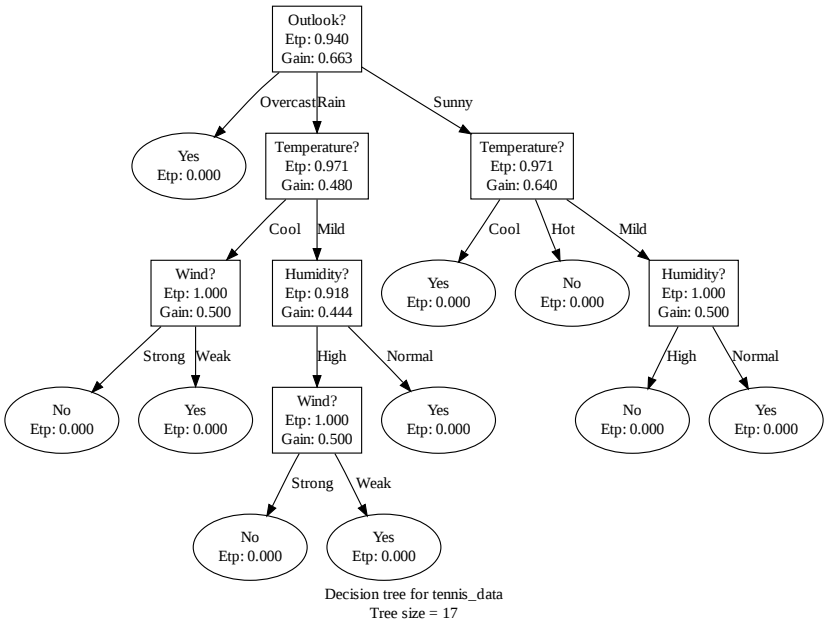

Figure 4: Arbre de décision pour le dataset Tennis (ID3, Gini).

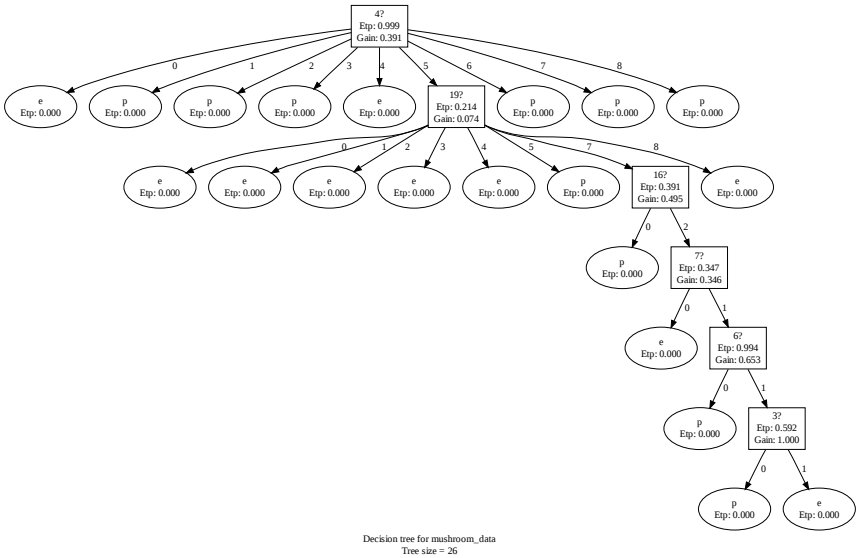

Figure 5: Arbre de décision pour le dataset Mushroom (ID3, GainRatio).

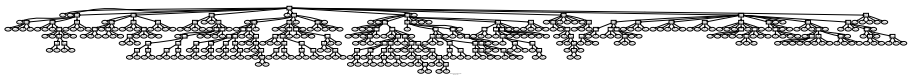

Figure 6: Arbre de décision pour le dataset Mushroom (ID3, Gini).

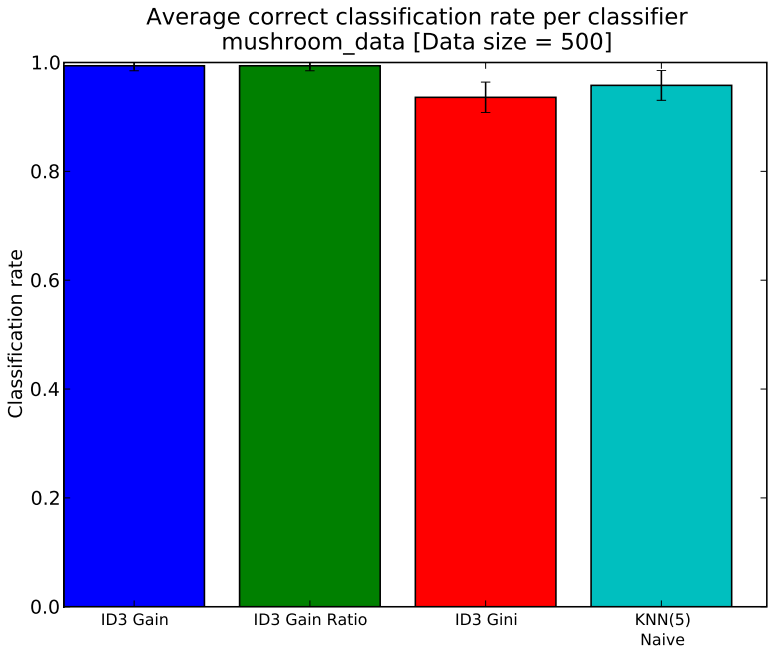

Figure 7: Taux de reconnaissance sur K-Fold validation (ID3, Extrait du dataset Mushroom).

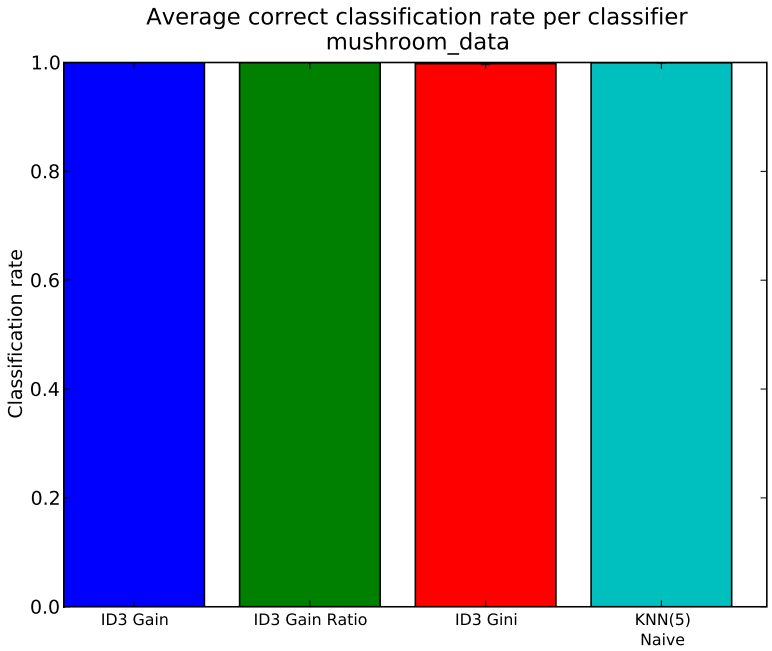

Figure 8: Taux de reconnaissance sur K-Fold validation (ID3, dataset Mushroom).

1.2.2 Gestion des attributs continus

Pour gérer les attributs continus, les méthodes EFD et EWD ont été implémentées. En pratique, elles ont des résultats similaires Figure 10. Les tests ont étés effectués sur le dataset Donut (1000 points, 2 attributs, 2 classes, 10% de bruit), ainsi que sur le dataset Glass (214 points, 6 attributs, 7 classes) Figure 9 illustre le dataset Glass à travers l'arbre induit par ID3 sur le dataset discrétisé par EFD.

Figure 11 et Figure 12 illustrent l'utilité de la discrétisation sur des datasets à valeurs continues: l'arbre induit par ID3 n'est efficace que lorsque les données sont discrétisées et atteint la même précision que KNN. On note également que KNN est moins efficaces sur les données discrétisées que sur les données continues, ce qui n'est pas surprenant.

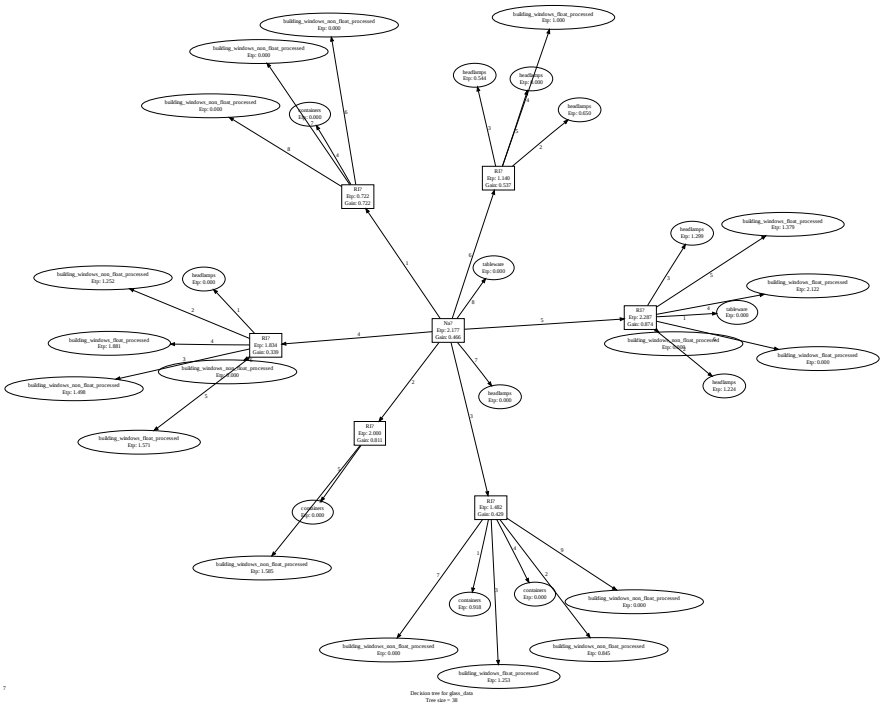

Figure 9: Arbre de décision pour le dataset Glass (ID3, discrétisation EFD).

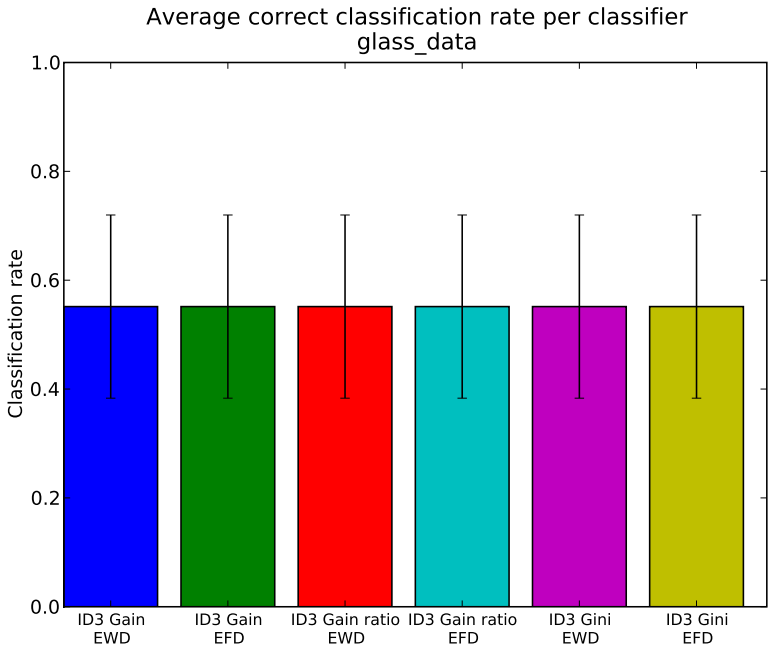

Figure 10: Comparaison des méthodes de discrétidsation (K-Fold validation, ID3, dataset Glass + discrétisation).

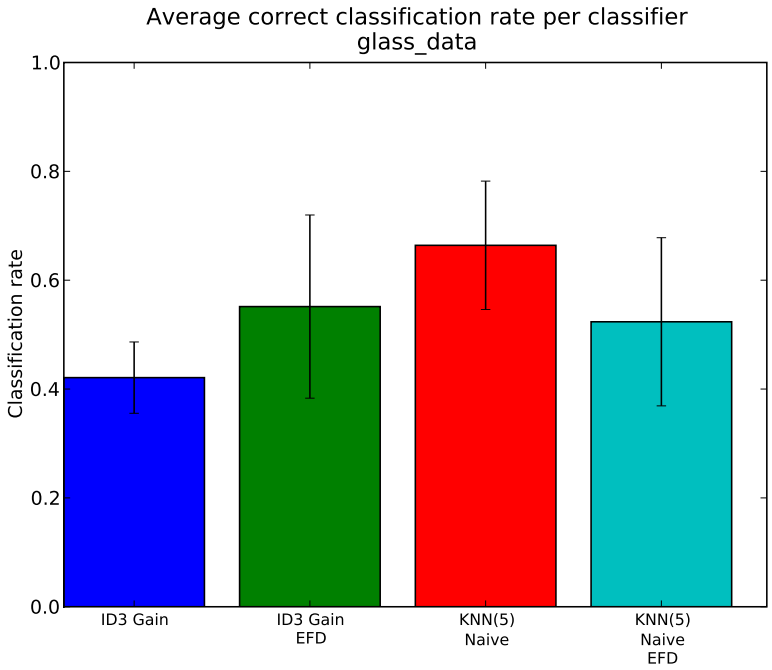

Figure 11: Taux de reconnaissance sur K-Fold validation (ID3, dataset Glass + discrétisation).

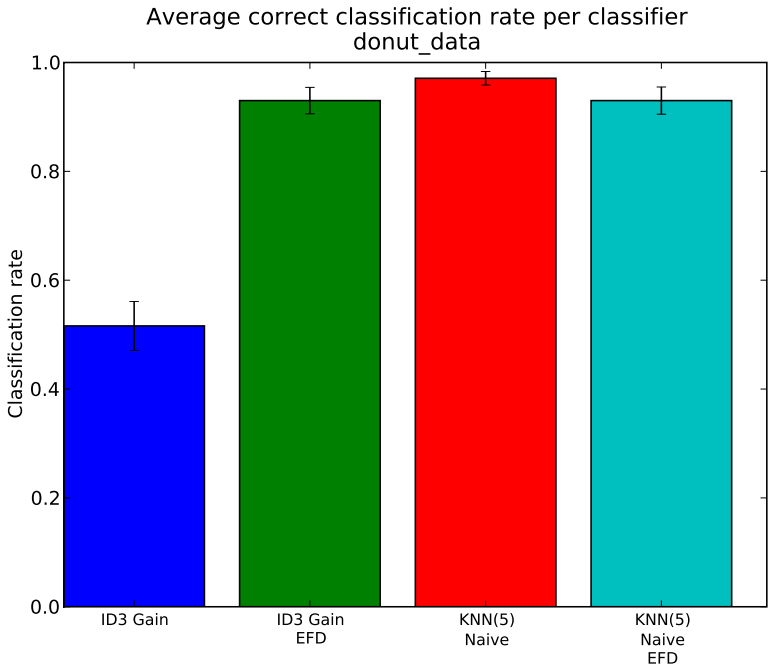

Figure 12: Taux de reconnaissance sur K-Fold validation (ID3, dataset Donut + discrétisation).

1.2.3 Élagage

La méthode d'élagage REP a été implémentée depuis le papier listé sur le sujet et quelques recherches dans la littérature. Lors de l'apprentissage, 25% des données sont mises à part pour l'élagage (pruning set). L'élagage se fait par itérations successives afin d'obtenir l'arbre minimum qui a l'erreur minimale sur le pruning set. Figure 13 illustre le pruning sur un arbre de petite taille: l'entropie en feuille devient parfois non nulle (par rapport à l'arbre Figure 5) parce que le pruning set ne représente pas toutes les données d'apprentissage.

Table 1 donne la taille de l'arbre obtenu après élagage pour différents datasets. On observe un gain proportionnel au rapport de la taille de l'arbre sur la taille du dataset. Lorsque l'arbre est très petit par rapport au dataset (e.g. Mushroom avec Gain et GainRatio), l'effet de l'élagage est difficile à évaluer.

En pratique, le taux de reconnaissance d'un arbre est similaire qu'il soit élagué ou non (Figure 14 et Figure 15).

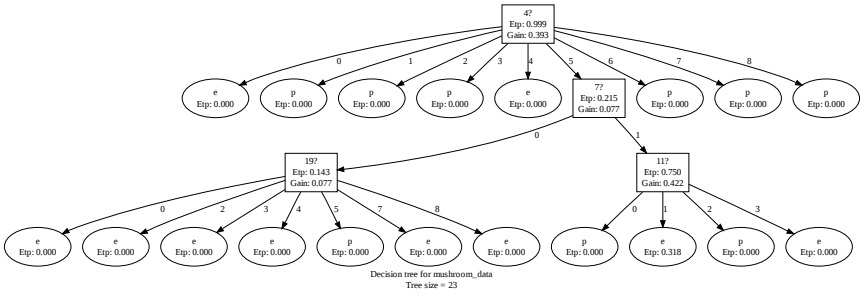

Figure 13: Arbre de décision élagué pour le dataset Mushroom (ID3, GainRatio, REP).

Dataset	Taille	Gain	Complet	REP	Taux d'élagage
Mushroom	8124	Gain	29	26	10%
Mushroom	8124	GainRatio	26	23	12%
Mushroom	8124	Gini	467	320	31%
Glass (EFD)	214	Gain	38	15	61%
Glass (EFD)	214	GainRatio	38	15	61%
Glass (EFD)	214	Gini	38	17	55%
Donut (EFD)	2000	Gain	145	130	10%
Donut (EFD)	2000	GainRatio	145	130	10%
Donut (EFD)	2000	Gini	145	130	10%

Table 1: Taille de l'arbre induit selon plusieurs variantes de ID3 (dataset Mushroom).

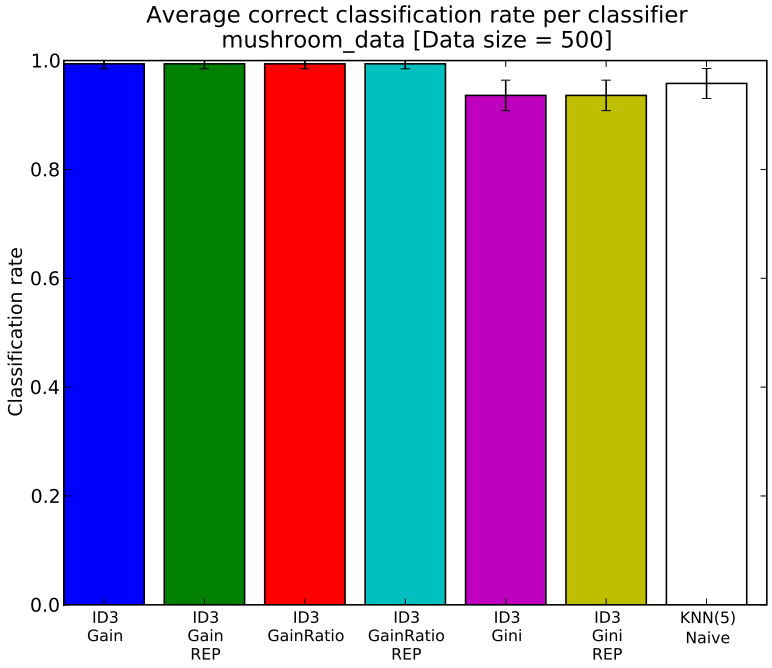

Figure 14: Effet de l'élagage sur le taux de reconnaissance (dataset Mushroom).

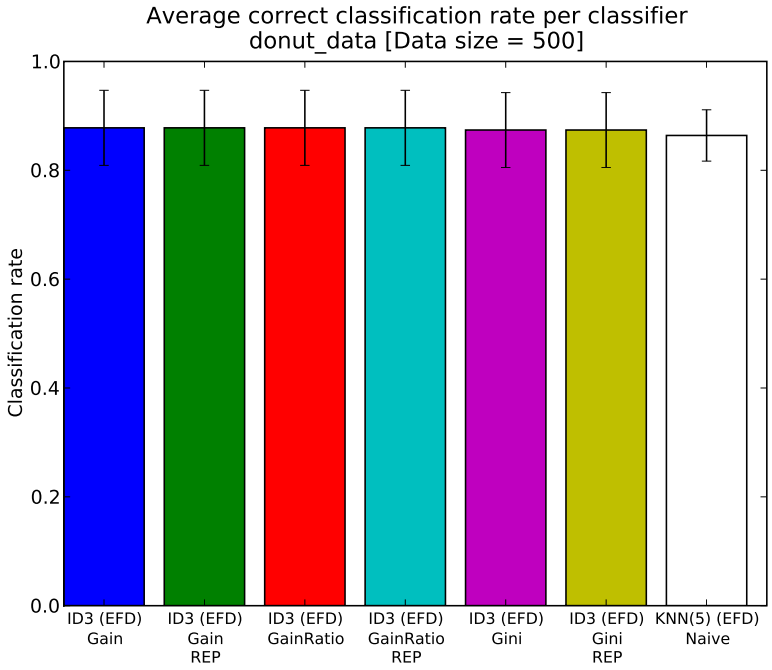

Figure 15: Effet de l'élagage sur le taux de reconnaissance (dataset Donut).

2 SVM

2.1 Notes sur l'implémentation

Test SVM:

```
make <Dataset>
python svm.py <Dataset>_data [<Kernel> <Kernel Parameter>] --test
```

Classification:

```
make <Dataset>
python classification.py <Dataset>_data <Positive label> <Data Size>
```

Note: Étant donné le grand nombre de tests possibles, choisir les tests désirés en modifiant classification.py (fin du fichier).

Datasets:

```
donut
donut_simple
linear
linear_simple
xor
xor_simple
tennis
glass
iris
optdigits_tes
optdigits_tra
krkopt
```

Kernel & Kernel parameter:

```
--linear_k
--poly_k <degré du polynome>
--rbf_k <gamma>
```

Fichiers:

- `<Dataset>_data`: Dataset.
- `<Dataset>.py`: Extraction du dataset.
- `svm.py`: Implémentation du SVM.
- `kfcv.py`: K-Fold Cross-Validation.

- `classification.py`: Tests et génération des courbes.
- `discretization.py`: Méthodes de discrétisation.

L'implémentation est basée sur les slides de cours. Les datasets de tests sont:

- Linear (linéairement séparable, bruit 10%): Figure 16.
- Linear simple (non linéairement séparable, bruit 0%): Figure 17.
- Donut (non linéairement séparable, bruit 10%): Figure 18.
- Donut simple (non linéairement séparable, bruit 0%): Figure 18.
- Xor (non linéairement séparable, bruit 10%): Figure 20.
- Xor simple (non linéairement séparable, bruit 0%): Figure 20.

2.2 Datasets

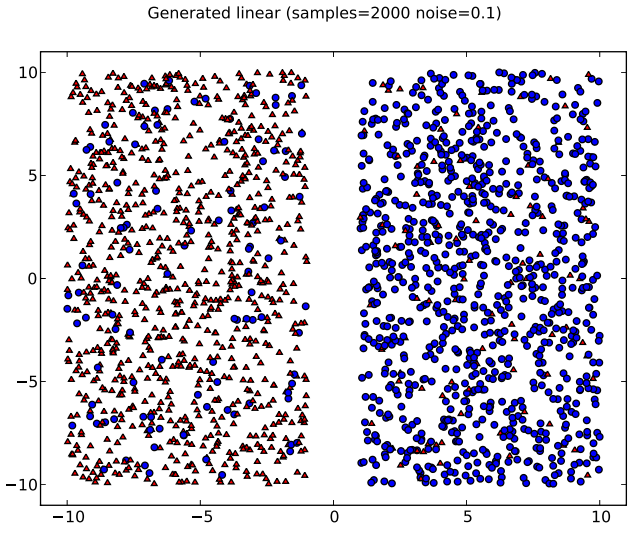

Figure 16: Dataset linear.

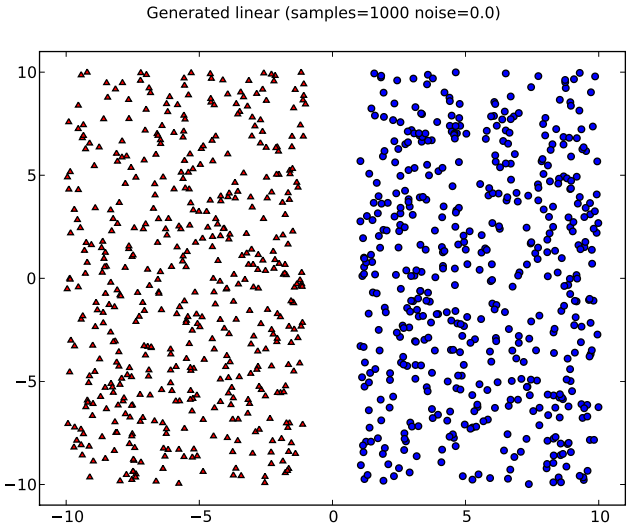

Figure 17: Dataset linear (simple).

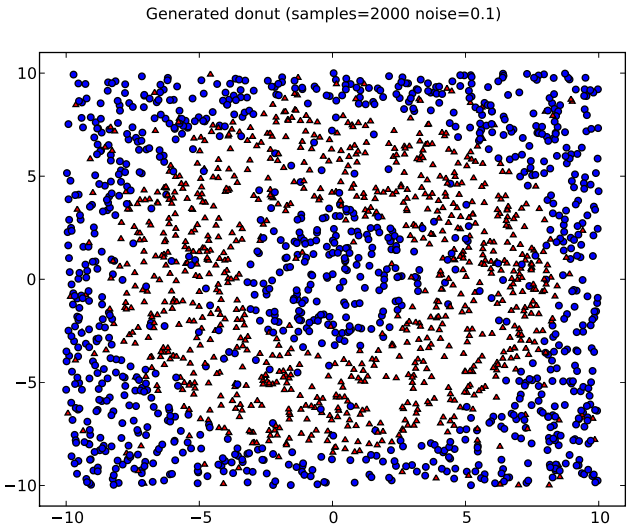

Figure 18: Dataset donut.

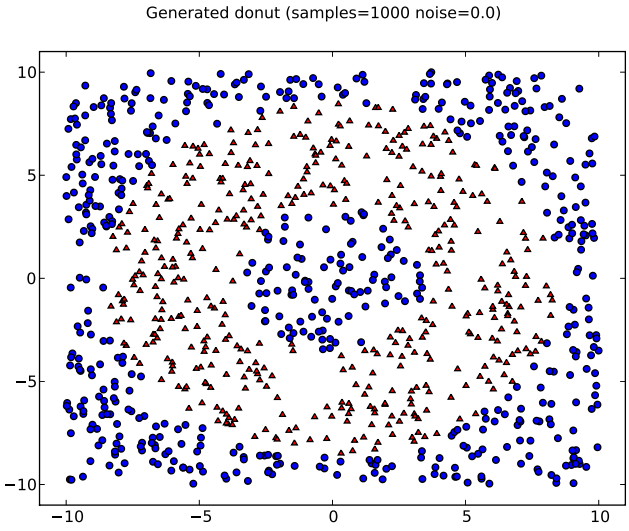

Figure 19: Dataset donut (simple).

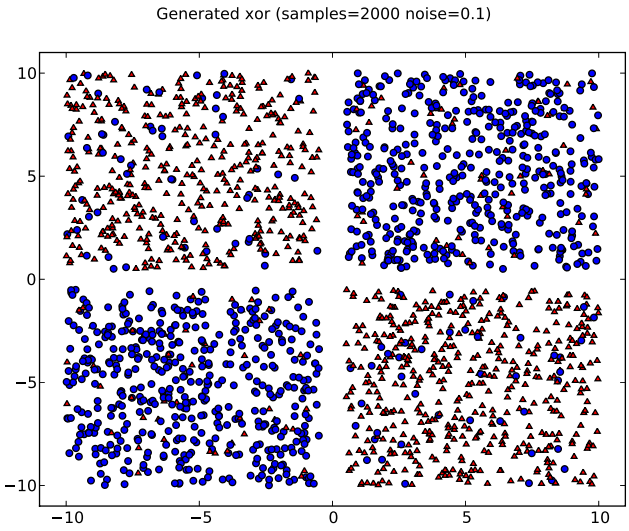

Figure 20: Dataset xor.

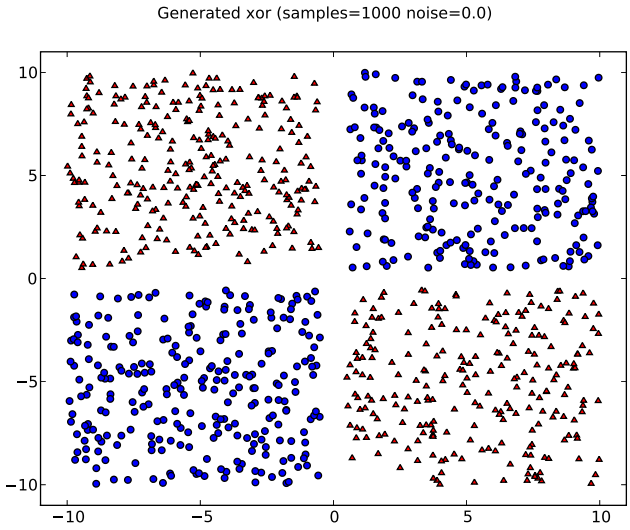

Figure 21: Dataset xor (simple).

2.3 Kernels

Le SVM linéaire ne fonctionne que sur les datasets linéairement séparables. Afin de pouvoir gérer les datasets non linéairement séparables, trois types de kernels ont été implémentés (linéaire, polynomial et RBF). Pour gérer les données bruitées, une version soft margin du svm a été implémentée.

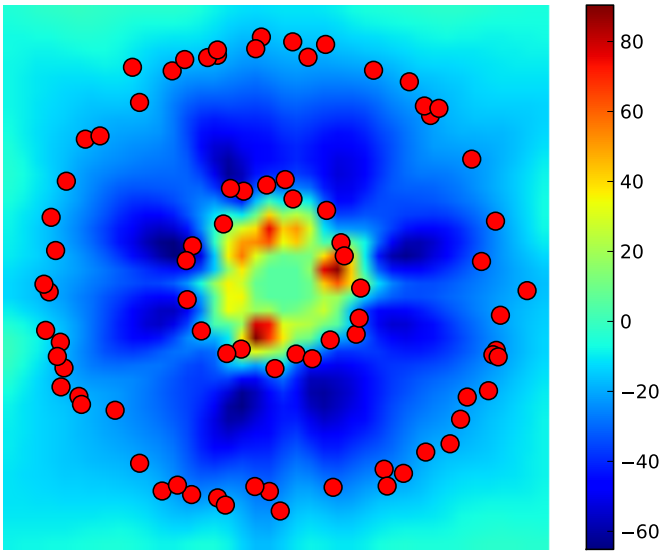

Figure 22: Fonction de décision d'un donut avec kernel polynomial p=32, s=80

2.4 Multiclass Management

Ce point n'a pas été implémenté.

2.5 Unbalanced Dataset

Ce point n'a pas été implémenté.

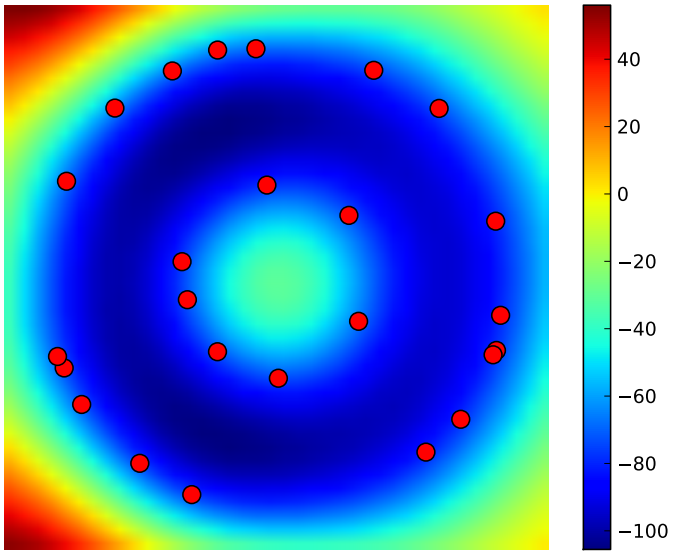

Figure 23: Fonction de décision d'un donut avec kernel rbf gamma=50, s=25

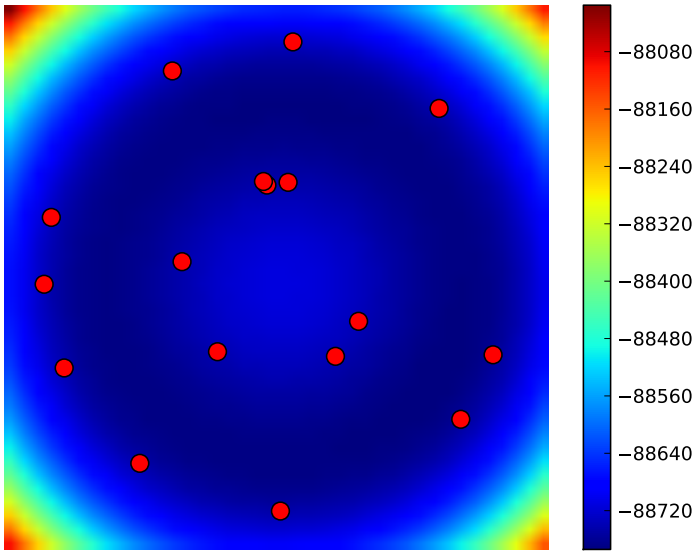

Figure 24: Fonction de décision d'un donut avec kernel rbf gamma=2000, s=17

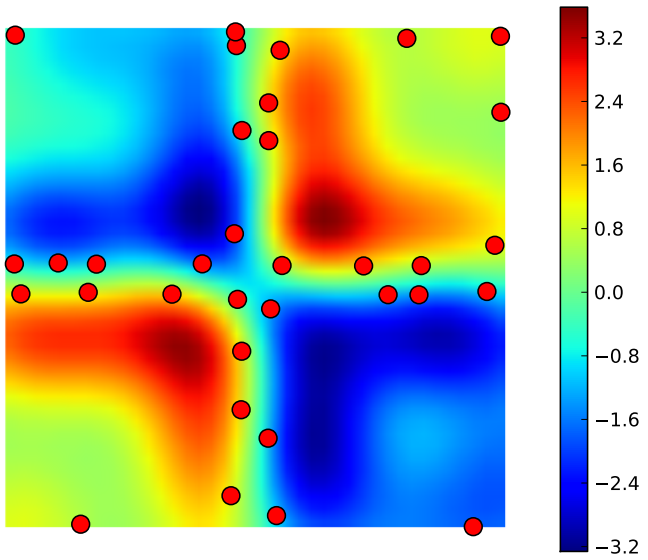

Figure 25: Fonction de décision d'un xor avec kernel rbf gamma=1, s=34

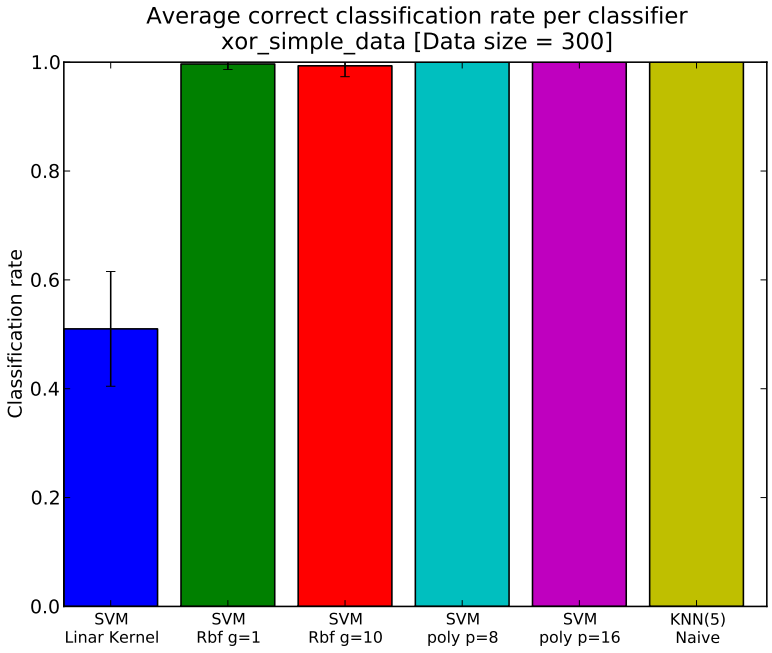

Figure 26: Kfold pour un dataset Xor non buité

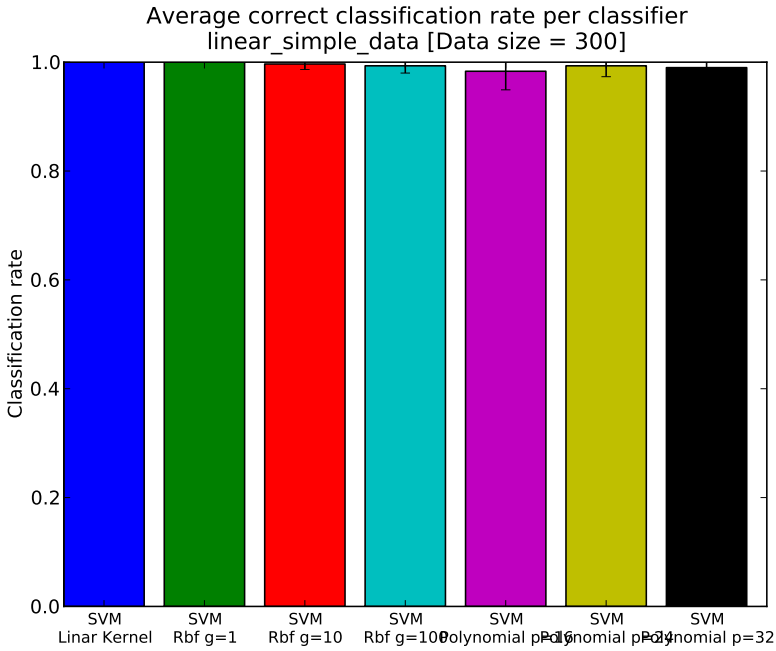

Figure 27: Kfold pour un dataset lineaire non buité

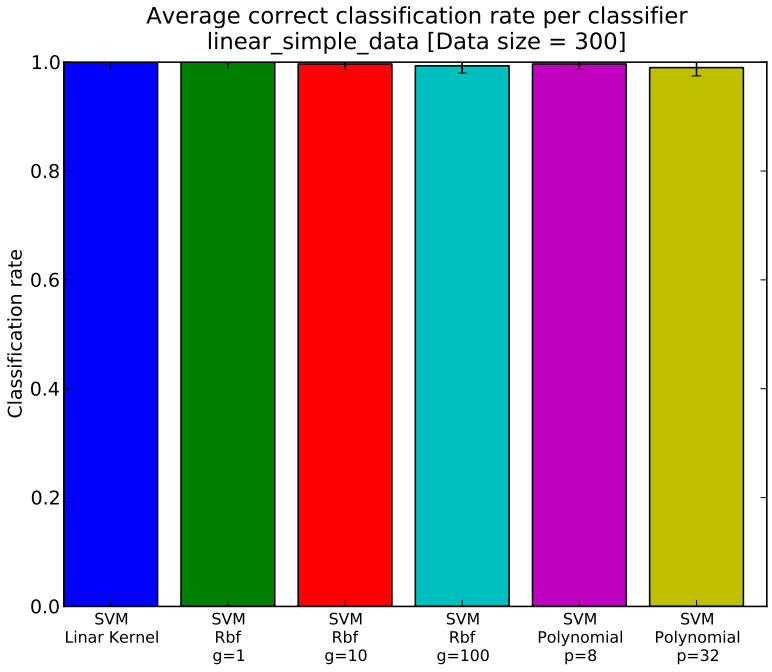

Figure 28: Kfold pour un dataset lineaire buité

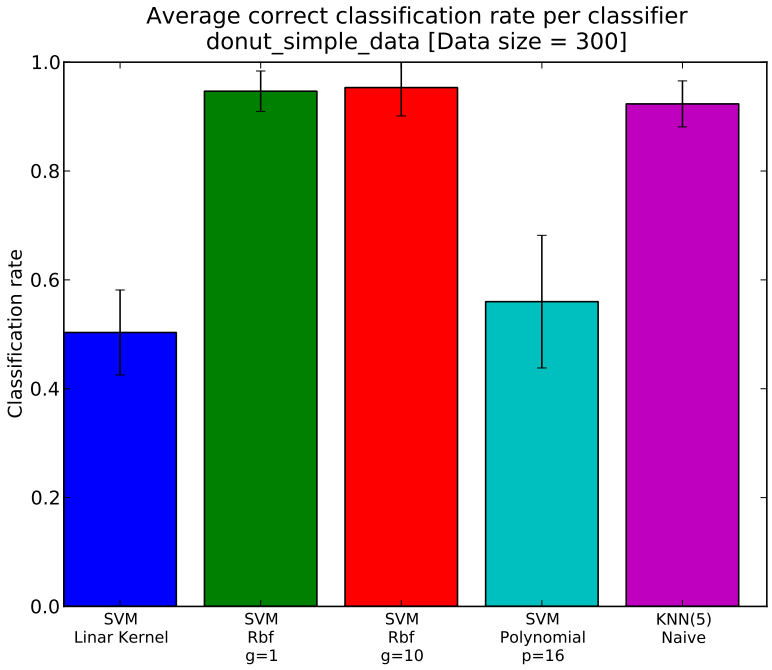

Figure 29: Kfold pour un dataset donut non buité

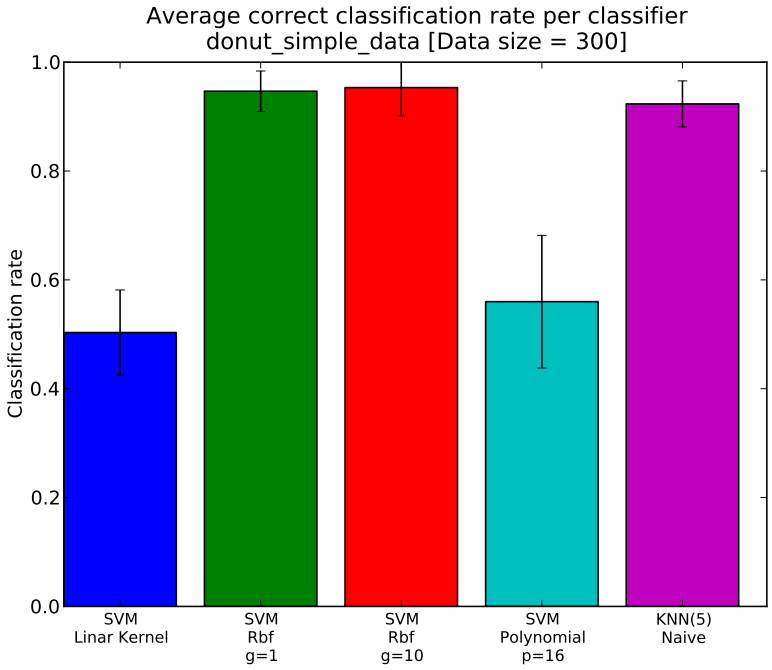

Figure 30: Kfold pour un dataset donut non buité

3 Final benchmarking

Le benchmarking final a été réalisé sur deux types de classifieurs et datasets:

- Les classifieurs et datasets à données continues (Figure 31, Figure 32).
- Les classifieurs et datasets à données discrètes (Figure 33, Figure 34).

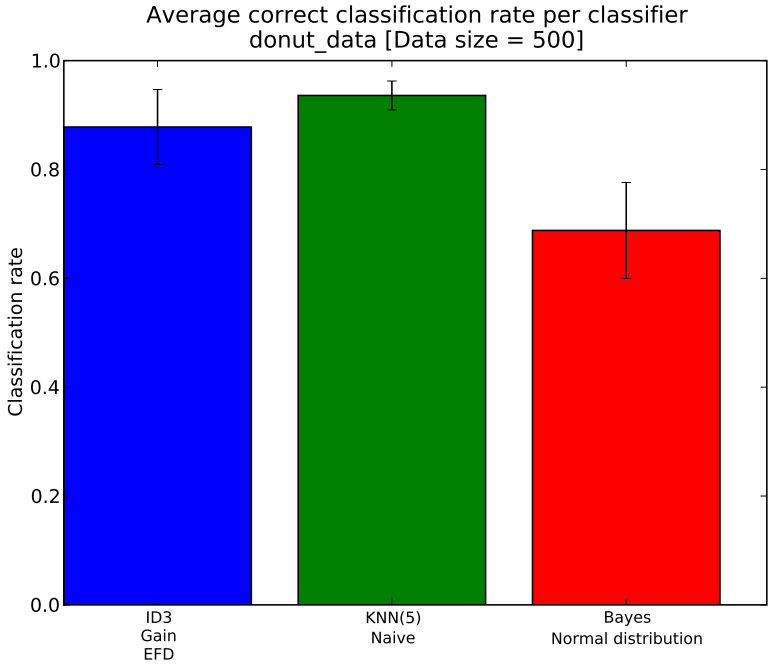

Figure 31: K-Fold Cross-validation (classifieurs sur données continues, dataset Donut).

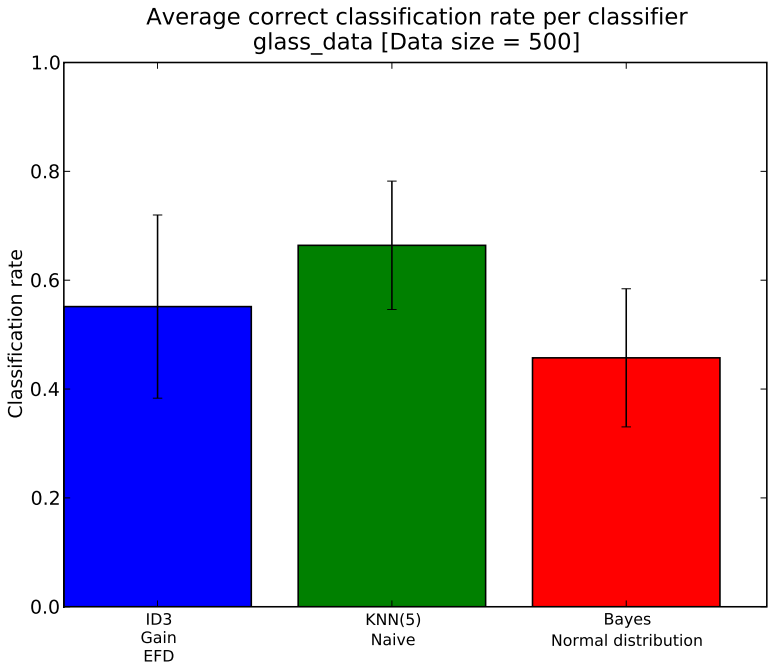

Figure 32: K-Fold Cross-validation (classifieurs sur données continues, dataset Glass).

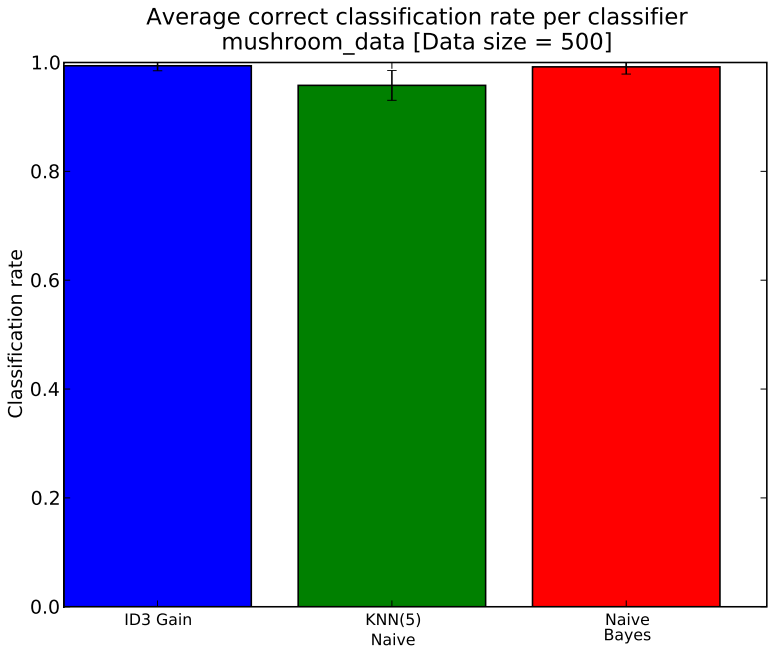

Figure 33: K-Fold Cross-validation (classifieurs sur données discrètes, dataset Mushroom).

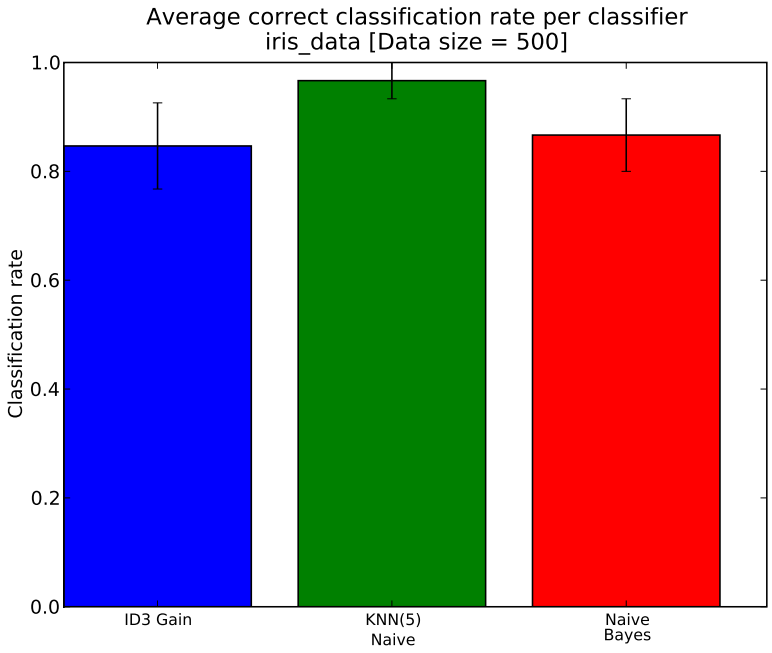

Figure 34: K-Fold Cross-validation (classifieurs sur données discrètes, dataset Iris).